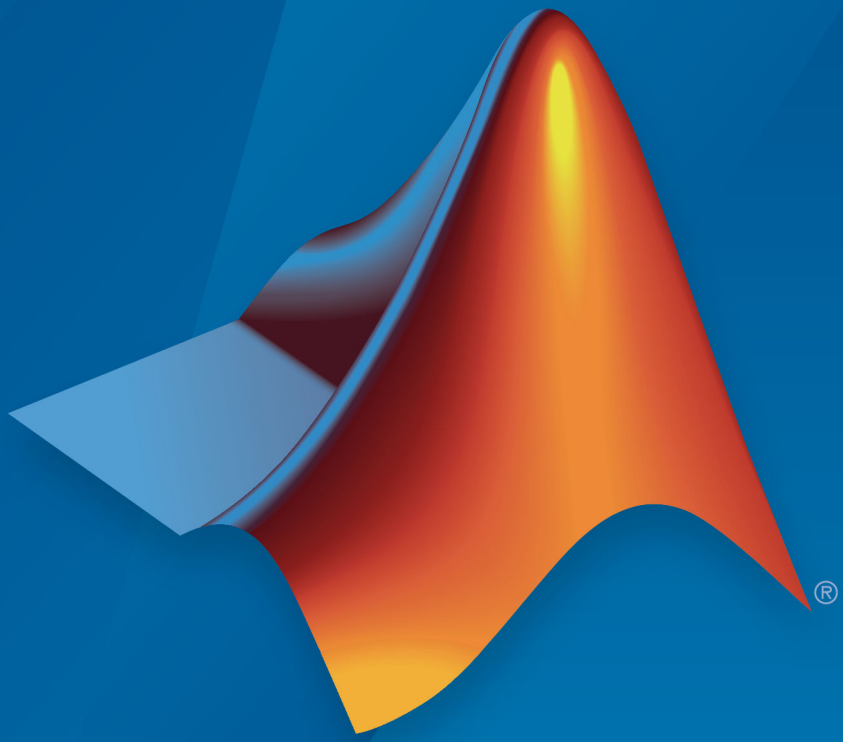
Statistics and Machine Learning Toolbox™ Release Notes



MATLAB®

MathWorks®

How to Contact MathWorks

Latest news: `www.mathworks.com`

Sales and services: `www.mathworks.com/sales_and_services`

User community: `www.mathworks.com/matlabcentral`

Technical support: `www.mathworks.com/support/contact_us`

Phone: 508-647-7000

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098

Statistics and Machine Learning Toolbox ™ Release Notes

Contents

R2018a

R2017b

R2017a

R2016b

R2015b

R2014b

R2014a

R2013b

R2013a

R2012b

R2012a

R2011b

R2011a

R2010b

R2010a

R2009b

R2009a

R2008b

R2008a

R2007b

R2007a

R2006b

R2006a

R14SP3

R14SP2

R2018a

Code Generation: Generate C code for distance calculation on vectors and matrices, and for prediction by using k-nearest neighbor with Kd-tree search and nontree ensemble models (requires MATLAB Coder)

The following functions support code generation:

- `grp2idx` — Create an index vector from a grouping variable.
- `pdist` — Find the pairwise distance between pairs of observations.
- `squareform` — Format a distance matrix.

When you train a *k*-nearest neighbor classification model by using `fitcknn` for code generation, you can now use the *K*d-tree search algorithm. For more details, see the "Code Generation" section of the `ClassificationKNN` class.

When you find nearest neighbors by using the functions `knnsearch` and `rangesearch` and the object functions `knnsearch` and `rangesearch` for code generation, you can now use the *K*d-tree search algorithm.

When you train an ensemble by using `fitcensemble` for code generation, you can now specify `'discriminant'` or `'knn'` as weak learners by using the `'Learners'` name-value pair argument. For more details, see the "Code Generation" section of the `CompactClassificationEnsemble` class and "Code Generation Workflow Using MATLAB Coder App".

Nonlinear Regression for Big Data: Fit kernel SVM regression models by using random feature expansion

For nonlinear regression for big data, use the `fitrkernel` function to train a Gaussian kernel regression model using feature expansion. `fitrkernel` creates an instance of the `RegressionKernel` object.

`RegressionKernel` is a new model object for accessing and performing operations on the training data, and is especially suited for using tall arrays. `RegressionKernel` is compact; that is, it does not store the training data. With some exceptions, the syntax and methods of the `RegressionKernel` model object resemble those for other regression model objects that store the training data. For example, to predict responses of a `RegressionKernel` model, pass the trained model object and new predictor data to

`predict`. To continue training, pass a trained model object and the training data to `resume`.

Big Data Algorithms: Compute confusion matrices and create nonstratified partitions for cross-validation on out-of-memory data

These functions support tall arrays:

- `confusionmat` displays confusion matrices.
- `cvpartition` supports nonstratified holdout cross-validation partitions with the `'Stratify'` name-value pair argument.

For a complete list of supported statistics functions, see “Tall Array Support, Usage Notes, and Limitations”.

Classification Learner App: Visualize and investigate high-density data with improved scatter plots

In the scatter plot of Classification Learner, you can now zoom in and out, pan across the plot, and change the stacking order of the plotted classes. This new functionality makes it easier to:

- Inspect and analyze your data.
- Determine the separation of classes for different input features.
- Inspect model results such as misclassification.
- Visually compare the performance of classification models in different regions of the predictor space.

To zoom or pan, point to the scatter plot and click one of the buttons that appear near the top-right corner of the plot. For examples and more information, see “Classification Learner App”.

cvpartition Function: Create nonstratified partitions of data for cross-validation

If the first input to `cvpartition` is `group`, then you can create nonstratified random partitions for k-fold and holdout cross-validation by using the `'Stratify'` name-value pair argument.

Bayesian optimization example

A new featured example, “Bayesian Optimization with Tall Arrays” shows how to use `bayesopt` to select optimal parameters for training a kernel classifier on tall arrays.

Code generation example

A new featured example “Human Activity Recognition Simulink Model for Smartphone Deployment” shows how to prepare a Simulink® model that classifies human activity based on smartphone sensor signals for code generation and smartphone deployment.

Functionality Being Removed or Changed

Functionality	Result	Use Instead	Compatibility Considerations
`classregtree`	Errors	`fitctree` or `fitrtree`	Replace all instances of `classregtree` with `fitctree` or `fitrtree`.
`svmtrain`	Errors	`fitcsvm`	Replace all instances of `svmtrain` with `fitcsvm`.
`svmclassify`	Errors	`predict` of `ClassificationSVM`	Replace all instances of `svmclassify` with `predict`.
`princomp`	Errors	`pca`	Replace all instances of `princomp` with `pca`.

Functionality	Result	Use Instead	Compatibility Considerations
`fitNaiveBayes`	Removed	`fitcnb`	Replace all instances of `fitNaiveBayes` with `fitcnb`.
`ProbDist`	Removed	`makedist` and `fitdist`	To create and fit probability distribution objects, use `makedist` and `fitdist` instead.
`ProbDistParametric`	Removed	`makedist` and `fitdist`	To create and fit probability distribution objects, use `makedist` and `fitdist` instead.
`ProbDistKernel`	Removed	`makedist` and `fitdist`	To create and fit probability distribution objects, use `makedist` and `fitdist` instead.
`ProbDistUnivKernel`	Removed	`makedist` and `fitdist`	To create and fit probability distribution objects, use `makedist` and `fitdist` instead.
`ProbDistUnivParam`	Removed	`makedist` and `fitdist`	To create and fit probability distribution objects, use `makedist` and `fitdist` instead.

R2017b

Code Generation: Generate C code for prediction by using discriminant analysis, k-nearest neighbor, SVM regression, regression tree ensemble, and Gaussian process regression models (requires MATLAB Coder)

The following functions support code generation:

- `predict` (`CompactClassificationDiscriminant`) — Classify observations or estimate classification scores and costs by applying a discriminant analysis classification to new data.
- `predict` (`ClassificationKNN`) — Classify observations or estimate classification scores and costs by applying *k*-nearest neighbor classification, based on an exhaustive search, to new data.
- `predict` (`CompactRegressionSVM`) — Predict responses by applying a support vector machine (SVM) regression to new data.
- `predict` (`CompactRegressionEnsemble`) — Predict responses by applying ensembles of regression trees to new data.
- `predict` (`RegressionLinear`) — Predict responses by applying a linear regression to new data.
- `predict` (`CompactRegressionGP`) — Predict responses or estimate confidence intervals on predictions by applying a Gaussian process regression to new data.
- `knnsearch` (`ExhaustiveSearcher`) and `knnsearch`— Identify the *k*-nearest neighbors using the exhaustive search algorithm.
- `rangesearch` (`ExhaustiveSearcher`) and `rangesearch` — Identify all neighbors within a specified distance using the exhaustive search algorithm.
- `pdist2` — Compute the pairwise distance between two sets of observations.

When you train an SVM model by using `fitcsvm` for code generation, you can now specify a score transformation function by using the `'ScoreTransform'` name-value pair argument or by assigning the `ScoreTransform` object property. Therefore, `saveCompactModel` can accept compact SVM models equipped to estimate class posterior probabilities, that is, models returned by `fitposterior` or `fitSVMPosterior`. Also, you can now implement one-class learning.

When you train a linear classification model by using `fitclinear` for code generation, you can now specify either `'svm'` or `'logistic'` for the `'Learner'` name-value pair argument.

Big Data Algorithms: Fit kernel SVM classification models by using random feature expansion, fit linear SVM regression models, grow decision trees, and draw weighted random samples from out-of-memory data

These functions support tall arrays:

- `fitckernel` (new function) for Gaussian kernel classification using feature expansion
- `fitrlinear` for SVM regression
- `fitctree` for classification decision trees

Additionally, `datasample` supports weighted sampling without replacement with the `'Weights'` name-value pair argument.

For a complete list of supported statistics functions, see Tall Array Support, Usage Notes, and Limitations.

Parallel Bayesian Optimization: Tune hyperparameters faster by using parallel function evaluation (requires Parallel Computing Toolbox)

By computing in parallel, you can speed your Bayesian optimization. You can optimize hyperparameters in parallel by using `bayesopt` or any of the fit functions that support Bayesian optimization. For details, see Parallel Bayesian Optimization.

When calling a fit function that supports Bayesian optimization, you can limit the total optimization time by setting the `MaxTime` field of the `HyperparameterOptimizationOptions` structure. For a list of the fit functions that support Bayesian optimization, see Bayesian Optimization Using a Fit Function.

Machine Learning Apps: Select training data more efficiently in the Classification Learner and Regression Learner Apps

The New Session dialog box allows you to easily select a data set and response and predictor variables. This dialog box is well suited for large data sets or data sets with many variables. For more details, see Select Data and Validation for Classification Problem and Select Data and Validation for Regression Problem.

Partial Dependence Plots: Visualize relationships between features and predicted responses through marginalization

The `plotPartialDependence` function shows relationships between selected features and predicted responses for a trained regression model object. You can create a partial dependence plot (PDP), individual conditional expectation (ICE) plots, and centered ICE plots. You can also provide additional predictor data to evaluate and plot.

Nonlinear Classification for Big Data: Use fitckernel to train a Gaussian kernel classifier using feature expansion

For nonlinear classification for big data, use the `fitckernel` function to train a binary, Gaussian kernel classification model using feature expansion. `fitckernel` creates an object of the new class `ClassificationKernel`.

`ClassificationKernel` is a new class for accessing and performing operations on the training data. The `ClassificationKernel` model object does not store the training data in a similar way as `ClassificationLinear` and `RegressionLinear`. However, with some exceptions, the syntax and methods resemble those for the other classification model objects that store the training data. For example, to classify observations or estimate classification scores and costs for new data, pass a trained model object and new predictor data to `predict`. To continue training, pass a trained model object and the training data to `resume`.

For all methods and properties of the new objects, see the `ClassificationKernel` class page.

Gaussian Processes: Supply the initial step size for likelihood optimization, or optimize by using LBFGS

You can use LBFGS as an optimizer while training Gaussian process regression models. Specify this option using the `'Optimizer'` name-value pair argument of `fitrgp`.

If `'Optimizer'` is `'quasinewton'` or `'lbfgs'`, then you can specify the approximate maximum absolute value of the first optimization step using the `'InitialStepSize'` name-value pair argument. This specification can improve training time.

ksdensity and mvksdensity Functions: Specify a boundary correction method

For probability density function estimation, `ksdensity` and `mvksdensity` support the reflection method for boundary correction. You can specify this method by using the `'BoundaryCorrection','Reflection'` name-value pair argument.

regularize Function: Specify the maximum number of iterations allowed

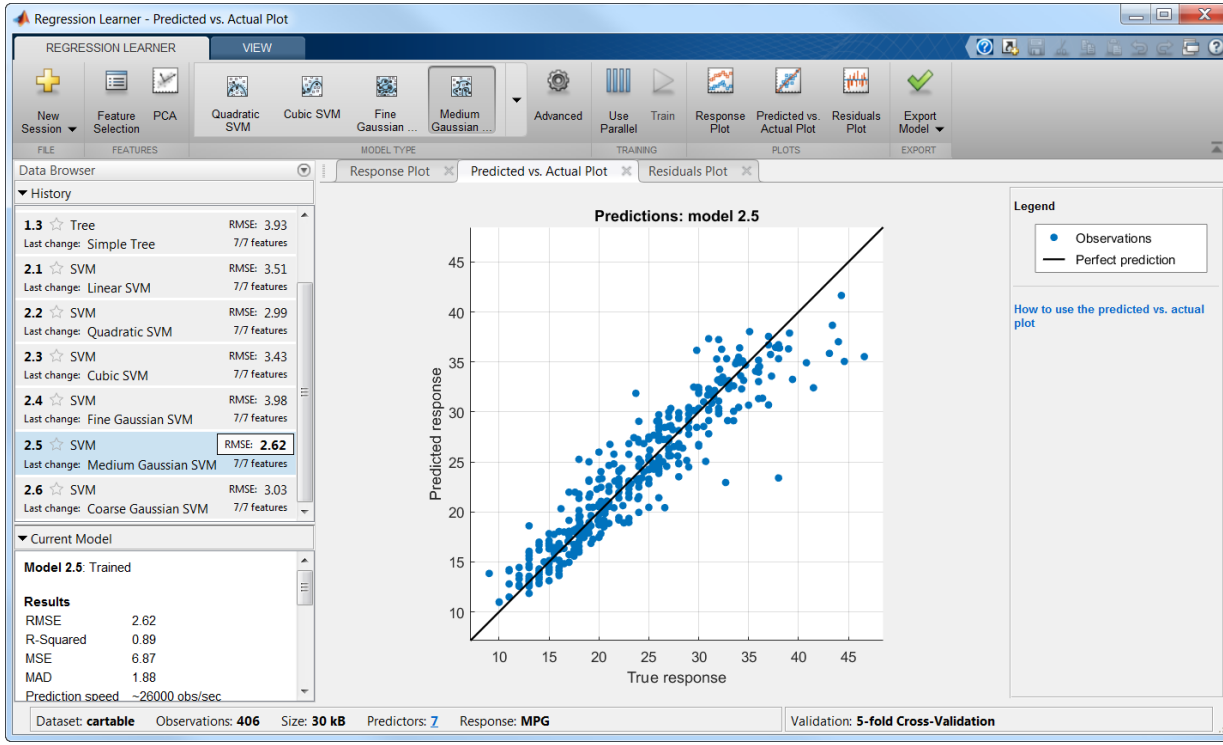
You can specify the maximum number of iterations allowed by using the `'MaxIter'` name-value pair argument in the call to the `regularize` function of a `RegressionEnsemble` object.

R2017a

Regression Learner App: Train regression models using supervised machine learning

Regression Learner is a new app that you can use to train regression models to predict data. Using this app, you can explore your data, select features, specify validation schemes, train models, and assess results. You can perform automated training to search for the best regression model type, including linear regression models, regression trees, Gaussian process models, support vector machines, and ensembles of regression trees.

To use the model with new data, or to learn about programmatic regression, you can export the model to the workspace or generate MATLAB® code to recreate the trained model.



For more information, see Regression Learner App.

Big Data Algorithms: Perform support vector machine (SVM) and Naive Bayes classification, create bags of decision trees, and fit lasso regression on out-of-memory data

Several classification and regression functions add support for tall arrays:

- `fitclinear` for support vector machine classification
- `fitcnb` for Naive Bayes classification
- `TreeBagger` for creating bags of decision trees using ADMM algorithm
- `lasso` for fitting lasso regression using ADMM algorithm
- The `loss` and `predict` methods of these regression classes:
  - `CompactRegressionSVM`
  - `CompactRegressionGP`
  - `CompactRegressionTree`
  - `CompactRegressionEnsemble`
  - `RegressionLinear`
- The `predict`, `loss`, `margin`, and `edge` methods of these classification classes:
  - `CompactClassificationEnsemble`
  - `CompactClassificationTree`
  - `CompactClassificationTree`
  - `CompactClassificationDiscriminant`
  - `CompactClassificationNaiveBayes`
  - `CompactClassificationSVM`
  - `CompactClassificationECOC`
  - `ClassificationKNN`
  - `ClassificationLinear`

For a complete list of supported functions, see Tall Array Support, Usage Notes, and Limitations.

Code Generation: Generate C code for prediction by using linear models, generalized linear models, decision trees, and ensembles of classification trees (requires MATLAB Coder)

You can generate C code that predicts responses by using trained linear models, generalized linear models (GLM), decision trees, or ensembles of classification trees. The following prediction functions support code generation:

- `predict` — Predict responses or estimate confidence intervals on predictions by applying a linear model to new predictor data.
- `predict` or `glmval` — Predict responses or estimate confidence intervals on predictions by applying a GLM to new predictor data.
- `predict` or `predict` — Classify observations or estimate classification scores by applying a classification tree or ensemble of classification trees, respectively, to new data.
- `predict` — Predict responses by applying a regression tree to new data.

You can generate C code to simulate responses from a linear model or a generalized linear model using `random` or `random`, respectively.

Bayesian Statistics: Perform gradient-based sampling using Hamiltonian Monte Carlo (HMC) sampler

You can now perform Hamiltonian Monte Carlo (HMC) sampling from a probability density function. Use the `hmcSampler` function to create a `HamiltonianSampler` object for the log probability density that you specify. The object samples from this density by generating a Markov chain with the corresponding equilibrium distribution using HMC.

After creating a `HamiltonianSampler` object, you can use:

- `tuneSampler` to tune the HMC sampler prior to drawing samples
- `drawSamples` to draw samples from the density
- `estimateMAP` to estimate the maximum of the log probability density
- `diagnostics` to assess the convergence

For a workflow example, see Bayesian Linear Regression Using Hamiltonian Monte Carlo.

Feature Extraction: Perform unsupervised feature learning by using sparse filtering and reconstruction independent component analysis (RICA)

Sparse filtering and reconstruction independent component analysis (RICA) are unsupervised, feature learning techniques for producing informative representations, including overcomplete representations, of high-dimensional predictor data. That is, they combine the raw predictor variables to produce an output feature set containing possibly more variables. Such representations can expose underlying correlations among the raw predictor variables, which can lead to improved predictive accuracy. These techniques are appropriate for image and video processing problems.

`sparsefilt` performs regularized sparse filtering. `rica` performs RICA, a variation of ICA. Whereas ICA enforces the feature weight matrix to be orthonormal during optimization, RICA includes a penalty term in the objective function allowing the feature weight matrix to deviate from orthonormality. Given a dimension for the output feature set, both methods learn predictor weights while imposing sparse/independent representations.

`sparsefilt` returns results as a `SparseFiltering` object and `rica` returns results as a `ReconstructionICA` object. After learning the feature weights, pass the object and predictor data to `transform` to transform the predictor data to the learned representation. You can train any classification or regression model using the new representation. For details, see the reference pages and Feature Extraction Workflow.

t-SNE: Visualize high-dimensional data

Using the `tsne` function you can embed high-dimensional data into two or three dimensions in order to view natural clustering. For details, see the function reference page and Visualize High-Dimensional Data Using t-SNE.

Survival Analysis: Fit Cox proportional hazards models with time-dependent covariates

Use `coxphfit` to fit a Cox proportional hazards model to data with time-dependent covariates. `coxphfit` uses the counting process type input to handle this type of data. You can enter the risk intervals for each subject using the `T` argument in the call to `coxphfit`.

Distribution Fitting App: dfittool Renamed to distributionFitter

To start the Distribution Fitter app at the command line, enter `distributionFitter`. The `dfittool` function also starts the Distribution Fitter app.

lasso and lassoglm Functions: Specify maximum number of iterations allowed

You can specify the maximum number of iterations allowed using the new `'MaxIter'` name-value pair argument in the call to `lasso` and `lassoglm`. For details, see the function reference pages.

Functionality Being Changed

The following functionality is removed or will be removed in a future release. Use the newer functionality instead.

Functionality	What Happens When You Use This Functionality?	Use This Functionality Instead	Compatibility Considerations
`treedisp`	Removed	`view` (`ClassificationTree`) or `view` (`RegressionTree`)	Use `fitctree` or `fitrtree` to grow a tree. Replace all instances of `treedisp` with `view` (`ClassificationTree`) or `view` (`RegressionTree`).

Functionality	What Happens When You Use This Functionality?	Use This Functionality Instead	Compatibility Considerations
`treefit`	Removed	`fitctree` or `fitrtree`	Replace all instances of `treefit` with `fitctree` or `fitrtree`.
`treeprune`	Removed	`prune` (`ClassificationTree`) or `prune` (`RegressionTree`)	Use `fitctree` or `fitrtree` to grow a tree. Replace all instances of `treeprune` with `prune` (`ClassificationTree`) or `prune` (`RegressionTree`) .

Functionality	What Happens When You Use This Functionality?	Use This Functionality Instead	Compatibility Considerations
`treetest`	Removed	• `resubLoss` (`ClassificationTree`) or `resubLoss` (`RegressionTree`) • `loss` (`ClassificationTree`) or `loss` (`RegressionTree`) • `cvloss` (`ClassificationTree`) or `cvLoss` (`RegressionTree`)	Use `fitctree` or `fitrtree` to grow a tree. Replace all instances of • `treetest(T,'resubstitution')` with `resubLoss` (`ClassificationTree`) or `resubLoss` (`RegressionTree`) • `treetest(T,'test',X,Y)` with `loss` (`ClassificationTree`) or `loss` (`RegressionTree`) • `treetest(T,'crossvalidate',X,Y)` with `cvloss` (`ClassificationTree`) or `cvloss`

Functionality	What Happens When You Use This Functionality?	Use This Functionality Instead	Compatibility Considerations
			(RegressionTree)
treeval	Removed	predict (ClassificationTree) or predict (RegressionTree)	Use fitctree or fitrtree to grow a tree. Replace all instances of treeval with predict (ClassificationTree) or predict (RegressionTree).
classregtree	Warning	fitctree or fitrtree	Replace all instances of classregtree with fitctree or fitrtree.
svmtrain	Warning	fitcsvm	Replace instances of svmtrain with fitcsvm.
svmclassify	Warning	fitcsvm	Replace instances of svmclassify with fitcsvm.
princomp	Warning	pca	Replace all instances of princomp with pca.

Functionality	What Happens When You Use This Functionality?	Use This Functionality Instead	Compatibility Considerations
`fitNaiveBayes`	Error	`fitcnb`	Replace all instances of `fitNaiveBayes` with `fitcnb`.
`ProbDist`	Error	`makedist` and `fitdist`	To create and fit probability distribution objects, use `makedist` and `fitdist` instead.
`ProbDistParametric`	Error	`makedist` and `fitdist`	To create and fit probability distribution objects, use `makedist` and `fitdist` instead.
`ProbDistKernel`	Error	`makedist` and `fitdist`	To create and fit probability distribution objects, use `makedist` and `fitdist` instead.
`ProbDistUnivKernel`	Error	`makedist` and `fitdist`	To create and fit probability distribution objects, use `makedist` and `fitdist` instead.

Functionality	What Happens When You Use This Functionality?	Use This Functionality Instead	Compatibility Considerations
`ProbDistUnivParam`	Error	`makedist` and `fitdist`	To create and fit probability distribution objects, use `makedist` and `fitdist` instead.

R2016b

Big Data Algorithms: Perform dimension reduction, descriptive statistics, k-means clustering, linear regression, logistic regression, and discriminant analysis on out-of-memory data

Tall arrays provide a way to work naturally with out-of-memory data. You can create tall numeric arrays, cell arrays, categoricals, and you can use any of these tall types as variables in a tall table. For more information, see Tall Arrays.

You can perform various statistical and machine learning analyses on tall arrays. For a complete list of supported functions, see Tall Array Support, Usage Notes, and Limitations.

Bayesian Optimization: Tune machine learning algorithms by searching for optimal hyperparameters

Bayesian optimization is an algorithm for optimizing functions that can be nondifferentiable, discontinuous, and take a significant amount of time to evaluate. The algorithm internally maintains a Gaussian process model of the objective function, and uses objective function evaluations to train the model.

Bayesian optimization is well suited to optimizing hyperparameters of classification and regression algorithms. Hyperparameters are internal parameters of classification and regression functions, and can affect the performance of these functions. You can tune hyperparameters using Bayesian optimization in two ways:

- Using `bayesopt` to optimize a custom objective function over the parameters you define for tuning. This option gives you more control over optimization.
- Using the `OptimizeHyperparameters` name-value pair in training functions for classification and nonparametric regression. This option minimizes a cross-validation error over a preselected choice of hyperparameters.

For details, see Bayesian Optimization Workflow.

Feature Selection: Use neighborhood component analysis (NCA) to choose features for machine learning models

Neighborhood component analysis (NCA) is a nonparametric and embedded method for selecting features with the goal of maximizing prediction accuracy of regression and

classification algorithms. `fscnca` and `fsrnca` perform NCA with regularization to learn feature weights for minimization of an objective function that measures the average leave-one-out classification or regression loss over the training data. `fscnca` and `fsrnca` support various optimization methods:

- Limited memory BFGS (LBFGS) — Recommended when the number of features is large.
- Stochastic gradient descent (SGD) — Recommended when the number of observations is large.
- Mini-batch LBFGS — Hybrid method that combines LBFGS and SGD. This method might converge faster than LBFGS or SGD.

`fscnca` returns results as a `FeatureSelectionNCAClassification` object, and `fsrnca` returns results as a `FeatureSelectionNCARegression` object.

Using these objects you can:

- Predict continuous responses or class labels using `predict`.
- Compute prediction or classification error using `loss` to estimate the prediction accuracy of selected features or to tune the regularization parameter.
- Refit the model using modified settings or continue iterations starting from the current solution using `refit`.

Code Generation: Generate C code for prediction by using SVM and logistic regression models (requires MATLAB Coder)

You can generate C code that classifies new observations by using trained, binary support vector machine (SVM) or logistic regression models, or multiclass SVM or logistic regression via error-correcting output codes (ECOC).

- `saveCompactModel` compacts and saves the trained model to disk.
- `loadCompactModel` loads the compact model in a prediction function that you declare. The prediction function can, for example, accept new observations and return labels and scores.
- `predict` classifies and estimates scores for the new observations in the prediction function.

    - To classify by using binary SVM models, see `predict`.

- To classify by using binary logistic regression models, see `predict`.
- To classify by using multiclass SVM or logistic regression via ECOC, see `predict`.

Classification Learner: Train classifiers in parallel (requires Parallel Computing Toolbox)

If you have Parallel Computing Toolbox™, you can train models in parallel using Classification Learner. When you train classifiers, the app automatically starts a parallel pool of workers, unless you turn off the default parallel pool preference.

Parallel training allows you to train multiple classifiers at once and continue working. During training, you can examine results and plots from models, and initiate training of more classifiers.

For details, see Parallel Classifier Training.

Machine Learning Performance: Speed up Gaussian mixture modeling, SVM with duplicate observations, and distance calculations for sparse data

The algorithms for `pdist` and `fitgmdist` show improved performance.

When using `fitrsvm` and `fitcsvm`, you can remove duplicate observations for improved training time. This option replaces duplicate observations with a single observation whose weight is equal to the cumulative weight of these observations. To remove the duplicate observations, use the `RemoveDuplicates` name-value pair argument in the call to `fitrsvm` and `fitcsvm`.

Survival Analysis: Fit Cox proportional hazards models with new options for residuals and handling ties

Use `coxphfit` to fit Cox proportional hazards model to stratified data or data with ties. Use the `Strata` name-value pair argument to specify the stratification in the data. Use the `Ties` name-value pair argument to specify the method (Breslow or Efron) to handle tied failure times.

`coxphfit` also returns various residuals, including Cox-Snell, deviance, martingale, Schoenfeld, and score residuals.

Ensemble Methods Usability: Use simpler functions to train classification or regression ensembles

The `fitcensemble` and `fitrensemble` functions provide simpler interfaces to fit ensemble learners for classification and regression, respectively.

Unlike `fitensemble`, `fitcensemble` and `fitrensemble` provide options for Bayesian optimization.

Compatibility Considerations

For ensembles of boosted decision trees, `fitcensemble` and `fitrensemble` set their default tree depths to allow a maximum of ten splits, whereas `fitensemble` allows one split only by default.

Quantile Regression: Use bagged regression trees (TreeBagger) to implement quantile regression

In addition to predicting the conditional mean of a continuous response by growing a random forest [9], `TreeBagger` can predict conditional quantiles for robust regression and conditional distribution estimation.

- To predict quantiles of observations using a `TreeBagger` regression model, use `quantilePredict`.
- To estimate the quantile loss, use `quantileLoss`.
- To make out-of-bag predictions, use `oobQuantilePredict`.
- To estimate the out-of-bag quantile loss, use `oobQuantileLoss`.

GPU support: pdist, pdist2, and knnsearch accept gpuArray

`pdist`, `pdist2`, and `knnsearch` functions are enhanced to accept `gpuArray` input arguments so that they execute on the GPU. This support requires Parallel Computing Toolbox.

Gaussian Processes: Use additional popular kernel functions

You can use four new kernel (covariance) functions while training Gaussian Process Regression models. The new options are: exponential, rational quadratic, ARD

exponential, and ARD rational quadratic kernel. You can specify the kernel function using the `KernelFunction` name-value pair argument in `fitrgp`. For more information on the kernel functions, see Kernel (Covariance) Function Options.

coxphfit Function: Specify coefficient initial values and observation weights

The name of the `'Init'` name-value pair argument is now `B0`. Use `B0` to specify the initial values for the estimated model coefficients. The name `Init` still works.

You can pass observation weights using the `'Frequency'` name-value pair argument, which now can be an array containing nonnegative scalar values.

fitgmdist Function: Set initial values using kmeans++ algorithm by default

The default value for the `Start` name-value pair argument of `fitgmdist` has changed to `'plus'`. Previously it was `'randSample'`.

Compatibility Considerations

To use `'randSample'` as the initial value setting method, specify `'Start','randSample'` in the call to `fitgmdist`.

fitgmdist Function: Specify tolerance for posterior probabilities

You can specify the tolerance for posterior probability estimates using the new `'ProbabilityTolerance'` name-value pair argument in the call to `fitgmdist`. In each iteration, after the estimation of posterior probabilities, `fitgmdist` sets any posterior probability that is not larger than the tolerance value to zero.

The `gmdistribution` object stores the tolerance value in the new `ProbabilityTolerance` property.

fitctree, fitrtree, and templateTree Functions: Unbiased feature selection for decision trees

For more flexibility in growing decision trees, `fitctree`, `fitrtree`, and `templateTree` offer several predictor-splitting algorithms. You can choose these alternatives using the `'PredictorSelection'` name-value pair argument.

- `'allsplits'`: At each node, MATLAB chooses the predictor that maximizes the split-criterion gain over all possible splits. This is the default and the expected algorithm. This algorithm is biased toward choosing predictors with many distinct values.
- `'curvature'`: MATLAB chooses the predictor by minimizing the p-value of a χ^2 test of independence between each predictor and response.
- `'interaction-curvature'`: MATLAB chooses the predictor to split by minimizing the smallest p-value of χ^2 tests of independence between:
  - Each predictor and response.
  - Each pair of predictors and response.

The algorithms that use χ^2 tests to split predictors are unbiased with respect to the number of distinct values in a predictor. Also, you can use these algorithms for feature selection.

R2016a

Machine Learning for High-Dimensional Data: Perform fast fitting of linear classification and regression models with techniques such as stochastic gradient descent and (L)BFGS using fitclinear and fitrlinear functions

For faster training on high-dimensional data sets, use `fitrlinear` and `fitclinear` to fit regularized, linear regression and binary classification models, respectively. For multiclass classification problems, specify parameters by creating a linear classification model template using `templateLinear`, and then pass the template object to `fitcecoc` for training.

Models for linear regression include support vector machine (SVM) and least squares regression, and models for binary classification include SVM and logistic regression. You can additionally include a lasso or ridge penalty to the objective function. The software optimizes the objective function using any of these algorithms:

- Stochastic gradient descent (SGD)
- Average SGD
- Broyden-Fletcher-Goldfarb-Shanno (BFGS)
- Limited-memory BFGS (LBFGS)
- Sparse Reconstruction by Separable Approximation (SpaRSA)

To increase the execution speed when training using `fitclinear`, `fitrlinear`, or `fitcecoc`, orient the predictor data so that columns correspond to observations, and set `'ObservationsIn','columns'`.

`fitrlinear` and `fitclinear` return:

- `RegressionLinear` and `ClassificationLinear` model objects, respectively, by default
- `RegressionPartitionedLinear` and `ClassificationPartitionedLinear` model objects, respectively, when you specify to cross-validate

For multiclass classification problems using linear classification models, `fitcecoc` returns:

- A `CompactClassificationECOC` model composed of `ClassificationLinear` model objects

- A `ClassificationPartitionedLinearECOC` model object when you specify to cross-validate

Unlike other regression and classification model objects in Statistics and Machine Learning Toolbox, these objects do not store the training data. However, with some exceptions, the syntax and methods resemble those for the other regression and classification model objects. For example, to predict responses or classes for new data, pass a trained linear regression or classification model object to `predict`.

Classification Learner: Train multiple models automatically, visualize results by class labels, and perform logistic regression classification

Classification Learner helps you explore methods for training models to classify data using supervised machine learning. In R2016a, new features in the app include:

- Automated classifier training. Get started by automatically training a selection of different classification models on your data with one click. Use automated training to quickly try a selection of model types, then explore promising models interactively. Models show progress bars while training, and you can interrupt training between models or between validation folds. The app highlights the best accuracy score.
- Results visualized by class. In the scatter plot, show or hide particular classes, or focus only on correct or incorrect predictions.
- Logistic regression classification. Try a popular baseline classification technique on your data.

For details, see Train Classification Models in Classification Learner App.

Performance: Perform clustering using kmeans, kmedoids, and Gaussian mixture models faster when data has a large number of clusters

The algorithms for `kmeans`, `kmedoids`, and `fitgmdist` (for *plus* initialization method) show improved performance, particularly when there is a large number of clusters in the data.

Probability Distributions: Fit kernel smoothing density to multivariate data using the ksdensity and mvksdensity functions

`ksdensity` now supports fitting and plotting a probability density estimate for bivariate sample data. Use the new name-value pair `'PlotFcn'` to select the plot type for bivariate sample data. Choose from a contour plot, 3-D line plot, 3-D shaded surface plot, or contour plot under a 3-D shaded surface plot.

Use `mvksdensity` to fit a probability density estimate to multivariate data.

Stable Distributions: Model financial and other data that requires heavy-tailed distributions

There is a new probability distribution object for the stable distribution. This distribution is commonly used to model financial and other data that requires heavy-tailed distributions. Use `fitdist` to fit this distribution to data. Use `makedist` to specify the distribution parameters directly. Either function produces a probability distribution object that you can use to generate random samples or compute functions such as pdf and cdf.

Half-Normal Distributions: Model truncated data and create half-normal probability plots

There is a new probability distribution object for the half-normal distribution. This distribution is commonly used to model truncated data.

- Use `fitdist` to fit this distribution to data. Use `makedist` to specify the distribution parameters directly. Either function produces a probability distribution object that you can use to generate random samples or compute functions such as pdf and cdf.
- Use `probplot` to create a half-normal probability plot.

Linear Regression: CompactLinearModel object reduces memory footprint of linear regression model

`CompactLinearModel` is a new class for storing configurations of fitted linear regression models without storing fitting data or residuals. Fit a full `LinearModel` object using `fitlm`, then use the new compact method to create a `CompactLinearModel` object that retains only summary information about the model, such as coefficient values.

For all methods and properties of the new objects, see the `CompactLinearModel` and `LinearModel` class pages.

Robust covariance estimation for multivariate sample data using robustcov

The new function `robustcov` estimates a robust covariance matrix for multivariate sample data. Robust covariance estimates are less sensitive to outliers in the sample data than classical estimation methods. Estimation options available using robustcov include the FAST-MCD (Minimum Covariance Determinant) estimate, the Orthogonalized Gnanadesikan-Kettenring (OGK) estimate, and an estimate based on “concentration” techniques.

Squared Euclidean distance measure for pdist and pdist2 functions

`pdist` and `pdist2` support the option of returning the squared Euclidean distance. You can specify this measure by setting the `distance` positional argument to `'squaredeuclidean'`.

Performance enhancements for nearest neighbor search using kd-tree

For dual-core systems and above, the `knnsearch` method of `KDTreeSearcher` parallelizes the k-nearest neighbor search using Intel® Threading Building Blocks (TBB). For details on Intel TBB, see https://software.intel.com/en-us/intel-tbb.

GPU support for extreme value distribution functions and kmeans

The following Statistics and Machine Learning Toolbox functions are enhanced to accept `gpuArray` input arguments so that they execute on the GPU. This support requires Parallel Computing Toolbox.

```
evcdf
evpdf
evinv
evlike
evrnd
evstat
kmeans
```

Changes to default online update phase for kmeans function

The default value of `OnlinePhase` name-value pair argument for `kmeans` is now `'off'`. Previously, the default value was `'on'`.

Compatibility Considerations

`kmeans` might return different results than when the default value of `OnlinePhase` was `'on'`. To turn the online update phase on, use the `'OnlinePhase','on'` name-value pair argument in the call to `kmeans`.

Name change in ksdensity

The name-value pair `'npoints'` for `ksdensity` has changed to `'NumPoints'`.

Compatibility Considerations

The new name-value pair argument name `'NumPoints'` does not work in previous releases. Please refer to the documentation for the correct version of Statistics and Machine Learning Toolbox you use.

Name change in paretotails

The following property names for the `paretotails` class have changed.

New Property Name	Old Property Name	Class

`NumSegments`
`UpperParameters`
`LowerParameters`
	`nsegments`
	`lowerparams`
	`upperparams`
		`paretotails`

Compatibility Considerations

The new property names do not work in previous releases. Please refer to the documentation for the correct version of Statistics and Machine Learning Toolbox you use.

Functionality Being Changed

Following functionality will be removed in a future release. Use the newer functionality instead.

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`treedisp`	Error	`view` (`ClassificationTree`) or `view` (`RegressionTree`)	Use`fitctree` or `fitrtree` to grow a tree. Replace all instances of `treedisp` with`view` (`ClassificationTree`) or `view` (`RegressionTree`).
`treefit`	Error	`fitctree` or `fitrtree`	Replace all instances of `treefit` with `fitctree` or `fitrtree`.
`treeprune`	Error	`prune` (`ClassificationTree`) or `prune` (`RegressionTree`)	Use `fitctree` or `fitrtree` to grow a tree. Replace all instances of `treeprune` with `prune` (`ClassificationTree`) or `prune` (`RegressionTree`) .

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`treetest`	Error	• `resubLoss` (`ClassificationTree`) or `resubLoss` (`RegressionTree`) • `loss` (`ClassificationTree`) or `loss` (`RegressionTree`) • `cvLoss` (`ClassificationTree`) or `cvLoss` (`RegressionTree`)	Use `fitctree` or `fitrtree` to grow a tree. Replace all instances of • `treetest(T,'resubstitution')` with `resubLoss` (`ClassificationTree`) or `resubLoss` (`RegressionTree`) • `treetest(T,'test',X,Y)` with `loss` (`ClassificationTree`) or `loss` (`RegressionTree`) • `treetest(T,'crossvalidate',X,Y)` with `cvLoss` (`ClassificationTree`) or `cvLoss` (`RegressionTree`)

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`treeval`	Error	`predict` (`ClassificationTree`) or `predict` (`RegressionTree`)	Use `fitctree` or `fitrtree` to grow a tree. Replace all instances of `treeval` with `predict` (`ClassificationTree`) or `predict` (`RegressionTree`).
`classify`	Warning	`fitcdiscr`	Replace all instances of `classify` with `fitcdiscr`.
`fitNaiveBayes`	Warning	`fitcnb`	Replace all instances of `fitNaiveBayes` with `fitcnb`.
`ProbDist`	Warning	`makedist` and `fitdist`	To create and fit probability distribution objects, use `makedist` and `fitdist` instead.
`ProbDistParametric`	Warning	`makedist` and `fitdist`	To create and fit probability distribution objects, use `makedist` and `fitdist` instead.
`ProbDistKernel`	Warning	`makedist` and `fitdist`	To create and fit probability distribution objects, use `makedist` and `fitdist` instead.

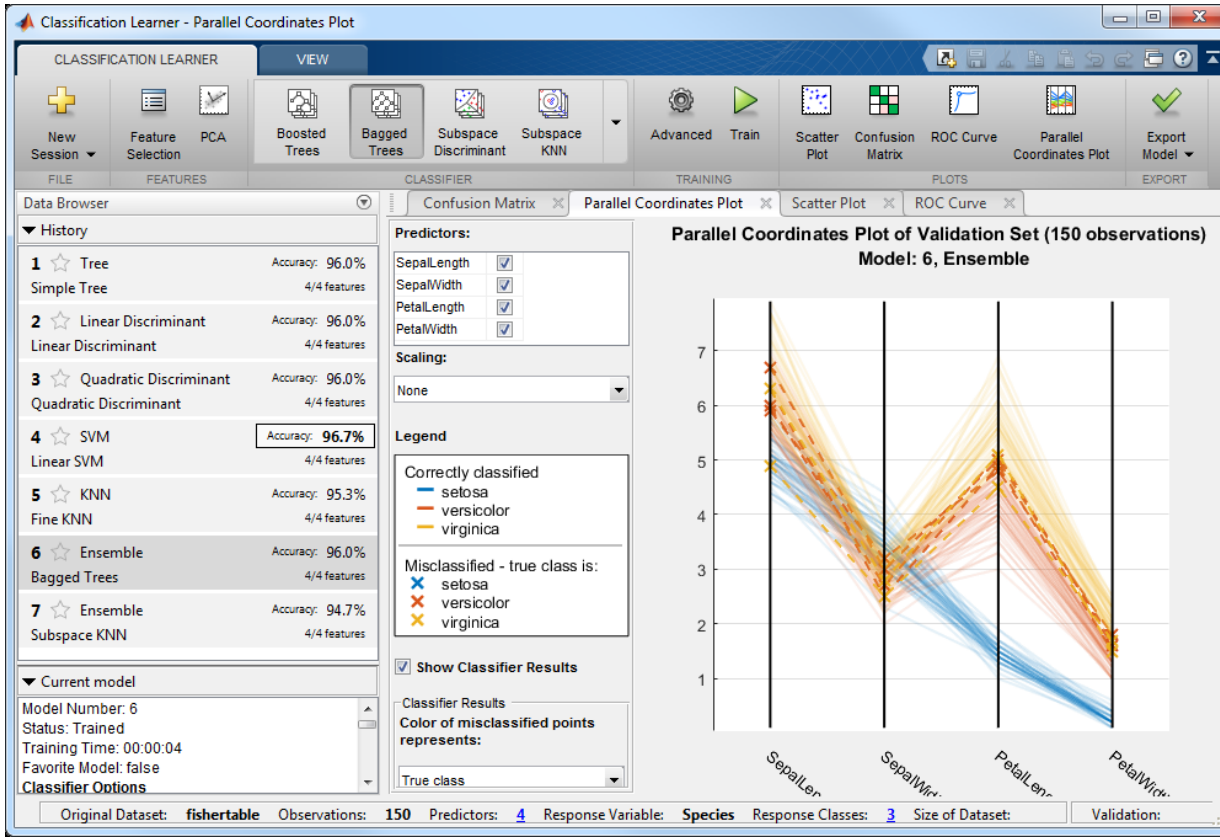
Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`ProbDistUnivKernel`	Warning	`makedist` and `fitdist`	To create and fit probability distribution objects, use `makedist` and `fitdist` instead.
`ProbDistUnivParam`	Warning	`makedist` and `fitdist`	To create and fit probability distribution objects, use `makedist` and `fitdist` instead.

R2015b

Classification Learner: Train discriminant analysis to classify data, train models using categorical predictors, and perform dimensionality reduction using PCA

Classification Learner helps you explore methods for training models to classify data using supervised machine learning. In R2015b, new features in the app include:

- Discriminant analysis classifier: Train classifiers with fast, accurate, and easy to interpret discriminant analysis, which is good for wide datasets.
- Principal component analysis (PCA): Reduce the dimensionality of the predictor space using PCA to help prevent overfitting.
- Categorical predictors: Train classification models when some or all predictors are categorical variables. Previously you could only have numeric predictors in the app.
- Data import from file: Import spreadsheets, text, csv, and other files into the app. Previously you could only select data from the workspace.
- Parallel coordinates plot: Visualize training data and misclassified points to investigate features to include or exclude. Parallel coordinates can help visualize 3 to 10 dimensions of data on a single plot and see patterns. This can help you understand relationships between features and identify useful predictors for separating classes.
- ROC Threshold: Assess classifier performance by seeing where the threshold for your trained classifier lies on the ROC curve.



For details, see Explore Classification Models Interactively.

Compatibility Considerations

If you exported a classification model from Classification Learner to the workspace and wrote a script to make predictions with new data in R2015a, you must change your code to use models exported from the app in R2015b. Use the new `trainedClassifier.predictFcn`.

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
In user scripts, `predict` function used on model exported from Classification Learner in R2015a	Errors in R2015b	`trainedClassifier.predictFcn`	Classification models exported from Classification Learner changed from a classification object in R2015a to a structure in R2015b. The new structure contains a classification object and a new predict function, `predictFcn`. The new structure allows you to make predictions for models that include principal component analysis (PCA).

When you export a model from Classification Learner, the app displays information about the exported model in the command window. The message shows how to make predictions using the model.

To fix your code from R2015a to work with classifiers exported in R2015b, change `predict` to `trainedClassifier.predictFcn`, where `trainedClassifier` is the name of your struct variable.

For example, change this R2015a code:

```
yfit = predict(trainedClassifier,T{:,trainedClassifier.PredictorNames})
```

To this R2015b code:

```
yfit = trainedClassifier.predictFcn(T)
```

Supply the data `T` in same data type as your training data used in the app (table or matrix).

- If you supply a table, ensure it contains the same predictor names as your training data. The `predictFcn` ignores additional variables in tables.

- If you supply a matrix, it must contain the same predictor columns or rows as your training data, and no response variable or other unused variables.

Note that the default name `trainedClassifier` increments every time you export to avoid overwriting your classifiers, e.g., `trainedClassifier1`. Make sure your code uses the correct name of your struct variable.

You can also extract the classification object from the exported struct for further analysis (e.g., `trainedClassifier.ClassificationSVM`, `trainedClassifier.ClassificationTree`, etc., depending on your model type). Be aware that if you used feature selection such as PCA in the app, you will need to take account of this transformation by using the information in the PCA fields of the struct.

Nonparametric Regression: Fit models using support vector regression (SVR) or Gaussian processes (Kriging)

You can train nonparametric regression models using support vector machine (SVM) regression or Gaussian process regression (GPR).

- **SUPPORT VECTOR REGRESSION:** The `fitrsvm` function trains a SVM regression model. Using the new functionality, you can:

  - Specify the kernel function
  - Provide observation weights
  - Train a cross-validated model
  - Predict responses using the trained model
  - Compute resubstitution statistics

  `fitrsvm` creates a `RegressionSVM` or `RegressionPartitionedSVM` object. `RegressionSVM` is a new class for accessing and performing operations on the training data. `CompactRegressionSVM` is a new class for storing configurations of trained models without storing training data. `RegressionPartitionedSVM` is a new class for a set of cross-validated SVM regression models trained on cross-validated folds.

  For all methods and properties of the new objects, see the `RegressionSVM`, `CompactRegressionSVM`, and `RegressionPartitionedSVM` class pages.
- **GAUSSIAN PROCESS REGRESSION:** The `fitrgp` function trains a Gaussian process regression (GPR) model. Using the new functionality, you can:

- Specify the fitting, prediction, and active set selection methods
- Specify the kernel (covariance) function and provide initial values for the hyperparameters
- Train a cross-validated model
- Compute response predictions along with the prediction intervals using the trained model
- Compute resubstitution statistics
- Compute post-fit statistics

`fitrgp` creates a `RegressionGP` or `RegressionPartitionedModel` object. `RegressionGP` is a new class for accessing and performing operations on the training data. `CompactRegressionGP` is a new class for storing configurations of trained models without storing training data. `RegressionPartitionedModel` is an existing class for a set of cross-validated GPR models trained on cross-validated folds.

For all methods and properties of the new objects, see the `RegressionGP`, `CompactRegressionGP`, and `RegressionPartitionedModel` class pages.

Tables and Categorical Data for Machine Learning: Use table and categorical predictors in classification and nonparametric regression functions and in Classification Learner

The classification functions `fitctree`, `fitcsvm`, `fitcdiscr`, `fitcnb`, and `fitcknn`, non-parametric regression functions `fitrtree`, `fitrsvm`, and `fitrgp`, and the ensemble learner `fitensemble` accept data in `table`. Except for `fitcdiscr`, all of the above listed functions and Classification Learner accept `categorical` predictors. For more information on these data types, see Tables and Categorical Arrays.

Code Generation: Automatically generate C and C++ code for kmeans and randsample functions (using MATLAB Coder)

`kmeans` and `randsample` are now supported for code generation. For a full list of Statistics and Machine Learning Toolbox functions that are supported by MATLAB Coder™, see Statistics and Machine Learning Toolbox.

GPU Acceleration: Speed up computation for over 65 functions including probability distributions, descriptive

statistics, and hypothesis testing (using Parallel Computing Toolbox)

The following Statistics and Machine Learning Toolbox functions are enhanced to accept `gpuArray` input arguments so that they execute on the GPU.

betacdf*	expfit*	geomean*	mvncdf*	poisscdf*	tpdf
betainv*	expinv*	geopdf*	mvnpdf*	poissinv*	tstat
betapdf	explike*	geornd*	mvnrnd*	poisspdf	ttest*
betastat	exppdf*	geostat	nanmax*	poisstat,	ttest2*
binofit*	exprnd*	harmmean*	nanmean*	prctile	trimmean
binoinv	expstat	iqr*	nanmedian	quantile	unidcdf
binopdf	fcdf*	kurtosis*	nanmin*	range*	unidinv
binornd*	finv*	logncdf*	nanstd*	raylcdf	unidpdf
binostat	fstat	lognfit*	nansum*	raylfit*	unidrnd*
chi2cdf*	gamcdf*	logninv*	nanvar*	raylinv	unidstat
chi2gof*	gaminv*	lognlike*	normcdf*	raylpdf	unifinv
chi2inv*	gamlike*	lognpdf*	norminv*	raylrnd*	unifrnd*
chi2stat	gampdf	lognrnd*	normlike*	raylstat	unifstat
cholcov*	gamstat	lognstat	normpdf*	skewness*	vartest
corrcov*	geocdf*	mad*	normrnd*	tcdf*	vartest2*
expcdf*	geoinv*	moment*	normstat	tinv*	zscore*
					ztest*

* These functions perform faster on the GPU than the CPU. Other functions in the table show similar performance on the GPU and on the CPU.

Option to turn off clipping of Alpha coefficients in fitcsvm

You can specify not to clip the Alpha coefficient for an observation to zero or the box-constraint value for that observation, using the `'ClipAlphas',false` name-value pair argument of `fitcsvm`, while training a support vector machine for classification.

Name changes in TreeBagger

Some property and method names for `TreeBagger` and `CompactTreeBagger` classes and name-value pair argument names for the methods of these classes have changed as follows.

New Property Name	Old Property Name	Class
InBagFraction TreeArguments ComputeOOBPredictorImportance NumPredictorsToSample NumTrees PredictorNames OOBPermutedPredictorDeltaError OOBPermutedPredictorDeltaMeanMargin OOBPermutedPredictorCountRaiseMargin NumPredictorSplit SurrogateAssociation MinLeafSize DeltaCriterionDecisionSplit	FBoot TreeArgs ComputeOOBVarImp NVarToSample NTrees VarNames OOBPermutedVarDeltaError OOBPermutedVarDeltaMeanMargin OOBPermutedVarCountRaiseMargin NVarSplit VarAssoc MinLeaf DeltaCritDecisionSplit	TreeBagger
NumTrees PredictorNames NumPredictorSplit SurrogateAssociation	NTrees VarNames DeltaCritDecisionSplit VarAssoc	CompactTreeBagger

New Name-Value Pair Name	Old Name-Value Pair Name	Method	Class
InBagFraction NumPredictorsToSample OOBPrediction OOBPredictorImportance NumPredictorsToSample NumPrint MinLeafSize MinParentSize Cost Method Options Prior SampleWithReplacement Weights	FBoot NVarToSample oobpred oobvarimp nvartosample nprint minleaf minparent cost method options prior samplewithreplacement weights	TreeBagger (constructor)	TreeBagger
NumPrint	nprint	growTrees, fillprox	TreeBagger
UseInstanceForTree	useifort	predict, error, margin, meanMargin	TreeBagger
Options	options	growTrees	TreeBagger
Trees	trees	fillprox, predict, oobPredict, oobError, oobMargin, oobMeanMargin, error, margin, meanMargin,	TreeBagger
TreeWeights	treeweights	predict, oobPredict, oobError, oobMargin, oobMeanMargin, error, margin, meanMargin	TreeBagger

New Name-Value Pair Name	Old Name-Value Pair Name	Method	Class
`Mode`	`mode`	`oobError`, `oobMargin`, `oobMeanMargin`, `error`, `margin`, `meanMargin`	`TreeBagger`

`Keep`
`Colors`
`MDSCoordinates`
	`keep`
	`colors`
	`mdscoords`
		`mdsprox`	`TreeBagger`

`UseInstanceForTree`
`Trees`
`TreeWeights`
	`useifort`
	`trees`
	`treeweights`
		`predict`, `error`, `margin`, `meanMargin`	`CompactTreeBagger`

`Mode`	`mode`	`error`, `margin`, `meanMargin`	`CompactTreeBagger`
`Weights`	`weights`	`error`, `meanMargin`	`CompactTreeBagger`
`Data` `Labels`	`data` `labels`	`outlierMeasure`, `mdsprox`	`CompactTreeBagger`
`Colors` `MDSCoordinates`	`colors` `mdscoords`	`mdsprox`	`CompactTreeBagger`

New Method Name	Old Method Name	Class
`mdsprox`	`mdsProx`	`TreeBagger`, `CompactTreeBagger`
`fillprox`	`fillProximities`	`TreeBagger`, `CompactTreeBagger`

Compatibility Considerations

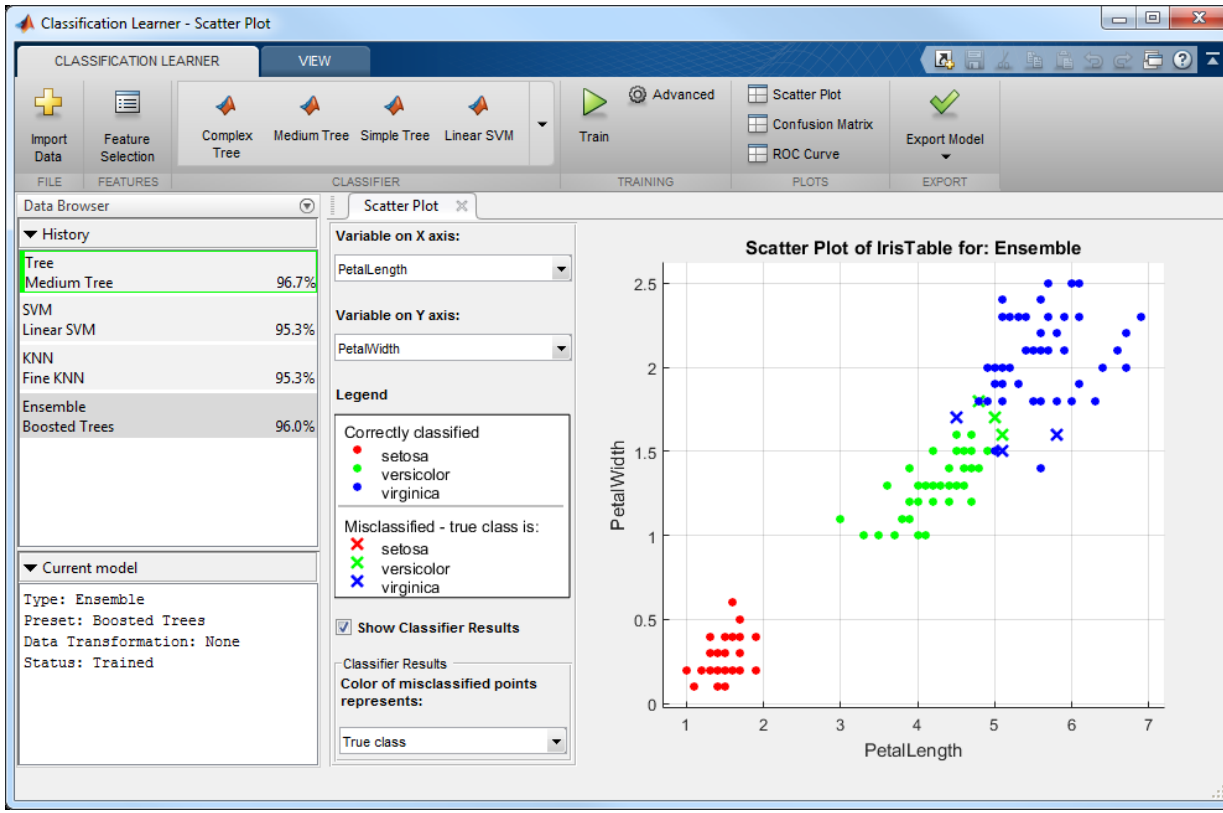
The new property, method, and name-value pair argument names do not work in the previous releases. Please refer to the documentation for the correct version of Statistics and Machine Learning Toolbox you use.

R2015a

Classification app to train models and classify data using supervised machine learning

Classification Learner is a new app that lets you train models to classify data using supervised machine learning. You can explore your data, select features, specify cross-validation schemes, train models, and assess results. You can choose from several classification types including decision trees, support vector machines, nearest neighbors, and ensemble classification.

Perform supervised machine learning by supplying a known set of input data (observations or examples) and known responses to the data (i.e., labels or classes). Use the data to train a model that generates predictions for the response to new data. To use the model with new data, or to learn about programmatic classification, you can export the model to the workspace or generate MATLAB code to recreate the trained model.



For details, see Explore Classification Models Interactively.

Statistical tests for comparing accuracies of two classification models using compareHoldout, testcholdout, and testckfold functions

You can statistically assess the predictive accuracies of two classification models using holdout sample predictions or repeated cross validation.

- The `testcholdout` accepts holdout sample predicted labels from both classification models and the true labels. This function implements the asymptotic, exact, or mid-*p* version of McNemar's test. If you specify misclassification costs, `testcholdout` compares the models using a likelihood ratio or a chi-square test.
- The `compareHoldout` object function accepts any two trained classification model objects in Statistics and Machine Learning Toolbox, sets of holdout predictor data for both models, and corresponding true labels. Like `testcholdout`, this object function implements the asymptotic, exact, or mid-*p* version of McNemar's test. If you specify misclassification costs, `compareHoldout` compares the models using a likelihood ratio or a chi-square test.
- The `testckfold` function accepts any two trained classification model objects or templates in Statistics and Machine Learning Toolbox, and repeatedly applies *k*-fold cross validation using two sets of out-of-sample predictor data and true labels. Then, `testckfold` assesses the resulting accuracies using a *t* or an *F* test.

Speedup of kmedoids, fitcknn, and other functions when using cosine, correlation, or spearman distance calculations

Pairwise distance calculations (by `pdist` and `pdist2`) in `kmedoids` and `fitcknn` use Basic Linear Algebra Subroutines (BLAS) libraries based on the Intel Math Kernel Library (MKL). For details on Intel MKL, see https://software.intel.com/en-us/intel-mkl.

Performance enhancements for decision trees and performance curves

- For dual-core systems and above, `fitctree`, `fitrtree`, and `fitensemble` parallelize training decision trees using using Intel Threading Building Blocks (TBB). For details on Intel TBB, see https://software.intel.com/en-us/intel-tbb.

- You can parallelize computation of pointwise confidence intervals `perfcurve` returns for the *x*- and *y*-coordinates, thresholds, or the area under the curve measure. You need Parallel Computing Toolbox to use this option.

Additional option to control decision tree depth using 'MaxNumSplits' argument in fitctree, fitrtree, and templateTree functions

You can control the depth of a decision tree by choosing the maximal number of splits (branch nodes) rather than choosing the minimum leaf size or minimum parent size. Specify this option using the `'MaxNumSplits'` name-value pair argument in the `fitctree`, `fitrtree`, or `templateTree`. Full trees (`ClassificationTree` or `RegressionTree` classifiers) contain the field `MaxNumSplits` in the property `ModelParameters` to store the specified maximal number of splits.

Code generation for pca and probability distribution functions (using MATLAB Coder)

`pca`, `betafit`, `betalike` and `pearsrnd` are now supported for code generation. For a full list of Statistics and Machine Learning Toolbox functions that are supported by MATLAB Coder, see Statistics and Machine Learning Toolbox.

Power and sample size for two-sample t-test using sampsizepwr function

`sampsizepwr` returns the power, sample size, or alternative hypothesis value for a two-sample *t*-test for populations with equal variances. Specify the two-sample *t*-test using `'t2'` as the `'testtype'` input variable.

Discard support vectors of SVM and ECOC models

You can reduce the memory footprint of a linear support vector machine (SVM) model by discarding their support vectors. Pass a trained SVM model (i.e., a `ClassificationSVM` or `CompactClassificationSVM` object) to `discardSupportVectors` to discard:

- The α coefficients (stored in the `Alpha` property)
- The support vectors (stored in the `SupportVectors` property)

- The support vector labels (stored in the `SupportVectorLabels` property)

By default, `fitcsvm` and `compact` do not discard the α coefficients, support vectors, and the support vector labels.

You can pass a trained error correcting output codes (ECOC) model (i.e., a `ClassificationECOC` or `CompactClassificationECOC` object) to `discardSupportVectors` to similarly discard the α coefficients, support vectors, and the support vector labels from all linear SVM binary learners. To control whether linear SVM binary learners store support vectors, create an SVM template using `templateSVM` and set the `'SaveSupportVectors'` name-value pair argument.

Compatibility Considerations

By default, `fitcecoc` discards the α coefficients, support vectors, and the support vector labels from all linear SVM binary learners. To store these estimates, create an SVM template and specify `'SaveSupportVectors',true`. Then, pass the SVM template to `fitcecoc`.

Minimum leaf size for boosted regression trees

The default minimum leaf size for boosted regression trees is `5`.

Compatibility Considerations

To train boosted regression trees using the previous defaults, construct a regression tree template using `templateTree`, and specify `'MinLeafSize',1` and `'MaxNumSplits', 1`. Then, pass the regression tree template to `fitensemble`.

Additional option to plot grouped histograms using the scatterhist and gplotmatrix functions

The `scatterhist` function includes two new name-value pair arguments that allow you to display grouped histograms of the marginal distributions along the *x*- and *y*-axes of the scatter plot:

- `'PlotGroup'` allows you to specify whether to plot the marginal distributions by group or for the entire data set.

- `'Style'` allows you to specify whether to display a stairstep plot, which shows the outline of a histogram without filling in the bars, or a histogram bar plot.

If you specify a grouping variable that contains more than one group, then by default `scatterhist` displays grouped stairstep plots. If you specify a grouping variable that contains only one group, then `scatterhist` displays a histogram bar plot. To display kernel density plots, use the `'Kernel'` name-value pair argument.

The positional argument `'dispopt'` in `gplotmatrix` supports two additional options for controlling the appearance of the plots along the diagonal of the plot matrix:

- `'stairs'` displays a stairstep plot, which shows the outline of the grouped histograms without filling in the bars.
- `'grpbars'` displays a standard grouped histogram bar plot.

Confidence interval computation for residuals using the function regress

`regress` computes the confidence intervals for studentized residuals using the degree of freedom $n – p – 1$, where n is the number of observations and p is the number of predictor variables.

Compatibility Considerations

The degrees of freedom in the computation of confidence intervals for studentized residuals that `regress` returns is $n – p – 1$ rather than $n – p$. Results may differ from those in previous releases.

Functionality Being Changed

Following functionality will be removed in a future release. Use the newer functionality instead.

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`princomp`	Warns	`pca`	Replace instances of `princomp` with `pca`.

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`treedisp`	Warns	`view` (`ClassificationTree`) or `view` (`RegressionTree`)	Use `fitctree` or `fitrtree` to grow a tree. Replace instances of `treedisp` with`view` (`ClassificationTree`) or `view` (`RegressionTree`).
`treefit`	Warns	`fitctree` or `fitrtree`	Replace instances of `treefit` with `fitctree` or `fitrtree`.
`treeprune`	Warns	`prune` (`ClassificationTree`) or `prune` (`RegressionTree`)	Use `fitctree` or `fitrtree` to grow a tree. Replace instances of `treeprune` with `prune` (`ClassificationTree`) or `prune` (`RegressionTree`) .

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`treetest`	Warns	• `resubLoss` (`ClassificationTree`) or `resubLoss` (`RegressionTree`) • `loss` (`ClassificationTree`) or `loss` (`RegressionTree`) • `cvLoss` (`ClassificationTree`) or `cvLoss` (`RegressionTree`)	Use `fitctree` or `fitrtree` to grow a tree. Replace instances of • `treetest(T,'resubstitution')` with `resubLoss` (`ClassificationTree`) or `resubLoss` (`RegressionTree`) • `treetest(T,'test',X,Y)` with `loss` (`ClassificationTree`) or `loss` (`RegressionTree`) • `treetest(T,'crossvalidate',X,Y)` with `cvLoss` (`ClassificationTree`) or `cvLoss` (`RegressionTree`)
`treeval`	Warns	`predict` (`ClassificationTree`) or `predict` (`RegressionTree`)	Use `fitctree` or `fitrtree` to grow a tree. Replace instances of `treeval` with `predict` (`ClassificationTree`) or `predict` (`RegressionTree`).
`classify`	Still runs	`fitcdiscr`	Replace instances of `classify` with `fitcdiscr`.

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`classregtree`	Still runs	`fitctree` or `fitrtree`	Replace instances of `classregtree` with `fitctree` or `fitrtree`.
`fitNaiveBayes`	Still runs	`fitcnb`	Replace instances of `fitNaiveBayes` with `fitcnb`.
`ProbDist`	Still runs	`makedist` and `fitdist`	To create and fit probability distribution objects, use `makedist` and `fitdist` instead.
`ProbDistParametric`	Still runs	`makedist` and `fitdist`	To create and fit probability distribution objects, use `makedist` and `fitdist` instead.
`ProbDistKernel`	Still runs	`makedist` and `fitdist`	To create and fit probability distribution objects, use `makedist` and `fitdist` instead.
`ProbDistUnivKernel`	Still runs	`makedist` and `fitdist`	To create and fit probability distribution objects, use `makedist` and `fitdist` instead.
`ProbDistUnivParam`	Still runs	`makedist` and `fitdist`	To create and fit probability distribution objects, use `makedist` and `fitdist` instead.
`svmclassify`	Still runs	`fitcsvm`	Replace instances of `svmclassify` with `fitcsvm`.
`svmtrain`	Still runs	`fitcsvm`	Replace instances of `svmtrain` with `fitcsvm`.

R2014b

Multiclass learning for support vector machines and other classifiers using the fitcecoc function

The `fitcecoc` function fits an error-correcting output code (ECOC) model for multiclass learning. Using training data and a coding scheme, `fitcecoc` combines a set of binary learners, such as SVM classifiers, using a coding design to create a multiclass model. You can use a supported coding scheme, or specify your own using the `designecoc` function. The new functionality also supports fitting posterior probabilities for most methods. `fitcecoc` creates an object of the new class `ClassificationECOC` or `ClassificationPartitionedECOC`.

`ClassificationECOC` is a new class for accessing and performing operations on the training data. `CompactClassificationECOC` is a new class for storing configurations of trained models without storing training data. `ClassificationPartitionedECOC` is a new class for a set of cross-validated ECOC models trained on cross-validated folds.

`ClassificationECOC` is built on the same framework as `ClassificationTree`, `ClassificationDiscriminant`, `ClassificationKNN`, and `ClassificationSVM`, so you have a variety of options and methods, including:

- Cross validation
- Resubstitution statistics
- Generalization statistics
- Weighted classification

For all methods and properties of the new objects, see the `ClassificationECOC`, `CompactClassificationECOC`, and `ClassificationPartitionedECOC` class pages.

Generalized linear mixed-effects models using the fitglme function

`GeneralizedLinearMixedModel` is a new class for fitting generalized linear mixed-effects (GLME) models. Fit GLME models using `fitglme`. You can:

- Specify GLME models using the formula notation.
- Fit GLME models for a response with conditional distribution of normal, binomial, poisson, gamma, or inverse Gaussian.
- Specify the link function using a string or a structure.

- Fit GLME models using maximum pseudo likelihood (MPL), restricted maximum pseudo likelihood (REMPL), maximum likelihood using Laplace approximation, or maximum likelihood using approximate Laplace approximation with fixed effects profiled out.
- Specify a covariance pattern for the random effects.
- Calculate estimates of the empirical Bayes predictors (EBPs) for random effects.
- Perform custom hypothesis tests on fixed effects.
- Compute confidence intervals on fixed effects, random effects, and covariance parameters.
- Examine residuals, diagnostic plots, fitted values, and design matrices.
- Compare two different models using the theoretical likelihood ratio test.
- Make predictions on new data using the fitted GLME model.
- Generate random data using the fitted GLME model at new design points.
- Refit a new GLME model based on the previously fitted model, using a new response vector.

For the properties and methods of this object, see the class page for `GeneralizedLinearMixedModel`.

Clustering that is robust to outliers using the kmedoids function

The `kmedoids` function partitions data into *k* clusters using the k-medoids algorithm. This functionality provides clustering on categorical data, clustering using arbitrary distance metrics, robustness to outliers, and scaling to large data sets.

Speedup of the kmeans and gmdistribution clustering using the kmeans++ algorithm

The `kmeans` and `fitgmdist` functions perform clustering using the k-means++ initialization algorithm. The default initialization algorithm for `kmeans` is now set to k-means++.

Fisher’s exact test for 2-by-2 contingency tables

The `fishertest` function performs Fisher’s exact test on 2-by-2 contingency tables. The new functionality is appropriate for small sample sizes.

templateEnsemble function for creating ensemble learning template

You can use the `templateEnsemble` function to create an ensemble learning template suitable for training error-correcting output code (ECOC) multiclass classifiers. In particular, you can perform multiclass classification by specifying binary learners that use the ensemble methods GentleBoost, LogitBoost, and RobustBoost.

templateSVM function for creating SVM learning template

You can use the `templateSVM` function to create an SVM learning template suitable for training error-correcting output code (ECOC) multiclass classifiers. In particular, you can perform multiclass classification by specifying binary learners that standardize predictor data or use a particular box constraint.

Standardizing training data in k-nearest neighbor classification

You can standardize the training data before fitting the model in k-nearest neighbor classification. The standardization takes into account the observation weights and missing data. You can specify this option using the `'Standardize'` name-value pair argument in the `fitcknn` function.

fitcnb function for naive Bayes classification

You can use the `fitcnb` function to train a multiclass naive Bayes model. `fitcnb` creates an object of the new class `ClassificationNaiveBayes`.

`ClassificationNaiveBayes` is a new class for accessing and performing operations on the training data. `CompactClassificationNaiveBayes` is a new class for storing configurations of trained models without storing training data.

The `fitcnb` function and `ClassificationNaiveBayes` and `CompactClassificationNaiveBayes` classes include the functionality of the

`fitNaiveBayes` function and `NaiveBayes` class. `ClassificationNaiveBayes` is built on the same framework as `ClassificationTree`, `ClassificationDiscriminant`, `ClassificationKNN`, and `ClassificationSVM`, so you have a variety of additional options and methods, including:

- Cross validation
- Resubstitution statistics
- Generalization statistics
- Weighted classification

You can also use the `templateNaiveBayes` function to create a naive Bayes classifier template suitable for training error-correcting output code (ECOC) multiclass classifiers.

For all methods and properties of the new objects, see the `ClassificationNaiveBayes` and `CompactClassificationNaiveBayes` class pages.

R2014a

Repeated measures modeling for data with multiple measurements per subject

`fitrm` is a new function for fitting models to repeated measures data, where each subject has multiple response measurements. It produces an object of the new `RepeatedMeasuresModel` class. You can:

- Perform analysis of variance for between-subjects factors using `anova`.
- Perform multivariate analysis of variance using `manova`.
- Perform hypothesis tests on the coefficients using `coeftest`.
- Perform repeated measures analysis of variance using `ranova`.
- Test for sphericity (compound symmetry) with Mauchly's test using `mauchly`.
- Plot data and estimated marginal means with optional grouping using `plot` and `plotprofile`.
- Compute summary statistics organized by group using `grpstats`.
- Perform multiple comparisons of marginal means using `multcompare`.
- Make predictions on new data with the fitted repeated measures model using `predict`.
- Generate random data with the fitted repeated measures model at new design points using `random`.

For the properties and methods of this object, see the `RepeatedMeasuresModel` class page.

fitcsvm function for enhanced performance of support vector machines (SVMs) for binary classification

You can now use the new `fitcsvm` function to train an SVM classifier for one- or two-class learning. `fitcsvm` creates an object of the new class `ClassificationSVM` or existing class `ClassificationPartitionedModel`.

`ClassificationSVM` is a new class for accessing and performing operations on the training data. `CompactClassificationSVM` is a new class for storing configurations of trained models without storing training data. The syntax and methods resemble those in the existing `ClassificationTree` and `CompactClassificationTree` classes.

The new `fitcsvm` function and `ClassificationSVM` and `CompactClassificationSVM` classes include the functionality of the `svmtrain` and

`svmclassify` functions. `ClassificationSVM` provides several benefits compared to the `svmtrain` and `svmclassify` functions:

- The new functionality
  - Supports computation of soft classification scores
  - Supports fitting posterior probabilities
  - Has improved training speed, especially on big data with well-separated classes by providing shrinkage
  - Allows a warm restart by accepting an initial α value
  - Allows training to resume after the maximum number of iterations is exceeded
  - Supports robust learning in the presence of outliers
- `ClassificationSVM` is built on the same framework as `ClassificationTree`, `ClassificationDiscriminant`, and `ClassificationKNN`, so you have a variety of options and methods, including:
  - Cross validation
  - Resubstitution statistics
  - Generalization statistics
  - Weighted classification

For all methods and properties of the new objects, see the `ClassificationSVM` and `CompactClassificationSVM` class pages.

evalclusters methods to expand the number of clusters and number of gap criterion simulations

There are two new methods for the objects created using the `evalclusters` function:

- `addK` adds additional number of clusters to be evaluated. This method applies to all classes of cluster evaluation (i.e., `clustering.evaluation.GapEvaluation`, `clustering.evaluation.SilhouetteEvaluation`, `clustering.evaluation.CalinskiHarabaszEvaluation`, and `clustering.evaluation.DaviesBouldinEvaluation`).
- `increaseB` increases the number of reference data sets for gap criterion simulations. This method applies to the `clustering.evaluation.GapEvaluation` class.

The default value of the `'SearchMethod'` name-value pair argument for `clustering.evaluation.GapEvaluation` objects is now always `'globalMaxSE'`.

Compatibility Considerations

The default value of the `'SearchMethod'` name-value pair argument for `clustering.evaluation.GapEvaluation` objects is now always `'globalMaxSE'` and does not change depending on the value of the `'KList'` name-value pair argument.

p-value output from the multcompare function

`multcompare` now returns the *p*-value of each pairwise comparison of group means. `multcompare` returns the *p*-value in the sixth column of its first output argument. The *p*-value is the overall significance level at which the individual comparison is borderline significant.

Compatibility Considerations

The first output argument of `multcompare` now has six columns, instead of five. The sixth column contains the *p*-value.

mnrfit, lassoglm, and fitglm functions accept categorical variables as responses

`mnrfit` now accepts a categorical variable as the response. The `lassoglm`, `fitglm`, and `glmfit` functions now accept a two-level categorical variable as the response. The `random` method for the `GeneralizedLinearModel` class now also returns categorical responses.

Functions accept table inputs as an alternative to dataset array inputs

The following functions and methods now accept table inputs as alternative to dataset array inputs.

Functions and Methods	Class
`fitlm`, `fitglm`, `fitlme`, `fitnlm`, `stepwiseglm`, `stepwiselm`, `grpstats`, `datasample`	N/A
`predict`, `random`, `feval`	`LinearModel`
`devianceTest`, `random`, `predict`, `feval`	`GeneralizedLinearModel`
`random`, `predict`, `feval`	`NonLinearModel`
`random`, `predict`	`LinearMixedModel`

Functions and model properties return a table rather than a dataset array

The following functions, methods, and model properties now return a `table` rather than a `dataset` array.

Functions and Methods	Class
`xptread`, `grpstats`*	N/A
`anova`	`LinearModel`
`devianceTest`	`GeneralizedLinearModel`
`fixedEffects`, `randomEffects`	`LinearMixedModel`

Property	Class
`VariableInfo`, `ObservationInfo`, `Variables`, `Diagnostics`, `Residuals`, `Coefficients`	`LinearModel`
`VariableInfo`, `ObservationInfo`, `Variables`, `Diagnostics`, `Residuals`, `Fitted`, `Coefficients`	`GeneralizedLinearModel`
`VariableInfo`, `ObservationInfo`, `Variables`, `Diagnostics`, `Residuals`, `Coefficients`	`NonLinearModel`

Property	Class
`VariableInfo`, `ObservationInfo`, `Variables`, `Coefficients`, `ModelCriterion`	`LinearMixedModel`

*`grpstats` now matches the output with input type.

Compatibility Considerations

The functions and properties listed now return a `table` instead of a `dataset` array. You can convert them to dataset arrays using the `table2dataset` function.

Default value of 'EmptyAction' on kmeans is now 'singleton'.

The default value of the `'EmptyAction'` name-value pair argument of the `kmeans` function is now `'singleton'`.

Compatibility Considerations

To set the value of `'EmptyAction'` to `'error'`, you must explicitly specify `'EmptyAction','error'`.

Functions for classification methods and clustering

The following are new functions for classification and regression trees, discriminant analysis, nearest neighbors, Naive Bayes classification, and Gaussian mixture models.

New Function	Replacing
`fitcdiscr`	`ClassificationDiscriminant.fit`
`fitcknn`	`ClassificationKNN.fit`
`fitctree`	`ClassificationTree.fit`
`fitrtree`	`RegressionTree.fit`
`fitNaiveBayes`	`NaiveBayes.fit`
`fitgmdist`	`gmdistribution.fit`

New Function	Replacing
`templateDiscriminant`	`ClassificationDiscriminant.template`
`templateKNN`	`ClassificationKNN.template`
`templateTree`	`ClassificationTree.template` or `RegressionTree.template`
`makecdiscr`	`ClassificationDiscriminant.make`

Functionality being changed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`ClassificationDiscriminant.fit`	Still runs	`fitcdiscr`	Replace instances of `ClassificationDiscriminant.fit` with `fitcdiscr`.
`ClassificationKNN.fit`	Still runs	`fitcknn`	Replace instances of `ClassificationKNN.fit` with `fitcknn`.
`ClassificationTree.fit`	Still runs	`fitctree`	Replace instances of `ClassificationTree.fit` with `fitctree`.
`RegressionTree.fit`	Still runs	`fitrtree`	Replace instances of `RegressionTree.fit` with `fitrtree`.
`NaiveBayes.fit`	Still runs	`fitNaiveBayes`	Replace instances of `NaiveBayes.fit` with `fitNaiveBayes`.
`gmdistribution.fit`	Still runs	`fitgmdist`	Replace instances of `gmdistribution.fit` with `fitgmdist`.

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`ClassificationDiscriminant.template`	Still runs	`templateDiscriminant`	Replace instances of `ClassificationDiscriminant.template` with `templateDiscriminant`.
`ClassificationKNN.template`	Still runs	`templateKNN`	Replace instances of `ClassificationKNN.template` with `templateKNN`.
`ClassificationTree.template` or `RegressionTree.template`	Still runs	`templateTree`	Replace instances of `ClassificationTree.template` or `RegressionTree.template` with `templateTree`.
`ClassificationDiscriminant.make`	Still runs	`makecdiscr`	Replace instances of `ClassificationDiscriminant.make` with `makecdiscr`.

R2013b

Linear mixed-effects models

`LinearMixedModel` is a new class for fitting linear mixed-effects (LME) models. Fit multi-level LME models or LME models with nested and/or crossed random effects using the `fitlme` or `fitlmematrix` function. You can:

- Specify LME models using either the formula notation or via matrix input.
- Fit LME models using maximum likelihood (ML) or restricted maximum likelihood (REML).
- Specify a covariance pattern for the random effects.
- Calculate estimates of best linear unbiased predictors (BLUPs) for random effects.
- Perform custom joint hypothesis tests on fixed and random effects.
- Compute confidence intervals on fixed effects, random effects, and covariance parameters.
- Examine residuals, diagnostic plots, fitted values, and design matrices.
- Compare two different models via theoretical or simulated likelihood ratio tests.
- Make predictions on new data using the fitted LME model.
- Generate random data using the fitted LME model at new design points.

For the properties and methods of this object, see the class page for `LinearMixedModel`.

Code generation for probability distribution and descriptive statistics functions (using MATLAB Coder)

Many probability distribution and descriptive statistics functions are now supported for code generation. For a full list of Statistics Toolbox™ functions that are supported by MATLAB Coder, see Statistics Toolbox Functions.

evalclusters function for estimating the optimal number of clusters in data

The new function `evalclusters` estimates the optimal number of clusters for various criterion values, and returns the clustering solution corresponding to the estimated optimal value.

You can provide clustering solutions, ask `evalclusters` to use one of the built-in clustering algorithms, `'kmeans'`, `'linkage'`, or `'gmdistribution'`, or provide a function handle.

The following criteria are available:

- The Calinski-Harabasz (CH) index
- The Silhouette index
- The Gap statistic
- The Davies-Bouldin (DB) index

mvregress function that now accepts a design matrix even if Y has multiple columns

`mvregress` now accepts an n-by-(p + 1) design matrix `X`, when the response `Y` is an n-by-d matrix with $d > 1$, where n is the number of observations, p is the number of predictor variables, d is the number of dimensions in the response, and `X` includes a column of ones for the intercept (constant) term.

Upper tail probability calculations for cumulative distribution functions

Statistics Toolbox now provides upper tail probability calculations for cumulative distribution functions. You can compute the upper tail probabilities using a trailing `'upper'` argument in the following functions:

- `cdf` function for probability distribution objects, returned by `pd = makedist(distname)` or `pd = fitdist(X,distname)`:

  ```
  cdf(pd,X,'upper')
  ```

- `cdf` function:

  ```
  Y = cdf('name',X,A,'upper')

  Y = cdf('name',X,A,B,'upper')

  Y = cdf('name',X,A,B,C,'upper')
  ```

- Distribution-specific `cdf` functions:

Distribution	New Syntax
Beta	`p = betacdf(X,A,B,'upper')`
Binomial	`Y = binocdf(X,N,P,'upper')`
Chi-square	`p = chi2cdf(X,V,'upper')`

Extreme Value	`P = evcdf(X,mu,sigma,'upper')`
	
	`[P,PLO,PUP] = evcdf(X,mu,sigma,pcov,'upper')`

Exponential	`P = expcdf(X,mu,'upper')`
	
	`[P,PLO,PUP] = expcdf(X,mu,pcov,'upper')`

F	`P = fcdf(X,V1,V2,'upper')`

Gamma	`P = gamcdf(X,A,B,'upper')`
	
	`[P,PLO,PUP] = gamcdf(X,A,B,pcov,'upper')`

Geometric	`Y = geocdf(X,P,'upper')`
Generalized Extreme Value	`P = gevcdf(X,k,sigma,mu,'upper')`
Generalized Pareto	`P = gpcdf(X,sigma,theta,'upper')`
Hypergeometric	`P = hygecdf(X,M,K,N,'upper')`

Lognormal	`P = logncdf(X,mu,sigma,'upper')`
	
	`[P,PLO,PUP] = logncdf(X,mu,sigma,pcov,'upper')`

Negative Binomial	`Y = nbincdf(X,R,P,'upper')`
Non-central F	`P = ncfcdf(X,NU1,NU2,DELTA,'upper')`
Non-central t	`P = nctcdf(X,NU,DELTA,'upper')`
Non-central Chi-square	`P = ncx2cdf(X,V,DELTA,'upper')`

Normal	`P = normcdf(X,mu,sigma,'upper')`
	
	`[P,PLO,PUP] = normcdf(X,mu,sigma,pcov,'upper')`

Distribution	New Syntax
Poisson	`P = poisscdf(X,lambda,'upper')`
t	`P = tcdf(X,V,'upper')`
Rayleigh	`P = raylcdf(X,B,'upper')`
Uniform Discrete	`P = unidcdf(X,N,'upper')`
Uniform Continuous	`P = unidcdf(X,A,B,'upper')`

Weibull	`P = wblcdf(X,A,B,'upper')`
	
	`[P,PLO,PUP] = wblcdf(X,A,B,pcov,'upper')`

partialcorri function for partial correlation with asymmetric treatment of inputs and outputs

The new function `partialcorri` computes linear partial correlation coefficients with internal adjustments. You can compute partial correlation between pairs of variables in `Y` and `X`, adjusting for the remaining variables in `X`, or between pairs of variables in `Y` and `X`, adjusting for the remaining variables in `X`, after first controlling both `X` and `Y` for the variables in `Z`.

You can also:

- Specify whether to use Pearson or Spearman partial correlations.
- Specify how to handle missing values.
- Perform hypotheses test of zero correlation against a one-sided or two-sided alternative.

Fitting functions for linear, generalized linear, and nonlinear models

There are new functions for the fitting and stepwise algorithms of linear and generalized linear models, and the fitting algorithm of nonlinear models. The new functions are as follows.

New Function	Replacing
`fitlm`	`LinearModel.fit`
`stepwiselm`	`LinearModel.stepwise`
`fitglm`	`GeneralizedLinearModel.fit`
`stepwiseglm`	`GeneralizedLinearModel.stepwise`
`fitnlm`	`NonLinearModel.fit`

Functionality being changed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`LinearModel.fit`	Still runs	`fitlm`	Replace instances of `LinearModel.fit` with `fitlm`
`LinearModel.stepwise`	Still runs	`stepwiselm`	Replace instances of `LinearModel.stepwise` with `stepwiselm`
`GeneralizedLinearModel.fit`	Still runs	`fitglm`	Replace instances of `GeneralizedLinearModel.fit` with `fitglm`
`GeneralizedLinearModel.stepwise`	Still runs	`stepwiseglm`	Replace instances of `GeneralizedLinearModel.stepwise` with `stepwiseglm`
`NonLinearModel.fit`	Still runs	`fitnlm`	Replace instances of `NonLinearModel.fit` with `fitnlm`

R2013a

Support vector machines (SVMs) for binary classification (formerly in Bioinformatics Toolbox)

Support vector machines are now in Statistics Toolbox. Train support vector machine classifier using `svmtrain` and classify data using `svmclassify`.

Probabilistic PCA and alternating least-squares algorithms for principal component analysis with missing data

Two new features handle missing data in principal component analysis:

- The new function `ppca` uses probabilistic principal components analysis, which is based on an isotropic error model.
- The function `pca` has a new alternating least squares (ALS) algorithm. Use the name-value pair argument `'algorithm'` with the value `'als'`.

Anderson-Darling goodness-of-fit test

The new function `adtest` performs the Anderson-Darling goodness-of-fit test. `adtest` can perform:

- Simple test: Test against a specific distribution with parameters specified. You can test against any continuous univariate parametric distribution.
- Composite test: Test against a specified distribution family (also called an omnibus test). You can test against the normal, exponential, extreme-value, lognormal, or weibull distribution families.

Decision-tree performance improvements and categorical predictors with many levels

- The training speed for decision trees and their ensembles is improved. The improvement is best seen in decision tree ensembles obtained using the `fitensemble` function or `TreeBagger` class.
- Improved efficiency of `TreeBagger` when used in parallel mode.
- You can specify the number of surrogate splits saved in decision trees using the `'surrogate'` name-value pair argument in the `fit` and `template` methods of the `ClassificationTree` and `RegressionTree` classes.

- `ClassificationTree.fit` and `ClassificationTree.template` provide several heuristic methods for splitting on categorical predictors with many levels. Use the `'AlgorithmForCategorical'` name-value pair argument to specify the algorithm to find the best split and the `'MaxCat'` name-value pair argument to specify the maximum number of categories you allow.

Grouping and kernel density options in scatterhist function

The `scatterhist` function has these name-value pair arguments:

- `'Group'` lets you specify a grouping variable and produces a grouped scatter plot.
- `'Kernel'` lets you use grouped kernel density plots instead of overall histograms for the marginal distributions.
- Additional options let you change colors, line properties, legends, and more.

Nonlinear model enhancements

These functions now accept additional error models and fixed or fit-dependent weights.

`NonLinearModel` methods: • `NonLinearModel.fit` • `predict` • `random`	• Use the `'ErrorModel'` and `'ErrorParameters'` name-value pair arguments to define the error models and `'Weights'` to enter weights. • The `NonLinearModel` object has a new property, `WeightedResiduals`.
`nlinfit`	• Use `'ErrorModel'` and `'ErrorParameters'` name-value pair arguments to define the error models and `'Weights'` name-value pair to enter weights. • `nlinfit` returns a structure containing information about the error model you define.
`nlpredci`	• Accepts the error model structure returned by `nlinfit`. • Adjusts Scheffe type simultaneous confidence intervals for weights, error models, and rank deficient Jacobians.

Additional functionality changes are:

- `disp` (`NonLinearModel` method) shows only estimable coefficients, and shows `NaN` for inestimable coefficients.
- Ftest (`NonLinearModel` method) automatically decides whether to compare the full model against an intercept-only model or zero.
- `NonLinearModel` properties such as `Diagnostics`, `Residuals`, `LogLikelihood`, `SSE`, and `SST` account for weights and error models.

Syntax changes in parametric hypothesis test functions

Parametric hypothesis test functions accept optional input arguments as name-value pair arguments.

`adtest`	Anderson-Darling goodness-of-fit test
`ansaribradley`	Ansari-Bradley test
`dwtest`	Durbin-Watson test
`kstest`	One-sample Kolmogorov-Smirnov test
`kstest2`	Two-sample Kolmogorov-Smirnov test
`lillietest`	Lilliefors test
`ttest`	One-sample *t*-test
`ttest2`	Two-sample *t*-test
`vartest`	One-sample variance chi-square test
`vartest2`	Two-sample variance *F*-test
`vartestn`	Variance test across multiple groups
`ztest`	*z*-test

Probability distribution enhancements

New probability distribution objects provide the following new functionality:

- Create a distribution without fitting to data using the new `makedist` function.
- Assign directly to parameter values.
- Create truncated distributions.
- Create and operate on arrays of distribution objects.

- Create custom distributions. To begin, use `dfittool` and select **Edit > Define Custom Distributions**. Use the provided template to define the `'Laplace'` distribution, or modify it to create your own.
- Compute and plot likelihood ratio confidence intervals and profile likelihood for fitted probability distributions.
- Additional distributions in the probability distribution framework:
  - Multinomial
  - Piecewise Linear
  - Triangular
  - Uniform

You can continue fitting distributions to data using the existing `fitdist` function.

Compatibility Considerations

The class names of probability distribution objects returned by `fitdist` are different than in earlier releases.

R2012b

Boosting algorithms for imbalanced data, sparse ensembles, and multiclass boosting, with self termination

There are three new boosting algorithms for classification:

- **RUSBoost** (boosting by random undersampling) for imbalanced data (data in which one class has many more observations than the other).
- **LPBoost** (linear programming) and **TotalBoost** (totally corrective boosting) which self-terminate, can lead to a sparse ensemble, and can be used for multiclass boosting.

Burr distribution for expressing a wide range of distribution shapes while preserving a single functional form for the density

There is a new probability distribution object for the Burr Type XII distribution, a three-parameter family of continuous distributions on the real line. Use `fitdist` to fit this distribution to data. Use `ProbDistUnivParam` to specify the distribution parameters directly. Either function produces a distribution you can use to generate random samples or compute functions such as `pdf` and `cdf`.

Data import to a dataset array with the MATLAB Import Tool

You can now import data from a file directly into a `dataset` array using the MATLAB Import Tool.

Principal component analysis enhancements for handling NaN as missing data, weighted PCA, and choosing between EIG or SVD as the underlying algorithm

The new `pca` function includes additional functionality for principal component analysis. Features of `pca` include:

- Handling of NaN as missing data values.
- Weighted principal component analysis with user-specified weights.
- Choice of SVD or EIG algorithm for computing principal components.
- Option to specify number of components to return.

- Option to not center before computing principal components.

Compatibility Considerations

The new `pca` function replaces the `princomp` function.

Speedup of k-means clustering using Parallel Computing Toolbox

Statistics Toolbox now supports parallel execution for `kmeans`.

One-sided nonparametric hypothesis tests

An option to test one-sided right- or left-tailed alternatives is available for these nonparametric hypothesis tests:

- `signrank`
- `ranksum`
- `signtest`

Reorder nodes in dendrogram plots

- The `dendrogram` function has new options for reordering the nodes of hierarchical binary cluster trees:
  - The `reorder` option allows you to specify a permutation vector for the order of nodes in a dendrogram plot.
  - The `checkcrossings` option checks whether a requested permutation vector leads to crossing branches in a dendrogram plot.
- The function `optimalleaforder` generates an optimal permutation of nodes.

Nonlinear model enhancements

You can add a vector of observation weights, or a handle to a function that returns a vector of observation weights, to these functions:

- `NonLinearModel.fit`.

- `predict` and `random` (`NonLinearModel` methods).
- `nlinfit` and `nlpredci`.

For an example of weighted fitting, see Weighted Nonlinear Regression.

Compatibility Considerations

Use either `Weights` or `RobustWgtFun` when performing weighted nonlinear regression.

Changes to LinearModel diagnostics

The diagnostics in the `Diagnostics` dataset array for `LinearModel` objects are in a new order, and no longer appear in the Variables editor. The new order is:

- `Leverage`
- `CooksDistance`
- `Dffits`
- `S2_i`
- `CovRatio`
- `Dfbetas`
- `HatMatrix`

Compatibility Considerations

To access the correct diagnostics, you should update any code that indexes the diagnostics dataset array columns by number.

Functionality being changed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`princomp`	Still runs	`pca`	Replace instances of `princomp` with `pca`

R2012a

Linear, Generalized Linear, and Nonlinear Models for Regression

`LinearModel` is a new class for performing linear regression. `LinearModel.fit` creates a model that:

- Lets you fit models with both categorical and continuous predictor variables
- Contains information about the quality of the fit, such as residuals and ANOVA tables
- Lets you easily plot the fit
- Allows for automatic or manual exclusion of unimportant variables
- Enables robust fitting for reduced influence of outliers
- Lets you specify quadratic and other models using a symbolic formula
- Enables stepwise model selection

There are similar improvements for generalized linear and nonlinear modeling using the `GeneralizedLinearModel` and `NonLinearModel` classes. For details, see the class reference pages in the reference material, or Linear Regression, Stepwise Regression, Robust Regression — Reduce Outlier Effects, Generalized Linear Regression, or Nonlinear Regression in the User's Guide.

Variable Editor for Dataset Arrays

You can now edit, sort, plot, and select portions of dataset arrays from the MATLAB Variable Editor. For details, see Using Dataset Arrays in the User's Guide.

Lasso for Generalized Linear Regression

The `lassoglm` function regularizes generalized linear models. Use `lassoglm` to examine model alternatives and to constrain or remove redundant or unimportant variables in generalized linear regression. For details, see the function reference page, or Lasso Regularization of Generalized Linear Models in the User's Guide.

K-Nearest Neighbor Classification

`ClassificationKNN.fit` creates a classification model that performs *k*-nearest neighbor classification. You can check the quality of the model with cross validation or

resubstitution. For details, see the `ClassificationKNN` page in the reference material, or Classification Using Nearest Neighbors in the User's Guide.

Random Subspace Ensembles

`fitensemble` can construct random subspace ensembles to improve the classification accuracy of both *k*-nearest neighbor classifiers and discriminant analysis classifiers. For details, see Ensemble Methods or Random Subspace Classification in the User's Guide.

Regularized Discriminant Analysis with Variable Selection

`ClassificationDiscriminant` models now have two parameters, `Gamma` and `Delta`, for regularization and lowering the number of variables. Set `Gamma` to regularize the discriminant. Set `Delta` to eliminate variables. Use `cvshrink` to obtain optimal `Gamma` and `Delta` parameters by cross validation. For details, see the reference pages, or Regularize a Discriminant Analysis Classifier in the User's Guide.

stepwisefit Coefficient History

The `stepwisefit` function now returns the fitted coefficient history in the `history.B` field.

RobustWgtFun Replaces WgtFun

The `WgtFun` option is now called `RobustWgtFun` in the `nlinfit`, `statget`, and `statset` functions. `RobustWgtFun` also makes the `Robust` option superfluous.

Compatibility Considerations

The `WgtFun` and `Robust` options are currently accepted by all functions. To avoid potential future incompatibilities, update code that uses the `WgtFun` and `Robust` options to use the `RobustWgtFun` option.

ClassificationTree Now Predicts Class with Minimal Misclassification Cost

The `ClassificationTree` `predict` method now chooses the class with minimal expected misclassification cost. Previously, it chose the class with maximal posterior

probability. The new behavior is consistent with the `cvLoss` method. Furthermore, both `ClassificationDiscriminant` and `ClassificationKNN` predict using minimal expected misclassification cost. For details, see `predict` and `loss`.

Compatibility Considerations

If you use a nondefault cost matrix, some `ClassificationTree` classification predictions can differ from those in previous versions.

fpdf Improvements

The `fpdf` function now accepts a wider range of parameter values, including `Inf`.

R2011b

Lasso Regularization for Linear Regression

The `lasso` function incorporates both the lasso regularization algorithm and the elastic net regularization algorithm. Use `lasso` to remove redundant or unimportant variables in linear regression. The `lassoPlot` function helps you visualize `lasso` results, with a variety of coefficient trace plots and a cross-validation plot.

For details, see Lasso and Elastic Net.

Discriminant Analysis Classification Object

You can now use the `ClassificationDiscriminant` and `CompactClassificationDiscriminant` classes for classification via discriminant analysis. The syntax and methods resemble those in the existing `ClassificationTree` and `CompactClassificationTree` classes. The `ClassificationDiscriminant` class includes the functionality of the `classify` function. `ClassificationDiscriminant` provides several benefits compared to the `classify` function:

- After you fit a classifier, you can predict without refitting.
- `ClassificationDiscriminant` is built on the same framework as `ClassificationTree`, so you have a variety of options and methods, including:
  - Cross validation
  - Resubstitution statistics
  - A choice of cost functions
  - Weighted classification
- `ClassificationDiscriminant` can fit several models, including linear, quadratic, and linear or quadratic with pseudoinverse.

For details, see Discriminant Analysis.

Nearest Neighbor Searching for Points Within a Fixed Distance

The `rangesearch` function finds all members of a data set that are within a specified distance of members of another data set. As with the `knnsearch` function, you can set a variety of distance metrics, or program your own. `rangesearch` has counterparts that are methods of the `ExhaustiveSearcher` and `KDTreeSearcher` classes.

datasample Function for Random Sampling

The `datasample` function samples with or without replacement from a data set. It can also perform weighted sampling, with or without replacement.

Fractional Factorial Design Improvements

The `fracfactgen` function now allows up to `52` factors, instead of the previous limit of `26` factors. Specify factors as case-sensitive strings, using `'a'` through `'z'` for the first `26` factors, and `'A'` through `'Z'` for the remaining factors.

`fracfact` now checks for an arbitrary level of interaction in confounding, instead of the previous limit of confounding up to products of two factors. Set the `MaxInt` name-value pair to the level of interaction you want. You can also set names for the factors using the `FactorNames` name-value pair.

nlmefit Returns the Covariance Matrix of Estimated Coefficients

The `nlmefit` function now returns the covariance matrix of the estimated coefficients as the `covb` field of the `stats` structure.

signrank Change

The `signrank` test now defines ties to be entries that differ by `2*eps` or less. Previously, ties were entries that were identical to machine precision.

Conversion of Error and Warning Message Identifiers

For R2011b, error and warning message identifiers have changed in Statistics Toolbox.

Compatibility Considerations

If you have scripts or functions that use message identifiers that changed, you must update the code to use the new identifiers. Typically, message identifiers are used to turn off specific warning messages, or in code that uses a `try/catch` statement and performs an action based on a specific error identifier.

For example, if you use the `'resubstitution'` method, the `'stats:plsregress:InvalidMCReps'` identifier has changed to `'stats:plsregress:InvalidResubMCReps'`. If you use the `'resubstitution'` method and your code checks for `'stats:plsregress:InvalidMCReps'`, you must update it to check for `'stats:plsregress:InvalidResubMCReps'` instead.

To determine the identifier for a warning, run the following command just after you see the warning:

```
[MSG,MSGID] = lastwarn;
```

This command saves the message identifier to the variable `MSGID`.

To determine the identifier for an error, run the following command just after you see the error:

```
exception = MException.last;
MSGID = exception.identifier;
```

Tip Warning messages indicate a potential issue with your code. While you can turn off a warning, a suggested alternative is to change your code so it runs warning free.

R2011a

Version: 7.5

New Features

Bug Fixes

Boosted Decision Trees for Classification and Regression

The new `fitensemble` function constructs ensembles of decision trees. It provides:

- Several popular boosting algorithms (`AdaBoostM1`, `AdaBoostM2`, `GentleBoost`, `LogitBoost`, and `RobustBoost`) for classification
- Least-squares boosting (`LSBoost`) for regression
- Most `TreeBagger` functionality for ensembles of bagged decision trees

There is also an improved interface for classification trees (`ClassificationTree`) and regression trees (`RegressionTree`), encompassing the functionality of `classregtree`.

For details, see Ensemble Methods.

Memory and Performance Improvements in Linkage Methods

The `linkage` and `clusterdata` functions have a new `savememory` option that can use less memory than before. With `savememory` set to `'on'`, the functions do not build a pairwise distance matrix, so use less memory and, depending on problem size, can use less time. You can use the `savememory` option when:

- The linkage `method` is `'ward'`, `'centroid'`, or `'median'`
- The linkage distance `metric` is `'euclidean'` (default)

For details, see the `linkage` and `clusterdata` function reference pages.

Conditional Weighted Residuals and Derivative Step Control in nlmefit and nlmefitsa

The `nlmefit` and `nlmefitsa` functions now provide the conditional weighted residuals of the fit. Use this information to assess the quality of the model; see Example: Examining Residuals for Model Verification.

The `statset` `Options` structure now includes `'DerivStep'`, which enables you to set finite differences for gradient estimation.

Detecting Ties in k-Nearest Neighbor Search

`knnsearch` now optionally returns all *k*th nearest neighbors of points, instead of just one. The `knnsearch` methods for `ExhaustiveSearcher` and `KDTreeSearcher` also have this option.

Distribution Fitting Tool Uses fitdist Function

MATLAB functions generated with the Distribution Fitting Tool now use the `fitdist` function to create fitted probability distribution objects. The generated functions return probability distribution objects as output arguments.

Speed and Accuracy Improvements in Noncentral Chi-Square CDF

`ncx2cdf` is now faster and more accurate for large values of the noncentrality parameter.

Perfect Separation in Binomial Regression

If the two categories in a binomial regression model (such as `logit` or `probit`) are perfectly separated, the best-fitting model is degenerate with infinite coefficients. In this case, the `glmfit` function is likely to exceed its iteration limit. `glmfit` now tries to detect this perfect separation and display a diagnostic message.

Sign Convention in mdscale

`mdscale` now enforces that, in each column of the output `Y`, the value with the largest magnitude has a positive sign. This change makes results consistent across releases and platforms—small changes used to lead to sign reversals.

Demo of Credit Rating Classification Via Bagged Decision Trees

The credit rating demo that used to be exclusively in Financial Toolbox™ is now available in Statistics Toolbox. The demo uses bagged decision trees for classifying creditworthiness.

To view the demo at the MATLAB command line, enter:

```
showdemo creditratingdemo
```

R2010b

Parallel Computing Support for More Functions

Statistics Toolbox now supports parallel execution for the following functions:

- `candexch`
- `cordexch`
- `daugment`
- `dcovary`
- `nnmf`
- `plsregress`
- `rowexch`
- `sequentialfs`

For more information, see the Parallel Statistics chapter in the User's Guide.

Algorithm to Rank Features in Classification and Regression

New filter algorithm, `relieff`, is based on nearest neighbors. The ReliefF algorithm accounts for correlations among predictors by computing the effect of every predictor on the class label (or true response for regression) locally and then integrates these local estimates over the entire predictor space.

nlmefit Support for Error Models, and nlmefitsa changes

`nlmefit` now supports the following error models:

- `combined`
- `constant`
- `exponential`
- `proportional`

You can specify an error model with both `nlmefitsa` and `nlmefit`.

The `nlmefit` `bic` calculation has changed. Now the degrees of freedom value is based on the number of groups rather than the number of observations. This conforms with the `bic` definition used by the `nlmefitsa` function.

Both `nlmefit` and `nlmefitsa` now store the estimated error parameters in the `errorparm` field of the output `stats` structure. The `rmse` field of the structure now contains the root mean squared residual for all error models; this value is computed on the log scale for the `exponential` model.

Compatibility Considerations

In the previous release, the `rmse` field was used by `nlmefitsa` for both mean squared residual and the estimated error parameter. Change your code, if necessary, to address the appropriate field in the `stats` structure.

As described in “nlmefit Support for Error Models, and nlmefitsa changes” on page 16-2, `nlmefit` now calculates different `bic` values than in previous releases.

Surrogate Splits for Decision Trees

The new surrogate splits feature in `classregtree` allows for better handling of missing values, more accurate estimation of variable importance, and calculation of the predictive measure of association between variables.

New Bagged Decision Tree Properties

`TreeBagger` and `CompactTreeBagger` classes have two new properties:

- `NVarSplit` provides the number of decision splits for each predictor variable.
- `VarAssoc` provides a measure of association between pairs of predictor variables.

Enhanced Cluster Analysis Performance

The `linkage` function has improved performance for the `centroid`, `median`, and `single` linkage methods.

The `linkage` and `pdist` hierarchical cluster analysis functions support larger array dimensions with 64-bit platforms, so can handle larger problems.

Export Probability Objects with dfittool

The distribution fitting GUI (`dfittool`) now allows you to export fits to the MATLAB workspace as probability distribution fit objects. For more information, see Modeling Data Using the Distribution Fitting Tool.

Compatibility Considerations

If you load a distribution fitting session that was created with previous versions of Statistics Toolbox, you cannot save an existing fit. Fit the distribution again to enable saving.

Compute Partial Correlation of Two Variables Correcting for All Other Variables

`partialcorr` now accepts a new syntax, `RHO = partialcorr(X)`, which returns the sample linear partial correlation coefficients between pairs of variables in `X`, controlling for the remaining variables in `X`. For more information, see the function reference page.

Specify Number of Evenly Spaced Quantiles

`quantile` now accepts a new syntax, `Y = quantile(X,N,...)`, which returns quantiles at the cumulative probabilities (1:`N`)/(`N`+1) where `N` is a scalar positive integer value.

Control Location and Orientation of Marginal Histograms with scatterhist

`scatterhist` now accepts three parameter name/value pairs that control where and how the histogram plots appear. The new parameter names are `NBins`, `Location`, and `Direction`. For more information, see the function reference page.

Return Bootstrapped Statistics with bootci

`bootci` has a new output option which returns the bootstrapped statistic computed for each of the `NBoot` bootstrap replicate samples. For more information, see the function reference page.

R2010a

Version: 7.3

New Features

Bug Fixes

Stochastic Algorithm Functionality in NLME Models

New stochastic algorithm for fitting NLME models is more robust with respect to starting values, enables parameter transformations, and relaxes assumption of constant error variance. See `nlmefitsa`.

k-Nearest Neighbor Searching

New functions for *k*-Nearest Neighbor (*k*NN) search efficiently to find the closest points to any query point. For information, see k-Nearest Neighbor Search and Radius Search.

Confidence Intervals Option in perfcurve

A new option in the `perfcurve` function computes confidence intervals for classifier performance curves.

Observation Weights Options in Resampling Functions

New options to weight resampling probabilities broaden the range of models supported by `bootstrp`, `bootci`, and `perfcurve` functions.

R2009b

Version: 7.2

New Features

Bug Fixes

New Parallel Computing Support for Certain Functions

Statistics Toolbox now supports parallel execution for the following functions:

- `bootci`
- `bootstrp`
- `crossval`
- `jackknife`
- `TreeBagger`

For more information on parallel computing in the Statistics Toolbox, see Parallel Computing Support for Resampling Methods.

New Stack and Unstack Methods for Dataset Arrays

`dataset.unstack` converts a “tall” dataset array to an equivalent dataset array that is in "wide format", by "unstacking" a single variable in the tall dataset array into multiple variables in wide. `dataset.stack` reverses this manipulation by converting a “wide” dataset array to an equivalent dataset array that is in "tall format", by "stacking up" multiple variables in the wide dataset array into a single variable in tall.

New Support for SAS Transport (.xpt) Files

Statistics Toolbox now supports importing and exporting files in SAS Transport (.xpt) format. For more information, see the `xptread` and `dataset.export` reference pages.

New Output Function in nlmefit for Monitoring or Canceling Calculations

The `nlmefit` function now supports using an output function to monitor or cancel calculations. For more information, see the `nlmefit` reference page.

R2009a

Enhanced Dataset Functionality

- An enhanced `dataset.join` method provides additional types of join operations:
  - `join` can now perform more complicated inner and outer join operations that allow a many-to-many correspondence between dataset arrays `A` and `B`, and allow unmatched observations in either `A` or `B`.
  - `join` can be of `Type` `'inner'`, `'leftouter'`, `'rightouter'`, `'fullouter'`, or `'outer'` (which is a synonym for `'fullouter'`). For an inner join, the dataset array, `C`, only contains observations corresponding to a combination of key values that occurred in both `A` and `B`. For a left (or right) outer join, `C` also contains observations corresponding to keys in `A` (or `B`) that did not match any in `B` (or `A`).
  - `join` can now return index vectors indicating the correspondence between observations in `C` and those in `A` and `B`.
  - `join` now supports using multiple keys.
  - `join` now supports an optional parameter for specifying missing key behavior rather than raising an error.
- An enhanced `dataset.export` method now supports exporting directly to Microsoft® Excel® files.

New Naïve Bayes Classification

- The `NaiveBayes` classification object is suitable for data sets that contain many predictors or features.
- It supports normal, kernel, multinomial, and multivariate multinomial distributions.

New Ensemble Methods for Classification and Regression Trees

- New classification objects, `TreeBagger` and `CompactTreeBagger`, provide improved performance through bootstrap aggregation (bagging).
- Includes Breiman's "random forest" method.
- Enhanced `classregtree` has more options for growing and pruning trees.

New Performance Curve Function

- New `perfcurve` function provides graphical method to evaluate classification results.
- Includes ROC (receiver operating characteristic) and other curves.

New Probability Distribution Objects

- Provides a consistent interface for working with probability distributions.
- Can be created directly using the `ProbDistUnivParam` constructor, or fit to data using the `fitdist` function.
- Option to fit distributions by group.
- Includes kernel object methods and parametric object methods that you can use to analyze the distribution represented by the object.
- Includes kernel object properties and parametric object properties that you can access to determine the fit results and evaluate their accuracy.
- Related enhancements in the `chi2gof`, `histfit`, `kstest`, `probplot`, and `qqplot` functions.

R2008b

Version: 7.0

New Features

Compatibility Considerations

Classification

The new `confusionmat` function tabulates misclassifications by comparing known and predicted classes of observations.

Data Organization

Dataset arrays constructed by the `dataset` function can now be written to an external text file using the new `export` function.

When reading external text files into a dataset array, `dataset` has a new `'TreatAsEmpty'` parameter for specifying strings to be treated as empty.

Compatibility Considerations

In previous versions, `dataset` used `eval` to evaluate strings in external text files before writing them into a dataset array. As a result, strings such as `'1/1/2008'` were treated as numerical expressions with two divides. Now, `dataset` treats such expressions as strings, and writes a string variable into the dataset array whenever a column in the external file contains a string that does not represent a valid scalar value.

Model Assessment

The cross-validation function, `crossval`, has new options for directly specifying loss functions for mean-squared error or misclassification rate, without having to provide a separate function M-file.

Multivariate Methods

The `procrustes` function has new options for computing linear transformations without scale or reflection components.

Probability Distributions

The multivariate normal functions `mvnpdf`, `mvncdf`, and `mvnrnd` now accept vector specification of diagonal covariance matrices, with corresponding gains in computational efficiency.

The hypergeometric distribution has been added to both the `disttool` and `randtool` graphical user interfaces.

Compatibility Considerations

The `ksdensity` function may give different answers for the case where there are censoring times beyond the last observed value. In this case, `ksdensity` tries to reduce the bias in its density estimate by folding kernel functions across a folding point so that they do not extend into the area that is completely censored. Two things have changed for this release:

1. In previous releases the folding point was the last observed value. In this release it is the first censoring time after the last observed value.
2. The folding procedure is applied not just when the `'function'` parameter is `'pdf'`, but for all `'function'` values.

Regression Analysis

The new `nlmefit` function fits nonlinear mixed-effects models to data with both fixed and random sources of variation. Mixed-effects models are commonly used with data over multiple groups, where measurements are correlated within groups but independent between groups.

Statistical Visualization

The `boxplot` function has new options for handling multiple grouping variables and extreme outliers.

The `lsline`, `gline`, `refline`, and `refcurve` functions now work with scatter plots produced by the `scatter` function. In previous versions, these functions worked only with scatter plots produced by the `plot` function.

The following visualization functions now have custom data cursors, displaying information such as observation numbers, group numbers, and the values of related variables:

- `andrewsplot`
- `biplot`

- `ecdf`
- `glyphplot`
- `gplotmatrix`
- `gscatter`
- `normplot`
- `parallelcoords`
- `probplot`
- `qqplot`
- `scatterhist`
- `wblplot`

Compatibility Considerations

Changes to `boxplot` have altered a number of default behaviors:

- Box labels are now drawn as text objects rather than tick labels. Any code that customizes the box labels by changing tick marks should now set the tick locations as well as the tick labels.
- The function no longer returns a handles array with a fixed number handles, and the order and meaning of the handles now depends on which options are selected. To locate a handle of interest, search for its `'Tag'` property using `findobj`. `'Tag'` values for box plot components are listed on the `boxplot` reference page.
- There are now valid handles for outliers, even when boxes have no outliers. In previous releases, the handles array returned by the function had `NaN` values in place of handles when boxes had no outliers. Now the `'xdata'` and `'ydata'` for outliers are `NaN` when there are no outliers.
- For small groups, the `'notch'` parameter sometimes produces notches that extend outside of the box. In previous releases, the notch was truncated to the extent of the box, which could produce a misleading display. A new value of `'markers'` for this parameter avoids the display issue.

As a consequence, the `anova1` function, which displays notched box plots for grouped data, may show notches that extend outside the boxes.

Utility Functions

The statistics options structure created by `statset` now includes a `Jacobian` field to specify whether or not an objective function can return the Jacobian as a second output.

R2008a

Descriptive Statistics

Bootstrap confidence intervals computed by `bootci` are now more accurate for lumpy data.

Compatibility Considerations

The formula for `bootci` confidence intervals of type `'bca'` or `'cper'` involves the proportion of bootstrap statistics less than the observed statistic. The formula now takes into account cases where there are many bootstrap statistics exactly equal to the observed statistic.

Model Assessment

Two new cross-validation functions, `cvpartition` and `crossval`, partition data and assess models in regression, classification, and clustering applications.

Multivariate Methods

A new sequential feature selection function, `sequentialfs`, selects predictor subsets that optimize user-defined prediction criteria.

The new `nnmf` function performs nonnegative matrix factorization (NMF) for dimension reduction.

Probability Distributions

The new `sobolset` and `haltonset` functions produce quasi-random point sets for applications in Monte Carlo integration, space-filling experimental designs, and global optimization. Options allow you to skip, leap over, and scramble the points. The `qrandstream` function provides corresponding quasi-random number streams for intermittent sampling.

Regression Analysis

The new `plsregress` function performs partial least-squares regression for data with correlated predictors.

Statistical Visualization

The `normspec` function now shades regions of a normal density curve that are either inside or outside specification limits.

Utility Functions

The statistics options structure created by `statset` now includes fields for `TolTypeFun` and `TolTypeX`, to specify tolerances on objective functions and parameter values, respectively.

R2007b

Cluster Analysis

The new `gmdistribution` class represents Gaussian mixture distributions, where random points come from different multivariate normal distributions with certain probabilities. The `gmdistribution` constructor creates mixture models with specified means, covariances, and mixture proportions, or by fitting a mixture model with a specified number of components to data. Methods for the class include:

- `fit` — Distribution fitting function
- `pdf` — Probability density function
- `cdf` — Cumulative distribution function
- `random` — Random number generator
- `cluster` — Data clustering
- `posterior` — Cluster posterior probabilities
- `mahal` — Mahalanobis distance

The `cluster` function for hierarchical clustering now accepts a vector of cutoff values, and returns a matrix of cluster assignments, with one column per cutoff value.

Compatibility Considerations

The `kmeans` function now returns a vector of cluster indices of length *n*, where *n* is the number of rows in the input data matrix `X`, even when `X` contains `NaN` values. In the past, rows of `X` with `NaN` values were ignored, and the vector of cluster indices was correspondingly reduced in size. Now the vector of cluster indices contains `NaN` values where rows have been ignored, consistent with other toolbox functions.

Design of Experiments

A new option in the *D*-optimal design function `candexch` specifies fixed design points in the row-exchange algorithm. A similar feature is already available for the `daugment` function, which uses the coordinate-exchange algorithm.

Hypothesis Tests

The `kstest` function now uses a more accurate method to calculate the *p*-value for a single-sample Kolmogorov-Smirnov test.

Compatibility Considerations

`kstest` now compares the computed *p*-value to the desired cutoff, rather than comparing the test statistic to a table of values. Results may differ from those in previous releases, especially for small samples in two-sided tests where an asymptotic formula was used in the past.

Probability Distributions

A new fitting function, `copulafit`, has been added to the family of functions that describe dependencies among variables using copulas. The function fits parametric copulas to data, providing a link between models of marginal distributions and models of data correlations.

A number of probability functions now have improved accuracy, especially for extreme parameter values. The functions are:

- `betainv` — More accurate for probabilities in `P` near 1.
- `binocdf` — More efficient and less likely to run out of memory for large values in `X`.
- `binopdf` — More accurate when the probabilities in `P` are on the order of `eps`.
- `fcdf` — More accurate when the parameter ratios `V2./V1` are much less than the values in `X`.
- `ncx2cdf` — More accurate in some extreme cases that previously returned `0`.
- `poisscdf` — More efficient and less likely to run out of memory for large values in `X`.
- `tcdf` — More accurate when the squares of the values in `X` are much less than the parameters in `V`.
- `tinv` — More accurate when the probabilities in `P` are very close to 0.5 and the outputs are very small in magnitude.

Function-style syntax for `paretotails` objects has been removed.

Compatibility Considerations

The changes to the probability functions listed above may lead to different, but more accurate, outputs than in previous releases.

In previous releases, syntax of the form `obj(x)` for a `paretotails` objects `obj` invoked the `cdf` method. This syntax now produces a warning. To evaluate the cumulative distribution function, use the syntax `cdf(obj,x)`.

Regression Analysis

The new `corrcov` function converts a covariance matrix to the corresponding correlation matrix.

The `mvregress` function now supports an option to force the estimated covariance matrix to be diagonal.

Compatibility Considerations

In previous releases the `mvregress` function, when using the `'cwls'` algorithm, estimated the covariance of coefficients `COVB` using the estimated, rather than the initial, covariance of the responses `SIGMA`. The initial `SIGMA` is now used, and `COVB` differs to a degree dependent on the difference between the initial and final estimates of `SIGMA`.

Statistical Visualization

The `boxplot` function has a new `'compact'` plot style suitable for displaying large numbers of groups.

R2007a

Data Organization

New categorical and dataset arrays are available for organizing and processing statistical data.

- Categorical arrays facilitate the use of nominal and ordinal categorical data.
- Dataset arrays provide a natural way to encapsulate heterogeneous statistical data and metadata, so that it can be accessed and manipulated using familiar methods analogous to those for numerical matrices.
- Categorical and dataset arrays are supported by a variety of new functions for manipulating the encapsulated data.
- Categorical arrays are now accepted as input arguments in all Statistics Toolbox functions that make use of grouping variables.

Hypothesis Testing

Expanded options are available for linear hypothesis testing.

- The new `linhyptest` function performs linear hypothesis tests on parameters such as regression coefficients. These tests have the form `H*b = c` for specified values of `H` and `c`, where `b` is a vector of unknown parameters.
- The `covb` output from `regstats` and the `SIGMA` output from `nlinfit` are suitable for use as the covariance matrix input argument required by `linhyptest`. The following functions have been modified to return a `covb` output for use with `linhyptest`: `coxphfit`, `glmfit`, `mnrfit`, `robustfit`.

Multivariate Statistics

The new `cholcov` function computes a Cholesky-like decomposition of a covariance matrix, even if the matrix is not positive definite. Factors are useful in many of the same ways as Cholesky factors, such as imposing correlation on random number generators.

The `classify` function for discriminant analysis has been improved.

- The function now computes the coefficients of the discriminant functions that define boundaries between classification regions.
- The output of the function is now of the same type as the input grouping variable `group`.

Compatibility Considerations

The `classify` function now returns outputs of different type than it did in the past. If the input argument `group` is a logical vector, output is now converted to a logical vector. In the past, output was returned as a cell array of `0`s and `1`s. If `group` is numeric, the output is now converted to the same type. For example, if `group` is of type `uint8`, the output will be of type `uint8`.

Probability Distributions

New `paretotails` objects are available for modeling distributions with an empirical cdf or similar distribution in the center and generalized Pareto distributions in the tails.

- The `paretotails` function converts a data sample to a `paretotails` object. The objects are useful for generating random samples from a distribution similar to the data, but with tail behavior that is less discrete than the empirical distribution.
- Objects from the `paretotails` class are supported by a variety of new methods for working with the piecewise distribution.
- The `paretotails` class provides function-like behavior, so that `p(x)` evaluates the cdf of `p` at values `x`.

Regression Analysis

The new `mvregresslike` function is a utility related to the `mvregress` function for fitting regression models to multivariate data with missing values. The new function computes the objective (log likelihood) function, and can also compute the estimated covariance matrix for the parameter estimates.

New `classregtree` objects are available for creating and analyzing classification and regression trees.

- The `classregtree` function fits a classification or regression tree to training data. The objects are useful for predicting response values from new predictors.
- Objects from the `classregtree` class are supported by a variety of new methods for accessing information about the tree.
- The `classregtree` class provides function-like behavior, so that `t(X)` evaluates the tree `t` at predictor values in `X`.

- The following functions now create or operate on objects from the new `classregtree` class: `treefit`, `treedisp`, `treeval`, `treefit`, `treeprune`, `treetest`.

Compatibility Considerations

Objects from the `classregtree` class are intended to be compatible with the structure arrays that were produced in previous versions by the classification and regression tree functions listed above. In particular, `classregtree` supports dot indexing of the form `t.property` to obtain properties of the object `t`. The class also provides function-like behavior through parenthesis indexing, so that `t(x)` uses the tree `t` to classify or compute fitted values for predictors `x`, rather than index into `t` as a structure array as it did in the past. As a result, cell arrays should now be used to aggregate `classregtree` objects.

Statistical Visualization

The new `scatterhist` function produces a scatterplot of 2D data and illustrates the marginal distributions of the variables by drawing histograms along the two axes. The function is also useful for viewing properties of random samples produced by functions such as `copularnd`, `mvnrnd`, and `lhsdesign`.

Other Improvements

- The `mvtrnd` function now produces a single random sample from the multivariate *t* distribution if the `cases` input argument is absent.
- The `zscore` function, which centers and scales input data by mean and standard deviation, now returns the means and standard deviations as additional outputs.

R2006b

Demos

The following demo has been updated:

- Selecting a Sample Size — Modified to highlight the new `sampsizepwr` function

Design of Experiments

The following visualization functions, commonly used in the design of experiments, have been added:

- `interactionplot` — Two-factor interaction plot for the mean
- `maineffectsplot` — Main effects plot for the mean
- `multivarichart` — Multivari chart for the mean

Hypothesis Tests

The following functions for hypothesis testing have been added or improved:

- `jbtest` — Replaces the chi-square approximation of the test statistic, which is asymptotic, with a more accurate algorithm that interpolates *p*-values from a table of quantiles. A new option allows you to run Monte Carlo simulations to compute *p*-values outside of the table.
- `lillietest` — Uses an improved version of Lilliefors' table of quantiles, covering a wider range of sample sizes and significance levels, with more accurate values. New options allow you to test for exponential and extreme value distributions, as well as normal distributions, and to run Monte Carlo simulations to compute *p*-values outside of the tables.
- `runstest` — Adds a test for runs up and down to the existing test for runs above or below a specified value.
- `sampsizepwr` — New function to compute the sample size necessary for a test to have a specified power. Options are available for choosing a variety of test types.

Compatibility Considerations

If the significance level for a test lies outside the range of tabulated values, [0.001, 0.5], then both `jbtest` and `lillietest` now return an error. In previous versions, `jbtest` returned an approximate *p*-value and `lillietest` returned an error outside a smaller

range, [0.01, 0.2]. Error messages suggest using the new Monte Carlo option for computing values outside the range of tabulated values.

If the data sample for a test leads to a *p*-value outside the range of tabulated values, then both `jbtest` and `lillietest` now return, with a warning, either the smallest or largest tabulated value. In previous versions, `jbtest` returned an approximate *p*-value and `lillietest` returned `NaN`.

Multinomial Distribution

The multinomial distribution has been added to the list of almost 50 probability distributions supported by the toolbox.

- `mnpdf` — Multinomial probability density function
- `mnrnd` — Multinomial random number generator

Regression Analysis

Multinomial Regression

Support has been added for multinomial regression modeling of discrete multi-category response data, including multinomial logistic regression. The following new functions supplement the regression models in `glmfit` and `glmval` by providing for a wider range of response values:

- `mnrfit` — Fits a multinomial regression model to data
- `mnrval` — Computes predicted probabilities for the multinomial regression model

Multivariate Regression

The new `mvregress` function carries out multivariate regression on data with missing response values. An option allows you to specify how missing data is handled.

Survival Analysis

`coxphfit` — A new option allows you to specify the values at which the baseline hazard is computed.

Statistical Process Control

The following new functions consolidate and expand upon existing functions for statistical process control:

- `capability` — Computes a wider range of probabilities and capability indices than the `capable` function found in previous releases
- `controlchart` — Displays a wider range of control charts than the `ewmaplot`, `schart`, and `xbarplot` functions found in previous releases
- `controlrules` — Supplements the new `controlchart` function by providing for a wider range of control rules (Western Electric and Nelson)
- `gagerr` — Performs a gage repeatability and reproducibility study on measurements grouped by operator and part

Compatibility Considerations

The `capability` function subsumes the `capable` function that appeared in previous versions of Statistics Toolbox software, and the `controlchart` function subsumes the functions `ewmaplot`, `schart`, and `xbarplot`. The older functions remain in the toolbox for backwards compatibility, but they are no longer documented or supported.

R2006a

Version: 5.2

New Features

Bug Fixes

Analysis of Variance

Support for nested and continuous factors has been added to the `anovan` function for *N*-way analysis of variance.

Bootstrapping

The following functions have been added to supplement the existing `bootstrp` function for bootstrap estimation:

- `bootci` — Computes confidence intervals of a bootstrapped statistic. An option allows you to choose the type of the bootstrap confidence interval.
- `jackknife` — Draws jackknife samples from a data set and computes statistics on each sample

Demos

The following demos have been added to the toolbox:

- Bayesian Analysis for a Logistic Regression Model
- Time Series Regression of Airline Passenger Data

The following demo has been updated to demonstrate new features:

- Random Number Generation

Design of Experiments

The new `fracfactgen` function finds a set of fractional factorial design generators suitable for fitting a specified model.

The following functions for *D*-optimal designs have been enhanced:

- `cordexch`, `daugment`, `dcovary`, `rowexch` — New options specify the range of values and the number of levels for each factor, exclude factor combinations, treat factors as categorical rather than continuous, control the number of iterations, and repeat the design generation process from random starting points
- `candexch` — New options control the number of iterations and repeat the design generation process from random starting points

- `candgen` — New options specify the range of values and the number of levels for each factor, and treat factors as categorical rather than continuous
- `x2fx` — New option treats factors as categorical rather than continuous

Hypothesis Tests

The new `dwtest` function performs a Durbin-Watson test for autocorrelation in linear regression.

Multivariate Distributions

Two new functions have been added to compute multivariate cdfs. These supplement existing functions for pdfs and random number generators for the same distributions.

- `mvncdf` — Cumulative distribution function for the multivariate normal distribution
- `mvtcdf` — Cumulative distribution function for the multivariate *t* distribution

Random Number Generation

Copulas

New functions have been added to the toolbox that allow you to use copulas to model correlated multivariate data and generate random numbers from multivariate distributions.

- `copulacdf` — Cumulative distribution function for a copula
- `copulaparam` — Copula parameters as a function of rank correlation
- `copulapdf` — Probability density function for a copula
- `copularnd` — Random numbers from a copula
- `copulastat` — Rank correlation for a copula

Markov Chain Monte Carlo Methods

The following functions generate random numbers from nonstandard distributions using Markov Chain Monte Carlo methods:

- `mhsample` — Generate random numbers using the Metropolis-Hasting algorithm

- `slicesample` — Generate random numbers using a slice sampling algorithm

Pearson and Johnson Systems of Distributions

Support has been added for random number generation from Pearson and Johnson systems of distributions.

- `pearsrnd` — Random numbers from a distribution in the Pearson system
- `johnsrnd` — Random numbers from a distribution in the Johnson system

Robust Regression

To supplement the `robustfit` function, the following functions now have options for robust fitting:

- `nlinfit` — Nonlinear least-squares regression
- `nlparci` — Confidence intervals for parameters in nonlinear regression
- `nlpredci` — Confidence intervals for predictions in nonlinear regression

Statistical Process Control

The following control chart functions now support time-series objects:

- `xbarplot` — Xbar plot
- `schart` — Standard deviation chart
- `ewmaplot` — Exponentially weighted moving average plot

R14SP3

Version: 5.1

New Features

Demos

The following demos have been added to the toolbox:

- Curve Fitting and Distribution Fitting
- Fitting a Univariate Distribution Using Cumulative Probabilities
- Fitting an Orthogonal Regression Using Principal Components Analysis
- Modelling Tail Data with the Generalized Pareto Distribution
- Pitfalls in Fitting Nonlinear Models by Transforming to Linearity
- Weighted Nonlinear Regression

The following demo has been updated:

- Modelling Data with the Generalized Extreme Value Distribution

Descriptive Statistics

The new `partialcorr` function computes the correlation of one set of variables while controlling for a second set of variables.

The `grpstats` function now computes a wider variety of descriptive statistics for grouped data. Choices include the mean, standard error of the mean, number of elements, group name, standard deviation, variance, confidence interval for the mean, and confidence interval for new observations. The function also supports the computation of user-defined statistics.

Hypothesis Tests

Chi-Square Goodness-of-Fit Test

The new `chi2gof` function tests if a sample comes from a specified distribution, against the alternative that it does not come from that distribution, using a chi-square test statistic.

Variance Tests

Three functions have been added to test sample variances:

- `vartest` — One-sample chi-square variance test. Tests if a sample comes from a normal distribution with specified variance, against the alternative that it comes from a normal distribution with a different variance.
- `vartest2` — Two-sample *F*-test for equal variances. Tests if two independent samples come from normal distributions with the same variance, against the alternative that they come from normal distributions with different variances.
- `vartestn` — Bartlett multiple-sample test for equal variances. Tests if multiple samples come from normal distributions with the same variance, against the alternative that they come from normal distributions with different variances.

Ansari-Bradley Test

The new `ansaribradley` function tests if two independent samples come from the same distribution, against the alternative that they come from distributions that have the same median and shape but different variances.

Tests of Randomness

The new `runstest` function tests if a sequence of values comes in random order, against the alternative that the ordering is not random.

Probability Distributions

Support has been added for two new distributions:

- “Generalized Extreme Value Distribution” on page 26-3
- “Generalized Pareto Distribution” on page 26-4

Generalized Extreme Value Distribution

The Generalized Extreme Value distribution combines the Gumbel, Frechet, and Weibull distributions into a single distribution. It is used to model extreme values in data.

The following distribution functions have been added:

- `gevcdf` — Cumulative distribution function
- `gevfit` — Parameter estimation function
- `gevinv` — Inverse cumulative distribution function
- `gevlike` — Negative log-likelihood function

- `gevpdf` — Probability density function
- `gevrnd` — Random number generator
- `gevstat` — Distribution statistics

Generalized Pareto Distribution

The Generalized Pareto distribution is used to model the tails of a data distribution.

The following distribution functions have been added:

- `gpcdf` — Cumulative distribution function
- `gpfit` — Parameter estimation function
- `gpinv` — Inverse cumulative distribution function
- `gplike` — Negative log-likelihood function
- `gppdf` — Probability density function
- `gprnd` — Random number generator
- `gpstat` — Distribution statistics

Regression Analysis

- The new `coxphfit` function fits Cox's proportional hazards regression model to data.
- The new `invpred` function estimates the inverse prediction intervals for simple linear regression.
- The `polyconf` function has new options to let you specify the confidence interval computed.

Statistical Visualization

Both the `ecdf` and `ksdensity` functions now produce plots when no output arguments are specified.

R14SP2

Version: 5.0.2

New Features

Bug Fixes

Multivariate Statistics

The `cophenet` function now returns cophenetic distances as well as the cophenetic correlation coefficient.