preliminaries

2017 年 6 月 28 日

0.0.1 Learn More and Get Help

Documentation: http://statsmodels.sf.net

Mailing List: http://groups.google.com/group/pystatsmodels

Use the source: https://github.com/statsmodels/statsmodels

0.0.2 Tutorial Import Assumptions

```
In [ ]: import numpy as np
        import statsmodels.api as sm
        import matplotlib.pyplot as plt
        import pandas
        from scipy import stats

        np.set_printoptions(precision=4, suppress=True)
        pandas.set_printoptions(notebook_repr_html=False,
                                precision=4,
                                max_columns=12)
```

0.0.3 Statsmodels Import Convention

```
In [1]: import statsmodels.api as sm
```

Import convention for models for which a formula is available.

```
In [1]: from statsmodels.formula.api import ols, rlm, glm, #etc.
```

0.0.4 Package Overview

Regression models in statsmodels.regression

Discrete choice models in statsmodels.discrete

Robust linear models in statsmodels.robust

Generalized linear models in statsmodels.genmod

Time Series Analysis in statsmodels.tsa

Nonparametric models in statsmodels.nonparametric

Plotting functions in statsmodels.graphics

Input/Output in statsmodels.iolib (Foreign data, ascii, HTML, $\LaTeX$ tables)

Statistical tests, ANOVA in statsmodels.stats

Datasets in statsmodels.datasets (See also the new GPL package Rdatasets: https://github.com/vincentarelbundock/Rdatasets)

0.0.5 Base Classes

```
In [5]: from statsmodels.base import model
```

```
In [6]: help(model.Model)
```

```
Help on class Model in module statsmodels.base.model:

class Model(__builtin__.object)
 |  A (predictive) statistical model. The class Model itself is not to be used.
 |
 |  Model lays out the methods expected of any subclass.
 |
 |  Parameters
 |  ----------
 |  endog : array-like
 |      Endogenous response variable.
 |  exog : array-like
 |      Exogenous design.
 |
 |  Notes
 |  -----
 |  `endog` and `exog` are references to any data provided.  So if the data is
 |  already stored in numpy arrays and it is changed then `endog` and `exog`
 |  will change as well.
 |
 |  Methods defined here:
 |
 |  __init__(self, endog, exog=None)
 |
 |  fit(self)
 |      Fit a model to data.
 |
 |  predict(self, params, exog=None, *args, **kwargs)
 |      After a model has been fit predict returns the fitted values.
 |
 |      This is a placeholder intended to be overwritten by individual models.
 |
 |  ----------------------------------------------------------------------
 |  Class methods defined here:
 |
 |  from_formula(cls, formula, df, subset=None, *args, **kwargs) from __builtin__.t
 |      Create a Model from a formula and dataframe.
 |
 |      Parameters
 |      ----------
 |      formula : str or generic Formula object
 |          The formula specifying the model
 |      df : array-like
 |          The data for the model. See Notes.
 |      subset : array-like
 |          An array-like object of booleans, integers, or index values that
 |          indicate the subset of df to use in the model. Assumes df is a
 |          `pandas.DataFrame`
 |      args : extra arguments
 |          These are passed to the model
 |      kwargs : extra keyword arguments
 |          These are passed to the model.
 |
 |      Returns
 |      -------
 |      model : Model instance
 |
 |      Notes
 |      ------
```

```
 |      df must define __getitem__ with the keys in the formula terms
 |      args and kwargs are passed on to the model instantiation. E.g.,
 |      a numpy structured or rec array, a dictionary, or a pandas DataFrame.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  endog_names
 |
 |  exog_names
```

```
In [7]: help(model.LikelihoodModel)
```

```
Help on class LikelihoodModel in module statsmodels.base.model:

class LikelihoodModel(Model)
 |  Likelihood model is a subclass of Model.
 |
 |  Method resolution order:
 |      LikelihoodModel
 |      Model
 |      __builtin__.object
 |
 |  Methods defined here:
 |
 |  __init__(self, endog, exog=None)
 |
 |  fit(self, start_params=None, method='newton', maxiter=100, full_output=True, di
 |      Fit method for likelihood based models
 |
 |       Parameters
 |       ----------
 |       start_params : array-like, optional
 |           Initial guess of the solution for the loglikelihood maximization.
 |           The default is an array of zeros.
 |       method : str {'newton','nm','bfgs','powell','cg', or 'ncg'}
 |           Method can be 'newton' for Newton-Raphson, 'nm' for Nelder-Mead,
 |           'bfgs' for Broyden-Fletcher-Goldfarb-Shanno, 'powell' for modified
 |           Powell's method, 'cg' for conjugate gradient, or 'ncg' for Newton-
 |           conjugate gradient. `method` determines which solver from
 |           scipy.optimize is used.  The explicit arguments in `fit` are passed
 |           to the solver.  Each solver has several optional arguments that are
 |           not the same across solvers.  See the notes section below (or
 |           scipy.optimize) for the available arguments.
 |       maxiter : int
 |           The maximum number of iterations to perform.
 |       full_output : bool
 |           Set to True to have all available output in the Results object's
 |           mle_retvals attribute. The output is dependent on the solver.
 |           See LikelihoodModelResults notes section for more information.
 |       disp : bool
 |           Set to True to print convergence messages.
 |       fargs : tuple
 |           Extra arguments passed to the likelihood function, i.e.,
 |           loglike(x,*args)
 |       callback : callable callback(xk)
 |           Called after each iteration, as callback(xk), where xk is the
 |           current parameter vector.
 |       retall : bool
 |           Set to True to return list of solutions at each iteration.
 |           Available in Results object's mle_retvals attribute.
 |
 |       Notes
 |       -----
 |       Optional arguments for the solvers (available in Results.mle_settings):
 |
 |           'newton'
 |               tol : float
 |                   Relative error in params acceptable for convergence.
 |           'nm' -- Nelder Mead
 |               xtol : float
 |                   Relative error in params acceptable for convergence
 |               ftol : float
 |                   Relative error in loglike(params) acceptable for
 |                   convergence
 |               maxfun : int
 |                   Maximum number of function evaluations to make.
 |           'bfgs'
 |               gtol : float
 |                   Stop when norm of gradient is less than gtol.
 |               norm : float
 |                   Order of norm (np.Inf is max, -np.Inf is min)
 |               epsilon
 |                   If fprime is approximated, use this value for the step
 |                   size. Only relevant if LikelihoodModel.score is None.
 |           'cg'
 |               gtol : float
 |                   Stop when norm of gradient is less than gtol.
 |               norm : float
 |                   Order of norm (np.Inf is max, -np.Inf is min)
 |               epsilon : float
 |                   If fprime is approximated, use this value for the step
 |                   size. Can be scalar or vector.  Only relevant if
 |                   Likelihoodmodel.score is None.
 |           'ncg'
 |               fhess_p : callable f'(x,*args)
 |                   Function which computes the Hessian of f times an arbitrary
 |                   vector, p.  Should only be supplied if
 |                   LikelihoodModel.hessian is None.
 |               avextol : float
 |                   Stop when the average relative error in the minimizer
 |                   falls below this amount.
 |               epsilon : float or ndarray
 |                   If fhess is approximated, use this value for the step size.
 |                   Only relevant if Likelihoodmodel.hessian is None.
 |           'powell'
 |               xtol : float
 |                   Line-search error tolerance
 |               ftol : float
 |                   Relative error in loglike(params) for acceptable for
 |                   convergence.
 |               maxfun : int
 |                   Maximum number of function evaluations to make.
 |               start_direc : ndarray
 |                   Initial direction set.
 |
 |   hessian(self, params)
 |       The Hessian matrix of the model
 |
 |   information(self, params)
 |       Fisher information matrix of model
 |
 |       Returns -Hessian of loglike evaluated at params.
 |
 |   initialize(self)
 |       Initialize (possibly re-initialize) a Model instance. For
 |       instance, the design matrix of a linear model may change
 |       and some things must be recomputed.
 |
 |   loglike(self, params)
 |       Log-likelihood of model.
 |
 |   score(self, params)
 |       Score vector of model.
 |
 |       The gradient of logL with respect to each parameter.
 |
 |   ----------------------------------------------------------------------
 |   Methods inherited from Model:
 |
 |   predict(self, params, exog=None, *args, **kwargs)
 |      After a model has been fit predict returns the fitted values.
 |
 |      This is a placeholder intended to be overwritten by individual models.
 |
 |  ----------------------------------------------------------------------
 |  Class methods inherited from Model:
 |
 |  from_formula(cls, formula, df, subset=None, *args, **kwargs) from __builtin__.t
 |      Create a Model from a formula and dataframe.
 |
 |      Parameters
 |      ----------
 |      formula : str or generic Formula object
 |          The formula specifying the model
 |      df : array-like
 |          The data for the model. See Notes.
 |      subset : array-like
 |          An array-like object of booleans, integers, or index values that
 |          indicate the subset of df to use in the model. Assumes df is a
 |          `pandas.DataFrame`
 |      args : extra arguments
 |          These are passed to the model
 |      kwargs : extra keyword arguments
 |          These are passed to the model.
 |
 |      Returns
 |      -------
 |      model : Model instance
 |
 |      Notes
 |      ------
 |      df must define __getitem__ with the keys in the formula terms
 |      args and kwargs are passed on to the model instantiation. E.g.,
 |      a numpy structured or rec array, a dictionary, or a pandas DataFrame.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors inherited from Model:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  endog_names
 |
 |  exog_names
```

```
In [8]: help(model.LikelihoodModelResults)
```

```
Help on class LikelihoodModelResults in module statsmodels.base.model:

class LikelihoodModelResults(Results)
 |  Class to contain results from likelihood models
 |
 |  Parameters
 |  -----------
 |  model : LikelihoodModel instance or subclass instance
 |      LikelihoodModelResults holds a reference to the model that is fit.
 |  params : 1d array_like
 |      parameter estimates from estimated model
 |  normalized_cov_params : 2d array
 |     Normalized (before scaling) covariance of params. (dot(X.T,X))**-1
 |  scale : float
 |      For (some subset of models) scale will typically be the
 |      mean square error from the estimated model (sigma^2)
 |
 |  Returns
 |  -------
 |  **Attributes**
 |  mle_retvals : dict
 |      Contains the values returned from the chosen optimization method if
 |      full_output is True during the fit.  Available only if the model
 |       is fit by maximum likelihood.  See notes below for the output from
 |       the different methods.
 |   mle_settings : dict
 |       Contains the arguments passed to the chosen optimization method.
 |       Available if the model is fit by maximum likelihood.  See
 |       LikelihoodModel.fit for more information.
 |   model : model instance
 |       LikelihoodResults contains a reference to the model that is fit.
 |   params : ndarray
 |       The parameters estimated for the model.
 |   scale : float
 |       The scaling factor of the model given during instantiation.
 |   tvalues : array
 |       The t-values of the standard errors.
 |
 |
 |   Notes
 |   --------
 |   The covariance of params is given by scale times normalized_cov_params.
 |
 |   Return values by solver if full_ouput is True during fit:
 |
 |       'newton'
 |           fopt : float
 |               The value of the (negative) loglikelihood at its
 |               minimum.
 |           iterations : int
 |               Number of iterations performed.
 |           score : ndarray
 |               The score vector at the optimum.
 |           Hessian : ndarray
 |               The Hessian at the optimum.
 |           warnflag : int
 |               1 if maxiter is exceeded. 0 if successful convergence.
 |           converged : bool
 |               True: converged. False: did not converge.
 |           allvecs : list
```

```
 |              List of solutions at each iteration.
 |      'nm'
 |          fopt : float
 |              The value of the (negative) loglikelihood at its
 |              minimum.
 |          iterations : int
 |              Number of iterations performed.
 |          warnflag : int
 |              1: Maximum number of function evaluations made.
 |              2: Maximum number of iterations reached.
 |          converged : bool
 |              True: converged. False: did not converge.
 |          allvecs : list
 |              List of solutions at each iteration.
 |      'bfgs'
 |          fopt : float
 |              Value of the (negative) loglikelihood at its minimum.
 |          gopt : float
 |              Value of gradient at minimum, which should be near 0.
 |          Hinv : ndarray
 |              value of the inverse Hessian matrix at minimum.  Note
 |              that this is just an approximation and will often be
 |              different from the value of the analytic Hessian.
 |          fcalls : int
 |              Number of calls to loglike.
 |          gcalls : int
 |              Number of calls to gradient/score.
 |          warnflag : int
 |              1: Maximum number of iterations exceeded. 2: Gradient
 |              and/or function calls are not changing.
 |          converged : bool
 |              True: converged.  False: did not converge.
 |          allvecs : list
 |              Results at each iteration.
 |      'powell'
 |          fopt : float
 |              Value of the (negative) loglikelihood at its minimum.
```

```
|           direc : ndarray
|               Current direction set.
|           iterations : int
|               Number of iterations performed.
|           fcalls : int
|               Number of calls to loglike.
|           warnflag : int
|               1: Maximum number of function evaluations. 2: Maximum number
|               of iterations.
|           converged : bool
|               True : converged. False: did not converge.
|           allvecs : list
|               Results at each iteration.
|       'cg'
|           fopt : float
|               Value of the (negative) loglikelihood at its minimum.
|           fcalls : int
|               Number of calls to loglike.
|           gcalls : int
|               Number of calls to gradient/score.
|           warnflag : int
|               1: Maximum number of iterations exceeded. 2: Gradient and/
|               or function calls not changing.
|           converged : bool
|               True: converged. False: did not converge.
|           allvecs : list
|               Results at each iteration.
|       'ncg'
|           fopt : float
|               Value of the (negative) loglikelihood at its minimum.
|           fcalls : int
|               Number of calls to loglike.
|           gcalls : int
|               Number of calls to gradient/score.
|           hcalls : int
|               Number of calls to hessian.
|           warnflag : int
|               1: Maximum number of iterations exceeded.
|           converged : bool
|               True: converged. False: did not converge.
|           allvecs : list
|               Results at each iteration.
|
|   Method resolution order:
|       LikelihoodModelResults
|       Results
|       __builtin__.object
|
|   Methods defined here:
|
|   __init__(self, model, params, normalized_cov_params=None, scale=1.0)
|
|   conf_int(self, alpha=0.05, cols=None, method='default')
|       Returns the confidence interval of the fitted parameters.
|
|       Parameters
|       ----------
|       alpha : float, optional
|           The `alpha` level for the confidence interval.
|           ie., The default `alpha` = .05 returns a 95% confidence interval.
|       cols : array-like, optional
|           `cols` specifies which confidence intervals to return
|       method : string
|           Not Implemented Yet
|           Method to estimate the confidence_interval.
|           "Default" : uses self.bse which is based on inverse Hessian for MLE
|           "jhj" :
|           "jac" :
|           "boot-bse"
|           "boot_quant"
|           "profile"
|
|
|       Returns
|      --------
|      conf_int : array
|          Each row contains [lower, upper] confidence interval
|
|      Examples
|      --------
|      >>> import statsmodels.api as sm
|      >>> data = sm.datasets.longley.load()
|      >>> data.exog = sm.add_constant(data.exog)
|      >>> results = sm.OLS(data.endog, data.exog).fit()
|      >>> results.conf_int()
|      array([[ -1.77029035e+02,   2.07152780e+02],
|      [ -1.11581102e-01,   3.99427438e-02],
|      [ -3.12506664e+00,  -9.15392966e-01],
|      [ -1.51794870e+00,  -5.48505034e-01],
|      [ -5.62517214e-01,   4.60309003e-01],
|      [  7.98787515e+02,   2.85951541e+03],
|      [ -5.49652948e+06,  -1.46798779e+06]])
|
|      >>> results.conf_int(cols=(1,2))
|      array([[-0.1115811 ,  0.03994274],
|      [-3.12506664, -0.91539297]])
|
|      Notes
|      -----
|      The confidence interval is based on the standard normal distribution.
|      Models wish to use a different distribution should overwrite this
|      method.
|
|  cov_params(self, r_matrix=None, column=None, scale=None, cov_p=None, other=None
|      Returns the variance/covariance matrix.
|
|      The variance/covariance matrix can be of a linear contrast
|      of the estimates of params or all params multiplied by scale which
|      will usually be an estimate of sigma^2.  Scale is assumed to be
|      a scalar.
|
|      Parameters
|      ----------
|      r_matrix : array-like
|          Can be 1d, or 2d.  Can be used alone or with other.
|      column :  array-like, optional
|          Must be used on its own.  Can be 0d or 1d see below.
|      scale : float, optional
|          Can be specified or not.  Default is None, which means that
|          the scale argument is taken from the model.
|      other : array-like, optional
|          Can be used when r_matrix is specified.
|
|      Returns
|      -------
|      (The below are assumed to be in matrix notation.)
|
|      cov : ndarray
|
|      If no argument is specified returns the covariance matrix of a model
|      (scale)*(X.T X)^(-1)
|
|      If contrast is specified it pre and post-multiplies as follows
|      (scale) * r_matrix (X.T X)^(-1) r_matrix.T
|
|      If contrast and other are specified returns
|      (scale) * r_matrix (X.T X)^(-1) other.T
|
|      If column is specified returns
|      (scale) * (X.T X)^(-1)[column,column] if column is 0d
|
|      OR
|
|      (scale) * (X.T X)^(-1)[column][:,column] if column is 1d
|
|  f_test(self, r_matrix, q_matrix=None, cov_p=None, scale=1.0, invcov=None)
|      Compute an F-test for a joint linear hypothesis.
|
|       Parameters
|       ----------
|       r_matrix : array-like, str, or tuple
|           - array : An r x k array where r is the number of restrictions to
|             test and k is the number of regressors.
|           - str : The full hypotheses to test can be given as a string.
|             See the examples.
|           - tuple : A tuple of arrays in the form (R, q), since q_matrix is
|             deprecated.
|       q_matrix : array-like
|           This is deprecated. See `r_matrix` and the examples for more
|           information on new usage. Can be either a scalar or a length p
|           row vector. If omitted and r_matrix is an array, `q_matrix` is
|           assumed to be a conformable array of zeros.
|       cov_p : array-like, optional
|           An alternative estimate for the parameter covariance matrix.
|           If None is given, self.normalized_cov_params is used.
|       scale : float, optional
|           Default is 1.0 for no scaling.
|       invcov : array-like, optional
|           A q x q array to specify an inverse covariance matrix based on a
|           restrictions matrix.
|
|       Examples
|       --------
|       >>> import numpy as np
|       >>> import statsmodels.api as sm
|       >>> data = sm.datasets.longley.load()
|       >>> data.exog = sm.add_constant(data.exog)
|       >>> results = sm.OLS(data.endog, data.exog).fit()
|       >>> A = np.identity(len(results.params))
|       >>> A = A[:-1,:]
|
|       This tests that each coefficient is jointly statistically
|       significantly different from zero.
|
|       >>> print results.f_test(A)
|      <F contrast: F=330.28533923463488, p=4.98403052872e-10,
|       df_denom=9, df_num=6>
|
|      Compare this to
|
|      >>> results.F
|      330.2853392346658
|      >>> results.F_p
|      4.98403096572e-10
|
|      >>> B = np.array(([0,1,-1,0,0,0,0],[0,0,0,0,1,-1,0]))
|
|      This tests that the coefficient on the 2nd and 3rd regressors are
|      equal and jointly that the coefficient on the 5th and 6th regressors
|      are equal.
|
|      >>> print results.f_test(B)
|      <F contrast: F=9.740461873303655, p=0.00560528853174, df_denom=9,
|       df_num=2>
|
|      Alternatively, you can specify the hypothesis tests using a string
|
|      >>> from statsmodels.datasets import longley
|      >>> from statsmodels.formula.api import ols
|      >>> dta = longley.load_pandas().data
|      >>> formula = 'TOTEMP ~ GNPDEFL + GNP + UNEMP + ARMED + POP + YEAR'
|      >>> results = ols(formula, dta).fit()
|      >>> hypotheses = '(GNPDEFL = GNP), (UNEMP = 2), (YEAR/1829 = 1)'
|      >>> f_test = results.new_f_test(hypotheses)
|      >>> print f_test
|
|      See also
|      --------
|      statsmodels.contrasts
|      statsmodels.model.t_test
|      patsy.DesignInfo.linear_constraint
|
|      Notes
|      -----
|      The matrix `r_matrix` is assumed to be non-singular. More precisely,
|
|      r_matrix (pX pX.T) r_matrix.T
|
|      is assumed invertible. Here, pX is the generalized inverse of the
|      design matrix of the model. There can be problems in non-OLS models
|      where the rank of the covariance of the noise is not full.
|
|  normalized_cov_params(self)
|
|  remove_data(self)
|      remove data arrays, all nobs arrays from result and model
|
|      This reduces the size of the instance, so it can be pickled with less
|      memory. Currently tested for use with predict from an unpickled
|      results and model instance.
|
|      .. warning:: Since data and some intermediate results have been removed
|         calculating new statistics that require them will raise exceptions.
|         The exception will occur the first time an attribute is accessed that
|         has been set to None.
|
|      Not fully tested for time series models, tsa, and might delete too much
|      for prediction or not all that would be possible.
|
|      The list of arrays to delete is maintained as an attribute of the
|      result and model instance, except for cached values. These lists could
|      be changed before calling remove_data.
|
|  save(self, fname, remove_data=False)
|      save a pickle of this instance
|
|      Parameters
|      ----------
|      fname : string or filehandle
```

```
|           fname can be a string to a file path or filename, or a filehandle.
|       remove_data : bool
|           If False (default), then the instance is pickled without changes.
|           If True, then all arrays with length nobs are set to None before
|           pickling. See the remove_data method.
|           In some cases not all arrays will be set to None.
|
|       Notes
|       -----
|       If remove_data is true and the model result does not implement a
|       remove_data method then this will raise an exception.
|
|   t(self, column=None)
|       deprecated: Return the t-statistic for a given parameter estimate.
|
|       FutureWarning: use attribute tvalues instead, t will be removed
|       in the next release
|
|       Parameters
|       ----------
|       column : array-like
|           The columns for which you would like the t-value.
|           Note that this uses Python's indexing conventions.
|
|       See also
|       ---------
|       Use t_test for more complicated t-statistics.
|
|       Examples
|       --------
|       >>> import statsmodels.api as sm
|       >>> data = sm.datasets.longley.load()
|       >>> data.exog = sm.add_constant(data.exog)
|       >>> results = sm.OLS(data.endog, data.exog).fit()
|       >>> results.tvalues
|       array([ 0.17737603, -1.06951632, -4.13642736, -4.82198531, -0.22605114,
|       4.01588981, -3.91080292])
|      >>> results.tvalues[[1,2,4]]
|      array([-1.06951632, -4.13642736, -0.22605114])
|      >>> import numpy as np
|      >>> results.tvalues[np.array([1,2,4]]
|      array([-1.06951632, -4.13642736, -0.22605114])
|
|  t_test(self, r_matrix, q_matrix=None, cov_p=None, scale=None)
|      Compute a t-test for a joint linear hypothesis of the form Rb = q
|
|      Parameters
|      ----------
|      r_matrix : array-like, str, tuple
|          - array : If an array is given, a p x k 2d array or length k 1d
|            array specifying the linear restrictions.
|          - str : The full hypotheses to test can be given as a string.
|            See the examples.
|          - tuple : A tuple of arrays in the form (R, q), since q_matrix is
|            deprecated.
|      q_matrix : array-like or scalar, optional
|          This is deprecated. See `r_matrix` and the examples for more
|          information on new usage. Can be either a scalar or a length p
|          row vector. If omitted and r_matrix is an array, `q_matrix` is
|          assumed to be a conformable array of zeros.
|      cov_p : array-like, optional
|          An alternative estimate for the parameter covariance matrix.
|          If None is given, self.normalized_cov_params is used.
|      scale : float, optional
|          An optional `scale` to use.  Default is the scale specified
|          by the model fit.
|
|      Examples
|      --------
|      >>> import numpy as np
|      >>> import statsmodels.api as sm
|      >>> data = sm.datasets.longley.load()
|      >>> data.exog = sm.add_constant(data.exog)
|      >>> results = sm.OLS(data.endog, data.exog).fit()
|       >>> r = np.zeros_like(results.params)
|       >>> r[4:6] = [1,-1]
|       >>> print r
|       [ 0.  0.  0.  0.  1. -1.  0.]
|
|       r tests that the coefficients on the 5th and 6th independent
|       variable are the same.
|
|       >>>T_Test = results.t_test(r)
|       >>>print T_test
|       <T contrast: effect=-1829.2025687192481, sd=455.39079425193762,
|       t=-4.0167754636411717, p=0.0015163772380899498, df_denom=9>
|       >>> T_test.effect
|       -1829.2025687192481
|       >>> T_test.sd
|       455.39079425193762
|       >>> T_test.t
|       -4.0167754636411717
|       >>> T_test.p
|       0.0015163772380899498
|
|       Alternatively, you can specify the hypothesis tests using a string
|
|       >>> dta = sm.datasets.longley.load_pandas().data
|       >>> formula = 'TOTEMP ~ GNPDEFL + GNP + UNEMP + ARMED + POP + YEAR'
|       >>> results = ols(formula, dta).fit()
|       >>> hypotheses = 'GNPDEFL = GNP, UNEMP = 2, YEAR/1829 = 1'
|       >>> t_test = results.new_t_test(hypotheses)
|       >>> print t_test
|
|       See also
|       ---------
|       tvalues : individual t statistics
|       f_test : for F tests
|       patsy.DesignInfo.linear_constraint
|
|   ----------------------------------------------------------------------
|  Class methods defined here:
|
|  load(cls, fname) from __builtin__.type
|      load a pickle, (class method)
|
|      Parameters
|      ----------
|      fname : string or filehandle
|          fname can be a string to a file path or filename, or a filehandle.
|
|      Returns
|      -------
|      unpickled instance
|
|  ----------------------------------------------------------------------
|  Data descriptors defined here:
|
|  bse
|
|  llf
|
|  pvalues
|
|  tvalues
|
|  ----------------------------------------------------------------------
|  Methods inherited from Results:
|
|  initialize(self, model, params, **kwd)
|
|  predict(self, exog=None, transform=True, *args, **kwargs)
|      Call self.model.predict with self.params as the first argument.
|
|      Parameters
|      ----------
|      exog : array-like, optional
|          The values for which you want to predict.
```

```
 |      transform : bool, optional
 |          If the model was fit via a formula, do you want to pass
 |          exog through the formula. Default is True. E.g., if you fit
 |          a model y ~ log(x1) + log(x2), and transform is True, then
 |          you can pass a data structure that contains x1 and x2 in
 |          their original form. Otherwise, you'd need to log the data
 |          first.
 |
 |      Returns
 |      -------
 |      See self.model.predict
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors inherited from Results:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
```

```
In [9]: from statsmodels.regression.linear_model import RegressionResults

In [10]: help(RegressionResults)
```

```
Help on class RegressionResults in module statsmodels.regression.linear_model:

class RegressionResults(statsmodels.base.model.LikelihoodModelResults)
 |  This class summarizes the fit of a linear regression model.
 |
 |  It handles the output of contrasts, estimates of covariance, etc.
 |
 |  Returns
 |  -------
 |  **Attributes**
 |
 |  aic
 |      Aikake's information criteria :math:`-2llf + 2(df_model+1)`
 |  bic
 |      Bayes' information criteria :math:`-2llf + \log(n)(df_model+1)`
 |  bse
 |      The standard errors of the parameter estimates.
 |  pinv_wexog
 |      See specific model class docstring
 |  centered_tss    sum(y-y_mean)^2
 |      The total sum of squares centered about the mean
 |  cov_HC0
 |      See HC0_se below.  Only available after calling HC0_se.
 |  cov_HC1
 |      See HC1_se below.  Only available after calling HC1_se.
 |  cov_HC2
 |      See HC2_se below.  Only available after calling HC2_se.
 |  cov_HC3
 |      See HC3_se below.  Only available after calling HC3_se.
 |  df_model :
 |      Model degress of freedom. The number of regressors p - 1 for the
 |      constant  Note that df_model does not include the constant even though
 |      the design does.  The design is always assumed to have a constant
 |      in calculating results for now.
 |  df_resid
 |      Residual degrees of freedom. n - p.  Note that the constant *is*
 |      included in calculating the residual degrees of freedom.
 |  ess
 |      Explained sum of squares.  The centered total sum of squares minus
 |      the sum of squared residuals.
 |  fvalue
 |      F-statistic of the fully specified model.  Calculated as the mean
 |      squared error of the model divided by the mean squared error of the
 |      residuals.
 |  f_pvalue
 |      p-value of the F-statistic
 |  fittedvalues
 |      The predicted the values for the original (unwhitened) design.
 |  het_scale
 |      Only available if HC#_se is called.  See HC#_se for more information.
 |  HC0_se
 |      White's (1980) heteroskedasticity robust standard errors.
 |      Defined as sqrt(diag(X.T X)^(-1)X.T diag(e_i^(2)) X(X.T X)^(-1)
 |      where e_i = resid[i]
 |      HC0_se is a property.  It is not evaluated until it is called.
 |      When it is called the RegressionResults instance will then have
 |      another attribute cov_HC0, which is the full heteroskedasticity
 |      consistent covariance matrix and also `het_scale`, which is in
 |      this case just resid**2.  HCCM matrices are only appropriate for OLS.
 |  HC1_se
 |      MacKinnon and White's (1985) alternative heteroskedasticity robust
 |      standard errors.
 |      Defined as sqrt(diag(n/(n-p)*HC_0)
 |      HC1_se is a property.  It is not evaluated until it is called.
 |      When it is called the RegressionResults instance will then have
 |      another attribute cov_HC1, which is the full HCCM and also `het_scale`,
 |      which is in this case n/(n-p)*resid**2.  HCCM matrices are only
 |      appropriate for OLS.
 |  HC2_se
 |      MacKinnon and White's (1985) alternative heteroskedasticity robust
 |      standard errors.
 |      Defined as (X.T X)^(-1)X.T diag(e_i^(2)/(1-h_ii)) X(X.T X)^(-1)
 |      where h_ii = x_i(X.T X)^(-1)x_i.T
 |      HC2_se is a property.  It is not evaluated until it is called.
 |      When it is called the RegressionResults instance will then have
 |      another attribute cov_HC2, which is the full HCCM and also `het_scale`,
 |      which is in this case is resid^(2)/(1-h_ii).  HCCM matrices are only
 |      appropriate for OLS.
 |  HC3_se
 |      MacKinnon and White's (1985) alternative heteroskedasticity robust
 |      standard errors.
 |      Defined as (X.T X)^(-1)X.T diag(e_i^(2)/(1-h_ii)^(2)) X(X.T X)^(-1)
 |      where h_ii = x_i(X.T X)^(-1)x_i.T
 |      HC3_se is a property.  It is not evaluated until it is called.
 |      When it is called the RegressionResults instance will then have
 |      another attribute cov_HC3, which is the full HCCM and also `het_scale`,
 |      which is in this case is resid^(2)/(1-h_ii)^(2).  HCCM matrices are
 |      only appropriate for OLS.
 |  model
 |      A pointer to the model instance that called fit() or results.
 |  mse_model
 |      Mean squared error the model. This is the explained sum of squares
 |      divided by the model degrees of freedom.
 |  mse_resid
 |      Mean squared error of the residuals.  The sum of squared residuals
 |      divided by the residual degrees of freedom.
 |  mse_total
 |      Total mean squared error.  Defined as the uncentered total sum of
 |      squares divided by n the number of observations.
 |  nobs
 |      Number of observations n.
 |  normalized_cov_params
 |      See specific model class docstring
 |  params
 |      The linear coefficients that minimize the least squares criterion.  This
 |      is usually called Beta for the classical linear model.
 |  pvalues
 |      The two-tailed p values for the t-stats of the params.
 |  resid
 |      The residuals of the model.
 |  rsquared
 |      R-squared of a model with an intercept.  This is defined here as
 |      1 - `ssr`/`centered_tss`
 |  rsquared_adj
 |      Adjusted R-squared.  This is defined here as
 |      1 - (n-1)/(n-p)*(1-`rsquared`)
 |  scale
 |      A scale factor for the covariance matrix.
 |      Default value is ssr/(n-p).  Note that the square root of `scale` is
 |      often called the standard error of the regression.
 |  ssr
 |      Sum of squared (whitened) residuals.
 |  uncentered_tss
 |      Uncentered sum of squares.  Sum of the squared values of the
 |      (whitened) endogenous response variable.
 |  wresid
 |      The residuals of the transformed/whitened regressand and regressor(s)
 |
 |  Method resolution order:
 |      RegressionResults
 |      statsmodels.base.model.LikelihoodModelResults
 |      statsmodels.base.model.Results
 |      __builtin__.object
 |
 |  Methods defined here:
 |
 |  __init__(self, model, params, normalized_cov_params=None, scale=1.0)
 |
 |  __str__(self)
 |
 |  compare_f_test(self, restricted)
 |      use F test to test whether restricted model is correct
 |
 |      Parameters
 |      ----------
 |      restricted : Result instance
 |          The restricted model is assumed to be nested in the current
 |          model. The result instance of the restricted model is required to
 |          have two attributes, residual sum of squares, `ssr`, residual
 |          degrees of freedom, `df_resid`.
 |
 |      Returns
 |      -------
 |      f_value : float
 |          test statistic, F distributed
 |      p_value : float
 |          p-value of the test statistic
 |      df_diff : int
 |          degrees of freedom of the restriction, i.e. difference in df between
 |           models
 |
 |       Notes
 |       -----
 |       See mailing list discussion October 17,
 |
 |   compare_lr_test(self, restricted)
 |       Likelihood ratio test to test whether restricted model is correct
 |
 |       Parameters
 |       ----------
 |       restricted : Result instance
 |           The restricted model is assumed to be nested in the current model.
 |           The result instance of the restricted model is required to have two
 |           attributes, residual sum of squares, `ssr`, residual degrees of
 |           freedom, `df_resid`.
 |
 |       Returns
 |       -------
 |       lr_stat : float
 |           likelihood ratio, chisquare distributed with df_diff degrees of
 |           freedom
 |       p_value : float
 |           p-value of the test statistic
 |       df_diff : int
 |           degrees of freedom of the restriction, i.e. difference in df between
 |           models
 |
 |       Notes
 |       -----
 |
 |       .. math:: D=-2\log\left(\frac{\mathcal{L}_{null}}
 |          {\mathcal{L}_{alternative}}\right)
 |
 |       where :math:`\mathcal{L}` is the likelihood of the model. With :math:`D`
 |       distributed as chisquare with df equal to difference in number of
 |       parameters or equivalently difference in residual degrees of freedom
```

```
|
|      TODO: put into separate function, needs tests
|
|  conf_int(self, alpha=0.05, cols=None)
|      Returns the confidence interval of the fitted parameters.
|
|      Parameters
|      ----------
|      alpha : float, optional
|          The `alpha` level for the confidence interval.
|          ie., The default `alpha` = .05 returns a 95% confidence interval.
|      cols : array-like, optional
|          `cols` specifies which confidence intervals to return
|
|      Notes
|      -----
|      The confidence interval is based on Student's t-distribution.
|
|  norm_resid(self)
|      Residuals, normalized to have unit length and unit variance.
|
|      Returns
|      -------
|      An array wresid/sqrt(scale)
|
|      Notes
|      -----
|      This method is untested
|
|  summary(self, yname=None, xname=None, title=None, alpha=0.05)
|      Summarize the Regression Results
|
|      Parameters
|      -----------
|      yname : string, optional
|          Default is `y`
|      xname : list of strings, optional
|           Default is `var_##` for ## in p the number of regressors
|       title : string, optional
|           Title for the top table. If not None, then this replaces the
|           default title
|       alpha : float
|           significance level for the confidence intervals
|
|       Returns
|       -------
|       smry : Summary instance
|           this holds the summary tables and text, which can be printed or
|           converted to various output formats.
|
|       See Also
|       --------
|       statsmodels.iolib.summary.Summary : class to hold summary
|           results
|
|   summary_old(self, yname=None, xname=None, returns='text')
|       returns a string that summarizes the regression results
|
|       Parameters
|       -----------
|       yname : string, optional
|           Default is `Y`
|       xname : list of strings, optional
|           Default is `X.#` for # in p the number of regressors
|
|       Returns
|       -------
|       String summarizing the fit of a linear model.
|
|       Examples
|       --------
|       >>> import statsmodels.api as sm
|       >>> data = sm.datasets.longley.load()
|       >>> data.exog = sm.add_constant(data.exog)
|      >>> ols_results = sm.OLS(data.endog, data.exog).fit()
|      >>> print ols_results.summary()
|      ...
|
|      Notes
|      -----
|      All residual statistics are calculated on whitened residuals.
|
|  ----------------------------------------------------------------------
|  Data descriptors defined here:
|
|  HC0_se
|      See statsmodels.RegressionResults
|
|  HC1_se
|      See statsmodels.RegressionResults
|
|  HC2_se
|      See statsmodels.RegressionResults
|
|  HC3_se
|      See statsmodels.RegressionResults
|
|  aic
|
|  bic
|
|  bse
|
|  centered_tss
|
|  df_model
|
|  df_resid
|
|  ess
|
|  f_pvalue
|
|  fittedvalues
|
|  fvalue
|
|  mse_model
|
|  mse_resid
|
|  mse_total
|
|  nobs
|
|  pvalues
|
|  resid
|
|  rsquared
|
|  rsquared_adj
|
|  scale
|
|  ssr
|
|  uncentered_tss
|
|  wresid
|
|  ----------------------------------------------------------------------
|  Methods inherited from statsmodels.base.model.LikelihoodModelResults:
|
|  cov_params(self, r_matrix=None, column=None, scale=None, cov_p=None, other=None
|      Returns the variance/covariance matrix.
|
|      The variance/covariance matrix can be of a linear contrast
|      of the estimates of params or all params multiplied by scale which
|      will usually be an estimate of sigma^2.  Scale is assumed to be
|      a scalar.
|
|      Parameters
|      ----------
|      r_matrix : array-like
|          Can be 1d, or 2d.  Can be used alone or with other.
|      column :  array-like, optional
|          Must be used on its own.  Can be 0d or 1d see below.
|      scale : float, optional
|          Can be specified or not.  Default is None, which means that
|          the scale argument is taken from the model.
|      other : array-like, optional
|          Can be used when r_matrix is specified.
|
|      Returns
|      -------
|      (The below are assumed to be in matrix notation.)
|
|      cov : ndarray
|
|      If no argument is specified returns the covariance matrix of a model
|      (scale)*(X.T X)^(-1)
|
|      If contrast is specified it pre and post-multiplies as follows
|      (scale) * r_matrix (X.T X)^(-1) r_matrix.T
|
|      If contrast and other are specified returns
|      (scale) * r_matrix (X.T X)^(-1) other.T
|
|      If column is specified returns
|      (scale) * (X.T X)^(-1)[column,column] if column is 0d
|
|      OR
|
|      (scale) * (X.T X)^(-1)[column][:,column] if column is 1d
```

```
 |
 |  f_test(self, r_matrix, q_matrix=None, cov_p=None, scale=1.0, invcov=None)
 |      Compute an F-test for a joint linear hypothesis.
 |
 |      Parameters
 |      ----------
 |      r_matrix : array-like, str, or tuple
 |          - array : An r x k array where r is the number of restrictions to
 |            test and k is the number of regressors.
 |          - str : The full hypotheses to test can be given as a string.
 |            See the examples.
 |          - tuple : A tuple of arrays in the form (R, q), since q_matrix is
 |            deprecated.
 |      q_matrix : array-like
 |          This is deprecated. See `r_matrix` and the examples for more
 |          information on new usage. Can be either a scalar or a length p
 |          row vector. If omitted and r_matrix is an array, `q_matrix` is
 |          assumed to be a conformable array of zeros.
 |      cov_p : array-like, optional
 |          An alternative estimate for the parameter covariance matrix.
 |          If None is given, self.normalized_cov_params is used.
 |      scale : float, optional
 |          Default is 1.0 for no scaling.
 |      invcov : array-like, optional
 |          A q x q array to specify an inverse covariance matrix based on a
 |          restrictions matrix.
 |
 |      Examples
 |      --------
 |      >>> import numpy as np
 |      >>> import statsmodels.api as sm
 |      >>> data = sm.datasets.longley.load()
 |      >>> data.exog = sm.add_constant(data.exog)
 |      >>> results = sm.OLS(data.endog, data.exog).fit()
 |      >>> A = np.identity(len(results.params))
 |      >>> A = A[:-1,:]
 |
 |      This tests that each coefficient is jointly statistically
 |      significantly different from zero.
 |
 |      >>> print results.f_test(A)
 |      <F contrast: F=330.28533923463488, p=4.98403052872e-10,
 |       df_denom=9, df_num=6>
 |
 |      Compare this to
 |
 |      >>> results.F
 |      330.2853392346658
 |      >>> results.F_p
 |      4.98403096572e-10
 |
 |      >>> B = np.array(([0,1,-1,0,0,0,0],[0,0,0,0,1,-1,0]))
 |
 |      This tests that the coefficient on the 2nd and 3rd regressors are
 |      equal and jointly that the coefficient on the 5th and 6th regressors
 |      are equal.
 |
 |      >>> print results.f_test(B)
 |      <F contrast: F=9.740461873303655, p=0.00560528853174, df_denom=9,
 |       df_num=2>
 |
 |      Alternatively, you can specify the hypothesis tests using a string
 |
 |      >>> from statsmodels.datasets import longley
 |      >>> from statsmodels.formula.api import ols
 |      >>> dta = longley.load_pandas().data
 |      >>> formula = 'TOTEMP ~ GNPDEFL + GNP + UNEMP + ARMED + POP + YEAR'
 |      >>> results = ols(formula, dta).fit()
 |      >>> hypotheses = '(GNPDEFL = GNP), (UNEMP = 2), (YEAR/1829 = 1)'
 |      >>> f_test = results.new_f_test(hypotheses)
 |      >>> print f_test
 |
 |      See also
 |      --------
```

```
 |      statsmodels.contrasts
 |      statsmodels.model.t_test
 |      patsy.DesignInfo.linear_constraint
 |
 |      Notes
 |      -----
 |      The matrix `r_matrix` is assumed to be non-singular. More precisely,
 |
 |      r_matrix (pX pX.T) r_matrix.T
 |
 |      is assumed invertible. Here, pX is the generalized inverse of the
 |      design matrix of the model. There can be problems in non-OLS models
 |      where the rank of the covariance of the noise is not full.
 |
 |  normalized_cov_params(self)
 |
 |  remove_data(self)
 |      remove data arrays, all nobs arrays from result and model
 |
 |      This reduces the size of the instance, so it can be pickled with less
 |      memory. Currently tested for use with predict from an unpickled
 |      results and model instance.
 |
 |      .. warning:: Since data and some intermediate results have been removed
 |         calculating new statistics that require them will raise exceptions.
 |         The exception will occur the first time an attribute is accessed that
 |         has been set to None.
 |
 |      Not fully tested for time series models, tsa, and might delete too much
 |      for prediction or not all that would be possible.
 |
 |      The list of arrays to delete is maintained as an attribute of the
 |      result and model instance, except for cached values. These lists could
 |      be changed before calling remove_data.
 |
 |  save(self, fname, remove_data=False)
 |      save a pickle of this instance
 |
 |      Parameters
 |      ----------
 |      fname : string or filehandle
 |          fname can be a string to a file path or filename, or a filehandle.
 |      remove_data : bool
 |          If False (default), then the instance is pickled without changes.
 |          If True, then all arrays with length nobs are set to None before
 |          pickling. See the remove_data method.
 |          In some cases not all arrays will be set to None.
 |
 |      Notes
 |      -----
 |      If remove_data is true and the model result does not implement a
 |      remove_data method then this will raise an exception.
 |
 |  t(self, column=None)
 |      deprecated: Return the t-statistic for a given parameter estimate.
 |
 |      FutureWarning: use attribute tvalues instead, t will be removed
 |      in the next release
 |
 |      Parameters
 |      ----------
 |      column : array-like
 |          The columns for which you would like the t-value.
 |          Note that this uses Python's indexing conventions.
 |
 |      See also
 |      ---------
 |      Use t_test for more complicated t-statistics.
 |
 |      Examples
 |      --------
 |      >>> import statsmodels.api as sm
 |      >>> data = sm.datasets.longley.load()
 |      >>> data.exog = sm.add_constant(data.exog)
```

```
|      >>> results = sm.OLS(data.endog, data.exog).fit()
|      >>> results.tvalues
|      array([ 0.17737603, -1.06951632, -4.13642736, -4.82198531, -0.22605114,
|      4.01588981, -3.91080292])
|      >>> results.tvalues[[1,2,4]]
|      array([-1.06951632, -4.13642736, -0.22605114])
|      >>> import numpy as np
|      >>> results.tvalues[np.array([1,2,4]]
|      array([-1.06951632, -4.13642736, -0.22605114])
|
|  t_test(self, r_matrix, q_matrix=None, cov_p=None, scale=None)
|      Compute a t-test for a joint linear hypothesis of the form Rb = q
|
|      Parameters
|      ----------
|      r_matrix : array-like, str, tuple
|          - array : If an array is given, a p x k 2d array or length k 1d
|            array specifying the linear restrictions.
|          - str : The full hypotheses to test can be given as a string.
|            See the examples.
|          - tuple : A tuple of arrays in the form (R, q), since q_matrix is
|            deprecated.
|      q_matrix : array-like or scalar, optional
|          This is deprecated. See `r_matrix` and the examples for more
|          information on new usage. Can be either a scalar or a length p
|          row vector. If omitted and r_matrix is an array, `q_matrix` is
|          assumed to be a conformable array of zeros.
|      cov_p : array-like, optional
|          An alternative estimate for the parameter covariance matrix.
|          If None is given, self.normalized_cov_params is used.
|      scale : float, optional
|          An optional `scale` to use.  Default is the scale specified
|          by the model fit.
|
|      Examples
|      --------
|      >>> import numpy as np
|       >>> import statsmodels.api as sm
|       >>> data = sm.datasets.longley.load()
|       >>> data.exog = sm.add_constant(data.exog)
|       >>> results = sm.OLS(data.endog, data.exog).fit()
|       >>> r = np.zeros_like(results.params)
|       >>> r[4:6] = [1,-1]
|       >>> print r
|       [ 0.  0.  0.  0.  1. -1.  0.]
|
|       r tests that the coefficients on the 5th and 6th independent
|       variable are the same.
|
|       >>>T_Test = results.t_test(r)
|       >>>print T_test
|       <T contrast: effect=-1829.2025687192481, sd=455.39079425193762,
|       t=-4.0167754636411717, p=0.0015163772380899498, df_denom=9>
|       >>> T_test.effect
|       -1829.2025687192481
|       >>> T_test.sd
|       455.39079425193762
|       >>> T_test.t
|       -4.0167754636411717
|       >>> T_test.p
|       0.0015163772380899498
|
|       Alternatively, you can specify the hypothesis tests using a string
|
|       >>> dta = sm.datasets.longley.load_pandas().data
|       >>> formula = 'TOTEMP ~ GNPDEFL + GNP + UNEMP + ARMED + POP + YEAR'
|       >>> results = ols(formula, dta).fit()
|       >>> hypotheses = 'GNPDEFL = GNP, UNEMP = 2, YEAR/1829 = 1'
|       >>> t_test = results.new_t_test(hypotheses)
|       >>> print t_test
|
|       See also
|       ---------
|       tvalues : individual t statistics
|      f_test : for F tests
|      patsy.DesignInfo.linear_constraint
|
|  ----------------------------------------------------------------------
|  Class methods inherited from statsmodels.base.model.LikelihoodModelResults:
|
|  load(cls, fname) from __builtin__.type
|      load a pickle, (class method)
|
|      Parameters
|      ----------
|      fname : string or filehandle
|          fname can be a string to a file path or filename, or a filehandle.
|
|      Returns
|      -------
|      unpickled instance
|
|  ----------------------------------------------------------------------
|  Data descriptors inherited from statsmodels.base.model.LikelihoodModelResults:
|
|  llf
|
|  tvalues
|
|  ----------------------------------------------------------------------
|  Methods inherited from statsmodels.base.model.Results:
|
|  initialize(self, model, params, **kwd)
|
|  predict(self, exog=None, transform=True, *args, **kwargs)
|      Call self.model.predict with self.params as the first argument.
|
|      Parameters
|      ----------
|      exog : array-like, optional
|          The values for which you want to predict.
|      transform : bool, optional
|          If the model was fit via a formula, do you want to pass
|          exog through the formula. Default is True. E.g., if you fit
|          a model y ~ log(x1) + log(x2), and transform is True, then
|          you can pass a data structure that contains x1 and x2 in
|          their original form. Otherwise, you'd need to log the data
|          first.
|
|      Returns
|      -------
|      See self.model.predict
|
|  ----------------------------------------------------------------------
|  Data descriptors inherited from statsmodels.base.model.Results:
|
|  __dict__
|      dictionary for instance variables (if defined)
|
|  __weakref__
|      list of weak references to the object (if defined)
```