Statsmodels 基础

金　林
中南财经政法大学统计系
jinlin82@qq.com

2020 年 2 月 16 日

Facts

1. Initial release: 2009
2. Website: https://www.statsmodels.org/stable/index.html
3. History:
    1. The models module of scipy.stats was originally written by Jonathan Taylor.
    2. For some time it was part of scipy but was later removed.
    3. During the Google Summer of Code 2009, statsmodels was corrected, tested, improved and released as a new package.
    4. Since then, the statsmodels development team has continued to add new models, plotting tools, and statistical methods.

What is statsmodels

1. statsmodels is a Python module that provides classes and functions for the estimation of many different statistical models,
2. as well as for conducting statistical tests, and statistical data exploration.
3. An extensive list of result statistics are available for each estimator.
4. The results are tested against existing statistical packages to ensure that they are correct.

A simple example using ordinary least squares

```python
import numpy as np
import statsmodels.api as sm
import statsmodels.formula.api as smf

dat = sm.datasets.get_rdataset("Guerry", site="C:/github_repo/
    Rdatasets", package="HistData").data
results = smf.ols('Lottery ~ Literacy + np.log(Pop1831)', data=
    dat).fit()
print(results.summary())

nobs = 100
X = np.random.random((nobs, 2))
X = sm.add_constant(X)
beta = [1, .1, .5]
e = np.random.random(nobs)
y = np.dot(X, beta) + e

results = sm.OLS(y, X).fit()
print(results.summary())
```

- Import Paths and Structure
- Fitting models using R-style formulas

two ways of importing

1. two ways of importing functions and classes from statsmodels:
    1. API import for interactive use: Allows tab completion
    2. Direct import for programs: Avoids importing unnecessary modules and commands

API Import for interactive use

1. For interactive use the recommended import is:
   `import statsmodels.api as sm`
2. Importing statsmodels.api will load most of the public parts of statsmodels.
3. This makes most functions and classes conveniently available within one or two levels, without making the “sm” namespace too crowded.
4. list functions with `dir(sm)`

Direct import for programs

1. statsmodels submodules are arranged by topic (e.g. discrete for discrete choice models, or tsa for time series analysis).

   Functions and classes:

```
    from statsmodels.regression.linear_model import OLS, WL
from statsmodels.tools.tools import rank, add_constant
```

   Modules

```
    from statsmodels.datasets import macrodata
 import statsmodels.stats import diagnostic
```

   Modules with aliases

```
    import statsmodels.regression.linear_model as lm
 import statsmodels.stats.diagnostic as smsdia
import statsmodels.stats.outliers_influence as oi
```

- Import Paths and Structure
- Fitting models using R-style formulas

R-style formulas and pasty

1. statsmodels allows users to fit statistical models using R-style formulas.
2. Internally, statsmodels uses the patsy package to convert formulas and data to the matrices that are used in model fitting.
3. patsy is a Python package for describing statistical models (especially linear models, or models that have a linear component) and building design matrices. It is closely inspired by and compatible with the formula mini-language used in R and S.
4. For instance, if we have some variable y, and we want to regress it against some other variables x, a, b, and the interaction of a and b, then we simply write:

```
patsy.dmatrices("y ~ x + a + b + a:b", data)
```

dmatrices

1. split the categorical Region variable into a set of indicator variables.
2. added a constant to the exogenous regressors matrix.
3. returned pandas DataFrames instead of simple numpy arrays. This is useful because DataFrames allow statsmodels to carry-over meta-data (e.g. variable names) when reporting results.

Model fit and and summary

1. Fitting a model in statsmodels typically involves 3 easy steps:
    1. Use the model class to describe the model
    2. Fit the model using a class method
    3. Inspect the results using a summary method
    4. Type `dir(res)` for a full list of attributes.

```
mod = sm.OLS(y, X)     # Describe model
res = mod.fit()        # Fit model
print(res.summary())    # Summarize model
```

main statsmodels API

1. `statsmodels.api` : Cross-sectional models and methods. Canonically imported using `import statsmodels.api as sm.`
2. `statsmodels.tsa.api` : Time-series models and methods. Canonically imported using `import statsmodels.tsa.api as tsa.`
3. `statsmodels.formula.api` : A convenience interface for specifying models using formula strings and DataFrames. This API directly exposes the from_formula class method of models that support the formula API. Canonically imported using `import statsmodels.formula.api as smf`
4. The API focuses on models and the most frequently used statistical test, and tools.

statsmodels.api

1. Regression
2. Imputation
3. Generalized Estimating Equations
4. Generalized Linear Models
5. Discrete and Count Models
6. Multivariate Models
7. Misc Models
8. Graphics
9. Tools

statsmodels.tsa.api

1. Statistics and Tests
2. Univariate Time-Series Analysis
3. Exponential Smoothing
4. Multivariate Time Series Models
5. Filters and Decompositions
6. Markov Regime Switching Models
7. Time-Series Tools
8. X12/X13 Interface

statsmodels.formula.api

1. The function descriptions of the methods exposed in the formula API are generic. See the documentation for the parent model for details.
    - gls(formula, data[, subset, drop_cols])
    - wls(formula, data[, subset, drop_cols])
    - ols(formula, data[, subset, drop_cols])
    - mixedlm(formula, data[, re_formula, ...])
    - glm(formula, data[, subset, drop_cols])
    - mnlogit(formula, data[, subset, drop_cols])
    - logit(formula, data[, subset, drop_cols])
    - probit(formula, data[, subset, drop_cols])
    - poisson(formula, data[, subset, drop_cols])
    - negativebinomial(formula, data[, subset, ...])
    - quantreg(formula, data[, subset, drop_cols])
    - ordinal_gee(formula, groups, data[, subset, ...])
    - nominal_gee(formula, groups, data[, subset, ...])
    - gee(formula, groups, data[, subset, time, ...])
    - glmgam(formula, data[, subset, drop_cols])

The Datasets Package

1. statsmodels provides data sets (i.e. data and meta-data) for use in examples, tutorials, model testing, etc.
2. load statsmodels Available Datasets
3. Using Datasets from Stata
4. Using Datasets from R

statsmodels Available Datasets

1. Available Datasets list: `Anaconda3\Lib\site-packages\statsmodels\datasets`
2. Load a dataset: `sm.datasets.datasets_name.load_pandas()`
3. Loading data as pandas objects: `load_pandas()` method
4. The Dataset object follows the bunch pattern. The full dataset is available in the `data` attribute.
5. Most datasets hold convenient representations of the data in the attributes `endog` and `exog`
6. Univariate datasets, however, do not have an `exog` attribute.
7. Variable names can be obtained by typing: `endog_name` and `exog_name`

例子

```python
import statsmodels.api as sm
dat = sm.datasets.longley.load_pandas()

dat.data
dat.endog
dat.exog
dat.endog_name
dat.exog_name
dat.names
```

Using Datasets from Stata

1. `webuse(data[, baseurl, as_df])` : Download and return an example dataset from Stata.

```python
import statsmodels.api as sm

auto=sm.datasets.webuse('auto')
```

Using Datasets from R

1. The Github Rdatasets project gives access to the datasets available in R' s core datasets package and many other common R packages.
2. All of these datasets are available to statsmodels by using the `get_rdataset` function.
3. The actual data is accessible by the `data` attribute.
4. __doc__ 属性可以查看数据帮助信息
5. 由于 Github 网站文件库被封，可以采取以下方法解决：
   1. 把 Rdatasets 项目下载到本地计算机.
   2. 修改 `Anaconda3\Lib\site-packages\statsmodels\datasets\utils.py` 中的 `get_rdataset` 函数.

```python
import statsmodels.api as sm

iris=sm.datasets.get_rdataset('iris', site="C:/github_repo/
    Rdatasets")
iris.data
print(iris.__doc__)
```