ARIMA

July 21, 2017

1 ARIMA Models with Python

Autoregressive Integrated Moving Average (ARIMA) models are a forecasting technique that projects the future values of a series. ARIMA is particularly useful when the data is reasonably long (38+ points) and the correlation between past observations is stable.

2 statsmodels.tsa.arima_model

```
import statsmodels.tsa.arima_model
```

2.1 parameters

class statsmodels.tsa.arima_model.ARIMA(endog, order, exog=None, dates=None, freq=None, missing='none')
endog: the endogenous variable
order: the order of the model for the number of AR parameters, differences, and MA parameters to use
exog: optional array for exogenous variables
dates: collection of datetime object
freq: optional string for the frequency of the time series

2.2 attributes

endog_names: names of endogenous variables
exog_names: names of exogenous variables

2.2.1 ARIMAResults

aic()
arfreq() # Returns the frequency of the AR roots
arparams()
arroots()
bic()
bse()
conf_int([alpha, cols, method]) # Returns the confidence interval of the fitted parameters
cov_params()
f_test(r_matrix[, cov_p, scale, invcov]) # Computes the F-test for a joint linear hypothesis

fittedvalues()
forecast([steps, exog, alpha])
hqic()
`initialize(model, params, **kwd)`
llf()
load(fname) # loads a pickle
mafreq() #Returns the frequency of the MA roots.
maparams()
maroots()
normalized_cov_params()
plot_predict([start, end, exog, dynamic, ...]) # plots forecasts
predict([start, end, exog, typ, dynamic]) # ARIMA model in-sample and out-of-sample prediction
pvalues()
remove_data() # removes nobs arrays from result and model
resid()
save(fname[, remove_data]) # saves a pickle of this instance
summary([alpha]) # summarizes the model
summary2([title, alpha, float_format]) # experimental summary function for ARIMA Results
t_test(r_matrix[, cov_p, scale, use_t]) # computes a t-test for each linear hypothesis in the form Rb = q
tvalues() # returns the t-statistic for a given parameter estimate
wald_test(r_matrix[, cov_p, scale, invcov, ...]) # computes a Wald-test for a joint linear hypothesis
wald_test_terms([skip_single, ...]) # computes a sequence of Wald tests for terms over multiple columns

2.2.2 scores

ARIMA.score(params) # computes the score function

2.2.3 predict

ARIMA.predict(params, start=None, end=None, exog=None, typ='linear', dynamic=False) # predicts values

2.2.4 loglike

ARIMA.loglike(params, set_sigma2=True) # computes the log-likelihood

2.2.5 loglike_kalman

ARIMA.loglike_kalman(params, set_sigma2=True) # computes exact loglikelihood with the Kalman Filter

2.2.6 loglike_css

ARIMA.loglike_css(params, set_sigma2=True) # computes the loglikelihood with Conditional Sum of Squares likelihood function.

2.2.7 initialize

ARIMA.initialize() # initializes a model instance

2.2.8 information

ARIMA.information(params) # provides fisher information matrix of model

2.2.9 hessian

ARIMA.hessian(params) # computes the Hessian

2.2.10 geterrors

ARIMA.geterrors(params)

2.2.11 from_formula

`ARIMA.from_formula(formula, data, subset=None, drop_cols=None, *args, **kwargs)` # creates a Model from a formula and dataframe

2.2.12 fit()

`ARIMA.fit(start_params=None, trend='c', method='css-mle', transparams=True, solver='lbfgs', maxiter=50, full_output=1, disp=5, callback=None, start_ar_lags=None, **kwargs)` # fits ARIMA(p,d,q) model by exact maximum likelihood via Kalman filter

In [ ]: