|  |
| --- |
| National Bank of Belgium |
| JD+ |
| Quality diagnostics |
|  |
| **Jean Palate** |
| **9/5/2016** |

JDemetra+ provides a set of quality diagnostics on seasonal adjustment. In the first part of this document, we shortly describe them. In the second part, we consider the design of the classes that provide the diagnostics. We also explain how the current diagnostics can be extended.

## Generalities

The quality diagnostics that can be built on the different seasonal adjustment procedures are very heterogeneous. Moreover, their interpretation might be difficult for many users. That is why we have chosen to give a summary of information they provide by means of a very simple qualitative indicator, which is defined in the next table. That indicator is used in the batch processing module of JD+. Such a simple approach doesn't prevent that much more complicated and much richer information could be provided to the user through more sophisticated interface. The interactive module offers many details on the different diagnostics.

*Meaning of the quality indicator[[1]](#footnote-1)*

|  |  |
| --- | --- |
| ***Value*** | ***Meaning*** |
| Undefined | The quality is undefined: unprocessed test, meaningless test, failure in the computation of the test... |
| **Error** | There is an error in the results. The processing should be rejected (for instance, it contains aberrant values or some numerical constraints are not fulfilled |
| **Severe** | There is no logical error in the results but they should not be accepted for some statistical reasons |
| Bad | The quality of the results is bad, following a specific criterion, but there is no actual error and the results could be used. |
| Uncertain | The result of the test is uncertain. Consider it with caution |
| Good | The result of the test is good |

Several qualitative indicators can be combined following the next rules.

Given a set of n diagnostics, the sum of the results is:

|  |  |
| --- | --- |
| ***Sum*** | ***Rules*** |
| Undefined | All diagnostics are Undefined |
| Error | There is at least 1 error |
| Severe | There is at least 1 "severe" diagnostic but no error |
| Bad | No error, no severe diagnostics; the average of the (defined) diagnostics (Bad=1, Uncertain=2, Good=3) is < 1.5 |
| Uncertain | No error, no severe diagnostics; the average of the (defined) diagnostics (Bad=1, Uncertain=2, Good=3) is in [1.5, 2.5[ |
| Good | No error, no severe diagnostics; the average of the (defined) diagnostics (Bad=1, Uncertain=2, Good=3) is ≥ 2.5 |

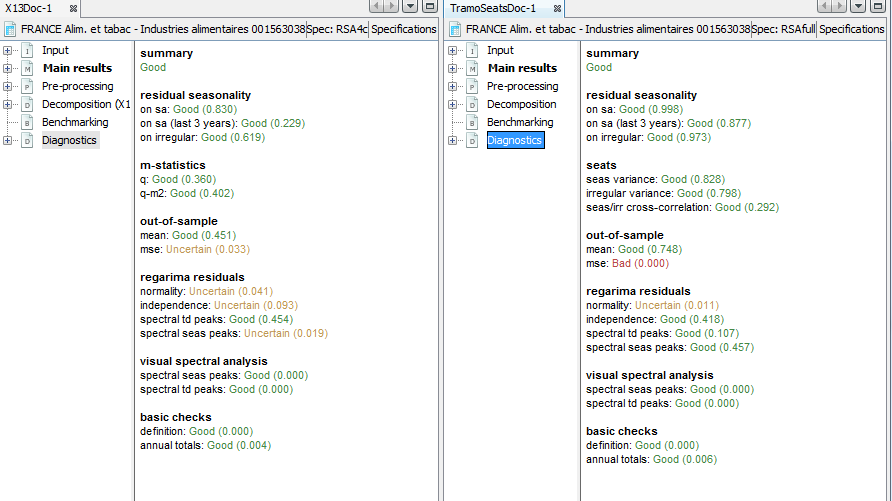
So, errors and severe diagnostics are absorbent results. The global "quality" indicator of the seasonal adjustments displayed in the multi-processing window is the sum of all the defined quality diagnostics, using the rules defined above.

Finally, diagnostics can throws warnings, which are indicated by exclamation marks and tooltips in the multi-processing output panel.

***Users should consider the quality indicator as a tool to detect rapidly possible problems in a large set of processing. For important series, a more complete examination of the results should always be considered.***

## Description of the diagnostics of JD+

The different diagnostics are put in several groups, corresponding to different modules (classes).



The current software contains diagnostics on the coherence of the decomposition ("Basic checks" group), on visual spectral inspection ("Visual spectral analysis"), on the residuals of the RegArima pre-processing ("RegArima residuals" group), on the residual seasonality ("Residual seasonality" group) and on the decomposition ("Seats" group for Tramo-Seats, M-Statistics group for X12).

Most of them use parameters (usually thresholds) that can be modified by means of the options dialog box (**TO DO**). Finally, each group of diagnostics might be disabled, when it is considered as meaningless.

We describe below the different items of the diagnostics, using the default options.

### Basic checks

#### Definition

A first set of diagnostics consists in verifying that the definition constraints implied by the model of the series are well respected (see the description of the model for more details).

The maximum of the absolute differences is computed for the different equations and related to the Euclidean norm of the initial series (Q).

*Results of the test*

|  |  |
| --- | --- |
| ***Q (see above)*** | ***Diagnostic*** |
| > 0.000001 | **Error** |
| <= 0.000001 | Good |

#### Annual totals

The annuals totals of the original series and those of the seasonally adjusted series are compared.

The maximum of their absolute differences is computed and related to the Euclidean norm of the initial series.

*Results of the test*

|  |  |
| --- | --- |
| ***Q (see above)*** | ***Diagnostic*** |
| > 0.5 | **Error** |
| ]0.1, 0.5] | **Severe** |
| ]0.05, 0.1] | Bad |
| ]0.01, 0.05] | Uncertain |
| <=0.01 | Good |

The fact the test above throws an error doesn’t mean that the computation is wrong. It could also indicate some limits of the method. See below for an actual example.

## production of sugar in Belgium

#### Warnings

A warning is thrown when the series is short (less than 7 years)

### Visual spectral analysis

The visual spectral analysis used in Demetra+ follows the method developed at the US Census Bureau. The default spectrum estimator used to detect seasonal and trading day effects is an autoregressive spectral estimator , expressed in decibel units (see appendix 1).

The visual inspection method consider the frequencies . An empirically criterion of ”visual significance” is determined as follows. To be ”visually significant”, the value at a trading day or seasonal frequency must be above the median of the plotted values of and must be larger than both neighboring values by at least , with by default.

It should be noted that the auto-regressive diagnostics of JD+ are computed on the last 8 years of each series as it is done in X12 (but contrary to the other diagnostics).

*Results of the test*

|  |  |
| --- | --- |
| ***Presence of a visual peeks*** | ***Diagnostic*** |
| On irregular and on sa | **Severe** |
| On irregular or on sa | Bad |
| No visual peek | Good |

#### Warnings

A warning is displayed when the differenced original series doesn't contain seasonal peak, which means that it should probably not be seasonally adjusted.

### Residuals diagnostics

Several tests are computed on the residuals of the RegArima model. The exact definition of what we mean by "residuals" should be clarified. Indeed, X12 and Tramo are based on different estimation procedures of the likelihood of the RegArima models, which lead to slightly different definitions of the residuals.

In most cases, the different sets of residuals yield slightly different diagnostics. However, their global messages are near always very similar.

JD+ uses a solution - "the full residuals" - which is also available in Tramo.

#### Normality test

The joint normality test (which combines skewness and kurtosis tests) is the Doornik-Hansen test (see appendix 3), which is distributed as a .

*Results of the test*

|  |  |
| --- | --- |
| ***Pr(>val)*** | ***Diagnostic*** |
| <0.01 | Bad |
| [0.01, 0.1[ | Uncertain |
| ≥0.1 | Good |

#### Independence test

The independence test is the Ljung-Box test (see appendix 4), which is distributed as ,

where k depends on the frequency of the series (24 for monthly series, 8 for quarterly series, 4\*freq for other frequencies) and np is the number of hyper-parameters of the model (number of parameters in the Arima model)

*Results of the test*

|  |  |
| --- | --- |
| ***Pr(>val)*** | ***Diagnostic*** |
| <0.01 | Bad |
| [0.01, 0.1[ | Uncertain |
| ≥0.1 | Good |

### Spectral tests

The software provides tests based on the periodogram of the residuals, for the trading days frequencies and for the seasonal frequencies.

The periodogram is computed at the so-called Fourier frequencies, which present good statistical properties. Under the hypothesis of Gaussian white noise of the residual, it is possible to derive simple test on the periodogram, around specific (groups of) frequencies. The exact definition and the used test are described in the appendix 5.

*Results of the test*

|  |  |
| --- | --- |
| ***P(stat>val)*** | ***Diagnostic*** |
| <0.001 | **Severe** |
| [0.001, 0.01[ | Bad |
| [0.01, 0.1[ | Uncertain |
| ≥0.1 | Good |

### Out-of-sample diagnostics

The out-of-sample diagnostics follow the method developed in Tramo.

Using the linearized series of the model estimated on the whole series, the model (ARIMA [+ mean]) are re-estimated on a shorter time span (the last 1.5 year is dropped).

The in-sample errors (n[[2]](#footnote-2)-nback data) and of the out-of-sample errors (nback) are computed by the Kalman filter (one-step-ahead forecast errors). Their sample mean and variance are then computed.

The means tests compare the sample means to 0, using the in-sample variance.

The variance test is the usual F test of equality of two sample variances. That test should be used with caution (when the distribution of the residuals is far to be normal).

*Results of the test*

|  |  |
| --- | --- |
| ***Pr(>val) or Pr(>val)*** | ***Diagnostic*** |
| <0.01 | Bad |
| [0.01, 0.1[ | Uncertain |
| ≥0.1 | Good |

### Seats diagnostics

JD+ provides some model-based diagnostics for Seats, similar to those provided in the original program. They correspond to measures of over/under estimation of the seasonal and of the irregular components and of their cross-correlation. The variances of the theoretical estimators of the (stationary) components and of their estimates are compared; the Bartlett's approximation is used to build statistical tests on those measures. More detailed information, on the other components and on their auto-correlation functions, can be found in the interactive module.

It should be noted that the considered models are those of the final estimators and that the complete time span of the estimates is used, though the (preliminary) models for the first and for the last observations can present significantly different properties.

*Results of the test*

|  |  |
| --- | --- |
| ***Pr(N>val)*** | ***Diagnostic*** |
| <0.01 | Bad |
| [0.01, 0.05[ | Uncertain |
| ≥0.5 | Good |

#### Warnings

Warning are also displayed when parameters were modified (quasi-unit roots in the moving average polynomials) or/and when a non-decomposable model was changed by Seats.

### X11 diagnostics (M-statistics)

The M-diagnostics correspond to the statistics “Q” and “Q-M2” developed by the US Census Bureau.

See for instance Ladiray-Quenneville [1999] for a complete description of the tests.

*Results of the test*

|  |  |
| --- | --- |
| ***M*** | ***Diagnostic*** |
| ≥2 | **Severe** |
| [1, 2[ | Bad |
| <1 | Good |

### Residual seasonality diagnostics

The residual seasonality diagnostic corresponds to the test developed in X12/X13

The F-Test on stable seasonality (see appendix 6) is computed on the differences of the seasonally adjusted series (component CSA, see above) and on the irregular component (CI, see above).

The differencing is done with a lag of 3 periods for monthly series and with a lag of 1 period in the other cases. For the seasonally adjusted series, one test is computed on the complete time span and another one on the last 3 years.

*Results of the test*

|  |  |
| --- | --- |
| ***Pr(F>val)*** | ***Diagnostic*** |
| <0.01 | **Severe** |
| [0.01, 0.05[ | Bad |
| [0.05, 0.1[ | Uncertain |
| ≥0.1 | Good |

**TO DO**

*Maravall has introduced in the last releases of Tramo numerous tests on seasonality, which should complete this diagnostics. See also D. Findley (internal document). Most of the new tests are already displayed in the details of the diagnostics. We could complete the current diagnostics by the QS test of Maravall and by the fixed seasonal effects test on the last years (F-test) of the US census Bureau (probably the two most robust tests).*

## Design

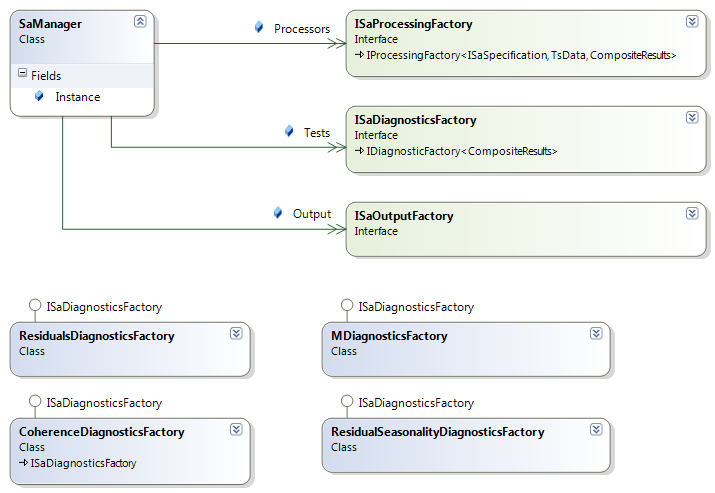
For each set of diagnostics, JD+ contains 3 classes. By convention, they names are XXXDiagnostics,

XXXDiagnostcsFactory, XXXDiagnosticsConfiguration

The configuration class contains the entire settings specific to a set of diagnostics; the factory, which contains a configuration object, must be able to create a corresponding diagnostics object for a given SA processing.

All the current diagnostic factories are stored in the central **SaManger** instance.

The definition of the different classes are formalized by several generic interfaces, as displayed in the next diagram



### Implementation classes

We list below the current implementations of diagnostics on seasonal adjustment

|  |  |
| --- | --- |
| **Diagnostics** | **Classes** |
| Residual seasonality diagnostics | ec.tss.sa.ResidualSeasonalityDiagnostics |
| Seats diagnostics | ec.tss.sa.SeatsDiagnostics |
| M\_Statistics | ec.tss.sa.MDiagnostics |
| Out of sample diagnostics | ec.tss.sa.OutOfSampleDiagnostics |
| Visual spectral peaks | ec.tss.sa.SpectralDiagnostics |
| Outliers | ec.tss.sa.OutliersDiagnostics |
| RegArima residuals diagnostics | ec.tss.sa.ResidualsDiagnostics |
| Basic checks | ec.tss.sa.CoherenceDiagnostics |

### Adding a new diagnostic

We explain below the different steps to add a completely new set of diagnostics. The example will use the new seasonality tests of Tramo to provide Qs tests and F-tests on regression models with seasonal dummies applied to the seasonally adjusted and on the irregular series.

1. Define a configuration class that will contain the parameters of the test (to be displayed in the future in a graphical interface). That class should remain very light. Moreover, it should implement the Cloneable interface.

|  |
| --- |
| public class MyDiagnosticsConfiguration implements Cloneable {  public static final double SEV = .001, BAD = .01, UNC = .05;  private double sev\_ = SEV, bad\_ = BAD, unc\_ = UNC;  private boolean enabled\_ = true;  @Override  public MyDiagnosticsConfiguration clone() {  try {  return (MyDiagnosticsConfiguration) super.clone();  } catch (CloneNotSupportedException ex) {  return null;  }  }  public double getSevereThreshold() {  return sev\_;  }  public double getBadThreshold() {  return bad\_;  }  public double getUncertainThreshold() {  return unc\_;  }  public boolean isEnabled() {  return enabled\_;  }  public void setEnabled(boolean enabled) {  enabled\_ = enabled;  }  …  } |

1. Define a factory class that will create the actual diagnostics, using a given configuration

|  |
| --- |
| public class MyDiagnosticsFactory implements ISaDiagnosticsFactory {    static final String NAME="New Seasonality tests", DESC="New Seasonality tests";  private MyDiagnosticsConfiguration config\_;  public MyDiagnosticsFactory() {  config\_ = new MyDiagnosticsConfiguration();  }  public MyDiagnosticsFactory(MyDiagnosticsConfiguration config) {  config\_ = config;  }  @Override  public Scope getScope() {//Used to organize the diagnostics  return Scope.Final;  }  @Override  public int getOrder() {{//Used to organize the diagnostics  return 0; //High priority  }  @Override  public void dispose() {  }  @Override  public String getName() {  return NAME;  }  @Override  public String getDescription() {  return DESC;  }  @Override  public boolean isEnabled() {  return config\_.isEnabled();  }  @Override  public void setEnabled(boolean enabled) {  config\_.setEnabled(enabled);  }  @Override  public Object getProperties() { // To be shown in a graphical interface  return config\_.clone();  }  @Override  public void setProperties(Object obj) {// Set the new configuration  if (obj instanceof MyDiagnosticsConfiguration) {  MyDiagnosticsConfiguration nconfig = (MyDiagnosticsConfiguration) obj;  config\_ = nconfig.clone();  }  }  @Override  public IDiagnostics create(CompositeResults rslts) { // Generate the diagnostics  return MyDiagnostics.create(rslts, config\_);  }  } |

1. Define the actual diagnostics class

|  |
| --- |
| public class MyDiagnostics implements IDiagnostics {  static final String QS\_SA = "Qs test on SA", QS\_I = "Qs test on I", FTEST\_SA = "F-Test on SA (seasonal dummies)", FTEST\_I = "F-Test on I (seasonal dummies)";  static final String[] ALL = new String[]{QS\_SA, QS\_I, FTEST\_SA, FTEST\_I};  private StatisticalTest qs\_sa, qs\_i, f\_sa, f\_i;  // All the computations are done here  static IDiagnostics create(CompositeResults rslts, MyDiagnosticsConfiguration config) {  try {  MyDiagnostics test = new MyDiagnostics();  TsData sa = rslts.getData(ModellingDictionary.SA\_LIN, TsData.class);  TsData i = rslts.getData(ModellingDictionary.I\_LIN, TsData.class);  if (sa == null && i == null) {  return null;  }  if (sa != null) {  SeasonalityTests satest = SeasonalityTests.seasonalityTest(sa, 1, true, true);  test.qs\_sa = satest.getQs();  FTest F = new FTest();  if (F.test(sa)) {  test.f\_sa = F.getFTest();  }  }  if (i != null) {  SeasonalityTests itest = SeasonalityTests.seasonalityTest(i, 0, true, true);  test.qs\_i = itest.getQs();  FTest F = new FTest();  if (F.test(i)) {  test.f\_i = F.getFTest();  }  }  return test;  } catch (Exception err) {  return null;  }  }  @Override  public String getName() { // The name that will appear in the diagnostics  return MyDiagnosticsFactory.NAME;  }  @Override  public List<String> getTests() {  // The tests that will appear in the diagnostics  ArrayList<String> tests = new ArrayList<String>();  if (qs\_sa != null) {  tests.add(QS\_SA);  }  if (f\_sa != null) {  tests.add(FTEST\_SA);  }  if (qs\_i != null) {  tests.add(QS\_I);  }  if (f\_i != null) {  tests.add(FTEST\_I);  }  return tests;  }  @Override  public ProcQuality getDiagnostic(String test) {  // The quality indicator of the given test  switch (test) {  case QS\_SA:  return quality(qs\_sa);  case FTEST\_SA:  return quality(f\_sa);  case QS\_I:  return quality(qs\_i);  case FTEST\_I:  return quality(f\_i);  default:  return ProcQuality.Undefined;  }  }  @Override  public double getValue(String test) {  // The value associated with the given test (displayed in the summary)  switch (test) {  case QS\_SA:  return pvalue(qs\_sa);  case FTEST\_SA:  return pvalue(f\_sa);  case QS\_I:  return pvalue(qs\_i);  case FTEST\_I:  return pvalue(f\_i);  default:  return Double.NaN;  }  }  @Override  public List<String> getWarnings() {  // Possible warnings  return Collections.EMPTY\_LIST;  }  // implementation details  private ProcQuality quality(StatisticalTest test) {  if (test == null) {  return ProcQuality.Undefined;  }  double pval = test.getPValue();  if (pval < .001) {  return ProcQuality.Severe;  } else if (pval < .01) {  return ProcQuality.Bad;  } else if (pval < .05) {  return ProcQuality.Uncertain;  } else {  return ProcQuality.Good;  }  }  private double pvalue(StatisticalTest test) {  return test == null ? Double.NaN : test.getPValue();  }  } |

1. Add an instance of the new diagnostics factory into the current SaManager. The example below uses a feature provided by NetBeans. Other solutions are possible.

|  |
| --- |
| public class Installer extends ModuleInstall{  @Override  public void restored() {  super.restored();  ec.tss.sa.SaManager.instance.add(new MyDiagnosticsFactory());  }    }  // Don’t forget to add the following line in the manifest.mf file of the project  // OpenIDE-Module-Install: be/nbb/demetra/tutorial/plugin/sadiags/Installer.class |

The new diagnostics will be available for any seasonal adjustment processing.

## Appendices

### 1. Auto-regressive spectrum

For the series (for example, the model residuals), autoregressive spectrum estimates (in decibel units) have the form

,

where the coefficient estimates are those of the linear regression of on , with and where is the sample variance of the resulting regression residuals.

JD+ uses, like X12, .

### 2. Doornik-Hansen test.

The Doornik-Hansen is defined as follows:

let s = skweness, k=kurtosis of the n (non missing) residuals

We make the following transformations:

*Transformation of the skewness (D'Agostino)*

*Transformation of the kurtosis (Wilson-Hilferty)*

### 3. Ljung-Box test.

The Ljung-Box test is defined as follows:

let = the sample autocorrelation at rank k, of the n residuals is

If the residuals are random, It should be distributed as where np is the number of hyper-parameters of the model from which the residuals are derived.

### 4. Periodogram

#### Definition of the periodogram

The periodogram of the series is computed as follows:

1. The are standardized

2. The periodogram is computed on the standardized

where

and

#### Periodogram at the Fourier frequencies

The Fourier frequencies are defined by

If the are iid , it is easy to see that the corresponding quantities are iid .

We have indeed that

and

,

so that and are uncorrelated random variables.

#### Test on the periodogram

Under the hypothesis that is a Gaussian white noise, and considering subset J of Fourier frequencies, we have:

If we consider the sets of Fourier frequencies on or near the trading days frequencies on one side and on or near the seasonal frequencies on the other side, we can use the above formula as rough tests on the absence of trading days/seasonal effects in the considered series.

The software considers the Fourier frequencies which are on or near the following frequencies (the two nearest frequencies are chosen):

|  |  |  |
| --- | --- | --- |
| Annual frequency | Seasonal | Trading days |
| 12 | 2π/12, 4π/12, 6π/12, 8π/12, 10π/12 | d |
| 6 | 2π/6, 4π/6 | d |
| 4 | 2π/4 | d, 1.292, 1.850, 2.128 |
| 3 | - | d |
| 2 | - | d |

where d is computed as follows, if s is the frequency of the series:

### 5. Stable seasonality test

The stable seasonality test is a F-test used in the context of a single-factor ANOVA model, where the different categories are defined by the different periods (month, quarter...) of the considered series.

The F-test measures the probability that the observations for each period come from distributions that have the same mean.

If we write , the number of periods by year, the number of observations for the period ( , the total number of observations), we have the following decomposition of the variance:

The test is then

1. The model also contain a flag "Accepted", which simply means that the statistician decided to accept the results, no matter what are the different diagnostics. [↑](#footnote-ref-1)
2. N is adjusted by -1 in the case of a mean effect. [↑](#footnote-ref-2)