

# ADAMS

Advanced **D**ata mining **A**nd **M**achine learning **S**ystem

Module: adams-timeseries



Peter Reutemann

January 8, 2020

©2013-2014



THE UNIVERSITY OF  
**WAIKATO**  
*Te Whare Wānanga o Waikato*



Except where otherwise noted, this work is licensed under  
<http://creativecommons.org/licenses/by-sa/4.0/>

# Contents

<b>1</b>	<b>Introduction</b>	<b>7</b>
<b>2</b>	<b>Flow</b>	<b>9</b>
<b>3</b>	<b>Tools</b>	<b>13</b>
3.1	Timeseries Explorer . . . . .	13
3.2	Weka Explorer . . . . .	17
<b>4</b>	<b>Setup</b>	<b>19</b>
	<b>Bibliography</b>	<b>21</b>



# List of Figures

1.1	Image of random data plus trend, with best-fit line and different smoothings[4]. . . . .	7
3.1	Timeseries Explorer displaying wine sales, raw and smoothed with a Savitzky-Golay filter. . . . .	13
3.2	Wizard: start page. . . . .	14
3.3	Wizard: connection parameters. . . . .	14
3.4	Wizard: SQL queries for obtaining timeseries data and associated meta-data. . . . .	15
3.5	Wizard: selected IDs of timeseries to load. . . . .	15
3.6	Timeseries loaded from database. . . . .	16
3.7	WEKA Explorer displaying forecasts for Australian wine sales. .	17



# Chapter 1

## Introduction

According to Wikipedia<sup>1</sup>, a timeseries is *is a sequence of data points, measured typically at successive points in time spaced at uniform time intervals*. ADAMS offers standalone tools, like the Timeseries Explorer, and flow components to loading, processing and saving of timeseries data. Performing predictions using the Weka timeseries forecasting module, is possible as well. The following sections explain the available functionality in depth.

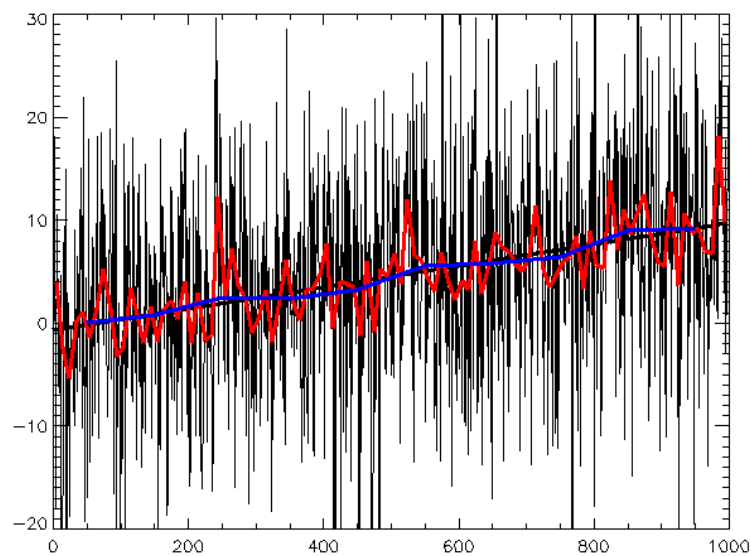


Figure 1.1: Image of random data plus trend, with best-fit line and different smoothings[4].

---

<sup>1</sup>[https://en.wikipedia.org/wiki/Time\\_series](https://en.wikipedia.org/wiki/Time_series)





# Chapter 2

## Flow

The ADAMS flow allows you to comprehensively load, process and save time-series data, as well as make future predictions.

The following data conversions are available:

- *SpreadSheetToTimeseries* – generates a timeseries from two spreadsheet columns (timestamp and value).
- *TimeseriesToArray* – extracts the values of a timeseries to be further analyzed with array statistics, for instance.
- *TimeseriesToSpreadSheet* – turns a timeseries back into spreadsheet.
- *TimeseriesToWekaInstances* – generates data suitable for use with WEKA.
- *WekaForecastContainerToArray* – extracts the WEKA forecasts as an array for further processing.
- *WekaForecastContainerToTimeseries* – turns the WEKA forecasts into a proper timeseries data structure.<sup>1</sup>
- *WekaInstancesToTimeseries* – generates a timeseries from two attributes of a WEKA dataset (timestamp and value).<sup>2</sup>

The following filters are available:

- *TimeseriesChangeResolution* – uses fixed-length interval between data points.
- *TimeseriesDerivative* – calculates a derivative from the timeseries data.
- *TimeseriesEquiDistance* – ensures that the data points are evenly spaced, can also interpolate to create timeseries with the same number of points.
- *TimeseriesFastWavelet* – applies a wavelet transform to a timeseries.
- *TimeseriesFFT* – applies fast fourier transformation to timeseries.
- *TimeseriesHistogram* – generates a histogram from timeseries.
- *TimeseriesLOWESS* – applies LOWESS smoothing [6].
- *TimeseriesResetTimestamps* – lets the timeseries start at a new timestamp.
- *TimeseriesRound* – allows the rounding of the values associated with timestamps in the timeseries.

---

<sup>1</sup>adams-timeseries-build\_and\_use\_forecaster2.flow

<sup>2</sup>adams-timeseries-build\_and\_use\_forecaster2.flow

- *TimeseriesRowNorm* – applies row normalization to the timeseries.
- *TimeseriesSavitzkyGolay* – transforms the timeseries using the Savitzky-Golay filter (smoothing and differentiation) [5]
- *TimeseriesSAX* – performs Symbolic Aggregate approximation.
- *TimeseriesSetStart* – shifts all data points relatively to a new starting point for the timeseries.
- *TimeseriesShiftValues* – shifts the timeseries y values up or down.
- *TimeseriesValueSubset* – first block of values that fits inside defined y range.
- *TimeseriesWindow* – extracts a specified window (or the inverse) from a timeseries.

The following outlier detectors are available:

- *TimeseriesMinPoints* – requires the timeseries to have a certain number (absolute or percentage) of values above a specified value.
- *TimeseriesRange* – ensures that y values lie within defined range
- *TimeseriesTimestampCheck* – checks whether timeseries point is before/after specified timestamp

The following smoothers are available:

- *TimeseriesLOWESSBased* – uses a LOWESS filter for smoothing [6].
- *TimeseriesSavitzkyGolayBased* – uses the *SavitzkyGolay* filter to smooth the timeseries.[5]
- *TimeseriesSlidingWindow* – wrapper smoother that applies the base smoother to a sliding window.

The following sources are available:

- *TimeseriesDbReader* – reads timeseries data from any JDBC database (required columns: ID, timestamp, value).
- *WekaForecasterSetup* – contains the configuration for a WEKA forecaster, i.e., classifier and how the meta-dataset is generated.<sup>3</sup>
- *WekaForecasting* – uses a trained and primed Weka forecaster model from storage to generate a fixed number of predictions.<sup>4</sup>

The following transformers are available:

- *MakeForecastPlotContainer* – turns predictions generated by a WEKA forecaster into plotable data.<sup>5</sup>
- *SpreadSheetRowToTimeseries* – similar to the *TimeseriesDbReader* source, this transformer generates a timeseries container each of the rows in a spreadsheet.
- *SpreadSheetToTimeseries* – similar to the *TimeseriesDbReader* source, this transformer generates one or more timeseries containers from specified columns in a spreadsheet.

---

<sup>3</sup>adams-timeseries-build\_and\_use\_forecaster.flow,      adams-timeseries-sliding\_window.flow,  
adams-timeseries-use\_saved\_forecaster.flow

<sup>4</sup>adams-timeseries-build\_and\_use\_forecaster.flow,      adams-timeseries-sliding\_window.flow,  
adams-timeseries-use\_saved\_forecaster.flow

<sup>5</sup>adams-timeseries-build\_and\_use\_forecaster.flow,      adams-timeseries-sliding\_window.flow,  
adams-timeseries-use\_saved\_forecaster.flow

- *TimeseriesDbReader* – reads timeseries data from any JDBC database (required columns: timestamp, value) using the incoming token as identifier for the timeseries.
- *TimeseriesFeatureGenerator* – extracts features from a timeseries and generates, e.g., spreadsheet data from it.
- *TimeseriesFileReader* – loads timeseries data from disk.<sup>6</sup>
- *TimeseriesFileWriter* – writes timeseries data to disk.<sup>7</sup>
- *TimeseriesFilter* – applies one of the aforementioned filters to the timeseries passing through.
- *TimeseriesInfo* – allows to generate some basic information for a timeseries container.
- *TimeseriesReportDbReader* – adds all key-value pairs obtained from an SQL query to the report of the timeseries passing through.
- *TimeseriesSplit* – splits the incoming timeseries into subsets using the specified algorithm.
- *WekaPrimeForecaster* – primes a forecaster obtained from a callable actor.<sup>8</sup>
- *WekaTrainForecaster* – trains a callable forecaster on the incoming dataset.<sup>9</sup>

The following sinks are available:

- *TimeseriesDisplay* – displays one or more timeseries.<sup>10</sup>

---

<sup>6</sup>adams-timeseries-load\_and\_display.flow, adams-timeseries-load\_from\_csv\_and\_save.flow

<sup>7</sup>adams-timeseries-load\_from\_csv\_and\_save.flow

<sup>8</sup>adams-timeseries-build\_and\_use\_forecaster.flow, adams-timeseries-sliding\_window.flow,  
adams-timeseries-use\_saved\_forecaster.flow

<sup>9</sup>adams-timeseries-build\_and\_use\_forecaster.flow, adams-timeseries-sliding\_window.flow,  
adams-timeseries-use\_saved\_forecaster.flow

<sup>10</sup>adams-timeseries-build\_and\_use\_forecaster2.flow



# Chapter 3

## Tools

### 3.1 Timeseries Explorer

The *Timeseries Explorer* allows you to load timeseries data from files and directly from a database (using JDBC). Figure 3.1 shows a timeseries loaded from a WEKA ARFF file, containing data on Australian wine sales over time.

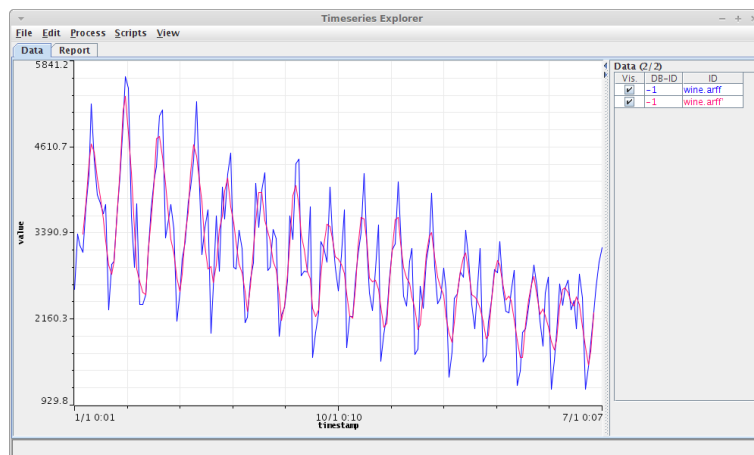
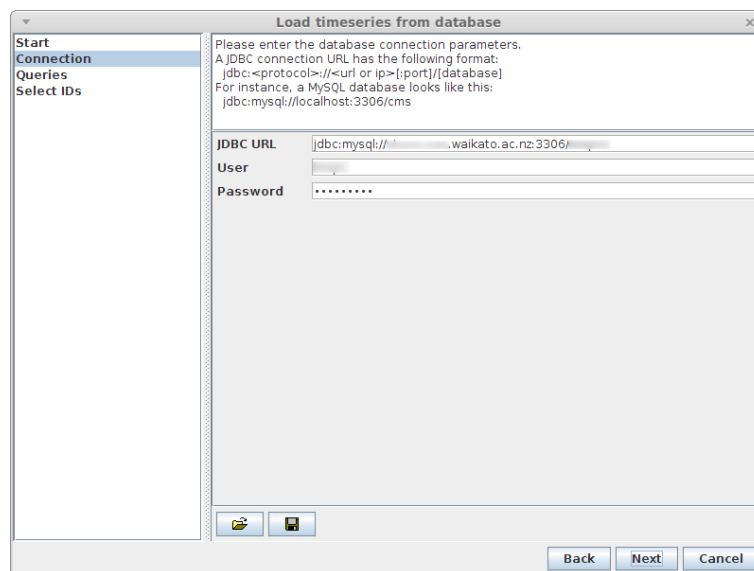
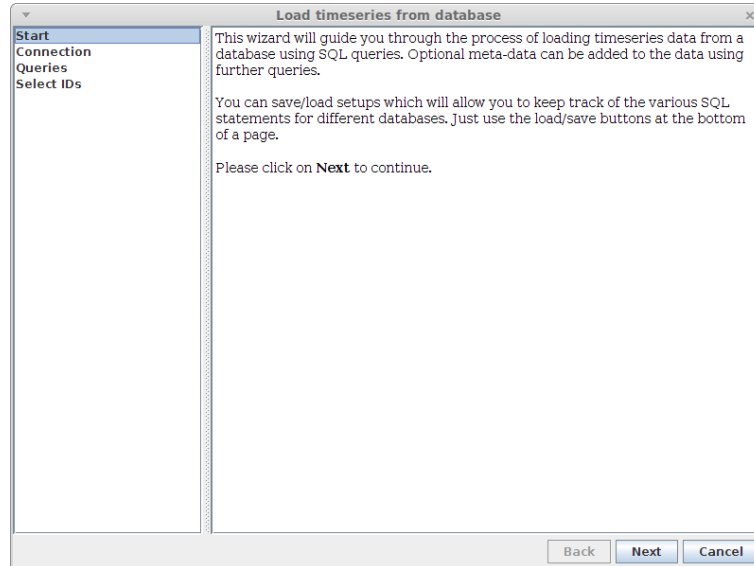


Figure 3.1: Timeseries Explorer displaying wine sales, raw and smoothed with a Savitzky-Golay filter.

Figures 3.2 to 3.6 show the wizard for loading timeseries data from a database using JDBC.



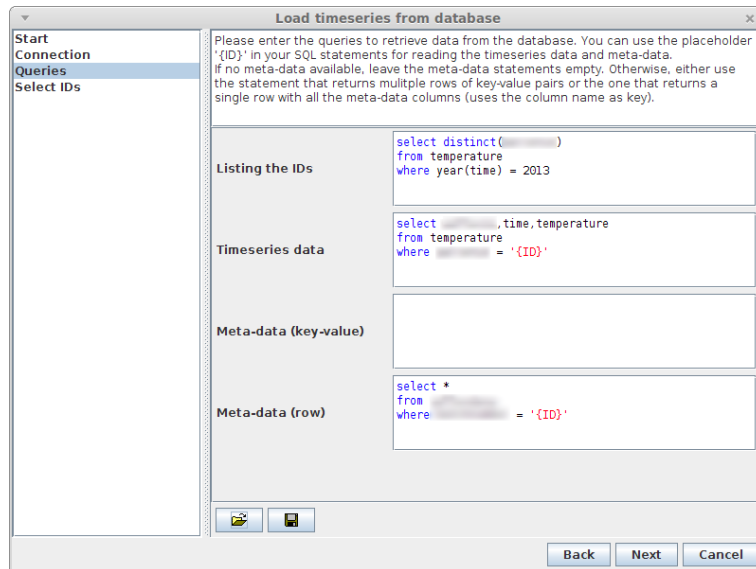


Figure 3.4: Wizard: SQL queries for obtaining timeseries data and associated meta-data.

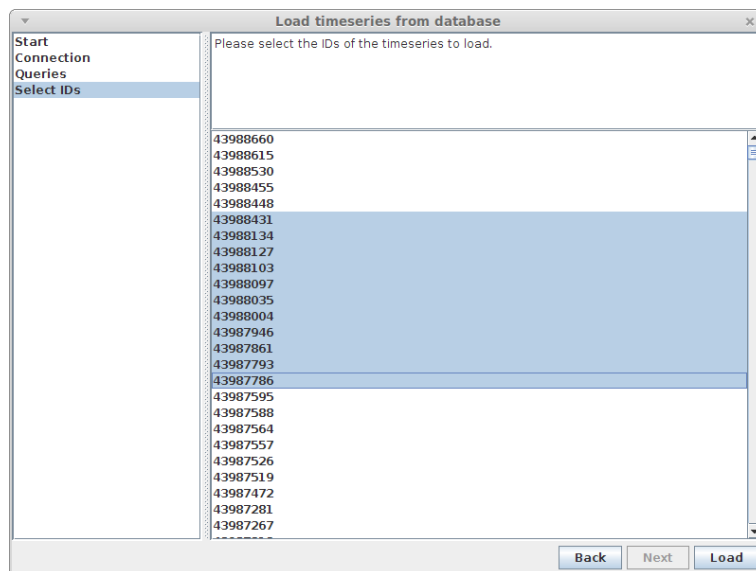


Figure 3.5: Wizard: selected IDs of timeseries to load.

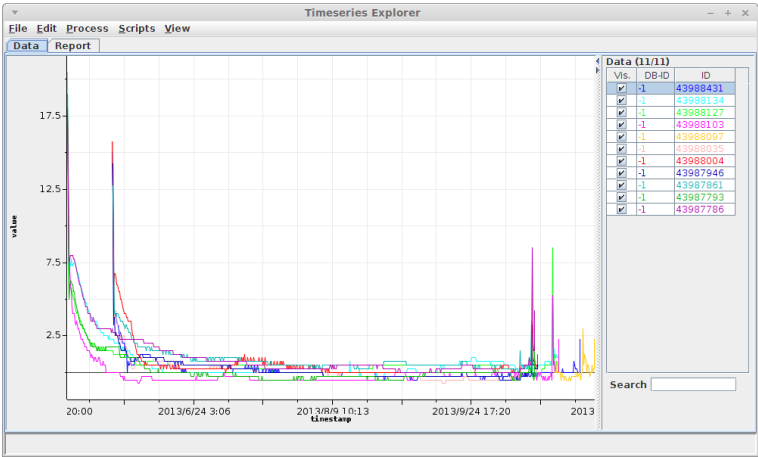


Figure 3.6: Timeseries loaded from database.



## 3.2 Weka Explorer

Using the *Timeseries Forecasting* package[3], you can perform basic timeseries analysis in the Weka Explorer as well. The *Forecast* tab in the Explorer allows you to configure a forecaster, specifying the attribute to forecast, what periodicity to use, etc. Figure 3.7 shows a screenshot of a forecast for Australian wine sales of fortified wine. An overall downward trend, despite seasonal ups and downs, is clearly visible.

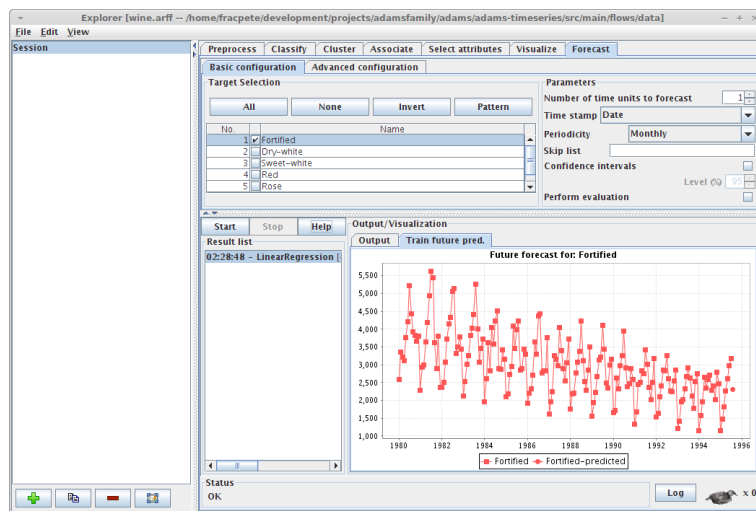


Figure 3.7: WEKA Explorer displaying forecasts for Australian wine sales.



## Chapter 4

# Setup

The following properties file contains the default format strings for periodicity types:

```
adams/data/timeseries/Periodicity.props
```



# Bibliography

- [1] *ADAMS* – Advanced Data mining and Machine learning System  
<https://adams.cms.waikato.ac.nz/>
- [2] *WikiPedia* – Time series  
[https://en.wikipedia.org/wiki/Time\\_series](https://en.wikipedia.org/wiki/Time_series)
- [3] Time Series Analysis and Forecasting with Weka  
<http://wiki.pentaho.com/display/DATAMINING/Time+Series+Analysis+and+Forecasting+with+Weka>
- [4] *WikiPedia* – Image of random data plus trend, with best-fit line and different smoothings  
<https://en.wikipedia.org/wiki/File:Random-data-plus-trend-r2.png>
- [5] *WikiPedia* – SavitzkyGolay filter  
[https://en.wikipedia.org/wiki/Savitzky%E2%80%93Golay\\_filter\\_for\\_smoothing\\_and\\_differentiation](https://en.wikipedia.org/wiki/Savitzky%E2%80%93Golay_filter_for_smoothing_and_differentiation)
- [6] *WikiPedia* – locally weighted scatterplot smoothing (LOWESS)  
<https://en.wikipedia.org/wiki/Lowess>