DNA

DYNAMIC NETWORK ANALYZER

LaTeX
Instruction Manual

René Wilmes

10.02.2015

Dresden University of Technology

Contents

Chapter 1

Introduction

The DNA – Dynamic Network Analyzer is a framework for graph-theoretic Analysis of Dynamic Networks. The built-in LaTeX-output feature allows to illustrate and compare data by generating LaTeX-documents containing data and plots.

This manual will show how DNA can be used to generate LaTeX-documents aswell as what options and parameters may be utilized in order to customize the output. Examples are going to be shown to give the user an insight on the impact of certain customizations.

Chapter 2

LaTeX in DNA

This chapter will deal with the general LaTeX-output mechanisms in DNA and how to use them. All plots that may be generated throughout the output process are created by using gnuplot.

2.1 What can be exported?

In general it is possible to output all values that are being gathered during generation. This covers runtimes and statistics aswell as data contained in metrics, namely values, distributions and nodevaluelists. This can be done in two different ways. Either export each series in one separate chapter each or export one combined chapter containing multiple series.

2.2 How to get started?

The LaTeX class *dna.latex.Latex.java* holds several public methods, which can be called to start the output process. If the plotting has been done beforehand the latex process may be started immediately. If this is not the case, there are additional methods which first plot the data and then generate a latex document. See the section 2.6 for further details.

2.3 LatexConfig

The LatexConfig class provides a way for the user to easily configure the output process. A TexConfig object can be instantiated, configured and then handed over to a latex-method. It holds several methods to customize the resulting document, for example set a timestamp window and stepsize for the data. Furthermore PlottingConfig objects can be handed over to customize the shown plots of data. See table 2.1 for more details about the TexConfig-API. More information about PlottingConfig objects may be found in the plotting manual.

Method	Description
setDateFormat(SimpleDateFormat dateFormat)	Sets a DateFormat that will be used for the timestamp representation. For more information check: Java Docs - SimpleDateFormat
setDateFormat(String pattern)	See above.
setFontSize(int fontSize)	Sets the fontsize for the document in points. Default is 12pt.
setMapping(HashMap ⟨Long,Long⟩ map)	Sets a map that maps timestamps.
setMultipleTableMode(TableMode tableMode)	Sets how the values will be ordered in multiple series tables.
setMultipleSeriesTables(boolean flag)	Set to add multiple series values into the same table.
setOutputInterval(long from, long to, long stepsize)	Sets an output interval. Will only output batches who are in the interval.
setOutputIntervalByIndex(long from, long to, long stepsize)	Sets an output interval. Will only output batches whose indices are in the interval.
setPlotFlags(PlotFlag... flags)	Sets what data will be added.
setScaling(String scaling)	Sets a scaling for the batch timestamps.
setTableFlags(TableFlag... flags)	Sets what type of data will be shown in data tables. Example: Average & Max.

Table 2.1 – TexConfig API methods.

2.4 TableFlag

TableFlag's define what types of data will be added to the data tables and in which order. They may be handed over to a TexConfig constructor at initialization or with the setTableFlags(..) method. Each TableFlag represents one column in the data table. For example: Handing over TableFlag.Average and TableFlag.Median the data tables will have two columns, the values average in the first and its median in the second column. If no flag is handed over the program will include the average data by default. The table 2.2 shows all available TableFlag's.

2.5 PlotFlag

PlotFlag objects are used to define what data will be plotted and added to the document. They can either be handed over to the TexConfig directly when the constructor is called or later with the setPlotFlags(..) method. Note that if no flag is handed over, the program will use a plotAll-flag and therefore include all available data. Table 2.3 shows all flags that can be used.

Flag	Description
all	Includes all available data.
Average	Includes the average data.
ConfLow	Includes the confidence intervals lower bound.
ConfUp	Includes the confidence intervals upper bound.
Max	Includes the maximum data.
Median	Includes the median of the data.
Min	Includes the minimum data.
Var	Includes the variance data.
VarLow	Includes the variances lower bound.
VarUp	Includes the variances upper bound.

Table 2.2 – List of TableFlag's.

Flag	Description
plotDistributions	Includes distributions.
plotMetricValues	Includes metric values.
plotNodeValueLists	Includes nodevaluelists.
plotRuntimes	Includes runtimes.
plotStatistics	Includes statistics.
plotMetricEntirely	Includes all metrics entirely. Has the same effect as the combination of plotDistribution, plotMetricValues and plotNodeValueLists.
plotSingleScalarValues	Includes all single scalar values. Has the same effect as the combination of plotCustomValues, plotMetricValues, plotRuntimes and plotStatistics.
plotMultiScalarValues	Includes all multi scalar values. Has the same effect as the combination of plotDistributions and plotNodeValueLists.
plotAll	Enables all flags.
plotCustomValues	Is used for plotting purposes only, has no effect.

Table 2.3 – List of PlotFlag's.

2.6 Latex methods

The following is a list of supported latex methods in the current build and a brief explanation.

Creates the document at the destination directory.
- writeTex(SeriesData seriesData, String dstDir, String filename, TexConfig config, PlottingConfig pconfig) (Note: It is necessary to set the plot directory in the TexConfig)
- writeTex(SeriesData[] seriesData, String dstDir, String filename, String[] plotDirs, TexConfig config, PlottingConfig pconfig)

Creates the document at the destination directory but only includes data within the timestamp window.
- writeTexFromTo(SeriesData seriesData, String dstDir, String filename, String plotDir, long from, long to, long stepsize, PlottingConfig pconfig)

Plots the series and then creates the document at the destination directory.
- writeTexAndPlot(SeriesData seriesData, String dstDir, String filename)
- writeTexAndPlot(SeriesData seriesData, String dstDir, String filename, TexConfig config)
- writeTexAndPlot(SeriesData seriesData, String dstDir, String filename, TexConfig config, PlottingConfig pconfig)
- writeTexAndPlot(SeriesData[] seriesData, String dstDir, String filename, TexConfig config, PlottingConfig pconfig)

Plots data in the given timestamp window and then creates the document at the destination directory but only includes data within the window.
- writeTexAndPlotFromTo(SeriesData seriesData, String dstDir, String filename, long from, long to, long stepsize)

Creates a combined LaTeX-document from multiple series.
- writeTexCombined(SeriesData[] seriesData, String dstDir, String filename, String plotDir, TexConfig config, PlottingConfig pconfig)

Creates combined plots and a combined document.
- writeTexAndPlotCombined(SeriesData[] seriesData, String dstDir, String filename, TexConfig config, PlottingConfig pconfig)

2.7 Example

The following code snippet shows an example on how to use a TexConfig and PlottingConfig object in order to configure the latex process.

```
public static void main(String[] args) throws AggregationException,
    IOException, MetricNotApplicableException,
    InterruptedException {
  String seriesDir = dir + "series/";

  // initialization
  GraphGenerator gg1 = new RandomGraph(GDS.undirected, 100, 300);
  BatchGenerator bg1 = new RandomBatch(0, 0, 10, 15);
  Metric[] m1 = new Metric[] { new DegreeDistributionR(),
      new UndirectedClusteringCoefficientR() };

  // create series and generate it
  Series s = new Series(gg1, bg1, m1, seriesDir, "series");
  SeriesData sd = s.generate(1, 100);

  // create configs, only plot metric
  TexConfig config = new TexConfig(dir, sd.getDir(), dir + "plots/",
      new PlotFlag[] {PlotFlag.plotMetricEntirely},
      TableFlag.VarLow, TableFlag.Var, TableFlag.VarUp);
  PlottingConfig pconfig = new PlottingConfig(PlotFlag.plotMetricEntirely);

  // plot and tex
  Latex.writeTexAndPlot(sd, dir, "preamble.tex", config, pconfig);
}
```

Figure 2.1 – LaTeX example.