Enhancing Spotfire with the Power of R

Charles Roosen (Mango Solutions)

Difei Luo (TIBCO Software)

Overview

- **What is Spotfire?**
  - The Spotfire Platform
  - The Spotfire SDK
- **R Extensions for Spotfire**
  - Motivation
  - Extension Types
  - Implementation Details

The Spotfire Platform

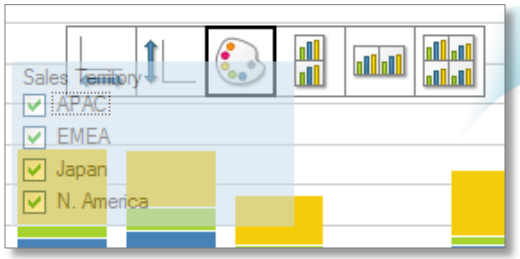
TIBCO Spotfire Enterprise Analytics overview

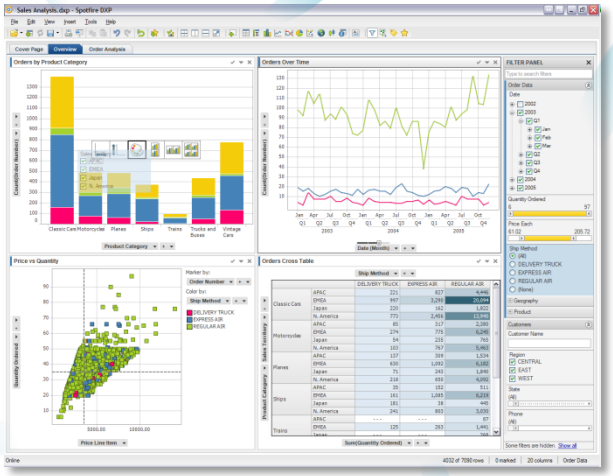
TIBCO Spotfire

Professional – In-memory interactive analytics

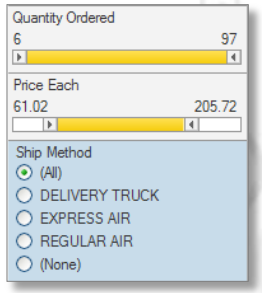
Explore data with multiple, linked visualizations
From simple Bar charts to powerful statistics at the tip of your fingers

Intuitive, user-driven interface
Completely interactive user experience

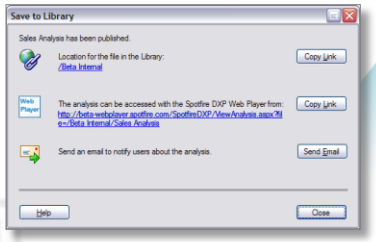




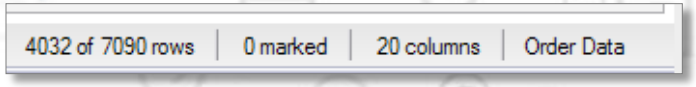
Auto-generated dynamic filters
Real-time, in-memory filtering on any field



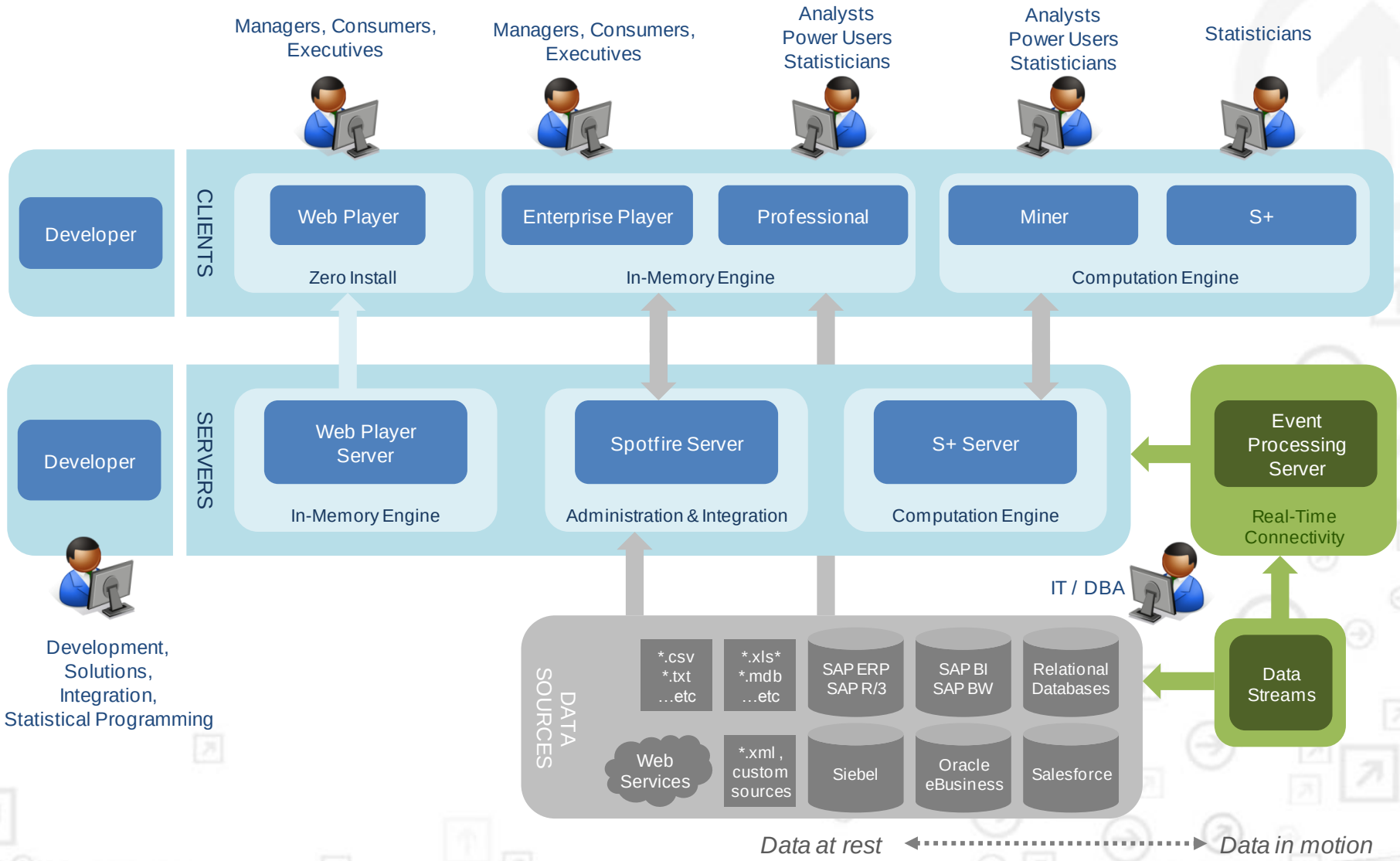
Instantly share any analysis
No separate publishing step



Load data from multiple sources
ODBC/OleDB access, TS As *Information Links*, Excel Spreadsheets, flat data files, …



TS Enterprise Analytics – Platform Overview



Demo: Spotfire

The Spotfire SDK

What is the Spotfire SDK?

- **The TIBCO Spotfire platform has a software development kit (SDK) for configuring, automating, and extending the platform.**
  - The SDK is used to create custom TIBCO Spotfire *Extensions*.
  - Extensions add features that from the end users perspective are virtually indistinguishable from the built-in features.
- **Extensions are implemented in C# using Visual Studio**
  - Well documented API's
  - Well designed architecture
  - Good examples on the Spotfire Technology Network

Extension Types

- **TIBCO Spotfire Extension types include:**
  - **Data Reader:** Import data from custom file, database, and web services data sources
  - **Data Writer:** Export data to custom file, database, and web services data sources
  - **Calculation:** Add custom aggregation, expression function, and statistical methods that automatically re-execute on data update events
  - **Dialog Tool:** Prompt users for input into custom application and document-level automation jobs
  - **Control Panel:** Collect user input from and display feedback to an always available custom control panel
  - **Visualization Panel:** Display custom charts, plots and other forms of presentation based on the current set of filtered and marked data
  - **Automation Interface:** Send and receive properties, events, and commands to and from an external application

R Extensions for Spotfire

Motivation

- **R has a wealth of statistical techniques of value to Spotfire users**
  - R core features
  - Bioconductor
  - PK modeling
- **Focus is on using R for statistical computation**
  - R does the computation
  - Spotfire does the user interface, graphics, reporting, and automation
- **Why not just use S+?**
  - R has functionality not available for S+, e.g. Bioconductor

Extensions Implemented

- **Open R Data File**
  - Load the first object in an “RData” file as a data.frame
- **Open R Script File**
  - Execute an R script file creating a data.frame
- **Open From R Expression**
  - Execute a one-liner expression creating a data.frame
- **R Column Calculation**
  - Execute an R expression that creates a column
- **R Custom Expression**
  - Execute an R expression computing a summary such as “mean()” or “mad()”
- **Create and Manage General R Calculations**
  - Execute a block of R code creating a table

Demo: R Extensions

Implementation Details

- **General approach**
  - Spotfire writes data to a tab-delimited file
  - R reads the data, does some processing, and writes results to a tab-delimited file
  - Spotfire reads the tab-delimited file
- **The C# code**
  - On the C# side, most of the code is somewhat generic code for invoking some general process and communicating to it via text files
  - The only R-specific code is that related to what “Process” is invoked
- **The R code**
  - A general R template script handles the data exchange and errors
  - Each type of extension has an R script, e.g. “Create Column”
  - Varying values such as file names or R expressions are inserted into the script as “spotfire.parameters”

C# Process Code

```
File.WriteAllText(generatedScriptFile, masterScript);
string args = string.Format("CMD BATCH {0} {1} {2}", batchOptions,
   scriptFileName, stdoutFileName);

string rExecPath = RCalculationUtilities.GetRExecutableFilePath();
ProcessStartInfo info = new ProcessStartInfo(rExecPath, args);
info.WorkingDirectory = sessionDir;
info.WindowStyle = ProcessWindowStyle.Hidden;
try {
       using (Process rProc = Process.Start(info)) {
           rProc.WaitForExit();
           if (rProc.ExitCode > 1) {
               string errorMsgFileName = Path.Combine(sessionDir,
                 "errormsg.txt");
               if (File.Exists(errorMsgFileName)) {
                 errorMsg = File.ReadAllText(errorMsgFileName);
               }
           }
       }
   }
```

R Master Template

```
# If an error occurs, quit with a non-zero status code
options(error = expression({cat(geterrmessage(),
  file="errormsg.txt");q(status=100)}))

# Parameters for passing settings other than data to R.
spotfire.parameters <- {1}

# Load data
spotfire.inputFilename <- "{2}"
spotfire.data <- read.delim(spotfire.inputFilename, check.names = FALSE)

# BEGIN User defined script
{0}
# END User defined script

# Save results
spotfire.outputFilename <- "{3}"
write.table(spotfire.results, file = spotfire.outputFilename, sep = "\t",
  na = "", row.names = FALSE)
```

R Extension Scripts

```
## R objects used:
##
##   spotfire.data: data.frame passed from Spotfire to R
##   spotfire.parameters: named character vector of parameters
##   spotfire.results: data.frame passed from R to Spotfire
##
## Note: Error and special-case handling code left out for space reasons

# Example 1: Load the data from an Rdata file
spotfire.results <- get(load(file = spotfire.parameters["FilePath"])[1])

# Example 2: Execute an expression creating a data.frame
expr <- spotfire.parameters["DataFrameCreationExpression"]
spotfire.results <- eval(parse(text=expr))

# Example 3: Execute an expression creating a column
expr <- spotfire.parameters["CreateColumnExpression"]
spotfire.results <- with(spotfire.data, eval(parse(text=expr)))
```

Summary

- Spotfire is a useful tool for interactive data visualization
- R provides rich computational capabilities for Spotfire users
- The Spotfire SDK makes it easy to integrate the two
- Simple communication via text files and a child process works pretty well