Final Project Report



*Approvals:*

|  |  |  |  |
| --- | --- | --- | --- |
| Team Leader, Interface Specialist | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Alex Laird | Date: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Programming Specialist | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Ryan Morehart | Date: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Statistics and R Programmer | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Andrew Sterling | Date | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

*Document History:*

04-29-2011 – Initial Draft

Contents

[1 Abstract 4](#_Toc292055959)

[2 Project Definition Statement 5](#_Toc292055960)

[3 Introduction and Background 6](#_Toc292055961)

[3.1 Purpose 6](#_Toc292055962)

[3.2 Customer and Application Area 6](#_Toc292055963)

[3.3 Related Tools 6](#_Toc292055964)

[4 Requirements 7](#_Toc292055965)

[4.1 Guidance Requirements 7](#_Toc292055966)

[4.2 Interface Requirements 7](#_Toc292055967)

[4.3 Functional Requirements 7](#_Toc292055968)

[5 Constraints and Assumptions 8](#_Toc292055969)

[6 Design 9](#_Toc292055970)

[6.1 GUI 9](#_Toc292055971)

[6.2 Problem 9](#_Toc292055972)

[6.3 Operations 9](#_Toc292055973)

[6.4 R Processor 10](#_Toc292055974)

[6.5 Exporter 10](#_Toc292055975)

[6.6 Operation Editor 11](#_Toc292055976)

[7 Conclusion 12](#_Toc292055977)

[8 Individual Achievement 13](#_Toc292055978)

[8.1 Alex Laird 13](#_Toc292055979)

[8.2 Ryan Morehart 13](#_Toc292055980)

[8.3 Andrew Sterling 14](#_Toc292055981)

[9 Bibliography 15](#_Toc292055982)

# 

# 1 Abstract

At its core, The maRla Project intends to eliminate the LaTeX and R learning curve from a first-semester probability and statistics course. It does so in an easy to use graphical environment that allows users to manipulate a drag-and-drop interface to perform complex statistics calculations in a simple manner.

The maRla Project strives for intuitiveness, relying on common paradigms that students have likely experienced in other programs. Throughout development, The maRla Project attempted to stay true to the principle of teaching concepts, not tool-specific functions. This is in contrast to many other statistical Integrated Development Environments (IDEs) that exhibit steep learning curves and make their use in short-term classes impractical. To that end, the user should be able to use the vast majority of maRla without having to even consult a manual.

The features that do require documentation are advanced functions that the average student does not necessarily need to employ. However, these more advanced features meet the critical goal of extensibility. The most notable of these is the ability for the user to create additional operations for The maRla Project to employ. By editing a single file and without recompiling or even restarting the maRla IDE, any user of The maRla Project can extend its capabilities.

While a student may not be interested in consulting the online documentation for how to create new operations, a professor who is planning on utilizing maRla within multiple classes they are teaching might find the time investment well worthwhile. This may include simply making new statistical operations, but no reason prevents a professor from completely revamping the operations to be usable in another course. As a result, The maRla Project could foreseeably be found in any math class or even potentially used in the work environment or a graduate project or study.

Ultimately, The maRla Project represents a simple, robust, and flexible statistical package. On their first encounter with the maRla IDE, a student will find an easy to use environment that simply lets them get their homework done. A professor with more experience will find that maRla can be employed in more classes that one would anticipate.

# 2 Project Definition Statement

The maRla Project is a cross-platform statistics IDE for students to help them work through and solve complicated statistical calculations. It strives to aid the student in the entire homework cycle, from defining the problem to the final submittable document.

# 3 Introduction and Background

## 3.1 Purpose

The maRla Project intends to eliminate the LaTeX and R learning curve from a first-semester probability and statistics course. It does so in an easy to use graphical environment that allows users to manipulate a drag-and-drop interface to perform complex statistics calculations in a simple manner.

## 3.2 Customer and Application Area

The maRla Project was developed with the classroom in mind. The maRla IDE—the core application—strives to aid in the entire homework cycle for a student. This cycle is comprised of four steps:

1. Establishing the problem
2. Performing work
3. Analyzing the results
4. Submitting to professor

The maRla Project accomplishes each step in an intuitive manner. Given that most students only take one statistics course in their college careers, maRla makes itself usable without the consultation of hefty manuals. Unlike when a student must learn how to use a complex full programming language such as R, The maRla Project can be used right away to perform virtually all of the computation a student will encounter.

However, maRla does not take the burden of learning the purpose of each statistical operation off of the student. The maRla Project is not intended to necessarily make the actual statistics part of the course easier, as the student must still learn the purpose behind the operations that maRla allows them to perform maRla places the student’s focus back on the concepts that are being taught, not on learning how the tool itself functions.

While the focus of maRla does rest on the student, the professor also plays an important role in maRla’s operation. Through a flexible system of XML files, professors can extend maRla’s operation set to match their classroom needs more specifically. The creation of these operations is aided by the maRla Operation Editor, a tool which validates the XML in real time and allows the user to see the results of their operation without leaving the tool.

## 3.3 Related Tools

There are a few other projects that attempt this concept of easing some of the burden of learning R (or other statistical packages). Most notable are the Java GUI for R (JGR) and S+, both of which run around R (technically, S+ uses S, but it’s very closely related). These tend to be simple IDEs that aid in determining function name and the parameters to give to those functions. In the cases where they do actually take the burden of learning the underlying language off of the user, interactions are driven through complex dialogs that overwhelm the user with options.

# 4 Requirements

## 4.1 Guidance Requirements

* The application shall be programmed in Java.
* The application shall use R and LaTeX as both a middle layer (for exports to PDF) and an end result (for exports to R).
* The application shall use Sweave to perform R interactions within exported LaTeX code.
* The application shall remove any code-based learning curve for first-semester statistics students.
* The application shall not implement any statistical calculations, but shall make the appropriate calls to R, which shall do them for the application.

## 4.2 Interface Requirements

* The application shall have a simplistic, drag-and-drop interface.
* Components in the workspace shall interact with each other and will be movable and reconnectable.
* Components shall snap together when allowed.
* The application shall allow the user to enter data in a clean, graphical manner.
* Users shall be able to construct an entire problem via the New Problem Wizard, which will start their workspace for them.
* Users shall be able to right-click on individual components to edit the individual component or group it is connected to.
* Users shall be able to tie a group of components (and their solution) to a given problem or subproblem (specified through the New Problem Wizard).
* Users shall be able to load and save workspaces.

## 4.3 Functional Requirements

* The application shall be able to produce PDF output, preferably through LaTeX.
* The application shall be able to receive data response from R.
* The user shall be able to specify assumptions for the statistical data.
* The application shall notify the user if assumptions are violated, but shall allow the user to continue.
* The user shall be able to export R commands for a specific block.
* The user shall be able to export LaTeX code for a specific block.
* The application shall allow the user to perform, at a minimum, T-Test, ANOVA, and basic statistical commands.

# 5 Constraints and Assumptions

* Installation and configuration posed the largest constraints on maRla
  + Limited-user accounts on the school computers – everything within maRla must operate under a single-user install. The installer should not require students to get admin privileges in any way
  + Support for XP, Vista, and 7 can be complicated on occasion, particularly when dealing with 32-bit verses 64-bit versions.
* Testing
  + Extensive JUnit testing was implemented. However, the operations themselves escape this testing currently, leading to possible problems with them during releases
  + Additional testing was performed through Professor Schumacher releasing beta versions of The maRla Project to students. Each student submitted notes on issues found. This led to increased robustness, the clarification of some usages, and additional features.

# 6 Design

Five major components comprise The maRla Project’s primary IDE: GUI, Problem, Operations, R processor, and the exporter. Each will be detailed below. In addition, The maRla Project includes an operation editor which will be briefly discussed in its own section.

## 6.1 GUI

The interface completed all goals we originally intended for it, along with several we had not originally intended. The core interface of the workspace is drag-and-drop based, as we originally intended, and trees within the workspace can branch out infinitely, creating a further branch at any point.

The palette visually is dynamically generated. It can be expanded and collapsed by categories as it becomes too large.

The New Problem Wizard successfully holds the hand of the student as they walk through the creation of the problem. Specifically, it allows ease when importing data values from a CSV or from an R library.

The interface provides a Settings dialog, which allows the user to customize back-end paths, libraries, and interactions from the front-end. It also allows for user-interface customization.

From the interface, the user has the ability to customize, edit, load, save, and close their problem. They can import data, export data, export the problem as a LaTeX or PDF file, and print the workspace.

## 6.2 Problem

The Problem class and package represent the head of the internal structure of maRla. The GUI keeps a reference to the current Problem and most interaction with the internal state of maRla occurs through it. Examples of what the Problem class keeps track of include the problem description and conclusion, sub problems associated with it, and any datasets.

Sub problems, part of the problem package, keep track of their own descriptions and conclusions and the steps (operations and/or datasets) that comprise their solution. Datasets are considered part of the problem package as well and are composed of columns of data that may be in either numeric or string mode.

There were pretty vast changes to the structure of these components as maRla evolved, particularly with regards to sub problems. However, it seems pretty good now; I do not envision any serious restructuring or additions.

## 6.3 Operations

Operations are the heart of maRla, as all logic for computations occurs here. Operations may be assigned to a parent and then calculate results based on that parent, caching computations until the parent changes. In addition, operations define their own names and categories for display to the user and may specify questions to ask the user.

An abstract Operation class defines the vast majority of the functionality that operations exhibit. This includes the caching/dirtying functionality, the handling of user prompts, pulling data from the parent, and recording the actual R commands used in computation.

From there concrete classes may be defined that implement the abstract methods defined by Operation. A derivative operation must define their description, how to perform computation, whether they have a plot and—if they do—where the image file is located, and finally how to copy themselves.

The only example of this within maRla currently is the OperationXML class, which is capable of performing all calculations based on an XML file read from disk. The XML language is fully documented on The maRla Project’s wiki. The 70 operations currently defined for maRla exist entirely in the XML specification, clearly demonstrating the power and flexibility available through this class.

The operations and XML operations fell into place largely as expected, although obviously they both evolved along the way. At this time I do not foresee major overhauls to the structure of these systems. The XML could expand to include more options, although there are no obvious deficiencies I can think of at this time.

## 6.4 R Processor

The R processor is a relatively simple class. It contains a processor running an R interpreter which it is capable of interacting with, both to send commands and to receive results. The class ensures that the R process is setup and loaded correctly and ensures that if it crashes those that depend on its services are informed with a reasonable exception.

At the processor’s heart is a single execution function. This function takes a single R command and hands it off to R itself, along with a marker print call. It then watches for R to return results, waiting until it sees the result of the marker print come back. The result from R is saved to a string and returned to the caller. If at any time an error or warning is thrown by R, the execution function wraps the string R gives in an exception and throws that.

Around this execution function are a wide variety of helper functions that are designed to handle typical returns. Parsing helper functions exist for strings, numbers, booleans, and vectors. Additional functions allow uniquely-named variables to be set in R and to handle generating images and getting the path to the resulting file.

This design is wildly different than our original plan, which involved using an external library. Configuration and poor documentation made that fall through though. Using this method we have created an extremely robust system that could be easily incorporated into any other Java program that needs to access R (dependencies on maRla itself are minimal and only exist to warn maRla that R isn’t configured, it could be easily changed).

There were some concerns about the speed of the new design, but real-life use did not bear out those fears. While no direct performance tests have bave been performed, there’s certainly no visible difference between commands run through maRla and directly through R.

## 6.5 Exporter

The final, critical component of the maRla IDE is the exporter. Without this module maRla would not accomplish the final step of the homework cycle, turning results in the professor. The exporter is capable of producing both plain LaTeX files and PDFs, depending on the needs of the student.

The exporter relies on an XML template for most of its work, dropping literal LaTeX code into the export file and doing substitutions when it encounters “replacement” XML tags. Limited support for loops and conditionals exist, allowing the template to specify how to handle sub problems. For areas that the template marks as the “solution” location, the exporter interleaves solution steps with the associated remarks and analysis. Solution steps are given in a summarized table format that shows the parameters used and the resulting data and/or plot.

If a PDF file is requested, the normal export process is used to create a temporary LaTeX file. That file is then run through Sweave, producing a file than can be passed off to pdflatex. Through a series of four runs (ensures long tables get generated correctly), a PDF file is generated.

There were various plans for the exporter, but nothing was firmly put down because the exporter was built at a much later stage in maRla’s development. The most needed improvement is to expand the templating system to specify how operations will be displayed. The table format is currently hardcoded (and not pretty at that). Basically the entire exporter could use some work to make the results better.

## 6.6 Operation Editor

The operation editor is designed to ease the creation of XML operations. It is a stand-alone program from the maRla IDE. The editor features real-time notification of errors, syntax highlighting, the results produced given an input dataset, and the actual R commands that are being used in the background. While this tool currently still requires consultation of the documentation on XML operation, it can greatly ease in discovering errors.

There were absolutely zero plans for the editor before it was created, so we did not deviate at all. The best improvement that could be brought to it is a most walk-through style system. Maybe a section to add queries, add/remove categories, and only a small editing area for the computation itself (this is in contrast to the current, largely manual editing).

# 7 Conclusion

The maRla Project succeed in meeting all of its original goals. Additionally, several originally unintended features became core functionality of the software package. It removes the need for students to learn R and LaTeX themselves, greatly reducing the frustration associated with classes that employ them. Without that additional pain, students are able to focus on learning what actually matters: the concepts and principles of statistics.

The maRla Project’s advisor and customer, Professor Schumacher, is quite happy with the results. At the Release Candidate 2 stage, he reported he was unable to break anything, even while employing it in class. In addition, we personally have found no critical, breaking issues within maRla for several weeks. Given that, maRla appears to have matured into a fully-functional product, ready for deployment into the classroom.

All code within The maRla Project is released under the GNU GPL v3 and is freely available to the public on Google Code. The team is comfortable with the quality of the code base and believes that it is a valuable, usable contribution to the open source community.

# 8 Individual Achievement

## 8.1 Alex Laird

For The maRla Project, I was responsible for designing and implementing the Graphical User Interface (GUI). This development entailed two software packages within The maRla Project: the maRla IDE and the maRla Operation Editor. These are two distinct applications contained within the project.

For each user interface, we strove to balance strength and flexibility with intuitive user interactions. We regularly released the software to statistics classes to test how comfortable non-computer students were with its interface, and I was primarily responsible for implementing the change requests the students had.

In source detail, I was responsible for all classes contained within:

* marla.ide.gui.\*
* marla.ide.gui.colorpicker.\*
* marla.opedit.gui.\*

This source work includes all form source files that connect directly to the user interface class files as well as extended GUI functional classes, including extensible tables, models, color choosers, toolbars, extended panels (specifically for the workspace), and an extended multi-line input dialog for capturing user remark information.

In general, my responsibility was developing how the user indirectly interacted with the back-end of the software. Obviously, the project would not have been successful if it did not have an interface that the user could interact with, and the functionality of the back-end would have been largely worthless without an ability to interact with it.

## 8.2 Ryan Morehart

Within The maRla Project, I was responsible for the entire backend. This included everything from the top-level Problem class to the R processor itself. Additionally, I handled all jUnit testing, the documentation and writing of the actual operations and export template, and the exception-reporting site.

Breaking it down into details, I was responsible for the following classes and/or packages:

* marla.ide.latex.\*
* marla.ide.operation.\*
* marla.ide.problem.\*
* marla.ide.r.\*
* marla.ide.resource.Configuration
* marla.ide.resource.ConfigurationException
* marla.ide.resource.DebugThread
* marla.ide.resource.UndoRedo
* marla.ide.resource.Updater
* marla.opedit.resource.operation.\*
* Everything within the maRla IDE test directory

Generally speaking, the usefulness of this is fairly obvious, as nothing in The maRla Project would do anything without my code. No computation, no saving and loading, no operations, no exporting, no ability to undo that mistake you just made, and no ability for maRla to be configured.

## 8.3 Andrew Sterling

For The maRla Project, I was responsible for many of the behind the scenes details. Initially, I did much of the research and development relating to the R processor, specifically finding external libraries that work with R. The solution I arrived at was the use of the Java/R Interface. I was able to get some of it working until we realized that it was not feasible to do with a library not developed by us. The functionality did not meet our needs.

I was also responsible for developing the Windows installer. I built an installer that was able to install The maRla Project programs, TeXMaker, and R, without having to download and install them separately. They are able to get the installers both from the local Cedarville network for quicker access and from the website of the software itself. They are also able to get and install related software that helps the user with TeX document publishing.

I was also responsible for user-level documentation. Initially, had the documentation in a separated HTML format, but we decided to use the Google Code project wiki to host all of our help documentation. I created an html formatted page as well as a PDF format.

# 9 Bibliography

Laird, Alex D., Andrew T. Sterling, and Ryan A. Morehart. SDD - The maRla Project. Cedarville:

Cedarville, 2010. Print.

Laird, Alex D., Andrew T. Sterling, and Ryan A. Morehart. SRD - The maRla Project. Cedarville:

Cedarville, 2010. Print.

Laird, Alex D., Andrew T. Sterling, and Ryan A. Morehart. SPMP - The maRla Project. Cedarville:

Cedarville, 2010. Print

Laird, Alex, Ryan Morehart, and Andrew Sterling. "MaRla Final Presentation." Software Engineering II. ENS, Cedarville. 27 Apr. 2011. Speech.