Guerrilla Automation with R

*R automation despite resource constraints*

**Introduction**

Guerrilla warfare is a type of asymmetric warfare wherein a smaller, less powerful military uses unconventional tactics to fight a stronger, more traditional military. The definition via Google of the adjective form of guerilla is “Referring to actions or activities performed in an impromptu way, often without authorization.” When a problem calls for a powerful technological solution, but no such solution is currently in-place, actuaries may require a low-cost, light-weight approach to meeting the needs of their stakeholders.

That’s where R comes in. R is a free and open source programming language with a powerful and continually-expanding set of libraries and packages. It is highly versatile and compatible with a wide variety of other systems. I am going to walk through a hypothetical use-case for R that highlights R’s advantages. Example code will be interspersed throughout the article. The full code, complete with example excel files can be accessed via Github at the following url:

<https://github.com/TimothyQuast/Guerrilla_Automation>

**The Use-Case**

Let’s suppose that our stakeholder has several ledger account balances they want to support using the actuarial workpapers that feed them. They want to break each balance into pieces, with each piece corresponding to one actuarial workpaper which contributes to that balance. The actuarial workpapers are stored in Excel in a regular format in different places throughout the stakeholder’s network drives, with a variety of teams contributing to the same account balances. Moreover, the workpapers represent an aggregated version of the actuarial results, so they aren’t overly granular. But they are granular enough to make a manual solution infeasible. How do we get the data we need to support the account balances?

The proper, traditional method is a big fancy subledger containing the supporting balances along with metadata that traces each balance. But let’s say the stakeholder doesn’t have a big fancy subledger yet. Further, let’s say that the stakeholder wants the balances supported *soon* – sooner than a big fancy subledger can be developed. We need a temporary solution to solve the problem quickly. In such a conundrum, one might try automating the task with R!

R is suitable for the task for several reasons:

* **It’s free!** It can be used for such a task without requiring investment dollars or licensing.
* **It’s open source with a broad user base.** Developers are continually producing marvelous new packages that can solve tough problems.
* **It’s highly versatile.** The variety of packages available and the non-proprietary nature of the system make it uniquely capable of interacting with other systems, such as Excel.
* **It’s suitable for rapidly prototype.** R is elegant and fairly high-level, allowing the user to do a lot with just a little code.

R does have some disadvantages too:

* **It’s less efficient.** R doesn’t do as well as other languages in terms of processing efficiency. That’s why it’s important that the workpapers are already aggregated. If they were extremely granular, then R might struggle.
* **There’s a learning curve.** Like all programming languages, R must be learned. It might be difficult for new personnel to learn the language, making the process less portable.

The drawbacks are not overwhelming and it won’t cost us anything but time to try, so let’s figure out how to solve our problem with R. I’ve constructed an example problem (which you can find in the GitHub repository) using 4 Excel files. **Ledger Balances.xlsx** contains the “ledger balances” we are trying to support. Three workpaper files contain the “workpapers” that support the “ledger balances”. Pretend that the workpaper files are located in disparate places through the stakeholder’s network drives!

|  |  |
| --- | --- |
| **Excel Files/Ledger Balances.xlsx** | **Excel Files/Workpaper 1.xlsx** |
| **Excel Files/Workpaper 2.xlsx** | **Excel Files/Workpaper 3.xlsx** |

You can manually verify that the account number totals in the 3 workpapers sum to the ledger balance totals, but then what’s the point of automating it? Notice that the workpapers keep the data in a regular format, but that **Workpaper 3.xlsx** has the data in different cells and it has an unusual sheet name! These differences are intentional, and we will handle them in our solution.

**The Solution**

* Describe the use-case and highlight key features.
  + Need to gather a moderate amount of data stored in a regular format
  + Lack of centralized data repository or need to refer to non-centralized data
  + Data is stored in excel or another annoying file-type/structure
  + Lack of impetus and/or resources to fix the previous two problems
* Why is R suitable for the task?
  + It’s free!
  + The open source nature and broad user base makes for a prolific developer community which continues to produce marvelous and effective packages.
  + Extremely compatible with a wide variety of data-types and systems, especially due to the large number of packages available
  + Well-suited for rapid prototyping
  + Some disadvantages
    - R can struggle with processing speed, so scalability is an issue.
    - R can be difficult to learn, making portability of the process (to other personnel) challenging.

The Solution

* Organize inputs by putting information need to gather data (e.g. file paths) into a spreadsheet or data table.
* Validate the inputs (e.g. are the file paths correct? Which ones are wrong?).
* Pull the data using the validated inputs.
* Organize/Summarize the data.
* Additional Commentary
  + If the data is not in a regular format, can use inputs to correct the process
  + Can summarize at different levels of aggregation, but should ensure ability to trace data back to inputs for ease of validation and auditability

Appendix

* Example Code

|  |  |  |  |
| --- | --- | --- | --- |
| **Ledger Balances.xlsx** | **Workpaper 1** | **Workpaper 2** | **Workpaper 3** |
|  |  |  |  |