Combining "R engines" with 3rd party software

Adi Sarid

Department of Industrial Engineering, Faculty of Engineering, Tel-Aviv university
Partner and Head of Operations Research at Sarid Research Institute
adisarid@gmail.com, *adi@sarid-ins.co.il*

July 2017

Background

- R has packages to almost any statistical problem
- Recent developments have made R very efficient (e.g., Rcpp, dplyr, tidyr)
- Makes it suited to solve a lot of industry problems in production environments, but:
  - You can't compile R code into an executable (not "directly")
- I'm going to present some ideas and personal experience for integrating R built engines with 3rd party software

This is not a technical discussion but a conceptual discussion

A use case: the retailer

A retailer manages his inventory in some ERP system

- **Recurring:** each week the retailer has to decide on order qunatities months ahead
  - Requires **forecasting** (statistical know-how)
  - **Complex:** thousands of items + effects (promotions, out of stock, holidays)

A use case: the retailer

A retailer manages his inventory in some ERP system

- **Recurring:** each week the retailer has to decide on order qunatities months ahead
  - Requires **forecasting** (statistical know-how)
  - **Complex:** thousands of items + effects (promotions, out of stock, holidays)
- **No out-of-the-box modules**: solutions are costly and poorly fitted
- **R can help:** state of the art forecasting tools are already implemented in R (e.g., package `forecast` for ets, and package `hts` for hierarchical time series)

A use case: the retailer

A retailer manages his inventory in some ERP system

- **Recurring:** each week the retailer has to decide on order qunatities months ahead
  - Requires **forecasting** (statistical know-how)
  - **Complex:** thousands of items + effects (promotions, out of stock, holidays)
- **No out-of-the-box modules**: solutions are costly and poorly fitted
- **R can help:** state of the art forecasting tools are already implemented in R (e.g., package `forecast` for ets, and package `hts` for hierarchical time series)
- **Minimum hassle:** How can we harness R to solve the retailer's problem **seamlessly**?

I'm not going to talk about forecasting but on how to integrate a statistical engine with a 3rd party system (e.g., the ERP)

Comparison methodology

This is not an extensive review ("a taste" of four methods)

- "Quick and dirty"
- Script automation
- `Shiny` apps
- `RInside`

Comparison methodology

This is not an extensive review ("a taste" of four methods)

- "Quick and dirty"
- Script automation
- `Shiny` apps
- `RInside`

Things to consider:

- How **easy it is to implement** the method? (to the R developer?, to the customer?)
- Data **security**
- Safety of the **source code**? (intellectual property)
- Handling over **long periods of time** (recurring use)

Comparison methodology

This is not an extensive review ("a taste" of four methods)

- "Quick and dirty"
- Script automation
- `Shiny` apps
- `RInside`

Things to consider:

- How **easy it is to implement** the method? (to the R developer?, to the customer?)
- Data **security**
- Safety of the **source code**? (intellectual property)
- Handling over **long periods of time** (recurring use)
- Additional considerations: Pilot/POC versus upscale, Cloud versus on-premise

Quick and dirty

Import/export csv files

The simplest most intuitive solution:

1. Get data files from the customer (e.g., csv)
2. (Develop your engine)
3. Run and export results to customer
4. Customer needs to import back

Works fine for a pilot/proof of concept, but some limitations

- Easy and quick to implement
- Over time requires a lot of overhead – not practical over time
- Sending CSVs back and forth might be unsecure
- Source code is safe (but customer can't run code himself)

Scheduling R scripts

A **simple** extension to automate the previous approach, based on a scheduler

1. Have the customer provide a data dump or a read/write credentials to the DB. R should:

Scheduling R scripts

A **simple** extension to automate the previous approach, based on a scheduler

1. Have the customer provide a data dump or a read/write credentials to the DB. R should:
   - 1.1 Read the data (use `dplyr` or `RODBC` for sql/sqlite/mysql, `mongolite` for MongoDB, etc.)
   - 1.2 Perform the analysis you need
   - 1.3 Write the output to a data dump or write directly to the DB

Scheduling R scripts

A **simple** extension to automate the previous approach, based on a scheduler

1. Have the customer provide a data dump or a read/write credentials to the DB. R should:
   - 1.1 Read the data (use `dplyr` or `RODBC` for sql/sqlite/mysql, `mongolite` for MongoDB, etc.)
   - 1.2 Perform the analysis you need
   - 1.3 Write the output to a data dump or write directly to the DB
2. To schedule the script use task scheduler in windows or `crontab` in linux

Scheduling R scripts

A **simple** extension to automate the previous approach, based on a scheduler

1. Have the customer provide a data dump or a read/write credentials to the DB. R should:
   - 1.1 Read the data (use `dplyr` or `RODBC` for sql/sqlite/mysql, `mongolite` for MongoDB, etc.)
   - 1.2 Perform the analysis you need
   - 1.3 Write the output to a data dump or write directly to the DB
2. To schedule the script use task scheduler in windows or `crontab` in linux
3. To execute, use `Rscript` (windows or linux) in scheduler:

```
Rscript script_file.r [arguments]
Read arguments from command line:
args = commandArgs(trailingOnly=TRUE) #within the R script
```

Scheduling R scripts – continued

This is a nice and simple extension to the "quick and dirty" method

- Requires some knowledge of the customer's DB + access
- Automated approach - not a lot of overhead over time
- Can be implemented securely over SSL or on-premise
- Code safety? can work either way
  - On customer's server (on-premise/cloud) = code is compromised
  - Your own cloud (SaaS to customer) = code is safe
- Can also be run as "script on demand"
  (instead of scheduling, customer's software connects via SSH and runs server's script on demand)

Use Shiny to run the script

(Yet another small improvement)

Building on the previous method

- Use the same principles to exchange the required data back and forth but through a shiny app
- Can be easily run on demand (by visiting the shiny app)
- Can be problematic for very long operations (no feedback while running)
- Added value – can provide an interface and visualizations to customer
- Data is secured over SSL
- Code is safe (customer only sees client side)

RInside

Embed R code in C++

Embed R within C++ using the `RInside` package
(Personally, I'm not C++ proficient – didn't go there)

- Requires knowledge of C++ and some effort on both sides (you and customer)
- Can reach seamless integration to the end user
- Can be implemented securely over SSL or on-premise
- Code is safe if you do the compiling and provide a binary file

Summary

A qualitative and subjective summary

Short summary, according to my impression of each method. Where: 🟢 = Good, 🟠 = Sub optimal, 🔴 = Bad

	Method	Easy to implement	Scalable	Data security	Code safety
1	Quick and dirty	🟢	🔴	🟠	🟢
2	Scheduling R scripts	🟠	🟢	🟢	🟢🔴
3	Shiny apps	🟠	🟠	🟢	🟢
4	RInside	🔴	🟢	🟢	🟢🔴

In new projects I usually start with (1) and then switch to (2), (3) or a combination. Method (3) is good if the operation is not too long and you actually need to show something to the user.

Get in touch

Adi Sarid
adi@sarid-ins.co.il
Download these slides at:
`https://www.overleaf.com/read/rtxncqktyttp`