

Dashboards project

Richard White

2018-10-18

Contents

1	Introduction	1
1.1	Executive summary	1
1.2	What is an automated analysis?	2
1.3	Why not have one project for each automated analysis?	2
1.4	Important repositories	2
2	Integrating the R package into the physical system	3
2.1	Summary	3
2.2	RunProcess.R	3
2.3	0_run.sh	4
2.4	RunTest.R	5
3	R packages	5
3.1	Summary	5
3.2	Requirements	5
3.3	Deployment via travis-ci and drat	5
4	Umbrella Infrastructure	6
4.1	Executive summary	6
4.2	What is an automated analysis?	6
4.3	Why not have one project for each automated analysis?	6

List of Tables

List of Figures

1 Introduction

1.1 Executive summary

The dashboards project is a project at FHI concerned with running automated analyses on data.

In principle, the dashboards project is split up into three parts:

1. The umbrella infrastructure (i.e. Docker containers, continuous integration, cron jobs, etc.)
2. The R package for each automated analysis
3. Integrating the R package into the physical system (e.g. specifying where the data is)

1.2 What is an automated analysis?

An automated analysis is any analysis that:

1. Will be repeated multiple times in the future
2. Always has an input dataset with consistent file structure
3. Always has the same expected output (e.g. tables, graphs, reports)

1.3 Why not have one project for each automated analysis?

Automated analyses have a lot of code and infrastructure in common.

Automated analyses:

1. Need their code to be tested via unit testing to ensure the results are correct
2. Need their code to be tested via integration testing to ensure everything runs
3. Need to be run at certain times
4. Need to be able to send emails notifying people that the analyses have finished running
5. Need to make their results accessible to the relevant people

By combining them all in one umbrella project we can force everyone to use the same infrastructure and coding principles, so we:

1. Only need to solve a problem once
2. Only need to maintain one system
3. Can easily work on multiple projects, as we all speak the same language

1.4 Important repositories

1.4.1 Infrastructure

https://github.com/raubreywhite/dashboards_control/ (private)

This contains the Docker files, cronfiles, all bash scripts, etc.

<https://folkehelseinstituttet.github.io/dashboards/> (this one)

This contains the R executable for each automated analysis.

<https://folkehelseinstituttet.github.io/fhi/>

This is an R package that contains helper functions.

1.4.2 Automated analyses R packages

https://folkehelseinstituttet.github.io/dashboards_sykdomspuls/

https://folkehelseinstituttet.github.io/dashboards_normomo/

https://folkehelseinstituttet.github.io/dashboards_sykdomspuls_pdf/

https://folkehelseinstituttet.github.io/dashboards_sykdomspuls_log/

2 Integrating the R package into the physical system

2.1 Summary

An R package is not enough to run an analysis – something needs to physically call the functions inside the R package. That is, the R package needs to be integrated into the physical system.

Everything related to integrating the R package into the physical system lives in the [dashboards](#) repository.

Inside the [dashboards](#) repository we have:

```
- dev/
|-- src/
|   |-- sykdomspuls/
|       |-- 0_run.sh
|       |-- RunProcess.R
|       |-- RunTest.R
|   |-- normomo/
|       |-- 0_run.sh
|       |-- RunProcess.R
|       |-- RunTest.R
|   |-- sykdomspuls_log/
|       |-- 0_run.sh
|       |-- RunProcess.R
|       |-- RunTest.R
|   |-- sykdomspuls_pdf/
|       |-- 0_run.sh
|       |-- RunProcess.R
|       |-- RunTest.R
```

2.2 RunProcess.R

2.2.1 Aim

An automated analysis needs to:

1. Know the location of the data/results folders.

2. Check for new data in these folders. If no new data - then quit.
3. Load in the data.
4. Load in the analysis functions.
5. Run the analyses.
6. Save the results.

The R executable (commonly called `RunProcess.R`) is responsible for these tasks.

We can think of it as an extremely short and extremely high-level script that implements the analysis scripts.

Depending on the automated analysis, `RunProcess.R` can be run every two minutes (constantly checking for new data), or once a week (when we know that data will only be available on a certain day/time).

2.2.2 Bounded context

1. Only one instance of `RunProcess.R` can be run at a time.
2. Data only exists on physical folders on the system.

Point #1 is important because if `RunProcess.R` is run every 2 minutes (constantly checking for new data) but the analyses take 3 hours to run, then we need to ensure that only one instance of `RunProcess.R` can be run at a time.

Point #2 is important because sometimes:

1. Data files need to be downloaded from external SFTP servers ([normomo](#), [sykdomspul-slog](#)).
2. Results files need to be uploaded to external SFTP servers ([sykdomspuls](#)).

If we include code to download/upload the files from SFTP servers inside `RunProcess.R` then it makes it very difficult to test `RunProcess.R` (because we will then need to simulate SFTP servers inside our testing infrastructure). If we know that `RunProcess.R` only accesses files that are available on physical folders in the system, then our testing infrastructure is a lot easier to create and maintain.

2.3 0_run.sh

2.3.1 Aim

The aim of `0_run.sh` is to ensure that:

1. The bounded context of `RunProcess.R` happens
2. Run `RunProcess.R`

Point #1 is done through the use of `flock`.

(If necessary) point #2 uses `sshpass`, `sftp`, and `ncftpput` to download/upload files from SFTP servers.

2.4 RunTest.R

2.4.1 Aim

The aim of `RunTest.R` is to perform integration testing on the automated analysis. This integration testing is performed as part of the Jenkins build pipeline.

3 R packages

3.1 Summary

Each automated analysis has its own R package:

- [sykdomspuls](#)
- [normomo](#)
- [sykdomspulspdf](#)
- [sykdomspulslog](#)

Each R package contains all of the code necessary for that automated analysis. Typical examples are:

- Data cleaning
- Signal analysis
- Graph generation
- Report generation

3.2 Requirements

The R packages should be developed using unit testing as implemented in the [testthat](#) package.

Furthermore, the R package should operate (and be able to be tested) independently from the real datasets on the system. This is because the real datasets cannot be shared publically or uploaded to github. To circumvent this issue, each package will need to develop functions that can generate fake data. [GenFakeDataRaw](#) is one example from [sykdomspuls](#).

We also require that unit tests are created to test the formatting/structure of results. [ValidateAnalysisResults](#) is one example from [sykdomspuls](#), where the names of the `data.table` are checked against reference values to ensure that the structure of the results are not accidentally changed.

3.3 Deployment via travis-ci and drat

Unit testing is then automatically run using [travis-ci](#). If the R package passes all tests, then we use [drat](#) to deploy a built version of the package to Folkehelseinstituttet's R repository: <https://folkehelseinstituttet.github.io/drat/>.

4 Umbrella Infrastructure

4.1 Executive summary

The dashboards project is a project at FHI concerned with running automated analyses on data.

In principle, the dashboards project is split up into three parts:

1. The overarching infrastructure (i.e. Docker containers, continuous integration, cron jobs, etc.)
2. The R package for each automated analysis
3. The executable for each automated analysis

4.2 What is an automated analysis?

An automated analysis is any analysis that:

1. Will be repeated multiple times in the future
2. Always has an input dataset with consistent file structure
3. Always has the same expected output (e.g. tables, graphs, reports)

4.3 Why not have one project for each automated analysis?

Automated analyses have a lot of code and infrastructure in common.

Automated analyses:

1. Need their code to be tested via unit testing to ensure the results are correct
2. Need their code to be tested via integration testing to ensure everything runs
3. Need to be run at certain times
4. Need to be able to send emails notifying people that the analyses have finished running
5. Need to make their results accessible to the relevant people

By combining them all in one umbrella project we can force everyone to use the same infrastructure, so we:

1. Only need to solve a problem once
2. Only need to maintain one system
3. Can easily work on multiple projects, as we all speak the same language