Migrating a Shiny live dashboard from R Markdown to Quarto

R Basel 2023

Mirai Solutions

July 21, 2023

guido.maggio@mirai-solutions.com

Mirai Solutions

- Zurich-based data science company founded in 2009
- Interdisciplinary team of experienced software engineers, statisticians, quantitative analysts, economists and scientists
- Delivering effective solutions to the financial services industry

More about Mirai Solutions

Services by Mirai Solutions:

- Data analytics & software development: from customer’s needs to a prototype and to a productive implementation
- IT architecture and design, project management
- Training customized to your business case - technical and agile topics

Our links:

- website: **www.mirai-solutions.ch**
- LinkedIn: **linkedin.com/…/mirai-solutions-gmbh**
- email: **info@mirai-solutions.com**
- twitter: **twitter.com/MiraiSolutions**

Agenda

Our presentation

- High level comparison **R Markdown Quarto**
- Why moving from R Markdown to Quarto
- The R Markdown dashboard app we want to migrate
- Replication of the original app with `qmd` report
- Quarto project with a document, what have we lost?
- Considerations

Quarto vs R Markdown

Quarto

An open-source scientific and technical publishing system built on **Pandoc**.
Quarto Documents are authored using markdown, an easy way to write plain text format.
Some features:

- Embedding code and output integrating with `Jupyter` `Knitr` and `Observable`
- **Extensions** to Pandoc Markdown
- It allows a project System
- Various editors and notebooks: *JupyterLab*, *Rstudio*, *books*, *VSC*
- It includes a visual markdown editor

Quarto vs R Markdown - software

Quarto:

- Command Line Interface (CLI), i.e it requires a software installation
- `quarto` is an R package that interfaces with the CLI

R Markdown:

- based on R package `rmarkdown`

Both Quarto and R Markdown are supported by the Posit RStudio IDE.
Quarto is easier to use outside RStudio.

Why from R Markdwon to Quarto

- Quarto offers many out of the box features (website, blogs, etc.), R Markdown must use extension R packages
- Quarto is multi-language and multi engine
- Easy migration, compatibility with Markdown and Jupyter, low effort (replace `rmd` with `qmd`)
- R Markdown is still suitable to R users, however Quarto is likely to be the next standard and focus
- Quarto handles configurations from `YAML` file, standardized and flexible w.r.t. layout and output format
- Quarto's additional features: tab panels, code highlighting, notations, freeze results etc. Many layout-specific **templates available**.

Quarto vs R Markdown - extensions

Markdown requires extension packages to support different applications:

- `blogdown` for blogs
- `bookdown` for books
- `xaringan` for improved presentations
- `distill` for scientific publishing

Quarto offers them all together.

- avoid further R package dependencies
- manage only one configuration file `_quarto.yml`
- facilitate the transition from an application to another one
- easier to specify configurations in only one file.

Quarto vs R Markdown - language

Quarto does not depend on R, unlike R Markdown, it can be called from other languages through its CLI.

Code of different languages can be used (R, Python, Javascript, Julia, flexible for future languages).

Developers with different skill-set can collaborate more easily, and this would fit quite well companies like Mirai Solutions.
Quarto supported engines: `knitr`, `Jupyter`, `Observable`.

Quarto vs R Markdown - others

Some other benefits from Quarto in styling and layout:

- `yml` options are available where R code was required

```
execute:
  echo: false
```

vs

```
knitr::opts_chunk$set(echo = FALSE)
```

- Easier use of content across multiple documents, vs **R Markdown child-documents**.
- More options already for styling your documents in the `qmd`.
- Easier format conversion (e.g. from html to pdf). Multiple formats can be specified from `Quarto 1.3`.

Our app: an R Markdown live article

Our app to migrate

An R package with R Markdown article with Shiny content

In our GitHub it can be visible as a **Public repository**

Developed during the pandemic with the goal to provide a dashboard article analysing the past 4 weekly vaccination reports from **BAG** (*BundesAmt für Gesundheit*). [^Note]

The report compares *Vaccinated* and *Unvaccinated* populations and verifies differences in the impact of *Deaths* and *Hospitalized* figures.

The app is visible in our **gallery**.

Note: BAG stopped providing data in September 2022, the article is currently frozen with no data updates.

Technical features

The **repository** is an R package: `covid19vaccinationch`.

```
remotes::install_github("miraisolutions/covid19-vaccination-ch")
```

- unit tests with `testthat`
  - test also that the app launches
- documentation with `roxygen2`
- package dependency managed with `renv`
- **CI CD** with GitHub Actions (see **`workflow.yml`**)
- **`index.Rmd`** file in the `inst` folder of the package.

It incorporates all the benefits of an R package in terms of robustness, deployment, documentation.

Technical features (2)

Every Tuesday, via GitHub Actions **workflow**, the app fetched BAG data and re-built the datasets stored as `.rds` in the `inst` folder.

```
on
 schedule:
  - cron: "0 15,18 * * 2" # every Tuesday (2) at 15:00 and 18:00 UTC
 - name: Fetch and rebuild latest BAG data
  if: github.event_name == 'schedule'
  run: |
   pkgload::load_all(export_all = TRUE, helpers = FALSE, attach_testthat = FALSE)
   build_data()
  shell: Rscript {0}
```

This allows:

- avoiding expensive operations on the `rmd` file, included in `build_data()` function
- checking data quality as part of **CI** avoiding crashes of the app.
- updating the numbers shown in the text and in the graphs

Technical features (3)

The data are pushed by GitHub actions to the GitHub Repository before re-deploying to **shinyapps.io** via a script `deploy-shinyapps.R`.

```
- name: Check package
  uses: r-lib/actions/check-r-package@v2
- name: Commit and push updated BAG data
  if: github.event_name == 'schedule'
  run: |
    git config --local user.email "actions@github.com"
    git config --local user.name "GitHub Actions"
    git add inst/bag_data/\*
    git commit -m "Update BAG data" || echo "No changes to commit"
    git pull --ff-only
    git push origin
- name: Deploy to shinyapps.io
  # Continuous deployment only for pushes to the main / master branch
  if: github.ref == 'refs/heads/main' || github.ref == 'refs/heads/master'
  env:
    SHINYAPPS_ACCOUNT: ${{ secrets.SHINYAPPS_ACCOUNT }}
    SHINYAPPS_TOKEN: ${{ secrets.SHINYAPPS_TOKEN }}
    SHINYAPPS_SECRET: ${{ secrets.SHINYAPPS_SECRET }}
  run: Rscript deploy/deploy-shinyapps.R
```

Technical features (4)

The `index.Rmd` file contains text and chunks of R code including.

- html tables: `htmlTable`
- static graphs: `ggplot2`
- interactive graphs: `plotly`
- shiny chunks for dynamic graphs: `shiny`

The `runtime: shiny_prerendered` allows speeding up the rendering of the document and of the shiny chunks.
Ideally we would like to maintain all these features.

Migrating to Quarto

Solutions available

What solution to use:

- The Quarto website and blog is ultimately **a static website**
- Quarto Documents support interactive graphs and tables
- Quarto Documents with shiny components require an R session as a backend
- Hosting services like `quarto-pub`, `netlify`, and `gh-pages` serve the html pages of the site, not running the R code necessary to operate the shiny app.

We keep the same original structure and we try a 1 to 1 migration of the original app, keeping the deployment on **shinyapps.io** **https://github.com/miraisolutions/covid19-vaccination-ch-qmd-pkg**

Migration from Rmd to Qmd

- Update the `format` in the YAML and other options, set global options in `_quarto.yml`
- rename the `index.rmd` file into `index.qmd`, or open a new `qmd` file.

```r
knitr::convert_chunk_header("inst/report/index.Rmd",
                            output = "inst/report/index.qmd")
```

- clean up the chunks

```
` `{r ageclasses-pandemic-month-calc, warning=FALSE, include=FALSE, cache=TRUE}`
```

Options are moved into the code using chunk `#|`

```
#| warning: false
#| include: false
```

Plots adjust well in the page without specifying properties (width etc.). No problems observed with `plotly` and html tables in a Quarto Document.

Migration from Rmd to Qmd - shiny

The integration of shiny content in Quarto and Markdown is quite similar. Declare the server side in the chunk option.

```
#| context: server
```

Options for the layout can also be added (only Quarto)

```
#| panel: sidebar
## Code
```

Options to be loaded in `#| context: setup`

```
#| context: setup
```

Data to be made available in server must be loaded with `#| context: data`

```
#| context: data
```

See example: `shinyexample.qmd`. Use `server: shiny` in `YAML`.

Migration from Rmd to Qmd - shiny

Example in quarto.org/docs/interactive/shiny/

UI chunk {r}

```
sliderInput("bins", "Number of bins:", min = 1, max = 50, value = 30)
plotOutput("distPlot")
```

SERVER chunk {r}

```
#| context: server
output$distPlot <- renderPlot({
   x <- faithful[, 2]  # Old Faithful Geyser data
   bins <- seq(min(x), max(x), length.out = input$bins + 1)
   hist(x, breaks = bins, col = 'darkgray', border = 'white',
        xlab = 'Waiting time to next eruption (in mins)',
        main = 'Histogram of waiting times')
})
```

Migration from Rmd to Qmd - shiny

Example in line with our app, where we first have some data preparation. Data preparation chunk {r}

```
#| context: data
data_old_faith <- faithful[, 2]
```

UI chunk {r}

```
sliderInput("bins", "Number of bins:", min = 1, max = 50, value = 30)
plotOutput("distPlot")
```

SERVER chunk {r}

```
#| context: server
output$distPlot <- renderPlot({
   x <- data_old_faith[, 2]  # Old Faithful Geyser data from first chunk
   bins <- seq(min(x), max(x), length.out = input$bins + 1)
   hist(x, breaks = bins, col = 'darkgray', border = 'white',
        xlab = 'Waiting time to next eruption (in mins)',
        main = 'Histogram of waiting times')
})
```

Without `#| context: data`, the code in `#| context: server` will fail to find `data_old_faith`.

Migration from Rmd to Qmd - CI CD

Modifications in the **`worflow.yml`**: use **quarto actions**.
Install Quarto prior to deploying to **shinyapps.io**:

```
- name: Install Quarto
  uses: quarto-dev/quarto-actions/setup@v2
```

Update the deploy script, use the quarto deploy function (`quarto_publish_app`):

```r
file.copy("inst/report/index.qmd", "index.qmd", overwrite = TRUE)
quarto::quarto_publish_app(
  input = "index.qmd",
  server = "shinyapps.io",
  name = "covid-19-vaccination-ch-qmd-pkg",
  title = "Covid19 Vaccination CH",
  account = "miraisolutions",
  render = "local" # to render before deployment
)
```

Conclusions - 1 to 1 Migration

- Easy change from `rmd` to `qmd`
- Package structure could be maintained
- Adjustments in GitHub actions required (quarto actions)
- Update in deployment script, function `quarto_publish_app`
- Care required for the shiny content
- **shinyapps.io** or **Posit Connect** the solutions for publishing interactive documents

The **shiny app** on **shinyapps.io**.

Alternative

A Quarto project

The application port is an R package, what if we try with a Quarto project? See the corresponding **github repository**

- R scripts sourced (`source("R/")`)
- Roxygen manual missing
- `renv` allowed
- Missing `DESCRIPTION` file
- Tests in a separate `Tests.qmd` file, rendered also in CI (‘Render Quarto Project’ step)
- **GitHub Actions** updated removing package check (`renv` step stays)

See the app on **shinyapps.io** with plots grouped in tabs.

Final Remarks

Is it worth moving from Rmd to Qmd?

Definitely worth starting new projects and documents using Quarto, for all the reasons mentioned. Presentations have more options with Quarto, and switching from HTML to PDF is easier.

For applications similar to ours, Quarto is not adding benefits from the architecture point of view.

Quarto allows however using more features in terms of layout, annotations, etc. therefore if you plan to develop an app farther and make it more user-friendly, you can benefit from a move to Quarto.

Pure R users do not have to be in a rush, however Quarto is likely to become the new standard.

References

- **Mirai Solutions Technical Guides** on Renv, GitHub Actions CI-CD, roxygen2.
- Example Shiny package with CI-CD on **Mirai Repo**:
- From Quarto’s documentation, **Interactivity - Shiny**
- **R Quarto Tutorial – How To Create Interactive Markdown Documents**
- **ShinyApp Quarto blog**, how to make a shinyapp as a blog post in quarto blog
- **Quarto or R Markdown**, article on differences
- **Quarto FAQ for R Markdown users**
- **Engineering Shiny deploy chapter**

Thank you!