Quarto はじめて良かった

司馬博文

11/04/2023

Abstract　Quarto は TeX のような使用感で，数式とコードが併存する文章を書き，１つのソースファイルから PDF，HTML，Word，Reveal.js，PowerPoint などの多様な形式に出力できる次世代の執筆環境である．TeX，RStudio，Jupyter Notebook のいずれかに慣れている人であれば，極めて手軽に Quarto を使うことができる．

使い方の概要

導入

本サイトは Quarto と，GitHub Actions によってホスティングされている．

Quarto ではこのような Notebook-like なドキュメントが，極めて簡単に＋凡ゆるフォーマットで作成できる．

特に VSCode の拡張機能と組み合わせれば，RStudio のような隙のない統合開発環境が得られる．[1]

基本的な仕組みとして，自分で作成するのは `.qmd` ファイルのみである．

その後は `quarto render` コマンドにより，

> ! Important
>
> - コードブロックは Jupyter によって処理され，
> - 全体は markdown に変換され，
> - Pandoc によって `pdf`，`html`，`word` など好きな形式に最終出力できる．

拡張機能をオンにした VSCode では `Run Cell` ボタンもあるので，ノートブック全体を毎度ビルドせずとも，コードブロックごとに実行して結果を見ることもできる．

`Ctrl+Enter` で１行ごとに実行できる操作感は RStudio と同じである．

美点

[1] 特に，VSCode ではビジュアルモードでの編集もサポートされており，Jupyter Notebook と全く同じ使用感で始められる．

Tip

- レンダリングがとんでもなく速い．体感で TeX の 10 分の 1 である．
- それでいて数式とコードブロックを併在させることが出来る．なお，明かに TeX を意識していることがわかる使用感になっているし，本の作成も可能としている．
- （ちょっと使いにくい）ブラウザ上ではなく，好きなエディタで動く．Jupyter Notebook が続かない筆者にとって，この点は肝要である．
- 私用の勉強ノートとしても使えると同時に，内容そのままブログとして公開できる．
- プレゼンテーションの作成にも使える．
- すごい細かいが，例えば project type を website としたリポジトリで quarto render をしても，不要なファイルが自動で削除される．このような点がライトユーザーでもとにかく使いやすい．
- さらにインタラクティブな機能を実現したブログを作ってみたい．

YAML Header

各ファイルの冒頭に YAML block を用意することで，ノートブックの詳細を調整できる（参照：HTML Options）．

例えば本ページでは次のとおり：

```yaml
---
title: "Quarto はじめて良かった"
author: "司馬博文"
date: "11/4/2023"
date-modified: "7/7/2024"
categories: [Lifestyle]
abstract: Quarto は TeX のような使用感で，数式とコードが併存する文章を書き，1 つのソースファイルから PDF, HTML, Word, Reveal.js, PowerPoint などの多様な形式に出力できる次世代の執筆環境である．TeX, RStudio, Jupyter Notebook のいずれかに慣れている人であれば，極めて手軽に Quarto を使うことができる．
abstract-title: 概要
format:
  html:
    mainfont: "Gill Sans"
    theme: minty
    css: styles.css
    toc: true
    number-sections: true
    highlight-style: ayu
    code-block-border-left: "#7CC4AC"
    code-overflow: scroll
    toc-title: "目次"
    abstract-title: "概要"
---
```

本文の書き方

数式

本文は markdown 記法で書く．数式も使える：

$$P[|\xi| < t] \leq 2e^{-\frac{t^2}{2\sigma^2}}, \qquad t > 0.$$

コード

また，コードブロックにもコメントアウトと接頭辞の組み合わせ #| を前につけることで YAML で指示が出せる（参照：指示のリスト）．上のコードブロックには

```
```

```
#| label: fig-polar
#| fig-cap: "A line plot on a polar axis"
```

と追加されているために，出力された図にラベリングとキャプションが付いているのである．

```
pip3 install jupyter-cache
```

が必要であることに注意．

Website の作り方

公式 Guide を参考．

Source Branch を main と別ける

まず gh-pages という全く新しいブランチを作成する．既存のリポジトリのコミット履歴とは独立している新しいブランチを作るときは--orphan オプションが利用される．

```
git checkout --orphan gh-pages
git reset --hard # make sure all changes are committed before running this!
git commit --allow-empty -m "Initialising gh-pages branch"
git push origin gh-pages
git checkout main
```

基本 gh-pages ブランチには自分では立ち入らない．

Publish コマンドによるサイトの公開

main ブランチにいることを確認して，

```
quarto publish gh-pages
```

を実行．

GitHub の方の設定 **Settings: Pages** で，Source を gh-pages ブランチの/(root)にしていることを確認すれば，これで無事サイトが公開されていることが確認できる．

GitHub Action の使用

さらに，ローカル上で render するのではなく，コミットする度に GitHub 上でレンダリングしてもらえるように自動化することもできる．こうするとスマホからも自分のサイトが更新できる．

まず，GitHub の設定の **Actions** セクションの **Workflow permissions** から，読み書きの権限を GitHub Action に付与する．

続いて，次の内容のファイルを.github/workflows/publish.yml に書き込む：

```
on:
  workflow_dispatch:
  push:
    branches: main

name: Quarto Publish

jobs:
  build-deploy:
    runs-on: ubuntu-latest
    permissions:
      contents: write
    steps:
      - name: Check out repository
```

```
        uses: actions/checkout@v4

      - name: Set up Quarto
        uses: quarto-dev/quarto-actions/setup@v2
        with:
          tinytex: true  # https://github.com/quarto-dev/quarto-actions/tree/main/setup
        env:
         GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}  # Setting GH_TOKEN is recommended as installing TinyTeX
will query the github API.

      - name: Render and Publish
        uses: quarto-dev/quarto-actions/publish@v2
        with:
          target: gh-pages
          # render: false  # https://quarto.org/docs/publishing/github-pages.html#additional-options
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

途中，`tinytex: true` とすることで，1 つのページを HTML と pdf の両方で閲覧可能になる．本ブログでは，CV のページ でこの機能を使っている．

これで，`main` ブランチにコミットする度に，GitHub 上で `render` が実行されることとなる．

PDF の作り方

LuaLaTeX を使う方法

LuaLaTeX を利用することで日本語を含んだ PDF を作成できる．

```
title: "タイトル"
author: 司馬博文
date: 2023/12/11
format:
  pdf:
    toc: true
    number-sections: true
    urlcolor: minty
    template-partials:
      - ../../../before-title.tex
    keep-tex: true
    block-headings: false
    pdf-engine: lualatex
    documentclass: ltjsarticle
```

LuaLaTeX の注意

```
\int_{\mathbb{R}}
```

のような記法は，pdfLaTeX ではなぜかコンパイルが通るが，LuaLaTeX （や殆どの pdfLaTeX 以外のエンジン）ではエラーになる．

LuaLaTeX の欠点

`ltjsarticle` クラスでは

```
Font \JY3/mc/m/n/10=file:HaranoAjiMincho-Regular.otf:-kern;jfm=ujis at 9.24713pt not loadable: metric data
not found or bad.
<to be read again>
relax
```

```
l.79 \kanjiencoding{JY3}\selectfont
                                \adjustbaseline
```

というエラーが．一方で，bxjsarticle クラスでは

```
LaTeX Error: File `haranoaji.sty' not found.

Type X to quit or <RETURN> to proceed,
or enter new name. (Default extension: sty)

Enter file name:
! Emergency stop.
<read *>
```

というエラーが出る．

ローカルではインストールすれば良いだけであるが，これを GitHub Actions 上で実現する方法を考えあぐねていた．

! 注（TeX Live のアップデート方法）

年度を跨いだ TeX Live manager のアップデートは，次のようにする必要がある：

```
wget http://mirror.ctan.org/systems/texlive/tlnet/update-tlmgr-latest.sh
chmod +x update-tlmgr-latest.sh
sudo ./update-tlmgr-latest.sh
```

GitHub Actions の修正

次のようにして，Set up Quarto と Render and Publish の間に，TinyTeX と haranoaji.sty のインストールを使いすることで，GitHub 上でもレンダリングが可能になる．

```
{yml filename="publish.yml"} - name: 'Install TinyTeX'  # https://github.com/quarto-dev/quarto-actions/tree/main/setup  env:    QUARTO_PRINT_STACK: true    GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}  # Setting GH_TOKEN is recommended as installing TinyTeX will query the github API.  run: |    quarto install tool tinytex --log-level warning    case $RUNNER_OS in       "Linux")          echo "$HOME/bin" >> $GITHUB_PATH          export PATH="$HOME/bin:$PATH"           ;;       "macOS")           TLMGR_PATH=$(dirname $(find ~/Library/TinyTeX -name tlmgr))           echo $TLMGR_PATH >> $GITHUB_PATH           export PATH="$TLMGR_PATH:$PATH"           ;;       "Windows")          TLMGR_PATH=$(dirname $(find $APPDATA/TinyTeX -name tlmgr.bat))          echo $TLMGR_PATH >> $GITHUB_PATH          export PATH="$TLMGR_PATH:$PATH"          ;;       *)          echo "$RUNNER_OS not supported"          exit 1          ;;    esac    echo "TinyTeX installed !"    tlmgr install haranoaji # Install haranoaji.sty  shell: bash
```

ローカルの TinyTeX に haranoaji.sty をインストールする方法

```
tlmgr install haranoaji
```

だと，すでに TeX Live がローカルに存在する場合は，そちらにインストールされてしまう．

```
quarto install tinytex
```

でインストールされる TinyTeX は，ホームディレクトリ下の ~/Liberary/TinyTeX/ にインストールされる．[2]

そこの，tlmgr がインストールされている場所まで行って，

```
tlmgr install haranoaji
```

[2] MaxOS では．

を実行すると良い.

だが，まだ見つからないと言われる……

```
❯ ./tlmgr install haranoaji
tlmgr: package repository https://mirror.las.iastate.edu/tex-archive/systems/texlive/tlnet/ (verified)
[1/1, ??:??/??:??] install: haranoaji [25570k]
running mktexlsr ...
done running mktexlsr.
tlmgr: package log updated: ~/Library/TinyTeX/texmf-var/web2c/tlmgr.log
tlmgr: command log updated: ~/Library/TinyTeX/texmf-var/web2c/tlmgr-commands.log
```

Typst を用いる方法

HP

使うフォントは次のように，Google Fonts を通じて，GitHub Actions 上でインストールすることもできるだろう：

```
wget https://github.com/google/fonts/raw/main/ofl/bizudpgothic/BIZUDPGothic-Regular.ttf
wget https://github.com/google/fonts/raw/main/ofl/bizudpgothic/BIZUDPGothic-Bold.ttf
```

スライドの作り方

Bibliography