Automating your R-based research project

Introduction to GitHub Actions
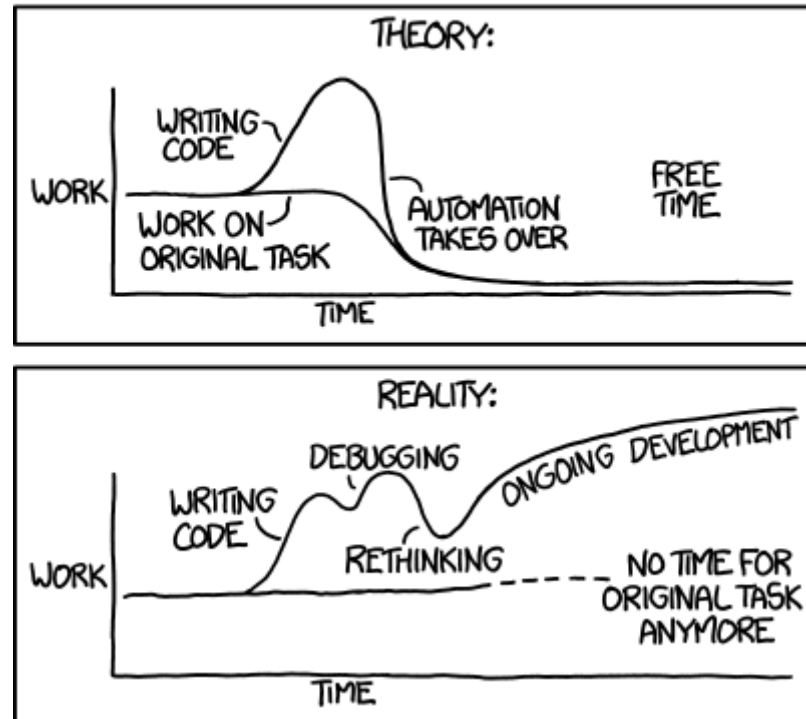
Ernest Guevarra

2022-03-13

Outline

- Overview of automation

- Introduction to GitHub Actions

- Demonstration of automation

Automation

"I SPEND A LOT OF TIME ON THIS TASK.
I SHOULD WRITE A PROGRAM AUTOMATING IT!"

THEORY:

WORK

WRITING CODE

WORK ON ORIGINAL TASK

AUTOMATION TAKES OVER

FREE TIME

TIME

REALITY:

WORK

WRITING CODE

DEBUGGING

ONGOING DEVELOPMENT

RETHINKING

NO TIME FOR ORIGINAL TASK ANYMORE

TIME

'Automating' comes from the roots 'auto-' meaning 'self-', and 'mating', meaning 'screwing'
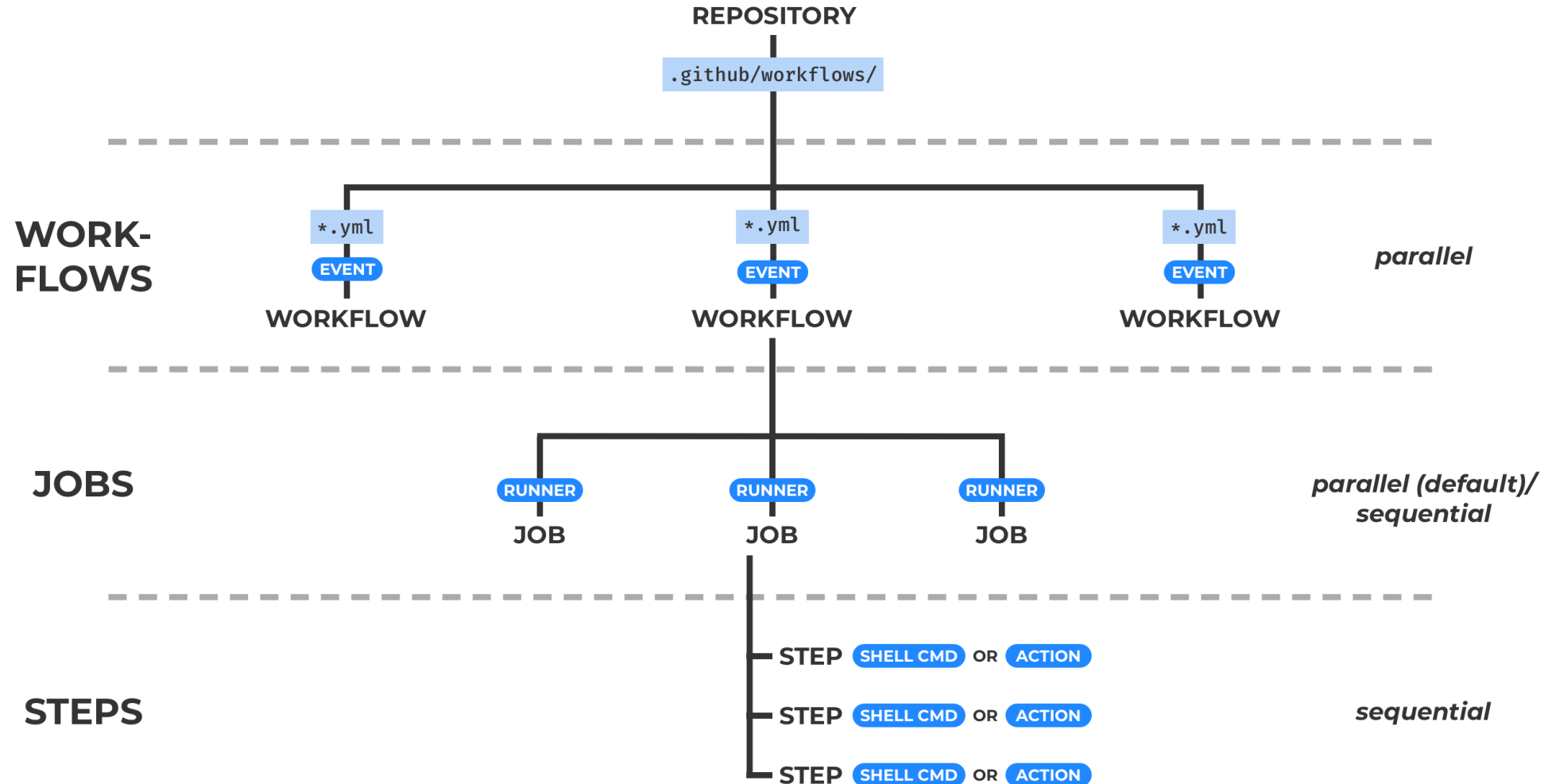
What can be automated?

- works best with a **project workflow** (sometimes called a **project pipeline**)

  - our recommended solution for this is the `{targets}` workflow/pipeline

- works best if project workflow is **portable**

  - our recommended solution for this is `{renv}`

Why automate?

- to reproduce previous results

- to recreate results deleted by *fat fingers*

- to rerun the workflow/pipeline with updated software

- to run the same workflow/pipeline on a new dataset

GitHub Actions

REPOSITORY

`.github/workflows/`

WORK-FLOWS

`*.yml`   `*.yml`   `*.yml`

EVENT   EVENT   EVENT

WORKFLOW   **WORKFLOW**   **WORKFLOW**

parallel

JOBS

RUNNER   RUNNER   RUNNER

JOB   **JOB**   **JOB**

parallel (default)/ sequential

STEPS

STEP SHELL CMD OR ACTION

STEP SHELL CMD OR ACTION

STEP SHELL CMD OR ACTION

sequential

Example GitHub Actions automation

- Example automation built-in to the latest assignment on creating targets workflow

- Can be found at `.github/workflows/check-assignment.yml`

Setting the `trigger` for the automation

```
name: check assignment

on:
  push:
    branches: [main, master]
  pull_request:
    branches: [main, master]
  workflow_dispatch:
    branches:
      - '*'
```

Defining jobs to be implemented in the workflow

```
jobs:
  check-assignment:
    runs-on: ubuntu-latest
    container: rocker/geospatial:4.1.2
    steps:
      - uses: actions/checkout@v2
```

Specifying `steps` in the workflow

Workflow step 1: reproduce the system environment

```
- name: Install system dependencies
  run: |
    apt-get update && apt-get install -y --no-install-recommends \
    git-crypt libxt6 libglpk-dev
```

Workflow step 2: Reproduce the R project-local environment

```
- name: Install packages from renv.lock (with cache)
  uses: r-lib/actions/setup-renv@master
  with:
    cache-version: 2
```

Specifying `steps` in the workflow

Workflow step 3: Perform tests/checks

```
- name: Check that targets workflow has been enabled
  run: |
    testthat::test_that("Target script file exists",
      testthat::expect_true(
        list.files(pattern = "_targets.R") == "_targets.R"
      )
    )
  shell: Rscript {0}

- name: Check that renv has been enabled/initialised
  run: |
    testthat::test_that("renv has been enabled", {
      testthat::expect_true(
        list.files(pattern = "renv.lock") == "renv.lock",
      )
      testthat::expect_true(
        list.files(pattern = ".Rprofile", all.files = TRUE) == ".Rprofile"
      )
      testthat::expect_true(
        any(list.dirs(recursive = FALSE) == "./renv")
      )
```

Specifying `steps` in the workflow

Workflow step 4: Run the targets workflow

```
- name: Check that targets workflow runs without errors
  run: |
    targets::tar_make()
  shell: Rscript {0}
```

Questions?

Thank you!

Slides can be viewed at https://oxford-ihtm.io/open-reproducible-science/session11.html

PDF version of slides can be downloaded at https://oxford-ihtm.io/open-reproducible-science/pdf/session11-automating-r-projects.pdf

R scripts for slides available here