



Appsilon
DATA SCIENCE

Be proud of your code!

Tools and patterns for making production-ready R code

Marcin Dubel

marcin@appsilon.com

17 - 19 June | eRum 2020

Marcin Dubel

marcin@appsilon.com

Data Scientist and
Software Engineer

Helping teams optimize
R project structures



Challenges Ahead Data Science Teams

CREATE VALUE

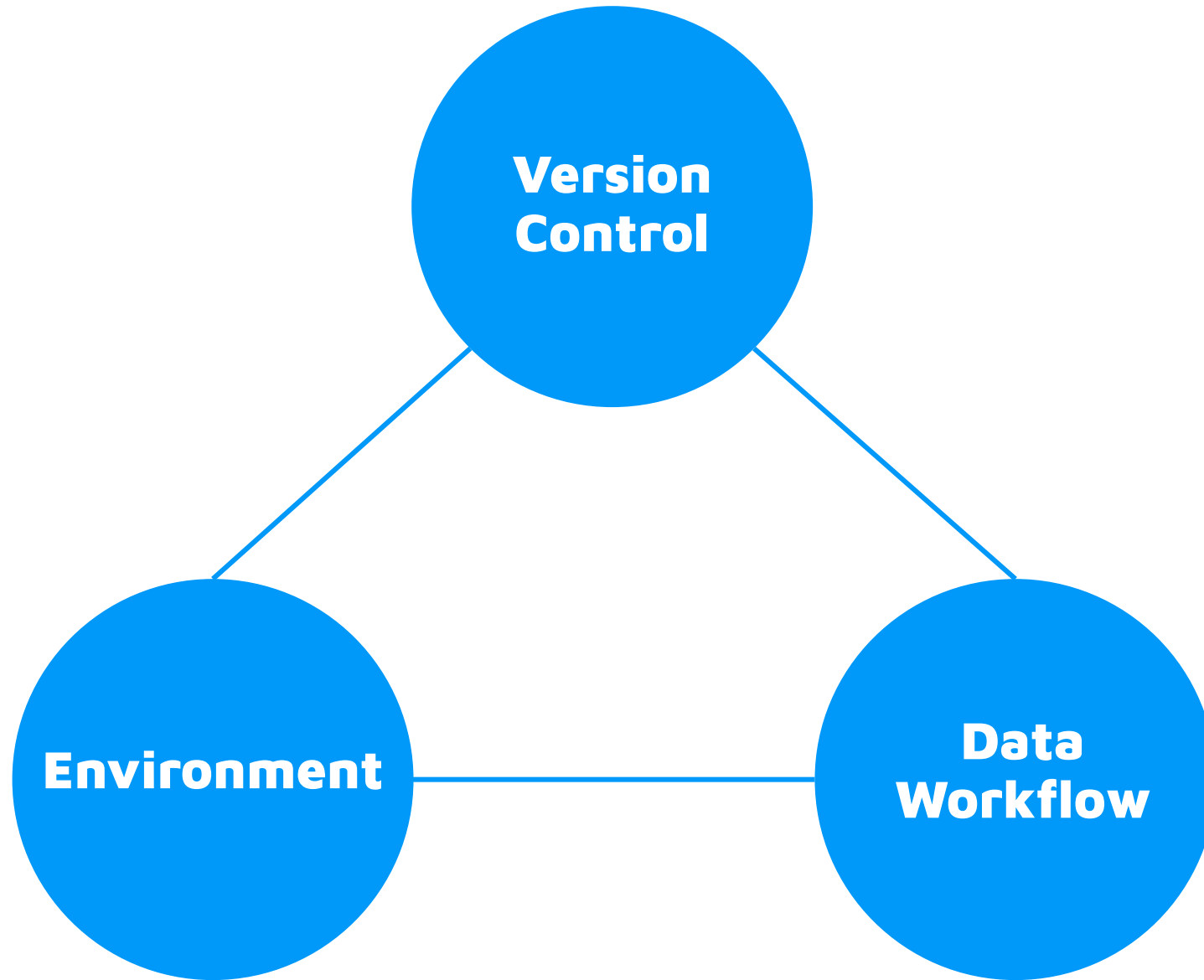
models
analysis
predictions
genomic plots

SHARE VALUE

reproducibility
stability
scalability
interactivity

from **Edwin Thoen's** talk @ eRum 2020





Version Control

Follow the Rules

Version Control

→ START STRONG

Use version control from the very start. This allows you to start collaborating at any time and helps track changes from the beginning.

→ USE BRANCHES

Branches let you collaborate easily and independently. They prevent you from getting stuck on a difficult or problematic feature.

→ INFORMATIVE COMMITS

Clearly describe each commit. Then you can easily find a change in the project's history and diagnose issues.

→ MANAGE TASK BOARD

Organize and share your work in a clear way. Prevent redundant coding with a clear project map.

Continuous Integration

Github Actions

→ TRIGGER ON EVENT

Automatically run e.g. unit tests before each merge and check the build on different operating systems

→ READY TO USE TOOLS

Find what you want in the marketplace or build your own

Marketplace / Search results

Types

Apps

Actions

Categories

API management

Chat

Code quality

Code review

Continuous integration

Dependency management

Deployment

IDEs

Learning

Localization

Mobile

Monitoring









Project management

Search for apps and actions

Actions

An entirely new way to automate your development workflow.

3934 results filtered by Actions

- **ansible-lint**
By ansible ✓
Run Ansible Lint
62 stars
- **Trigger Buildkite Pipeline**
By buildkite ✓
A GitHub Action for triggering a build on a Buildkite pipeline
14 stars
- **GitHub Action for Cloudflare Workers**
By cloudflare ✓
Deploy a Cloudflare Worker with the Serverless Framework
59 stars
- **Velocity deploy action**
By codeclimate ✓
A GitHub Action for sending deployment information to Velocity
16 stars
- **Codecov**
By codecov ✓
GitHub Action that uploads coverage reports for your repository to codecov.io
325 stars
- **Coveralls GitHub Action**
By coverallsapp ✓
Send test coverage data to Coveralls.io for analysis, change tracking, and notifications
121 stars
- **Glo Parse Card Links**
By AxoSoft ✓
GitHub Action to parse links to Glo Boards cards
- **Glo Add Label To Cards**
By AxoSoft ✓
GitHub action to add a label to Glo Boards cards

Continuous Integration

Github Actions

→ TRIGGER ON EVENT

Automatically run e.g. unit tests before each merge and check the build on different operating systems

→ READY TO USE TOOLS

Find what you want in the marketplace or build your own

GitHub Actions documentation:

<https://help.github.com/en/actions>

Actions Marketplace

<https://github.com/marketplace?type=actions>

GitHub Actions with R

https://ropenscilabs.github.io/actions_sandbox/

Examples of Actions for R

<https://github.com/r-lib/actions>

R package with Actions templates

<https://github.com/maxheld83/ghactions>



Github Template

ProTip

collabrica / shakecast

Watch ▾

88

★ Star

247

🍴 Fork

14

<> Code

🔔 Issues 1

🔗 Pull requests 0

▶ Actions

📁 Projects 0

📖 Wiki

🛡 Security

📊 Insights

⚙ Settings

Options

[Collaborators](#)

[Branches](#)

[Webhooks](#)

[Notifications](#)

[Integrations & services](#)

[Deploy keys](#)

[Secrets](#)

[Moderation](#)

[Interaction limits](#)

Settings

Repository name

shakecast

Rename

☒ Template repository

Template repositories let users generate new repositories with the same directory structure and files. Indicate if collabrica/shakecast can be used as a template for creating other repositories.

Social preview

Upload an image to customize your repository's social media preview.

Images should be at least 640×320px (1280×640px for best display).

[Download template](#)



Environment

Lack of Development Environment

antipattern



CODE ERRORS

The same code can behave differently on different machines, resulting in errors.



HARD DEPLOYMENTS

Establishing infrastructure is challenging because it is not known which packages in which versions are used in the code.



AFFECTING OTHER PROJECTS

In a global environment, shared packages can crash unrelated projects that depend on those packages.



PAINFUL COLLABORATION

Other team members will waste time setting up a new environment rather than jumping into code work.

renv to the rescue!



→ SEPARATE ENVIRONMENTS

renv creates an independent environment for each project so projects do not affect each other!

→ SHAREABLE STATES

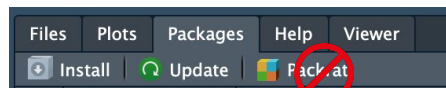
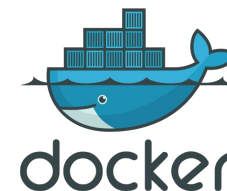
The current state of the environment is saved in a single file that can be shared with the team.

→ SIMPLE SETUP

In contrast to **Packrat**, using **renv** is simple and intuitive.

→ SHARE DOCKER IMAGES

[Building new images](#) is super fast because a snapshot is enough. All package information is in *renv.lock*



Development Environment

renv

<https://rstudio.github.io/renv/articles/renv.html>

1) **renv::init()**

Initializes new local environment, adds *renv.lock* file

2) **renv::install("package")**

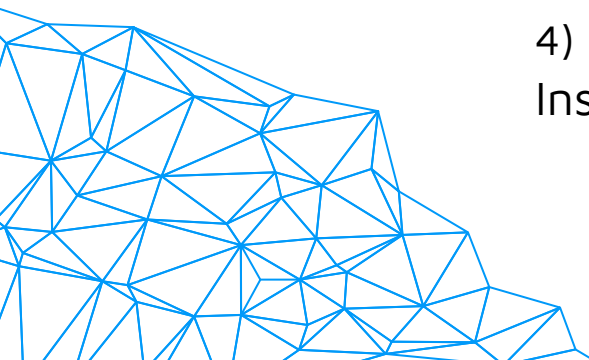
Install new package or take packages from cache

3) **renv::snapshot()**

Saves the current state of your local library to *renv.lock*

4) **renv::restore()**

Installs all packages based on *renv.lock*



Data Workflow

Keep the Data Process Clean

drake

<https://github.com/ropensci/drake>

→ ORGANIZE YOUR PROJECT

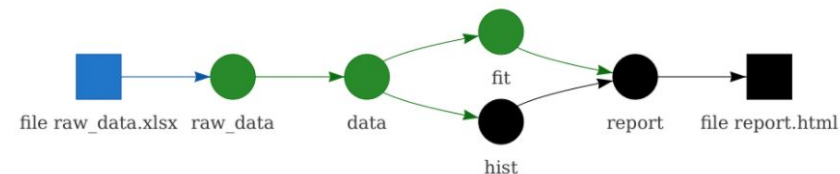
Define the connections between all files, functions, and results in a single plan.

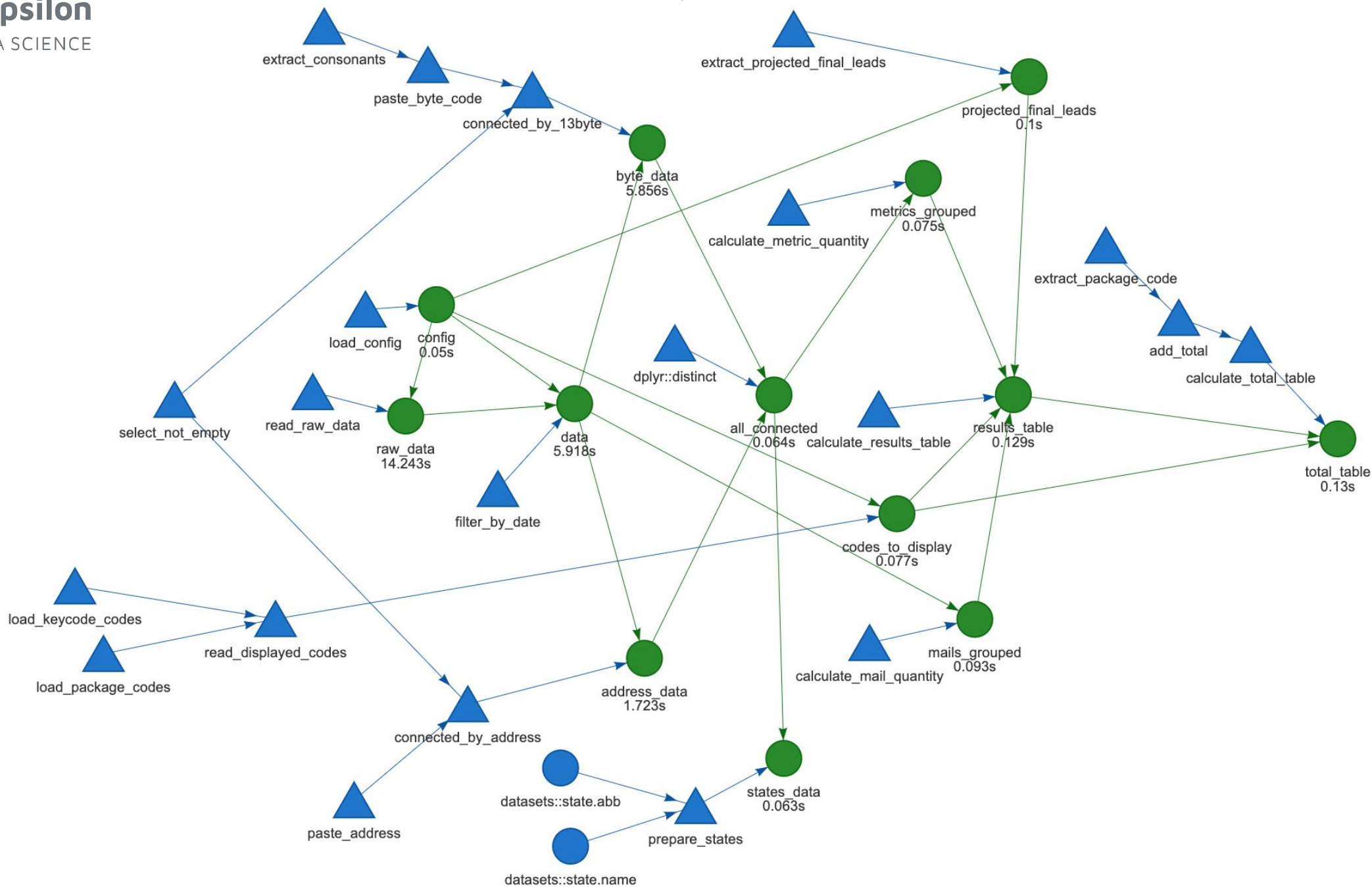
→ CLEAR VISUAL STRUCTURE

Visualise the flow which makes it easy for explanation and documentation.

```
plan <- drake_plan(
  raw_data = readxl::read_excel(file_in("raw_data.xlsx")),
  data = raw_data %>%
    mutate(Ozone = replace_na(Ozone, mean(Ozone, na.rm = TRUE))),
  hist = create_plot(data),
  fit = lm(Ozone ~ Wind + Temp, data),
  report = rmarkdown::render(
    knitr_in("report.Rmd"),
    output_file = file_out("report.html"),
    quiet = TRUE
  )
)
```

Dependency graph





data.validator

New Package from Appsilon

<https://github.com/Appsilon/data.validator>



→ SIMPLE

Easy to create validation rules based on **assertr** package.

→ SHAREABLE

Produces visual interactive HTML report, opened in the browser and easily shareable.

→ AUTOMATED

The report can be triggered manually or added to the automated data update process.

Data validation report

2020-03-30 13:12:59

4

Failed

1

Warnings

1

Passed

mtcars

4 Failed

Validation rule	Status	Error details
vs and am values equal 0 or 2 only	✗	Show
For wt and qsec we have: $\text{abs}(\text{col}) < 2 * \text{sd}(\text{col})$	✗	Show
Column drat has only values larger than 3	✗	Show
Each row sum for am:vs columns is less or equal 1	✗	Show

1 Warnings

Loading Data Into Shiny

plumber

<https://www.rplumber.io/>



→ LOAD ONLY WHAT IS NEEDED

The entire dataset is rarely needed in the application. Usually the first user action within the app is to filter/select a subset of data. First, select – then load.

→ USE EFFICIENT DATA LIBRARIES

Manipulate data with package like **fst**, **data.table**, **arrow**. These packages will often improve app performance.

→ BUILD REST API

Wrap data extraction logic into a simple API with **plumber** by adding special comments.

→ DEPLOY EASILY

Use RStudio Connect or Docker to host your API.

Takeaways

- ALWAYS USE VERSION CONTROL AND USE IT PROPERLY
- SET PROJECT WORKING ENVIRONMENT WITH RENV AND/OR DOCKER
- ORGANISE AND VALIDATE THE DATA, USE PLUMBER WHEN NEEDED

Let's discuss!

 @DubelMarcin

 marcin@appsilon.com

appsilon.com

