



Harmonizing your Traditional Workflow with R

PHUSE US Connect 2022

Nick Masel (Janssen), Mike Stackhouse (Atorus)

Peter Sas
Spring in the Forest

Why Harmonize?

Consistency is key

- Different language, not different process
- Want a consistent and understandable workflow for users

Project structures should be standardized

- Setups should generally be future proof
- Hard coded paths are a no-no

"Harmonize" – not mimic

- Let R be R
- Let SAS[®] be SAS[®]
- Leverage capabilities that are *helpful* and *practical* for highly standardized environments

Worthwhile Capabilities to Replicate



SAS[®] Autos

Hierarchical search path to load available SAS[®] macros

Look at the [docs](#)



SAS[®] Libnames

Hierarchical search path to search for available datasets on disk

Look at the [docs](#)

Autos? Why not Packages?

Not everything belongs in a package – but reusable code still belongs in a function

- Might be reusable across an analysis
- Might be reusable across a protocol

Want those functions to automatically be available

- Auto-load the function into the session
- Minimal set up from a standard user perspective

Why are "libnames" Useful?

Don't hardcode
directory paths

Process should hold up to
system migrations

Reproducibility
requirements could outlast
the lifespan of the system

What about
packages like [here](#)

Depends on locality

*Let's all agree that no one
can agree on standard
directory structures*

What if one dataset can
live in multiple locations?

"libnames" work on
a search path

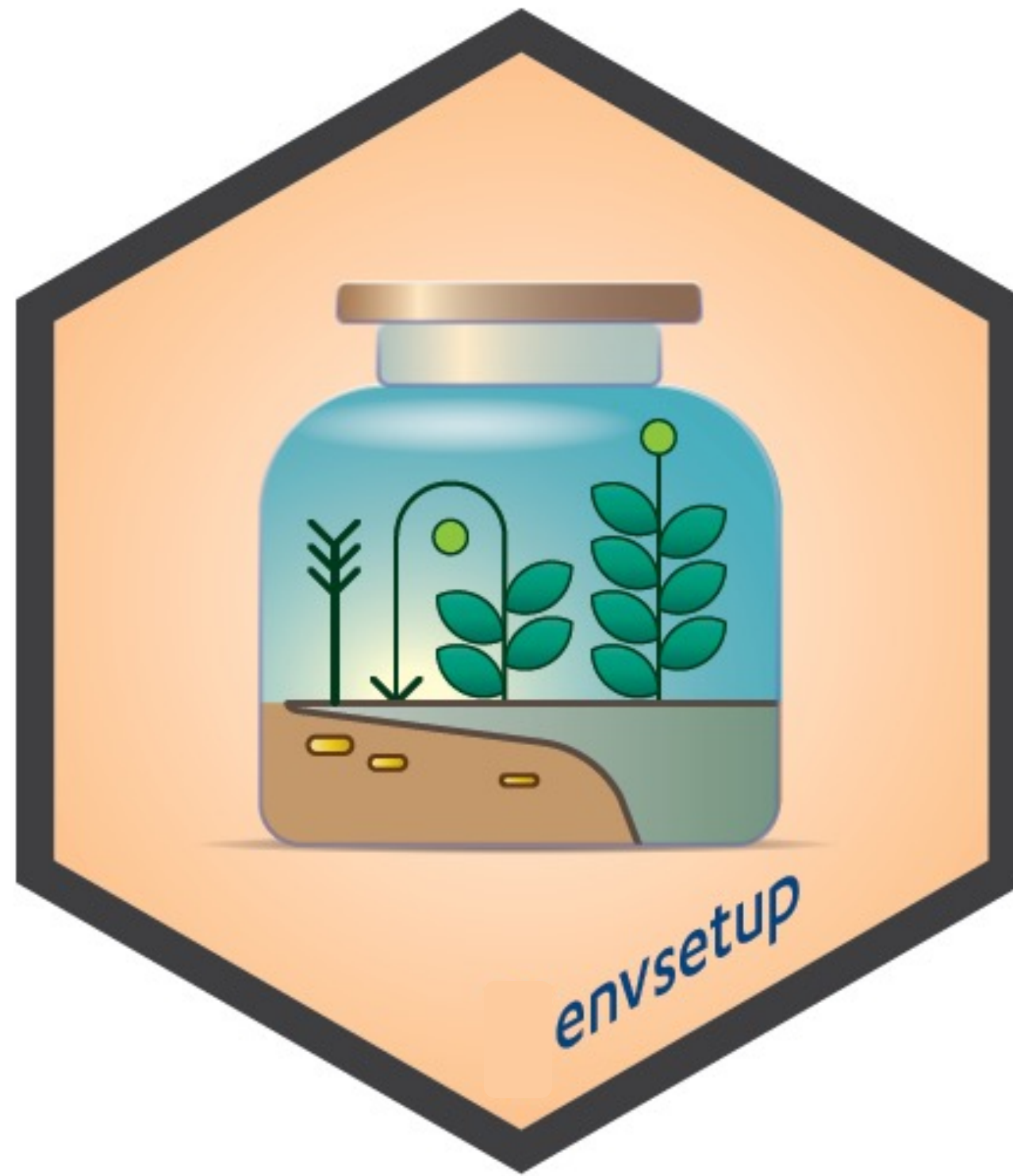
Allows for portability

Works on a hierarchy

Simple named reference

So how do we handle this in





envsetup

A Package Supporting your R environment setup



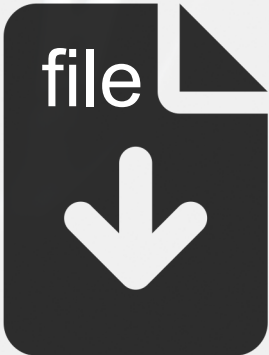
Photo by [Markus Spiske](#) on [Unsplash](#)
Photo by [Lili Popper](#) on [Unsplash](#)
Photo by [Girl with red hat](#) on [Unsplash](#)

autos

source()

envir

attach()



1



2



paths

- One or multiple paths per path object (i.e. dev, qa, prod)
- Stored in path environment
- Environment on search path
- Read and Write helper functions



Customized Configuration

_envsetup.yml

config package



autos

```
default:
  autos:
    my_script: "test/PROD/project1/functions/my_script.R"
    dev: "test/DEV/username/project1/functions"
    qa: "test/QA/project1/functions"
    prod: "test/PROD/project1/functions"
```

paths

```
default:
  paths:
    data: !expr list(DEV = "test/DEV/username/project1/data"),
              QA = "test/QA/project1/data"),
              PROD = "test/PROD/project1/data")
    programs: !expr list(DEV = "test/DEV/username/project1/programs"),
              QA = "test/QA/project1/programs"),
              PROD = "test/PROD/project1/programs")
    functions: !expr list(DEV = "test/DEV/username/project1/functions"),
              QA = "test/QA/project1/functions"),
              PROD = "test/PROD/project1/functions")
    output: !expr list(DEV = "test/DEV/username/project1/output"),
              QA = "test/QA/project1/output"),
              PROD = "test/PROD/project1/output")
```

Additional envsetup Features

Templates, renv, Addins

.Rprofile template

- activate renv
- Read `_envsetup.yml` configuration file
- `envsetup::rprofile()`
 - autos
 - path

renv.lock

- One lock file per unique use case or per container
- Store centrally
- Retrieve at project setup to initialize
- `isTRUE(Interactive == Batch)`

Addins

- Helper functions!
- Easily select correct setup for the use case or container
- Copy or link appropriate files to project location

Q&A

Viral exacerbation at 40x magnification