Reproducing you Environment

RStudio®

Game Plan

- **Strategies** for creating reproducible environments
- **Use cases** for reproducible environments
- **Tools** to implement a strategy for a use case

I want _____ with _____ using _____
(use case) (strategy) (tools)

RStudio®

Game Plan

- **Strategies** for creating reproducible environments
- **Use cases** for reproducible environments
- **Tools** to implement a strategy for a use case

I want _____ with _____ using _____
(use case) (strategy) (tools)

I want to bake a cake with Mary Berry's recipe using an oven.

RStudio

Game Plan

- **Strategies** for creating reproducible environments
- **Use cases** for reproducible environments
- **Tools** to implement a strategy for a use case

I want _____ with _____ using _____

(use case) (strategy) (tools)

I want to bake a cake with Mary Berry's recipe using an oven.

RStudio

Game Plan - Specifics

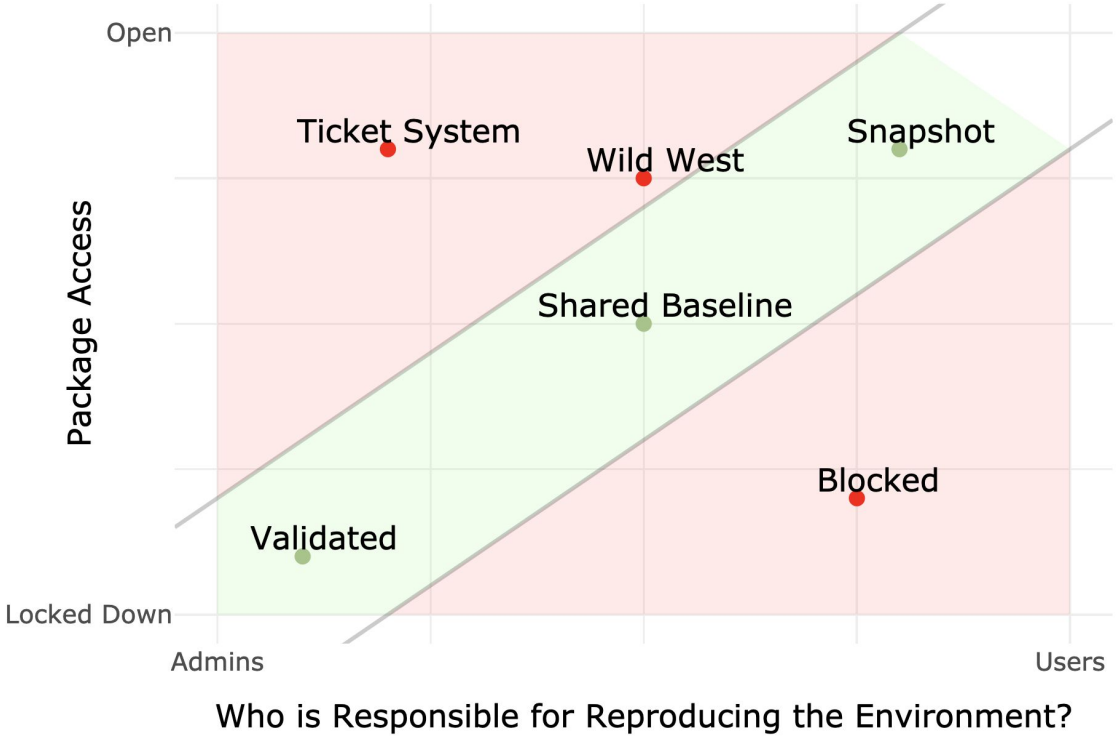
- Strategy Map

- ***Collaborating on a team*** with a **Shared Baseline** using a **Frozen Repository**
- ***Safely upgrading packages*** with **Snapshot and Restore** using **renv**
- ***Using approved packages*** with **Validation** using **Docker**

https://environments.rstudio.com/

RStudio

Strategy Map

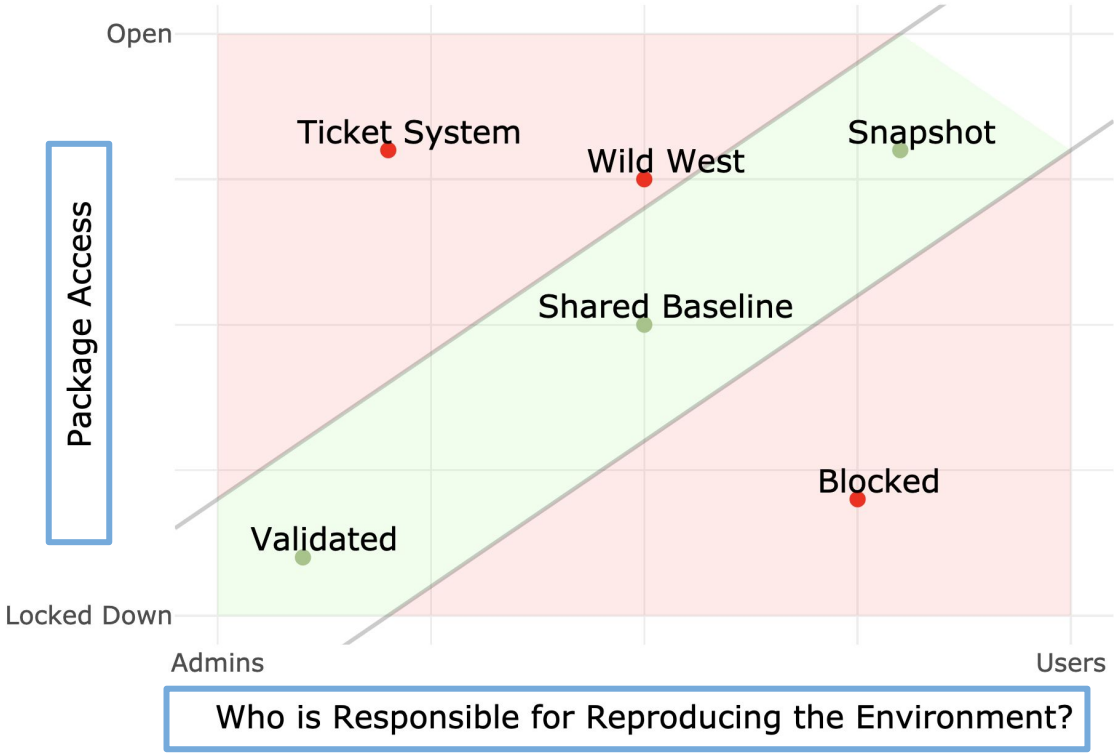


RStudio

Strategy Map

1. Who is responsible?

2. Are there restrictions? (e.g. Licensing, approval, test coverage)

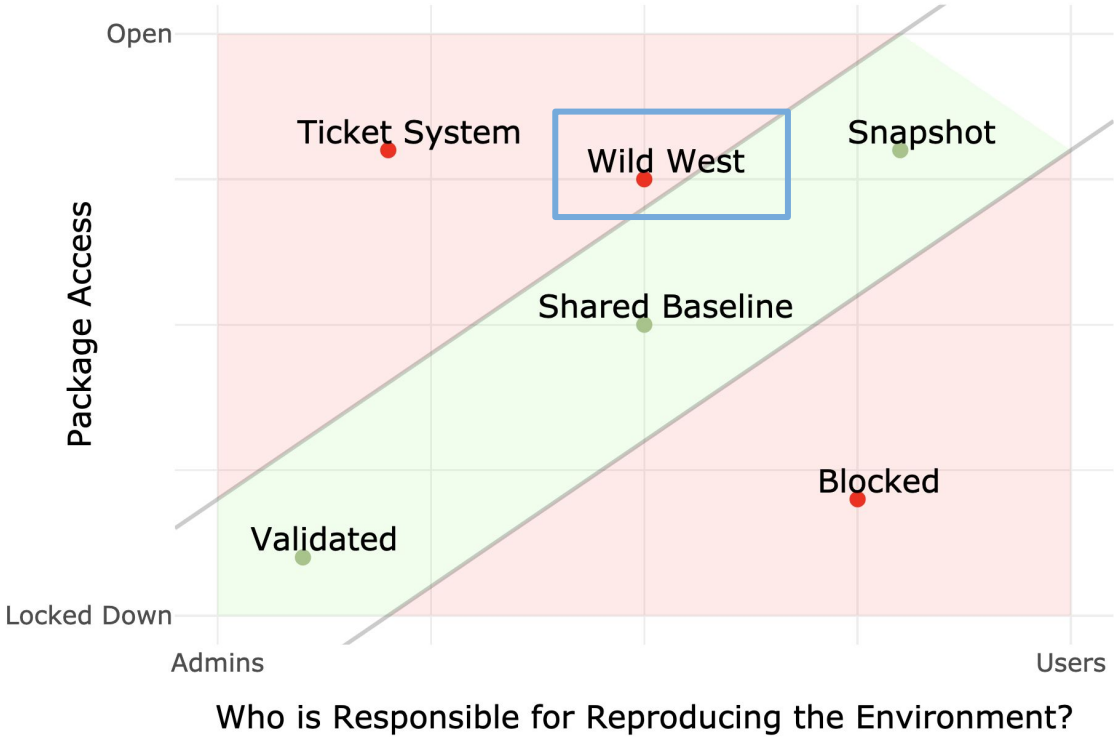


RStudio

Strategy Map

Wild West

- Where we learn
- Open access
- No one is responsible

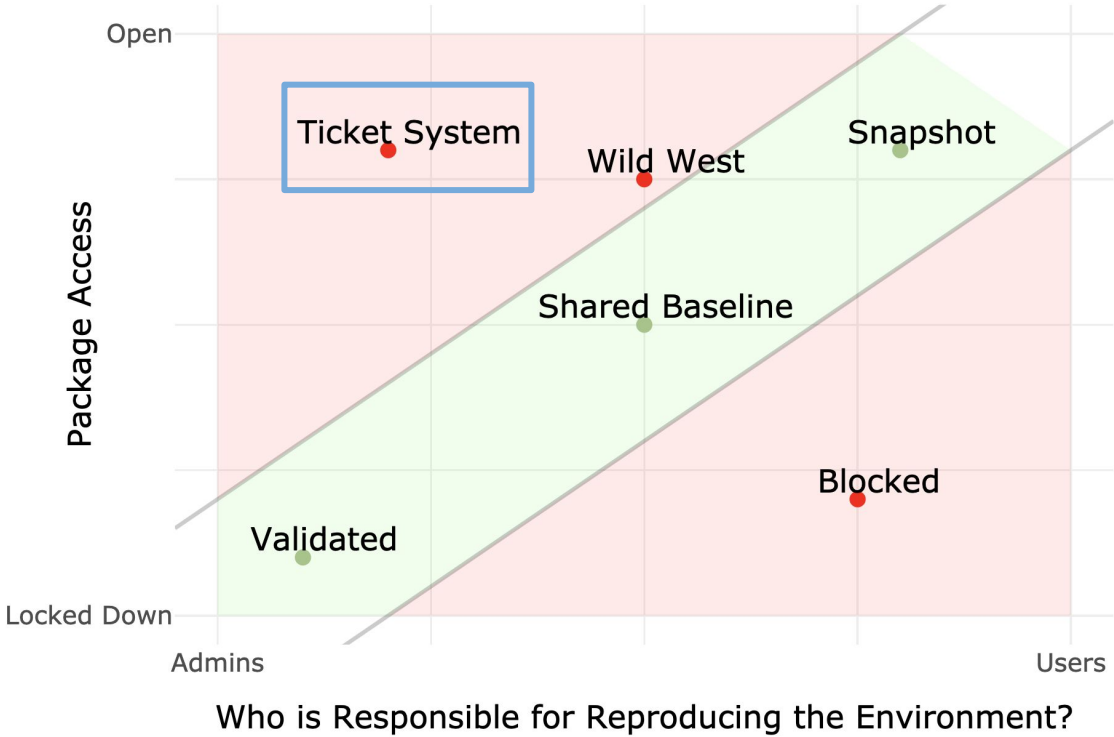


RStudio

Strategy Map

Ticket System

- Admins are mechanically responsible
- Still open access - just slow
- Upgrades often break things

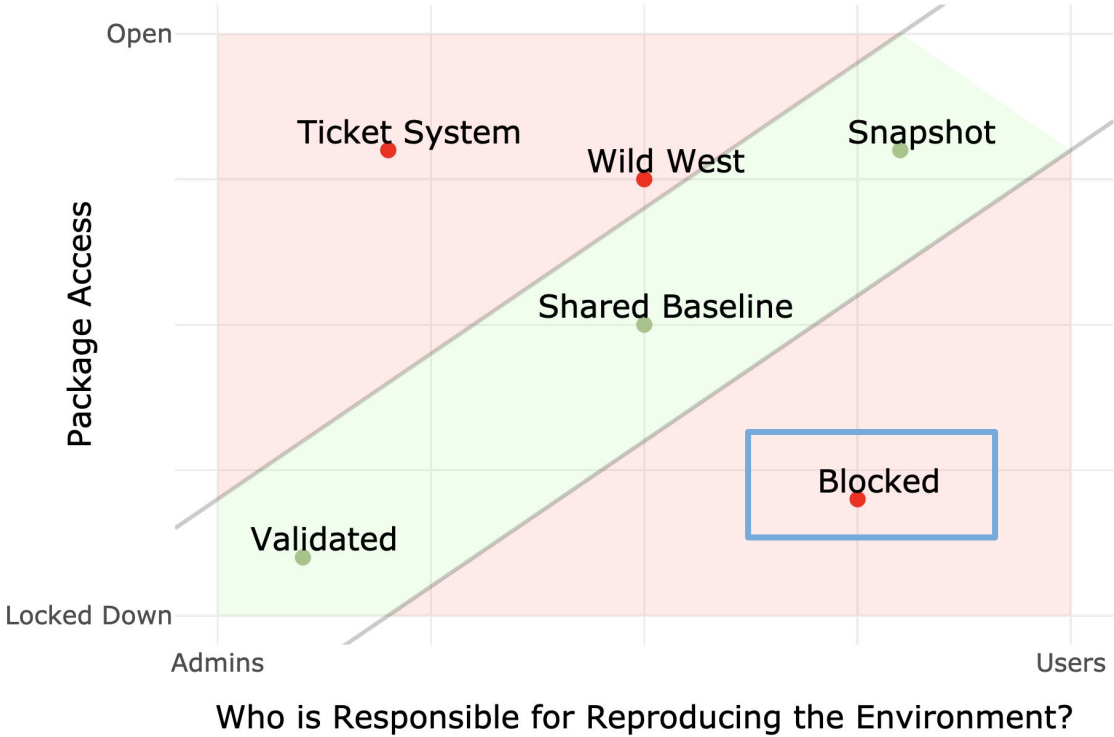


RStudio

Strategy Map

Blocked

- Environment is locked down
- No affordance for packages
- Backdoor behavior

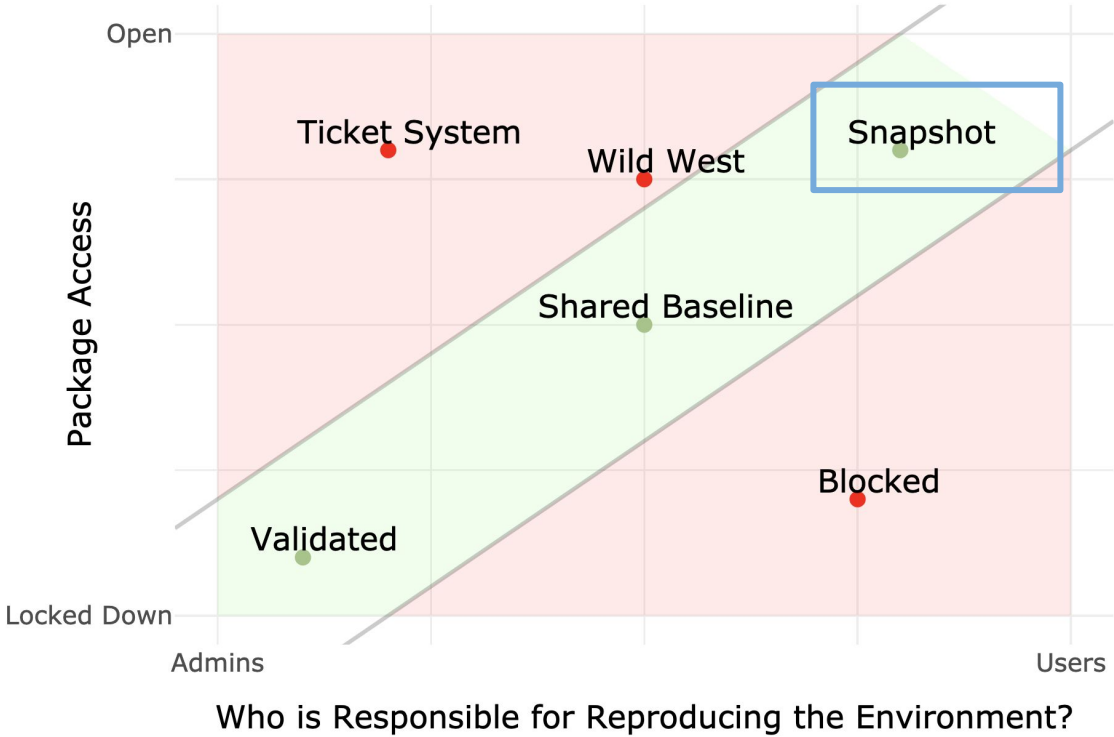


RStudio

Strategy Map

Snapshot

- Users record what they are doing

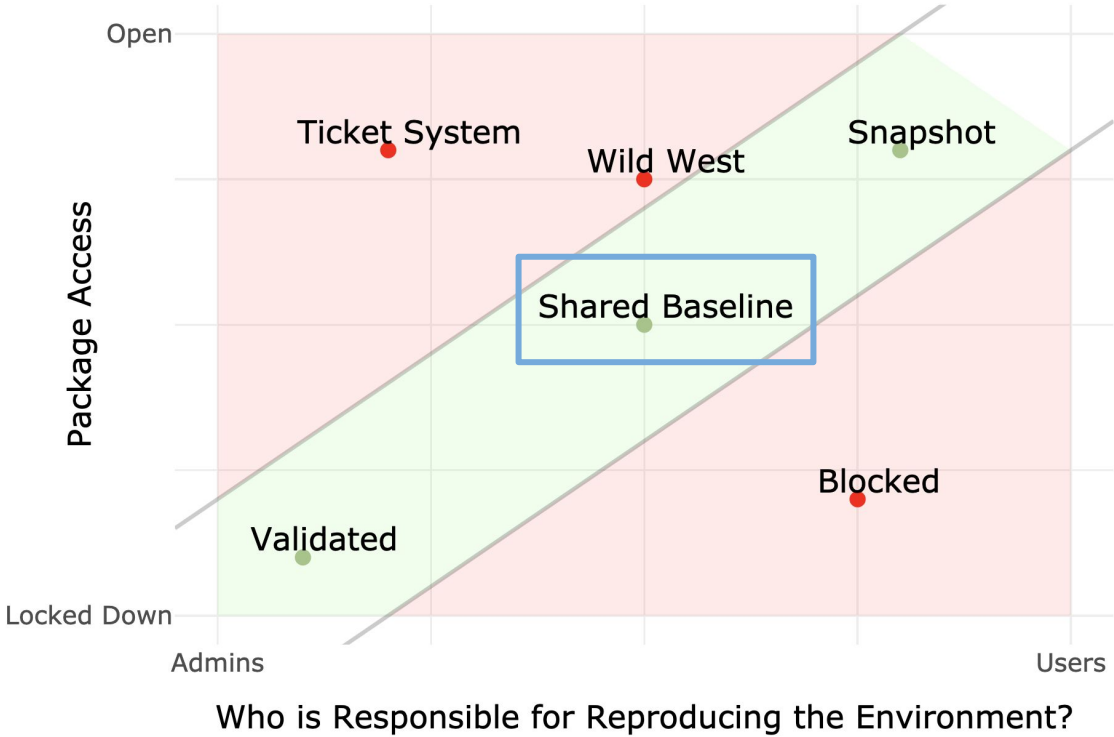


RStudio

Strategy Map

Shared Baseline

- Admins create stable baseline environments
- Immediate package access is traded for consistency and stability

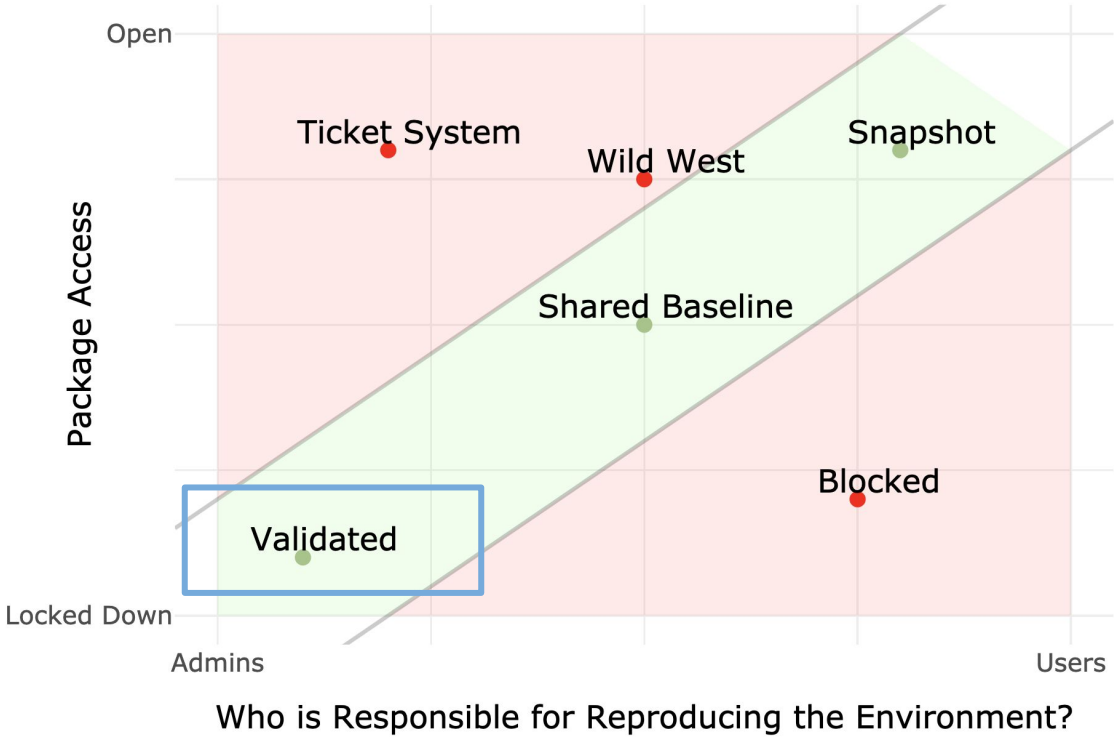


RStudio

Strategy Map

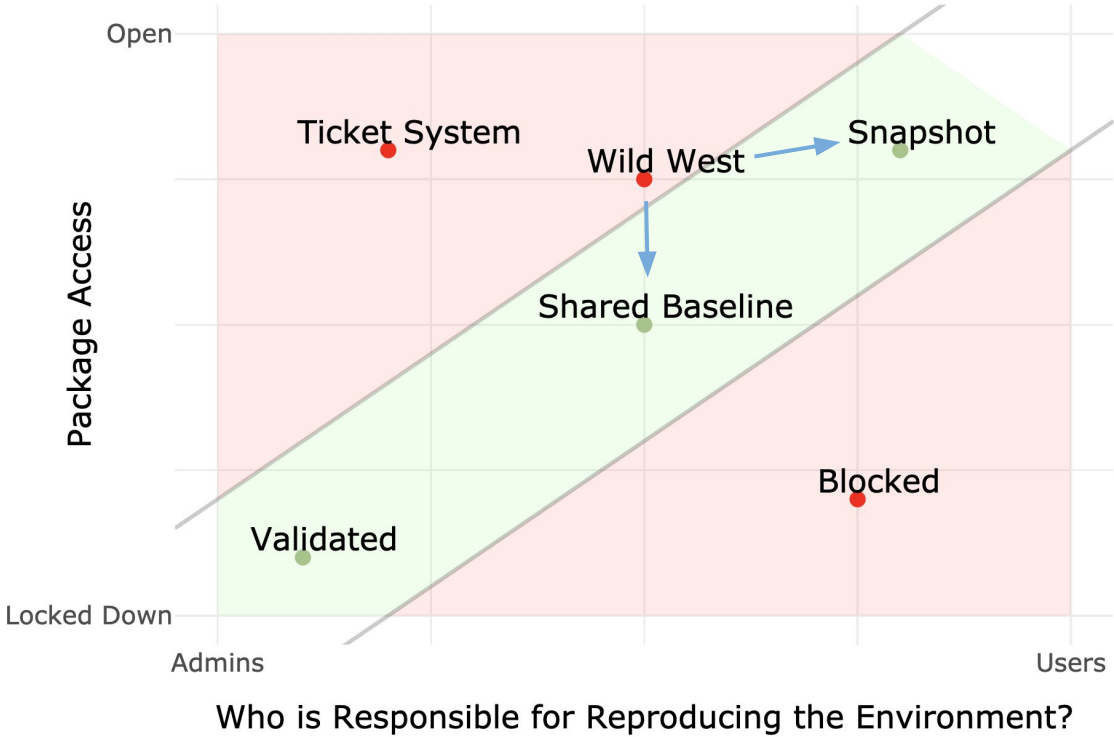
Validated

- Admins control **and test** the environment with approved package subsets



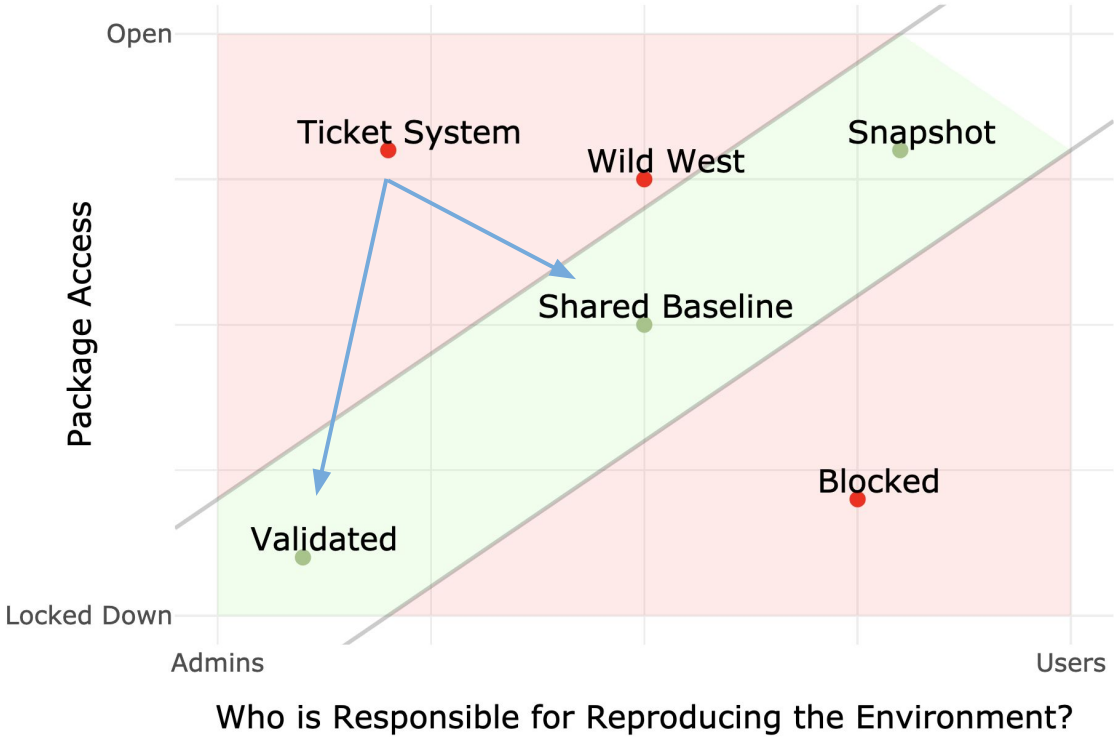
RStudio

Strategy Map



RStudio

Strategy Map



RStudio

Exercise

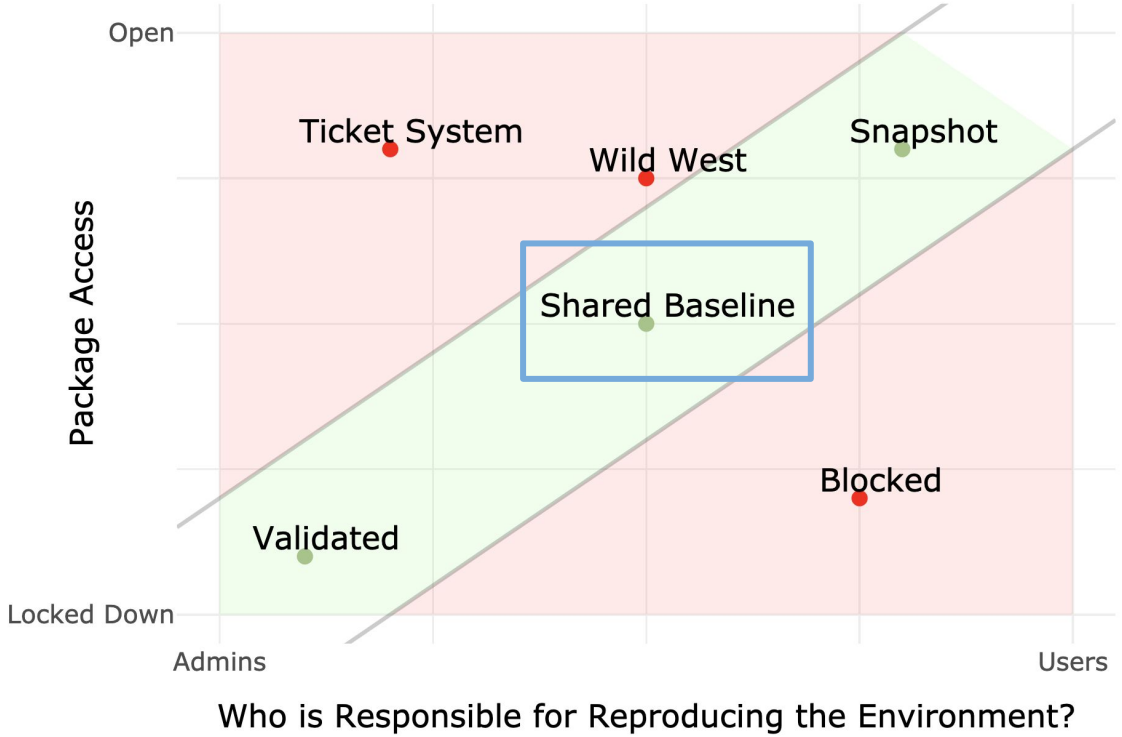
Form a group of 3-4 and discuss:

1. Where is your team currently?
2. What strategy(ies) do you think are a good fit?

Collaborating on a team with a **Shared Baseline** using a **Frozen Repository**

Goal: *Share code with others and they can easily run it*

Implied Goal: Everyone has quick access to the same sets of installed packages (library)

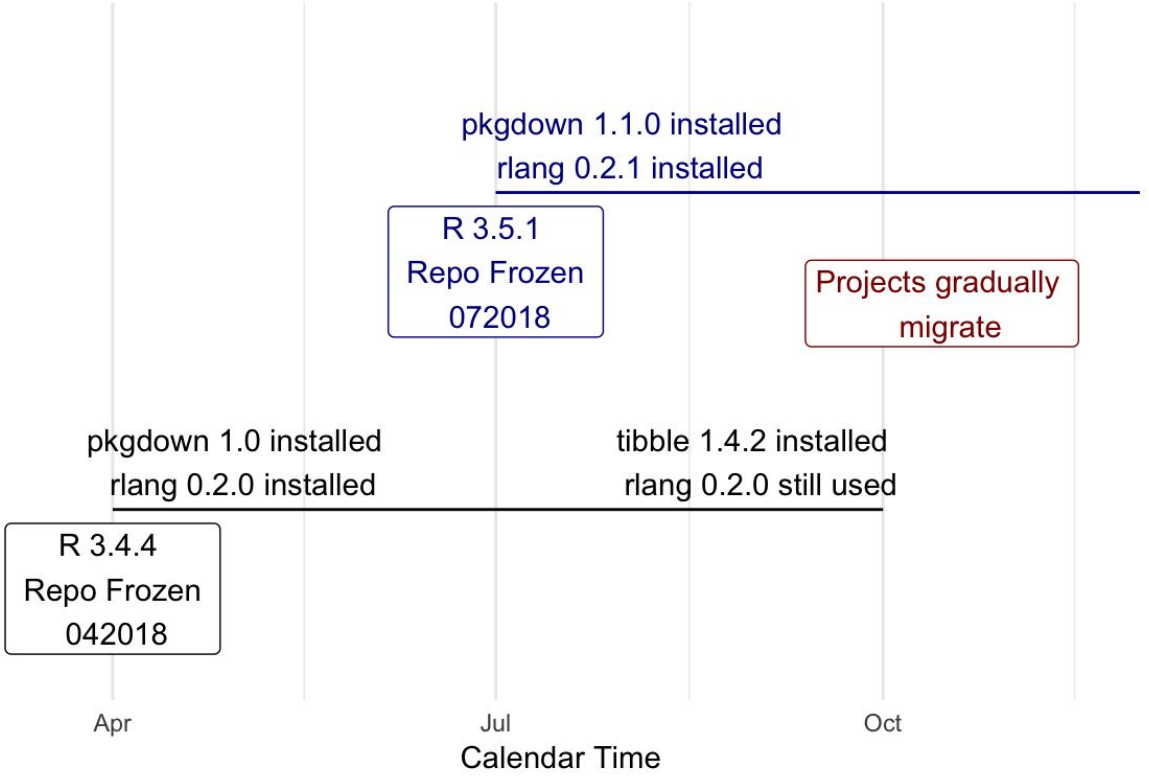


RStudio

Collaborating with a team

Steps:

1. Setup a shared development environment (e.g. RStudio Server)
2. Admins install multiple versions of R
3. Each version of R is tied to a frozen repository



Exercise

Open a session in R 3.5.2 and R 3.4.4

1. In each session, install the tibble package. What are the dependencies and what versions of each are installed? Why are the versions different?

2. What is the result of options(“repos”) in each session?

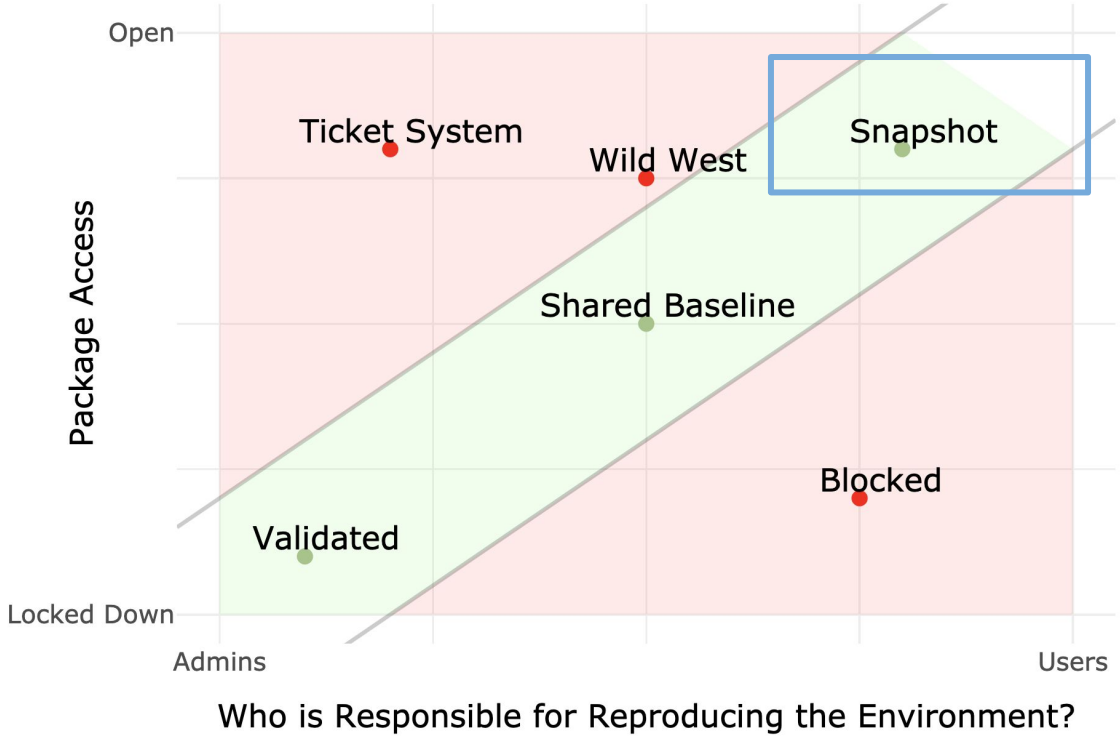
3. Where is the repo option set?

Safely upgrading packages with **Snapshot and Restore** using **renv**

Goal:
Upgrade or add new packages to access exciting new things.

Implied Goals:

- Don't break other things (isolate)
- Safely roll back changes.

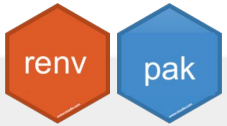


RStudio

Safely upgrading packages

Goal:
Upgrade or add new packages to access exciting new things.

Implied Goals:
- Don't break other things (isolate)
- Safely roll back changes.

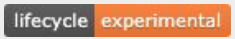



```
# create an isolated per-project library
renv::init()

# snapshot state (and commit lock file)
renv::snapshot()

# optionally upgrade packages
renv::install("ggplot2")

# fall back to older versions
renv::history()
renv::revert("commit123abc")
renv::restore()
```



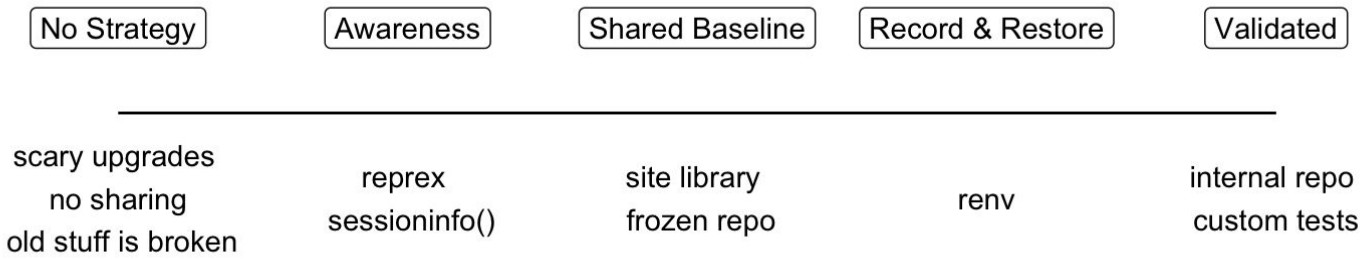
RStudio

Exercise

Create a `renv-test` project

1. Run `renv::init()` - what is added to the project? What is the result of `renv::status()`?
2. Follow README instructions to commit `renv.lock` to git
3. Simulate a package upgrade. Run:
   `remotes::install_github("glue")`
4. What is the new result of `renv::status()`?
5. Run `renv::snapshot()` and `renv::status()`. What happened?
6. Explore `renv::history`, `renv::revert`, and `renv::restore`. Call `renv::status` along the way. Can you roll back your changes?

Recap



Reproducibility isn’t binary. Pick a strategy based on your use cases and implement it with appropriate tools

RStudio®