Interoperability strategy for the Epiverse

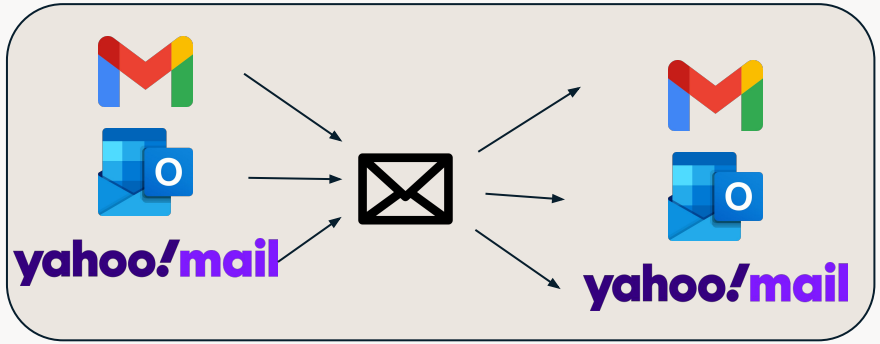
Hugo Gruson
(October 2023)

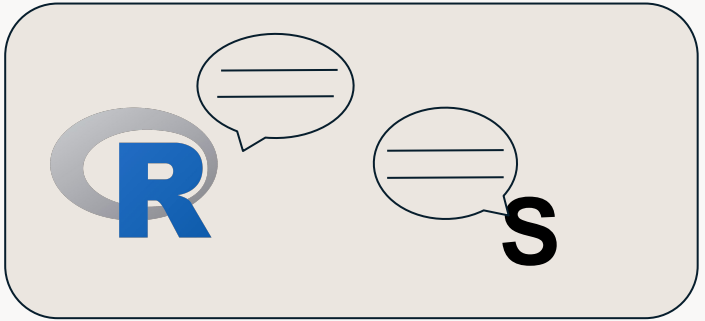
What do we mean by interoperability?

Implicit assumption that interoperability == integration.

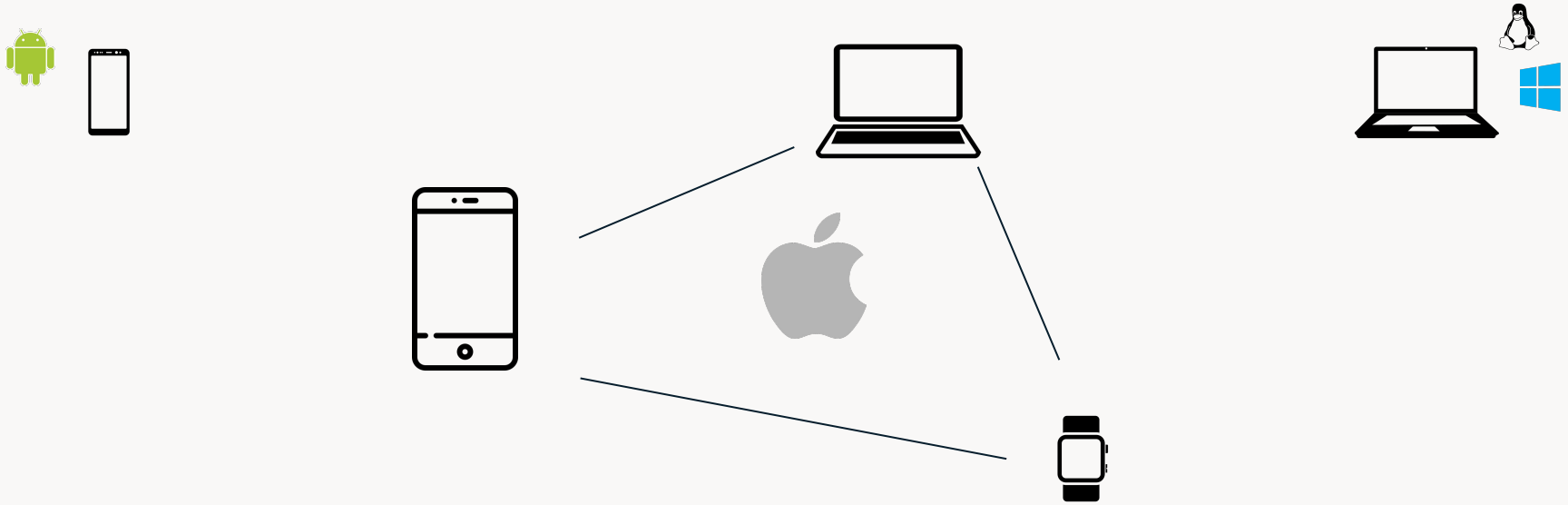
This is a restrictive and risky view of interoperability.

Dependencies are one type of interoperability





Trade-off between integration & interoperability



Dependencies are one **costly** type of interoperability

(especially for emerging ecosystems)

- Time dependence & complex ordering in the release process
- Reverse dependency checking
- Extra development work in the case of (inevitable) breaking changes

Reverse dependency checking for CRAN release

- Loop through your recursive dependencies
- Run R CMD check in each one them
- Submit a PR to fix these errors

“Each of these steps can require considerable work and judgment.

So, if you have no reverse dependencies, you should rejoice that you can skip this step.”

R Packages (2e), Wickham & Bryan

Semantic shift

“Package A **can** be used with package B”

“Package A **must** be used with package B”

Impact on adoption

~~“Delivering the packages as a bundle drives adoption.”~~

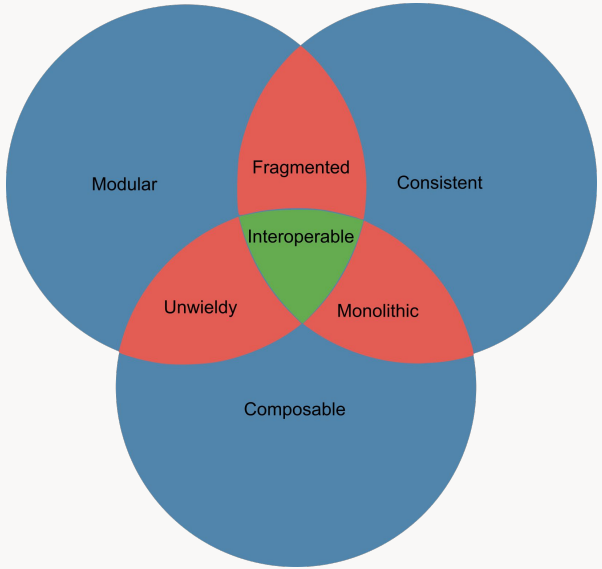
We now have to “sell” 2 packages to users.

We are dealing with busy users who can only afford stepwise changes in their workflows.

3 principles for interoperability strategy in the Epiverse

Interoperability in the Epiverse

- Consistency
- Composability
- Modularity



Interoperability in the Epiverse: Consistency

Shared standards & processes:

- Consistent design philosophy
- Consistent documentation format & organization

The more packages from the Epiverse you use or know, the easier it is to learn a new one.

Interoperability in the Epiverse: Composability

Composability & compatibility guarantee through integration testing

Practical examples:

- Ensure that readepi works well with linelist by ensuring that tags are not dropped while cleaning
- How linelist ensures interoperability with dplyr

Composability can be provided without an explicit strong dependency link

Interoperability in the Epiverse: Modularity

- Packages as small re-usable units
- Plugin system
- Method dispatch for different inputs

- Reduces lock-in and facilitates progressive replacement of various parts

Non-code elements of interoperability

- Connect to community to limit duplication and improve diffusion
- Ways of working

Conclusion

Interoperability is supported by:

- Consistency
- Composability
- Modularity

And is a necessary condition for:

- Adoption
- Sustainability
- Scalability