

Building Interoperability in Existing Software Ecosystems with S3 Classes

Hugo Gruson, Lead Software Architect @ data.org

Why do we care about interoperability?



Piping system on a chemical tanker, by Hervé Cozanet, CC BY-SA

Why do we care about interoperability?



Piping system on a chemical tanker, by Hervé Cozanet, CC BY-SA

Precious time wasted reformatting inputs and outputs.

Very costly in times of emergency.

Why do we care about interoperability?



Piping system on a chemical tanker, by Hervé Cozanet, CC BY-SA

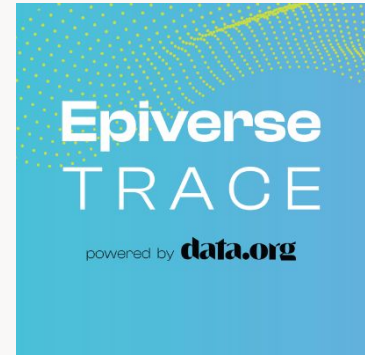
Precious time wasted reformatting inputs and outputs.

Very costly in times of emergency.

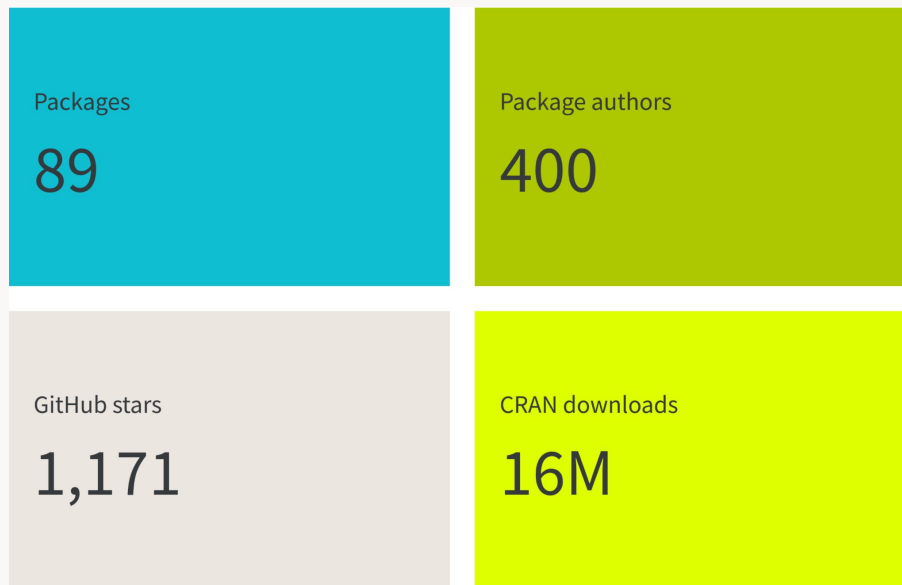
Efforts invested now in interoperability will pay important dividends later down the line.

An international multi-stakeholder project to
harmonise the ecosystem of epidemiology tools in R

- ✓ Make existing tools interoperable
- ✓ Support existing tools to adopt global standards
- ✓ Develop a sustainable community

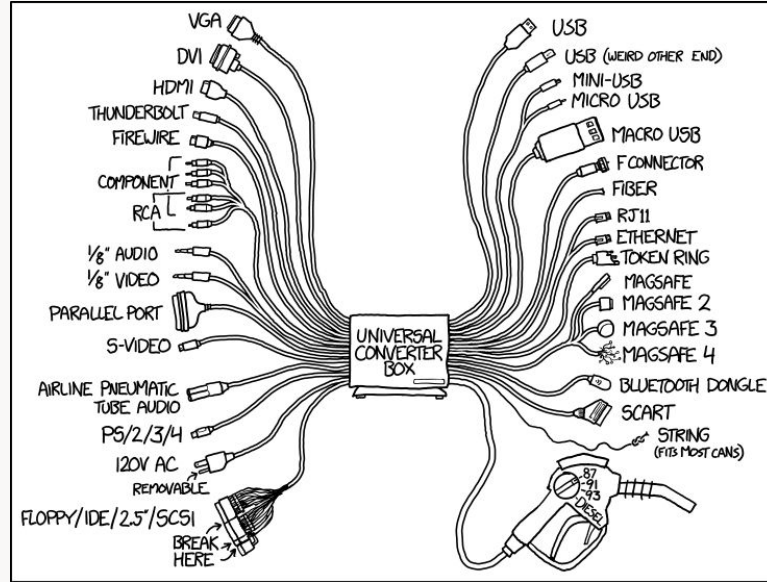


But we also care about preserving the ecosystem!



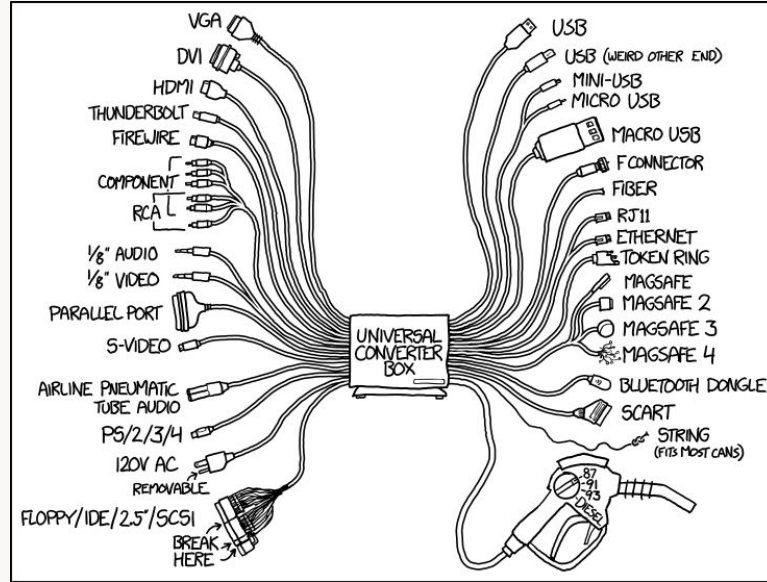
More about this in my [“CRAN Task View Analysis”](#) poster tomorrow

Conversion functions do not scale



Related article: <https://voltrondata.com/codex/open-standards>

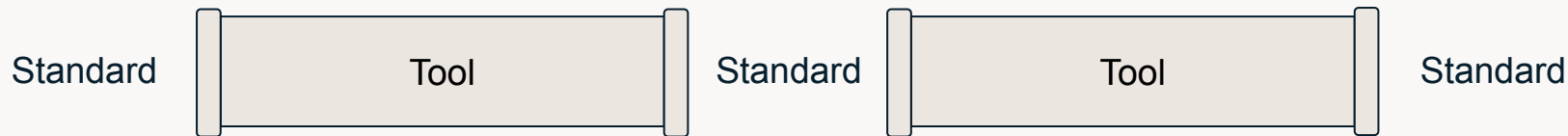
Conversion functions do not scale



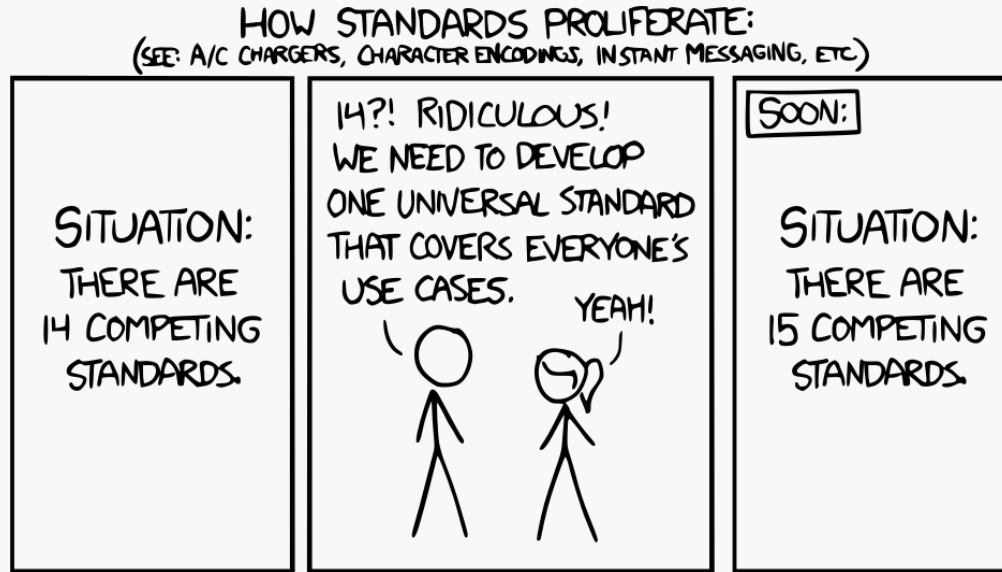
**Standards are the
only viable solution**

Related article: <https://voltrondata.com/codex/open-standards>

How to fix both ends of the pipe?

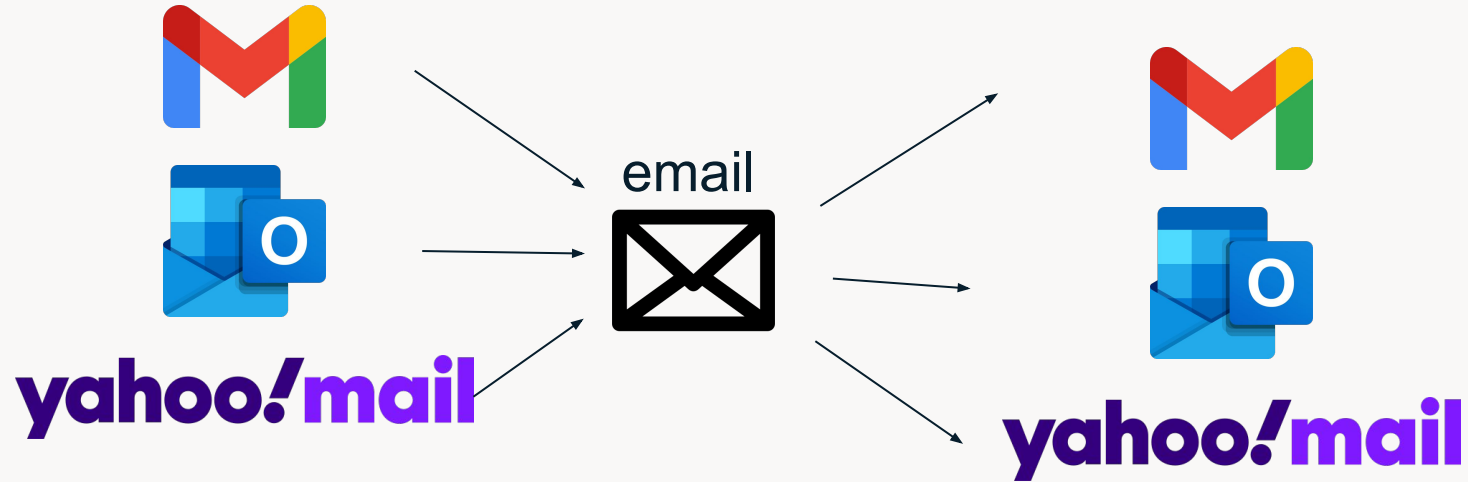


Standards are hard...



XKCD 927: Standards, by Randall Munroe, CC BY-NC

Standards are hard... but not impossible!



How do we go about this in Epiverse-TRACE?

How do we go about this in Epiverse-TRACE?

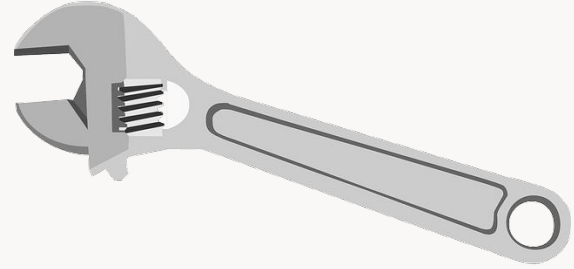


1. Community engagement

How do we go about this in Epiverse-TRACE?



1. Community engagement



2. Technical strategy

Technical strategy: S3 classes and methods

What is S3?

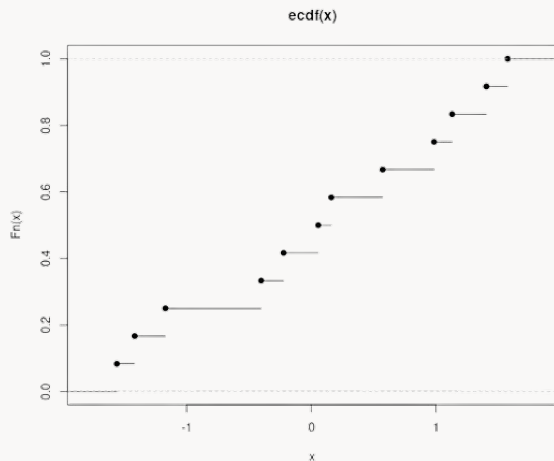
“S3 is informal and ad hoc, but there is a certain elegance in its minimalism: you can’t take away any part of it and still have a useful OO system.” Hadley Wickham

- Mostly an advanced dispatch system based on the presence of the specific attribute
- Used by all R users, even if they don’t realize it!

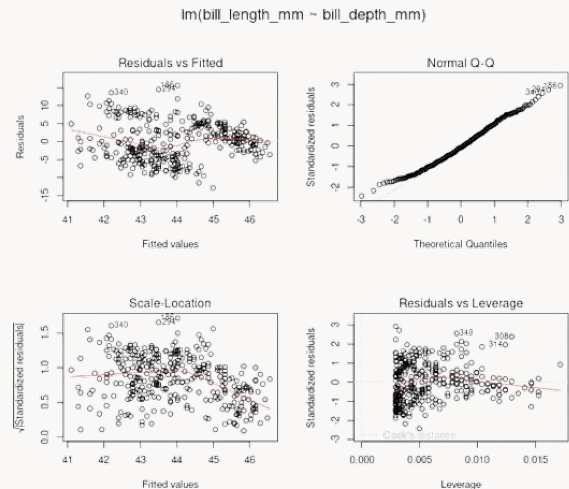
Technical strategy: S3 classes and methods



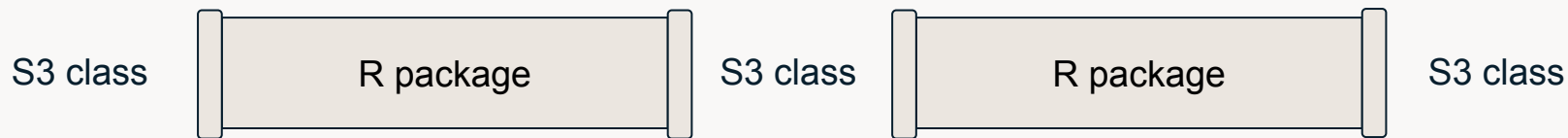
```
fn <- ecdf(rnorm(12))  
plot(fn)
```



```
m <- lm(bill_length_mm ~ bill_depth_mm, data = penguins)  
plot(m)
```



How to fix both ends of the pipe the R way?



Step 1: Develop standards.

More details at <https://epiverse-trace.github.io/posts/parent-class>

Step 1: Develop standards.

What makes a good S3 class for interoperability?

More details at <https://epiverse-trace.github.io/posts/parent-class>

Step 1: Develop standards.

What makes a good S3 class for interoperability?

- Similar to what the ecosystem is using

More details at <https://epiverse-trace.github.io/posts/parent-class>

Step 1: Develop standards.

What makes a good S3 class for interoperability?

- Similar to what the ecosystem is using
- Can work with the popular external ecosystem (e.g., tidyverse)

More details at <https://epiverse-trace.github.io/posts/parent-class>

Step 1: Develop standards.

What makes a good S3 class for interoperability?

- Similar to what the ecosystem is using
- Can work with the popular external ecosystem (e.g., tidyverse)
- Feels familiar to users

More details at <https://epiverse-trace.github.io/posts/parent-class>

Step 1: Develop standards.

What makes a good S3 class for interoperability?

- Similar to what the ecosystem is using
- Can work with the popular external ecosystem (e.g., tidyverse)
- Feels familiar to users

➡ Inherit from `data.frame` where possible

More details at <https://epiverse-trace.github.io/posts/parent-class>

Step 1: Develop standards.

Go the extra mile to provide support for the tidyverse:

- Support for tibbles as well as data.frames
- Support for dplyr verbs

More details at <https://hugogrison.fr/posts/compa-tibble/> & <https://epiverse-trace.github.io/posts/extend-dataframes/>

Step 1: Develop standards.

Support for tibbles as well as data.frames:

- Do not rely on implicit `drop` value (`df[i, j, drop = TRUE]`)
- Do not rely on partial matching (`df$c` instead of `df$col`)

More details at <https://hugogrison.fr/posts/compa-tibble/>

Step 1: Develop standards.

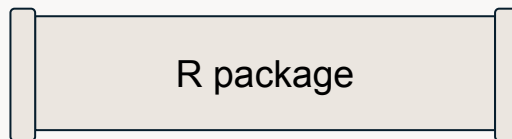
Support for dplyr verbs:

- Methods for `names()<-`, `[<-`, will provide automatic support for most dplyr verbs
- If full compatibility is required, you need extra methods for `dplyr_row_slice()`, `dplyr_col_modify()`, `dplyr_reconstruct()`

More details at <https://epiverse-trace.github.io/posts/extend-dataframes/>

How to fix both ends of the pipe the R way?

S3 class



How to fix both ends of the pipe without disrupting users?

Transposition of “**progressive enhancement**” web development concept.

Two key ideas:

- Adding a new method is invisible
- Adding a new attribute is invisible

More details at <https://epiverse-trace.github.io/posts/progressive-enhancement/>

Step 2: Make functions interoperable with standard S3 inputs

How to add S3 support “invisibly”,
without breaking changes?



```
#' @export
centroid <- function(coords, weights) {

  # ...

}
```

More details & caveats at

<https://epiverse-trace.github.io/posts/s3-generic/>

Step 2: Make functions interoperable with standard S3 inputs

How to add S3 support “invisibly”,
without breaking changes?

```
#' @export
centroid <- function(coords, weights) {

  # ...

}
```



```
#' @export
centroid <- function(coords, weights) {

  UseMethod("centroid")

}

#' @rdname centroid
#'
#' @export
centroid.default <- function(coords, weights) {

  # ...

}
```

More details & caveats at

<https://epiverse-trace.github.io/posts/s3-generic/>

Step 2: Make functions interoperable with standard S3 inputs

How to add S3 support “invisibly”,
without breaking changes?

```
#' @export
centroid <- function(coords, weights) {

  # ...

}
```



```
#' @export
centroid <- function(coords, weights) {

  UseMethod("centroid")

}

#' @rdname centroid
#'
#' @export
centroid.default <- function(coords, weights) {

  # ...

}

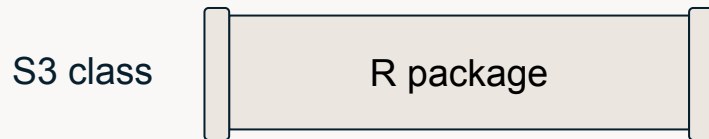
#' @rdname centroid
#'
#' @export
centroid.pointset <- function(coords, weights = NULL) {

  centroid(coords$coords, coords$weights)

}
```

More details & caveats at
<https://epiverse-trace.github.io/posts/s3-generic/>

How to fix both ends of the pipe the R way?



Step 3: Return newly classed input without breaking changes

- If already returning parent from standard, update to return standard
- If not possible to update to standard, return a classed output to allow custom dispatch or conversion functions



```
class(x) <- c("data.frame")  
return(x)
```



```
class(x) <- c("subclass", "data.frame")  
return(x)
```

More details at <https://epiverse-trace.github.io/posts/progressive-enhancement/>

Step 3: Return newly classed input without breaking changes



```
class(x) <- c("data.frame")  
return(x)
```

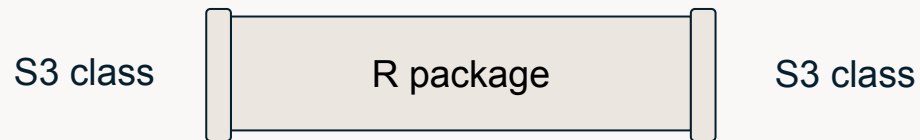


```
class(x) <- c("subclass", "data.frame")  
return(x)
```

This is why class inheritance should never be tested with `==`!

More details at <https://developer.r-project.org/Blog/public/2019/11/09/when-you-think-class.-think-again/index.html>

How to fix both ends of the pipe the R way?



Conclusion

Three steps to add interoperability in an existing ecosystem:

1. Develop standards inheriting from well-established classes
(e.g., `data.frame`)
2. Add support for these standards in function inputs by adding new methods
3. Add support for these standards in function outputs

Conclusion

This is not necessarily the ideal way to design and implement S3 support in general.

This approach is specifically thought to add S3 support in an existing ecosystem with minimal disruption.

Thanks to collaborators and for your attention!

