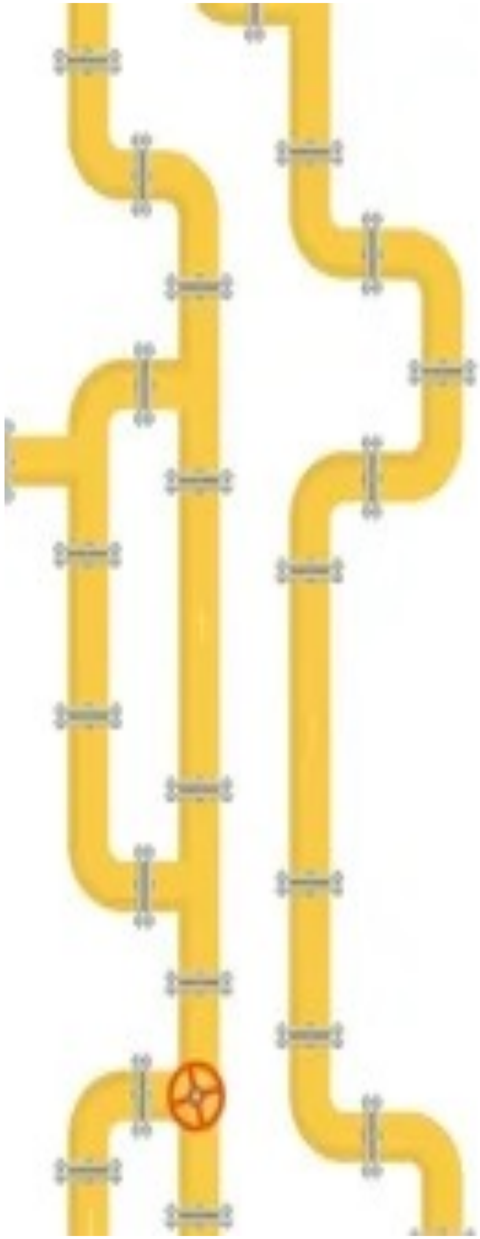


Epiverse pipeline applications: challenges and lessons learned

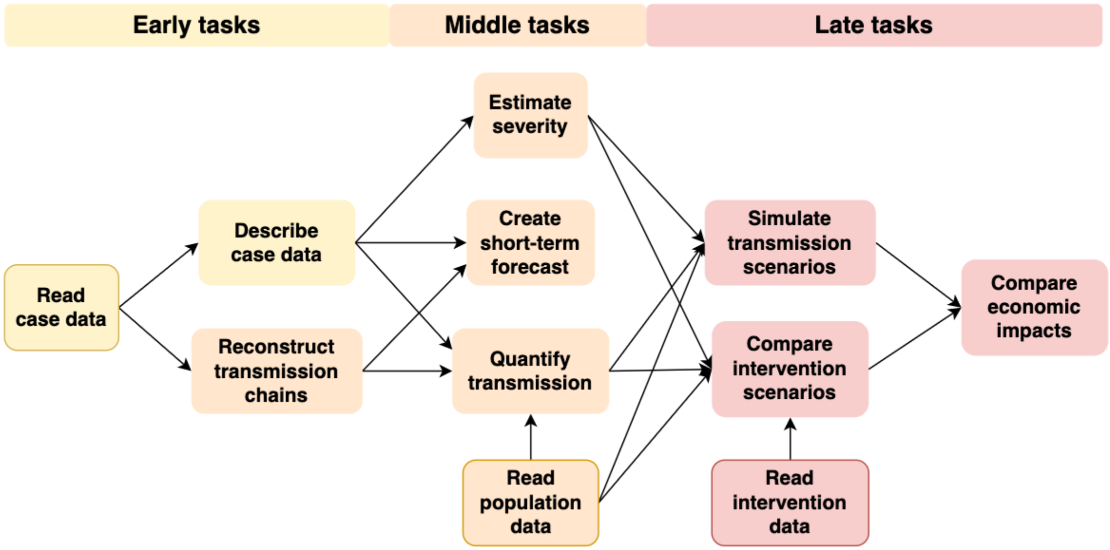
Carmen Tamayo, LSHTM

Epiverse
powered by data.org

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Epiverse pipelines

- Aim: to contain the steps to conduct outbreak analytic tasks from start to finish in a reliable manner
- Divided in early, middle, and late tasks
- Collected as R markdown templates in the `` `episoap` `` package



Early tasks

Middle tasks

Late tasks

Read case data

Describe case data

Reconstruct transmission chains

Estimate severity

Create short-term forecast

Quantify transmission

Read population data

Simulate transmission scenarios

Compare intervention scenarios

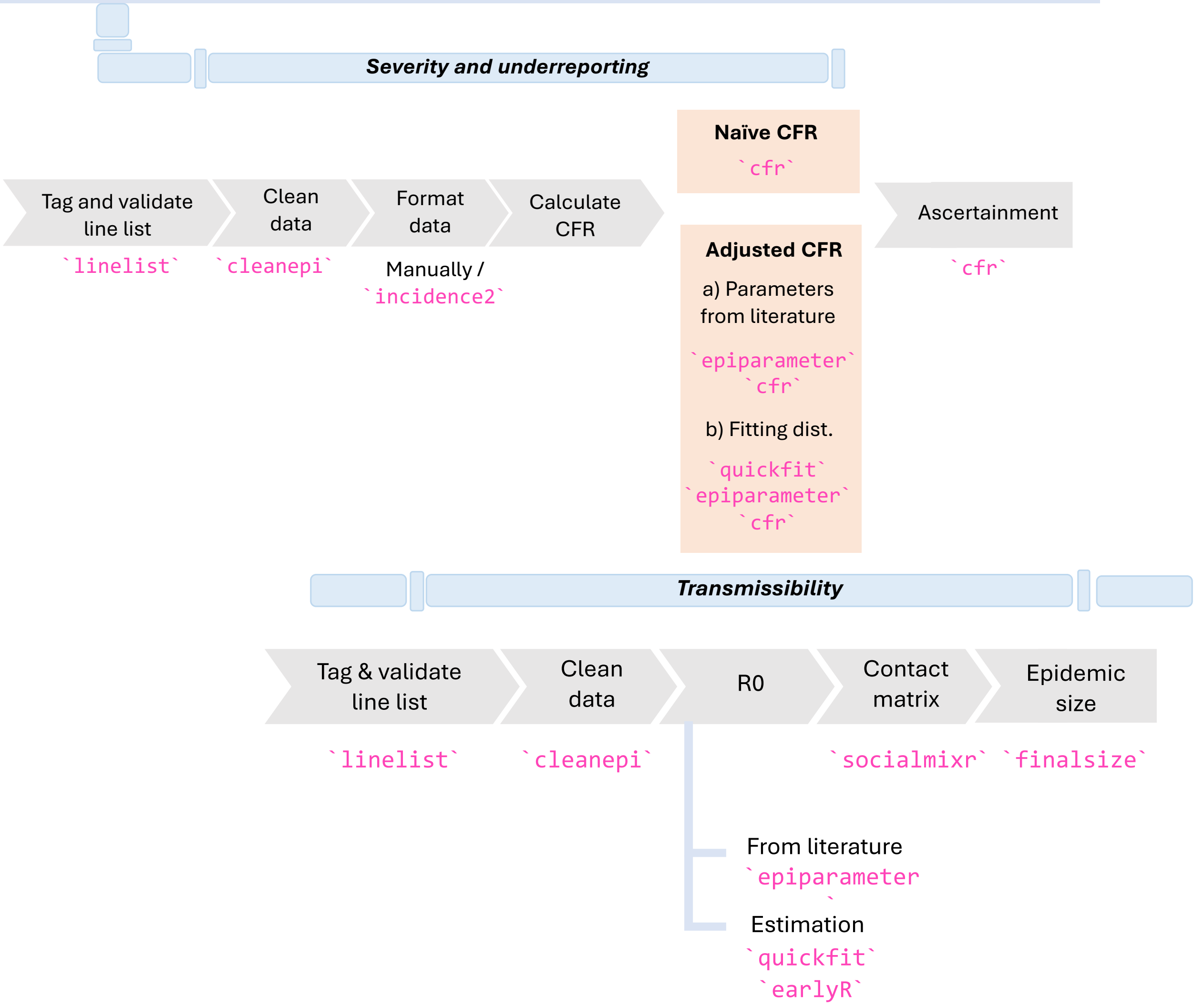
Read intervention data

Compare economic impacts

Epiverse pipelines roadmap, Andree Valle

Pipeline applications/ Case studies

- Aim: to showcase the functionality of Epiverse's pipelines to potential users
- R packages applied to real outbreak data
- Tailored to specific end-user needs and interests
  - ➢ MVD-Severity and underreporting
  - ➢ Cholera-Transmissibility



Severity and underreporting

Tag and validate line list – `linelist`

Clean data – `cleanepi`

Format data – Manually / `incidence2`

Calculate CFR

Naïve CFR
`cfr`

Adjusted CFR
a) Parameters from literature
`epiparameter`
`cfr`
b) Fitting dist.
`quickfit`
`epiparameter`
`cfr`

Ascertainment – `cfr`

Transmissibility

Tag & validate line list – `linelist`

Clean data – `cleanepi`

R0
- From literature
  `epiparameter`
- Estimation
  `quickfit`
  `earlyR`

Contact matrix – `socialmixr`

Epidemic size – `finalsize`

Challenges - Pipeline applications

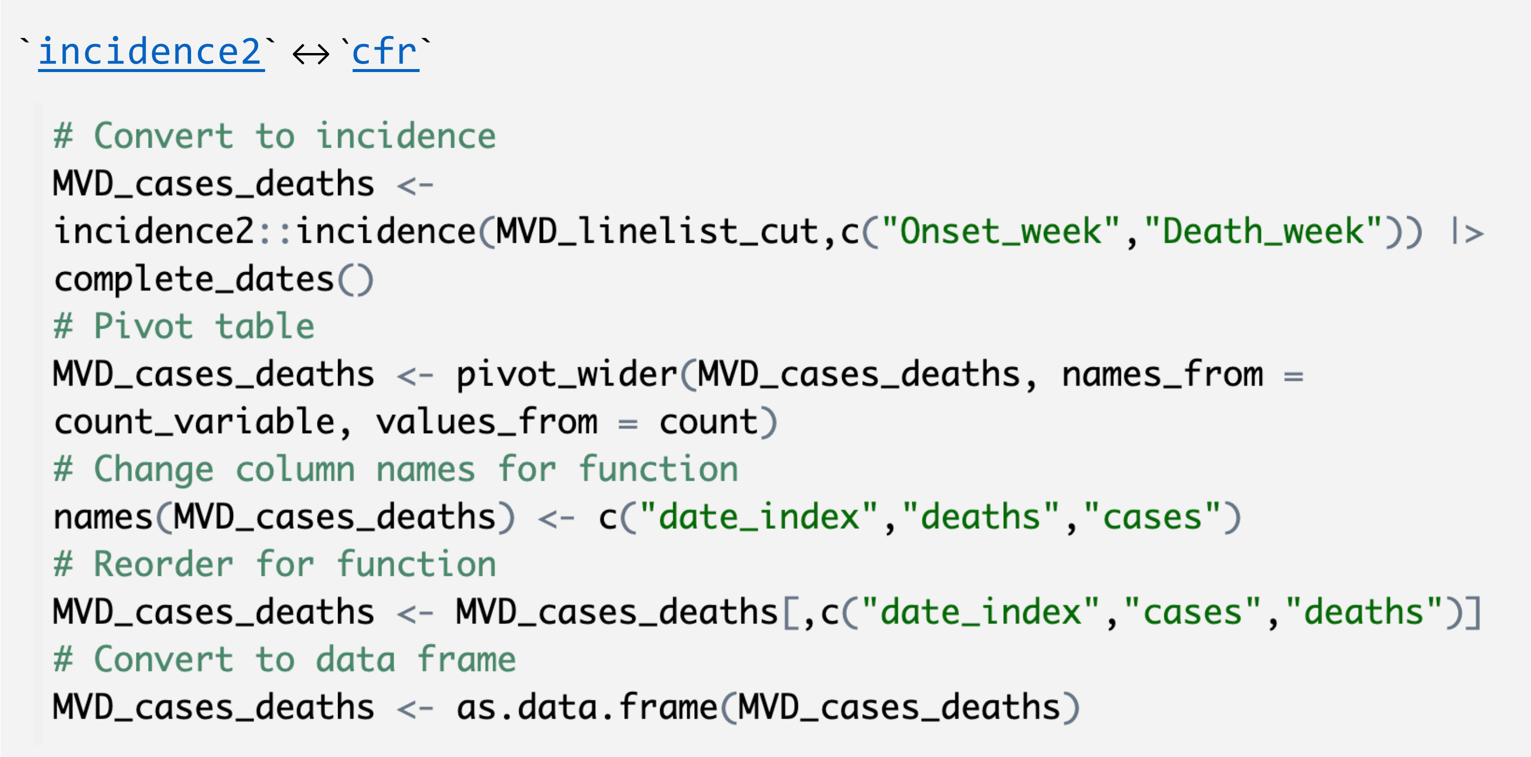
1. Interoperability
   a. Incompatible packages

`linelist` ↔ `dplyr`

↳ *Linelist* objects not usable with functions such as `mutate()` or `filter()`

Challenges - Pipeline applications

1. Interoperability
   a. Incompatible packages
   b. Output vs input format

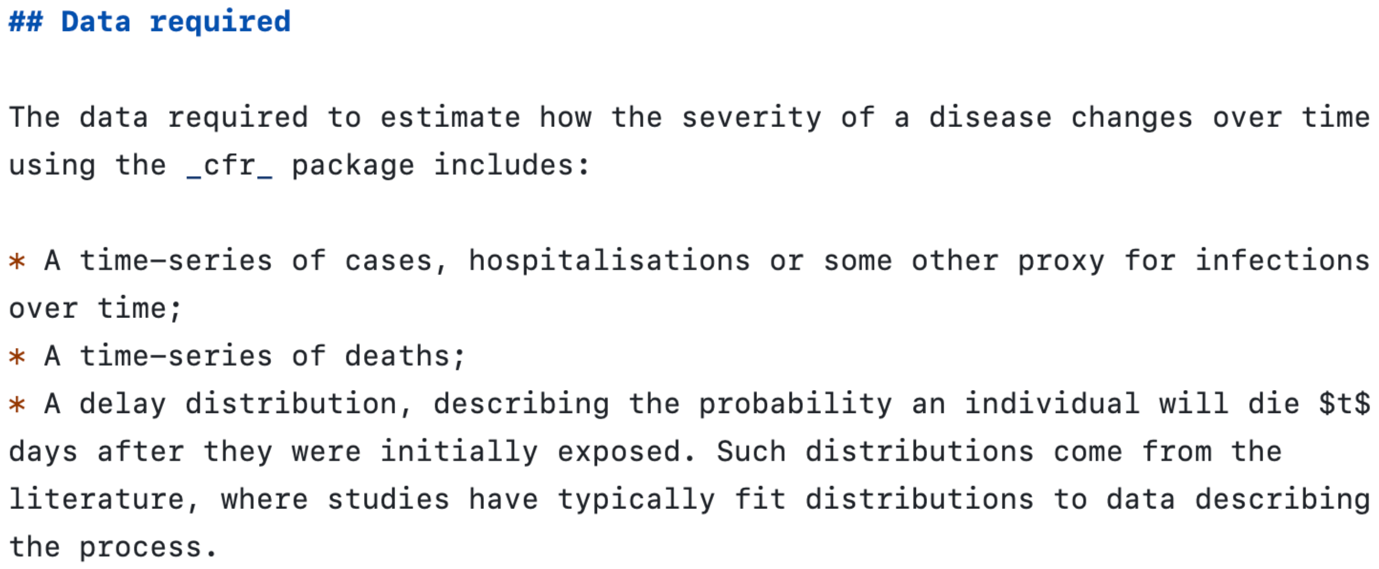


`incidence2` ↔ `cfr`

```r
# Convert to incidence
MVD_cases_deaths <-
incidence2::incidence(MVD_linelist_cut,c("Onset_week","Death_week")) |>
complete_dates()
# Pivot table
MVD_cases_deaths <- pivot_wider(MVD_cases_deaths, names_from =
count_variable, values_from = count)
# Change column names for function
names(MVD_cases_deaths) <- c("date_index","deaths","cases")
# Reorder for function
MVD_cases_deaths <- MVD_cases_deaths[,c("date_index","cases","deaths")]
# Convert to data frame
MVD_cases_deaths <- as.data.frame(MVD_cases_deaths)
```

Challenges - Pipeline applications

1. Interoperability
   a. Incompatible packages
   b. Output vs input format
2. Lack of user friendliness
   a. Documentation



```
## Data required

The data required to estimate how the severity of a disease changes over time
using the _cfr_ package includes:

* A time-series of cases, hospitalisations or some other proxy for infections
over time;
* A time-series of deaths;
* A delay distribution, describing the probability an individual will die $t$
days after they were initially exposed. Such distributions come from the
literature, where studies have typically fit distributions to data describing
the process.
```

Challenges - Pipeline applications

1. Interoperability
    a. Incompatible packages
    b. Output vs input format
2. Lack of user friendliness
    a. Documentation
    b. Non-informative error messages

Previous error message of `estimate_static()` function from `cfr`:

```
Error in data.frame(severity_me = severity_me, severity_lo =
severity_lims[[1]], : arguments imply differing number of rows: 0, 1
```

Challenges - Pipeline applications

1. Interoperability
   a. Incompatible packages
   b. Output vs input format
2. Lack of user friendliness
   a. Insufficient documentation
   b. Non-informative error messages

Previous error message of `estimate_static()` function from `cfr`:

```
Error in data.frame(severity_me = severity_me, severity_lo =
severity_lims[[1]], : arguments imply differing number of rows: 0, 1
```

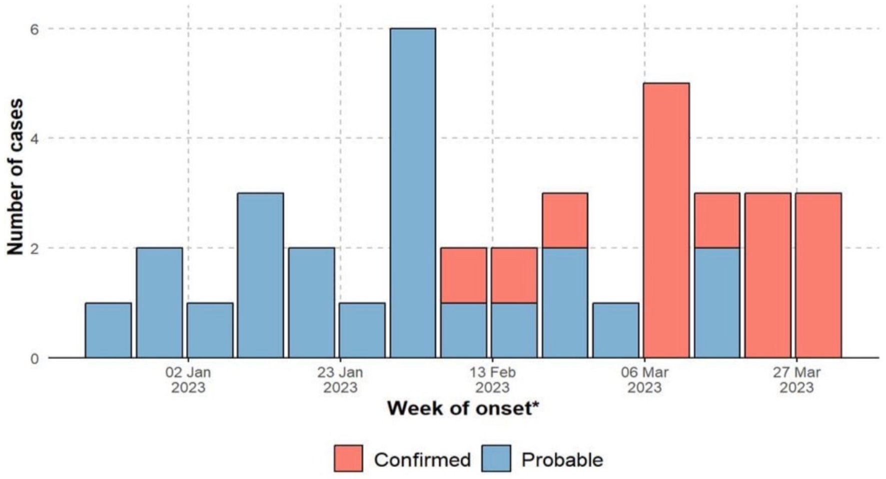
Current error message of `estimate_static()` function from `cfr`:

```
Error in estimate_static(daily_cases_deaths_missing_data, epi_dist =
onset_to_death_ebola, : Input data must have sequential dates with
none missing or duplicated
```

Challenges - Pipeline applications

1. Interoperability
    a. Incompatible packages
    b. Output vs input format
2. Lack of user friendliness
    a. Insufficient documentation
    b. Non-informative error messages
3. Sometimes don’t translate to certain real-life scenarios

`cfr` currently uses *days* as input for dates, whereas some data sources provide weeks of onset/death



Challenges - Pipeline applications

1. Interoperability
   a. Incompatible packages
   b. Output vs input format
2. Lack of user friendliness
   a. Insufficient documentation
   b. Non-informative error messages
3. Sometimes don’t translate to certain real-life scenarios
4. Packages under development

Unstable functions, features that are removed, names changed, etc. → difficult for users to keep track

```
Error in format_output(estimate_static(df_ebola_subset,
correct_for_delays = TRUE,  :
  could not find function "format_output"
```

Challenges - Pipeline applications

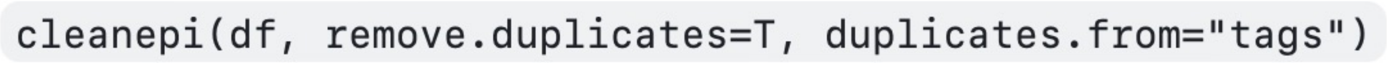
1. Interoperability
   a. Incompatible packages
   b. Output vs input format
2. Lack of user friendliness
   a. Insufficient documentation
   b. Non-informative error messages
3. Sometimes don’t translate to certain real-life scenarios
4. Packages under development

Unstable functions, features that are removed, names changed, etc. → difficult for users to keep track

```
Error in delay_opts(list(mean = onset_to_death_logmean, mean_sd = 0.1,  :
  Delay distributions must be of given either using a call to `dist_spec` or one of the
`get_...` functions such as `get_incubation_period`. This behaviour has changed from
previous versions of `EpiNow2` and any code using it may need to be updated. For
examples and more information, see the relevant documentation pages using
`?delay_opts`.
```

Lessons learned - Pipeline applications

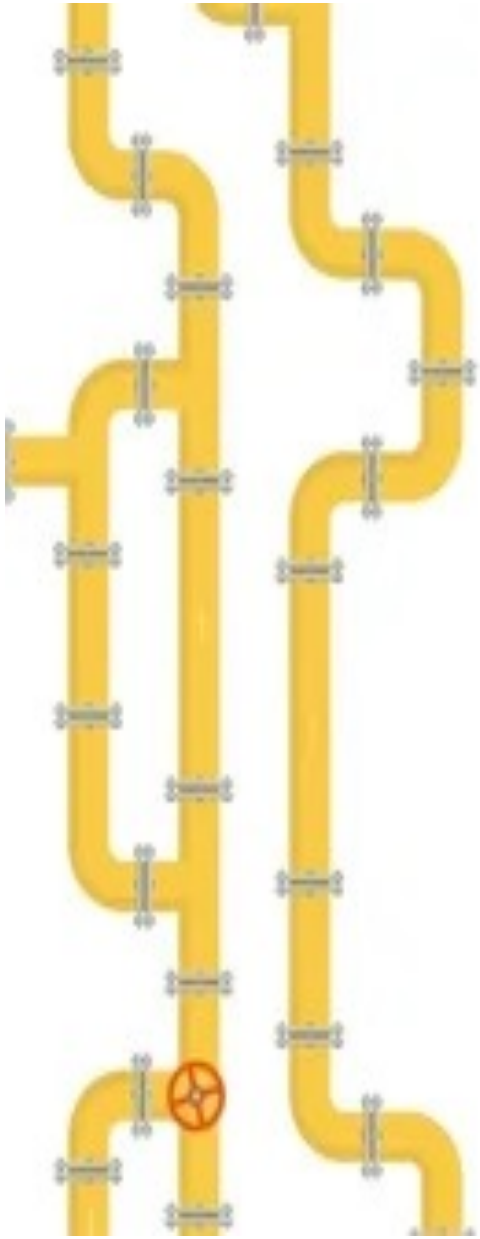
- Testing data pipelines is as relevant as testing the functionality of packages individually
- Ideally there would be at least one person within the team to carry out the testing
- Challenges are also an opportunity to optimise the pipelines
  - E.g.: `cleanepi` to remove duplicated data across `linelist` tags



```
cleanepi(df, remove.duplicates=T, duplicates.from="tags")
```

Lessons learned - Pipeline applications

- Testing data pipelines is as relevant as testing the functionality of packages individually
- Ideally there would be at least one person within the team to carry out the testing
- Challenges are also an opportunity to optimise the pipelines
- ... and an opportunity for RSEs and RFs for collaborative development
- In the future, this process must be carried out also by users outside the team



Thank you for your attention!

Carmen.Tamayo-Cuartero@lshtm.ac.uk

Epiverse Blog: https://epiverse-trace.github.io/blog.html

Epiverse repo: https://github.com/epiverse-trace