Practical Strategies for R-Based Research in Secure Data Environments

Aleksi Lahtinen[1], Leo Lahti[1]

[1]Department of Computing
University of Turku, Finland

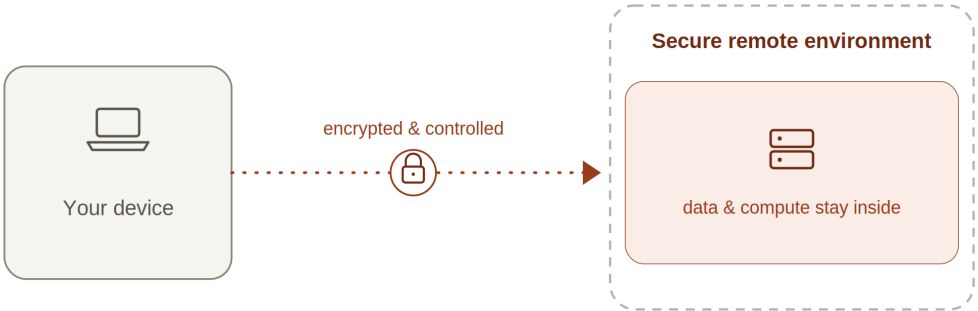
useR! Conference, July 2026, Warsaw, Poland

Motivation

- Sensitive data requires secure computing environments
- Security restrictions create practical challenges for researchers
- R can help overcome many of these challenges

What are secure remote access systems?

- Controlled environments for working with sensitive data remotely
- Commonly used for administrative, health, and other confidential data
- Strict controls govern access, software, and data exports

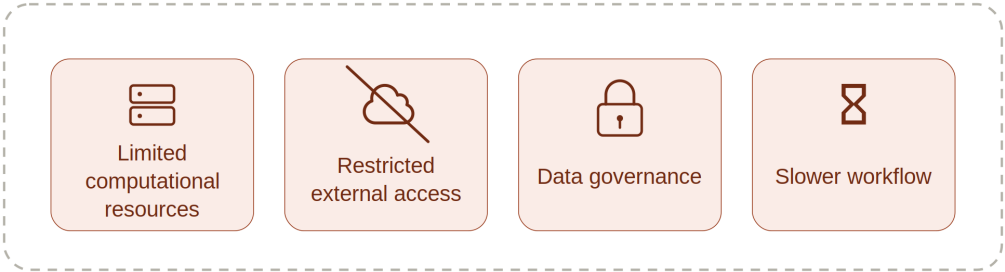


Out of Despair project

- Population-based study using large-scale administrative register data
- Analysis conducted in the FIONA secure remote access environment
- Presentation will have examples relating to the project

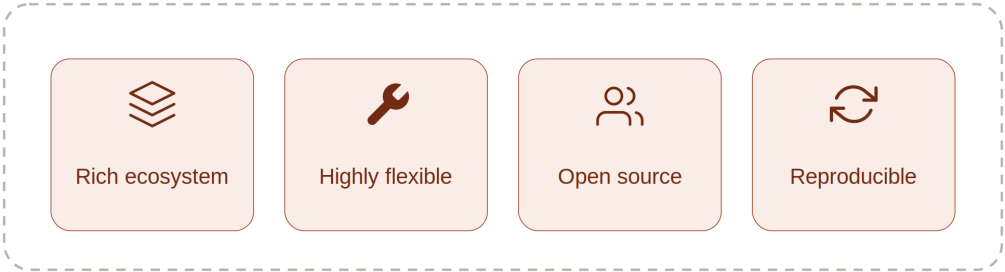
Why secure environments change R workflows

Several factors reshape how R is used day-to-day in these settings:



Why R works well in secure environments

R's design lines up well with what these environments demand:



Efficient data handling for large datasets

- Large register datasets may not fit fully into memory
- Focus on early filtering and selecting only necessary variables
- Use efficient formats and tools (`arrow`, `fst`, `data.table`)

Example of a register

- Register of doctor visits
- 3 653 451 rows and 116 columns
- Many of the columns not relevant

id	visit start	visit end	icd10 code	
1	12-04-2018 12:00:00	12-04-2018 12:36:00	Z00.00	$\cdots$
1	02-11-2018 09:30:00	02-11-2018 10:03:00	E11.9	$\cdots$
2	21-04-2016 15:00:00	21-04-2016 15:24:00	I10	$\cdots$
3	17-01-2014 13:00:00	17-01-2014 13:16:00	J45.909	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

arrow example

Filtering a dataset before loading the result into memory:

```
dt <- open_dataset("data/panel_dataset.parquet") %>%
    select(id, visit_start, visit_end, icd10) %>%
    filter(grepl("^F", icd10)) %>%
    collect()
```

data.table example

Comparison for creating a new variable in dataset with 3 million rows:

Tidyverse

```
panel <- panel %>%
  mutate(.row = row_number()) %>%
  left_join(
    data %>% select(id = time),
    by = join_by(id, month >= time)
  ) %>%
  summarise(
    times = sum(!is.na(time)),
    .by = .row
  ) %>%
  right_join(
    panel %>% mutate(.row = row_number()),
    by = ".row"
  ) %>%
  select(-.row)
```

Time: 44 s

data.table

```
panel[
 , attempts :=
   records[
     .SD,
     on = .(id, time <= month),
     .N,
     by = .EACHI
   ]$N
]
```

Time: 20 s

Efficient computation in constrained systems

- Shared compute environments limit scalability
- Inefficient code quickly becomes a bottleneck
- Use parallelisation when appropriate (`future`, `furrr`, `future.apply`)

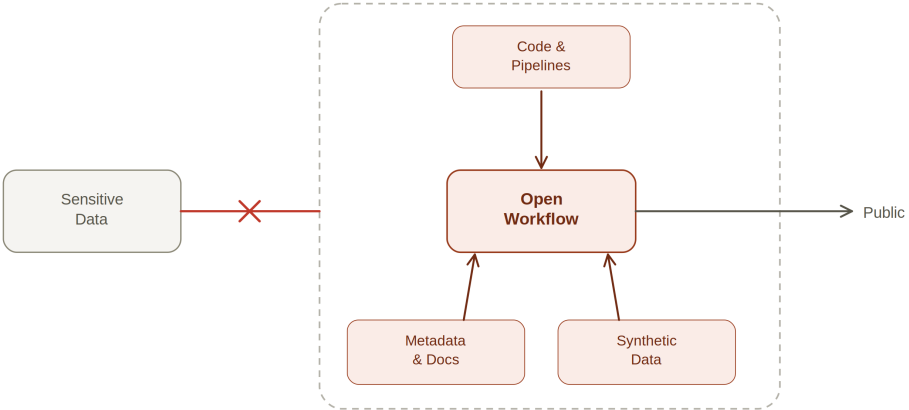
future example

Using parallelisation when fitting nested cross-validation using `mlr3`:

```
future::plan("multisession", workers = 5)
rr <- resample(
  tsk,
  at,
  rsmp_cv
  )
future::plan("sequential")
```

Open science without open data

- Sensitive data cannot be shared publicly
- Transparency shifts from data $\rightarrow$ workflow



AI and advanced computation in secure environments

- Modern AI and computational tools offer new opportunities
- Secure environments impose practical limits on their use
- Service providers should offer secure, local alternatives to external tools

Conclusion

- Secure data environments introduce inconveniences
- Optimise code and analyses (e.g. `data.table`, `arrow`, parallelisation)
- Publish code along with detailed workflows and documentation.

Thank you!

Questions? Suggestions?

email: allaht@utu.fi