Streamlining Data Access in Clinical Research: Introducing the {connector} R Package

Cervan Girard and Aksel Thomsen

17 Apr 2025

What about this presentation

All presenters are employees of Novo Nordisk A/S.
Views and opinions expressed are those of the presenters and not necessarily Novo Nordisk A/S.
Neither Novo Nordisk A/S nor the presenters have any financial interests in any of the software mentioned during the presentation.

Introduction to {connector}

- R package for **unified interface** with **various data sources**
- Simplifies **connection management** and **data operations**
- **Flexible** and **extensible** for different types of data sources
- Giving back to the **Open-Source** community

Why use {connector}?

- Modern data analysis often involves **multiple data sources**
  - Databricks, SharePoint, local fileshare, etc…
- Each source may have its own **connection method** and **API**
- Maintaining different connection protocols in each script can be cumbersome
- Switching between data sources often requires code rewriting

Life without {connector}

Manipulate files and databases:

```r
library(DBI)

# File access - hardcoded paths
adam_path <- "/path/study/adam"

# PostgreSQL connection - parameters everywhere
con <- dbConnect(RPostgres::Postgres(),
  host = "db.server",
  port = 5432,
  dbname = "clinical_db",
  user = "user",
  password = "test"
)

# Read tables
adsl <- read.csv(file.path(adam_path, "adsl.csv"))
subjects <- dbGetQuery(con, "SELECT * FROM subjects")

# Writing results
write.csv(results, file.path(adam_path, "results.csv"))
```

Repeat in your next script:

```r
library(DBI)

# File access - hardcoded paths
adam_path <- "/path/study/adam"

# PostgreSQL connection - parameters everywhere
con <- dbConnect(RPostgres::Postgres(),
  host = "db.server",
  port = 5432,
  dbname = "clinical_db",
  user = "user",
  password = "test"
)

# Read tables
subjects <- dbGetQuery(con, "SELECT * FROM labo")
adae <- read.csv(file.path(adam_path, "adae.csv"))

# Writing results
write.csv(results, file.path(adam_path, "results_adae.csv"))
```

- Copy paste a lot of code
- Not easy to understand what have changed between scripts
- Easy to forget to update everywhere

What to improve with {connector}

- **How can we manage file paths across different team members' computers?**
- **How can we simplify reading/writing files without repeating path logic?**
- **How can we standardize database connections across different projects?**
- **What's the safest way to manage database credentials in a shared codebase?**
- **and more….**

Life with {connector}

The same example

```r
# Load the connector package
library(connector)

# Connect to all datasources in one step
cnts <- connect("_connector.yml")

# Read data files using a consistent interface
adae <- cnts$adam |> read_cnt("adae.csv")
# Query the database using the same connector approach
subjects <- cnts$db |> read_cnt("subjects")

# Write results back to both locations
cnts$adam |> write_cnt(results, "results.csv")
cnts$db |> write_cnt(db_results, "analysis")
```

How it works

Configuration

- Flexible configuration options:
  - YAML files for human-readable setup
  - R lists for programmatic configuration
  - JSON format for more flexibility
- Centralized configuration management
- Easy to version control and share configurations

Standardized Operations

- Consistent **CRUD** (Create, Read, Update, Delete) operations
- Methods include:
  - `list_content_cnt()` : List available content
  - `read_cnt()` : Read data from the source
  - `write_cnt()` : Write data to the source
  - `remove_cnt()` : Delete data from the source

Means whatever datasource you use, you will have the possibility to use those methods!

File System Example

Create configuration file

The simplest way to use it is to make YAML configuration file for ADAM data on your file system:

`_connector.yml`:

```yaml
datasources:
  - name: "adam"
    backend:
      type: connector_fs
      path: "datasets/adam"
```

Connect to data

Create connection to the data using configuration file:

```r
cnts <- connect() # use "_connector.yml"
```

```
──────────────────────────────────────────────────

Connection to:

→ adam

• connector_fs

• datasets/adam
```

```r
print(cnts)
```

```
<connectors>

  $adam <ConnectorFS>
```

Basic Operations

Here are some basic operations you can use with any type of data source:

Write a data.frame:

```
cnts$adam |>
  write_cnt(iris, "iris.csv",
            overwrite = TRUE)
```

List content:

```
cnts$adam |>
  list_content_cnt()
```

```
[1] "adae.csv" "adsl.csv" "iris.csv"
```

Read a file:

```
cnts$adam |>
  read_cnt("adae") |>
  dplyr::select(
    STUDYID, DOMAIN,
    USUBJID, AESEQ)
```

```
# A tibble: 1,191 × 4
   STUDYID      DOMAIN USUBJID     AESEQ
   <chr>        <chr>  <chr>       <dbl>
 1 CDISCPILOT01 AE     01-701-1015     1
 2 CDISCPILOT01 AE     01-701-1015     2
 3 CDISCPILOT01 AE     01-701-1015     3
 4 CDISCPILOT01 AE     01-701-1023     2
 5 CDISCPILOT01 AE     01-701-1023     1
 6 CDISCPILOT01 AE     01-701-1023     4
 7 CDISCPILOT01 AE     01-701-1023     3
 8 CDISCPILOT01 AE     01-701-1028     1
 9 CDISCPILOT01 AE     01-701-1028     2
10 CDISCPILOT01 AE     01-701-1034     1
# ℹ 1,181 more rows
```

A DBI integration as well

Using a database such as postgres SQL :

`_connector_dbi.yml`:

```yaml
datasources:
  - name: "adam"
    backend:
      type: connector_dbi
      drv: "RPostgres::Postgres"
      password: "test"
      host: "localhost"
      user: "postgres"
```

Connect in the same way:

```
db <- connect("_connector_dbi.yml")
```

```
──────────────────────────────────────────────────────────────────────────────

Connection to:

→ adam

• connector_dbi

• RPostgres::Postgres, test, localhost, and postgres
```

Same Basic Operations

Write a data.frame:

```
db$adam |>
  write_cnt(pharmaverseadam::adae, "adae", ove
```

List content:

```
db$adam |>
  list_content_cnt()
```

```
[1] "adae" "iris"
```

Read a file:

```
db$adam |>
  read_cnt("adae") |>
  dplyr::select(
    STUDYID, DOMAIN,
    USUBJID, AESEQ)
```

	STUDYID	DOMAIN	USUBJID	AESEQ
1	CDISCPILOT01	AE	01-701-1015	1
2	CDISCPILOT01	AE	01-701-1015	2
3	CDISCPILOT01	AE	01-701-1015	3
4	CDISCPILOT01	AE	01-701-1023	2
5	CDISCPILOT01	AE	01-701-1023	1
6	CDISCPILOT01	AE	01-701-1023	4
7	CDISCPILOT01	AE	01-701-1023	3
8	CDISCPILOT01	AE	01-701-1028	1
9	CDISCPILOT01	AE	01-701-1028	2
10	CDISCPILOT01	AE	01-701-1034	1
11	CDISCPILOT01	AE	01-701-1034	2
12	CDISCPILOT01	AE	01-701-1047	1
13	CDISCPILOT01	AE	01-701-1047	2
14	CDISCPILOT01	AE	01-701-1047	3
15	CDISCPILOT01	AE	01-701-1047	4

{connector} in a trial

How to use it in a trial

- Used postgres for ADaM dataset
- Used folders for Outputs

```yaml
datasources:
  - name: "adam"
    backend:
      type: connector_dbi
      drv: "RPostgres::Postgres"
      password: "test"
      host: "localhost"
      user: "postgres"
  - name: "outputs"
    backend:
      type: connector_fs
      path: "datasets/outputs"

```

How to use it in a trial

```r
cnts <- connect("_connector_both.yml")
```

```
──────────────────────────────────────────────────────────────────────────────

Connection to:

→ adam

• connector_dbi

• RPostgres::Postgres, test, localhost, and postgres

──────────────────────────────────────────────────────────────────────────────

Connection to:

→ outputs

• connector_fs

• datasets/outputs
```

How to use it in a trial

```
cnts$adam |>
  write_cnt(pharmaverseadam::adae, "adae", overwrite = TRUE)

cnts$adam |>
  list_content_cnt()
```

```
[1] "adae" "iris"
```

How to use it in a trial

```r
table_output <- cnts$adam |>
  read_cnt("adae") |>
  dplyr::group_by(ACTARM) |>
  dplyr::summarise(
    N = dplyr::n_distinct(USUBJID),
    n = dplyr::n()
  )

print(table_output)
```

```
# A tibble: 3 × 3
  ACTARM                 N     n
  <chr>              <int> <int>
1 Placebo               69   301
2 Xanomeline High Dose  70   436
3 Xanomeline Low Dose   86   454
```

How to use it in a trial

```
cnts$outputs |>
  write_cnt(table_output, "mytable.csv")

cnts$outputs |>
  write_cnt(table_output, "mytable.xlsx")
```

How to use it in a trial

```
cnts$outputs |>
  list_content_cnt()
```

```
[1] "mytable.csv"  "mytable.xlsx"
```

Extra functionalites

Using Metadata to simplify your setup

```yaml
metadata:
  trial_id: "id_01"

datasources:
  - name: "adam"
    backend:
      type: connector_fs
      path: "studies/{metadata.trial_id}/adam"
  - name: "sdtm"
    backend:
      type: connector_fs
      path: "studies/{metadata.trial_id}/sdtm"
```

```r
# trial ID used id_01
trial_01 <- connect()
```

Using Metadata to simplify your setup

- Overwrite metadata to make study setup easy to change:

```
# trial ID used id_02
trial_02 <- connect(
  metadata = list(
    trial_id = "id_02"
  )
)
```

Use Environment Variables

- Enables secure handling of sensitive information by integrating with system environment variables.
- Allows for separation of credentials from code.

`_connector_dbi_env.yml`:

```yaml
datasources:
  - name: "adam"
    backend:
      type: connector_dbi
      drv: "RPostgres::Postgres"
      password: "{env.password}"
      host: "localhost"
      user: "postgres"
```

`.Renviron`:

```
password=test
```

Connect using the environment variable:

```
trial <- connect("_connector_dbi_env.yml")
```

Optional Logging System

- Built-in logging capabilities.
- Will track operations performed on data sources.
- Useful for debugging and for creating audit logs.
- Integrated with {whirl}.

```
1  logs <- connect(logging = TRUE)
```

```
──────────────────────────────────────────────────────────────────────────────

Connection to:

→ adam

• connector_fs

• datasets/adam
```

Log read operation:

```
1  adae <- logs$adam |>
2    read_cnt("adae")
```

```
{"time":"2025-04-24 12:33:43","type":"read","file":"adae @ datasets/adam"}
```

Use in {whirl}

`demo.R`:

```r
library(connector)

logs <- connect(logging = TRUE)

adae <- logs$adam |>
  read_cnt("adae")
```

Run the script to generate log:

```r
whirl::run("demo.R")
```

```
✔ demo.R: Completed succesfully
```

demo_log.html

demo.R

EXECUTION END TIME

2025-04-24T12:26:49 Central European Summer Time

PATH TO SCRIPT

/Users/oath/Projects/pRublications-1/PHUSE/2025-SDE-UTRECHT/demo.R

Summary

success

Input

time	file
2025-04-24 12:26:48	adae @ datasets/adam

Output

Table of contents

Other datasources

{connector.databricks} & {connector.sharepoint}

Example with the Databricks extension

- `connector.databricks` utilizes **Databricks API** (+more) to manipulate resources.
- Access both Databricks **Volumes** and **Tables**.
- Shares almost **identical interface** as for {connector}.
- Tranfering from file system to Databricks:
  - Our scripts don't change
  - Just change **configuration file** to point to Databricks.

```yaml
datasources:
  - name: "output"
    backend:
      type: "connector.databricks::connector_databricks_volume"
      full_path: "mycatalogue/myschema/output"
  - name: "adam"
    backend:
      type: "connector.databricks::connector_databricks_table"
      http_path: "https://databricks/"
      catalog: "mycatalogue"
      schema: "myschema_adam"
```

Conclusion

- {connector} **simplifies data source management**
- Offers a **unified interface for diverse data sources**
- Improves **productivity** and **code maintainability**
- Ideal for projects involving **multiple data sources**
- Allows for **extensibility** (Databricks, SharePoint, BigQuery, etc.)
- Empowers data scientists to focus on analysis, not connection logistics

Useful links

- {connector}: Connect to your data easily
- {connector.databricks}: Connect to Databricks
- {connector.sharepoint}: Connect to SharePoint
- {whirl}: Logging of scripts suitable for clinical trials
- NovoNordisk-OpenSource/R-packages

Thank you for your attention!