Siphiwe Bogatsu and Elisha Banda

24 April 2026

UCTSAD 2006-2025 Yearly Cleaning Script

A practical guide for rerunning, explaining, and adapting the R cleaning workflow

Purpose

This document explains the cleaning script that prepares yearly UCTSAD data files. It is written for someone who may need to run the script again later, explain what was done, or adapt the script for a newer data extract. The aim is not to describe every line of R code. The aim is to make the workflow understandable and reproducible.

The script reads grouped source files, creates yearly application files, and then uses those yearly application files to create matching person, secondary-school-subject, and tertiary-education files. The main output is a set of Stata .dta files named by year and file type.

Main idea

The application file initiates the whole workflow. Once each application record has an application year, the script uses that year to decide which person files. The person file is then used to create school records, and tertiary records belong in each yearly file.

What you need before running the script

Place the R script and the input .dta files in the same folder, unless you choose to set a different input folder in the script. The script uses the two path settings below. Keep them near the top of the script so they are easy to find.

```
input_dir  <- "."
output_dir <- "."
```

The dot means “the current working folder”. If the files are in another folder, replace the dot with that folder path. The output folder is created automatically if it does not already exist.

The script uses these R packages:

```
library(tidyverse)
library(haven)
library(lubridate)
library(purrr)
library(skimr)
```

Install any missing packages before running the script:

```
install.packages(c("tidyverse", "haven", "lubridate", "purrr", "skimr"))
```

The script expects the following input files. File names must match unless you update the file-name list near the top of the script.

Data block	Exact input file name
Application, 2006-2014	UCTSAD 2006-2014 Application Form v0.1.dta
Application, 2015-2019	uctsad-2015-2019-application-form-v1.dta
Application, 2020-2024	uctsad-2020-2024-application-form-v0.1.dta
Person, 2006-2014	uctsad-2006-2014-person-v0.2.dta
Person, 2015-2019	uctsad-2015-2019-person-v1.dta
Person, 2020-2024	uctsad-2020-2024-person-v0.2.dta
Secondary school subject, 2006-2014	uctsad-2006-2014-secondary-school-subject-v0.2.dta
Secondary school subject, 2015-2019	uctsad-2015-2019-secondary-school-subject-v1.dta
Secondary school subject, 2020-2024	uctsad-2020-2024-secondary-school-subject-v0.1.dta
Tertiary education, 2006-2014	uctsad-2006-2014-tertiary-education-v0.2.dta
Tertiary education, 2015-2019	uctsad-2015-2019-tertiary-education-v1.dta
Tertiary education, 2020-2024	uctsad-2020-2024-tertiary-education-v0.1.dta

What is the application process?

An application process consists of multiple stages. For example, an application may move from “application”, “defer decision”, “admit”. In this case, the application has three stages; each with their own instance (row). Each instance has its own unique application identifier. An application process is the grouping of all application stages relating to an single application.

What the script does

The workflow has eight main steps. The order matters because later files depend on earlier files.

1. Load all source .dta files. The script reads application form, person, secondary-school-subject, and tertiary-education files for the three source periods.
2. Prepare the application form file for each source period. The script sorts the application form data by person, application, and application begin date. This makes the application history easier to process in the correct order.
3. Create an application-process identifier. The script uses *app_programaction* and *app_endeffectivedate* to group related rows into the same application process. This is stored in *app_process_id*.
4. Create the application year. For each application process and person, the script takes the earliest *app_begineffectivedate* year and stores it as *year_app*.
5. Apply known anomaly fixes. These are specific corrections for known records where the first calculated year is not the intended application year.
6. Split application data into yearly files. Most years are written directly. The year 2019 is handled separately because both the 2015-2019 and 2020-2024 application files contain 2019 records.
7. Create yearly person files. For each year, the script starts from the yearly application file, keeps the people who applied in that year, and joins their person records.
8. Create yearly secondary-school-subject and tertiary-education files. These files are joined through the yearly person files. Records are kept only when their school or tertiary year is less than or equal to the application year.

Why this matters

The script does not simply split every source file by its own year column. It first decides the year the application was **initiated**, then uses that decision to pull related records from the other files. This keeps the yearly files linked to that application cohort.

The application-process identifier is one of the most important parts of the script. A new process starts when *app_programaction* matches one of the start actions. The original logic tries the actions in this order: application, defer decision, admit, and matriculation.

Action code	Meaning used in the script
6	application
9	defer decision
4	admit
14	matriculation

An application process continues until the first end-date equal to *9999-12-31*. After that point, rows are left without a process ID until another start action appears. This follows the original rule in the cleaning code.

Important choices to know

- The year 2019 is special. Application records for 2019 are taken from both 2015-2019 and 2019-2025 source files, harmonized, combined, sorted, and given a new *app_process_id*. Initially, the 2015-2019 source file application records for 2019 end prematurely relative to the same application records in the 2019-2025 source file. Additionally, application records for 2019 in the 2015-2019 source file had information not in the 2019-2025 source, and vice versa.
- Column names differ across the 3 grouped source files. Older files use personid. Some newer application files use person_id or application_id. The script renames these only where needed.
- For secondary-school-subject files, records are kept when *seced_schoolyear* is less than or equal to the application year, or when *seced_schoolyear* is missing.
- For tertiary-education files, records are kept when terted_year is less than or equal to the application year.

How to reproduce the output

To reproduce the output, follow the same setup each time. This helps avoid accidental differences caused by file paths, renamed files, or skipped steps.

1. Create one project folder for the cleaning task.
2. Put the R script in that folder.
3. Put all required input .dta files in the same folder, or update input_dir to point to the folder where the files are stored.
4. Set output_dir to the folder where the cleaned yearly files should be saved.
5. Open R or RStudio in the project folder.
6. Install any missing packages.
7. Run the script from top to bottom without skipping sections.

8. Check the output folder for the expected yearly .dta files.

```
# Example path setup
input_dir  <- "data/raw"
output_dir <- "data/cleaned"
```

The output file names follow this pattern:

```
uctsad-<year>-<file-type>-v1.dta
```

File type	Years written	Example file name	Note
application-form	2006-2025	uctsad-2020-application-form-v1.dta	2019 is the combined dat2 + dat3 application file.
person	2006-2018, 2020-2025	uctsad-2020-person-v1.dta	The original workflow does not write a 2019 person file.
secondary-school-subject	2006-2018, 2020-2025	uctsad-2020-secondary-school-subject-v1.dta	Uses the yearly person file and keeps records up to the application year.
tertiary-education	2006-2018, 2020-2025	uctsad-2020-tertiary-education-v1.dta	Uses the yearly person file and keeps records up to the application year.

Reproducibility habit

Do not edit the input files by hand before running the script. If a correction is needed, put the correction in the R script so the same result can be produced again later.

How future analysts can adapt the script

The refactored script is designed so that future changes are made in a few clear places instead of being copied across many repeated blocks.

- To use a different folder, change input_dir and output_dir near the top of the script.
- To use updated source files, change the file names inside the files list. Do not search through the whole script for file names.
- To add a new year, update the year ranges used when splitting application files and writing outputs.
- To add a new known correction, edit the relevant anomaly-fix function. Add a short comment explaining why the correction is needed.
- To change the action codes that start an application process, edit the start_values argument in *add_application_process_id()*. Only do this if the data definition has changed.
- To change how person, secondary, or tertiary files are created, edit the helper function for that file type instead of editing every year separately.

Before sharing adapted outputs, a future analyst should check that the number of rows looks reasonable, that each output file has the expected columns, and that the application years in the output match the intended cohort years.

Check	Why it matters	Simple R idea
Files were written	Confirms the script completed and saved outputs.	list.files(output_dir, pattern = "\\.dta$")
Application years look right	Confirms records were split into the intended years.	count(dat2_app, year_app)
No unexpected missing IDs	Helps find join or naming problems.	summarise(data, missing_id = sum(is.na(personid)))
2019 was combined	Confirms dat2 and dat3 2019 application records were both included.	count(dat23_app_2019, year_app)

Check	Why it matters	Simple R idea
Joined files have rows	Confirms person, secondary, and tertiary joins worked.	map_int(dat3_person_by_year, nrow)

A good rule is to save the script, this documentation, the input file list, and a short output log together. That makes it easier for someone else to understand not only what was produced, but also how it was produced.

Summary

This script turns grouped UCTSAD source files into yearly Stata files. It uses the application form data to define the year in which the application was initiated, then creates linked person, secondary-school-subject, and tertiary-education files for those yearly cohorts. The most important logic is the creation of *app_process_id* and the special handling of 2019 application records. To rerun the workflow, keep the input file names stable, set the input and output folders, install the required R packages, and run the script from top to bottom.