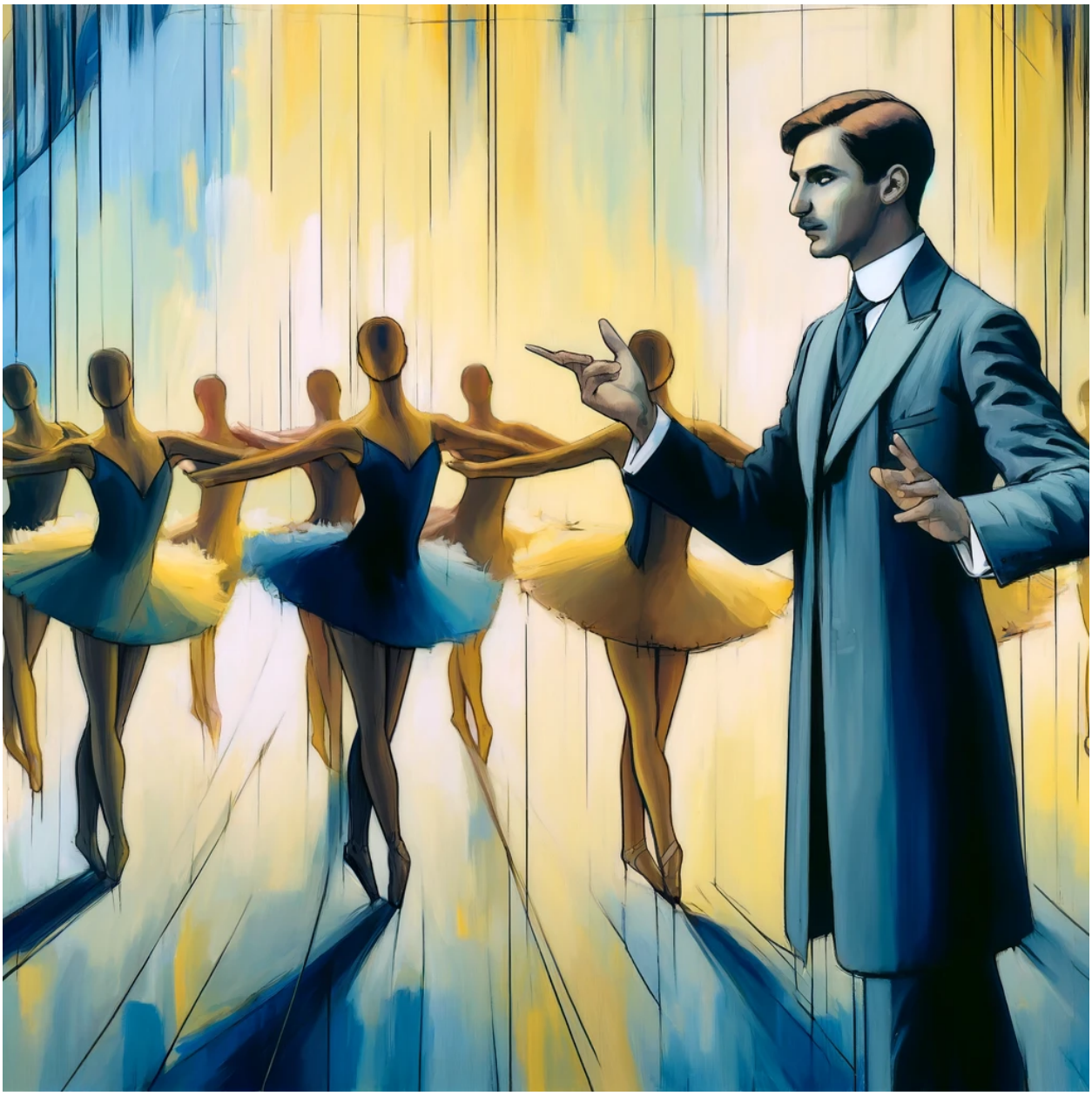
GoodProgrammingPractice

Programming rules for Statistics Denmark

Christian Torp-Pedersen Kathrine Kold Sørensen Filip Gnesin

2024-04-26



Contents

Preamble

This document sets up rules for programming in the research environment of Statistics Denmark. Some of the rules are not relevant elsewhere. The rules in this particular document are not specific to choice of program and equally relevant to users of R, SAS and STATA. Adherence to these rules protect our environment from microdata violations, ease bug finding and ease collaboration.

Updated January 7, 2024 - with recommendation only to use original source data Updated June 7, 2024 - A note regarding "disk full" messages when using SAS has been added to the resource section.

Security first

Programs are not only the paths from data to result, but also the only documentation a user may have of this path which makes safe keeping of programs essential in case critical questions to results should arise. It is therefore essential that programs are build such that they can be exported from Statistics Denmark - which implies that they do not contain what Statistics Denmark call "microdata". The most serious of microdata is the encrypted cpr-number and these numbers may therefore never appear in programs. When it appears convenient to base program logic on encrypted individual cpr-numbers, these statements should be replaced with the logic that was used to identify the encrypted numbers. Because rules may be violated it is particularly important that programs are checked extremely carefully prior to rare cases of export (or transfer to another project). In the export chapter of "Rules of Engagement" there are suggestions for search strings to use.

Project structure

- Each project (resulting in a manuscript) requires its own folder - see "Rules of Engagement".
- This folder should include a "readme" file which describes briefly the project. This is important if another project with relation to the current project is started and old programs needs to be interrogated after people have forgotten many details.
- The scripts creating the results need to appear in very few files. Old versions and various side projects should be placed in a "sandbox" or similar subfolder clearly indicating the secondary importance.

Program structure

- Separate data management from analysis. Often it is useful to have two files for a project, one for data management and the other containing the analyses and statements that produce the tables and graphics for a paper.

- All your programming should start with original source programs. This is to make sure that you in the future can document how results were obtained. If you start with data produced by another user there is a risk that these data have changed at a time where you need to document your results. It is fine to copy the program another user has made to your folder for use there.
- For R-users the "targets" pipeline structure is highly recommended since it creates an easily identifiable structure and makes it easy to ask for help from others.
- Programs should not have excessive length. Many programs have sequences of similar chunks that for example perform the same calculations for several outcomes or several subgroups. Instead of repeating chunks with small changes it is important to contain such structures in loops. There are a number of efficient ways to construct loops with R and for SAS the method is included in macro programming. Programs of excessive length are difficult to interrogate, difficult to debug and hinders collaboration.
- Programs need to be richly commented - while avoiding microdata. Note that a remark such as "This removed one individual" is disclosure of microdata.

House keeping

- During development of a program many little tests are performed, which is encouraged. However, if you want to keep testing versions of a program, use a separate directory such as the "sandbox", and remove any testing statements from the main programs, when testing has been done.
- If in any way possible all tables and graphs for the projects should be completed within the program. Combining multiple results manually in tables increases risk of errors. We have many projects using similar data and the final publications of these should not have data that obviously differ. Note also that a table only rarely is prepared a single time, but it prepared multiple times with small changes. Having a program shape tables reduces work load over the course of a project.
- Make programs where not everything has to be rerun after changes. For many projects a single final dataset is used for many calculations. This suboptimal case is often generated by consecutive addition of additional variables from other datasets to a main dataset. In this case however, if just one of the steps require a change everything has to be rerun. It is wiser to create the components independently and then have a final large merge step in the end. In this case only the component needing change and the final merge needs to be rerun. Using targets pipelines is an elegant solution to this.

Files and history

- When you make major changes, then keep the old version. You can have a directory for old versions or setup GIT to keep record. If your results suddenly change this can be the only way of determining whether the changes reflect correcting a problem or creating a problem.
- Keep only moderate sized analysis datasets as files. Huge datasets can be created and clog the resources of our environment. Old datasets can be recreated by rerunning programs.
- Keep the final analysis dataset until the paper is published. Data on Statistics Denmark are periodically updated which can result in changes.
- All datasets should be deleted when the paper has been published, but you need to make sure that the programs can be found even years later. In line with this requirement it is unwise to have programming which reads derived data from other users. If you need such data, then copy the programming that generated the data to your folder to make sure that your results can be derived with your programs directly from the raw data provided by Statistics Denmark.

Resource use

- Keep track of resource use. For R you need to check RAM consumption and regularly run the garbage collector **gc()**. You can check your RAM use in DST by using Ctrl-Alt-Delete –> Task Manager –> Users.
- SAS places default work-data in a system specified drive. This drive may be overfilled on occasion. The immediate solution is to create a directory on Z og V named something like "mywork", give it a lib-reference and use that for data.
- Keep the final analysis dataset until publication to ensure that data are not updated while waiting for journal response, but do not keep excessive intermediate datasets. You need to be aware of your use of disk space. SAS users should rely on the "work" directory and only keep final datasets in permanent folders.
- Parallel processing is encouraged, but do not take so many resources that others cannot work.
- Having check on resource use is particularly important when you are not actively working. We have common problems of users clogging RAM but not using it actively. It should be a firm habit to check your use of resources prior to discontinuating for the day.