Reproducibility when data are confidential

Lars Vilhuber, with contributions by Laurel Krovetz

Aug 01, 2025

CONTENTS

Journals require that you **share your code and data** in a replication package at the end of your research project. That can be a challenge, when some parts of your research data are confidential. This document provides an overview of ensuring the reproducibility of your research when **data are confidential**. It is not meant to be exhaustive, and it is not meant to be prescriptive. There are many ways to construct a replication package, and even more situations in which confidential data are housed.

Following some best practices from day 1 can not only **help you prepare** this package later, but also make you **more productive** researchers. Following some best practices before releasing a package can **avoid costly revisions**.

Before we start

Many of the methods and techniques described here are not specific to confidential data. Before we go into the details, we suggest that you read the following chapters and presentations. We will refer to them at particular points in this document.

- Reproducibility from Day 1
- Verifying Reproducibility Yourself
- The Ideal README

Alternate formats

This subject is also available as

- an online presentation and its printable PDF (also in Spanish *)
- a printable PDF *

(* indicates work-in-progress, WIP)

TL;DR

Techy lingo for “too long, didn’t read”. A summary of the most important takeaways will be at the top of each section.

How to contribute

Open a pull request at the repository, which can be done from every page using the buttons at the top right.

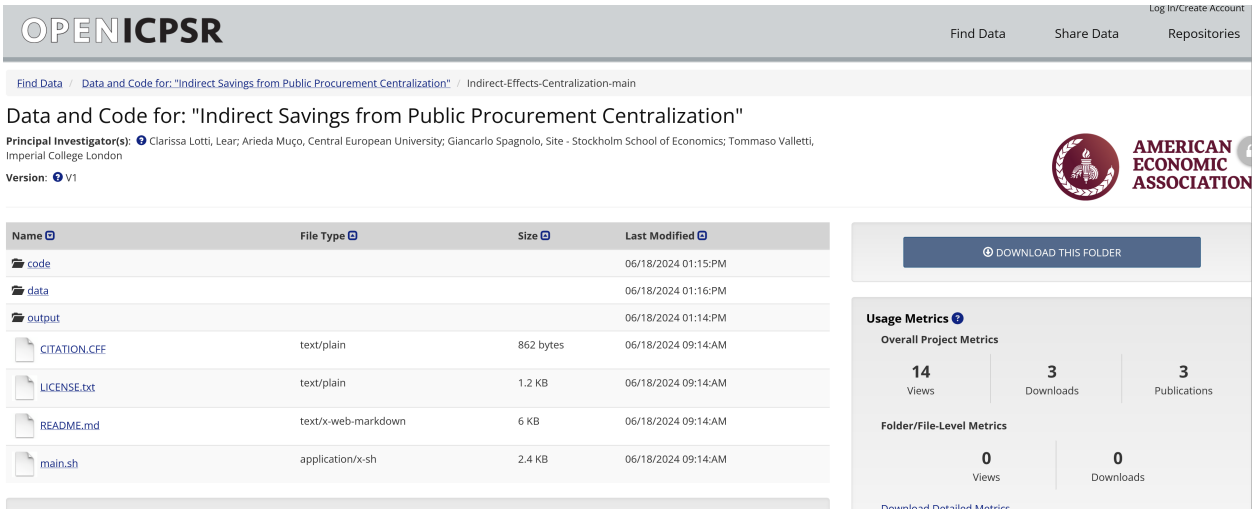
CHAPTER

ONE

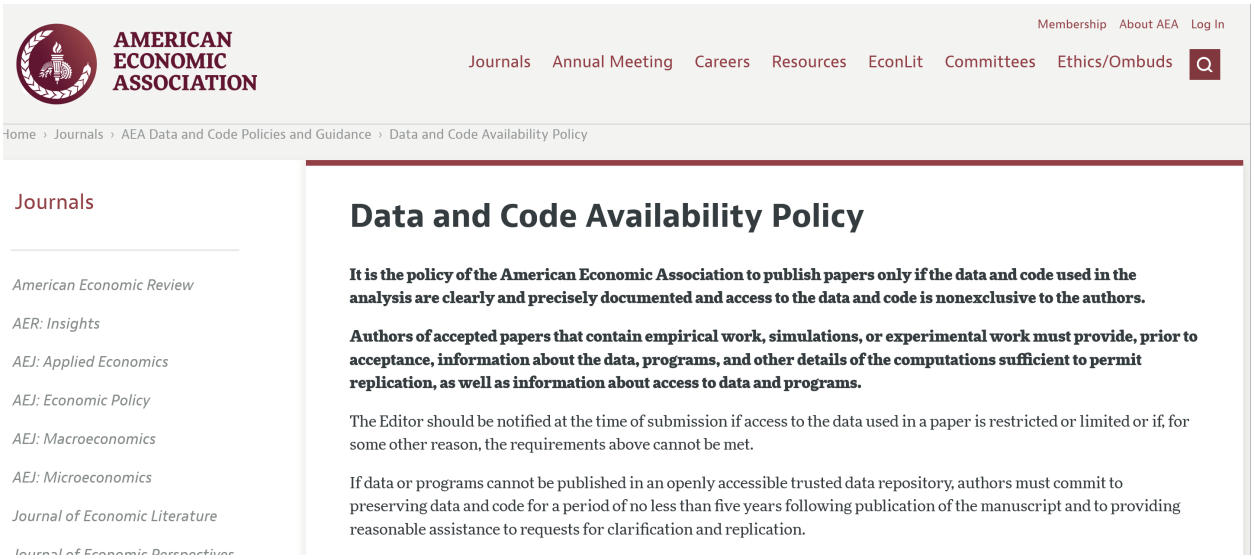
WHAT IS A REPLICATION PACKAGE?

- AEA Data and Code Availability policy
- Data and Code Availability Standard
- AEA Data and Code Repository

1.1 Example of deposit



1.2 AEA policy



AMERICAN ECONOMIC ASSOCIATION

Membership About AEA Log In

Journals Annual Meeting Careers Resources EconLit Committees Ethics/Ombuds

Home › Journals › AEA Data and Code Policies and Guidance › Data and Code Availability Policy

Journals

American Economic Review

AER: Insights

AEJ: Applied Economics

AEJ: Economic Policy

AEJ: Macroeconomics

AEJ: Microeconomics

Journal of Economic Literature

Journal of Economic Perspectives

Data and Code Availability Policy

It is the policy of the American Economic Association to publish papers only if the data and code used in the analysis are clearly and precisely documented and access to the data and code is nonexclusive to the authors.

Authors of accepted papers that contain empirical work, simulations, or experimental work must provide, prior to acceptance, information about the data, programs, and other details of the computations sufficient to permit replication, as well as information about access to data and programs.

The Editor should be notified at the time of submission if access to the data used in a paper is restricted or limited or if, for some other reason, the requirements above cannot be met.

If data or programs cannot be published in an openly accessible trusted data repository, authors must commit to preserving data and code for a period of no less than five years following publication of the manuscript and to providing reasonable assistance to requests for clarification and replication.

CHAPTER

TWO

GOAL

- Provide guidance on structure of replication packages when data are confidential
- Provide guidance on documentation
- Keep it simple

CHAPTER

THREE

THE FINAL REPLICATION PACKAGE

The finished product replication package should have some version of the following contents and structure. It should include all code (whether used in RDC or not), all public data (whether used in RDC or not). A full description can be found here.

```
README.md
README.pdf
code/
├── fsrdc/
│   ├── 01-prepare-data.R
│   ├── 02-analyze-data.R
│   └── 03-create-disclosable-data.R
├── public/
│   ├── 04-create-tables.do
│   ├── 05-create-figures.do
│   └── 06-create-intext.do
data/
├── public/
│   ├── dist_cepii.dta
│   └── usa_00010.dta
run.sh
```

CHAPTER

FOUR

THE README FILE

There are three key components to the README file:

- Data availability (and citations)
- Computer requirement
- Description of processing

4.1 Data availability

This is easy: it's the data you requested to have included in your FSRDC project. So you had this info from **day -90** of the project!

In order to describe the data availability, split it into two:

- how did **you** get access to the data (that's old)
- how can **others** get access to the same data (this might be different!)

The above two might not always be the same, but both are relevant.

4.1.1 Examples

- This excellent description from a paper by Teresa Fort (ReStud):
  1. All the results in the paper use confidential microdata from the U.S. Census Bureau. To gain access to the Census microdata, follow the directions here on how to write a proposal for access to the data via a Federal Statistical Research Data Center: https://www.census.gov/ces/rdcresearch/howtoapply.html.
  2. You must request the following datasets in your proposal:
     - Longitudinal Business Database (LBD), 2002 and 2007
     - Foreign Trade Database – Import (IMP), 2002 and 2007
     - Annual Survey of Manufactures (ASM), including the Computer Network Use Supplement (CNUS), 1999
     - […]
     - Annual Survey of Magical Inputs (ASMI), 2002 and 2007
  3. Reference

  - "Technology and Production Fragmentation: Domestic versus Foreign Sourcing" by Teresa Fort, project number br1179 in the proposal. This will give you access to the programs and input datasets required to reproduce the results. Requesting a search of archives with the articles DOI ("10.1093/restud/rdw057") should yield the same results.

    NOTE: Project-related files are available for 10 years as of 2015.

- This description by Fadlon and Nielsen about Danish data
  - The information used in the analysis combines several Danish administrative registers (as described in the paper). The data use is subject to the European Union's General Data Protection Regulation(GDPR) per new Danish regulations from May 2018. The data are physically stored on computers at Statistics Denmark and, due to security considerations, the data may not be transferred to computers outside Statistics Denmark.

    Researchers interested in obtaining access to the register data employed in this paper are required to submit a written application to gain approval from Statistics Denmark. The application must include a detailed description of the proposed project, its purpose, and its social contribution, as well as a description of the required datasets, variables, and analysis population.

    Applications can be submitted by researchers who are affiliated with Danish institutions accepted by Statistics Denmark, or by researchers outside of Denmark who collaborate with researchers affiliated with these institutions.

(Example taken from Fadlon and Nielse, AEJ:Applied 2021).

- Also grant permission to your project files: I grant any researchers with appropriate Census-approved project permission to use my exact research files provided that those files were among the ones that they requested when the approval was obtained (a Census Bureau requirement). These files can be found by searching for the DOI of [this archive/ this article] amongst backups/archives made in [month of archive].

4.1.2 Don't forget to cite the data

Bureau of the Census. (release year). American Community Survey-Master Address File Crosswalk YYYY-YYZZ [Data File]. Federal Statistical Research Data Center [distributor].

Graf, Tobias; Grießemer, Stephan; Köhler, Markus; Lehnert, Claudia; Moczall, Andreas; Oertel, Martina; Schmucker, Alexandra; Schneider, Andreas; Seth, Stefan; Thomsen, Ulrich; vom Berge, Philipp (2023): "Weakly anonymous Version of the Sample of Integrated Labour Market Biographies (SIAB) – Version 7521 v1". Research Data Centre of the Federal Employment Agency (BA) at the Institute for Employment Research (IAB). https:/doi.org/10.5164/IAB.SIAB7521.de.en.v1

- Further examples on Zotero for FSRDC (this is possibly not the most current).
- Ideally, every research data center would have "landing pages" for the data (the IAB example does).

4.2 Computer requirements

In most confidential environments, such as FSRDC/IRE, this part is out of your control. But you should describe it anyway! You should include the approximate description of computer/nodes used. This includes memory size (though we are interested in actual usage, not the maximum of what the system has), and compute time (how long does a clean run, from top to bottom, take?), and number of nodes (any parallel processing?). You should also detail the software used, that is, the version of software (Stata 17, update level), and all packages, ideally version of package (which `estout`).

4.2.1 FSRDC

Did you use PBS? Include the `qsub` files. Or if you used `qstata` or such, describe that:

```
...
run.sh
qsub-complete.sh
```

4.3 Description of processing

That's easy: you've been keeping clean instructions since the start, right?

- Run `main.do` or `run.sh`
- Describe what parts might be skipped
- Describe what the various parts do
- Describe which parts use confidential data

You've been doing that **since day 1!**

4.4 Three parts to README: timing

- Data availability (and citations) – start of the project, edit at the end
- Computer requirement – middle of the project
- Description of processing – middle of the project

With the end really just a last read/edit.

CHAPTER

FIVE

ENVIRONMENTS IN STATA

5.1 TL;DR

- Creating virtual environments in Stata is feasible
- Doing so stabilizes the code, and makes it more transportable

5.2 Search paths in Stata

In Stata, we typically do not talk about environments, but the same basic structure applies: Stata searches along a set order for its commands. Some commands are built into the executable (the software that is opened when you click on the Stata icon), but most other internal, and all external commands, are found in a search path. This is typically the `ado` directory in the Stata installation directory, and one will find replication packages that contain instructions to copy files into that directory. Once we've shown how environments work in Stata, this will become a lot simpler!

5.2.1 The `sysdir` directories

The default set of directories which can be searched, from a freshly installed Stata, can be queried with the `sysdir` command, and will look something like this:

```
sysdir
```

```
   STATA:  C:\Program Files\Stata18\
    BASE:  C:\Program Files\Stata18\ado\base\
    SITE:  C:\Program Files\Stata18\ado\site\
    PLUS:  C:\Users\lv39\ado\plus\
PERSONAL:  C:\Users\lv39\ado\personal\
OLDPLACE:  c:\ado\
```

5.2.2 The `adopath` search order

The search paths where Stata looks for commands is queried by `adopath`, and looks similar, but now has an order assigned to each entry:

```
adopath
```

```
  [1]  (BASE)      "C:\Program Files\Stata18\ado\base/"
  [2]  (SITE)      "C:\Program Files\Stata18\ado\site/"
  [3]              "."
  [4]  (PERSONAL)  "C:\Users\lv39\ado\personal/"
  [5]  (PLUS)      "C:\Users\lv39\ado\plus/"
  [6]  (OLDPLACE)  "c:\ado/"
```

To look for a command, say `reghdfe`, Stata will look in the first directory, then the second, and so on, until it finds it. If it does not find it, it will return an error. We can query the location of `reghdfe` explicitly with `which`:

```
which reghdfe
```

```
command reghdfe not found as either built-in or ado-file
r(111);
```

5.2.3 Where are packages installed?

When we install a package, using one of the various package installation commands (`net install`, `ssc install`)[1], only one of the (`sysdir`) paths is relevant: `PLUS`. So if we install `reghdfe` with `ssc install reghdfe`, it will be installed in the `PLUS` directory, and we can see that with `which`:

```
ssc install reghdfe
which reghdfe
```

```
. ssc install reghdfe
checking reghdfe consistency and verifying not already installed...
installing into C:\Users\lv39\ado\plus\...
installation complete.

. which reghdfe
C:\Users\lv39\ado\plus\r\reghdfe.ado
*! version 6.12.3 08aug2023
```

Important

It is important here to recognize that it is the value of the special `sysdir` directory `PLUS` that determines where Stata installs commands, but the separate list of `adopath` locations where it looks for commands. It is possible to install a command in a location that Stata does not look for commands!

[1] `net install` refererence. Strictly speaking, the location where ado packages are installed can be changed via the `net set ado` command, but this is rarely done in practice, and we won't do it here.

5.3 Using environments in Stata

But the `(PLUS)` directory can be manipulated, and that creates the opportunity to create an "environment".

```
* Set the root directory

global rootdir : pwd

* Define a location where we will hold all packages in THIS project (the "environment
 ↪")

global adodir "$rootdir/ado"

* make sure it exists, if not create it.

cap mkdir "$adodir"

* Now let's simplify the adopath
* - remove the OLDPLACE and PERSONAL paths
* - NEVER REMOVE THE SYSTEM-WIDE PATHS - bad things will happen!

adopath - OLDPLACE
adopath - PERSONAL

* modify the PLUS path to point to our new location, and move it up in the order

sysdir set PLUS "$adodir"
adopath ++ PLUS

* verify the path

adopath
```

which should show something like this:

```
. adopath
  [1]  (PLUS)      "C:\Users\lv39\Documents/PROJECT123/ado/"
  [2]  (BASE)      "C:\Program Files\Stata18\ado\base/"
  [3]  (SITE)      "C:\Program Files\Stata18\ado\site/"
  [4]              "."
```

Let's verify again where the `reghdfe` package is:

```
which reghdfe
```

```
. which reghdfe
command reghdfe not found as either built-in or ado-file
r(111);
```

So it is no longer found. Why? Because we have removed the previous location (the old `PLUS` path) from the search sequence. It's as if it didn't exist.

5.4 Installing packages when an environment is active

When we now install `reghdfe` again:

```
. ssc install reghdfe
checking reghdfe consistency and verifying not already installed...
installing into C:\Users\lv39\Documents\PROJECT123\ado\plus\...
installation complete.

. which reghdfe
C:\Users\lv39\Documents\PROJECT123\ado\plus\r\reghdfe.ado
*! version 6.12.3 08aug2023
```

We now see it in the **project-specific** directory, which we can distribute with the whole project (more on that *later*).

5.5 Installing precise versions of packages

Let's imagine we need an older version of `reghdfe`. In general, it is **not** possible in Stata to install an older version of a package in a straightforward fashion. You *may* have success with the Wayback Machine archive of SSC, which in some cases goes back to 2000, by carefully reconstructing the necessary files.

Most package repositories are versioned:

- R: CRAN, Bioconductor
- Python: PyPI
- Julia: "General" default Julia package registry.

State does not (as of 2024). But see the full site for one approach.

CHAPTER

SIX

TAKEAWAYS

From the earlier desiderata of *environments*:

- **Isolated**: Installing a new or updated package for one project won't break your other projects, and vice versa.
- **Portable**: Easily transport your projects from one computer to another, even across different platforms.
- **Reproducible**: Records the exact package versions you depend on, and ensures those exact versions are the ones that get installed wherever you go.

CHAPTER

SEVEN

SECRETS IN THE CODE

What are the secrets?

- API keys
- Login credentials for data ccess
- file paths (FSRDC!)
- Variable names (IRS!)

Store secrets in environment variables or files that are not published. Some services are serious about this, such as Github secret scanning:

> **About secret scanning**
>
> GitHub scans repositories for known types of secrets, to prevent fraudulent use of secrets that were committed accidentally.

7.1 Where to store secrets

Store secrets in **environment variables** such as "dot-env" files in Python, "Renviron" files in R, or some other clearly identified file in the project or home directory.

Example typed interactively (for Linux and Mac):

```
MYSECRET="dfad89ald"
CONFDATALOC="/path/to/irs/files"
```

The above is **not** recommended.

For storing secrets in files, use the same syntax as for contents of "dot-env" or "Renviron" files, and in fact `bash` or `zsh` files (`.bash_profile`, `.zshrc`).

R

Edit `.Renviron` (note the dot!) files:

```
# Edit global (personal) Renviron
usethis::edit_r_environ()
# You can also consider creating project-specific settings:
usethis::edit_r_environ(scope = "project")
```

And use the variable defined in `.Renviron`:

```
mysecret <- Sys.getenv('MYSECRET')
```

Python

Loading regular environment variables:

```
import os
mysecret = os.getenv("MYSECRET")  # will load environment variables
```

Loading with `dotenv`:

```
from dotenv import load_dotenv
load_dotenv()  # take environment variables from project .env.
mysecret = os.getenv("MYSECRET")  # will load environment variables
```

Stata

Yes, this also works in Stata

```
// load from environment
global mysecret : env MYSECRET
display "$mysecret"  // don't actually do this in code
```

and via (what else) a user-written package for loading from files:

```
net install doenv, from(https://github.com/vikjam/doenv/raw/master/)
doenv using ".env"
global mysecret "`r(MYSECRET)'"
display "$mysecret"
```

Simplest solution:

```
//============ non-confidential parameters =========
include "config.do"

//============ confidential parameters =============
capture confirm file "$code/confidential/confparms.do"
if _rc == 0 {
    // file exists
    include "$code/confidential/confparms.do"
} else {
    di in red "No confidential parameters found"
}
//============ end confidential parameters =========
```

CHAPTER

EIGHT

CONFIDENTIAL CODE

What is confidential code?

- In the United States, some variables on **IRS databases** are considered super-top-secret. So you can't name that-variable-that-you-filled-out-on-your-Form-1040 in your analysis code of same data. (They are often referred to in jargon as "Title 26 variables").
- Your code contains the random seed you used to anonymize the sensitive identifiers. This might allow someone to reverse-engineer the anonymization, and is not a good idea to publish.
- You used a look-up table hard-coded in your Stata code to anonymize the sensitive identifiers (replace `anon-county=1 if county="Tompkins, NY"`). A **really bad idea**, but yes, you probably want to hide that.
- Your IT specialist or disclosure officer thinks publishing the **exact path** to your copy of the confidential 2010 Census data, e.g., "/data/census/2010", is a security risk and refuses to let that code through.
- You have adhered to disclosure rules, but for some reason, the precise minimum cell size is a confidential parameter.

So whether reasonable or not, **this is an issue**. How do you do that, without messing up the code, or spending hours redacting your code?

8.1 Example

This will serve as an example. None of this is specific to Stata, and the solutions for R, Python, Julia, Matlab, etc. are all quite similar.

Assume that variables `q2f` and `q3e` are considered confidential by some rule, and that the minimum cell size `10` is also confidential.

```
set seed 12345
use q2f q3e county using "/data/economic/cmf2012/extract.dta", clear
gen logprofit = log(q2f)
by county: collapse (count)  n=q3e (mean) logprofit
drop if n<10
graph twoway n logprofit
```

The final line is the only line of code that does not contain "confidential" information.

8.1.1 Bad

A bad example, because it is literally making more work for you and for future replicators, is to manually redact the confidential information with text that is not legitimate code (do not do this!):

```
set seed NNNNN
use <removed vars> county using "<removed path>", clear
gen logprofit = log(XXXX)
by county: collapse (count)  n=XXXX (mean) logprofit
drop if n<XXXX
graph twoway n logprofit
```

The redacted program above will no longer run, and will be very tedious to un-redact if a subsequent replicator obtains legitimate access to the confidential data.

8.1.2 Better

Simply replace the confidential data with replacement that are valid placeholders in the programming language of your choice is already better. Here's the confidential version of the file:

```
//============ confidential parameters =============
global confseed    12345
global confpath    "/data/economic/cmf2012"
global confprofit  q2f
global confemploy  q3e
global confmincell 10
//============ end confidential parameters =========
set seed $confseed
use $confprofit county using "${confpath}/extract.dta", clear
gen logprofit = log($confprofit)
by county: collapse (count)  n=$confemploy (mean) logprofit
drop if n<$confmincell
graph twoway n logprofit
```

And the following could be the released file, part of the replication package:

```
//============ confidential parameters =============
global confseed    XXXX    // a number
global confpath    "XXXX"  // a path that will be communicated to you
global confprofit  XXX     // Variable name for profit T26
global confemploy  XXX     // Variable name for employment T26
global confmincell XXX     // a number
//============ end confidential parameters =========
set seed $confseed
use $confprofit county using "${confpath}/extract.dta", clear
gen logprofit = log($confprofit)
by county: collapse (count)  n=$confemploy (mean) logprofit
drop if n<$confmincell
graph twoway n logprofit
```

While the code won't run as-is, it is easy to un-redact, regardless of how many times you reference the confidential values, e.g., `q2f`, anywhere in the code.

8.1.3 Best

- Main file
- Conditional processing
- Separate file for confidential parameters which can simply be excluded from disclosure request

Main file `main.do`:

```
//============ confidential parameters =============
capture confirm file "$code/confidential/confparms.do"
if _rc == 0 {
    // file exists
    include "$code/confidential/confparms.do""
} else {
    di in red "No confidential parameters found"
}
//============ end confidential parameters =========

//============ non-confidential parameters =========
global safepath "$rootdir/releasable"
cap mkdir "$safepath"

//============ end parameters ======================

// :::: Process only if confidential data is present

capture confirm  file "${confpath}/extract.dta"
if _rc == 0 {
   set seed $confseed
   use $confprofit county using "${confpath}/extract.dta", clear
   gen logprofit = log($confprofit)
   by county: collapse (count)  n=$confemploy (mean) logprofit
   drop if n<$confmincell
   save "${safepath}/figure1.dta", replace
} else { di in red "Skipping processing of confidential data" }

//============ at this point, the data is releasable ======
// :::: Process always

use "${safepath}/figure1.dta", clear
graph twoway n logprofit
graph export "${safepath}/figure1.pdf", replace
```

Auxiliary file `$code/confidential/confparms.do"` (not released):

```
//============ confidential parameters =============
global confseed    12345
global confpath    "/data/economic/cmf2012"
global confprofit  q2f
global confemploy  q3e
global confmincell 10
//============ end confidential parameters =========
```

Auxiliary file `$code/include/confparms_template.do` (this is released):

```
//============ confidential parameters =============
// Copy this file to $code/confidential/confparms.do and edit
```

(continues on next page)

(continued from previous page)

```
global confseed    XXXX    // a number
global confpath    "XXXX"  // a path that will be communicated to you
global confprofit  XXX     // Variable name for profit T26
global confemploy  XXX     // Variable name for employment T26
global confmincell XXX     // a number
//============ end confidential parameters =========
```

Thus, the best replication package would have:

```
...
code/main.do
README.md
include/confparms_template.do
releasable/figure1.dta
releasable/figure1.pdf
```

CHAPTER

NINE

AVOIDING CONFIDENTIAL DATA IN YOUR CODE

9.1 The problem

We often see code that "fixes" problems in the data by hard-coding a mapping:

```
# ... 1000 lines of code above...
# Bad practice
data$name[data$name == "Joe Biden"] <- "Joseph Robinette Biden Jr."
data$county[data$county == "Tompins, NY"] <- "Tompkins County, NY"
# ... 500 lines of code below ...
```

Why is this a problem? The information in columns `name` or `county` might be confidential. By coding this information as part of your programs, you have made the **code** confidential! And so you may now have to redact the code before releasing.

9.1.1 One solution

As before, you might move this code into a separate file:

```
# ... 1000 lines of code above...
# Better practice
source("confidential/mappings.R")
# ... 500 lines of code below ...
```

9.1.2 Better solution

If you realize that the mapping is actually **data**, then treating it as any other data (much of which might also be confidential) is both more robust and more manageable while being secure.

```
if (!file.exists("data/confidential/names_mapping.csv")) {
    names_confidential %>%
        left_join(read_csv("data/confidential/names_mapping.csv"), by = "name") %>%
        # replace name with name_alt if the latter is not NA
        mutate(name = if_else(!is.na(name_alt), name_alt, name)) %>%
        # drop the name_alt column
        select(-name_alt) -> names_clean
}
```

Note: You may still want to de-identify the data before releasing it! The code, however, is now **free of confidential information**.

See sample R code in this Github repository for an example where we treat presidents' names as confidential data.

CHAPTER

TEN

WRAPPING UP

Public replication package contains intelligible code, omits confidential details (but provides template code), has detailed data provenance statements. Confidential replication package contains all the same, plus the confidential code, is archived in the FSRDC.

10.1 Things to remember

- Use code to save figures and tables (`estout`, `graph export`, `regsave`).
- Create log files for each run (`stata -b do file.do` not fine-grained enough) (link)
- Run it all again, top to bottom!
- When doing a disclosure review request, remember to request the **code**.
- When outputting statistics, *consider the disclosure rules* - the less changes, the faster the output (in theory), but in particular fewer surprises.
- Do not think *"nobody will ever read this code"* - somebody is very likely to!
-

CHAPTER

ELEVEN

APPENDIX

11.1 Keeping on top of provenance

- Licenses
- Streamlining for reproducibility

11.1.1 Licenses

Where does the file come from?

- How can we describe this later to somebody?
  - Point and click is long to describe.
  - What are the rights we have?

Examples:

- Creative Commons licenses, used for artistic products and data
- Open Source licenses (BSD, GPL, MIT, etc.), used for software (code)

License applying to Geodist data

- CEPII GeoDist is under an "Etalab 2.0 license"

11.2 Downloading via code

Easiest:

Stata

```
use "$URL" , clear
```

Why not?

- Will it be there in two months? In six years?
- What if the internet connection is down?

Easy:

Stata

```
global URL "https://www.cepii.fr/distance/dist_cepii.dta"
copy "$URL" (outputfile), replace
```

R

```
rootdir <- getwd()
datadir <- paste(rootdir, "data", sep = "/")
```

11.3 Creating a README

- Template README
- Cite both dataset and working paper.
- Add data URL and time accessed (can you think of a way to automate this?).
- Add a link to license (also: download and store the license).

11.4 Links

Some additional guidance can be found on the website of the Social Science Data Editors (URLs subject to change):

- https://social-science-data-editors.github.io/guidance/Guidance/Requested_information_hosting.html
- https://social-science-data-editors.github.io/guidance/DCAS_Restricted_data.html#us-census-bureau-and-fsrdc

11.5 Additional training resources

- "Day 1 Tutorial": Presented on Sept 12, 2024 at FSRDC conference (pre-program), subject to change: https://larsvilhuber.github.io/day1-tutorial/
- General purpose guidance about "self checking" your reproducibility package: https://larsvilhuber.github.io/self-checking-reproducibility/

11.6 Examples of replication packages

- https://doi.org/10.3886/E154241V2 not only code, but faces the problem that IRS data cannot have variables revealed. Their workaround is not the same one as in this tutorial.
- https://www.openicpsr.org/openicpsr/project/162581/version/V1/view

This textbook's source: https://github.com/labordynamicsinstitute/reproducibility-confidential