Reproducibility when data are confidential

Lars Vilhuber

2025-01-01

Follow along

labordynamicsinstitute.github.io/reproducibility-confidential/ (PDF)

Reproducibility when data are confidential

Journals require that you **share your code and data** in a replication package at the end of your research project.

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**.

Aside

When typing

> Following some best practices before releasing a package can **avoid costly revisions**.

my coding AI suggested that I add

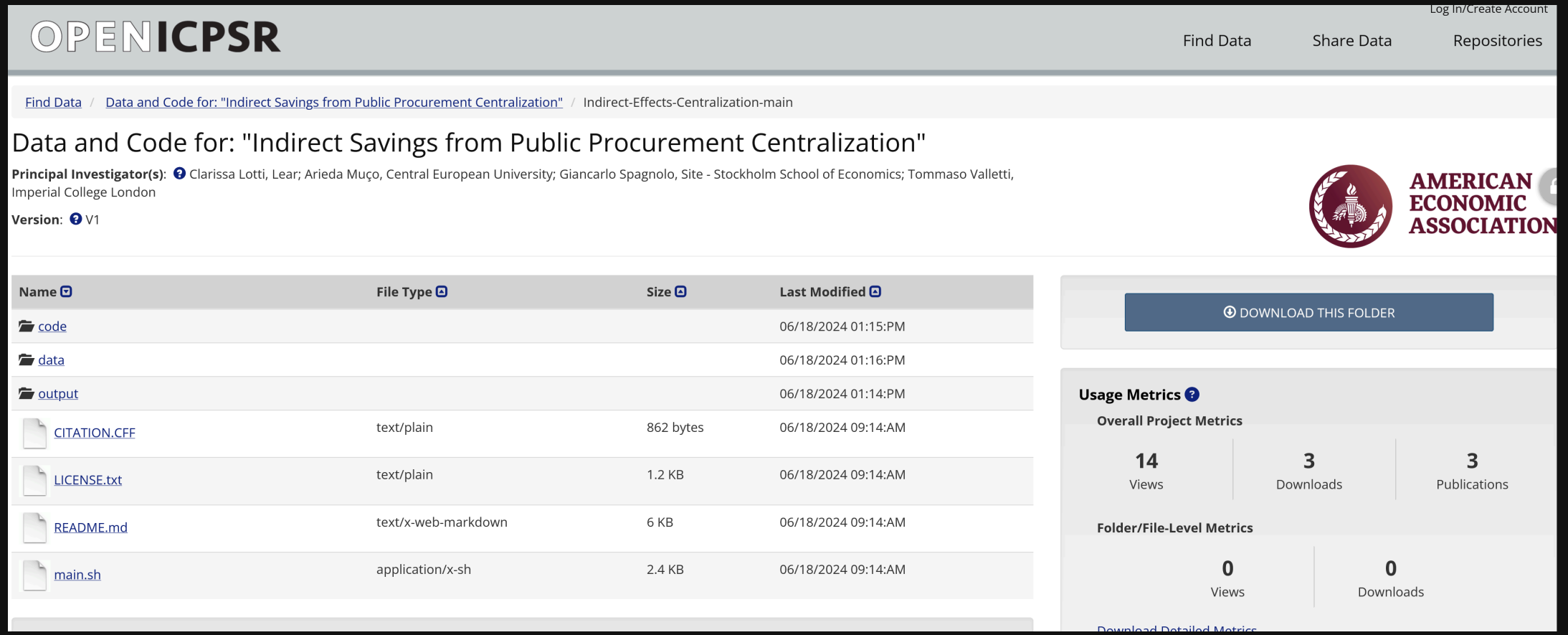
> “and embarrassing retractions”…

What is a replication package?

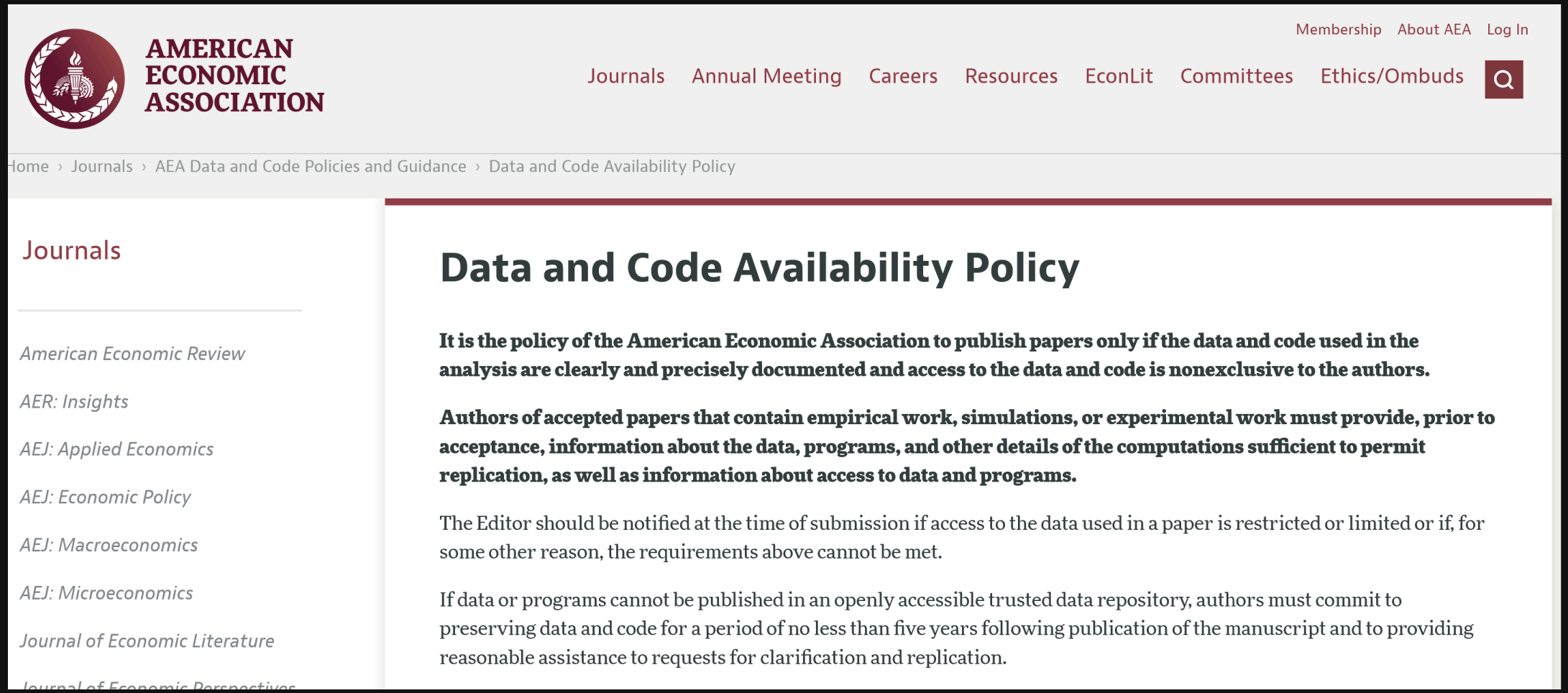
- AEA Data and Code Availability policy
- Data and Code Availability Standard 

- AEA Data and Code Repository

Example of deposit



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.

Goal

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

The final replication package

Files

Contents of a package

```
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
```

Files

All code, whether used in RDC or not

```
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
```

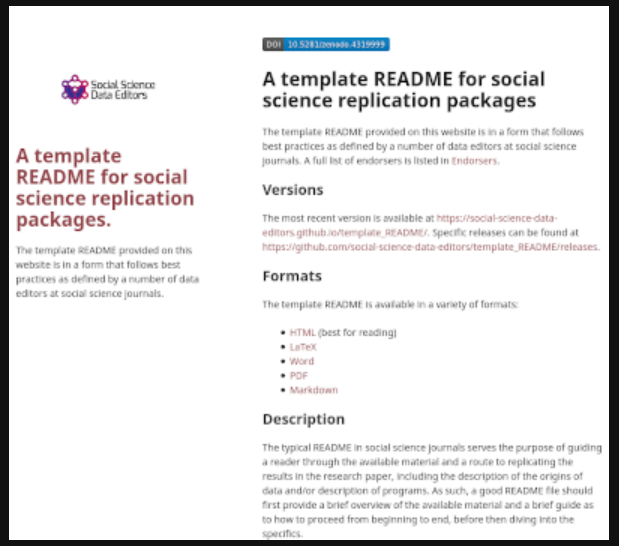
Files

All public data, whether used in RDC or not

```
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
```

Content

Full description as per the (template) README



README

The README

Three parts to README

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

Start with the last part

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!**

Computer requirements

In most confidential environments, such as FSRDC/ IRE, this part is out of your control. But describe it anyway!

Computer requirements

- Approximate description of computers/nodes used
  - memory size (but interested in actual usage, not max of what the system has!)
  - compute time! How long does a clean run, from top to bottom, take?
  - number of nodes: any parallel processing?
- Software
  - Version of software (Stata 17, update level)
  - All packages! Ideally, version of package (`which`

Computer requirements (FSRDC)

- Did you use PBS? Sure you did.

Include the `qsub` files! (Or if you used `qstata` or such, describe that).

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

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!

Data availability redux

In order to describe data availability, split into two:

- how did YOU get access to the data (that's old)
- how can OTHERS get access to the same data (that might be different!)
- The two are not always the same, but are both relevant.

Examples

Examples include

- this excellent description from a paper by Teresa Fort (ReStud):

Examples

Examples include

- 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.
>
> •

Examples

Examples include

- this excellent description from a paper by Teresa Fort (ReStud):

> 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

Examples

Examples include

- this excellent description from a paper by Teresa Fort (ReStud):

> 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.
> -

Examples

Examples include

- this excellent description from a paper by Teresa Fort (ReStud):

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

Examples

Examples include

- 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.

Examples

Examples include

- this description by Fadlon and Nielsen about Danish data

> 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.

Examples

Examples include

- this description by Fadlon and Nielsen about Danish data

> 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 Nielsen, AEJ:Applied 2021).

Examples

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].

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 (possibly not the most current).
- Ideally, every research data center would have “landing pages” for the data (the IAB example does)

Three parts to README: timing

- Data availability (and citations):	**Start of project, edit at the end**
- Computer requirements:	Middle of project
- Description of processing:	Middle of project

with the end really just a last read/edit.

Environments in Stata

TL;DR

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

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.

Search paths in Stata

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.

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:

```
1  sysdir
```

```
1     STATA:  C:\Program Files\Stata18\
2      BASE:  C:\Program Files\Stata18\ado\base\
3      SITE:  C:\Program Files\Stata18\ado\site\
4      PLUS:  C:\Users\lv39\ado\plus\
5  PERSONAL:  C:\Users\lv39\ado\personal\
6  OLDPLACE:  c:\ado\
```

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:

```
1  adopath
```

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

The path at work

To look for a command, 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.

```
1  which reghdfe
```

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

Where are packages installed?

When we install a package (`net install`, `ssc install`)[1], only one of the (`sysdir`) paths is relevant: `PLUS`.

```
1    [1]   (BASE)       "C:\Program Files\Stata1
2    [2]   (SITE)       "C:\Program Files\Stata1
3    [3]                "."
4    [4]   (PERSONAL)   "C:\Users\lv39\ado\perso
5    [5]   (PLUS)       "C:\Users\lv39\ado\plus/
6    [6]   (OLDPLACE)   "c:\ado/"
```

Installing packages

```
1  ssc install reghdfe
2  which reghdfe
```

```
1  . ssc install reghdfe
2  checking reghdfe consistency and verifying not already install
3  installing into C:\Users\lv39\ado\plus\...
4  installation complete.
5
6  . which reghdfe
7  C:\Users\lv39\ado\plus\r\reghdfe.ado
8  *! version 6.12.3 08aug2023
```

Using environments in Stata

But the `(PLUS)` directory can be manipulated

```
 1  * Set the root directory
 2  global rootdir : pwd
 3  * Define a location where we will hold all
 4  global adodir "$rootdir/ado"
 5  * make sure it exists, if not create it.
 6  cap mkdir "$adodir"
 7  * Now let's simplify the adopath
 8  * - remove the OLDPLACE and PERSONAL paths
 9  * - NEVER REMOVE THE SYSTEM-WIDE PATHS - ba
10  adopath - OLDPLACE
11  adopath - PERSONAL
12  * modify the PLUS path to point to our new
13  sysdir set PLUS "$adodir"
14  adopath ++ PLUS
15  * verify the path
16  adopath
```

Using environments in Stata

```
1  * Set the root directory
2  global rootdir : pwd
3  * Define a location where we will hold a
4  global adodir "$rootdir/ado"
5  * make sure it exists, if not create it
6  cap mkdir "$adodir"
7  * Now let's simplify the adopath
8  * - remove the OLDPLACE and PERSONAL pat
9  * - NEVER REMOVE THE SYSTEM-WIDE PATHS -
10 adopath - OLDPLACE
11 adopath - PERSONAL
12 * modify the PLUS path to point to our p
13 sysdir set PLUS "$adodir"
14 adopath ++ PLUS
15 * verify the path
16 adopath
```

```
1  . adopath
2    [1]  (PLUS)      "C:\Users\lv39\Documents/PROJECT123/ado/"
3    [2]  (BASE)      "C:\Program Files\Stata18\ado\base/"
4    [3]  (SITE)      "C:\Program Files\Stata18\ado\site/"
5    [4]               "."
```

Using environments in Stata

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

```
1  which reghdfe
```

```
1  . which reghdfe
2  command reghdfe not found as either built-i
3  r(111);
```

Using environments in Stata

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.

Previously:

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

```
. adopath
  [1]  (PLUS)      "C:\Users\lv39\Documents/PROJECT
  [2]  (BASE)      "C:\Program Files\Stata18\ado\ba
  [3]  (SITE)      "C:\Program Files\Stata18\ado\si
  [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.

Installing precise versions of Stata 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.

Package repositories

Most package repositories are versioned:

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

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

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.

Takeaways

- ☑ your code runs without problem, after all the debugging.
- ☑ your code runs without manual intervention, and with low effort
- ☑ it actually produces all the outputs
- ☑ your code generates a log file that you can inspect, and that you could share with others.
- ☑ ❓ it will run on somebody else's computer

Secrets in the code

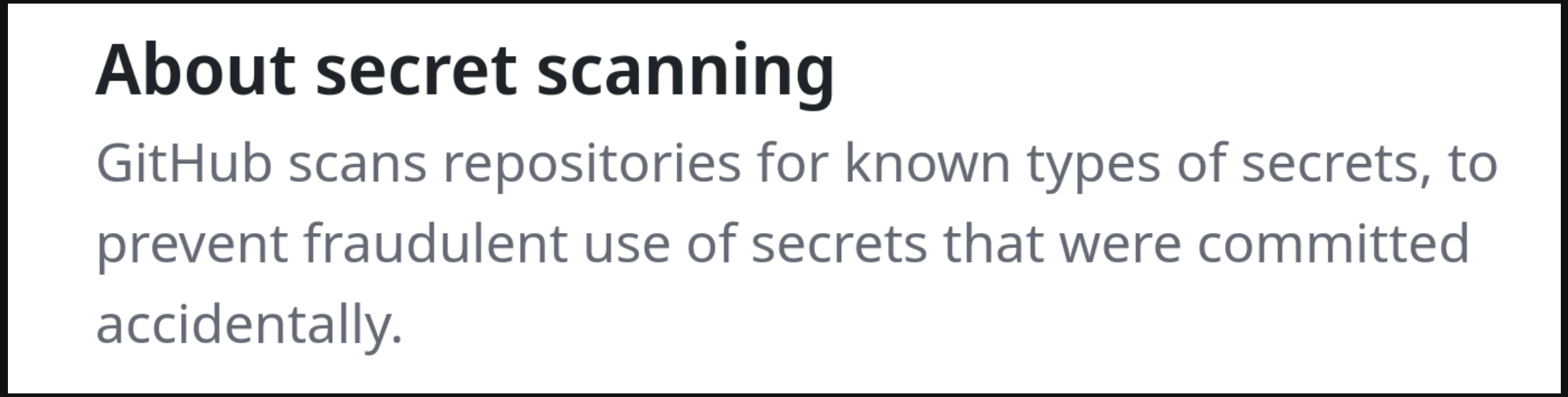
What are secrets?

- API keys
- Login credentials for data access
- File paths (FSRDC!)
- Variable names (IRS!)

Standard practice

Store secrets in environment variables or files that are not published.

Some services are serious about this



Github secret scanning

Where to store secrets

- **environment variables**
- “dot-env” files (Python), “Renviron” files (R)
- or some other clearly identified file in the project or home directory

Environment variables

Typed interactively (here for Linux and Mac)

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

(this is **not** recommended)

Storing these in files

Same syntax used for contents of “dot-env” or “Renviron” files, and in fact `bash` or `zsh` startup files (`.bash_profile`, `.zshrc`)

Using In 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")
```

Use the variables defined in `.Renviron`:

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

Using In Python

Loading regular environment variables:

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

Loading with `dotenv`

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

Using in 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

```
1  //============ non-confidential parameters =========
2  include "config.do"
3
4  //============ confidential parameters =============
5  capture confirm file "$code/confidential/confparms.do"
6  if _rc == 0 {
7      // file exists
8      include "$code/confidential/confparms.do"
9  } else {
10     di in red "No confidential parameters found"
11 }
12 //============ end confidential parameters =========
```

Confidential code?

What is confidential code, you say?

- 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”).

What is confidential code, you say?

- Your code contains the **random seed you used to anonymize** the sensitive identifiers. This might allow to reverse-engineer the anonymization, and is not a good idea to publish.

What is confidential code, you say?

- You used a **look-up table hard-coded** in your Stata code to anonymize the sensitive identifiers (`replace anoncounty=1 if county="Tompkins, NY"`).

A **really bad idea**, but yes, you probably want to hide that.

What is confidential code, you say?

- 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.

What is confidential code, you say?

- You have adhered to disclosure rules, but for some reason, the precise minimum cell size is a confidential parameter.

What is confidential code, you say?

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?

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.

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

Example

Only one line that does not contain “confidential” information.

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

Do not do this

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

```
1  set seed NNNNN
2  use <removed vars> county using "<removed path>", clear
3  gen logprofit = log(XXXX)
4  by county: collapse (count)  n=XXXX (mean) logprofit
5  drop if n<XXXX
6  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.

Better

Simply replacing 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
```

Better

and this 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.

Best

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

Best

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 ======================
```

Best

Main file `main.do` (continued)

```
// ::::  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
```

Best

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

```
1  //============ confidential parameters =============
2  global confseed    12345
3  global confpath    "/data/economic/cmf2012"
4  global confprofit  q2f
5  global confemploy  q3e
6  global confmincell 10
7  //============ end confidential parameters =========
```

Best

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

```
1  //============ confidential parameters =============
2  // Copy this file to $code/confidential/confparms.do and edit
3  global confseed    XXXX    // a number
4  global confpath    "XXXX"  // a path that will be communicated to you
5  global confprofit  XXX     // Variable name for profit T26
6  global confemploy  XXX     // Variable name for employment T26
7  global confmincell XXX     // a number
8  //============ end confidential parameters =========
```

Best replication package

Thus, the replication package would have:

```
1 ...
2 code/main.do
3 README.md
4 include/confparms_template.do
5 releasable/figure1.dta
6 releasable/figure1.pdf
```

Avoiding confidential data in your code

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!

- You may now have to redact the code before releasing

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 ...
```

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*.

Better solution

```
1
2 if (!file.exists("data/confidential/names_mapping.csv")) {
3     names_confidential %>%
4         left_join(read_csv("data/confidential/names_mapping.csv"), by = "na
5         # replace name with name_alt if the latter is not NA
6         mutate(name = if_else(!is.na(name_alt), name_alt, name)) %>%
7         # drop the name_alt column
8         select(-name_alt) -> names_clean
9 }
```

Note

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

Tutorial example

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

Wrapping it all up

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

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

Things to remember

Run it all again, top to bottom!

Things to remember

- 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!

End

Now you wait for the replicators to show up!

Appendix

Keeping on top of provenance

- Licenses
- Streamlining for reproducibility

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?

What is a license?

> A license (licence) is an official permission or permit to do, use, or own something (as well as the document of that permission or permit).[2] [3]

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”

Can we re-publish the file?

Downloading via code

Easiest:

Stata

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

Why not?

- will it be there in two months? in 6 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

```
download.file(url="$URL",destfile="(outputfile)")
```

We will get to even better methods a bit later

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)

Link

Step 1: Stata, R [4]

Links

Guidance

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

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/

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://doi.org/10.3886/E162581V1

- This presentation’s source:
- Licensed under 

Footnotes

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.

2. Cambridge Dictionary

3. Wikipedia

4. 🔒Tag: stage1