THE WORLD BANK
IBRD • IDA | WORLD BANK GROUP
Development Economics • Impact

DIME
ANALYTICS

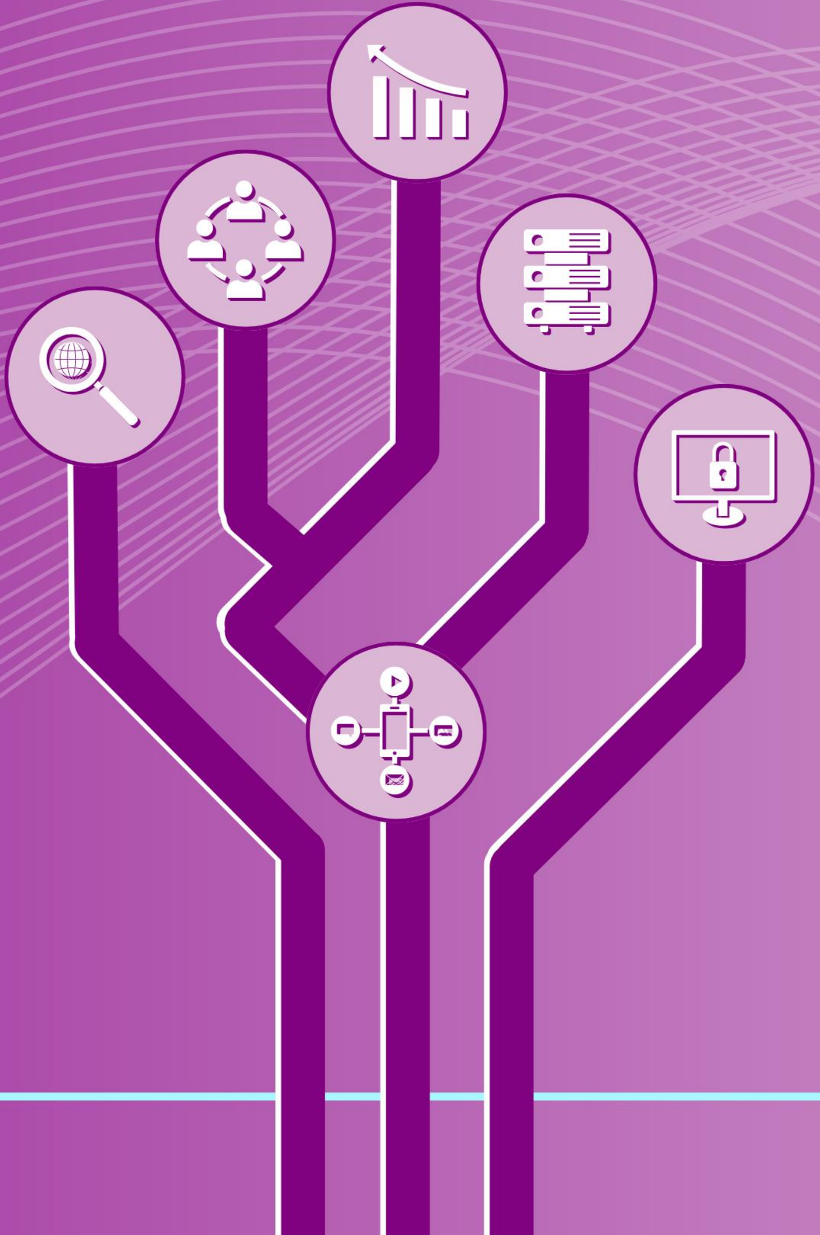
Stata Conference 2024

reproot & repado

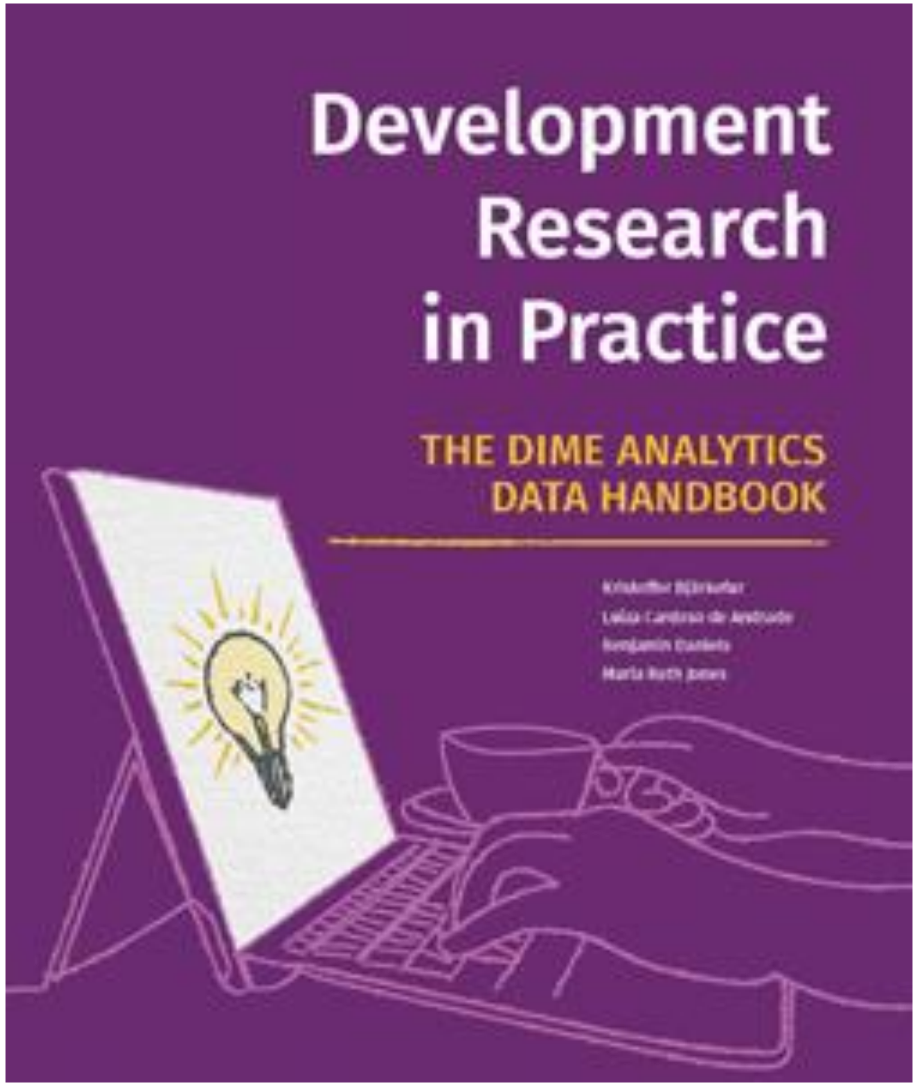
Kristoffer Bjärkefur

DIME Analytics & LSMS

World Bank



DIME Analytics



Development Research in Practice

THE DIME ANALYTICS DATA HANDBOOK

Kristoffer Bjärkefur
Luíza Cardoso de Andrade
Benjamin Daniels
Maria Ruth Jones

DIME Analytics

- Team page with resources

Development Research in Practice

- Hard copy
- PDF
- E-book

repkit

The commands introduced in this presentation are all part of the package **repkit**, which is available through SSC.

Installation :	`ssc install repkit`
Documentation :	https://worldbank.github.io/repkit
Source code :	https://github.com/worldbank/repkit

reproot

Automatically set root paths
in multi-rooted projects

Definitions

$$\texttt{"}\underbrace{\texttt{C:\textbackslash Users\textbackslash WB462869\textbackslash Dropbox\textbackslash Projects/}\overbrace{\texttt{ProjectA}}^{\text{Root folder}}}_{\text{Root path}}\underbrace{\texttt{/code/analysis/table1.do}}_{\text{Project path}}\texttt{"}$$

- The **root path** is the location of a project's **root folder** on a specific computer
- The **root path** is almost always unique on each computer
- The **project path** is a relative path from the project's **root folder**

An efficient system for managing root paths

is required for effective collaboration.

Overview of existing system

Name	Short description	Root path set by action outside code	User-Specific Manual Project Setup	Support for Multi-Rooted Projects
Implied working directory	Have all files in a single folder. When opening a file, the working directory is set to that file's location. Then all files are accessed in the code with just the file name.	Yes	No	No
Explicit working directory	Set root path using command **cd**. Then all paths in the code are just the project path	No	Yes – change the **cd** path each session	Not with a single line with **cd**
Stata project files	Opening the project file first. That sets the working directory to the location of that file. Then all paths in the code are just the project path.	Yes	No	No
Root Globals	A main file with if-blocks sets the specific root path for each user in globals. Then all paths in the code are the root global + project path.	No	Yes – once per project	Yes
setroot	A command that looks for a "root file" by searching parent folder starting from the working directory. When that file is found, it's location is stored in a root global. Then all paths in the code are the root global + project path.	In rare cases	No	No

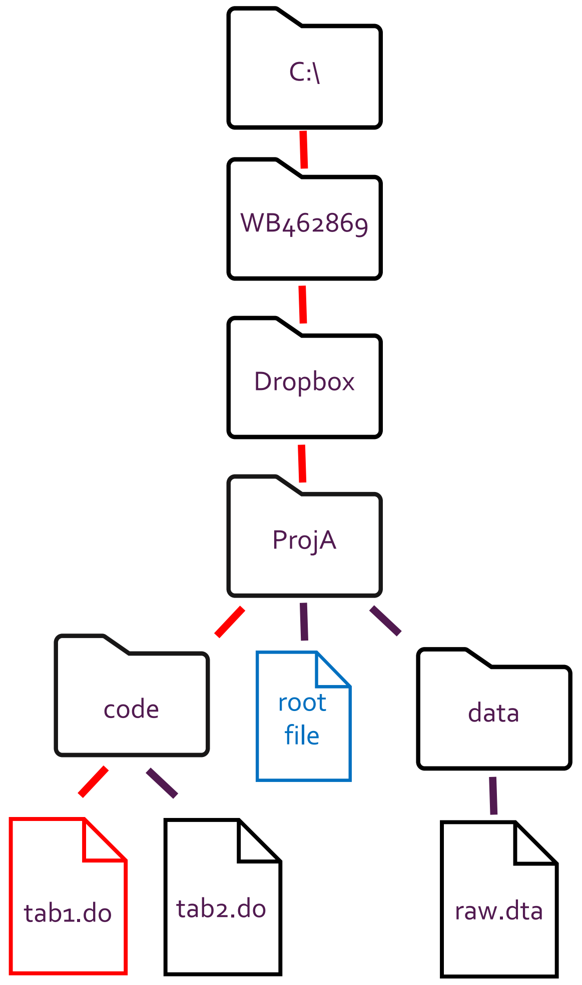
Root globals

- **c(username)** is used for code to know which users is running the code
- Each user defines the root path in a projects main-file
- A path **"${root}/data/raw.dta"** evaluates correctly regardless of where each user saved the **"ProjectA"** folder on their computer

```
* Kristoffer's root path
if "`c(username)'" == "wb462869" {
    global root "C:\Users\wb462869\Dropbox\Projects\ProjectA"
}
* Ben's root path
if "`c(username)'" == "bbdaniels" {
    global root "/Users/bbdaniels/Dropbox/ProjectA"
}
```

How setroot finds root paths

"C:\WB462869\Dropbox\ProjA/code/table1.do"



- **tab1.do** has this code at the top of the file:

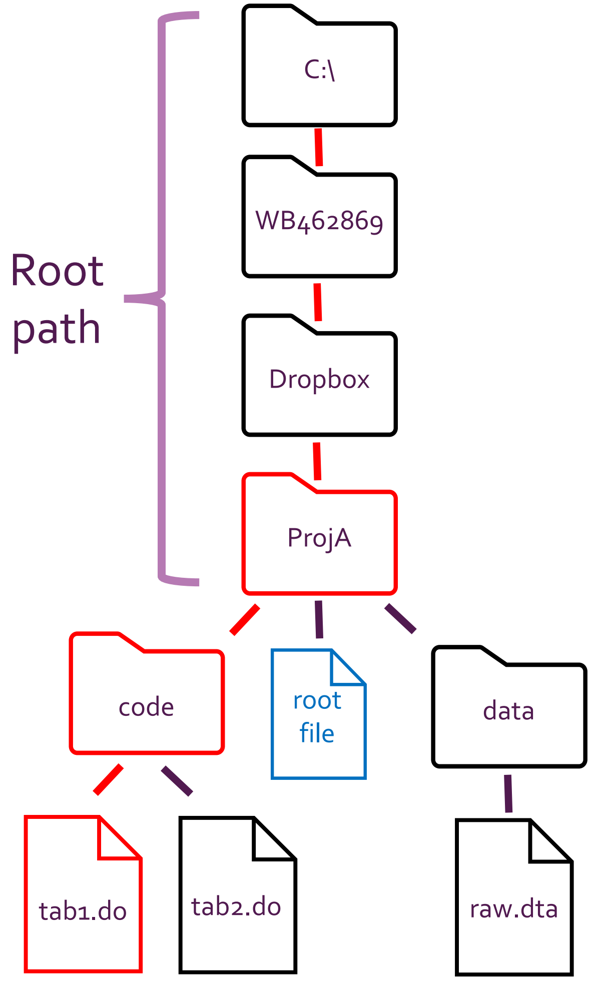
```
setroot
use "${root}/data/raw.dta"
```

- If Stata is opened by clicking the file **tab1.do**, then the working directory is set to **"C:\WB462869\Dropbox\ProjA\code"**

Reference: Sergio Correia, 2023, setroot

How setroot finds root paths

"C:\WB462869\Dropbox\ProjA/code/table1.do"



- **tab1.do** has this code at the top of the file:

```
setroot
use "${root}/data/raw.dta"
```

- If Stata is opened by clicking the file **tab1.do**, then the working directory is set to **"C:\WB462869\Dropbox\ProjA\code"**
- When running **tab1.do**:
  - **setroot** first looks for the root file in: **"C:\WB462869\Dropbox\ProjA\code"**,
  - then in **"C:\WB462869\Dropbox\ProjA"** where it finds the root file
  - then sets global **root** to **"C:\WB462869\Dropbox\ProjA"**
- This way, **"${root}/data/raw.dta"** evaluates to the correct path regardless of were **"ProjA"** is located on that computer

Reference: Sergio Correia, 2023, setroot

Multi-rooted projects

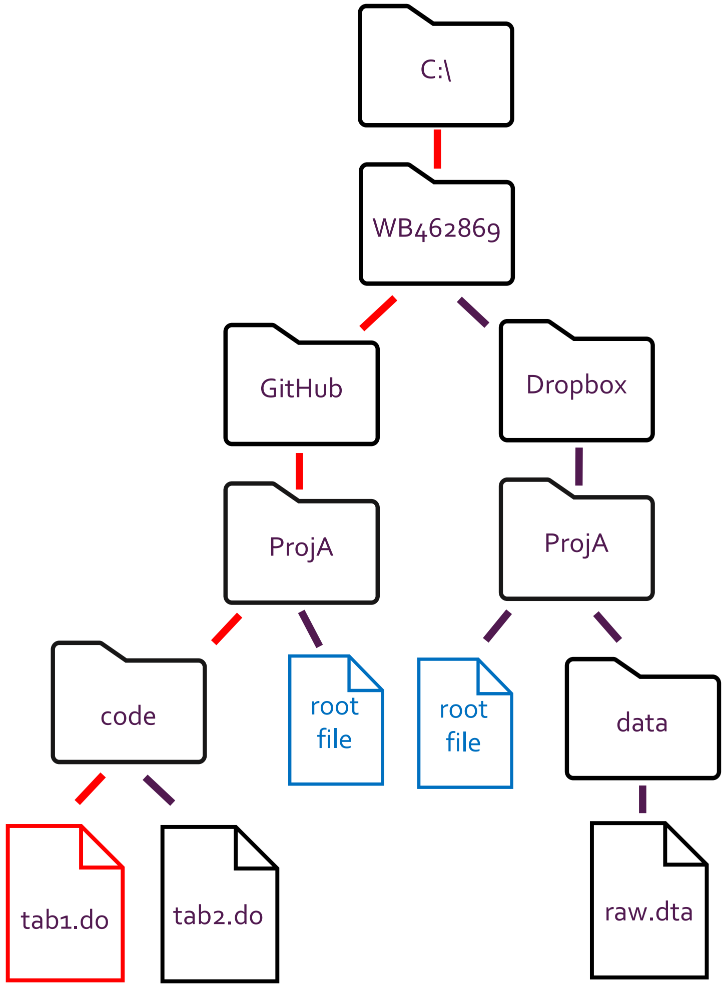
- Some projects have files in more than one root. For example:
  - **Sync services (Dropbox/OneDrive etc.)**
  - **GitHub**
  - **Network drives**
  - **External hard drives**
  - **Encrypted folders**
- We call such projects **multi-rooted**
- Multiple roots cannot be found in a bottom-up search along the working directory, only one root can be found

Multi-Rooted - Root Globals

- This can be solved in root globals by setting multiple roots
- However, the amount of manual setup increases with number of roots and number of projects one collaborate on

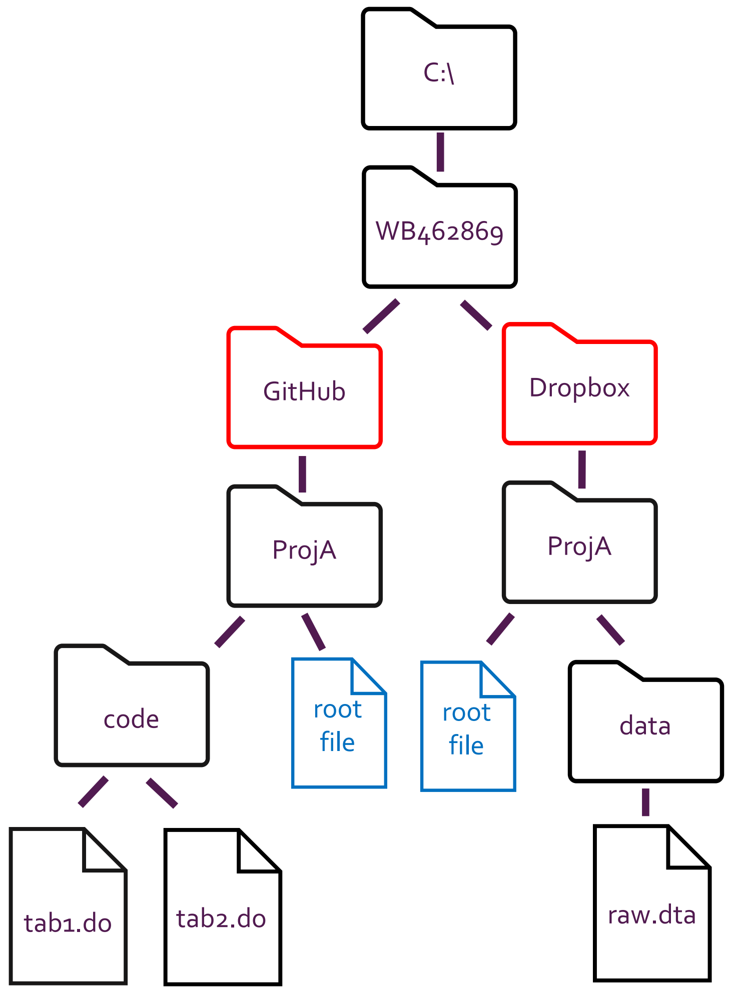
```
* Kristoffer's root path
if "`c(username)'" == "wb462869" {
    global data "C:\Users\wb462869\Dropbox\Projects\ProjectA"
    global code "C:\Users\wb462869\GitHub\ProjectA"
}
* Ben's root path
if "`c(username)'" == "bbdaniels" {
    global data "/Users/bbdaniels/Dropbox/ProjectA"
    global code "/Users/bbdaniels/GitHub/ProjectA"
}
```

Multi-Rooted - Root Files



- Issues:
  - Multiple root files can typically not be found through a bottom-up search along a single working directory path
  - The working directory might be set such that no root file is found
- Potential approach:
  - Do a top-down search from root of computer
  - Issues -> Prohibitively slow
  - Can we make assumptions such that the search is fast enough?

Multi-Rooted - Root Files



- Assumptions:
  - The possible locations for a root files are few and easy to define: DropBox, GitHub, network drive etc.
  - The root file is located "near" (few sub-folders away) these possible root locations
- **reproot** solution:
  - Define *starting points* and *search depth*
  - Do an exhaustive recursive top-down search from the starting points and stop the recursion at the search depth

reproot – set-up

Set-up:

- **Once per computer:** Set up an *environment file* with the possible root locations
- **One user once per project:** Set up *root files* in each root

Environment file – reproot-env.yaml

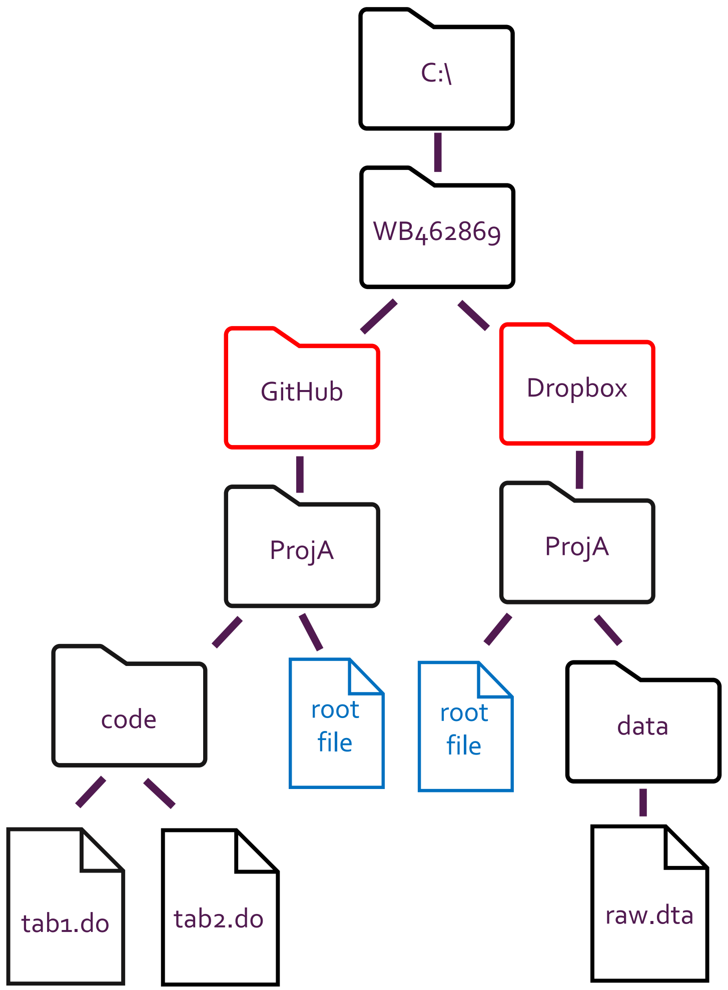
```
recursedepth: 4
paths:
    - "2:C:\Users\WB462869\github"
    - "C:\Users\WB462869\Dropbox"
    - "C:\Users\WB462869\OneDrive - WBG"
skipdirs:
    - ".git"
```

Root files – reproot.yaml

```
project_name : proj-a
root_name    : code
```

```
project_name : proj-a
root_name    : data
```

reproot – set-up



Environment file – reproot-env.yaml

```
recursedepth: 4
paths:
    - "2:C:\Users\WB462869\github"
    - "C:\Users\WB462869\Dropbox"
    - "C:\Users\WB462869\OneDrive - WBG"
skipdirs:
    - ".git"
```

Root files – reproot.yaml

```
project_name : proj-a
root_name    : code
```

```
project_name : proj-a
root_name    : data
```

reproot - output

```
* Use reproot to get root paths
reproot, project("proj-a") roots("code data") prefix("prja_")
```

```
Searching folder: C:/Users/WB462869/github, with folder depth: 2

In this search directory, 51 directories were searched in 0.01 seconds, and 5 unique reproot root(s) were found.

Searching folder: C:\Users\WB462869\github\repkit\src\tests\reproot, with folder depth: 4

In this search directory, 7 directories were searched in 0.00 seconds, and 4 unique reproot root(s) were found.

Searching folder: C:\Users\WB462869\Dropbox, with folder depth: 4

In this search directory, 311 directories were searched in 0.10 seconds, and 1 unique reproot root(s) were found.

In total, 369 directories were searched in 0.12 seconds, and 10 unique reproot root(s) were found.

Root code was set to C:/Users/WB462869/github/ProjectA using global ${prja_code}

Root data was set to C:\Users\WB462869\Dropbox/Projects/ProjectA using global ${prja_data}
```

Where to store the environment file?

- If we need the environment file to find files, how do we find the environment file?
- Any hardcoded path simply brings back the problem we are trying to solve
- There is one special path that is hardcoded in code, but dynamic in practice: **"~"**
- **reproot** has a utility that helps you set up the environment file in the correct location

Environment file – reproot-env.yaml

```
recursedepth: 4
paths:
    - "2:C:\Users\WB462869\github"
    - "C:\Users\WB462869\Dropbox"
    - "C:\Users\WB462869\OneDrive - WBG"
skipdirs:
    - ".git"
```

```
* Find your home folder
cd ~
local home_folder "`c(pwd)'"
di "`home_folder'"
```

reproot - usage

```
* Use reproot to get root paths
reproot, project("proj-a") ///
  roots("code data") prefix("prja_")
* Load data
use "${prja_data}/data/raw.dta"
* Run analysis
do  "${prja_code}/code/tab1.do"
do  "${prja_code}/code/tab2.do"
```

Root files found

```
project_name : proj-a
root_name    : code
```

```
project_name : proj-a
root_name    : data
```

```
project_name : proj-b
root_name    : code
```

```
project_name : abcimpact
root_name    : picture
```

- In this example, **reproot** searches for root files matching the project and the roots
- This search takes seconds
- No search is done if the globals were already found – i.e. the global is not empty
- We recommend using prefix such that **reproot** does not confound a root of another project with this root

Comparison to existing system

Name	Short description	Root path set by action outside code	User-Specific Manual Project Setup	Support for Multi-Rooted Projects
Stata project	Open a special file	Yes	No	No
Root Globals	User-specific hard coded roots in main file	No	Yes – once per project	Yes
setroot	Search bottom-up for root file along working directory	In rare cases	No	No
reproot	Search top-down for root files according to computer specific settings	No	No	Yes

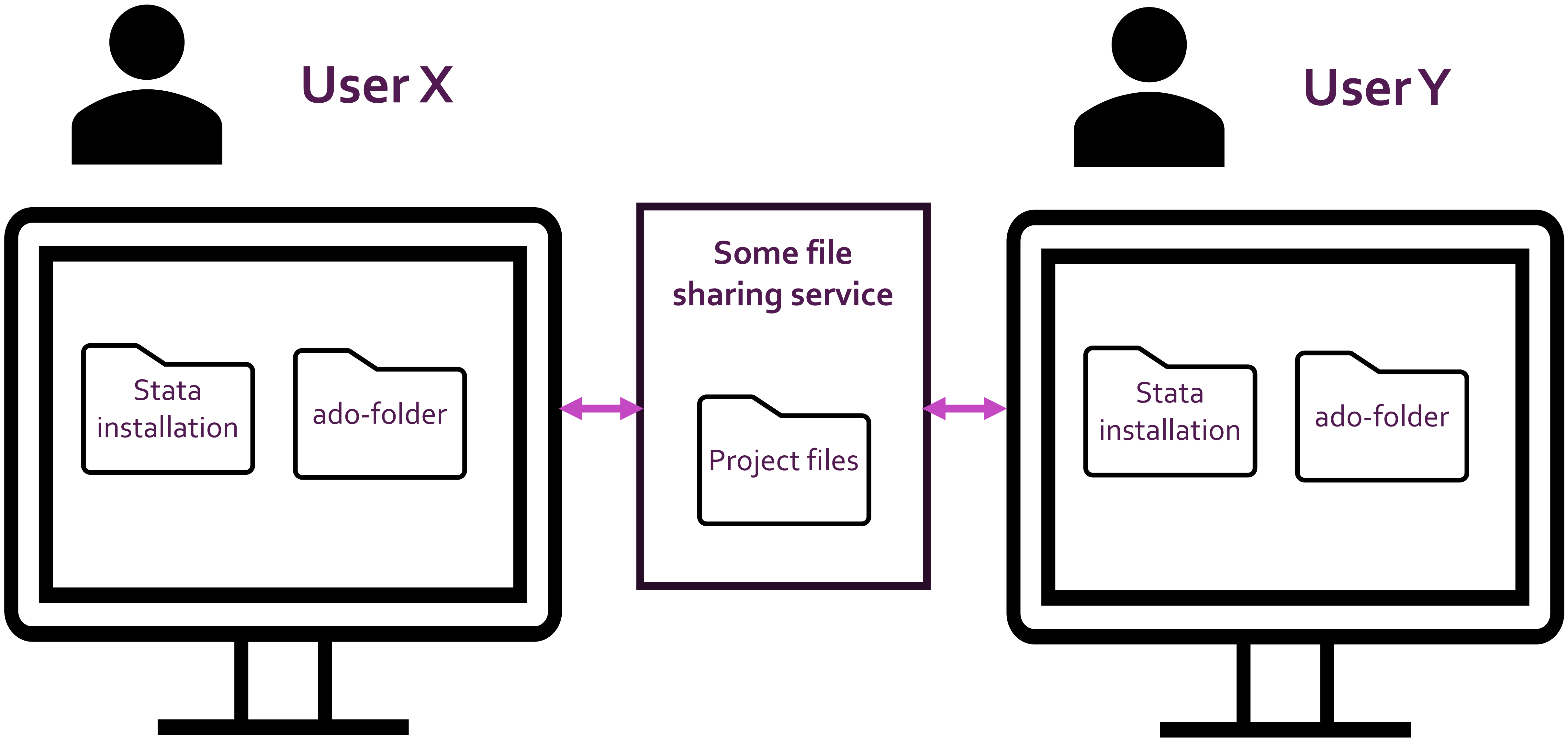
repado

A very exact and reliable way of managing project specific dependencies

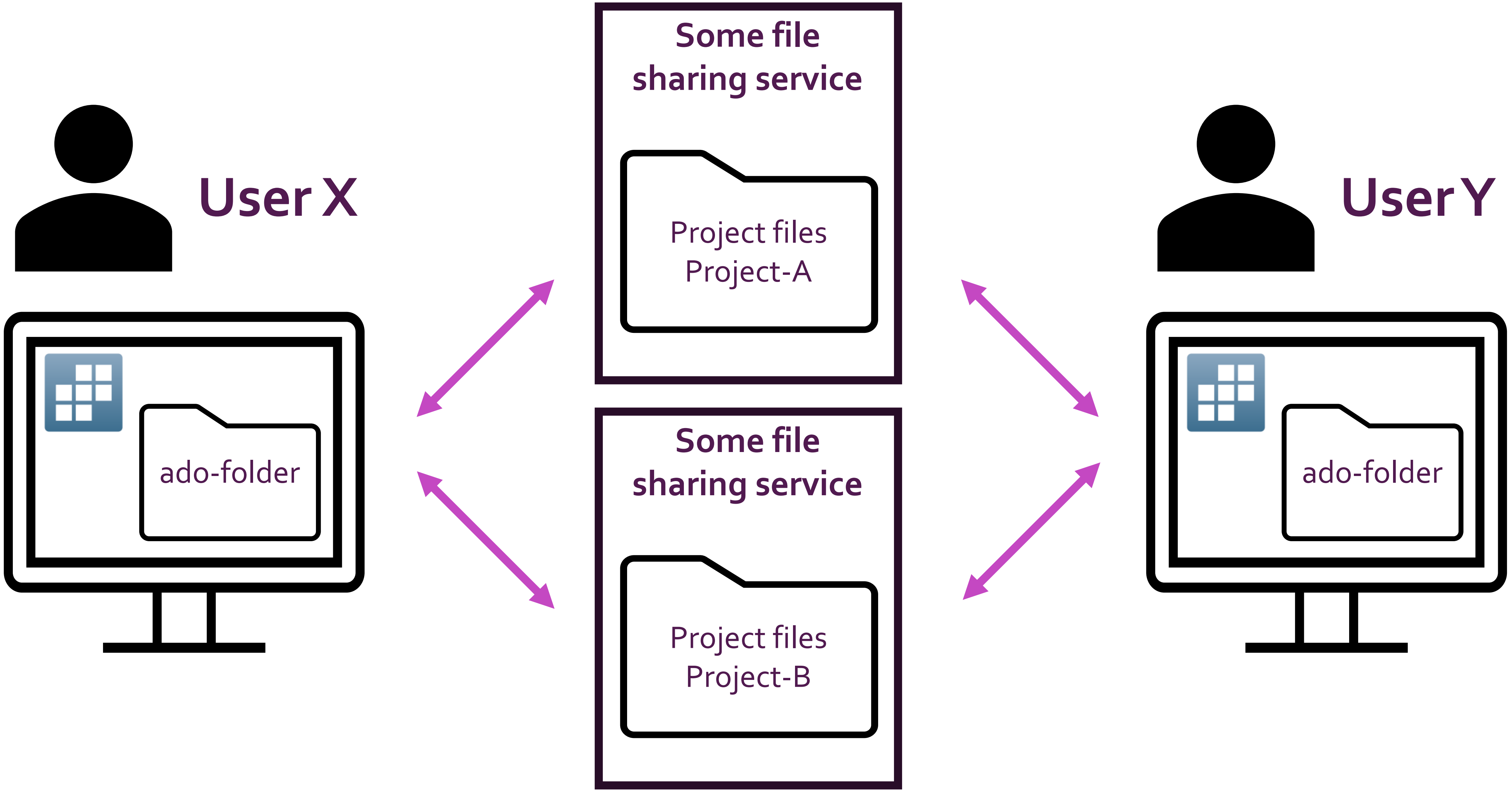
repado - motivation

- To accurately reproduce a project, we need ***the exact same data*** and ***the exact same code***
- "*the exact same code*" - this includes:
  - The exact same version of *the project code*
  - The exact same version of *Stata's build in commands*
  - The exact same version of *community contributed ado-files*
- Best practices for sharing project code is out of scope of this presentation
- Stata has version control for built-in commands using **version**
- **SSC and other sources for community contributed commands are not versioned**

Common practice



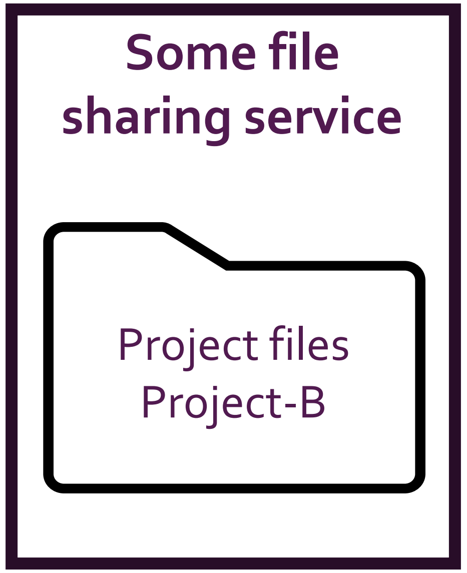
Common practice



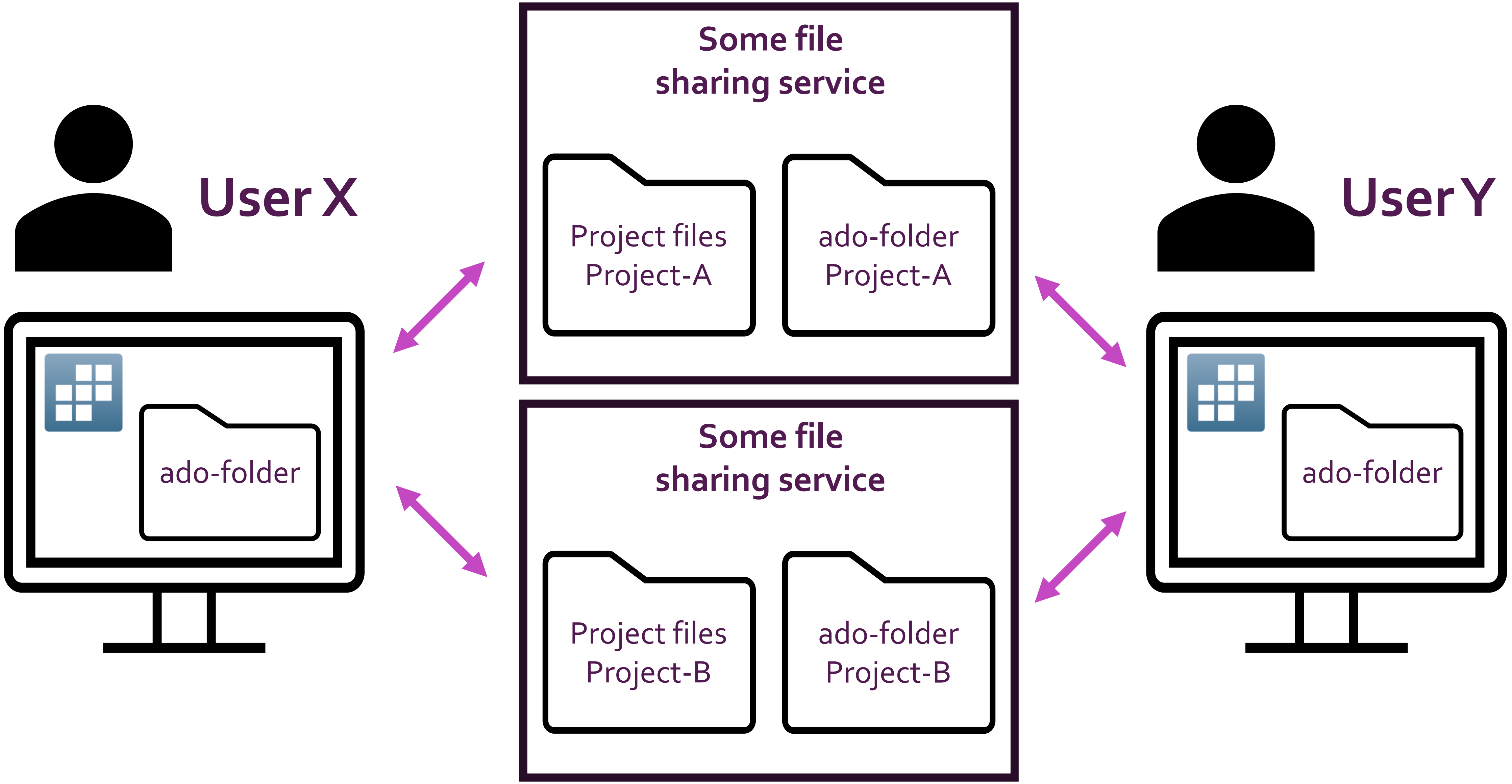
Issues

- Project-A and Project-B are likely to have different dependencies – different packages *and* different versions of those packages
- To satisfy "*the exact same code*" requirement, we need to make sure all users has the same packages and the same version of them
- Always using the latest version does not always work - to future proof reproducibility - we need to make sure the exact version of those packages are available

Some file sharing service
Project files Project-A

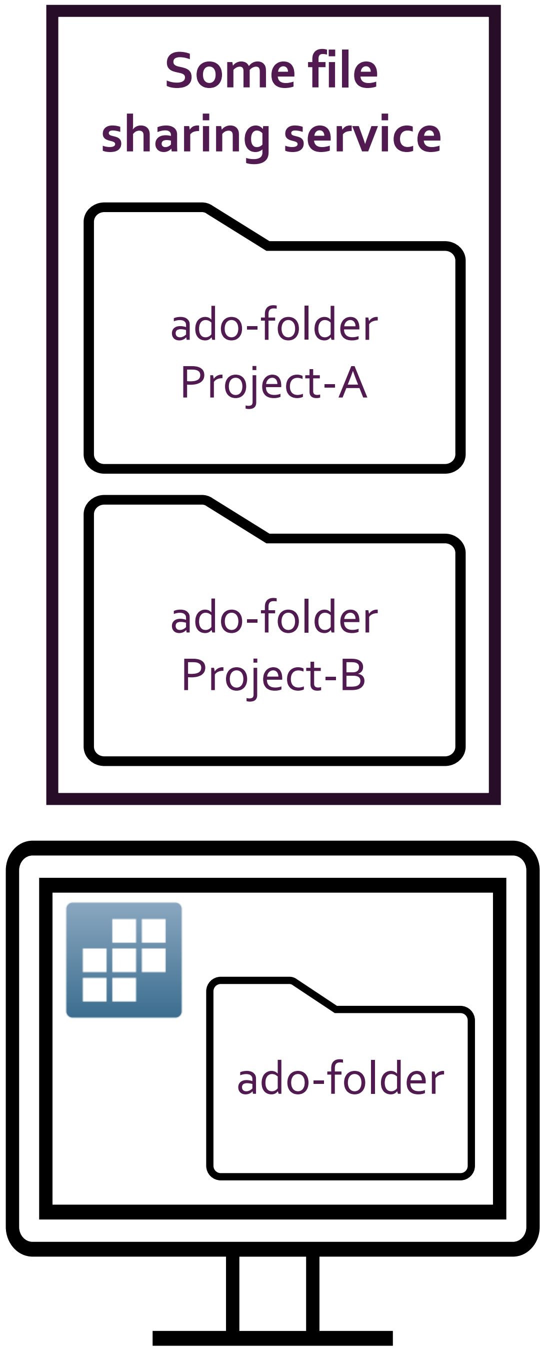


Best practice – share project dependencies



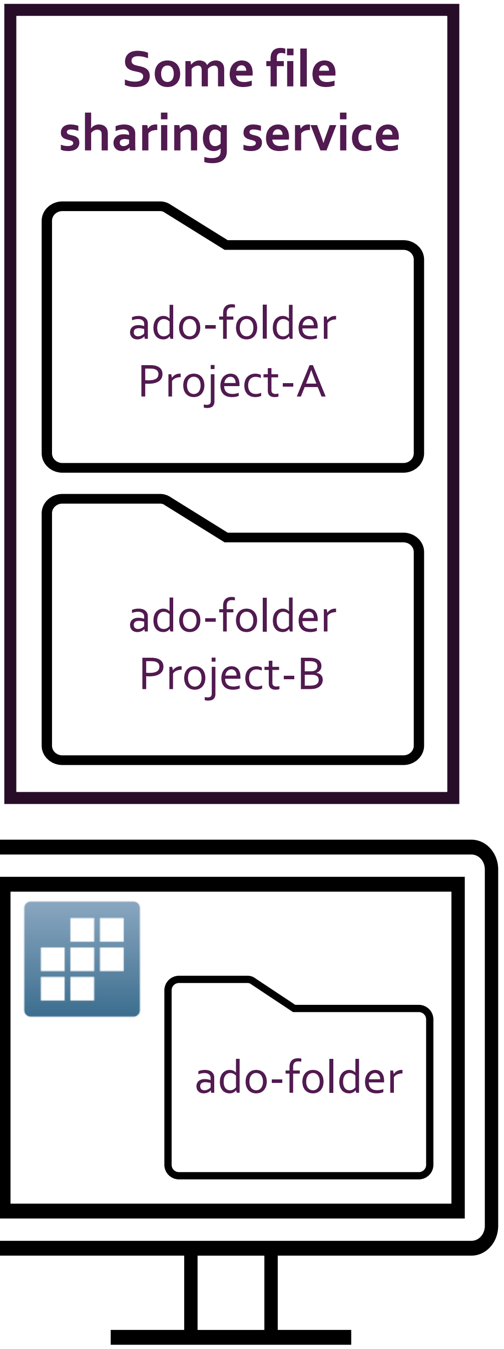
How to use multiple ado-folders

- Installing a package is simply saving files in a structured way – this can be done in multiple locations
- Stata gives us access to manage the ado-paths where it expects installed packages to be found – we can point this to a shared location
- When working in a project with dedicated ado-folder, we should only use that ado-folder



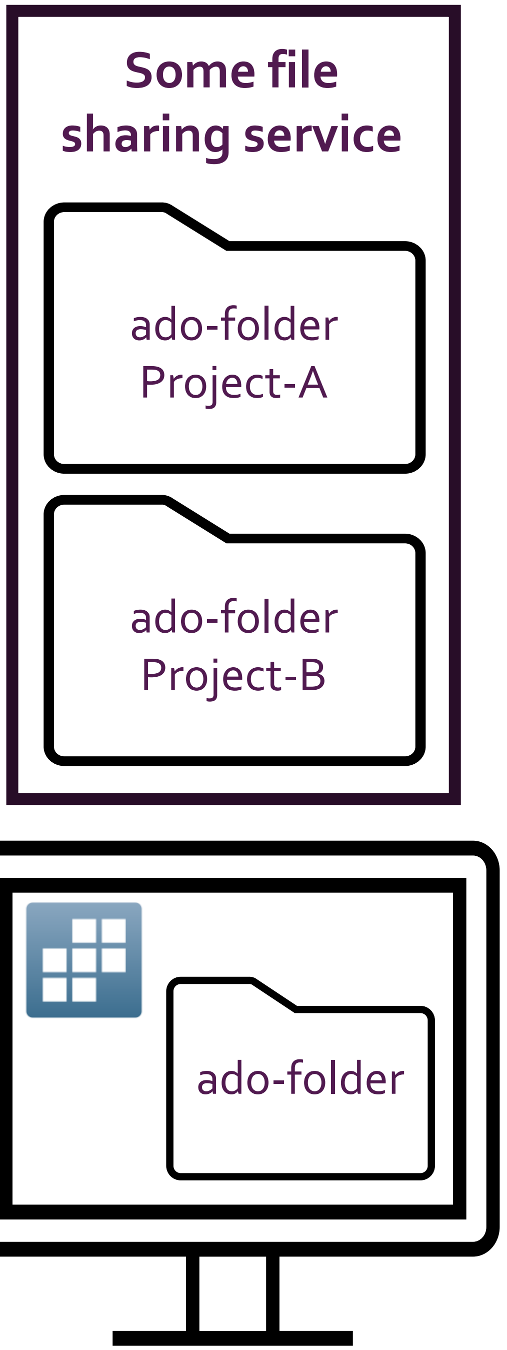
How to install in multiple ado-folders

- Commands installed with **net install** (which **ssc install** uses) packages are installed in the PLUS folder
- We can set the PLUS folder to whichever ado-folder we want – we call this: "*the current ado-folder*"
- If the "*current*" ado-folder is shared, then installing or updating a package takes effect for all users that folder is shared with



How to manage multiple ado-folders

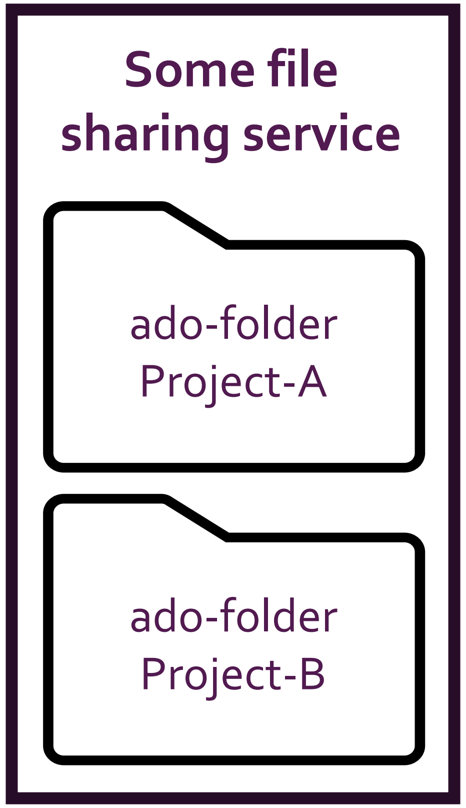
- "*This sounds hard and technical. Maybe I can learn to do this, but my team would never adopt this…*"
- There are many approaches for how to set this manually by understanding the details – we abstracted that away into a command: **repado**



How to manage multiple ado-folders

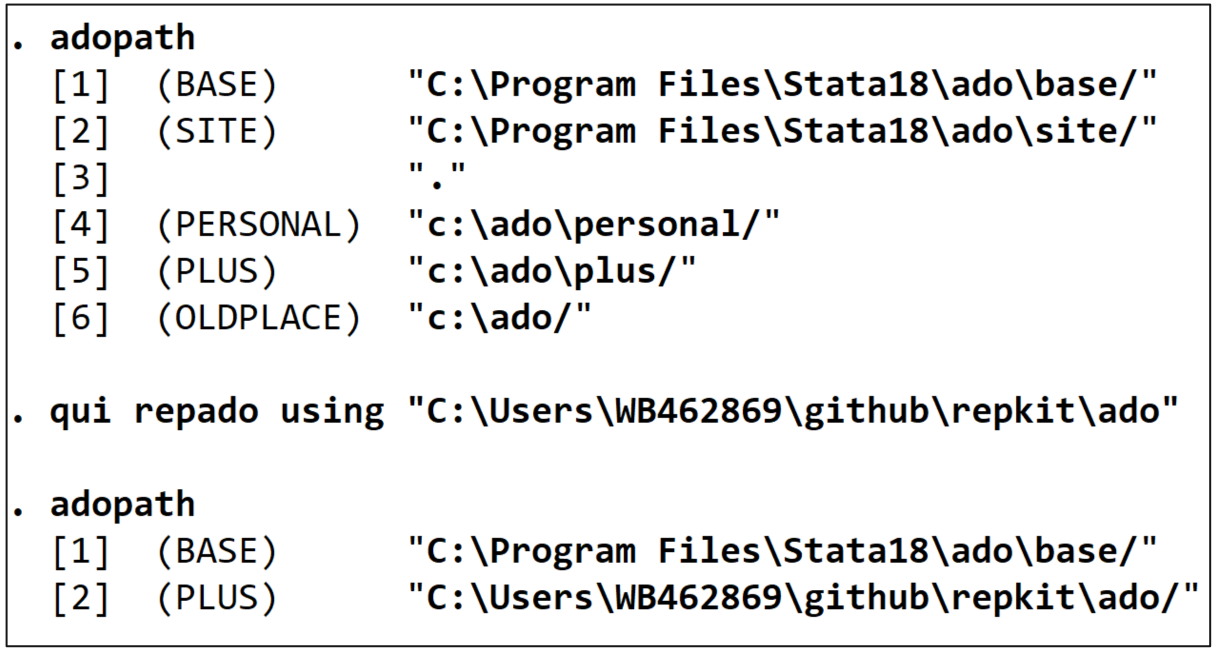
```
* Use the ado-environment from ProjectA
repado using "${prja_code}/code/ado"

* Use the ado-environment from ProjectB
repado using "${prjb_code}/code/ado"
```

- Create a folder in your project and set the ado-path to that folder – can be called anything but **ado** is a common name
- This setting has the same scope as a global – i.e., the default ado-path is reset when restarting Stata
- After using **repado** packages will be installed in the project ado-folder as it is *the current ado-folder*



Some technical details

- In addition to adding the project ado-folder to the ado-paths, the default behavior of **repado** removes all other paths (apart from **BASE**)
- This means that only Stata's built-in commands and the commands installed in that project's ado-folder are available



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

. qui repado using "C:\Users\WB462869\github\repkit\ado"

. adopath
  [1]  (BASE)      "C:\Program Files\Stata18\ado\base/"
  [2]  (PLUS)      "C:\Users\WB462869\github\repkit\ado/"
```

More than just setting the PLUS path

- Common way to set the PLUS is:

```
* Use the ado-environment from ProjectA
sysdir set PLUS "${prja_code}/code/ado"
```

- This is what repado does under-the-hood, but repado makes it simple and does more:
  - Keeping only BASE and PLUS guarantees that all packages required for a project is in a project's ado-folder
  - Rebuild the mata list of libraries: **mata: mata mlib index**

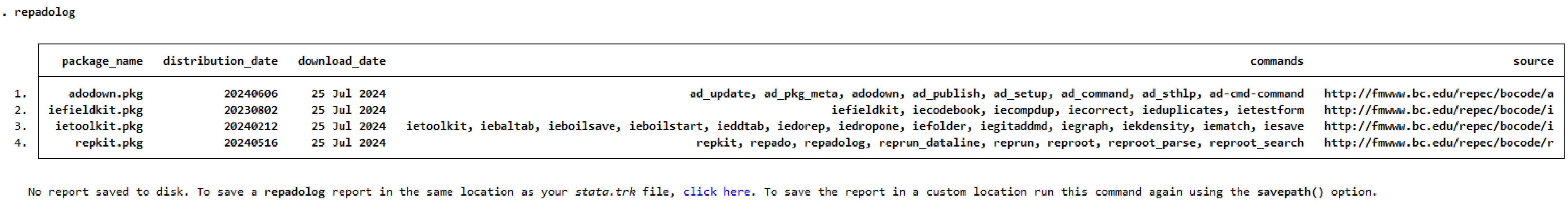
Licensing

- The repado workflow requires that you share someone else's ado-files
- If you share this publicly, make sure you are allowed to do so – you typically are, but please make sure
- SSC commands are shared under the GPL which allows this. Many commands shared on GitHub are shared under MIT or CC licenses which allows this. Otherwise email the author and ask.

repadolog

Output a report of what is installed in a project's ado-folder

repadolog



```
. repadolog

      package_name   distribution_date   download_date                                                                                      commands                         source

 1.    adodown.pkg            20240606     25 Jul 2024                                ad_update, ad_pkg_meta, adodown, ad_publish, ad_setup, ad_command, ad_sthlp, ad-cmd-command   http://fmwww.bc.edu/repec/bocode/a
 2. iefieldkit.pkg            20230802     25 Jul 2024                                                    iefieldkit, iecodebook, iecompdup, iecorrect, ieduplicates, ietestform   http://fmwww.bc.edu/repec/bocode/i
 3.  ietoolkit.pkg            20240212     25 Jul 2024   ietoolkit, iebaltab, ieboilsave, ieboilstart, ieddtab, iedorep, iedropone, iefolder, iegitaddmd, iegraph, iekdensity, iematch, iesave   http://fmwww.bc.edu/repec/bocode/i
 4.     repkit.pkg            20240516     25 Jul 2024                                 repkit, repado, repadolog, reprun_dataline, reprun, reproot, reproot_parse, reproot_search   http://fmwww.bc.edu/repec/bocode/r

  No report saved to disk. To save a repadolog report in the same location as your stata.trk file, click here. To save the report in a custom location run this command again using the savepath() option.
```

- **repadolog** is a command that outputs what packages you have installed in your current ado-folder
- It parses through the *stata.trk* folder in the current ado-folder and makes it easier for a human to consume
- It is possible to export the report to a csv-file that is shared in a reproducibility package

Thank you!

Kristoffer Bjärkefur - kbjarkefur@worldbank.org
DIME Analytics – dimeanalytics@worldbank.org