Programming Workflow Guide for Research Projects

Ganesh Karapakula

October 2022

Abstract

This document contains some resources and guidelines on programming workflow and practices for producing reproducible research outputs. This guide first discusses how to organize workflow using a version control system called Git, before addressing some basic aspects of reproducible code. This document also provides R- and Stata-based guidelines for writing customized statistical programs and for automating creation of customized tables in TeX format. Other aspects of programming workflow are also discussed. This guide is intended for readers with at least some basic knowledge of R, Stata, and TeX, but no prior knowledge of Git is assumed. A folder containing code templates for a worked example accompanies this guide.

1 Organizing Workflow Using Git

Imagine a research project involving several collaborators who all may be involved in preparing the manuscript and writing code (to generate tables and graphs to be included in the manuscript). Over the course of the project, it is useful to keep track of the various versions of the project folder and to have a record of which files were modified by whom, how, and when. In the old days, collaborators would exchange files either via email or through a shared directory, and they would try to archive each version of each file (e.g., by including the date and author's initials in the file's name). However, this method is outdated because it is cumbersome, inefficient, and error-prone. The modern way to organize project workflow is through version control software, which records changes to a folder's contents over time, allowing one to review edits made over time and also to revert files (or even the entire folder) to a previous state if necessary.

Git is a widely used, free, open-source distributed version control system. Unlike centralized systems where files are stored on a single server that the collaborators access remotely, Git uses a distributed approach to version control: the complete source code, including its full version history, is mirrored on every collaborator's computer; in addition, collaborators have the ability to work offline and independently and then to synchronize their changes. When collaborators make updates, Git records the edits made to the modified files and also keeps a list of the unchanged files. Thus, Git allows the project's participants to collaborate efficiently by allowing them to merge and synchronize changes to the project folder and by tracking edits made to the files over time.

A Git-tracked repository, also known as "repo," is a collection of source code, usually a folder, with a `.git` subfolder. GitHub is an online file hosting platform that can be used to create and host Git-tracked repositories remotely. Once a repo is created on GitHub, one can add or remove files or make any other changes to them both remotely (on the GitHub website) and locally (on one's personal computer). You can start organizing your project workflow using Git by following the below introductory steps:

1. Create an account at `https://github.com` and create or join a repository.

2. Download Git Bash from `https://git-scm.com/downloads` and install it.

3. Set up your identity by opening Git Bash on Windows 10 OS or Terminal on macOS and then entering the following two commands:
   `git config --global user.name "Name"`
   `git config --global user.email example@example.com`
   where `Name` should be replaced with your name and `example@example.com` should be replaced with your email address. For further help, see `https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup`.

4. Create a personal access token (PAT) and copy it using the instructions at the following link: `https://docs.github.com/en/github/authenticating-to-github/keeping-your-account-and-data-secure/creating-a-personal-access-token`.

5. On Windows 10 OS, it is possible to store GitHub credentials in order to avoid having to enter the password each time the repository needs to be updated from a local computer. Open "Credential Manager," click on "Windows Credentials," and select "Add a generic credential." Enter `git:https://github.com` in the "Internet or network address" field. Enter the GitHub username and the PAT in the "User name" and "Password" fields, respectively, and click on "OK." On macOS, the PAT can be entered using the instructions here: `https://docs.github.com/en/github/getting-started-with-github/getting-started-with-git/updating-credentials-from-the-macos-keychain`.

6. To "clone" a repository on a local computer, open Git Bash (or Terminal on macOS) and use the `cd` command to change directory to the location where you wish to place the local repository. Then, if the repository is located at `https://github.com/username/reponame`, where `reponame` is the name of the repository, then entering the following command (in Git Bash on Windows 10 OS or Terminal on macOS) should clone the repository locally:
   `git clone https://github.com/username/reponame.git`
   where `username` and `reponame` should be modified appropriately.

7. To update the local repository on Windows 10 OS, first right-click on the folder and select "Git Bash Here." To do so on macOS, open Terminal and change directory to the local repository folder (using the `cd` command). Then, entering the command
   `git pull`
   should usually do the job. When this does not work, entering
   `git reset --hard HEAD`
   should fully reset the local repository according to the latest version of the remote repository, although any previous changes on the local repository may be erased if they were not updated on the remote repository.

8. After making changes to the local repository (by modifying, adding, or deleting files), the following commands should be entered sequentially (using Git Bash on Windows 10 OS or Terminal on macOS) to "push" these changes onto the remote repository:
   `git add .`
   `git commit -m "message"`
   `git push`
   so that the first command adds all new changes, the second command (in which `message` should be replaced with a description of the changes) commits changes, and the third command completes the process by pushing changes to the remote repository.

The above guidelines usually suffice for small-scale and non-complex collaborations. Git has several other features, such as "branching" and "merging," for more complex workflows and collaborations. The following resources provide additional help and guidance on these features:

- `https://training.github.com/downloads/github-git-cheat-sheet`
- `https://education.github.com/git-cheat-sheet-education.pdf`
- `http://book.git-scm.com/book/en/v2`

2 Writing Reproducible Code

A common situation in empirical research is that large or restricted-access data files are often stored in a directory separate from the repository containing the code files. Suppose Alice and Bob are collaborators on a project in this situation, and suppose their panel data files (in `.dta` format) from two survey waves are located on their macOS and Windows 10 OS computers, respectively, as shown in the following diagram:

Alice's data directory on macOS:

```
/Users/alice/Desktop
└── projAB
    └── Data
        ├── wave1
        │   ├── file1a.dta
        │   └── file1b.dta
        └── wave2
            ├── file2a.dta
            └── file2b.dta
```

Bob's data directory on Windows 10:

```
C:/Users/Bob/Documents
└── ProjectBA
    └── Data
        ├── wave1
        │   ├── file1a.dta
        │   └── file1b.dta
        └── wave2
            ├── file2a.dta
            └── file2b.dta
```

Suppose Bob creates Stata do-files containing lines of code such as the following:

```
use "C:/Users/Bob/Documents/ProjectBA/Data/wave1/file1a.dta", clear
```

If Alice wants to edit Bob's code files and run them on her own computer, she will have to first undertake the tedious task of modifying the file paths in Bob's code, since Alice does not have the directory `C:/Users/Bob/Documents/ProjectBA` on her macOS computer. For example, she will have to replace the above example line of code with the following:

```
use "/Users/alice/Desktop/projAB/Data/wave1/file1a.dta", clear
```

Moreover, if another person acquires the dataset and wants to check Alice and Bob's code, that third party will have to further modify Alice's paths to be able to run the project's code. To avoid such problems, Alice and Bob could set up an "environment variable," say `projectab`, on their computers and use it in their code files to refer to the project's directory containing the data. This approach would enable Alice, Bob, and the third party to be able to run the code seamlessly regardless of where the data files are located on their individual computers (as long as each of them sets up the required environment variables correctly).

One can set up environment variables on Windows 10 OS by following the below instructions:

1. Open "Control Panel," select "System and Security," and then select "System." Then, on the left pane, select "Advanced system settings." This will open "System Properties."

2. Select the "Envrionment Variables..." option at the bottom. Then, click on the "New..." option under the "User variables" section of the "Environment Variables" window.

3. Then, in the "New User Variable" dialog box, enter the "Variable name" (e.g., `projectab` in Bob's case) and "Variable value" (e.g., `C:\Users\Bob\Documents\ProjectBA` in Bob's case using a backslash `\` rather than a forward slash `/` in the file path). Then, click on "OK." Again click on "OK" in the "System Properties" interface window. This completes the setup.

On macOS computers, one can use the `env` command or the `printenv` command to see the existing environment variables. One can set up new environment variables on macOS by following the below instructions:

1. Open Script Editor (using Launchpad) and enter the following line into the editor:
   `do shell script "launchctl setenv [NAME] [PATH]"`
   where `[NAME]` and `[PATH]` should be replaced with the environment variable name and the corresponding folder path, respectively. For example, Alice could set up the environment variable `projectab` by entering the following line into the editor:
   `do shell script "launchctl setenv projectab /Users/alice/Desktop/projAB"`
   If other environment variables need to be set up, a new line (using the same syntax) should be used in the editor for each of them.

2. Then, save the script as "Environments" (or any other name, since the file name per se does not matter) on Desktop or any other easily accessible location. When saving, make sure to select "Application" in the drop-down list next to "File Format."

3. In the upper left corner, click on the Apple icon and choose "System Preferences." Select "Users & Groups" and then your user account. Next, click on "Login Items" at the top of the window and add the saved "Environemnts" script to "Login Items."

4. To apply changes, restart your computer. Check that the environment variable(s) work(s) by opening Terminal and using the command `env` or `printenv`.

After completing the setup of environment variable(s), which in Alice and Bob's case is called `projectab`, it is possible to write reproducible code that can run on any computer containing the required directories (e.g., data folders) that are connected to the correct common environment variable(s). For example, the following Stata code would produce the same result on the computers of both Alice and Bob:

```
global projectab: env projectab
use $projectab/Data/wave1/file1a.dta, clear
```

In the above code, the name of the global need not be the same as that of the environment variable. For example, the below code produces the same result as the above code:

```
global pab_dir: env projectab
use $pab_dir/Data/wave1/file1a.dta, clear
```

Similarly, R code achieving the same result is as follows:

```
library(haven)
pab_dir <- Sys.getenv("projectab")
file1a_data <- read_dta(file = file.path(pab_dir,"Data/wave1/file1a.dta"))
```

Using environment variables is of course only one of the many steps involved in writing reproducible code. The following sections use a fully worked example to provide many more guidelines.

3 A Worked Example of Project Workflow for Supplemental Analysis of a Microfinance Experiment in India

Henceforth this guide uses a fully worked example to illustrate project workflow for conducting supplemental analysis of a publicly available dataset. The data used for this worked example comes from a field experiment conducted by Erica Field, Rohini Pande, John Papp, and Natalia Rigol. To understand whether increasing repayment flexibility in microfinance contracts can promote entrepreneurship among the poor, Field et al. (2013) collaborated with an Indian microfinance institution called Village Financial Services (VFS). In 2007, they recruited 169 five-member groups of women from low-income regions of Kolkata, India. Before assigning the groups/clusters to experimental conditions, each of the 845 clients was approved to receive an individual-liability loan that varied in size from 4000 INR to 10,000 INR. Before the loans were disbursed, the clusters were randomized into two repayment schedules: regular contract and grace-period contract.

Number of control and treatment clusters within each stratum

Stratum	1	2	3	4	5	6	7	8	9	Total
Control clusters	10	12	10	9	8	11	10	10	5	85
Treatment clusters	10	8	10	11	12	9	10	10	4	84
All clusters	20	20	20	20	20	20	20	20	9	169

Note: Each cluster consists of 5 women, so the total sample size is 845.

As shown in the above contingency table, the field experiment has a completely randomized design within each of the nine strata. The control clusters, which were assigned the regular contract, had to initiate repayment two weeks after loan receipt as usual in the classic microfinance model. The treatment clusters, which received the grace-period contract, were given a two-month grace period before repayment commenced. There was no attrition of clients between randomization and loan disbursement. Then, over the next three years, Field et al. (2013) collected rich administrative and survey data on the clients, including information on their investment behavior, household income, microenterprise profit and assets, repayment behavior, and business practices. For each outcome, the parameter of interest is the average treatment effect (ATE), which has a population version $\vartheta = \mathbb{E}[Y^1_{ck} - Y^0_{ck}]$ as well as a finite-sample version τ, which is given by

$$\tau \equiv \frac{1}{5n} \sum_{c=1}^{n} \sum_{k=1}^{5} Y^1_{ck} - Y^0_{ck},$$

where Y^1_{ck} and Y^0_{ck} denote the potential outcomes for the k-th client in cluster c under the grace-period and regular contracts, respectively, and $n = 169$ is the number of experimental clusters.

Since all units within each cluster receive the same treatment, the observed outcome of the k-th client in cluster c is

$$Y_{ck} = D_c\, Y_{ck}^1 + (1 - D_c)\, Y_{ck}^0,$$

where D_c is a binary indicator of whether cluster c was randomly assigned the grace-period contract. Field et al. (2013) make inferences about the ATE by estimating the parameters (and the associated clustered standard errors) of following equation:

$$Y_{ck} = \beta\, D_c + \eta_{S_c} + \gamma\, W_{ck} + \epsilon_{ck},$$

where: $S_c \in \{1, \ldots, 9\}$ is a categorical variable indicating the stratum of cluster c; η_s denotes fixed effect for stratum $s \in \{1, \ldots, 9\}$; W_{ck} is a vector of baseline characteristics for client k in cluster c; and ϵ_{ck} is her idiosyncratic error term in the equation. See Field et al. (2013) for the list of covariates that comprise the vector W_{ck}. When most or all elements of γ are restricted to be zero, β can be estimated using a simple ordinary least squares (*simple OLS*) estimator $\widehat{\delta}$. Otherwise β can be estimated using an *adjusted OLS* estimator $\widehat{\theta}$ that controls for the baseline covariates W_{ck}. Given the experimental design, both $\widehat{\delta}$ and $\widehat{\theta}$ are good estimators of the ATE. In addition to these main estimators used by Field et al. (2013), it is also possible to construct a supplemental estimator by incorporating propensity scores so that it becomes robust to misspecification of the regression equation. The augmented inverse probability weighting (*augmented IPW*) estimator $\widehat{\tau}$, which is given later below, achives such double robustness. Ding and Li (2018), Athey et al. (2017), and Lunceford and Davidian (2004) provide useful surveys on this topic. Before constructing $\widehat{\tau}$, it is useful to recognize that

$$\tau \equiv \frac{1}{5n} \sum_{c=1}^{n} \sum_{k=1}^{5} Y_{ck}^1 - Y_{ck}^0 = \frac{1}{n} \sum_{c=1}^{n} \left(\frac{1}{5} \sum_{k=1}^{5} Y_{ck}^1 - Y_{ck}^0 \right) = \frac{1}{n} \sum_{c=1}^{n} \bar{Y}_c^1 - \bar{Y}_c^0,$$

where $\bar{Y}_c^d = \frac{1}{5} \sum_{k=1}^{5} Y_{ck}^d$ for all $d \in \{0, 1\}$ and $c \in \{1, \ldots, n\}$. Thus, individual-level data are not necessary for estimating the ATE in a double robust manner. It suffices to have the cluster-level data $(\bar{Y}_c, D_c, S_c, \bar{X}_c)_{c=1}^n$, where $\bar{Y}_c = D_c\, \bar{Y}_c^1 + (1 - D_c)\, \bar{Y}_c^0$, $\bar{X}_c = \frac{1}{5} \sum_{k=1}^{5} X_{ck}$, and X_{ck} is a vector containing one (the number) and the following baseline covariates: age, marital status, religion indicator, household size, years of education, loan amount, indicators of financial control, any household shock, business ownership, home ownership, and any drain in the neighborhood, as well as dummy variables for strata, for loan officers, and for loan amounts (in four bins). Since $(\bar{Y}_c^1, \bar{Y}_c^0) \perp\!\!\!\perp D_c \mid S_c$ because of the experimental design, the following augmented IPW estimator $\widehat{\tau}$ incorporating the propensity score $\hat{\varphi}_c = (\sum_{l=1}^{n} 1\{S_l = S_c\}\, D_l)/(\sum_{l=1}^{n} 1\{S_l = S_c\})$ is a valid, double

robust estimator of ATE (Athey et al., 2017; Ding and Li, 2018; Lunceford and Davidian, 2004):

$$\widehat{\tau} = \frac{1}{n}\sum_{c=1}^{n}\widehat{\tau}_c \equiv \frac{1}{n}\sum_{c=1}^{n}\Big\{[\hat{Y}_c^1 + D_c\,(\bar{Y}_c - \hat{Y}_c^1)/\hat{\varphi}_c] - [\hat{Y}_c^0 + (1-D_c)\,(\bar{Y}_c - \hat{Y}_c^0)/(1-\hat{\varphi}_c)]\Big\},$$

where $\hat{Y}_c^d$ is based on a regression model for $d \in \{0,1\}$: $\hat{Y}_c^d = \bar{X}_c'(\sum_{l:D_l=d}\bar{X}_l\bar{X}_l')^{-1}(\sum_{l:D_l=d}\bar{X}_l\bar{Y}_l)$. For hypothesis testing purposes, the estimate $\widehat{\tau}$ can be studentized as $\widehat{\xi}_\tau = \widehat{\tau}/\widehat{\sigma}_\tau$ using an empirical sandwich method-based standard error $\widehat{\sigma}_\tau = \sqrt{n^{-2}\sum_{c=1}^{n}(\widehat{\tau}_c - \widehat{\tau})^2}$, which seems to work well in practice (Lunceford and Davidian, 2004).

To make inferences regarding average treatment effects on outcomes, Field et al. (2013) report cluster-robust standard errors (for ATE estimates) using asymptotic formulas. This information can be supplemented by other measures of statistical uncertainty. The stratified and completely randomized nature of the experimental design enables alternative modes of inference using the block bootstrap procedure and various types of randomization tests, which have finite-sample validity in this setting. There are nine strata and two experimental conditions (i.e., regular and grace-period contracts), and so there are 18 blocks in total, as shown in the contingency table at beginning of this section. To conduct the block bootstrap procedure, one can use simulation methods to resample clusters with replacement but within each of the 18 blocks. The bootstrap standard errors $\widetilde{\sigma}_\delta$, $\widetilde{\sigma}_\theta$, and $\widetilde{\sigma}_\tau$ associated respectively with the ATE estimates $\widehat{\delta}$, $\widehat{\theta}$, and $\widehat{\tau}$ can be obtained by computing the standard deviations of the corresponding bootstrap distributions, which can be obtained by applying the estimators to simulated block bootstrap-based resamples of the original data.

It is possible to use randomization inference for conducting an exact test of the sharp null hypothesis that $\tau_c \equiv \bar{Y}_c^1 - \bar{Y}_c^0 = 0\ \forall c$. To do this, one can generate M simulations $(D_m^*)_{m=1}^M \equiv ((D_{m,c}^*)_{c=1}^n)_{m=1}^M$ of the treatment indicator vector (containing binary treatment indicators for the n clusters) using the random assignment mechanism used in the experiment (i.e., complete randomization within each of the nine strata). Because of the experimental design, the contingency table (showing the number of control and treatment clusters within each stratum) remains invariant under all of these simulated treatment status vectors $(D_m^*)_{m=1}^M$. For each simulation $m \in \{1,\ldots,M\}$ under the sharp null hypothesis, one can apply the above ATE estimators to $(\bar{Y}_c, D_{m,c}^*, S_c, \bar{X}_c)_{c=1}^n$ and obtain $\widehat{\delta}_m^*$, $\widehat{\theta}_m^*$, $\widehat{\tau}_m^*$, and $\widehat{\xi}_m^*$, which are the simulated counterparts of $\widehat{\delta}$, $\widehat{\theta}$, $\widehat{\tau}$, and $\widehat{\xi}$, respectively. The exact p-values for these test statistics are given by $\tilde{p}_\delta^\circ = (1 + \sum_{m=1}^M 1\{|\widehat{\delta}_m^*| \geq |\widehat{\delta}|\})/(1+M)$, $\tilde{p}_\theta^\circ = (1 + \sum_{m=1}^M 1\{|\widehat{\theta}_m^*| \geq |\widehat{\theta}|\})/(1+M)$, $\tilde{p}_\tau^\circ = (1 + \sum_{m=1}^M 1\{|\widehat{\tau}_m^*| \geq |\widehat{\tau}|\})/(1+M)$, which are all nonstudentized test-based exact single p-values, and $\tilde{q}_\tau^\circ = (1 + \sum_{m=1}^M 1\{|\widehat{\xi}_m^*| \geq |\widehat{\xi}|\})/(1+M)$, which is a studentized test-based exact single p-value. If there is no heterogeneity in cluster-level treatment effects $(\tau_c)_{c=1}^n$, the sharp null hypothesis is equivalent to the weak null hypothesis that $\tau = 0$. In this case, $\tilde{p}_\delta^\circ$, $\tilde{q}_\theta^\circ$, $\tilde{p}_\tau^\circ$, and $\tilde{q}_\tau^\circ$ would all be valid as exact p-values for testing $\tau = 0$. However, if there is heterogeneity in cluster-level treatment effects $(\tau_c)_{c=1}^n$, the former three p-values may not

have good properties even asymptotically for testing $\tau = 0$. In this case, it is preferable to use the studentized test-based p-value $\tilde{q}^{\circ}_{\tau}$ rather than the nonstudentized test-based p-values $\tilde{p}^{\circ}_{\delta}$, $\tilde{q}^{\circ}_{\theta}$, and $\tilde{p}^{\circ}_{\tau}$ (see, e.g., Wu and Ding, 2020).

Tables 1 through 4 of Field et al. (2013) report on the impact of grace period on various blocks or categories of (three or four) outcomes, such as investment behavior, household income, microenterprise profit and assets, repayment behavior, and business practices. The aforementioned supplemental statistics related to these blocks of outcomes can be presented in two tables. Let $\widehat{\alpha}_{i.j}$ represent the sample mean of control clusters for outcome j in block or category i. Let $\widehat{\delta}_{i.j}$, $\widehat{\theta}_{i.j}$, and $\widehat{\tau}_{i.j}$ be the ATE estimates using the simple OLS, adjusted OLS, and augmented IPW estimators, respectively. In addition, let $\widetilde{\sigma}_{\delta,i.j}$, $\widetilde{\sigma}_{\theta,i.j}$, $\widetilde{\sigma}_{\tau,i.j}$ be their bootstrap standard errors, and $\tilde{p}^{\circ}_{\delta,i.j}$, $\tilde{p}^{\circ}_{\theta,i.j}$, $\tilde{p}^{\circ}_{\tau,i.j}$ be the nonstudentized randomization test-based exact single p-values. All of these statistics can be reported in a single table as follows, enabling evaluation of robustness of the results to various estimation and inference choices.

Desired format for table on impact of grace period on business activity and repayment outcomes

Outcome	*Control mean*	*Simple OLS estimate of ATE*	*Adjusted OLS estimate of ATE*	*Augmented IPW estimate of ATE*
$(i.j)$ Outcome label	$\widehat{\alpha}_{i.j}$	$\widehat{\delta}_{i.j}$ $(\widetilde{\sigma}_{\delta,i.j})$ $[\tilde{p}^{\circ}_{\delta,i.j}]$	$\widehat{\theta}_{i.j}$ $(\widetilde{\sigma}_{\theta,i.j})$ $[\tilde{p}^{\circ}_{\theta,i.j}]$	$\widehat{\tau}_{i.j}$ $(\widetilde{\sigma}_{\tau,i.j})$ $[\tilde{p}^{\circ}_{\tau,i.j}]$

In a table with the above format, the outcomes are treated as separate. However, it is also useful to simultaneously make inferences about the multiple outcomes within each block or category. As discussed before, the following studentized test-based exact single p-value for outcome j in block i is not adjusted for multiple testing: $\tilde{q}^{\circ}_{\tau,i.j} = (1 + \sum_{m=1}^{M} 1\{|\widehat{\xi}^{*}_{m,i.j}| \geq |\widehat{\xi}_{i.j}|\})/(1+M)$, where $(\widehat{\xi}^{*}_{m,i.j})_{m=1}^{M}$ and $\widehat{\xi}_{i.j}$ are versions of $(\widehat{\xi}^{*}_{m})_{m=1}^{M}$ and $\widehat{\xi}$ specific to outcome j in block i. This issue can be addressed by using the stepdown procedure suggested by Romano and Wolf (2005, 2016). Applying their procedure results in stepdown p-values $\tilde{q}^{\star}_{\tau,i.j}$. These account for multiple testing within each block of outcomes and can be presented along with single p-values (for the estimate $\widehat{\tau}_{i.j}$) as follows.

Desired table format for stepdown inference on impacts of grace period for microfinance loans

Outcome	*Control mean*	*Augmented IPW estimate of ATE*	*Nonstudentized test-based exact single p-value*	*Studentized test-based exact single p-value*	*Studentized test-based exact stepdown p-value*
$(i.j)$ Outcome label	$\widehat{\alpha}_{i.j}$	$\widehat{\tau}_{i.j}$	$\tilde{p}^{\circ}_{\tau,i.j}$	$\tilde{q}^{\circ}_{\tau,i.j}$	$\tilde{q}^{\star}_{\tau,i.j}$

The computationally efficient algorithm proposed by Romano and Wolf (2005, 2016) for computing stepdown p-values $\tilde{q}^{\star}_{\tau,i.j}$ proceeds as follows. Within block i of outcomes $j \in \{1,\ldots,S\}$, let $\{r_1,\ldots,r_S\}$ be a reordering of $\{1,\ldots,S\}$ such that $|\widehat{\xi}_{i.r_1}| \geq |\widehat{\xi}_{i.r_2}| \geq \cdots \geq |\widehat{\xi}_{i.r_S}|$ is satisfied. Then, let $v^{*}_{m,i.j} = \max\{|\widehat{\xi}^{*}_{m,i.r_j}|,\cdots,|\widehat{\xi}^{*}_{m,i.r_S}|\}$ for $j \in \{1,\ldots,S\}$ and $m \in \{1,\ldots,M\}$, and let the initial versions of stepdown p-values be given by $\tilde{q}'_{\tau,i.j} = (1+\sum_{m=1}^{M} 1\{v^{*}_{m,i.j} \geq |\widehat{\xi}_{i.j}|\})/(1+M)$; these may not necessarily be monotonic according to the ordering $\{r_1,\ldots,r_S\}$, and so a further adjustment is needed. The final stepdown p-values are given by $\tilde{q}^{\star}_{\tau,i.r_1} = \tilde{q}'_{\tau,i.r_1}$ and $\tilde{q}^{\star}_{\tau,i.r_s} = \max\{\tilde{q}'_{\tau,i.r_s}, \tilde{q}^{\star}_{\tau,i.r_{s-1}}\}$ for $s \in \{2,\ldots,S\}$.

The problem at hand is to create the above two desired tables (as TeX files) in the specified format. The goal of this worked example, which assumes that the reader has at least some basic knowledge of R or Stata, is to solve the problem using only a single run of the main (or master) code file. To this end, it is useful to organize the project folder as follows.

Project folder's subfolders:

```
..
├── scripts
│   ├── programs
│   ├── simulations
│   └── outputs
├── store
├── tables
├── drafts
└── .git
```

File structure of `scripts` subfolder:

```
../scripts
├── master.{do,R}
├── programs
│   ├── aipw_strata.{do,R}
│   └── stepdown_pval.{do,R}
├── simulations
│   ├── outcomes.{do,R}
│   ├── ate_estimators.{do,R}
│   ├── observed_estimates.{do,R}
│   ├── bootstrap_dist.{do,R}
│   └── randomization_dist.{do,R}
└── outputs
    ├── appended_results.{do,R}
    ├── table_single_pvals.{do,R}
    └── table_stepdown_pvals.{do,R}
```

There are four main subfolders in the project folder: `scripts`, `store`, `tables`, and `drafts`. The `scripts` subfolder contains code files for producing the tables. The role of the subfolder `store` is to act as a reserve for any intermediate outputs, which are created and then later accessed as needed by a sequence of code files. The subfolder `tables` is where a single run of the code generates and saves the TeX files containing customized tables in desired formats. These `.tex` tables are then integrated into the manuscript files stored in the `drafts` subfolder.

The `scripts` subfolder is further organized into these folders: `programs`, `simulations`, and `outputs`. The folder `programs` contains the following general-purpose programs (in Stata do-files and R source files): `aipw_strata`, which implements the augmented IPW estimator,

and stepdown_pval, which generates stepdown p-values for specified blocks of outcomes. The simulations folder contains several interconnected files: outcomes, which specifies the outcomes for analysis; ate_estimators, which is a project-specific program that takes the dataset as input and computes the simple OLS, adjusted OLS, and augmented IPW estimates of ATE for the specified outcomes; observed_estimates applies the ate_estimators program to the original experimental dataset; bootstrap_dist generates block bootstrap-based resamples of the original dataset and then applies the ate_estimators program to the bootstrapped datasets; and randomization_dist simulates random assignment of treatment using the experimental design and then applies the ate_estimators program to the simulated datasets, which do not modify the observed outcomes and baseline variables but replace the original treatment assignment variable with the simulated vector of treatment indicators. Using these observed estimates and resampled estimates generated in the store folder, the code files in the scripts/outputs folder finish the task at hand. Specifically, appended_results uses the stored estimates (both observed and resampled ones) to compute the needed statistics. Then, the code files table_single_pvals and table_stepdown_pvals generate and save (in the tables folder) the TeX files containing the formatted tables. At a single click, running master.do or master.R, which sequentially run all of the code files, will produce the following tables in TeX format in tables folder: tables_single_pvals_{stata,r}.tex and tables_stepdown_pvals_{stata,r}.tex.

master.do

```
cd simulations
do "observed_estimates.do"
do "bootstrap_dist.do"
do "randomization_dist.do"

cd ../outputs
do "appended_results.do"
do "table_single_pvals.do"
do "table_stepdown_pvals.do"
```

master.R

```
install.packages("dplyr")
install.packages("pbapply")
install.packages("haven")

setwd(file.path(getwd(),"/simulations"))
source("observed_estimates.R")
source("bootstrap_dist.R")
source("randomization_dist.R")

setwd(file.path(getwd(),"../outputs"))
source("appended_results.R")
source("table_single_pvals.R")
source("table_stepdown_pvals.R")
```

4 Writing Customized Statistical Programs

This section provides some examples and basic guidelines for writing customized statistical programs in R and Stata. As mentioned in the previous section, the project folder is organized such that general-purpose programs are stored in `../scripts/programs`. The Stata do-file `aipw_strata.do` and the R source file `aipw_strata.R` define programs for implementing the augmented IPW estimator discussed earlier:

$$\widehat{\tau} = \frac{1}{n}\sum_{c=1}^{n}\widehat{\tau}_c \equiv \frac{1}{n}\sum_{c=1}^{n}\left\{[\hat{Y}_c^1 + D_c\,(\bar{Y}_c - \hat{Y}_c^1)/\hat{\varphi}_c] - [\hat{Y}_c^0 + (1-D_c)\,(\bar{Y}_c - \hat{Y}_c^0)/(1-\hat{\varphi}_c)]\right\},$$

where $\hat{\varphi}_c = (\sum_{l=1}^{n} 1\{S_l = S_c\}\,D_l)/(\sum_{l=1}^{n} 1\{S_l = S_c\})$ and $\hat{Y}_c^d$ is based on a regression model for $d \in \{0,1\}$: $\hat{Y}_c^d = \bar{X}_c'(\sum_{l:D_l=d}\bar{X}_l\bar{X}_l')^{-1}(\sum_{l:D_l=d}\bar{X}_l\bar{Y}_l)$. The estimate's empirical sandwich method-based standard error is given by $\widehat{\sigma}_\tau = \sqrt{n^{-2}\sum_{c=1}^{n}(\widehat{\tau}_c - \widehat{\tau})^2}$. The following code in the do-file `aipw_strata.do` defines a command in Stata for computing the estimator and its standard error. Its R counterpart `aipw_strata.R` is provided in the Appendix.

In Stata, customized programs are usually written using the `program` syntax. (Stata also has its own programming language called Mata. See `https://www.stata.com/manuals/m.pdf`.) When user-written programs are stored as `.ado` files in the appropriate location (i.e., a directory where Stata automatically searches for programs), they can be used just like other commands such as `summarize` or `regress`. Alternatively, for project-specific purposes, they can be written in an ordinary do-file, such as `aipw_strata.do`, and can be manually loaded as necessary.

The program begins with the line `program aipw_strata, rclass` and ends simply with the word `end`. The `rclass` option allows for storage of the desired results $\widehat{\tau}$ (the augmented IPW estimate) and $\widehat{\sigma}_\tau$ (its standard error) in `r(aipw_strata_est)` and `r(aipw_strata_se)`, respectively. As discussed in `https://www.stata.com/manuals/pprogram.pdf`, there are alternatives to the `rclass` option. The syntax for the program `aipw_strata` involves four input strings. The first string, which is stored in the `local` variable `yvar`, should be the name of the outcome variable in the dataset. The strings specifying the binary treatment variable and categorical strata variable are stored in `local` variables `tvar` and `svar`, respectively, while the baseline covariates are stored in `xvars`. These `local` variables are internal to the program, as opposed to `global` variables, which remain operational throughout a Stata session. The macros `local` and `global` are explained more at `https://www.stata.com/manuals13/u18.pdf#u18ProgrammingStata`.

The program `aipw_strata` first creates strata-based propensity scores and then generates predicted values from regressions for treated and untreated units in a completely randomized stratified experiment. Using these quantities, the augmented IPW estimate $\widehat{\tau}$ and its standard error $\widehat{\sigma}_\tau$ are computed according the above formulas are returned as outputs after the command is executed.

begin aipw_strata.do

```
*------------------------------------------------------------------------------*
* Augmented inverse probability weighting estimator for stratified experiments  *
*------------------------------------------------------------------------------*
/*
Note: This file defines a program for estimating the average treatment effect
using the augmented inverse probability weighting (AIPW) estimator for
completely randomized stratified experiments without any missing data.
*/
*---------------------------------- Syntax ----------------------------------*
/*
The syntax is:
aipw_strata "[yvar]" "[tvar]" "[svar]" "[xvars]"
where
[yar] is the name of the outcome variable,
[tvar] is the name of the binary treatment variable,
[svar] is the name of the categorical strata variable, and
[xvars] is the list of baseline covariates.

Example:
aipw_strata "outcome" "treated" "stratum_id" "xvar1 xvar2 xvar3"
*/
*---------------------------------- Program ---------------------------------*

version 17.0

program aipw_strata, rclass

        local yvar "`1'" // outcome variable
        local tvar "`2'" // binary treatment variable
        local svar "`3'" // categorical strata variable
        local xvars "`4'" // baseline covariates

        egen _pr_treat = mean(`tvar'), by(`svar') // strata-based propensity score
        reg `yvar' `xvars' if `tvar'==1 // regression model for treated units
        predict _yhat_1
        reg `yvar' `xvars' if `tvar'==0 // regression model for untreated units
        predict _yhat_0
        gen _aipw_te = (_yhat_1 + (`tvar'/_pr_treat) * (`yvar' - _yhat_1)) ///
        - (_yhat_0 + ((1 - `tvar')/(1 - _pr_treat)) * (`yvar' - _yhat_0))
        reg _aipw_te, robust // AIPW estimator
        return scalar aipw_strata_est = _b[_cons] // store AIPW estimate
        return scalar aipw_strata_se = _se[_cons] // store its standard error
        drop _pr_treat _yhat_1 _yhat_0 _aipw_te // revert dataset to prior state

end
```

end aipw_strata.do

The ../scripts/programs folder also contains a set of general-purpose programs (in the Stata do-file stepdown_pval.do and the R source file stepdown_pval.R) for computing step-down *p*-values that are adjusted for multiple testing. As mentioned earlier, the computationally

efficient algorithm proposed by Romano and Wolf (2005, 2016) for computing stepdown p-values $\tilde{q}^{\star}_{\tau,i.j}$ proceeds as follows. Within block i of outcomes $j \in \{1,\ldots,S\}$, let $\{r_1,\ldots,r_S\}$ be a reordering of $\{1,\ldots,S\}$ such that $|\widehat{\xi}_{i.r_1}| \geq |\widehat{\xi}_{i.r_2}| \geq \cdots \geq |\widehat{\xi}_{i.r_S}|$ is satisfied. Then, let $v^{*}_{m,i.j} = \max\{|\widehat{\xi}^{*}_{m,i.r_j}|,\cdots,|\widehat{\xi}^{*}_{m,i.r_S}|\}$ for $j \in \{1,\ldots,S\}$ and $m \in \{1,\ldots,M\}$. Furthermore, let the initial versions of stepdown p-values be given by $\tilde{q}'_{\tau,i.j} = (1 + \sum_{m=1}^{M} 1\{v^{*}_{m,i.j} \geq |\widehat{\xi}_{i.j}|\})/(1+M)$; these may not necessarily be monotonic according to the ordering $\{r_1,\ldots,r_S\}$, and so a further adjustment is needed. The final stepdown p-values are given by $\tilde{q}^{\star}_{\tau,i.r_1} = \tilde{q}'_{\tau,i.r_1}$ and $\tilde{q}^{\star}_{\tau,i.r_s} = \max\{\tilde{q}'_{\tau,i.r_s}, \tilde{q}^{\star}_{\tau,i.r_{s-1}}\}$ for $s \in \{2,\ldots,S\}$. The following code in the R script `stepdown_pval.R` defines a function in R for computing the stepdown p-values using the supplied null distributions of test statistics for specified outcomes. Its Stata counterpart `stepdown_pval.do`, which is much longer and relatively inefficient, is provided in the Appendix. The function `stepdown_pval` defined in the following R code `stepdown_pval.R` takes two inputs: a data frame containing the simulated null distributions of test statistics for a specified block of outcomes; and the corresponding vector of observed test statistics. The function then applies the above algorithm and outputs the stepdown p-values.

begin `stepdown_pval.R`

```
stepdown_pval <- function(null_tstats_dataframe, obs_tstats_vec) {

  require("dplyr")

  S <- length(obs_tstats_vec)

  # r = indices of highest to lowest observed t-statistics for outcomes
  r <- unlist(sort(obs_tstats_vec, decreasing = TRUE, index.return = TRUE)[2])

  maxt <- data.frame(matrix(ncol = S, nrow = nrow(null_tstats_dataframe)))
  colnames(maxt) <- colnames(null_tstats_dataframe)
  for (j in 1:(S-1)) {
    maxt[,r[j]] <- apply(null_tstats_dataframe[,r[j:S]], 1, max)
  }
  maxt[,r[S]] <- null_tstats_dataframe[,r[S]]

  stepdown_pval_vec_initial <- NULL
  for (j in 1:S) {
    stepdown_pval_vec_initial[j] <- mean(c(1,maxt[,j] >= obs_tstats_vec[j]))
  }
  stepdown_pval_vec_adjusted <- NULL
  stepdown_pval_vec_adjusted[r[1]] <- stepdown_pval_vec_initial[r[1]]
  for (k in 2:S) {
    stepdown_pval_vec_adjusted[r[k]] <-
      max(stepdown_pval_vec_initial[c(r[k],r[k-1])])
  }
  return(list(stepdown_pval_vec_adjusted,maxt))

}
```

end `stepdown_pval.R`

In fact, stepdown p-values are not the only results that the function `stepdown_pval` produces. It also returns a data frame containing the adjusted null distributions that are used for stepdown multiple testing. Upon applying the `stepdown_pval` function to the appropriate inputs, it returns both the outputs in the form of a `list`, which is a R object that contains elements of different types, such as a number, vector, string, data frame, or even another nested list. This aspect of R can be more convenient relative to Stata in some contexts. The l-th element of a list, say `example_list`, can be retrieved using the syntax `example_list[[l]]`.

As can be seen in the previous code, the `stepdown_pval` makes use of several `for` loops. Its usual syntax starts with the form `for (j in 1:S) {`, where `j` should be replaced with the desired notation for the index and `1:S` should be replaced with the appropriate vector. The loop should be closed with `}` after specifying the operation inside the loop. In Stata, there are two types of for-loops in which the index is a `local`: count-controlled loops, e.g., `forval j = 1/10 { }` or `forval j = 1(2)10 {}`; and collection-controlled loops, e.g., `foreach var in ‘varlist’ { }`, where `var` is the index and `varlist` is a `local` macro containing a list of items intended for iteration in the loop. Stata and R also accommodate `while` loops with a similar syntax as follows: `while (` logical condition `) {` expression `}`. In R, after initializing `i <- 0`, the following example prints the numbers 1 through 5: `while (i <= 4) {i <- i + 1; print(i)}`. In Stata, after initializing `local i = 0`, the same output can be achieved using the condition `(‘i’ <= 4)` and the following two lines of expressions: `local i = ‘i’ + 1` and `display ‘i’`. Stata and R also support conditional expressions such as if-else statements: `if (` logical condition `) {` statements `}` followed (optionally) by `else {` statements `}`.

While the folder `../scripts/programs` contains general-purpose programs `aipw_strata` and `stepdown_pval`, the folder `../scripts/simulations` contains project-specific programs for computing the ATE estimators as well as their sampling and null distributions.

begin `outcomes.do`

```
/* Outcomes in Table 1 of Field et al. (2013) */ global loan_use_vars ///
"Business_Expenditures Non_Business_Exp New_Business_Ap15"
/* Outcomes in Table 2 of Field et al. (2013) */ global profits_vars ///
"Profit ln_Q50 Capital"
/* Outcomes in Table 3 (left) of Field et al. (2013) */ global default_vars ///
"Late_Days_364 Late_Days_476 not_finished_aug19 Outstanding_Loan"
/* Outcomes in Table 3 (right) of Field et al. (2013) */ global repayment_vars ///
"Fifty_Percent_Loan_Paid Made_First_11 Made_First_Pay"
/* Outcomes in Table 4 (left) of Field et al. (2013) */ global business_vars ///
"atleastone Max_Min Q68"
/* Outcomes in Table 4 (right) of Field et al. (2013) */ global customers_vars ///
"Q35_ Q37_ Q11_Together_max"
global last_dvars "$default_vars $repayment_vars $business_vars $customers_vars"
global all_dvars "$loan_use_vars $profits_vars $last_dvars"
```

end `outcomes.do`

The files `outcomes.do`, shown above, and `outcomes.R`, presented in the Appendix, contain code for storing outcomes of interest for the supplemental analysis. The stored outcome variable names are based on the experimental dataset that Field et al. (2013) made publicly avaiable at `https://www.openicpsr.org/openicpsr/project/112672/version/V1/view?path=/openicpsr/112672/fcr:versions/V1/Publication-Data-and-Do-Files/Grace-Period-Data.dta&type=file`. The Stata do-file shown above uses `global` macros to store these outcome names, while the R source file uses a vector of strings as usual. These are then fed into the project-specific programs `ate_estimators.do`, shown below, and `ate_estimators.R`, given in the Appendix. The general-purpose program `aipw_strata` is first loaded before defining the project-specific `ate_estimators` program. Then, `for` loops are used to compute and store the three different ATE estimates as well as the control mean for each outcome of interest. This program can be applied to either the original experimental data or a simulated dataset with the same form.

begin `ate_estimators.do`

```
do ../programs/aipw_strata.do
do outcomes.do

program ate_estimators, rclass

local loan_use_vars $loan_use_vars
local profits_vars $profits_vars
local default_vars $default_vars
local repayment_vars $repayment_vars
local business_vars $business_vars
local customers_vars $customers_vars
local last_dvars $last_dvars
local all_dvars $all_dvars

local baseline_vars1 "i.Stratification_Dummies i.sec_loanamount"
local baseline_vars2 "Age_C Married_C Muslim_C HH_Size_C Years_Education_C shock_any_C"
local baseline_vars3 "Has_Business_C Financial_Control_C homeowner_C No_Drain_C"
local baseline_vars "`baseline_vars1' `baseline_vars2' `baseline_vars3'"

gen Outstanding_Loan = Outstanding_Loan_Amount_Default
gen Made_First_11 = Made_First_11_Pay_On_Time
gen atleastone = atleastone_bizshutdown_alt

foreach dvar in `all_dvars' {
        su `dvar' if sec_treat==0
        return scalar `dvar'_ctrl_est = r(mean)
}

foreach dvar in `loan_use_vars' {
        regress `dvar' sec_treat i.Stratification_Dummies ///
        i.sec_loanamount Match3rd_in3rd, cluster(sec_group_name)
```

```
        return scalar ‘dvar’_ols1_est = _b[sec_treat]

        regress ‘dvar’ sec_treat ‘baseline_vars’ miss_* ///
        Match3rd_in3rd, cluster(sec_group_name)
        return scalar ‘dvar’_ols2_est = _b[sec_treat]
}

foreach dvar in ‘profits_vars’ {
        regress ‘dvar’ sec_treat i.Stratification_Dummies, cluster(sec_group_name)
        return scalar ‘dvar’_ols1_est = _b[sec_treat]

        regress ‘dvar’ sec_treat ‘baseline_vars’ miss_* ///
        i.sec_loan_officer, cluster(sec_group_name)
        return scalar ‘dvar’_ols2_est = _b[sec_treat]
}


foreach dvar in ‘last_dvars’ {
        regress ‘dvar’ sec_treat i.Stratification_Dummies, cluster(sec_group_name)
        return scalar ‘dvar’_ols1_est = _b[sec_treat]

        regress ‘dvar’ sec_treat ‘baseline_vars’ miss_* ///
        i.sec_loan_officer Literate_C, cluster(sec_group_name)
        return scalar ‘dvar’_ols2_est = _b[sec_treat]
}


foreach var in ‘baseline_vars2’ ‘baseline_vars3’ {
        replace ‘var’ = . if miss_‘var’==1
}
tab sec_loan_officer, gen(loan_officer)
gen loan_amount1 = inlist(sec_loanamount,4000,5000)
gen loan_amount2 = inlist(sec_loanamount,6000,7000)
gen loan_amount3 = inlist(sec_loanamount,8000,9000)
gen loan_amount4 = inlist(sec_loanamount,10000)

order sec_group_name *
collapse sec_loanamount - loan_amount4, by(sec_group_name)

local baseline_xvars1 "‘baseline_vars2’ ‘baseline_vars3’ i.Stratification_Dummies"
local baseline_xvars2 "loan_officer1 loan_officer2 loan_officer3 loan_officer4"
local baseline_xvars3 "sec_loanamount loan_amount1 loan_amount2 loan_amount3"
local baseline_xvars "‘baseline_xvars1’ ‘baseline_xvars2’ ‘baseline_xvars3’"

foreach dvar in ‘all_dvars’ {
        aipw_strata "‘dvar’" "sec_treat" "Stratification_Dummies" "‘baseline_xvars’"
        return scalar ‘dvar’_aipw_est = r(aipw_strata_est)
        return scalar ‘dvar’_aipw_tst = r(aipw_strata_est)/r(aipw_strata_se)
}

end
```

end ate_estimators.do

Using the previous code files, it is possible to apply the `ate_estimators` program to the original data. This is done in the R source file `observed_estimates.R`, shown below, as well as the Stata do-file `observed_estimates.do`, which is presented in the Appendix. The code assumes that the user's computer has an environment variable called `aer_103_6_2196_data` for the folder that contains the file `Grace-Period-Data.dta`, which is publicly available for download at `https://www.openicpsr.org/openicpsr/project/112672/version/V1/view?path=/openicpsr/112672/fcr:versions/V1/Publication-Data-and-Do-Files/Grace-Period-Data.dta&type=file`.

begin `observed_estimates.R`

```
source(file.path(getwd(),"ate_estimators.R"))

library(dplyr, warn.conflicts = FALSE)
library(haven)

grace_period_data_path <- Sys.getenv("aer_103_6_2196_data")
gpdata <- read_dta(file=file.path(grace_period_data_path,"Grace-Period-Data.dta"))

obs_estimates <- ate_estimators(gpdata)
saveRDS(obs_estimates, file = "../../store/obs_estimates")

rm(list = ls(all.names = TRUE))
```

end `observed_estimates.R`

The file `observed_estimates.R` first loads the required packages (i.e., `dplyr` and `haven`), obtains the path for the aforementioned environment variable leading to the directory containing the data, loads the experimental dataset, and then applies the `ate_estimators` function to that dataset, before saving (in the subfolder `../store`) the estimates of ATE obtained using the observed data. The above script specifies `file = "../../store/obs_estimates"` to indicate the location and file name for saving the output. The reason is that `observed_estimates.R` is located in `../scripts/simulations`, and so the syntax `../../` indicates that two levels must be moved up before reaching the `../store` subfolder, where the output can be saved with the name `obs_estimates`.

When the dataset is small in size and if it is not a restricted-access file, there is an alternative to the use of the environment variable for specifying the data directory. The dataset can be stored within the project folder in a subfolder `../data`. In this case, there is no need to set up an environment variable called `aer_103_6_2196_data`. The data can be directly read using the syntax `gpdata <- read_dta("../../data/Grace-Period-Data.dta")`. The equivalent syntax in Stata is `use "../../data/Grace-Period-Data.dta", clear`. This approach, when feasible, avoids the burden of setting up the environment variables, making the project folder self-contained.

5 Implementing Simulation Methods

The previously described program `ate_estimators`, which implements three ATE estimators, can also be applied to simulated datasets that are similar in structure to the original dataset. This is useful for obtaining the bootstrap and randomization distributions of test statistics. The R code in `randomization_dist.R` below and its Stata counterpart in `randomization_dist.do` generate simulations of the treatment status vector according to the experimental design and then apply the ATE estimators to the simulated datasets, thereby creating approximate randomization distributions that can be used for hypothesis testing.

Since the simulations need not be conducted sequentially (at least in the current setting), it is efficient to use parallel computing procedures. R is particularly well-suited for parallelization when implementing simulation methods. R supports different types of parallelization (e.g., forks and sockets); for more information, see the notes on parallel processing at a course website: `http://dept.stat.lsa.umich.edu/~jerrick/courses/stat701/`. For additional useful information, see `https://www.programmingr.com` and `https://support.rstudio.com/hc/en-us/articles/201057987-Quick-list-of-useful-R-packages`.

The R code file `randomization_dist.R` first sets a seed to ensure reproducibility of the final output and then loads the required packages. It then starts a cluster using about three-quarters of the available processor cores so as to not overburden the computer. The code then uses the socket method for parallelization. Specifically, the function `clusterEvalQ` executes the first code block on each process. Then the code defines a function called `randomization_test_estimates` that generates a simulated treatment status vector and then applies the ATE estimators to the simulated dataset to obtain random draws from the randomization null distributions of the test statistics. Finally, the code uses the `pblapply` function to generate the desired simulations in a parallel fashion. (Alternatively, the function `parLapply` can also used to obtain the same output, but the advantage of `pblapply` is its ability to display progress and a real-time estimate of time needed for completion.) After the simulations are completed, the code stops the cluster and saves the simulated estimates in the file `rand_dist` in the subfolder `../store`.

begin `randomization_dist.R`

```
ptm <- proc.time()

set.seed(12345)

library(parallel)
library(pbapply)

cl <- makeCluster(round(detectCores() * 0.75))

clusterEvalQ(cl, {
```

```
## START OF CODE BLOCK 1 ##

source(file.path(getwd(),"ate_estimators.R"))

library(dplyr, warn.conflicts = FALSE)
library(haven)

grace_period_data_path <- Sys.getenv("aer_103_6_2196_data")
gpdata <- read_dta(file=file.path(grace_period_data_path,"Grace-Period-Data.dta"))

gpdata_cluster_vars <- gpdata %>%
  select(sec_group_name, Stratification_Dummies, sec_treat) %>%
  group_by(sec_group_name) %>% filter(row_number() == 1)

table(gpdata_cluster_vars$sec_treat, gpdata_cluster_vars$Stratification_Dummies)

## END OF CODE BLOCK 1 ##

})

randomization_test_estimates <- function(...) {

  ## START OF CODE BLOCK 2 ##

  cluster_vars_temp <- gpdata_cluster_vars
  cluster_vars_temp$streat <-
    ave(cluster_vars_temp$sec_treat, cluster_vars_temp$Stratification_Dummies, FUN =
        sample)
  sum(cluster_vars_temp$streat != cluster_vars_temp$sec_treat)
  cluster_vars_temp <- select(cluster_vars_temp, sec_group_name, streat)
  gpdata_temp <- left_join(gpdata, cluster_vars_temp,
                           by = c(sec_group_name = "sec_group_name"))
  sum(gpdata_temp$streat != gpdata_temp$sec_treat)
  gpdata_temp <- gpdata_temp %>% mutate(sec_treat = streat)
  gpdata_temp <- gpdata_temp[,!(names(gpdata_temp) %in% c("streat"))]

  return(ate_estimators(gpdata_temp))

  ## END OF CODE BLOCK 2 ##

}

# rand_dist_temp <- parLapply(cl, 1:2000, randomization_test_estimates)
system.time(rand_dist <- pblapply(1:2000, randomization_test_estimates, cl = cl))

stopCluster(cl)

saveRDS(rand_dist, file = "../../store/rand_dist")

proc.time() - ptm

rm(list = ls(all.names = TRUE))
```

```
#####################################

### ALTERNATIVE CODE:

# set.seed(12345)
#
# ## BLOCK 1 ##
#
# randomization_test_estimates <- function(...) {
#   ## BLOCK 2 ##
# }
#
# rand_dist <- list()
#
# for (sim_num in 1:2000) {
#   rand_dist[[sim_num]] <- randomization_test_estimates()
#   print(sim_num)
# }
#
# saveRDS(rand_dist, file = "../../store/rand_dist")
```

end `randomization_dist.R`

An advantage of R compared to Stata is R's ability to store output in the form of lists, which are very flexible. For example, the code `randomization_dist.R` stores five test statistics for each of 19 outcomes in a 5 by 19 matrix or data frame for each simulation. These 5 by 19 data frames are then collated in a `list` named `rand_dist` that is saved in `../store`.

The Stata counterpart of `randomization_dist.R` is the do-file `randomization_dist.do` shown below. Although Stata/MP automatically parallelizes some basic commands (such as linear regression), Stata's parallelization capabilities for general purposes are currently very limited. For example, Stata's built-in command `simulate` conducts simulations in a sequential fashion. One must resort to some user-written commands such as `parallel` for implementing parallelization in a limited way, as shown in the following do-file.

begin `randomization_dist.do`

```
set seed 12345

net install parallel, from("https://raw.github.com/gvegayon/parallel/stable/") replace
mata mata mlib index

global aer_103_6_2196_data: env aer_103_6_2196_data
global current_dir: pwd

do outcomes.do

program randx, rclass

cap program drop aipw_strata
```

```
cap program drop ate_estimators
cd $current_dir
do ate_estimators.do

use $aer_103_6_2196_data/Grace-Period-Data.dta, clear

gen id = _n
egen group_id = group(sec_group_name)
bysort group_id (id): gen temp_ind = 1 if _n==1
replace temp_ind = 2 if temp_ind==.
egen number_treated = total(sec_treat) if temp_ind==1, by(Stratification_Dummies)
gen rand_uniform = runiform()
sort temp_ind Stratification_Dummies rand_uniform
bysort temp_ind Stratification_Dummies: gen temp_num = _n
gen randomized_sec_treat_temp = (temp_num <= number_treated) if temp_ind==1
egen randomized_sec_treat = mean(randomized_sec_treat_temp), by(sec_group_name)
sort id
replace sec_treat = randomized_sec_treat
drop id group_id temp_ind number_treated rand_uniform ///
temp_num randomized_sec_treat_temp randomized_sec_treat

ate_estimators

foreach dvar in $all_dvars {
        foreach result in ctrl_est ols1_est ols2_est aipw_est aipw_tst {
                return scalar `dvar'_`result' = r(`dvar'_`result')
        }
}

end

global storeresults ""
foreach dvar in $all_dvars {
        foreach result in ctrl_est ols1_est ols2_est aipw_est aipw_tst {
                global storeresults "$storeresults `dvar'_`result' = r(`dvar'_`result')"
        }
}

timer on 1

//simulate $storeresults, reps(2000) saving("../../store/rand_dist.dta", replace every
    (10)): randx

parallel initialize
parallel sim, expr($storeresults) reps(2000) randtype(current) saving("../../store/
    rand_dist.dta", replace every(10)): randx
parallel clean

timer off 1
timer list 1
timer clear 1
```

end randomization_dist.do

While Stata does allow users to store various kinds of simulation results in `.dta` format, this approach is relatively not as flexible as R's lists feature. Whereas R stores simulation output as a list of 2000 data frames (each of which is a 5 by 19 matrix) for the 2000 simulations, Stata stores the simulation results as a dataset of 2000 rows (i.e., one row per simulation) and 95 (i.e., 5 times 19) columns. Thus, R's approach is a bit more flexible than Stata's approach in this respect. To implement stratified block bootstrap, the R source file `bootstrap_dist.R` and the Stata do-file `bootstrap_dist.do`, which are provided in the Appendix, adapt the previous code templates to simulate the desired bootstrap distributions.

Once the original ATE estimates as well as their bootstrap and randomization distribution simulations are saved in the subfolder `../store`, they can be used to calculate the desired statistics for reporting in the final tables. The Stata do-file `appended_results.do` shown below does this and puts the calculated statistics in the form of a dataset so that they can be retrieved easily as needed during the later step of table creation. (The R counterpart of the do-file is `appended_results.R` and is provided in the Appendix.) The following code first collects the original ATE estimates and stores them in a temporary file, before using the simulated bootstrap distribution to compute the bootstrap standard errors (for the three ATE estimates) and storing them in another temporary file. The following statistics are also computed and stored in separate temporary files: exact p-values based on nonstudentized test statistics (based on simple OLS, adjusted OLS, and augmented IPW methods) as well as exact single and stepdown p-values based on studentized AIPW test statistic. This results in 12 temporary files for the aforementioned twelve types of statistics. These files are then appended to form a single dataset that has 19 columns (one for each outcome) and 12 rows (one for each type of statistic). This dataset, named `appended_results`, is again stored in `../store` so that it can be used for creating customized tables.

begin `appended_results.do`

```
*** Estimates ***

foreach estim in ctrl ols1 ols2 aipw {
        use "../../store/obs_estimates.dta", clear
        keep *_`estim'_est
        rename *_`estim'_est *

        tempfile estimate_`estim'
        save "`estimate_`estim''"
}

*** Bootstrap standard errors ***

foreach estim in ols1 ols2 aipw {
        use "../../store/boot_dist.dta", clear
        keep *_`estim'_est
```

```
        rename *_`estim'_est *
        collapse (sd) *

        tempfile boot_se_`estim'
        save "`boot_se_`estim''"
}

*** Exact p-values for nonstudentized test statistics ***

foreach estim in ols1 ols2 aipw {
        use "../../store/obs_estimates.dta", clear
        append using "../../store/rand_dist.dta"
        keep *_`estim'_est
        rename *_`estim'_est *
        ds
        local variable_list "`r(varlist)'"
        foreach var in `variable_list' {
                replace `var' = abs(`var')
                gen temp_`var' = (`var' >= `var'[1])
                drop `var'
                rename temp_`var' `var'
        }
        collapse (mean) *

        tempfile exact_p_`estim'_est
        save "`exact_p_`estim'_est'"
}

*** Exact p-values for studentized AIPW test statistic ***

foreach estim in aipw {
        use "../../store/obs_estimates.dta", clear
        append using "../../store/rand_dist.dta"
        keep *_`estim'_tst
        rename *_`estim'_tst *
        ds
        local variable_list "`r(varlist)'"
        foreach var in `variable_list' {
                replace `var' = abs(`var')
                gen temp_`var' = (`var' >= `var'[1])
                drop `var'
                rename temp_`var' `var'
        }
        collapse (mean) *

        tempfile exact_p_`estim'_tst
        save "`exact_p_`estim'_tst'"
}


*** Stepdown p-values ***

do ../programs/stepdown_pval.do

use "../../store/obs_estimates.dta", clear
```

```
append using "../../store/rand_dist.dta"
keep *_aipw_tst
rename *_aipw_tst *
ds
local variable_list "`r(varlist)'"
foreach var in `variable_list' {
        replace `var' = abs(`var')
}


stepdown_pval "6" ///
"Business_Expenditures Non_Business_Exp New_Business_Ap15" ///
"Profit ln_Q50 Capital" ///
"Late_Days_364 Late_Days_476 not_finished_aug19 Outstanding_Loan" ///
"Fifty_Percent_Loan_Paid Made_First_11 Made_First_Pay" ///
"atleastone Max_Min Q68" ///
"Q35_ Q37_ Q11_Together_max"

tempfile stepdown_p_aipw
save "`stepdown_p_aipw'"

*** Appended results ***

use "`estimate_ctrl'", clear

foreach estim in ols1 ols2 aipw {
        append using "`estimate_`estim''"
        append using "`boot_se_`estim''"
        append using "`exact_p_`estim'_est'"
}

append using "`exact_p_aipw_tst'"
append using "`stepdown_p_aipw'"

save "../../store/appended_results.dta", replace
```

end `appended_results.do`

6 Automating Formatted Creation of Customized Tables

In order to avoid manually making changes to both the content and the formatting of the desired tables, it is possible to automate creation of customized tables using R or Stata, ensuring reproducibility (and automatic updating) of the tables inserted in the manuscript. In Stata, one way to do this is first to retrieve and store (e.g., as `local` macros) the desired numbers from the dataset containing the appended results. Then, `for` loops (and `if`-`else` conditional statements) can be used to create desired formatting. Finally, the `file write` command (preceded by `file open` and succeeded by `file close` command) can be used to create and modify tables in `.tex` file format (or other types of desired file formats). In R, the `writeLines` function can be used to create an empty `.tex` file, which can be modified using the `write` function with the `append=TRUE` option.

(A caveat in R is that two backslashes, i.e., `\\`, should be used in strings within the `write` function to produce a single backslash `\` in the `.tex` output file.) To recall, the desired table formats are:

Desired format for table on impact of grace period on business activity and repayment outcomes

Outcome	*Control mean*	*Simple OLS estimate of ATE*	*Adjusted OLS estimate of ATE*	*Augmented IPW estimate of ATE*
$(i.j)$ Outcome label	$\widehat{\alpha}_{i.j}$	$\widehat{\delta}_{i.j}$ $(\widetilde{\sigma}_{\delta,i.j})$ $[\tilde{p}^{\circ}_{\delta,i.j}]$	$\widehat{\theta}_{i.j}$ $(\widetilde{\sigma}_{\theta,i.j})$ $[\tilde{p}^{\circ}_{\theta,i.j}]$	$\widehat{\tau}_{i.j}$ $(\widetilde{\sigma}_{\tau,i.j})$ $[\tilde{p}^{\circ}_{\tau,i.j}]$

Desired table format for stepdown inference on impacts of grace period for microfinance loans

Outcome	*Control mean*	*Augmented IPW estimate of ATE*	*Nonstudentized test-based exact single p-value*	*Studentized test-based exact single p-value*	*Studentized test-based exact stepdown p-value*
$(i.j)$ Outcome label	$\widehat{\alpha}_{i.j}$	$\widehat{\tau}_{i.j}$	$\tilde{p}^{\circ}_{\tau,i.j}$	$\tilde{q}^{\circ}_{\tau,i.j}$	$\tilde{q}^{\star}_{\tau,i.j}$

A table in the first format can be created using the code files `table_single_pvals.do` and `table_single_pvals.R` provided in the Appendix. A table in the second format can be created using the below code files: `table_stepdown_pvals.do` and `table_stepdown_pvals.R`.

begin `table_stepdown_pvals.do`

```
clear all
set more off
version 17.0

use "../../store/appended_results.dta", clear
do "../simulations/outcomes.do"

global note "\textit{Note}: For each outcome of interest, this table reports the control
     mean, augmented inverse probability weighting (AIPW) estimate of the average
    treatment effect (ATE), and the following three $ p$-values for testing sharp null
    hypotheses of no treatment effects: exact single $ p$-value based on the
    nonstudentized AIPW test statistic; exact single $ p$-value based on the studentized
     AIPW test statistic; and exact stepdown $ p$-value based on the studentized AIPW
    test statistic. The latter $ p$-value (i.e., the stepdown $ p$-value) accounts for
    multiple testing but not the former two single $ p$-values. The blocks of outcomes
    used for multiple testing are separated in the above table using a horizontal
    divider line."

local l_Business_Expenditures "Total business spending"
local l_Non_Business_Exp "Total nonbusiness spending"
local l_New_Business_Ap15 "New business"
local l_Profit "Average weekly profits"
local l_ln_Q50 "Log of monthly household income"
local l_Capital "Capital"
```

```
local l_Late_Days_364 "Not repaid 8 weeks after due date"
local l_Late_Days_476 "Not repaid 24 weeks after due date"
local l_not_finished_aug19 "Not repaid 52 weeks after due date"
local l_Outstanding_Loan "Outstanding loan 52 weeks after due date"
local l_Fifty_Percent_Loan_Paid "Repaid at least 50\% of loan"
local l_Made_First_11 "Made first half of repayments on time"
local l_Made_First_Pay "Made first payment"
local l_atleastone "Business closure"
local l_Max_Min "Profit range length"
local l_Q68 "Reapid by selling items at discount"
local l_Q35_ "Customers buy on credit"
local l_Q37_ "Customers pre-order items"
local l_Q11_Together_max "Number of items for sale"

file open latex_table using "../../tables/table_stepdown_pvals_stata.tex", write replace

local alignment_line "l|cc|ccc"

file write latex_table "\begin{table}[!ht]" _n
file write latex_table "\begin{center}" _n
file write latex_table "\caption{\textit{\textbf{Stepdown inference on impacts of grace
    period for microfinance loans (using Stata)}}}" _n
file write latex_table "\label{table:table_stepdown_pvals_stata}" _n
file write latex_table "\scriptsize \vspace{2mm}" _n
file write latex_table "\setstretch{1.5}" _n
file write latex_table "\begin{tabular}{`alignment_line'}" _n
file write latex_table "\hline\hline" _n

local header_line1 " &  &  & \textit{Nonstudentized}  & \textit{Studentized} & \textit{
    Studentized} "
local header_line2 " & \textit{Control} & \textit{Augmented IPW} & \textit{test-based
    exact} & \textit{test-based exact} & \textit{test-based exact} "
local header_line3 " \textit{Outcome} & \textit{mean} & \textit{estimate of ATE} & \
    textit{single $ p$-value} & \textit{single $ p$-value} & \textit{stepdown $ p$-value
    } "

foreach dvar in $all_dvars {
        local `dvar'_ctrl = `dvar'[1]
        local row = 7
        foreach stat in est bse epn eps sdp {
                local row = `row' + 1
                local `dvar'_`stat' = `dvar'[`row']
                if ``dvar'_`stat'' <= 0.1 {
                        local `dvar'_`stat'_l "\mathbf {"
                        local `dvar'_`stat'_r "}"
                }
                else {
                        local `dvar'_`stat'_l ""
                        local `dvar'_`stat'_r ""
                }
        }


        local `dvar'_line "`l_`dvar'' & $ `: di %5.2f ``dvar'_ctrl' ' $ & $ `: di %5.2f
            ``dvar'_est' ' $"
```

```
        foreach stat in epn eps sdp {
                local `dvar'_line "``dvar'_line' & $ ``dvar'_`stat'_l' `: di %5.4f ``
                    dvar'_`stat'' ' ``dvar'_`stat'_r' $"
        }

}

file write latex_table "`header_line1' \\ [-1mm]" _n
file write latex_table "`header_line2' \\ [-1mm]" _n
file write latex_table "`header_line3' \\ \hline " _n

local blocknum = 0
foreach block in loan_use profits default repayment business customers {
        local dvarnum = 0
        local blocknum = `blocknum' + 1
        foreach dvar in ${`block'_vars} {
                local dvarnum = `dvarnum' + 1
                file write latex_table " $ (`blocknum'.`dvarnum') $ ``dvar'_line' \\ "
                    _n
        }
        file write latex_table "\hline" _n
}

file write latex_table "\hline" _n
file write latex_table "\end{tabular}" _n
file write latex_table "\end{center} \vspace{-2mm}" _n
file write latex_table "\setstretch{1}\noindent \scriptsize" _n
file write latex_table "$note" _n
file write latex_table "\end{table}" _n

file close latex_table
```

end table_stepdown_pvals.do

begin table_stepdown_pvals.R

```
loan_use_vars <- c("Business_Expenditures", "Non_Business_Exp", "New_Business_Ap15")
profits_vars <- c("Profit", "ln_Q50", "Capital")
default_vars <- c("Late_Days_364", "Late_Days_476",
                "not_finished_aug19", "Outstanding_Loan_Amount_Default")
repayment_vars <- c("Fifty_Percent_Loan_Paid", "Made_First_11_Pay_On_Time",
                    "Made_First_Pay")
business_vars <- c("atleastone_bizshutdown_alt", "Max_Min", "Q68")
customers_vars <- c("Q35_", "Q37_", "Q11_Together_max")
all_dvars <- c(loan_use_vars, profits_vars, default_vars,
               repayment_vars,business_vars,customers_vars)
block_names <- c("loan_use_vars", "profits_vars", "default_vars",
                 "repayment_vars", "business_vars", "customers_vars")

library(dplyr, warn.conflicts = FALSE)

ar <- readRDS("../../store/appended_results")

l_Business_Expenditures <- "Total business spending"
l_Non_Business_Exp <- "Total nonbusiness spending"
```

```
l_New_Business_Ap15 <- "New business"
l_Profit <- "Average weekly profits"
l_ln_Q50 <- "Log of monthly household income"
l_Capital <- "Capital"
l_Late_Days_364 <- "Not repaid 8 weeks after due date"
l_Late_Days_476 <- "Not repaid 24 weeks after due date"
l_not_finished_aug19 <- "Not repaid 52 weeks after due date"
l_Outstanding_Loan_Amount_Default <- "Outstanding loan 52 weeks after due date"
l_Fifty_Percent_Loan_Paid <- "Repaid at least 50\\% of loan"
l_Made_First_11_Pay_On_Time <- "Made first half of repayments on time"
l_Made_First_Pay <- "Made first payment"
l_atleastone_bizshutdown_alt <- "Business closure"
l_Max_Min <- "Profit range length"
l_Q68 <- "Reapid by selling items at discount"
l_Q35_ <- "Customers buy on credit"
l_Q37_ <- "Customers pre-order items"
l_Q11_Together_max <- "Number of items for sale"

latex_lines <- c("\\begin{table}[!ht]", "\\begin{center}",
"\\caption{\\textit{\\textbf{Stepdown inference on impacts of grace period for
    microfinance loans (using R)}}}",
"\\label{table:table_stepdown_pvals_r}", "\\scriptsize \\vspace{2mm}",
"\\setstretch{1.5}",
"\\begin{tabular}{l|cc|ccc}",
"\\hline\\hline",
"&  &  & \\textit{Nonstudentized}  & \\textit{Studentized} & \\textit{Studentized} \\\\
    [-1mm]",
" & \\textit{Control} & \\textit{Augmented IPW} & \\textit{test-based exact} & \\textit{
    test-based exact} & \\textit{test-based exact} \\\\ [-1mm]",
"\\textit{Outcome} & \\textit{mean} & \\textit{estimate of ATE} & \\textit{single $ p$-
    value} & \\textit{single $ p$-value} & \\textit{stepdown $ p$-value} \\\\ \\hline")

fileTeX <-file("../../tables/table_stepdown_pvals_r.tex")
writeLines(latex_lines, fileTeX)
close(fileTeX)

blocknum <- 0
for (i in 1:length(block_names)) {
  dvarnum <- 0
  blocknum <- blocknum + 1
  for (dvar in get(block_names[i])) {
    dvarnum <- dvarnum + 1

    dvar_line <- paste0(" $ (",blocknum,".",dvarnum,") $ ",
                        get(paste0("l_",dvar)),"  & $ ",
                        format(ar["ctrl_est",dvar], digits=1, nsmall=2)," $ ")


      dvar_line <- paste0(dvar_line," & $ ",
        format(ar["aipw_est",dvar], digits=1, nsmall=2)," $ ")

      for (stat in c("aipw_epv","saipw_epv","saipw_spv")) {
        epl <- ifelse(ar[all_of(stat),dvar] <= 0.1," \\mathbf { ","")
        epr <- ifelse(ar[all_of(stat),dvar] <= 0.1," } ","")
        dvar_line <- paste0(dvar_line," & $ ",epl,
```

```
                            format(ar[all_of(stat),dvar], digits=0, nsmall=4),epr," $")
      }


    latex_lines <- paste0(dvar_line,"\\\\")
    write(latex_lines,file="../../tables/table_stepdown_pvals_r.tex",append=TRUE)
  }
  write("\\hline",file="../../tables/table_stepdown_pvals_r.tex",append=TRUE)
}

latex_lines <- c("\\hline","\\end{tabular}","\\end{center} \\vspace{-2mm}",
                 "\\setstretch{1}\\noindent \\scriptsize",
                 "\\textit{Note}: For each outcome of interest, this table reports the
                     control mean, augmented inverse probability weighting (AIPW)
                     estimate of the average treatment effect (ATE), and the following
                     three $ p$-values for testing sharp null hypotheses of no treatment
                      effects: exact single $ p$-value based on the nonstudentized AIPW
                     test statistic; exact single $ p$-value based on the studentized
                     AIPW test statistic; and exact stepdown $ p$-value based on the
                     studentized AIPW test statistic. The latter $ p$-value (i.e., the
                     stepdown $ p$-value) accounts for multiple testing but not the
                     former two single $ p$-values. The blocks of outcomes used for
                     multiple testing are separated in the above table using a
                     horizontal divider line.","\\end{table}")
write(latex_lines,file="../../tables/table_stepdown_pvals_r.tex",append=TRUE)

rm(list = ls(all.names = TRUE))
```

end `table_stepdown_pvals.R`

Above Stata and R codes produce the `.tex` files `table_stepdown_pvals_stata.tex` and `table_stepdown_pvals_r.tex`, respectively, in the subfolder `../tables`. Such `.tex` files containing the desired tables can be produced using just a single run of `master.do` or `master.R` in the `../scripts` subfolder. These TeX tables can be easily included in the main manuscript files using a simple line, e.g., `\input {../tables/table_stepdown_pvals_stata.tex}`, without having to manually make any formatting changes to the tables in the manuscript. This approach also ensures that any updates made to the table results will also be automatically reflected in the manuscript.

To illustrate the output of the code files for the worked example, Tables 1 and 2, which are similar up to simulation error, present stepdown inference based on the augmented IPW estimates, while Tables 3 and 4 present three ATE estimates along with their bootstrap standard errors and exact p-values for each outcome. All of these supplementary results, and especially those in Tables 1 and 2 based on doubly robust estimates and design-based finite-sample inferences corrected for multiple testing, bolster Field et al.'s (2013) conclusion that "debt contracts that require early repayment discourage illiquid risky investment and thereby limit the potential impact of microfinance on microenterprise growth and household poverty."

Table 1: ***Stepdown inference on impacts of grace period for microfinance loans (using Stata)***

Outcome	*Control mean*	*Augmented IPW estimate of ATE*	*Nonstudentized test-based exact single p-value*	*Studentized test-based exact single p-value*	*Studentized test-based exact stepdown p-value*
(1.1) Total business spending	6278.81	426.98	**0.0435**	**0.0465**	**0.0465**
(1.2) Total nonbusiness spending	1119.35	−458.99	**0.0115**	**0.0150**	**0.0325**
(1.3) New business	0.02	0.03	**0.0165**	**0.0170**	**0.0430**
(2.1) Average weekly profits	1586.80	923.61	**0.0215**	**0.0225**	**0.0290**
(2.2) Log of monthly household income	9.32	0.20	**0.0145**	**0.0145**	**0.0290**
(2.3) Capital	35730.16	38151.95	**0.0155**	**0.0175**	**0.0290**
(3.1) Not repaid 8 weeks after due date	0.04	0.08	**0.0275**	**0.0265**	**0.0540**
(3.2) Not repaid 24 weeks after due date	0.02	0.06	**0.0640**	**0.0555**	**0.0850**
(3.3) Not repaid 52 weeks after due date	0.02	0.06	**0.0620**	**0.0490**	**0.0790**
(3.4) Outstanding loan 52 weeks after due date	69.65	126.66	0.2144	0.2009	0.2009
(4.1) Repaid at least 50% of loan	0.99	−0.01	0.7011	0.6932	0.9045
(4.2) Made first half of repayments on time	0.50	−0.02	0.7086	0.7146	0.9045
(4.3) Made first payment	0.95	0.03	0.3328	0.3338	0.7231
(5.1) Business closure	0.39	−0.06	0.1499	0.1479	0.3723
(5.2) Profit range length	2361.63	466.49	0.4043	0.3893	0.6102
(5.3) Reapid by selling items at discount	0.05	−0.01	0.4048	0.3923	0.6102
(6.1) Customers buy on credit	0.43	0.10	**0.0095**	**0.0085**	**0.0205**
(6.2) Customers pre-order items	0.40	0.10	**0.0225**	**0.0225**	**0.0280**
(6.3) Number of items for sale	5.61	6.67	**0.0720**	**0.0870**	**0.0870**

Note: For each outcome of interest, this table reports the control mean, augmented inverse probability weighting (AIPW) estimate of the average treatment effect (ATE), and the following three p-values for testing sharp null hypotheses of no treatment effects: exact single p-value based on the nonstudentized AIPW test statistic; exact single p-value based on the studentized AIPW test statistic; and exact stepdown p-value based on the studentized AIPW test statistic. The latter p-value (i.e., the stepdown p-value) accounts for multiple testing but not the former two single p-values. The blocks of outcomes used for multiple testing are separated in the above table using a horizontal divider line.

Table 2: ***Stepdown inference on impacts of grace period for microfinance loans (using R)***

Outcome	*Control mean*	*Augmented IPW estimate of ATE*	*Nonstudentized test-based exact single p-value*	*Studentized test-based exact single p-value*	*Studentized test-based exact stepdown p-value*
(1.1) Total business spending	6278.81	426.98	**0.0440**	**0.0505**	**0.0510**
(1.2) Total nonbusiness spending	1119.35	−458.99	**0.0205**	**0.0220**	**0.0435**
(1.3) New business	0.02	0.03	**0.0220**	**0.0230**	**0.0505**
(2.1) Average weekly profits	1586.80	923.61	**0.0235**	**0.0240**	**0.0270**
(2.2) Log of monthly household income	9.32	0.20	**0.0180**	**0.0175**	**0.0315**
(2.3) Capital	35730.16	38151.95	**0.0140**	**0.0185**	**0.0315**
(3.1) Not repaid 8 weeks after due date	0.04	0.08	**0.0315**	**0.0295**	**0.0600**
(3.2) Not repaid 24 weeks after due date	0.02	0.06	**0.0655**	**0.0575**	**0.0850**
(3.3) Not repaid 52 weeks after due date	0.02	0.06	**0.0615**	**0.0525**	**0.0775**
(3.4) Outstanding loan 52 weeks after due date	69.65	126.66	0.2050	0.1915	0.1919
(4.1) Repaid at least 50% of loan	0.99	−0.008	0.6780	0.6660	0.8836
(4.2) Made first half of repayments on time	0.50	−0.02	0.6980	0.7025	0.8836
(4.3) Made first payment	0.95	0.03	0.3410	0.3400	0.6987
(5.1) Business closure	0.39	−0.06	0.1380	0.1350	0.3638
(5.2) Profit range length	2361.63	466.49	0.3815	0.3700	0.6002
(5.3) Reapid by selling items at discount	0.05	−0.01	0.4060	0.3970	0.6002
(6.1) Customers buy on credit	0.43	0.10	**0.0090**	**0.0080**	**0.0155**
(6.2) Customers pre-order items	0.40	0.10	**0.0175**	**0.0155**	**0.0220**
(6.3) Number of items for sale	5.61	6.67	**0.0695**	**0.0760**	**0.0765**

Note: For each outcome of interest, this table reports the control mean, augmented inverse probability weighting (AIPW) estimate of the average treatment effect (ATE), and the following three p-values for testing sharp null hypotheses of no treatment effects: exact single p-value based on the nonstudentized AIPW test statistic; exact single p-value based on the studentized AIPW test statistic; and exact stepdown p-value based on the studentized AIPW test statistic. The latter p-value (i.e., the stepdown p-value) accounts for multiple testing but not the former two single p-values. The blocks of outcomes used for multiple testing are separated in the above table using a horizontal divider line.

Table 3: ***Impact of grace period on business activity and repayment outcomes (using Stata)***

Outcome	*Control mean*	*Simple OLS estimate of ATE*	*Adjusted OLS estimate of ATE*	*Augmented IPW estimate of ATE*
(1.1) Total business spending	6278.81	364.89 (173.59) [**0.0520**]	383.92 (179.73) [**0.0450**]	426.98 (244.73) [**0.0435**]
(1.2) Total nonbusiness spending	1119.35	−356.08 (167.77) [**0.0380**]	−371.60 (174.51) [**0.0320**]	−458.99 (227.44) [**0.0115**]
(1.3) New business	0.02	0.03 (0.01) [**0.0570**]	0.03 (0.01) [**0.0770**]	0.03 (0.02) [**0.0165**]
(2.1) Average weekly profits	1586.80	906.57 (358.47) [**0.0060**]	902.91 (374.57) [**0.0080**]	923.61 (536.53) [**0.0215**]
(2.2) Log of monthly household income	9.32	0.19 (0.08) [**0.0155**]	0.20 (0.08) [**0.0120**]	0.20 (0.09) [**0.0145**]
(2.3) Capital	35730.16	28770.20 (10919.67) [**0.0075**]	35733.14 (13271.83) [**0.0010**]	38151.95 (18302.87) [**0.0155**]
(3.1) Not repaid 8 weeks after due date	0.04	0.09 (0.03) [**0.0145**]	0.08 (0.03) [**0.0205**]	0.08 (0.04) [**0.0275**]
(3.2) Not repaid 24 weeks after due date	0.02	0.07 (0.03) [**0.0190**]	0.06 (0.03) [**0.0300**]	0.06 (0.03) [**0.0640**]
(3.3) Not repaid 52 weeks after due date	0.02	0.06 (0.02) [**0.0285**]	0.06 (0.02) [**0.0320**]	0.06 (0.03) [**0.0620**]
(3.4) Outstanding loan 52 weeks after due date	69.65	148.69 (80.63) [0.1064]	148.99 (81.34) [0.1219]	126.66 (91.47) [0.2144]
(4.1) Repaid at least 50% of loan	0.99	−0.01 (0.01) [0.5207]	−0.02 (0.02) [0.4093]	−0.01 (0.02) [0.7011]
(4.2) Made first half of repayments on time	0.50	−0.01 (0.06) [0.9130]	−0.02 (0.05) [0.6802]	−0.02 (0.06) [0.7086]
(4.3) Made first payment	0.95	0.03 (0.03) [0.3573]	0.02 (0.02) [0.3738]	0.03 (0.03) [0.3328]
(5.1) Business closure	0.39	−0.07 (0.03) [**0.0375**]	−0.07 (0.03) [**0.0625**]	−0.06 (0.04) [0.1499]
(5.2) Profit range length	2361.63	686.63 (364.71) [**0.0820**]	713.87 (393.05) [**0.0785**]	466.49 (482.00) [0.4043]
(5.3) Reapid by selling items at discount	0.05	−0.02 (0.01) [**0.0995**]	−0.02 (0.01) [0.2459]	−0.01 (0.02) [0.4048]
(6.1) Customers buy on credit	0.43	0.10 (0.04) [**0.0100**]	0.11 (0.04) [**0.0025**]	0.10 (0.05) [**0.0095**]
(6.2) Customers pre-order items	0.40	0.10 (0.03) [**0.0110**]	0.11 (0.03) [**0.0065**]	0.10 (0.04) [**0.0225**]
(6.3) Number of items for sale	5.61	5.54 (2.44) [**0.0050**]	6.05 (2.66) [**0.0060**]	6.67 (4.07) [**0.0720**]

Note: For each outcome of interest, the above table reports the control mean and three ATE estimates (based on simple OLS, adjusted OLS, and augmented IPW methods) along with their bootstrap standard errors in parentheses (next to the estimates). Below each ATE estimate is its exact p-value (reported within brackets) based on a randomization test using the raw estimate as the test statistic without studentization.

Table 4: *Impact of grace period on business activity and repayment outcomes (using R)*

Outcome	*Control mean*	*Simple OLS estimate of ATE*	*Adjusted OLS estimate of ATE*	*Augmented IPW estimate of ATE*
(1.1) Total business spending	6278.81	364.89 (176.51) [**0.0560**]	383.92 (183.40) [**0.0485**]	426.98 (241.94) [**0.0440**]
(1.2) Total nonbusiness spending	1119.35	−356.08 (171.44) [**0.0455**]	−371.60 (178.76) [**0.0390**]	−458.99 (226.87) [**0.0205**]
(1.3) New business	0.02	0.03 (0.01) [**0.0595**]	0.03 (0.01) [**0.0710**]	0.03 (0.02) [**0.0220**]
(2.1) Average weekly profits	1586.80	906.57 (364.54) [**0.0040**]	902.91 (374.67) [**0.0040**]	923.61 (539.09) [**0.0235**]
(2.2) Log of monthly household income	9.32	0.19 (0.08) [**0.0140**]	0.20 (0.08) [**0.0090**]	0.20 (0.09) [**0.0180**]
(2.3) Capital	35730.16	28770.19 (11123.63) [**0.0065**]	35733.14 (13393.51) [**0.0010**]	38151.95 (18367.38) [**0.0140**]
(3.1) Not repaid 8 weeks after due date	0.04	0.09 (0.03) [**0.0160**]	0.08 (0.03) [**0.0225**]	0.08 (0.04) [**0.0315**]
(3.2) Not repaid 24 weeks after due date	0.02	0.07 (0.03) [**0.0140**]	0.06 (0.03) [**0.0305**]	0.06 (0.03) [**0.0655**]
(3.3) Not repaid 52 weeks after due date	0.02	0.06 (0.02) [**0.0220**]	0.06 (0.02) [**0.0265**]	0.06 (0.03) [**0.0615**]
(3.4) Outstanding loan 52 weeks after due date	69.65	148.69 (80.92) [0.1030]	148.99 (83.54) [0.1120]	126.66 (91.64) [0.2050]
(4.1) Repaid at least 50% of loan	0.99	−0.01 (0.01) [0.4315]	−0.02 (0.02) [0.3660]	−0.008 (0.02) [0.6780]
(4.2) Made first half of repayments on time	0.50	−0.008 (0.06) [0.9000]	−0.02 (0.05) [0.6665]	−0.02 (0.06) [0.6980]
(4.3) Made first payment	0.95	0.03 (0.02) [0.3510]	0.02 (0.02) [0.3785]	0.03 (0.03) [0.3410]
(5.1) Business closure	0.39	−0.07 (0.03) [**0.0325**]	−0.07 (0.03) [**0.0610**]	−0.06 (0.04) [0.1380]
(5.2) Profit range length	2361.63	686.63 (363.09) [**0.0845**]	713.87 (396.78) [**0.0820**]	466.49 (488.63) [0.3815]
(5.3) Reapid by selling items at discount	0.05	−0.02 (0.01) [0.1090]	−0.02 (0.01) [0.2435]	−0.01 (0.02) [0.4060]
(6.1) Customers buy on credit	0.43	0.10 (0.04) [**0.0165**]	0.11 (0.04) [**0.0010**]	0.10 (0.04) [**0.0090**]
(6.2) Customers pre-order items	0.40	0.10 (0.03) [**0.0030**]	0.11 (0.03) [**0.0035**]	0.10 (0.04) [**0.0175**]
(6.3) Number of items for sale	5.61	5.54 (2.47) [**0.0075**]	6.05 (2.57) [**0.0040**]	6.67 (4.06) [**0.0695**]

Note: For each outcome of interest, the above table reports the control mean and three ATE estimates (based on simple OLS, adjusted OLS, and augmented IPW methods) along with their bootstrap standard errors in parentheses (next to the estimates). Below each ATE estimate is its exact p-value (reported within brackets) based on a randomization test using the raw estimate as the test statistic without studentization.

References

Athey, S., G. Imbens, T. Pham, and S. Wager (2017). Estimating average treatment effects: Supplementary analyses and remaining challenges. *American Economic Review: Papers & Proceedings 107*(5), 278–281.

Ding, P. and F. Li (2018). Causal inference: A missing data perspective. *Statistical Science 33*(2), 214–237.

Field, E., R. Pande, J. Papp, and N. Rigol (2013). Does the classic microfinance model discourage entrepreneurship among the poor? Experimental evidence from India. *American Economic Review 103*(6), 2196–2226.

Lunceford, J. K. and M. Davidian (2004). Stratification and weighting via the propensity score in estimation of causal treatment effects: A comparative study. *Statistics in Medicine 23*(19), 2937–2960.

Romano, J. P. and M. Wolf (2005). Exact and approximate stepdown methods for multiple hypothesis testing. *Journal of the American Statistical Association 100*(469), 94–108.

Romano, J. P. and M. Wolf (2016). Efficient computation of adjusted p-values for resampling-based stepdown multiple testing. *Statistics & Probability Letters 113*, 38–40.

Wu, J. and P. Ding (2020). Randomization tests for weak null hypotheses in randomized experiments. Forthcoming in the *Journal of the American Statistical Association*.

Appendix

begin aipw_strata.R

```
aipw_strata <- function(dataset,...) {

  require("dplyr")
  tempdata <- select(dataset,...)
  colnames(tempdata) <- c("yvar","tvar","svar",colnames(tempdata)[4:ncol(tempdata)])

  tempdata <- tempdata %>% group_by(svar) %>% mutate(pr_treat = mean(tvar))
  tempdata$yhat_1 <- suppressWarnings(predict(lm(as.formula(paste("yvar", "~", paste(
      colnames(tempdata)[4:(ncol(tempdata)-1)], collapse = " + "), sep="")), data =
      tempdata[tempdata$tvar==1,]), newdata = tempdata))
  tempdata$yhat_0 <- suppressWarnings(predict(lm(as.formula(paste("yvar", "~", paste(
      colnames(tempdata)[4:(ncol(tempdata)-2)], collapse = " + "), sep="")), data =
      tempdata[tempdata$tvar==0,]), newdata = tempdata))
  tempdata <- tempdata %>% mutate(aipw_te = yhat_1 + (tvar/pr_treat) * (yvar - yhat_1)
                                  - (yhat_0 + ((1 - tvar)/(1 - pr_treat)) * (yvar -
                                      yhat_0)))
  return(summary(lm(aipw_te ~ 1, data = tempdata))$coefficients[1,c(1,2)])

}
```

end aipw_strata.R

begin stepdown_pval.do

```
ssc install rowranks, replace

program stepdown_pval

        local nvarlists "`1'"
        forval i = 1/`nvarlists' {
                local stepdown_dvlist`i' "``=`i'+1'"
                di "`stepdown_dvlist`i''"
        }
        forval j = 1/`nvarlists' {
                global temp_vvlist ""
                global temp_vlist `stepdown_dvlist`j''
                foreach var in $temp_vlist {
                        global temp_vvlist "$temp_vvlist `var'"
                }
                global vlist`j' $temp_vvlist
        }

        forval j = 1/`nvarlists' {

                qui {

                        local varlist: display "${vlist`j'}"
                        local numoutcomes: word count `varlist'
                        di `numoutcomes'
```

```
foreach var in ‘varlist’ {
        gen t_‘var’ = ‘var’
}

foreach var in ‘varlist’ {
        if t_‘var’[1] < 0 {
                replace t_‘var’ = - t_‘var’
        }
}

local tvarlist ""
foreach var in ‘varlist’ {
        local tvarlist "‘tvarlist’ t_‘var’"
}
local tt0varlist ""
foreach var in ‘varlist’ {
        local tt0varlist "‘tt0varlist’ tt0_‘var’"
}
local ttvarlist ""
foreach var in ‘varlist’ {
        local ttvarlist "‘ttvarlist’ tt_‘var’"
}
//ssc install rowranks
rowranks ‘tvarlist’, gen(‘tt0varlist’) field lowrank
foreach tt0var in ‘tt0varlist’ {
        replace ‘tt0var’ = . if _n > 1
}

local incr = 0
foreach tt0var in ‘tt0varlist’ {
        local incr = ‘incr’ + 1/(3 * ‘numoutcomes’)
        replace ‘tt0var’ = ‘tt0var’ + ‘incr’
}

//the "3" is arbitrary.. anything above 1 should work

rowranks ‘tt0varlist’, gen(‘ttvarlist’)

drop ‘tt0varlist’
// tt0varlist was created and modified to avoid any ties
// after getting ttvarlist, we’re dropping tt0varlist


// use the ranks per se to generate p1, p2, ..., p6
// then, make a local to store rank of a variable
// then p_‘variable’ = p‘local_rank’


forval s = 1/‘numoutcomes’ {

        local currentvar ""

        foreach var in ‘varlist’ {
                replace tt_‘var’ = t_‘var’ if _n > 1 & tt_‘var
```

```
                                        '[1] >= `s'
                                if tt_`var'[1] == `s' {
                                        local currentvar "`var'"
                                }
                        }

                        egen maxtt = rowmax(`ttvarlist') if _n > 1

                        count if maxtt >= t_`currentvar'[1] & _n > 1
                        local p`s'init = (r(N) + 1)/(_N)

                        if `s' == 1 {
                                local p`s'adj = min(1,`p`s'init')
                        }

                        if `s' > 1 {
                                local sm1 = `s' - 1
                                local p`s'adj = min(1,max( `p`s'init' , `p`sm1'
                                    adj' ))
                        }

                        replace `currentvar' = `p`s'adj' if _n==1

                        foreach var in `varlist' {
                                replace tt_`var' = . if _n > 1
                        }

                        drop maxtt
                }

                drop t_* tt_*

        }

        di "Romano-Wolf stepdown procedure for block " `j' " complete"

    }


    drop if _n > 1

end
```

end stepdown_pval.do

begin outcomes.R

```
all_dvars <- c("Business_Expenditures", "Non_Business_Exp", "New_Business_Ap15",
               "Profit", "ln_Q50", "Capital", "Late_Days_364", "Late_Days_476",
               "not_finished_aug19", "Outstanding_Loan_Amount_Default",
               "Fifty_Percent_Loan_Paid", "Made_First_11_Pay_On_Time",
               "Made_First_Pay", "atleastone_bizshutdown_alt", "Max_Min", "Q68",
               "Q35_", "Q37_", "Q11_Together_max")
```

```
loan_use_vars <- c("Business_Expenditures", "Non_Business_Exp", "New_Business_Ap15")

profits_vars <- c("Profit", "ln_Q50", "Capital")

last_dvars <- c("Late_Days_364", "Late_Days_476",
                "not_finished_aug19", "Outstanding_Loan_Amount_Default",
                "Fifty_Percent_Loan_Paid", "Made_First_11_Pay_On_Time",
                "Made_First_Pay", "atleastone_bizshutdown_alt", "Max_Min", "Q68",
                "Q35_", "Q37_", "Q11_Together_max")

baseline_vars <- c("factor(Stratification_Dummies)", "factor(sec_loanamount)", "Age_C",
                   "Married_C", "Muslim_C", "HH_Size_C", "Years_Education_C",
                   "shock_any_C", "Has_Business_C", "Financial_Control_C",
                   "homeowner_C", "No_Drain_C")

miss_ind_vars <- c("miss_Age_C", "miss_Married_C", "miss_Literate_C",
                   "miss_Muslim_C", "miss_HH_Size_C", "miss_Years_Education_C",
                   "miss_shock_any_C", "miss_Has_Business_C",
                   "miss_Financial_Control_C", "miss_homeowner_C",
                   "miss_sec_loanamount", "miss_No_Drain_C")

baseline_xvars <- c("Age_C", "Married_C", "Muslim_C", "HH_Size_C", "Years_Education_C",
                    "shock_any_C", "Has_Business_C", "Financial_Control_C",
                    "homeowner_C", "No_Drain_C", "sec_loanamount", "loan_officer1",
                    "loan_officer2", "loan_officer3", "loan_officer4", "loan_amount1",
                    "loan_amount2", "loan_amount3", "stratum1", "stratum2", "stratum3",
                    "stratum4", "stratum5", "stratum6", "stratum7", "stratum8")
```

end outcomes.R

begin ate_estimators.R

```
source(file.path(getwd(),"../programs/aipw_strata.R"))
source(file.path(getwd(),"outcomes.R"))

ate_estimators <- function(dataset) {

require("dplyr")

ctrl_est <- NULL
i <- 0
for (dvar in all_dvars) {
  i <- i + 1
  ctrl_est[i] <- mean(dataset[dataset$sec_treat == 0,][[as.name(dvar)]], na.rm=TRUE)
  ### equivalent to:
  # assign(paste0(dvar,"_ctrl_est"),
  #        mean(dataset[dataset$sec_treat == 0,][[as.name(dvar)]], na.rm=TRUE))
  # ctrl_est[i] <- get(paste0(dvar,"_ctrl_est"))
}
rm(i,dvar)

loan_use_vars_ols1 <- unlist(lapply(loan_use_vars, function(x) {
  lm(substitute(i ~ sec_treat + factor(Stratification_Dummies) +
                  factor(sec_loanamount) + Match3rd_in3rd, list(i = as.name(x))),
```

```
    data = dataset)$coefficients[2]
}))


profits_vars_ols1 <- unlist(lapply(profits_vars, function(x) {
  lm(substitute(i ~ sec_treat + factor(Stratification_Dummies),
                list(i = as.name(x))), data = dataset)$coefficients[2]
}))

last_dvars_ols1 <- unlist(lapply(last_dvars, function(x) {
  lm(substitute(i ~ sec_treat + factor(Stratification_Dummies),
                list(i = as.name(x))), data = dataset)$coefficients[2]
}))

ols1_est <- unname(c(loan_use_vars_ols1, profits_vars_ols1, last_dvars_ols1))
rm(loan_use_vars_ols1, profits_vars_ols1, last_dvars_ols1)

loan_use_vars_ols2 <- NULL
i <- 0
for (dvar in loan_use_vars) {
  i <- i + 1
  loan_use_vars_ols2[i] <- lm(as.formula(paste(dvar, "~",
  paste(c("sec_treat", baseline_vars, miss_ind_vars, "Match3rd_in3rd"),
  collapse = " + "), sep=""")), data = dataset)$coefficients[2]
}

profits_vars_ols2 <- NULL
i <- 0
for (dvar in profits_vars) {
  i <- i + 1
  profits_vars_ols2[i] <- lm(as.formula(paste(dvar, "~",
  paste(c("sec_treat", baseline_vars, miss_ind_vars, "factor(sec_loan_officer)"),
  collapse = " + "), sep=""")), data = dataset)$coefficients[2]
}

last_dvars_ols2 <- NULL
i <- 0
for (dvar in last_dvars) {
  i <- i + 1
  last_dvars_ols2[i] <- lm(as.formula(paste(dvar, "~",
  paste(c("sec_treat", baseline_vars, miss_ind_vars, "factor(sec_loan_officer)",
  "Literate_C"), collapse = " + "), sep=""")), data = dataset)$coefficients[2]
}

ols2_est <- unname(c(loan_use_vars_ols2, profits_vars_ols2, last_dvars_ols2))
rm(loan_use_vars_ols2, profits_vars_ols2, last_dvars_ols2)

for (bvar in baseline_vars[3:length(baseline_vars)]) {
  dataset[dataset[,all_of(paste0("miss_",bvar))]==1, all_of(bvar)] <- NA
}
rm(bvar)

loanofficerids <- as.numeric(rownames(table(dataset$sec_loan_officer)))
for (i in 1:length(loanofficerids)) {
  dataset[,paste0("loan_officer",i)] <- ifelse(dataset$sec_loan_officer ==
```

```
      loanofficerids[i], 1, 0)
}

dataset$loan_amount1 <- ifelse(dataset$sec_loanamount %in% c(4000,5000), 1, 0)
dataset$loan_amount2 <- ifelse(dataset$sec_loanamount %in% c(6000,7000), 1, 0)
dataset$loan_amount3 <- ifelse(dataset$sec_loanamount %in% c(8000,9000), 1, 0)
dataset$loan_amount4 <- ifelse(dataset$sec_loanamount == 10000, 1, 0)

group_level_dataset <- dataset %>% group_by(sec_group_name) %>%
                       summarize_all(mean, na.rm = TRUE)

strata_ids <- as.numeric(rownames(table(dataset$Stratification_Dummies)))
for (i in 1:length(strata_ids)) {
  group_level_dataset[,paste0("stratum",i)] <-
    ifelse(group_level_dataset$Stratification_Dummies == strata_ids[i], 1, 0)
}

aipw_est <- NULL
aipw_tst <- NULL
i <- 0
for (dvar in all_dvars) {
  i <- i + 1
  aipw_stats_temp <- aipw_strata(group_level_dataset, as.name(dvar), sec_treat,
                                 Stratification_Dummies, all_of(baseline_xvars))
  aipw_est[i] <- aipw_stats_temp[1]
  aipw_tst[i] <- aipw_stats_temp[1]/aipw_stats_temp[2]
  rm(aipw_stats_temp)
}
rm(i,dvar)

temp_dataframe <- data.frame(matrix(ncol = length(all_dvars), nrow = 5))
temp_dataframe[1:5,] <- rbind(ctrl_est, ols1_est, ols2_est, aipw_est, aipw_tst)
rownames(temp_dataframe) <- c("ctrl_est", "ols1_est",
                              "ols2_est", "aipw_est", "aipw_tst")
colnames(temp_dataframe) <- all_dvars
return(temp_dataframe)

}
```

end ate_estimators.R

begin observed_estimates.do

```
global aer_103_6_2196_data: env aer_103_6_2196_data

do ate_estimators.do

program estimatesx, rclass

use $aer_103_6_2196_data/Grace-Period-Data.dta, clear

ate_estimators

foreach dvar in $all_dvars {
```

```
        foreach result in ctrl_est ols1_est ols2_est aipw_est aipw_tst {
                return scalar `dvar'_`result' = r(`dvar'_`result')
        }
}

end

global storeresults ""
foreach dvar in $all_dvars {
        foreach result in ctrl_est ols1_est ols2_est aipw_est aipw_tst {
                global storeresults "$storeresults `dvar'_`result' = r(`dvar'_`result')"
        }
}

timer on 1

simulate $storeresults, seed(12345) reps(1) saving("../../store/obs_estimates.dta",
    replace): estimatesx

timer off 1
timer list 1
timer clear 1
```

end observed_estimates.do

begin bootstrap_dist.R

```
ptm <- proc.time()

set.seed(12345)

library(parallel)
library(pbapply)

cl <- makeCluster(round(detectCores() * 0.75))

clusterEvalQ(cl, {

source(file.path(getwd(),"ate_estimators.R"))

library(dplyr, warn.conflicts = FALSE)
library(haven)

grace_period_data_path <- Sys.getenv("aer_103_6_2196_data")
gpdata <- read_dta(file=file.path(grace_period_data_path,"Grace-Period-Data.dta"))

gpdata_cluster_vars <- gpdata %>%
  select(sec_group_name, Stratification_Dummies, sec_treat) %>%
  group_by(sec_group_name) %>% filter(row_number() == 1)

table(gpdata_cluster_vars$sec_treat, gpdata_cluster_vars$Stratification_Dummies)

})
```

```
bootstrap_estimates <- function(...) {

  gpdata_temp <- data.frame(matrix(nrow = 1, ncol = ncol(gpdata)))
  colnames(gpdata_temp) <- colnames(gpdata)

  row_count <- 0

  for (i in unique(gpdata$Stratification_Dummies)) {
    for (j in unique(gpdata$sec_treat)) {
      cluster_vars_temp <- gpdata_cluster_vars %>%
        filter(Stratification_Dummies==i & sec_treat==j)
      vec_temp <- sample(cluster_vars_temp$sec_group_name, replace = TRUE)
      for (k in 1:length(vec_temp)) {
        gpdata_temp[(row_count + 1):(row_count + 5),] <- gpdata %>%
          filter(sec_group_name == vec_temp[k])
        row_count = row_count + 5
      }
    }
  }


  table(gpdata_temp$sec_treat, gpdata_temp$Stratification_Dummies)

  gpdata_temp$sec_group_name <- ceiling(c(1:nrow(gpdata_temp))/5)

  return(ate_estimators(gpdata_temp))

}

system.time(boot_dist <- pblapply(1:2000, bootstrap_estimates, cl = cl))

stopCluster(cl)

saveRDS(boot_dist, file = "../../store/boot_dist")

proc.time() - ptm

rm(list = ls(all.names = TRUE))
```

end bootstrap_dist.R

begin bootstrap_dist.do

```
set seed 12345

net install parallel, from("https://raw.github.com/gvegayon/parallel/stable/") replace
mata mata mlib index

global aer_103_6_2196_data: env aer_103_6_2196_data
global current_dir: pwd

do outcomes.do

program bootx, rclass
```

```
cap program drop aipw_strata
cap program drop ate_estimators
cd $current_dir
do ate_estimators.do

use $aer_103_6_2196_data/Grace-Period-Data.dta, clear

bsample, cluster(sec_group_name) strata(Stratification_Dummies sec_treat) idcluster(
    new_sec_group_name)
drop sec_group_name
gen sec_group_name = new_sec_group_name
drop new_sec_group_name

ate_estimators

foreach dvar in $all_dvars {
        foreach result in ctrl_est ols1_est ols2_est aipw_est aipw_tst {
                return scalar `dvar'_`result' = r(`dvar'_`result')
        }
}

end

global storeresults ""
foreach dvar in $all_dvars {
        foreach result in ctrl_est ols1_est ols2_est aipw_est aipw_tst {
                global storeresults "$storeresults `dvar'_`result' = r(`dvar'_`result')"
        }
}

timer on 1

//simulate $storeresults, reps(2000) saving("../../store/boot_dist.dta", replace every
    (10)): bootx

parallel initialize
parallel sim, expr($storeresults) reps(2000) randtype(current) saving("../../store/
    boot_dist.dta", replace every(10)): bootx
parallel clean

timer off 1
timer list 1
timer clear 1
```

end bootstrap_dist.do

begin appended_results.R

```
loan_use_vars <- c("Business_Expenditures", "Non_Business_Exp", "New_Business_Ap15")
profits_vars <- c("Profit", "ln_Q50", "Capital")
default_vars <- c("Late_Days_364", "Late_Days_476",
                  "not_finished_aug19", "Outstanding_Loan_Amount_Default")
repayment_vars <- c("Fifty_Percent_Loan_Paid", "Made_First_11_Pay_On_Time",
```

```
                    "Made_First_Pay")
business_vars <- c("atleastone_bizshutdown_alt", "Max_Min", "Q68")
customers_vars <- c("Q35_", "Q37_", "Q11_Together_max")
all_dvars <- c(loan_use_vars, profits_vars, default_vars,
               repayment_vars,business_vars,customers_vars)

library(dplyr, warn.conflicts = FALSE)

obs_estimates <- readRDS("../../store/obs_estimates")
boot_dist <- readRDS("../../store/boot_dist")
rand_dist <- readRDS("../../store/rand_dist")

bootstrap_se <- data.frame(matrix(ncol = length(all_dvars), nrow = 3))
col_num = 0
for (dvar in all_dvars) {
  col_num <- col_num + 1
  boot_se <- t(data.frame(t(matrix(unlist(lapply(boot_dist,
             function(x) x[,c(dvar)])), nrow = 5))) %>% summarize_all(sd))
  bootstrap_se[,col_num] <- boot_se[2:4]
}
rm(boot_se, col_num)
rownames(bootstrap_se) <- c("ols1_bse","ols2_bse", "aipw_bse")
colnames(bootstrap_se) <- all_dvars

exact_single_pval <- data.frame(matrix(ncol = length(all_dvars), nrow = 4))
col_num = 0
for (dvar in all_dvars) {
  col_num <- col_num + 1
  exact_p <- t(data.frame(t(matrix(unlist(lapply(rand_dist,
             function(x) abs(x[,c(dvar)]) >= abs(obs_estimates[,c(dvar)]))),
             nrow = 5))) %>% summarize_all(mean))
  exact_single_pval[,col_num] <- exact_p[2:5]
}
rm(exact_p, col_num)
rownames(exact_single_pval) <- c("ols1_epv","ols2_epv",
                                 "aipw_epv", "saipw_epv")
colnames(exact_single_pval) <- all_dvars

#### stepdown p-values

source("../programs/stepdown_pval.R")

exact_stepdown_pval <- data.frame(matrix(ncol = length(all_dvars), nrow = 1))
rownames(exact_stepdown_pval) <- "saipw_spv"
colnames(exact_stepdown_pval) <- all_dvars

block_names <- c("loan_use_vars", "profits_vars", "default_vars",
                 "repayment_vars", "business_vars", "customers_vars")

for (i in 1:length(block_names)) {
  block_vars <- get(block_names[i])
  null_t_df <- data.frame(matrix(unlist(lapply(rand_dist,
               function(x) x[5,all_of(block_vars)])), ncol = length(block_vars),
               byrow = TRUE))
  colnames(null_t_df) <- block_vars
```

```
  obs_t_vec <- unlist(obs_estimates[5,all_of(block_vars)])
  exact_stepdown_pval[1,all_of(block_vars)] <-
    stepdown_pval(abs(null_t_df), abs(obs_t_vec))[[1]]
}

#####

estimates <- obs_estimates[1:4,]

appended_results <- rbind(estimates, bootstrap_se, exact_single_pval)
appended_results <- appended_results[c(1,2,5,8,3,6,9,4,7,10,11),]
appended_results <- rbind(appended_results, exact_stepdown_pval)

saveRDS(appended_results, file = "../../store/appended_results")

rm(list = ls(all.names = TRUE))
```

end appended_results.R

begin table_single_pvals.do

```
clear all
set more off
version 17.0

use "../../store/appended_results.dta", clear

do "../simulations/outcomes.do"

*****************************************
global note "\textit{Note}: For each outcome of interest, the above table reports the
    control mean and three ATE estimates (based on simple OLS, adjusted OLS, and
    augmented IPW methods) along with their bootstrap standard errors in parentheses (
    next to the estimates). Below each ATE estimate is its exact $ p$-value (reported
    within brackets) based on a randomization test using the raw estimate as the test
    statistic without studentization."

*****************************************

local l_Business_Expenditures "Total business spending"
local l_Non_Business_Exp "Total nonbusiness spending"
local l_New_Business_Ap15 "New business"
local l_Profit "Average weekly profits"
local l_ln_Q50 "Log of monthly household income"
local l_Capital "Capital"
local l_Late_Days_364 "Not repaid 8 weeks after due date"
local l_Late_Days_476 "Not repaid 24 weeks after due date"
local l_not_finished_aug19 "Not repaid 52 weeks after due date"
local l_Outstanding_Loan "Outstanding loan 52 weeks after due date"
local l_Fifty_Percent_Loan_Paid "Repaid at least 50\% of loan"
local l_Made_First_11 "Made first half of repayments on time"
local l_Made_First_Pay "Made first payment"
local l_atleastone "Business closure"
local l_Max_Min "Profit range length"
```

```
local l_Q68 "Reapid by selling items at discount"
local l_Q35_ "Customers buy on credit"
local l_Q37_ "Customers pre-order items"
local l_Q11_Together_max "Number of items for sale"

file open latex_table using "../../tables/table_single_pvals_stata.tex", write replace

local alignment_line "l|c@{\hskip 10pt}|c@{\hskip 10pt}|c@{\hskip 10pt}|c"

file write latex_table "\begin{table}[!ht]" _n
file write latex_table "\begin{center}" _n
file write latex_table "\caption{\textit{\textbf{Impact of grace period on business
    activity and repayment outcomes (using Stata)}}}" _n
file write latex_table "\label{table:table_single_pvals_stata}" _n
file write latex_table "\scriptsize \vspace{2mm}" _n
file write latex_table "\setstretch{1.5}" _n
file write latex_table "\begin{tabular}{`alignment_line'}" _n
file write latex_table "\hline\hline" _n

local header_line1 " & \textit{Control} & \textit{Simple OLS} & \textit{Adjusted OLS} &
    \textit{Augmented IPW} "
local header_line2 " \textit{Outcome} & \textit{mean} & \textit{estimate of ATE} & \
    textit{estimate of ATE} & \textit{estimate of ATE} "

foreach dvar in $all_dvars {
        local row = 1
        local `dvar'_ctrl = `dvar'[`row']
        foreach estim in ols1 ols2 aipw {
                foreach stat in est bse epv {
                        local row = `row' + 1
                        local `dvar'_`estim'`stat' = `dvar'[`row']
                }
                if ``dvar'_`estim'epv' <= 0.1 {
                        local `dvar'_`estim'epl "\mathbf {"
                        local `dvar'_`estim'epr "}"
                }
                else {
                        local `dvar'_`estim'epl ""
                        local `dvar'_`estim'epr ""
                }
        }

        local `dvar'_line1 "`l_`dvar'' & $ `: di %5.2f ``dvar'_ctrl' ' $"
        local `dvar'_line2 " & "
        foreach estim in ols1 ols2 aipw {
                local `dvar'_line1 "``dvar'_line1' & $ `: di %5.2f ``dvar'_`estim'est' '
                    "
                local `dvar'_line1 "``dvar'_line1' \;\; ( `: di %5.2f ``dvar'_`estim'bse
                    ' ' ) $ "
                local `dvar'_line2 "``dvar'_line2' & $ [ ``dvar'_`estim'epl' `: di %5.4f
                     ``dvar'_`estim'epv' ' ``dvar'_`estim'epr' ] $"
        }
}
```

```
file write latex_table "`header_line1' \\ [-1mm]" _n
file write latex_table "`header_line2' \\ \hline \hline" _n

local blocknum = 0
foreach block in loan_use profits default repayment business customers {
        local dvarnum = 0
        local blocknum = `blocknum' + 1
        foreach dvar in ${`block'_vars} {
                local dvarnum = `dvarnum' + 1
                file write latex_table " $ (`blocknum'.`dvarnum') $ ``dvar'_line1' \\
                    [-1mm] " _n
                file write latex_table " ``dvar'_line2' \\ \hline " _n
        }
        file write latex_table "\hline" _n
}

file write latex_table "\end{tabular}" _n
file write latex_table "\end{center} \vspace{-2mm}" _n
file write latex_table "\setstretch{1}\noindent \scriptsize" _n
file write latex_table "$note" _n
file write latex_table "\end{table}" _n

file close latex_table
```

end table_single_pvals.do

begin table_single_pvals.R

```
loan_use_vars <- c("Business_Expenditures", "Non_Business_Exp", "New_Business_Ap15")
profits_vars <- c("Profit", "ln_Q50", "Capital")
default_vars <- c("Late_Days_364", "Late_Days_476",
                "not_finished_aug19", "Outstanding_Loan_Amount_Default")
repayment_vars <- c("Fifty_Percent_Loan_Paid", "Made_First_11_Pay_On_Time",
                    "Made_First_Pay")
business_vars <- c("atleastone_bizshutdown_alt", "Max_Min", "Q68")
customers_vars <- c("Q35_", "Q37_", "Q11_Together_max")
all_dvars <- c(loan_use_vars, profits_vars, default_vars,
               repayment_vars,business_vars,customers_vars)

block_names <- c("loan_use_vars", "profits_vars", "default_vars",
                 "repayment_vars", "business_vars", "customers_vars")

library(dplyr, warn.conflicts = FALSE)

ar <- readRDS("../../store/appended_results")

l_Business_Expenditures <- "Total business spending"
l_Non_Business_Exp <- "Total nonbusiness spending"
l_New_Business_Ap15 <- "New business"
l_Profit <- "Average weekly profits"
l_ln_Q50 <- "Log of monthly household income"
l_Capital <- "Capital"
l_Late_Days_364 <- "Not repaid 8 weeks after due date"
l_Late_Days_476 <- "Not repaid 24 weeks after due date"
```

```
l_not_finished_aug19 <- "Not repaid 52 weeks after due date"
l_Outstanding_Loan_Amount_Default <- "Outstanding loan 52 weeks after due date"
l_Fifty_Percent_Loan_Paid <- "Repaid at least 50\\% of loan"
l_Made_First_11_Pay_On_Time <- "Made first half of repayments on time"
l_Made_First_Pay <- "Made first payment"
l_atleastone_bizshutdown_alt <- "Business closure"
l_Max_Min <- "Profit range length"
l_Q68 <- "Reapid by selling items at discount"
l_Q35_ <- "Customers buy on credit"
l_Q37_ <- "Customers pre-order items"
l_Q11_Together_max <- "Number of items for sale"


latex_lines <- c("\\begin{table}[!ht]", "\\begin{center}",
"\\caption{\\textit{\\textbf{Impact of grace period on business activity and repayment
    outcomes (using R)}}}",
"\\label{table:table_single_pvals_r}", "\\scriptsize \\vspace{2mm}",
"\\setstretch{1.5}",
"\\begin{tabular}{l|c@{\\hskip 10pt}|c@{\\hskip 10pt}|c@{\\hskip 10pt}|c}",
"\\hline\\hline",
" & \\textit{Control} & \\textit{Simple OLS} & \\textit{Adjusted OLS} & \\textit{
    Augmented IPW} \\\\ [-1mm]",
" \\textit{Outcome} & \\textit{mean} & \\textit{estimate of ATE} & \\textit{estimate of
    ATE} & \\textit{estimate of ATE} \\\\ \\hline \\hline")

fileTeX <-file("../../tables/table_single_pvals_r.tex")
writeLines(latex_lines, fileTeX)
close(fileTeX)

blocknum <- 0

for (i in 1:length(block_names)) {
  dvarnum <- 0
  blocknum <- blocknum + 1
  for (dvar in get(block_names[i])) {

    dvarnum <- dvarnum + 1

    dvar_line1 <- paste0(" $ (",blocknum,".",dvarnum,") $ ",
                         get(paste0("l_",dvar)),"  & $ ",
                         format(ar["ctrl_est",dvar], digits=1, nsmall=2)," $ ")
    dvar_line2 <- " & "

    for (estim in c("ols1","ols2","aipw")) {

      epl <- ifelse(ar[paste0(estim,"_epv"),dvar] <= 0.1," \\mathbf { ","")
      epr <- ifelse(ar[paste0(estim,"_epv"),dvar] <= 0.1," } ","")

      dvar_line1 <- paste0(dvar_line1," & $ ",
        format(ar[paste0(estim,"_est"),dvar], digits=1, nsmall=2)," \\;\\; ( ",
        format(ar[paste0(estim,"_bse"),dvar], digits=1, nsmall=2)," ) $ ")
      dvar_line2 <- paste0(dvar_line2," & $ [",epl,
        format(ar[paste0(estim,"_epv"),dvar], digits=0, nsmall=4),epr," ] $")
    }
```

```
    latex_lines <- c(paste0(dvar_line1,"\\\\ [-1mm]"), paste0(dvar_line2," \\\\ \\hline
        "))
    write(latex_lines,file="../../tables/table_single_pvals_r.tex",append=TRUE)
  }
  write("\\hline",file="../../tables/table_single_pvals_r.tex",append=TRUE)
}

latex_lines <- c("\\end{tabular}","\\end{center} \\vspace{-2mm}",
                 "\\setstretch{1}\\noindent \\scriptsize",
                 "\\textit{Note}: For each outcome of interest, the above table reports
                     the control mean and three ATE estimates (based on simple OLS,
                     adjusted OLS, and augmented IPW methods) along with their bootstrap
                      standard errors in parentheses (next to the estimates). Below each
                      ATE estimate is its exact $ p$-value (reported within brackets)
                     based on a randomization test using the raw estimate as the test
                     statistic without studentization.","\\end{table}")
write(latex_lines,file="../../tables/table_single_pvals_r.tex",append=TRUE)

rm(list = ls(all.names = TRUE))
```

end table_single_pvals.R