Center for Enterprise Dissemination

Thin Client User Guide

Integrated Research Environment (IRE) Edition

May 2025

Blank page

Table of Contents

1 The First Day

On the first day, the RDC Administrator will help you log on to the server using a thin client and show you some basic functions. After this you should work through the first section of this guide on your own.

1.1 A Note on User ID and Passwords

Your User ID is on your Account and Data Request Form. You should memorize your User ID.

The system requires a strong password that:

- is at least 15 characters long
- contains **at least** three of the following four categories:
  - upper case letters
  - lower case letters
  - numbers
  - symbols
- **does not** contain portions of your account name, your email address, your first name, or your last name.

Note that **your password expires if either it is older than 89 days** *or* **if you do not log on for 25 days**. If you will not visit the FSRDC for more than three weeks, maintain your password at https://pss.tco.census.gov – select Update my Security Questions. This will prompt you for your userid and password, thus creating a login for your account; or prompt you for a password change if expiration is reached. Complete password change or security question authentication, then simply close the window.

1.2 Intent and Philosophy of the Thin Client User Guide

This guide is designed to familiarize you with the operating environment. The tasks set forth in Section 1 are intended (a) to make your use of the system easier and (b) to be done once and completed in one session.

After you have worked through this first section, we recommend that you read the rest of the guide *before* you are in a pickle, frantically looking for answers. At a minimum it is important to read sections:

- 1 The First Day,
- 2 Detailed Information on the Basics,
- 3.1 PBS Pro, and
- any additional section for applications you intend to use.

This guide is full of helpful tips and information that will make your everyday use of the system easier and if you are in a jam, it is helpful to know that you have seen the answer before and, therefore, have an idea where to find it.

NOTE: in sections that follow anything that is typed at a prompt or in a script will **`look like this so that spaces and other elements are clear`**. Also, where you see <something>, you provide information where “something” is *without* the angle characters, <>.

The *Thin Client User Guide* is a working document: it is constantly under revision. If something is unclear, something important is omitted, or you simply can contribute something useful, please let us know about it. Send an e-mail to your administrator with “TCUG suggestion - <brief description>” as the subject.

Brief Tour of the Desktop and Customization

This section will give you the basics to navigate the IRE computing environment once logged in to the thin client and tell you how to make some helpful changes to the default settings on the system. Since your administrator has just helped you log in for the first time, let's start with the desktop.

1.3 Desktop

In RedHat Linux the taskbar and dock are called panels. At the top of the desktop is Panel 1 and at the bottom is Panel 2.

Panel 1

Reading left to right on the top panel you will see

- Applications button: click to list and select from applications available on the *login* node
- Section to select open windows
- Date and time (in *eastern* time zone)
- Your name

Note that intensive processes are to be run on a compute node, not the login node. Examples of "intensive processes" are compressing files using gzip, word processing (text editor is okay), and running analysis. All of these can be run after qsub to a compute node, gzip and analysis can also be run in batch.

Panel 2

Reading left to right on the bottom panel you will see

- Window minimizer
- Open Terminal Emulator
- Open File Manager
- Open Firefox
- Open Application Finder
- Open Home directory (Note that no project related files may be saved in your home directory. The home directory is for configuration files (e.g., autoexec.sas) only.)

1.3.1 Quick Tips and Make Useful Customizations

- To lock your workstation: Window+L
- Turn off screen saver: Panel 1 → Applications → Settings → Screensaver. Start on the Lock Screen tab, go to the bottom and disable each item as you move up in the window until you get to the top. Next go to the screensaver tab and do the same thing.
- Alt+Tab does not cycle through the windows in NoMachine; a suggested alternate key combination is Alt+` as the ` (accent) is close to the Tab key on the keyboard. To set up walk-through windows within the NoMachine session: Panel 1 → Applications → Settings → Window Manager → Keyboard → Cycle windows → Edit → Alt`
- Spend time with data documentation and disclosure avoidance materials **before** you make *any* data extracts.
- When reporting trouble, be sure to provide the error message as well as which login node you are on and PBS Pro job # if applicable.
- Do not use spaces in directory or file names.

1.3.2 File Manager: Thunar

Thunar can be opened many ways. To open it directly to your project space, at Terminal prompt type: **`thunar /projects &`**

Depending on how you open Thunar, it may open to the file system (root), your project space or in your home directory (which starts with /home). Use the navigation bar to get your bearings. For important information on your home directory see section 2.3.1 on page 5.

1.4 Help Resources

1.4.1 Thin Client User Guide

This document (TCUG) is your primary resource for using the computing system in the FSRDCs.

1.4.2 Documentation Subdirectory

There is documentation for your project-approved data set(s) in the directory for that data set in the subdirectory "Documentation." To locate your data sets in the data warehouse, open pdata.sas in a text editor. This file is located at /projects/lib/sas.

1.4.3 Intranet Site: rdcdoc

The RDC intranet has user guides for SAS, R, Stata and PBS Pro, as well as a digital copy of TCUG, and much more.

Open Firefox by clicking the globe in Panel 2, navigate to **http://rdcdoc.cods.census.gov** and set this site as your homepage. To set your homepage either highlight the complete URL and drag it to the house icon to the left or from Open application menu (3 horizontal lines on the right) → Settings → Home → Use Current Page

1.4.4 Terminal Command Information: "man"

For information on Terminal commands type **`man <command>`** [ENTER] at the prompt. (To open Terminal click on the Terminal Emulator icon in Panel 2.) This will bring up the **man**ual page for the specified command. You can even **`man man`** to learn about how to customize the information you receive using the **man** command. To page forward press [PAGEDOWN]; to page backward press [PAGEUP]. To exit the manual, type **`q`**.

1.5 Things that you should ALWAYS do

Following the recommendations in this section will greatly reduce the number of difficulties you and others will experience when using the RDC computer system.

1.5.1 Secure Your Session

When you leave your workstation, you must either lock your session or log out.

If you will be away from your workstation for less than an hour, lock your session by pressing the key combination Window+L.

If you will be away from your workstation for more than an hour, you must log out completely from your workstation. You input your credentials 3 times to log in, so you also log out 3 times:

- in Panel 1 either click your name or Applications → Log Out → Log Out
- in Edge click gear on the right → Log off
- ctrl+alt+delete → Log Off

1.5.2 Run Debugged Programs in Batch Using PBS Pro

The most efficient use of resources is to run debugged programs as batch programs. These are submitted via PBS Pro, our job management application. For full directions on how to use PBS Pro and run batch programs, see section 3.

Note that you cannot run a process interactively overnight. Interactive sessions have a 12-hour time limit.

1.5.3 Follow Good Citizen Practices

All researchers share disk space, RAM, and processing resources on the RDC servers. It is very important that everyone use only what they **need**. Good citizen practices are highlighted throughout this guide, but here, for easy reference, are a key few:

- Do program development and debugging on a subset of data
- Interactive sessions are appropriate only for debugging programs or viewing results
- Once debugged, run programs in batch
- Don't use the whole data set if you don't need to
- Close Stata and Matlab if you are not actively using them
- Don't open more than one interactive session of Stata
- Save only the files you need
- Compress archival data sets using gzip on a compute node
- delete files that are no longer needed

Not following "Good Citizen Practices" can result in extremely slow processing times, system problems, and poor popularity among your peers. We reserve the right to kill processes that are creating problems such as system instability or problematic resource constraints. If you have a resource intensive program, especially one that is memory intensive, please notify your Administrator in advance with the program name(s) and when you plan to run it (them) so that your Administrator can protect your process(es) from being killed and ensure smooth operation of the computing system.

2 Detailed Information on the Basics

2.1 Logging In

You will log in three times to get to your project space. The first login is to the client, the second is to VDI and the third is to a specific project.

At your workstation press ctrl+alt+delete and enter your JBID and password (“credentials”). Microsoft Edge will open (sometimes there is a wait), enter your credentials again. You are now logged in to VDI. Once logged in to VDI in the RDC, you have two icons you can select: Enterprise Password Self-Service and NoMachine. To log in to your project, select the NoMachine icon. At the next screen, double-click your project. Your username should be pre-populated. In the password field, type your password.

If the thin client has recently been rebooted, then a window will pop up that asks about a "RSA key fingerprint," click "yes."

When you log in, you will be on one of the login nodes. Analytical processes cannot be run here. PBS Pro (job scheduling software) will route your job to one of the other nodes of the cluster for processing.

2.1.1 Changing Your Password

Only change your password using Enterprise Password Self-Service outside of the RDC, website: https://pss.tco.census.gov.

2.2 Some Terminology

There are two methods to tell the computer what you want it to do: CLI (Command Line Interface) and GUI (Graphical User Interface). Most tasks can be done by either method so use whichever you prefer; however, sometimes you will need one or the other to do certain things.

Terminal: is a XFCE application that manages your terminal sessions; here it is effectively synonymous with terminal, shell, command prompt, and prompt. You will use Terminal when you submit CLI instructions to the computer. This is necessary to launch programs like SAS and Stata, and to issue jobs in batch.

Thunar is the file management application for XFCE, and as such is synonymous with file manager. You will use Thunar when you want to access directories and files by GUI and to submit certain instructions to the computer.

XFCE is a graphical desktop environment for Linux.

2.3 Directory Structure

Researchers have access to the following directories (a) their home directory, (b) their project directory, (c) the data directories for their approved data sets (the “data warehouse”), and (d) the temporary directories. GUI access to all directories is available via Thunar.

2.3.1 Home Directory

Your home directory is located at **/home/<a-z>/<User ID>**.

In general, you should limit the things that you save to your home directory to things that "must" go there because that is where a program expects to find them (i.e., .bash_profile, .bashrc, autoexec.sas, sasv9.cfg, and other such files). Save **project files** in the "project directory." **Never** save project files in the home directory. All users' home directories are on the same file system. If that file system becomes full, then no one can log in to the server.

2.3.2 Project Directory

To access the project directories:

CLI: With Terminal type **`cd /projects/`** [ENTER]

GUI: With Thunar, find on the right side of the window the tab with a red folder on it, click this tab. This will bring you to the root directory. There will now be a list of subdirectories, one of which will be projects.

By default, each project directory will have several directories. The intended purpose of each of these directories are listed below. Users can create their own subdirectories inside these pre-created subdirectories. Save all project files (i.e., data sets, programs, etc.) in your project directory.

- bin – not for researcher use – location of scripts and programs for project-specific maintenance
- **data** – location for data extracts and input/output files or information amongst project users
- **disclosure** – location for placing YYYYMMDD directory(ies) for disclosure avoidance review requests
- etc – not for researcher use – storing system-level configurations for per-project usage
- **lib** – lib/sas is the location of file containing path names to all project approved Census provided data (pdata.sas)
- logs – location to hold logs files generated from batch jobs (and others, if desired)
- **programs** – location for scripts and programs for the project
- (dstaff)transfer – location for data sets placed into the project by the data staff
- users – location of per-user directories

You may create additional directories, but do not delete or rename any of the default directories.

Currently, there are no quotas on the project directories, but keep in mind that you are sharing server space with all RDC users and Census employees. **Poor data management can bring the system down.** It's important to clean out data files you will not need again and to compress large files that you need to keep, but will not use soon, using gzip.

2.3.3 Data Directories – the Warehouse

All the source data is located in the various classification directories (/data**/economic**, /data**/decennial**, etc.). Under the appropriate classification directory, you will find a subdirectory for each project approved dataset labeled with an abbreviation for that survey. For a complete listing of Census-provided data (and their locations) view /projects/lib/sas/pdata.sas in a text editor. Within a dataset subdirectory, you will see all the files available for the project for that dataset, possibly by year or state. You will make your own extracts from these data. Any extracts you create must be saved in your project directory. Most data directories also contain subdirectories with documentation for those data, either for the whole dataset in /data/<type>/<dataset>/doc or within specific year subdirectories for that dataset.

2.4 Managing Your Files

2.4.1 Moving, Copying, etc.

Most file management can be done with Thunar. You can drag and drop files to move or copy them to another directory. If it is easier, then you can open **two** Thunar windows. You can also find the commands to copy, move, and delete files and create new directories in the menus.

If you prefer to use the Terminal, the basic file management commands are as follows. Note that you will need the prompt pointing at the appropriate directory to use FileName in the below commands, otherwise you will need to type the complete path and file name.

man	**man** is is short for **man**ual and shows the help information for a given command. Syntax: **man <command>** [ENTER].
cd	changes the directory you are in. Syntax: **cd <DirectoryPath>** [ENTER]. Simply typing **cd** [ENTER] will return you to your home directory and typing **cd ..** [ENTER] will move you up one directory.
evince	opens pdf files. Syntax: **evince <FileName>.pdf &** [ENTER].
cp	copies a file. Syntax: **cp <FileName> <NewPath/FileName>** [ENTER].
ls	lists files in a directory. For more information on **ls**, see section 2.4.3.1.
mkdir	Creates a new directory. Syntax: **mkdir <NewDirectoryName>** [ENTER].
mv	moves files. Syntax: **mv <FileName> <PathOfNewDirectory>** [ENTER]. You can use a wildcard to move more than one file at a time (e.g., use *.* to move all files in the current directory). For more information on wildcards, see section 2.4.4.3 on page 9.
pwd	**p**rints the full pathname of the **w**orking **d**irectory (the directory you are in).
rm	deletes specified file(s). Syntax: **rm <FileName>** [ENTER]. Wildcards are permitted as in **mv** above.
rmdir	deletes specified directory. Syntax: **rmdir <DirectoryName>** [ENTER].

Other useful commands are explained and itemized in relevant sections of this guide.

2.4.2 Compressing Files

gzip is a compression command. It allows compression of individual files or the contents of entire directories easily while maintaining the properties of the file (i.e., ownership, creation date, etc.). A gzipped file will be “renamed” to include a **.gz** file extension. For example, file mystuff.log would become mystuff.log.gz once compressed using **gzip**.

Gzip should be run on a compute node, not a login node. If you have only a few small files to run, you can use gzip interactively. (i.e., type **qsub -IX** first, then use **gzip <FileName>** [ENTER]). Large files, however, will take a long time to compress. Use a variation of this batch script if you have large or many files to compress. We’ll call the script compress.bash.

```
#!/bin/bash

clear

# change working directory to main project space (1)
cd /projects

# create log file, if you like, to be saved in directory (1)
# replace xxxx below to whatever you like to keep track of the
# activity
exec 1>gzip_log_xxxx.txt 2>&1

echo "Starting gzip script!"

# START REPEAT
# replace DIR with directory path that contains files that should be
# gzipped
cd /DIR
# You can then either move through the directory and its
# subdirectories to gzip all files:
gzip -rf *
# OR gzip individual files in DIR only:
gzip filename1
gzip filename2
# etc.
# END REPEAT

# repeat lines between START REPEAT and END REPEAT for each directory
# containing files needing gzip

echo "gzip script is done!"
```

With the prompt pointing at the directory where compress.bash is saved, type:
`qsub compress.bash &` [ENTER]

gunzip is just as easy to use. Simply replace **gzip** with **gunzip** in the above instructions to undo the compression.

2.4.3 Searching for Files and Processes

2.4.3.1 Finding Files with "ls"

ls is a Linux command that lists the contents of a directory. Two common uses are (a) discover if the file the user is looking for is in a certain directory, and (b) to get an overall feel for the contents of a directory. For example, if after typing **`ls`** at the prompt, the user sees a list of files ending with ".gif" then the user might surmise that this directory is used to store image files.

ls Syntax **`ls <options> <DirectoryPath> or <SearchTerm>`**

When **ls** is specified alone, then no options are assumed, and files are listed for the current directory. Typing **`ls`** at the prompt will simply list the contents of the current directory. Typing a

different directory path will list the contents of the specified directory. Example: **`ls /usr/bin/`** lists the contents of the directory /usr/bin.

You can also specify that only certain files are listed. Example: **`ls log`** lists a file called "log" if it exists in the current directory.

Common ls Switches
The options that accompany a command are called switches in Linux. Some of the common switches for ls are

`-a` lists **a**ll the files in a directory, including hidden files (those starting with ".").
`-l` gives the list in **l**ong format, which includes information about the last date the file was modified and who has permissions to access the file.
`-R` **r**ecursively lists all subdirectories in the directory.

You can also combine switches. To specify both the –a and the –l switches type **`ls -al /usr/bin/happ*`** [ENTER] at the prompt. This displays the long form of all files in the directory /usr/bin whose first four letters are happ.

Note that you cannot recursively search for a term using ls; this is why you would pipe to grep…

2.4.3.2 Piping to grep

Piping your output to grep is a clever use of commands in Linux. Here is a summary.

Piping
Piping is adding a "|" after any command. This takes the output from that command and uses it as an input for the command that follows instead of immediately displaying the output. The key combination for | is [SHIFT]+[\].

grep
grep is a command that matches information specified by the user. This is useful when you have a large amount of text that you need to search.

Piping to grep
This is how you can use these two together: Let's say you have many files in your directory and you're looking for one that you created in July. You could type **`ls -l`** [ENTER], but since you have a lot of files, that would give you several pages to look through. A more effective way to do it would be to pipe the output to grep with "Jul" as the search term. You would type **`ls -l | grep Jul`**. This way only the lines of output from **ls –l** that have Jul somewhere in the line will be displayed.

Note that to search for a file name that contains a space (we strongly discourage the use of spaces in either a file or directory name), put quotes around the search term, otherwise the space will be interpreted as indicating the start of a new input. For example, to search for a file called my file.txt, you would type **`ls "my file.txt"`**. The same applies to the grep command as well.

2.4.3.3 Wildcards

You can also use wildcards to make searches more versatile and useful.

* The star will search for any number of characters in addition to your search term.

Examples:
ls *.log will list all files that end in .log regardless of how long the file name is.
ls extract.* will list all files that start with extract regardless of the file extension. So you might see extract.log, extract.lst, extract.sas7bdat, extract.dta.gz, etc.

? The question mark will search for as many random characters as there are ?s.

Examples:
ls ????.log will list all files that end in log and have a four character file name.
ls extract.??? will list all files that start with extract and have a three character file extension. As above, extract.log and extract.lst would be listed, but extract.sas7bdat and extract.dta.gz would not.

[ ] The square brackets can be used to contain choices for a single character and lists. Important: square brackets work as a wildcard only when they are in the first position of the search term!

Example:
If you have a number of files in a directory and you would only like to list those that either start with the word “bat” or “sat” you could type **ls [bs]at***. Notice the [ ] contain the two options, in this case either “b” or “s." This search will yield results like battery.dta and satire.sas7bdat.

You can also use square brackets to contain an ordered list, for example if you have a directory full of census files called XXXXCensus where XXXX stands for the year (1997, 1892, etc.), you can list only those between 1950 and 1980 by using a list in the square brackets,
ls [1950-1981]Census*.

2.4.3.4 Finding Files with “find”

The **find** command is a powerful searching tool that helps you find a file *and* its path.
find has a difficult syntax which is beyond the scope of this guide. Rather than trying to explain it completely, an example will be given along with some explanation. This should be enough to allow you to use it easily. Here's the example.

```
find –type f –name "census*" –print [ENTER]
```

This command asks the system to find *files* called census and census+something. Wildcards work within the quotes, but the remaining elements in the quotes are case sensitive. The search is started in your current directory, continuing down through the directory tree. You can match dates, file sizes, etc., and you can search for directories instead of files; however, as mentioned, the syntax is rather involved. Using the above example but replacing census* with your search term should be enough in most cases.

3 Software

We offer many applications for statistical analysis. It is important to point out, though, that we offer support for two: SAS and Stata. With that said, we try to provide as much information about other applications as possible. If you have familiarity with specific applications or can offer guidance that might benefit other users, then your contribution would be much appreciated.

What is available?

- Anaconda/Python
- Grass
- Mathematica
- Matlab
- PBS Pro
- R and Rstudio
- SAS
- Stata and Stata-MP
- Stat/Transfer
- Tomlab, Knittro, MADD

For version number, refer to the application itself.

3.1 PBS Pro

About the RDC Server Cluster

The IRE server cluster is a group of servers that communicate with one another but operate independently. There are two types of servers (or "nodes") – login nodes and compute nodes.

Nodes nhpc-login1 and nhpc-login2 serve as the "login nodes." The login nodes are the only nodes that you will have direct login access to. You will access the compute nodes via PBS Professional (PBS Pro). Only the compute nodes are able to run statistical analysis applications. There are 44 of these: 10 with 88 CPU and 1500 GB and 34 with 56 CPU and 767 GB. Remember all researchers share disk space, RAM, and processing resources on the RDC servers. It is very important that everyone use only what they **need**.

When setting up NoMachine sessions to login use nhpc-login.rm.census.gov as the server name. This will enable the system to load balance the sessions between the login. This scheme also allows more login nodes to be added as needed.

The login nodes provide a place for you to login and set up compute jobs (e.g., data cleaning, extract production and analysis programs) that you will submit to the compute nodes. You can use the login nodes to write and edit code using a text editor, check the status of jobs you've submitted, prepare files for disclosure review, create tables, or other "small" tasks (things that are not computationally intensive). All of your "compute jobs" should be submitted to the compute nodes using PBSPro.

What is PBS Professional (PBS Pro)?

PBS Pro is a job scheduler that works across all nodes of the IRE server cluster. It evaluates the available resources on each node and assigns your job(s) accordingly. Use of PBS Pro improves efficiency, stability, and maximizes the resources currently available on the servers.

PBS Pro runs ALL statistical analysis jobs on the cluster – batch jobs and interactive jobs. An interactive job is one where use of the application is tied to your login – you are actively working within an application to develop code. A batch job is one that continues to run even once you log out. These are typically executed when debugging is complete and the job is expected to run for more than a few hours.

PBS Pro matches the resources requested by each job to the resources available on the cluster nodes, and assigns the job to a node that can accommodate the request. If there is no available node that can satisfy the request, the job is held in the queue until the resources are available to run the job. **Note – you should use batch mode except for when debugging a program.**

3.1.1 Batch Job Submission

Users run batch jobs by expressing those jobs as "job scripts" and submitting those scripts to a queue. We have pre-written scripts for some applications for ease of use. These are called "wrappers." You can submit batch jobs by using one of the wrapper scripts for SAS, Stata and R. From the appropriate directory or using the complete pathname:

- **`qsas program_name.sas &`** [ENTER]
- **`qstata program_name.do &`** [ENTER]
- **`qR program_name &`** [ENTER]

Once you submit a job to PBS Pro you will see output like **`12345.hpc-pbs`**
The five-digit number is the PBS Pro job identifier which is used for monitoring the process.

These wrappers may have additional parameters you can set to request more resources if your process requires them. By default, all jobs submitted to PBS Pro are allocated 1 CPU and 5 GB of memory.

To see what can be changed while still using the wrapper script, type the wrapper name on the command line by itself and press enter.

3.1.1.1 Writing Batch Scripts

You may need to write your own batch script to submit jobs to PBS Pro if you use an application that does not have a wrapper or if the wrapper cannot accommodate the parameter changes you need. Writing your own script gives you more control and allows you to use one or more "PBS directives." (See *PBS Pro User Guide* on rdcdoc.)

All basic PBS Pro batch scripts begin with the same three lines. The second line, even though it looks like it is commented out, is where you specify the PBS Pro directives for the program. In the example below we are requesting 1 CPU and 8 GB of memory and using Stata for the analysis.

The script name is example.bash and it is saved in /projects/programs:

	Script text	Description
(1)	`#!/bin/bash`	(1) Specifies bash script, do not change this.
(2)	`#PBS -l ncpus=1,mem=8gb`	(2) Tells PBS Pro to expect directives and what they are. For clarity, the items after #PBS is dash "L", and ncpus how many cpu you need.
(3)	`cd /projects/programs`	(3) Change directory to full path of program file.
(4)	`stata-se -b do program_name.do`	(4) Command to launch job.

You will submit the script to PBS Pro at the command line. With the prompt pointing at the directory where example.bash is saved, type: **`qsub example.bash &`** [ENTER].

Obviously, you will change the last line of the script to use a different application. Refer to the appropriate section for information on the application you would like to use.

3.1.2 Interactive Job Submission

PBS Pro supports "interactive batch jobs." While we understand that you will need to run interactive jobs, we ask that you use batch mode whenever possible, particularly for long running jobs that you are going to leave unattended. When a batch job completes, it terminates and the resources that were allocated to it are released back into the pool where they can be used by others. An interactive batch job completes when the user types "exit" to terminate it.

To launch an interactive batch session, enter type **`qsub -IX`** [ENTER].

This will place you in a shell session on a compute node selected by PBS Pro with the default resources 1 CPU and 5 GB of memory. From here, you can modify and run your programs or run GUI versions of your desired applications.

Type "exit" to exit the interactive session when you are finished. **Do not leave the interactive session running when you close NoMachine! Also note that interactive jobs will be automatically terminated after the 12-hour time limit.**

If you need more resources than the default, you can include a resource specification. For example, **`qsub -IX -l ncpus=1,mem=7gb`** where after **`-IX`** is "dash lower case L". See the man page for pbs_resources (**`man pbs_resources`**) for more information.

3.1.3 Monitoring Jobs

When you submit a PBS Pro job, the system will return a number to identify the process. For example, **23313**.hpc-app1. You will use PBS Pro and this number to monitor your job.

To view the job queue and check the status of your job, type **`qstat`** [ENTER] at the prompt. The output will look like

```
Job id          Name    User       Time Use   S  Queue
------------    ------  ---------  ---------  -  -----
23313.hpc-pbs  myjob   fghij111   01:13:40   R  workq
24158.hpc-pbs  STDIN   abcde999   00:00:17   R  interq
```

Column "Name" is the name of the job, usually truncated. Here the first on the list is a batch job where the script is named myjob and the second on the list is an interactive session. All interactive jobs are named STDIN by default. Column S is job status. Entries you could see in this column: **Q** means the job is waiting for resources, **R** means the job is running, **S** means the job is suspended, **H** means the job is held, and **E** means exiting.

There are many options to use with qstat to get more detailed information if desired: **`qstat -a`** shows **a**ll jobs, **`qstat -n`** shows execution **n**ode, **`qstat -f`** shows **f**ull details, and really nice to use sometimes is **`qstat –1fan.`**

To check the status of only *your* running jobs**: `qstat -n1 -u <jbid>`** [ENTER].

3.1.4 Useful PBS Pro Commands and Examples

Commands

`pbsnodes -a` Gives the status of the execution nodes, shows what jobs are running on them, what resources are available, what resources are allocated, etc.

`qstat -f <job_id> | grep resources_used.vmem` Check the RAM usage (in kb) of a *running job*

`qstat -f <job_id> | grep resources_used.cpupercent` Check the CPU usage of a *running job* (max usage per CPU is 100, i.e., max of 4 CPU is 400)

`qstat -xf <job_id> | grep resources_used.vmem` To check RAM and CPU for a *completed or killed job*, replace the switch with **-xf**

`qdel < job_id >` Kills a running job or deletes queued job. You can only delete your own jobs.

Tips

Directory paths: When you submit a PBS Pro job, it will by default “begin” in /projects. It is often convenient to change to the directory that you are working in so that you do not have to specify complete pathnames.

Stdout & Stderr files: When you submit a PBS Pro job, it will create an output file, jobnumber.hpc-app1.rm.census.gov.OU and possibly an error file, jobnumber.hpc-app1.rm.census.gov.ER, in the /projects/logs directory. These can be useful for troubleshooting problems; however, many applications will write the job output to a specific file rather than to the standard output when run in batch. In such cases, the *.OU and *.ER file corresponding to the job may be empty.

3.2 SAS

All data sets provided to you by Census are in SAS format. Note that if you don’t use SAS for your analysis, SAS is best for data steps. This section briefly summarizes a few points about using SAS using an example of exporting a dataset; extensive documentation about SAS is available on rdcdoc (see section 1.4.3 on page 3).

3.2.1 Getting Started

To launch a basic SAS batch job, at the terminal prompt type
`qsas program_name.sas` [ENTER]

To launch a SAS batch job adjusting memory allotment, at the terminal prompt type
`qsas --sasprog=program_name.sas -memsize=14000` [ENTER]
(14000 equals 14 GB)

Note that qsas will not execute if there is a space in either the path or program name.

To start an interactive SAS session, at the prompt of a terminal window, type **`qsub -IX`** [ENTER] to get to a research node, then type **`sas &`** [ENTER].

3.2.2 Libraries and Members

Libraries

SAS has its own file system that consists of “Libraries” and “Members." Libraries are user-defined "pointers" that instruct SAS where to look when you refer to a specific library; you want

SAS to read data from a certain directory and save the files it creates to a directory. A **libname** statement in SAS specifies the directory in which SAS should look. This is established by typing a statement in this format. **libname <libref> '<PathNameOfDirectory>';**

For example, let's say you have a directory, /projects/data/SasFiles, in which you want to save your SAS data and programs. You need to make a new library in SAS. You call it whatever you'd like (within the rules of library names), for example library1, and tell SAS that this library references the directory /projects/data/SasFiles. To assign the library, type this in the program editor.

libname library1 '/projects/data/SasFiles';

So, during this SAS session, whenever you tell SAS to read from or save to library1 it actually reads from or saves to /projects/data/SasFiles.

Note that a SAS libname file is automatically created for the Census provided data for each project in a file pdata.sas (see below for information on how to use this file), librefs still need to be made for your extracts (see 3.2.3 page 15 for information about this and autoexec.sas).

Members

Within Libraries are Members; members are just the files in a Library. When you want to reference a file in SAS you need to reference it by **<LibraryName>.<MemberName>**. So if you had a SAS data set called "output1" saved in your "/projects/data/SasFiles" directory, you would reference it in SAS as **library1.output1**.

pdata Library

A SAS libname file is created for each project. The file resides in **/projects/lib/sas/pdata.sas**. If you include this file in your SAS program, all data files accessible to the project will be available under the libname 'pdata'.

To include the file:

```
%include '/projects/lib/sas/pdata.sas';
```

Example of libname use:

```
proc contents data=pdata.asm1973;
run;
```

3.2.3 autoexec.sas

libname and **options** statements are often included in a file called autoexec.sas. This is a user created program file residing in the user's home directory. It is automatically run when SAS is invoked. Like all SAS programs, autoexec.sas can be created using any text editor.

Important: Stating default options and librefs in autoexec.sas instead of individual programs eliminates the need to change programs when data are moved. It is recommended that ONLY **options** and **libname**s be set in the autoexec file even though any legal SAS command can be included in the autoexec file; do not run **data** and/or **proc** steps from this file. Some important options are shown in this command line.

```
options compress=yes mlogic mprint ls = 80 ps = 60;
```

The second and third options deal with macro programming. **mlogic** outputs the value of each macro in the log window as it is encountered. **mprint** prints out the program code as reflected

by the value of the macro in the log statement. These two options are very useful in debugging macro programs; they are not turned on by default.

Macros are probably the most efficient way of writing SAS code that repeats itself. A well-written macro program a couple of screens long can execute hundreds of **data** steps and **proc** statements. As such, we encourage the use of macros.

Caution: turning on **mprint** for a macro program that generates large amounts of program code could fill up the log window during an interactive SAS session. This will cause SAS to stop until you manually confirm that you would like to clear out the log window. For a batch job, this will not be a problem.

3.2.4 SAS Help

Reference materials are available on rdcdoc (see section 1.4.3 on page 3).

Outside of the RDC, you can access SAS online documentation. A couple good links are http://support.sas.com/documentation/onlinedoc/base/index.html and http://support.sas.com/documentation/cdl/en/allprodsproc/61917/HTML/default/a003135046.htm

3.2.5 SAS Memory Error

If you are using really big data sets in your analyses, you will sometimes need to change the memory settings in SAS. Add these lines to the bottom of your .sas program (see 3.1.1.1).

```
OPTIONS FULLSTIMER;
PROC OPTIONS GROUP=MEMORY;
RUN;
PROC OPTIONS;
RUN;
```

This will print information about resources used for the program's processes in the log file. Then line (4) of the batch script should read: **`sas <file_name> –MEMSIZE 2G –REALMEMSIZE 0 –SUMSIZE 512M –SORTSIZE 512M –SYNCHIO –NOTERMINAL –NOTHREADS –VERBOSE &`**

3.2.6 A SAS Example

Below is a very short SAS program that uses some basic SAS commands. Each line has a comment that starts with /* and ends with */ that explains what the code is doing. SAS commands end with a semicolon.

```
%include '/projects/lib/sas/pdata.sas';
      /* This is not necessary if this statement is already in your autoexec.sas file, but it
      establishes pdata libref. (This is the source library of the project datasets.) */
libname to '/projects/data/SasFiles';
      /* This is not necessary if this statement is already in your autoexec.sas file, but it
      establishes a libref named to. (This is the destination library of the data set in this
      example) */
data to.data1 (keep = var3 var17 var53 var101 var3_2);
      /* Either creates a new member called data1 in the library to or replaces an existing
      member by the same name. The keep statement drops any variable not specified after
      the equal sign. That is, only the variables var3, var17, var53, var101, and var3_2 will be
      present in the new data set so be sure to list any variables that you create in the keep
      statement, too. */
   set pdata.data1992b;
```

```
      /* Takes the data that is in member data1992b in the library pdata and copies it into
         member data1 in library to. */
   if var101 = 2046;
      /* Drops all observations where the variable var101 is not equal to 2046. */
   var3_2 = var3 * 1000;
      /* Generates a new variable called var3_2; for every observation in the data set, it takes
         the value in var3, multiplies it by 1000, and stores it in var3_2. Note that if a variable
         called var3_2 already exists, then this statement would overwrite its contents. */
   if var17 = 0 then var17 = 496;
      /* Searches through every observation and if the value of var17 equals zero, then the
         value of var17 is changed to 496. */
run;  /* Executes the program; if you submit your statements without a run command, then
         nothing will happen. */
```

3.2.7 Exporting SAS Data Sets

Since all the data provided to you by Census are SAS data sets, you will need to convert the data to a different format if you need or want to use a different program. Note that Stata can natively use a SAS dataset (see section 3.3.3 on page 19), other applications will need a csv file.

3.2.7.1 Creating a SAS Data Extract

For either of the two exporting instructions that follow, you will first need to create a SAS data set that selects the variables that you want in the final data set. (Prior to creating the extract, you can determine which variables you want by doing a proc contents on the original SAS data set. Let's say that you want to create this extract in your project's data directory, /rdcprojects/br/br00999/data, and name it data2export.

```
%include '/projects/lib/sas/pdata.sas';
      /* This is not necessary if this statement is already in your autoexec.sas file, but it
         establishes a libref named frompdata libref. (This is the source library of the data set in
         this exampleproject datasets.) */
libname to '/projects/data';
      /* This is not necessary if this statement is already in your autoexec.sas file, but it
         establishes a libref named to. (This is the destination data set library.) */
data to.data2export (keep = var1 var4 var73 var105);
      /* List the variables from data set data2004a that you want in the new data set after keep
         = , no separator is need. */
   set pdata.data2004a;
run;
```

3.2.7.2 Exporting to csv

As far as we know, and contribution by users would be appreciated on this, all applications can import .csv files. Once you have created the extract to.data2export, then you can write code to create a csv file. Let's say that you want to save this in the same directory and that you want to name the new csv file mydata.csv. Here is the SAS program will write.

```
proc export data = to.data2export
      /* The libname statement was already issued either in your autoexec.sas file or in the
         "Creating a SAS Data Extract" step. */
   outfile = "/projects/data/mydata.csv"
   dbms = csv replace;
      /* Notice that there is no semicolon after the first and second line of this code! */
run;
```

You will then need to create code to import this new data set to the application you want to use for data analysis. You need to be careful as there are known problems with this method: PPNs can inadvertently be converted to a numeric format; if a PPN starts with a zero, the zero could be removed. The exact difficulties in using this method will vary depending on your set of circumstances. You should always check your data after you've converted it in order to identify any problems. Some problems are minor and may only be annoying (such as converting a ten-digit PPN with a leading zero into a nine-digit number) others may critically effect your program.

To get basic statistics on the original data set, type the following in SAS.

```
proc contents data = to.data2export;
run; /* Lists basic information on the data set: size, variables, etc. */
proc means data = to.data2export;
run; /* Gives basic statistics on numeric variables: mean, min, max, etc. */
proc print data = to.data2export (obs=5);
run; /* Prints the first 5 observations in the data set. */
```

You will need to write equivalent code in the application you want to use for data analysis to determine if there are any potential problems. Again, any assistance by users would be appreciated on this topic.

3.2.8 SAS Data Quality

To use SAS-DQ, you need to tell SAS where the DQ file is located. Add this line to the top of your SAS program.

```
%DQLOAD (DQLOCALE=(ENUSA), DQSETUPLOC='
/apps/SAS/v9.4/SASHome/SASFoundation/9.4/misc/dquality/dqsetup.txt');
```

If your program will not run, regardless of the error message (or lack thereof), try running your program as if you were having a memory error with SAS. Instructions are specified in section 3.2.5 on page 16.

3.3 Stata

3.3.1 Getting Started

To launch a Stata batch job, at the terminal prompt type **`qstata program_name.do`** [ENTER]

To launch a Stata batch job adjusting memory allotment, at the terminal prompt type
`qstata --program_name.do --memsize=14000` [ENTER]
(14000 equals 14 GB)

To start an interactive Stata session, at the prompt of a terminal window, type
`qsub -IX` [ENTER] to get to a research node, then type **`xstata-se &`** [ENTER].

Stata-MP is available and to use it, follow the instructions above by substituting mp for se. Use your good judgement as to whether you **need** to use mp or if se is suitable. If in doubt, use Stata SE.

3.3.2 Stata Frames and .do Files

Stata Frames
Stata opens as one large window that is divided into several sections termed "frames."

Output frame	Contains commands you send to Stata and Stata's response.
Command frame	You issue commands to Stata from this frame.
Variables frame	Every variable in the loaded data set is listed here.
Previous Commands frame	Every command you have issued to Stata during the session is stored here.

The "Previous commands" and "Variables" frames are both intended to make Stata easier/faster to use interactively since you can simply click on variable names or previous commands to load them into your command window instead of typing the entire command or list again.

Stata .do Files

Do-files are ASCII files that contain a set of Stata commands to run specific procedures. It is highly recommended to use do-files to store your commands so you do not have to type them again should you need to re-do your work. You can use any text editor and save the file in ASCII format; or, while in interactive mode, you can use Stata's 'do-file editor' with the advantage that you can run the commands from there. Alternatively, in the command window type doedit.

3.3.3 Importing SAS data Files

All data files provided by Census are in SAS format. Stata can now natively use these files. You can convert SAS files by importing them in Stata and saving them to your project directory. Before you do this, be sure to spend time with your data documentation and disclosure avoidance materials so that you understand which variables you should keep for your analysis and for production of disclosure avoidance review statistics. Remember, the IRE resources are limited, so **do not convert entire datasets** and store them in your project space. When importing to Stata, select only the variables you know you will be using for your research. See section 3.3.5 for additional information on conserving resources when using Stata.

3.3.4 Using .ado Files

To use an .ado file you can

(a) specify the full path when you call it from your do program (e.g., **`/apps/shared/stata/X/<ado_file_name>`**),
(b) copy it to your Stata working directory where Stata will see it (type adopath in Stata to see the order Stata looks for ado files), or
(c) set a system directory with the stata sysdir command and define its path. For example, **`sysdir set PLUS "/projects/programs/ado"`** will reassign PLUS (this usually points to ~/ado/plus/ which is not a directory in our systems) to your personal ado directory in your project space (you will need to create this directory). Type the command sysdir in Stata to check that this worked.

The RDC servers are intentionally isolated from the web, which prohibits the standard Stata procedure for applying updates to the software. If you need an ado file, first double check that it is not on the server. To check you can use Thunar and drill down in directory /apps/shared/stata or use the command line to look for a specific item. From a compute node, at the command line type

`find /apps/shared/stata -name "*look*"` [ENTER]

replacing **look** with what you are looking for. If what you need is not on the server, find the link to it outside the RDC from the Stata web site, and submit the link to your administrator. Please note that not all requests can be met due to OS or version incompatibility.

3.3.5 System Performance for Stata Users

It is CRITICAL to manage memory when using Stata. Stata sessions require dedicated system memory allocated to them and this may cause the server to run out of system memory. Please use the following guidelines.

1) Do program development and debugging on a subset of data. Most interactive use of Stata can be performed with a subset of observations. The command to get a subset is **use <FileName> if _n<=200** (This uses only the first 200 observations of the file.)
2) Don't use the whole data set if you don't need to. Regularly, to open a Stata data set, you type **use data_set**. If you want to run analysis on specific variables in that data set, let's say variables named a, b, and c; then in Stata type **use a b c using data_set**.
3) Close out Stata to release memory. Memory is not released by Stata until the application itself is closed out, even if the program finishes running or if one resets memory with the set memory command. So, if you are not actively using Stata, then please close it.
4) Do not open multiple interactive sessions of Stata. (This follows from number 3.)
5) Run debugged Stata jobs in batch mode.
   qstata program_name.do &
6) Be courteous to your fellow users.
7) Kill off abandoned jobs. If you exit abnormally, then in many instances the system leaves your job(s) running or idling with the memory tied up. This strains the system and causes it to behave erratically. Use **qdel <job_id>** [ENTER] to delete jobs that are running or in the PBS Pro queue.

3.3.6 Stata Help

Some RDCs have a set of Stata manuals and there is reference material on rdcdoc. Outside of the RDC, you can access Stata online documentation. A couple good links are http://www.stata.com/links/resources1.html and http://www.stata.com/support/faqs/.

3.4 MATLAB

To launch a MATLAB batch job, create a batch script. We name the below script matlab.bash.

	Script text	Description
(1)	`#!/bin/bash`	(1) Specifies bash script, do not change this.
(2)	`#PBS -l ncpus=1,mem=13gb`	(2) Tells PBS Pro to expect directives and what they are. For clarity, the items after #PBS is dash "L", and ncpus how many cpu you need.
(3)	`cd /projects/programs`	(3) Change directory to full path of program file.
(4)	`matlab -nodisplay -nosplash -r program_name > program_name.log`	(4) Command to launch job. This line is recreated below to show where spaces are.

```
matlab -nodisplay -nosplash -r program_name > program_name.log
```

Then at the prompt pointing to the path where the script resides
qsub matlab.bash [ENTER].

All users of the RDC share a limited number of MATLAB licenses. If you absolutely must run more than one MATLAB job at a time, then be sure to run them on the same node. This is because licenses are assigned based on node and user combination. To run jobs on the same node, create your matlab job scripts as above – say they are called matlab1 and matlab2. Then to ensure that they run on the same node, add that requirement to your resource request list in the qsub command. For example:

```
qsub -l host=hpc-compute1 matlab1
qsub -l host=hpc-compute1 matlab2
```

will require that the jobs matlab1 and matlab2 be run on the host hpc-scompute1. If hpc-compute1 is busy, the jobs will queue until resources become available.

Outside of the RDC, you can access MATLAB online documentation.
http://www.mathworks.com/access/helpdesk/help/techdoc/index.html?/access/helpdesk/help/techdoc/

3.5 R

To launch a R batch job, at the terminal prompt type **qR program_name** [ENTER]
To launch a R batch job specifying resources, at the terminal prompt type
qR --program=program_name --cpucount=1 --memsize=15000 [ENTER]
Note: example is for 15 GB of memory; adjust the value in accordance with your needs.

To start an interactive R session, you can either use the terminal or RStudio. In either case you start by initiating an interactive PBSPro session. At the prompt of a terminal window, type **qsub -IX** [ENTER] to get to a research node.

For a terminal session, then type **R** [ENTER]. Be sure not to use lowercase “r” as this will give you an error message. Your shell will be occupied; open another shell in the same Terminal window if you need it. (This creates tabs for the shell sessions and you can easily toggle between them.)

For RStudio, then type **rstudio** [ENTER]

To launch a R batch job, create a batch script. We name the below script r_batch.bash.

	Script text	Description
(1)	`#!/bin/bash`	(1) Specifies bash script, do not change this.
(2)	`#PBS -l ncpus=1,mem=13gb`	(2) Tells PBS Pro to expect directives and what they are. For clarity, the items after #PBS is dash “L”, and ncpus how many cpu you need.
(3)	`cd /projects/programs`	(3) Change directory to full path of program file.
(4)	`R CMD BATCH r_prog.R`	(4) Command to launch job.

Then at the prompt pointing to the path where the script resides
qsub r_batch.bash [ENTER]

The RDC servers are intentionally isolated from the web, which prohibits installing packages. If you need specific package, first double check that it is not on the server. From a compute node, at the command line type

`find /apps/R/lib -name "*look*"` [ENTER]

replacing **`look`** with what you are looking for. If what you need is not on the server, locate it outside of the RDC on the Comprehensive R Archive Network (CRAN) https://cran.r-project.org/ and email your administrator with the request. Please note that not all requests can be met due to OS or version incompatibility.

Outside of the RDC, you can access R online documentation.
http://cran.r-project.org/doc/manuals/R-intro.html and http://cran.r-project.org/manuals.html

3.6 Python – Anaconda

We use the Anaconda distribution of Python in IRE, and we use conda whenever possible to manage various environments, each of which contains different packages, versions of Python, etc.

To activate Anaconda Repository in IRE type: **`qsub -IX`** [ENTER] then
`source /apps/anaconda/bin/activate py3` [ENTER]
or
`source /apps/anaconda/bin/activate py3_spatial` [ENTER] (for mapping features)
or for the most current Python with the most complete set of additional packages from both Anaconda and PyPI (currently conda forge version 3.8) *use:*
`source /apps/anaconda/bin/activate py3cf` [ENTER]

To view the packages in the chosen environment: **`conda list`** [ENTER]
To view the list of available environments: **`conda env list`** [ENTER]

To start Jupyter notebook go to a compute node using the command: **`qsub -IX`** [ENTER], then
`source /apps/anaconda/bin/activate py3` [ENTER]

Note: If you do not include the name of the environment -- "py3" -- you will activate the "base" environment, which might not even have jupyter installed. Once you've activated the py3 environment, navigate to the directory you'd like to be in (i.e., where your notebooks are), and run: **`jupyter notebook`** [ENTER]. To assign a browser do:
`jupyter notebook --browser='/usr/bin/firefox'` [ENTER]

Note there are some things to keep in mind when running jupyter notebooks having to do with situations where you get unexpectedly disconnected. There is not a good way of getting back to your disconnected session right now, and disconnected sessions get killed after four hours, whether that jupyter notebook you have running in it is finished or not. It is possible to run notebooks in batch mode (instructions below), which you should probably do for long running jobs.

Things you should NOT do:

- Don't run jobs (including jupyter and spyder) on the login nodes
- Don't create your own conda environments, install packages, etc. This is for several reasons:
    - Each new conda full environment takes approximately 3GB of space.

- These individual environments are not shareable across projects, and are by default created in /home, which is much smaller than /projects.
- Creating such an environment prevents coordinated patching efforts in the Python setup.

To launch a Python batch job, create a batch script. We name the below script python.bash.

	Script text	Description
(1)	`#!/bin/bash`	(1) Specifies bash script, do not change this.
(2)	`#PBS -l ncpus=1,mem=8gb`	(2) Tells PBS Pro to expect directives and what they are. For clarity, the items after #PBS is dash "L", and ncpus how many cpu you need.
(3)	`source /apps/anaconda/bin/activate py3cf`	(3) Activate Python environment.
(4)	`cd /projects/programs`	(4) Change directory to full path of program file.
(5)	`python python_prog.py`	(5) Command to launch job.

Then at the prompt pointing to the path where the script resides
`qsub python.bash` [ENTER]

If something is not present in the environment that you are using, locate the desired package outside the RDC on the Python Package Index (https://pypi.org/) and e-mail your Administrator with the request to install it. We will install it if possible. It is not always possible.

3.7 Stat/Transfer

A few notes to start:

- Do not copy entire datasets from the data warehouse to the project space. Before you make any extracts from the Census provided data in the warehouse, you must spend time with the documentation to understand which variables you need to conduct your analysis *and* which variables will be needed to produce statistics for disclosure avoidance review.
- Stata can use SAS datasets natively.

To open interactive StatTransfer, first at the prompt of a terminal window, type
`qsub -IX` [ENTER] and then **`stattransfer &`** [ENTER].

On Transfer tab select type (SAS) and location of source dataset in the first box and then select data type and destination of the output dataset.

On Variables tab, if your source dataset has only a few variables it might make sense to deselect variables individually by unchecking the radio button to the variable name on the left and then create your extract; however, in most cases, it makes more sense to start with none selected and specify the variables you want. To do this, in Quick Variable Selector box (on the right), type * and then click Drop. Then in Quick Variable Selector box type the variables you need, then click Keep.

To transfer your dataset, return to the Transfer tab and click Transfer.

For more information, the StatTransfer manual is on rdcdoc.

3.8 GRASS

To open interactive GRASS, first at the prompt of a terminal window, type
`qsub -IX` [ENTER] and then **`grass`** [ENTER].

We recommend creating directory /projects/GRASS for your database directory; populate this in field 1 of GRASS interface. **DO NOT** create a database in your home directory.

3.9 Other Applications

3.9.1 Terminal

Terminal is a program that manages shells for you; each terminal session is given a tab. Some helpful commands for Terminal are

[CTRL]+c ends the command and returns to prompt. This is helpful if you submit a command and are getting *a lot* of results.

↑ and ↓ at the prompt, these browse the command history.

[SHIFT]+[PAGEUP] and [SHIFT]+[PAGEDOWN] allows you to brows the terminal buffer (to see the text that has scrolled off the screen).

[TAB][TAB] (tab pressed quickly in succession) at the prompt will list all commands available on the system that fit the criterion you specify. For example, if you type **`b`** at the prompt and then press [TAB][TAB], all commands that begin with b will be shown.

Terminal will not automatically open pointing to the desired directory. You will need to
`cd /projects/programs/` `[ENTER]` for example.

3.9.2 LibreOffice

LibreOffice is an open-source office software suite. This and any other word processor or spreadsheet application should be run only on a compute node. To open LibreOffice, first at the prompt of a terminal window, type
`qsub -IX` [ENTER] and then **`ooffice &`** [ENTER]

From there you can “Open File” from the left navigation bar.

If you wish to disclose a formatted document that you created using LibreOffice, you can either "save as" another extension, though Microsoft Office and Google Docs can use and manipulate odt and ods files. Note if using “save as” and you do not see file type, you can either provide both the name of the file and the file extension in the name field or try resizing the window. I attempting the latter and having trouble, let your administrator know and s/he will request assistance from IT.

3.9.3 Text Editors

We offer a number of text editors on the thin client system. These include gedit, nano, Mousepad and emacs. You can launch most of them from Panel 1 → Applications → Accessories.

3.9.3.1 Visual Studio Code

We recommend that you use the text editor that you are comfortable or familiar with; however, if you are not committed to one or the other, you might want to use Visual Studio Code because it will highlight your code making it easier to edit programs without running session of the relevant application (SAS, Stata, etc.). Visual Studio Code should be run only on a compute node:
`qsub -IX` [ENTER] and then **`code &`** [ENTER]

3.9.4 Version Control

To create a local repository for your project, follow instructions at rdcdoc → IRE Documentation → Using GIT in IRE. You will follow instructions for "Local Repository."

3.9.5 Chat Feature

You can communicate freely to other logged-in project researchers using the chat feature within IRE. To start a chat at the prompt type **`ire-chat`** [ENTER].
Any other researcher logged in to the project space can do the same to join the chat.

If you prefer to get a log of your conversation history, instead type at the prompt
`ire-chat –write-logs /projects/log_name.txt` [ENTER].

4 Trouble Shooting

If you are experiencing computing troubles, then these are the things to try and/or check before notifying your Administrator. Once you have located the description of your problem, it is best to follow the remedial steps in the order given.

4.1 The Description of the Problem is…

4.1.1 An Authentication Error

The problem is that the system cannot confirm your authority to log in.

1) Check your username and password.
2) Reboot the client. Do NOT just press the power button to reboot the thin client! Click "Cancel" to return to the launch menu. From there, click client → restart. After reboot, attempt log in again.
3) The problem is with your account, which can be due to many things (e.g., the number of failed log in attempts, an expired password, etc.). Call the IT Service Desk at 301-763-3333. Follow the prompts for VDI password.

4.1.2 A Connection Time-out

The problem is that the thin client is not communicating with the server.

1) Make sure the data cable is connected properly to the thin client. If it was attached improperly, then try to log in again.
2) If the cable was fine or if the second log in attempt also failed, reboot the client.
3) Try a different thin client.
4) Notify your Administrator.

4.1.3 Session Terminates

Although this usually happens when there is a significant problem with the servers, sometimes the system hiccups and sessions are terminated for no real reason.

1) Reboot the client and attempt log in again. If asked if you would like to resume your previous session, say yes. If this does not work, then
2) Return to the sign-in desktop and try again; but if given the option, then *terminate* the previous session.
3) Notify your Administrator.

4.1.4 Session Will Not Spread Across Monitors

1) Hover in the upper right corner of the NoMachine! session until the corner peels backward and click on it → Display → Fullscreen on all Monitors.
2) Notify your Administrator.

4.1.5 Session Will Not Unlock

1) Get help from your administrator, then
2) Follow instructions to disable screen saver in section 1.3.1 page 2.

4.1.6 Problem with Permissions

1) Notify your administrator of the affected directory(ies) and/or file(s). This issue needs to be sent to IT by the administrator.

4.1.7 Citrix Receiver Uninstalled Error

1) Report this to your administrator. S/He needs to issue a IT support ticket to resolve.

4.1.8 LibreOffice Will Not Open

1) At prompt type, **`pgrep <jbid> "oosplash"`** [ENTER], then
2) Kill all resultant processes listed: **`kill ######`** [ENTER].
3) If this doesn't help, report to your administrator.

4.1.9 Small, Unusable Window

1) Find tile in Panel 1 → right-click it → "Maximize All" or
2) Find tile in Panel 1 → right-click it → "Resize Window". Then use cursor to make larger.

4.2 Reporting Problems

Any time you have a problem with the thin clients or IRE, please notify your Administrator. The more detail you can give them about what went wrong, how you tried to fix it, whether you were successful or not, the more likely it is they will be able to help you fix it. You should **always** provide your Administrator with the following information.

1) Your JBID
2) Date and time the problem began
3) The node(s) you were using. – login 1 or 2, compute##
4) If it is a PBS Pro job, PBS Pro job number(s)
5) **The exact wording of any error message**
6) A detailed description of the problem, including what applications and/or data sets you were using, what you were doing when the problem occurred
7) Description of any remedial steps you took to diagnose and solve the problem

If you are unable to contact your local RDC Administrator in a reasonable amount of time, you may try contacting the Administrator at a different RDC. You can find contact information at https://www.census.gov/about/adrm/fsrdc/locations.html, but you will need to leave the RDC to access this information because devices with internet connectivity are prohibited in the RDC. All computer problems should be reported to a RDC Administrator.