Intro to the Command Line (using Bash)

Contents

[WHAT IS A COMMAND LINE INTERFACE (CLI)? 1](#_Toc126058131)

[WHY SHOULD I CARE ABOUT A CLI ? 1](#_Toc126058132)

[DIFFERENT CLIs FOR DIFFERENT COMPUTER SYSTEMS 2](#_Toc126058133)

[What is the "Bash Shell" program? 2](#_Toc126058134)

[How to use Bash in RStudio's "Terminal" window 3](#_Toc126058135)

[The short answer: 3](#_Toc126058136)

[Problems running Bash in RStudio? Try changing some options in RStudio. 3](#_Toc126058137)

[What is the Bash "prompt" ? 4](#_Toc126058138)

[How to change the Bash prompt 5](#_Toc126058139)

[How can I confirm that I'm using the "Bash" Command Line Interface program? 5](#_Toc126058140)

["Bash commands" or "Linux commands" ? 6](#_Toc126058141)

# WHAT IS A COMMAND LINE INTERFACE (CLI)?

Modern computers and the programs that they run allow you to click and drag icons, use menus, open many programs at once, etc. It's hard to imagine a computer that doesn't have these features. However, a long time ago (pre 1990'ish) none of this was possible. In order to use a computer you had to memorize many different commands. You would type these commands into the computer in order to run programs and interact with the computer.

Today the modern computer "interface" (the one that has icons, menus, etc.) is known as a Graphical User Interface or GUI (pronounced "gooey"). The old-style interface that required that you type commands to do ANYTHING is still able to be used. This type of interface is known as a "Command Line Interface" or CLI.

# WHY SHOULD I CARE ABOUT A CLI ?

For day to day casual computer work, it may be hard to understand why anyone would use a CLI instead of a modern GUI interface. Using a CLI requires memorizing many different seemingly arcane commands instead of just "pointing and clicking" on what you want. Why would anyone want to do that?

* **Reason 1:** A CLI allows you to do powerful things quickly. Many of the CLI commands can be quite powerful and can cut down the amount of time it takes to do things by A LOT. For example, suppose a folder contains 1000 files. For example, "expenseReport-Jan2020.xlsx", "meetingMinutes-Jan012020.docx", etc. (1000 such files) You are told by your boss that you must add your name as part of the filename for each of these 1000 files. If you are using a GUI, that could take hours. However, for someone who is proficient with a using a CLI, it could take seconds to type in the correct commands to do that for all 1000 files.
* **Reason 2:** You can create "scripts" to automate many things. Suppose your boss told you that everyone in the company needs to make the same changes to their filenames. However, not everyone knows how to use a CLI. You could theoretically create a small "script" (ie. a program) using the CLI commands that would allow anyone to "run the script" to make the changes. Tasks that need to be done repeatedly are often best accomplished by creating a CLI script. The script can then be run anytime the task needs to be done. Note that you can create similar scripts or programs using another programming language but a CLI script is often the easiest and most appropriate method to create such scripts.
* **Reason 3:** Many basic concepts from the command line carry over directly to any programming language (including R) in relation to interacting with the operating system. Concepts such as absolute and relative paths, wildcards in filenames, etc. come up over and over again when you work with data.
* **Reason 4:** Technology and data professionals are often expected to understand at least the basics of using a CLI. Often many advanced commands are only available to be run from the command line. For example, the wget program is a command line program that can be used to download the code for an entire website or just specific files that you'd like to get.

# DIFFERENT CLIs FOR DIFFERENT COMPUTER SYSTEMS

Just as the Graphical User Interface (GUI) for the Mac is different from the GUI for Windows, the standard CLI available for the Mac is different from the CLI(s) that are available on Windows. This means that the commands that you need to type on Mac are different from the commands that you would type on Windows.

* Mac  
    
  The Mac CLI is used by running the Terminal program that comes with the Mac. The terminal program actually allows
* Windows  
    
  Windows comes with two different CLIs built into the system:
  + **The "Command Prompt" (cmd) :** This was the original CLI for Windows. It is a little bit dated and not as powerful as other options. Nevertheless it is still very popular.
  + **"Powershell" :** This is a newer CLI that Microsoft first introduced in 2006. It is more complicated to use than the "Command Prompt" but in many ways is more powerful. Today Windows comes with both CLIs.
  + **Bash:** You can also download a version of the "Bash Shell" CLI program that will work under Windows. See below for more info about Bash.

# What is the "Bash Shell" program?

While most computer users today use either the Windows or Mac operating system, the Linux operating system is very popular for running "server" computers. These are the computers that support many of the "backoffice" tasks in companies today. These include computers that host company websites as well as company databases and many other critical corporate technology services.

Linux is a derivative of the Unix operating system. These operating systems, (Linux and Unix), have many different "command line interface" (CLI) programs available to choose from. A very popular CLI on these computers is the "Bash Shell". The word "shell" is a synonym for a "command line interface". *The word "shell" was first coined by the creators of Unix. The idea is that to interact with guts of the operating system you must "go through" the shell (i.e. use the CLI commands), similar to the way you must "go through the shell" to get to the inside of an egg.*

It should be noted that

Note that the "Bash Shell" has become somewhat of a standard. Apple Mac is also derived from the original Unix operating system and by default, the Mac Terminal program also runs a version of the Bash Shell. While Windows doesn't natively come with the Bash Shell, you can download many different versions of a "Bash" CLI program that can be used on Windows. Some of these are:

* git bash - In my opinion (in 2023) for someone who just want to learn the basics of Bash, this is the easiest option to try. The other options may be a little more work to install. To install this option, go to gitforwindows.org , download the program and install it. For more info, see the MS Word document entitled "HowToInstallBashOnWindows.docx".
* Cygwin
* mingw64
* etc.

# How to use Bash in RStudio's "Terminal" window

## The short answer:

You can type Command Line Interface (CLI) commands directly in RStudio. To do so, choose the following menu choices to open the Terminal tab: ***Tools menu > Terminal > Move Focus to Terminal***

Some versions of Mac and all versions of Windows come with different CLI programs. To use the Bash CLI, choose Bash from the following menu (this is covered in more detail in the following section of this document):

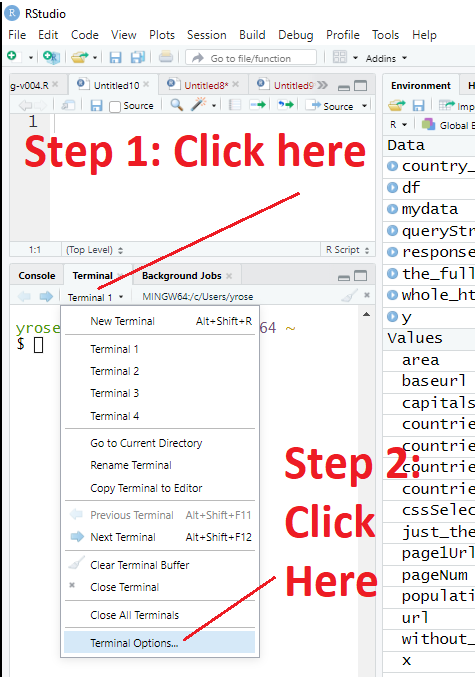
***Tools menu > Global Options > Terminal > New terminals open with (dropdown list) >  
 (choose Bash or Git Bash or whatever version of Bash you have installed)***

## Problems running Bash in RStudio? Try changing some options in RStudio.

There are MANY ways to configure the RStudio environment. If you're having touble running Bash from the Terminal window in RStudio, follow the steps shown below. (To learn a lot more about how to use the Terminal window in RStudio, see the official documentation here:  
 https://support.posit.co/hc/en-us/articles/115010737148-Using-the-RStudio-Terminal   
(You can also get to this page by clicking on the link entitled "Using the RStudio terminal" in the RStudio Terminal options window, as shown in the screen-shots below.)

**Are you having problems getting the "Bash shell" command line interface (CLI) to run in RStudio?**

1. If you don't see the RStudio "Terminal" tab anywhere in the RStudio interface then choose, the following menu choices from RStudio's main menu: Tools > Terminal > Move Focus to Terminal
2. If you DO see the Terminal tab but you aren't running the "Bash shell" command line interface in the Terminal, then do the following:

 Graphical user interface, text, application

Description automatically generated

RStudio is highly configurable. To modify any of the plethora of options for RStudio - choose the "tools" menu in RStudio and select "General Options". That will open the window shown in the 2nd screenshot above. You can then choose from any of the categories on the left hand side of that window to access the options under that category. For example, to change the font used in the editor you can click on the "Appearance" category (on the left side of the Options window) - at which point you will see several options including those allowing you to specify the font, font-size, etc.

# What is the Bash "prompt" ?

When you type commands in the R console, R "prompts" you to enter a new command every time you see the "> " at the beginning of a line. For example, the following shows an R command being run in the R Console window. The ">" at the beginning of the second line is the R "prompt", where you can enter your next command:

***The "R" prompt below is ">"*** 

In a similar way, before you type a command in Bash, the CLI "prompts" you to enter a new command. However, in Bash the exact "prompt" can be different on different computers and for different users. Often the Bash prompt is something like this:

username@hostname:~$ (TYPE YOUR NEXT BASH COMMAND HERE)

For example, on my computer, the following shows when happens on my computer when I run the Bash command, "expr 5 \\* 100" , which simply multiplies 5\*100 to get 500. The next line shows the prompt waiting for the user to type another command:

***The Bash prompt below is: yrose@LAPTOP-IGVE3JIE:~$***A picture containing text

Description automatically generated

In this example, "yrose" is my username on my computer and LAPTOP-IGVE3JIE is the "name" of my computer *(many people don't realize that their computer has a name)*. This type of prompt became popular since power‑users of Bash often connect remotely to many different computers from their Bash window. By just looking at the Bash prompt a power‑user who might be connecting to different machines can know immediately which computer they are currently connected to.

The ~ is called a "tilde". The significance of the tilde, ~, will be discussed more later. Finally, the $ at the end of the prompt is there to show where you are to enter your next command. Historically, the Bash prompt was JUST a dollar sign, $ , similar to how the R prompt is just a greater than sign, >. In subsequent years, Bash evolved to show more information in the prompt. The prompt can be configured to show a lot of different types of information (see the next section).

## How to change the Bash prompt

As noted above, the Bash prompt can be configured to include different types of information. Sometimes it is simplest to just see a simple '$' prompt. To do so, type the following command. There shouldn't be any spaces on either side of the equals sign:

PS1='\$ '

After doing so your Bash prompt should become a single dollar sign, $. To change the prompt to the more typical version shown in the previous section type the following command:

PS1='\u@\h:\w\$ '

The \u stands for the username, \h for the hostname and \w for the "working directory" (we'll discuss what that means later). See the following page for more information about how to configure your prompt in all sorts of interesting ways,

<https://linuxhint.com/bash-ps1-customization/>

Changing your prompt to the "working directory" makes navigating the Bash environment much easier. Once you are running Bash, to see the "working directory" in the prompt, type the following Bash command. NOTE: there are no spaces ANYWHERE in this command other than the one space that appears before the final single quote (i.e. apostrophe):  
  
 **PS1='\w\$ '**

Now a user, joe, who runs the following commands will see the results shown below.  
  
 $ cd # change the working directory to /Users/joe  
 $ PS1='\w\$ ' # change the prompt to include the working directory  
 ~$ pwd # ~ is shorthand for /Users/joe  
 /Users/joe  
 ~$ cd Documents # move to another directory  
 ~/Documents$ pwd # show the path for the current directory  
 /Users/joe/Documents

If you want this to happen every time you start a new Terminal window, then run the following command. This command creates a file named .profile (there is a period at the very beginning of the filename) in your "home directory". This file contains options that control how Bash runs.

**echo "PS1='\w\$ '" >> $HOME/.profile**

In the rest of this document, I will use a simple $ prompt. This makes it easier to read the information in this document.

# How can I confirm that I'm using the "Bash" Command Line Interface program?

As noted above, both Windows and Mac can be configured with different CLI programs. The RStudio Terminal window can be configured to use any CLI that is installed on your computer. The instructions to do so appear earlier in this document. To confirm that you are actually using the Bash CLI, you can run the following command:  
 echo $0  
If you are running the Bash shell/CLI, then the output should include the word "bash" somewhere. For example the following is what I get when I run this command on my computer:

  
If you do not get a result showing the word "bash" somewhere, refer to the sections above in this document that describe how to configure RStudio to use Bash.

# "Bash commands" or "Linux commands" ?

Technically, many of the commands that you run in Bash are actually "Linux" commands (or Unix commands). Therefore when searching online for information, it is often more appropriate to use the words "Linux command" in your search instead of "Bash command". As we said above, the Bash shell, is only one of many CLIs that are available on Linux. However, the majority of commands we will learn are able to be used with ANY Linux CLI and not just Bash.

In addition to being able to use Bash on Apple Mac, you can also access many of the vast majority of these "Linux" commands on Mac. (Remember that Linux and Mac are both derived from the Unix operating system). Similarly, when you install a Bash environment on Windows, it will include many of these "Linux" programs too.