

# External Server Guide

Banco de Portugal's Microdata Research Laboratory (BPLIM)

October 2025

## Contents

<b>1 Access to the External Server</b>	<b>3</b>
1.1 Overview . . . . .	3
1.2 Upon Access Approval . . . . .	3
1.3 Password Policy . . . . .	3
1.4 First Steps . . . . .	3
1.4.1 Logging In . . . . .	4
1.4.2 Accessing Your Project . . . . .	4
1.4.3 Understanding Your Project Structure . . . . .	6
1.4.4 Logging Out . . . . .	6
<b>2 Important Guidelines</b>	<b>7</b>
2.1 Managing Disk Space . . . . .	7
2.1.1 Critical Rule: Do Not Save Files in Your Home Folder . . . . .	8
2.1.2 Monitoring and Cleaning Your Project Folder . . . . .	8
2.2 Using the Terminal . . . . .	8
2.2.1 Accessing the Terminal . . . . .	9
2.2.2 Essential Terminal Tips . . . . .	9
<b>3 Working with Containers</b>	<b>10</b>
3.1 What Are Containers? . . . . .	10
3.2 Starting Containers . . . . .	10
3.2.1 Method 1: Using Launcher Scripts (Recommended) . . . . .	10
3.2.2 Method 2: From the Terminal . . . . .	10
3.2.3 Method 3: Direct Execution . . . . .	11
<b>4 Statistical Software</b>	<b>11</b>
4.1 Stata . . . . .	11
4.1.1 Starting Stata . . . . .	11
4.1.2 Ado-files . . . . .	11
4.1.3 Temporary Files . . . . .	12
4.1.4 Running Stata in Batch Mode . . . . .	13
4.1.5 Running Stata with Screen . . . . .	13
4.2 R . . . . .	13
4.2.1 Starting R/RStudio . . . . .	13
4.2.2 Running R in Batch Mode . . . . .	13
4.2.3 Running R with Screen . . . . .	14
4.3 Python . . . . .	14

4.4	Julia . . . . .	14
4.5	Structuring and Writing Code . . . . .	15
4.5.1	Organize your project . . . . .	15
4.5.2	Naming and layout . . . . .	15
4.5.3	Paths and reproducibility . . . . .	15
4.5.4	Ado-files and packages . . . . .	15
4.5.5	Batch and logging . . . . .	15
4.5.6	Editors and tools . . . . .	16
4.5.7	Data handling . . . . .	16
4.5.8	Using the BPLIM templates . . . . .	16
4.6	Updates to Commands and Packages . . . . .	16
4.7	Build a Container to Fine-Tune Your Statistical Packages . . . . .	16
<b>5</b>	<b>Running Programs in Batch Mode</b>	<b>17</b>
5.1	Why Use Batch Mode? . . . . .	17
5.2	Basic Batch Workflow . . . . .	17
5.3	Scheduling Jobs . . . . .	17
5.4	Managing Batch Jobs . . . . .	18
5.5	Monitoring Running Programs . . . . .	18
5.5.1	Using top . . . . .	18
5.5.2	Using tail to monitor log files . . . . .	18
<b>6</b>	<b>Using Screen for Persistent Sessions</b>	<b>18</b>
6.1	Basic Screen Usage . . . . .	18
6.2	Example: Running Stata with Screen . . . . .	19
6.3	Example: Running R with Screen . . . . .	19
6.4	When to Use Screen vs Batch Mode . . . . .	19
<b>7</b>	<b>Allowed Outputs</b>	<b>19</b>
<b>8</b>	<b>Requesting Outputs</b>	<b>20</b>
8.1	Projects Using Modified Data . . . . .	20
8.2	Projects NOT Using Modified Data . . . . .	20
<b>9</b>	<b>Managing Your Home Folder</b>	<b>21</b>
<b>10</b>	<b>Project Archival Policy</b>	<b>21</b>
<b>11</b>	<b>Appendix</b>	<b>21</b>
11.1	Basic Shell Commands on Linux . . . . .	21
11.2	Using the vi File Editor . . . . .	23
11.3	Password Requirements . . . . .	24
11.4	NoMachine: Frequently Asked Questions . . . . .	24
11.5	Download, Install and Configure NoMachine Client . . . . .	27
11.6	Version Control with GitLab . . . . .	37
11.6.1	Benefits of Using Version Control . . . . .	37
11.6.2	Getting Started with Git . . . . .	37
11.7	Building Custom Containers . . . . .	41
11.7.1	Steps to Build a Custom Container . . . . .	41
11.8	Jupyter Lab . . . . .	42
11.8.1	Starting JupyterLab . . . . .	42
11.8.2	Example JupyterLab Session . . . . .	43

# 1 Access to the External Server

## 1.1 Overview

This guide will help you connect to BPLIM's external server, navigate your project environment, and use statistical software (Stata, R, Python, Julia) for your research. The server runs on Linux and uses containerized environments to ensure reproducibility and consistency across projects.

### Key features:

- Secure remote access via NoMachine client
- Isolated project environments with defined folder structures
- Statistical software running in containers
- Support for interactive and batch processing modes

## 1.2 Upon Access Approval

Once access is approved, you can connect to the external server using the **NoMachine** client. See [Download, install and configure NoMachine client](#) for detailed instructions.

---

## 1.3 Password Policy

Passwords are a critical security component. Your initial password must be changed at first login, and all passwords must comply with the requirements below.

### Password requirements:

- **Minimum length:** 8 characters
- **Character classes:** At least 4 different types (uppercase, lowercase, numbers, punctuation)
- **History:** Cannot reuse any of your last 7 passwords
- **Expiration:** Passwords expire after 60 days and must be changed at next login
- **Failed attempts:** After 6 consecutive failed login attempts, your account will be locked for 10 minutes

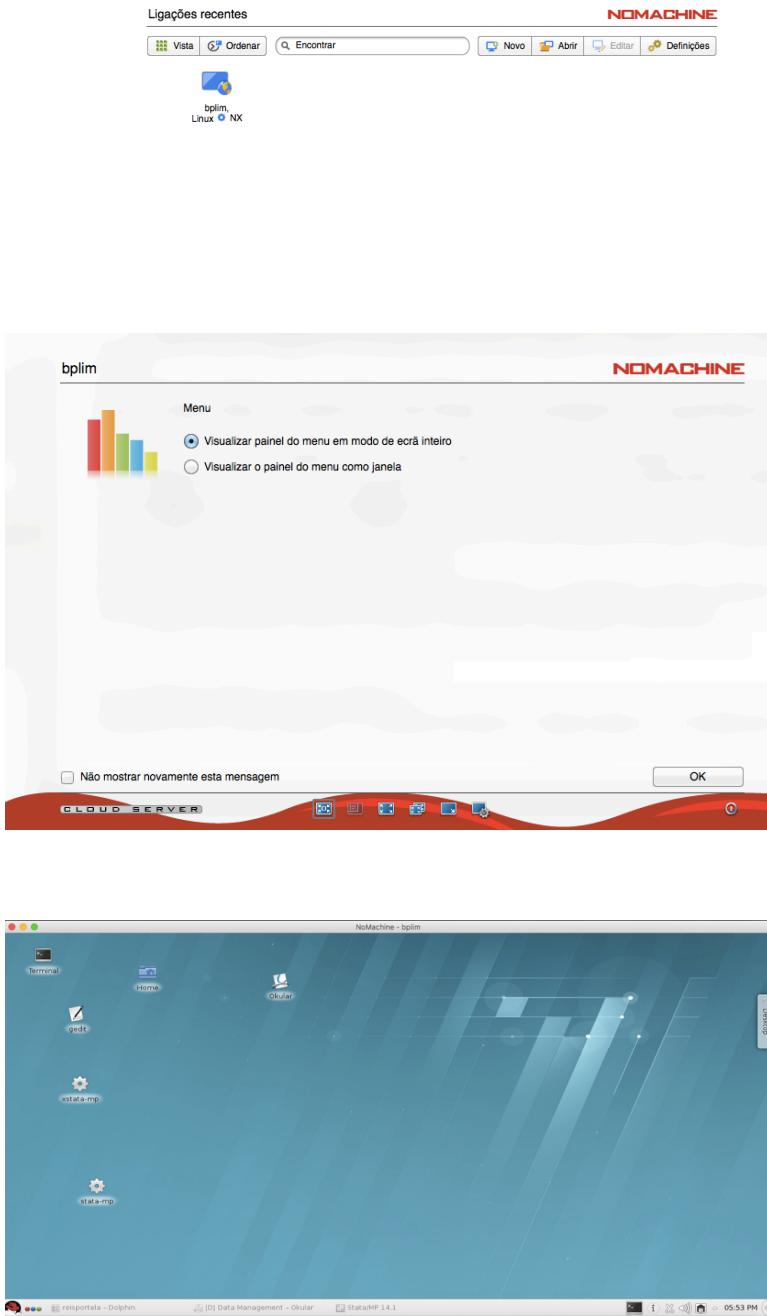
For complete password policy details, see [Appendix: Password Requirements](#)

## 1.4 First Steps

This section guides you through your initial login and introduces the basic interface and folder structure.

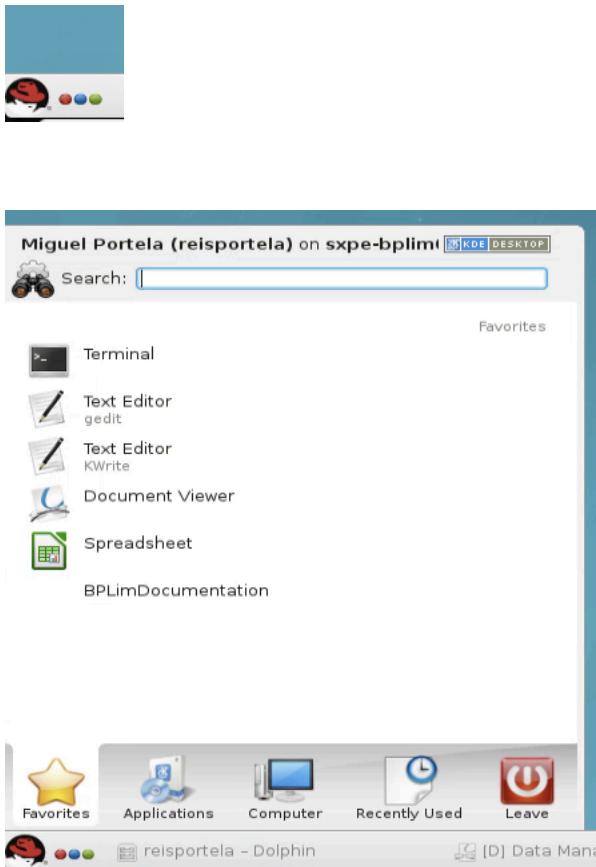
### 1.4.1 Logging In

1. When you start **NoMachine**, you will see the following connection screens:



### 1.4.2 Accessing Your Project

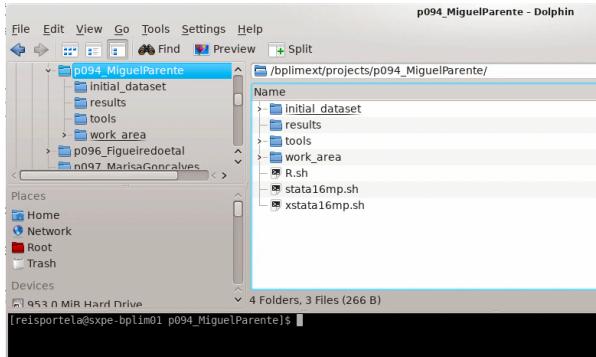
2. Once logged in, select the **Kickoff Application Launcher** menu:



3. Navigate to your project:

1. Click on **Applications**.
2. Select **BPLIM** and click on your project (e.g., P999\_research\_project).

This opens the **Dolphin** file manager<sup>1</sup>, showing your project folder:



You can display the Terminal (command line) alongside Dolphin by pressing **F4**.

---

<sup>1</sup>**Dolphin** is the file manager included with the KDE desktop environment. You can browse folders, create/delete files and folders (right-click for context menu), and manage your project files. More information: <https://userbase.kde.org/Dolphin>

### 1.4.3 Understanding Your Project Structure

4. **Launcher scripts:** Files with the .sh extension are scripts that launch applications or enter containerized environments. For example, `stata_container.sh` starts Stata.<sup>2</sup>

You can run launcher scripts in two ways:

- **GUI method:** Click the .sh file in Dolphin
- **Terminal method:** Type `./stata_container.sh` in the Terminal

5. **Project folders:** Your project folder contains the following directories:

Directory	Purpose	Access
<code>initial_dataset</code>	Data sources provided by BPLIM	Read-only
<code>— external</code>	Data provided by the researcher	Read-only
<code>— intermediate</code>	Intermediate files	Read-only
<code>— modified</code>	Modified data provided by BPLIM	Read-only
<code>results</code>	Output files generated by researchers	Read-write
<code>tools</code>	Project-specific analysis tools	Read-only
<code>work_area</code>	Temporary working directory	Read-write

**Note:** Your `work_area` folder also contains templates for Stata, R and/or Python, depending on your project requirements. By default, these template files are read-only.

### 1.4.4 Logging Out

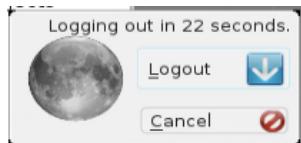
6. To properly disconnect, log out as shown below, then close the NoMachine window:<sup>3</sup>

<sup>2</sup>Containers are self-contained environments that include software and all its dependencies (libraries, configurations, packages). This ensures consistent behavior across sessions and enables reproducible research. Each container is isolated, preventing conflicts between different projects' software requirements.

<sup>3</sup>Click the cross button in the upper-right corner of the NoMachine window to close the connection.



Confirm by clicking **Logout**:<sup>4</sup>



### Important notes about session management:

- **Persistent sessions:** If you do not log out, your session remains open until your next login. While this keeps programs running, it consumes server resources.
- **Best practice for long-running tasks:** Use **batch mode** (see [Running Programs in Batch Mode](#)) instead of leaving sessions open.
- **Server maintenance:** During server reboots, open sessions are terminated and unsaved work is lost. Save your work regularly.

## 2 Important Guidelines

### 2.1 Managing Disk Space

Proper disk space management is essential for maintaining access to the server.

---

<sup>4</sup>Before logging out, ensure all active programs are closed (unless running in batch mode). Batch mode is recommended for computationally intensive or long-running tasks.

### **2.1.1 Critical Rule: Do Not Save Files in Your Home Folder**

Never save files in your home folder (/home/USER\_LOGIN). If you exceed its size limit, you will not be able to log in. Always save files in your project's work\_area folder.

### **2.1.2 Monitoring and Cleaning Your Project Folder**

Check your project size regularly to avoid exceeding storage limits. Follow these steps in the Terminal:

1. Navigate to your project folder:

```
cd /bplimext/projects/P999_research_project/
```

2. List the total project size:

```
du -h
```

3. Check folder sizes and list those larger than or equal to 1 GB:

```
du --max-depth=1 -h | sort -h | grep G
```

4. Move to the work\_area folder:

```
cd work_area
```

5. Repeat the size check in this folder:

```
du --max-depth=1 -h | sort -h | grep G
```

6. Identify duplicate or temporary files and remove them:

```
rm FILE_TO_DELETE
```

7. Compress large files or folders you are not currently using:

- Compress a folder:

```
tar -zcvf YOUR_FOLDER.tar.gz YOUR_FOLDER
```

- Compress an individual file:

```
gzip YOUR_FILE
```

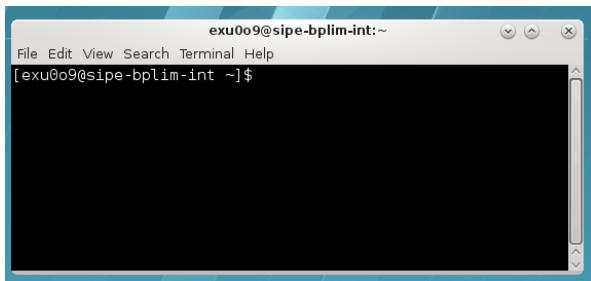
## **2.2 Using the Terminal**

The Terminal is a command-line interface for interacting with the Linux system. It is essential for running programs in batch mode, managing files, and monitoring processes.

### 2.2.1 Accessing the Terminal

You can access the Terminal in two ways:

- **From the menu:** Red Hat → Applications → System → Terminal



- **From Dolphin:** Press **F4** to open an integrated Terminal at the current location

### 2.2.2 Essential Terminal Tips

**Quick reference:**

1. **Case sensitivity:** Linux commands are case-sensitive (`ls` is not the same as `LS`)
2. **Command history:** Use arrow keys (up/down) to scroll through previous commands
3. **Auto-completion:** Press **Tab** to auto-complete file names and commands
4. **Keyboard layouts:** Non-English keyboards may have different symbol mappings. For example, on a Portuguese keyboard, `+` is on the `?` key
5. **Common commands:** See [Appendix: Shell Commands](#) for a comprehensive list

**Example command:**

```
ls -lArth
```

Lists files in human-readable format (`h`), long format (`l`), reverse order (`r`), sorted by modification time (`t`), including hidden files (`A`)

## 3 Working with Containers

All statistical software on the server runs inside **containers** – isolated, self-contained environments that ensure reproducibility and consistency.<sup>5</sup> Understanding how to work with containers is essential before using any statistical software.

### 3.1 What Are Containers?

A container is a self-contained environment that includes a program along with all its dependencies, libraries, and configurations. This ensures:

- **Consistency:** Programs behave identically across different sessions
- **Reproducibility:** Your analysis can be replicated exactly
- **Isolation:** Project-specific packages don't conflict with other projects

### 3.2 Starting Containers

There are three ways to start a container:

#### 3.2.1 Method 1: Using Launcher Scripts (Recommended)

Each project includes launcher scripts (.sh files) for different software:

- `stata_container.sh` - Launches Stata
- `r_container.sh` - Launches R/RStudio
- `python_container.sh` - Launches Python/Jupyter
- `julia_container.sh` - Launches Julia

**To use:** Click the script in Dolphin or run `./script_name.sh` in the Terminal.

#### 3.2.2 Method 2: From the Terminal

```
cd /bplimext/projects/P999_research_project  
singularity shell tools/_container/CONTAINER_ID.sif
```

After entering the container, the Terminal prompt changes to show `Singularity>`. You can then launch applications manually.

---

<sup>5</sup>Containers are self-contained environments that include software and all its dependencies (libraries, configurations, packages). This ensures consistent behavior across sessions and enables reproducible research. Each container is isolated, preventing conflicts between different projects' software requirements.

### 3.2.3 Method 3: Direct Execution

Run commands directly inside the container without entering an interactive shell:

```
singularity exec tools/_container/CONTAINER_ID.sif <command>
```

**When to use each method:**

- **Method 1:** Best for interactive work with graphical interfaces
- **Method 2:** Good for running multiple commands or programs within the same container session
- **Method 3:** Ideal for batch processing and automated scripts

## 4 Statistical Software

This section covers how to use Stata, R, Python, and Julia on the server. All software runs inside containers (see [Working with Containers](#) above).

### 4.1 Stata

#### 4.1.1 Starting Stata

Use the `stata_container.sh` launcher script in your project folder:

- **GUI method:** Click `stata_container.sh` in Dolphin
- **Terminal method:** Run `./stata_container.sh` from your project folder

**Manual container access:**

If you need to manually access the Stata container:

```
cd /bplimext/projects/P999_research_project  
singularity shell tools/_container/CONTAINER_ID.sif
```

Then launch Stata:

- **Graphical version:** `xstata-mp`
- **Command-line version:** `stata-mp`

#### 4.1.2 Ado-files

Ado-files are text files containing Stata programs. It is advisable to create and save your ado-files so results can be replicated later when running them on BPLIM datasets.

Stata looks for ado-files in several locations, typically organized as:

- **SITE** – system-wide ado-files
- **PLUS** – user-installed ado-files
- **PERSONAL** – user-created ado-files
- **OLDPLACE** – legacy location for ado-files

Stata always searches the current directory (.) and a set of predefined folders for ado-files. For BPLIM projects, ado-files provided by BPLIM are either built into the container or stored under /bplimext/projects/P999\_research\_project/tools. Ado-files that you create yourself should be saved in your project's work\_area (for example, in a dedicated ado/ subfolder)." Keep the adopath + "/bplimext/projects/P999\_research\_project/tools" code snippet immediately after this, as you already have.

To make sure Stata recognizes this directory, add the following line at the beginning of your .do file:

```
adopath + "/bplimext/projects/P999_research_project/tools"
```

The sysdir command within Stata will list all directories currently in use:

```
r; t=0.00 11:35:59
  sysdir
  SITE: /opt/bplimext/stata15/
  BASE: /opt/bplimext/stata15/ado/base/
  SITE: /opt/bplimext/ado/
  PLUS: ~/ado/plus/
  PERSONAL: /bplimext/projects/testeExterno/tools/
  OLDPLACE: ~/ado/
r; t=0.00 11:36:03
```

In addition, the adopath command lists all directories where Stata searches for ado-files, including any paths you add yourself (such as the project tools directory). Whereas sysdir shows the base system directories (SITE, PLUS, PERSONAL, OLDPLACE), adopath displays the full search path.

#### 4.1.3 Temporary Files

To manage Stata's temporary files:

1. Check the current temporary folder:

```
 tempfile junk
 display `^junk'"
```

2. It should display something like /tmp/St98278.000001
3. If the temporary file is not in the path /tmp, exit Stata and edit your .bashrc in your home directory (cd ~):

**Using Dolphin (GUI):** In Dolphin, enable Show Hidden Files (or press **Ctrl + H**), then locate .bashrc and open it with KWrite.

**Using the Terminal:** Run **kwrite ~/.bashrc** or **vi ~/.bashrc**. If you use **vi**, press **ESC**, type **:wq** and press **Enter** to save and exit (see Using the vi File Editor for more details)."

```
export STATATMP="/tmp"
```

1. Apply the changes:

```
source .bashrc
```

2. Start a new Stata session (inside the container).

#### 4.1.4 Running Stata in Batch Mode

Batch mode is the recommended method for long-running or computationally intensive programs.

For the full workflow, see [Running Programs in Batch Mode](#). In most cases your batch command will look like:

```
stata-mp do /bplimext/projects/P999_research_project/work_area/prog1.do
```

#### 4.1.5 Running Stata with Screen

To run Stata interactively (not in batch mode) while preserving your session in the Terminal if the connection drops, use `screen`. For long-running interactive work in the Terminal, using `screen` is strongly recommended. See [Using Screen for Persistent Sessions](#) for detailed instructions.

## 4.2 R

### 4.2.1 Starting R/RStudio

Use the `r_container.sh` launcher script in your project folder:

- **GUI method:** Click `r_container.sh` in Dolphin
- **Terminal method:** Run `./r_container.sh` from your project folder

This launches RStudio inside the container environment.

#### Manual container access:

If you need to manually access the R container:

```
cd /bplimext/projects/P999_research_project
singularity shell tools/_container/CONTAINER_ID.sif
rstudio
```

**IMPORTANT:** When exiting R, do **not** save your workspace image to your home folder. If you need to preserve your workspace, save it in your project's `work_area` folder.

### 4.2.2 Running R in Batch Mode

Batch mode is recommended for long-running R scripts. For the full workflow, see [Running Programs in Batch Mode](#). In most cases your batch command will look like:

```
Rscript /bplimext/projects/P999_research_project/work_area/analysis.R > analysis.log 2>&1
```

### 4.2.3 Running R with Screen

To run R interactively (not in batch mode) while preserving your session in the Terminal if the connection drops, use `screen`. For long-running interactive work in the Terminal, using `screen` is strongly recommended. See [Using Screen for Persistent Sessions](#) for detailed instructions.

## 4.3 Python

### Starting Python/Jupyter

Use the `python_container.sh` launcher script in your project folder:

- **GUI method:** Click `python_container.sh` in Dolphin
- **Terminal method:** Run `./python_container.sh` from your project folder

### Manual container access:

```
cd /bplimext/projects/P999_research_project  
singularity shell tools/_container/CONTAINER_ID.sif
```

Once inside the container, launch Jupyter Notebook:

```
jupyter notebook
```

This opens Jupyter in Firefox. Click **New** and select the **Python** kernel to create a notebook.

You can also use VSCode:

```
vscode
```

## 4.4 Julia

### Starting Julia

Use the `julia_container.sh` launcher script in your project folder:

- **GUI method:** Click `julia_container.sh` in Dolphin
- **Terminal method:** Run `./julia_container.sh` from your project folder

### Manual container access:

```
cd /bplimext/projects/P999_research_project  
singularity shell tools/_container/CONTAINER_ID.sif
```

Once inside the container, you can launch:

- **Julia REPL:** `julia`
- **Jupyter Notebook:** `jupyter notebook` (opens in Firefox; select the Julia kernel)
- **VSCODE:** `vscode` (opens the BPLIM-configured VSCode environment)

## 4.5 Structuring and Writing Code

**Goal:** keep code readable, reproducible, and easy to hand over.

### 4.5.1 Organize your project

- Work in `work_area`; folders `initial_dataset`, `external`, `intermediate`, `tools` and `modified` are read-only.
- Keep scripts in a dedicated folder (e.g., `work_area/scripts`), and store outputs in `results` (logs under `results/logs`).
- Do not save anything in your home folder.

### 4.5.2 Naming and layout

- Use short, descriptive names with underscores (e.g., `01_import.do`, `02_clean.R`, `03_analysis.py`).
- Use YYYYMMDD for dated files (e.g., `analysis_20250312.log`).
- Add a brief `README` in `work_area` explaining the script order and entry points.

### 4.5.3 Paths and reproducibility

- Prefer relative paths from the `work_area` instead of hard-coded absolute paths or home-relative paths.
- Set seeds for random routines (Stata: `set seed`, R: `set.seed()`, Python: `random.seed()/numpy.random.seed()`).
- Use the provided launcher scripts (`stata_container.sh`, `r_container.sh`, `python_container.sh`, `julia_container.sh`) to ensure you are inside the correct container.

### 4.5.4 Ado-files and packages

- Save your own ado-files in `work_area/ado/` (or similar) and add it to the search path, for example:

```
adopath + "/bplimext/projects/P999_research_project/work_area/ado"  
adopath + "/bplimext/projects/P999_research_project/tools"
```

- Packages or commands not already available must be requested from the BPLIM Team.

### 4.5.5 Batch and logging

- For long runs, use batch mode (see [Running Programs in Batch Mode](#)); redirect output to a dated log in `results/logs` (e.g., `stata-mp do ... > results/logs/run_20250312.log`).
- Keep batch scripts (e.g., `batch_run1`) alongside the code they execute, and reference them from the batch section.

#### 4.5.6 Editors and tools

- Use the provided wrappers (`vscode` for VSCode) or the software-specific editors inside each container.
- If editing via GUI, remember to show hidden files when needed (e.g., `.bashrc`); if using `vi`, see *Using the vi File Editor*.
- Track code with GitLab (see *Version Control with GitLab*) and commit regularly with clear messages.

#### 4.5.7 Data handling

- Never copy data to your home folder.
- Keep only essential intermediate files; clean temporary artifacts in `work_area` to manage space.
- Place final, non-sensitive outputs in `results` in line with the output extraction rules.

#### 4.5.8 Using the BPLIM templates

Your `work_area` folder contains template files for Stata, R and/or Python, depending on your project. These templates are pre-configured to follow the recommended project structure (reading from `initial_dataset`, writing to `results`, working under `work_area`).

- Copy a template to `work_area/scripts` (or a similar folder) and rename it (e.g., `01_import.do`, `01_import.R`, `01_import.py`).
- Adjust only the parts that are project-specific, such as replacing `P999_research_project` with your actual project ID.
- Keep all input and output paths inside your project folders (`initial_dataset`, `results`, `work_area`) and avoid using your home folder.

### 4.6 Updates to Commands and Packages

Requests for additional commands or packages, as well as updates to existing ones, must be submitted to the **BPLIM Team**.

### 4.7 Build a Container to Fine-Tune Your Statistical Packages

The server uses **Apptainer** (formerly **Singularity**) containers. To request one, please send the BPLIM Team the **definition file**. We will build the image and place it in your project's `tools`. Detailed information about Apptainer/Singularity containers is available at <https://sylabs.io/>.<sup>6</sup> Additional notes are provided in the Appendix.

---

<sup>6</sup>**Singularity** is now called **Apptainer**. Both names refer to the same container technology. Documentation: <https://apptainer.org>

## 5 Running Programs in Batch Mode

Batch mode is the **recommended method** for running long-running or computationally intensive programs. Instead of keeping an interactive session open, batch mode allows you to submit jobs that run in the background.

### 5.1 Why Use Batch Mode?

- **Efficiency:** Frees up your interactive session
- **Reliability:** Programs continue running even if you disconnect
- **Resource management:** Better server performance for all users
- **Recommended for:** Any program that runs longer than 30 minutes

### 5.2 Basic Batch Workflow

1. Navigate to your working folder:

```
cd /bplimext/projects/P999_research_project/work_area/
```

2. Prepare your analysis script and batch file:

- First, create or reuse a Stata do-file in your `work_area`, for example `prog1.do`.
- Then create a batch script file (plain text), for example `batch_prog1`, containing the command to execute your do-file.

Example for Stata (`batch_prog1`):

```
stata-mp do /bplimext/projects/P999_research_project/work_area/prog1.do
```

Example for R (`batch_r_analysis`):

```
Rscript /bplimext/projects/P999_research_project/work_area/analysis.R /  
analysis.log 2>&1
```

3. Enter the container environment:

```
singularity shell ../tools/_container/CONTAINER_ID.sif
```

4. Submit the batch job:

```
at now -f batch_prog1
```

### 5.3 Scheduling Jobs

The `at` command allows you to schedule jobs:

- **Run immediately:** `at now -f batch_script`
- **Run in 5 hours:** `at now + 5 hours -f batch_script`
- **Run in 30 minutes:** `at now + 30 minutes -f batch_script`

For more options, type `man at` in the Terminal.

## 5.4 Managing Batch Jobs

**View queued/running jobs:**

```
atq
```

- = indicates the job is currently running
- a indicates the job is queued with its scheduled time

**Remove a job from the queue:**

```
atrm <job_number>
```

## 5.5 Monitoring Running Programs

### 5.5.1 Using top

```
top
```

- Press i to hide background processes
- Press k to kill a process (enter PID, then type 9 to force termination)
- Press q to exit

### 5.5.2 Using tail to monitor log files

```
tail -f logfile.log
```

This continuously displays new lines as they are written. Press CTRL + C to stop monitoring.

## 6 Using Screen for Persistent Sessions

screen is a session manager that allows you to run programs interactively while preserving your session if your network connection drops.

### 6.1 Basic Screen Usage

**Start a new screen session:**

```
screen -S my_session_name
```

**Detach from a screen session (keeps it running):**

Press CTRL + A, then D

**List all running screen sessions:**

```
screen -ls
```

**Reattach to a screen session:**

```
screen -r my_session_name
```

Or, if only one session exists:

```
screen -r
```

**Reattach to a specific session by PID:**

```
screen -r <pid>
```

## 6.2 Example: Running Stata with Screen

```
screen -S stata_session
singularity shell tools/_container/CONTAINER_ID.sif
stata-mp
```

## 6.3 Example: Running R with Screen

```
screen -S r_session
singularity shell tools/_container/CONTAINER_ID.sif
```

To detach from the session while keeping R running, press **CTRL + A**, then **D**. To **return later**, run:`screen -r stata_session``

## 6.4 When to Use Screen vs Batch Mode

- **Use screen:** For interactive work where you need to see results immediately and may want to modify your approach
- **Use batch mode:** For fully scripted analyses that don't require interaction

# 7 Allowed Outputs

Results can be exported to disk in the following formats (see the *Output Control Guide*):

1. **ASCII files** — e.g., log files
2. **Graphs** — export as .png
3. **CSV** — Comma-Separated Values, for use with MS Excel or similar
4. **TEX** — LaTeX format for integration into TeX documents

### Visualizing LaTeX tables

If you want to preview a table exported to LaTeX as a PDF, create a simple file named `main.tex`:

```
\documentclass{article}
\begin{document}
\input{your_table.tex}
\end{document}
```

Replace `your_table.tex` with the name of your table file. Compile it in the Terminal with:

```
pdflatex main.tex
```

This generates `main.pdf`, which you can view with:

```
okular main.pdf
```

## 8 Requesting Outputs

All output files (log files, tables, graphs) must be requested from the **BPLIM Team** at [bplim@bportugal.pt](mailto:bplim@bportugal.pt). Researchers cannot independently extract files from the server. All outputs must comply with the [output control rules](#).

After validation, approved results will be sent to you by email. The extraction process depends on your project type:

### 8.1 Projects Using Modified Data

If your project uses modified data provided by BPLIM:

1. **Run the replication app** successfully before requesting outputs. See the [Replication App manual](#) for instructions.
2. **Send an email** to [bplim@bportugal.pt](mailto:bplim@bportugal.pt) with the subject line:  
**Subject:** P999\_research\_project: request replication  
Replace P999\_research\_project with your actual project ID.

### 8.2 Projects NOT Using Modified Data

If your project does not use modified data:

1. **Place all outputs** in the `results` folder within your project.<sup>7</sup>
2. **Send an email** to [bplim@bportugal.pt](mailto:bplim@bportugal.pt) with the subject line:  
**Subject:** P999\_research\_project: request for result extraction  
Replace P999\_research\_project with your actual project ID.

---

<sup>7</sup>**Output extraction rules:** Only non-sensitive text files without identifiable information can be extracted. For each graph requested, you must provide the corresponding data table for replication. Graphs must be in PNG format; vector graphics are not permitted.

## 9 Managing Your Home Folder

Your home folder (`/home/USER_ID/`) has a strict size limit. Exceeding this limit will prevent you from logging in.

### Critical guidelines:

1. **Never save work files in your home folder.** Always use your project's `work_area` folder.
2. **Regularly empty your Trash folder.** To clean the Trash via Terminal:

```
rm -rf ~/.local/share/Trash/*
```

3. **Keep your home folder minimal.** Only configuration files and small settings should be stored there.

If you cannot log in due to disk quota issues, contact the BPLIM Team for assistance.

## 10 Project Archival Policy

Projects that remain **inactive for more than 2 years** will be archived automatically. A project is considered **inactive** if there are no logins to the external server, or no access to folders or files, under that project **and** no contact with the BPLIM Team about that project during this period.

### What this means:

- Archived projects are no longer directly accessible on the server
- All project data is preserved and can be restored
- To reactivate an archived project, contact the **BPLIM Team** at [bplim@bportugal.pt](mailto:bplim@bportugal.pt)

## 11 Appendix

### 11.1 Basic Shell Commands on Linux

- **top:** List processes currently running on the server
  - Press **i** to hide background processes.
  - Press **h** to display the **help menu** for available options.
- **pwd:** Show the current working directory
- **cd:** Change directory

```
cd /bplimext/projects/PXXX_name/work_area/
```
- **cd ~:** Move to your home folder
- **cp:** Copy file(s) to a given path

```
cp prog1.do /bplimext/projects/PXXX_name/results
```

- **mv**: Move file(s) or rename file(s)

```
mv prog1.do /bplimext/projects/PXXX_name/results
```

- **rm**: Delete a file

```
rm /bplimext/projects/PXXX_name/results/prog1.do
```

- **mkdir**: Create a directory

```
mkdir programs
```

- **rmdir**: Delete an empty directory

```
rmdir programs
```

- **screen**: Start a session manager that allows running programs in the background and resuming them later

```
screen top
```

- **man**: Show the manual page for a given command

```
man ls
```

- **du -h**: Display disk usage of files and directories in human-readable format

```
du /bplimext/projects/PXXX_name/work_area/
```

- **df -h**: Show disk space utilization in human-readable format

- **vi**: View or edit ASCII text files (e.g., .do files, logs)

- **ghostscript**: Preview files with .eps or .pdf extensions

```
ghostscript /bplimext/projects/PXXX_name/results/file_name.pdf
```

- **okular**: View PDF files

- **find**: Search for files

– Basic structure: `find /path options pattern`

```
find . -name "*.do"
```

– Save search results to a file:

```
find . -name "*.do" > find_results.txt
```

– Search for a string within filenames:

```
find . -name "*.do" | grep "analysis"
```

– Identify .do files containing the word `graph export`:

```
find . -name "*.do" -exec grep "graph export" '{}' \; -print
```

- **passwd**: Change your password

- **Exit a program**: Press **CTRL + C** to terminate the current process in the shell

## 11.2 Using the vi File Editor

1. Open a file in `vi` from the shell, for example:

```
vi batch1.txt
```

2. Common shortcut keys in `vi`:

- `i`: Insert text
- `ESC`: Exit insert mode
- `x`: Delete the character under the cursor
- `dd`: Delete the current line
- `10 dd`: Delete 10 lines
- `yy`: Copy (yank) the current line
- `p`: Paste the copied (yanked) text
- `SHIFT + G`: Go to the last line
- `gg`: Go to the first line
- `ESC + :q!`: Quit without saving changes
- `ESC + :w!`: Write (save) and overwrite the file
- `ESC + :q`: Quit if no changes have been made

For a more complete guide, see: <https://www.cs.colostate.edu/helpdocs/vi.html>

3. Easier alternative: use the `gedit` text editor for a graphical interface:

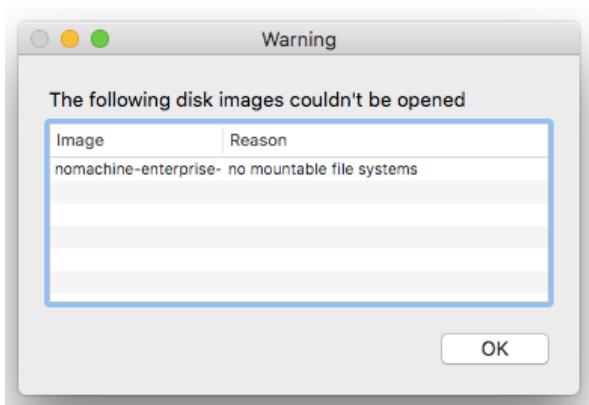
```
gedit batch1.txt
```

### 11.3 Password Requirements

Rule	Value	Notes
Max. Password Life-time	60 days	After 60 days the password will expire and must be changed at the next login. The password can be changed at any time using: (1) Red Hat icon → Applications → Settings → System Settings → Account Details, click <i>Change Password</i> ; or (2) in the Terminal type <code>passwd</code> .
Min. Character Classes	4	You should include at least 4 classes of characters in the password. For example: small letters, capital letters, numbers and punctuation marks.
		There are a total of five classes: <b>A–Z</b> (capitals); <b>a–z</b> (lowercase); <b>0–9</b> (numbers); <b>punctuation:</b> <code>! " # \$ ' - / \ ^ _ `   ;</code> <b>chars above 127:</b> ( <code>á, á, à, à, etc.; @, £, §, º, ª, «, »</code> ).
		<b>Note:</b> Using the same character 3+ times may require an additional class. Recommended: don't repeat the same character more than twice consecutively.
Min. Length	8	The minimum size of the password is 8 characters (it may be higher if you repeat characters).
Password History	7	You cannot reuse a password from the previous 7 passwords.
Max. Failures	6	If the user fails 6 consecutive times, the account will be locked for the time defined in <i>Lockout Time</i> .
Fail Interval	60 sec	Time interval to count attempts as consecutive. If more than 60 seconds have elapsed since the last attempt, the failure count resets to 1.
Lockout Time	600 sec	Time (10 minutes) during which the account will be locked if the maximum number of failed attempts is reached.

### 11.4 NoMachine: Frequently Asked Questions

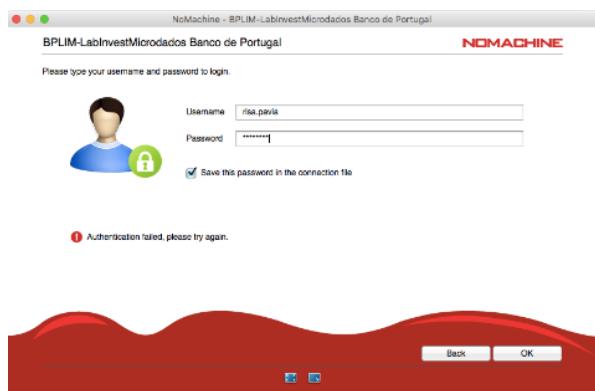
1. Mac users cannot install NoMachine and receive the error below:



- Ensure your **macOS** is up to date.
- As a temporary solution, download the **NoMachine Enterprise Client** from the official website and run the installation file.

### [NoMachine Client](#)

## 2. NoMachine authentication failure



- This may happen due to a mismatched keyboard layout.  
For example, if you use a **Portuguese keyboard** but the system assumes a **US keyboard**, a password containing **ç** may be rejected as “wrong password.”  
Verify your keyboard layout or change your password after the first login using:

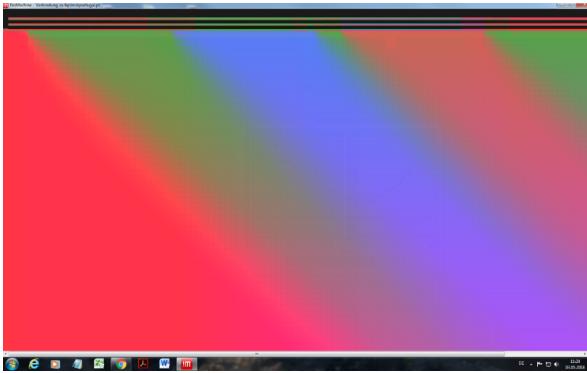
[passwd](#)

- If login fails with the error:  
*“Could not connect to the server. Error is 138: Connection is timed out”*  
check whether your network firewall is blocking the connection.  
Some university networks block external connections to BPLIM’s server.  
Test from another location (e.g., your home network).

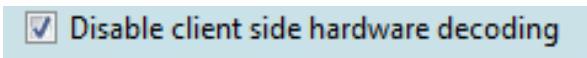
## 3. User pressed ‘Lock’ instead of ‘Log out’ and cannot unlock

- Check that the keyboard layout is correct (e.g., PT or UK).
- Close the NoMachine session and start a new one. Before the final **Login** step, right-click and choose **Logout**, then click to reconnect.

## 1. “Cannot see the screen in NoMachine”

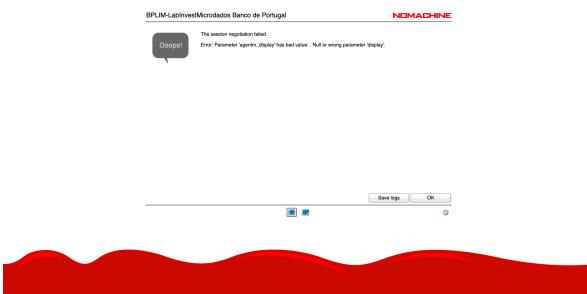


- **Option A:** Move your mouse to the top-right corner of NoMachine. A folded-sheet icon will appear. Left-click → **Display** → **Change settings** → enable **Disable client-side hardware decoding**.



- **Option B:** Close the NoMachine connection and start a new one. Before the final **Login** step, right-click and choose **Logout**, then click to reconnect.

## 5. “Error: Parameter ‘agentm\_display’ has bad value”



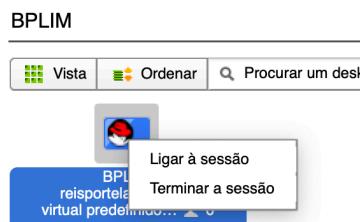
- This usually means your home folder is full (/home/USER\_LOGIN). **Do not save files in your home folder.**
- Ask the BPLIM Team to free up space in your home directory.

## 6. Session is frozen

- From the first NoMachine screen, click the following icon:



- Then right-click the icon below and choose **Terminar sessão**:

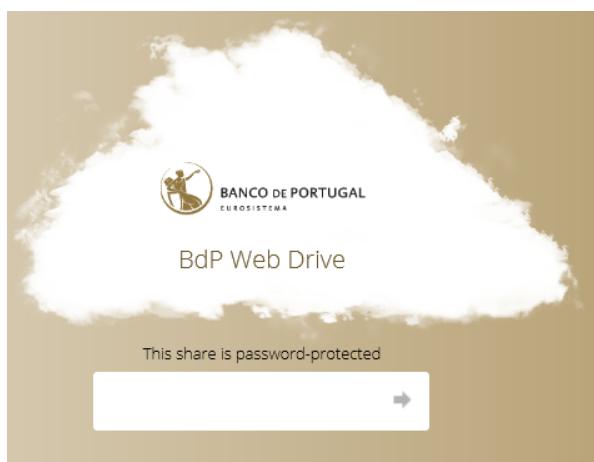


## 11.5 Download, Install and Configure NoMachine Client

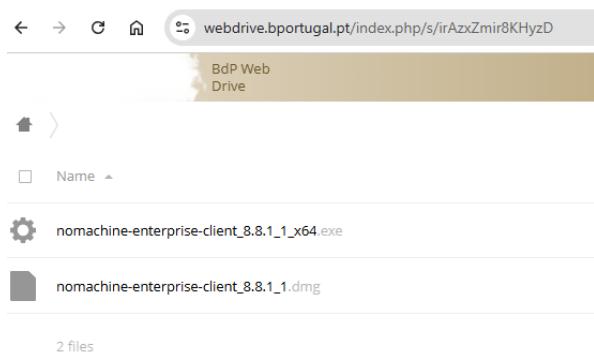
**Step 1:** Go to the following link and use the credentials provided by BPLIM to access the site:

[Banco de Portugal Webdrive](#)

**Note:** sometimes the internet provider, *e.g.*, a University, may block access to this particular website. Please check with your provider in case you get an error while trying to use the link.

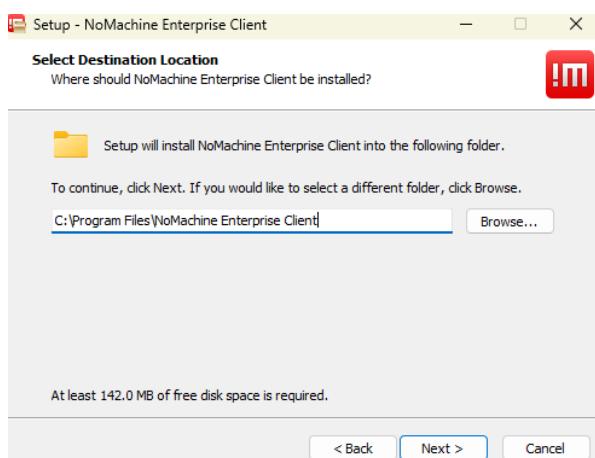
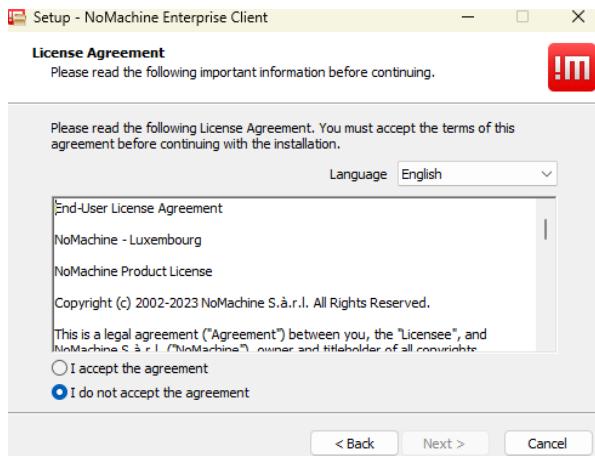


**Step 2:** Download the file with an extension compatible with your OS (Operating System).

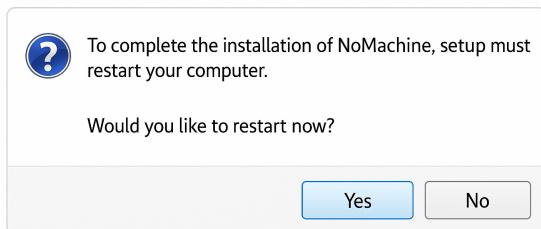


**Step 3:** Install 'NoMachine'.



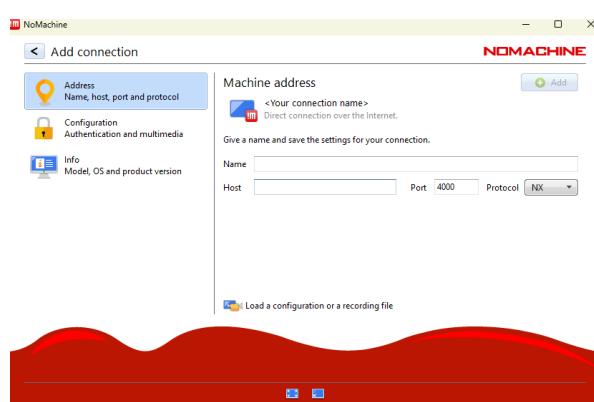
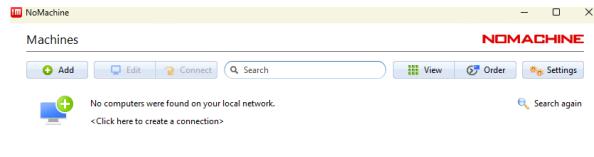


**Step 4:** Reboot your computer

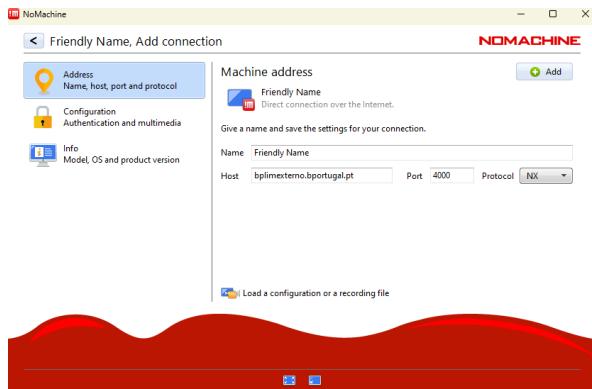


**Step 5:** NoMachine client access configuration.

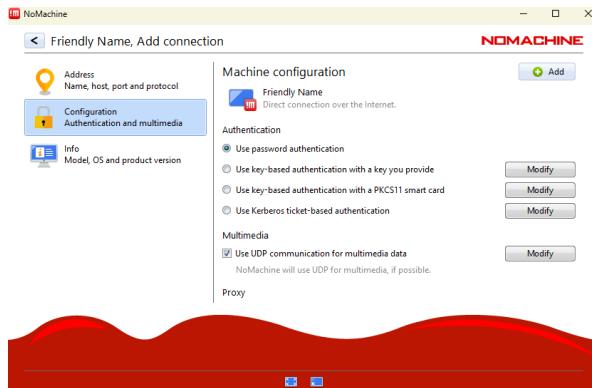
**Step 5.1:** Start 'NoMachine' and create a new connection.



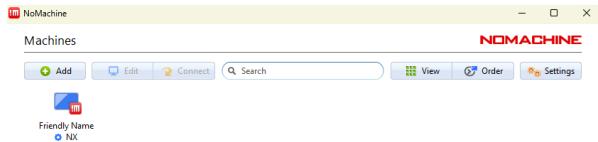
**Step 5.2:** Define the ‘Host’ as bplimexterno.bportugal.pt, ‘Port’ 4000, ‘Protocol’ NX and set a ‘Friendly Name’ for ‘Name’.



**Step 5.3:** Use password authentication – with or without a proxy – according to the instructions provided by your network administrator or IT support. The proxy settings can be customized under Proxy in the bottom-right corner. After completing the configurations, click Add to create the connection.



**Step 5.4:** Once the entry for bplimexterno.bportugal.pt has been created, connect:

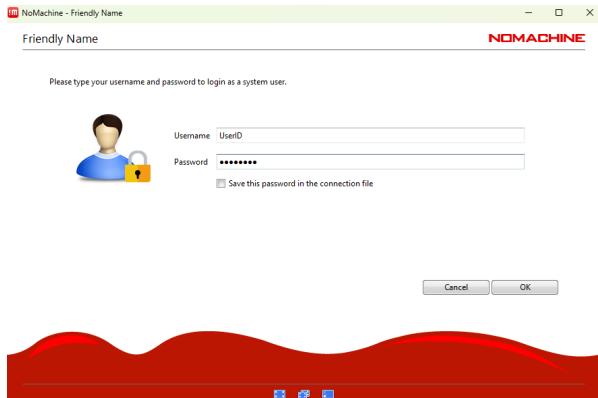


**Step 5.5:** Before the first effective connection, it may be necessary to accept the certificate from bplimexterno.bportugal.pt. You should verify that the "fingerprint" (verification code) is:

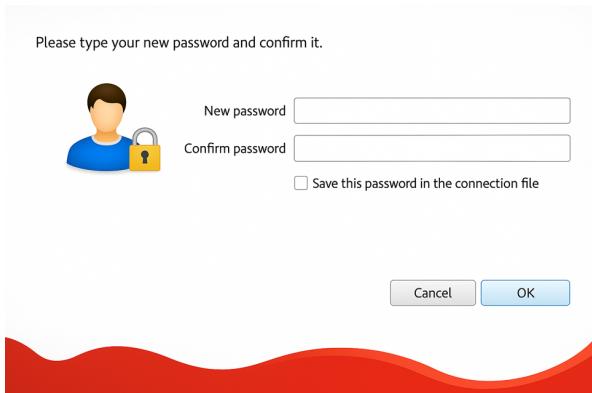
**SHA256 ED 1B D9 E2 C2 F8 C6 08 1A 53 5F 97 DA 71 77 D9 D2 EE  
7A 5F 9C 35 87 B3 19 F4 7E A1 CB 2C 68 0B**



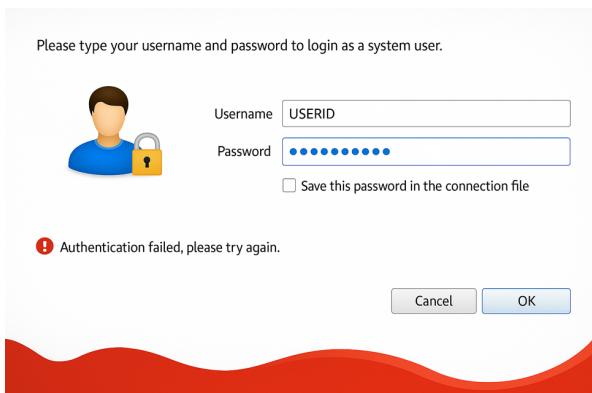
**Step 5.6:** Connect with the UserID (**case sensitive**) and password provided by Banco de Portugal:



**Step 5.7:** After the first successful login, it is necessary to change the password, which must comply with the Password Policy defined Section 1.2.

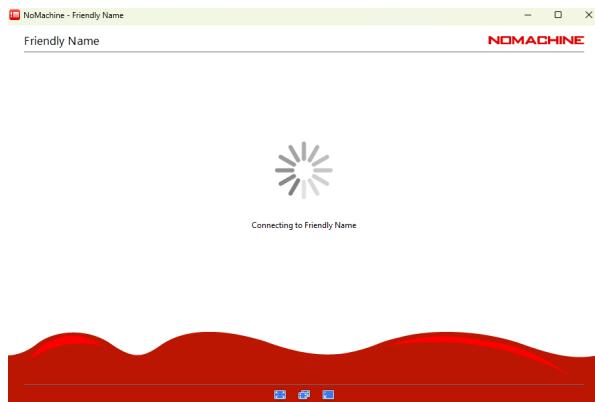


If the new password does not comply with the Password Policy, the original password provided by the Banco de Portugal will be re-requested. You get the message “Authentication failed, please try again.” See Appendix 3 for details.

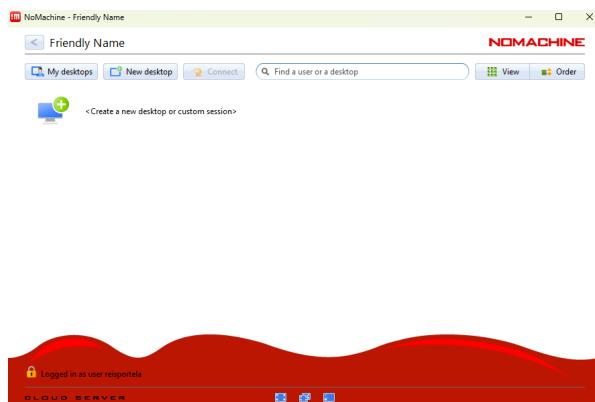


The NoMachine client does not tell you why the new password was not accepted – it is the responsibility of the user to verify that the new password is in compliance.

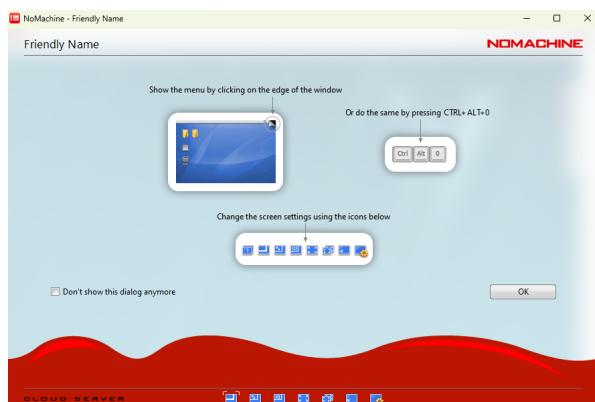
**Step 5.8:** Upon login success, the following screens should appear.

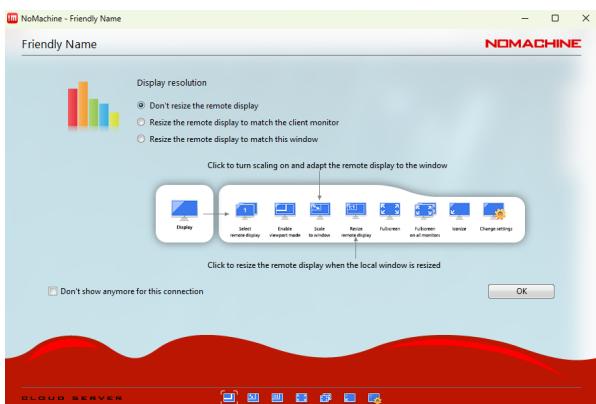
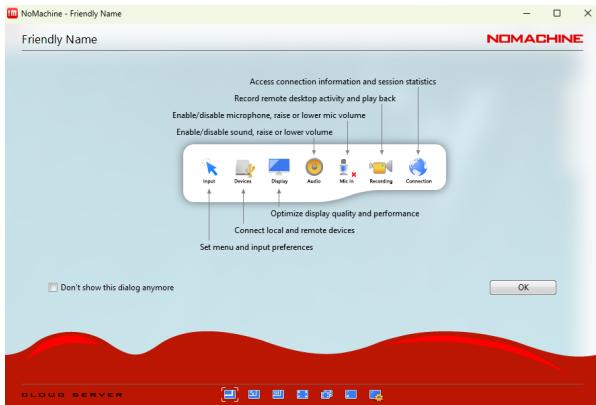


Create a new desktop.



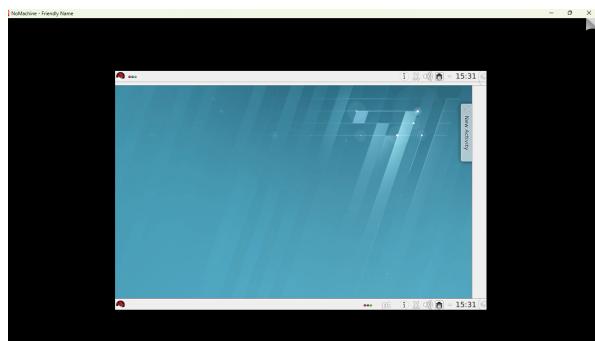
**Step 5.9:** In the following screen define the settings of your monitor.





**Step 5.10:** Upon login success, the following screens should appear.

Once logged in and with access to a KDE session, click on the upper right corner of the KDE desktop, as shown below, to access the menu and then expand the screen as exemplified for greater ease of use.



**Step 5.11:** You should see the following screen.



**Step 5.12:** Click 'Display'.



**Step 5.13:** Choose the option that best fits your monitor.



**Tip:** At any time during a NoMachine session you can press **Ctrl + Alt + 0** to open the NoMachine menu. From there, select **Display → Change settings** to adjust the resolution or scaling mode (for example, Fit to window).

## 11.6 Version Control with GitLab

The server provides [GitLab](#) for version control. Git is a distributed version-control system for tracking changes in files, ideal for managing code and scripts across your research project.

**To request Git access:** Contact the BPLIM Team at [bplim@bportugal.pt](mailto:bplim@bportugal.pt).

### 11.6.1 Benefits of Using Version Control

- Track all changes to your code and scripts
- Revert to previous versions if needed
- Collaborate with team members
- Maintain a complete history of your research workflow

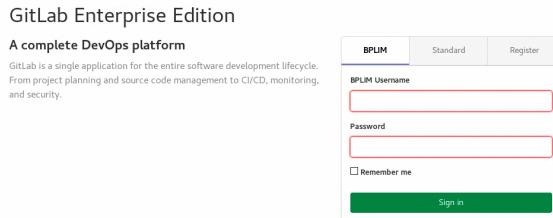
### 11.6.2 Getting Started with Git

**11.6.2.1 Generate an SSH Key** Open a Terminal in your home folder and generate an SSH key:

```
cd ~  
ssh-keygen -t rsa -C "BPLIM git"  
cat ~/.ssh/id_rsa.pub
```

Highlight the generated key, right-click, and select **Copy** to copy it to your clipboard.

**11.6.2.2 Access GitLab** Open Firefox (Red Hat → Search → Firefox) and navigate to:  
<https://vxpp-bplimgit.bplim.local/>

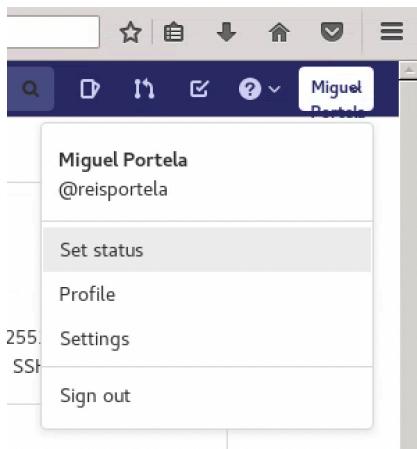


The image shows the GitLab Enterprise Edition login page. At the top, it says "GitLab Enterprise Edition" and "A complete DevOps platform". Below that, there's a brief description: "GitLab is a single application for the entire software development lifecycle. From project planning and source code management to CI/CD, monitoring, and security." On the right side, there's a login form with fields for "BPLIM Username" and "Password", a "Remember me" checkbox, and a green "Sign in" button. There are also links for "Standard" and "Register".

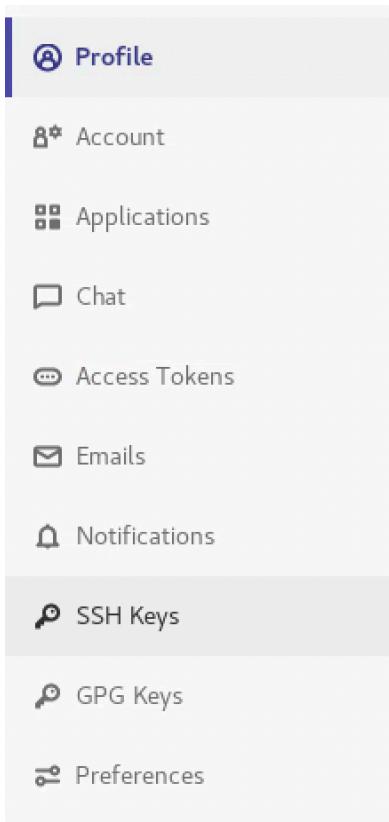
Log in with your external server credentials.

### 11.6.2.3 Add Your SSH Key in GitLab

- Navigate to your profile by clicking **Settings** in the top-right corner



- In the left sidebar, click **SSH Keys**



- Paste your SSH key in the **Key** text box

User Settings > **SSH Keys**

**SSH Keys**  
SSH keys allow you to establish a secure connection between your computer and GitHub.

**Add an SSH key**  
To add an SSH key you need to [generate one](#) or use an [existing key](#).

**Key**  
Paste your public SSH key, which is usually contained in the file `~/.ssh/id_ed25519.pub` or `~/.ssh/id_rsa.pub` and begins with `ssh-ed25519` or `ssh-rsa`. Don't use your private SSH key.

Typically starts with "ssh-ed25519 ..." or "ssh-rsa ...".

**Title**  **Expires at**

Give your individual key a title. This will be publicly visible.

**Add key**

- Enter a title (e.g., “BPLIM git”) and click **Add key**

**11.6.2.4 Create a New GitLab Project** Go to Projects → New project and create a repository (e.g., scripts\_P999).

The top half of the image shows a screenshot of a web browser window for 'SSH Keys - User Set...' at 'https://vxpp-bplimgit.bplim.local/profile/keys'. The page title is 'SSH Keys' under 'User Settings'. Below it, a section titled 'SSH Keys' explains their purpose: 'SSH keys allow you to easily connect between you'. The bottom half of the image shows a screenshot of the KWrite text editor. The title bar says 'New project'. The main area contains a configuration file with the following content:

```
[cola]
spellcheck = false
[user]
name = Your Name
email = your_username@sxpe-bplim01.bplim.local
[gui]
editor = kwrite
```

**11.6.2.5 Configure Git** Create or edit the `.gitconfig` file in your home folder. Use KWrite (Red Hat → Search → KWrite):

```
[cola]
spellcheck = false
[user]
name = Your Name
email = your_username@sxpe-bplim01.bplim.local
[gui]
editor = kwrite
```

**11.6.2.6 Clone Your Project** In the Terminal, navigate to your `work_area` and clone the repository:

```
cd /bplimext/projects/P999_research_project/work_area/
git clone git@vxpp-bplimgit.bplim.local:username/scripts_P999.git
```

**11.6.2.7 Add `.gitignore` File** Copy the `.gitignore` template from your project's `tools` folder:

```
cd scripts_P999
cp /bplimext/projects/P999_research_project/tools/.gitignore .
```

### 11.6.2.8 Make Your First Commit

```
git add *
git commit -a -m "Initial commit"
git push
```

### 11.6.2.9 Best Practices

- Store all your scripts and code in the Git repository folder (e.g., `scripts_P999`)
- Commit changes regularly with descriptive messages
- Pull before you push to avoid conflicts
- Use branches for experimental work

## 11.7 Building Custom Containers

If you need custom software packages or specific versions, you can request a custom container.

### 11.7.1 Steps to Build a Custom Container

**11.7.1.1 Create a Container Definition** Use the template files available in the [BPLIM Containers GitHub repository](#).

### 11.7.1.2 Test and Build Using Sylabs Cloud

- Sign in to [Sylabs Cloud](#) (use your GitHub account)
- Click **CREATE**:



- Upload your `.def` file or paste its contents:

A screenshot of the Sylabs Remote Builder interface. At the top left is the Sylabs.io logo. The main header says 'Remote Builder'. Below the header, there is a brief description of Singularity's ability to build containers in the cloud. A large input area is labeled 'Build a Container' with the sub-instruction 'Use the editor below to provide a definition file to build. You can drag &amp; drop definition files onto the editor, or use the text area below.' There is a 'Upload a definition file' button and a text input field with some placeholder text. At the bottom of the page, there is a navigation bar with links to Home, Singularity Library, Remote Builder, and Keystore.

- Sylabs validates your script. Once successful, click **Build**
- Monitor the build process for any errors
- After successful build, send the **definition** file to the BPLIM Team

**11.7.1.3 Using Your Custom Container** Once the BPLIM Team builds your container, it will be placed in your project's `tools/_container` folder.

To use it:

```
cd /bplimext/projects/P999_research_project/tools/_container
singularity shell YOUR_CONTAINER_ID.sif
```

The Terminal prompt changes to `Singularity>`, indicating you're inside the container. You now have access to your custom software environment.



A screenshot of a terminal window. The prompt is "Singularity>". The command "rstudio" is typed into the input field.

Launch applications as needed (e.g., `rstudio` for RStudio).

## 11.8 Jupyter Lab

[JupyterLab](#) is a web-based interactive development environment for notebooks, code, and data. It provides a flexible interface for data science, scientific computing, and machine learning workflows.

### 11.8.1 Starting JupyterLab

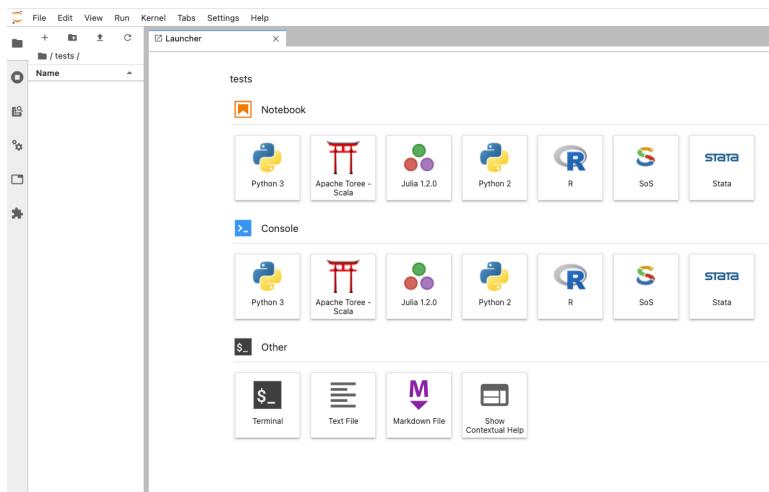
From within a container (Python or Julia), run:

```
jupyter lab --browser=firefox
```

This opens JupyterLab in Firefox, providing an integrated environment for:

- Interactive notebooks (Python, Julia, R)
- Code editing and execution
- Data visualization
- Terminal access

## 11.8.2 Example JupyterLab Session



**Tip:** JupyterLab is ideal for exploratory data analysis and prototyping. For production scripts, consider using dedicated .py, .R, or .jl files.