

# cyTRON: Installation Guide

---

The aim of this guide is to support the user to install the software, get the R libraries, and setup the environmental variables required for the correct execution of cyTRON. This is not intended as an exhaustive step-by-step guide; all the instructions have been tested on Linux Ubuntu 18.04.1 64 bit, macOS High Sierra 10.13.1 64 bit and Microsoft Windows 10 64 bit on October 13rd, 2018.

## Install Required Software

cyTRON requires the download and installation of the following software for its execution:

- Java SE Development Kit (in this guide we use jre 8)
- Cytoscape (in this guide we used Cytoscape 3.6.1)
- R (in this guide we used R 3.5.1)

Please refer to the respective Web resources to download the proper version of the tools and install them before the next steps.

Also, please note that cyTRON requires the R TRONCO library: check the minimum R version compatible with TRONCO. The command Rscript that comes with R needs to be correctly configured and runnable without administration privileges on the machine.

## Setup on Linux Ubuntu 18.04.1 64 bit

### R Setup

Before setting up R directly, open a Terminal session and run the following commands:

- `sudo apt-get update`
- `sudo apt-get install libcurl4-openssl-dev libssl-dev`

Install R and open an R session. Run the following commands:

- `update.packages(ask = FALSE)`
- `install.packages("rJava", dependencies=TRUE)`
- `install.packages("devtools", dependencies=TRUE)`
- `library("devtools")`
- `install_github("BIMIB-DISCo/TRONCO")`
- `system.file("jri", package="rJava")`

- Sys.getenv("R\_HOME")

At this point, do not close the R session as we will get back to it later to copy the displayed paths.

```
> system.file("jri", package="rJava")
[1] "/home/cytron2018/R/x86_64-pc-linux-gnu-library/3.5/rJava/jri"
> Sys.getenv("R_HOME")
[1] "/usr/lib/R"
>
```

## Environment Setup

The first step aims at making JRI available for JVMs as follow:

- open a Terminal session
- go to the folder previously returned by the R command `system.file("jri", package="rJava")`
- execute the command `cp libjri.so /usr/lib`

Note that the `cp` command may require `sudo` privileges to run. Also, in the case the directory `"/usr/lib"` does not represent a valid Java Library Path, you will have to copy the file into a valid one.

As a last step, we now need to set the `"R_HOME"` environmental variables:

- open a Terminal session
- run the command `nano /etc/environment`
- add the variable `R_HOME` with the value obtained by the R command `Sys.getenv("R_HOME")`. Here is an example of line to be included in the file: `R_HOME="/usr/lib/R"`
- reboot the system

At this point cyTRON can be installed and executed.

## Setup on macOS High Sierra 10.13.1 64 bit

### R Setup

Install R and open an R session. Run the following commands:

- `update.packages(ask = FALSE)`
- `install.packages("rJava", dependencies=TRUE)`
- `install.packages("devtools", dependencies=TRUE)`
- `library("devtools")`
- `install_github("BIMIB-DISCO/TRONCO")`
- `system.file("jri", package="rJava")`
- `Sys.getenv("R_HOME")`

We note that for some configurations of OS X 32/64 bits, rJava as for its defaults fails to load. A comprehensive guide on how to fix this problem is out of the scope of this guide; thus we refer to available online resources.

At this point, do not close the R session as we will get back to it later to copy the displayed paths.

```
> system.file('jri',package='rJava')
[1] "/Library/Frameworks/R.framework/Versions/3.5/Resources/library/rJava/jri"
> Sys.getenv('R_HOME')
[1] "/Library/Frameworks/R.framework/Resources"
```

## Environment Setup

The first step aims at making JRI available for JVMs as follow:

- open a Terminal session
- go to the folder previously returned by the R command `system.file("jri", package="rJava")`
- execute the command `cp libjri.jnilib /Library/Java/Extensions`

Note that the `cp` command may require `sudo` privileges to run. Also, in the case the directory `"/Library/Java/Extensions"` does not represent a valid Java Library Path, you will have to copy the file into a valid one.

As a last step, we now need to set the `"R_HOME"` environmental variables:

- open a Terminal session
- run the command `nano ~/.bash_profile`
- export the variable `R_HOME` with the value obtained by the R command `Sys.getenv("R_HOME")`. Here is an example of line to be included: `export R_HOME="/Library/Frameworks/R.framework/Resources"`
- reboot the system

At this point `cyTRON` can be installed and executed.

## Setup on Microsoft Windows 10 64 bit

### R Setup

Install R and open an R session. Run the following commands:

- `update.packages(ask = FALSE)`
- `install.packages("rJava", dependencies=TRUE)`
- `install.packages("devtools", dependencies=TRUE)`
- `library("devtools")`
- `install_github("BIMIB-DISCo/TRONCO")`

- `system.file("jri", package="rJava")`
- `Sys.getenv("R_HOME")`
- `.libPaths()`

At this point, do not close the R session as we will get back to it later to copy the displayed paths.

```
> system.file('jri', package='rJava')
[1] "C:/Users/cyTRON2018/Documents/R/win-library/3.5/rJava/jri"
> Sys.getenv('R_HOME')
[1] "C:/PROGRA~1/R/R-35~1.1"
> .libPaths()
[1] "C:/Users/cyTRON2018/Documents/R/win-library/3.5" "C:/Program Files/R/R-3.5.1/library"
> |
```

## Environment Setup

The first step aims at making JRI available for JVMs as follow:

- go to the folder previously returned by the R command `system.file("jri", package="rJava")`
- locate and copy the `jri.dll` file to the folder `C:\Windows\System32`

Note that if you are running an x64 OS, you have to copy the `jri.dll` file inside the x64 sub-folder located within the folder returned by the R command `system.file("jri", package="rJava")`.

The second step of this guide aims at exporting R to the JVMs:

- go to the installation folder of R and proceed to the `bin` sub-folder
- choosed the sub-folder `x64` or `i386` depending on you OS
- copy all the `.dll` files located here (namely, `R.dll`, `Rblas.dll`, `Rgraphapp.dll`, `Rinconv.dll`, and `Rlapack.dll`) to the `C:\Windows\System32` folder

Notice that if `C:\Windows\System32` is not a valid Java Library Path, you will have to move the copied `.dll` files into a valid one.

As a last step, we now need to set two environmental variables:

- `R_HOME` with the value obtained by the R command `Sys.getenv("R_HOME")`
- `R_LIBS_USER` with the values obtained by the R command `.libPaths()` divided by semicolons

A comprehensive guide on how to set environmental variables in Microsoft Windows is out of the scope of this guide; thus we refer to available online resources.

At this point `cyTRON` can be installed and executed.