Run R on BioHPC

R shell

R

```
qisun@cbsum1c2b010:~
[qisun@cbsum1c2b010 ~]$ R

R version 4.0.5 (2021-03-31) -- "Shake and Throw"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

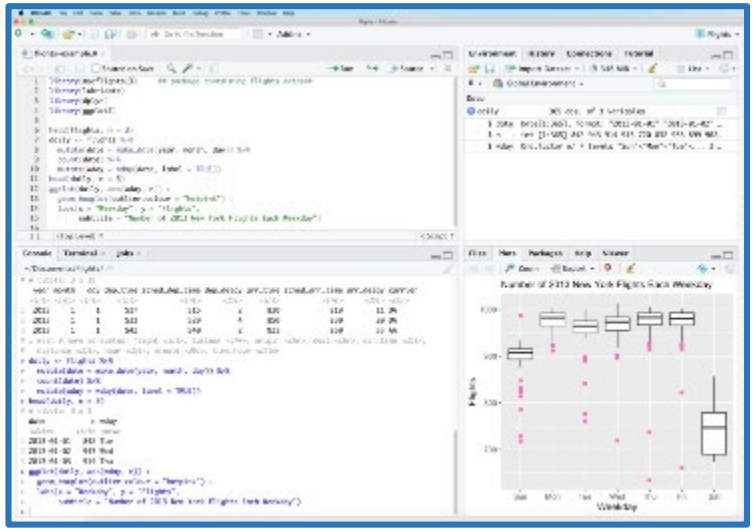
R script

Rscript myscript.R

```
qisun@cbsum1c2b010:~
[qisun@cbsum1c2b010 ~]$ which Rscript
/programs/bin/R/Rscript
[qisun@cbsum1c2b010 ~]$ Rscript myscript.R
```

Rstudio

Through a web browser



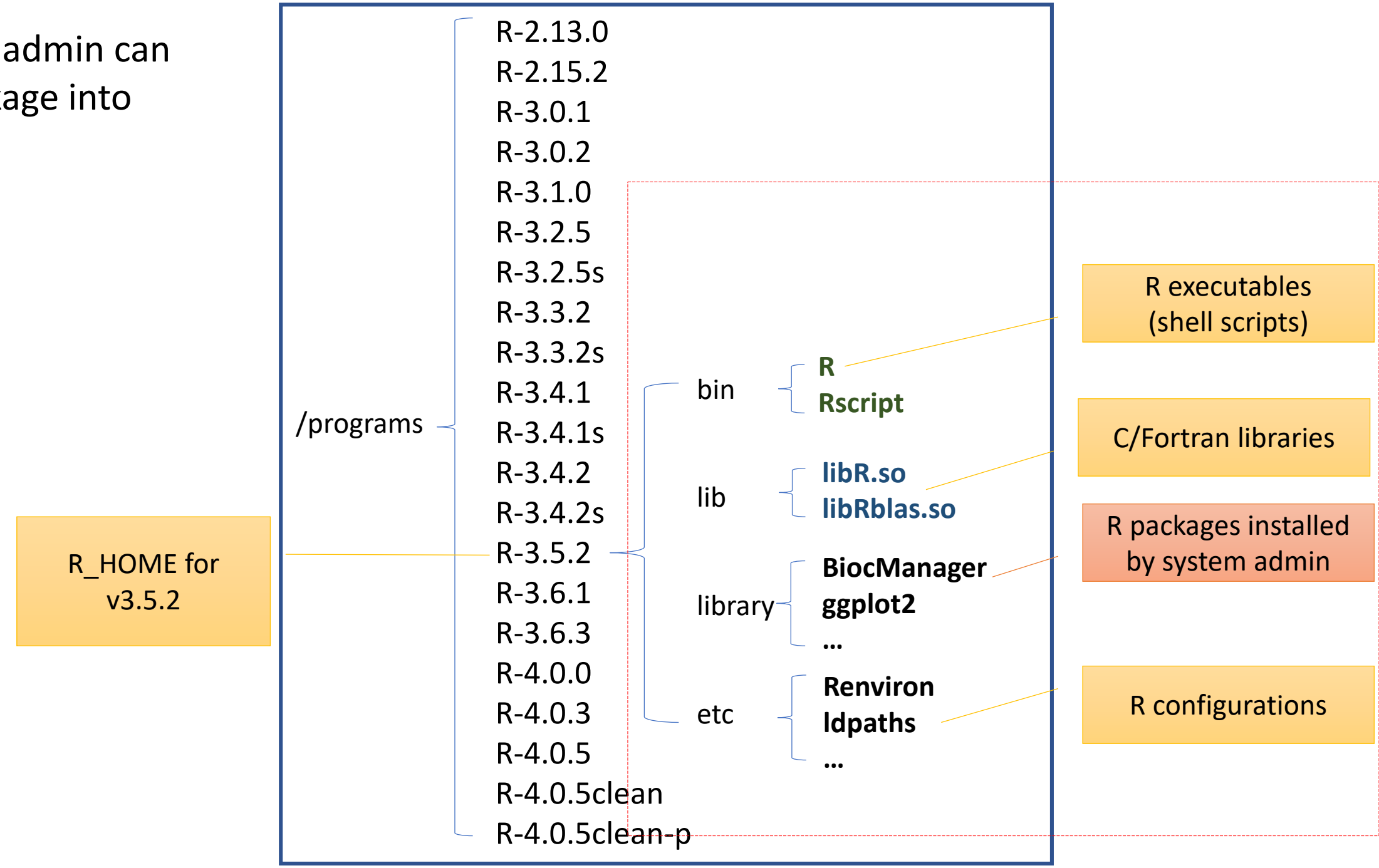
Versions of R

Default R on BioHPC (10/11/2021)

R	**4.0.5**
Bioconductor	**3.12**
R packages	**-**
GCC compiler	**10**
BLAS	**ref. BLAS**

R 4.1.1 (August, 2021)
R 4.1.0 (May, 2021)
R 4.0.5 (March, 2021)
R 4.0.4 (February, 2021)
R 4.0.3 (October, 2020)
R 4.0.2 (June, 2020)
R 4.0.1 (June, 2020)
R 4.0.0 (April, 2020)
R 3.6.3 (February, 2020)
R 3.6.2 (December, 2019)
R 3.6.1 (July, 2019)
R 3.6.0 (April, 2019)
R 3.5.3 (March, 2019)
R 3.5.2 (December, 2018)
R 3.5.1 (July, 2018)
R 3.5.0 (April, 2018)
R 3.4.4 (March, 2018)
R 3.4.3 (November, 2017)
R 3.4.2 (September, 2017)
R 3.4.1 (June, 2017)
R 3.4.0 (April, 2017)
R 3.3.3 (March, 2017)
R 3.3.2 (October, 2016)
R 3.3.1 (June, 2016)
R 3.3.0 (April, 2016)
R 3.2.5 (April, 2016)
R 3.2.4 (March, 2016)
R 3.2.3 (December, 2015)
R 3.2.2 (August, 2015)
R 3.2.1 (June, 2015)
R 3.2.0 (April, 2015)
R 3.1.3 (March, 2015)

Only System admin can install a package into "$R_HOME"



Switching to a different version of R

Switching versions for R shell and Rscript

Check available versions

```
[qisun@cbsumlc2b010 ~]$ module avail

------------------------ /usr/share/Modules/modulefiles ------------------------
R/2.13.0        R/3.5.0         R/4.0.5clean    java/13.0.2     python/2.7.15
R/2.15.2        R/3.5.0s        dot             module-git      python/2.7.5
R/3.0.2         R/3.5.2         gcc/10.2.0      module-info     python/3.6.7
R/3.1.0         R/3.6.1         gcc/5.5.0       modules         python/3.9.6
R/3.2.5         R/3.6.3         gcc/7.3.0       null            use.own
R/3.3.2         R/4.0.0         java/1.7.0      perl/5.16.3
R/3.4.2         R/4.0.5         java/1.8.0      perl/5.22.0
```

Switch to a different version

```
module load R/3.5.2
```

Switch back to default

```
module unload R/3.5.2
```

Behind scene: export PATH=/programs/R-3.5.2/bin:$PATH

Switching versions for Rstudio

```
/programs/rstudio_server/rstudio_stop
rm -fr /home/$USER/.rstudio
rm -fr /workdir/$USER/rstudio
/programs/rstudio_server/mv_dir
/programs/rstudio_server/rstudio_start 3.6.3
```

Details in
https://biohpc.cornell.edu/lab/userguide.aspx?a=software&i=266#c

BLAS library of R: reference BLAS vs OpenBLAS

(Basic Linear Algebra Subprograms)

On BioHPC, before v4.0.5, default R uses **OpenBLAS**; since v4.0.5, default R uses **Reference BLAS**.

- Before v4.0.5, "_s" in the version label indicates "reference BLAS";
- From v4.0.5, "_p" in the version label indicates "OpenBLAS";

How to check: `sessionInfo()`

Open BLAS

```
BLAS/LAPACK: /programs/OpenBLAS/lib/libopenblas_sandybridgep-r0.2.18.so
```

Reference BLAS

```
BLAS:   /programs/R-4.0.5/lib/libRblas.so
LAPACK: /programs/R-4.0.5/lib/libRlapack.so
```

Why does it matter?

- OpenBLAS could be >10x faster if BLAS library is used;

- However, OpenBLAS could crash some R packages (error message: illegal operand)

R Packages/Libraries

R packages vs **shared libraries**

- **R packages** (e.g. Seurat, sf, ggplot2)
  - Written in R scripting language;
  - You can check the paths by .libPaths();

- **C/Fortran libraries** (e.g. libgsl.so, libMagick++-6.Q16.so)
  - Written in C or Fortran, requiring compilation;
  - Some are difficult to install without root privilege;

Where are R package located on BioHPC? It depends on who installed the package.

If installed by system admin

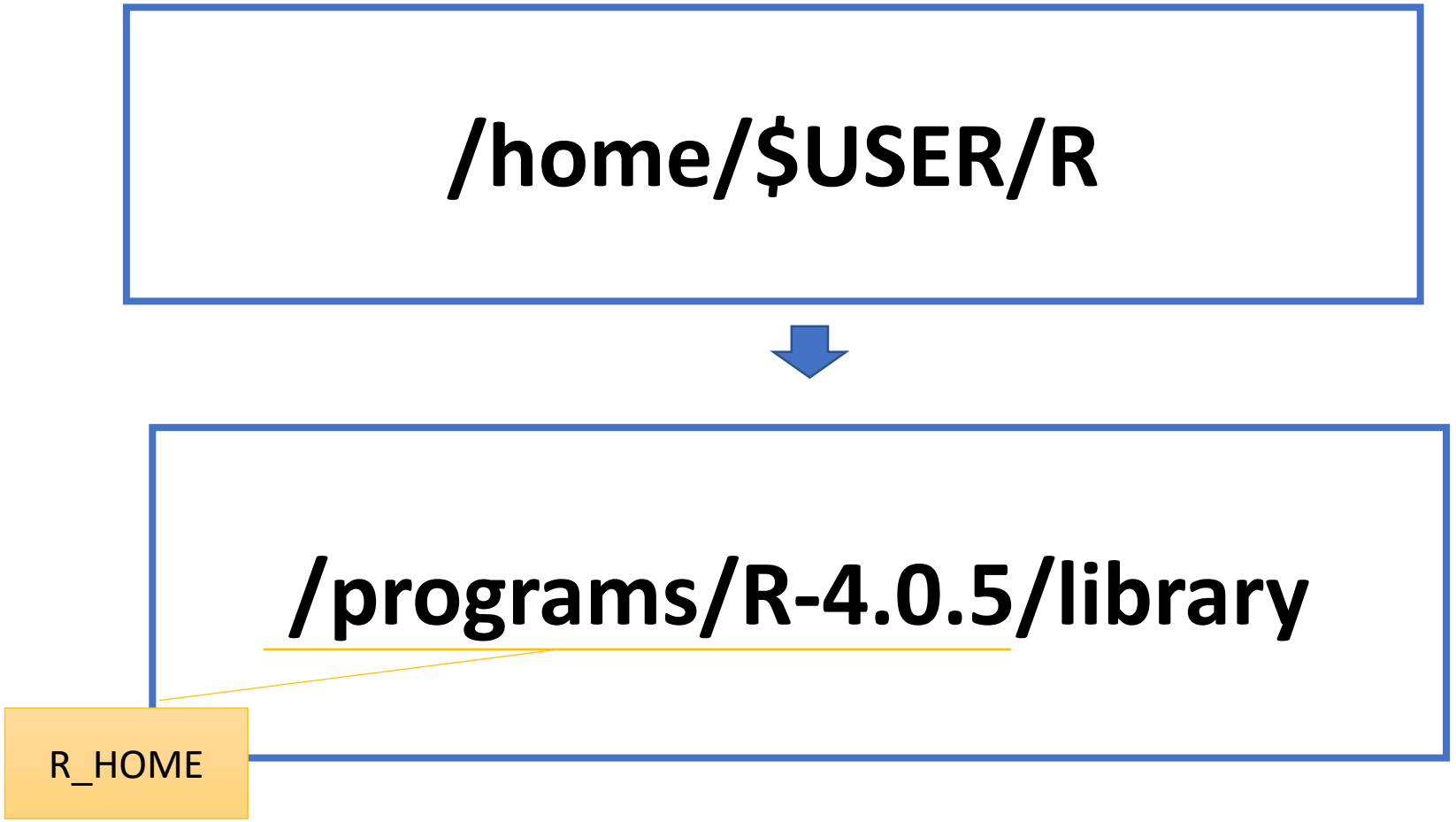
/programs/R-4.0.5/library

R_HOME

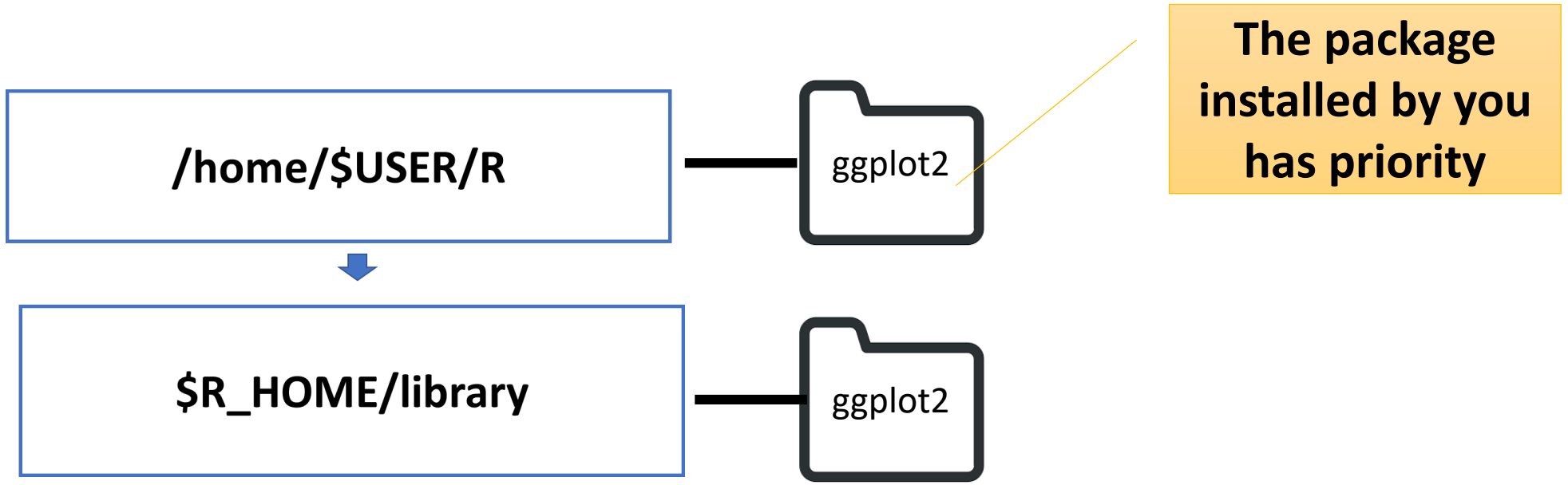
If installed by yourself

/home/$USER/R

When running the command: library(packageName), R follows this order to find the package:



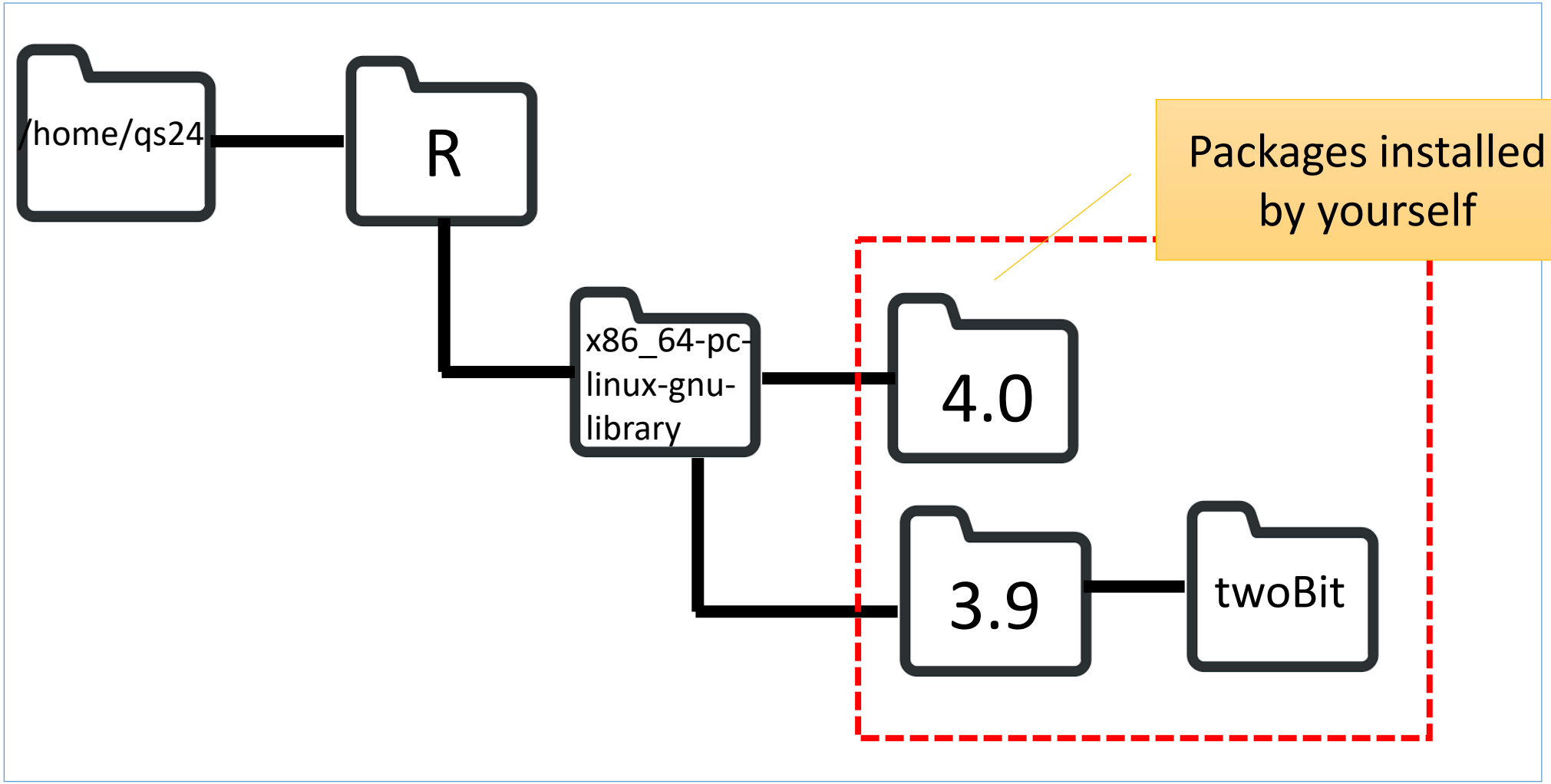
* R would use the first found package;



If “ggplot2” are installed both in your home directory and $R_HOME, R would pick “ggplot2” in your home directory.

If you want to switch to “ggplot2” installed by system admin, either delete your “ggplot2” directory, or change it to a different name.

If installed by yourself, the packages are under your home directory



To remove a package, for example to remove the “twoBit” package, run remove.packages(“twoBit"), or simply delete the “twoBit” directory and change the directory to a different name.

Install R packages by yourself:

- The package will be installed into $HOME/R;
- If you switch to a different versions of R, you need to install the package again.

Install R packages from 4 different sources

From CRAN

```
install.packages ("GD")
```

From BioConductor

```
BiocManager::install("edgeR")
```

From github

```
llibrary(devtools)
install_github("rqtl/qtl2geno")
```

From source file

```
install.packages("mypackage.tar.gz",
repos = NULL, type ="source")
```

- When working with a new system, install at least one package through “install.package()” before installing through other methods, because the “install.package()” function establishes all required settings for personal installation. You can install a dummy package, for example, “install.packages(“testit”).

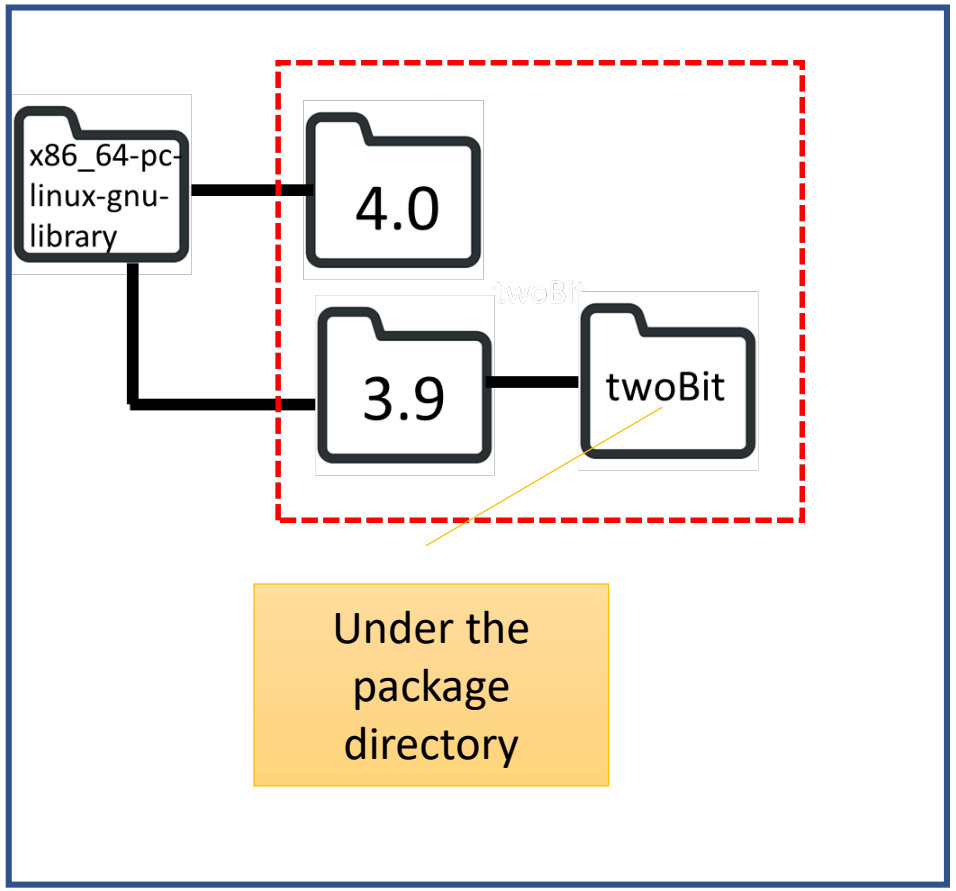
When troubleshooting, verify the version and path of an R package

Some useful R-shell commands:

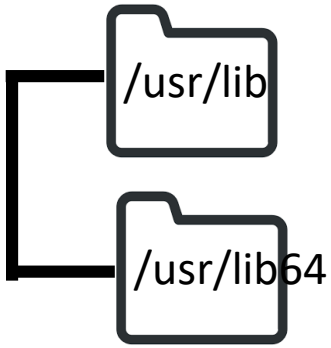
```
# Find package location
find.package("edgeR")

#Check package version
packageVersion("edgeR")

#Check package search path
.libPaths()
```

When installing a R package, the most common problem is missing C/Fortran libraries (e.g missing “libgsl.so” when installing sf package)

In most cases, these libraries are installed in each package separately



In some cases, R packages use libraries in system library directory.



- Many R packages are developed for Ubuntu, but BioHPC runs CentOS. These two Linux have different system libraries with different versions.
- Sometimes, you can install a library in a custom directory, and set LD_LIBRARY_PATH. For example:

export LD_LIBRARY_PATH=/home/qs24/lib

When installing C/Fortran libraries, compilation is needed

Version for default compiler on BioHPC:

Linux system:	gcc 4.8.5
R-3.5.0:	gcc 7.3.0
R-4.0.5:	gcc 10.2.0

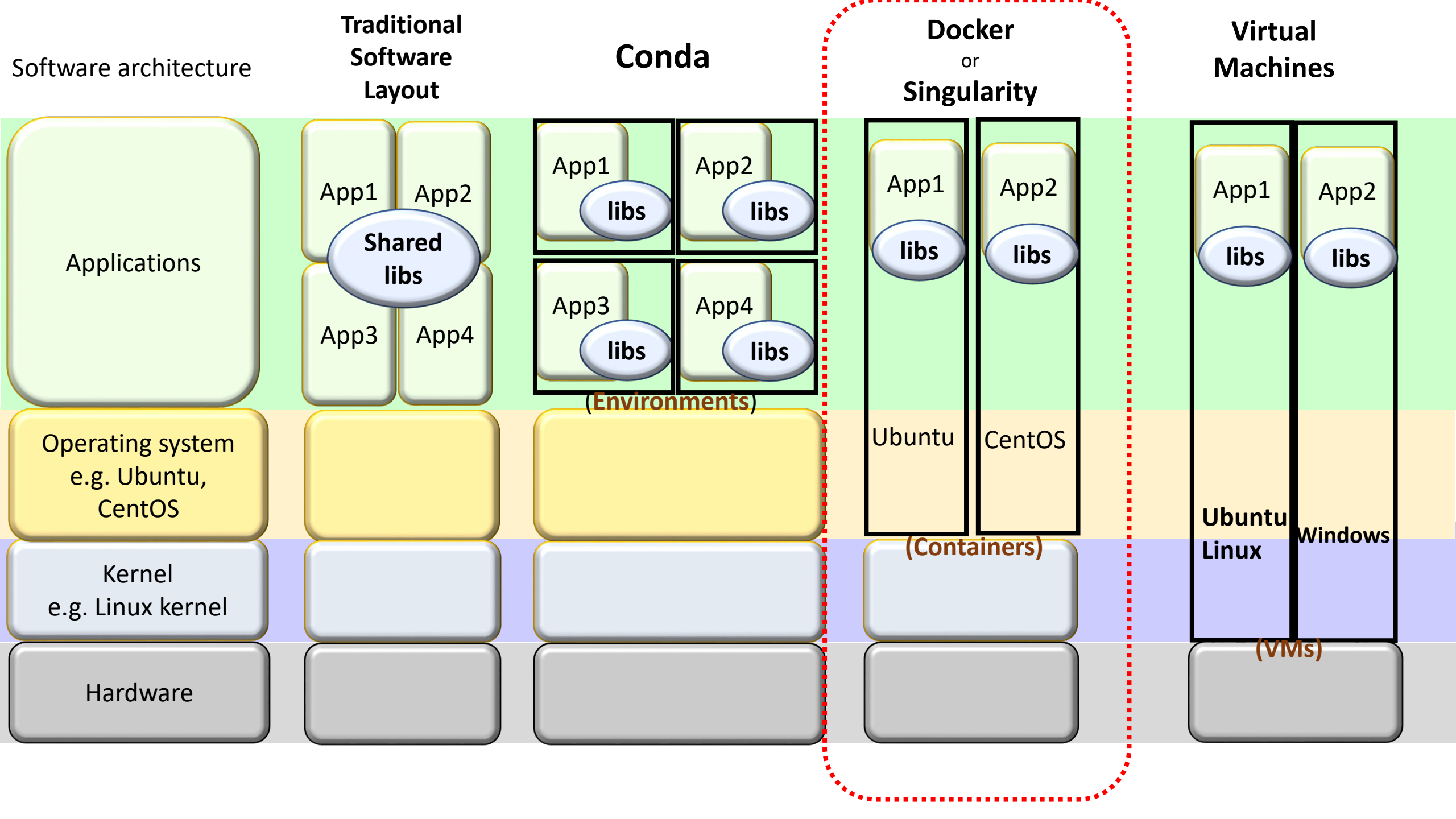
R is getting more complicated

The problems:

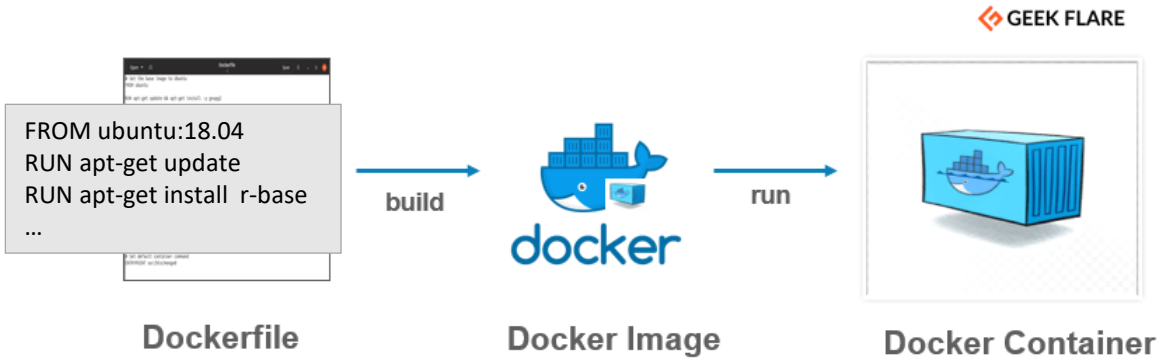
- Some package can only work in certain versions of R, and BioHPC does not maintain all R versions.
- Installing an R package would break other packages in the system. It would be nice to have an isolated “sandbox” to try things out.
- It is important to have a software environment that is reproducible and easy to share with other researchers.

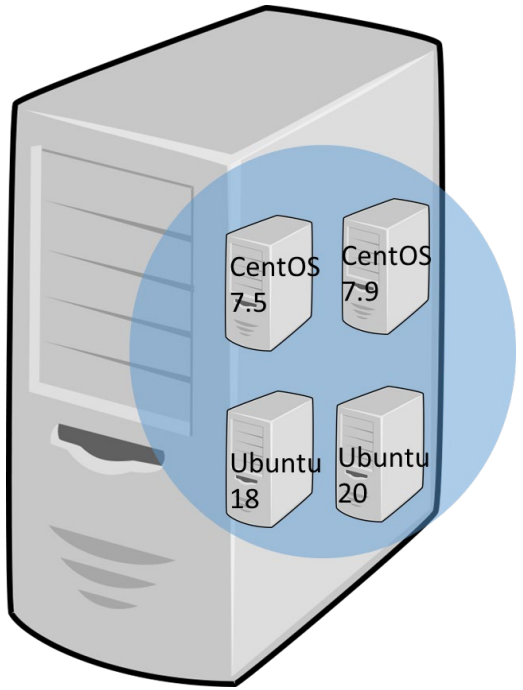
The solutions: **Isolated environment**.

Run R through Docker


Software architecture
Traditional Software Layout
Conda
Docker
or
Singularity
Virtual Machines
Applications
Operating system
e.g. Ubuntu, CentOS
Kernel
e.g. Linux kernel
Hardware
App1
App2
App3
App4
Shared libs
libs
(Environments)
Ubuntu
CentOS
(Containers)
Ubuntu Linux
Windows
(VMs)

Overview of Docker





Docker file:

a text file (script) with instructions how to build a Docker image.

- Including name of operating system, its version and where to download;
- Software/libraries, versions and where to download;
- Environment variable in the system

Docker image:

A software file built from the Docker file.

- Include the actual operating system;
- Software, libraries.

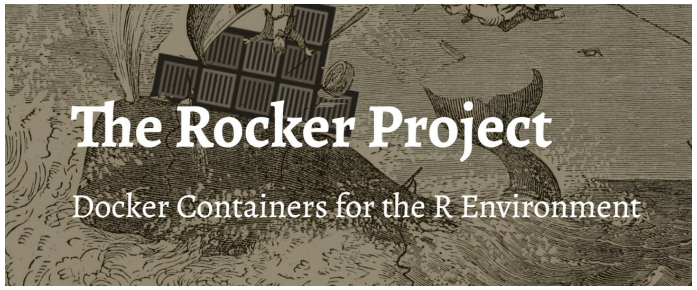
Docker container:

A running instance of the Docker image.

Docker file is not always reproducible for two reasons:

1. The developer often omits the version;
2. The software download link stops working;

Docker image is reproducible.



Docker images with R built-in

-- multiple different types and versions

image	description
r-ver	Specify R version in docker tag. Builds on `debian:stable`
rstudio	Adds rstudio
tidyverse	Adds tidyverse & devtools
verse	Adds tex & publishing-related packages
geospatial	Adds geospatial libraries
shiny	shiny-server on r-base

Sf, rgeos, rgdal, et al. pre-installed.

On BioHPC, use **“docker1”** command

What is “docker1”?

A script to scan the parameters before passing on to the Docker software, to ensure security of the host.

Features of docker1?

- Only directories under /workdir/$USER can be mounted in Docker container;

- /workdir/$USER is automatically mounted as /workdir in Container;

How to use Rocker images　1. Start a container

Download images from Docker hub	docker1 pull rocker/r-ver
Check images	docker1 images
Start a container	docker1 run -dit rocker/r-ver /bin/bash
Check containers	docker1 ps -a

How to use Rocker images 2. Run R in container

Start the shell of the active container	docker1 exec -it 4d6b1fb2d504
Start R in container	R

* Replace 4d6b1fb2d504 with your container ID

- Now you are running R inside the container. As you are running as the root user in container, you have full privilege to install any software.
- Any new files created in the container are owned by root. “docker1 claim” command would give you the ownership.
- After you install all packages and libraries in the container, you might want to save the container as a new image, so that it can be used later. Otherwise, a container will be lost after you terminate the container.

Using Rocker images 3. Save container as a new image

Save the container as a new image	docker1 commit 4d6b1fb2d504 rocker_new
Export the image as a portable file	docker1 export -o rocker_new.tar rocker_new

- You can load the image file rocker_new.tar later in a different computer with the “docker1 load” command;
- You can also push the image file to the Docker hub to share with other users.

You can also use Rocker images through Singularity

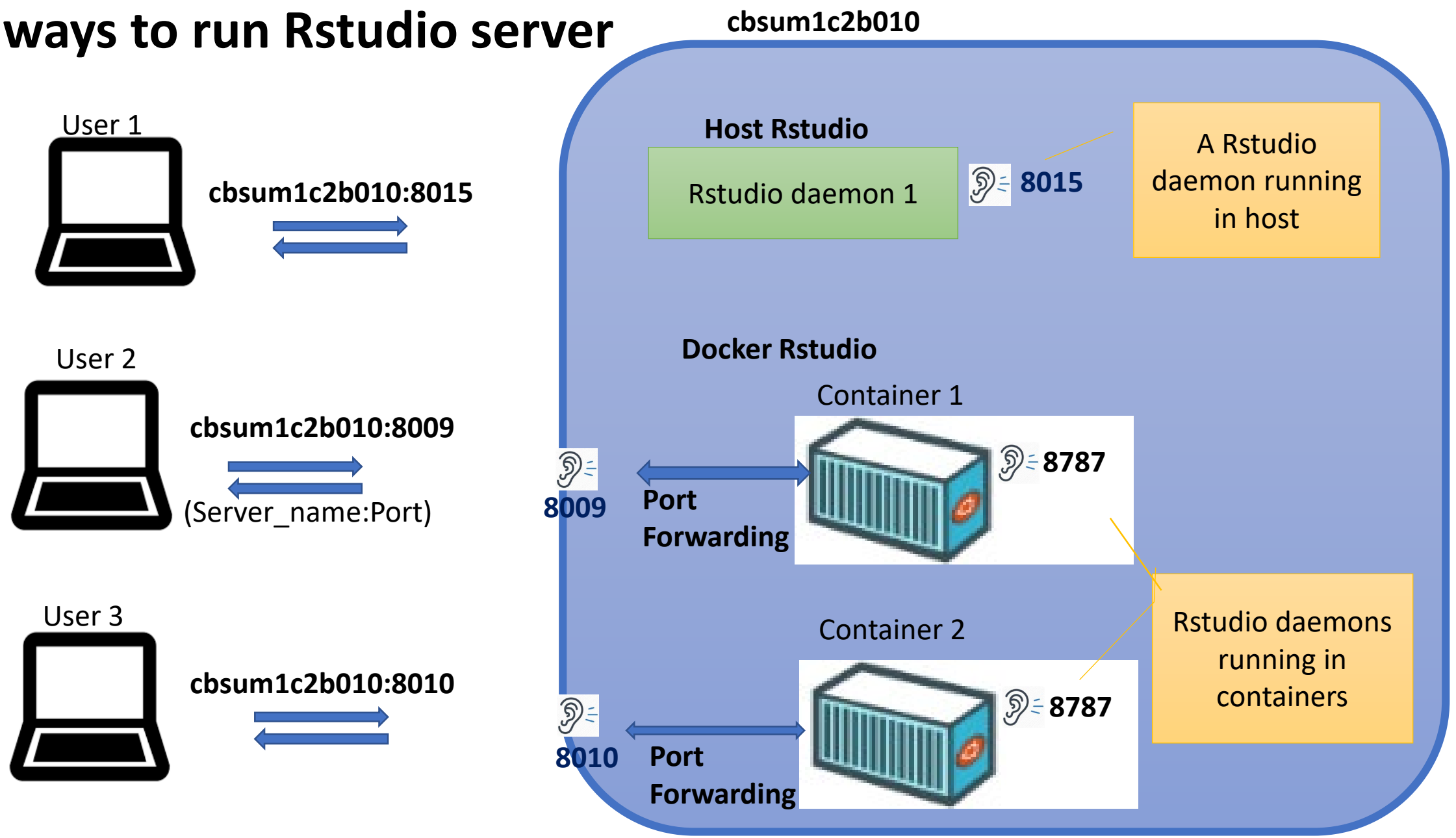
- BioHPC supports both **Docker** (through docker1 command) and **Singularity**, but most other HPC centers only support Singularity;

- Docker is good for setting up services in a server, e.g. web services like Rstudio. Singularity is easier to use for computing.

Rstudio

Two different ways to launch Rstudio in BioHPC

Two ways to run Rstudio server



https://biohpc.cornell.edu/lab/userguide.aspx?a=software&i=266#c

Two different ways to start Rstudio on BioHPC

Host Rstudio: Rstudio server running in the host system

```
/programs/rstudio_server/mv_dir

/programs/rstudio_server/rstudio_start 3.6.3
```

Port: 8015

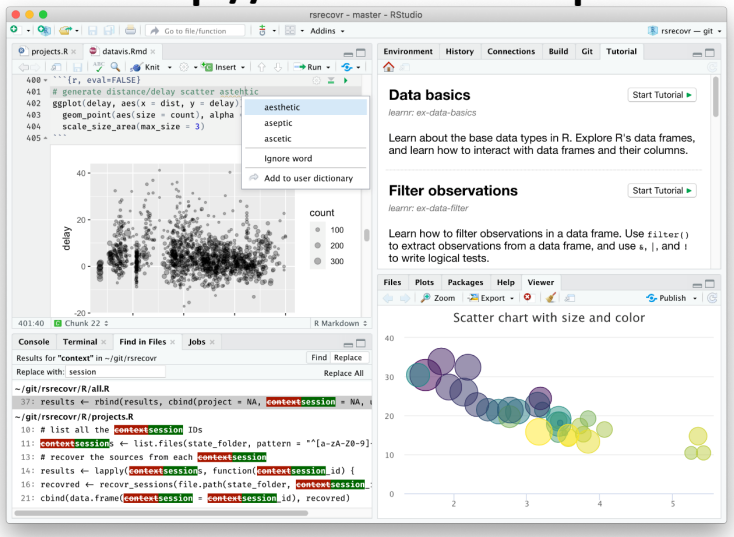
Docker Rstudio: Rstudio server running in Docker container

```
#start Docker container, which automatically start Rstudio
docker1 run -d -p 8009:8787 -e PASSWORD=yourPassword
rocker/rstudio:4.1.0

#add your BioHPC user ID into Docker container(replace xxxxxxx with container ID)
#and set password for this user ID
docker1 exec xxxxxxx useradd -m -u `id -u` -d /home/$USER $USER
docker1 exec -it xxxxxxx passwd $USER
#make the user a sudo user
docker1 exec xxxxxxx  usermod -aG sudo $USER
```

Use this command to start Rstudio, with port 8009. The password here is for a built-in user “rstudio”. If you do not set the password, it has a published default password “rstudio”, and anyone can login as this user.

Use these three commands to add your BioHPC user ID into the container, set password, and make it a “sudo” user.

After Rstudio server is started, you can connect to Rstudio server from a browser on your laptop. The URL is: http://cbsuxxxxx.biohpc.cornell.edu:80xx



Host Rstudio vs **Docker Rstudio**

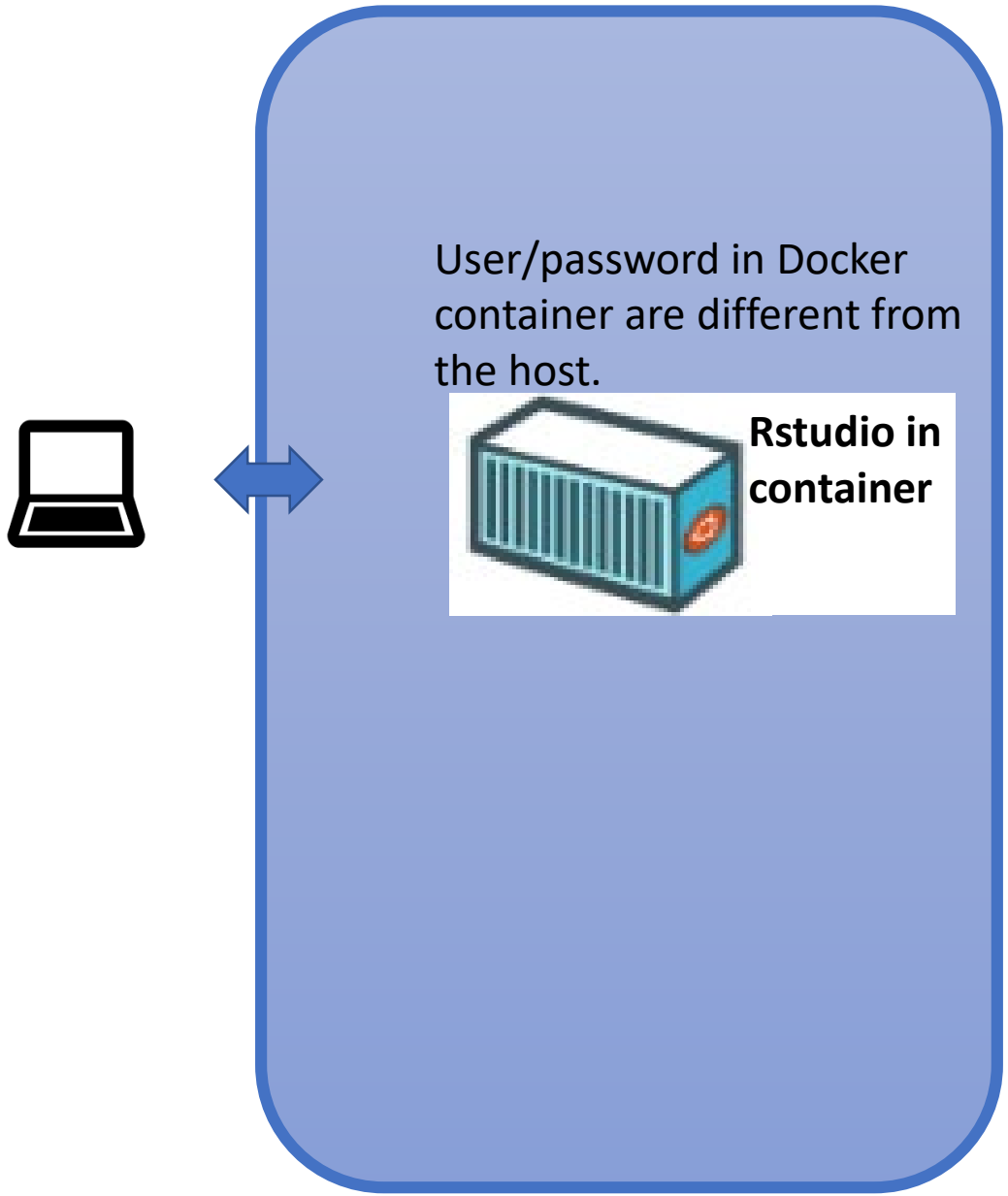
Host Rstudio

- Easy to start;
- R versions are limited to what are available on BioHPC
- Only one instance can run on the server, shared by all users;
  - If one user crashs Rstudio, all other users' work are killed;
  - As different users share the same environment, you might have to compromise with package versions.

Docker Rstudio

- Starting server take a few more steps, and you need to manage user/password inside container;
- You can use any R versions available
- Multiple instances can run on the server, no need to share with other users;

With **Docker Rstudio**, you need to manage users inside the container



- You should add your BioHPC user ID into the container. You can optionally make it a “sudo” user. (Being a “sudo” user makes it easer to install software.)

```
#add your BioHPC user ID
docker1 exec xxxxxxx useradd -m -u `id -u` -d /home/$USER $USER

#Set password for the BioHPC user in container
docker1 exec -it xxxxxxx passwd $USER

#Optionally, make the user as a “sudo” user in container
docker1 exec xxxxxxx  usermod -aG sudo $USER
```

(xxxxxxx is the Docker container ID that you can get with “docker1 ps” command)

Access Docker Rstudio

- Open web browser, login as BioHPC user ID;

- Set working directory to “/workdir” by this command: setwd("/workdir")
  (“/workdir/” in container is the “/workdir/$USER” directory on host).

- Click the “Terminal” tab to install extra software/libraries as needed.

Rstudio save your session data to a file in your home directory after 2 hours of inactivity.

If you work with very big data set, this is not good for both Host Rstudio and Docker Rstudio.

For details, read https://support.rstudio.com/hc/en-us/articles/218417097-Filling-up-the-home-directory-with-RStudio-Workbench-RStudio-Server

Host Rstudio

1. In BioHPC, your home directory is slow and has limited space.

2. We provide a script "/programs/rstudio_server/mv_dir" that moves the directory to /workdir, and put a symbolic link of the directory in your home directory.

3. Alternatively, you could start Rstudio with the "noswap", Rstudio would not automatically write session to file, but that would cause large amount of data staying in memory.

Docker Rstudio

1. You do not want the large session files get committed into your next image file. Delete the /home/$USER/.rstudio directory before next committing.

2. One option is to move the session files to /workdir, and make a symbolic link in home directory.

3. Alternatively, you can modify the file /etc/rstudio/rsession.conf by adding this line, so that Rstudio does not save session data to a file.

```
session-timeout-minutes=0
```