The University of Queensland

Research Computing Centre

Installation of Additional Capability into R/Perl/Python

Document Status

Last updated on July 13 2020 by David.Green@uq.edu.au

The scripting languages R, Perl and Python are supported by a plethora of libraries/packages/modules from their open source communities.
The RCC cannot, and will not, install every interesting package. (Sorry about that!)

We keep the central language installations fairly clean and enhance their functionality in three ways:

1. add system packages that provide required functionalities (for example, the perl-JSON system package is installed)
2. build a core set of highly regarded packages (for example the contents of /opt/R/local/lib/ contains 248 R libraries)
3. supporting the use of personal libraries/packages/modules (which is the subject of this user guide)

Personal R Libraries

The process of installing R libraries can be done from inside the R language. However, there are a couple of steps that need to be followed first.

Prepare a location to install your packages.

The standard package install locations require administration privileges to add new packages.
Therefore, you need to use a location that you are permitted to write into when you need to build packages.

You can either

- let R choose a standard location for you
- create your own location and configure your R sessions to use it.
  Both approaches have merit.

Recent versions of R use a standard location that includes information about the system architecture:

```
${HOME}/R/x86_64-pc-linux-gnu-library/3.4 ...
${HOME}/R/x86_64-pc-linux-gnu-library/3.5 ...
${HOME}/R/x86_64-pc-linux-gnu-library/3.6 ...
${HOME}/R/x86_64-pc-linux-gnu-library/4.0 ...
```

This is where R will install your personal packages unless you take steps to the contrary.
You should not require all the .Rprofile settings that are mentioned in the Using R user guide.
This is advantageous if you use different versions of R that are always on "bare metal" (that is R versions installed in /opt and some in /sw).

However, using standard locations may present difficulties if you switch between bare metal and versions of R that are provided using software containers R (e.g. the RStudio and RJAGS containers.)
The R software provided in software containers is usually configured to use GNU compilers that are within the container.
Mixing of R packages that are built inside and outside software containers **is not recommended**.
Having a distinct location that you can use as required gives you much more flexibility and control, however you need to careful to ensure that you get the right versions of packages each time.
Using the .Rprofile and .libPaths() is one way to ensure you get the correct libraries loaded each time.

In this guide we will assume the directory .R/library (but you could equally use some project specific location).
You may wish to use this approach to separate R packages built for/by software containers, from those built for/by the centrally installed R.

To create your own R installation location you would make the directory:

```
user@awoonga1:~> mkdir -p ~/.R/library
```

The -p creates the parent directories if they do not exist.

Now that you have a location for installing packages, you need to let R know about it. This is done by appending the path to the libPaths. While this can be typed into the R interpreter every time start R, adding this to the ~/.Rprofile will mean that it is done every time you start R.

```
user@awoonga1:~> echo '.libPaths( c("~/.R/library", .libPaths()) )' >> ~/.Rprofile
```

A cat of your .Rprofile should now contain at least the following line.

```
user@awoonga1:~> cat ~/.Rprofile
.libPaths( c("~/.R/library", .libPaths()) )
```

Search First, Build Second

More competent R users than me will know how to do this within R itself.
Before you launch into (re-)building R packages and dependencies, you should check they don't already exist in your .libPaths().

I find this sort of search using the find command works for me.

```
module load R
echo $R_LIBS
find $R_LIBS $HOME/R/ -type d -name NAME_OF_R_PACKAGE
```

e.g.

```
uqdgree5@flashlite1:.../Clients> module list

Currently Loaded Modules:
  1) intel/2018.2.046   2) R/3.5.0

uqdgree5@flashlite1:.../Clients> find $R_LIBS $HOME/R/ -type d -name data.table
/opt/R/local/lib/data.table

uqdgree5@flashlite1:.../Clients/uqrdomin/scenarios_q1> find $R_LIBS $HOME/R/ -type d -name spatstat
/home/uqdgree5/R/x86_64-pc-linux-intel-library/3.5/spatstat
```

Option 1: Install packages the Rstudio way

RStudio (http://www.rstudio.com/) Version 1.0.143 has been deployed on HPC as a software container. The container includes GCC build tools and a specific R version 3.4.0. It can not be used with other versions of R, nor probably any packages built with bare metal versions of R on the system. As with other software modules, an up to date list of which versions of Rstudio can be queried using

```
user@awoonga1:~> module avail rstudio
```

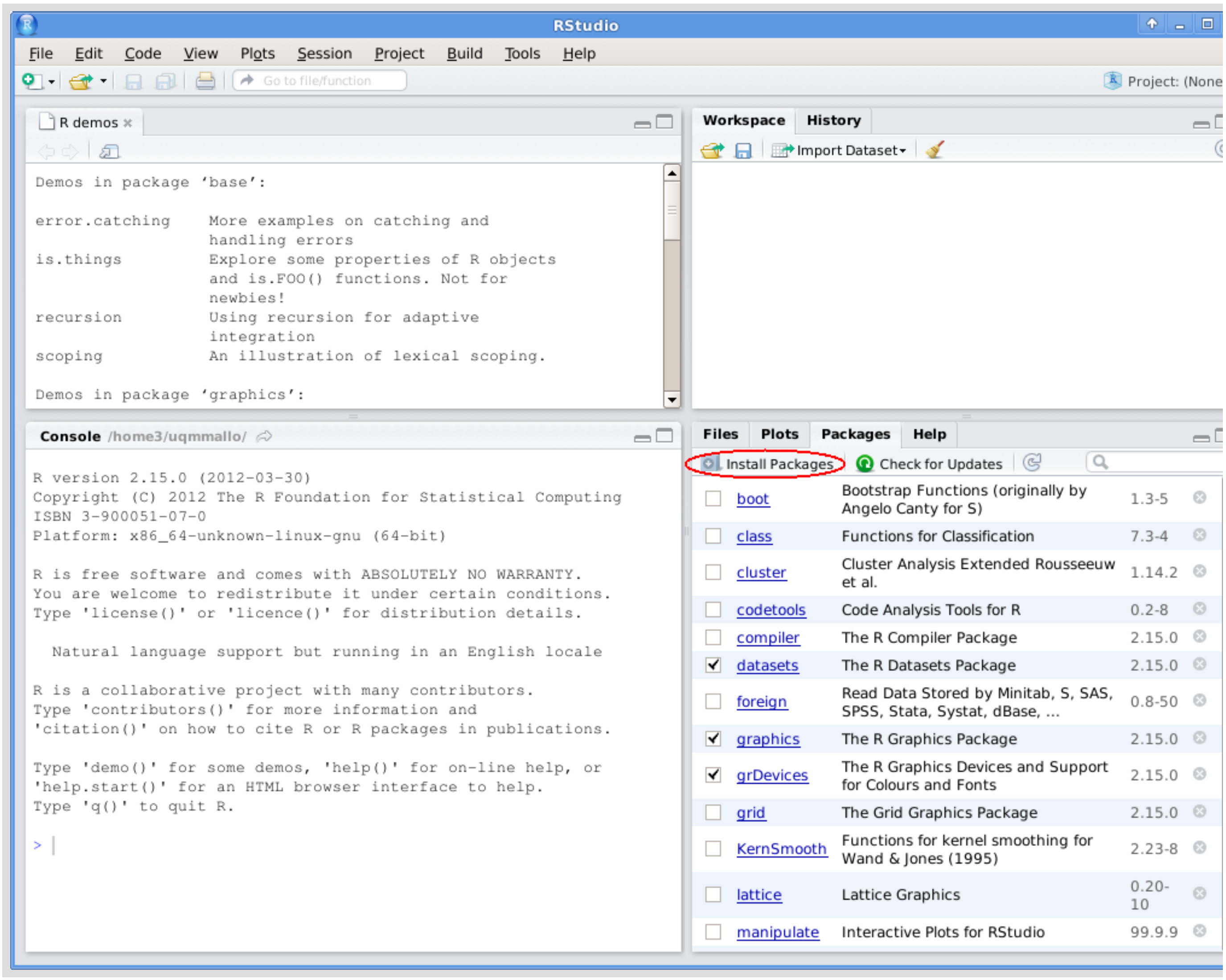
Using Rstudio requires X11 forwarding. Assuming that X11 forwarding has been correctly setup, Rstudio can be used by:

```
user@awoonga1:~> module load rstudio
user@awoonga1:~> rstudio
```

This will load the default version of RStudio container. The rstudio module will automatically load the associated version of R and will create an alias that will launch RStudio.

Once RStudio starts, select a CRAN mirror by going to tools > options > Packages and change the CRAN mirror. The Australia (Melbourne) mirror is recommended as it is on AARNet.

Packages can be installed via the "install packages" button under the packages tab.

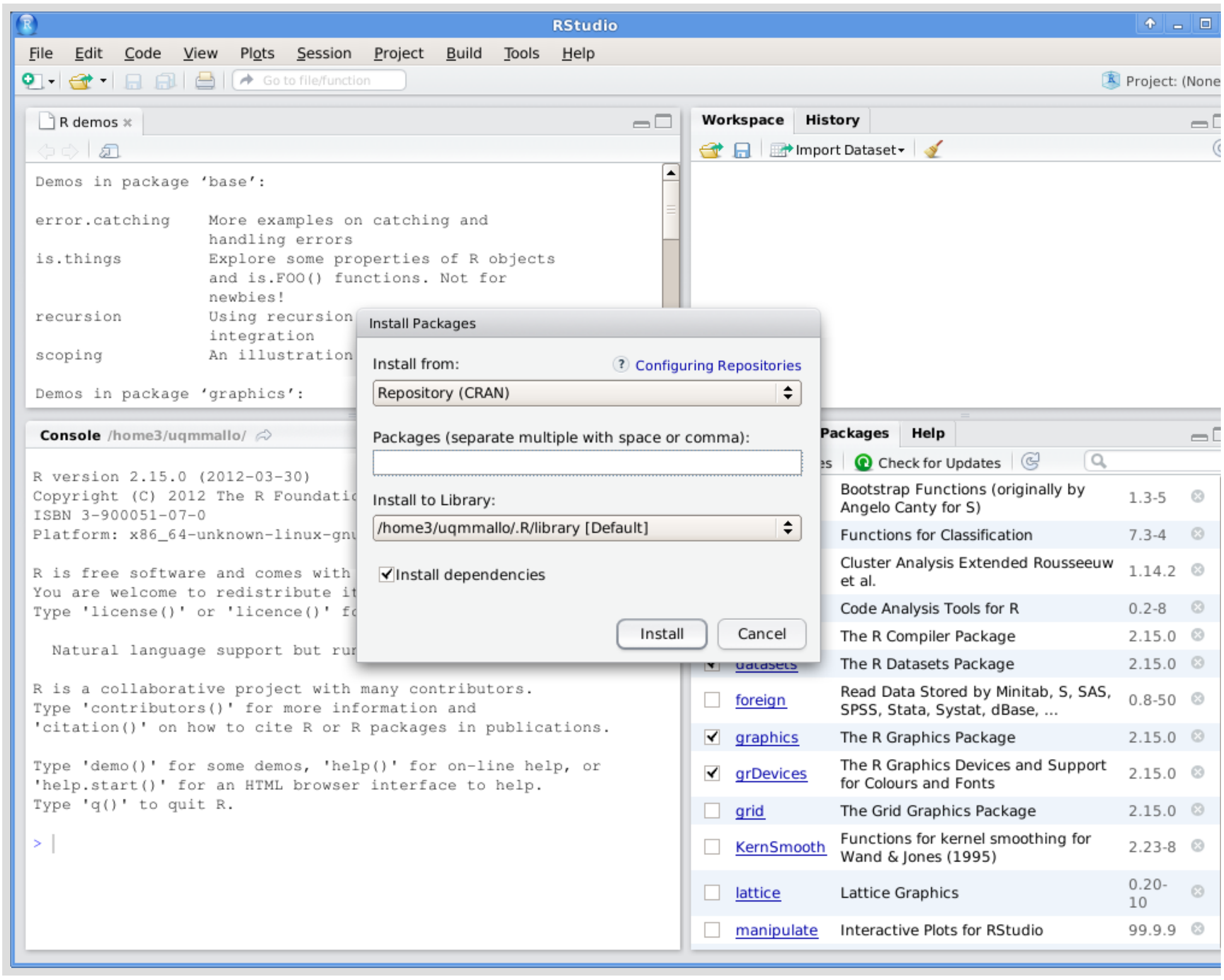


```
Demos in package 'base':

error.catching    More examples on catching and
                  handling errors
is.things         Explore some properties of R objects
                  and is.FOO() functions. Not for
                  newbies!
recursion         Using recursion for adaptive
                  integration
scoping           An illustration of lexical scoping.

Demos in package 'graphics':
```

```
R version 2.15.0 (2012-03-30)
Copyright (C) 2012 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-unknown-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.

> 
```

☐	boot	Bootstrap Functions (originally by Angelo Canty for S)	1.3-5
☐	class	Functions for Classification	7.3-4
☐	cluster	Cluster Analysis Extended Rousseeuw et al.	1.14.2
☐	codetools	Code Analysis Tools for R	0.2-8
☐	compiler	The R Compiler Package	2.15.0
✓	datasets	The R Datasets Package	2.15.0
☐	foreign	Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, dBase, ...	0.8-50
✓	graphics	The R Graphics Package	2.15.0
✓	grDevices	The R Graphics Devices and Support for Colours and Fonts	2.15.0
☐	grid	The Grid Graphics Package	2.15.0
☐	KernSmooth	Functions for kernel smoothing for Wand & Jones (1995)	2.23-8
☐	lattice	Lattice Graphics	0.20-10
☐	manipulate	Interactive Plots for RStudio	99.9.9

Clicking on install packages will bring up a dialog. Ensure that "Install to library" is set to the ~/.R/library directory. If the directory was correctly added to .libPaths, then it will be listed in the drop down menu.



```
Demos in package 'base':

error.catching      More examples on catching and
                    handling errors
is.things           Explore some properties of R objects
                    and is.FOO() functions. Not for
                    newbies!
recursion           Using recursion
                    integration
scoping             An illustration

Demos in package 'graphics':
```

```
R version 2.15.0 (2012-03-30)
Copyright (C) 2012 The R Foundatic
ISBN 3-900051-07-0
Platform: x86_64-unknown-linux-gnu

R is free software and comes with
You are welcome to redistribute it
Type 'license()' or 'licence()' fc

  Natural language support but rur

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.

>
```

Install Packages

Install from: Repository (CRAN) — Configuring Repositories

Packages (separate multiple with space or comma):

Install to Library: /home3/uqmmallo/.R/library [Default]

✓ Install dependencies

Install — Cancel

		Bootstrap Functions (originally by Angelo Canty for S)	1.3-5
		Functions for Classification	7.3-4
		Cluster Analysis Extended Rousseeuw et al.	1.14.2
		Code Analysis Tools for R	0.2-8
		The R Compiler Package	2.15.0
✓	datasets	The R Datasets Package	2.15.0
☐	foreign	Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, dBase, ...	0.8-50
✓	graphics	The R Graphics Package	2.15.0
✓	grDevices	The R Graphics Devices and Support for Colours and Fonts	2.15.0
☐	grid	The Grid Graphics Package	2.15.0
☐	KernSmooth	Functions for kernel smoothing for Wand & Jones (1995)	2.23-8
☐	lattice	Lattice Graphics	0.20-10
☐	manipulate	Interactive Plots for RStudio	99.9.9

Add the list of packages and click install. Rstudio will then download and install each package.

Option 2: Install packages from the interactive R prompt

The process for installing R packages from the interactive R prompt is fairly straightforward. X11 forwarding is helpful but not required.

1. Get an R prompt

```
user@awoonga1:~> module load R
user@awoonga1:~> R
>
```

2. Choose a CRAN mirror

```
> chooseCRANmirror()
```

   If X11 forwarding is enabled, a window will pop up that lets you select which mirror to use, otherwise, a list of mirrors with a numerical value can be selected from a prompt. It's recommended that the Australia (Melbourne) mirror is used as it is on AARNet.

3. Install packages

```
> install.packages(c("pkg1", "pkg2"))
```

   If you want to install a package and all of its dependencies, use

```
> install.packages("Rcmdr", dependencies = TRUE)
```

Compilation Woes

Sometimes you can struggle with getting libraries built.

Precompiled R packages

Sometimes an R library package is distributed as a pre-compiled package directory that is "ready-to-use" - and no compilation is required!
It just needs to be placed in the correct location for the version of R that you are using.

You do need to ensure that if the package was built using the GNU C compiler GCC (which it almost certainly was), that you only use it with a GCC build of R.
On the HPC currently `module load R/3.5.0-gnu` or a container versions of R were built using GCC.

Missing a Compiler, anybody?

Some packages are intended to be built with a particular version of a compiler so that newer software features can be utilised by the software.
If you see a message like this

```
configure: error: "No C++14 compiler identified by R CMD config CXX14"
```

then your library build script is unable to locate a suitable compiler (in this case one that supports the C++ V14 standard).
The fix can be as simple as helping the R build mechanim to find the appropriate compiler by customising your R makevars

```
echo "CXX14=icc" >> ~/.R/Makevars
```

So in this case, we tricked R into using the Intel icc compiler which is C++ V14 compliant.

Similar customisation of Makevars can be useful to ensure you are picking up the non-standard locations for INCLUDEPATH and the like.

Missing operating system packages

Building R packages from source can sometimes fail because the build mechanism expects to have access to components of what are referred to as "dev packages" for the operating system.
The symptoms are often messages in the "configure" stage saying that such and such was not found.
In that case the sensible way forward is to have the operating system packages installed into the HPC, or, for a software container to be built that contains all the required operating system packages as well as a usable installation of R. Occastionally, RCC has been able to build a bespoke software container for a specific purpose and project, but this is not often feasible.

You would need to request installation of operating system packages or the creation of a customised R container via email to rcc-support@uq.edu.au
If we can accommodate it safely in the HPC operating systems we will, otherwise we will have to build a software container built from a different linux distribution and provide a complete environment for using that R package.

Missing R packages

Building R packages from source can sometimes fail because they expect to have access to other R libraries that are not yet built.
Before you rush to `install.packages` with the `dependencies=TRUE` option it is worth checking that your .libPaths() includes the centrally provided libraries.
The software modules for bare metal installations of R include an environment variable R_LIBS which is the location of additional libraries that are provided with that module load.
It will save you time effort and filesystem quota, if you are able to use the pre-existing libraries that are provided in the bare metal installations.

I have experienced situations where dependencies have been built unnecessarily because of the ordering of library paths in the `.libPaths()`.
A workaround I use is to reverse the order of the entries in the path vector and resave it using `.libPaths()`:

```
#Need to ugly-fix the library path ...
pp <- .libPaths()
.libPaths(c(pp[3],pp[2],pp[1]))
```

I recommend that you do not blindly follow this advice and check what .libPaths() you have in the R instance that you using, then use, modify, or ignore the suggestion as appropriate.

Ensuring you have access to the include files and libraries

If you are building a package for R that depends on other software packages, you may find that the build process fails with a message such as

```
configure: error: proj_api.h not found in standard or given locations.
```

For example, the RGDAL package depends external packages GDAL and PROJ and will not build without access to the header files gdal.h and proj_api.h.
These can be provided by module load-ing GDAL and PROJ modules (which amend the INCLUDE and LD_LIBRARY_PATH variables) before running R to build the package.

```
user@awoonga1:~> module load gdal

user@awoonga1:~> module load proj

user@awoonga1:~> echo $INCLUDE
/sw/PROJ/4.8.0/include:/sw/GDAL/1.9.1/include

user@awoonga1:~> R
```

and during the successful build process

```
.
.
checking gdal.h usability... yes
checking gdal.h presence... yes
checking for gdal.h... yes
checking gdal: linking with --libs only... yes
checking GDAL: /sw/GDAL/1.9.1/share/gdal/pcs.csv readable... yes
checking proj_api.h usability... yes
checking proj_api.h presence... yes
checking for proj_api.h... yes
checking for pj_init_plus in -lproj... yes
configure: PROJ.4 version: 4.8.0
.
.
```

Some Notes about Geospatial Packages

Some of our HPC users have experienced difficulty building R packages that require operating system components of a geospatial type (eg . GDAL and PROJ4)

The rgdal Package

The `rgdal` package is provided as part of the central R installation and should be available once you load the `R/3.5.0` module.

You would still need to load the gdal/2.2.0 environment module (and possibly also the proj/4.9.3 environment module) to make use of it.

If you modify your R environment, or need to build your own version of rgdal, then proceed as follows.

As at November 2019, the following formulation is known to work for building the rgdal package:

```
uqdgree5@tinaroo1:~> module purge
uqdgree5@tinaroo1:~> module load gdal/2.2.0 proj/4.9.3 R/3.5.0
uqdgree5@tinaroo1:~> export PKG_CONFIG_PATH=/opt/proj/lib/pkgconfig/
uqdgree5@tinaroo1:~> R

R version 3.5.0 (2018-04-23) -- "Joy in Playing"
```

SNIP

```
> install.packages("rgdal", dependencies=TRUE, type="source", configure.args=c('--with-gdal-config=/opt/gdal/bin/gdal-config
```

SNIP

```
** testing if installed package can be loaded
* DONE (rgdal)
```

The sf Package

After installing rgdal as above, quit R and then

```
module load ncview geos
export INCLUDE=$INCLUDE:/opt/ncview/include
R
```

then within R

```
install.packages("units", configure.args=c('--with-udunits2-lib=/opt/ncview/lib', '--with-udunits2-include=/opt/ncview/inclu
```

SNIP

```
** testing if installed package can be loaded
* DONE (units)
```

```
> install.packages("sf")
```

SNIP

```
** testing if installed package can be loaded
* DONE (sf)
```

Some notes about bio-related packages

monocle 3

There is an installation guide so follow it but with the following modifications

Mention is made of issues with rgdal and sf packages, so build them first using the suggestions elsewhere in this User Guide.

In order that you may use the devtools technique to build software into your personal library location, you need to modify the way you call devtools as follows:

> devtools::install_github('cole-trapnell-lab/leidenbase', args = c('--library="/home/uqdgree5/R/x86_64-pc-linux-intel-library/3.5"'))
> devtools::install_github('cole-trapnell-lab/monocle3', args = c('--library="/home/uqdgree5/R/x86_64-pc-linux-intel-library/3.5"'))

You may also find that devtools is unable to install dependenciesinto your personal R library direcetory.
You may need to ensure that all dependencies are available before issuing the devtools::install_github command.

The R Packages HOWTO Module

A documentation module is available on the HPC to assist you with building packages in a few scenarios.

```
uqdgree5@tinaroo1:~> module help HOWTO/R-Packages

----------- Module Specific Help for 'HOWTO/R-Packages' -----------

    This HOWTO module attempts to answer the often asked question:
    Where do I build extra packages for R on Tinaroo or FlashLite ?

    The R statistical software provides a mechanism to compile additional packages
    that enhance its functionality.
    Although additional packages can be installed centrally by system administrators,
    it is preferrable that they be installed into a user's personal library.

    R is provided on Euramoo as part of the installed operating system software. Just type R to use it.

    R is provided on Tinaroo and FlashLite via a loadable module.
    This R has been built as a ROCKS "roll" using the intel compilers.

    uqdgree5@tinaroo1:~> module list
    No Modulefiles Currently Loaded.

    uqdgree5@tinaroo1:~> module load R

    uqdgree5@tinaroo1:~> module list
    Currently Loaded Modulefiles:
    1) intel/2016.1.150   2) R/3.2.3

    If you are familiar with using R then you should already know about using personal library paths.
    If not, then please read
    https://cran.r-project.org/doc/manuals/r-release/R-admin.html or
    the RCC User Guide on "Installation of Packages for Scripting Languages like R and Python"

    Often, you would just need to
    module purge
    module load R
    R
        [set the repo to use]
        install.packages("somePackage", dependencies = TRUE)

    Sometimes, you may need to have available other system installed software.
    The R package rgeos depends on GEOS being installed.
    To build rgeos with the cluster version of GEOS, you would
```

```
module purge
module load R
module load geos
R
    [set the repo]
    install.packages("rgeos", dependencies = TRUE)

Sometimes, you may need to feed configure options to the build mechanism.
The R package Rmpi needs OpenMPI (bin, include and lib files)
module purge
module load R/3.2.3
module load intel/2016.1.150
module load openmpi_ib/1.8.4
#the latter is only visible after intel is loaded
R
    [set the repo]
    install.packages('Rmpi', dependencies = TRUE, type = "source", configure.args = c('--with-mpi=/opt/openmpi/intel/ib'

Sometimes, you may need to provide more information about the system installed dependencies.
For example, rgdal is a R package that depends on PROJ and GDAL packages and
you may need to help it find all the pieces of the puzzle:
module purge
module load R
module load gdal
module load proj
R
    [set the repo]
    install.packages('rgdal', dependencies = TRUE, type = "source", configure.args=c('--with-gdal-config=/opt/gdal/bin/g

In rare circumstances, you may encounter a situation where the build is disallowed with an error like
package ‘INLA’ is not available (for R version 3.2.3)
It may be possible to download and build directly from a development repository.
module purge
module load R
R
    install.packages("INLA", repos="http://www.math.ntnu.no/inla/R/stable")

Some suites of extra R software have their own installer mechanism. Bioconductor is an example.
The set of default packages can be installed (and updates to dependencies) simply by

            source("http://bioconductor.org/biocLite.R")
            biocLite()

This installer will be busy for some time downloading and installing packages.
You will need to answer some questions going through:

    Would you like to use a personal library instead?  (y/n) y
    Would you like to create a personal library
    ~/R/x86_64-pc-linux-gnu-library/3.2
    to install packages into?  (y/n) y

A specific component can be installed (without updating other components) using a variation on this mechanism:
    module load R
    R
    source("http://bioconductor.org/biocLite.R")
    biocLite("DESeq", suppressUpdates=TRUE)

The ENTIRE bioconductor suite can be installed using a variation on this mechanism.
    module load R
    R
            source("http://bioconductor.org/biocLite.R")
            biocLite(all_group(), suppressUpdates=TRUE)

Warning: This all_group installation will take a VERY long time and will also download a lot of sample data.
```

Personal Perl Packages

There is a perlmonks discussion item that summarises one of several approaches.

To build a package for use from, and installation into, a personal location called $MYSHARE, (rinceWind) suggests the following:

```
perl Makefile.PL LIB=$MYSHARE/lib PREFIX=$MYSHARE
make
make test
make install
```

> Note that $MYSHARE is a shell environment variable pointing to a path to which you as installer have write access. This even means that you can set up modules without needing root or sysadmin rights.
> To use said directory, set PERL5LIB to include $MYSHARE in the path. It also makes installation possible to other nodes without invoking perl (just tar up the directory and distribute it).
>
> I tend to specify both LIB and PREFIX because for the following reasons: LIB is specified so that all .pm and XS code can be found. PREFIX is specified to provide placeholders for manpages and any other collateral that does not end up in the LIB. Note: you might get problems between perl versions if you have incompatible XS binaries.

You might consider creating the "$MYSHARE" within your home directory.

Personal Python Modules

Python distutils

Distutils (http://docs.python.org/2/install/index.html) is the standard for Python package distribution. Installing packages using distutils is a 3 step process.

1. Download the package you want to install from the package website (usually something like <name>-<version>.tar.gz)
2. Extract the package archive and move into that directory (`tar -zxf <name>-<version>.tar.gz` and then `cd <name>-<version>`)
3. Execute the python setup script as follows
   `python setup.py install --user`

This will install the package in your user site-packages directory (eg. ~/.local/lib/python2.7/site-packages).

Some things to keep in mind.

- Barrine's system python does not have distutils. To use distutils, you need to load one of the python modules.
- Distutils does not perform dependency resolution.
- Distutils does not provide any update or uninstall mechanisms.

Using pip

Pip is a tool which simplifies the process for installing packages.
It automatically downloads and installs a package and its dependencies just by providing the package name.

By using the `--user` option to pip you will be permitted to install your python "extras" into `$HOME/.local/lib/python2.7/site-packages`

The syntax for installing user packages is:

```
uqmmallo@b10b10:~> pip install --user <package_name1> <package_name2> ...
```

Pip also provides an uninstall mechanism.

```
pip uninstall <package_name1> <package_name2> ...
```

While file permissions will stop you being able to remove global packages, please take care to only remove packages that you installed.

Use the --user option to pip to avoid disappointments!

Example of Using pip (IsoCon)

IsoCon is a python package with some dependencies.
We found that it needed to have additional package (wheel) installed as the user to get past an error with the installation of parasail.
Then things went smoothly through to completion.

Here is a screen scrape of the session:

```
uqdgree5@awoonga1:.../uqdgree5/Support/uquser> module load python

uqdgree5@awoonga1:.../uqdgree5/Support/uquser> which pip
/opt/python/bin/pip
uqdgree5@awoonga1:.../uqdgree5/Support/uquser> cat requirements.txt
edlib>=1.1.2
pysam>=0.11
networkx>=1.10
parasail>=1.1.10

uqdgree5@awoonga1:.../uqdgree5/Support/uquser> /opt/python/bin/pip2.7 install  --user --upgrade pip
Collecting pip
  Using cached pip-9.0.3-py2.py3-none-any.whl
Installing collected packages: pip
Successfully installed pip-9.0.3

uqdgree5@awoonga1:.../uqdgree5/Support/uquser> /home/uqdgree5/.local/bin/pip2.7 install  --user wheel
Collecting wheel
  Downloading wheel-0.30.0-py2.py3-none-any.whl (49kB)
    100% |████████████████████████████████| 51kB 2.3MB/s
Installing collected packages: wheel
Successfully installed wheel-0.30.0

uqdgree5@awoonga1:.../uqdgree5/Support/uquser> /home/uqdgree5/.local/bin/pip2.7 install  --user parasail
Collecting parasail
```

```
  Using cached parasail-1.1.11.tar.gz
Requirement already satisfied: numpy in /opt/python/lib/python2.7/site-packages/numpy-1.12.1-py2.7-linux-x86_64.egg (from pa
Building wheels for collected packages: parasail
  Running setup.py bdist_wheel for parasail ... done
  Stored in directory: /home/uqdgree5/.cache/pip/wheels/e6/2c/d5/7175fbdcc9a8a1b7759e4bdc2cb8a31332af9a2b78416a12b3
Successfully built parasail
Installing collected packages: parasail
Successfully installed parasail-1.1.11

uqdgree5@awoonga1:.../uqdgree5/Support/uquser> /home/uqdgree5/.local/bin/pip2.7 install  --user --requirement requirements.t
Collecting IsoCon
  Using cached IsoCon-0.3.1.tar.gz
Collecting edlib>=1.1.2 (from -r requirements.txt (line 1))
  Using cached edlib-1.2.3.tar.gz
Collecting pysam>=0.11 (from -r requirements.txt (line 2))
  Using cached pysam-0.14.1-cp27-cp27m-manylinux1_x86_64.whl
Collecting networkx>=1.10 (from -r requirements.txt (line 3))
  Using cached networkx-2.1.zip
Requirement already satisfied: parasail>=1.1.10 in /gpfs1/homes/uqdgree5/.local/lib/python2.7/site-packages (from -r require
Collecting biopython>=1.66 (from IsoCon)
  Downloading biopython-1.70-cp27-cp27m-manylinux1_x86_64.whl (2.2MB)
    100% |████████████████████████████████| 2.2MB 258kB/s
Collecting decorator>=4.1.0 (from networkx>=1.10->-r requirements.txt (line 3))
  Downloading decorator-4.2.1-py2.py3-none-any.whl
Requirement already satisfied: numpy in /opt/python/lib/python2.7/site-packages/numpy-1.12.1-py2.7-linux-x86_64.egg (from pa
Building wheels for collected packages: IsoCon, edlib, networkx
  Running setup.py bdist_wheel for IsoCon ... done
  Stored in directory: /home/uqdgree5/.cache/pip/wheels/70/96/d6/aa8d054b7e07dfbecfd1bb146cadb3500e2f4ef1cef5233121
  Running setup.py bdist_wheel for edlib ... done
  Stored in directory: /home/uqdgree5/.cache/pip/wheels/5d/47/87/8c9699daa744d6a0d604d0a571d87c78b90013e40ef6c4759d
  Running setup.py bdist_wheel for networkx ... done
  Stored in directory: /home/uqdgree5/.cache/pip/wheels/a0/33/4e/7c9228ea77f8090e895d8d2b76f3b5a76997a5b3edeb4e2c6f
Successfully built IsoCon edlib networkx
Installing collected packages: edlib, pysam, decorator, networkx, biopython, IsoCon
Successfully installed IsoCon-0.3.1 biopython-1.70 decorator-4.2.1 edlib-1.2.3 networkx-2.1 pysam-0.14.1
```

Footnote: R Versions on HPC

Each HPC node has access to several versions of R. This sections will hopefully illuminate that situation.

```
---------------------------------- /opt/modulefiles/applications ----------------------------------
R/3.4.0(default)

------------------------------------ /sw/Modules/ContainedApps ------------------------------------
R/3.5.0

---------------------------------------- /sw/B18a/modulefiles ---------------------------------------
R/3.4.3
```

ROCKS R

A ROCKS rolled version of R is installed into the operating system of each compute node into `/opt/modulefiles/applications`.

Currently that is `module load R/3.4.0` and is built with, and configured to use, the Intel compiler.

```
uqdgree5@awoonga1:~> module display R/3.4.0
-------------------------------------------------------------------
/opt/modulefiles/applications/R/3.4.0:

module-whatis   R
module-whatis   Version: 3.4.0
module-whatis   Description: R
prepend-path    PATH /opt/R/bin
prepend-path    R_LIBS /opt/R/local/lib
prepend-path    LD_LIBRARY_PATH /opt/R/lib64/R/lib
setenv          RHOME /opt/R
module          load intel
-------------------------------------------------------------------
```

A well hidden secret is the existence of many R library packages for this "on board" version of R.

```
uqdgree5@awoonga1:~> ls  /opt/R/local/lib/
abind          cubature        GenomeInfoDbData   logspline     purrr          sna
acepack        curl            GenomicAlignments  magrittr      quadprog       snow
akima          DatABEL         GenomicFeatures    maps          quantreg       sp
alr3           data.table      GenomicRanges      maptools      R6             spacetime
annotate       DBI             ggplot2            markdown      randomForest   SparseM
AnnotationDbi  DelayedArray    git2r              matlab        raster         stabs
aoos           DEoptimR        glue               matrixcalc    RColorBrewer   statmod
ape            desc            gplots             MatrixModels  Rcpp           statnet.common
arm            DESeq2          graph              matrixStats   RcppArmadillo  stringi
assertthat     devtools        gridExtra          mboost        RcppEigen      stringr
backports      dichromat       gstat              mclust        RCurl          strucchange
base64         digest          gtable             mcmc          reshape2       SummarizedExperiment
```

```
base64enc      doMC            gtools             memoise       rgdal          survey
bdsmatrix      dplyr           GWAF               mi            rgenoud        survival
BH             e1071           haplo.stats        mime          rgeos          tcltk2
bindr          Ecdat           hexbin             minqa         Rgraphviz      TeachingDemos
bindrcpp       Ecfun           highr              miscTools     rhdf5          TH.data
Biobase        edgeR           Hmisc              mix           rjson          tibble
BiocGenerics   ellipse         htmlTable          mlbench       rlang          tidyr
BiocInstaller  epitools        htmltools          modeltools    rlecuyer       tidyselect
BiocParallel   evaluate        htmlwidgets        modules       rmeta          timeDate
biomaRt        exomePeak       httr               multcomp      Rmpi           tis
Biostrings     fda             hwriter            munsell       rms            tkrplot
bit            fmcsR           ineq               mvtnorm       robustbase     tripack
bit64          FNN             InteractionSet     ncdf4         ROCR           urca
bitops         foreach         intervals          network       roxygen2       VGAM
blob           foreign         IRanges            nlme          rpart.plot     viridis
bnlearn        formatR         iterators          nloptr        rprojroot      viridisLite
brew           Formula         jpeg               nnls          RSAGA          whisker
BSgenome       fts             jsonlite           numDeriv      Rsamtools      withr
car            futile.logger   kernlab            nws           RSQLite        XML
caTools        futile.options  knitr              openssl       rstudioapi     xml2
checkmate      gap             labeling           oz            rtracklayer    xtable
ChemmineR      gdata           lambda.r           party         RUnit          xts
chron          gee             lattice            pbkrtest      S4Vectors      XVector
coda           geepack         latticeExtra       PBSmodelling  sandwich       yaml
coin           GenABEL         lazyeval           pkgconfig     scales         zlibbioc
colorspace     GenABEL.data    ldlasso            plogr         scatterplot3d  zoo
combinat       genefilter      leaps              plyr          sem
commonmark     geneplotter     limma              polspline     sgeostat
coxme          genetics        lme4               PredictABEL   shapefiles
crayon         GenomeInfoDb    locfit             pspline       ShortRead
```

/sw R

Unfortunately, in some circumstances, the on-board version of R exhibited memory leaks.

A (newer) version of R was built from source and installed into /sw.

```
-------------------------------------------------------------------
/sw/B18a/modulefiles/R/3.4.3:

module-whatis    R
module-whatis    Version: 3.4.3
module-whatis    Description: R
setenv           RHOME /sw/B18a/R
module           load intel/2017.4
prepend-path     PATH /sw/B18a/R/bin
prepend-path     R_LIBS /sw/B18a/R/local/lib
prepend-path     LD_LIBRARY_PATH /sw/B18a/R/lib64/R/lib
-------------------------------------------------------------------
```

It does have most of the same extensive set of additional library packages available. You may need to build some additional packages for yourself.
See the Building Packages for R and Python User Guide for information about how to do this.

Singularity R

To rapidly respond to a request for the R 3.5.0 release we have deployed a software container which is based on a newer operating system image.
The container is not a comprehensive installation of R. It is based on a packaged version provided by a developer.

We are still awaiting the release of complete R 3.5.0 packages for an operating system that can be supported by our container mechanism (no, we cannot use the latest Arch Linux)

```
uqdgree5@awoonga1:~> module display R/3.5.0
-------------------------------------------------------------------
/sw/Modules/ContainedApps/R/3.5.0:

module-whatis
module-whatis    R V3.5.0 within a container
module-whatis    david.green@uq.edu.au
module-whatis    20180518
module-whatis    For more detail, run
module-whatis    module help R
module-whatis
module           load singularity
system           shopt -s expand_aliases
set-alias  R     /sw/Containers/singularity/bin/run_singularity run /sw/Containers/singularity/images/R-3.5.0
set-alias  shell /sw/Containers/singularity/bin/run_singularity shell /sw/Containers/singularity/images/R-3.5.0
-------------------------------------------------------------------
```

To use this version of R interactively, you just need load the module and type the command (alias) R.

To use this version of R in a regular batch job you should consult the relevant section of the Containers User Guide.