marshal: Saving and Loading Objects that Otherwise Cannot be Saved or Exported to Parallel Workers
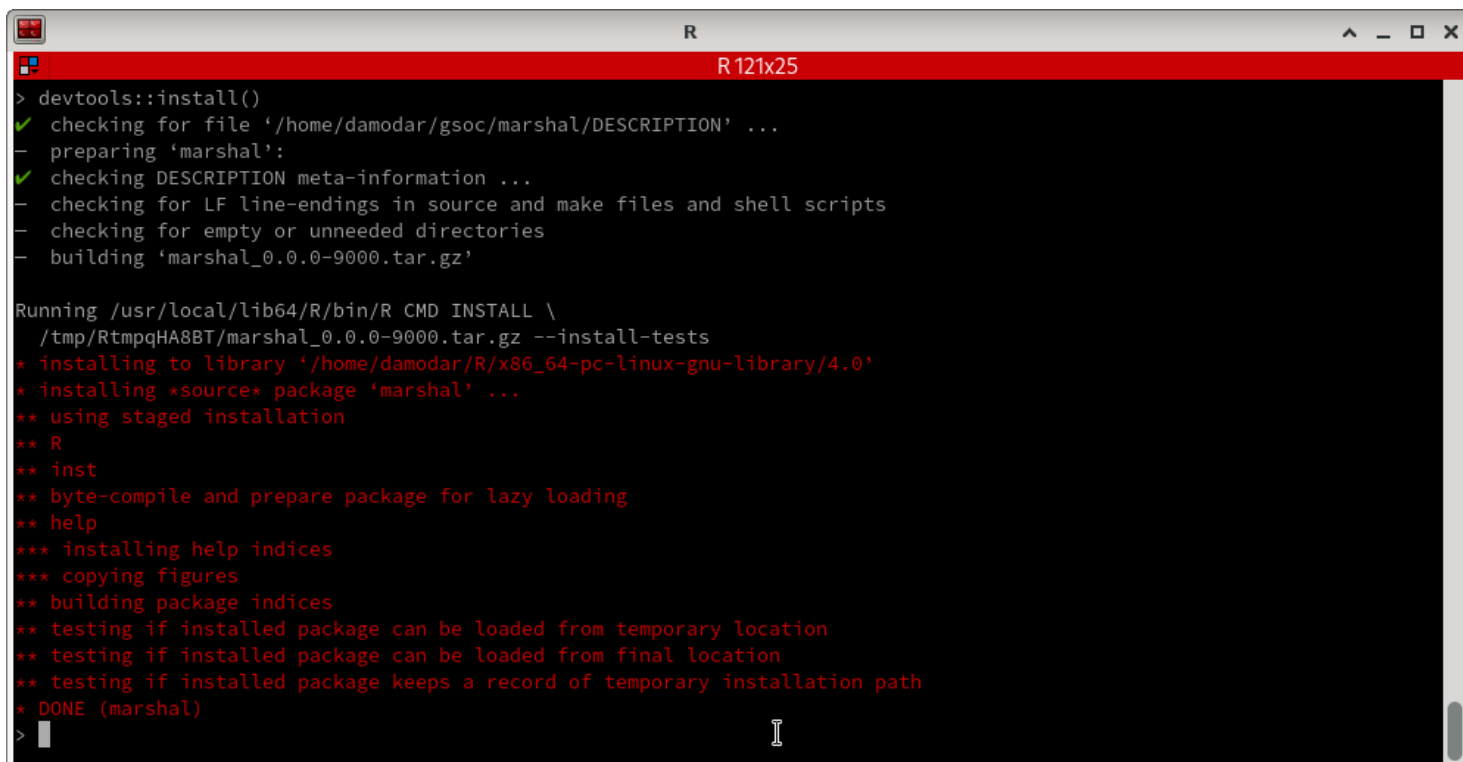
Student Name - Damodar Dhakad

1. Easy Task :Git and R package testing

Fork the marshal repository and build the package and check it with both devtools::check() and R CMD check --as-cran.

The devtools way -

Each repository has its own way to install a package from them, devtools packages helps to simplify this process. For example install_github() use to install packages from GitHub.

To install Marshal package available at [GitHub](), type devtools::install_github("HenrikBengtsson/marshal") onto R command prompt It will downlaoad the repository into temprary folder and proceed with installaation. Other way is to cd into you self git clone repository and run devtools::install() Output will look like this -
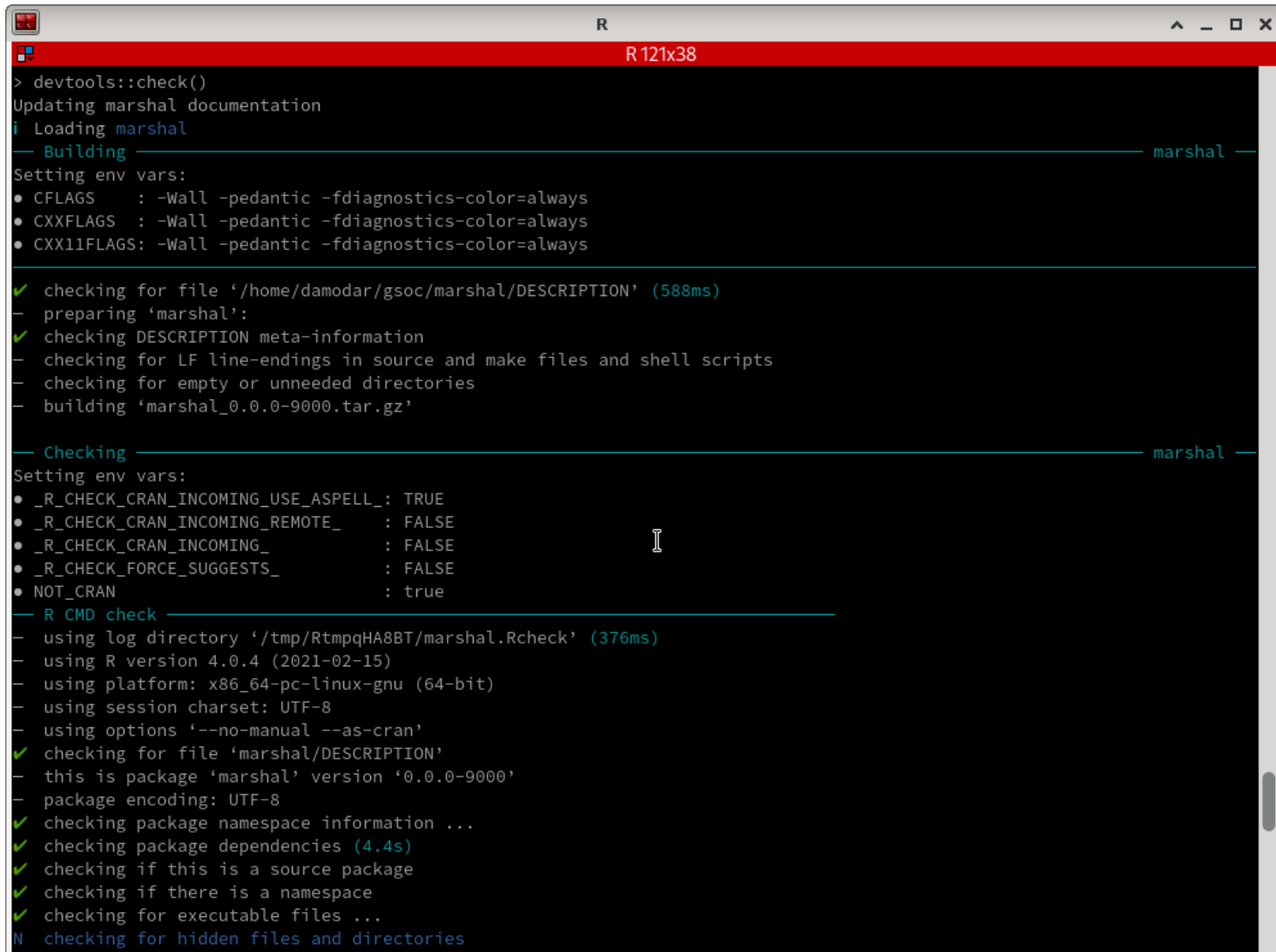
We use devtools::check() function available from devtools package to check the functionality of our package.
Output will look like this -

```
   Found the following hidden files and directories:
     .github
   These were most likely included in error. See section 'Package
   structure' in the 'Writing R Extensions' manual.
✔  checking for portable file names
✔  checking for sufficient/correct file permissions
✔  checking whether package 'marshal' can be installed (3.3s)
✔  checking installed package size ... OK
✔  checking package directory ...
✔  checking for future file timestamps (2.7s)
✔  checking DESCRIPTION meta-information (538ms)
N  checking top-level files
   Non-standard file/directory found at top level:
     'Makefile'
✔  checking for left-over files
✔  checking index information
✔  checking package subdirectories ...
✔  checking R files for non-ASCII characters ...
✔  checking R files for syntax errors ...
✔  checking whether the package can be loaded ...
✔  checking whether the package can be loaded with stated dependencies ...
✔  checking whether the package can be unloaded cleanly ...
✔  checking whether the namespace can be loaded with stated dependencies ...
✔  checking whether the namespace can be unloaded cleanly ...
✔  checking loading without being on the library search path ...
✔  checking dependencies in R code ...
✔  checking S3 generic/method consistency (1.2s)
✔  checking replacement functions ...
✔  checking foreign function calls ...
✔  checking R code for possible problems (2.6s)
✔  checking Rd files ...
✔  checking Rd metadata ...
✔  checking Rd line widths ...
✔  checking Rd cross-references (348ms)
✔  checking for missing documentation entries ...
✔  checking for code/documentation mismatches (530ms)
✔  checking Rd \usage sections (1.3s)
✔  checking Rd contents ...
```

```
✔ checking foreign function calls ...
✔ checking R code for possible problems (2.6s)
✔ checking Rd files ...
✔ checking Rd metadata ...
✔ checking Rd line widths ...
✔ checking Rd cross-references (348ms)
✔ checking for missing documentation entries ...
✔ checking for code/documentation mismatches (530ms)
✔ checking Rd \usage sections (1.3s)
✔ checking Rd contents ...
✔ checking for unstated dependencies in examples ...
✔ checking line endings in Makefiles
✔ checking for GNU extensions in Makefiles
✔ checking include directives in Makefiles
— checking examples ... NONE
✔ checking for non-standard things in the check directory
✔ checking for detritus in the temp directory

    See
      '/tmp/RtmpqHA8BT/marshal.Rcheck/00check.log'
    for details.


── R CMD check results ──────────────────────── marshal 0.0.0-9000 ──
Duration: 21s

❯ checking for hidden files and directories ... NOTE
  Found the following hidden files and directories:
    .github
  These were most likely included in error. See section 'Package
  structure' in the 'Writing R Extensions' manual.

❯ checking top-level files ... NOTE
  Non-standard file/directory found at top level:
    'Makefile'

0 errors ✔ | 0 warnings ✔ | 2 notes ✖
>
```
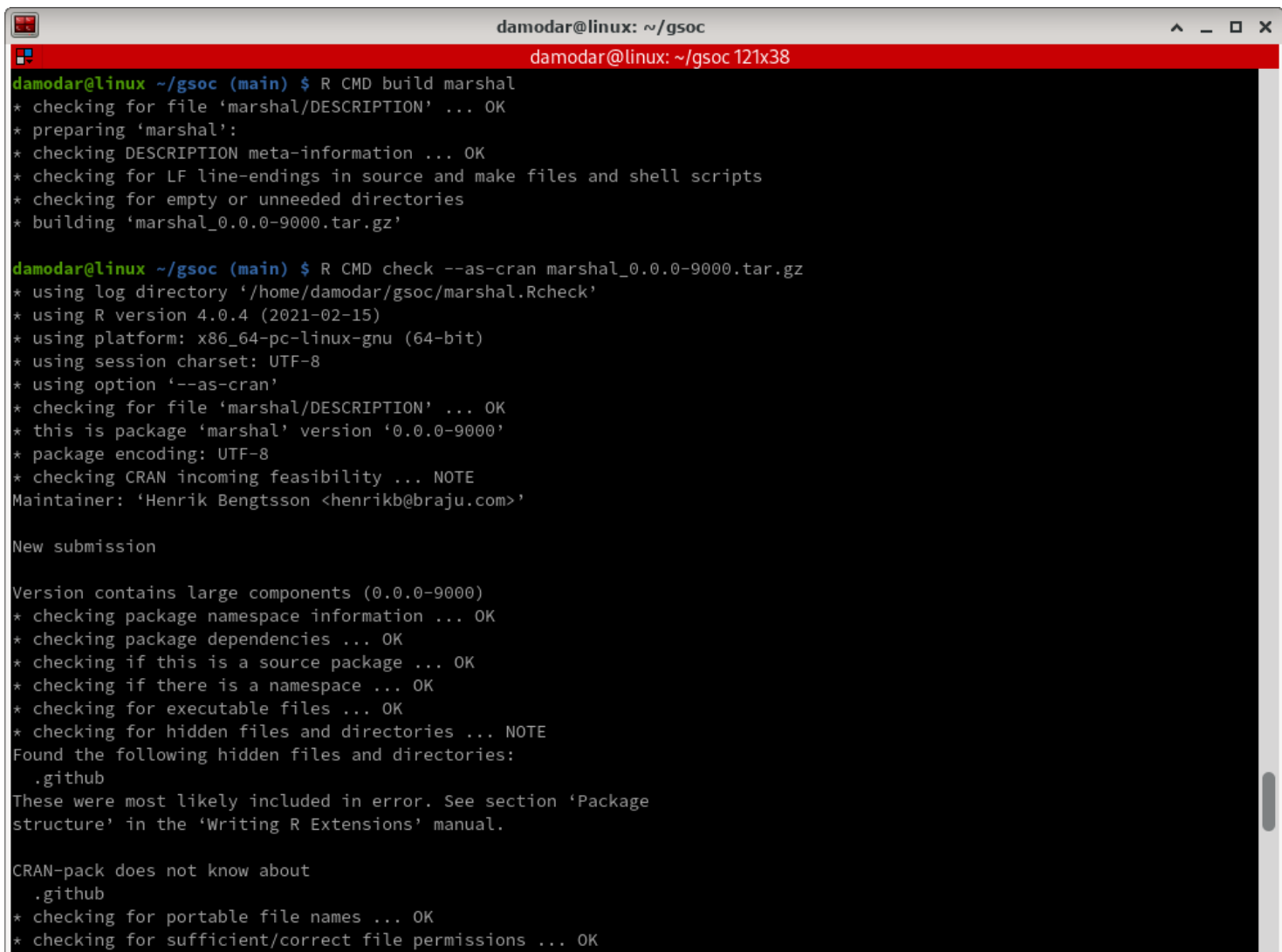
R CMD check way -

 R CMD check runs all sorts of checks on the contents of an R package, and gives
warnings and error messages when it finds things that aren't right. It also will run
the examples in the .Rd files for each of your functions, as well as other
automated tests that we've included. First we need to build package file which in
this case by running `R CMD build marshal`. And then to verify the fucntionalities
of package, run `R CMD check --as-cran marshal_0.0.0-9000.tar.gz`.

```
* checking whether package 'marshal' can be installed ... OK
* checking installed package size ... OK
* checking package directory ... OK
* checking for future file timestamps ... OK
* checking DESCRIPTION meta-information ... OK
* checking top-level files ... NOTE
Files 'README.md' or 'NEWS.md' cannot be checked without 'pandoc' being installed.
Non-standard file/directory found at top level:
  'Makefile'
* checking for left-over files ... OK
* checking index information ... OK
* checking package subdirectories ... OK
* checking R files for non-ASCII characters ... OK
* checking R files for syntax errors ... OK
* checking whether the package can be loaded ... OK
* checking whether the package can be loaded with stated dependencies ... OK
* checking whether the package can be unloaded cleanly ... OK
* checking whether the namespace can be loaded with stated dependencies ... OK
* checking whether the namespace can be unloaded cleanly ... OK
* checking loading without being on the library search path ... OK
* checking use of S3 registration ... OK
* checking dependencies in R code ... OK
* checking S3 generic/method consistency ... OK
* checking replacement functions ... OK
* checking foreign function calls ... OK
* checking R code for possible problems ... OK
* checking Rd files ... OK
* checking Rd metadata ... OK
* checking Rd line widths ... OK
* checking Rd cross-references ... OK
* checking for missing documentation entries ... OK
* checking for code/documentation mismatches ... OK
* checking Rd \usage sections ... OK
* checking Rd contents ... OK
* checking for unstated dependencies in examples ... OK
* checking line endings in Makefiles ... OK
* checking for GNU extensions in Makefiles ... OK
* checking include directives in Makefiles ... OK
```

```
                                      damodar@linux: ~/gsoc                                    ∧  _  □  ✕
                                 damodar@linux: ~/gsoc 121x38
* checking for missing documentation entries ... OK
* checking for code/documentation mismatches ... OK
* checking Rd \usage sections ... OK
* checking Rd contents ... OK
* checking for unstated dependencies in examples ... OK
* checking line endings in Makefiles ... OK
* checking for GNU extensions in Makefiles ... OK
* checking include directives in Makefiles ... OK
* checking examples ... NONE
* checking PDF version of manual ... WARNING
LaTeX errors when creating PDF version.
This typically indicates Rd problems.
LaTeX errors found:
! LaTeX Error: File `inconsolata.sty' not found.

Type X to quit or <RETURN> to proceed,
or enter new name. (Default extension: sty)

! Emergency stop.
<read *>

l.287 ^^M

!  ==> Fatal error occurred, no output PDF file produced!
* checking PDF version of manual without hyperrefs or index ... OK
* checking for non-standard things in the check directory ... NOTE
Found the following files/directories:
  'marshal-manual.tex'
* checking for detritus in the temp directory ... OK
* DONE

Status: 1 WARNING, 4 NOTEs
See
  '/home/damodar/gsoc/marshal.Rcheck/00check.log'
for details.

damodar@linux ~/gsoc (main) $
```

Reference -
https://kbroman.org/pkg_primer/pages/build.html
https://kbroman.org/pkg_primer/pages/check.html

2. Easy: Prototyping in R

2.a Write `XML_serialize_refhook()` and `XML_unserialize_refhook()` functions that are simple wrappers that calls `XML::xmlSerializeHook()` and `XML::xmlDeserializeHook()` internally such that they can be used in place of the latter two function.

2.b Write a package help page with an example illustrating how to use your new functions for saving and reading XML objects to file.

2.c Try to make the package passes `R CMD check` will all OKs.