# Contributing guide

## Installing daru development dependencies

Either nmatrix or rb-gsl are NOT NECESSARY for using daru. They are just required for an optional speed up and for running the test suite.

To install dependencies, execute the following commands:

``` bash

sudo apt-get update -qq

sudo apt-get install -y libgsl0-dev r-base r-base-dev

sudo Rscript -e "install.packages(c('Rserve','irr'),,'http://cran.us.r-project.org')"

sudo apt-get install libmagickwand-dev imagemagick

export DARU\_TEST\_NMATRIX=1 # for running nmatrix tests.

export DARU\_TEST\_GSL=1 # for running rb-GSL tests.

bundle install

```

You don't need `DARU\_TEST\_NMATRIX` or `DARU\_TEST\_GSL` if you don't want to make changes

to those parts of the code. However, they will be set in CI and will raise a test failure

if something goes wrong.

And run the test suite (should be all green with pending tests):

`bundle exec rspec`

If you have problems installing nmatrix, please consult the [nmatrix installation wiki](https://github.com/SciRuby/nmatrix/wiki/Installation) or the [mailing list](https://groups.google.com/forum/#!forum/sciruby-dev).

While preparing your pull requests, don't forget to check your code with Rubocop:

`bundle exec rubocop`

[Optional] Install all Ruby versions which Daru currently supports with `rake spec setup`.

## Basic Development Flow

1. Create a new branch with `git checkout -b <branch\_name>`.

2. Make your changes. Write tests covering every case how your feature will be used. If creating new files for tests, refer to the 'Testing' section [below](#Testing).

3. Try out these changes with `rake pry`.

4. Run the test suite with `rake spec`. (Alternatively you can use `guard` as described [here](https://github.com/SciRuby/daru/blob/master/CONTRIBUTING.md#testing). Also run Rubocop coding style guidelines with `rake cop`.

5. Commit the changes with `git commit -am "briefly describe what you did"` and submit pull request.

[Optional] You can run rspec for all Ruby versions at once with `rake spec run all`. But remember to first have all Ruby versions installed with `ruby spec setup`.

## Testing

Daru has automatic testing with Guard. Just execute the following code before you start editting a file and any change you make will trigger the appropriate tests-

```

guard

```

\*\*NOTE\*\*: Please make sure that you place test for your file at the same level and with same itermediatary directories. For example if code file lies in `lib/xyz/abc.rb` then its corresponding test should lie in `spec/xyz/abc\_spec.rb`. This is to ensure correct working of Guard.

## Daru internals

To get an overview of certain internals of daru and their implementation, go over [this blog post](http://v0dro.github.io/blog/2015/08/16/elaboration-on-certain-internals-of-daru/).