

Fundamentals of Creating R Package

Riaz Khan, MS, Statistics, SDSU

02/02/2020

Table of Contents

System preparation.....	2
Create library.....	2
Package naming guidelines	3
Package configuration.....	3
Build and check package	4
Package metadata	5
Version	6
Multiple authors	6
License.....	6
Dependencies.....	7
Package documentation.....	7
Writing own function	9
straight line 101.....	9
Function straighten.....	10
Function intersection_stline:.....	12
Coding style: best practices.....	13
S3 method: print for class stline	14
External data.....	15
Dependency.....	16
Package vignette.....	16
Package testing	17
Test Exmaple.....	17
Package release.....	19
README file	19
NEWS file.....	19
Release.....	20

System preparation

- Install the latest version of R and RStudio:
 - R (3.6.2): <https://cran.r-project.org/>
 - RStudio (1.2.5033):
<https://rstudio.com/products/rstudio/download/#download>
- Install the following packages:
 - devtools
 - roxygen2
 - testthat
 - knitr

```
install.packages("devtools")
install.packages("roxygen2")
install.packages("testthat")
install.packages("knitr")
```

- Get updated version of devtools from GitHub:

```
devtools::install_github("r-lib/devtools")
```

- Check if the system is ready for package building:

```
devtools::has_devel()
```

If following message is shown, it suggests that the system is ready for package development. The C compiler installation part can be skipped.

Your system is ready to build packages!

Depending on the system, a C compiler may be required. If required:

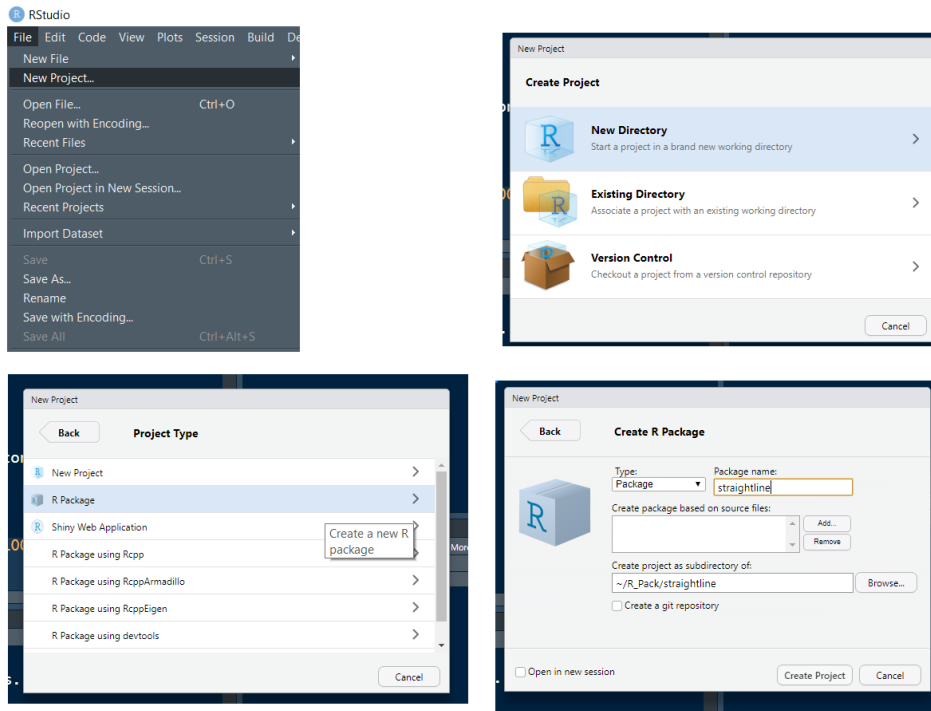
- For windows, install Rtools <https://cran.r-project.org/bin/windows/Rtools/>
- For Mac, get Xcode (available for free in the App Store)

Create library

To create a new package, open RStudio

- File >> New Project >> New Directory >> R Package

Give the name of the package straightline (or anything you like...)

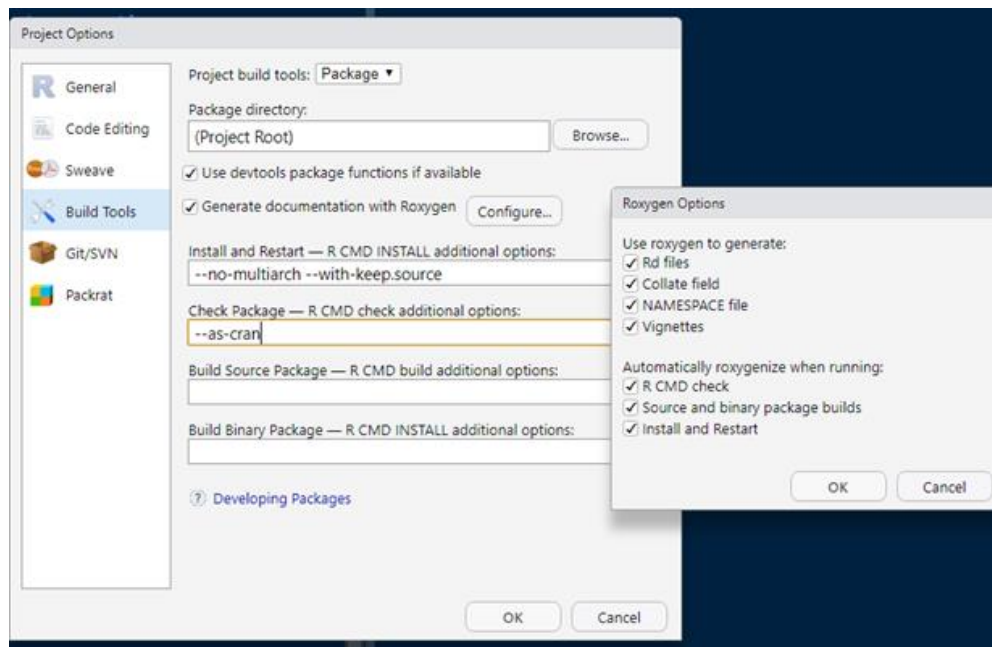


Package naming guidelines

- Unique name
- Matches with the type of problem the library deals with
 - Easy to google
 - Example neuralnet, NLP
- Case consistency
 - Same case preferred
 - Mixing not uncommon (randomForest)
- It is not uncommon to see an extra r in the name
 - Example: stringr, knitr
 - straightliner? Instead of straightline
- Formal requirements for an R package:
 - Contains only alphabetic letter, number and period
 - Cannot end with a period
 - Can start with only letter

Package configuration

To let roxygen to some heavy lifting, go to Build > Configure Build Tools and configure as follows:



Build and check package

RStudio includes a hello function with documentation in a default package when a package is created. To build the package, Build > Install and Restart (Ctrl + Shift + B). This will install the package in the system and will be included in R session.

After building, check the package, Build > Check Package (Ctrl + Shift + E). This command will check the package using best practices.

If everything left unaltered, the check should return with a warning for non-standard license.

```
-- R CMD check results -----
Duration: 22.8s

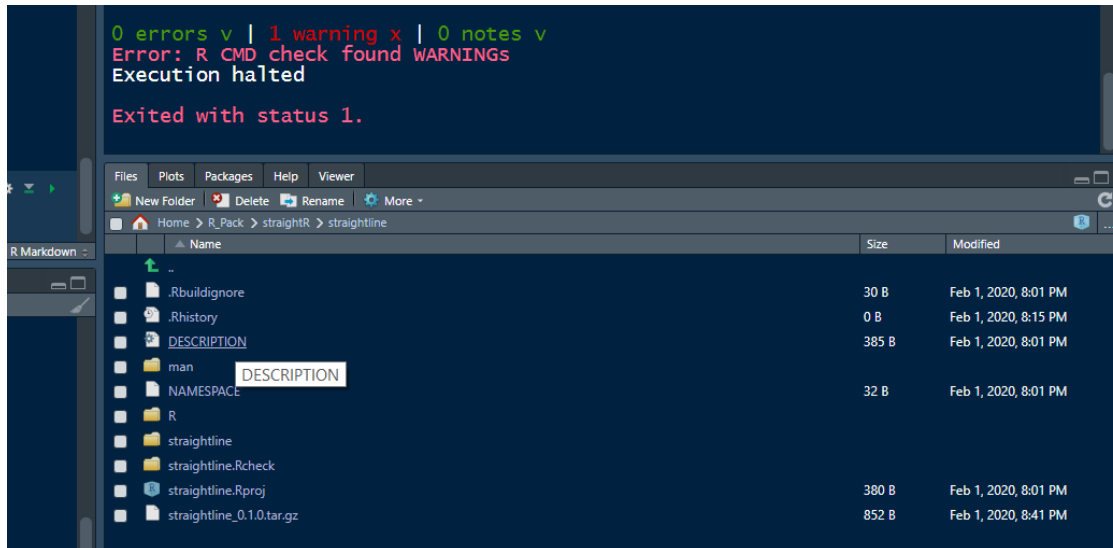
> checking DESCRIPTION meta-information ... WARNING
Non-standard license specification:
  what license is it under?
Standardizable: FALSE

0 errors v | 1 warning x | 0 notes v
Error: R CMD check found WARNINGS
Execution halted

Exited with status 1.
```

Package metadata

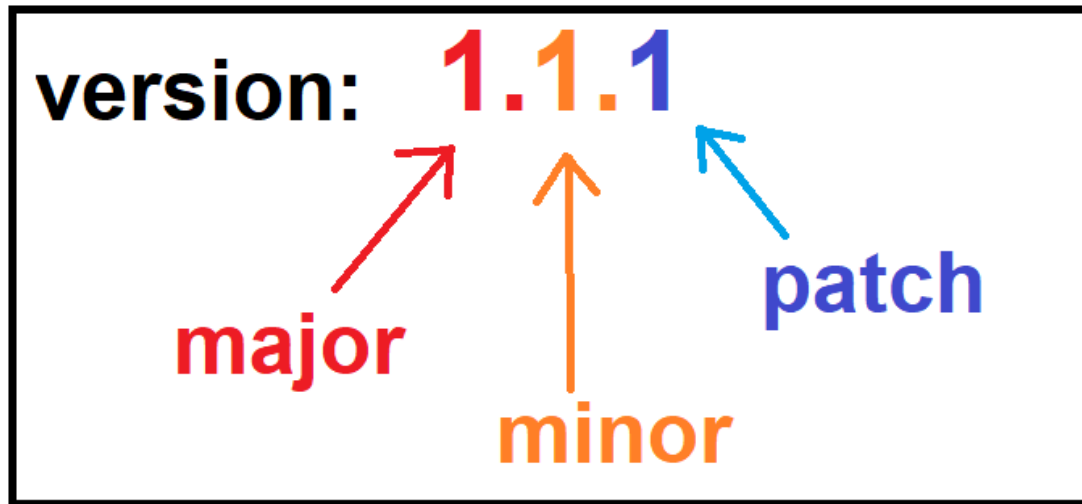
- Every package must have a description
- There will be an auto-generated DESCRIPTION file
- Using the file explorer-plot window of RStudio, go to the package folder and double click the DESCRIPTION file. This will open the file in RStudio editor window.



Edit Title, Version, Author, Maintainer, Description and License in the DESCRIPTION file. An example:

```
Package: straightline
Type: Package
Title: An Example R Package for Straight Line in 2d Coordinate System
Version: 1.1.1
Author: Riaz Khan
Maintainer: Riaz Khan <rk@sdsu.net>
Description: This package will do basic calculations of a straight
  line in 2D coordinate system. This package will do basic
  calculations of a straight line in 2D coordinate system. This
  package will do basic calculations of a straight line in 2D
  coordinate system.
License: GPL-3
Encoding: UTF-8
LazyData: true
RoxygenNote: 7.0.2
```

Version



Multiple authors

If there are multiple authors/contributors, then Authors@R field can be used:

```
Authors@R: c(person("Riaz", "Khan",  
  email = "rk@sdsu.net",  
  role = c("aut", "cre")),  
  person("Second", "Author",  
    email = "se@sdsu.net",  
    role = "aut" ))
```

Following roles can be added in the Authors@R filed:

- cre: maintainer
- aut: author having significant contributions
- ctb: contributors with smaller contributions
- cph: copyright holder, in case where the copyright holder is other than the author (e.g employer of the author)

License

There are three that can be used in an R package:

- MIT: Simple permissive. Anyone can do anything as long as the original copyright and license notice included. See <https://tldrlegal.com/license/mit-license>

- GPL -2, GPL -3: If the code is distributed in a bundle, then the whole bundle must be in GPL license. In case of code modification, the source code must be available. GPL-3 is little stricter than GPL-2. See <https://tldrlegal.com/license/gnu-general-public-license-v2>, [https://tldrlegal.com/license/gnu-general-public-license-v3-\(gpl-3\)](https://tldrlegal.com/license/gnu-general-public-license-v3-(gpl-3))

Dependencies

Frequently, a new package uses the functionality of other packages. For this package dependency, Imports field must be used. If a package is used, but not required (likely for examples), then those should be declared as Suggests. For example:

```
Package: straightline
Type: Package
...
...
Imports: ggplot2
Suggests: MASS
```

Package documentation

So far, we have only one function in our package, called hello. Let us see the documentation of this function.

```
?hello
```

A description, usage and example is given in the function Help section. This documentation is created by the default Rd file. Go to ~/man/hello.Rd to see the actual codes that generated this documentation.



Note:

- Documentation and R files live in separate folders

- ~/man and ~/R folders respectively
 - R and Rd documents need to be written separately
 - Roxygen- codes and documentation simultaneously
 - Roxygen auto generates the Rd files
 - When we configured at the beginning, we told roxygen to handle the Rd files along with other things
-

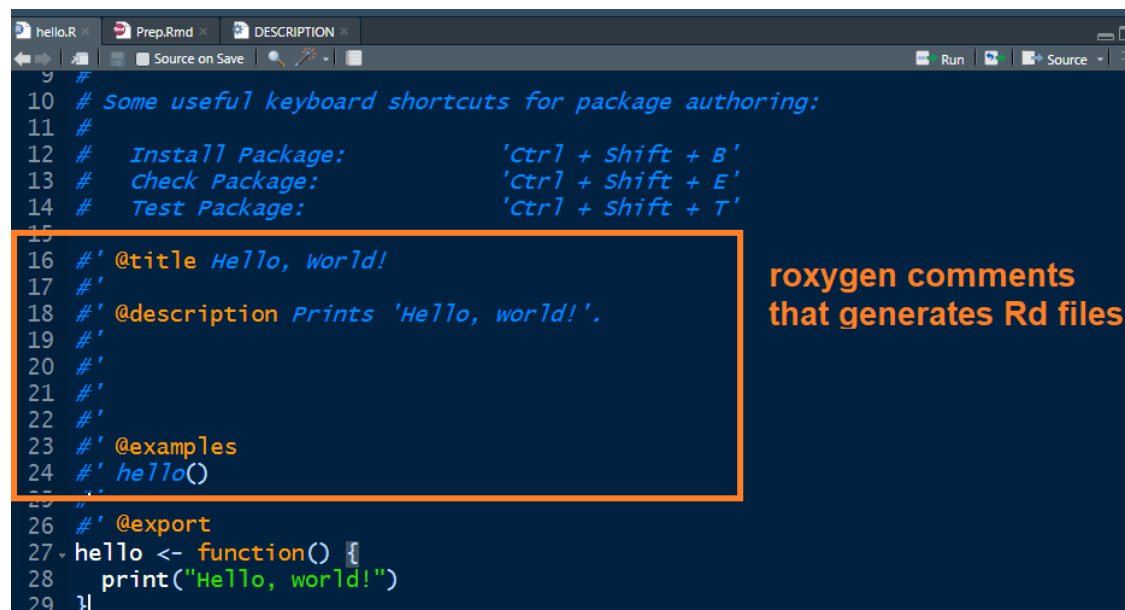
Go to the manual directory and delete the `hello.Rd` file. Then build the package (`ctrl+Shift+B`) and run the following again.

```
?hello
```

Now there is no documentation for the `hello` function. Now we will try to re-create the documentation.

- Go to the `hello.R`, put the cursor on `hello` where the function is defined. Then click `Code > Insert Roxygen Skeleton`. This will create some roxygen style field for function documentation. These fields can be used for function documentation.

Note: The title can be documented using the `@title` tag.



```
10 # Some useful keyboard shortcuts for package authoring:
11 #
12 # Install Package:      'ctrl + shift + B'
13 # Check Package:       'ctrl + shift + E'
14 # Test Package:        'ctrl + shift + T'
15
16 #' @title Hello, world!
17 #'
18 #' @description Prints 'Hello, world!'.
19 #'
20 #'
21 #'
22 #'
23 #' @examples
24 #' hello()
25 #'
26 #' @export
27 hello <- function() {
28   print("Hello, world!")
29 }
```

roxygen comments
that generates Rd files

Now, build the package again. Examine the Rd file that is generated by roxygen. Now, documentation will be available for the `hello` function.

Note: It is recommended that we check the package periodically (`ctrl+Shift+E`). If checking gives warning that the NAMESPACE will not be overwritten, then delete the

NAMESPACE and build+check again. Roxygen generates the NAMESPACE and the Rd files each time we build package.

Writing own function

Functions are main drivers of specific tasks that the package is intended for. Now we will make our own function and document it.

We will write a function (let us call it `straighten`) that

Takes three inputs a , b , and c , which are the parameters of a straight line of form $ax + by + c = 0$ in 2D coordinate system. The function will throw error message if a and b both are zero. We want the function to return the followings:

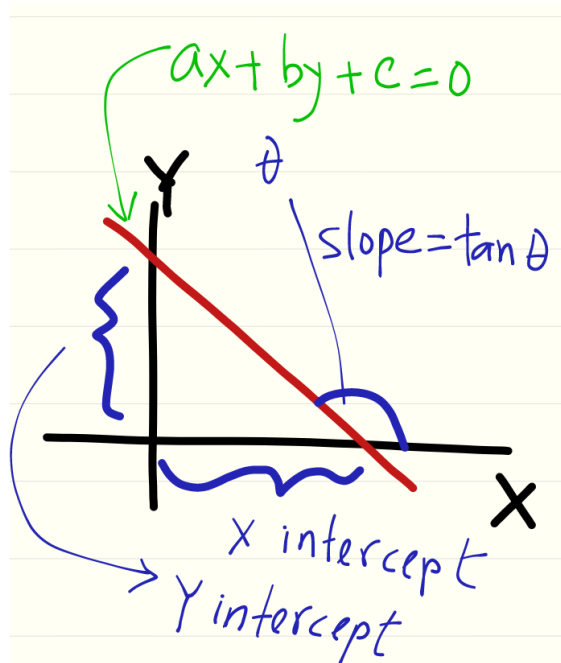
- the value of a
- the value of b
- the value of c
- the slope of the straight line
- x intercept of the straight line
- y intercept of the straight line
- the angle between x axis and the straight line in degrees

Additionally, we want the return object of the function of its own class, `stline`

straight line 101

In 2D coordinate system, if a straight line has the form of $ax + by + c = 0$, then:

- $slope = -a/b$
- $x\ intercept = -c/a$
- $y\ intercept = -c/b$
- $angle\ with\ x\ axis = \tan^{-1}(slope)$
- line parallel to x axis, if $a = 0$
- line parallel to y axis, if $b = 0$
- passes through the origin, if $c = 0$



Function straighten

- Function in ~/R folder
- file: straighten.R

```
straighten <- function(a, b, c){
  if(a==0 & b==0){
    # error message
    stop("a and b both cannot be zero")
  }
  # calculations
  slope <- - a / b
  x_intercept <- - c / a
  y_intercept <- - c / b
  angle_x <- atan(slope) * (180 / pi)
  # outputs in a list
  value <- as.list(c(a = a, b = b, c = c,
    slope = slope,
    x_intercept = x_intercept,
    y_intercept = y_intercept,
    angle_x = angle_x))
  # output class
  class(value) <- "stline"
  # return output
  return(value)
}
```

Add roxygen comments to document the function.

```
1
2 #' @title Straight Line Measures
3 #'
4 #' @param a Parameter  $\text{eqn}\{a\}$ 
5 #' @param b Parameter  $\text{eqn}\{b\}$ 
6 #' @param c Parameter  $\text{eqn}\{c\}$ 
7 #'
8 #' @description The function calculates slope, x intercept, y intercept
9 #' and angle with x axis of a straight line
10 #'
11 #' @details If a straight line is given by  $\text{eqn}\{ax+by+c=0\}$ 
12 #' then slope, x intercept, y intercept and the angle made
13 #' by the line with x axis are given by  $\text{eqn}\{-a/b\}$ ,  $\text{eqn}\{-c/a\}$ ,
14 #'  $\text{eqn}\{-c/b\}$  and  $\text{eqn}\{\arctan(\text{slope})\}$ 
15 #'
16 #' @return A list of class "stline", with all the input parameters
17 #' and measures described in details.
18 #'
19 #' @author Jon Smith
20 #'
21 #' @seealso \link{hello}
22 #'
23 #' @references url\{https://github.com/riazakhan94/straightline\}
24 #'
25 #' @source url\{https://github.com/riazakhan94/straightline\}
26 #'
27 #' @export
28 #'
29 #' @examples
30 #' line <- straighten(-4,4,30)
31 #' class(line)
32 #' message("The slope of the line is ", line$slope)
33 #'
```

straighten (straightline) R Documentation

Straight Line Measures

Description

The function calculates slope, x intercept, y intercept and angle with x axis of a straight line

Usage

straighten(a, b, c)

Arguments

a

Parameter a

b

Parameter b

c

Parameter c

Details

If a straight line is given by $ax+by+c=0$ then slope, x intercept, y intercept and the angle made by the line with x axis are given by $-a/b$, $-c/a$, $-c/b$ and $\arctan(\text{slope})$

Value

A list of class "stline", with all the input parameters and measures described in details.

Author(s)

Jon Smith

Source

<https://github.com/riazakhan94/straightline>

References

<https://github.com/riazakhan94/straightline>

See Also

[hello](#)

Examples

line <- straighten(-4,4,30)
class(line)
message("The slope of the line is ", line\$slope)

[Package straightline version 1.1.1 | Index]

Function `intersect_stline`:

```
intersect_stline <- function(x, y){  
  if(class(x) != "stline"){  
    stop("x must be of class 'stline'")  
  }  
  if(class(y) != "stline"){  
    stop("y must be of class 'stline'")  
  }  
  
  A <- matrix(c(x$a, y$a, x$b, y$b), nrow = 2)  
  determinant_A <- det(A)  
  
  if(determinant_A == 0){  
    stop("The straightlines are either parallel or identical")  
  }  
  
  b <- matrix(c(-x$c, -y$c), nrow = 2)  
  
  solution <- solve(A, b)  
  
  solution <- c(x_intersect = solution[1,1],  
               y_intersect = solution[2,1])  
  
  return(solution)  
}
```

Create documentation for the `intersect_stline` function.

R: Intersection of Two Straightlines ▾ Find in Topic

intersect_stline {straightline} R Documentation

Intersection of Two Straightlines

Description

A system of linear equation can be solved using Cramer's rule. If the system is described by $Ax=b$, then no solution exists if A is singular.

Usage

```
intersect_stline(x, y)
```

Arguments

x An object of class `stline`

y An object of class `stline`

Value

A numeric object of length 2, indicating the x coordinate and the y coordinate of the intersecting point.

See Also

[straighten](#)

Examples

```
line_1 <- straighten(1, 1, -1)
line_2 <- straighten(1, -1, 0)
# intersect of line_1 and line_2
print(intersect_stline(line_1, line_2))
```

[Package *straightline* version 1.1.1 [Index](#)]

Coding style: best practices

- Lower case recommended for function and object, with efficient use of underscore (`_`)
- There should be spaces before and after all infix operators (e.g: `+`, `-`, `*`, `/`, `=`, `==`, `<-`, `->`)
 - Space after comma, not before
- Use of `<-` for assignment (not `=`)
- Spelling out `TRUE` and `FALSE`, as opposed to just using `T` and `F`
- Opening curly brace followed by a new line
- Closing curly brace gets its own line
 - Exception: use of `else`
- Limited number of characters per line
- Use of unique name to avoid conflicts
- Proper indentations for easy understanding

For indentation, select the code block, then `Code > Reindent Lines`

S3 method: print for class stline

It is recommended that print method can be applied to an object of user defined class. Here we will define the print function applicable for stline object.

```
#'
#' @title Print \code{"stline"} Object
#'
#' @param object An object of class \code{"stline"}
#'
#' @param ... NULL; supplied for S3 method consistency
#'
#' @examples
#' line <- straighten(1, 0, 5)
#' print(line)
#'
#' line2 <- straighten(7, -5, 0)
#' print(line2)
#'
#' @method print stline
#'
#' @export
print.stline <- function(x, ... = NULL){
  char_a <- paste0("(", as.character(round(x$a, 3)), ")", "*x")
  char_b <- paste0("(", as.character(round(x$b, 3)), ")", "*y")
  char_c <- paste0("(", as.character(round(x$c, 3)), ")")
  Equation <- paste(char_a, "+", char_b, "+", char_c)
  message("Straight Line Equation      : ", Equation)
  message("Slope                        : ", x$slope)
  message("x intercept                       : ", x$x_intercept)
  message("y intercept                       : ", x$y_intercept)
  message("Angle with x axis (degrees) : ", x$angle_x)
  if(x$a == 0){
    message("Line parallel to x axis")
  }
  if(x$b == 0){
    message("Line parallel to y axis")
  }
  if(x$c == 0){
    message("Line passes through origin")
  }
}
```

External data

- External data lives in the ~/data folder, with extension .RData
- Data can be saved in .RData format using save
- Example

```
set.seed(10)
a <- runif(100, 0, 100)
set.seed(20)
b <- runif(100, 0, 100)
set.seed(30)
c <- runif(100, 0, 100)
example_data <- data.frame(cbind(a = a, b = b, c = c))
save(example_data, file = "example_data.RData")
```

- Create a folder called data and keep a copy of the .RData file in it
- Create documentation of the data

```
#' @title \code{"example_data"}: An Example Data
#'
#' @description A data containing information about 900 borrowers. It is a
#' modified version of publicly available real data.
#'
#'
#'
#' @format A data frame with 100 rows and 3 variables:
#' \describe{
#'   \item{a}{A series of parameter  $a$ }
#'   \item{b}{A series of parameter  $b$ }
#'   \item{c}{A series of parameter  $c$ }
#' }
#'
#' @examples
#' data("example_data")
#' plot(example_data$a~example_data$b)
#' grid()
#'
#'
#' @source \url{https://github.com/riazakhan94/straightline}
"example_data"
```

Note: Adding .RData may require to add dependency. If so, edits need to be made in the DESCRIPTION file (Depends: R (>= 3.5.0)).

Dependency

```
#' @title Dependency Example
#'#' @param x A numeric data series
#'#' @description This is an exmple to show the use of dependencies
#'#'#' @examples
#'#' df <- MASS::Boston
#'#' age <- df$age
#'#' example_dependency(age)
#'#'#'#' @export
example_dependency <- function(x){
  ggplot2::qplot(x, geom = "histogram", binwidth = 10)
}
```

Package vignette

- Inclusion of package vignette recommended.
 - Long guide of the package
 - What is does
 - How to use
 - Markdown file
3. In the vignettes folder
- To create:
 - create a folder called vignette and create an Rmd file (for example, stline.Rmd)
 - add knitr and rmarkdown to the Suggests field in DESCRIPTION
 - add VignetteBuilder field in the DESCRIPTION

```
Suggests: MASS, knitr, rmarkdown
VignetteBuilder: knitr
```

Add the following at the top of the vignette, before starting the document:


```

1 |---
2 title: "An Example R Package for Straight Line in 2d Coordinate System"
3 author: "Jon Smith"
4 date: "`r Sys.Date()`"
5 output: rmarkdown::html_vignette
6 fig_caption: yes
7 vignette: >
8   %\VignetteIndexEntry{An Example R Package for Straight Line in 2d
   Coordinate System}
9   %\VignetteEngine{knitr::rmarkdown}
10  %\VignetteEncoding{UTF-8}
11 |---
12
13 {r setup, include = FALSE}
14 knitr::opts_chunk$set(
15   collapse = TRUE,
16   comment = "#>"
17 )
18
19
20

```

See [RStudio guide of R Markdown](#)

Package testing

Automated tests are useful for robust code, provides all the tests in an organized way. To set up:

- Make a `~/tests/testthat` folder
- Add `testthat` to the `Suggests` field of `DESCRIPTION`
- Make a `~/tests/testthat.R` file

A test lives in the `~/tests/testthat` directory. All files must start with `test`. Tests in `~/tests/testthat.R` file are checked during the package check.

Test Exmample

- Create `~/tests/testthat/test_classtype.R` file:

```

# file: test_classtype.R
context("Class of function straighten")
library(straightline)

myline <- straighten(pi, exp(1), log(7))

test_that("Class of the function return is stline", {
  expect_equal(class(myline), "stline")
})

```

```
# direct access
test_that("Class of the function return is stline", {
  expect_equal(class(straighten(1,1,0)), "stline")
  expect_equal(class(straighten(2,15,7)), "stline")
  expect_equal(class(straighten(0,1,5)), "stline")
})
```

- Create ~/tests/testthat/test_anlge.R file:

```
# file: test_anlge.R
context("Angle of a straight line")
library(straightline)

test_that("Angle of a straight line 45/-45", {
  expect_equal(straighten(1, -1, 0)$angle_x, 45)
  expect_equal(straighten(1, 1, 0)$angle_x, -45)
})

test_that("Angle of a straight line 60/-60", {
  expect_equal(straighten(1, 1/sqrt(3), 0)$angle_x, -60)
  expect_equal(straighten(1, -1/sqrt(3), 0)$angle_x, 60)
})

test_that("Angle of a straight line 30/-30", {
  expect_equal(straighten(1, sqrt(3), 0)$angle_x, -30)
  expect_equal(straighten(1, -sqrt(3), 0)$angle_x, 30)
})
```

- Create ~/tests/testthat/test_error.R file:

```
# file: test_error.R

context("Error message if both a = b = 0")
library(straightline)

test_that("a, b both cannot be zero", {
  expect_error(straighten(0, 0, 3), "a and b both cannot be zero")
})
```

- More test functions:
 - expect_length
 - expect_message
 - expect_match
 - expect_null
 - See [testthat documentation](#)

- Add all the tests in `~/tests/testthat.R` file:
 - the library needs to be added once for all the tests

After writing all the tests in `~/tests/testthat` directory, run all the tests by running `devtools::test()`, or use `Ctrl+Shift+T`.

Check the package (`Ctrl+Shift+E`)

Package release

If checking returns 0 errors v | 0 warnings v | 0 notes then we are ready for submission

README file

- README is usually aimed for the new users.
- README should contain some high level information about the package and how to use the package.
- README has extension `.md`.
- Possible to use R Markdown (`.Rmd`) to create the `.md` file.
- Example using `.Rmd`
 - Copy the contents from the `README.Rmd` file and run. It will create the `README.md` file. Include the `README.md` file in the package directory.

NEWS file

- Aimed for the old users.
- Brief description of changes made in the new version
- An example `NEWS.md` for a new package

```
straightline 1.1.1
=====

Author
=====

Riaz Khan `rk@sdsu.net`
Second Autor `se@sdsu.net`

Maintainer
=====
Riaz Khan `rk@sdsu.net`
```

Release

- Run `devtools::release()`
- Iterative process

Go through all the process.

Congratulations! :)

Reference: [R packages by Hadley Wickham](#)