The structure of an R package

An R package can include:

- Functions
- Data
- Documentation
- Vignettes
- Tests

As a minimum your package must include:

- R directory
- man directory
- NAMESPACE file
- DESCRIPTION file

Core devtools functions:

- `create()`
- `document()`
- `check()`
- `build()`
- `test()`

create()

```
library(devtools)
create("simutils")
```

*Note: Avoid names already taken on **CRAN**.*

The create() function will create the basic package structure for us, generating a

* NAMESPACE file,
* DESCRIPTION file
* the R directory

It doesn't create the man directory yet - you'll see that happens later.

In this example, we will create a package called "simutils" in a new folder in the current working directory.

Take care when naming your package, in particular avoid names already taken on CRAN.

Example DESCRIPTION file

```
Package: simutils
Title: Simulation Analysis Tools
Version: 1.0.0.0
Authors@R: c(
    person("Nic", "Crane", email = "ncrane@mango-solutions.com", role = c("aut", "cre")),
    person("Aimee", "Gott", email = "agott@mango-solutions.com", role = c("aut", "ctb")))
Description: A series of tools for simulation analysis used for learning about
    distributions.
Depends:
    R (>= 3.4.2)
License: GPL-2
LazyData: true
RoxygenNote: 6.0.1
Imports: dplyr,
    purrr
Suggests: testthat,
    knitr,
    rmarkdown
VignetteBuilder: knitr
```

"cre" for the package maintainer,
"aut" for an author,
"ctb" for contributors
"cph" for the package copyright holder.

Optional directories

We can also include:

- Data
- Vignettes
- Tests
- Compiled code
- Translations
- Demos

```
use_data(sim_dat, pkg = "simutils")
```

```
use_vignette("my_first_vignette", pkg = "simutils")
```

Guidelines for the R directory:

- No subdirectories
- Don't have everything in one script
- Don't have a large number of small files
- Group similar functions together

Help files

sample_from_data {simutils} R Documentation

Sample from data

Description

Samples rows from a dataset.

Usage

```
sample_from_data(data, size, replace = TRUE)
```

Arguments

`data`	A data frame or matrix from which rows are to be sampled
`size`	Numeric. Number of rows to return
`replace`	Logical. Sample with replacement? TRUE by default.

Details

This function has been designed to sample from the rows of a two dimensional data set, returning all columns of sampled rows. Sampling is done with replacement by default.

Value

A data set of the same type as input with `size` rows.

Author(s)

Nic Crane

Introduction to roxygen2

roxygen headers

```
#' Sample from data
#'
#' Samples rows from a dataset.
#'
#' This function has been designed to sample from the rows of a two dimensional
#' data set, returning all columns of sampled rows.  Sampling is done with replacement
#' by default.
#'
#' @param data A data frame or matrix from which rows are to be sampled
#' @param size Numeric. Number of rows to return
#' @param replace Logical. Sample with replacement? TRUE by default.
#'
#' @author Nic Crane
#'
#' @import dplyr
#'
#' @return A data set of the same type as input with \code{size} rows.
#' @export
#'
#' @examples
#' sample_from_data(airquality, size=10)
sample_from_data <- function(data, size, replace=TRUE) {

  if(!is.numeric(size)){
    stop("size must be a numeric value")
  }

  if(is.matrix(data)){
    data = as.data.frame(data)
  }
```

Short Description

Description

Detail after params

or @importFrom tidyr gather

1. For code you use \code{text to format}

2. To link to other functions you use \link[packageName]{functionName}, although note the package name is only required if the function is not in your package

3. To include an unordered list you use \itemize{}. Inside the brackets you mark new items with \item followed by the item text.

@author to identify who wrote the function.

@seealso to list other functions that may be of interest to users.

@notes to add any other notes relating to the function (e.g. if its experimental, likely to change etc.)

https://github.com/davidgohel/flextable/blob/master/R/themes.R

```
#' @export
#' @title Apply booktabs theme
#' @description Apply theme booktabs to a flextable
#' @param x a flextable object
#' @param bold_header header will be bold if TRUE.
#' @param ... unused
#' @family functions related to themes
#' @inheritSection theme_vanilla behavior
#' @examples
#' ft <- flextable(head(airquality))
#' ft <- theme_booktabs(ft)
#' ft
#' @section Illustrations:
#'
#' \if{html}{\figure{fig_theme_booktabs_1.png}{options: width="400"}}
```

https://github.com/tidyverse/tidyr/blob/main/R/pivot.R

```
#' @description
#' `r lifecycle::badge("superseded")`
#'
#' Development on `gather()` is complete, and for new code we recommend
#' switching to `pivot_longer()`, which is easier to use, more featureful, and
#' still under active development.
#' `df %>% gather("key", "value", x, y, z)` is equivalent to
#' `df %>% pivot_longer(c(x, y, z), names_to = "key", values_to = "value")`
#'
#' See more details in `vignette("pivot")`.
"'
```

Non-running examples

```
#' Count NAs in a vector
#'
#' @param x A vector
#'
#' @return Number of NAs in x
#'
#' @examples
#' \dontrun{
#'   sum_na(airquality$Ozone)
#' }
sum_na <- function(x) {
  sum(is.na(x))
}
```

Calling non-exported functions

```
simutils:::sum_na(airquality$Ozone)
```

Package level documentation

```
#' simutils: A package for performing common simulation tasks
#'
#' This package provides functionality for a variety of simulation tasks,
#' and plotting tools for viewing the results.
#'
#' @author Nic Crane \email{ncrane@mango-solutions.com}
#' @docType package
#' @name simutils
"_PACKAGE"
```

You'd save it in a file called "simutils.R" in the R directory

Documenting data objects

```
use_data(sim_dat, pkg = "simutils")
```

```
#' sim_dat data set
#'
#' We made some data for the package
#'
#' @format A data.frame with 3 columns
#' \describe{
#' \item{ID}{ID value}
#' \item{Value}{Measured value in pounds}
#' \item{Apples}{Logical. Do they like apples}
#' }
#' @source Simulated Data
#'
"sim_dat"
```

we can save as airquality.R or data.R

Minimum level of documentation

For each function, document:

- Title
- Description
- Arguments
- Exported (for exported functions only)

Creating man files

```
document("simutils")
```

Why check an R package?

DEVELOPING R PACKAGES

Aimée Gott

Education Practice Lead, Mango Solutions

What should you check?

An R package check includes:

- If the package can be installed
- Description information is correct
- Dependencies
- Code syntax errors
- Documentation is complete
- Tests run
- Vignettes build

Running a check

```
check("simutils")
```

Note: To remove this warning, you'll need to update the documentation for the parameter in the function's .R file, and then run check() again. It automatically runs document() for you before completing its checks.

Missing information

```
...
* DONE

Status: 1 ERROR
See
  'C:/Users/ncrane/AppData/Local/Temp/RtmpWwK0AG/simutils.R
check/00check.log'
for details.

R CMD check results
1 error  | 0 warnings | 0 notes
checking for file 'simutils/DESCRIPTION' ... ERROR
Required field missing or empty:
  'License'
```

Package dependencies not instaled

```
check("simutils")
```

```
...
Updating simutils documentation
Loading simutils
Error in (function (dep_name, dep_ver = NA, dep_compare = NA)  :
 Dependency package sasMap not available.
...
```

Package or Argument Documentation Error

```
check("simutils")
```

```
...
* checking Rd \usage sections ... WARNING
Undocumented arguments in documentation object 'sample_from_data'
  'replace'

Functions with \usage entries need to have the appropriate \alias
entries, and all their arguments documented.
The \usage entries must correspond to syntactically valid R code.
See chapter 'Writing R documentation files' in the 'Writing R
Extensions' manual.
...
```

Note: you wouldn't normally get this error for non-exported functions.

Error running examples

```
check("simutils")
```

```
...
* checking examples ... ERROR
Running examples in 'simutils-Ex.R' failed
The error most likely occurred in:

> base::assign(".ptime", proc.time(), pos = "CheckExEnv")
> ### Name: sample_from_data
> ### Title: Sample from data
> ### Aliases: sample_from_data
>
> ### ** Examples
>
> sample_from_data(airuality, size=10)
Error in sample_from_data(airuality, size = 10) :
  object 'airuality' not found
...
```

The way in which you define variables in tidyverse package functions can cause confusion for the R CMD check, which sees column names and the name of your dataset, and flags them as "undefined global variables".

```
utils::globalVariables(c("dataset_name",
                         "col_name_1",
                         "col_name_2"))
```

PDF documentation and Test check

```
check("simutils", manual = TRUE)
```

```
...
pdflatex not found! Not building PDF manual or vignettes.
If you are planning to release this package, please run a check
with manual and vignettes beforehand.
...
```

```
...
* checking tests ...
  Running 'testthat.R'
Warning message:
running command '"C:/PROGRA~1/R/R-34~1.2/bin/x64/R" CMD BATCH --vanilla
"testthat.R" "testthat.Rout"' had status 1
 ERROR
Running the tests in 'tests/testthat.R' failed.
```

```
check("simutils")
```

Differences in package dependencies

DEVELOPING R PACKAGES

Aimée Gott

Education Practice Lead, Mango Solutions

The Depends field

```
search()
```

```
 [1] ".GlobalEnv"        "tools:rstudio"     "package:stats"
 [4] "package:graphics"  "package:grDevices" "package:utils"
 [7] "package:datasets"  "package:methods"   "Autoloads"
[10] "package:base"
```

```
library(tidyverse)
search()
```

```
 [1] ".GlobalEnv"        "package:dplyr"     "package:purrr"
 [4] "package:readr"     "package:tidyr"     "package:tibble"
 [7] "package:ggplot2"   "package:tidyverse" "tools:rstudio"
[10] "package:stats"     "package:graphics"  "package:grDevices"
[13] "package:utils"     "package:datasets"  "package:methods"
[16] "Autoloads"         "package:base"
```

Imports

```
Package: simutils
Title: Simulation Analysis Tools
Version: 1.0.0.0
Authors@R: c(
    person("Nic", "Crane", email = "ncrane@mango-solutions.com", role = c("aut", "cre")),
    person("Aimee", "Gott", email = "agott@mango-solutions.com", role = c("aut", "ctb")))
Description: A series of tools for simulation analysis used for learning about
    distributions.
Depends:
    R (>= 3.4.2)
License: GPL-2
LazyData: true
RoxygenNote: 6.0.1
Imports: dplyr,
    purrr
Suggests: testthat,
    knitr,
    rmarkdown
VignetteBuilder: knitr
```

```
use_package("dplyr")
```

Suggests

```
Package: simutils
Title: Simulation Analysis Tools
Version: 1.0.0.0
Authors@R: c(
    person("Nic", "Crane", email = "ncrane@mango-solutions.com", role = c("aut", "cre")),
    person("Aimee", "Gott", email = "agott@mango-solutions.com", role = c("aut", "ctb")))
Description: A series of tools for simulation analysis used for learning about
    distributions.
Depends:
    R (>= 3.4.2)
License: GPL-2
LazyData: true
RoxygenNote: 6.0.1
Imports: dplyr,
    purrr
Suggests: testthat,
    knitr,
    rmarkdown
VignetteBuilder: knitr
```

```
use_package("ggplot2", "suggests")
```

Building packages with continuous integration

DEVELOPING R PACKAGES

Nic Crane

Data Science Consultant, Mango Solutions

Building with devtools

Build the source:

```
build("simutils")
```

Build the binary:

```
build("simutils", binary = TRUE)
```

What is continuous integration?

- Automatically runs checks when code changed
- Used with version control
- Runs every time you make an update

Setting a package up for using Travis

You can run `use_github()` and `use_travis()` to set up your package for use with GitHub and Travis CI. GitHub is a popular website used for storing code and version control, and Travis CI is used for continuous integration, but what actually is continuous integration?

What are unit tests and why write them?

DEVELOPING R PACKAGES

Aimée Gott

Education Practice Lead, Mango Solutions

Why write unit tests?

A function that works correctly now may not behave as expected in the future if:

- Supporting or connected code could be added or modified
- A later version of R and/or later versions of packages are used
- The code is run on new data
- The code is run on a different operating system

Setting up the test structure

- Call `use_testthat` to set up the test framework
- This creates a `test` directory in the package root directory
- Within the `test` directory, there is a script `testthat.R` which contains code to run the tests
- Within the `test` directory is a directory `testthat` where you save all of your test scripts

Writing an individual test

Some of the most common expect statements:

- `expect_identical` - Checks for exact equality
- `expect_equal` - Checks for equality with numerical tolerance
- `expect_equivalent` - More relaxed version of equals
- `expect_error` - Checks that an expression throws an error
- `expect_warning` - Checks that an expression gives a warning
- `expect_output` - Checks that output matches a specified value

When you run your tests, you might notice that there is no output.
You will only see an output message if the test has failed.

expect_identical

- Strictest numerical comparison
- Compares **values**, **attributes**, and **type**

Passes

```
library(testthat)

my_vector <- c("First" = 1, "Second" = 2)

expect_identical(my_vector, c("First" = 1, "Second" = 2))
```

expect_identical

Fails

```
expect_identical(myvector, c(1, 2))
```

```
Error: `vec1` not identical to c(1, 2).
names for target but not for current
```

In this example, the test fails because myvector contains a named vector, whereas the vector we're comparing it against contains an unnamed vector, so the name attribute does not match.

expect_equal

- Compares **values** and **attributes**
- Doesn't compare **type**

Passes

```
expect_equal(my_vector, c("First" = 1L, "Second" = 2L))
```

- Can set `tolerance` parameter to allow for small differences
- Only differences larger than the `tolerance` value will cause the test to fail

Fails

```
expect_equal(my_vector, c(First = 1.1, Second = 2.1))
```

```
Error: `my_vector` not equal to c(First = 1.1, Second = 2.1).
2/2 mismatches (average diff: 0.1)
[1] 1 - 1.1 == -0.1
[2] 2 - 2.1 == -0.1
```

Passes

```
expect_equal(my_vector, c(First = 1.1, Second = 2.1), tolerance = 0.1)
```

expect_equivalent

- Least strict numerical comparison
- Compares **values** only
- Doesn't compare **attributes** or **type**

Passes

```
expect_equivalent(my_vector, c(1, 2))
```

Testing errors and warnings

Warning

```
sqrt(-1)
```

```
NaN
Warning message:
In sqrt(-1) : NaNs produced
```

Error

```
sqrt("foo")
```

```
Error in sqrt("foo") : non-numeric argument to mathematical function
```

Testing warnings

Passes

```
expect_warning(sqrt(-1))
```

Testing errors

Passes

```
expect_error(sqrt("foo"))
```

Fails

```
expect_error(sqrt(-1))
```

```
Error: sqrt(-1) did not throw an error.
In addition: Warning message:
In sqrt(-1) : NaNs produced
```

Testing specific warning and error messages

Passes

```
expect_error(sqrt("foo"), "non-numeric argument to mathematical function")
```

Fails

```
expect_error(sqrt("foo"), "NaNs produced")
```

```
Error: error$message does not match "NaNs produced".
Actual value: "non-numeric argument to mathematical function"
```

Testing specific output and non-exported functions

DEVELOPING R PACKAGES

Aimée Gott

Education Practice Lead, Mango Solutions

Testing specific output

```
str(airquality)
```

```
'data.frame':    153 obs. of  6 variables:
 $ Ozone  : int  41 36 12 18 NA 28 23 19 8 NA ...
 $ Solar.R: int  190 118 149 313 NA NA 299 99 19 194 ...
 $ Wind   : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
 $ Temp   : int  67 72 74 62 56 66 65 59 61 69 ...
 $ Month  : int  5 5 5 5 5 5 5 5 5 5 ...
 $ Day    : int  1 2 3 4 5 6 7 8 9 10 ...
```

Calling the structure function on the airquality dataset results in the printed message here. The printed message is a side effect rather than a returned object, so how can you test this using testthat?

Testing for expected output

Passes

```
expect_output(str(airquality), "41 36 12 18 NA 28 23 19 8 NA")
```

Fails

```
expect_output(str(airquality), "air")
```

```
Error: str(airquality) does not match "air".
Actual value: "'data.frame':\t153 obs. of  6 variables:\n $ Ozone  : int
41 36 12 18 NA 28 23 19 8 NA ...\n $ Solar.R: int  190 118 149 313 NA NA
299 99 19 194 ...\n $ Wind   : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8
20.1 8.6 ...\n $ Temp   : int  67 72 74 62 56 66 65 59 61 69 ...\n $
Month  : int  5 5 5 5 5 5 5 5 5 5 ...\n $ Day    : int  1 2 3 4 5 6 7 8 9
10 ..."
```

Testing for expected output from a file

First run - create file

```
expect_output_file(str(airquality), "airq.txt", update = TRUE)
```

1

```
Error: str(airquality) not equal to safe_read_lines("airq.txt").
Lengths differ: 7 vs 0
In addition: Warning messages:
1: In file(con, "r") :
  cannot open file 'airq.txt': No such file or directory
2: In value[[3L]](cond) : cannot open the connection
```

Subsequent runs - comparing to file

```
expect_output_file(str(airquality), "airq.txt")
```

2

Testing non-exported functions

Fails

```
expect_equal(sum_na(airquality$Ozone), 37)
```

```
Error in compare(object, expected, ...) :
    could not find function "sum_na"
```

Passes

```
expect_equal(simutils:::sum_na(airquality$Ozone), 37)
```

Grouping tests and execution output

DEVELOPING R PACKAGES

Nic Crane

Data Science Consultant, Mango Solutions

Context and test_that

```
test_that("na_counter correctly counts NA values", {

  test_matrix = matrix(c(NA, 1, 4, NA, 5, 6), nrow = 2)

  air_expected = c(Ozone = 37, Solar.R = 7, Wind = 0,
                   Temp = 0, Month = 0, Day = 0)

  mat_expected = c(V1 = 1, V2 = 1, V3 = 0)

  expect_equal(na_counter(airquality), air_expected)

  expect_equal(na_counter(test_matrix), mat_expected)

})
```

Context

```
context("na_counter checks")

test_that("na_counter correctly counts NA values", {

  test_matrix = matrix(c(NA, 1, 4, NA, 5, 6), nrow = 2)

  air_expected = c(Ozone = 37, Solar.R = 7, Wind = 0,
                   Temp = 0, Month = 0, Day = 0)

  mat_expected = c(V1 = 1, V2 = 1, V3 = 0)

  expect_equal(na_counter(airquality), air_expected)
  expect_equal(na_counter(test_matrix), mat_expected)

})

test_that("na_counter returns error if data is wrong object type", {

  expect_error(na_counter(c(1, 2, 3, NA)))

})
```

Executing unit tests

```
test("simutils")
```

```
Loading simutils
Loading required package: testthat
Testing simutils
na_counter checks: ...
sample_from_data tests: ...

DONE ================================================
```

Testing during the check process

```
* checking tests ...
  Running 'testthat.R'
Warning message:
running command '"C:/PROGRA~1/R/R-34~1.2/bin/x64/R" CMD BATCH --vanilla
"testthat.R" "testthat.Rout"' had status 1
 ERROR
Running the tests in 'tests/testthat.R' failed.
Last 13 lines of output:
  > library(simutils)
  > test_check("simutils")
  1. Failure: sample_from_data returns correct output
  (@test-sample_from_data.R#11)
  df$Ozone not equal to c(22, 47, 45, 80, 7, 21, 23, 23, 16, 44).
  1/10 mismatches
  [10] 45 - 44 == 1

  testthat results ========================================================
  OK: 4 SKIPPED: 0 FAILED: 1
```

Understanding a failing test

Repeat until all test pass:

- Identify the cause
- Determine whether it's the test or the function that needs updating
- Fix your code!
- Run tests again

Congratulations!

DEVELOPING R PACKAGES