

Wrapping Up

July 2018

Charlotte Wickham

@cvwickham

cwickham@gmail.com

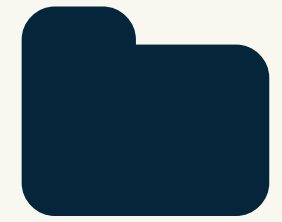
cwick.co.nz



Reusing functions

A starting workflow

Functions are defined where they are used



my-project/



01-import-and-clean.R

```
...  
fix_missing <- function(x){  
  x[x == -99] <- NA  
  x  
}  
  
df <- modify(df, fix_missing)  
...
```

As you use functions in many places in **one** project

Functions are defined in one place



my-project/



01-import-and-clean.R



functions.R

```
source("functions.R")  
...  
df <- modify(df, fix_missing)  
...
```

```
fix_missing <- function(x)  
{  
  x[x == -99] <- NA  
  x  
}
```

As you use functions in many projects

Sub-optimal Functions live in **one** project



my-project/



01-import-and-clean.R



functions.R

```
fix_missing <- function(x)
{
  x[x == -99] <- NA
  x
}
```



my-other-project/



01-import-and-clean.R

```
source("../my-project/functions.R")
...
df <- modify(df, fix_missing)
...
```

As you use functions in many projects

Sub-optimal Functions live in **both** projects



my-project/



01-import-and-clean.R



functions.R

```
fix_missing <- function(x)
{
  x[x == -99] <- NA
  x
}
```



my-other-project/



01-import-and-clean.R

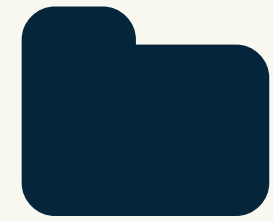


functions.R

```
fix_missing <- function(x)
{
  x[x == -99] <- NA
  x
}
```

As you use functions in many projects

Optimal Functions live in a **package**



my-project/



01-import-and-clean.R

```
library("missing")  
...  
df <- modify(df, fix_missing)  
...
```



my-other-project/



01-import-and-clean.R

```
library("missing")  
...  
df <- modify(df, fix_missing)  
...
```



missing/



R/



fix_missing.R

```
fix_missing <- function(x)  
{  
  x[x == -99] <- NA  
  x  
}
```

Principle:

As soon as you need the same function over more than one project, turn it into a package.

Put your functions in a package to get:

1. Improved workflow with `devtools::load_all()`
2. Easy sharing across projects with `devtools::install() + library()`
3. Easy to add formal documentation and testing.


Put your functions in a package to get:

1. Improved workflow with
`devtools::load_all()`
- 2. Easy sharing across projects with
`devtools::install() + library()`**
3. Easy to add formal documentation
and testing.

Your Turn

What happens?

```
usethis::create_package("~/Desktop/missing")
```



A directory you
have write
access

Package name

An R package is just a set of conventions

 missing/

 R/

Functions go in .R files in here



DESCRIPTION

General package info

... some other stuff we won't worry about

Your Turn

```
# This will create and open R/fix_missing.R  
usethis::use_r("fix_missing")
```

```
# Copy and paste this function in  
fix_missing <- function(x){  
  x[x == -99] <- NA  
  x  
}
```

```
# Cmd/Ctrl + Shift + L
```

Install package with devtools::install()

```
devtools::install()  
# Installing missing  
# '/Library/Frameworks/R.framework/Resources/bin/R' --no-site-file \  
# --no-environ --no-save --no-restore --quiet CMD INSTALL \  
# '/Users/wickhamc/Desktop/missing' --library='/Users/wickhamc/R' \  
# --install-tests  
#  
# * installing *source* package 'missing' ...  
# ** R  
# ** byte-compile and prepare package for lazy loading  
# ** help  
# No man pages found in package 'missing'  
# *** installing help indices  
# ** building package indices  
# ** testing if installed package can be loaded  
# * DONE (missing)  
# Reloading installed missing
```

Next? Document and test

Document <http://r-pkgs.had.co.nz/man.html>

You are bound to forget how your functions work, document to remind yourself.

Test <http://r-pkgs.had.co.nz/tests.html>

You will want to add features without breaking existing functionality.

These become increasingly important the more you and others rely on these functions.

Wrapping Up

Where have we been?

Writing Functions

Extract actions into functions, so you can reduce repetitive code that distracts from your purpose.

Functional Programming

Let other functions write for loops for you, so you can concentrate on the actions not the details.

Tidy Evaluation

Do all of the above with functions in the tidyverse.

Overall goal:

Code that expresses what you did, not
the details of how you did it.

Learning more

In addition to resources in each section:

1. Revisit old analyses, can you rewrite them to reduce repetition?
2. Read other people's code - analyses, functions in published packages etc.
3. Review other peoples packages, e.g. <https://github.com/ropensci/onboarding#why-review>

Thank you!

Charlotte Wickham

@cvwickham

cwickham@gmail.com

cwick.co.nz

This work is licensed as
Creative Commons
Attribution-ShareAlike 4.0
International

To view a copy of this license, visit
[https://creativecommons.org/licenses/by-sa/
4.0/](https://creativecommons.org/licenses/by-sa/4.0/)