R packages

Based on this example: https://github.com/isteves/r-pkg-intro

Why?

- Package your work and share it!
  - organize code **and** data
  - reuse and share your work more easily
- test code
- share code with others

What?

Mostly organizing your work in a file structure plus a few special files that help to bundle everything together.

Package structure

- R code (`/R`)
- Documentation (`/man`)
- Tests (`/tests`)
- Package metadata (`DESCRIPTION`)
- Namespace (`NAMESPACE`)
- Data (`/data`)
- Vignettes (`/vignettes`)
- Compiled code (`/src`)
- Installed files (`/inst`)
- Other components

How?

2 helpful packages when developing R packages:

- `devtools`
- `usethis`

And of course, RStudio helps you to build packages 🙂

Demo

Let's reproduce the greetings package: https://github.com/brunj7/greetings

And yes, you can install an R package directly from GitHub 🤯

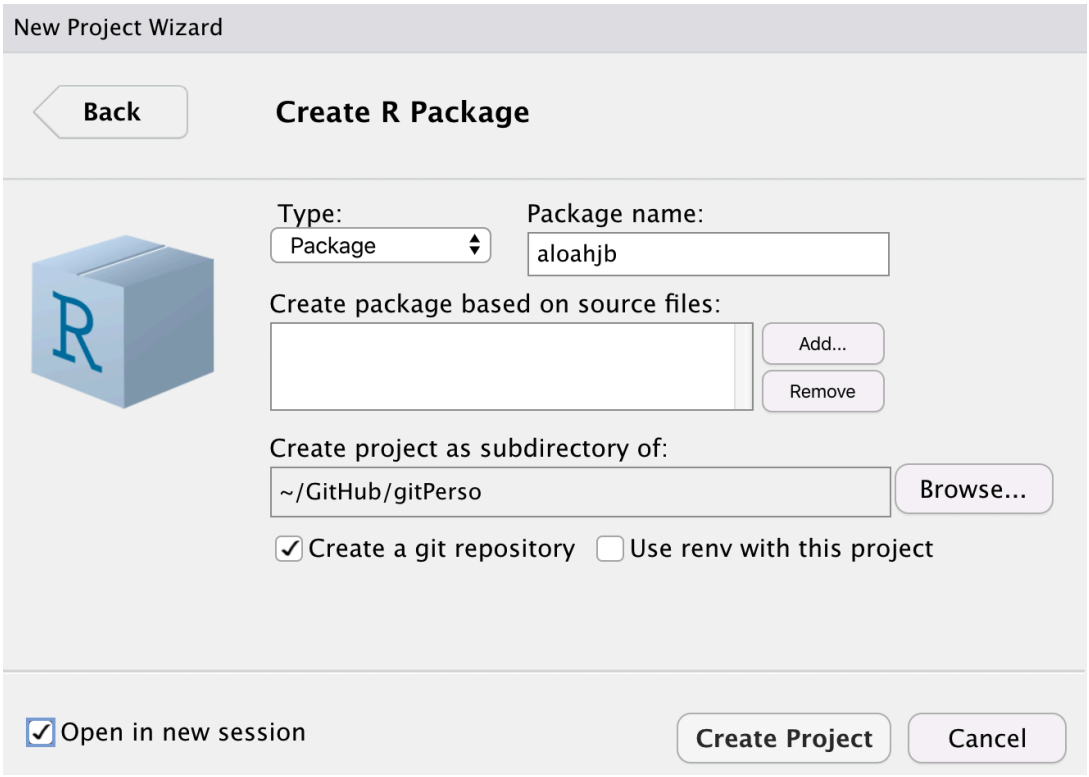
```
devtools::install_github("brunj7/greetings")
library(greetings)
greetings::say_aloha("Allison")

#> Aloha, Allison 🌴 ☀️ 🌊
```

Let's recreate this package!!

Assignment

Create your version of the aloha R package!

1. Using RStudio. Here we will be using RStudio to create a new project selecting Create R package



2. Add some files to your package infrastructure

a. Add a ReadMe (run in Console)
`usethis::use_readme_md()`

b. Add a license (run in Console)
`usethis::use_mit_license("Your Name")`

c. Create a new R Script
Copy/paste the function `say_aloha` (below) into the script

```
say_aloha <- function(name, print = TRUE) {

  message <- paste("Aloha,",
                   name,
                   emo::ji("palm_tree"),
                   emo::ji("sunny"),
                   emo::ji("ocean"))

  if (print) {
    cat(crayon::bgGreen(message))
  }

  invisible(message)
}
```

d. Save the script in the /R folder
e. Commit your changes with git

3. Link this local repository to GitHub by running the following in Console (this creates a GitHub repo and configures as git remote for your project)

`usethis::use_github()`

Follow the instructions until your web browser opens your repository.

4. Add documentation

a. Document the function using a Roxygen skeleton (Code > Insert Roxygen Skeleton)
b. Run `devtools::document()` in the Console to update the package documentation
See now there is a .Rd file in the /man folder

Notes:

- this also adds the function name to the NAMESPACE file to make it available to the user of the package.
- This creates the documentation of the function that you access with `?say_aloha`

Don't forget to commit changes using git!

c. Update the DESCRIPTION file following instructions
d. Add the dependencies of your package at the bottom of the Description file

```
Imports: crayon, emo
Remotes: hadley/emo
Suggests: testthat,
    knitr,
    rmarkdown
VignetteBuilder: knitr
```

e. Run `devtools::document()` to update the package documentation

5. Build your package!

Run `devtools::build()` to build the package, or use the "install and restart" button in the Build tab in the RStudio

Now your package has been built (and it will automatically attach it so it's ready to use). Try your function `say_aloha("Someone's Name")`!!!

- Iterate if not working; good reference: https://r-pkgs.org/
- Make some modifications to the function to personalize it!
- Push everything to GitHub and ask a team member to try it!!

Other Thoughts

you can create a package by many ways. If you were using a computing environment without RStudio installed, the with the {`usethis` package (https://usethis.r-lib.org/) is a great way to do so:

a. In RStudio (not in a project), in the Console run:
   `pkgpath <- file.path("~/Desktop", "alohayourinitials")`

   For example, Julien's would be alohajb. Then, create the package by running (in the Console):

   `usethis::create_package(pkgpath)`

   Then initialize the git repository by running (in the Console):
   `usethis::use_git()`

b. Commit the R package file structure