Intro to R Packages

<u>R Package Anatomy</u>

- **Functions (in R folder)**
- Documentation
  - Rd files in ~/man folder
  - DESCRIPTION
  - Vignettes (optional)
- Tests in ~/test folder (optional)

<u>Reasons to Build an R package</u>

- Organize and document your code
- One place to update code
- Easier to share across projects and with others

Note: Not everything needs to be in a package! But, if you find yourself doing the same tasks over and over, consider breaking your steps into functions and organizing them into a package. It will save so much time in the end!

Steps to build a new R package*

Requirements

- R 4.x and recent version of RStudio
- Rtools for R 4.x (assuming Windows OS)
- Packages: devtools, roxygen2

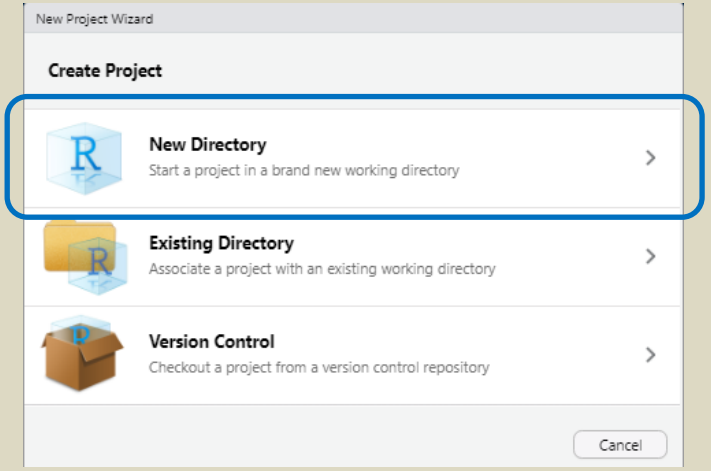
Suggested

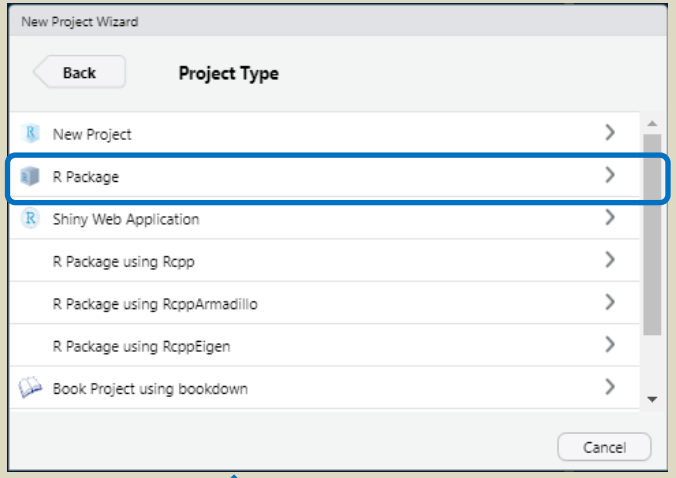
- GitHub account to store package and for version control

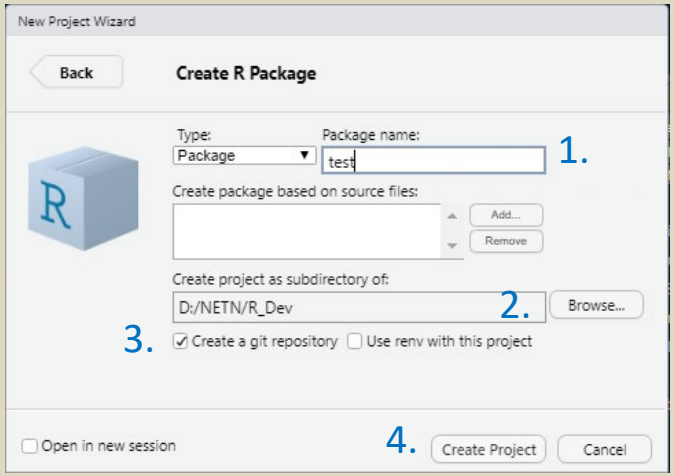
*This is the process recommended by RStudio as of October 2020, but RStudio is continually improving and making this process easier and things may change in the future.

Steps to build a new R package

1. Open R Studio and create an R package:
   - File > New Project > New Directory > R Package





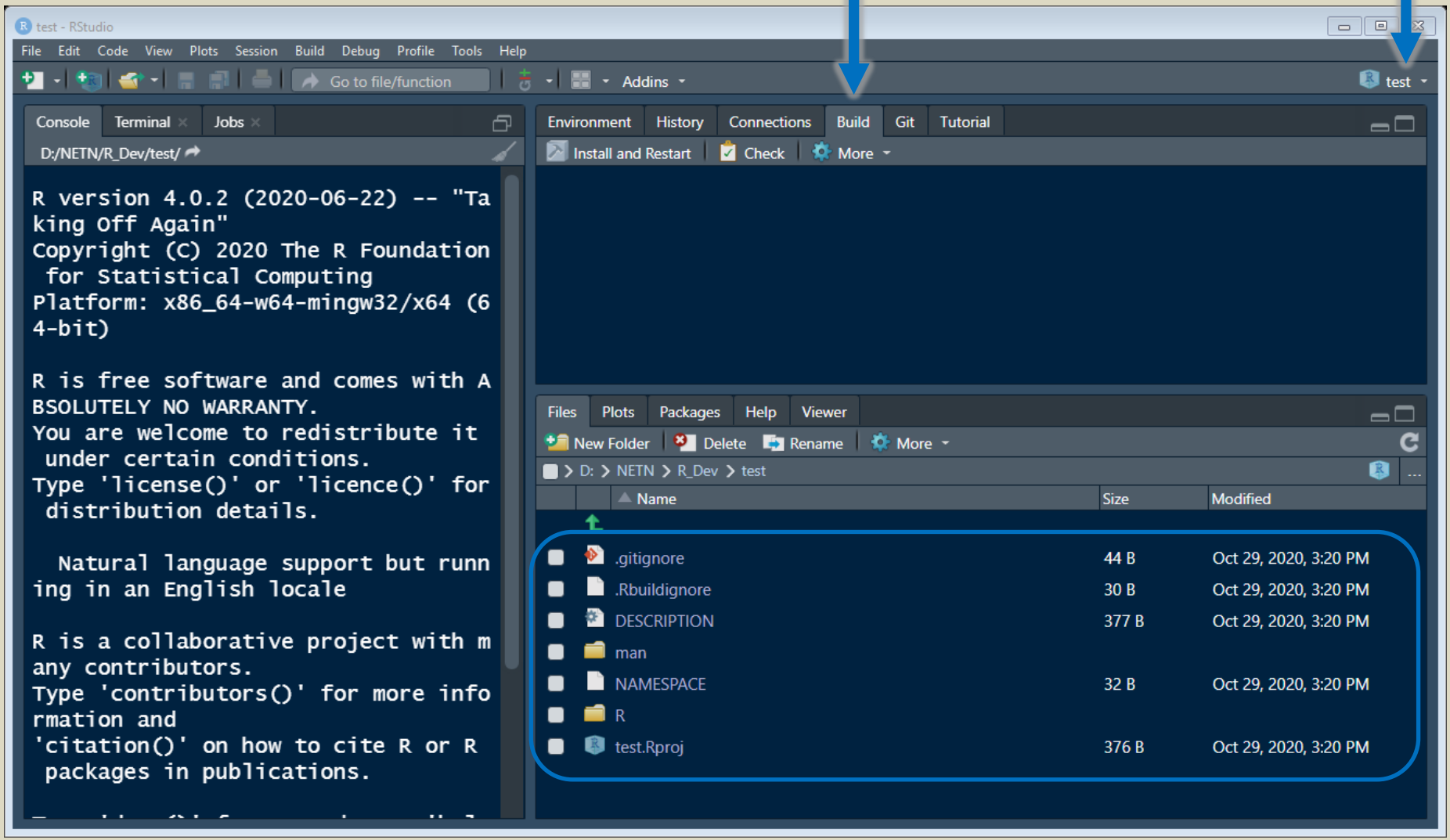


1. Name the package
2. Specify directory
3. Check Create git repo
4. Click Create Project

Alternative: devtools::create_package("D:/NETN/R_Dev/test")
usethis::use_git() # to set up git for new test package

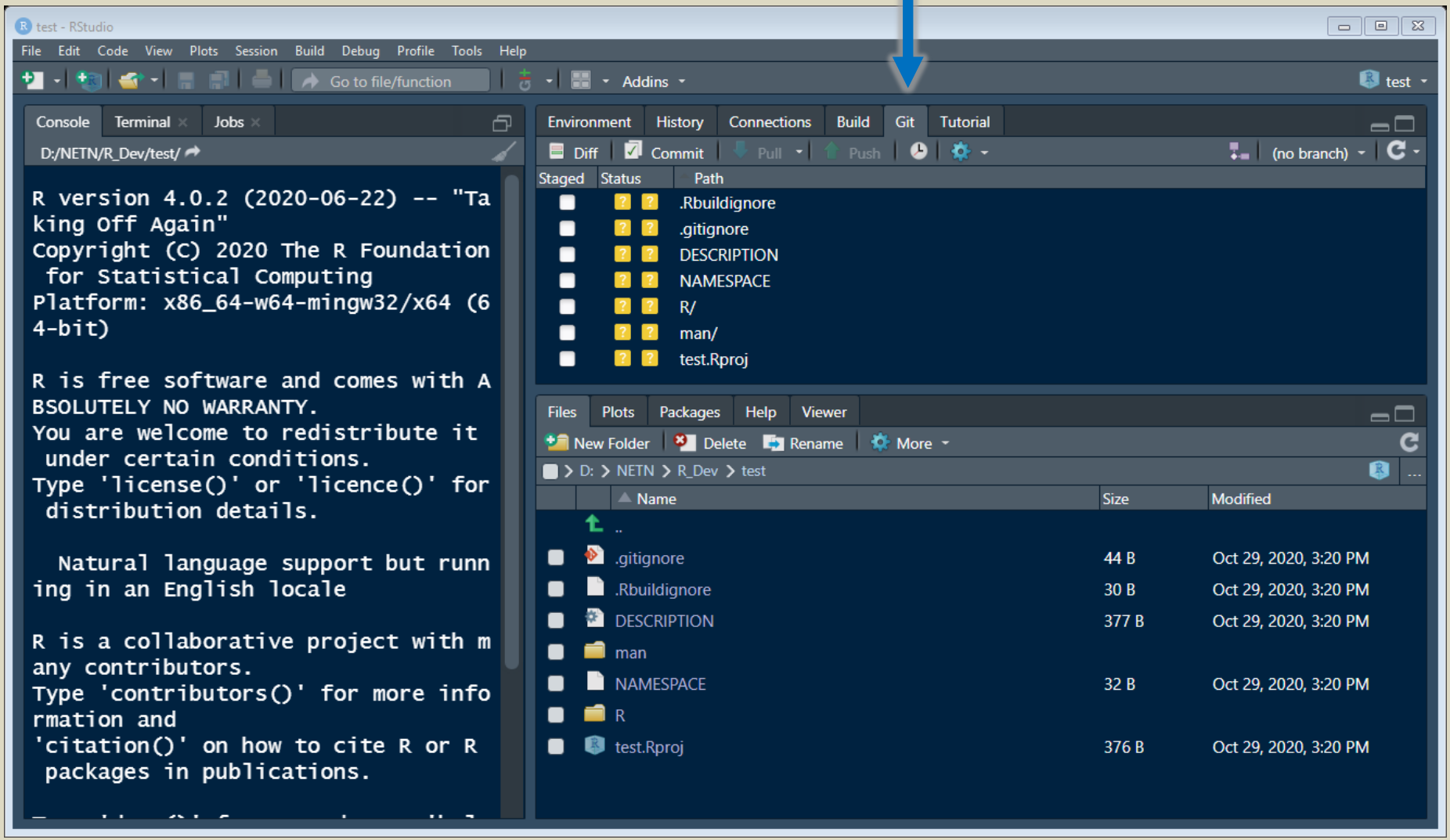
Steps to build a new R package

2. Check out new options and project files:



Steps to build a new R package

2. Check out new options and project files:



Steps to build a new R package

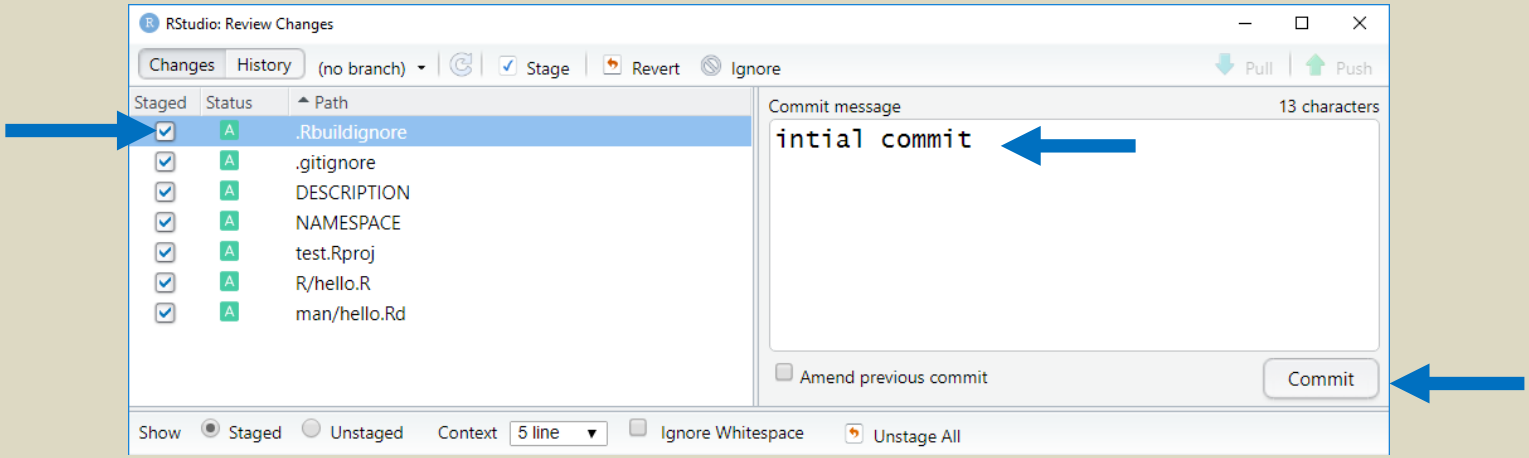
3. Set Build Options in RStudio

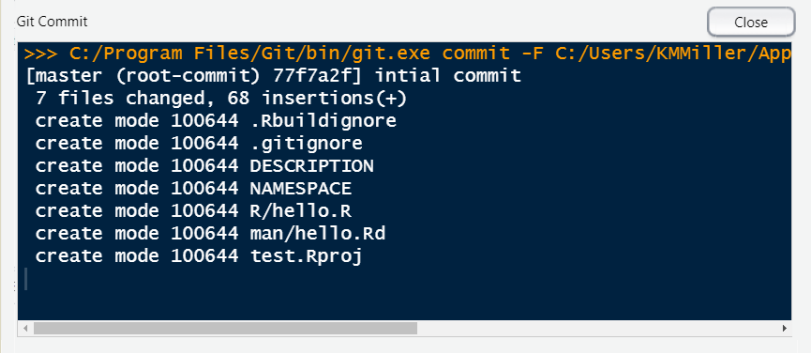
   Tools > Project Options > Build Tools

   - Check Generate documentation with Roxygen

4. Set up local Git repo (note: must be signed into GitHub account)

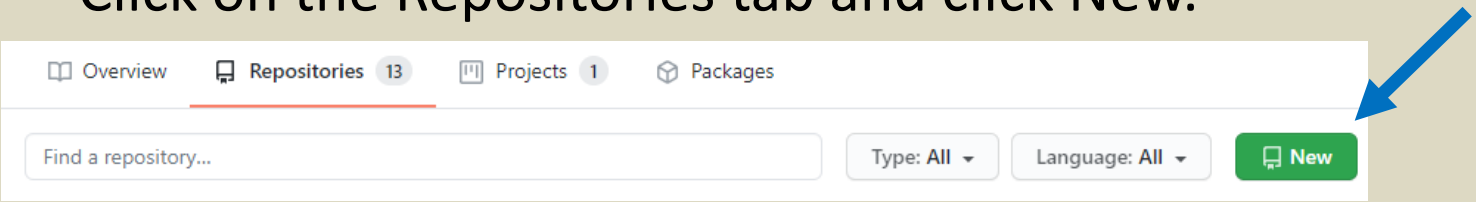
   - In Git tab in RStudio Click on the Commit (Commit) and stage the files in your new package.

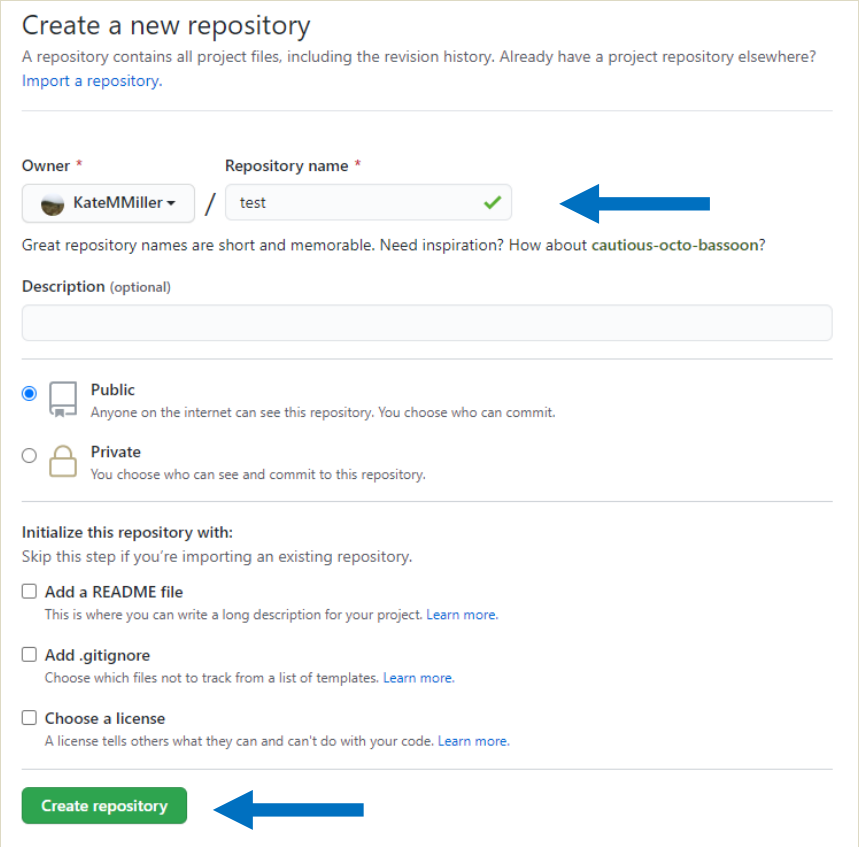




Steps to build a new R package

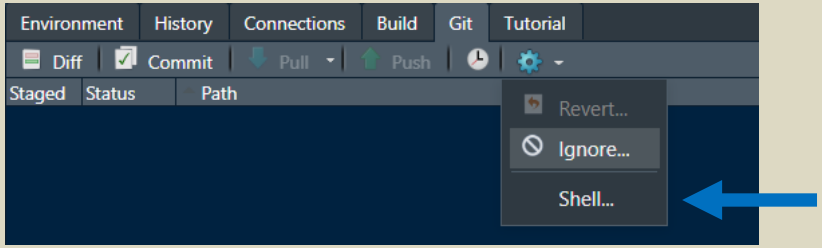
5. Create GitHub repo for package
   - Go to your main GitHub page (eg github.com/KateMMiller)
   - Click on the Repositories tab and click New.

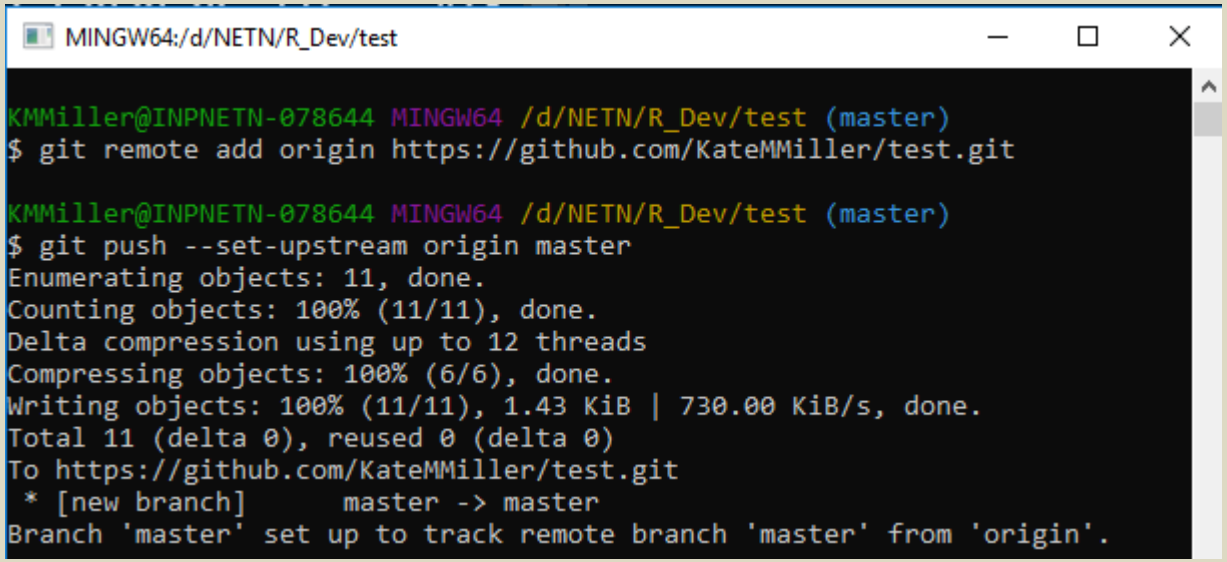




Steps to build a new R package

6. Open Git Shell to connect and push to GitHub
    - git remote add origin https://github.com/KateMMiller/test.git
    - git push --set-upstream origin master





7. Check your new repo on GitHub. You should see the files you just pushed.

Next steps for package development

8. Write a function and documentation for your package

- Anatomy of a function

```
function_name <- function(arguments){
  function body
}
```

make_sppcode.R

```r
library(tidyverse)
make_sppcode <- function(df){
  df2 <- df %>% mutate(genus = word(Latin_Name, 1),
                       species = ifelse(is.na(word(Latin_Name, 2)), "spp",
                                        word(Latin_Name,2)),
                       sppcode = toupper(paste0(substr(genus, 1, 3),
                                                substr(species, 1, 3))))
    return(df2)
}
```

make_sppcode.R | Untitled1*

```r
source("make_sppcode.R")
ex_df <- data.frame(Latin_Name = c("Abies balsamea", "Acer rubrum",
                                   "Betula", "Pinus strobus"))
make_sppcode(df = ex_df)
```

2:42 (Top Level)

Console | Terminal | Jobs

D:/NETN/R_Dev/test/

```
> make_sppcode(df = ex_df)
      Latin_Name  genus  species sppcode
1 Abies balsamea  Abies balsamea  ABIBAL
2    Acer rubrum   Acer   rubrum  ACERUB
3         Betula Betula      spp  BETSPP
4  Pinus strobus  Pinus  strobus  PINSTR
```

Next steps for package development

8. Write a function and documentation for your package
   - Anatomy of a function
   - Roxygen2 documentation

make_sppcode.R | DESCRIPTION | NAMESPACE

Source on Save

```r
#' @title make_sppcode: Convert Latin Name to 6-letter code
#'
#' @importFrom magrittr %>%
#' @importFrom dplyr mutate
#'
#' @description This function takes a data frame with a column called Latin Name
#' and adds new columns for genus, species and a 6-letter code.
#'
#' @param df Specify the name of the data frame
#'
#' @examples
#' ex_df <- data.frame(Latin_Name = c("Abies balsamea", "Acer rubrum",
#'                                    "Betula", "Pinus strobus"))
#'
#' make_sppcode(df = ex_df)
#'
#' @return returns a data frame with genus, species and sppcode columns
#'
#' @export

make_sppcode <- function(df){
  df2 <- df %>% mutate(genus = stringr::word(Latin_Name, 1),
                       species = ifelse(is.na(stringr::word(Latin_Name, 2)), "spp",
                                        stringr::word(Latin_Name,2)),
                       sppcode = toupper(paste0(substr(genus, 1, 3),
                                                substr(species, 1, 3))))
    return(df2)
}
```

Next steps for package development

9. Update DESCRIPTION

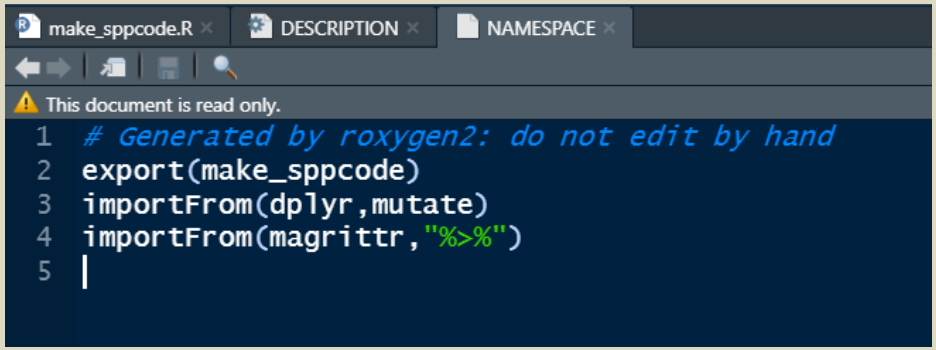
make_sppcode.R | DESCRIPTION | NAMESPACE

```
Package: test
Type: Package
Title: Test: a test package
Version: 0.1.0
Author: K. Miller
Maintainer: The package maintainer <kathryn_miller@nps.gov>
Description: This package is a test package for training purposes
License: Public Domain
Encoding: UTF-8
LazyData: true
RoxygenNote: 7.1.1
Depends:
    R(>= 4.0)
Collate:
    'make_sppcode.R'
Imports:
    dplyr,
    magrittr
Suggests:
    stringr
```

Next steps for package development

10. Build/Rebuild package and check functions and documentation

- NAMESPACE



- Skim .Rd files

- Check that functions still work

Next steps for package development

11. Commit/Push changes to GitHub
    - Use git workflow discussed in previous meeting

12. Share with others: devtools::install_github("username/package")

```r

devtools::install_github("KateMMiller/test")
library(test)
ex_df <- data.frame(Latin_Name = c("Abies balsamea", "Acer rubrum",
                                   "Betula", "Pinus strobus"))

make_sppcode(df = ex_df)
```

Other tasks:

- Unit testing (see Sarah's presentation from last meeting)
- Build vignettes and/or GitHub Page for your packages
- Comprehensive *.md (markdown) and roxygen 'documentation'
- Includes: lookup content, etc.

Steps to build a new R package

1. Create an R Package in RStudio (via New Project)
2. Check out new options and project files in RStudio
3. Set Build Options to Generate documentation with Roxygen
4. Set up local repo for package in RStudio
5. Create repo for package on GitHub
6. Open Git Shell to connect and push local repo to GitHub
7. Check that your files were pushed to your new GitHub repo
8. Write functions and Roxygen2 for your package
9. Update DESCRIPTION
10. Build/Rebuild package and check functions & documentation
11. Commit/Push changes to GitHub
12. Share with others: devtools::install_github(“username/package”)

Resources

- RStudio Package Development Tutorial:
  https://support.rstudio.com/hc/en-us/sections/200130627-Package-Development
- Hadley Wickham's R Packages Book: https://r-pkgs.org/
- R4DS Chapter on writing functions: https://r4ds.had.co.nz/functions.html
- R package primer: https://kbroman.org/pkg_primer/
- Intro to Roxygen2:
  https://cran.r-project.org/web/packages/roxygen2/vignettes/roxygen2.html
- Making your first R package blog:
  https://tinyheero.github.io/jekyll/update/2015/07/26/making-your-first-R-package.html
- IMD Repository Info (basics – update needed):
  https://irma.nps.gov/DataStore/Reference/Profile/2267327