

Intro to R Packages

R Package Anatomy

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

Reasons to Build an R package

- 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

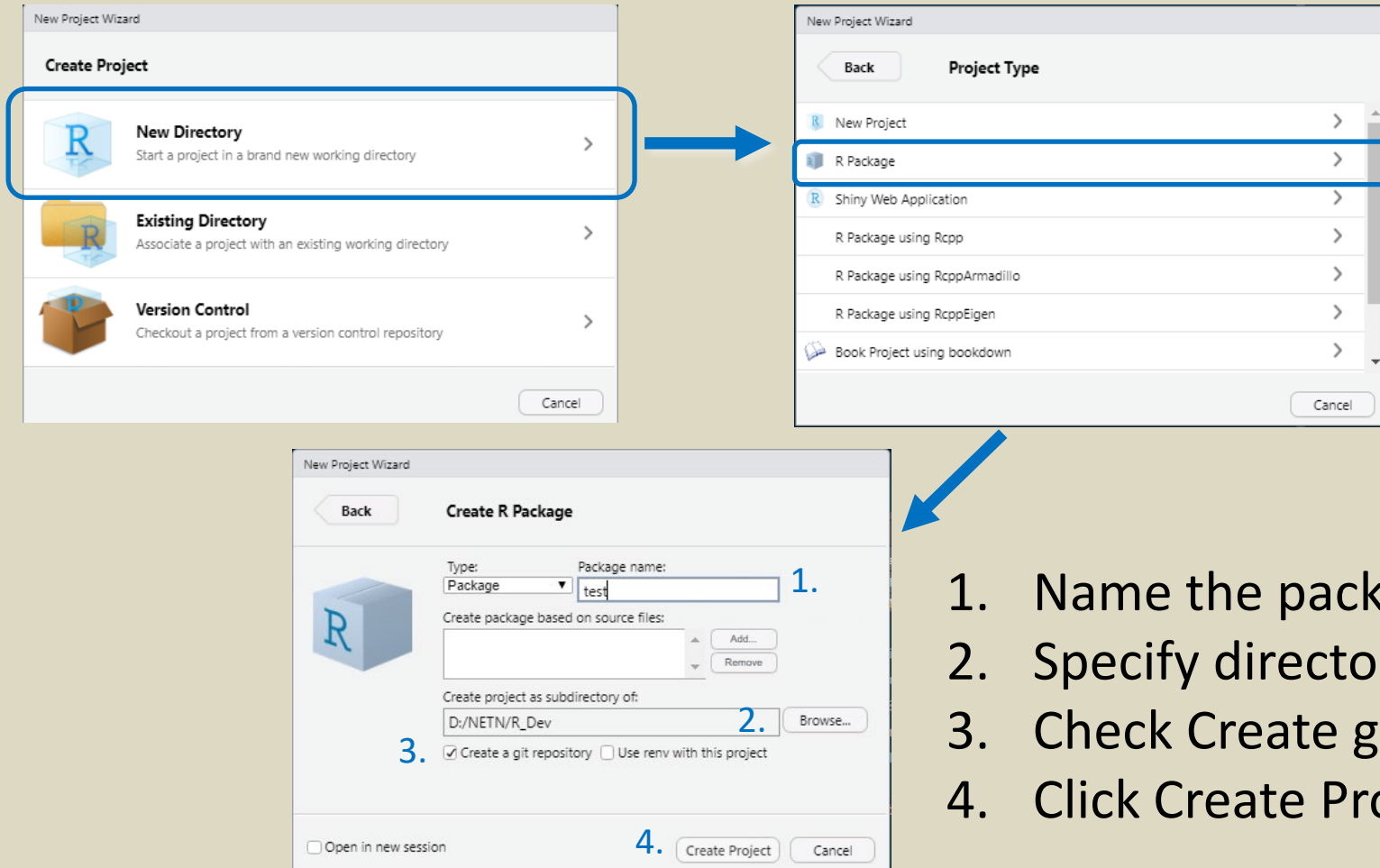
- GitHub account to store package and for version control

*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

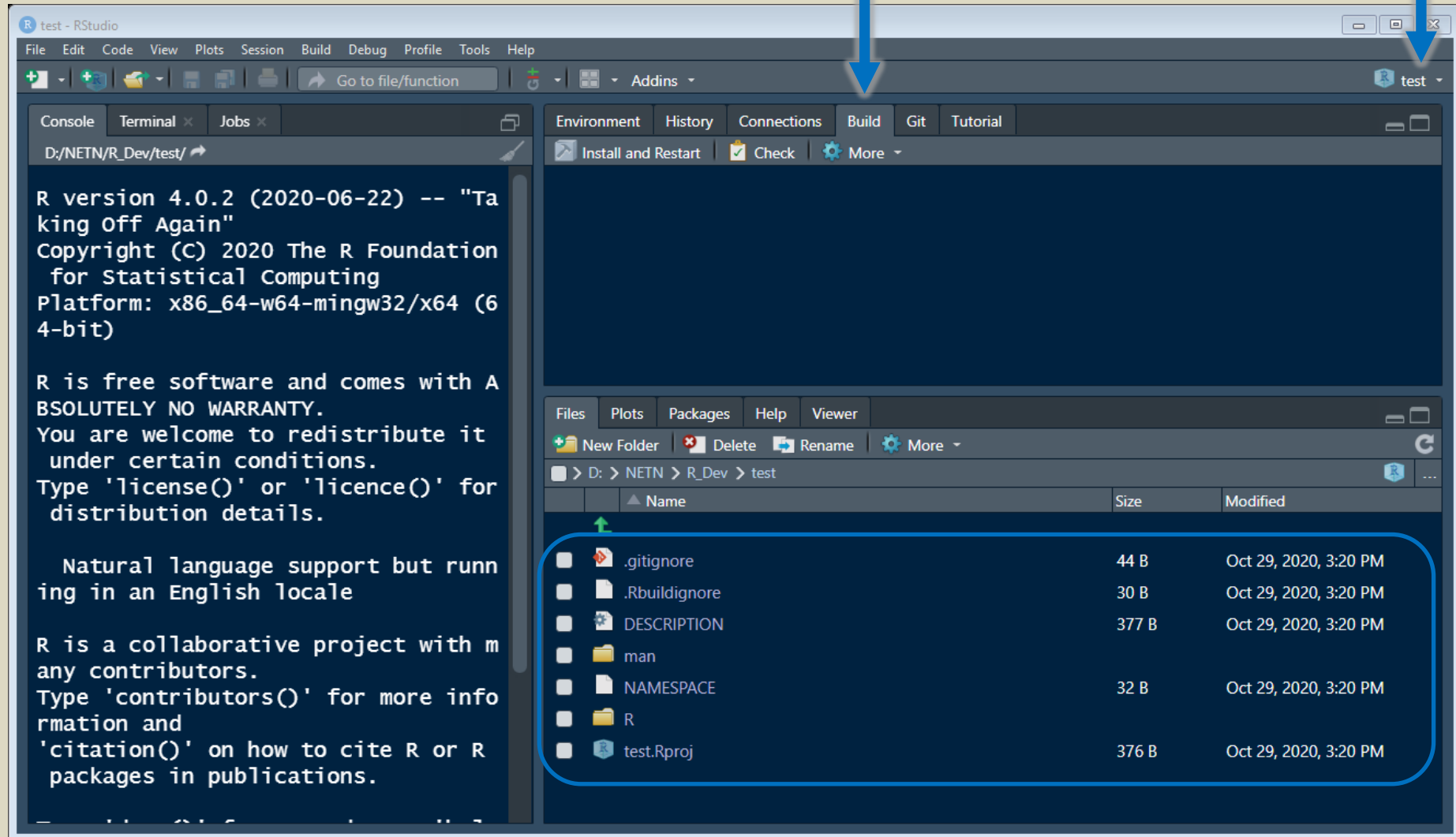


1. Name the package
2. Specify directory
3. Check Create git repo
4. Click Create Project

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

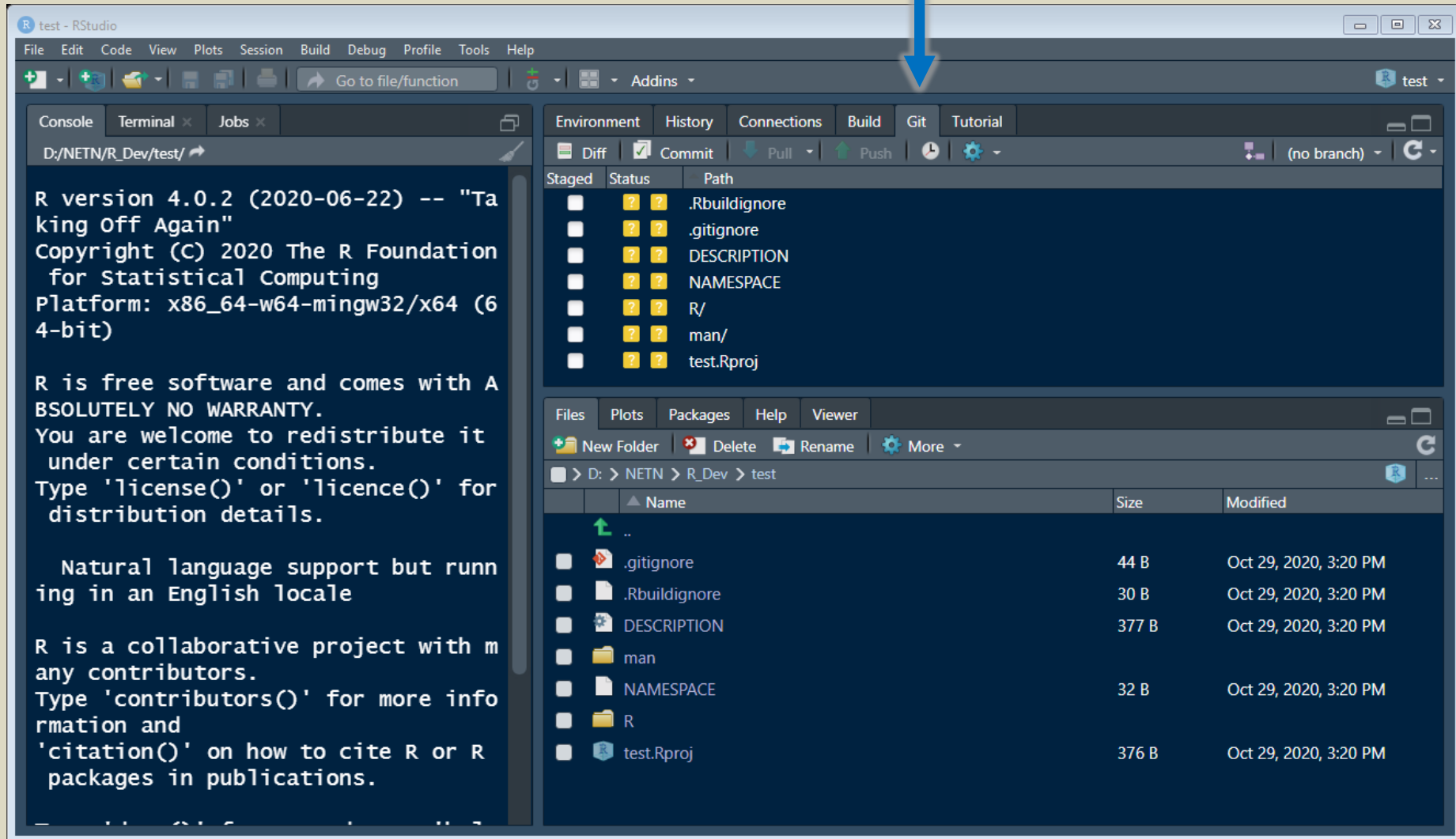
Steps to build a new R package

2. Check out new options and project files:



Steps to build a new R package

2. Check out new options and project files:



The screenshot shows the RStudio interface with the console and Git pane. A blue arrow points to the 'Git' tab in the top right pane.

Console Output:

```
R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
```

Git Pane:

Staged	Status	Path
<input type="checkbox"/>	?	.Rbuildignore
<input type="checkbox"/>	?	.gitignore
<input type="checkbox"/>	?	DESCRIPTION
<input type="checkbox"/>	?	NAMESPACE
<input type="checkbox"/>	?	R/
<input type="checkbox"/>	?	man/
<input type="checkbox"/>	?	test.Rproj

Files Pane:

Name	Size	Modified
..		
.gitignore	44 B	Oct 29, 2020, 3:20 PM
.Rbuildignore	30 B	Oct 29, 2020, 3:20 PM
DESCRIPTION	377 B	Oct 29, 2020, 3:20 PM
man		
NAMESPACE	32 B	Oct 29, 2020, 3:20 PM
R		
test.Rproj	376 B	Oct 29, 2020, 3:20 PM


Steps to build a new R package

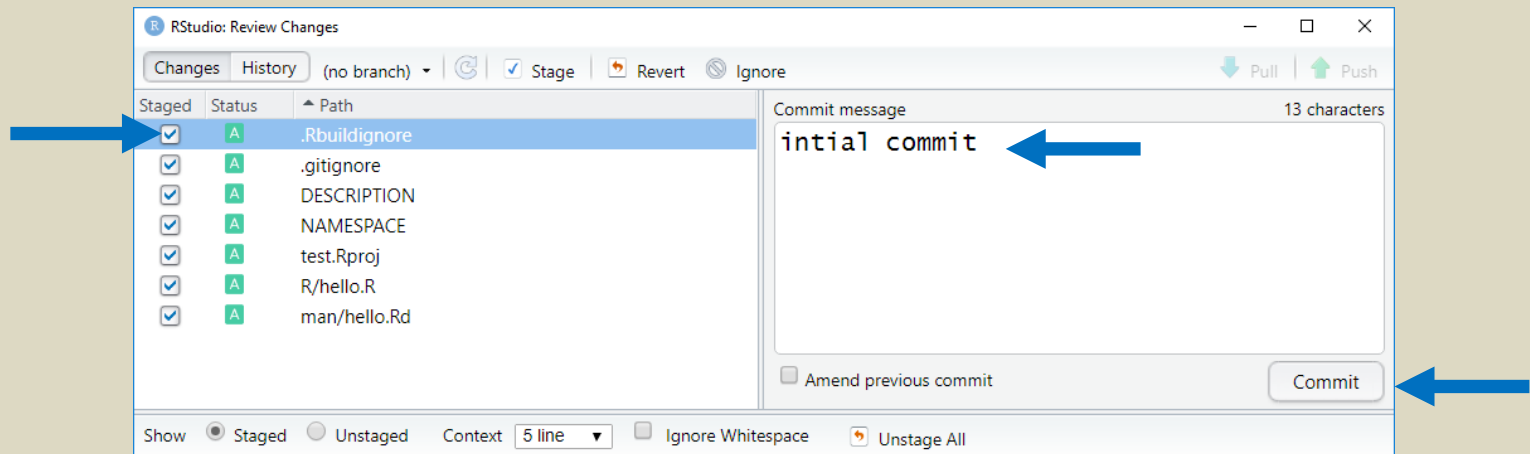
3. Set Build Options in RStudio

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.



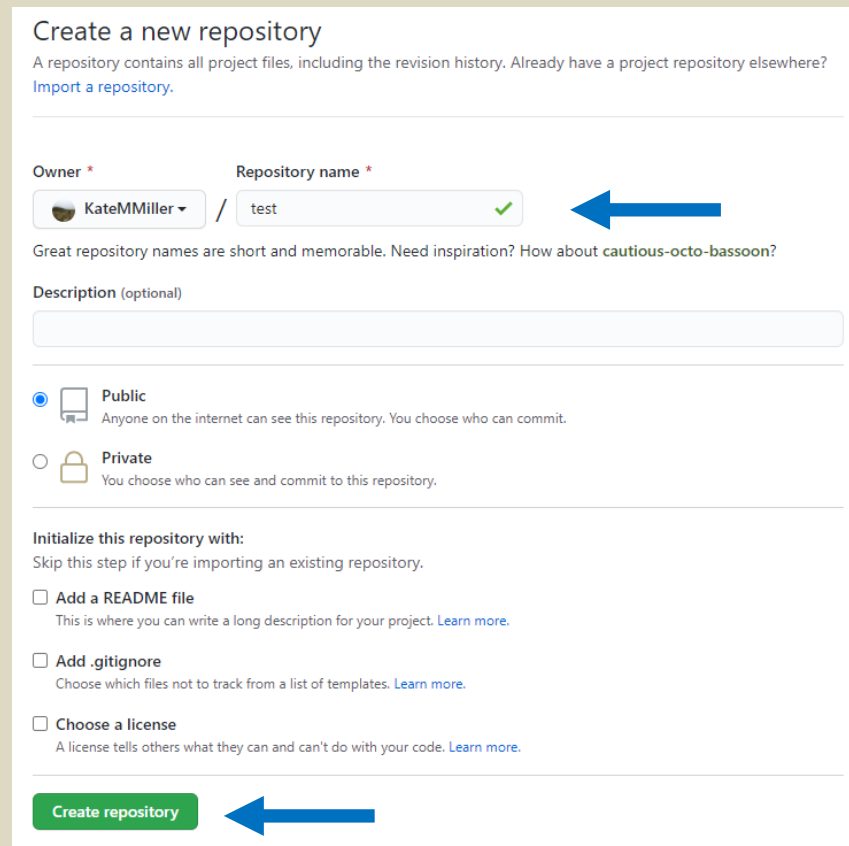
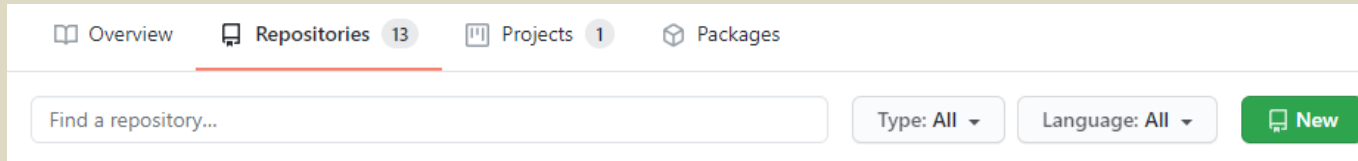
```
Git Commit
Close

>>> C:/Program Files/Git/bin/git.exe commit -F C:/Users/KMMiller/App
[master (root-commit) 77f7a2f] intial commit
7 files changed, 68 insertions(+)
create mode 100644 .Rbuildignore
create mode 100644 .gitignore
create mode 100644 DESCRIPTION
create mode 100644 NAMESPACE
create mode 100644 R/hello.R
create mode 100644 man/hello.Rd
create mode 100644 test.Rproj
```

Steps to build a new R package

5. Create GitHub repo for package

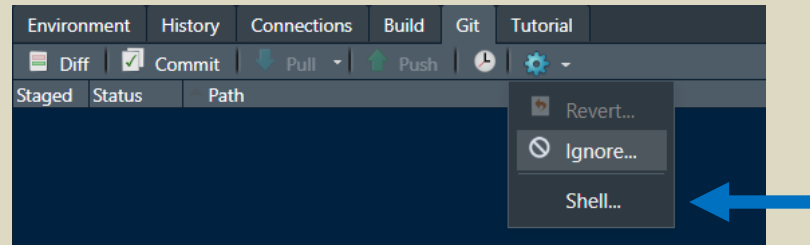
- Go to your main GitHub page (eg github.com/KateMMiller)
- Click on the Repositories tab and click New.

A screenshot of the 'Create a new repository' form on GitHub. The form has a title 'Create a new repository' and a subtitle 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)'. Below this is a section for 'Owner' and 'Repository name'. The 'Owner' is 'KateMMiller' and the 'Repository name' is 'test', which has a green checkmark next to it. A blue arrow points to the 'test' field. Below this is a text input field for 'Description (optional)'. Then there are two radio buttons: 'Public' (selected) and 'Private'. Below these are three checkboxes: 'Add a README file', 'Add .gitignore', and 'Choose a license'. At the bottom is a green button labeled 'Create repository', which is highlighted with a blue arrow.

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`



```
MINGW64:/d/NETN/R_Dev/test
KMMiller@INPNETN-078644 MINGW64 /d/NETN/R_Dev/test (master)
$ git remote add origin https://github.com/KateMMiller/test.git

KMMiller@INPNETN-078644 MINGW64 /d/NETN/R_Dev/test (master)
$ git push --set-upstream origin master
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (11/11), 1.43 KiB | 730.00 KiB/s, done.
Total 11 (delta 0), reused 0 (delta 0)
To https://github.com/KateMMiller/test.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

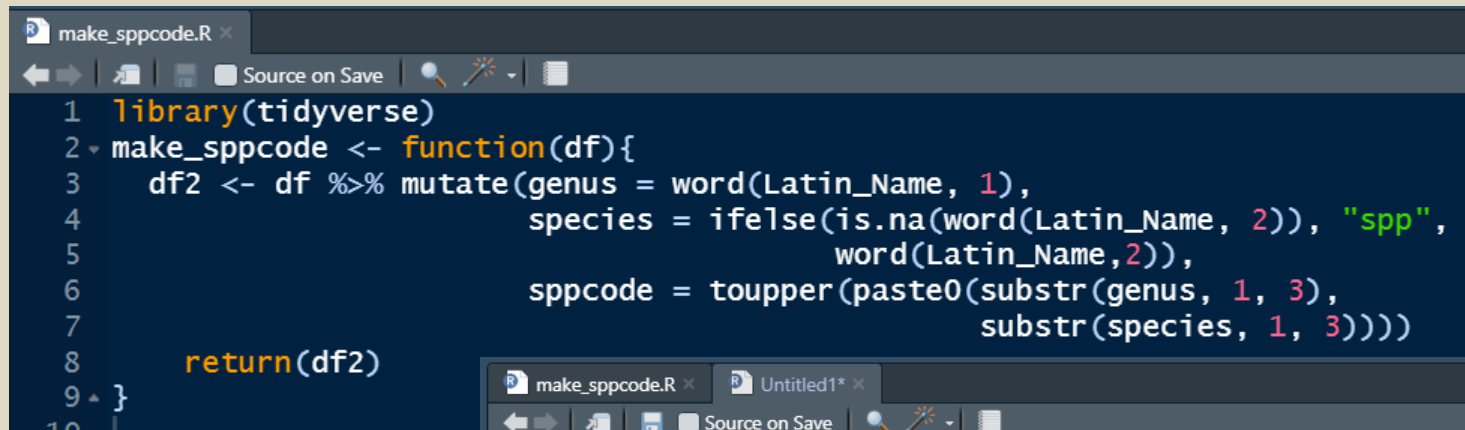
7. Check your new repo on GitHub. You should see the files you just pushed.

Next steps for package development

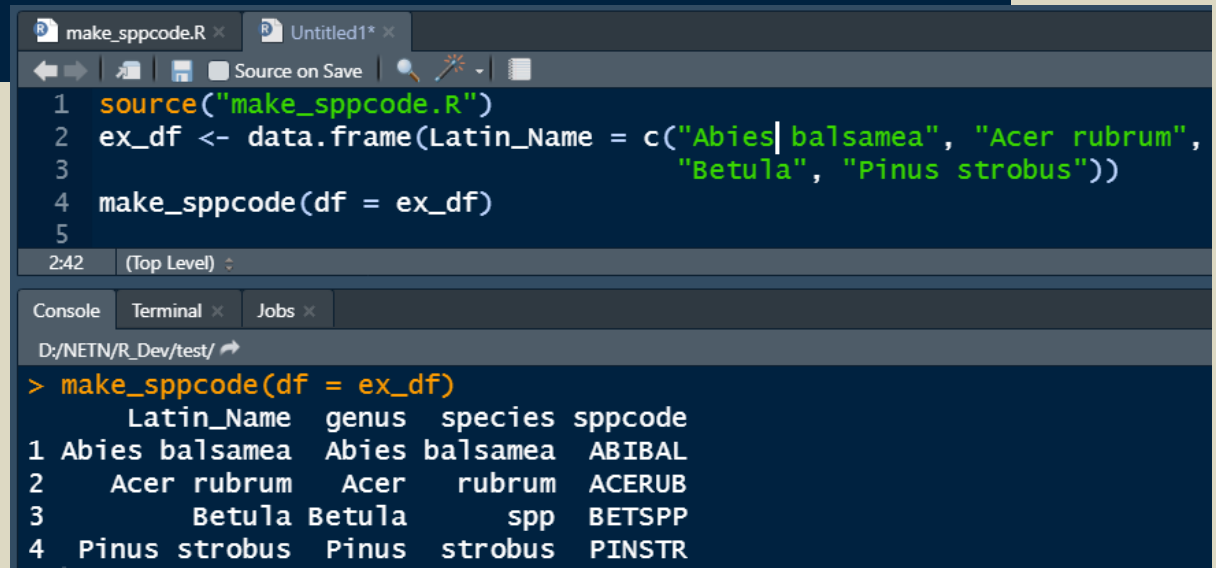
8. Write a function and documentation for your package

- Anatomy of a function

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



```
make_sppcode.R x  
Source on Save  
1 library(tidyverse)  
2 make_sppcode <- function(df){  
3   df2 <- df %>% mutate(genus = word(Latin_Name, 1),  
4                         species = ifelse(is.na(word(Latin_Name, 2)), "spp",  
5                                           word(Latin_Name, 2)),  
6                         sppcode = toupper(paste0(substr(genus, 1, 3),  
7                                                  substr(species, 1, 3))))  
8   return(df2)  
9 }  
10
```

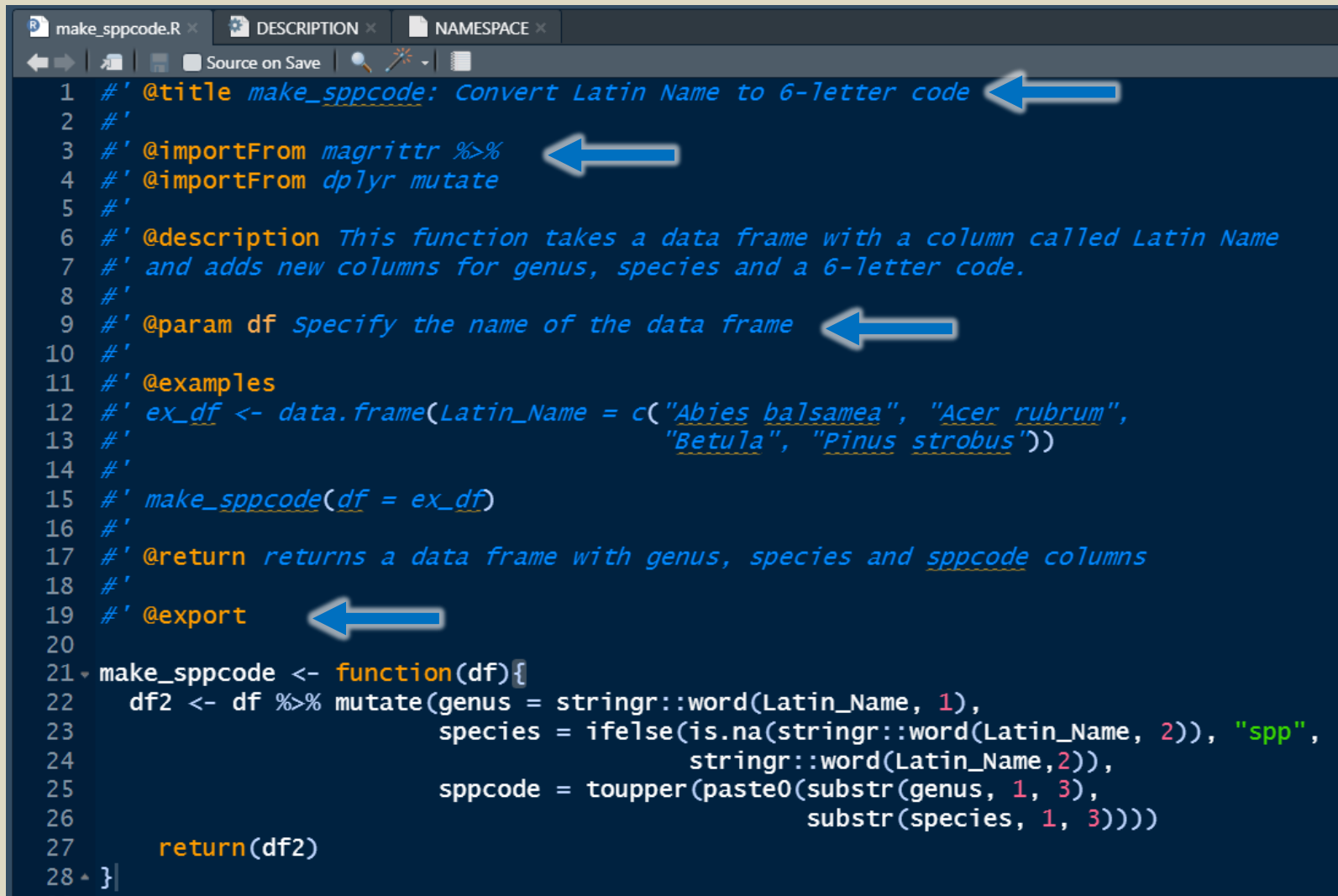


```
make_sppcode.R x  Untitled1* x  
Source on Save  
1 source("make_sppcode.R")  
2 ex_df <- data.frame(Latin_Name = c("Abies balsamea", "Acer rubrum",  
3                                     "Betula", "Pinus strobus"))  
4 make_sppcode(df = ex_df)  
5  
2:42 (Top Level) :  
Console Terminal x Jobs x  
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

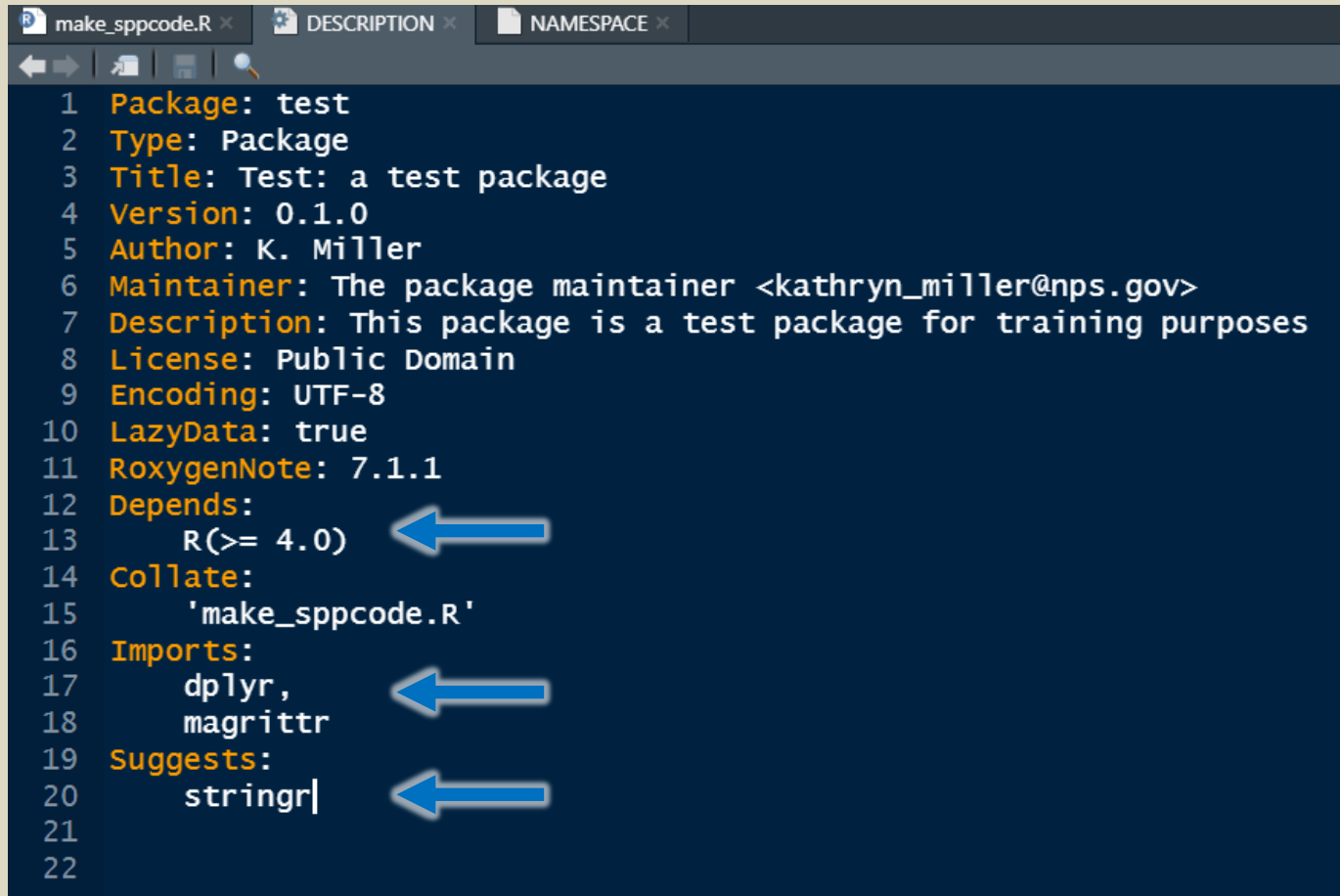
- Anatomy of a function
- Roxygen2 documentation



```
1 #' @title make_sppcode: Convert Latin Name to 6-letter code
2 #'
3 #' @importFrom magrittr %>%
4 #' @importFrom dplyr mutate
5 #'
6 #' @description This function takes a data frame with a column called Latin Name
7 #' and adds new columns for genus, species and a 6-letter code.
8 #'
9 #' @param df Specify the name of the data frame
10 #'
11 #' @examples
12 #' ex_df <- data.frame(Latin_Name = c("Abies balsamea", "Acer rubrum",
13 #'                                     "Betula", "Pinus strobus"))
14 #'
15 #' make_sppcode(df = ex_df)
16 #'
17 #' @return returns a data frame with genus, species and sppcode columns
18 #'
19 #' @export
20
21 make_sppcode <- function(df){
22   df2 <- df %>% mutate(genus = stringr::word(Latin_Name, 1),
23                       species = ifelse(is.na(stringr::word(Latin_Name, 2)), "spp",
24                                       stringr::word(Latin_Name, 2)),
25                       sppcode = toupper(paste0(substr(genus, 1, 3),
26                                                  substr(species, 1, 3))))
27   return(df2)
28 }
```

Next steps for package development

9. Update DESCRIPTION

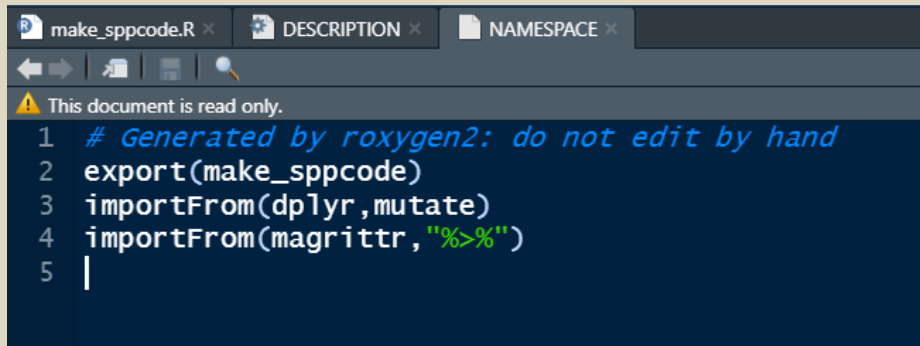


```
1 Package: test
2 Type: Package
3 Title: Test: a test package
4 Version: 0.1.0
5 Author: K. Miller
6 Maintainer: The package maintainer <kathryn_miller@nps.gov>
7 Description: This package is a test package for training purposes
8 License: Public Domain
9 Encoding: UTF-8
10 LazyData: true
11 RoxygenNote: 7.1.1
12 Depends:
13     R(>= 4.0)
14 Collate:
15     'make_sppcode.R'
16 Imports:
17     dplyr,
18     magrittr
19 Suggests:
20     stringr
21
22
```

Next steps for package development

10. Build/Rebuild package and check functions and documentation

- NAMESPACE



A screenshot of an R editor window showing the NAMESPACE file. The window has tabs for 'make_sppcode.R', 'DESCRIPTION', and 'NAMESPACE'. The NAMESPACE file contains the following code:

```
1 # Generated by roxygen2: do not edit by hand
2 export(make_sppcode)
3 importFrom(dplyr, mutate)
4 importFrom(magrittr, "%>%")
5 |
```

- Skim .Rd files



A screenshot of the R Documentation window for the 'make_sppcode' function. The window title is 'R: make_sppcode: Convert Latin Name to 6-letter code'. The content includes the following sections:

- make_sppcode {test}**
- make_sppcode: Convert Latin Name to 6-letter code**
- 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.
- Usage**
`make_sppcode(df)`
- Arguments**
`df` Specify the name of the data frame
- Value**
returns a data frame with genus, species and sppcode columns
- Examples**

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

[Package test version 0.1.0 [Index](#)]

- 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")

```
1  
2 devtools::install_github("KateMMiller/test")  
3 library(test)  
4 ex_df <- data.frame(Latin_Name = c("Abies balsamea", "Acer rubrum",  
5                                   "Betula", "Pinus strobus"))  
6  
7 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>