

# **Software Design, Development, and Distribution in R**

Lindsey Dietz, PhD<sup>1</sup>    Christina Knudson, PhD<sup>2</sup>

<sup>1</sup>Financial Economist, Federal Reserve Bank of Minneapolis

<sup>2</sup>Asst. Professor of Statistics, University of Saint Thomas

2020-10-30

## Dr. Dietz's Disclaimer

The views expressed in this presentation are strictly my own. They do not necessarily represent the position of the Federal Reserve Bank of Minneapolis or the Federal Reserve System.

# About Us

- ▶ Met in our Statistics PhD program in 2011
- ▶ Co-organizers of R Ladies-Twin Cities and the noRth conference
- ▶ Cyclists & coffee lovers



# Objectives of this talk

```
library(MyFirstPackage)

# Best practices for R package design
design_package()

# Building an R package
build_package()

# Distribution of your R package
distribute_package()
```

# Best practices for R package design



You nailed it.

# What is a design document

# Why use a design document

- ▶ splits the work into two steps (thinking and coding)
- ▶ makes you explain everything in detail (so you can know if you understand the details)
- ▶ helps you predict problems/tricky points
- ▶ helps you divide the work into reasonable modules so you can split it between days or people and make sure it will come together seamlessly
- ▶ Helps future you/developers understand what you had done so that you can create improvements or additions

# What to include in your design document

- ▶ goal of each function
- ▶ inputs and outputs of each
- ▶ flow chart between functions
- ▶ calculations/equations
- ▶ any tricky points
- ▶ numerical stability considerations
- ▶ how you will approach each function, including pseudo code if it's nontrivial
- ▶ tests you will implement (again goals, details)
- ▶ Helpful sketches
- ▶ Major updates
- ▶ Things you want to add/change in the future

# Building an R package

How it started



How it's going



## Create the Package

Building a package used to take a lot of expert knowledge. However, several R packages exist that now make the process extremely accessible.

```
#install.packages('testthat', 'devtools',
#                  'roxygen2', 'usethis')
library(testthat)
library(devtools)
library(roxygen2)
library(usethis)

usethis::create_package("~/MyFirstPackage")
# usethis::use_testthat() #if you want to add tests
usethis::use_git() #if you want to use Git
```

## Add your functions

```
usethis::use_r('target_psrf')
usethis::use_r('minESS')
```

# Write (or copy/paste) functions

```
target_psrf <- function(m, p, alpha = 0.05, epsilon = 0.05) {  
  
  # Calculate the minimum effective sample size for the given input parameters  
  Tee <- as.numeric(minESS(p = p, alpha = alpha, epsilon = epsilon))  
  
  # Calculate PSRF  
  psrf <- sqrt(1 + m / Tee)  
  
  return(list(psrf = psrf, epsilon = epsilon))  
}
```

# Add documentation for functions - Roxygen

```
'@title Target potential scale reduction factor (PSRF)
#' @description This function calculates the target PSRF for a set of MCMC chains.
#' This is adapted from the more complex version in stableGR
#' @param m Number of MCMC chains, e.g. 3 chains implies m = 3
#' @param p Number of parameters being sampled, e.g. (beta1, beta2, beta3) implies p = 3
#' @param alpha Significance level used to compute ESS; defaults to alpha = 0.05 i.e. 5%
#' @param epsilon Relative precision term; fixing all other elements,
#'                   as precision is set smaller, sample size increases; defaults to 0.05
#' @examples
#' target_psrf(m = 2, p = 2, alpha = 0.05, epsilon = 0.05)
#' target_psrf(m = 5, p = 2, alpha = 0.10, epsilon = 0.05)
#' @export target_psrf
#' @references D. Vats and C. Knudson. Revisiting the Gelman-Rubin Diagnostic.
#'             https://arxiv.org/abs/1812.09384

target_psrf <- function(m, p, alpha = 0.05, epsilon = 0.05) {

  # Calculate the minimum effective sample size for the given input parameters
  Tee <- as.numeric(minESS(p = p, alpha = alpha, epsilon = epsilon))

  # Calculate PSRF
  psrf <- sqrt(1 + m / Tee)

  return(list(psrf = psrf, epsilon = epsilon))
}
```

# Add documentation for functions - Roxygen

The screenshot shows the RStudio interface with the following details:

- Top Bar:** Files, Plots, Packages, Help, Viewer.
- Toolbar:** Back, Forward, Home, Find, Search bar containing "target\_psrf".
- Text Area:** Shows the R code for the target\_psrf function and its documentation.
- Code:**

```
target_psrf {MyFirstPackage}
target_psrf(m, p, alpha = 0.05, epsilon = 0.05)
```
- Description:**

This function calculates the target PSRF for a set of MCMC chains. This is adapted from the more complex version in stableGR
- Usage:**

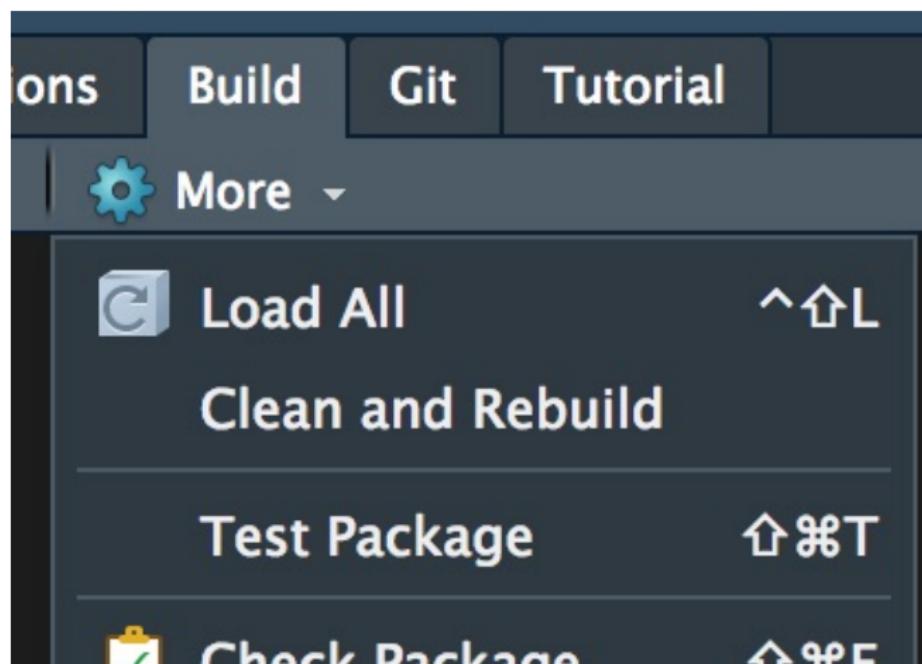
`target_psrf(m, p, alpha = 0.05, epsilon = 0.05)`
- Arguments:**
  - m**: Number of MCMC chains, e.g. 3 chains implies m = 3
  - p**: Number of parameters being sampled, e.g. (beta1, beta2, beta3) implies p = 3
  - alpha**: Significance level used to compute ESS; defaults to alpha = 0.05 i.e. 5%
  - epsilon**: Relative precision term; fixing all other elements, as precision is set smaller, sample size increases; defaults to 0.05
- References:**

D. Vats and C. Knudson. Revisiting the Gelman-Rubin Diagnostic.  
<https://arxiv.org/abs/1812.09384>

## Check, build, and install your package

```
devtools::check()  
devtools::build()  
devtools::install()
```

Or use the build tools in Rstudio



# Customize!

- ▶ Add tests
- ▶ Add other package dependencies
- ▶ Add vignettes



# Distribution of Your R Package



# Why Distribute Your R Package?

From David Robinson's excellent talk at rstudio::conf(2019)

## How I used to think of my goals:



## How I should have been thinking of them:



# Why Distribute Your R Package?

- ▶ Gains in usership/citations for those in the public domain (academics, nonprofits)
- ▶ Gains in productivity for those in private industries
- ▶ Saving future you time

# Using Version Control



*Ann, you're beautiful and you're organized!*

# Using Version Control

- ▶ This is not a Git talk, but Git has become a dominant version control technique so we are demo-ing our work on Github
- ▶ An amazing and free resource for R users is Jenny Brian's book:  
<https://happygitwithr.com/>

How this is useful to audit Production version control - put R in prod Focus more on why to use - maybe RStudio easy integration with github

How this is useful to audit/third parties Easy to track changes over time Issue driven development (working with teams)  
<https://github.com/tidyverse/dplyr/issues> Production version control - put R in prod reference to talk

# Where to Find Us

## *Lindsey Dietz*

- ▶ Email: [lindseydietz13@gmail.com](mailto:lindseydietz13@gmail.com)
- ▶ LinkedIn: <https://www.linkedin.com/in/lindseydietz/>
- ▶ Twitter: <https://twitter.com/lindseyditz13>

## *Christina Knudson*

- ▶ Email: [knud8583@stthomas.edu](mailto:knud8583@stthomas.edu)
- ▶ LinkedIn: <https://www.linkedin.com/in/christinaknudson/>
- ▶ Twitter: <https://twitter.com/canoodleson>