Software Design, Development, and Distribution in R

Lindsey Dietz, PhD[1] Christina Knudson, PhD[2]

[1]Financial Economist, Federal Reserve Bank of Minneapolis

[2]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

- Friends since meeting in our Statistics PhD program in 2011
- Co-organizers of R Ladies-Twin Cities and the noRth conference
- Both 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

What is a design document?

- A blueprint or recipe for your project with plenty of details
- Doc written with the understanding that future-you will forget these details otherwise

Why use a design document? (1 of 5)

- Watch this video first!

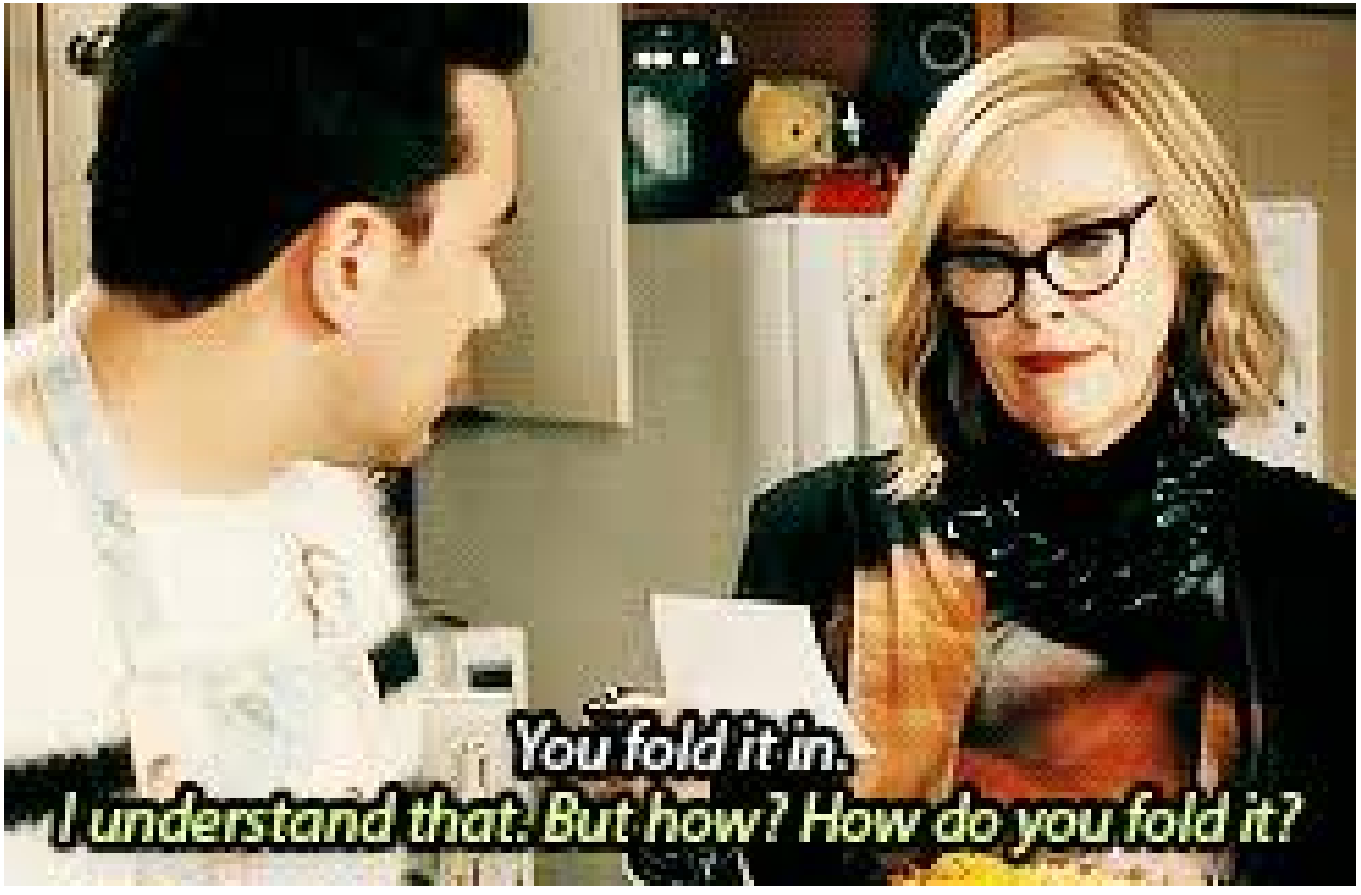
Why use a design document? (2 of 5)

- Separates thinking and coding

Imagine creating enchiladas without a recipe!

Why use a design document? (3 of 5)

- Forces you to explain everything in detail
- Helps you predict problems and tricky points



Why use a design document? (4 of 5)

- Helps you divide the work into reasonable modules so you can split it between days or people and make sure it it will come together seamlessly

No need to ladle and stir at the same time

Why use a design document? (5 of 5)

- Helps future you/developers understand what you had done so that you can create improvements or additions
- Helps you remember everything you'll inevitably forget

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 (and some pseudo code)
- Tests you will implement (again goals, details)
- Helpful sketches
- Major updates
- Things you want to add/change in the future

Examples of Design Docs

- glmm
- stableGR

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')
```

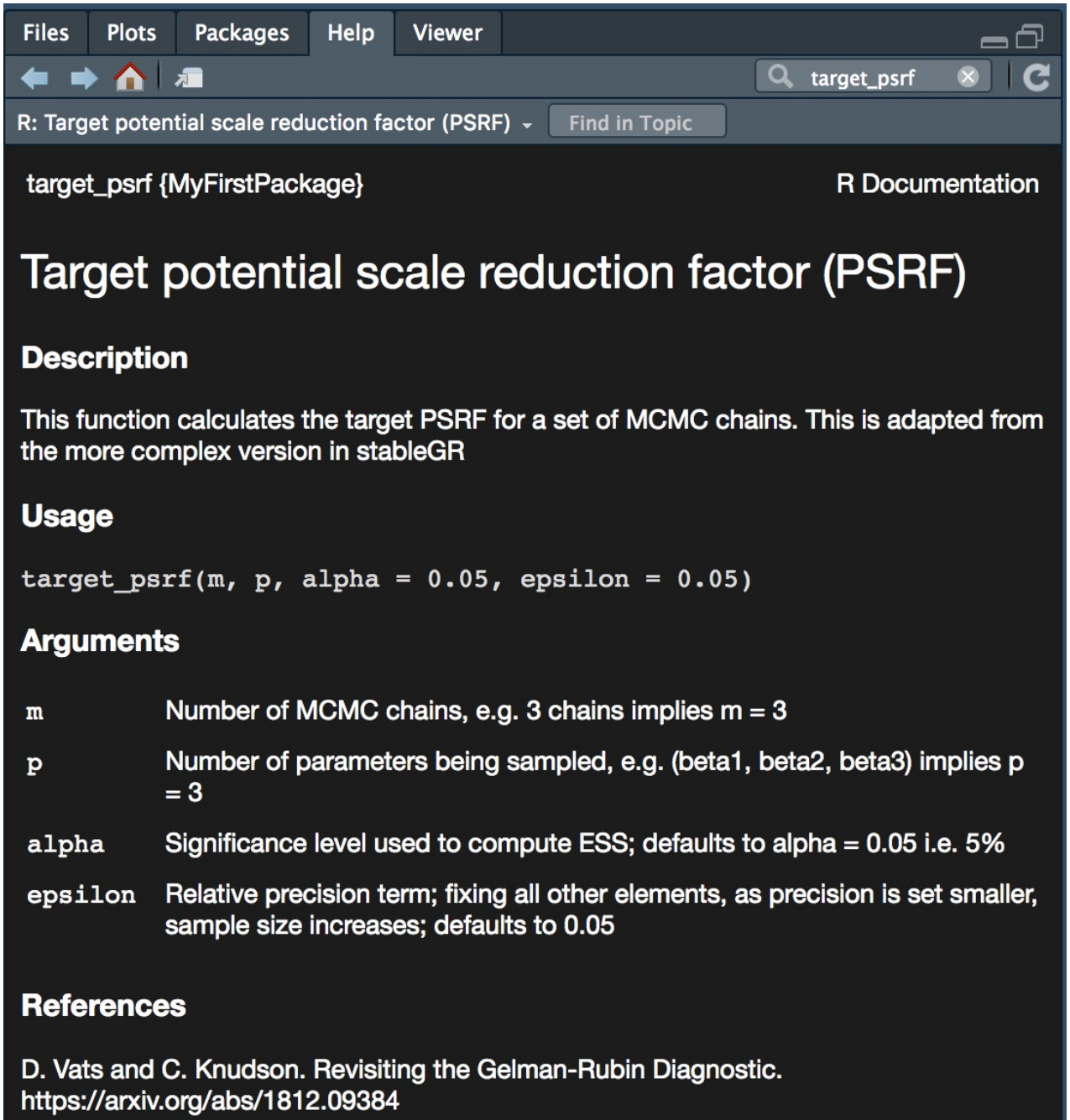
Write (or copy/paste) functions

```r
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

```r
#' @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))
}
```

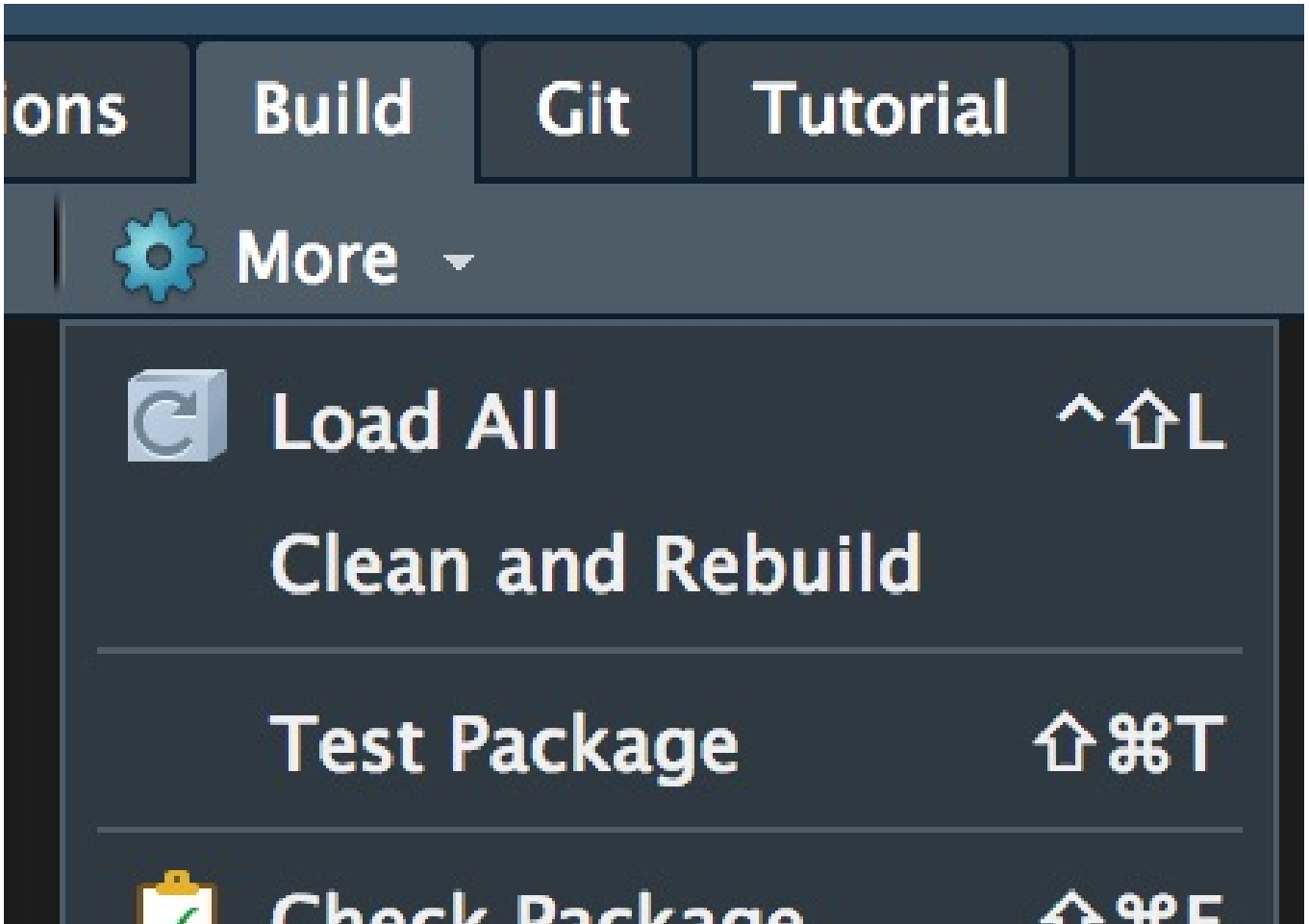
Add documentation for functions - Roxygen



Check, build, and install your package

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

Or use the build tools in Rstudio



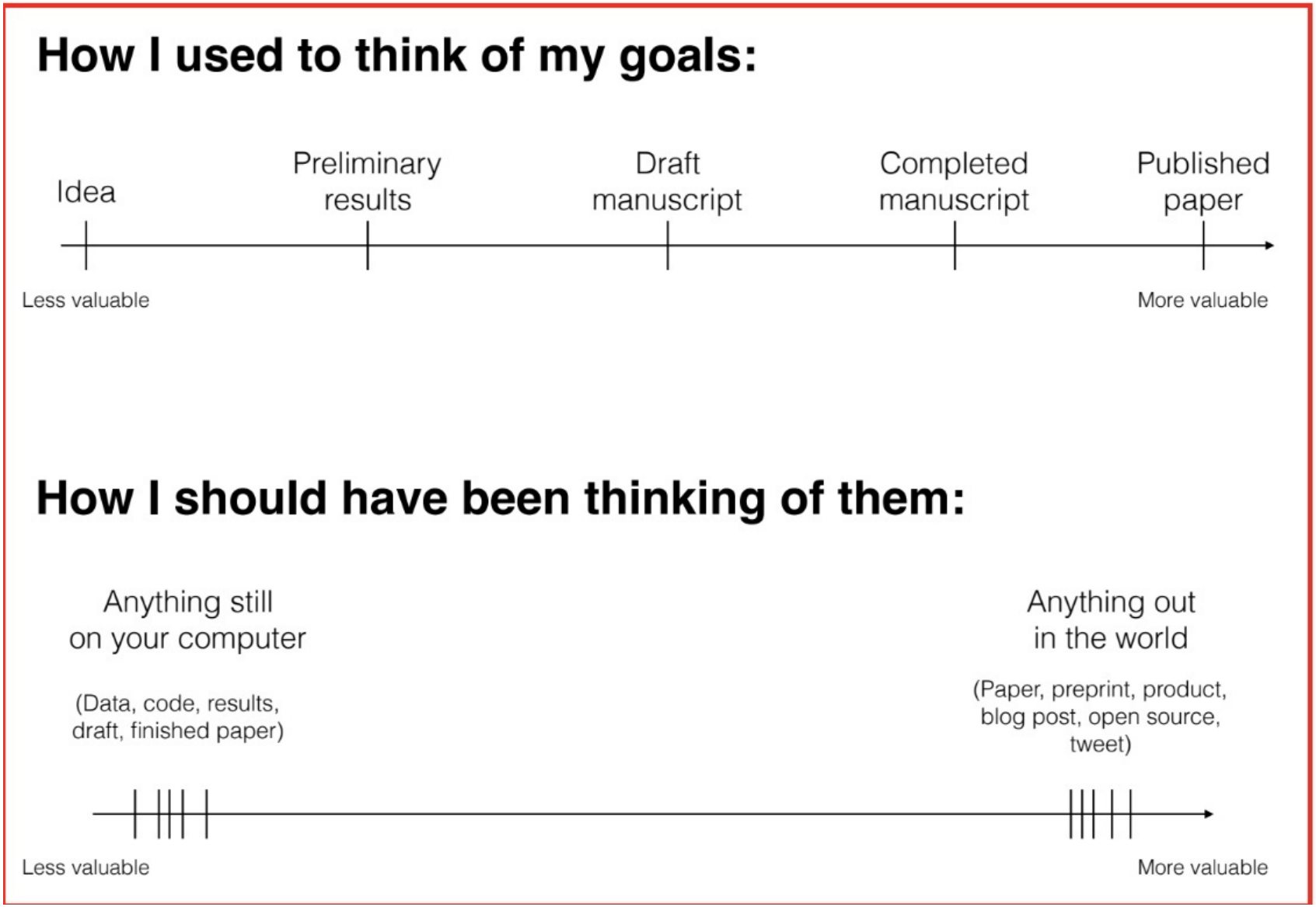
Customize!

- 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)



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

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

Find this presentation at: https://github.com/knudson1/WiADS2020/blob/master/doc/Presentation_Slides.pdf

Lindsey Dietz

- Email: lindseydietz13@gmail.com
- LinkedIn: https://www.linkedin.com/in/lindseydietz/
- Twitter: https://twitter.com/lindseydietz13

Christina Knudson

- Email: knud8583@stthomas.edu
- LinkedIn: https://www.linkedin.com/in/christinaknudson/
- Twitter: https://twitter.com/canoodleson