

Data Tools in R

Emily Riederer
emilyriederer@gmail.com
[@emilyriederer](#)

About Emily



R hackathon co-organizers, Feb 2019

UNC Chapel Hill (2012 - 2016)

B.S. Mathematics, Statistics & Analytics



US Card Analytics (2016 - Present)

Analyst to Senior Manager

Founded Data & Analytics Tooling team

Hobbies

#rstats Twitter (@emilyriederer)

Blogging at <https://emilyriederer.netlify.app/>

rOpenSci editor

CRC Press reviewer



Data

- Datamart design
- Data pipeline development
- Data quality
- Data discoverability

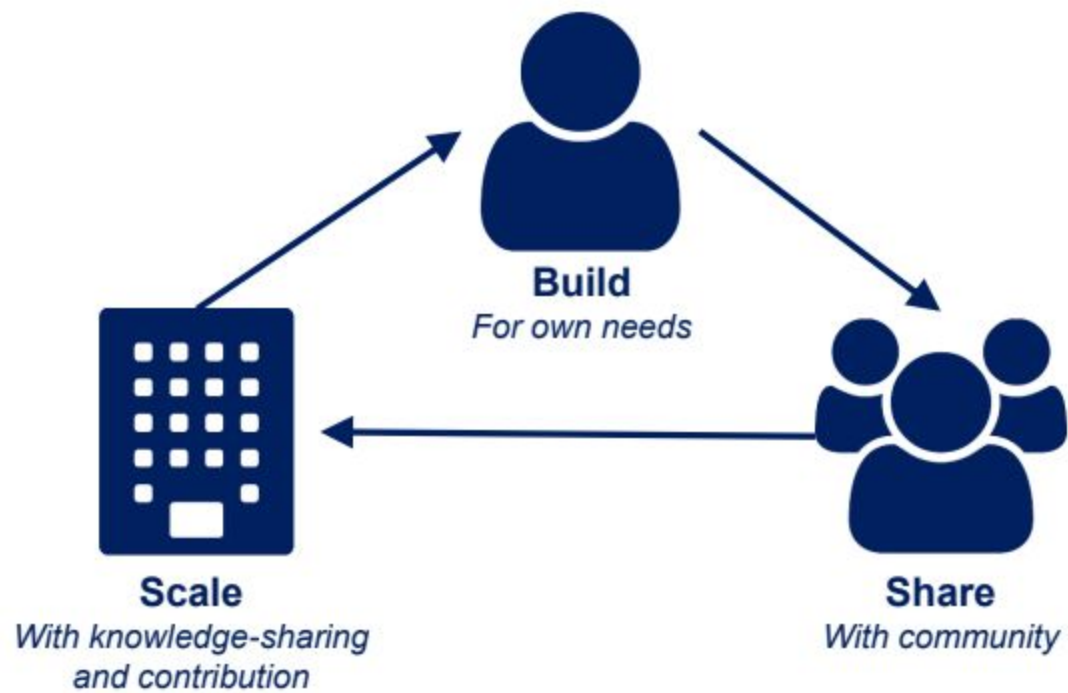
Analysis Tools

- Conceptual frameworks & R packages for common analytical tasks
- Spanning utilities, devtools, analysis

Community

- Training
- Consulting / mentorship
- Hackathons

Bring reproducibility and extensibility to business analysis



My career path is predicated on very poor assumptions

High School

Math

Expectation

Reality



University

Statistical Modeling

Apply the same rigor and certainty
as math to “human” problems

Art more than science



Industry

Data Science / Analytics

Apply rich statistical methods to
make decisions and find “insights”

Data, infrastructure, context

My career path gave me an eclectic set of ideas that proved quite useful

High School

Math

- Rigor and precision in defining abstractions
- Proving things (to myself!)
- Literacy to keep learning



University

Statistical Modeling

- Toolbag of methods
- Distribution thinking
- Unbridled fury at bad workflows



Industry

Data Science / Analytics

- Working with technology
- Exciting others about new, niche ideas

Key career realizations

1. R Markdown is the gateway to more powerful tools
2. Good workflows provide an incredible amount of leverage
3. Packages are more than functions on CRAN
4. Packages can play many roles in an organization

Key career realizations

1. R Markdown is the gateway to more powerful tools
2. Good workflows provide an incredible amount of leverage
3. Packages are more than functions on CRAN
4. Packages can play many roles in an organization

**R Markdown Driven
Development**

**R Packages in
Organizations**

Beyond R Markdown's polished outputs, it's also a powerful prototyping platform



```
---
title: "My Analysis"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = FALSE)
```

```{r pkg-load}
library(dplyr)
library(tidyr)
library(survival)
library(ggfortify)
```

```{r data-load}
outcomes_df <- readr::read_csv('outcomes.csv')
```

## Introduction

In this analysis, we report the...

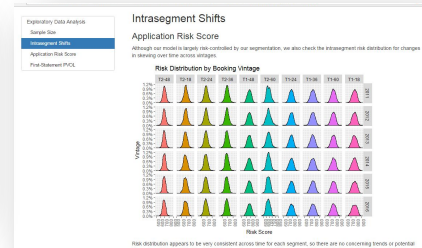
```{r all-the-good-code}
```

## Dashboards

Region	2014	2015	2016	2017	2018	2019
Region 1	10%	12%	14%	16%	18%	20%
Region 2	8%	10%	12%	14%	16%	18%
Region 3	6%	8%	10%	12%	14%	16%
Region 4	4%	6%	8%	10%	12%	14%
Region 5	2%	4%	6%	8%	10%	12%
Region 6	1%	2%	4%	6%	8%	10%

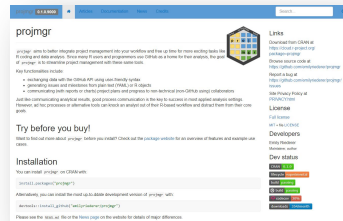
flexdashboard  
plotly + crosstalk

## Analysis Reports



knitr  
rmarkdown

## Websites



pkgdown

## Slides



xaringan

# Each analysis depends on a latent tool custom-fit to your domain-specific workflow

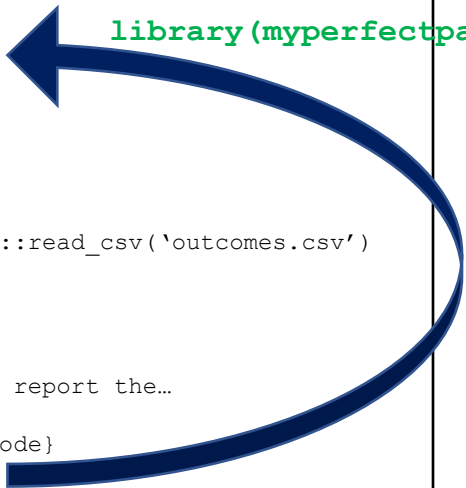


```

title: "My Analysis"
output: html_document

```${r setup, include=FALSE}  
knitr::opts_chunk$set(echo = FALSE)  
```${r pkg-load}  
library(dplyr)
library(tidyverse)
library(survival)
library(ggfortify)
```${r data-load}  
outcomes_df <- readr::read_csv('outcomes.csv')  
```${r all-the-good-code}
```

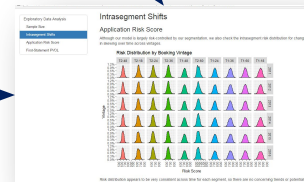
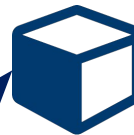
library(myperfectpackage)



```

title: "My Analysis"
output: html_document

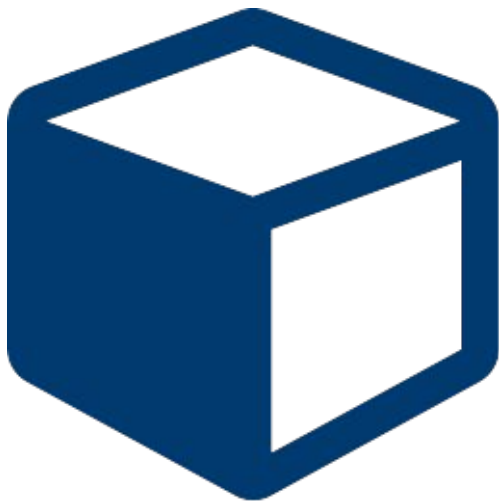
```${r setup, include=FALSE}  
knitr::opts_chunk$set(echo = FALSE)  
```${r pkg-load}  
library(dplyr)
library(tidyverse)
library(survival)
library(ggfortify)
```${r data-load}  
outcomes_df <- readr::read_csv('outcomes.csv')  
```${r all-the-good-code}
```



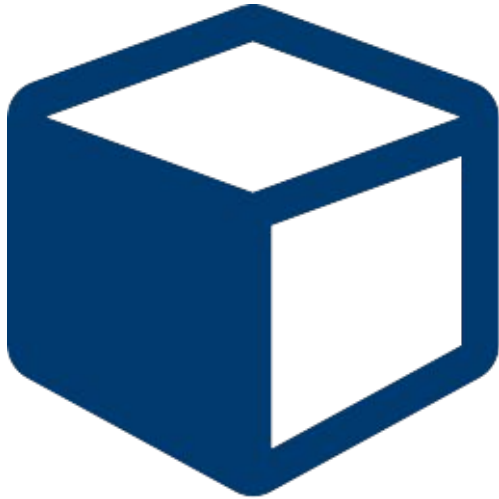
**This implicit analysis tool already contains core development and design components**



- Curated set of related libraries
- Working and “tested” code



# This implicit analysis tool already contains core development and design components



- Curated set of related libraries
- Working and “tested” code

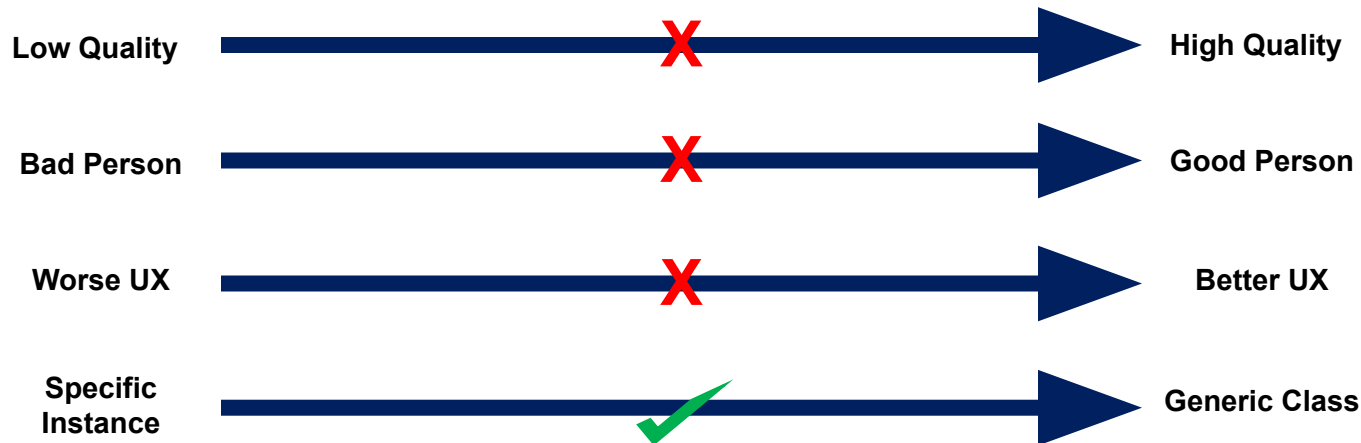
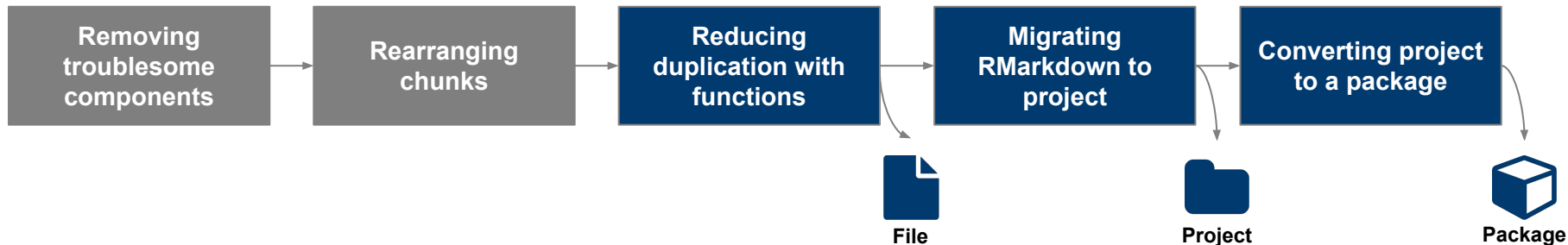


- Comprehensive understanding of user (producer & consumer) requirements
- Sane workflow
- Complete & compelling example

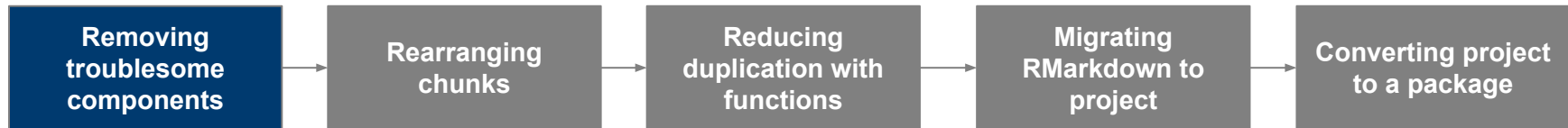
## RMarkdown Driven Development (RmdDD) has five main steps



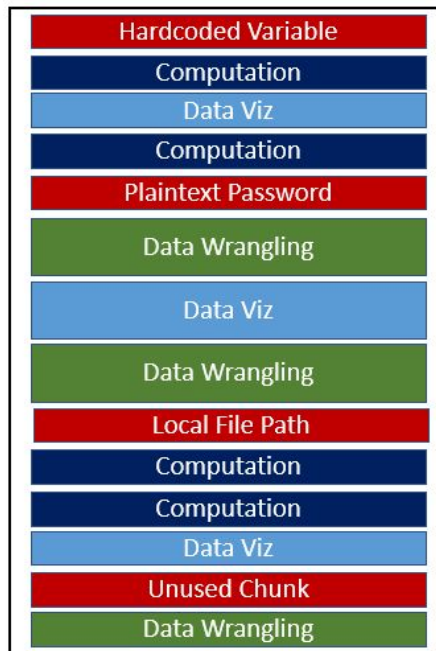
# RmdDD has multiple endpoints, so you can take the right exit ramp for your destination



# Eliminate clutter to make your own code more trustworthy for its initial use



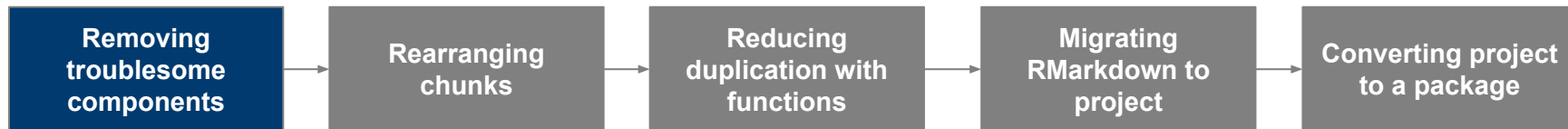
“Dirty” RMarkdown



Original RMarkdown



# Parameters can protect the integrity of your analysis and your credentials



## “Dirty” RMarkdown

Hardcoded Variable
Computation
Data Viz
Computation
Plaintext Password
Data Wrangling
Data Viz
Data Wrangling
Local File Path
Computation
Computation
Data Viz
Unused Chunk
Data Wrangling

```

title: "My Analysis"
output: html_document

{{package loads, data loads, etc.}}

```{r}  
data_lastyr <- data %>%  
  filter(between(date, '2018-01-01', '2018-12-31'))  
```
```

```

title: "My Analysis"
output: html_document
params:
 start: '2018-01-01'
 end: '2018-12-31'

{{package loads, data loads, etc.}}

```{r}  
data_lastyr <- data %>%  
  filter(between(date, params$start, params$end))  
```
```




# Parameters can protect the integrity of your analysis and your credentials



## “Dirty” RMarkdown

|                    |
|--------------------|
| Hardcoded Variable |
| Computation        |
| Data Viz           |
| Computation        |
| Plaintext Password |
| Data Wrangling     |
| Data Viz           |
| Data Wrangling     |
| Local File Path    |
| Computation        |
| Computation        |
| Data Viz           |
| Unused Chunk       |
| Data Wrangling     |



```

title: "My Analysis"
output: html_document
params:
 username: emily
 password: x

{{package loads, data loads, etc.}}

```${r}  
con <-  
  connect_to_database(  
    username = params$username,  
    password = params$password  
  )  
```${r}
```



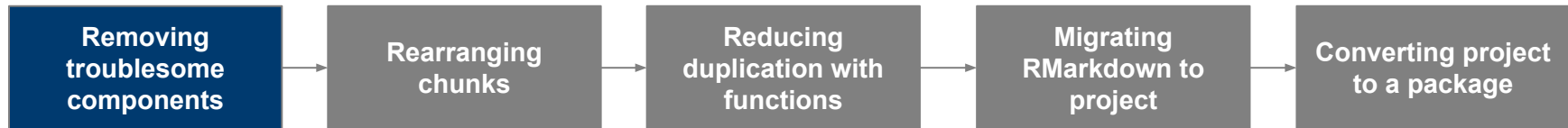
## RStudio: Knit > Knit with Parameters...

username  
emily

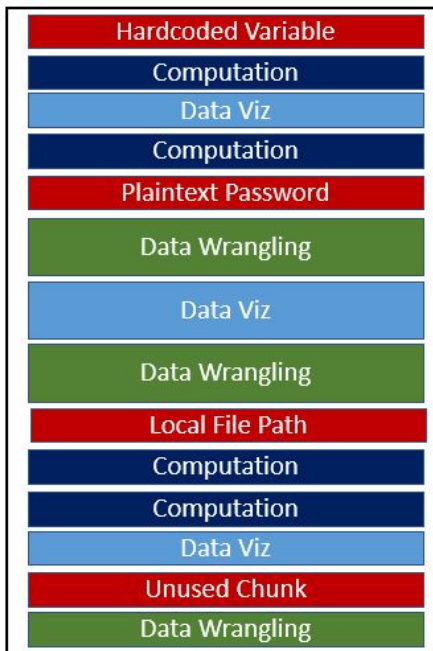
password  
x

Cancel Knit

# Local file paths nearly guarantee that your project will not work on someone else's machine



## “Dirty” RMarkdown



Not resilient to any file structure change:

```
data <- readRDS('C:\Users\me\Desktop\my-project\data\my-data.rds')
```



Resilient to movement of *working directory*:

```
data <- readRDS('data\my-data.rds')
```

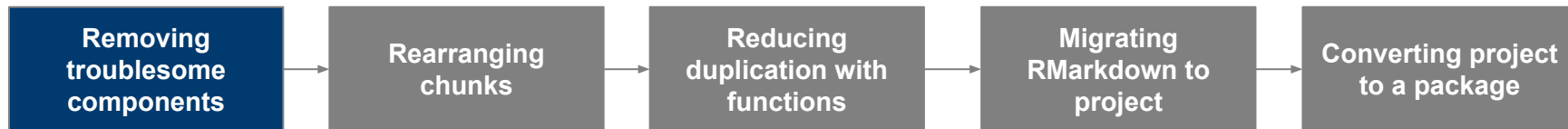


Resilient to movement of Rmd *within* working directory or across OS:

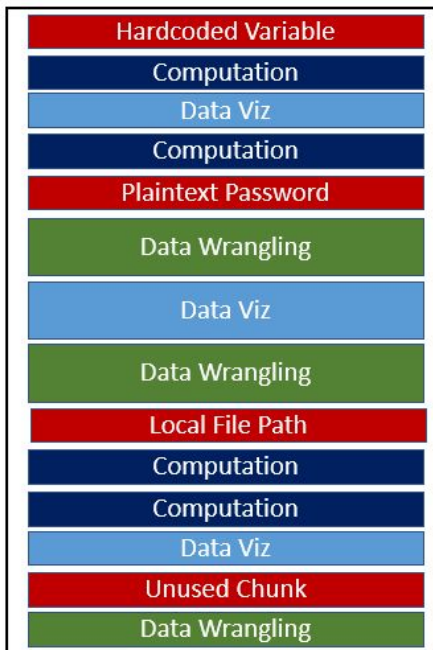
```
data <- readRDS(here::here('data', 'my-data.rds'))
```



# Don't let your script become a junk drawer



## “Dirty” RMarkdown



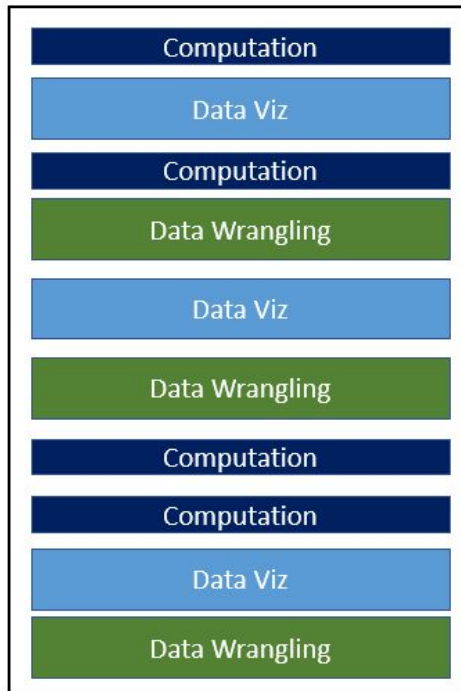
**X Unused package loads**

**X Unsuccessful coding experiments**

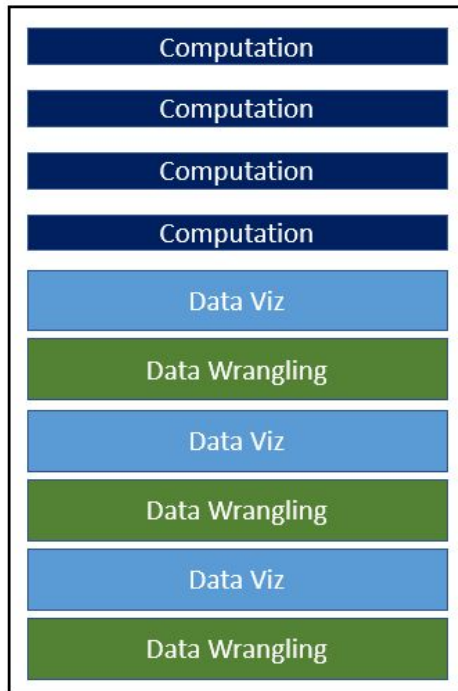
# RMarkdown is (too) good at capturing our non-linear thought processes



**Original RMarkdown**



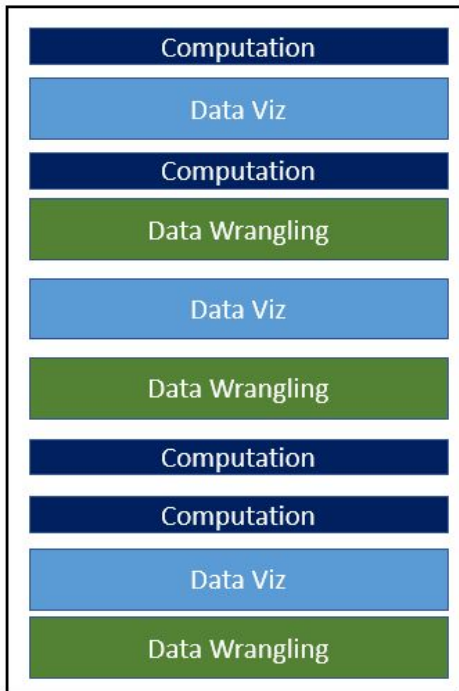
**Rearranged Chunks**



# Clustering quantitative and narrative components makes both easier to iterate on



Original RMarkdown



Rearranged Chunks



Infrastructure &  
Computing to the top



Communication &  
Narration to the bottom

- Clear dependencies
- Frontloaded errors

- Increased likelihood of noticing repeated code

- Consolidated story
- Easier for non-coder to contribute

# Enhance the navigability of your file in RStudio with chunk names and special comments



Named chunks create bookmark on nav bar and encourage semantically grouped chunks

```
1 title: "My Analysis"
2 output: html_document
3 ----
4
5
6 ### {r setup, include=FALSE}
7 knitr::opts_chunk$set(echo = TRUE)
8 ###
9
10 ### {r pkg-load}
11 ###
12
13 ### {r data-load}
14 ###
15
16 ### {r data-clean}
17 # apply data quality checks ----
18 # identify and remove outliers ----
19 # impute missing values ----
20 ###
21
22 ## Introduction
23
24 ## Exploratory Data Analysis
25
26 My Analysis
27
28 Chunk 1: setup
29 Chunk 2: pkg-load
30 Chunk 3: data-load
31 Chunk 4: data-clean
32
```

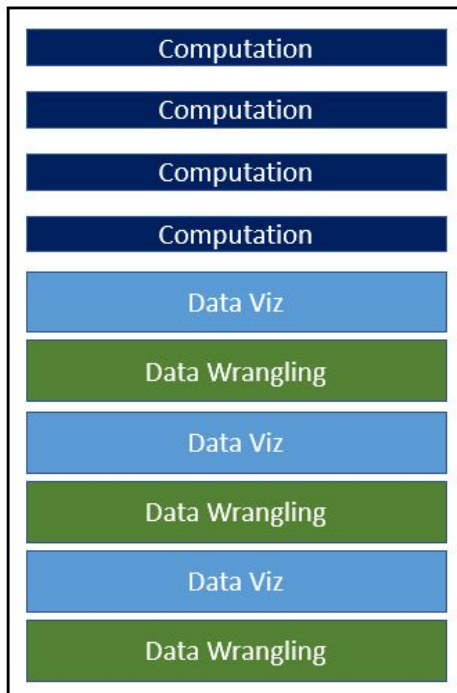
Expandable TOC allows you to jump to your Markdown headers (#)

Comments followed by four dashes create expand/contract button in margin and bookmark on nav bar

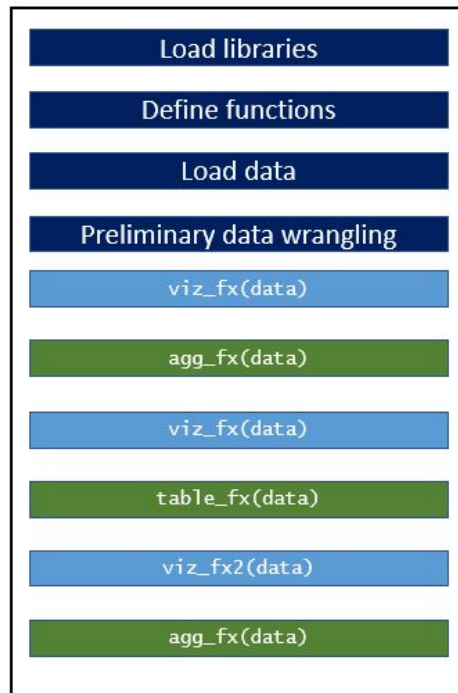
# Writing functions eliminates duplication and increases code readability



**Rearranged Chunks**



**Modularized Chunks**



# Writing functions eliminates duplication and increases code readability



```
Exploratory Analysis

````{r}  
ggplot(data, aes(x,y)) + geom_point()  
````  

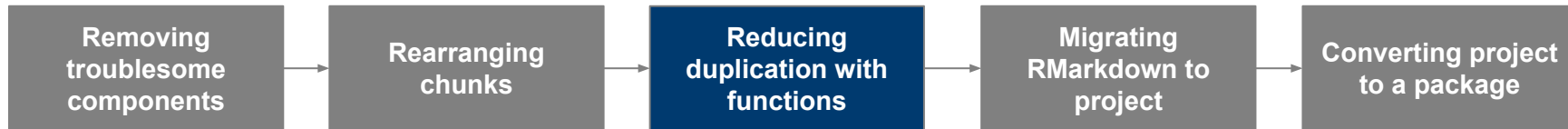
````{r}  
ggplot(data, aes(x,z)) + geom_point()  
````  

````{r}  
ggplot(data, aes(x,w)) + geom_point()  
````
```



```
````{r fx-viz-scatter-x}  
  
viz_scatter_x <- function(data, vbl) {  
  ggplot(  
    data = data,  
    mapping = aes(x = x, y = {{vbl}}) +  
    geom_point()  
  )  
  ...  
}  
  
## Exploratory Analysis  
  
````{r viz-scatter-x }  
viz_scatter_x(data, y)
viz_scatter_x(data, z)
viz_scatter_x(data, w)
````
```


roxygen2 function documentation can give your script a package-like understandability



```
```{r fx-viz-scatter-x}

#' Scatterplot of variable versus x
#'
#' @param data Dataset to plot. Must contain variable named x
#' @param vbl Name of variable to plot on y axis
#'
#' @return ggplot2 object
#' @import ggplot2
#' @export

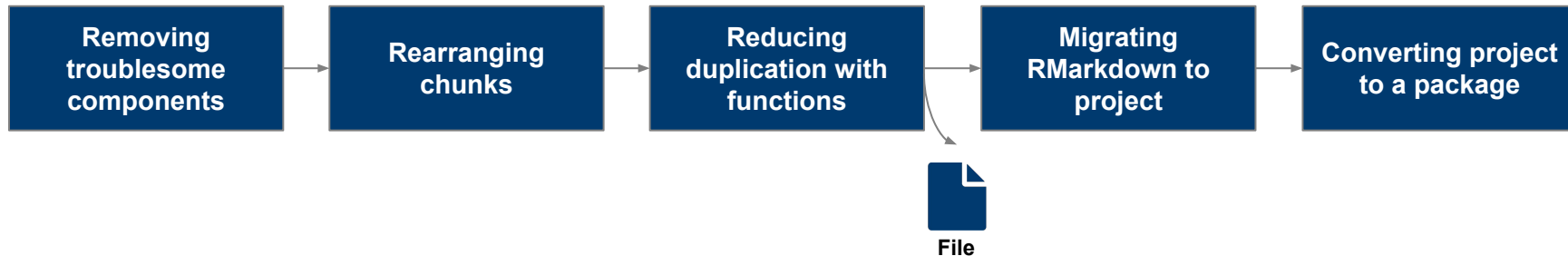
viz_scatter_x <- function(data, vbl) {
 ggplot(
 data = data,
 mapping = aes(x = x, y = {{vbl}}) +
 geom_point()
)
}

```
```



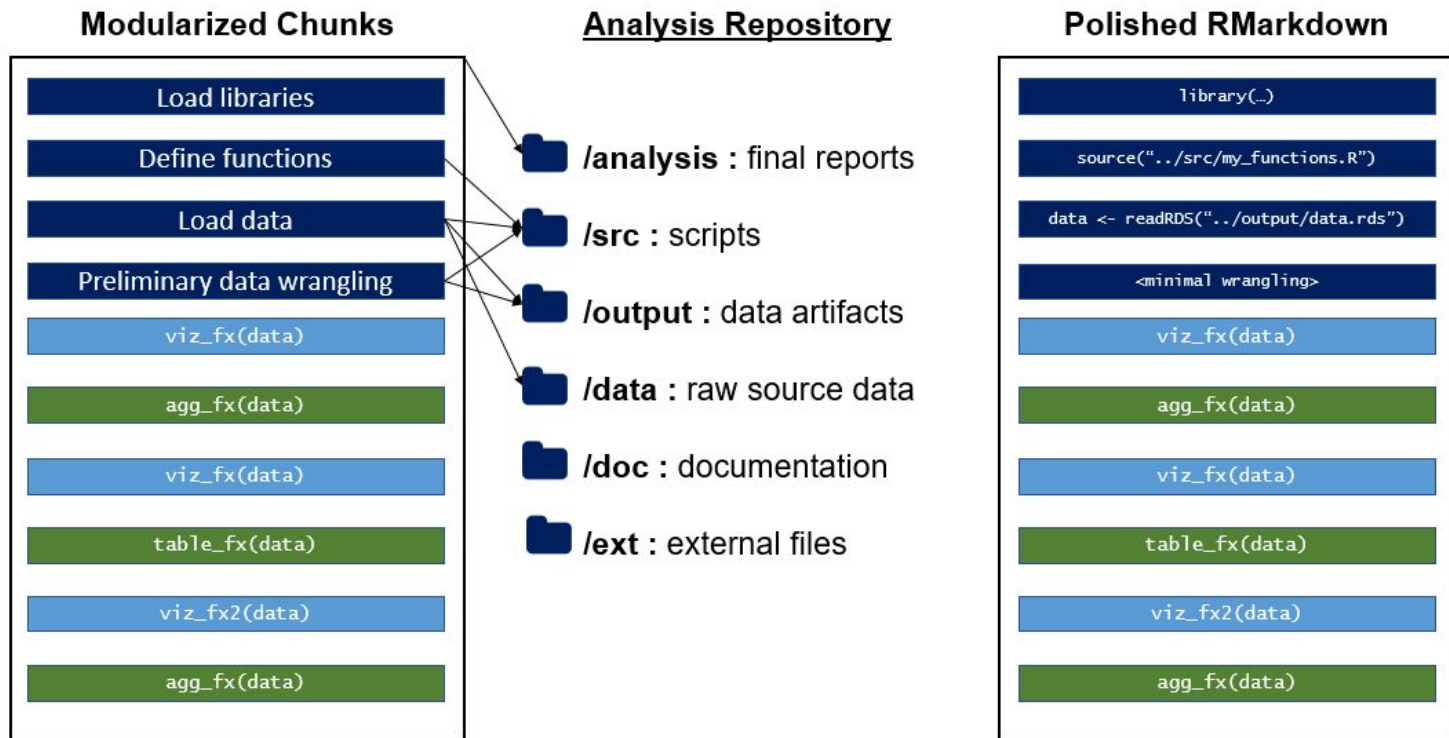
RStudio: Ctrl + Alt + Shift + R for skeleton

A polished single-file RMarkdown can be a very practical end-state for maximum portability



| Benefits | Pitfalls |
|--|---|
| <ul style="list-style-type: none">• Portable without formal repository• Easy to compare versions with diffs without formal version control• One-push execution / refresh | <ul style="list-style-type: none">• Can be lengthy, monolithic, and intimidating• Potentially slow to run and relies on RMarkdown to play role of job scheduler• Enables antipatterns (e.g. not saving artifacts) |

Projects modularize components and make it easy to access individual project assets



The source () function enables us to execute R code from another script



```
```${r fx-viz-scatter-x}
#' Scatterplot of variable versus x
#'
#' @param data Dataset to plot. Must contain variable named x
#' @param vbl Name of variable to plot on y axis
#'
#' @return ggplot2 object
#' @import ggplot2
#' @export
viz_scatter_x <- function(data, vbl) {
 ggplot(
 data = data,
 mapping = aes(x = x, y = {{vbl}}) +
 geom_point()
)
}
```

## Exploratory Analysis

```${r viz-scatter-x }
viz_scatter_x(data, y)
viz_scatter_x(data, z)
viz_scatter_x(data, w)
```
```



```
```${r load-fx}
source(here('src', 'viz-scatter-x.R'))
```

## Exploratory Analysis

```${r viz-scatter-x }
viz_scatter_x(data, y)
viz_scatter_x(data, z)
viz_scatter_x(data, w)
```
```



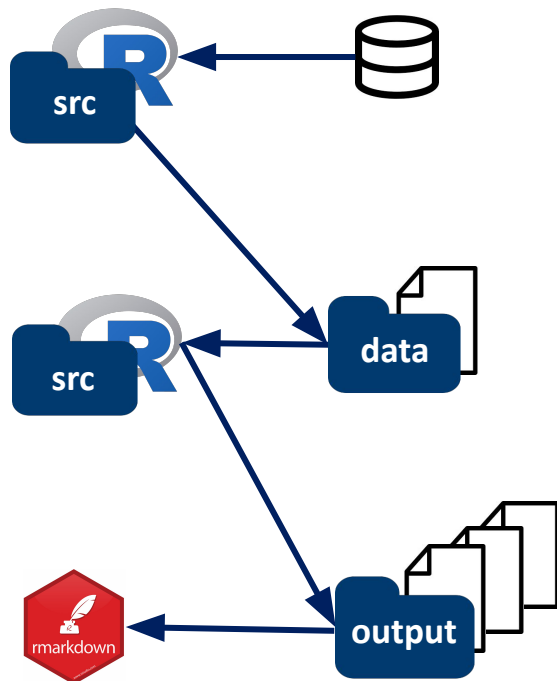
```
## Exploratory Analysis

```${r viz-scatter-x }
viz_scatter_x(data, y)
viz_scatter_x(data, z)
viz_scatter_x(data, w)
```

#' Scatterplot of variable versus x
#'
#' @param data Dataset to plot. Must contain variable named x
#' @param vbl Name of variable to plot on y axis
#'
#' @return ggplot2 object
#' @import ggplot2
#' @export
viz_scatter_x <- function(data, vbl) {
  ggplot(
    data = data,
    mapping = aes(x = x, y = {{vbl}}) +
    geom_point()
  )
}
```



Pre-processing data decreases external system dependencies and knitting time

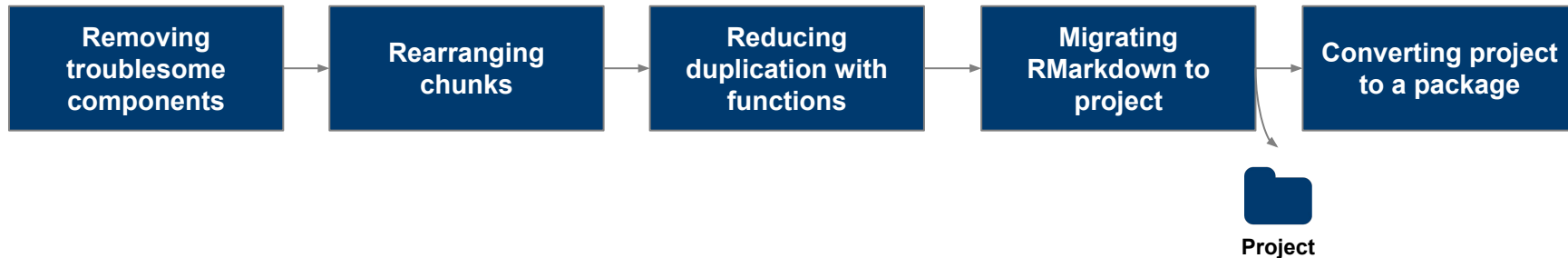


Load data outside of Rmd to eliminate dependence on API, Database, etc. being 'up' when need to knit

Store 'raw' data for posterity and reproducibility

Store analytical artifacts (e.g. lean models, aggregate data) to read in to final report

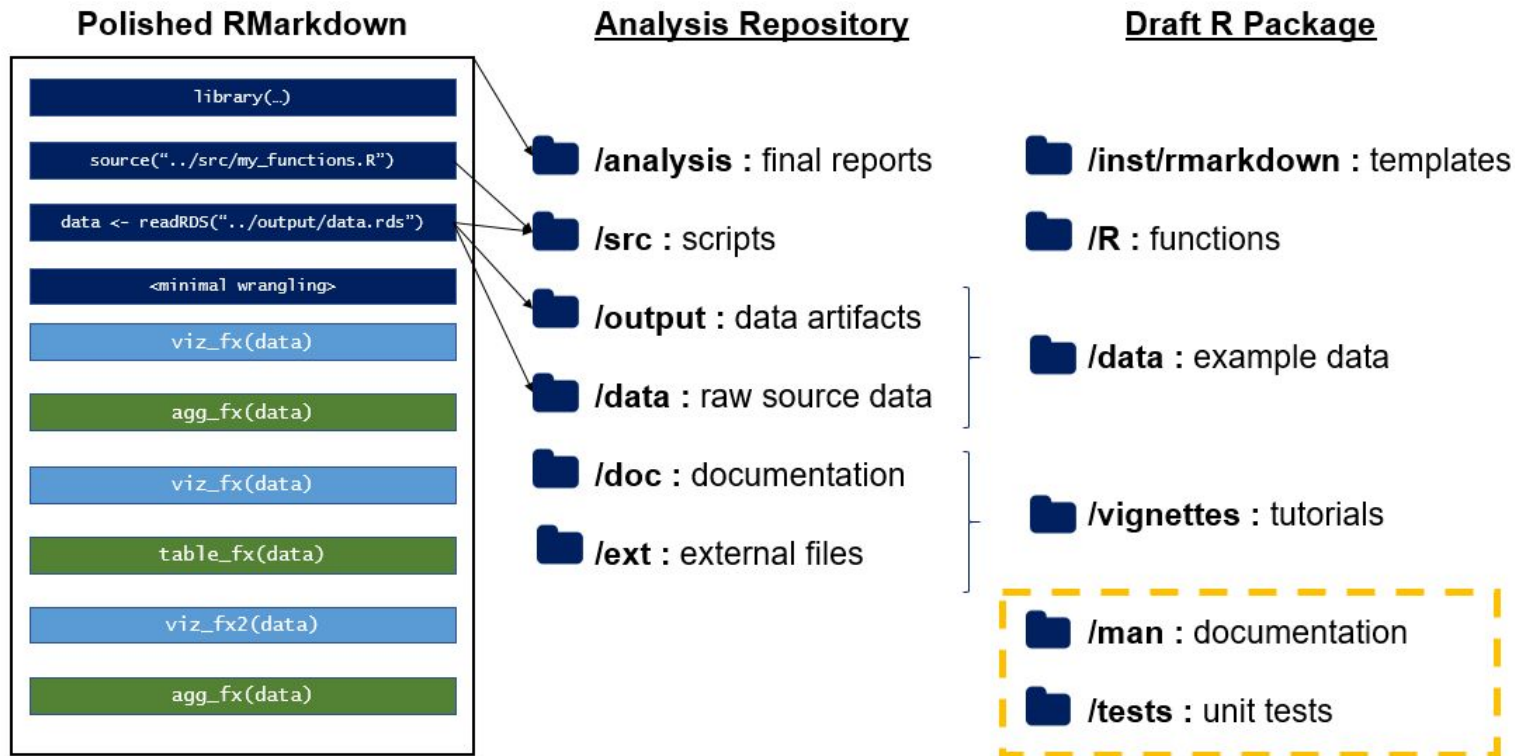
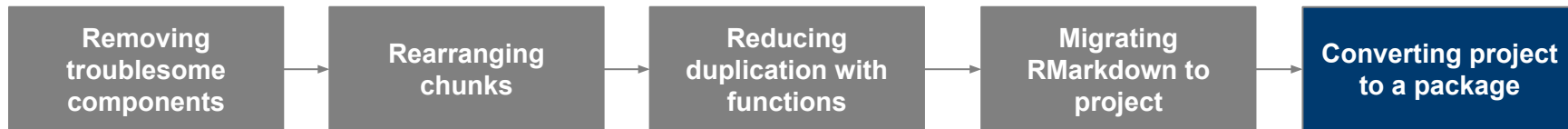
R projects preserve problem-specific context while making it easy to reapply components



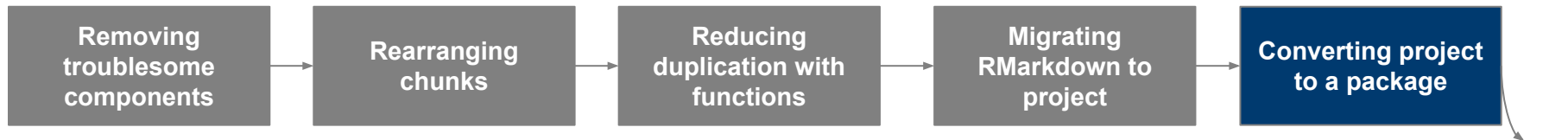
Project

| Benefits | Pitfalls |
|---|--|
| <ul style="list-style-type: none">• Flexible to extract small proportion of functionality or modify at will• Preserves problem-specific context (when desirable) | <ul style="list-style-type: none">• The line between analysis and code may be unclear• Can't make full use of developer tools |




There is a near one-to-one mapping between the components of a project and a package



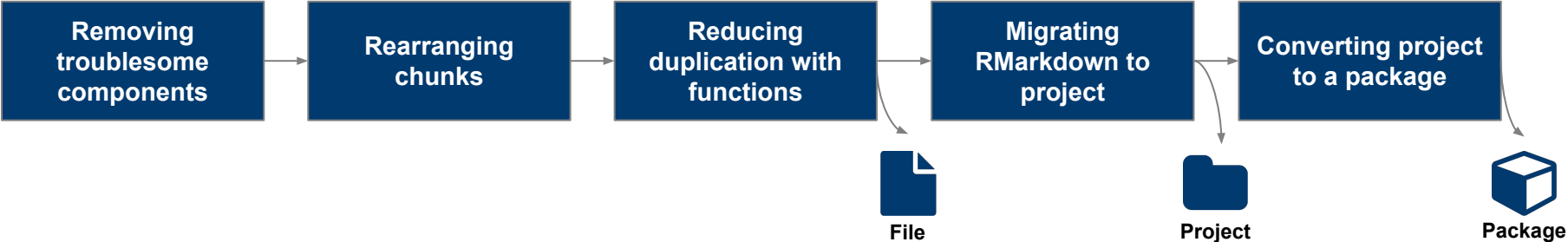
Developer tools exist to help us create everything we need – and more!



Different stopping points optimize for recreation versus extension of your work

| | Benefits | Pitfalls | |
|---|--|---|-------------------|
| 
Standalone File | <ul style="list-style-type: none">• Portable without formal repository• Easy to compare versions with diffs without formal version control• One-push execution / refresh | <ul style="list-style-type: none">• Can be lengthy, monolithic, and intimidating• Potentially slow to run and relies on RMarkdown to play role of job scheduler• Enables antipatterns (e.g. not saving artifacts) | Specific Instance |
| 
Project | <ul style="list-style-type: none">• Flexible to extract small proportion of functionality or modify at will• Preserves problem-specific context (when desirable) | <ul style="list-style-type: none">• The line between analysis and code may be unclear• Can't make full use of developer tools | |
| 
Package | <ul style="list-style-type: none">• Formal mechanisms for distributing at scale (e.g. CRAN)• Familiar format for others to learn and use | <ul style="list-style-type: none">• May be too narrowly focused and inflexible if built towards specific project• Potentially more challenging to extract specific features from for interactive use | Generic Class |

No matter what path you chose, your RMarkdown analysis is closer to a sustainable and empathetic data product than you may think!



Key career realizations

1. R Markdown is the gateway to more powerful tools
2. Good workflows provide an incredible amount of leverage
3. Packages are more than functions on CRAN
4. Packages can play many roles in an organization

**R Markdown Driven
Development**

**R Packages in
Organizations**



data access
server connection
proxies, ssh, ssl

right problems
tribal knowledge
intuition

team norms
meetings
communication



data access
server connection
proxies, ssh, ssl

right problems
tribal knowledge
intuition

team norms
meetings
communication



**Internal
Packages**

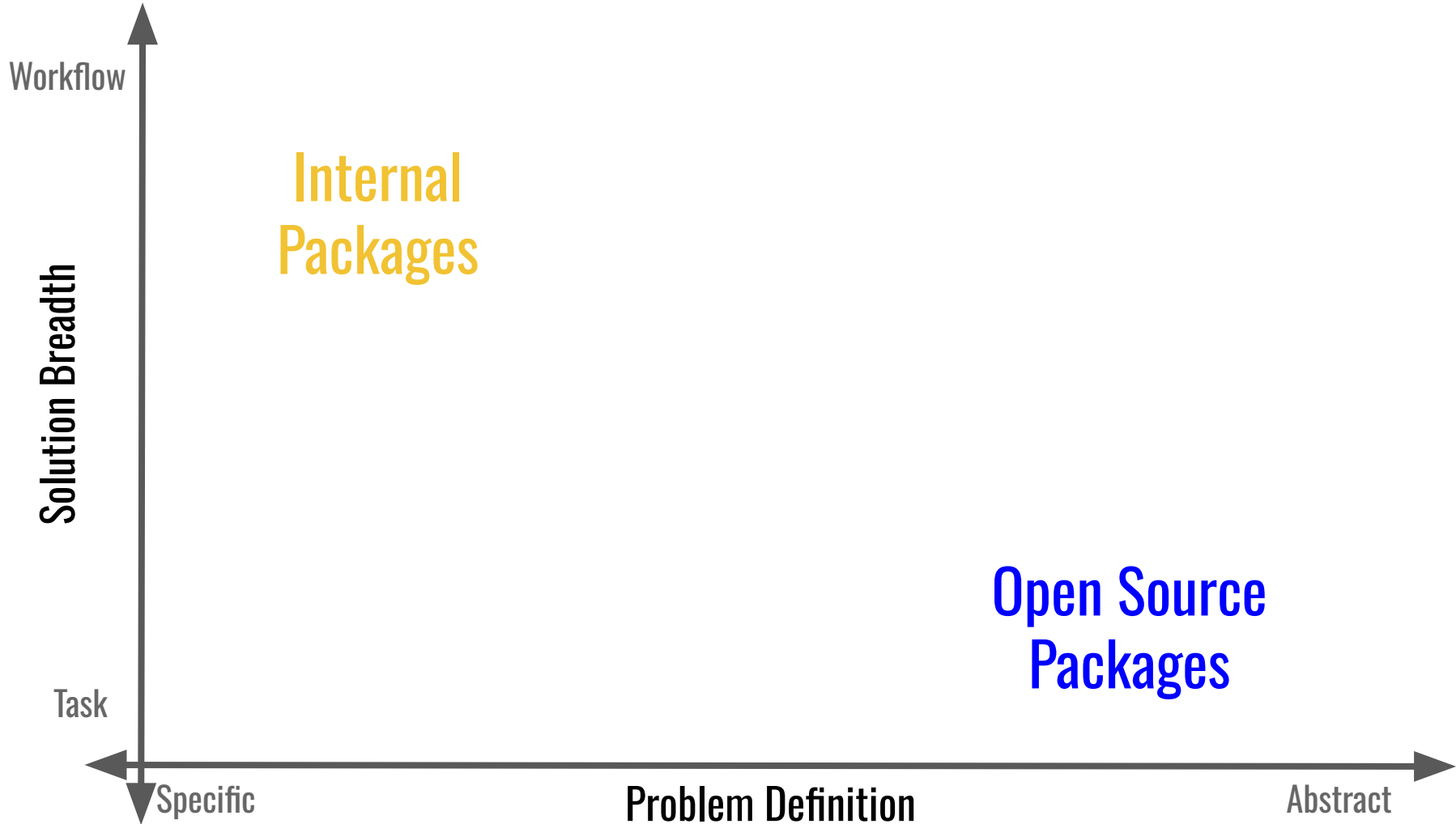
**Open Source
Packages**



Specific

Problem Definition

Abstract



utilities packages

data access
server connection
proxies, ssh, ssl

e.g. abstraction layer for
infrastructure

analysis packages

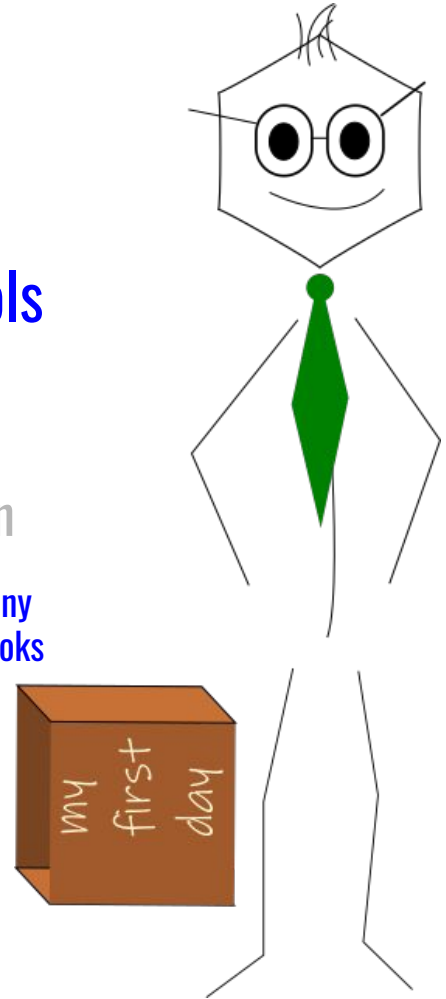
right problems
tribal knowledge
intuition

e.g. curated workflow, tailored
function calls, automated
result generation

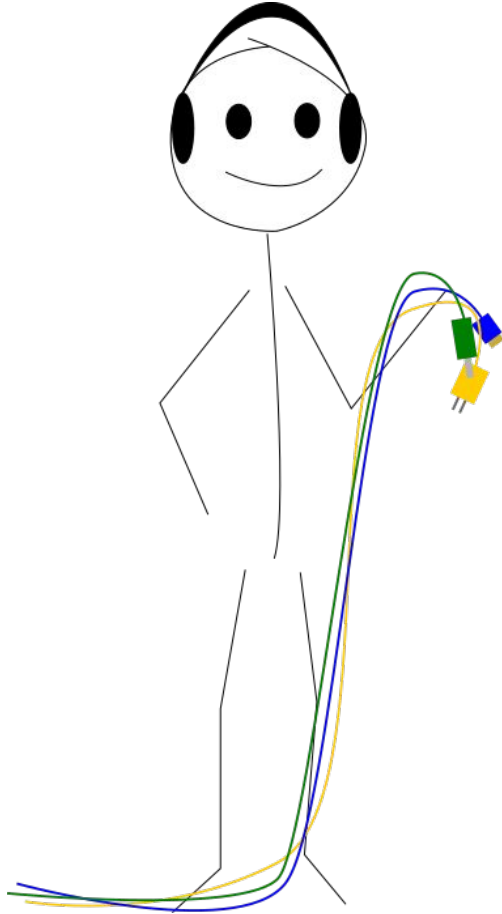
developer tools

team norms
meetings
communication

e.g. color palettes, Shiny
modules, linters, git hooks



The IT Guy

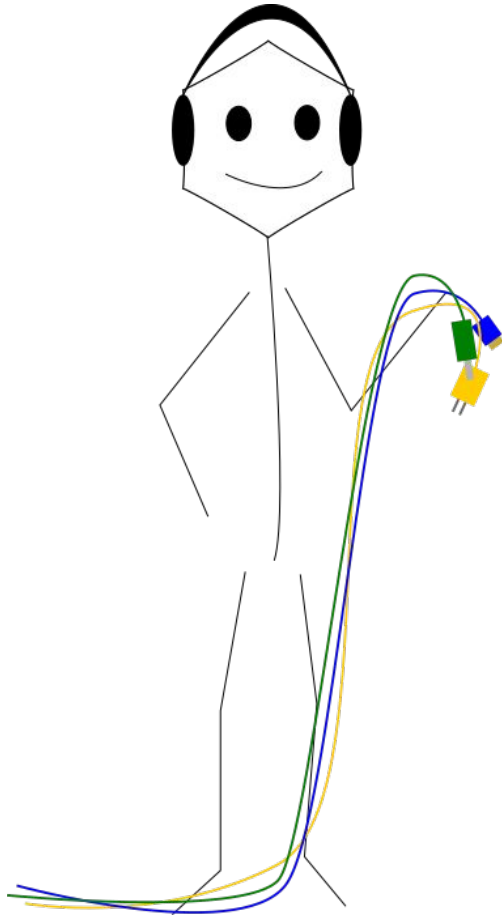


functional handle quirks of infrastructure

social promote or enforce good practices

emotional avoid frustration or stress of time lost

The IT Guy



functional handle quirks of infrastructure

social promote or enforce good practices

emotional avoid frustration or stress of time lost

-> utility functions

-> opinionated design

-> helpful error messages

```
get_database_conn <- function(username, password) {  
  
  conn <-  
    DBI::dbConnect(  
      drv = odbc::odbc(),  
      driver = {driver name},  
      server = {server},  
      UID = username,  
      PWD = password,  
      port = {port number}  
    )  
  
  return(conn)  
  
}
```

```
get_database_conn <- function(username, password) {  
  
  conn <-  
    DBI::dbConnect(  
      drv = odbc::odbc(),  
      driver = {driver name},  
      server = {server},  
      UID = Sys.getenv("DB_USER") username,  
      PWD = Sys.getenv("DB_PASS") password,  
      port = {port number}  
    )  
  
  return(conn)  
  
}
```

```
get_database_conn <- function() {  
  
  if (any(Sys.getenv(c("DB_USER", "DB_PASS")) == "")) {  
    stop(  
      "DB_USER or DB_PASS environment variables are missing.",  
      "Please read set-up vignette to configure your system."  
    )  
  }  
  
  conn <-  
    DBI::dbConnect(  
      drv = odbc::odbc(),  
      driver = {driver name},  
      server = {server},  
      UID = Sys.getenv("DB_USER"),  
      PWD = Sys.getenv("DB_PASS"),  
      port = {port number}  
    )  
  
  return(conn)  
  
}
```

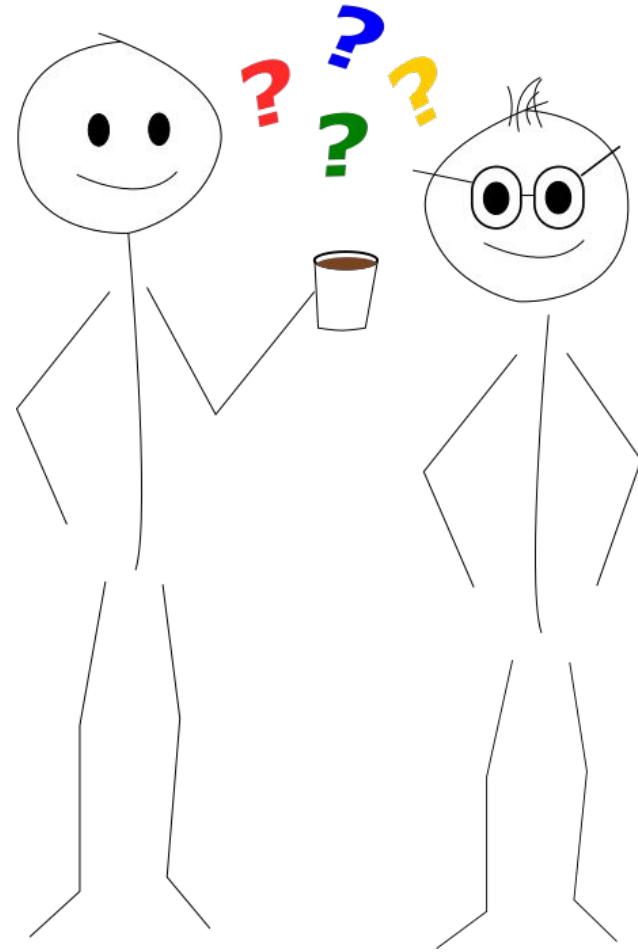
```
get_database_conn <- function() {  
  
  if (any(Sys.getenv(c("DB_USER", "DB_PASS")) == "")) {  
    stop(  
      "DB_USER or DB_PASS environment variables are missing.",  
      "Please read set-up vignette to configure your system."  
    )  
  }  
  
  conn <-  
    DBI::dbConnect(  
      drv = odbc::odbc(),  
      driver = {driver name},  
      server = {server},  
      UID = Sys.getenv("DB_USER"),  
      PWD = URLencode(Sys.getenv("DB_PASS"), reserved = TRUE),  
      port = {port number}  
    )  
  
  return(conn)  
  
}
```


The Junior Analyst / Trainee

functional perform work with reasonable assumptions

social flexible to feedback, trying new things

emotional builds trust so you can focus on other things



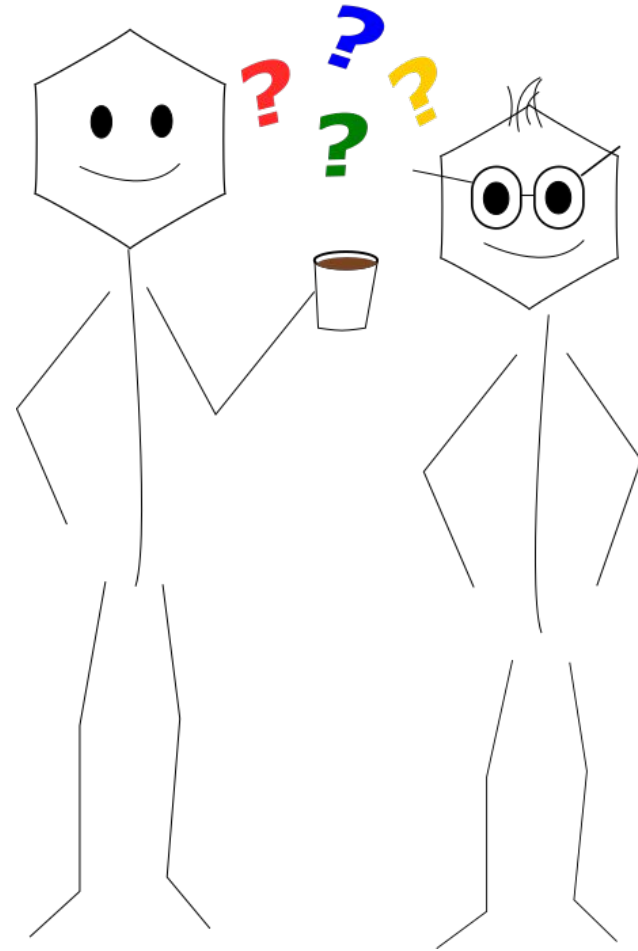
The Junior Analyst / Trainee

functional perform work with reasonable assumptions

social flexible to feedback, trying new things

emotional builds trust so you can focus on other things

- > default arguments
- > reserved keywords
- > ellipsis



```
viz_cohort <- function(data, time, metric, group) {  
  
  gg <-  
    ggplot(data) +  
    aes(x = .data[[time]],  
        y = .data[[metric]],  
        group = .data[[group]]) +  
    geom_line() +  
    my_org_theme()  
  
  return(gg)  
  
}
```

```
viz_cohort <- function(data, time, metric, group) {  
  
  gg <-  
    ggplot(data) +  
    aes(x = .data[["MONTHS_SUBSCRIBED"]],  
        y = .data[[metric]],  
        group = .data[[group]]) +  
    geom_line() +  
    my_org_theme()  
  
  return(gg)  
  
}
```

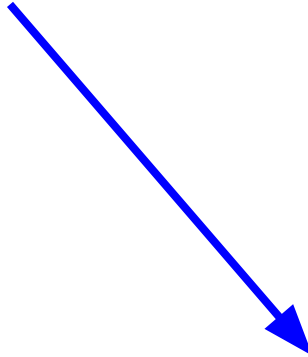
```
viz_cohort <- function(data,  
                        metric = "IND_ACTIVE",  
                        time = "MONTHS_SUBSCRIBED",  
                        group = "COHORT") {  
  
  gg <-  
    ggplot(data) +  
    aes(x = .data[[time]],  
        y = .data[[metric]],  
        group = .data[[group]]) +  
    geom_line() +  
    my_org_theme()  
  
  return(gg)  
  
}
```

```
viz_cohort <- function(data,
                        metric = "IND_ACTIVE",
                        time = "MONTHS_SUBSCRIBED",
                        group = "COHORT") {

  gg <-
    ggplot(data) +
      aes(x = .data[[time]],
          y = .data[[metric]],
          group = .data[[group]]) +
      geom_line() +
      my_org_theme()

  return(gg)

}
```



Reserved Keywords:

TIME_SUBSCRIBED
CUSTOMER_COHORT
CUSTOMER_SEGMENT
...

```
viz_cohort <- function(data,
                        time = "MONTHS_SUBSCRIBED",
                        metric = "IND_ACTIVE",
                        group = "COHORT",
                        ...) {

  gg <-
    ggplot(data) +
      aes(x = .data[[time]],
          y = .data[[metric]],
          group = .data[[group]]) +
      geom_line(aes(...)) +
      my_org_theme()

  return(gg)

}
```

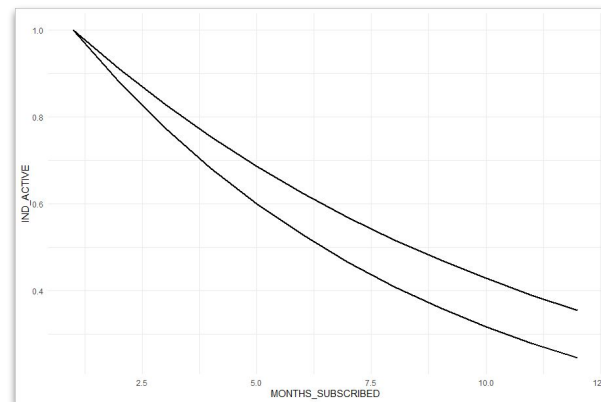
```
viz_cohort <- function(data,  
                        time = "MONTHS_SUBSCRIBED",  
                        metric = "IND_ACTIVE",  
                        group = "COHORT",  
                        ...) {
```

```
  gg <-  
    ggplot(data) +  
    aes(x = .data[[time]],  
        y = .data[[metric]],  
        group = .data[[group]]) +  
    geom_line(aes(...)) +  
    my_org_theme()
```

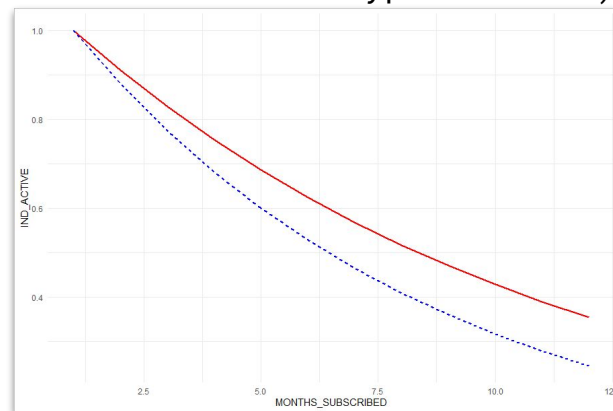
```
  return(gg)
```

```
}
```

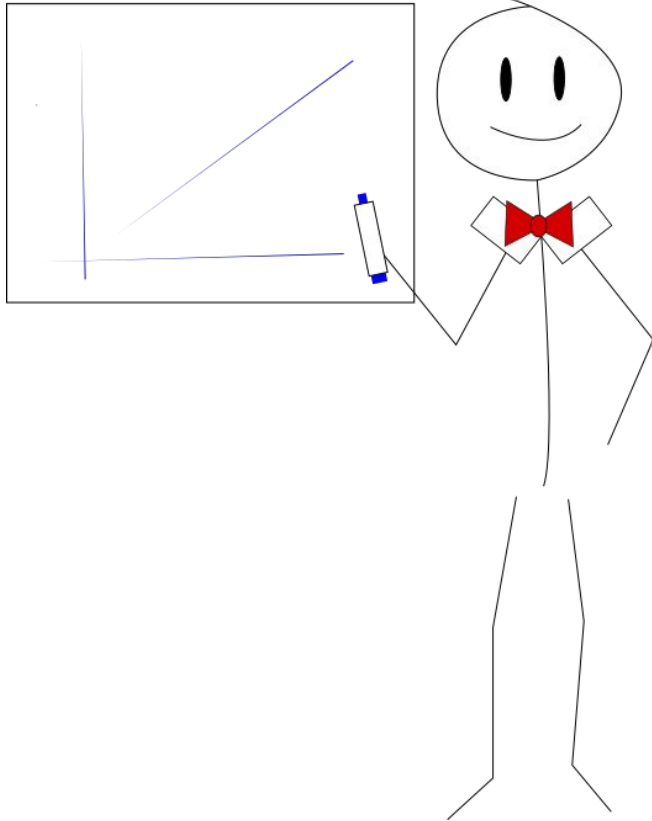
```
> viz_cohort(my_data)
```



```
> viz_cohort(my_data,  
             color = COHORT,  
             linetype = COHORT)
```



The Tech Lead / Principal Investigator

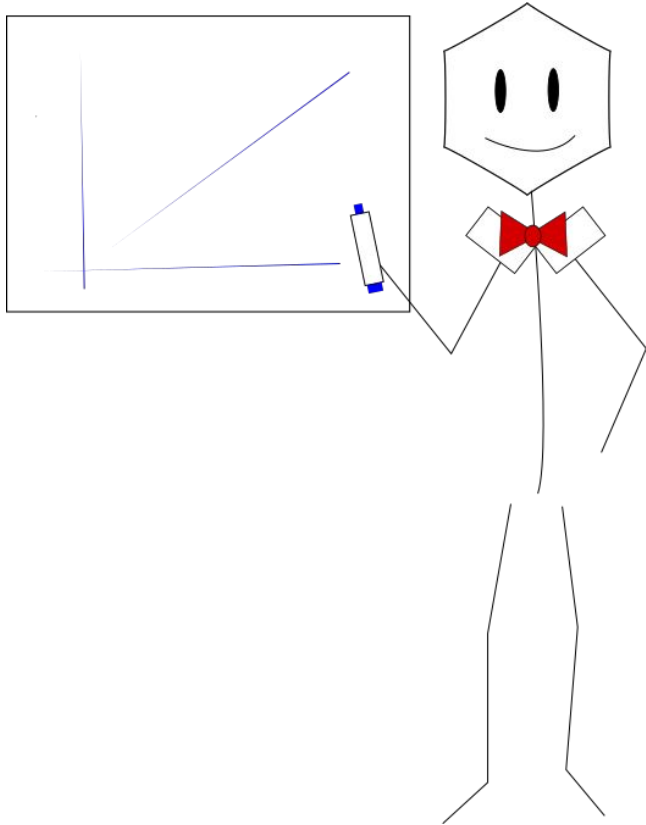


functional coach you through issues & alternatives

social share collected knowledge

emotional inspire you to do your best work

The Tech Lead / Principal Investigator



functional help navigate common issues & alternatives

social share collected knowledge

emotional connect to latent community of practice

-> vignettes

-> templates

Vignettes as a time capsule for knowledge transfer

Concordance

Terry Therneau, Elizabeth Atkinson

September 25, 2020

1 The concordance statistic

Use of the concordance statistic for Cox models was popularized by most used measure of goodness-of-fit in survival models. One advantage it is well defined not just for survival models, but also for logistic as well. In general let y_i and x_i be observed and predicted data values, in the linear predictor from a fitted model. The concordance is defined as the probability that the prediction x goes in the same direction as the observations i, j is considered concordant if the prediction and the observations, $(y_i > y_j, x_i > x_j)$ or $(y_i < y_j, x_i < x_j)$. The concordance is the For a Cox model remember that the predicted survival \hat{y} is longer so we have to flip the definitions of concordant and discordant. For use the usual definition for exposition.

One wrinkle is what to do with ties in either y or x . Such pairs (treated as incomparable), treated as discordant, or given a score T_{xy} be a count of the pairs that are concordant, discordant, and tie (y, tied on y (but not x), and tied on both. Then

$$\tau_a = \frac{C - D}{C + D + T_x + T_y + T_{xy}}$$
$$\tau_b = \frac{C - D}{\sqrt{(C + D + T_x)(C + D + T_y)}}$$
$$\gamma = \frac{C - D}{C + D}$$
$$d = \frac{C - D}{C + D + T_x}$$

- Kendall's tau-a (1) is the most conservative; essentially treats
- The Goodman-Kruskal γ statistic (3) ignores ties in either y
- Somers' d (4) treats ties in y as incomparable; pairs that are 1/2. The AUC measure commonly used in logistic regression

All three of the above range from -1 to 1. The concordance is $(d +$

1

Crash course (dplyr)

Introduction to dplyr

When working with data you must:

- Figure out what you want to do.
- Describe those tasks in the form of a computer program.
- Execute the program.

The dplyr package makes these steps fast and easy:

- By constraining your options, it helps you think about your data manipulation challenges
- It provides simple "verbs", functions that correspond to the most common data manipulation tasks, to help you translate your thoughts into code.
- It uses efficient backends, so you spend less time waiting for the computer.

This document introduces you to dplyr's basic set of tools, and shows you how to apply them to data frames. dplyr also supports databases via the dbplyr package, once you've installed, read `vignette("dbplyr")` to learn more.

Data: starwars

To explore the basic data manipulation verbs of dplyr, we'll use the dataset `starwars`. This dataset contains 87 characters and comes from the [Star Wars API](#), and is documented in `starwars`

```
dim(starwars)
#> [1] 87 14
starwars
#> # A tibble: 87 x 14
#>   name height mass hair_color skin_color eye_color birth_year sex gender
#>   <chr> <int> <dbl> <chr> <chr> <chr> <dbl> <chr> <chr>
#> 1 Luke... 172 77 blond fair blue 19 male mascu...
#> 2 C-3PO 167 75 <NA> gold yellow 112 none mascu...
#> 3 R2-D2 96 32 <NA> white, bl... red 33 none mascu...
#> 4 Dart... 282 136 none white yellow 41.9 male mascu...
#> # ... with 83 more rows, and 5 more variables: homeworld <chr>, species <chr>,
#> # films <list>, vehicles <list>, starships <list>
```

Note that `starwars` is a tibble, a modern reimagining of the data frame. It's particularly useful for large datasets because it only prints the first few rows. You can learn more about tibbles at <http://tibble.tidyverse.org>; in particular you can convert data frames to tibbles with `as_tibble()`.

Single table verbs

dplyr aims to provide a function for each basic verb of data manipulation. These verbs can be organised into three categories based on the component of the dataset that they work with.

Method Overview (survival)

Vignettes as a time capsule for knowledge transfer

Concordance

Terry Therneau, Elizabeth Atkinson

September 25, 2020

1 The concordance statistic

Use of the concordance statistic for Cox models was popularized by most used measure of goodness-of-fit in survival models. One advantage is well defined not just for survival models, but also for logistic models. In general let y_i and x_i be observed and predicted data values, in the linear predictor from a fitted model. The concordance is defined as the probability that the prediction x goes in the same direction as the observations i, j is considered concordant if the prediction and the data i.e., $(y_i > y_j, x_i > x_j)$ or $(y_i < y_j, x_i < x_j)$. The concordance is the For a Cox model remember that the predicted survival \hat{y} is longer so we have to flip the definitions of concordant and discordant. For use the usual definition for exposition.

One wrinkle is what to do with ties in either y or x . Such pairs (treated as incomparable), treated as discordant, or given a score T_{xy} be a count of the pairs that are concordant, discordant, and tied y , tied on y (but not x), and tied on both. Then

$$\tau_a = \frac{C - D}{C + D + T_x + T_y + T_{xy}}$$
$$\tau_b = \frac{C - D}{\sqrt{(C + D + T_x)(C + D + T_y)}}$$
$$\gamma = \frac{C - D}{C + D}$$
$$d = \frac{C - D}{C + D + T_x}$$

- Kendall's tau-a (1) is the most conservative; essentially treats
- The Goodman-Kruskal γ statistic (3) ignores ties in either y
- Somers' d (4) treats ties in y as incomparable; pairs that are 1/2. The AUC measure commonly used in logistic regression

All three of the above range from -1 to 1. The concordance is $(d +$

1

Introduction to dplyr

When working with data you must:

- Figure out what you want to do.
- Describe those tasks in the form of a computer program.
- Execute the program.

The dplyr package makes these steps fast and easy:

- By constraining your options. It helps you think about your data manipulation challenges
- It provides simple "verbs", functions that correspond to the most common data manipulation tasks, to help you translate your thoughts into code.
- It uses efficient backends, so you spend less time waiting for the computer.

This document introduces you to dplyr's basic set of tools, and shows you how to apply them to data frames. dplyr also supports databases via the dbplyr package. Once you've installed, read `vignette("dbplyr")` to learn more.

Data: starwars

To explore the basic data manipulation verbs of dplyr, we'll use the dataset `starwars`. This dataset contains 87 characters and comes from the [Star Wars API](#) and is documented in `starwars`.

```
dim(starwars)
#> [1] 87 14
starwars
#> # A tibble: 87 x 14
#>   name height mass hair_color skin_color eye_color birth_year sex gender
#>   <chr> <int> <dbl> <chr> <chr> <chr> <dbl> <chr> <chr>
#> 1 Luke.. 172 77 blond fair blue 19 male mascu..
#> 2 C-3PO 167 75 cmk gold yellow 112 none mascu..
#> 3 R2-D2 96 32 cmk white, bl red 33 none mascu..
#> 4 Darth.. 202 136 none white yellow 41.9 male mascu..
#> # ... with 83 more rows, and 5 more variables: homeworld <chr>, species <chr>,
#> #   films <list>, vehicles <list>, starships <list>
```

Note that `starwars` is a tibble, a modern reimagining of the data frame. It's particularly useful for large datasets because it only prints the first few rows. You can learn more about tibbles at <http://tibble.tidyverse.org>; in particular you can convert data frames to tibbles with `as_tibble()`.

Single table verbs

dplyr aims to provide a function for each basic verb of data manipulation. These verbs can be organised into three categories based on the component of the dataset that they work with:

Conceptual Overview

Workflow & Key Questions

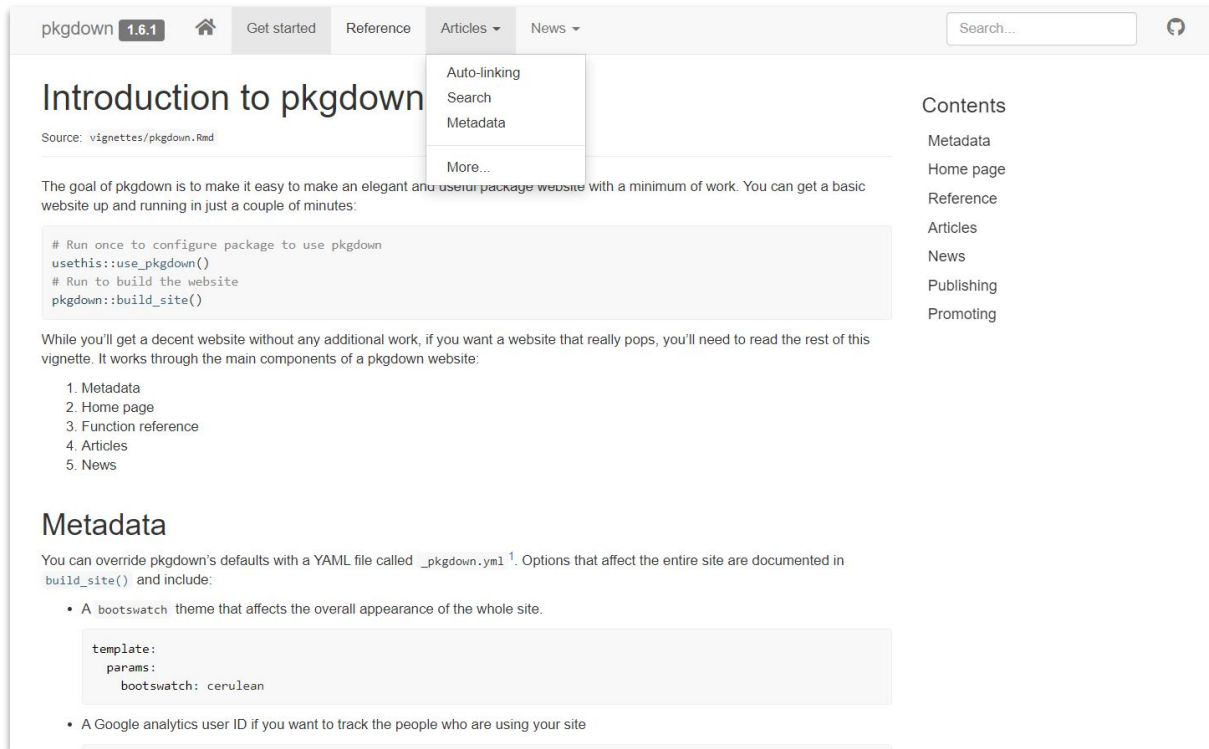
Process Documentation

Technical Overview

Methods Comparison

Expand your reach with pkgdown

```
> pkgdown::build_site()
```



The screenshot shows the pkgdown website interface. At the top, there is a navigation bar with the following items: 'pkgdown 1.6.1', a home icon, 'Get started', 'Reference', 'Articles' (with a dropdown arrow), and 'News' (with a dropdown arrow). A search bar is located on the right side of the navigation bar. The main content area is titled 'Introduction to pkgdown' and includes a source link 'vignettes/pkgdown.Rmd'. The text describes the goal of pkgdown and provides a code block for configuration and building the website. A 'Contents' sidebar is visible on the right, listing various sections of the site.

pkgdown 1.6.1 [Home](#) [Get started](#) [Reference](#) [Articles](#) [News](#)

Introduction to pkgdown

Source: [vignettes/pkgdown.Rmd](#)

The goal of pkgdown is to make it easy to make an elegant and user package website with a minimum of work. You can get a basic website up and running in just a couple of minutes:

```
# Run once to configure package to use pkgdown
usethis::use_pkgdown()
# Run to build the website
pkgdown::build_site()
```

While you'll get a decent website without any additional work, if you want a website that really pops, you'll need to read the rest of this vignette. It works through the main components of a pkgdown website:

1. Metadata
2. Home page
3. Function reference
4. Articles
5. News

Metadata

You can override pkgdown's defaults with a YAML file called `__pkgdown.yml`¹. Options that affect the entire site are documented in `build_site()` and include:

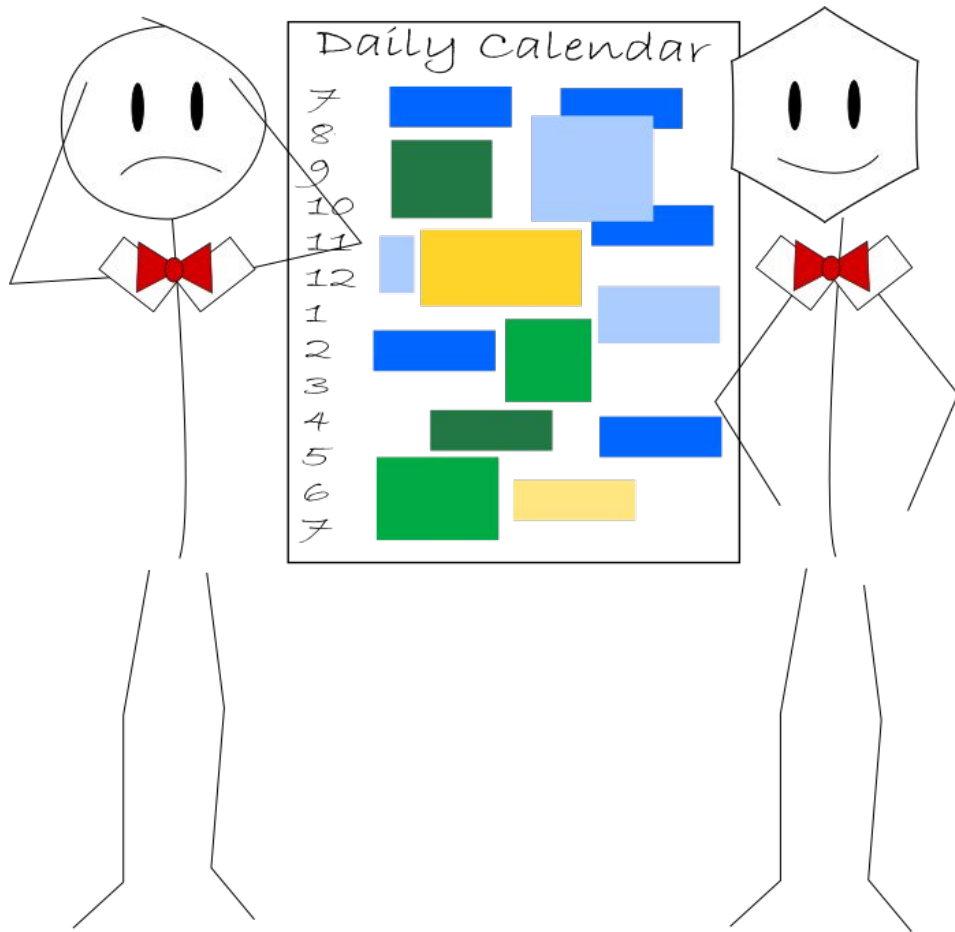
- A `bootswatch` theme that affects the overall appearance of the whole site.

```
template:
  params:
    bootswatch: cerulean
```

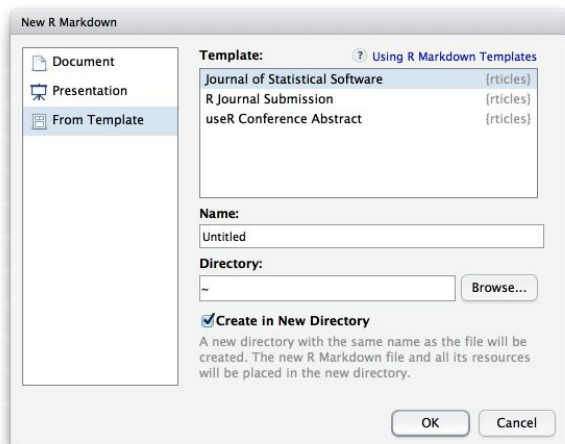
- A Google analytics user ID if you want to track the people who are using your site

Contents

- Metadata
- Home page
- Reference
- Articles
- News
- Publishing
- Promoting



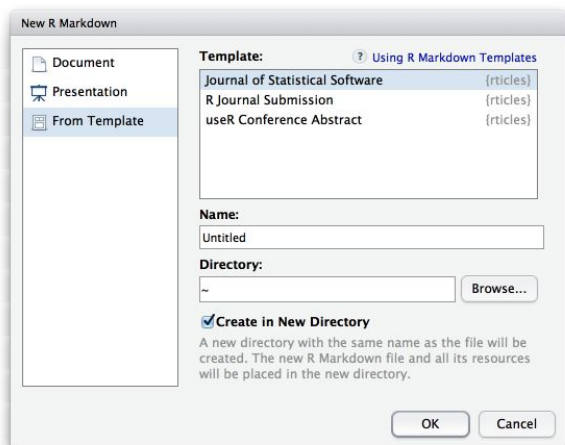
Templates as coach



Structure (flexdashboard)

```
---  
title: "Untitled"  
output:  
  flexdashboard::flex_dashboard:  
    orientation: columns  
    vertical_layout: fill  
---  
  
```${r setup, include=FALSE}  
library(flexdashboard)
```${r}  
  
Column {data-width=650}  
-----  
  
### Chart A  
  
```${r}  
```${r}  
  
Column {data-width=350}  
-----  
  
### Chart B  
  
```${r}  
```${r}
```

Templates as coach



Process walk-through

```
---  
title: "Data Validation"  
output: html_document  
---  
  
## Censored Data  
  
Run the following code to visualize how many  
observations were censored. Depending on what  
you find you will want to...  
  
```${r censored}```
```

## Analysis outline

```

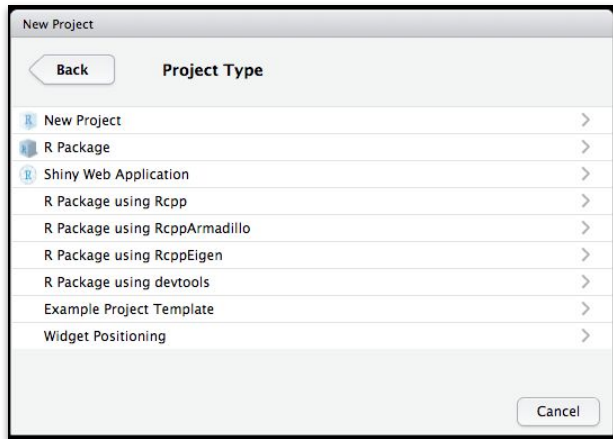
title: "Final Report"
output: html_document
params:
 month: September

Final Report

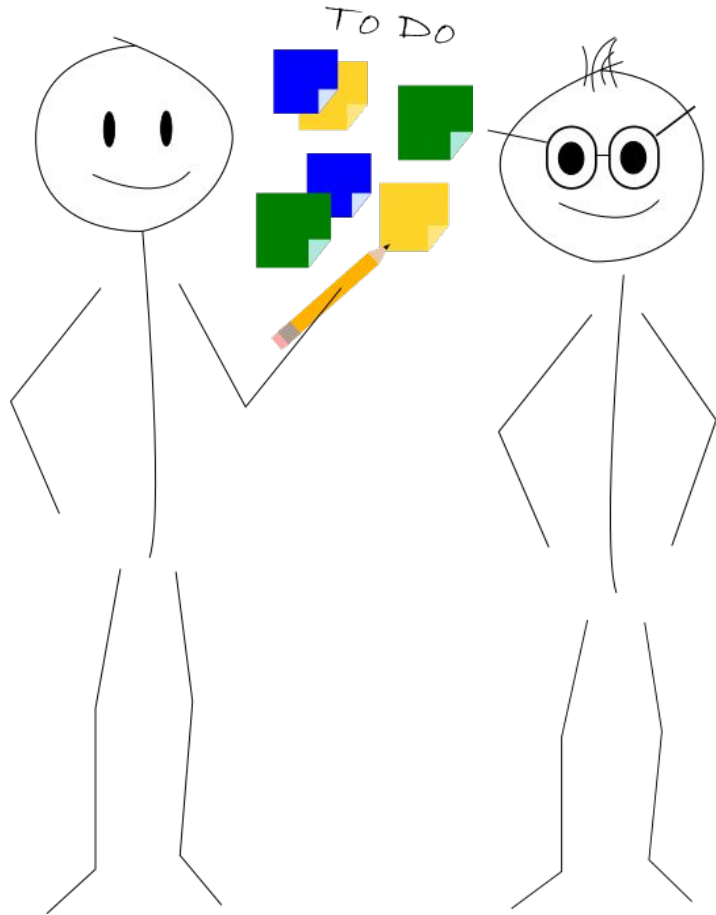
TODO: UPDATE COMMENTARY SUMMARIZING TRENDS

```${r dashboard}```
```

Templates as code reviewer



Collaboration

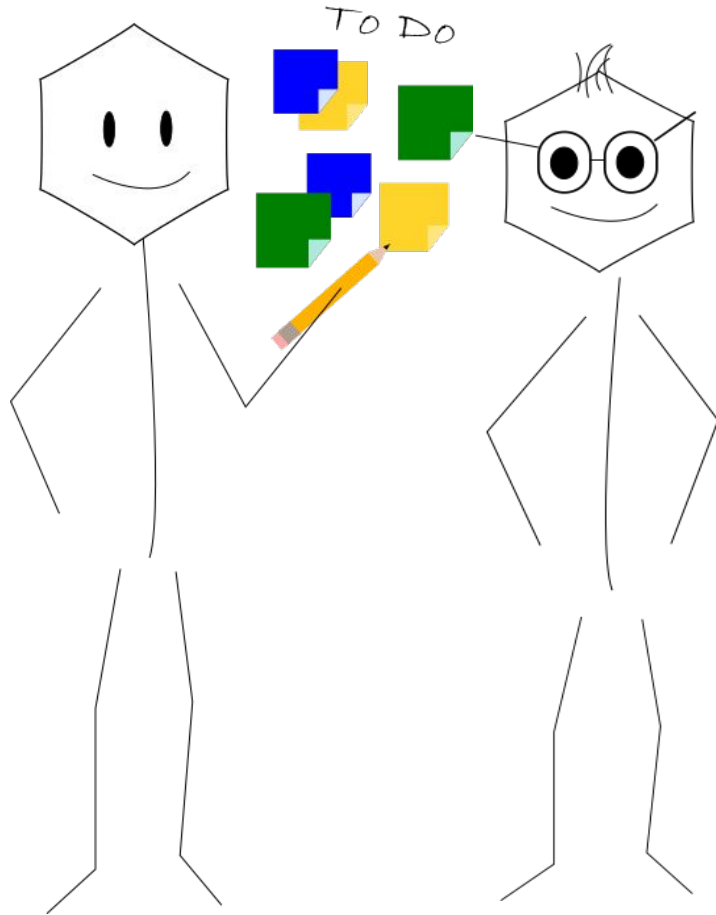


functional clear communication

social keeps promises

emotional confident yet engaged

Collaboration



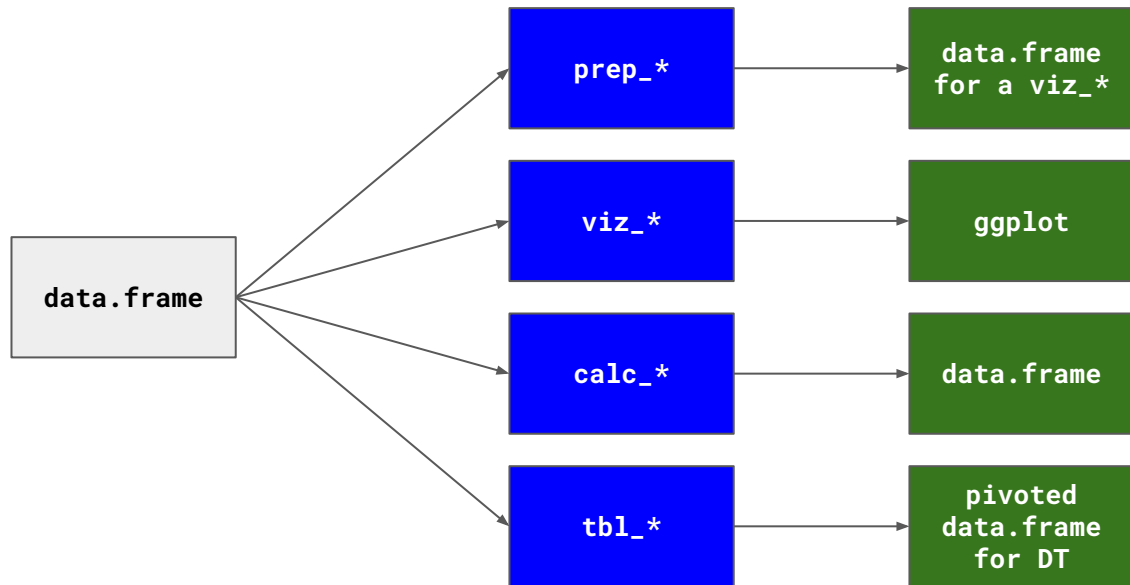
functional clear communication

social keeps promises

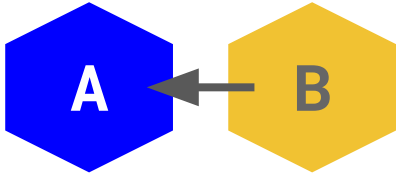
emotional confident yet engaged

- > naming
- > dependencies
- > testing

Clear communication



Dependency structures

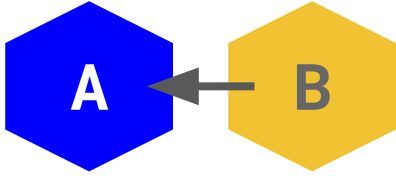


```
a_fx <- function() {...}
```

```
b_fx <- function() {  
  ...  
  a_fx()  
  ...  
}
```

Direct Dependency

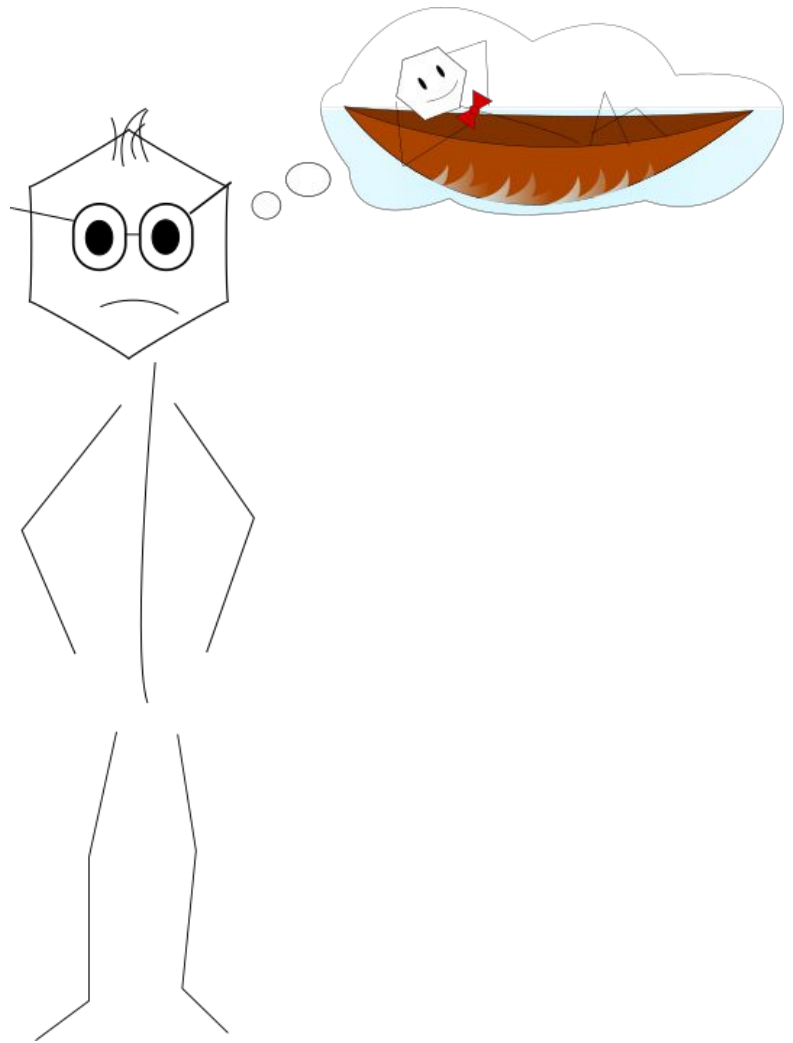
Dependency structures



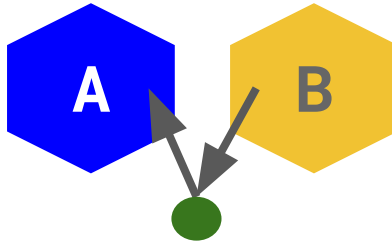
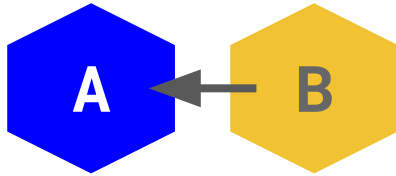
```
a_fx <- function() {...}
```

```
b_fx <- function() {  
  ...  
  a_fx()  
  ...  
}
```

Direct Dependency



Dependency structures



```
a_fx <- function() {...}
```

```
b_fx <- function() {  
  ...  
  a_fx()  
  ...  
}
```

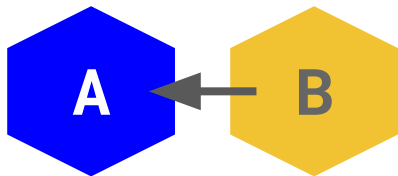
Direct Dependency

```
a_fx <- function() {...}
```

```
b_fx <- function(a_input) {  
  ...  
  do_something(a_input)  
  ...  
}
```

Clean Hand Off

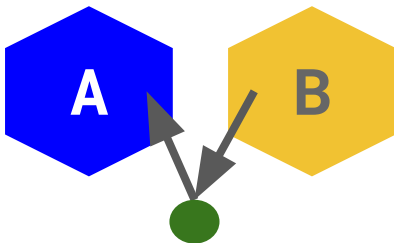
Dependency structures



```
a_fx <- function() {...}
```

```
b_fx <- function() {  
  ...  
  a_fx()  
  ...  
}
```

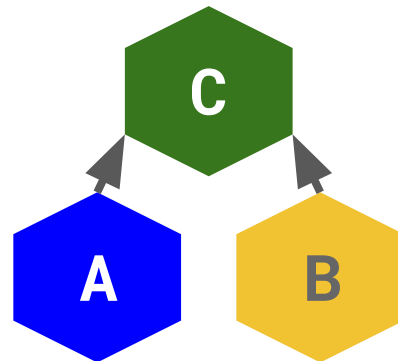
Direct Dependency



```
a_fx <- function() {...}
```

```
b_fx <- function(a_input) {  
  ...  
  do_something(a_input)  
  ...  
}
```

Clean Hand Off

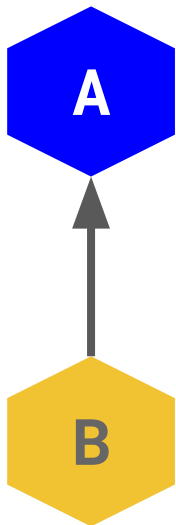


```
b_fx <- function() {  
  ...  
  c_fx()  
  ...  
}
```

```
b_fx <- function() {  
  ...  
  c_fx()  
  ...  
}
```

Common Parent

Typical unit test with dependency



b/tests/testthat/test-pkga.R

```
test_that(  
  "Receives input correctly from a",  
  {  
    expect_error(fxb(fxa(1)), NA)  
  }  
)
```

Integration tests

 `a/tests/testthat/test-pkgb.R`

```
test_that(  
  "Preps input correctly for b",  
  {  
    expect_error(fxb(fxa(1)), NA)  
  }  
)
```

 `b/tests/testthat/test-pkga.R`

```
test_that(  
  "Receives input correctly from a",  
  {  
    expect_error(fxb(fxa(1)), NA)  
  }  
)
```

Key career realizations

1. R Markdown is the gateway to more powerful tools
2. Good workflows provide an incredible amount of leverage
3. Packages are more than functions on CRAN
4. Packages can play many roles in an organization

Building tools is an increasingly important skill in data science

Lead Data Scientist - Ecolab

What You Will Do

- Actively engage with internal business teams to understand supply chain challenges and deliver robust, data driven solutions.
- Lead ongoing analytics delivery of currently deployed products, live with our field.
- Work alongside global counterparts to solve data-intensive problems using **standard analytical frameworks and tools.**
- Be encouraged and expected to innovate and be creative in your data analysis, problem solving and presentation of solutions.
- Network and collaborate with a broad range of internal business stakeholders to define and deliver joint solutions.
- Leverage cutting edge technology to creatively solve problems and disrupt existing business models.

Principal Data Scientist - Starbucks

The decision scientist principal is primarily responsible for analyzing HR metrics with the goal toward building solutions that foster a culture of inclusion. Provides thought leadership for the I&D team and partners with PRO, Analytics & Insights and Legal to understand business objectives and how HR data can enable the business to achieve commitments in support of the enterprise I&D strategy. We are looking for someone with deep level of subject matter expertise in predictive analytics and with the ability to implement I&D solutions using HR data and analytics that can facilitate business decisions. This role will be a solid line on the I&D team but sit with the Analytics & Insights teams to deliver the long-term I&D strategy working collaboratively to **build and deploy analytic solutions that can then be transitioned into new frameworks** and organizational practices.

Data Scientist - Twitch

About The Position

Data is central to Twitch's decision-making process, and analysts are essential to promoting data-driven decision-making in all of our operations. As an analyst at Twitch, you will structure our team's data to inform the way we build and refine operational processes, delivering community insights that influence the way Twitch products are built, and measuring the user sentiment of policy. You will **determine** what questions should be asked, and **scale analytics methods and tools to support our growing business**, leading the way for high-quality, high velocity decisions for your team.

Staff Data Scientist - Twitter

Qualifications

- Advanced degree in a quantitative field and 5+ years of experience (or 7+ years of experience)
- Strong track record of forming effective cross-functional partnerships
- Experience using data science to meaningfully impact product strategy and execution
- Expertise solving complex and highly impactful quantitative business problems with at least one scripting language (Python, R, etc.) and SQL
- Experience with one or more of the following in an applied setting developing statistical frameworks to understand customers and their behaviors, advanced statistical techniques for A/B testing, methods for experimental design, causal inference, or quasi-experimental analysis
- **[Bonus] Ability to create/improve reproducible analysis libraries**
- **[Bonus] Experience with PySpark or BigQuery**
- **[Bonus] Software engineering experience** in a production environment, especially for relevance/ranking systems

Inspirations

Big Ideas:

- [Good Enough Practices in Scientific Computing](#)
- [Opinionated Analysis Development](#)
- <<Anything from the R community with the word “workflow”>>

Applications:

- [Understanding the InnerSource Checklist](#)
- [rOpenSci Packages: Development, Maintenance, and Peer Review](#)
- [How R Helps AirBnb Make the Most of Its Data](#)