INSTITUTE FOR DEFENSE ANALYSES

Reproducible Research Mini-Tutorial

Heather Wojton, Project Leader

Andrew Flack
John Haman
Kevin Kirshenbaum

April 2019

IDA Document NS D-10581

Log: H-2019-000162

INSTITUTE FOR DEFENSE ANALYSES
4850 Mark Center Drive
Alexandria, Virginia 22311-1882

The Institute for Defense Analyses is a non-profit corporation that operates three federally funded research and development centers to provide objective analyses of national security issues, particularly those requiring scientific and technical expertise, and conduct related research on other national challenges.

About This Publication
This work was conducted by the Institute for Defense Analyses (IDA) under contract HQ0034-14-D-0001, Task BD-9-229990, "Test Science Applications," for the Office of the Director, Operational Test and Evaluation. The views, opinions, and findings should not be construed as representing the official position of either the Department of Defense or the sponsoring organization.

Acknowledgments
Review was conducted by Robert R. Soule, Director, and Matthew R. Avery from the Operational Evaluation Division, and William E.J. Doane from the Science and Technology Policy Institute.

For more information:
Heather Wojton, Project Leader
hwojton@ida.org • (703) 845-6811

Robert R. Soule, Director, Operational Evaluation Division
rsoule@ida.org • (703) 845-2482

INSTITUTE FOR DEFENSE ANALYSES

IDA Document NS D-10581

Reproducible Research Mini-Tutorial

Heather Wojton, Project Leader

Andrew Flack
John Haman
Kevin Kirshenbaum

Executive Summary

Analyses are "reproducible" if the same methods applied to the same data produce identical results when run again by another researcher (or you in the future). Reproducible analyses are transparent and easy for reviewers to verify, as results and figures can be traced directly to the data and methods that produced them. There are also direct benefits to the researcher. Real-world analysis workflows inevitably require changes to incorporate new or additional data, or to address feedback from collaborators, reviewers, or sponsors. These changes are easier to make when reproducible research best practices have been considered from the start.

Poor reproducibility habits result in analyses that are difficult or impossible to review, prone to compounded mistakes, and inefficient to re-run in the future. They can lead to duplication of effort or even loss of accumulated knowledge when a researcher leaves your organization. With larger and more complex datasets, along with more complex analysis techniques, reproducibility is more important than ever.

Although reproducibility is critical, it is often not prioritized due to either a lack of time or an incomplete understanding of end-to-end opportunities to improve reproducibility.

This tutorial will discuss the benefits of reproducible research and will demonstrate ways that analysts can introduce reproducible research practices during each phase of the analysis workflow: preparing for an analysis, performing the analysis, and presenting results. A motivating example will be carried throughout to demonstrate specific techniques, useful tools, and other tips and tricks where appropriate. The discussion of specific techniques and tools is non-exhaustive; we focus on things that are accessible and immediately useful for someone new to reproducible research. The methods will focus mainly on work performed using R, but the general concepts underlying reproducible research techniques can be implemented in other analysis environments, such as JMP and Excel, which are briefly discussed.

By implementing the approaches and concepts discussed during this tutorial, analysts in defense and aerospace will be equipped to produce more credible and defensible analyses of T&E data.

Reproducible Research Mini Tutorial

Andrew Flack
John Haman
Kevin Kirshenbaum

11 April 2019

What is Reproducible Research?

> "An article about computational science in a scientific publication is **not** the scholarship itself, it is merely **advertising** of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures." (Buckheit and Donoho, 1995)

John Claerbout, Stanford earth scientist

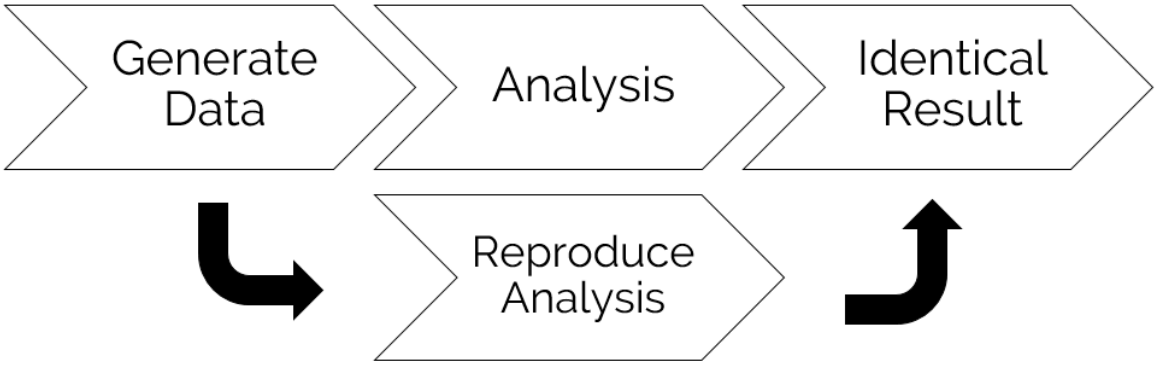
Same Data + Same Methods = Same Results

- Same Data: The original inputs for the analysis are preserved
- Same Methods: Analysis tools or scripts are saved in a way that they can be applied directly to your data
- Same Results: All of your figures, tables, and conclusions can be reproduced by another researcher (or you in the future)

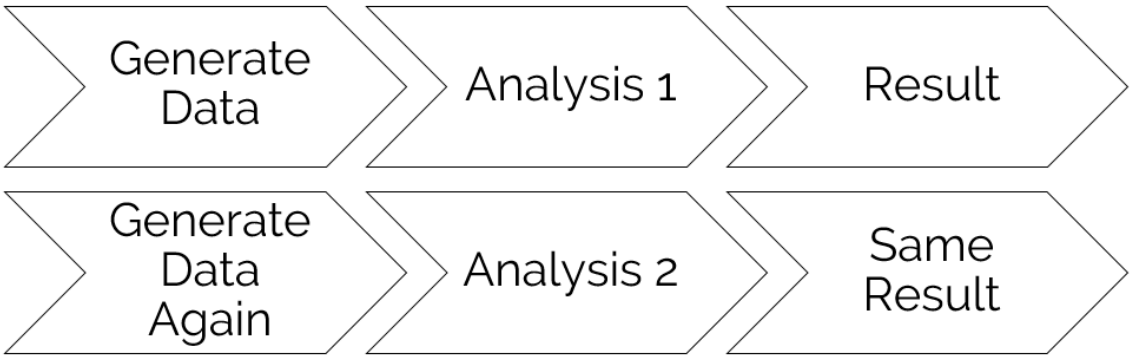
Reproducible vs. Replicable

- "Reproducible" means that if we take the same data, we get the identical result
- "Replicable" means if we did the experiment again, our result would be the same

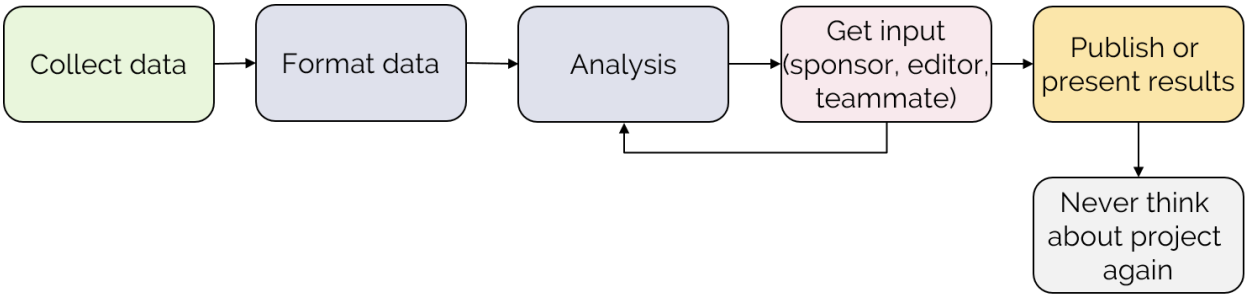
Reproducible



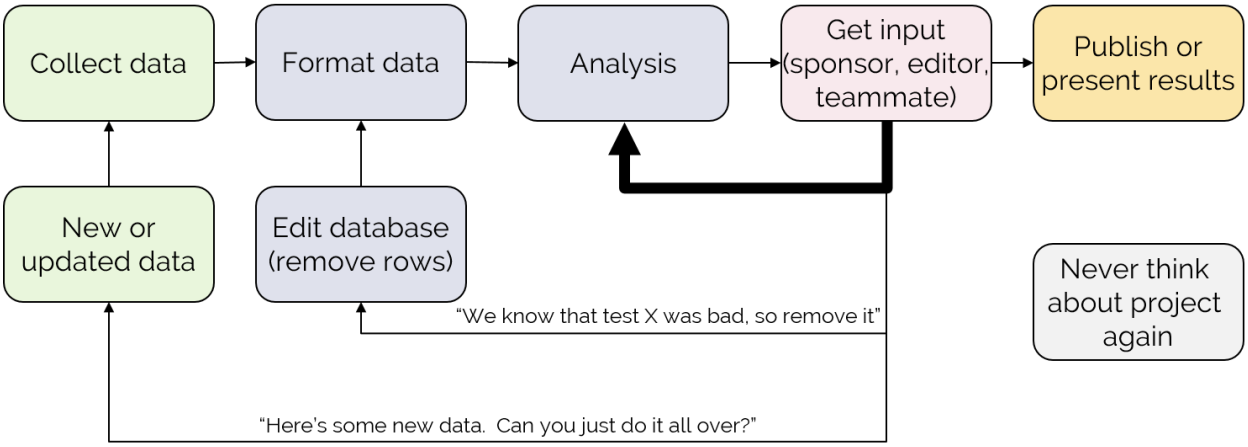
Replicable



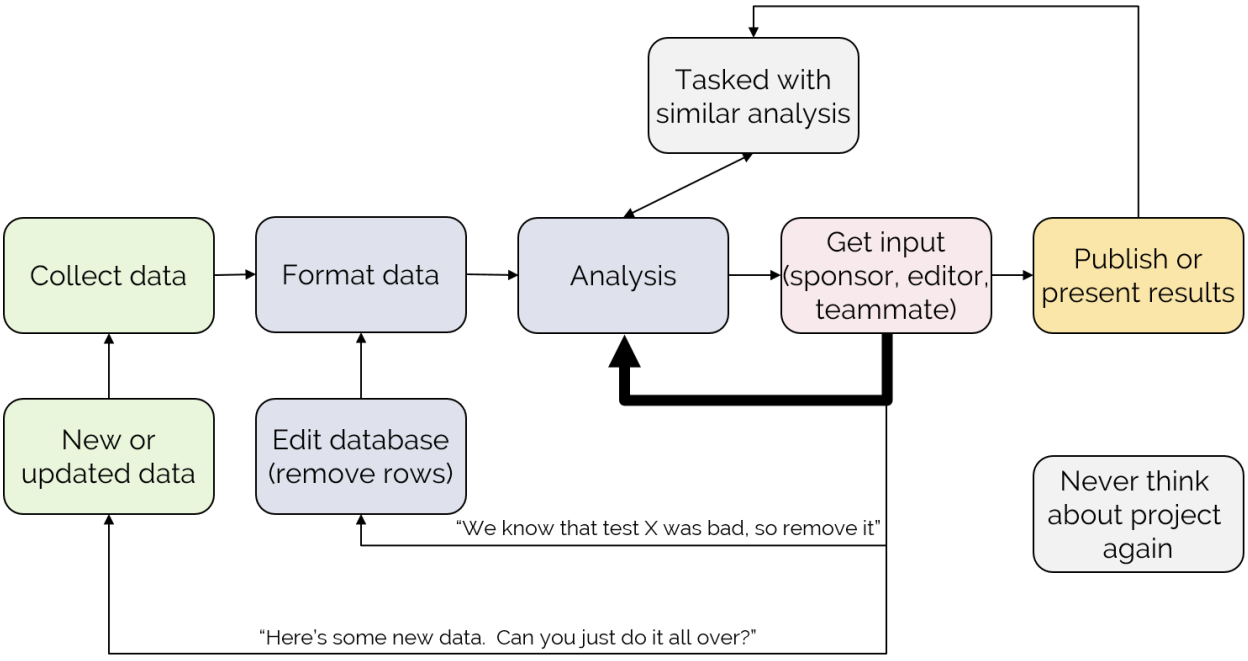
Ideal Analysis Workflow



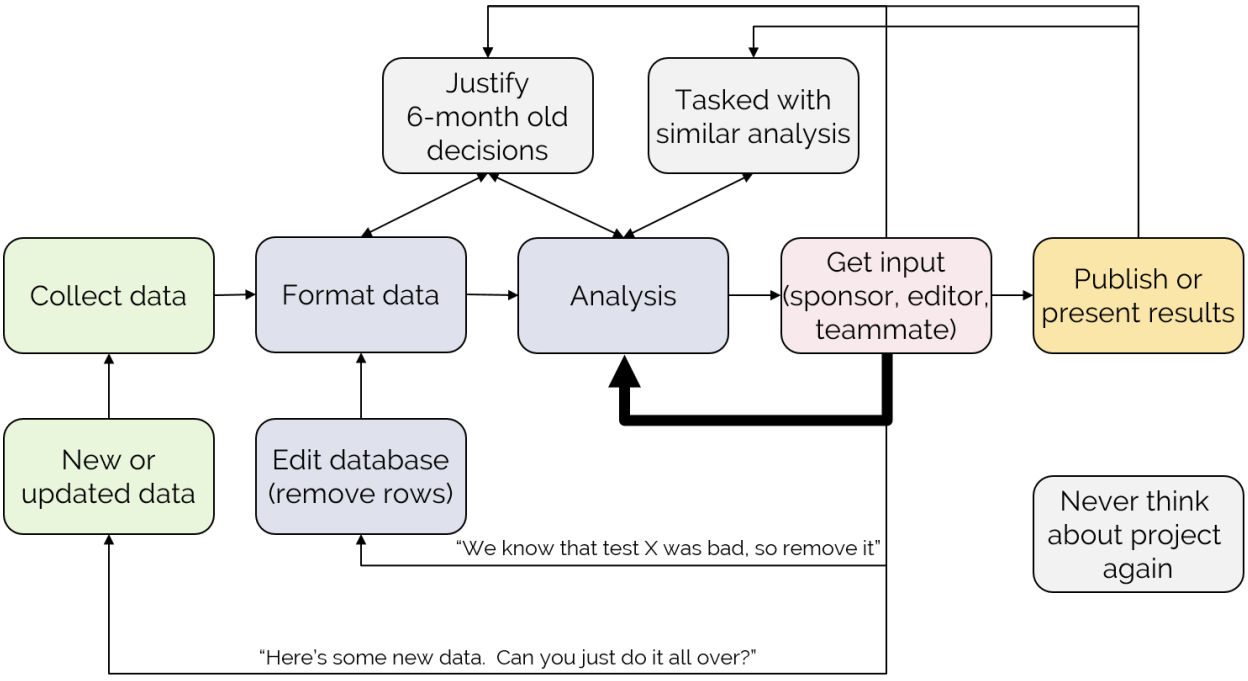
Reality



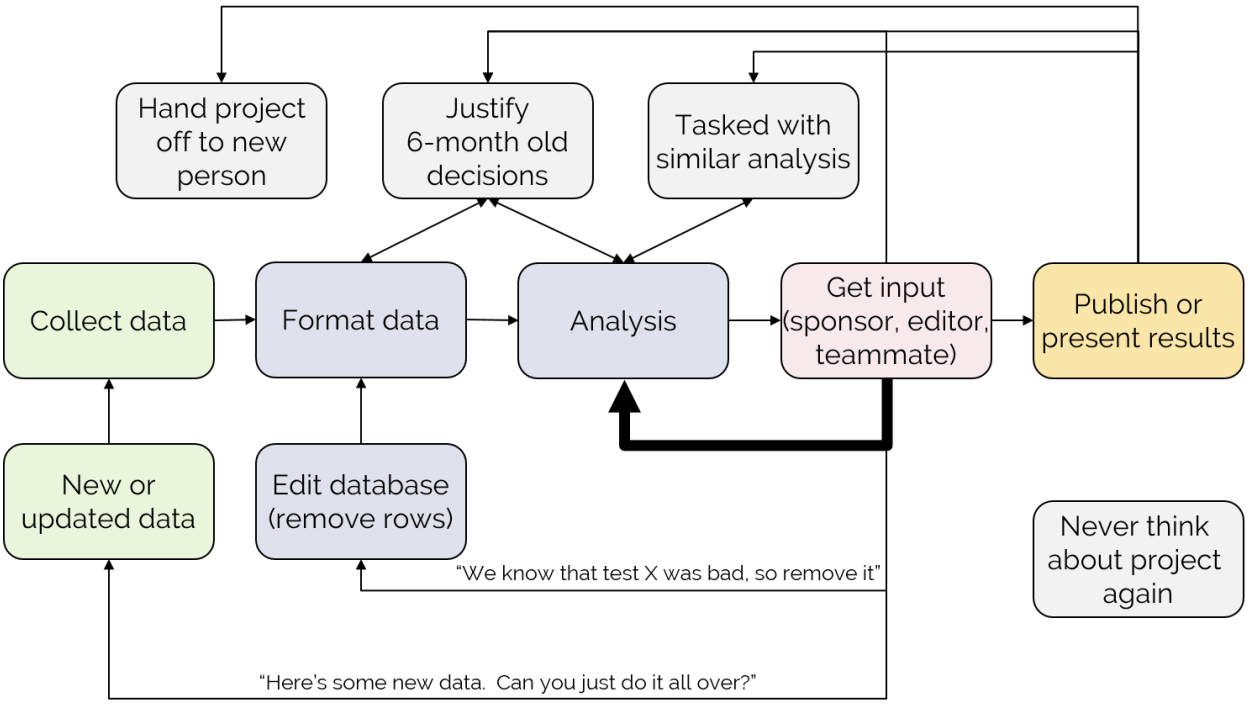
Reality



Reality



Reality



Benefits of Reproducible Research

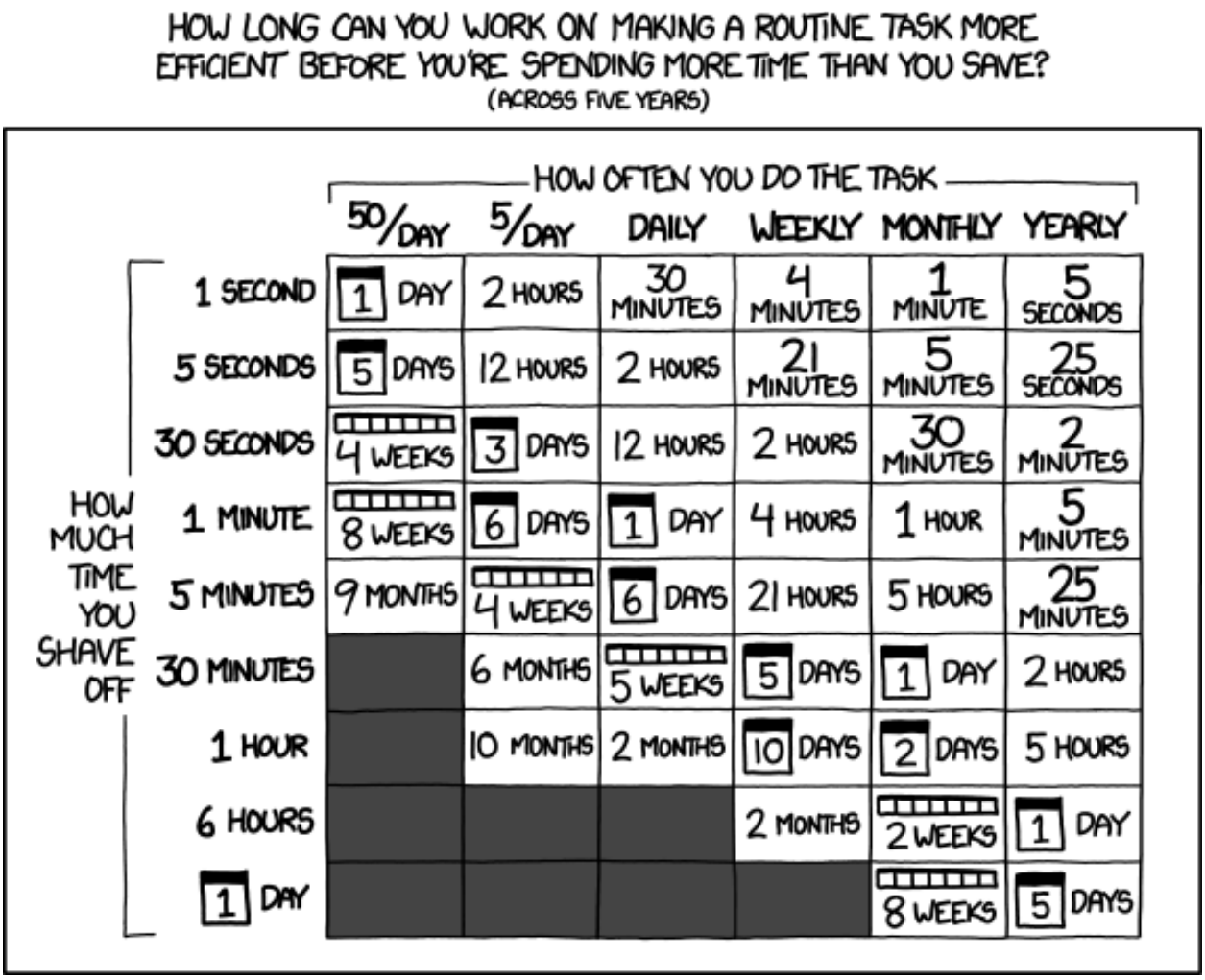
- Benefits for current you:
  - Better work habits
  - Better teamwork
  - Better work output
- Benefits for future you:
  - Changes (to code or data) are easier
  - Easier to redo analyses
  - Easier to pick up a project again
- Benefits for others:
  - Easier uptake of your hand-off project
  - Better view 'under the hood' promotes continuity, supports transparency, and increases cumulative impact

Isn't it extra work?

It can be, especially when breaking your old habits

Gets faster and easier as you get more comfortable with the tools, techniques, and workflows

It may save you time later!



Credit: xkcd

If you learn nothing else today...

Document everything!

Stay organized and write readable code

Keep raw data Read Only

Think about how to use this in your research

Incremental improvement is OK!

Motivating Example

Rigid-Hulled Inflatable Boats

The Navy wants to acquire a new rigid-hulled inflatable boat (RHIB). They have designed a test to measure the time required to launch the boats under different conditions.

(Fake) RHIB Test Data

light	length	200kg-2pass	200kg-4pass	100kg-2pass	100kg-4pass
day	7	14	17	12	16
day	13	16	18	13	15
night	7	20	25	19	23
night	13	21	25	18	22

Considerations while preparing to start your analysis

In this section:

- Get organized
  - Set up folder structure
  - Create an R Project
  - Create a version control repo
- Manage and prepare your data

Organizing files

All project-related files and scripts should be in a single overarching project directory

```
RHIB_Analysis/
|-- data_raw/
|-- data_clean/
|-- docs/
|-- figures/
|-- lib/
|-- munge/
|-- reports/
|-- src/
README
RHIB_Analysis.Rproj
run_all.R
TODO
```

R Projects – Why use them?

R projects make it straightforward to divide your work into multiple contexts, each with their own working directory, workspace, history, and source documents

.Rproj sets project-specific variables and formatting niceties

`RHIB_Analysis.Rproj`

```
Version: 1.0

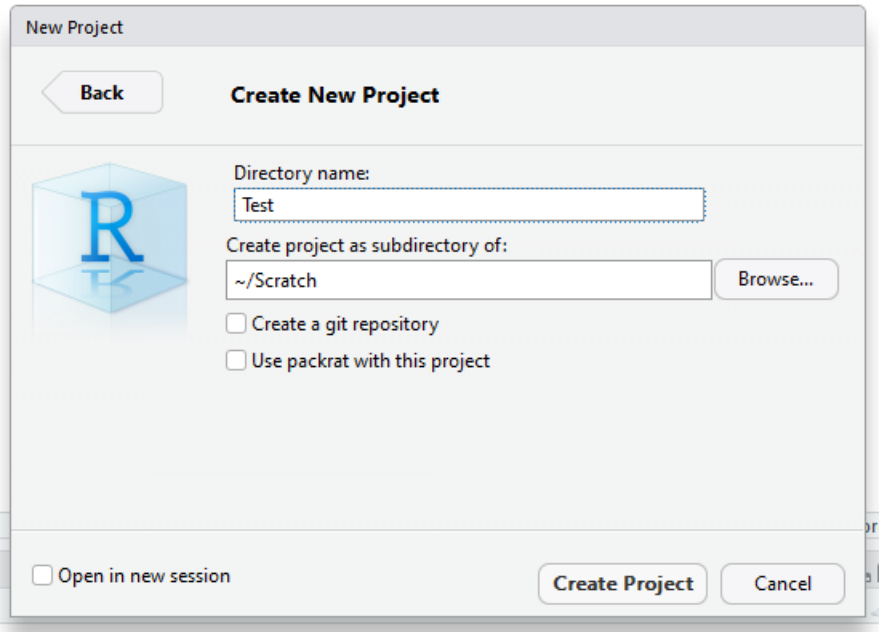
RestoreWorkspace: Default
```

```
SaveWorkspace: Default
AlwaysSaveHistory: Default

EnableCodeIndexing: Yes
UseSpacesForTab: Yes
NumSpacesForTab: 2
Encoding: UTF-8

RnwWeave: Sweave
LaTeX: pdfLaTeX
```

How do I set up an R project?



.Rproj files should be created automatically for you

Version Control – What?

Turn project (directory) into a database (repository)

Git is Ctrl-s on steroids

Version Control – Why?

- Traverse project history
- Collaborate asynchronously
- Work offline
- Decentralize storage of data and code
- Preserve sanity

Version control should be ubiquitous

- Data
- **Source Code**
- Graphs
- Manuscripts
- Presentations
- Bibliographies

Set yourself up for git success

To take full advantage of version control:

- Data – .csv
- **Source Code** – .R, .py
- Graphs – .R, .py
- Manuscripts – markdown
- Presentations – markdown
- Bibliographies – BibTeX

Plain text is light, readable, portable, and universal

Resources to get started

Git is built into Rstudio

`happygitwithr.com`

Managing your data

Raw data folder should be treated as Read Only

Consider removing 'write' permissions from raw data files

Plain text data are best for short and medium duration projects

Reshaping Data

Recall our example dataset...

light	length	200kg-2pass	200kg-4pass	100kg-2pass	100kg-4pass
day	7	14	17	12	16
day	13	16	18	13	15
night	7	20	25	19	23
night	13	21	25	18	22

This "wide" format is common – logical to record data in this way (looks a lot like the DOE)

Reshaping Data

"Wide" format is good for answering simple questions

> What was the time to launch a **13 m** boat during the **day** when loaded with **200 kg** and **4 passengers**?

18 minutes

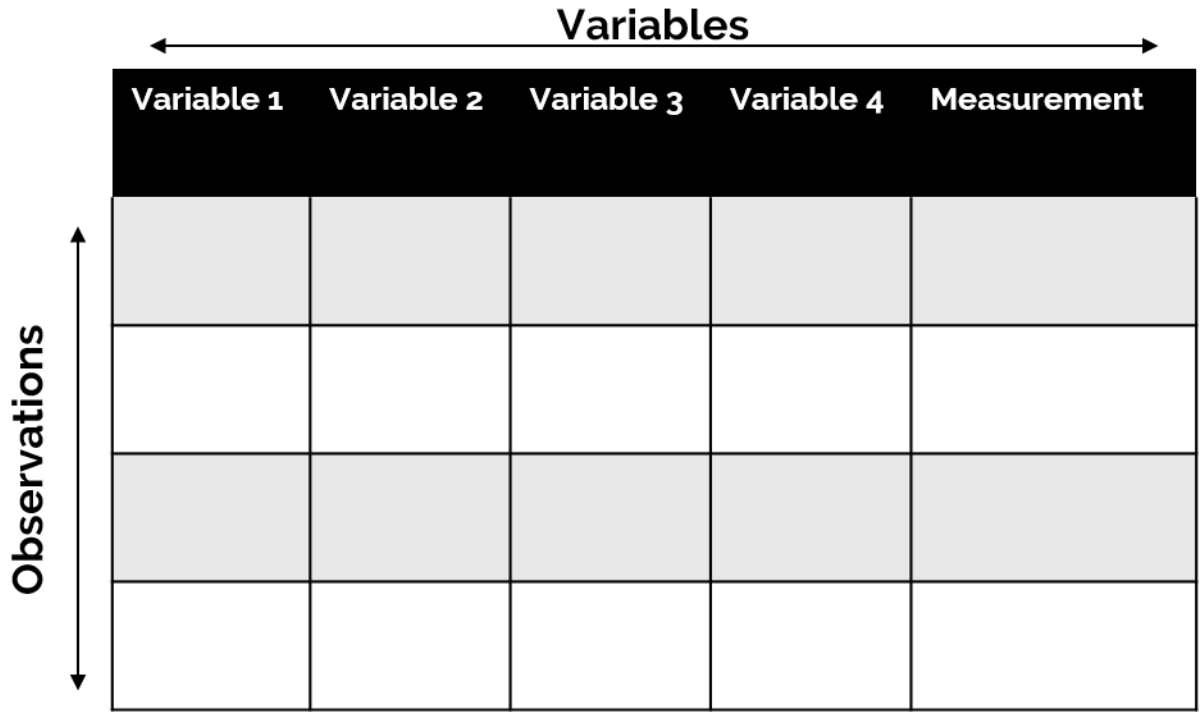
Reshaping Data

... but not good for answering more complex questions

> How does the time to launch a **13 m** boat with **4 passengers** change between **night and day**?

?

Reshaping Data – Tidy Data



Prepared Data

light	length	load	passengers	launch_time
day	7	200	2	14
day	7	200	4	17
day	7	100	2	12
day	7	100	4	16
day	13	200	2	16
day	13	200	4	18

Notes on Reshaping Data

Use code to reshape data (not manual modification)

```
RHIB %>%
  gather("loadpassengers", "launch_time", -light, -length) %>%
  separate(loadpassengers, into = c("load", "passengers"), sep = "-") %>%
  # extract numeric portion of load and passenger columns
  mutate(load = as.numeric(str_extract(load, "[:digit:]{3,}")),
         passengers = as.numeric(str_extract(passengers, "[:digit:]{1,}")))
```

```
## # A tibble: 6 x 5
##   light length  load passengers launch_time
##   <chr>  <int> <int>      <int>       <int>
## 1 day        7   200          2          14
## 2 day        7   200          4          17
## 3 day        7   100          2          12
## 4 day        7   100          4          16
## 5 day       13   200          2          16
## 6 day       13   200          4          18
```

Notes on Reshaping Data

Save data creation code as function

```
reshape_RHIB_data <- function(wide_data){

  wide_data %>%
    gather("loadpassengers", "launch_time", -light, -length) %>%
    separate(loadpassengers, into = c("load", "passengers"), sep = "-") %>%
    # extract numeric portion of load and passenger columns
    mutate(load = as.numeric(str_extract(load, "[:digit:]{3,}")),
           passengers = as.numeric(str_extract(passengers, "[:digit:]{1,}")))

}

RHIB_tidy <- reshape_RHIB_data(RHIB)
```

What if I have to change one or two points?

Reproducible workflows can include data editing

Document your steps and reasoning

```
# Per email conversation on 1 March 2019, remove the 13m 200kg 2 passenger night trial
RHIB_tidy <- RHIB_tidy[-which(RHIB_tidy$light == "night" &
                                RHIB_tidy$length == 13 &
                                RHIB_tidy$load == 200 &
                                RHIB_tidy$passengers == 2),]
```

Considerations during your analysis

In this section:

- Write functional code
- Make code human readable
- Set seeds
- Document everything

D.R.Y.

Don't repeat yourself

Turn repeated code into functions

This data needs scrubb'n

light	length	load	passengers	launch_time
night	7	100	2	19
night	7	100	4	23
night	-999	-999	2	-999
night	-999	-999	4	-999
night	13	100	2	18
night	-999	-999	4	-999

Bad

```
RHIB$launch_time[RHIB$launch_time ==  -998] <- NA
RHIB$length[RHIB$length ==  -999] <- NA
RHIB$length[RHIB$load ==  -999] <- NA
```

This code is "brittle"

Good

```
fix_one_value <- function(df, code){
    ## Replace a single miscoded value
    df[df == code] <- NA
}
```

```
fix_one_value(df = RHIB, code = -999)
```

Make your code easy to read

```
my_fun<-function(x=3,y=3*2^2){y%%x+1}
```

```
my_fun <- function(x = 3, y = 3 * 2 ^ 2) {
    ## `my_fun` calculates y mod x, then adds 1.
    y %% x + 1
}
```

Style guides are available – `style.tidyverse.org`

Nested parentheses can be difficult to follow

```
bop_on(scoop_up(hop_through(little_bunny, forest), field mice), head)
```

Adding white space and new lines help

```
bop_on(
  scoop_up(
    hop_through(little_bunny, forest),
    field mice),
  head
)
```

Too many intermediate steps leads to nondescript variable names and cluttered workspaces

```
foofoo <- little_bunny
bunnyHop <- hop_through(foofoo, forest)
bunnyHopScoop <- scoop_up(bunnyHop , field mice)
foofooFinal <- bop_on(bunnyHopScoop , head)
```

Use the pipe operator to "chain" steps together

```
foofoo <- little_bunny %>%
    hop_through(forest) %>%
    scoop_up(field_mice) %>%
    bop_on(head)
```

(Example from Hadley Wickham)

Use good file-naming practices

BAD	GOOD

BAD	GOOD
update.R	clean_data.R
John's new file with punctuation and whitespace.R	fit_model.R
figure 1.png	fig_hist_residuals.png
-TheFinal Version-.R	2019-04-02_fit_model.R

Use good file-naming practices

- Letters, numbers, periods, hyphens, and underscores *only*
- Please, no whitespace!
- Machine readable
- Human readable
- Plays well with default ordering

See "naming things" talk by Jenny Bryan

`broom`

Use the `broom` package for script-friendly model results

```
fit <- lm(launch_time ~ ., data = RHIB_tidy)
summary(fit)
```

```
##
## Call:
## lm(formula = launch_time ~ ., data = RHIB_tidy)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.8750 -0.5000 -0.1250  0.4375  1.3750
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.083333   1.132063   5.374 0.000226 ***
## lightnight  6.500000   0.405922  16.013 5.72e-09 ***
## length      0.041667   0.067654   0.616 0.550504
## load        0.022500   0.004059   5.543 0.000175 ***
## passengers  1.750000   0.202961   8.622 3.18e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8118 on 11 degrees of freedom
## Multiple R-squared:  0.9705, Adjusted R-squared:  0.9598
## F-statistic: 90.47 on 4 and 11 DF,  p-value: 2.433e-08
```

With `broom`

```
library(broom)

fit <- lm(launch_time ~ ., data = RHIB_tidy)
tidy(fit)
```

```
##          term   estimate   std.error  statistic      p.value
## 1 (Intercept) 6.08333333 1.132062567  5.3736724 2.255182e-04
## 2  lightnight 6.50000000 0.405922070 16.0129258 5.717861e-09
## 3      length 0.04166667 0.067653678  0.6158818 5.505037e-01
## 4        load 0.02250000 0.004059221  5.5429359 1.745980e-04
## 5  passengers 1.75000000 0.202961035  8.6223447 3.182041e-06
```

Output amenable to plotting and simulation studies

Use seeds to make analyses "reproducibly random"

```
x <- rnorm(1e5, mean = 0, sd = 1)
y <- 2*x + rnorm(1e5, mean = 0, sd = .01)

lm(y ~ x)$coefficients[2]
```

```
##        x
## 1.999977
```

```
x <- rnorm(1e5, mean = 0, sd = 1)
y <- 2*x + rnorm(1e5, mean = 0, sd = .01)

lm(y ~ x)$coefficients[2]
```

```
##        x
## 1.999985
```

```
set.seed(4850)

x <- rnorm(1e5, mean = 0, sd = 1)
y <- 2*x + rnorm(1e5, mean = 0, sd = .01)

lm(y ~ x)$coefficients[2]
```

```
##        x
## 2.000088
```

```
set.seed(4850)

x <- rnorm(1e5, mean = 0, sd = 1)
y <- 2*x + rnorm(1e5, mean = 0, sd = .01)

lm(y ~ x)$coefficients[2]
```

```
##        x
## 2.000088
```

Document everything

Consider comment headers on all scripts

What does the script do? Who wrote it? Last modified? etc.

```
################################################################################
# This script generates RHIB launch time estimates with confidence intervals.
#
# Author: John Doe
# Created: 22 Feb 2019
# Modified: 1 April 2019
################################################################################
```

Considerations after your analysis is complete

In this section:

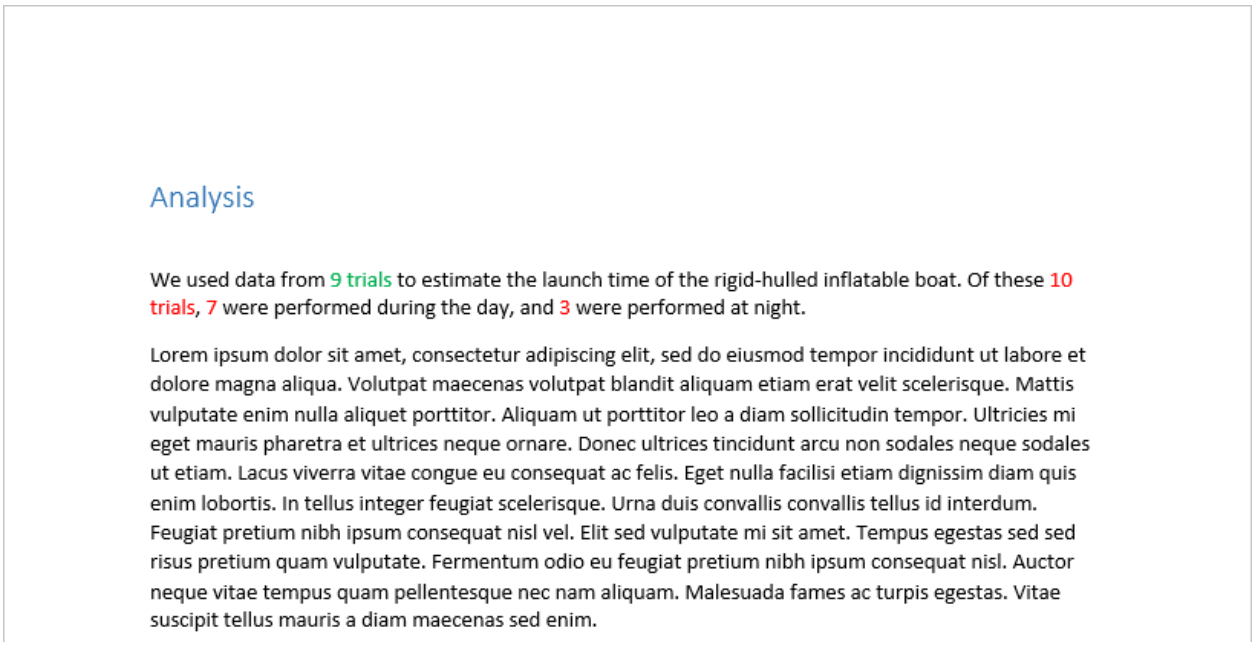
- Report your results
- Share your analyses

Link your analysis with the presentation of your results

Why?

- See the code that generated each figure or produced a value
- No copy/paste transcription errors
- Always up-to-date

You are told that there was a problem with one trial, and it should be removed from your analysis. You update your analysis, and also update the numbers in your report.



But you don't catch all references to the number of trials...

- Compile a single R Markdown document to a report in different formats, such as PDF, HTML, or Word
- Make slides for presentations (HTML5, LaTeX Beamer, or PowerPoint)
- More: dashboards, interactive applications, books, ...

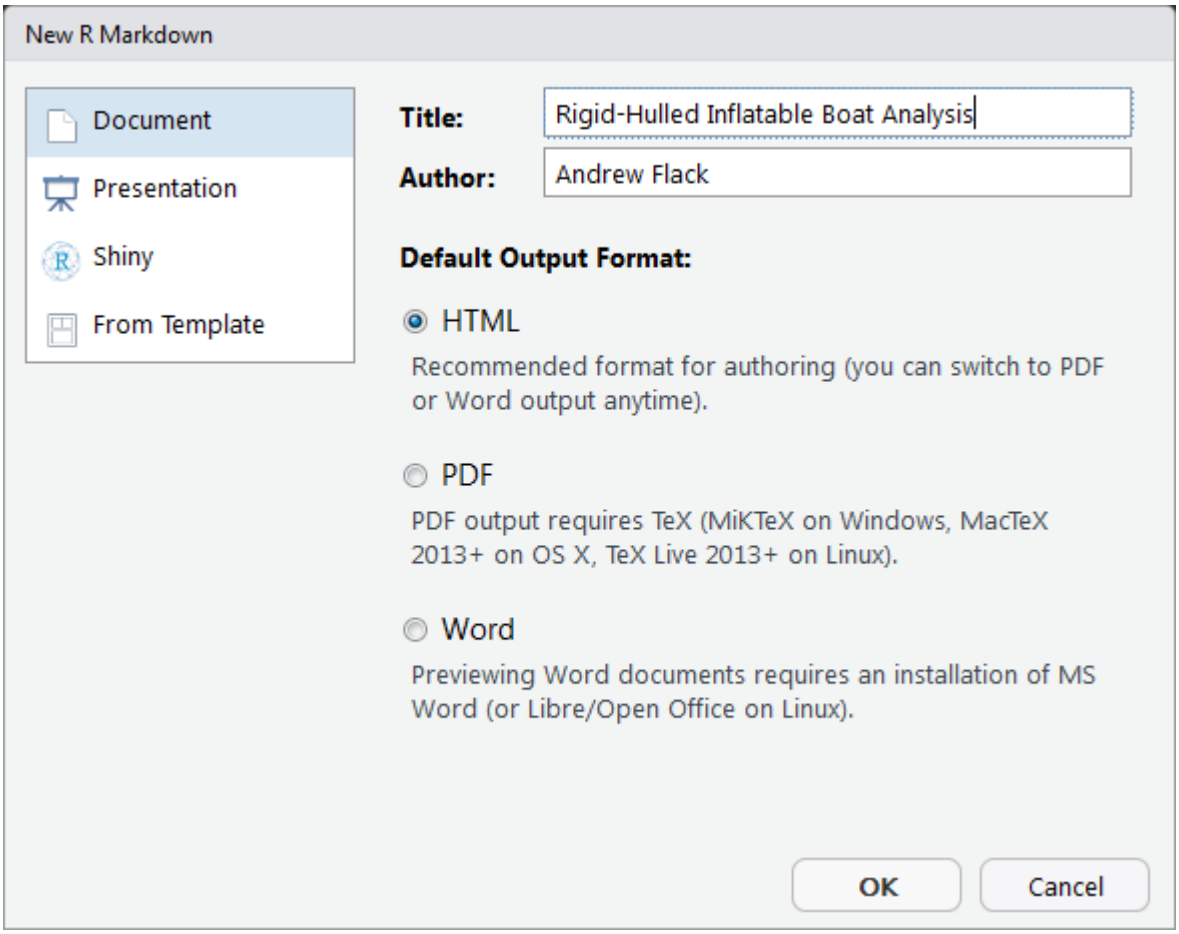
R Markdown relies on `knitr` and Pandoc

- `knitr`
  - Executes code embedded in the document
  - Converts R Markdown to Markdown
- Pandoc
  - Renders Markdown to desired output format (PDF, HTML, Word, etc.)

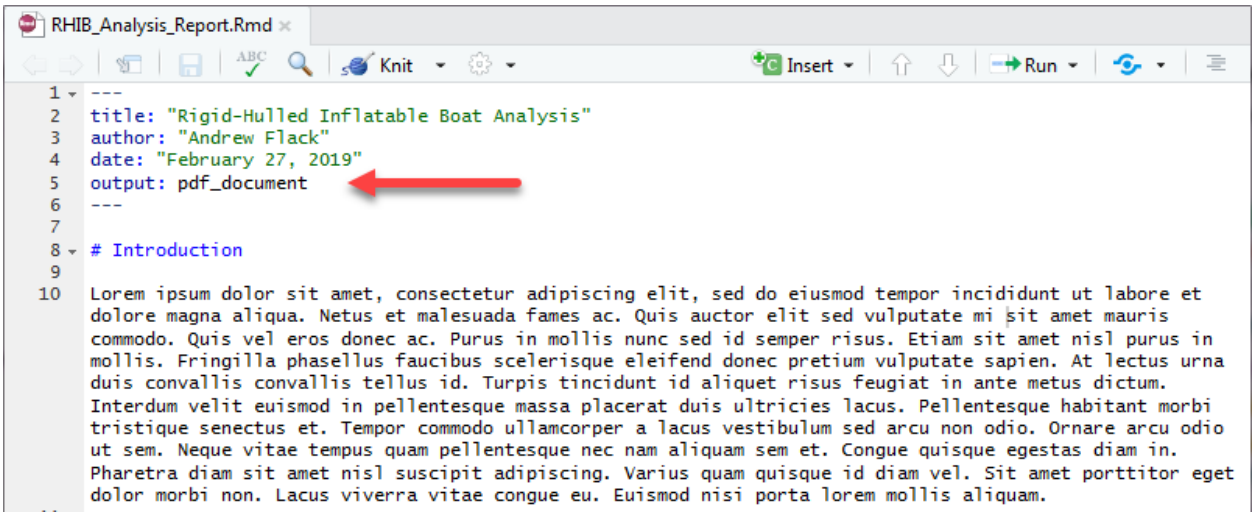
Basic Steps

1. Create new .Rmd file
2. Specify output format
3. Write narrative and incorporate code
4. Knit

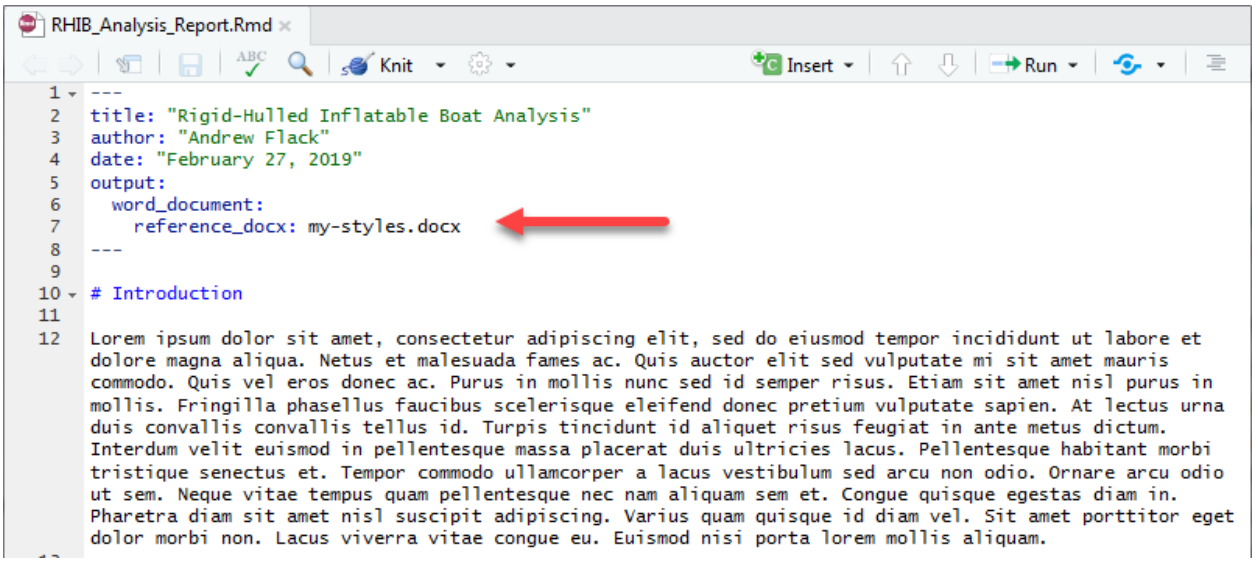
1. Create new .Rmd file



2. Specify output format in header



Use your organization's template by saving a reference document in your project directory



3. Write narrative and incorporate code

Code chunk:

```
```{r}
head(RHIB_tidy)
```
```

```
## # A tibble: 6 x 5
##   light length  load passengers launch_time
##   <chr>  <int> <int>      <int>       <int>
## 1 day        7   200          2          14
## 2 day        7   200          4          17
## 3 day        7   100          2          12
## 4 day        7   100          4          16
## 5 day       13   200          2          16
## 6 day       13   200          4          18
```

Inline code:

```
The average launch time is `r mean(RHIB_tidy$launch_time)` minutes.
```

"The average launch time is 18.375 minutes."

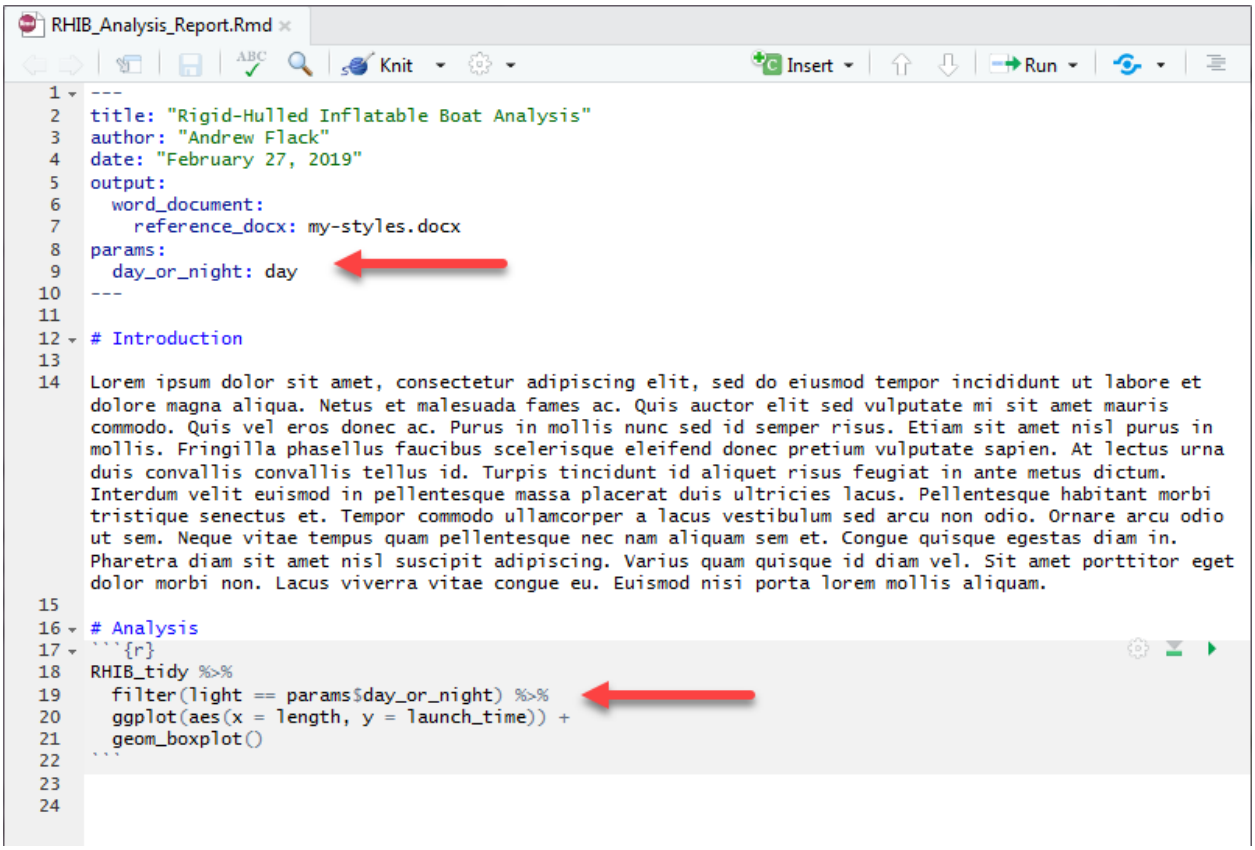
Modular or complex code can be referenced rather than incorporating it directly into your document

In your script: "01_read_and_tidy.R"

```r
## @knitr tidy_RHIB_data
RHIB_tidy <- RHIB %>%
  gather("loadpassengers", "launch_time", -light, -length) %>%
  separate(loadpassengers, into = c("load", "passengers"), sep = "-") %>%
  # extract numeric portion of load and passenger columns
  mutate(load = as.numeric(str_extract(load, "[:digit:]{3,}")),
         passengers = as.numeric(str_extract(passengers, "[:digit:]{1,}"))
```

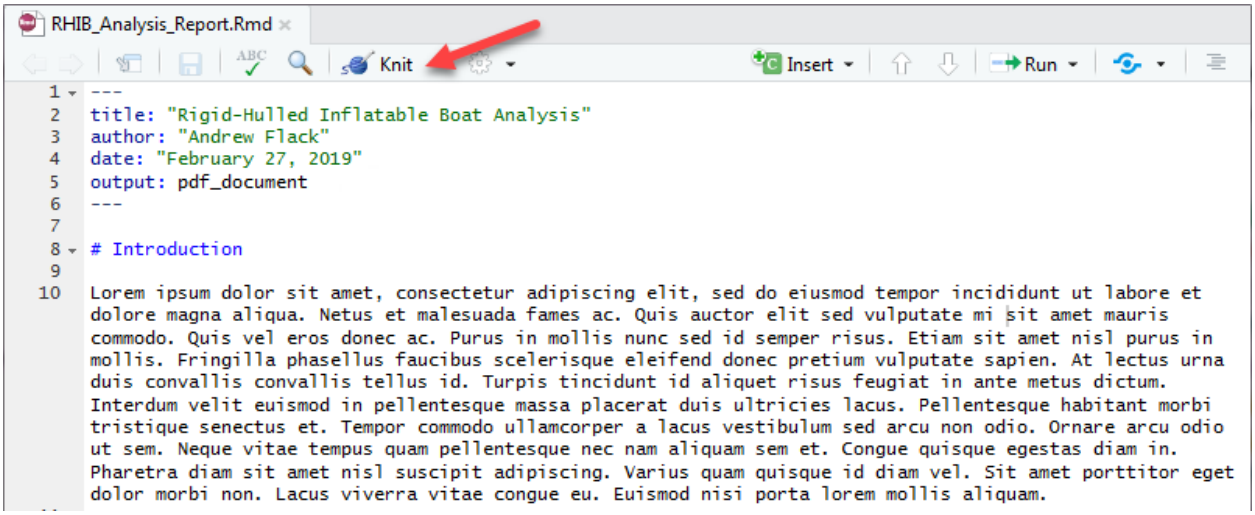
Modular or complex code can be referenced rather than incorporating it directly into your document

In your R Markdown document:

```
# Analysis
```{r, include = FALSE}
knitr::read_chunk("01_read_and_tidy.R")
<<tidy_RHIB_data>>
```
```

You can pass parameters into reports for further control



Cache code chunks to facilitate rapid development

```
# Analysis
```{r, include = FALSE, chache = TRUE}
knitr::read_chunk("01_read_and_tidy.R")
<<tidy_RHIB_data>>
```
```

Cached chunks are evaluated only when necessary

4. Knit document



Result

Rigid-Hulled Inflatable Boat Analysis

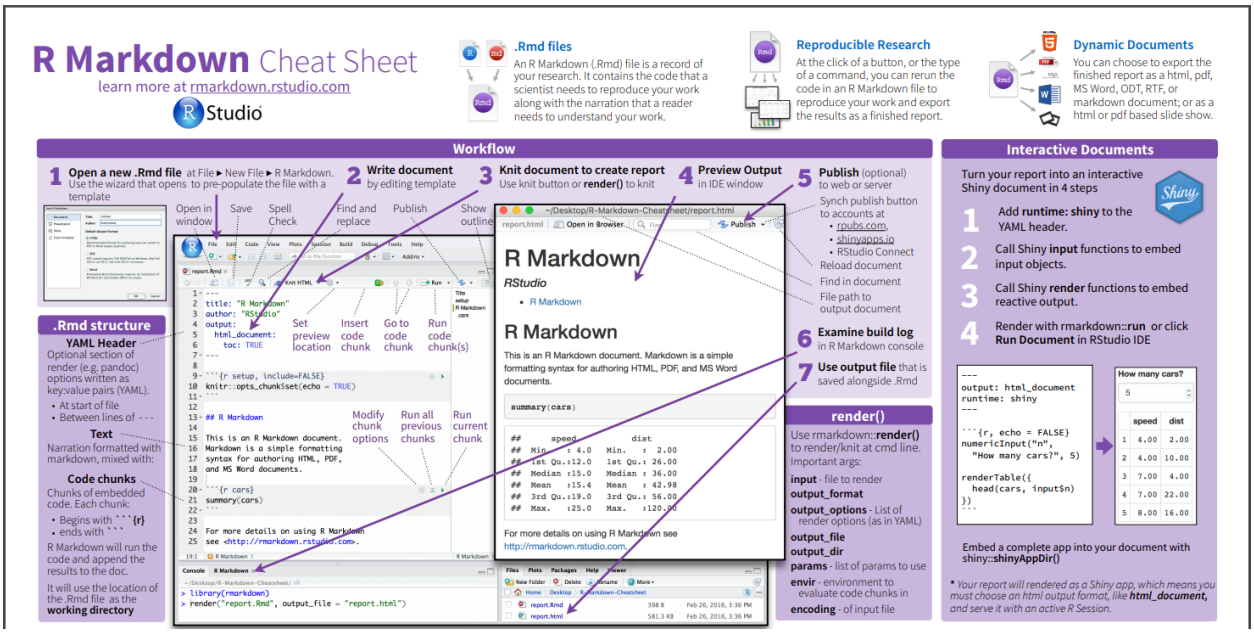
Andrew Flack

February 27, 2019

Introduction

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Netus et malesuada fames ac. Quis auctor elit sed vulputate mi sit amet mauris commodo. Quis vel eros donec ac. Purus in mollis nunc sed id semper risus. Etiam sit amet nisl purus in mollis. Fringilla phasellus faucibus scelerisque eleifend donec pretium vulputate sapien. At lectus urna duis convallis convallis tellus id. Turpis tincidunt id aliquet risus feugiat in ante metus dictum. Interdum velit euismod in pellentesque massa placerat duis ultricies lacus. Pellentesque habitant morbi tristique senectus et. Tempor commodo ullamcorper a lacus vestibulum sed arcu non odio. Ornare arcu odio ut sem. Neque vitae tempus quam pellentesque nec nam aliquam sem et. Congue quisque egestas diam in. Pharetra diam sit amet nisl suscipit adipiscing. Varius quam quisque id diam vel. Sit amet porttitor eget dolor morbi non. Lacus viverra vitae congue eu. Euismod nisi porta lorem mollis aliquam.

R Markdown Resources



rmarkdown.rstudio.com

Sharing Analyses

- Good:
  - Create README files
  - Embed helpful comments throughout analysis
- Better:
  - Check for required packages, package versions
  - Use smart, relative file paths
  - Unit test functions
  - Practice defensive programming

Mechanisms for sharing analyses

- Common:
  - Zip directory and send to collaborator
  - Collaborator clones repo
- Uncommon, but intriguing:
  - As an R package

We will focus on the common mechanisms and come back to the idea of sharing analyses as an R package at the end

To ensure a collaborator (or you) can easily run your analysis...

Operate under the following assumptions:

- Your project is self-contained in a directory
- The working directory is set to the project directory
- Scripts will be run from a brand new R session

Use RStudio Projects

RStudio Project files offer a simple solution to ensuring working directories are set appropriately.

Double-clicking on the `.Rproj` file starts a new R session and sets the working directory to the location of the `.Rproj` file.

Use relative paths in your code

Write file paths using the `here` package

`here` automatically detects the root directory of your analysis project and helps write platform-independent file paths

```
library(here)

here()
```

```
## [1] "C:/Users/aflack/Desktop/reproducible-research-mini-tutorial"
```

Write file paths using the `here` package

Especially helpful when running a script from a subdirectory (like your reports folder)

```
head(read_csv(here("example_data", "RHIB_tidy.csv")))
```

```
## # A tibble: 6 x 5
##   light length  load passengers launch_time
##   <chr>  <int> <int>      <int>       <int>
## 1 day        7   200          2          14
## 2 day        7   200          4          17
## 3 day        7   100          2          12
## 4 day        7   100          4          16
## 5 day       13   200          2          16
## 6 day       13   200          4          18
```

```
ggsave(here("figures", "my_figure.png"))
```

What's wrong with `setwd()`?

```
setwd("/Users/andrew/my_projects/2019/foo/bar/")
```

Using `setwd()` makes your code brittle

Your collaborator has a different directory structure and your script won't work

You might move the file or change your directory structure and your script won't work

What about `rm(list = ls())`?

Common first line of a script

```
# clear the workspace
rm(list = ls())
```

Deletes user-created objects from the environment, but does **not** create a new R session

Bad form to wipe a collaborator's environment!

Restart R liberally to ensure everything works in a clean environment

Do your scripts need to be run in a certain order?

Describe that order in your README

```
# README

1. Run `munge/01_read_and_tidy.R`
2. Run `munge/02_remove_bad_trial.R`
3. Open `reports/RHIB_Analysis_Report.Rmd` and click the "knit" button.
```

OR

Consider a `run_all.R` script

`run_all.R`

```
##############################################################################
# This script runs the full RHIB Analysis and generates the report.
#
# Author: John Doe
# Created: 1 March 2019
# Modified: 1 April 2019
##############################################################################

# Load required packages
library(tidyverse)

# Source custom functions
source("lib/calculate_foobar_metric.R")

# Clean and prepare data
source("munge/01_read_and_tidy.R")
source("munge/02_remove_bad_trial.R")
source("munge/03_add_new_test_data.R")

# Generate report
knit("reports/RHIB_Analysis_Report.Rmd")
```

(Courteous) Install and loading of required packages and dependencies

```
install_if_needed <- function(required_pkg){
  is_installed <- required_pkg %in% installed.packages()
  if(!is_installed){
    message("Attempting to download and install: ", required_pkg)
    install.packages(required_pkg,
                     dependencies = TRUE,
                     repos = "https://cran.revolutionanalytics.com")
  }
}

pkgs <- c("devtools", "stringr", "lubridate",    # utilities
          "rvest", "httr",                       # data acquisition
          "readxl", "readr",                     # data loading
          "dplyr", "tidyr",                      # data wrangling
          "ggplot2")                             # data visualization

invisible(lapply(pkgs, install_if_needed))
invisible(lapply(pkgs, library, character.only = TRUE))
```

(Example from Wil Doane (IDA))

Package Versioning

Installing and loading packages is a a good start, but versions might be important

- Options:
  - Document specific version requirements in README (`sessionInfo()` can be helpful here)
  - Check for proper package versions in `run_all.R` script and stop if necessary
  - Use `packrat` or other more advanced tools (not discussed)

Unit Test Functions

The `testthat` package makes it easy to test that your functions actually do what you think they do, and it is easy to integrate into your workflow

`my_function.R`

```
my_function <- function(a, b){
  sum(a, b)
}
```

Unit Test Functions

`test_my_function.R`

```
test_that('output values are correct', {
    expect_equal(my_function(1, 1), 2)
    expect_equal(my_function(0, 0), 0)
    expect_equal(my_function(-1, -1), -2)
```

```
    expect_equal(my_function(-1, 1), 0)
})

test_that('data types correct', {
    expect_is(my_function(1, 1), 'numeric')
})
```

Unit Test Functions

run_tests.R

```
library(testthat)

source("path/to/my_function.R")

test_dir("path/to/tests")
```

You can `source()` this in your `run_all.R` script

Defensive Programming

Sometimes called "assertions-based" programming

Verify assumptions about data input to analysis pipelines

This can be accomplished with base R conditional checks or `stopifnot()`, but they're not pipe-friendly

Defensive Programming

`assertr` package enables assertions-based checks within pipelines

```
reshape_RHIB_data <- function(wide_data){

  wide_data %>%
    # verify that input data has columns named "light" and "length"
    verify(has_all_names("light", "length")) %>%
    gather("loadpassengers", "launch_time", -light, -length) %>%
    # verify that input data has at least 3 digit load, at least 1 digit passengers,
    # and that they are separated by "-"
    verify(str_detect(loadpassengers, "[:digit:]{3,}kg-[:digit:]{1,}pass")) %>%
    separate(loadpassengers, into = c("load", "passengers"), sep = "-") %>%
    # extract numeric portion of load and passenger columns
    mutate(load = as.numeric(str_extract(load, "[:digit:]{3,}")),
           passengers = as.numeric(str_extract(passengers, "[:digit:]{1,}"))) %>%
    # verify that all recorded values for launch time are positive
    verify(launch_time > 0)

}
```

Bonus: Sharing Analyses as an R Package

You're already most of the way there!

Our recommended folder structure is very similar to the folder structure for an R package

```
/data
/R
/inst
/vignettes
README
DESCRIPTION
NAMESPACE
```

Instead, save data in the `/data` folder, analysis scripts and functions in `/R`, and reports in `/vignettes`

Bonus: Sharing Analyses as an R Package

- Benefits
  - `R CMD build package` will knit your report
  - A reviewer only needs to `install.packages("your_analysis_package")` and examine the vignette(s)
  - Can add a `/man` directory with rich documentation through `roxygen2`
  - Can add a `/tests` directory for unit tests

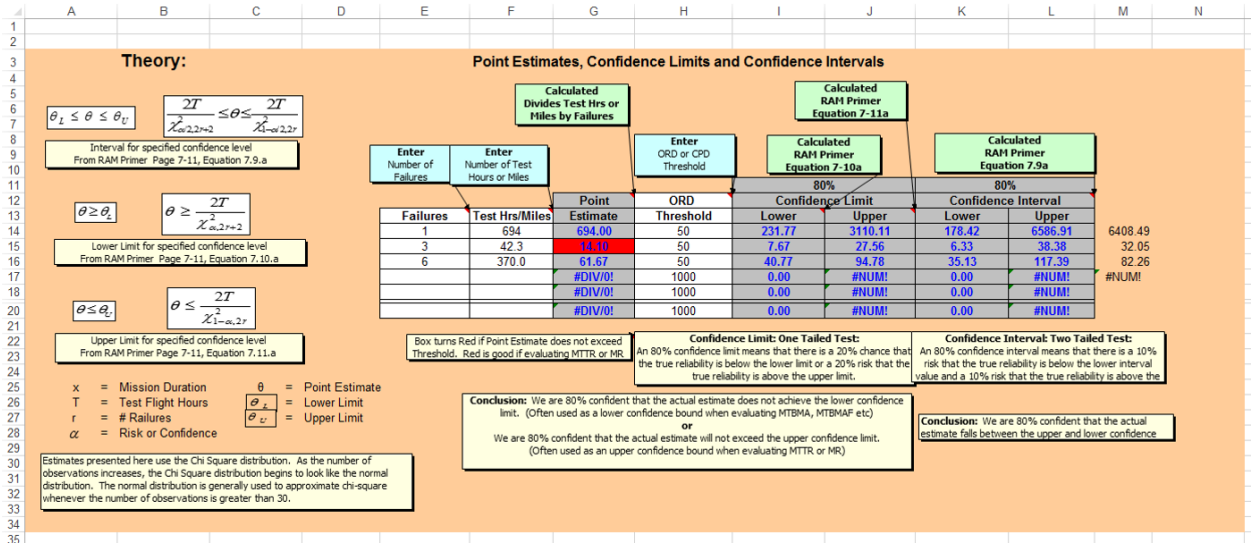
Source: "Packaging Your Reproducible Analysis", Thomas Leeper

Wrap up

Basic principles can be implemented in any analysis workflow

Whether you're using R, JMP, Excel, or any other tool, basic principles of reproducible research can still be applied

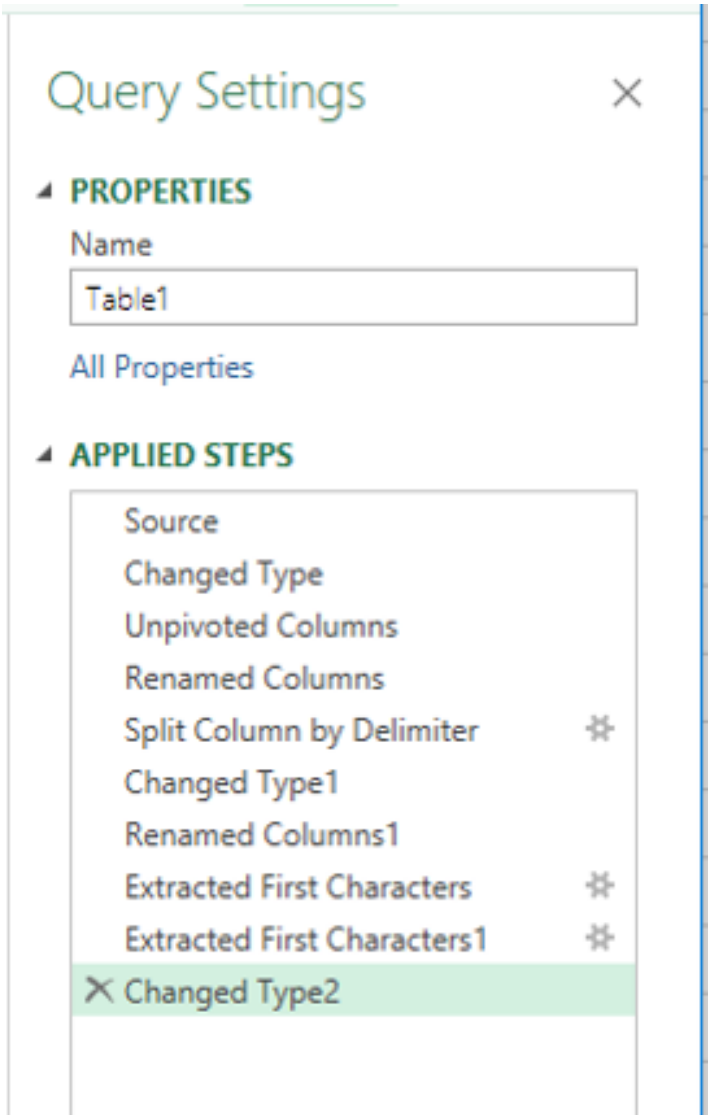
Excel - Liberal use of comments



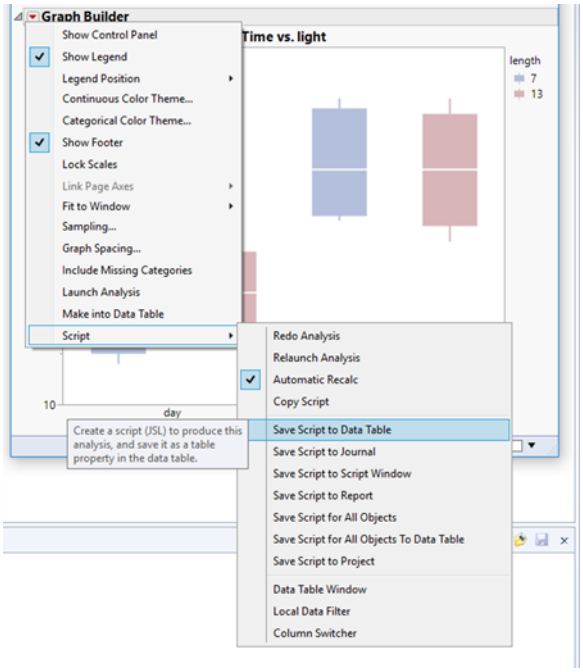
Rather than using Excel calculators interactively, make a new copy of the spreadsheet for each new analysis or use separate tabs for varying input parameters

(Example from Jon Bell (IDA))

Excel - Document data transformation steps with Power Query



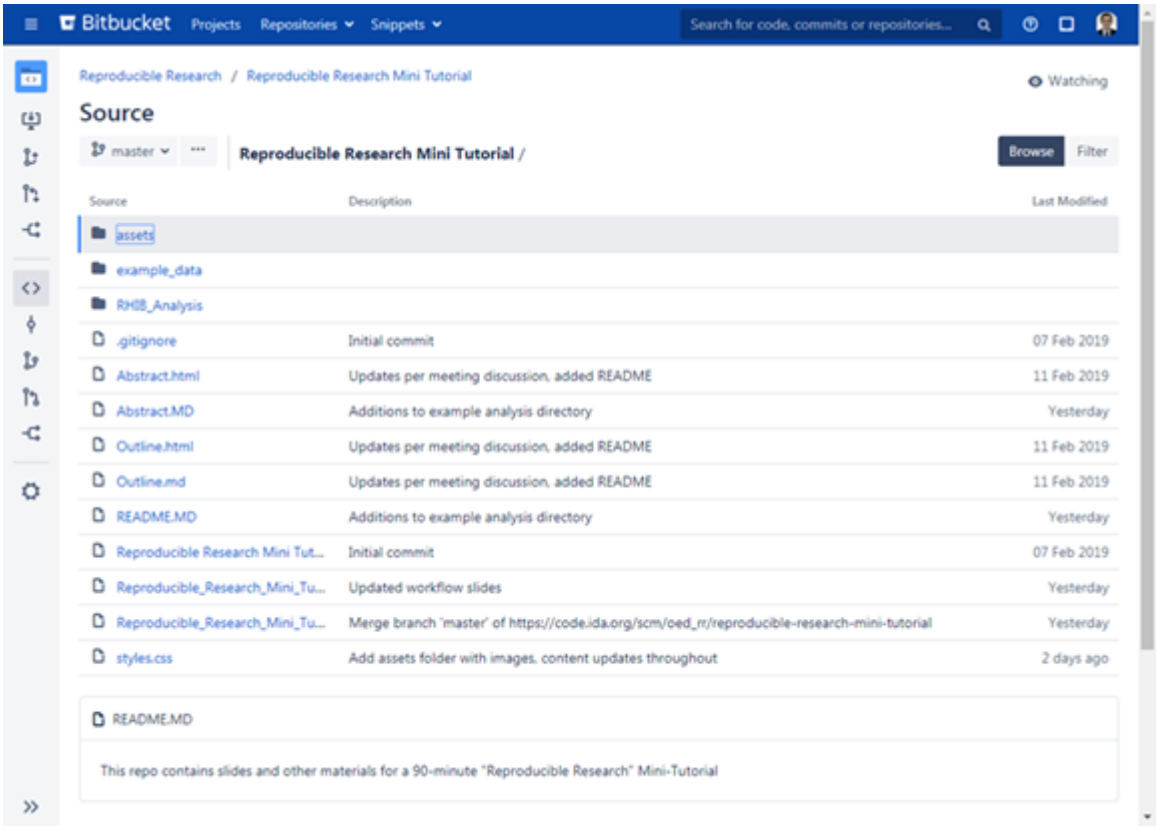
JMP - Use scripts instead of point and click analyses



Resources

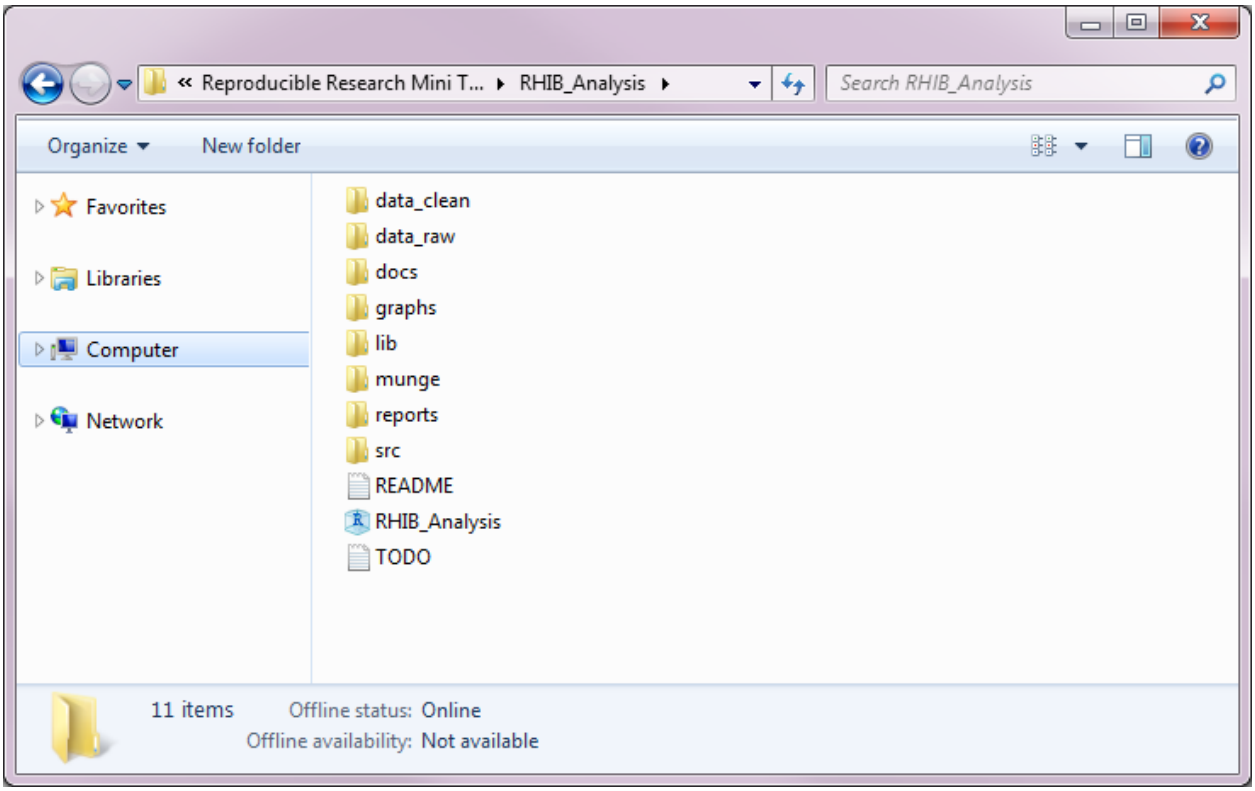
Reproduce these slides for yourself

Source code for this presentation is available for download



Resources

A full end-to-end analysis, including a dynamically-generated report, is included in the repo



References

- Jonathan B. Buckheit and David L. Donoho. Wavelab and reprodcuible research. In A. Antoniadis, editor, Wavelets and Statistics, pages 55–81. Springer, New York, 1995.
- David L Donoho. An invitation to reproducible computational research. Biostatistics, 11(3):385–388, 2010.
- Gandud, Christopher. Reproducible Research with R and R Studio, 2013.
- Xie, Yihui, Allaire, J.J., Grolemund, Garrett. R Markdown: The Definitive Guide, Chapman and Hall/CRC, 2018.

`SessionInfo()`

```
sessionInfo()
```

```
## R version 3.4.2 (2017-09-28)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 7 x64 (build 7601) Service Pack 1
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
```

```
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] here_0.1        broom_0.4.2     forcats_0.2.0   stringr_1.2.0
##  [5] dplyr_0.7.8     purrr_0.3.0     readr_1.1.1     tidyr_0.8.2
##  [9] tibble_1.4.2    ggplot2_3.0.0   tidyverse_1.2.1
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_0.2.5 reshape2_1.4.3   haven_1.1.0      lattice_0.20-35
##  [5] colorspace_1.3-2 htmltools_0.3.6  yaml_2.2.0       utf8_1.1.4
##  [9] rlang_0.3.1      pillar_1.2.3     foreign_0.8-69   glue_1.3.0
## [13] withr_2.1.2      modelr_0.1.1     readxl_1.0.0     bindrcpp_0.2.2
## [17] bindr_0.1.1      plyr_1.8.4       munsell_0.4.3    gtable_0.2.0
## [21] cellranger_1.1.0 rvest_0.3.2      psych_1.7.8      evaluate_0.10.1
## [25] knitr_1.20       parallel_3.4.2   highr_0.6        Rcpp_1.0.0
## [29] backports_1.1.1  scales_0.5.0     jsonlite_1.5     mnormt_1.5-5
## [33] hms_0.3          digest_0.6.18    stringi_1.1.5    rprojroot_1.2
## [37] grid_3.4.2       cli_1.0.0        tools_3.4.2      magrittr_1.5
## [41] lazyeval_0.2.1   crayon_1.3.4     pkgconfig_2.0.1  xml2_1.1.1
## [45] lubridate_1.7.4  assertthat_0.2.0 rmarkdown_1.11   httr_1.3.1
## [49] rstudioapi_0.7   R6_2.2.2         nlme_3.1-131     compiler_3.4.2
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
02-04-2019	IDA Draft Final	

4. TITLE AND SUBTITLE

Reproducible Research Mini-Tutorial

5a. CONTRACT NUMBER

HQ0034-14-D-0001

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Andrew C. Flack (OED); John T. Haman (OED); Kevin C. Kirshenbaum (OED)

5d. PROJECT NUMBER

BD-9-2299

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Institute for Defense Analyses
4850 Mark Center Drive
Alexandria, Virginia 22311-1882

8. PERFORMING ORGANIZATION REPORT NUMBER

D-10581

H-2019-000162

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Director, Operational Test and Evaluation
1700 Defense Pentagon
Washington, DC 20301

10. SPONSOR/MONITOR'S ACRONYM(S)

DOT&E

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release. Distribution is unlimited.

13. SUPPLEMENTARY NOTES

Project Leader:

14. ABSTRACT

Analyses are reproducible if the same methods applied to the same data produce identical results when run again by another researcher (or you in the future). Reproducible analyses are transparent and easy for reviewers to verify, as results and figures can be traced directly to the data and methods that produced them. There are also direct benefits to the researcher. Real-world analysis workflows inevitably require changes to incorporate new or additional data, or to address feedback from collaborators, reviewers, or sponsors.

15. SUBJECT TERMS

Reproducible analyses

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Heather Wojton (OED)
Unclassified	Unclassified	Unclassified	Unlimited	38	19b. TELEPHONE NUMBER *(Include area code)* (703) 845-6811