

The results in Table 1 don't seem to correspond to those in Figure 2

Mine Çetinkaya-Rundel
University of Edinburgh + Duke University + RStudio

 bit.ly/tab1-fig2-pydata

@minebocek 
mine-cetinkaya-rundel 
cetinkaya.mine@gmail.com 



The results in Table 1
don't seem to correspond to
those in Figure 2!

61

94

45

20

12

44

3

4



```
# set.seed  
set.seed(20190314)  
  
# generate 8 random numbers between 0 and 99  
runif(8, 0, 99) %>% round()
```

more than

percent

have tried and **failed** to reproduce
another scientist's experiments

more than



percent

have tried and **failed** to reproduce
their **own** experiments

Google Scholar yields



results containing the term **reproducibility crisis**
just in **2020**



earliest reference **reproducibility research***

Claerbout, Jon F., and Martin Karrenbach. "Electronic documents give reproducible research a new meaning." SEG Technical Program Expanded Abstracts 1992. Society of Exploration Geophysicists, 1992. 601-604.

that I could find...

SUMMARY

A revolution in education and technology transfer follows from the marriage of word processing and software command scripts. In this marriage an author attaches to every figure caption a pushbutton or a name tag usable to recalculate the figure from all its data, parameters, and programs. This provides a concrete definition of reproducibility in computationally oriented research. Experience at the Stanford Exploration Project shows that preparing such electronic documents is little effort beyond our customary report writing; mainly, we need to file everything in a systematic way.

In 1990 we began experimenting with electronic documents that merge our scientific software with our word-processing software. A year later we manufactured a CD-ROM containing a new textbook, Joe Dellinger's doctoral dissertation, and two progress reports of the Stanford Exploration Project. We distributed these CD-ROMs¹ to sponsors and many friends at the 1991 SEG meeting.

In 1990, we set this sequence of goals:

- Learn how to merge a publication with its underlying computational analysis.
- Teach researchers how to prepare a document in a form where they themselves can reproduce their own research results a year or more later by "pressing a single button".
- Learn how to leave finished work in a condition where coworkers can reproduce the calculation including the final illustration by pressing a button in its caption.
- Prepare a complete copy of our local software environment so that graduating students can take their work away with them to other sites, press a button, and reproduce their Stanford work.
- Merge electronic documents written by multiple authors (SEP reports).
- Export electronic documents to numerous other sites (sponsors) so they can readily reproduce a substantial portion of our Stanford research.

We met all these goals and set new ones:

- produce all new documents in this form, including lab reports in formal classes and "lab notebooks" of research progress.

¹SEP-CD-I is available from Stanford University Press, \$15 plus shipping, tel 415-723-1593

- make incremental improvements in electronic-document software
- seek partners for broadening standards (and making incremental improvements).

Our basic goal is reproducible research. The electronic document is our means to this end. In principle, reproducibility in research can be achieved without electronic documents and that is how we started. Our first nonelectronic reproducible document was a textbook in which the paper document contained the name of a program script in every figure caption. The program scripts were organized by book chapter and section so they could be correlated to an accompanying magnetic tape dump of the file system. The magnetic tape also contained all the necessary data to feed the program script.

Now that we have begun using CD-ROM publication, we can go much further. Every figure caption contains a pushbutton that jumps to the appropriate science directory (folder) and initiates a figure rebuild command and then displays the figure, possibly as a movie or interactive program. We normally display seismic images of the earth's interior, but to reach wider audiences, Figure 1 shows a satellite weather picture which the pushbutton will animate as seen on commercial television. We include all our plot software as well as freely available software from many sources, including compilers and the \LaTeX word processing system. Naturally we cannot include licensed software, but with the exception of Fortran and C compilers and the UNIX system itself, our publication includes source code for everything needed. The CD-ROM, at 680 megabytes, is so large we have had room for many executable programs on popular brands of workstations. The presence of these executables gives our readers a fast start.

Nearly everyone would rather read a paper book than the bitmapped page images on a screen that you see with an electronic document. But the illustrations in the electronic book are mostly in color, many are movies, and some are interactive. So the electronic book gives the reader a better understanding of the results. We typically use an interactive movie program to compare seismic sections where successive frames include processing with various parameters. The movie medium is much more informative than comparing seismic sections side by side. 3-D volumes are much better exhibited by movies than static paper illustrations. We are delivering a volume of software that is accessed like a book.

In 1990, we set this sequence of goals:

- Learn how to merge a publication with its underlying computational analysis.
- Teach researchers how to prepare a document in a form where they themselves can reproduce their own research results a year or more later by "pressing a single button".
- Learn how to leave finished work in a condition where coworkers can reproduce the calculation including the final illustration by pressing a button in its caption.
- Prepare a complete copy of our local software environment so that graduating students can take their work away with them to other sites, press a button, and reproduce their Stanford work.
- Merge electronic documents written by multiple authors (SEP reports).
- Export electronic documents to numerous other sites (sponsors) so they can readily reproduce a substantial portion of our Stanford research.

SUMMARY

A revolution in education and technology transfer follows from the marriage of word processing and software command scripts. In this marriage an author attaches to every figure caption a pushbutton or a name tag usable to recalculate the figure from all its data, parameters, and programs. This provides a concrete definition of reproducibility in computationally oriented research. Experience at the Stanford Exploration Project shows that preparing such electronic documents is little effort beyond our customary report writing; mainly, we need to file everything in a systematic way.

In 1990 we began experimenting with electronic documents that merge our scientific software with our word-processing software. A year later we manufactured a CD-ROM containing a new textbook, Joe Dellinger's doctoral dissertation, and two progress reports of the Stanford Exploration Project. We distributed these CD-ROMs¹ to sponsors and many friends at the 1991 SEG meeting.

In 1990, we set this sequence of goals:

- Learn how to merge a publication with its underlying computational analysis.
- Teach researchers how to prepare a document in a form where they themselves can reproduce their own research results a year or more later by "pressing a single button".
- Learn how to leave finished work in a condition where coworkers can reproduce the calculation including the final illustration by pressing a button in its caption.
- Prepare a complete copy of our local software environment so that graduating students can take their work away with them to other sites, press a button, and reproduce their Stanford work.
- Merge electronic documents written by multiple authors (SEP reports).
- Export electronic documents to numerous other sites (sponsors) so they can readily reproduce a substantial portion of our Stanford research.

We met all these goals and set new ones:

- produce all new documents in this form, including lab reports in formal classes and "lab notebooks" of research progress.

- make incremental improvements in electronic-document software
- seek partners for broadening standards (and making incremental improvements).

Our basic goal is reproducible research. The electronic document is our means to this end. In principle, reproducibility in research can be achieved without electronic documents and that is how we started. Our first nonelectronic reproducible document was a textbook in which the paper document contained the name of a program script in every figure caption. The program scripts were organized by book chapter and section so they could be correlated to an accompanying magnetic tape dump of the file system. The magnetic tape also contained all the necessary data to feed the program script.

Now that we have begun using CD-ROM publication, we can go much further. Every figure caption contains a pushbutton that jumps to the appropriate science directory (folder) and initiates a figure rebuild command and then displays the figure, possibly as a movie or interactive program. We normally display seismic images of the earth's interior, but to reach wider audiences, Figure 1 shows a satellite weather picture which the pushbutton will animate as seen on commercial television. We include all our plot software as well as freely available software from many sources, including compilers and the L^AT_EX word processing system. Naturally we cannot include licensed software, but with the exception of Fortran and C compilers and the UNIX system itself, our publication includes source code for everything needed. The CD-ROM, at 680 megabytes, is so large we have had room for many executable programs on popular brands of workstations. The presence of these executables gives our readers a fast start.

Nearly everyone would rather read a paper book than the bitmapped page images on a screen that you see with an electronic document. But the illustrations in the electronic book are mostly in color, many are movies, and some are interactive. So the electronic book gives the reader a better understanding of the results. We typically use an interactive movie program to compare seismic sections where successive frames include processing with various parameters. The movie medium is much more informative than comparing seismic sections side by side. 3-D volumes are much better exhibited by movies than static paper illustrations. We are delivering a volume of software that is accessed like a book.

¹SEP-CD-1 is available from Stanford University Press, \$15 plus shipping, tel 415-723-1593

Now that we have begun using CD-ROM publication, we can go much further. Every figure caption contains a pushbutton that jumps to the appropriate science directory (folder) and initiates a figure rebuild command and then displays the figure, possibly as a movie or interactive program. We normally display seismic images of the earth's inter-

Pioneering ‘live-code’ article allows scientists to play with each other’s results

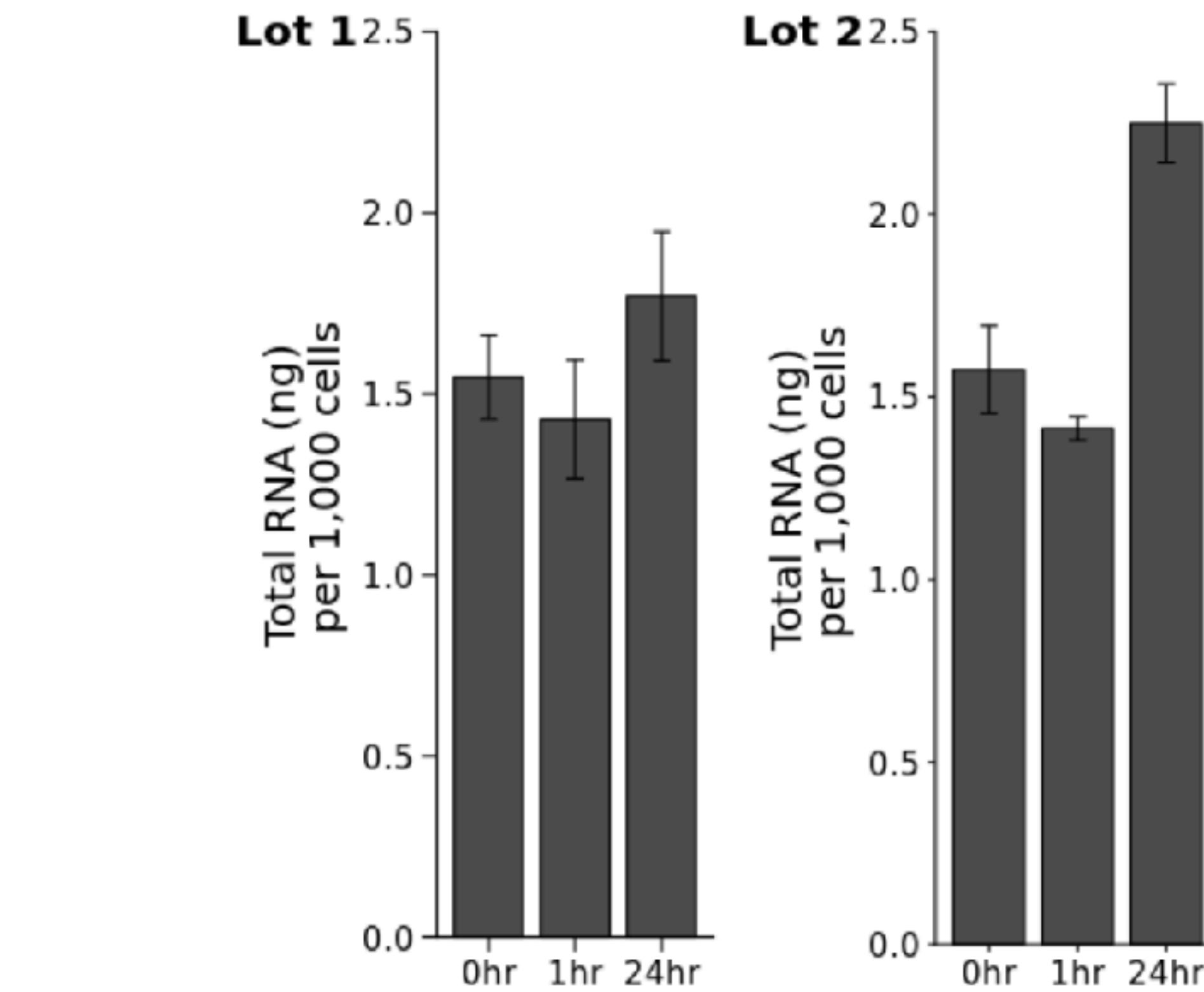
eLife’s prototype lets scientists *modify the software underlying figures to validate, build on, or better understand the work.*

This is a [Reproducible document](#). See the [original article or source](#).

NOTE: Below is a reproducible version of Figure 1B. You can inspect the code, make changes and run the code by pressing SHIFT+ENTER. The data used can be [downloaded here](#).

R Script

status: **ready** (run code with ↵)



Total RNA levels following c-Myc overexpression



setting the stage

replicability

same research question

same results

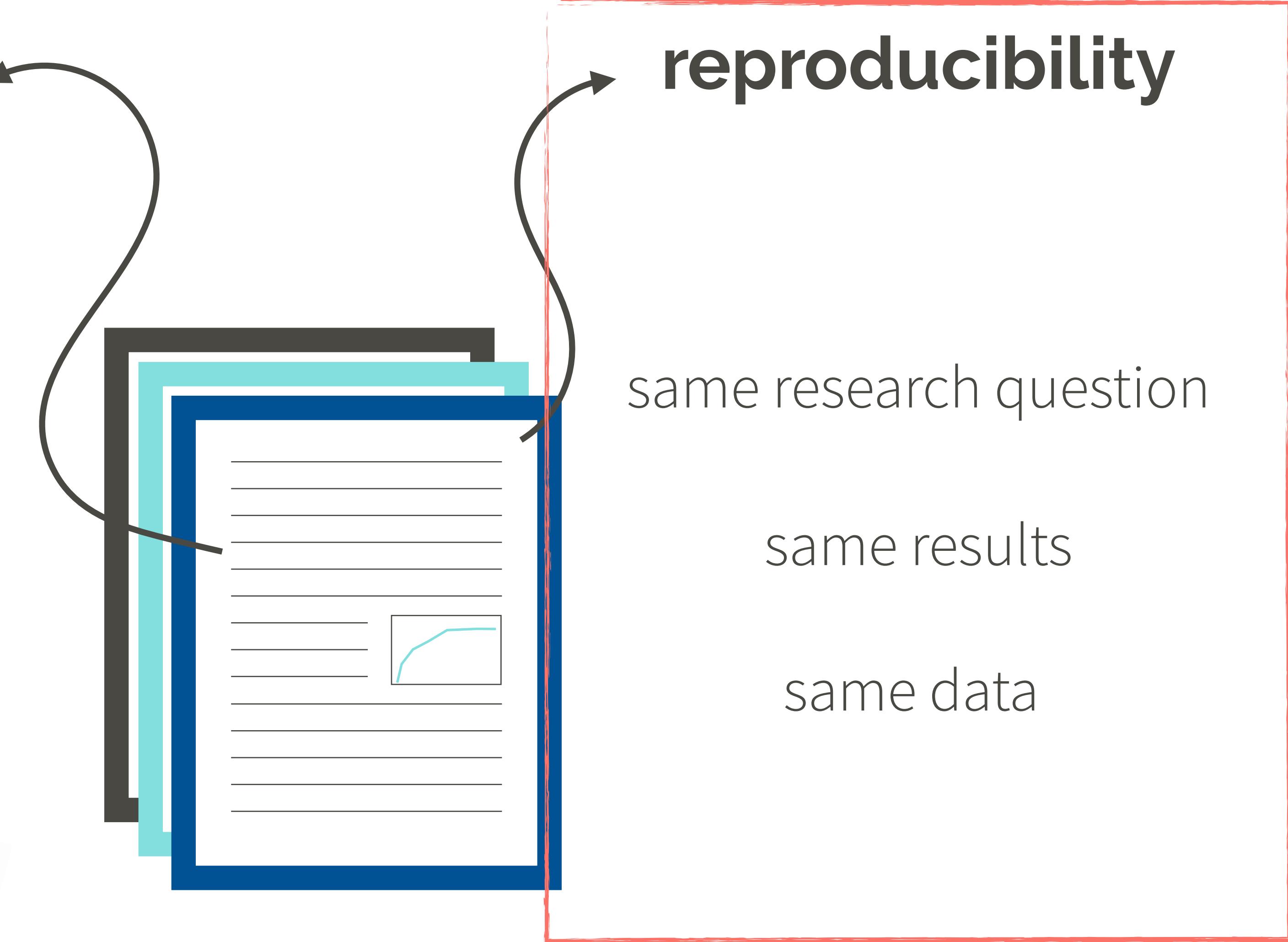
new data

reproducibility

same research question

same results

same data

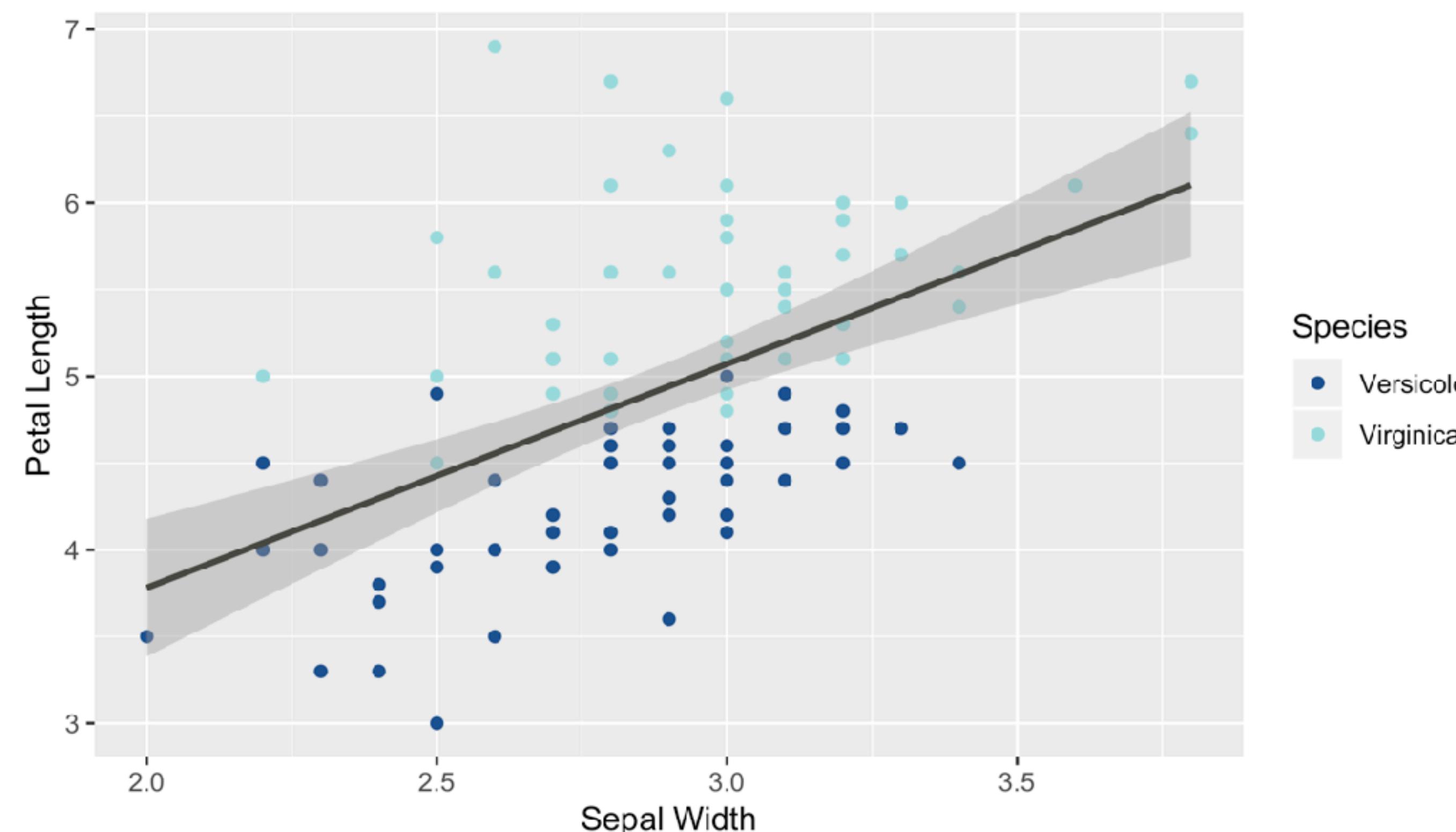


e.g.

Table 1. Regression output for predicting petal length from sepal width.

Term	Estimate	Std. Error	Statistic	p-value
(Intercept)	9.06	0.929	9.76	1.13e-17
Sepal.Width	-1.74	0.301	-5.77	4.51e- 8

Figure 2. Relationship between petal length and sepal width

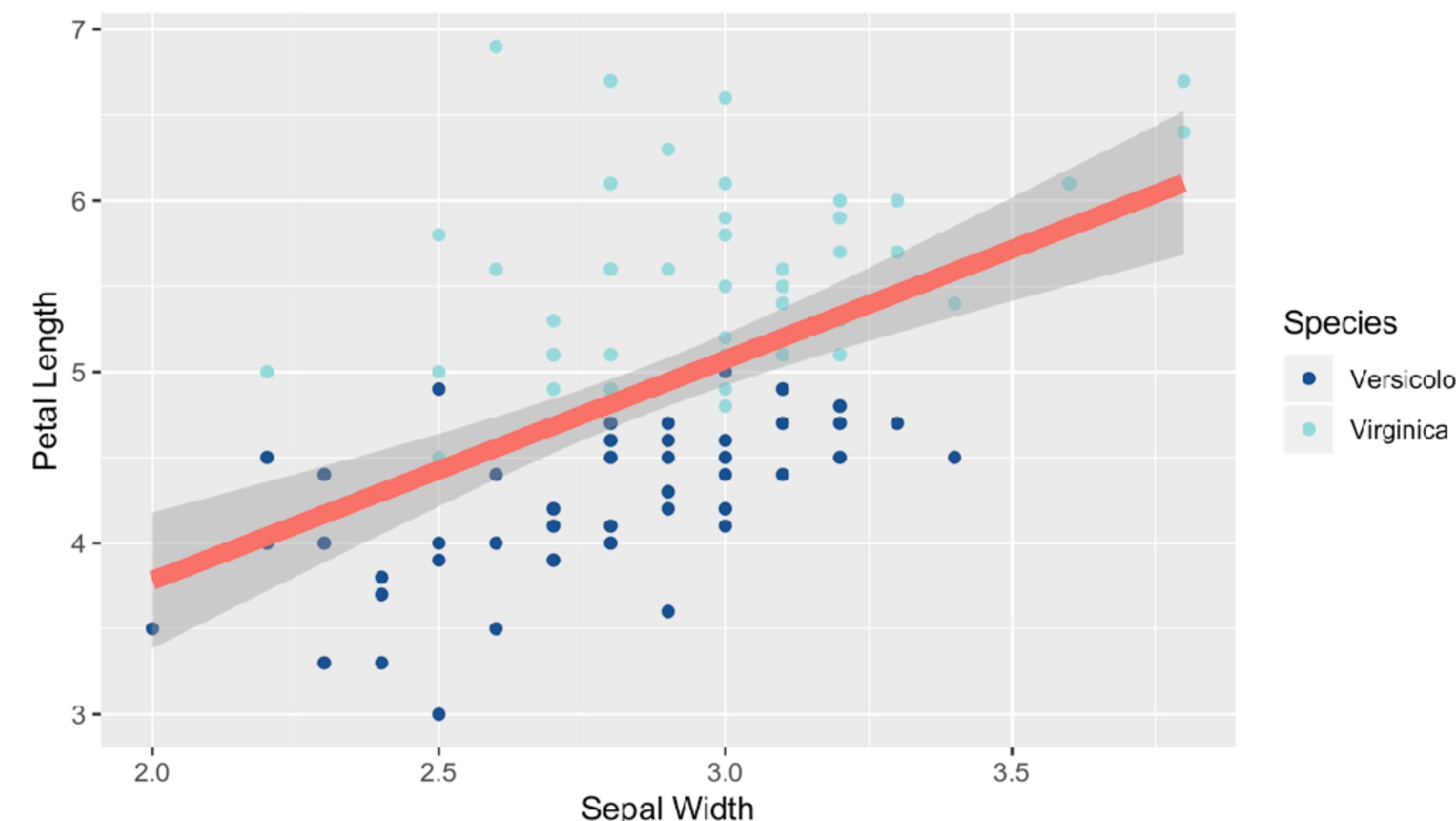


e.g.

Table 1. Regression output for predicting petal length from sepal width.

Term	Estimate	Std. Error	Statistic	p-value
(Intercept)	9.06	0.929	9.76	1.13e-17
Sepal.Width	-1.74	0.301	-5.77	4.51e- 8

Figure 2. Relationship between petal length and sepal width



analysis

report



```
# fit model  
model <- lm(Petal.Length ~ Sepal.Width, data = iris)  
  
# print model summary  
tidy(model)
```

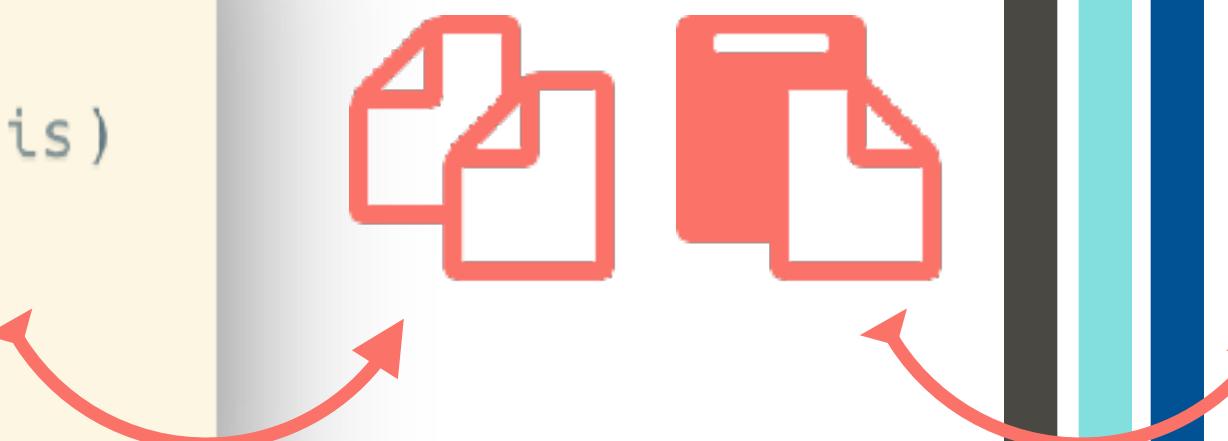


Table 1. Regression output for predicting petal length from sepal width.

Term	Estimate	Std. Error	Statistic	p-value
(Intercept)	9.06	0.929	9.76	1.13e-17
Sepal.Width	-1.74	0.301	-5.77	4.51e- 8

analysis

report



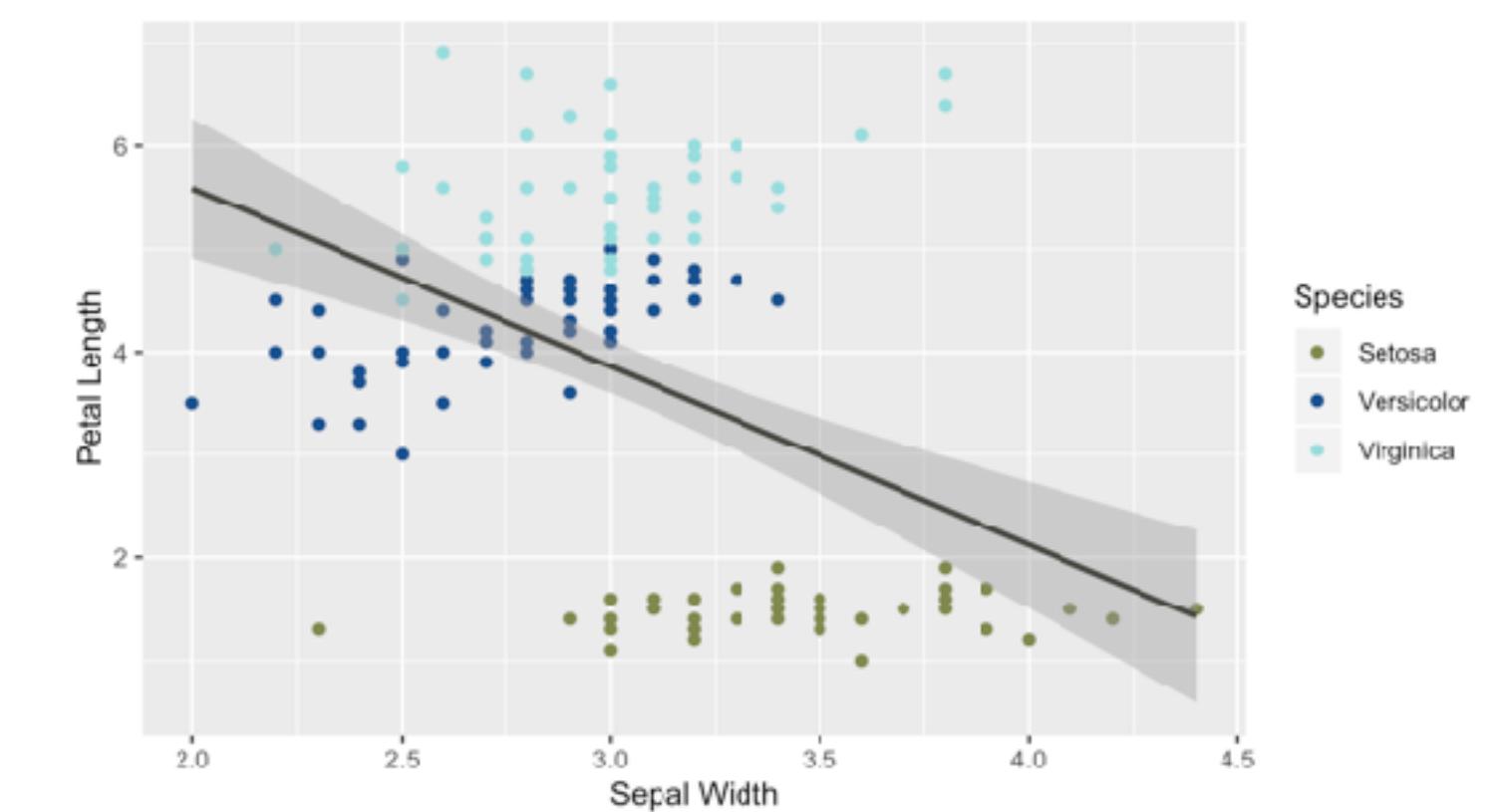
```
# visualize the relationship
ggplot(iris) +
  geom_point(
    aes(x = Sepal.Width, y = Petal.Length, color = Species))
  ) +
  geom_smooth(
    aes(x = Sepal.Width, y = Petal.Length),
    method = "lm"
  )
```



Table 1. Regression output for predicting petal length from sepal width.

Term	Estimate	Std. Error	Statistic	p-value
(Intercept)	9.06	0.929	9.76	1.13e-17
Sepal.Width	-1.74	0.301	-5.77	4.51e- 8

Figure 2. Relationship between petal length and sepal width



analysis

report



```
# filter out Setosas
iris_nonsetosa <- iris %>%
  filter(Species != "setosa")

# visualize the relationship
ggplot(iris_nonsetosa) +
  geom_point(
    aes(x = Sepal.Width, y = Petal.Length, color = Species)
  ) +
  geom_smooth(
    aes(x = Sepal.Width, y = Petal.Length),
    method = "lm"
  )
```

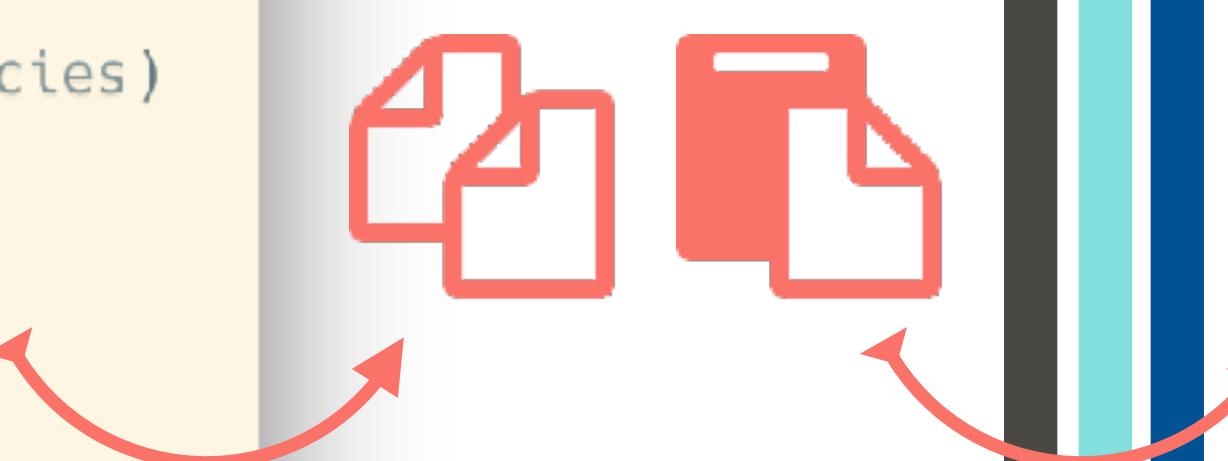


Table 1. Regression output for predicting petal length from sepal width.

Term	Estimate	Std. Error	Statistic	p-value
(Intercept)	9.06	0.929	9.76	1.13e-17
Sepal.Width	-1.74	0.301	-5.77	4.51e- 8

Figure 2. Relationship between petal length and sepal width

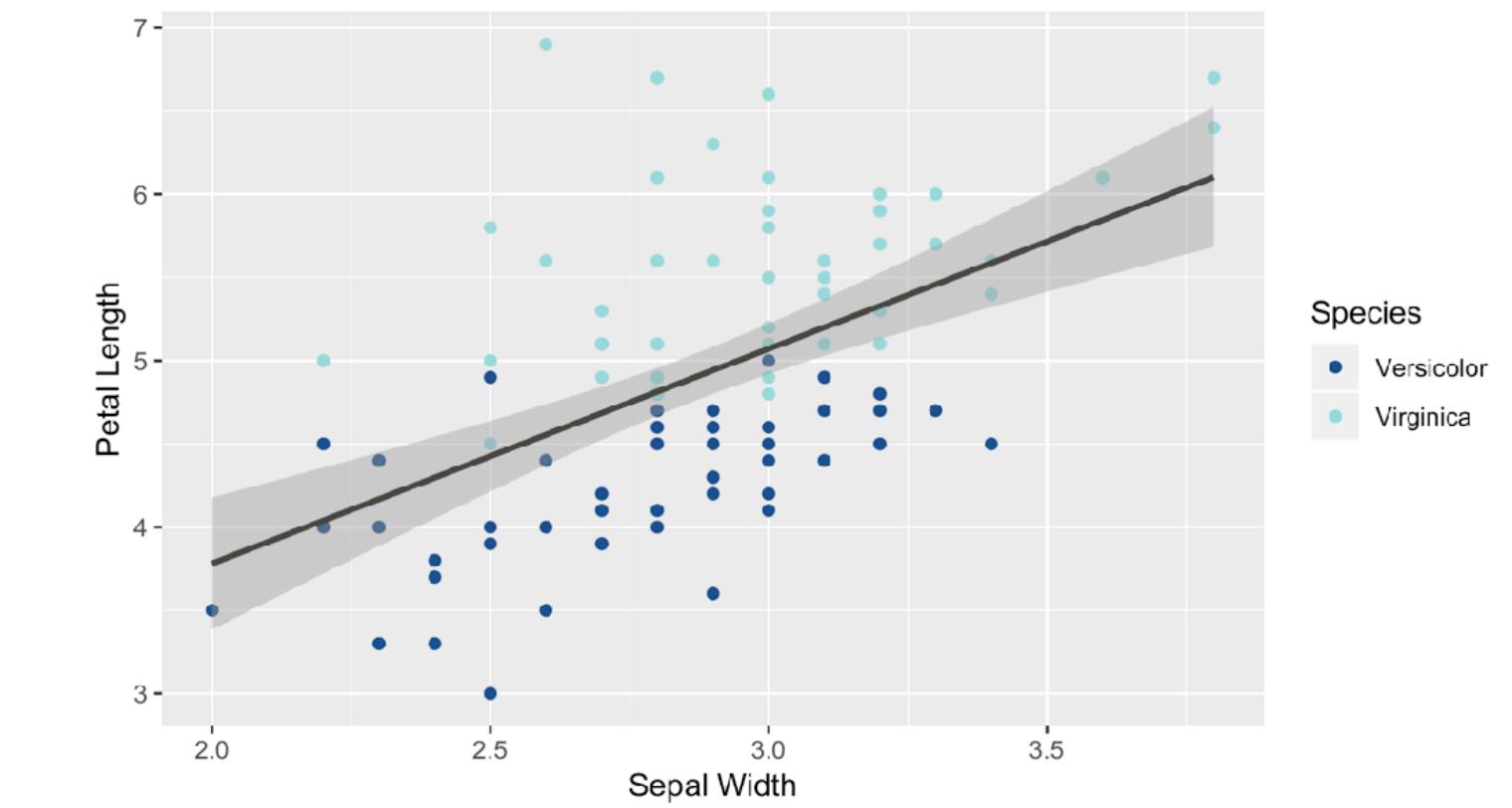
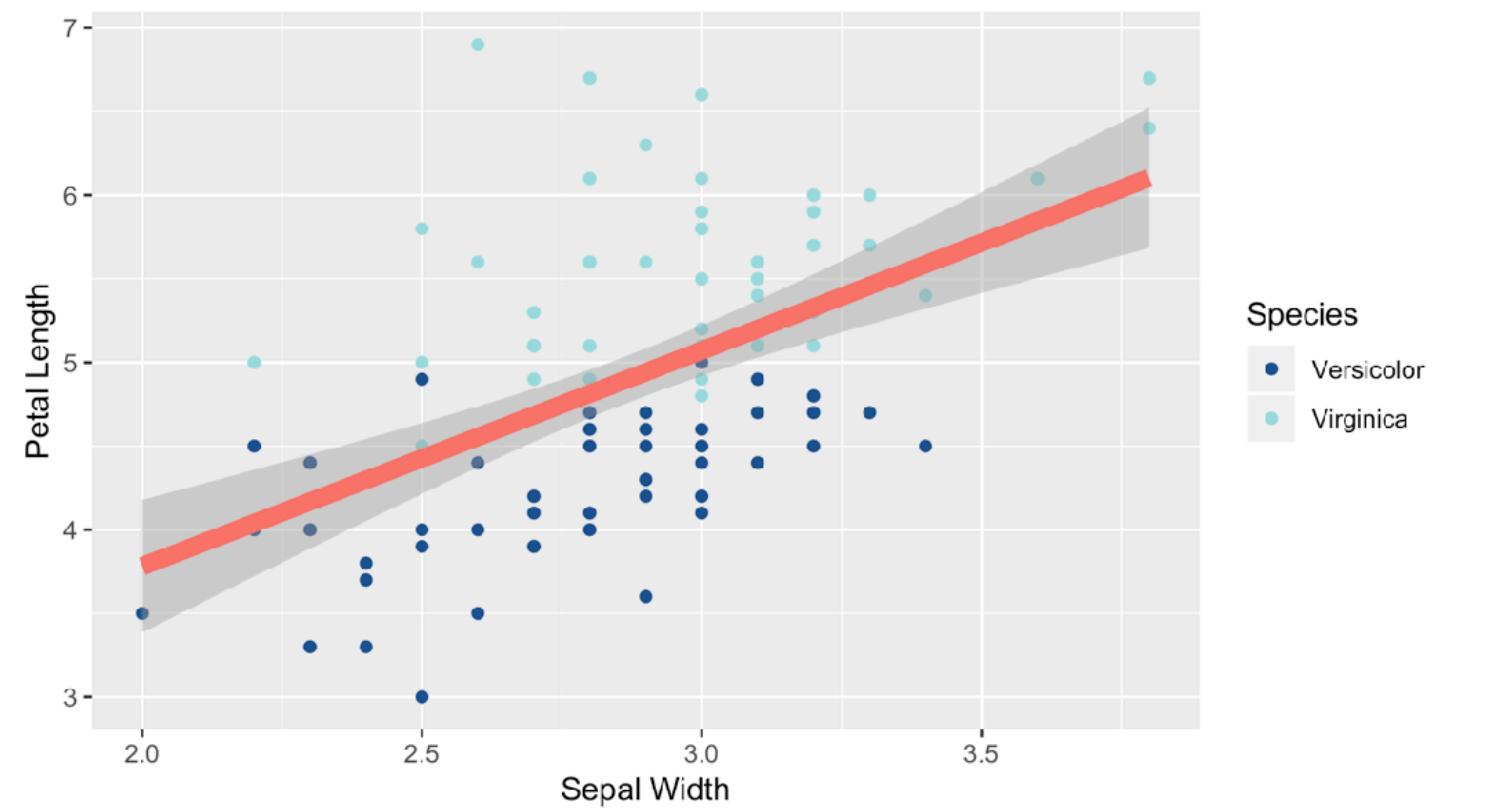
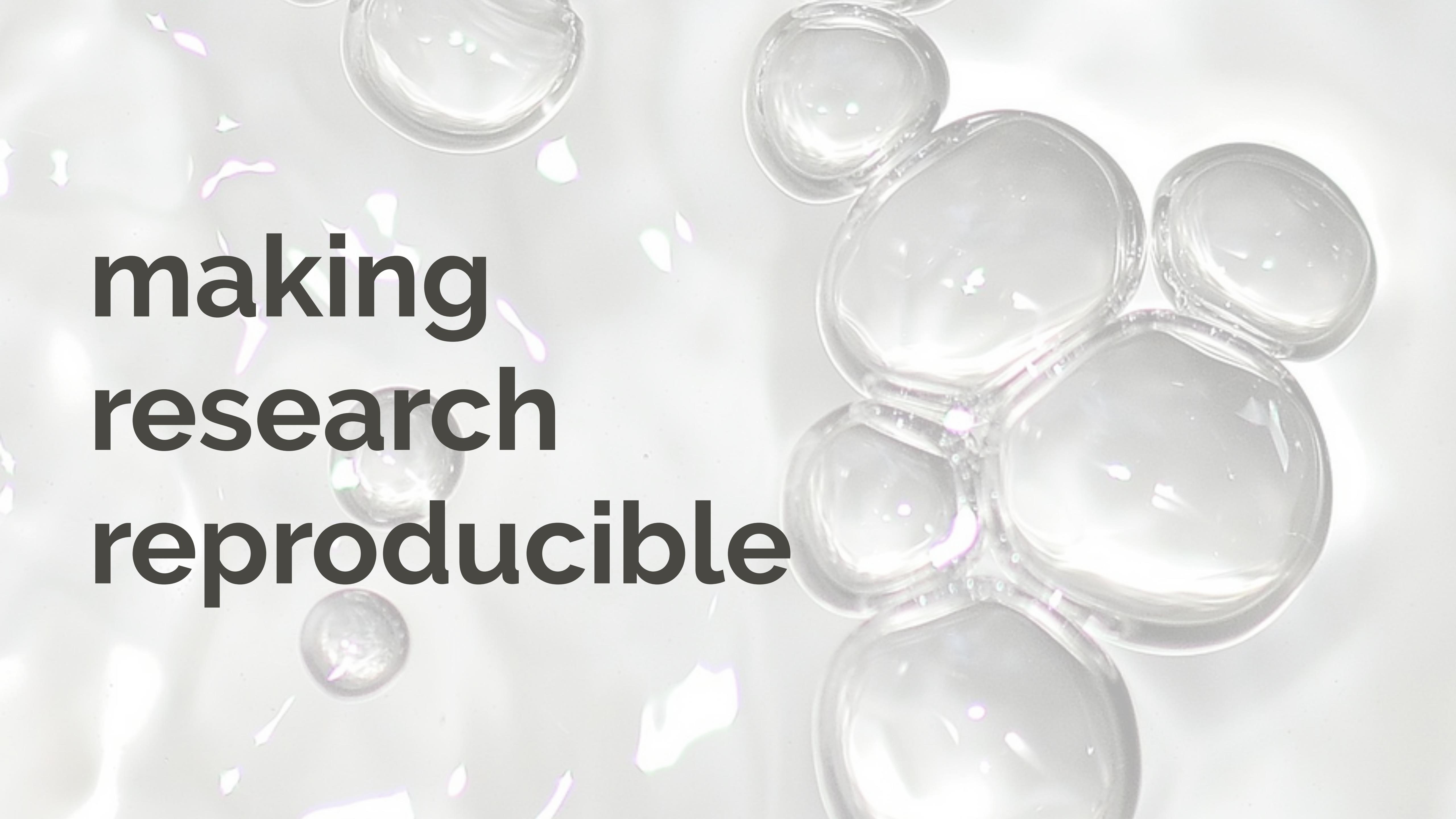


Table 1. Regression output for predicting petal length from sepal width.

Term	Estimate	Std. Error	Statistic	p-value
(Intercept)	9.06	0.929	9.76	1.13e-17
Sepal.Width	-1.74	0.301	-5.77	4.51e- 8

Figure 2. Relationship between petal length and sepal width



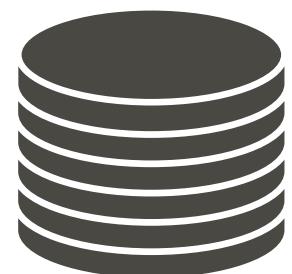


**making
research
reproducible**

“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.”

– David Donoho, paraphrasing Jon Claerbout

make



raw data



code & documentation to reproduce the analysis



specifications of your computational environment

available and accessible

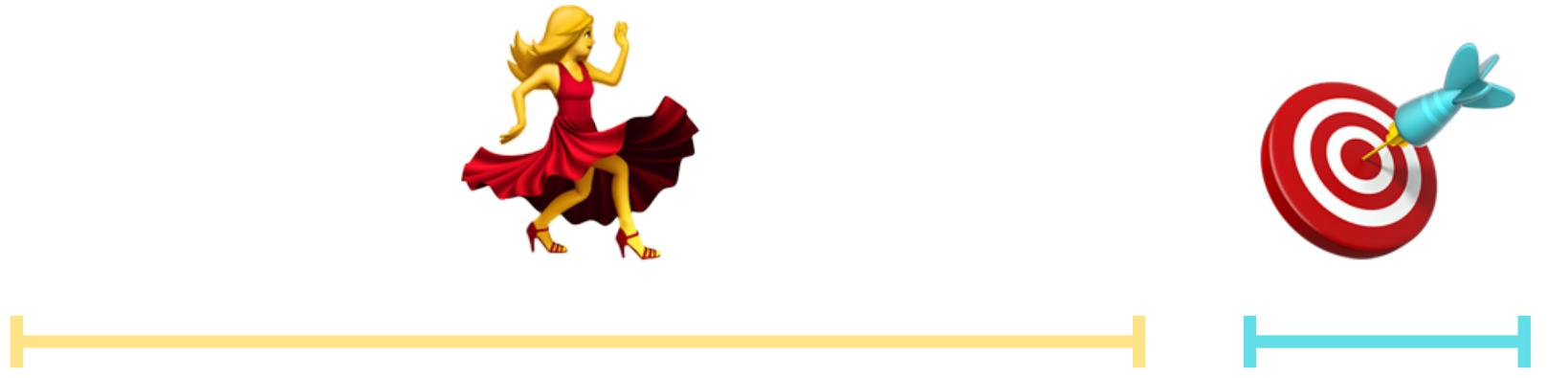
Peng, Roger. "The reproducibility crisis in science: A statistical counterattack." *Significance* 12.3 (2015): 30-32.

Gentleman, Robert, and Duncan Temple Lang. "Statistical analyses and reproducible research." *Journal of Computational and Graphical Statistics* 16.1 (2007): 1-23.

“The most important tool is
the **mindset**, when starting,
that the end product will be
reproducible.”

– Keith Baggerly

nobody,
not even yourself,
can recreate any part
of your analysis



push button
reproducibility
in published work

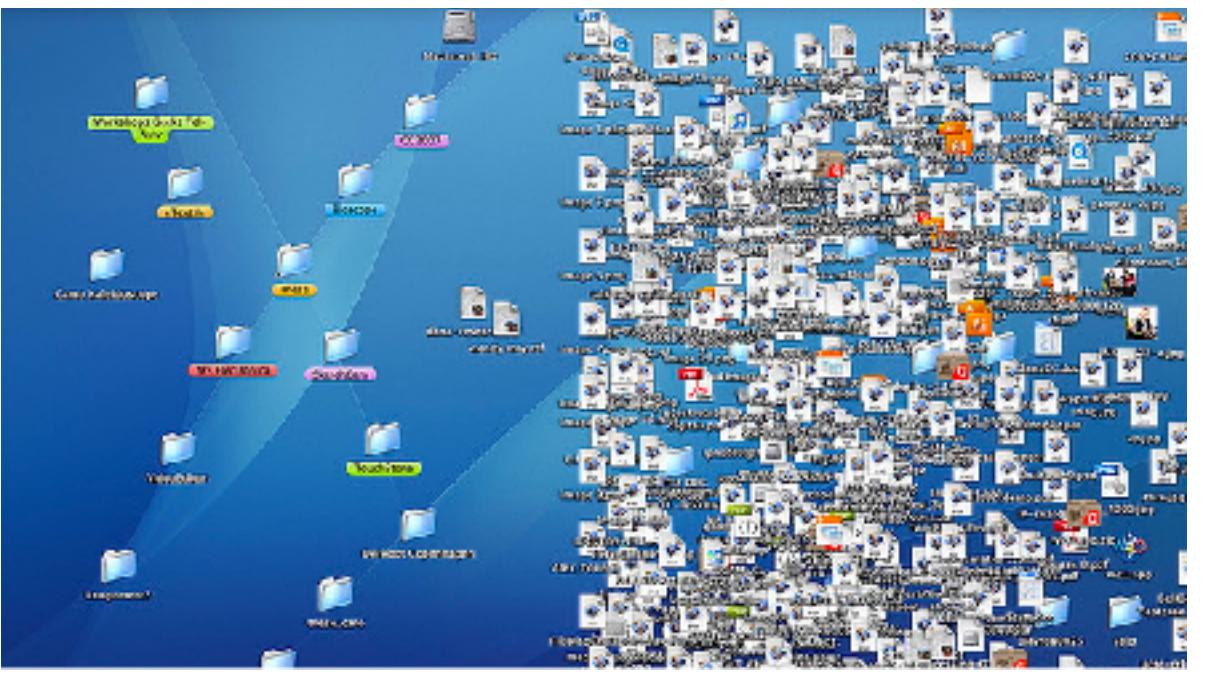
“There’s **no one-size-fits-all solution**
for computational reproducibility.”

8 principles
but the following[^] might help...

1

organize
your
project

level of organization



simpler analysis

-  raw-data
-  processed-data
-  manuscript
 - └ manuscript.Rmd

more complex analysis

-  raw-data
-  processed-data
-  scripts
-  figures
-  manuscript
 - └ manuscript.Rmd

stick with the conventions of your peers



write
READMEs
liberally



raw-data

- └ README.md
- └ airlines.csv
- └ airports.csv
- └ flights.csv
- └ planes.csv
- └ weather.csv



processed-data



scripts



figures



manuscript



README

This folder contains the raw data for the project.

All datasets were downloaded from openflights.org/data.html on 2019-04-01.

- airlines: Airline names
- airports: Airports metadata
- flights: Flight data
- planes: Plane metadata
- weather: Hourly weather data



keep data
tidy &
machine readable

Student	Exam Grade		
Name	1	2	Major
Barney Donaldson	89	76	Data Science, Public Policy
Clay Whelan	67	83	Public Policy
Simran Bass	82	90	Statistics
Chante Munro	45	72	Political Science, Statistics
Gabrielle Cherry	32	79	.
Kush Piper	98	sick	Statistics
Faizan Ratliff	82	75	Data Science
Torin Ruiz	70	80	Sociology, Statistics
Reiss Richardson	missed exam	34	Neuroscience
Ajwa Cochran	50	65	Data Science

→ record
code +
document
non-code
steps +
write
tests

name	exam_1	exam_2	first_major	second_major	participation
Barney Donaldson	89	76	Data Science	Public Policy	ok
Clay Whelan	67	83	Public Policy	NA	ok
Simran Bass	82	90	Statistics	NA	ok
Chante Munro	45	72	Political Science	Statistics	Low
Gabrielle Cherry	32	79	NA	NA	ok
Kush Piper	98	NA	Statistics	NA	ok
Faizan Ratliff	82	75	Data Science	NA	ok
Torin Ruiz	70	80	Sociology	Statistics	ok
Reiss Richardson	NA	34	Neuroscience	NA	low
Ajwa Cochran	50	65	Data Science	NA	low

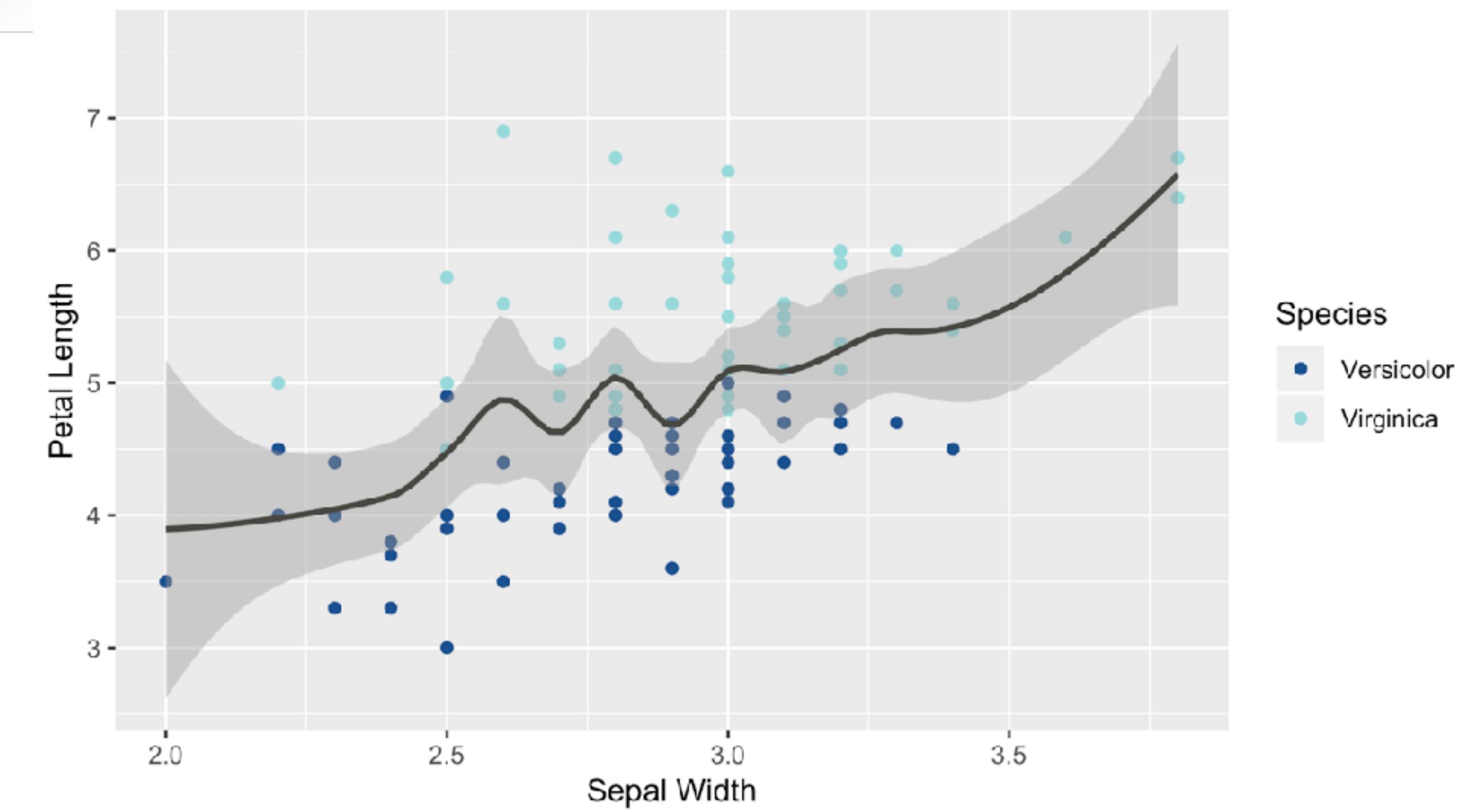
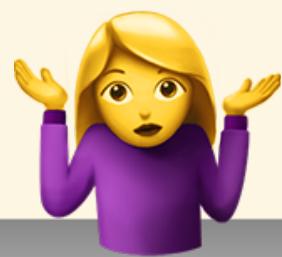
Low participation

1

comment
your
code



```
# use loess smoothing
ggplot(iris_nonsetosa) +
  geom_point(
    aes(x = Sepal.Width, y = Petal.Length, color = Species)
  ) +
  geom_smooth(
    aes(x = Sepal.Width, y = Petal.Length),
    method = "loess", span = 0.375
  )
```



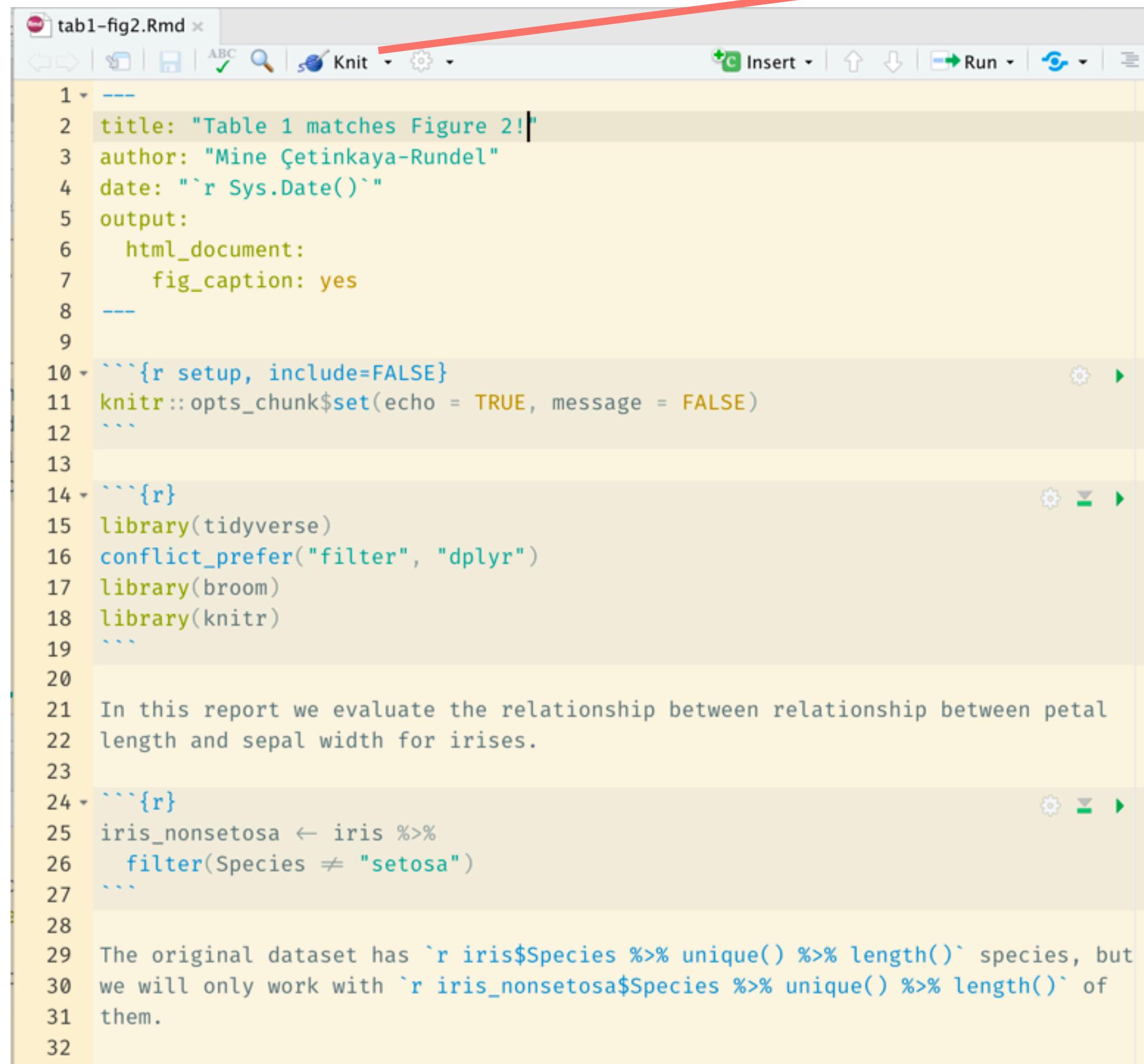


use
literate
programming

tab1-fig2.Rmd

ABC Knit Insert Run R Markdown

```
1 ---  
2 title: "Table 1 matches Figure 2!"  
3 author: "Mine Çetinkaya-Rundel"  
4 date: "`r Sys.Date()`"  
5 output:  
6   html_document:  
7     fig_caption: yes  
8 ---  
9  
10 ````{r setup, include=FALSE}  
11 knitr::opts_chunk$set(echo = TRUE, message = FALSE)  
12 ````  
13  
14 ````{r}  
15 library(tidyverse)  
16 conflict_prefer("filter", "dplyr")  
17 library(broom)  
18 library(knitr)  
19 ````  
20  
21 In this report we evaluate the relationship between relationship between petal  
22 length and sepal width for irises.  
23  
24 ````{r}  
25 iris_nonsetosa <- iris %>%  
26   filter(Species != "setosa")  
27 ````  
28  
29 The original dataset has `r iris$Species %>% unique() %>% length()` species, but  
30 we will only work with `r iris_nonsetosa$Species %>% unique() %>% length()` of  
31 them.  
32  
33 ## Model  
34  
35 The model results are below.  
36
```



```
tab1-fig2.Rmd x
Insert Run Knit ABC Find Publish C

1 ---
2 title: "Table 1 matches Figure 2!"
3 author: "Mine Çetinkaya-Rundel"
4 date: `r Sys.Date()`
5 output:
6   html_document:
7     fig_caption: yes
8 ---

9
10 ```{r setup, include=FALSE}
11 knitr::opts_chunk$set(echo = TRUE, message = FALSE)
12 ...
13
14 ```{r}
15 library(tidyverse)
16 conflict_prefer("filter", "dplyr")
17 library(broom)
18 library(knitr)
19 ...
20
21 In this report we evaluate the relationship between relationship between petal
22 length and sepal width for irises.
23
24 ```{r}
25 iris_nonsetosa <- iris %>%
26   filter(Species != "setosa")
27 ...
28
29 The original dataset has `r iris$Species %>% unique() %>% length()` species, but
30 we will only work with `r iris_nonsetosa$Species %>% unique() %>% length()` of
31 them.
32
```

~/Desktop/Talks/DataTech/datatech-2019/scripts/tab1-fig2/tab1-fig2.html
tab1-fig2.html | Open in Browser | Find | Publish | C

Table 1 matches Figure 2!

Mine Çetinkaya-Rundel
2019-03-14

```
library(tidyverse)
conflict_prefer("filter", "dplyr")
library(broom)
library(knitr)
```

In this report we evaluate the relationship between relationship between petal length and sepal width for irises.

```
iris_nonsetosa <- iris %>%
  filter(Species != "setosa")
```

The original dataset has 3 species, but we will only work with 2 of them.

Model

The model results are below.

```
m_pl_sw <- lm(Petal.Length ~ Sepal.Width, data = iris_nonsetosa)
tidy_m_pl_sw <- tidy(m_pl_sw)
kable(tidy_m_pl_sw,
      caption = "Table 1. Regression output for predicting petal length
from sepal width.",
      digits = 2)
```

Table 1. Regression output for predicting petal length from sepal width.

term	estimate	std.error	statistic	p.value
(Intercept)	1.20	0.62	1.94	0.06
Sepal.Width	1.29	0.21	6.02	0.00

```

tab1-fig2.Rmd x
ABC Knit Insert Run Find
1 ---
2 title: "Table 1 matches Figure 2!"
3 author: "Mine Çetinkaya-Rundel"
4 date: `r Sys.Date()`
5 output:
6   html_document:
7     fig_caption: yes
8 ---
9
10 ````{r setup, include=FALSE}
11 knitr::opts_chunk$set(echo = FALSE, message = FALSE)
12 ...
13 ````{r}
14 library(tidyverse)
15 conflict_prefer("filter", "dplyr")
16 library(broom)
17 library(knitr)
18 ...
19
20 In this report we evaluate the relationship between petal length and sepal width for irises.
21
22 The original dataset has `r iris$Species %>% unique() %>% length()` species, but
23 we will only work with `r iris_nonsetosa$Species %>% unique() %>% length()` of
24 them.
25 iris_nonsetosa <- iris %>%
26   filter(Species != "setosa")
27 ...
28
29 The original dataset has `r iris$Species %>% unique() %>% length()` species, but
30 we will only work with `r iris_nonsetosa$Species %>% unique() %>% length()` of
31 them.
32

```

Table 1 matches Figure 2!

Mine Çetinkaya-Rundel

2019-03-14

In this report we evaluate the relationship between relationship between petal length and sepal width for irises.

The original dataset has 3 species, but we will only work with 2 of them.

Model

The model results are below.

Table 1. Regression output for predicting petal length from sepal width.

term	estimate	std.error	statistic	p.value
(Intercept)	1.20	0.62	1.94	0.06
Sepal.Width	1.29	0.21	6.02	0.00

The slope of the regression model is 1.29.

Visualize

The figure below shows the relationship between these variables, and we observe a positive slope in this visualization as well.



```
tab1-fig2.Rmd x
ABC Knit Insert Run Find
1 ---
2 title: "Table 1 matches Figure 2!"
3 author: "Mine Çetinkaya-Rundel"
4 date: "`r Sys.Date()`"
5 output:
6   html_document:
7     fig_caption: yes
8 ---
9
10 ``{r setup, include=FALSE}
11 knitr::opts_chunk$set(echo = FALSE, message = FALSE)
12 ...
13 ``{r}
14 library(tidyverse)
15 conflict_prefer("filter", "dplyr")
16 library(broom)
17 library(knitr)
18 ...
19
20 In this report we evaluate the relationship between petal length and sepal width for irises.
21
22 The original dataset has `r iris$Species %>% unique() %>% length()` species, but
23 we will only work with `r iris_nonsetosa$Species %>% unique() %>% length()` of
24 them.
25 iris_nonsetosa <- iris %>%
26   filter(Species != "setosa")
27 ...
28
29 The original dataset has `r iris$Species %>% unique() %>% length()` species, but
30 we will only work with `r iris_nonsetosa$Species %>% unique() %>% length()` of
31 them.
32
```

Table 1 matches Figure 2!

Mine Çetinkaya-Rundel

2019-03-14

In this report we evaluate the relationship between relationship between petal length and sepal width for irises.

The original dataset has 3 species, but we will only work with 2 of them.

Model

The model results are below.

Table 1. Regression output for predicting petal length from sepal width.

term	estimate	std.error	statistic	p.value
(Intercept)	1.20	0.62	1.94	0.06
Sepal.Width	1.29	0.21	6.02	0.00

The slope of the regression model is 1.29.

Visualize

The figure below shows the relationship between these variables, and we observe a positive slope in this visualization as well.



tab1-fig2-py.Rmd

```
1 ---  
2 title: "Python in R Markdown"  
3 output: html_document  
4 ---  
5  
6 ```{r setup, include=FALSE}  
7 knitr::opts_chunk$set(echo = TRUE)  
8 ...  
9  
10 ```{r}  
11 library(reticulate)  
12 use_virtualenv(virtualenv = "r-reticulate")  
13 ...  
14  
15 ```{python}  
16 import pandas as pd  
17 import numpy as np  
18  
19 df = pd.DataFrame({'A': 1.,  
20                     'B': pd.Timestamp('20130102'),  
21                     'C': pd.Series(1, index=list(range(4)), dtype='float32'),  
22                     'D': np.array([3] * 4, dtype='int32'),  
23                     'E': pd.Categorical(["test", "train", "test", "train"]),  
24                     'F': 'foo'})  
25 df  
26  
27 df.describe()  
28 ...  
29  
30
```

~/Desktop/tab1-fig2-py.html

Python in R Markdown

```
library(reticulate)  
use_virtualenv(virtualenv = "r-reticulate")  
  
import pandas as pd  
import numpy as np  
  
df = pd.DataFrame({'A': 1.,  
                   'B': pd.Timestamp('20130102'),  
                   'C': pd.Series(1, index=list(range(4)), dtype='float32'),  
                   'D': np.array([3] * 4, dtype='int32'),  
                   'E': pd.Categorical(["test", "train", "test", "train"]),  
                   'F': 'foo'})  
df  
  
##      A          B    C   D      E    F  
## 0  1.0 2013-01-02  1.0  3  test  foo  
## 1  1.0 2013-01-02  1.0  3  train  foo  
## 2  1.0 2013-01-02  1.0  3  test  foo  
## 3  1.0 2013-01-02  1.0  3  train  foo  
  
df.describe()  
  
##      A      C      D  
## count  4.0  4.0  4.0  
## mean   1.0  1.0  3.0  
## std    0.0  0.0  0.0  
## min    1.0  1.0  3.0  
## 25%   1.0  1.0  3.0  
## 50%   1.0  1.0  3.0  
## 75%   1.0  1.0  3.0  
## max   1.0  1.0  3.0
```

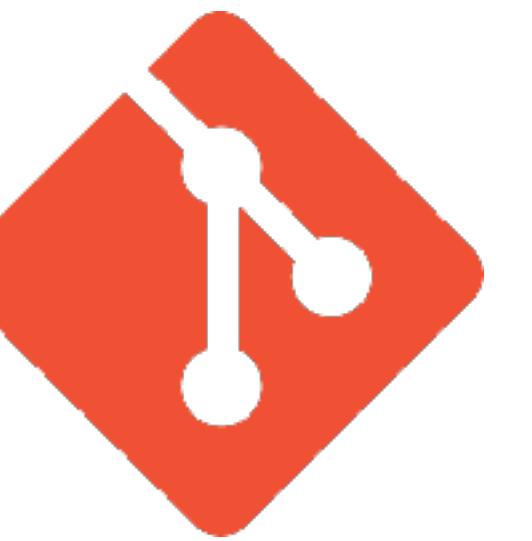


use
version
control

```
tab1-fig2.Rmd x ABC Knit Insert Run
```

```
1 ---  
2 title: "Table 1 matches Figure 2!"  
3 author: "Mine Çetinkaya-Rundel"  
4 date: "`r Sys.Date()`"  
5 output:  
6   html_document:  
7     fig_caption: yes  
---  
  
10 ````{r setup, include=FALSE}  
11 knitr::opts_chunk$set(echo = FALSE, message = FALSE)  
12 ````  
13  
14 ````{r}  
15 library(tidyverse)  
16 conflict_prefer("filter", "dplyr")  
17 library(broom)  
18 library(knitr)  
19 ````  
20  
21 In this report we evaluate the relationship between relationship between petal  
22 length and sepal width for irises.  
23  
24 ````{r}  
25 iris_nonsetosa <- iris %>%  
26   filter(Species != "setosa")  
27 ````  
28  
29 The original dataset has `r iris$Species %>% unique() %>% length()` species, but  
30 we will only work with `r iris_nonsetosa$Species %>% unique() %>% length()` of  
31 them.  
32
```

changes
tracked by



hosted
on



[Let's Git started](#)[License](#)[1 Why Git? Why GitHub?](#)[2 Contributors](#)[3 Workshops](#)[I Installation](#)[Half the battle](#)[4 Register a GitHub account](#)[5 Install or upgrade R and RStudio](#)[6 Install Git](#)[7 Introduce yourself to Git](#)[8 Install a Git client](#)[II Connect Git, GitHub, RStudio](#)[Can you hear me now?](#)[9 Connect to GitHub](#)[10 Cache credentials for HTTPS](#)[11 Set up keys for SSH](#)[12 Connect RStudio to Git and GitHub](#)[13 Detect Git from RStudio](#)

Happy Git and GitHub for the useR

Jenny Bryan, the STAT 545 TAs, Jim Hester

Let's Git started



?

automate
your
process



raw-data



processed-data



scripts

- └ 00-analyse.R
- └ 01-load-packages.R
- └ 02-load-data.R
- └ 03-clean-data.R
- └ 04-explore.R
- └ 05-model.R
- └ 06-summarise.R



```
00-analyse.R x
Source on Save | Run | Source | R Script
# run all -----
source("01-load-packages.R")
source("02-load-data.R")
source("03-clean-data.R")
source("04-explore.R")
source("05-model.R")
source("06-summarise.R")
```



figures



manuscript

minimal make

A minimal tutorial on make

I would argue that the most important tool for reproducible research is not [Sweave](#) or [knitr](#) but [GNU make](#).

Consider, for example, all of the files associated with a manuscript. In the simplest case, I would have an [R](#) script for each figure plus a [LaTeX](#) file for the main text. And then a [BibTeX](#) file for the references.

Compiling the final PDF is a bit of work:

- Run each R script through R to produce the relevant figure.
- Run latex and then bibtex and then latex a couple of more times.

And the R scripts need to be run before latex is, and only if they've changed.

A simple example

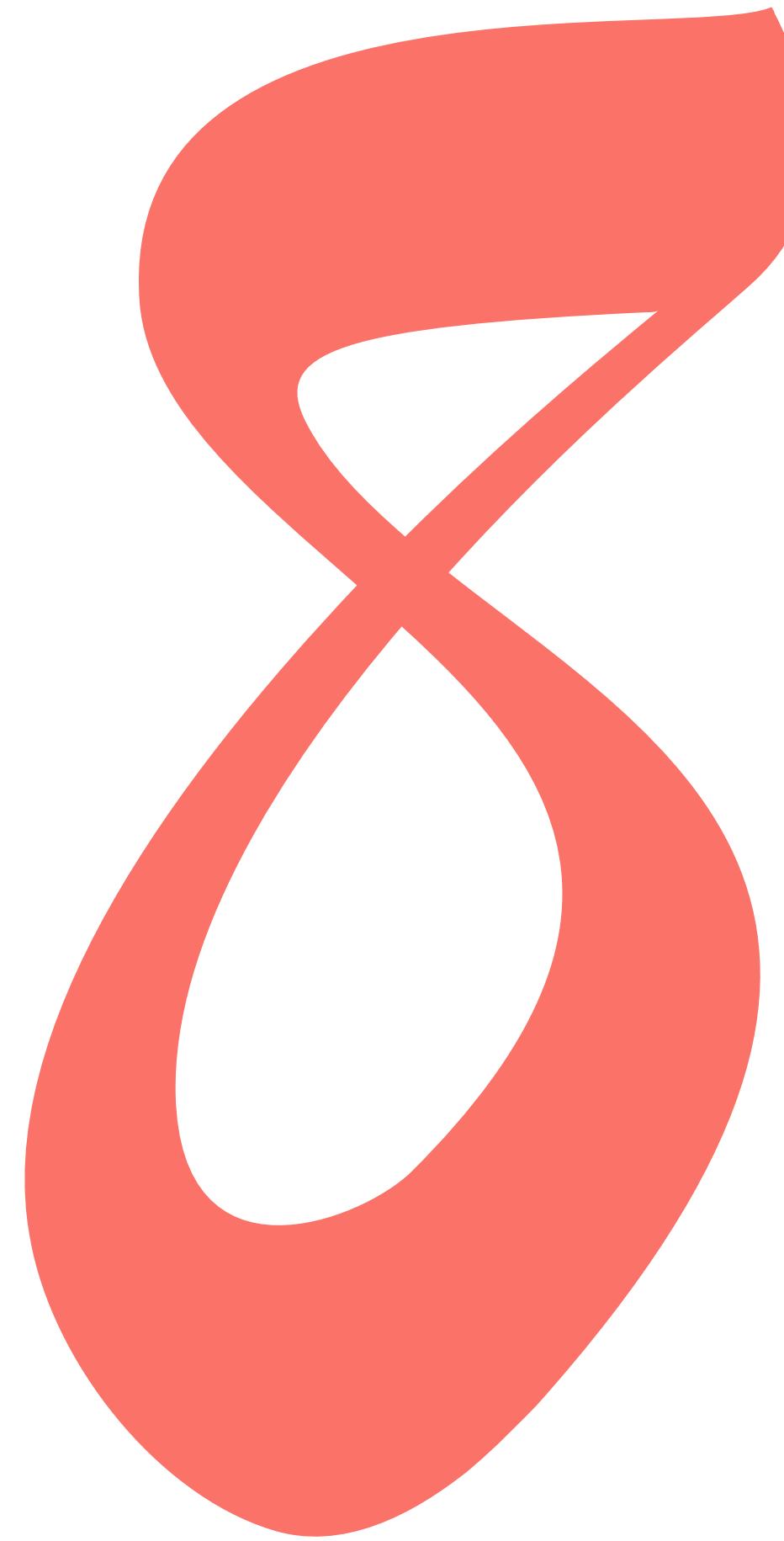
[GNU make](#) makes this easy. In your directory for the manuscript, you create a text file called `Makefile` that looks something like [the following](#) (here using [pdflatex](#)).

```
mypaper.pdf: mypaper.bib mypaper.tex Figs/fig1.pdf Figs/fig2.pdf
    pdflatex mypaper
    bibtex mypaper
    pdflatex mypaper
    pdflatex mypaper

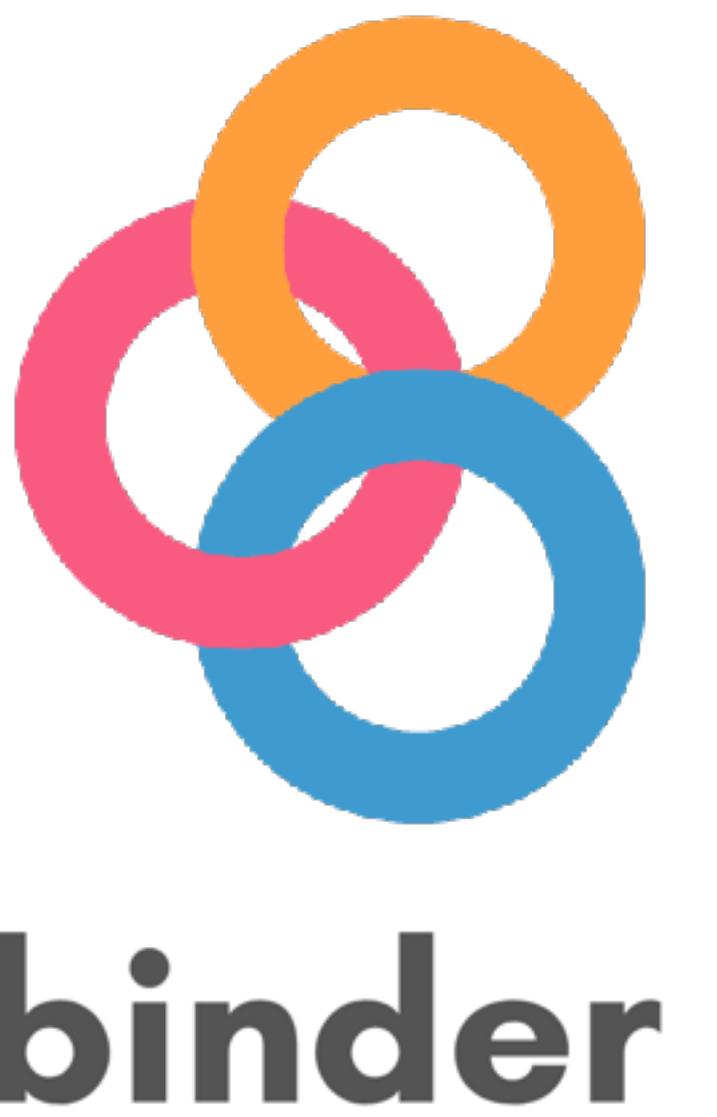
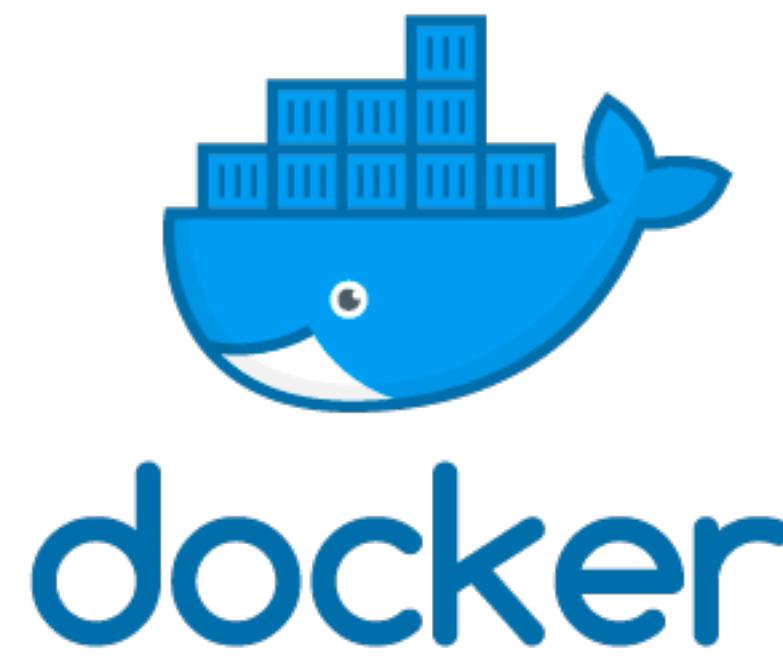
Figs/fig1.pdf: R/fig1.R
    cd R;R CMD BATCH fig1.R

Figs/fig2.pdf: R/fig2.R
    cd R;R CMD BATCH fig2.R
```

Each batch of lines indicates a file to be created (the *target*), the files it depends on (the *prerequisites*), and then a set of commands needed to construct the target from the dependent files. Note that the lines with the commands *must* start with a **tab** character (**not spaces**).



share
computing
environment



- 1 organize your project
- 2 write **READMEs** liberally
- 3 keep data **tidy & machine readable**
- 4 comment your code
- 5 use **literate programming**
- 6 use **version control**
- 7 automate your process
- 8 share computing **environment**

PERSPECTIVE

Good enough practices in scientific computing

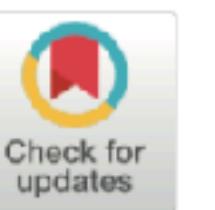
Greg Wilson^{1*}, Jennifer Bryan², Karen Cranston³, Justin Kitzes⁴, Lex Nederbragt⁵, Tracy K. Teal⁶

1 Software Carpentry Foundation, Austin, Texas, United States of America, **2** RStudio and Department of Statistics, University of British Columbia, Vancouver, British Columbia, Canada, **3** Department of Biology, Duke University, Durham, North Carolina, United States of America, **4** Energy and Resources Group, University of California, Berkeley, Berkeley, California, United States of America, **5** Centre for Ecological and Evolutionary Synthesis, University of Oslo, Oslo, Norway, **6** Data Carpentry, Davis, California, United States of America

* These authors contributed equally to this work.

* gwilson@software-carpentry.org

Greg Wilson, Jennifer Bryan, Karen Cranston,
Justin Kitzes, Lex Nederbragt, Tracy K. Teal
“Good enough practices in scientific computing.”
PLoS computational biology 13.6 (2017): e1005510.



OPEN ACCESS

Citation: Wilson G, Bryan J, Cranston K, Kitzes J, Nederbragt L, Teal TK (2017) Good enough practices in scientific computing. PLoS Comput Biol 13(6): e1005510. <https://doi.org/10.1371/journal.pcbi.1005510>

Editor: Francis Ouellette, Ontario Institute for Cancer Research, CANADA

Published: June 22, 2017

Copyright: © 2017 Wilson et al. This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Funding: The authors received no specific funding for this work.

Competing interests: The authors have declared that no competing interests exist.

Author summary

Computers are now essential in all branches of science, but most researchers are never taught the equivalent of basic lab skills for research computing. As a result, data can get lost, analyses can take much longer than necessary, and researchers are limited in how effectively they can work with software and data. Computing workflows need to follow the same practices as lab projects and notebooks, with organized data, documented steps, and the project structured for reproducibility, but researchers new to computing often don't know where to start. This paper presents a set of good computing practices that every researcher can adopt, regardless of their current level of computational skill. These practices, which encompass data management, programming, collaborating with colleagues, organizing projects, tracking work, and writing manuscripts, are drawn from a wide variety of published sources from our daily lives and from our work with volunteer organizations that have delivered workshops to over 11,000 people since 2010.

Overview

We present a set of computing tools and techniques that every researcher can and should consider adopting. These recommendations synthesize inspiration from our own work, from the experiences of the thousands of people who have taken part in Software Carpentry and Data Carpentry workshops over the past 6 years, and from a variety of other guides. Our recommendations are aimed specifically at people who are new to research computing.

Introduction

Three years ago, a group of researchers involved in Software Carpentry and Data Carpentry wrote a paper called “Best Practices for Scientific Computing” [1]. That paper provided recommendations for people who were already doing significant amounts of computation in their research. However, as computing has become an essential part of science for all researchers, there is a larger group of people new to scientific computing, and the question then becomes, “where to start?”

**looking
into the
future**



“The most important tool is
the **mindset**, when starting,
that the end product will be
reproducible.”

– Keith Baggerly

“The second most important
tool is **training**. ”

– Karl Broman

#1:

Convince data
scientists to adopt a
reproducible data
analysis workflow



#2:

Train new data
scientists who don't
have any other
workflow

statistics and data science educators who teach data analysis should be instilling best practices in students before they set out to do research



The results in Table 1
don't seem to correspond to
those in Figure 2



bit.ly/tab1-fig2-pydata



bit.ly/tab1-fig2

@minebocek

mine-cetinkaya-rundel

cetinkaya.mine@gmail.com