#76 Tokyo.R

2019.03.02

BeginneR Session
- Data Pipeline -

@kilometer00

Who！？

Who！？

名前： 三村 @kilometer

職業： ポスドク (こうがくはくし)

専門： 行動神経科学(霊長類)

脳イメージング

医療システム工学

R歴： ~ 10年ぐらい

流行: 青岛啤酒

BeginneR Session

2018.03.03 Tokyo.R #68
BeginneR Session – Data import / Export

2018.04.21 Tokyo.R #69
BeginneR Session – Data import / Export

2018.06.09 Tokyo.R #70
BeginneR Session – Bayesian Modeling

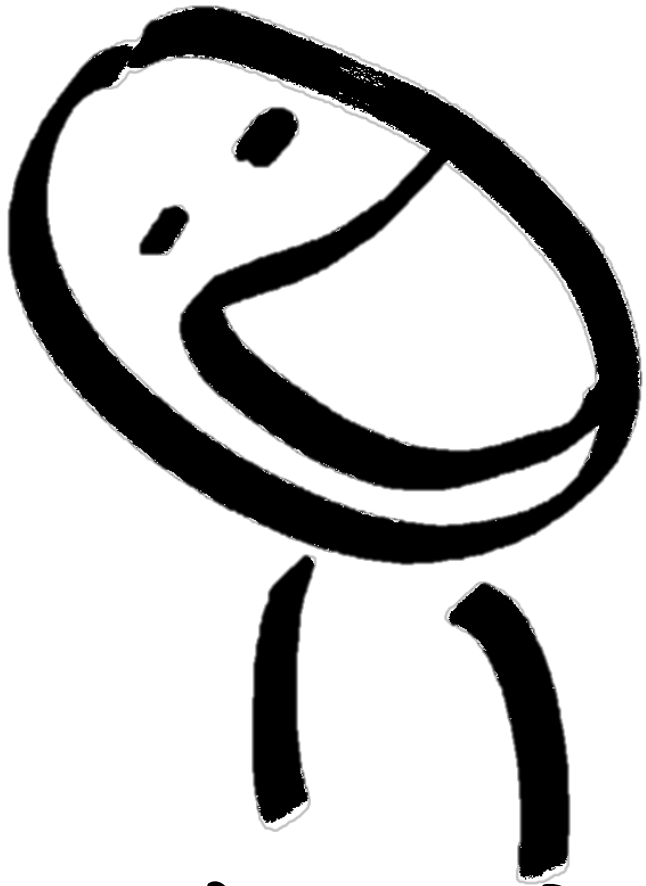
2018.07.15 Tokyo.R #71
Landscape with R – the Japanese R community

2018.10.20 Tokyo.R #73
BeginneR Session – Visualization & Plot

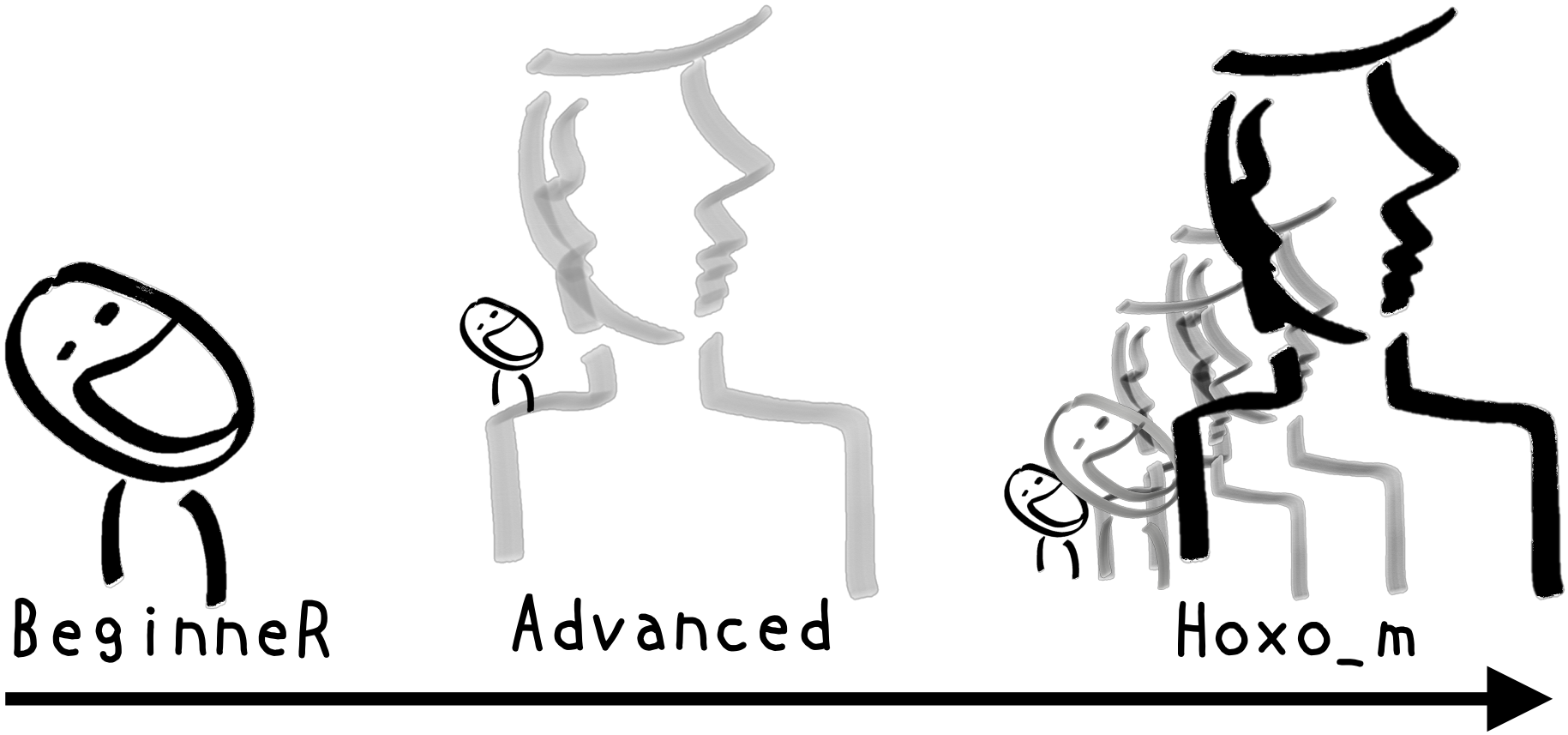
2019.01.19 Tokyo.R #75
BeginneR Session – Data pipeline

2019.03.02 Tokyo.R #76
BeginneR Session – Data pipeline

Today



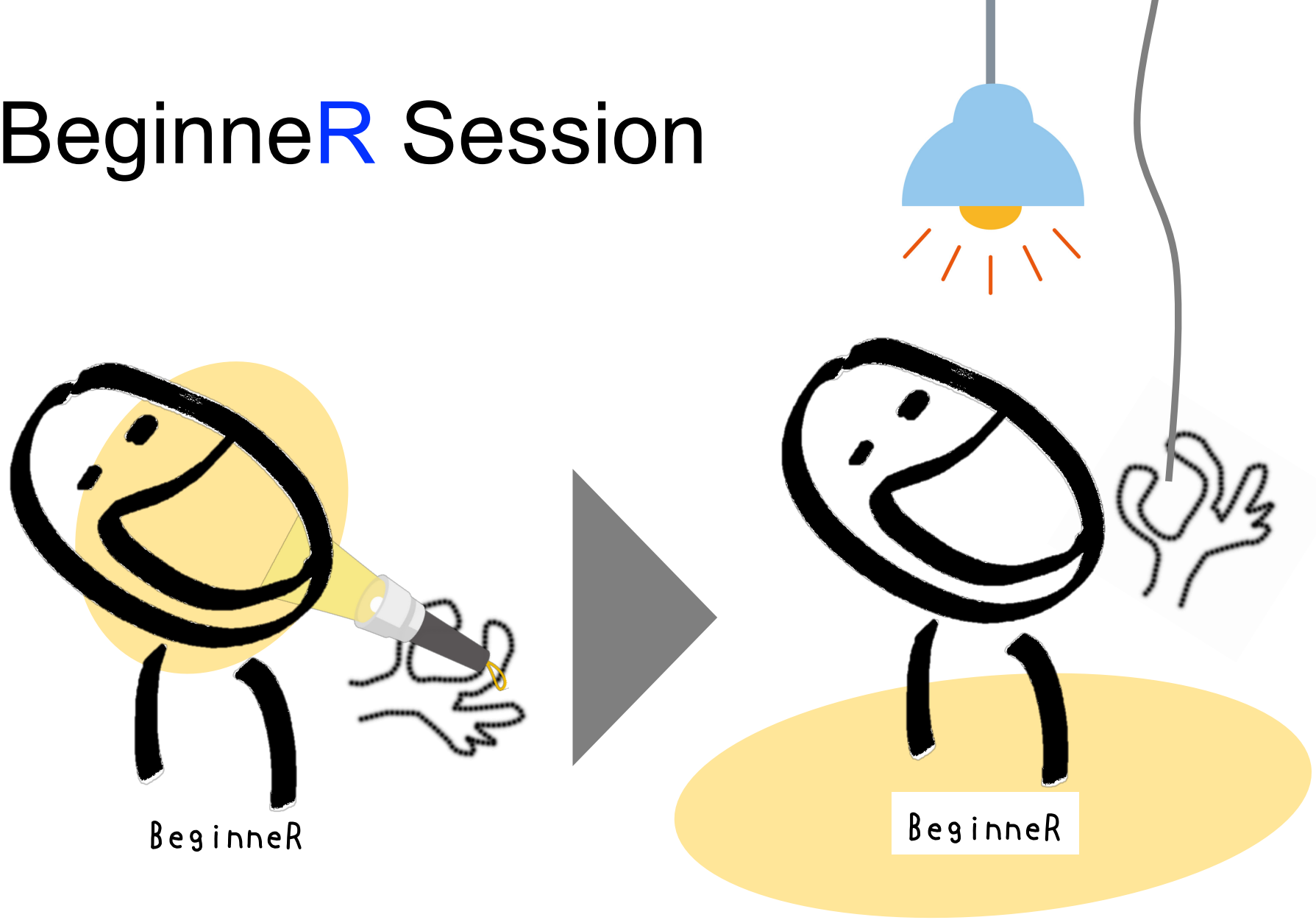
BeginneR



If I have seen further it is by standing on the shoulders of Giants.

-- Sir Isaac Newton, 1676

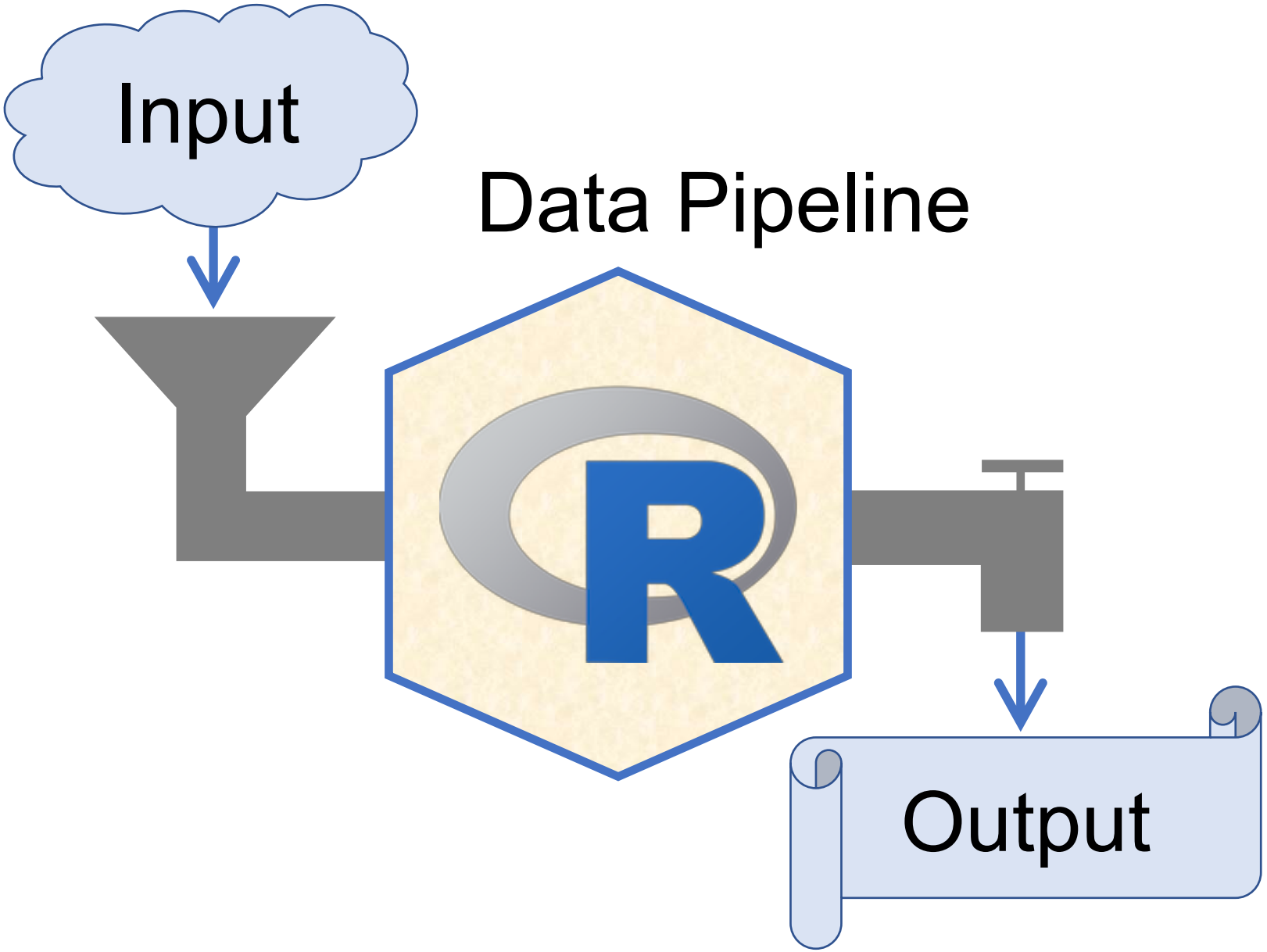
BeginneR Session



Before

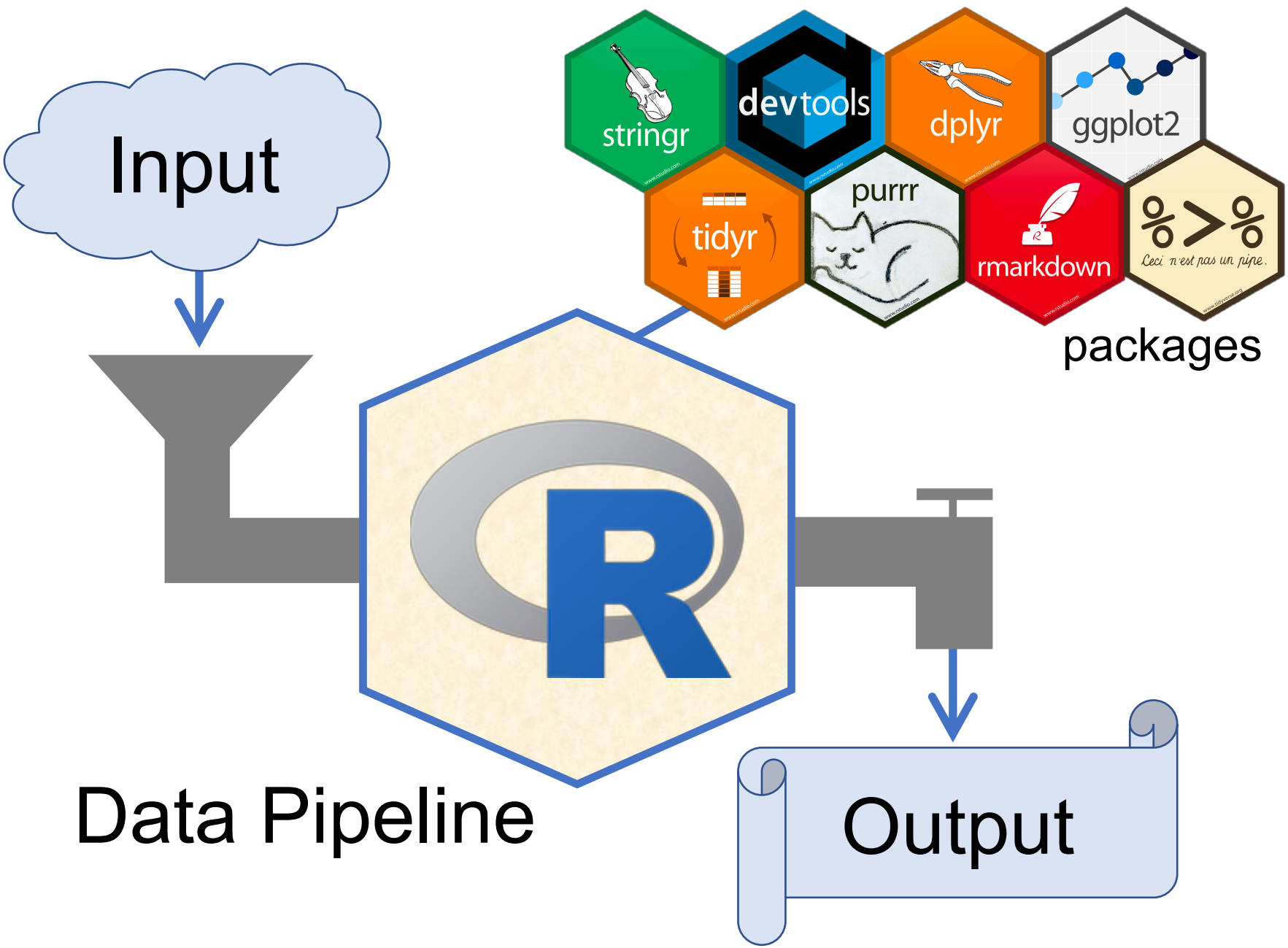
After

BeginneR Session
- Data Pipeline -


Input
Data Pipeline
R
Output

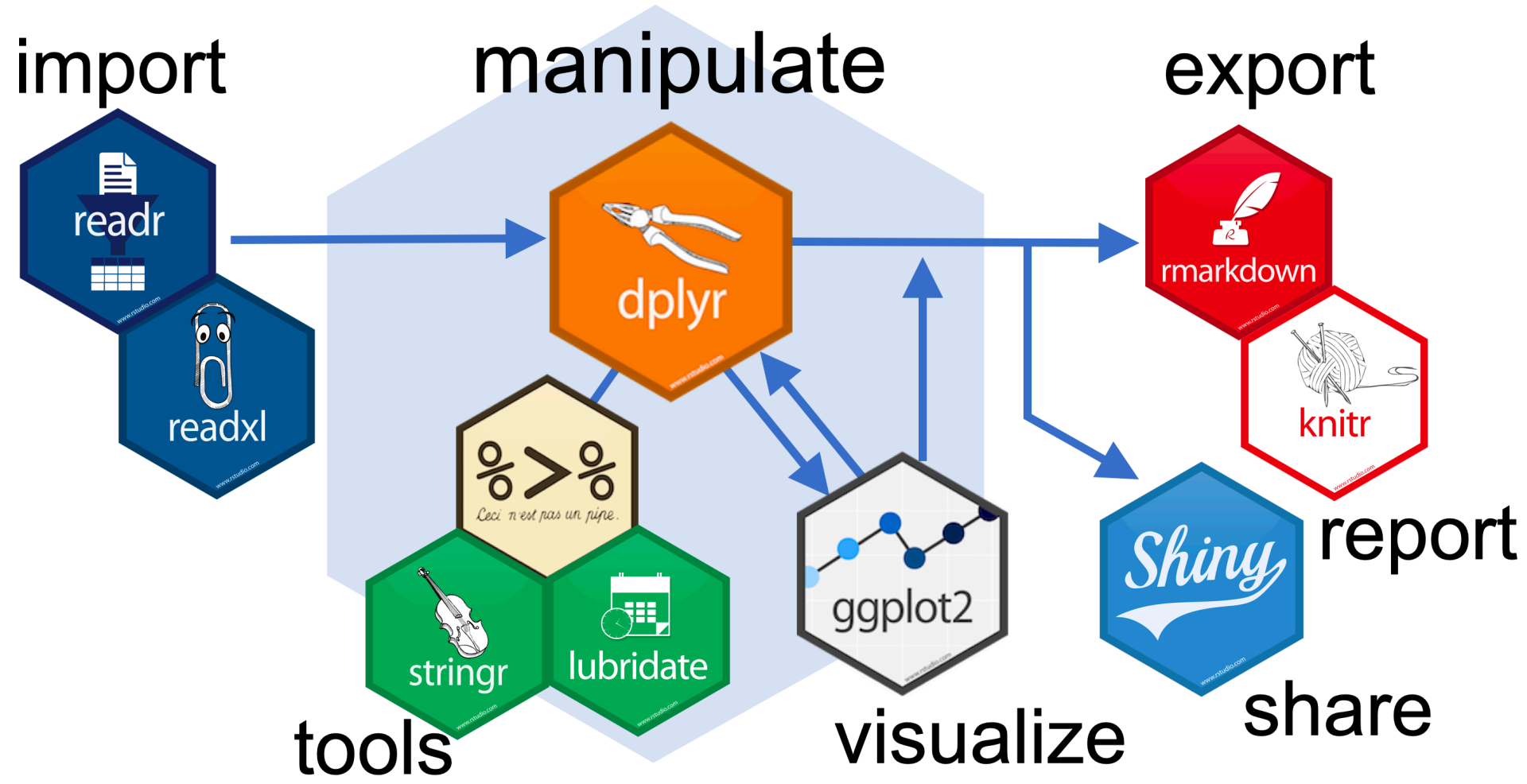
you

packages


Input
stringr
devtools
dplyr
ggplot2
tidyr
purrr
rmarkdown
%>%
Leci n'est pas un pipe.
packages
R
Data Pipeline
Output

Tidyverse

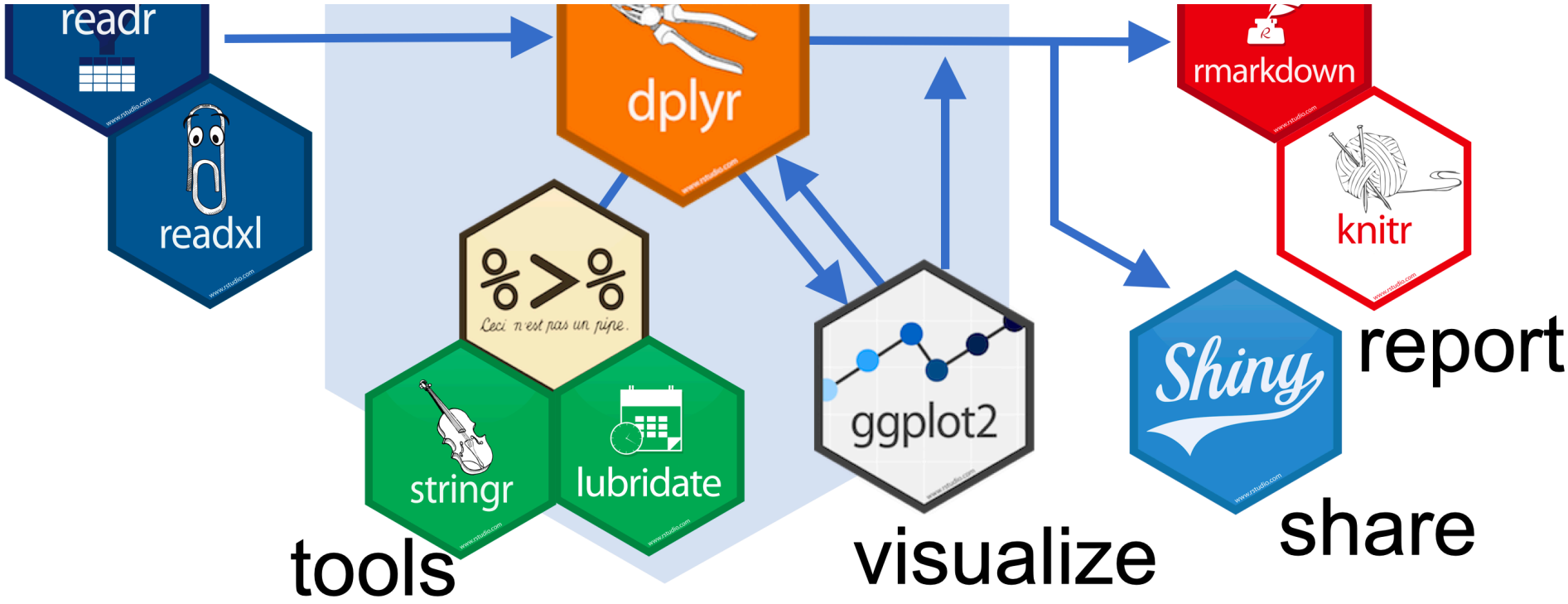
https://www.tidyverse.org/

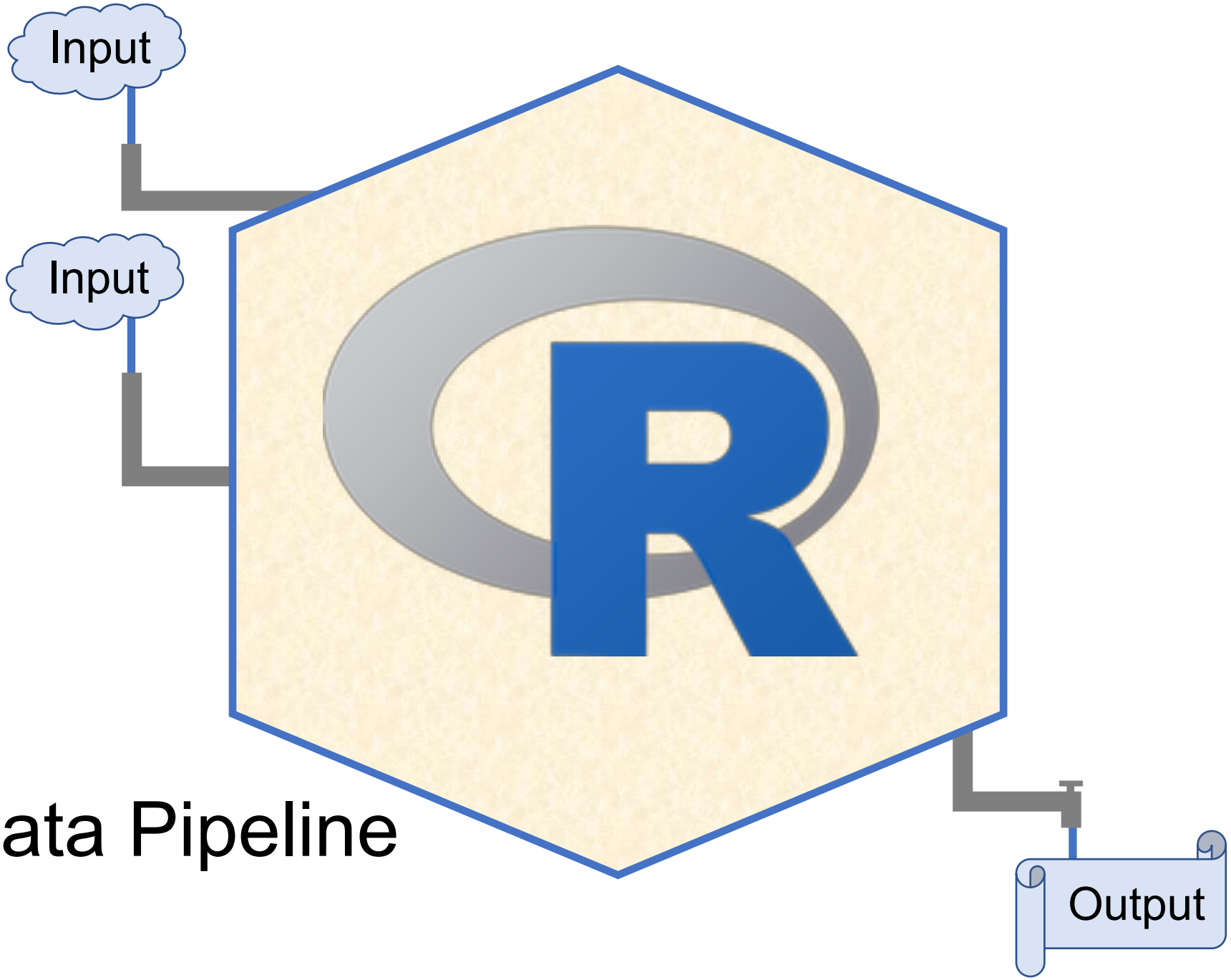


Tidyverse

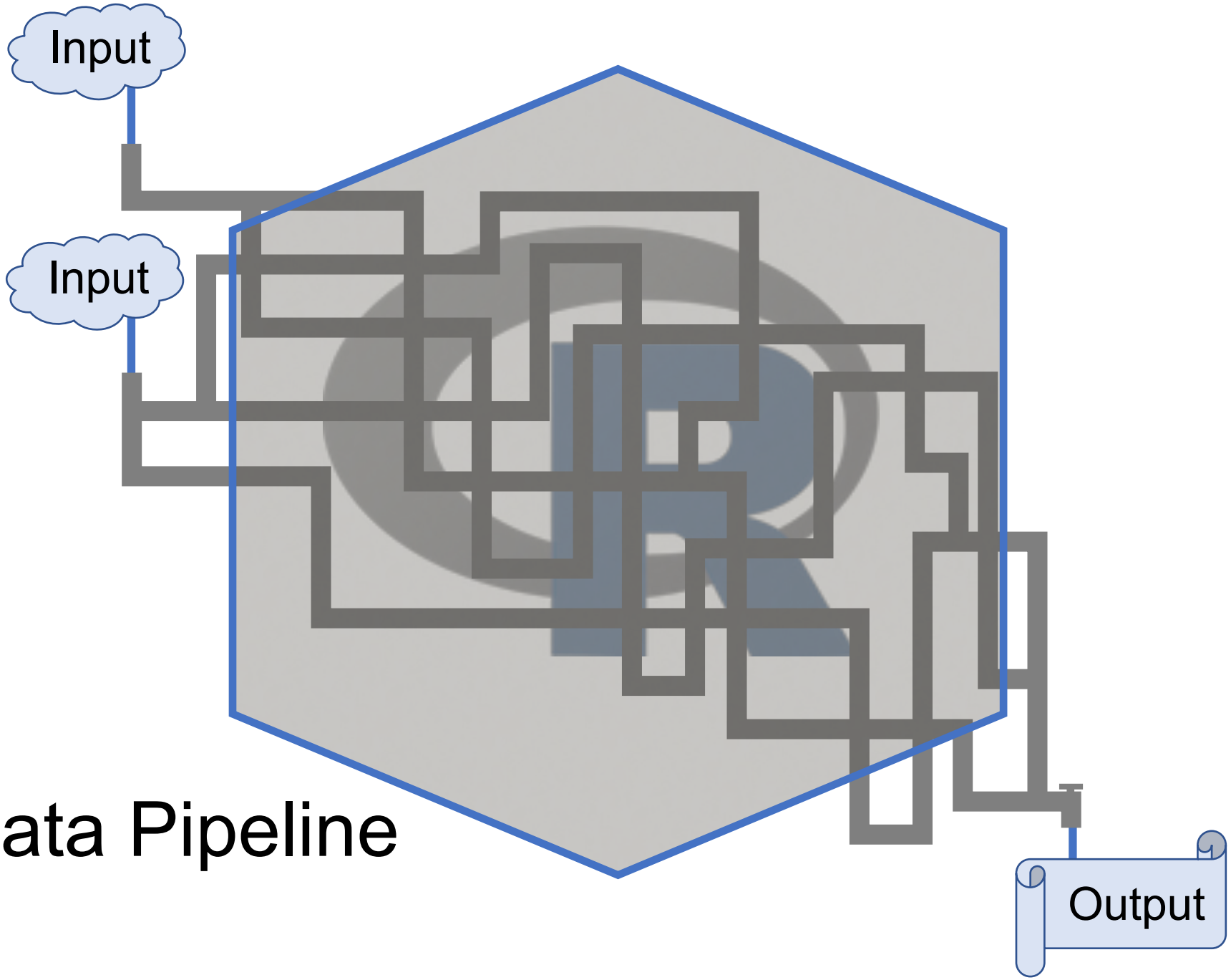
https://www.tidyverse.org/

```
install.packages("tidyverse")
```

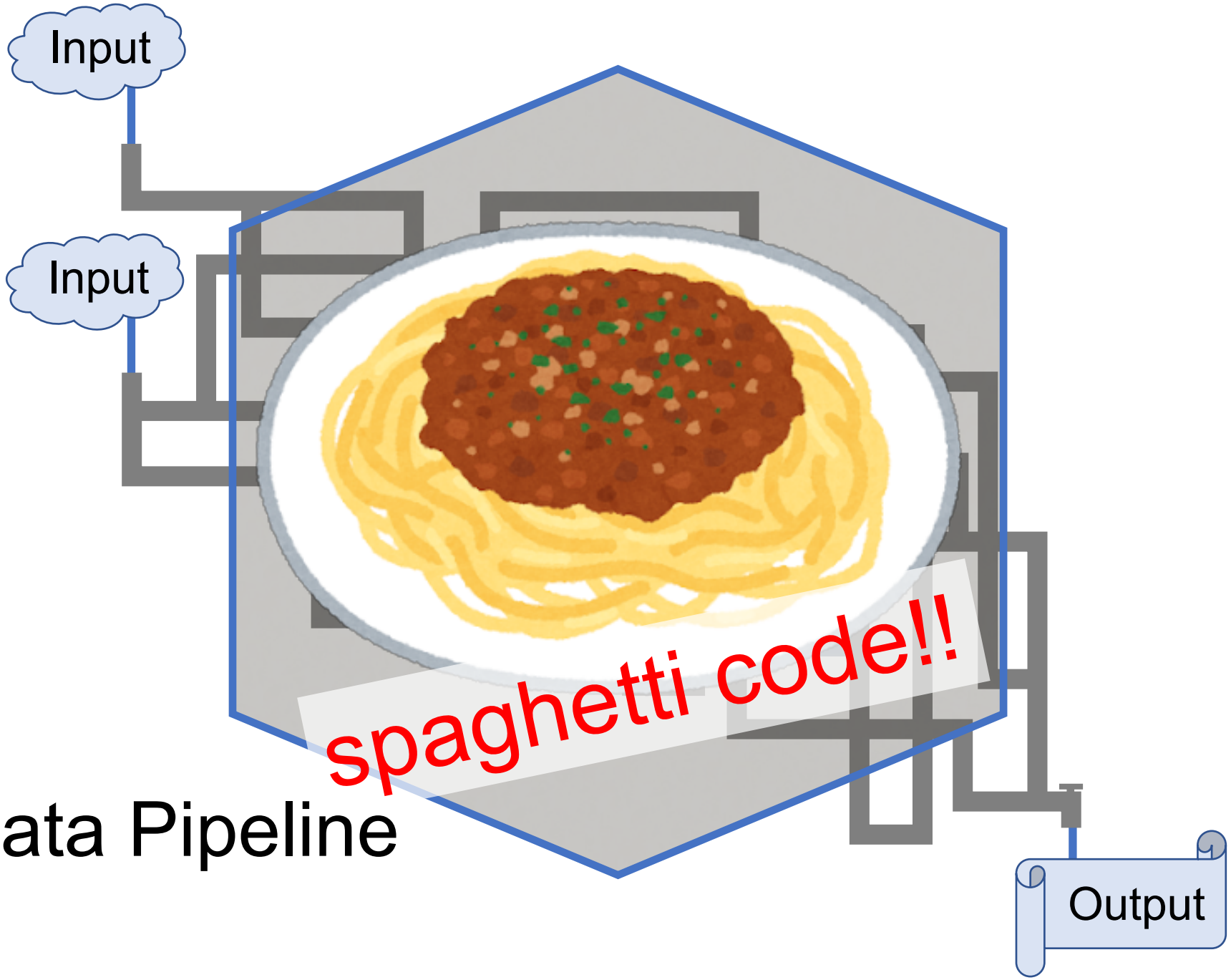




Data Pipeline

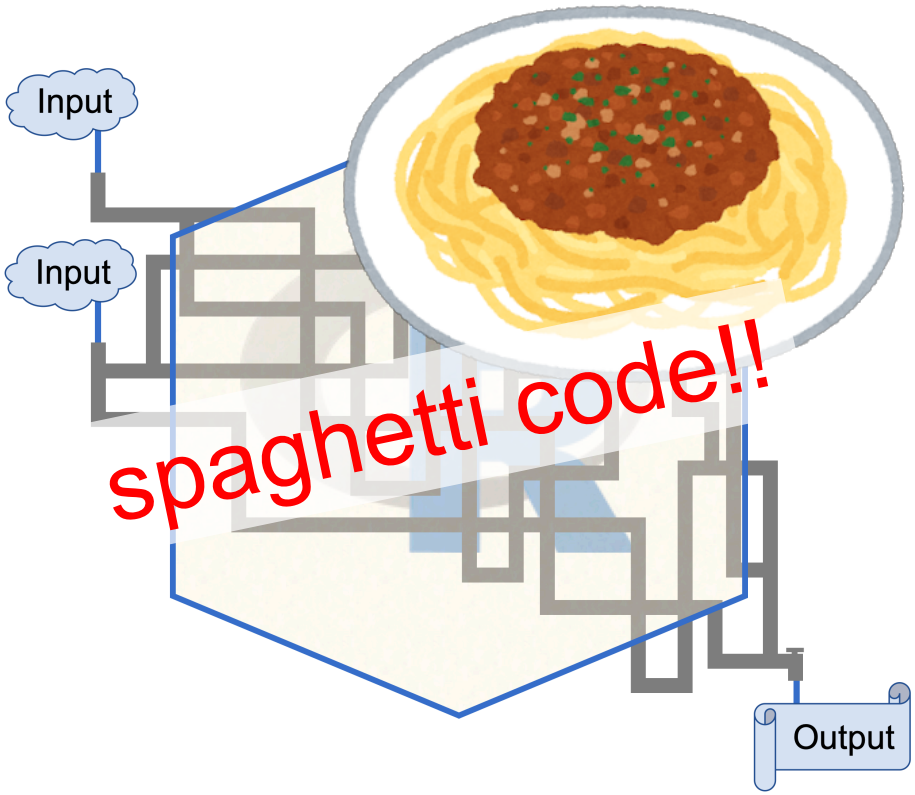


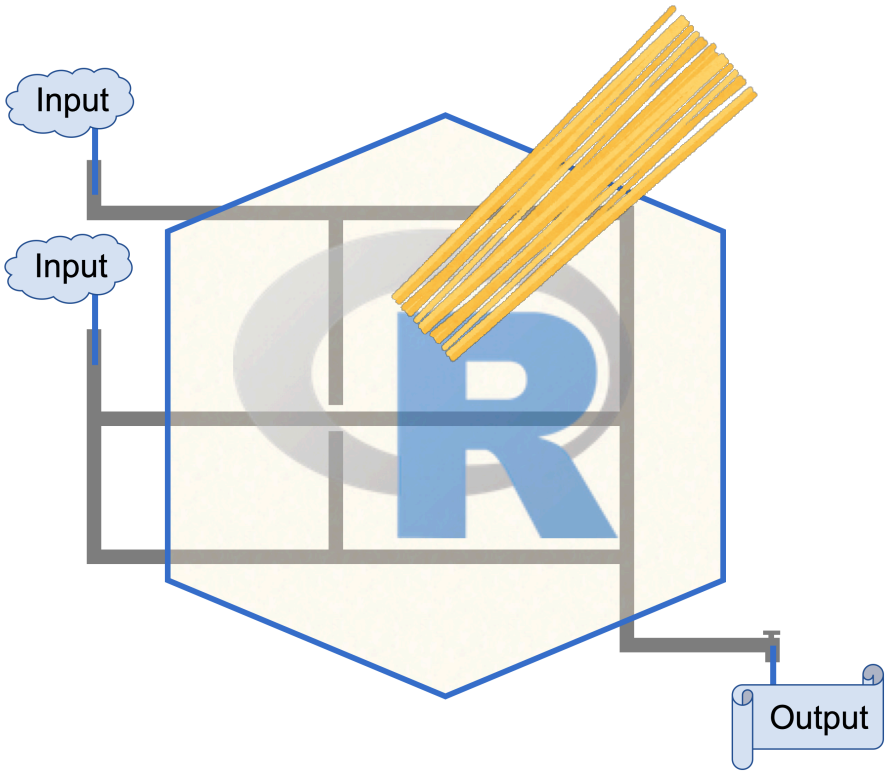
Data Pipeline



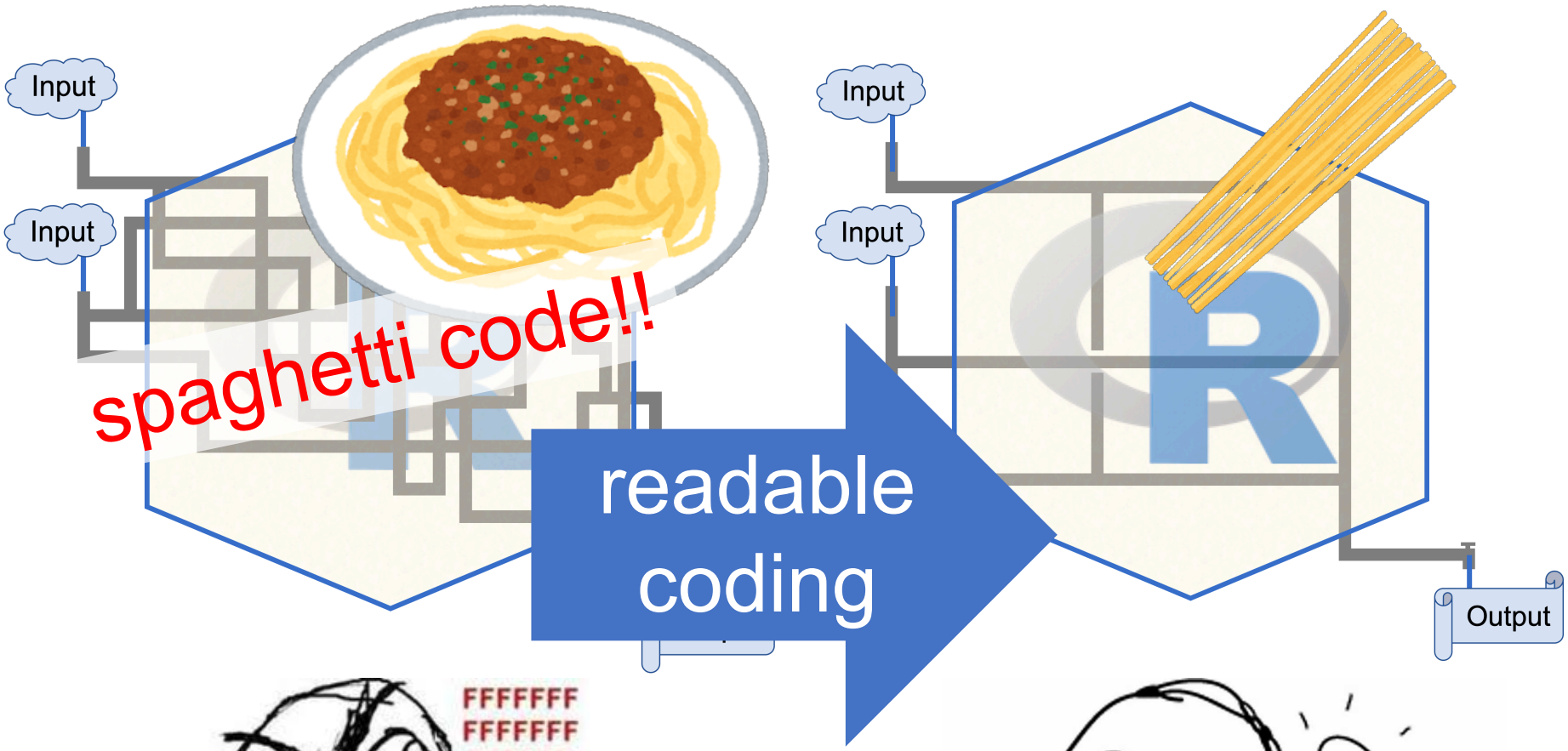
Data Pipeline

Data Pipeline

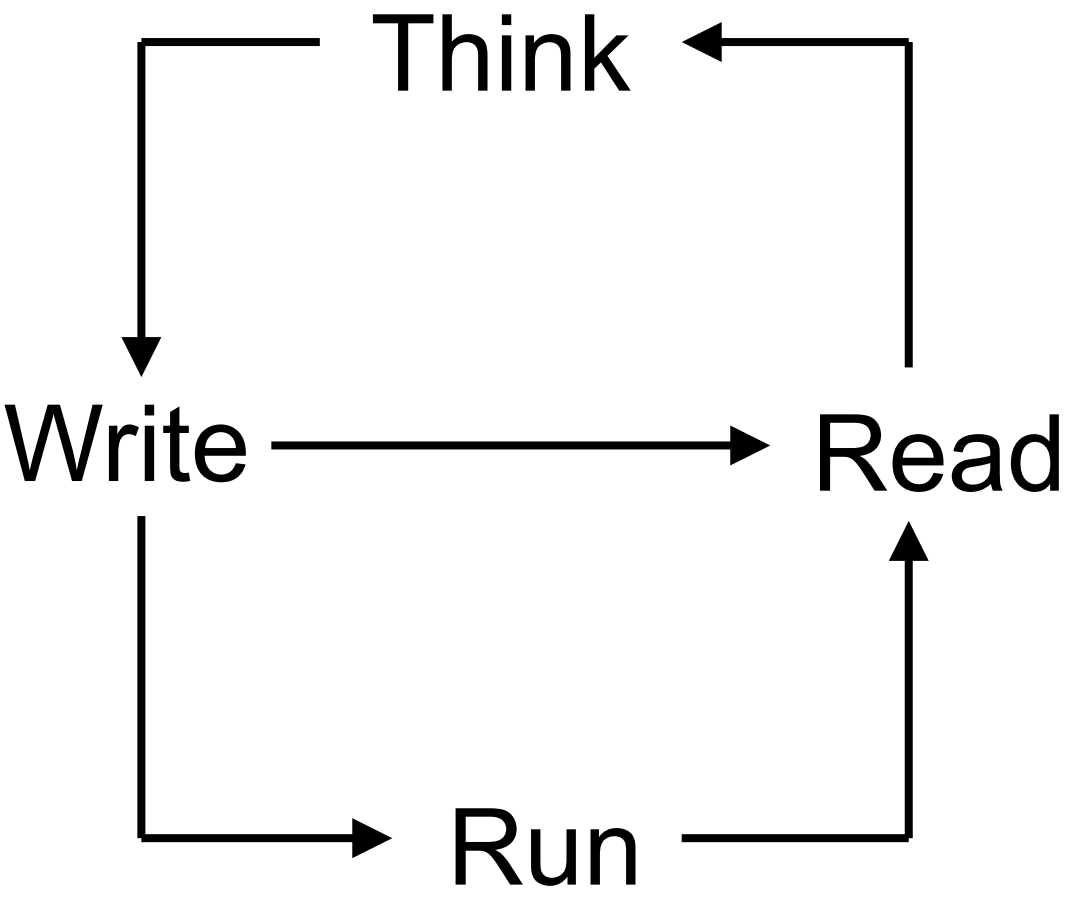




Data Pipeline



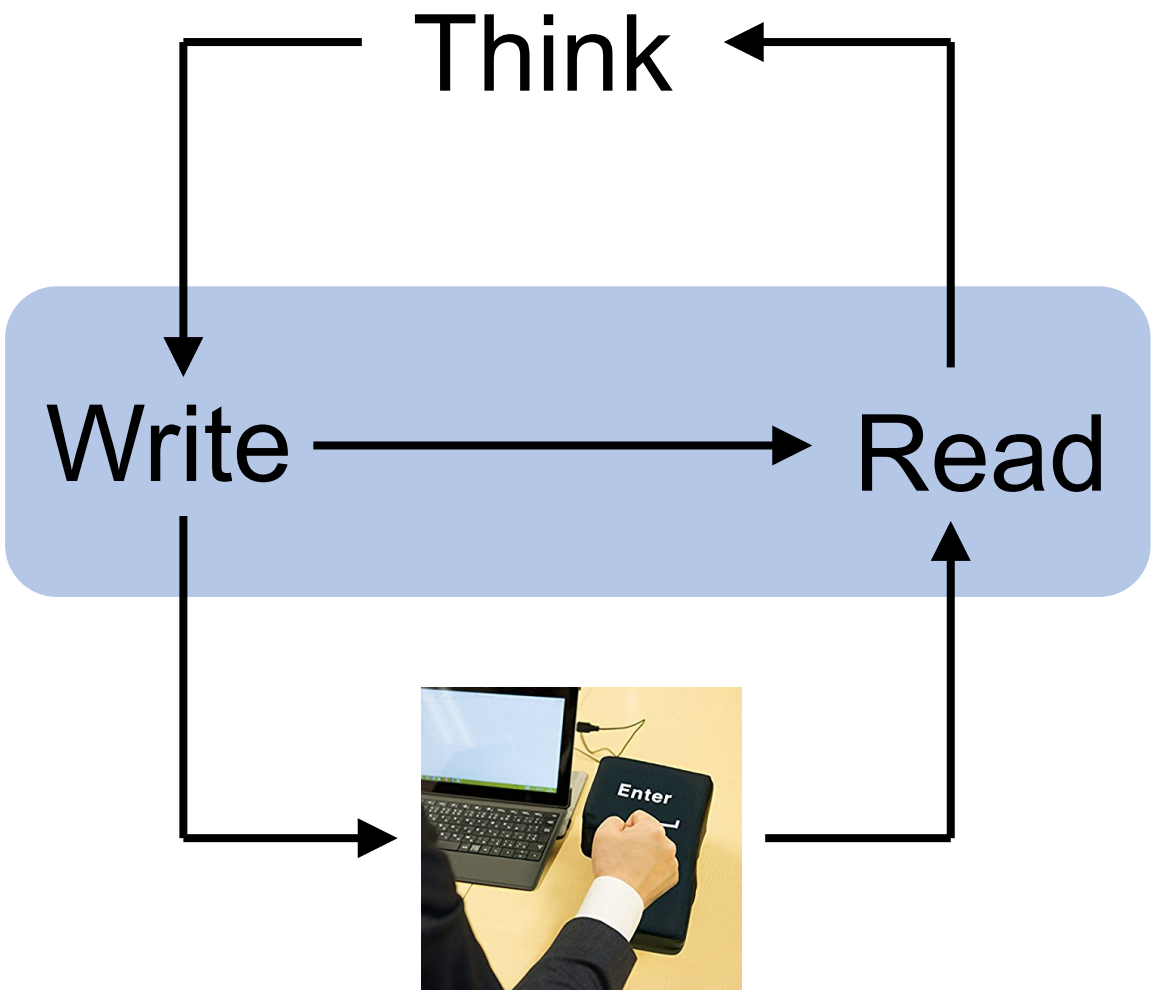
Programing



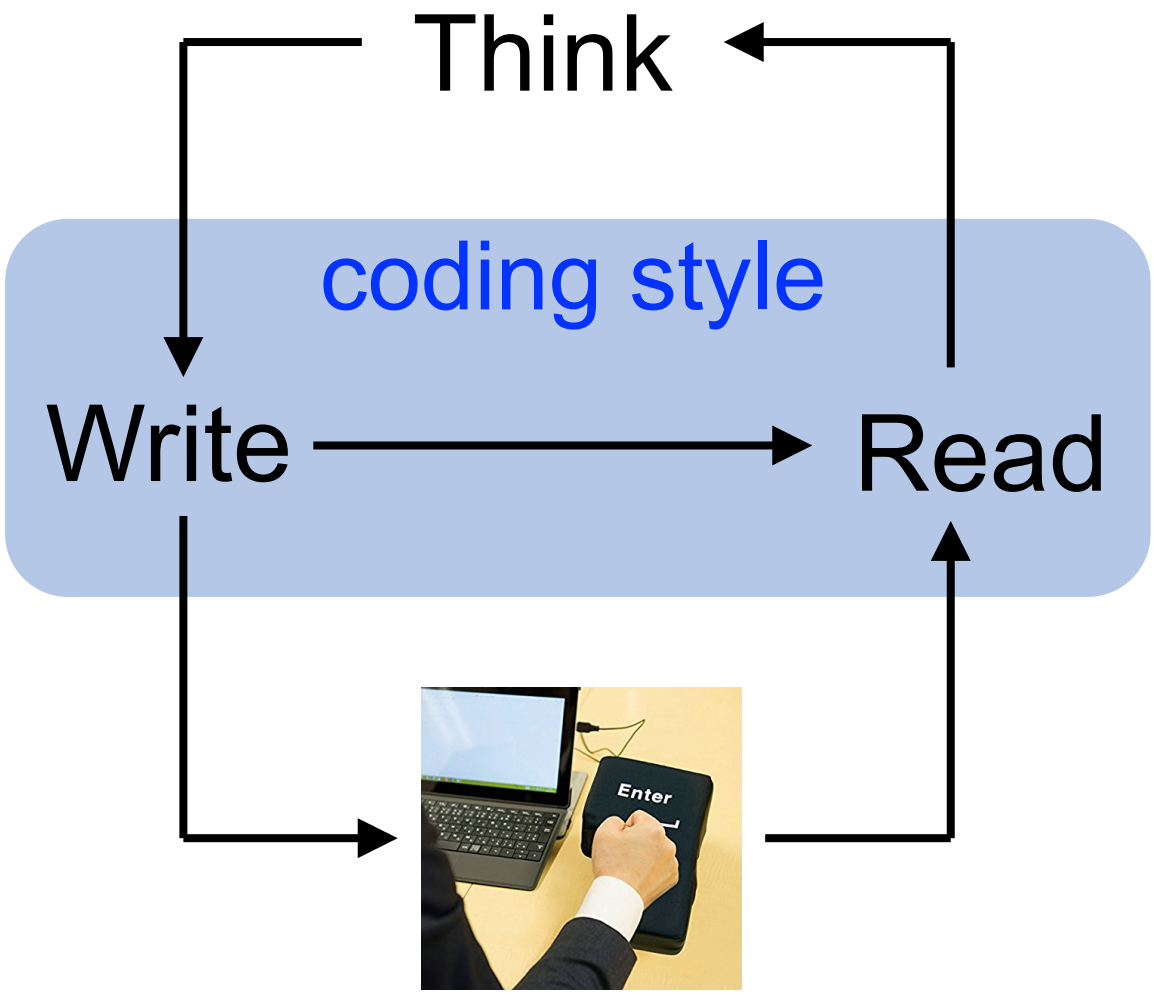
Run!!!

https://www.amazon.co.jp/dp/B00Y0UI990/

Programing



Programing



R coding style guides

The tidyverse style guide

https://style.tidyverse.org/

"Good coding style is like correct punctuation:
you can manage without it, butitsuremakesthingseasiertoread."

Google's R Style Guide

https://style.tidyverse.org/

"The goal of the R Programming Style Guide is to make our
R code easier to read, share, and verify."

The tidyverse style guides

2.1 Object names

> "There are only two hard things in Computer Science: cache invalidation and naming things."
>
> — Phil Karlton

Variable and function names should use only lowercase letters, numbers, and `_`. Use underscores (`_`) (so called snake case) to separate words within a name.

```
# Good
day_one
day_1

# Bad
DayOne
dayone
```

Base R uses dots in function names (`contrib.url()`) and class names (`data.frame`), but it's better to reserve dots exclusively for the S3 object system. In S3, methods are given the name `function.class`; if you also use `.` in function and class names, you end up with confusing methods like `as.data.frame.data.frame()`.

https://style.tidyverse.org/syntax.html#object-names

The tidyverse style guides

Generally, variable names should be nouns and function names should be verbs. Strive for names that are concise and meaningful (this is not easy!).

```
# Good
day_one

# Bad
first_day_of_the_month
djm1
```

Where possible, avoid re-using names of common functions and variables. This will cause confusion for the readers of your code.

```
# Bad
T <- FALSE
c <- 10
mean <- function(x) sum(x)
```

https://style.tidyverse.org/syntax.html#object-names

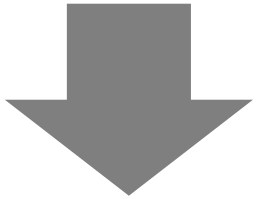
"Where possible, avoid re-using names of common functions and variables. This will cause confusion for the readers of your code."

```
# Good
dat <- read.csv("hoge.csv")

# Bad
data <- read.csv("hoge.csv")
```

```
> data
function(..., list = character(), package = NULL,
    lib.loc = NULL, verbose = getOption("verbose"),
    envir = .GlobalEnv) {
    fileExt <- function(x) {
        db <- grepl("¥¥.[^.]+¥¥.(gz|bz2|xz)$", x)
        ans <- sub(".*¥¥.", "", x)
    ...
```

Auto-indentation (in RStudio)

```
# Bad
for(i in 1:10){
print(i)
}
```



copy (cut) & paste

```
# Good
for(i in 1:10){
  print(i)
}
```

Details: RStudio > Preference > Code > Editing

R coding style guides

The tidyverse style guide

https://style.tidyverse.org/

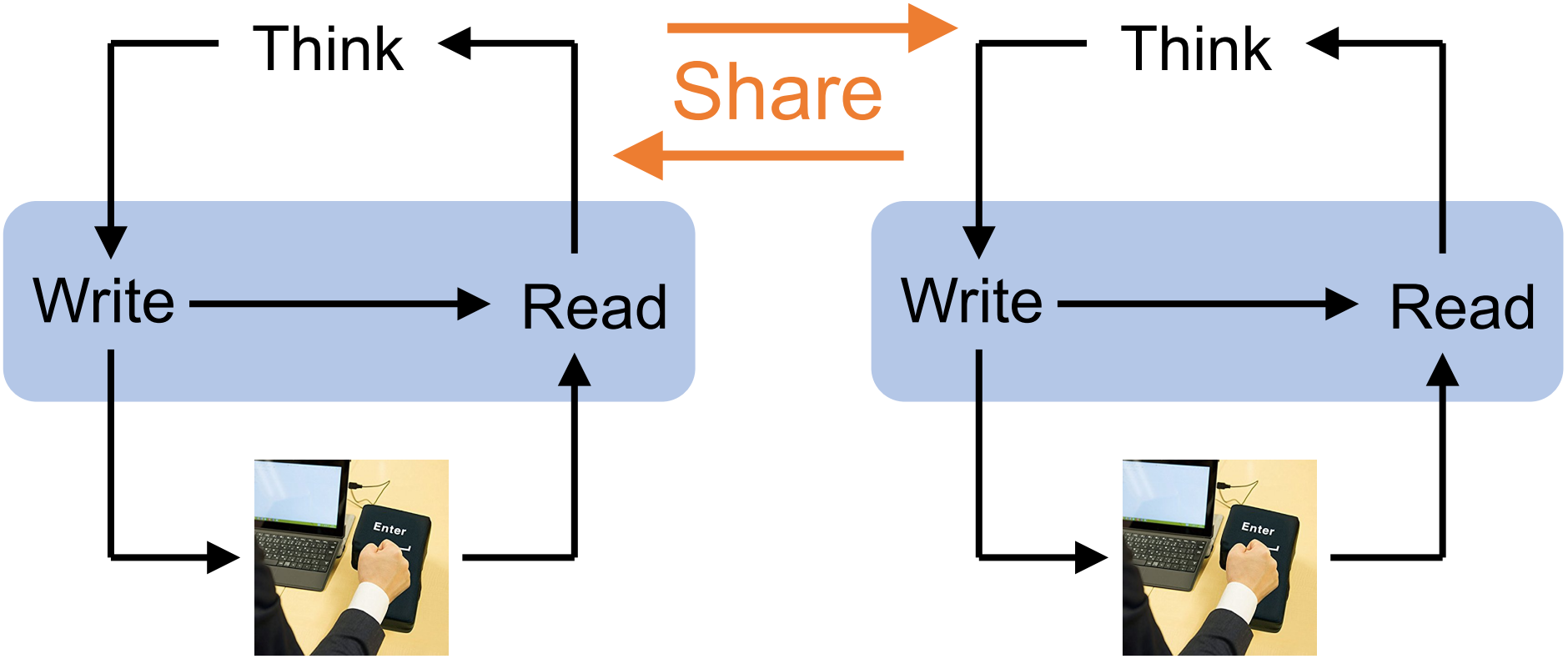
"Good coding style is like correct punctuation:
you can manage without it, butitsuremakesthingseasiertoread."

Google's R Style Guide

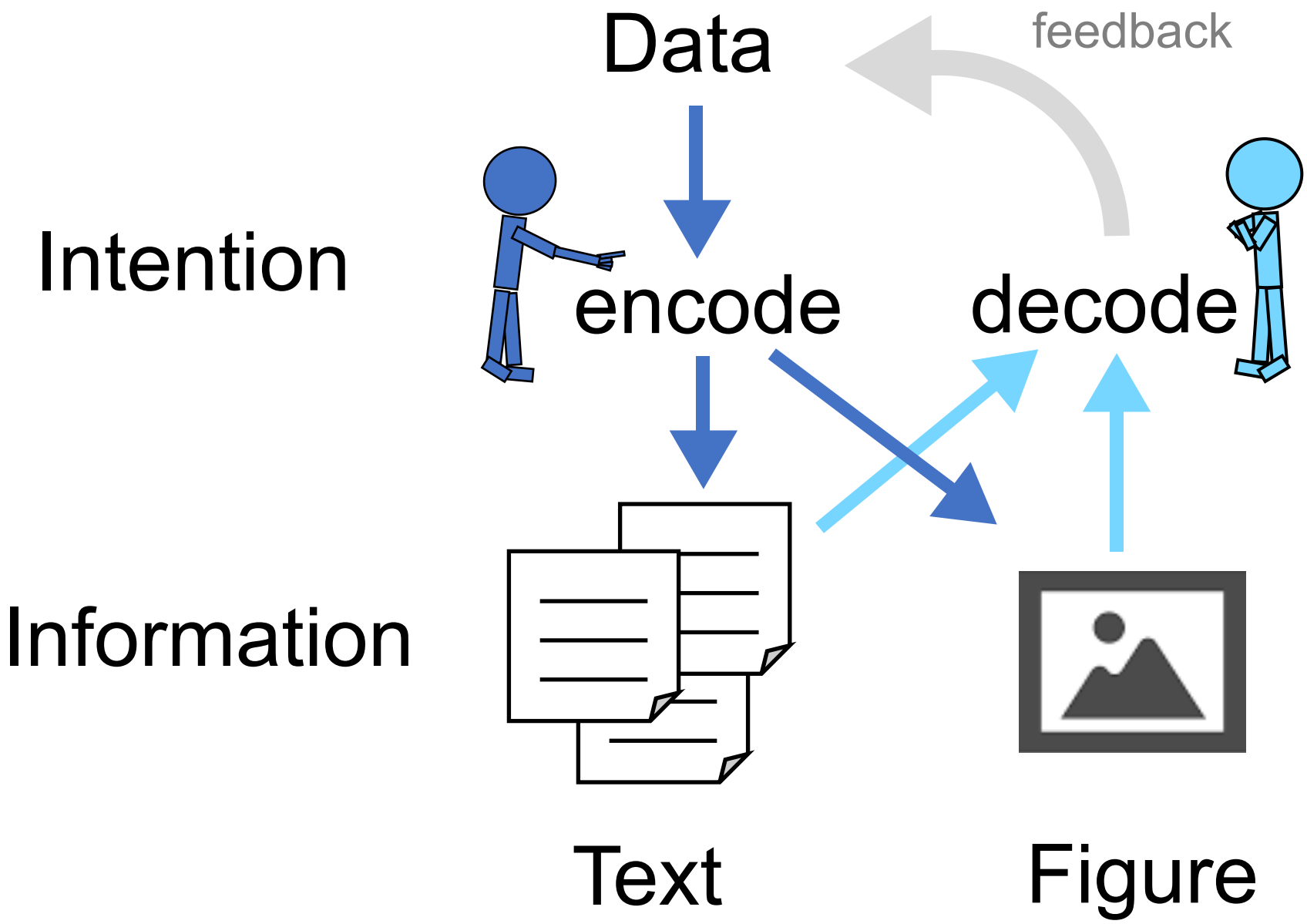
https://style.tidyverse.org/

"The goal of the R Programming Style Guide is to make our
R code easier to read, **share**, and verify."

Programing



Programing


Data
feedback
Intention
encode
decode
Information
Text
Figure

ブール演算子 Boolean Algebra

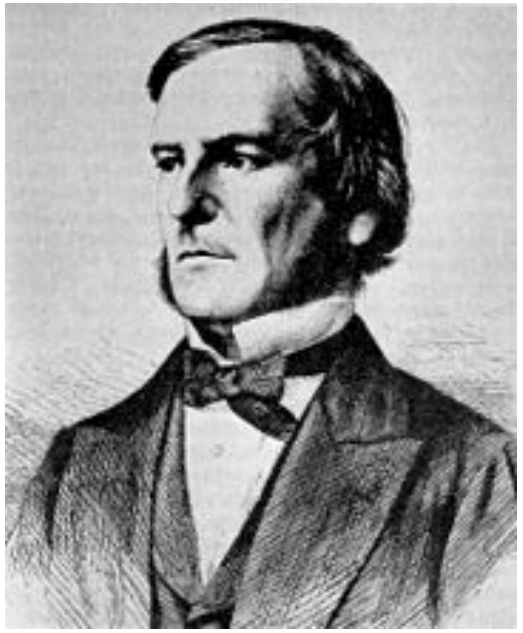
```
# equal to
A == B
```

```
# not equal to
A != B
```

```
# or
A | B
```

```
# and
A & B
```

```
# is A in B?
A %in% B
```



George Boole
1815 - 1864

wikipedia

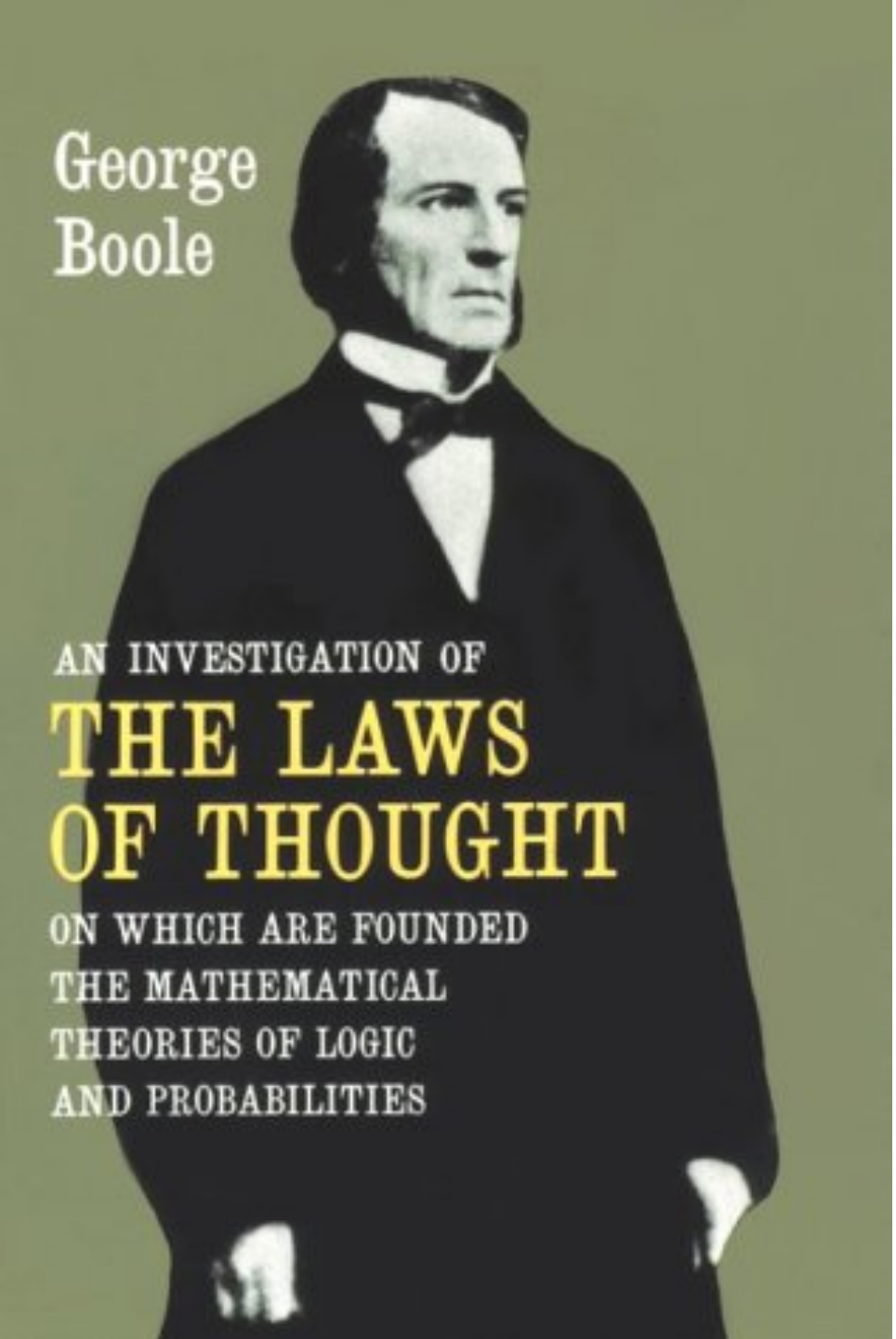
ブール演算子 Boolean Algebra

```
# is A in B?
"a" != "b"
```

```
[1] TRUE
```

```
# is A in B?
1 %in% 10:100
```

```
[1] FALSE
```



George Boole
1815 - 1864

Mathematician

&

Philosopher

A Class-Room Introduction to Logic
https://niyamaklogic.wordpress.com/category/laws-of-thoughts/

ブール演算子 Boolean Algebra

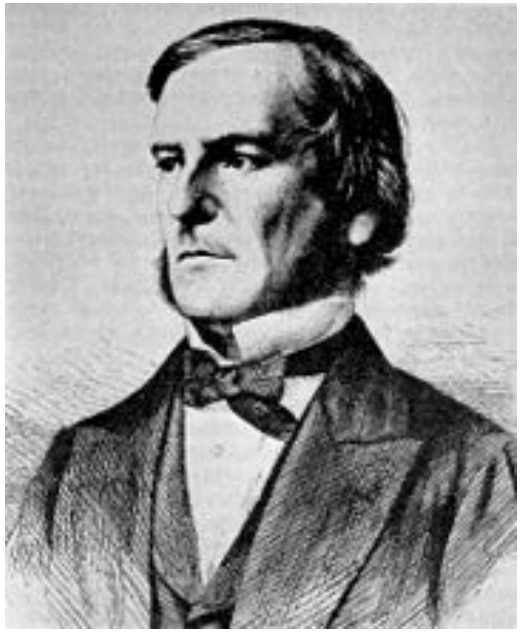
```
# equal to
A == B
```

```
# not equal to
A != B
```

```
# or
A | B
```

```
# and
A & B
```

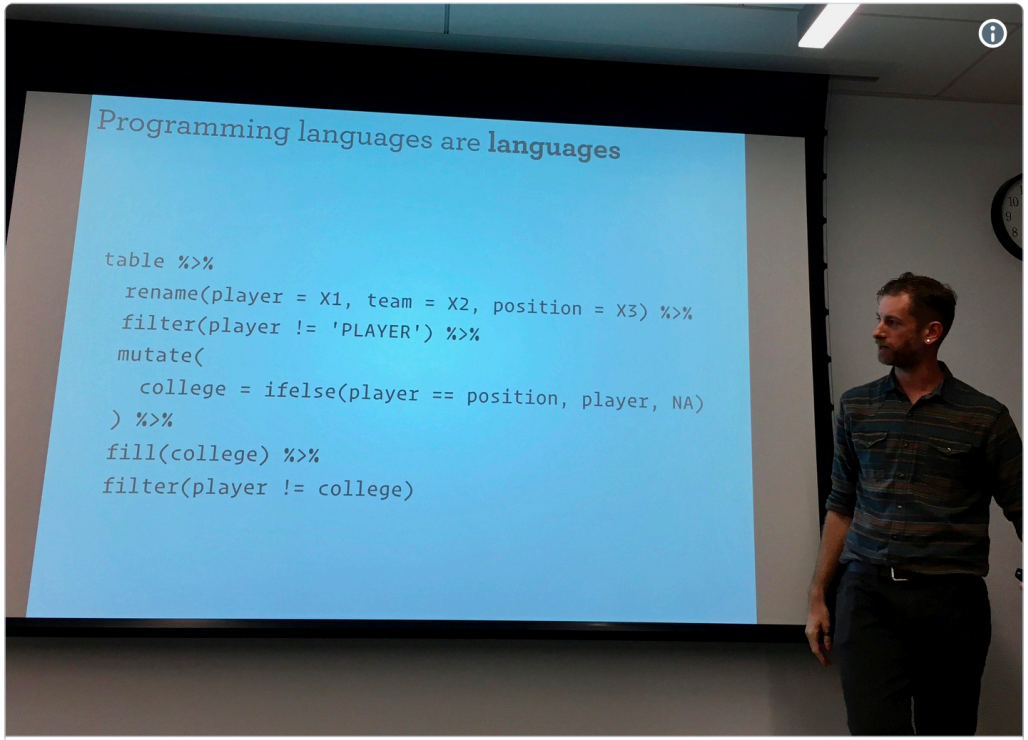
```
# is A in B?
A %in% B
```



George Boole
1815 - 1864

wikipedia

Programing



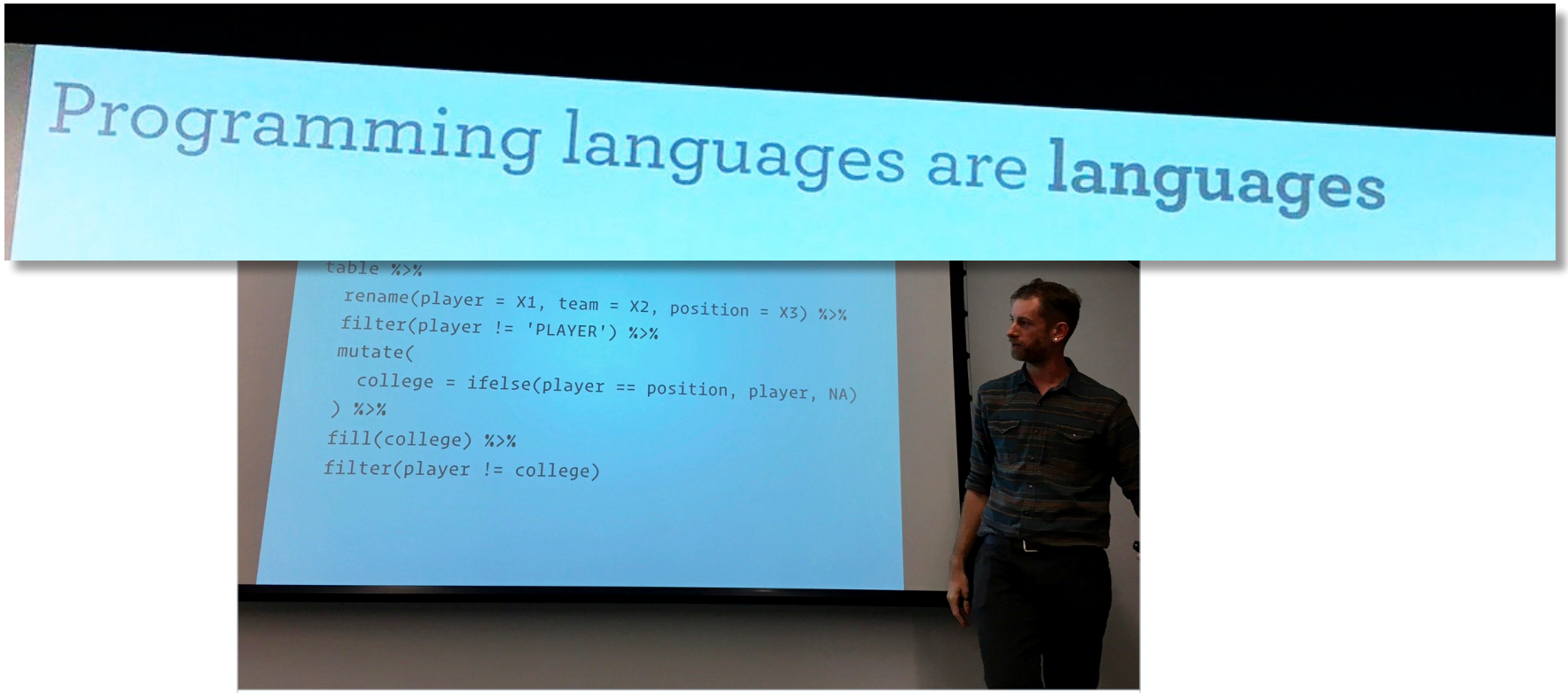
Ramona Walls
@Ramona_Walls

This slide in @hadleywickham's talk at UA today really hit home for me. It doesn't matter if you're good at math! If you can talk an d read, you can probably learn to code.

♡ 23 5:45 AM - Dec 8, 2018

See Ramona Walls's other Tweets

Programing



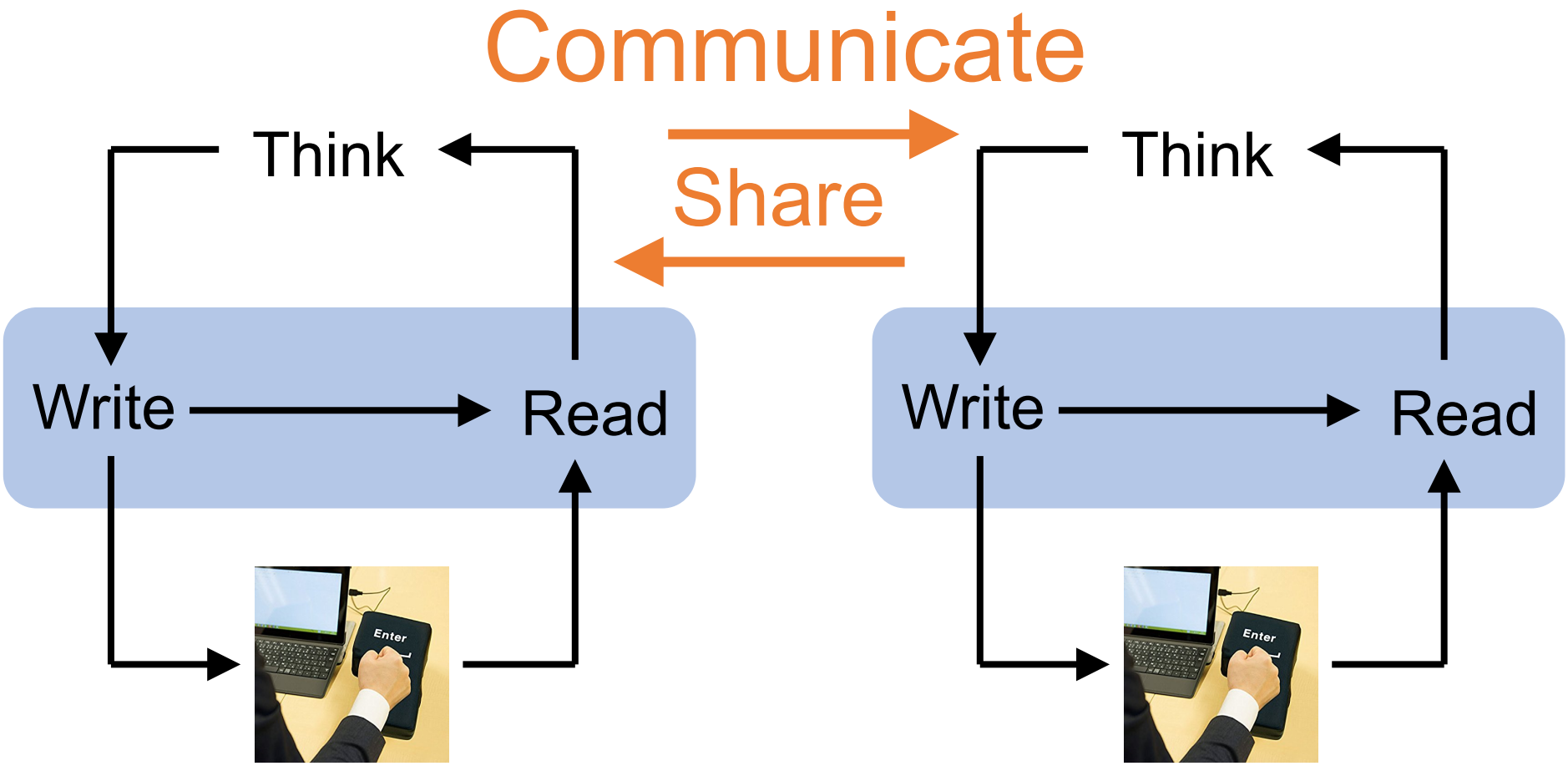
Ramona Walls
@Ramona_Walls

This slide in @hadleywickham's talk at UA today really hit home for me. It doesn't matter if you're good at math! If you can talk an d read, you can probably learn to code.

♡ 23 5:45 AM - Dec 8, 2018

See Ramona Walls's other Tweets

Programing


Communicate
Think
Share
Think
Write
Read
Write
Read
Enter
Enter

Agenda

済 0. Introduction

1. data.frame

2. Pipe

3. Verbs

4. Tidy data

vector

in Excel

	A	B	C
1		5	
2			
3	pre	post	
4	1	5	= A4 * B$1
5	2	10	= A5 * B$1
6	3	15	= A6 * B$1
7	4	20	= A7 * B$1
8	5	25	= A8 * B$1
9			

vector

in Excel

	A	B	C
1		5	
2			
3	pre	post	
4	1	5	= A4 * B$1
5	2	10	= A5 * B$1
6	3	15	= A6 * B$1
7	4	20	= A7 * B$1
8	5	25	= A8 * B$1
9			

in R

```
pre <- c(1, 2, 3, 4, 5)
post <- pre * 5
```

```
> pre
[1] 1 2 3 4 5

> post
[1]  5 10 15 20 25
```

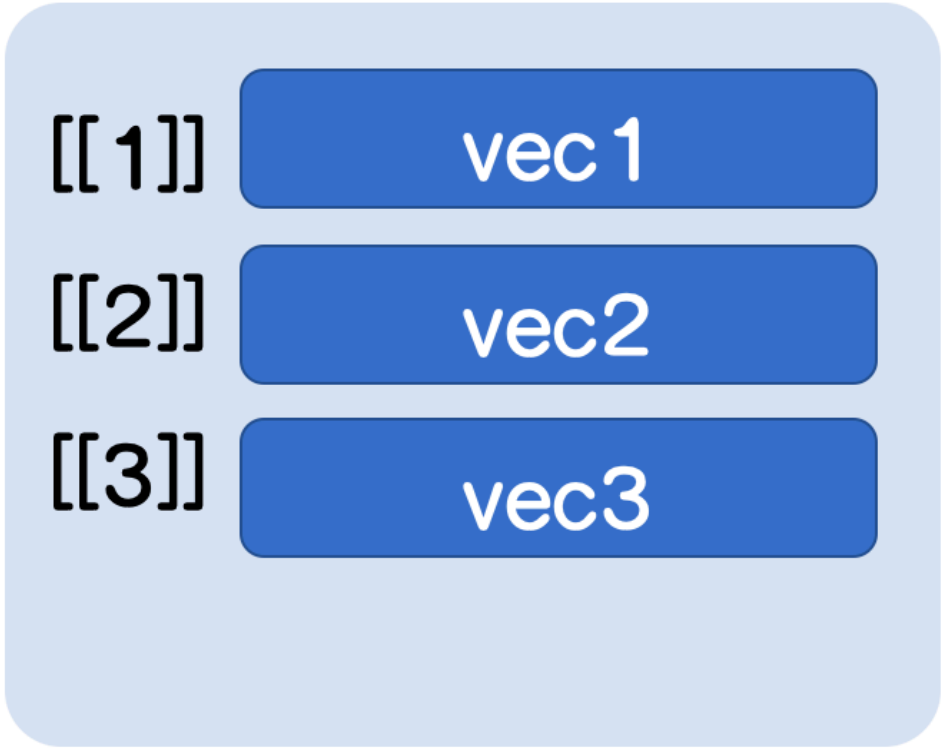
vector

```
pre <- c(1, 2, 3, 4, 5)
post <- pre * 5
```

```
> pre
[1] 1 2 3 4 5
> post
[1]  5 10 15 20 25
> post[c(3, 5)]
[1] 15 25
```

list

```
list(vec1,
     vec2,
     vec3, ...)
```



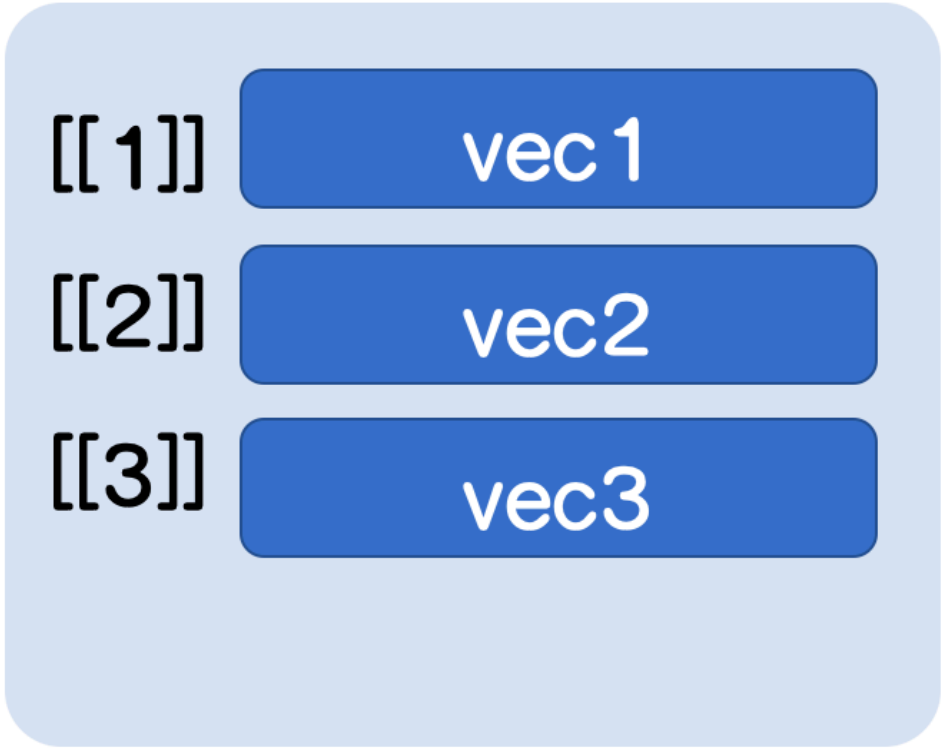
vector

```
pre <- c(1, 2, 3, 4, 5)
post <- pre * 5
```

```
> pre
[1] 1 2 3 4 5
> post
[1]  5 10 15 20 25
> post[c(3, 5)]
[1] 15 25
```

list

```
list(vec1,
     vec2,
     vec3, ...)
```



list

```
list(vec1,
     vec2,
     vec3, ...)
```

named list

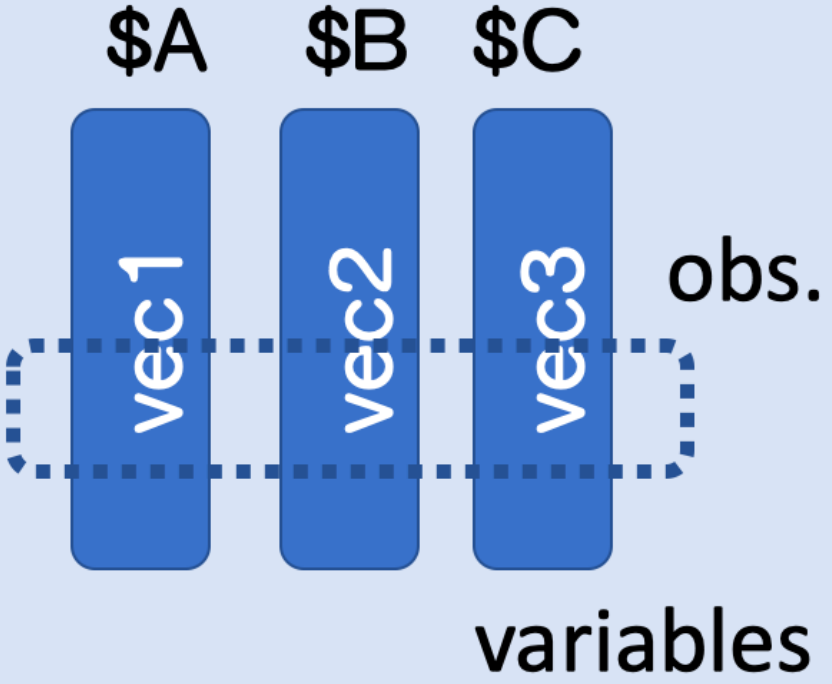
```
list(A = vec1,
     B = vec2,
     C = vec3, ...)
```

named list

```
list(A = vec1,
     B = vec2,
     C = vec3, ...)
```

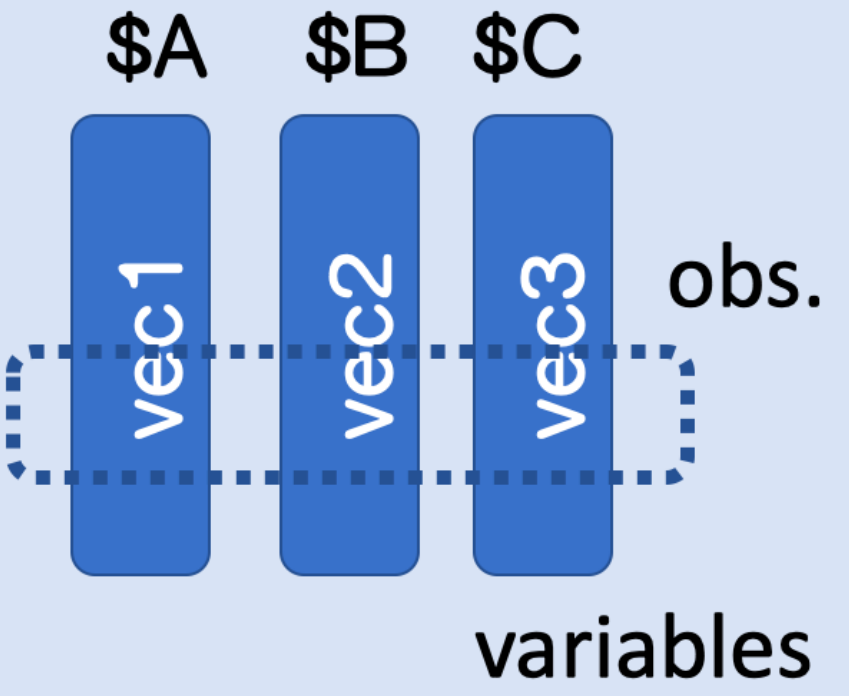
data.frame

```
data.frame(A = vec1,
           B = vec2,
           C = vec3, ...)
```



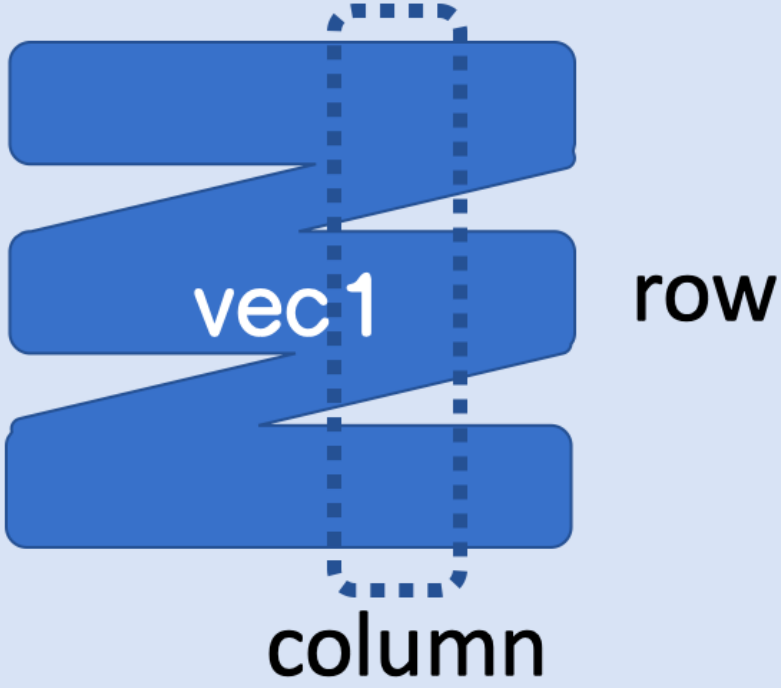
data.frame

```
data.frame(A = vec1,
           B = vec2,
           C = vec3, ...)
```



matrix

```
matrix(vec1,
       nrow = 3,
       byrow = TRUE, ...)
```

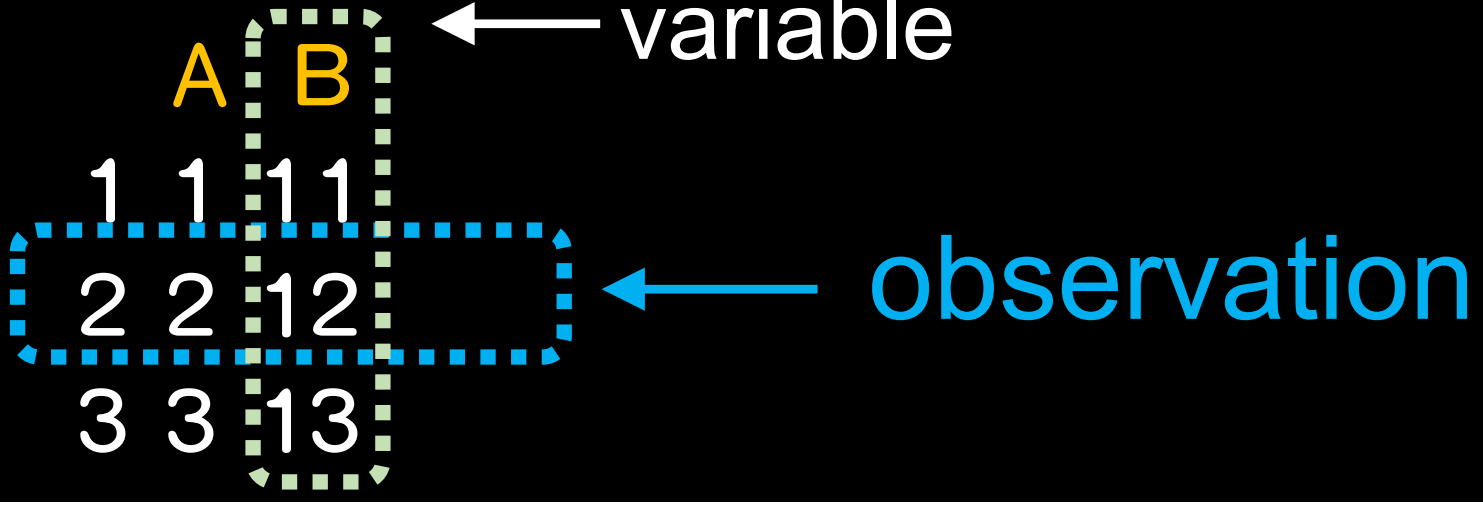


data.frame

```
df1 <- data.frame(A = 1:3, B = 11:13)
```

```
> str(df1)
'data.frame':  3 obs. of  2 variables:
$ A: int  1 2 3
$ B: int  11 12 13
```

```
> df1
  A  B
1 1 11
2 2 12
3 3 13
```



Agenda

済 0. Introduction

済 1. data.frame

2. Tidy data

3. Pipe

4. Verbs

Journal of Statistical Software

August 2014, Volume 59, Issue 10. *http://www.jstatsoft.org/*

Tidy Data

Hadley Wickham
RStudio

http://vita.had.co.nz/papers/tidy-data.html

R for Data Science

Garrett Grolemund

Hadley Wickham

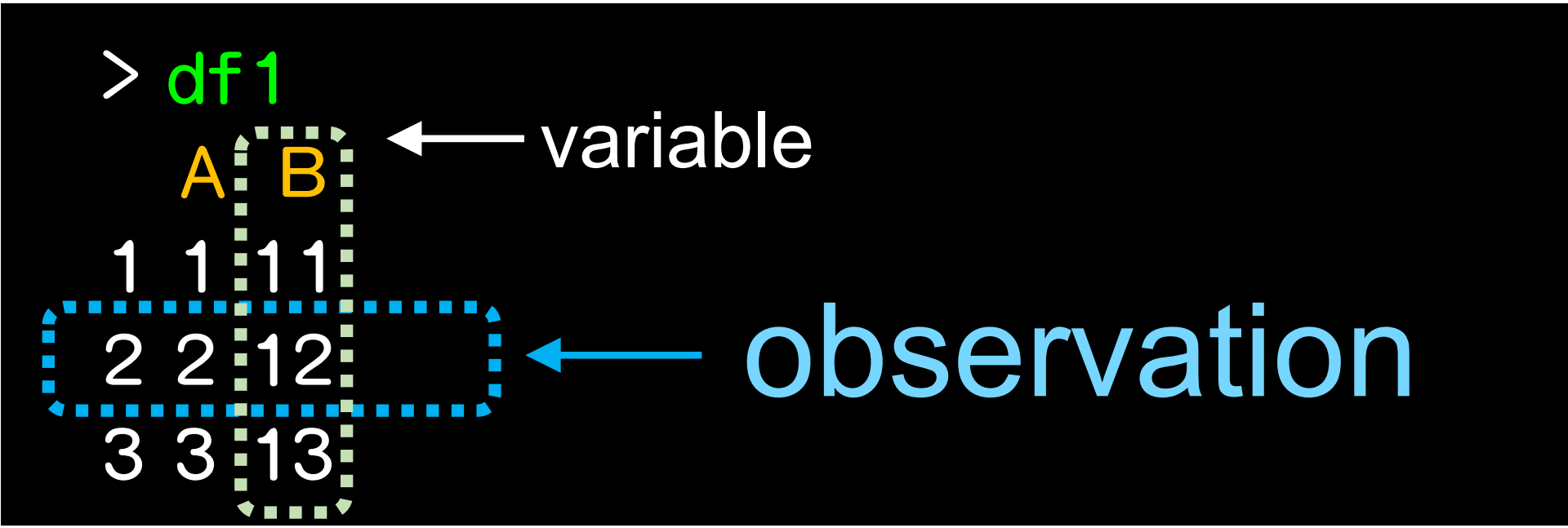
https://r4ds.had.co.nz/

12 Tidy data

In tidy data:

1. Each variable forms a column.
2. Each observation forms a row.
3. Each value must have its own cell.

```
df1 <- data.frame(A = 1:3, B = 11:13)
```



In tidy data:

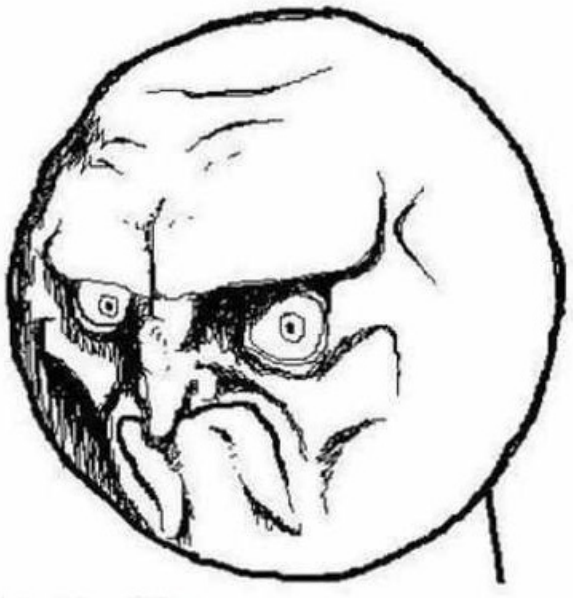
1. Each variable forms a column.
2. Each observation forms a row.
3. Each value must have its own cell.

country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766

In tidy data:

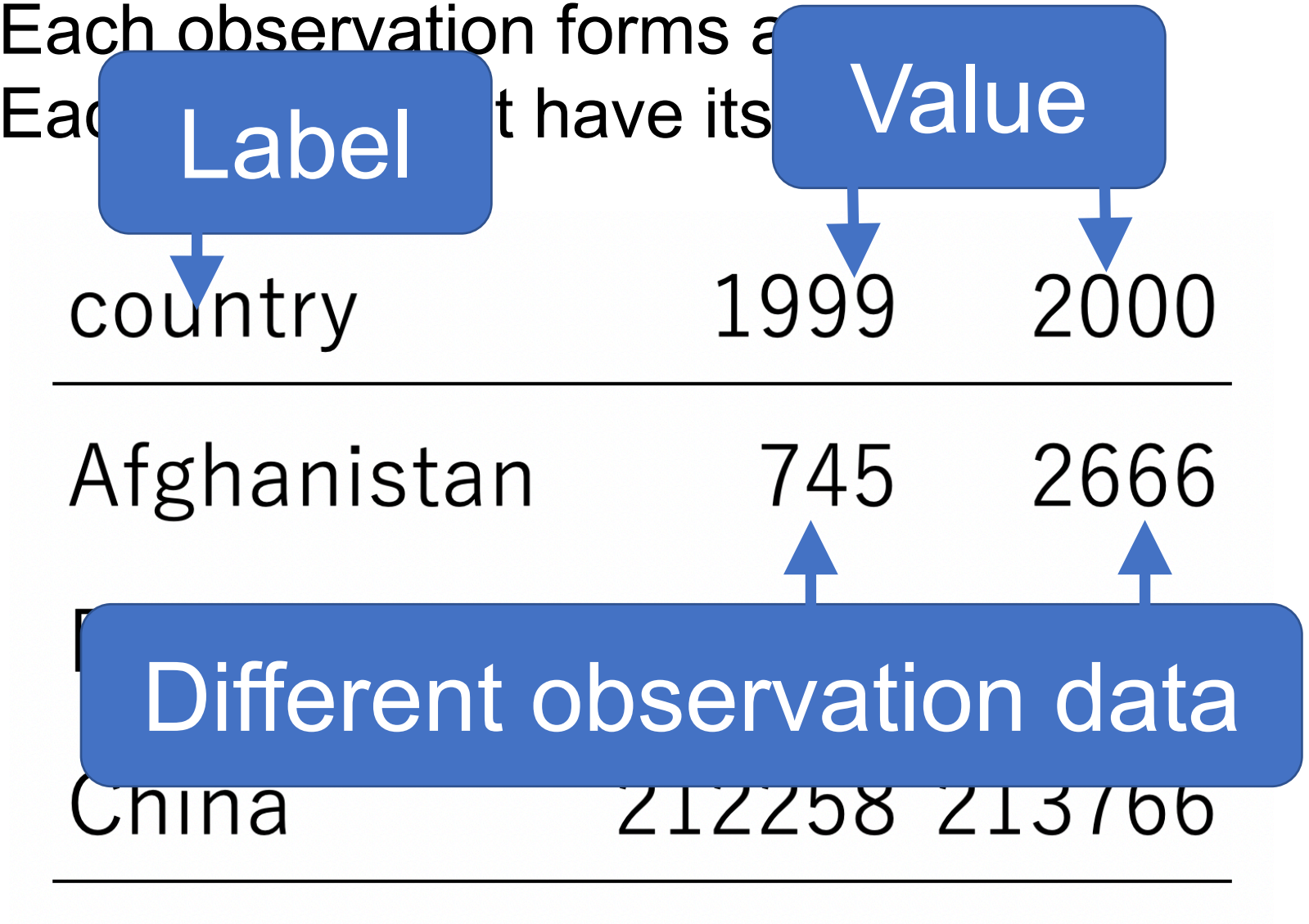
1. Each variable forms a column.
2. Each observation forms a row.
3. Each value must have its own cell.

country	1999	2000
Afghanistan	745	
Brazil	37737	
China	212258	2



In tidy data:

1. Each variable forms a column.
2. Each observation forms a
3. Eac t have its



In tidy data:

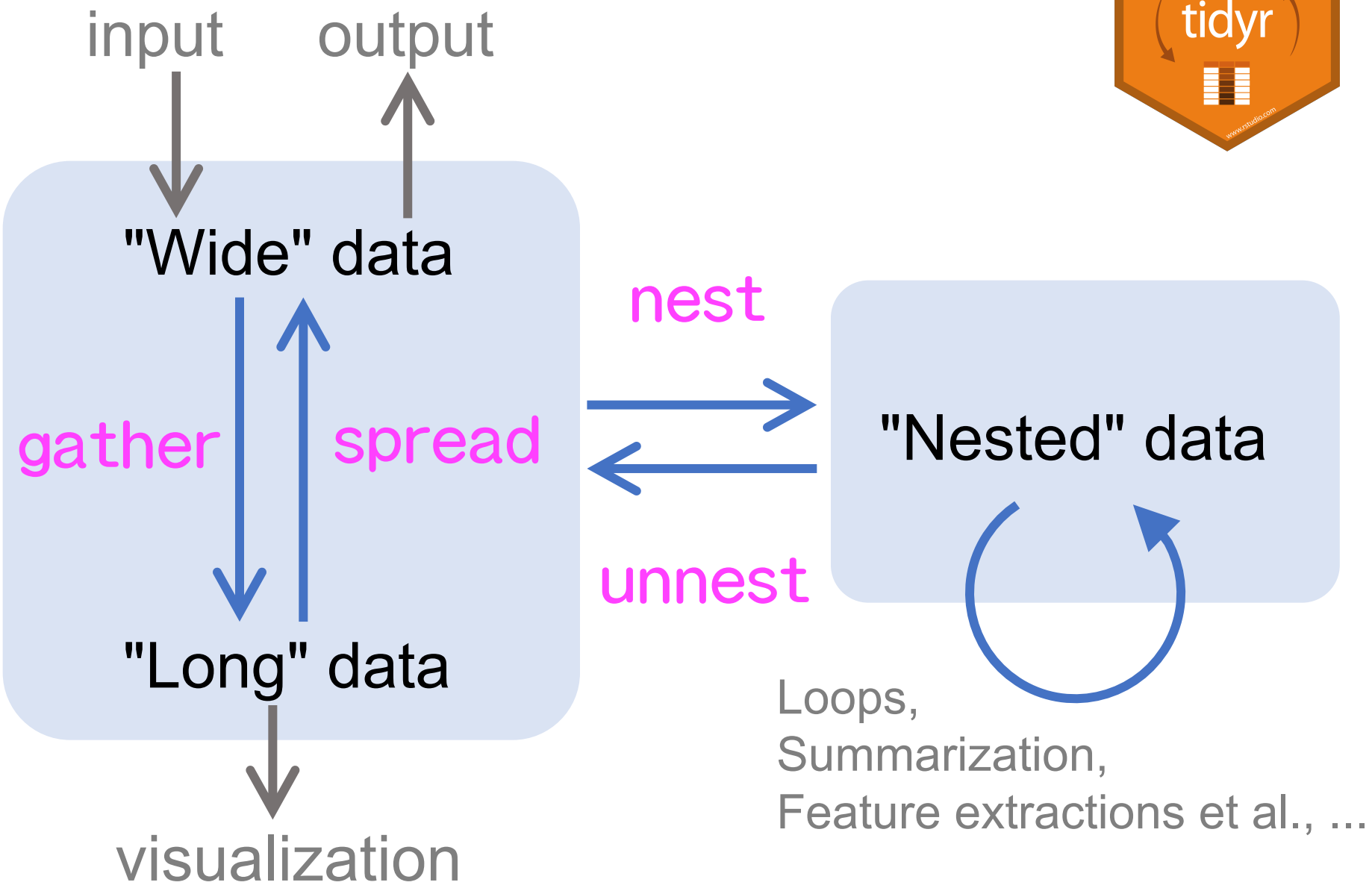
1. Each variable forms a column.
2. Each observation forms a row.
3. Each value must have its own cell.

country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766

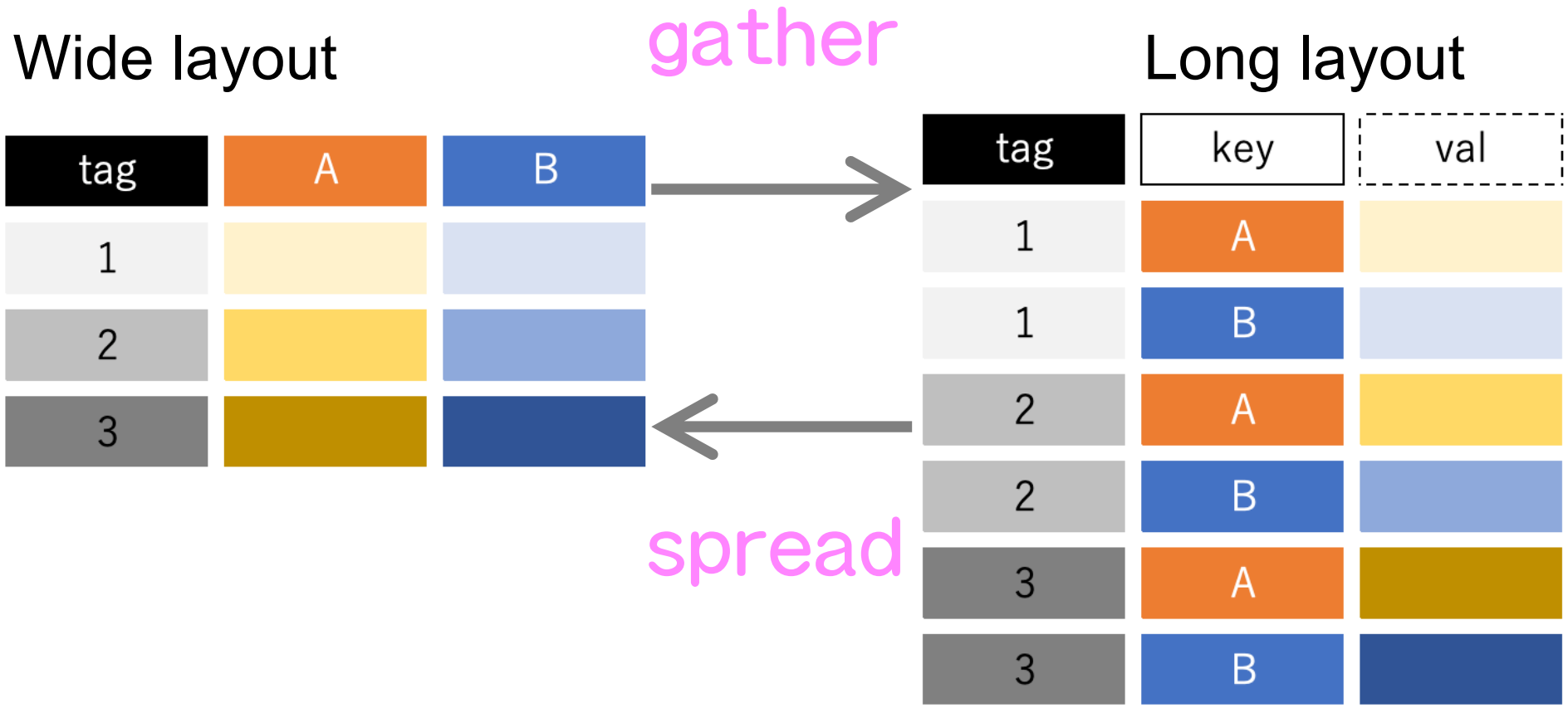
country	year	value
Afghanistan	1999	745
Brazil	1999	37737
China	1999	212258
Afghanistan	2000	2666
Brazil	2000	80488
China	2000	213766

data tidying

Data layout


input
output
"Wide" data
gather
spread
"Long" data
visualization
nest
unnest
"Nested" data
Loops,
Summarization,
Feature extractions et al., ...

Data layout




```
long_dat <- gather(wide_dat, key, val, -tag)
wide_dat <- spread(long_dat, key, val)
```

"Wide" data

```
> df
  obsid group top left width height
1     1     A 575  132    15     24
2     2     A 604  113    21      6
3     3     B 422  439    28     15
4     4     B 401  245    11      3
5     5     A  40  490    29     25
```

variables

"Long" data

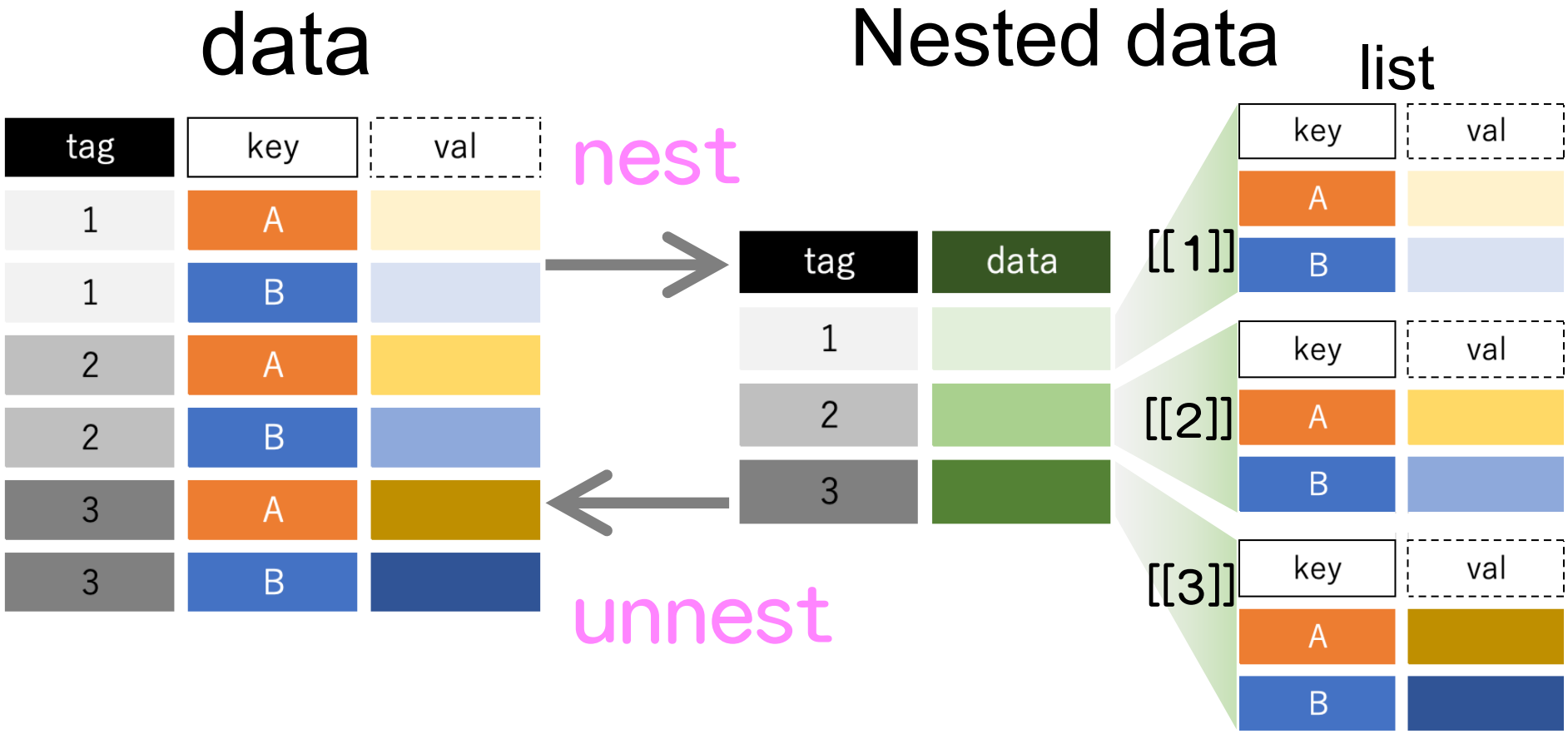
```
gather(df, key, value, -c(obsid, group))
```

{tidyr}

```
  obsid group   key value
1     1     A   top   575
2     2     A   top   604
3     3     B   top   422
4     4     B   top   401
5     5     A   top    40
6     1     A  left   132
```

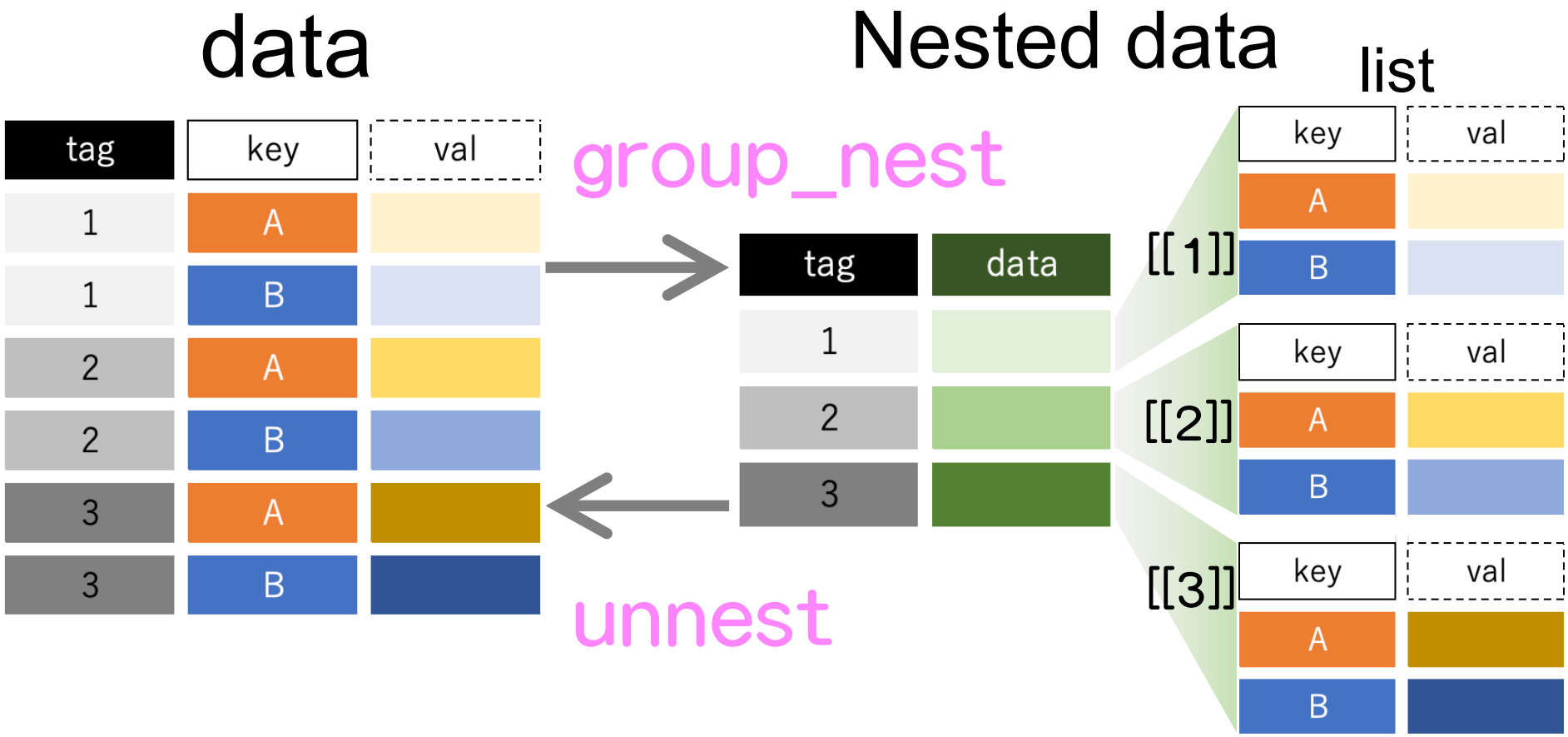
variables

Data layout




```
n_dat <- nest(group_by(dat, tag))
dat <- unnest(n_dat)
```

Data layout




```
n_dat <- group_nest(dat, tag)    # dplyr 0.8 ↑
dat <- unnest(n_dat)
```

"Wide" data

```
> df
  obsid group top left width height
1     1     A 575  132    15     24
2     2     A 604  113    21      6
3     3     B 422  439    28     15
4     4     B 401  245    11      3
5     5     A  40  490    29     25
```

"Nested" data

```
group_nest(df, group)
```

```
# A tibble: 2 x 2
  group data
  <fct> <list>
1 A     <tibble [3 x 5]>
2 B     <tibble [2 x 5]>
```

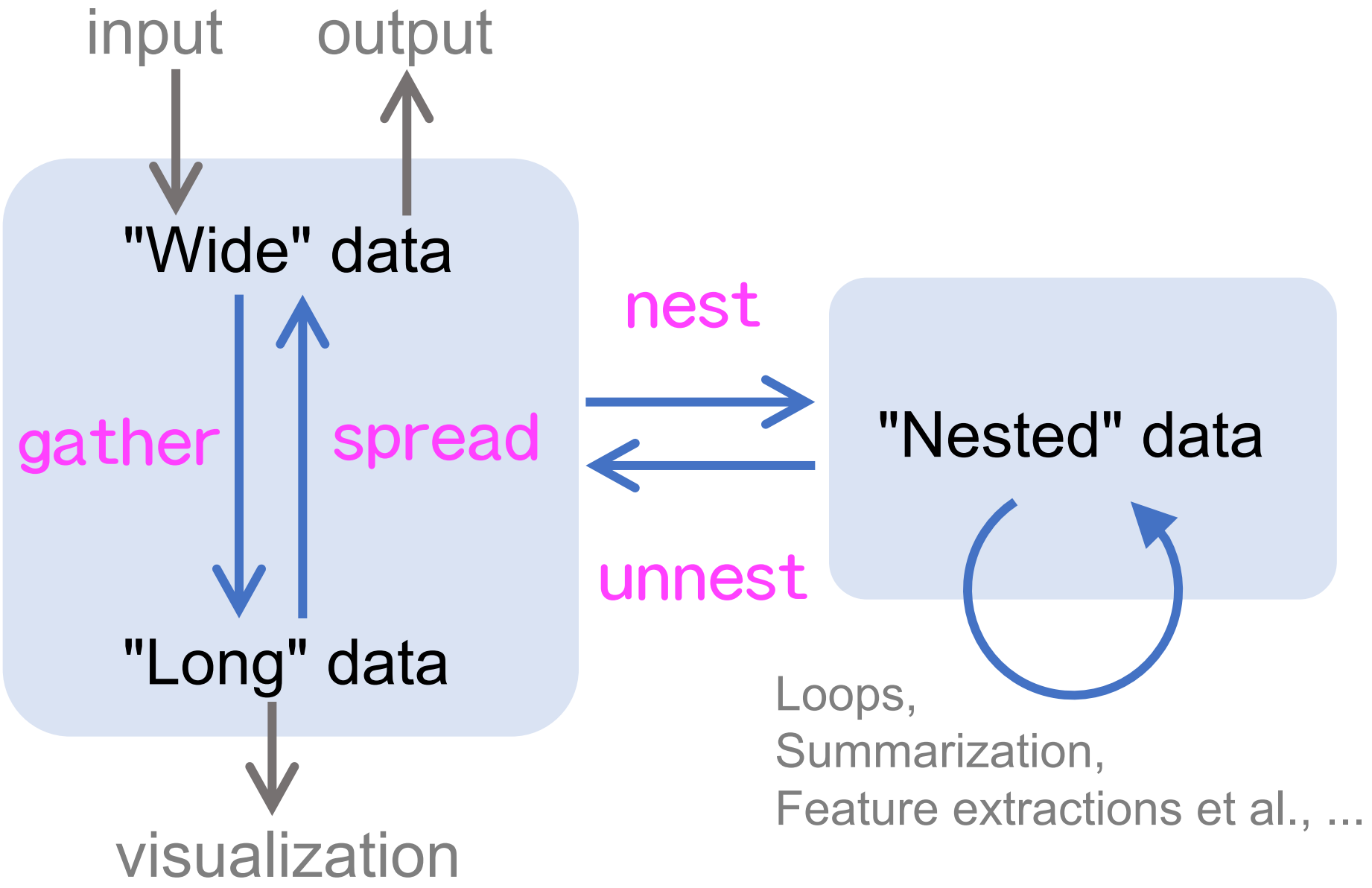
"Nested" data

```
df2 <- group_nest(df, group)
```

```
# A tibble: 2 x 2
  group data
  <fct> <list>
1 A     <tibble [3 × 5]>
2 B     <tibble [2 × 5]>
```

```
> df2$data
[[1]]
# A tibble: 3 x 5
  obsid   top  left width height
  <int> <int> <int> <int>  <int>
1     1   575   132    15     24
2     2   604   113    21      6
3     5    40   490    29     25

[[2]]
# A tibble: 2 x 5
  obsid   top  left width height
  <int> <int> <int> <int>  <int>
1     3   422   439    28     15
2     4   401   245    11      3
```

Data layout in {tidyr}



Agenda

- 済 0. Introduction
- 済 1. data.frame
- 済 2. Tidy data
- 3. Pipe
- 4. Verbs

Pipe %>%

{magrittr}

```
X %>% f          <-->  f(X)
X %>% f(y)       <-->  f(X, y)
X %>% f %>% g    <-->  g(f(X))
X %>% f(y, .)    <-->  f(y, X)
```

「dplyr再入門（基本編）」yutanihilation

https://speakerdeck.com/yutannihilation/dplyrzai-ru-men-ji-ben-bian

Pipe %>%

{magrittr}

【~~中毒~~ 愛用者たちの声】

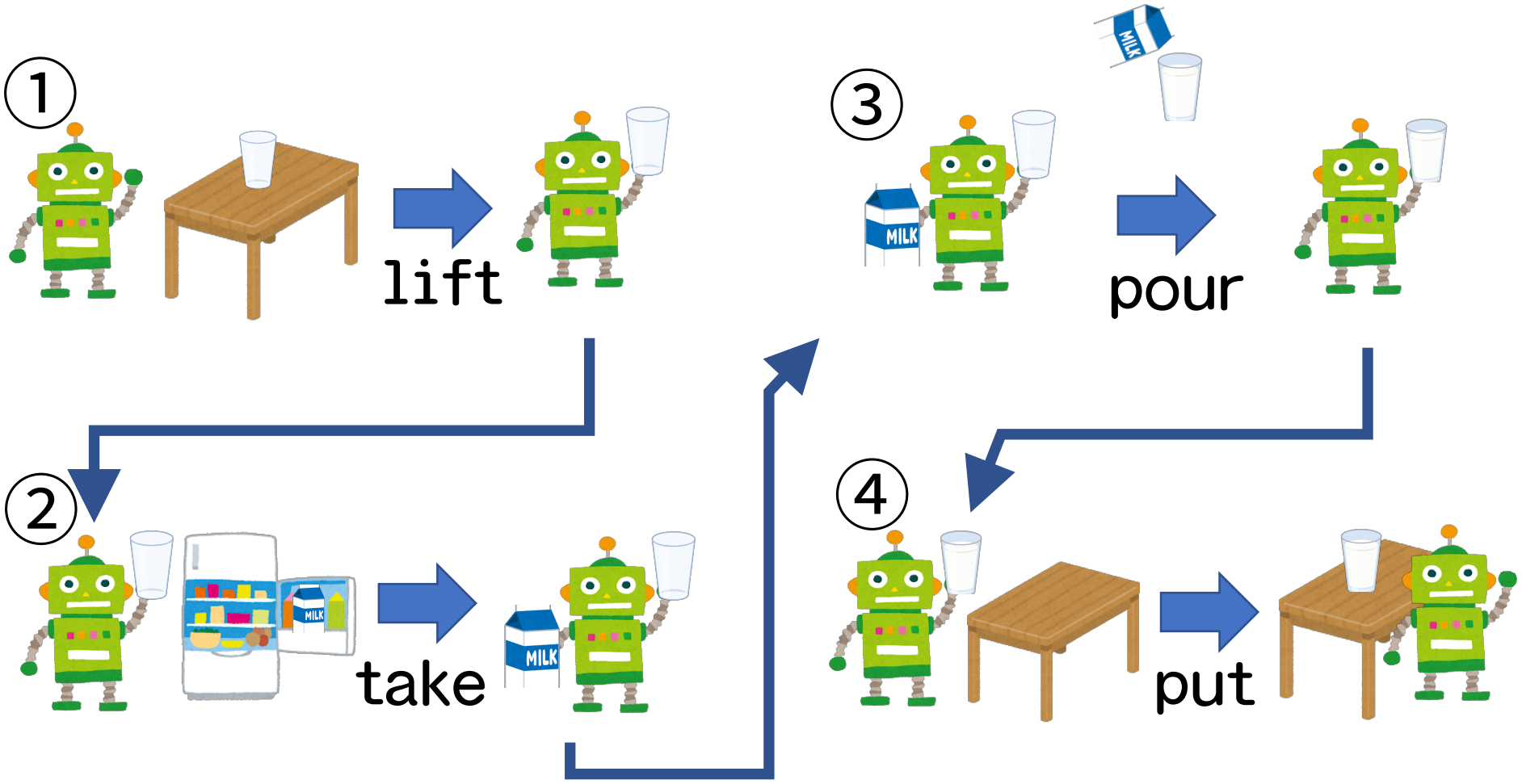
「最近パイプしか打ってないです」

「パイプ、あれはいいよなって
他の言語の人も皆んな思ってますよ」

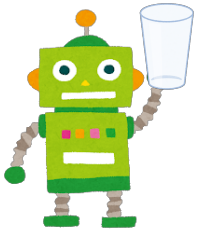
「1年ぐらいかけてゆっくりこっち
（パイプ）にシフトしましたね」

「Rコミュニティ四方山話」https://rlangradio.org/

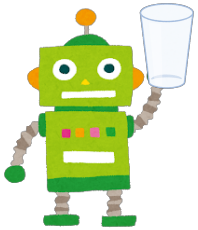
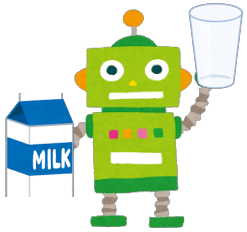
Bring milk from the kitchen!



Bring milk from the kitchen!

①   lift 

lift(Robot, glass, table) -> Robot′

②    take 

take(Robot′, fridge, milk) -> Robot″

Bring milk from the kitchen!

```
Robot′ <- lift(Robot, glass, table)          # ①
Robot″ <- take(Robot′, fridge, milk)         # ②
Robot‴ <- pour(Robot″, milk, glass)          # ③
result <- put(Robot‴, glass, table)          # ④
```

by using pipe,

```
result <- Robot %>%
    lift(glass, table) %>%                   # ①
    take(fridge, milk) %>%                   # ②
    pour(milk, glass) %>%                    # ③
    put(glass, table)                        # ④
```

The tidyverse style guides

2.1 Object names

"There are only two hard things in Computer Science: cache invalidation and **naming things**"

(so called snake case) to separate words within a name.

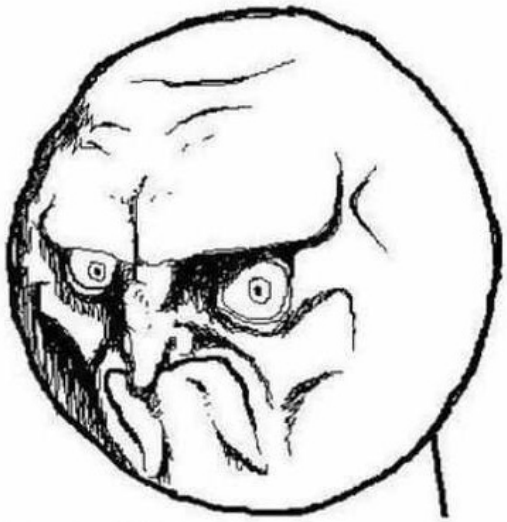
```
# Good
day_one
day_1

# Bad
DayOne
dayone
```

Base R uses dots in function names (`contrib.url()`) and class names (`data.frame`), but it's better to reserve dots exclusively for the S3 object system. In S3, methods are given the name `function.class` ; if you also use `.` in function and class names, you end up with confusing methods like `as.data.frame.data.frame()` .

https://style.tidyverse.org/syntax.html#object-names

Bring milk from the kitchen!

```
Robot′ <- lift(Robot, glass, tabl
Robot′′ <- take(Robot′, fridge, m
Robot′′′ <- pour(Robot′′, milk, gl
result <- put(Robot′′′, glass, tab
```



by using pipe,

```
result <- Robot %>%
    lift(glass, table) %>%
    take(fridge, milk) %>%
    pour(milk, glass) %>%
    put(glass, table)
```

Bring milk from the kitchen!

Thinking

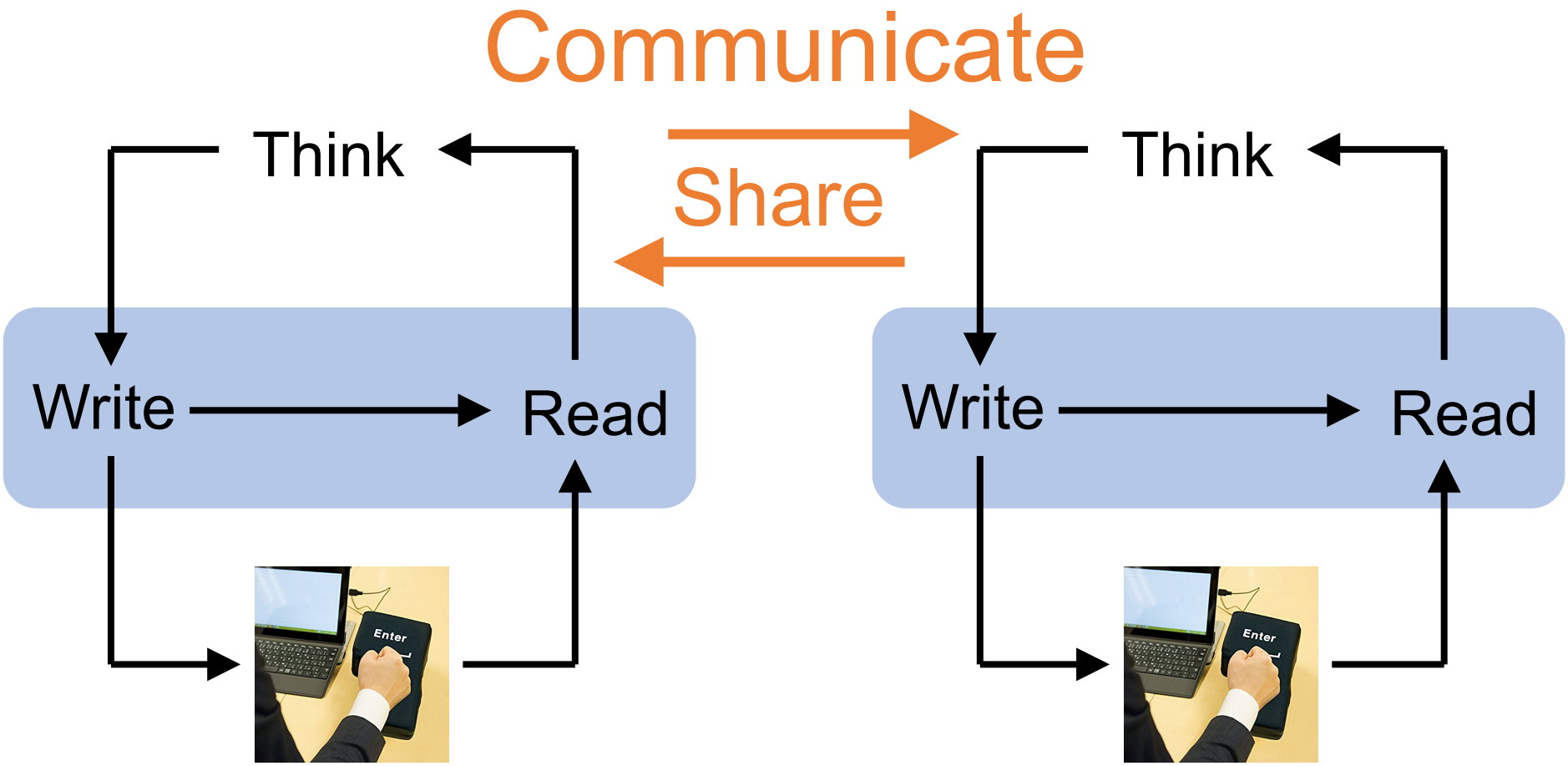
Reading

```
Robot′ <- lift(Robot, glass, table)          # ①
Robot″ <- take(Robot′, fridge, milk)         # ②
Robot‴ <- pour(Robot″, milk, glass)          # ③
result <- put(Robot‴, glass, table)          # ④
```

by using pipe,

```
result <- Robot %>%
  lift(glass, table) %>%                     # ①
  take(fridge, milk) %>%                     # ②
  pour(milk, glass) %>%                      # ③
  put(glass, table)                          # ④
```

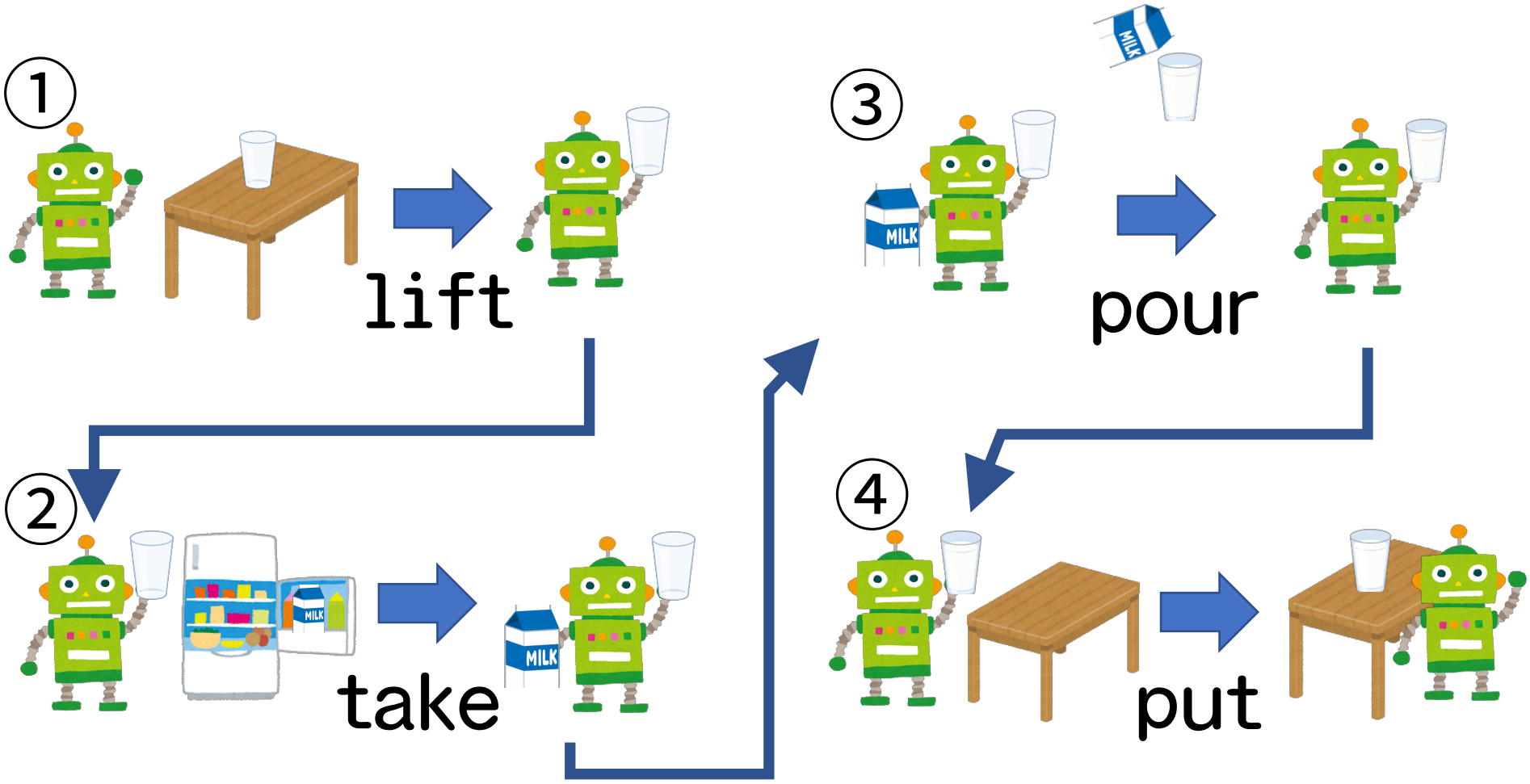
Programing


Communicate
Think
Share
Think
Write
Read
Write
Read
Enter
Enter

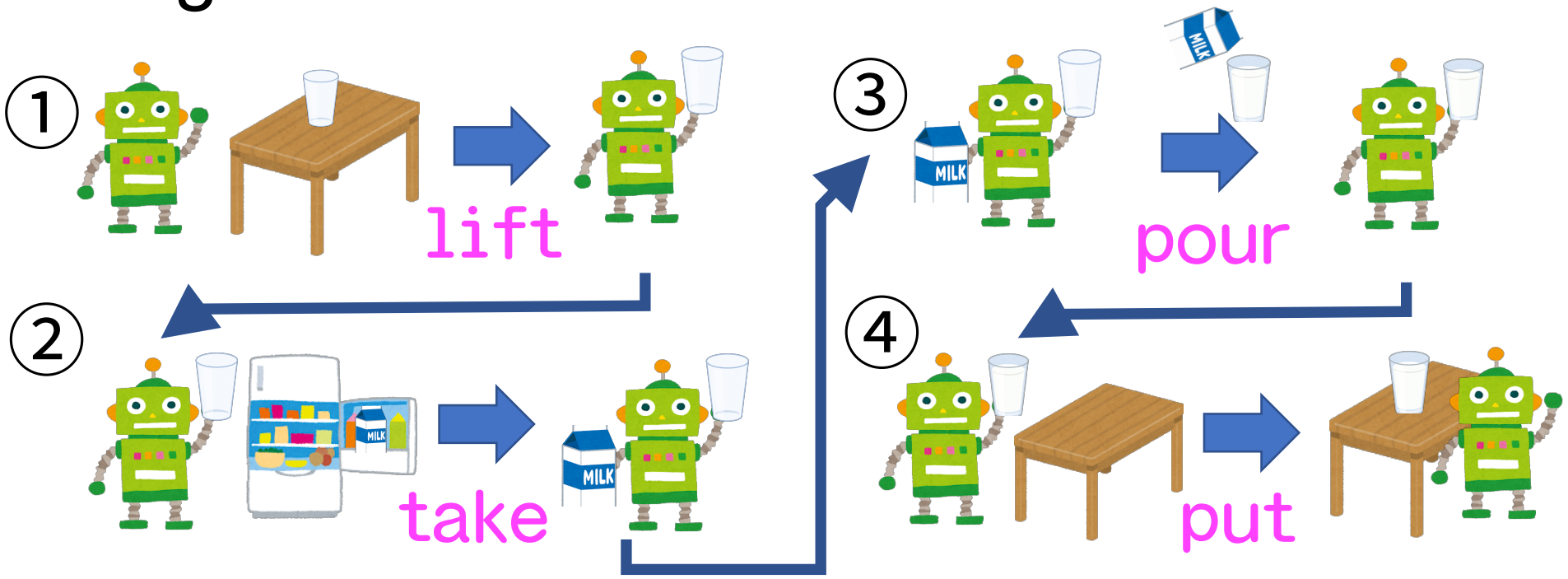
Agenda

- 済 0. Introduction
- 済 1. data.frame
- 済 2. Tidy data
- 済 3. Pipe
- 4. Verbs

Bring milk from the kitchen!



Bring milk from the kitchen!




```
result <- Robot %>%
    lift(glass, table) %>%
    take(fridge, milk) %>%
    pour(milk, glass) %>%
    put(glass, table)
```

Define an original function

```
please_bring <- function(someone, milk, glass,
                         table = dining_table,
                         fridge= kitchen_fridge){
  someone %>%
    lift(glass, table) %>%
    take(fridge, milk) %>%
    pour(milk, glass) %>%
    put(glass, table)
}
```

Usage

```
RobotA %>% please_bring(milk, my_glass)

RobotB %>% please_bring(cold_tea, her_glass)
```

General naming guidance to naming things

- nouns for variables
- verbs for functions

General naming guidance to naming things

- nouns for variables
- verbs for functions


The Highly Experimental Blog of Erik Rose

Function Names: To Verb Or Not To Verb?

Wed 20 March 2013

https://www.grinchcentral.com/function-names-to-verb-or-not-to-verb

General naming guidance to naming things

- nouns for variables
- verbs for functions

Conversely,

- variables are nouns
- functions are verbs

Functions are verbs.

```r
# noun (名詞的)
df[df$x == "a" & df$y == 1, ]
```

```r
# verb (動詞的)                                {dplyr}
filter(df, x == "a", y == 1)
```

```r
# verb with pipe
df %>%
  filter(x == "a", y == 1)
```

verbs (verb functions) {dplyr}

```
mutate      # add column
select      # select column
filter      # select row
arrange     # arrange row
summaries   # summary of vars
```

verbs

{dplyr}

It (dplyr) provides simple “verbs” to help you translate your thoughts into code.

functions that correspond to the most common data manipulation tasks

Introduction to dplyr

https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html

verbs

{dplyr}

dplyrは、あなたの考えをコードに翻訳するための【動詞】を提供する。

データ操作における基本のキを、
シンプルに実行できる関数 (群)

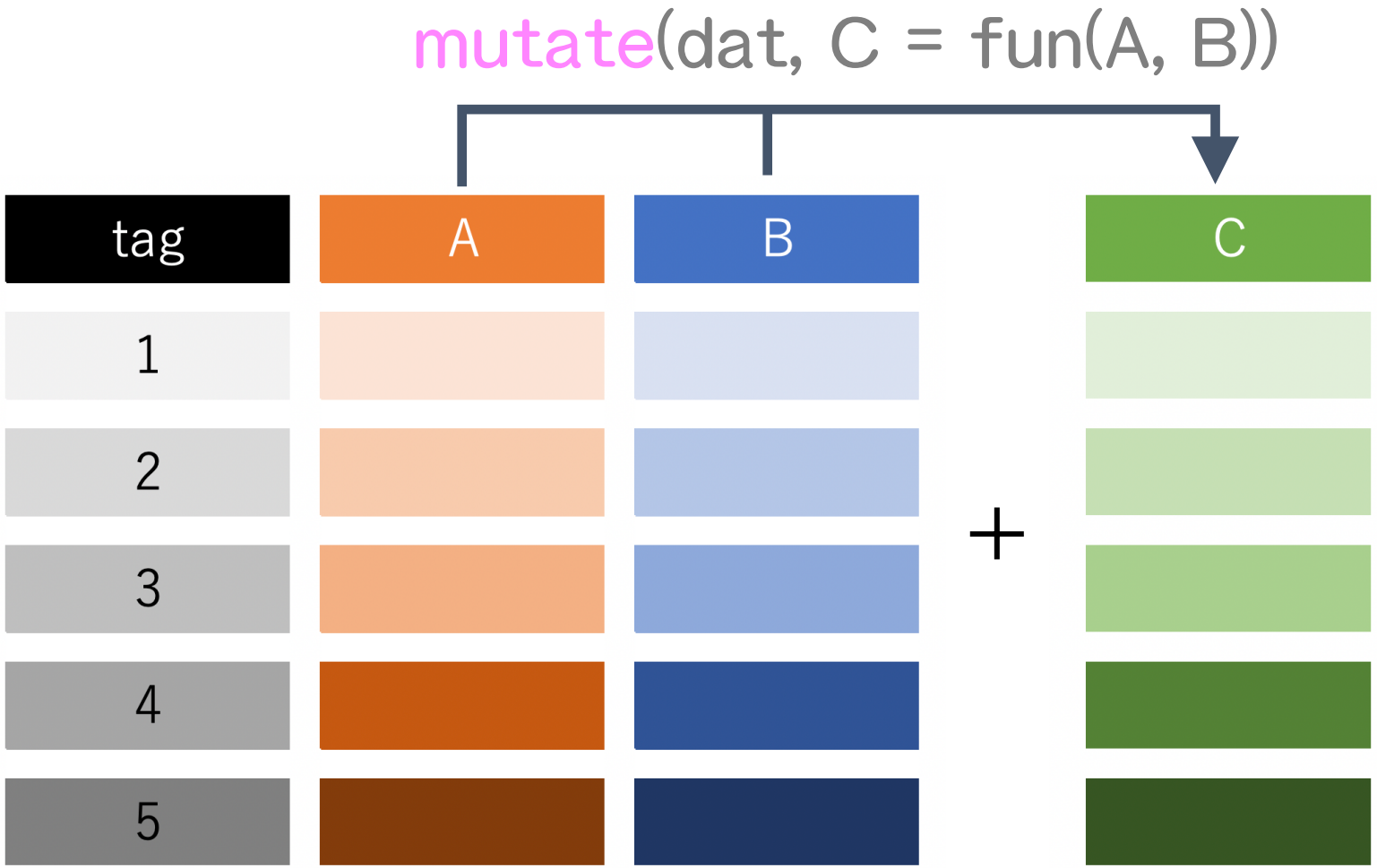
※ かなり意訳

Introduction to dplyr
https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html

verbs

{dplyr}

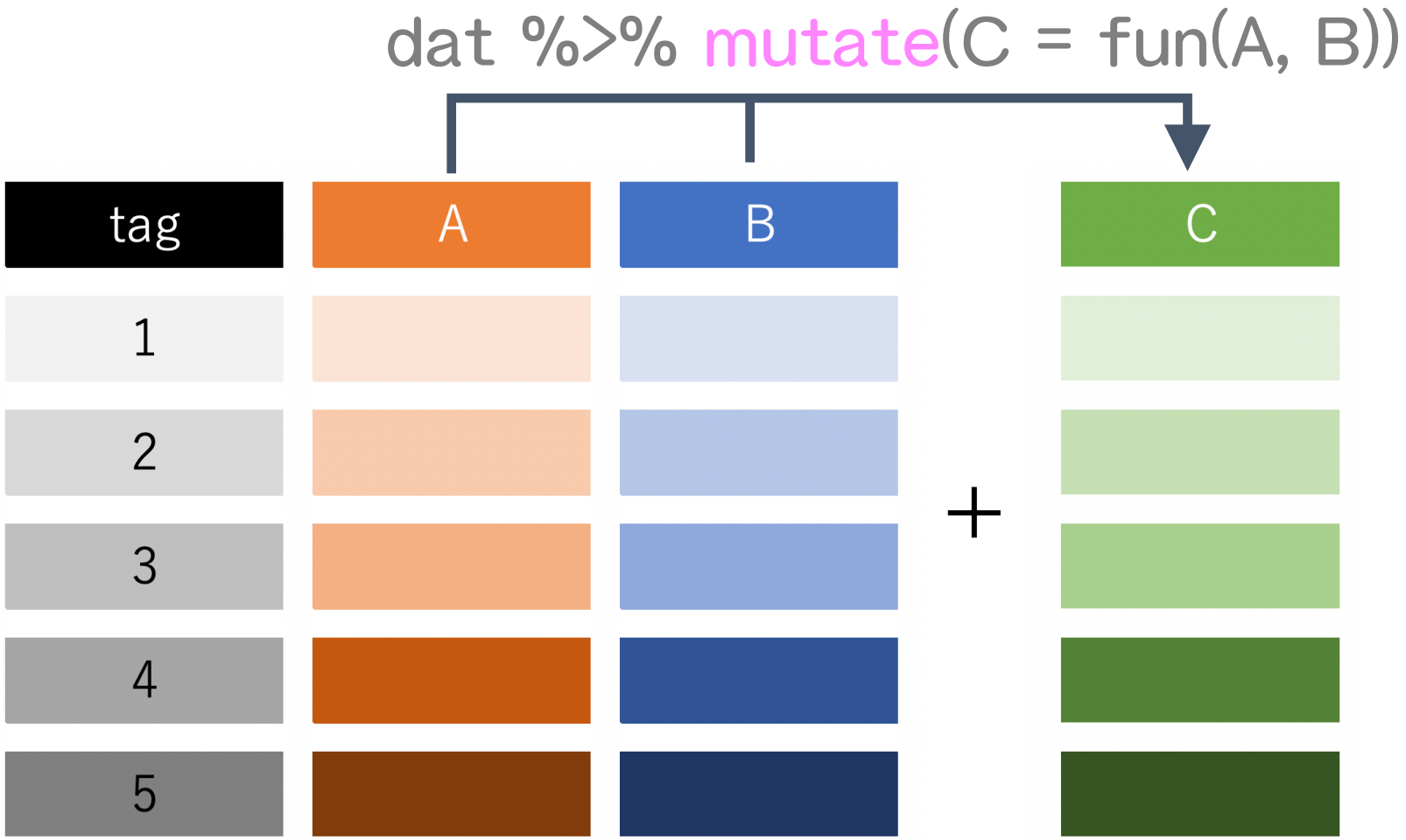
mutate　　＃カラムの追加



verbs

{dplyr}

mutate　　＃カラムの追加



verbs

{dplyr}

mutate　　＃カラムの追加

```
df1 <- data.frame(A = 1:3,
                  B = 11:13)
```

```
> df1
  A  B
1 1 11
2 2 12
3 3 13
```

```
df2 <- df1 %>%
  mutate(C = A * B)
```

```
> df2
  A  B  C
1 1 11 11
2 2 12 24
3 3 13 39
```

verbs

{dplyr}

```
library(dplyr)
iris %>% mutate(a = 1:nrow(.)) %>% str
```

```
'data.frame':    150 obs. of  6 variables:
$ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 ...
$ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 ...
$ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7...
$ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 ...
$ Species     : Factor w/ 3 levels "setosa", ...
$ a           : int  1 2 3 4 5 6 7 8 9 10 ...
```

verbs

{dplyr}

```
library(dplyr)
iris %>% mutate(a = 1:nrow(.),
                a = a * 5/3 %>% round)
```

over write

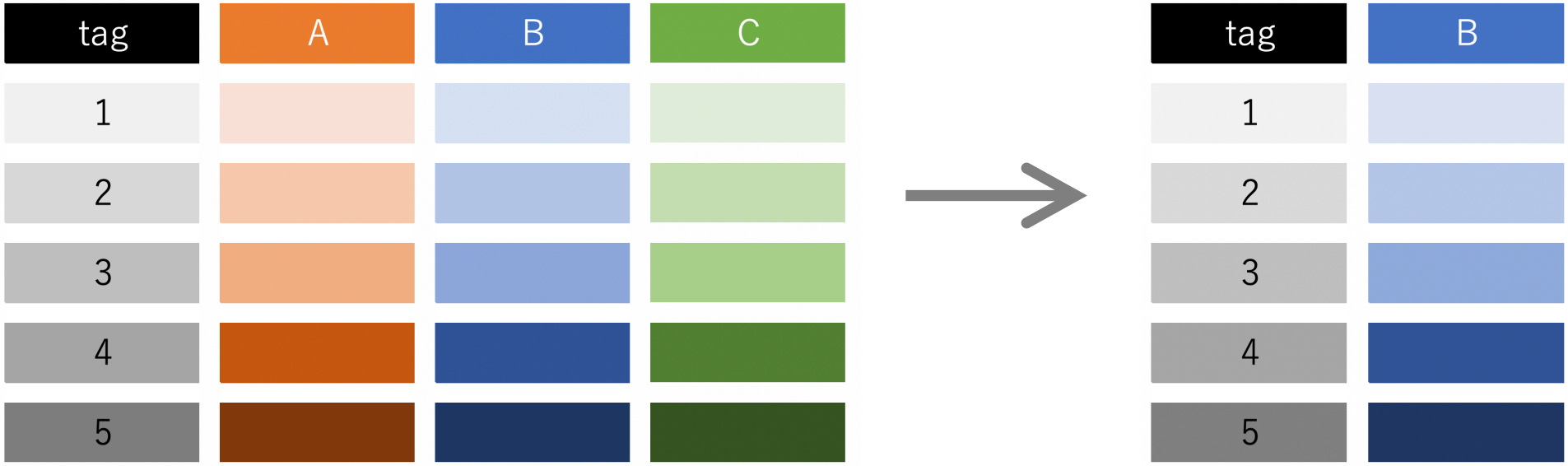
```
'data.frame':    150 obs. of  6 variables:
$ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 ...
$ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 ...
$ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7...
$ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 ...
$ Species     : Factor w/ 3 levels  "setosa" , ...
$ a           : num  1.67 3.33 5 6.67 8.33 ... ...
```

verbs

{dplyr}

select　# カラムの選択

dat %>% select(tag, B)



verbs

{dplyr}

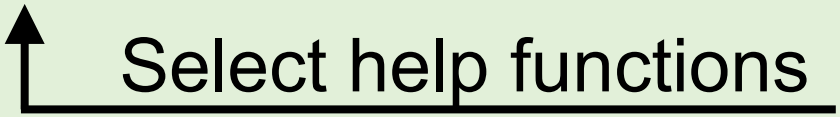
```
library(dplyr)
iris %>% select(Sepal.Length, Sepal.Width)
```

```
'data.frame':    150 obs. of  6 variables:
$ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 ...
$ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 ...
```

verbs

{dplyr}

```
library(dplyr)
iris %>% select(contains("Width"))
```




```
'data.frame':    150 obs. of  6 variables:
$ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 ...
$ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 ...
```

verbs

{dplyr}

Select help functions

```
starts_with("s")    ends_with("s")

contains("se")      matches("^.e")

one_of(c("Sepal.Length", "Species"))

everything()
```

「dplyr::selectの活用例メモ」kazutan
https://kazutan.github.io/blog/2017/04/dplyr-select-memo/

verbs (verb関数群)

{dplyr}

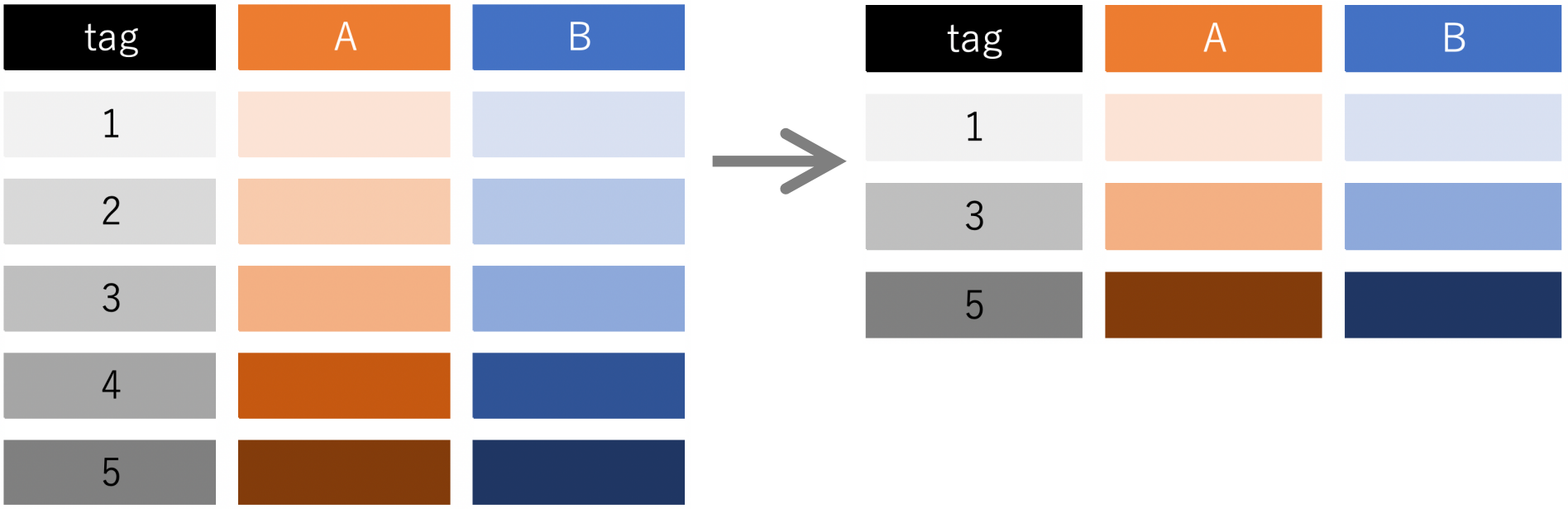
```
mutate      # カラムの追加
select      # カラムの選択
filter      # 行の絞り込み
arrange     # 行の並び替え
summaries   # 値の集約
```

verbs

{dplyr}

filter　＃行の絞り込み

```
dat %>% filter(tag %in% c(1, 3, 5))
```



verbs

{dplyr}

```
library(dplyr)
iris %>% filter(Species == "versicolor")
```

```
'data.frame':    50 obs. of  5 variables:
$ Sepal.Length: num  7 6.4 6.9 5.5 6.5 5.7 6.3 ...
$ Sepal.Width : num  3.2 3.2 3.1 2.3 2.8 2.8 ...
$ Petal.Length: num  4.7 4.5 4.9 4 4.6 4.5 4.7 ...
$ Petal.Width : num  1.4 1.5 1.5 1.3 1.5 1.3 ...
$ Species     : Factor w/ 3 levels
      "setosa","versicolor",..: 2 2 2 2 2 2 2 2 2 2 ...
```

verbs

{dplyr}

```
library(dplyr)
iris %>% filter(Species == "versicolor")
```

NSE (Non-Standard Evaluation)

```
'data.frame':    50 obs. of  5 variables:
$ Sepal.Length: num  7 6.4 6.9 5.5 6.5 5.7 6.3 ...
$ Sepal.Width : num  3.2 3.2 3.1 2.3 2.8 2.8 ...
$ Petal.Length: num  4.7 4.5 4.9 4 4.6 4.5 4.7 ...
$ Petal.Width : num  1.4 1.5 1.5 1.3 1.5 1.3 ...
$ Species     : Factor w/ 3 levels
      "setosa","versicolor",..: 2 2 2 2 2 2 2 2 2 2 ...
```

NSEの話

NSE (Non-Standard Evaluation)

```
filter(df, x == "a", y == 1)
```

SE (Standard Evaluation)

```
df[df$x == "a" & df$y == 1, ]
```

Programming with dplyr
http://dplyr.tidyverse.org/articles/programming.html

NSEの話

NSEを使うと、

- dfの名前を何回も書かなくていいよ
- SQLっぽく書けるよ

```
filter(df, x == "a", y == 1)
```

```
df[df$x == "a" & df$y == 1, ]
```

Programming with dplyr
http://dplyr.tidyverse.org/articles/programming.html

NSEの話

書きやすく、読みやすく。
思考と実装の距離を近く。

NSEを使うと、

色々あるけどスッキリしているのは正義 (私見)

```
filter(df, x == "a", y == 1)       # verb (動詞的)
```

```
df[df$x == "a" & df$y == 1, ]   # noun (名詞的)
```

Programming with dplyr
http://dplyr.tidyverse.org/articles/programming.html

NSEの話

Because of NSE..

```
df <- data.frame(x = 1:3, y = 1:3)
filter(df, x == 1)
```

This  do NOT work

```
my_var <- "x"
filter(df, my_var == 1)
```

There is No “my_var” column in df

Programming with dplyr
http://dplyr.tidyverse.org/articles/programming.html

NSEの話

ど〜〜〜してもやりたければ、

```
my_var <- quo(x)
filter(df, (!! my_var) == 1)
```

何故こうなるかは、

「dplyr再入門（Tidyval編）」を参照。

「dplyr再入門（Tidyval編）」yutanihilation
https://speakerdeck.com/yutannihilation/dplyrzai-ru-men-tidyevalbian

NSEの話

ど～～～～し

可読性が上がる？下がる？
それは、あなたと読み手次第。

```
my_var <- quo(x)

filter(df, (!! my_var) == 1)
```

何故こうなるかは、

「dplyr再入門（Tidyval編）」を参照。

「dplyr再入門（Tidyval編）」yutanihilation
https://speakerdeck.com/yutannihilation/dplyrzai-ru-men-tidyevalbian

verbs (verb関数群)

{dplyr}

```
mutate      # カラムの追加
select      # カラムの選択
filter      # 行の絞り込み
arrange     # 行の並び替え
summaries   # 値の集約
```

Grammar of data manipulation

By **constraining** your options,

it helps you think about your

data manipulation challenges.

Introduction to dplyr

https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html

Grammar of data manipulation

選択肢を制限することで、
データ解析のステップを
シンプルに考えられます㐂。

※ まさに意訳

Introduction to dplyr
https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html

より多くの制約を課す事で、

魂の足枷から、より自由になる。

The more constraints one imposes,
the more one frees one's self of the
chains that shackle the spirit.

Igor Stravinsky

Игорь Ф Стравинский

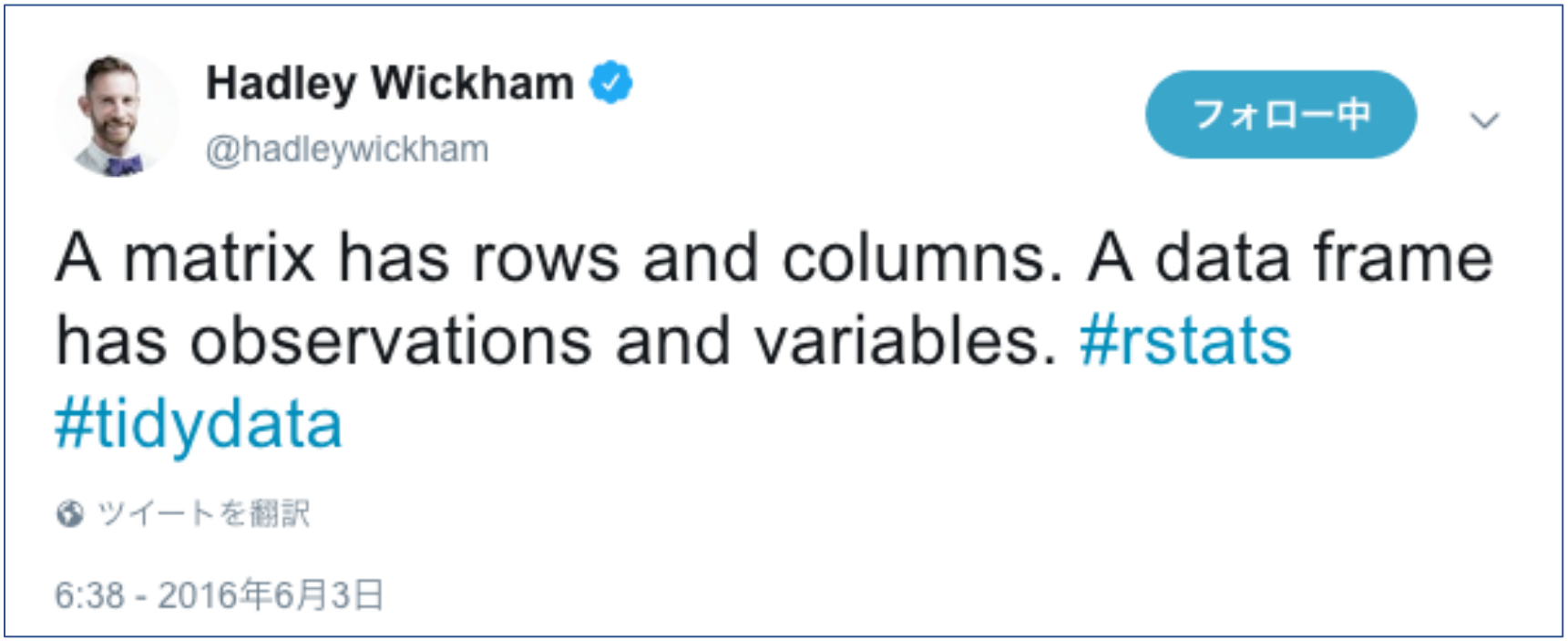
1882 - 1971

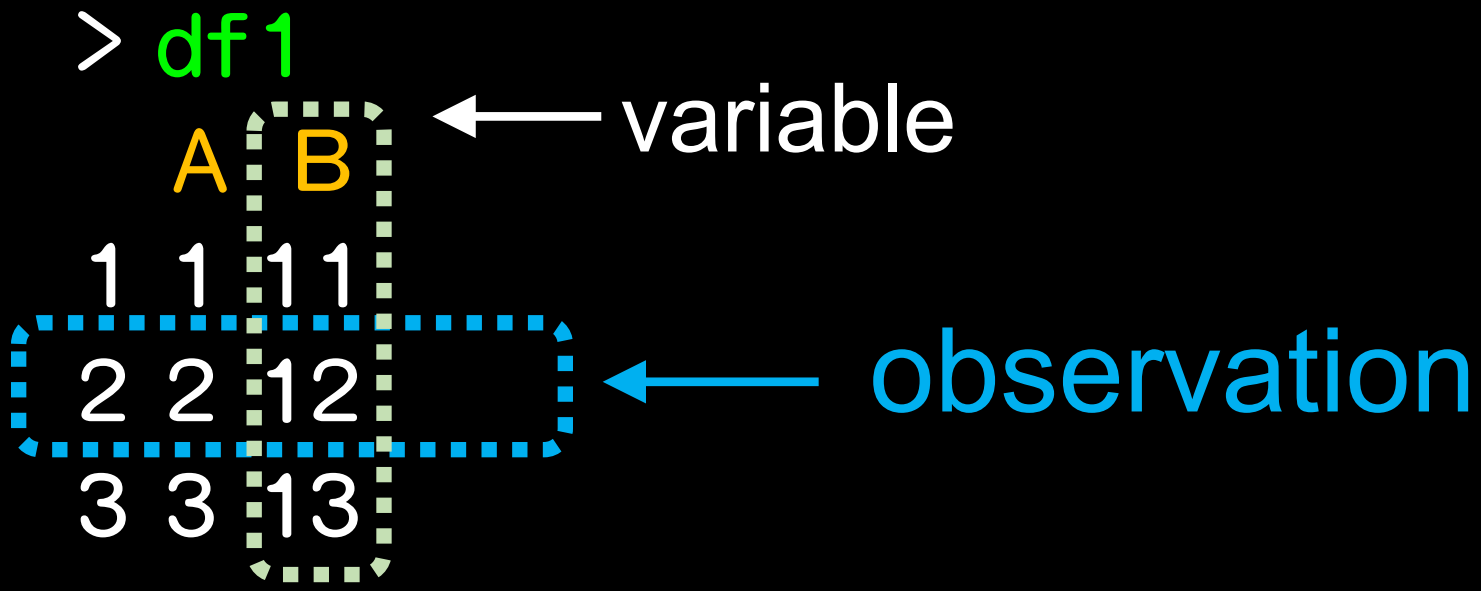
※ 割と意訳

Agenda

- 済 0. Introduction
- 済 1. data.frame
- 済 2. Tidy data
- 済 3. Pipe
- 今 4. Verbs

Summary


Hadley Wickham
@hadleywickham
フォロー中
A matrix has rows and columns. A data frame has observations and variables. #rstats #tidydata
ツイートを翻訳
6:38 - 2016年6月3日


> df1
A B
1 1 11
2 2 12
3 3 13
variable
observation

In tidy data:

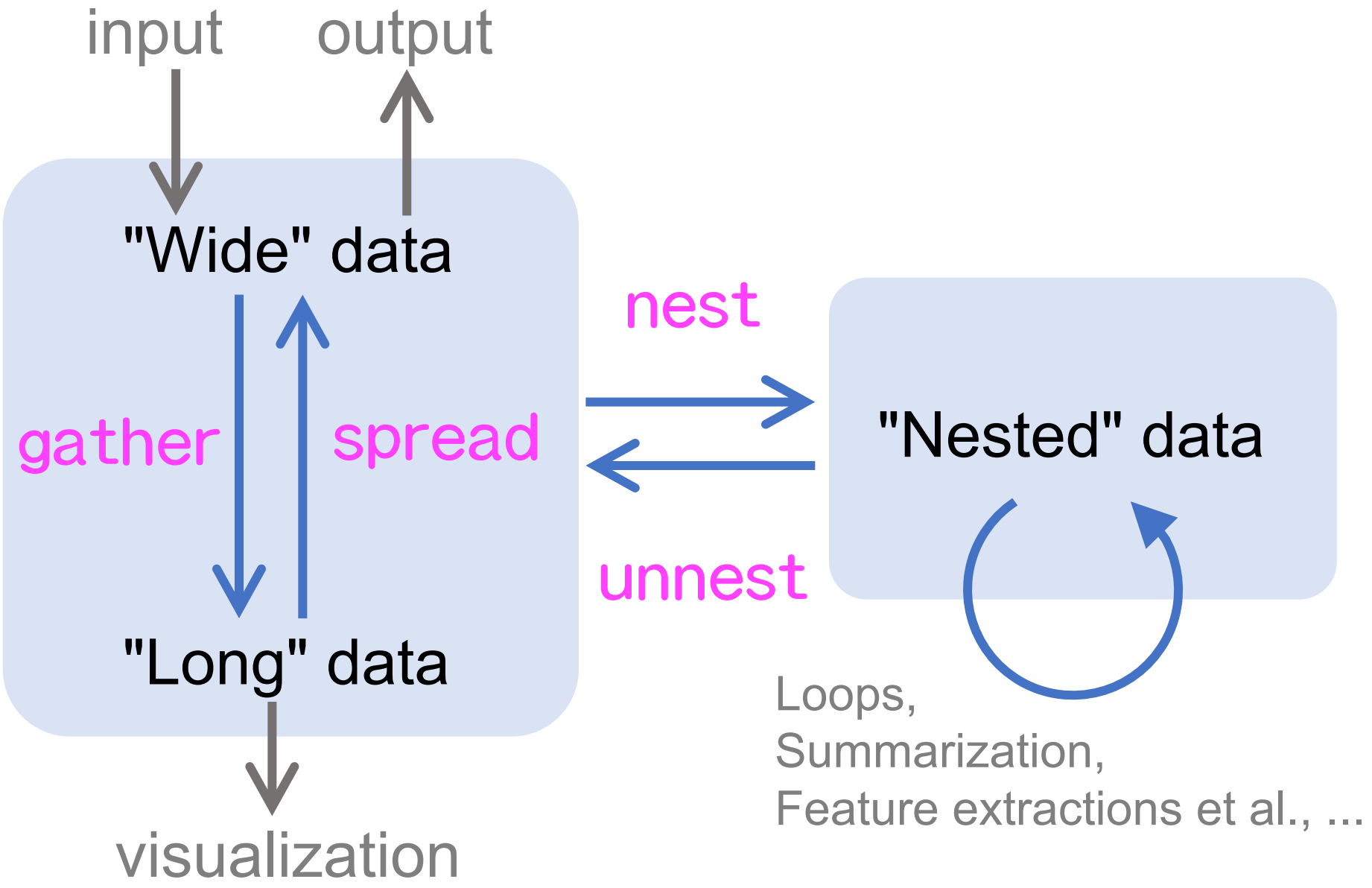
1. Each variable forms a column.
2. Each observation forms a row.
3. Each value must have its own cell.

country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766

country	year	value
Afghanistan	1999	745
Brazil	1999	37737
China	1999	212258
Afghanistan	2000	2666
Brazil	2000	80488
China	2000	213766

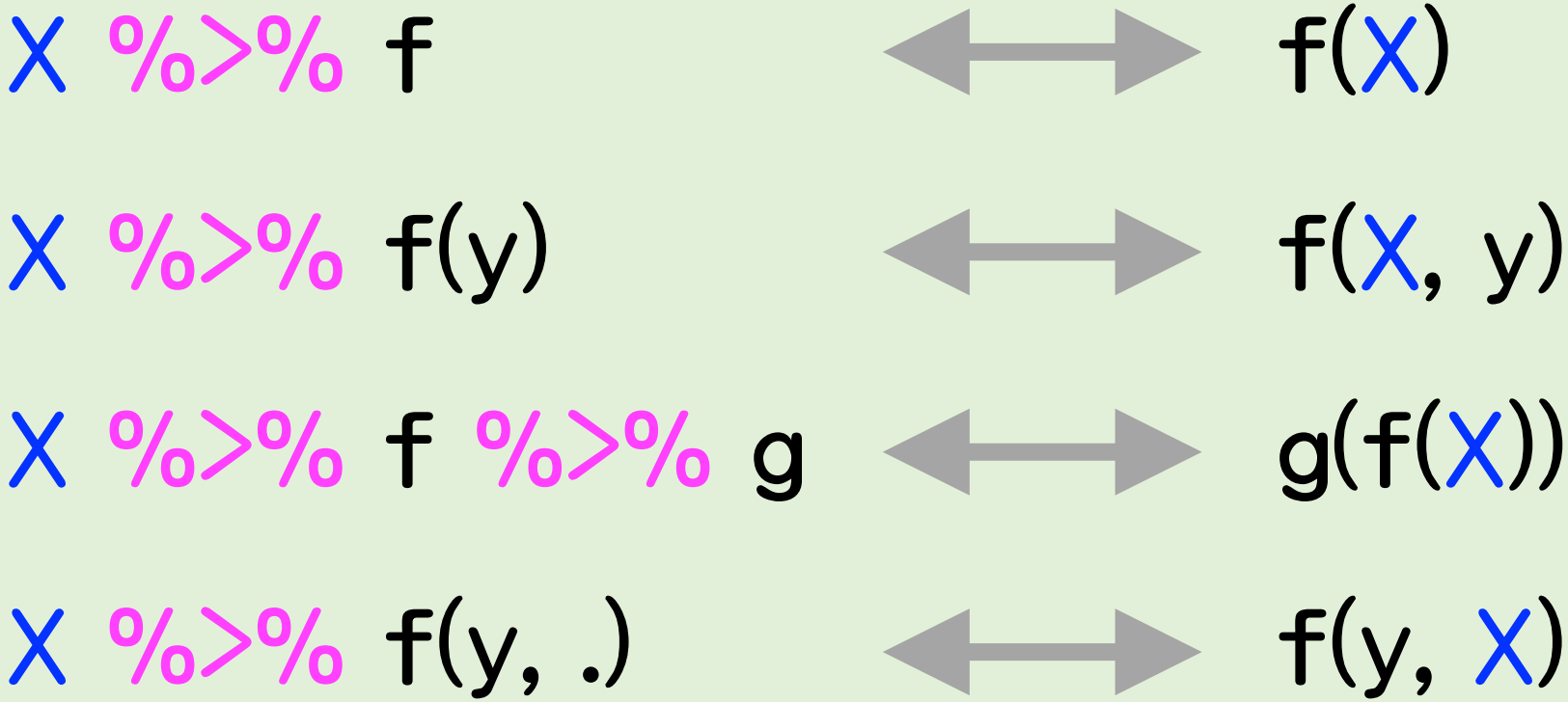
data tidying

Data style manipulation in {tidyr}



Pipe %>%

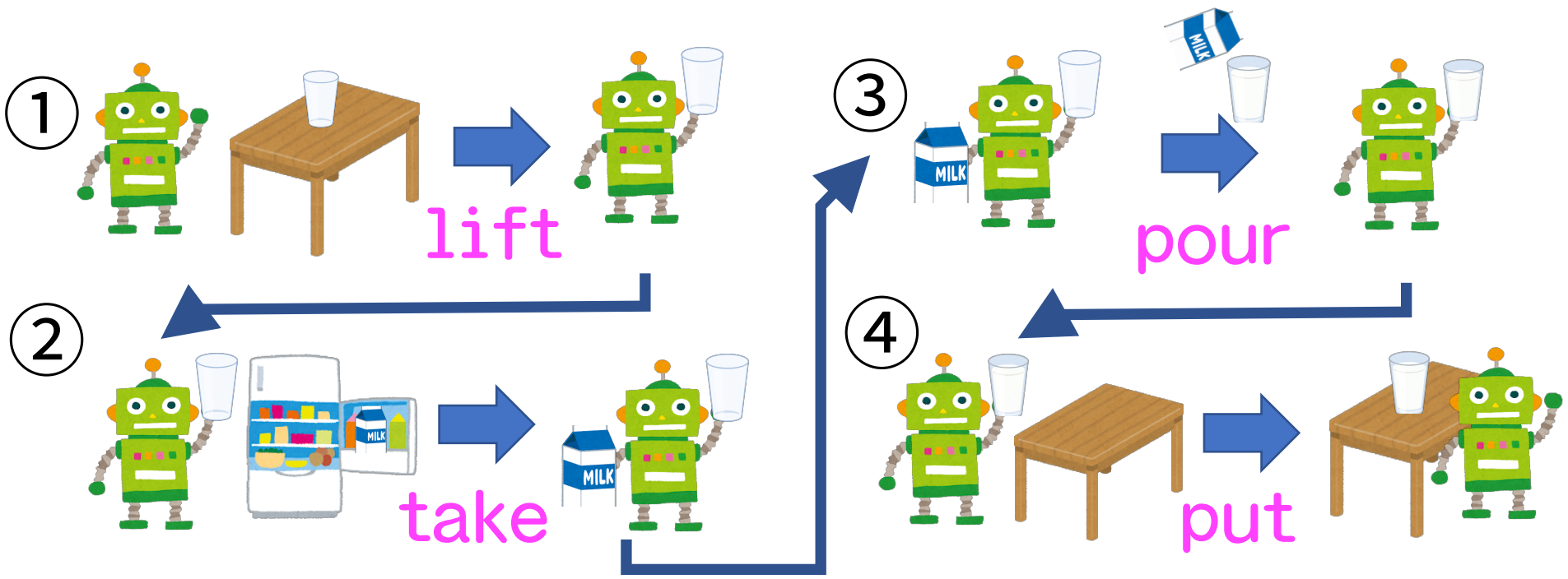
{magrittr}



「dplyr再入門（基本編）」yutanihilation

https://speakerdeck.com/yutannihilation/dplyrzai-ru-men-ji-ben-bian

Functions are verbs




```
result <- Robot %>%
    lift(glass, table) %>%
    take(fridge, milk) %>%
    pour(milk, glass) %>%
    put(glass, table)
```

noun (名詞的)

```
df[df$x == "a" & df$y == 1, ]
```

verb (動詞的)

{dplyr}

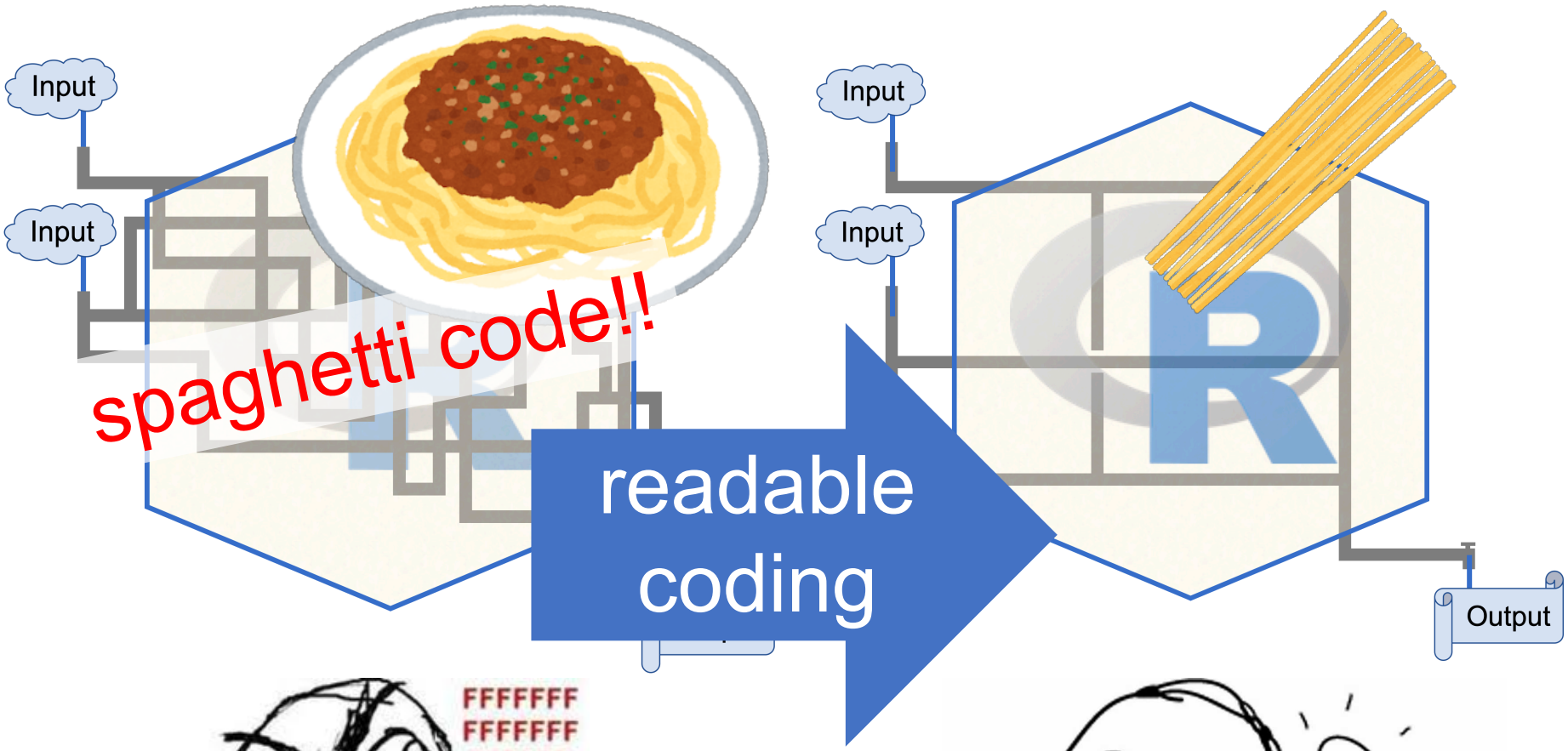
```
filter(df, x == "a", y == 1)
```

```
df %>%
  filter(x == "a", y == 1)
```

verbs (verb functions) {dplyr}

```
mutate      # add column
select      # select column
filter      # select row
arrange     # arrange row
summaries   # summary of vars
```

Data Pipeline



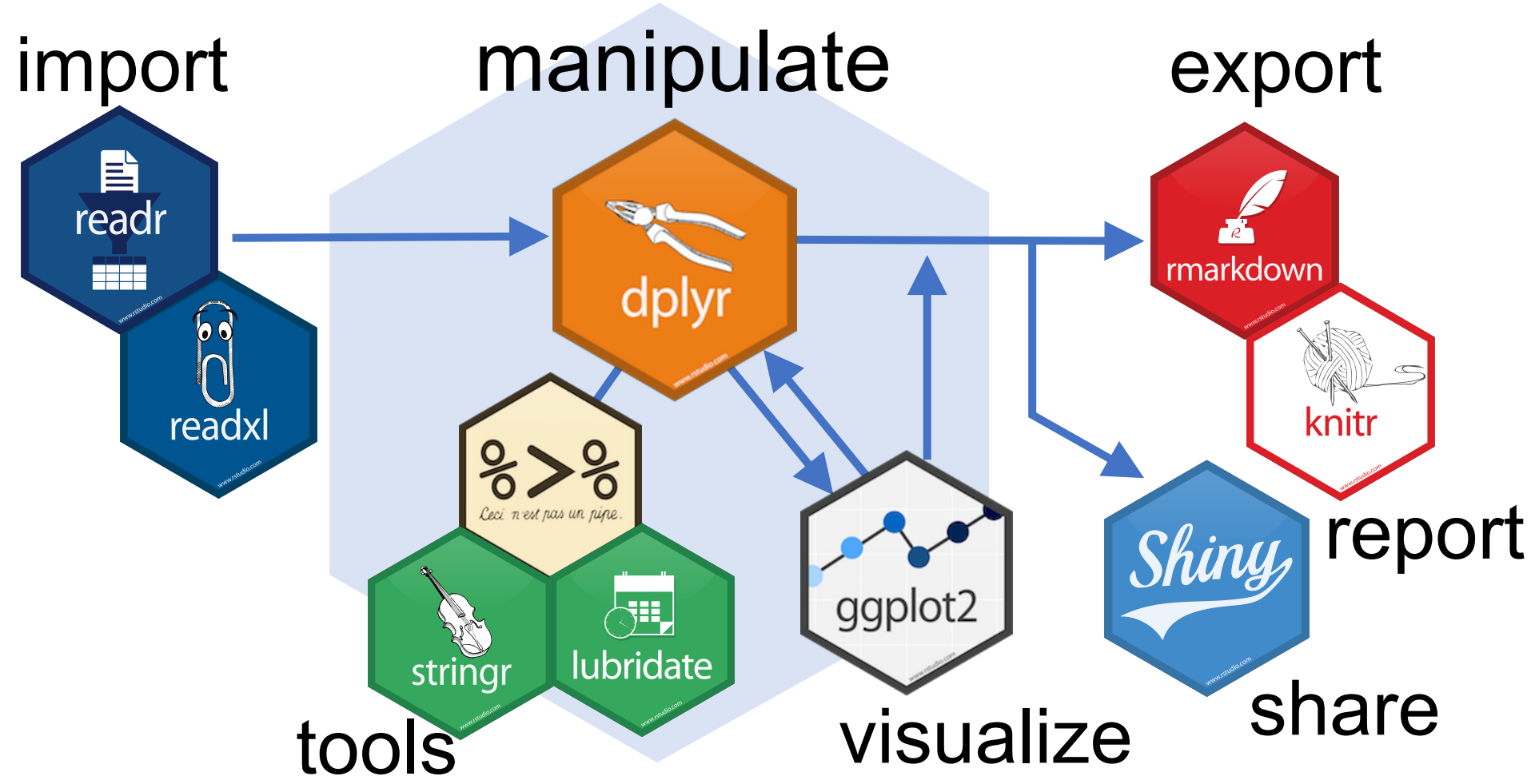
Tidyverse

https://www.tidyverse.org/

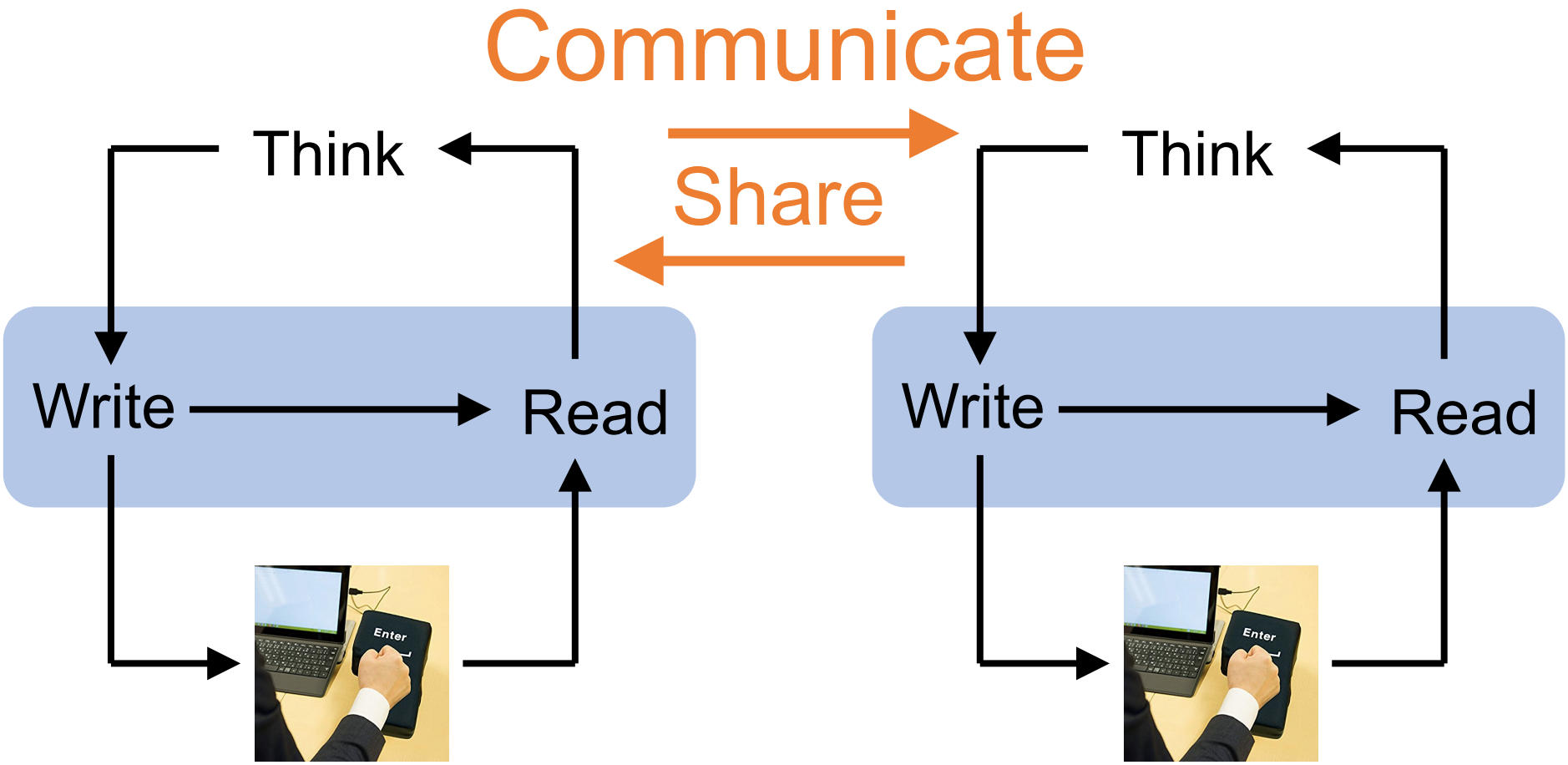
```
install.packages("tidyverse")
```

Tidyverse

https://www.tidyverse.org/



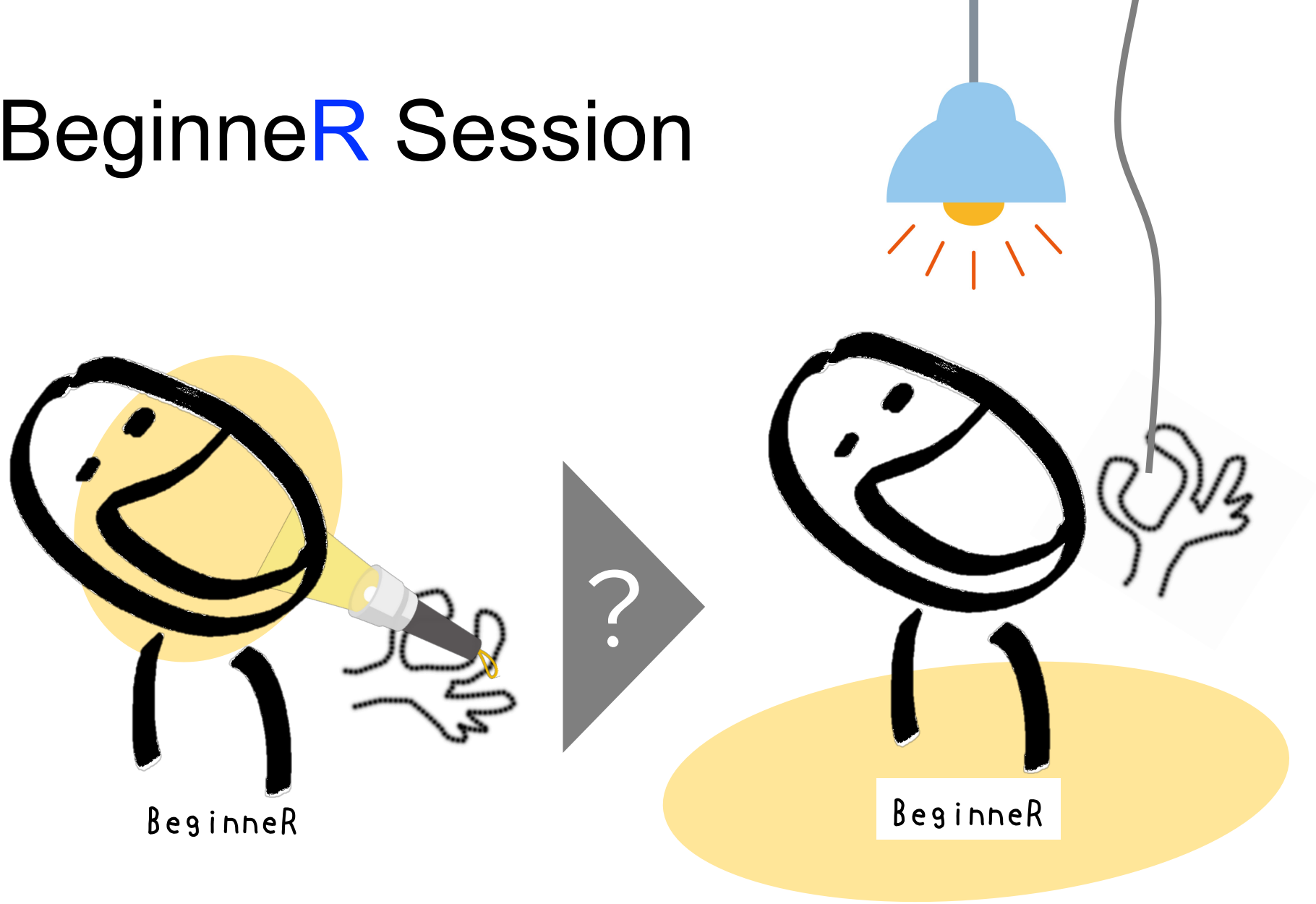
Programing languages are language



“Life shrinks or expands to one’s courage.”

-- Anaïs Nin, 2000

BeginneR Session



Before

After

enjoy!


bar dradra
THE PENINSULA