data_wrangling() && ("manipulation" %in% R)

🐨🐟🌿 %>% %>% %>% 🤓📊🥰

```
postgraduate_workshop(
  dept = "Biological Sciences",
  presenter = c(
    "Ruan van Mazijk",
    "MSc candidate"
  )
)
```

`> face()`

`> logos()`

> introduce( )

> introduce( )

- BSc + Hons here at UCT

> introduce( )

- BSc + Hons here at UCT

- Ecology & evolution
- (Mostly plant) comparative biology
- Biogeography

> introduce( )

- BSc + Hons here at UCT

- Ecology & evolution
- (Mostly plant) comparative biology
- Biogeography

- Been working with R for 4½ years
  - Every major project I've done…

> introduce( )

Tetraria ustulata
Marloth NR
R. van Mazijk 2018

Tetraria thermalis
Silvermine, Table Mountain NP
R. van Mazijk 2018

Schoenus compar
Silvermine, Table Mountatin NP
R. van Mazijk 2018

```
> workshop$goals
```

> workshop$goals

- More reproducible science

> workshop$goals

- More reproducible science
- Save time by:
  - Automating repetitive tasks
  - Eliminating human error

> workshop$goals

- More reproducible science
- Save time by:
  - Automating repetitive tasks
  - Eliminating human error
- Boost your skills
- Think about your data programmatically

Notes & slides will go up here:

tinyurl.com/r-with-ruan

(But I encourage you to make your own notes!)

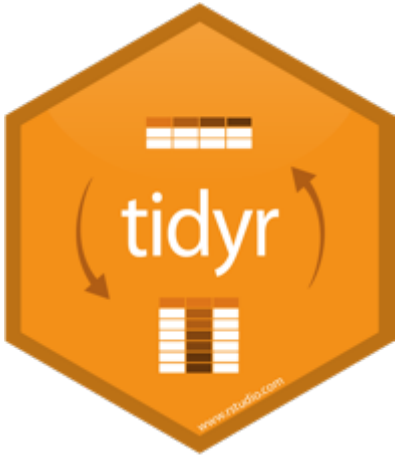
```
> workshop$outline
```

```
> workshop$outline[1:3]
```

`> workshop$outline[1:3]`

DAY 1

Tidy data principles
& `tidyr`



`> workshop$outline[1:3]`

DAY 1

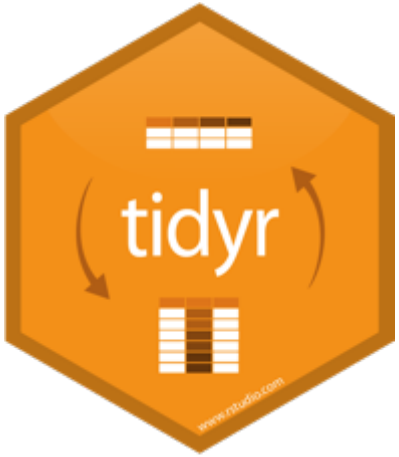
Tidy data principles
& `tidyr`

DAY 2

Manipulating data
& an intro to `dplyr`

DAY 3

Extending your data
with `mutate()`,
`summarise()`
& friends



```
> workshop$outline[-(1:3)]
```

```
> workshop$outline[-(1:3)]
```

2 dialects of R:

`> workshop$outline[-(1:3)]`

2 dialects of R:

`base`

```
$    []    [[]]

apply() which() subset()
```

`> workshop$outline[-(1:3)]`

2 dialects of R:

`base`

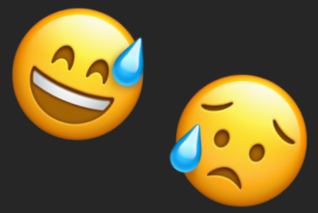
`$` `[]` `[[]]`

`apply()` `which()` `subset()`

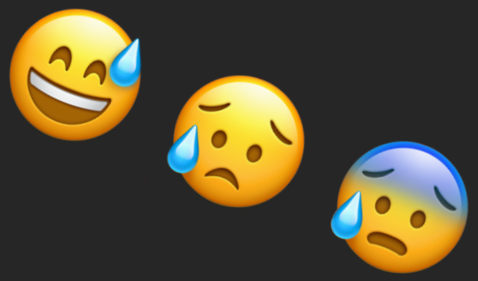
`tidyverse`

```
data <- read.csv("my-data.csv")
```

```r
data <- read.csv("my-data.csv")

data1 <- f(data, arg1 = "something")
```



```
data <- read.csv("my-data.csv")

data1 <- f(data, arg1 = "something")
data2 <- g(data1, another.thing = "blah")
```



```r
data <- read.csv("my-data.csv")

data1 <- f(data, arg1 = "something")
data2 <- g(data1, another.thing = "blah")
data3 <- h(data2, a.setting = TRUE)
```

```r
data <- read.csv("my-data.csv")

data1 <- f(data, arg1 = "something")
data2 <- g(data1, another.thing = "blah")
data3 <- h(data2, a.setting = TRUE)
data4 <- data3[data3$a.column == "cough", ]
```

```
data <- read.csv("my-data.csv")

data1 <- f(data, arg1 = "something")
data2 <- g(data1, another.thing = "blah")
data3 <- h(data2, a.setting = TRUE)
data4 <- data3[data3$a.column == "cough", ]
```

```
data <- read.csv("my-data.csv")
```

```r
data <- read.csv("my-data.csv")

data <-

        data
```

```r
data <- read.csv("my-data.csv")

data <-

      f(data, arg1 = "something")
```

```r
data <- read.csv("my-data.csv")

data <-

    g(
      f(data, arg1 = "something"),
      another.thing = "blah"
    )
```

```r
data <- read.csv("my-data.csv")

data <-
  h(
    g(
      f(data, arg1 = "something"),
      another.thing = "blah"
    ),
    a.setting = TRUE
  )
```

```r
data <- read.csv("my-data.csv")

data <-
  h(
    g(
      f(data, arg1 = "something"),
      another.thing = "blah"
    ),
    a.setting = TRUE
  )
```

```r
data <- read.csv("my-data.csv")

data <-
  h(
    g(
      f(data, arg1 = "something"),
      another.thing = "blah"
    ),
    a.setting = TRUE
  )
```

```r
data <- read.csv("my-data.csv")

data <-
  h(
    g(
      f(data, arg1 = "something"),
      another.thing = "blah"
    ),
    a.setting = TRUE
  )
```

```r
data <- read.csv("my-data.csv")

data <-
  h(
    g(
      f(data, arg1 = "something"),
      another.thing = "blah"
    ),
    a.setting = TRUE
  )
```

```
data <- read.csv("my-data.csv")

data <-
  h(
    g(
      f(data, arg1 = "something"),
      another.thing = "blah"
    ),
    a.setting = TRUE
  )
```

```
data <- data[data$a.column == "cough", ]
```

Solution: the pipe!

%>%

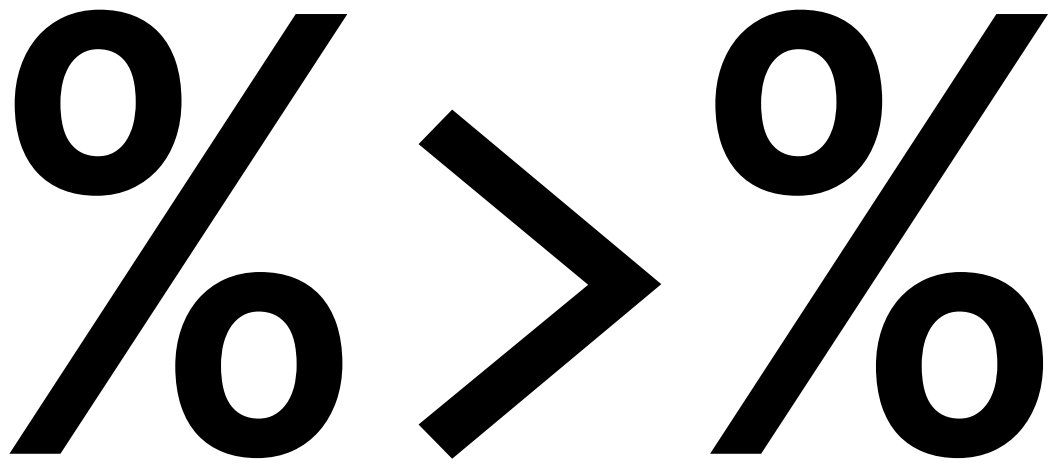
Solution: the pipe!

%>%

{ } [ ] [[ ]] <- = ( ) , " " ' '

Read: “then”

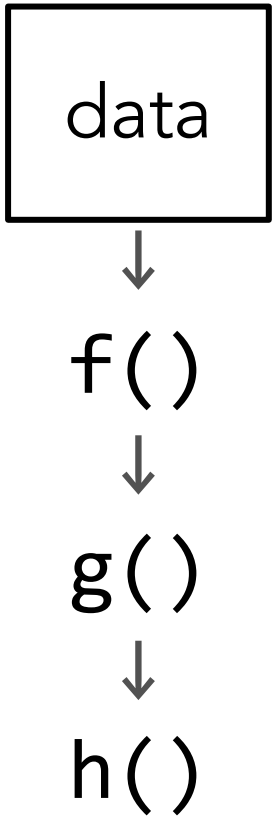
Solution: the pipe!

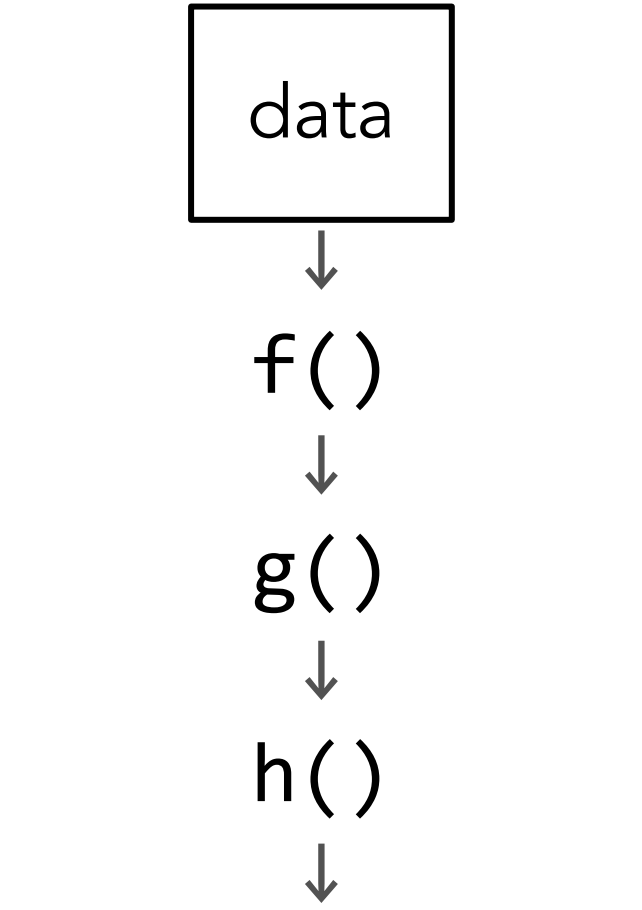


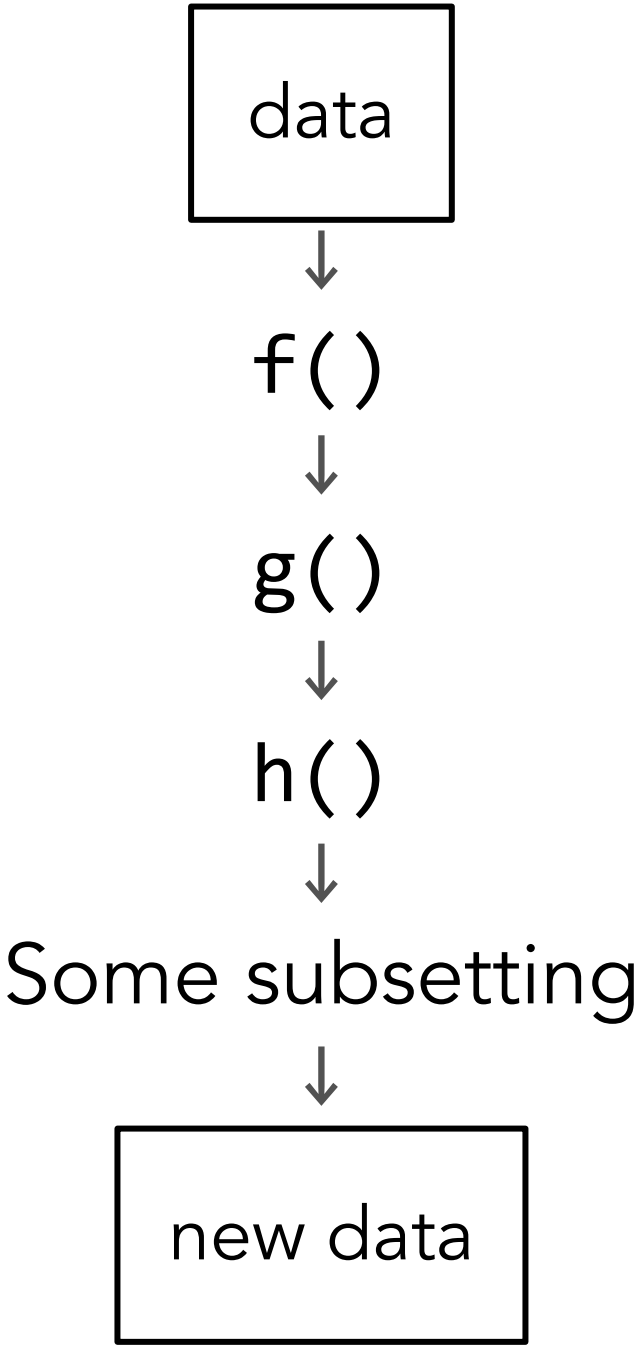
{ } [ ] [[ ]] <- = ( ) , " " ' '

```
data <- read.csv("my-data.csv")

data1 <- f(data, arg1 = "something")
data2 <- g(data1, another.thing = "blah")
data3 <- h(data2, a.setting = TRUE)
data4 <- data3[data3$a.column == "cough", ]
```

data


data
f()
g()
h()


data
f()
g()
h()
Some subsetting


data
f()
g()
h()
Some subsetting
new data

$f(x)$

```
f(x)
```

```
sort(1:10)
```

```
f(x)
```

```
sort(1:10)
```

```
x %>% f()
```

```
f(x)
```

```
x %>% f()
```

```
sort(1:10)
```

```
1:10 %>% sort()
```

```
f(x, y)
```

```
t.test(data$x, data$y)
```

```r
f(x, y)
```

```r
t.test(data$x, data$y)
```

```r
x %>% f(y)
```

```r
data$x %>% t.test(data$y)
```

```
data <- read.csv("my-data.csv")

data1 <- f(data, arg1 = "something")
data2 <- g(data1, another.thing = "blah")
data3 <- h(data2, a.setting = TRUE)
data4 <- data3[data3$a.column == "cough", ]
```

```
data <- read.csv("my-data.csv")

data <-
  h(
    g(
      f(data, arg1 = "something"),
      another.thing = "blah"
    ),
    a.setting = TRUE
  )
```

```
data <- data[data$a.column == "cough", ]
```

```
h(g(f(x)))
```

```
h(g(f(x)))
```

```
x %>%
```

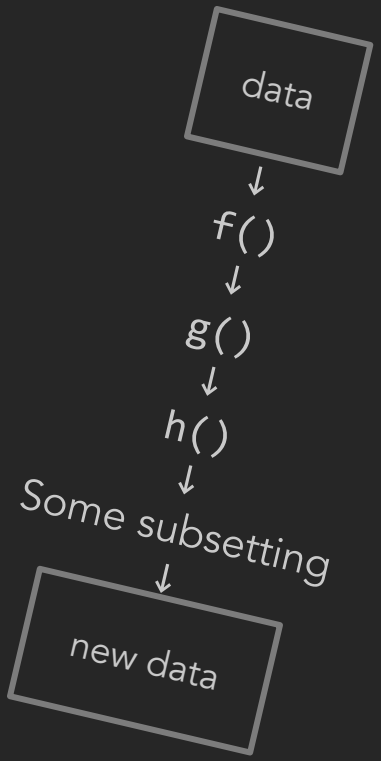
```
h(g(f(x)))

x %>%
   f() %>%
```

```
h(g(f(x)))
```

```
x %>%
  f() %>%
  g() %>%
```

```
h(g(f(x)))
```

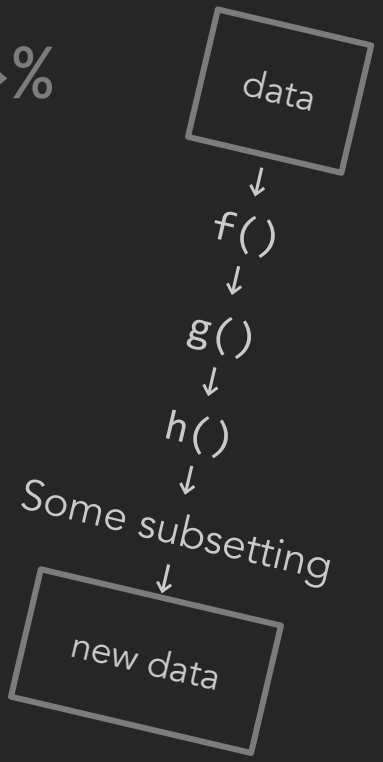
```
x %>%
  f() %>%
  g() %>%
  h()
```



```r
data <- read.csv("my-data.csv")

data <-
  h(
    g(
      f(data, arg1 = "something"),
      another.thing = "blah"
    ),
    a.setting = TRUE
  )
```

```r
data <- read.csv("my-data.csv")

data <- data %>%
  f(arg1 = "something") %>%
  g(another.thing = "blah") %>%
  h(a.setting = TRUE)
```

```r
data <- read.csv("my-data.csv")

data <- data %>%
  f(arg1 = "something") %>%
  g(another.thing = "blah") %>%
  h(a.setting = TRUE)
```



```r
data <- read.csv("my-data.csv")

data <- data %>%
  f(arg1 = "something") %>%
  g(another.thing = "blah") %>%
  h(a.setting = TRUE)
```

```r
data <- data[data$a.column == "cough", ]
```

???????

`> workshop$outline[1:3]`

DAY 1

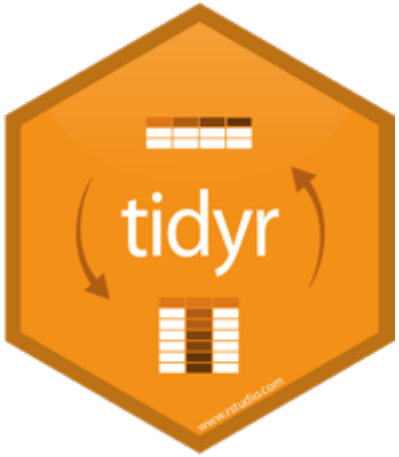
Tidy data principles
& `tidyr`

DAY 2

Manipulating data
& an intro to `dplyr`

DAY 3

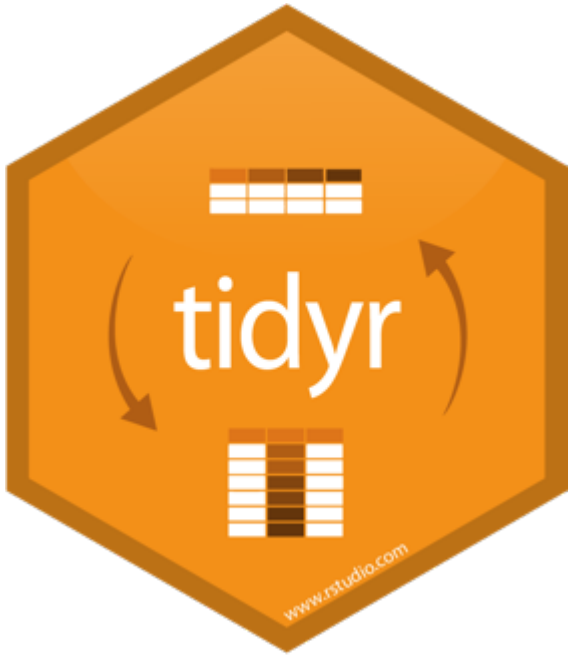
Extending your data
with `mutate()`,
`summarise()`
& friends



`> workshop$outline[[1]]`

DAY 1

Tidy data principles
& `tidyr`



`> workshop$outline[[1]]`

DAY 1

Tidy data principles
& `tidyr`

A motivating example…

Kogelberg NR,
R. van Mazijk

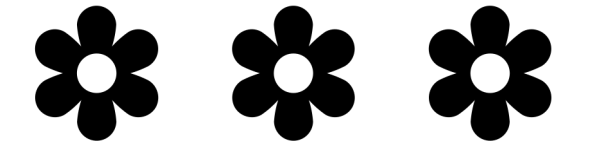
Observation Pk,
R. van Mazijk 2018

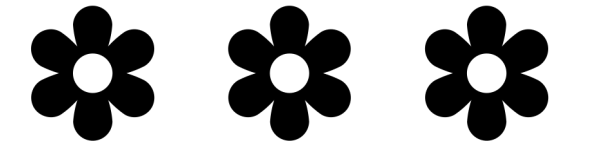
Near Pearly Beach, Agulhas Plains,
R. van Mazijk 2018

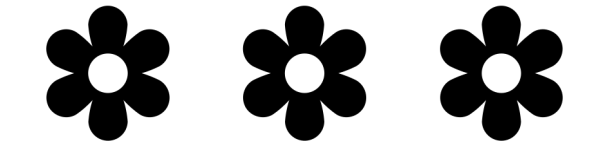
An example data-collection scenario in biology

An example data-collection scenario in biology

An example data-collection scenario in biology



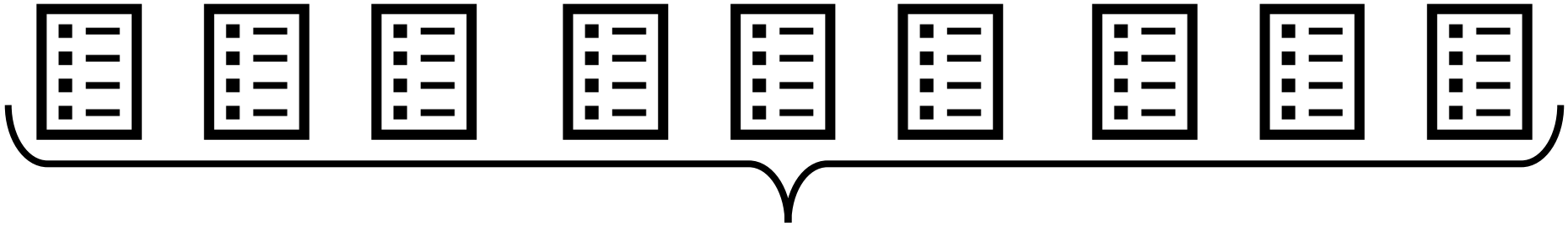


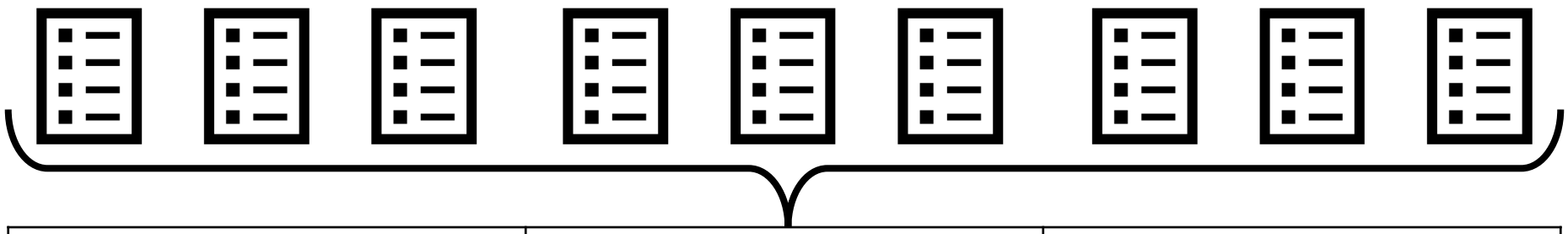


An example data-collection scenario in biology

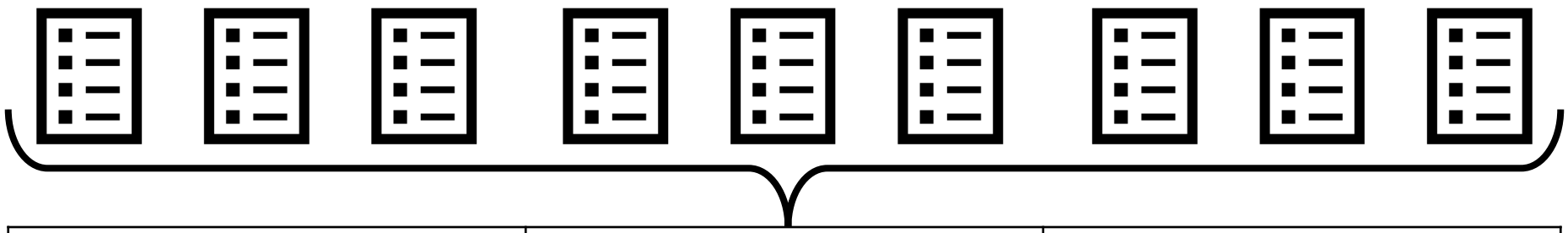
An example data-collection scenario in biology

(A good way to *collect* your data!)



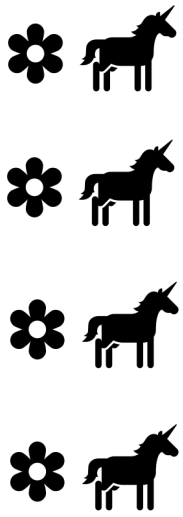
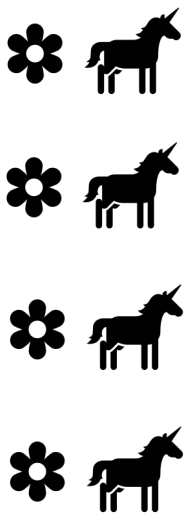
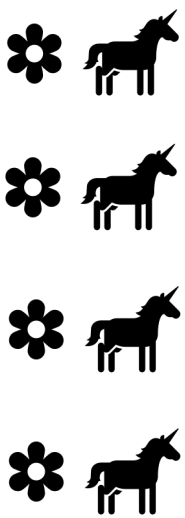
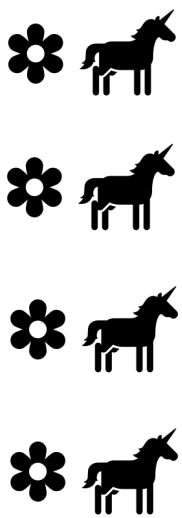
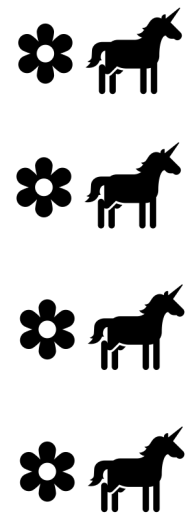
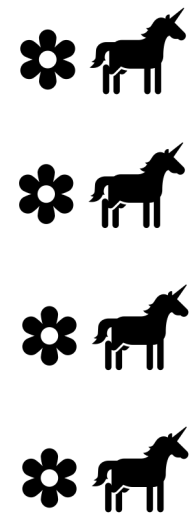
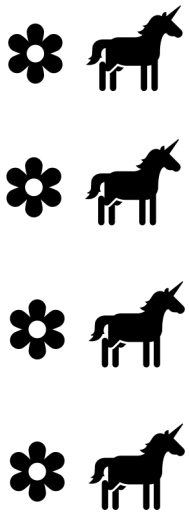
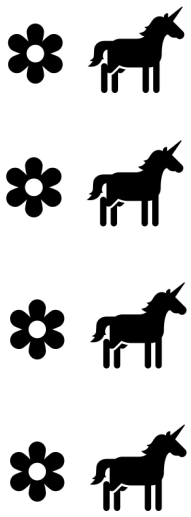
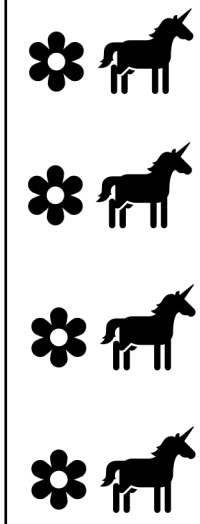


Site 1			Site 2			Site 3		
Sp 1	Sp 2	Sp 3	Sp 1	Sp 2	Sp 3	Sp 1	Sp 2	Sp 3

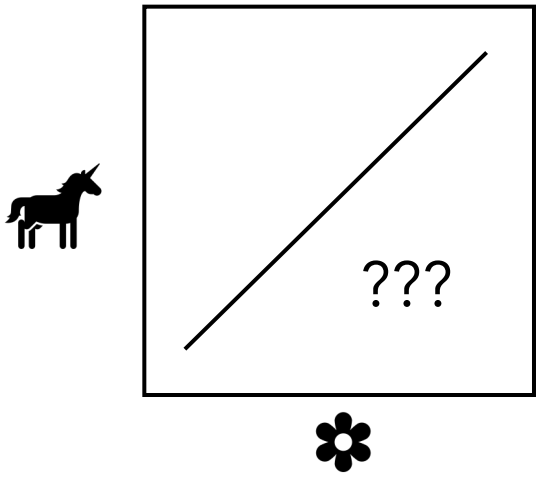


Site 1			Site 2			Site 3		
Sp 1	Sp 2	Sp 3	Sp 1	Sp 2	Sp 3	Sp 1	Sp 2	Sp 3

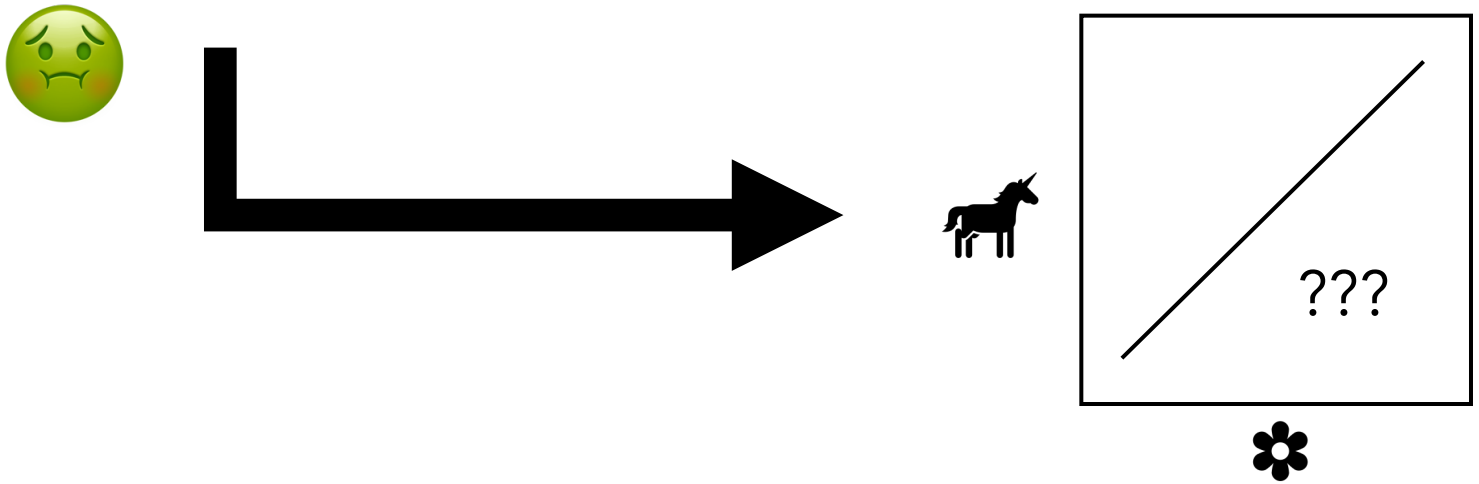
One way to lay out your collected data…

Site 1			Site 2			Site 3		
Sp 1	Sp 2	Sp 3	Sp 1	Sp 2	Sp 3	Sp 1	Sp 2	Sp 3
								

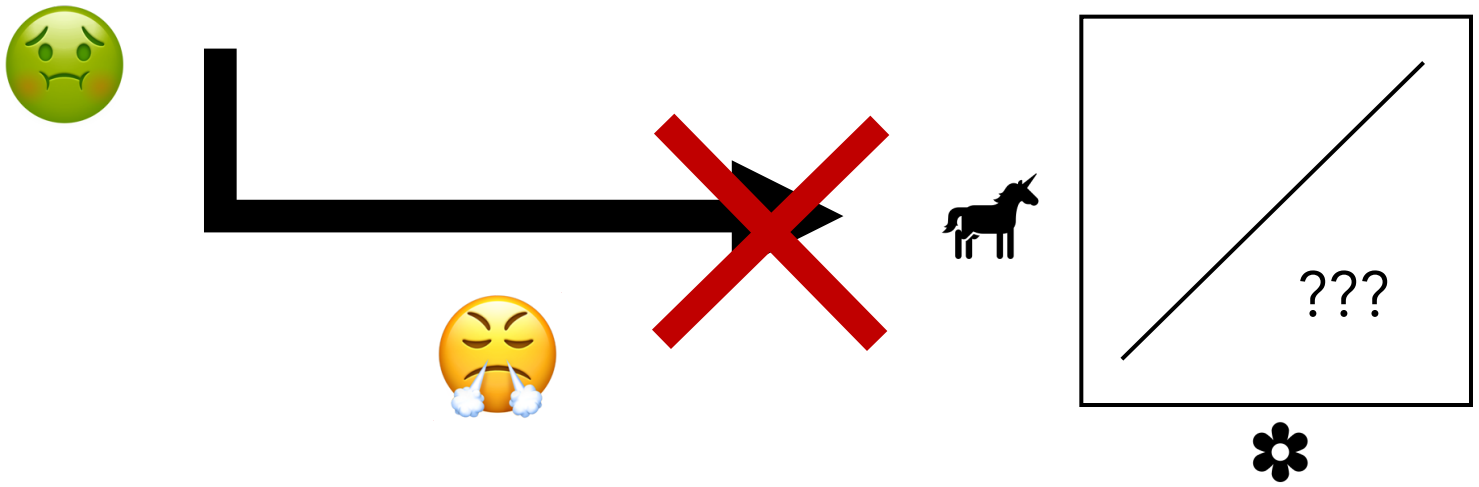
Site 1			Site 2			Site 3		
Sp 1	Sp 2	Sp 3	Sp 1	Sp 2	Sp 3	Sp 1	Sp 2	Sp 3



Site 1			Site 2			Site 3		
Sp 1	Sp 2	Sp 3	Sp 1	Sp 2	Sp 3	Sp 1	Sp 2	Sp 3



Site 1			Site 2			Site 3		
Sp 1	Sp 2	Sp 3	Sp 1	Sp 2	Sp 3	Sp 1	Sp 2	Sp 3



	Site 1		Site 2		Site 3	
Sp						

Another way…

Sp	Site	✿	

The “best” way.
(Will make your life easiest in the long-term.)

TIDY DATA

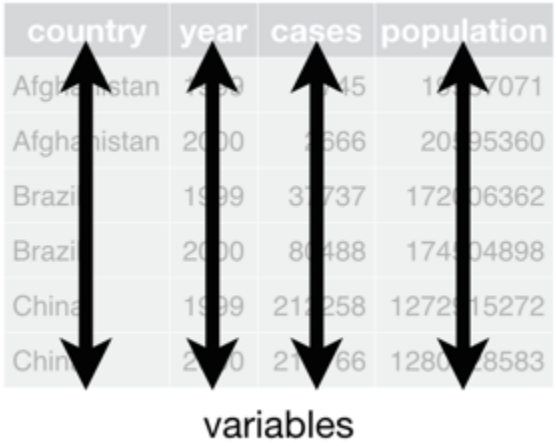
Sp	Site	✿	

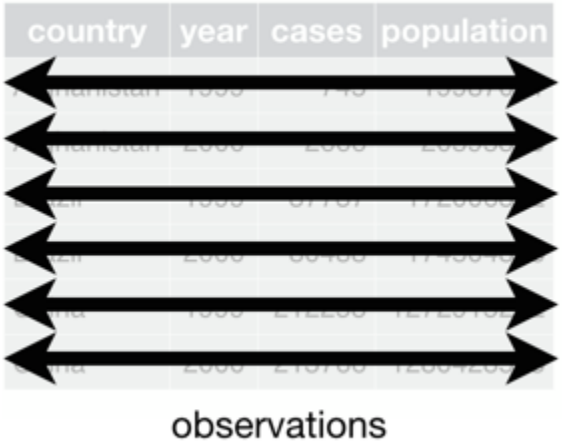
The “best” way.
(Will make your life easiest in the long-term.)

TIDY DATA

CC BY-NC-ND 3.0 Grolemund & Wickham 2017. *R for Data Science*

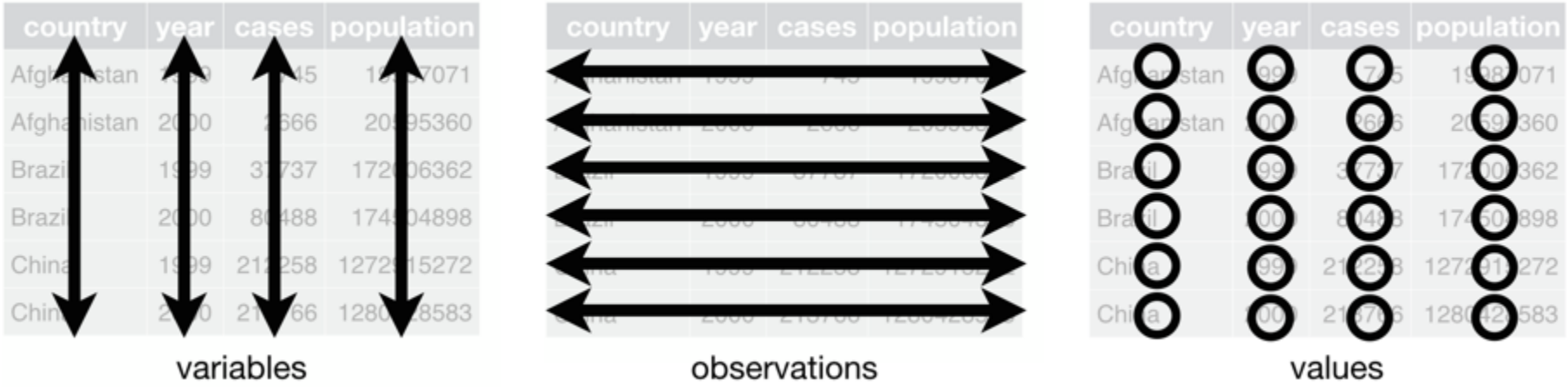
TIDY DATA





CC BY-NC-ND 3.0 Grolemund & Wickham 2017. *R for Data Science*

TIDY DATA

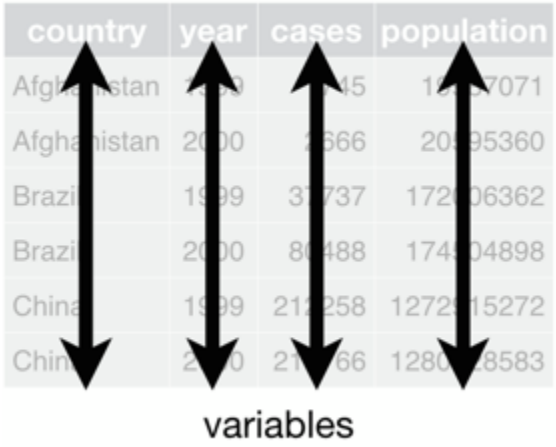


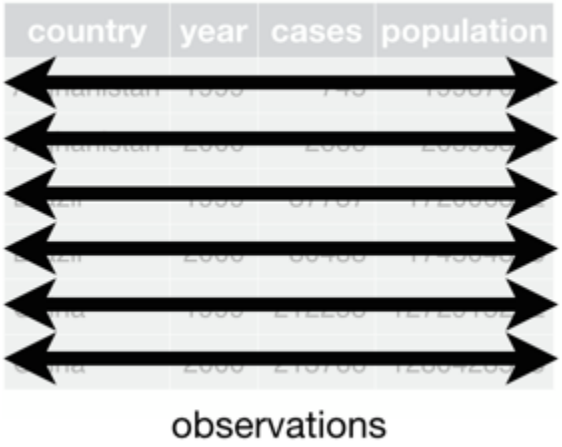
1. Each **variable** must have its own **column**

2. Each **observation** must have its own **row**

3. Each **value**, therefore, must have its own **cell**

CC BY-NC-ND 3.0 Grolemund & Wickham 2017. *R for Data Science*





`tidyr`

An R-package all about getting to this:

CC BY-NC-ND 3.0 Grolemund & Wickham 2017. *R for Data Science*

Verbs to tidy your data

Verbs to tidy your data

```
# Untidy observations?
gather()    # if > 1 observation per row
spread()    # if observations live in > 1 row
```

Verbs to tidy your data

```
# Untidy observations?
gather()    # if > 1 observation per row
spread()    # if observations live in > 1 row

# Untidy variables?
separate()  # if > 1 variable per column
unite()     # if variables live in > 1 column
```

Note the following when choosing `tidyr`-verbs:

Note the following when choosing `tidyr`-verbs:

- Be clear on what your **observations** are:
  - Like, what **unit** of your study "counts" as an observation
    - E.g. Leaf traits: plant leaf vs plant individual
    - E.g. Reproductive success: egg size vs clutch size

Note the following when choosing `tidyr`-verbs:

- Be clear on what your **observations** are:
  - Like, what **unit** of your study "counts" as an observation
    - E.g. Leaf traits: plant leaf vs plant individual
    - E.g. Reproductive success: egg size vs clutch size
  - This will depend on your study &/or data!

Note the following when choosing `tidyr`-verbs:

- Be clear on what your **observations** are:
  - Like, what **unit** of your study "counts" as an observation
    - E.g. Leaf traits: plant leaf vs plant individual
    - E.g. Reproductive success: egg size vs clutch size
  - This will depend on your study &/or data!

- **Variables** are discrete, separate ideas!

Note the following when choosing `tidyr`-verbs:

- Be clear on what your **observations** are:
  - Like, what **unit** of your study "counts" as an observation
    - E.g. Leaf traits: plant leaf vs plant individual
    - E.g. Reproductive success: egg size vs clutch size
  - This will depend on your study &/or data!

- **Variables** are discrete, separate ideas!
  - But again, this will depend on your study &/or data!

```
> demo()
```