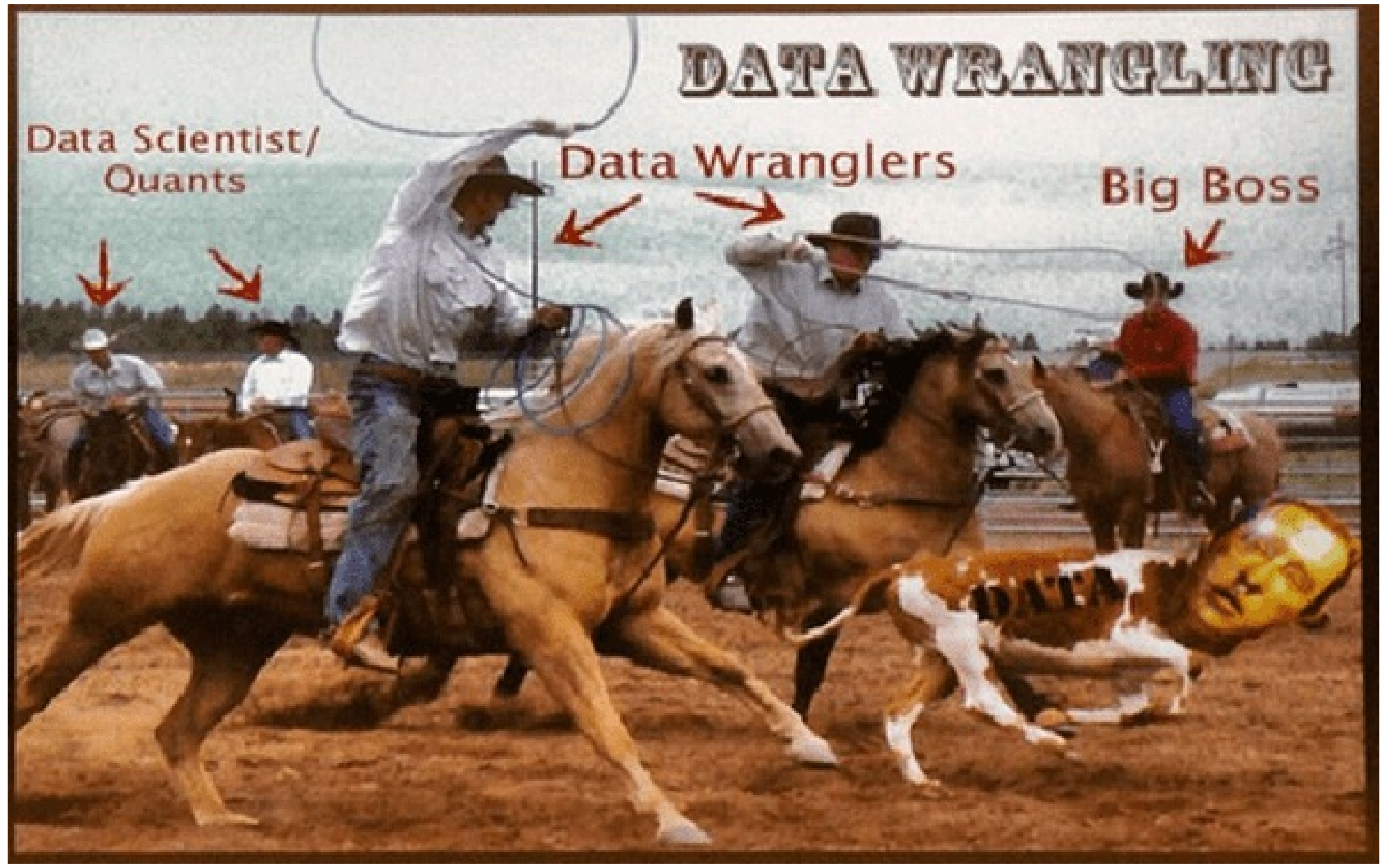
Wrangling

R for Data Science

Basel R Bootcamp

February 2019

What is wrangling?



from datasciencebe.com

This is wrangling!

Transform

Change variable names

Add new variables

Organise

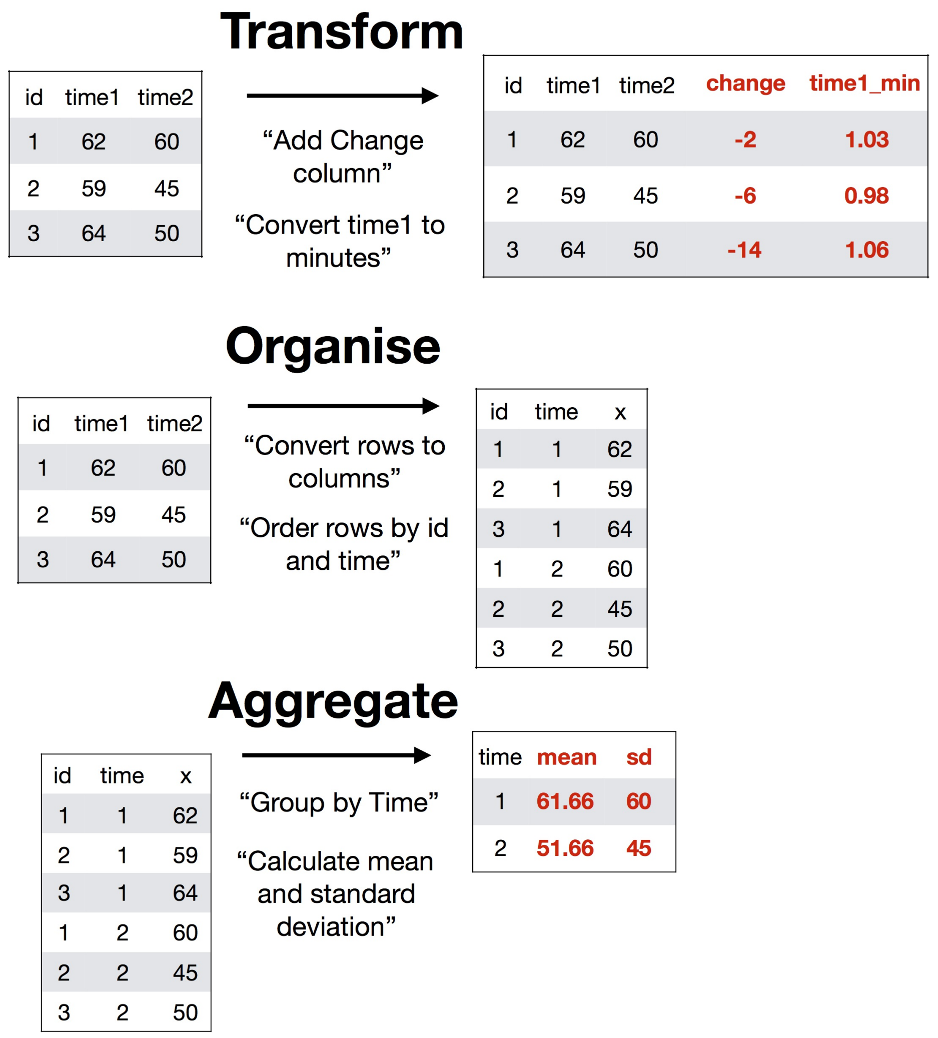
Sort data by variables

Merging data from two separate dataframes

Move data between variables and rows

Aggregate and summarise

Group data and calculate and summary stats



Transform

id	time1	time2
1	62	60
2	59	45
3	64	50

$\longrightarrow$ “Add Change column” “Convert time1 to minutes”

id	time1	time2	change	time1_min
1	62	60	-2	1.03
2	59	45	-6	0.98
3	64	50	-14	1.06

Organise

id	time1	time2
1	62	60
2	59	45
3	64	50

$\longrightarrow$ “Convert rows to columns” “Order rows by id and time”

id	time	x
1	1	62
2	1	59
3	1	64
1	2	60
2	2	45
3	2	50

Aggregate

id	time	x
1	1	62
2	1	59
3	1	64
1	2	60
2	2	45
3	2	50

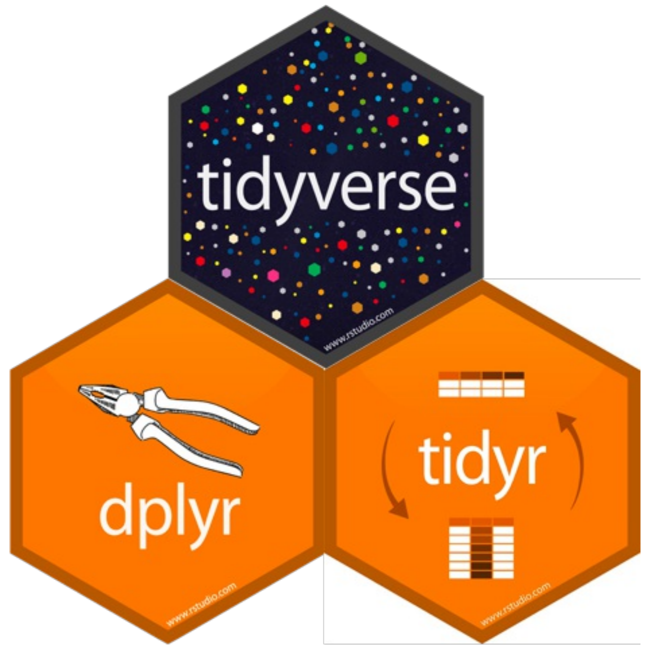
$\longrightarrow$ “Group by Time” “Calculate mean and standard deviation”

time	mean	sd
1	61.66	60
2	51.66	45

dplyr + tidyr

To wrangle data in R, we will use the `dplyr` and `tidyr` packages, which are part of the `tidyverse`.

Package	Function	Function
dplyr	Transformation	`rename()`, `mutate()`, `case_when()`, `*_join()`
dplyr	Organisation	`arrange()`, `slice()`, `filter()`, `select()`
tidyr	Organisation	`gather()`, `spread()`
dplyr	Aggregation	`group_by()`, `summarise()`



The Pipe! %>%

dplyr makes extensive use of a new operator called the %>%

Read the %>% as "And Then..."

```
# Start with data
data %>% # AND THEN...

DO_SOMETHING %>% # AND THEN...

DO_SOMETHING %>% # AND THEN...

DO_SOMETHING %>% # AND THEN...
```

%>%

The Pipe! %>%

```r
#  Vector of `scores`
score <- c(8, 4, 6, 3, 7, 3)
score
```

```
## [1] 8 4 6 3 7 3
```

```r
# Mean: Base-R-way
mean(score)
```

```
## [1] 5.167
```

```r
# Mean: Tidyverse-style (with %>%)
score %>%  # AND THEN
  mean()
```

```
## [1] 5.167
```

FUN(OBJECT, ...)

Is the same thing as...

OBJECT %>% **FUN(___ , ...)**

The **OBJECT** to the left of the pipe %>% becomes the first argument to the **FUN( )** to the right of the pipe

The Pipe! %>%

```r
#  Vector of `scores`
score <- c(8, 4, 6, 3, 7, 3)
score
```

```
## [1] 8 4 6 3 7 3
```

```r
# Mean: Base-R-way
round(mean(score), digits = 1)
```

```
## [1] 5.2
```

```r
# Mean: Tidyverse-style (with %>%)
score %>%     # AND THEN
  mean() %>%  # AND THEN
  round(digits = 1)
```

```
## [1] 5.2
```

FUN(OBJECT, ...)

Is the same thing as...

OBJECT %>% **FUN(___ , ...)**

The **OBJECT** to the left of the pipe %>% becomes the first argument to the **FUN( )** to the right of the pipe

2 dirty data sets

Goals

1 - Give meaningful **variable names**.

2 - Use appropriate **units** and **labels** for nominal variables.

3 - **Combine** datasets.

4 - **Sort** tibble by age.

5 - Select relevant **cases**.

6 - Select relevant **variables**.

7 - Change to **long format**.

```
# patients tibble
patients
```

```
## # A tibble: 5 x 3
##       id    X1    X2
##    <dbl> <dbl> <dbl>
## 1      1    37     1
## 2      2    65     2
## 3      3    57     2
## 4      4    34     1
## 5      5    45     2
```

```
# results tibble
results
```

```
## # A tibble: 5 x 3
##       id   t_1   t_2
##    <dbl> <dbl> <dbl>
## 1      4   100   105
## 2     92   134   150
## 3      1   123   135
## 4      2   143   140
## 5     99   102    68
```

Transformation

Transformation functions are used to **alter the content** of a `tibble`.

Function	Description
`rename()`	**Change names** of variables
`mutate()`	**Create variable** from existing variables
`case_when()`	**Recode values** from a vector to another
`left_join()`	**Combine tibbbles**

```
patients    # patients data
```

```
## # A tibble: 5 x 3
##        id    X1    X2
##     <dbl> <dbl> <dbl>
## 1       1    37     1
## 2       2    65     2
## 3       3    57     2
## 4       4    34     1
## 5       5    45     2
```

rename()

Change variable names with rename().

```
patients %>%
  rename(NEW = OLD,
         NEW = OLD)
```

```
patients   # Original
```

```
## # A tibble: 5 x 3
##      id    X1    X2
##   <dbl> <dbl> <dbl>
## 1     1    37     1
## 2     2    65     2
## 3     3    57     2
## 4     4    34     1
## 5     5    45     2
```

Change X1 to age, and X2 to arm.

```
# 0) Start with patients data
patients %>%

# 1) Change variable names with rename()
  rename(age = X1,   # New = Old
         arm = X2)   # New = Old
```

```
## # A tibble: 5 x 3
##      id   age   arm
##   <dbl> <dbl> <dbl>
## 1     1    37     1
## 2     2    65     2
## 3     3    57     2
## 4     4    34     1
## 5     5    45     2
```

mutate()

Create **new variables**, or **change existing ones**, with `mutate()`.

```
tibble %>%
  mutate(
   NEW1 = DEFINITION1,
   NEW2 = DEFINITION2,
   NEW3 = DEFINITION3,
   ...
  )
```

Calculate two new variables `age_months` and `age_decades`.

```
patients %>%

  rename(age = X1,
         arm = X2) %>%    # AND THEN...

# Create new variables with mutate()
  mutate(age_months = age * 12,
         age_decades = age / 10)
```

```
## # A tibble: 5 x 5
##      id   age   arm age_months age_decades
##   <dbl> <dbl> <dbl>      <dbl>       <dbl>
## 1     1    37     1        444         3.7
## 2     2    65     2        780         6.5
## 3     3    57     2        684         5.7
## 4     4    34     1        408         3.4
## 5     5    45     2        540         4.5
```

case_when()

Use `case_when()` with `mutate()` to define **new variables based on logical conditions**.

```
# Using mutate(case_when())
tibble %>%
  mutate(
    NEW = case_when(
      COND1 ~ VAL1,
      COND2 ~ VAL2
    ))
```

Create `arm_lab` that carries `'placebo'` for `arm == 1` and `'drug'` for `arm == 2`.

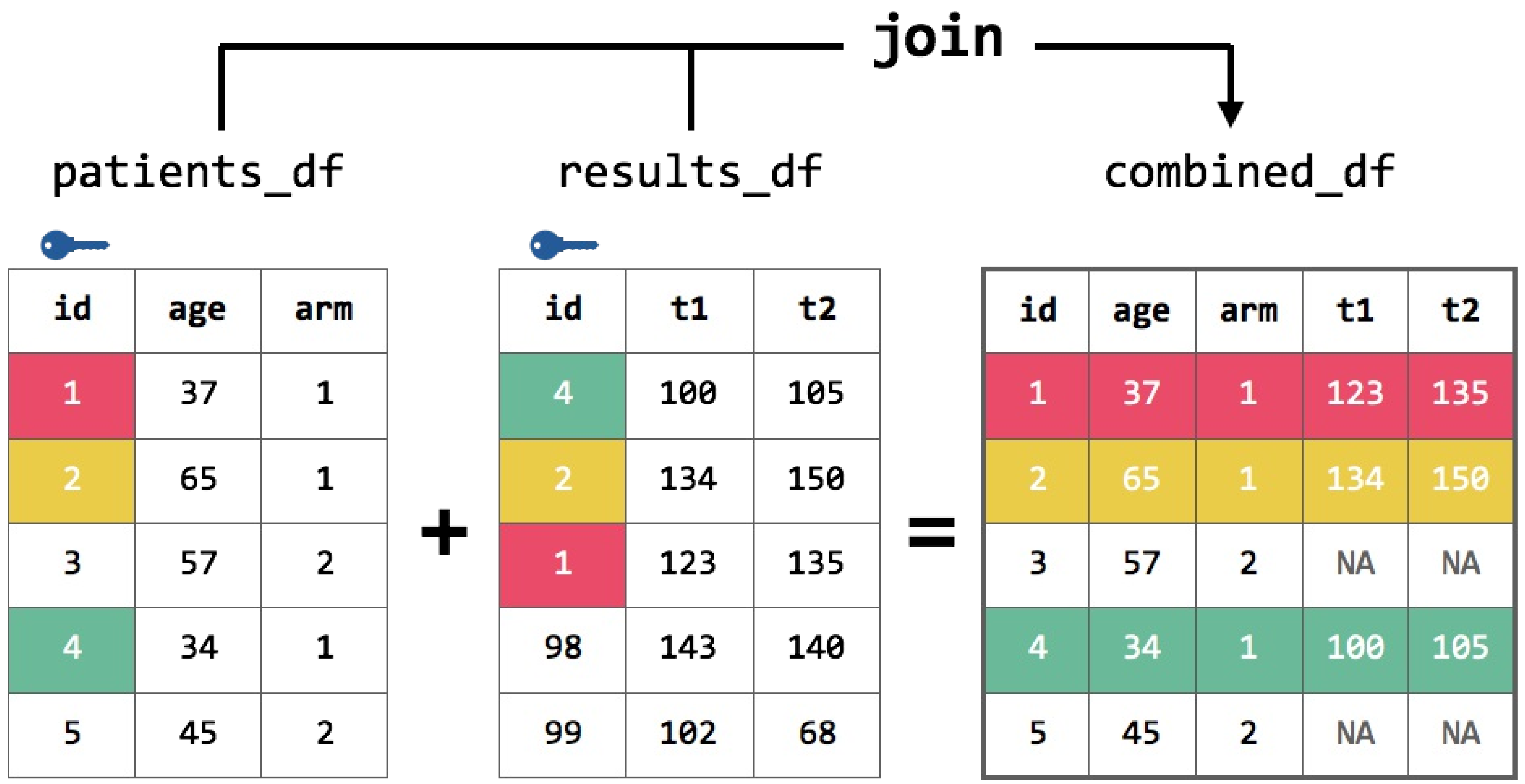
```
patients %>%

  rename(age = X1,
         arm = X2) %>%

  # Create arm_lab from arm
  mutate(arm_lab = case_when(arm == 1 ~ "placebo",
                             arm == 2 ~ "drug"))
```

```
## # A tibble: 5 x 4
##      id   age   arm arm_lab
##   <dbl> <dbl> <dbl> <chr>
## 1     1    37     1 placebo
## 2     2    65     2 drug
## 3     3    57     2 drug
## 4     4    34     1 placebo
## 5     5    45     2 drug
```

Joining data



patients_df

id	age	arm
1	37	1
2	65	1
3	57	2
4	34	1
5	45	2

results_df

id	t1	t2
4	100	105
2	134	150
1	123	135
98	143	140
99	102	68

combined_df

id	age	arm	t1	t2
1	37	1	123	135
2	65	1	134	150
3	57	2	NA	NA
4	34	1	100	105
5	45	2	NA	NA

left_join()

Use left_join() to **combine two data frames** based on one or more key variables.

```
# Join tibble_2 to tibble_1
#   matched by KEY
tibble_1 %>%
  left_join(tibble_2,
      by = c("KEY"))
```

Other *_join() functions: right_join(), full_join(), inner_join(), anti_join(), semi_join().

```
# Join patients with results to create combined
combined <- patients %>%
  rename(age = X1, arm = X2) %>%
  mutate(arm_lab = case_when(arm == 1 ~ "placebo",
                             arm == 2 ~ "drug")) %>%

# Join with results with left_join()
  left_join(results, by = "id")
```

```
# Show combined data set
combined
```

```
## # A tibble: 5 x 6
##      id   age   arm arm_lab   t_1   t_2
##   <dbl> <dbl> <dbl> <chr>   <dbl> <dbl>
## 1     1    37     1 placebo   123   135
## 2     2    65     2 drug      143   140
## 3     3    57     2 drug       NA    NA
## 4     4    34     1 placebo   100   105
## 5     5    45     2 drug       NA    NA
```

Organisation

Organisation functions help you change the organisation of your data by **sorting rows** by variables, **filter rows** based on criteria, **select variables** (etc).

Function	Description
`arrange()`	**Sort rows** by variables
`slice()`	**Select rows** by location
`filter()`	**Select rows** by criteria
`select()`	**Select variables**

```
# combined tibble
combined
```

```
## # A tibble: 5 x 6
##      id   age   arm arm_lab   t_1   t_2
##   <dbl> <dbl> <dbl> <chr>   <dbl> <dbl>
## 1     1    37     1 placebo   123   135
## 2     2    65     2 drug      143   140
## 3     3    57     2 drug       NA    NA
## 4     4    34     1 placebo   100   105
## 5     5    45     2 drug       NA    NA
```

arrange()

Use `arrange()` to **sort rows** in increasing or decreasing (using `desc()`) order of one or more variables.

```
tibble %>%
  arrange(A, B)
```

To sort in descending order, use `desc()`

```
tibble %>%
  arrange(desc(A), B)
```

Sort by `arm`.

```
combined %>%
  arrange(arm)    # Sort by arm
```

```
## # A tibble: 5 x 6
##      id   age   arm arm_lab   t_1   t_2
##   <dbl> <dbl> <dbl> <chr>   <dbl> <dbl>
## 1     1    37     1 placebo   123   135
## 2     4    34     1 placebo   100   105
## 3     2    65     2 drug      143   140
## 4     3    57     2 drug       NA    NA
## 5     5    45     2 drug       NA    NA
```

arrange()

Use `arrange()` to **sort rows** in increasing or decreasing (using `desc()`) order of one or more variables.

```
tibble %>%
  arrange(A, B)
```

To sort in descending order, use `desc()`

```
tibble %>%
  arrange(desc(A), B)
```

Sort by `arm` and then `age`.

```
combined %>%
  arrange(arm, age)  # Sort by arm then age
```

```
## # A tibble: 5 x 6
##      id   age   arm arm_lab   t_1   t_2
##   <dbl> <dbl> <dbl> <chr>   <dbl> <dbl>
## 1     4    34     1 placebo   100   105
## 2     1    37     1 placebo   123   135
## 3     5    45     2 drug       NA    NA
## 4     3    57     2 drug       NA    NA
## 5     2    65     2 drug      143   140
```

slice()

Use `slice()` to **select rows** (or remove) by row number.

Use `c()`, `a:b`, or `seq()` to create row numbers

```
# Integer vector
c(2, 6, 10)
```

```
## [1]  2  6 10
```

```
# Integer vector of 0 to 5
0:5
```

```
## [1] 0 1 2 3 4 5
```

Select rows 3 and 5.

```
# Rows 3 and 5 only
combined %>%
  slice(c(3, 5))
```

```
## # A tibble: 2 x 6
##      id   age   arm arm_lab   t_1   t_2
##   <dbl> <dbl> <dbl> <chr>   <dbl> <dbl>
## 1     3    57     2 drug       NA    NA
## 2     5    45     2 drug       NA    NA
```

slice()

Use `slice()` to **select rows** (or remove) by row number.

Use `c()`, `a:b`, or `seq()` to create row numbers

```
# Integer vector
c(2, 6, 10)
```

```
## [1]  2  6 10
```

```
# Integer vector of 0 to 5
0:5
```

```
## [1] 0 1 2 3 4 5
```

Select rows 1 through 4.

```
# First 4 rows
combined %>%
  slice(1:4)
```

```
## # A tibble: 4 x 6
##      id   age   arm arm_lab   t_1   t_2
##   <dbl> <dbl> <dbl> <chr>   <dbl> <dbl>
## 1     1    37     1 placebo   123   135
## 2     2    65     2 drug      143   140
## 3     3    57     2 drug       NA    NA
## 4     4    34     1 placebo   100   105
```

filter()

Use `filter()` to **select rows** (or remove) based on logical statements.

Chain logical comparison operators with & (AND) and | (OR).

== - is equal to

<, > - smaller/greater than

≤, ≥ - smaller/greater than or equal

&, && - logical AND

|, || - logical OR

Select patients over 30.

```
# Filter patients older than 30
combined %>%
  filter(age > 30)
```

```
## # A tibble: 5 x 6
##      id   age   arm arm_lab   t_1   t_2
##   <dbl> <dbl> <dbl> <chr>   <dbl> <dbl>
## 1     1    37     1 placebo   123   135
## 2     2    65     2 drug      143   140
## 3     3    57     2 drug       NA    NA
## 4     4    34     1 placebo   100   105
## 5     5    45     2 drug       NA    NA
```

`filter()`

Use `filter()` to **select rows** (or remove) based on logical statements.

Chain logical comparison operators with & (AND) and | (OR).

`==` - is equal to

`<`, `>` - smaller/greater than

`<=`, `>=` - smaller/greater than or equal

`&`, `&&` - logical AND

`|`, `||` - logical OR

Select patients over `30` given arm is `'drug'`.

```
# Filter patients older than 30 given drug
combined %>%
  filter(age > 30 & arm_lab == "drug")
```

```
## # A tibble: 3 x 6
##      id   age   arm arm_lab   t_1   t_2
##   <dbl> <dbl> <dbl> <chr>   <dbl> <dbl>
## 1     2    65     2 drug      143   140
## 2     3    57     2 drug       NA    NA
## 3     5    45     2 drug       NA    NA
```

select()

Use select() to **select variables** (and remove all others)

```
# Select variables A, B
tibble %>%
  select(A, B)
```

Remove variables with -.

```
# Select everything BUT A
tibble %>%
  select(-A)
```

Select variables id and arm.

```
combined %>%
  select(id, arm) # Select id and arm variables
```

```
## # A tibble: 5 x 2
##      id   arm
##   <dbl> <dbl>
## 1     1     1
## 2     2     2
## 3     3     2
## 4     4     1
## 5     5     2
```

select()

Use `select()` to **select variables** (and remove all others)

```r
# Select variables A, B
tibble %>%
  select(A, B)
```

Remove variables with `-`.

```r
# Select everything BUT A
tibble %>%
  select(-A)
```

Select everything id

```r
combined %>%
  select(-id) # Everything BUT id
```

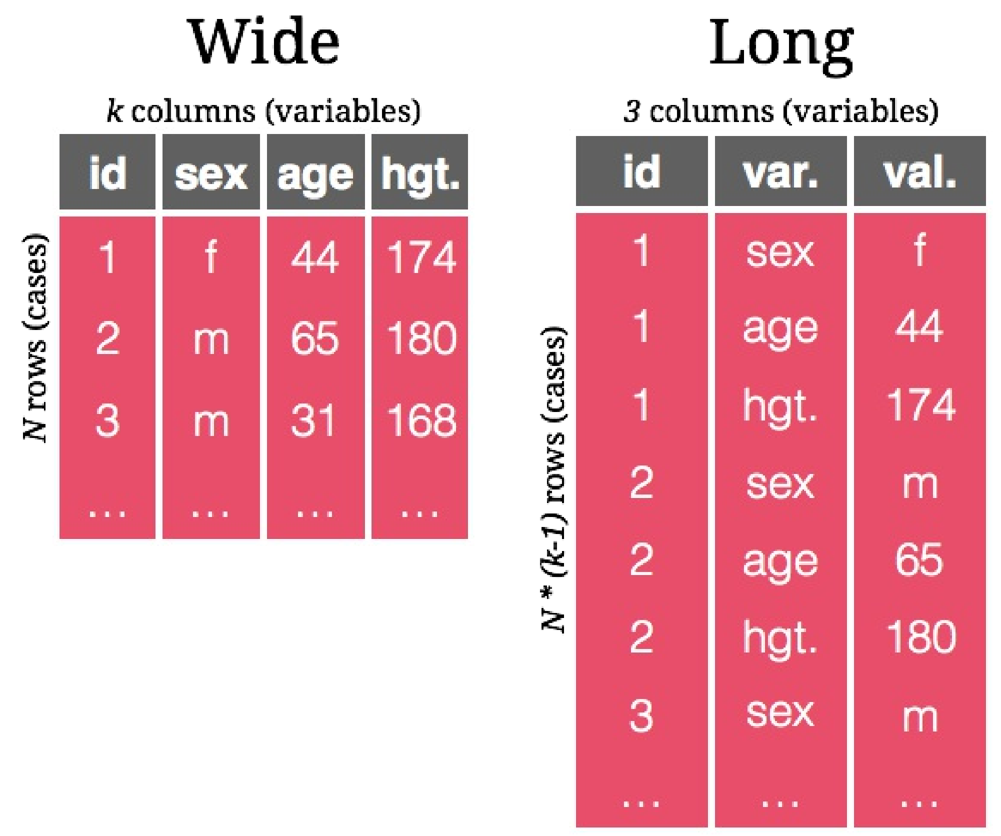
```
## # A tibble: 5 x 5
##     age   arm arm_lab   t_1   t_2
##   <dbl> <dbl> <chr>   <dbl> <dbl>
## 1    37     1 placebo   123   135
## 2    65     2 drug      143   140
## 3    57     2 drug       NA    NA
## 4    34     1 placebo   100   105
## 5    45     2 drug       NA    NA
```

Long and wide formats

Some functions require data to be in a certain shape, that is to be either in a wide or a long format.

Use `gather()` and `spread()` from the `tidyr` package to change a tibble between **wide** and **long** formats.

Function	Result
`gather()`	**wide** → **long** format
`spread()`	**long** → **wide** format



id	sex	age	hgt.
1	f	44	174
2	m	65	180
3	m	31	168
…	…	…	…

id	var.	val.
1	sex	f
1	age	44
1	hgt.	174
2	sex	m
2	age	65
2	hgt.	180
3	sex	m
…	…	…

gather()

```r
# Show original data (wide)
combined
```

```
## # A tibble: 5 x 6
##      id   age   arm arm_lab   t_1   t_2
##   <dbl> <dbl> <dbl> <chr>   <dbl> <dbl>
## 1     1    37     1 placebo   123   135
## 2     2    65     2 drug      143   140
## 3     3    57     2 drug       NA    NA
## 4     4    34     1 placebo   100   105
## 5     5    45     2 drug       NA    NA
```

```r
# "Gather" wide data to long
combined %>%
  gather(time,  # New group variable
         value, # New target variable
         -id)   # Omit id
```

```
## # A tibble: 25 x 3
##       id time  value
##    <dbl> <chr> <chr>
##  1     1 age   37
##  2     2 age   65
##  3     3 age   57
##  4     4 age   34
##  5     5 age   45
##  6     1 arm   1
##  7     2 arm   2
##  8     3 arm   2
##  9     4 arm   1
## 10     5 arm   2
## # … with 15 more rows
```

gather()

```r
# Show original data (wide)
combined
```

```
## # A tibble: 5 x 6
##      id   age   arm arm_lab   t_1   t_2
##   <dbl> <dbl> <dbl> <chr>   <dbl> <dbl>
## 1     1    37     1 placebo   123   135
## 2     2    65     2 drug      143   140
## 3     3    57     2 drug       NA    NA
## 4     4    34     1 placebo   100   105
## 5     5    45     2 drug       NA    NA
```

```r
# "Gather" wide data to long
combined %>%
  gather(time,  # New group variable
         value, # New target variable
         -id, -age, -arm, -arm_lab) # Omit var
```

```
## # A tibble: 10 x 6
##       id   age   arm arm_lab time  value
##    <dbl> <dbl> <dbl> <chr>   <chr> <dbl>
##  1     1    37     1 placebo t_1     123
##  2     2    65     2 drug    t_1     143
##  3     3    57     2 drug    t_1      NA
##  4     4    34     1 placebo t_1     100
##  5     5    45     2 drug    t_1      NA
##  6     1    37     1 placebo t_2     135
##  7     2    65     2 drug    t_2     140
##  8     3    57     2 drug    t_2      NA
##  9     4    34     1 placebo t_2     105
## 10     5    45     2 drug    t_2      NA
```

spread()

```
# Show long data
combined %>%
  gather(time,  # New group variable
         value, # New target variable
         -id, -age, -arm, -arm_lab) # Omit var
```

```
## # A tibble: 10 x 6
##       id   age   arm arm_lab time  value
##    <dbl> <dbl> <dbl> <chr>   <chr> <dbl>
##  1     1    37     1 placebo t_1     123
##  2     2    65     2 drug    t_1     143
##  3     3    57     2 drug    t_1      NA
##  4     4    34     1 placebo t_1     100
##  5     5    45     2 drug    t_1      NA
##  6     1    37     1 placebo t_2     135
##  7     2    65     2 drug    t_2     140
##  8     3    57     2 drug    t_2      NA
##  9     4    34     1 placebo t_2     105
## 10     5    45     2 drug    t_2      NA
```

```
# "Gather" wide data to long
long_combined = combined %>%
  gather(time,  # New group variable
         value, # New target variable
         -id, -age, -arm, -arm_lab) # Omit var

# "Spread" long data to wide
long_combined %>%
  spread(time,   # Old group variable
         value)  # Old target variable
```

```
## # A tibble: 5 x 6
##      id   age   arm arm_lab   t_1   t_2
##   <dbl> <dbl> <dbl> <chr>   <dbl> <dbl>
## 1     1    37     1 placebo   123   135
## 2     2    65     2 drug      143   140
## 3     3    57     2 drug       NA    NA
## 4     4    34     1 placebo   100   105
## 5     5    45     2 drug       NA    NA
```

Practical