# Intro to R for Biologists

## IBiS Special Topics, Fall 2021
## Class 5: Oct. 7, 2021

**Erik Andersen and Shelby Blythe**

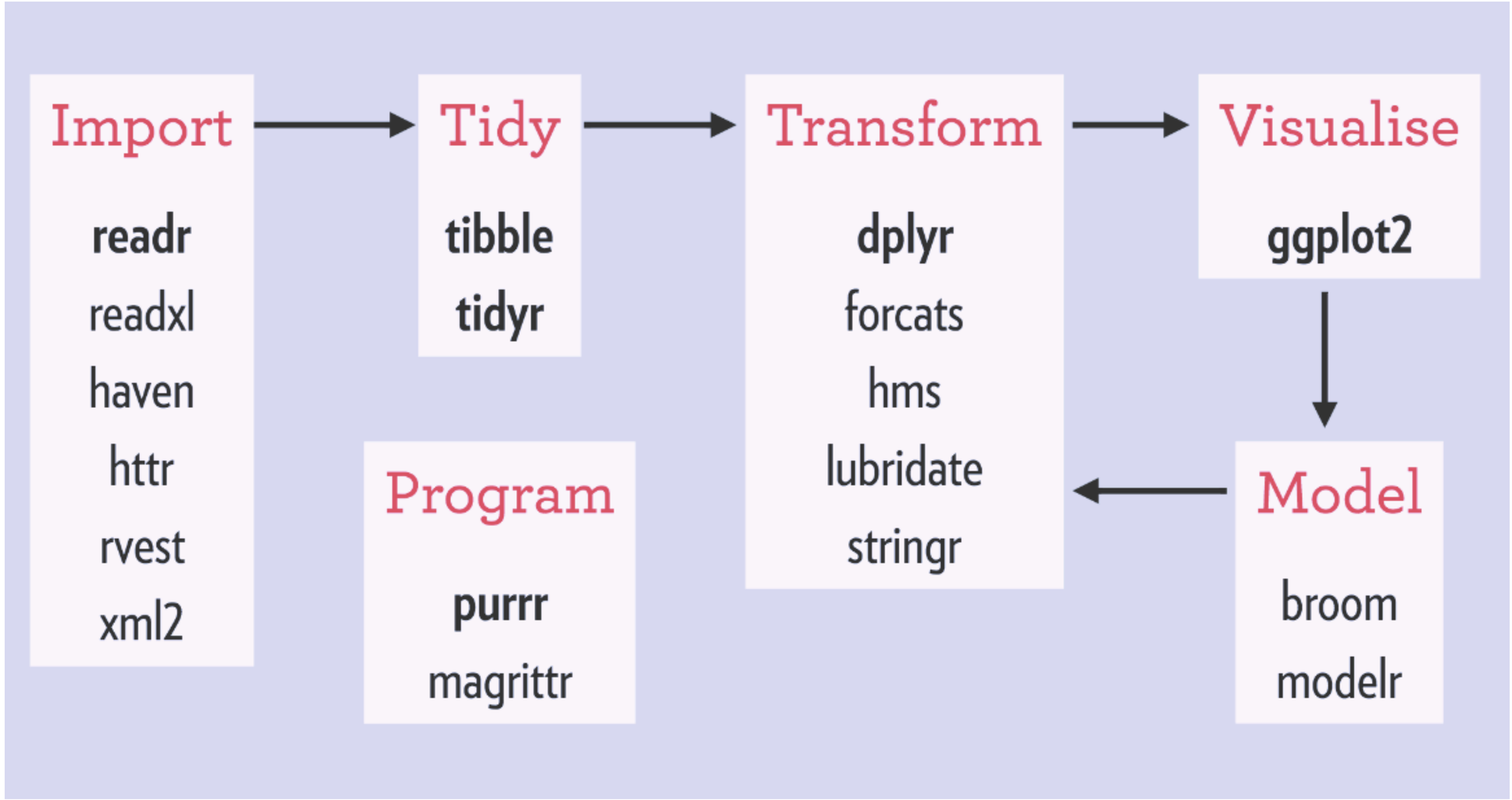
# Thinking like a computer

1. Figure out what you want to do
2. Describe those tasks in words
3. Describe those tasks in code

# What is the Tidyverse?

- Collection of packages for data manipulation, exploration, and visualization that share a common syntax
- Intended to make data scientists more productive by guiding them through workflows
- Allows for connections between tools

# Packages in Tidyverse



# Starwars dataset


```
library(tidyverse)
data(starwars)
View(starwars)
```

variables

| name | height | mass | hair_color | skin_color | eye_color | birth_year | gender | homeworld | species | films | vehicles | starships |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Luke Skywalker | 172 | 77.0 | blond | fair | blue | 19.0 | male | Tatooine | Human | c("Revenge of the Sith", "Return of the Jedi", "The Emp... | c("Snowspeeder", "Imperial Speeder Bike") | c("X-wing", "Imperial shuttle") |
| C-3PO | 167 | 75.0 | *NA* | gold | yellow | 112.0 | *NA* | Tatooine | Droid | c("Attack of the Clones", "The Phantom Menace", "Rev... | character(0) | character(0) |
| R2-D2 | 96 | 32.0 | *NA* | white, blue | red | 33.0 | *NA* | Naboo | Droid | c("Attack of the Clones", "The Phantom Menace", "Rev... | character(0) | character(0) |
| Darth Vader | 202 | 136.0 | none | white | yellow | 41.9 | male | Tatooine | Human | c("Revenge of the Sith", "Return of the Jedi", "The Emp... | character(0) | TIE Advanced x1 |
| Leia Organa | 150 | 49.0 | brown | light | brown | 19.0 | female | Alderaan | Human | c("Revenge of the Sith", "Return of the Jedi", "The Emp... | Imperial Speeder Bike | character(0) |
| Owen Lars | 178 | 120.0 | brown, grey | light | blue | 52.0 | male | Tatooine | Human | c("Attack of the Clones", "Revenge of the Sith", "A New... | character(0) | character(0) |
| Beru Whitesun lars | 165 | 75.0 | brown | light | blue | 47.0 | female | Tatooine | Human | c("Attack of the Clones", "Revenge of the Sith", "A New... | character(0) | character(0) |
| R5-D4 | 97 | 32.0 | *NA* | white, red | red | *NA* | *NA* | Tatooine | Droid | A New Hope | character(0) | character(0) |
| Biggs Darklighter | 183 | 84.0 | black | light | brown | 24.0 | male | Tatooine | Human | A New Hope | character(0) | X-wing |
| Obi-Wan Kenobi | 182 | 77.0 | auburn, white | fair | blue-gray | 57.0 | male | Stewjon | Human | c("Attack of the Clones", "The Phantom Menace", "Rev... | Tribubble bongo | c("Jedi starfighter", "Trade Federation cruiser", "Naboo... |
| Anakin Skywalker | 188 | 84.0 | blond | fair | blue | 41.9 | male | Tatooine | Human | c("Attack of the Clones", "The Phantom Menace", "Rev... | c("Zephyr-G swoop bike", "XJ-6 airspeeder") | c("Trade Federation cruiser", "Jedi Interceptor", "Nabo... |
| Wilhuff Tarkin | 180 | *NA* | auburn, grey | fair | blue | 64.0 | male | Eriadu | Human | c("Revenge of the Sith", "A New Hope") | character(0) | character(0) |
| Chewbacca | 228 | 112.0 | brown | unknown | blue | 200.0 | male | Kashyyyk | Wookiee | c("Revenge of the Sith", "Return of the Jedi", "The Emp... | AT-ST | c("Millennium Falcon", "Imperial shuttle") |
| Han Solo | 180 | 80.0 | brown | fair | brown | 29.0 | male | Corellia | Human | c("Return of the Jedi", "The Empire Strikes Back", "A N... | character(0) | c("Millennium Falcon", "Imperial shuttle") |
| Greedo | 173 | 74.0 | *NA* | green | black | 44.0 | male | Rodia | Rodian | A New Hope | character(0) | character(0) |
| Jabba Desilijic Tiure | 175 | 1358.0 | *NA* | green-tan, brown | orange | 600.0 | hermaphrodite | Nal Hutta | Hutt | c("The Phantom Menace", "Return of the Jedi", "A New ... | character(0) | character(0) |
| Wedge Antilles | 170 | 77.0 | brown | fair | hazel | 21.0 | male | Corellia | Human | c("Return of the Jedi", "The Empire Strikes Back", "A N... | Snowspeeder | X-wing |
| Jek Tono Porkins | 180 | 110.0 | brown | fair | blue | *NA* | male | Bestine IV | Human | A New Hope | character(0) | X-wing |
| Yoda | 66 | 17.0 | white | green | brown | 896.0 | male | *NA* | Yoda's species | c("Attack of the Clones", "The Phantom Menace", "Rev... | character(0) | character(0) |
| Palpatine | 170 | 75.0 | grey | pale | yellow | 82.0 | male | Naboo | Human | c("Attack of the Clones", "The Phantom Menace", "Rev... | character(0) | character(0) |
| Boba Fett | 183 | 78.2 | black | fair | brown | 31.5 | male | Kamino | Human | c("Attack of the Clones", "Return of the Jedi", "The Em... | character(0) | Slave 1 |
| IG-88 | 200 | 140.0 | none | metal | red | 15.0 | none | *NA* | Droid | The Empire Strikes Back | character(0) | character(0) |
| Bossk | 190 | 113.0 | none | green | red | 53.0 | male | Trandosha | Trandoshan | The Empire Strikes Back | character(0) | character(0) |
| Lando Calrissian | 177 | 79.0 | black | dark | brown | 31.0 | male | Socorro | Human | c("Return of the Jedi", "The Empire Strikes Back") | character(0) | Millennium Falcon |

observations

values

# Introduction: **tidyr**

- Collection of functions as **verbs** to easily “tidy” your data

Functions:

- `pivot_longer()` to collapse multiple columns
- `pivot_wider()` to expand one column to multiple
- `unite()` to combine multiple columns into one
- `separate()` to split one column into two

# Wide vs. Long data

**Wide**

| | | | |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

**Long**

| | | |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

# Wide vs. Long data

**Wide**

| name | midterm_1 | midterm_2 | midterm_3 |
|---|---|---|---|
| samantha | 72 | 80 | 81 |
| taylor | 91 | 92 | 90 |
| kelsey | 83 | 74 | 90 |
| ramona | 65 | 71 | 75 |

**Long**

| name | midterm | score |
|---|---|---|
| samantha | midterm_1 | 72 |
| samantha | midterm_2 | 80 |
| samantha | midterm_3 | 81 |
| taylor | midterm_1 | 91 |
| taylor | midterm_2 | 92 |
| taylor | midterm_3 | 90 |
| kelsey | midterm_1 | 83 |
| kelsey | midterm_2 | 74 |
| kelsey | midterm_3 | 90 |
| ramona | midterm_1 | 65 |
| ramona | midterm_2 | 71 |
| ramona | midterm_3 | 75 |

***Which is better?***

# Wide vs. Long data

**Wide**

| name | midterm_1 | midterm_2 | midterm_3 |
|---|---|---|---|
| samantha | 72 | 80 | 81 |
| taylor | 91 | 92 | 90 |
| kelsey | 83 | 74 | 90 |
| ramona | 65 | 71 | 75 |

**Long**

| name | midterm | score |
|---|---|---|
| samantha | midterm_1 | 72 |
| samantha | midterm_2 | 80 |
| samantha | midterm_3 | 81 |
| taylor | midterm_1 | 91 |
| taylor | midterm_2 | 92 |
| taylor | midterm_3 | 90 |
| kelsey | midterm_1 | 83 |
| kelsey | midterm_2 | 74 |
| kelsey | midterm_3 | 90 |
| ramona | midterm_1 | 65 |
| ramona | midterm_2 | 71 |
| ramona | midterm_3 | 75 |

***Q1: Find the average of each student's midterms***

# Wide vs. Long data

**Wide**

| name | midterm_1 | midterm_2 | midterm_3 |
|---|---|---|---|
| samantha | 72 | 80 | 81 |
| taylor | 91 | 92 | 90 |
| kelsey | 83 | 74 | 90 |
| ramona | 65 | 71 | 75 |

```
df %>%
  dplyr::mutate(average = mean(midterm_1, midterm_2, midterm_3))
```

***Q1: Find the average of each student's midterms***

# Wide vs. Long data

**Wide**

| name | midterm_1 | midterm_2 | midterm_3 | average |
|---|---|---|---|---|
| samantha | 72 | 80 | 81 | 77.6 |
| taylor | 91 | 92 | 90 | 91 |
| kelsey | 83 | 74 | 90 | 82.3 |
| ramona | 65 | 71 | 75 | 70.3 |

```
df %>%
  dplyr::mutate(average = mean(midterm_1, midterm_2, midterm_3))
```

*Imagine you have 100 midterms to average… this would be difficult to script*

***Q1: Find the average of each student's midterms***

# Wide vs. Long data

```
df %>%
  dplyr::group_by(name) %>%
  dplyr::mutate(average = mean(score))
```

**Long**

| name | midterm | score |
|---|---|---|
| samantha | midterm_1 | 72 |
| samantha | midterm_2 | 80 |
| samantha | midterm_3 | 81 |
| taylor | midterm_1 | 91 |
| taylor | midterm_2 | 92 |
| taylor | midterm_3 | 90 |
| kelsey | midterm_1 | 83 |
| kelsey | midterm_2 | 74 |
| kelsey | midterm_3 | 90 |
| ramona | midterm_1 | 65 |
| ramona | midterm_2 | 71 |
| ramona | midterm_3 | 75 |

***Q1: Find the average of each student's midterms***

# Wide vs. Long data

```
df %>%
  dplyr::group_by(name) %>%
  dplyr::mutate(average = mean(score))
```

*The script won't change no matter how many midterms you have to score!*

**Long**

| name | midterm | score | average |
|---|---|---|---|
| samantha | midterm_1 | 72 | 77.6 |
| samantha | midterm_2 | 80 | 77.6 |
| samantha | midterm_3 | 81 | 77.6 |
| taylor | midterm_1 | 91 | 91 |
| taylor | midterm_2 | 92 | 91 |
| taylor | midterm_3 | 90 | 91 |
| kelsey | midterm_1 | 83 | 82.3 |
| kelsey | midterm_2 | 74 | 82.3 |
| kelsey | midterm_3 | 90 | 82.3 |
| ramona | midterm_1 | 65 | 70.3 |
| ramona | midterm_2 | 71 | 70.3 |
| ramona | midterm_3 | 75 | 70.3 |

***Q1: Find the average of each student's midterms***

# Wide vs. Long data

**Wide**

| name | midterm_1 | midterm_2 | midterm_3 |
|---|---|---|---|
| samantha | 72 | 80 | 81 |
| taylor | 91 | 92 | 90 |
| kelsey | 83 | 74 | 90 |
| ramona | 65 | 71 | 75 |

**Long**

| name | midterm | score |
|---|---|---|
| samantha | midterm_1 | 72 |
| samantha | midterm_2 | 80 |
| samantha | midterm_3 | 81 |
| taylor | midterm_1 | 91 |
| taylor | midterm_2 | 92 |
| taylor | midterm_3 | 90 |
| kelsey | midterm_1 | 83 |
| kelsey | midterm_2 | 74 |
| kelsey | midterm_3 | 90 |
| ramona | midterm_1 | 65 |
| ramona | midterm_2 | 71 |
| ramona | midterm_3 | 75 |

***Q3: Find the ratio between midterm 1 and 2***

# Wide vs. Long data

**Wide**

| name | midterm_1 | midterm_2 | midterm_3 |
|---|---|---|---|
| samantha | 72 | 80 | 81 |
| taylor | 91 | 92 | 90 |
| kelsey | 83 | 74 | 90 |
| ramona | 65 | 71 | 75 |

```
df %>%
  dplyr::mutate(ratio = midterm_1 / midterm_2)
```

***Q3: Find the ratio between midterm 1 and 2***

# Wide vs. Long data

**Wide**

| name | midterm_1 | midterm_2 | midterm_3 | ratio |
|---|---|---|---|---|
| samantha | 72 | 80 | 81 | 0.9 |
| taylor | 91 | 92 | 90 | 0.989 |
| kelsey | 83 | 74 | 90 | 1.12 |
| ramona | 65 | 71 | 75 | 0.915 |

```
df %>%
  dplyr::mutate(ratio = midterm_1 / midterm_2)
```

*This would be more difficult to do with the long data…*

***Q3: Find the ratio between midterm 1 and 2***

# Starwars dataset

**WIDE**

**LONG**

variables

| | name | height | mass | hair_color | skin_color | eye_color | birth_year | gender | homeworld | species | films | vehicles | starships |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Luke Skywalker | 172 | 77.0 | blond | fair | blue | 19.0 | male | Tatooine | Human | c("Revenge of the Sith", "Return of the Jedi", "The Emp... | c("Snowspeeder", "Imperial Speeder Bike") | c("X-wing", "Imperial shuttle") |
| 2 | C-3PO | 167 | 75.0 | *NA* | gold | yellow | 112.0 | *NA* | Tatooine | Droid | c("Attack of the Clones", "The Phantom Menace", "Rev... | character(0) | character(0) |
| 3 | R2-D2 | 96 | 32.0 | *NA* | white, blue | red | 33.0 | *NA* | Naboo | Droid | c("Attack of the Clones", "The Phantom Menace", "Rev... | character(0) | character(0) |
| 4 | Darth Vader | 202 | 136.0 | none | white | yellow | 41.9 | male | Tatooine | Human | c("Revenge of the Sith", "Return of the Jedi", "The Emp... | character(0) | TIE Advanced x1 |
| 5 | Leia Organa | 150 | 49.0 | brown | light | brown | 19.0 | female | Alderaan | Human | c("Revenge of the Sith", "Return of the Jedi", "The Emp... | Imperial Speeder Bike | character(0) |
| 6 | Owen Lars | 178 | 120.0 | brown, grey | light | blue | 52.0 | male | Tatooine | Human | c("Attack of the Clones", "Revenge of the Sith", "A New... | character(0) | character(0) |
| 7 | Beru Whitesun lars | 165 | 75.0 | brown | light | blue | 47.0 | female | Tatooine | Human | c("Attack of the Clones", "Revenge of the Sith", "A New... | character(0) | character(0) |
| 8 | R5-D4 | 97 | 32.0 | *NA* | white, red | red | *NA* | *NA* | Tatooine | Droid | A New Hope | character(0) | character(0) |
| 9 | Biggs Darklighter | 183 | 84.0 | black | light | brown | 24.0 | male | Tatooine | Human | A New Hope | character(0) | X-wing |
| 10 | Obi-Wan Kenobi | 182 | 77.0 | auburn, white | fair | blue-gray | 57.0 | male | Stewjon | Human | c("Attack of the Clones", "The Phantom Menace", "Rev... | Tribubble bongo | c("Jedi starfighter", "Trade Federation cruiser", "Naboo... |
| 11 | Anakin Skywalker | 188 | 84.0 | blond | fair | blue | 41.9 | male | Tatooine | Human | c("Attack of the Clones", "The Phantom Menace", "Rev... | c("Zephyr-G swoop bike", "XJ-6 airspeeder") | c("Trade Federation cruiser", "Jedi Interceptor", "Nabo... |
| 12 | Wilhuff Tarkin | 180 | *NA* | auburn, grey | fair | blue | 64.0 | male | Eriadu | Human | c("Revenge of the Sith", "A New Hope") | character(0) | character(0) |
| 13 | Chewbacca | 228 | 112.0 | brown | unknown | blue | 200.0 | male | Kashyyyk | Wookiee | c("Revenge of the Sith", "Return of the Jedi", "The Emp... | AT-ST | c("Millennium Falcon", "Imperial shuttle") |
| 14 | Han Solo | 180 | 80.0 | brown | fair | brown | 29.0 | male | Corellia | Human | c("Return of the Jedi", "The Empire Strikes Back", "A N... | character(0) | c("Millennium Falcon", "Imperial shuttle") |
| 15 | Greedo | 173 | 74.0 | *NA* | green | black | 44.0 | male | Rodia | Rodian | A New Hope | character(0) | character(0) |
| 16 | Jabba Desilijic Tiure | 175 | 1358.0 | *NA* | green-tan, brown | orange | 600.0 | hermaphrodite | Nal Hutta | Hutt | c("The Phantom Menace", "Return of the Jedi", "A New ... | character(0) | character(0) |
| 17 | Wedge Antilles | 170 | 77.0 | brown | fair | hazel | 21.0 | male | Corellia | Human | c("Return of the Jedi", "The Empire Strikes Back", "A N... | Snowspeeder | X-wing |
| 18 | Jek Tono Porkins | 180 | 110.0 | brown | fair | blue | *NA* | male | Bestine IV | Human | A New Hope | character(0) | X-wing |
| 19 | Yoda | 66 | 17.0 | white | green | brown | 896.0 | male | *NA* | Yoda's species | c("Attack of the Clones", "The Phantom Menace", "Rev... | character(0) | character(0) |
| 20 | Palpatine | 170 | 75.0 | grey | pale | yellow | 82.0 | male | Naboo | Human | c("Attack of the Clones", "The Phantom Menace", "Rev... | character(0) | character(0) |
| 21 | Boba Fett | 183 | 78.2 | black | fair | brown | 31.5 | male | Kamino | Human | c("Attack of the Clones", "Return of the Jedi", "The Em... | character(0) | Slave 1 |
| 22 | IG-88 | 200 | 140.0 | none | metal | red | 15.0 | none | *NA* | Droid | The Empire Strikes Back | character(0) | character(0) |
| 23 | Bossk | 190 | 113.0 | none | green | red | 53.0 | male | Trandosha | Trandoshan | The Empire Strikes Back | character(0) | character(0) |
| 24 | Lando Calrissian | 177 | 79.0 | black | dark | brown | 31.0 | male | Socorro | Human | c("Return of the Jedi", "The Empire Strikes Back") | character(0) | Millennium Falcon |

observations

values

# Starwars dataset

☐ **WIDE**

☑ **LONG**

variables

| name | characteristic | value |
|---|---|---|
| Luke Skywalker | height | 172 |
| Luke Skywalker | mass | 77 |
| Luke Skywalker | hair_color | blond |
| Luke Skywalker | skin_color | fair |
| Luke Skywalker | eye_color | blue |
| Luke Skywalker | birth_year | 19 |
| Luke Skywalker | gender | male |
| Luke Skywalker | homeworld | Tatooine |
| Luke Skywalker | species | Human |
| Luke Skywalker | films | c("Revenge of the Sith", "Return of the Jedi", "The Emp... |
| Luke Skywalker | vehicles | c("Snowspeeder", "Imperial Speeder Bike") |
| Luke Skywalker | starships | c("X-wing", "Imperial shuttle") |
| Luminara Unduli | height | 170 |
| Luminara Unduli | mass | 56.2 |
| Luminara Unduli | hair_color | black |
| Luminara Unduli | skin_color | yellow |
| Luminara Unduli | eye_color | blue |
| Luminara Unduli | birth_year | 58 |
| Luminara Unduli | gender | female |

observations

values

**MESSY DATA**

- Each **value** must have its own **cell**



# `tidyr::pivot_longer()`

`tidyr::pivot_longer()` to collapse multiple columns

# `tidyr::pivot_longer()`

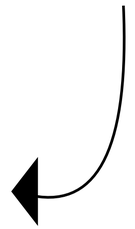
```
tidyr::pivot_longer(columns_to_include, key, value)
```

`tidyr::pivot_longer()` to collapse multiple columns

# tidyr::pivot_longer()

`tidyr::pivot_longer(columns_to_include, key, value)`

| name | midterm_1 | midterm_2 | midterm_3 |
|---|---|---|---|
| samantha | 72 | 80 | 81 |
| taylor | 91 | 92 | 90 |
| kelsey | 83 | 74 | 90 |
| ramona | 65 | 71 | 75 |



| name | midterm | score |
|---|---|---|
| samantha | midterm_1 | 72 |
| samantha | midterm_2 | 80 |
| samantha | midterm_3 | 81 |
| taylor | midterm_1 | 91 |
| taylor | midterm_2 | 92 |
| taylor | midterm_3 | 90 |
| kelsey | midterm_1 | 83 |
| kelsey | midterm_2 | 74 |
| kelsey | midterm_3 | 90 |
| ramona | midterm_1 | 65 |
| ramona | midterm_2 | 71 |
| ramona | midterm_3 | 75 |

`tidyr::pivot_longer()` to collapse multiple columns

# tidyr::pivot_longer()

```
tidyr::pivot_longer(columns_to_include, key, value)
```

| name | midterm_1 | midterm_2 | midterm_3 |
|---|---|---|---|
| samantha | 72 | 80 | 81 |
| taylor | 91 | 92 | 90 |
| kelsey | 83 | 74 | 90 |
| ramona | 65 | 71 | 75 |

| name | midterm | score |
|---|---|---|
| samantha | midterm_1 | 72 |
| samantha | midterm_2 | 80 |
| samantha | midterm_3 | 81 |
| taylor | midterm_1 | 91 |
| taylor | midterm_2 | 92 |
| taylor | midterm_3 | 90 |
| kelsey | midterm_3 | 90 |
| ramona | midterm_1 | 65 |
| ramona | midterm_2 | 71 |
| ramona | midterm_3 | 75 |

```
long_df <- tidyr::pivot_longer(midterm_1:midterm_3, midterm, score,)
```

`tidyr::pivot_longer()` to collapse multiple columns

# tidyr::pivot_longer()

`tidyr::pivot_longer(columns_to_include, key, value)`

**Convert the starwars dataframe from wide to long, containing three columns:**
***name, characteristic, value***

| name | characteristic | value |
|---|---|---|
| Luke Skywalker | height | 172 |
| C-3PO | height | 167 |
| R2-D2 | height | 96 |
| Darth Vader | height | 202 |
| Leia Organa | height | 150 |
| Owen Lars | height | 178 |
| Beru Whitesun lars | height | 165 |
| R5-D4 | height | 97 |
| Biggs Darklighter | height | 183 |
| Obi-Wan Kenobi | height | 182 |
| Anakin Skywalker | height | 188 |
| Wilhuff Tarkin | height | 180 |

`tidyr::pivot_longer()` to collapse multiple columns

# tidyr::**pivot_longer**()

`tidyr::pivot_longer(columns_to_include, key, value)`

**Convert the starwars dataframe from wide to long, containing three columns:**
***name, characteristic, value***

```
lgstarwars <- starwars %>%
  tidyr::pivot_long(cols=c(height, mass, birth_year),
                    names_to="characteristic",
                    values_to = "value") %>%
  dplyr::select(name, characteristic, value)
```

`tidyr::pivot_longer()` to collapse multiple columns

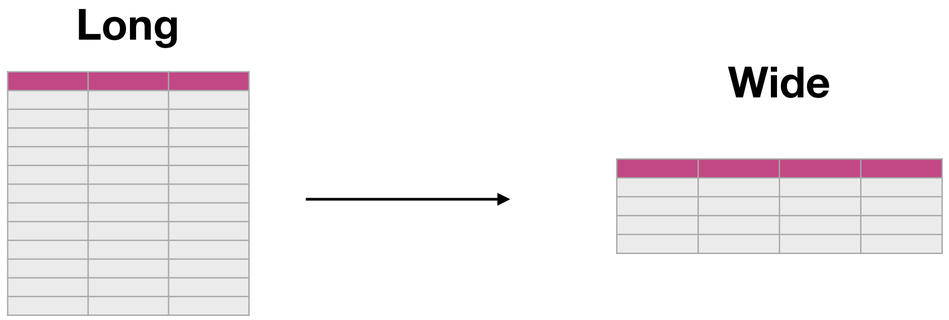
# tidyr::pivot_longer()

```
tidyr::pivot_longer(columns_to_include, key, value)
```

**Convert the starwars dataframe from wide to long, containing three columns:**
***name, characteristic, value***

| | name | characteristic | value |
|---|---|---|---|
| 1 | Luke Skywalker | height | 172.0 |
| 2 | Luke Skywalker | mass | 77.0 |
| 3 | Luke Skywalker | birth_year | 19.0 |
| 4 | C-3PO | height | 167.0 |
| 5 | C-3PO | mass | 75.0 |
| 6 | C-3PO | birth_year | 112.0 |
| 7 | R2-D2 | height | 96.0 |
| 8 | R2-D2 | mass | 32.0 |
| 9 | R2-D2 | birth_year | 33.0 |
| 10 | Darth Vader | height | 202.0 |
| 11 | Darth Vader | mass | 136.0 |
| 12 | Darth Vader | birth_year | 41.9 |
| 13 | Leia Organa | height | 150.0 |
| 14 | Leia Organa | mass | 49.0 |
| 15 | Leia Organa | birth_year | 19.0 |
| 16 | Owen Lars | height | 178.0 |
| 17 | Owen Lars | mass | 120.0 |
| 18 | Owen Lars | birth_year | 52.0 |
| 19 | Beru Whitesun lars | height | 165.0 |
| 20 | Beru Whitesun lars | mass | 75.0 |
| 21 | Beru Whitesun lars | birth_year | 47.0 |
| 22 | R5-D4 | height | 97.0 |
| 23 | R5-D4 | mass | 32.0 |
| 24 | R5-D4 | birth_year | NA |

`tidyr::pivot_longer()` to collapse multiple columns





# `tidyr::pivot_wider()`

`tidyr::pivot_wider()` to expand one column to multiple

# `tidyr::pivot_wider()`

`tidyr::pivot_wider(dataframe, key, value)`

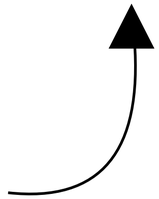
`tidyr::pivot_wider()` to expand one column to multiple

# tidyr::pivot_wider()

`tidyr::pivot_wider(dataframe, key, value)`

| name | midterm_1 | midterm_2 | midterm_3 |
|---|---|---|---|
| samantha | 72 | 80 | 81 |
| taylor | 91 | 92 | 90 |
| kelsey | 83 | 74 | 90 |
| ramona | 65 | 71 | 75 |

| name | midterm | score |
|---|---|---|
| samantha | midterm_1 | 72 |
| samantha | midterm_2 | 80 |
| samantha | midterm_3 | 81 |
| taylor | midterm_1 | 91 |
| taylor | midterm_2 | 92 |
| taylor | midterm_3 | 90 |
| kelsey | midterm_1 | 83 |
| kelsey | midterm_2 | 74 |
| kelsey | midterm_3 | 90 |
| ramona | midterm_1 | 65 |
| ramona | midterm_2 | 71 |
| ramona | midterm_3 | 75 |



`tidyr::pivot_wider()` to expand one column to multiple

# tidyr::pivot_wider()

`tidyr::pivot_wider(dataframe, key, value)`

| name | midterm_1 | midterm_2 | midterm_3 |
|---|---|---|---|
| samantha | 72 | 80 | 81 |
| taylor | 91 | 92 | 90 |
| kelsey | 83 | 74 | 90 |
| ramona | 65 | 71 | 75 |

| name | midterm | score |
|---|---|---|
| samantha | midterm_1 | 72 |
| samantha | midterm_2 | 80 |
| samantha | midterm_3 | 81 |
| taylor | midterm_1 | 91 |
| taylor | midterm_2 | 92 |
| taylor | midterm_3 | 90 |
| kelsey | midterm_1 | 83 |
| kelsey | midterm_2 | 74 |
| kelsey | midterm_3 | 90 |
| ramona | midterm_1 | 65 |
| ramona | midterm_2 | 71 |
| ramona | midterm_3 | 75 |

`tidyr::pivot_wider()` to expand one column to multiple

# tidyr::pivot_wider()

`tidyr::pivot_wider(dataframe, key, value)`

| name | midterm_1 | midterm_2 | midterm_3 |
|---|---|---|---|
| samantha | 72 | 80 | 81 |
| taylor | 91 | 92 | 90 |
| kelsey | 83 | 74 | 90 |
| ramona | 65 | 71 | 75 |

| name | midterm | score |
|---|---|---|
| samantha | midterm_1 | 72 |
| samantha | midterm_2 | 80 |
| samantha | midterm_3 | 81 |
| taylor | midterm_1 | 91 |
| taylor | midterm_2 | 92 |
| taylor | midterm_3 | 90 |
| kelsey | midterm_3 | 90 |
| ramona | midterm_1 | 65 |
| ramona | midterm_2 | 71 |
| ramona | midterm_3 | 75 |

```
wide_df <- tidyr::pivot_wider(long_df, names_from = midterm, values_from = score)
```

`tidyr::pivot_wider()` to expand one column to multiple

# tidyr::pivot_wider()

`tidyr::pivot_wider(dataframe, key, value)`

**Un-gather (spread) the starwars data frame**

```
ungathered <- gathered %>%
  tidyr::pivot_wider(names_from = characteristic, values_from = value)
```

| name | characteristic | value |
|---|---|---|
| Luke Skywalker | height | 172 |
| C-3PO | height | 167 |
| R2-D2 | height | 96 |
| Darth Vader | height | 202 |
| Leia Organa | height | 150 |
| Owen Lars | height | 178 |
| Beru Whitesun lars | height | 165 |
| R5-D4 | height | 97 |
| Biggs Darklighter | height | 183 |

| name | height | mass | hair_color | skin_color | eye_color | bir |
|---|---|---|---|---|---|---|
| Luke Skywalker | 172 | 77.0 | blond | fair | blue | |
| C-3PO | 167 | 75.0 | *NA* | gold | yellow | |
| R2-D2 | 96 | 32.0 | *NA* | white, blue | red | |
| Darth Vader | 202 | 136.0 | none | white | yellow | |
| Leia Organa | 150 | 49.0 | brown | light | brown | |
| Owen Lars | 178 | 120.0 | brown, grey | light | blue | |
| Beru Whitesun lars | 165 | 75.0 | brown | light | blue | |
| R5-D4 | 97 | 32.0 | *NA* | white, red | red | |
| Biggs Darklighter | 183 | 84.0 | black | light | brown | |

`tidyr::pivot_wider()` to expand one column to multiple

# tidyr::pivot_wider()

name = key

`tidyr::pivot_wider(dataframe, key, value)`

***GOAL:***

| Ackbar | Adi Gallia | Anakin Skywalker | Arvel Crynyd | Ayla Secura | Bail Prestor Organa | Barriss Offee | BB8 | Ben Quadinaros | Beru Whitesun lars | Bib Fortuna | Biggs Darklighter | Boba Fett | Bossk | C-3PO | Captain Phasma | Chewbacca |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| orange | blue | blue | brown | hazel | brown | blue | black | orange | blue | pink | brown | brown | red | yellow | unknown | blue |

eye_color = value

**1. Select columns to keep**

name, eye_color

**2. Spread data frame**

*What is the key? What is the value?*

```
eyes <- starwars %>%
  dplyr::select(name, eye_color) %>%
  tidyr::pivot_wider(name, eye_color)
```

`tidyr::pivot_wider()` to expand one column to multiple

# tidyr::pivot_wider()

name = key

`tidyr::pivot_wider(dataframe, key, value)`

***GOAL:***

| Ackbar | Adi Gallia | Anakin Skywalker | Arvel Crynyd | Ayla Secura | Bail Prestor Organa | Barriss Offee | BB8 | Ben Quadinaros | Beru Whitesun lars | Bib Fortuna | Biggs Darklighter | Boba Fett | Bossk | C-3PO | Captain Phasma | Chewbacca |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| orange | blue | blue | brown | hazel | brown | blue | black | orange | blue | pink | brown | brown | red | yellow | unknown | blue |

eye_color = value

~~**1. Select columns to keep**~~

~~name, eye_color~~

**2. Spread data frame**

*What is the key? What is the value?*

```
eyes <- starwars %>%
  ~~dplyr::select(name, eye_color) %>%~~
  tidyr::pivot_wider(name, eye_color)
```

`tidyr::pivot_wider()` to expand one column to multiple

# tidyr::pivot_wider()

`tidyr::pivot_wider(dataframe, key, value)`

| vehicles | starships | Ackbar | Adi Gallia | Anakin Skywalker | Arvel Crynyd | Ayla Secura | Bail Prestor Organa | Barriss Offee | BB8 | Ben Quadinaros | Beru Whitesun lars | Bib Fortuna | B D |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| c("Snowspeeder", "Imperial Speeder Bike") | c("X-wing", "Imperial shuttle") | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | TIE Advanced x1 | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| Imperial Speeder Bike | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | blue | NA | |
| character(0) | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | X-wing | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| Tribubble bongo | c("Jedi starfighter", "Trade Federation cruiser", "Naboo... | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| c("Zephyr-G swoop bike", "XJ-6 airspeeder") | c("Trade Federation cruiser", "Jedi Interceptor", "Nabo... | NA | NA | blue | NA | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| AT-ST | c("Millennium Falcon", "Imperial shuttle") | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | c("Millennium Falcon", "Imperial shuttle") | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| Snowspeeder | X-wing | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | X-wing | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | Slave 1 | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | Millennium Falcon | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | character(0) | orange | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | A-wing | NA | NA | NA | brown | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | Millennium Falcon | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| Tribubble bongo | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |
| character(0) | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | |

`tidyr::pivot_wider()` to expand one column to multiple

# tidyr::pivot_wider()

`tidyr::pivot_wider(dataframe, key, value)`

| vehicles | starships | Ackbar | Adi Gallia | Anakin Skywalker | Arvel Crynyd | Ayla Secura | Bail Prestor Organa | Barriss Offee | BB8 | Ben Quadinaros | Beru Whitesun lars | Bib Fortuna |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| c("Snowspeeder", "Imperial Speeder Bike") | c("X-wing", "Imperial shuttle") | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA |
| character(0) | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA |
| character(0) | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA |
| character(0) | TIE Advanced x1 | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA |
| Imperial Speeder Bike | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA |
| character(0) | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA |
| character(0) | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | blue | NA |
| character(0) | character(0) | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA |
| character(0) | X-wing | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA |
| Tribubble bongo | c("Jedi starfighter", "Trade Federation cruiser", "Naboo... | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA |
| c("Zephyr-G swoop bike", "XJ-6 airspeeder") | c("Trade Federation cruiser", "Jedi Interceptor", "Nabo... | NA | NA | blue | NA | NA | NA | NA | NA | NA | NA | NA |

Can fill empty values with `values_fill = "xxx"`

```
eyes <- starwars %>%
  ~~dplyr::select(name, eye_color) %>%~~
  tidyr::pivot_wider(name, eye_color, values_fill = 0)
```

`tidyr::pivot_wider()` to expand one column to multiple

# tidyr::pivot_wider()

`tidyr::pivot_wider(dataframe, key, value)`

| 0 hicles | starships | Ackbar | Adi Gallia | Anakin Skywalker | Arvel Crynyd | Ayla Secura | Bail Prestor Organa | Barriss Offee | BB8 | Ben Quadinaros | Beru Whitesun lars | Bib Fortuna |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| c("Snowspeeder", "Imperial Speeder Bike") | c("X-wing", "Imperial shuttle") | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| character(0) | character(0) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| character(0) | character(0) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| character(0) | TIE Advanced x1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Imperial Speeder Bike | character(0) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| character(0) | character(0) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| character(0) | character(0) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | blue | 0 |
| character(0) | character(0) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| character(0) | X-wing | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Tribubble bongo | c("Jedi starfighter", "Trade Federation cruiser", "Naboo... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| c("Zephyr-G swoop bike", "XJ-6 airspeeder") | c("Trade Federation cruiser", "Jedi Interceptor", "Nabo... | 0 | 0 | blue | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Can fill empty values with `values_fill = "xxx"`

```
eyes <- starwars %>%
  ~~dplyr::select(name, eye_color) %>%~~
  tidyr::pivot_wider(name, eye_color, values_fill = 0)
```

`tidyr::pivot_wider()` to expand one column to multiple


# `tidyr::unite()`

`tidyr::unite()` to combine multiple columns into one

# tidyr::unite()

`tidyr::unite(dataframe, old_columns, col = new_column, sep = separator)`

**Combine name and homeworld into one column named character**
*(e.g. Luke Skywalker, Tatooine)*

```
united <- starwars %>%
  tidyr::unite(name, homeworld, col = "character", sep = ", ")
```

| character | height | mass | hair_color | skin_color | eye_color | birth_year |
|---|---|---|---|---|---|---|
| Luke Skywalker, Tatooine | 172 | 77.0 | blond | fair | blue | 19.0 |
| C-3PO, Tatooine | 167 | 75.0 | *NA* | gold | yellow | 112.0 |
| R2-D2, Naboo | 96 | 32.0 | *NA* | white, blue | red | 33.0 |
| Darth Vader, Tatooine | 202 | 136.0 | none | white | yellow | 41.9 |
| Leia Organa, Alderaan | 150 | 49.0 | brown | light | brown | 19.0 |
| Owen Lars, Tatooine | 178 | 120.0 | brown, grey | light | blue | 52.0 |
| Beru Whitesun lars, Tatooine | 165 | 75.0 | brown | light | blue | 47.0 |

`tidyr::unite()` to combine multiple columns into one

# `tidyr::unite()`

`tidyr::unite(`dataframe, old_columns, `col =` new_column, `sep =` separator`)`

**Combine name, homeworld, and species into one column named character**
*(e.g. Luke Skywalker, Tatooine, Human)*

```
united <- starwars %>%
  tidyr::unite(name, homeworld, species,
               col = "character", sep = ", ")
```

Keep the original columns with argument `remove = FALSE`

```
united <- starwars %>%
  tidyr::unite(name, homeworld, species,
               col = "character", sep = ", ‟, remove = FALSE)
```

`tidyr::unite()` to combine multiple columns into one

# `tidyr::unite()`

`tidyr::unite(dataframe, old_columns, col = new_column, sep = separator)`

Keep the original columns with argument `remove = FALSE`

| character | name | height | mass | hair_color | skin_color | eye_color | birth_year | gender | homeworld |
|---|---|---|---|---|---|---|---|---|---|
| Luke Skywalker, Tatooine | Luke Skywalker | 172 | 77.0 | blond | fair | blue | 19.0 | male | Tatooine |
| C-3PO, Tatooine | C-3PO | 167 | 75.0 | *NA* | gold | yellow | 112.0 | *NA* | Tatooine |
| R2-D2, Naboo | R2-D2 | 96 | 32.0 | *NA* | white, blue | red | 33.0 | *NA* | Naboo |
| Darth Vader, Tatooine | Darth Vader | 202 | 136.0 | none | white | yellow | 41.9 | male | Tatooine |
| Leia Organa, Alderaan | Leia Organa | 150 | 49.0 | brown | light | brown | 19.0 | female | Alderaan |
| Owen Lars, Tatooine | Owen Lars | 178 | 120.0 | brown, grey | light | blue | 52.0 | male | Tatooine |
| Beru Whitesun lars, Tatooine | Beru Whitesun lars | 165 | 75.0 | brown | light | blue | 47.0 | female | Tatooine |
| R5-D4, Tatooine | R5-D4 | 97 | 32.0 | *NA* | white, red | red | *NA* | *NA* | Tatooine |
| Biggs Darklighter, Tatooine | Biggs Darklighter | 183 | 84.0 | black | light | brown | 24.0 | male | Tatooine |

`tidyr::unite()` to combine multiple columns into one

# tidyr::unite()

`tidyr::unite(dataframe, old_columns, col = new_column, sep = separator)`

**Combine name, homeworld, and species into one column named character**
*(e.g. Luke Skywalker, Tatooine (Human))*

*Hint: This might take multiple steps…*

Hint: `paste("My name is", "Katie", sep = " ")` = "My name is Katie"

```
united <- starwars %>%
  tidyr::unite(name, homeworld, col = "character", sep = ", ") %>%
  dplyr::mutate(species = paste("(", species, ")", sep = "")) %>%
  tidyr::unite(character, species, col = "character", sep = " ")
```

`tidyr::unite()` to combine multiple columns into one


# `tidyr::separate()`

`tidyr::separate()` to split one column into two

# tidyr::separate()

`tidyr::separate(`dataframe, old_column, `into = c(`new_columns), `sep =` separator`)`

**Separate the 'character' column back into name and homeworld**

```
separated <- united %>%
 tidyr::separate(character, into = c("name", "homeworld"), sep = ",")
```

Keep the original columns with argument `remove = FALSE`

```
separated <- united %>%
 tidyr::separate(character, into = c("name", "homeworld"),
        sep = ",", remove = FALSE)
```

`tidyr::separate()` to split one column into two

# tidyr::separate()

`tidyr::separate(dataframe, old_column, into = c(new_columns), sep = separator)`

**Separate characters into first and last names**

```
names <- starwars %>%
 tidyr::separate(name, into = c("first_name", "last_name"), sep = " ")
```

| first_name | last_name | height | mass | hair_color | skin_color | eye_color | birth_year | gender | homeworld | species |
|---|---|---|---|---|---|---|---|---|---|---|
| Luke | Skywalker | 172 | 77.0 | blond | fair | blue | 19.0 | male | Tatooine | Human |
| C-3PO | *NA* | 167 | 75.0 | *NA* | gold | yellow | 112.0 | *NA* | Tatooine | Droid |
| R2-D2 | *NA* | 96 | 32.0 | *NA* | white, blue | red | 33.0 | *NA* | Naboo | Droid |
| Darth | Vader | 202 | 136.0 | none | white | yellow | 41.9 | male | Tatooine | Human |
| Leia | Organa | 150 | 49.0 | brown | light | brown | 19.0 | female | Alderaan | Human |
| Owen | Lars | 178 | 120.0 | brown, grey | light | blue | 52.0 | male | Tatooine | Human |
| Beru | Whitesun | 165 | 75.0 | brown | light | blue | 47.0 | female | Tatooine | Human |
| R5-D4 | *NA* | 97 | 32.0 | *NA* | white, red | red | *NA* | *NA* | Tatooine | Droid |

*Beru Whitesun lars*

`tidyr::separate()` to split one column into two

# Data tidying with tidyr : : **CHEAT SHEET**

**Tidy data** is a way to organize tabular data in a consistent data structure across packages.
A table is tidy if:


Each **variable** is in its own **column**

Each **observation**, or **case**, is in its own row


Access **variables** as **vectors**

Preserve **cases** in vectorized operations

## Tibbles

### AN ENHANCED DATA FRAME

Tibbles are a table format provided by the **tibble** package. They inherit the data frame class, but have improved behaviors:

- **Subset** a new tibble with ], a vector with [[ and $.
- **No partial matching** when subsetting columns.
- **Display** concise views of the data on one screen.

**options(**tibble.print_max = n, tibble.print_min = m, tibble.width = Inf**)** Control default display settings.

**View()** or **glimpse()** View the entire data set.

### CONSTRUCT A TIBBLE

**tibble(**…**)** Construct by columns.
tibble(x = 1:3, y = c("a", "b", "c"))

**tribble(**…**)** Construct by rows.
tribble(~x, ~y,
1, "a",
2, "b",
3, "c")

**Both make this tibble**

```
A tibble: 3 × 2
      x     y
  <int> <chr>
1     1     a
2     2     b
3     3     c
```

**as_tibble(**x, …**)** Convert a data frame to a tibble.

**enframe(**x, name = "name", value = "value"**)** Convert a named vector to a tibble. Also **deframe()**.

**is_tibble(**x**)** Test whether x is a tibble.

## Reshape Data - Pivot data to reorganize values into a new layout.

table4a

| country | 1999 | 2000 |
|---|---|---|
| A | 0.7K | 2K |
| B | 37K | 80K |
| C | 212K | 213K |

→

| country | year | cases |
|---|---|---|
| A | 1999 | 0.7K |
| B | 1999 | 37K |
| C | 1999 | 212K |
| A | 2000 | 2K |
| B | 2000 | 80K |
| C | 2000 | 213K |

**pivot_longer(**data, cols, names_to = "name", values_to = "value", values_drop_na = FALSE**)**

"Lengthen" data by collapsing several columns into two. Column names move to a new names_to column and values to a new values_to column.

pivot_longer(table4a, cols = 2:3, names_to ="year", values_to = "cases")

table2

| country | year | type | count |
|---|---|---|---|
| A | 1999 | cases | 0.7K |
| A | 1999 | pop | 19M |
| A | 2000 | cases | 2K |
| A | 2000 | pop | 20M |
| B | 1999 | cases | 37K |
| B | 1999 | pop | 172M |
| B | 2000 | cases | 80K |
| B | 2000 | pop | 174M |
| C | 1999 | cases | 212K |
| C | 1999 | pop | 1T |
| C | 2000 | cases | 213K |
| C | 2000 | pop | 1T |

→

| country | year | cases | pop |
|---|---|---|---|
| A | 1999 | 0.7K | 19M |
| A | 2000 | 2K | 20M |
| B | 1999 | 37K | 172M |
| B | 2000 | 80K | 174M |
| C | 1999 | 212K | 1T |
| C | 2000 | 213K | 1T |

**pivot_wider(**data, names_from = "name", values_from = "value"**)**

The inverse of pivot_longer(). "Widen" data by expanding two columns into several. One column provides the new column names, the other the values.

pivot_wider(table2, names_from = type, values_from = count)

## Split Cells - Use these functions to split or combine cells into individual, isolated values.

table5

| country | century | year |
|---|---|---|
| A | 19 | 99 |
| A | 20 | 00 |
| B | 19 | 99 |
| B | 20 | 00 |

→

| country | year |
|---|---|
| A | 1999 |
| A | 2000 |
| B | 1999 |
| B | 2000 |

**unite(**data, col, …, sep = "_", remove = TRUE, na.rm = FALSE**)** Collapse cells across several columns into a single column.

unite(table5, century, year, col = "year", sep = "")

table3

| country | year | rate |
|---|---|---|
| A | 1999 | 0.7K/19M |
| A | 2000 | 2K/20M |
| B | 1999 | 37K/172M |
| B | 2000 | 80K/174M |

→

| country | year | cases | pop |
|---|---|---|---|
| A | 1999 | 0.7K | 19M |
| A | 2000 | 2K | 20M |
| B | 1999 | 37K | 172 |
| B | 2000 | 80K | 174 |

**separate(**data, col, into, sep = "[^[:alnum:]]+", remove = TRUE, convert = FALSE, extra = "warn", fill = "warn", …**)** Separate each cell in a column into several columns. Also **extract()**.

separate(table3, rate, sep = "/", into = c("cases", "pop"))

table3

| country | year | rate |
|---|---|---|
| A | 1999 | 0.7K/19M |
| A | 2000 | 2K/20M |
| B | 1999 | 37K/172M |
| B | 2000 | 80K/174M |

→

| country | year | rate |
|---|---|---|
| A | 1999 | 0.7K |
| A | 1999 | 19M |
| A | 2000 | 2K |
| A | 2000 | 20M |
| B | 1999 | 37K |
| B | 1999 | 172M |
| B | 2000 | 80K |
| B | 2000 | 174M |

**separate_rows(**data, …, sep = "[^[:alnum:].]+", convert = FALSE**)** Separate each cell in a column into several rows.

separate_rows(table3, rate, sep = "/")

## Expand Tables

Create new combinations of variables or identify implicit missing values (combinations of variables not present in the data).

x

| x1 | x2 | x3 |
|---|---|---|
| A | 1 | 3 |
| B | 1 | 4 |
| B | 2 | 3 |

→

| x1 | x2 |
|---|---|
| A | 1 |
| A | 2 |
| B | 1 |
| B | 2 |

**expand(**data, …**)** Create a new tibble with all possible combinations of the values of the variables listed in … Drop other variables.
expand(mtcars, cyl, gear, carb)

x

| x1 | x2 | x3 |
|---|---|---|
| A | 1 | 3 |
| B | 1 | 4 |
| B | 2 | 3 |

→

| x1 | x2 | x3 |
|---|---|---|
| A | 1 | 3 |
| A | 2 | NA |
| B | 1 | 4 |
| B | 2 | 3 |

**complete(**data, …, fill = list()**)** Add missing possible combinations of values of variables listed in … Fill remaining variables with NA.
complete(mtcars, cyl, gear, carb)

## Handle Missing Values

Drop or replace explicit missing values (NA).

x

| x1 | x2 |
|---|---|
| A | 1 |
| B | NA |
| C | NA |
| D | 3 |
| E | NA |

→

| x1 | x2 |
|---|---|
| A | 1 |
| D | 3 |

**drop_na(**data, …**)** Drop rows containing NA's in … columns.
drop_na(x, x2)

x

| x1 | x2 |
|---|---|
| A | 1 |
| B | NA |
| C | NA |
| D | 3 |
| E | NA |

→

| x1 | x2 |
|---|---|
| A | 1 |
| B | 1 |
| C | 1 |
| D | 3 |
| E | 3 |

**fill(**data, …, .direction = "down"**)** Fill in NA's in … columns using the next or previous value.
fill(x, x2)

x

| x1 | x2 |
|---|---|
| A | 1 |
| B | NA |
| C | NA |
| D | 3 |
| E | NA |

→

| x1 | x2 |
|---|---|
| A | 1 |
| B | 2 |
| C | 2 |
| D | 3 |
| E | 2 |

**replace_na(**data, replace**)** Specify a value to replace NA in selected columns.
replace_na(x, list(x2 = 2))

# Data tidying with tidyr : : **CHEAT SHEET**

**Tidy data** is a way to organize tabular data in a consistent data structure across packages.
A table is tidy if:


Each **variable** is in its own **column**

Each **observation**, or **case**, is in its own row


Access **variables** as **vectors**

Preserve **cases** in vectorized operations

## Tibbles


### AN ENHANCED DATA FRAME

Tibbles are a table format provided by the **tibble** package. They inherit the data frame class, but have improved behaviors:

- **Subset** a new tibble with ], a vector with [[ and $.
- **No partial matching** when subsetting columns.
- **Display** concise views of the data on one screen.

**options(**tibble.print_max = n, tibble.print_min = m, tibble.width = Inf**)** Control default display settings.

**View()** or **glimpse()** View the entire data set.

### CONSTRUCT A TIBBLE

**tibble(**…**)** Construct by columns.
tibble(x = 1:3, y = c("a", "b", "c"))

**tribble(**…**)** Construct by rows.
tribble(~x, ~y,
1, "a",
2, "b",
3, "c")

**Both make this tibble**

```
A tibble: 3 × 2
      x     y
  <int> <chr>
1     1     a
2     2     b
3     3     c
```

**as_tibble(**x, …**)** Convert a data frame to a tibble.

**enframe(**x, name = "name", value = "value"**)** Convert a named vector to a tibble. Also **deframe()**.

**is_tibble(**x**)** Test whether x is a tibble.

## Reshape Data - Pivot data to reorganize values into a new layout.

table4a

| country | 1999 | 2000 |
|---|---|---|
| A | 0.7K | 2K |
| B | 37K | 80K |
| C | 212K | 213K |

→

| country | year | cases |
|---|---|---|
| A | 1999 | 0.7K |
| B | 1999 | 37K |
| C | 1999 | 212K |
| A | 2000 | 2K |
| B | 2000 | 80K |
| C | 2000 | 213K |

**pivot_longer(**data, cols, names_to = "name", values_to = "value", values_drop_na = FALSE**)**

"Lengthen" data by collapsing several columns into two. Column names move to a new names_to column and values to a new values_to column.

pivot_longer(table4a, cols = 2:3, names_to ="year", values_to = "cases")

table2

| country | year | type | count |
|---|---|---|---|
| A | 1999 | cases | 0.7K |
| A | 1999 | pop | 19M |
| A | 2000 | cases | 2K |
| A | 2000 | pop | 20M |
| B | 1999 | cases | 37K |
| B | 1999 | pop | 172M |
| B | 2000 | cases | 80K |
| B | 2000 | pop | 174M |
| C | 1999 | cases | 212K |
| C | 1999 | pop | 1T |
| C | 2000 | cases | 213K |
| C | 2000 | pop | 1T |

→

| country | year | cases | |
|---|---|---|---|
| A | 1999 | 0.7K | 1 |
| A | 2000 | 2K | 20M |
| B | 1999 | 37K | 172M |
| B | 2000 | 80K | 174M |
| C | 1999 | 212K | 1T |
| C | 2000 | 213K | 1T |

**pivot_wider(**data, names_from = "name", values_from = "value"**)**

The inverse of pivot_longer(). "Widen" data by expanding two columns into several. One column provides the new column names, the other the values.

pivot_wider(table2, names_from = type, values_from = count)

## Split Cells - Use these functions to split or combine cells into individual, isolated values.

table5

| country | century | year |
|---|---|---|
| A | 19 | 99 |
| A | 20 | 00 |
| B | 19 | 99 |
| B | 20 | 00 |

→

| country | year |
|---|---|
| A | 1999 |
| A | 2000 |
| B | 1999 |
| B | 2000 |

**unite(**data, col, …, sep = "_", remove = TRUE, na.rm = FALSE**)** Collapse cells across several columns into a single column.

unite(table5, century, year, col = "year", sep = "")

table3

| country | year | rate |
|---|---|---|
| A | 1999 | 0.7K/19M |
| A | 2000 | 2K/20M |
| B | 1999 | 37K/172M |
| B | 2000 | 80K/174M |

→

| country | year | cases | pop |
|---|---|---|---|
| A | 1999 | 0.7K | 19 |
| A | 2000 | 2K | 20 |
| B | 1999 | 37K | 172 |
| B | 2000 | 80K | 174 |

**separate(**data, col, into, sep = "[^[:alnum:]]+", remove = TRUE, convert = FALSE, extra = "warn", fill = "warn", …**)** Separate each cell in a column into several columns. Also **extract()**.

separate(table3, rate, sep = "/", into = c("cases", "pop"))

table3

| country | year | rate |
|---|---|---|
| A | 1999 | 0.7K/19M |
| A | 2000 | 2K/20M |
| B | 1999 | 37K/172M |
| B | 2000 | 80K/174M |

→

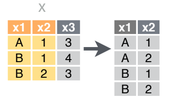
| country | year | rate |
|---|---|---|
| A | 1999 | 0.7K |
| A | 1999 | 19M |
| A | 2000 | 2K |
| A | 2000 | 20M |
| B | 1999 | 37K |
| B | 1999 | 172M |
| B | 2000 | 80K |
| B | 2000 | 174M |

**separate_rows(**data, …, sep = "[^[:alnum:].]+", convert = FALSE**)** Separate each cell in a column into several rows.

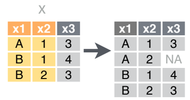
separate_rows(table3, rate, sep = "/")

## Expand Tables

Create new combinations of variables or identify implicit missing values (combinations of variables not present in the data).



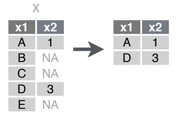

**expand(**data, …**)** Create a new tibble with all possible combinations of the values of the variables listed in … Drop other variables.
expand(mtcars, cyl, gear, carb)



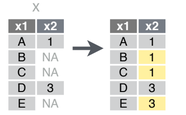

**complete(**data, …, fill = list()**)** Add missing possible combinations of values of variables listed in … Fill remaining variables with NA.
complete(mtcars, cyl, gear, carb)

## Handle Missing Values

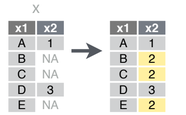
Drop or replace explicit missing values (NA).




**drop_na(**data, …**)** Drop rows containing NA's in … columns.
drop_na(x, x2)




**fill(**data, …, .direction = "down"**)** Fill in NA's in … columns using the next or previous value.
fill(x, x2)




**replace_na(**data, replace**)** Specify a value to replace NA in selected columns.
replace_na(x, list(x2 = 2))

# Introduction: **stringr**

- Collection of functions to work with character strings
- Output of stringr functions is a vector of TRUE/FALSE

Sections:

- Detect matches
- Subset strings
- Manage lengths
- Mutate strings
- Join and split
- Order strings
- ***Regular expressions***


*Output of* `dplyr::filter()`
*is an object (e.g. dataframe)*

Can use stringr functions
**in combination with** dplyr!

# stringr - detect matches

```
stringr::str_detect(string, pattern)
```

**Keep all hair colors that mention brown**

*Hint: combine `filter()` and `stringr()`*

```
stringr::str_detect(starwars$hair_color, "brown")
```

```
  [1] FALSE    NA    NA FALSE  TRUE  TRUE  TRUE    NA FALSE FALSE FALSE FALSE  TRUE  TRUE    NA    NA
 [38] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [75] FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE  TRUE
```

```
starwars %>%
  dplyr::filter(stringr::str_detect(hair_color, "brown"))
```

| name | height | mass | hair_color |
|---|---|---|---|
| Leia Organa | 150 | 49 | brown |
| Owen Lars | 178 | 120 | brown, grey |
| Beru Whitesun lars | 165 | 75 | brown |
| Chewbacca | 228 | 112 | brown |
| Han Solo | 180 | 80 | brown |
| Wedge Antilles | 170 | 77 | brown |

# stringr - detect matches

```
stringr::str_count(string, pattern)
```

**Make a new column counting how many ‘a’s are in each character’s name**

```
starwars %>%
  dplyr::mutate(letter_a = stringr::str_count(name, “a”))
```

**Make a new column counting how many vowels are in each character’s name**

*Hint: “this|that” reads “this **OR** that”*

```
starwars %>%
  dplyr::mutate(vowels = stringr::str_count(name, “a|e|i|o|u”))
```

| name | vowels |
|---|---|
| Luke Skywalker | 4 |
| C-3PO | 0 |
| R2-D2 | 0 |
| Darth Vader | 3 |
| Leia Organa | ~~5~~ 6 |

# stringr - detect matches

```
stringr::str_count(string, pattern)
```

**Make a new column counting how many ‘a’s are in each character’s name**

```
starwars %>%
  dplyr::mutate(letter_a = stringr::str_count(name, “a”))
```

**Make a new column counting how many vowels are in each character’s name**

*Hint: “this|that” reads “this **OR** that”*

```
starwars %>%
  dplyr::mutate(vowels = stringr::str_count(name,
                                            “a|e|i|o|u|A|E|I|O|U”))
```

| name | vowels |
|---|---|
| Luke Skywalker | 4 |
| C-3PO | 1 |
| R2-D2 | 0 |
| Darth Vader | 3 |
| Leia Organa | 6 |

# stringr - mutate strings

```
stringr::str_to_lower(string)
```

**Make a new column counting how many vowels are in each character's name**

*hint: change all names to lower case first*

```
starwars %>%
  dplyr::mutate(name = stringr::str_to_lower(name)) %>%
  dplyr::mutate(vowels = stringr::str_count(name, "a|e|i|o|u"))
```

```
stringr::str_to_upper(string)
```

**Change string to all uppercase**

| name | vowels |
|---|---|
| luke skywalker | 4 |
| c-3po | 1 |
| r2-d2 | 0 |
| darth vader | 3 |
| leia organa | 6 |

```
stringr::str_to_title(string)
```

**Change string to "title" case (First Letter Upper)**

# stringr - mutate strings

```
stringr::str_replace(string, pattern, replacement)
```

**Replace all mentions of "Human" with "Homo sapien"**

```
species <- starwars %>%
  dplyr::mutate(species =
         stringr::str_replace(species, "Human", "Homo sapiens"))
```

| name | species |
|---|---|
| Luke Skywalker | Homo sapiens |
| C-3PO | Droid |
| R2-D2 | Droid |
| Darth Vader | Homo sapiens |
| Leia Organa | Homo sapiens |
| Owen Lars | Homo sapiens |
| Beru Whitesun lars | Homo sapiens |
| R5-D4 | Droid |
| Biggs Darklighter | Homo sapiens |
| Obi-Wan Kenobi | Homo sapiens |
| Anakin Skywalker | Homo sapiens |

# stringr - join and split

```
stringr::str_c(string1, string2)
```

**Give each starwars character a PhD**

*hint: add ", PhD" to the end of the name column*

```
doctorates <- starwars %>%
  dplyr::mutate(name = stringr::str_c(name, ", PhD"))
```

| name |
| --- |
| Luke Skywalker, PhD |
| C-3PO, PhD |
| R2-D2, PhD |
| Darth Vader, PhD |
| Leia Organa, PhD |
| Owen Lars, PhD |


**Compare to `paste()`**

```
doctorates2 <- starwars %>%
  dplyr::mutate(name = paste(name, ", PhD", sep = ""))
```

# stringr - join and split

```
stringr::str_split_fixed(string, pattern, n)
```

**Split “name” into “first name” and “last name”**

*hint: try `str_split_fixed(“Luke Skywalker”, “ “, 2)` first to see output*

```
> stringr::str_split_fixed("Luke Skywalker", " ", 2)
     [,1]   [,2]
[1,] "Luke" "Skywalker"
```

```
names <- starwars %>%
  dplyr::mutate(first_name = stringr::str_split_fixed(name, " ", 2)[,1],
                last_name = stringr::str_split_fixed(name, " ", 2)[,2])
```

**Compare to `tidyr::separate()`**


```
names <- starwars %>%
  tidyr::separate(name, c(“first_name”, “last_name”), sep = “ “)
```

# stringr - join and split

```
stringr::str_split_fixed(string, pattern, n)
```

**stringr::str_split_fixed()**

| name | first_name | last_name |
|---|---|---|
| Luke Skywalker | Luke | Skywalker |
| C-3PO | C-3PO | |
| R2-D2 | R2-D2 | |
| Darth Vader | Darth | Vader |
| Leia Organa | Leia | Organa |
| Owen Lars | Owen | Lars |
| Beru Whitesun lars | Beru | Whitesun lars |
| R5-D4 | R5-D4 | |
| Biggs Darklighter | Biggs | Darklighter |
| Obi-Wan Kenobi | Obi-Wan | Kenobi |

**tidyr::separate()**

| first_name | last_name | |
|---|---|---|
| Luke | Skywalker | *NA* |
| C | 3PO | *NA* |
| R2 | D2 | *NA* |
| Darth | Vader | *NA* |
| Leia | Organa | *NA* |
| Owen | Lars | *NA* |
| Beru | Whitesun | lars |
| R5 | D4 | *NA* |
| Biggs | Darklighter | *NA* |
| Obi | Wan | Kenobi |

# stringr - manage lengths

```
stringr::str_length(string)
```

**Add a new column that counts the number of letters in each name**

```
counts <- starwars %>%
  dplyr::mutate(name_counts = stringr::str_length(name))
```

```
stringr::str_trim(string, side = c("left", "right", "both"))
```

**Remove whitespace surrounding " test "**

```
stringr::str_trim(" test ")
```

# stringr - regular expressions

## Regular Expressions - Regular expressions, or *regexps*, are a concise language for describing patterns in strings.

### MATCH CHARACTERS

see <- function(rx) str_view_all("abc ABC 123\t.!?\\(){}\n", rx)

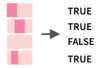
| **string** (type this) | **regexp** (to mean this) | **matches** (which matches this) | **example** | |
|---|---|---|---|---|
| | **a (etc.)** | a (etc.) | see("a") | abc ABC 123 .!?\(){} |
| \\. | \. | . | see("\\.") | abc ABC 123 .!?\(){} |
| \\! | \! | ! | see("\\!") | abc ABC 123 .!?\(){} |
| \\? | \? | ? | see("\\?") | abc ABC 123 .!?\(){} |
| \\\\ | \\ | \ | see("\\\\") | abc ABC 123 .!?\(){} |
| \\( | \( | ( | see("\\(") | abc ABC 123 .!?\(){} |
| \\) | \) | ) | see("\\)") | abc ABC 123 .!?\(){} |
| \\{ | \{ | { | see("\\{") | abc ABC 123 .!?\(){} |
| \\} | \} | } | see( "\\}") | abc ABC 123 .!?\(){} |
| \\n | \n | new line (return) | see("\\n") | abc ABC 123 .!?\(){} |
| \\t | \t | tab | see("\\t") | abc ABC 123 .!?\(){} |
| \\s | \s | any whitespace *(\S for non-whitespaces)* | see("\\s") | abc ABC 123 .!?\(){} |
| \\d | \d | any digit *(\D for non-digits)* | see("\\d") | abc ABC 123 .!?\(){} |
| \\w | \w | any word character *(\W for non-word chars)* | see("\\w") | abc ABC 123 .!?\(){} |
| \\b | \b | word boundaries | see("\\b") | abc ABC 123 .!?\(){} |
| | [:digit:] [1] | digits | see("[:digit:]") | abc ABC 123 .!?\(){} |
| | [:alpha:] [1] | letters | see("[:alpha:]") | abc ABC 123 .!?\(){} |
| | [:lower:] [1] | lowercase letters | see("[:lower:]") | abc ABC 123 .!?\(){} |
| | [:upper:] [1] | uppercase letters | see("[:upper:]") | abc ABC 123 .!?\(){} |
| | [:alnum:] [1] | letters and numbers | see("[:alnum:]") | abc ABC 123 .!?\(){} |
| | [:punct:] [1] | punctuation | see("[:punct:]") | abc ABC 123 .!?\(){} |
| | [:graph:] [1] | letters, numbers, and punctuation | see("[:graph:]") | abc ABC 123 .!?\(){} |
| | [:space:] [1] | space characters (i.e. \s) | see("[:space:]") | abc ABC 123 .!?\(){} |
| | [:blank:] [1] | space and tab (but not new line) | see("[:blank:]") | abc ABC 123 .!?\(){} |
| | . | every character except a new line | see(".") | abc ABC 123 .!?\(){} |

[1] Many base R functions require classes to be wrapped in a second set of [ ], e.g. **[[:digit:]]**

# String manipulation with stringr : : **CHEAT SHEET**

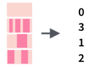
The **stringr** package provides a set of internally consistent tools for working with character strings, i.e. sequences of characters surrounded by quotation marks.

## Detect Matches




**str_detect**(string, **pattern**) Detect the presence of a pattern match in a string.
*str_detect(fruit, "a")*


**str_which**(string, **pattern**) Find the indexes of strings that contain a pattern match.
*str_which(fruit, "a")*




**str_count**(string, **pattern**) Count the number of matches in a string.
*str_count(fruit, "a")*


**str_locate**(string, **pattern**) Locate the positions of pattern matches in a string. Also **str_locate_all**. *str_locate(fruit, "a")*

## Subset Strings

**str_sub**(string, start = 1L, end = -1L) Extract substrings from a character vector.
*str_sub(fruit, 1, 3); str_sub(fruit, -2)*

**str_subset**(string, **pattern**) Return only the strings that contain a pattern match.
*str_subset(fruit, "b")*


**str_extract**(string, **pattern**) Return the first pattern match found in each string, as a vector. Also **str_extract_all** to return every pattern match. *str_extract(fruit, "[aeiou]")*


**str_match**(string, **pattern**) Return the first pattern match found in each string, as a matrix with a column for each ( ) group in pattern. Also **str_match_all**.
*str_match(sentences, "(a|the) ([^ ]+)")*

## Manage Lengths

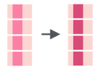

**str_length**(string) The width of strings (i.e. number of code points, which generally equals the number of characters). *str_length(fruit)*

**str_pad**(string, width, side = c("left", "right", "both"), pad = " ") Pad strings to constant width. *str_pad(fruit, 17)*

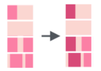
**str_trunc**(string, width, side = c("right", "left", "center"), ellipsis = "...") Truncate the width of strings, replacing content with ellipsis.
*str_trunc(fruit, 3)*

**str_trim**(string, side = c("both", "left", "right")) Trim whitespace from the start and/or end of a string. *str_trim(fruit)*

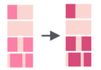
## Mutate Strings



**str_sub**() **<-** value. Replace substrings by identifying the substrings with str_sub() and assigning into the results.
*str_sub(fruit, 1, 3) <- "str"*



**str_replace**(string, **pattern**, replacement) Replace the first matched pattern in each string. *str_replace(fruit, "a", "-")*



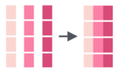
**str_replace_all**(string, **pattern**, replacement) Replace all matched patterns in each string. *str_replace_all(fruit, "a", "-")*


**str_to_lower**(string, locale = "en")[1] Convert strings to lower case.
*str_to_lower(sentences)*

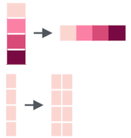

**str_to_upper**(string, locale = "en")[1] Convert strings to upper case.
*str_to_upper(sentences)*


**str_to_title**(string, locale = "en")[1] Convert strings to title case. *str_to_title(sentences)*

## Join and Split



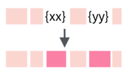
**str_c**(..., sep = "", collapse = NULL) Join multiple strings into a single string.
*str_c(letters, LETTERS)*



**str_c**(..., sep = "", **collapse = ""**) Collapse a vector of strings into a single string.
*str_c(letters, collapse = "")*

**str_dup**(string, times) Repeat strings times times. *str_dup(fruit, times = 2)*

**str_split_fixed**(string, **pattern**, n) Split a vector of strings into a matrix of substrings (splitting at occurrences of a pattern match). Also **str_split** to return a list of substrings.
*str_split_fixed(fruit, " ", n=2)*




**str_glue**(…, .sep = "", .envir = parent.frame()) Create a string from strings and {expressions} to evaluate. *str_glue("Pi is {pi}")*

**str_glue_data**(.x, ..., .sep = "", .envir = parent.frame(), .na = "NA") Use a data frame, list, or environment to create a string from strings and {expressions} to evaluate.
*str_glue_data(mtcars, "{rownames(mtcars)} has {hp} hp")*

## Order Strings


**str_order**(x, decreasing = FALSE, na_last = TRUE, locale = "en", numeric = FALSE, ...)[1] Return the vector of indexes that sorts a character vector. *x[str_order(x)]*

**str_sort**(x, decreasing = FALSE, na_last = TRUE, locale = "en", numeric = FALSE, ...)[1] Sort a character vector.
*str_sort(x)*

## Helpers

```
apple
banana
pear
```

**str_conv**(string, encoding) Override the encoding of a string. *str_conv(fruit,"ISO-8859-1")*

```
apple
banana
pear
```

**str_view**(string, **pattern**, match = NA) View HTML rendering of first regex match in each string. *str_view(fruit, "[aeiou]")*

**str_view_all**(string, **pattern**, match = NA) View HTML rendering of all regex matches.
*str_view_all(fruit, "[aeiou]")*

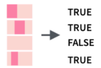
**str_wrap**(string, width = 80, indent = 0, exdent = 0) Wrap strings into nicely formatted paragraphs. *str_wrap(sentences, 20)*

[1] See bit.ly/ISO639-1 for a complete list of locales.

# String manipulation with stringr : : **CHEAT SHEET**

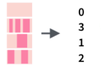
The **stringr** package provides a set of internally consistent tools for working with character strings, i.e. sequences of characters surrounded by quotation marks.

## Detect Matches




**str_detect**(string, **pattern**) Detect the presence of a pattern match in a string. *str_detect(fruit, "a")*


**str_which**(string, **pattern**) Find the indexes of strings that contain a pattern match. *str_which(fruit, "a")*




**str_count**(string, **pattern**) Count the number of matches in a string. *str_count(fruit, "a")*


**str_locate**(string, **pattern**) Locate the positions of pattern matches in a string. Also **str_locate_all**. *str_locate(fruit, "a")*

## Subset Strings

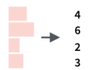
**str_sub**(string, start = 1L, end = -1L) Extract substrings from a character vector. *str_sub(fruit, 1, 3); str_sub(fruit, -2)*

**str_subset**(string, **pattern**) Return only the strings that contain a pattern match. *str_subset(fruit, "b")*


**str_extract**(string, **pattern**) Return the first pattern match found in each string, as a vector. Also **str_extract_all** to return every pattern match. *str_extract(fruit, "[aeiou]")*


**str_match**(string, **pattern**) Return the first pattern match found in each string, as a matrix with a column for each ( ) group in pattern. Also **str_match_all**. *str_match(sentences, "(a|the) ([^ ]+)")*

## Manage Lengths



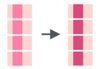

**str_length**(string) The width of strings (i.e. number of code points, which generally equals the number of characters). *str_length(fruit)*

**str_pad**(string, width, side = c("left", "right", "both"), pad = " ") Pad strings to constant width. *str_pad(fruit, 17)*

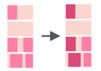
**str_trunc**(string, width, side = c("right", "left", "center"), ellipsis = "...") Truncate the width of strings, replacing content with ellipsis. *str_trunc(fruit, 3)*

**str_trim**(string, side = c("both", "left", "right")) Trim whitespace from the start and/or end of a string. *str_trim(fruit)*

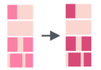
## Mutate Strings



**str_sub**() **<-** value. Replace substrings by identifying the substrings with str_sub() and assigning into the results. *str_sub(fruit, 1, 3) <- "str"*



**str_replace**(string, **pattern**, replacement) Replace the first matched pattern in each string. *str_replace(fruit, "a", "-")*



**str_replace_all**(string, **pattern**, replacement) Replace all matched patterns in each string. *str_replace_all(fruit, "a", "-")*

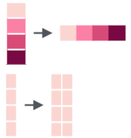

**str_to_lower**(string, locale = "en")[1] Convert strings to lower case. *str_to_lower(sentences)*


**str_to_upper**(string, locale = "en")[1] Convert strings to upper case. *str_to_upper(sentences)*


**str_to_title**(string, locale = "en")[1] Convert strings to title case. *str_to_title(sentences)*

## Join and Split

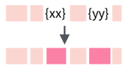
**str_c**(..., sep = "", collapse = NULL) Join multiple strings into a single string. *str_c(letters, LETTERS)*



**str_c**(..., sep = "", **collapse = ""**) Collapse a vector of strings into a single string. *str_c(letters, collapse = "")*

**str_dup**(string, times) Repeat strings times times. *str_dup(fruit, times = 2)*

**str_split_fixed**(string, **pattern**, n) Split a vector of strings into a matrix of substrings (splitting at occurrences of a pattern match). Also **str_split** to return a list of substrings. *str_split_fixed(fruit, " ", n=2)*




**str_glue**(…, .sep = "", .envir = parent.frame()) Create a string from strings and {expressions} to evaluate. *str_glue("Pi is {pi}")*

**str_glue_data**(.x, ..., .sep = "", .envir = parent.frame(), .na = "NA") Use a data frame, list, or environment to create a string from strings and {expressions} to evaluate. *str_glue_data(mtcars, "{rownames(mtcars)} has {hp} hp")*

## Order Strings


**str_order**(x, decreasing = FALSE, na_last = TRUE, locale = "en", numeric = FALSE, ...)[1] Return the vector of indexes that sorts a character vector. *x[str_order(x)]*

**str_sort**(x, decreasing = FALSE, na_last = TRUE, locale = "en", numeric = FALSE, ...)[1] Sort a character vector. *str_sort(x)*

## Helpers

```
apple
banana
pear
```

**str_conv**(string, encoding) Override the encoding of a string. *str_conv(fruit,"ISO-8859-1")*

```
apple
banana
pear
```

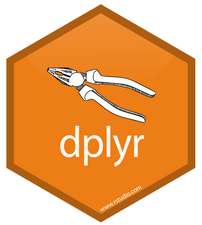
**str_view**(string, **pattern**, match = NA) View HTML rendering of first regex match in each string. *str_view(fruit, "[aeiou]")*

**str_view_all**(string, **pattern**, match = NA) View HTML rendering of all regex matches. *str_view_all(fruit, "[aeiou]")*

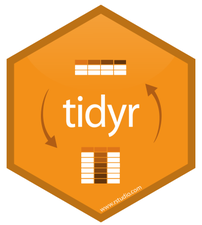
**str_wrap**(string, width = 80, indent = 0, exdent = 0) Wrap strings into nicely formatted paragraphs. *str_wrap(sentences, 20)*

[1] See bit.ly/ISO639-1 for a complete list of locales.

# Overview of Tidyverse




The **dplyr** package is the most useful package in R for data manipulation. One of the greatest advantages of the package is that you can use the pipe function (%>%) to combine different functions.




The **tidyr** package complements dplyr perfectly. It boosts the power of dplyr for data manipulation and pre-processing


The **stringr** package is used for strings. It provides a cohesive set of functions designed to make working with strings as easy as possible.

# Other Tidyverse packages


Data scientists universally love using **ggplot2** to produce their charts and visualizations!


The **lubridate** package is the best way to deal with dates and times in R! From converting strings to dates to calculating hours between two time points.


The **purrr** package in R provides a complete toolkit for enhancing R's functional programming. We can use the functions provide by purrr to avoid many loops with just one line of code.


The **forcats** package is dedicated to dealing with categorical variables or factors.


The **broom** package takes the messy output of built-in functions in R and turns them into tidy dataframes

# Tidyverse resources

- https://www.tidyverse.org
- https://www.rstudio.com/resources/cheatsheets
- https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html
- https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html

# Tidyverse