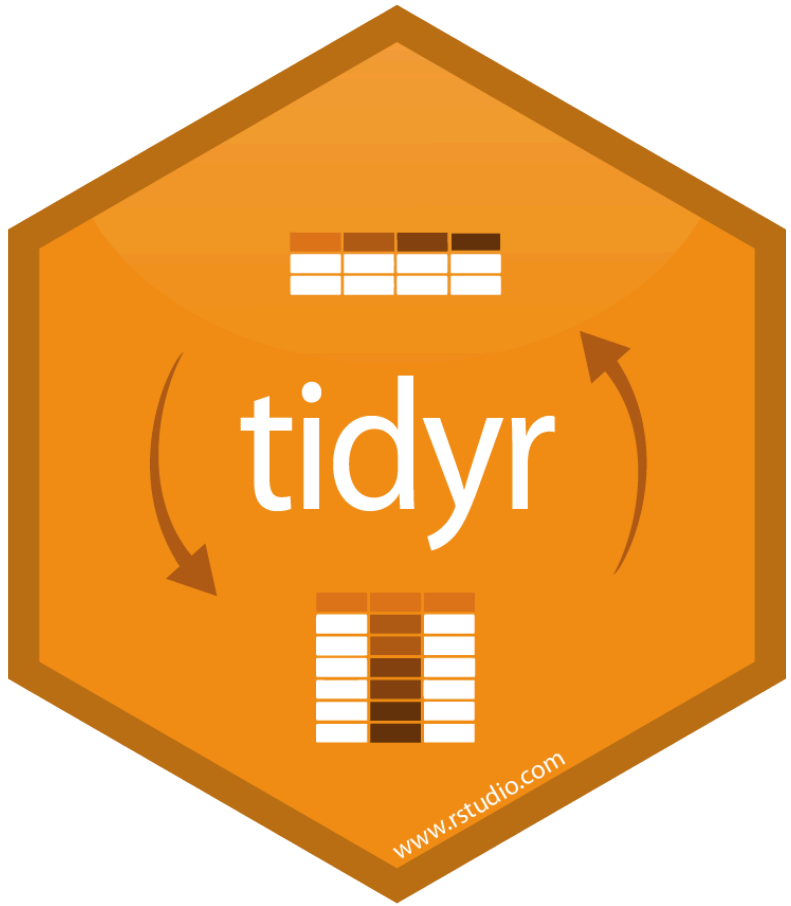
tidyr

Lecture 07

Dr. Colin Rundel


tidyr
www.rstudio.com

Reshaping data (Wide vs. Long)

Wide -> Long

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` (previously `gather`)

From Data tidying with tidyr

Syntax

```
(d = tibble::tribble(
  ~country, ~"1999",  ~"2000",
        "A", "0.7K",     "2K",
        "B",  "37K",    "80K",
        "C", "212K",   "213K"
))
```

```
# A tibble: 3 × 3
  country `1999` `2000`
  <chr>   <chr>  <chr>
1 A       0.7K   2K
2 B       37K    80K
3 C       212K   213K
```

```
pivot_longer(
  d,
  cols = "1999":"2000",
  names_to = "year",
  values_to = "cases"
)
```

```
# A tibble: 6 × 3
  country year  cases
  <chr>   <chr> <chr>
1 A       1999  0.7K
2 A       2000  2K
3 B       1999  37K
4 B       2000  80K
5 C       1999  212K
6 C       2000  213K
```

Long -> Wide

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` (previously `spread`)

From Data tidying with tidyr

Syntax

```
( d = tibble::tribble(
    ~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"
  )
)
```

```
# A tibble: 12 × 4
   country  year type  count
   <chr>   <dbl> <chr> <chr>
 1 A        1999 cases "0.7K"
 2 A        1999 pop   "19M"
 3 A        2000 cases "2K"
 4 A        2000 pop   "20M"
 5 B        1999 cases "37K"
 6 B        1999 pop   "172M"
 7 B        2000 cases " 80K"
 8 B        2000 pop   "174M"
 9 C        1999 cases "212K"
10 C        1999 pop   "1T"
11 C        2000 cases "213K"
12 C        2000 pop   "1T"
```

```
pivot_wider(
  d,
  id_cols = country:year,
  names_from = type,
  values_from = count
)
```

```
# A tibble: 6 × 4
  country  year cases  pop
  <chr>   <dbl> <chr>  <chr>
1 A        1999 "0.7K" 19M
2 A        2000 "2K"   20M
3 B        1999 "37K"  172M
4 B        2000 " 80K" 174M
5 C        1999 "212K" 1T
6 C        2000 "213K" 1T
```

Exercise 1

The `palmerpenguin` package contains measurement data on various penguin species on islands near Palmer Station in Antarctica. The code below shows the # of each species measured on each of the three islands (missing island, penguin pairs implies that species does not occur on that island).

```
palmerpenguins::penguins |>
  count(island, species)
```

```
# A tibble: 5 × 3
  island    species       n
  <fct>     <fct>     <int>
1 Biscoe    Adelie       44
2 Biscoe    Gentoo      124
3 Dream     Adelie       56
4 Dream     Chinstrap    68
5 Torgersen Adelie       52
```

Starting from these data construct a contingency table of counts for island (rows) by species (columns) using the pivot functions we've just discussed.

05:00

Separate - wider

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

```r
separate_wider_delim(d, rate, delim = "/", names = c("cases", "pop"))
```

```
# A tibble: 6 × 4
  country  year cases pop
  <chr>   <dbl> <chr> <chr>
1 A        1999 0.7K  19M
2 A        2000 2K    20M
3 B        1999 37K   172M
4 B        2000 80K   174M
5 C        1999 212K  1T
6 C        2000 213K  1T
```

From Data tidying with tidyr

Separate - longer

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

```
1 separate_longer_delim(d, rate, delim = "/")
```

```
# A tibble: 12 × 3
   country  year rate
   <chr>   <dbl> <chr>
 1 A        1999 0.7K
 2 A        1999 19M
 3 A        2000 2K
 4 A        2000 20M
 5 B        1999 37K
 6 B        1999 172M
 7 B        2000 80K
 8 B        2000 174M
 9 C        1999 212K
10 C        1999 1T
11 C        2000 213K
12 C        2000 1T
```

From Data tidying with tidyr

Other separates

In previous versions of tidyr there was a single catch-all `separate()` function. This still exists and is available in the package but it is **superseded**.

Other helpful separate functions:

- `separate_longer_position()`
- `separate_wider_position()`
- `separate_wider_regex()`

Unite

country	century	year
Afghan	19	99
Afghan	20	0
Brazil	19	99
Brazil	20	0
China	19	99
China	20	0

country	year
Afghan	1999
Afghan	2000
Brazil	1999
Brazil	2000
China	1999
China	2000

```
1 unite(d, century, year, col = "year", sep = "")
```

```
# A tibble: 6 × 2
  country year
  <chr>   <chr>
1 Afghan  1999
2 Afghan  2000
3 Brazil  1999
4 Brazil  2000
5 China   1999
6 China   2000
```

From Data tidying with tidyr

Example 1 - tidy grades

Is the following data tidy?

```
grades = tibble::tribble(
  ~name,   ~hw_1, ~hw_2, ~hw_3, ~hw_4, ~proj_1, ~proj_2,
  "Alice",    19,    19,    18,    20,      89,      95,
  "Bob",      18,    20,    18,    16,      77,      88,
  "Carol",    18,    20,    18,    17,      96,      99,
  "Dave",     19,    19,    18,    19,      86,      82
)
```

How would we calculate a final score based on the following formula,

$$\text{score} = 0.5 \frac{\sum_i \text{hw}_i}{80} + 0.5 \frac{\sum_j \text{proj}_j}{200}$$

Semi-tidy approach

```
grades |>
  mutate(
    hw_avg = (hw_1+hw_2+hw_3+hw_4)/4,
    proj_avg = (proj_1+proj_2)/2
  ) |>
  mutate(
    overall = 0.5*(proj_avg/100) + 0.5*(hw_avg/20)
  )
```

```
# A tibble: 4 × 10
  name   hw_1  hw_2  hw_3  hw_4 proj_1 proj_2 hw_avg proj_avg overall
  <chr> <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>    <dbl>   <dbl>
1 Alice    19    19    18    20     89     95   19        92     0.935
2 Bob      18    20    18    16     77     88   18        82.5   0.862
3 Carol    18    20    18    17     96     99   18.2      97.5   0.944
4 Dave     19    19    18    19     86     82   18.8      84     0.889
```

pivot_longer (Wide -> Long)

```
tidyr::pivot_longer(
  grades,
  cols = hw_1:proj_2,
  names_to = "assignment",
  values_to = "score"
)
```

```
# A tibble: 24 × 3
   name  assignment score
   <chr> <chr>      <dbl>
 1 Alice hw_1          19
 2 Alice hw_2          19
 3 Alice hw_3          18
 4 Alice hw_4          20
 5 Alice proj_1        89
 6 Alice proj_2        95
 7 Bob   hw_1          18
 8 Bob   hw_2          20
 9 Bob   hw_3          18
10 Bob   hw_4          16
# ℹ 14 more rows
```

Split type and id

```
tidyr::pivot_longer(
  grades,
  cols = hw_1:proj_2,
  names_to = c("type", "id"),
  names_sep = "_",
  values_to = "score"
)
```

```
# A tibble: 24 × 4
   name  type  id    score
   <chr> <chr> <chr> <dbl>
 1 Alice hw    1        19
 2 Alice hw    2        19
 3 Alice hw    3        18
 4 Alice hw    4        20
 5 Alice proj  1        89
 6 Alice proj  2        95
 7 Bob   hw    1        18
 8 Bob   hw    2        20
 9 Bob   hw    3        18
10 Bob   hw    4        16
# ℹ 14 more rows
```

Tidy approach?

```
grades |>
  tidyr::pivot_longer(
    cols = hw_1:proj_2,
    names_to = c("type", "id"),
    names_sep = "_",
    values_to = "score"
  ) |>
  summarize(
    total = sum(score),
    .by = c(name, type)
  )
```

```
# A tibble: 8 × 3
  name  type  total
  <chr> <chr> <dbl>
1 Alice hw       76
2 Alice proj    184
3 Bob   hw       72
4 Bob   proj    165
5 Carol hw       73
6 Carol proj    195
7 Dave  hw       75
8 Dave  proj    168
```

pivot_wider - (Long -> Wide)

```
grades |>
  tidyr::pivot_longer(
    cols = hw_1:proj_2,
    names_to = c("type", "id"),
    names_sep = "_",
    values_to = "score"
  ) |>
  summarize(
    total = sum(score),
    .by = c(name, type)
  ) |>
  tidyr::pivot_wider(
    names_from = type,
    values_from = total
  )
```

```
# A tibble: 4 × 3
  name     hw  proj
  <chr> <dbl> <dbl>
1 Alice    76   184
2 Bob      72   165
3 Carol    73   195
4 Dave     75   168
```

Wrapping up

```
grades |>
  tidyr::pivot_longer(
    cols = hw_1:proj_2,
    names_to = c("type", "id"),
    names_sep = "_",
    values_to = "score"
  ) |>
  summarize(
    total = sum(score),
    .by = c(name, type)
  ) |>
  tidyr::pivot_wider(
    names_from = type,
    values_from = total
  ) |>
  mutate(
    score = 0.5*(hw/80) + 0.5*(proj/200
  )
```

```
# A tibble: 4 × 4
  name     hw  proj score
  <chr> <dbl> <dbl> <dbl>
1 Alice    76   184 0.935
2 Bob      72   165 0.862
3 Carol    73   195 0.944
4 Dave     75   168 0.889
```

Rectangling

Star Wars & repurrrsive

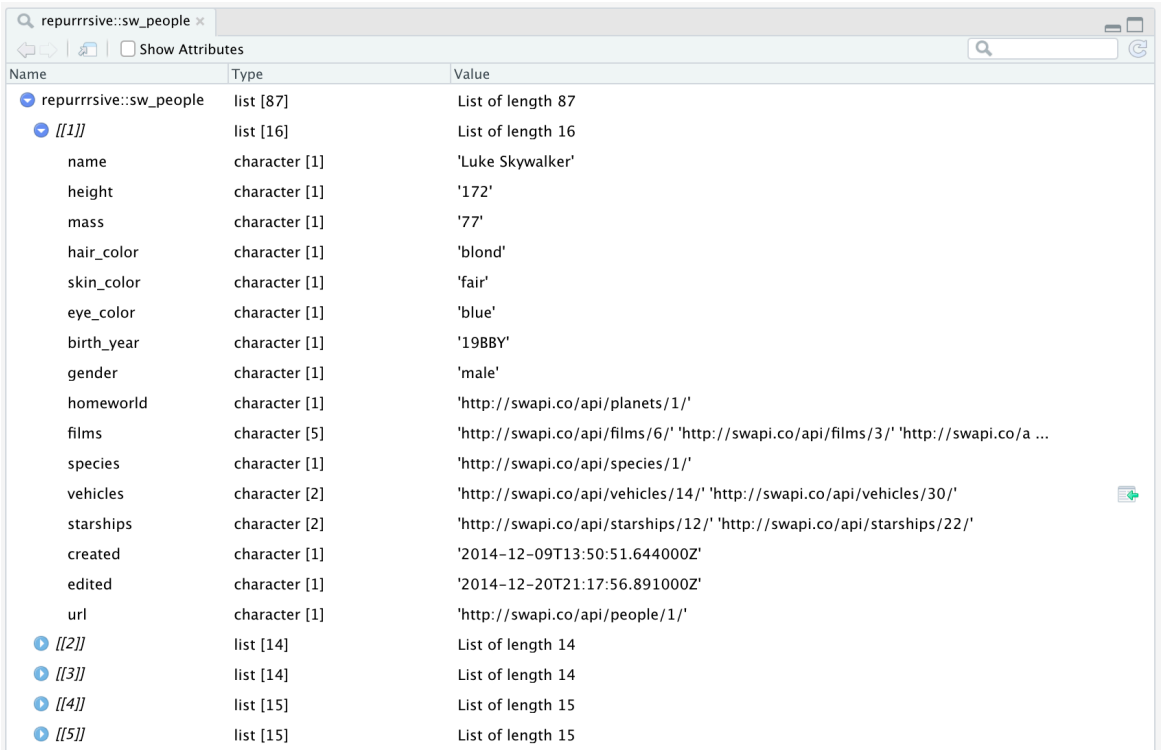
`repurrrsive` is a package that contains a number of interesting example data sets that are stored in a hierarchical format. Many come from web-based APIs which provide results as JSON.

```
1 str(repurrrsive::sw_people)
```

```
List of 87
 $ :List of 16
  ..$ name      : chr "Luke Skywalker"
  ..$ height    : chr "172"
  ..$ mass      : chr "77"
  ..$ hair_color: chr "blond"
  ..$ skin_color: chr "fair"
  ..$ eye_color : chr "blue"
  ..$ birth_year: chr "19BBY"
  ..$ gender    : chr "male"
  ..$ homeworld : chr "http://swapi.co/api/planets/1/"
  ..$ films     : chr [1:5] "http://swapi.co/api/films/6/"
"http://swapi.co/api/films/3/" "http://swapi.co/api/films/2/"
"http://swapi.co/api/films/1/" ...
```

RStudio data viewer

```
1 View(repurrrsive::sw_people)
```



repurrrsive::sw_people

Show Attributes

Name	Type	Value
repurrrsive::sw_people	list [87]	List of length 87
[[1]]	list [16]	List of length 16
name	character [1]	'Luke Skywalker'
height	character [1]	'172'
mass	character [1]	'77'
hair_color	character [1]	'blond'
skin_color	character [1]	'fair'
eye_color	character [1]	'blue'
birth_year	character [1]	'19BBY'
gender	character [1]	'male'
homeworld	character [1]	'http://swapi.co/api/planets/1/'
films	character [5]	'http://swapi.co/api/films/6/' 'http://swapi.co/api/films/3/' 'http://swapi.co/a ...
species	character [1]	'http://swapi.co/api/species/1/'
vehicles	character [2]	'http://swapi.co/api/vehicles/14/' 'http://swapi.co/api/vehicles/30/'
starships	character [2]	'http://swapi.co/api/starships/12/' 'http://swapi.co/api/starships/22/'
created	character [1]	'2014-12-09T13:50:51.644000Z'
edited	character [1]	'2014-12-20T21:17:56.891000Z'
url	character [1]	'http://swapi.co/api/people/1/'
[[2]]	list [14]	List of length 14
[[3]]	list [14]	List of length 14
[[4]]	list [15]	List of length 15
[[5]]	list [15]	List of length 15

Tidy data from nested lists

In addition to `pivot_*` the `tidyr` package also has a number of functions that are designed to aide in the tidying of hierarchical / nested data.

For today we will be discussing the `unnest_longer()`, and `unnest_wider()` functions and next week we will see `hoist()`.

Much like the functions we saw last time in `dplyr`, these functions are designed to work with data frames (which may seem odd at first).

List columns

We can make `sw_people` into a data frame by treating the original list as a single column in a data frame.

```
(sw_df = tibble::tibble(
  people = repurrrsive::sw_people
))
```

```
# A tibble: 87 × 1
   people
   <list>
 1 <named list [16]>
 2 <named list [14]>
 3 <named list [14]>
 4 <named list [15]>
 5 <named list [15]>
 6 <named list [14]>
 7 <named list [14]>
 8 <named list [14]>
 9 <named list [15]>
10 <named list [16]>
# ℹ 77 more rows
```

```
as.data.frame(sw_df) |> head()
```

```
people
1 Luke Skywalker, 172, 77, blond, fair, blue,
19BBY, male, http://swapi.co/api/planets/1/,
http://swapi.co/api/films/6/,
http://swapi.co/api/films/3/,
http://swapi.co/api/films/2/,
http://swapi.co/api/films/1/,
http://swapi.co/api/films/7/,
http://swapi.co/api/species/1/,
http://swapi.co/api/vehicles/14/,
http://swapi.co/api/vehicles/30/,
http://swapi.co/api/starships/12/,
http://swapi.co/api/starships/22/, 2014-12-
09T13:50:51.644000Z, 2014-12-20T21:17:56.891000Z,
http://swapi.co/api/people/1/
2
```

Unnesting

```
1 sw_df |>
2   unnest_wider(people)
```

```
# A tibble: 87 × 16
   name           height mass  hair_color skin_color eye_color birth_year gender
   <chr>          <chr>  <chr> <chr>      <chr>      <chr>     <chr>      <chr>
 1 Luke Skywalker 172    77    blond      fair       blue      19BBY      male
 2 C-3PO          167    75    n/a        gold       yellow    112BBY     n/a
 3 R2-D2          96     32    n/a        white, bl… red       33BBY      n/a
 4 Darth Vader    202    136   none       white      yellow    41.9BBY    male
 5 Leia Organa    150    49    brown      light      brown     19BBY      female
 6 Owen Lars      178    120   brown, gr… light      blue      52BBY      male
 7 Beru Whitesun… 165    75    brown      light      blue      47BBY      female
 8 R5-D4          97     32    n/a        white, red red       unknown    n/a
 9 Biggs Darklig… 183    84    black      light      brown     24BBY      male
10 Obi-Wan Kenobi 182    77    auburn, w… fair       blue-gray 57BBY      male
# ℹ 77 more rows
# ℹ 8 more variables: homeworld <chr>, films <list>, species <chr>,
#   vehicles <list>, starships <list>, created <chr>, edited <chr>, url <chr>
```

Unnesting - column types

```
sw_df |>
  unnest_wider(people) |>
  pull(height)
```

```
 [1] "172"     "167"     "96"      "202"     "150"     "178"     "165"
 [8] "97"      "183"     "182"     "188"     "180"     "228"     "180"
[15] "173"     "175"     "170"     "180"     "66"      "170"     "183"
[22] "200"     "190"     "177"     "175"     "180"     "150"     "unknown"
[29] "88"      "160"     "193"     "191"     "170"     "196"     "224"
[36] "206"     "183"     "137"     "112"     "183"     "163"     "175"
[43] "180"     "178"     "94"      "122"     "163"     "188"     "198"
[50] "196"     "171"     "184"     "188"     "264"     "188"     "196"
[57] "185"     "157"     "183"     "183"     "170"     "166"     "165"
[64] "193"     "191"     "183"     "168"     "198"     "229"     "213"
[71] "167"     "79"      "96"      "193"     "191"     "178"     "216"
[78] "234"     "188"     "178"     "206"     "unknown" "unknown" "unknown"
[85] "unknown" "unknown" "165"
```

More list columns

```
1 sw_df |>
2   unnest_wider(people) |>
3   select(name, starships)
```

```
# A tibble: 87 × 2
   name               starships
   <chr>              <list>
 1 Luke Skywalker     <chr [2]>
 2 C-3PO              <NULL>
 3 R2-D2              <NULL>
 4 Darth Vader        <chr [1]>
 5 Leia Organa        <NULL>
 6 Owen Lars          <NULL>
 7 Beru Whitesun lars <NULL>
 8 R5-D4              <NULL>
 9 Biggs Darklighter  <chr [1]>
10 Obi-Wan Kenobi     <chr [5]>
# ℹ 77 more rows
```

```
1 sw_df |>
2   unnest_wider(people) |>
3   select(name, starships) |>
4   pull(starships) |>
5   str()
```

```
List of 87
 $ : chr [1:2]
"http://swapi.co/api/starships/12/"
"http://swapi.co/api/starships/22/"
 $ : NULL
 $ : NULL
 $ : chr
"http://swapi.co/api/starships/13/"
 $ : NULL
 $ : NULL
 $ : NULL
 $ : NULL
 $ : chr
"http://swapi.co/api/starships/12/"
```

Unnest Longer

```
unnest_wider(sw_df, people) |>
  select(name, starships) |>
  unnest_longer(starships)
```

```
# A tibble: 31 × 2
   name              starships
   <chr>             <chr>
 1 Luke Skywalker    http://swapi.co/api/starships/12/
 2 Luke Skywalker    http://swapi.co/api/starships/22/
 3 Darth Vader       http://swapi.co/api/starships/13/
 4 Biggs Darklighter http://swapi.co/api/starships/12/
 5 Obi-Wan Kenobi    http://swapi.co/api/starships/48/
 6 Obi-Wan Kenobi    http://swapi.co/api/starships/59/
 7 Obi-Wan Kenobi    http://swapi.co/api/starships/64/
 8 Obi-Wan Kenobi    http://swapi.co/api/starships/65/
 9 Obi-Wan Kenobi    http://swapi.co/api/starships/74/
10 Anakin Skywalker  http://swapi.co/api/starships/59/
# ℹ 21 more rows
```

Aside - sw_starships

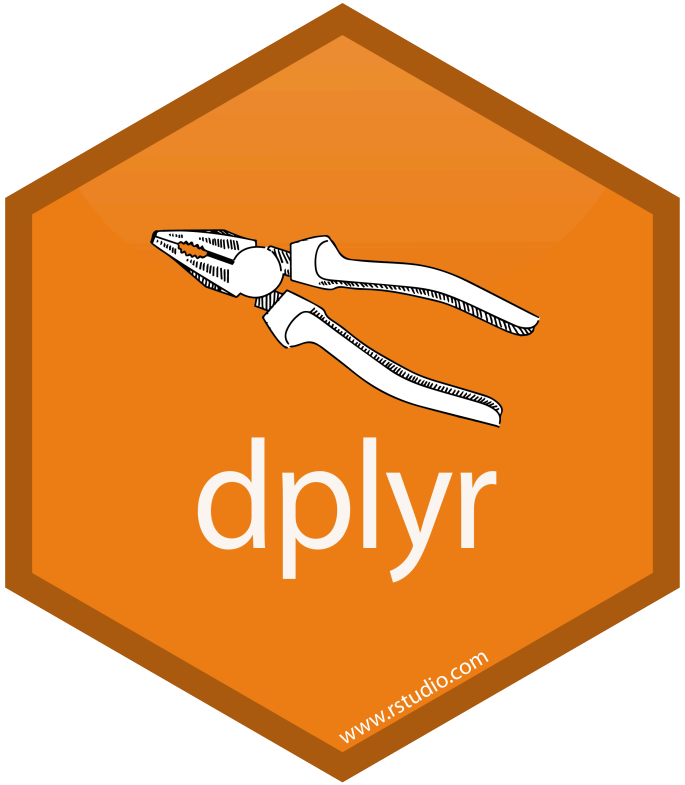
```
(ships = tibble(ships = repurrrsive::sw_starships) |>
   unnest_wider(ships) |>
   select(ship = name, url)
)
```

```
# A tibble: 37 × 2
   ship                          url
   <chr>                         <chr>
 1 Sentinel-class landing craft  http://swapi.co/api/starships/5/
 2 Death Star                    http://swapi.co/api/starships/9/
 3 Millennium Falcon             http://swapi.co/api/starships/10/
 4 Y-wing                        http://swapi.co/api/starships/11/
 5 X-wing                        http://swapi.co/api/starships/12/
 6 TIE Advanced x1               http://swapi.co/api/starships/13/
 7 Executor                      http://swapi.co/api/starships/15/
 8 Slave 1                       http://swapi.co/api/starships/21/
 9 Imperial shuttle              http://swapi.co/api/starships/22/
10 EF76 Nebulon-B escort frigate http://swapi.co/api/starships/23/
# ℹ 27 more rows
```

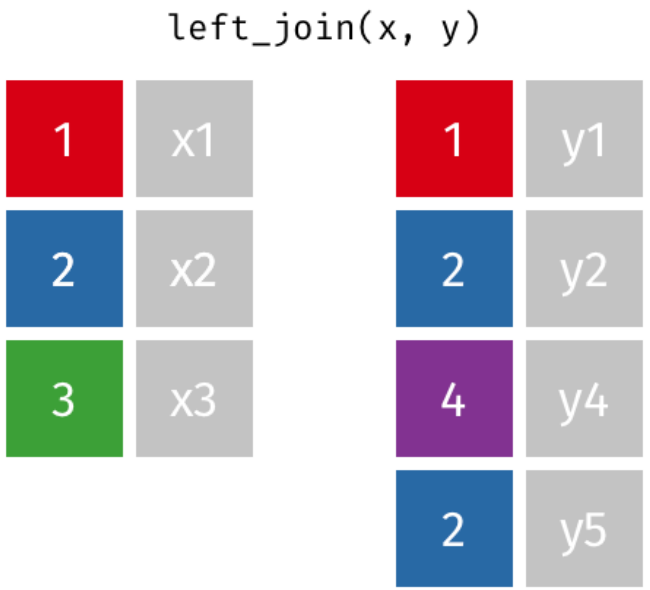
General advice

- If there is a consistent set of entries (usually named) in the list column, use `unnest_wider()`
- If there is an inconsistent set of entries (usually unnamed) in the list column, use `unnest_longer()`
- Never use just `unnest()` - it can be inconsistent depending on input data
- Think about if you need all the data or not - `unnest_*()` are not always the best choice

Aside - Joins

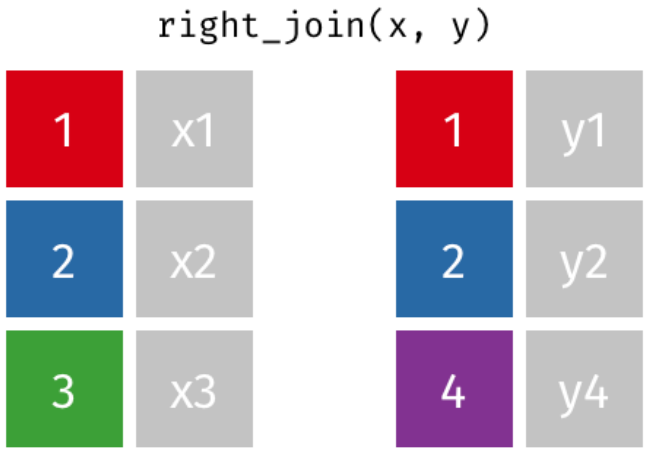


Joins (left)



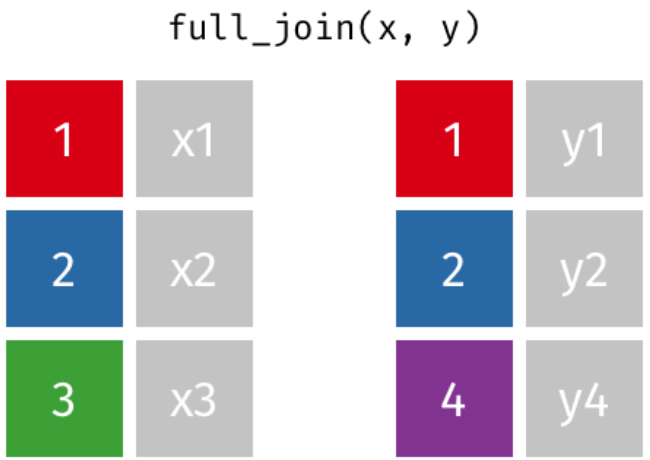
From gadenbuie/tidyexplain

Joins (right)



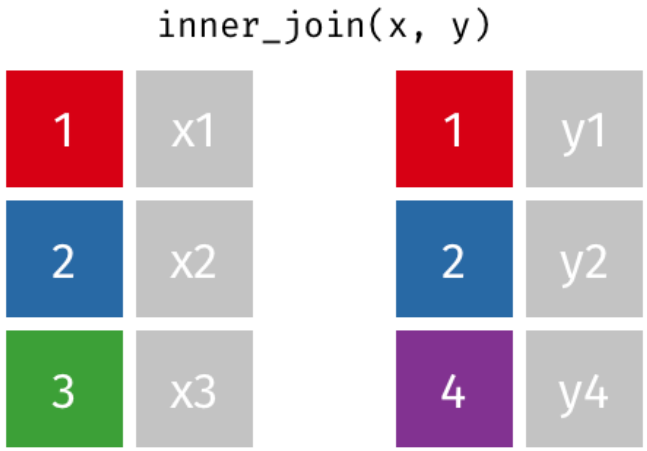
From gadenbuie/tidyexplain

Joins (full / outer)



From gadenbuie/tidyexplain

Joins (inner)



From gadenbuie/tidyexplain

join by

By default dplyr's join functions will join based on matching column names between the two data frames.

To specify the columns to join by (or to handle non-matching names) pass in a character vector of column names (or a named character vector where the names match the left data frame and the values match the right).

More recently more advanced joins have been allowed via the `join_by()` construct which allows for: equality, inequality, rolling, overlap, and cross joins. See `?join_by` for details.

Joining people and starships

```
sw_df |>
  unnest_wider(people) |>
  select(name, starships) |>
  unnest_longer(starships) |>
  left_join(ships, by = c("starships" = "url"))
```

```
# A tibble: 31 × 3
   name              starships                       ship
   <chr>             <chr>                           <chr>
 1 Luke Skywalker    http://swapi.co/api/starships/12/ X-wing
 2 Luke Skywalker    http://swapi.co/api/starships/22/ Imperial shuttle
 3 Darth Vader       http://swapi.co/api/starships/13/ TIE Advanced x1
 4 Biggs Darklighter http://swapi.co/api/starships/12/ X-wing
 5 Obi-Wan Kenobi    http://swapi.co/api/starships/48/ Jedi starfighter
 6 Obi-Wan Kenobi    http://swapi.co/api/starships/59/ Trade Federation cruiser
 7 Obi-Wan Kenobi    http://swapi.co/api/starships/64/ Naboo star skiff
 8 Obi-Wan Kenobi    http://swapi.co/api/starships/65/ Jedi Interceptor
 9 Obi-Wan Kenobi    http://swapi.co/api/starships/74/ Belbullab-22 starfighter
10 Anakin Skywalker  http://swapi.co/api/starships/59/ Trade Federation cruiser
# ℹ 21 more rows
```

Putting it together

```
sw_df |>
  unnest_wider(people) |>
  select(name, starships) |>
  unnest_longer(starships) |>
  inner_join(ships, by = c("starships" = "url")) |>
  select(-starships) |>
  group_by(name) |>
  summarize(ships = list(ship), .groups = "drop")
```

```
# A tibble: 20 × 2
   name              ships
   <chr>             <list>
 1 Anakin Skywalker  <chr [3]>
 2 Arvel Crynyd      <chr [1]>
 3 Biggs Darklighter <chr [1]>
 4 Boba Fett         <chr [1]>
 5 Chewbacca         <chr [2]>
 6 Darth Maul        <chr [1]>
 7 Darth Vader       <chr [1]>
 8 Gregar Typho      <chr [1]>
 9 Grievous          <chr [1]>
10 Han Solo          <chr [2]>
11 Jek Tono Porkins  <chr [1]>
12 Lando Calrissian  <chr [1]>
13 Luke Skywalker    <chr [2]>
14 Nien Nunb         <chr [1]>
```

```r
sw_df |>
  unnest_wider(people) |>
  select(name, starships) |>
  unnest_longer(starships) |>
  inner_join(ships, by = c("starships" = "url")) |>
  select(-starships) |>
  group_by(name) |>
  summarize(ships = paste(ship, collapse = ", "), .groups = "drop")
```

```
# A tibble: 20 × 2
   name              ships
   <chr>             <chr>
 1 Anakin Skywalker  Trade Federation cruiser, Jedi Interceptor, Naboo fighter
 2 Arvel Crynyd      A-wing
 3 Biggs Darklighter X-wing
 4 Boba Fett         Slave 1
 5 Chewbacca         Millennium Falcon, Imperial shuttle
 6 Darth Maul        Scimitar
 7 Darth Vader       TIE Advanced x1
 8 Gregar Typho      Naboo fighter
 9 Grievous          Belbullab-22 starfighter
10 Han Solo          Millennium Falcon, Imperial shuttle
11 Jek Tono Porkins  X-wing
12 Lando Calrissian  Millennium Falcon
13 Luke Skywalker    X-wing, Imperial shuttle
14 Nien Nunb         Millennium Falcon
```

Exercise 2

1. Which planet appeared in the most starwars film (according to the data in `sw_planets`)?
2. Which planet was the homeworld of the most characters in the starwars films?

05:00