

Lec 07 - `tidyverse`

Statistical Programming

Fall 2021

Dr. Colin Rundel



Reshaping data (Wide vs. Long)

Wide -> Long

The diagram illustrates the transformation of a wide data frame into a long data frame. On the left, a wide data frame is shown with columns for country (A, B, C) and years (1999, 2000). The values are 0.7K, 2K for country A; 37K, 80K for country B; and 212K, 213K for country C. An arrow points to the right, indicating the transformation. On the right, a long data frame is shown with columns for country, year, and cases. The data is now row-oriented, with each row representing a specific country and year combination, and the values are 0.7K, 37K, 212K for year 1999 and 2K, 80K, 213K for year 2000.

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)

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
```

Long -> Wide

The diagram illustrates the transformation of a long-format data frame into a wide-format data frame. On the left, a long-format data frame is shown with columns: country, year, type, and count. The data consists of 12 rows for three countries (A, B, C) across two years (1999, 2000), with each country having two entries per year, one for 'cases' and one for 'pop'. An arrow points from this long-format frame to a wide-format data frame on the right. In the wide-format frame, the columns are country, year, cases, and pop. The data is now organized by country and year, with all 'cases' values grouped under the 'cases' column and all 'pop' values grouped under the 'pop' column.

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)

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"  
)
```

```
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
```

Separate

The diagram illustrates the process of separating a single column into two distinct columns. On the left, there is a table with three columns: 'country', 'year', and 'rate'. The 'rate' column contains values like '0.7K/19M', '2K/20M', etc., separated by a slash. An arrow points from this table to the right, indicating the transformation. On the right, the same data is shown in a new table with four columns: 'country', 'year', 'cases', and 'pop'. The 'cases' column now contains the first part of the rate (e.g., '0.7K'), and the 'pop' column contains the second part (e.g., '19M').

country	year	rate
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

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

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

Unite

The diagram illustrates the 'unite' function's behavior. On the left, a wide data frame is shown with columns 'country', 'century', and 'year'. It contains six rows of data for 'Afghan', 'Brazil', and 'China'. The 'century' and 'year' columns are highlighted in yellow. An arrow points from this frame to the right, indicating the transformation. On the right, a long data frame is shown with columns 'country' and 'year'. It contains twelve rows of data, where each country appears twice: once with the century value ('19') and once with the year value ('99' or '0'). The 'year' column is highlighted in orange.

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

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

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  
))
```

```
## # A tibble: 4 × 7  
##   name   hw_1   hw_2   hw_3   hw_4 proj_1 proj_2  
##   <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 Alice     19     19     18     20      89      95  
## 2 Bob       18     20     18     16      77      88  
## 3 Carol     18     20     18     17      96      99  
## 4 Dave      19     19     18     19      86      82
```

How would we calculate a final score based on the following formula,
 $\text{score} = 0.5 \sum_i \text{hw}_i + 0.5 \sum_j \text{proj}_j$

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  
## # ... with 14 more rows
```

What does this get us?

```
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  
## # ... with 14 more rows
```

Tidy approach?

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

```
## # 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"  
) %>%  
  group_by(name, type) %>%  
  summarize(  
    total = sum(score),  
    .groups = "drop"  
) %>%  
  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"  
) %>%  
  group_by(name, type) %>%  
  summarize(  
    total = sum(score),  
    .groups = "drop"  
) %>%  
  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
```

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.

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.

```
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/ap
##   ..$ species    : chr "http://swapi.co/api/species/1/"
##   ..$ vehicles   : chr [1:2] "http://swapi.co/api/vehicles/14/" "http://swapi.co/api/vehicles/30/"
##   ..$ starships  : chr [1:2] "http://swapi.co/api/starships/12/" "http://swapi.co/api/starships/22/"
##   ..$ created    : chr "2014-12-09T13:50:51.644000Z"
##   ..$ edited     : chr "2014-12-20T21:17:56.891000Z"
```

View(repurrrsive::sw_people)

The screenshot shows a variable named `repurrrsive::sw_people` expanded to show its properties and values. The properties listed are: name, height, mass, hair_color, skin_color, eye_color, birth_year, gender, homeworld, films, species, vehicles, starships, created, edited, url, [[1]], [[2]], [[3]], [[4]], and [[5]]. The values for these properties are: '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/a ...', '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/', List of length 14, List of length 14, List of length 15, and List of length 15.

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

Recent versions of `tidyverse` have added several functions that are designed to aid in the tidying of hierarchical data. Since they are part of `tidyverse` all of the following functions work with data frames.

From `tidyverse` - `hoist()`, `unnest_longer()`, and `unnest_wider()` provide tools for rectangling, collapsing deeply nested lists into regular columns.

Lists as columns

```
(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]>  
## # ... with 77 more rows
```

```
is.data.frame(sw_df)
```

```
## [1] TRUE
```

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

```
##  
## 1 Luke Skywalker, 172, 77, blond, fair, blue, 19BBY,  
## 2  
## 3  
## 4  
## 5  
## 6
```

Unnesting

```
sw_df %>%  
  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, grey 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 Darklight 183    84  black     light      brown     24BBY     male  
## 10 Obi-Wan Kenobi 182    77  auburn, whi... fair      blue-gray 57BBY     male  
## # ... with 77 more rows, and 8 more variables: homeworld <chr>, films <list>,  
## #   species <chr>, vehicles <list>, starships <list>, created <chr>,  
## #   edited <chr>, url <chr>
```

More list columns

```
sw_df %>%  
  unnest_wider(people) %>%  
  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]>  
## # ... with 77 more rows
```

```
sw_df %>%  
  unnest_wider(people) %>%  
  select(name, starships) %>%  
  pull(starships) %>%  
  str()
```

```
## List of 87  
## $ : chr [1:2] "http://swapi.co/api/starships/12/" "  
## $ : NULL  
## $ : NULL  
## $ : chr "http://swapi.co/api/starships/13/"  
## $ : NULL  
## $ : chr "http://swapi.co/api/starships/12/"  
## $ : chr [1:5] "http://swapi.co/api/starships/48/" "  
## $ : chr [1:3] "http://swapi.co/api/starships/59/" "  
## $ : NULL  
## $ : chr [1:2] "http://swapi.co/api/starships/10/" "  
## $ : chr [1:2] "http://swapi.co/api/starships/10/" "  
## $ : NULL  
## $ : NULL
```

Unnest Longer

```
unnest_wider(sw_df, people) %>%  
  select(name, starships) %>%  
  unnest_longer(starships)  
  
## # A tibble: 98 × 2  
##   name          starships  
##   <chr>        <chr>  
## 1 Luke Skywalker http://swapi.co/api/starships/12/  
## 2 Luke Skywalker http://swapi.co/api/starships/22/  
## 3 C-3PO          <NA>  
## 4 R2-D2          <NA>  
## 5 Darth Vader   http://swapi.co/api/starships/13/  
## 6 Leia Organa    <NA>  
## 7 Owen Lars     <NA>  
## 8 Beru Whitesun lars <NA>  
## 9 R5-D4          <NA>  
## 10 Biggs Darklighter http://swapi.co/api/starships/12/  
## # ... with 88 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/  
## # ... with 27 more rows
```

Joins (left)

`left_join(x, y)`

1	x1	1	y1
2	x2	2	y2
3	x3	4	y4
		2	y5

Joins (right)

`right_join(x, y)`

1	x1	1	y1
2	x2	2	y2
3	x3	4	y4

Joins (full / outer)

`full_join(x, y)`

1	x1	1	y1
2	x2	2	y2
3	x3	4	y4

Joins (inner)

inner_join(x, y)

1	x1	1	y1
2	x2	2	y2
3	x3	4	y4

Joining people and starships

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

```
## # A tibble: 98 × 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 C-3PO           <NA>             <NA>  
## 4 R2-D2           <NA>             <NA>  
## 5 Darth Vader   http://swapi.co/api/starships/13/ TIE Advanced x1  
## 6 Leia Organa    <NA>             <NA>  
## 7 Owen Lars     <NA>             <NA>  
## 8 Beru Whitesun lars <NA>             <NA>  
## 9 R5-D4           <NA>             <NA>  
## 10 Biggs Darklighter http://swapi.co/api/starships/12/ X-wing  
## # ... with 88 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]>
```

```
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  
## 15 Obi-Wan Kenobi Jedi starfighter, Trade Federation cruiser, Naboo star ski...
```

Exercise 2

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