Reshaping data from long to wide format, or wide to long format, is a common task in data science. Until recently, the best functions for performing this task in R were the `gather` and `spread` functions from the `tidyr` package. However, these functions had limitations, such as only being able to reshape one variable at a time, that required creative workarounds. The newest version of `tidyr` introduces the `pivot_longer()` and `pivot_wider()` functions that perform the same tasks, but that also handle a wider variety of use cases. Additionally, the function and argument names have been changed to be more intuitive. The purpose of this blog post is to help make the transition from `gather()` and `spread()` to the new pivoting functions.

It is commonly said that data scientists spend 80% of their time data cleaning and only 20% actually analyzing the data. Every dataset is messy in its own way, and it can take a while to get the data into a format that your analysis tools can work with. The package `tidyr` provides tools to help you get your input data into a standardized tidy dataframe.

Some of the tasks that `tidyr` can help with include:

- **pivoting**: changing the representation of a rectangular dataset (e.g. reshaping from long to wide format)
- **rectangling**: turning nested lists into tibbles
- **nesting**: dataframe where a column is a list of data-frames
- **separating/combining columns**: splitting a single character vector into multiple, or combining multiple into one
- **missing values**: tools for handling missing values and converting between implicit and explicit missing values

In this post we will focus on pivoting. In particular, `tidyr`'s change in syntax from the `gather()` and `spread()` functions to `pivot_longer()` and `pivot_wider()`.

`tidyr` syntax changes

The most popular functions from `tidyr` are those used to **pivot** a rectangular dataset to a longer or wider format, `gather()` and `spread()`. However, with the release of `tidyr` version 1.0.0 (09/11/19), `pivot_longer()` and `pivot_wider()` have been released to replace them.

A high-level comparison of the old and new syntax:

- Pivot to a wider format
  - `spread(data, key, value)`
    - `key` – Values of the `key` column will become column names
    - `value` – Cell values will be taken from the `value` column
  - `pivot_wider(data, names_from, values_from)`
    - `names_from` – Values of the `names_from` column will become column names
    - `values_from` – Cell values will be taken from the `values_from` column
- Pivot to a longer format
  - `gather(data, key, value, ...)`
    - `key` – Name of column to be created which contains the column names of gathered columns as values
    - `value` – Name of column to be created with the data stored in cell values of gathered columns
    - `...` – Columns to pivot to longer format
  - `pivot_longer(data, cols, names_to, values_to)`
    - `cols` – Columns to pivot to longer format
    - `names_to` – Name of column to be created which contains the column names of gathered columns as values
    - `values_to` – Name of column to be created with the data stored in cell values of

gathered columns

The usage of the functions remains the same, but the function and argument names have been changed to be more intuitive.

Example Data

As an example, we will look at how to use `tidyr` to change between three representations of the `gapminder::gapminder` dataset.

We first load in the packages that we'll use and create two additional representations of the data. Don't worry about understanding this code for now.

```
library(tidyverse)
library(gapminder)

gapminder_long <- gapminder %>%
  pivot_longer(
    lifeExp:gdpPercap,
    names_to = "measure",
    values_to = "value"
  )

gapminder_wide <- gapminder %>%
  pivot_wider(
    names_from = year,
    values_from = c(lifeExp, pop, gdpPercap)
  ) %>%
  select(
    country, continent, ends_with("52"), ends_with("57"),
    ends_with("62"), ends_with("67"), ends_with("72"),
    ends_with("77"), ends_with("82"), ends_with("87"),
    ends_with("92"), ends_with("97"), ends_with("02"),
    ends_with("07")
  )
```

We now have three representations of the same dataset, `gapminder`, `gapminder_long`, and `gapminder_wide`.

```
gapminder
```

```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp      pop gdpPercap
##
##  1 Afghanistan Asia       1952    28.8  8425333      779.
##  2 Afghanistan Asia       1957    30.3  9240934      821.
##  3 Afghanistan Asia       1962    32.0 10267083      853.
##  4 Afghanistan Asia       1967    34.0 11537966      836.
##  5 Afghanistan Asia       1972    36.1 13079460      740.
##  6 Afghanistan Asia       1977    38.4 14880372      786.
##  7 Afghanistan Asia       1982    39.9 12881816      978.
##  8 Afghanistan Asia       1987    40.8 13867957      852.
##  9 Afghanistan Asia       1992    41.7 16317921      649.
## 10 Afghanistan Asia       1997    41.8 22227415      635.
## # ... with 1,694 more rows
```

`gapminder` has one row for each pair of `country` and `year`, and one column for each measure (`lifeExp`, `pop`, `gdpPercap`).

```
gapminder_long
```

```
## # A tibble: 5,112 x 5
##    country     continent  year measure        value
##
##  1 Afghanistan Asia       1952 lifeExp         28.8
##  2 Afghanistan Asia       1952 pop        8425333
##  3 Afghanistan Asia       1952 gdpPercap      779.
##  4 Afghanistan Asia       1957 lifeExp         30.3
##  5 Afghanistan Asia       1957 pop        9240934
##  6 Afghanistan Asia       1957 gdpPercap      821.
##  7 Afghanistan Asia       1962 lifeExp         32.0
##  8 Afghanistan Asia       1962 pop       10267083
##  9 Afghanistan Asia       1962 gdpPercap      853.
## 10 Afghanistan Asia       1967 lifeExp         34.0
## # ... with 5,102 more rows
```

We can notice that the three measure columns from before have been combined into two columns: `measure` and `value`. Also, the data now has three rows for each pair of `country` and `year`. This is considered to be in a **longer** format, because columns were collapsed and the information is stored as additional rows.

```
gapminder_wide
```

```
## # A tibble: 142 x 38
##    country continent lifeExp_1952 pop_1952 gdpPercap_1952 lifeExp_1957
##
##  1 Afghan~ Asia              28.8  8425333           779.         30.3
##  2 Albania Europe            55.2  1282697          1601.         59.3
##  3 Algeria Africa            43.1  9279525          2449.         45.7
##  4 Angola  Africa            30.0  4232095          3521.         32.0
##  5 Argent~ Americas          62.5 17876956          5911.         64.4
##  6 Austra~ Oceania           69.1  8691212         10040.         70.3
##  7 Austria Europe            66.8  6927772          6137.         67.5
##  8 Bahrain Asia              50.9   120447          9867.         53.8
##  9 Bangla~ Asia              37.5 46886859           684.         39.3
## 10 Belgium Europe            68    8730405          8343.         69.2
## # ... with 132 more rows, and 32 more variables: pop_1957 ,
## #   gdpPercap_1957 , lifeExp_1962 , pop_1962 ,
## #   gdpPercap_1962 , lifeExp_1967 , pop_1967 ,
## #   gdpPercap_1967 , lifeExp_1972 , pop_1972 ,
## #   gdpPercap_1972 , lifeExp_1977 , pop_1977 ,
## #   gdpPercap_1977 , lifeExp_1982 , pop_1982 ,
## #   gdpPercap_1982 , lifeExp_1987 , pop_1987 ,
## #   gdpPercap_1987 , lifeExp_1992 , pop_1992 ,
## #   gdpPercap_1992 , lifeExp_1997 , pop_1997 ,
## #   gdpPercap_1997 , lifeExp_2002 , pop_2002 ,
## #   gdpPercap_2002 , lifeExp_2007 , pop_2007 ,
## #   gdpPercap_2007
```

In `gapminder_wide` the year variable has been spread into multiple columns. There is now only one row per `country`, but a column for each pair of `measure` and `year`. This is considered to be a **wider** representation, because information that was being stored as rows are now additional columns.

`pivot_wider()` example

Suppose we start with `gapminder_long`, but we need the data to be formatted like `gapminder`.

```
gapminder_long
```

```
## # A tibble: 5,112 x 5
##    country     continent  year measure        value
```

```
##
##   1 Afghanistan Asia       1952 lifeExp         28.8
##   2 Afghanistan Asia       1952 pop        8425333
##   3 Afghanistan Asia       1952 gdpPercap      779.
##   4 Afghanistan Asia       1957 lifeExp         30.3
##   5 Afghanistan Asia       1957 pop        9240934
##   6 Afghanistan Asia       1957 gdpPercap      821.
##   7 Afghanistan Asia       1962 lifeExp         32.0
##   8 Afghanistan Asia       1962 pop       10267083
##   9 Afghanistan Asia       1962 gdpPercap      853.
##  10 Afghanistan Asia       1967 lifeExp         34.0
## # ... with 5,102 more rows
```

```
gapminder
```

```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp      pop gdpPercap
##
##   1 Afghanistan Asia       1952    28.8  8425333      779.
##   2 Afghanistan Asia       1957    30.3  9240934      821.
##   3 Afghanistan Asia       1962    32.0 10267083      853.
##   4 Afghanistan Asia       1967    34.0 11537966      836.
##   5 Afghanistan Asia       1972    36.1 13079460      740.
##   6 Afghanistan Asia       1977    38.4 14880372      786.
##   7 Afghanistan Asia       1982    39.9 12881816      978.
##   8 Afghanistan Asia       1987    40.8 13867957      852.
##   9 Afghanistan Asia       1992    41.7 16317921      649.
##  10 Afghanistan Asia       1997    41.8 22227415      635.
## # ... with 1,694 more rows
```

We'd like there to be columns for `lifeExp`, `pop`, and `gdpPercap`.

We need to:

- **pivot** the dataset to a **wider** format (`pivot_wider()`)
- **names** of the new columns come **from** the `measure` column (`names_from = measure`)
- **values** for the new columns come **from** the `value` column (`values_from = value`)

```
gapminder_long %>%
  pivot_wider(
    names_from = measure,
    values_from = value
  )
```

```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp      pop gdpPercap
##
##   1 Afghanistan Asia       1952    28.8  8425333      779.
##   2 Afghanistan Asia       1957    30.3  9240934      821.
##   3 Afghanistan Asia       1962    32.0 10267083      853.
##   4 Afghanistan Asia       1967    34.0 11537966      836.
##   5 Afghanistan Asia       1972    36.1 13079460      740.
##   6 Afghanistan Asia       1977    38.4 14880372      786.
##   7 Afghanistan Asia       1982    39.9 12881816      978.
##   8 Afghanistan Asia       1987    40.8 13867957      852.
##   9 Afghanistan Asia       1992    41.7 16317921      649.
##  10 Afghanistan Asia       1997    41.8 22227415      635.
## # ... with 1,694 more rows
```

With `spread()`, the syntax is the same, but the arguments are named `key` and `value`.

```
gapminder_long %>%
  spread(
    key = measure,
    value = value
  )
```

```
## # A tibble: 1,704 x 6
##    country     continent  year gdpPercap lifeExp      pop
##
##  1 Afghanistan Asia       1952      779.    28.8  8425333
##  2 Afghanistan Asia       1957      821.    30.3  9240934
##  3 Afghanistan Asia       1962      853.    32.0 10267083
##  4 Afghanistan Asia       1967      836.    34.0 11537966
##  5 Afghanistan Asia       1972      740.    36.1 13079460
##  6 Afghanistan Asia       1977      786.    38.4 14880372
##  7 Afghanistan Asia       1982      978.    39.9 12881816
##  8 Afghanistan Asia       1987      852.    40.8 13867957
##  9 Afghanistan Asia       1992      649.    41.7 16317921
## 10 Afghanistan Asia       1997      635.    41.8 22227415
## # ... with 1,694 more rows
```

`pivot_longer()` example

For this example, we will format `gapminder_wide` so that there is a row for every country and year pair.

```
gapminder_wide
```

```
## # A tibble: 142 x 38
##    country continent lifeExp_1952 pop_1952 gdpPercap_1952 lifeExp_1957
##
##  1 Afghan~ Asia              28.8  8425333           779.          30.3
##  2 Albania Europe            55.2  1282697          1601.          59.3
##  3 Algeria Africa            43.1  9279525          2449.          45.7
##  4 Angola  Africa            30.0  4232095          3521.          32.0
##  5 Argent~ Americas          62.5 17876956          5911.          64.4
##  6 Austra~ Oceania           69.1  8691212         10040.          70.3
##  7 Austria Europe            66.8  6927772          6137.          67.5
##  8 Bahrain Asia              50.9   120447          9867.          53.8
##  9 Bangla~ Asia              37.5 46886859           684.          39.3
## 10 Belgium Europe            68    8730405          8343.          69.2
## # ... with 132 more rows, and 32 more variables: pop_1957 ,
## #   gdpPercap_1957 , lifeExp_1962 , pop_1962 ,
## #   gdpPercap_1962 , lifeExp_1967 , pop_1967 ,
## #   gdpPercap_1967 , lifeExp_1972 , pop_1972 ,
## #   gdpPercap_1972 , lifeExp_1977 , pop_1977 ,
## #   gdpPercap_1977 , lifeExp_1982 , pop_1982 ,
## #   gdpPercap_1982 , lifeExp_1987 , pop_1987 ,
## #   gdpPercap_1987 , lifeExp_1992 , pop_1992 ,
## #   gdpPercap_1992 , lifeExp_1997 , pop_1997 ,
## #   gdpPercap_1997 , lifeExp_2002 , pop_2002 ,
## #   gdpPercap_2002 , lifeExp_2007 , pop_2007 ,
## #   gdpPercap_2007
```

We want columns 3-38 to become `year`, `lifeExp`, `pop`, and `gdpPercap`.

We'll come back to this problem, but for now let's look at a simplified version with only the `gdpPercap` columns.

```
gapminder_wide_gdp <- gapminder_wide %>%
  select(country, continent, starts_with("gdp"))

gapminder_wide_gdp
```

```
## # A tibble: 142 x 14
##    country continent gdpPercap_1952 gdpPercap_1957 gdpPercap_1962
##
##  1 Afghan~ Asia                779.           821.           853.
##  2 Albania Europe             1601.          1942.          2313.
##  3 Algeria Africa             2449.          3014.          2551.
##  4 Angola  Africa             3521.          3828.          4269.
##  5 Argent~ Americas           5911.          6857.          7133.
##  6 Austra~ Oceania           10040.         10950.         12217.
##  7 Austria Europe             6137.          8843.         10751.
##  8 Bahrain Asia               9867.         11636.         12753.
##  9 Bangla~ Asia                684.           662.           686.
## 10 Belgium Europe             8343.          9715.         10991.
## # ... with 132 more rows, and 9 more variables: gdpPercap_1967 ,
## #   gdpPercap_1972 , gdpPercap_1977 , gdpPercap_1982 ,
## #   gdpPercap_1987 , gdpPercap_1992 , gdpPercap_1997 ,
## #   gdpPercap_2002 , gdpPercap_2007
```

We want columns 3-14 to become two columns: `year` and `gdpPercap`. To do this we **pivot** the data to a **longer** format (`pivot_longer()`).

```
gapminder_wide_gdp %>%
  pivot_longer(
   gdpPercap_1952:gdpPercap_2007
  )
```

```
## # A tibble: 1,704 x 4
##    country     continent name           value
##
##  1 Afghanistan Asia      gdpPercap_1952  779.
##  2 Afghanistan Asia      gdpPercap_1957  821.
##  3 Afghanistan Asia      gdpPercap_1962  853.
##  4 Afghanistan Asia      gdpPercap_1967  836.
##  5 Afghanistan Asia      gdpPercap_1972  740.
##  6 Afghanistan Asia      gdpPercap_1977  786.
##  7 Afghanistan Asia      gdpPercap_1982  978.
##  8 Afghanistan Asia      gdpPercap_1987  852.
##  9 Afghanistan Asia      gdpPercap_1992  649.
## 10 Afghanistan Asia      gdpPercap_1997  635.
## # ... with 1,694 more rows
```

```
gapminder_wide_gdp %>%
  gather(
    gdpPercap_1952:gdpPercap_2007
  )
```

```
## Must supply a symbol or a string as argument
```

This intuitive syntax doesn’t work for `gather()`. We have to remember to first pass names of new columns to `key` and `value`.

```
gapminder_wide_gdp %>%
  gather(
    "key",
    "value",
    gdpPercap_1952:gdpPercap_2007
  )

## # A tibble: 1,704 x 4
##    country     continent key            value
##
##  1 Afghanistan Asia      gdpPercap_1952   779.
##  2 Albania     Europe    gdpPercap_1952  1601.
##  3 Algeria     Africa    gdpPercap_1952  2449.
##  4 Angola      Africa    gdpPercap_1952  3521.
##  5 Argentina   Americas  gdpPercap_1952  5911.
##  6 Australia   Oceania   gdpPercap_1952 10040.
##  7 Austria     Europe    gdpPercap_1952  6137.
##  8 Bahrain     Asia      gdpPercap_1952  9867.
##  9 Bangladesh  Asia      gdpPercap_1952   684.
## 10 Belgium     Europe    gdpPercap_1952  8343.
## # ... with 1,694 more rows
```

- Column **names** should go **to** a `year` variable (`names_to = year`)
- Cell **values** should go **to** a `gdpPercap` variable (`values_to = gdpPercap`)

```
gapminder_wide_gdp %>%
  pivot_longer(
    gdpPercap_1952:gdpPercap_2007,
    names_to = "year",
    values_to = "gdpPercap"
  )

## # A tibble: 1,704 x 4
##    country     continent year           gdpPercap
##
##  1 Afghanistan Asia      gdpPercap_1952      779.
##  2 Afghanistan Asia      gdpPercap_1957      821.
##  3 Afghanistan Asia      gdpPercap_1962      853.
##  4 Afghanistan Asia      gdpPercap_1967      836.
##  5 Afghanistan Asia      gdpPercap_1972      740.
##  6 Afghanistan Asia      gdpPercap_1977      786.
##  7 Afghanistan Asia      gdpPercap_1982      978.
##  8 Afghanistan Asia      gdpPercap_1987      852.
##  9 Afghanistan Asia      gdpPercap_1992      649.
## 10 Afghanistan Asia      gdpPercap_1997      635.
## # ... with 1,694 more rows
```

```
gapminder_wide_gdp %>%
  gather(
    key = "year",
    value = "gdpPercap",
    gdpPercap_1952:gdpPercap_2007
  )

## # A tibble: 1,704 x 4
##    country     continent year           gdpPercap
##
##  1 Afghanistan Asia      gdpPercap_1952      779.
##  2 Albania     Europe    gdpPercap_1952     1601.
##  3 Algeria     Africa    gdpPercap_1952     2449.
##  4 Angola      Africa    gdpPercap_1952     3521.
##  5 Argentina   Americas  gdpPercap_1952     5911.
##  6 Australia   Oceania   gdpPercap_1952    10040.
##   7 Austria     Europe    gdpPercap_1952     6137.
##   8 Bahrain     Asia      gdpPercap_1952     9867.
##   9 Bangladesh  Asia      gdpPercap_1952      684.
##  10 Belgium     Europe    gdpPercap_1952     8343.
## # ... with 1,694 more rows
```

The `year` column needs some cleaning, but this is the structure that we were looking for.

New Features

Aside from the minor syntax changes, the new pivoting functions have additional features that its predecessors do not.

- `pivot_wider()`:
  - `names_from` and `values_from` can be multiple columns rather than one
    - `names_sep`: when there are multiple `names_from` or `values_from` columns, `names_sep` will be used to join values together to form column names
  - `names_prefix`: append a string to the beginning of every variable name
- `pivot_longer()`:
  - `names_to` can be a character vector, creating multiple columns (requires `names_sep` or `names_pattern`)
    - `names_sep`: numeric vector (specifying positions to break on), or a single string (specifying a regular expression to split on) (`separate()`)
    - `names_pattern`: regular expression containing matching groups (specified by `()`) (`extract()`)
  - `names_prefix`: remove matching text from the beginning of every variable name
  - `names_ptypes` and `values_ptypes` allows you to specify the column types of the newly created name and value columns

`pivot_wider()` new features

`names_prefix`

```
gapminder_gdp <- gapminder %>%
  select(country, continent, year, gdpPercap)

gapminder_gdp
```

```
## # A tibble: 1,704 x 4
##    country     continent  year gdpPercap
##
##  1 Afghanistan Asia       1952      779.
##  2 Afghanistan Asia       1957      821.
##  3 Afghanistan Asia       1962      853.
##  4 Afghanistan Asia       1967      836.
##  5 Afghanistan Asia       1972      740.
##  6 Afghanistan Asia       1977      786.
##  7 Afghanistan Asia       1982      978.
##  8 Afghanistan Asia       1987      852.
##  9 Afghanistan Asia       1992      649.
## 10 Afghanistan Asia       1997      635.
## # ... with 1,694 more rows
```

Suppose we want this data in a wide format, with only one row per country. We can do this by pivoting such that there is a column for each year.

```
gapminder_gdp %>%
  pivot_wider(
    names_from = year,
    values_from = gdpPercap
  )
```

```
## # A tibble: 142 x 14
##    country continent `1952` `1957` `1962` `1967` `1972` `1977` `1982`
##
##  1 Afghan~ Asia        779.   821.   853.   836.   740.   786.   978.
##  2 Albania Europe     1601.  1942.  2313.  2760.  3313.  3533.  3631.
##  3 Algeria Africa     2449.  3014.  2551.  3247.  4183.  4910.  5745.
##  4 Angola  Africa     3521.  3828.  4269.  5523.  5473.  3009.  2757.
##  5 Argent~ Americas   5911.  6857.  7133.  8053.  9443. 10079.  8998.
##  6 Austra~ Oceania   10040. 10950. 12217. 14526. 16789. 18334. 19477.
##  7 Austria Europe     6137.  8843. 10751. 12835. 16662. 19749. 21597.
##  8 Bahrain Asia       9867. 11636. 12753. 14805. 18269. 19340. 19211.
##  9 Bangla~ Asia        684.   662.   686.   721.   630.   660.   677.
## 10 Belgium Europe     8343.  9715. 10991. 13149. 16672. 19118. 20980.
## # ... with 132 more rows, and 5 more variables: `1987` ,
## #   `1992` , `1997` , `2002` , `2007`
```

These column names are not syntactically valid, because names are not supposed to start with a number.

`names_prefix` allows us to easily add a string to the start of each created name.

```
gapminder_gdp %>%
  pivot_wider(
    names_from = year,
    names_prefix = "year_",
    values_from = gdpPercap
  )
```

```
## # A tibble: 142 x 14
##    country continent year_1952 year_1957 year_1962 year_1967 year_1972
##
##  1 Afghan~ Asia           779.      821.      853.      836.      740.
##  2 Albania Europe        1601.     1942.     2313.     2760.     3313.
##  3 Algeria Africa        2449.     3014.     2551.     3247.     4183.
##  4 Angola  Africa        3521.     3828.     4269.     5523.     5473.
##  5 Argent~ Americas      5911.     6857.     7133.     8053.     9443.
##  6 Austra~ Oceania      10040.    10950.    12217.    14526.    16789.
##  7 Austria Europe        6137.     8843.    10751.    12835.    16662.
##  8 Bahrain Asia          9867.    11636.    12753.    14805.    18269.
##  9 Bangla~ Asia           684.      662.      686.      721.      630.
## 10 Belgium Europe        8343.     9715.    10991.    13149.    16672.
## # ... with 132 more rows, and 7 more variables: year_1977 ,
## #   year_1982 , year_1987 , year_1992 , year_1997 ,
## #   year_2002 , year_2007
```

Multiple `values_from` columns

Suppose that we have `gapminder` and we need there to be one row per country like `gapminder_wide`.

```
gapminder
```

```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp      pop gdpPercap
##
##  1 Afghanistan Asia       1952    28.8  8425333      779.
##  2 Afghanistan Asia       1957    30.3  9240934      821.
##  3 Afghanistan Asia       1962    32.0 10267083      853.
##  4 Afghanistan Asia       1967    34.0 11537966      836.
##  5 Afghanistan Asia       1972    36.1 13079460      740.
##  6 Afghanistan Asia       1977    38.4 14880372      786.
##  7 Afghanistan Asia       1982    39.9 12881816      978.
##  8 Afghanistan Asia       1987    40.8 13867957      852.
##  9 Afghanistan Asia       1992    41.7 16317921      649.
## 10 Afghanistan Asia       1997    41.8 22227415      635.
## # ... with 1,694 more rows
```

```
gapminder_wide
```

```
## # A tibble: 142 x 38
##    country continent lifeExp_1952 pop_1952 gdpPercap_1952 lifeExp_1957
##
##  1 Afghan~ Asia              28.8  8425333           779.         30.3
##  2 Albania Europe            55.2  1282697          1601.         59.3
##  3 Algeria Africa            43.1  9279525          2449.         45.7
##  4 Angola  Africa            30.0  4232095          3521.         32.0
##  5 Argent~ Americas          62.5 17876956          5911.         64.4
##  6 Austra~ Oceania           69.1  8691212         10040.         70.3
##  7 Austria Europe            66.8  6927772          6137.         67.5
##  8 Bahrain Asia              50.9   120447          9867.         53.8
##  9 Bangla~ Asia              37.5 46886859           684.         39.3
## 10 Belgium Europe            68    8730405          8343.         69.2
## # ... with 132 more rows, and 32 more variables: pop_1957 ,
## #   gdpPercap_1957 , lifeExp_1962 , pop_1962 ,
## #   gdpPercap_1962 , lifeExp_1967 , pop_1967 ,
## #   gdpPercap_1967 , lifeExp_1972 , pop_1972 ,
## #   gdpPercap_1972 , lifeExp_1977 , pop_1977 ,
## #   gdpPercap_1977 , lifeExp_1982 , pop_1982 ,
## #   gdpPercap_1982 , lifeExp_1987 , pop_1987 ,
## #   gdpPercap_1987 , lifeExp_1992 , pop_1992 ,
## #   gdpPercap_1992 , lifeExp_1997 , pop_1997 ,
## #   gdpPercap_1997 , lifeExp_2002 , pop_2002 ,
## #   gdpPercap_2002 , lifeExp_2007 , pop_2007 ,
## #   gdpPercap_2007
```

With `spread()` it isn't possible to pivot multiple value columns based on a single key. The hack was to first use `gather()` and `unite()` to create a single value column to spread.

```
gapminder
```

```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp      pop gdpPercap
##
##  1 Afghanistan Asia       1952    28.8  8425333      779.
##  2 Afghanistan Asia       1957    30.3  9240934      821.
##  3 Afghanistan Asia       1962    32.0 10267083      853.
##  4 Afghanistan Asia       1967    34.0 11537966      836.
##  5 Afghanistan Asia       1972    36.1 13079460      740.
##  6 Afghanistan Asia       1977    38.4 14880372      786.
##  7 Afghanistan Asia       1982    39.9 12881816      978.
##  8 Afghanistan Asia       1987    40.8 13867957      852.
##  9 Afghanistan Asia       1992    41.7 16317921      649.
## 10 Afghanistan Asia       1997    41.8 22227415      635.
## # ... with 1,694 more rows
```

```
gapminder %>%
  gather(
    key = "key",
    value = "value",
    lifeExp:gdpPercap
  )
```

```
## # A tibble: 5,112 x 5
##    country     continent  year key     value
##
##  1 Afghanistan Asia       1952 lifeExp  28.8
##  2 Afghanistan Asia       1957 lifeExp  30.3
##  3 Afghanistan Asia       1962 lifeExp  32.0
##  4 Afghanistan Asia       1967 lifeExp  34.0
##  5 Afghanistan Asia       1972 lifeExp  36.1
##  6 Afghanistan Asia       1977 lifeExp  38.4
##  7 Afghanistan Asia       1982 lifeExp  39.9
##  8 Afghanistan Asia       1987 lifeExp  40.8
##  9 Afghanistan Asia       1992 lifeExp  41.7
## 10 Afghanistan Asia       1997 lifeExp  41.8
## # ... with 5,102 more rows
```

```
gapminder %>%
  gather(
    key = "key",
    value = "value",
    lifeExp:gdpPercap
  ) %>%
  unite(temp, key, year)
```

```
## # A tibble: 5,112 x 4
##    country     continent temp         value
##
##  1 Afghanistan Asia      lifeExp_1952  28.8
##  2 Afghanistan Asia      lifeExp_1957  30.3
##  3 Afghanistan Asia      lifeExp_1962  32.0
##  4 Afghanistan Asia      lifeExp_1967  34.0
##  5 Afghanistan Asia      lifeExp_1972  36.1
##  6 Afghanistan Asia      lifeExp_1977  38.4
##  7 Afghanistan Asia      lifeExp_1982  39.9
##  8 Afghanistan Asia      lifeExp_1987  40.8
##  9 Afghanistan Asia      lifeExp_1992  41.7
## 10 Afghanistan Asia      lifeExp_1997  41.8
## # ... with 5,102 more rows
```

```
gapminder %>%
  gather(
    key = "key",
    value = "value",
    lifeExp:gdpPercap
  ) %>%
  unite(temp, key, year) %>%
  spread(
    key = temp,
    value = value
  )
```

```
## # A tibble: 142 x 38
##    country continent gdpPercap_1952 gdpPercap_1957 gdpPercap_1962
##
##  1 Afghan~ Asia                779.           821.           853.
##  2 Albania Europe            1601.          1942.          2313.
##  3 Algeria Africa            2449.          3014.          2551.
##  4 Angola  Africa            3521.          3828.          4269.
##  5 Argent~ Americas          5911.          6857.          7133.
##  6 Austra~ Oceania          10040.         10950.         12217.
##  7 Austria Europe            6137.          8843.         10751.
##  8 Bahrain Asia              9867.         11636.         12753.
##  9 Bangla~ Asia               684.           662.           686.
## 10 Belgium Europe            8343.          9715.         10991.
## # ... with 132 more rows, and 33 more variables: gdpPercap_1967 ,
## #   gdpPercap_1972 , gdpPercap_1977 , gdpPercap_1982 ,
## #   gdpPercap_1987 , gdpPercap_1992 , gdpPercap_1997 ,
## #   gdpPercap_2002 , gdpPercap_2007 , lifeExp_1952 ,
## #   lifeExp_1957 , lifeExp_1962 , lifeExp_1967 ,
## #   lifeExp_1972 , lifeExp_1977 , lifeExp_1982 ,
## #   lifeExp_1987 , lifeExp_1992 , lifeExp_1997 ,
## #   lifeExp_2002 , lifeExp_2007 , pop_1952 ,
## #   pop_1957 , pop_1962 , pop_1967 , pop_1972 ,
## #   pop_1977 , pop_1982 , pop_1987 , pop_1992 ,
## #   pop_1997 , pop_2002 , pop_2007
```

Now multiple value columns can be added to the `values_from` argument.

```
gapminder %>%
  pivot_wider(
    names_from = year,
    values_from = c(lifeExp, pop, gdpPercap)
  )
```

```
## # A tibble: 142 x 38
##    country continent lifeExp_1952 lifeExp_1957 lifeExp_1962 lifeExp_1967
##
##  1 Afghan~ Asia              28.8         30.3         32.0         34.0
##  2 Albania Europe            55.2         59.3         64.8         66.2
##  3 Algeria Africa            43.1         45.7         48.3         51.4
##  4 Angola  Africa            30.0         32.0         34           36.0
##  5 Argent~ Americas          62.5         64.4         65.1         65.6
##  6 Austra~ Oceania           69.1         70.3         70.9         71.1
##  7 Austria Europe            66.8         67.5         69.5         70.1
##  8 Bahrain Asia              50.9         53.8         56.9         59.9
##  9 Bangla~ Asia              37.5         39.3         41.2         43.5
## 10 Belgium Europe            68           69.2         70.2         70.9
## # ... with 132 more rows, and 32 more variables: lifeExp_1972 ,
## #   lifeExp_1977 , lifeExp_1982 , lifeExp_1987 ,
## #   lifeExp_1992 , lifeExp_1997 , lifeExp_2002 ,
## #   lifeExp_2007 , pop_1952 , pop_1957 , pop_1962 ,
## #   pop_1967 , pop_1972 , pop_1977 , pop_1982 ,
## #   pop_1987 , pop_1992 , pop_1997 , pop_2002 ,
## #   pop_2007 , gdpPercap_1952 , gdpPercap_1957 ,
## #   gdpPercap_1962 , gdpPercap_1967 , gdpPercap_1972 ,
## #   gdpPercap_1977 , gdpPercap_1982 , gdpPercap_1987 ,
## #   gdpPercap_1992 , gdpPercap_1997 , gdpPercap_2002 ,
## #   gdpPercap_2007
```

Multiple `names_from` columns

Now suppose that our starting dataset is `gapminder_long` and we want one row per country.

```
gapminder_long
```

```
## # A tibble: 5,112 x 5
##    country     continent  year measure        value
##
##  1 Afghanistan Asia       1952 lifeExp         28.8
##  2 Afghanistan Asia       1952 pop        8425333
##  3 Afghanistan Asia       1952 gdpPercap      779.
##  4 Afghanistan Asia       1957 lifeExp         30.3
##  5 Afghanistan Asia       1957 pop        9240934
##  6 Afghanistan Asia       1957 gdpPercap      821.
##  7 Afghanistan Asia       1962 lifeExp         32.0
##  8 Afghanistan Asia       1962 pop       10267083
##  9 Afghanistan Asia       1962 gdpPercap      853.
## 10 Afghanistan Asia       1967 lifeExp         34.0
## # ... with 5,102 more rows
```

In this situation, we want both the values of `measure` and `year` to make up the new column names. Rather than having to combine them first, we can pass both into the `names_from` argument.

```
gapminder_long %>%
  pivot_wider(
    names_from = c(measure, year),
    values_from = value
  )
```

```
## # A tibble: 142 x 38
##    country continent lifeExp_1952 pop_1952 gdpPercap_1952 lifeExp_1957
##
##  1 Afghan~ Asia              28.8  8425333           779.         30.3
##  2 Albania Europe            55.2  1282697          1601.         59.3
##  3 Algeria Africa            43.1  9279525          2449.         45.7
##  4 Angola  Africa            30.0  4232095          3521.         32.0
##  5 Argent~ Americas          62.5 17876956          5911.         64.4
##  6 Austra~ Oceania           69.1  8691212         10040.         70.3
##  7 Austria Europe            66.8  6927772          6137.         67.5
##  8 Bahrain Asia              50.9   120447          9867.         53.8
##  9 Bangla~ Asia              37.5 46886859           684.         39.3
## 10 Belgium Europe            68    8730405          8343.         69.2
## # ... with 132 more rows, and 32 more variables: pop_1957 ,
## #   gdpPercap_1957 , lifeExp_1962 , pop_1962 ,
## #   gdpPercap_1962 , lifeExp_1967 , pop_1967 ,
## #   gdpPercap_1967 , lifeExp_1972 , pop_1972 ,
## #   gdpPercap_1972 , lifeExp_1977 , pop_1977 ,
## #   gdpPercap_1977 , lifeExp_1982 , pop_1982 ,
## #   gdpPercap_1982 , lifeExp_1987 , pop_1987 ,
## #   gdpPercap_1987 , lifeExp_1992 , pop_1992 ,
## #   gdpPercap_1992 , lifeExp_1997 , pop_1997 ,
## #   gdpPercap_1997 , lifeExp_2002 , pop_2002 ,
## #   gdpPercap_2002 , lifeExp_2007 , pop_2007 ,
## #   gdpPercap_2007
```

`pivot_longer()` new features

`names_prefix` and `names_ptypes`

Earlier when pivoting `gapminder_wide_gdp` we noticed that it would require additional cleaning to extract the `year` out of the original column names.

```
gapminder_wide_gdp
```

```
## # A tibble: 142 x 14
##    country continent gdpPercap_1952 gdpPercap_1957 gdpPercap_1962
##                                                                  
##  1 Afghan~ Asia                779.           821.           853.
##  2 Albania Europe             1601.          1942.          2313.
##  3 Algeria Africa             2449.          3014.          2551.
##  4 Angola  Africa             3521.          3828.          4269.
##  5 Argent~ Americas           5911.          6857.          7133.
##  6 Austra~ Oceania           10040.         10950.         12217.
##  7 Austria Europe             6137.          8843.         10751.
##  8 Bahrain Asia               9867.         11636.         12753.
##  9 Bangla~ Asia                684.           662.           686.
## 10 Belgium Europe             8343.          9715.         10991.
## # ... with 132 more rows, and 9 more variables: gdpPercap_1967 ,
## #   gdpPercap_1972 , gdpPercap_1977 , gdpPercap_1982 ,
## #   gdpPercap_1987 , gdpPercap_1992 , gdpPercap_1997 ,
## #   gdpPercap_2002 , gdpPercap_2007
```

```
gapminder_wide_gdp %>%
  pivot_longer(
    gdpPercap_1952:gdpPercap_2007,
    names_to = "year",
    values_to = "gdpPercap"
  )
```

```
## # A tibble: 1,704 x 4
##    country     continent year           gdpPercap
##                                                  
##  1 Afghanistan Asia      gdpPercap_1952      779.
##  2 Afghanistan Asia      gdpPercap_1957      821.
##  3 Afghanistan Asia      gdpPercap_1962      853.
##  4 Afghanistan Asia      gdpPercap_1967      836.
##  5 Afghanistan Asia      gdpPercap_1972      740.
##  6 Afghanistan Asia      gdpPercap_1977      786.
##  7 Afghanistan Asia      gdpPercap_1982      978.
##  8 Afghanistan Asia      gdpPercap_1987      852.
##  9 Afghanistan Asia      gdpPercap_1992      649.
## 10 Afghanistan Asia      gdpPercap_1997      635.
## # ... with 1,694 more rows
```

The argument `names_prefix` allows us to remove the prefix from the column names.

```
gapminder_wide_gdp %>%
  pivot_longer(
    gdpPercap_1952:gdpPercap_2007,
    names_to = "year",
    names_prefix = "gdpPercap_",
    values_to = "gdpPercap"
  )
```

```
## # A tibble: 1,704 x 4
##    country     continent year  gdpPercap
##                                         
##  1 Afghanistan Asia      1952       779.
##  2 Afghanistan Asia      1957       821.
##  3 Afghanistan Asia      1962       853.
##  4 Afghanistan Asia      1967       836.
##  5 Afghanistan Asia      1972       740.
##  6 Afghanistan Asia      1977       786.
##  7 Afghanistan Asia      1982       978.
##  8 Afghanistan Asia      1987       852.
##  9 Afghanistan Asia      1992       649.
## 10 Afghanistan Asia      1997       635.
## # ... with 1,694 more rows
```

Additionally, `year` shouldn't be a character vector, it makes more sense as an integer. We can set the type using `names_ptypes`.

```
gapminder_wide_gdp %>%
  pivot_longer(
    gdpPercap_1952:gdpPercap_2007,
    names_to = "year",
    names_prefix = "gdpPercap_",
    names_ptypes = list(year = integer()),
    values_to = "gdpPercap"
  )
```

```
## # A tibble: 1,704 x 4
##    country     continent  year gdpPercap
##
##  1 Afghanistan Asia       1952      779.
##  2 Afghanistan Asia       1957      821.
##  3 Afghanistan Asia       1962      853.
##  4 Afghanistan Asia       1967      836.
##  5 Afghanistan Asia       1972      740.
##  6 Afghanistan Asia       1977      786.
##  7 Afghanistan Asia       1982      978.
##  8 Afghanistan Asia       1987      852.
##  9 Afghanistan Asia       1992      649.
## 10 Afghanistan Asia       1997      635.
## # ... with 1,694 more rows
```

Multiple `names_to` columns

As promised, let's revisit `gapminder_wide`. In a prior section we tidied a simplified version of this, now let's try to do the whole thing.

```
gapminder_wide
```

```
## # A tibble: 142 x 38
##    country continent lifeExp_1952 pop_1952 gdpPercap_1952 lifeExp_1957
##
##  1 Afghan~ Asia              28.8  8425333           779.         30.3
##  2 Albania Europe            55.2  1282697          1601.         59.3
##  3 Algeria Africa            43.1  9279525          2449.         45.7
##  4 Angola  Africa            30.0  4232095          3521.         32.0
##  5 Argent~ Americas          62.5 17876956          5911.         64.4
##  6 Austra~ Oceania           69.1  8691212         10040.         70.3
##  7 Austria Europe            66.8  6927772          6137.         67.5
##  8 Bahrain Asia              50.9   120447          9867.         53.8
##  9 Bangla~ Asia              37.5 46886859           684.         39.3
## 10 Belgium Europe            68    8730405          8343.         69.2
## # ... with 132 more rows, and 32 more variables: pop_1957 ,
## #   gdpPercap_1957 , lifeExp_1962 , pop_1962 ,
## #   gdpPercap_1962 , lifeExp_1967 , pop_1967 ,
## #   gdpPercap_1967 , lifeExp_1972 , pop_1972 ,
## #   gdpPercap_1972 , lifeExp_1977 , pop_1977 ,
## #   gdpPercap_1977 , lifeExp_1982 , pop_1982 ,
## #   gdpPercap_1982 , lifeExp_1987 , pop_1987 ,
## #   gdpPercap_1987 , lifeExp_1992 , pop_1992 ,
## #   gdpPercap_1992 , lifeExp_1997 , pop_1997 ,
## #   gdpPercap_1997 , lifeExp_2002 , pop_2002 ,
## #   gdpPercap_2002 , lifeExp_2007 , pop_2007 ,
## #   gdpPercap_2007
```

The final goal is to have columns `country`, `continent`, `year`, `lifeExp`, `pop`, and `gdpPercap`. We can't do this all in one step, so let's first just gather all of the value columns.

```
gapminder_wide %>%
  pivot_longer(
    lifeExp_1952:gdpPercap_2007
  )
```

```
## # A tibble: 5,112 x 4
##    country     continent name                 value
##
##  1 Afghanistan Asia      lifeExp_1952          28.8
##  2 Afghanistan Asia      pop_1952        8425333
##  3 Afghanistan Asia      gdpPercap_1952      779.
##  4 Afghanistan Asia      lifeExp_1957          30.3
##  5 Afghanistan Asia      pop_1957        9240934
##  6 Afghanistan Asia      gdpPercap_1957      821.
##  7 Afghanistan Asia      lifeExp_1962          32.0
##  8 Afghanistan Asia      pop_1962       10267083
##  9 Afghanistan Asia      gdpPercap_1962      853.
## 10 Afghanistan Asia      lifeExp_1967          34.0
## # ... with 5,102 more rows
```

The `name` column has two parts, the `measure` and the `year`. We can use `tidyr::separate()` to break it up.

```
gapminder_wide %>%
  pivot_longer(
    lifeExp_1952:gdpPercap_2007
  ) %>%
  separate(
    col = "name",
    into = c("measure", "year"),
    sep = "_"
  )
```

```
## # A tibble: 5,112 x 5
##    country     continent measure   year       value
##
##  1 Afghanistan Asia      lifeExp   1952        28.8
##  2 Afghanistan Asia      pop       1952   8425333
##  3 Afghanistan Asia      gdpPercap 1952       779.
##  4 Afghanistan Asia      lifeExp   1957        30.3
##  5 Afghanistan Asia      pop       1957   9240934
##  6 Afghanistan Asia      gdpPercap 1957       821.
##  7 Afghanistan Asia      lifeExp   1962        32.0
##  8 Afghanistan Asia      pop       1962  10267083
##  9 Afghanistan Asia      gdpPercap 1962       853.
## 10 Afghanistan Asia      lifeExp   1967        34.0
## # ... with 5,102 more rows
```

Rather than using `separate()`, we can specify multiple `names_to` columns in `pivot_longer()` along with the `names_sep` argument.

```
gapminder_wide %>%
  pivot_longer(
    lifeExp_1952:gdpPercap_2007,
    names_to = c("measure", "year"),
    names_sep = "_"
  )
```

```
## # A tibble: 5,112 x 5
##    country     continent measure   year       value
##
##  1 Afghanistan Asia      lifeExp   1952        28.8
##  2 Afghanistan Asia      pop       1952   8425333
##  3 Afghanistan Asia      gdpPercap 1952       779.
##  4 Afghanistan Asia      lifeExp   1957        30.3
##  5 Afghanistan Asia      pop       1957   9240934
##  6 Afghanistan Asia      gdpPercap 1957       821.
##  7 Afghanistan Asia      lifeExp   1962        32.0
##  8 Afghanistan Asia      pop       1962  10267083
##  9 Afghanistan Asia      gdpPercap 1962       853.
## 10 Afghanistan Asia      lifeExp   1967        34.0
## # ... with 5,102 more rows
```

`names_pattern` is a more flexible way to specify how to split up the names. It uses regex and will be necessary for more complex naming patterns.

In our previous example, we can get the same behavior by using the regex `"(.+)_(.+)"`.

```
gapminder_wide %>%
  pivot_longer(
    lifeExp_1952:gdpPercap_2007,
    names_to = c("measure", "year"),
    names_pattern = "(.+)_(.+)"
  )
```

```
## # A tibble: 5,112 x 5
##    country     continent measure   year       value
##
##  1 Afghanistan Asia      lifeExp   1952        28.8
##  2 Afghanistan Asia      pop       1952   8425333
##  3 Afghanistan Asia      gdpPercap 1952       779.
##  4 Afghanistan Asia      lifeExp   1957        30.3
##  5 Afghanistan Asia      pop       1957   9240934
##  6 Afghanistan Asia      gdpPercap 1957       821.
##  7 Afghanistan Asia      lifeExp   1962        32.0
##  8 Afghanistan Asia      pop       1962  10267083
##  9 Afghanistan Asia      gdpPercap 1962       853.
## 10 Afghanistan Asia      lifeExp   1967        34.0
## # ... with 5,102 more rows
```

Now that we've broken up the column names, the final step is to use `pivot_wider()` to create columns for `lifeExp`, `pop`, and `gdpPercap`.

```
gapminder_wide %>%
  pivot_longer(
    lifeExp_1952:gdpPercap_2007,
    names_to = c("measure", "year"),
    names_sep = "_"
  ) %>%
  pivot_wider(
    names_from = measure,
    values_from = value
  )
```

```
## # A tibble: 1,704 x 6
##    country     continent year  lifeExp      pop gdpPercap
##                                                          
##  1 Afghanistan Asia      1952     28.8  8425333      779.
##  2 Afghanistan Asia      1957     30.3  9240934      821.
##  3 Afghanistan Asia      1962     32.0 10267083      853.
##  4 Afghanistan Asia      1967     34.0 11537966      836.
##  5 Afghanistan Asia      1972     36.1 13079460      740.
##  6 Afghanistan Asia      1977     38.4 14880372      786.
##  7 Afghanistan Asia      1982     39.9 12881816      978.
##  8 Afghanistan Asia      1987     40.8 13867957      852.
##  9 Afghanistan Asia      1992     41.7 16317921      649.
## 10 Afghanistan Asia      1997     41.8 22227415      635.
## # ... with 1,694 more rows
```

Conclusion

The new `tidyr` functions have intuitive syntax, are easy to use, and are more flexibile than the prior functions. Several of the new arguments and features are extremely useful, and will save lots of time on common tasks.