Wide and long data format - use of `{tidyr}`

Overview

> Data scientists, according to interviews and expert estimates, spend from 50 percent to 80 percent of their time mired in the mundane labor of collecting and preparing data, before it can be explored for useful information.

This tutorial covers the *very basics* of `tidyr` package. Usage:

- General data handling & processing
- `ggplot2` plots (long format required)
- Eurostat: data in long format, wide format often required for regression analysis
- Panel data analysis

Wide format

- For **cross sections**, row corresponds to an individual (person, firm, etc.) and each column corresponds to a variable (age, height, weight).
- For **time series**, row corresponds to a time period (year, quarter, day, etc.) and each column corresponds to a variable (GDP per capita, Unemployment rate).
- Wide format is used in many `R`-based applications: linear regression, VAR models, etc.

Long format

- For **cross sections**, data is stretched so that a single individual may occupy multiple rows.
- The same applies to **time series**

Time Series Example - wide format

```r
GDPwide <- data.frame(
  Year = c(2009:2018),
  GER = c(411,723,325,456,579,612,709,513,527,379),
  FRA = c(123,300,400,500,600,654,789,906,413,567),
  USA = c(957,1000,569,896,956,1345,780,599,1023,678)
)
GDPwide # artifical/made-up data used here
```

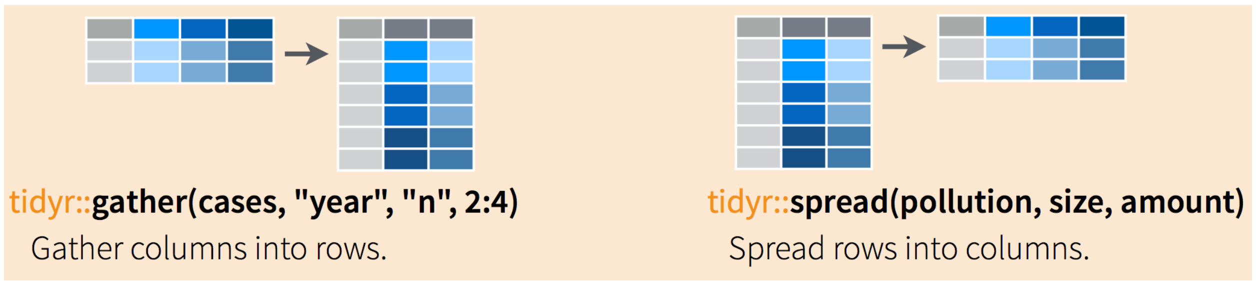
```
##    Year GER FRA  USA
## 1  2009 411 123  957
## 2  2010 723 300 1000
## 3  2011 325 400  569
## 4  2012 456 500  896
## 5  2013 579 600  956
## 6  2014 612 654 1345
## 7  2015 709 789  780
## 8  2016 513 906  599
## 9  2017 527 413 1023
## 10 2018 379 567  678
```

- Above is the **wide format** - GDP data are shown for different states and years
- In **long format** (below), the same data are shown, yet each `Year` and `Country` combination is on a separate row of the dataset.
  - Basically, columns (*variables* in R) `Year`, `Country` and `GDP` are used in the long format to display the same data.
  - Note that `Country` and `GDP` *variables* need to be created to get the long format (such *variables* do not exist in the wide format dataset)

```
##    Year State  GDP
## 1  2009   GER  411
## 2  2010   GER  723
## 3  2011   GER  325
## 4  2012   GER  456
## 5  2013   GER  579
## 6  2014   GER  612
## 7  2015   GER  709
## 8  2016   GER  513
## 9  2017   GER  527
## 10 2018   GER  379
## 11 2009   FRA  123
## 12 2010   FRA  300
## 13 2011   FRA  400
## 14 2012   FRA  500
## 15 2013   FRA  600
## 16 2014   FRA  654
## 17 2015   FRA  789
## 18 2016   FRA  906
## 19 2017   FRA  413
## 20 2018   FRA  567
## 21 2009   USA  957
## 22 2010   USA 1000
## 23 2011   USA  569
## 24 2012   USA  896
## 25 2013   USA  956
## 26 2014   USA 1345
## 27 2015   USA  780
## 28 2016   USA  599
## 29 2017   USA 1023
## 30 2018   USA  678
```

`gather()` and `spread()` commands

- `gather()` converts wide format to long
- `spread()` converts long format to wide


tidyr::gather(cases, "year", "n", 2:4)
Gather columns into rows.

tidyr::spread(pollution, size, amount)
Spread rows into columns.

Convert `GDPwide` to long format

`gather(GDPwide, key = "State", value = "GDP", 2:4)`

- `GDPwide` dataframe to reshape - convert to long format
- `"State"` is the name of the new key column - a new *variable* (can be any character string you supply). The keys are generated based on column names of the columns being transformed (`2:4`)
  - *variable*-names in the wide format dataframe (`GER`,`FRA`,`USA`) are used as "entries" (keys) in the new column (i.e. variable `State`).
- `"GDP"` name of the new value column (any character string you supply)
- `2:4` numeric indexes (or names) of columns to collapse.
- Please note that column `Year` (1st column) is left out (Year is the grey column as in the illustration).

```r
GDPlong <- gather(GDPwide, key = "State", value = "GDP", 2:4)
print(GDPlong)
```

```
##    Year State  GDP
## 1  2009   GER  411
## 2  2010   GER  723
## 3  2011   GER  325
## 4  2012   GER  456
## 5  2013   GER  579
## 6  2014   GER  612
## 7  2015   GER  709
## 8  2016   GER  513
## 9  2017   GER  527
## 10 2018   GER  379
## 11 2009   FRA  123
## 12 2010   FRA  300
## 13 2011   FRA  400
## 14 2012   FRA  500
## 15 2013   FRA  600
## 16 2014   FRA  654
## 17 2015   FRA  789
## 18 2016   FRA  906
## 19 2017   FRA  413
## 20 2018   FRA  567
## 21 2009   USA  957
## 22 2010   USA 1000
## 23 2011   USA  569
## 24 2012   USA  896
## 25 2013   USA  956
## 26 2014   USA 1345
## 27 2015   USA  780
## 28 2016   USA  599
## 29 2017   USA 1023
## 30 2018   USA  678
```

Please note that `GDPlong` is suitable for panel data analysis - both `id` and `time` identification is provided in each row.

Convert `GDPlong` to wide format

`spread(GDPlong, key = "State", value = "GDP")`

- `GDPlong` dataframe to reshape - convert to wide format
- `key = "State"` column to use for keys (new columns names, i.e. new *variables*)
- `value = "GDP"` column to use for values (data-cells in the new (wide) dataframe)
- Please note that column `Year` is left out from the syntax (Year is the grey column as in the illustration).

```
GDPwide_2 <- spread(GDPlong, key = "State", value = "GDP")
GDPwide_2
```

```
##    Year FRA GER  USA
## 1  2009 123 411  957
## 2  2010 300 723 1000
## 3  2011 400 325  569
## 4  2012 500 456  896
## 5  2013 600 579  956
## 6  2014 654 612 1345
## 7  2015 789 709  780
## 8  2016 906 513  599
## 9  2017 413 527 1023
## 10 2018 567 379  678
```

```
GDPwide # for comparison
```

```
##    Year GER FRA  USA
## 1  2009 411 123  957
## 2  2010 723 300 1000
## 3  2011 325 400  569
## 4  2012 456 500  896
## 5  2013 579 600  956
## 6  2014 612 654 1345
## 7  2015 709 789  780
## 8  2016 513 906  599
## 9  2017 527 413 1023
## 10 2018 379 567  678
```

Note that `GDPwide_2` columns are alphabetically organized now.

Alternatively, we may choose a *"transposed"* wide format:

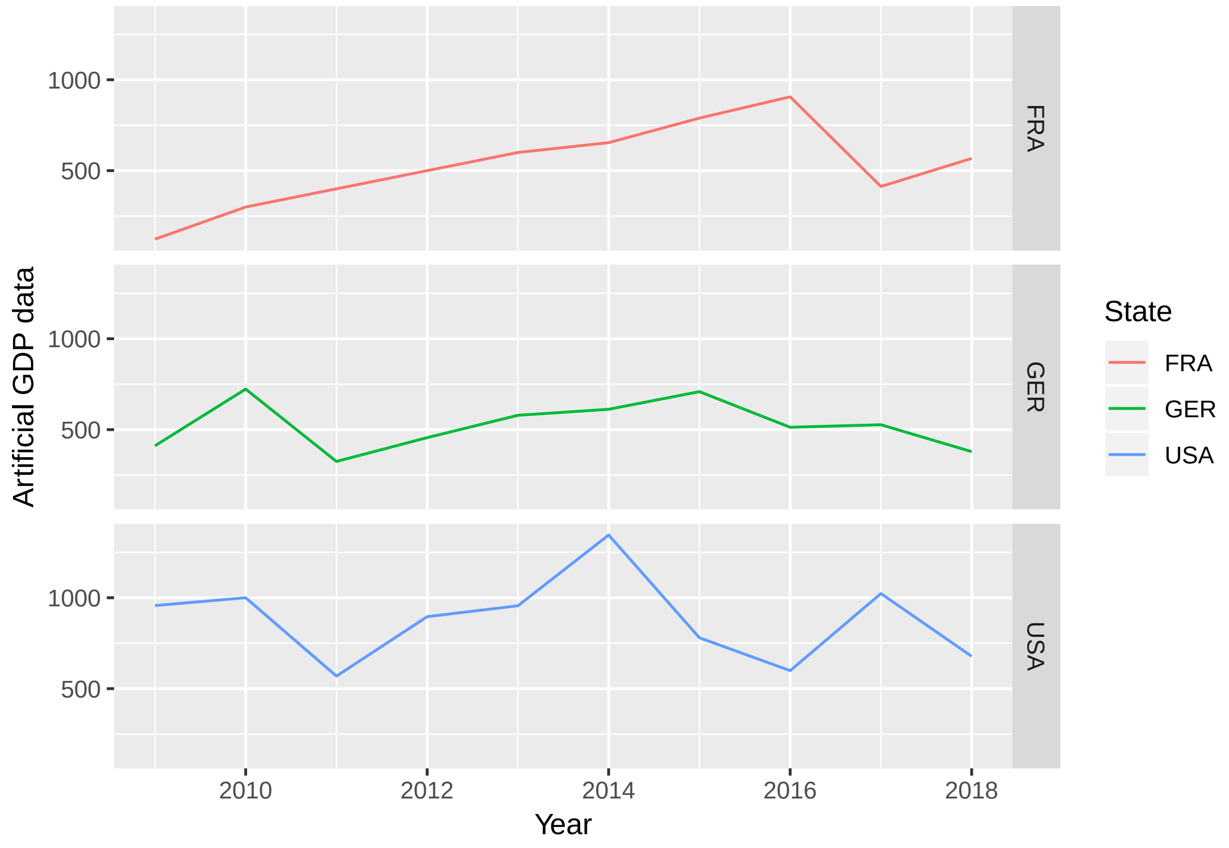
```
spread(GDPlong, key = "Year", value = "GDP")
```

```
##   State 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018
## 1   FRA  123  300  400  500  600  654  789  906  413  567
## 2   GER  411  723  325  456  579  612  709  513  527  379
## 3   USA  957 1000  569  896  956 1345  780  599 1023  678
```

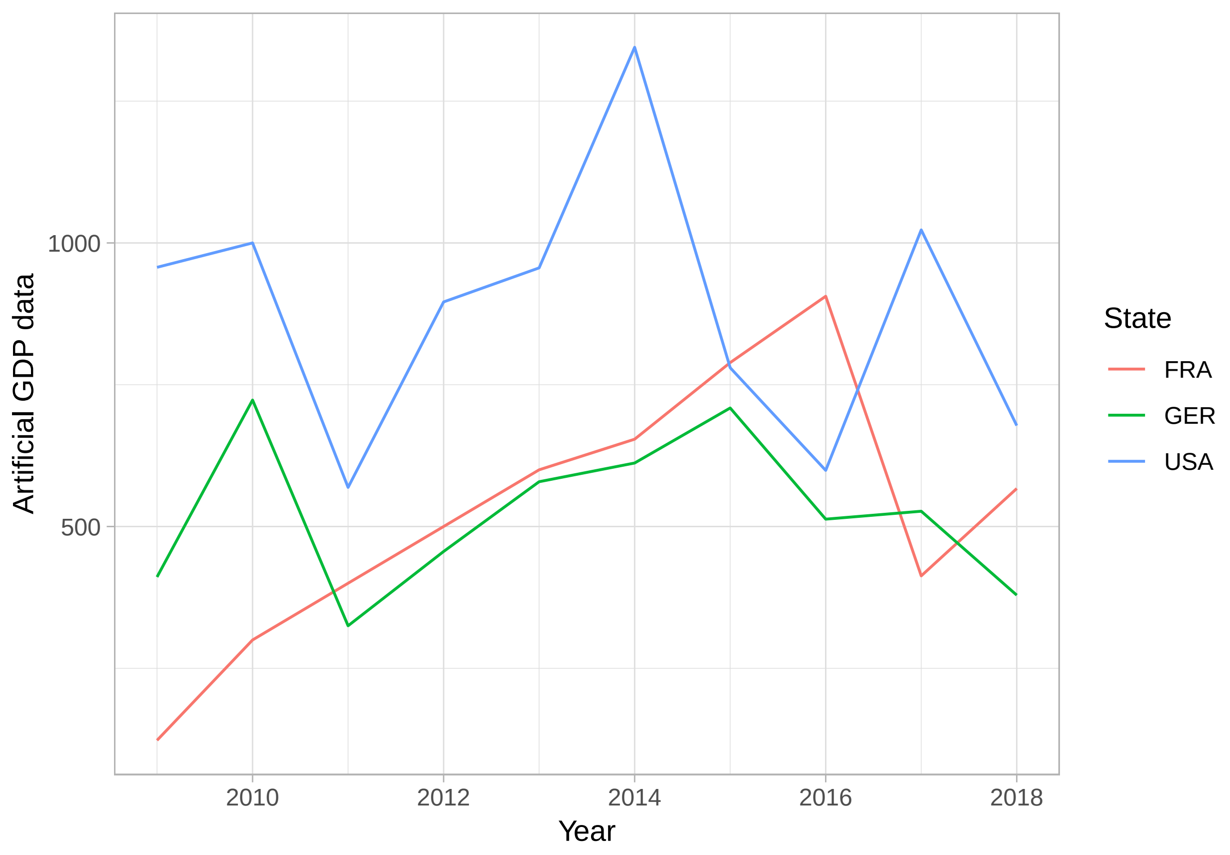
- However, this formatting is not very useful for most econometric applications.

Data in long format are used in panel data regression, plotting by {ggplot2}, etc.

```r
Plot1 <- ggplot(data = GDPlong, aes(x = Year, y = GDP))+
  geom_line(aes(color=State))+
  xlab("Year")+ylab("Artificial GDP data")+
  facet_grid(State~.)
Plot1
```



```r
Plot2 <- ggplot(data = GDPlong, aes(x = Year, y = GDP))+
  geom_line(aes(color=State))+
  xlab("Year")+ylab("Artificial GDP data")+
  theme_light() # remove faceting, include theme "light"
Plot2
```



Advanced data reshaping tools

`{tidyr}` provides sufficient functionality for many empirical tasks.

If necessary, additional control over reshaping data may be obtained throught the `{reshape2}` package:

- `melt()` from `{reshape2}` expands the functionality of `gather()`
- `dcast()` from `{reshape2}` expands the functionality of `spread()`

For additional information on `{reshape2}`, see:

- Tutorial by Timothy Carsel
- Tutorial by Sean C. Anderson
- reshape2 pdf
- `R09_Eurostat_in_depth_self_study_material.R` (in your R working directory)

`{tidyr}` Excercise

The following wide-format dataset contains actual GDP per capita in Spain (from Eurostat)

- Euro per inhabitant in percentage of the EU average (100 = EU's average in a given year)
- 2005 - 2016 data
- Measured at the NUTS2 level

```r
GDPSpain <- read.csv("datasets/GDPSpain.csv")
head(GDPSpain[,1:8],12) # only columns 1 to 8 are shown
```

```
##    time ES11 ES12 ES13 ES21 ES22 ES23 ES24
## 1  2005   75   80   85  113  113   97   98
## 2  2006   77   83   86  116  114   98   99
## 3  2007   79   83   86  116  113   98  100
## 4  2008   81   86   87  120  115  100  102
## 5  2009   84   86   89  122  118  101  103
## 6  2010   81   83   85  118  113   98  101
## 7  2011   77   80   81  114  109   94   96
## 8  2012   73   75   77  110  103   90   91
## 9  2013   73   73   75  108  102   89   91
## 10 2014   71   71   74  107  101   88   89
## 11 2015   71   70   72  106  100   87   87
## 12 2016   74   72   74  109  103   87   90
```

```r
str(GDPSpain)
```

```
## 'data.frame':    12 obs. of  20 variables:
##  $ time: int  2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 ...
##  $ ES11: int  75 77 79 81 84 81 77 73 73 71 ...
##  $ ES12: int  80 83 83 86 86 83 80 75 73 71 ...
##  $ ES13: int  85 86 86 87 89 85 81 77 75 74 ...
##  $ ES21: int  113 116 116 120 122 118 114 110 108 107 ...
##  $ ES22: int  113 114 113 115 118 113 109 103 102 101 ...
##  $ ES23: int  97 98 98 100 101 98 94 90 89 88 ...
##  $ ES24: int  98 99 100 102 103 101 96 91 91 89 ...
##  $ ES30: int  120 122 121 123 128 122 119 114 113 111 ...
##  $ ES41: int  84 84 85 86 89 86 83 79 77 76 ...
##  $ ES42: int  73 74 74 75 77 74 71 67 66 63 ...
##  $ ES43: int  61 61 62 64 66 64 61 58 57 55 ...
##  $ ES51: int  107 109 108 108 111 107 102 98 97 96 ...
##  $ ES52: int  83 84 83 84 84 81 77 72 72 71 ...
##  $ ES53: int  101 100 98 98 99 95 91 87 86 85 ...
##  $ ES61: int  70 71 71 71 72 69 66 63 61 60 ...
##  $ ES62: int  76 77 76 78 78 75 71 68 68 66 ...
##  $ ES63: int  79 79 78 79 82 77 73 69 69 66 ...
##  $ ES64: int  78 77 74 75 77 72 68 63 62 60 ...
##  $ ES70: int  84 83 81 81 82 79 76 71 70 68 ...
```

```r
dim(GDPSpain)
```

```
## [1] 12 20
```

Using the `gather()` command, convert `GDPSpain` to a long format

```r
# uncomment the following line and comlete the command
# GDPSpain_L <- gather()
```

Use the `spread()` command to transform `GDPSpain_L` (long format) back to the original wide format

```r
# uncomment the following line and comlete the command
# GDPSpain_W2 <- spread()
```

As a bonus task, you may use `ggplot2` to plot the data (selected regions).

This is just a quick introduction to `tidyr`. For detailed discussion on the topic, you may have a look here