8. Preparing Data for Analysis - tidyr

Data Science for OR - J. Duggan

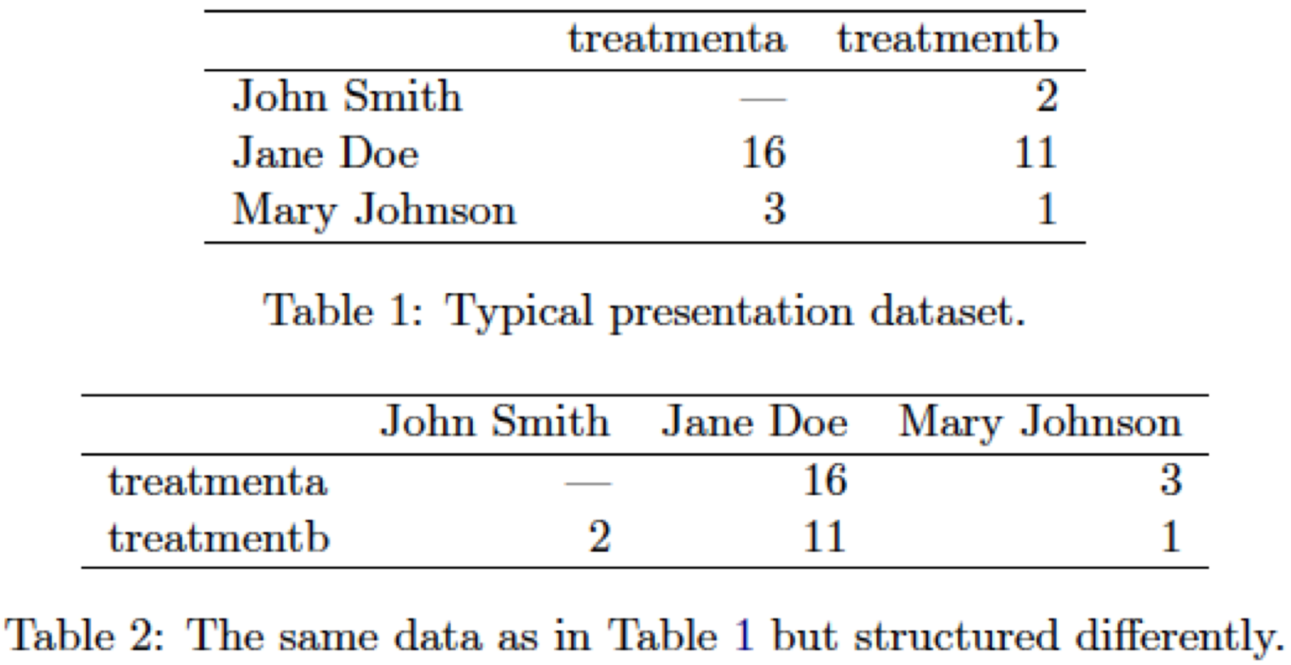
Tidy Data - Overview

- What is data tidying?
  - Structuring datasets to facilitate analysis
- The tidy data standard is designed to:
  - Facilitate initial exploration and analysis of data
  - Simplify the development of data analysis tools that work well together
- Principles closely related to relational algebra (Codd 1990)

Why Tidy Data (Wickham 2017)

- Advantage to picking one consistent way of storing data. Easier to learn tools that work with tidy data because they have a underlying uniformity
- Specific advantage to placing variables in columns because it allows R's vectorised functions to shine.
- dplyr, ggplot2 designed to work with tidy data

A Typical Presentation Data Set (Wickham 2014)



	treatmenta	treatmentb
John Smith	—	2
Jane Doe	16	11
Mary Johnson	3	1

Table 1: Typical presentation dataset.

	John Smith	Jane Doe	Mary Johnson
treatmenta	—	16	3
treatmentb	2	11	1

Table 2: The same data as in Table 1 but structured differently.

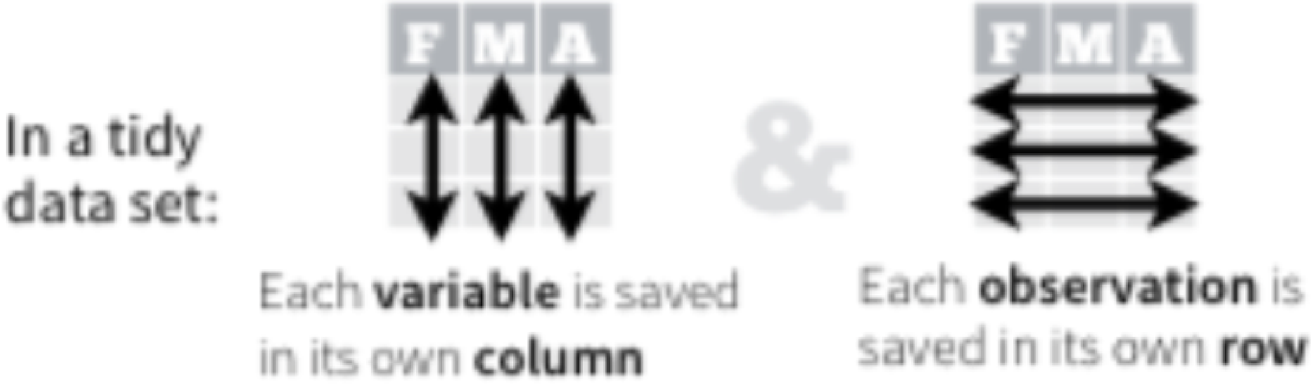
In R

```
untidy <- tibble(name=c("John Smith","Jane Doe",
                        "Mary Johnson"),
                 treatmenta=c(NA, 16, 3),
                 treatmentb = c(2, 11, 1))
untidy
```

```
## # A tibble: 3 x 3
##   name         treatmenta treatmentb
##   <chr>             <dbl>      <dbl>
## 1 John Smith           NA          2
## 2 Jane Doe             16         11
## 3 Mary Johnson          3          1
```

Rules for a Tidy Data Set

- Each variable must have its own column
- Each observation must have its own row
- Each value must have its own cell


In a tidy data set:
F
M
A
&
F
M
A
Each **variable** is saved in its own **column**
Each **observation** is saved in its own **row**

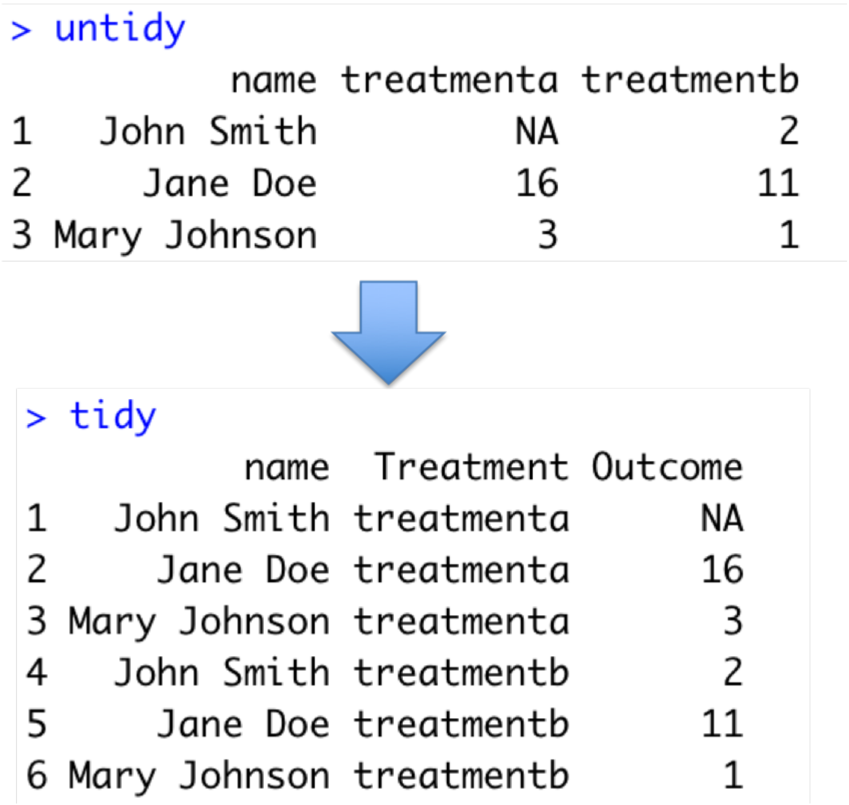
Problems with the data set

- Treatment types (treatmenta or treatmentb) are column names
- Good for presentation, not for automated analysis
- There are 6 observations, and three variables (Person, Treatment, Outcome)

```
untidy
```

```
## # A tibble: 3 x 3
##   name         treatmenta treatmentb
##   <chr>             <dbl>      <dbl>
## 1 John Smith           NA          2
## 2 Jane Doe             16         11
## 3 Mary Johnson          3          1
```

The Goal



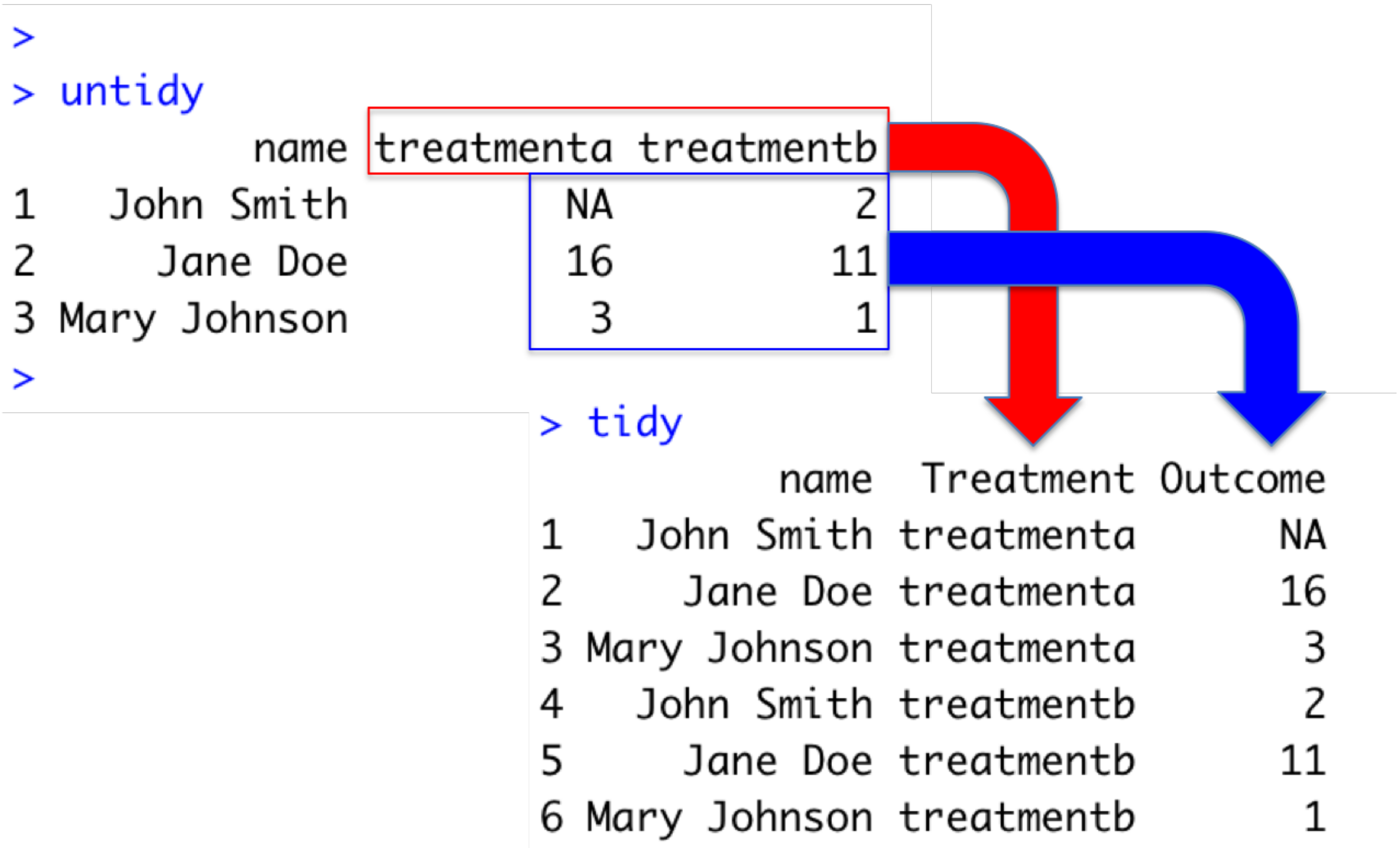
```
> untidy
          name treatmenta treatmentb
1   John Smith         NA          2
2     Jane Doe         16         11
3 Mary Johnson          3          1
```

```
> tidy
          name  Treatment Outcome
1   John Smith treatmenta      NA
2     Jane Doe treatmenta      16
3 Mary Johnson treatmenta       3
4   John Smith treatmentb       2
5     Jane Doe treatmentb      11
6 Mary Johnson treatmentb       1
```

tidyr package - 4 key functions

- **gather()** takes multiple columns, and gathers them into key-value pairs: it makes "wide" data longer
- **separate()** splits a single column into multiple columns
- **spread()** takes two columns (key and value) and spreads into multiple columns, it makes long data wider
- **unite()** combines multiple columns into a single column

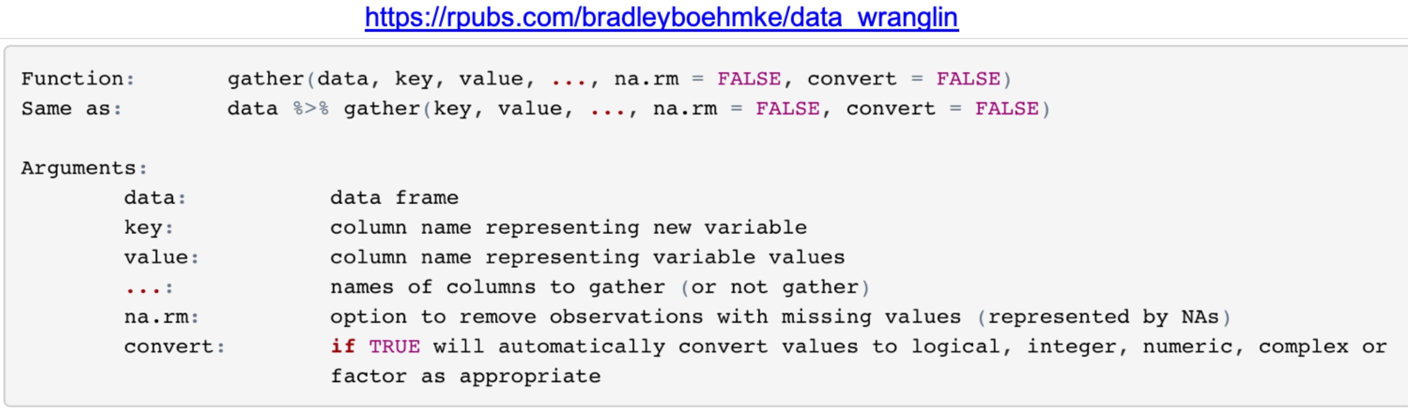
The Gather Process



```
>
> untidy
          name treatmenta treatmentb
1   John Smith         NA          2
2     Jane Doe         16         11
3 Mary Johnson          3          1
>
```

```
> tidy
          name  Treatment Outcome
1   John Smith treatmenta      NA
2     Jane Doe treatmenta      16
3 Mary Johnson treatmenta       3
4   John Smith treatmentb       2
5     Jane Doe treatmentb      11
6 Mary Johnson treatmentb       1
```

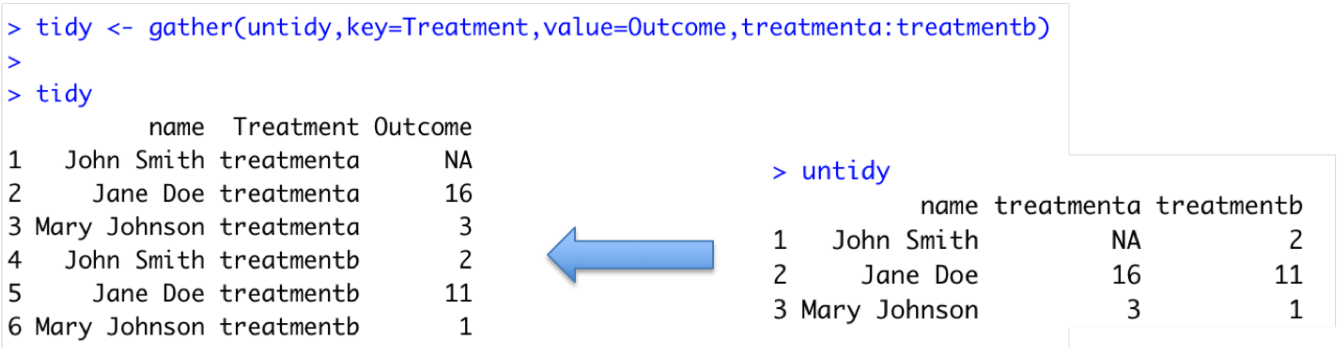
Function Call



https://rpubs.com/bradleyboehmke/data_wranglin

```
Function:       gather(data, key, value, ..., na.rm = FALSE, convert = FALSE)
Same as:        data %>% gather(key, value, ..., na.rm = FALSE, convert = FALSE)

Arguments:
        data:           data frame
        key:            column name representing new variable
        value:          column name representing variable values
        ...:            names of columns to gather (or not gather)
        na.rm:          option to remove observations with missing values (represented by NAs)
        convert:        if TRUE will automatically convert values to logical, integer, numeric, complex or
                        factor as appropriate
```

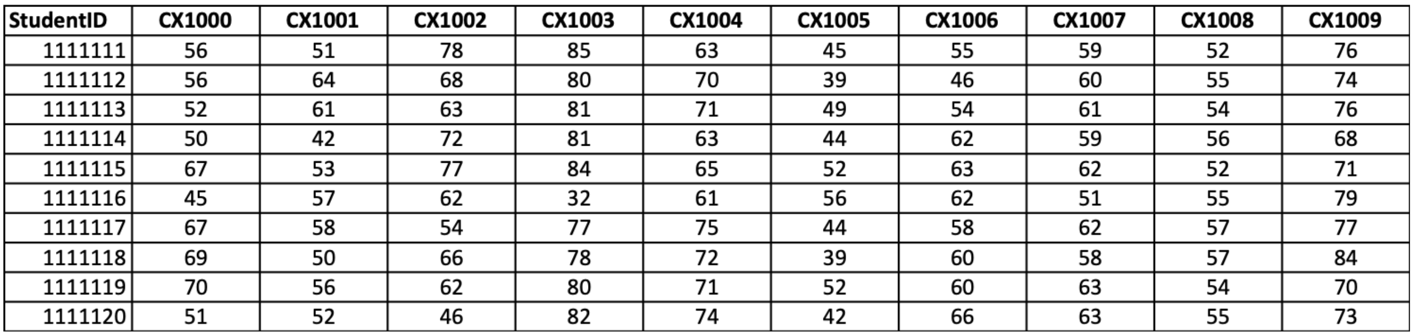


```
> tidy <- gather(untidy,key=Treatment,value=Outcome,treatmenta:treatmentb)
>
> tidy
          name  Treatment Outcome
1   John Smith treatmenta      NA
2     Jane Doe treatmenta      16
3 Mary Johnson treatmenta       3
4   John Smith treatmentb       2
5     Jane Doe treatmentb      11
6 Mary Johnson treatmentb       1
```

```
> untidy
          name treatmenta treatmentb
1   John Smith         NA          2
2     Jane Doe         16         11
3 Mary Johnson          3          1
```

Challenge 3.2

Convert the following data to tidy data format. Process the resulting data using ggplot2 and dplyr.

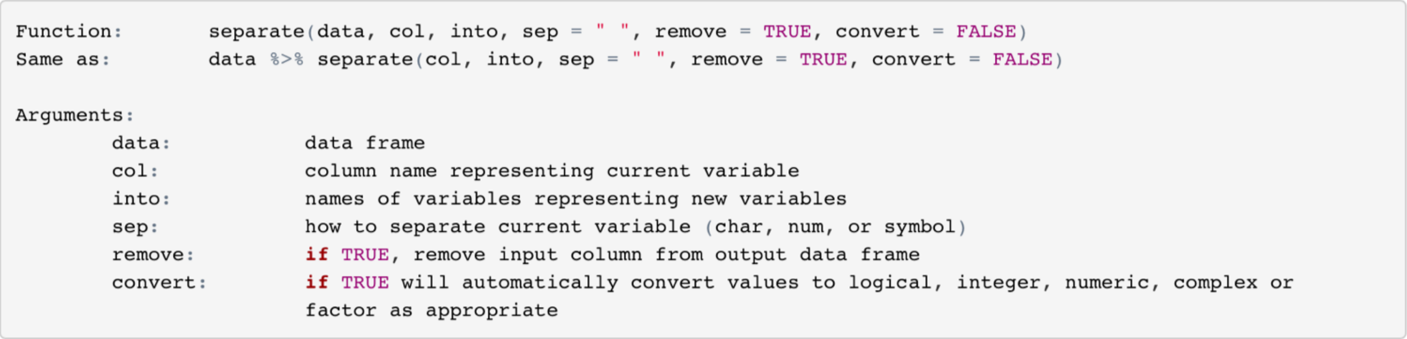


StudentID	CX1000	CX1001	CX1002	CX1003	CX1004	CX1005	CX1006	CX1007	CX1008	CX1009
1111111	56	51	78	85	63	45	55	59	52	76
1111112	56	64	68	80	70	39	46	60	55	74
1111113	52	61	63	81	71	49	54	61	54	76
1111114	50	42	72	81	63	44	62	59	56	68
1111115	67	53	77	84	65	52	63	62	52	71
1111116	45	57	62	32	61	56	62	51	55	79
1111117	67	58	54	77	75	44	58	62	57	77
1111118	69	50	66	78	72	39	60	58	57	84
1111119	70	56	62	80	71	52	60	63	54	70
1111120	51	52	46	82	74	42	66	63	55	73

separate()

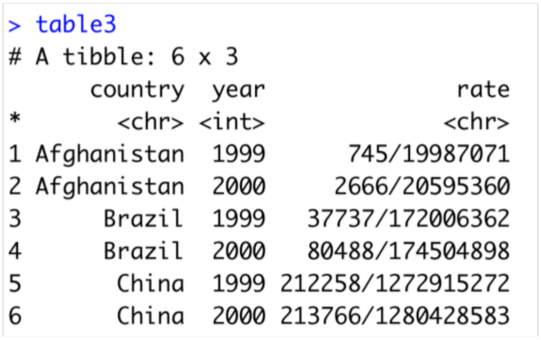
- Separate pulls apart one column into multiple columns
- It splits the information based on finding a non-alphanumeric character
- Separator can be defined (sep="/")
- A converter can find best type for the result, if needed.

Example using tidyr::table3

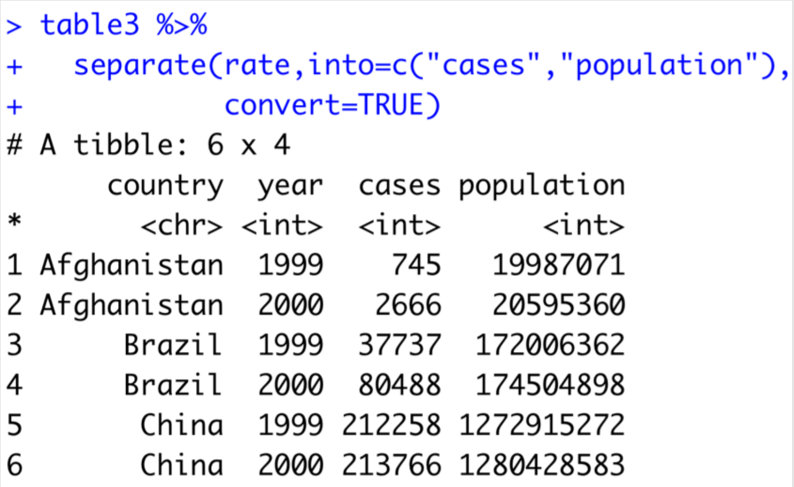


```
Function:       separate(data, col, into, sep = " ", remove = TRUE, convert = FALSE)
Same as:        data %>% separate(col, into, sep = " ", remove = TRUE, convert = FALSE)

Arguments:
        data:           data frame
        col:            column name representing current variable
        into:           names of variables representing new variables
        sep:            how to separate current variable (char, num, or symbol)
        remove:         if TRUE, remove input column from output data frame
        convert:        if TRUE will automatically convert values to logical, integer, numeric, complex or
                        factor as appropriate
```



```
> table3
# A tibble: 6 x 3
      country  year                 rate
*       <chr> <int>                <chr>
1 Afghanistan  1999         745/19987071
2 Afghanistan  2000        2666/20595360
3      Brazil  1999      37737/172006362
4      Brazil  2000      80488/174504898
5       China  1999   212258/1272915272
6       China  2000   213766/1280428583
```



```
> table3 %>%
+   separate(rate,into=c("cases","population"),
+            convert=TRUE)
# A tibble: 6 x 4
      country  year  cases population
*       <chr> <int>  <int>      <int>
1 Afghanistan  1999    745   19987071
2 Afghanistan  2000   2666   20595360
3      Brazil  1999  37737  172006362
4      Brazil  2000  80488  174504898
5       China  1999 212258 1272915272
6       China  2000 213766 1280428583
```

spread() function

- Spreading is the opposite of gathering
- Useful when observations are scattered across multiple rows

```
untidy <- spread(tidy,Treatment,Outcome)
untidy
```

```
## # A tibble: 3 x 3
##   name         treatmenta treatmentb
##   <chr>             <dbl>      <dbl>
## 1 Jane Doe             16         11
## 2 John Smith           NA          2
## 3 Mary Johnson          3          1
```

unite()

- The inverse of **separate()**
- Combines multiple columns into a single column
- Can use this to revert the transformed table3 back to its original

```
table3new <- separate(table3, rate,
                      into=c("cases","population"),
                      convert=T)
table3new
```

```
## # A tibble: 6 x 4
##   country      year  cases population
##   <chr>       <int>  <int>      <int>
## 1 Afghanistan  1999    745   19987071
## 2 Afghanistan  2000   2666   20595360
## 3 Brazil       1999  37737  172006362
## 4 Brazil       2000  80488  174504898
## 5 China        1999 212258 1272915272
```

unite() - sample code

```
unite(table3new,"rate", c("cases","population"),
      sep = "/")
```

```
## # A tibble: 6 x 3
##   country      year rate
##   <chr>       <int> <chr>
## 1 Afghanistan  1999 745/19987071
## 2 Afghanistan  2000 2666/20595360
## 3 Brazil       1999 37737/172006362
## 4 Brazil       2000 80488/174504898
## 5 China        1999 212258/1272915272
## 6 China        2000 213766/1280428583
```

Summary

- Tidy Data
    - every row is an obervations
    - Every column a variable
- **tidyr** provides tools to reshape data
- **dplyr** and **ggplot2** operate on tidy data