Introduction

library function loads the tidyr package into the global environment. The *data* function has the effects of loading **who** dataset from package tidyr into the global environment

In [17]:
```
library(tidyr)
data(who, package = "tidyr")
```

---

1. Gathering columns

The *pivot_longer* has the effect of gathering all the columns specified by **cols** keyword parameter under column name specified as **key** in the **names_to** keyword parameter.

In [4]:
```
#1. (2 marks) Gather together all the columns from _new_spm014 to _newrelf65
who1 <- pivot_longer(who,
             cols = c("new_sp_m014":"newrel_f65"),
             names_to = "key",
             values_to = "cases", values_drop_na = TRUE)
```

In [5]:
```
#view the first six rows to see the effect of function pivot_longer
head(who1)
```

A tibble: 6 × 6

country	iso2	iso3	year	key	cases
<chr>	<chr>	<chr>	<int>	<chr>	<int>
Afghanistan	AF	AFG	1997	new_sp_m014	0
Afghanistan	AF	AFG	1997	new_sp_m1524	10
Afghanistan	AF	AFG	1997	new_sp_m2534	6
Afghanistan	AF	AFG	1997	new_sp_m3544	3
Afghanistan	AF	AFG	1997	new_sp_m4554	5
Afghanistan	AF	AFG	1997	new_sp_m5564	2

---

2. Make variable names consistent

Load stringr and dplyr packages using function *library*. Function *str_replace* from stringr package takes three arguments, the first being the string to be manipulated; the pattern to look for and thirdly the replacement string.

In [7]:
```
#2. (2 marks) Make variable names consistent
library(stringr)
library(dplyr)
who2 <- who1 %>% mutate(key = str_replace(string = key,
                                          pattern = "newrel_",
                                          replacement = "new_rel_"))
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

In [8]:
```
#view the last six rows to see the result of str_replace
tail(who2)
```

A tibble: 6 × 6

country	iso2	iso3	year	key	cases
<chr>	<chr>	<chr>	<int>	<chr>	<int>
Zimbabwe	ZW	ZWE	2013	new_rel_f1524	2069
Zimbabwe	ZW	ZWE	2013	new_rel_f2534	4649
Zimbabwe	ZW	ZWE	2013	new_rel_f3544	3526
Zimbabwe	ZW	ZWE	2013	new_rel_f4554	1453
Zimbabwe	ZW	ZWE	2013	new_rel_f5564	811
Zimbabwe	ZW	ZWE	2013	new_rel_f65	725

---

3. Run the following code

In [10]:
```
#3. (2 mark) Run the following code
who3 <- who2 %>% separate(key, into = c("new", "type", "sexage"), sep = "_")
head(who3)
```

A tibble: 6 × 8

country	iso2	iso3	year	new	type	sexage	cases
<chr>	<chr>	<chr>	<int>	<chr>	<chr>	<chr>	<int>
Afghanistan	AF	AFG	1997	new	sp	m014	0
Afghanistan	AF	AFG	1997	new	sp	m1524	10
Afghanistan	AF	AFG	1997	new	sp	m2534	6
Afghanistan	AF	AFG	1997	new	sp	m3544	3
Afghanistan	AF	AFG	1997	new	sp	m4554	5
Afghanistan	AF	AFG	1997	new	sp	m5564	2

Purpose of using %>%

%>% is called the pipe operator. R being a functional language, it means that code will often have a lot of ( ) as the result of one function is being input into another function. This eventually results into code that is hard to read. The %>% operator simplifies code by allowing the result of one function to be piped into another. As a result, code is much more readable.

---

4. Separate sexage into sex and age

separate function

The *separate* function separates a data frame into multiple columns.

In [14]:
```
# 4. (1 mark) Separate sexage into sex and age: Use the function s --------

who4 <- who3 %>% separate(sexage, into = c("sex", "age"), sep = 1)
tail(who4)
```

A tibble: 6 × 9

country	iso2	iso3	year	new	type	sex	age	cases
<chr>	<chr>	<chr>	<int>	<chr>	<chr>	<chr>	<chr>	<int>
Zimbabwe	ZW	ZWE	2013	new	rel	f	1524	2069
Zimbabwe	ZW	ZWE	2013	new	rel	f	2534	4649
Zimbabwe	ZW	ZWE	2013	new	rel	f	3544	3526
Zimbabwe	ZW	ZWE	2013	new	rel	f	4554	1453
Zimbabwe	ZW	ZWE	2013	new	rel	f	5564	811
Zimbabwe	ZW	ZWE	2013	new	rel	f	65	725

---

5. Print the first 5 rows and the last 5 rows

head and *tail* functions

The head and the tail function can be used to print the first five and the last five rows respectively by providing the dataset as the first argument and the number of rows that you intend to print as the second argument.

In [9]:
```
#first 5
head(who4, 5)
```

A tibble: 5 × 9

country	iso2	iso3	year	new	type	sex	age	cases
<chr>	<chr>	<chr>	<int>	<chr>	<chr>	<chr>	<chr>	<int>
Afghanistan	AF	AFG	1997	new	sp	m	014	0
Afghanistan	AF	AFG	1997	new	sp	m	1524	10
Afghanistan	AF	AFG	1997	new	sp	m	2534	6
Afghanistan	AF	AFG	1997	new	sp	m	3544	3
Afghanistan	AF	AFG	1997	new	sp	m	4554	5

In [10]:
```
#last 5
tail(who4, 5)
```

A tibble: 5 × 9

country	iso2	iso3	year	new	type	sex	age	cases
<chr>	<chr>	<chr>	<int>	<chr>	<chr>	<chr>	<chr>	<int>
Zimbabwe	ZW	ZWE	2013	new	rel	f	2534	4649
Zimbabwe	ZW	ZWE	2013	new	rel	f	3544	3526
Zimbabwe	ZW	ZWE	2013	new	rel	f	4554	1453
Zimbabwe	ZW	ZWE	2013	new	rel	f	5564	811
Zimbabwe	ZW	ZWE	2013	new	rel	f	65	725

---

6. Export who4 as an csv file and save it in your local directory.

write.csv function

The *write.csv* function takes the dataframe as the first argument and path you want to save your file to as the second argument. In our case, we have specified the file "who4.csv" to be saved in the Documents folder.

In [15]:
```
path = "C:\\Users\\joshua\\Documents\\who4.csv"
write.csv(who4, path)
```