pipe and tidyr

Niels Aka

%>%

Example 1

```
x <- rnorm(1000)
mean(x)       # standard
```

```
## [1] 0.01612787
```

```
library(magrittr)
x %>% mean() # piped
```

```
## [1] 0.01612787
```

Example 1 - add arguments

```
mean(x, trim = 0.2)
```

```
## [1] 0.01455955
```

```
x %>% mean(trim = 0.2)
```

```
## [1] 0.01455955
```

Example 2 - Chain of functions

```
sprintf("%.15f", sqrt(var(x)))
```

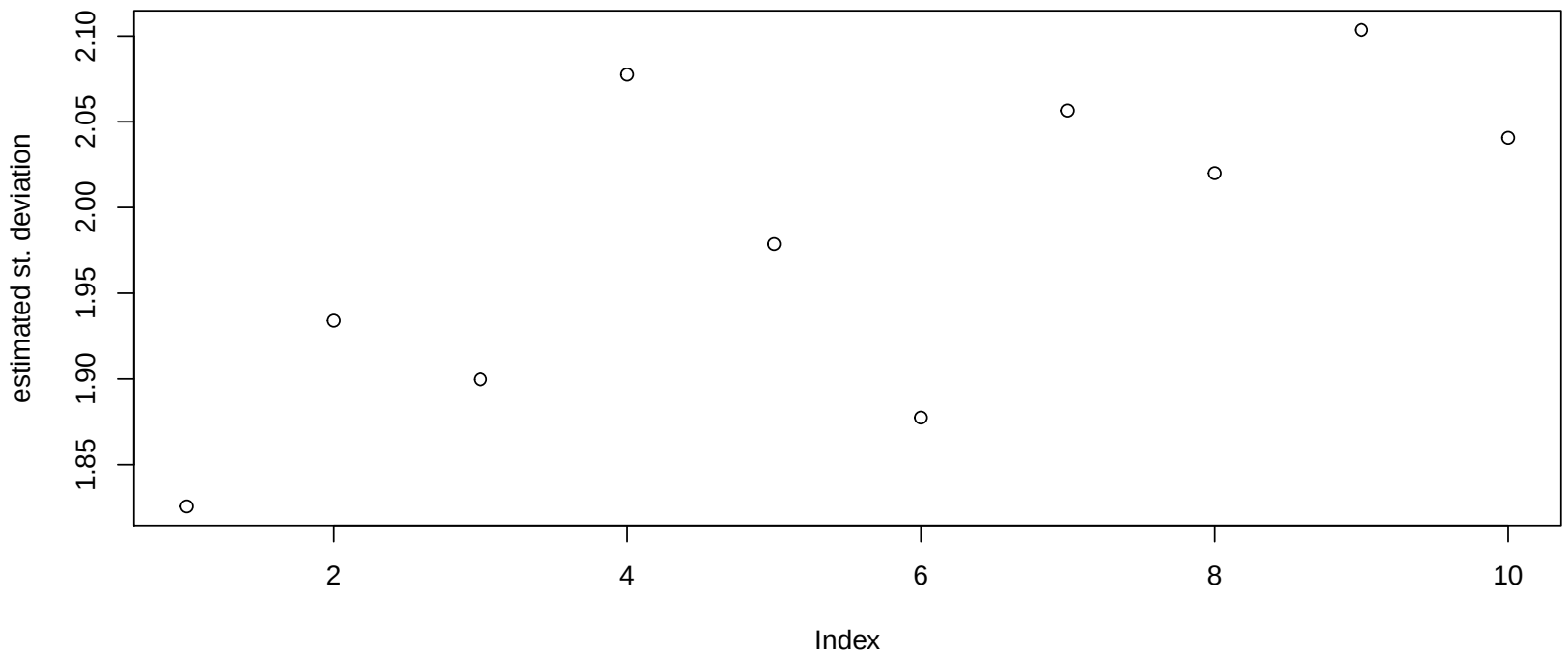
```
## [1] "0.991694976399348"
```

```
x %>% var() %>% sqrt() %>% sprintf("%.15f", .)
```

```
## [1] "0.991694976399348"
```

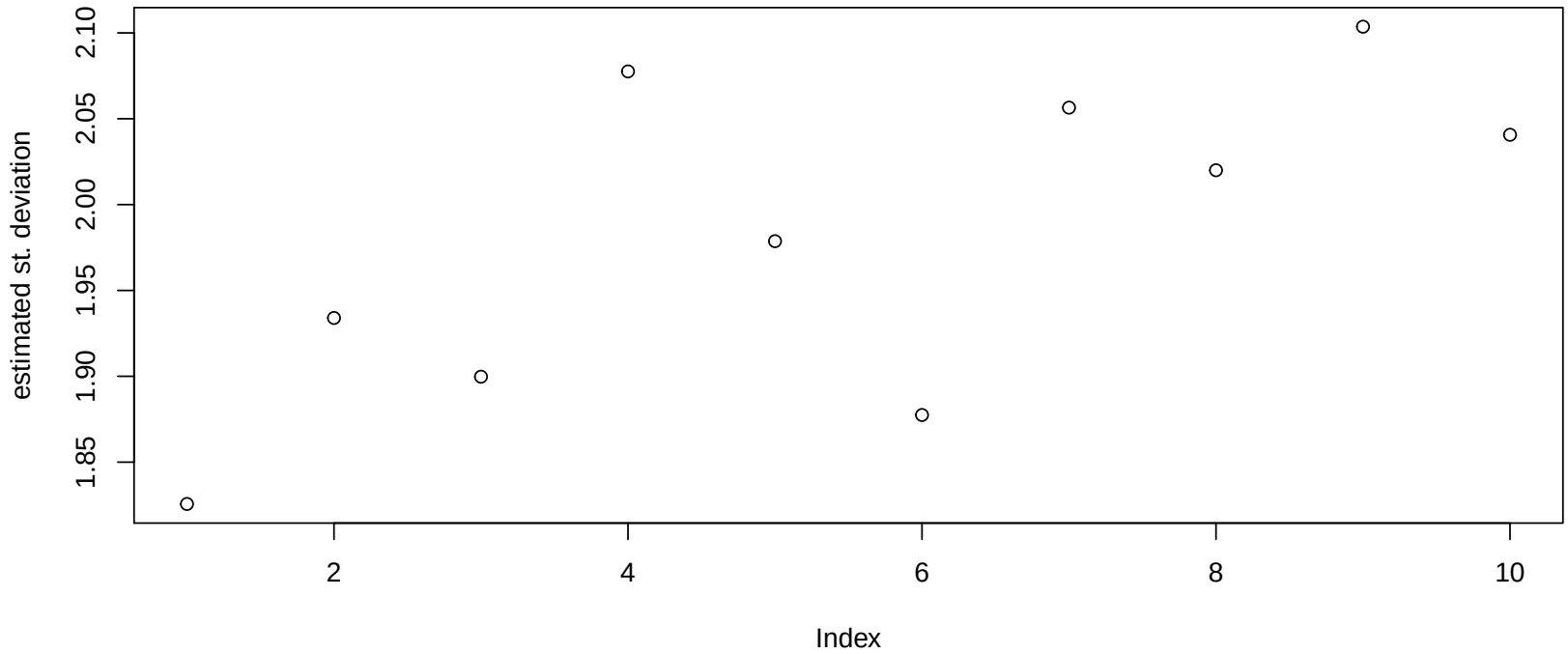
Example 3 - Read inside out

```
plot(apply(matrix(rnorm(1000, sd = 2),
                  ncol = 10),
           MARGIN = 2,
           FUN = function(x) sqrt(var(x))),
     ylab = "estimated st. deviation")
```



Example 3 - Read top to bottom, left to right

```
rnorm(1000, sd = 2) %>%
  matrix(ncol = 10) %>%
  apply(MARGIN = 2, FUN = . %>% var() %>% sqrt()) %>%
  plot(ylab = "estimated st. deviation")
```



Example 4 - sequence of objects

```
mtcars_sub       <- subset(mtcars, hp > 120 & cyl == 6)
mtcars_sub_trans <- transform(mtcars_sub,
                              kpl = mpg * 0.4251,
                              kw = hp * 0.7457)
mtcars_sub_trans_select <-
  mtcars_sub_trans[, c("cyl", "kpl", "kw")]
mtcars_sub_trans_select
```

```
##                 cyl      kpl       kw
## Merc 280          6 8.16192  91.7211
## Merc 280C         6 7.56678  91.7211
## Ferrari Dino      6 8.37447 130.4975
```

Example 4 - piped

```
mtcars %>%
  subset(hp > 120 & cyl == 6) %>%
  transform(kpl = mpg * 0.4251,
            kw  = hp  * 0.7457) %>%
  `[`(c("cyl", "kpl", "kw"))
```

```
##                cyl      kpl       kw
## Merc 280         6 8.16192  91.7211
## Merc 280C        6 7.56678  91.7211
## Ferrari Dino     6 8.37447 130.4975
```

Summary

- read top to bottom
- avoid unnecessary naming
- relevant info stands out

The pipe is particularly useful when operating repeatedly on a single object, e.g. a `data.frame`. Maybe less useful for actual programming.

In RStudio, insert with ctrl/strg + shift + m.

Best practices

- Pipe should have several lines.
- Each line one comprehensible piece of code.
- Not too long.

Debug: either rely on error message or grow pipe piece by piece until error is thrown.

tidyr

tidyr principles

Use standardized format

- each observation in one row
- each variable in one column

Other packages (dplyr etc.) will rely on this format.

Before starting, think about what is a variable in your analysis and what isn't.

Tuberculosis Report by the WHO. Tidy?

```
## # A tibble: 7,190 × 18
##    country  iso2  iso3  year  m014 m1524 m2534 m3544 m4554 m5564
##      <chr> <chr> <chr> <int> <int> <int> <int> <int> <int> <int>
## 1  Albania    AL   ALB  1996    NA    NA    NA    NA    NA    NA
## 2  Albania    AL   ALB  1997     0    23    43    33    25    21
## 3  Albania    AL   ALB  1998     1    17    21    24    18    26
## 4  Albania    AL   ALB  1999     0    13    23    25    19    15
## 5  Albania    AL   ALB  2000     2    19    21    14    24    19
## 6  Albania    AL   ALB  2001     3    13    18    17    19    20
## 7  Albania    AL   ALB  2002     0    21    27    29    19    23
## 8  Albania    AL   ALB  2003     0    28    19    32    16    22
## 9  Albania    AL   ALB  2004     5    12    19    21    24    23
## 10 Albania    AL   ALB  2005     0    26    21    16    31    20
## # ... with 7,180 more rows, and 8 more variables: m6599 <int>,
## #   f014 <int>, f1524 <int>, f2534 <int>, f3544 <int>, f4554 <int>,
## #   f5564 <int>, f6599 <int>
```

Hint: m = male; f = female; 014 = age 0 to 14. Cell entries are the number of tuberculosis cases.

tidyr functions

tidyr provides functions to reorganise `data.frames` by changing the number of columns.

Change wide to long by gather()ing multiple columns into fewer ones.

Change long to wide by spread()ing one column out across many.

Break one column into multiple with separate() or reverse with unite().

Make it tidy

```
who %>%
  gather(key = gender_age, value = count, -(country:year))
```

```
## # A tibble: 100,660 × 6
##    country  iso2  iso3  year gender_age count
##      <chr> <chr> <chr> <int>      <chr> <int>
## 1  Albania    AL   ALB  1996       m014    NA
## 2  Albania    AL   ALB  1997       m014     0
## 3  Albania    AL   ALB  1998       m014     1
## 4  Albania    AL   ALB  1999       m014     0
## 5  Albania    AL   ALB  2000       m014     2
## 6  Albania    AL   ALB  2001       m014     3
## 7  Albania    AL   ALB  2002       m014     0
## 8  Albania    AL   ALB  2003       m014     0
## 9  Albania    AL   ALB  2004       m014     5
## 10 Albania    AL   ALB  2005       m014     0
## # ... with 100,650 more rows
```

Make it tidy

```
who %>%
  gather(key = gender_age, value = count, -(country:year)) %>%
  separate(gender_age, into = c("gender", "cohort"), sep = 1)
```

```
## # A tibble: 100,660 × 7
##    country  iso2  iso3  year gender cohort count
## *    <chr> <chr> <chr> <int>  <chr>  <chr> <int>
## 1  Albania    AL   ALB  1996      m    014    NA
## 2  Albania    AL   ALB  1997      m    014     0
## 3  Albania    AL   ALB  1998      m    014     1
## 4  Albania    AL   ALB  1999      m    014     0
## 5  Albania    AL   ALB  2000      m    014     2
## 6  Albania    AL   ALB  2001      m    014     3
## 7  Albania    AL   ALB  2002      m    014     0
## 8  Albania    AL   ALB  2003      m    014     0
## 9  Albania    AL   ALB  2004      m    014     5
## 10 Albania    AL   ALB  2005      m    014     0
## # ... with 100,650 more rows
```

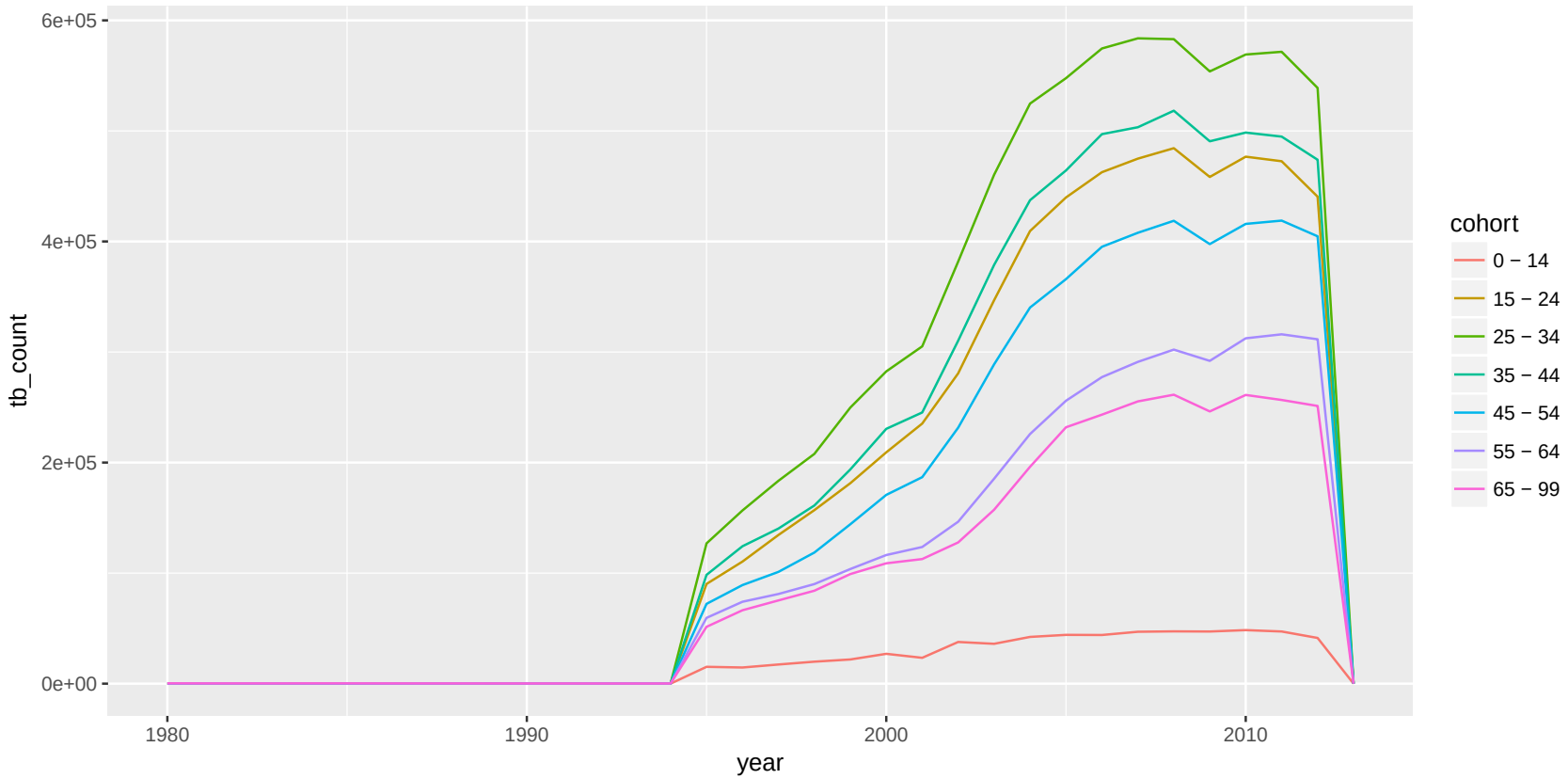
Mutate cohort for better readability

```
library(dplyr)
who %>%
  gather(key = gender_age, value = count, -(country:year)) %>%
  separate(gender_age, into = c("gender", "cohort"), sep = 1) %>%
  mutate(cohort = gsub("^(.+)(.{2})$", "\\1 - \\2", cohort))
```

```
## # A tibble: 100,660 × 7
##    country  iso2  iso3  year gender cohort count
##      <chr> <chr> <chr> <int>  <chr>  <chr> <int>
## 1  Albania    AL   ALB  1996      m 0 - 14    NA
## 2  Albania    AL   ALB  1997      m 0 - 14     0
## 3  Albania    AL   ALB  1998      m 0 - 14     1
## 4  Albania    AL   ALB  1999      m 0 - 14     0
## 5  Albania    AL   ALB  2000      m 0 - 14     2
## 6  Albania    AL   ALB  2001      m 0 - 14     3
## 7  Albania    AL   ALB  2002      m 0 - 14     0
## 8  Albania    AL   ALB  2003      m 0 - 14     0
## 9  Albania    AL   ALB  2004      m 0 - 14     5
## 10 Albania    AL   ALB  2005      m 0 - 14     0
## # ... with 100,650 more rows
```

Easy to plot

```
library(ggplot2)
library(dplyr)
who %>%
  gather(key = gender_age, value = count, -(country:year)) %>%
  separate(gender_age, into = c("gender", "cohort"), sep = 1) %>%
  mutate(cohort = gsub("^(.+)(.{2})$", "\\1 - \\2", cohort)) %>%
  group_by(year, cohort) %>%
  summarise(tb_count = sum(count, na.rm = TRUE)) %>%
  ggplot(aes(x = year, y = tb_count, colour = cohort)) + geom_line()
```



Remark

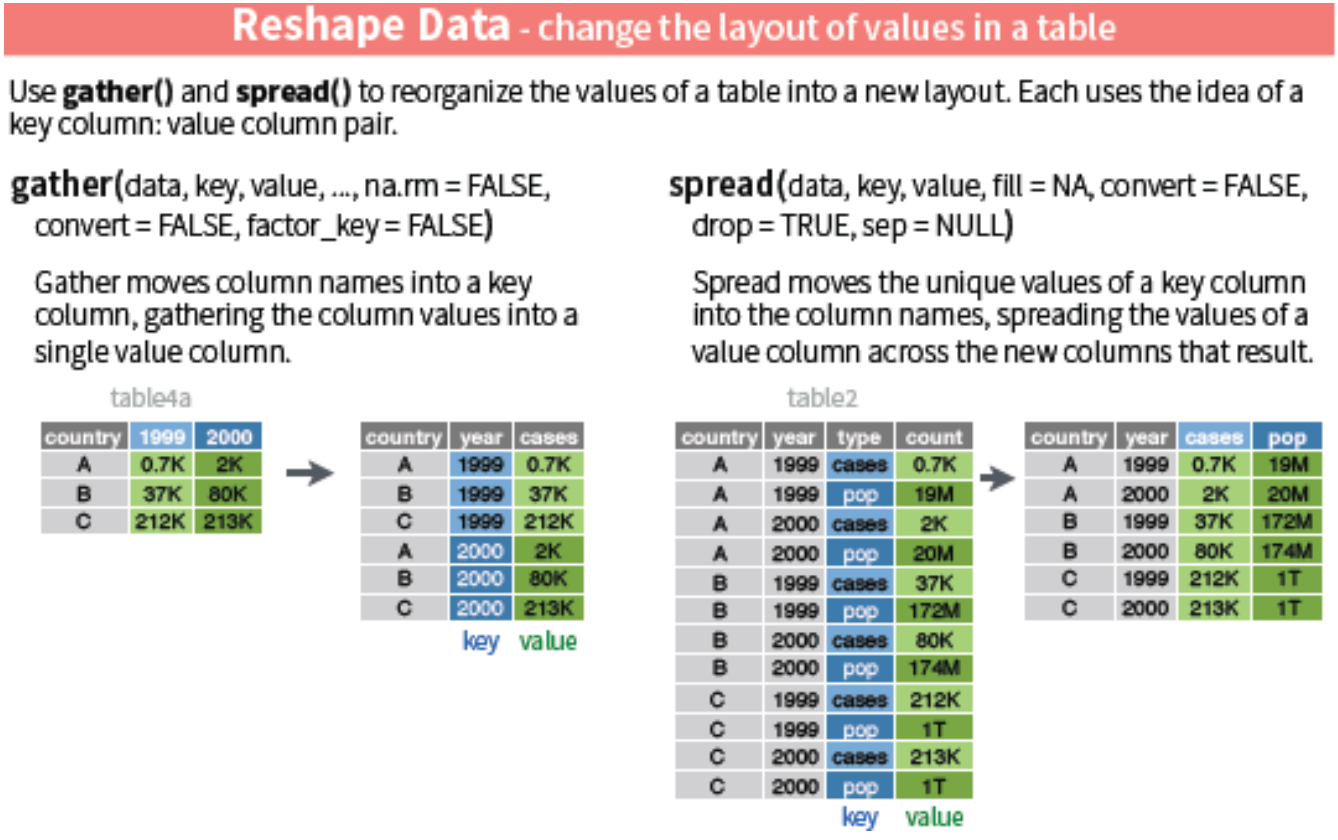
Here, `cohort` is a variable. In other instances, we might be more interested in analysing tuberculosis counts in relation to the maximum or minimum age.

```
who %>%
  gather(key = gender_age, value = count, -(country:year)) %>%
  separate(gender_age, into = c("gender", "cohort"), sep = 1) %>%
  separate(cohort, into = c("min_age", "max_age"), sep = -3)
```

```
## # A tibble: 100,660 × 8
##    country  iso2  iso3  year gender min_age max_age count
## *    <chr> <chr> <chr> <int>  <chr>   <chr>   <chr> <int>
## 1  Albania    AL   ALB  1996      m       0      14    NA
## 2  Albania    AL   ALB  1997      m       0      14     0
## 3  Albania    AL   ALB  1998      m       0      14     1
## 4  Albania    AL   ALB  1999      m       0      14     0
## 5  Albania    AL   ALB  2000      m       0      14     2
## 6  Albania    AL   ALB  2001      m       0      14     3
## 7  Albania    AL   ALB  2002      m       0      14     0
## 8  Albania    AL   ALB  2003      m       0      14     0
## 9  Albania    AL   ALB  2004      m       0      14     5
## 10 Albania    AL   ALB  2005      m       0      14     0
## # ... with 100,650 more rows
```

What is a variable depends on the situation.

RStudio Cheatsheet



Reshape Data - change the layout of values in a table

Use **gather()** and **spread()** to reorganize the values of a table into a new layout. Each uses the idea of a key column: value column pair.

gather(data, key, value, ..., na.rm = FALSE, convert = FALSE, factor_key = FALSE)

Gather moves column names into a key column, gathering the column values into a single value column.

table4a

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

key value

spread(data, key, value, fill = NA, convert = FALSE, drop = TRUE, sep = NULL)

Spread moves the unique values of a key column into the column names, spreading the values of a value column across the new columns that result.

table2

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

key value

→

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

Found here or in
Rstudio: Help > Cheatsheets > Data Manipulation with dplyr, tidyr.

For frequent use, create a hotkey (e.g. ctrl/strg + alt + d) to open this.