`tidyr` and `stringr`

Silvie Cinková

2025-08-13

Table of contents

1 Libraries

```
library(readr)
library(dplyr)
```

```
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
```

```r
library(tidyr)
library(magrittr)
```

```

Attaching package: 'magrittr'


The following object is masked from 'package:tidyr':

    extract
```

```r
library(gapminder)
library(ggplot2)
library(tidytext)
```

2 Data

```r
billionaires_df <- read_tsv("datasets_ATRIUM/billionaires_combined.tsv",
 ↪  show_col_types = FALSE)
bil_unite <- billionaires_df %>% distinct(name.x, birth_comb) %>%
 ↪  drop_na(birth_comb)
bil_separate <- billionaires_df %>% distinct(name.x, person) %>%
 ↪  drop_na(name.x)
gap_cze_ger_gdp <- gapminder %>%
  filter(country %in% c("Czech Republic", "Germany")) %>%
  select(country, year, gdpPercap)
```

3 tidyr

- lumps and splits column values into new columns

- transforms several variable columns into categories of a new variable and the other way round
- completes observations with **missing values**
- manages **nested columns** (vector/list inside column!)

This library is a “heavy-duty” assistant to `dplyr`. You use it when you need to make your data tidy for your purposes. Remember, tidy means that each row represents an observation and each column represents one variable. Sometimes it is a matter of perspective. Maybe we could say that the data is tidy when you can map all relevant variable to ggplot aesthetic scales.

The original four big verbs of tidyr were two pairs of twins: `unite` and `separate`, and `pivot_longer` (formerly `gather`) and `pivot_wider` (formerly `spread`)

4 `tidyr::unite` (before)

```
bil_unite %>% slice(1:10)
```

```
# A tibble: 10 x 2
   name.x                         birth_comb
   <chr>                               <dbl>
 1 A. Jerrold Perenchio                 1931
 2 Abdulla bin Ahmad Al Ghurair         1955
 3 Abdullah bin Sulaiman Al Rajhi       1929
 4 Abdulsamad Rabiu                     1960
 5 Abhay Soi                            1973
 6 Abhay Vakil                          1952
 7 Abigail Johnson                      1962
 8 Abilio dos Santos Diniz              1937
 9 Achal Bakeri                         1961
10 Acharya Balakrishna                  1972
```

5 `tidyr::unite` (after)

```r
bil_unite %>% slice(1:10) %>%
  unite(col = ID, name.x, birth_comb,                         sep = "***",
↪  remove = FALSE) %>%
  kableExtra::kable()
```

ID	name.x	birth_comb
A. Jerrold Perenchio***1931	A. Jerrold Perenchio	1931
Abdulla bin Ahmad Al Ghurair***1955	Abdulla bin Ahmad Al Ghurair	1955
Abdullah bin Sulaiman Al Rajhi***1929	Abdullah bin Sulaiman Al Rajhi	1929
Abdulsamad Rabiu***1960	Abdulsamad Rabiu	1960
Abhay Soi***1973	Abhay Soi	1973
Abhay Vakil***1952	Abhay Vakil	1952
Abigail Johnson***1962	Abigail Johnson	1962
Abilio dos Santos Diniz***1937	Abilio dos Santos Diniz	1937
Achal Bakeri***1961	Achal Bakeri	1961
Acharya Balakrishna***1972	Acharya Balakrishna	1972

6 tidyr::separate

- splits values into two or more new columns

```r
bil_separate %>% slice(1:2) %>%
  separate(col = name.x, into = c("firstnames", "middlenames", "lastnames" ),
↪  sep = " ", fill = "left")
```

```
# A tibble: 2 x 4
  firstnames middlenames lastnames person
  <chr>      <chr>       <chr>     <chr>
1 A.         Jayson      Adair     a_jayson_adair
2 A.         Jerrold     Perenchio a_jerrold_perenchio
```

```r
set.seed(15)
bil_separate %>% slice_sample(n = 10) %>%
  separate(col = name.x,
           into = c("firstnames",  "lastnames" ),
           sep = " ", fill = "left", remove = FALSE)
```

```
# A tibble: 10 x 4
   name.x                    firstnames lastnames         person
   <chr>                     <chr>      <chr>             <chr>
 1 Xu Xiaoqun                Xu         Xiaoqun           xu_xiaoqun
 2 Irwin Jacobs              Irwin      Jacobs            irwin_jacobs
 3 Wang Chaobin              Wang       Chaobin           wang_chaobin
 4 Anton Schlecker           Anton      Schlecker         anton_schlecker
 5 Thomas James              Thomas     James             thomas_james
 6 GSK Velu                  GSK        Velu              gsk_velu
 7 Charoen Sirivadhanabhakdi Charoen    Sirivadhanabhakdi charoen_sirivadhanabh~
 8 Yuriy Kosiuk              Yuriy      Kosiuk            yuriy_kosiuk
 9 Elaine Marshall           Elaine     Marshall          elaine_marshall
10 Jaime Botin               Jaime      Botin             jaime_botin_1
```

In some functions in `tidyr`, but also in `dplyr`, you will benefit of a good command of string operations. That is, when you can find general patterns in strings, mainly with Regular Expressions. When your data is structured very well, it can be easy; such as separate first names from surnames, when they are all in the same order and separated by comma. Most often this is not the case, but you can at least approximate a good result with a more advanced pattern. The current version of `tidyr` even has a separate function that works with regular expressions, but I was not able to get with it smarter results than with the ordinary one that only works with the delimiter.

At any rate, R has a wonderful library for work with strings: `stringr`. I will include `stringr` functions wherever relevant.

7 `tidyr::pivot_wider`

```
gap_czger_gdp_wide <- gap_cze_ger_gdp %>%
  pivot_wider(names_from = country, values_from = gdpPercap)
gap_czger_gdp_wide %>% slice(1:3) %>% kableExtra::kable()
```

year	Czech Republic	Germany
1952	6876.140	7144.114
1957	8256.344	10187.827
1962	10136.867	12902.463

```
gap_czger_gdp_wide %$%
  cor(x = `Czech Republic`, y = Germany, method = "pearson")
```

```
[1] 0.946459
```

Imagine you want to compute a correlation of the temporal development between two countries, considering an indicator, such as GDP per capita. To compute the correlation, you must have the data in two separate variables. This is how you would prepare `gapminder` to correlate GDP per capita 1952 - 2007 between Czechia and Germany.

Format changes in data frames: wider = fewer rows and more columns, longer = (sometimes) fewer columns but definitely more rows, both compared to the state before the manipulation.

8 `tidyr::pivot_longer`

```
gap_czger_gdp_wide %>% slice(1:10) %>%
  pivot_longer(cols = c(`Czech Republic`, Germany), names_to = "COUNTRY",
↪  values_to = "GDPperCap")
```

```
# A tibble: 20 x 3
    year COUNTRY        GDPperCap
   <int> <chr>              <dbl>
 1  1952 Czech Republic     6876.
 2  1952 Germany            7144.
 3  1957 Czech Republic     8256.
 4  1957 Germany           10188.
 5  1962 Czech Republic    10137.
 6  1962 Germany           12902.
 7  1967 Czech Republic    11399.
 8  1967 Germany           14746.
 9  1972 Czech Republic    13108.
10  1972 Germany           18016.
11  1977 Czech Republic    14800.
12  1977 Germany           20513.
13  1982 Czech Republic    15377.
14  1982 Germany           22032.
15  1987 Czech Republic    16310.
16  1987 Germany           24639.
17  1992 Czech Republic    14297.
18  1992 Germany           26505.
19  1997 Czech Republic    16049.
20  1997 Germany           27789.
```

… but you definitely prefer the GDP in one column and countries in the other column when you want to plot the development comparison. Wider tables are often considered to be more human-readable, and therefore you typically get them from sources that were primarily designed for print.

9 Separating into rows

```
industry_terms <- billionaires_df %>%
  select(c(person, time, countries, industry, income_groups, world_6region))
↪  %>%
  separate_longer_delim(cols = c("industry"), delim = ";")
set.seed(33)
industry_terms %>% slice_sample(n = 10)
```

```
# A tibble: 10 x 6
   person           time countries industry            income_groups world_6region
   <chr>           <dbl> <chr>     <chr>               <chr>         <chr>
 1 yang_huiyan      2022 chn       "Real Estate, Edu~ upper_middle~ east_asia_pa~
 2 horst_wortmann   2016 deu       "Apparel & Textil~ high_income   europe_centr~
 3 wang_wenxue      2018 chn       "Real Estate"      upper_middle~ east_asia_pa~
 4 b_wayne_hughes   2005 usa       "Service"          high_income   america
 5 daniela_herz     2013 deu       "Investments"      high_income   europe_centr~
 6 edward_lampert   2017 usa       " Finance"         high_income   america
 7 xu_jingren       2020 chn       "Healthcare"       upper_middle~ east_asia_pa~
 8 qiu_guanghe      2021 chn       "Apparel"          upper_middle~ east_asia_pa~
 9 elon_musk        2019 usa       "Automotive"       high_income   america
10 ennio_doris      2009 ita       " Finance"         high_income   europe_centr~
```

10 continuation

```
industry_terms %<>%
  separate_longer_delim(cols = c("industry"), delim = "&") %>%
  separate_longer_delim(cols = c("industry"), delim = "and") %>%
  separate_longer_delim(cols = c("industry"), delim = ",")
set.seed(10)
industry_terms %>% slice_sample(n = 10)
```

```
# A tibble: 10 x 6
```

```
   person                  time countries industry   income_groups world_6region
   <chr>                  <dbl> <chr>     <chr>      <chr>         <chr>
 1 christy_walton          2013 usa       "Fashion " high_income   america
 2 ty_warner               2004 usa       "Hospital~ high_income   america
 3 mustafa_rahmi_koc       2018 tur       " Gas"     upper_middle~ europe_centr~
 4 oleg_deripaska          2008 rus       "Metals "  upper_middle~ europe_centr~
 5 doris_fisher            2021 usa       " Fashion~ high_income   america
 6 gerald_ford             2017 usa       "Finance " high_income   america
 7 gregorio_perez_companc  2005 arg       " Food"    upper_middle~ america
 8 robert_bass             2012 usa       "Energy"   high_income   america
 9 norma_lerner            2018 usa       "Financia~ high_income   america
10 yuri_kovalchuk          2018 rus       "Finance " upper_middle~ europe_centr~
```

```r
industry_tf_idf <- industry_terms %>% filter(nchar(industry) > 1) %>%
  group_by(industry, countries, world_6region) %>%
  count(name = "freq") %>% ungroup() %>%
  tidytext::bind_tf_idf(term = industry,
                        document = countries,
                        n = freq) %>%
  group_by(countries, world_6region) %>%
  slice_max(order_by = tf_idf, n = 3) %>% ungroup()
```

```r
library(dplyr)
library(tidygraph)
```

```

Attaching package: 'tidygraph'

The following object is masked from 'package:stats':

    filter
```

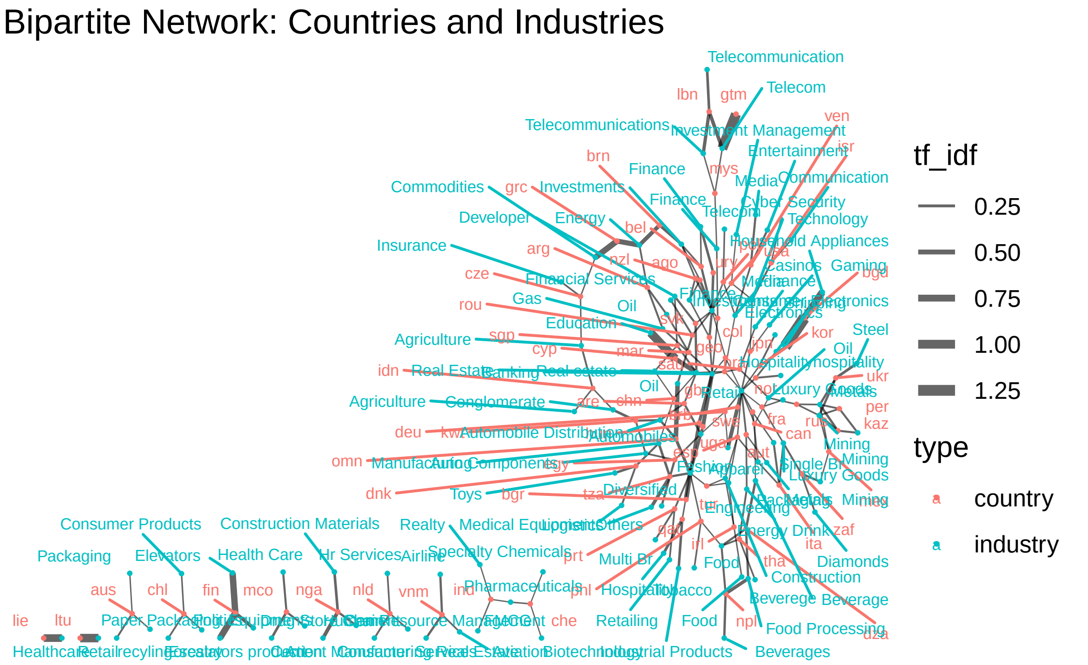
```r
library(ggraph)

# Prepare edge list
edges <- industry_tf_idf %>%
  select(countries, industry, tf_idf)

# Create node list
nodes <- tibble(name = unique(c(edges$countries, edges$industry))) %>%
  mutate(type = if_else(name %in% edges$countries, "country", "industry"))

# Create graph object
```

```r
graph <- tbl_graph(nodes = nodes, edges = edges, directed = FALSE)
graph
```

```
# A tbl_graph: 176 nodes and 204 edges
#
# A bipartite simple graph with 11 components
#
# Node Data: 176 x 2 (active)
   name  type
   <chr> <chr>
 1 ago   country
 2 are   country
 3 arg   country
 4 aus   country
 5 aut   country
 6 bel   country
 7 bgd   country
 8 bgr   country
 9 bra   country
10 brb   country
# i 166 more rows
#
# Edge Data: 204 x 3
   from    to tf_idf
  <int> <int>  <dbl>
1     1    74 0.168
2     1    75 0.115
3     1    76 0.0968
# i 201 more rows
```

```r
# Plot with ggraph
ggraph(graph, layout = "auto") +
  geom_edge_link(aes(edge_width = tf_idf), alpha = 0.6) +
  geom_node_point(aes(color = type), size = 0.3) +
  geom_node_text(aes(label = name, color = type), repel = TRUE, size = 2,
↪   max.overlaps = 100) +
  scale_edge_width(range = c(0.2, 2)) +
  theme_void() +
  labs(title = "Bipartite Network: Countries and Industries")
```

```
Using "stress" as default layout
```



```
ggsave(filename = "my_output_files/billionairs_industry_country_ggraph.pdf",
↪  width = 7 * 2, height = 7 * 2.2)
```

```
Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
conversion failure on 'Paper Packaging ' in 'mbcsToSbcs': for   (U+3001)
```