Tidyverse を活用したデータ前処理の実践：初級編
– mJOHNSNOW R 解析 Peer 勉強会 vol.4 –

森　和貴　（静岡市立清水病院）

目次

1. はじめに

政府の統計総合窓口 e-Stat で公開されている厚生労働省 人口動態統計 より、2023 年の死因別死亡数のデータを使用して、悪性腫瘍の臓器別死亡数の変遷のグラフを作成します。

1-1. データクリーニングの方針

「見る」あるいは「見せる」ための表や雑然とした表を、コンピューターで認識できて（機械判読可能）解析に使用できる表に変形・変換する方針を考えます。100% 従えなくとも大まかな目標としていくつか代表的なルールを示します。

また、**元のファイルは極力編集せず編集の内容や経過はコードに残す**ようにします。

1-1-1. 総務省「統計表における機械判読可能なデータの表記方法の統一ルール」

2020 年に政府統計の総合窓口（e-Stat）に掲載する統計表におけるデータ表記方法の統一ルールとして総務省が策定した、「統計表における機械判読可能なデータの表記方法の統一ルールの策定」では、機械判読可能であることを重視して以下のようなルールが示されています。

Excel など特定のソフトや政府統計に依存しない部分を抜粋すると、

- **1 セル 1 データ**となっている
- **数値データは数値属性**とし、文字列（注：単位、注釈など）を含まない

- スペースや改行等で体裁を整えていない
- **項目名等を省略していない**（「薬剤 A」「B」「C」は「薬剤 A」「薬剤 B」「薬剤 C」とする）
- データが分断されていない、1 シートに複数の表が掲載されていない

1-1-2. Tidy data（整然データ）: Wickham H. 2014

`tidyverse` の作者である Hadley Wickham 氏が提唱 (Journal of Statistical Software. 59: 1-23, 2014) した概念で、「データの構造 (structure)」と「意味 (semantic)」を一致させることを目指しています。

1. 個々の変数 (variable) が 1 つの列 (column) をなす。
2. 個々の観測 (observation) が 1 つの行 (row) をなす。
3. 個々の観測の構成単位の類型 (type of observational unit) が 1 つの表 (table) をなす。
4. 個々の値 (value) が 1 つのセル (cell) をなす

Tidyverse はこの思想に基づいて作成されており、これらのパッケージをフルに活用するためには tidy data を意識することが必要です。

1-2. コーディングスタイル

R に限らず、プログラムのコードは読みやすさやメンテナンス性を確保するため一定のルールに沿って記載することが勧められています。R ではいくつかの有名なコーディングスタイルのガイドがあります。

- Google's R Style Guide
- Tidyverse style guide

カッコ類の使い方、1 行の文字数、使用するパイプ（magrittr pipe (`%>%`) vs. base pipe (`|>`)）、変数などの命名規則 (下記) などが記載されています。全部従うのはなかなか大変ですが、変数名のルールなどは一貫性を持って付けた方が良いです。

命名規則	説明	採用例
`snake_case`	“_” で小文字の単語をつなぐ	`Tidyverse style guide`
`camelCase`	最初の単語は小文字、2 語目以降は単語のはじめを大文字にしてつなぐ	
`PascalCase (BigCamelCase)`	すべての単語のはじめを大文字にしてつなぐ	`Google's R Style Guide`
`dot.case`	“.” で小文字の単語をつなぐ	(R の関数等でよく使われる)
`kebab-case`	“-” で小文字の単語をつなぐ	(R の文法上使えない)

今回は Tidyverse style guide でも使われており R 界隈で使用する人が多い（私見）、小文字の単語を _ でつなぐ `snake_case` と mJOHNSNOW の講義でこれまでも使われている magrittr pipe (`%>%`) を使用します。
ちなみに、Tidyverse style guide では base pipe (`|>`) の機能が充実してきたことから現在は base pipe を使用するように推奨されています（筆者も普段は base pipe 派です）。

また、外部関数の名前空間 (R の標準パッケージ以外のパッケージ名) を `package::function()` で明示することを基本とします（ただしパイプ演算子など一部例外あり）。

2. サンプルデータ

厚生労働省 人口動態統計 より、2023 年の死因別死亡数のデータを使用します。

https://www.e-stat.go.jp/stat-search/files?tclass=000001041646&cycle=7&year=20230

より、「5-13 死因（死因簡単分類）別にみた性・年次別死亡数及び死亡率（人口 10 万対）/ 2023 年」（公開日 2024-09-17）の CSV ファイル（mc130000.csv）を Working Directory にダウンロードして下さい。

> **i** e-Stat API
>
> 今回はファイルから読み込んでデータクリーニングを行う練習として CSV ファイルをダウンロードして解析しますが、データベースへの収録が進んでいる過去年度のものは e-Stat API を用いてソフトウェア的に取得することもできます。興味のある方は `estatapi` パッケージを調べてみて下さい。該当の `statsDataId` は “0003411657” です。

2-1. ファイルの構造

いきなり R で触る前に、まずは Excel などで開いてファイルの中身を俯瞰します。

- 1-13 行目：ファイルの説明
- 14-15 行目：表の見出し。14 行目の「死亡数」と「死亡率」の下に各年が並ぶ
- 16 行目〜：死因別の死亡数・死亡率
  - 16-156 行目：総数
  - 157-297 行目：男性
  - 298-438 行目：女性

内容としても、整形のためのスペースが多数あったり数値が存在しない部分に色々な記号が使用されたりしている事がわかります。

2-2. 読み込み

tidyverse の一角である `readr::read_csv()` を使用します。文字コードは Shift-JIS、先頭の解説部分から見出し 1 行目までの 14 行は読み飛ばします。

```
library(tidyverse)

data_raw <- readr::read_csv(file   = "mc130000.csv",
                            locale = readr::locale(encoding = "Shift-JIS"),
                            skip   = 14)
```

たくさんメッセージが出ますがここでは省略しています。読み込む際にどのような問題があってどのように変換されたか書かれているので、一度は目を通してみて下さい。

表部分の見出しは 14-15 行目の 2 行が使われており、14 行目にある「死亡数」や「死亡率」を反映していないため重複が発生し、もともとの列番号をつけた列名が付与されています。

```
data_raw %>% colnames()
```

```
 [1] "...1"      "1995...2"  "2000...3"  "2005...4"  "2010...5"  "2014...6"
 [7] "2015...7"  "2016...8"  "2017...9"  "2018...10" "2019...11" "2020...12"
[13] "2021...13" "2022...14" "2023...15" "1995...16" "2000...17" "2005...18"
[19] "2010...19" "2014...20" "2015...21" "2016...22" "2017...23" "2018...24"
[25] "2019...25" "2020...26" "2021...27" "2022...28" "2023...29"
```

3. 死亡数について整理

1 ページの表に粗死亡数と人口 10 万人対の死亡率が一緒に掲載されていますが、ここでは死亡数の方を見ていきます。

必要そうな処理としては以下のようなものが考えられます：

1. 列名（変数名）を R で扱いやすいように修正する
2. 死因欄に性別の見出しも入っているので、性別を先に分離する
3. 死因欄の整形のためのスペースを取り除く
4. 数値であるべきところを数値として扱えるように変換する
5. （オプション）Tidy data に整形する

3-1. Step1: 列名の付け直し【処理 1】

まだ列名を整理していないので、`dplyr::select()` で列番号を指定して 1-15 列目を抽出した後で、`dplyr::rename_with()` で列名をまとめて扱いやすいよう英数字に付け直します。

また、R では数字で始まる変数名は好ましくないので、各年の頭に “Death_” をつけておきます。（後で文字列を区切るための目印です。Death@ や単に D_ などでも構いませんが、演算記号を含む Death- や +Death+ は文法上エラーになります）

> ! 重要
>
> 以降、作業内容がわかるよう可能な限り `tidylog` パッケージの関数に置き換えています。慣れてきたら適宜 `dplyr::` や `tidyr::` に戻してください。
> `library(tidylog)` で読み込んでいなければ、`tidylog::` を消せば元の `tidyverse` のものに戻ります。

```
data_step1 <- data_raw %>%
  tidylog::select(1:15) %>%
  tidylog::rename_with(
    ~ c("Cause",
        paste0("Death_", c(1995, 2000, 2005, 2010, 2014:2023)))
  )
## select: dropped 14 variables (1995...16, 2000...17, 2005...18, 2010...19, 2014...20, …)
## rename_with: renamed 15 variables (Cause, Death_1995, Death_2000, Death_2005, Death_2010, …)
```

```r
head(data_step1)    # 内容の確認
```

```
# A tibble: 6 × 15
  Cause       Death_1995 Death_2000 Death_2005 Death_2010 Death_2014 Death_2015
  <chr>       <chr>      <chr>      <chr>      <chr>      <chr>      <chr>
1 総数        <NA>       <NA>       <NA>       <NA>       <NA>       <NA>
2 ⬚死亡総数… 922139     961653     1083796    1197014    1273025    1290510
3 01000 ⬚感… 18925      19858      23538      25863      25569      25241
4 01100 ⬚⬚… 1097       1212       1752       2313       2417       2333
5 01200 ⬚⬚… 3178       2656       2296       2129       2100       1956
6 01201 ⬚⬚… 2986       2461       2086       1880       1836       1723
# ℹ 8 more variables: Death_2016 <chr>, Death_2017 <chr>, Death_2018 <chr>,
#   Death_2019 <chr>, Death_2020 <chr>, Death_2021 <chr>, Death_2022 <chr>,
#   Death_2023 <chr>
```

（PDF 版では、元ファイル中に含まれているいわゆる全角空白を可視化するため、 PlemolJP というプログラミングフォントを使っています。通常は全角空白は見た目には分かりにくく落とし穴になりがちです）

3-2. Step2: 死因列（Cause）の整理【処理 2+3】

手順は大まかに以下のようになります。

- 死因と同じ欄にある「総数」、「男」、「女」は性別の見出しなので「性別」列を作って分離
- Cause 列の数字 5 桁コード（病名簡単分類）と死因は “_” でつなぐ
- 死因「死亡総数」は「00000_死亡総数」とする（e-Stat API のデータに準拠）
- 余計な空白は削除する
- 最後に 5 桁コードと病名を分離する

いくつもの段階を踏むような長いパイプラインは、はじめから全部書かず 1 段階ずつ確認しながら書いていくことをお勧めします。
また、このセクションでは普段の操作で Console に出力される結果をそのまま掲載するために結果の表示の仕方を変えています。

> **パイプの途中経過を確認したいとき**
>
> 既にある長いパイプラインのコードの動作を確認したいときは、`%>%` の後に `return()` 関数をつなぐことでそこまでで一旦終了としてその段階の結果を確認することが出来ます。（なお、base pipe では `|> return()` を使うことは出来ません）。

> **頻出関数 *stringr::str_replace_all(string, pattern, replacement)***
>
> - 文字列 `string` 中の `pattern` に当てはまる部分を `replacement` に置き換える
> - `pattern` にはワイルドカード・正規表現が使用可能
>   - String manipulation with stringr :: CHEATSHEET （注：リンク先は PDF）2 ページ目の一覧を参照
> - `pattern` 中で `()` で囲むことで、`replacement` の中で登場順に `\\1`, `\\2`, … で引用可能
> - おもに `dplyr::mutate()` の中で使われる

3-2-1. 性別列を作って分離

```
data_step2_1 <- data_step1 %>%
  # 性別列を作る
  tidylog::mutate(
    # 死因列の空白より後ろを削除すると、「総数」「男」「女」以外は空か数字のみになる
    Sex = stringr::str_replace_all(Cause, "\\s.*$", ""),
    # 数字も削除すると、Sex は「総数」「男」「女」しか残らない
    Sex = stringr::str_replace_all(Sex, "\\d+", ""),
    # 空になった行は NA_character_（NA の文字列型）＝空欄に置き換える
    Sex = dplyr::if_else(Sex == '', NA_character_, Sex),
    # 説明用に、性別列を死因列の次につくる（特に指定しないと末尾になる）
    .after = "Cause"
  )
## mutate: new variable 'Sex' (character) with 4 unique values and 99% NA

data_step2_1
## # A tibble: 423 × 16
##    Cause Sex   Death_1995 Death_2000 Death_2005 Death_2010 Death_2014 Death_2015
##    <chr> <chr> <chr>      <chr>      <chr>      <chr>      <chr>      <chr>
##  1 総数  総数  <NA>       <NA>       <NA>       <NA>       <NA>       <NA>
##  2 ⬚死… <NA>  922139     961653     1083796    1197014    1273025    1290510
##  3 0100… <NA>  18925      19858      23538      25863      25569      25241
##  4 0110… <NA>  1097       1212       1752       2313       2417       2333
##  5 0120… <NA>  3178       2656       2296       2129       2100       1956
##  6 0120… <NA>  2986       2461       2086       1880       1836       1723
##  7 0120… <NA>  192        195        210        249        264        233
##  8 0130… <NA>  4905       6216       8504       10676      11279      11357
##  9 0140… <NA>  5029       5121       6042       5614       4747       4514
## 10 0140… <NA>  880        885        786        539        482        407
## # ℹ 413 more rows
## # ℹ 8 more variables: Death_2016 <chr>, Death_2017 <chr>, Death_2018 <chr>,
## #   Death_2019 <chr>, Death_2020 <chr>, Death_2021 <chr>, Death_2022 <chr>,
## #   Death_2023 <chr>
```

```
data_step2_2 <- data_step2_1 %>%
  # Sex 列の空欄を下向きに埋めた後、実死亡数がない行＝もともと性別のみ書かれていた行を消す
  tidylog::fill(Sex, .direction = "down") %>%
  tidylog::drop_na(Death_2023)
## fill: changed 420 values (99%) of 'Sex' (420 fewer NAs)
## drop_na: removed 3 rows (1%), 420 rows remaining

data_step2_2
## # A tibble: 420 × 16
##    Cause Sex   Death_1995 Death_2000 Death_2005 Death_2010 Death_2014 Death_2015
##    <chr> <chr> <chr>      <chr>      <chr>      <chr>      <chr>      <chr>
##  1 ⬚死… 総数  922139     961653     1083796    1197014    1273025    1290510
##  2 0100… 総数  18925      19858      23538      25863      25569      25241
##  3 0110… 総数  1097       1212       1752       2313       2417       2333
##  4 0120… 総数  3178       2656       2296       2129       2100       1956
##  5 0120… 総数  2986       2461       2086       1880       1836       1723
##  6 0120… 総数  192        195        210        249        264        233
##  7 0130… 総数  4905       6216       8504       10676      11279      11357
##  8 0140… 総数  5029       5121       6042       5614       4747       4514
##  9 0140… 総数  880        885        786        539        482        407
## 10 0140… 総数  3542       3756       4855       4754       4033       3881
## # i 410 more rows
## # i 8 more variables: Death_2016 <chr>, Death_2017 <chr>, Death_2018 <chr>,
## #   Death_2019 <chr>, Death_2020 <chr>, Death_2021 <chr>, Death_2022 <chr>,
## #   Death_2023 <chr>
```

3-2-2. 病名（Cause）列の数字 5 桁コードと死因を “_” でつなぐ

```
data_step2_3 <- data_step2_2 %>%
  # 死因簡単分類の整理
  tidylog::mutate(
    # 空白を全部消す
    Cause = stringr::str_replace_all(Cause, "\\s*", ""),
    # 死亡総数は 00000_ をつける。それ以外は先頭の連続した数字のあとに _ をつける
    Cause = dplyr::if_else(Cause == "死亡総数",
                           "00000_死亡総数",
                           stringr::str_replace_all(Cause, "^(\\d+)", "\\1_"))
  )
## mutate: changed 420 values (100%) of 'Cause' (0 new NAs)

data_step2_3 %>% select(Cause)
## # A tibble: 420 × 1
##    Cause
##    <chr>
##  1 00000_死亡総数
##  2 01000_感染症及び寄生虫症
##  3 01100_腸管感染症
##  4 01200_結核
##  5 01201_呼吸器結核
##  6 01202_その他の結核
##  7 01300_敗血症
##  8 01400_ウイルス性肝炎
##  9 01401_B 型ウイルス性肝炎
## 10 01402_C 型ウイルス性肝炎
## # i 410 more rows
```

3-2-3. 死因の 5 桁コードと病名を分離して並べ替える

```
data_step2_4 <- data_step2_3 %>%
  # コードと病名を分離する
  tidylog::mutate(
    Cause_Code = stringr::str_split(Cause, "_", simplify = TRUE)[,1],
    Cause_Name = stringr::str_split(Cause, "_", simplify = TRUE)[,2]
  ) %>%
  # 必要な項目を並べ替え
  tidylog::select(Cause_Code, Cause_Name, Sex, starts_with("Death_"))
## mutate: new variable 'Cause_Code' (character) with 140 unique values and 0% NA
##         new variable 'Cause_Name' (character) with 140 unique values and 0% NA
## select: dropped one variable (Cause)

data_step2_4
## # A tibble: 420 × 17
##    Cause_Code Cause_Name      Sex   Death_1995 Death_2000 Death_2005 Death_2010
##    <chr>      <chr>           <chr> <chr>      <chr>      <chr>      <chr>
##  1 00000      死亡総数        総数  922139     961653     1083796    1197014
##  2 01000      感染症及び寄生… 総数  18925      19858      23538      25863
##  3 01100      腸管感染症      総数  1097       1212       1752       2313
##  4 01200      結核            総数  3178       2656       2296       2129
##  5 01201      呼吸器結核      総数  2986       2461       2086       1880
##  6 01202      その他の結核    総数  192        195        210        249
##  7 01300      敗血症          総数  4905       6216       8504       10676
##  8 01400      ウイルス性肝炎  総数  5029       5121       6042       5614
##  9 01401      B 型ウイルス性… 総数  880        885        786        539
## 10 01402      C 型ウイルス性… 総数  3542       3756       4855       4754
## # i 410 more rows
## # i 10 more variables: Death_2014 <chr>, Death_2015 <chr>, Death_2016 <chr>,
## #   Death_2017 <chr>, Death_2018 <chr>, Death_2019 <chr>, Death_2020 <chr>,
## #   Death_2021 <chr>, Death_2022 <chr>, Death_2023 <chr>
```

3-2-4. まとめて処理

少しずつ確認しながら進めてきた操作をひとまとめにします。(ここは `tidylog::` ではなく元の関数で示します)

```
data_step2 <- data_step1 %>%
  # 性別列を作る
  dplyr::mutate(
    # 死因列の空白より後ろを削除すると、「総数」「男」「女」以外は空か数字のみになる
    Sex = stringr::str_replace_all(Cause, "\\s.*$", ""),
    # 数字も削除すると、Sex は「総数」「男」「女」しか残らない
    Sex = stringr::str_replace_all(Sex, "\\d+", ""),
    # 空になった行は NA_character_（NA の文字列型）＝空欄に置き換える
    Sex = dplyr::if_else(Sex == '', NA_character_, Sex),
  ) %>%
  # Sex 列の空欄を下向きに埋めた後、実死亡数がない行＝もともと性別のみ書かれていた行を消す
  tidyr::fill(Sex, .direction = "down") %>%
  tidyr::drop_na(Death_2023) %>%
  # 死因簡単分類の整理
  dplyr::mutate(
    # 空白を全部消す
    Cause = stringr::str_replace_all(Cause, "\\s*", ""),
    # 死亡総数は 00000_ をつける。それ以外は先頭の連続した数字のあとに _ をつける
    Cause = dplyr::if_else(Cause == "死亡総数",
                           "00000_死亡総数",
                           stringr::str_replace_all(Cause, "^(\\d+)", "\\1_"))
  ) %>%
```

```r
  # コードと病名を分離する
  dplyr::mutate(
    Cause_Code = stringr::str_split(Cause, "_", simplify = TRUE)[,1],
    Cause_Name = stringr::str_split(Cause, "_", simplify = TRUE)[,2]
  ) %>%
  # 必要な項目を並べ替え
  dplyr::select(Cause_Code, Cause_Name, Sex, starts_with("Death_"))

head(data_step2)  # 確認
```

```
# A tibble: 6 × 17
  Cause_Code Cause_Name        Sex   Death_1995 Death_2000 Death_2005 Death_2010
  <chr>      <chr>             <chr> <chr>      <chr>      <chr>      <chr>
1 00000      死亡総数          総数  922139     961653     1083796    1197014
2 01000      感染症及び寄生虫… 総数  18925      19858      23538      25863
3 01100      腸管感染症        総数  1097       1212       1752       2313
4 01200      結核              総数  3178       2656       2296       2129
5 01201      呼吸器結核        総数  2986       2461       2086       1880
6 01202      その他の結核      総数  192        195        210        249
# i 10 more variables: Death_2014 <chr>, Death_2015 <chr>, Death_2016 <chr>,
#   Death_2017 <chr>, Death_2018 <chr>, Death_2019 <chr>, Death_2020 <chr>,
#   Death_2021 <chr>, Death_2022 <chr>, Death_2023 <chr>
```

3-3. Step3: 死亡数を数値として扱えるようにする【処理 4】

```r
dplyr::glimpse(data_step2)      # 各列の型を確認
```

```
Rows: 420
Columns: 17
$ Cause_Code <chr> "00000", "01000", "01100", "01200", "01201", "01202", "0130…
$ Cause_Name <chr> "死亡総数", "感染症及び寄生虫症", "腸管感染症", "結核", "呼…
$ Sex        <chr> "総数", "総数", "総数", "総数", "総数", "総数", "総数", "総…
$ Death_1995 <chr> "922139", "18925", "1097", "3178", "2986", "192", "4905", "…
$ Death_2000 <chr> "961653", "19858", "1212", "2656", "2461", "195", "6216", "…
$ Death_2005 <chr> "1083796", "23538", "1752", "2296", "2086", "210", "8504", …
$ Death_2010 <chr> "1197014", "25863", "2313", "2129", "1880", "249", "10676",…
$ Death_2014 <chr> "1273025", "25569", "2417", "2100", "1836", "264", "11279",…
$ Death_2015 <chr> "1290510", "25241", "2333", "1956", "1723", "233", "11357",…
$ Death_2016 <chr> "1308158", "25107", "2551", "1893", "1663", "230", "11512",…
$ Death_2017 <chr> "1340567", "24760", "2358", "2306", "2002", "304", "10213",…
$ Death_2018 <chr> "1362470", "24127", "2363", "2204", "1939", "265", "10312",…
$ Death_2019 <chr> "1381093", "23544", "2267", "2087", "1801", "286", "10217",…
$ Death_2020 <chr> "1372755", "22129", "2153", "1909", "1664", "245", "9801", …
$ Death_2021 <chr> "1439856", "22160", "1949", "1845", "1570", "275", "9989", …
$ Death_2022 <chr> "1569050", "23726", "2037", "1664", "1423", "241", "11346",…
$ Death_2023 <chr> "1576016", "24237", "2137", "1587", "1344", "243", "11619",…
```

男性の婦人科癌など値がない病名については「・」や「-」などで埋められているため、本来数値であるはずの `Death_****` がすべて “chr” つまり文字列扱いになっており、それを数値に変換します。

```
data_step3 <- data_step2 %>%
  # Death_1995 列から Death_2023 列にそれぞれ as.integer() を適用する
  tidylog::mutate(dplyr::across(Death_1995:Death_2023, as.integer))
## Warning: There were 14 warnings in `.fun()`.
## The first warning was:
## ℹ In argument: `dplyr::across(Death_1995:Death_2023, as.integer)`.
## Caused by warning:
## !  強制変換により NA が生成されました
## ℹ Run `dplyr::last_dplyr_warnings()` to see the 13 remaining warnings.
## mutate: converted 'Death_1995' from character to integer (23 new NA)
##         converted 'Death_2000' from character to integer (22 new NA)
##         converted 'Death_2005' from character to integer (23 new NA)
##         converted 'Death_2010' from character to integer (23 new NA)
##         converted 'Death_2014' from character to integer (22 new NA)
##         converted 'Death_2015' from character to integer (22 new NA)
##         converted 'Death_2016' from character to integer (23 new NA)
##         converted 'Death_2017' from character to integer (23 new NA)
##         converted 'Death_2018' from character to integer (22 new NA)
##         converted 'Death_2019' from character to integer (23 new NA)
##         converted 'Death_2020' from character to integer (13 new NA)
##         converted 'Death_2021' from character to integer (11 new NA)
##         converted 'Death_2022' from character to integer (11 new NA)
##         converted 'Death_2023' from character to integer (11 new NA)
```

```
dplyr::glimpse(data_step3)
```

```
Rows: 420
Columns: 17
$ Cause_Code <chr> "00000", "01000", "01100", "01200", "01201", "01202", "0130…
$ Cause_Name <chr> "死亡総数", "感染症及び寄生虫症", "腸管感染症", "結核", "呼…
$ Sex        <chr> "総数", "総数", "総数", "総数", "総数", "総数", "総数", "総…
$ Death_1995 <int> 922139, 18925, 1097, 3178, 2986, 192, 4905, 5029, 880, 3542…
$ Death_2000 <int> 961653, 19858, 1212, 2656, 2461, 195, 6216, 5121, 885, 3756…
$ Death_2005 <int> 1083796, 23538, 1752, 2296, 2086, 210, 8504, 6042, 786, 485…
$ Death_2010 <int> 1197014, 25863, 2313, 2129, 1880, 249, 10676, 5614, 539, 47…
$ Death_2014 <int> 1273025, 25569, 2417, 2100, 1836, 264, 11279, 4747, 482, 40…
$ Death_2015 <int> 1290510, 25241, 2333, 1956, 1723, 233, 11357, 4514, 407, 38…
$ Death_2016 <int> 1308158, 25107, 2551, 1893, 1663, 230, 11512, 3851, 407, 32…
$ Death_2017 <int> 1340567, 24760, 2358, 2306, 2002, 304, 10213, 3743, 419, 31…
$ Death_2018 <int> 1362470, 24127, 2363, 2204, 1939, 265, 10312, 3055, 368, 24…
$ Death_2019 <int> 1381093, 23544, 2267, 2087, 1801, 286, 10217, 2657, 336, 21…
$ Death_2020 <int> 1372755, 22129, 2153, 1909, 1664, 245, 9801, 2201, 353, 168…
$ Death_2021 <int> 1439856, 22160, 1949, 1845, 1570, 275, 9989, 1943, 337, 143…
$ Death_2022 <int> 1569050, 23726, 2037, 1664, 1423, 241, 11346, 1799, 353, 12…
$ Death_2023 <int> 1576016, 24237, 2137, 1587, 1344, 243, 11619, 1645, 350, 11…
```

3-4. Step4: 年も数値にする【処理 4】＋Tidy data に整形する【処理 5】

`tidyr::pivot_longer()` で年代が縦に並ぶ縦長の表に変形し、あわせて年も数値として扱えるようにします。

> **tidyr::pivot_longer() / pivot_wider()**
>
> 表を１例１行形式の横持ち（wide）と解析に使いやすい縦持ち（long）に相互に変換する重要関数です。 詳細は、 Data tidying with tidyr :: CHEATSHEET （注：リンク先は PDF）の Reshape Data セクションの図を見てください。

```
data_step4_long <- data_step3 %>%
  tidylog::pivot_longer(
    cols            = dplyr::starts_with("Death_"),   # Death_**** の列を変換対象にする
    names_to        = "Year",                         # 対象列の見出しを Year 列に格納
    names_prefix    = "Death_",                       #   その際に冒頭の "Death_" は除く
    names_transform = as.integer,                     #   年は整数として扱えるようにする
    values_to       = "Death"                         # 対象列の値（死亡数）は Death 列に格納
  )
## pivot_longer: reorganized (Death_1995, Death_2000, Death_2005, Death_2010,
## Death_2014, …) into (Year, Death) [was 420x17, now 5880x5]
```

```
head(data_step4_long, n = 8)
```

```
# A tibble: 8 × 5
  Cause_Code Cause_Name Sex    Year   Death
  <chr>      <chr>      <chr> <int>   <int>
1 00000      死亡総数   総数   1995  922139
2 00000      死亡総数   総数   2000  961653
3 00000      死亡総数   総数   2005 1083796
4 00000      死亡総数   総数   2010 1197014
5 00000      死亡総数   総数   2014 1273025
6 00000      死亡総数   総数   2015 1290510
7 00000      死亡総数   総数   2016 1308158
8 00000      死亡総数   総数   2017 1340567
```

このままでは、Sex 列に男女別の値と総数という意味がやや異なるデータが混在しているので、Sex 列を横に展開します。（総数の表と男女の表の 2 つに分割しても tidy data と言えるでしょう）

```
data_step4_wide <- data_step4_long %>%
  tidylog::pivot_wider(
    names_from  = "Sex",     # 新たな列名にするのは Sex
    values_from = "Death"    # (Cause + Year) x Sex に対応する死因をいれる
  ) %>%
  tidylog::rename(Total = 総数, Male = 男, Female = 女)    # 列名を英数化
## pivot_wider: reorganized (Sex, Death) into (総数, 男, 女) [was 5880x5, now 1960x6]
## rename: renamed 3 variables (Total, Male, Female)
```

```
head(data_step4_wide, n = 10)
```

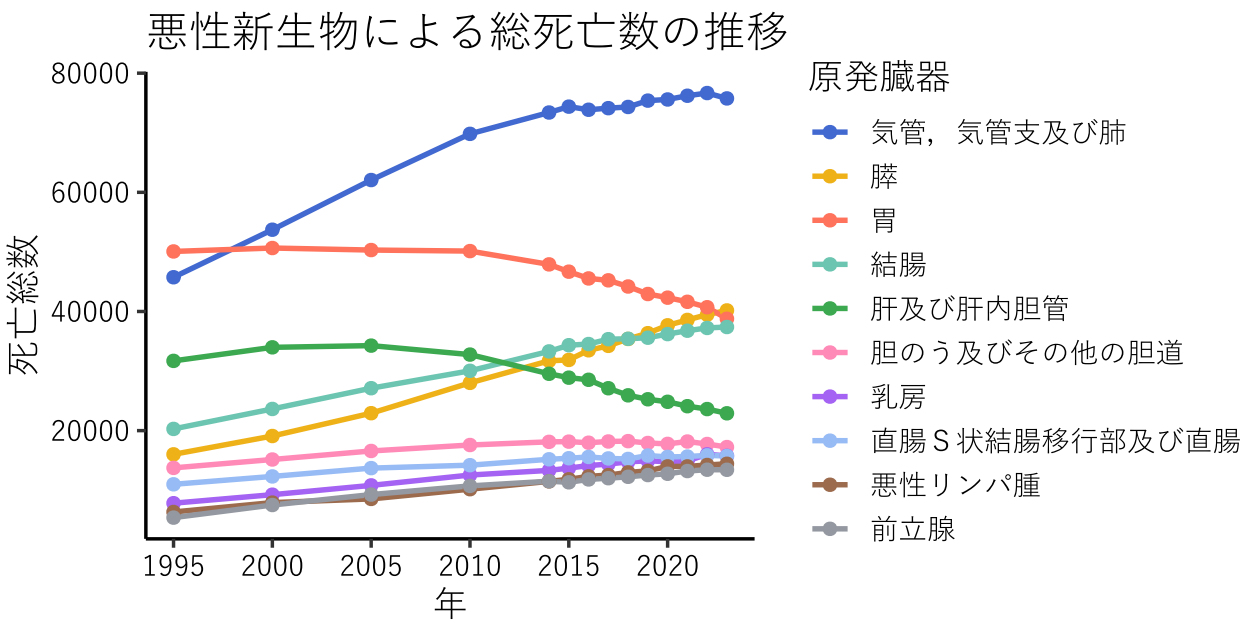
```
# A tibble: 10 × 6
   Cause_Code Cause_Name  Year   Total   Male Female
   <chr>      <chr>      <int>   <int>  <int>  <int>
 1 00000      死亡総数    1995  922139 501276 420863
 2 00000      死亡総数    2000  961653 525903 435750
 3 00000      死亡総数    2005 1083796 584970 498826
 4 00000      死亡総数    2010 1197014 633701 563313
 5 00000      死亡総数    2014 1273025 660340 612685
 6 00000      死亡総数    2015 1290510 666728 623782
 7 00000      死亡総数    2016 1308158 674946 633212
 8 00000      死亡総数    2017 1340567 690770 649797
 9 00000      死亡総数    2018 1362470 699138 663332
10 00000      死亡総数    2019 1381093 707421 673672
```

これで tidy data の要件を満たす状態となりましたが、解析については内容によっては Step3 のデータの方が扱いやすい場合もあります。

4. 集計例

```
# OS を見てフォントを選択
font_sans <- switch(Sys.info()["sysname"],
                    "Windows" = "Yu Gothic",
                    "Darwin"  = "Hiragino Sans",
                    "Noto Sans CJK JP")

data_step3 %>%
  # 性別「総数」のみに絞る
  tidylog::filter(Sex == "総数") %>%
  # 疾患「悪性新生物＜腫瘍＞」( コード 021** ) を抽出、全臓器の合計 ( コード 02100 ) は除外
  tidylog::filter(stringr::str_starts(Cause_Code, "021")) %>%
  tidylog::filter(Cause_Code != "02100") %>%
  # ほぼ全てに共通する「～の悪性新生物＜腫瘍＞」を削除
  tidylog::mutate(Cause_Name = stringr::str_remove_all(Cause_Name, "の悪性新生物＜腫瘍＞")) %>%
  # 2023 年の上位 10 臓器を抽出 ( その他は除外 )
  tidylog::filter(!stringr::str_starts(Cause_Name, "その他")) %>%
  dplyr::arrange(desc(Death_2023)) %>%
  tidylog::slice_head(n = 10) %>%
  # 病名を多い順に並べ替えておく ( 順序変数化 )
  tidylog::mutate(
    Cause_Name = forcats::fct_relevel(Cause_Name,
                                      Cause_Name[order(Death_2023, decreasing = TRUE)])
  ) %>%
  # 作図用に縦持ちに変換
  tidylog::pivot_longer(
    cols            = dplyr::starts_with("Death_"),  # Death_**** の列を変換対象にする
    names_to        = "Year",                        # 対象列の見出しを Year 列に格納
    names_prefix    = "Death_",                      #   その際に冒頭の "Death_" は除く
    names_transform = as.integer,                    #   年は整数として扱えるようにする
    values_to       = "Death"                        # 死亡数は Death 列に格納
  ) %>%
  # 作図
  ggplot2::ggplot(aes(x = Year, y = Death, colour = Cause_Name, group = Cause_Name)) +
    ggplot2::geom_line(linewidth = 1) +
    ggplot2::geom_point(size = 2) +
    ggplot2::labs(
      title  = "悪性新生物による総死亡数の推移",
      x      = "年",
      y      = "死亡総数",
      colour = "原発臓器"
    ) +
    ggsci::scale_colour_observable() +
    ggplot2::theme_classic(base_family = font_sans, base_size = 14) +
    ggplot2::theme(axis.text = element_text(colour = "black"))
## filter: removed 280 rows (67%), 140 rows remaining
## filter: removed 118 rows (84%), 22 rows remaining
## filter: removed one row (5%), 21 rows remaining
## mutate: changed 19 values (90%) of 'Cause_Name' (0 new NAs)
## filter: removed 2 rows (10%), 19 rows remaining
## slice_head: removed 9 rows (47%), 10 rows remaining
## mutate: converted 'Cause_Name' from character to factor (0 new NA)
## pivot_longer: reorganized (Death_1995, Death_2000, Death_2005, Death_2010, Death_2014, ⋯) into
(Year, Death) [was 10x17, now 140x5]
```


悪性新生物による総死亡数の推移
80000
60000
40000
20000
死亡総数
1995
2000
2005
2010
2015
2020
年
原発臓器
気管，気管支及び肺
膵
胃
結腸
肝及び肝内胆管
胆のう及びその他の胆道
乳房
直腸Ｓ状結腸移行部及び直腸
悪性リンパ腫
前立腺