Tidy Data: An Introduction To The Tidyverse

MA2003B Application of Multivariate Methods in Data Science

Raul Gomez

Tecnológico de Monterrey

Tidy Data and the Tidyverse

The Tidyverse is a collection of R packages designed for data science.

Tidy Data and the Tidyverse

The Tidyverse is a collection of R packages designed for data science.

It provides a consistent and user-friendly interface for data manipulation, visualization, and analysis.

Tidy Data and the Tidyverse

The Tidyverse is a collection of R packages designed for data science.

It provides a consistent and user-friendly interface for data manipulation, visualization, and analysis.

The core idea of behind the design of the Tidyverse is the concept of “tidy data”.

Definition

Tidy Data is a way of structuring datasets to simplify its use. In tidy data:

1. Each variable forms a column.

Definition

Tidy Data is a way of structuring datasets to simplify its use. In tidy data:

1. Each variable forms a column.
2. Each observation forms a row.

Definition

Tidy Data is a way of structuring datasets to simplify its use. In tidy data:

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.

Definition

Tidy Data is a way of structuring datasets to simplify its use. In tidy data:

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.

This structure allows for easier data manipulation, analysis, and visualization, as it aligns with the way data is typically processed in R.

Tibbles and pipes

In the Tidyverse, data is represented using a special type of data frame called a **tibble**.

Tibbles and pipes

In the Tidyverse, data is represented using a special type of data frame called a **tibble**.

Tibbles are more user-friendly than traditional data frames, as they provide better printing and subsetting behavior.

Tibbles and pipes

In the Tidyverse, data is represented using a special type of data frame called a **tibble**.

Tibbles are more user-friendly than traditional data frames, as they provide better printing and subsetting behavior.

They also allow for more intuitive handling of data. In particular, tibbles are designed to work seamlessly with the pipe operator (%>%), which allows for chaining together multiple operations in a clear and concise manner.

Tibbles and pipes

In the Tidyverse, data is represented using a special type of data frame called a **tibble**.

Tibbles are more user-friendly than traditional data frames, as they provide better printing and subsetting behavior.

They also allow for more intuitive handling of data. In particular, tibbles are designed to work seamlessly with the pipe operator (%>%), which allows for chaining together multiple operations in a clear and concise manner.

To illustrate these ideas, in the next section we will perform data cleaning on the “WHO” dataset, to make sure that the associated tibble satisfies the tidy data principles.

Tibbles and pipes

In the Tidyverse, data is represented using a special type of data frame called a **tibble**.

Tibbles are more user-friendly than traditional data frames, as they provide better printing and subsetting behavior.

They also allow for more intuitive handling of data. In particular, tibbles are designed to work seamlessly with the pipe operator (%>%), which allows for chaining together multiple operations in a clear and concise manner.

To illustrate these ideas, in the next section we will perform data cleaning on the "WHO" dataset, to make sure that the associated tibble satisfies the tidy data principles.

This is usually the first step in Exploratory Data Analysis (EDA) and is crucial for ensuring that the data is in a suitable format for analysis and visualization.

Tidying the WHO dataset

Before starting, we need to load the required libraries and load the WHO dataset:

```r
library("tidyverse")
library("here")
library("cowplot")
library("patchwork")
library("krulRutils")
library("ISLR2")
library("magrittr")

options(scipen = 999) # Disable scientific notation
data(who)
```

We can now start our “tidying” process.

We can now start our “tidying” process.

The main problem with this dataset is that it contains variables (like “new_sp_m014”) that are not very clear and that contain more than one piece of information.

We can now start our “tidying” process.

The main problem with this dataset is that it contains variables (like “new_sp_m014”) that are not very clear and that contain more than one piece of information.

So we need to separate these variables and introduce better names for them and their associated values.

```r
who_tidy <- who %>%
  pivot_longer(
    cols = starts_with("new"),
    names_to = "key",
    values_to = "cases",
    values_drop_na = TRUE
  ) %>%
  mutate(
    key = if_else(
      startsWith(key, "newrel"),
      sub("newrel", "new_rel", key),
      key
    ),
    cases = as.integer(cases)
  )
```

```r
who_tidy <- who_tidy %>%
  separate(key, into = c("new", "type", "sexage"), sep = "_") %>%
  separate(sexage, into = c("sex", "age"), sep = 1) %>%
  select(-new, -iso2, - iso3)
```

We now want to convert the columns "type", "sex", and "age" into factors. We start by constructing the following lookup tables:

```r
who_type_lookup_tbl <- tibble(
  code = c("ep", "rel", "sn", "sp"),
  label = c(
    "Extrapulmonary TB",
    "Relapse case",
    "Smear-Negative pulmonary TB",
    "Smear-Positive pulmonary TB"
  )
)
```

```r
who_sex_lookup_tbl <- tibble(
  code = c("f", "m"),
  label = c("Female", "Male")
)
```

```r
who_age_lookup_tbl <- tibble(
  code = c("014", "1524", "2534", "3544", "4554", "5564", "65"),
  label = c(
    "0-14",
    "15-24",
    "25-34",
    "35-44",
    "45-54",
    "55-64",
    "65+"
  )
)
```

We can now append factor columns for the corresponding variables in the WHO dataset:

```r
who_factor <- who_tidy %>%
  convert_codes_to_factor(
    code_col = type,
    lookup_tbl = who_type_lookup_tbl,
    lookup_code_col = code,
    lookup_label_col = label,
  )
```

```r
who_factor <- who_factor %>%
  convert_codes_to_factor(
    code_col = sex,
    lookup_tbl = who_sex_lookup_tbl,
    lookup_code_col = code,
    lookup_label_col = label,
  )
```

```r
who_factor <- who_factor %>%
  convert_codes_to_factor(
    code_col = age,
    lookup_tbl = who_age_lookup_tbl,
    lookup_code_col = code,
    lookup_label_col = label,
  )
```

Finally, we want to use this information to analyze the number of cases of tuberculosis by type and sex. We start by generating the corresponding frequency tibble.

```r
who_type_sex_tbl <- who_factor %>%
  count(type_factor, sex_factor, wt = cases, name = "cases") %>%
  print(n = Inf)
```

```
# A tibble: 8 x 3
  type_factor                 sex_factor    cases
  <fct>                       <fct>         <int>
1 Extrapulmonary TB           Female       941880
2 Extrapulmonary TB           Male        1044299
3 Relapse case                Female      1201596
4 Relapse case                Male        2018976
5 Smear-Negative pulmonary TB Female      2439139
6 Smear-Negative pulmonary TB Male        3840388
7 Smear-Positive pulmonary TB Female     11324409
8 Smear-Positive pulmonary TB Male       20586831
```

The “Grammar of Graphics” and ggplot2

The concept of the “Grammar of Graphics” was introduced by Leland Wilkinson in his 1999 book **The Grammar of Graphics**.

The “Grammar of Graphics” and ggplot2

The concept of the “Grammar of Graphics” was introduced by Leland Wilkinson in his 1999 book **The Grammar of Graphics**.

This provides a framework for understanding how to create visualizations in a systematic way. It breaks down the process of creating a plot into components such as data, aesthetics, geometries, statistics, coordinates, and themes.

The “Grammar of Graphics” and ggplot2

The concept of the “Grammar of Graphics” was introduced by Leland Wilkinson in his 1999 book **The Grammar of Graphics**.

This provides a framework for understanding how to create visualizations in a systematic way. It breaks down the process of creating a plot into components such as data, aesthetics, geometries, statistics, coordinates, and themes.

The `ggplot2` package implements this grammar in R, allowing users to build complex visualizations by layering these components.

Layers in ggplot2

In `ggplot2`, plots are constructed by adding multiple **layers** that define the components of the visualization.

Layers in ggplot2

In `ggplot2`, plots are constructed by adding multiple **layers** that define the components of the visualization.

Each layer corresponds to a specific aspect of the plot, and together they form the complete graphic.

These layers include:

1. **Data:** The dataset to be visualized. This is the source of the information for the plot.
2. **Aesthetics:** Mappings that relate variables in the data to visual properties of the plot, such as:
   - Position on the x- and y-axes.
   - Color, fill.
   - Size, shape.
   - Transparency.

   These mappings define **what** data is shown and **how** it is represented visually.

3. **Geometries:** Geometric objects that display the data, for example:
   - `geom_point()` for scatterplots.
   - `geom_line()` for line graphs.
   - `geom_col()` for bar charts.
   - `geom_histogram()` for histograms.

   The geometry controls **how** the data is drawn.

4. **Statistical transformations:** Optional calculations applied to the data before plotting, such as:
   - `stat_bin()` for binning data in histograms.
   - `stat_smooth()` for fitting smooth curves.

   These are preprocessing steps for the data visualization.
5. **Scales:** Define how data values are translated into visual properties, for example mapping numeric values to colors or shapes. Scales also control axis ticks, legends, and guides.
6. **Coordinates:** The coordinate system used for the plot, such as Cartesian (`coord_cartesian()`), polar (`coord_polar()`), or flipped coordinates (`coord_flip()`).

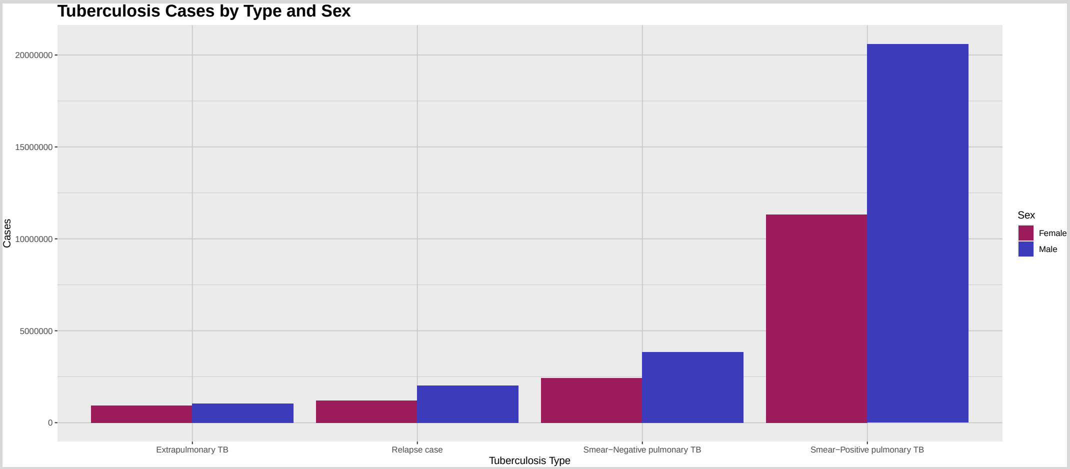
7. **Facets:** Methods to split the data into subsets and display multiple plots arranged in a grid:
   - `facet_wrap()`
   - `facet_grid()`
8. **Labels:** `labs()` adds titles, axis labels, and captions.
9. **Themes:** `theme()` controls the overall appearance of the plot (fonts, background, gridlines).

Each layer can be combined and customized to build complex and elegant plots. This **layered grammar** allows you to think of your graphic as a composition of independent components rather than a single, monolithic object.

We will illustrate these concepts by plotting the number of cases of tuberculosis by type and sex, using the frequency tibble we created earlier.

```r
who_type_sex_plot <- who_type_sex_tbl %>%
  ggplot(aes(x = type_factor, y = cases, fill = sex_factor)) +
  geom_col(position = "dodge") +
  c_scale_fill("C rose", "C blue") +
  labs(
    title = "Tuberculosis Cases by Type and Sex",
    x = "Tuberculosis Type",
    y = "Cases",
    fill = "Sex"
  ) +
  theme_krul()

who_type_sex_plot
```


Tuberculosis Cases by Type and Sex
Cases
20000000
15000000
10000000
5000000
0
Extrapulmonary TB
Relapse case
Smear−Negative pulmonary TB
Smear−Positive pulmonary TB
Tuberculosis Type
Sex
Female
Male

The Billboard dataset

The problem with the Billboard dataset, according to the principles of “tidy data”, is that the weeks in which a song appeared in the Billboard chart are represented as columns, which makes it difficult to analyze the data.

The Billboard dataset

The problem with the Billboard dataset, according to the principles of “tidy data”, is that the weeks in which a song appeared in the Billboard chart are represented as columns, which makes it difficult to analyze the data.

Using the techniques we have learned so far, we should tidy the Billboard dataset, so that it can be used to analyze the evolution of the ranking of the song “Who Let The Dogs Out” by “Baha Men” in the Billboard Hot 100 chart.

The Billboard dataset

The problem with the Billboard dataset, according to the principles of “tidy data”, is that the weeks in which a song appeared in the Billboard chart are represented as columns, which makes it difficult to analyze the data.

Using the techniques we have learned so far, we should tidy the Billboard dataset, so that it can be used to analyze the evolution of the ranking of the song “Who Let The Dogs Out” by “Baha Men” in the Billboard Hot 100 chart.

The final visualization should look like the one shown below.

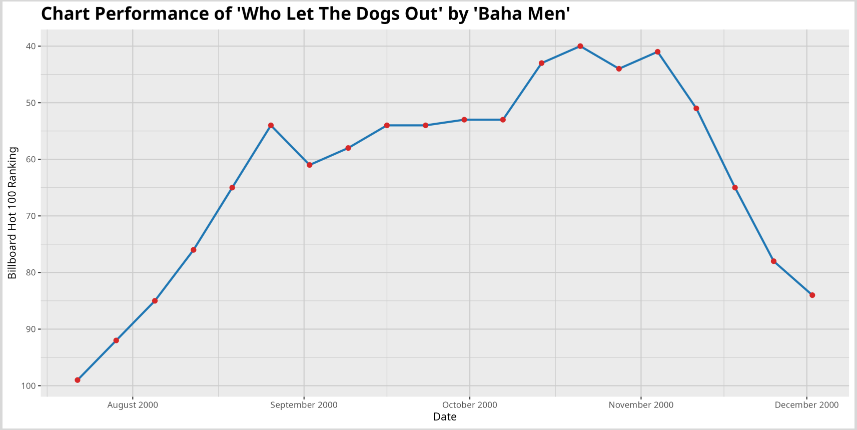


Chart Performance of 'Who Let The Dogs Out' by 'Baha Men'
Billboard Hot 100 Ranking
40
50
60
70
80
90
100
August 2000
September 2000
October 2000
November 2000
December 2000
Date

Bibliography

Cleveland, W. S. (1993) *The Elements of Graphing Data*. Hobart Press.

Hyndman, R. J., and Athanasopoulos, G. (2021) *Forecasting: Principles and Practice* (3rd ed.). OTexts. https://otexts.com/fpp3/

Wickham, H. (2014) Tidy data. *Journal of Statistical Software*, *59*(10), 1–23. https://doi.org/10.18637/jss.v059.i10

Wickham, H. (2016) *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag.

Wickham, H., and Grolemund, G. (2017) *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. O'Reilly Media.

Wickham, H. (2019) Functional programming with purrr. *UseR! Conference*.

Bibliography

Wickham, H., et al. (2019) Welcome to the tidyverse. *Journal of Open Source Software*, *4*(43), 1686. https://doi.org/10.21105/joss.01686

Wilkinson, L. (1999) *The Grammar of Graphics*. Springer-Verlag.