MA2003B Application of Multivariate Methods in Data Science

Module 1: Interdependence Relationships

Instructor: Raul Gomez
Email: rgomez@tec.mx
Office: A4-123A

August 13, 2025

1 Tidy Data: An Introduction to 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. Although formally introduced as an ecosystem in 2016, many of these packages were developed by Hadley Wickham between 2007–2014.

1.1 Tidy Data

The core idea of behind the design of the Tidyverse is Wickham's definition of "tidy data" [3].

Definition 1.1. *Tidy Data* is a way of structuring datasets to make them easier to work with. 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. In this section we will describe the basics of the Tidyverse but, for more information on its use, the interested reader can check the following references: [5, 6, 7].

1.1.1 Tibbles and pipes

In the Tidyverse, data is often 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 (usually known as "verbs" in the context of the tidyverse) 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.

1.1.2 An Example: Tidying the WHO dataset

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

```
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. 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.

```
# We start by creating the variable "who_tidy"
# that contains the cleaned WHO dataset
who_tidy <- who %>%
  # We use pivot_longer to eliminate the variables starting with "new"
  # and use them as values instead
  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)
  ) %>%
  separate(key, into = c("new", "type", "sexage"), sep = "_") %>%
  separate(sexage, into = c("sex", "age"), sep = 1) %>%
  select(-new, -iso2, - iso3)  %T>%
  # Finally, we save the tidy data in both RDS and CSV formats
  saveRDS(here("data", "who_tidy.rds")) %>%
  write_csv(here("data", "who_tidy.csv")) %>%
  glimpse()
```

```
Rows: 76,046
```

```
Columns: 6
$ country <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", "A~
$ year    <dbl> 1997, 1997, 1997, 1997, 1997, 1997, 1997, 1997, 1997, 1997, 19~
$ type    <chr> "sp", "sp", "sp", "sp", "sp", "sp", "sp", "sp", "sp", "sp", "s~
$ sex     <chr> "m", "m", "m", "m", "m", "m", "m", "f", "f", "f", "f", "f", "f~
$ age     <chr> "014", "1524", "2534", "3544", "4554", "5564", "65", "014", "1~
$ cases   <int> 0, 10, 6, 3, 5, 2, 0, 5, 38, 36, 14, 8, 0, 1, 30, 129, 128, 90~
```

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

```
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"
  )
)

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

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:

```
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,
  ) %>%
  convert_codes_to_factor(
    code_col = sex,
    lookup_tbl = who_sex_lookup_tbl,
    lookup_code_col = code,
    lookup_label_col = label,
  ) %>%
  convert_codes_to_factor(
    code_col = age,
    lookup_tbl = who_age_lookup_tbl,
    lookup_code_col = code,
    lookup_label_col = label,
  ) %>%
  glimpse()
```

```
Rows: 76,046
Columns: 9
$ country     <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan"~
$ year        <dbl> 1997, 1997, 1997, 1997, 1997, 1997, 1997, 1997, 1997, 1997~
$ type        <chr> "sp", "sp", "sp", "sp", "sp", "sp", "sp", "sp", "sp", "sp"~
$ sex         <chr> "m", "m", "m", "m", "m", "m", "m", "f", "f", "f", "f", "f"~
$ age         <chr> "014", "1524", "2534", "3544", "4554", "5564", "65", "014"~
$ cases       <int> 0, 10, 6, 3, 5, 2, 0, 5, 38, 36, 14, 8, 0, 1, 30, 129, 128~
$ type_factor <fct> Smear-Positive pulmonary TB, Smear-Positive pulmonary TB, ~
$ sex_factor  <fct> Male, Male, Male, Male, Male, Male, Male, Female, Female, ~
$ age_factor  <fct> 0-14, 15-24, 25-34, 35-44, 45-54, 55-64, 65+, 0-14, 15-24,~
```

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
```

1.2 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* [8]. 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 [4] implements this grammar in R, allowing users to build complex visualizations by layering these components. For more information on the Grammar of Graphics, its history and applications, the interested reader can consult the following references: [1, 2].

1.2.1 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 (usually a data frame or tibble). 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.

1.2.2 Plotting with ggplot2

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()
```

We can now save and plot the graph:

```r
ggsave(
  filename = here("images", "who_plot.png"),
  plot = who_type_sex_plot,
  width = 12,
  height = 6,
  dpi = 300
)
who_type_sex_plot
```

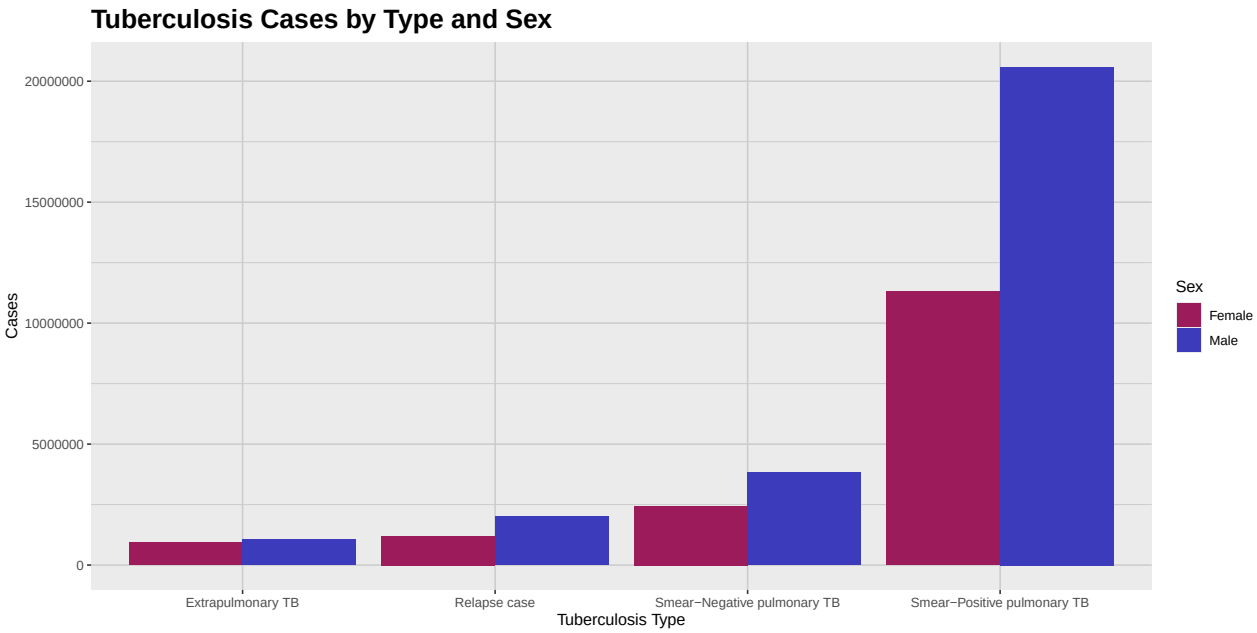

Figure 1: This plot shows a comparative analysis of the number of cases of tuberculosis by type and sex. As we can see from the plot, the number of cases is consistently higher for the male population across all types of tuberculosis.

1.3 Exercise: Tidying 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, the reader is required to 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.

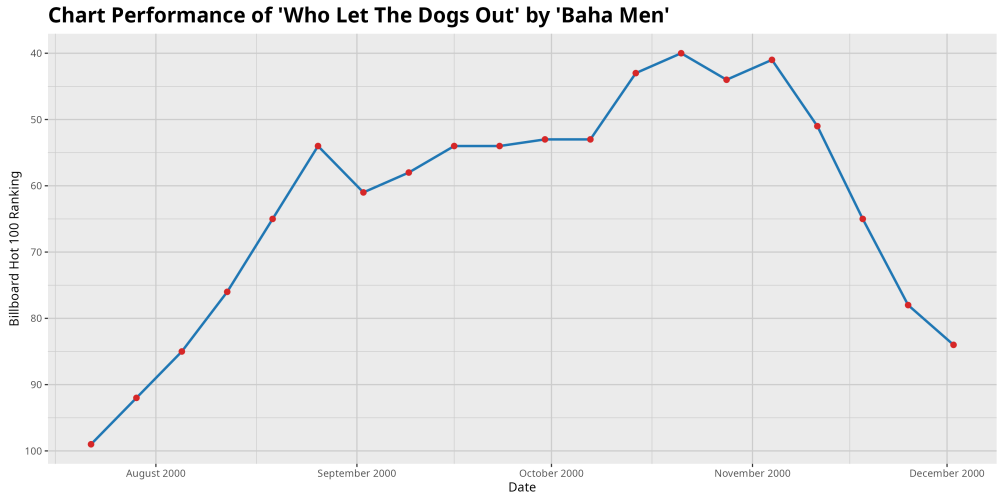

Figure 2: Evolution of the ranking of the song 'Who Let The Dogs Out' by 'Baha Men' in the Billboard Hot 100 chart. The song peaked at rank 40 in the week of October 21st, 2000, and remained in the top 100 for several weeks.

References

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

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

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

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

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

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

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

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