Tidying Data in R

Bas Machielsen

Utrecht University

December 29, 2019

What is tidy data?

- In order for your data to be ready for analysis, it needs to be **tidy**.
- Tidy data is defined as:
  data sets that are arranged such that each variable is a column and each observation (or case) is a row.
- I will attempt to explain how to put this principle in practice using a set of packages called the **tidyverse**, but strictly speaking, only the `tidyr` package is required.
- You can download the .Rmd and .csv files from my **github page**, and follow along!

What is (un)tidy data?

All tidy data are similar to each other. All untidy data are untidy in their own way.

- Although tidy data shares the properties mentioned in the previous slide, untidy data refers (broadly speaking) to unstructured data.
- An example (from www.tidyverse.org):

Table 1: This is what an untidy dataset looks like

name	treatmenta	treatmentb
John Smith	NA	18
Jane Doe	4	1
Mary Johnson	6	7

An example of untidy data

- Is each obervation in a row?
- Is each variable in a column?
- What are the variables here?

Table 2: This is also what an untidy dataset looks like

treatment	John.Smith	Jane.Doe	Mary.Johnson
a	NA	4	6
b	18	1	7

Example: from untidy to tidy data

- The variables are (i) the treatment, (ii) the subject, and (iii) the 'score' corresponding to each subject-treatment observations.

```
preg %>%
    pivot_longer(treatmenta:treatmentb,
                 names_to = "treatment", values_to = "score") %>%
    mutate(treatment = gsub("treatment", "", treatment)) %>%
    arrange(name, treatment) %>%
    kable(caption = "Test", booktabs = TRUE,
          row.names = FALSE) %>%
    kable_styling(latex_options = "striped")
```

Example: from untidy to tidy data

- The former chunk of code transforms the data in the first table to the following table:

Table 3: This is tidy data

name	treatment	score
Jane Doe	a	4
Jane Doe	b	1
John Smith	a	NA
John Smith	b	18
Mary Johnson	a	6
Mary Johnson	b	7

Transforming the data

- The key command in the former chunk of code is `pivot_longer`. With the arguements data, columns, names_to and values_to.
- All the other functions are essentially layout changes, not fundamental transformations.
- The command pivot_longer basically transforms the dataset from:

Table 4: This is what an untidy dataset looks like

name	treatmenta	treatmentb
John Smith	NA	18
Jane Doe	4	1
Mary Johnson	6	7

Transforming the data

- to :

```
pivot_longer(data = preg, cols = 2:3, names_to = "treatment",
             values_to = "score") %>%
    kable(caption = "This is what a tidy dataset looks like",
      booktabs = TRUE, row.names = FALSE) %>%
  kable_styling(latex_options = "striped")
```

Table 5: This is what a tidy dataset looks like

name	treatment	score
John Smith	treatmenta	NA
John Smith	treatmentb	18
Jane Doe	treatmenta	4
Jane Doe	treatmentb	1
Mary Johnson	treatmenta	6
Mary Johnson	treatmentb	7

Contents

- The best way to see how tidy data works is by using examples. In this lecture, I will demonstrate extensively how to create tidy data that is ready for analysis using two examples.
- First, I will show how to combine and tidy data on economic development from the World Bank.
- Afterwards, I show how to combine and tidy data from Amadeus, a commercial database.
- On the fly, I will demonstrate how simple it is to analyze or create graphs with tidy data.
- Finally, I will use other examples to show how to tidy more messy data, inspired by examples from the **tidyverse website**

World Bank Dataset

- Let us now download some data from the World Bank database. I extract the 55 basic World Development Indicators from **here** for 20 countries.
- Importing the data..

```
worldbank <- read_csv("wb.csv")

dim(worldbank)
```

```
## [1] 1105   14
```

- The data has 1105 observations and 14 variables. That is not at all what we wanted. What do the data look like?

World Bank Dataset

```r
kable(worldbank[1:4,c(1,2,4,5)],
      caption = "Work Bank Data - Untidy!",
      booktabs = TRUE, row.names = FALSE) %>%
  kable_styling(latex_options = "striped")
```

Table 6: Work Bank Data - Untidy!

Country Name	Country Code	Series Code	2009 [YR2009]
Argentina	ARG	SP.ADO.TFRT	63.1896
Argentina	ARG	NV.AGR.TOTL.ZS	5.27362346890139
Argentina	ARG	ER.H2O.FWTL.ZS	..
Argentina	ARG	SH.STA.BRTC.ZS	97.9

- Very untidy dataset! NA observations are entered as `..`, and variable names require an extensive definition before you know what they mean.
- Furthermore, years are not notated straightforwardly, and the data violates the tidy data principles.

What to do?

- First, let us try to save the variable names and series code to another dataset (step 1), and delete the names from the worldbank dataset (step 2).

```
#Step 1
wbnames <- cbind(unique(worldbank[,3]),unique(worldbank[,4])) %>%
    na.omit()

#Step 2
worldbank <- worldbank[1:1100,-3]
```

Almost there..

- Now, let's try to unpivot the columns containing the years..

```
worldbank <- pivot_longer(worldbank, 4:13,
             names_to = "years",
             values_to = "value")
```

- This is what the data looks like right now.

Table 7: The data then looks like this.

Country Code	Series Code	years	value
ARG	SP.ADO.TFRT	2009 [YR2009]	63.1896
ARG	SP.ADO.TFRT	2010 [YR2010]	63.3154
ARG	SP.ADO.TFRT	2011 [YR2011]	63.4412
ARG	SP.ADO.TFRT	2012 [YR2012]	63.567

Final steps

- Let's now clean the "years" variable (step 1), set the `..` observations to NA (step 2), and "widen" the data so that every separate indicator gets a column (step 3).
- Step 3 is done using the `pivot_wider` command: it widens the data and shortens the number of observations, because it transfers information from rows to columns.

```r
#Step 1
worldbank <- worldbank %>%
    mutate(years = substring(years, 1, 4))

#Step 2
worldbank$value[worldbank$value == ".."] <- NA

#Step 3
worldbank <- pivot_wider(data = worldbank,
                         names_from = `Series Code`,
                         values_from = `value`)
```

Done!

- `pivot_wider`, takes three arguments: the data, from which column it should take the new column names (names_from), and from which column it should take the values belonging to the corresonding cels (values_from).
- This is what it looks like!

Table 8: Tidy Data

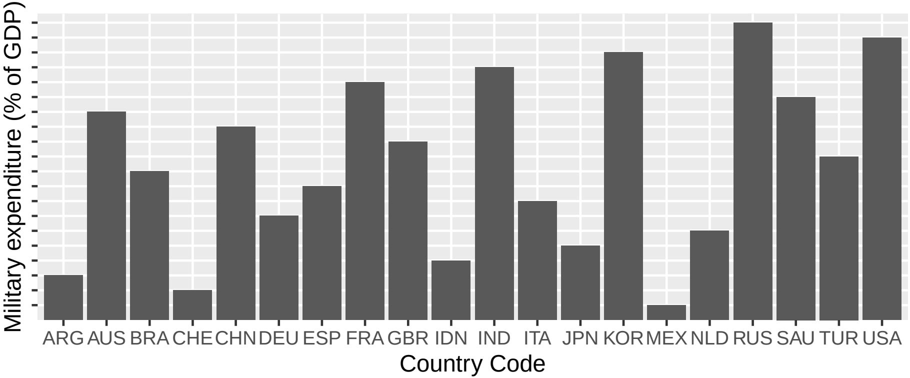
Country Name	Country Code	years	SP.ADO.TFRT	NV.AGR.TOTL.ZS
Argentina	ARG	2009	63.1896	5.27362346890139
Argentina	ARG	2010	63.3154	7.13216745078964
Argentina	ARG	2011	63.4412	6.99873377022519
Argentina	ARG	2012	63.567	5.7817442068501

Military Spending

- Suppose I want to know the military expenditures as a percentage of GDP MS.MIL.XPND.GD.ZS for each country in the dataset in 2015.
- I can use the `wbnames` information we've saved to add labels to the data.

```
worldbank %>%
    filter(years == "2015") %>%
    ggplot(aes(x = `Country Code`, y = `MS.MIL.XPND.GD.ZS`)) +
    geom_col() +
    labs(y = as.character(wbnames[match("MS.MIL.XPND.GD.ZS",
                                        wbnames[,2]),1])) +
    theme(axis.text.y =element_blank())
```

Military Spending

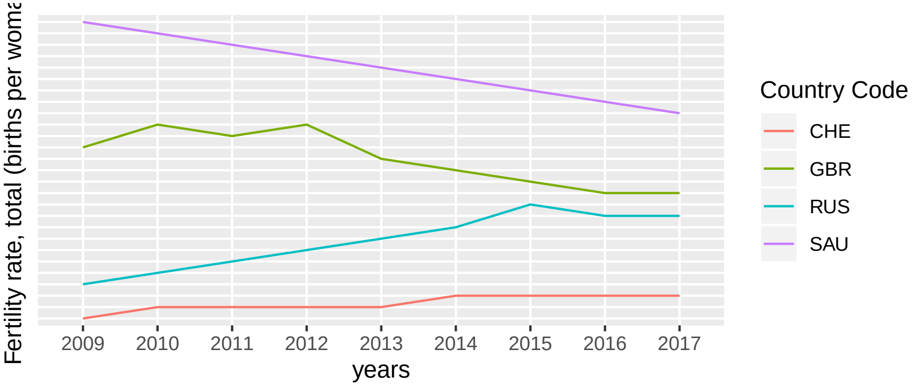


Fertility Rate

- Suppose now you want to track the evolution of the fertility rate over the last 10 years of some countries in the dataset.

```
worldbank %>%
    filter(years < 2018,
           `Country Code` == "CHE" |
             `Country Code` == "GBR"
           | `Country Code` == "RUS" |
             `Country Code` == "SAU") %>%
    ggplot(aes(x = years, y = `SP.DYN.TFRT.IN`,
                group = `Country Code`,
                color = `Country Code`)) +
    geom_line() +
    labs(y = as.character(wbnames[match("SP.DYN.TFRT.IN",
                                          wbnames[,2]),1])) +
    theme(axis.text.y =element_blank(),
          axis.ticks.y = element_blank())
```

Fertility Rate



Another untidy dataset..

- Now, I will download some data from Amadeus, a commercial database collected information about firms across the world.
- I collect data from 2000 European companies on their financial information, the number of employees, and the number of directors/managers.

```
amadeus <- read.csv("Amadeus_Export_1.csv", fileEncoding = "UCS-2LE'

dim(amadeus)
```

```
## [1] 2000   68
```

What does the data look like?

- These are some of the variables..

Table 9: Variable names

Mark
Company.name
City
Country.ISO.code
NACE.code
Cons..code
Last.year
Operating.revenue..Turnover..th.EUR.2019
Operating.revenue..Turnover..th.EUR.2018

Tidying the Amadeus data

- The data look very untidy. Column names contain variables and years, and there are various superfluous columns, such as `Mark`, and last year. NA observations are coded as `n.a.` but are not recognized by are as such.
- The first thing we need to do is to change the NA obs to functioning NA obs, and remove some superfluous columns

```
#Step 1
amadeus[amadeus == "n.a."] <- NA
amadeus[amadeus == "n.s."] <- NA
amadeus <- amadeus[,-c(1,5:7)]
```

Using pivot to transform the data

- Next, we want to convert the data into a **long** format, using `pivot_longer`.

```
amadeus <- amadeus %>%
  mutate_all(as.character)

amadeus <- pivot_longer(data = amadeus,
                        cols = 4:64,
                        names_to = "variable",
                        values_to = "value")
```

Using pivot to transform the data

- This is what the data looks like now:

Table 10: The amadeus dataset

x
Company.name
City
Country.ISO.code
variable
value

String correction

- Now, we correct the strings using the following steps:

```r
#First, remove "th.EUR" from the string
amadeus$variable <- sub("\\.th.EUR.","",
                        amadeus$variable)
#Extract the year and put it into a new variable
amadeus$year <- as.numeric(
  str_extract(amadeus$variable, "[0-9]+"))

#Include only letters from the alphabet as variable names
amadeus$variable <- sapply(
  str_extract_all(
      amadeus$variable,"[aA-zZ]+"),
  paste, collapse = "_")
```

Pivoting...

- Finally, we can expand the table to adhere to the principal of one obs. per row, one variable per column.

```
amadeus <- pivot_wider(data = amadeus, names_from = variable,
                       values_from = value)

#There is only one problem, No. of directors/managers
#violates the tidy data principles.
nodir <- amadeus %>%
  filter(is.na(year))
amadeus <- amadeus %>%
  filter(!is.na(year))

amadeus <- merge(amadeus[,1:16],
                 nodir[,c(1,17)],
                 by = 1)

amadeus[,5:17] <- data.frame(sapply(amadeus[,5:17],
                                    function(x) as.numeric(x)))
```

Done!

- Now, the data looks like this:

Table 11: Tidy data

Company.name	year	Operating_revenue_Turnover
@JCG-GMAO-CONSULTING	2019	88
@JCG-GMAO-CONSULTING	2018	133
@JCG-GMAO-CONSULTING	2017	62
@JCG-GMAO-CONSULTING	2016	51
@JCG-GMAO-CONSULTING	2015	NA

- All other variables are located in the remaining columns.

Some analyses

Summarizing

- It seems that cleaning this type of data already required a fair amount of work, including some programming that requires knowledge of regular expressions, and some fairly non-standard transformation.
- In the remainder, I show some other examples of `pivot_longer` and `pivot_wider` in non-standard situations, so you do not get stuck when coming across these types of data.