**Data Formatting and Transformation [tidyr]**

pogRomcy danych (Data Masters) Season 1 / Episode 24

All rights reserved.  
The use shall be subject to prior consent of the author.

Press A to see the plain text instead of the slides.  
Press T to display the table of contents.

**What Is This Episode About?**

The data are usually presented in a tabular form. They can be, however, differently formatted in such table. There are, among others, a wide data representation, a narrow data representation, and the mixed representations.

Why such diversity? Well, it depends on what we want to do with the data. Sometimes one of their forms is better than the other. The package tidyr provides the functions to quickly and conveniently switch from one method of representation to another.

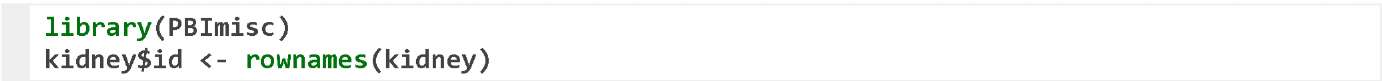
In this Episode, we will learn:

* How to switch from the narrow to wide representation.
* How to switch from the wide to narrow representation.

To illustrate these issues, we will use two sets of data. The first is the data set kidney of the package PBImisc, which will illustrate switching from wide to narrow format. The second data set includes the Eurostat data downloaded using the package PogromcyDanych. They will allow illustrating switching from narrow to wide data representation.

**Kidney Function Data**

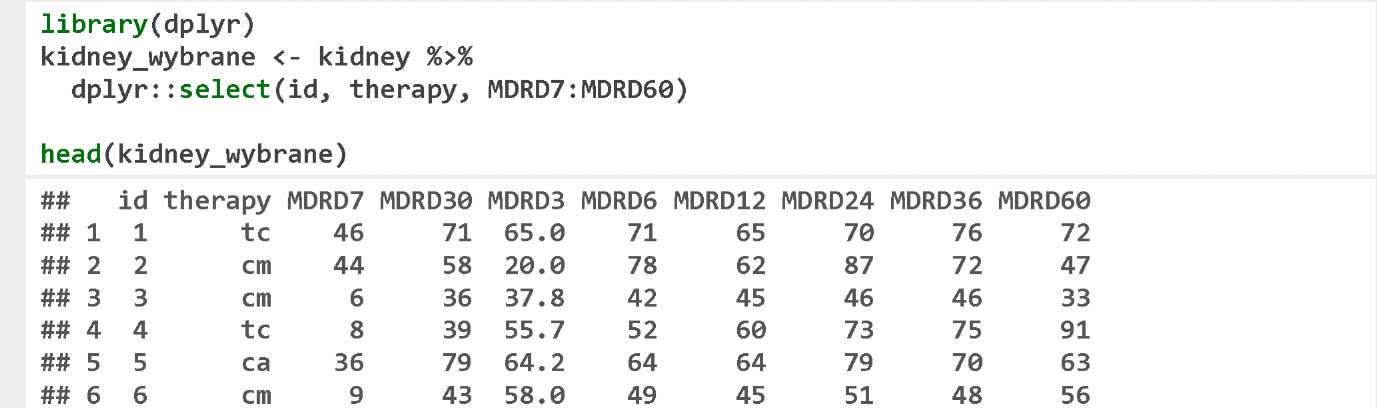
In order to illustrate what the wide data representation is and how to switch from it to the narrow representation, we will use the data set kidney of the package PBImisc. This package is not available with the basic version of the R. It needs to be installed using the function install.packages() (see the Episode 2 for details). Run this package after installation. To the data set kidney, add the column with the patient’s id.



Next, select several columns of your choice from the data set kidney . Each row in this set of data describes one patient. In the column therapy there is a qualitative variable with three levels, and in the columns MDRD7 to MDRD60, there are the kidney function measurements (measured with the MDRD) in 7, 30 days, and 3, 6, 12, 24, 36, and 60 months after the transplantation.

To select the variables of our interest, use the function select () of the package dplyr for your convenience.

Such a data representation, where the subsequent MDRD measurements are included in the subsequent columns, is called a wide representation. The more MDRD measuring points, the more columns in the set of data.



**Kidney Function Data – Narrow Form**

What is that narrow form? It is easier to explain with an example.

We will use the function gather() of the package tidyr. The first argument is the wide data frame. The next two arguments (in the below example they are time and MDRD) are the variable names, which will describe the keys and values in the new data set. The other arguments are a list of the columns to be converted to the narrow representation (this includes all MDRDs here...).

The result is a narrow representation, in which the successive MDRD measurements are represented by the different rows. In the narrow representation, in the column time, there is a name of the column from the original set of data, and in the column MDRD, there are the values.

The set of data kidney\_wybrane included 334 rows and ten columns and the set of data kidney\_waska contains 2,672 rows and four columns. Each of 334 patients is described by eight rows, each row relates to another MDRD measuring point.



**Kidney Function Data – Narrow Form**

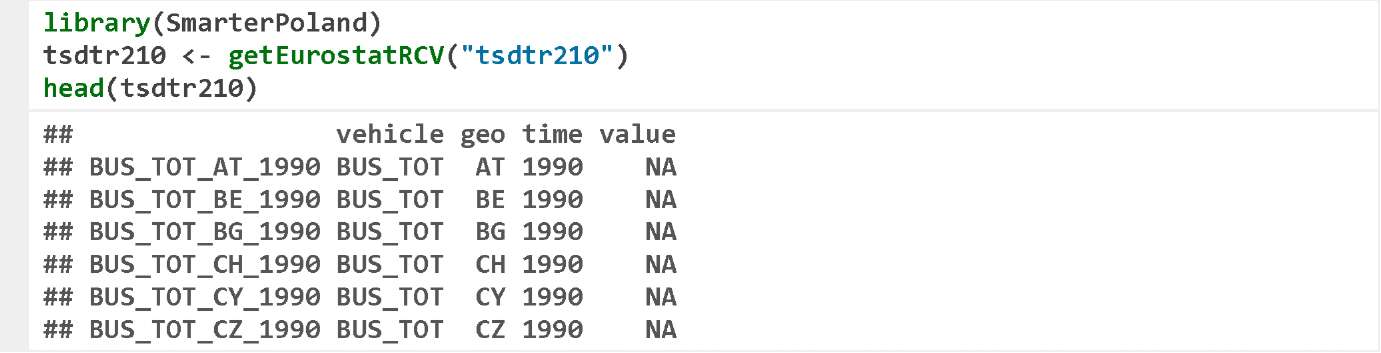
It should be noted that the variables not mentioned in the function gather() (here these include the variable id and therapy) will remain in the data frame and will not be converted into a narrow form.



**Eurostat Data**

To illustrate the narrow – wide transformation, we will use the Eurostat data. The Eurostat data are stored along with several dimensions. These data are easier to work with, if they are in a narrow form. However, as far as displaying these data on the screen is concerned, it is more convenient to convert them to the wide form.

To illustrate these functions, we will use the set of Eurostat data on the transport popularity in various countries. These data are contained in the table tsdtr210 available on the Eurostat servers. To download these data, we will use the function getEurostatRCV() of the package SmarterPoland.



The downloaded data are presented in a narrow form. The column geo defines a country, the column time defines a year, the column vehicle defines a mean of transport, and the column value defines the popularity of a given mean of transport in a defined country, and in a defined year.

**From Narrow to Wide Form**

In order to switch from the narrow to wide form, you can use the function spread().

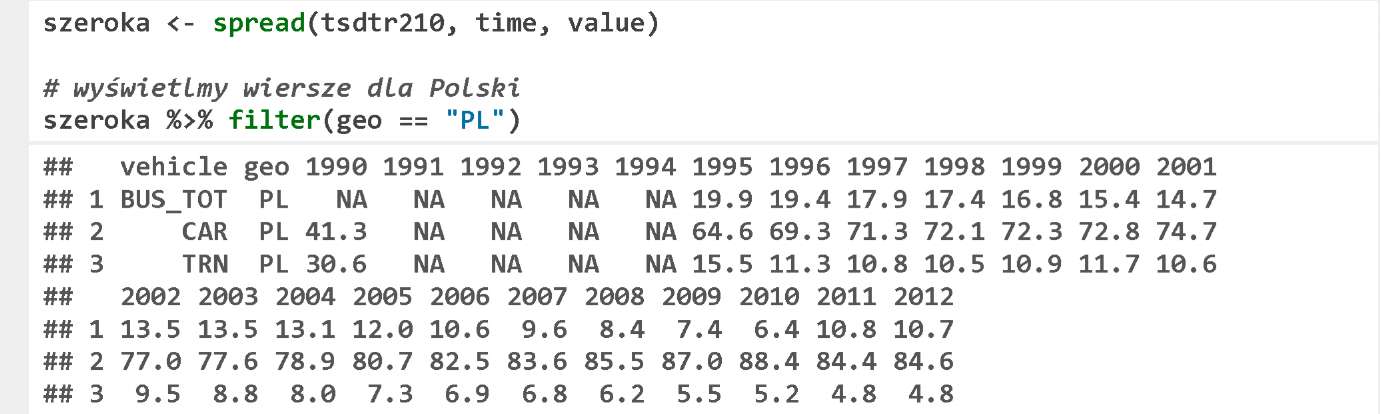
Its first argument is a frame containing the narrow form data. Two next variables define the columns, in which the keys and values are stored.

The function spread() converts the data frame in such a manner that the values of the second column of the input set of data become the names of the columns of the output set of data.

In the below example, the variable time is defined to be a key, so the values of this variable will become the names of the columns in the new set of data. The variable value is indicated as the values, and it will fill in the new columns.

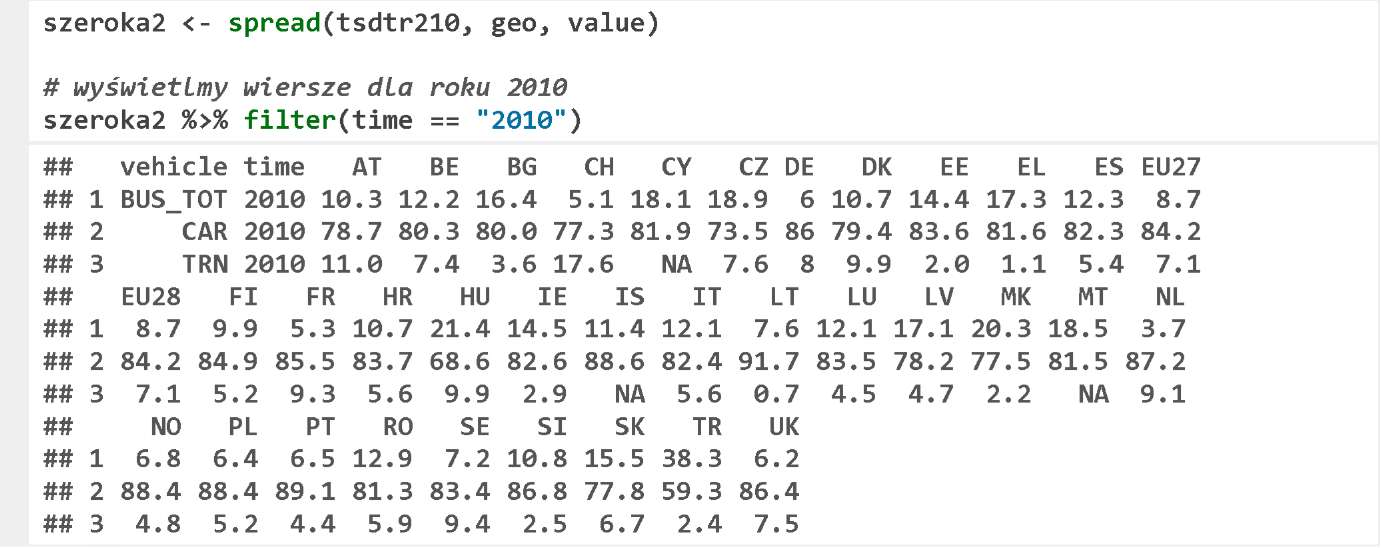
The other columns, in this case vehicle and geo, will remain unchanged.

Let’s highlight the rows that meet the condition geo == "PL". As we can see, the resulting set of data szeroka includes 13 columns describing the popularity of various means of transport in the subsequent years.



**From Narrow to Wide Form**

The Eurostat data featured more dimensions, and each of them can be used to create new columns. For example, we will use the variable geo to expand the data frame.

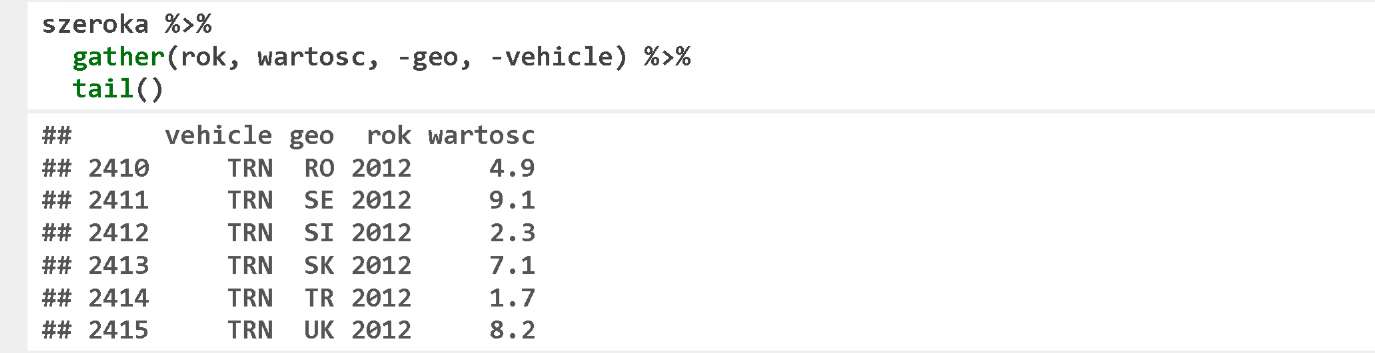


**From Wide to Narrow Form**

The process of switching from the narrow form to the wide form, and from the wide form to the narrow form is invertible. Let’s show how to re-switch from the wide to narrow form using the function gather().

Below the first argument is defined using the operator %>%, and the next two of them define the names of the new columns, in which the keys and values will be stored. The third argument is the description of the columns, which are to be transformed to the narrow form.

In the below example, the function gather() transforms the frame szeroka in such a manner that all columns except geo and vehicle (i.e. the columns including the year names) will be encoded with two keys, i.e. value (wartosc) and year (rok).



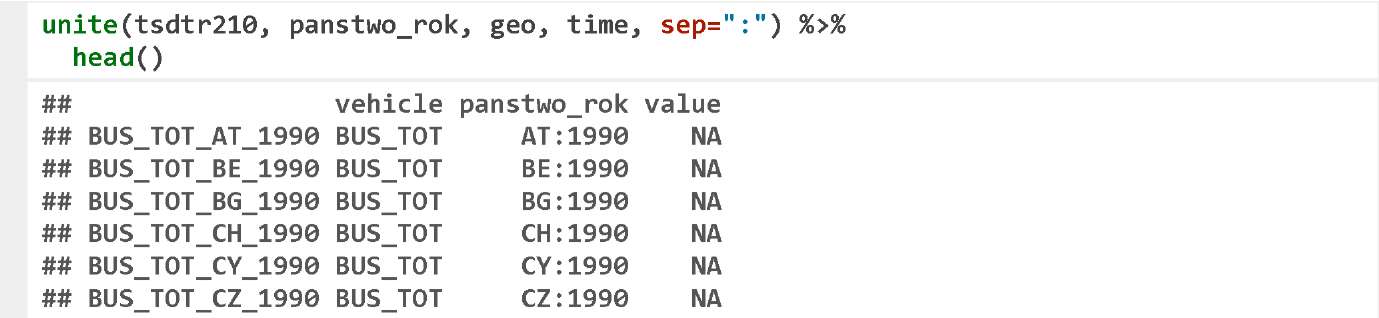
To display six exemplary rows, the function tail() was used here (it displays last six rows), as there are NA values in the first six rows.

**Column Pasting**

It happens that we would like to unite the values from several columns to create one column. It can be performed with the function unite().

The first argument of this function is a data frame. The second one is the name of a column, which will be created through uniting the columns, the others are the arguments. The argument sep defines what will separate the values in the united columns.

In the below set of data tsdtr210, we create a new column named panstwo\_rok, the values of which will be created by uniting the values in the columns geo and time by separating them with the character :.



**Column Separation**

The operation that is reverse to uniting, i.e. column separation, can be performed with the function separate(). We are going to present this function with an example of the artificial set of data including two columns – date (data) and id.



The first argument of the function separate() is the data frame to be converted. The second argument is the name of a column to be separated. The third argument is the vector of character strings, which will be the names of the new, separated columns. The fourth argument is a separator – a character, which will separate the columns. Below, we present an example, where the column daty of the set of data df will be divided into the values separated with -. The individual components will be year, month, and day.



**Exercises**

* The Eurostat database named prc\_ppp\_ind contains information on the average purchasing power. Read this data using the function getEurostatRCV(), and transform them from the narrow to the wide form.
* From the set of data kidney, select only the columns MDRD12, MDRD24, MDRD36, and MDRD60, and next convert them from the wide to narrow form.
* In the set of data kidney, the discrepancies in the antigens AB and DR are described by the columns discrepancy.AB and discrepancy.DR. Convert them into one column named discrepancy, where both discrepancies are pasted and separated with \_.

You can find the exemplary answers at https://rawgit.com/pbiecek/MOOC/master/0\_dane/9\_zadania.html.