Journal of Statistical Software

August 2014, Volume 59, Issue 10. *http://www.jstatsoft.org/*

Tidy Data

Hadley Wickham
RStudio

Abstract

A huge amount of effort is spent cleaning data to get it ready for analysis, but there has been little research on how to make data cleaning as easy and effective as possible. This paper tackles a small, but important, component of data cleaning: data tidying. Tidy datasets are easy to manipulate, model and visualize, and have a specific structure: each variable is a column, each observation is a row, and each type of observational unit is a table. This framework makes it easy to tidy messy datasets because only a small set of tools are needed to deal with a wide range of un-tidy datasets. This structure also makes it easier to develop tidy tools for data analysis, tools that both input and output tidy datasets. The advantages of a consistent data structure and matching tools are demonstrated with a case study free from mundane data manipulation chores.

Keywords: data cleaning, data tidying, relational databases, R.

1. Introduction

It is often said that 80% of data analysis is spent on the process of cleaning and preparing the data (Dasu and Johnson 2003). Data preparation is not just a first step, but must be repeated many times over the course of analysis as new problems come to light or new data is collected. Despite the amount of time it takes, there has been surprisingly little research on how to clean data well. Part of the challenge is the breadth of activities it encompasses: from outlier checking, to date parsing, to missing value imputation. To get a handle on the problem, this paper focuses on a small, but important, aspect of data cleaning that I call data *tidying*: structuring datasets to facilitate analysis.

The principles of tidy data provide a standard way to organize data values within a dataset. A standard makes initial data cleaning easier because you do not need to start from scratch and reinvent the wheel every time. The tidy data standard has been designed to facilitate initial exploration and analysis of the data, and to simplify the development of data analysis tools that work well together. Current tools often require translation. You have to spend time

munging the output from one tool so you can input it into another. Tidy datasets and tidy tools work hand in hand to make data analysis easier, allowing you to focus on the interesting domain problem, not on the uninteresting logistics of data.

The principles of tidy data are closely tied to those of relational databases and Codd's relational algebra (Codd 1990), but are framed in a language familiar to statisticians. Computer scientists have also contributed much to the study of data cleaning. For example, Lakshmanan, Sadri, and Subramanian (1996) define an extension to SQL to allow it to operate on messy datasets, Raman and Hellerstein (2001) provide a framework for cleaning datasets, and Kandel, Paepcke, Hellerstein, and Heer (2011) develop an interactive tool with a friendly user interface which automatically creates code to clean data. These tools are useful but they are presented in a language foreign to most statisticians, they fail to give much advice on how datasets should be structured, and they lack connections to the tools of data analysis.

The development of tidy data has been driven by my experience from working with real-world datasets. With few, if any, constraints on their organization, such datasets are often constructed in bizarre ways. I have spent countless hours struggling to get such datasets organized in a way that makes data analysis possible, let alone easy. I have also struggled to impart these skills to my students so they could tackle real-world datasets on their own. In the course of these struggles I developed the **reshape** and **reshape2** (Wickham 2007) packages. While I could intuitively use the tools and teach them through examples, I lacked the framework to make my intuition explicit. This paper provides that framework. It provides a comprehensive "philosophy of data": one that underlies my work in the **plyr** (Wickham 2011) and **ggplot2** (Wickham 2009) packages.

The paper proceeds as follows. Section 2 begins by defining the three characteristics that make a dataset tidy. Since most real world datasets are not tidy, Section 3 describes the operations needed to make messy datasets tidy, and illustrates the techniques with a range of real examples. Section 4 defines tidy tools, tools that input and output tidy datasets, and discusses how tidy data and tidy tools together can make data analysis easier. These principles are illustrated with a small case study in Section 5. Section 6 concludes with a discussion of what this framework misses and what other approaches might be fruitful to pursue.

2. Defining tidy data

> Happy families are all alike; every unhappy family is unhappy in its own way.
>
> Leo Tolstoy

Like families, tidy datasets are all alike but every messy dataset is messy in its own way. Tidy datasets provide a standardized way to link the structure of a dataset (its physical layout) with its semantics (its meaning). In this section, I will provide some standard vocabulary for describing the structure and semantics of a dataset, and then use those definitions to define tidy data.

	treatmenta	treatmentb
John Smith	—	2
Jane Doe	16	11
Mary Johnson	3	1

Table 1: Typical presentation dataset.

	John Smith	Jane Doe	Mary Johnson
treatmenta	—	16	3
treatmentb	2	11	1

Table 2: The same data as in Table 1 but structured differently.

2.1. Data structure

Most statistical datasets are rectangular tables made up of *rows* and *columns*. The columns are almost always labeled and the rows are sometimes labeled. Table 1 provides some data about an imaginary experiment in a format commonly seen in the wild. The table has two columns and three rows, and both rows and columns are labeled.

There are many ways to structure the same underlying data. Table 2 shows the same data as Table 1, but the rows and columns have been transposed. The data is the same, but the layout is different. Our vocabulary of rows and columns is simply not rich enough to describe why the two tables represent the same data. In addition to appearance, we need a way to describe the underlying semantics, or meaning, of the values displayed in tables.

2.2. Data semantics

A dataset is a collection of *values*, usually either numbers (if quantitative) or strings (if qualitative). Values are organized in two ways. Every value belongs to a *variable* and an *observation*. A variable contains all values that measure the same underlying attribute (like height, temperature, duration) across units. An observation contains all values measured on the same unit (like a person, or a day, or a race) across attributes.

Table 3 reorganizes Table 1 to make the values, variables and observations more clear. The dataset contains 18 values representing three variables and six observations. The variables are:

1. `person`, with three possible values (John Smith, Mary Johnson, and Jane Doe).
2. `treatment`, with two possible values (a and b).
3. `result`, with five or six values depending on how you think of the missing value (—, 16, 3, 2, 11, 1).

The experimental design tells us more about the structure of the observations. In this experiment, every combination of `person` and `treatment` was measured, a completely crossed design. The experimental design also determines whether or not missing values can be safely dropped. In this experiment, the missing value represents an observation that should have

person	treatment	result
John Smith	a	—
Jane Doe	a	16
Mary Johnson	a	3
John Smith	b	2
Jane Doe	b	11
Mary Johnson	b	1

Table 3: The same data as in Table 1 but with variables in columns and observations in rows.

been made, but was not, so it is important to keep it. Structural missing values, which represent measurements that cannot be made (e.g., the count of pregnant males) can be safely removed.

For a given dataset, it is usually easy to figure out what are observations and what are variables, but it is surprisingly difficult to precisely define variables and observations in general. For example, if the columns in the Table 1 were `height` and `weight` we would have been happy to call them variables. If the columns were `height` and `width`, it would be less clear cut, as we might think of height and width as values of a `dimension` variable. If the columns were `home phone` and `work phone`, we could treat these as two variables, but in a fraud detection environment we might want variables `phone number` and `number type` because the use of one phone number for multiple people might suggest fraud. A general rule of thumb is that it is easier to describe functional relationships between variables (e.g., `z` is a linear combination of `x` and `y`, `density` is the ratio of `weight` to `volume`) than between rows, and it is easier to make comparisons between groups of observations (e.g., average of group a vs. average of group b) than between groups of columns.

In a given analysis, there may be multiple levels of observations. For example, in a trial of new allergy medication we might have three observational types: demographic data collected from each person (`age`, `sex`, `race`), medical data collected from each person on each day (`number of sneezes`, `redness of eyes`), and meteorological data collected on each day (`temperature`, `pollen count`).

2.3. Tidy data

Tidy data is a standard way of mapping the meaning of a dataset to its structure. A dataset is messy or tidy depending on how rows, columns and tables are matched up with observations, variables and types. 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 is Codd's 3rd normal form (Codd 1990), but with the constraints framed in statistical language, and the focus put on a single dataset rather than the many connected datasets common in relational databases. *Messy data* is any other arrangement of the data.

Table 3 is the tidy version of Table 1. Each row represents an observation, the `result` of one `treatment` on one `person`, and each column is a variable.

Tidy data makes it easy for an analyst or a computer to extract needed variables because it provides a standard way of structuring a dataset. Compare Table 3 to Table 1: in Table 1 you need to use different strategies to extract different variables. This slows analysis and invites errors. If you consider how many data analysis operations involve all of the values in a variable (every aggregation function), you can see how important it is to extract these values in a simple, standard way. Tidy data is particularly well suited for vectorized programming languages like R (R Core Team 2014), because the layout ensures that values of different variables from the same observation are always paired.

While the order of variables and observations does not affect analysis, a good ordering makes it easier to scan the raw values. One way of organizing variables is by their role in the analysis: are values fixed by the design of the data collection, or are they measured during the course of the experiment? Fixed variables describe the experimental design and are known in advance. Computer scientists often call fixed variables dimensions, and statisticians usually denote them with subscripts on random variables. Measured variables are what we actually measure in the study. Fixed variables should come first, followed by measured variables, each ordered so that related variables are contiguous. Rows can then be ordered by the first variable, breaking ties with the second and subsequent (fixed) variables. This is the convention adopted by all tabular displays in this paper.

3. Tidying messy datasets

Real datasets can, and often do, violate the three precepts of tidy data in almost every way imaginable. While occasionally you do get a dataset that you can start analyzing immediately, this is the exception, not the rule. This section describes the five most common problems with messy datasets, along with their remedies:

- Column headers are values, not variable names.
- Multiple variables are stored in one column.
- Variables are stored in both rows and columns.
- Multiple types of observational units are stored in the same table.
- A single observational unit is stored in multiple tables.

Surprisingly, most messy datasets, including types of messiness not explicitly described above, can be tidied with a small set of tools: melting, string splitting, and casting. The following sections illustrate each problem with a real dataset that I have encountered, and show how to tidy them. The complete datasets and the R code used to tidy them are available online at `https://github.com/hadley/tidy-data`, and in the online supplementary materials for this paper.

3.1. Column headers are values, not variable names

A common type of messy dataset is tabular data designed for presentation, where variables form both the rows and columns, and column headers are values, not variable names. While

religion	<$10k	$10–20k	$20–30k	$30–40k	$40–50k	$50–75k
Agnostic	27	34	60	81	76	137
Atheist	12	27	37	52	35	70
Buddhist	27	21	30	34	33	58
Catholic	418	617	732	670	638	1116
Don't know/refused	15	14	15	11	10	35
Evangelical Prot	575	869	1064	982	881	1486
Hindu	1	9	7	9	11	34
Historically Black Prot	228	244	236	238	197	223
Jehovah's Witness	20	27	24	24	21	30
Jewish	19	19	25	25	30	95

Table 4: The first ten rows of data on income and religion from the Pew Forum. Three columns, $75–100k, $100–150k and >150k, have been omitted.

row	a	b	c
A	1	4	7
B	2	5	8
C	3	6	9

(a) Raw data

row	column	value
A	a	1
B	a	2
C	a	3
A	b	4
B	b	5
C	b	6
A	c	7
B	c	8
C	c	9

(b) Molten data

Table 5: A simple example of melting. (a) is melted with one colvar, row, yielding the molten dataset (b). The information in each table is exactly the same, just stored in a different way.

I would call this arrangement messy, in some cases it can be extremely useful. It provides efficient storage for completely crossed designs, and it can lead to extremely efficient computation if desired operations can be expressed as matrix operations. This issue is discussed in depth in Section 6.

Table 4 shows a subset of a typical dataset of this form. This dataset explores the relationship between income and religion in the US. It comes from a report[1] produced by the Pew Research Center, an American think-tank that collects data on attitudes to topics ranging from religion to the internet, and produces many reports that contain datasets in this format.

This dataset has three variables, `religion`, `income` and `frequency`. To tidy it, we need to *melt*, or stack it. In other words, we need to turn columns into rows. While this is often described as making wide datasets long or tall, I will avoid those terms because they are imprecise. Melting is parameterized by a list of columns that are already variables, or *colvar*s for short. The other columns are converted into two variables: a new variable called `column`

[1] `http://religions.pewforum.org/pdf/comparison-Income%20Distribution%20of%20Religious%20Traditions.pdf`

religion	income	freq
Agnostic	<$10k	27
Agnostic	$10–20k	34
Agnostic	$20–30k	60
Agnostic	$30–40k	81
Agnostic	$40–50k	76
Agnostic	$50–75k	137
Agnostic	$75–100k	122
Agnostic	$100–150k	109
Agnostic	>150k	84
Agnostic	Don't know/refused	96

Table 6: The first ten rows of the tidied Pew survey dataset on income and religion. The `column` has been renamed to `income`, and `value` to `freq`.

year	artist	track	time	date.entered	wk1	wk2	wk3
2000	2 Pac	Baby Don't Cry	4:22	2000-02-26	87	82	72
2000	2Ge+her	The Hardest Part Of ...	3:15	2000-09-02	91	87	92
2000	3 Doors Down	Kryptonite	3:53	2000-04-08	81	70	68
2000	98^0	Give Me Just One Nig...	3:24	2000-08-19	51	39	34
2000	A*Teens	Dancing Queen	3:44	2000-07-08	97	97	96
2000	Aaliyah	I Don't Wanna	4:15	2000-01-29	84	62	51
2000	Aaliyah	Try Again	4:03	2000-03-18	59	53	38
2000	Adams, Yolanda	Open My Heart	5:30	2000-08-26	76	76	74

Table 7: The first eight Billboard top hits for 2000. Other columns not shown are `wk4`, `wk5`, ..., `wk75`.

that contains repeated column headings and a new variable called `value` that contains the concatenated data values from the previously separate columns. This is illustrated in Table 5 with a toy dataset. The result of melting is a *molten* dataset.

The Pew dataset has one colvar, `religion`, and melting yields Table 6. To better reflect their roles in this dataset, the `variable` column has been renamed to `income`, and the `value` column to `freq`. This form is tidy because each column represents a variable and each row represents an observation, in this case a demographic unit corresponding to a combination of `religion` and `income`.

Another common use of this data format is to record regularly spaced observations over time. For example, the Billboard dataset shown in Table 7 records the date a song first entered the Billboard Top 100. It has variables for `artist`, `track`, `date.entered`, `rank` and `week`. The rank in each week after it enters the top 100 is recorded in 75 columns, `wk1` to `wk75`. If a song is in the Top 100 for less than 75 weeks the remaining columns are filled with missing values. This form of storage is not tidy, but it is useful for data entry. It reduces duplication since otherwise each song in each week would need its own row, and song metadata like title and artist would need to be repeated. This issue will be discussed in more depth in Section 3.4.

This dataset has colvars `year`, `artist`, `track`, `time`, and `date.entered`. Melting yields Table 8. I have also done a little cleaning as well as tidying: `column` has been converted to

year	artist	time	track	date	week	rank
2000	2 Pac	4:22	Baby Don't Cry	2000-02-26	1	87
2000	2 Pac	4:22	Baby Don't Cry	2000-03-04	2	82
2000	2 Pac	4:22	Baby Don't Cry	2000-03-11	3	72
2000	2 Pac	4:22	Baby Don't Cry	2000-03-18	4	77
2000	2 Pac	4:22	Baby Don't Cry	2000-03-25	5	87
2000	2 Pac	4:22	Baby Don't Cry	2000-04-01	6	94
2000	2 Pac	4:22	Baby Don't Cry	2000-04-08	7	99
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-02	1	91
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-09	2	87
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-16	3	92
2000	3 Doors Down	3:53	Kryptonite	2000-04-08	1	81
2000	3 Doors Down	3:53	Kryptonite	2000-04-15	2	70
2000	3 Doors Down	3:53	Kryptonite	2000-04-22	3	68
2000	3 Doors Down	3:53	Kryptonite	2000-04-29	4	67
2000	3 Doors Down	3:53	Kryptonite	2000-05-06	5	66

Table 8: First fifteen rows of the tidied Billboard dataset. The `date` column does not appear in the original table, but can be computed from `date.entered` and `week`.

country	year	m014	m1524	m2534	m3544	m4554	m5564	m65	mu	f014
AD	2000	0	0	1	0	0	0	0	—	—
AE	2000	2	4	4	6	5	12	10	—	3
AF	2000	52	228	183	149	129	94	80	—	93
AG	2000	0	0	0	0	0	0	1	—	1
AL	2000	2	19	21	14	24	19	16	—	3
AM	2000	2	152	130	131	63	26	21	—	1
AN	2000	0	0	1	2	0	0	0	—	0
AO	2000	186	999	1003	912	482	312	194	—	247
AR	2000	97	278	594	402	419	368	330	—	121
AS	2000	—	—	—	—	1	1	—	—	—

Table 9: Original TB dataset. Corresponding to each 'm' column for males, there is also an 'f' column for females, `f1524`, `f2534` and so on. These are not shown to conserve space. Note the mixture of 0s and missing values (—). This is due to the data collection process and the distinction is important for this dataset.

`week` by extracting the number, and `date` has been computed from `date.entered` and `week`.

3.2. Multiple variables stored in one column

After melting, the `column` variable names often becomes a combination of multiple underlying variable names. This is illustrated by the tuberculosis (TB) dataset, a sample of which is shown in Table 9. This dataset comes from the World Health Organization, and records the counts of confirmed tuberculosis cases by `country`, `year`, and demographic group. The demographic groups are broken down by `sex` (m, f) and `age` (0–14, 15–25, 25–34, 35–44, 45–54, 55–64, unknown).

country	year	column	cases
AD	2000	m014	0
AD	2000	m1524	0
AD	2000	m2534	1
AD	2000	m3544	0
AD	2000	m4554	0
AD	2000	m5564	0
AD	2000	m65	0
AE	2000	m014	2
AE	2000	m1524	4
AE	2000	m2534	4
AE	2000	m3544	6
AE	2000	m4554	5
AE	2000	m5564	12
AE	2000	m65	10
AE	2000	f014	3

(a) Molten data

country	year	sex	age	cases
AD	2000	m	0–14	0
AD	2000	m	15–24	0
AD	2000	m	25–34	1
AD	2000	m	35–44	0
AD	2000	m	45–54	0
AD	2000	m	55–64	0
AD	2000	m	65+	0
AE	2000	m	0–14	2
AE	2000	m	15–24	4
AE	2000	m	25–34	4
AE	2000	m	35–44	6
AE	2000	m	45–54	5
AE	2000	m	55–64	12
AE	2000	m	65+	10
AE	2000	f	0-14	3

(b) Tidy data

Table 10: Tidying the TB dataset requires first melting, and then splitting the `column` column into two variables: `sex` and `age`.

Column headers in this format are often separated by some character (`.`, `-`, `_`, `:`). While the string can be broken into pieces using that character as a divider, in other cases, such as for this dataset, more careful string processing is required. For example, the variable names can be matched to a lookup table that converts single compound value into multiple component values.

Table 10(a) shows the results of melting the TB dataset, and Table 10(b) shows the results of splitting the single column `column` into two real variables: `age` and `sex`.

Storing the values in this form resolves another problem in the original data. We want to compare rates, not counts. But to compute rates, we need to know the population. In the original format, there is no easy way to add a population variable. It has to be stored in a separate table, which makes it hard to correctly match populations to counts. In tidy form, adding variables for population and rate is easy. They are just additional columns.

3.3. Variables are stored in both rows and columns

The most complicated form of messy data occurs when variables are stored in both rows and columns. Table 11 shows daily weather data from the Global Historical Climatology Network for one weather station (MX17004) in Mexico for five months in 2010. It has variables in individual columns (`id`, `year`, `month`), spread across columns (`day`, d1–d31) and across rows (`tmin`, `tmax`) (minimum and maximum temperature). Months with less than 31 days have structural missing values for the last day(s) of the month. The `element` column is not a variable; it stores the names of variables.

To tidy this dataset we first melt it with colvars `id`, `year`, `month` and the column that contains variable names, `element`. This yields Table 12(a). For presentation, we have dropped the

id	year	month	element	d1	d2	d3	d4	d5	d6	d7	d8
MX17004	2010	1	tmax	—	—	—	—	—	—	—	—
MX17004	2010	1	tmin	—	—	—	—	—	—	—	—
MX17004	2010	2	tmax	—	27.3	24.1	—	—	—	—	—
MX17004	2010	2	tmin	—	14.4	14.4	—	—	—	—	—
MX17004	2010	3	tmax	—	—	—	—	32.1	—	—	—
MX17004	2010	3	tmin	—	—	—	—	14.2	—	—	—
MX17004	2010	4	tmax	—	—	—	—	—	—	—	—
MX17004	2010	4	tmin	—	—	—	—	—	—	—	—
MX17004	2010	5	tmax	—	—	—	—	—	—	—	—
MX17004	2010	5	tmin	—	—	—	—	—	—	—	—

Table 11: Original weather dataset. There is a column for each possible day in the month. Columns d9 to d31 have been omitted to conserve space.

id	date	element	value
MX17004	2010-01-30	tmax	27.8
MX17004	2010-01-30	tmin	14.5
MX17004	2010-02-02	tmax	27.3
MX17004	2010-02-02	tmin	14.4
MX17004	2010-02-03	tmax	24.1
MX17004	2010-02-03	tmin	14.4
MX17004	2010-02-11	tmax	29.7
MX17004	2010-02-11	tmin	13.4
MX17004	2010-02-23	tmax	29.9
MX17004	2010-02-23	tmin	10.7

(a) Molten data

id	date	tmax	tmin
MX17004	2010-01-30	27.8	14.5
MX17004	2010-02-02	27.3	14.4
MX17004	2010-02-03	24.1	14.4
MX17004	2010-02-11	29.7	13.4
MX17004	2010-02-23	29.9	10.7
MX17004	2010-03-05	32.1	14.2
MX17004	2010-03-10	34.5	16.8
MX17004	2010-03-16	31.1	17.6
MX17004	2010-04-27	36.3	16.7
MX17004	2010-05-27	33.2	18.2

(b) Tidy data

Table 12: (a) Molten weather dataset. This is almost tidy, but instead of values, the `element` column contains names of variables. Missing values are dropped to conserve space. (b) Tidy weather dataset. Each row represents the meteorological measurements for a single day. There are two measured variables, minimum (`tmin`) and maximum (`tmax`) temperature; all other variables are fixed.

missing values, making them implicit rather than explicit. This is permissible because we know how many days are in each month and can easily reconstruct the explicit missing values.

This dataset is mostly tidy, but we have two variables stored in rows: `tmin` and `tmax`, the type of observation. Not shown in this example are the other meteorological variables `prcp` (precipitation) and `snow` (snowfall). Fixing the issue with the type of observation requires the cast, or unstack, operation. This performs the inverse of melting by rotating the `element` variable back out into the columns (Table 12(b)). This form is tidy. There is one variable in each column, and each row represents a day's observations. The cast operation is described in depth in Wickham (2007).

id	artist	track	time
1	2 Pac	Baby Don't Cry	4:22
2	2Ge+her	The Hardest Part Of ...	3:15
3	3 Doors Down	Kryptonite	3:53
4	3 Doors Down	Loser	4:24
5	504 Boyz	Wobble Wobble	3:35
6	98^0	Give Me Just One Nig...	3:24
7	A*Teens	Dancing Queen	3:44
8	Aaliyah	I Don't Wanna	4:15
9	Aaliyah	Try Again	4:03
10	Adams, Yolanda	Open My Heart	5:30
11	Adkins, Trace	More	3:05
12	Aguilera, Christina	Come On Over Baby	3:38
13	Aguilera, Christina	I Turn To You	4:00
14	Aguilera, Christina	What A Girl Wants	3:18
15	Alice Deejay	Better Off Alone	6:50

id	date	rank
1	2000-02-26	87
1	2000-03-04	82
1	2000-03-11	72
1	2000-03-18	77
1	2000-03-25	87
1	2000-04-01	94
1	2000-04-08	99
2	2000-09-02	91
2	2000-09-09	87
2	2000-09-16	92
3	2000-04-08	81
3	2000-04-15	70
3	2000-04-22	68
3	2000-04-29	67
3	2000-05-06	66

Table 13: Normalized Billboard dataset split up into song dataset (left) and rank dataset (right). First 15 rows of each dataset shown; `genre` omitted from song dataset, `week` omitted from rank dataset.

3.4. Multiple types in one table

Datasets often involve values collected at multiple levels, on different types of observational units. During tidying, each type of observational unit should be stored in its own table. This is closely related to the idea of database normalization, where each fact is expressed in only one place. If this is not done, it is possible for inconsistencies to occur.

The Billboard dataset described in Table 8 actually contains observations on two types of observational units: the song and its rank in each week. This manifests itself through the duplication of facts about the song: `artist` and `time` are repeated for every song in each week. The Billboard dataset needs to be broken down into two datasets: a song dataset which stores `artist`, `song name` and `time`, and a ranking dataset which gives the `rank` of the `song` in each `week`. Table 13 shows these two datasets. You could also imagine a week dataset which would record background information about the week, maybe the total number of songs sold or similar demographic information.

Normalization is useful for tidying and eliminating inconsistencies. However, there are few data analysis tools that work directly with relational data, so analysis usually also requires denormalization or merging the datasets back into one table.

3.5. One type in multiple tables

It is also common to find data values about a single type of observational unit spread out over multiple tables or files. These tables and files are often split up by another variable, so that each represents a single year, person, or location. As long as the format for individual records is consistent, this is an easy problem to fix:

1. Read the files into a list of tables.

2. For each table, add a new column that records the original file name (because the file name is often the value of an important variable).

3. Combine all tables into a single table.