Coding Grace - Workshop 2016

Tidy Data with R

Tidy datasets are all alike but every messy dataset is messy in its own way. Hadley Wickham

Data science, at its heart, is a computer programming exercise. Data scientists use computers to store, transform, visualize, and model their data. As a result, every data science project begins with the same task: you must prepare your data to use it with a computer.

- **dplyr** - data manipulation
- **magrittr** - pipe operator

Tidy Data

Abstract from **dplyr** talk

- To make the most of dplyr, Hadley Wickham recommends that you familiarise yourself with the **principles of tidy data**.
- This will help you get your data into a form that works well with **dplyr**, **ggplot2** and `R`'s many modelling functions.

Tidy Data With R

- **tidyr** is a reframing of reshape2 designed to accompany the tidy data framework, and to work hand-in-hand with **magrittr** and **dplyr** to build a solid pipeline for data analysis.
  (from Hadley Wickham's abstract)

Principles of Tidy Data

- Tidy data was popularized by Hadley Wickham, and it serves as the basis for many R packages and functions.
- You can learn more about tidy data by reading **Tidy Data** a paper written by Hadley Wickham and published in the Journal of Statistical Software.
- Tidy Data is available online at `www.jstatsoft.org/v59/i10/paper`.

Tidy Data

Three Principles from Hadley Wickham's paper

1. Each variable forms a column,
2. Each observation forms a row,
3. Each table/file stores data about one kind of observation.

Remark:
The paper "**Tidy Data**" by Hadley Wickham (RStudio) can be downloaded from

`http://vita.had.co.nz/papers/tidy-data.pdf`

Tidy Data with R

R follows a set of conventions that makes one layout of tabular data much easier to work with than others.

- **Each column is a variable:** Each variable in the data set is placed in its **own** column
- **Each row is an observation:** Each observation is placed in its **own** row
- Each value is placed in its own cell

Data that satisfies these rules is known as **tidy data**.

Tidy Data with R

Dataframes (if you are not familiar)

- A data frame is a list of vectors that R displays as a table.
- When your data is tidy, the values of each variable fall in their own column vector.

```
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
>
```

tidyr: Easily Tidy Data with 'spread()' and 'gather()' Functions

An evolution of 'reshape2'. It's designed specifically for data tidying (not general reshaping or aggregating) and works well with 'dplyr' data pipelines.

Version:	0.3.1
Depends:	R (≥ 3.1.0)
Imports:	dplyr (≥ 0.4), stringi, lazyeval, magrittr, Rcpp
LinkingTo:	Rcpp
Suggests:	knitr, testthat, data.table, covr
Published:	2015-09-10
Author:	Hadley Wickham [aut, cre], RStudio [cph]
Maintainer:	Hadley Wickham <hadley at rstudio.com>
BugReports:	https://github.com/hadley/tidyr/issues
License:	MIT + file LICENSE
URL:	https://github.com/hadley/tidyr
NeedsCompilation:	yes
Materials:	README
CRAN checks:	tidyr results

tidyR

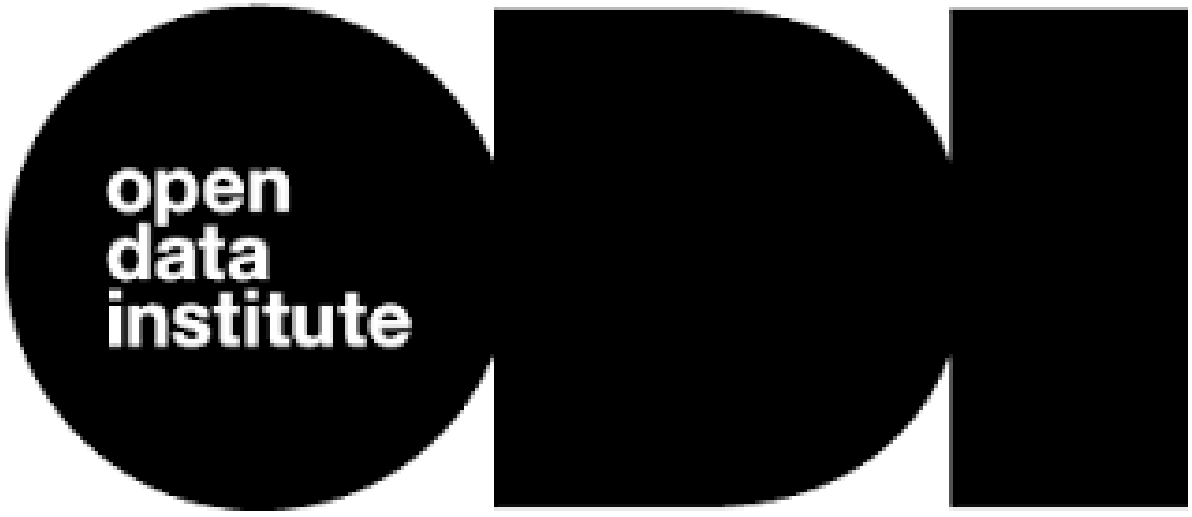
Abstract for tidyr

- tidyr is new package that makes it easy to "tidy your data.
- Tidy data is data thats easy to work with: its easy to munge (with **dplyr**), visualise (with **ggplot2** or **ggvis**) and model (with R's hundreds of modelling packages).

tidyr's verbs

- **tidyr** provides four main functions for tidying your messy data: `gather()`, `separate()`, `spread()` and `unite()`.

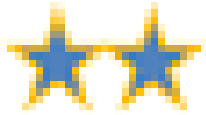
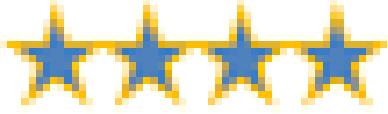
```
gather()
spread()
separate()
unite()
```


open
data
institute

“Open data and a change of mindset is the next step in the internet revolution”

- Sir Tim Berners-Lee

To Get to 5-Stars With Open Data

Star	Definition	Example / Tool*
	Make your stuff available on the Web (whatever format) under an open license	Web Story / MindTouch
	Make it available as structured data (e.g., Excel instead of image scan of a table)	Spreadsheet / Excel
	Use non-proprietary formats (e.g., CSV instead of Excel)	Table / MindTouch and Spotfire
	Use URIs to identify things, so that people can point at your stuff	Table of Contents / MindTouch and Spotfire
	Link your data to other data to provide context	Table / MindTouch and Spotfire

* Examples of tools used.

Source of Star and Definition: http://www.w3.org/DesignIssues/LinkedData.html

Tidy Data with R

- *Data Science with R* by Garrett Grolemund
- Chapter: Data Tidying

DSR Example Data Sets

- The data sets in the DSR Package show the same data, but organized in four entirely different ways.
- Each data set shows the same values of four variables country, year, population, and cases, but each data set organizes the values into a different layout.

Tidy Data with R

- To install DSR, run the command

```
install.packages(c("tidyr", "devtools"))

devtools::install_github("garrettgman/DSR")
```

Tidy Data with R- Data Set 1

```
library(DSR)
# Data set one
table1
## Source: local data frame [6 x 4]
##
##       country year  cases population
## 1 Afghanistan 1999    745   19987071
## 2 Afghanistan 2000   2666   20595360
## 3      Brazil 1999  37737  172006362
## 4      Brazil 2000  80488  174504898
## 5       China 1999 212258 1272915272
## 6       China 2000 213766 1280428583
```

Tidy Data with R

- In `table1` (previous slide), each variable is placed in its own column, each observation in its own row, and each value in its own cell.
- It properly complies with the *Principles of Tidy Data*.

- On the next set of slides - we will look at three more layouts (with the last layout comprising two separate tables)

Table 2 (sed)

```
# Data set two
table2

## Source: local data frame [12 x 4]
##
##         country year         key      value
## 1   Afghanistan 1999       cases        745
## 2   Afghanistan 1999 population   19987071
## 3   Afghanistan 2000       cases       2666
## 4   Afghanistan 2000 population   20595360
## 5        Brazil 1999       cases      37737
## 6        Brazil 1999 population  172006362
..........
```

Tidy Data with R

```
# Data set three
table3

## Source: local data frame [6 x 3]
##
##       country year              rate
## 1 Afghanistan 1999      745/19987071
## 2 Afghanistan 2000     2666/20595360
## 3      Brazil 1999   37737/172006362
## 4      Brazil 2000   80488/174504898
## 5       China 1999 212258/1272915272
## 6       China 2000 213766/1280428583
```

Tidy Data with R

The last data set is a collection of two tables: cases and populations

```
# Data set four
table4  # cases

## Source: local data frame [3 x 3]
##
##       country   1999   2000
## 1 Afghanistan    745   2666
## 2      Brazil  37737  80488
## 3       China 212258 213766
```

Tidy Data with R

```
table5  # population

## Source: local data frame [3 x 3]
##
##       country        1999        2000
## 1 Afghanistan    19987071    20595360
## 2      Brazil   172006362   174504898
## 3       China  1272915272  1280428583
```

Using this data for analysis

- Assume that in these data sets, cases refers to the number of people diagnosed with TB per country per year.
- To calculate the rate of TB cases per country per year (i.e, the number of people per 10,000 diagnosed with TB), you will need to do four separate approach with the data, one for each layout.

Tidy Data with R

Each approach will do the following

1. Extract the number of TB cases per country per year
2. Extract the population per country per year (in the same order as above)
3. Divide cases by population
4. Multiply by 10000

Data set one

Since *table1* is organized in a tidy fashion, you can calculate the rate like this,

```
# Data set one
table1$cases / table1$population * 10000
```

Quick and Relatively Simple

Data set two

- Data set two intermingles the values of population and cases in the same columns.
- As a result, you will need to untangle the values whenever you want to work with each variable separately.
- Youll need to perform an extra step to calculate the rate.

Tidy Data with R

```
# Data set two
case_rows <- c(1, 3, 5, 7, 9, 11, 13, 15, 17
pop_rows <- c(2, 4, 6, 8, 10, 12, 14, 16, 18

table2$value[case_rows]
   / table2$value[pop_rows] * 10000
```

Not overly complicated, but requires specification of rows (may or may not be automatable)

Data set three

- Data set three combines the values of cases and population into the same cells.
- It may seem that this would help you calculate the rate, but that is not so.
- You will need to separate the population values from the cases values if you wish to conduct an analysis them.
- This can be done, but not with basic R syntax.

Data set four

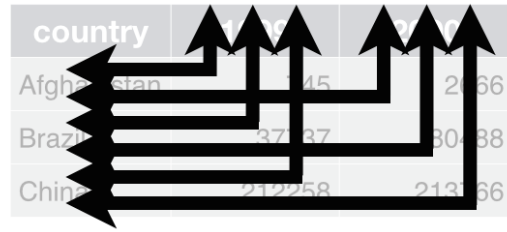
- Data set four stores each variable in a different format: as a column, a set of column names, or a field of cells.
- As a result, you will need to work with each variable differently.

Data set four

- This makes code written for data set four hard to generalize.
- The code that extracts the values of year, `names(table4)[-1]`, cannot be generalized to extract the values of population, `c(table5$1999, table5$2000, table5$2001)`.

country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766

table4



country	1999	2000
Afghanistan	19987071	20595360
Brazil	172006362	174504898
China	1272915272	1280428583

table5

variables

observations

Tidy Data with R

- The organization of data set four is inefficient in a second way as well.
- Data set four separates the values of some variables across two separate tables.
- This is inconvenient because you will need to extract information from two different places whenever you want to work with the data.

Tidy Data with R

- The **tidyr** package by Hadley Wickham is designed to help you tidy your data.
- **tidyr** contains four functions that alter the layout of tabular data sets, while preserving the values and relationships contained in the data sets.
- The two most important functions in tidyr are `gather()` and `spread()`.
- Each relies on the idea of a key value pair.

Tidy Data with R

- A key value pair is a simple way to record information.
- A pair contains two parts: a **key** that explains what the information describes, and a **value** that contains the actual information.

```
Password: 0123456789
```

- *0123456789* is the **value**, and it is associated with the **key** *Password*.

Tidy Data with R

- You could decompose table1 into a group of key value pairs, but it would cease to be a useful data set because you no longer know which values belong to the same observation (next slides).
- In tidy data, each cell will contain a value and each column name will contain a key, but this doesnt need to be the case for untidy data.

Tidy Data with R

```
Country: Afghanistan
Country: Brazil
Country: China
Year: 1999
Year: 2000
Year: 2001
```

Tidy Data with R

```
Population:   19987071
Population:   20595360
Population:  172006362
Population:  174504898
Population: 1272915272
Population: 1280428583
Cases:     745
Cases:    2666
Cases:   37737
Cases:   80488
Cases:  212258
Cases:  213766
```

Tidy Data with R

```
## Source: local data frame [12 x 4]
##
##         country year         key      value
## 1   Afghanistan 1999       cases        745
## 2   Afghanistan 1999 population   19987071
## 3   Afghanistan 2000       cases       2666
## 4   Afghanistan 2000 population   20595360
## 5        Brazil 1999       cases      37737
## 6        Brazil 1999 population  172006362
## 7        Brazil 2000       cases      80488
## 8        Brazil 2000 population  174504898
## 9         China 1999       cases     212258
## 10        China 1999 population 1272915272
## 11        China 2000       cases     213766
## 12        China 2000 population 1280428583
```

spread()

- In table2, the key column contains only keys (and not just because the column is labelled key).
- Conveniently, the value column contains the values associated with those keys.
- You can use the `spread()` function to tidy this layout.

Tidy Data with R

`spread()`

- `spread()` turns a pair of key:value columns into a set of tidy columns.
- To use `spread()`, pass it the name of a data frame, then the name of the key column in the data frame, and then the name of the value column.
- Pass the column names as they are; do not use quotes.
- To tidy table2, you would pass `spread()` the key column and then the value column.

Tidy Data with R

```
## Source: local data frame [12 x 4]
##
##         country year         key      value
## 1   Afghanistan 1999       cases        745
## 2   Afghanistan 1999 population   19987071
## 3   Afghanistan 2000       cases       2666
## 4   Afghanistan 2000 population   20595360
.....
```

Tidy Data with R

```
library(tidyr)
spread(table2, key, value)

## Source: local data frame [6 x 4]
##
##       country year  cases population
## 1 Afghanistan 1999    745   19987071
## 2 Afghanistan 2000   2666   20595360
## 3      Brazil 1999  37737  172006362
## 4      Brazil 2000  80488  174504898
## 5       China 1999 212258 1272915272
## 6       China 2000 213766 1280428583
```

Tidy Data with R

- `spread()` returns a copy of your data set that has had the **key** and **value** columns removed.
- In their place, `spread()` adds a new column for each unique value of the key column (i.e. new columns: cases and populations).
- These unique values will form the column names of the new columns.
- `spread()` distributes the cells of the former value column across the cells of the new columns and truncates any non-key, non-value columns in a way that prevents duplication.

Tidy Data with R

- `spread()` distributes a pair of key:value columns into a field of cells. The unique values of the key column become the column names of the field of cells.
- You can see that `spread()` maintains each of the relationships expressed in the original data set. The output contains the four original variables, country, year, population, and cases.
- And the values of these variables are grouped according to the orginal observations, but now the layout of these relationships is tidy.

`spread()` takes three optional arguments in addition to data, key, and value:

- `fill`
- `convert`
- `drop`

Tidy Data with R

`fill`

- If the tidy structure creates combinations of variables that do not exist in the original data set, `spread()` will place an `NA` in the resulting cells.
- (`NA` is Rs missing value symbol).
- You can change this behaviour by passing `fill` an alternative value to use.

`convert`

- If a value column contains multiple types of data, its elements will be saved as a single type, usually character strings.
- As a result, the new columns created by `spread()` will also contain character strings.
- If you set `convert = TRUE`, `spread()` will run `type.convert()` on each new column, which will convert strings to *doubles (numerics)*, *integers*, *logicals*, *complexes*, or *factors*.

`drop`

- The `drop` argument controls how `spread()` handles factors in the key column.
- If you set `drop = FALSE`, spread will keep factor levels that do not appear in the key column, filling in the missing combinations with the value of fill.

gather()

- gather() does the reverse of spread().
- gather() collects a set of column names and places them into a single key column.
- It also collects the cells of those columns and places them into a single value column.
- You can use gather() to tidy table4.

Tidy Data with R

```
table4  # cases

## Source: local data frame [3 x 3]
##
##       country   1999   2000
## 1 Afghanistan    745   2666
## 2      Brazil  37737  80488
## 3       China 212258 213766
```

Tidy Data with R

- To use `gather()`, pass it the name of a data frame to reshape.
- Then pass `gather()` a character string to use for the name of the key column that it will make, as well as a character string to use as the name of the value column that it will make.
- Finally, specify which columns `gather()` should collapse into the key value pair (here with integer notation).

Tidy Data with R

```
gather(table4, "year", "cases", 2:3)
```

```
## Source: local data frame [6 x 3]
##
##       country year  cases
## 1 Afghanistan 1999    745
## 2      Brazil 1999  37737
## 3       China 1999 212258
## 4 Afghanistan 2000   2666
## 5      Brazil 2000  80488
## 6       China 2000 2137664
```

Tidy Data with R

- `gather()` returns a copy of the data frame with the specified columns removed.
- To this data frame, `gather()` has added two new columns: a key column that contains the former column names of the removed columns, and a value column that contains the former values of the removed columns.

Tidy Data with R

- `gather()` repeats each of the former column names (as well as each of the original columns) to maintain each combination of values that appeared in the original data set.
- `gather()` uses the first string that you supplied as the name of the new key column, and it uses the second string as the name of the new value column.

Tidy Data with R

- Just like `spread()`, gather maintains each of the relationships in the original data set.
- `gather()` also maintains each of the observations in the original data set, organizing them in a tidy fashion.

Tidy Data with R

```
table5  # population

## Source: local data frame [3 x 3]
##
##       country       1999       2000
## 1 Afghanistan   19987071   20595360
## 2      Brazil  172006362  174504898
## 3       China 1272915272 1280428583
```

Tidy Data with R

```
gather(table5, "year", "population", 2:3)
```

```
## Source: local data frame [6 x 3]
##
##       country year population
## 1 Afghanistan 1999   19987071
## 2      Brazil 1999  172006362
## 3       China 1999 1272915272
## 4 Afghanistan 2000   20595360
## 5      Brazil 2000  174504898
## 6       China 2000 1280428583
```

Tidy Data with R

- Here we identified the columns to collapse with a series of integers. 2:3 describes the second and third columns of the data frame.
- You can identify the same columns with each of the commands below.
- You can also identify columns by name with the notation introduced by the select function in dplyr

```
gather(table5, "year", "population", c(2, 3)
gather(table5, "year", "population", -1)
```

Tidy Data with R

`separate()` and `unite()`

- `spread()` and `gather()` help you reshape the layout of your data to place variables in columns and observations in rows.

- `separate()` and `unite()` allow you split and combine cells to place a single, complete value in each cell.

`separate()`

- `separate()` turns a single character column into multiple columns by splitting the values of the column wherever a separator character appears.
- So, for example, we can use `separate()` to tidy table3, which combines values of cases and population in the same column.

Tidy Data with R

(**BEFORE**)

```
# Data set three
table3

## Source: local data frame [6 x 3]
##
##       country year              rate
## 1 Afghanistan 1999       745/19987071
## 2 Afghanistan 2000      2666/20595360
## 3      Brazil 1999    37737/172006362
## 4      Brazil 2000    80488/174504898
## 5       China 1999 212258/1272915272
## 6       China 2000 213766/1280428583
```

Tidy Data with R

```
separate(table3, rate,
    into = c("cases", "population"))
```

```
## Source: local data frame [6 x 4]
##
##       country year  cases population
## 1 Afghanistan 1999    745   19987071
## 2 Afghanistan 2000   2666   20595360
## 3      Brazil 1999  37737  172006362
## 4      Brazil 2000  80488  174504898
## 5       China 1999 212258 1272915272
## 6       China 2000 213766 1280428583
```

Tidy Data with R

- To use `separate()` pass separate the name of a data frame to reshape and the name of a column to separate.
- Also give `separate()` an into argument, which should be a vector of character strings to use as new column names.
- `separate()` will return a copy of the data frame with the column removed.
- The previous values of the column will be split across several columns, one for each name in into.

Where to Separate?

- By default, `separate()` will split values wherever a non-alphanumeric character appears.
- Non-alphanumeric characters are characters that are neither a number nor a letter.
- For example, in the code above, `separate()` split the values of rate at the forward slash characters.

Specifying a Character

If you wish to use a specific character to separate a column, you can pass the character to the sep argument of `separate()`.

```
separate(table3, rate,
    into = c("cases", "population"),
    sep = "/")
```

Multiple Separation

- You can also pass an integer or vector of integers to `sep`. `separate()` will interpret the integers as positions to split at.
- Positive values start at 1 at the far-left of the strings;
- negative value start at -1 at the far-right of the strings.
- The length of `sep` should be one less than the number of names in into.

Tidy Data with R

- **Example:** You can use this arrangement to separate the last two digits of each year.

(Mid Columns : year into century and year)

```
separate(table3, year,
   into = c("century", "year"), sep = 2)
```

```
## Source: local data frame [6 x 4]
##
##       country century year              rate
## 1 Afghanistan      19   99       745/19987071
## 2 Afghanistan      20   00      2666/20595360
## 3      Brazil      19   99    37737/172006362
## 4      Brazil      20   00    80488/174504898
## 5       China      19   99 212258/1272915272
## 6       China      20   00 213766/1280428583
```

`unite()`

- `unite()` does the opposite of `separate()`: it combines multiple columns into a single column.
- We can use `unite()` to rejoin the century and year columns that we created in the last example.

Tidy Data with R

```
table6

## Source: local data frame [6 x 4]
##
##       country century year              rate
## 1 Afghanistan      19   99       745/19987071
## 2 Afghanistan      20   00      2666/20595360
## 3      Brazil      19   99   37737/172006362
## 4      Brazil      20   00   80488/174504898
## 5       China      19   99 212258/1272915272
## 6       China      20   00 213766/1280428583
```

Tidy Data with R

```
unite(table6, "new", century, year, sep = "")

## Source: local data frame [6 x 3]
##
##       country  new              rate
## 1 Afghanistan 1999     745/19987071
## 2 Afghanistan 2000    2666/20595360
## 3      Brazil 1999  37737/172006362
## 4      Brazil 2000  80488/174504898
## 5       China 1999 212258/1272915272
## 6       China 2000 213766/1280428583
```

Tidy Data with R

- Give `unite()` the name of the data frame to reshape, the name of the new column to create (as a character string), and the names of the columns to unite.
- `unite()` will place an underscore (_) between values from separate columns.

Tidy Data with R

- If you would like to use a different separator, or no separator at all, pass the separator as a character string to `sep`.
- `unite()` returns a copy of the data frame that includes the new column, but not the columns used to build the new column.
- If you would like to retain these columns, add the argument `remove = FALSE`.