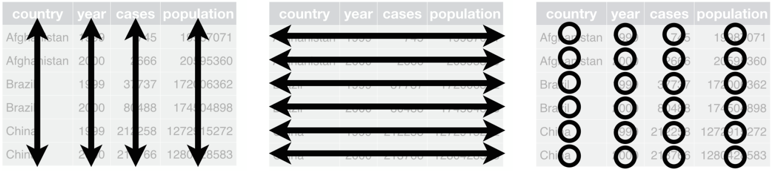
Tutorial - R - Data Wrangling

Psychology Tutorial Series - Emily Towner

May 09, 2021

What is tidy data?

- A way to organize your data that is consistent across datasets
- Each variable is a column, observation is a row, and cell is a value



variables observations values

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

What is tidyverse?

There are several ways to wrangle your data into a tidy format, but ***tidyverse*** has some great/easy functions.

The ***tidyverse*** is a coherent system of packages for data manipulation, exploration and visualization.

Includes:

- ggplot2 - graphics and plots
- dplyr - data manipulation
- tidyr - to create tidy data format
- & others

My favorite dplyr functions

Filter

- Use ***filter*** to include only observations that meet a specific criteria.
- For example, if I want a dataset consisting only of females, or only of those individuals with depression above the average depression score in this sample.
- This is useful when your data contains many waves/arms.

```r
data_female <- filter(data, sex == "Female")
data_above_average <- filter(data, depression_1 >
                             mean(depression_1, na.rm = T))
```

My favorite dplyr functions

Filter

```
head(data_female[,c(1,2,4)])
```

```
## # A tibble: 6 x 3
##   participant group               sex
##         <dbl> <chr>               <chr>
## 1           1 Early-Adolescence Female
## 2           3 Mid-Adolescence   Female
## 3           7 Late-Adolescence  Female
## 4           9 Early-Adolescence Female
## 5          10 Early-Adolescence Female
## 6          13 Early-Adolescence Female
```

My favorite dplyr functions

Filter

```
mean(data$depression_1, na.rm = T)
```

```
## [1] 55.71905
```

```
head(data_above_average[,c(1,2,4,9)])
```

```
## # A tibble: 6 x 4
##   participant group             sex    depression_1
##         <dbl> <chr>             <chr>         <dbl>
## 1           6 Mid-Adolescence   Male           64.7
## 2           7 Late-Adolescence  Female         66.6
## 3          10 Early-Adolescence Female         55.9
## 4          11 Late-Adolescence  Male           70.5
## 5          13 Early-Adolescence Female         79.1
## 6          14 Mid-Adolescence   Female         56.5
```

My favorite dplyr functions

Select

- Use ***select*** to include only the variables of interest
- For example, in subset 1 if I wanted data that only contains "age, sex, anxiety and early life stress" variables
- Or, use a selection helper like ***contains***, to subset data with all variables that contain the string "depression"
- Other helpers include ***starts_with***, ***ends_with***, and ***matches***
- This is useful when subsetting and scoring individual measures that are named with a convention (e.g. bdi_1, bdi_2, etc.)

```
data_subset_1 <- select(data, participant,
                 age, sex, anxiety, early_life_stress)
data_subset_2 <- select(data, participant,
                 contains('depression'))
```

My favorite dplyr functions

Select

```r
head(data_subset_1)
```

```
## # A tibble: 6 x 5
##   participant   age sex    anxiety early_life_stress
##         <dbl> <dbl> <chr>    <dbl>             <dbl>
## 1           1    10 Female    54.9              42.8
## 2           2    10 Male      74.4              27.2
## 3           3    16 Female    77.1              48.5
## 4           4    17 Male      73.0              60.1
## 5           5    23 Male      58.5              48.6
## 6           6    15 Male      93.1              57.1
```

My favorite dplyr functions

Select

```r
head(data_subset_2)
```

```
## # A tibble: 6 x 4
##   participant depression_1 depression_2 depression_3
##         <dbl>        <dbl>        <dbl>        <dbl>
## 1           1         43.8         36.5         24.3
## 2           2         52.1         43.4         28.9
## 3           3         26.4         26.3         24.7
## 4           4         41.6         29.7         17.5
## 5           5         45.6         45.6         46.3
## 6           6         64.7         46.2         27.2
```

My favorite dplyr functions

Mutate

- Use mutate to add new variables while preserving existing ones

```r
data_new <- mutate(data,
              anxiety_norm =
              anxiety / mean(anxiety))
head(data_new[,c(1,2,6,12)])
```

```
## # A tibble: 6 x 4
##   participant group             anxiety anxiety_norm
##         <dbl> <chr>               <dbl>        <dbl>
## 1           1 Early-Adolescence    54.9        0.730
## 2           2 Early-Adolescence    74.4        0.989
## 3           3 Mid-Adolescence      77.1        1.03
## 4           4 Mid-Adolescence      73.0        0.971
## 5           5 Late-Adolescence     58.5        0.777
## 6           6 Mid-Adolescence      93.1        1.24
```

BONUS

Normal Syntax

- Can be difficult to read when things get complicated (focus on nouns, must read from inside out).

```r
data_new <- mutate(data,
           anxiety_norm =
           anxiety / mean(anxiety))
```

- In an object called data_new, mutate a variable using data.

Pipe Operator (technically from magrittr package)

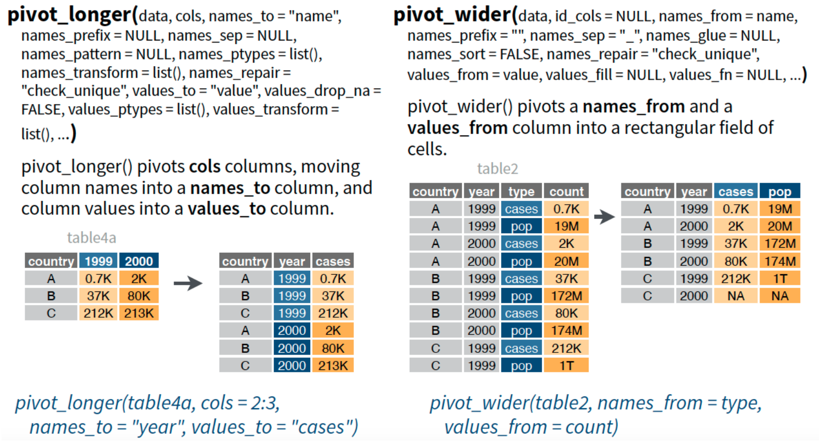
```r
data_new <- data %>%
           mutate(anxiety_norm =
           anxiety / mean(anxiety))
```

- Focuses on verbs, easier to read
- In an object called data_new, take my data and then mutate it.
- Can be used with all functions

My favorite tidyr functions

Pivot

- Convert data from long to wide format and vice versa



pivot_longer(data, cols, names_to = "name", names_prefix = NULL, names_sep = NULL, names_pattern = NULL, names_ptypes = list(), names_transform = list(), names_repair = "check_unique", values_to = "value", values_drop_na = FALSE, values_ptypes = list(), values_transform = list(), ...**)**

pivot_longer() pivots **cols** columns, moving column names into a **names_to** column, and column values into a **values_to** column.

table4a

country	1999	2000
A	0.7K	2K
B	37K	80K
C	212K	213K

→

country	year	cases
A	1999	0.7K
B	1999	37K
C	1999	212K
A	2000	2K
B	2000	80K
C	2000	213K

pivot_longer(table4a, cols = 2:3, names_to = "year", values_to = "cases")

pivot_wider(data, id_cols = NULL, names_from = name, names_prefix = "", names_sep = "_", names_glue = NULL, names_sort = FALSE, names_repair = "check_unique", values_from = value, values_fill = NULL, values_fn = NULL, ...**)**

pivot_wider() pivots a **names_from** and a **values_from** column into a rectangular field of cells.

table2

country	year	type	count
A	1999	cases	0.7K
A	1999	pop	19M
A	2000	cases	2K
A	2000	pop	20M
B	1999	cases	37K
B	1999	pop	172M
B	2000	cases	80K
B	2000	pop	174M
C	1999	cases	212K
C	1999	pop	1T

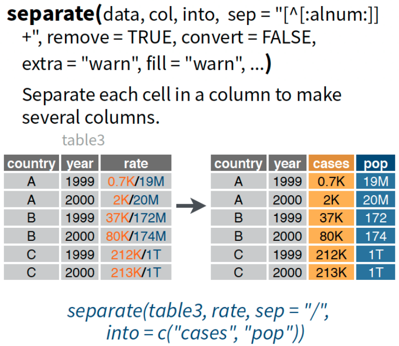
→

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172M
B	2000	80K	174M
C	1999	212K	1T
C	2000	NA	NA

pivot_wider(table2, names_from = type, values_from = count)

My favorite tidyr functions

Separate



separate(data, col, into, sep = "[^[:alnum:]]+", remove = TRUE, convert = FALSE, extra = "warn", fill = "warn", ...**)**

Separate each cell in a column to make several columns.

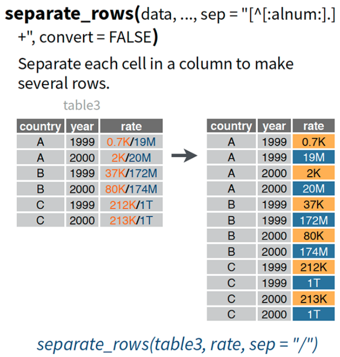
table3

country	year	rate
A	1999	0.7K/19M
A	2000	2K/20M
B	1999	37K/172M
B	2000	80K/174M
C	1999	212K/1T
C	2000	213K/1T

→

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172
B	2000	80K	174
C	1999	212K	1T
C	2000	213K	1T

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



separate_rows(data, ..., sep = "[^[:alnum:].]+", convert = FALSE**)**

Separate each cell in a column to make several rows.

table3

country	year	rate
A	1999	0.7K/19M
A	2000	2K/20M
B	1999	37K/172M
B	2000	80K/174M
C	1999	212K/1T
C	2000	213K/1T

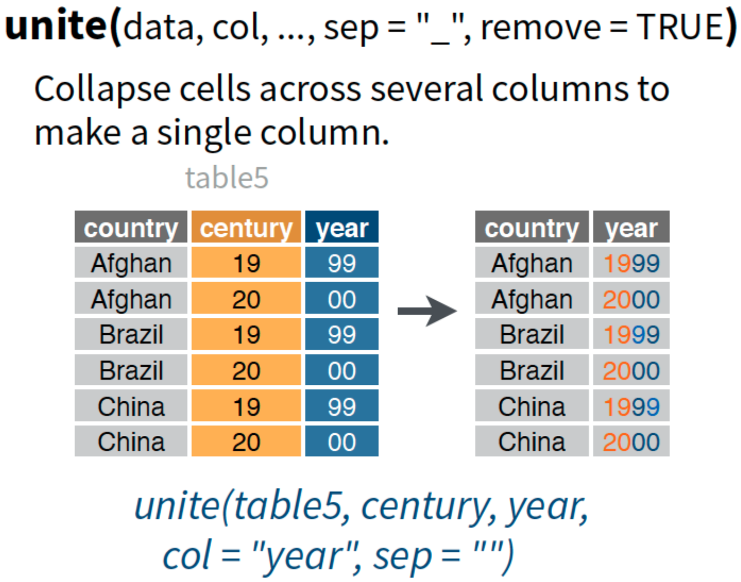
→

country	year	rate
A	1999	0.7K
A	1999	19M
A	2000	2K
A	2000	20M
B	1999	37K
B	1999	172M
B	2000	80K
B	2000	174M
C	1999	212K
C	1999	1T
C	2000	213K
C	2000	1T

separate_rows(table3, rate, sep = "/")

My favorite tidyr functions

Unite



unite(data, col, ..., sep = "_", remove = TRUE**)**

Collapse cells across several columns to make a single column.

table5

country	century	year
Afghan	19	99
Afghan	20	00
Brazil	19	99
Brazil	20	00
China	19	99
China	20	00

$\rightarrow$

country	year
Afghan	1999
Afghan	2000
Brazil	1999
Brazil	2000
China	1999
China	2000

unite(table5, century, year, col = "year", sep = "")

Cheatsheets and More

- Find cheatsheets and more for each package at: https://tidyverse.tidyverse.org/

Cheat Sheet

Data Transformation with dplyr : : CHEAT SHEET

Manipulate Cases

Manipulate Variables

Summarise Cases

Group Cases

Vector Functions

Summary Functions

Combine Tables

Row Names

- Now for a data cleaning tutorial!