Data wrangling

Cleaning and manipulating data

Jordan Creed
Moffitt Cancer Center

June 23, 2021

Complex wrangling

Mutating `join()`

Mutating joins combine variables from two data frames

```
XXX_join(left_data, right_data,by = "id")

XXX_join(left_data, right_data, by = c("left_id" = "ri
```

Filtering `join()`

`semi_join()` : returns rows from "left" data with matching values in "right" data, but only returns columns from "left" data

`anti_join()` : returns rows from "left" data with no matching values in "right" data and only returns columns from "left" data

`tidyr` actions

🕵️‍♀️ Tidy data isn't necessarily always long or wide.

`pivot_longer()` and `pivot_wider()` reshape data

- `pivot_longer(tibble_name, cols = variable1:variable5, names_to = "year", values_to = "count")`
- `pivot_wider(tibble_name, id_cols = variable1, names_from = variable2, values_from = variable3)`

There may be times where you need to combine `pivot_wider()` and `pivot_longer()`

Missing values propagate

```r
NA + 2
```

```
## [1] NA
```

```r
sum(NA, 6, 12)
```

```
## [1] NA
```

👨‍💻 `filter()` on missing values requires special treatment

FAILS: `filter(data, variable == NA)`

WORKS: `filter(data, is.na(variable))`

`tidyr` actions

`drop_na()`, `fill()`, and `replace_na()` format missing values

- `drop_na(variable2)` : removes rows with NA for that column
- `fill(variable2)` : fills in missing values with the previous value (up or down)
- `replace_na(list(variable2 = "replacement value"))` : replaces `NA`s with a specified value

`tidyr` actions

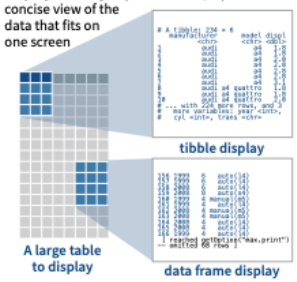
`separate()`, `extract()` and `unite()` split and combine character columns

- `separate(tibble_name, variable1, into = c("new_variable1", "new_variable2"), sep = "-")`
- `extract(tibble_name, variable1, regex="[[:alnum:]]+)")`
  - `extract` uses regex which is outside the scope of this class - so we will not be covering this function
- `unite(tibble_name, "combined_var_name", variable1, variable2, sep = ":")`

Tibbles - an enhanced data frame

The **tibble** package provides a new S3 class for storing tabular data, the tibble. Tibbles inherit the data frame class, but improve three behaviors:

- **Subsetting** - [always returns a new tibble, [[and $ always return a vector.
- **No partial matching** - You must use full column names when subsetting
- **Display** - When you print a tibble, R provides a concise view of the data that fits on one screen



- Control the default appearance with options:
  options(tibble.print_max = n, tibble.print_min = m, tibble.width = Inf)
- View full data set with **View()** or **glimpse()**
- Revert to data frame with **as.data.frame()**

CONSTRUCT A TIBBLE IN TWO WAYS

tibble(…)
Construct by columns.
tibble(x = 1:3, y = c("a", "b", "c"))

Both make this tibble

tribble(…)
Construct by rows.
tribble(~x, ~y,
1, "a",
2, "b",
3, "c")

```
A tibble: 3 × 2
      x     y
  <int> <chr>
1     1     a
2     2     b
3     3     c
```

as_tibble(x, …) Convert data frame to tibble.

enframe(x, name = "name", value = "value")
Convert named vector to a tibble

is_tibble(x) Test whether x is a tibble.

Tidy Data with tidyr

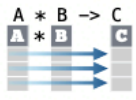
Tidy data is a way to organize tabular data. It provides a consistent data structure across packages.

A table is tidy if:

Each **variable** is in its own **column**

Each **observation**, or **case**, is in its own **row**

Tidy data:



Makes variables easy to access as vectors

Preserves cases during vectorized operations

Reshape Data - change the layout of values in a table

Use **pivot_longer()** and **pivot_wider()** to reorganize the values of a table into a new layout.

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)

Handle Missing Values

drop_na(data, …)
Drop rows containing NA's in … columns.

x

x1	x2
A	1
B	NA
C	NA
D	3
E	NA

→

x1	x2
A	1
D	3

drop_na(x, x2)

fill(data, …, .direction = c("down", "up"))
Fill in NA's in … columns with most recent non-NA values.

x

x1	x2
A	1
B	NA
C	NA
D	3
E	NA

→

x1	x2
A	1
B	1
C	1
D	3
E	3

fill(x, x2)

replace_na(data, replace = list(), …)
Replace NA's by column.

x

x1	x2
A	1
B	NA
C	NA
D	3
E	NA

→

x1	x2
A	1
B	2
C	2
D	3
E	2

replace_na(x, list(x2 = 2))

Expand Tables - quickly create tables with combinations of values

complete(data, …, fill = list())
Adds to the data missing combinations of the values of the variables listed in …
complete(mtcars, cyl, gear, carb)

expand(data, …)
Create new tibble with all possible combinations of the values of the variables listed in …
expand(mtcars, cyl, gear, carb)

Split Cells

Use these functions to split or combine cells into individual, isolated values.

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 = "/")

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

→

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

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