Exam 2 cheat sheet

Main wrangling verbs

verb	action
`arrange`	arrange the *rows* according to some *column*
`filter`	filter out or obtain a subset of the *rows* (==, >=, <=, !=, etc)
`select`	select a subset of *columns* (`starts_with("___")`, `ends_with("___")`, or `contains("___")`)
`mutate`	mutate or create a *column*
`summarize`	calculate a numerical summary of a *column* (tot, min, max, mean, median)
`group_by`	group the *rows* by a specified *column*

symbol	meaning
`==`	equal to
`!=`	not equal to
`>`	greater than
`>=`	greater than or equal to
`<`	less than
`<=`	less than or equal to
`%in% c(???, ???)`	a list of multiple values

Lubridate functions
year()
month()/month(__,lable = TRUE)
mday() : day of month
wday(__)/ wday(__, label = TRUE) : day of week

Reshaping

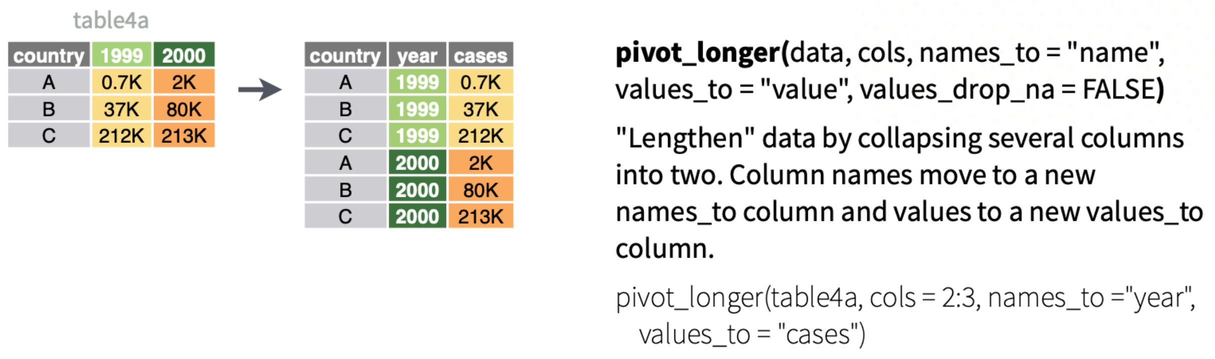


table4a

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(data, cols, names_to = "name", values_to = "value", values_drop_na = FALSE**)**

"Lengthen" data by collapsing several columns into two. Column names move to a new names_to column and values to a new values_to column.

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

combine values from multiple variables into 1 variable

- `cols` = the columns (variables) to collect into a single, new variable. We can also specify what variables we *don't* want to collect
- `names_to` = the name of the new variable which will include the *names* or labels of the collected variables
- `values_to` = the name of the new variable which will include the *values* of the collected variables

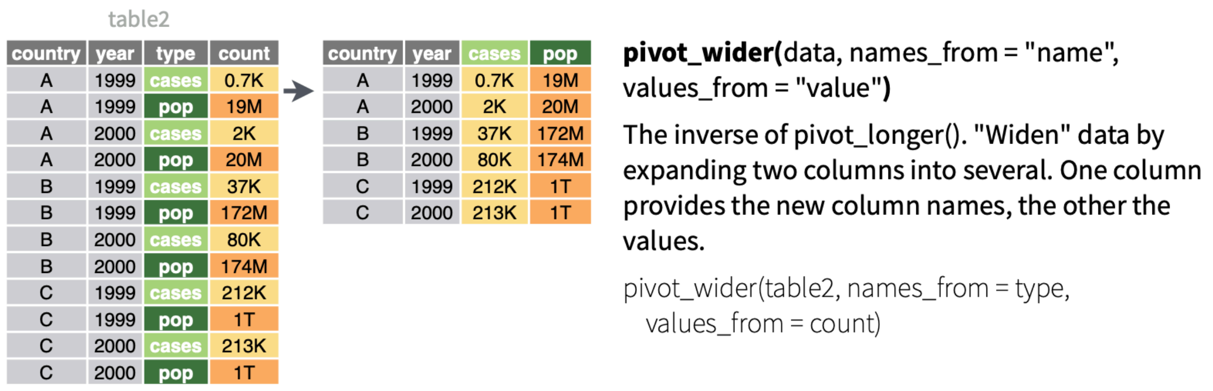


table2

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
C	2000	cases	213K
C	2000	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	213K	1T

pivot_wider(data, names_from = "name", values_from = "value"**)**

The inverse of pivot_longer(). "Widen" data by expanding two columns into several. One column provides the new column names, the other the values.

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

Make the data *wider*, i.e. spread out the values across new variables

- `names_from` = the variable whose values we want to separate into their own columns, i.e. where we want to get the new column *names from*
- `values_from` = which variable to take the new column *values from*

Joins

Mutating joins

`left_data |> mutating_join(right_data)`

The most common mutating joins are:

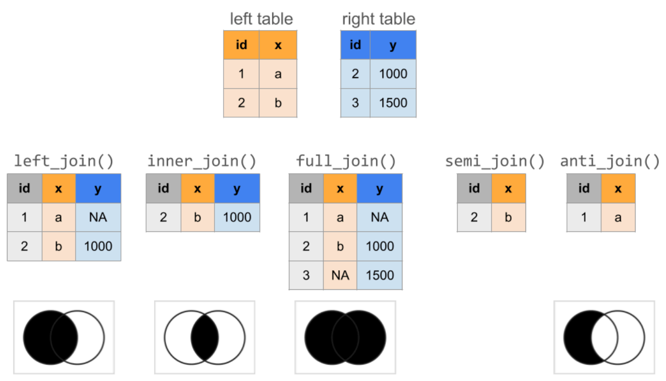
- `left_join()`
  Keeps *all* observations from the left, but discards any observations in the right that do not have a match in the left.1
- `inner_join()`
  Keeps *only* the observations from the left with a match in the right.

- `full_join()`
  Keeps *all* observations from the left *and* the right. (This is less common than `left_join()` and `inner_join()`).

Filtering joins

Filtering joins keep specific observations from the left table based on whether they match an observation in the right table.

- `semi_join()`
  Discards any observations in the left table that *do not* have a match in the right table. If there are multiple matches of right cases to a left case, it keeps just one copy of the left case.
- `anti_join()`
  Discards any observations in the left table that *do* have a match in the right table.



left table

id	x
1	a
2	b

right table

id	y
2	1000
3	1500

`left_join()`

id	x	y
1	a	NA
2	b	1000

`inner_join()`

id	x	y
2	b	1000

`full_join()`

id	x	y
1	a	NA
2	b	1000
3	NA	1500

`semi_join()`

id	x
2	b

`anti_join()`

id	x
1	a

Factors

- functions for changing the order of factor levels
  - `fct_relevel()` = *manually* reorder levels
  - `fct_reorder()` = reorder levels according to values of another *variable*
  - `fct_infreq()` = order levels from highest to lowest frequency
  - `fct_rev()` = reverse the current order
- functions for changing the labels or values of factor levels
  - `fct_recode()` = *manually* change levels
  - `fct_lump()` = *group together* least common levels

Strings

- *`str_replace(x, pattern, replacement)` finds the first part of `x` that matches the `pattern` and replaces it with `replacement`*
- *`str_replace_all(x, pattern, replacement)` finds all instances in `x` that matches the `pattern` and replaces it with `replacement`*
- *`str_to_lower(x)` converts all upper case letters in `x` to lower case*
- *`str_sub(x, start, end)` only keeps a subset of characters in `x`, from `start` (a number indexing the first letter to keep) to `end` (a number indexing the last letter to keep)*
- *`str_length(x)` records the number of characters in `x`*
- *`str_detect(x, pattern)` is TRUE if `x` contains the given `pattern` and FALSE otherwise*