


COMP 112: DATA WRANGLING SUMMARY SHEET

Sam Kennedy

- Pipe operator `|>`
 - Passes object to function
 - Tidyverse syntax: `%>%`
- Wrangling verbs
 - Select
 - Select a subset of columns
 - Mutate
 - Mutate existing columns or add new ones
 - Filter
 - Filter subset of rows based on values of 1+ columns
 - Boolean condition
 - Arrange
 - Arrange rows based on values of 1+ columns
 - Summarize
 - Calculate numerical summary of a column
 - Ex: mean, median, max, min
 - Group_by
 - Group rows based on values of 1+ columns
- Reshaping
 - Pivot longer
 - Join several columns into two columns

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")
```

- Pivot wider
 - Expand two columns into several columns

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)`

- Joining datasets

- Left_join()

- Keeps all observations from the left, drops those on the right without a match in the left

- Inner_join()

- Keeps only the observations from the left with a match in the right.

- Full_join()

- Keeps all observations from both the left and the right

left table		right table	
id	x	id	y
1	a	2	1000
2	b	3	1500

left_join()

id	x	y
1	a	NA
2	b	1000

inner_join()

id	x	y
2	b	1000

full_join()

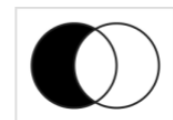
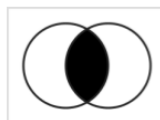
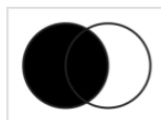
id	x	y
1	a	NA
2	b	1000
3	NA	1500

semi_join()

id	x
2	b

anti_join()

id	x
1	a



- Factors

- Fct_recode

- Change labels of factor

- Fct_relevel

- Change char variables to factors
 - Specify meaningful order for factor variables
 - Reorder factor variables

- Fct_reorder

- Reorder factor variables using another variable

- Strings

- `str_replace()`
 - finds the first part of `x` that matches the pattern and replaces it with replacement
- `str_replace_all()`
 - finds all instances in `x` that matches the pattern and replaces it with replacement
- `str_to_lower()`
 - Converts uppercase characters to lowercase
- `str_sub()`
 - Only keeps subset of characters in string from 'start' value to 'end' value
- `str_length()`
 - Returns number of characters in string
- `str_detect()`
 - Returns TRUE if string contains given pattern, otherwise returns FALSE

Essential Functions

The `stringr` package within `tidyverse` contains lots of functions to help process strings. We'll focus on the most common. Letting `x` be a string variable...

function	arguments	returns
<code>str_replace()</code>	<code>x, pattern, replacement</code>	a modified string
<code>str_replace_all()</code>	<code>x, pattern, replacement</code>	a modified string
<code>str_to_lower()</code>	<code>x</code>	a modified string
<code>str_sub()</code>	<code>x, start, end</code>	a modified string
<code>str_length()</code>	<code>x</code>	a number
<code>str_detect()</code>	<code>x, pattern</code>	TRUE/FALSE

-