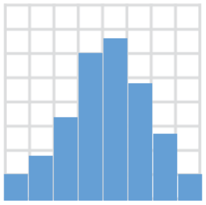
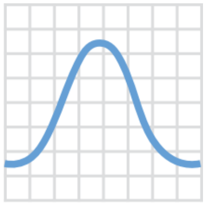
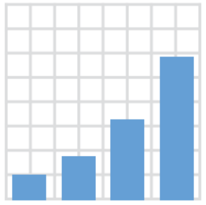
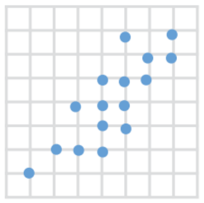
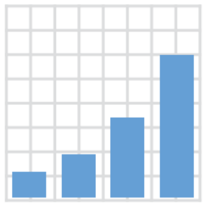
Unit 2 Cheat Sheet

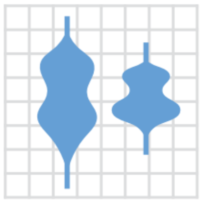
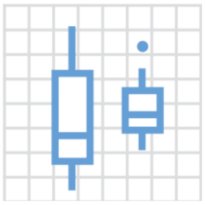
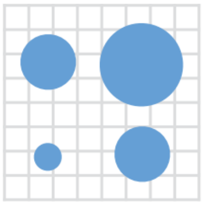
Tibbles:

- `%>%` or `|>` Pipe operator passes the output of one expression directly into the next function as the first argument.
- `data.frame(col1 = vector1, col2 = vector2)` Creates and returns a data frame from the given vectors.
- `n()` Returns the number of observations or rows in a group when used inside a `summarize()` or `mutate()` call within grouped data.
- `nrow()` Returns the number of rows in a data frame.
- `ncol()` Returns the number of columns in a data frame.
- `dim()` Returns the number of rows and columns in a data frame.
- `colnames()` Returns the names of the columns in a data frame.
- `head(n = 5)` Returns the specified number of rows from the beginning of a data frame.
- `tail(n = 5)` Returns the specified number of rows from the end of a data frame.
- `rbind(data_frame, new_vec)` Adds an extra row or rows onto a data frame.
- `data.frame$col1` Dollar sign used to access columns of data frames, can be used to modify, add, and remove columns.
- `as.data.frame(tibble)` Converts a tibble to a data frame.
- `tibble(col1 = vector1, col2 = vector2)` Creates and returns a tibble from the given vectors.
- `read_csv(file_name)` Reads in a csv file as a tibble.
- `read_tsv(file_name)` Reads in a tsv file as a tibble.
- `write_csv(tibble, file_name)` Writes a tibble as a csv file to the specified path.
- `write_tsv(tibble, file_name)` Writes a tibble as a tsv file to the specified path.
- `print( … , n = NULL, width = NULL)` In tibbles print has additional arguments, `n` specifies how many rows to print and `width` controls the total width of the output in characters. By default these arguments are set to `NULL` with `n` showing the first 10 rows, and `width` fitting as many columns as can fit in the console width. Can set each to an integer or `Inf` to show all.

Visualizing data:

- `ggplot(data, aes(x=col1, y=col2, color=col3, fill=col4, size=col5))` Generates a graph using a tibble, with its columns providing the data to populate the graph.
- `labs(x = "n1", y = "n2", color = "n3", size = "n4")` Renames the labels of graph attributes.
- `facet_wrap(~col5)` Creates panels of the graph based subsections of the data.
- `theme_bw()` Changes the color scheme to have white background with grid lines.
- `theme(legend.position = "none")` Removes legend.

Plot Type	Description	# Variables + Data Type	Image
`geom_histogram()`	Creates a one variable histogram	1 variable continuous	
`geom_density()`	Creates a density plot	1 variable continuous	
`geom_bar()`	Creates a one variable bar plot	1 variable discrete	
`geom_point()`	Creates a scatter plot	2 variables both continuous	
`geom_col()`	Creates a two variable bar plot	2 variables one discrete, one continuous	

Plot Type	Description	# Variables + Data Type	Image
`geom_violin`	Creates a violin plot	2 variables one discrete, one continuous	
`geom_boxplot()`	Creates a box plot	2 variables one discrete, one continuous	
`geom_count()`	Creates a 2-dimensional frequency graph (point plot)	2 variables both discrete	
`geom_line()`	Creates a two variable line graph	2 variables continuous function	

Transforming data:

- `filter(data, col1 >= value)` Eliminates rows from a tibble on the given conditional.
- `arrange(data, col1)` Sorts rows in a tibble in ascending order.
- `arrange(data, desc(col1))` Sorts rows in a tibble in descending order.
- `distinct(data)` Keeps the first occurrence of each unique row while preserving the original column order.
- `select(data, col1, col2)` Keeps only the specified columns in the tibble.
- `mutate(data, new_col = col1 * col2)` Creates a new column in the tibble.
- `mutate(data, col1 = col1 ^ 2)` Modifies an existing column in the tibble.
- `group_by(data, col1) %>% summarize(mean_col = mean(col2))` both `group_by()` and `summarize()` are always used in tandem to create a summary statistic for the tibble based on the specified column in `group_by()`.
- `rename(new_name = old_name)` Renames a column in a tibble.
- `vector = pull(data, col1)` Pulls out a column from a tibble as a vector.

Strings:

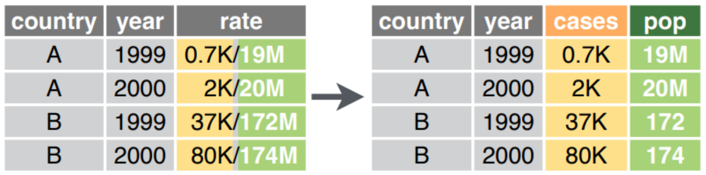
- `str_detect(string, "substring")` Returns a boolean value to detect presence of a substring in a given string.
- `str_starts(string, "starting")` Returns boolean value to detect presence of a substring at the beginning of a given string.
- `str_ends(string, "ending")` Returns boolean value to detect presence of a substring at the end of a given string.
- `str_length(string)` Returns the character length of a string.
- `str_to_lower(string)` Converts each character in the string to lower case.
- `str_to_upper(string)` Converts each character in the string to upper case.
- `str_to_title(string)` Converts the first character in each word to upper case, and the rest to lower case.
- `str_replace_all(string, "original", "replace")` Replaces each substring with a replacement string.
- `str_replace_na(string, "replace")` Replaces all NA values with a replacement string.
- `str_sub(string, 3, 6)` Returns the substring between the passed indices.

Factors:

- `factor(vector, levels = c("level1", "level2", "level3"))` Creates a factor from the vector with specified levels.
- `combined_factor = fct_c(factor1, factor2)` Combines two factors.
- `fct_recode(factor, "new_level1" = "level1", "new_level2" = "level2)` Renames the levels in a factor.
- `vector = as.vector(factor)` Converts a factor to a vector.
- `factor[1:3]` Subset first 3 levels of a factor.

Tidy data:

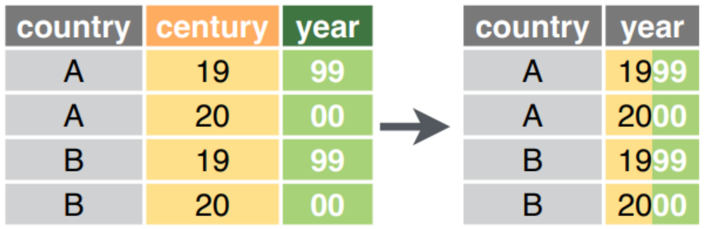
- `separate(data, existing_column, into = c("new_col1", "new_col2"), sep = "-", convert = False)` Separates a column into two columns on a given character or substring.
  - ex. `separate(table, rate, sep = "/", into = c("cases","pop"))`



country	year	rate
A	1999	0.7K/19M
A	2000	2K/20M
B	1999	37K/172M
B	2000	80K/174M

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172
B	2000	80K	174

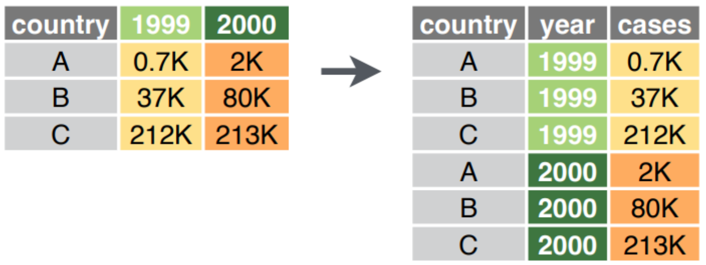
- `unite(data, new_col, old_col1, old_col2, sep = "-")` Combines two or more columns into a single column.
  - ex. `unite(table, century, year, col = "year", sep = "")`



country	century	year
A	19	99
A	20	00
B	19	99
B	20	00

country	year
A	1999
A	2000
B	1999
B	2000

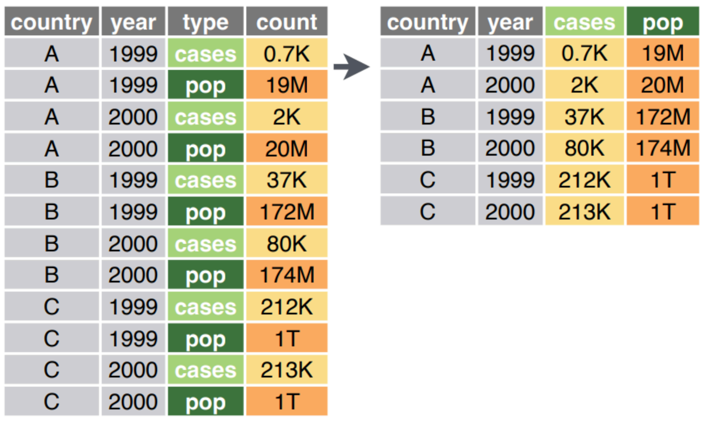
- `pivot_longer(data, col1:col4, names_to = "new_col1", values_to = "new_col2")` Pivots the tibble such that the specified columns are converted to a single column, creating more rows in the tibble.
  - ex. `pivot_longer(table, cols = 2:3, names_to = "year", values_to = "cases")`



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_wider(data, names_from = "col1", values_from = "col2")` Pivots the tibble such that the values from the specified columns are converted to multiple columns, creating more columns in the tibble.
  - ex. `pivot_wider(table, names_from = "type", values_from = "count")`

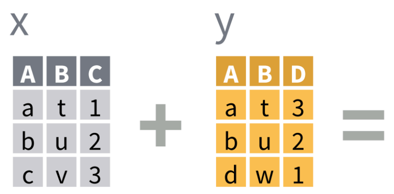


country	year	type	count
A	1999	cases	0.7K
A	1999	pop	19M
A	2000	cases	2K
A	2000	pop	20M
B	1999	cases	37K
B	1999	pop	172M
B	2000	cases	80K
B	2000	pop	174M
C	1999	cases	212K
C	1999	pop	1T
C	2000	cases	213K
C	2000	pop	1T

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

Relational data:

- Syntax for all joins: `join(tibbleX, tibbleY, by = c("col1" = "col2"))`



x

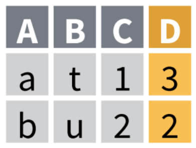
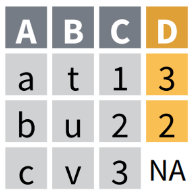
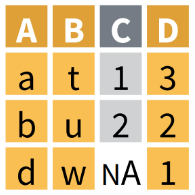
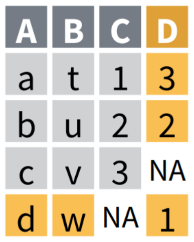
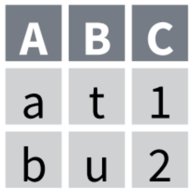
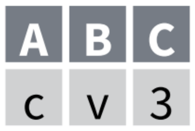
A	B	C
a	t	1
b	u	2
c	v	3

+

y

A	B	D
a	t	3
b	u	2
d	w	1

=

Formula	Description	Image
`inner_join()`	Only keeps rows present in both tibbles	 A B C D a t 1 3 b u 2 2
`left_join()`	Only keeps rows present in the left tibble	 A B C D a t 1 3 b u 2 2 c v 3 NA
`right_join()`	Only keeps rows present in the right tibble	 A B C D a t 1 3 b u 2 2 d w NA 1
`full_join()`	Keeps all rows from both tibbles	 A B C D a t 1 3 b u 2 2 c v 3 NA d w NA 1
`semi_join()`	Only keeps rows present in both tibbles, and only columns from the left tibble	 A B C a t 1 b u 2
`anti_join()`	Only keeps rows that do not overlap between the two tibbles, and only columns from the left tibble	 A B C c v 3

This document was created by Aidan Cardall, Ethan Bang, and Megan Rodabough at Brigham Young University. Images from ggplot2 and dplyr Posit Cheatsheets were obtained from https://posit.co/resources/cheatsheets.