Data Summarization

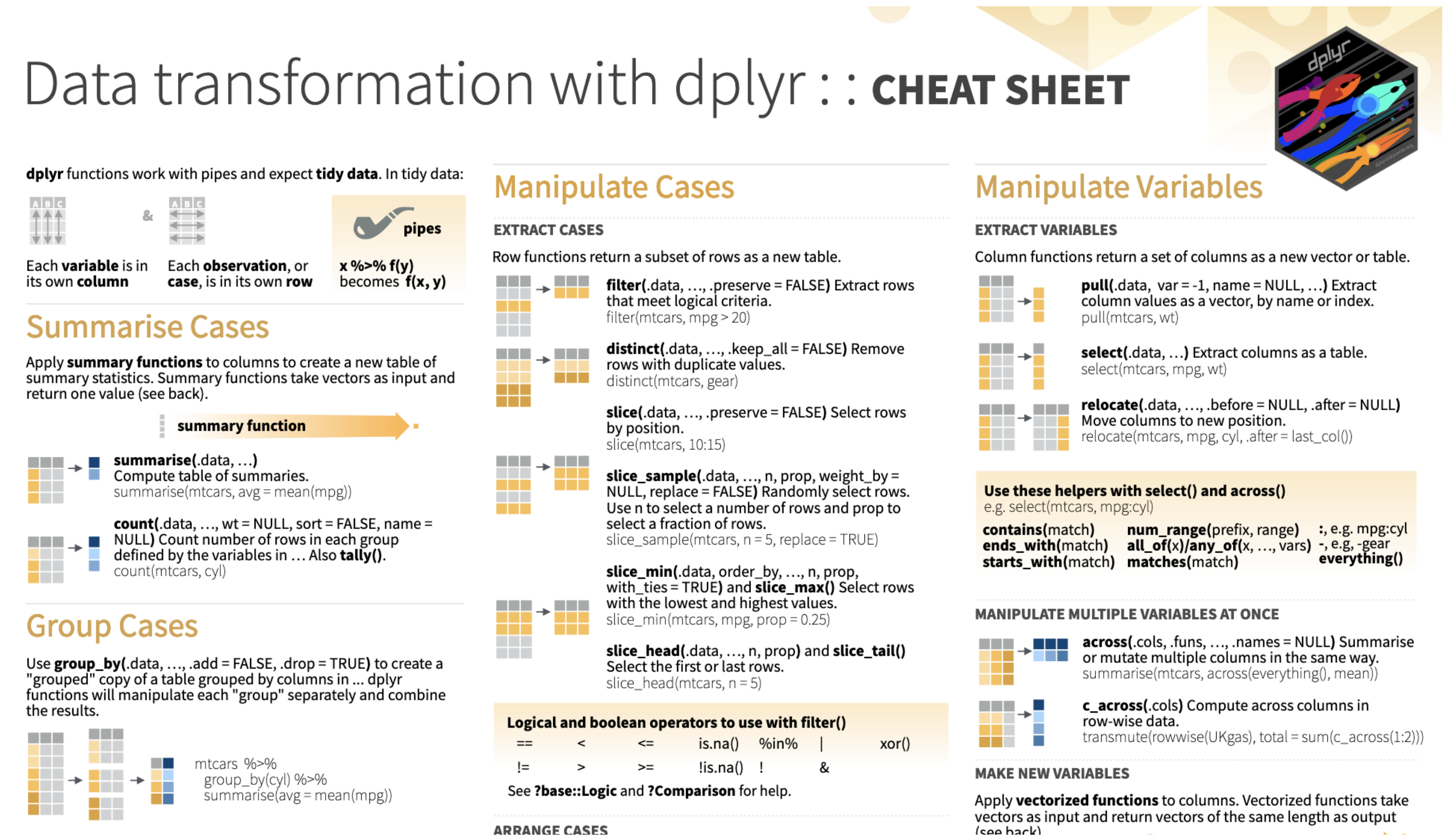
Recap

- `select()`: subset and/or reorder columns
- `filter()`: remove rows
- `arrange()`: reorder rows
- `mutate()`: create new columns or modify them
- `select()` and `filter()` can be combined together
- remove a column: `select()` with `!` mark (`!col_name`)
- you can do sequential steps: especially using pipes `%>%`

Cheatsheet

Another Cheatsheet

https://raw.githubusercontent.com/rstudio/cheatsheets/main/data-transformation.pdf



Data transformation with dplyr : : **CHEAT SHEET**

dplyr functions work with pipes and expect **tidy data**. In tidy data:

Each **variable** is in its own **column** & Each **observation**, or **case**, is in its own **row**

pipes

x %>% f(y) becomes **f(x, y)**

Summarise Cases

Apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).

summary function

summarise(.data, …**)** Compute table of summaries.
summarise(mtcars, avg = mean(mpg))

count(.data, …, wt = NULL, sort = FALSE, name = NULL**)** Count number of rows in each group defined by the variables in … Also **tally()**.
count(mtcars, cyl)

Group Cases

Use **group_by(**.data, …, .add = FALSE, .drop = TRUE**)** to create a "grouped" copy of a table grouped by columns in ... dplyr functions will manipulate each "group" separately and combine the results.

```
mtcars %>%
  group_by(cyl) %>%
  summarise(avg = mean(mpg))
```

Manipulate Cases

EXTRACT CASES

Row functions return a subset of rows as a new table.

filter(.data, …, .preserve = FALSE**)** Extract rows that meet logical criteria.
filter(mtcars, mpg > 20)

distinct(.data, …, .keep_all = FALSE**)** Remove rows with duplicate values.
distinct(mtcars, gear)

slice(.data, …, .preserve = FALSE**)** Select rows by position.
slice(mtcars, 10:15)

slice_sample(.data, …, n, prop, weight_by = NULL, replace = FALSE**)** Randomly select rows. Use n to select a number of rows and prop to select a fraction of rows.
slice_sample(mtcars, n = 5, replace = TRUE)

slice_min(.data, order_by, …, n, prop, with_ties = TRUE**)** and **slice_max()** Select rows with the lowest and highest values.
slice_min(mtcars, mpg, prop = 0.25)

slice_head(.data, …, n, prop**)** and **slice_tail()** Select the first or last rows.
slice_head(mtcars, n = 5)

Logical and boolean operators to use with filter()

==	<	<=	is.na()	%in%	|	xor()
!=	>	>=	!is.na()	!	&	

See **?base::Logic** and **?Comparison** for help.

ARRANGE CASES

Manipulate Variables

EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.

pull(.data, var = -1, name = NULL, …**)** Extract column values as a vector, by name or index.
pull(mtcars, wt)

select(.data, …**)** Extract columns as a table.
select(mtcars, mpg, wt)

relocate(.data, …, .before = NULL, .after = NULL**)** Move columns to new position.
relocate(mtcars, mpg, cyl, .after = last_col())

Use these helpers with select() and across()
e.g. select(mtcars, mpg:cyl)

contains(match**)**	**num_range(**prefix, range**)**	**:**, e.g. mpg:cyl
ends_with(match**)**	**all_of(**x**)/any_of(**x, …, vars**)**	**-**, e.g, -gear
starts_with(match**)**	**matches(**match**)**	**everything()**

MANIPULATE MULTIPLE VARIABLES AT ONCE

across(.cols, .funs, …, .names = NULL**)** Summarise or mutate multiple columns in the same way.
summarise(mtcars, across(everything(), mean))

c_across(.cols**)** Compute across columns in row-wise data.
transmute(rowwise(UKgas), total = sum(c_across(1:2)))

MAKE NEW VARIABLES

Apply **vectorized functions** to columns. Vectorized functions take vectors as input and return vectors of the same length as output (see back).

Data Summarization

- Basic statistical summarization
  - `mean(x)`: takes the mean of x
  - `sd(x)`: takes the standard deviation of x
  - `median(x)`: takes the median of x
  - `quantile(x)`: displays sample quantiles of x. Default is min, IQR, max
  - `range(x)`: displays the range. Same as `c(min(x), max(x))`
  - `sum(x)`: sum of x
  - `max(x)`: maximum value in x
  - `min(x)`: minimum value in x

Some examples

We can use the `CO_heat_ER` object from the `dasehr` package to explore different ways of summarizing data. (This dataset contains information about the number and rate of visits for heat-related illness to ERs in Colorado from 2011-2022, adjusted for age.) The `head` command displays the first rows of an object:

```
library(dasehr)
head(CO_heat_ER)
```

```
# A tibble: 6 × 7
  county     rate lower95cl upper95cl visits  year gender
  <chr>     <dbl>     <dbl>     <dbl>  <dbl> <dbl> <chr>
1 Statewide  5.64      4.70      6.59    140  2011 Female
2 Statewide  7.39      6.30      8.47    183  2011 Male
3 Statewide  6.51      5.80      7.23    323  2011 Both genders
4 Statewide  5.64      4.72      6.57    146  2012 Female
5 Statewide  7.56      6.48      8.65    193  2012 Male
6 Statewide  6.58      5.88      7.29    339  2012 Both genders
```

Behavior of `pull()` function

`pull()` converts a single data column into a vector. This allows you to run summary functions.

```
CO_heat_ER %>% pull(visits)
```

Statistical summarization the “tidy” way

Add the `na.rm =` argument for missing data

```
CO_heat_ER %>% pull(visits) %>% mean()
```

```
[1] NA
```

```
CO_heat_ER %>% pull(visits) %>% mean(na.rm=T)
```

```
[1] 9.791114
```

Summarization on tibbles (data frames)

Summarize the data: `dplyr summarize()` function

`summarize` creates a summary table.

Multiple summary statistics can be calculated at once (unlike `pull()` which can only do a single calculation on one column).

```
# General format - Not the code!
{data to use} %>%
   summarize({summary column name} = {function(source column)},
             {summary column name} = {function(source column)})
```

Summarize the data: **`dplyr`** **`summarize()`** function

```
CO_heat_ER %>%
  summarize(mean_visits = mean(visits))
```

```
# A tibble: 1 × 1
  mean_visits
        <dbl>
1          NA
```

```
CO_heat_ER %>%
  summarize(mean_visits = mean(visits, na.rm = TRUE))
```

```
# A tibble: 1 × 1
  mean_visits
        <dbl>
1        9.79
```

Summarize the data: **`dplyr summarize()`** function

`summarize()` can do multiple operations at once. Just separate by a comma.

```
CO_heat_ER %>%
  summarize(mean_visits = mean(visits, na.rm = TRUE),
            median_visits = median(visits, na.rm = TRUE),
            mean_rate = mean(rate, na.rm = TRUE))
```

```
# A tibble: 1 × 3
  mean_visits median_visits mean_rate
        <dbl>         <dbl>     <dbl>
1        9.79             0      1.87
```

Summarize the data: `dplyr` `summarize()` function

Note that `summarize()` creates a separate tibble from the original data.

If you want to save a summary statistic in the original data, use `mutate()` instead to create a new column for the summary statistic.

`summary()` Function

Using `summary()` can give you rough snapshots of each numeric column (character columns are skipped):

```
summary(CO_heat_ER)
```

```
    county               rate           lower95cl         upper95cl      
 Length:2340        Min.   : 0.000   Min.   : 0.000   Min.   :  0.000  
 Class :character   1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.:  0.000  
 Mode  :character   Median : 0.000   Median : 0.000   Median :  0.000  
                    Mean   : 1.869   Mean   : 1.119   Mean   :  2.755  
                    3rd Qu.: 0.000   3rd Qu.: 0.000   3rd Qu.:  0.000  
                    Max.   :89.275   Max.   :43.398   Max.   :151.420  
                    NA's   :832      NA's   :832      NA's   :832      
     visits             year         gender         
 Min.   :  0.000   Min.   :2011   Length:2340       
 1st Qu.:  0.000   1st Qu.:2014   Class :character  
 Median :  0.000   Median :2016   Mode  :character  
 Mean   :  9.791   Mean   :2016                     
 3rd Qu.:  0.000   3rd Qu.:2019                     
 Max.   :494.000   Max.   :2022                     
 NA's   :832                                        
```

Summary & Lab Part 1

- summary stats (`mean()`) work with `pull()`
- don't forget the `na.rm = TRUE` argument!
- `summary(x)`: quantile information
- `summarize`: creates a summary table of columns of interest

Class Website

Lab

CO ER Heat Illness Visits

Let’s filter the dataset of CO ER visits for any records with >0 heat-related illness.

```r
ER_filt <- CO_heat_ER %>%
  filter(visits > 0)
```

`distinct()` values

`distinct(x)` will return the unique elements of column `x`.

```
ER_filt %>%
  distinct(county)
```

```
# A tibble: 15 × 1
   county
   <chr>
 1 Statewide
 2 Adams
 3 Arapahoe
 4 Boulder
 5 Denver
 6 Douglas
 7 El Paso
 8 Jefferson
 9 Larimer
10 Logan
11 Mesa
12 Morgan
13 Prowers
14 Pueblo
15 Weld
```

How many `distinct()` values?

`n_distinct()` tells you the number of unique elements. *Must pull the column first!*

```
ER_filt %>%
  pull(county) %>%
  n_distinct()
```

```
[1] 15
```

dplyr: count

Use `count` to return a frequency table of row count by category.

```
ER_filt %>% count(county)
```

```
# A tibble: 15 × 2
   county         n
   <chr>      <int>
 1 Adams         30
 2 Arapahoe      29
 3 Boulder       15
 4 Denver        31
 5 Douglas        4
 6 El Paso       30
 7 Jefferson     33
 8 Larimer       29
 9 Logan          2
10 Mesa          17
11 Morgan         1
12 Prowers        1
13 Pueblo        30
14 Statewide     36
15 Weld          27
```

dplyr: count

Multiple columns listed further subdivides the count.

```
ER_filt %>% count(county, gender)
```

```
# A tibble: 36 × 3
   county   gender            n
   <chr>    <chr>         <int>
 1 Adams    Both genders     12
 2 Adams    Female            8
 3 Adams    Male             10
 4 Arapahoe Both genders     11
 5 Arapahoe Female            9
 6 Arapahoe Male              9
 7 Boulder  Both genders     12
 8 Boulder  Male              3
 9 Denver   Both genders     12
10 Denver   Female            8
# ℹ 26 more rows
```

Grouping

Perform Operations By Groups: dplyr

`group_by` allows you group the data set by variables/columns you specify:

```
CO_heat_ER_grouped <- CO_heat_ER %>% group_by(county)
CO_heat_ER_grouped
```

```
# A tibble: 2,340 × 7
# Groups:   county [65]
   county     rate lower95cl upper95cl visits  year gender
   <chr>     <dbl>     <dbl>     <dbl>  <dbl> <dbl> <chr>
 1 Statewide  5.64      4.70      6.59    140  2011 Female
 2 Statewide  7.39      6.30      8.47    183  2011 Male
 3 Statewide  6.51      5.80      7.23    323  2011 Both genders
 4 Statewide  5.64      4.72      6.57    146  2012 Female
 5 Statewide  7.56      6.48      8.65    193  2012 Male
 6 Statewide  6.58      5.88      7.29    339  2012 Both genders
 7 Statewide  4.94      4.06      5.82    124  2013 Female
 8 Statewide  6.72      5.72      7.72    178  2013 Male
 9 Statewide  5.82      5.16      6.49    302  2013 Both genders
10 Statewide  3.52      2.80      4.25     92  2014 Female
# ℹ 2,330 more rows
```

Summarize the grouped data

It’s grouped! Grouping doesn’t change the data in any way, but how **functions operate on it**. Now we can summarize `visits` by group:

```
CO_heat_ER_grouped %>%
  summarize(avg_visits = mean(visits, na.rm = TRUE))
```

```
# A tibble: 65 × 2
   county     avg_visits
   <chr>           <dbl>
 1 Adams            22.7
 2 Alamosa           0
 3 Arapahoe         20.8
 4 Archuleta         0
 5 Baca              0
 6 Bent              0
 7 Boulder          14.5
 8 Broomfield        0
 9 Chaffee           0
10 Cheyenne          0
# ℹ 55 more rows
```

Use the **pipe** to string these together!

Pipe `CO_heat_ER` into `group_by`, then pipe that into `summarize`:

```
CO_heat_ER %>%
  group_by(county) %>%
  summarize(avg_visits = mean(visits, na.rm = TRUE))
```

```
# A tibble: 65 × 2
   county     avg_visits
   <chr>           <dbl>
 1 Adams            22.7
 2 Alamosa           0  
 3 Arapahoe         20.8
 4 Archuleta         0  
 5 Baca              0  
 6 Bent              0  
 7 Boulder          14.5
 8 Broomfield        0  
 9 Chaffee           0  
10 Cheyenne          0  
# ℹ 55 more rows
```

Group by as many variables as you want

`group_by` county and gender:

```
CO_heat_ER %>%
  group_by(county, gender) %>%
  summarize(avg_visits = mean(visits, na.rm = TRUE))
```

```
# A tibble: 195 × 3
# Groups:   county [65]
   county    gender       avg_visits
   <chr>     <chr>             <dbl>
 1 Adams     Both genders       30.4
 2 Adams     Female             15.8
 3 Adams     Male               18.9
 4 Alamosa   Both genders        0  
 5 Alamosa   Female              0  
 6 Alamosa   Male                0  
 7 Arapahoe  Both genders       28.7
 8 Arapahoe  Female             14.4
 9 Arapahoe  Male               17.3
10 Archuleta Both genders        0  
# ℹ 185 more rows
```

Counting

There are other functions, such as `n()` count the number of observations (NAs included).

```
CO_heat_ER %>%
  group_by(county) %>%
  summarize(n = n(),
            mean = mean(visits, na.rm = TRUE))
```

```
# A tibble: 65 × 3
   county         n  mean
   <chr>      <int> <dbl>
 1 Adams         36  22.7
 2 Alamosa       36   0
 3 Arapahoe      36  20.8
 4 Archuleta     36   0
 5 Baca          36   0
 6 Bent          36   0
 7 Boulder       36  14.5
 8 Broomfield    36   0
 9 Chaffee       36   0
10 Cheyenne      36   0
# ℹ 55 more rows
```

Counting

`count()` and `n()` can give very similar information.

```
CO_heat_ER %>% count(county)
```

```
# A tibble: 65 × 2
   county         n
   <chr>      <int>
 1 Adams         36
 2 Alamosa       36
 3 Arapahoe      36
 4 Archuleta     36
 5 Baca          36
 6 Bent          36
 7 Boulder       36
 8 Broomfield    36
 9 Chaffee       36
10 Cheyenne      36
# ℹ 55 more rows
```

```
CO_heat_ER %>% group_by(county) %>% summarize(n()) # n() typically used with summarize
```

```
# A tibble: 65 × 2
   county     `n()`
   <chr>      <int>
 1 Adams         36
 2 Alamosa       36
 3 Arapahoe      36
 4 Archuleta     36
 5 Baca          36
```

A few miscellaneous topics ..

Base R functions you might see: `length` and `unique`

These functions require a column as a vector using `pull()`.

```
CO_heat_ER_loc <- CO_heat_ER %>% pull(county) # pull() to make a vector
CO_heat_ER_loc %>% unique() # similar to distinct()
```

```
 [1] "Statewide"   "Adams"       "Alamosa"     "Arapahoe"    "Archuleta"  
 [6] "Baca"        "Bent"        "Boulder"     "Broomfield"  "Chaffee"    
[11] "Cheyenne"    "Clear Creek" "Conejos"     "Costilla"    "Crowley"    
[16] "Custer"      "Delta"       "Denver"      "Dolores"     "Douglas"    
[21] "Eagle"       "Elbert"      "El Paso"     "Fremont"     "Garfield"   
[26] "Gilpin"      "Grand"       "Gunnison"    "Hinsdale"    "Huerfano"   
[31] "Jackson"     "Jefferson"   "Kiowa"       "Kit Carson"  "Lake"       
[36] "La Plata"    "Larimer"     "Las Animas"  "Lincoln"     "Logan"      
[41] "Mesa"        "Mineral"     "Moffat"      "Montezuma"   "Montrose"   
[46] "Morgan"      "Otero"       "Ouray"       "Park"        "Phillips"   
[51] "Pitkin"      "Prowers"     "Pueblo"      "Rio Blanco"  "Rio Grande" 
[56] "Routt"       "Saguache"    "San Juan"    "San Miguel"  "Sedgwick"   
[61] "Summit"      "Teller"      "Washington"  "Weld"        "Yuma"       
```

Base R functions you might see: `length` and `unique`

These functions require a column as a vector using `pull()`.

```
CO_heat_ER_loc %>% unique() %>% length() # similar to n_distinct()
```

```
[1] 65
```

* New! * Many dplyr functions now have a `.by=` argument

Pipe `CO_heat_ER` into `group_by`, then pipe that into `summarize`:

```
CO_heat_ER %>%
  group_by(county) %>%
  summarize(avg_visits = mean(visits, na.rm = TRUE),
            max_visits = max(visits, na.rm = TRUE))
```

is the same as..

```
CO_heat_ER %>%
  summarize(avg_visits = mean(visits, na.rm = TRUE),
            max_visits = max(visits, na.rm = TRUE),
            .by = county)
```

`summary()` vs. `summarize()`

- `summary()` (base R) gives statistics table on a dataset.
- `summarize()` (dplyr) creates a more customized summary tibble/dataframe.

Summary & Lab Part 2

- `count(x)`: what unique values do you have?
  - `distinct()`: what are the distinct values?
  - `n_distinct()` with `pull()`: how many distinct values?
- `group_by()`: changes all subsequent functions
  - combine with `summarize()` to get statistics per group
  - combine with `mutate()` to add column
- `summarize()` with `n()` gives the count (NAs included)

Class Website

Lab

Extra Slides: More advanced summarization

Data Summarization on data frames

- Statistical summarization across the data frame
    - `rowMeans(x)`: takes the means of each row of x
    - `colMeans(x)`: takes the means of each column of x
    - `rowSums(x)`: takes the sum of each row of x
    - `colSums(x)`: takes the sum of each column of x

```
yearly_co2 <- yearly_co2_emissions
```

`rowMeans()` example

Get means for each row.

Let's see what the mean CO2 emissions is across years for each row (country):

```
yearly_co2 %>%
  select(starts_with("201")) %>%
  rowMeans(na.rm = TRUE) %>%
  head(n = 5)
```

```
[1]  10254   5106 129800    487  32040
```

```
yearly_co2 %>%
  group_by(country) %>%
  summarize(mean = rowMeans(across(starts_with("201")), na.rm = TRUE)) %>%
  head(n = 5)
```

```
# A tibble: 5 × 2
  country       mean
  <chr>        <dbl>
1 Afghanistan  10254
2 Albania       5106
3 Algeria     129800
4 Andorra        487
5 Angola       32040
```

`colMeans()` example

Get means for each column.

Let’s see what the mean is across each column (year):

```
yearly_co2 %>%
  select(starts_with("201")) %>%
  colMeans(na.rm = TRUE) %>%
  head(n = 5)
```

```
    2010     2011     2012     2013     2014 
165334.1 171764.9 174033.4 174856.2 175992.5 
```

```
yearly_co2 %>%
  summarize(across(starts_with("201"), ~mean(.x, na.rm = TRUE)))
```

```
# A tibble: 1 × 5
   `2010`  `2011`  `2012`  `2013`  `2014`
    <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
1 165334. 171765. 174033. 174856. 175993.
```