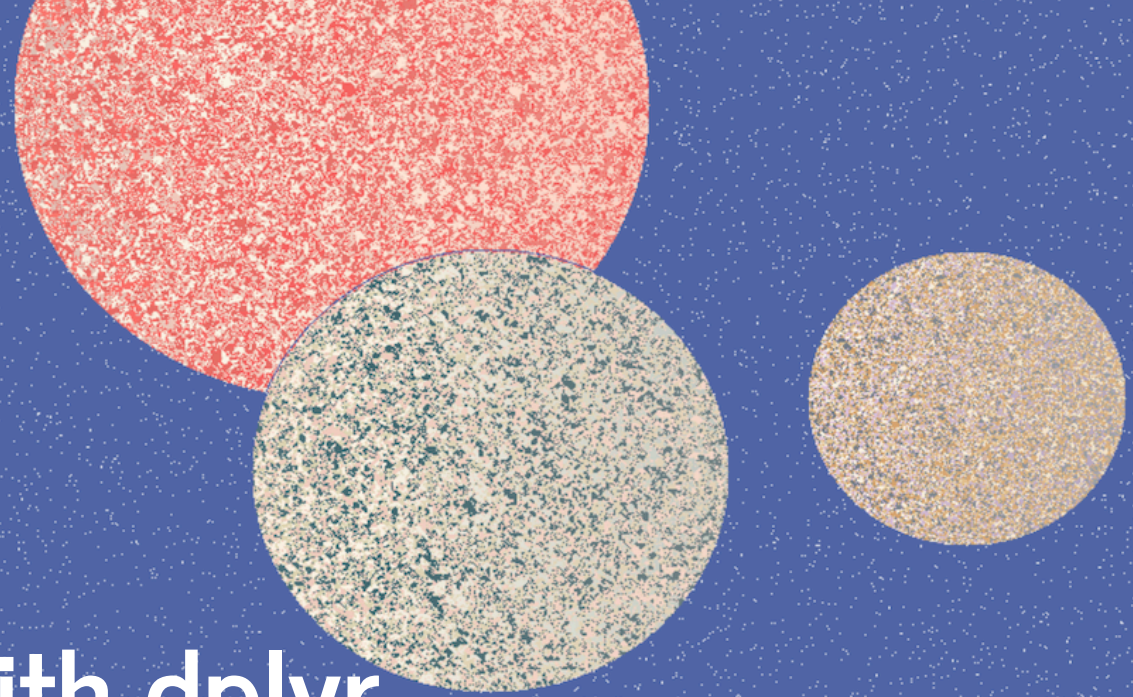

Data Manipulation with dplyr

ID 529: Data Management and Analytic Workflows in R

Dean Marengi | Wednesday, January 15th, 2025

Motivation

- **We've now learned a bit about:**
  - **Fundamentals** of R programming using base R syntax
  - **Importing** data into R
  - **Visualizing** data using `ggplot`
- **Data are rarely free of issues when they are first collected or received**
  - We need efficient tools to process and clean them!
- **Base R is very powerful for data manipulation, but can be difficult to write and read**
  - Complex code that's time-consuming to write can threaten reproducibility
  - `dplyr` and other R packages emphasize writing clean, readable code
- **We can leverage these R packages to:**
  - Write efficient code to perform most data manipulation tasks
  - Chain together data manipulation operations in a concise sequence

Learning objectives

- **Understand the basic principles of dplyr**
  - Core functions for data manipulation
- **Learn how to implement `dplyr` functions to prepare data for analysis**
  - Identifying data quality issues
  - Restructuring and organizing data
  - Deriving new variables
- **Learn about other functions that help core `dplyr` functions perform specific tasks**
  - Transforming multiple columns at once
  - Selecting multiple columns at once
  - Using conditional logic to create new columns

What is data manipulation?

- **The common tasks**
  - Cleaning and renaming variables
  - Selecting a subset of columns to work with from a larger dataset
  - Creating new variables (e.g., based on conditionals or calculations involving other columns)
  - Filtering data for a subset of rows (e.g., based on a specific group)
  - Summarizing data

`dplyr::` a grammar of data manipulation

`dplyr` overview

- Part of the core `tidyverse` package ecosystem
- Functions for performing common data manipulation tasks
- Fast and efficient with concise syntax
  - Chain together data cleaning steps
  - Improves code readability
- **Core single table functions (verbs):**
  - `rename()`: Modify variable names
  - `select()`: Pick variables by name
  - `mutate()`: Create new or modify existing variables
  - `filter()`: Subset observations using conditionals
  - `arrange()`: Reorder observations based on data
  - `summarize()`: Reduce rows into a summary value

`dplyr` syntax overview

- First argument in all `dplyr` functions is always a data frame or tibble
- Variables referenced by name and without quotes (not `df$variable`)
- `dplyr` functions always return a new data frame or tibble
- **Uses the `%>%` or `|>` ("pipe") operator**
  - Can "pipe" function output from one data manipulation step to the next
  - Produces clean, readable code that reads from left to right, top to bottom
  - **Note:** The `|>` reads as "then"

Example

```
# Print first 10 rows
print(our_data, n = 10)
```

```
# A tibble: 15 × 3
      id group                num_gold_stars
   <int> <chr>                         <int>
 1     2 i <3 dplyr                      103
 2    10 dplyr rocks                      96
 3     7 dplyr rocks                     102
 4    11 i <3 dplyr                      106
 5     5 i <3 dplyr                       97
 6    12 clean code or bust              99
 7    15 clean code or bust              93
 8     8 i <3 dplyr                       92
 9     3 clean code or bust              110
10     9 clean code or bust               93
# ℹ 5 more rows
```

```
# Take our data, which is stored as a tibble
new_data <- our_data |>
  # THEN filter our data for a subset of rows
  filter(group == "i <3 dplyr") |>
  # THEN arrange the result by number of gold stars
  arrange(desc(num_gold_stars))
```

```
# A tibble: 5 × 3
     id group      num_gold_stars
  <int> <chr>                <int>
1    11 i <3 dplyr            106
2    14 i <3 dplyr            104
3     2 i <3 dplyr            103
4     5 i <3 dplyr             97
5     8 i <3 dplyr             92
```

dplyr vs base R syntax

dplyr	base
`arrange(df, x)`	`df[order(x), , drop = FALSE]`
`distinct(df, x)`	`df[!duplicated(x), , drop = FALSE]`, `unique()`
`filter(df, x)`	`df[which(x), , drop = FALSE]`, `subset()`
`mutate(df, z = x + y)`	`df$z <- df$x + df$y`, `transform()`
`pull(df, 1)`	`df[[1]]`
`pull(df, x)`	`df$x`
`rename(df, y = x)`	`names(df)[names(df) == "x"] <- "y"`
`relocate(df, y)`	`df[union("y", names(df))]`
`select(df, x, y)`	`df[c("x", "y")]`, `subset()`
`select(df, starts_with("x"))`	`df[grepl(names(df), "^x")]`
`summarise(df, mean(x))`	`mean(df$x)`, `tapply()`, `aggregate()`, `by()`
`slice(df, c(1, 2, 5))`	`df[c(1, 2, 5), , drop = FALSE]`

Example dataset

Overview

- NHANES dataset available on the ID 529 GitHub
- The dataset includes individual-level:
  - Demographic and clinical characteristics
  - Socioeconomic parameters
  - Blood measures of PFAS/PFOA
  - Dietary intake parameters
- **For our examples, we will include a subset of these variables**
  - `id`, `age`, `race_ethnicity`
  - `mean_BP`, `height`, `weight`
  - `pfos`, `pfoa`, `pfna`, `pfhs`, `pfde`

```
1 glimpse(data, width = 50)
```

```
Rows: 2,339
Columns: 12
$ id             <chr> "73568", "73571", "73574"…
$ age            <int> 26, 76, 33, 16, 32, 18, 1…
$ race_ethnicity <fct> Non-Hispanic White, Non-H…
$ mean_BP        <dbl> 104.6667, 126.0000, 121.3…
$ height         <dbl> 152.5, 172.5, 158.0, 170.…
$ weight         <dbl> 47.1, 102.4, 56.8, 67.3, …
$ poverty_ratio  <dbl> 5.00, 5.00, 2.10, 1.58, 0…
$ PFOS           <dbl> 2.2, 10.2, NA, 4.7, 3.0, …
$ PFOA           <dbl> 3.00, 4.77, NA, 2.37, 1.4…
$ PFNA           <dbl> 0.5, 1.3, 0.7, 0.6, 0.4, …
$ PFHS           <dbl> 3.0, 2.0, 0.2, 7.6, 1.2, …
$ PFDE           <dbl> 0.2, 0.3, 0.1, 0.2, 0.1, …
```

dplyr::rename()

Function(s)

```
rename(.data, ...)
rename_with(.data, .fn, .cols = everything(), ...)
```

Main arguments

- `.data` = a data frame
- `rename()`
  - `...` = variables to replace
    - format: `new_name` = `old_name`
- `rename_with()`
  - `.fn` = function to apply over multiple selected columns
  - `.cols` = the columns to rename

Description

- **Rename individual variables, or multiple variables by applying a function**

Examples

```
# Explicitly rename variables in the dataset
rename(data,
       sbp = mean_BP,
       pov_ratio = poverty_ratio,
       race_eth = race_ethnicity,
       ) |>
  glimpse()
```

```
Rows: 2,339
Columns: 12
$ id        <chr> "73568", "73571", "73574", "73…
$ age       <int> 26, 76, 33, 16, 32, 18, 13, 14…
$ race_eth  <fct> Non-Hispanic White, Non-Hispan…
$ sbp       <dbl> 104.6667, 126.0000, 121.3333, …
$ height    <dbl> 152.5, 172.5, 158.0, 170.4, 16…
$ weight    <dbl> 47.1, 102.4, 56.8, 67.3, 79.7,…
$ pov_ratio <dbl> 5.00, 5.00, 2.10, 1.58, 0.29, …
$ PFOS      <dbl> 2.2, 10.2, NA, 4.7, 3.0, NA, 7…
$ PFOA      <dbl> 3.00, 4.77, NA, 2.37, 1.47, NA…
$ PFNA      <dbl> 0.5, 1.3, 0.7, 0.6, 0.4, NA, 0…
$ PFHS      <dbl> 3.0, 2.0, 0.2, 7.6, 1.2, NA, 0…
$ PFDE      <dbl> 0.2, 0.3, 0.1, 0.2, 0.1, NA, 0…
```

dplyr::rename() cont.

```
# Convert all columns names to lower case
rename_with(data, .fn = tolower) |>
  glimpse()
```

```
Rows: 2,339
Columns: 12
$ id             <chr> "73568", "73571", "73574", "73576", "73577", "73578", "73584", "73587", "73…
$ age            <int> 26, 76, 33, 16, 32, 18, 13, 14, 50, 20, 13, 37, 69, 16, 43, 36, 31, 80, 56,…
$ race_ethnicity <fct> Non-Hispanic White, Non-Hispanic White, NA, Non-Hispanic Black, Hispanic, H…
$ mean_bp        <dbl> 104.6667, 126.0000, 121.3333, 109.3333, 119.3333, 122.6667, 109.3333, 112.0…
$ height         <dbl> 152.50, 172.50, 158.00, 170.40, 166.20, 175.20, 144.90, 168.80, 180.50, 165…
$ weight         <dbl> 47.1, 102.4, 56.8, 67.3, 79.7, 109.4, 53.1, 110.2, 104.4, 86.7, 44.9, 126.2…
$ poverty_ratio  <dbl> 5.00, 5.00, 2.10, 1.58, 0.29, 0.58, 3.07, 3.33, 2.18, NA, 1.52, 0.63, 2.44,…
$ pfos           <dbl> 2.2, 10.2, NA, 4.7, 3.0, NA, 7.0, 35.5, NA, 4.7, 4.5, 6.3, 2.5, NA, NA, 2.0…
$ pfoa           <dbl> 3.00, 4.77, NA, 2.37, 1.47, NA, 2.37, 6.17, NA, 1.80, 1.87, 1.67, 2.87, NA,…
$ pfna           <dbl> 0.5, 1.3, 0.7, 0.6, 0.4, NA, 0.8, 3.3, NA, 0.5, 1.7, 0.5, 1.0, NA, 0.7, 0.3…
$ pfhs           <dbl> 3.0, 2.0, 0.2, 7.6, 1.2, NA, 0.8, 6.3, NA, 1.6, 0.8, 1.6, 2.1, NA, 3.6, 0.6…
$ pfde           <dbl> 0.20, 0.30, 0.10, 0.20, 0.10, NA, 0.20, 1.70, NA, 0.20, 0.20, 0.20, 0.30, N…
```

```
# Convert column names that start with PF to upper case
rename_with(data, .fn = toupper, starts_with("pf")) |>
  glimpse()
```

```
Rows: 2,339
Columns: 12
$ id             <chr> "73568", "73571", "73574", "73576", "73577", "73578", "73584", "73587", "73…
$ age            <int> 26, 76, 33, 16, 32, 18, 13, 14, 50, 20, 13, 37, 69, 16, 43, 36, 31, 80, 56,…
$ race_ethnicity <fct> Non-Hispanic White, Non-Hispanic White, NA, Non-Hispanic Black, Hispanic, H…
$ mean_BP        <dbl> 104.6667, 126.0000, 121.3333, 109.3333, 119.3333, 122.6667, 109.3333, 112.0…
$ height         <dbl> 152.50, 172.50, 158.00, 170.40, 166.20, 175.20, 144.90, 168.80, 180.50, 165…
$ weight         <dbl> 47.1, 102.4, 56.8, 67.3, 79.7, 109.4, 53.1, 110.2, 104.4, 86.7, 44.9, 126.2…
$ poverty_ratio  <dbl> 5.00, 5.00, 2.10, 1.58, 0.29, 0.58, 3.07, 3.33, 2.18, NA, 1.52, 0.63, 2.44,…
$ PFOS           <dbl> 2.2, 10.2, NA, 4.7, 3.0, NA, 7.0, 35.5, NA, 4.7, 4.5, 6.3, 2.5, NA, NA, 2.0…
$ PFOA           <dbl> 3.00, 4.77, NA, 2.37, 1.47, NA, 2.37, 6.17, NA, 1.80, 1.87, 1.67, 2.87, NA,…
$ PFNA           <dbl> 0.5, 1.3, 0.7, 0.6, 0.4, NA, 0.8, 3.3, NA, 0.5, 1.7, 0.5, 1.0, NA, 0.7, 0.3…
$ PFHS           <dbl> 3.0, 2.0, 0.2, 7.6, 1.2, NA, 0.8, 6.3, NA, 1.6, 0.8, 1.6, 2.1, NA, 3.6, 0.6…
$ PFDE           <dbl> 0.20, 0.30, 0.10, 0.20, 0.10, NA, 0.20, 1.70, NA, 0.20, 0.20, 0.20, 0.30, N…
```

dplyr::filter()

Function(s)

```
filter(.data, ...)
```

Main arguments

- `.data` = a data frame
- `...` = Expressions for filtering the data frame, which evaluate to TRUE or FALSE

Description

- **Subsets observations based on their values**
- Expressions use operators
  - **Comparison:** >, >=, <, <=, !=, ==
  - **Logical:** !, &, |, xor
  - **Binary:** %in%
- Returns a data frame with a subset of rows where conditions evaluated to TRUE

Examples

```
# Filter for ages greater than or equal to 50
filter(data, age >= 50)

# Filter for ages between 18 and 60
filter(data, age  %in% c(18:60))
```

```
# Filter for Hispanic participants only
filter(data, race_ethnicity == "Hispanic")

# Filter for Hispanic participants between
# 18 and 60 years old
filter(data,
       race_ethnicity == "Hispanic" & age %in% c(18:60)
       )
```

```
# Subset data for participants whose age
# is not greater than 60 and mean SBP
# is not greater than 120
filter(data, !(age > 60 | mean_BP > 120))

# This logic achieves the same result
# Here ',' is equivalent to '&'
filter(data, age <= 60, mean_BP <= 120)
```

```
# Subset data for observations where poverty
# ratio IS missing (NA)
filter(data, is.na(poverty_ratio))

# Subset data for observations where poverty
# ratio is NOT missing (NA)
filter(data, !is.na(poverty_ratio))
```

dplyr::filter()

```
1 # Subset data for observations where poverty
2 # ratio IS missing (NA)
3 filter(data, is.na(poverty_ratio))
```

```
# A tibble: 203 × 12
   id      age race_ethnicity     mean_BP height weight poverty_ratio  PFOS  PFOA  PFNA  PFHS  PFDE
   <chr> <int> <fct>                <dbl>  <dbl>  <dbl>         <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 73598    20 Hispanic               112    165    86.7            NA   4.7  1.8    0.5   1.6  0.2
 2 73699    20 Non-Hispanic White      98    184.   77              NA   5.6  4.6    0.5   4.4  0.1
 3 73703    38 Hispanic                99.3  156.   63.7            NA   5    0.97   0.5   1.6  0.2
 4 73724    26 Non-Hispanic Black     117.   184.   65.1            NA  14.8  2.9    0.8   2.1  0.1
 5 73733    23 Hispanic               112    154    75.8            NA   2.9  1.27   0.4   1.1  0.1
 6 73756    56 Non-Hispanic White     119.   171.   75              NA   4.6  3.2    0.5   0.9  0.2
 7 73774    80 Non-Hispanic Black     NaN    159.   58.9            NA   6.5  1.97   0.8   2.7  0.3
 8 73872    63 <NA>                   118    158.   70.3            NA  11.7  1.87   1.5   1    3.3
 9 73914    52 <NA>                   111.   160.   63.6            NA   1.4  1.07   0.4   0.6  0.07
10 73935    63 Hispanic               130    174.   73.6            NA   5.2  2.6    0.9   2.6  0.2
# ℹ 193 more rows
```

dplyr::arrange()

Function(s)

```
arrange(.data, ..., .by_group = FALSE)
```

Main arguments

- `.data` = a data frame
- `...` = Variables or expressions to order rows by
- `.by_group` = Sort by first grouping variable (grouped data frames, only)

Description

Examples

```
# Take the data frame (data)
# Then arrange rows by age (in ascending order)
# Then print the first 5 rows
arrange(data, age) |>
  head(n = 5)
```

```
# A tibble: 5 × 12
  id      age race_ethnicity mean_BP height weight
  <chr> <int> <fct>            <dbl>  <dbl>  <dbl>
1 74182    12 Hispanic          98.7   148.   37.6
2 74339    12 Hispanic         110     157.   52.7
3 74659    12 Non-Hispanic …    96.7   156.   48.5
4 74753    12 Non-Hispanic …   107.    160    46.6
5 74795    12 Hispanic          98.7   161.   71.2
# ℹ 6 more variables: poverty_ratio <dbl>,
#   PFOS <dbl>, PFOA <dbl>, PFNA <dbl>,
#   PFHS <dbl>, PFDE <dbl>
```

```
# Take the data frame (data)
# Then arrange rows by age, then mean SBP (descending)
# Then print the first 5 rows
arrange(data, desc(age), desc(mean_BP)) |>
  head(n = 5)
```

```
# A tibble: 5 × 12
  id      age race_ethnicity mean_BP height weight
  <chr> <int> <fct>            <dbl>  <dbl>  <dbl>
1 79199    80 Hispanic           196    154.   77.1
2 83662    80 Non-Hispanic …     194     NA    85.2
3 73747    80 Hispanic           193.   161.   78.4
4 77661    80 Non-Hispanic …     186    153.   95.8
5 79095    80 <NA>               186    145.   62.2
# ℹ 6 more variables: poverty_ratio <dbl>,
#   PFOS <dbl>, PFOA <dbl>, PFNA <dbl>,
#   PFHS <dbl>, PFDE <dbl>
```

dplyr::select()

Function(s)

```
select(.data, ..., .by_group = FALSE)
```

Main arguments

- `.data` = a data frame
- `...` = Variable name(s) and/or expressions to select columns

Description

- **Selects variables in a data frame**
  - Variable names referenced without quotes
  - `select` **helper functions** can select columns using specific operations
    - E.g., select variables where column names contain the string "bp"

Examples

```
 [1] "id"              "age"             "race_ethnicity"
 [4] "mean_BP"         "height"          "weight"
 [7] "poverty_ratio"   "PFOS"            "PFOA"
[10] "PFNA"            "PFHS"            "PFDE"
```

```
# From the data frame, select columns explicitly by name
data |>
  select(age, height, weight, mean_BP) |>
  head(n = 4)
```

```
# A tibble: 4 × 4
    age height weight mean_BP
  <int>  <dbl>  <dbl>   <dbl>
1    26   152.   47.1    105.
2    76   172.  102.     126
3    33   158    56.8    121.
4    16   170.   67.3    109.
```

```
# From the data frame, select sequences of consecutive columr
data |>
  select(id:race_ethnicity, PFOS:PFDE) |>
  head(n = 4)
```

```
# A tibble: 4 × 8
  id      age race_ethnicity      PFOS  PFOA  PFNA  PFHS  PFDE
  <chr> <int> <fct>              <dbl> <dbl> <dbl> <dbl> <dbl>
1 73568    26 Non-Hispanic White   2.2  3      0.5   3     0.2
2 73571    76 Non-Hispanic White  10.2  4.77   1.3   2     0.3
3 73574    33 <NA>                NA    NA     0.7   0.2   0.1
4 73576    16 Non-Hispanic Black   4.7  2.37   0.6   7.6   0.2
```

`dplyr::select()` helpers

- **Select variables by matching patterns in their names**
  - `everything()`: Matches all variables.
  - `last_col()`: Select last variable, possibly with an offset.
- **Select variables by matching patterns in their names**
  - `starts_with()`: Starts with a prefix
  - `ends_with()`: Ends with a suffix
  - `contains()`: Contains a literal string
  - `matches()`: Matches a regular expression

https://dplyr.tidyverse.org/reference/select.html

- **Select variables from a character vector:**
  - `all_of()`: Matches variable names in a character vector. All names must be present, otherwise an out-of-bounds error is thrown.
  - `any_of()`: Same as all_of(), except that no error is thrown for names that don't exist.

dplyr::select() helpers

```r
# From the data frame, select id and columns that
# are prefixed with PF
data |>
  select(id, starts_with("PF")) |>
  head(n = 4)
```

```
# A tibble: 4 × 6
  id     PFOS  PFOA  PFNA  PFHS  PFDE
  <chr> <dbl> <dbl> <dbl> <dbl> <dbl>
1 73568   2.2  3      0.5   3     0.2
2 73571  10.2  4.77   1.3   2     0.3
3 73574  NA   NA      0.7   0.2   0.1
4 73576   4.7  2.37   0.6   7.6   0.2
```

```r
# Bring ID and PFAS variables to the beginning of the dataset
# then position all non-selected variables after them
data |>
  select(id, starts_with("PF"), everything()) |>
  head(n = 4)
```

```
# A tibble: 4 × 12
  id     PFOS  PFOA  PFNA  PFHS  PFDE   age race_ethnicity     mean_BP height weight poverty_ratio
  <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <int> <fct>                <dbl>  <dbl>  <dbl>         <dbl>
1 73568   2.2  3      0.5   3     0.2    26 Non-Hispanic White   105.   152.   47.1           5
2 73571  10.2  4.77   1.3   2     0.3    76 Non-Hispanic White   126    172.  102.            5
3 73574  NA   NA      0.7   0.2   0.1    33 <NA>                 121.   158    56.8           2.1
4 73576   4.7  2.37   0.6   7.6   0.2    16 Non-Hispanic Black   109.   170.   67.3           1.58
```

dplyr::mutate()

Function(s)

```
mutate(.data, ...)
```

Main arguments

- `.data` = a data frame
- `...` = Name value pairs

Description

- **Changes the values of columns and creates new columns.**
  - i.e., adds new variables
- Will overwrite variables that share the same name

Examples

```
# Derive new variables
data |>
  mutate(weight_lbs = round(weight * 2.20462, 2),
         height_in = round(height * 0.393701, 2),
         age_60 = if_else(age >= 60, 1, 0)
         ) |>
  select(id, age, age_60, height, height_in, weight, weight_lbs)
```

```
# A tibble: 2,339 × 7
   id      age age_60 height height_in weight weight_lbs
   <chr> <int>  <dbl>  <dbl>     <dbl>  <dbl>      <dbl>
 1 73568    26      0   152.      60.0   47.1       104.
 2 73571    76      1   172.      67.9  102.        226.
 3 73574    33      0   158       62.2   56.8       125.
 4 73576    16      0   170.      67.1   67.3       148.
 5 73577    32      0   166.      65.4   79.7       176.
 6 73578    18      0   175.      69.0  109.        241.
 7 73584    13      0   145.      57.0   53.1       117.
 8 73587    14      0   169.      66.5  110.        243.
 9 73597    50      0   180.      71.1  104.        230.
10 73598    20      0   165       65.0   86.7       191.
# ℹ 2,329 more rows
```

dplyr::summarize()

Function(s)

```
summarize(.data, ...)
```

Main arguments

- `.data` = a data frame
- `...` = Name value pairs of the summary function(s) to apply

Description

- **Creates a new data frame based on one or more summary functons**
  - Data frame has >1 rows if the data are grouped
  - Has single row if there are no groups supplied

Examples

```
# Compute summary statistics for age by race/ethnicity
data |>
  filter(!is.na(race_ethnicity)) |>
  group_by(race_ethnicity) |>
  summarize(age_mean = mean(age, na.rm = T),
            age_sd  = sd(age, na.rm = T),
            age_min = min(age, na.rm = T),
            age_max = max(age, na.rm = T))
```

```
# A tibble: 3 × 5
  race_ethnicity   age_mean age_sd age_min age_max
  <fct>               <dbl>  <dbl>   <int>   <int>
1 Non-Hispanic Wh…     47.1   21.4      12      80
2 Non-Hispanic Bl…     41.2   21.0      12      80
3 Hispanic             39.7   20.4      12      80
```

Example

Original nhanes_ID529 dataset

```
[1] "id | race_ethnicity | sex_gender | age | poverty_ratio | days_dental_floss | PFAS_total | PFOS | PFOA | PFNA | PFHS | PFDE |
total_energy | fast_food_energy_no_popcorn_no_seafood | restaurant_energy_no_popcorn_no_seafood |
non_fast_food_or_restaurant_energy_no_popcorn_no_seafood | popcorn_energy | shellfish_energy | fish_energy | mean_BP | weight |
height"
```

```r
# Print the data frame
head(nhanes_id529, n = 100)
```

	id	race_ethnicity	sex_gender	age	poverty_ratio	days_dental_floss	PFAS_total	PFOS	PFOA	PFNA	PFHS	PFDE	total_energy
1	73568	Non-Hispanic White	Female	26	5.00	NA	8.90	2.2	3.00	0.5	3.0	0.20	3145
2	73571	Non-Hispanic White	Male	76	5.00	2	18.57	10.2	4.77	1.3	2.0	0.30	1076
3	73574	<NA>	Female	33	2.10	7	NA	NA	NA	0.7	0.2	0.10	5621
4	73576	Non-Hispanic Black	Male	16	1.58	NA	15.47	4.7	2.37	0.6	7.6	0.20	1012
5	73577	Hispanic	Male	32	0.29	0	6.17	3.0	1.47	0.4	1.2	0.10	3194
6	73578	Hispanic	Male	18	0.58	NA	NA	NA	NA	NA	NA	NA	955
7	73584	Non-Hispanic White	Male	13	3.07	NA	11.17	7.0	2.37	0.8	0.8	0.20	663
8	73587	<NA>	Male	14	3.33	NA	52.97	35.5	6.17	3.3	6.3	1.70	875
9	73597	Non-Hispanic Black	Female	50	2.18	0	NA	NA	NA	NA	NA	NA	1239
10	73598	Hispanic	Male	20	NA	NA	8.80	4.7	1.80	0.5	1.6	0.20	4865
11	73599	Non-Hispanic White	Female	13	1.52	NA	9.07	4.5	1.87	1.7	0.8	0.20	1540
12	73600	Non-Hispanic Black	Male	37	0.63	0	10.27	6.3	1.67	0.5	1.6	0.20	2857
13	73604	Non-Hispanic White	Female	69	2.44	7	8.77	2.5	2.87	1.0	2.1	0.30	1301
14	73606	Non-Hispanic White	Male	16	3.23	NA	NA	NA	NA	NA	NA	NA	1692
15	73610	Non-Hispanic White	Male	43	2.03	3	NA	NA	NA	0.7	3.6	0.20	1783
16	73619	Hispanic	Female	36	0.84	0	3.97	2.0	0.97	0.3	0.6	0.10	1331
17	73624	Non-Hispanic White	Female	31	0.47	0	3.34	1.9	0.67	0.2	0.5	0.07	1723
18	73628	Non-Hispanic White	Male	80	3.00	7	22.60	12.4	5.50	1.1	3.3	0.30	1972
19	73631	Non-Hispanic Black	Female	56	0.71	0	4.57	2.1	1.17	0.6	0.5	0.20	2117
20	73633	Non-Hispanic White	Female	43	3.56	0	21.77	9.0	7.97	2.8	1.4	0.60	1391
21	73639	Non-Hispanic White	Male	71	1.45	0	25.27	13.3	6.37	1.5	3.6	0.50	1298
22	73642	Non-Hispanic White	Female	57	2.27	0	9.07	5.4	1.97	0.9	0.5	0.30	769
23	73647	Hispanic	Female	61	3.53	0	3.64	1.4	1.17	0.2	0.8	0.07	739
24	73655	Non-Hispanic Black	Male	44	1.79	4	8.57	5.7	0.97	0.8	1.0	0.10	5012
25	73663	Non-Hispanic White	Male	25	5.00	NA	12.80	7.1	3.10	0.9	1.5	0.20	1880

Notice that the standard data frame does not print with information about data types, dataset dimensions, etc. Additionally, constraints of the slide width prevent information about other columns in the data frame from being displayed. Standard data frames run into similar print issues in the R console, which can make them difficult to work with

Example

```
# Coerce the data frame to a tibble data frame, and assign it to 'data'
data <- as_tibble(nhanes_id529)

# Print the tibble data frame
data
```

```
# A tibble: 2,339 × 22
   id    race_ethnicity     sex_gender   age poverty_ratio days_dental_floss
   <chr> <fct>              <fct>      <int>         <dbl>              <int>
 1 73568 Non-Hispanic White Female        26           5                    NA
 2 73571 Non-Hispanic White Male          76           5                     2
 3 73574 <NA>               Female        33           2.1                   7
 4 73576 Non-Hispanic Black Male          16           1.58                 NA
 5 73577 Hispanic           Male          32           0.29                  0
 6 73578 Hispanic           Male          18           0.58                 NA
 7 73584 Non-Hispanic White Male          13           3.07                 NA
 8 73587 <NA>               Male          14           3.33                 NA
 9 73597 Non-Hispanic Black Female        50           2.18                  0
10 73598 Hispanic           Male          20          NA                    NA
# ℹ 2,329 more rows
# ℹ 16 more variables: PFAS_total <dbl>, PFOS <dbl>, PFOA <dbl>, PFNA <dbl>,
#   PFHS <dbl>, PFDE <dbl>, total_energy <int>,
#   fast_food_energy_no_popcorn_no_seafood <int>,
#   restaurant_energy_no_popcorn_no_seafood <int>,
#   non_fast_food_or_restaurant_energy_no_popcorn_no_seafood <int>,
#   popcorn_energy <int>, shellfish_energy <int>, fish_energy <int>, …
```

Okay, much better! The tibble data frame provides information about the dataset dimensions, data types, and is also more accommodating of the constrained print width (it still prints details about variables that are not able to be fully displayed).

Example

Let's use `dplyr` to organize these data

```
# Print variable names for reference
names(data)
```

```
 [1] "id"
 [2] "race_ethnicity"
 [3] "sex_gender"
 [4] "age"
 [5] "poverty_ratio"
 [6] "days_dental_floss"
 [7] "PFAS_total"
 [8] "PFOS"
 [9] "PFOA"
[10] "PFNA"
[11] "PFHS"
[12] "PFDE"
[13] "total_energy"
[14] "fast_food_energy_no_popcorn_no_seafood"
[15] "restaurant_energy_no_popcorn_no_seafood"
[16] "non_fast_food_or_restaurant_energy_no_popcorn_no_seafood"
[17] "popcorn_energy"
[18] "shellfish_energy"
[19] "fish_energy"
[20] "mean_BP"
[21] "weight"
[22] "height"
```

Clean-up steps

- Change all variable names to lower case
- Explicitly rename a few variables
- Select sequences of columns of interest
  - `pf.*` looks for names that begin with 'pf' (with no specific text thereafter)
- Move the `pfas_total` column to the end
- **Output on next slide!**

```
data <- data |>
  rename_with(~ tolower(.)) |>
  rename(
    race_eth = race_ethnicity,
    mean_sbp = mean_bp,
    pov_ratio = poverty_ratio
    ) |>
  select(id:race_eth,
         mean_sbp:height,
         pov_ratio,
         matches("pf.*")
         ) |>
  relocate(pfas_total, .after = last_col())
data
```

Example

```
# A tibble: 2,339 × 12
   id    race_eth           mean_sbp weight height pov_ratio  pfos  pfoa  pfna  pfhs  pfde pfas_total
   <chr> <fct>                 <dbl>  <dbl>  <dbl>     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>      <dbl>
 1 73568 Non-Hispanic White     105.   47.1   152.       5      2.2  3      0.5   3     0.2        8.9
 2 73571 Non-Hispanic White     126   102.    172.       5     10.2  4.77   1.3   2     0.3       18.6
 3 73574 <NA>                   121.   56.8   158        2.1   NA   NA      0.7   0.2   0.1       NA
 4 73576 Non-Hispanic Black     109.   67.3   170.       1.58   4.7  2.37   0.6   7.6   0.2       15.5
 5 73577 Hispanic               119.   79.7   166.       0.29   3    1.47   0.4   1.2   0.1        6.17
 6 73578 Hispanic               123.  109.    175.       0.58  NA   NA     NA    NA    NA         NA
 7 73584 Non-Hispanic White     109.   53.1   145.       3.07   7    2.37   0.8   0.8   0.2       11.2
 8 73587 <NA>                   112   110.    169.       3.33  35.5  6.17   3.3   6.3   1.7       53.0
 9 73597 Non-Hispanic Black     NaN   104.    180.       2.18  NA   NA     NA    NA    NA         NA
10 73598 Hispanic               112    86.7   165       NA      4.7  1.8    0.5   1.6   0.2        8.8
# ℹ 2,329 more rows
```

Example

Derive a variable for hypertension status

- Use `case_when()` to apply conditional logic
  - Derive hypertension categories

```
data <- data |>
  mutate(htn_cat = factor(
    case_when(
      mean_sbp < 120 ~ "normal",
      mean_sbp >= 120 & mean_sbp < 130 ~ "elevated",
      mean_sbp >= 130 & mean_sbp < 140 ~ "stage1",
      mean_sbp >= 140 ~ "stage2",
      is.na(mean_sbp) ~ NA_character_
    )
  ))

# Print
glimpse(data)
```

```
Rows: 2,339
Columns: 13
$ id         <chr> "73568", "73571", "73574", "7…
$ race_eth   <fct> Non-Hispanic White, Non-Hispa…
$ mean_sbp   <dbl> 104.6667, 126.0000, 121.3333,…
$ weight     <dbl> 47.1, 102.4, 56.8, 67.3, 79.7…
$ height     <dbl> 152.5, 172.5, 158.0, 170.4, 1…
$ pov_ratio  <dbl> 5.00, 5.00, 2.10, 1.58, 0.29,…
$ pfos       <dbl> 2.2, 10.2, NA, 4.7, 3.0, NA, …
$ pfoa       <dbl> 3.00, 4.77, NA, 2.37, 1.47, N…
$ pfna       <dbl> 0.5, 1.3, 0.7, 0.6, 0.4, NA, …
$ pfhs       <dbl> 3.0, 2.0, 0.2, 7.6, 1.2, NA, …
$ pfde       <dbl> 0.2, 0.3, 0.1, 0.2, 0.1, NA, …
$ pfas_total <dbl> 8.90, 18.57, NA, 15.47, 6.17,…
$ htn_cat    <fct> normal, elevated, elevated, n…
```

Compute summary statistics for PFAS variables

- Some methods implemented in this example will be discussed more next week

```
data |>
  # Change to a long data format (more on this soon!)
  pivot_longer(matches("pf.*"),
               names_to = "pfas_parameter",
               values_to = "concentration"
               ) |>
  # Group the data by PFAS variable name
  group_by(pfas_parameter) |>
  # Compute some grouped summary statistics
  summarize(mean = mean(concentration, na.rm = T),
            sd = sd(concentration, na.rm = T),
            med = median(concentration, na.rm = T),
            min = min(concentration, na.rm = T),
            max = max(concentration, na.rm = T)
            ) |>
  arrange(desc(mean))
```

```
# A tibble: 6 × 6
  pfas_parameter   mean     sd   med   min    max
  <chr>           <dbl>  <dbl> <dbl> <dbl>  <dbl>
1 pfas_total      13.5   34.0    9.97  0.49 1423.
2 pfos             8.02  32.7    5.1   0.14 1403
3 pfoa             2.33   3.01   1.87  0.14   85.3
4 pfhs             1.94   2.21   1.3   0.07   33.9
5 pfna             0.871  0.826  0.7   0.07   16.3
6 pfde             0.313  1.21   0.2   0.07   51.3
```

`dplyr` `learnr` tutorials

`learnr` is an R package for creating interactive tutorials with R markdown.

- Install and load the `learnr` package
- Select one of the following three exercises to complete
  - `ex-data-filter`: Filtering observations
  - `ex-data-mutate`: Creating new variables
  - `ex-data-summarise`: Summarizing data

```
# Install the learnr package
install.packages("learnr")

# Load learnr
library("learnr")

# Filtering observations
run_tutorial(name = "ex-data-filter", package = "learnr")

# Creating new variables
run_tutorial(name = "ex-data-mutate", package = "learnr")

# Summarizing data
run_tutorial(name = "ex-data-summarise", package = "learnr")
```

Resources

- **`dplyr` function reference:** https://dplyr.tidyverse.org/reference/index.html
- **`dplyr` documentation:** https://dplyr.tidyverse.org/
- **Intro to `dplyr` vignette:** https://dplyr.tidyverse.org/articles/dplyr.html
- **`dplyr` vs base R vignette:** https://dplyr.tidyverse.org/articles/base.html
- **Programming with `dplyr` vignette:** https://dplyr.tidyverse.org/articles/programming.html
- **Column-wise operations vignette:** https://dplyr.tidyverse.org/articles/colwise.html
- **Row-wise operations vignette:** https://dplyr.tidyverse.org/articles/rowwise.html
- **Window functions vignette:** https://dplyr.tidyverse.org/articles/window-functions.html

- **aRtsy package for creating art using R:** https://cran.r-project.org/web/packages/aRtsy/readme/README.html