R Programming For Natural Resource Professionals

Lecture 4
Data Wrangling I:
Subset and summarizing data

Pseudo-code

- A plain language skeleton of comments that is written out prior to coding it.

Preview the week

Subsetting variables using dplyr

```
tib %>% select(cols)
```

Purpose: subset variables (columns)

- Declare variables using:
  - Their names
  - Their index
    - 1:4 subsets variables 1-4
  - Helper functions
  - Inverse statements
    - !1:4 subsets everything except variables 1-4
  - Drop columns using “-”
    - -colName
- Can be used to rearrange columns

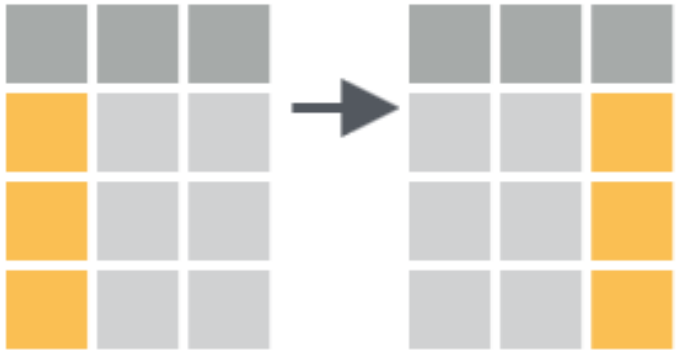
Subsetting variables using dplyr

```
tib %>% pull(cols)
```

Purpose: subset variables (columns) without header

- Just like “$” to subset variable value
- “It's mostly useful because it looks a little nicer in pipes”

Subsetting variables using dplyr

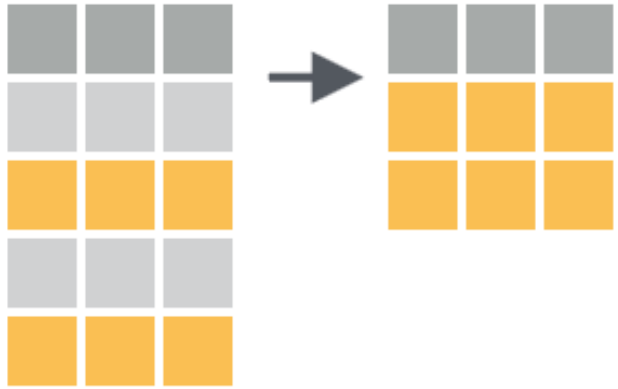


`tib %>% relocate(cols, .before` or `.after)`

Purpose: move columns around in a tibble

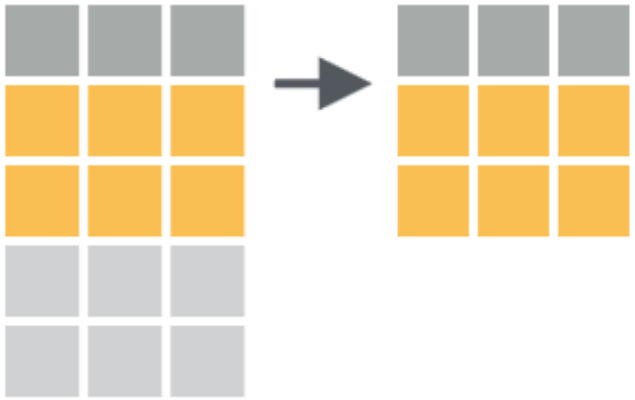
- Can use helper functions
- Remember the "." in front of before/after
- Default relocation is to the first column positions

Subsetting observations using dplyr



Purpose: subset (or sample) observations

```
tib %>% slice(rowIndex)
tib %>% slice_sample(n or proportion)
```



```
tib %>% slice_head(n or proportion)
tib %>% slice_tail(n or proportion)
```

Subsetting observations using dplyr

```
tib %>% distinct()
```

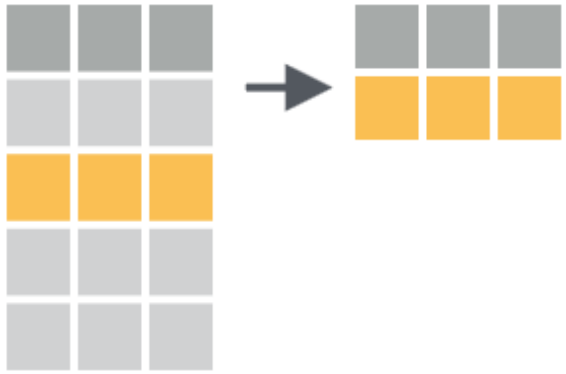
Purpose: Remove rows with duplicate values

No column listed = overall unique observations

`Col` = unique values of that column

`.keep_all` = whether to retain rest of tibble

Subsetting observations using dplyr



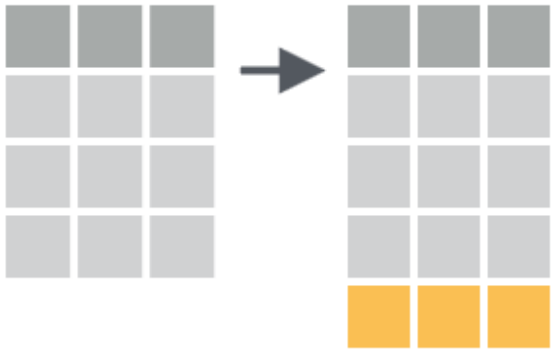
```
tib %>% filter()
```

Purpose: Extract rows meeting certain criteria

Use `&` or `|` for multiple criteria

Other useful operators: `=>`, `=<`, `==`, `!`

Adding observations using dplyr



```
tib %>% add_rows()
```

Purpose: Extract rows meeting certain criteria

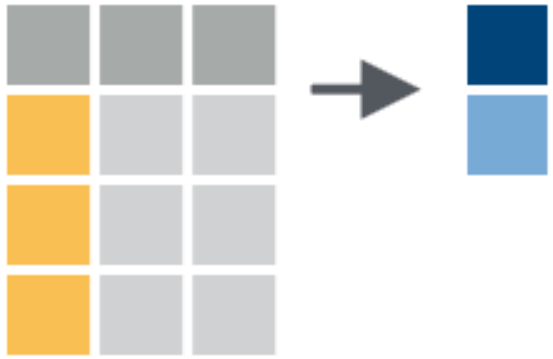
`.before` or `.after` indicates where to add to

Arranging observations using dplyr

```
tib %>% arrange(col)
tib %>% arrange(desc(col))
```

Purpose: Arrange rows based on certain criteria

Summarizing data using dplyr



`tib %>% summarize()`

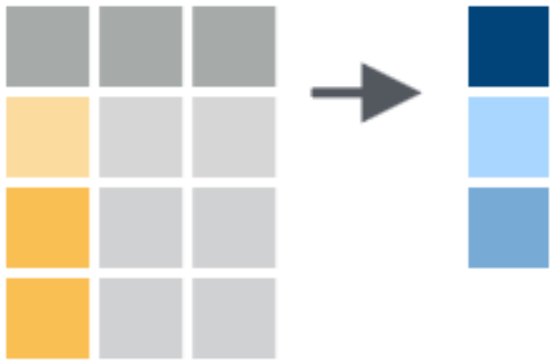
Purpose: Use math or logic to produce data summaries

Examples include:

- `min()`
- `max()`
- `mean()`
- `sd()`
- `+, -, *, /`

- Declare a name for the column first
  - `tib %>% summarize(mean_x = mean(x))`

Summarizing data using dplyr



```
tib %>% count(col)
```

Purpose: a shortcut for summarizing counts

Does same job as: `summarize(n = n(col))`

Summarizing data using dplyr

```
tib %>% group_by()
```

Purpose: Establishing grouping scheme for data summary

```
tib %>%
    group_by(var2) %>%
    summarize(avg = mean(var3))
```

In class exercises

Exercise 1

- Step 1: Pivot the data to start tidying it
- Step 2: Remove NAs
- Step 3: Subset the data to include only one observation per track per artist. Keep all columns after subsetting.
- Step 4: Count the number of distinct tracks for each artist that were on the billboard music chart that year.
- Step 5: Arrange the tibble in order of artist with the most tracks on the chart to the least.

<u>In other words:</u>

- Write a tidyverse pipeline to generate a tibble that displays the number of tracks per artist in the data set.