R Programming For Natural Resource Professionals

Lecture 4
Data Wrangling I:
Subset and summarizing data

Paper discussions

- Open this link now by typing it into a browser: https://bit.ly/3GVPwuV
- Talk with your group to identify one or two thoughts, questions, and epiphanies that resonate then record them in the Google Doc.
- Read through the list as it updates.

This week’s learning objectives

1. Subsetting variables
2. Subsetting observations
3. Generating summary statistics on tibbles

Homework 2 feedback

- Ensure your code is robust enough that you don’t need to transcribe a previous result.

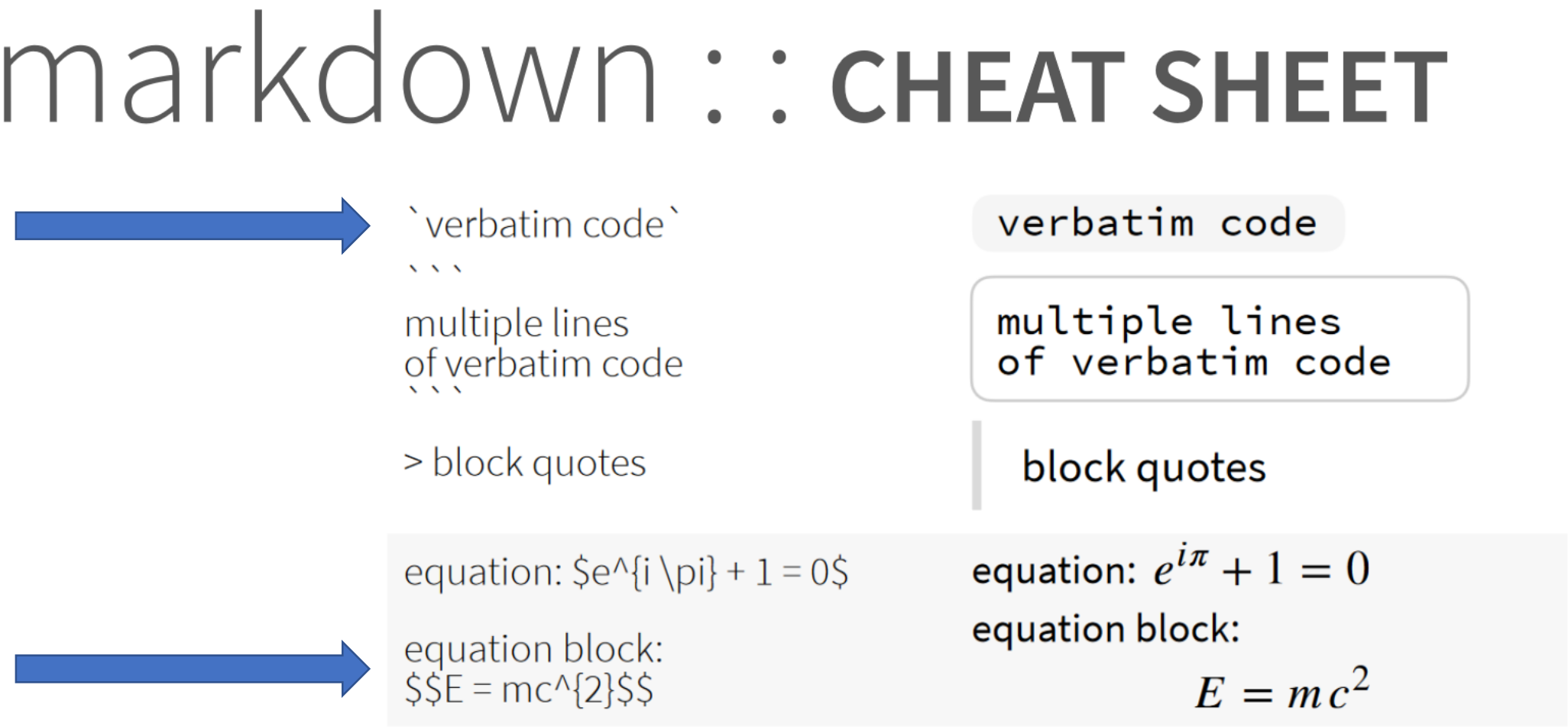
````
Questions 2c. Which bison had the greatest weight in the data set? How old was it at this age? Again, calculate the age by subsetting the data set- not transcribing the values.
```{r}
# Answer 2c
max(bison_dat$animal_weight, na.rm = TRUE)
max_wt_bison <- subset(bison_dat, animal_weight == "2050",)
age_max_wt_bison <- max_wt_bison$rec_year - max_wt_bison$animal_yob
age_max_wt_bison

```
````

````
```{r}
dat <- knz_bison[max(knz_bison$animal_weight, na.rm = TRUE),]
dat$rec_year - dat$animal_yob
```
````

Homework 2 feedback

- Use the cheat sheets!



Pseudo-code

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

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

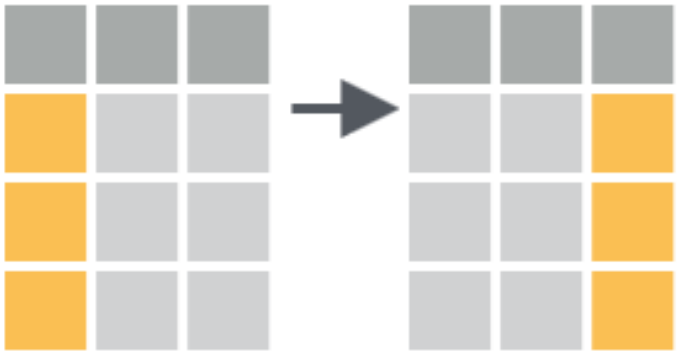
Subsetting variables using dplyr

```
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

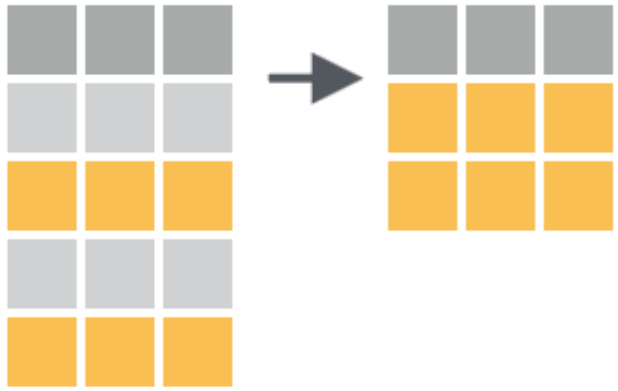


`tib %>% relocate(cols, .before` or `.after)`

Purpose: move columns around in a tibble

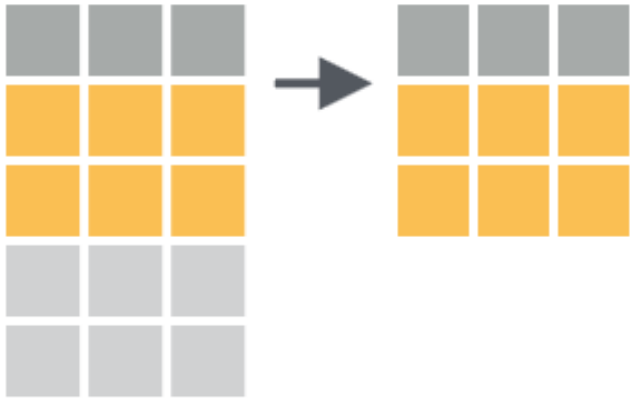
- Can use helper functions
- Remember the "." in front of before/after
- Default relocation is to the first column positions

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()
```

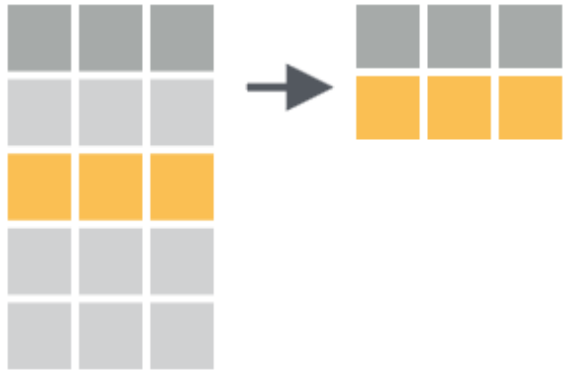
Purpose: Remove rows with duplicate values

No column listed = overall unique observations

`Col` = unique values of that column

`.keep_all` = whether to retain rest of tibble

Subsetting observations using dplyr



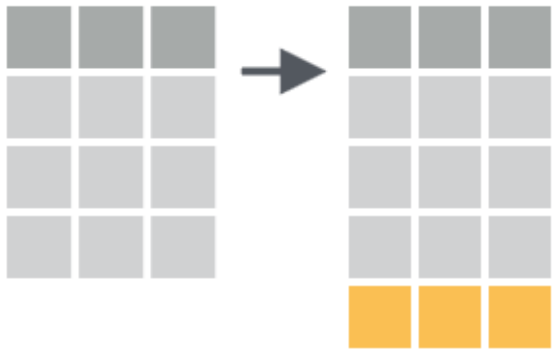
```
tib %>% filter()
```

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

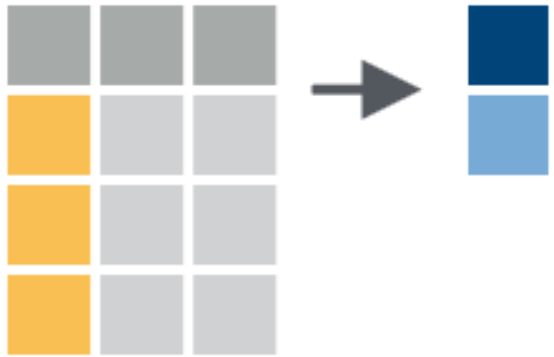
`.before` or `.after` indicates where to add to

Arranging observations using dplyr

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

Purpose: Arrange rows based on certain criteria

Summarizing data using dplyr



`tib %>% summarize()`

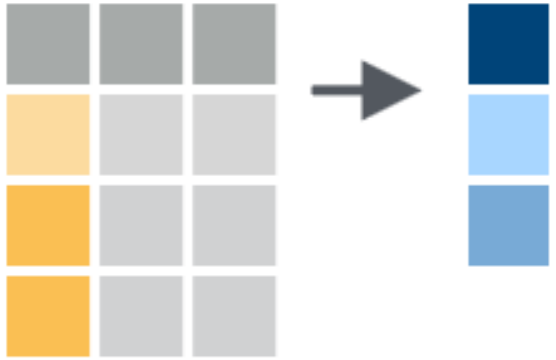
Purpose: Use math or logic to produce data summaries

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))
```