Lecture 9: Rectangular data in R

Content so far

- Simulation
- Iteration
- Vectors and lists
- Functions

What's missing: actual data sets!

Learning goals

- Review/refresh data manipulation from STA 112
- Explore different data objects in R and Python
- Work with more challenging data, requiring more difficult manipulation
- Combine information from multiple datasets
- Learn tools for different data types (strings, factors, dates and times)

Next steps

- Today: overview of matrices and data frames in R; basic data wrangling in R
- Next week: Introduction to Python
- After that: data wrangling with both R and Python

Rectangular data

```r
library(dplyr)
starwars
```

```
# A tibble: 87 × 14
   name     height  mass hair_color skin_color eye_color birth_year sex
gender
   <chr>     <int> <dbl> <chr>      <chr>      <chr>          <dbl>
<chr> <chr>
 1 Luke Sk…    172    77 blond      fair       blue            19   male
mascu…
 2 C-3PO       167    75 <NA>       gold       yellow         112   none
mascu…
 3 R2-D2        96    32 <NA>       white, bl… red             33   none
mascu…
 4 Darth V…    202   136 none       white      yellow          41.9 male
mascu…
 5 Leia Or…    150    49 brown      light      brown           19
```

Rectangular data in R

In R, there are two main ways of storing rectangular data:

- matrices
- data frames

Matrices

A *matrix* generalizes a vector to *two* dimensions:

```r
x <- matrix(c(1, 2, 3, 4, 5, 6), nrow=2)
x
```

```
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

x[2, 3]

- Each row is a vector
- Each column is a vector

Indexing matrices

If you want to fill in rows first: byrow=TRUE

```r
x <- matrix(c(1, 2, 3, 4, 5, 6), nrow=2)
x
```

```
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

```r
x[1,]
```

← everything in the first row

```
[1] 1 3 5
```

```r
x[,1]
```

← everything in the first column

```
[1] 1 2
```

← row #

```r
x[1,2]
```

← column #

```
[1] 3
```

- Use single square brackets [ ] to index
- The first coordinate is the row, the second coordinate is the column

Uses and limitations of matrices

- Correspond to the matrices we know and love from linear algebra
- Usually the right way to store 2-d data for doing math (like matrix multiplication)
- Like vectors, contain only one type of data

```r
x <- matrix(c(1, 2, 3, 'a', 5, 6), nrow=2)
x
```

```
     [,1] [,2] [,3]
[1,] "1"  "3"  "5"
[2,] "2"  "a"  "6"
```

Data frames

naming columns

```r
example_df <- data.frame(x = c(1, 2, 3),
                         y = c('a', 'b', 'c'))

example_df
```

create a data frame

```
  x y
1 1 a
2 2 b
3 3 c
```

x is a column of doubles

y is a column of characters

Aside: what *are* data frames?

```r
example_df <- data.frame(x = c(1, 2, 3),
                         y = c('a', 'b', 'c'))
typeof(example_df)
```

```
[1] "list"
```

- Matrices are like a 2-d vector
- Data frames are a special type of list! With some requirements:
  - Each component is a vector
  - Each component has the same length

Indexing data frames

[ ] can work for indexing data frames, just like matrices:

```
example_df <- data.frame(x = c(1, 2, 3),
                         y = c('a', 'b', 'c'))
example_df
```

```
  x y
1 1 a
2 2 b
3 3 c
```

row column

```
example_df[2, 1]
```

```
[1] 2
```

Indexing data frames

Like lists, `[[ ]]` and `$` can also be used:

```
example_df <- data.frame(x = c(1, 2, 3),
                         y = c('a', 'b', 'c'))
example_df
```

```
  x y
1 1 a
2 2 b
3 3 c
```

get column named x

```
example_df$x
```

```
[1] 1 2 3
```

```
example_df[["x"]]
```

```
[1] 1 2 3
```

get column named x

What do you do with a data frame?

- Input for modeling
- EDA (summary statistics, visualization, etc.)
- Data manipulation & cleaning ← often takes the most work

Data manipulation

```r
glimpse(starwars)
```

```
Rows: 87
Columns: 14
$ name       <chr> "Luke Skywalker", "C-3PO", "R2-D2", "Darth Vader",
"Leia Or…
$ height     <int> 172, 167, 96, 202, 150, 178, 165, 97, 183, 182, 188,
180, 2…
$ mass       <dbl> 77.0, 75.0, 32.0, 136.0, 49.0, 120.0, 75.0, 32.0,
84.0, 77.…
$ hair_color <chr> "blond", NA, NA, "none", "brown", "brown, grey",
"brown", N…
$ skin_color <chr> "fair", "gold", "white, blue", "white", "light",
"light", "…
$ eye_color  <chr> "blue", "yellow", "red", "yellow", "brown", "blue",
"blue",…
```

What manipulation might I want to do with the `starwars` data?

- handle NAs (missing values)
- look at a subset of rows
- create new variables, transform existing variables
- calculate some summary statistics

`dplyr`: Tools for data wrangling

- part of the tidyverse
- provides a “grammar of data manipulation”: useful verbs (functions) for manipulating data
- we will cover the key `dplyr` functions

Verbs for data wrangling

- `select()`: take a subset of the columns (i.e., features, variables)
- `filter()`: take a subset of the rows (i.e., observations)
- `mutate()`: add or modify existing columns
- `arrange()`: sort the rows
- ~~`summarize()`: aggregate the data across rows~~ (e.g., ~~group it according to some criteria~~)

summarize(): calculates summary statistics

group_by(): aggregate data

Creating a subset of the rows

Question: Suppose I only want the droids in the `starwars` data. How would I choose only those rows?

```
subset( starwars,   starwars $species == "Droid")

starswars[ starwars $ species == "Droid" , ]
```

Creating a subset of the rows

Question: Suppose I only want the droids in the `starwars` data. How would I choose only those rows?

```
filter(starwars, species == "Droid")
```

first argument: dataset

specify criterion for the rows we want to keep

```
# A tibble: 6 × 14
  name   height  mass hair_color skin_color  eye_color birth_year sex
gender
  <chr>   <int> <dbl> <chr>      <chr>       <chr>           <dbl> <chr>
<chr>
1 C-3PO     167    75 <NA>       gold        yellow            112 none
masculi…
2 R2-D2      96    32 <NA>       white, blue red                33 none
masculi…
3 R5-D4      97    32 <NA>       white, red  red               NA none
masculi…
4 IG-88     200   140 none       metal       red                15 none
masculi…
5 R4-P17     96    NA none       silver, red red, blue         NA none
```

Creating a subset of the rows

```
starwars |>
  filter(species == "Droid")
```

equiv. to filter(starwars, species == "Droid")

```
# A tibble: 2 × 14
  name  height  mass hair_color skin_color  eye_color birth_year sex
gender
  <chr>  <int> <dbl> <chr>      <chr>       <chr>           <dbl> <chr>
<chr>
1 C-3PO    167    75 <NA>       gold        yellow            112 none
masculine
2 R2-D2     96    32 <NA>       white, blue red                33 none
masculine
# ℹ 5 more variables: homeworld <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>
```

- |> is called the *pipe*. It means "take <this>, THEN do <that>"
- `filter` keeps only the rows which satisfy a specific condition

older syntax for pipe: %>%

Calculating summary statistics

Question: What is the average height for droids in the dataset?

Calculating summary statistics

Question: What is the average height for droids in the dataset?

```
starwars |>
  filter(species == "Droid") |>
  summarize(mean_height = mean(height))
```

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

choose an informative name

Calculating the mean

- pipes (`|>`) can be chained together
- `summarize` calculates summary statistics
- Why am I getting `NA`?

Some droids are missing height!

Handling missing values

```
# A tibble: 6 × 14
  name   height  mass hair_color skin_color  eye_color birth_year sex
gender
  <chr>   <int> <dbl> <chr>      <chr>       <chr>           <dbl> <chr>
<chr>
1 C-3PO     167    75 <NA>       gold        yellow            112 none
masculi…
2 R2-D2      96    32 <NA>       white, blue red                33 none
masculi…
3 R5-D4      97    32 <NA>       white, red  red               NA none
masculi…
4 IG-88     200   140 none       metal       red                15 none
masculi…
5 R4-P17     96    NA none       silver, red red, blue         NA none
```

```
starwars |>
  filter(species == "Droid") |>
  summarize(mean_height = mean(height, na.rm=T))
```

```
# A tibble: 1 × 1
  mean_height
        <dbl>
1        131.
```

"ignore missing values (NAs) when calculating the mean"

Calculating summary statistics

Question: What if I want the average height for *humans*?

```
starwars |>
  filter(species == "Droid") |>
  summarize(mean_height = mean(height, na.rm=T))
```

Calculating summary statistics

Question: What if I want the average height for *humans*?

```r
starwars |>
  filter(species == "Human") |>
  summarize(mean_height = mean(height, na.rm=T))
```

```
# A tibble: 1 × 1
  mean_height
        <dbl>
1        177.
```

Calculating summary statistics

Question: What is the average body mass for *each* species?

Calculating summary statistics

Question: What is the average ~~body mass~~ height for *each* species?

```
starwars |>
  group_by(species) |>
  summarize(mean_height = mean(height, na.rm=T))
```

group by species

```
# A tibble: 38 × 2
   species   mean_height
   <chr>           <dbl>
 1 Aleena           79
 2 Besalisk        198
 3 Cerean          198
 4 Chagrian        196
 5 Clawdite        168
 6 Droid           131.
 7 Dug             112
 8 Ewok             88
 9 Geonosian       183
10 Gungan          209.
# ℹ 28 more rows
```

Creating new variables

Question: What is the distribution of the ratio of body mass to height?

Creating new variables

Question: What is the distribution of the ratio of body mass to height?

```
starwars |>
  mutate(body_ratio = mass/height)
```

create a new variable (or modify an existing variable)

define new variable (in terms of existing variables)

Creating new variables

```r
starwars |>
  mutate(body_ratio = mass/height) |>
  group_by(species) |>
  summarize(mean_ratio = mean(body_ratio, na.rm=T),
            sd_ratio = sd(body_ratio, na.rm=T))
```

```
# A tibble: 38 × 3
   species   mean_ratio sd_ratio
   <chr>          <dbl>    <dbl>
 1 Aleena         0.190  NA
 2 Besalisk       0.515  NA
 3 Cerean         0.414  NA
 4 Chagrian     NaN      NA
 5 Clawdite       0.327  NA
 6 Droid          0.453   0.174
 7 Dug            0.357  NA
 8 Ewok           0.227  NA
 9 Geonosian      0.437  NA
10 Gungan         0.351   0.0207
# ℹ 28 more rows
```

Creating new variables

```
starwars |>
  mutate(body_ratio = mass/height) |>
  group_by(species) |>
  summarize(mean_ratio = mean(body_ratio, na.rm=T),
            sd_ratio = sd(body_ratio, na.rm=T),
            N = n())
```

need at least 2 obs. to calculate sd

counts # of rows (no arguments needed)

```
# A tibble: 38 × 4
   species   mean_ratio sd_ratio     N
   <chr>          <dbl>    <dbl> <int>
 1 Aleena         0.190  NA          1
 2 Besalisk       0.515  NA          1
 3 Cerean         0.414  NA          1
 4 Chagrian     NaN      NA          1
 5 Clawdite       0.327  NA          1
 6 Droid          0.453   0.174      6
 7 Dug            0.357  NA          1
 8 Ewok           0.227  NA          1
 9 Geonosian      0.437  NA          1
10 Gungan         0.351   0.0207     3
# ℹ 28 more rows
```

Summary so far

- `filter`: choose certain rows
- `summarize`: calculate summary statistics
- `group_by`: group rows together
- `mutate`: create new columns

Class activity

https://sta279-s24.github.io/class_activities/ca_lecture_9.html