

More data wrangling with **dplyr**

Stat 133 with Gaston Sanchez

Creative Commons Attribution Share-Alike 4.0 International CC BY-SA

Data Wrangling Pipelines

Toy Data

| name | gender | height |
|--------|--------|--------|
| Anakin | male | 1.88 |
| Padme | female | 1.65 |
| Luke | male | 1.72 |
| Leia | female | 1.50 |

```
dat <- data.frame(  
  name = c('Anakin', 'Padme', 'Luke', 'Leia'),  
  gender = c('male', 'female', 'male', 'female'),  
  height = c(1.88, 1.65, 1.72, 1.50)  
)
```

Function calls in dplyr

dplyr functional calls

An “ugly” side of dplyr is that if you want to do many operations at once, it does not lead to particularly elegant code.

You either have to do computations, step-by-step, with separate commands...

Or you have to wrap several function calls inside each other (making your code hard to read)

dat

| name | gender | height |
|--------|--------|--------|
| Anakin | male | 1.88 |
| Padme | female | 1.65 |
| Luke | male | 1.72 |
| Leia | female | 1.50 |

**output**

| gender | avg | sd |
|--------|------|-------|
| male | 1.8 | 0.113 |
| female | 1.58 | 0.106 |

Example: For each gender category, get the average and standard deviation of height, arranging output by average in descending order.

dat

| name | gender | height |
|--------|--------|--------|
| Anakin | male | 1.88 |
| Padme | female | 1.65 |
| Luke | male | 1.72 |
| Leia | female | 1.50 |



| gender | avg | sd |
|--------|------|-------|
| male | 1.8 | 0.113 |
| female | 1.58 | 0.106 |

Step-by-step computations

```
dat1 = group_by(dat, gender)
dat2 = summarise(dat1,
  avg = mean(height), sd = sd(height))
dat3 = arrange(dat2, desc(avg))
dat3
```

dat

| name | gender | height |
|--------|--------|--------|
| Anakin | male | 1.88 |
| Padme | female | 1.65 |
| Luke | male | 1.72 |
| Leia | female | 1.50 |



| gender | avg | sd |
|--------|------|-------|
| male | 1.8 | 0.113 |
| female | 1.58 | 0.106 |

*Function calls
inside each other*

```
arrange (  
  summarise (group_by (dat, gender) ,  
    avg = mean (height) ,  
    sd = sd (height) ) ,  
  desc (avg) )
```


Pipe Operators

|> or %>%

dat

| name | gender | height |
|--------|--------|--------|
| Anakin | male | 1.88 |
| Padme | female | 1.65 |
| Luke | male | 1.72 |
| Leia | female | 1.50 |



dat3

| gender | avg | sd |
|--------|------|-------|
| male | 1.8 | 0.113 |
| female | 1.58 | 0.106 |

*Pipeline of commands: easier
to write and understand*

```
dat3 = dat |>  
  group_by(gender) |>  
  summarise(avg = mean(height) ,  
            sd = sd(height)) |>  
  arrange(desc(avg))
```

Pipe operators



Base R



magrittr

Pipe operators

A pipe operator lets you write:

$$f(x, y)$$

as:

$$x \mid > f(y)$$

or equivalently as:

$$x \%>\% f(y)$$

Example

```
x = c(2, 4, 6, NA)
```

```
# without the pipe
```

```
mean(x, na.rm = TRUE)
```

```
# with the pipe
```

```
x |> mean(na.rm = TRUE)
```

```
set.seed(12345)
n = 10
x1 = runif(n, min = -3, max = 3)
x2 = round(x1, 2)
x3 = abs(x2)
x4 = sum(x3)
x4
```

no pipe

```
set.seed(12345)
10 |>
  runif(min = -3, max = 3) |>
  round(2) |>
  abs() |>
  sum()
```

pipeline

Pipe Operators: %>% and |>

Pipe operators, denoted as %>% and also as |>, allow you to write function calls in a more human-readable way.

These operators are heavily used among the ecosystem of "tidyverse" packages, and they are becoming more common in traditional R code.

Technically speaking, %>% is known as the “**magrittr**” pipe operator (from its homonym package, introduced in 2014).

In turn, |> is the **base R pipe operator** (introduced in May 2021 with R version 4.1.0.)

dat

| name | gender | height |
|--------|--------|--------|
| Anakin | male | 1.88 |
| Padme | female | 1.65 |
| Luke | male | 1.72 |
| Leia | female | 1.50 |



| name | gender | height |
|-------|--------|--------|
| Padme | female | 1.65 |
| Leia | female | 1.50 |

```
filter(dat, gender == "female")
```

is equivalent to

```
dat |> filter(gender == "female")
```


dat

| name | gender | height |
|--------|--------|--------|
| Anakin | male | 1.88 |
| Padme | female | 1.65 |
| Luke | male | 1.72 |
| Leia | female | 1.50 |



| name | gender |
|--------|--------|
| Anakin | male |
| Padme | female |
| Luke | male |
| Leia | female |

```
select (dat, name:gender)
```

is equivalent to

```
dat |> select (name:gender)
```

dat

| name | gender | height |
|--------|--------|--------|
| Anakin | male | 1.88 |
| Padme | female | 1.65 |
| Luke | male | 1.72 |
| Leia | female | 1.50 |



| name | gender | height |
|--------|--------|--------|
| Anakin | male | 1.88 |
| Luke | male | 1.72 |
| Padme | female | 1.65 |
| Leia | female | 1.50 |

```
arrange(dat, desc(height))
```

is equivalent to

```
dat |> arrange(desc(height))
```

dat

| name | gender | height |
|--------|--------|--------|
| Anakin | male | 1.88 |
| Padme | female | 1.65 |
| Luke | male | 1.72 |
| Leia | female | 1.50 |



?

Example: number of non-male individuals with heights greater than 1.40 meters

dat

| name | gender | height |
|--------|--------|--------|
| Anakin | male | 1.88 |
| Padme | female | 1.65 |
| Luke | male | 1.72 |
| Leia | female | 1.50 |

→ 2

dat |>

filter(gender != "male") |>

filter(height > 1.4) |>

summarise(n())

dat

| name | gender | height |
|--------|--------|--------|
| Anakin | male | 1.88 |
| Padme | female | 1.65 |
| Luke | male | 1.72 |
| Leia | female | 1.50 |



Example: convert height into inches, and display name and height (inches) of male individuals, arranging content by height in descending order.

dat

| name | gender | height |
|--------|--------|--------|
| Anakin | male | 1.88 |
| Padme | female | 1.65 |
| Luke | male | 1.72 |
| Leia | female | 1.50 |



| name | ht_in |
|--------|---------|
| Anakin | 74.0156 |
| Luke | 67.7164 |

dat |>

mutate(ht_in = height * 39.37) |>

filter(gender == "male") |>

select(name, ht_in) |>

arrange(desc(ht_in))

dat

| name | gender | height |
|--------|--------|--------|
| Anakin | male | 1.88 |
| Padme | female | 1.65 |
| Luke | male | 1.72 |
| Leia | female | 1.50 |



| name | ht_in |
|--------|---------|
| Anakin | 74.0156 |
| Luke | 67.7164 |

*Equivalent
pipeline*

dat |>

filter(gender == "male") |>

mutate(ht_in = height * 39.37) |>

select(name, ht_in) |>

arrange(desc(ht_in))

Merging Tables with joins()

Data Tables

tbl1

| id | year | coffee |
|------|------|--------|
| Luke | 1 | no |
| Leia | 3 | yes |
| Han | 4 | yes |

tbl2

| id | gpa | lunch |
|---------|-----|----------|
| Padme | 3.9 | pizza |
| Leia | 4.0 | tacos |
| Luke | 3.7 | burrito |
| Obi-Wan | 3.8 | pad thai |

Data Tables

```
tbl1 <- data.frame(  
  id = c('Luke', 'Leia', 'Han'),  
  year = c(1, 3, 4),  
  coffee = c('no', 'yes', 'yes')  
)
```

```
tbl2 <- data.frame(  
  id = c('Padme', 'Leia', 'Luke', 'Obi-Wan'),  
  gpa = c(3.9, 4.0, 3.7, 3.8),  
  lunch = c('pizza', 'tacos', 'burrito', 'pad thai')  
)
```

tbl1

| id | year | coffee |
|------|------|--------|
| Luke | 1 | no |
| Leia | 3 | yes |
| Han | 4 | yes |

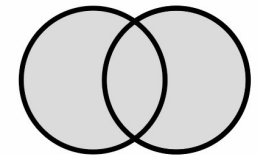
tbl2

| id | gpa | lunch |
|---------|-----|----------|
| Padme | 3.9 | pizza |
| Leia | 4.0 | tacos |
| Luke | 3.7 | burrito |
| Obi-Wan | 3.8 | pad thai |

keeps all observations in tbl1 and tbl2

full_join(tbl1, tbl2, by = "id")

| id | year | coffee | gpa | lunch |
|---------|------|--------|-----|----------|
| Luke | 1 | no | 3.7 | burrito |
| Leia | 3 | yes | 4.0 | tacos |
| Han | 4 | yes | NA | NA |
| Padme | NA | NA | 3.9 | pizza |
| Obi-Wan | NA | NA | 3.8 | pad thai |



full_join(x, y)

tbl1

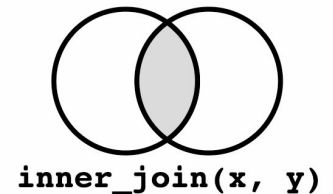
| id | year | coffee |
|------|------|--------|
| Luke | 1 | no |
| Leia | 3 | yes |
| Han | 4 | yes |

tbl2

| id | gpa | lunch |
|---------|-----|----------|
| Padme | 3.9 | pizza |
| Leia | 4.0 | tacos |
| Luke | 3.7 | burrito |
| Obi-Wan | 3.8 | pad thai |

keeps obs in tbl1 that have matching key in tbl2
inner_join(tbl1, tbl2, by = "id")

| id | year | coffee | gpa | lunch |
|------|------|--------|-----|---------|
| Luke | 1 | no | 3.7 | burrito |
| Leia | 3 | yes | 4.0 | tacos |



tbl1

| id | year | coffee |
|------|------|--------|
| Luke | 1 | no |
| Leia | 3 | yes |
| Han | 4 | yes |

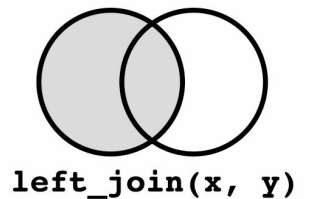
tbl2

| id | gpa | lunch |
|---------|-----|----------|
| Padme | 3.9 | pizza |
| Leia | 4.0 | tacos |
| Luke | 3.7 | burrito |
| Obi-Wan | 3.8 | pad thai |

keeps all observations tbl1

```
left_join(tbl1, tbl2, by = "id")
```

| id | year | coffee | gpa | lunch |
|------|------|--------|-----|---------|
| Luke | 1 | no | 3.7 | burrito |
| Leia | 3 | yes | 4.0 | tacos |
| Han | 4 | yes | NA | NA |



tbl1

| id | year | coffee |
|------|------|--------|
| Luke | 1 | no |
| Leia | 3 | yes |
| Han | 4 | yes |

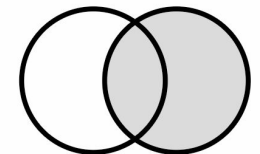
tbl2

| id | gpa | lunch |
|---------|-----|----------|
| Padme | 3.9 | pizza |
| Leia | 4.0 | tacos |
| Luke | 3.7 | burrito |
| Obi-Wan | 3.8 | pad thai |

keeps all observations in tbl2

right_join(tbl1, tbl2, by = "id")

| id | year | coffee | gpa | lunch |
|---------|------|--------|-----|----------|
| Luke | 1 | no | 3.7 | burrito |
| Leia | 3 | yes | 4.0 | tacos |
| Padme | NA | NA | 3.9 | pizza |
| Obi-Wan | NA | NA | 3.8 | pad thai |



right_join(x, y)

tbl1

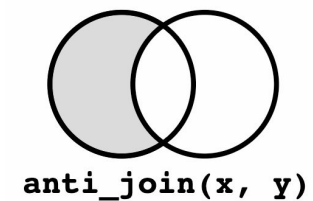
| id | year | coffee |
|------|------|--------|
| Luke | 1 | no |
| Leia | 3 | yes |
| Han | 4 | yes |

tbl2

| id | gpa | lunch |
|---------|-----|----------|
| Padme | 3.9 | pizza |
| Leia | 4.0 | tacos |
| Luke | 3.7 | burrito |
| Obi-Wan | 3.8 | pad thai |

return rows from tbl1 without a match in tbl2
anti_join(tbl1, tbl2, by = "id")

| id | year | coffee |
|-----|------|--------|
| Han | 4 | yes |



tbl1

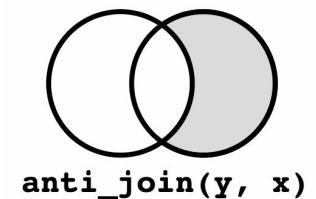
| id | year | coffee |
|------|------|--------|
| Luke | 1 | no |
| Leia | 3 | yes |
| Han | 4 | yes |

tbl2

| id | gpa | lunch |
|---------|-----|----------|
| Padme | 3.9 | pizza |
| Leia | 4.0 | tacos |
| Luke | 3.7 | burrito |
| Obi-Wan | 3.8 | pad thai |

return rows from tbl2 without a match in tbl1
anti_join(tbl2, tbl1, by = "id")

| id | gpa | lunch |
|---------|-----|----------|
| Padme | 3.9 | pizza |
| Obi-Wan | 3.8 | pad thai |



tbl1

| id | year | coffee |
|------|------|--------|
| Luke | 1 | no |
| Leia | 3 | yes |
| Han | 4 | yes |

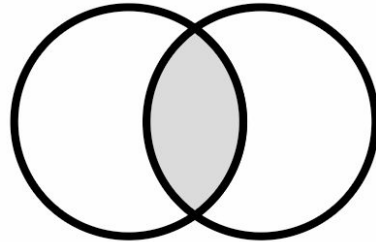
tbl2

| id | gpa | lunch |
|---------|-----|----------|
| Padme | 3.9 | pizza |
| Leia | 4.0 | tacos |
| Luke | 3.7 | burrito |
| Obi-Wan | 3.8 | pad thai |

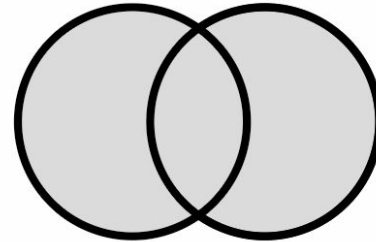
return rows from tbl1 with a match in tbl2
semi_join(tbl1, tbl2, by = "id")

| id | year | coffee |
|------|------|--------|
| Luke | 1 | no |
| Leia | 3 | yes |

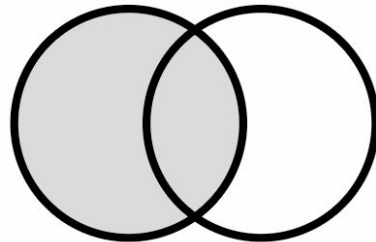
Joins



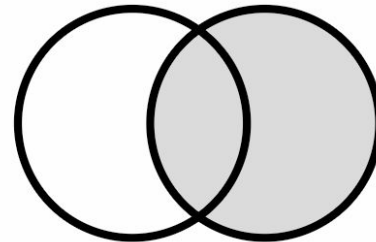
inner_join(x, y)



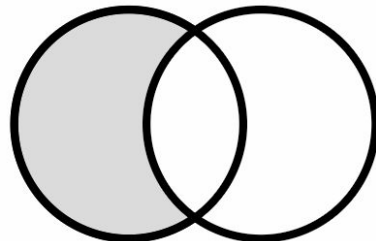
full_join(x, y)



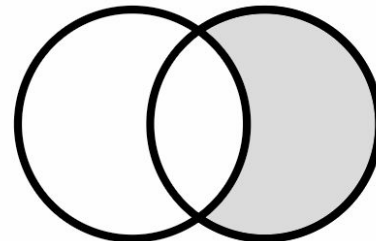
left_join(x, y)



right_join(x, y)



anti_join(x, y)



anti_join(y, x)

What if tables had keys with
different names?

Tables with different key names

tbl1

| id1 | year | coffee |
|------|------|--------|
| Luke | 1 | no |
| Leia | 3 | yes |
| Han | 4 | yes |

tbl2

| id2 | gpa | lunch |
|---------|-----|----------|
| Padme | 3.9 | pizza |
| Leia | 4.0 | tacos |
| Luke | 3.7 | burrito |
| Obi-Wan | 3.8 | pad thai |

Data Tables

```
tbl1 <- data.frame(  
  id1 = c('Luke', 'Leia', 'Han'),  
  year = c(1, 3, 4),  
  coffee = c('no', 'yes', 'yes')  
)
```

```
tbl2 <- data.frame(  
  id2 = c('Padme', 'Leia', 'Luke', 'Obi-Wan'),  
  gpa = c(3.9, 4.0, 3.7, 3.8),  
  lunch = c('pizza', 'tacos', 'burrito', 'pad thai')  
)
```

tbl1

| id1 | year | coffee |
|------|------|--------|
| Luke | 1 | no |
| Leia | 3 | yes |
| Han | 4 | yes |

tbl2

| id2 | gpa | lunch |
|---------|-----|----------|
| Padme | 3.9 | pizza |
| Leia | 4.0 | tacos |
| Luke | 3.7 | burrito |
| Obi-Wan | 3.8 | pad thai |

keeps all observations in tbl1 and tbl2

`full_join(tbl1, tbl2, join_by("id1" == "id2"))`

| id1 | year | coffee | gpa | lunch |
|---------|------|--------|-----|----------|
| Luke | 1 | no | 3.7 | burrito |
| Leia | 3 | yes | 4.0 | tacos |
| Han | 4 | yes | NA | NA |
| Padme | NA | NA | 3.9 | pizza |
| Obi-Wan | NA | NA | 3.8 | pad thai |

tbl1

| id1 | year | coffee |
|------|------|--------|
| Luke | 1 | no |
| Leia | 3 | yes |
| Han | 4 | yes |

tbl2

| id2 | gpa | lunch |
|---------|-----|----------|
| Padme | 3.9 | pizza |
| Leia | 4.0 | tacos |
| Luke | 3.7 | burrito |
| Obi-Wan | 3.8 | pad thai |

equivalent command

```
full_join(tbl1, tbl2, by = c("id1" = "id2"))
```

| id1 | year | coffee | gpa | lunch |
|---------|------|--------|-----|----------|
| Luke | 1 | no | 3.7 | burrito |
| Leia | 3 | yes | 4.0 | tacos |
| Han | 4 | yes | NA | NA |
| Padme | NA | NA | 3.9 | pizza |
| Obi-Wan | NA | NA | 3.8 | pad thai |