More data wrangling with **dplyr**

Stat 133 with Gaston Sanchez

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

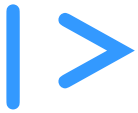
dat3

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 |> 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 `|>`

Oipe 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))
```