

Why do we need a grammar?

Supplement to: Data science lecture 3

Who are the end users of our (research) code?

You in the **present**

You in the **future**

Your **peer reviewers**, i.e., open source developers + work colleagues

Anyone in the world interested in your research

Takeaway: writing code should be done **empathetically**, especially to **yourself**

What are the useful characteristics of great code?

A **pleasure** to read

Easy to **explain** and **reason with**

Easy to **extend** and **modify** (refactor)

Consistent **formatting style**

Takeaway: code should ideally match the flow of structured natural language!

What are the key characteristics of natural language prose?

First we just *happen* to use English, so let's restrict to English prose

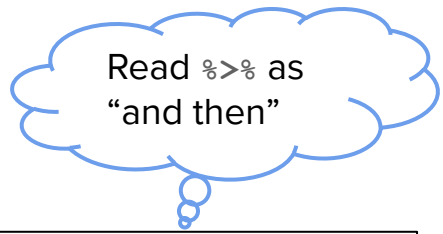
Written and **read left-to-right**

Based on a (largely*) **consistent grammar**

Room for **extensibility** and **adaptation**, e.g., emojis 😊

Takeaway: need code style that has a nice “grammar” and easy to understand

{dplyr} uses tidy grammar for data wrangling



```
tibble
data.frame
```

Objects = "nouns"

+

```
select()
filter()
slice()
group_by()
summary()
mutate()
...
<prefix_action>()
```

Functions = "verbs"

%>%

```
year_batting_summary <-
Batting %>%
  filter(lgID %in% c("AL")) %>%
  group_by(yearID) %>%
  summarize_at(vars(H, HR),
               sum,
               na.rm = TRUE) %>%
  mutate(batting_avg = H / AB)
```

Code = "composition"

The %>% is really cool for many reasons

Allows us to **write** function composition in a **left-to right** manner

```
Batting %>%  
select(yearID, SO) %>%  
  group_by(yearID)
```

vs.

```
group_by(select(Batting,  
  yearID, SO),  
  yearID)
```

Write comp
Left-to-right.

Allows us to interactively **highlight and run** incremental compositions

```
Batting %>%  
select(yearID, SO) %>%  
  group_by(yearID)
```

vs.

```
group_by(select(Batting,  
  yearID, SO),  
  yearID)
```

Interactively highlight
& run inc. comp.

Allows us to interactively **flip** incremental compositions

```
Batting %>%  
group_by(yearID) %>%  
  select(yearID, SO)
```

vs.

```
select(group_by(Batting,  
  yearID), yearID, SO)
```

Interactively **flip comp.**