Data transformation

The pipe `%>%` operator

Structure of the `select` function

```
select(name_of_dataset, column1, column2, ...)
```

Replace `name_of_dataset` with the variable storing your dataset and `column1`, `column2`, and so on with actual names of the columns you want to keep.

Using the pipe %>% operator

- We will need a convenient way to write out a sequence of data transformations

Using the pipe %>% operator

- We will need a convenient way to write out a sequence of data transformations
- The symbol `%>%` is called the pipe operator, and it is available for you to use after running `library(tidyverse)`

Using the pipe %>% operator

- We will need a convenient way to write out a sequence of data transformations
- The symbol `%>%` is called the pipe operator, and it is available for you to use after running `library(tidyverse)`

```
select(presidential, name, party)
```

Using the pipe %>% operator

- We will need a convenient way to write out a sequence of data transformations
- The symbol `%>%` is called the pipe operator, and it is available for you to use after running `library(tidyverse)`

```
presidential %>%
  select(name, party)
```

Chaining transformations with %>%

Apply the sequence of functions `transform1`, `transform2`, `transform3`, and `transform4` to the data frame stored in a variable named `data`.

Chaining transformations with %>%

Apply the sequence of functions `transform1`, `transform2`, `transform3`, and `transform4` to the data frame stored in a variable named `data`.

```
data %>%
  transform1() %>%
  transform2(input1, input2) %>%
  transform3(input3) %>%
  transform4()
```

Chaining transformations with %>%

Apply the sequence of functions `transform1`, `transform2`, `transform3`, and `transform4` to the data frame stored in a variable named `data`.

```
data %>%
  transform1() %>%
  transform2(input1, input2) %>%
  transform3(input3) %>%
  transform4()
```

Using `%>%` shows the order of transformations in a clear and readable format.

Chaining transformations with %>%

Apply the sequence of functions `transform1`, `transform2`, `transform3`, and `transform4` to the data frame stored in a variable named `data`.

```
data %>%
  transform1() %>%
  transform2(input1, input2) %>%
  transform3(input3) %>%
  transform4()
```

Using `%>%` shows the order of transformations in a clear and readable format.

If we didn't use the pipe operator, then our code would look this way instead:

```
transform4(transform3(transform2(transform1(data), input1, input2), input3))
```

Credits

License

Acknowledgments

Ideas and examples for the dplyr demos adapted from *Modern Data Science with R* by Benjamin Baumer, Daniel Kaplan, and Nicholas Horton, chapter 4.