Introduction to R

10 Apr 2024
Xian Ji
Introduction to R

10 Apr 2024
Xian Ji
What is tidyverse

Most powerful collection of R packages for preparing, wrangling and visualizing data.

Install the complete tidyverse with:
install.packages("tidyverse")

Use pipes to clean up your R code

Pipes help your code flow better, making it cleaner and more efficient.

R Logical Operators

With logical operators, we want to return values inside the vector based on logical conditions. Following is a detailed list of logical operators of data types in R programming.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>==</td>
<td>Equal to</td>
</tr>
<tr>
<td>!=</td>
<td>Not equal to</td>
</tr>
<tr>
<td>!x</td>
<td>Not x</td>
</tr>
<tr>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>x &amp; y</td>
<td>x AND y</td>
</tr>
</tbody>
</table>
What is tidyverse

Most powerful collection of R packages for preparing, wrangling and visualizing data.

Install the complete tidyverse with:
install.packages("tidyverse")
Use pipes to clean up your R code

Pipes help your code flow better, making it cleaner and more efficient.

Using pipes in R

```
sum(c(1:10))
c(1:10) %>% sum()
c(1:10) %>% sum()
```

Ctrl + Shift + M
R Logical Operators

With logical operators, we want to return values inside the vector based on logical conditions. Following is a detailed list of logical operators of data types in R programming.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>==</td>
<td>Equal to</td>
</tr>
<tr>
<td>!=</td>
<td>Not equal to</td>
</tr>
<tr>
<td>!x</td>
<td>Not x</td>
</tr>
<tr>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>x &amp; y</td>
<td>x AND y</td>
</tr>
</tbody>
</table>
What is tidyverse

Most powerful collection of R packages for preparing, wrangling and visualizing data.

Install the complete tidyverse with:
install.packages("tidyverse")

Use pipes to clean up your R code
Pipes help your code flow better, making it cleaner and more efficient.

R Logical Operators
With logical operators, we want to return values inside the vector based on logical conditions. Following is a detailed list of logical operators of data types in R programming.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>==</td>
<td>Equal to</td>
</tr>
<tr>
<td>!=</td>
<td>Not equal to</td>
</tr>
<tr>
<td>!x</td>
<td>Not x</td>
</tr>
<tr>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>x &amp; y</td>
<td>x AND y</td>
</tr>
</tbody>
</table>
Data Manipulation: dplyr & tidyr

Preparation
Understanding the Environment
- The environment is a collection of objects.
- Objects are stored in variables.
- Functions create objects, or can modify existing ones.

Creating the Workspace
- A workspace is a collection of objects that can be accessed from the environment.
- Functions can create objects in the environment, or modify existing ones.

Setting the Working Directory
- The working directory is where R looks for files. It can be set using the `setwd()` function.

Saving workspace
- Use `saveRDS()` to save the workspace.
- Use `loadRDS()` to load the workspace.

Manipulate (Columns)
- Use `select()` and `arrange()` to manipulate columns.

Manipulate (Rows)
- Use `filter()` to filter rows.
- Use `mutate()` to add new columns.
- Use `summarize()` to summarize data.

Manipulate (Group)
- Use `group_by()` to group data.
- Use `summarize_at()` to summarize columns.

Reshape
- Use `pivot_wider()` to reshape data.
- Use `pivot_longer()` to reshape data.

Chinstrap! Gentoo! Adélie!
Preparation

Understanding the Environment
• The environment is a collection of symbols and their values.
• \texttt{ls()}: Lists all objects (variables, functions, etc.) in the current environment.

Clearing the Workspace
• When starting a new project or analysis, it's essential to have a clean workspace.
• \texttt{rm(objectname)}: Removes objects
• \texttt{rm(list = ls())}: Removes all objects from the environment, leaving an empty workspace.

Loading Libraries
• \texttt{library(tidyverse)} for data manipulation and visualization.
• equal to load tidyr, dplyr, ggplot2... at the same time

Setting the Working Directory
• The working directory is where R looks for files and saves outputs.
• \texttt{setwd("C:/path/to_your/working_directory")}: Sets the working directory to the specified folder.
CHINSTRAP!  GENTOO!  ADÉLIE!
Read the data into R

read.csv()
<table>
<thead>
<tr>
<th>id</th>
<th>species</th>
<th>island</th>
<th>bill_length_mm</th>
<th>bill_depth_mm</th>
<th>flipper_length_mm</th>
<th>body_mass_g</th>
<th>sex</th>
<th>year</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Adelie</td>
<td>Torgersen</td>
<td>39.1</td>
<td>18.7</td>
<td>181</td>
<td>3750</td>
<td>male</td>
<td>2007</td>
</tr>
<tr>
<td>2</td>
<td>Adelie</td>
<td>Torgersen</td>
<td>39.5</td>
<td>17.4</td>
<td>186</td>
<td>3800</td>
<td>female</td>
<td>2007</td>
</tr>
<tr>
<td>3</td>
<td>Adelie</td>
<td>Torgersen</td>
<td>40.3</td>
<td>18.0</td>
<td>195</td>
<td>3250</td>
<td>female</td>
<td>2007</td>
</tr>
<tr>
<td>4</td>
<td>Adelie</td>
<td>Torgersen</td>
<td>36.7</td>
<td>19.3</td>
<td>193</td>
<td>3450</td>
<td>female</td>
<td>2007</td>
</tr>
<tr>
<td>5</td>
<td>Adelie</td>
<td>Torgersen</td>
<td>39.3</td>
<td>20.6</td>
<td>190</td>
<td>3650</td>
<td>male</td>
<td>2007</td>
</tr>
<tr>
<td>6</td>
<td>Adelie</td>
<td>Torgersen</td>
<td>38.9</td>
<td>17.8</td>
<td>181</td>
<td>3625</td>
<td>female</td>
<td>2007</td>
</tr>
<tr>
<td>7</td>
<td>Adelie</td>
<td>Torgersen</td>
<td>39.2</td>
<td>19.6</td>
<td>195</td>
<td>4675</td>
<td>male</td>
<td>2007</td>
</tr>
<tr>
<td>8</td>
<td>Adelie</td>
<td>Torgersen</td>
<td>41.1</td>
<td>17.6</td>
<td>182</td>
<td>3200</td>
<td>female</td>
<td>2007</td>
</tr>
<tr>
<td>9</td>
<td>Adelie</td>
<td>Torgersen</td>
<td>38.6</td>
<td>21.2</td>
<td>191</td>
<td>3800</td>
<td>male</td>
<td>2007</td>
</tr>
<tr>
<td>10</td>
<td>Adelie</td>
<td>Torgersen</td>
<td>34.6</td>
<td>21.1</td>
<td>198</td>
<td>4400</td>
<td>male</td>
<td>2007</td>
</tr>
<tr>
<td>11</td>
<td>Adelie</td>
<td>Torgersen</td>
<td>36.6</td>
<td>17.8</td>
<td>185</td>
<td>3700</td>
<td>female</td>
<td>2007</td>
</tr>
<tr>
<td>12</td>
<td>Adelie</td>
<td>Torgersen</td>
<td>38.7</td>
<td>19.0</td>
<td>195</td>
<td>3450</td>
<td>female</td>
<td>2007</td>
</tr>
<tr>
<td>13</td>
<td>Adelie</td>
<td>Torgersen</td>
<td>42.5</td>
<td>20.7</td>
<td>197</td>
<td>4500</td>
<td>male</td>
<td>2007</td>
</tr>
<tr>
<td>14</td>
<td>Adelie</td>
<td>Torgersen</td>
<td>34.4</td>
<td>18.4</td>
<td>184</td>
<td>3325</td>
<td>female</td>
<td>2007</td>
</tr>
<tr>
<td>15</td>
<td>Adelie</td>
<td>Torgersen</td>
<td>46.0</td>
<td>21.5</td>
<td>194</td>
<td>4200</td>
<td>male</td>
<td>2007</td>
</tr>
<tr>
<td>16</td>
<td>Adelie</td>
<td>Biscoe</td>
<td>37.8</td>
<td>18.3</td>
<td>174</td>
<td>3400</td>
<td>female</td>
<td>2007</td>
</tr>
<tr>
<td>17</td>
<td>Adelie</td>
<td>Biscoe</td>
<td>37.7</td>
<td>18.7</td>
<td>180</td>
<td>3600</td>
<td>female</td>
<td>2007</td>
</tr>
<tr>
<td>18</td>
<td>Adelie</td>
<td>Biscoe</td>
<td>35.9</td>
<td>19.2</td>
<td>189</td>
<td>3800</td>
<td>female</td>
<td>2007</td>
</tr>
<tr>
<td>19</td>
<td>Adelie</td>
<td>Biscoe</td>
<td>38.2</td>
<td>18.1</td>
<td>185</td>
<td>3050</td>
<td>male</td>
<td>2007</td>
</tr>
<tr>
<td>Column</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>ID of penguins</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>species</td>
<td>Species of penguins (Adelie, Gentoo, Chinstrap)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>island</td>
<td>The island where penguins live in Palmer Archipelago, Antarctica (Biscoe, Dream or Torgersen)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bill_length_mm</td>
<td>A number denoting bill length (millimeters)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bill_depth_mm</td>
<td>A number denoting bill depth (millimeters)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>flipper_length_mm</td>
<td>An integer denoting flipper length (millimeters)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>body_mass_g</td>
<td>An integer denoting body mass (grams)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sex</td>
<td>Penguin sex (female, male)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>year</td>
<td>The year of the study</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Manipulate (Columns)

**select()**
changes whether or not a column is included.

**mutate()**
creates new columns that are functions of existing variables.

**transmute()**
creates a new data frame containing only the specified computations.

**relocate()**
changes the order of the columns.

**rename()**
changes the name of the columns.

---

**Select helper functions**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>add_row</td>
<td>Add a new row to the data frame</td>
</tr>
<tr>
<td>add_column</td>
<td>Add a new column to the data frame</td>
</tr>
<tr>
<td>add_rows</td>
<td>Add multiple rows to the data frame</td>
</tr>
<tr>
<td>add_column_if_empty</td>
<td>Add a new column if the column is empty</td>
</tr>
<tr>
<td>add_row_if_empty</td>
<td>Add a new row if the row is empty</td>
</tr>
<tr>
<td>add_columns_if_empty</td>
<td>Add new columns if one or more of the columns are empty</td>
</tr>
<tr>
<td>add_rows_if_empty</td>
<td>Add new rows if one or more of the rows are empty</td>
</tr>
<tr>
<td>add_column_if_missing</td>
<td>Add a new column if the column is missing</td>
</tr>
<tr>
<td>add_row_if_missing</td>
<td>Add a new row if the row is missing</td>
</tr>
<tr>
<td>add_columns_if_missing</td>
<td>Add new columns if one or more of the columns are missing</td>
</tr>
<tr>
<td>add_rows_if_missing</td>
<td>Add new rows if one or more of the rows are missing</td>
</tr>
<tr>
<td>add_column_if_all_missing</td>
<td>Add a new column if all the columns are missing</td>
</tr>
<tr>
<td>add_row_if_all_missing</td>
<td>Add a new row if all the rows are missing</td>
</tr>
<tr>
<td>add_columns_if_all_missing</td>
<td>Add new columns if one or more of the columns are all missing</td>
</tr>
<tr>
<td>add_rows_if_all_missing</td>
<td>Add new rows if one or more of the rows are all missing</td>
</tr>
<tr>
<td>add_column_if_any_missing</td>
<td>Add a new column if any of the columns are missing</td>
</tr>
<tr>
<td>add_row_if_any_missing</td>
<td>Add a new row if any of the rows are missing</td>
</tr>
<tr>
<td>add_columns_if_any_missing</td>
<td>Add new columns if one or more of the columns are any missing</td>
</tr>
<tr>
<td>add_rows_if_any_missing</td>
<td>Add new rows if one or more of the rows are any missing</td>
</tr>
</tbody>
</table>

**Example**

- add a new column called body_mass_kg
- check difference between mutate and transmute categorize
- classify the penguins based on their body_mass_kg
  - logical operator
  - using if else
  - if else(condition, true, false, missing = NULL)
  - using case when

**Exercise**

- select column 'sex', 'year' and column starting with 'body_'
- rename sex as gender

**Example**

- make sex as the first column
- put sex as before the column island
select() changes whether or not a column is included.
## Select helper functions

<table>
<thead>
<tr>
<th>Helper Function</th>
<th>Description</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>starts_with()</td>
<td>Select columns whose names start with a specific prefix.</td>
<td>select(df, starts_with(&quot;prefix&quot;))</td>
</tr>
<tr>
<td>ends_with()</td>
<td>Select columns whose names end with a specific suffix.</td>
<td>select(df, ends_with(&quot;suffix&quot;))</td>
</tr>
<tr>
<td>contains()</td>
<td>Select columns whose names contain a specific substring.</td>
<td>select(df, contains(&quot;substring&quot;))</td>
</tr>
<tr>
<td>matches()</td>
<td>Select columns whose names match a regular expression pattern.</td>
<td>select(df, matches(&quot;^pattern.*$&quot;))</td>
</tr>
<tr>
<td>num_range()</td>
<td>Select columns with numerical suffixes within a range.</td>
<td>select(df, num_range(&quot;prefix&quot;, 1:5))</td>
</tr>
<tr>
<td>one_of()</td>
<td>Select columns listed in a character vector.</td>
<td>select(df, one_of(c(&quot;col1&quot;, &quot;col2&quot;, &quot;col3&quot;)))</td>
</tr>
<tr>
<td>everything()</td>
<td>Select all columns.</td>
<td>select(df, everything())</td>
</tr>
<tr>
<td>where()</td>
<td>Select columns based on a logical condition.</td>
<td>select(df, where(is.numeric))</td>
</tr>
</tbody>
</table>

## Example

Selecting Specific Columns by Name:
- select the column "bill_length_mm"

Selecting Columns with Helper Functions:
- Select the column that begins with bill
Example

Selecting Specific Columns by Name:
- select the column "bill_length_mm"

Selecting Columns with Helper Functions:
- Select the column that begins with bill
- Select the column that ends in "_mm"
- Select the column that contains "length".
- Select a numeric column
- Select a column of the string type
- Select a column other than the string type

Select a variety of combinations:
- select column "species" and column starting with "bill_

Dropping Columns:
- Drop the column "bill_length_mm"

Renaming Columns:
- Rename column "sex" as gender

Exercise

select column 'sex', 'year' and column starting with 'body_'
**mutate()**
creates new columns that are functions of existing variables.

**transmute()**
creates a new data frame containing only the specified computations.

Example:
- add a new column called `body_mass_kg`
Example

- add a new column called `body_mass_kg`
  - check difference between mutate and transmute

**categorize**
- classify the penguins based on their `body_mass_g`
  - logical operator
  - using `if_else`
    ```
    if_else(condition, true, false, missing = NULL)
    ```
  - using `case_when`
    ```
    case_when(condition1 ~ output1,
              TRUE ~ else_output)
    ```
  - This forces `case_when` to output "else_output", if none of the previous conditions were TRUE

- Categorizing penguin stature
- using if_else
  if_else(condition, true, false, missing = NULL)
- using case_when
  case_when(condition1 ~ output1, 
             TRUE ~ else_output)

This forces case_when to output "else_output", if none of the previous conditions were TRUE

• Categorizing penguin stature

Exercise: According to the length of the mouth peak, it is divided into 4 levels: A, B, C, and D
  "A": bill_length_mm < 35
  "B": bill_length_mm >= 35 & bill_length_mm < 45
  "C": bill_length_mm >= 45 & bill_length_mm < 55
  "D": bill_length_mm >= 55
relocate() changes the order of the columns.

dplyr::relocate() move Columns around!

Default: move for FRONT or move for before or after a specified column.

beep beep beep
Example

- make sex as the first column
- put sex as before the column island
rename() changes the name of columns.

Example

- rename sex as gender
Manipulate (Columns)

**select()**
changes whether or not a column is included.

**mutate()**
creates new columns that are functions of existing variables.

**transmute()**
creates a new data frame containing only the specified computations.

**relocate()**
changes the order of the columns.

**rename()**
changes the name of columns.

---

**Select helper functions**

<table>
<thead>
<tr>
<th>Select helper</th>
<th>Description</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>select()</td>
<td></td>
<td><img src="select_helper.png" alt="Select helper" /></td>
</tr>
<tr>
<td>mutate()</td>
<td></td>
<td><img src="mutate_helper.png" alt="Mutate helper" /></td>
</tr>
<tr>
<td>transmute()</td>
<td></td>
<td><img src="transmute_helper.png" alt="Transmute helper" /></td>
</tr>
<tr>
<td>relocate()</td>
<td></td>
<td><img src="relocate_helper.png" alt="Relocate helper" /></td>
</tr>
<tr>
<td>rename()</td>
<td></td>
<td><img src="rename_helper.png" alt="Rename helper" /></td>
</tr>
</tbody>
</table>

**Example**

Selecting Specific Columns by Name:
- select the column "bill_length_mm"
- select the column that begins with "bill"
- select the column that ends in "_mm"
- select the column that contains "length"
- select a numeric column
- select a column other than the string type
- select a variety of combinations:
  - select column "species" and column starting with "bill_"
  - Dropping Columns:
    - Drop the column "bill_length_mm"
  - Renaming Columns:
    - Rename column "sex" as gender

**Exercise**

- select column 'sex', 'year' and column starting with 'body_'

**Example**

- add a new column called `body_mass_kg`
- check difference between `mutate` and `transmute`
- categorize
  - classify the penguins based on their `body_mass_kg`
    - logical operator
    - using `if_else`
    - `if_else(condition, true, false, missing = NULL)`
    - `case_when` conditional output,
      * `TRUE = else_output`

- Categorizing penguin stature

**Exercise**

- rename sex as gender
Manipulate (Rows)

**filter()**
- chooses rows based on column values.

**Example**
- Filter male penguins
- Filter Adelie penguins
- Filter Adélie and Gentoo penguins
- Filter Chinstrap penguins which are smaller than 3000 g body mass
- Find the penguin with longest bill length

**slice()**
- chooses rows based on location.

**Example**
- Select the third row from the penguins dataset
- Select specific rows (2, 3, and 6) from the penguins dataset
- Select the first 3 rows of the penguins dataset
- Find the penguins with shortest bill length

**arrange()**
- changes the order of the rows.

**Example**
- Arrange the penguins dataset in ascending order based on bill length
- Arrange the penguins dataset in ascending order based on bill length within each species
- Arrange the penguins dataset in descending order based on species, sex, and flipper length

**Exercise**
- Same exercise using slice function: the Chinstrap penguin with longest bill length
- Find the top 4 rows with the maximum bill length from the penguins dataset
- Exercise to say the meaning of the following code:
  `penguins %>%
  filter(specific == "Chinstrap",
  bill_length_mm == max(bill_length_mm))`
  - the Chinstrap penguin with longest bill length how???
filter() chooses rows based on column values.
- filter male penguins
- filter Adelie penguins
- filter Adelie and Gentoo penguins
- filter Chinstrap penguins which are smaller than 3000 g body mass
- find the penguin with longest bill length
Exercise

#Exercise: to say the meaning of the following code
penguins %>%
  filter(species == "Chinstrap",
         bill_length_mm == max(bill_length_mm))

??? the Chinstrap penguin with longest bill length
how???
slice() chooses rows based on location.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>slice()</td>
<td>lets you index rows by their (integer) locations.</td>
</tr>
<tr>
<td>slice_head(), slice_tail()</td>
<td>select the first or last rows.</td>
</tr>
<tr>
<td>slice_sample()</td>
<td>randomly selects rows.</td>
</tr>
<tr>
<td>slice_min(), slice_max()</td>
<td>select rows with the smallest or largest values of a variable.</td>
</tr>
</tbody>
</table>

- select the third row from the penguins dataset
- select specific rows (2, 5, and 6) from the penguins dataset
- select the first 5 rows of the penguins dataset
- find the penguins with shorts bill length
Exercise

• Same exercise using slice function: the Chinstrap penguin with longest bill length

• find the top 4 rows with the maximum bill length from the penguins dataset
arrange() changes the order of the rows.
`arrange()` “arranges,” or organizes, our data in ascending order, starting from the lowest value and running to the highest (or in the case of character data, in alphabetical order).

- Arrange the penguins dataset in ascending order based on bill length
- Arrange the penguins dataset in descending order based on bill length
- Arrange the penguins dataset in ascending order based on bill length within each species
Exercise

• Arrange the penguins dataset in ascending order based on flipper length within each sex

• Arrange the penguins dataset in ascending order based on species, sex, and flipper length
Manipulate (Rows)

**filter()**
chooses rows based on column values.

- Filter male penguins
- Filter Adélie penguins
- Filter Adélie and Gentoo penguins
- Filter Chinstrap penguins which are smaller than 3000 g body mass
- Find the penguin with longest bill length

**slice()**
chooses rows based on location.

- Select the third row from the penguins dataset
- Select specific rows (2, 5, and 6) from the penguins dataset
- Select the first 5 rows of the penguins dataset
- Find the penguin with shortest bill length

**arrange()**
changes the order of the rows.

- Arrange the penguins dataset in ascending order on bill length
- Arrange the penguins dataset in descending order on bill length
- Arrange the penguins dataset in ascending order based on bill length within each species

**Example**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>slice()</td>
<td>lets you index rows by their (integer) locations.</td>
</tr>
<tr>
<td>slice(0:5)</td>
<td>select the first 5 rows.</td>
</tr>
<tr>
<td>slice(2:4)</td>
<td>select rows 2 to 4.</td>
</tr>
</tbody>
</table>

**Exercise**

# Exercise: to say the meaning of the following code
```r
group_by(species) %>%   
  filter(bill_length_mm > max(bill_length_mm))
```

- Find the Chinstrap penguin with the longest bill length.
- Find the top 4 rows with the maximum bill length from the penguins dataset.
- Arrange the penguins dataset in ascending order based on flipper length within each sex.
- Arrange the penguins dataset in ascending order based on species, sex, and flipper length.
Manipulate (Group)

\texttt{group\_by()}

\texttt{summarize()}

Example

- Calculate the average body mass for the entire penguins dataset
- Count the number of penguins for each sex category
- Count the number of penguins for each sex category (using count)
- Count the number of penguins for each sex and species (using summarise)
- Count the number of penguins for each sex and species (using count)
- Calculate the average body mass for each sex category
- Calculate the average body mass for each sex and species
- Calculate the mean body mass for each sex and display it in the detailed level
- Calculate the mean body mass for each species and display it in the detailed level

Exercise:

- Count the number of penguins for each sex, species, and island, arranged by species in descending order (2 ways: using count or using summarise)
Reshape

pivot_wider(
    data = long_data,
    name_from = name_column,
    values_from = value_column
)

pivot_longer(
    data = wide_data,
    cols = c(column1, column2, column3),
    names_to = "name",
    values_to = "value"
)

Example

- Pivoting from long to wide format based on sex categories (use data penguins_gw)
- Pivoting from wide to long format (use data from last example)

- Reshape the penguins data frame with year as columns, island as rows, and the number of species per island as the values
- Pivot the data frame longer so each row represents a unique island by year combination (use the data from last example)
- Pivot specific columns longer, creating a "year" column with corresponding "n_species" values
Saving workspace

You can use RStudio to save your workspace (the data objects listed in the Environment pane).

1. Quit R Session
   - Save workspace image to ~/.RData?
     - Cancel
     - Don't Save
     - Save

2. Environment pane
   - Global Environment
   - Import Dataset
   - 140 MiB

3. Save all: `save.image(file='myEnvironment.RData')`
   - Save specific object: `save(objectname, file = 'myEnvironment.RData')`

Loading back the entire list environment of objects

`load('myEnvironment.RData')`