ICA 2.1 Pipes, selecting and sorting (solutions)

```
library(tidyverse)
polyps <- read_csv("data/polyps.csv")
```

Task 1 - Use the pipe operator

(1) Rewrite the wrapped calls below to use the pipe operator. Make sure that you get the same result.

```
round(mean(c(265, 357, 154, 282, 173)), 0)
```

```
[1] 246
```

```
paste(signif(mean(polyps$age), 2), "years avg. age")
```

```
[1] "24 years avg. age"
```

(2) Consider the values below. Using the pipe operator, write the code to calculate their standard deviation (σ) and then the standard error of the sample (s or SE) using the equation: $\mathrm{SE} = \frac{\sigma}{\sqrt{n}}$, rounded to two decimal places. $n =$ number of values in sample.

```
c(158.4, 176.8, 162.7, 169.6, 163.0, 165.3, 134.3, 190.7)
```

```
[1] 158.4 176.8 162.7 169.6 163.0 165.3 134.3 190.7
```

Answer 1 - Use the pipe operator

```
# 1
c(265, 357, 154, 282, 173) %>%
  mean() %>%
  round(0)

mean(polyps$age) %>%
  signif(2) %>%
  paste("years avg. age")
```

Task 2 - Select variables of interest

(1) Get the variable names (column names) of `polyps`. How many are there? (Hints: `names()` and `length()`)

(2) Suppose we only want to examine the baseline, treatment, and 3-month polyp counts for a regression. Use the previous question's result with `dplyr::select()` to return a data frame that has only these three columns.

```
# 🅿
# your code here
```

(3) Consider the first, third, and fifth columns. What would you predict their names to be? Now, write a `select()` statement that selects these columns using numerical indices, and check your prediction.

(4) A collaborator tells you that the 12-month numbers are incomplete and need to be revised from another spreadsheet. They ask you to return all of the columns *except* for the 12-month numbers.

Answer 2 - Select variables of interest

```
# 1
names(polyps)
names(polyps) %>% length()
# 2
polyps %>% select(baseline, treatment, number3m)
# 3
polyps %>% select(1, 3, 5)
# 4
polyps %>% select(-number12m)
```

Task 3 - Arrange rows for browsing

(1) You are curious about the age makeup of the group. Use `dplyr::arrange()` to return `polyps` sorted by age.

```
# P
# your code here
```

(2) To avoid bias, you also want to browse the data in reverse age order. Unfortunately, you can't remember how to do that with `arrange()`. Run `?arrange` to find the solution in the function documentation. (Hint: you want to modify the *argument* you're giving to `arrange()`; check the Arguments section.) Then, write the code below and check your answer.

(3) Now you choose to sort by treatment group. However, since there are only two groups, more sorting might be desirable. Write the code for this and add the 3-month polyp count, descending, as a second sort.

(4) Consider the previous results. Now try reversing the sort order—3-month count descending, and then treatment group—and see how the data read differently.

Answer 3 - Arrange rows for browsing

```
# 1
polyps %>% arrange(age)
# 2
polyps %>% arrange(desc(age))
# 3
polyps %>% arrange(treatment, desc(number3m))
# 4
polyps %>% arrange(desc(number3m), treatment)
```

Different sort orders call our attention to different details in the observations.