dplyr - join and merge

Michael Mbajwa

2021-11-15

Judgment

This short tutorial will allow me to further explore `dplyr` functionality based on lectures I have had in my Data Science Master's course.

```
judgments <- read_delim("https://biostat2.uni.lu/practicals/data/judgments.tsv")
```

Import the data from the website.

```
## Rows: 188 Columns: 158
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr   (5): start_date, end_date, condition, gender, logbook
## dbl (153): finished, subject, age, mood_pre, mood_post, STAI_pre_1_1, STAI_p...
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Identify the moral dilemma with the highest average score across all participants. Guideline: The result will be a tibble containing the dilemma in rows(!) and the average such that the dilemma with the highest average in the first row.

```
judgments %>%
  select(contains("moral_dil")) %>% # I first select only columns with dilemma.
  # The selected tibble is then converted to a long format
  pivot_longer(contains("moral_dil"),
               names_to = "dilemma",
               values_to = "dilemma_val") %>%
  group_by(dilemma) %>%
  # After grouping by the dilemma's, I calculate the average of each dilemma
  summarise(dilemma_avg = mean(dilemma_val, na.rm = TRUE)) %>%
  arrange(desc(dilemma_avg)) # The dilemma's are then arranged in highest to lowest using the dilemma_a
```

```
## # A tibble: 7 x 2
##   dilemma                dilemma_avg
##   <chr>                        <dbl>
## 1 moral_dilemma_kitten          7.90
## 2 moral_dilemma_dog             7.35
## 3 moral_dilemma_wallet          7.14
## 4 moral_dilemma_plane           7
## 5 moral_dilemma_resume          6.92
## 6 moral_dilemma_control         6.34
## 7 moral_dilemma_trolley         3.57
```

Genetic variants

I will clean the table of genetic *variants* such that all variants appear as a column labeled by their position. The format in the input is the reference allele, the position and the variant, commonly called alternative allele. In `T6G`, `T` is the reference allele, `6` is the position (along the gene) and `G` is the variant allele.

```
variants <- tribble(
  ~sampleid, ~var1, ~var2, ~var3,
  "S1", "A3T", "T5G", "T6G",
  "S2", "A3G", "T5G", NA,
  "S3", "A3T", "T6C", "G10C",
  "S4", "A3T", "T6C", "G10C"
)
```

Guideline: The table should look something like this.

sampleid	3	5	6
S1	T	G	G
S2	G	G	NA

```
# THIS GIVES EXACT ANSWER AS STATED IN THE GUIDELINE
variants %>%
  # To make the output exact as the question, I first filter out rows that are not S1 or S2. A more rob
  filter(sampleid == "S1"|sampleid == "S2")%>%
  # The tibble is converted to a long format to be easily worked with
  pivot_longer(contains("var"),
               names_to = "var",
               values_to = "allele") %>%
  # I create two new columns, var and pos. var contains the variant which is usually the last char. So
  mutate(var = str_sub(allele, -1),
         pos = parse_number(allele)) %>%
  # I select the three columns I want to work with
  select(sampleid, var, pos) %>%
  # I transform the tibble to a wide format this time
  pivot_wider(names_from = pos,
              values_from = var)%>%
  # I filter out the NA column created as a result of the NA values
  select(sampleid, 2,3,4)
```

```
## # A tibble: 2 x 4
##   sampleid `3`   `5`   `6`
##   <chr>    <chr> <chr> <chr>
## 1 S1       T     G     G
## 2 S2       G     G     <NA>
```

```
# This is the similar but more robust code earlier referred to.
variants %>%
  pivot_longer(contains("var"),
               names_to = "var",
               values_to = "allele") %>%
  mutate(var = str_sub(allele, -1),
         pos = parse_number(allele)) %>%
  select(sampleid, var, pos) %>%
  pivot_wider(names_from = pos,
              values_from = var) %>%
  select(1:4, 6)
```

```
## # A tibble: 4 x 5
##   sampleid `3`   `5`   `6`   `10`
##   <chr>    <chr> <chr> <chr> <chr>
## 1 S1       T     G     G     <NA>
## 2 S2       G     G     <NA>  <NA>
## 3 S3       T     <NA>  C     C
## 4 S4       T     <NA>  C     C
```

Select relevant variants Genetic variants are labeled according to their effect on stability of the gene product.

I select the subjects in table *variants* that carry variants labeled as *damaging*. The final output would be vector of sample ids.

```
variant_significance <- tribble(
  ~variant, ~significance,
  "A3T", "unknown",
  "A3G", "damaging",
  "T5G", "benign",
  "T6G", "damaging",
  "T6C", "benign",
  "G10C", "unknown"
)
variant_significance
```

```
## # A tibble: 6 x 2
##   variant significance
##   <chr>   <chr>
## 1 A3T     unknown
## 2 A3G     damaging
## 3 T5G     benign
## 4 T6G     damaging
## 5 T6C     benign
## 6 G10C    unknown
```

```
variants %>%
  # Data is pivoted to the long format.
  pivot_longer(contains("var"),
               names_to = "var",
               values_to = "allele") %>%
  # I then select the two columns that I will work with
  select(1,3) %>%
  # I create a new column that maps the gene code to its significance
  mutate(sig = case_when(
    str_detect(allele, "A3T")  ~ "unknown",
    str_detect(allele, "A3G") ~ "damaging",
    str_detect(allele, "T5G") ~ "benign",
    str_detect(allele, "T6G") ~ "damaging",
    str_detect(allele, "T6C") ~ "benign",
    str_detect(allele, "G10C") ~ "unknown"
  )) %>%
  # I then select rows that have "damaging" under the sig column
  filter(str_detect(sig, "damaging")) %>%
  # I select only the sampleid
  select(sampleid)
```

```
## # A tibble: 2 x 1
##   sampleid
##   <chr>
## 1 S1
## 2 S2
```

```
variants %>%
  # Pivoting to a long format and storing the result
  pivot_longer(contains("var"),
               names_to = "var",
               values_to = "variant") %>%
  select(1, 3) -> variants_2

# To make it easier, I select only rows that have damaging under the significance column
variant_significance %>%
  filter(str_detect(significance, "damaging")) -> variant_sig

# I then carry out a semi-join with variant as the key
subj_damag_gene <- semi_join(variants_2, variant_sig, by = "variant") %>%
  select(sampleid)
subj_damag_gene
```

Try using semi-join to achieve the same result.

```
## # A tibble: 2 x 1
##   sampleid
##   <chr>
## 1 S1
## 2 S2
```