Babraham Bioinformatics

Exercises: Advanced R

(with Tidyverse)

Version 2022-10

Licence

Exercise 1: Reading data into tibbles

- Load the `tidyverse` library into your R session
- Read `small_file.txt` into an R variable called `small.file`. This is a tab delimited text file so you can try to use the `read_delim` to read it. The file has some comments at the top which you'll need to deal with by either skipping the lines or setting a suitable `comment` argument.
    - Look at the data types for the 3 columns, see what type is given to the columns containing text data
    - Rerun the import but forcing the Category column to be imported as a factor (`col_factor`)

- Read in the `Child_Variants.csv` file into a variable called `child`. Add the `lazy=FALSE` argument to `read_delim` so you see any problems immediately.
    - Do you get any problems reported when reading the file?
    - Run `problems()` on the variable you saved to see the details of any issues. Can you see why they arise?

If you have time…

- Create a `geom_point` plot from the `child` data plotting the chromosome against the coverage. Note that you will get a warning about missing values because of the parsing error but it's OK to ignore that in this instance.

Exercise 2: Filtering and selecting with dplyr

Read in the `genomes.csv` file into a variable called `genomes`.

Perform the following operations:

- Make a list of the 5 organisms with the largest genomes (`arrange genomes` by `desc`ending `Size` and then extract the first 5 rows with `slice`)

- Of the organisms which have more than 40 chromosomes which one comes first, alphabetically? (Sort by Organism after filtering)

- Do any organisms containing a plasmid, also have more than one chromosome

- Make a version of the data containing only the columns from `Chromosomes` onwards

- Move the `Size` column to the front of the tibble

- Select just the columns which start with "O" – use the `starts_with` helper to do this.

- How many different groups are there? Deduplicate based on group and then pipe the result to `nrow`

If you have time…

Plot a scatterplot of the number of chromosomes in an organism vs the size of its genome. We only want to show one organism for each number of chromosomes so where multiple species

have the same number of chromosomes pick the one with the smallest genome. Don't plot a species which has no listed chromosomes.

Exercise 3: More clever filtering

Read in the "`cancer_stats.csv`" file into a variable called `cancer` then answer the following questions.

- For which Digestive System cancer types are there more female cases than male cases?
- Which cancer types (Class and Site) have no data (NA) for males? Use the `is.na()` function as part of the filtering to do this.
- Which cancer types (Class and Site) have no data (NA) for females?
- Which cancer type has the best survival rate for males? Order the tibble by Male Deaths / Male Cases
- Which cancer Sites have "acute" in their names?
- Out of tongue, kidney, breast and pancreas, which is classed as a soft tissue cancer?

If you have time…

- Find sites which have 4 letters or fewer in their names. Remember that the `nchar()` function will tell you the number of letters in a string.
- Find sites whose names ends with a "y". Note that you can't use `ends_with()` – that's for selecting columns. You can either use `str_sub()` from the `stringR` package, using a start position of -1 along with `== "y"` or you can use the core `endsWith()` function to do this.

Exercise 4: Restructuring data into 'tidy' format

You have been provided with three CSV data files (`tidy_data1.csv`, `tidy_data2.csv`, `tidy_data3.csv`). Load each of these and put them into tidy format. After loading each of the data files think about the following.

- Which of the columns are annotations and which are measurements?
- How many different types of measurement are there?
- Are all of the measurements of the same type in a single column?
- Do any annotation columns contain multiple pieces of information which have been concatenated together and would benefit from being split apart?
- Are any of the columns purely for annotation and might benefit from being split into another tibble to avoid duplication?
- After tidying are there any NA values which should be removed?

If you have time…

- In your genomes data split the Groups column into 3 new columns based on the semi-colon delimiter. Call the new columns “Domain”, “Kingdom” and “Class”

- Remove any species with a single quote in their name, or which has a Kingdom or Class of “Other”

Exercise 5: Adding or Creating New Data

- In your `cancer` data generate new columns called `cases` and `deaths` which sum up (literally add them together) the male and female values you have been given and overwrite the original `cancer` variable with the expanded version.

- When you imported the `child` data you had a warning because some of the `COVERAGE` values weren’t numeric and were imported as `NA`. Replace these `NA` coverage values with a fixed value of 1000.

- In `child` create a new column called `Type` which has a value of `SNP` if both `REF` and `ALT` are only 1 letter (again, use `nchar()` for this), and `INDEL` for all other cases. Overwrite the original variable.

If you have time…

- Change the `Type` column in `child` so that it can contain 3 values, `SNP` if `ALT` and `REF` are the same length, `INSERTION` if `ALT` is longer than `REF` and `DELETION` if `REF` is longer than `ALT`.

Exercise 6: Grouping and Summarising

- In your `small_file` data calculate the mean and standard deviations of the lengths for each category
  - Use `group_by` to define the grouping
  - Use `summarise` to say how you want to combine the quantitative values

- In `child` find genes which have at least 3 novel SNPs in them and calculate their average `COVERAGE`. This means that
  - The SNPs must have a `Type` of `SNP` – you’ll need to use the new variable you created in the exercise just above.
  - The SNPs must not have a `dbSNP` identifier (it should be a dot)
  - You will need to group by `GENE` and then get the mean `COVERAGE`, and use the `n()` function (no arguments needed) in your `summarise` function to get the number of items which were summarised
  - You can then `filter` the number of items for those with counts >2
  - Finally, `arrange` the results by `COVERAGE`, with the highest coverage at the top.

If you have time…

- In `tidy2` get the mean value for each combination of chromosome and category (`A`,`B`,`C` and `D`).

- Find which cancer type has the closest incidence rate between males and females

- Find which cancer type has the largest discrepancy in survival rates between males and females (not including types which only one sex can get)
- For each class of cancer find out which Site has the best overall survival rate
- In `child` find which gene on each chromosome has the highest number of variants.

Exercise 7: Extending and Joining

- Load the `dna_methylation.csv` file. This file contains counts for the numbers of methylated and unmethylated observations for a number of genes. Firstly you need to calculate the percentage methylation for each gene. To do this you will need to:
    - Use `pivot_wider` to split the `State` and `Count` columns into `Meth` and `Unmeth` columns
    - Use `mutate` to calculate the percentage methylation and place this into a new column.

- Once you have the percentage methylation per sample then use your grouping knowledge from before to get the mean methylation for each combination of `Gene` and `Group`.

- In a separate file, `methylation_annotation.txt` we have some additional annotation for the two genes which were measured. Combine this annotation with the summary from the last exercise to get a single tibble containing both the annotation and the data.

    - To combine the two tibbles use `rename` to change the name of the `Gene` column in your methylation data to `Gene_name` so that it matches the corresponding column in the annotation. This should allow you to do the join without any additional options.

If you have time…

- In `child` make a sorted list (from most frequent to least frequent) of all of the different observed SNP mutations (`REF` > `ALT` combinations). You will need to build a new column from the combination of `REF` and `ALT` (plus a delimiter). You can use the `unite` function to achieve this.

- Load the `small_file.txt` file. Generate a new column called `normalized_length` which is the difference between the existing `length` column and the shortest length for the category that measurement came from. To do this you will need to:
    - Group by `Category`
    - `Summarise` to generate a `min_length` column containing the minimum length for each category
    - Use a `right_join` back to the original data to transfer these values onto the full dataset
    - Use `mutate` to subtract the `min_length` from the `length` column to create the `normalized_length` value
    - Once you have the values calculated you can plot out the category against the normalized length values.

Exercise 8: Custom Functions

- Write a function which calculates the lowest quality for the variants in `child` in a specific gene. Have it take the tibble of data and a gene name as its arguments. It should output a one-line tibble with just the `GENE` and `QUAL` values in it. Use it to find the lowest quality for the AGRN gene. Make sure it works with a pipe.

- Modify the script so that you can now specify the column for which you want to find the lowest value. Have this default to `QUAL` but show that it can also work for `MutantReads` or `COVERAGE`