Assignment 4

Re-submit Assignment

Due 12 Aug by 23:59 **Points** 40 **Submitting** a file upload
Available after 8 Aug at 13:00

Assignment 4 – Due date: Aug 12, 11.59pm

Log in to Marvin. In your home directory, create a directory called “Assignment_4”. In Assignment_4, create three other directories called CODE, INPUT and OUTPUT.

Copy the file “/tutorials/Assignment_4/SWB_Full_v2.txt” into your INPUT directory.

Each line in this file contains regression results (Beta, SE, Pval) of the same outcome (subjective well-being) on a genetic variant specified by the rsID column. Beta is the regression coefficient, SE is the standard error, and Pval is the P-value.

In this assignment, you will write a bash script that

- removes lines containing duplicate “rsID” values,
- reads the data into R, and calculates the Z-statistic using two different methods,
- plots the two Z-statistics against each other,
- marks the “outliers” by checking whether the ratio of the Z-statistics is greater than a certain value,
- drops the “outliers”,
- re-generates the R plot with the “cleaned” data.

Below is a description of each step you need to implement:

1. Write an R script (name it “add_Zs.R”) that takes a file name as argument from the command line, reads in the data from this file into a data frame and calculates the Z-statistic using two methods:
    1. from the Beta and SE columns
    2. from the Pval column.

To pass command line arguments to R, remember to include the line

args=commandArgs(trailingOnly=TRUE)

at the beginning of your R script, and to refer to the file name as **args[1]** in the R script.

The Z-statistic is a standard normally distributed test statistic, in this case equal to Beta/SE. The P-values in the file are calculated using this statistic. For a two-tailed test, the following is the formula for obtaining P-value from the Z-statistic:

P-value = 2* (1 - Prob(Z_{st}<|Z|)) = 2 – 2*F(Z)

where F is the cumulative distribution function of a standard normally distributed random variable. So you

can back out the Z-statistic from the P-value as well:

$F(Z) = (2 – \text{P-value})/2$ à $Z = F^{-1}(1 – \text{P-value}/2)$

After you read the data from the input file into R, first calculate the Z statistic using the formula Beta/SE. You need to assign this vector to a new column named Z1 in your data frame. To add a column to a data frame, use the following syntax:

data$new_col <- expression

Then, calculate the Z statistic again using the P-values, and assign this vector to another column in the data frame named Z2. Given the P-value column (named Pval), this is the R command (replaces the “expression” in the command above) you can use to obtain the Z-statistic: **qnorm(1 - data$Pval/2)**

Your R script should then write the data (including the newly obtained Z1 and Z2 columns into a text file named “Z1_Z2_**args[1]**”, where **args[1]** is the input file name.

2. Create another R script (“Plot_Zs.R”). This one will also take a file name as input, and it will plot the Z1 values in the file against the Z2 values.

Start by "setting the stage", i.e. use the **jpeg()** function. The jpeg() function requires a filename to save the figure to your WD. Use the filename "Zplot_**args[1]**.jpg", where **args[1]** is the input file name.

To paste a file path with a string variable use: jpeg(file = paste0("../OUTPUT/", args[1]). This avoids overwriting the output file when you process a new input file.

Read the input file into a data frame. Use the plot() function to plot the absolute value of Z-statistics derived from the Beta and SE columns (|Z1|) on the x-axis, and the absolute value of the Z-statistics derived from the Pval column (|Z2|) on the y-axis. The color of the points should be "blue".

Change the x-axis label to " Z_beta_se", and the y-axis label to "Z_pval".

Add the following title to the figure (and change Your_name to your first name):
"Your_name's Z-plot".

Remember to end the code with the **dev.off()** function to tell R to finalize the output file.

3. Now, you will write a bash script (Your_name.sh) that takes as argument a file name. The script should start with an if statement that checks the number of arguments supplied to the script, and displays two different error messages if the number of supplied is arguments is wrong:

“Error: No file name has been supplied.” if no argument has been supplied, and

“Error: You have supplied more than one file names. Please specify only one.” if more than one argument has been supplied.

If exactly one argument has been supplied, the if statement should display

“The script is being executed”.

Note: Since the input file name is supplied as an argument to the script, you will refer to the file name using “**$1**” in the script. You can also include the following line at the beginning of the script if using “**$1**” throughout

seems confusing:

file=$1

After this line, you can refer to the input file as **$file** instead of **$1**.

4. Next, you will write a function that removes rows with duplicate values in the first column. Name this function “remove_dups”. The function should take one argument, the file name (which you are now referring to as **$file** if you followed the suggestion in the previous note). It should, for example, remove the following two lines (among others) because they have the same value in the first column.

rs6587766	1	57708088	T	C	0.04104	0.047	0.009	2.515e-07
rs6587766	1	57708088	A	C	0.02340	0.051	0.008	1.771e-10

Redirect the output so that it’s written into another file with the “**Nodups**_” prefix, i.e. if the input file is called “**$file**”, the output should be called “**Nodups_$file**”.

Note: Remember that you have to refer to function arguments using positional parameters. So inside the remove_dups function, you have to refer to the file name as $1.

Hint: Check out the options of the uniq command!

5. Apply this function to your input file (**$file**), using the first column to check for duplicates.
6. Apply the add_Zs.R script to **Nodups_$file**, i.e. the file you obtain at the end of step 5.
7. Apply the Plot_Zs.R script to **Z1_Z2_Nodups_$file**, i.e. the file you obtain at the end of step 6.
8. Using **awk**, create another column in **Z1_Z2_Nodups_$file** that takes the value 1 if |Z1/Z2| > 1.1 or |Z2/Z1| > 1.1 and 0 otherwise. Name the column “Z_outlier”.
9. Using **awk**, remove all lines with Z_outlier=1. The output file should have the prefix “Cleaned_”, i.e. it should be named “**Cleaned_$file**”
10. Re-run Plot_Zs.R with argument “**Cleaned_$file**”.

Save your script (Your_name_Assignment_4.sh). Run it with the “SWB_Full_v2.txt” argument. The script should run without error and produce two plots.

Upload Your_name_Assignment_4.sh and the two plots to Canvas.