File contents

* multi\_dist\_test\_data.R: R script to generate known (uncorrelated) distributions, to test the program can identify the distributions correctly. In the script, set the correct working directory path on Line 2, number of rows (n) to generate on line 3, and filename on line 4 before you run it. Then it generates one each of the included distributions and writes it to CSV.
* Test data files for distributions:
  + Algerian\_forest\_fires\_dataset\_UPDATE-JH-edit2.csv <https://archive.ics.uci.edu/ml/datasets/Algerian+Forest+Fires+Dataset>++
  + abalone.csv <https://archive.ics.uci.edu/ml/datasets/abalone>
  + test\_uncorrelated\_distributions.csv (from above R script)
* launch\_dashboard.R: run this script to launch the shiny dashboard. In the script, set the correct working directory path on Line 1 and make sure required packages are installed (uncomment lines 3-11 if necessary).
* shiny\_dashboard: folder containing the dashboard files:
  + app.R: Shiny app launched by launch\_dashboard.R
  + fit\_distribution.R: helper function to fit the distributions given a dataframe of data and generate the synthetic data

app.R file

follows standard Shiny file format

* ui variable (fluidPage function) lines 8-73 defines dashboard UI
  + 1st row of dashboard (lines 9-32) contains:
    - In the first column: the input csv upload button, the number of rows to generate-defaulting to 100 (NOTE I have set a max of 100,000 rows to generate on lines 17 and 62 but this is arbitrary and could be increased), the fit method (AIC, BIC, or Loglikelihood)
    - In the second column: a sample of the uploaded data
    - In the third column: a button to download the generated synthetic data (in CSV format)
  + 2nd row (lines 33-37) contains the header for the correlation plots
  + 3rd row (lines 28-52) contains the correlation plots, one in each column, starting with the plot for the original (target) data, then the synthetic data, then the difference between the two.
  + 3rd row (lines 53-62) contains density plots for comparison of the same variable from input data and synthetic data.
  + 4th row(lines 63-73) contains percentage comparison for categorical variables from input data and synthetic data.
* Server variable (input, output function) lines 75-200 does the actual input/output/calculations of the dashboard and includes simple ggplot functions
  + Rows 78-84 use the shinyvalidate package to specify that the inputs are required, number of rows must be an integer between certain values.
  + Rows 86-104 do the calculations required for the dashboard:
    - dist\_data reads in the input CSV
    - dist\_corr calculates the correlation matrix of the input data (note chars\_to\_ints is a function from fit\_distribution.R lines 10-17 which convert a categorical variable in a dataframe to an integer variable, as correlations can’t be done directly on character variables)
    - synth\_data calls the fit\_distribution function (from fit\_distribution.R) to generate the synthetic data (see above chars\_to\_ints note) (NOTE if you want the program to display the best fit distribution parameters, set debug\_mode = TRUE on line 77)
    - synth\_corr calculates the correlation matrix of the synthetic data
  + Rows 106-209 create the outputs specified in the UI function:
    - output$contents is the file contents preview, done with a DataTable to enable horizontal scrolling (scrollX = TRUE)
    - output$corr\_plot\_target is the correlation plot (with the corrplot R package) of the input (target) data
    - output$corr\_plot\_synth is similarly the correlation plot of the synthetic data
    - output$corr\_plot\_diff is the difference of the target and synthetic correlations
    - output$density is simple ggplot function to generate density plots for all continuous variables
    - output$percent is similarly ggplot function to generate comparison charts for discrete variables

fit\_distribution.R file:

main function (fit\_distribution) takes an input dataframe, best fit statistic, and number of rows to create and identifies the distribution for each column of the dataframe, and then creates synthetic correlated data. NOTE debug\_mode=TRUE parameter outputs the best fit distribution parameters to the standard output.

* Lines 22-26 set up all possible distributions to test
  + For discrete, positive numbers: binomial, negative binomial, poisson
  + For continuous numbers between 0 and 1: beta
  + For numbers greater than 0: gamma, lognormal, Weibull
  + For all numbers: normal, uniform
* Lines 28-50 saves properties a given column of data has
  + If a variable is a character variable, it’s categorical
  + Otherwise, if it’s all whole numbers (is\_wholenumber function at top of file), it’s an integer
  + If it’s greater than zero
  + Or if it’s continuous and between 0 and 1
* Lines 52-63 build the list of allowed distributions to test for a given variable (for non-categorical variables)
  + Any variable gets tested with the normal and uniform distribution
  + Discrete, positive variables get tested with the binomial, negative binomial, poisson distributions
  + Continuous variables between 0 and 1 also get tested with the beta distribution
  + Variables > 0 are tested with the gamma, lognormal, Weibull distributions
* Lines 65-69 categorical variables are fit with a frequency table approach, so if e.g. the variable gender is 40% Male and 60% Female in the original data, that frequency table (40%/60%) is saved as the best fit for that variable (in best\_dist\_list)
* Lines 70-106, for non-categorical variables, fit each distribution from the allowed list of distributions for that variable
  + For each of the distributions to test, use the fitdist function in the fitdistrplus package to fit the distribution. A summary of the fits for a given variable is saved in the ‘fits’ data frame, and the detailed parameters for each distribution in the ‘res\_list’ variable.
    - For fitdist to fit a binomial distribution (line 77), the size (total number of trials) needs to be provided. This can be estimated as the maximum of the values in the column (the number of successes in a sequence of *n* independent experiments – assumes the max value is that all trials are successful). The size also needs to be saved as an additional parameter in the results summary so it can be used when the distribution is generated (line 86).
  + The loglikelihood statistic should be maximized, while AIC and BIC should be minimized, so we take the negative of the value (test statistic) in line 93 so that all test statistics (AIC, BIC, loglikelihood) can be minimized.
  + The best distribution for a given variable is the one with the lowest test statistic (line 96)
  + Then the parameters for the best distribution, as well as the variable name, and name of the distribution, and whether it is an integer are saved in best\_dist\_list (lines 97-104)
* Lines 108-110 if ‘debug\_mode’ is enabled, print the best fit distribution for each variable
* Lines 112-115 get the correlation from the original data (cor function), then create a multi-variate normal distribution with the correlations
* Lines 117- 152: generate the correlated synthetic data for each variable in the original data:
  + Line 118 transform the correlations to a uniform distribution
  + Transform the correlated uniform distribution to the distributions of choice (saved in best\_dist\_list) and save in the variable synthetic\_var
    - For categorical, use a frequency table approach (lines 121-135). The uniform distribution is broken into bits based on the frequency, and assigned to a category. So now synthetic\_var has categorical variables in the frequencies that match the original, with the correlations from the uniform distribution (here is an example with 20% CAT A, 30% CAT B, and 50% CAT C):  
      |------CAT A-----|----CAT B-----|-----CAT C-------|  
      **0** *0.2*  **0.2**  *0.3*  **0.5**  *0.5*  **1  
      Bold**=cum\_prob, *italic* = tab (best\_dist\_list$frequencies)
    - For non-categorical variables (lines 138-139), use the do.call function to turn a text string into a function to generate the synthetic distribution with the inverse cdf (qnorm/qbinom/qunif…) with the correct parameters and correlations from the uniform distribution, e.g. if ‘norm’ (normal) is the best distribution with mean 0 and SD 1, then it would generate the string:  
      qnorm(p=uniform\_dist, mean=0, sd=1)
  + For integer (discrete) variables, round the synthetic data so the outputs are integers (line 141-142)
  + Lines 145-152 create the output dataframe with the generated synthetic data