Unit 1 assignments – should run as 1 or more fully debugged python/R/julia script(s) in the OS of your choice. You should start working on these step by step throughout the semester and be asking for help/clarification where needed.

IMPORTANT – Sample R code for the class demos can be found at this GitHub website/repo which is also linked from my personal webpage

https://people.rit.edu/gabsbi/

https://github.com/gbabbitt/course-code-repo

1. Write a commented header at the top of your first script file (e.g. FullName_unit1_BIOL672.r) that includes your name and operating system upon which you have designed and implemented your Unit 1 script. Include here an updated list of all libraries and/or packages you have downloaded and used on your machine starting with ggplot2…etc. Also include a list of the data files will be supplying me with to test run and grade your assignment. You script can be designed to use data files found in the same folder in which your script lives if you prefer not to have hard coded paths specific to your own file system. You are welcome to start with others code that you find on the usual help sites (Quick R, or Stack Overflow), however be sure to properly comment sections of code to show me that you understand how the code works. Submit your script to the unit1 dropbox on myCourses when it is due during week 8. DO NOT PROCRASTINATE…work on this every week during studio time and additional time when needed. You may want to also maintain a GitHub account so you can safeguard versioning of your work and so I can more easily it when needed.

2. Create a dataset of 5000 random numbers generated in in R using any model distribution you like. Write an R script that reports the sample mean and sample standard deviation and then plots a histogram from this data and that includes both a density line laid over the histogram bars. For extra effort try adding a normal bell curve model fitted to the data and overlaid as well. Try using ggplot2 and/or plotly libraries for creating plots in this part of the assignment.

3. To automate the printing of stat test output in R you will want to explore the 'sink' and 'unsink' functions in R. Use your script to create a file called 'desc.txt' and print your mean and SD from #2 to it. Add lines to your script to rename and save Rplots.pdf as histo.pdf.

4. Find an appropriate data set for a one-way ANOVA. Shape the data so you can use the read.table function from an input data file (as demo during class). Extend your script so that that it conducts a oneway ANOVA (i.e. oneway.test(values, category)). Add to this script, an error bar chart output of your categories with colored bars of your choice. Lastly add pairwise t tests for each category pair and use both Bonferroni and Benjamini-Hochberg multiple test correction using the p.adjust function in R. As in #3, export output to appropriately named text and pdf files. Be sure to include verbal interpretation printed to screen and/or output files.

5. Use a Kruskal Wallis test applied to your ANOVA input data to examine it without assumptions of normality. https://www.statmethods.net/stats/nonparametric.html Choose two or more or your categories and test the correlation between them using both Pearson and Spearman rank

methods. https://www.statmethods.net/stats/correlations.html Make scatterplots showing these relations. Run a one sample KS test to test your assumption of normality. Did this assumption hold? Did outcomes of parametric and nonparametric tests appear consistent? Why or why not? Be sure to include verbal interpretation printed to screen and/or output files.

6. Run a simple linear regression (e.g. lm function in R) on the same comparison from your correlations in #5. How does result compare? When is regression appropriate instead of correlation? Be sure to include verbal interpretation printed to screen and/or output files. Be sure to plot your results. IMPORTANT: from this point forward and throughout the rest of the course, always use the ggplot2 package to create your plots. https://www.rdocumentation.org/packages/ggplot2/versions/3.3.2

7. Find an appropriate data set from your field (or use your own thesis data) for a series of multivariate statistical tests....similar to the Iris data set demo in class. It should ideally include 4 to 6 measured quantitative variables that are intercorrelated and 1 categorical variable. IMPORTANT: Shape the data so you can use read.table function to input data similarly to how I demonstrate this in class. NOTE: you may create a second R script for this latter part of the Unit assignment that is importing and analyzing your own data.

8. Use the 'manova' function in R or an equivalent in python or Julia to add a MANOVA to this script using the command 'cbind' to combine the measurements (the dependent variables) into a single test of significance across category. Be sure to include verbal interpretation printed to screen and/or output files.

9. Utilize the lm function in R or its equivalent in python or Julia to conduct a multiple regression that tells how one variable is predicted by the remaining variables. In your interpretation state which one is the best predictor? Conduct the same test again but within one of your categories instead of across all of them. Be sure to include verbal interpretation printed to screen and/or output files.

10. Create a composite variable (e.g. like 'size' in Iris dataset as =pl*pw+sl*sw) that makes sense to you using your own data set. Ratios combining variables are also indicative of many features. Use the 'aov' function in R to set up an ANCOVA that compares the relation of one variable predicted by by another while controlling for your composite feature. Be sure to include verbal interpretation printed to screen and/or output files.

11. Choose a favorite hypothesis test or statistical method and compare it on the original iris data set vs three or the corrupted iris data sets found here. https://github.com/gbabbitt/course-code-repo/tree/main/statistics_(BIOL470-672)/corrupted-extended_iris_data_sets

    Report how sensitive your method is to the modifications in these data sets.

12. Use the extended iris data sets found in the folder above to analyze the categorical and ordinal (i.e. Likert scaled) results from the purchase of iris flowers by customers. Report any significant trends you discover.

Use the following R functions princomp() and factanal() described on the Quick-R website https://www.statmethods.net/advstats/factor.html or their python/Julia equivalents

13. Conduct a principal components analysis of the individual measurements and output that covers the following questions. NOTE : do not include categorical variables in your dataframe. How successful was the reduction of the data (explain using the scree plot as a reference)? How would you interpret the loadings of the original measurements on each of the PC's? Which one best describes overall size variation (i.e. PC with all positive loadings)? Do any of the PC's appear strongly driven by one particular measurement (see loadings on PC)?

14. Conduct a Factor Analysis (using same or new data if applicable). Can you determine any common latent underlying 'factors' that in your data? How many such factors are significant? What important characteristics can you find that tend to cluster together? If two traits are far apart along the axis of a significant factor, what does this indicate? What if they are close together? Was this factor analysis particularly successful in identifying a 1 or 2 large underlying latent variables that define something about your data?

15. Create a scatter plot of two of your most interesting variables (or can use PC 1 against PC 2 from #11 if you prefer) and color the points of the plot according your main categorical variable. Visually determine how many potential 'clusters' of data points you see in your plot. This is 'k'. Use the kmeans function to run k means clustering (i.e. unsupervised machine learning feature extraction) and create a plot where the points are colored by cluster membership.

Fitting and evaluating models to complex data

16. Large complex data sets are becoming more common in this 'age of information'. Simple models of probability density, which often assume that underlying variability is due to a single process (e.g. sampling error) often can fail to describe fully the underlying 'latent' or unobservable aspects of this kind of data. Therefore we need to employ more sophisticated algorithms to fit models to complex data. Multimodal density distributions are often indicative of underlying latent variability. One common solution to fitting density functions to multimodal density is to use a Gaussian Mixture Model fitted (GMM) by the Expectation-Maximization (EM) algorithm as we discussed in class. In this last part of the assignment you will fit three simple probability density functions (normal, lognormal and another of your own choosing) to the distributions of one of the independent variables (or PC1 representing the most reduced form of your data if you prefer) . You will also fit a GMM as a fourth model of probability density. All models will be compared and evaluated using the Bayesian Information Criterion (BIC) as a method of multimodel inference. Your goal is to determine which model of the four (normal, lognormal, yourChoice and GMM) best fits the distribution. (Iris flower size will be demo in class)

17. Write a script that imports or loads the data set, fits the models to your distribution, calculates likelihoods and compares BIC and plots histogram(s) of size with each model fitted to the data. Did the model testing indicate the presence of latency (via GMM having best fit?) Be sure to include verbal interpretation printed to screen and/or output files.

Some helpful links for the R code

For overview of GMM and EM

http://tinyheero.github.io/2016/01/03/gmm-em.html

For basic probability functions built into R

http://www.stat.umn.edu/geyer/old/5101/rlook.html

For ML fitting of these simple functions with ‘fitdistr’ for package ‘MASS’

https://www.rdocumentation.org/packages/MASS/versions/7.3-47/topics/fitdistr

For fitting GMM with EM using ‘normalmixEM’ for package ‘mixtools’

https://www.rdocumentation.org/packages/mixtools/versions/1.0.4/topics/normalmixEM

For BIC function in R

https://www.rdocumentation.org/packages/nlme/versions/3.1-1/topics/BIC

NOTE: The BIC function only works with the simple built in density functions in R (dnorm, dlnorm and dexp), so to get the BIC from your fitted GMM you will need to call on the log-likelihood returned by the function ‘normalmixEM’ and manually plug it into the BIC equation (=-2log-likelihood – 4*log(N)) noting that your data set has N observations and the GMM has 4 fitted parameters (2 mu’s and 2 sigma’s)