MATH 185 – Final

Due Wednesday, 06/08/2016, by 11:59 PM

Send your code to `math185ucsd@gmail.com`. *Follow the following format exactly. In subject line write "MATH 185 (Final)" and nothing else in the body. There should only be one file attached to the email, named* `final-lastname-firstname.R`. *Make sure your code is clean, commented and running. Keep your code simple, using packages only if really necessary. If your code does not run, include an explanation of what is going on.*

AGREEMENT: By taking this exam, you agree to not discuss the exam with anyone, starting now, neither with a classmate or anyone else, neither in person nor through other means, including electronic. Unless otherwise specified, it is acceptable to copy-paste from the lecture or homework solution code.

Problem 1. (Tests for goodness of fit) The two main goodness-of-fit tests that we saw for two-sample numerical data are the two-sided Wilcoxon rank-sum test (same as the Mann-Whitney U-test) and the two-sided Kolmogorov-Smirnov test. Perform some simulations to compare their performance in the following two situations:

- **Location model.** Assume that the first sample (the X's say) is standard normal while the second sample (the Y's) is normal with mean μ and unit variance. Fix the sample sizes to the same number n. Consider three cases $n \in \{20, 50, 100\}$. For each n, vary μ in some carefully chosen grid (say of five different values) so we see the performance of the tests improve as μ gets larger. Repeat each setting (given by the pair (n, μ)) $B = 200$ times. [You can restrict yourself to $\mu > 0$ by symmetry. Please do so.]
- **Scale model.** Same as above except that here the second sample is normal with zero mean and standard deviation σ. Here σ plays the role of μ.

Problem 2. (A variant of the bootstrap test for the one-way layout) Consider a one-way layout with an arbitrary number of groups. There was some confusion among some students about how to use the bootstrap in this situation. In lecture, the bootstrap was done within each group, meaning, the bootstrap group j is made by sampling from group j with replacement. This allows one to really test for the equality of means (for example) as opposed to test for the equality of the distributions. Consider the following variant. Instead of the above, form each bootstrap group by sampling (with replacement) from the concatenated sample (all groups combined). Implement this method as a function `boot.combined.test(dat, B = 999)`, where `dat` is a data frame and `B` is the number of bootstrap repeats. As a test statistic, use the treatment sum of squares. The function returns the P-value, as usual. Test your function on the smokers dataset. What is the most appropriate null hypothesis in this case?

Problem 3. (Multiple testing for categorical data) In the context of two-sample categorical data, we saw how to test for goodness-of-fit. However, observing the test rejecting, we do not know how the two samples differ significantly from each other. In more detail, suppose we have two samples, $X_1, \ldots, X_m$ and $Y_1, \ldots, Y_n$, both taking values in $\{1, \ldots, k\}$ (this is without loss of generality as you know). We want to identify those j such that $\mathbb{P}(X = j) \neq \mathbb{P}(Y = j)$. For this purpose, implement a method in R, named `multiple.categorical.test(x, y, ...)`, taking the two samples as vectors `x` and `y`, and possibly other input variables, and returns a list of indexes $j \in \{1, \ldots, k\}$ where there is a (statistically) significant difference between the samples. Right before the code for this function, explain in words what the function is doing in a fair amount of detail. Perform some simulations to test your function — do so in a setting where k is not too small. [Take the multiple testing nature of the problem into account.]

Problem 4. (Cross-validation for smoothing splines) The `smooth.spline` function in the `splines` package, as we saw, allows one to choose the tuning parameter in various ways (including by hand). Write a function `smooth.spline.CV(x, y, K=10)` for choosing the tuning parameter by K-fold cross-validation. Test your function on the same dataset used in `part11-kernel-regression.R`.