[notations]

CCRAM = checkerboard copula regression association measure

SCCRAM = scaled CCRAM

CCR = checkerboard copula regression

Y = ordinal dependent variable

X_1, ..., X_p = p (nominal/ordinal) independent variables

1. bootstrap confidence interval for CCRAM or SCCRAM

The link below shows the python function to compute three types of bootstrap confidence intervals, {'percentile', 'basic', 'bca'}.

https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.bootstrap.html

Here are a few things to consider to use this function for computing the bootstrap confidence interval for CCRAM or SCCRAM.

(1) It seems that this function requires the case form data. This means that you need to change the table form data to the case form data to use this function.
(2) You may be able to use your CCRAM or SCCRAM function in "*statistic" input option of this function. Please look at the example given on the website. It seems that there is an example (with "my_statistic") which shows how to use the user-defined functions*
(3) It is important that we need to use all options available in this function.

Here are the outputs we need from this function

(1) Observed CCRAM or SCCRAM : value of CCRAM or SCCRAM computed from the original data
(2) Bootstrap CCRAM or SCCRAM : Bootstrap values for CCRAM or SCCRAM to visualize the bootstrap distribution of CCRAM or SCCRAM
(3) Bootstrap confidence intervals for CCRAM or SCCRAM based on the option "*method" and "confidence_level"*

2. uncertainty in the prediction of Y from CCR using bootstrap

We also need to obtain the bootstrap-based proportions of Y=j using CCR where j = 1, .., J and J is the number of category of Y.
Although there is no python code for this, I think you can use the bootstrap function above to achieve this goal. If you can find the source code of the python function above, I think you can find a code to generate B bootstrap samples generated from the original case form data. If this is the case, you can create a new function by combining bootstrap sampling code with your code to predict Y from CCR.

The outputs we need to obtain are

(1) Values of predicted category of Y at each combination of categories of X_1, ..., X_p computed from the original data
(2) Table output which shows the proportion of Y=j from CCR at each combination of categories of X_1, ..., X_p computed from B bootstrap samples where j=1, ,,,, J

3. permutation test for the null hypothesis of H0: SSCRAM = 0

The sample for the permutation test must be generated under the null hypothesis of interest. In our case, the hypothesis of interest is H0: SSCRAM = 0, meaning that Y and X_1 are independent.

In order to understand the permutation test for the contingency tables, please look at the section of "**Permutation Test for Independence**" in the following website:

https://alanarnholt.github.io/GeneralStatistics/rmarkdown/ContingencyTables.html

Although this website is based on R, it would be helpful for you to understand the idea. Note that we need to use CCRAM or SCCRAM instead of the statistic used in this website.

- The idea is that you need to change the table form data to the case form data, simulate B permutation samples under H0 (i.e., Y and X_1 are independent), change each permutation sample to the table form data again, and then compute CCRAM or SCCRAM for each permuted samples.
- To generate the permutation sample under H0, you need to permute Y and leave X_1 invariant. Note that the permutation of X_1 would lead to the same value of CCRAM and SCCAM due to Proposition 3-(vii) in my 2021 paper.

- Please read the paper that I attach to the email, “A Permutation-Based Kernel Conditional Independence Test”, which will provide you detailed explanation on the permutation sampling under H0.

The inputs for your function may include the data, B (number of permutation), and observed value of CCRAM or SCCRAM.

The outputs we need to obtain are

(1) Observed CCRAM or SCCRAM : value of CCRAM or SCCRAM computed from the original data
(2) B Permutation CCRAM or SCCRAM : values for CCRAM or SCCRAM computed from each of B permuted data to visualize the permutation distribution of CCRAM or SCCRAM
(3) P- value = # of B Permutation CCRAM or SCCRAM values in (2) which is less than equal to Observed CCRAM or SCCRAM in (1)

[Note]

The python has the permutation test function

https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.permutation_test.html#scipy.stats.permutation_test

However, it is not clear if you can use it for CCRAM or SCCRAM, though you may need to look at it and see if you can find useful information. For example, this function can provide the exact value of P-value if the number of distinct permutations in the data is smaller than B (number of permutation specified by the user).