Algorithm 1 Fast Conditional Independence Test

Result: Test-statistic and associated p-value for the null hypothesis that $X \perp\!\!\!\perp Y|Z$

function CROSSVAL(X, Y)

  return $Y = f(X, \beta) + e_i$

end function

function MSE(y, $\hat{y}$)

  return $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

end function

x $\leftarrow \mathbf{X} \in \mathbb{R}^{n\times m}$; y $\leftarrow \mathbf{Y} \in \mathbb{R}^{n\times l}$; z $\leftarrow \mathbf{Z} \in \mathbb{R}^{n\times m}$ where $l < m$;
numSamp $\leftarrow n$;
fracTest $\leftarrow f \in \mathbb{R}, 0 < f \leq 1$, the proportion of data to evaluate ;
numPerm $\leftarrow r \in \mathbb{R}$ where numPerm is the number of training repetitions;
nTest $\leftarrow$ floor(fracTest $\times$ numSamp);

bestTreeX $\leftarrow$ CrossVal(concat(x, z), y) regressor for $y = f(x, z)$;
bestTreeNoX $\leftarrow$ CrossVal(z,y) regressor for $y = f(z)$;

msesX $\leftarrow$ list(); msesNoX $\leftarrow$ list() for storage of mean squared errors;

for $j = 1 : r$ **do**

  idx $\leftarrow$ permutation(numSamp);
  Xtest, Xtrain $\leftarrow$ x[idx][:nTest], x[idx][nTest:];
  Ytest, Ytrain $\leftarrow$ y[idx][:nTest], y[idx][nTest:];
  Ztest, Ztrain $\leftarrow$ z[idx][:nTest], z[idx][nTest:];

  bestTreeX.train(concat(Xtrain, Ztrain), Ytrain);
  msesX.append(MSE(bestTreeX.predict(concat(Xtrain, Ztrain), Ytest));

  bestTreeNoX.train(Ztrain, Ytrain);
  msesNoX.append(MSE(bestTreeNoX.predict(Ztrain, Ytest));

end

stat, pval $\leftarrow$ KSampleTest(msesX, msesNoX);