```
maxk = 14; (*max k parameter*) n = 10; (*number of functions in each set*)
repsp = 10; (*times we use the same probability value k*)

complexity = Table[0, repsp]; (*list of K-complexity (1-D representation) for the same k value*)
complexitymat = Table[0, repsp];
(*list of K-complexity (2-D representation) for the same k value*)
entropy = Table[0, repsp]; (*list ofentropy values for the same k value*)
ShannonEntropy = Table[0, maxk]; (*list to save the entropy for each k value*)
seqnet = Table[0, maxk];
(*list to save the K-complexity for each k value (1-D representation)*)
Matnet = Table[0, maxk]; (*list to save the K-complexity for each k value (2-D representation)*)

Do[Do[k = l; (*parameter k*)
  booleanfunction = {}; (*boolean function*)
  Matrixbooleanfunctionlist = {}; (*matrix representation*)
  Clear[a];
  A = Table[a[i], {i, 1, k}]; (*boolean variables*)

      (*generate the set of boolean functions*)
  functions = Table[0, n];
  For[i = 0, i < n, i++,
   f = BooleanFunction[RandomInteger[{0, (2^2^k - 1)}], k];
   (*choose randomly one of the 2^2^k possible boolean functions*)
   AppendTo[booleanfunction, f];
   AppendTo[Matrixbooleanfunctionlist,
    Boole[BooleanTable[BooleanConvert[Apply[f, A], "NOR"], A]]];
   functions[[i + 1]] = f];

      (*measure the complexity of the set*)
  matrixrepresen = Matrixbooleanfunctionlist;
  sequencerepresen = Flatten[matrixrepresen];
  If[First[sequencerepresen] == 0, output = "0" <> ToString[FromDigits[sequencerepresen]],
   output = ToString[FromDigits[sequencerepresen]]];
  complexity[[j]] = StringBDM[output];
  complexitymat[[j]] = BDM[matrixrepresen, 4] // N;
  entropy[[j]] = Entropy[output];
  , {j, repsp}];

     (*perform a trimmed mean discarding the first and fourth quartiles*)
 seqnet[[l]] = TrimmedMean[complexity,
   {(Length[Select[complexity, # < Quantile[complexity, 1/4] &]])/Length[complexity],
    (Length[Select[complexity, # > Quantile[complexity, 3/4] &]])/Length[complexity]}];
 Matnet[[l]] = TrimmedMean[complexitymat,
   {(Length[Select[complexitymat, # < Quantile[complexitymat, 1/4] &]])/Length[complexitymat],
    (Length[Select[complexitymat, # > Quantile[complexitymat, 3/4] &]])/Length[complexitymat]}];
 ShannonEntropy[[l]] = TrimmedMean[entropy,
   {(Length[Select[entropy, # < Quantile[entropy, 1/4] &]])/Length[entropy],
    (Length[Select[entropy, # > Quantile[entropy, 3/4] &]])/Length[entropy]}];
 , {l, 1, maxk}]

    (*plot the results*)
ListLinePlot[{Rescale[seqnet, {0, Max[seqnet]}], Rescale[Matnet, {0, Max[Matnet]}],
  Rescale[ShannonEntropy, {0, Max[ShannonEntropy]}]}, AxesLabel → Automatic, PlotRange → All,
 PlotLegends → Placed[{"K-Complexity (1-D)", "K-Complexity (2-D)", "Entropy"}, {.25, .3}],
 Frame → True, GridLines → Automatic, FrameLabel → {"k", "C(f)"}]
```