Density Matrix Calculation

Misc.

```
n = Length[Union[SET]] (* i.e. n = number of unique elements in the set *)
H[α_, SET_] := 1/(1 - α) Log[Sum[(prob_i)^α, {i, 1, n}]]
```

```
Count[{a, b, a, d, b, c, b}, b]
Length[{a, b, a, d, b, c, b}]
Count[{a, b, a, d, b, c, b}, b] / Length[{a, b, a, d, b, c, b}] (* Prob of getting b at random *)
Length[Union[{a, b, a, d, b, c, b}]] (* i.e. = number of unique elements in the set *)
```

```
3
```

```
7
```

$\frac{3}{7}$

```
4
```

PrimePi Renyi Entropy Calc

PCV1 Renyi Entropy Calc

General Definitional Terms

```
(* M=PCV1_1; *)
(* M={Join[PCV1_1[[1]],PCV1_2[[1]],PCV1_3[[1]],PCV1_4[[1]]]}; *)
(* M={Table[PCV1_1[[1,i]],{i,1,64}]}; *)
```

```
lengthvec[M_] := Length[M[[1, All]]]
For[npow = 1, npow < 1000, npow++, If[lengthvec[M] < (2^(npow)), Break[]];
  FilledSize = 2^(npow + 1)];
Filler[M_] := Table[4, {i, 1, FilledSize - lengthvec[M]}]
FilledVec[M_] := Join[Flatten[M], Filler[M]]
```

```
lengthvec[M_] := Length[M[[1, All]]]
Filler[M_] := Table[4, {i, 1, FilledSize - lengthvec[M]}]
FilledVec[M_] := Join[Flatten[M], Filler[M]]
```

```
For[npow = 1, npow < 1000, npow++, If[lengthvec[M] ≤ (2^npow), Break[]]];
(* gives npow such that  2^npow > lengthvec[M] > 2^(npow -1) *)
FilledSize = 2^npow;
```

PCV Type 1 samples

Renyi Entropies of PCV1 data sets

Density matric calc Using 2x2^(n/2) defn of W0 (not valid)

Density Matrix Calculation

Wrong *ρ*’s (as inner and outer products of W, Transpose[W])

Defn of W (RUN THIS before doing the below)

Misc. Code

Pre/detailed

Final

Just pick a W and Run the blue subsection

Constructing W’s

To get a letter sequence go to https://www.ncbi.nlm.nih.gov/nuccore/
pick a gene from GenBank
go to FASTA
Copy + paste letter seq into data subsubsection

```
StringReplace[ToString[{StringReplace["ACGTAGTCAATT",
    {"A" → "0,", "C" → "1,", "G" → "2,", "T" → "3,"}]}], ",}" → "}"]
{0,1,2,3,0,2,3,1,0,0,3,3}
```

```
ToString["ACGTAGTCAATT"]
ToString[ACGTAGTCAATT]
ACGTAGTCAATT // ToString
StringQ[ACGTAGTCAATT // ToString]
StringQ[ACGTAGTCAATT]
```

```
ACGTAGTCAATT

ACGTAGTCAATT

ACGTAGTCAATT

True

False
```

```
sample = "ACGTAGTCAATT" // ToString;
StringReplace[ToString[{StringReplace[ToString[sample],
     {"A" → "0,", "C" → "1,", "G" → "2,", "T" → "3,"}]}], ",}" → "}"]
```

```
{0,1,2,3,0,2,3,1,0,0,3,3}
```

```
lettersample = {ACGTAGTCAATT} // ToString;
```

```
LetterDNAtoNum[Sample_] := ToExpression[StringReplace[ToString[
     {StringReplace[StringReplace[ToString[{Sample}], {"," → "", " " → "", "{" → "",
          "}" → "", "(" → "", ")" → "", "[" → "", "]" → "", ";" → "", ":" → "", "_" → "",
          "+" → "", "&" → "", "/" → "", "." → "", "RowBox" → "", "Null" → ""}],
        {"0" → "0,", "1" → "1,", "2" → "2,", "3" → "3,", "A" → "0,", "C" → "1,",
         "G" → "2,", "T" → "3,", "a" → "0,", "c" → "1,", "g" → "2,", "t" → "3,"}]}
    ],
    ",}" →
     "}"]]
```

```
LetterDNAtoNum[lettersample]
```

```
{{0, 1, 2, 3, 0, 2, 3, 1, 0, 0, 3, 3}}
```

```
numgenesample = LetterDNAtoNum[lettersample];
```

```
Flatten[numgenesample][[3]]
```

```
2
```

```
lettersample = {ACGTAGTCAATT} // ToString;
LetterDNAtoNum[Sample_] := ToExpression[StringReplace[ToString[
     {StringReplace[StringReplace[ToString[{Sample}], {"," → "", " " → "", "{" → "",
          "}" → "", "(" → "", ")" → "", "[" → "", "]" → "", ";" → "", ":" → "", "_" → "",
          "+" → "", "&" → "", "/" → "", "." → "", "RowBox" → "", "Null" → ""}],
        {"0" → "0,", "1" → "1,", "2" → "2,", "3" → "3,", "A" → "0,", "C" → "1,",
         "G" → "2,", "T" → "3,", "a" → "0,", "c" → "1,", "g" → "2,", "t" → "3,"}]}
    ], ",}" → "}"]]
numgenesample = LetterDNAtoNum[lettersample];
Flatten[numgenesample]
```

```
{0, 1, 2, 3, 0, 2, 3, 1, 0, 0, 3, 3}
```

```
basepairs = ToString[Input["Paste the base pair sequence (ex: AAGCTATGG) here"]];
```

```
Wgenesample = StringJoin[ToString[Input["What Gene is this?"]], " gene"]
(*Lets us know which gene we're dealing with,
used in pdf coding later, so be sure to name it *)
```

```
BRCA2 mRNA Wolf gene
```

```
lettersample = {ACGTAGTCAATT} // ToString;
```

```
LetterDNAtoNum[Sample_] := ToExpression[StringReplace[ToString[
    {StringReplace[StringReplace[ToString[{Sample}], {"," → "", " " → "", "{" → "",
        "}" → "", "(" → "", ")" → "", "[" → "", "]" → "", ";" → "", ":" → "", "_" → "",
        "+" → "", "&" → "", "/" → "", "." → "", "RowBox" → "", "Null" → ""}],
      {"0" → "0,", "1" → "1,", "2" → "2,", "3" → "3,", "A" → "0,", "C" → "1,",
       "G" → "2,", "T" → "3,", "a" → "0,", "c" → "1,", "g" → "2,", "t" → "3,"}]}
   ], ",}" → "}"]]
numgenesample = LetterDNAtoNum[lettersample];
Export[StringReplace["GENE_genesample.txt", "GENE_gene" → Wgenesample],
 Flatten[numgenesample]]
```

```
wolf genesample.txt
```

Test of method of Construction of W

```
H1Avec =
  {{3, 0, 2, 2, 1, 3, 2, 1, 2, 3, 3, 2, 2, 2, 2, 1, 1, 3, 3, 3, 3, 3, 3, 3, 1, 2, 1, 0, 3, 1, 1, 3, 2,
    1, 3, 3, 1, 2, 3, 1, 0, 2, 2, 3, 3, 3, 0, 3, 0, 1, 1, 0, 1, 3, 3, 3, 0, 3, 3, 3, 2, 2, 3, 2, 3,
    2, 1, 3, 2, 3, 2, 3, 3, 0, 2, 3, 1, 0, 1, 1, 0, 3, 2, 3, 1, 3, 2, 0, 0, 0, 1, 0, 2, 3, 2, 1, 1,
    3, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 3, 3, 1, 3, 2, 1, 3, 2, 1, 3, 1, 1, 3, 2, 0,
    2, 0, 0, 0, 1, 1, 3, 3, 3, 0, 2, 1, 3, 2, 2, 1, 0, 0, 2, 0, 0, 2, 2, 1, 0, 0, 0, 2, 0, 0,
    0, 1, 1, 3, 2, 1, 3, 0, 0, 2, 2, 1, 3, 2, 1, 0, 2, 1, 0, 2, 1, 1, 3, 1, 1, 0, 0, 2, 0, 0,
    0, 0, 0, 0, 1, 1, 1, 2, 1, 3, 2, 2, 1, 1, 1, 3, 3, 1, 1, 2, 3, 2, 3, 1, 0, 2, 0, 2, 1, 3,
    2, 0, 3, 1, 2, 3, 2, 1, 0, 2, 2, 1, 3, 2, 1, 3, 3, 1, 1, 3, 1, 1, 3, 1, 3, 0, 0, 2, 2, 0,
    2, 1, 2, 3, 2, 2, 3, 2, 2, 3, 2, 3, 2, 3, 1, 2, 3, 3, 2, 2, 1, 0, 2, 1, 3, 1, 3, 3, 0, 0,
    0, 0, 0, 2, 2, 1, 2, 1, 3, 2, 2, 1, 2, 2, 1, 1, 2, 1, 0, 2, 2, 1, 3, 0, 1, 2, 0, 1, 2, 3,
    2, 2, 0, 2, 0, 0, 2, 0, 0, 1, 0, 0, 1, 0, 2, 1, 1, 2, 1, 0, 3, 3, 0, 0, 2, 1, 3, 2, 2, 2,
    1, 0, 3, 3, 0, 0, 2, 0, 2, 1, 1, 3, 2, 2, 3, 0, 0, 2, 1, 0, 0, 2, 2, 2, 0, 0, 1, 2, 3, 3,
    2, 2, 3, 2, 1, 0, 2, 0, 1, 0, 0, 0, 2, 2, 2, 3, 0, 1, 1, 2, 2, 0, 2, 1, 1, 3, 1, 2, 2, 2,
    3, 3, 1, 1, 3, 3, 1, 0, 0, 2, 1, 3, 1, 0, 0, 1, 0, 0, 2, 0, 0, 2, 2, 1, 2, 3, 1, 1, 3, 1,
    1, 2, 3, 2, 2, 0, 0, 0, 1, 1, 0, 0, 2, 1, 1, 1, 2, 2, 1, 2, 1, 1, 3, 1, 0, 0, 0, 2, 2, 3,
    2, 2, 1, 3, 0, 1, 0, 0, 0, 0, 0, 1, 3, 0, 0, 2, 2, 1, 0, 0, 1, 2, 2, 2, 3, 2, 1, 0, 3, 1,
    3, 0, 0, 0, 0, 0, 2, 1, 3, 1, 0, 0, 0, 0, 0, 2, 2, 1, 1, 0, 1, 2, 2, 2, 2, 2, 1, 3, 0, 2,
    1, 0, 0, 0, 0, 0, 2, 0, 2, 1, 2, 3, 1, 0, 0, 2, 0, 1, 3, 1, 1, 2, 0, 0, 0, 0, 0, 2, 2, 1,
    3, 0, 0, 0, 0, 0, 2, 1, 1, 3, 2, 1, 2, 2, 1, 0, 0, 1, 0, 0, 2, 2, 0, 0, 0, 3, 1, 1, 3, 1,
    1, 0, 0, 2, 0, 0, 3, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 3, 2, 3, 0, 0, 0,
    2, 1, 1, 1, 0, 0, 2, 0, 0, 0, 2, 3, 0, 2, 1, 3, 0, 0, 0, 0, 2, 1, 1, 1, 3, 2, 1, 3, 0,
    0, 0, 2, 1, 3, 0, 0, 2, 2, 1, 3, 2, 3, 0, 0, 0, 0, 1, 1, 1, 0, 0, 2, 2, 1, 2, 2, 1, 1,
    0, 0, 2, 2, 1, 3, 0, 2, 2, 2, 3, 2, 0, 1, 2, 0, 0, 2, 1, 1, 0, 0, 0, 2, 0, 1, 3, 2, 1,
    1, 0, 0, 0, 1, 1, 1, 0, 0, 2, 0, 0, 0, 2, 1, 2, 2, 1, 0, 1, 1, 1, 0, 0, 2, 0, 0, 0, 0,
    0, 2, 3, 0, 0, 0, 3, 3, 1, 0, 2, 3, 3, 0, 2, 0, 0, 2, 3, 3, 3, 1, 3, 3, 1, 3, 0, 2, 3,
    0, 0, 1, 1, 1, 0, 0, 1, 2, 2, 1, 3, 1, 3, 3, 3, 3, 0, 0, 2, 0, 2, 1, 1, 0, 1, 1, 3, 0}};
```

```
H1AW =
  {{3, 0, 2, 2, 1, 3, 2, 1, 2, 3, 3, 2, 2, 2, 2, 1, 1, 3, 3, 3, 3, 3, 3, 3, 1, 2, 1, 0, 3, 1, 1, 3},
   {2, 1, 3, 3, 1, 2, 3, 1, 0, 2, 2, 3, 3, 3, 0, 3, 0, 1, 1, 0, 1, 3, 3, 3, 0, 3, 3, 3, 2, 2, 3, 2},
   {3, 2, 1, 3, 2, 3, 2, 3, 3, 0, 2, 3, 1, 0, 1, 1, 0, 3, 2, 3, 1, 3, 2, 0, 0, 0, 1, 0, 2, 3, 2, 1},
   {1, 3, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 3, 3, 1, 3, 2, 1, 3, 2, 1, 3, 1, 1, 3, 2, 0, 2},
   {0, 0, 0, 1, 1, 3, 3, 3, 0, 2, 1, 3, 2, 2, 1, 0, 0, 2, 0, 0, 2, 2, 1, 0, 0, 0, 2, 0, 0, 0, 1, 1},
   {3, 2, 1, 3, 0, 0, 2, 2, 1, 3, 2, 1, 0, 2, 1, 0, 2, 1, 1, 3, 1, 1, 0, 0, 2, 0, 0, 0, 0, 0, 0, 1},
   {1, 1, 2, 1, 3, 2, 2, 1, 1, 1, 3, 3, 1, 1, 2, 3, 2, 3, 1, 0, 2, 0, 2, 1, 3, 2, 0, 3, 1, 2, 3, 2},
   {1, 0, 2, 2, 1, 3, 2, 1, 3, 3, 1, 1, 3, 1, 1, 3, 1, 3, 0, 0, 2, 2, 0, 2, 1, 2, 3, 2, 2, 3, 2, 2},
   {3, 2, 3, 2, 3, 1, 2, 3, 3, 2, 2, 1, 0, 2, 1, 3, 1, 3, 3, 0, 0, 0, 0, 0, 2, 2, 1, 2, 1, 3, 2, 2},
   {1, 2, 2, 1, 1, 2, 1, 0, 2, 2, 1, 3, 0, 1, 2, 0, 1, 2, 3, 2, 2, 0, 2, 0, 0, 2, 0, 0, 1, 0, 0, 1},
   {0, 2, 1, 1, 2, 1, 0, 3, 3, 0, 0, 2, 1, 3, 2, 2, 2, 1, 0, 3, 3, 0, 0, 2, 0, 2, 1, 1, 3, 2, 2, 3},
   {0, 0, 2, 1, 0, 0, 2, 2, 2, 0, 0, 1, 2, 3, 3, 2, 2, 3, 2, 1, 0, 2, 0, 1, 0, 0, 0, 2, 2, 2, 3, 0},
   {1, 1, 2, 2, 0, 2, 1, 1, 3, 1, 2, 2, 2, 3, 3, 1, 1, 3, 3, 1, 0, 0, 2, 1, 3, 1, 0, 0, 1, 0, 0, 2},
   {0, 0, 2, 2, 1, 2, 3, 1, 1, 3, 1, 1, 2, 3, 2, 2, 0, 0, 0, 1, 1, 0, 0, 2, 1, 1, 1, 2, 2, 1, 2, 1},
   {1, 3, 1, 0, 0, 0, 2, 2, 3, 2, 2, 1, 3, 0, 1, 0, 0, 0, 0, 0, 1, 3, 0, 0, 2, 2, 1, 0, 0, 1, 2, 2},
   {2, 3, 2, 1, 0, 3, 1, 3, 0, 0, 0, 0, 0, 2, 1, 3, 1, 0, 0, 0, 0, 0, 2, 2, 1, 1, 0, 1, 2, 2, 2, 2},
   {2, 1, 3, 0, 2, 1, 0, 0, 0, 0, 0, 2, 0, 2, 1, 2, 3, 1, 0, 0, 2, 0, 1, 3, 1, 1, 2, 0, 0, 0, 0, 0},
   {2, 2, 1, 3, 0, 0, 0, 0, 0, 2, 1, 1, 3, 2, 1, 2, 2, 1, 0, 0, 1, 0, 0, 2, 2, 0, 0, 0, 3, 1, 1, 3},
   {1, 1, 0, 0, 2, 0, 0, 3, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 3, 2, 3, 0, 0, 0, 2},
   {1, 1, 1, 0, 0, 2, 0, 0, 0, 2, 3, 0, 2, 1, 3, 0, 0, 0, 0, 2, 1, 1, 1, 3, 2, 1, 3, 0, 0, 0, 2, 1},
   {3, 0, 0, 2, 2, 1, 3, 2, 3, 0, 0, 0, 0, 1, 1, 1, 0, 0, 2, 2, 1, 2, 2, 1, 1, 0, 0, 2, 2, 1, 3, 0},
   {2, 2, 2, 3, 2, 0, 1, 2, 0, 0, 2, 1, 1, 0, 0, 0, 2, 0, 1, 3, 2, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 2},
   {0, 0, 0, 2, 1, 2, 2, 1, 0, 1, 1, 1, 0, 0, 2, 0, 0, 0, 0, 0, 2, 3, 0, 0, 0, 3, 3, 1, 0, 2, 3, 3},
   {0, 2, 0, 0, 2, 3, 3, 3, 1, 3, 3, 1, 3, 0, 2, 3, 0, 0, 1, 1, 1, 0, 0, 1, 2, 2, 1, 3, 1, 3, 3, 3},
   {3, 0, 0, 2, 0, 2, 1, 1, 0, 1, 1, 3, 0, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4},
   {4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4},
   {4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4},
   {4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4},
   {4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4},
   {4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4},
   {4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4},
   {4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4}};

numgenesample = H1Avec;

M = numgenesample;
lengthvec[M_] := Length[M[[1, All]]]
For[npow = 1, npow < 1000, npow++, If[lengthvec[M] < (2^(npow)), Break[]];
  FilledSize = 2^(npow + 1)];
Filler[M_] := Table[4, {i, 1, FilledSize - lengthvec[M]}]
FilledVec[M_] := Join[Flatten[M], Filler[M]]
lengthvec[M_] := Length[M[[1, All]]]
Filler[M_] := Table[4, {i, 1, FilledSize - lengthvec[M]}]
FilledVec[M_] := Join[Flatten[M], Filler[M]]
For[npow = 1, npow < 1000, npow++, If[lengthvec[M] ≤ (2^npow), Break[]]];
(* gives npow such that  2^npow > lengthvec[M] > 2^(npow -1) *)
FilledSize = 2^npow;
FilledM = FilledVec[M];
```

```
npow
Length[FilledM] (*Should be an (even) power of 2*)
numrowsW = √Length[FilledM]
```

```
10

1024

32
```

```
(*Table[FilledM[[i]],{i,1,numrowsW}]
 Table[FilledM[[i]],{i,33,2*(numrowsW)}]
 Table[FilledM[[i]],{i,(2*(numrowsW))+1,3*(numrowsW)}]*)
```

```
testWW = Table[Table[FilledM[[i]],
    {i, (((j - 1) * (numrowsW)) + 1), (j * (numrowsW))}], {j, 1, numrowsW}];
```

```
Print["so this construction is valid if and only if the following:   ",
 H1AW - testWW // Union // Flatten // Union ,
 "   is exactly {0}, if not something is wrong"]
```

```
so this construction is valid if and only if the following:
 {0}   is exactly {0}, if not something is wrong
```

W for Wolf BRCA2 mRNA

Data

Source: GenBank: AB043895.5

```
(*lettersample={}//ToString;*)
```

```
lettersample = {AAAGAAGGTCGGCGGAGGCGGAGGCGGAGCTGCTGGGGCTTGGCGCTCTGGAAGTCGTCCCAGCCGCGGG ×
TCGCCGAGGAAAGGAGCCTGCGGGTCAGCTTTCTGGCCGAAGTGCCGGCGCGAATTTGTTAGCCGTCTCC ×
GGCCAAAAAGAGCGGCACCTCGGAAGGCGAGTTATTTACCAAGCACTGGAGTAATATTGTAGATAAAAAT ×
GCCTGTTGGATGCAAAGAGAGGCCAACATTTTTTGAAATTTTTAAGACGCGGTGCAATCAAGCAGATTTA ×
GGACCAATAAGCCTTAATTGGTTTGAAGAACTTTCTTTAGAAGCTCCACCCTATAATTCTGAACCCACAG ×
AAGAATCTGGTTATAAAATCAGCTATGAACCAAACCTATTTAAAACACCACAAAGGAAACCTTATAATCA ×
GTTGGCTTCAACTCCAATAGTATTCAGAGAGCCAATATACCAACAATCTCCTTTAAAAGAATTAGATAAA ×
TACAGATTAGATTCAGGAAAGGATATTACCGATAGTAAACATAAAAGTTGTTGCACAATGAAGTCTAAAA ×
TGGATCGAGCAAATGATGTTACCAGCCCACCTCTAAATTCTTATCTTAGTGAAAGTCCTGTTCTACGATG ×
TACACATGTAACACCACAGAGAGAAAAGTCGGTGGTATGTGGAAGCTTATTTCATACACCGAAGCTTATG ×
AAGGGTCAGACACCAAAACGCATTTCTGAAAGTCTAGGAGCTGAGGTGGATTCTGATATGTCTTGGTCAA ×
GTTCTTTAGCCACACCACCAACCCTTAGTTCTACTGTGCTAATAGTCAGAGATGAAGAAGTATCTGCAGC ×
TGTATTTCCTAATGACACTACTGCTATTTTTAAAAGCTGTTTTTCTAACCATGATGAAAGTCTGAAGAAA ×
AATGATCGGTTTATCCCTTGTGGTCCAGGCAAAGAAAACAAAAATCAAAGGGAAGCTAAAAGTCAAAGTT ×
TGGGGAATTCATTTGGTAAAGTAAATAGCACCAAAGACCATTTTGTAAAGTCTACACCAAATGTCCTAGA ×
GGATGAAGTACATGAAAAAGTTCTAGATGTTTCTGAAGAAGAAGATAGTTTTTCATTATGTGTTCCTAAA ×
TATAAAACAAGAAATCTACAAAAAATAAAAACTAGCAAAACTAGGAAAAATATTTTTAATGAGACAAAAA ×
CCAGTGAATGTGAAGAAGCTAAAAAGCAAATGAAAGAAAATAAACATTCATTGGTATCTGAAATGGAACC ×
AAATGACAGTCATCCATTAGATTGGAATGTAACACATGAGAAGCCCTTTGGGAATGGAACTGACAAAATC ×
TCCAAGGAAATTGTACTGTCTTCAGCCTCTGGATGTTCTGACCTAACCCTCTCAAGTCTAAATGGAGCTC ×
```

```
AGATGGAGAAAACACCTCTATTGCATACTTCTTATGACCAAAATAATTCAGAAAAAGACCTCATAATCAC ×
AGATAAAGAATGCACCAACTTCATTACTTTGGAAAATTCTTGGCCACAGATTTCAAATGTACCAAAGTAT ×
TCAGAGAAGACGTTAAATGAGGAAATAGTAGTAAATAAGATAAACGAAGGGCAGTGTCTTGAATCTCATG ×
AAGATTCCGTTGTTTCGGTAAAGCAAGCAATATATGAAACTACTTTAATAGCTTCTCCACTTCAGGGTAT ×
CAGAAAGTCTATATTCAGGATAAGAGAATCACCTGAAGGGATGTCCAATGCAATGTTCTCAAATAATATG ×
ACTAATCCAAACTTTAAAGAACCTGAAGCCTCTGAAAGTGGATTGGAAAAACATACTATTTGCTCTCAGA ×
AAGAGGATTCTTTATGTACAAGTTCAATTGATGATGGAAGCTGGCCAGCAACTATCAAACATACTTCTGT ×
AGCTTTGAAGAATTTAGGTTTAATATCTAGTTTGAAAAAGAAAACAAAAAAGTTTATTTACGTTATAAAT ×
GATGAAACATCTAATCAAGGCCTGAAAACACAGAAAGACCAAGAGTCAAGACTAATTAACCTTTCGACCC ×
AATTTGAAGCAAATGCTTTTGAAGGACCCCTGACATTTACAAATGCTGATTCAGGTTTATTGCATTCTTC ×
TTCCATCAAAAAAAACTGTTTACAGAATGACTCAGAAAAACCAGCTTTGTCTTTAACCAGCTCTTTTGGG ×
ACAATTCTGAGAAAAGTTTCCAGTAATGGAGCCAGTTCTCCTAATAATAAAATAATATCTCAGGATCCTG ×
ATTATAAAGAAGCAAAAATTAATAAGAAAAAATTGGAGTCATTTATAACCACAGAAACTGATTGTCTGTC ×
ATCCCTGCAGGAAAAACATTGGGAAGATGATGCAAAAAAACAAAGAGTTTCAGATATAAAAGAAAAAGTC ×
TTGCCTACAGTAAGTCACCCTCCTGTGCCACATTCAGAAGTGGAAGGTAGTGATATTCACTTTCAGTCTC ×
CAGAAAGCTTTTCATTTGACTGTGATAATACCAGTCTGTTAACTCCTAGCTCTAGGGATTCTCCATCAAG ×
CCTAGTTGTGATGTCTAGAGGAAAAGAATCATATAAAATATCAGAGAAACTAAAATGTAAGAATCATGAA ×
ACTGGTTTTGAATTAACCAAAAATATTCCCATGGAAAAGAATCAAGACATACATGTTTTAAATGCAGATT ×
CTAAAAATGCTAAACTGTTGTCAACTGAAAAACATATAACAGTAGCATCATCTTCAGTAAAGGTTCAGTT ×
CAACCAAAATGCAAATCTCACCACAATCCAAAAAGACCAAAAAGAAACTACTTTAATTTCAAAAATAACT ×
GTTAATCCAAACTCTGAAGAACTTTTCCCAGATGATGAAAATAATTTTGTCTTAAAGATAACTAATGAAA ×
GTAATACTCCTGTTTTAGGAAATACTAAGGAACTACATGATTCAAACCTCTGTTGTGTAAGAGATTCTGT ×
TCCTAAGAACTCTACCATGGTAGTATGTACAGACCTGGATGACAAACAAACAGCCAAAGTGTCGATTATG ×
AAAGATTGTTATTCATCAAGCATAGATGATCTTACAGAAAGGAACAGAAGTACCATAAAGCAACAACTAA ×
AAATGACTCTAGATCAAGATTCAAAATCAGACATTACCTCAGATATAGTTAGGAAATCAAATGGAAACAG ×
TGATTATATGGATAATTGGGCAAGACTGTCTGATCCAATTTCAAATCACAGTTTTGAAAATGGCTTCAAA ×
ACAGCTTCTAATAAAGAGATAAAACTCTCTGAAAACAACATTAGGAAAAGTAAAATGCTTTTCAAAGATA ×
TTGAGGAACATTATCCTACTAACTTAGCATGTCTTGAAATTGTAAATACTTCATCATTAGAAAGTCAAAA ×
GAAACCAAGCAAATCTCATGCACTTGATCCACAGTCAATTAATATCATATCTGGGTTTGTGCAGAATAGC ×
ACATATGTTTCTGATAGTGAAAGTGGTCACACAGCTCCTCCAACTTTATCTTTAAAGCAAGATTTTGATT ×
CAAATCGTAATTTAACTCCTAGTCAAAAGGCAGAAATTACAGAACTTTCTACTATTTTGGAAGAATCAGG ×
AAGCCAGTTTGAATTTACACAGTTTAGAAAACCAAGCCACATAATACAGAAAAATCCATTTGAAATGCCT ×
GAAAACCAGCTGACTATCTTGAATAGCACTTCTAAGGAATGGAAAGATGATGATCTTCATCTCACAACTA ×
ATGCTCCATCTATCAGTCAGGTAGATAGCAAGAAATCTGAAGGTATAATTGGAGGTAAGCAGAAGTTTGC ×
TTGCTTGTCAAGAACCAGCTGTAACAGAAGTGCTTCTGGCTATTCAACAGATAAAAATGAAGTGGAGTTT ×
AGAGGCTTTTATTCTGCTCGTGGCACAAAACTGAATGTTGGTAGTGAAGCATTGCAAAAAGCTAAGAAAC ×
TGTTCAGTGACCTTGAGAATATCAATGAGGAAACTTCTGTAGAAGTAGATAGAAGTTTCTCCTCAAGCAA ×
ATACAATGATTCTGTCTCAATGATTCAGATAGAAGATTGTAATGATAAAAATTTAAATGAGAAAAATAAT ×
AAATGCCGGCTAATACTACAAAATAATATTGAAATGACTACTGACATTTTTGTTGAAGAATATACTGAAA ×
GTTACAGGAGAAATACAGAAAATGAAGGTAACCAATGTACTGACGCTGGTAGAAATACTTGTAACTCAGA ×
ATCTGATGGCAGTGATTCAAGTAAAAATGATACAGTTTATATTCATGAAGAAGAAAATGGCTTGCCCTGT ×
ATTGATCAGCACAACATAGATCTGAAATTATTTAGCCAGTTTATGAAGGAGGGGAACACTCAAATTAAAG ×
AAGGTTTGTCAGATTTAACCTGTTTGGAAGTTATGAAAGCTGAAGAAACATCTCATGTTACTATGTCAAA ×
TAAACAGCAGTTAACAGCTAATACGGGGCAAAACATAAAAGATTTTGACACTTTTTATTTATCCTTTCAG ×
ACTGCAAGCAGAAAAAATATAAGGGTCTCCAAAGAGTCATTAAATAAAGCTAGAAGTCTCCTTAATCAAA ×
AATGGACAGAAGAAGAATTAAATAACTTTTCAGATTCCTTGAATTCTGAATTACTTCCTGGCATAGATAT ×
CAAGAAAACAGACATCTCAAATCATGAGGTAATAGAAAATACTGAAAGAAAAGACAAAATAACGAAAGAA ×
AGTGACCTAATTGGTACTGAAAATATATTACTGATCCTGCAGCAAAGACCAGAAAGTAAAATAAAAAAGA ×
TCAAAGAATCTGCTGTGTTGGGTTTTCATACAGCTAGTGGGAAAAAAATAGAAATTACAAAGGAATCTTT ×
GGACAAAGTAAAAAATCTTTTTGAAGAAAAAGAGCAAGATAATAGTGAAATCACTAATTTTAGCCATCGA ×
GGGGCAAAGATGTCCAAGGACAGAGAAGAATGTAAAGATGGGCGTGAATTAGCTTGTGGGACAACTGAAA ×
```

```
TAACAACTACCCCAGAGTATGAAGAAACTCACAGTTCTCTAGAGAAGAAAAAACTTGTTTCTAATGAGAT ×
TGCAGCCTTAAGACCCAGGCTCTTAAGTGATAATTTATACAAACAAACTGAAAATCTTAAAATATCAGAT ×
CATGCCTCTCAGAAAGTTGATGTACATGAAAATACAGAAAAAGAAACAGCAAAAAAGCCTACAATGTATA ×
CAAATCAATCCACTTATTCAGCCATTGAAAACTCACCTTTAACATTTTACACAGGACACGGAAGAAAAAT ×
TTCTGTGAGTGAGGCTTCACTATTTGAAGCAAAAAAATGGCTTAGAGAAGGAGAATGGGATGATCAATCA ×
GAAAGAATAAATGCTGCCAAGGTTAACTGCTTAAAAGAATATCCTGATGATTACGTAGAAAATCCTTCAT ×
GTGGAAATAGTTCAAATAGTGCCATAACTGAAAATGACAAAAATCATCTCTCTGAAAAACAAGGCTCAAC ×
TTATTTAAGTAATAGTACCATGTCTAACAGCTATTCATACCATCCTGGCTTTTGTCATTCTAGTGAAGTG ×
TATAATAAATCAGAATATCTTTCAAGAAGTAAAATTGATAATTCTGGTATTGAACCAGTAATAAAGAATA ×
TTAGAGAGAGAAAAAACATTGGTTTTTCTGAAATAATGTCCCCTGGAAGAGAAGCAGACACAGACCCACA ×
AAGTGTAAATGAAGATATTTGTGTTGAGAAACTTGCGACTAACTCTTCATGCAAAAATAAAAATACAGCC ×
ATTAAAGTGGCCATATCTGACTCAAATAATTTTAATACAATTCAAAAGTTGAATTCTGATTCAAATAATT ×
CTGTACCTGCATACAGTACAGTAAATAGTAAAAGAGTCTTTGTTGCACACCAGACAAAAGTGACAGAGGG ×
GTTTACAGACAACTGCAGCATGGTAACTAAACAAAACACCAAGAGTAAATCAGACACTTGCCATGCAGAA ×
ATTGTGGCAGATTATCCTAAGGCACTGGATGATTCAGAGGCTATTTTTCCTAACTCTCTGGGTGCTATAG ×
AATGTTCACCTTCACATAAGGTTTTTGCTGACATTCAAAGTGAACAAACTTCACAACTTAACCAAAGTAT ×
GTCTGGATTGGAGAAAGTTTCTGAAACACCACCTTGTCAGATTAATTCAAAAACTTCTGATAGATGTGAA ×
CTTCCTAGGGGGAAGCTTCCCAAGTCAGTCTCTTACACAAATGCATGTGGGATTTTTAGCACAGCAAGTG ×
GAAAATCTGTACAAGTATCAGATGCTGCAATACAAAAGGCAAGAGAGGTGTTTTCTAAGCTAGAAGATAG ×
TGCCAAGCAACTCTTTCCTGAAGTATCACTTAAAGATAATGAAGAACATTCAGAAAAGTTCACAAATGAA ×
GAAAATACTGTGATATATACCTCCCAAAATTTACTATCATCTGCTTTCTCTGGATTTAGGACAGCAAGTG ×
GGAAACAAGTTCCAGTTTCTGAAAGTGCCTTATGCAAAGTTAAGGGAATGTTAGAAGAATTCAATCTGAT ×
CAGAACTGAAAGTTGTCTTCAGCATTCATCTACTTCTAGACAAGATGTATCAAAAATGCCTCCTCCCTCT ×
TGTATTGGTAAGAGAACCCCAGAACACTCCAGAAACTCCAAATTGGATAAAGCCTGCAATAAAGAATTTA ×
GATTATCAAGTAACTGTAACAATCAGAGTGGTTCTTCAGAAAATCATCACTCTATTAAAGTTTCTCCATG ×
TCCCTCTCAATTGAAGCGAGACAAACCACAGTTGCTAGTCGGAAGCAAAGGATCACTTGTTGAGAACATT ×
CATCCTTTGGGAAAAGAACAAGCTTTACCTAAAAATATAAAAACAGAGATTGGGAAAGCTGAAACTTTTC ×
CTAATCTTCCTGTGAAAACAAATATAGAATTTTGTTCTACTTACTCCAAGGATCCAGAAAACTATTTTGA ×
AACAGAAACCGTAGAGATTGCCAAAGCTTTTATGGAAGATGGTGAGCTGACAGATTCCGAACTGCTAAGT ×
CATGCCAAACACTTTGTTTTTACATGCCAAAACACTAAGGAAATGGTTTTGTTAAATTCAAGAATTGGAA ×
AAAGAAGAGGAGATGCACTTGTCTCAGTTGGAGAACCCCCAATTAAAAGAAACTTGTTAAATGAATTCGA ×
CAGGATAATAAAAAATCAAGAAACATCTTTAAAAGCTTCAAAAAGCACTCCAGACGGCATCCTAAAAGAC ×
AGAAGCTTGTTTATGCATCATATTTCTTTAGAGCCAATTTCCTGTGGACCCTTTCGCACAACTGAGGAAC ×
GGCAAGAAATACAGAATCCAAATTTCACTGCACCTGGTCAAGAATTTTTGCCTAAATCTCATTTTTATGA ×
ACACCTGGCTTCAGAAAAATCTTCAAGTAATTTATCAGTTTCACGGCAACCATTTTGTATGGTTCCTGCC ×
ACAGGAAATGAAAAAAGGAGACACTTGATTGCTCCAGGCAAACCAGTGAAAGTCTTTGTCCCACCTTTTA ×
AAACTAAATCACATTTTCACAGAGATGAGCAGTGCATTAGCAAGAATACTAAATTGGAAAAAAACAAACA ×
AAACTCCAAAGACATAGATGAACTTGGCTCTGGTGATAGTGAAAAAAATATTAATGACAGTGGAATCCAT ×
CAGCTTAAGAAAAATAACTCCAATCAAGCAGCAACTATAATATTCACAAAGAATGAAAAAGAACCTTTAG ×
ATTTAATTACAAATCTTCAGAACGCCAGAGATATACAGGATATGCGGATTAAAAAGAAACAAAGGCAGCA ×
TATTTTTCCACAGCCAGGTAGTCTGTATCTTGCAAAAACCTCCACTTTGCCTAGAATCTCTCTGAGAGAA ×
GCAGTAGAAGGCCGAGTCCCCTCTGCATGTTCTCATAAACAGCTCTATATGTATGGTGTTTCCAAACATT ×
GTGTAAAAATAAACAGCAAAAATGCAGAGTCTTTTCAGTTTCATGCTCAGGATTATTTTGGTAAGGAAGG ×
CCTATGGTCTGGAGAAGGAATACAATTGGCTGATGGTGGATGGCTCATACCCTCCAATGATGGAAAGATT ×
GGAAAAGAAGAATTTTATAGGGCTCTGTGTGACACCCCAGGTGTGGATCCAAATTGTATTTCTAGAGTTT ×
GGGTATATAATCACTATAGATGGATTATATGGAAATTGGCAGCCATGGAATTTGCCTTTCCTAAGGAATT ×
TGCTAATAGGTGTCTAAGTCCAGAAAGAGTGCTTCTTCAACTAAAATACAGATATGATGTGGAAATTGAT ×
AAAAGCAGAAGATCAGCTATAAAGAAGATAATGGAAAGGGATGACACAGCTGCAAAAACACTTGTTCTCT ×
GTATTTCTGAAATCATTTCGTCAAGTGCAGATATATCTGAAACTTCTAGTAGTAAAACTAGTAGTGTGGG ×
TACCAAAAAAGTGGGCATTATTGAGCTCACAGATGGGTGGTATGCTATTAAGGCCCAGTTAGACCCTCCC ×
CTCTTAGCTCTCGTAAAGAACGGGAGATTGACTGTGGGTCAGAAGATCACTATTCATGGAGCAGAACTGG ×
```

```
TAGGCTCTCCTGATGCCTGCACACCACTTGAAGCCCCAGAATCTCTTATGTTAAAGATTTCTGCTAACAG ×
TACTCGTCCTGCTTGCTGGTATACCAAACTTGGATTCTCTCCTGATCCTAGACCTTTCCCTCTCCCCTTG ×
TCATCACTTTTCAGTGATGGAGGAAATGTTGGTTGTGTTGATGTAGTTGTTCAAAGAGCATACCCAATAC ×
AGTGGATGGAGAGGACCCCATCTGGATTATGCATATTTCGCAATGAAAGAGAGGAAGAAAAGGAAGCAAC ×
AAAATATGCAGAAATCCAACAAAAGAAACTAGAAGTTTTATTCAATAAAATTCAAGCAGAATTTGAAAAG ×
AATGATGAAAATATAACAAAGCAGTGTATACCATCATGTGCATTAACAAGACAGCAGATCTGTGCTCTGC ×
AAGATGGTGCAGAGCTTTATGAAGCAGTGACAAATGCACCAGACCCAAGTGACCTGGAGGGTTATTTTAG ×
TGAAGAGCAGTTAAGAGCCTTGAATAATCACAGACAGATGTTGAATGATAAGAAGCAAGCACAGATCCAG ×
TTAGAATTCAAGAAGGCTATGGAATCTGCTGAGCAAGGAGAACAAATTCTACCAAGGGATGTTACAACTG ×
TGTGGAAGTTACGTATCATAAGCTACAGGAAAAAAGAAAAAGATTCAGTTACATTGAGTATCTGGCGTCC ×
ATCACCAGATTTATATTCCCTGTTAATAGAAGGAAAGAGATACAGAATCTATCATCTTGCAGCATCACAA ×
TCTAAAAGTAAATCTGGAAAAGCCAACACACAGCTAACAGCAACAAAGAAAACTCAGTACCAGCAACTAC ×
CAGCATCAGATGAAATCCTATCCCAAGTTTATCAGCCAAGGGAACCCCTTTACTTCAACAAACTGTTGGA ×
TCCGGACTTCCAACCACCTTGTTCTGAGGTGGACCTAATAGGATTTGTAGTTTCTGTTGTGAAAAAAATA ×
GGTCTTGCTCCTGTGGTCTATTTGTCAGATGAATGCCATAATTTATTGGCAATAAAGTTCTGGACTGATT ×
TTAATGAAGACATTATTAAACCTTACACATTAATTGCTGCAAGCAACCTCCAGTGGCGACCAGAAGCCAA ×
ATCAGGAATTCCTACTTTATTTGCTGGAGATTTTTCCAGGTTTTCTGCCAGTCCAAAGGAGGAGCATTTT ×
CAAGAGACATTCCACAAAATGAAAAATACTGTTGAGAATATTGGTATGTTTTACAATGATGCAGAAAACA ×
AACTTGTGCATATACTTAATGCAAATGATCCCAAGTTGTCCACCCCGACTAAAGACTATGCTTCAGAGCC ×
ACACACAGCTCAAATAGTCCTTGGCATAGGAAATAAATTTCTGATGTCTTCTCCCAATAATGAGATGAAT ×
TATCAGAGTCCTTTATCACTTTGTAAGCCAAAAGAGAAGTCTGTCCCCATACCTGGATCAACCCAAATGA ×
CTTCAAAGTCTTATTGTAAAGAGGAGAAAGAGATGGATGACCCAAAAACCTGCAAAAAGAGAAGAGCCTT ×
GGACTTTTTGAGTAGAGTGCCTTTACCTCCATCTGTCAGTCCCATTTGTACATTTGTTTCTCCAGCTGCA ×
CAGAAGGCATTTCAGCCACCACGGAGTTGCGGCACCAAATATGAAACACTGATGAAGAAAGAGTTGAATT ×
CTCCACAGATGACTCCACGTAAATTTAATGACCTTTCCCTTTTGGAAAGTGATTCAATAGCAGACGAAGA ×
ACTCGCAATGATAAACACCCAAGCCCTTTTGTTGGGTTCACCAGGAGAACATCAACTTGTGTCTGTCAGT ×
GACTCTACCAGGACTGCTCCCACGAGCTCAAAAGATTATCTTGGACTGAAAAGGCATTCTACTGCACCCG ×
GGGTCAGAGGACCCGAGAGCCCCCAGGCCTGCACCAGGAAGCGGGAGCCCCGTGTACAGAACACAAGTGA ×
TCTGAAAAGGACATCTCTGAGACTGCAGAGGCAACAAACACAAAAATGACAATGAATTGGTGACTGACTC ×
AACCTTTCCAATGTGTGGAAAACACAGCCTCAACCTGTATGTCAAGATGTGCATAATGAGACAAGAAAGA ×
CCACATCCCAAATCTCCTGTGTGCTTGTCTATCTTAGGAAACCTGGCCTATCTCTGTACTGGTCGGTGTA ×
CTTTATTTCAGTTATGTGTCTGAAAATTGTGTATTTATTAGCTAATCAGGAAAAAAAATCTCCTTTAAAC ×
TCTTATGATTGGATATGATCAAGTATATTTTACAAAGTAAACACACTTTTTCTTTAAATTGTGTCCCTAA ×
TTAAATGAAAGTAGGTTTCAAAGTACTGTTATTTTGACTCCTGTAGTTCTTTTTAGGTGACTTGGTTTTG ×
TTTTGTTTTTCGGAGGTAACCTACTATGAACCAGTTTTCCTTAATAAACGTGTTGGTTCTCTTATAGTTG ×
TATCCTGATCAAAAGTCAGGAGGAGTAAGGAACAAACAGCAGTGCTCTCTCTGGACCAGTTCTTTAACCT ×
TACGTCAGCATAAGTGCAAGAAAAACAGAATCCTCAATGTGATTCCTTTTTATGATTCTAGTGTGATTGC ×
TGAATTATTTCAATTAAAAATTCAAATGCTTTTAAAAAAAAAAAAAAAAGAAAAAAAAAA} // ToString;
```

Process

```
SpecialNote = " ";
```

```
Wgenesample = "BRCA2 mRNA Wolf gene"
 (*Lets us know which gene we're dealing with,
used in pdf coding later, so be sure to name it *)
```

```
BRCA2 mRNA Wolf gene
```

```
LetterDNAtoNum[Sample_] := ToExpression[StringReplace[ToString[
    {StringReplace[StringReplace[ToString[{Sample}], {"," → "", " " → "", "{" → "",
        "}" → "", "(" → "", ")" → "", "[" → "", "]" → "", ";" → "", ":" → "", "_" → "",
        "+" → "", "&" → "", "/" → "", "." → "", "RowBox" → "", "Null" → ""}],
      {"0" → "0,", "1" → "1,", "2" → "2,", "3" → "3,", "A" → "0,", "C" → "1,",
       "G" → "2,", "T" → "3,", "a" → "0,", "c" → "1,", "g" → "2,", "t" → "3,"}]}
   ], ",}" → "}"]]
numgenesample = LetterDNAtoNum[lettersample];
Export[StringReplace["GENE_genesample.txt", "GENE_gene" → Wgenesample],
 Flatten[numgenesample]]
```

```
BRCA2 mRNA Wolf genesample.txt
```

```
lengthofgeneitself = Length[Flatten[numgenesample]]
 (*To make sure no base pairs are left out *)
```

```
11 190
```

Construction of W

Can compare to W constructen in Python file **W_hat_construction.py** if we want

```
lettersample = {basepairs} // ToString;
LetterDNAtoNum[Sample_] := ToExpression[StringReplace[ToString[
    {StringReplace[StringReplace[ToString[{Sample}], {"," → "", " " → "", "{" → "",
        "}" → "", "(" → "", ")" → "", "[" → "", "]" → "", ";" → "", ":" → "", "_" → "",
        "+" → "", "&" → "", "/" → "", "." → "", "RowBox" → "", "Null" → "", "
        " → "", "
" → ""}],
      {"0" → "0,", "1" → "1,", "2" → "2,", "3" → "3,", "A" → "0,", "C" → "1,",
       "G" → "2,", "T" → "3,", "a" → "0,", "c" → "1,", "g" → "2,", "t" → "3,"}]}
   ], ",}" → "}"]]

numgenesample = LetterDNAtoNum[lettersample];
lengthofgeneitself = Length[Flatten[numgenesample]];
M = numgenesample;

For[npow = 1, npow < 1000, npow++, If[Length[M] < (2^(npow)), Break[]];
  FilledSize = 2^(npow + 1)];
Filler[vecvar1_] := Table[4, {i, 1, FilledSize - lengthofgeneitself}]
FilledVec[vecvar2_] := Join[Flatten[vecvar2], Filler[vecvar2]]

Filler[vecvar4_] := Table[4, {i, 1, FilledSize - lengthofgeneitself}]
FilledVec[vecvar5_] := Join[Flatten[vecvar5], Filler[vecvar5]]
For[npow = 1, npow < 1000, npow++, If[lengthofgeneitself ≤ (2^npow), Break[]]];
(* gives npow such that  2^npow > lengthofgeneitself > 2^(npow -1) *)
FilledSize = 2^npow;
FilledM = FilledVec[M];
numrowsW = √Length[FilledM] ;
```

```
W = Table[Table[FilledM[[i]],
     {i, (((j - 1) * (numrowsW)) + 1), (j * (numrowsW))}], {j, 1, numrowsW}];
```

```
(*numgenesample=LetterDNAtoNum[lettersample];
lengthofgeneitself=Length[Flatten[numgenesample]];
M=numgenesample;
lengthvec[M_]:=Length[M[[1,All]]]
  For[npow=1,npow<1000,npow++,If[lengthvec[M]<(2^(npow)),Break[]];
   FilledSize=2^(npow+1)];
Filler[M_]:=Table[4,{i,1,FilledSize-lengthvec[M]}]
   FilledVec[M_]:=Join[Flatten[M],Filler[M]]
    lengthvec[M_]:=Length[M[[1,All]]]
     Filler[M_]:=Table[4,{i,1,FilledSize-lengthvec[M]}]
      FilledVec[M_]:=Join[Flatten[M],Filler[M]]
      For[npow=1,npow<1000,npow++,If[lengthvec[M]≤(2^npow),Break[]]];
(* gives npow such that  2^npow > lengthvec[M] > 2^(npow -1) *)
FilledSize=2^npow;
FilledM=FilledVec[M];
numrowsW=√Length[FilledM] ;*)
```

W for Human isolate NA19240 chromosome 9 genomic scaffold

Is a “genomic scaffold” a single gene?

Data

Process

```
SpecialNote =
  " Not sure if this is a single gene or multiple. Also not sure if this is necessarily
    a set of whole genes or if it's just the DNA sequence corresponding
    to a particular chromosomal structure. In that case it might
    includes only parts of some genes (not the whole genes)";
```

```
Wgenesample = "Human chromosome9 scaffold gene"
(*Lets us know which gene we're dealing with,
used in pdf coding later, so be sure to name it *)
```

```
Human chromosome9 scaffold gene
```

```
LetterDNAtoNum[Sample_] := ToExpression[StringReplace[ToString[
    {StringReplace[StringReplace[ToString[{Sample}], {"," → "", " " → "", "{" → "",
        "}" → "", "(" → "", ")" → "", "[" → "", "]" → "", ";" → "", ":" → "", "_" → "",
        "+" → "", "&" → "", "/" → "", "." → "", "RowBox" → "", "Null" → ""}],
      {"0" → "0,", "1" → "1,", "2" → "2,", "3" → "3,", "A" → "0,", "C" → "1,",
       "G" → "2,", "T" → "3,", "a" → "0,", "c" → "1,", "g" → "2,", "t" → "3,"}]}
   ], ",}" → "}"]]
numgenesample = LetterDNAtoNum[lettersample];
Export[StringReplace["GENE_genesample.txt", "GENE_gene" → Wgenesample],
 Flatten[numgenesample]]
```

```
Human chromosome9 scaffold genesample.txt
```

```
lengthofgeneitself = Length[Flatten[numgenesample]]
 (*To make sure no base pairs are left out *)
```

```
59 027
```

Construction of W

Can compare to W constructen in Python file **W_hat_construction.py** if we want

```
lettersample = {basepairs} // ToString;
LetterDNAtoNum[Sample_] := ToExpression[StringReplace[ToString[
    {StringReplace[StringReplace[ToString[{Sample}], {"," → "", " " → "", "{" → "",
        "}" → "", "(" → "", ")" → "", "[" → "", "]" → "", ";" → "", ":" → "", "_" → "",
        "+" → "", "&" → "", "/" → "", "." → "", "RowBox" → "", "Null" → "", "
       " → "", "
" → ""}],
      {"0" → "0,", "1" → "1,", "2" → "2,", "3" → "3,", "A" → "0,", "C" → "1,",
       "G" → "2,", "T" → "3,", "a" → "0,", "c" → "1,", "g" → "2,", "t" → "3,"}]}
   ], ",}" → "}"]]

numgenesample = LetterDNAtoNum[lettersample];
lengthofgeneitself = Length[Flatten[numgenesample]];
M = numgenesample;

For[npow = 1, npow < 1000, npow++, If[Length[M] < (2^(npow)), Break[]];
  FilledSize = 2^(npow + 1)];
Filler[vecvar1_] := Table[4, {i, 1, FilledSize - lengthofgeneitself}]
FilledVec[vecvar2_] := Join[Flatten[vecvar2], Filler[vecvar2]]

Filler[vecvar4_] := Table[4, {i, 1, FilledSize - lengthofgeneitself}]
FilledVec[vecvar5_] := Join[Flatten[vecvar5], Filler[vecvar5]]
For[npow = 1, npow < 1000, npow++, If[lengthofgeneitself ≤ (2^npow), Break[]]];
(* gives npow such that  2^npow > lengthofgeneitself > 2^(npow -1) *)
FilledSize = 2^npow;
FilledM = FilledVec[M];
numrowsW = √Length[FilledM] ;
```

```
W = Table[Table[FilledM[[i]],
    {i, (((j - 1) * (numrowsW)) + 1), (j * (numrowsW))}], {j, 1, numrowsW}];
```

```
(*numgenesample=LetterDNAtoNum[lettersample];
lengthofgeneitself=Length[Flatten[numgenesample]];
M=numgenesample;
lengthvec[M_]:=Length[M[[1,All]]]
  For[npow=1,npow<1000,npow++,If[lengthvec[M]<(2^(npow)),Break[]];
   FilledSize=2^(npow+1)];
Filler[M_]:=Table[4,{i,1,FilledSize-lengthvec[M]}]
   FilledVec[M_]:=Join[Flatten[M],Filler[M]]
    lengthvec[M_]:=Length[M[[1,All]]]
     Filler[M_]:=Table[4,{i,1,FilledSize-lengthvec[M]}]
      FilledVec[M_]:=Join[Flatten[M],Filler[M]]
      For[npow=1,npow<1000,npow++,If[lengthvec[M]≤(2^npow),Break[]]];
(* gives npow such that  2^npow > lengthvec[M] > 2^(npow -1) *)
FilledSize=2^npow;
FilledM=FilledVec[M];
numrowsW=√Length[FilledM] ;*)
```

W for PCV1 samples

W for H1A

Run this for output and pdf

```
Wgenesample
ρ = (W.Transpose[W]); (* ρ as inner product *)
rhoEigens = Sort[DeleteCases[Eigenvalues[ρ] // N, 0.], Greater];
(*DeleteCases Removes 0's from the set of Eigenvalues,
Sort puts the list in order of greatest to least *)
           rhoEigens
set = ────────────────;
       Total[rhoEigens]
(* This is the set of nonzero normalized eigenvalues in order of greatest to least *)
n = Length[set];
           1
H[α_] := ─────  Log[2, Sum[(set[[i]])^α, {i, 1, n}]] // N
         1 - α
H0 = Log[n] // N;(* H_0 = Hartley Entropy*)
H1 = -Sum[((set[[i]]) (Log[2, set[[i]]])), {i, 1, n}] // N;
(* H_1 = Shannon Entropy*)
H2onward = Table[H[a], {a, 2, 20}] // N;(* H_2 onward *)
RenyiEntropyofEigenvalues = Join[{H0}, {H1}, H2onward];
```

```
Human H1A gene
```

```
Null^2
```

```
button =
  Button["Click here for output and pdf", Print[Style[Wgenesample, Black, Bold, 28]]
```

```
Print[Style["The ", Blue, Italic, 18], Style[Wgenesample, Black, Italic, 18],
 Style["  has  ", Blue, Italic, 18], Style[lengthofgeneitself, Black, Italic, 18],
 Style["   base pairs   ", Blue, Italic, 18]] ×
If[StringLength[SpecialNote] > 3, Print[Style["(Special Note): ", Black, Bold, 16],
  Style[SpecialNote, Black, Italic, 12]], Print["  "]] ×
Print[Style["W is a   ", Blue, Italic, 18], Style[Length[W], Black, Italic, 18],
 Style["  by  ", Blue, Italic, 18], Style[Length[W[[1]]], Black, Italic, 18],
 Style["   matrix with   ", Blue, Italic, 18],
 Style[Length[W] * Length[W[[1]]], Black, Italic, 18],
 Style[" = 2^b elements", Blue, Italic, 18], Style["  for b = ", Blue, Italic, 18],
 Style[Log[2, Length[W] * Length[W[[1]]]], Black, Italic, 18]] ×
If[(Length[W] * Length[W[[1]]]) == (Length[W])^2,
 Print[Style["(If statement safecheck):  ", Black, Bold, 12],
  Style[Length[W], Black, Italic, 12], Style["  times  ", Red, Italic, 12],
  Style[Length[W[[1]]], Black, Italic, 12],
  Style["   equals   ", Red, Italic, 12], Style[(Length[W]^2), Black, Italic, 12],
  Style["  𝕎 is of the right size, you may proceed ", Red, Italic, 12]],
 Print[Style["(If statement safecheck):  ", Black, Bold, 12],
  Style["Warning!!!", Red, Italic, 28],
  Style["   𝕎 is of wrong size, STOP and check 𝕎 ", Red, Italic, 12]]] ×
Print["The number of nonzero eigenvalues is  =   ", Length[rhoEigens]] ×
Do[Print["The i-th Eigenvalue "λ_i, " is = ", (rhoEigens)[[i]]],
 {i, 1, Length[rhoEigens]}] ×
Print[Graphics[ListPlot[rhoEigens // N, AxesLabel → {Style["i", Medium, Bold],
     Style["λ_i", Medium, Bold]}, PlotLabel → "Eigenvalue PLOT"]]] ×
Print[Graphics[ListLogPlot[rhoEigens // N, AxesLabel → {Style["i", Medium, Bold],
     Style["Log[λ_i]", Medium, Bold]}, PlotLabel → "Eigenvalue Log PLOT"]]] ×
Print["Zooming in on the Log Plot so as to Exclude the first
   eigenvalue gives the following plot:"] ×
Print[Graphics[ListLogPlot[Table[{i, rhoEigens[[i]]}, {i, 2, Length[rhoEigens]}],
   AxesLabel → {Style["i", Medium, Bold], Style["Log[λ_i]", Medium, Bold]},
   PlotRange → {{10, 2 * rhoEigens[[2]]}}, PlotStyle → Red,
   PlotLabel → Style["Logplot of Eigenvalues, excluding λ_1", Red, Bold, 16]] ]] ×
Print["The approximate linearity of the above plot tells us
   that the eigenvalues decrease exponentially. If it's
   nowhere near linear try adjusting the plot range. "]
×
Print["  "] ×
Print[Style[
  "██████████████████████████████████████████████████████████████████", 18]] ×
Print["  "]
×
Print[Style["The First normalized eigenvector is:   ", Blue, Italic, 18],
 Style[set[[1]], Blue, Italic, 18]] ×
Print[Style["The Second normalized eigenvector is:   ", Blue, Italic, 18],
 Style[set[[2]], Blue, Italic, 18]] ×
Print[Style["The Last (n-th) normalized (nonzero) eigenvector is:   ",
  Blue, Italic, 18], Style[set[[n]], Blue, Italic, 16]] ×
If[Total[set] == 1, Print[Style["(If statement safecheck):  ", Black, Bold, 12],
  Style["Total[set] = ", Red, Italic, 12], Style[Total[set], Black, Italic, 12],
      Style[" = 1, so the Eigenvalue set is properly normalized", Red, Italic, 12]],
     Print[Style["(If statement safecheck):  ", Black, Bold, 12],
      Style["Warning!!!", Red, Italic, 28], Style["  Total[set] = ", Red, Italic, 12],
      Style[Total[set], Black, Italic, 12], Style[" ≠ 1, ", Red, Italic, 12],
      Style[" so the Eigenvalue set is NOT properly normalized.", Red, Italic, 12],
      Style[" This will render the entropies invalid. Fix it. ", Red, Italic, 12]]]
    ×
    Print["  "] ×
    Print[Style[
      "██████████████████████████████████████████████████████████████", 18]] ×
    Print["  "]
    ×
    Do[Print["The α-th Renyi Entropy Hα ->  "Hi-1, " is = ",
      RenyiEntropyofEigenvalues[[i]]], {i, 1, Length[RenyiEntropyofEigenvalues]}] ×
    Print[Graphics[Show[
       ListPlot[RenyiEntropyofEigenvalues, PlotRange → All,
        AxesLabel → {Style["α", Large, Bold], Style["Hα", Large, Bold]}],
       ListLinePlot[RenyiEntropyofEigenvalues, PlotStyle → {Red, Thin}]
      ]]] ×
    Export["rhoEigenEntropies.pdf", EvaluationNotebook[]] ×
    NotebookSave[EvaluationNotebook[], "rhoCalcOutput"];
   SystemOpen["rhoEigenEntropies.pdf"]
   (*DeleteFile[StringReplace["rhoEigenEntropies.pdf","rho"→ Wgenesample]]
    RenameFile["rhoEigenEntropies.pdf",
     StringReplace["rhoEigenEntropies.pdf","rho"→ Wgenesample]]
    SystemOpen[StringReplace["rhoEigenEntropies.pdf","rho"→ Wgenesample]]*)
   , Background → Green];
nb = CreateDocument[];
Paste[nb, button]
NotebookEvaluate[nb];
```

Possible Issues

Be sure that W contains only integer values (e.g. 0, 1, 2, 3, 4).
Including floats (e.g. 0.0, 1.0, 2.0, 3.0, 4.0) may incorrectly make some of the eigenvalues negative or imaginary because of how mathematica handles floats differntly than it handles integers.

H1A Renyi Entropy Calc

General Definitional Terms (let H1A = M)

Define a 1x781 vector H1Avec which contains all the base pair info of the H1A gene

```
H1Avec =
  {{3, 0, 2, 2, 1, 3, 2, 1, 2, 3, 3, 2, 2, 2, 2, 1, 1, 3, 3, 3, 3, 3, 3, 3, 1, 2, 1, 0, 3, 1, 1, 3, 2,
    1, 3, 3, 1, 2, 3, 1, 0, 2, 2, 3, 3, 3, 0, 3, 0, 1, 1, 0, 1, 3, 3, 3, 0, 3, 3, 3, 2, 2, 3, 2, 3,
    2, 1, 3, 2, 3, 2, 3, 3, 0, 2, 3, 1, 0, 1, 1, 0, 3, 2, 3, 1, 3, 2, 0, 0, 0, 1, 0, 2, 3, 2, 1, 1,
    3, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 3, 3, 1, 3, 2, 1, 3, 2, 1, 3, 1, 1, 3, 2, 0,
    2, 0, 0, 0, 1, 1, 3, 3, 3, 0, 2, 1, 3, 2, 2, 1, 0, 0, 2, 0, 0, 2, 2, 1, 0, 0, 0, 2, 0, 0,
    0, 1, 1, 3, 2, 1, 3, 0, 0, 2, 2, 1, 3, 2, 1, 0, 2, 1, 0, 2, 1, 1, 3, 1, 1, 0, 0, 2, 0, 0,
    0, 0, 0, 0, 1, 1, 1, 2, 1, 3, 2, 2, 1, 1, 1, 3, 3, 1, 1, 2, 3, 2, 3, 1, 0, 2, 0, 2, 1, 3,
    2, 0, 3, 1, 2, 3, 2, 1, 0, 2, 2, 1, 3, 2, 1, 3, 3, 1, 1, 3, 1, 1, 3, 1, 3, 0, 0, 2, 2, 0,
    2, 1, 2, 3, 2, 2, 3, 2, 2, 3, 2, 3, 2, 3, 1, 2, 3, 3, 2, 2, 1, 0, 2, 1, 3, 1, 3, 3, 0, 0,
    0, 0, 0, 2, 2, 1, 2, 1, 3, 2, 2, 1, 2, 2, 1, 1, 2, 1, 0, 2, 2, 1, 3, 0, 1, 2, 0, 1, 2, 3,
    2, 2, 0, 2, 0, 0, 2, 0, 0, 1, 0, 0, 1, 0, 2, 1, 1, 2, 1, 0, 3, 3, 0, 0, 2, 1, 3, 2, 2, 2,
    1, 0, 3, 3, 0, 0, 2, 0, 2, 1, 1, 3, 2, 2, 3, 0, 0, 2, 1, 0, 0, 2, 2, 2, 0, 0, 1, 2, 3, 3,
    2, 2, 3, 2, 1, 0, 2, 0, 1, 0, 0, 0, 2, 2, 2, 3, 0, 1, 1, 2, 2, 0, 2, 1, 1, 3, 1, 2, 2, 2,
    3, 3, 1, 1, 3, 3, 1, 0, 0, 2, 1, 3, 1, 0, 0, 1, 0, 0, 2, 0, 0, 2, 2, 1, 2, 3, 1, 1, 3, 1,
    1, 2, 3, 2, 2, 0, 0, 0, 1, 1, 0, 0, 2, 1, 1, 1, 2, 2, 1, 2, 1, 1, 3, 1, 0, 0, 0, 2, 2, 3,
    2, 2, 1, 3, 0, 1, 0, 0, 0, 0, 0, 1, 3, 0, 0, 2, 2, 1, 0, 0, 1, 2, 2, 2, 3, 2, 1, 0, 3, 1,
    3, 0, 0, 0, 0, 0, 2, 1, 3, 1, 0, 0, 0, 0, 0, 2, 2, 1, 1, 0, 1, 2, 2, 2, 2, 2, 1, 3, 0, 2,
    1, 0, 0, 0, 0, 0, 2, 0, 2, 1, 2, 3, 1, 0, 0, 2, 0, 1, 3, 1, 1, 2, 0, 0, 0, 0, 0, 2, 2, 1,
    3, 0, 0, 0, 0, 0, 2, 1, 1, 3, 2, 1, 2, 2, 1, 0, 0, 1, 0, 0, 2, 2, 0, 0, 0, 3, 1, 1, 3, 1,
    1, 0, 0, 2, 0, 0, 3, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 3, 2, 3, 0, 0, 0,
    2, 1, 1, 1, 0, 0, 2, 0, 0, 0, 2, 3, 0, 2, 1, 3, 0, 0, 0, 0, 2, 1, 1, 1, 3, 2, 1, 3, 0,
    0, 0, 2, 1, 3, 0, 0, 2, 2, 1, 3, 2, 3, 0, 0, 0, 0, 1, 1, 1, 0, 0, 2, 2, 1, 2, 2, 1, 1,
    0, 0, 2, 2, 1, 3, 0, 2, 2, 2, 3, 2, 0, 1, 2, 0, 0, 2, 1, 1, 0, 0, 0, 2, 0, 1, 3, 2, 1,
    1, 0, 0, 0, 1, 1, 1, 0, 0, 2, 0, 0, 0, 2, 1, 2, 2, 1, 0, 1, 1, 1, 0, 0, 2, 0, 0, 0, 0,
    0, 2, 3, 0, 0, 0, 3, 3, 1, 0, 2, 3, 3, 0, 2, 0, 0, 2, 3, 3, 3, 1, 3, 3, 1, 3, 0, 2, 3,
    0, 0, 1, 1, 1, 0, 0, 1, 2, 2, 1, 3, 1, 3, 3, 3, 3, 0, 0, 2, 0, 2, 1, 1, 0, 1, 1, 3, 0}};
```

```
M = H1Avec;
```

```
lengthvec[M_] := Length[M[[1, All]]]
For[npow = 1, npow < 1000, npow++, If[lengthvec[M] < (2^(npow)), Break[]];
  FilledSize = 2^(npow + 1)];
Filler[M_] := Table[4, {i, 1, FilledSize - lengthvec[M]}]
FilledVec[M_] := Join[Flatten[M], Filler[M]]
```

```
lengthvec[M_] := Length[M[[1, All]]]
Filler[M_] := Table[4, {i, 1, FilledSize - lengthvec[M]}]
FilledVec[M_] := Join[Flatten[M], Filler[M]]
```

```
For[npow = 1, npow < 1000, npow++, If[lengthvec[M] ≤ (2^npow), Break[]]];
(* gives npow such that  2^npow > lengthvec[M] > 2^(npow -1) *)
FilledSize = 2^npow;
```

H1A Renyi Entropies

```
Length[M[[1]]] (*Should be 781 *)
Length[FilledVec[M]](*Should be 1024 *)
```

```
781
```

```
1024
```

```
set = M[[1]]; (* Define the set here,
then the following functions are defined using this set *)
```

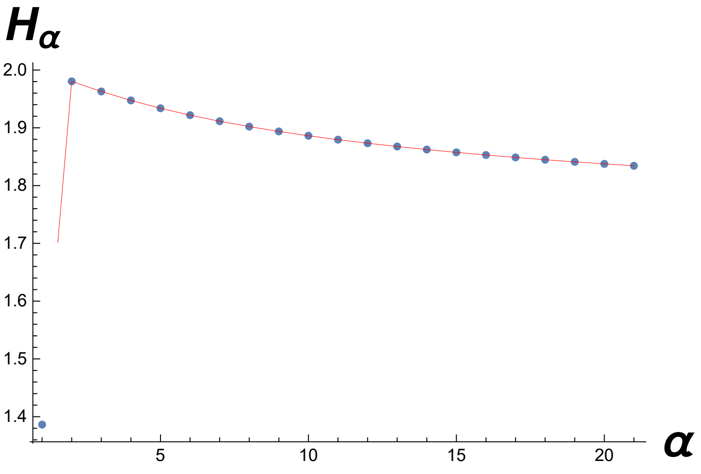
```
set = FilledVec[M]; (* Define the set here,
then the following functions are defined using this set *)
```

```
n = Length[Union[set]] (* i.e. n = number of unique elements in the set *)
prob[i_] := Count[set, Union[set][[i]]] / Length[set] // N
H[α_] := 1 / (1 - α) Log[2, Sum[(prob[i])^α, {i, 1, n}]] // N
```

```
4
```

```
(* Gives a plot of the Renyi Entropies for α ≥ 0, RUN BLUE CELL FIRST *)
H0 = Log[n] // N; (* H0 = Hartley Entropy*)
H1 = -Sum[((prob[i]) (Log[2, prob[i]])), {i, 1, n}]; (* H1 = Shannon Entropy*)
H2onward = Table[H[a], {a, 2, 20}] // N; (* H2 onward *)
RenyiEntropyofH1AData = Join[{H0}, {H1}, H2onward]
Show[
 ListPlot[RenyiEntropyofH1AData, PlotRange → All,
  AxesLabel → {Style["α", Large, Bold], Style["Hα", Large, Bold]}],
 ListLinePlot[RenyiEntropyofH1AData, PlotStyle → {Red, Thin}]
]
```

```
{1.38629, 1.98039, 1.96288, 1.94741, 1.93382, 1.92189,
 1.91138, 1.90206, 1.89375, 1.88627, 1.87949, 1.8733, 1.86762, 1.86237,
 1.85749, 1.85296, 1.84872, 1.84475, 1.84102, 1.83752, 1.83423}
```



```
Union[set] (* prob[i] measures prob of obtaining Union[set][[i]] *)
prob[1]
prob[2]
prob[3]
prob[4]
```

```
{0, 1, 2, 3}

0.297055

0.261204

0.256082

0.185659
```

```
Table[{"The", i - 1, "-th Renyi Entropy is =", RenyiEntropyofH1AData[[i]]}, {i, 1, 21}]
```

```
{{The, 0, -th Renyi Entropy is =, 1.38629},
 {The, 1, -th Renyi Entropy is =, 1.98039}, {The, 2, -th Renyi Entropy is =, 1.96288},
 {The, 3, -th Renyi Entropy is =, 1.94741}, {The, 4, -th Renyi Entropy is =, 1.93382},
 {The, 5, -th Renyi Entropy is =, 1.92189}, {The, 6, -th Renyi Entropy is =, 1.91138},
 {The, 7, -th Renyi Entropy is =, 1.90206}, {The, 8, -th Renyi Entropy is =, 1.89375},
 {The, 9, -th Renyi Entropy is =, 1.88627}, {The, 10, -th Renyi Entropy is =, 1.87949},
 {The, 11, -th Renyi Entropy is =, 1.8733}, {The, 12, -th Renyi Entropy is =, 1.86762},
 {The, 13, -th Renyi Entropy is =, 1.86237}, {The, 14, -th Renyi Entropy is =, 1.85749},
 {The, 15, -th Renyi Entropy is =, 1.85296}, {The, 16, -th Renyi Entropy is =, 1.84872},
 {The, 17, -th Renyi Entropy is =, 1.84475}, {The, 18, -th Renyi Entropy is =, 1.84102},
 {The, 19, -th Renyi Entropy is =, 1.83752}, {The, 20, -th Renyi Entropy is =, 1.83423}}
```