Project 3

Calvin Hoang

3/17/2023

Heart Disease Dataset

Divide the entire Kaggle version of BRFSS dataset with respect to categorical variable GenHealth into 5 sub-datasets. Again the focal measurement is BMI.

1/ First Sub-dataset

```
##    HeartDiseaseorAttack HighBP HighChol CholCheck BMI Smoker Stroke Diabetes
## 24                    0      1        0         1  27      0      0        2
## 30                    0      0        1         1  31      1      0        0
## 32                    0      1        0         1  33      1      0        0
## 33                    0      0        0         1  23      0      0        0
## 39                    0      0        0         1  26      1      0        0
## 56                    0      0        0         1  29      0      0        0
## 58                    0      0        1         1  24      1      0        0
## 70                    1      1        1         1  23      1      1        2
## 74                    0      0        0         0  23      1      0        0
## 84                    1      0        1         1  32      0      0        2
##    PhysActivity Fruits Veggies HvyAlcoholConsump AnyHealthcare NoDocbcCost
## 24            1      1       1                 0             1           0
## 30            1      1       1                 0             1           0
## 32            1      1       1                 0             0           0
## 33            1      1       1                 0             1           0
## 39            1      1       1                 0             0           0
## 56            1      0       1                 0             1           0
## 58            1      1       1                 0             1           0
## 70            0      1       0                 0             1           1
## 74            0      0       1                 0             1           0
## 84            1      0       0                 0             1           0
##    GenHlth MentHlth PhysHlth DiffWalk Sex Age Education Income
## 24       1        0        0        0   0  13         5      4
## 30       1        0        0        0   1  12         6      8
## 32       1        0        0        1   1  13         3      3
## 33       1        2        0        0   0   6         4      8
## 39       1        0        1        0   1   4         5      3
## 56       1        0       10        0   1  11         6      8
## 58       1        0        0        0   0  10         6      8
## 70       1        2        0        0   0   7         5      3
## 74       1        0        0        0   0   7         5      7
## 84       1        0        0        1   0  13         2      2
```

2/ Second Sub-dataset

```
##    HeartDiseaseorAttack HighBP HighChol CholCheck BMI Smoker Stroke Diabetes
## 4                     0      1        0         1  27      0      0        0
## 5                     0      1        1         1  24      0      0        0
## 6                     0      1        1         1  25      1      0        0
## 10                    0      0        0         1  24      0      0        0
## 16                    0      1        0         1  33      0      0        0
## 18                    0      0        0         1  23      1      0        2
## 19                    0      0        0         0  23      0      0        0
## 20                    0      0        1         1  28      0      0        0
## 26                    0      0        0         1  32      0      0        0
## 35                    0      1        1         1  24      1      0        2
##    PhysActivity Fruits Veggies HvyAlcoholConsump AnyHealthcare NoDocbcCost
## 4             1      1       1                 0             1           0
## 5             1      1       1                 0             1           0
## 6             1      1       1                 0             1           0
## 10            0      0       1                 0             1           0
## 16            1      0       0                 0             1           0
## 18            1      0       0                 0             1           0
## 19            0      0       1                 0             1           0
## 20            0      0       0                 1             1           0
## 26            1      1       1                 0             1           0
## 35            0      0       0                 0             1           0
##    GenHlth MentHlth PhysHlth DiffWalk Sex Age Education Income
## 4        2        0        0        0   0  11         3      6
## 5        2        3        0        0   0  11         5      4
## 6        2        0        2        0   1  10         6      8
## 10       2        0        0        0   1   8         4      3
## 16       2        5        0        0   0   6         6      8
## 18       2        0        0        0   1   7         5      6
## 19       2       15        0        0   0   2         6      7
## 20       2       10        0        0   1   4         6      8
## 26       2        0        0        0   0   5         6      8
## 35       2        0        0        0   0  12         3      3
```

3/ Third Subset

```
##    HeartDiseaseorAttack HighBP HighChol CholCheck BMI Smoker Stroke Diabetes
## 2                     0      0        0         0  25      1      0        0
## 7                     0      1        0         1  30      1      0        0
## 8                     0      1        1         1  25      1      0        0
## 11                    0      0        0         1  25      1      0        2
## 12                    0      1        1         1  34      1      0        0
## 13                    0      0        0         1  26      1      0        0
## 17                    0      1        1         1  21      0      0        0
## 21                    1      1        1         1  22      0      1        0
## 23                    0      0        0         1  28      1      0        0
## 25                    0      1        1         1  28      1      0        0
##    PhysActivity Fruits Veggies HvyAlcoholConsump AnyHealthcare NoDocbcCost
## 2             1      0       0                 0             0           1
## 7             0      0       0                 0             1           0
## 8             1      0       1                 0             1           0
## 11            1      1       1                 0             1           0
## 12            0      1       1                 0             1           0
## 13            0      0       1                 0             1           0
## 17            1      1       1                 0             1           0
## 21            0      1       0                 0             1           0
## 23            0      0       1                 0             1           0
## 25            0      1       1                 0             1           0
##    GenHlth MentHlth PhysHlth DiffWalk Sex Age Education Income
## 2        3        0        0        0   0   7         6      1
## 7        3        0       14        0   0   9         6      7
## 8        3        0        0        1   0  11         4      4
## 11       3        0        0        0   1  13         6      8
## 12       3        0       30        1   0  10         5      1
## 13       3        0       15        0   0   7         5      7
## 17       3        0        0        0   0  10         4      3
## 21       3       30        0        1   0  12         4      4
## 23       3        0        7        0   1   5         5      5
## 25       3        6        0        1   0   9         4      6
```

4/ Fourth Sub-dataset

```
##    HeartDiseaseorAttack HighBP HighChol CholCheck BMI Smoker Stroke Diabetes
## 14                    0      1        1         1  28      0      0        2
## 15                    0      0        1         1  33      1      1        0
## 28                    1      1        1         1  28      1      0        2
## 29                    0      1        1         1  27      1      0        2
## 31                    0      1        1         1  34      1      1        2
## 43                    0      0        0         1  28      1      1        0
## 51                    1      1        1         1  30      1      0        0
## 62                    0      1        0         1  27      1      0        0
## 65                    0      1        1         1  27      0      0        1
## 66                    1      1        1         1  38      1      1        0
##    PhysActivity Fruits Veggies HvyAlcoholConsump AnyHealthcare NoDocbcCost
## 14            0      0       1                 0             1           0
## 15            1      0       1                 0             1           1
## 28            0      0       1                 0             1           0
## 29            0      1       1                 0             1           0
## 31            1      0       0                 0             1           0
## 43            0      1       1                 0             1           1
## 51            1      0       0                 0             1           0
## 62            0      0       0                 0             1           0
## 65            1      1       1                 0             1           1
## 66            0      1       1                 0             1           0
##    GenHlth MentHlth PhysHlth DiffWalk Sex Age Education Income
## 14       4        0        0        1   0  11         4      6
## 15       4       30       28        0   0   4         6      2
## 28       4        0        0        0   1  12         2      4
## 29       4       20       20        1   0   8         4      7
## 31       4        0        7        1   0   9         5      4
## 43       4       15       30        1   0   7         4      3
## 51       4       10       17        1   0   9         4      1
## 62       4        0        5        1   0  11         4      2
## 65       4       30       30        1   0  10         4      3
## 66       4       10        5        1   1  11         5      6
```

5/ Fifth Sub-dataset

```
##     HeartDiseaseorAttack HighBP HighChol CholCheck BMI Smoker Stroke Diabetes
## 1                      0      1        1         1  40      1      0        0
## 3                      0      1        1         1  28      0      0        0
## 9                      1      1        1         1  30      1      0        2
## 22                     0      1        1         1  38      1      0        0
## 27                     1      1        1         1  37      1      1        2
## 40                     0      1        1         1  24      1      0        0
## 53                     0      1        1         1  27      1      0        2
## 96                     1      1        1         1  25      1      0        2
## 110                    0      1        1         1  33      0      0        0
## 112                    0      1        1         1  26      0      0        0
##     PhysActivity Fruits Veggies HvyAlcoholConsump AnyHealthcare NoDocbcCost
## 1              0      0       1                 0             1           0
## 3              0      1       0                 0             1           1
## 9              0      1       1                 0             1           0
## 22             0      1       1                 0             1           0
## 27             0      0       1                 0             1           0
## 40             0      1       1                 0             1           0
## 53             0      0       1                 0             1           0
## 96             0      1       1                 0             1           0
## 110            1      0       1                 0             1           0
## 112            0      1       1                 0             1           0
##     GenHlth MentHlth PhysHlth DiffWalk Sex Age Education Income
## 1         5       18       15        1   0   9         4      3
## 3         5       30       30        1   0   9         4      8
## 9         5       30       30        1   0   9         5      1
## 22        5       15       30        1   0  13         2      3
## 27        5        0        0        1   1  10         6      5
## 40        5        0       30        0   1   9         3      1
## 53        5        0       30        1   0  10         4      5
## 96        5       15       30        1   0   9         2      3
## 110       5        0       15        1   0   7         5      3
## 112       5        0       20        1   0  13         6      4
```

Within each sub-dataset, investigate the 3-way or 4-way or 5-way interacting effects on BMI distribution shapes pertaining to 3, 4 or 5 binary categorical variables of your choices. Apply HC-approach and Entropy approach.

1/ Investigating the 3-way interacting effects in GenHealth_1 sub-dataset:

Creating contigency tables based on the combination of these three selected binary variable

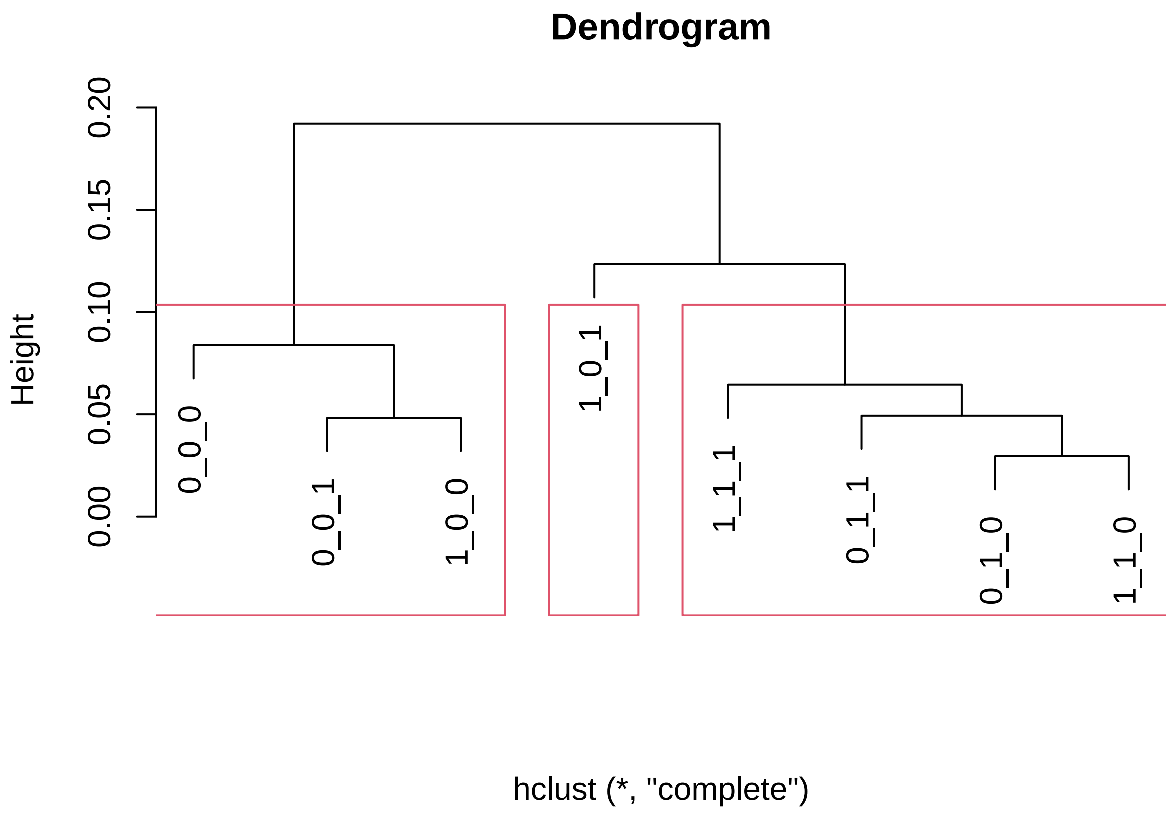
```
## Warning: package 'iNZightTools' was built under R version 4.2.2
```

```
##
## Attaching package: 'iNZightTools'
```

```
## The following object is masked from 'package:stats':
```

```
##
##     filter

##                 1         2         3           4           5            6
## 0_1_0 0.07974138 0.6635776 0.2178879 0.032543103 0.003448276 0.0006465517
## 0_0_1 0.11757526 0.7255221 0.1427990 0.010035259 0.001898563 0.0000000000
## 0_0_0 0.18499047 0.6833322 0.1164703 0.011043681 0.001305483 0.0001411333
## 1_1_1 0.05513784 0.6365915 0.2406015 0.060150376 0.007518797 0.0000000000
## 1_0_1 0.06010929 0.7377049 0.1857923 0.016393443 0.000000000 0.0000000000
## 1_0_0 0.13147410 0.6892430 0.1713147 0.007968127 0.000000000 0.0000000000
## 0_1_1 0.06213394 0.6740514 0.2378406 0.018589254 0.005092946 0.0005092946
## 1_1_0 0.07103825 0.6775956 0.1967213 0.043715847 0.005464481 0.0000000000
## Total 0.15017992 0.6871896 0.1436676 0.014304952 0.002008874 0.0001986799
##                   7            8            9           10
## 0_1_0 0.000000e+00 0.0008620690 0.0004310345 0.0008620690
## 0_0_1 0.000000e+00 0.0006780580 0.0005424464 0.0009492813
## 0_0_0 1.411333e-04 0.0010937831 0.0008467998 0.0006350999
## 1_1_1 0.000000e+00 0.0000000000 0.0000000000 0.0000000000
## 1_0_1 0.000000e+00 0.0000000000 0.0000000000 0.0000000000
## 1_0_0 0.000000e+00 0.0000000000 0.0000000000 0.0000000000
## 0_1_1 0.000000e+00 0.0010185893 0.0002546473 0.0005092946
## 1_1_0 0.000000e+00 0.0054644809 0.0000000000 0.0000000000
## Total 8.830217e-05 0.0009933994 0.0006843418 0.0006843418

##                 1         2         3          4           5            6
## 0_1   0.07167036 0.6683787 0.2270340 0.02614684 0.004202171 0.0005836349
## 0_0   0.17107179 0.6920428 0.1219061 0.01083548 0.001427931 0.0001119946
## 1_1   0.06013746 0.6494845 0.2268041 0.05498282 0.006872852 0.0000000000
## 1_0   0.10138249 0.7096774 0.1774194 0.01152074 0.000000000 0.0000000000
## Total 0.15017992 0.6871896 0.1436676 0.01430495 0.002008874 0.0001986799
##                   7            8            9           10
## 0_1   0.000000e+00 0.0009338158 0.0003501809 0.0007003619
## 0_0   1.119946e-04 0.0010079516 0.0007839624 0.0006999664
## 1_1   0.000000e+00 0.0017182131 0.0000000000 0.0000000000
## 1_0   0.000000e+00 0.0000000000 0.0000000000 0.0000000000
## Total 8.830217e-05 0.0009933994 0.0006843418 0.0006843418

##                 1         2         3          4           5            6
## 0_0   0.17018374 0.6805530 0.1307380 0.01406828 0.001606937 0.0002122370
## 0_1   0.09830988 0.7076365 0.1758251 0.01300770 0.003008583 0.0001769755
## 1_1   0.05670103 0.6683849 0.2233677 0.04639175 0.005154639 0.0000000000
## 1_0   0.10599078 0.6843318 0.1820276 0.02304147 0.002304147 0.0000000000
## Total 0.15017992 0.6871896 0.1436676 0.01430495 0.002008874 0.0001986799
##                   7            8            9           10
## 0_0   1.212783e-04 0.0010611849 0.0007883088 0.0006670305
## 0_1   0.000000e+00 0.0007963897 0.0004424387 0.0007963897
## 1_1   0.000000e+00 0.0000000000 0.0000000000 0.0000000000
## 1_0   0.000000e+00 0.0023041475 0.0000000000 0.0000000000
## Total 8.830217e-05 0.0009933994 0.0006843418 0.0006843418

##                 1         2         3          4           5            6
## 1_0   0.07941115 0.6641095 0.2170848 0.03296703 0.003524777 0.0006220195
## 0_1   0.11618367 0.7258171 0.1438401 0.01018923 0.001852587 0.0000000000
## 0_0   0.18452069 0.6833840 0.1169517 0.01101668 0.001294023 0.0001398944
## 1_1   0.06148867 0.6705964 0.2380952 0.02242256 0.005316690 0.0004623209
## Total 0.15017992 0.6871896 0.1436676 0.01430495 0.002008874 0.0001986799
##                  7            8            9           10
## 1_0   0.000000e+00 0.0010366991 0.0004146797 0.0008293593
## 0_1   0.000000e+00 0.0006616382 0.0005293106 0.0009262935
## 0_0   1.398944e-04 0.0010841814 0.0008393663 0.0006295247
## 1_1   0.000000e+00 0.0009246417 0.0002311604 0.0004623209
## Total 8.830217e-05 0.0009933994 0.0006843418 0.0006843418
```

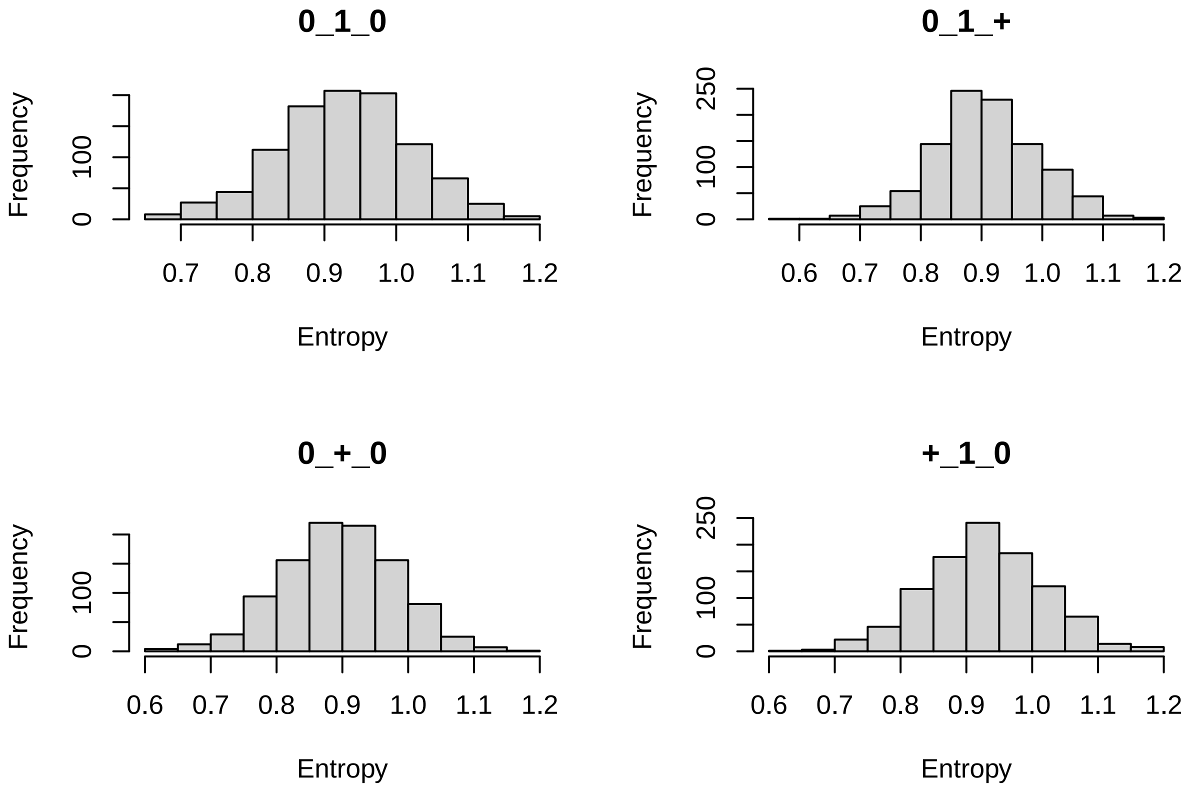
3-way HC-tree of GenHealth_1:



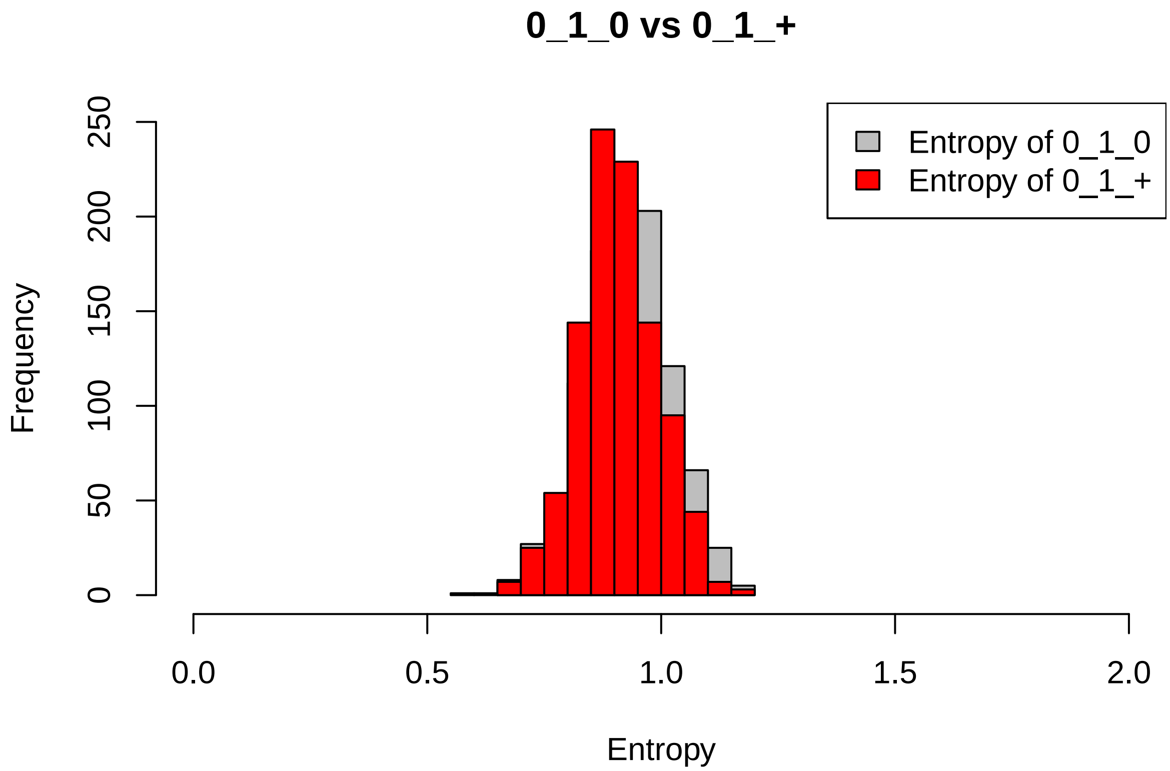
Comparing the entropies of categorical variables to see the effects

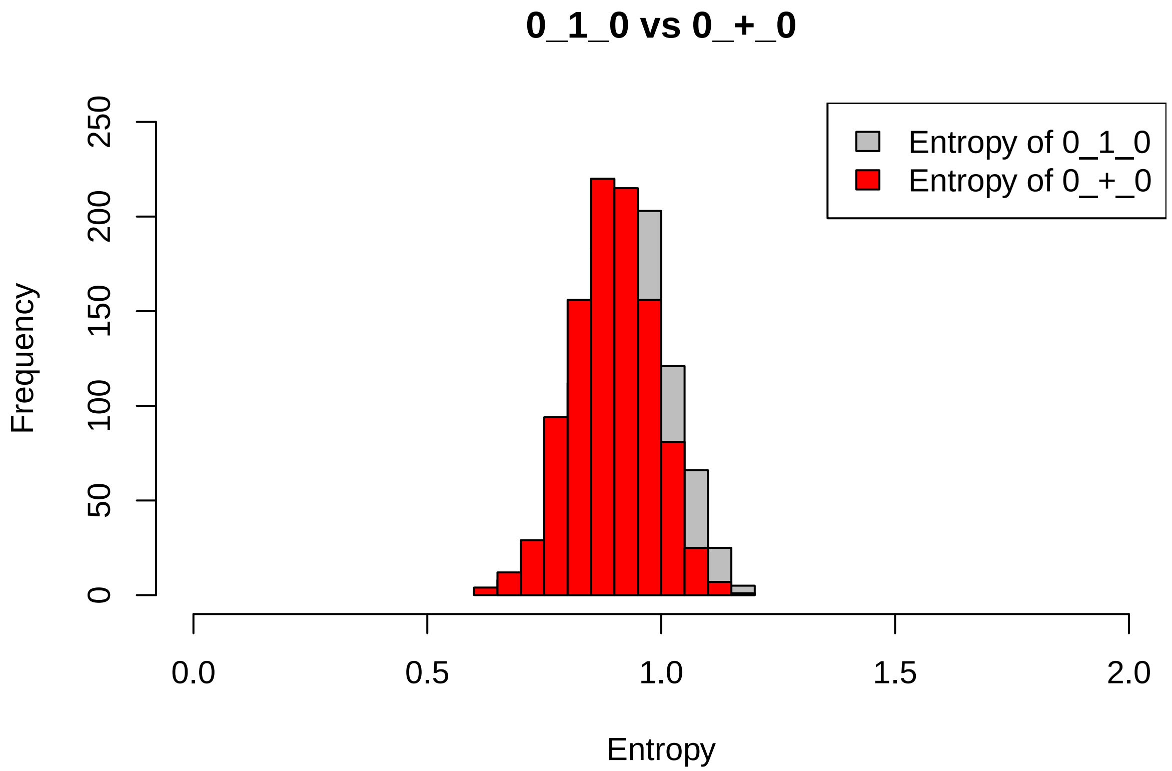
No HD __ Yes HighBP __ No HighChol

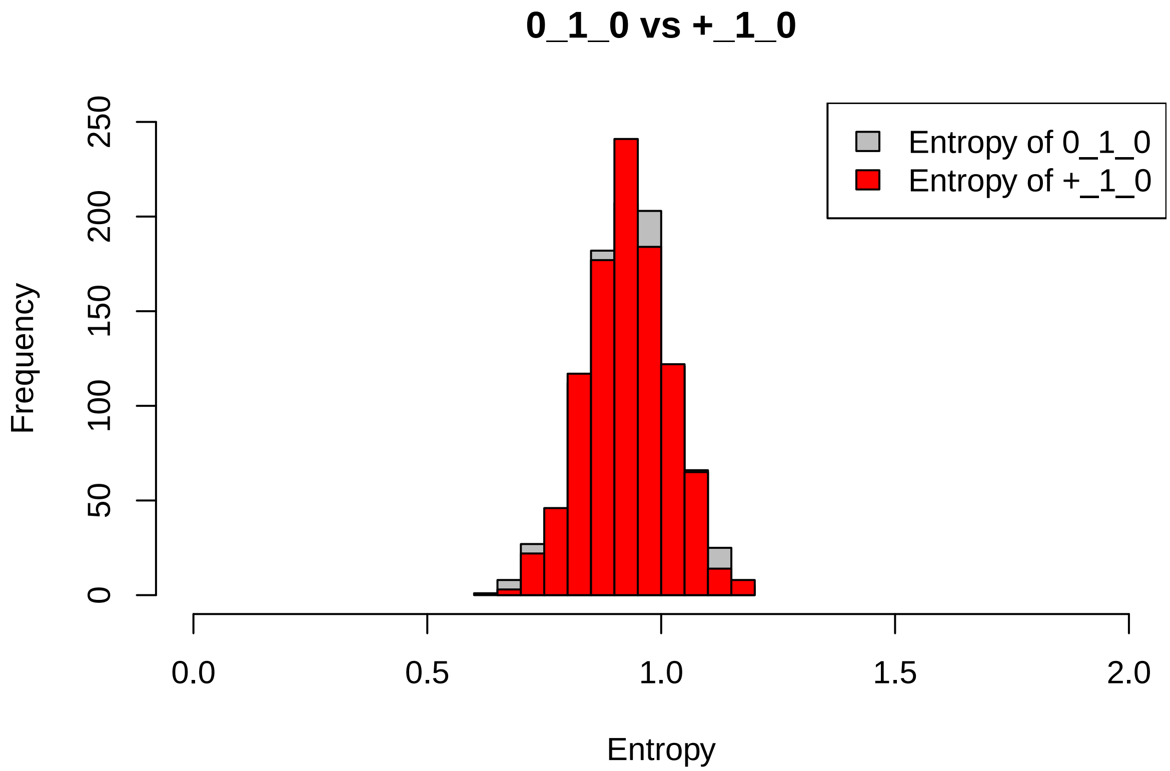
```
## Warning: package 'DescTools' was built under R version 4.2.2
```



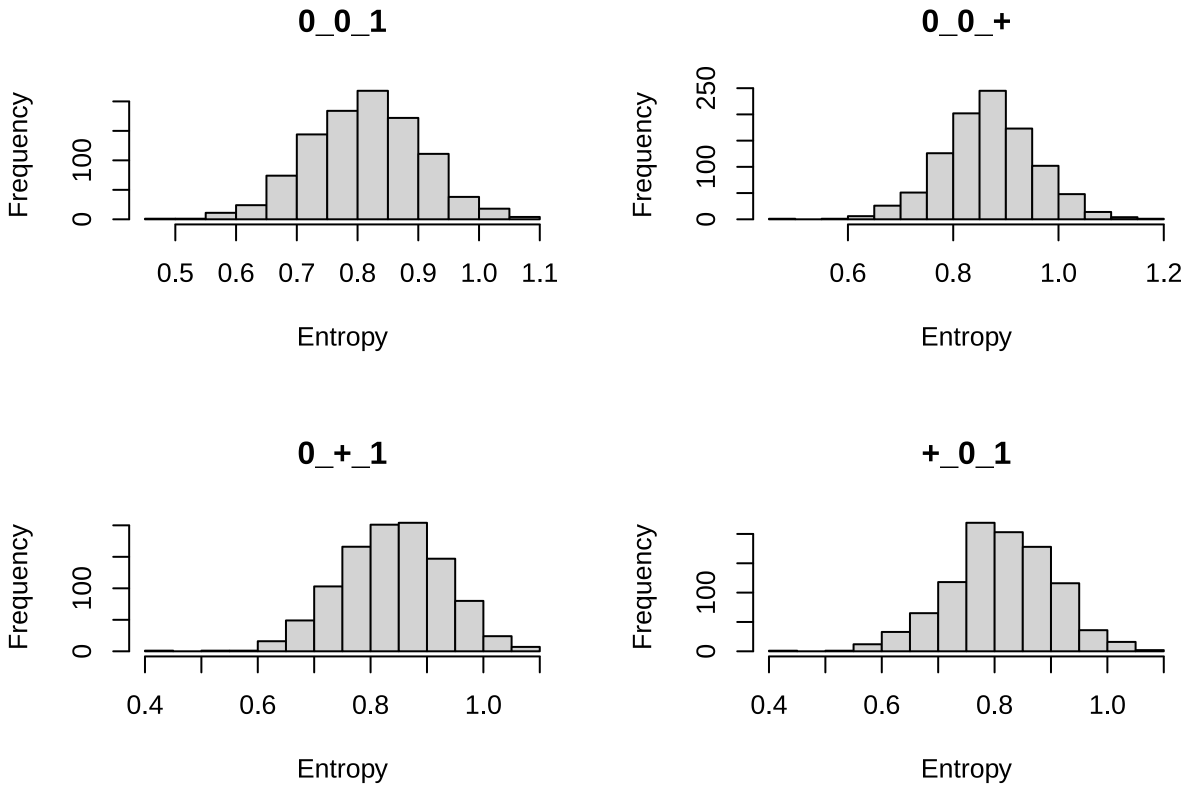
Looking at these graphs below, it seems that there isn't much shift happening in the two data. Thus, missing 1 bi-variable has little effect on 0_1_0. Since the graphs mostly overlap the origin, there is no interaction effect on the BMI distribution.


0_1_0 vs 0_1_+
Frequency
0
50
100
150
200
250
Entropy
0.0
0.5
1.0
1.5
2.0
Entropy of 0_1_0
Entropy of 0_1_+

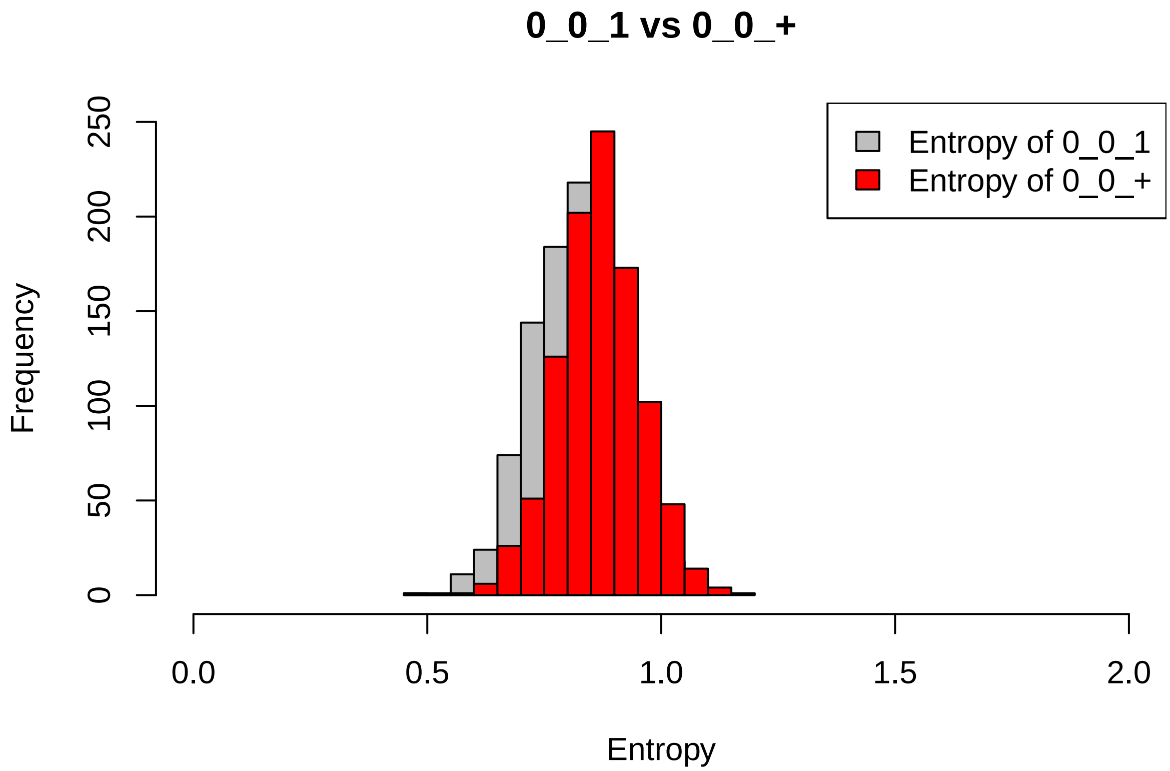

0_1_0 vs 0_+_0
Entropy of 0_1_0
Entropy of 0_+_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy


0_1_0 vs +_1_0
Entropy of 0_1_0
Entropy of +_1_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

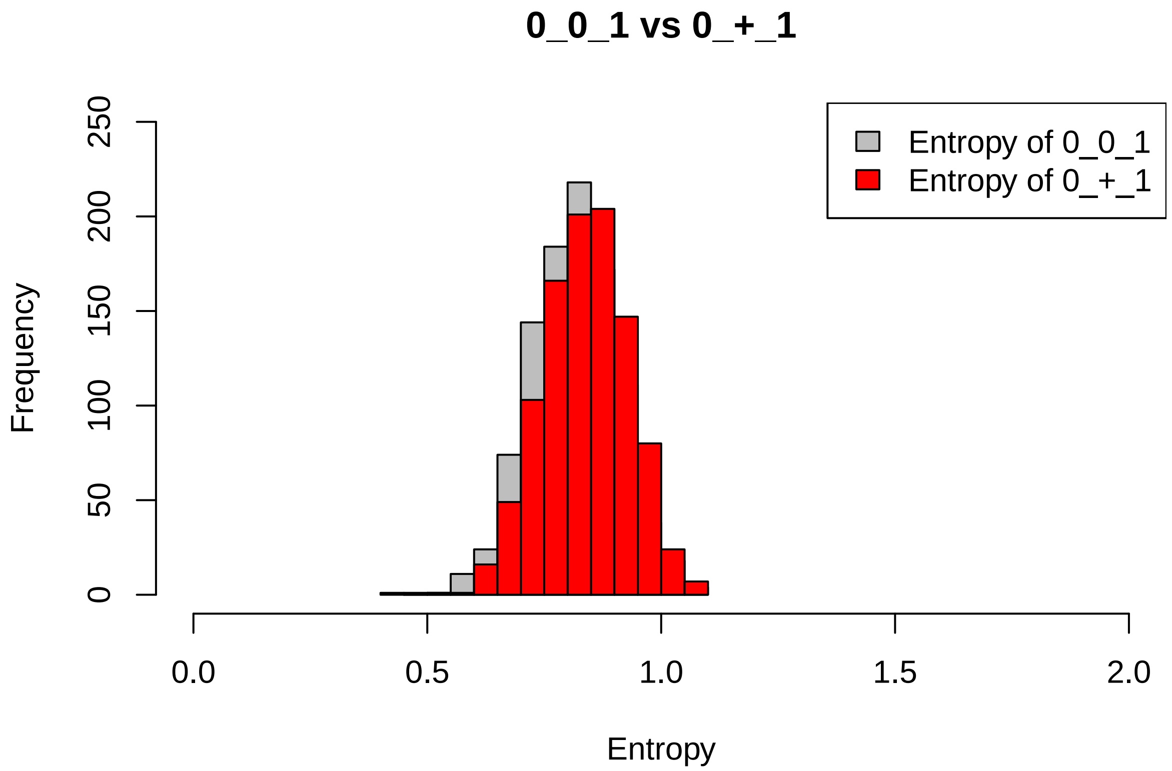
No HD __ No HighBP __ Yes HighChol

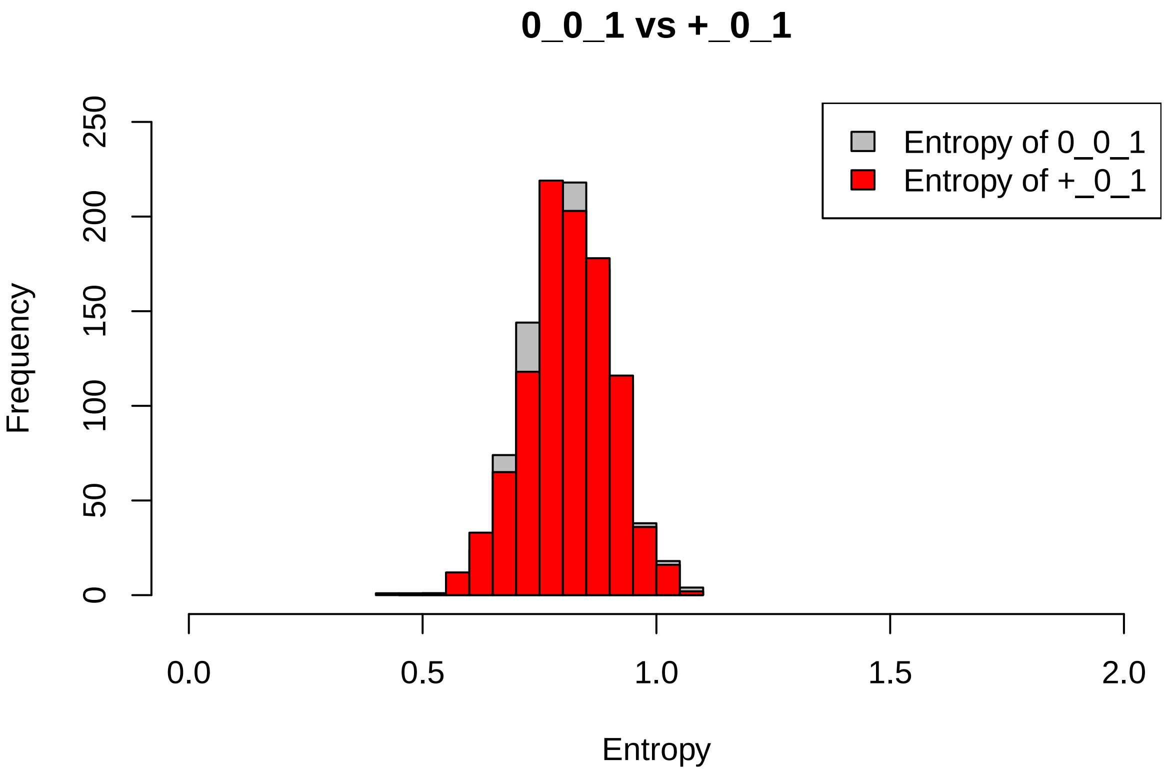


Looking at these graphs below, there is clearly a shift to the right when HighBP or HighChol is missing, which means the entropy level goes up and makes it less accurate. However, missing HD doesn't change much. Since the graphs of missing HighBP or missing HighChol doesn't overlap the origin, there is an interaction effect on the BMI distribution.

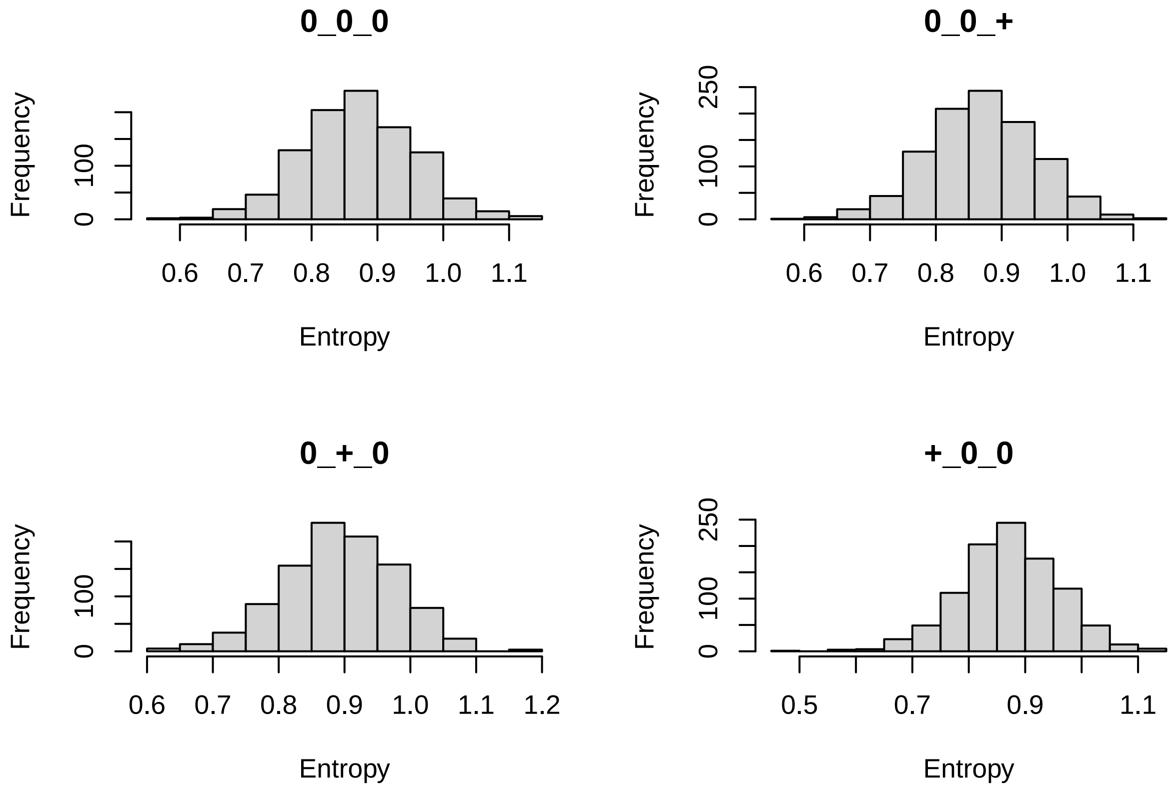


0_0_1 vs 0_0_+
Entropy of 0_0_1
Entropy of 0_0_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy


0_0_1 vs 0_+_1
Entropy of 0_0_1
Entropy of 0_+_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy


0_0_1 vs +_0_1
Entropy of 0_0_1
Entropy of +_0_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

No HD __ No HighBP __ No HighChol

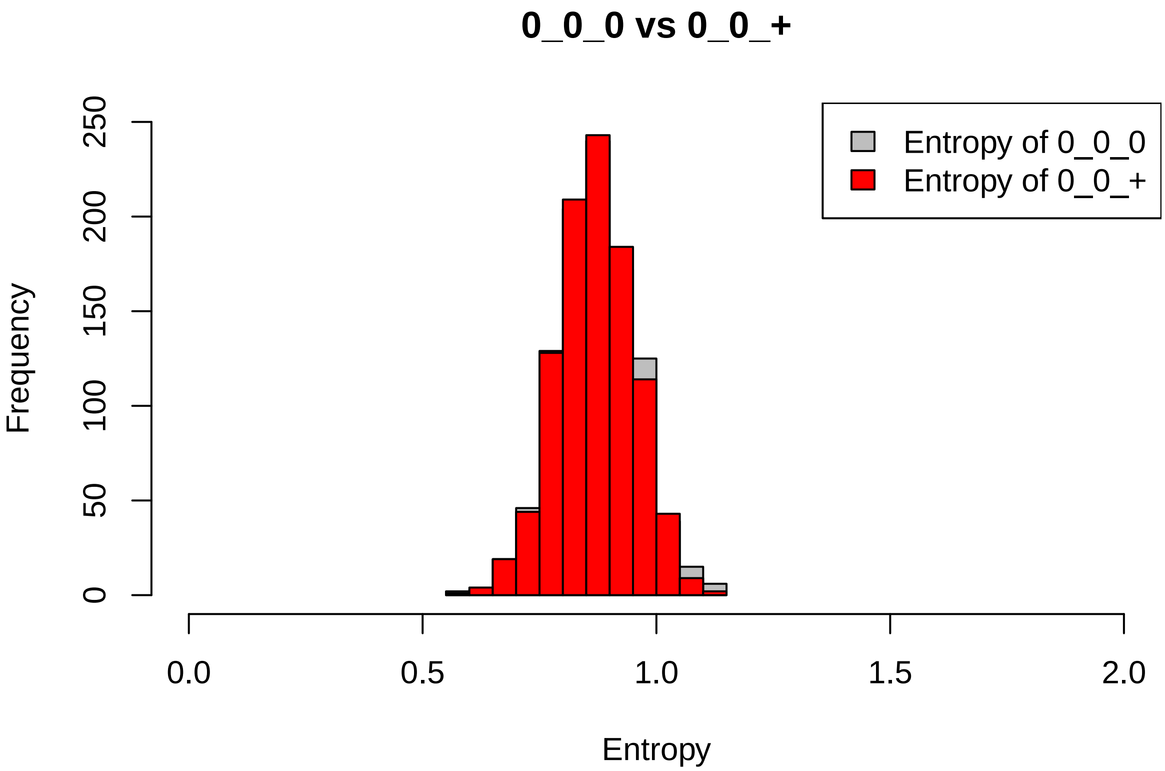

0_0_0

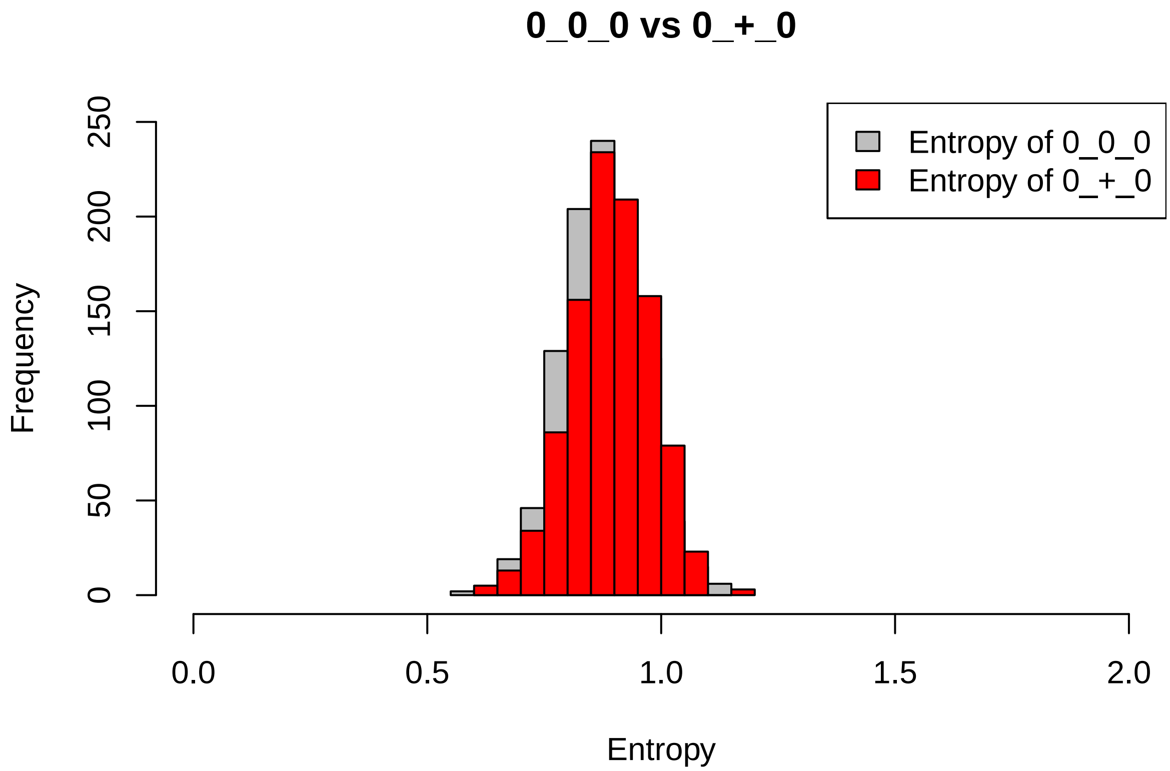
0_0_+

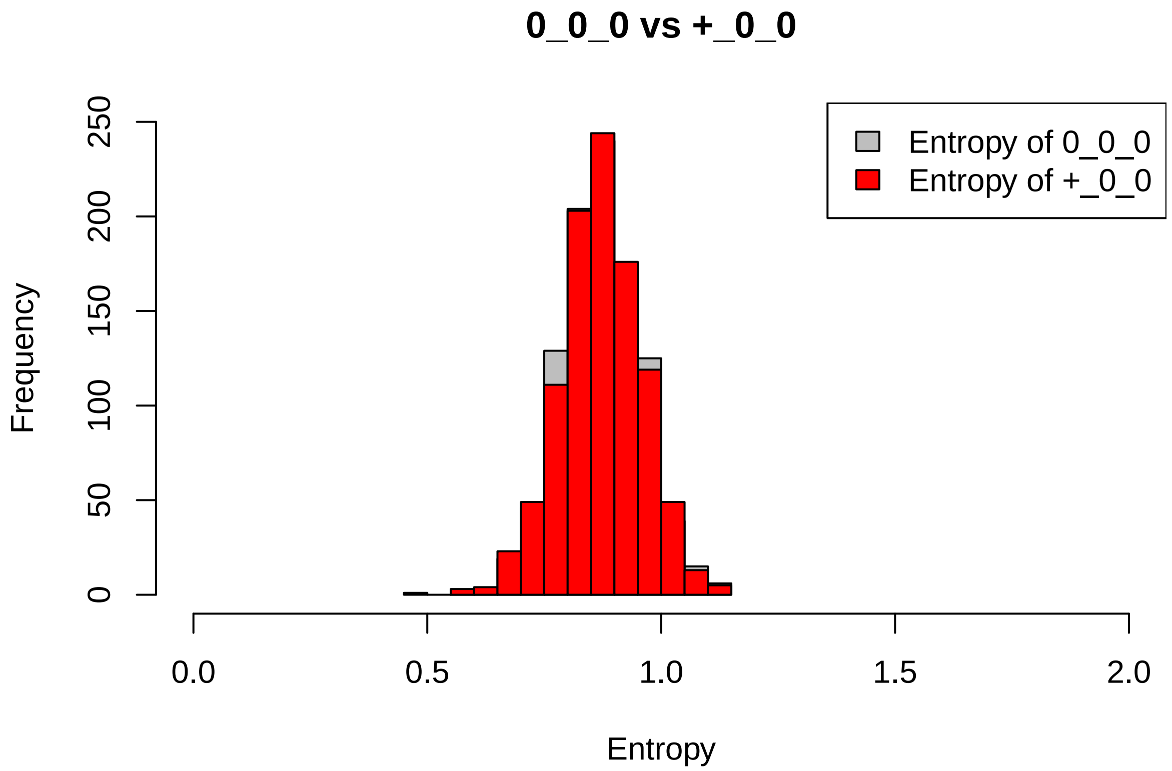
0_+_0

+_0_0

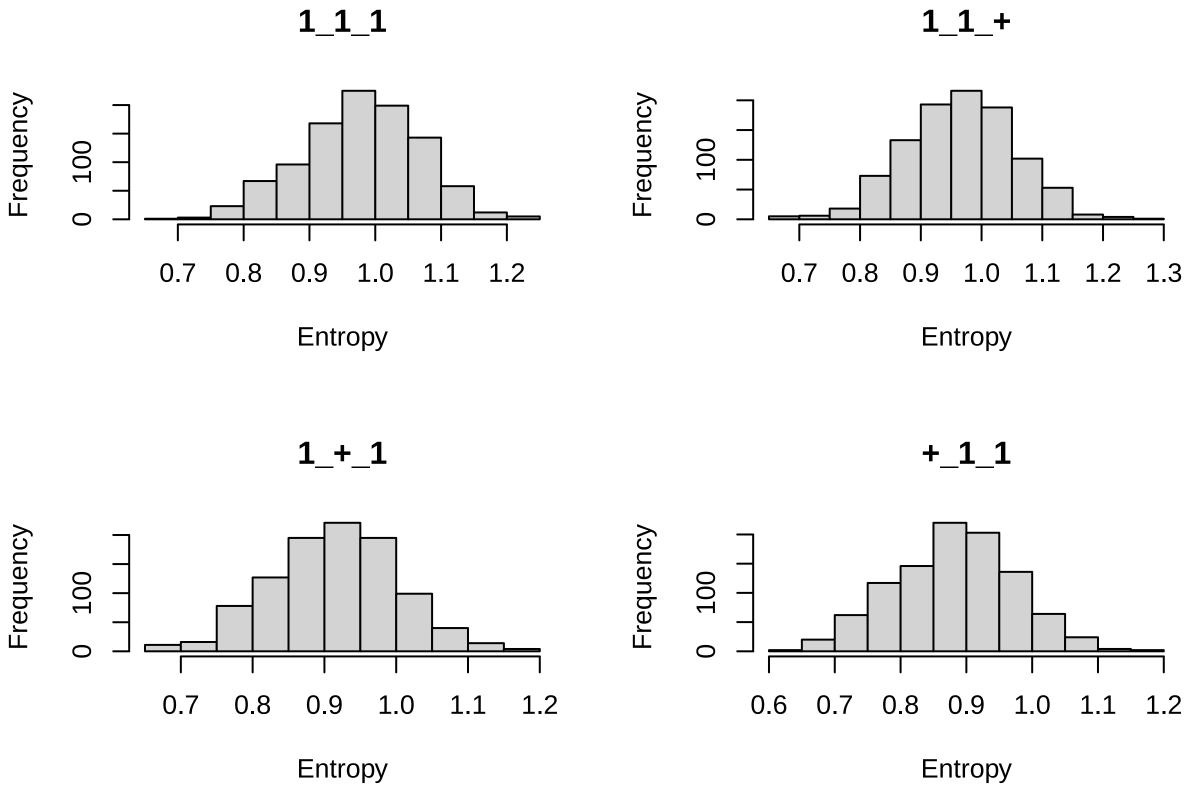
Looking at these graphs below, there is not much change going when missing one bi-variable. Since the graphs mostly overlap to the origin, there is no interaction effect on the BMI distribution.


0_0_0 vs 0_0_+
Entropy of 0_0_0
Entropy of 0_0_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

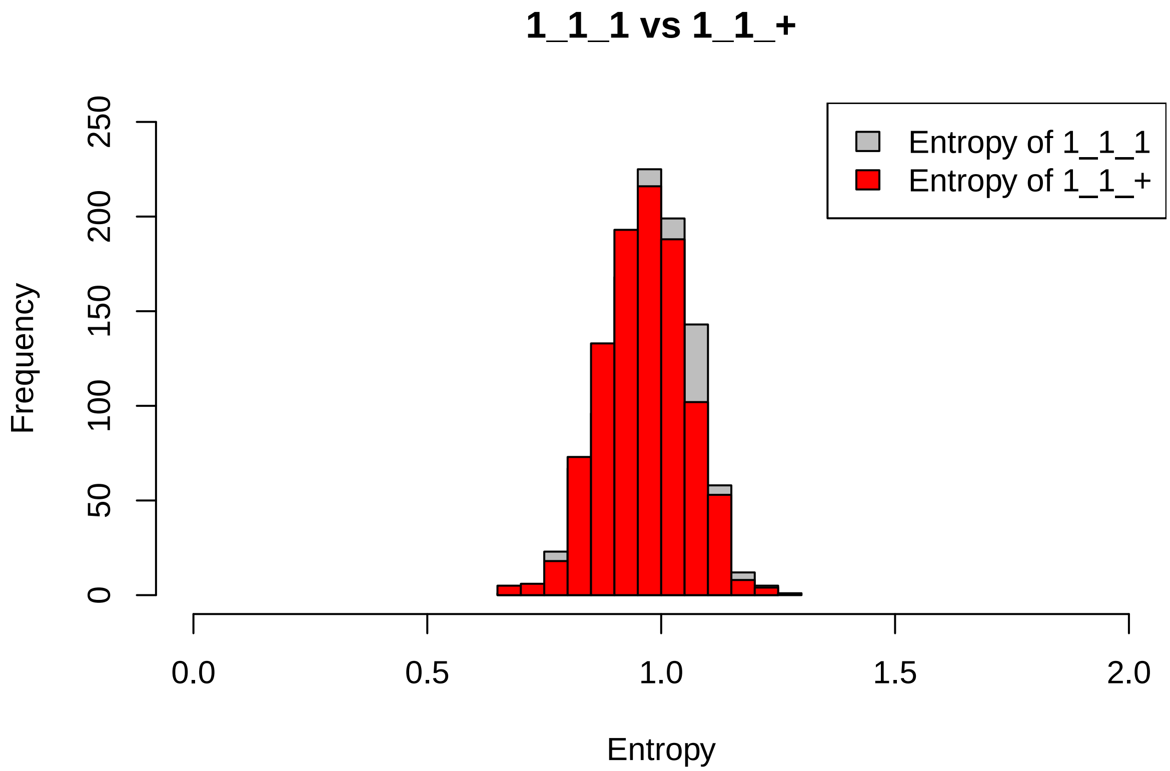

0_0_0 vs 0_+_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 0_0_0
Entropy of 0_+_0

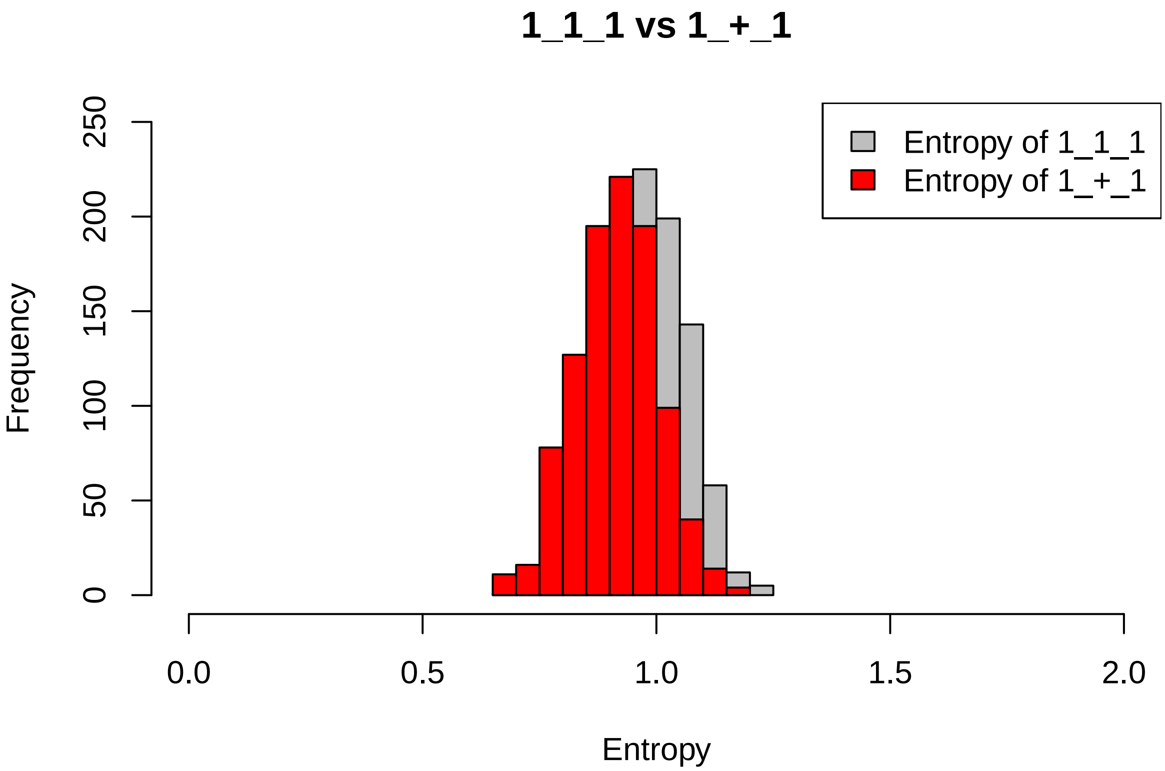

0_0_0 vs +_0_0
Entropy of 0_0_0
Entropy of +_0_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

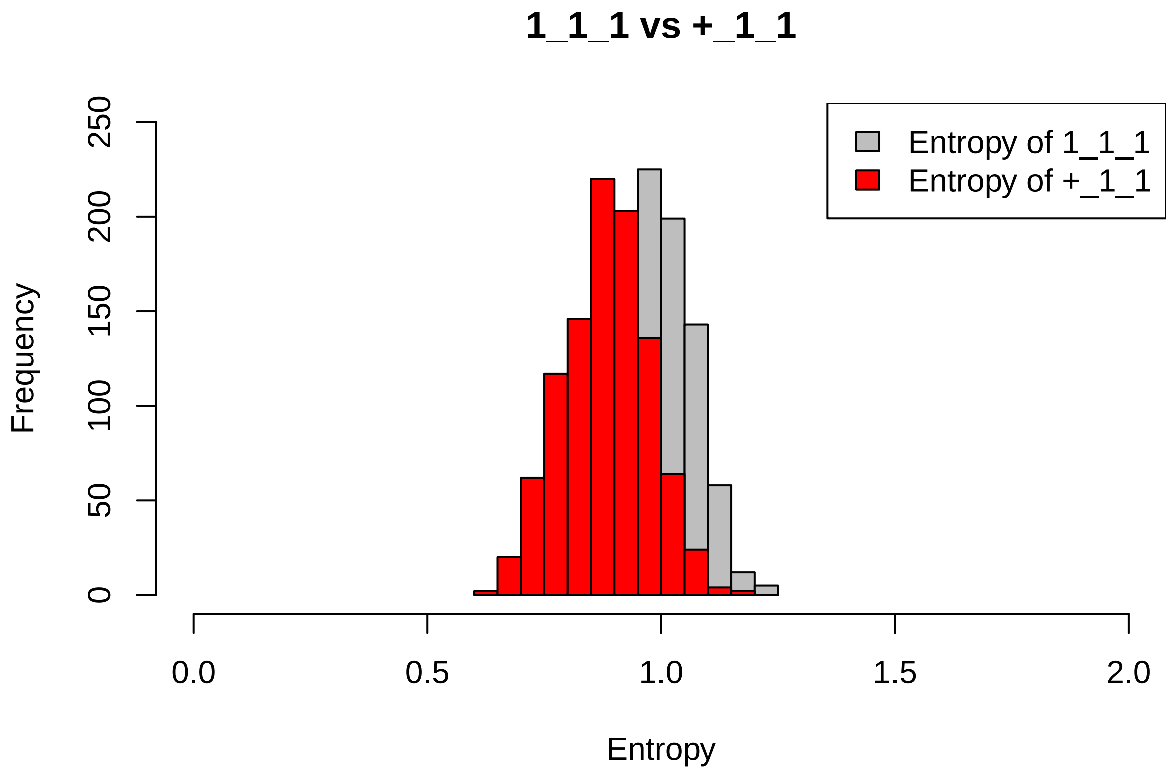
Yes HD __ Yes HighBP __ Yes HighChol



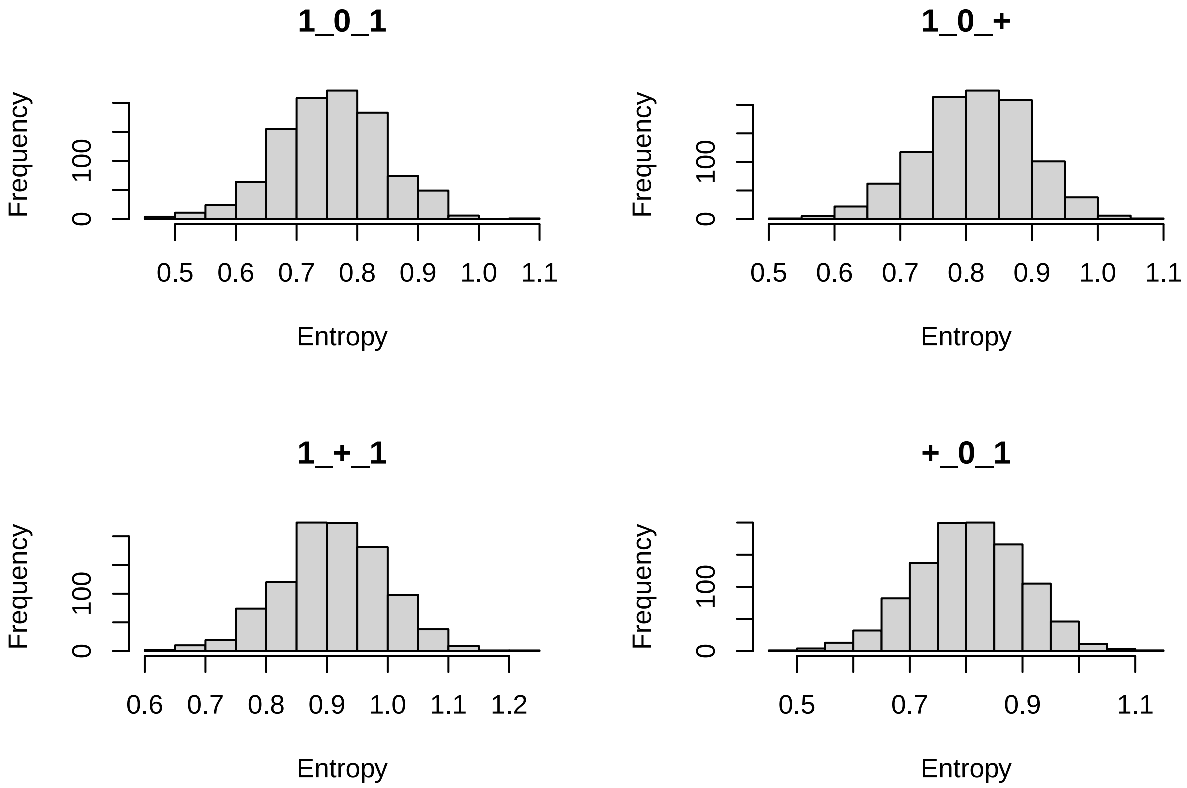
Looking at these graphs below, there is clearly a shift to the left when HighBP or HeartDisease is missing, which means the entropy level goes down and makes it more accurate. However, missing HighChol doesn't change much. Since the graphs of missing HighBP or missing HeartDisease doesn't overlap the origin, there is an interaction effect on the BMI distribution.


1_1_1 vs 1_1_+
Entropy of 1_1_1
Entropy of 1_1_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

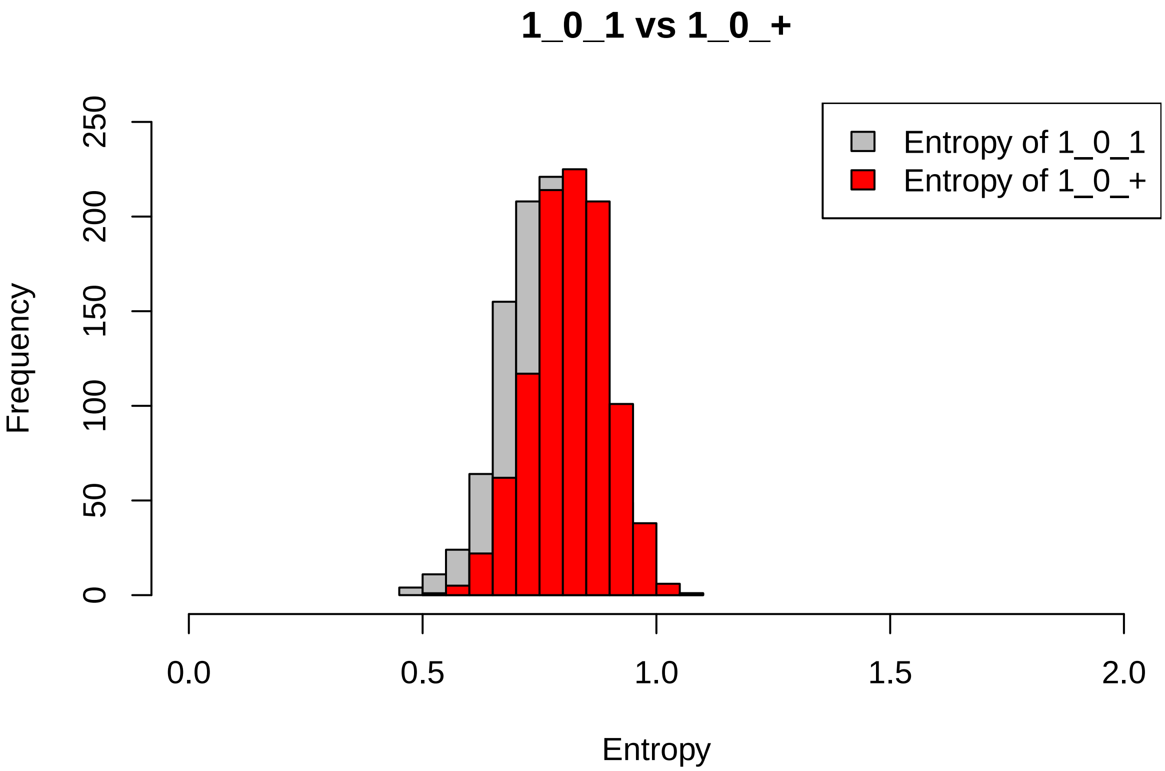

1_1_1 vs 1_+_1
Entropy of 1_1_1
Entropy of 1_+_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

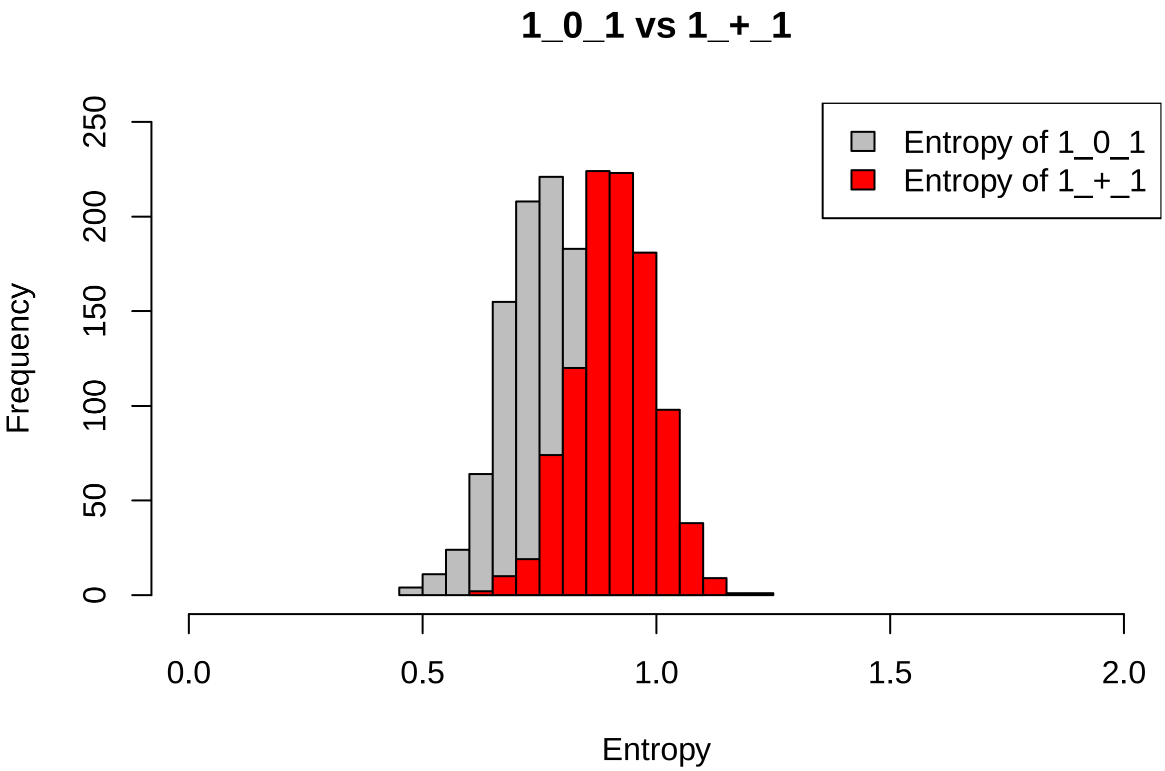

1_1_1 vs +_1_1
Entropy of 1_1_1
Entropy of +_1_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

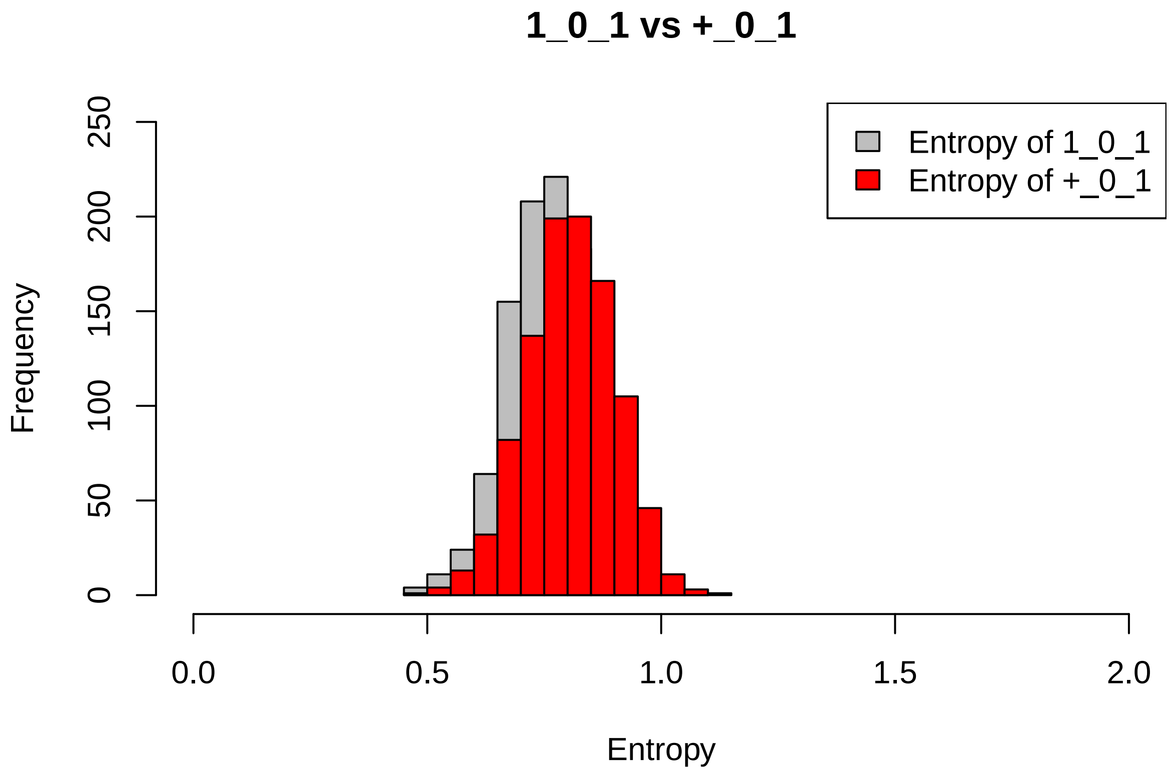
Yes HD __ No HighBP __ Yes HighChol



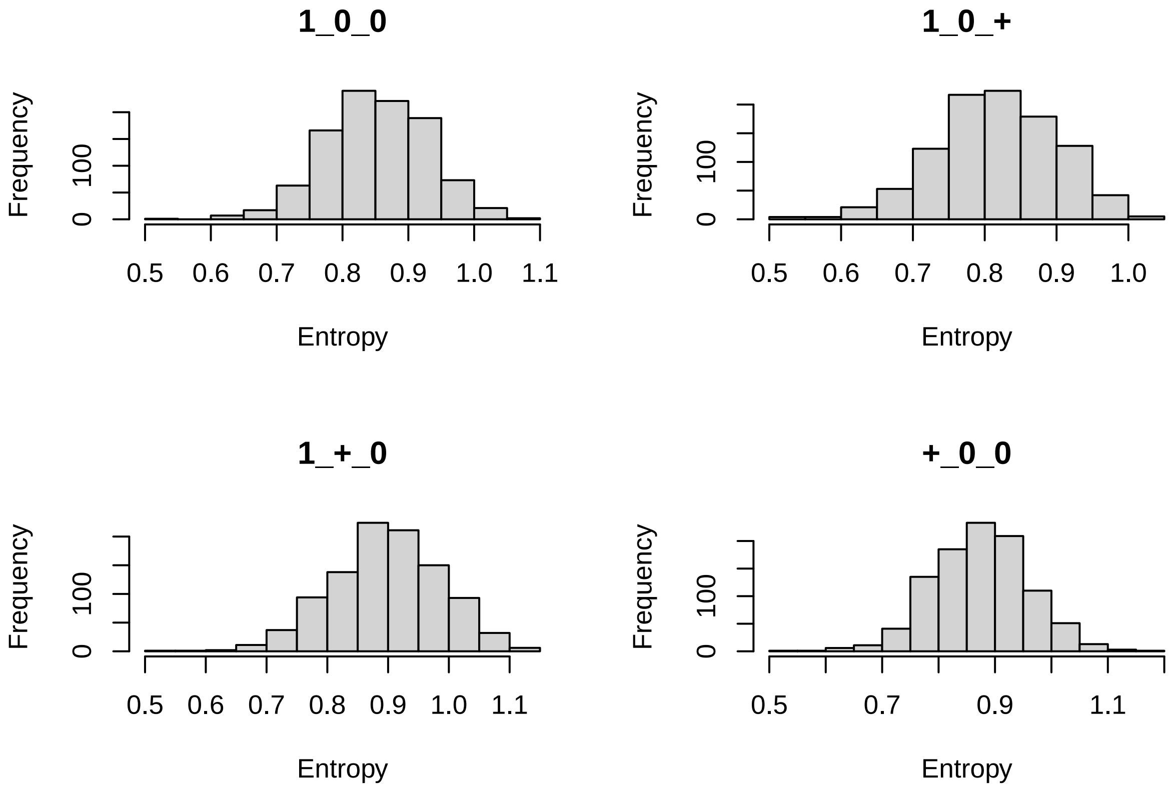
Looking at these graphs below, there is clearly a strong shift to the right when HighBP or HeartDisease or HighChol is missing, which means the entropy level goes up and makes it less accurate. Since all 3 graphs of missing HighBP or missing HeartDisease or missing HighChol don't overlap the origin, there is an interaction effect on the BMI distribution.


1_0_1 vs 1_0_+
Entropy of 1_0_1
Entropy of 1_0_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

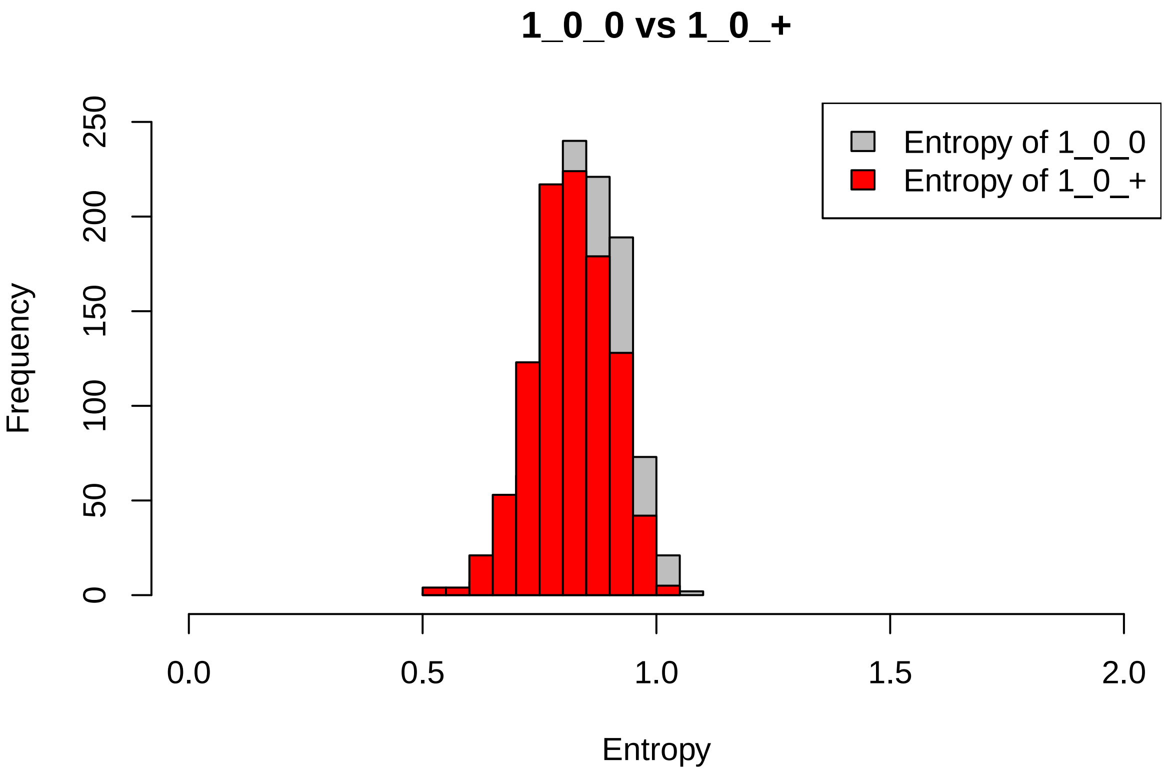

1_0_1 vs 1_+_1
Entropy of 1_0_1
Entropy of 1_+_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

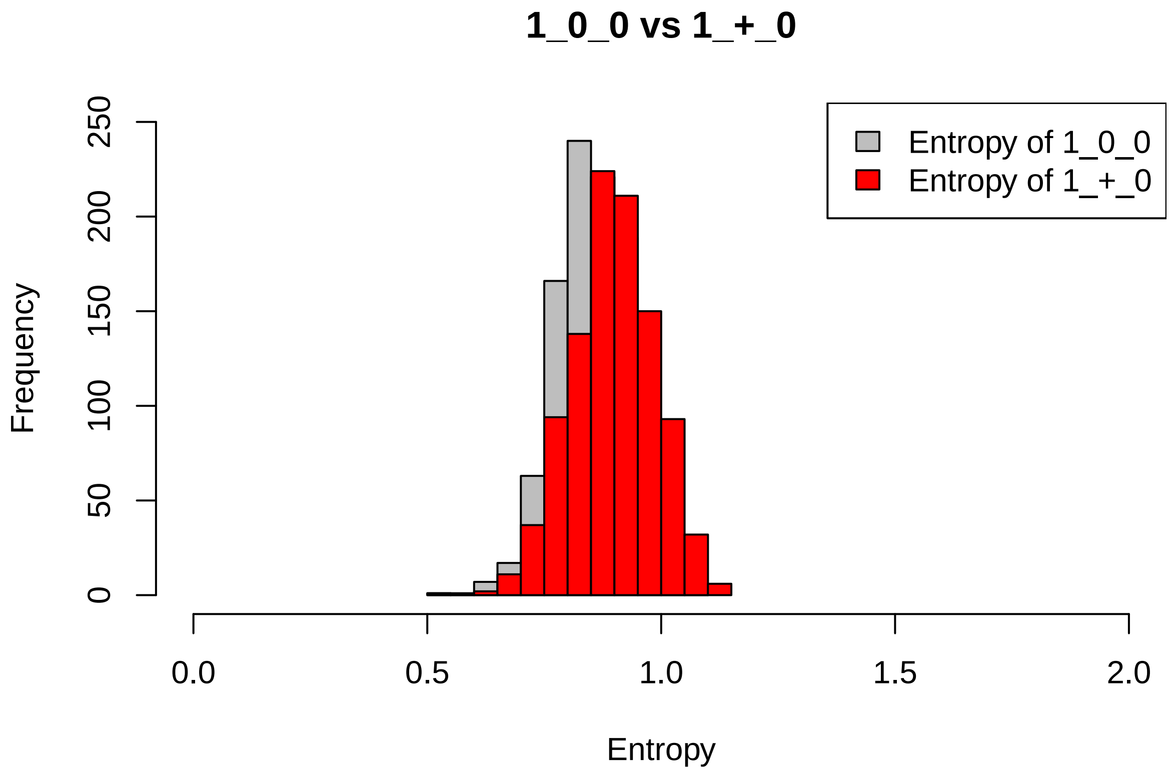

1_0_1 vs +_0_1
Entropy of 1_0_1
Entropy of +_0_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

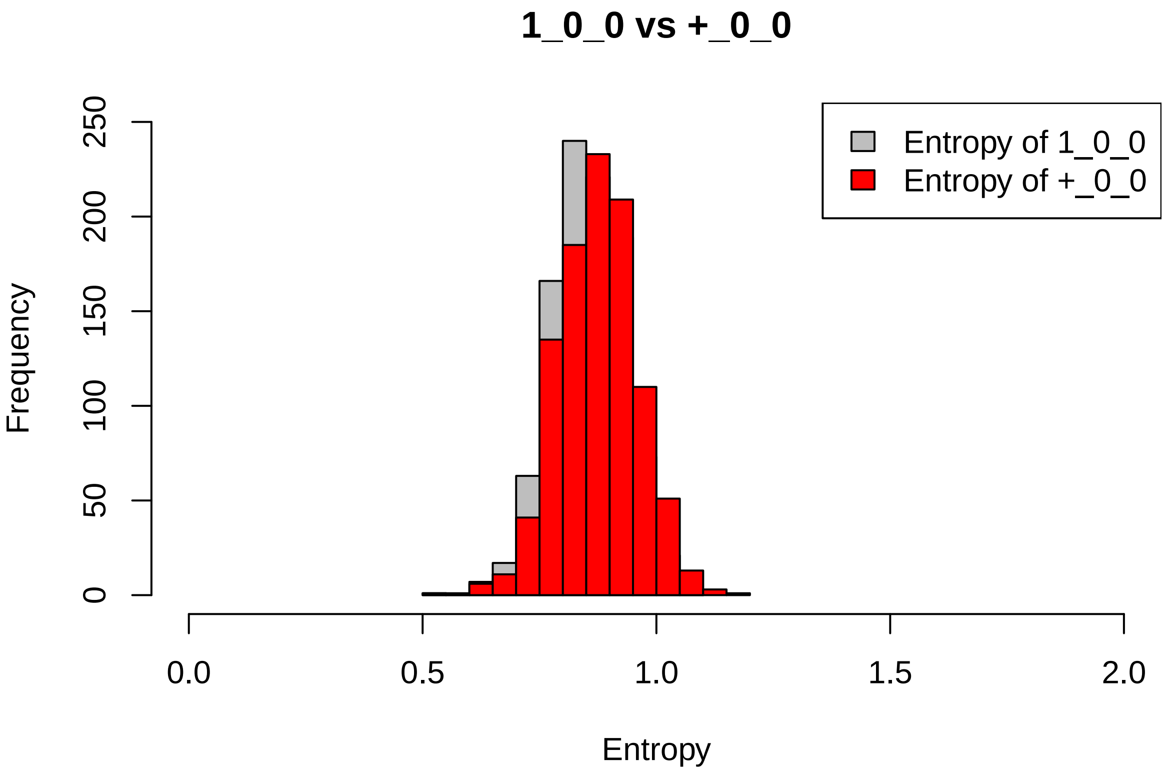
Yes HD __ No HighBP __ No HighChol



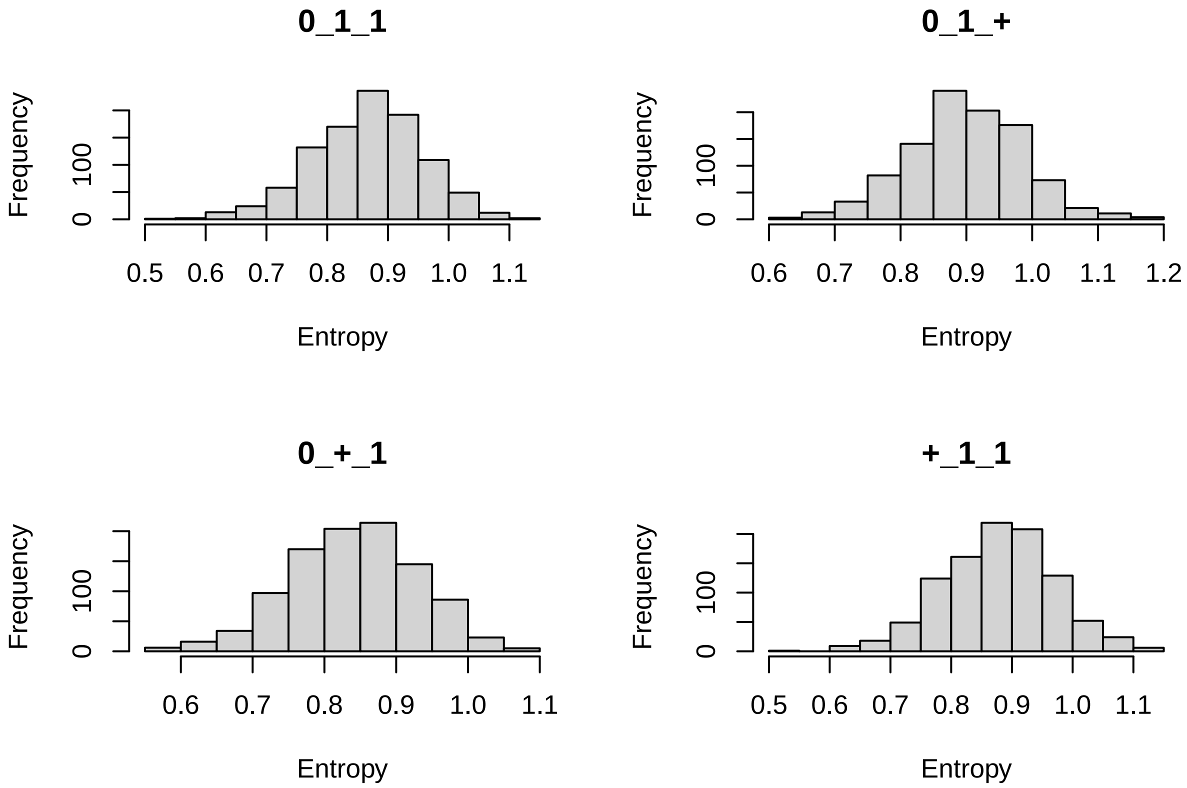
Looking at these graphs below, there is not much change happening. Thus, having missing 1 bi-variable have little effect on 1_0_0. Since the graphs mostly overlap the origin, there is no interaction effect on the BMI distribution.


1_0_0 vs 1_0_+
Entropy of 1_0_0
Entropy of 1_0_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

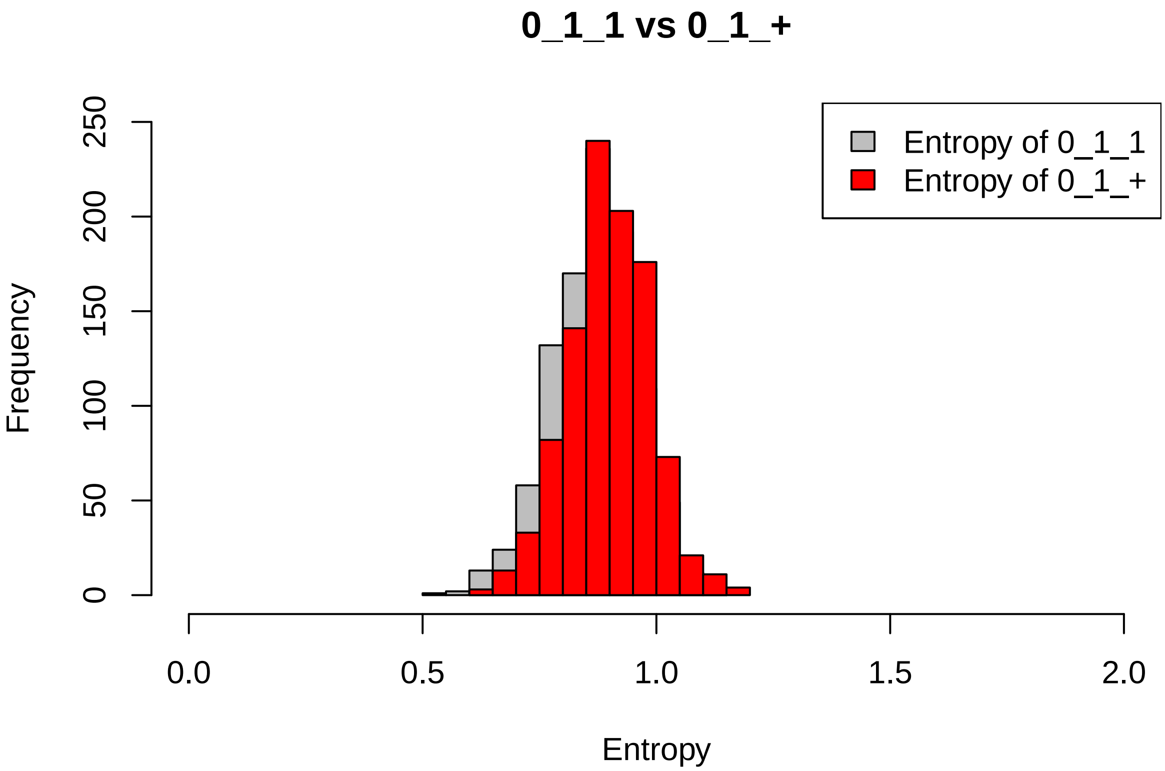

1_0_0 vs 1_+_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 1_0_0
Entropy of 1_+_0

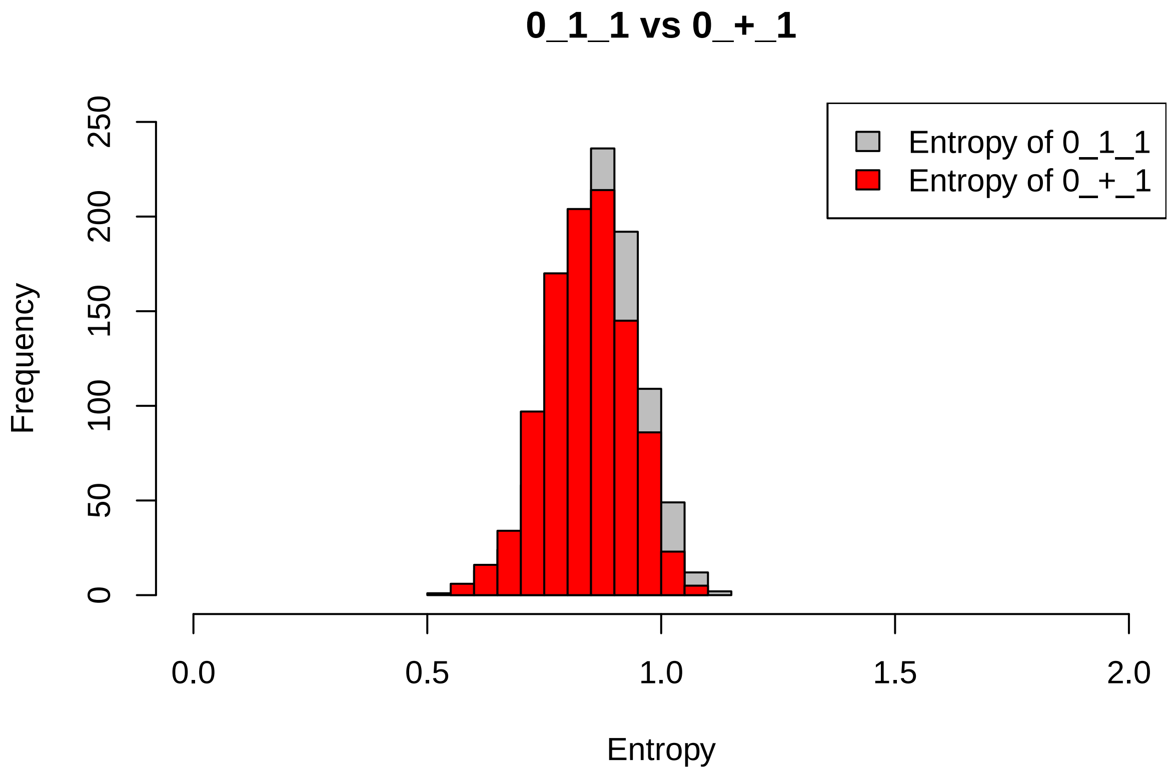

1_0_0 vs +_0_0
Entropy of 1_0_0
Entropy of +_0_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

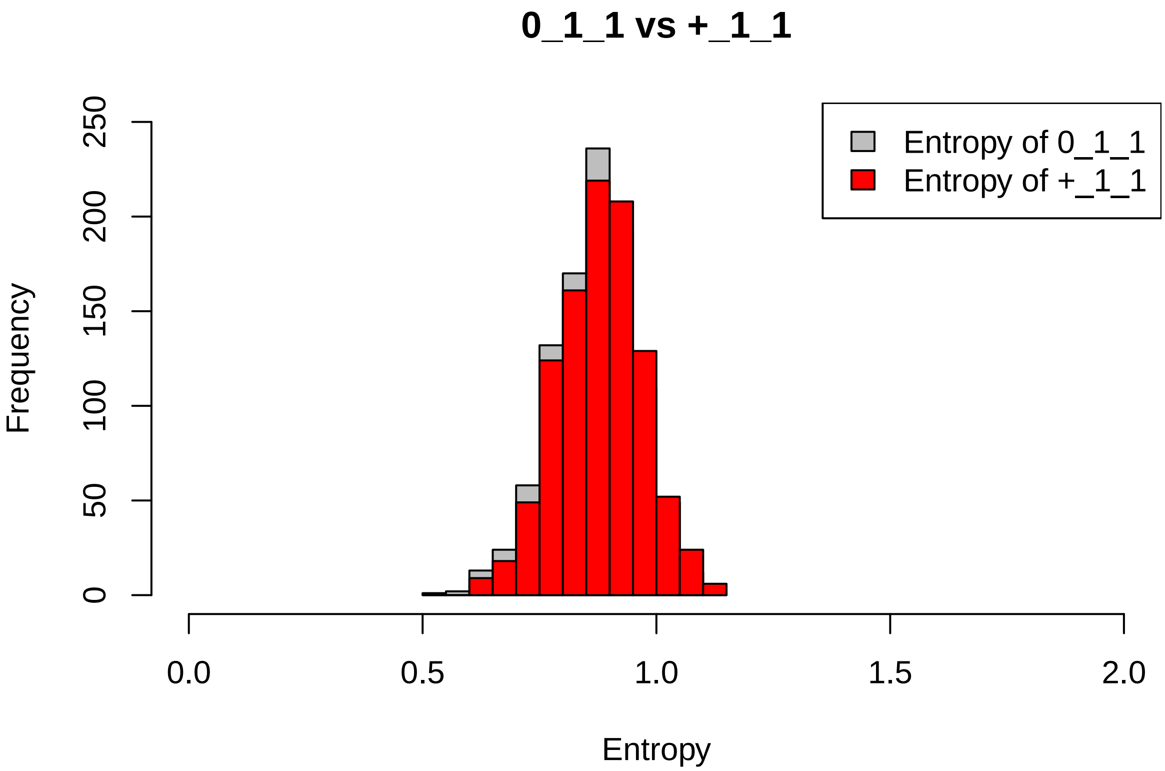
No HD _ Yes HighBP _ Yes HighChol



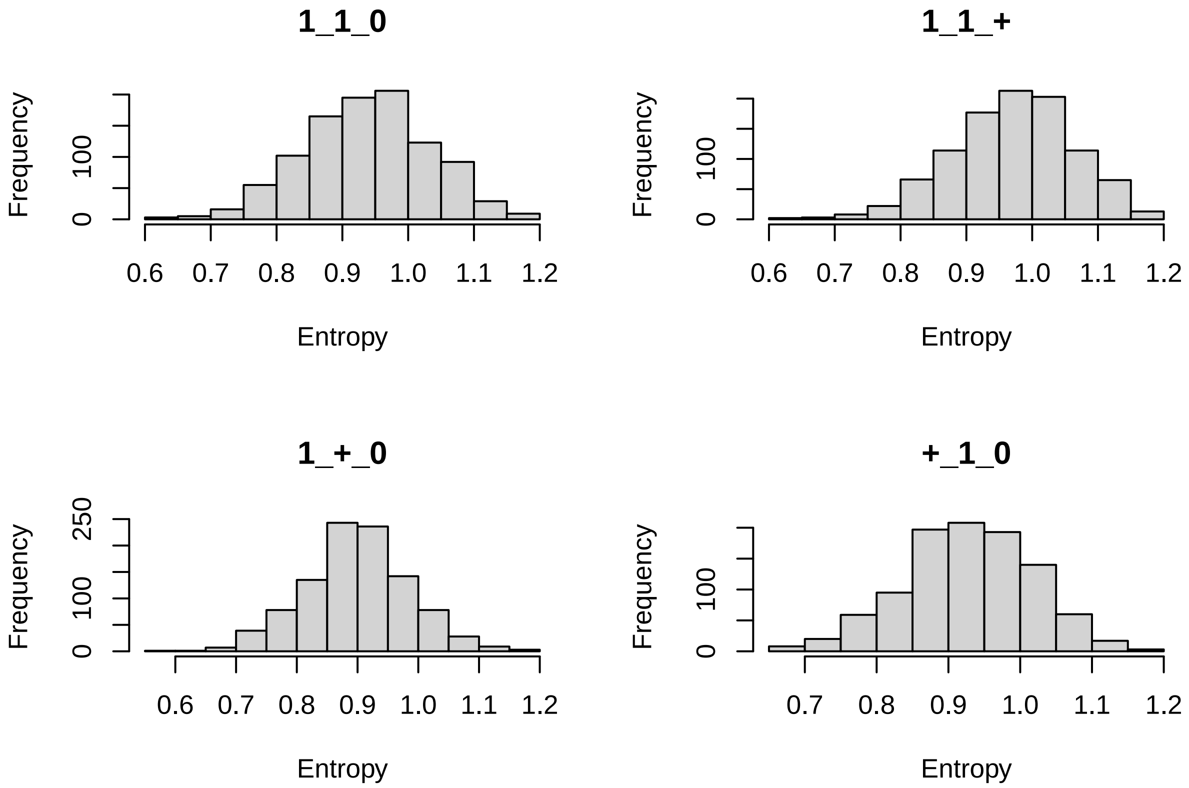
Looking at these graphs below, there is not much change happening. Thus, missing one bi-variable has little effect on 0_1_1. Since the graphs mostly overlap the origin, there is no interaction effect on the BMI distribution.


0_1_1 vs 0_1_+
Frequency
0
50
100
150
200
250
Entropy of 0_1_1
Entropy of 0_1_+
0.0
0.5
1.0
1.5
2.0
Entropy

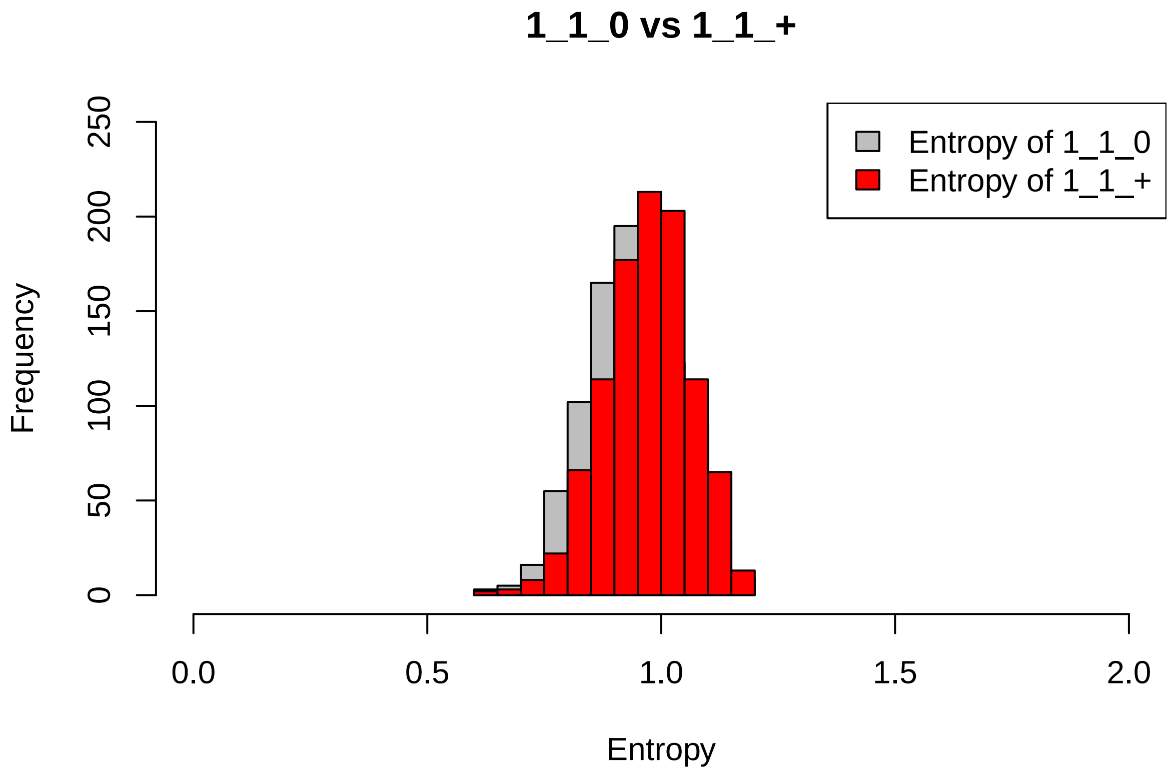

0_1_1 vs 0_+_1
Frequency
0
50
100
150
200
250
Entropy
0.0
0.5
1.0
1.5
2.0
Entropy of 0_1_1
Entropy of 0_+_1

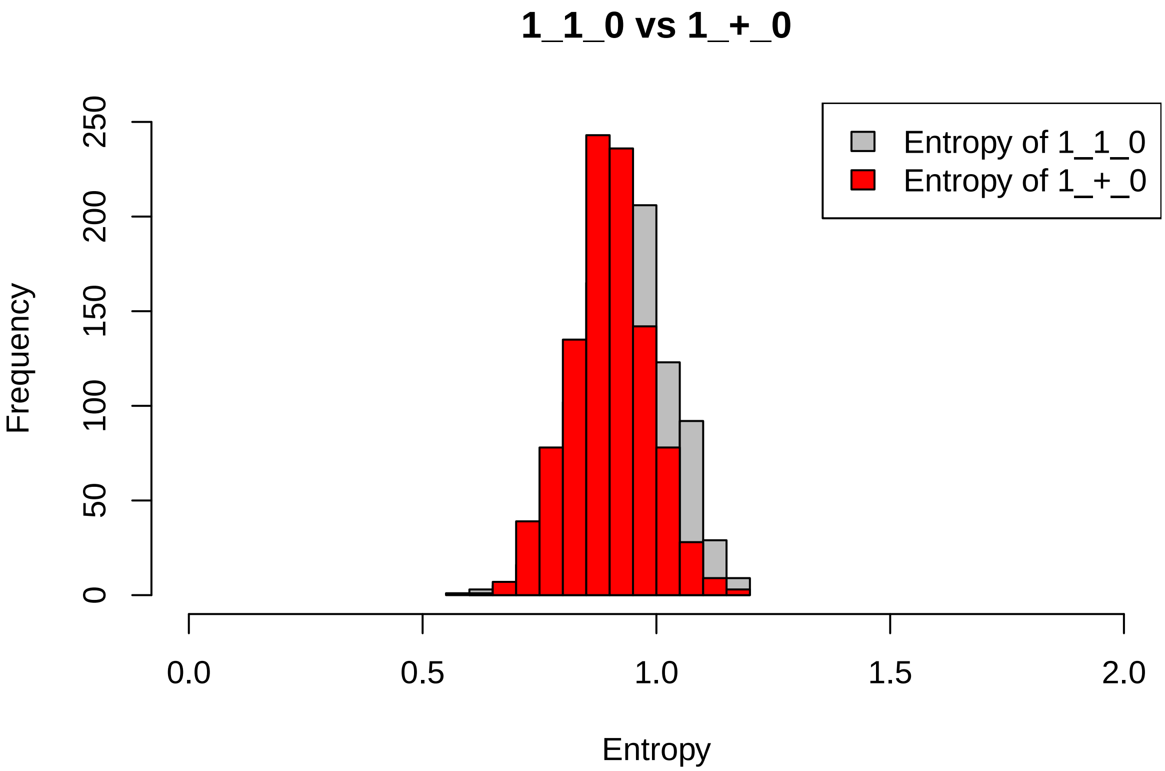

0_1_1 vs +_1_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 0_1_1
Entropy of +_1_1

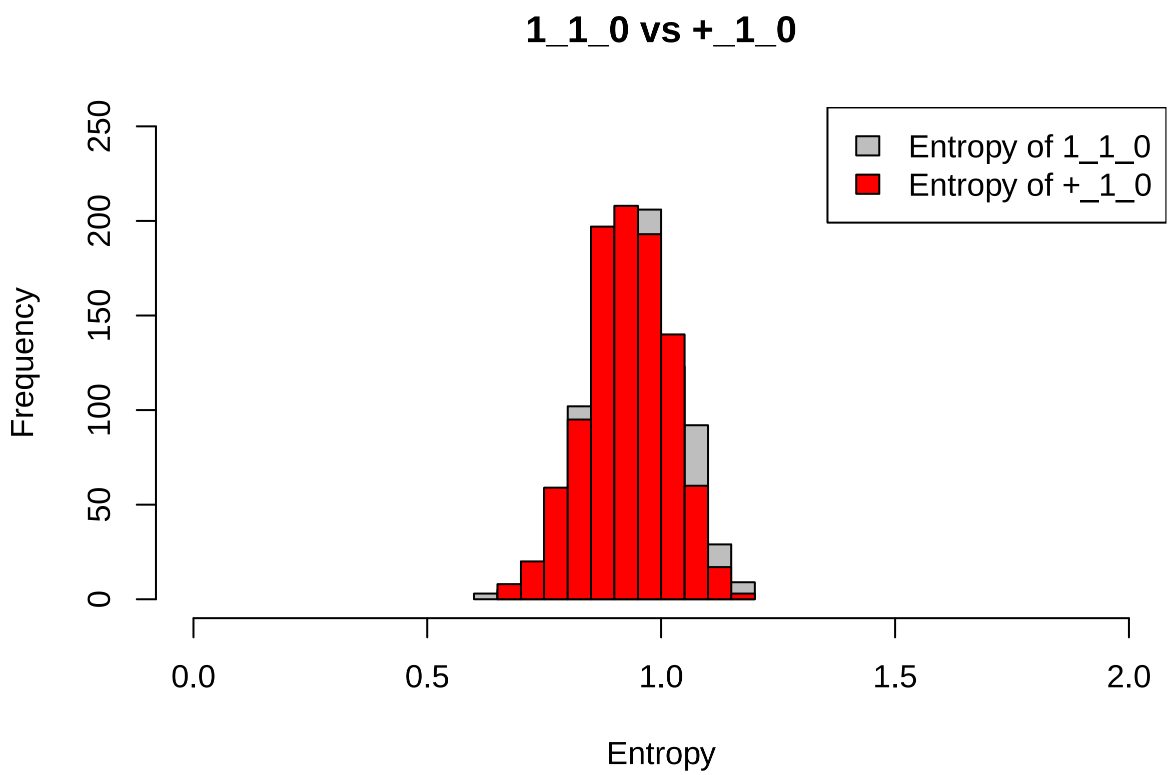
Yes HD __ Yes HighBP __ No HighChol



Looking at these graphs below, there is not much change happening. Thus, missing one bi-variable has little effect on 1_1_0. Since the graphs mostly overlap the origin, there is no interaction effect on the BMI distribution.


1_1_0 vs 1_1_+
Entropy of 1_1_0
Entropy of 1_1_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

1_1_0 vs 1_+_0
Entropy of 1_1_0
Entropy of 1_+_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy


1_1_0 vs +_1_0
Entropy of 1_1_0
Entropy of +_1_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

2/ Investigating the 3-way interacting effects in GenHealth_2 sub-dataset:

Creating contigency tables based on the combination of these three selected binary variable

```
## Warning: package 'iNZightTools' was built under R version 4.2.2
```

```
##
## Attaching package: 'iNZightTools'
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

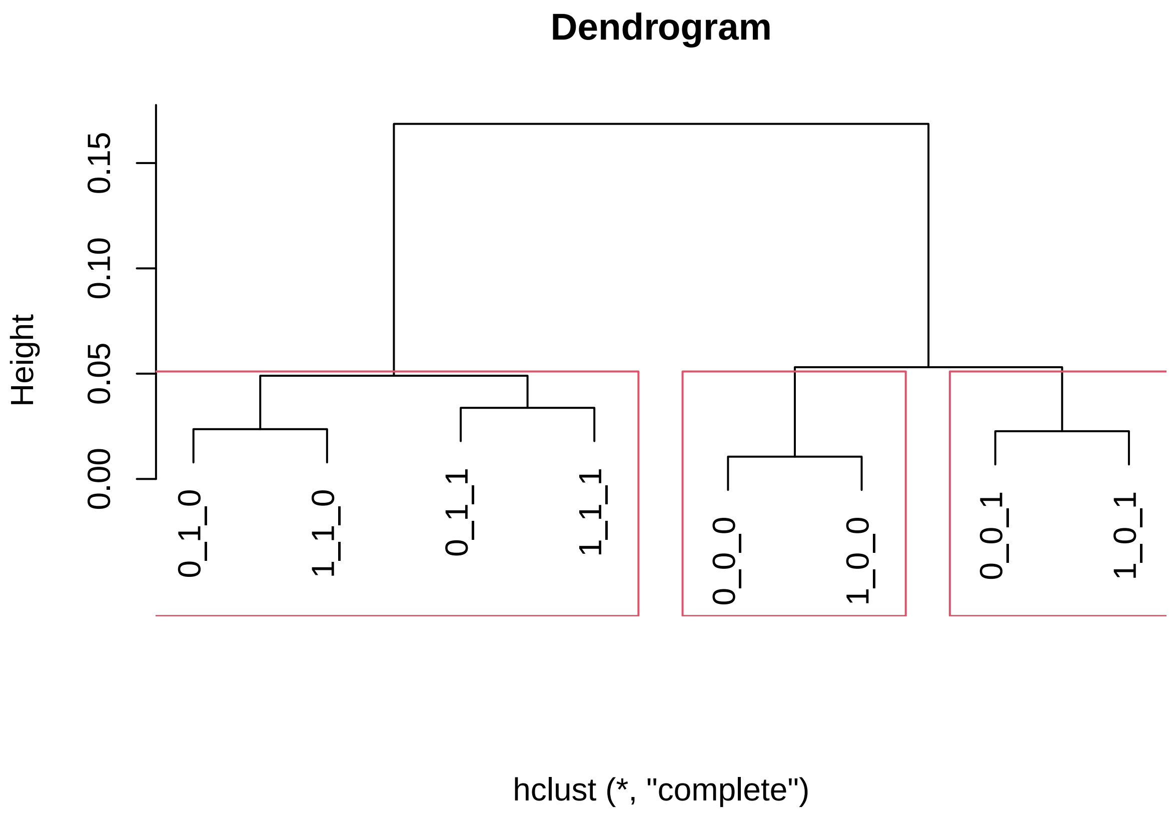
```
##                1         2         3           4           5            6
## 0_1_0 0.03053763 0.5779928 0.3170609 0.062724014 0.008602151 8.602151e-04
## 0_1_1 0.02143375 0.5959243 0.3237486 0.051704808 0.004814351 2.638000e-04
## 0_0_0 0.07175553 0.6870976 0.2024396 0.032168008 0.003848089 2.263581e-04
## 0_0_1 0.04339986 0.6922838 0.2306784 0.027482435 0.003108873 6.217745e-05
## 1_1_1 0.02643172 0.6206559 0.3024963 0.044542340 0.003915810 4.894763e-04
## 1_0_0 0.06550218 0.6943231 0.2052402 0.029112082 0.002911208 0.000000e+00
## 1_1_0 0.04748201 0.5841727 0.3050360 0.054676259 0.004316547 0.000000e+00
## 1_0_1 0.04267425 0.7055477 0.2361309 0.009957326 0.002844950 0.000000e+00
## Total 0.05011001 0.6533047 0.2495173 0.039692874 0.004613623 3.030847e-04
##                  7            8            9           10
## 0_1_0 0.0004301075 0.0010035842 0.0005734767 0.0002150538
## 0_1_1 0.0001978500 0.0008573501 0.0006595001 0.0003957001
## 0_0_0 0.0004024145 0.0010814889 0.0008299799 0.0001509054
## 0_0_1 0.0001865324 0.0008083069 0.0014300815 0.0005595971
## 1_1_1 0.0000000000 0.0009789525 0.0004894763 0.0000000000
## 1_0_0 0.0014556041 0.0000000000 0.0014556041 0.0000000000
## 1_1_0 0.0000000000 0.0014388489 0.0028776978 0.0000000000
## 1_0_1 0.0000000000 0.0014224751 0.0014224751 0.0000000000
## Total 0.0003255354 0.0009766063 0.0008868035 0.0002694086
```

```
##                1         2         3          4           5            6
## 0_1   0.02579604 0.5873321 0.3205441 0.05698485 0.006629341 0.0005495827
## 0_0   0.06358899 0.6885912 0.2105725 0.03081854 0.003635192 0.0001790735
## 1_1   0.03177502 0.6113952 0.3031410 0.04711468 0.004017531 0.0003652301
## 1_0   0.05395683 0.7000000 0.2208633 0.01942446 0.002877698 0.0000000000
## Total 0.05011001 0.6533047 0.2495173 0.03969287 0.004613623 0.0003030847
##                  7            8            9           10
## 0_1   0.0003091402 0.0009274207 0.0006182805 0.0003091402
## 0_0   0.0003402396 0.0010028115 0.0010028115 0.0002686102
## 1_1   0.0000000000 0.0010956903 0.0010956903 0.0000000000
## 1_0   0.0007194245 0.0007194245 0.0014388489 0.0000000000
## Total 0.0003255354 0.0009766063 0.0008868035 0.0002694086
```

```
##                1         2         3          4           5            6
## 0_0   0.06105008 0.6587600 0.2322100 0.04010426 0.005082852 0.0003909886
## 0_1   0.03274019 0.6455226 0.2758433 0.03923702 0.003936504 0.0001600205
## 1_1   0.03058995 0.6423889 0.2855062 0.03568827 0.003641661 0.0003641661
## 1_0   0.05643994 0.6389291 0.2554269 0.04196816 0.003617945 0.0000000000
## Total 0.05011001 0.6533047 0.2495173 0.03969287 0.004613623 0.0003030847
##                   7            8            9           10
## 0_0   0.0004096071 0.0010612549 0.0007633588 0.0001675666
## 0_1   0.0001920246 0.0008321065 0.0010561352 0.0004800614
## 1_1   0.0000000000 0.0010924982 0.0007283321 0.0000000000
## 1_0   0.0007235890 0.0007235890 0.0021707670 0.0000000000
## Total 0.0003255354 0.0009766063 0.0008868035 0.0002694086
```

```
##                 1         2         3          4           5            6
## 1_0   0.03134175 0.5782861 0.3164903 0.06234210 0.008398771 8.193923e-04
## 1_1   0.02202720 0.5988609 0.3212252 0.05085435 0.004707660 2.905963e-04
## 0_0   0.07164932 0.6872203 0.2024872 0.03211610 0.003832175 2.225134e-04
## 0_1   0.04336947 0.6928393 0.2309067 0.02674848 0.003097820 5.957345e-05
## Total 0.05011001 0.6533047 0.2495173 0.03969287 0.004613623 3.030847e-04
##                   7            8            9           10
## 1_0   0.0004096961 0.0010242404 0.0006828269 0.0002048481
## 1_1   0.0001743578 0.0008717889 0.0006393119 0.0003487156
## 0_0   0.0004203031 0.0010631196 0.0008406062 0.0001483423
## 0_1   0.0001787204 0.0008340284 0.0014297629 0.0005361611
## Total 0.0003255354 0.0009766063 0.0008868035 0.0002694086
```

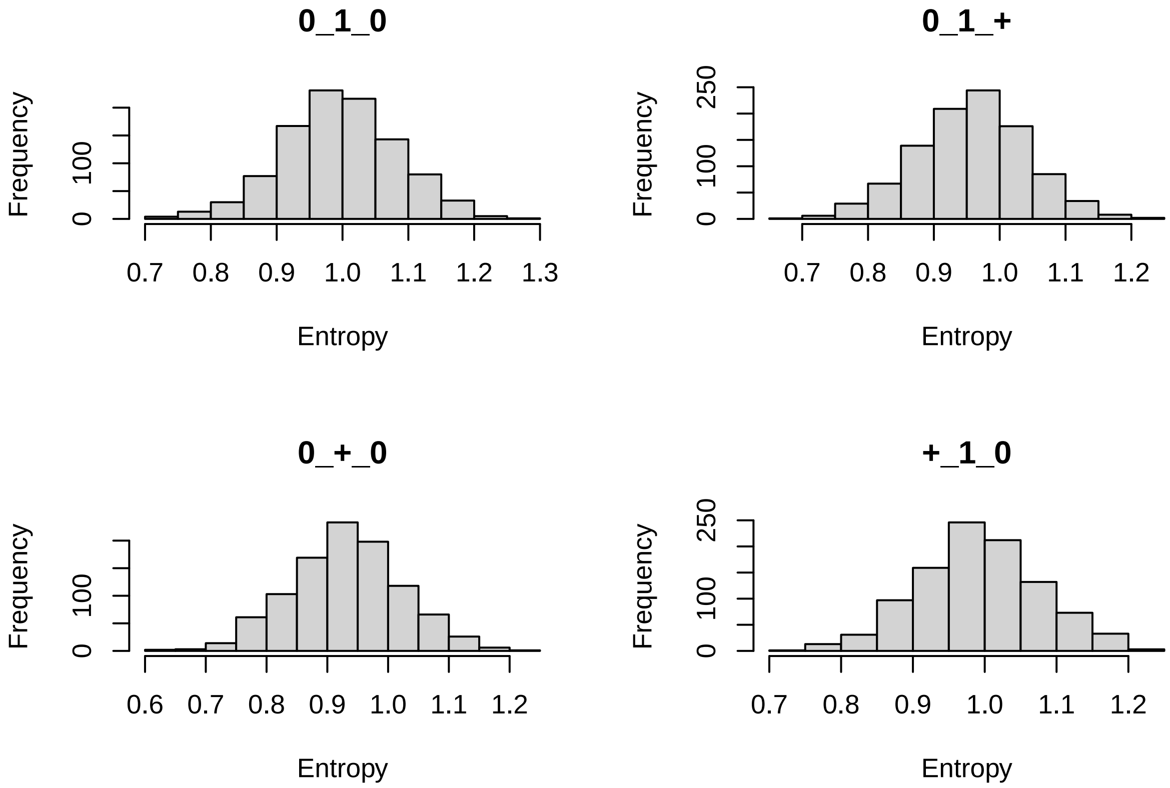
3-way HC-tree of GenHealth_2:



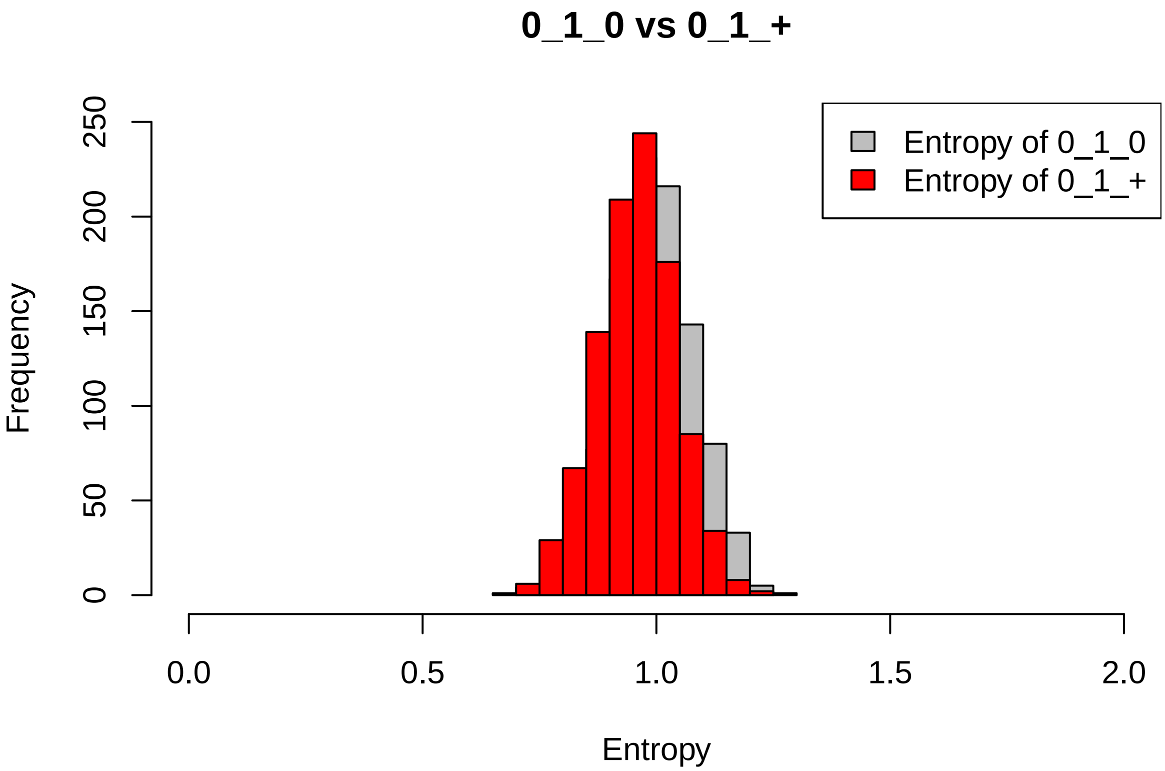
Comparing the entropies of categorical variables to see the effects

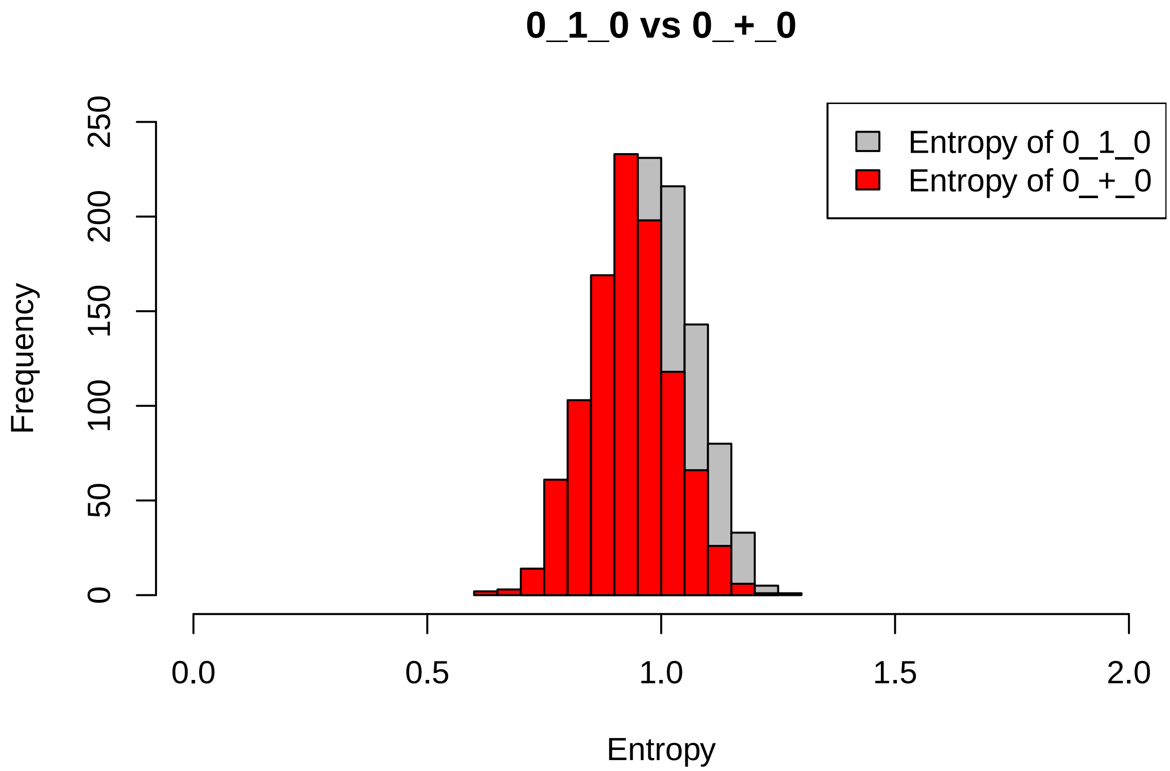
No HD __ Yes HighBP __ No HighChol

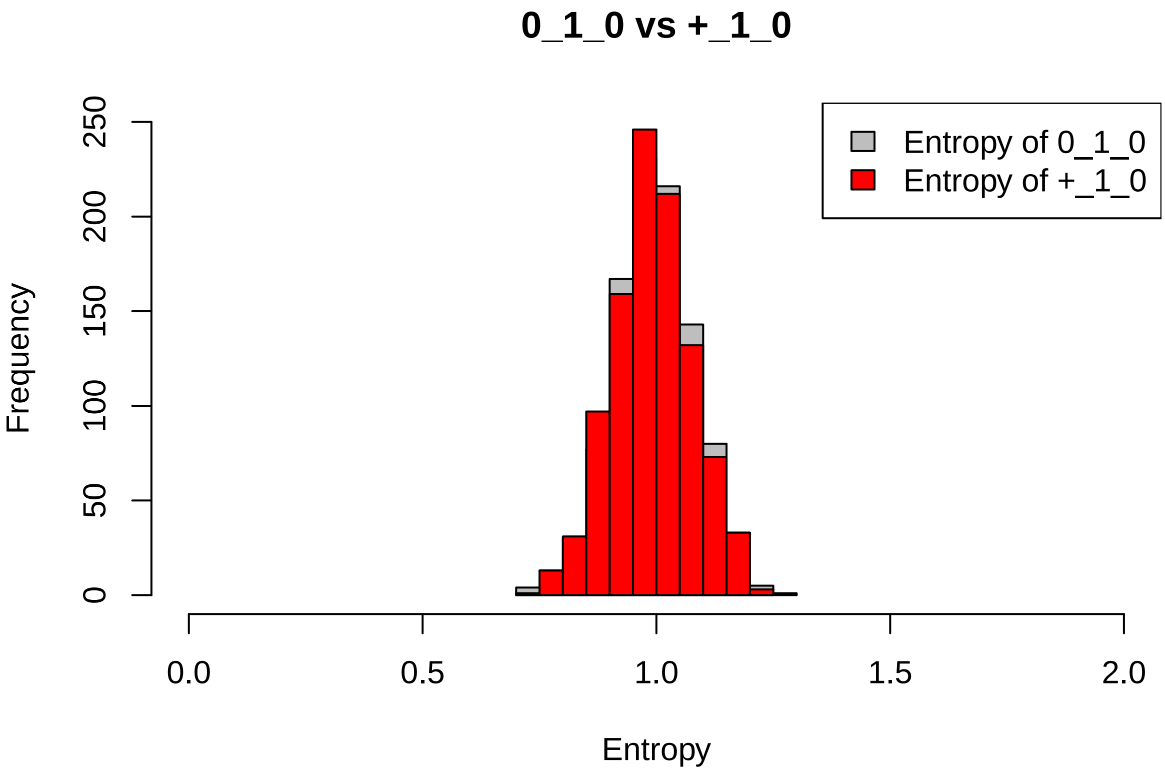
```
## Warning: package 'DescTools' was built under R version 4.2.2
```



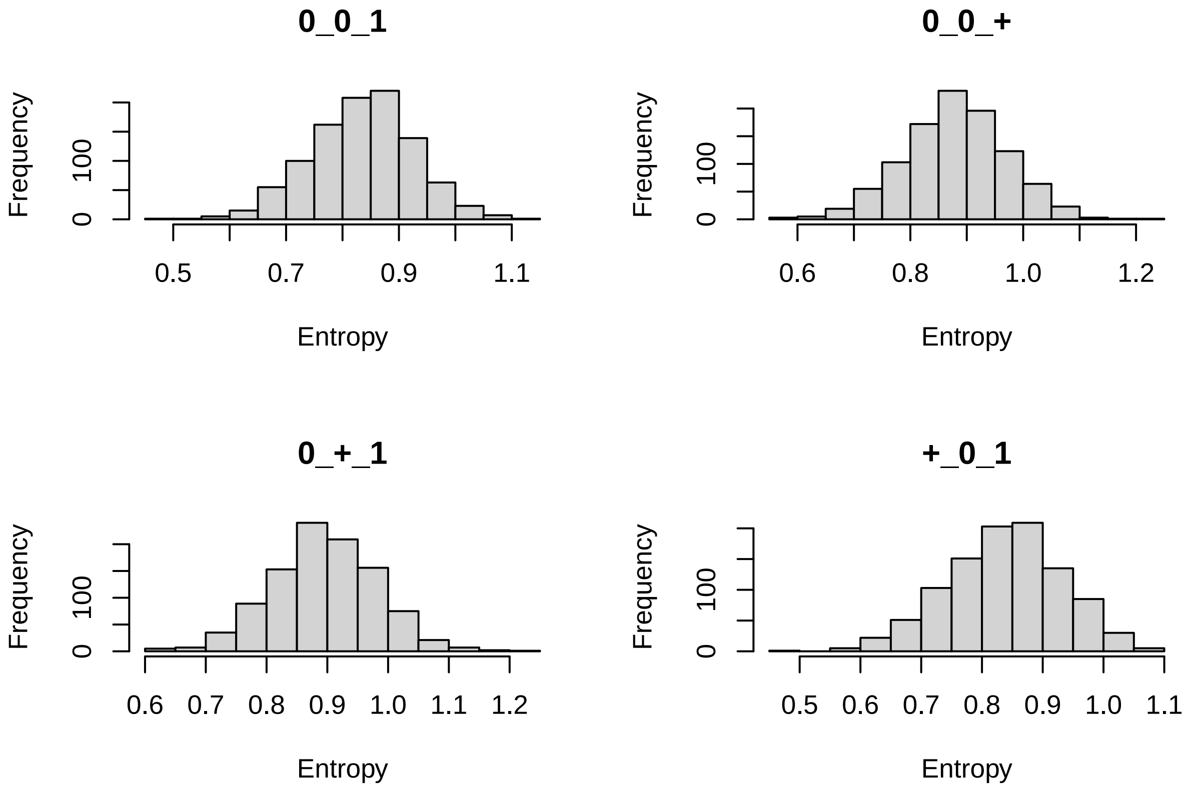
Looking at these graphs below, it seems that there is a little shift happening when HighChol or HighBP is missing, but not much. Thus, missing 1 bi-variable has little effect on 0_1_0. Since the graphs mostly overlap the origin, there is no interaction effect on the BMI distribution.


0_1_0 vs 0_1_+
Entropy of 0_1_0
Entropy of 0_1_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

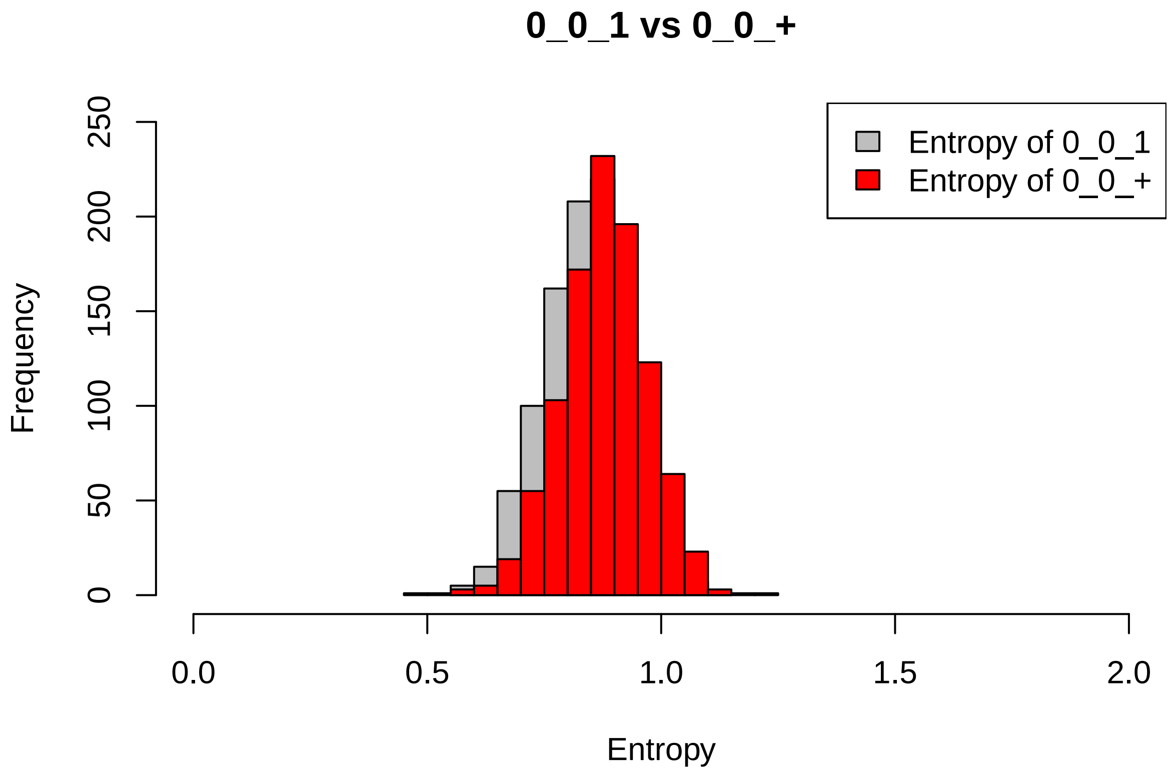

0_1_0 vs 0_+_0
Entropy of 0_1_0
Entropy of 0_+_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

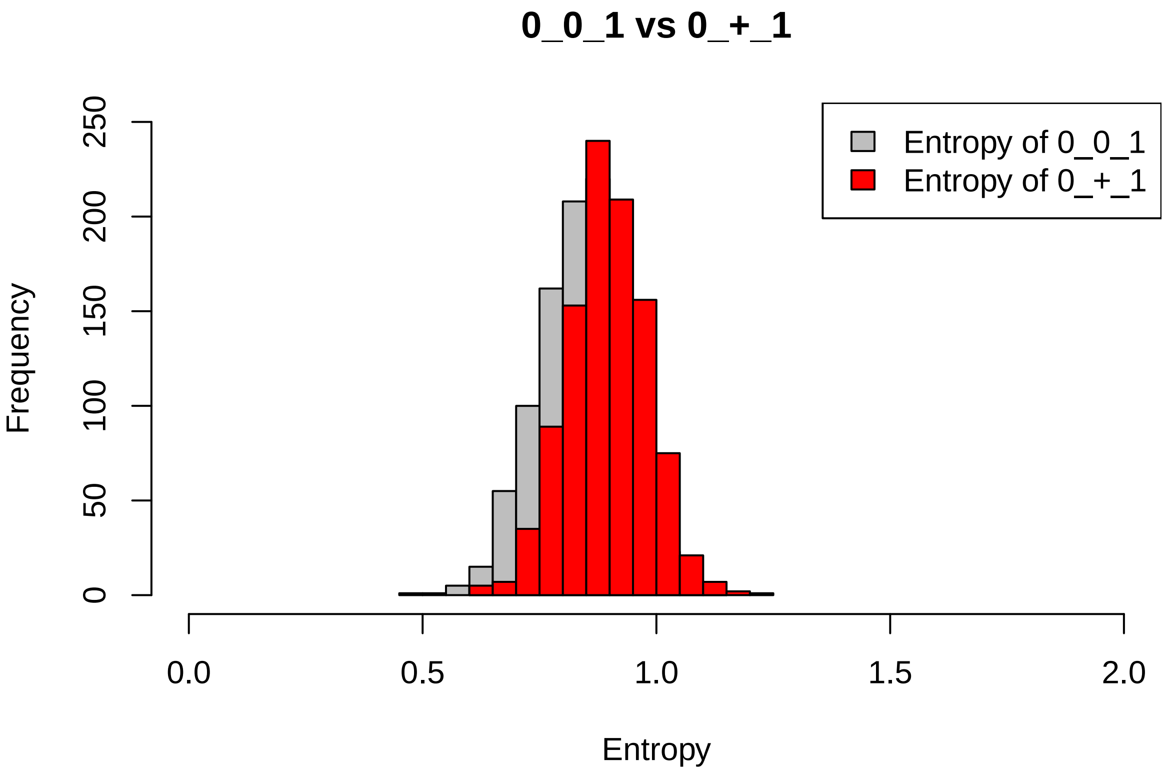

0_1_0 vs +_1_0
Entropy of 0_1_0
Entropy of +_1_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

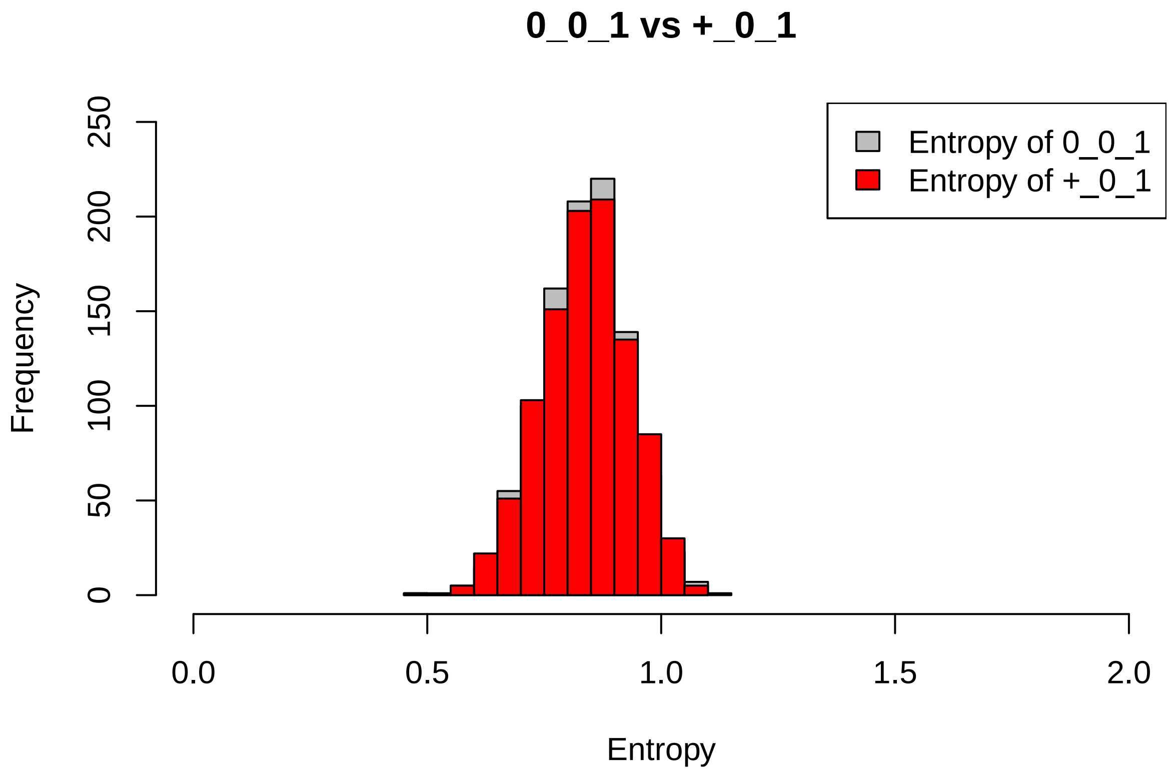
No HD __ No HighBP __ Yes HighChol



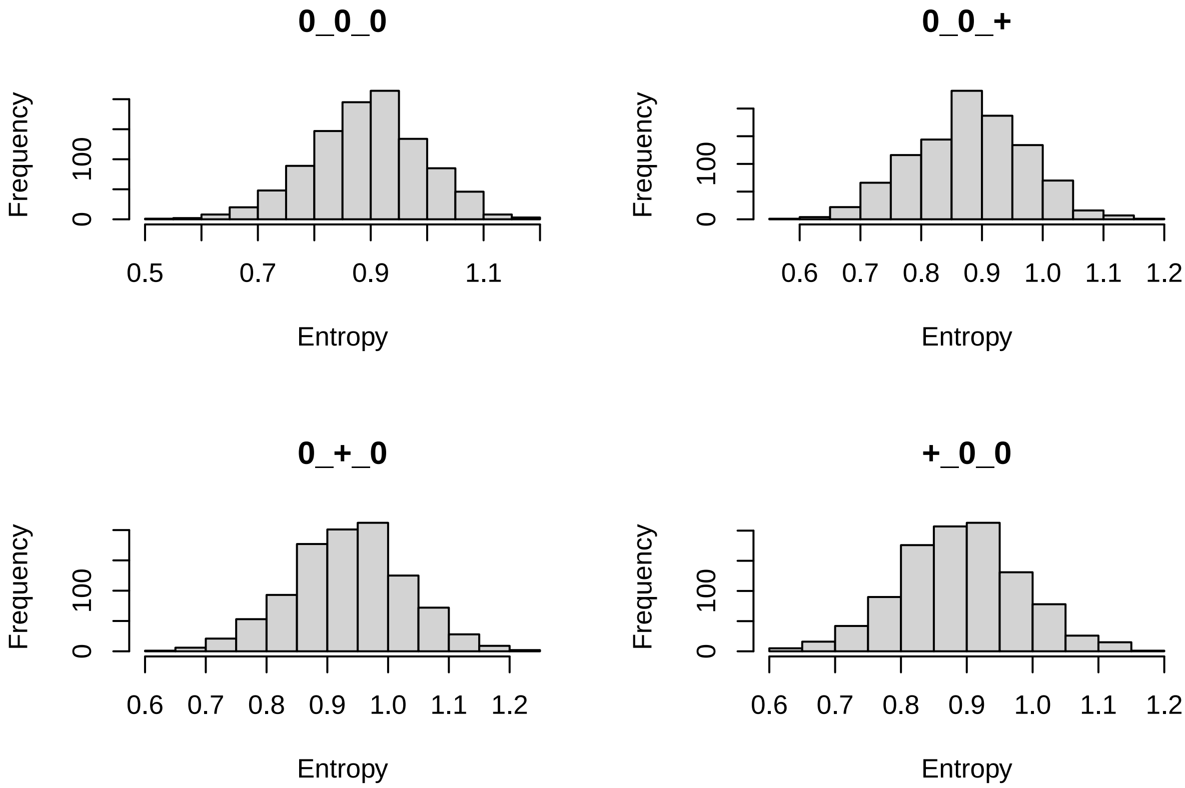
Looking at these graphs below, there is clearly a shift to the right when HighBP or HighChol is missing, which means the entropy level goes up and makes it less accurate. However, missing HD doesn't shift as much as the others. Since the graphs of missing HighBP or missing HighChol doesn't overlap the origin, there is an interaction effect on the BMI distribution.


0_0_1 vs 0_0_+
Frequency
0
50
100
150
200
250
Entropy of 0_0_1
Entropy of 0_0_+
0.0
0.5
1.0
1.5
2.0
Entropy

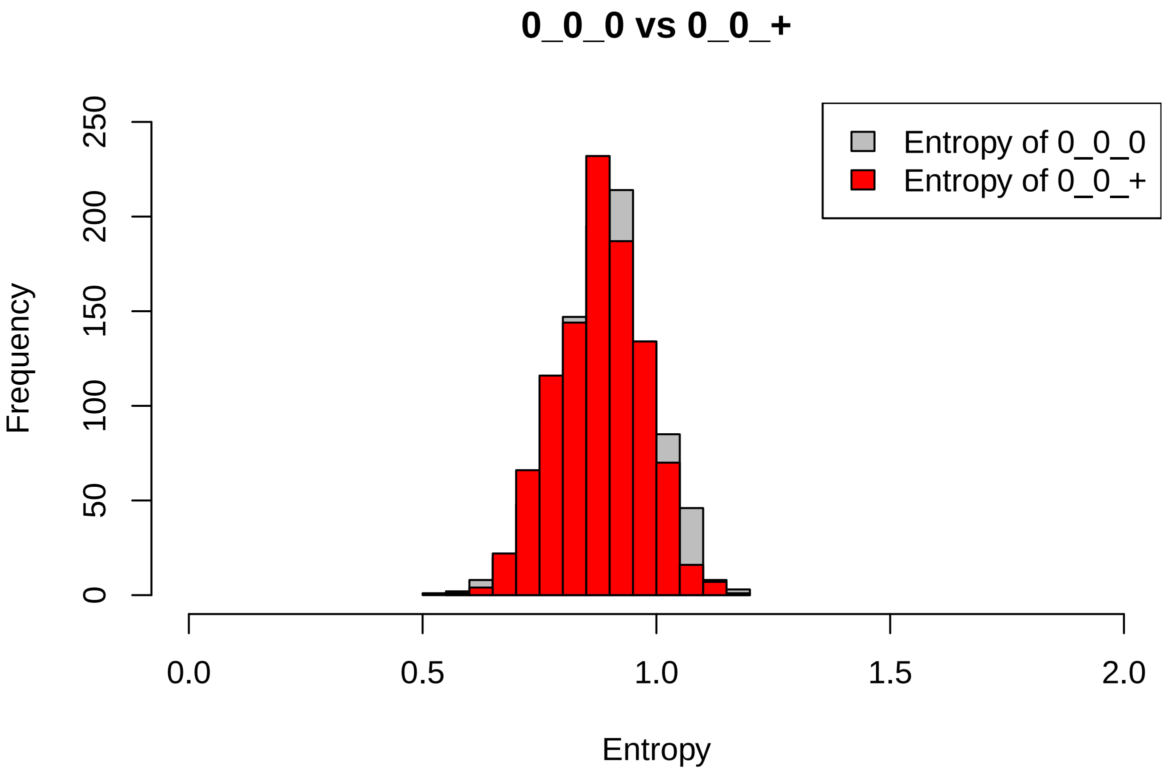

0_0_1 vs 0_+_1
Entropy of 0_0_1
Entropy of 0_+_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

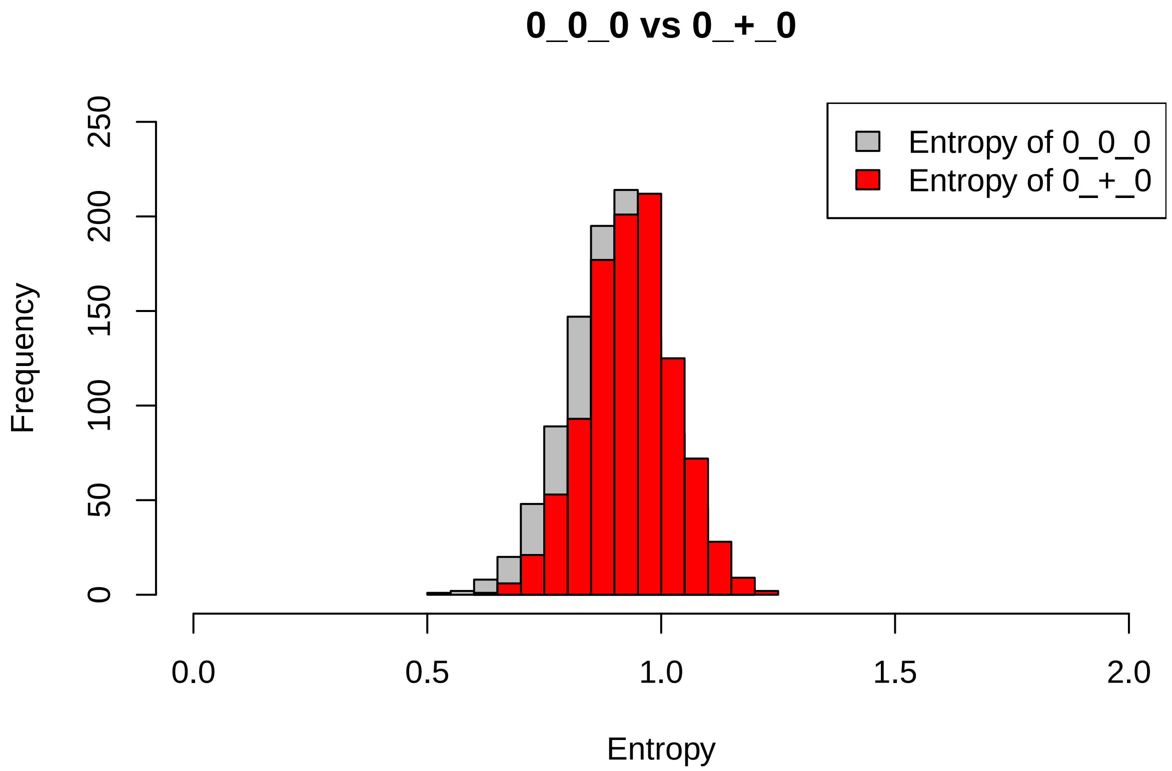

0_0_1 vs +_0_1
Entropy of 0_0_1
Entropy of +_0_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

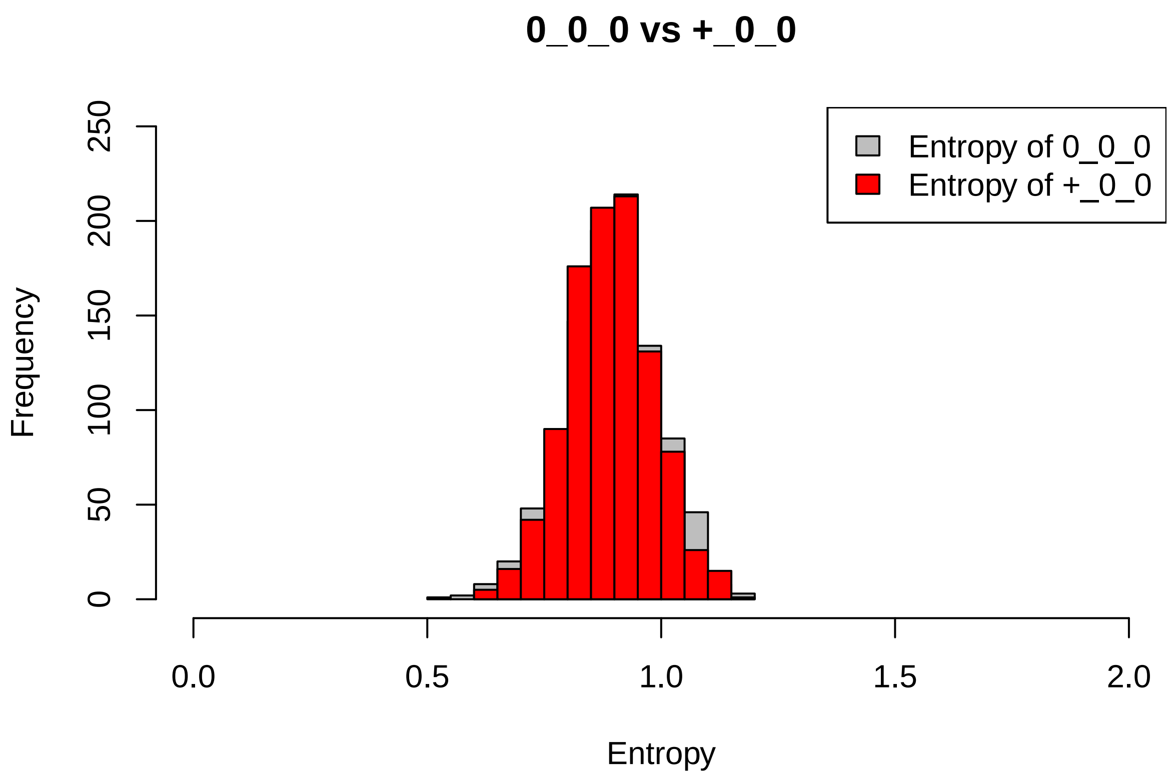
No HD __ No HighBP __ No HighChol



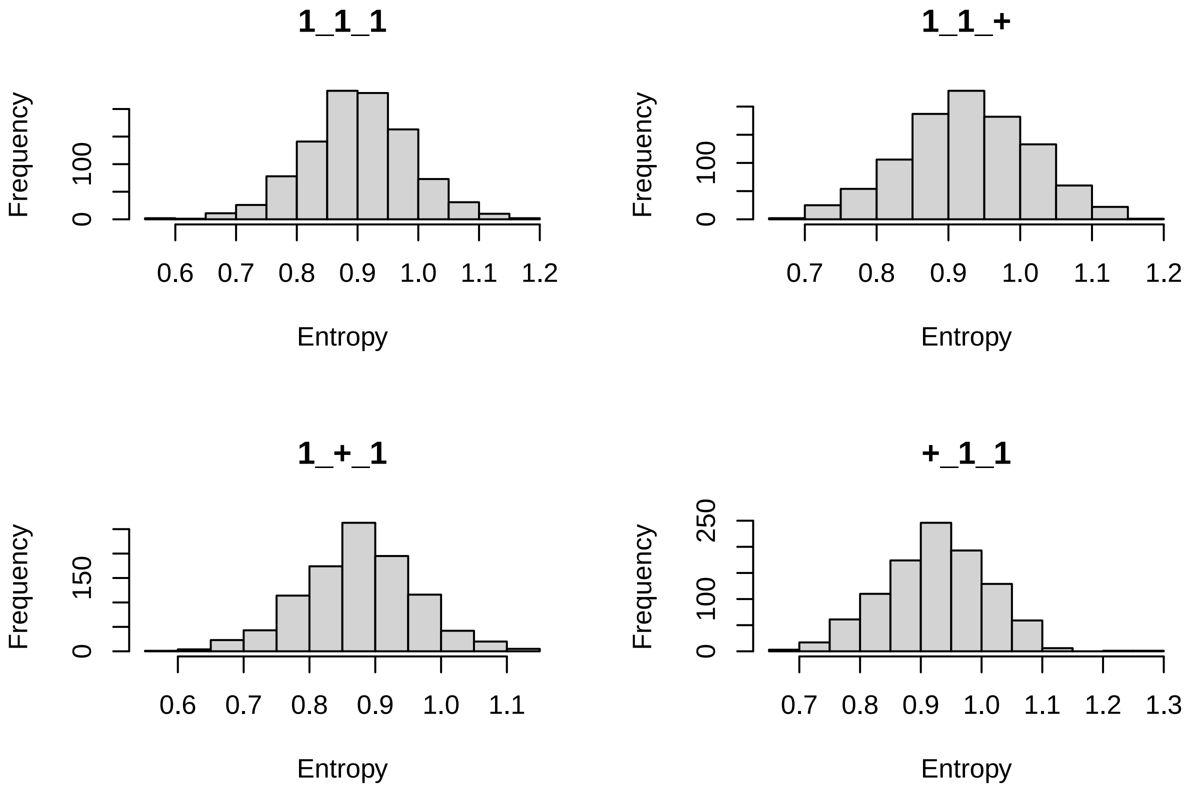
Looking at these graphs below, there is not much change going when missing one bi-variable. There is a little shift when missing HighBP, but not much. Since the graphs mostly overlap to the origin, there is no interaction effect on the BMI distribution.


0_0_0 vs 0_0_+
Entropy of 0_0_0
Entropy of 0_0_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

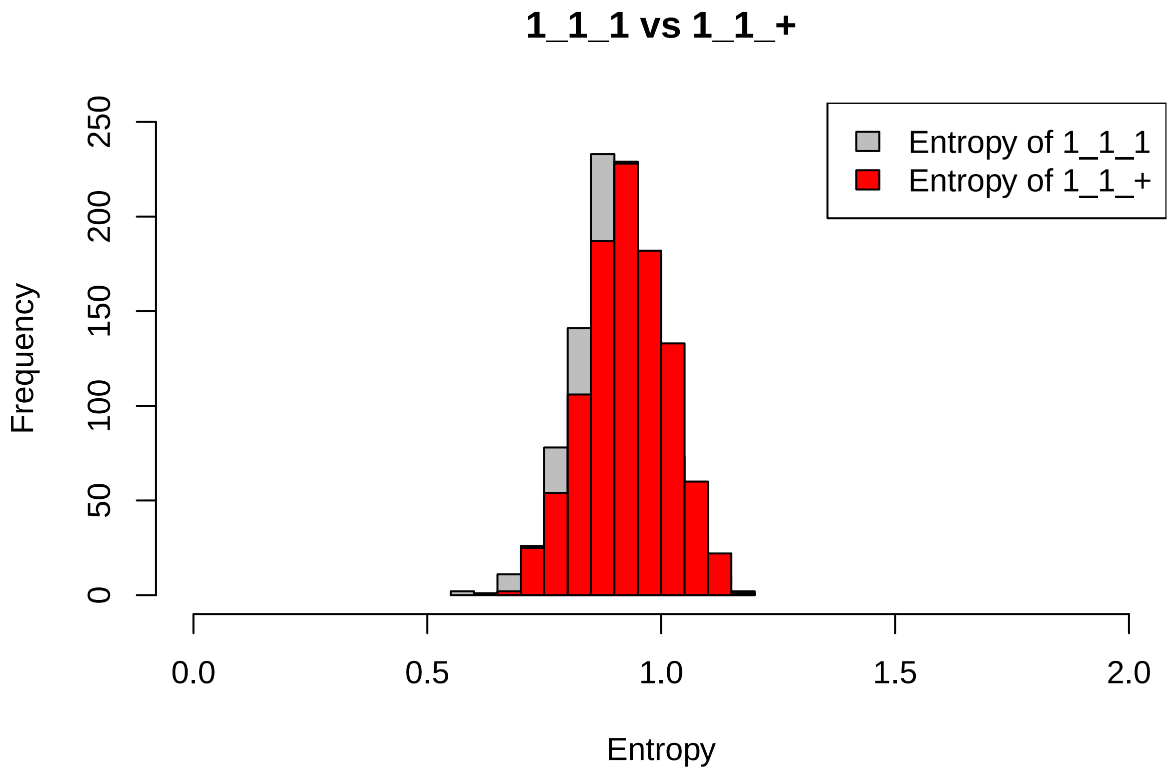

0_0_0 vs 0_+_0
Entropy of 0_0_0
Entropy of 0_+_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

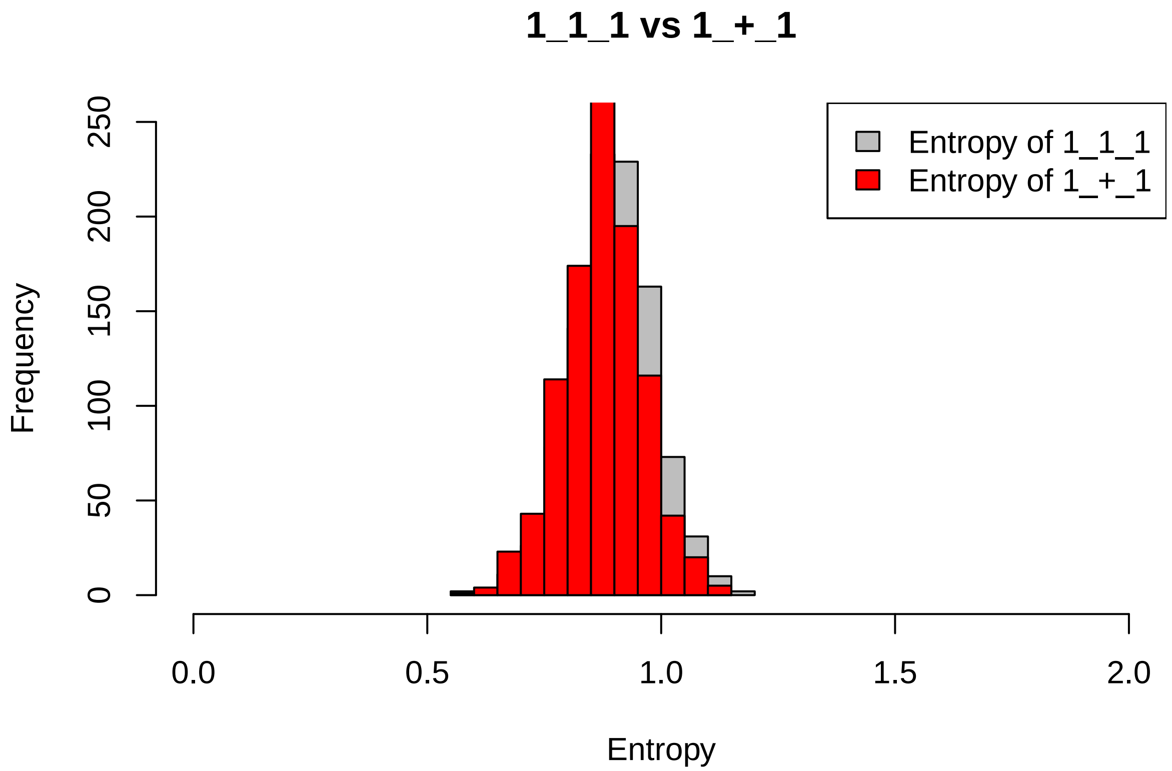

0_0_0 vs +_0_0
Entropy of 0_0_0
Entropy of +_0_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

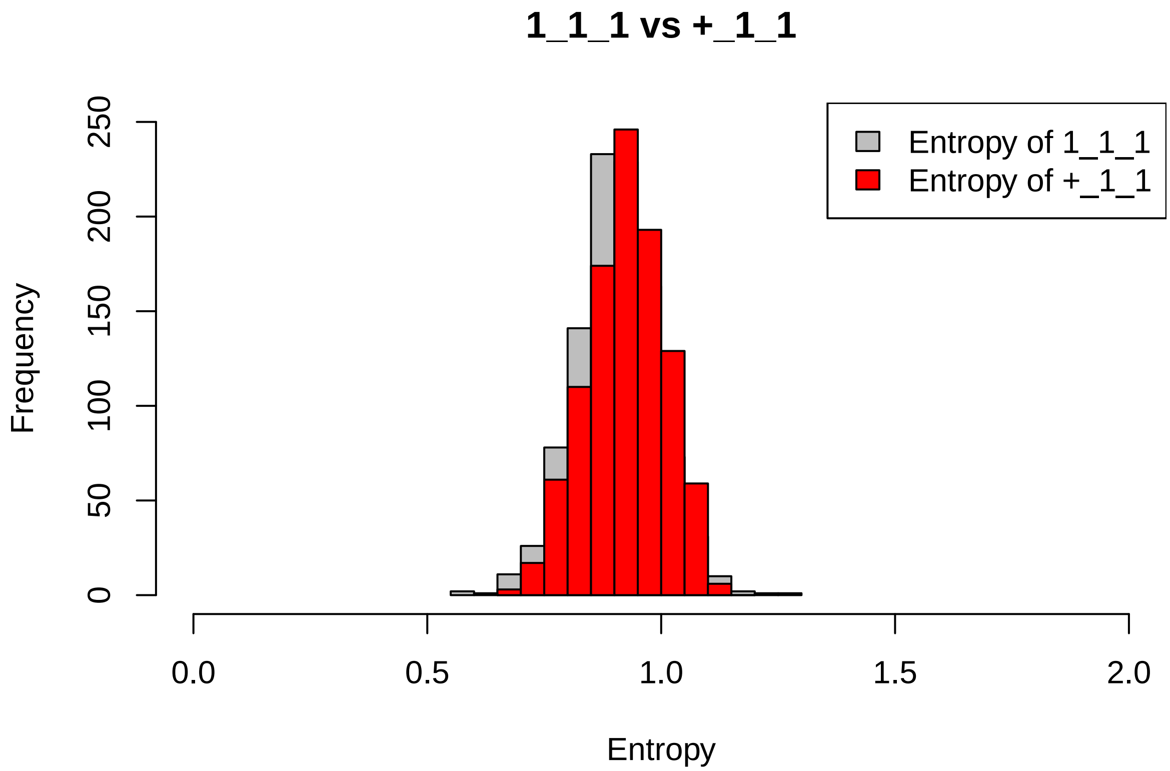
Yes HD __ Yes HighBP __ Yes HighChol



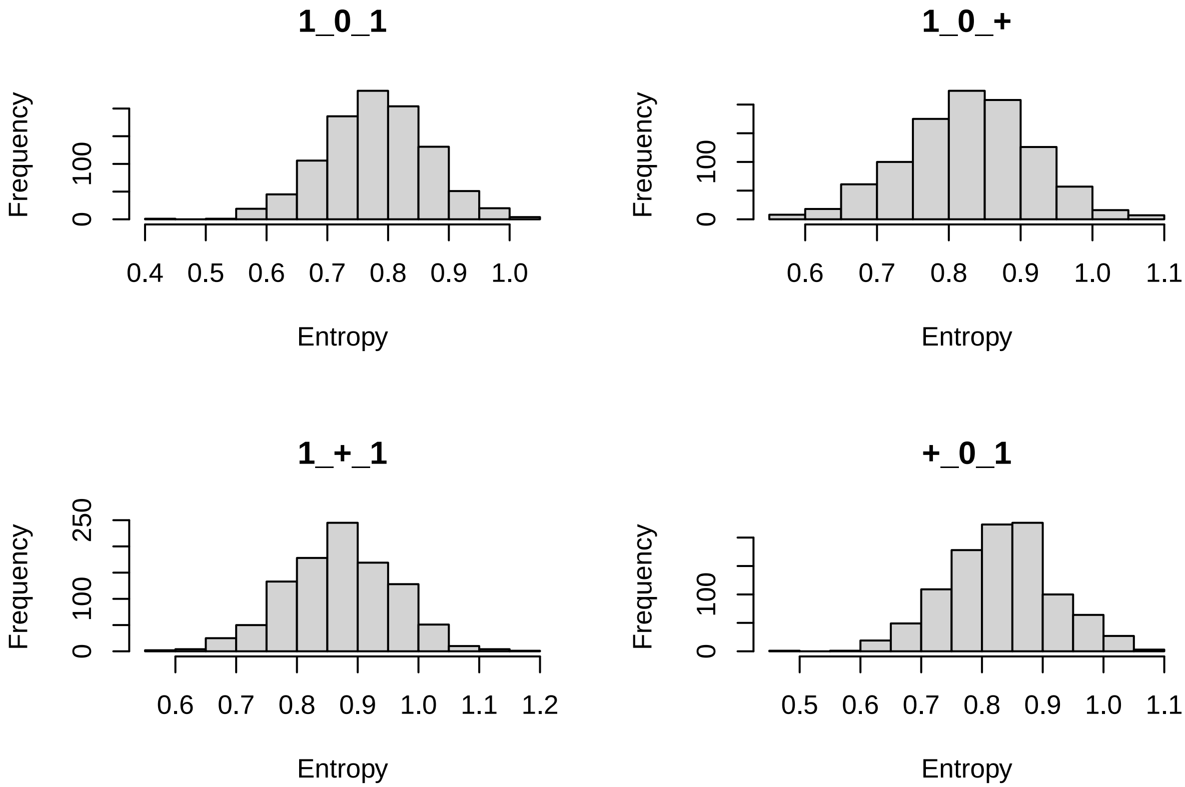
Looking at these graphs below, there is not much change happening. There is a little shift to the right when missing HighChol, and a little shift to the left when missing HighBP. Since the graphs mostly overlap the origin, there is no interaction effect on the BMI distribution.


1_1_1 vs 1_1_+
Entropy of 1_1_1
Entropy of 1_1_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

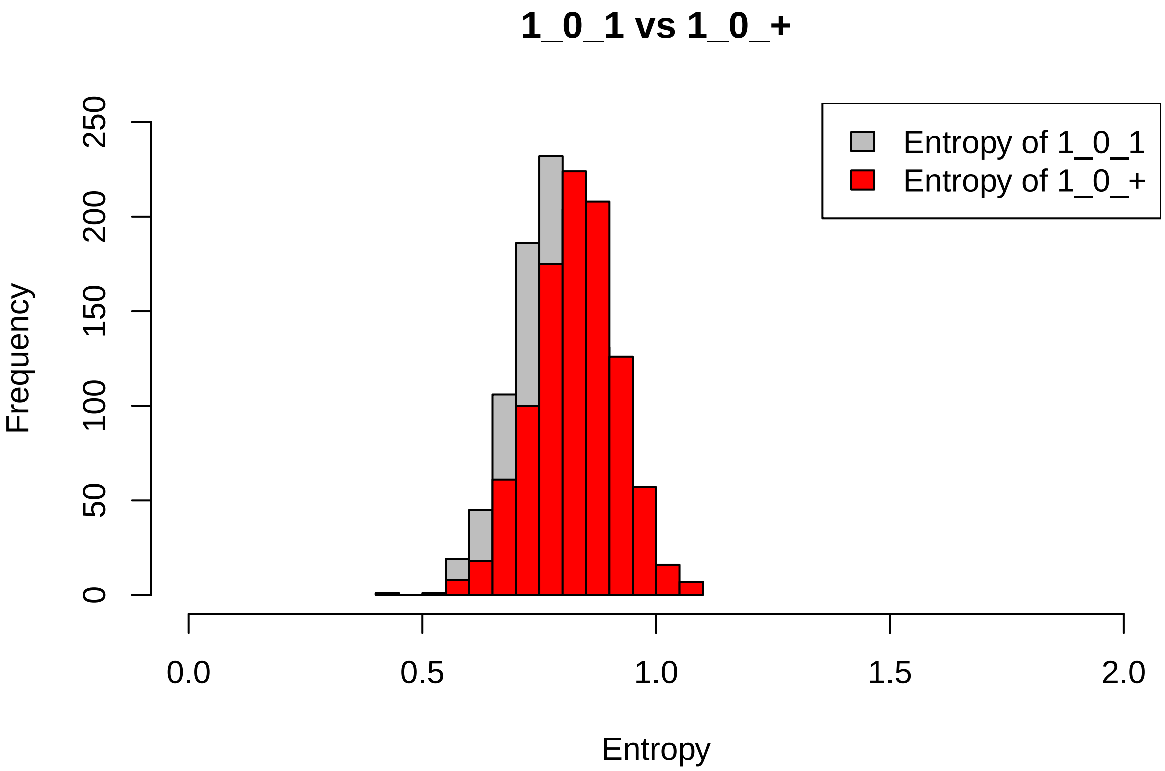

1_1_1 vs 1_+_1
Entropy of 1_1_1
Entropy of 1_+_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

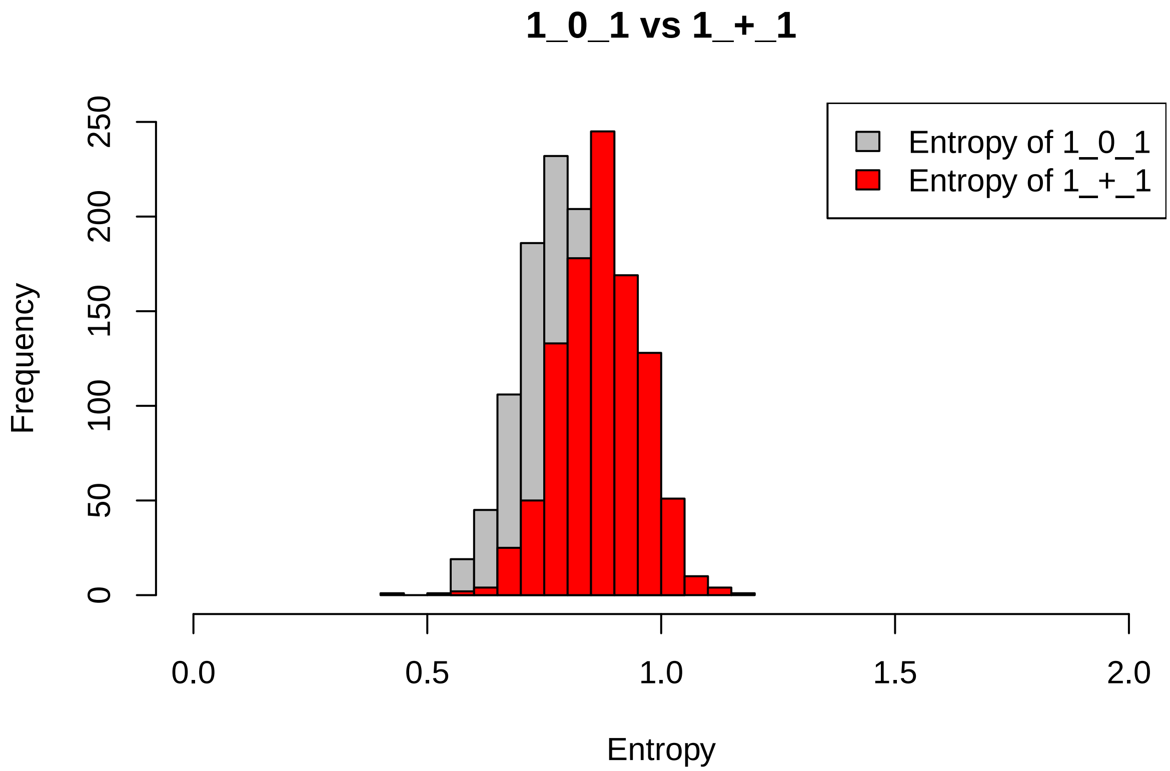

1_1_1 vs +_1_1
Entropy of 1_1_1
Entropy of +_1_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

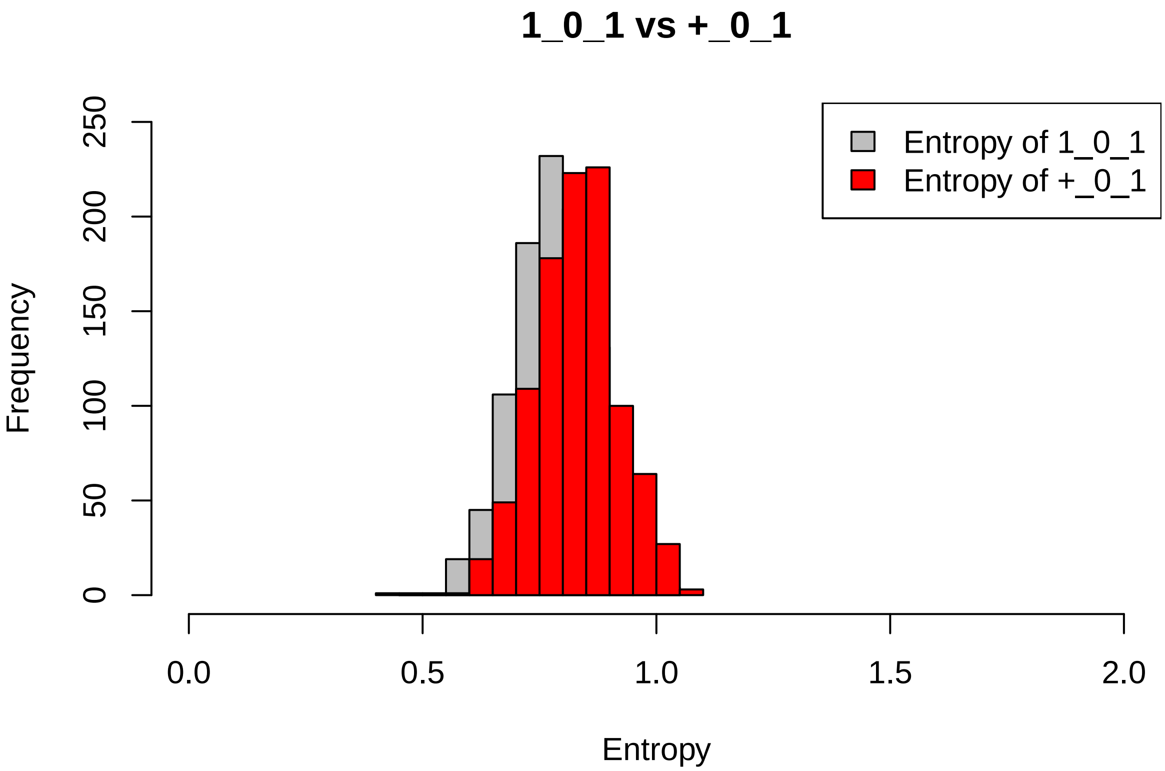
Yes HD __ No HighBP __ Yes HighChol



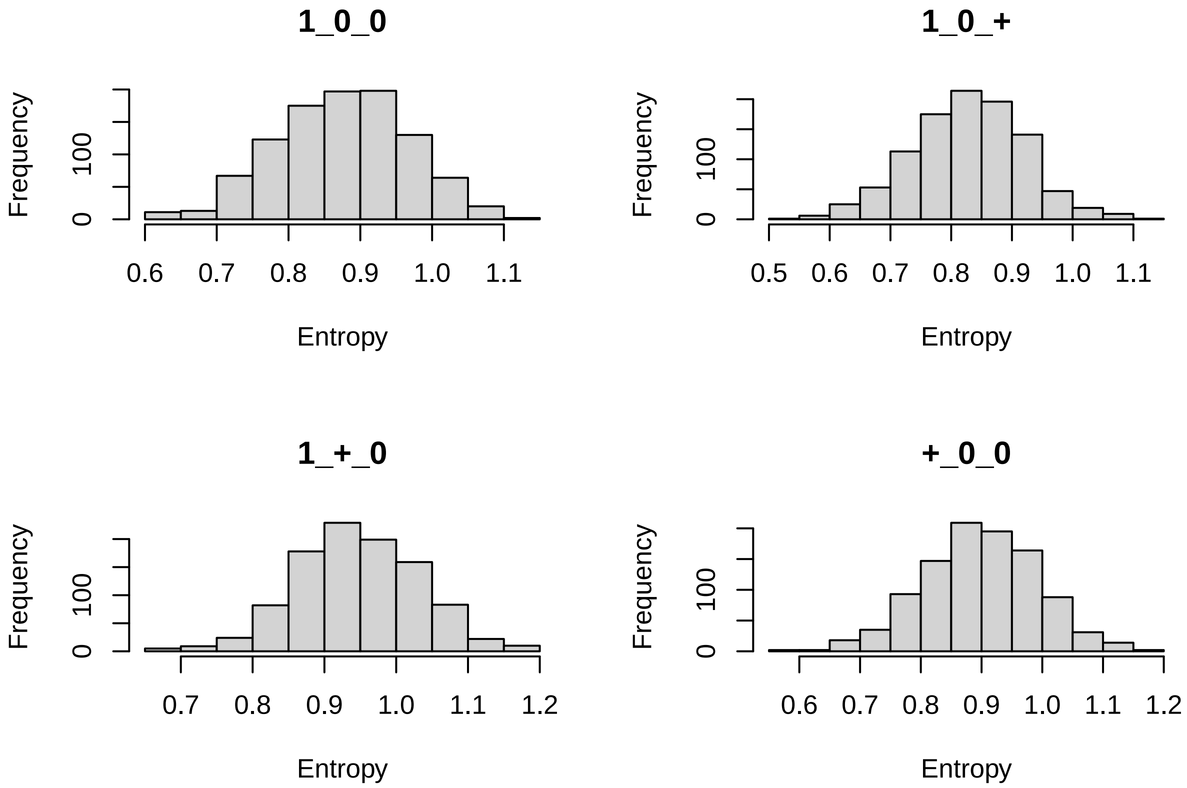
Looking at these graphs below, there is clearly a strong shift to the right when HighBP or HeartDisease or HighChol is missing, which means the entropy level goes up and makes it less accurate. Since all 3 graphs don't overlap the origin, there is an interaction effect on the BMI distribution.


1_0_1 vs 1_0_+
Entropy of 1_0_1
Entropy of 1_0_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

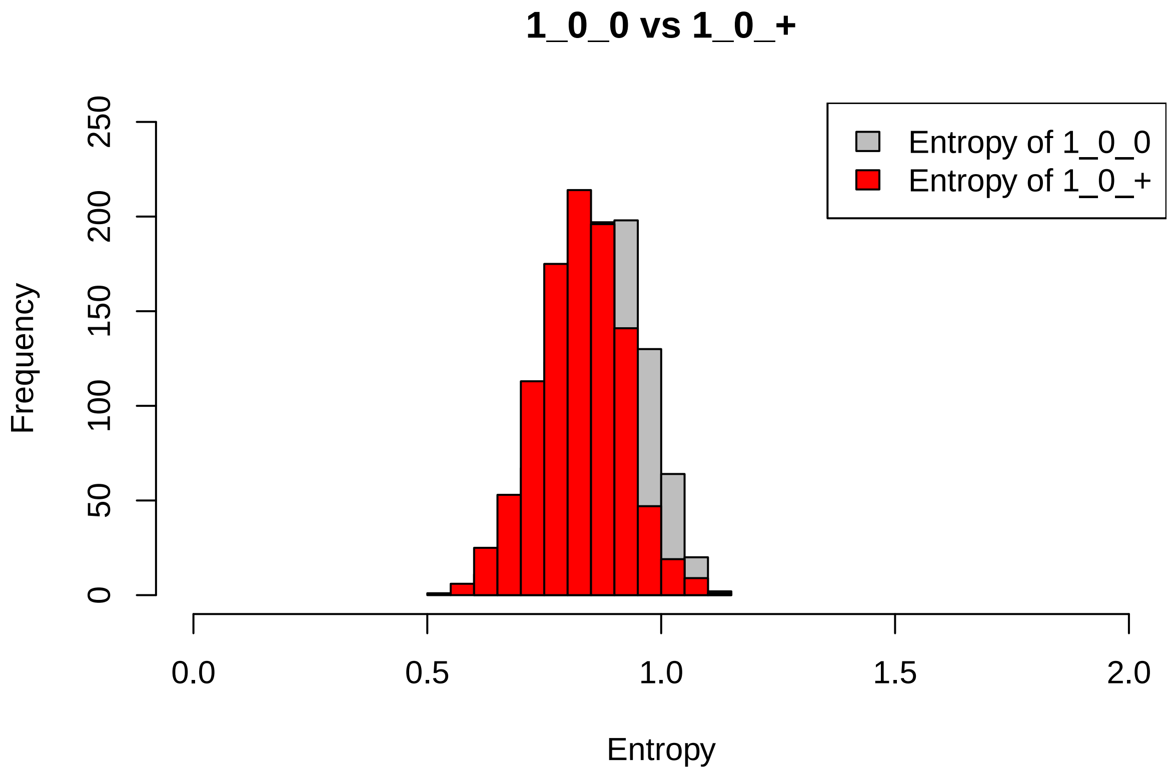

1_0_1 vs 1_+_1
Entropy of 1_0_1
Entropy of 1_+_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

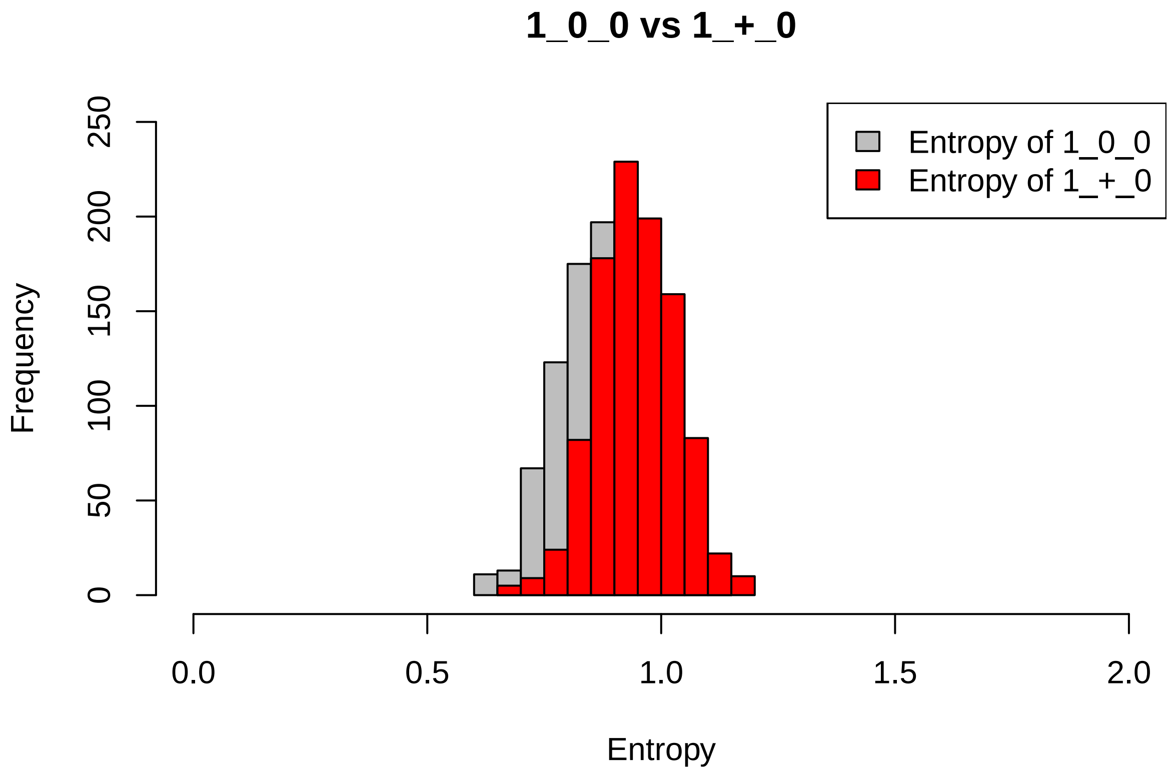

1_0_1 vs +_0_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 1_0_1
Entropy of +_0_1

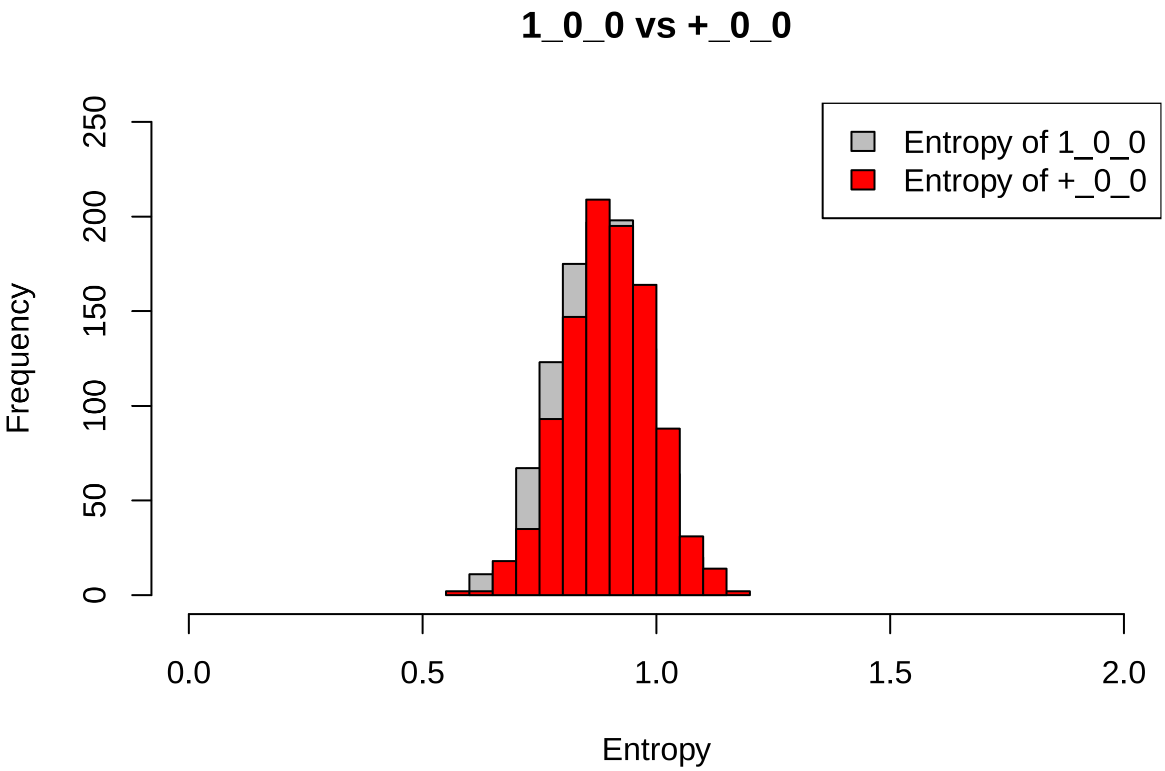
Yes HD __ No HighBP __ No HighChol



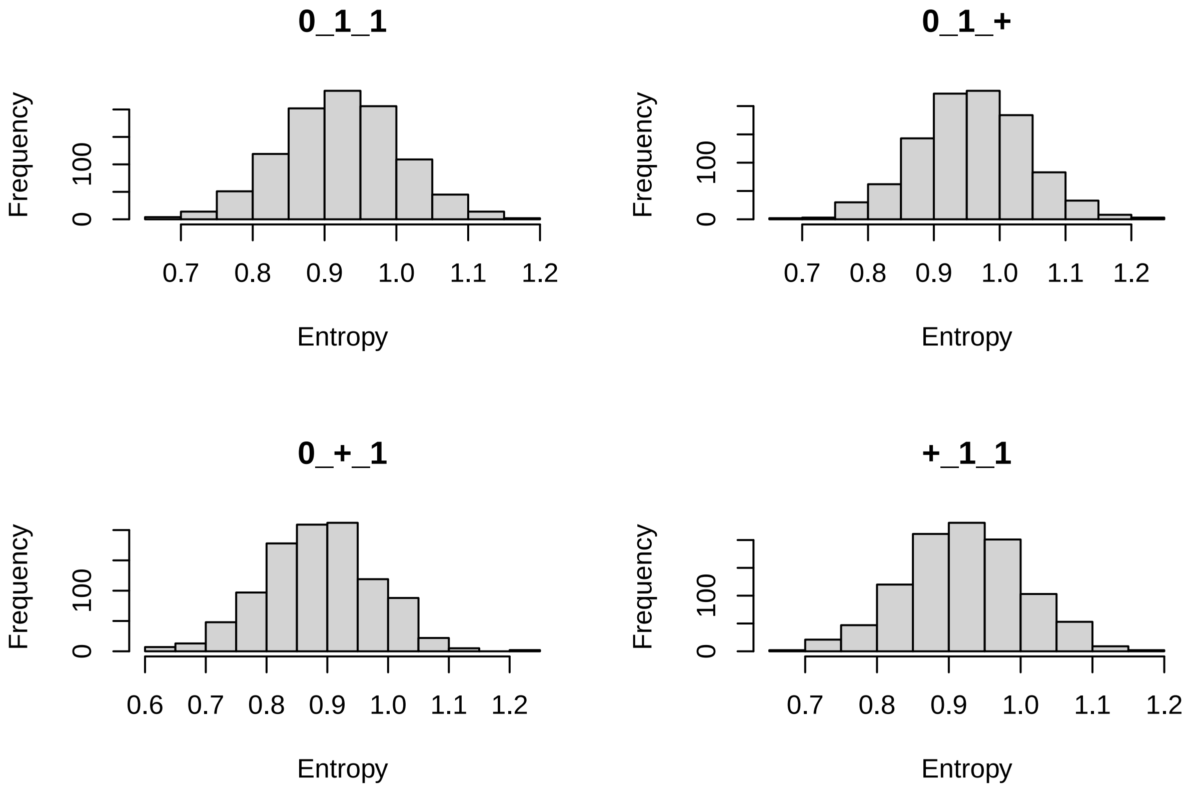
Looking at these graphs below, there is not much change happening. Thus, having missing 1 bi-variable have little effect on 1_0_0. Since the graphs mostly overlap the origin, there is no interaction effect on the BMI distribution.


1_0_0 vs 1_0_+
Entropy of 1_0_0
Entropy of 1_0_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

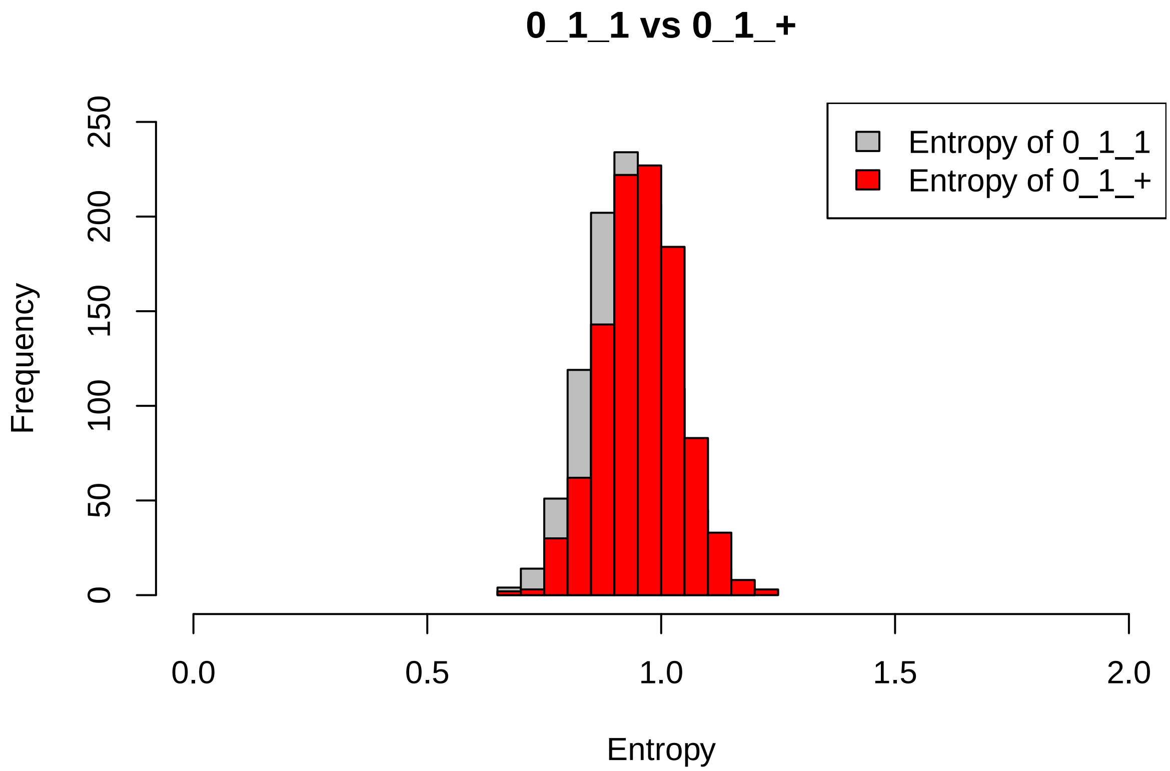

1_0_0 vs 1_+_0
Entropy of 1_0_0
Entropy of 1_+_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

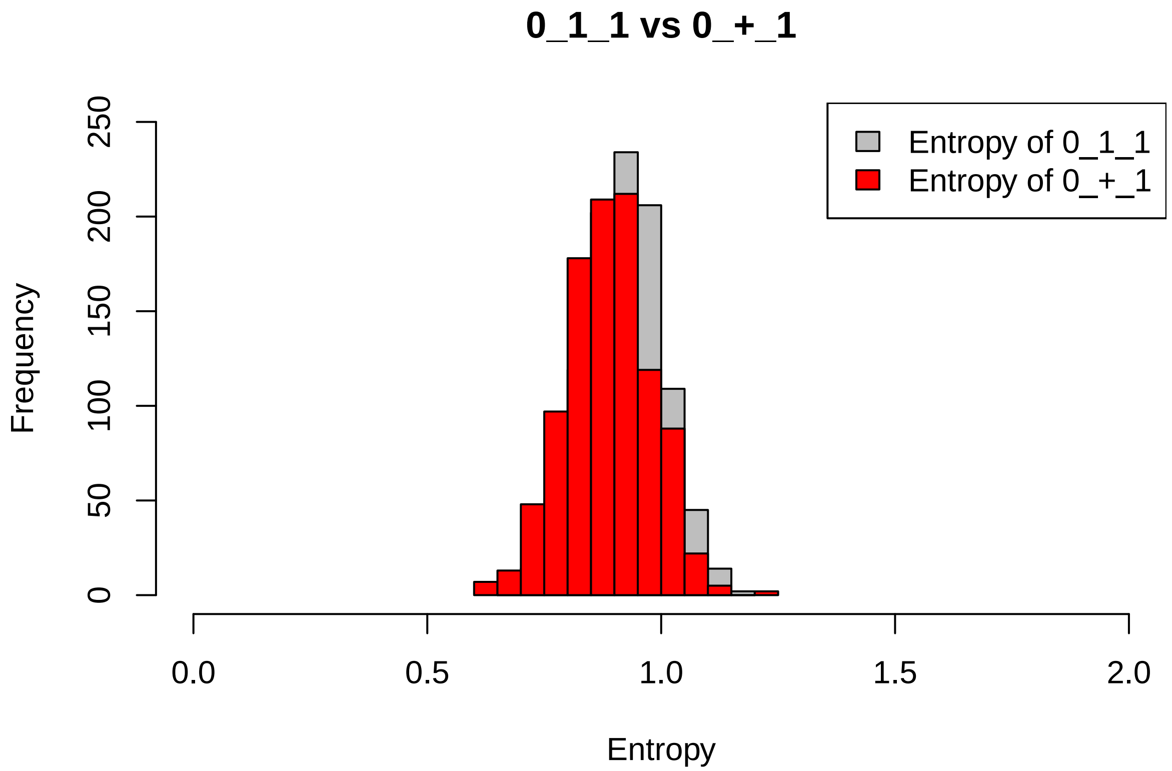

1_0_0 vs +_0_0
Entropy of 1_0_0
Entropy of +_0_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

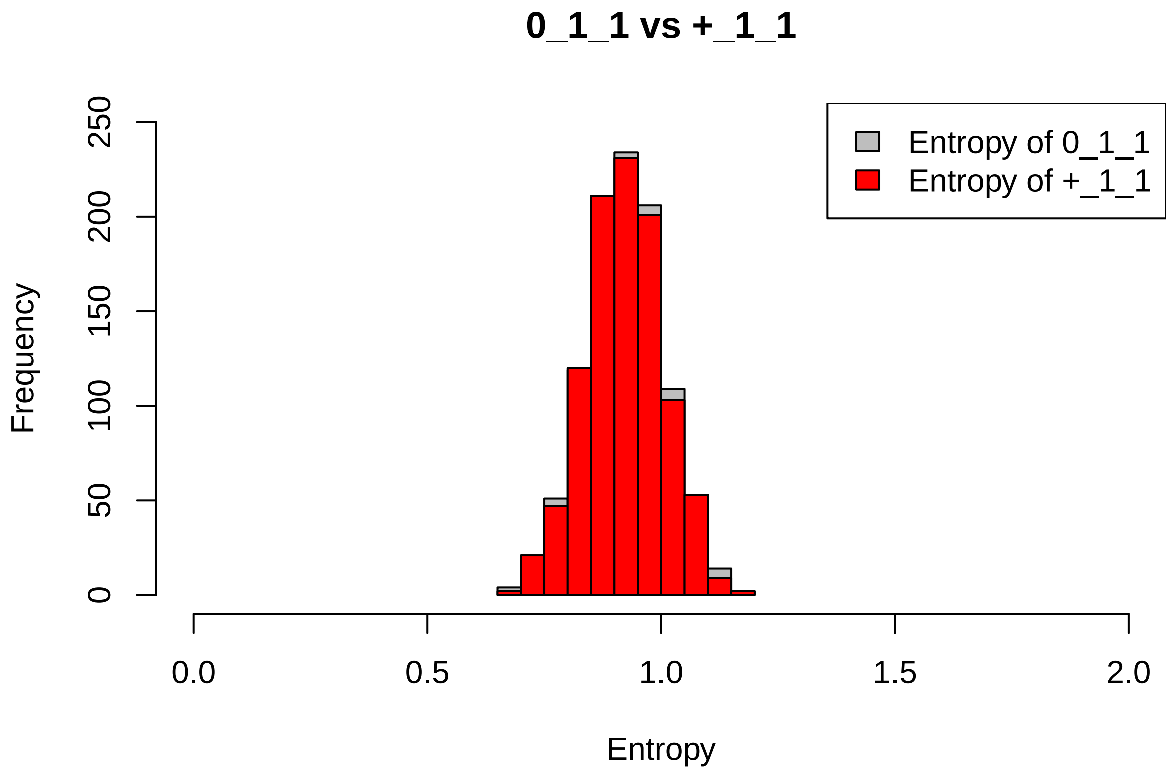
No HD __ Yes HighBP __ Yes HighChol



Looking at these graphs below, there is not much change happening. Thus, missing one bi-variable has little effect on 0_1_1. Since the graphs mostly overlap the origin, there is no interaction effect on the BMI distribution.

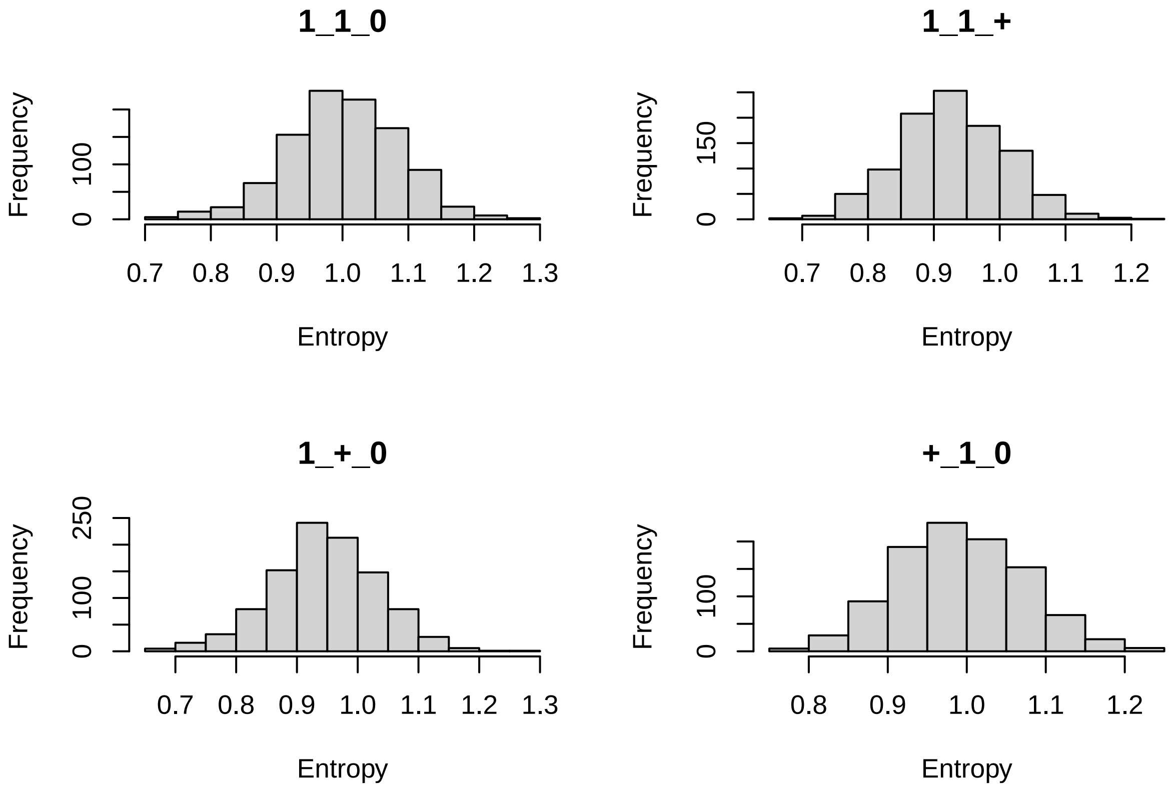

0_1_1 vs 0_1_+
Entropy of 0_1_1
Entropy of 0_1_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy


0_1_1 vs 0_+_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 0_1_1
Entropy of 0_+_1

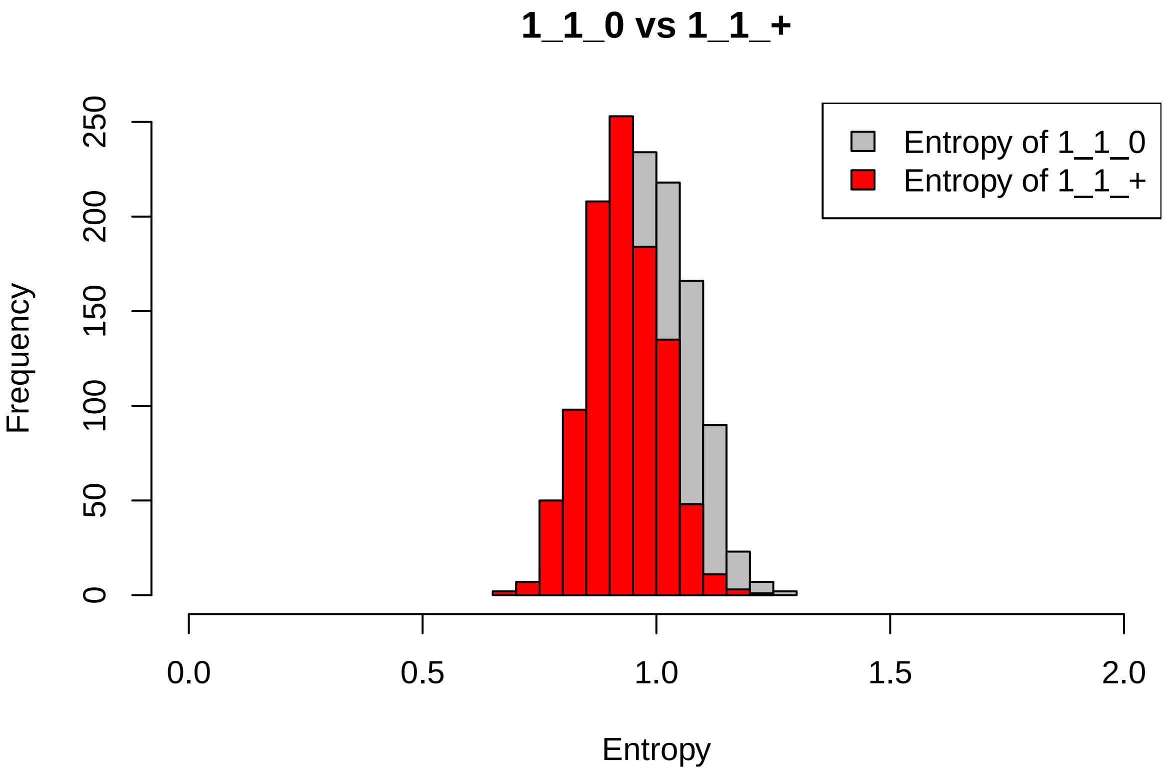


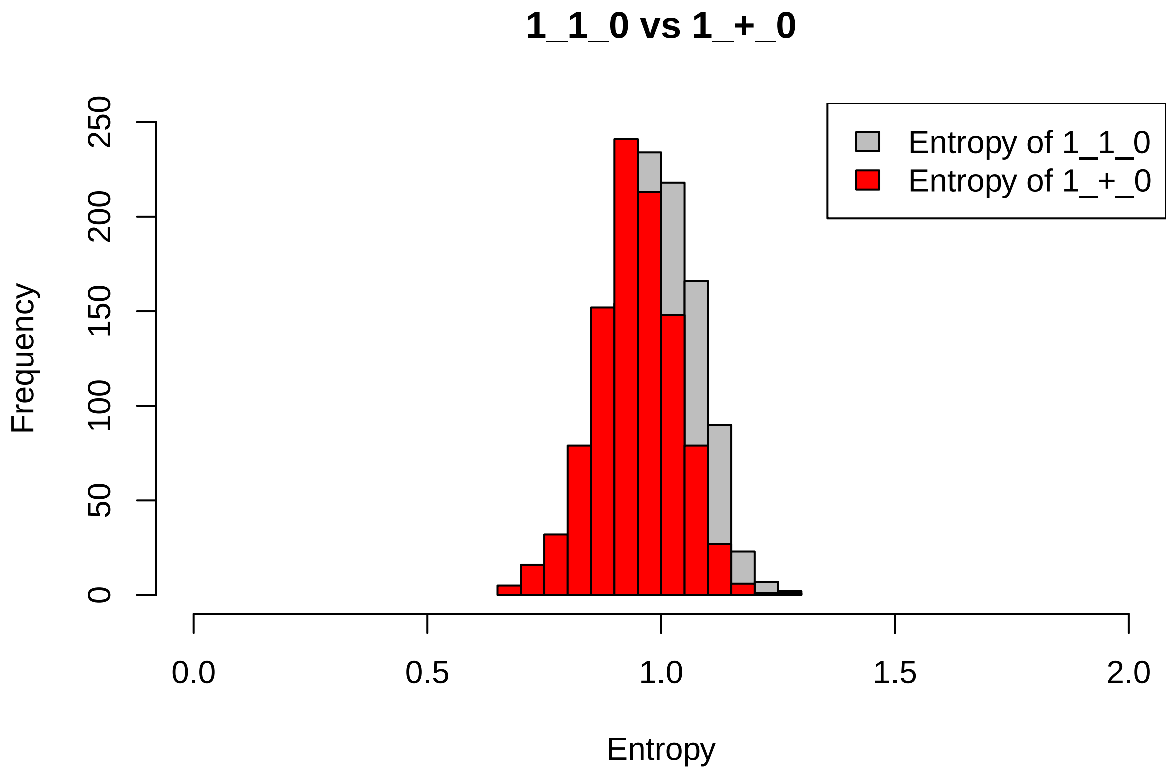
0_1_1 vs +_1_1
Frequency
0
50
100
150
200
250
Entropy of 0_1_1
Entropy of +_1_1
0.0
0.5
1.0
1.5
2.0
Entropy

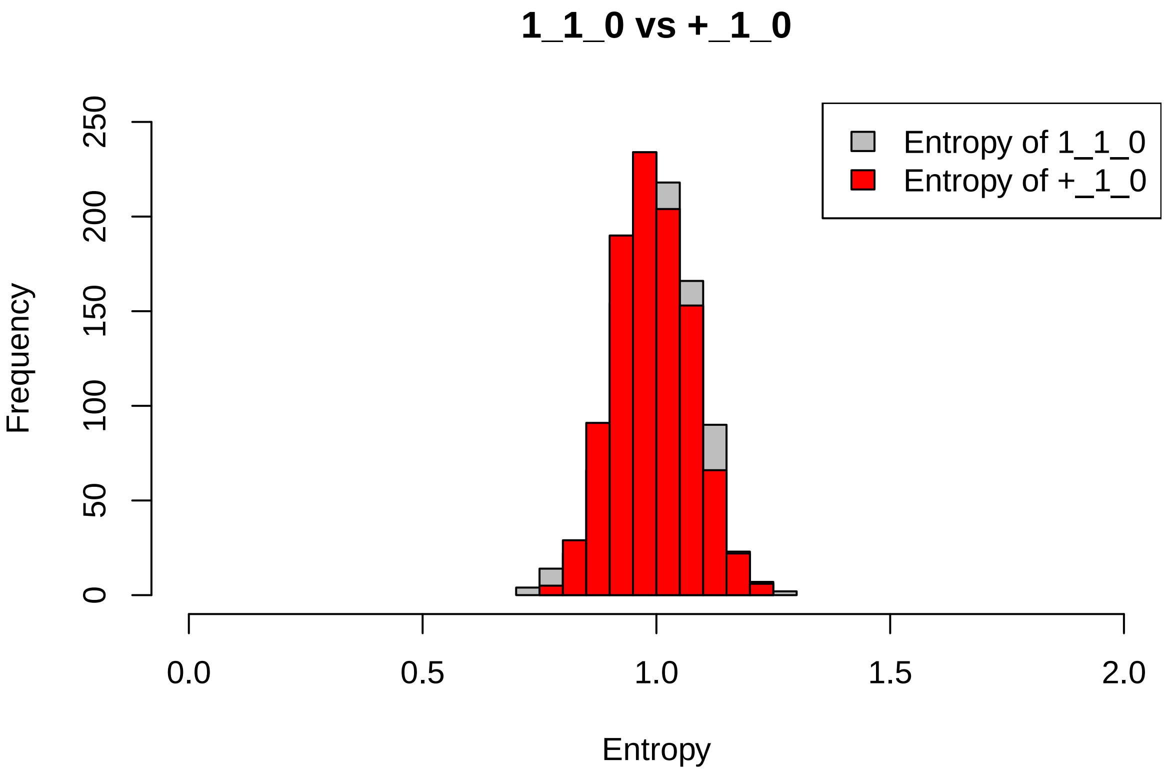
Yes HD __ Yes HighBP __ No HighChol



Looking at these graphs below, there is clearly a shift to the left when HighBP or HighChol is missing, which means the entropy level goes down and makes it more accurate. However, missing HD doesn't shift as much as the others. Since the graphs of missing HighBP or missing HighChol doesn't overlap the origin, there is an interaction effect on the BMI distribution.


1_1_0 vs 1_1_+
Entropy of 1_1_0
Entropy of 1_1_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy


1_1_0 vs 1_+_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 1_1_0
Entropy of 1_+_0


1_1_0 vs +_1_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 1_1_0
Entropy of +_1_0

3/ Investigating the 3-way interacting effects in GenHealth_3 sub-dataset:

Creating contigency tables based on the combination of these three selected binary variable

```
## Warning: package 'iNZightTools' was built under R version 4.2.2
```

```
##
## Attaching package: 'iNZightTools'
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

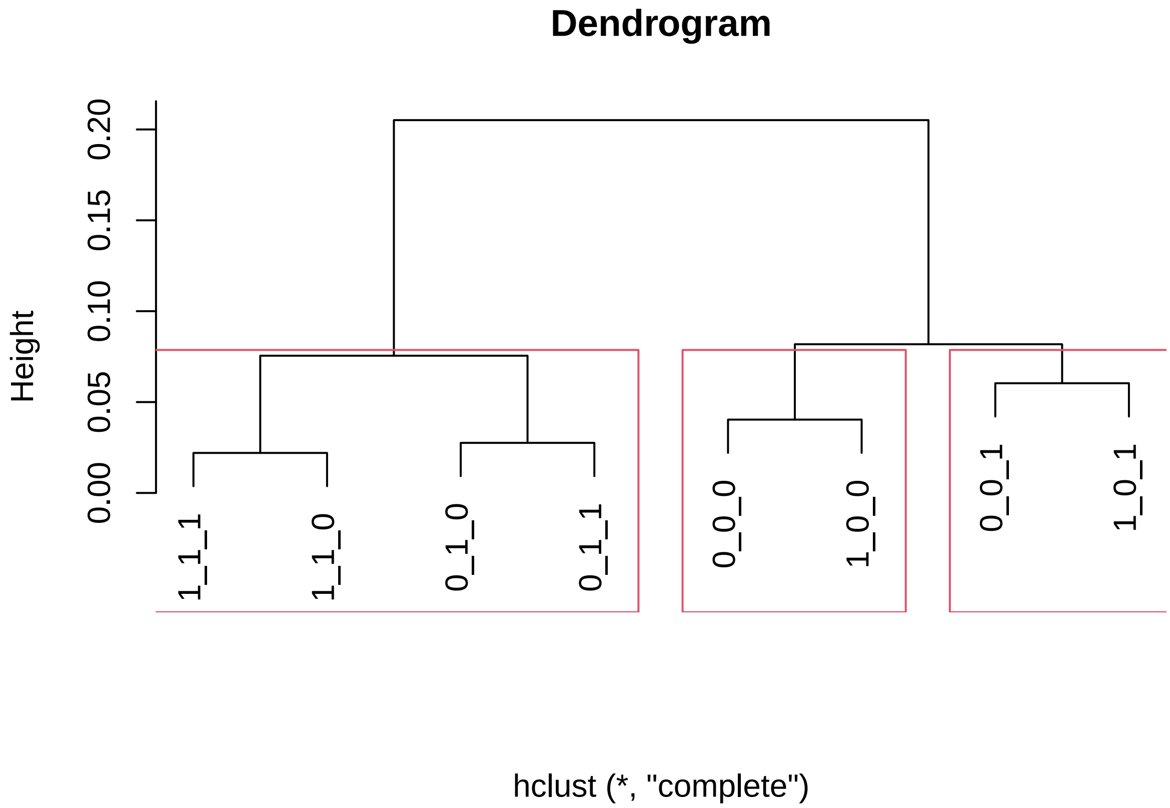
```
##                1         2         3          4           5            6
## 0_0_0 0.06289390 0.5854701 0.2699214 0.06569973 0.011870845 0.0016835017
## 0_1_0 0.02959045 0.4600815 0.3643056 0.11736116 0.022871846 0.0031448789
## 0_1_1 0.01967265 0.4746616 0.3798132 0.10450110 0.016420103 0.0025705592
## 1_1_1 0.02242655 0.5229872 0.3639830 0.07804440 0.010989011 0.0008970621
## 0_0_1 0.03865196 0.5686615 0.3129506 0.06740207 0.009120125 0.0006948667
## 1_0_1 0.03187614 0.6211293 0.3023679 0.04098361 0.003642987 0.0000000000
## 1_0_0 0.07419018 0.6112853 0.2601881 0.03866249 0.009404389 0.0020898642
## 1_1_0 0.03642857 0.5114286 0.3542857 0.08571429 0.010000000 0.0014285714
## Total 0.03897100 0.5275890 0.3290723 0.08568860 0.014396002 0.0019564815
##                  7            8            9           10
## 0_0_0 0.0003021670 0.0009065009 0.0011223345 1.295001e-04
## 0_1_0 0.0007147452 0.0010721178 0.0007862197 7.147452e-05
## 0_1_1 0.0004196831 0.0009967475 0.0007344455 2.098416e-04
## 1_1_1 0.0000000000 0.0006727966 0.0000000000 0.000000e+00
## 0_0_1 0.0001737167 0.0008685833 0.0013897333 8.685833e-05
## 1_0_1 0.0000000000 0.0000000000 0.0000000000 0.000000e+00
## 1_0_0 0.0000000000 0.0020898642 0.0010449321 1.044932e-03
## 1_1_0 0.0000000000 0.0000000000 0.0000000000 7.142857e-04
## Total 0.0003569257 0.0009253629 0.0008989239 1.454142e-04
```

```
##                1         2         3          4           5           6
## 0_0   0.05484587 0.5798898 0.2842066 0.06626489 0.010957640 0.001355287
## 0_1   0.02387075 0.4684900 0.3732490 0.10994463 0.019151060 0.002813663
## 1_1   0.02577232 0.5202253 0.3616658 0.07987711 0.010752688 0.001024066
## 1_0   0.05158151 0.6165450 0.2827251 0.03990268 0.006326034 0.000973236
## Total 0.03897100 0.5275890 0.3290723 0.08568860 0.014396002 0.001956482
##                  7            8            9           10
## 0_0   0.0002595231 0.0008939127 0.0012111076 0.0001153436
## 0_1   0.0005445799 0.0010286510 0.0007563610 0.0001512722
## 1_1   0.0000000000 0.0005120328 0.0000000000 0.0001706776
## 1_0   0.0000000000 0.0009732360 0.0004866180 0.0004866180
## Total 0.0003569257 0.0009253629 0.0008989239 0.0001454142
```

```
##                1         2         3          4           5            6
## 0_0   0.05035390 0.5382566 0.3054606 0.08515219 0.016013133 0.0022337648
## 0_1   0.02681930 0.5100572 0.3546361 0.09053148 0.013671300 0.0018642682
## 1_1   0.02429368 0.5423790 0.3518085 0.07072161 0.009537520 0.0007198128
## 1_0   0.05176071 0.5519728 0.3160798 0.06661010 0.009758167 0.0016970725
## Total 0.03897100 0.5275890 0.3290723 0.08568860 0.014396002 0.0019564815
##                  7            8            9           10
## 0_0   0.0004575181 0.0009688619 0.0009957747 0.0001076513
## 0_1   0.0003270646 0.0009484873 0.0009811938 0.0001635323
## 1_1   0.0000000000 0.0005398596 0.0000000000 0.0000000000
## 1_0   0.0000000000 0.0008485363 0.0004242681 0.0008485363
## Total 0.0003569257 0.0009253629 0.0008989239 0.0001454142
```

```
##                1         2         3          4           5            6
## 0_0   0.06334204 0.5864942 0.2695353 0.06462712 0.011772997 0.0016996228
## 1_0   0.03021246 0.4647521 0.3633942 0.11448249 0.021700994 0.0029887597
## 1_1   0.02019472 0.4838230 0.3768122 0.09948557 0.015390502 0.0022533056
## 0_1   0.03806201 0.5732297 0.3120292 0.06510190 0.008643248 0.0006343668
## Total 0.03897100 0.5275890 0.3290723 0.08568860 0.014396002 0.0019564815
##                  7            8            9           10
## 0_0   0.0002901795 0.0009534469 0.0011192638 1.658169e-04
## 1_0   0.0006497304 0.0009745955 0.0007147034 1.299461e-04
## 1_1   0.0003401216 0.0009353344 0.0005952128 1.700608e-04
## 0_1   0.0001585917 0.0007929585 0.0012687336 7.929585e-05
## Total 0.0003569257 0.0009253629 0.0008989239 1.454142e-04
```

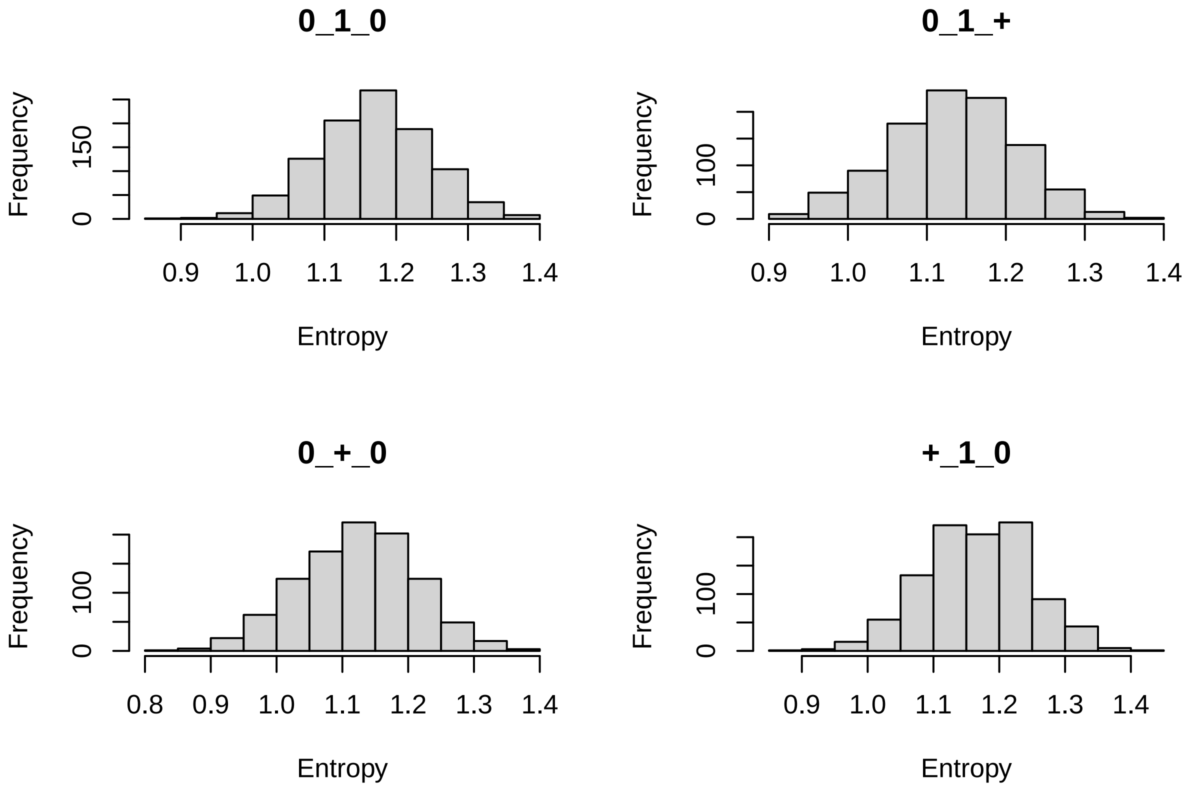
3-way HC-tree of GenHealth_3:



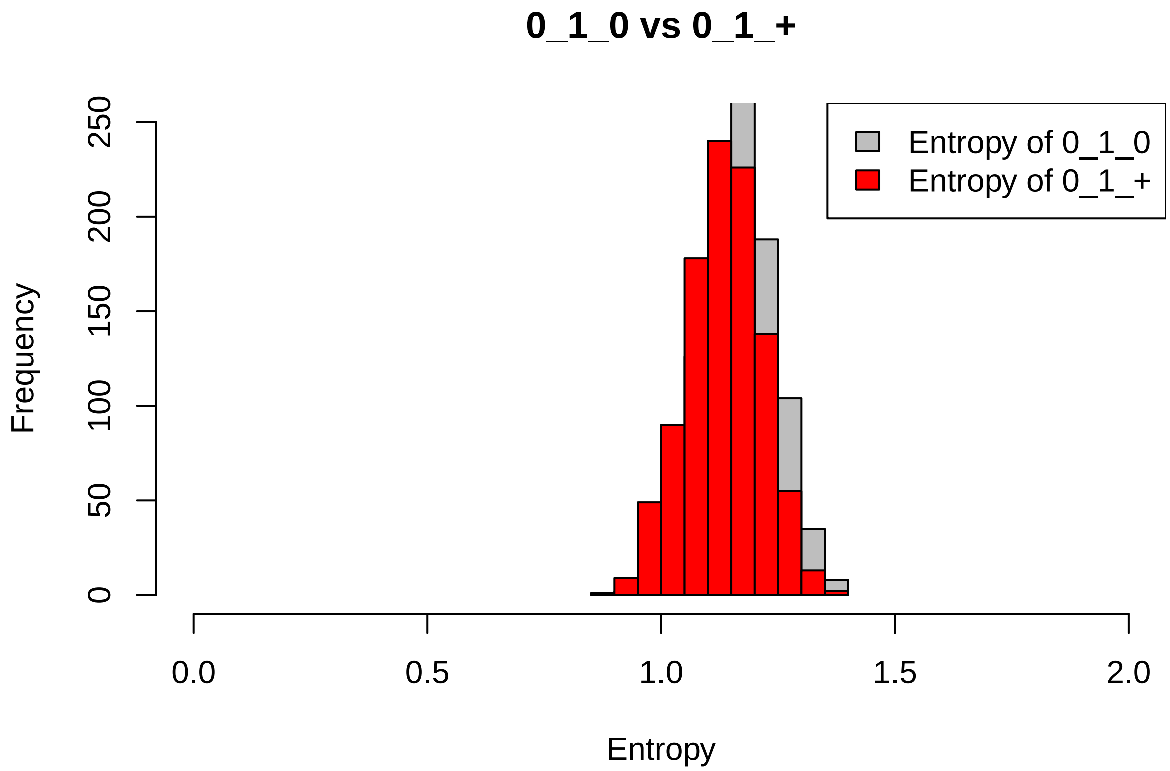
Comparing the entropies of categorical variables to see the effects

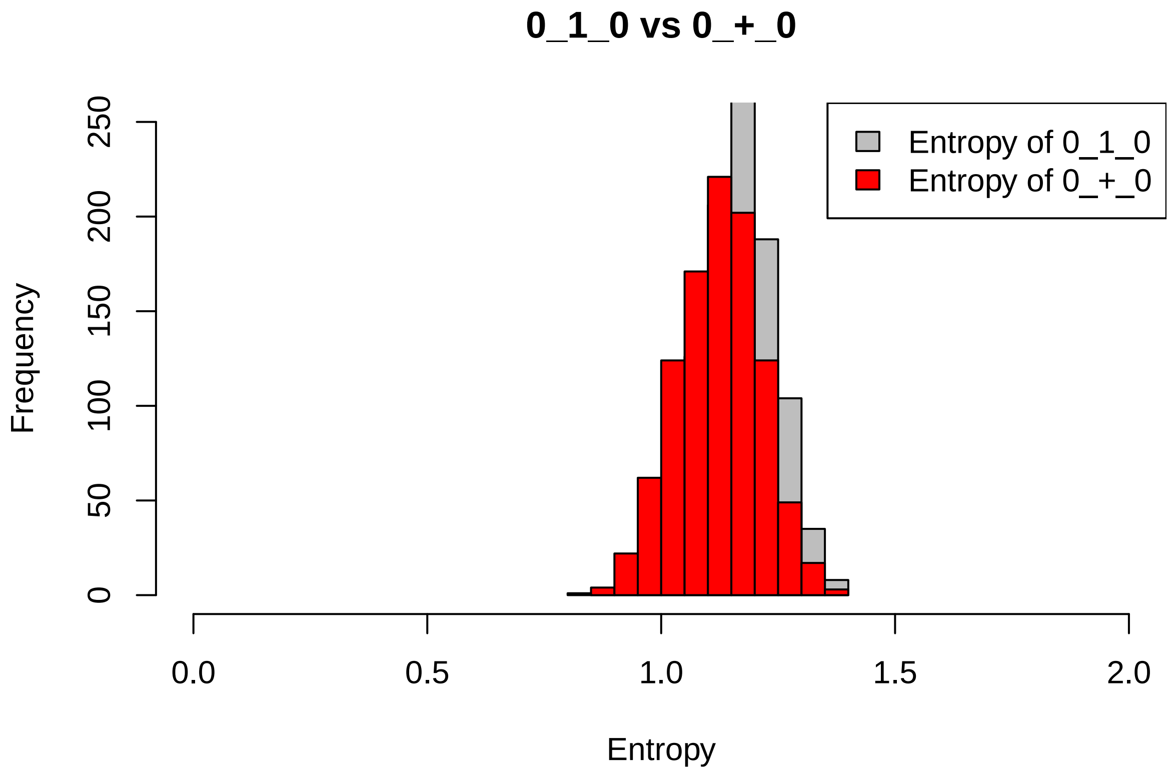
No HD __ Yes HighBP __ No HighChol

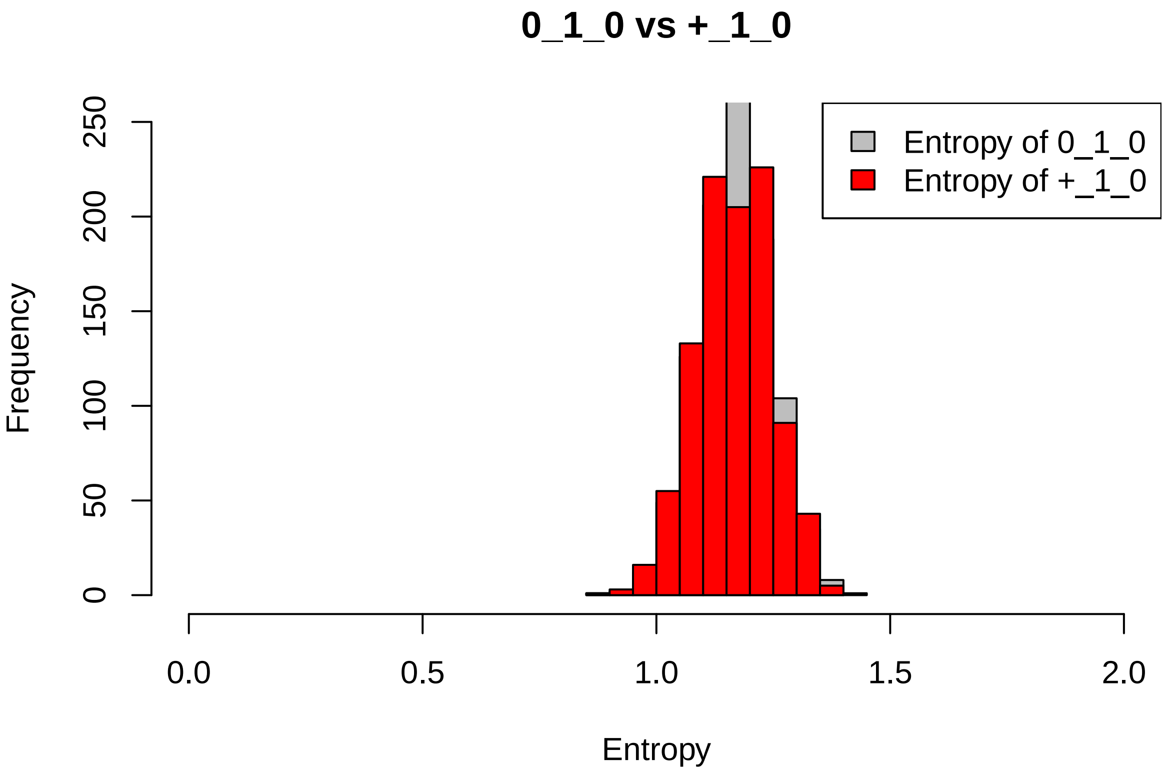
```
## Warning: package 'DescTools' was built under R version 4.2.2
```



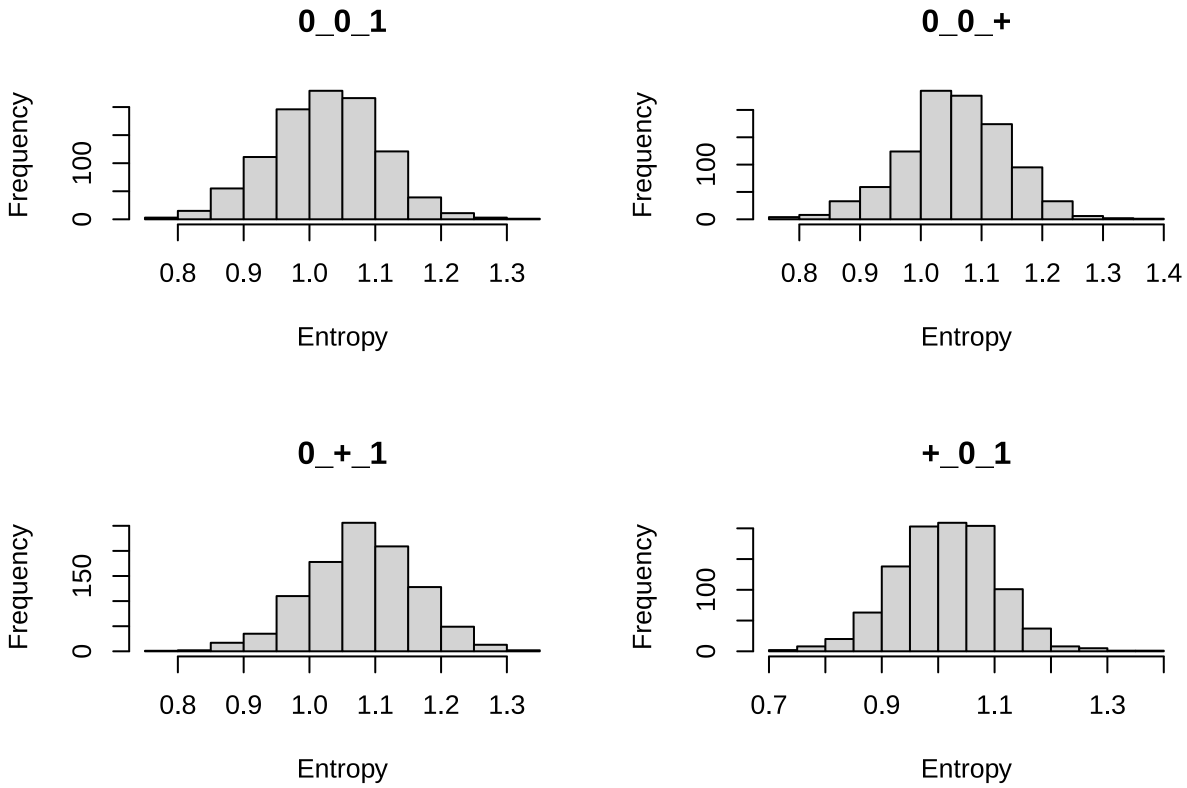
Looking at these graphs below, it seems that there isn't much shift happening in the two data. There is a small shift to the left when missing HighBP or HighChol. Thus, missing 1 bi-variable has little effect on 0_1_0. Since the graphs mostly overlap the origin, there is no interaction effect on the BMI distribution.


0_1_0 vs 0_1_+
Entropy of 0_1_0
Entropy of 0_1_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

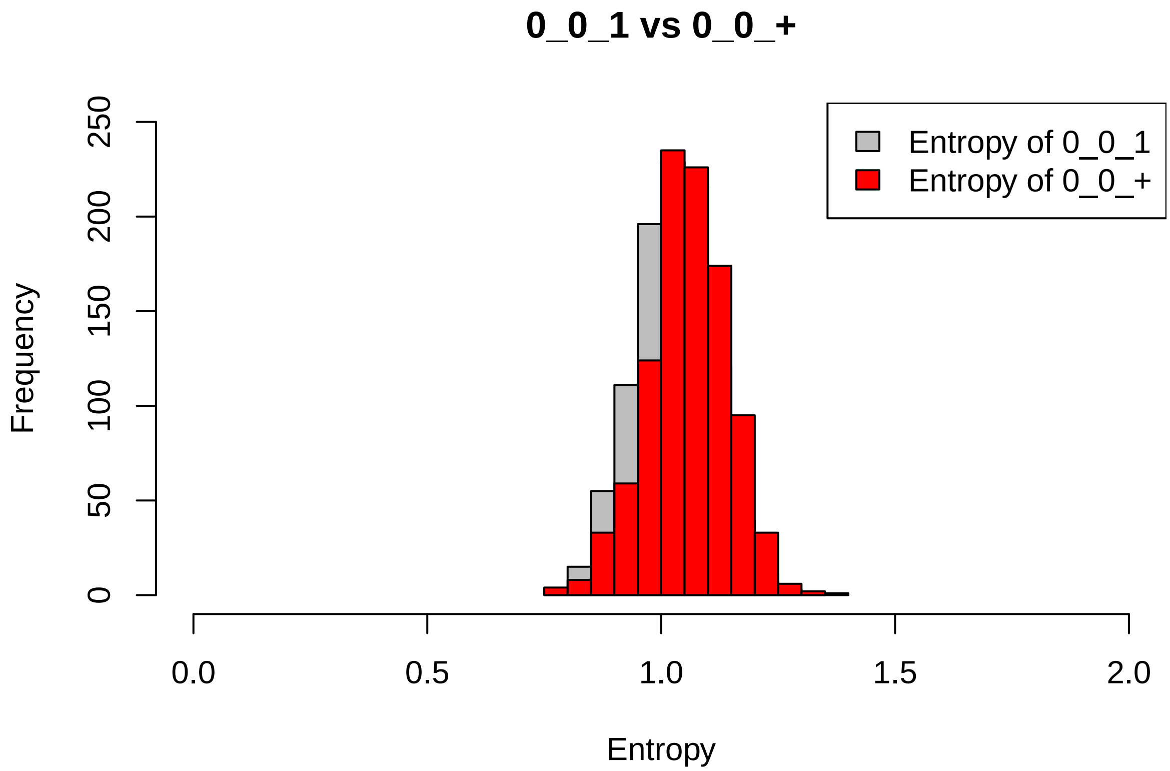

0_1_0 vs 0_+_0
Frequency
0
50
100
150
200
250
Entropy of 0_1_0
Entropy of 0_+_0
0.0
0.5
1.0
1.5
2.0
Entropy

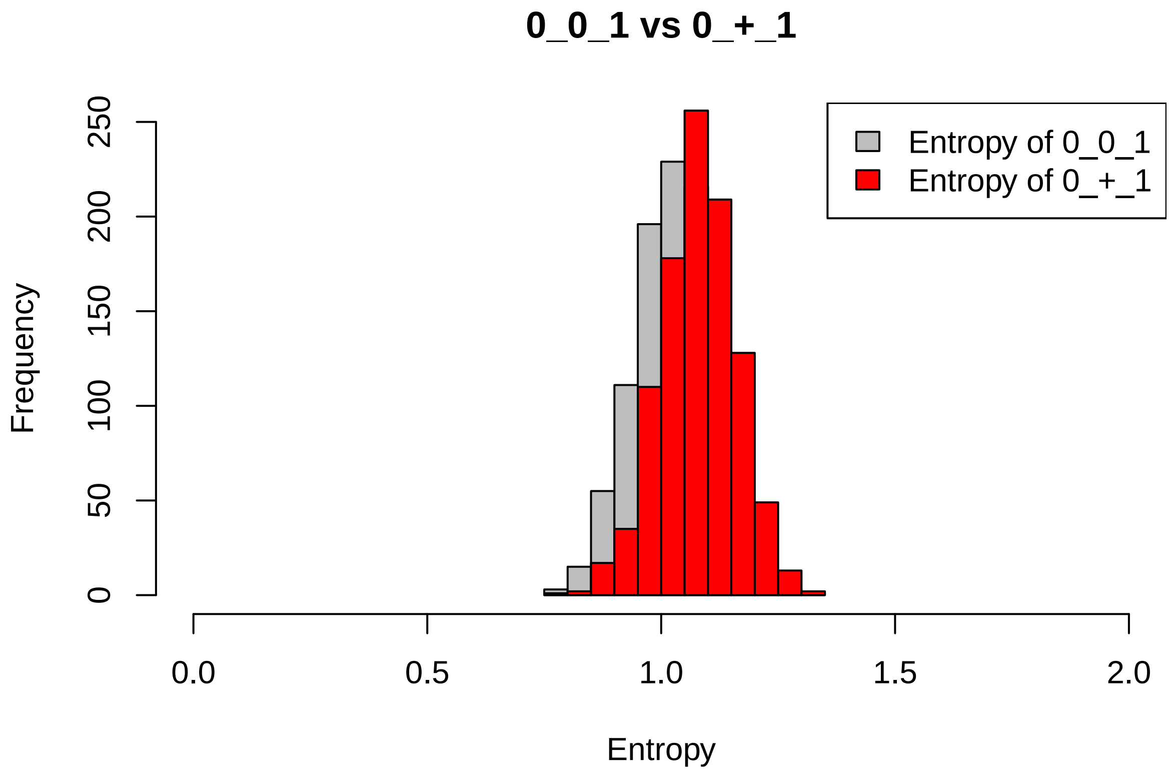

0_1_0 vs +_1_0
Frequency
0
50
100
150
200
250
Entropy of 0_1_0
Entropy of +_1_0
0.0
0.5
1.0
1.5
2.0
Entropy

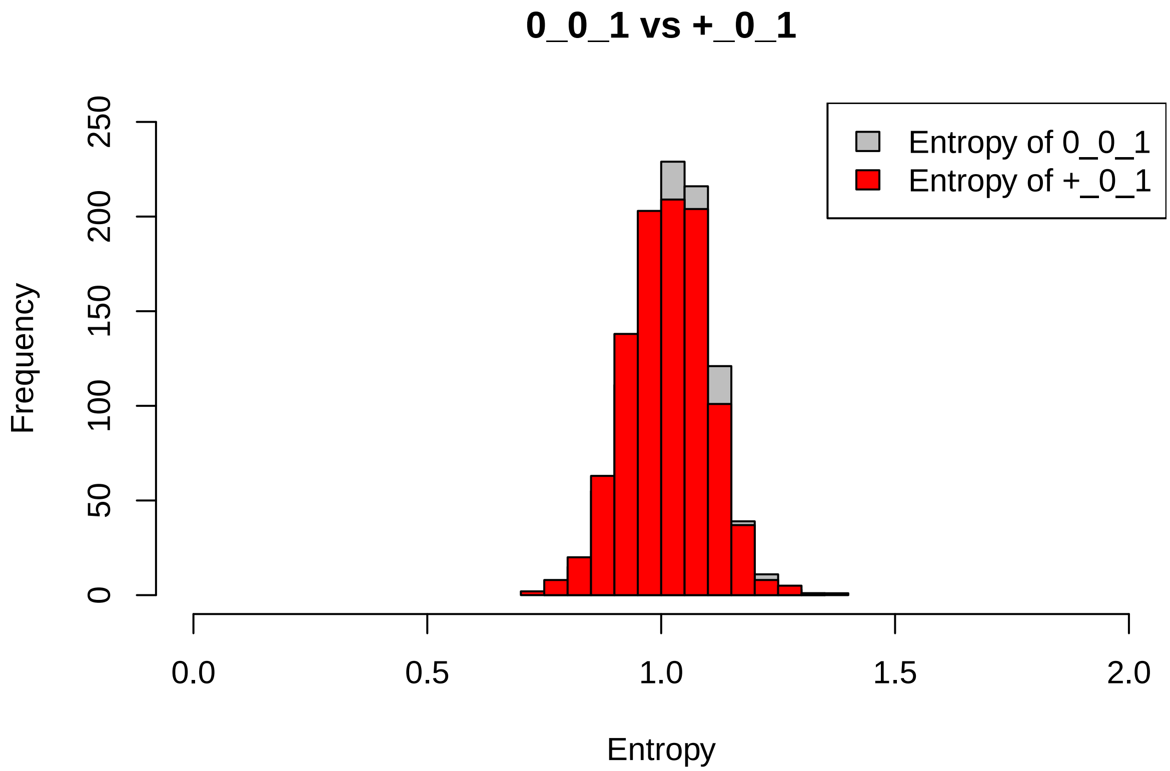
No HD __ No HighBP __ Yes HighChol



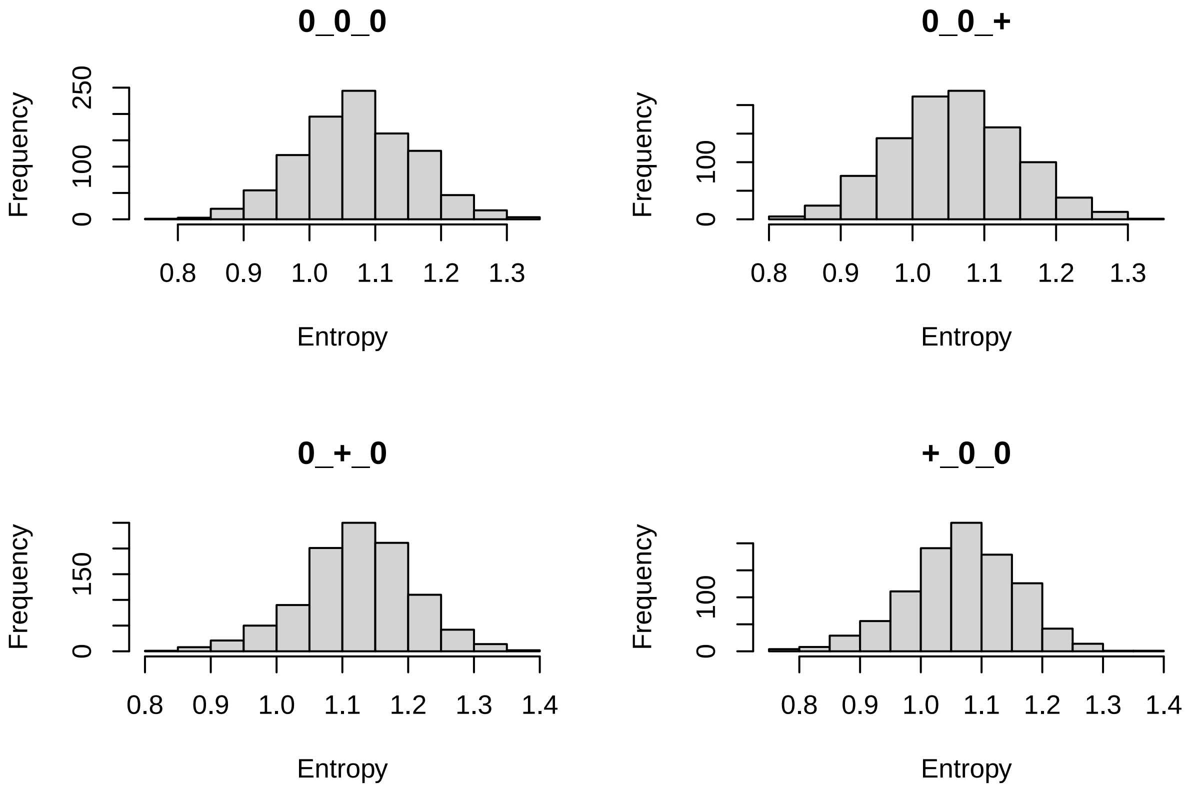
Looking at these graphs below, there is clearly a shift to the right when HighBP or HighChol is missing, which means the entropy level goes up and makes it less accurate. However, missing HD doesn't change much. Since the graphs of missing HighBP or missing HighChol doesn't overlap the origin, there is an interaction effect on the BMI distribution.


0_0_1 vs 0_0_+
Entropy of 0_0_1
Entropy of 0_0_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

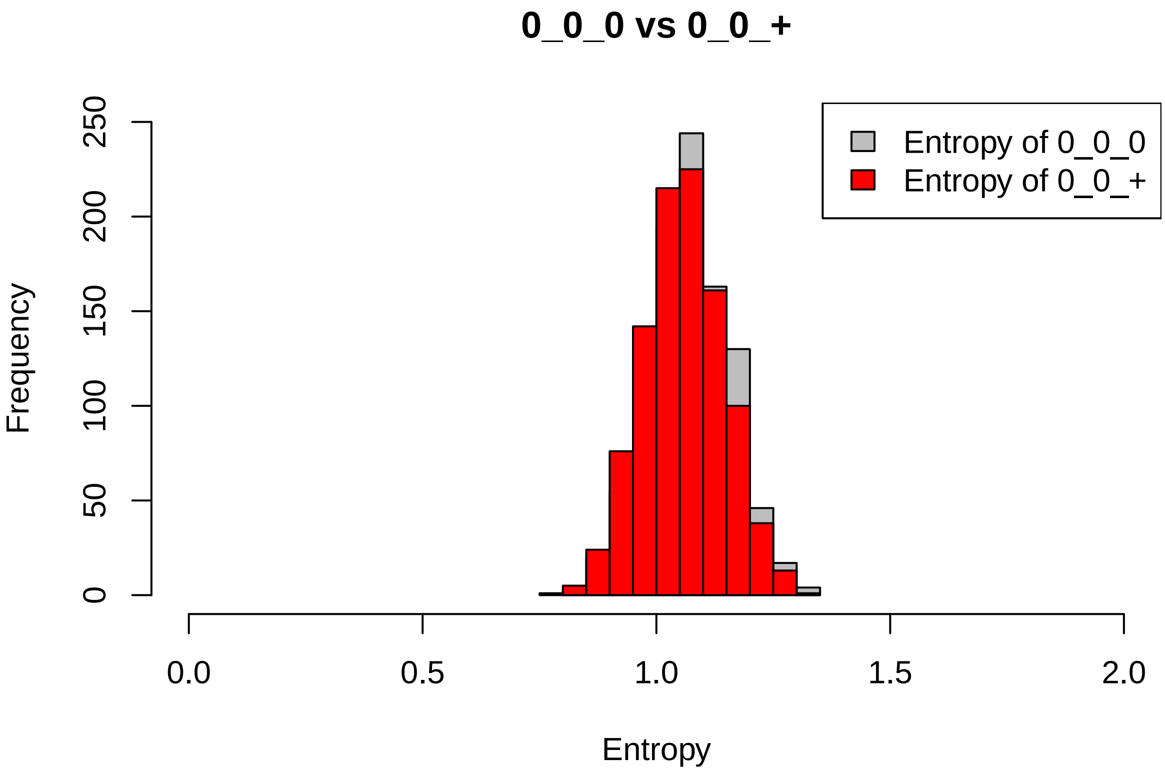

0_0_1 vs 0_+_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 0_0_1
Entropy of 0_+_1

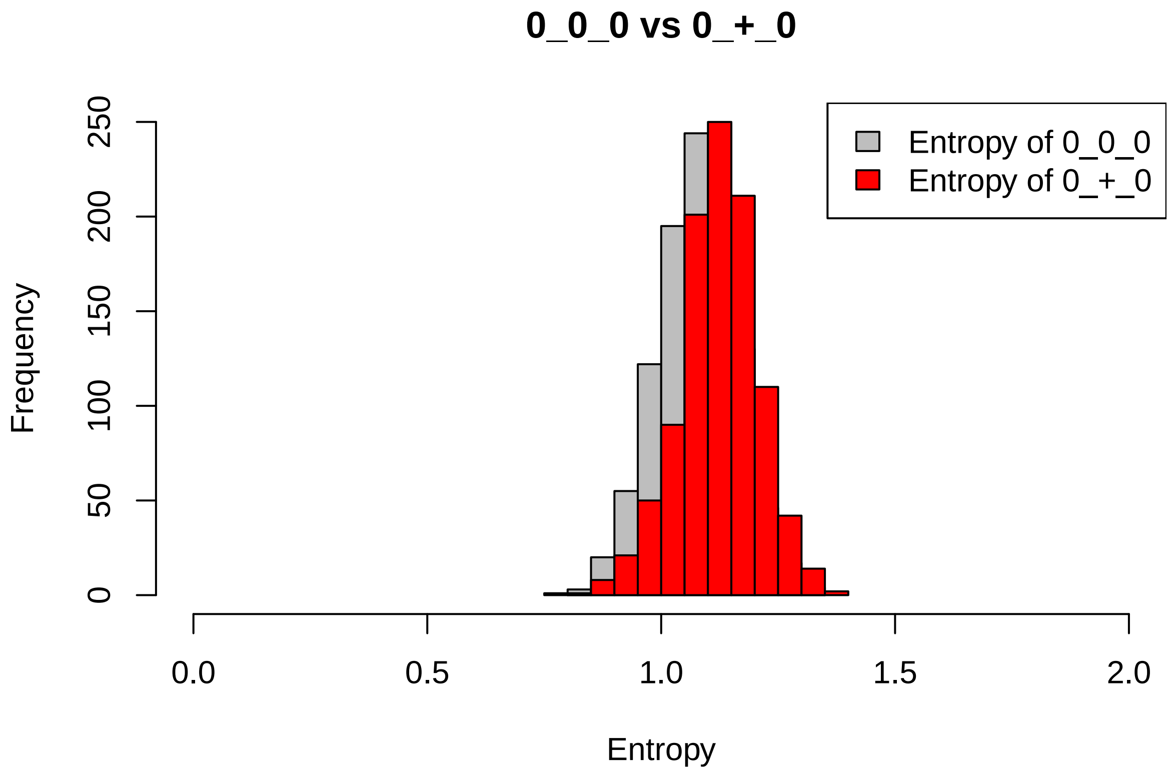

0_0_1 vs +_0_1
Entropy of 0_0_1
Entropy of +_0_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

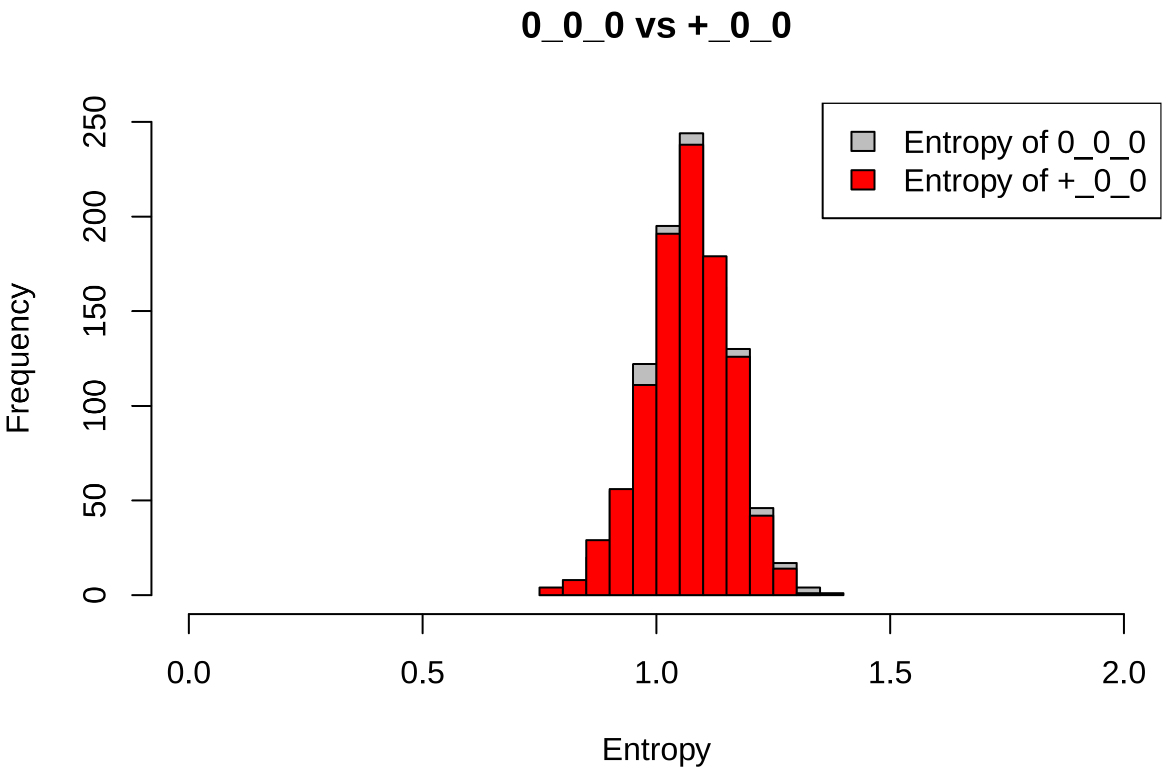
No HD __ No HighBP __ No HighChol



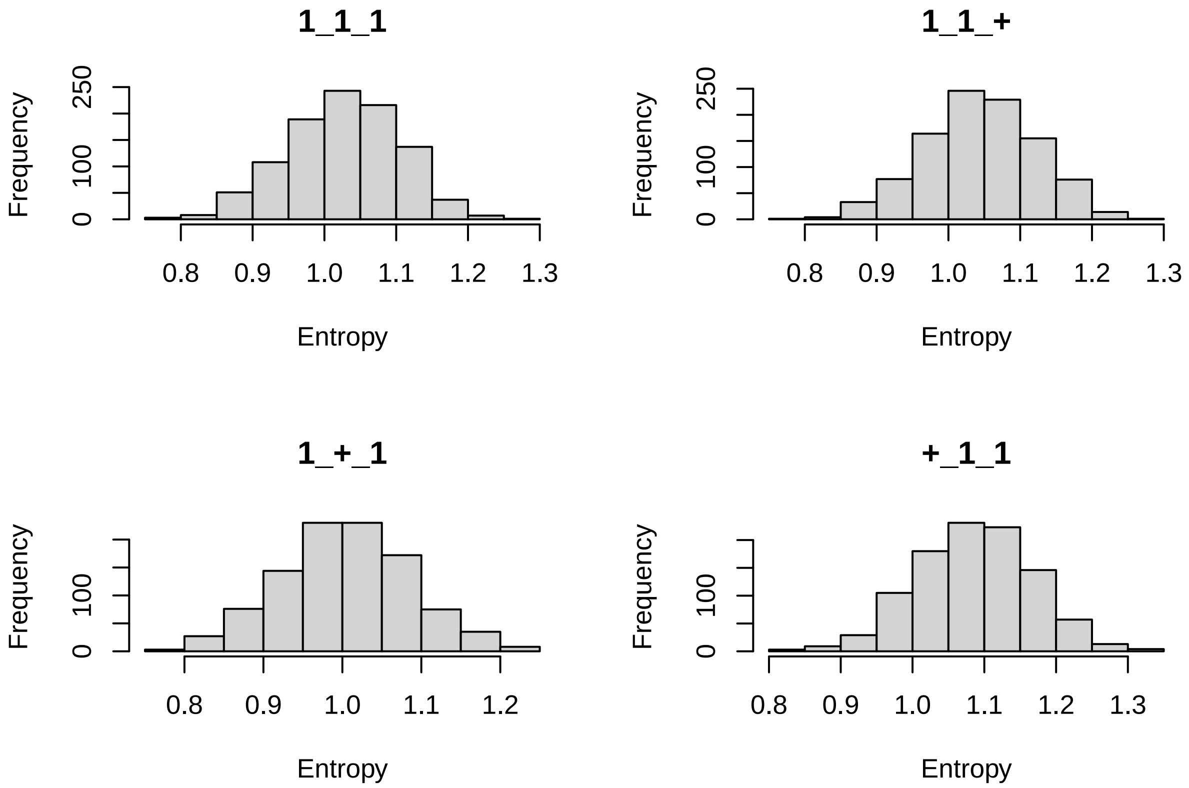
Looking at these graphs below, there is not much change going when missing one bi-variable. There is a small shift to the right when missing HighBP, but not much. Since the graphs mostly overlap to the origin, there is no interaction effect on the BMI distribution.


0_0_0 vs 0_0_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 0_0_0
Entropy of 0_0_+

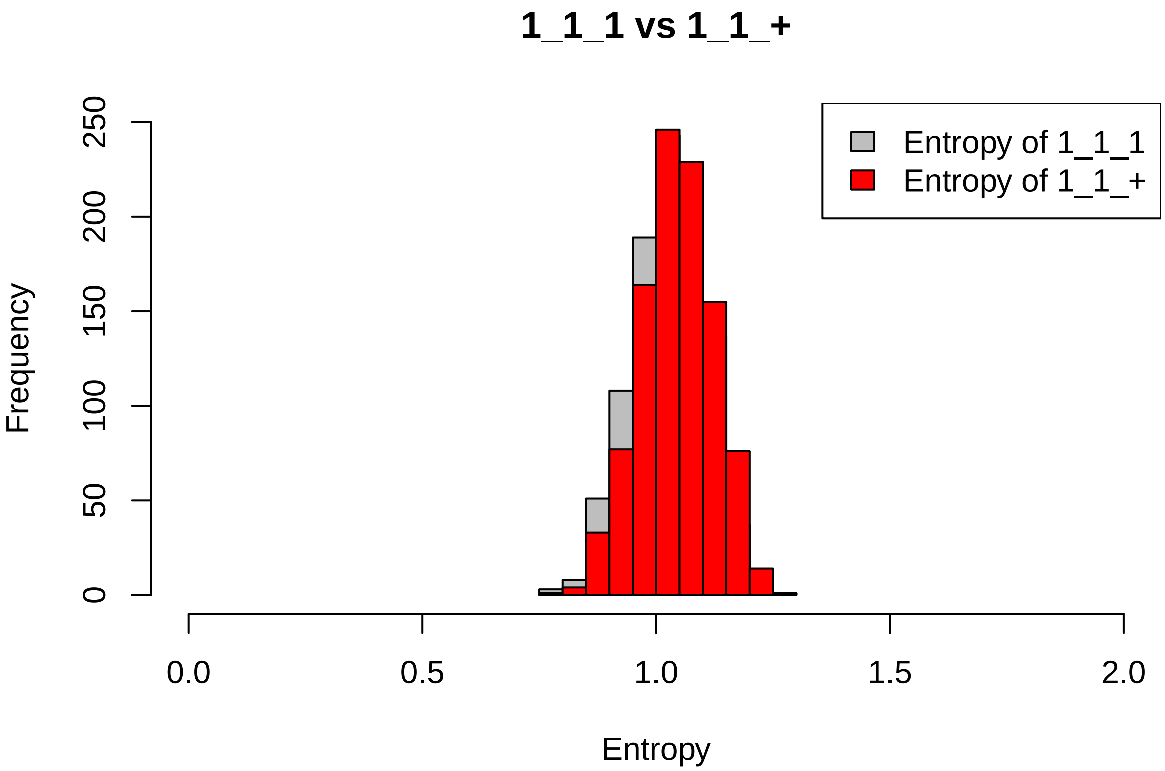

0_0_0 vs 0_+_0
Entropy of 0_0_0
Entropy of 0_+_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

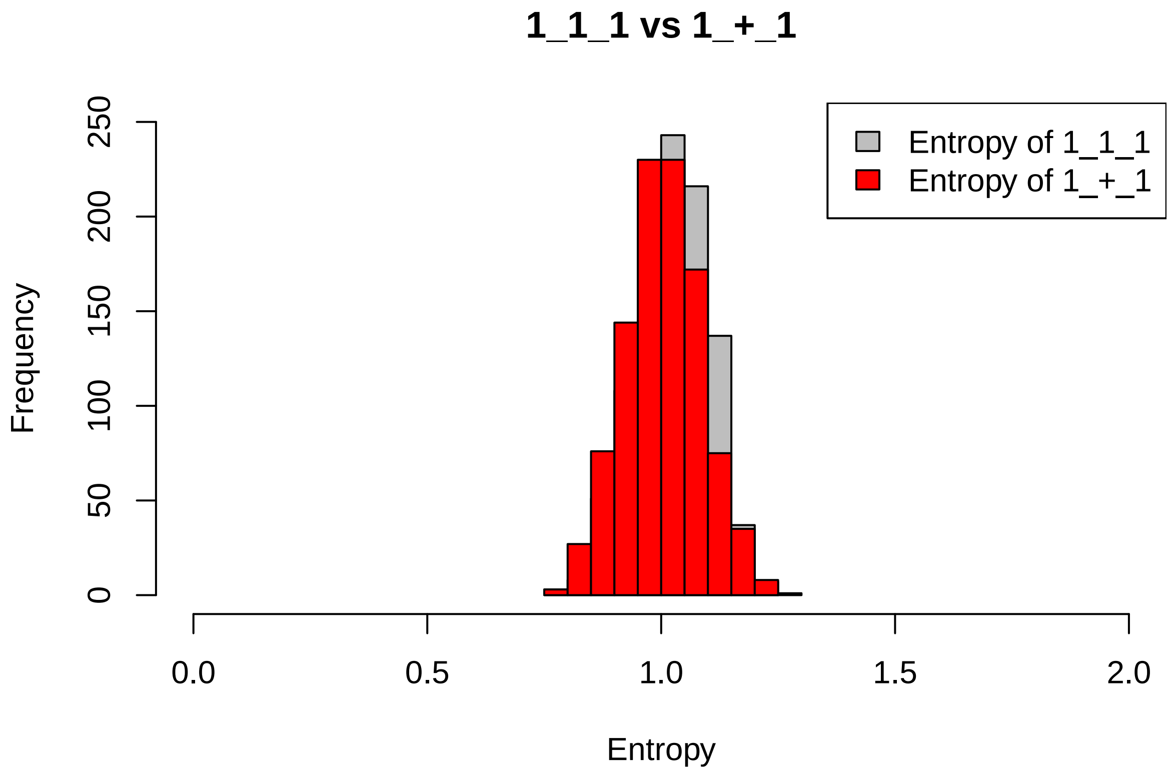

0_0_0 vs +_0_0
Entropy of 0_0_0
Entropy of +_0_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

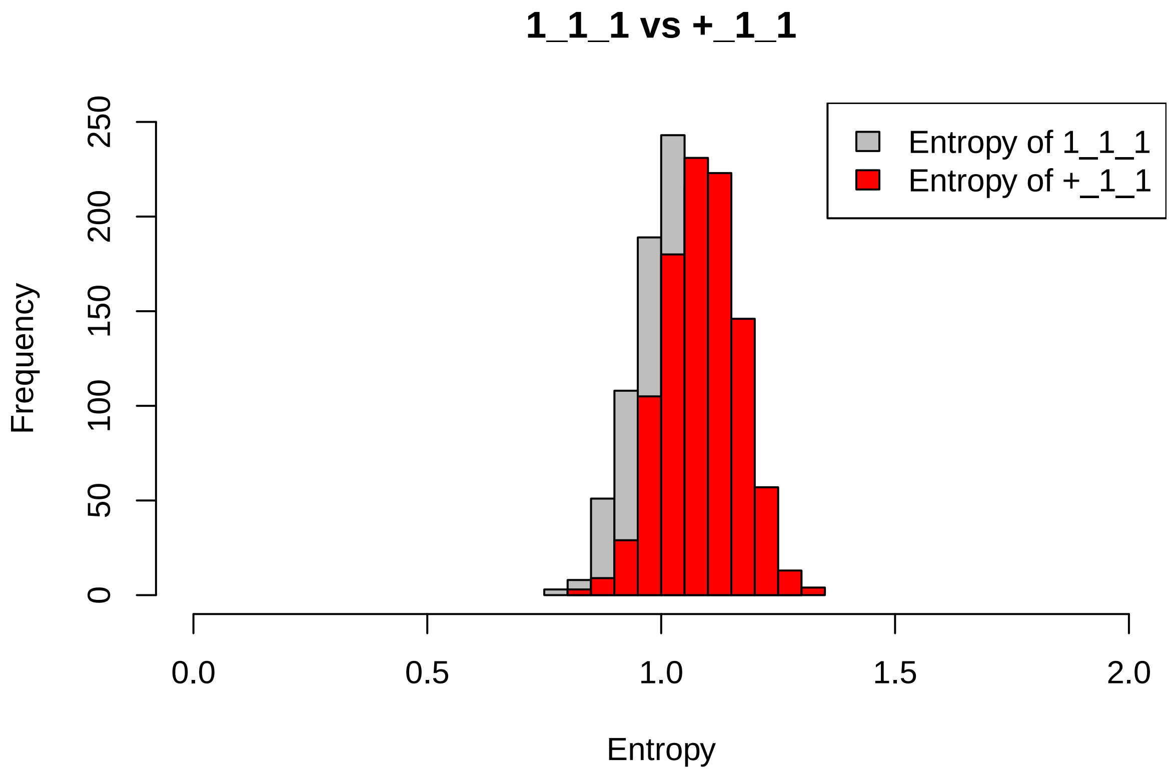
Yes HD __ Yes HighBP __ Yes HighChol



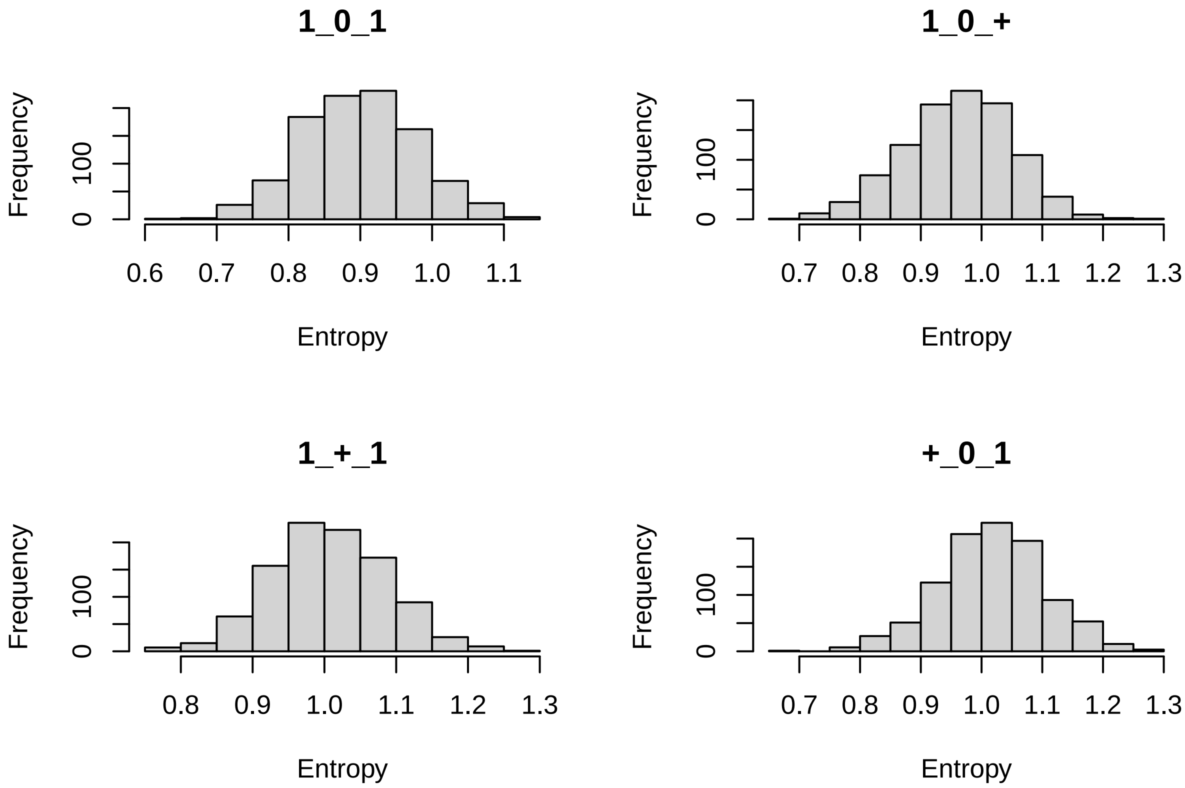
Looking at these graphs below, there is clearly a shift to the right when HeartDisease is missing, which means the entropy level goes up and makes it less accurate. However, missing High BP or HighChol doesn't change much. Since the graphs mostly overlap the origin, there is no interaction effect on the BMI distribution.


1_1_1 vs 1_1_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 1_1_1
Entropy of 1_1_+

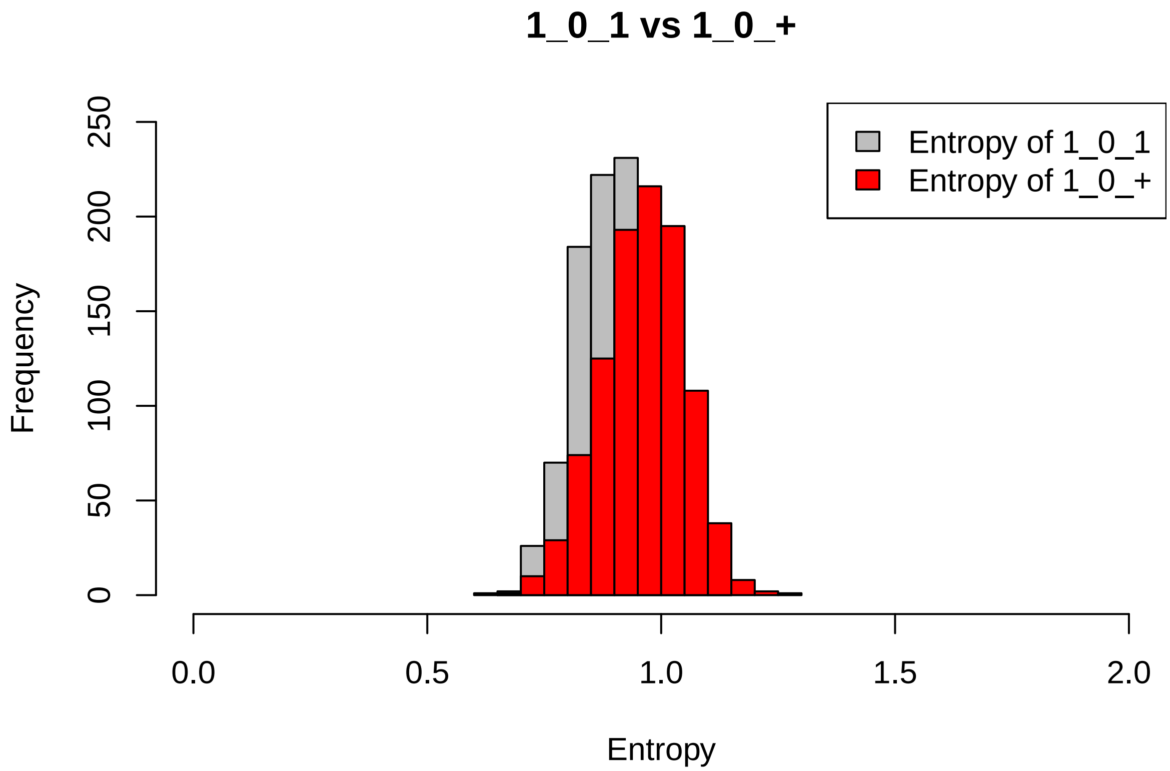

1_1_1 vs 1_+_1
Entropy of 1_1_1
Entropy of 1_+_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

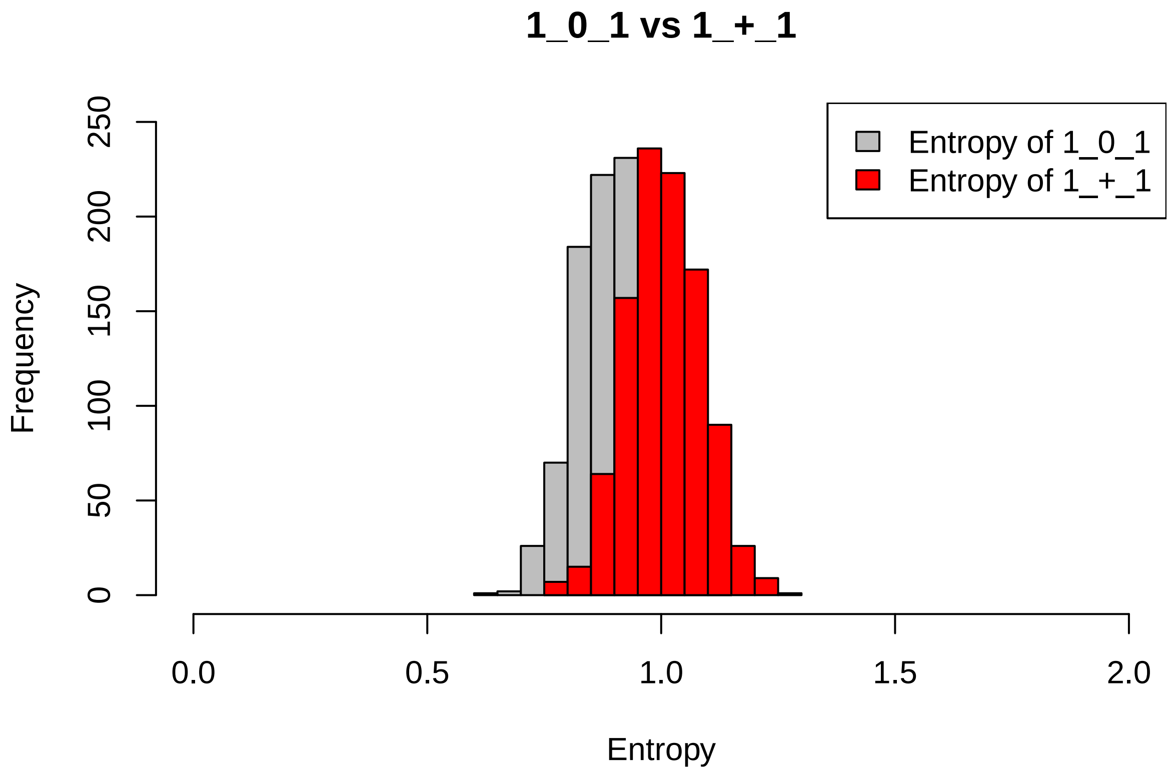

1_1_1 vs +_1_1
Entropy of 1_1_1
Entropy of +_1_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

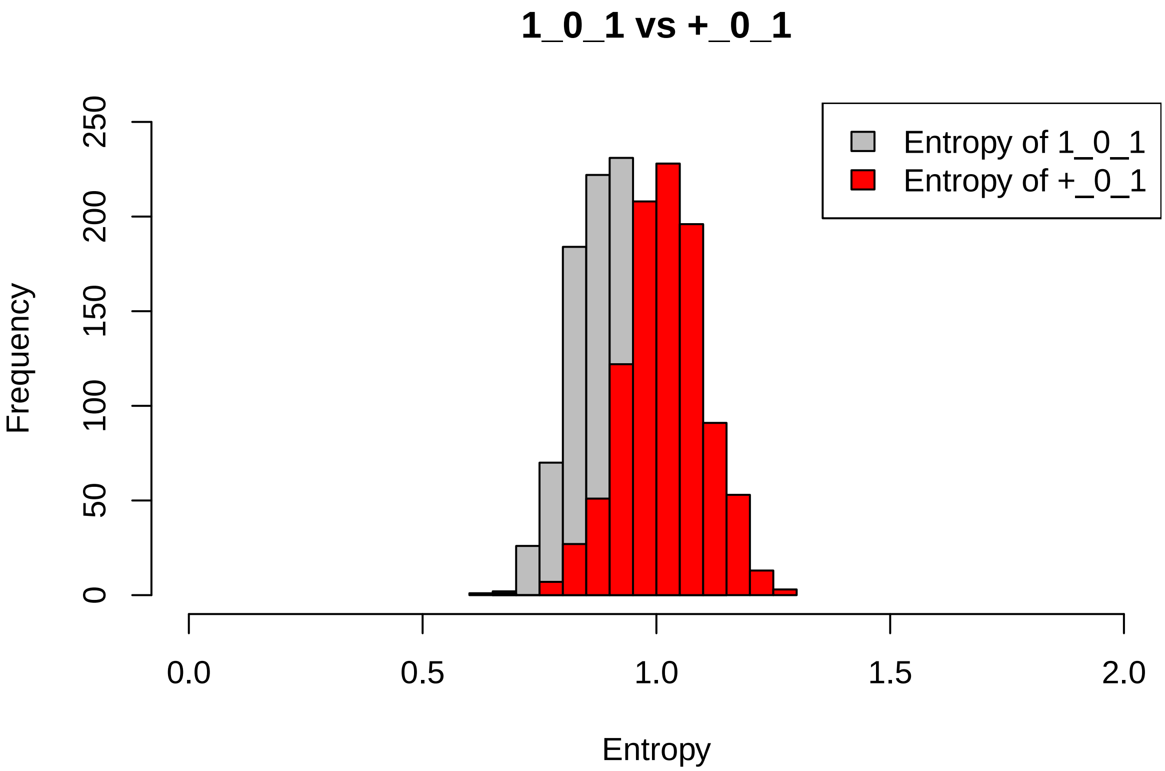
Yes HD __ No HighBP __ Yes HighChol



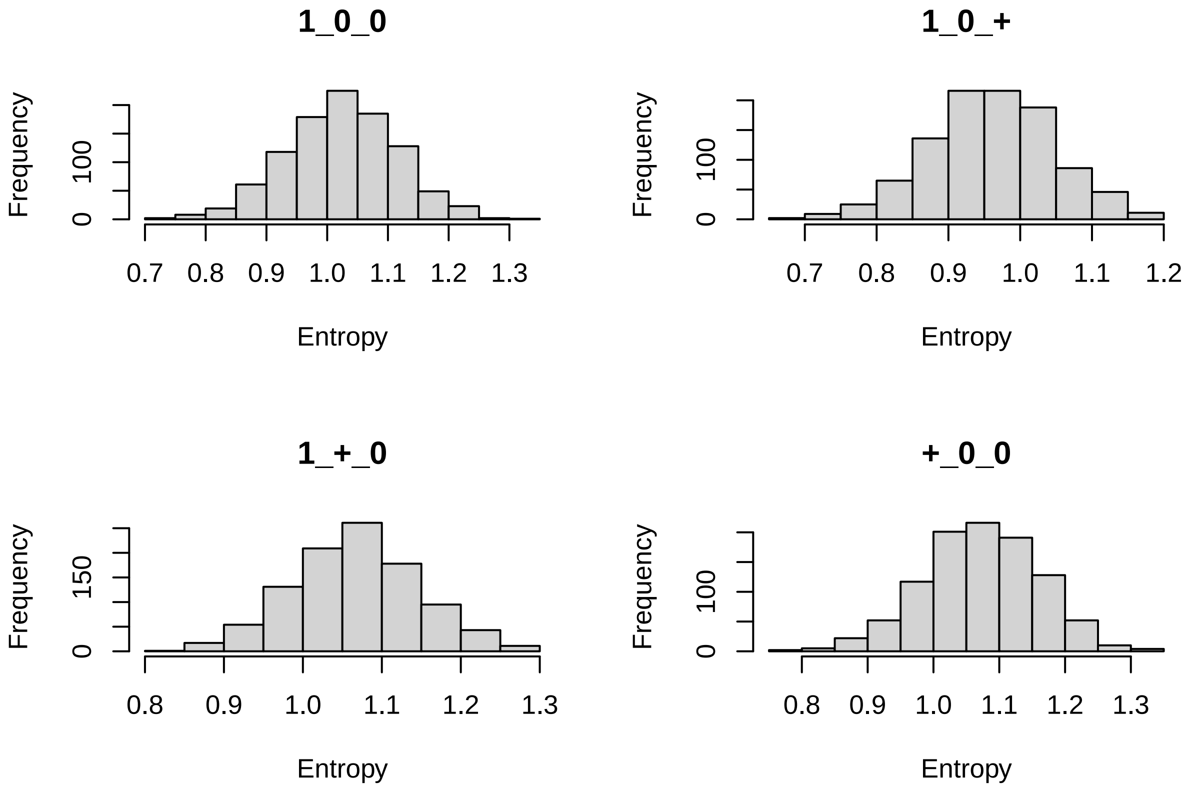
Looking at these graphs below, there is clearly a strong shift to the right when HighBP or HeartDisease or HighChol is missing, which means the entropy level goes up and makes it less accurate. Since all 3 graphs of missing HighBP or missing HeartDisease or missing HighChol don't overlap the origin, there is an interaction effect on the BMI distribution.


1_0_1 vs 1_0_+
Entropy of 1_0_1
Entropy of 1_0_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

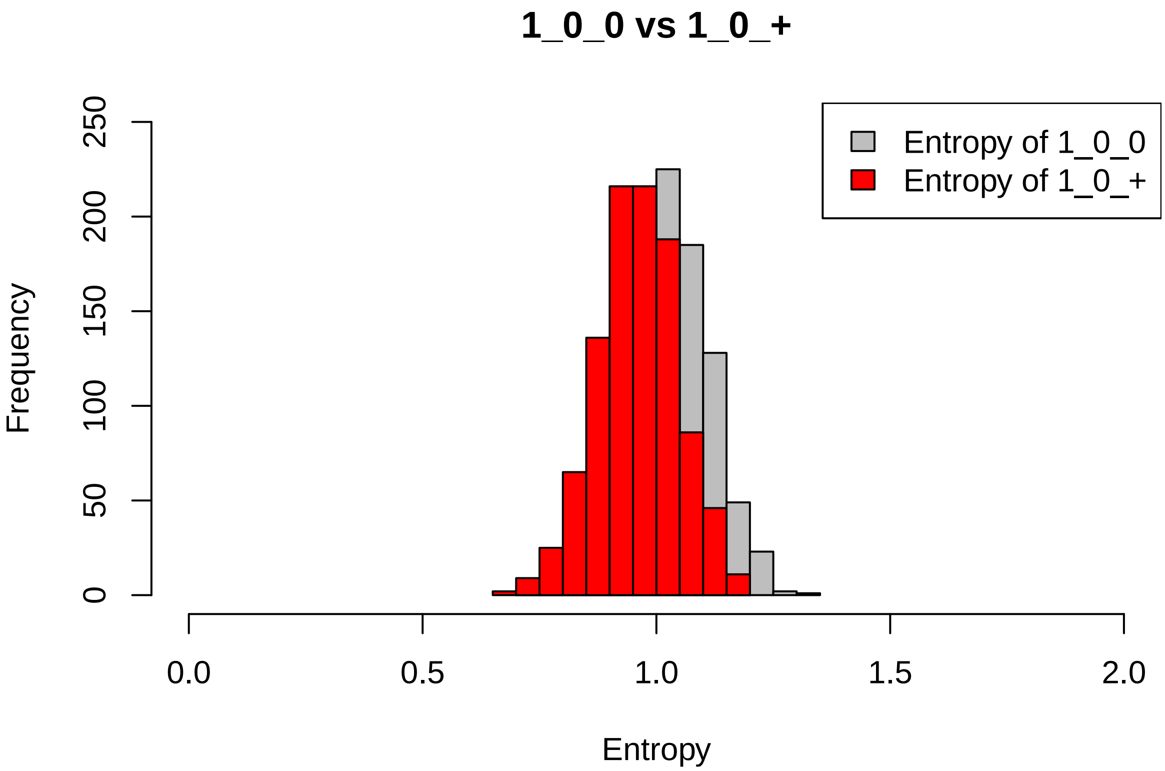

1_0_1 vs 1_+_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 1_0_1
Entropy of 1_+_1

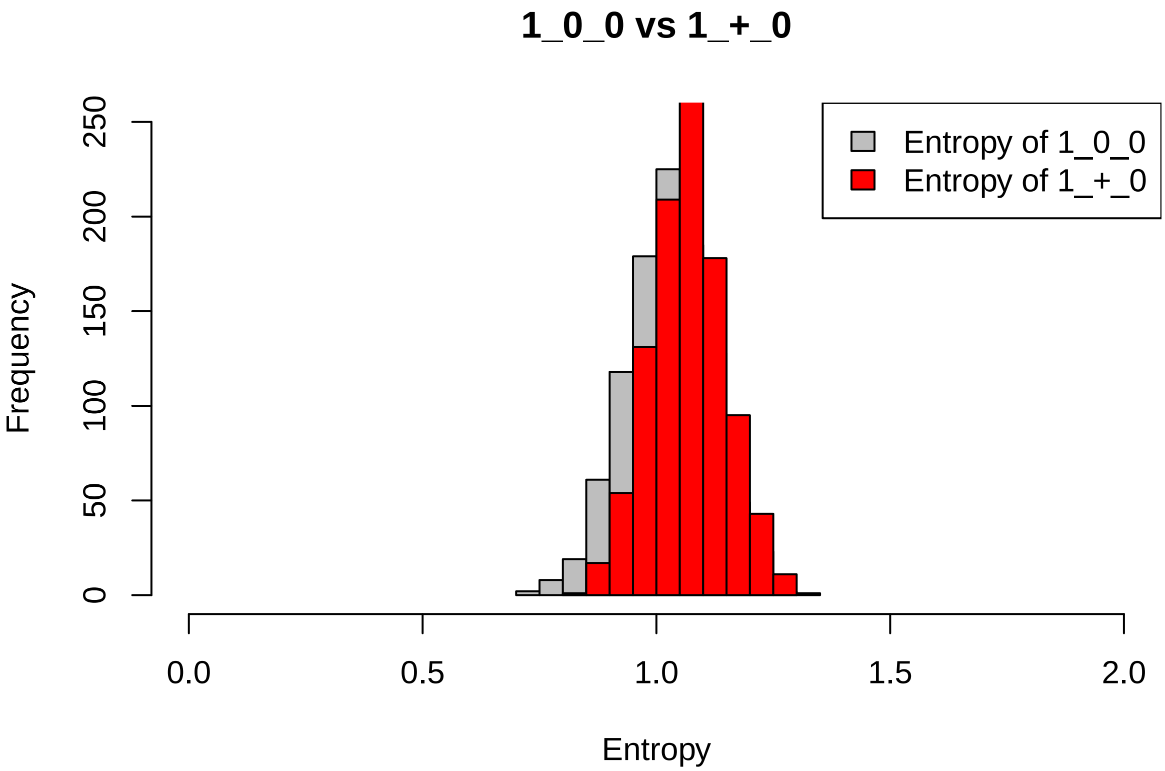

1_0_1 vs +_0_1
Entropy of 1_0_1
Entropy of +_0_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

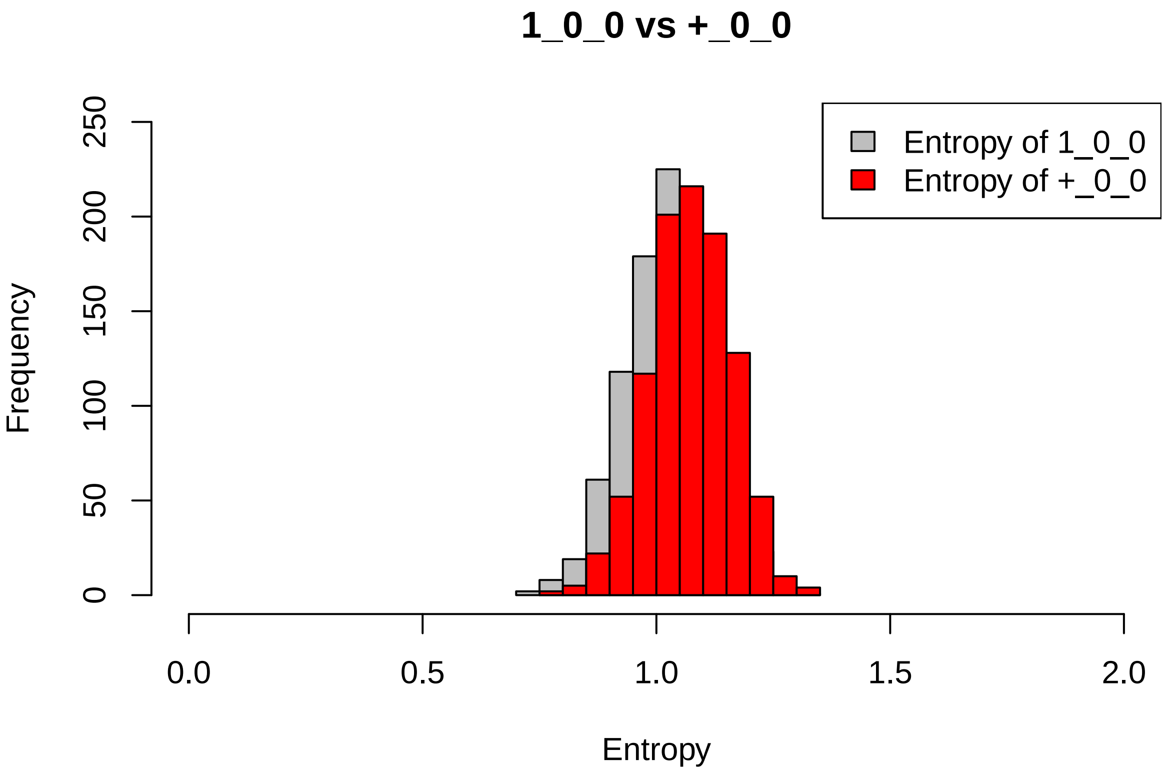
Yes HD _ No HighBP _ No HighChol



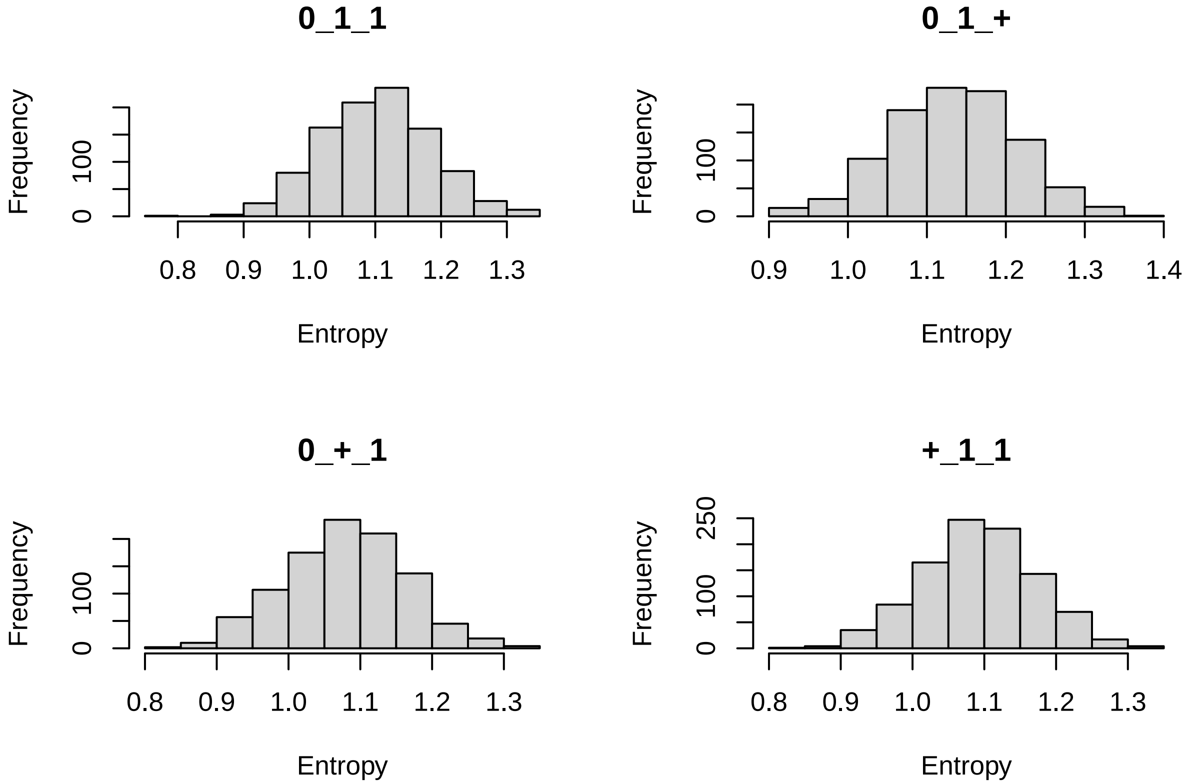
Looking at these graphs below, there is not much change happening. There is a little shift to the left when missing HighChol, and a little shift to the right when missing HeartDisease or HighBP. Thus, having missing 1 bi-variable have little effect on 1_0_0. Since the graphs mostly overlap the origin, there is no interaction effect on the BMI distribution.


1_0_0 vs 1_0_+
Entropy of 1_0_0
Entropy of 1_0_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy


1_0_0 vs 1_+_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 1_0_0
Entropy of 1_+_0


1_0_0 vs +_0_0
Entropy of 1_0_0
Entropy of +_0_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

No HD __ Yes HighBP __ Yes HighChol

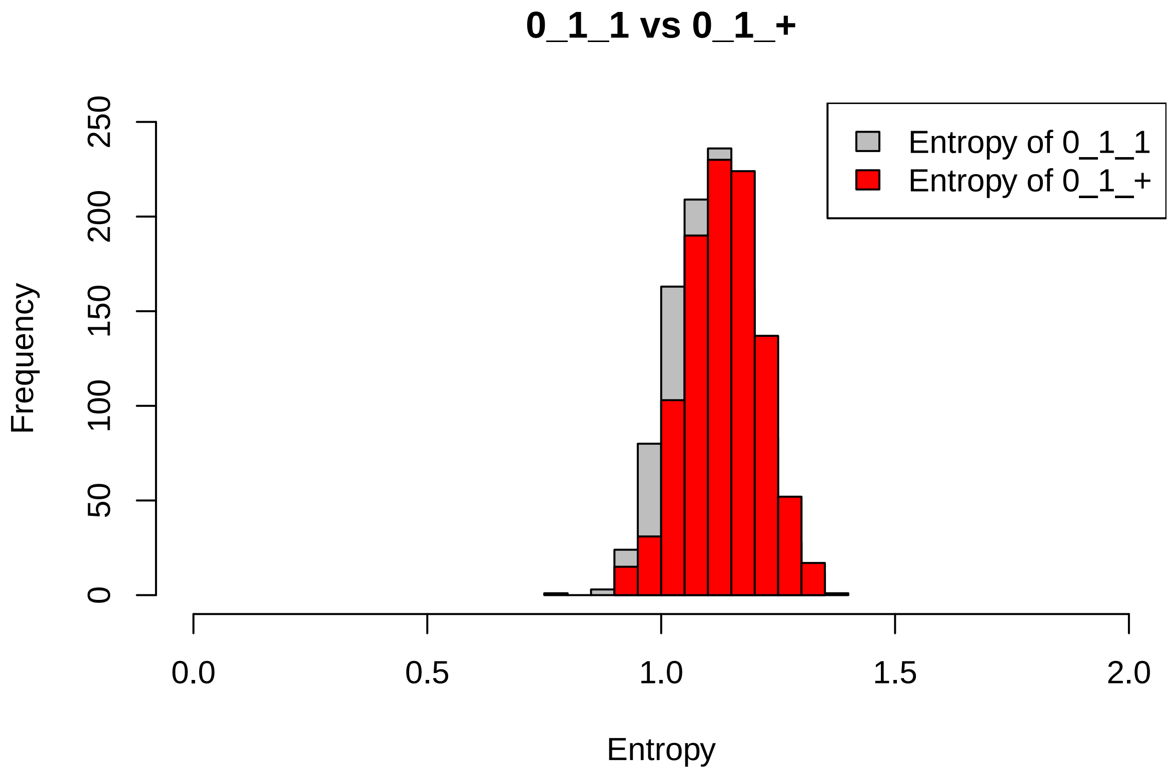

0_1_1

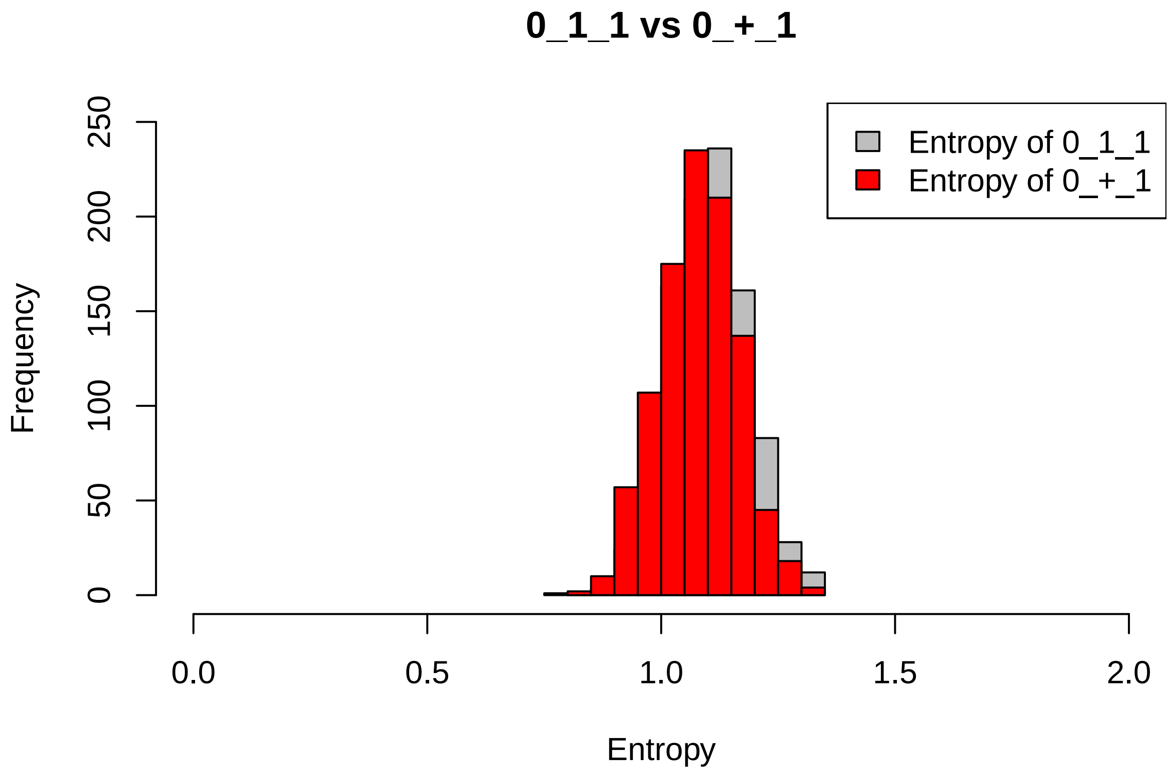
0_1_+

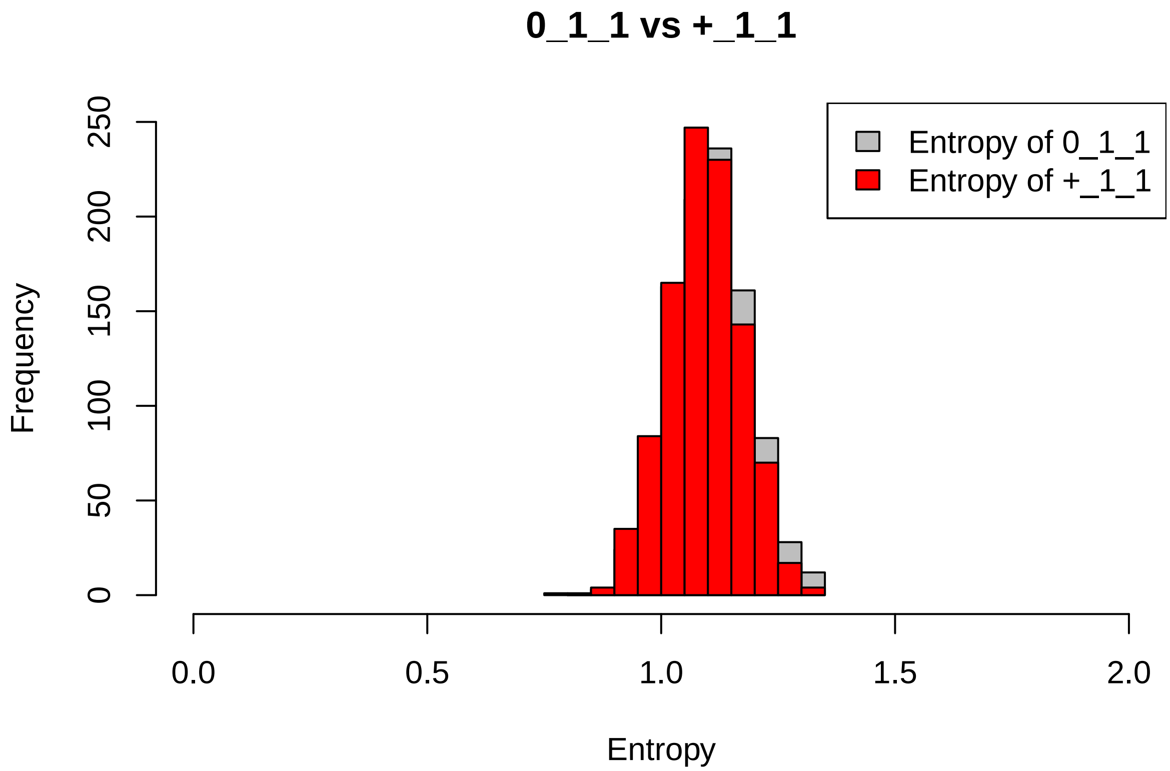
0_+_1

+_1_1

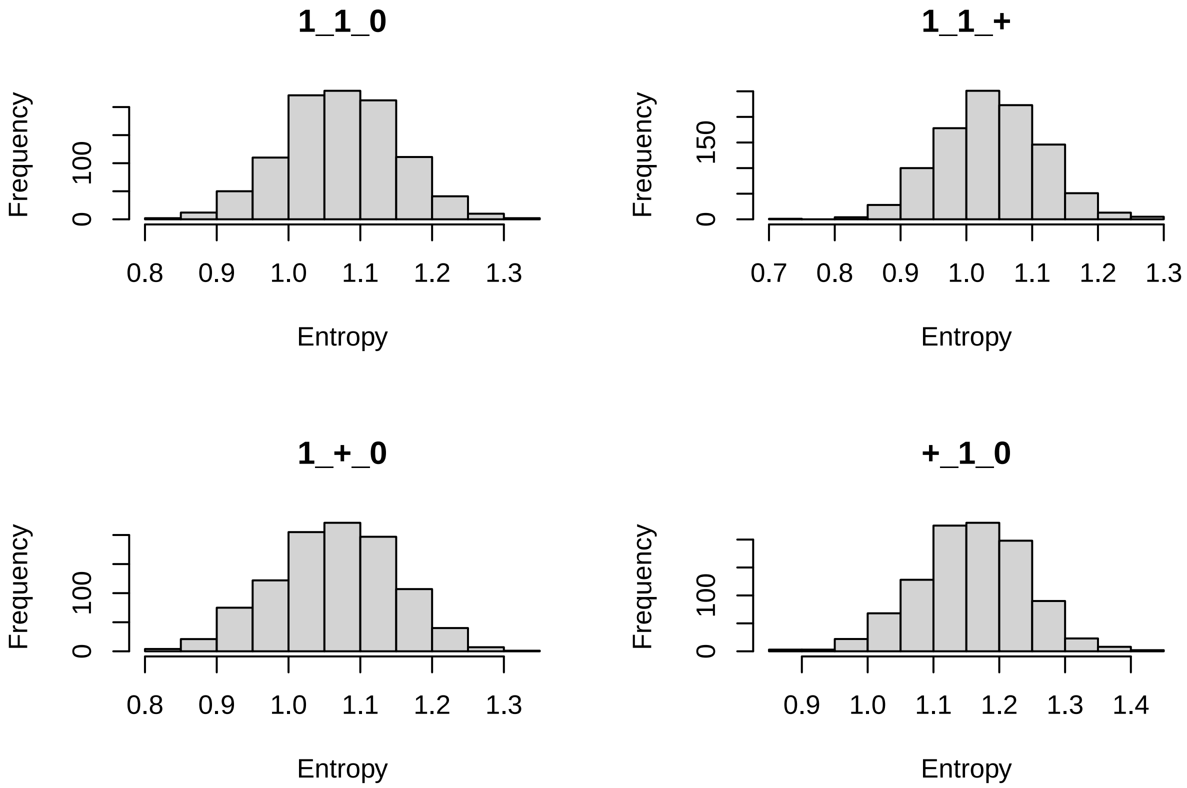
Looking at these graphs below, there is not much change happening. There is a small shift to the right when missing HighChol, and a small shift to the left when missing HighBP. Thus, missing one bi-variable has little effect on 0_1_1. Since the graphs mostly overlap the origin, there is no interaction effect on the BMI distribution.


0_1_1 vs 0_1_+
Entropy of 0_1_1
Entropy of 0_1_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

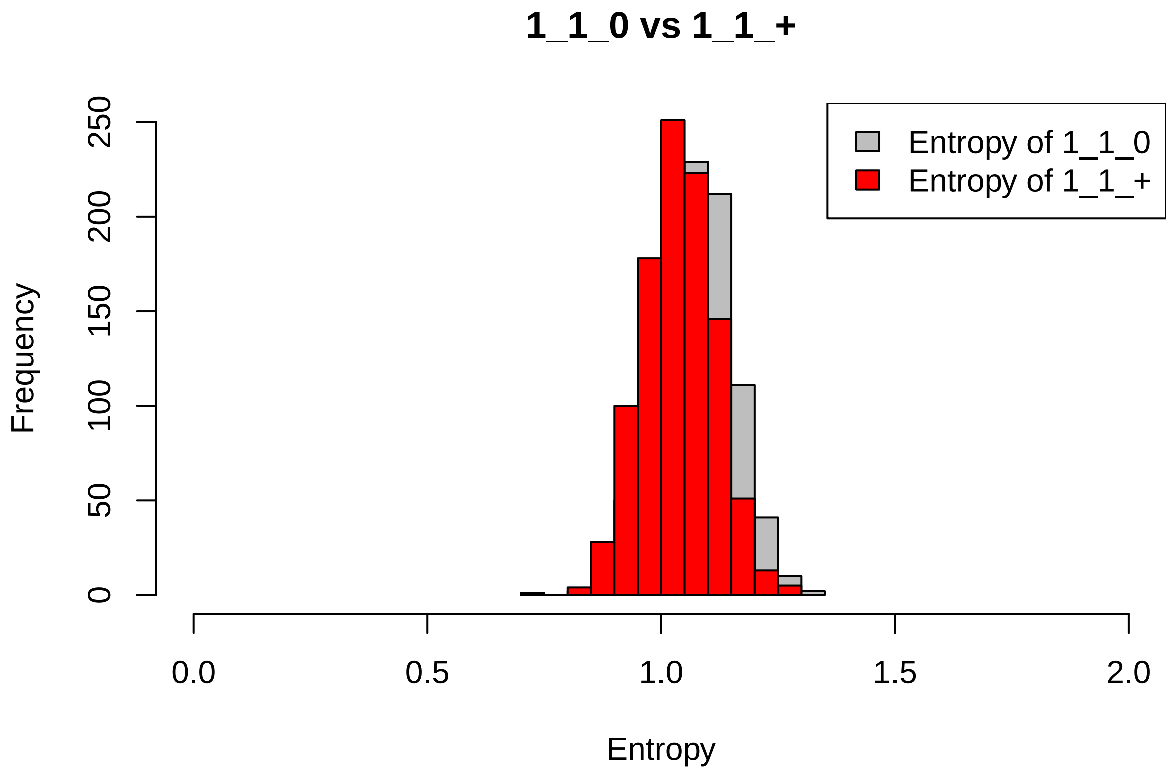

0_1_1 vs 0_+_1
Frequency
0
50
100
150
200
250
Entropy of 0_1_1
Entropy of 0_+_1
0.0
0.5
1.0
1.5
2.0
Entropy

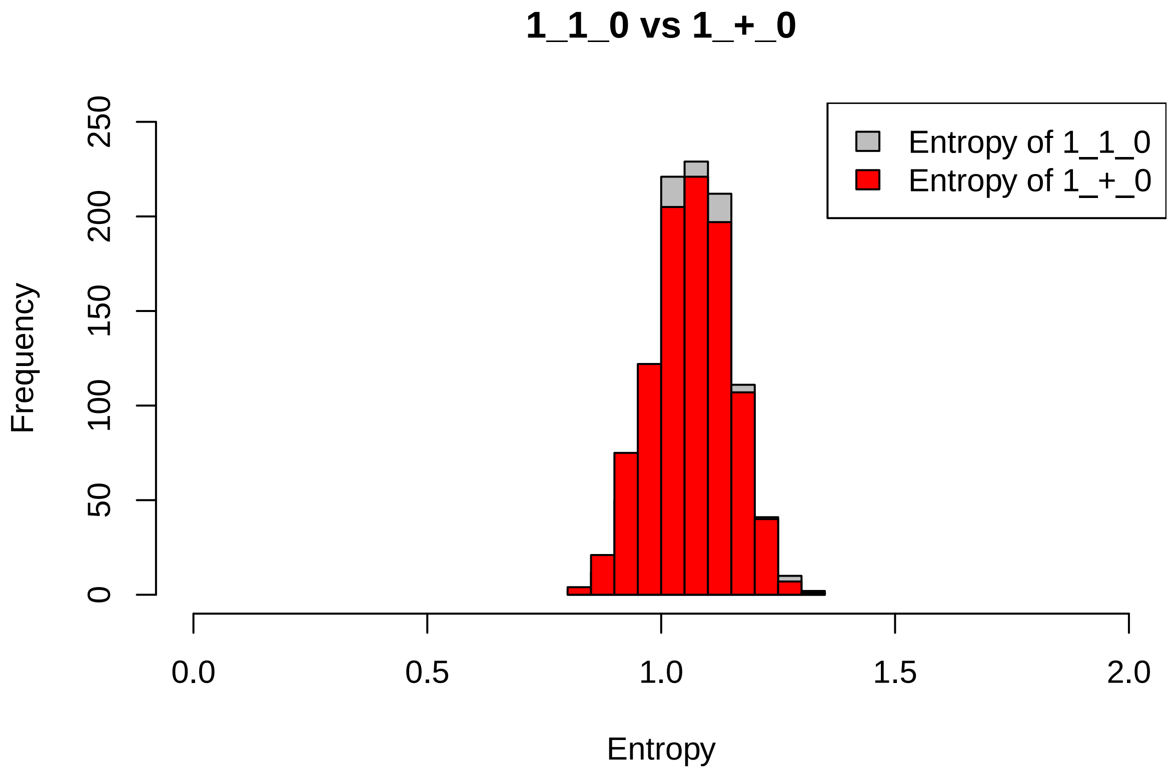

0_1_1 vs +_1_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 0_1_1
Entropy of +_1_1

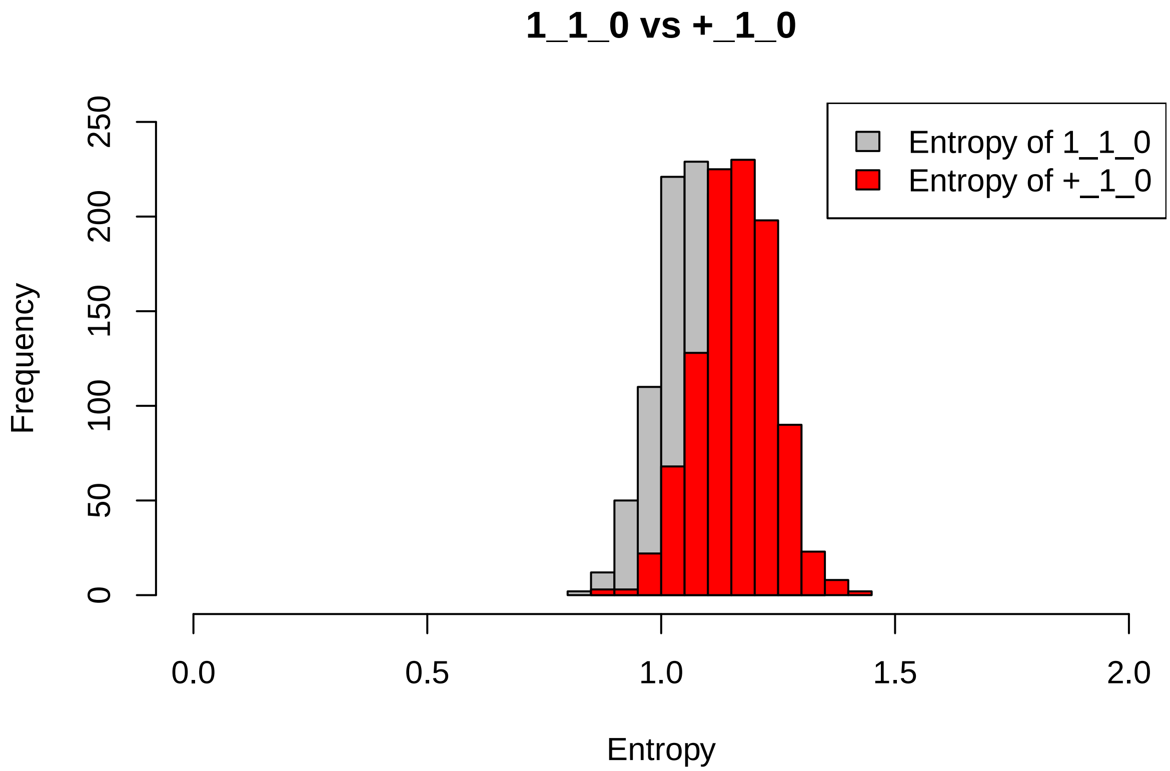
Yes HD __ Yes HighBP __ No HighChol



Looking at these graphs below, there is not much change happening when missing HighBP or HighChol. However, there is a clear shift to the right when missing HeartDisease, which means that the entropy level goes up and makes it less accurate. Since the graphs mostly overlap the origin, there is no interaction effect on the BMI distribution. Thus, missing one bi-variable has little effect on 1_1_0.


1_1_0 vs 1_1_+
Entropy of 1_1_0
Entropy of 1_1_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy


1_1_0 vs 1_+_0
Entropy of 1_1_0
Entropy of 1_+_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy


1_1_0 vs +_1_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 1_1_0
Entropy of +_1_0

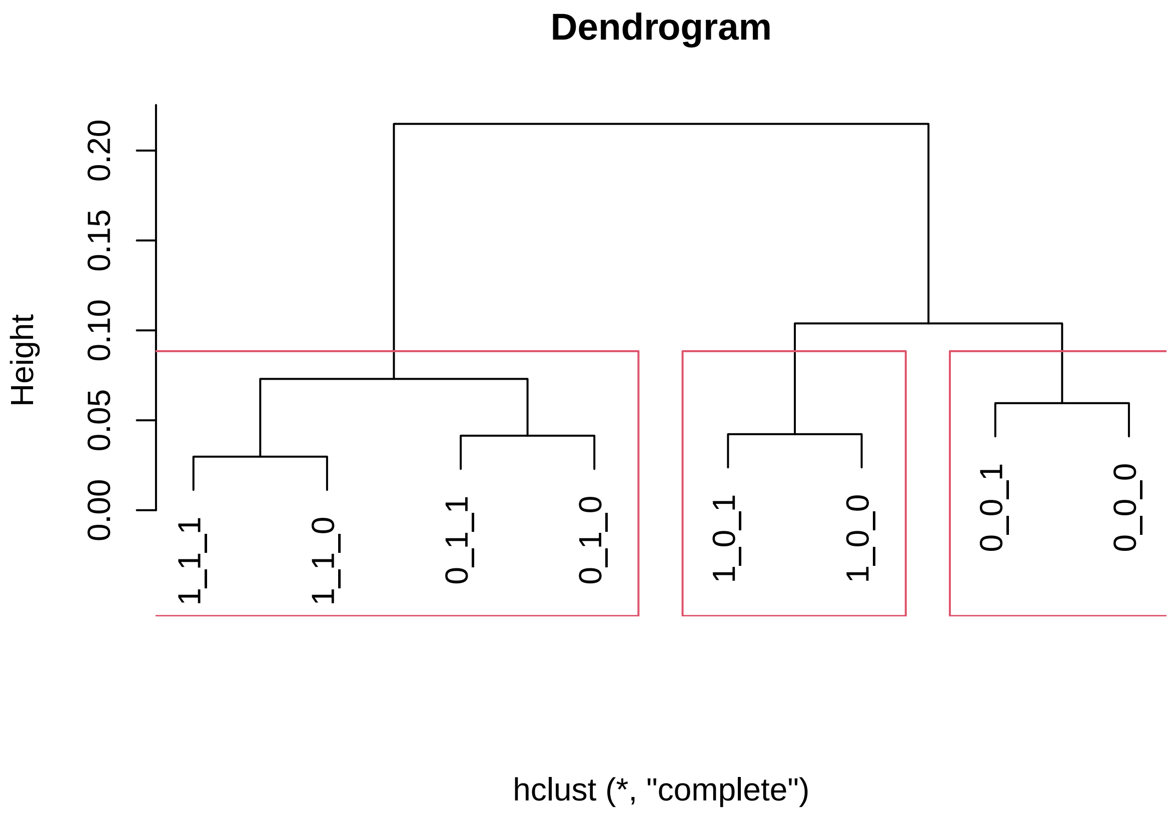
4/ Investigating the 3-way interacting effects in GenHealth_4 sub-dataset:

Creating contigency tables based on the combination of these three selected binary variable

```
## Warning: package 'iNZightTools' was built under R version 4.2.2

## 
## Attaching package: 'iNZightTools'

## The following object is masked from 'package:stats':
## 
##     filter

##                1         2         3          4           5           6
## 0_1_1 0.04566066 0.4501740 0.3413477 0.12664769 0.026152062 0.006959823
## 0_0_1 0.08429841 0.5496903 0.2631295 0.07971990 0.014543496 0.005386480
## 1_1_1 0.04721550 0.5019370 0.3307506 0.09443099 0.017433414 0.005326877
## 0_0_0 0.13165829 0.5257956 0.2366834 0.07755444 0.019262982 0.005360134
## 0_1_0 0.06501057 0.4490839 0.3058492 0.13125440 0.031183932 0.011804087
## 1_0_1 0.08446866 0.6130790 0.2438692 0.05040872 0.008174387 0.000000000
## 1_0_0 0.11475410 0.5950820 0.2213115 0.05081967 0.011475410 0.004918033
## 1_1_0 0.06459330 0.4904306 0.3118022 0.10127592 0.023923445 0.004784689
## Total 0.07313906 0.4909408 0.2988280 0.10421286 0.022458030 0.006841939
##                 7            8            9           10
## 0_1_1 0.001265422 0.0010545186 0.0007381630 0.0000000000
## 0_0_1 0.000000000 0.0016159440 0.0010772960 0.0005386480
## 1_1_1 0.000968523 0.0009685230 0.0002421308 0.0007263923
## 0_0_0 0.002010050 0.0013400335 0.0003350084 0.0000000000
## 0_1_0 0.004580691 0.0007047216 0.0003523608 0.0001761804
## 1_0_1 0.000000000 0.0000000000 0.0000000000 0.0000000000
## 1_0_0 0.001639344 0.0000000000 0.0000000000 0.0000000000
## 1_1_0 0.001594896 0.0000000000 0.0015948963 0.0000000000
## Total 0.001805512 0.0010136205 0.0005701615 0.0001900538

##                1         2         3          4           5           6
## 0_1   0.05290586 0.4497658 0.3280559 0.12837258 0.028036150 0.008773666
## 0_0   0.11349788 0.5349582 0.2468243 0.07838480 0.017453269 0.005370236
## 1_1   0.05126300 0.4992571 0.3263373 0.09602526 0.018945022 0.005200594
## 1_0   0.09821429 0.6049107 0.2336310 0.05059524 0.009672619 0.002232143
## Total 0.07313906 0.4909408 0.2988280 0.10421286 0.022458030 0.006841939
##                  7            8            9           10
## 0_1   0.0025067617 0.0009235438 0.0005937067 6.596741e-05
## 0_0   0.0012392853 0.0014458329 0.0006196427 2.065476e-04
## 1_1   0.0011144131 0.0007429421 0.0005572065 5.572065e-04
## 1_0   0.0007440476 0.0000000000 0.0000000000 0.000000e+00
## Total 0.0018055116 0.0010136205 0.0005701615 1.900538e-04

##                1         2         3          4          5           6
## 0_1   0.05653228 0.4781752 0.3193392 0.11344347 0.02288572 0.006517126
## 1_1   0.05283717 0.5187089 0.3176398 0.08778783 0.01603618 0.004523026
## 0_0   0.09917568 0.4884080 0.2703933 0.10372660 0.02507299 0.008500773
## 1_0   0.08100858 0.5246781 0.2821888 0.08476395 0.01984979 0.004828326
## Total 0.07313906 0.4909408 0.2988280 0.10421286 0.02245803 0.006841939
##                  7            8            9           10
## 0_1   0.0009093665 0.0012124886 0.0008335859 1.515611e-04
## 1_1   0.0008223684 0.0008223684 0.0002055921 6.167763e-04
## 0_0   0.0032629229 0.0010303967 0.0003434656 8.586639e-05
## 1_0   0.0016094421 0.0000000000 0.0010729614 0.000000e+00
## Total 0.0018055116 0.0010136205 0.0005701615 1.900538e-04
```

```
##                1         2         3          4          5           6
## 1_1   0.04613237 0.4658782 0.3381327 0.11687358 0.02350694 0.006464409
## 0_1   0.08432651 0.5601529 0.2599505 0.07488194 0.01349224 0.004497414
## 0_0   0.13009119 0.5322188 0.2352584 0.07507599 0.01854103 0.005319149
## 1_0   0.06493506 0.4565657 0.3069264 0.12582973 0.02987013 0.010533911
## Total 0.07313906 0.4909408 0.2988280 0.10421286 0.02245803 0.006841939
##                 7            8            9           10
## 1_1   0.001175347 0.0010284287 0.0005876735 0.0002203776
## 0_1   0.000000000 0.0013492242 0.0008994828 0.0004497414
## 0_0   0.001975684 0.0012158055 0.0003039514 0.0000000000
## 1_0   0.004040404 0.0005772006 0.0005772006 0.0001443001
## Total 0.001805512 0.0010136205 0.0005701615 0.0001900538
```

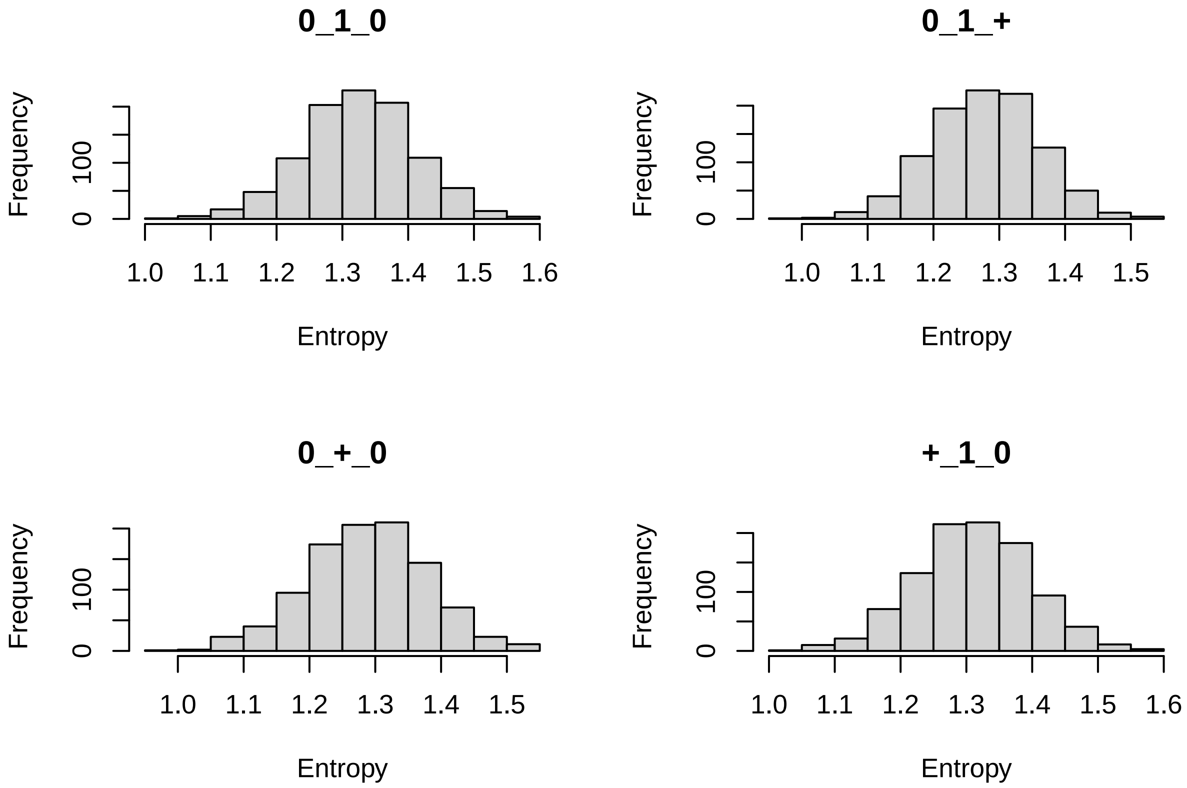
3-way HC-tree of GenHealth_4:



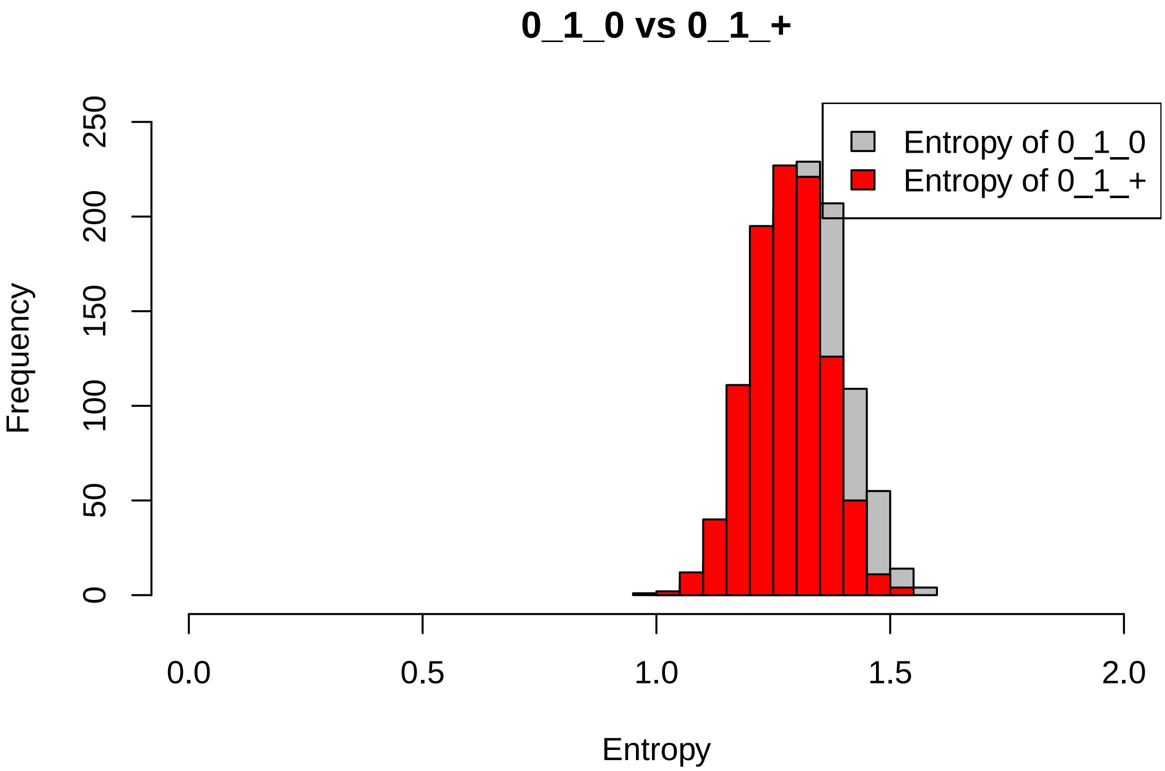
Comparing the entropies of categorical variables to see the effects

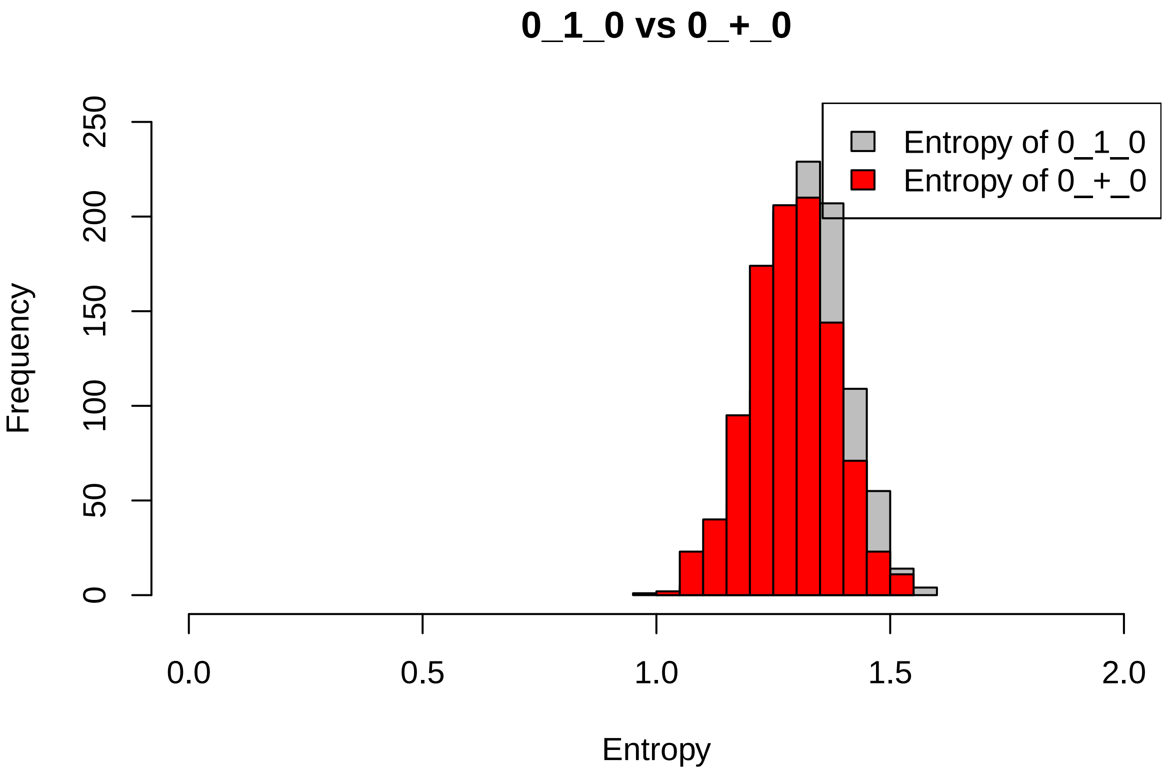
No HD __ Yes HighBP __ No HighChol

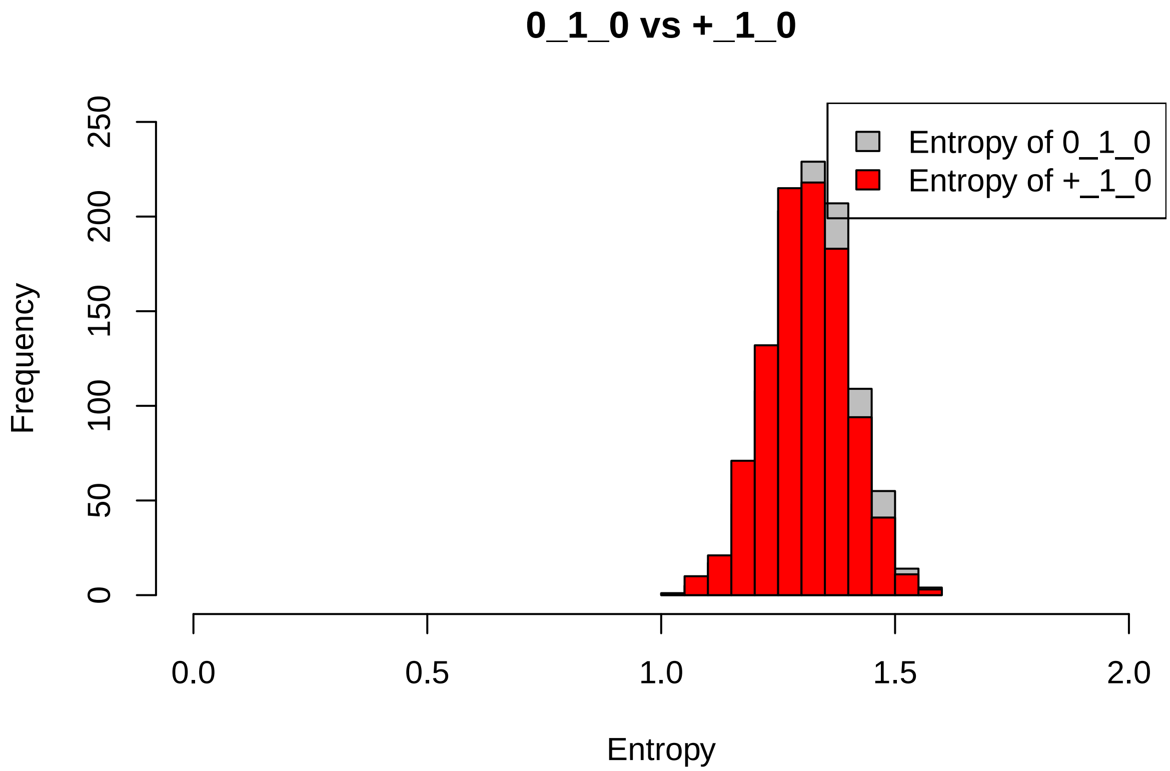
```
## Warning: package 'DescTools' was built under R version 4.2.2
```



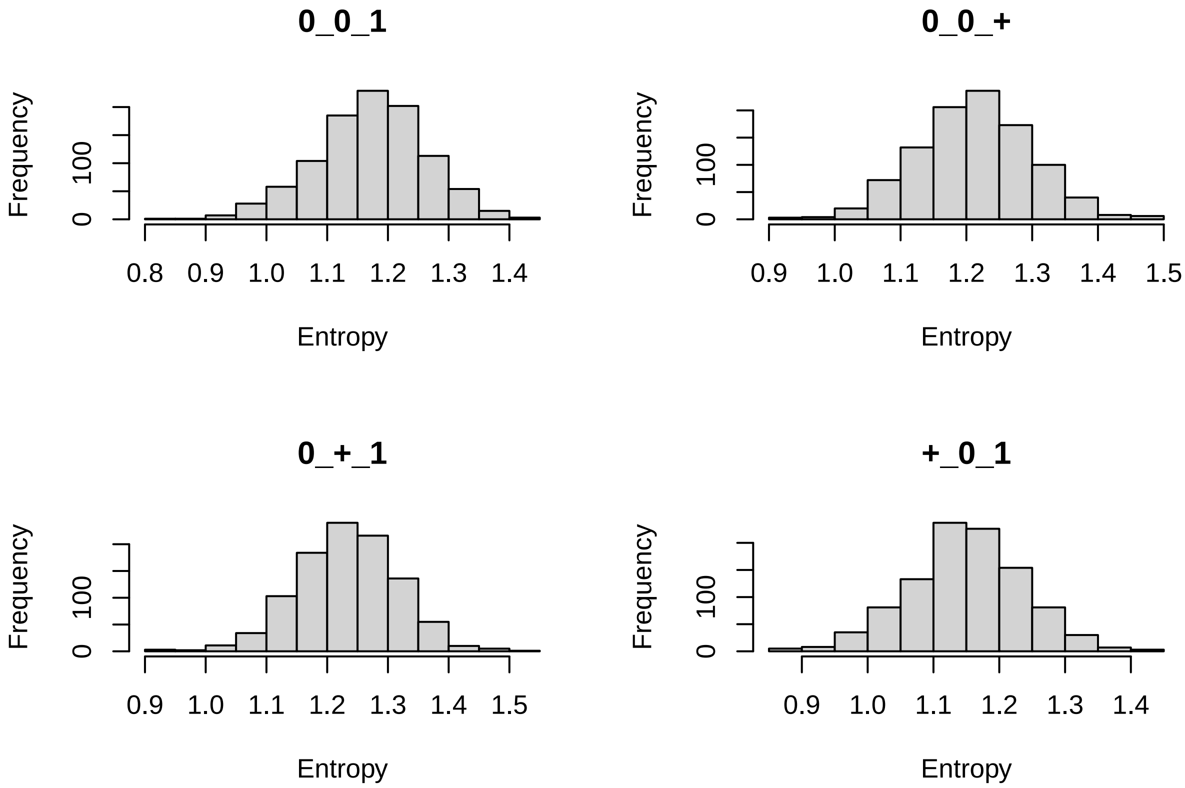
Looking at these graphs below, it seems that there isn't much shift happening in the two data. There is a small shift to the left when missing HighChol. Thus, missing 1 bi-variable has little effect on 0_1_0. Since the graphs mostly overlap the origin, there is no interaction effect on the BMI distribution.


0_1_0 vs 0_1_+
Entropy of 0_1_0
Entropy of 0_1_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

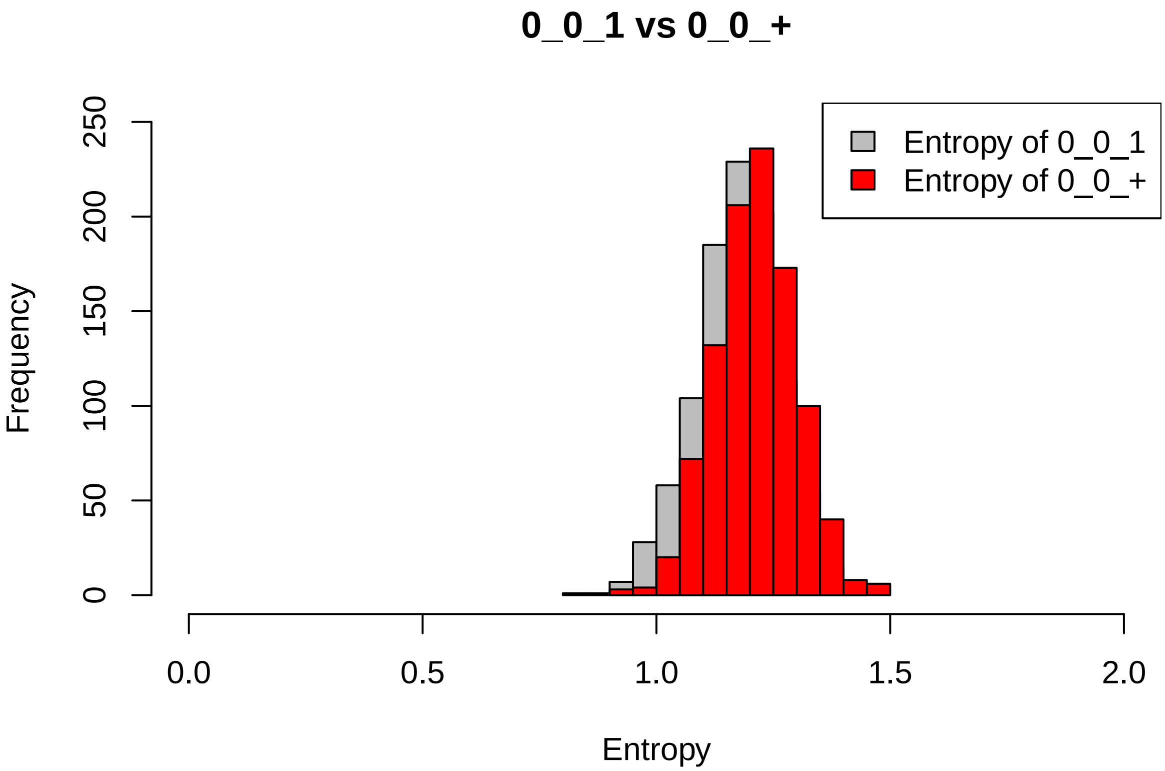

0_1_0 vs 0_+_0
Entropy of 0_1_0
Entropy of 0_+_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

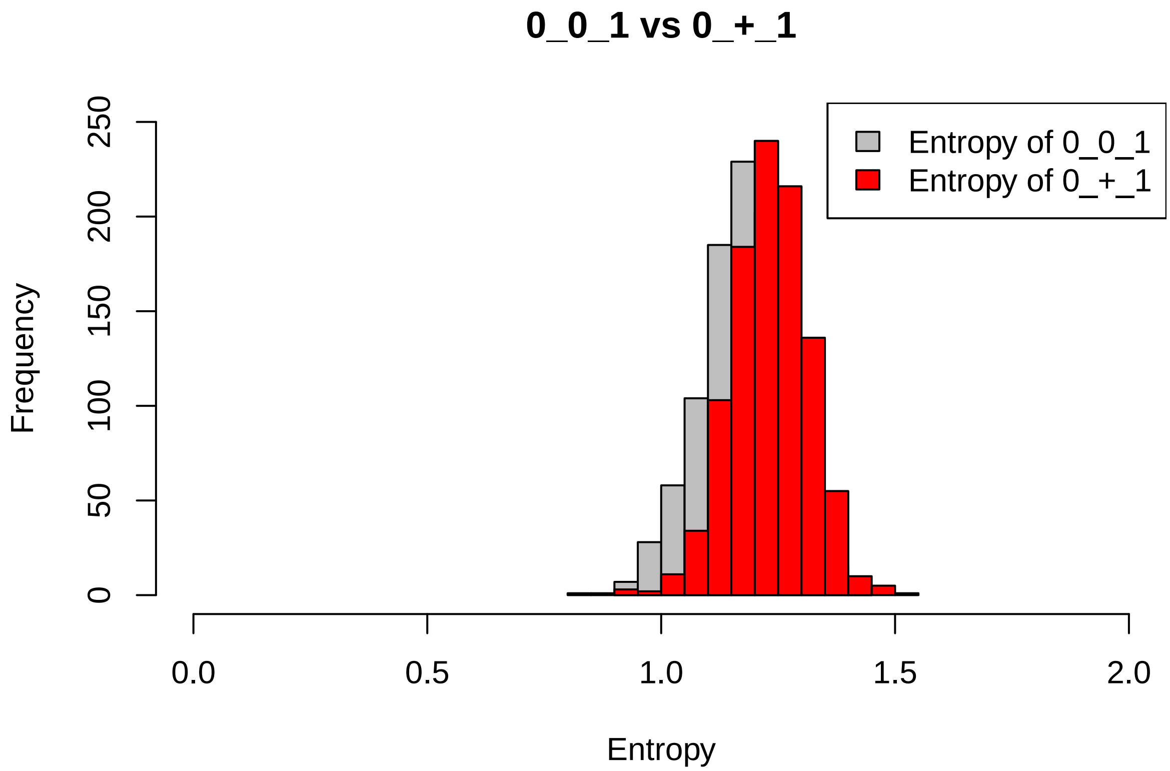

0_1_0 vs +_1_0
Entropy of 0_1_0
Entropy of +_1_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

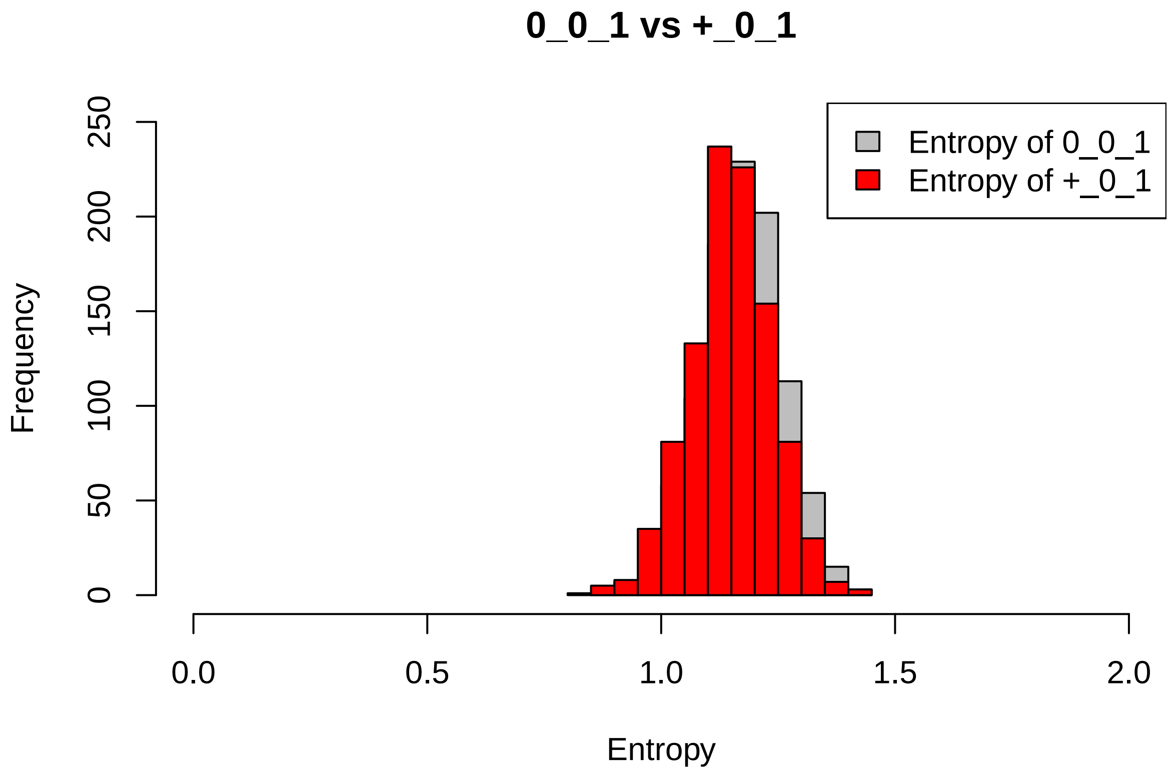
No HD __ No HighBP __ Yes HighChol



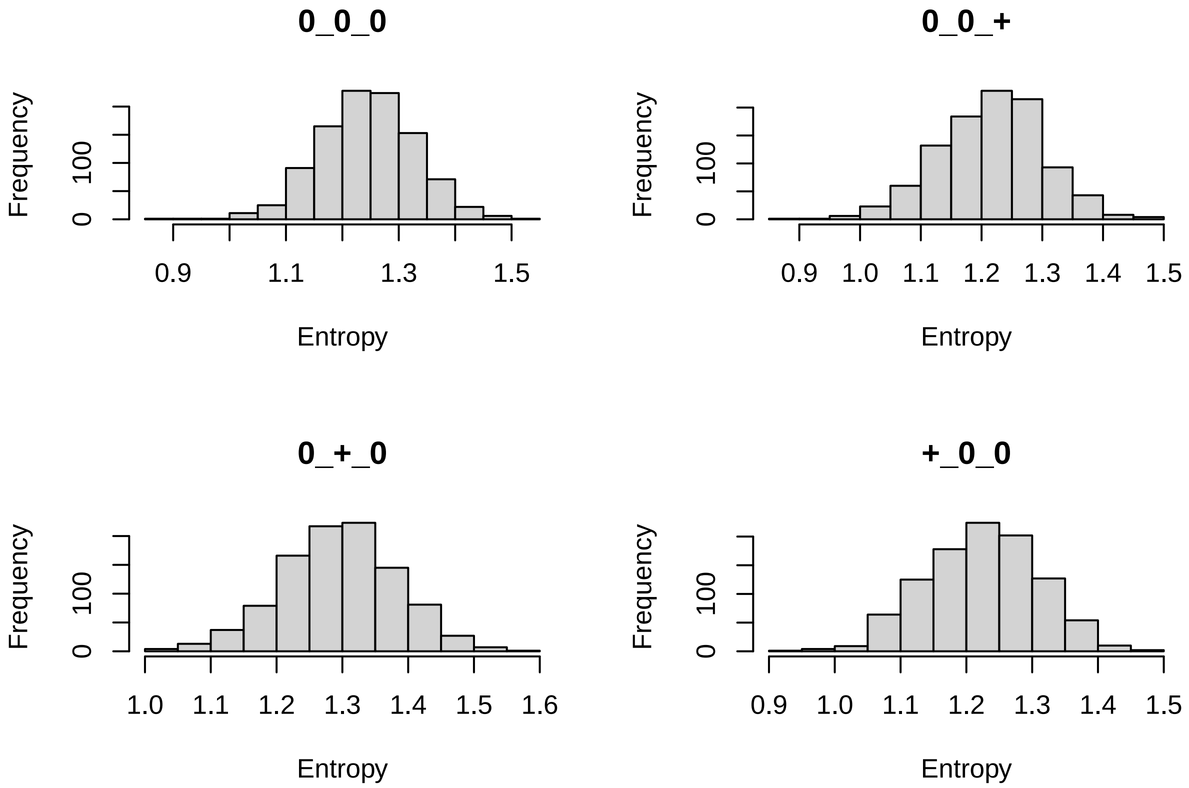
Looking at these graphs below, there is clearly a shift to the right when HighBP or HighChol is missing, which means the entropy level goes up and makes it less accurate. However, missing HD doesn't change much. Since the graphs mostly doesn't overlap the origin, there is an interaction effect on the BMI distribution.


0_0_1 vs 0_0_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 0_0_1
Entropy of 0_0_+

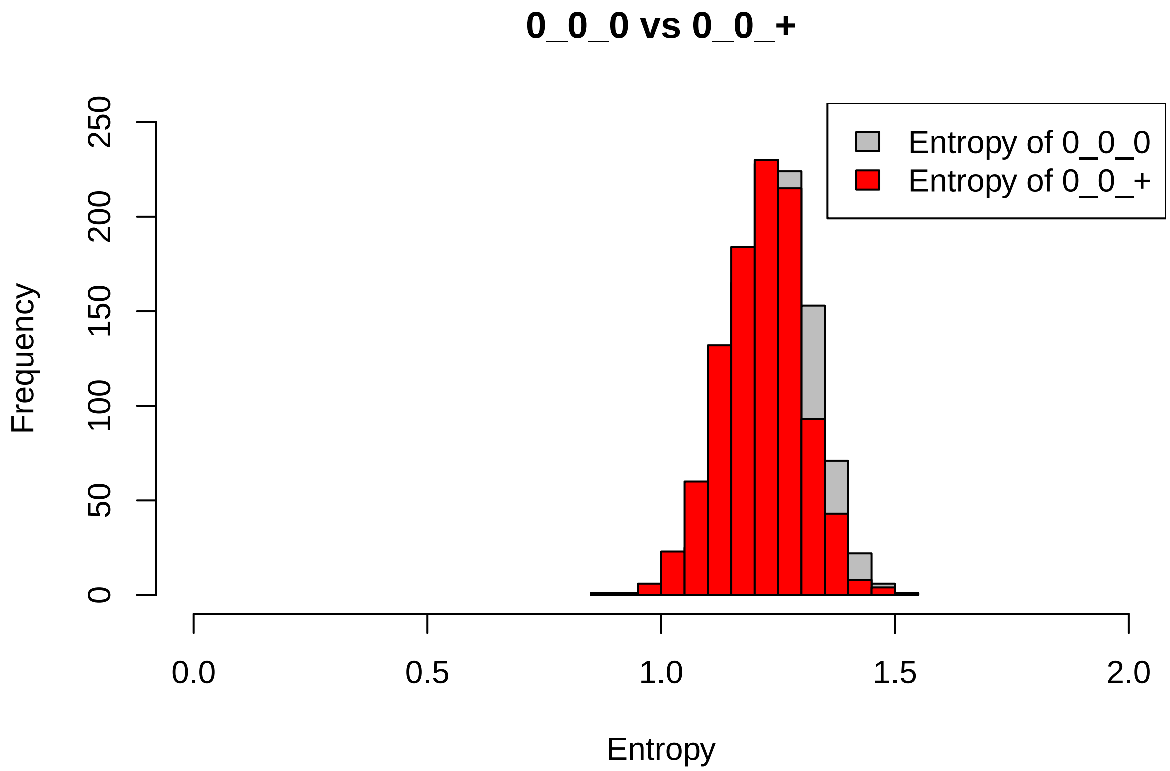

0_0_1 vs 0_+_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 0_0_1
Entropy of 0_+_1

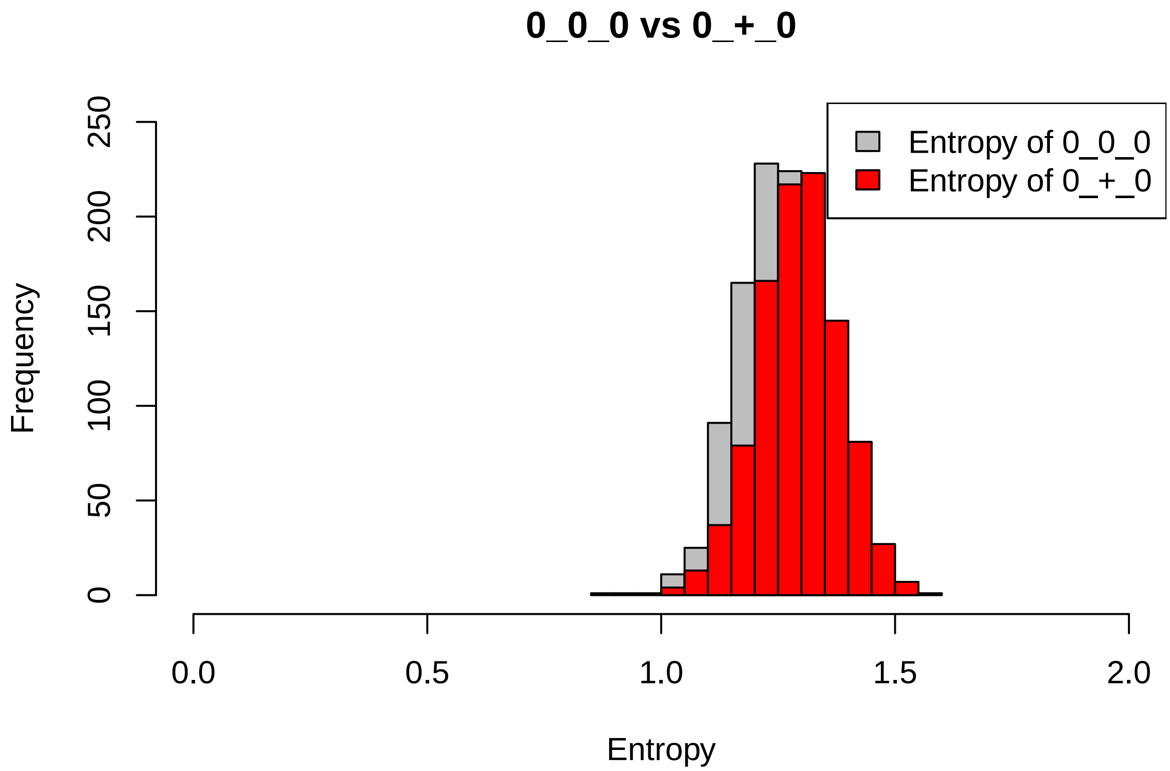

0_0_1 vs +_0_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 0_0_1
Entropy of +_0_1

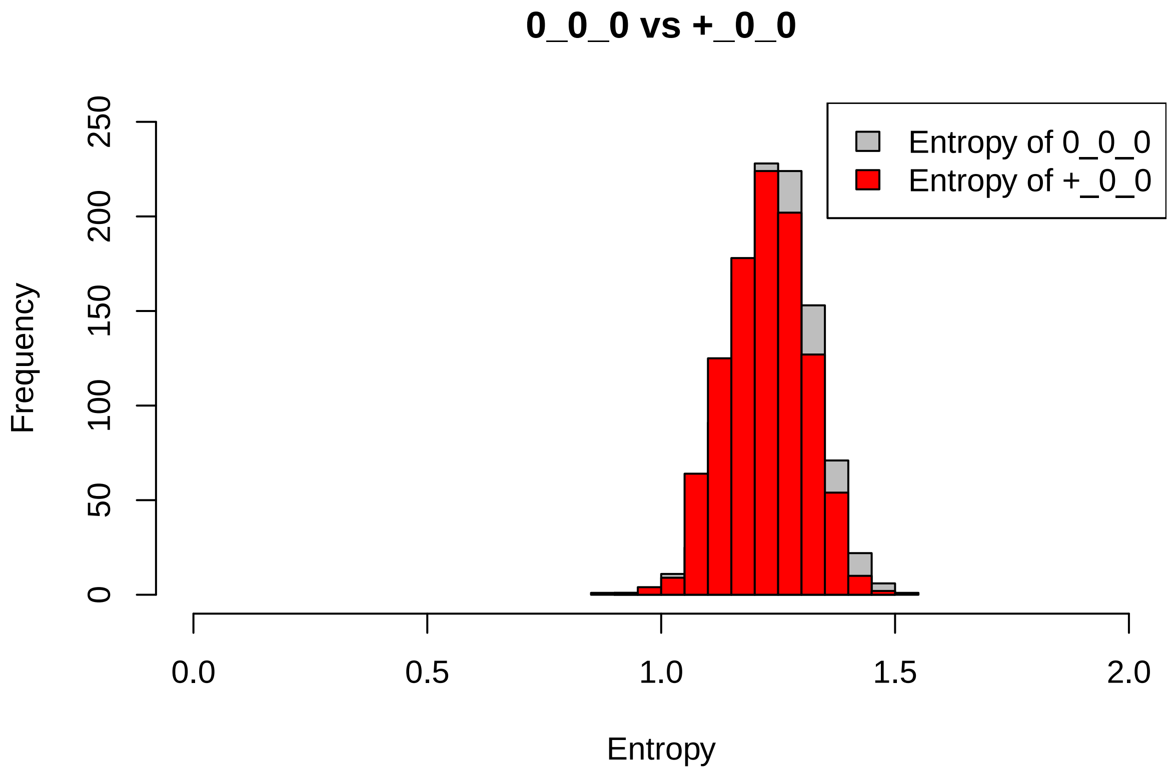
No HD __ No HighBP __ No HighChol



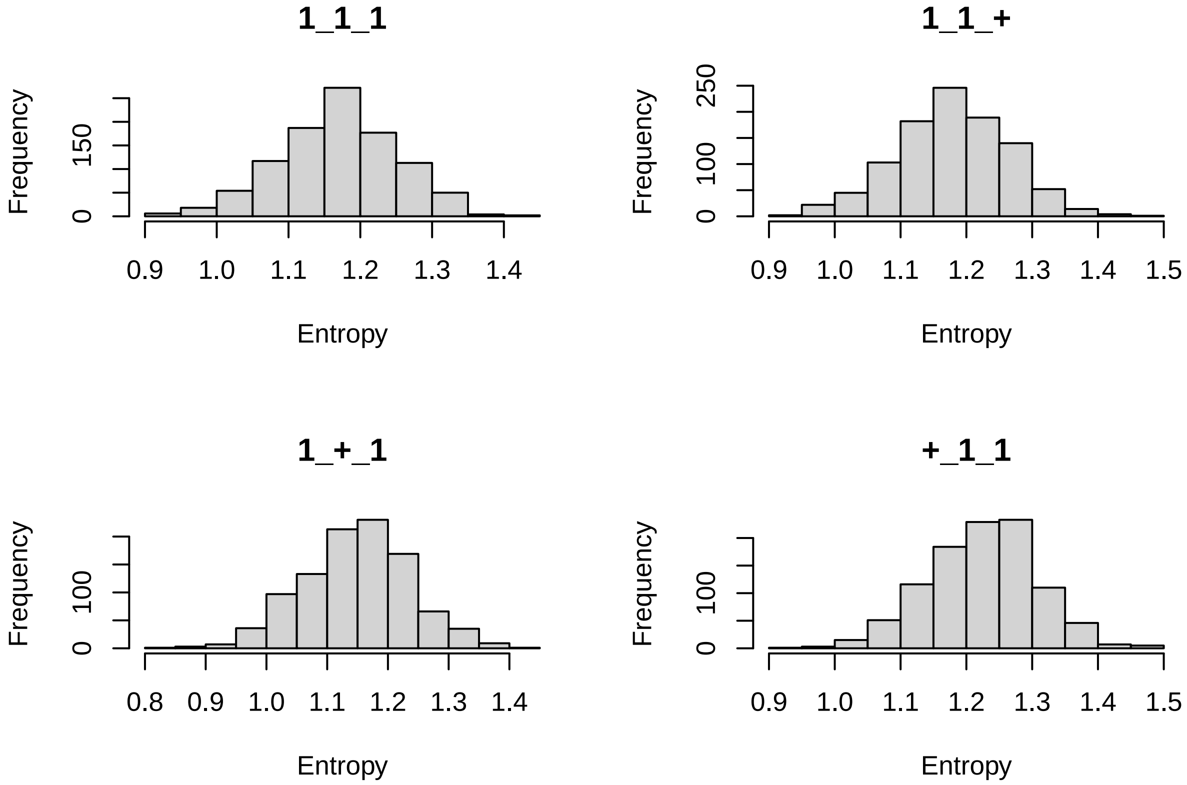
Looking at these graphs below, there is not much change happening. There is a small shift to the right when missing HighBP, but not much. Since the graphs mostly overlap to the origin, there is no interaction effect on the BMI distribution.


0_0_0 vs 0_0_+
Entropy of 0_0_0
Entropy of 0_0_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

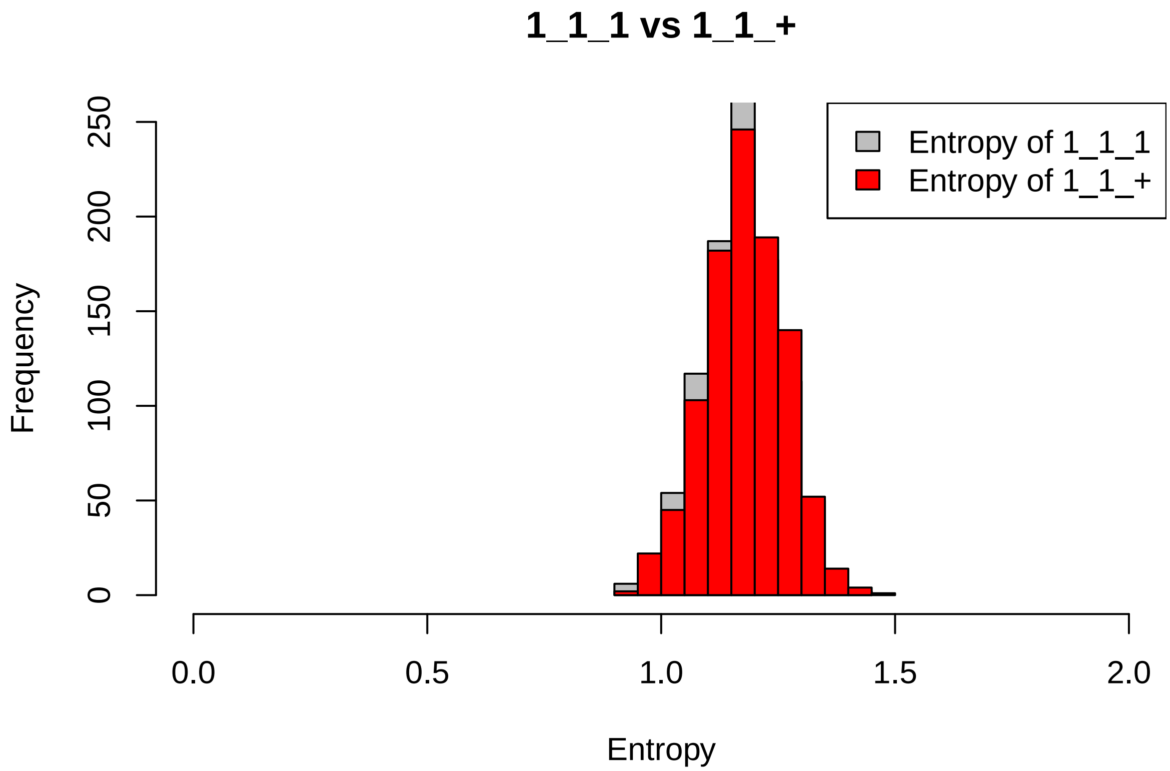

0_0_0 vs 0_+_0
Entropy of 0_0_0
Entropy of 0_+_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

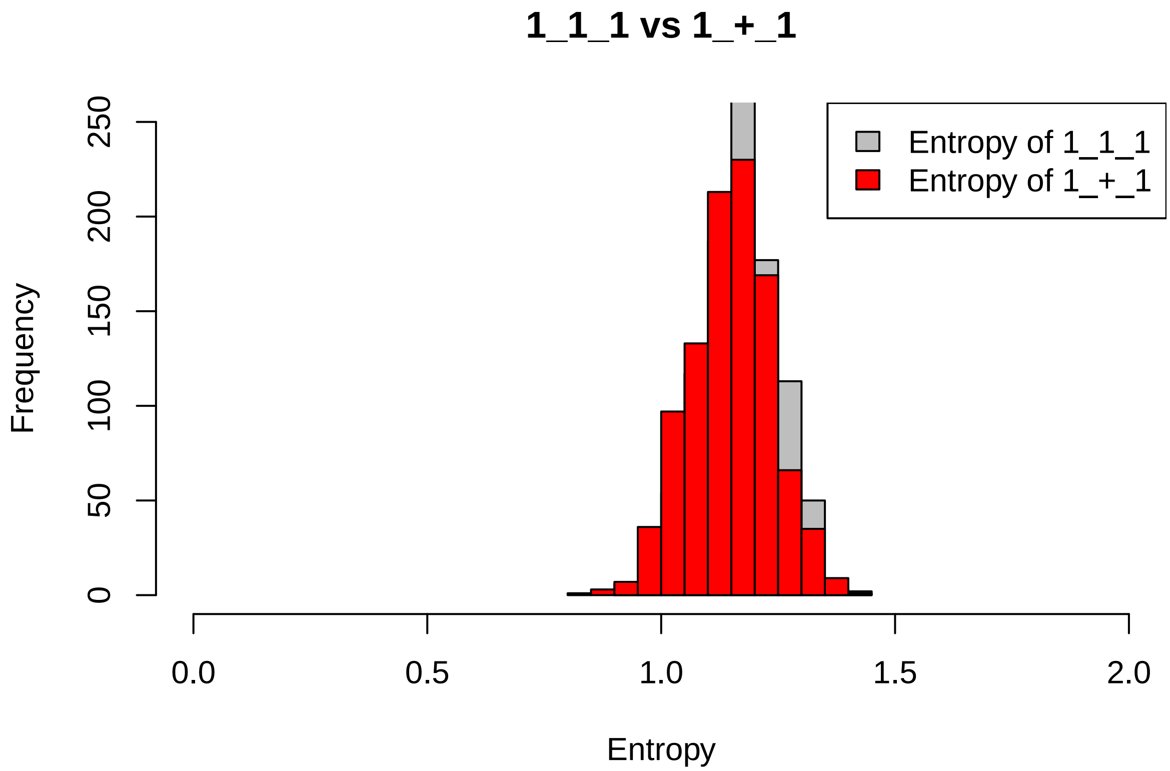

0_0_0 vs +_0_0
Entropy of 0_0_0
Entropy of +_0_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

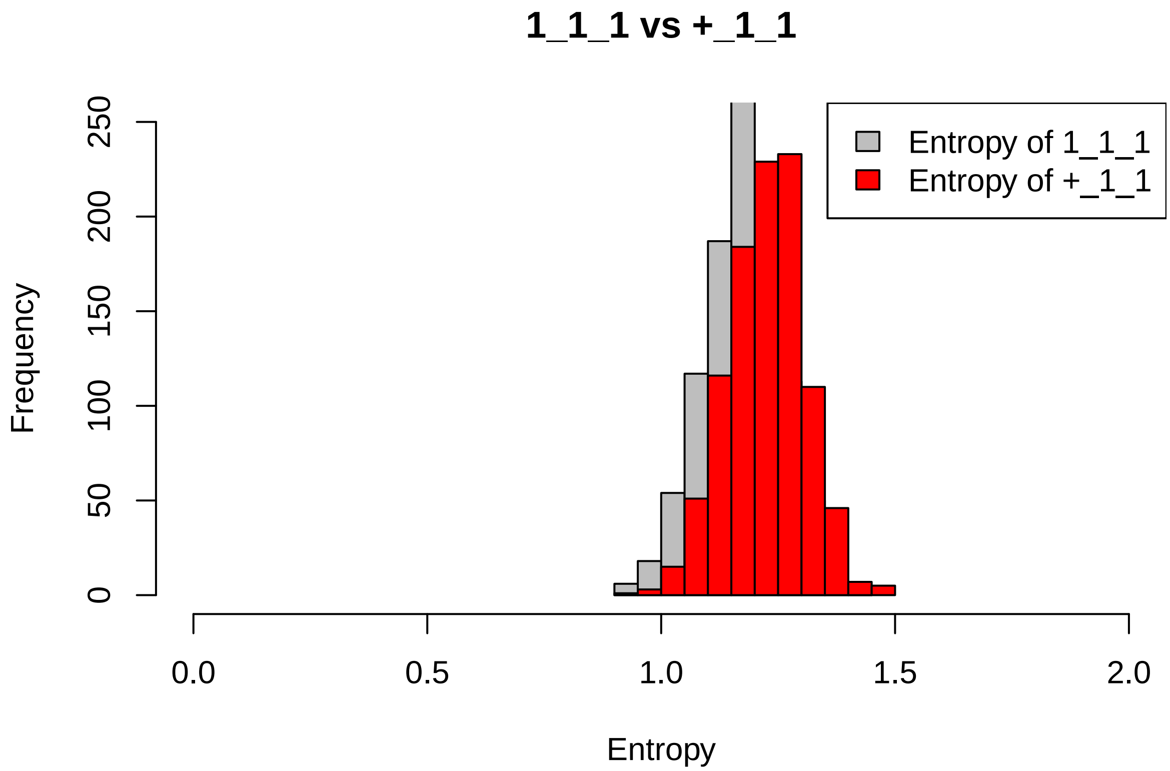
Yes HD __ Yes HighBP __ Yes HighChol



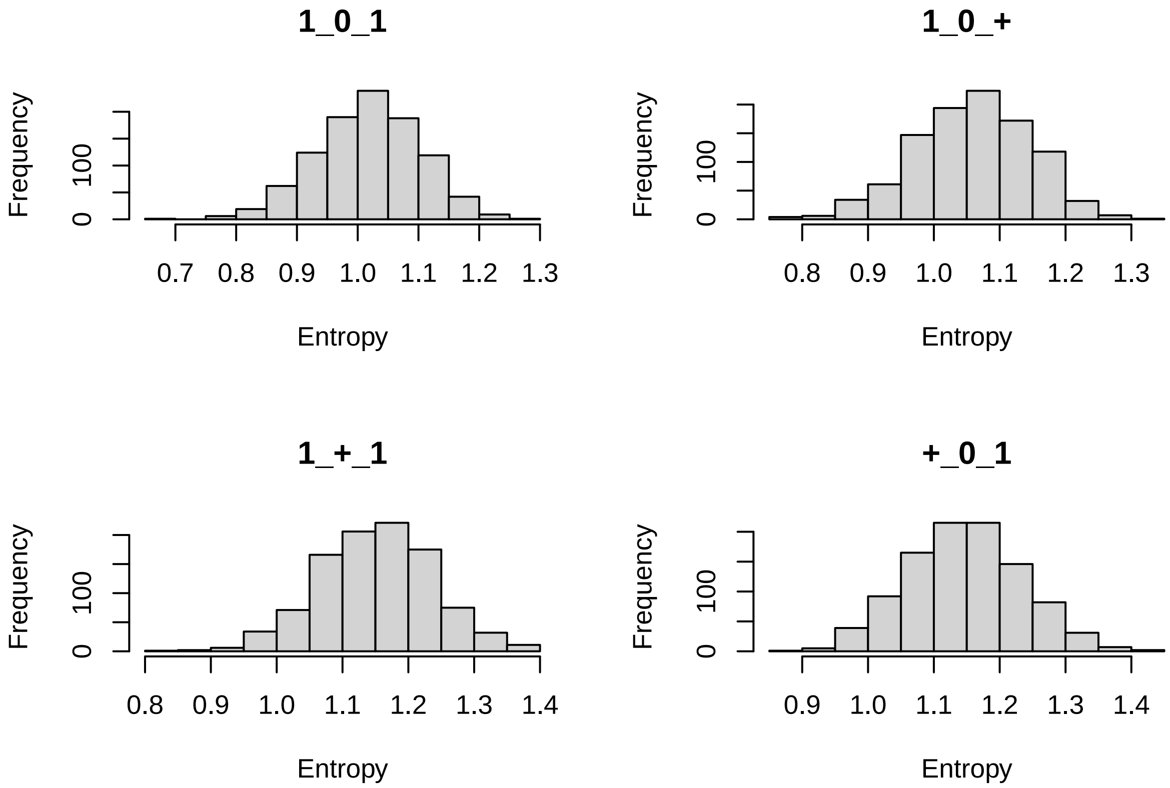
Looking at these graphs below, there is not much shift happening. There is a small shift to the right when missing HeartDisease, but not much. Since the graphs mostly overlap the origin, there is no interaction effect on the BMI distribution.


1_1_1 vs 1_1_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 1_1_1
Entropy of 1_1_+

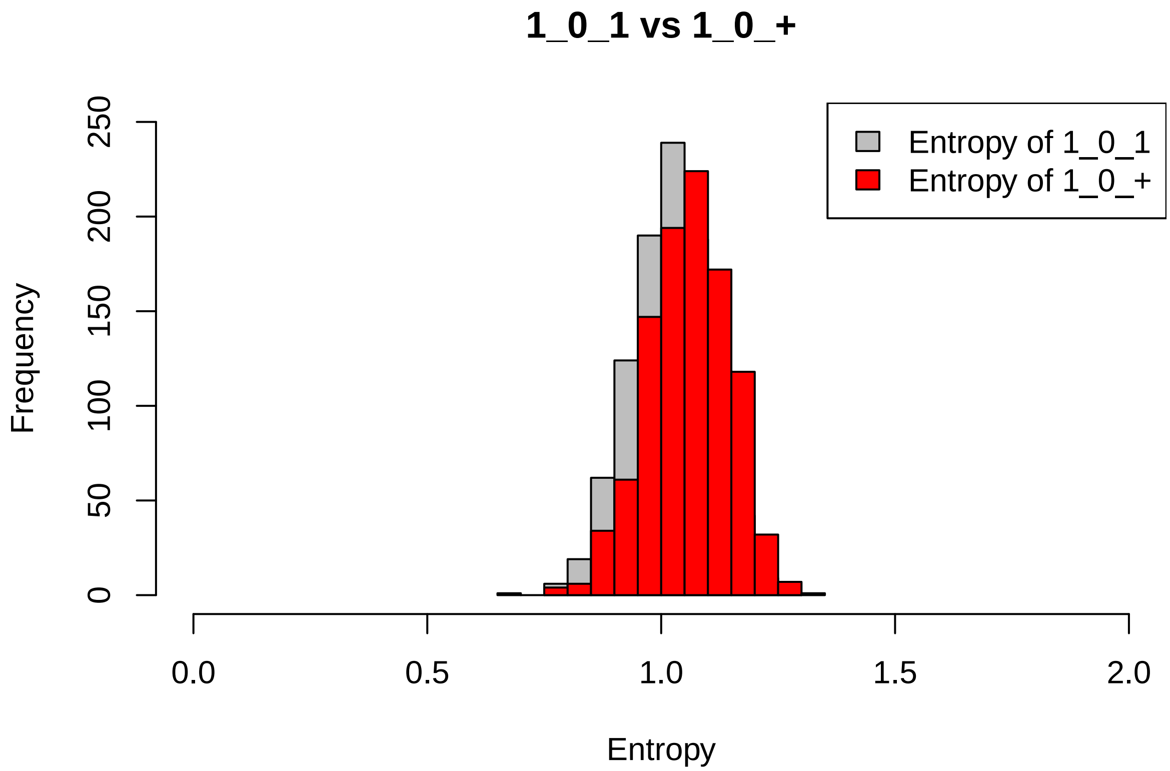

1_1_1 vs 1_+_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 1_1_1
Entropy of 1_+_1

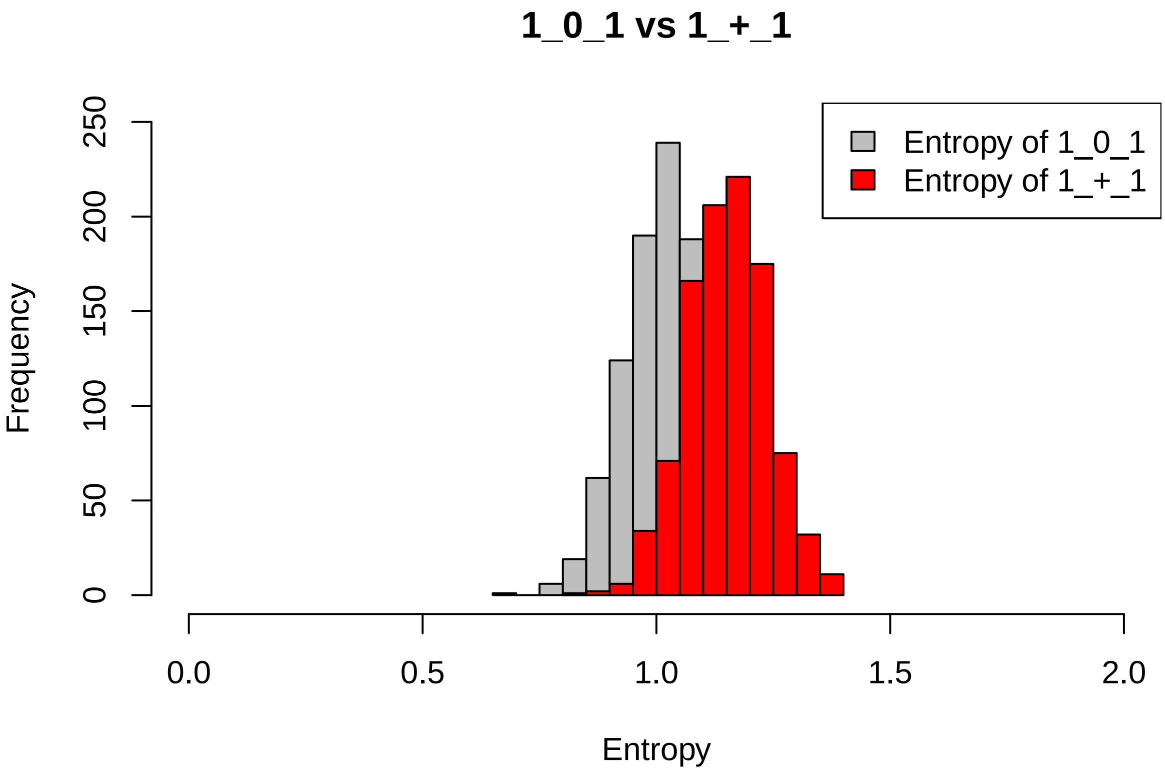

1_1_1 vs +_1_1
Entropy of 1_1_1
Entropy of +_1_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

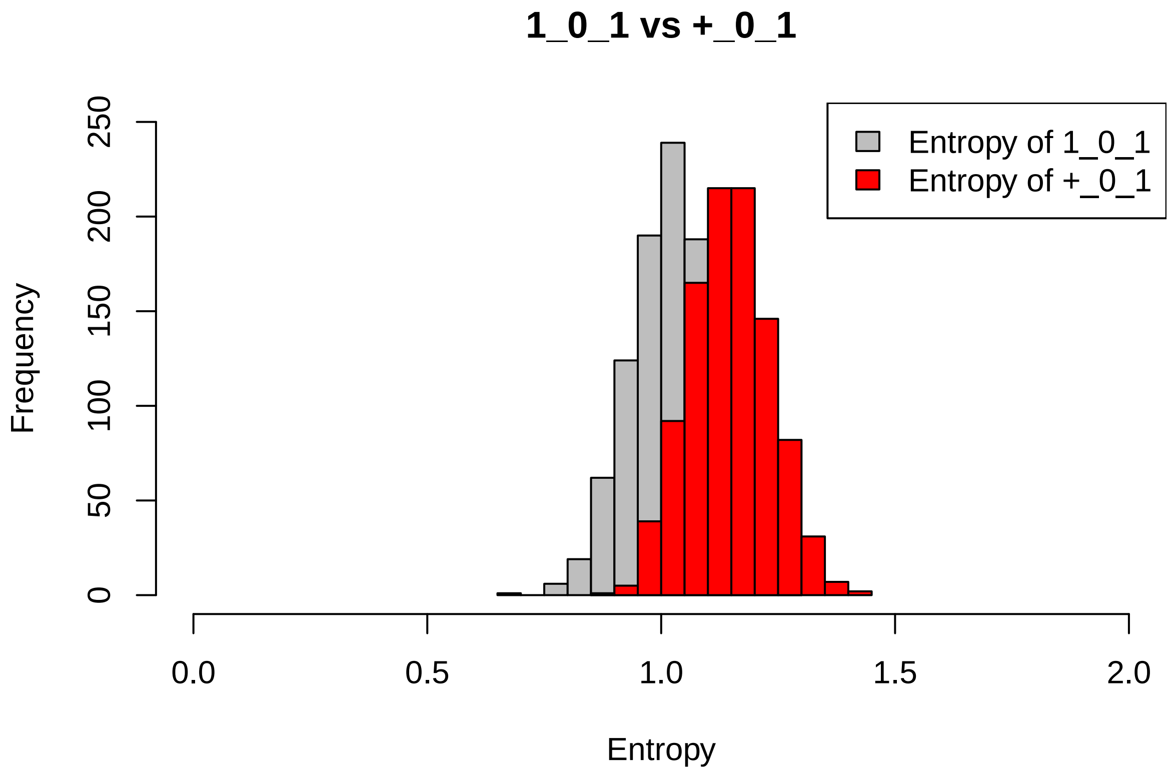
Yes HD __ No HighBP __ Yes HighChol



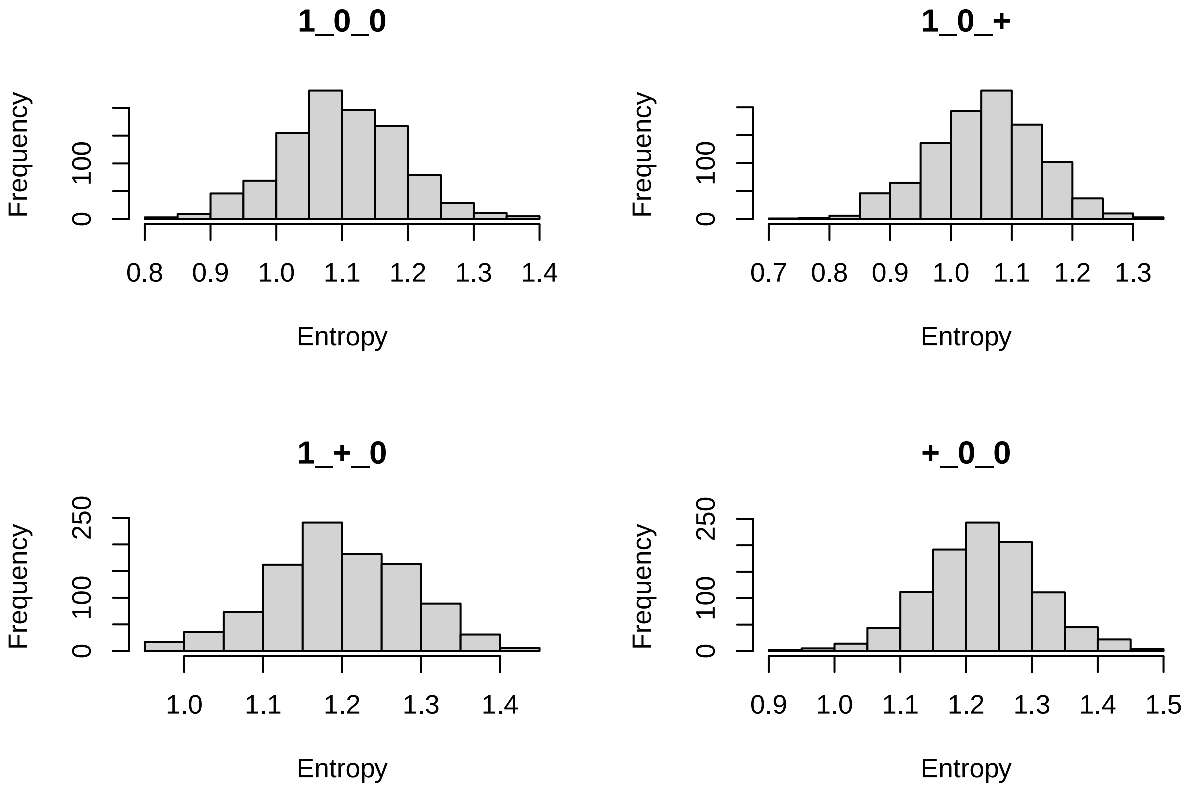
Looking at these graphs below, there is clearly a strong shift to the right when HighBP or HeartDisease or HighChol is missing, which means the entropy level goes up and makes it less accurate. Since all 3 graphs don't overlap the origin, there is an interaction effect on the BMI distribution.


1_0_1 vs 1_0_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 1_0_1
Entropy of 1_0_+

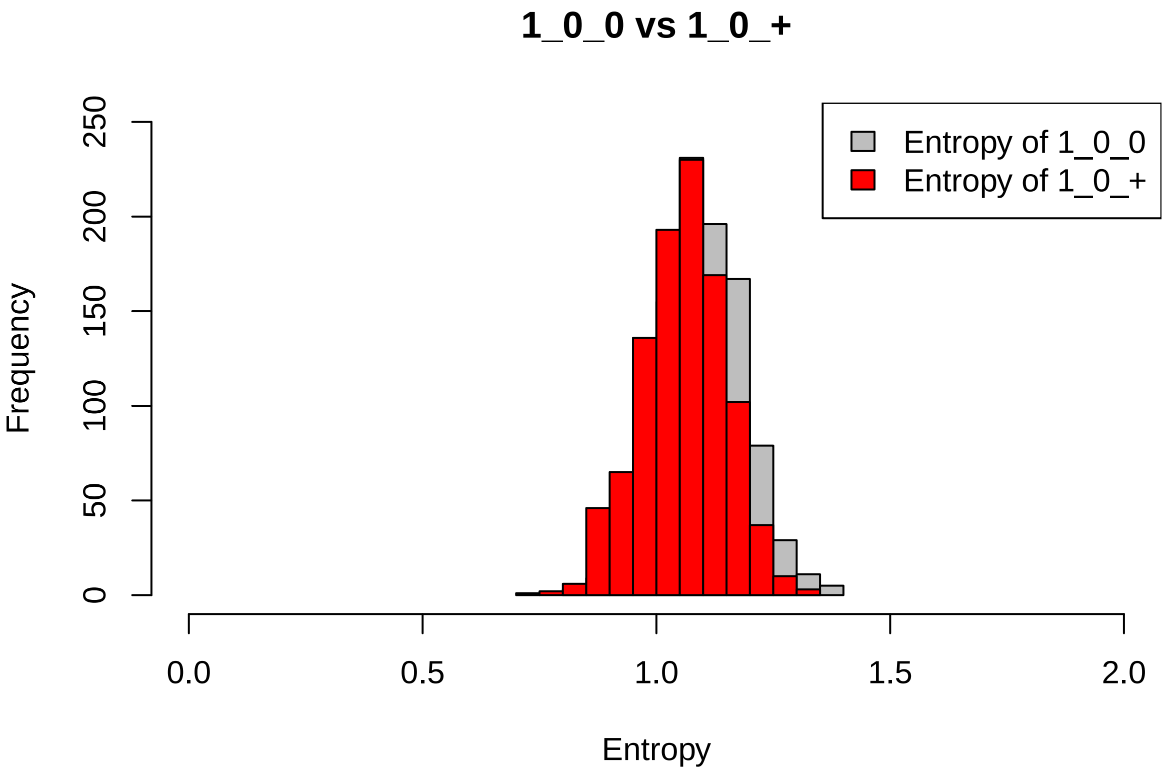

1_0_1 vs 1_+_1
Entropy of 1_0_1
Entropy of 1_+_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

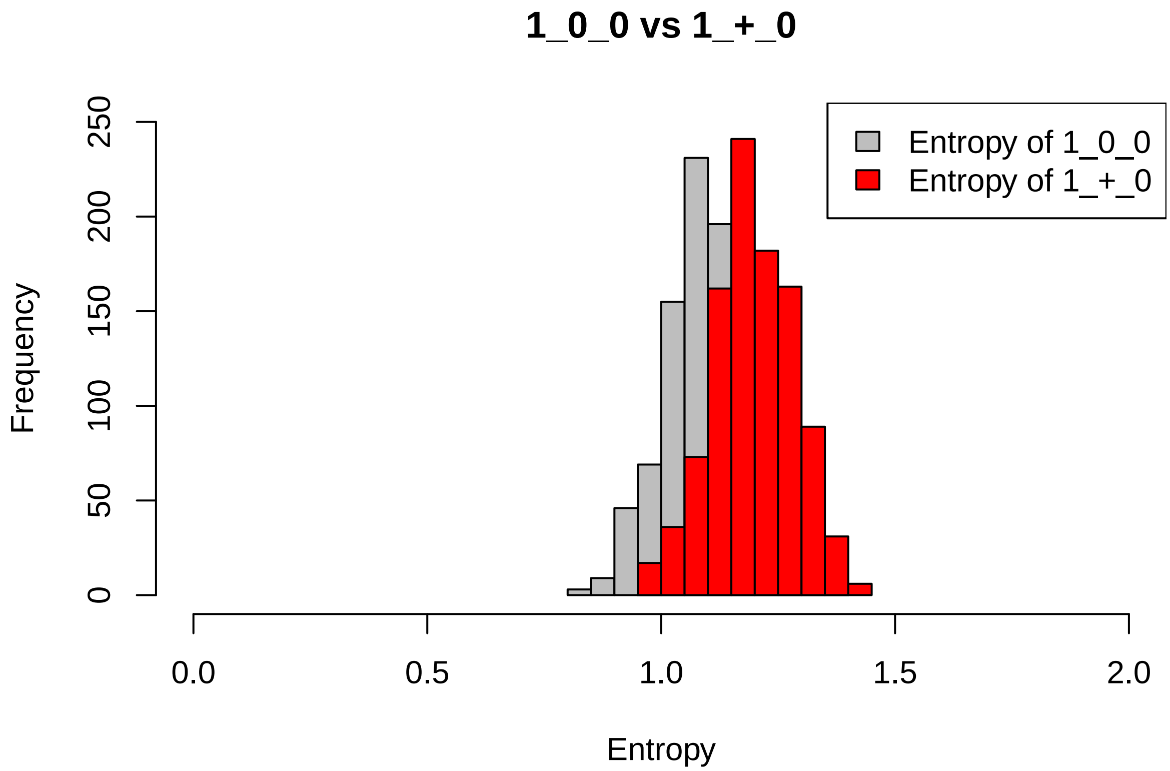

1_0_1 vs +_0_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 1_0_1
Entropy of +_0_1

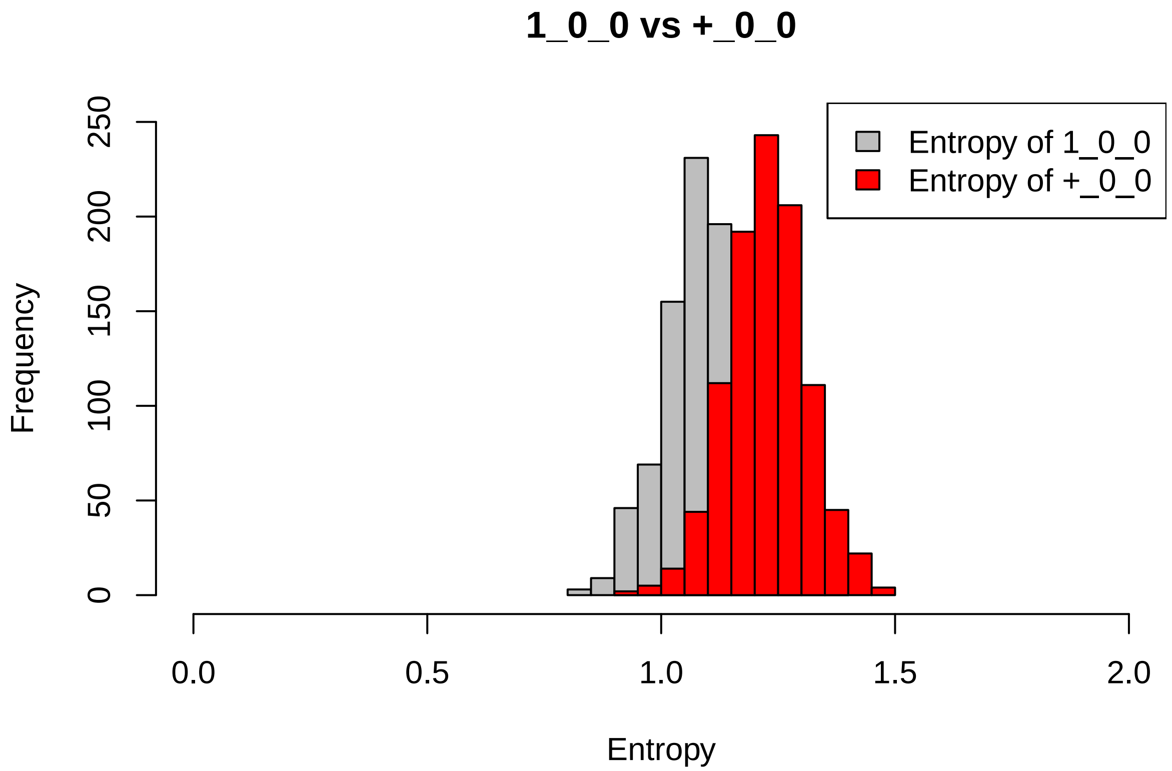
Yes HD __ No HighBP __ No HighChol



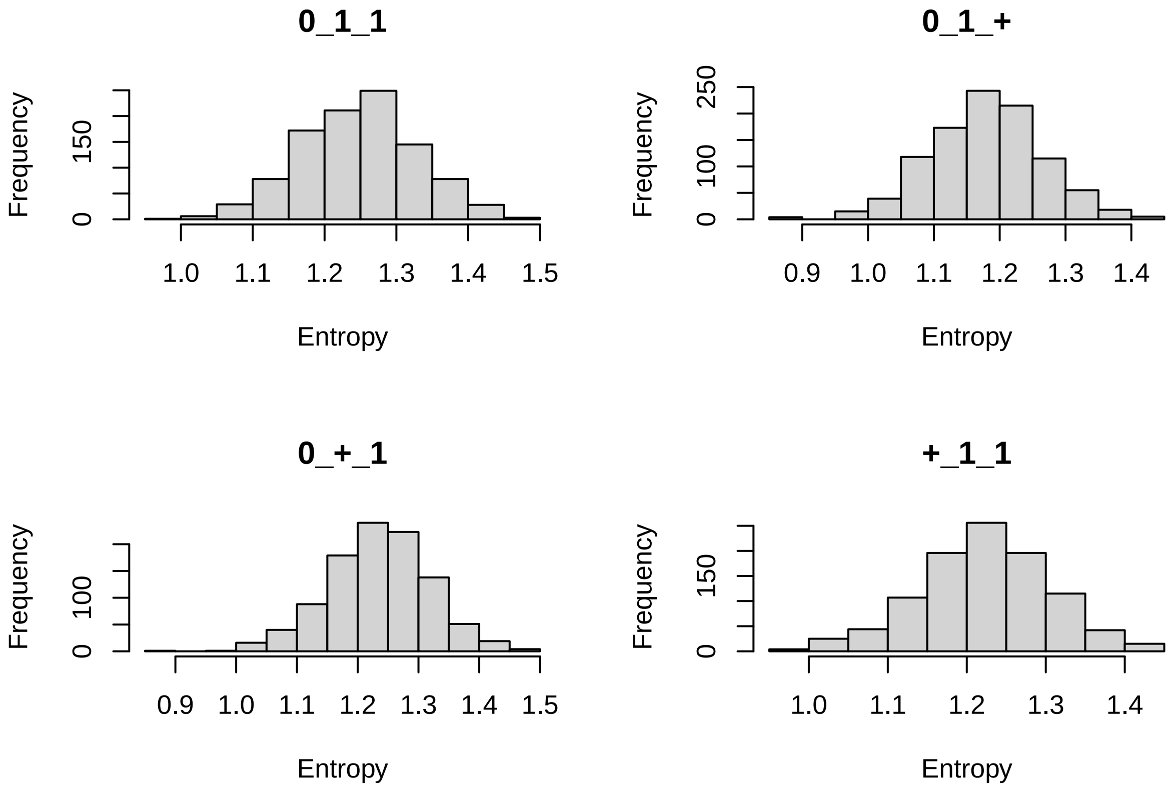
Looking at these graphs below, there is a clear shift to the right when missing HeartDisease or HighBP, which means the entropy level goes up, and makes it less accurate. However, there is not much change when missing HighChol. Since the graphs mostly don't overlap the origin, there is an interaction effect on the BMI distribution.


1_0_0 vs 1_0_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 1_0_0
Entropy of 1_0_+

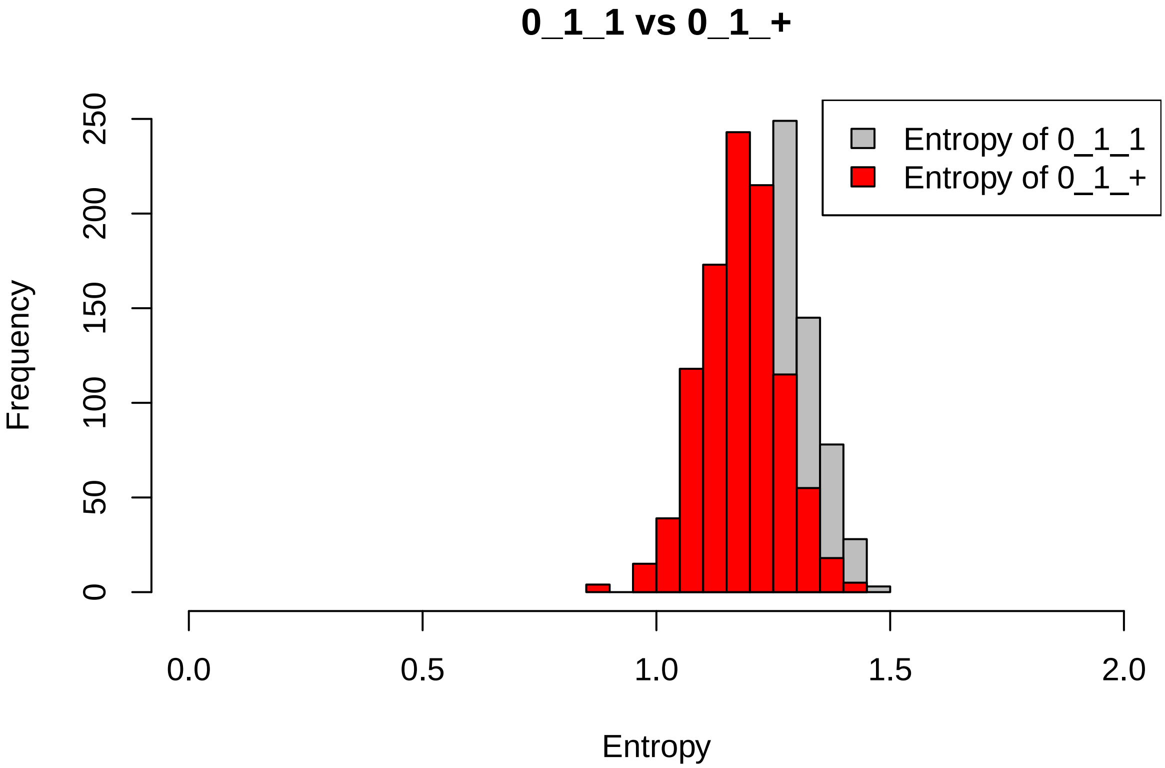

1_0_0 vs 1_+_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 1_0_0
Entropy of 1_+_0

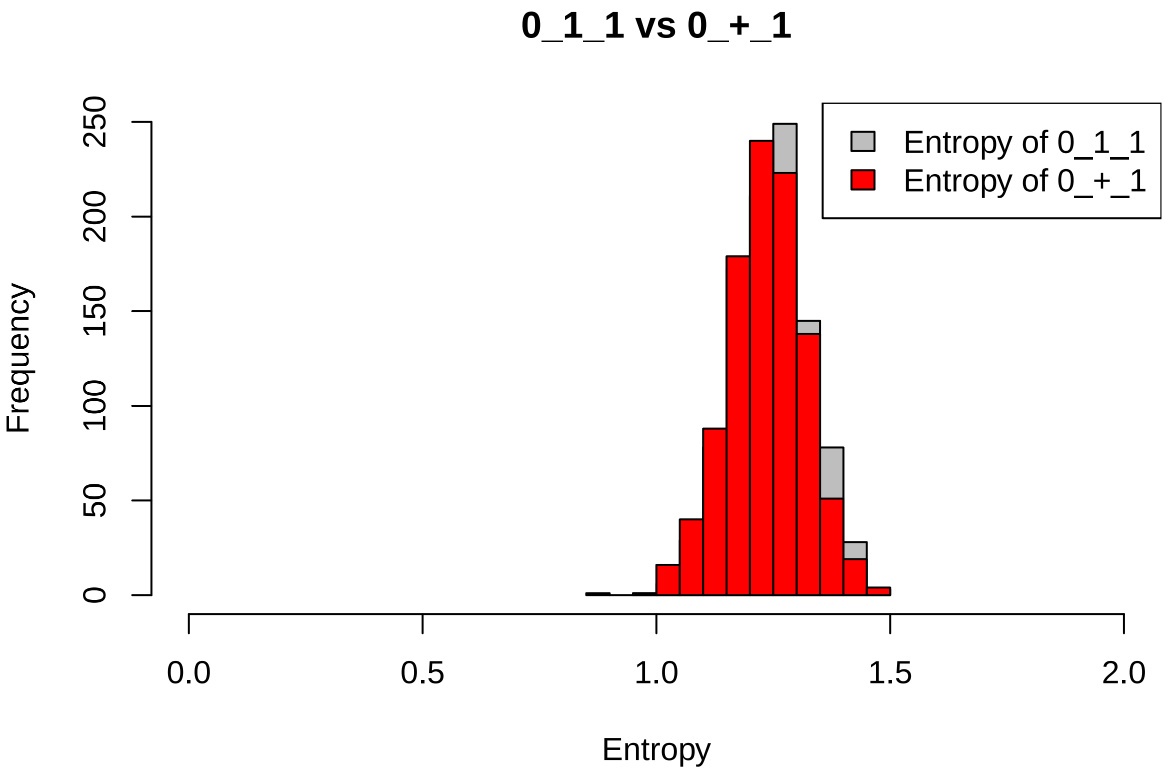

1_0_0 vs +_0_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 1_0_0
Entropy of +_0_0

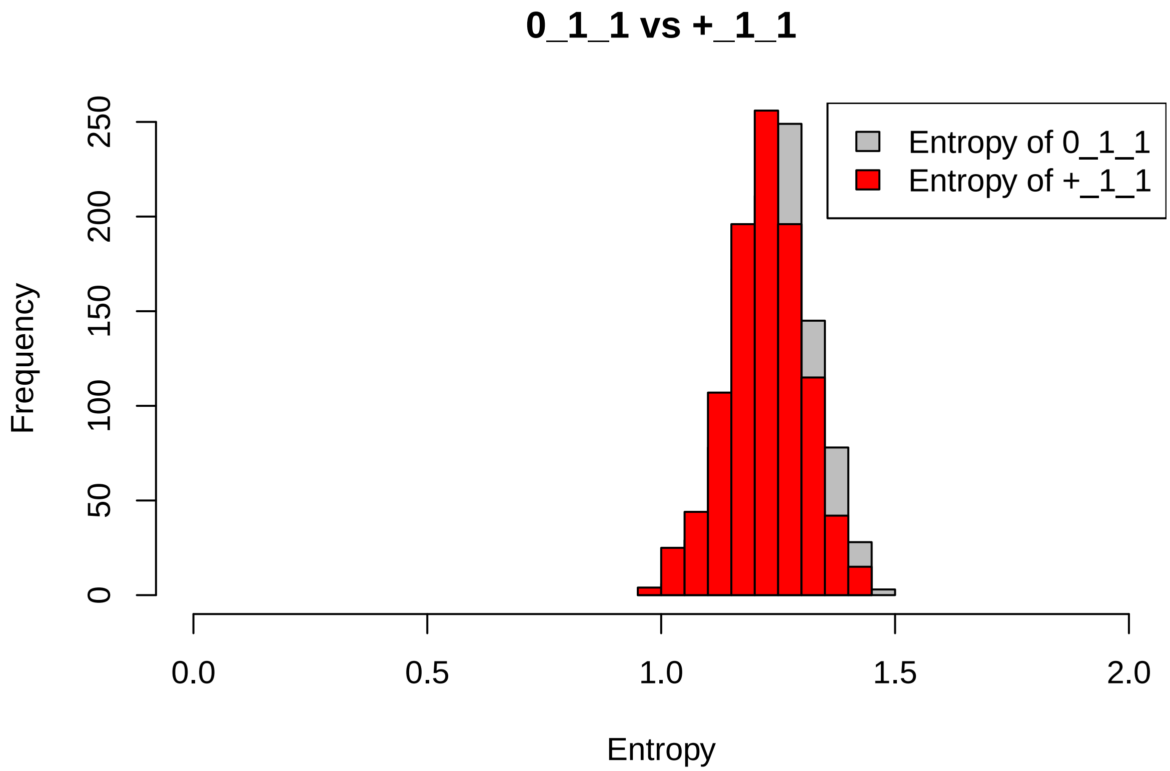
No HD __ Yes HighBP __ Yes HighChol



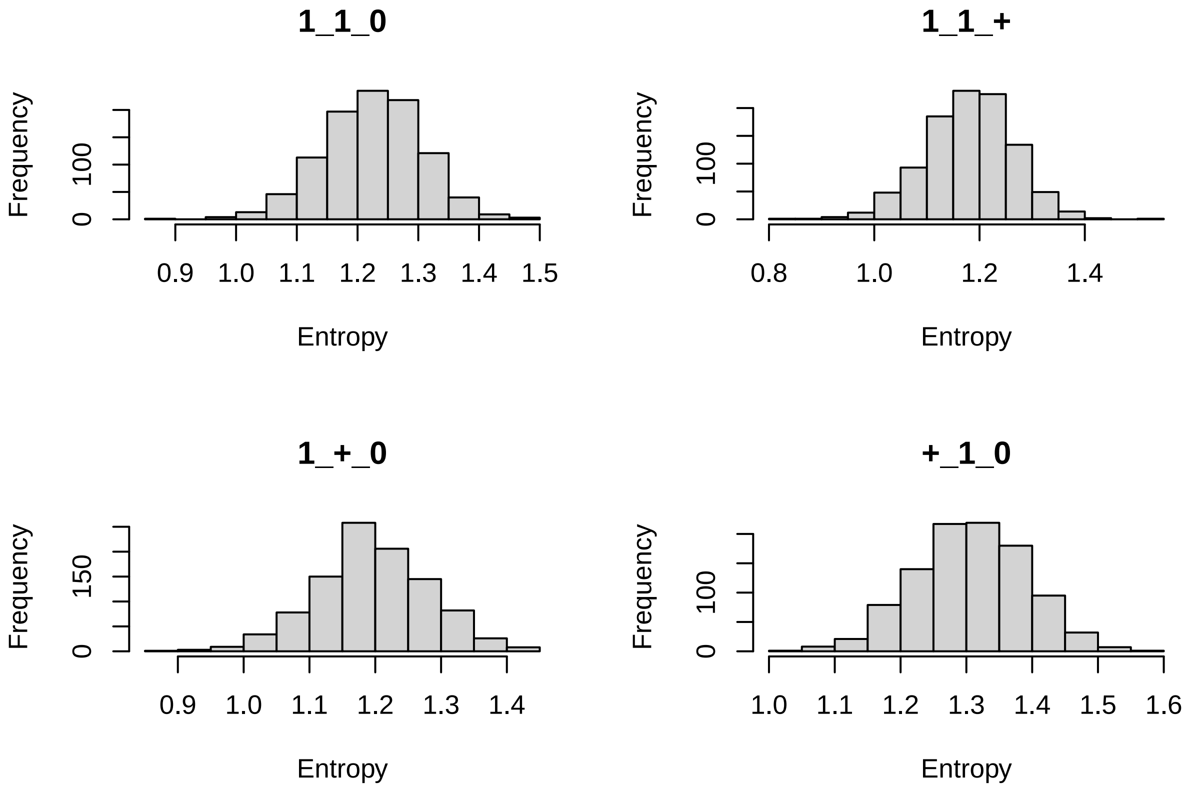
Looking at these graphs below, there is not much change happening. There is a clear shift to the left which means the entropy level goes down, and makes it more accurate. Since the graphs mostly overlap the origin, there is no interaction effect on the BMI distribution. Thus, missing one bi-variable has little effect on 0_1_1.


0_1_1 vs 0_1_+
Entropy of 0_1_1
Entropy of 0_1_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

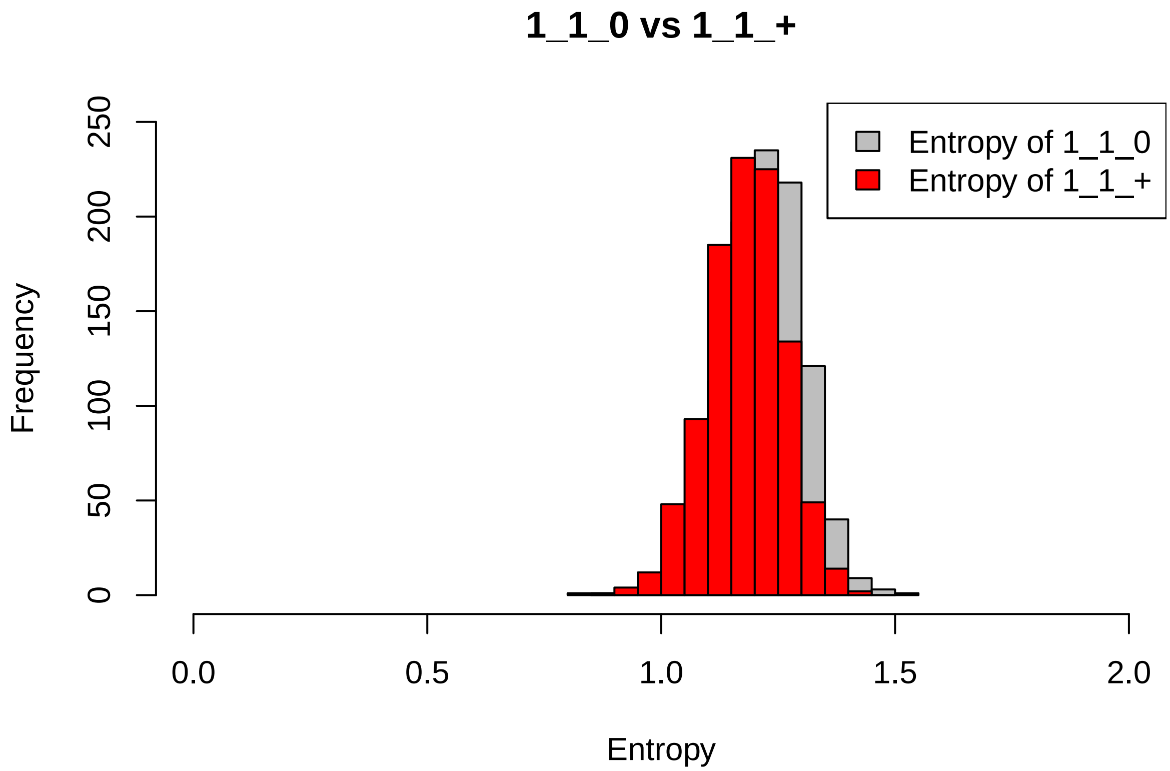

0_1_1 vs 0_+_1
Entropy of 0_1_1
Entropy of 0_+_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

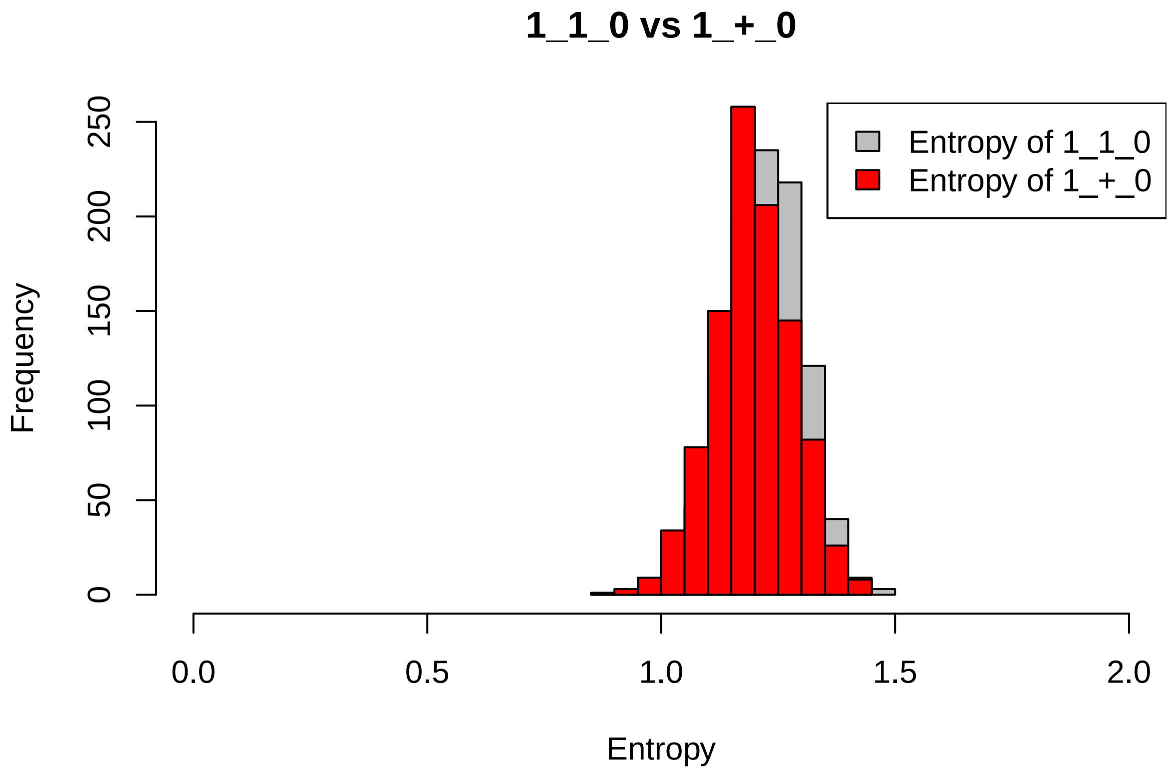

0_1_1 vs +_1_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 0_1_1
Entropy of +_1_1

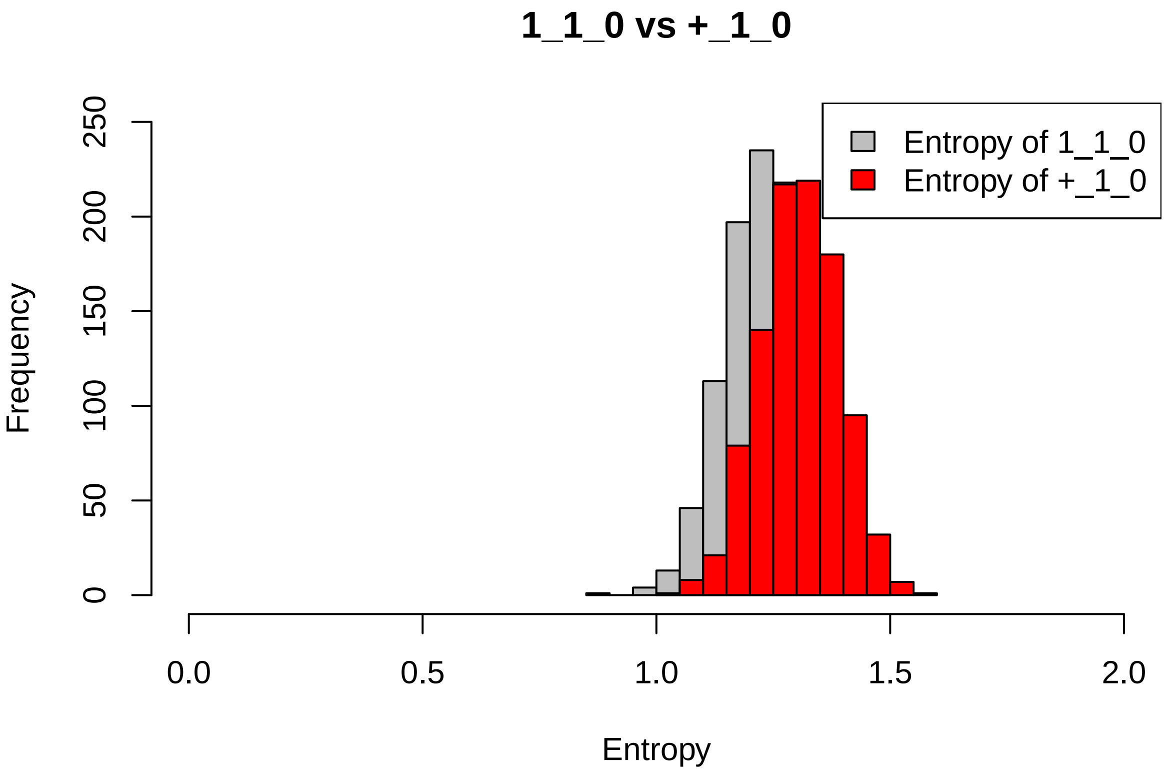
Yes HD __ Yes HighBP __ No HighChol



Looking at these graphs below, there is not much change happening when missing HighBP or HighChol. However, there is a clear shift to the right when missing HeartDisease, which means the entropy level goes up and makes it less accurate. Since the graphs mostly overlap the origin, there is no interaction effect on the BMI distribution. Thus, missing one bi-variable has little effect on 1_1_0.


1_1_0 vs 1_1_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 1_1_0
Entropy of 1_1_+


1_1_0 vs 1_+_0
Entropy of 1_1_0
Entropy of 1_+_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy


1_1_0 vs +_1_0
Entropy of 1_1_0
Entropy of +_1_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

5/ Investigating the 3-way interacting effects in GenHealth_5 sub-dataset:

Creating contigency tables based on the combination of these three selected binary variable

```
## Warning: package 'iNZightTools' was built under R version 4.2.2
```

```
##
## Attaching package: 'iNZightTools'
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

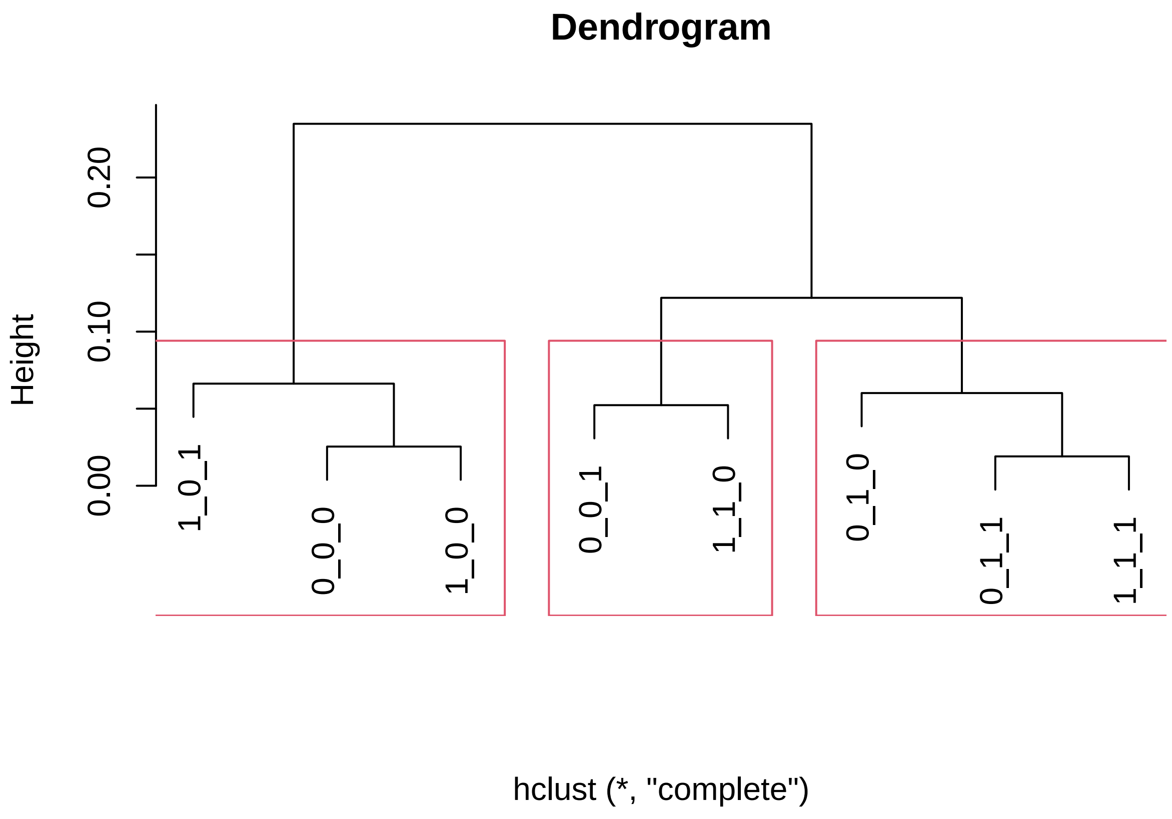
```
##                1         2         3          4          5           6
## 0_1_1 0.04492711 0.4064267 0.3448378 0.14340970 0.04284439 0.011603689
## 1_1_1 0.04443600 0.4189138 0.3532093 0.13444740 0.03949867 0.004937334
## 0_0_1 0.09059534 0.4900777 0.2873167 0.09663503 0.02847282 0.002588438
## 0_1_0 0.05521845 0.4308252 0.2985437 0.13592233 0.04793689 0.021844660
## 0_0_0 0.13067553 0.5614618 0.2104097 0.07198228 0.02159468 0.001107420
## 1_1_0 0.05464481 0.4904372 0.2745902 0.13114754 0.03005464 0.010928962
## 1_0_0 0.13477089 0.5471698 0.2102426 0.08894879 0.01078167 0.005390836
## 1_0_1 0.10512129 0.5822102 0.2371968 0.05121294 0.02156334 0.002695418
## Total 0.06862015 0.4584885 0.3030378 0.12002318 0.03584140 0.008608559
##                 7            8            9           10
## 0_1_1 0.002380244 0.0011901220 0.0014876525 0.0008925915
## 1_1_1 0.001898975 0.0011393847 0.0011393847 0.0003797949
## 0_0_1 0.001725626 0.0017256255 0.0008628128 0.0000000000
## 0_1_0 0.004247573 0.0024271845 0.0030339806 0.0000000000
## 0_0_0 0.001661130 0.0005537099 0.0005537099 0.0000000000
## 1_1_0 0.005464481 0.0013661202 0.0000000000 0.0013661202
## 1_0_0 0.002695418 0.0000000000 0.0000000000 0.0000000000
## 1_0_1 0.000000000 0.0000000000 0.0000000000 0.0000000000
## Total 0.002483238 0.0012416191 0.0012416191 0.0004138730
```

```
##                1         2         3          4          5           6
## 0_1   0.04831304 0.4144540 0.3296067 0.14094630 0.04451986 0.014973049
## 1_1   0.04665676 0.4344725 0.3361070 0.13372957 0.03744428 0.006240713
## 0_0   0.11500843 0.5335582 0.2404722 0.08161889 0.02428331 0.001686341
## 1_0   0.11994609 0.5646900 0.2237197 0.07008086 0.01617251 0.004043127
## Total 0.06862015 0.4584885 0.3030378 0.12002318 0.03584140 0.008608559
##                 7           8            9           10
## 0_1   0.002994610 0.001597125 0.0019964065 0.0005989219
## 1_1   0.002674591 0.001188707 0.0008915305 0.0005943536
## 0_0   0.001686341 0.001011804 0.0006745363 0.0000000000
## 1_0   0.001347709 0.000000000 0.0000000000 0.0000000000
## Total 0.002483238 0.001241619 0.0012416191 0.0004138730
```

```
##                1         2         3         4          5           6
## 0_1   0.05663717 0.4278761 0.3300885 0.1314159 0.03915929 0.009292035
## 1_1   0.05193076 0.4390812 0.3388815 0.1241678 0.03728362 0.004660453
## 0_0   0.09467284 0.4991314 0.2524609 0.1024899 0.03416329 0.011001737
## 1_0   0.08159565 0.5095195 0.2529465 0.1169538 0.02357208 0.009066183
## Total 0.06862015 0.4584885 0.3030378 0.1200232 0.03584140 0.008608559
##                 7            8            9           10
## 0_1   0.002212389 0.0013274336 0.0013274336 0.0006637168
## 1_1   0.001664447 0.0009986684 0.0009986684 0.0003328895
## 0_0   0.002895194 0.0014475970 0.0017371164 0.0000000000
## 1_0   0.004533092 0.0009066183 0.0000000000 0.0009066183
## Total 0.002483238 0.0012416191 0.0012416191 0.0004138730
```

```
##                1         2         3          4          5           6
## 1_1   0.04471138 0.4119119 0.3485152 0.13947281 0.04137471 0.008675342
## 0_1   0.09411765 0.5124183 0.2751634 0.08562092 0.02679739 0.002614379
## 1_0   0.05504202 0.4491597 0.2911765 0.13445378 0.04243697 0.018487395
## 0_0   0.13137345 0.5590262 0.2103813 0.07487368 0.01975195 0.001837391
## Total 0.06862015 0.4584885 0.3030378 0.12002318 0.03584140 0.008608559
##                 7            8            9           10
## 1_1   0.002168836 0.0011678345 0.0013346680 0.0006673340
## 0_1   0.001307190 0.0013071895 0.0006535948 0.0000000000
## 1_0   0.004621849 0.0021008403 0.0021008403 0.0004201681
## 0_0   0.001837391 0.0004593477 0.0004593477 0.0000000000
## Total 0.002483238 0.0012416191 0.0012416191 0.0004138730
```

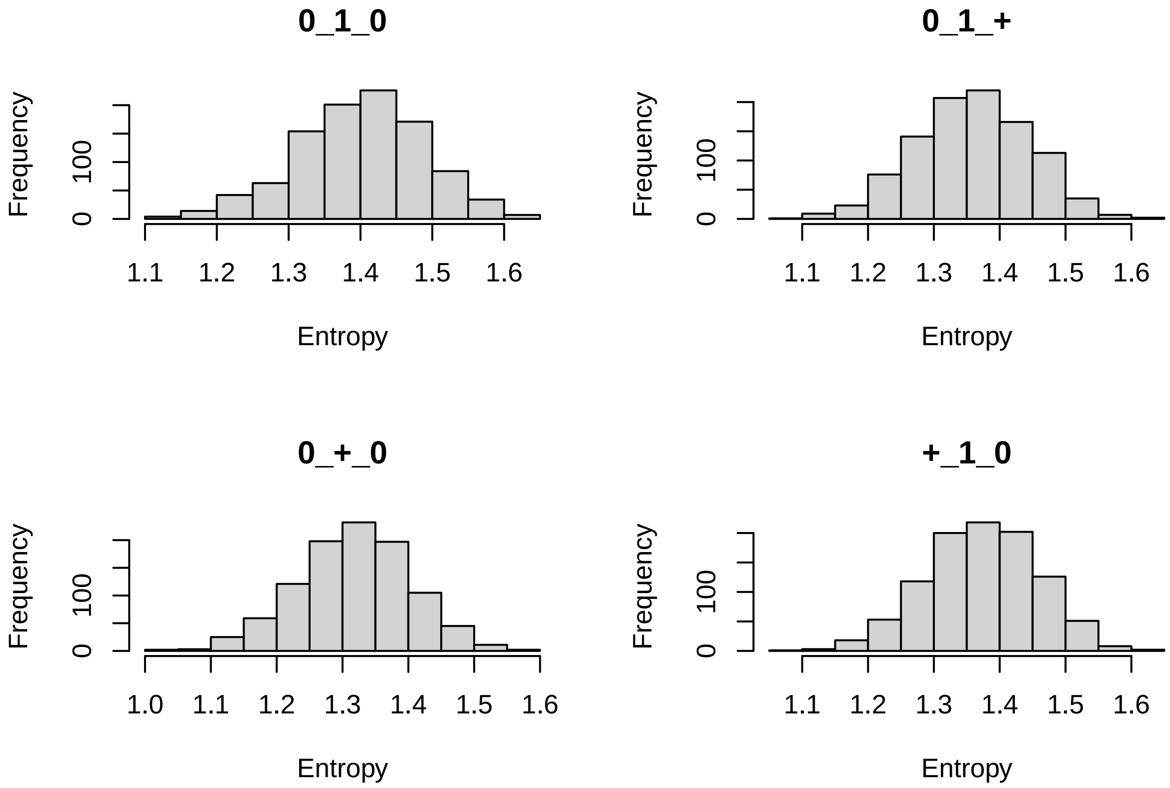
3-way HC-tree of GenHealth_5:



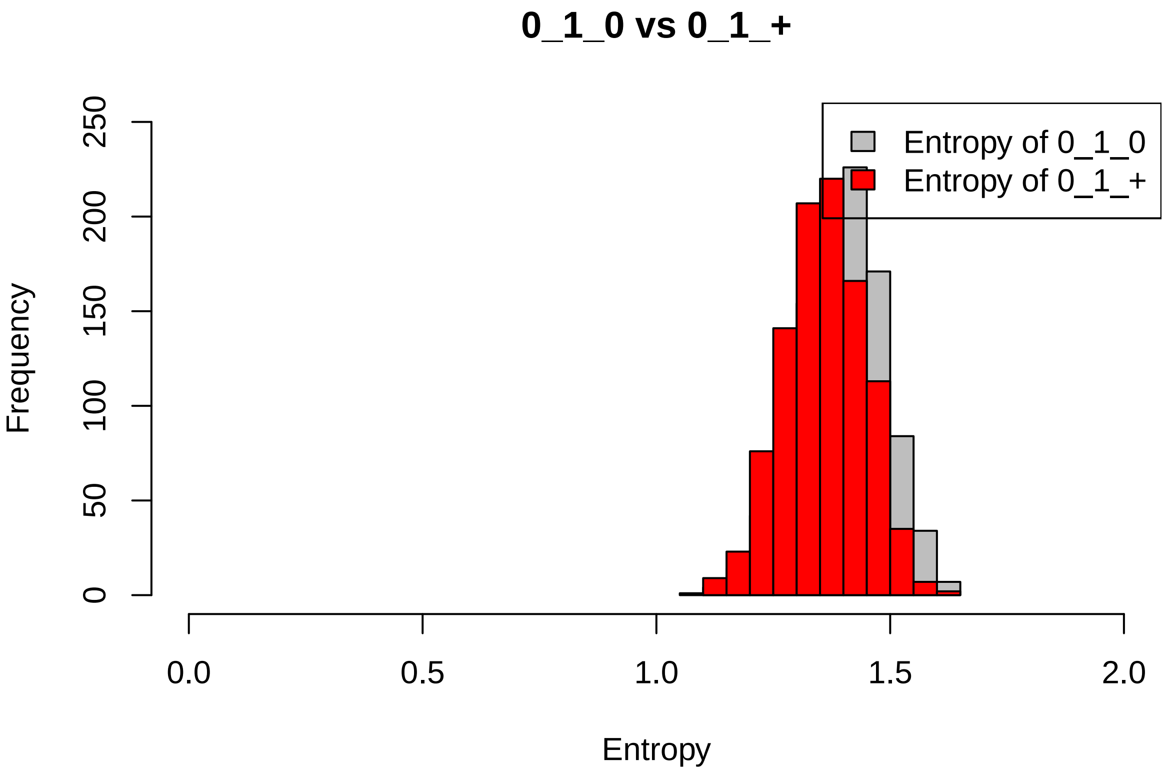
Comparing the entropies of categorical variables to see the effects

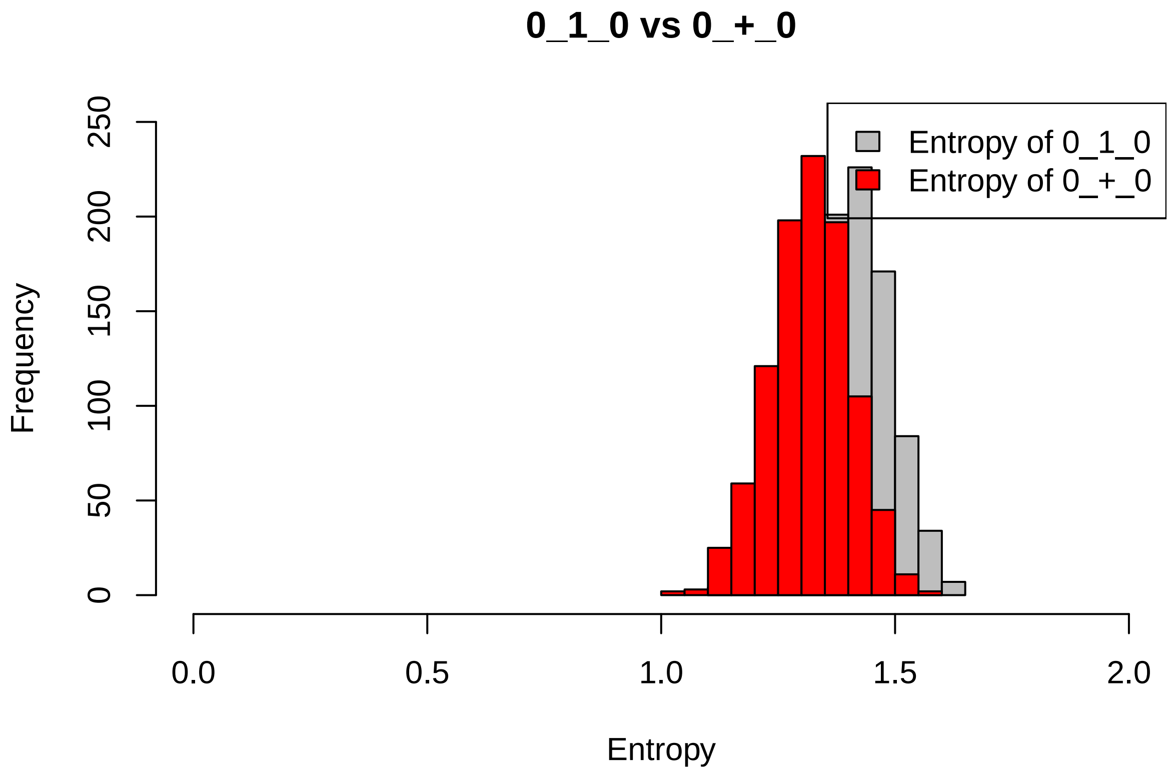
No HD __ Yes HighBP __ No HighChol

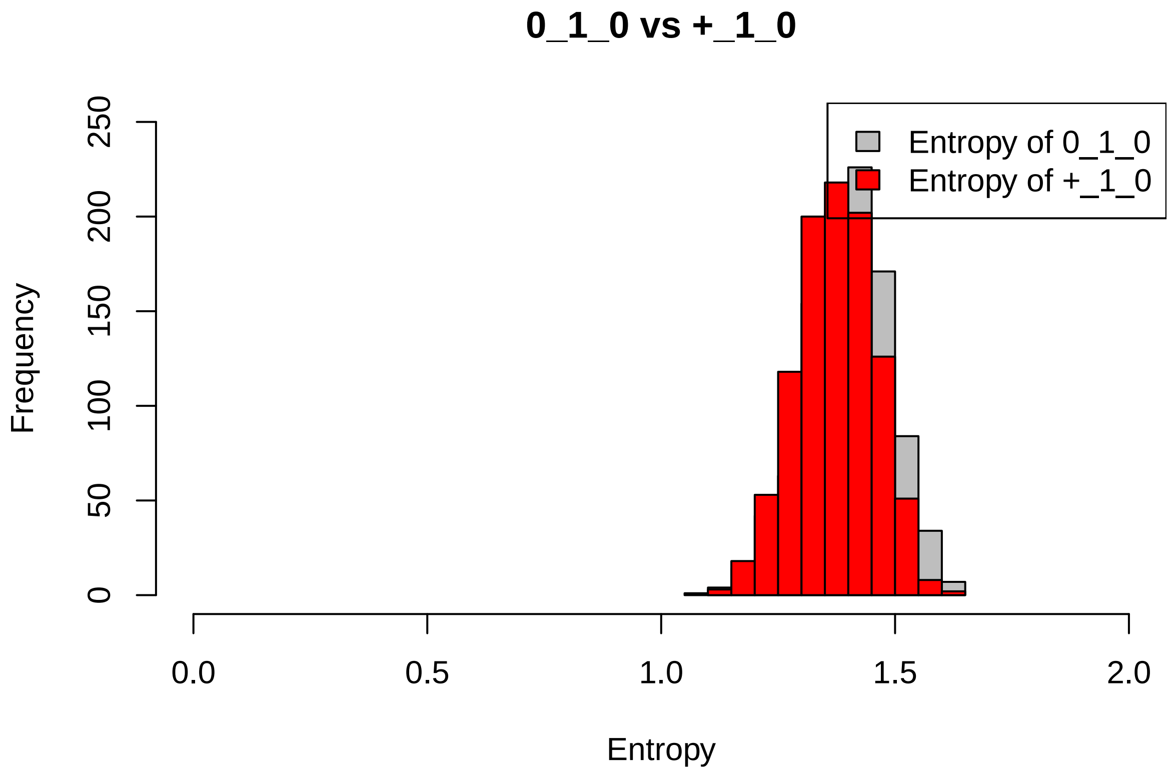
```
## Warning: package 'DescTools' was built under R version 4.2.2
```



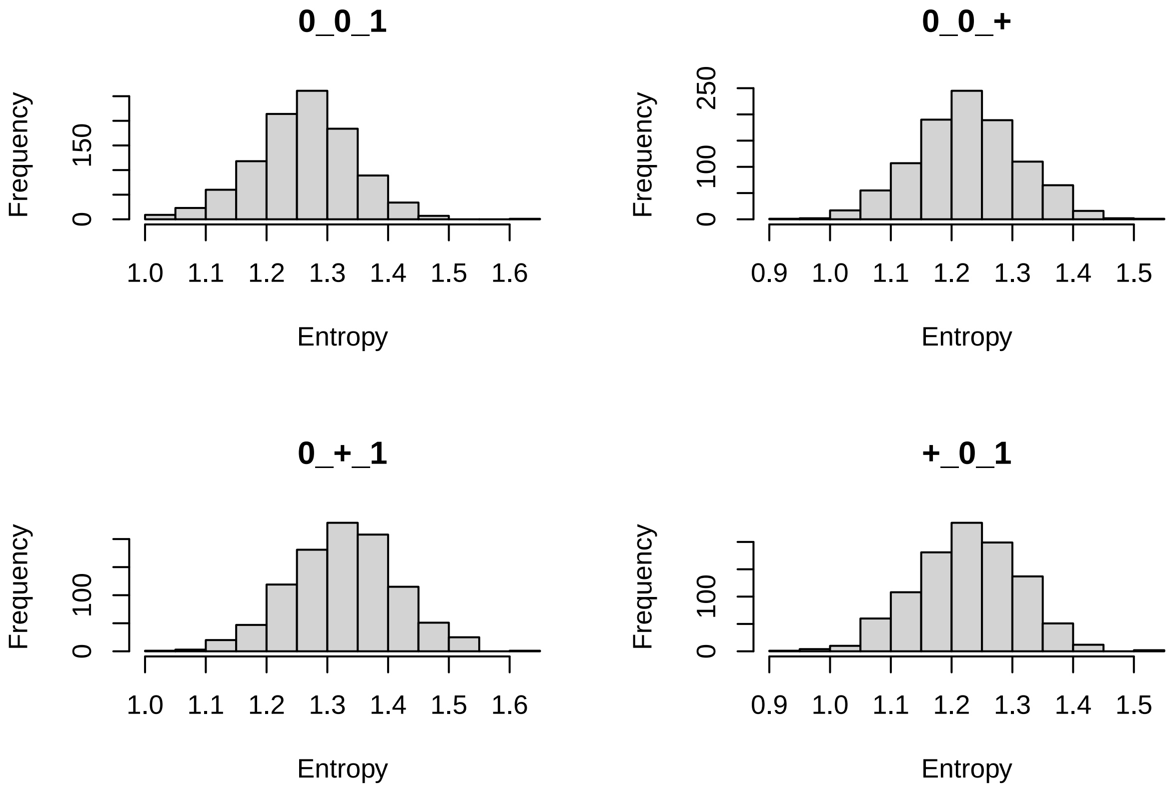
Looking at these graphs below, it seems that there isn't much shift happening. There is a clear shift to the left when missing HighBP, which means that the entropy level goes down, and makes it more accurate. Thus, missing 1 bi-variable has little effect on 0_1_0. Since the graphs mostly overlap the origin, there is no interaction effect on the BMI distribution.


0_1_0 vs 0_1_+
Entropy of 0_1_0
Entropy of 0_1_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

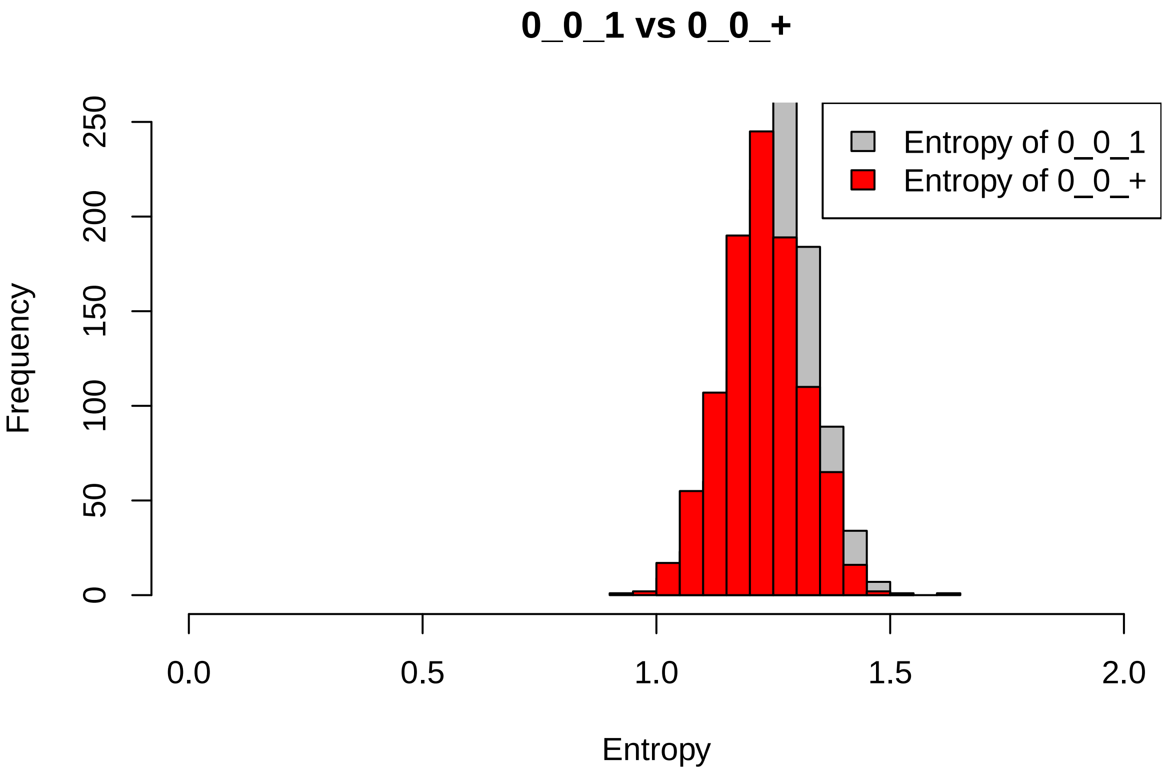

0_1_0 vs 0_+_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 0_1_0
Entropy of 0_+_0

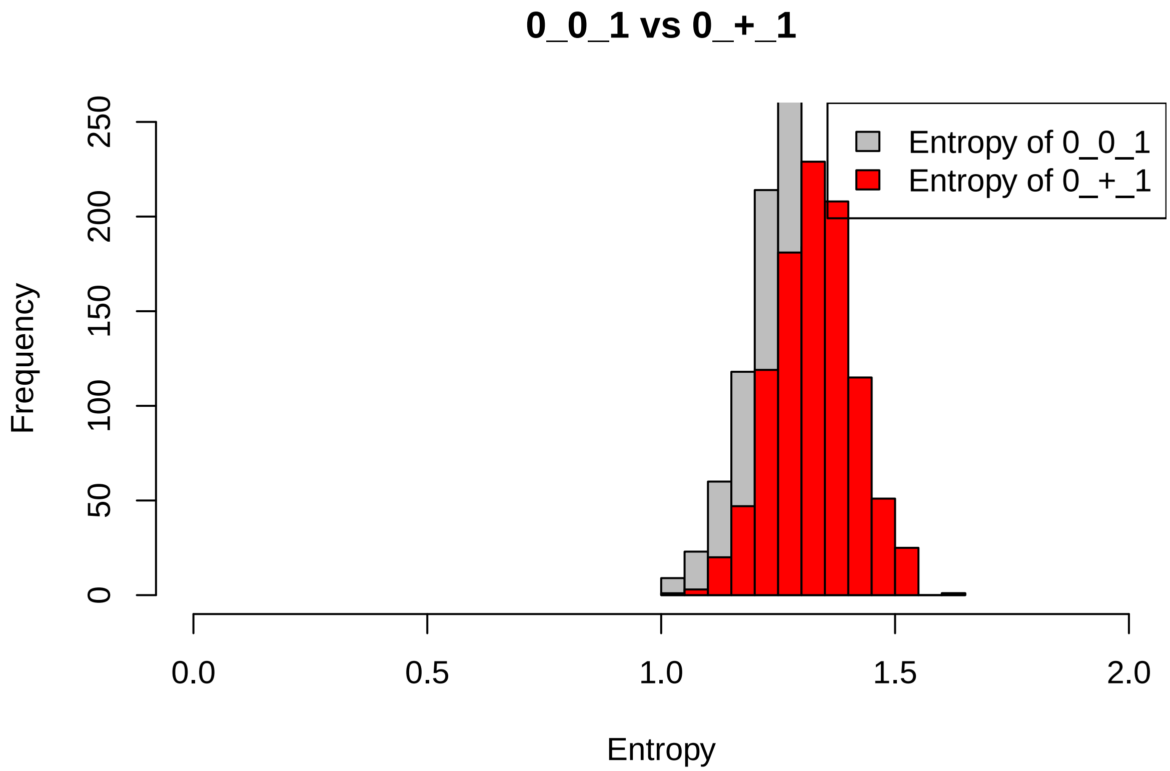

0_1_0 vs +_1_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 0_1_0
Entropy of +_1_0

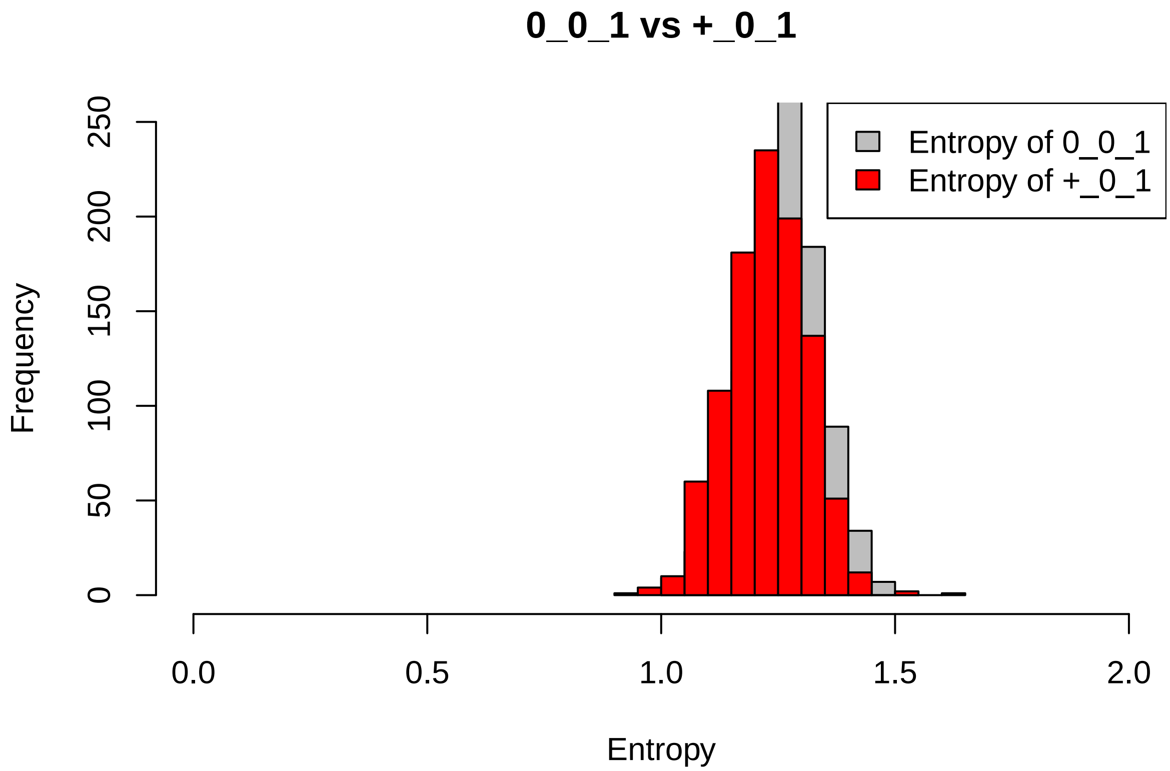
No HD __ No HighBP __ Yes HighChol



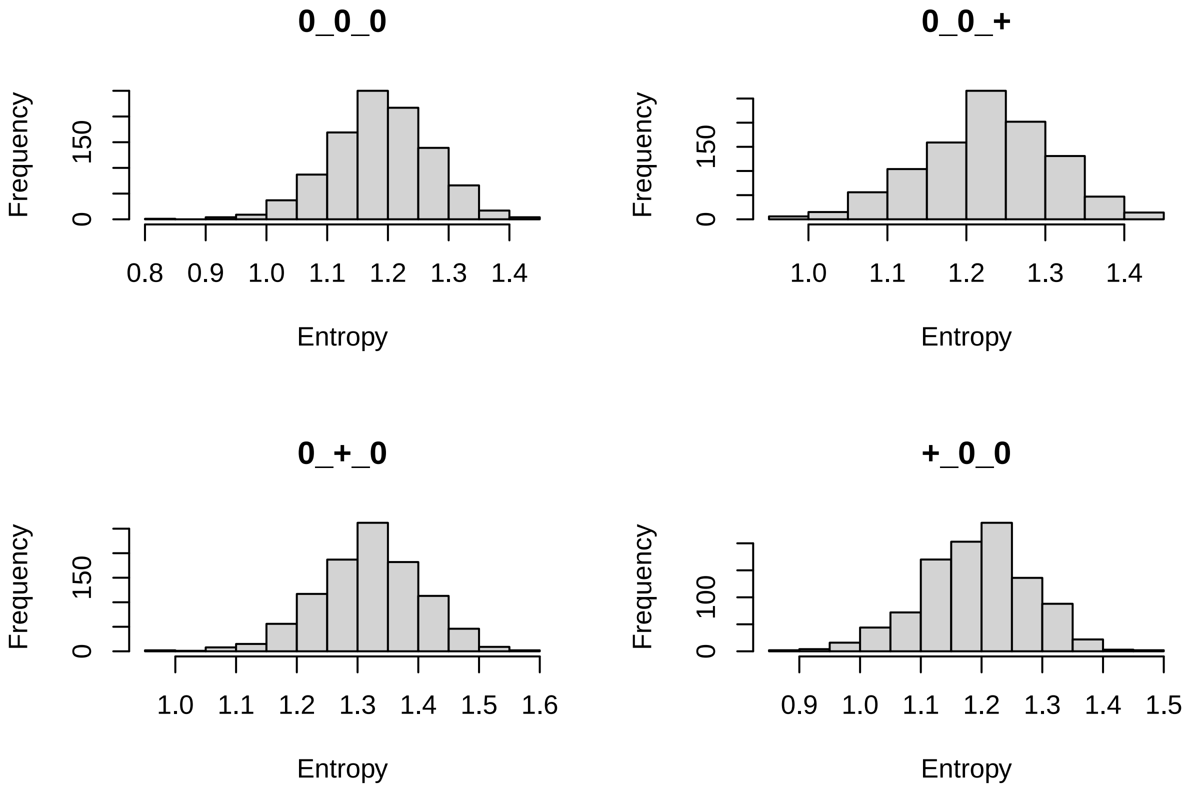
Looking at these graphs below, there is clearly a shift to the right when HighBP is missing, which means the entropy level goes up and makes it less accurate. In addition, there is a small shift to the left when missing HeartDisease or HighChol. However, missing HD doesn't change much. Since the 3 graphs do not overlap the origin, there is an interaction effect on the BMI distribution.


0_0_1 vs 0_0_+
Frequency
0
50
100
150
200
250
Entropy
0.0
0.5
1.0
1.5
2.0
Entropy of 0_0_1
Entropy of 0_0_+

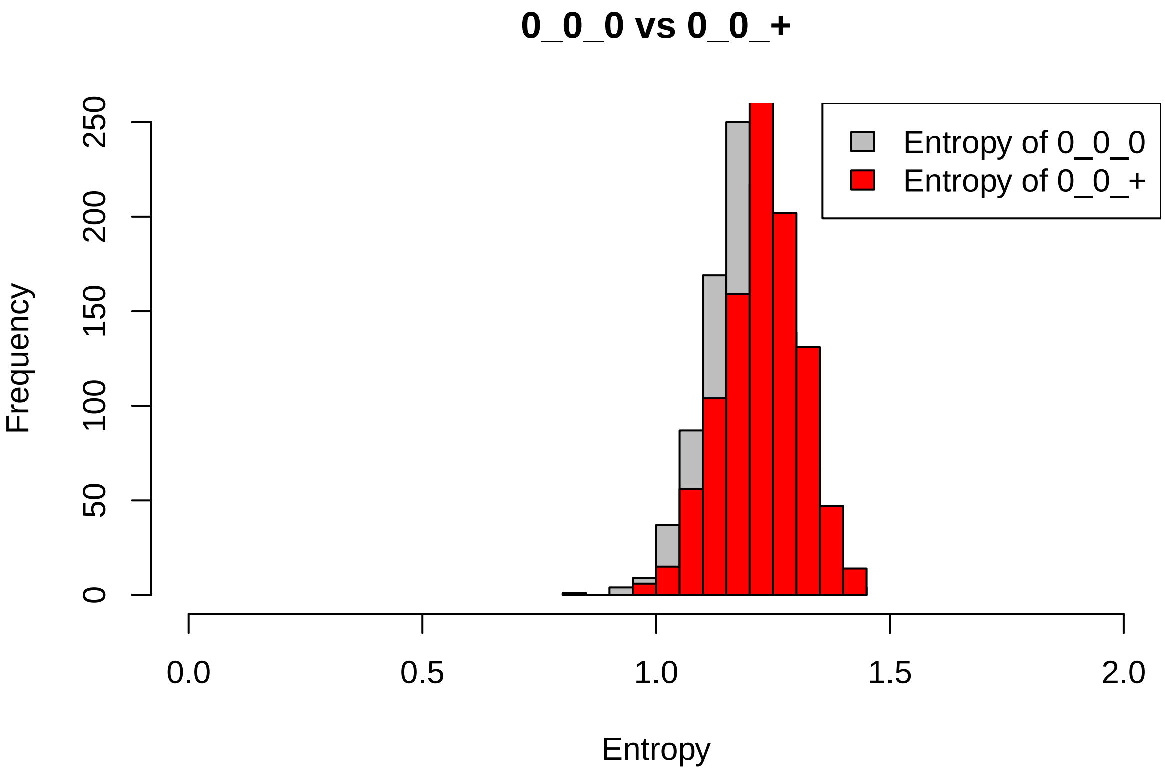

0_0_1 vs 0_+_1
Entropy of 0_0_1
Entropy of 0_+_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

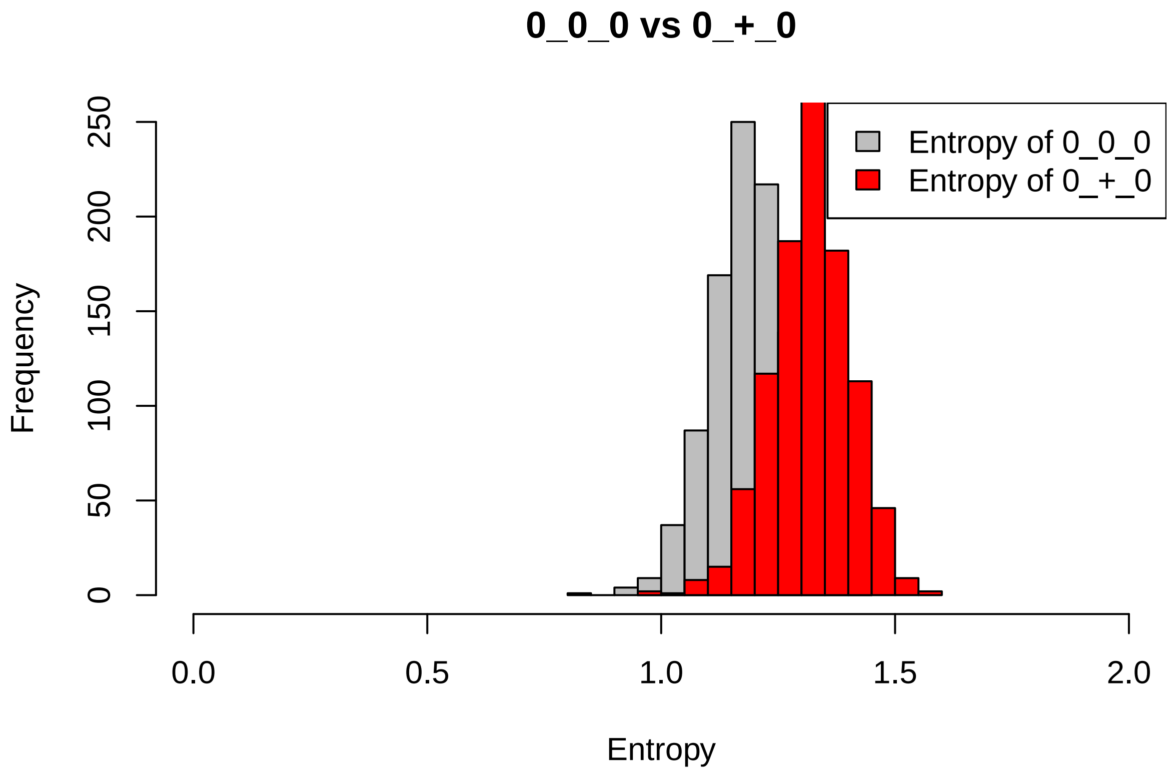

0_0_1 vs +_0_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 0_0_1
Entropy of +_0_1

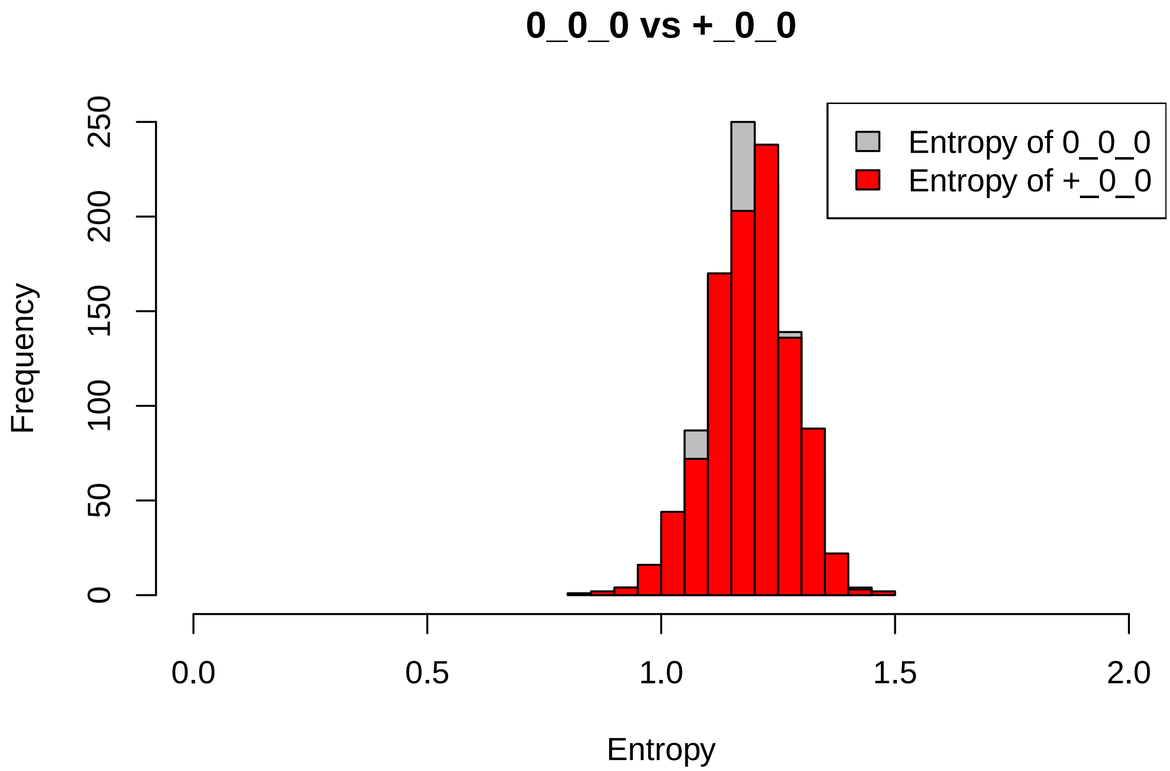
No HD __ No HighBP __ No HighChol



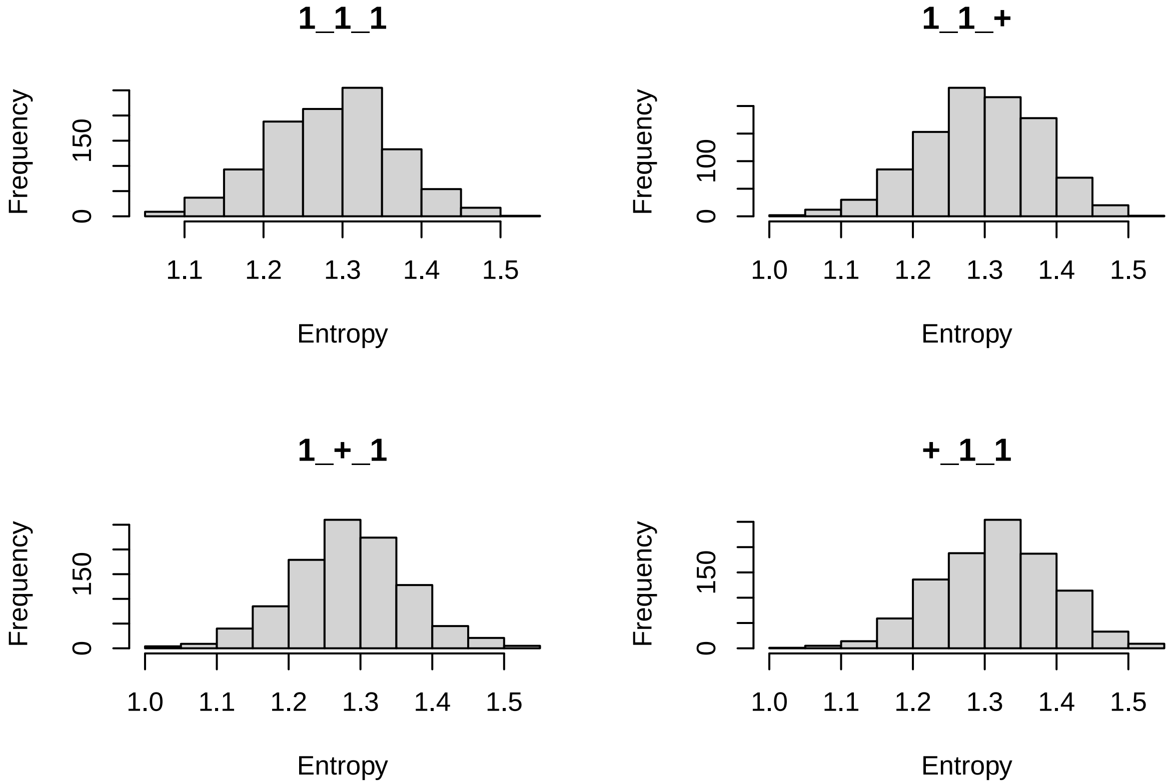
Looking at these graphs below, there is not much change happening except the graph of missing HighBP. There is a clear shift to the right when missing HighBP, which means that the entropy level goes up, and makes it less accurate. Since the graphs mostly overlap to the origin, there is no interaction effect on the BMI distribution.


0_0_0 vs 0_0_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 0_0_0
Entropy of 0_0_+

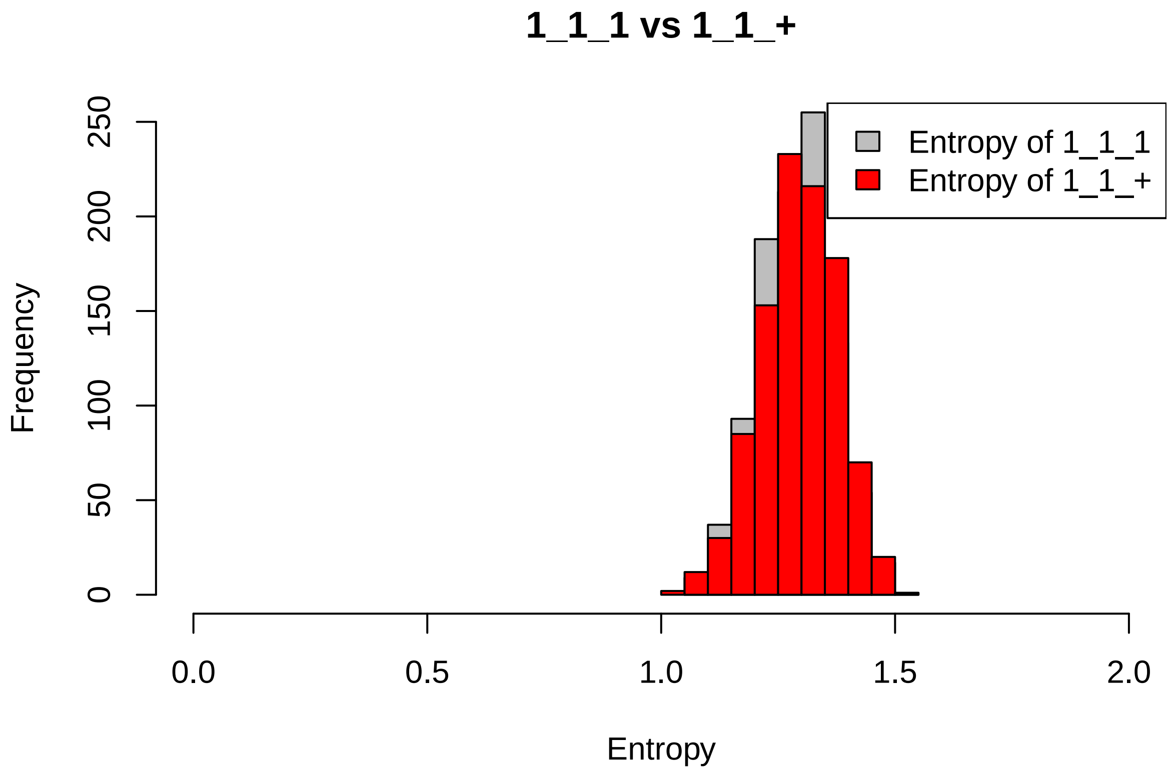

0_0_0 vs 0_+_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 0_0_0
Entropy of 0_+_0

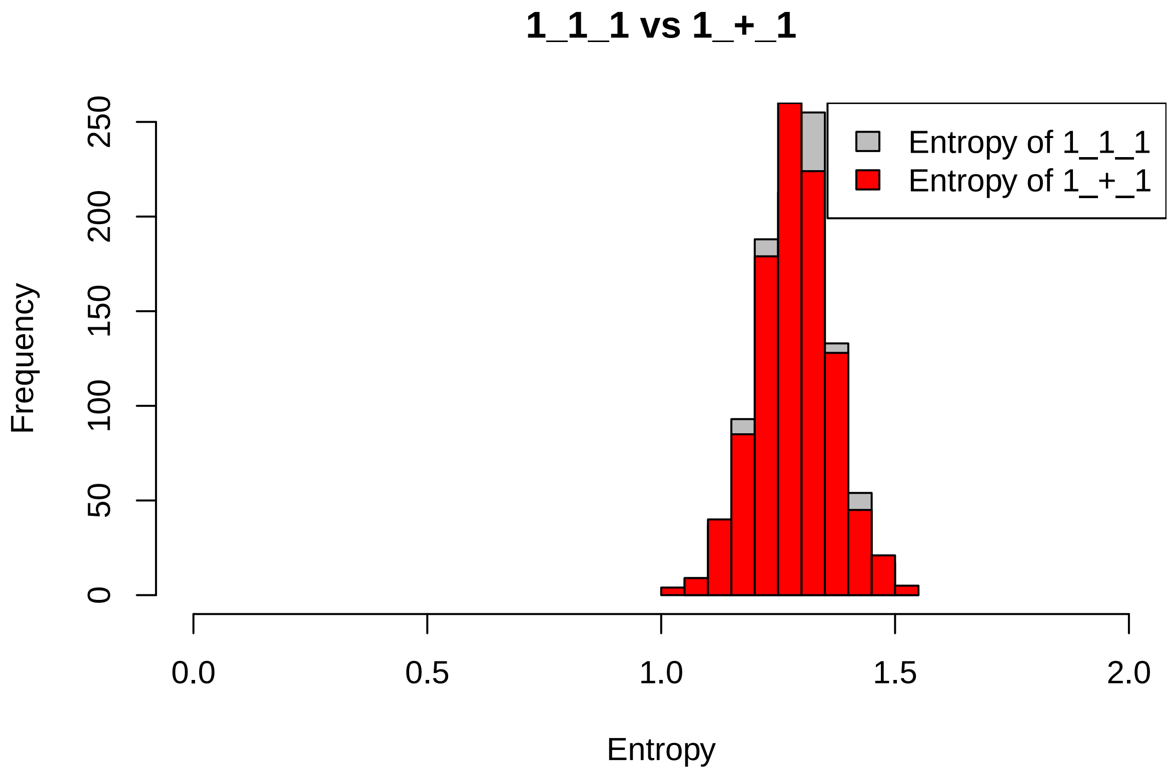

0_0_0 vs +_0_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 0_0_0
Entropy of +_0_0

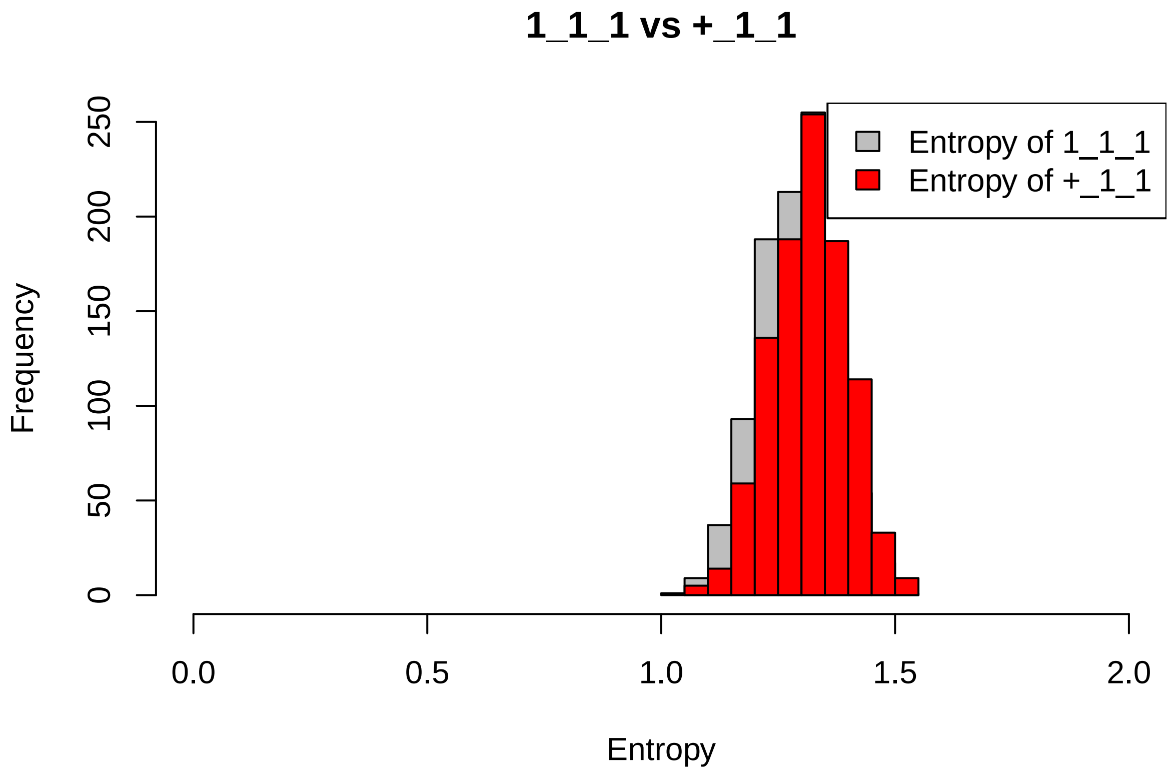
Yes HD __ Yes HighBP __ Yes HighChol



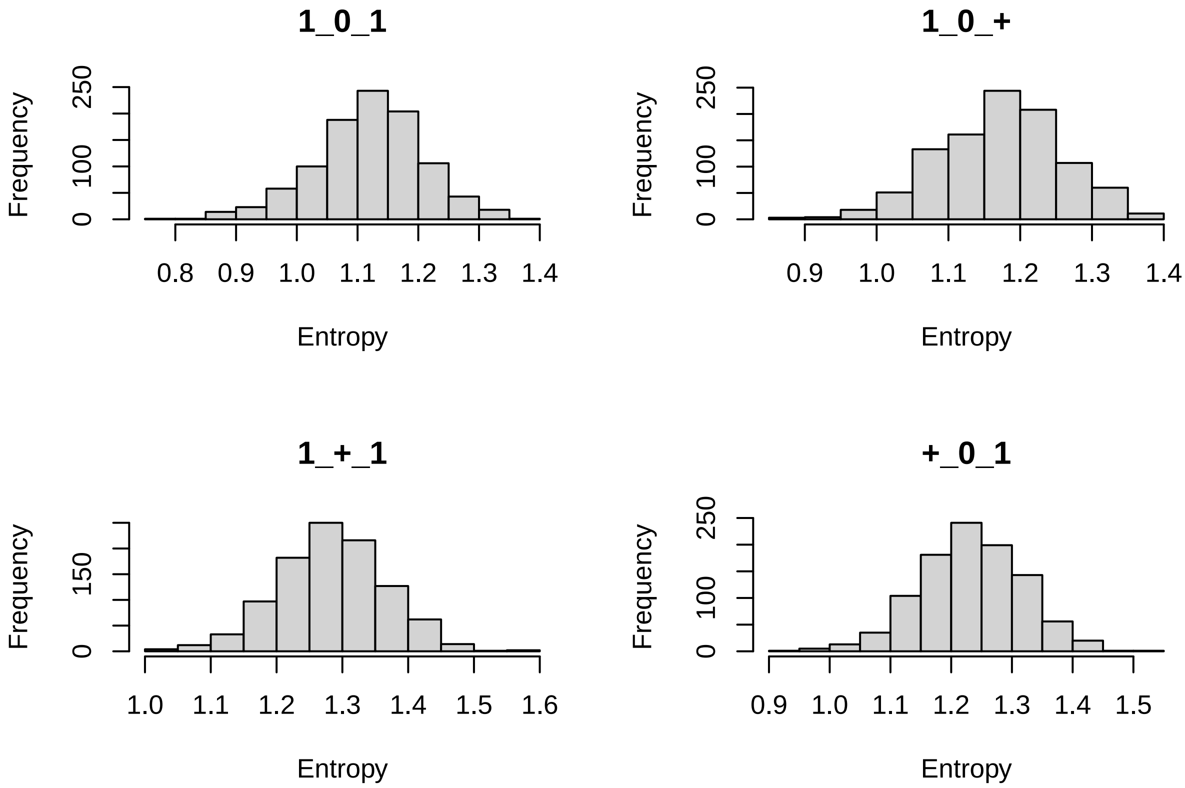
Looking at these graphs below, there is almost no change happening. Thus, there is little effect on 1_1_1. Since the 3 graphs of overlap the origin, there is no interaction effect on the BMI distribution.


1_1_1 vs 1_1_+
Entropy of 1_1_1
Entropy of 1_1_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

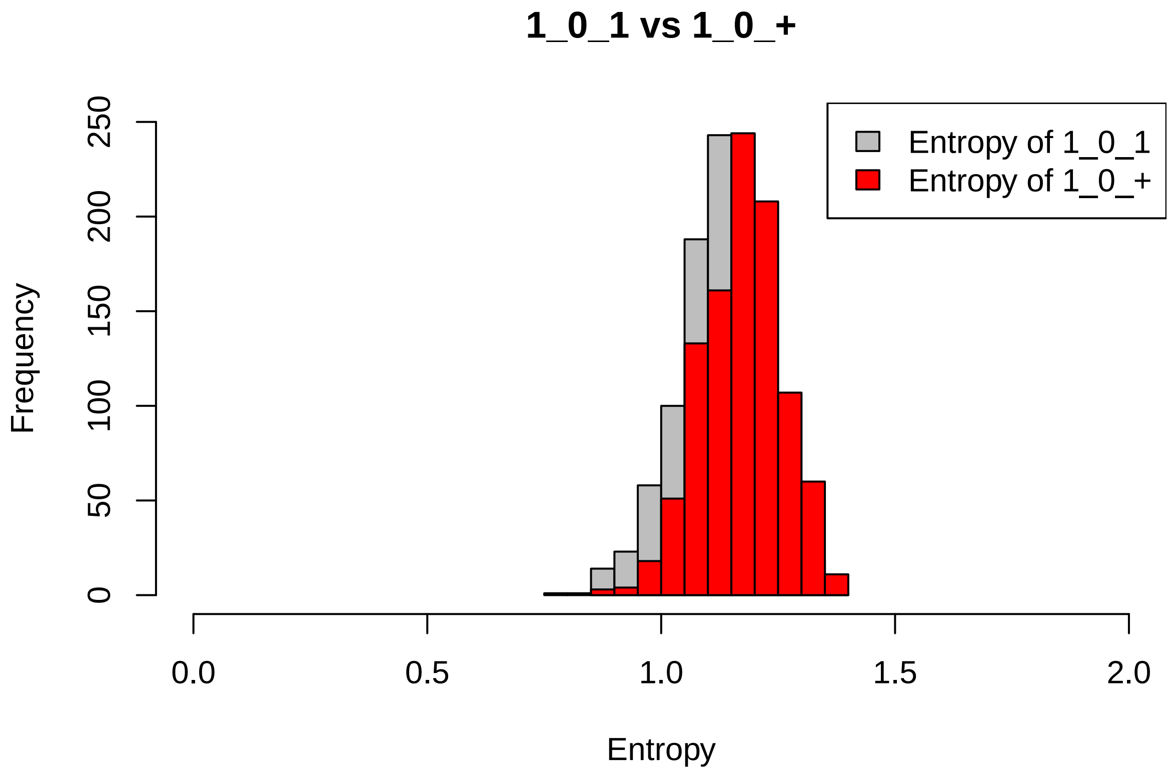

1_1_1 vs 1_+_1
Frequency
0
50
100
150
200
250
Entropy
0.0
0.5
1.0
1.5
2.0
Entropy of 1_1_1
Entropy of 1_+_1

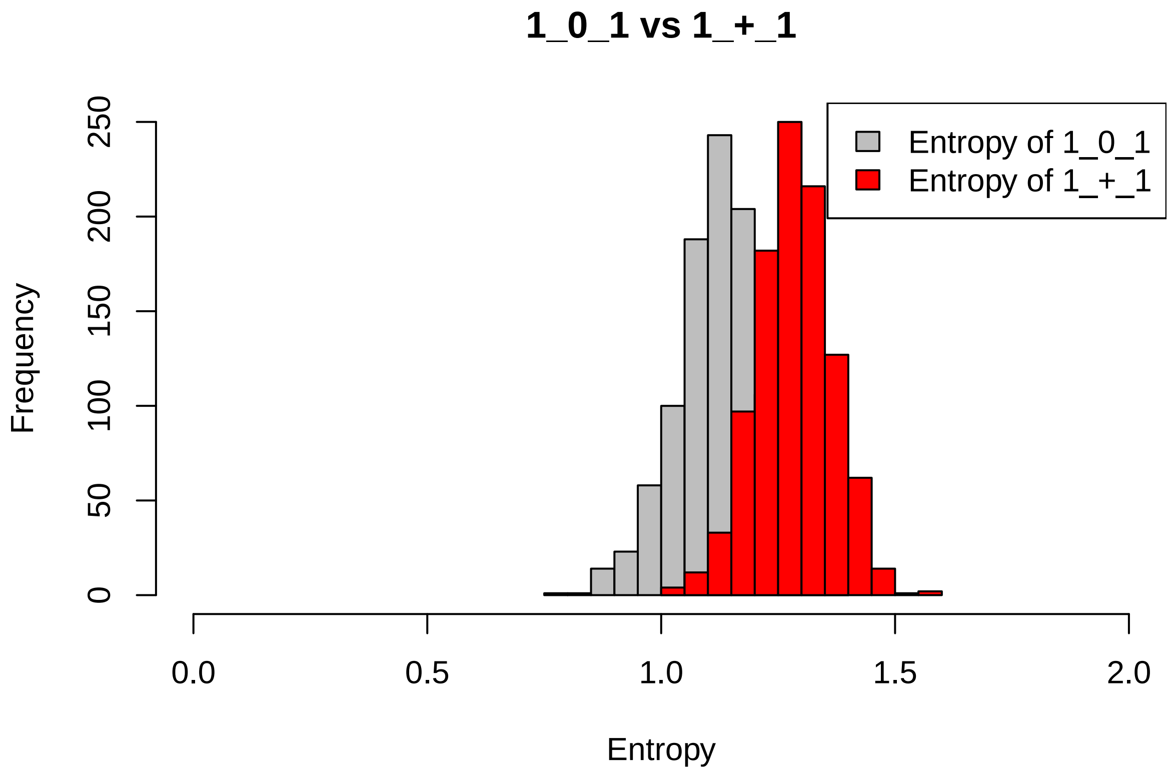

1_1_1 vs +_1_1
Entropy of 1_1_1
Entropy of +_1_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

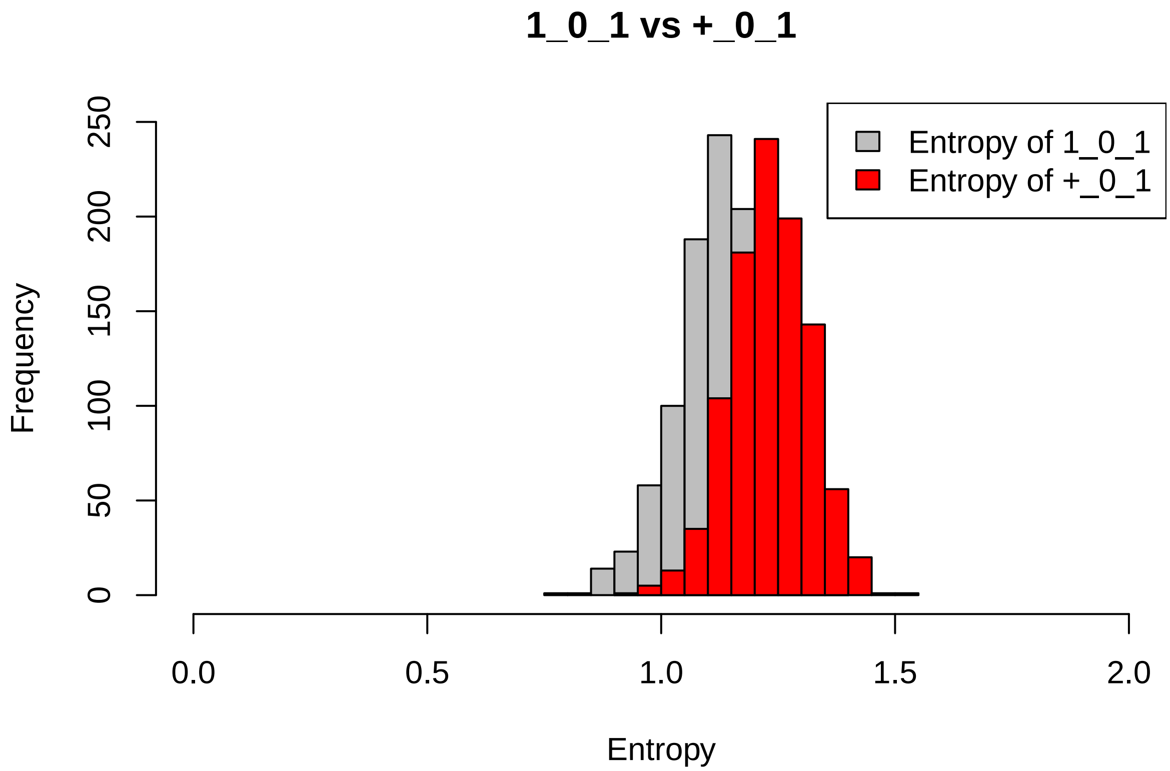
Yes HD __ No HighBP __ Yes HighChol



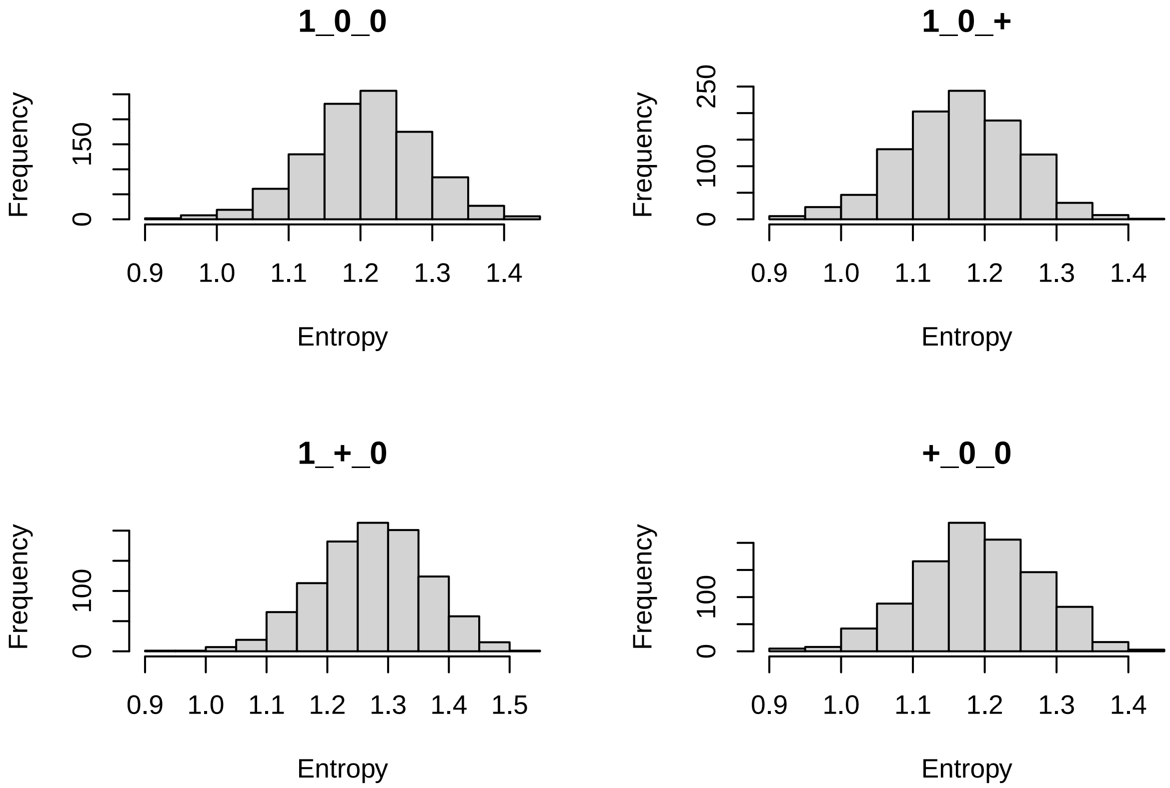
Looking at these graphs below, there is clearly a strong shift to the right when HighBP or HeartDisease or HighChol is missing, which means the entropy level goes up and makes it less accurate. Since all 3 graphs of missing HighBP or missing HeartDisease or missing HighChol don't overlap the origin, there is an interaction effect on the BMI distribution.


1_0_1 vs 1_0_+
Entropy of 1_0_1
Entropy of 1_0_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

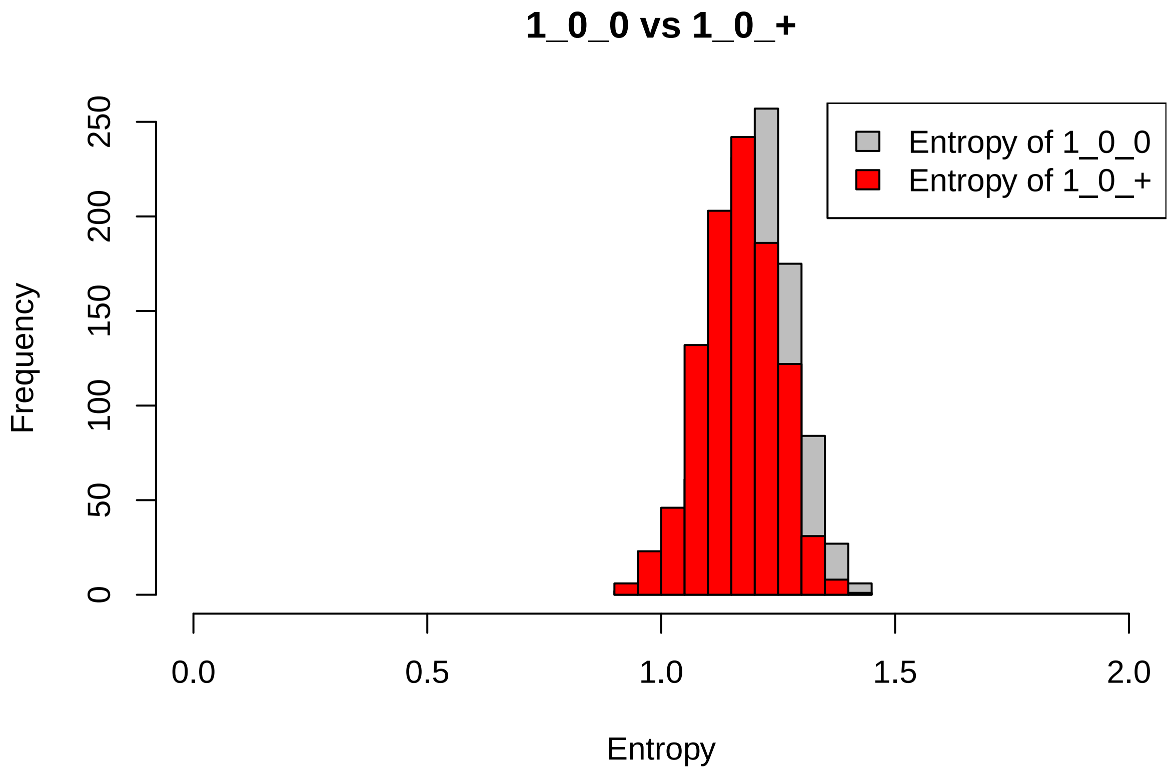

1_0_1 vs 1_+_1
Entropy of 1_0_1
Entropy of 1_+_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

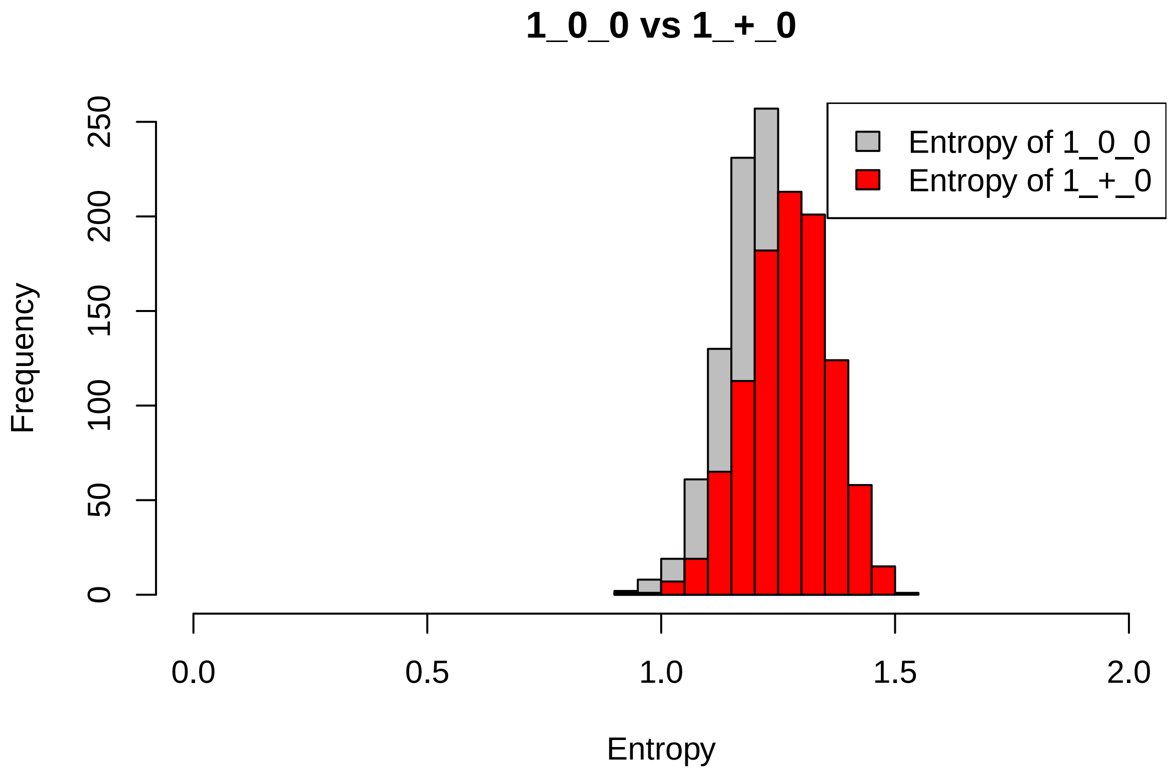

1_0_1 vs +_0_1
Entropy of 1_0_1
Entropy of +_0_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

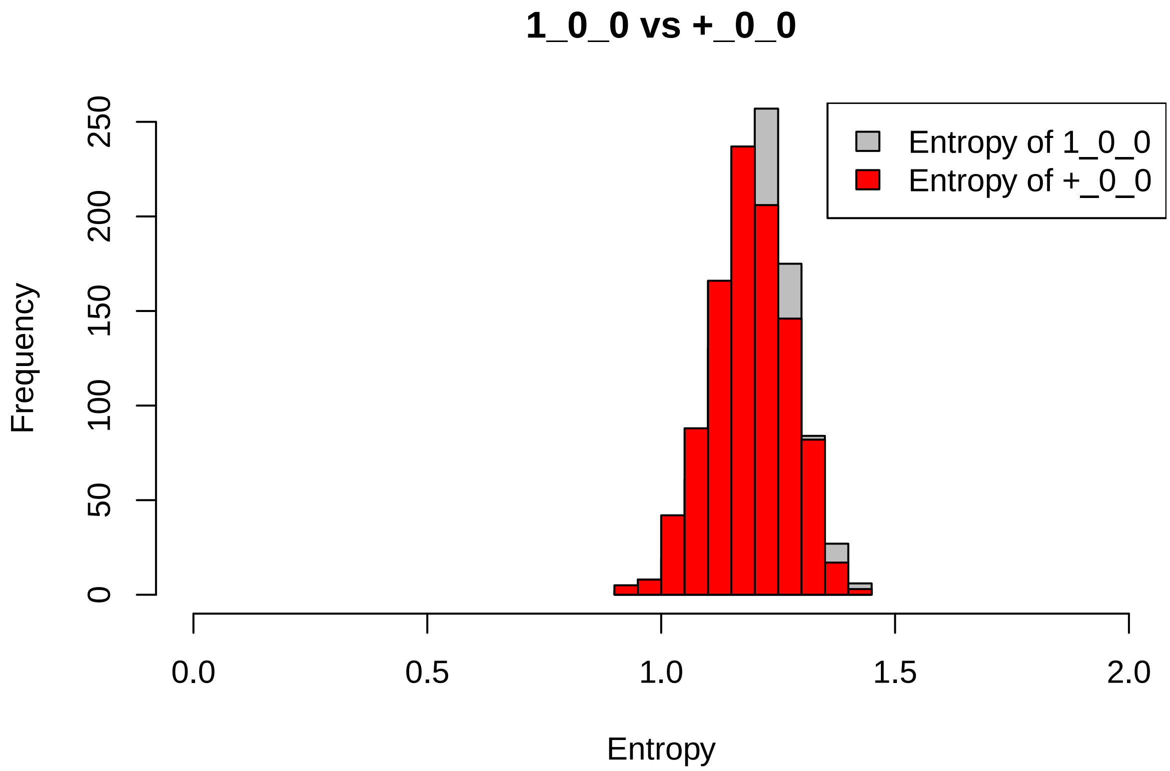
Yes HD __ No HighBP __ No HighChol



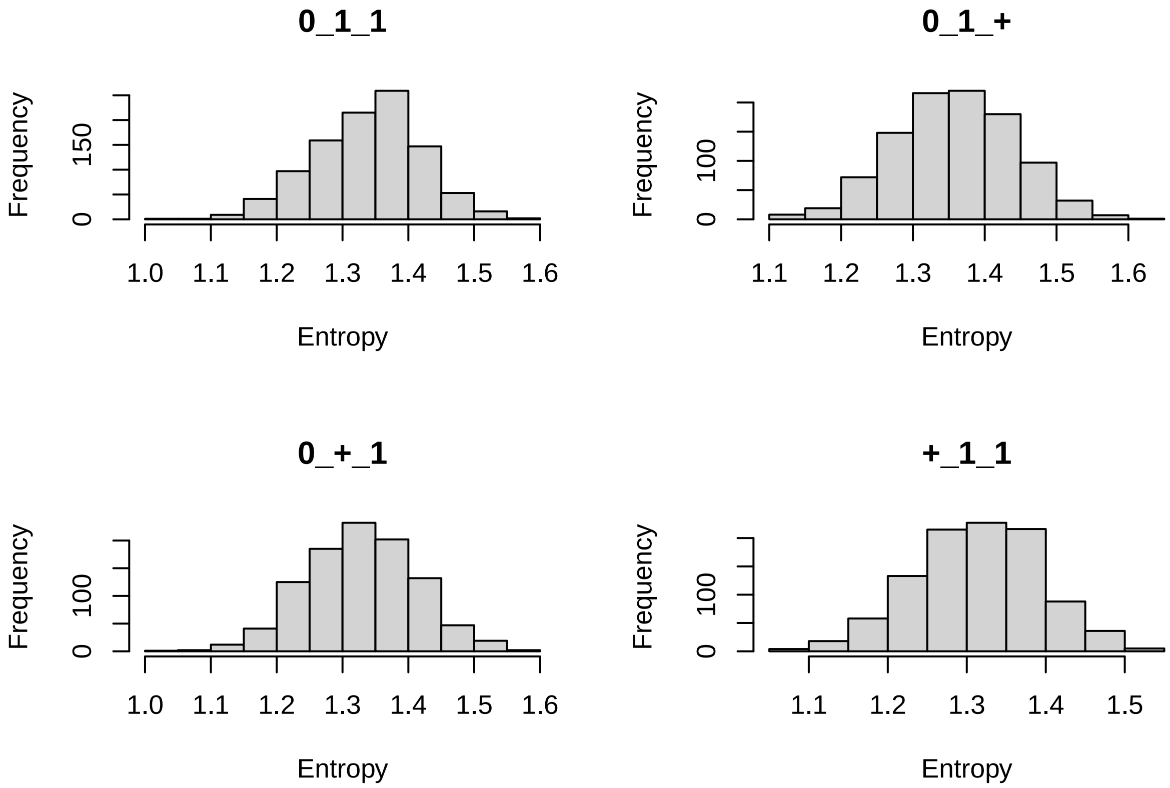
Looking at these graphs below, there is not much change happening. There is a clear shift to the right when missing HighBP, which means the entropy level goes up, makes it less accurate. Since the graphs mostly overlap the origin, there is no interaction effect on the BMI distribution. Thus, having missing 1 bi-variable have little effect on 1_0_0.


1_0_0 vs 1_0_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 1_0_0
Entropy of 1_0_+

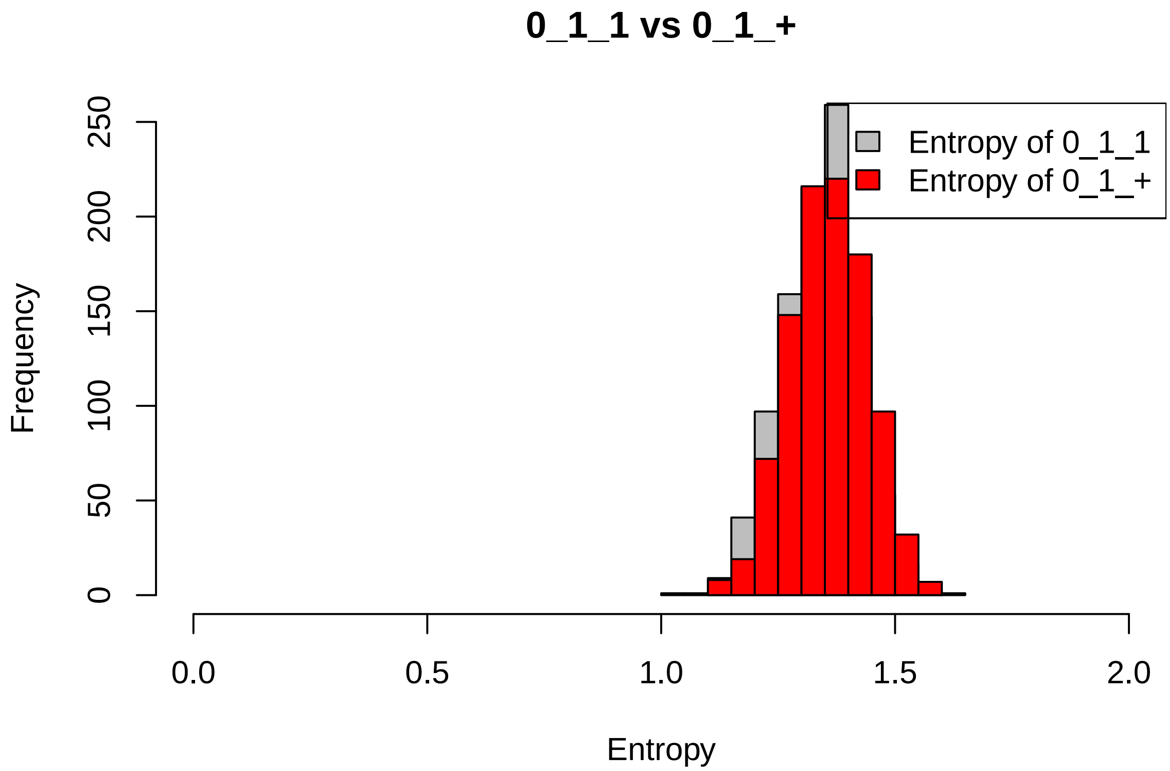

1_0_0 vs 1_+_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 1_0_0
Entropy of 1_+_0

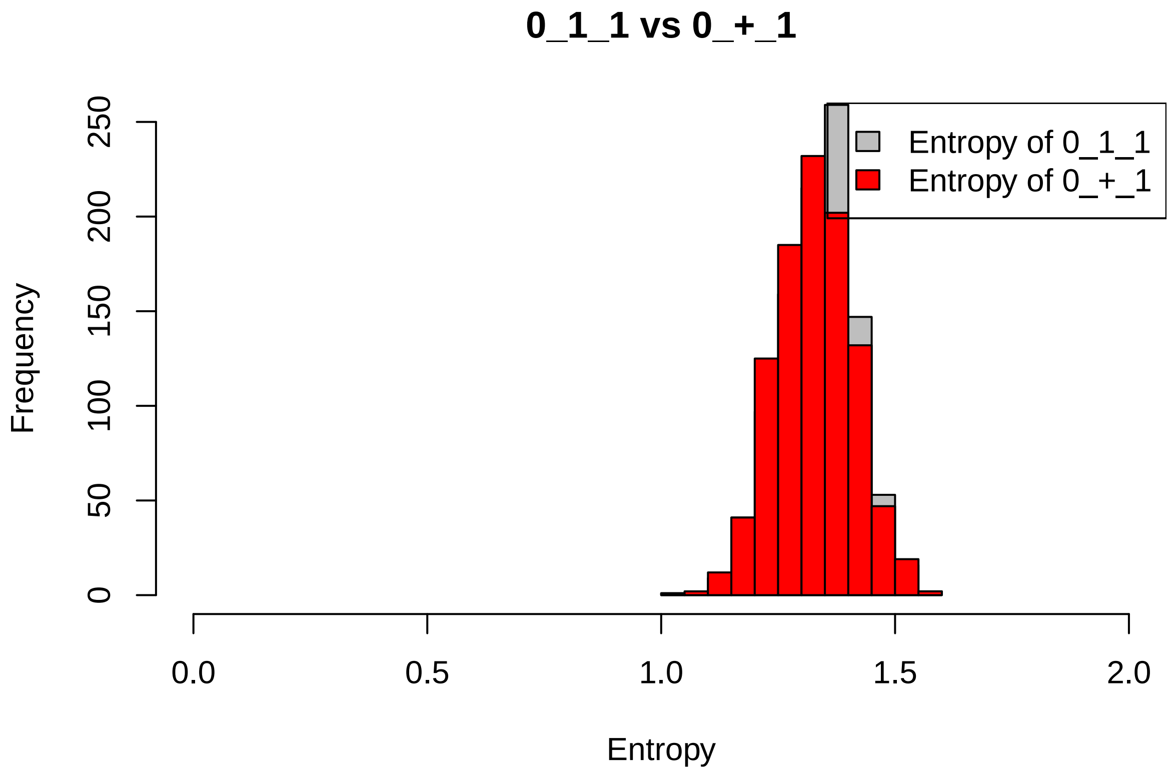

1_0_0 vs +_0_0
Entropy of 1_0_0
Entropy of +_0_0
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy

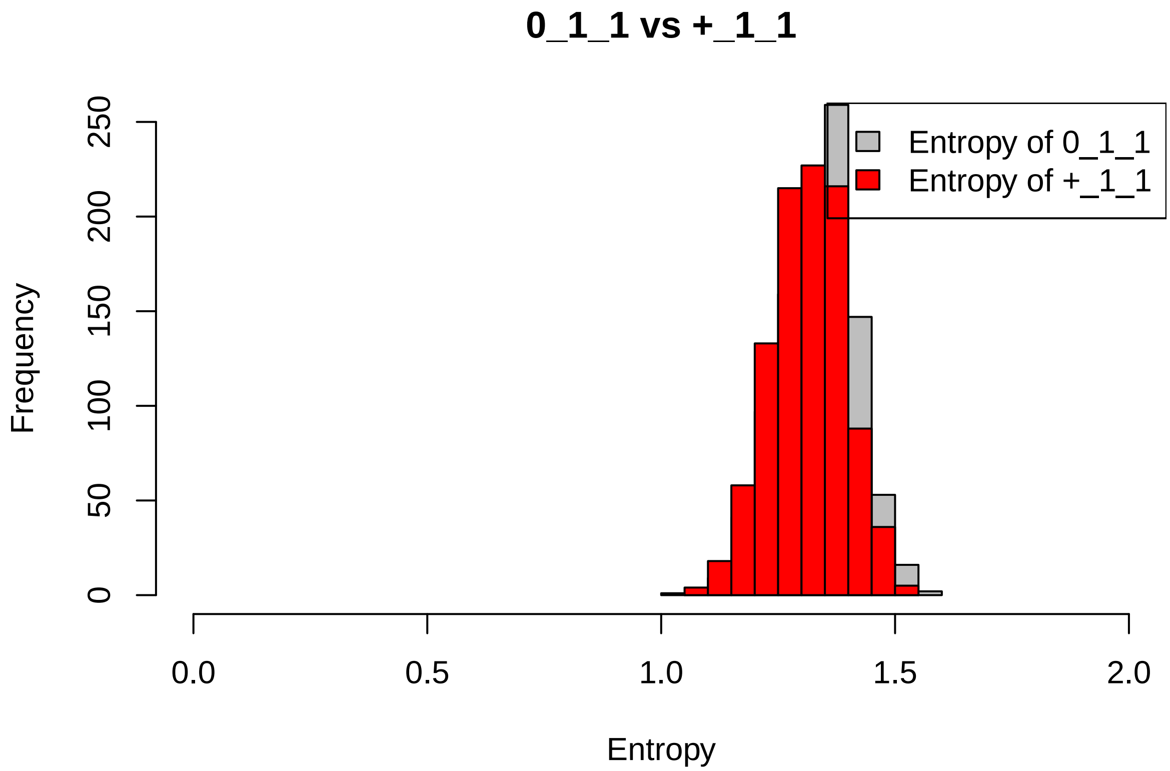
No HD __ Yes HighBP __ Yes HighChol



Looking at these graphs below, there is not much change happening. Thus, missing one bi-variable has little effect on 0_1_1. Since the graphs mostly overlap the origin, there is no interaction effect on the BMI distribution.

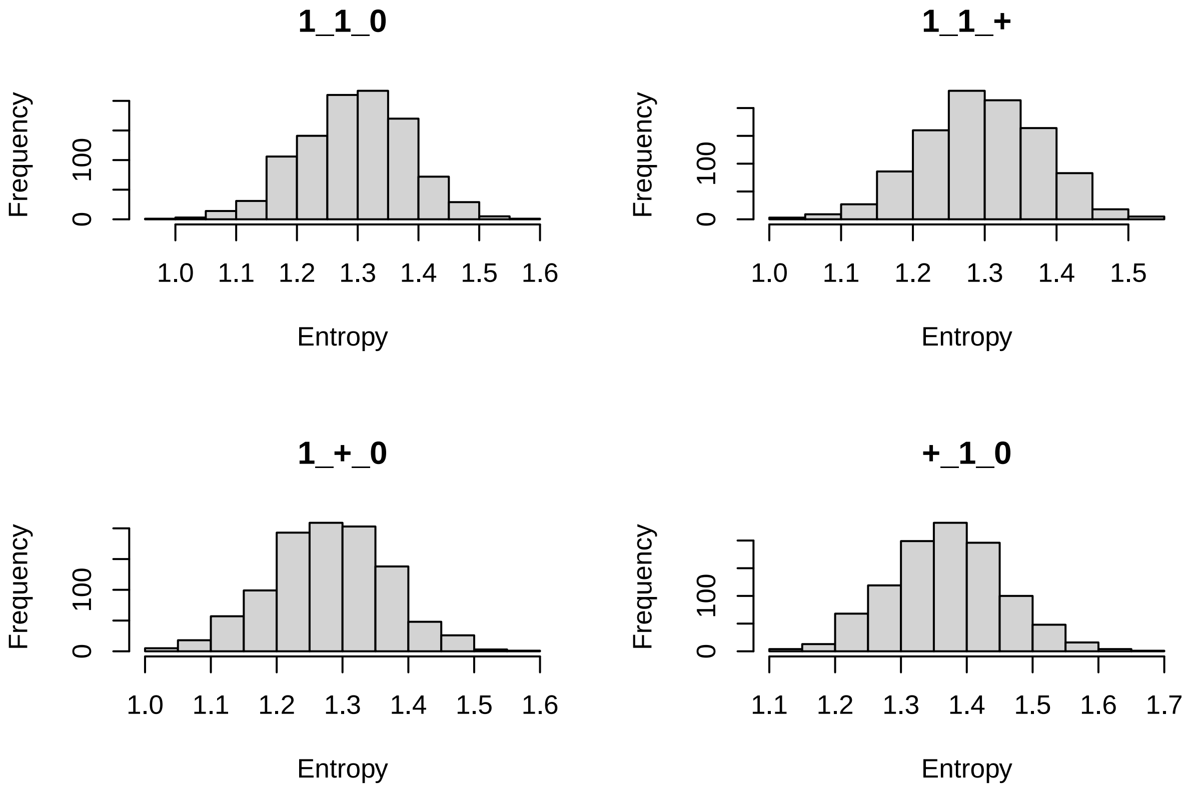

0_1_1 vs 0_1_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 0_1_1
Entropy of 0_1_+


0_1_1 vs 0_+_1
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 0_1_1
Entropy of 0_+_1

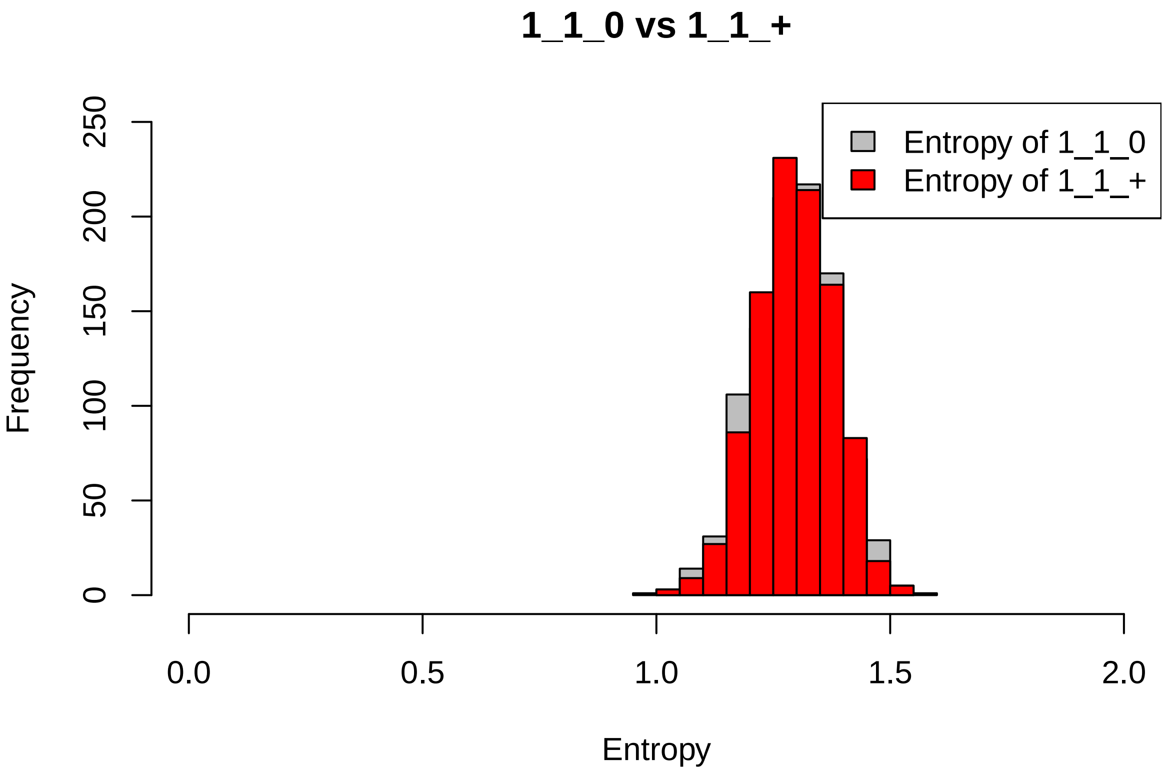


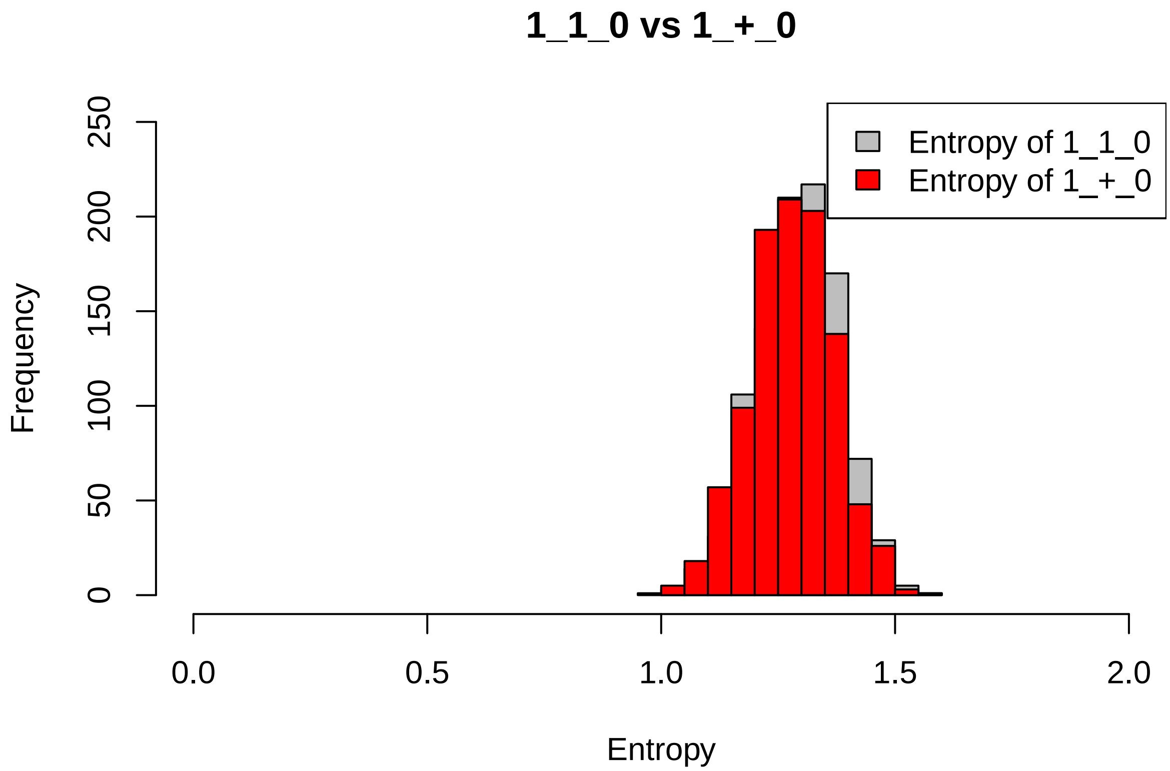
0_1_1 vs +_1_1
Frequency
0
50
100
150
200
250
Entropy of 0_1_1
Entropy of +_1_1
0.0
0.5
1.0
1.5
2.0
Entropy

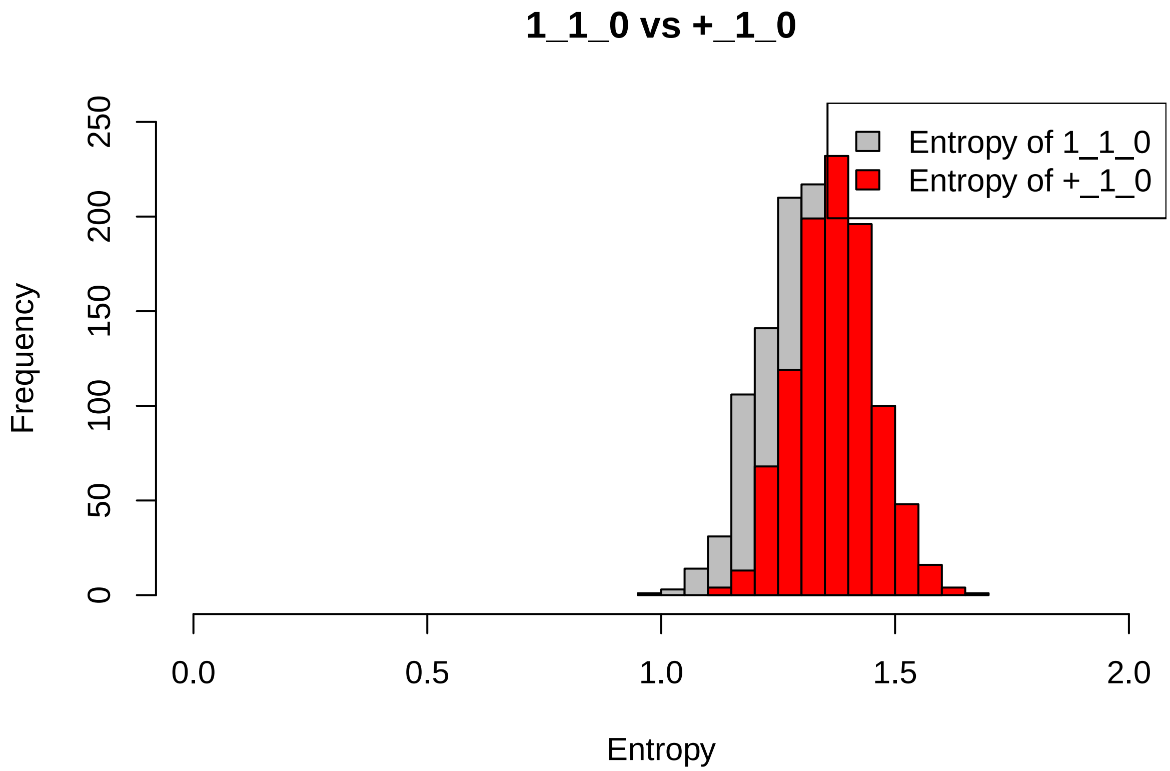
Yes HD _ Yes HighBP _ No HighChol



Looking at these graphs below, there is not much change happening. There is a clear shift to the right when missing HeartDisease, which means that the entropy level goes up, and makes it less accurate. Since the graphs mostly overlap the origin, there is no interaction effect on the BMI distribution. Thus, missing one bi-variable has little effect on 1_1_0.


1_1_0 vs 1_1_+
Frequency
0
50
100
150
200
250
0.0
0.5
1.0
1.5
2.0
Entropy
Entropy of 1_1_0
Entropy of 1_1_+


1_1_0 vs 1_+_0
Frequency
0
50
100
150
200
250
Entropy
0.0
0.5
1.0
1.5
2.0
Entropy of 1_1_0
Entropy of 1_+_0



Analysis and conclusion based on the results of the 5 investigations

HC Tree Analysis:

- There seems to be many interesting factors found in each HC-tree of the 5 sub-dataset. For example, the GenHealth_1's HC tree is completely different from the other trees. In the tree, there are two branches (clusters), and the majority of the leaves lie in the right branch. In addition, there is an interesting leaf "1_0_1" which separates itself from the other leaves.

- The HC trees of GenHealth_2, GenHealth_3, GenHealth_4 are almost similar to each other. They all have two branches (clusters), and the leaves are evenly divided in each branch. Furtheremore, the leaves pairing in each branch seem to be the same. For instance, "0_0_0" and "1_0_0" in the HC trees of GenHealth_2 and GenHealth_3 are always pair together in a branch, and they stay in the right cluster. However, the height of the trees is growing bigger, which indicates that the two branches are growing more distant from each other.

- For GenHealth_5's HC tree, it is also different from the rest of the graphs as its branches and leaves are more balanced. All leaves have a pair in each branch.

- Thus, we conclude that each sub-dataset have different behaviors based on the 3-way analysis which we have conducted. The HC trees of GenHealth_2, GenHealth_3, and GenHealth_4 have similarity in each other, but GenHealth_1 and GenHealth_5 have completely different trees compared to the others.

Entropy Analysis:

- Based on the observations of the entropy comparison in each sub-dataset, we can see that there are either a clear change or almost no change when we take away one categorical variable from each subunit category. In the entropy histograms of each subunit category, we can see a shift change either to the right or the left when we remove one categorical variable from the subunit category. For example, looking at the entropy histogram of the subunit category "1_1_1" in GenHealth_1, we can see that there is a clear shift to the left when the HeartDisease or HighBP variable is missing, and this shift indicates that there is no overlap happening when this shift occurs. Thus, we prove that there is an interaction effect on the BMI distribution. However, there is also the case where there is little change or almost no change when we remove one categorical variable from the subunit category. Looking back to the entropy histogram, when the HighChol variable is missing, we see that there is almost no change, which indicates that there is an overlap happening. Thus, it indicates that there is no interaction effect on BMI distribution when this happens.
- As we continue to the same process in each sub-data set, we can see some similarities and differences happening when we compare two of the same subunit categories in different sub-dataset. For instance, looking at the entropy histograms of the subunit category "1_1_0" in each sub-dataset, we can see that most of the graphs indicate a small shift from either the left or right which shows no significance change on the BMI distribution. However, in entropy histograms of GenHealth_2 sub-dataset, we can see a clear change to the left when missing HighBP or HighChol, and thus, we are able to prove that there is an interaction effect when a variable is missing in this case. Therefore, despite having different sub-dataset, the subunit category still indicates some similarity or difference in the entropy.
- However, there is a specific subunit category which keeps the same changes throughout all sub-dataset. In the entropy histograms of the subunit category "1_0_1" in each sub-dataset, there is always a strong shift to the right when missing HeartDisease or HighBP or HighChol. This indicates that this specific subunit category will never overlap to the origin. Thus, we have proven that there is a strong interaction effect on the BMI distribution.

Code Appendix

```
heart_disease <- read.csv("C:/Users/hoang/Downloads/STA
106/heart_disease_health_indicators_BRFSS2015.csv", header = TRUE)
# first subset: GenHealth_1
GenHealth_1 = heart_disease[heart_disease$GenHlth==1,]
head(GenHealth_1,10)
# second subset: GenHealth_2
GenHealth_2 = heart_disease[heart_disease$GenHlth==2,]
head(GenHealth_2,10)
# third subset: GenHealth_3
GenHealth_3 = heart_disease[heart_disease$GenHlth==3,]
head(GenHealth_3,10)
# fourth subset: GenHealth_4
GenHealth_4 = heart_disease[heart_disease$GenHlth==4,]
head(GenHealth_4,10)
# fifth subset: GenHealth_5
GenHealth_5 = heart_disease[heart_disease$GenHlth==5,]
head(GenHealth_5,10)
# combine several binary HeartDiseaseorAttack, HighBP, HighChol
# into one categorical variable
# as.factor(): covert to factor variable
GenHealth_1$HeartDiseaseorAttack=as.factor(GenHealth_1$HeartDiseaseorAttack)
GenHealth_1$HighBP=as.factor(GenHealth_1$HighBP)
GenHealth_1$HighChol=as.factor(GenHealth_1$HighChol)
# Combine specified categorical variables by
# concatenating their values into one character
# function combineCatVars(.data,vars,sep = ".")
# .data: a dataframe with the columns to be combined
# vars: a character vector of the categorical variables to be combined
# sep: the separator to combine the values of the variables in var
library(iNZightTools)
Combined_1 = combineCatVars(GenHealth_1, vars = c('HeartDiseaseorAttack', 'HighBP',
'HighChol'), sep = "_")
colnames(Combined_1)[23] <- "Combination_1"
# function Build_contigencytable(data,group,variable,bins=10,proportion=FALSE)
Build_contigencytable=function(data,group,variable,bins=10,proportion=FALSE){
table1=NULL
# create break points for the following histograms
# from minimum to maximum with equal distance
ax=seq(min(data[,variable]),max(data[,variable]),length.out=bins+1)
# Save histogram data
list_group=unique(data[,group])
for(i in list_group){
```

```
hg1=hist(data[data[,group]==i,variable], breaks = ax,plot = FALSE)
table1=rbind(table1,hg1$counts)
}
rownames(table1)=list_group
colnames(table1)=1:ncol(table1)
# calculate row sum and combine it with the current table
table1=cbind(table1, 'Total'=apply(table1,1,sum))
# calculate column sum and combine it with the current table
table1=rbind(table1, 'Total'=apply(table1,2,sum))
if(proportion){
# convert to proportions
n_col=ncol(table1)
for(i in 1:nrow(table1)){
table1[i,]=table1[i,]/table1[i,n_col]
}
}
table1
}
# build proportion contigency table for the histogram of BMI with group factor combination
# logical value T=TRUE F=FALSE
table_1=Build_contigencytable(Combined_1,"Combination_1","BMI",10,TRUE)
# discard the last column all are ones.
# 1:(ncol(df)-1): a consecutive vector of 1,2,.,. ncol(df)-1
# ncol(table_1): return the number of columns of df
table_1=table_1[,1:(ncol(table_1)-1)]
table_1
# combine several binary HeartDiseaseorAttack, HighBP
# into one categorical variable
# as.factor(): covert to factor variable
GenHealth_1$HeartDiseaseorAttack=as.factor(GenHealth_1$HeartDiseaseorAttack)
GenHealth_1$HighBP=as.factor(GenHealth_1$HighBP)
# Combine specified categorical variables by
# concatenating their values into one character
# function combineCatVars(.data,vars,sep = ".")
# .data: a dataframe with the columns to be combined
# vars: a character vector of the categorical variables to be combined
# sep: the separator to combine the values of the variables in var
library(iNZightTools)
Combined_2 = combineCatVars(GenHealth_1, vars = c('HeartDiseaseorAttack', 'HighBP'), sep =
"_")
colnames(Combined_2)[23] <- "Combination_2"
# function Build_contigencytable(data,group,variable,bins=10,proportion=FALSE)
Build_contigencytable=function(data,group,variable,bins=10,proportion=FALSE){
table2=NULL
```

```
# create break points for the following histograms
# from minimum to maximum with equal distance
ax=seq(min(data[,variable]),max(data[,variable]),length.out=bins+1)
# Save histogram data
list_group=unique(data[,group])
for(i in list_group){
hg2=hist(data[data[,group]==i,variable], breaks = ax,plot = FALSE)
table2=rbind(table2,hg2$counts)
}
rownames(table2)=list_group
colnames(table2)=1:ncol(table2)
# calculate row sum and combine it with the current table
table2=cbind(table2, 'Total'=apply(table2,1,sum))
# calculate column sum and combine it with the current table
table2=rbind(table2, 'Total'=apply(table2,2,sum))
if(proportion){
# convert to proportions
n_col=ncol(table2)
for(i in 1:nrow(table2)){
table2[i,]=table2[i,]/table2[i,n_col]
}
}
table2
}
# build proportion contigency table for the histogram of BMI with group factor combination
# logical value T=TRUE F=FALSE
table_2=Build_contigencytable(Combined_2,"Combination_2","BMI",10,TRUE)
# discard the last column all are ones.
# 1:(ncol(df)-1): a consecutive vector of 1,2,.,. ncol(df)-1
# ncol(table_1): return the number of columns of df
table_2=table_2[,1:(ncol(table_2)-1)]
table_2
# combine several binary HeartDiseaseorAttack, HighChol
# into one categorical variable
# as.factor(): covert to factor variable
GenHealth_1$HeartDiseaseorAttack=as.factor(GenHealth_1$HeartDiseaseorAttack)
GenHealth_1$HighChol=as.factor(GenHealth_1$HighChol)
# Combine specified categorical variables by
# concatenating their values into one character
# function combineCatVars(.data,vars,sep = ".")
# .data: a dataframe with the columns to be combined
# vars: a character vector of the categorical variables to be combined
# sep: the separator to combine the values of the variables in var
library(iNZightTools)
```

```
Combined_3 = combineCatVars(GenHealth_1, vars = c('HeartDiseaseorAttack', 'HighChol'), sep
= "_")
colnames(Combined_3)[23] <- "Combination_3"
# function Build_contigencytable(data,group,variable,bins=10,proportion=FALSE)
Build_contigencytable=function(data,group,variable,bins=10,proportion=FALSE){
table3=NULL
# create break points for the following histograms
# from minimum to maximum with equal distance
ax=seq(min(data[,variable]),max(data[,variable]),length.out=bins+1)
# Save histogram data
list_group=unique(data[,group])
for(i in list_group){
hg3=hist(data[data[,group]==i,variable], breaks = ax,plot = FALSE)
table3=rbind(table3,hg3$counts)
}
rownames(table3)=list_group
colnames(table3)=1:ncol(table3)
# calculate row sum and combine it with the current table
table3=cbind(table3, 'Total'=apply(table3,1,sum))
# calculate column sum and combine it with the current table
table3=rbind(table3, 'Total'=apply(table3,2,sum))
if(proportion){
# convert to proportions
n_col=ncol(table3)
for(i in 1:nrow(table3)){
table3[i,]=table3[i,]/table3[i,n_col]
}
}
table3
}
# build proportion contigency table for the histogram of BMI with group factor combination
# logical value T=TRUE F=FALSE
table_3=Build_contigencytable(Combined_3,"Combination_3","BMI",10,TRUE)
# discard the last column all are ones.
# 1:(ncol(df)-1): a consecutive vector of 1,2,.,. ncol(df)-1
# ncol(table_1): return the number of columns of df
table_3=table_3[,1:(ncol(table_3)-1)]
table_3
# combine several binary HighBP, HighChol
# into one categorical variable
# as.factor(): covert to factor variable
GenHealth_1$HighBP=as.factor(GenHealth_1$HighBP)
GenHealth_1$HighChol=as.factor(GenHealth_1$HighChol)
# Combine specified categorical variables by
# concatenating their values into one character
# function combineCatVars(.data,vars,sep = ".")
# .data: a dataframe with the columns to be combined
# vars: a character vector of the categorical variables to be combined
# sep: the separator to combine the values of the variables in var
library(iNZightTools)
Combined_4 = combineCatVars(GenHealth_1, vars = c('HighBP', 'HighChol'), sep = "_")
colnames(Combined_4)[23] <- "Combination_4"
# function Build_contigencytable(data,group,variable,bins=10,proportion=FALSE)
Build_contigencytable=function(data,group,variable,bins=10,proportion=FALSE){
table4=NULL
# create break points for the following histograms
# from minimum to maximum with equal distance
ax=seq(min(data[,variable]),max(data[,variable]),length.out=bins+1)
# Save histogram data
list_group=unique(data[,group])
for(i in list_group){
hg4=hist(data[data[,group]==i,variable], breaks = ax,plot = FALSE)
table4=rbind(table4,hg4$counts)
}
rownames(table4)=list_group
colnames(table4)=1:ncol(table4)
# calculate row sum and combine it with the current table
table4=cbind(table4, 'Total'=apply(table4,1,sum))
# calculate column sum and combine it with the current table
table4=rbind(table4, 'Total'=apply(table4,2,sum))
if(proportion){
# convert to proportions
n_col=ncol(table4)
for(i in 1:nrow(table4)){
table4[i,]=table4[i,]/table4[i,n_col]
}
}
table4
}
# build proportion contigency table for the histogram of BMI with group factor combination
# logical value T=TRUE F=FALSE
table_4=Build_contigencytable(Combined_4,"Combination_4","BMI",10,TRUE)
# discard the last column all are ones.
# 1:(ncol(df)-1): a consecutive vector of 1,2,.,. ncol(df)-1
# ncol(table_1): return the number of columns of df
table_4=table_4[,1:(ncol(table_4)-1)]
table_4
table_1 <- head(table_1, -1)
```

```
clusters <- hclust(dist(table_1))
plot(clusters,xlab='',main='Dendrogram')
# cut off the tree at the desired number of clusters using cutree.
clusterCut <- cutree(clusters, 3)
# plot the tree
rect.hclust(clusters, k=3)
# load package DescTools
library(DescTools)
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.
# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[1,])
sample2=rmultinom(1000,100,table_2[1,])
sample3=rmultinom(1000,100,table_3[1,])
sample4=rmultinom(1000,100,table_4[1,])
A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)
# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='0_1_0',xlab='Entropy')
hist(entropy_2,main='0_1_+',xlab='Entropy')
hist(entropy_3,main='0_+_0',xlab='Entropy')
hist(entropy_4,main='+_1_0',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_0 vs 0_1_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
```

```
legend('topright', c('Entropy of 0_1_0', 'Entropy of 0_1_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_0 vs 0_+_0', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_0', 'Entropy of 0_+_0'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_0 vs +_1_0', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_0', 'Entropy of +_1_0'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.
# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[2,])
sample2=rmultinom(1000,100,table_2[2,])
sample3=rmultinom(1000,100,table_3[2,])
sample4=rmultinom(1000,100,table_4[2,])
A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)
# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='0_0_1',xlab='Entropy')
hist(entropy_2,main='0_0_+',xlab='Entropy')
hist(entropy_3,main='0_+_1',xlab='Entropy')
hist(entropy_4,main='+_0_1',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_1 vs 0_0_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_1', 'Entropy of 0_0_+'), fill=c('gray', 'red'))
```

```
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_1 vs 0_+_1', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_1', 'Entropy of 0_+_1'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_1 vs +_0_1', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_1', 'Entropy of +_0_1'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.
# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[3,])
sample2=rmultinom(1000,100,table_2[2,])
sample3=rmultinom(1000,100,table_3[1,])
sample4=rmultinom(1000,100,table_4[3,])
A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)
# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='0_0_0',xlab='Entropy')
hist(entropy_2,main='0_0_+',xlab='Entropy')
hist(entropy_3,main='0_+_0',xlab='Entropy')
hist(entropy_4,main='+_0_0',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_0 vs 0_0_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_0', 'Entropy of 0_0_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_0 vs 0_+_0', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_0', 'Entropy of 0_+_0'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_0 vs +_0_0', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_0', 'Entropy of +_0_0'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.
# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[4,])
sample2=rmultinom(1000,100,table_2[3,])
sample3=rmultinom(1000,100,table_3[3,])
sample4=rmultinom(1000,100,table_4[4,])
A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)
# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='1_1_1',xlab='Entropy')
hist(entropy_2,main='1_1_+',xlab='Entropy')
hist(entropy_3,main='1_+_1',xlab='Entropy')
hist(entropy_4,main='+_1_1',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_1 vs 1_1_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_1', 'Entropy of 1_1_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_1 vs 1_+_1', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
```

```
legend('topright', c('Entropy of 1_1_1', 'Entropy of 1_+_1'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_1 vs +_1_1', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_1', 'Entropy of +_1_1'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.
# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[5,])
sample2=rmultinom(1000,100,table_2[4,])
sample3=rmultinom(1000,100,table_3[3,])
sample4=rmultinom(1000,100,table_4[2,])
A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)
# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='1_0_1',xlab='Entropy')
hist(entropy_2,main='1_0_+',xlab='Entropy')
hist(entropy_3,main='1_+_1',xlab='Entropy')
hist(entropy_4,main='+_0_1',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_1 vs 1_0_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_1', 'Entropy of 1_0_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_1 vs 1_+_1', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_1', 'Entropy of 1_+_1'), fill=c('gray', 'red'))
```

```
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_1 vs +_0_1', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_1', 'Entropy of +_0_1'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.
# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[6,])
sample2=rmultinom(1000,100,table_2[4,])
sample3=rmultinom(1000,100,table_3[4,])
sample4=rmultinom(1000,100,table_4[3,])
A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)
# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='1_0_0',xlab='Entropy')
hist(entropy_2,main='1_0_+',xlab='Entropy')
hist(entropy_3,main='1_+_0',xlab='Entropy')
hist(entropy_4,main='+_0_0',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_0 vs 1_0_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_0', 'Entropy of 1_0_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_0 vs 1_+_0', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_0', 'Entropy of 1_+_0'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_0 vs +_0_0', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_0', 'Entropy of +_0_0'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.
# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[7,])
sample2=rmultinom(1000,100,table_2[1,])
sample3=rmultinom(1000,100,table_3[2,])
sample4=rmultinom(1000,100,table_4[4,])
A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)
# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='1_1_1',xlab='Entropy')
hist(entropy_2,main='1_1_+',xlab='Entropy')
hist(entropy_3,main='1_+_1',xlab='Entropy')
hist(entropy_4,main='+_1_1',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_1 vs 0_1_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_1', 'Entropy of 0_1_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_1 vs 0_+_1', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_1', 'Entropy of 0_+_1'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_1 vs +_1_1', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
```

```
legend('topright', c('Entropy of 0_1_1', 'Entropy of +_1_1'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.
# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[8,])
sample2=rmultinom(1000,100,table_2[3,])
sample3=rmultinom(1000,100,table_3[4,])
sample4=rmultinom(1000,100,table_4[1,])
A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)
# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='1_1_0',xlab='Entropy')
hist(entropy_2,main='1_1_+',xlab='Entropy')
hist(entropy_3,main='1_+_0',xlab='Entropy')
hist(entropy_4,main='+_1_0',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_0 vs 1_1_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_0', 'Entropy of 1_1_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_0 vs 1_+_0', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_0', 'Entropy of 1_+_0'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_0 vs +_1_1', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_0', 'Entropy of +_1_0'), fill=c('gray', 'red'))
```

```
# combine several binary HeartDiseaseorAttack, HighBP, HighChol
# into one categorical variable

# as.factor(): covert to factor variable
GenHealth_2$HeartDiseaseorAttack=as.factor(GenHealth_2$HeartDiseaseorAttack)
GenHealth_2$HighBP=as.factor(GenHealth_2$HighBP)
GenHealth_2$HighChol=as.factor(GenHealth_2$HighChol)

# Combine specified categorical variables by
# concatenating their values into one character
# function combineCatVars(.data,vars,sep = ".")
# .data:    a dataframe with the columns to be combined
#  vars: a character vector of the categorical variables to be combined
#  sep: the separator to combine the values of the variables in var
library(iNZightTools)
Combined_1 = combineCatVars(GenHealth_2, vars = c('HeartDiseaseorAttack', 'HighBP',
'HighChol'), sep = "_")
colnames(Combined_1)[23]  <- "Combination_1"
# function Build_contigencytable(data,group,variable,bins=10,proportion=FALSE)
Build_contigencytable=function(data,group,variable,bins=10,proportion=FALSE){
  table1=NULL
  # create break points for the following histograms
  # from minimum to maximum with equal distance
  ax=seq(min(data[,variable]),max(data[,variable]),length.out=bins+1)
  # Save histogram data
  list_group=unique(data[,group])
  for(i in list_group){
    hg1=hist(data[data[,group]==i,variable], breaks = ax,plot = FALSE)
    table1=rbind(table1,hg1$counts)
  }
  rownames(table1)=list_group
  colnames(table1)=1:ncol(table1)
  # calculate row sum and combine it  with the current table
  table1=cbind(table1, 'Total'=apply(table1,1,sum))
  # calculate column sum and combine it  with the current table
  table1=rbind(table1, 'Total'=apply(table1,2,sum))

  if(proportion){
    # convert to proportions
    n_col=ncol(table1)
    for(i in 1:nrow(table1)){
      table1[i,]=table1[i,]/table1[i,n_col]
    }
  }
  table1
}

# build proportion contigency table for the histogram of BMI with group factor combination
# logical value T=TRUE F=FALSE
table_1=Build_contigencytable(Combined_1,"Combination_1","BMI",10,TRUE)

# discard the last column all are ones.
# 1:(ncol(df)-1): a consecutive vector of 1,2,.,. ncol(df)-1
# ncol(table_1): return the number of columns of df
table_1=table_1[,1:(ncol(table_1)-1)]
table_1
# combine several binary HeartDiseaseorAttack, HighBP
# into one categorical variable

# as.factor(): covert to factor variable
GenHealth_2$HeartDiseaseorAttack=as.factor(GenHealth_2$HeartDiseaseorAttack)
GenHealth_2$HighBP=as.factor(GenHealth_2$HighBP)

# Combine specified categorical variables by
# concatenating their values into one character
# function combineCatVars(.data,vars,sep = ".")
# .data:    a dataframe with the columns to be combined
#  vars: a character vector of the categorical variables to be combined
#  sep: the separator to combine the values of the variables in var
library(iNZightTools)
Combined_2 = combineCatVars(GenHealth_2, vars = c('HeartDiseaseorAttack', 'HighBP'), sep =
"_")
colnames(Combined_2)[23]  <- "Combination_2"
# function Build_contigencytable(data,group,variable,bins=10,proportion=FALSE)
Build_contigencytable=function(data,group,variable,bins=10,proportion=FALSE){
  table2=NULL
  # create break points for the following histograms
  # from minimum to maximum with equal distance
  ax=seq(min(data[,variable]),max(data[,variable]),length.out=bins+1)
  # Save histogram data
  list_group=unique(data[,group])
  for(i in list_group){
    hg2=hist(data[data[,group]==i,variable], breaks = ax,plot = FALSE)
    table2=rbind(table2,hg2$counts)
  }
  rownames(table2)=list_group
  colnames(table2)=1:ncol(table2)
  # calculate row sum and combine it  with the current table
  table2=cbind(table2, 'Total'=apply(table2,1,sum))
  # calculate column sum and combine it  with the current table
  table2=rbind(table2, 'Total'=apply(table2,2,sum))

  if(proportion){
    # convert to proportions
    n_col=ncol(table2)
    for(i in 1:nrow(table2)){
      table2[i,]=table2[i,]/table2[i,n_col]
    }
  }
  table2
}

# build proportion contigency table for the histogram of BMI with group factor combination
# logical value T=TRUE F=FALSE
table_2=Build_contigencytable(Combined_2,"Combination_2","BMI",10,TRUE)

# discard the last column all are ones.
# 1:(ncol(df)-1): a consecutive vector of 1,2,.,. ncol(df)-1
# ncol(table_1): return the number of columns of df
table_2=table_2[,1:(ncol(table_2)-1)]
table_2
# combine several binary HeartDiseaseorAttack, HighChol
# into one categorical variable

# as.factor(): covert to factor variable
GenHealth_2$HeartDiseaseorAttack=as.factor(GenHealth_2$HeartDiseaseorAttack)
GenHealth_2$HighChol=as.factor(GenHealth_2$HighChol)

# Combine specified categorical variables by
# concatenating their values into one character
# function combineCatVars(.data,vars,sep = ".")
# .data:    a dataframe with the columns to be combined
#  vars: a character vector of the categorical variables to be combined
#  sep: the separator to combine the values of the variables in var
library(iNZightTools)
Combined_3 = combineCatVars(GenHealth_2, vars = c('HeartDiseaseorAttack', 'HighChol'), sep
= "_")
colnames(Combined_3)[23]  <- "Combination_3"
# function Build_contigencytable(data,group,variable,bins=10,proportion=FALSE)
Build_contigencytable=function(data,group,variable,bins=10,proportion=FALSE){
  table3=NULL
  # create break points for the following histograms
  # from minimum to maximum with equal distance
  ax=seq(min(data[,variable]),max(data[,variable]),length.out=bins+1)
  # Save histogram data
  list_group=unique(data[,group])
  for(i in list_group){
    hg3=hist(data[data[,group]==i,variable], breaks = ax,plot = FALSE)
    table3=rbind(table3,hg3$counts)
  }
  rownames(table3)=list_group
  colnames(table3)=1:ncol(table3)
  # calculate row sum and combine it  with the current table
  table3=cbind(table3, 'Total'=apply(table3,1,sum))
  # calculate column sum and combine it  with the current table
  table3=rbind(table3, 'Total'=apply(table3,2,sum))

  if(proportion){
    # convert to proportions
    n_col=ncol(table3)
    for(i in 1:nrow(table3)){
      table3[i,]=table3[i,]/table3[i,n_col]
    }
  }
  table3
}

# build proportion contigency table for the histogram of BMI with group factor combination
# logical value T=TRUE F=FALSE
table_3=Build_contigencytable(Combined_3,"Combination_3","BMI",10,TRUE)

# discard the last column all are ones.
# 1:(ncol(df)-1): a consecutive vector of 1,2,.,. ncol(df)-1
# ncol(table_1): return the number of columns of df
table_3=table_3[,1:(ncol(table_3)-1)]
table_3
# combine several binary HighBP, HighChol
# into one categorical variable

# as.factor(): covert to factor variable

GenHealth_2$HighBP=as.factor(GenHealth_2$HighBP)
GenHealth_2$HighChol=as.factor(GenHealth_2$HighChol)

# Combine specified categorical variables by
# concatenating their values into one character
# function combineCatVars(.data,vars,sep = ".")
# .data:    a dataframe with the columns to be combined
#  vars: a character vector of the categorical variables to be combined
#  sep: the separator to combine the values of the variables in var
library(iNZightTools)
Combined_4 = combineCatVars(GenHealth_2, vars = c('HighBP', 'HighChol'), sep = "_")
colnames(Combined_4)[23]  <- "Combination_4"
# function Build_contigencytable(data,group,variable,bins=10,proportion=FALSE)
Build_contigencytable=function(data,group,variable,bins=10,proportion=FALSE){
  table4=NULL
  # create break points for the following histograms
  # from minimum to maximum with equal distance
  ax=seq(min(data[,variable]),max(data[,variable]),length.out=bins+1)
  # Save histogram data
  list_group=unique(data[,group])
  for(i in list_group){
    hg4=hist(data[data[,group]==i,variable], breaks = ax,plot = FALSE)
    table4=rbind(table4,hg4$counts)
  }
  rownames(table4)=list_group
  colnames(table4)=1:ncol(table4)
  # calculate row sum and combine it  with the current table
  table4=cbind(table4, 'Total'=apply(table4,1,sum))
  # calculate column sum and combine it  with the current table
  table4=rbind(table4, 'Total'=apply(table4,2,sum))

  if(proportion){
    # convert to proportions
    n_col=ncol(table4)
    for(i in 1:nrow(table4)){
      table4[i,]=table4[i,]/table4[i,n_col]
    }
  }
  table4
}

# build proportion contigency table for the histogram of BMI with group factor combination
# logical value T=TRUE F=FALSE
table_4=Build_contigencytable(Combined_4,"Combination_4","BMI",10,TRUE)

# discard the last column all are ones.
# 1:(ncol(df)-1): a consecutive vector of 1,2,.,. ncol(df)-1
# ncol(table_1): return the number of columns of df
```

```
table_4=table_4[,1:(ncol(table_4)-1)]
table_4
table_1 <- head(table_1, -1)
clusters <- hclust(dist(table_1))
plot(clusters,xlab='',main='Dendrogram')

# cut off the tree at the desired number of clusters using cutree.
clusterCut <- cutree(clusters, 3)
# plot the tree
rect.hclust(clusters, k=3)
# load package DescTools
library(DescTools)
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[1,])
sample2=rmultinom(1000,100,table_2[1,])
sample3=rmultinom(1000,100,table_3[1,])
sample4=rmultinom(1000,100,table_4[1,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
```

```
par(mfrow=c(2,2))
hist(entropy_1,main='0_1_0',xlab='Entropy')
hist(entropy_2,main='0_1_+',xlab='Entropy')
hist(entropy_3,main='0_+_0',xlab='Entropy')
hist(entropy_4,main='+_1_0',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_0 vs 0_1_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_0', 'Entropy of 0_1_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_0 vs 0_+_0', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_0', 'Entropy of 0_+_0'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_0 vs +_1_0', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_0', 'Entropy of +_1_0'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[4,])
sample2=rmultinom(1000,100,table_2[2,])
sample3=rmultinom(1000,100,table_3[2,])
sample4=rmultinom(1000,100,table_4[4,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
```

```r
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='0_0_1',xlab='Entropy')
hist(entropy_2,main='0_0_+',xlab='Entropy')
hist(entropy_3,main='0_+_1',xlab='Entropy')
hist(entropy_4,main='+_0_1',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_1 vs 0_0_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_1', 'Entropy of 0_0_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_1 vs 0_+_1', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_1', 'Entropy of 0_+_1'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_1 vs +_0_1', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_1', 'Entropy of +_0_1'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[3,])
sample2=rmultinom(1000,100,table_2[2,])
sample3=rmultinom(1000,100,table_3[1,])
sample4=rmultinom(1000,100,table_4[3,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
```

```
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='0_0_0',xlab='Entropy')
hist(entropy_2,main='0_0_+',xlab='Entropy')
hist(entropy_3,main='0_+_0',xlab='Entropy')
hist(entropy_4,main='+_0_0',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_0 vs 0_0_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_0', 'Entropy of 0_0_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_0 vs 0_+_0', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_0', 'Entropy of 0_+_0'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_0 vs +_0_0', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_0', 'Entropy of +_0_0'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[5,])
sample2=rmultinom(1000,100,table_2[3,])
sample3=rmultinom(1000,100,table_3[3,])
sample4=rmultinom(1000,100,table_4[2,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
```

```
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='1_1_1',xlab='Entropy')
hist(entropy_2,main='1_1_+',xlab='Entropy')
hist(entropy_3,main='1_+_1',xlab='Entropy')
hist(entropy_4,main='+_1_1',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_1 vs 1_1_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_1', 'Entropy of 1_1_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_1 vs 1_+_1', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_1', 'Entropy of 1_+_1'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_1 vs +_1_1', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_1', 'Entropy of +_1_1'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[8,])
sample2=rmultinom(1000,100,table_2[4,])
sample3=rmultinom(1000,100,table_3[3,])
sample4=rmultinom(1000,100,table_4[4,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='1_0_1',xlab='Entropy')
hist(entropy_2,main='1_0_+',xlab='Entropy')
hist(entropy_3,main='1_+_1',xlab='Entropy')
hist(entropy_4,main='+_0_1',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_1 vs 1_0_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_1', 'Entropy of 1_0_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_1 vs 1_+_1', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_1', 'Entropy of 1_+_1'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_1 vs +_0_1', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_1', 'Entropy of +_0_1'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[6,])
sample2=rmultinom(1000,100,table_2[4,])
sample3=rmultinom(1000,100,table_3[4,])
sample4=rmultinom(1000,100,table_4[3,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
```

```
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='1_0_0',xlab='Entropy')
hist(entropy_2,main='1_0_+',xlab='Entropy')
hist(entropy_3,main='1_+_0',xlab='Entropy')
hist(entropy_4,main='+_0_0',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_0 vs 1_0_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_0', 'Entropy of 1_0_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_0 vs 1_+_0', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_0', 'Entropy of 1_+_0'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_0 vs +_0_0', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_0', 'Entropy of +_0_0'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[2,])
sample2=rmultinom(1000,100,table_2[1,])
sample3=rmultinom(1000,100,table_3[2,])
sample4=rmultinom(1000,100,table_4[2,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
```

```
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='1_1_1',xlab='Entropy')
hist(entropy_2,main='1_1_+',xlab='Entropy')
hist(entropy_3,main='1_+_1',xlab='Entropy')
hist(entropy_4,main='+_1_1',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_1 vs 0_1_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_1', 'Entropy of 0_1_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_1 vs 0_+_1', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_1', 'Entropy of 0_+_1'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_1 vs +_1_1', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_1', 'Entropy of +_1_1'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[7,])
sample2=rmultinom(1000,100,table_2[3,])
sample3=rmultinom(1000,100,table_3[4,])
sample4=rmultinom(1000,100,table_4[1,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
```

```
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='1_1_0',xlab='Entropy')
hist(entropy_2,main='1_1_+',xlab='Entropy')
hist(entropy_3,main='1_+_0',xlab='Entropy')
hist(entropy_4,main='+_1_0',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_0 vs 1_1_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_0', 'Entropy of 1_1_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_0 vs 1_+_0', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_0', 'Entropy of 1_+_0'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_0 vs +_1_1', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_0', 'Entropy of +_1_0'), fill=c('gray', 'red'))

# combine several binary HeartDiseaseorAttack, HighBP, HighChol
# into one categorical variable

# as.factor(): covert to factor variable
GenHealth_3$HeartDiseaseorAttack=as.factor(GenHealth_3$HeartDiseaseorAttack)
GenHealth_3$HighBP=as.factor(GenHealth_3$HighBP)
GenHealth_3$HighChol=as.factor(GenHealth_3$HighChol)

# Combine specified categorical variables by
# concatenating their values into one character
# function combineCatVars(.data,vars,sep = ".")
# .data:    a dataframe with the columns to be combined
#  vars: a character vector of the categorical variables to be combined
#  sep: the separator to combine the values of the variables in var
library(iNZightTools)
```

```
Combined_1 = combineCatVars(GenHealth_3, vars = c('HeartDiseaseorAttack', 'HighBP',
'HighChol'), sep = "_")
colnames(Combined_1)[23]  <- "Combination_1"
# function Build_contigencytable(data,group,variable,bins=10,proportion=FALSE)
Build_contigencytable=function(data,group,variable,bins=10,proportion=FALSE){
  table1=NULL
  # create break points for the following histograms
  # from minimum to maximum with equal distance
  ax=seq(min(data[,variable]),max(data[,variable]),length.out=bins+1)
  # Save histogram data
  list_group=unique(data[,group])
  for(i in list_group){
    hg1=hist(data[data[,group]==i,variable], breaks = ax,plot = FALSE)
    table1=rbind(table1,hg1$counts)
  }
  rownames(table1)=list_group
  colnames(table1)=1:ncol(table1)
  # calculate row sum and combine it  with the current table
  table1=cbind(table1, 'Total'=apply(table1,1,sum))
  # calculate column sum and combine it  with the current table
  table1=rbind(table1, 'Total'=apply(table1,2,sum))

  if(proportion){
    # convert to proportions
    n_col=ncol(table1)
    for(i in 1:nrow(table1)){
      table1[i,]=table1[i,]/table1[i,n_col]
    }
  }
  table1
}

# build proportion contigency table for the histogram of BMI with group factor combination
# logical value T=TRUE F=FALSE
table_1=Build_contigencytable(Combined_1,"Combination_1","BMI",10,TRUE)

# discard the last column all are ones.
# 1:(ncol(df)-1): a consecutive vector of 1,2,.,. ncol(df)-1
# ncol(table_1): return the number of columns of df
table_1=table_1[,1:(ncol(table_1)-1)]
table_1
# combine several binary HeartDiseaseorAttack, HighBP
# into one categorical variable
# as.factor(): covert to factor variable
GenHealth_3$HeartDiseaseorAttack=as.factor(GenHealth_3$HeartDiseaseorAttack)
GenHealth_3$HighBP=as.factor(GenHealth_3$HighBP)

# Combine specified categorical variables by
# concatenating their values into one character
# function combineCatVars(.data,vars,sep = ".")
# .data:    a dataframe with the columns to be combined
#  vars: a character vector of the categorical variables to be combined
#  sep: the separator to combine the values of the variables in var
library(iNZightTools)
Combined_2 = combineCatVars(GenHealth_3, vars = c('HeartDiseaseorAttack', 'HighBP'), sep =
"_")
colnames(Combined_2)[23]  <- "Combination_2"
# function Build_contigencytable(data,group,variable,bins=10,proportion=FALSE)
Build_contigencytable=function(data,group,variable,bins=10,proportion=FALSE){
  table2=NULL
  # create break points for the following histograms
  # from minimum to maximum with equal distance
  ax=seq(min(data[,variable]),max(data[,variable]),length.out=bins+1)
  # Save histogram data
  list_group=unique(data[,group])
  for(i in list_group){
    hg2=hist(data[data[,group]==i,variable], breaks = ax,plot = FALSE)
    table2=rbind(table2,hg2$counts)
  }
  rownames(table2)=list_group
  colnames(table2)=1:ncol(table2)
  # calculate row sum and combine it  with the current table
  table2=cbind(table2, 'Total'=apply(table2,1,sum))
  # calculate column sum and combine it  with the current table
  table2=rbind(table2, 'Total'=apply(table2,2,sum))

  if(proportion){
    # convert to proportions
    n_col=ncol(table2)
    for(i in 1:nrow(table2)){
      table2[i,]=table2[i,]/table2[i,n_col]
    }
  }
  table2
}

# build proportion contigency table for the histogram of BMI with group factor combination
# logical value T=TRUE F=FALSE
table_2=Build_contigencytable(Combined_2,"Combination_2","BMI",10,TRUE)

# discard the last column all are ones.
# 1:(ncol(df)-1): a consecutive vector of 1,2,.,. ncol(df)-1
# ncol(table_1): return the number of columns of df
table_2=table_2[,1:(ncol(table_2)-1)]
table_2
# combine several binary HeartDiseaseorAttack, HighChol
# into one categorical variable

# as.factor(): covert to factor variable
GenHealth_3$HeartDiseaseorAttack=as.factor(GenHealth_3$HeartDiseaseorAttack)
GenHealth_3$HighChol=as.factor(GenHealth_3$HighChol)

# Combine specified categorical variables by
# concatenating their values into one character
# function combineCatVars(.data,vars,sep = ".")
# .data:    a dataframe with the columns to be combined
#  vars: a character vector of the categorical variables to be combined
#  sep: the separator to combine the values of the variables in var
library(iNZightTools)
Combined_3 = combineCatVars(GenHealth_3, vars = c('HeartDiseaseorAttack', 'HighChol'), sep
= "_")
colnames(Combined_3)[23]  <- "Combination_3"
# function Build_contigencytable(data,group,variable,bins=10,proportion=FALSE)
Build_contigencytable=function(data,group,variable,bins=10,proportion=FALSE){
  table3=NULL
  # create break points for the following histograms
  # from minimum to maximum with equal distance
  ax=seq(min(data[,variable]),max(data[,variable]),length.out=bins+1)
  # Save histogram data
  list_group=unique(data[,group])
  for(i in list_group){
    hg3=hist(data[data[,group]==i,variable], breaks = ax,plot = FALSE)
    table3=rbind(table3,hg3$counts)
  }
  rownames(table3)=list_group
  colnames(table3)=1:ncol(table3)
  # calculate row sum and combine it  with the current table
  table3=cbind(table3, 'Total'=apply(table3,1,sum))
  # calculate column sum and combine it  with the current table
  table3=rbind(table3, 'Total'=apply(table3,2,sum))
  if(proportion){
    # convert to proportions
    n_col=ncol(table3)
    for(i in 1:nrow(table3)){
      table3[i,]=table3[i,]/table3[i,n_col]
    }
  }
  table3
}

# build proportion contigency table for the histogram of BMI with group factor combination
# logical value T=TRUE F=FALSE
table_3=Build_contigencytable(Combined_3,"Combination_3","BMI",10,TRUE)

# discard the last column all are ones.
# 1:(ncol(df)-1): a consecutive vector of 1,2,.,. ncol(df)-1
# ncol(table_1): return the number of columns of df
table_3=table_3[,1:(ncol(table_3)-1)]
table_3
# combine several binary HighBP, HighChol
# into one categorical variable

# as.factor(): covert to factor variable

GenHealth_3$HighBP=as.factor(GenHealth_3$HighBP)
GenHealth_3$HighChol=as.factor(GenHealth_3$HighChol)

# Combine specified categorical variables by
# concatenating their values into one character
# function combineCatVars(.data,vars,sep = ".")
# .data:    a dataframe with the columns to be combined
#  vars: a character vector of the categorical variables to be combined
#  sep: the separator to combine the values of the variables in var
library(iNZightTools)
Combined_4 = combineCatVars(GenHealth_3, vars = c('HighBP', 'HighChol'), sep = "_")
colnames(Combined_4)[23]  <- "Combination_4"
# function Build_contigencytable(data,group,variable,bins=10,proportion=FALSE)
Build_contigencytable=function(data,group,variable,bins=10,proportion=FALSE){
  table4=NULL
  # create break points for the following histograms
  # from minimum to maximum with equal distance
  ax=seq(min(data[,variable]),max(data[,variable]),length.out=bins+1)
  # Save histogram data
  list_group=unique(data[,group])
  for(i in list_group){
    hg4=hist(data[data[,group]==i,variable], breaks = ax,plot = FALSE)
    table4=rbind(table4,hg4$counts)
  }
  rownames(table4)=list_group
  colnames(table4)=1:ncol(table4)
  # calculate row sum and combine it  with the current table
  table4=cbind(table4, 'Total'=apply(table4,1,sum))
  # calculate column sum and combine it  with the current table
  table4=rbind(table4, 'Total'=apply(table4,2,sum))

  if(proportion){
    # convert to proportions
    n_col=ncol(table4)
    for(i in 1:nrow(table4)){
      table4[i,]=table4[i,]/table4[i,n_col]
    }
  }
  table4
}

# build proportion contigency table for the histogram of BMI with group factor combination
# logical value T=TRUE F=FALSE
table_4=Build_contigencytable(Combined_4,"Combination_4","BMI",10,TRUE)

# discard the last column all are ones.
# 1:(ncol(df)-1): a consecutive vector of 1,2,.,. ncol(df)-1
# ncol(table_1): return the number of columns of df
table_4=table_4[,1:(ncol(table_4)-1)]
table_4
table_1 <- head(table_1, -1)
clusters <- hclust(dist(table_1))
plot(clusters,xlab='',main='Dendrogram')

# cut off the tree at the desired number of clusters using cutree.
clusterCut <- cutree(clusters, 3)
# plot the tree
rect.hclust(clusters, k=3)
# load package DescTools
library(DescTools)
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.
# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[2,])
sample2=rmultinom(1000,100,table_2[2,])
sample3=rmultinom(1000,100,table_3[1,])
sample4=rmultinom(1000,100,table_4[2,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='0_1_0',xlab='Entropy')
hist(entropy_2,main='0_1_+',xlab='Entropy')
hist(entropy_3,main='0_+_0',xlab='Entropy')
hist(entropy_4,main='+_1_0',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_0 vs 0_1_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_0', 'Entropy of 0_1_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_0 vs 0_+_0', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_0', 'Entropy of 0_+_0'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_0 vs +_1_0', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_0', 'Entropy of +_1_0'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[5,])
sample2=rmultinom(1000,100,table_2[1,])
sample3=rmultinom(1000,100,table_3[2,])
sample4=rmultinom(1000,100,table_4[4,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='0_0_1',xlab='Entropy')
hist(entropy_2,main='0_0_+',xlab='Entropy')
hist(entropy_3,main='0_+_1',xlab='Entropy')
hist(entropy_4,main='+_0_1',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_1 vs 0_0_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_1', 'Entropy of 0_0_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_1 vs 0_+_1', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_1', 'Entropy of 0_+_1'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_1 vs +_0_1', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_1', 'Entropy of +_0_1'), fill=c('gray', 'red'))
```

```
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[1,])
sample2=rmultinom(1000,100,table_2[1,])
sample3=rmultinom(1000,100,table_3[1,])
sample4=rmultinom(1000,100,table_4[1,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='0_0_0',xlab='Entropy')
hist(entropy_2,main='0_0_+',xlab='Entropy')
hist(entropy_3,main='0_+_0',xlab='Entropy')
hist(entropy_4,main='+_0_0',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_0 vs 0_0_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_0', 'Entropy of 0_0_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_0 vs 0_+_0', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_0', 'Entropy of 0_+_0'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_0 vs +_0_0', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_0', 'Entropy of +_0_0'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[4,])
sample2=rmultinom(1000,100,table_2[3,])
sample3=rmultinom(1000,100,table_3[3,])
sample4=rmultinom(1000,100,table_4[3,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='1_1_1',xlab='Entropy')
hist(entropy_2,main='1_1_+',xlab='Entropy')
hist(entropy_3,main='1_+_1',xlab='Entropy')
hist(entropy_4,main='+_1_1',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_1 vs 1_1_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_1', 'Entropy of 1_1_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_1 vs 1_+_1', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
```

```
legend('topright', c('Entropy of 1_1_1', 'Entropy of 1_+_1'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_1 vs +_1_1', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_1', 'Entropy of +_1_1'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[6,])
sample2=rmultinom(1000,100,table_2[4,])
sample3=rmultinom(1000,100,table_3[3,])
sample4=rmultinom(1000,100,table_4[4,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='1_0_1',xlab='Entropy')
hist(entropy_2,main='1_0_+',xlab='Entropy')
hist(entropy_3,main='1_+_1',xlab='Entropy')
hist(entropy_4,main='+_0_1',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_1 vs 1_0_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_1', 'Entropy of 1_0_+'), fill=c('gray', 'red'))
```

```
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_1 vs 1_+_1', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_1', 'Entropy of 1_+_1'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_1 vs +_0_1', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_1', 'Entropy of +_0_1'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[7,])
sample2=rmultinom(1000,100,table_2[4,])
sample3=rmultinom(1000,100,table_3[4,])
sample4=rmultinom(1000,100,table_4[1,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='1_0_0',xlab='Entropy')
hist(entropy_2,main='1_0_+',xlab='Entropy')
hist(entropy_3,main='1_+_0',xlab='Entropy')
hist(entropy_4,main='+_0_0',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_0 vs 1_0_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_0', 'Entropy of 1_0_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_0 vs 1_+_0', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_0', 'Entropy of 1_+_0'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_0 vs +_0_0', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_0', 'Entropy of +_0_0'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[3,])
sample2=rmultinom(1000,100,table_2[2,])
sample3=rmultinom(1000,100,table_3[2,])
sample4=rmultinom(1000,100,table_4[3,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='1_1_1',xlab='Entropy')
hist(entropy_2,main='1_1_+',xlab='Entropy')
hist(entropy_3,main='1_+_1',xlab='Entropy')
hist(entropy_4,main='+_1_1',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_1 vs 0_1_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_1', 'Entropy of 0_1_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_1 vs 0_+_1', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_1', 'Entropy of 0_+_1'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_1 vs +_1_1', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_1', 'Entropy of +_1_1'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[8,])
sample2=rmultinom(1000,100,table_2[3,])
sample3=rmultinom(1000,100,table_3[4,])
sample4=rmultinom(1000,100,table_4[2,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='1_1_0',xlab='Entropy')
hist(entropy_2,main='1_1_+',xlab='Entropy')
hist(entropy_3,main='1_+_0',xlab='Entropy')
hist(entropy_4,main='+_1_0',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_0 vs 1_1_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_0', 'Entropy of 1_1_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_0 vs 1_+_0', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_0', 'Entropy of 1_+_0'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_0 vs +_1_1', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_0', 'Entropy of +_1_0'), fill=c('gray', 'red'))

# combine several binary HeartDiseaseorAttack, HighBP, HighChol
# into one categorical variable

# as.factor(): covert to factor variable
GenHealth_4$HeartDiseaseorAttack=as.factor(GenHealth_4$HeartDiseaseorAttack)
GenHealth_4$HighBP=as.factor(GenHealth_4$HighBP)
GenHealth_4$HighChol=as.factor(GenHealth_4$HighChol)

# Combine specified categorical variables by
# concatenating their values into one character
# function combineCatVars(.data,vars,sep = ".")
# .data:    a dataframe with the columns to be combined
#  vars: a character vector of the categorical variables to be combined
#  sep: the separator to combine the values of the variables in var
library(iNZightTools)
Combined_1 = combineCatVars(GenHealth_4, vars = c('HeartDiseaseorAttack', 'HighBP',
'HighChol'), sep = "_")
colnames(Combined_1)[23]  <- "Combination_1"
# function Build_contigencytable(data,group,variable,bins=10,proportion=FALSE)
Build_contigencytable=function(data,group,variable,bins=10,proportion=FALSE){
  table1=NULL
  # create break points for the following histograms
  # from minimum to maximum with equal distance
  ax=seq(min(data[,variable]),max(data[,variable]),length.out=bins+1)
  # Save histogram data
  list_group=unique(data[,group])
  for(i in list_group){
    hg1=hist(data[data[,group]==i,variable], breaks = ax,plot = FALSE)
    table1=rbind(table1,hg1$counts)
  }
  rownames(table1)=list_group
  colnames(table1)=1:ncol(table1)
  # calculate row sum and combine it  with the current table
  table1=cbind(table1, 'Total'=apply(table1,1,sum))
  # calculate column sum and combine it  with the current table
  table1=rbind(table1, 'Total'=apply(table1,2,sum))

  if(proportion){
    # convert to proportions
    n_col=ncol(table1)
    for(i in 1:nrow(table1)){
      table1[i,]=table1[i,]/table1[i,n_col]
    }
  }
  table1
}

# build proportion contigency table for the histogram of BMI with group factor combination
# logical value T=TRUE F=FALSE
table_1=Build_contigencytable(Combined_1,"Combination_1","BMI",10,TRUE)

# discard the last column all are ones.
# 1:(ncol(df)-1): a consecutive vector of 1,2,.,. ncol(df)-1
# ncol(table_1): return the number of columns of df
table_1=table_1[,1:(ncol(table_1)-1)]
table_1
# combine several binary HeartDiseaseorAttack, HighBP
# into one categorical variable

# as.factor(): covert to factor variable
GenHealth_4$HeartDiseaseorAttack=as.factor(GenHealth_4$HeartDiseaseorAttack)
GenHealth_4$HighBP=as.factor(GenHealth_4$HighBP)

# Combine specified categorical variables by
# concatenating their values into one character
# function combineCatVars(.data,vars,sep = ".")
# .data:    a dataframe with the columns to be combined
#  vars: a character vector of the categorical variables to be combined
#  sep: the separator to combine the values of the variables in var
library(iNZightTools)
Combined_2 = combineCatVars(GenHealth_4, vars = c('HeartDiseaseorAttack', 'HighBP'), sep =
"_")
colnames(Combined_2)[23]  <- "Combination_2"
# function Build_contigencytable(data,group,variable,bins=10,proportion=FALSE)
Build_contigencytable=function(data,group,variable,bins=10,proportion=FALSE){
  table2=NULL
  # create break points for the following histograms
  # from minimum to maximum with equal distance
  ax=seq(min(data[,variable]),max(data[,variable]),length.out=bins+1)
  # Save histogram data
  list_group=unique(data[,group])
  for(i in list_group){
    hg2=hist(data[data[,group]==i,variable], breaks = ax,plot = FALSE)
    table2=rbind(table2,hg2$counts)
  }
  rownames(table2)=list_group
  colnames(table2)=1:ncol(table2)
  # calculate row sum and combine it  with the current table
  table2=cbind(table2, 'Total'=apply(table2,1,sum))
  # calculate column sum and combine it  with the current table
  table2=rbind(table2, 'Total'=apply(table2,2,sum))

  if(proportion){
    # convert to proportions
    n_col=ncol(table2)
    for(i in 1:nrow(table2)){
      table2[i,]=table2[i,]/table2[i,n_col]
    }
  }
  table2
}

# build proportion contigency table for the histogram of BMI with group factor combination
# logical value T=TRUE F=FALSE
table_2=Build_contigencytable(Combined_2,"Combination_2","BMI",10,TRUE)

# discard the last column all are ones.
# 1:(ncol(df)-1): a consecutive vector of 1,2,.,. ncol(df)-1
# ncol(table_1): return the number of columns of df
table_2=table_2[,1:(ncol(table_2)-1)]
table_2
# combine several binary HeartDiseaseorAttack, HighChol
# into one categorical variable

# as.factor(): covert to factor variable
GenHealth_4$HeartDiseaseorAttack=as.factor(GenHealth_4$HeartDiseaseorAttack)
GenHealth_4$HighChol=as.factor(GenHealth_4$HighChol)

# Combine specified categorical variables by
# concatenating their values into one character
# function combineCatVars(.data,vars,sep = ".")
# .data:    a dataframe with the columns to be combined
#  vars: a character vector of the categorical variables to be combined
#  sep: the separator to combine the values of the variables in var
library(iNZightTools)
Combined_3 = combineCatVars(GenHealth_4, vars = c('HeartDiseaseorAttack', 'HighChol'), sep
= "_")
colnames(Combined_3)[23]  <- "Combination_3"
# function Build_contigencytable(data,group,variable,bins=10,proportion=FALSE)
Build_contigencytable=function(data,group,variable,bins=10,proportion=FALSE){
  table3=NULL
  # create break points for the following histograms
  # from minimum to maximum with equal distance
  ax=seq(min(data[,variable]),max(data[,variable]),length.out=bins+1)
  # Save histogram data
  list_group=unique(data[,group])
  for(i in list_group){
    hg3=hist(data[data[,group]==i,variable], breaks = ax,plot = FALSE)
    table3=rbind(table3,hg3$counts)
  }
  rownames(table3)=list_group
  colnames(table3)=1:ncol(table3)
  # calculate row sum and combine it  with the current table
  table3=cbind(table3, 'Total'=apply(table3,1,sum))
  # calculate column sum and combine it  with the current table
  table3=rbind(table3, 'Total'=apply(table3,2,sum))

  if(proportion){
    # convert to proportions
    n_col=ncol(table3)
    for(i in 1:nrow(table3)){
      table3[i,]=table3[i,]/table3[i,n_col]
    }
  }
  table3
}

# build proportion contigency table for the histogram of BMI with group factor combination
# logical value T=TRUE F=FALSE
table_3=Build_contigencytable(Combined_3,"Combination_3","BMI",10,TRUE)

# discard the last column all are ones.
# 1:(ncol(df)-1): a consecutive vector of 1,2,.,. ncol(df)-1
# ncol(table_1): return the number of columns of df
table_3=table_3[,1:(ncol(table_3)-1)]
table_3
# combine several binary HighBP, HighChol
# into one categorical variable

# as.factor(): covert to factor variable

GenHealth_4$HighBP=as.factor(GenHealth_4$HighBP)
GenHealth_4$HighChol=as.factor(GenHealth_4$HighChol)

# Combine specified categorical variables by
# concatenating their values into one character
# function combineCatVars(.data,vars,sep = ".")
# .data:    a dataframe with the columns to be combined
#  vars: a character vector of the categorical variables to be combined
#  sep: the separator to combine the values of the variables in var
library(iNZightTools)
Combined_4 = combineCatVars(GenHealth_4, vars = c('HighBP', 'HighChol'), sep = "_")
colnames(Combined_4)[23]  <- "Combination_4"
# function Build_contigencytable(data,group,variable,bins=10,proportion=FALSE)
Build_contigencytable=function(data,group,variable,bins=10,proportion=FALSE){
  table4=NULL
  # create break points for the following histograms
  # from minimum to maximum with equal distance
  ax=seq(min(data[,variable]),max(data[,variable]),length.out=bins+1)
  # Save histogram data
  list_group=unique(data[,group])
  for(i in list_group){
    hg4=hist(data[data[,group]==i,variable], breaks = ax,plot = FALSE)
    table4=rbind(table4,hg4$counts)
  }
  rownames(table4)=list_group
  colnames(table4)=1:ncol(table4)
  # calculate row sum and combine it  with the current table
  table4=cbind(table4, 'Total'=apply(table4,1,sum))
  # calculate column sum and combine it  with the current table
  table4=rbind(table4, 'Total'=apply(table4,2,sum))

  if(proportion){
    # convert to proportions
    n_col=ncol(table4)
    for(i in 1:nrow(table4)){
      table4[i,]=table4[i,]/table4[i,n_col]
    }
  }
  table4
}

# build proportion contigency table for the histogram of BMI with group factor combination
# logical value T=TRUE F=FALSE
table_4=Build_contigencytable(Combined_4,"Combination_4","BMI",10,TRUE)

# discard the last column all are ones.
# 1:(ncol(df)-1): a consecutive vector of 1,2,.,. ncol(df)-1
# ncol(table_1): return the number of columns of df
table_4=table_4[,1:(ncol(table_4)-1)]
table_4
table_1 <- head(table_1, -1)
clusters <- hclust(dist(table_1))
plot(clusters,xlab='',main='Dendrogram')

# cut off the tree at the desired number of clusters using cutree.
clusterCut <- cutree(clusters, 3)
# plot the tree
rect.hclust(clusters, k=3)
# load package DescTools
library(DescTools)
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[5,])
sample2=rmultinom(1000,100,table_2[1,])
sample3=rmultinom(1000,100,table_3[3,])
sample4=rmultinom(1000,100,table_4[4,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
```

```
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='0_1_0',xlab='Entropy')
hist(entropy_2,main='0_1_+',xlab='Entropy')
hist(entropy_3,main='0_+_0',xlab='Entropy')
hist(entropy_4,main='+_1_0',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_0 vs 0_1_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_0', 'Entropy of 0_1_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_0 vs 0_+_0', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_0', 'Entropy of 0_+_0'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_0 vs +_1_0', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_0', 'Entropy of +_1_0'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[2,])
sample2=rmultinom(1000,100,table_2[2,])
sample3=rmultinom(1000,100,table_3[1,])
sample4=rmultinom(1000,100,table_4[2,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
```

```r
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='0_0_1',xlab='Entropy')
hist(entropy_2,main='0_0_+',xlab='Entropy')
hist(entropy_3,main='0_+_1',xlab='Entropy')
hist(entropy_4,main='+_0_1',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_1 vs 0_0_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_1', 'Entropy of 0_0_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_1 vs 0_+_1', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_1', 'Entropy of 0_+_1'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_1 vs +_0_1', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_1', 'Entropy of +_0_1'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[4,])
sample2=rmultinom(1000,100,table_2[2,])
sample3=rmultinom(1000,100,table_3[3,])
sample4=rmultinom(1000,100,table_4[3,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
```

```
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='0_0_0',xlab='Entropy')
hist(entropy_2,main='0_0_+',xlab='Entropy')
hist(entropy_3,main='0_+_0',xlab='Entropy')
hist(entropy_4,main='+_0_0',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_0 vs 0_0_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_0', 'Entropy of 0_0_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_0 vs 0_+_0', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_0', 'Entropy of 0_+_0'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_0 vs +_0_0', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_0', 'Entropy of +_0_0'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[3,])
sample2=rmultinom(1000,100,table_2[3,])
sample3=rmultinom(1000,100,table_3[2,])
sample4=rmultinom(1000,100,table_4[1,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)
```

```
# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='1_1_1',xlab='Entropy')
hist(entropy_2,main='1_1_+',xlab='Entropy')
hist(entropy_3,main='1_+_1',xlab='Entropy')
hist(entropy_4,main='+_1_1',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_1 vs 1_1_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_1', 'Entropy of 1_1_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_1 vs 1_+_1', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_1', 'Entropy of 1_+_1'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_1 vs +_1_1', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_1', 'Entropy of +_1_1'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[6,])
sample2=rmultinom(1000,100,table_2[4,])
sample3=rmultinom(1000,100,table_3[2,])
sample4=rmultinom(1000,100,table_4[2,])

A=ncol(sample1)
B=ncol(sample2)
```

```
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='1_0_1',xlab='Entropy')
hist(entropy_2,main='1_0_+',xlab='Entropy')
hist(entropy_3,main='1_+_1',xlab='Entropy')
hist(entropy_4,main='+_0_1',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_1 vs 1_0_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_1', 'Entropy of 1_0_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_1 vs 1_+_1', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_1', 'Entropy of 1_+_1'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_1 vs +_0_1', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_1', 'Entropy of +_0_1'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[7,])
sample2=rmultinom(1000,100,table_2[4,])
sample3=rmultinom(1000,100,table_3[4,])
sample4=rmultinom(1000,100,table_4[3,])
```

```
A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='1_0_0',xlab='Entropy')
hist(entropy_2,main='1_0_+',xlab='Entropy')
hist(entropy_3,main='1_+_0',xlab='Entropy')
hist(entropy_4,main='+_0_0',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_0 vs 1_0_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_0', 'Entropy of 1_0_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_0 vs 1_+_0', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_0', 'Entropy of 1_+_0'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_0 vs +_0_0', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_0', 'Entropy of +_0_0'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[1,])
sample2=rmultinom(1000,100,table_2[3,])
sample3=rmultinom(1000,100,table_3[1,])
```

```
sample4=rmultinom(1000,100,table_4[1,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='1_1_1',xlab='Entropy')
hist(entropy_2,main='1_1_+',xlab='Entropy')
hist(entropy_3,main='1_+_1',xlab='Entropy')
hist(entropy_4,main='+_1_1',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_1 vs 0_1_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_1', 'Entropy of 0_1_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_1 vs 0_+_1', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_1', 'Entropy of 0_+_1'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_1 vs +_1_1', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_1', 'Entropy of +_1_1'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[8,])
```

```
sample2=rmultinom(1000,100,table_2[3,])
sample3=rmultinom(1000,100,table_3[4,])
sample4=rmultinom(1000,100,table_4[4,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='1_1_0',xlab='Entropy')
hist(entropy_2,main='1_1_+',xlab='Entropy')
hist(entropy_3,main='1_+_0',xlab='Entropy')
hist(entropy_4,main='+_1_0',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_0 vs 1_1_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_0', 'Entropy of 1_1_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_0 vs 1_+_0', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_0', 'Entropy of 1_+_0'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_0 vs +_1_1', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_0', 'Entropy of +_1_0'), fill=c('gray', 'red'))


# combine several binary HeartDiseaseorAttack, HighBP, HighChol
# into one categorical variable
# as.factor(): covert to factor variable
GenHealth_5$HeartDiseaseorAttack=as.factor(GenHealth_5$HeartDiseaseorAttack)
GenHealth_5$HighBP=as.factor(GenHealth_5$HighBP)
GenHealth_5$HighChol=as.factor(GenHealth_5$HighChol)

# Combine specified categorical variables by
# concatenating their values into one character
# function combineCatVars(.data,vars,sep = ".")
# .data:    a dataframe with the columns to be combined
#  vars: a character vector of the categorical variables to be combined
#  sep: the separator to combine the values of the variables in var
library(iNZightTools)
Combined_1 = combineCatVars(GenHealth_5, vars = c('HeartDiseaseorAttack', 'HighBP',
'HighChol'), sep = "_")
colnames(Combined_1)[23]  <- "Combination_1"
# function Build_contigencytable(data,group,variable,bins=10,proportion=FALSE)
Build_contigencytable=function(data,group,variable,bins=10,proportion=FALSE){
  table1=NULL
  # create break points for the following histograms
  # from minimum to maximum with equal distance
  ax=seq(min(data[,variable]),max(data[,variable]),length.out=bins+1)
  # Save histogram data
  list_group=unique(data[,group])
  for(i in list_group){
    hg1=hist(data[data[,group]==i,variable], breaks = ax,plot = FALSE)
    table1=rbind(table1,hg1$counts)
  }
  rownames(table1)=list_group
  colnames(table1)=1:ncol(table1)
  # calculate row sum and combine it  with the current table
  table1=cbind(table1, 'Total'=apply(table1,1,sum))
  # calculate column sum and combine it  with the current table
  table1=rbind(table1, 'Total'=apply(table1,2,sum))

  if(proportion){
    # convert to proportions
    n_col=ncol(table1)
    for(i in 1:nrow(table1)){
      table1[i,]=table1[i,]/table1[i,n_col]
    }
  }
  table1
}
```

```
# build proportion contigency table for the histogram of BMI with group factor combination
# logical value T=TRUE F=FALSE
table_1=Build_contigencytable(Combined_1,"Combination_1","BMI",10,TRUE)

# discard the last column all are ones.
# 1:(ncol(df)-1): a consecutive vector of 1,2,.,. ncol(df)-1
# ncol(table_1): return the number of columns of df
table_1=table_1[,1:(ncol(table_1)-1)]
table_1
# combine several binary HeartDiseaseorAttack, HighBP
# into one categorical variable

# as.factor(): covert to factor variable
GenHealth_5$HeartDiseaseorAttack=as.factor(GenHealth_5$HeartDiseaseorAttack)
GenHealth_5$HighBP=as.factor(GenHealth_5$HighBP)

# Combine specified categorical variables by
# concatenating their values into one character
# function combineCatVars(.data,vars,sep = ".")
# .data:    a dataframe with the columns to be combined
#  vars: a character vector of the categorical variables to be combined
#  sep: the separator to combine the values of the variables in var
library(iNZightTools)
Combined_2 = combineCatVars(GenHealth_5, vars = c('HeartDiseaseorAttack', 'HighBP'), sep =
"_")
colnames(Combined_2)[23]  <- "Combination_2"
# function Build_contigencytable(data,group,variable,bins=10,proportion=FALSE)
Build_contigencytable=function(data,group,variable,bins=10,proportion=FALSE){
  table2=NULL
  # create break points for the following histograms
  # from minimum to maximum with equal distance
  ax=seq(min(data[,variable]),max(data[,variable]),length.out=bins+1)
  # Save histogram data
  list_group=unique(data[,group])
  for(i in list_group){
    hg2=hist(data[data[,group]==i,variable], breaks = ax,plot = FALSE)
    table2=rbind(table2,hg2$counts)
  }
  rownames(table2)=list_group
  colnames(table2)=1:ncol(table2)
  # calculate row sum and combine it  with the current table
  table2=cbind(table2, 'Total'=apply(table2,1,sum))
  # calculate column sum and combine it  with the current table
  table2=rbind(table2, 'Total'=apply(table2,2,sum))

  if(proportion){
    # convert to proportions
    n_col=ncol(table2)
    for(i in 1:nrow(table2)){
      table2[i,]=table2[i,]/table2[i,n_col]
    }
  }
  table2
}

# build proportion contigency table for the histogram of BMI with group factor combination
# logical value T=TRUE F=FALSE
table_2=Build_contigencytable(Combined_2,"Combination_2","BMI",10,TRUE)

# discard the last column all are ones.
# 1:(ncol(df)-1): a consecutive vector of 1,2,.,. ncol(df)-1
# ncol(table_1): return the number of columns of df
table_2=table_2[,1:(ncol(table_2)-1)]
table_2
# combine several binary HeartDiseaseorAttack, HighChol
# into one categorical variable

# as.factor(): covert to factor variable
GenHealth_5$HeartDiseaseorAttack=as.factor(GenHealth_5$HeartDiseaseorAttack)
GenHealth_5$HighChol=as.factor(GenHealth_5$HighChol)

# Combine specified categorical variables by
# concatenating their values into one character
# function combineCatVars(.data,vars,sep = ".")
# .data:    a dataframe with the columns to be combined
#  vars: a character vector of the categorical variables to be combined
#  sep: the separator to combine the values of the variables in var
library(iNZightTools)
Combined_3 = combineCatVars(GenHealth_5, vars = c('HeartDiseaseorAttack', 'HighChol'), sep
= "_")
colnames(Combined_3)[23]  <- "Combination_3"
# function Build_contigencytable(data,group,variable,bins=10,proportion=FALSE)
Build_contigencytable=function(data,group,variable,bins=10,proportion=FALSE){
  table3=NULL
  # create break points for the following histograms
  # from minimum to maximum with equal distance
  ax=seq(min(data[,variable]),max(data[,variable]),length.out=bins+1)
  # Save histogram data
  list_group=unique(data[,group])
  for(i in list_group){
    hg3=hist(data[data[,group]==i,variable], breaks = ax,plot = FALSE)
    table3=rbind(table3,hg3$counts)
  }
  rownames(table3)=list_group
  colnames(table3)=1:ncol(table3)
  # calculate row sum and combine it  with the current table
  table3=cbind(table3, 'Total'=apply(table3,1,sum))
  # calculate column sum and combine it  with the current table
  table3=rbind(table3, 'Total'=apply(table3,2,sum))

  if(proportion){
    # convert to proportions
    n_col=ncol(table3)
    for(i in 1:nrow(table3)){
      table3[i,]=table3[i,]/table3[i,n_col]
    }
  }
  table3
}

# build proportion contigency table for the histogram of BMI with group factor combination
# logical value T=TRUE F=FALSE
table_3=Build_contigencytable(Combined_3,"Combination_3","BMI",10,TRUE)

# discard the last column all are ones.
# 1:(ncol(df)-1): a consecutive vector of 1,2,.,. ncol(df)-1
# ncol(table_1): return the number of columns of df
table_3=table_3[,1:(ncol(table_3)-1)]
table_3
# combine several binary HighBP, HighChol
# into one categorical variable

# as.factor(): covert to factor variable

GenHealth_5$HighBP=as.factor(GenHealth_5$HighBP)
GenHealth_5$HighChol=as.factor(GenHealth_5$HighChol)

# Combine specified categorical variables by
# concatenating their values into one character
# function combineCatVars(.data,vars,sep = ".")
# .data:    a dataframe with the columns to be combined
#  vars: a character vector of the categorical variables to be combined
#  sep: the separator to combine the values of the variables in var
library(iNZightTools)
Combined_4 = combineCatVars(GenHealth_5, vars = c('HighBP', 'HighChol'), sep = "_")
colnames(Combined_4)[23]  <- "Combination_4"
# function Build_contigencytable(data,group,variable,bins=10,proportion=FALSE)
Build_contigencytable=function(data,group,variable,bins=10,proportion=FALSE){
  table4=NULL
  # create break points for the following histograms
  # from minimum to maximum with equal distance
  ax=seq(min(data[,variable]),max(data[,variable]),length.out=bins+1)
  # Save histogram data
  list_group=unique(data[,group])
  for(i in list_group){
    hg4=hist(data[data[,group]==i,variable], breaks = ax,plot = FALSE)
    table4=rbind(table4,hg4$counts)
  }
  rownames(table4)=list_group
  colnames(table4)=1:ncol(table4)
  # calculate row sum and combine it  with the current table
  table4=cbind(table4, 'Total'=apply(table4,1,sum))
  # calculate column sum and combine it  with the current table
  table4=rbind(table4, 'Total'=apply(table4,2,sum))

  if(proportion){
    # convert to proportions
    n_col=ncol(table4)
    for(i in 1:nrow(table4)){
      table4[i,]=table4[i,]/table4[i,n_col]
    }
  }
  table4
}

# build proportion contigency table for the histogram of BMI with group factor combination
# logical value T=TRUE F=FALSE
table_4=Build_contigencytable(Combined_4,"Combination_4","BMI",10,TRUE)

# discard the last column all are ones.
# 1:(ncol(df)-1): a consecutive vector of 1,2,.,. ncol(df)-1
# ncol(table_1): return the number of columns of df
table_4=table_4[,1:(ncol(table_4)-1)]
table_4
table_1 <- head(table_1, -1)
```

```
clusters <- hclust(dist(table_1))
plot(clusters,xlab='',main='Dendrogram')

# cut off the tree at the desired number of clusters using cutree.
clusterCut <- cutree(clusters, 3)
# plot the tree
rect.hclust(clusters, k=3)
# load package DescTools
library(DescTools)
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[4,])
sample2=rmultinom(1000,100,table_2[1,])
sample3=rmultinom(1000,100,table_3[3,])
sample4=rmultinom(1000,100,table_4[3,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='0_1_0',xlab='Entropy')
hist(entropy_2,main='0_1_+',xlab='Entropy')
hist(entropy_3,main='0_+_0',xlab='Entropy')
hist(entropy_4,main='+_1_0',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_0 vs 0_1_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_0', 'Entropy of 0_1_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_0 vs 0_+_0', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_0', 'Entropy of 0_+_0'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_0 vs +_1_0', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_0', 'Entropy of +_1_0'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[3,])
sample2=rmultinom(1000,100,table_2[3,])
sample3=rmultinom(1000,100,table_3[1,])
sample4=rmultinom(1000,100,table_4[2,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
```

```
hist(entropy_1,main='0_0_1',xlab='Entropy')
hist(entropy_2,main='0_0_+',xlab='Entropy')
hist(entropy_3,main='0_+_1',xlab='Entropy')
hist(entropy_4,main='+_0_1',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_1 vs 0_0_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_1', 'Entropy of 0_0_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_1 vs 0_+_1', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_1', 'Entropy of 0_+_1'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_1 vs +_0_1', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_1', 'Entropy of +_0_1'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[5,])
sample2=rmultinom(1000,100,table_2[3,])
sample3=rmultinom(1000,100,table_3[3,])
sample4=rmultinom(1000,100,table_4[4,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='0_0_0',xlab='Entropy')
hist(entropy_2,main='0_0_+',xlab='Entropy')
hist(entropy_3,main='0_+_0',xlab='Entropy')
hist(entropy_4,main='+_0_0',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_0 vs 0_0_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_0', 'Entropy of 0_0_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_0 vs 0_+_0', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_0', 'Entropy of 0_+_0'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_0_0 vs +_0_0', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 0_0_0', 'Entropy of +_0_0'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[2,])
sample2=rmultinom(1000,100,table_2[2,])
sample3=rmultinom(1000,100,table_3[2,])
sample4=rmultinom(1000,100,table_4[1,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
```

```
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='1_1_1',xlab='Entropy')
hist(entropy_2,main='1_1_+',xlab='Entropy')
hist(entropy_3,main='1_+_1',xlab='Entropy')
hist(entropy_4,main='+_1_1',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_1 vs 1_1_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_1', 'Entropy of 1_1_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_1 vs 1_+_1', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_1', 'Entropy of 1_+_1'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_1 vs +_1_1', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_1', 'Entropy of +_1_1'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[8,])
sample2=rmultinom(1000,100,table_2[4,])
sample3=rmultinom(1000,100,table_3[2,])
sample4=rmultinom(1000,100,table_4[2,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='1_0_1',xlab='Entropy')
hist(entropy_2,main='1_0_+',xlab='Entropy')
hist(entropy_3,main='1_+_1',xlab='Entropy')
hist(entropy_4,main='+_0_1',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_1 vs 1_0_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_1', 'Entropy of 1_0_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_1 vs 1_+_1', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_1', 'Entropy of 1_+_1'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_1 vs +_0_1', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_1', 'Entropy of +_0_1'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[7,])
sample2=rmultinom(1000,100,table_2[4,])
sample3=rmultinom(1000,100,table_3[4,])
sample4=rmultinom(1000,100,table_4[4,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
```

```
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='1_0_0',xlab='Entropy')
hist(entropy_2,main='1_0_+',xlab='Entropy')
hist(entropy_3,main='1_+_0',xlab='Entropy')
hist(entropy_4,main='+_0_0',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_0 vs 1_0_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_0', 'Entropy of 1_0_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_0 vs 1_+_0', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_0', 'Entropy of 1_+_0'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_0_0 vs +_0_0', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 1_0_0', 'Entropy of +_0_0'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[1,])
sample2=rmultinom(1000,100,table_2[1,])
sample3=rmultinom(1000,100,table_3[1,])
sample4=rmultinom(1000,100,table_4[1,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
```

```
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='1_1_1',xlab='Entropy')
hist(entropy_2,main='1_1_+',xlab='Entropy')
hist(entropy_3,main='1_+_1',xlab='Entropy')
hist(entropy_4,main='+_1_1',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_1 vs 0_1_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_1', 'Entropy of 0_1_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_1 vs 0_+_1', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_1', 'Entropy of 0_+_1'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='0_1_1 vs +_1_1', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 0_1_1', 'Entropy of +_1_1'), fill=c('gray', 'red'))
# Entropy(x,base): calculate Shannon entropy
# x: contingency table
# base: base of the logarithm to be used, defaults to 2.

# generate 1000 random samples from multinomial distribution
sample1=rmultinom(1000,100,table_1[6,])
sample2=rmultinom(1000,100,table_2[2,])
sample3=rmultinom(1000,100,table_3[4,])
sample4=rmultinom(1000,100,table_4[3,])

A=ncol(sample1)
B=ncol(sample2)
C=ncol(sample3)
D=ncol(sample4)

# initialize a vector to save all entropies.
entropy_1=numeric(A)
entropy_2=numeric(B)
entropy_3=numeric(C)
entropy_4=numeric(D)
for(i in 1:A){
  entropy_1[i]=Entropy(sample1[,i],base=exp(1))
}
for(i in 1:B){
  entropy_2[i]=Entropy(sample2[,i],base=exp(1))
}
for(i in 1:C){
  entropy_3[i]=Entropy(sample3[,i],base=exp(1))
}
for(i in 1:D){
  entropy_4[i]=Entropy(sample4[,i],base=exp(1))
}
par(mfrow=c(2,2))
hist(entropy_1,main='1_1_0',xlab='Entropy')
hist(entropy_2,main='1_1_+',xlab='Entropy')
hist(entropy_3,main='1_+_0',xlab='Entropy')
hist(entropy_4,main='+_1_0',xlab='Entropy')
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_0 vs 1_1_+', xlab='Entropy')
hist(entropy_2, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_0', 'Entropy of 1_1_+'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_0 vs 1_+_0', xlab='Entropy')
hist(entropy_3, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_0', 'Entropy of 1_+_0'), fill=c('gray', 'red'))
hist(entropy_1, col='gray', xlim = c(0, 2), ylim = c(0,250), main='1_1_0 vs +_1_1', xlab='Entropy')
hist(entropy_4, col='red', add=TRUE)
legend('topright', c('Entropy of 1_1_0', 'Entropy of +_1_0'), fill=c('gray', 'red'))
```