Multiscale Multivariate Sample Entropy as an Indicator of System Health

Robert Kramer

December 9, 2016

processing

I ran 20 runs for the test set and 200 for the training set in Netlogo. I imported the data and processed it for use.

Here are the functions I wrote for caculating the entropy of the sequence tags and making the random walk

```
# takes frequency data and returns entropy
gamma <- function(A){
  H_A = 0
  P_A <- as.matrix(A/sum(A))
  for(i in seq(P_A)){
  H_A = H_A + ifelse(P_A[i]>0, -P_A[i]*log2(P_A[i]), 0)
  }
  return((H_A))
}

# funcion for comparing signal to a random walk, modified for non standardization to
# better compare with signal 1. Don't really know whether to do this or standardize
# the total signal
# n is number of samples, v is standard deviation, m is mean
walk <- function(n, sd, m){
  x = m # won't have exactly the some mean, but should be similar
  for(i in 2:n){
    x[i] <- x[i-1]+rnorm(1, mean = 0, sd = sd )# need a much lower sd for similar w sd
    if(x[i] < 1){x[i]=1} # need strictly positive for some of the later calculations
  }
  #x <- (x-mean(x))/var(x) # standardization
  return(x)
}
```

The output is the head of the final form I used for analysis. It shows the initial conditions of the simulation and informs the calls later. I didn't echo the code in this document, but you can refer to the file for more info

```
##   X.run.number. X.step. O2.starv.danger.threshold sugar.levels
## 1             1       0                       30            8
## 2             1       1                       30            8
## 3             1       2                       30            8
## 4             1       3                       30            8
## 5             1       4                       30            8
## 6             1       5                       30            8
##   energy.take.from.host low.catab.value high.metabolism number.of.pills
## 1                    -1              1            1.6              25
## 2                    -1              1            1.6              25
## 3                    -1              1            1.6              25
## 4                    -1              1            1.6              25
## 5                    -1              1            1.6              25
```

```
## 6                    -1               1            1.6              25
##   number.of.pacmen swap.rate sugar.regrowth.rate energy.needed.repro
## 1               11       100                0.05                  27
## 2               11       100                0.05                  27
## 3               11       100                0.05                  27
## 4               11       100                0.05                  27
## 5               11       100                0.05                  27
## 6               11       100                0.05                  27
##   low.metabolism count.pills.with..color...red.
## 1              1                              4
## 2              1                              5
## 3              1                              7
## 4              1                             11
## 5              1                             19
## 6              1                             35
##   count.pills.with..color...pink. count.pills
## 1                              21          25
## 2                              21          26
## 3                              21          28
## 4                              20          31
## 5                              20          39
## 6                              23          58
##   count.pacmen.with..color...blue. count.supers count.pacmen
## 1                               11            0           11
## 2                               11            0           11
## 3                               11            0           11
## 4                               12            0           12
## 5                               12            0           12
## 6                               12            0           12
##
## 1
## 2
## 3                                                                                         1, 1,
## 4                                                                             1, 1, 1, 1, 1, 1,
## 5                                                     1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 6 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
##
## 1
## 2
## 3
## 4
## 5                                                                       29, 1, 5, 14, 29, 9, 29,
## 6 13, 29, 29, 29, 1, 29, 14, 5, 13, 29, 29, 29, 13, 29, 29, 6, 2, 29, 29, 9, 29, 29, 2, 5, 9, 10, 29
##   X1....sum..pO2..of.patches...sum..max.pO2..of.patches.
## 1                                               0.5249320
## 2                                               0.4533815
## 3                                               0.3854929
## 4                                               0.3231241
## 5                                               0.2689042
## 6                                               0.2226440
##   X1...sum..psugar..of.patches...sum..max.psugar..of.patches diversity
## 1                                                  0.4981499        12
## 2                                                  0.5031983        12
## 3                                                  0.5079951        12
```

```
## 4                                                 0.5113429        12
## 5                                                 0.5174755        12
## 6                                                 0.5268062        12
##   sequence.tag.entropy mean.hamming healthMeasure turtles
## 1             3.228322            1      1.608188      36
## 2             3.228322            1      1.624486      37
## 3             3.228322            1      1.639972      39
## 4             3.228322            1      1.650779      43
## 5             3.228322            1      1.670577      51
## 6             3.228322            1      1.700700      70
```

Non linear Analysis

To get an idea of the phase space and invariant quantities for the whole space I'm going to string together total population data for the entire Dtrain. Not just use them as one at a time. This particular non linear analysis doesn't allow for multivariate use. So some of my other quantities don't quite make sense.

Is my simulated data defferent from random noise. Are there underlying patterns? Is it regular?

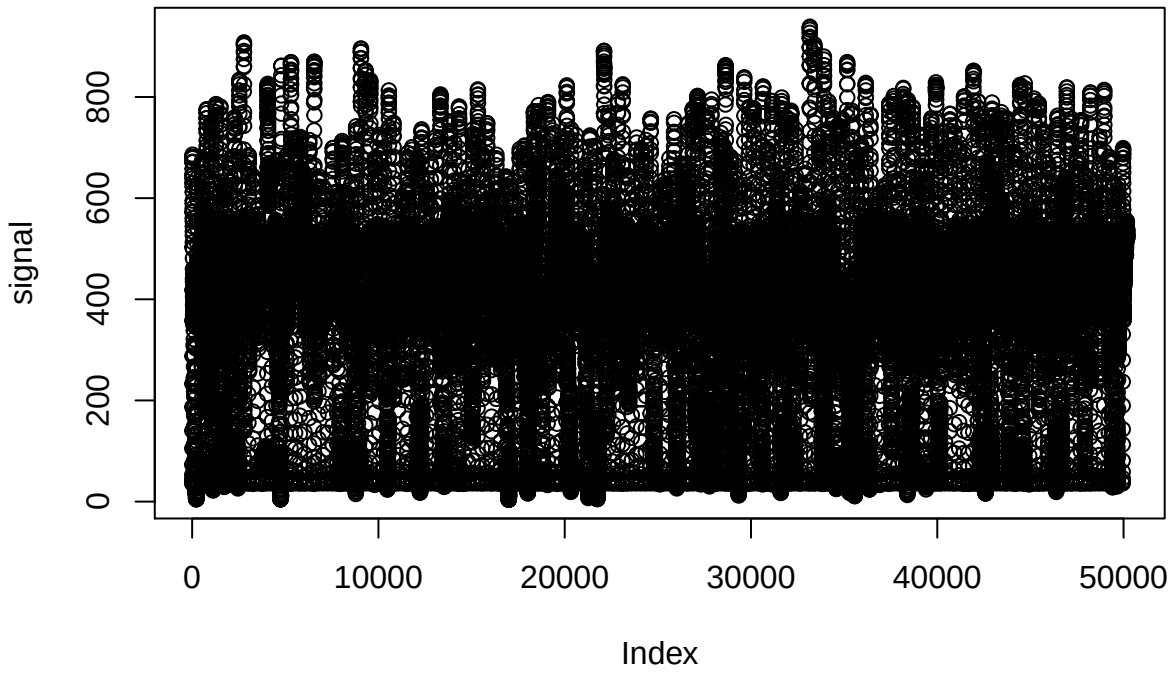
```
# framework from exploring_EntropyLab1.Rmd
signal <- Dtrain$turtles

# total variance
total.var <- var(signal) # total variance is ~27251
cat("variance of aggregate population signal is ", total.var, "\n")
```

```
## variance of aggregate population signal is  27251.83
```

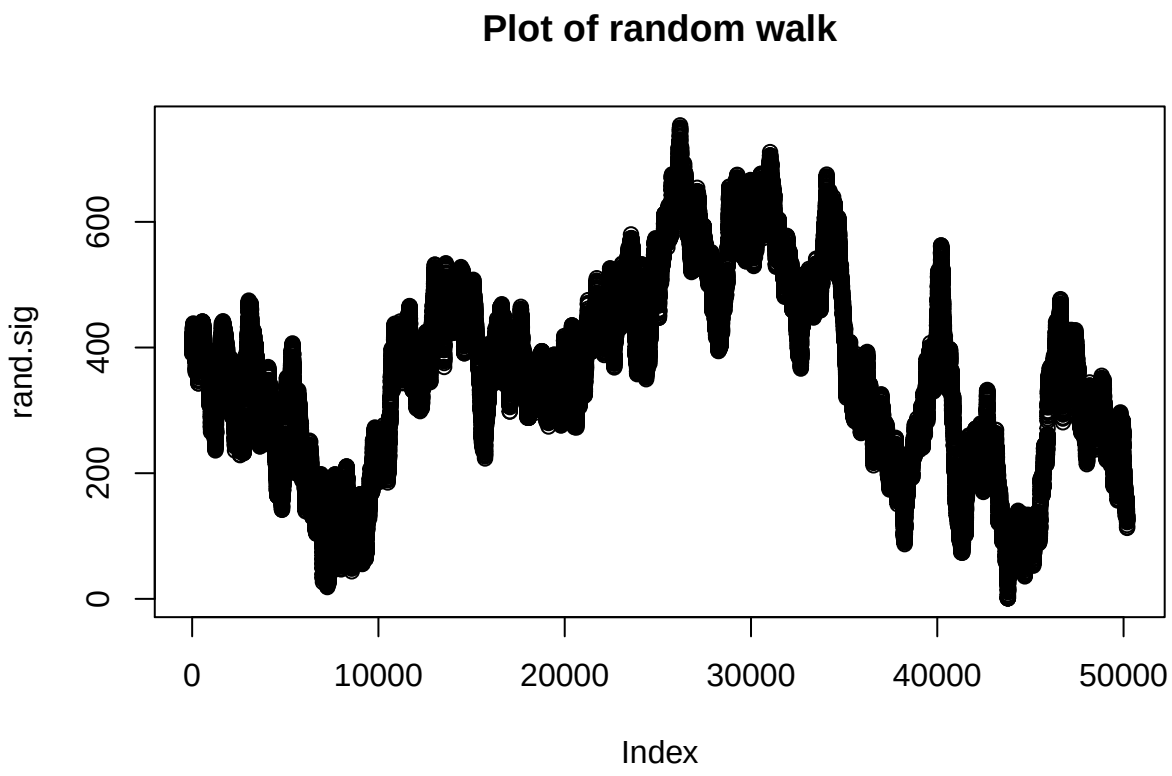
```
total.sd <- sqrt(total.var)
total.mean <- mean(signal) # ~394
cat("mean of aggregate population signal is ", total.mean, "\n")
```

```
## mean of aggregate population signal is  394.9588
```

```
# variance by run

run.var = 0

for(i in seq_len(max(Dtrain$X.run.number.))){
  run.var[i] = var(Dtrain$turtles[Dtrain$X.run.number.==i])
}



plot(signal) # rough idea of phase space if vars are total turtles and time steps
```




```
var.plt = qplot(run.var,xlab = 'run variance', main = 'Histogram of Variance by Run')

# saving the variance histogram to ./images as png --> useful for latex
# png(filename = "./images/var.by.run.png")
# var.plt
# dev.off()

# ## Creating a random walk with similar total variance to compare the signals
# sd <- total.sd/40 # need lower
#
# while(abs(rand.sig.mean - total.mean) > 150 || abs(rand.sig.var - total.var) > 5000){
#   rand.sig <- walk(n = length(signal), sd = sd, m = total.mean)
#   rand.sig.mean <- mean(rand.sig)
#   rand.sig.var <- var(rand.sig)
#   print(abs(rand.sig.mean - total.mean))
#   print(abs(rand.sig.var - total.var))
# }
#
# # rand.sig <- walk(n = length(signal), sd = sd, m = total.mean)
#  plot(rand.sig)
#
# mean(rand.sig)
# var(rand.sig)

#
#
# # So we always use the same test.sig
# saveRDS(object = rand.sig, file = "./data/rand.sig")

rand.sig = readRDS('./data/rand.sig')
plot(rand.sig, main="Plot of random walk")
```




```
rand.sig.mean <- mean(rand.sig)
rand.sig.var <- var(rand.sig)
cat("variance of random walk is ", rand.sig.var, "\n")
```

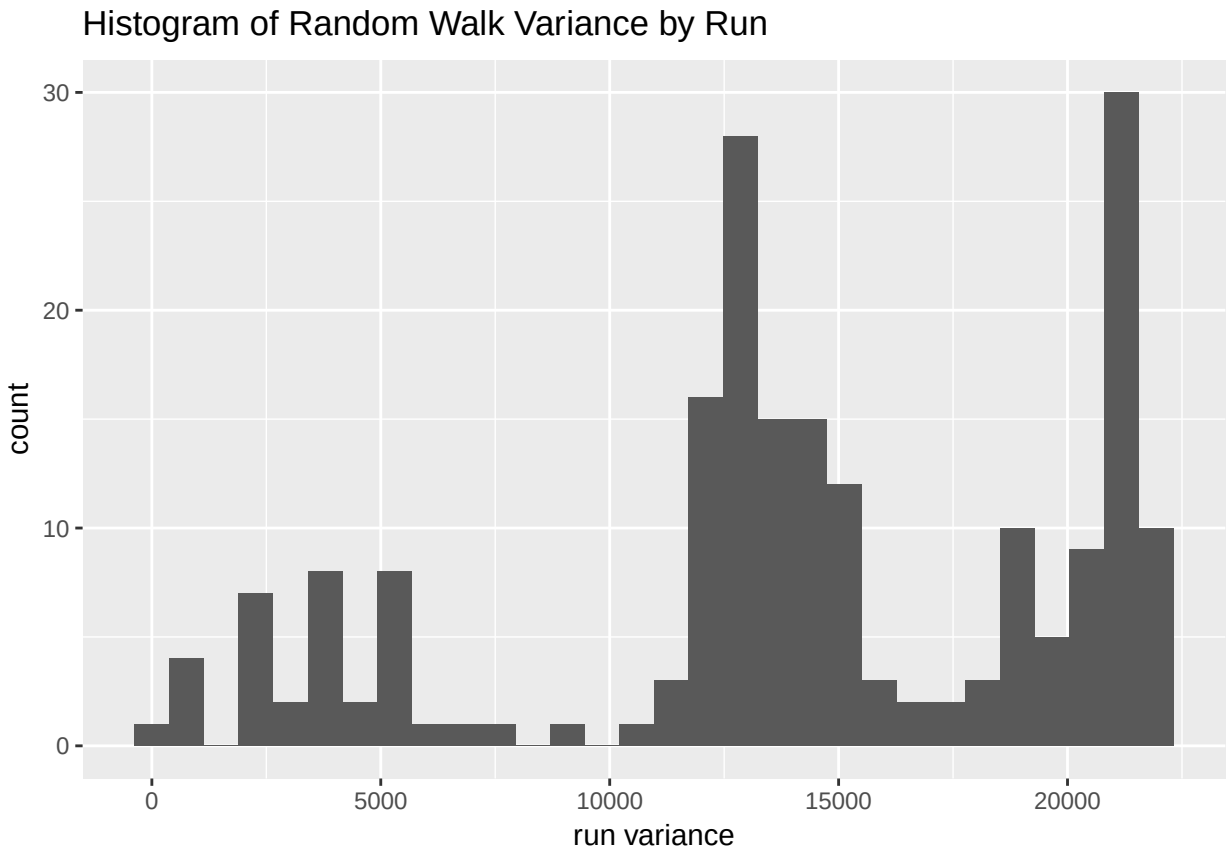
```
## variance of random walk is  22618.22
```

```
cat("Mean of random walk is ", rand.sig.mean, "\n")
```

```
## Mean of random walk is  357.3836
```

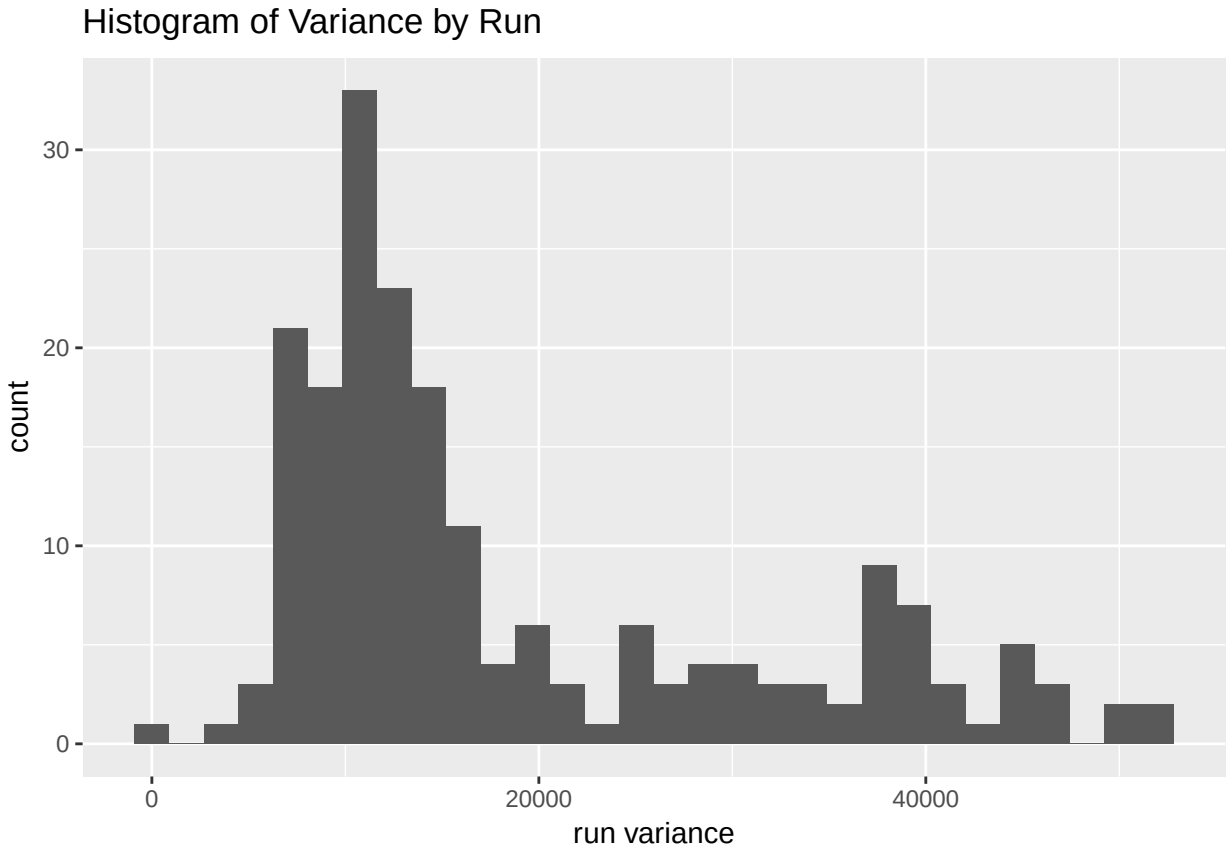
```
rand.run.var = 0


for(i in seq_len(max(Dtrain$X.run.number.))){
  rand.run.var[i] = var(rand.sig[i*(1:201)])
}

rand.var.plt = qplot(rand.run.var,xlab = 'run variance', main = 'Histogram of Random Walk Variance by Ru



# png(filename = "./images/rand.var.by.run.png")
# rand.var.plt
# dev.off()

rand.var.plt
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
var.plt
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# the variance looks normal for string of runs
```

For a real phase space, I should just plot vs time step 1 to 201 I think I had to run the random walk a few times to get one that is a random version with reasonalbe values for the number of turtles as well as similar overall variance and mean

The average variance per run for the signal is 2717, but the total variance for all the signals is 27251.

The random walk has a much lower variance, I think I'm going to try to get a better version

I ran the random walk until I got something strictly positive with a similar mean and variance

For the population signal, most runs had around a 15,000 variance, but there are also quite a few with around 38,000. I think I can see that there are different common behaivor outcomes. I can see this when just looking at simulations. One reasonable way to classify an outcome may be variance. I think there are some steady states with different species and the same variance though.

Most runs

?? what does a p value mean in this case. I could add confidence bars with a per run ?? I know much more than the total population. How can I use this?? Can I separate deterministic from stocastic. SHould I model with blocks / sets.

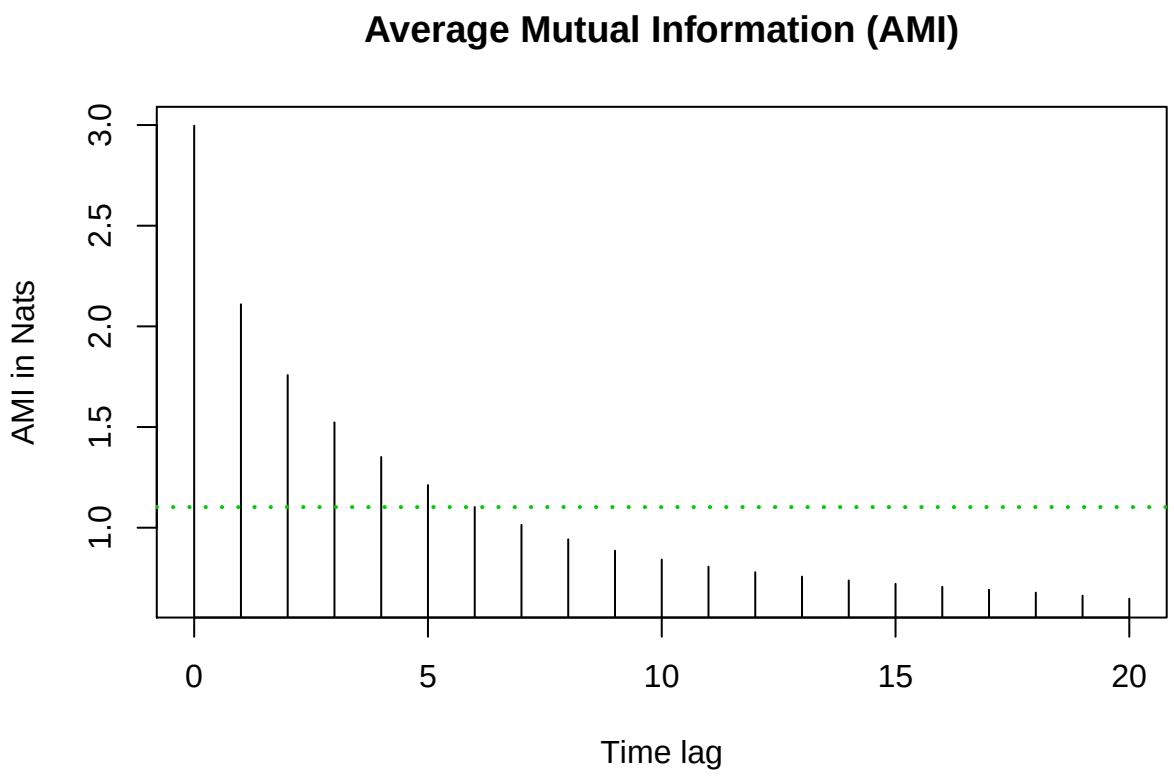
estimating the time lag τ

We use the first minimum of the mutual information to find an estimate of the appropriate τ

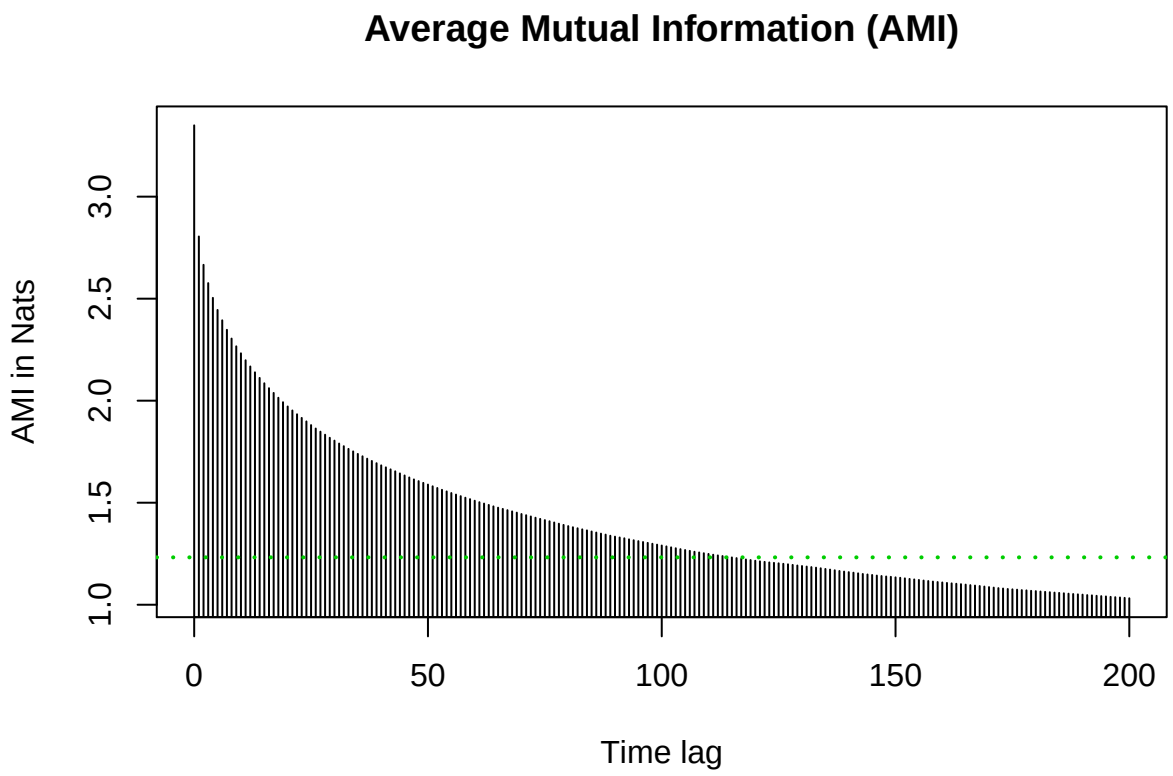
$$I(\tau) = \sum P(x(k), x(k+\tau)) \ln \frac{P(x(k), x(k+\tau))}{P(x(k)) \cdot P(x(k+\tau))}$$

This is the information difference between the data and the independence model for a given tau. When this is minimized our data is decoupled. The autocorrelation function could also have been used

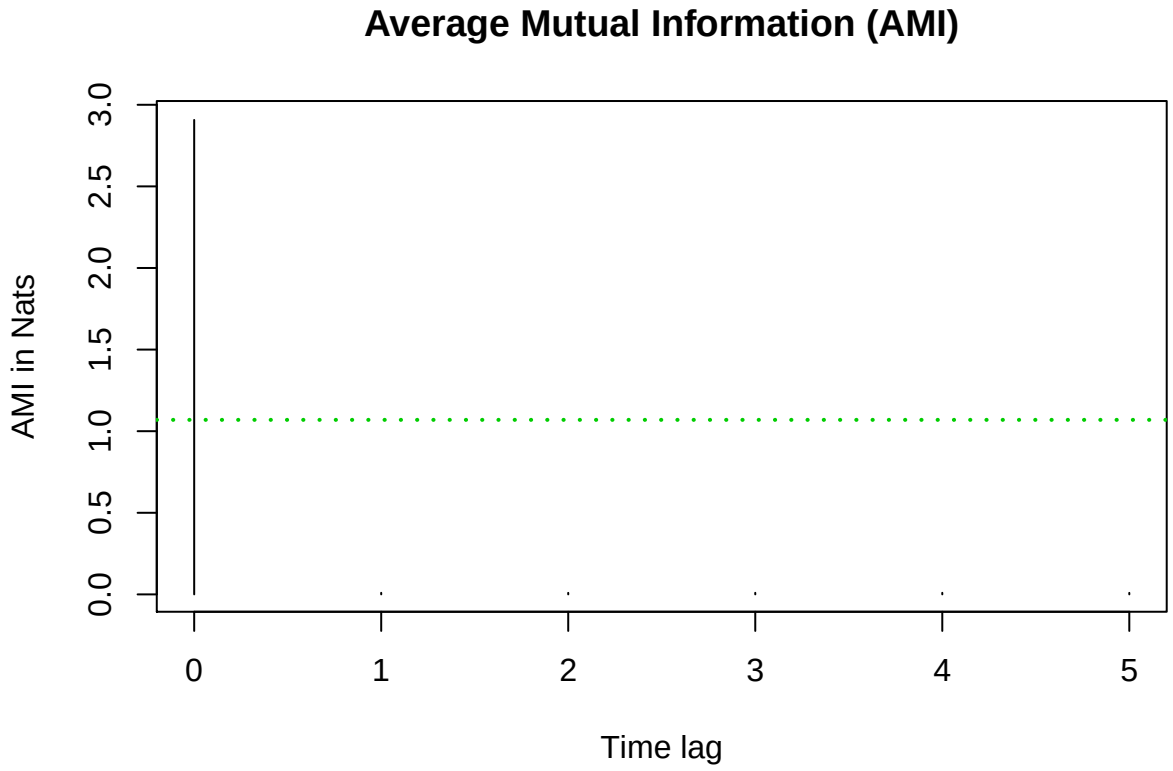
```
# using the package nonlinearTseries
lag.max <- 20
sig.tau.ami = timeLag(signal, technique = "ami",
                  lag.max =lag.max, do.plot = T)
```




```
# compare to random walk
# The dependence of using the last thing apparently creates depedencies that take
# a long time to get out. Needed tau of 115 for less than 1 nat ???
rand.sig.tau.ami = timeLag(rand.sig, technique = "ami",
                  lag.max =200, do.plot = T)
```




```
#compare to random noise random noise is zero at 1 which is expected. It's independent.
rad.tau <- timeLag(rnorm(50200), technique = "ami",
                   lag.max =5, do.plot = T)
```

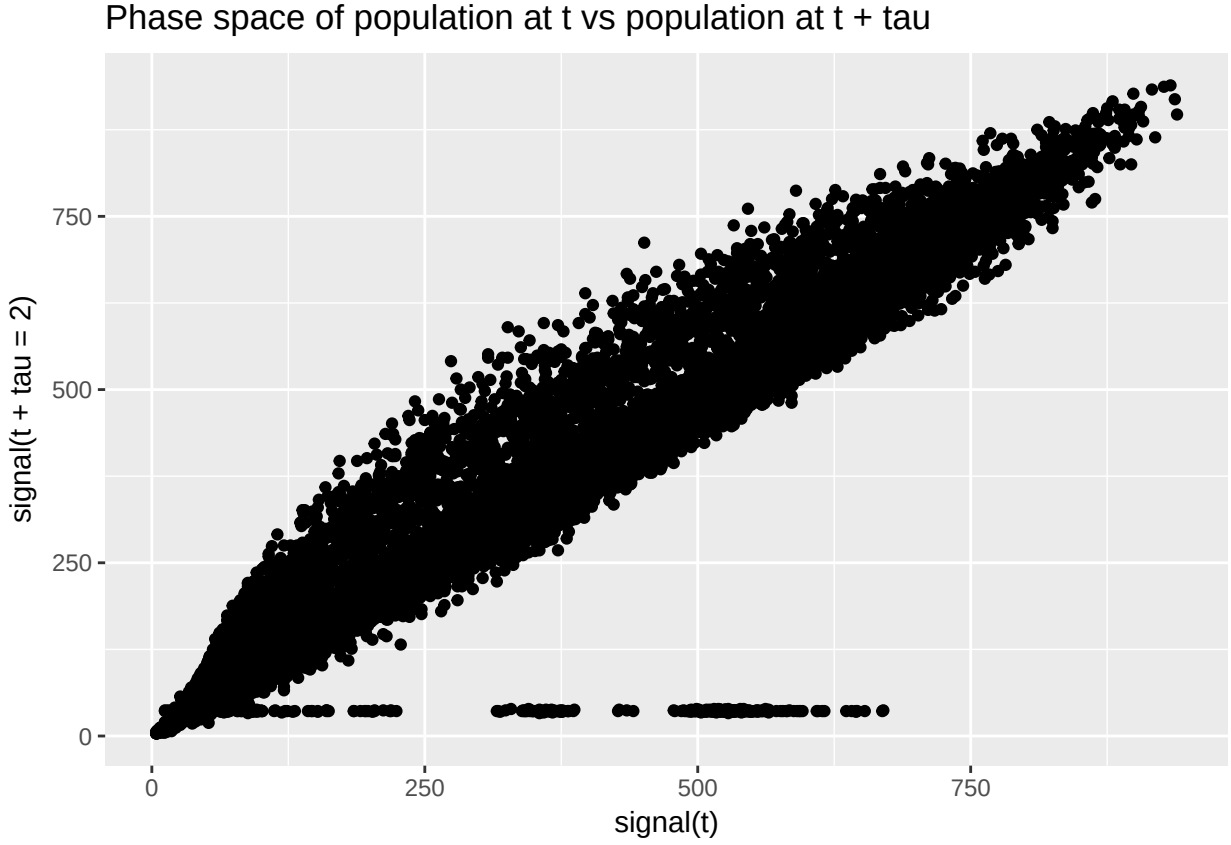


```
sprintf("The time lag calculated using mutual information is %s for the simulated population signal. %s
```

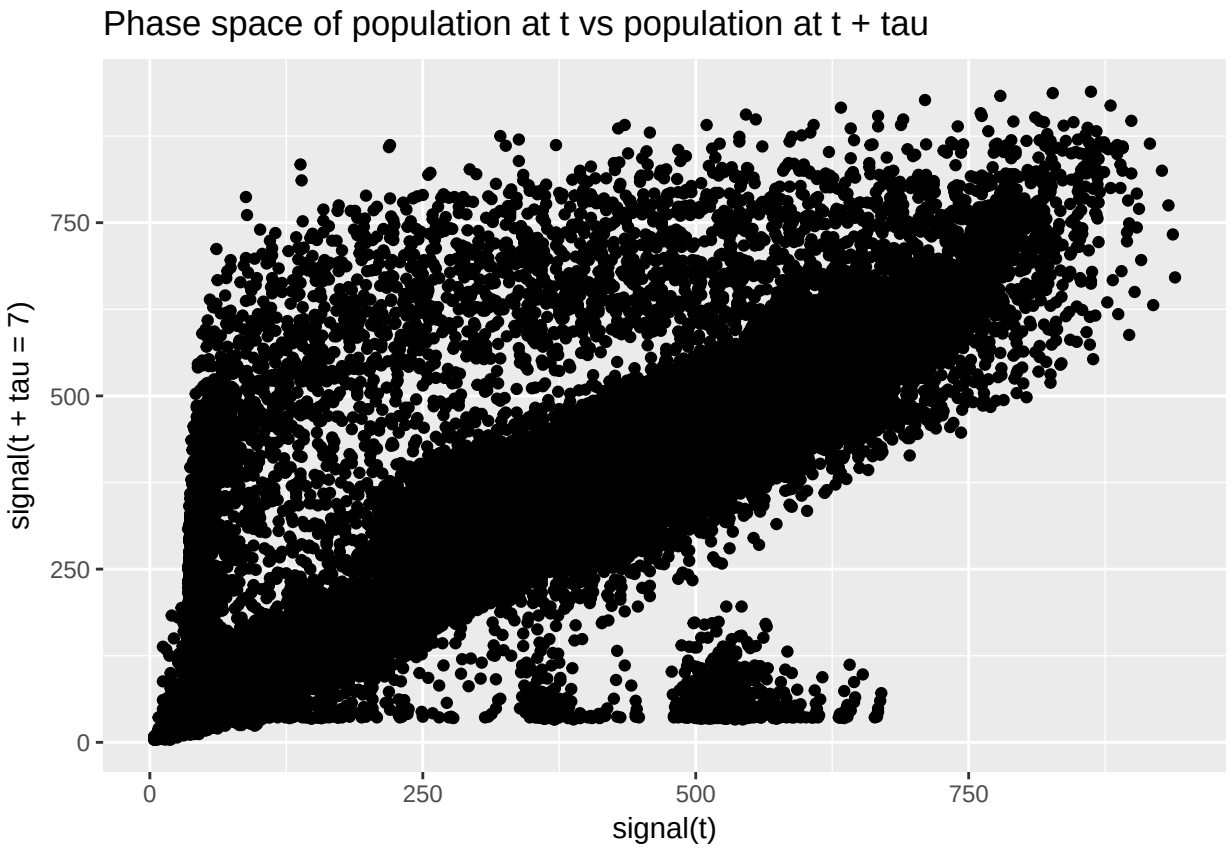
```
## [1] "The time lag calculated using mutual information is 7 for the simulated population signal. 115 i
```

I expected a non correlated signal from the random walk. The Average Mutual Information criteria would indicate otherwise. ## Now for finding invariant measures Using a 2d projection with τ for a semblence of "phase space"

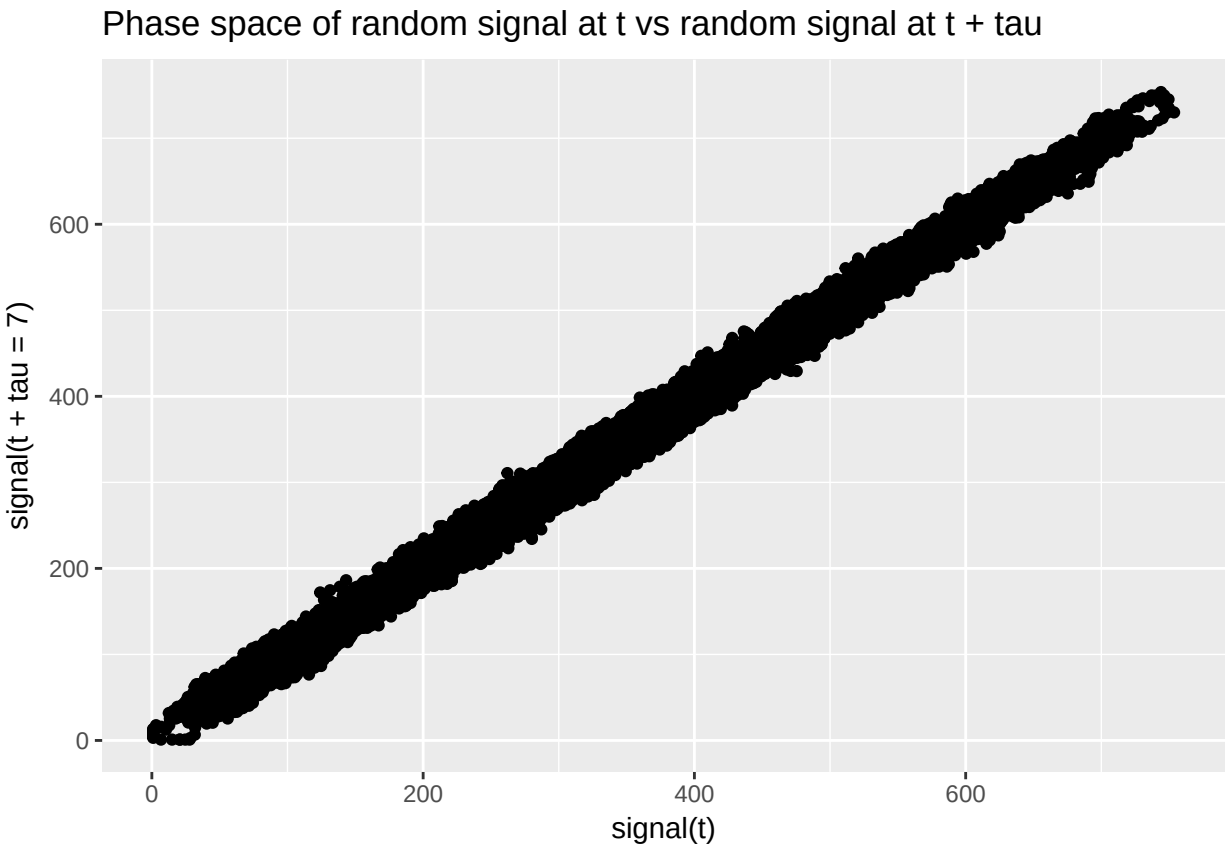
```
# I'd like to plot F(t) and F(t + tau)
sig.at.tau <- 0
tau <- 2 # before interesting behavior
for(i in 1:(length(signal)-tau)){sig.at.tau[i] <- signal[i+2]}
qplot(x = signal[1:length(sig.at.tau)],y = sig.at.tau, xlab = 'signal(t)', ylab = sprintf("signal(t + ta
```




```
tau <- sig.tau.ami # find out how to format label to include number
sig.at.tau <- 0
for(i in 1:(length(signal)-tau)){sig.at.tau[i] <- signal[i+tau]}
# png(filename = './images/sigTau.png')
qplot(x = signal[1:length(sig.at.tau)],y = sig.at.tau, xlab = 'signal(t)', ylab = sprintf("signal(t + ta
```




```
# dev.off()
rand.sig.at.tau <- 0
for(i in 1:(length(rand.sig)-tau)){rand.sig.at.tau[i] <- rand.sig[i+tau]}
# png(filename = './images/randSigTau.png')
qplot(x = rand.sig[1:length(sig.at.tau)],y = rand.sig.at.tau, xlab = 'signal(t)', ylab = sprintf("signa
```




```
# dev.off()

# The population signal has patterns after tau equal to 7
```

The random phase space shows what we would expect. The next time is simply some random amount from the mean. The line is formed because the next time step in a random walk is dependent on the last.

At a timestep of 2, the space looks like the random walk aside from the outcome case where the population stays low. We can see that the space forms an elispsiod likely representing the carrying capacity of the environment with fluctuating species.

When we look at the population signal with a time lag of 7 we get some intesting behavior. The empty middle space show a jump indicatings times of population explosion and collapse

I used the tau of 7 to calculate the correlation dimension using Cao's method ## using Cao's algorithm to find embedding dim "Practical method for determining the minimum embedding dimension of a scalar time series"

$$y(d) = (x_i, x_{i+\tau}, ..., x_{i+(d-1)\tau})$$

```
# these take a few minutes to run, saved plots using export
emb.dim = 11 #estimateEmbeddingDim(signal, time.lag = sig.tau.ami,
                                   # max.embedding.dim = 20) # takes forever

cat('The embedding dimension found for the population signal is ', emb.dim, '\n')
```

```
## The embedding dimension found for the population signal is  11
```

```
rand.emb.dim = 7#estimateEmbeddingDim(rand.sig, time.lag = rand.sig.tau.ami,
                                   # max.embedding.dim = 50) # it's 7 --> takes forever
```

```
cat('The embedding dimension found for the random walk is ', rand.emb.dim, '\n')
```

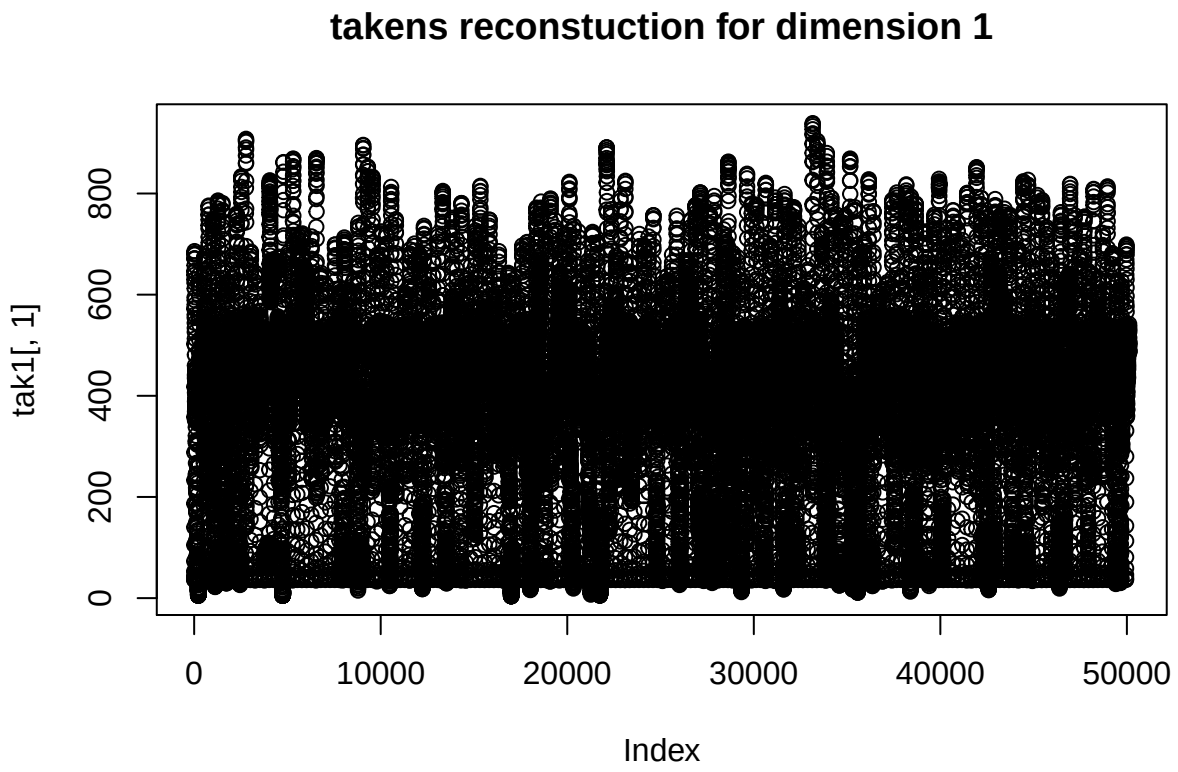
```
## The embedding dimension found for the random walk is  7
```

$E1(d)$ stops changing when $d \geq$ *embedding dimension* $E2(d)$ is 1 for stocastic systems and 0 (just not equal to 1 in paper??) for deterministic systems. This is a result of the embedding dimension capturing the underlying couplings of the nonlinear function if they exist. $E2(d)$ is a function of the differences between the ratio of $d+1$ vectors to d vectors.

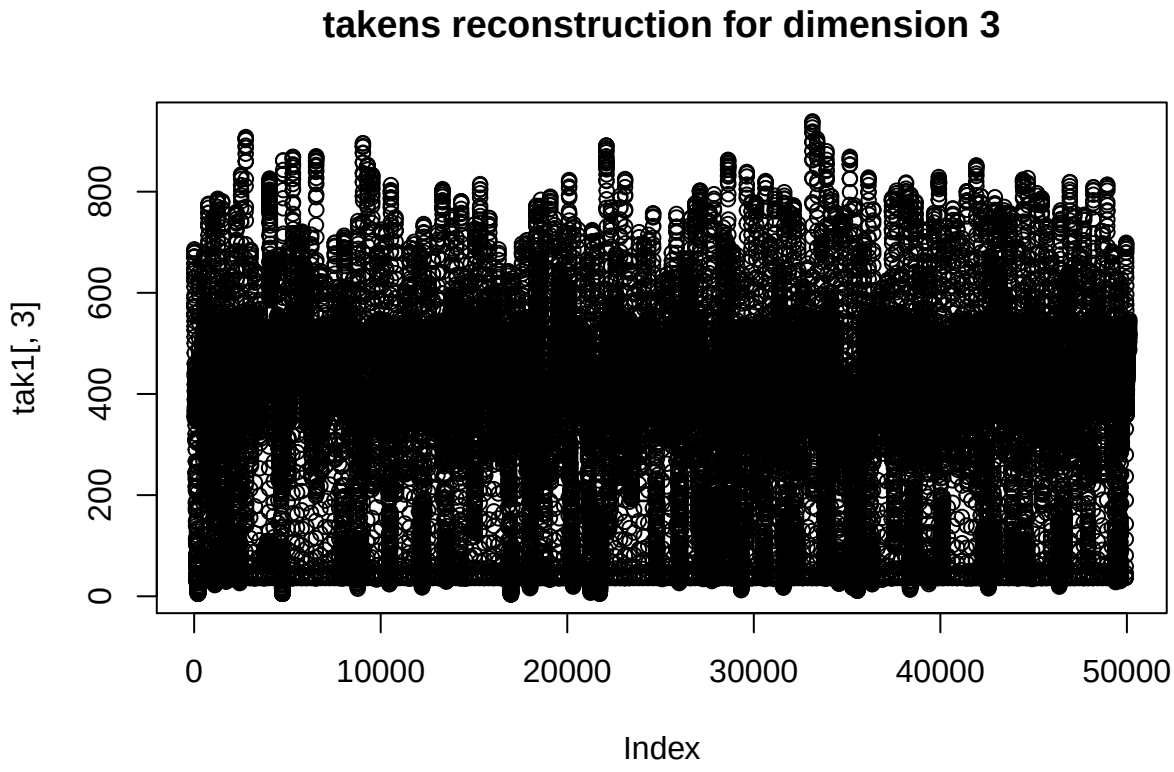
The function found an embedding dimension of 11 for the population and 7 for the random walk

On to building the takens to attempt to reconstruct the phase space

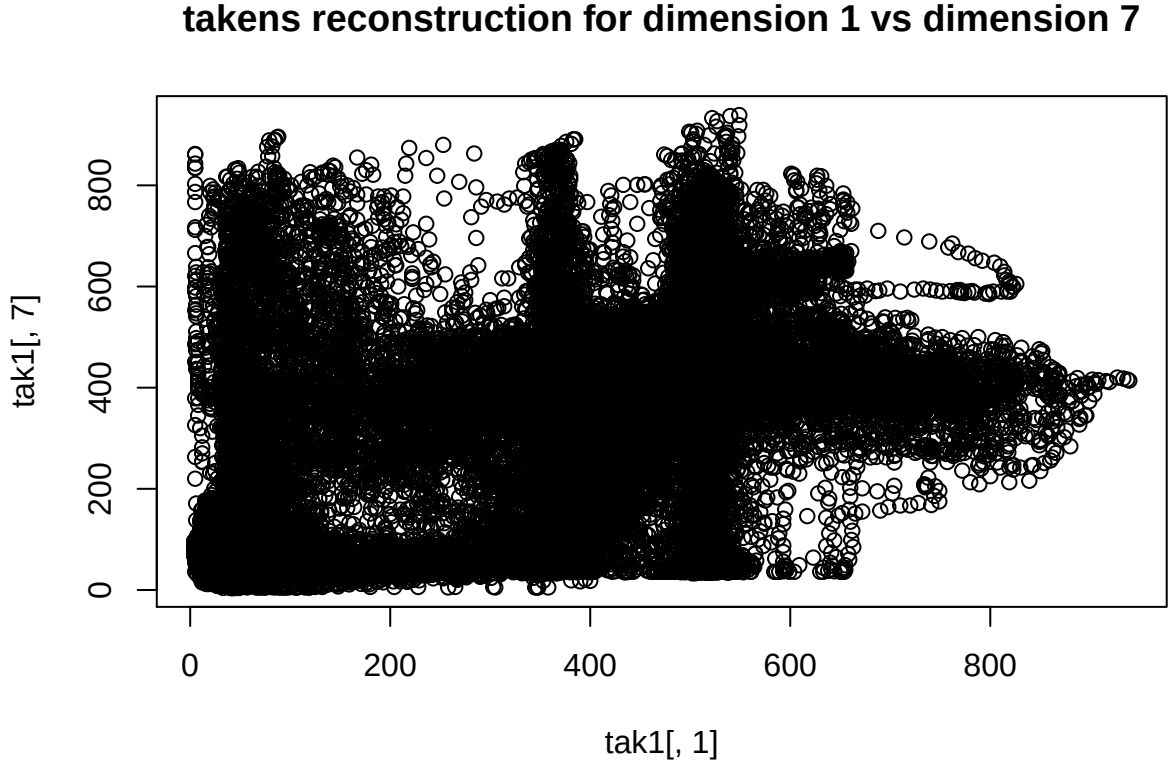
```
tak1 = buildTakens(signal,embedding.dim = emb.dim, time.lag = sig.tau.ami)
plot(tak1[,1], main = "takens reconstuction for dimension 1")
```




```
plot(tak1[,3], main = "takens reconstruction for dimension 3")
```




```
plot(tak1[,1], tak1[,7], main = "takens reconstruction for dimension 1 vs dimension 7")
```




```
tak2 = buildTakens(rand.sig,embedding.dim = rand.emb.dim, time.lag = sig.tau.ami)
#plot(tak2[,1])
```

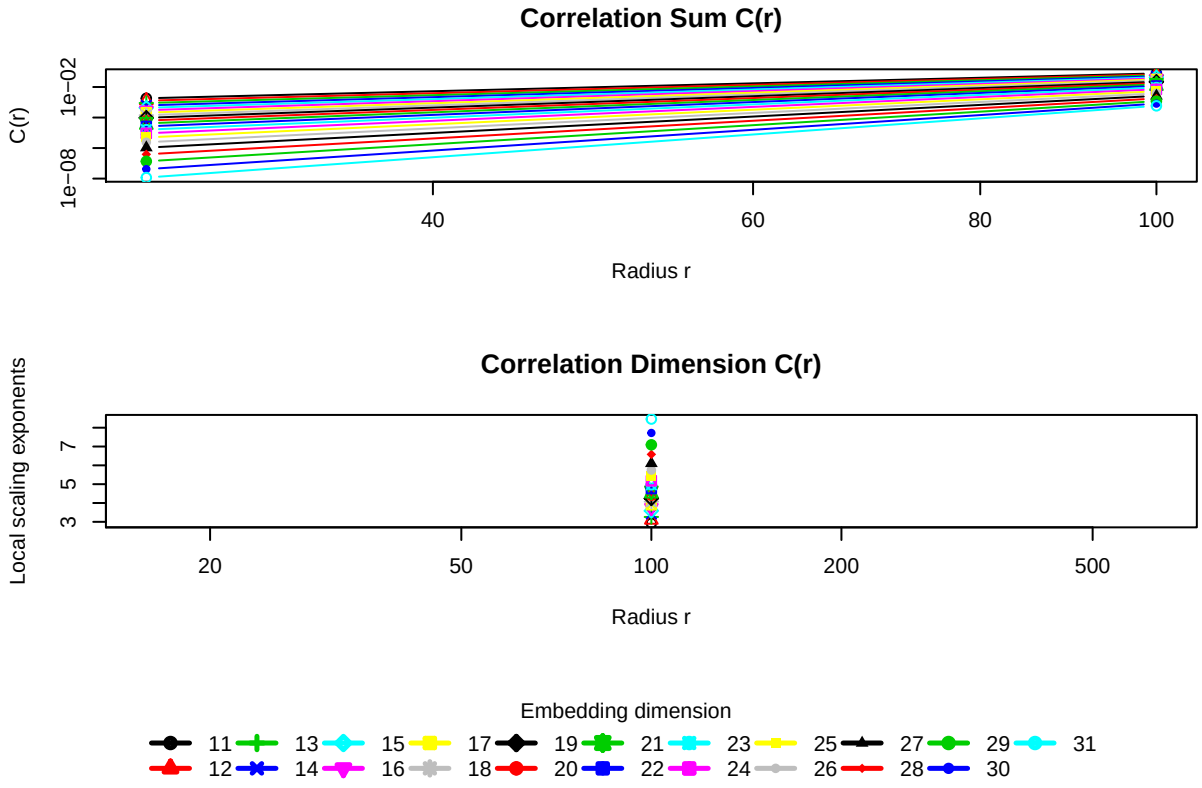
Lyapunov, Generalized Correlation dimension, Sample entropy

- heavy computations
  1. scaling behavior over time –> correlation dimension
  2. dynamical evolution over time –> Lyapunov exponent
- Find linear region for statistic
- the statistics should be invarient given a high enough embedding dimension
  - Look for linear behavior of a region in plots to recognize this
- estimation of invariant is found through a linear regression of previous region

Correlation Dimension

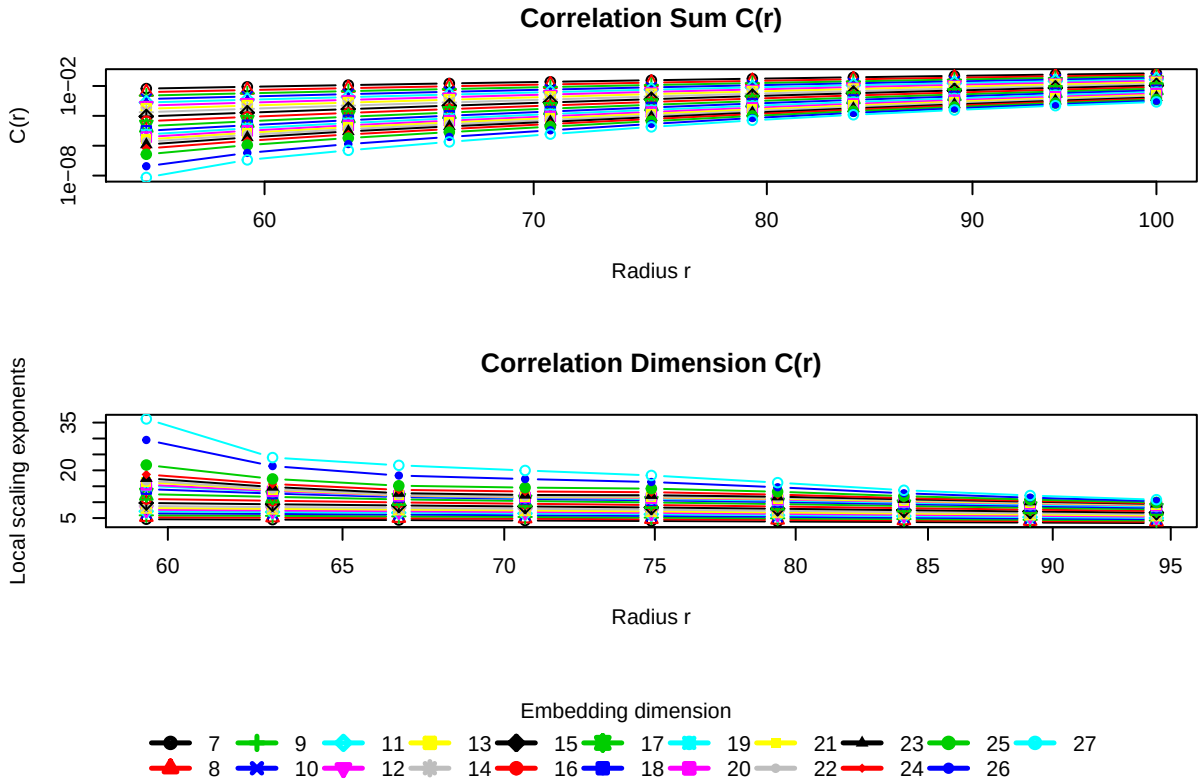
The correlation dimension measures the fractal dimension in phase space of a dynamical system. * Need to find an invarient region. * typically start with embedding region calculated before and check for like 5 more. * We vary our radius or the ϵ around the "matching" criteria to potentially compensate for noise. Shouldn't really matter as these systems are essentially noise

```
cd1 = corrDim(signal,
              min.embedding.dim = emb.dim,
              max.embedding.dim = emb.dim + 20,
              time.lag = sig.tau.ami,
              min.radius = 0.001, max.radius = 100,
              n.points.radius = 10,
              theiler.window = 1000,
              do.plot=T)
```

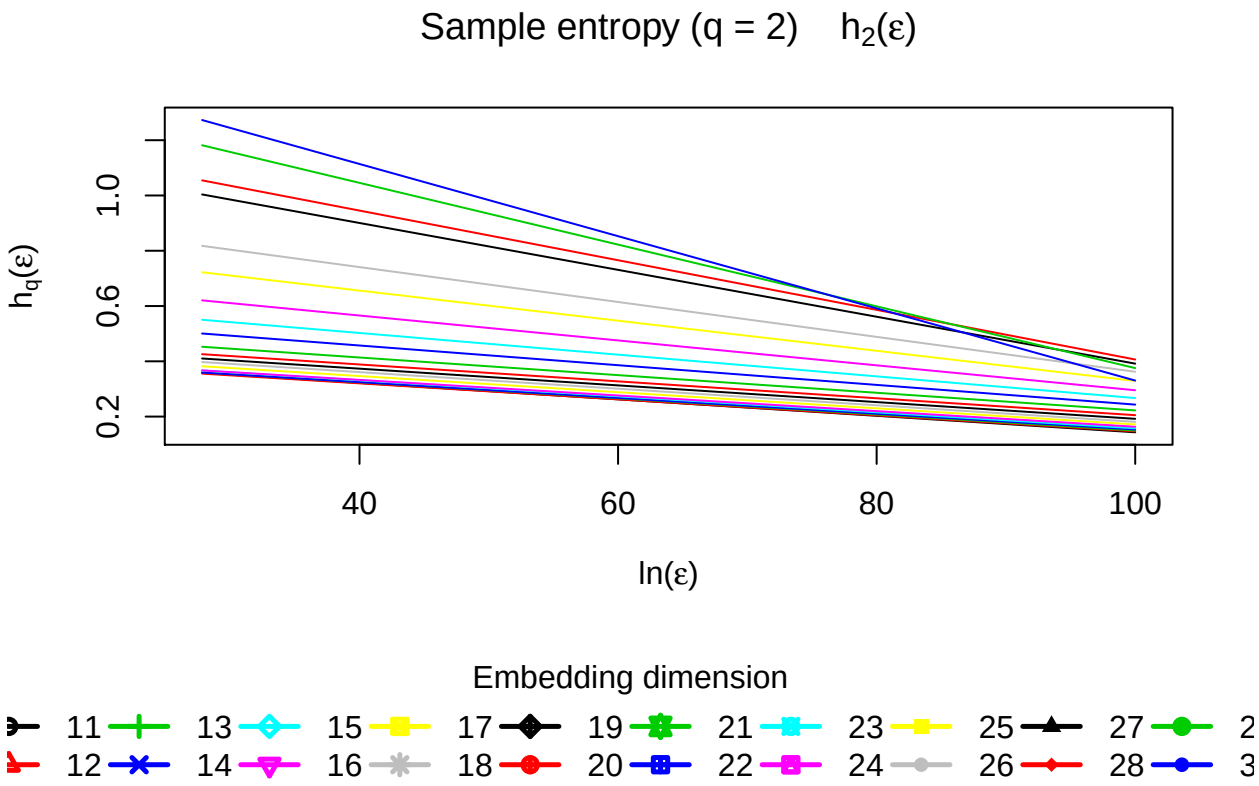



```
# error
cd2 = corrDim(rand.sig,
```

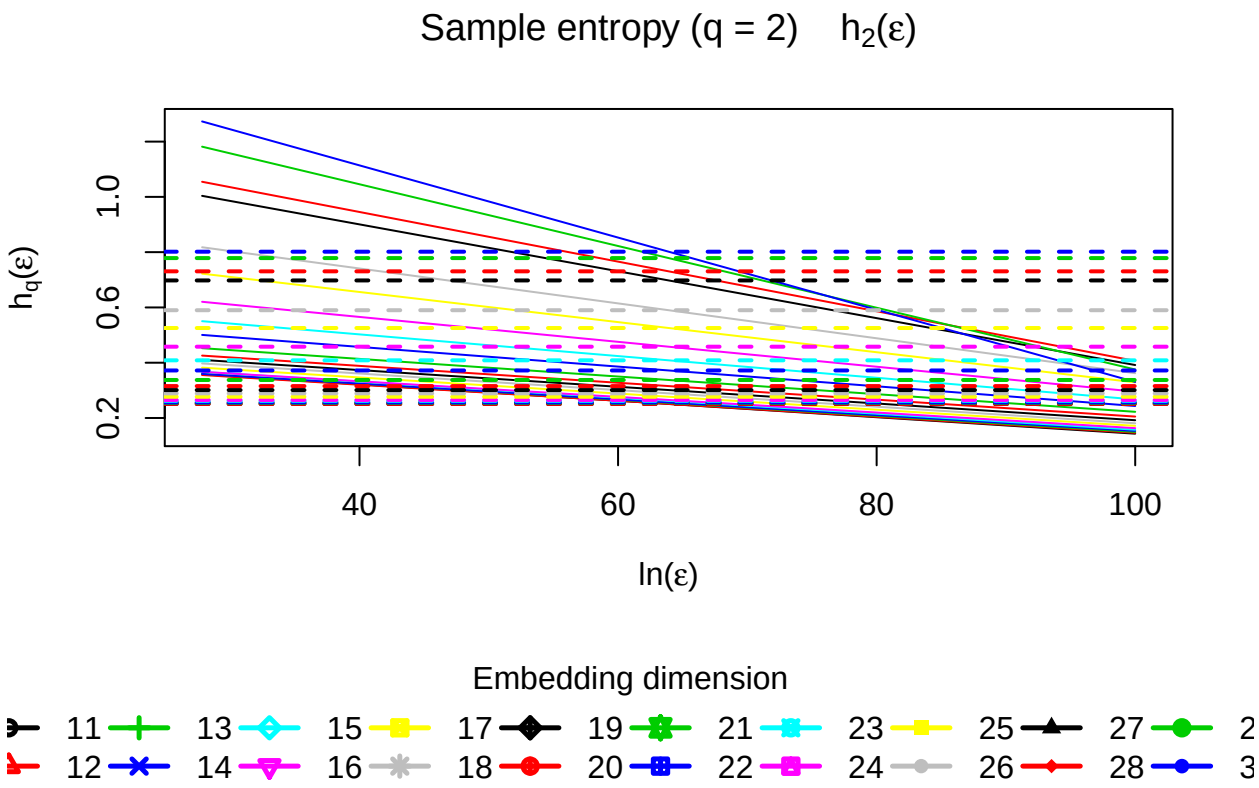
```
            min.embedding.dim = rand.emb.dim,
            max.embedding.dim = rand.emb.dim + 20,
            time.lag = rand.sig.tau.ami,
            min.radius = 0.001, max.radius = 100,
            n.points.radius = 200,
            theiler.window = 1000,
            do.plot=T)
```




```
# sample entropy ## They are exactly the same ?????
se1 = sampleEntropy(cd1, do.plot = T)
```



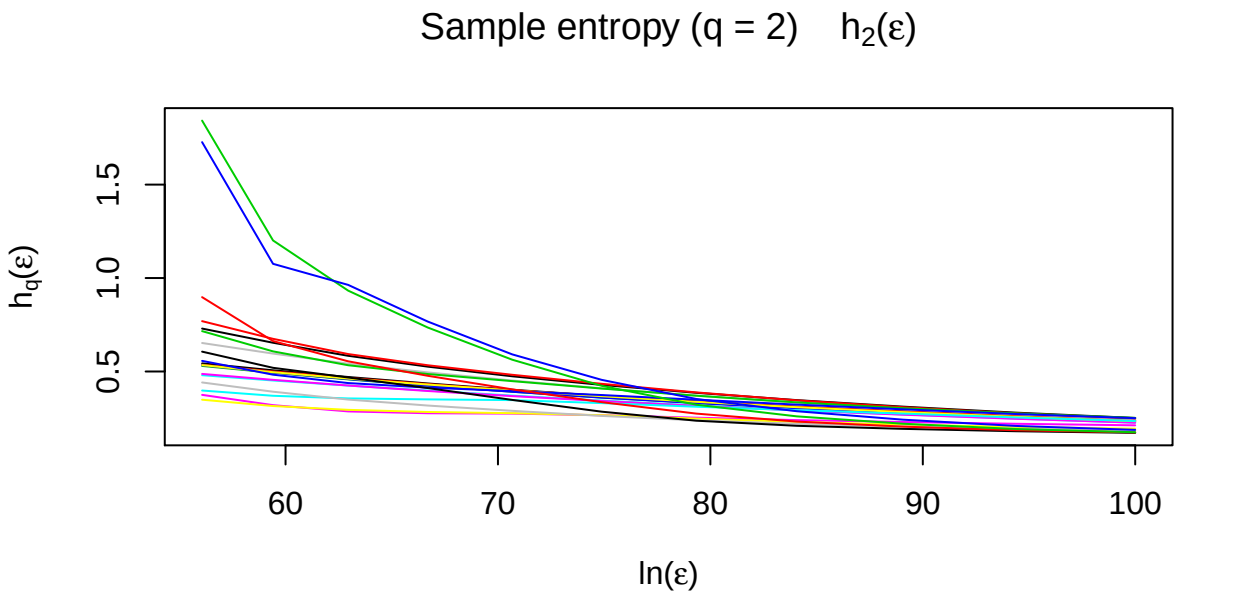

```
se.est1 = estimate(se1, do.plot = T,
                   regression.range = c(0,100))
```



```
cat("Sample entropy estimate for the total signal is: ", mean(se.est1), "\n") #0.4211443
```

```
## Sample entropy estimate for the total signal is:  0.4211443
```

```
se2 = sampleEntropy(cd2, do.plot = T)
```



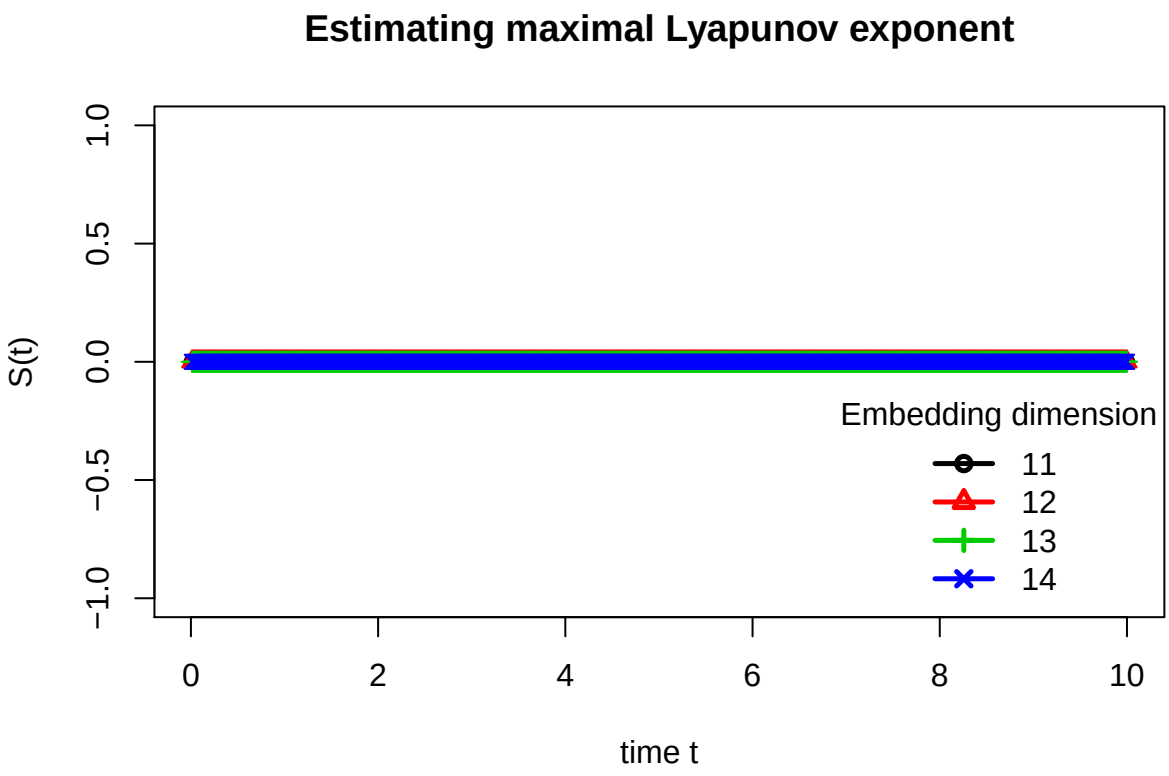

```
se.est2 = estimate(se2, do.plot = F,
                   regression.range = c(0,100))
cat("Sample entropy estimate for walk: ", mean(se.est2), "\n") #0.4211443
```

```
## Sample entropy estimate for walk:  0.390824
```

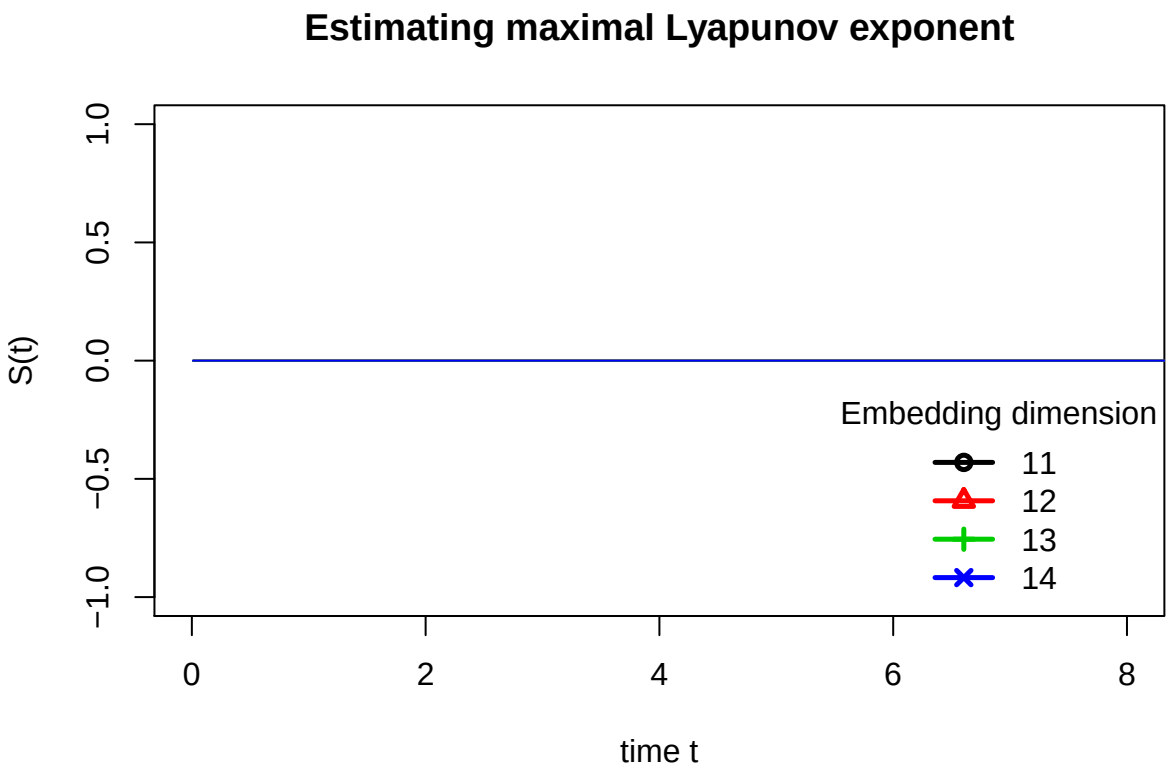
```
## saving env vars because I want to run the other file
#save(list = ls(),file = "./data/environment.Rdata")
#load(file = "./data/environment.Rdata") # 5/1/17 @3:30pm
```

Max Lyapunov

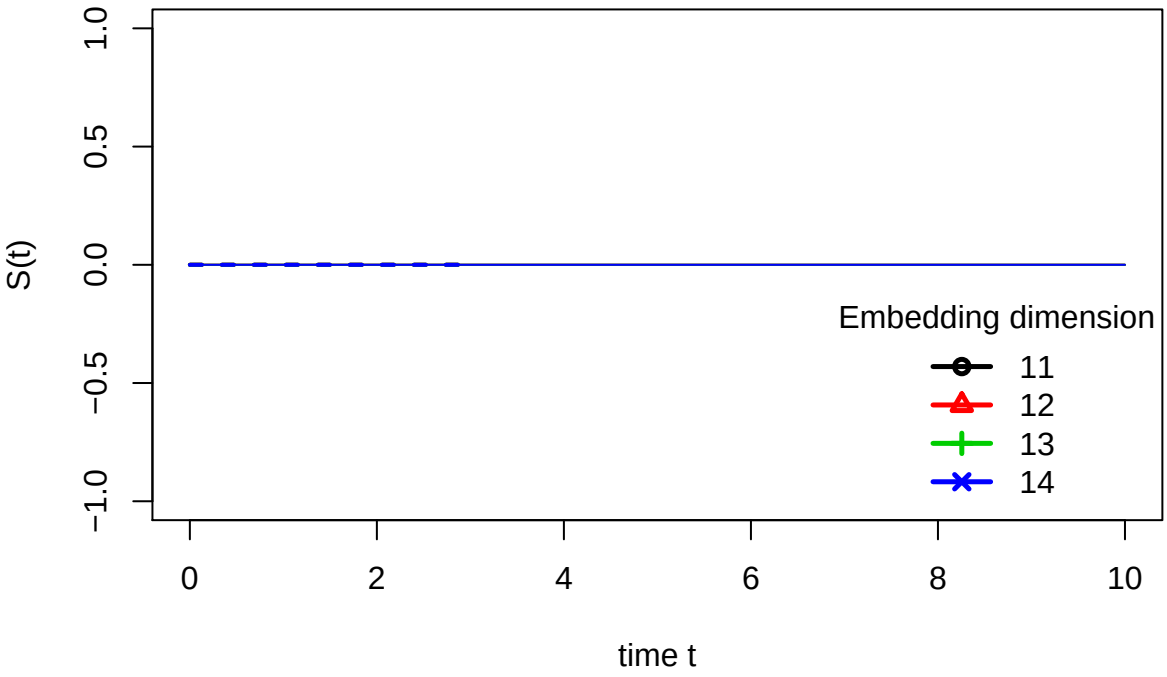
```
# get the sampling period of the lorenz simulation
# computing the differences of time (all differences should be equal)
# always get a zero Lyapunov
ml1 = maxLyapunov(signal,
                  sampling.period=0.01,
                  min.embedding.dim = emb.dim,
                  max.embedding.dim = emb.dim + 3,
                  time.lag = sig.tau.ami,
                  radius=1,
                  max.time.steps=1000,
                  do.plot=T)
```




```
plot(ml1,type="l", xlim = c(0,8))
```




```
ml.est = estimate(ml1, regression.range = c(0,3),
                  do.plot = T,type="l")
```




```
cat("max lyapunov =", ml.est,"\n")
```

```
## max lyapunov = 0
```

```
# table of findings for paper
non.lin.stats <- data.frame(
  'random walk' = c(rand.sig.tau.ami,
                    rand.emb.dim,
                    round(mean(se.est2),4)),
  'population signal' = c(sig.tau.ami,
                          emb.dim,
                          round(mean(se.est1),4)),
  row.names = (c('tau',
                 'embedding dimension',
                 'sample entropy'))
  )
# png('./images/nonLinTable')
# tbl <- tableGrob(non.lin.stats)
# grid.arrange(tbl)
# dev.off()

# print(xtable(non.lin.stats))

# here are a couple of ways to do tables in latex for future use
# #Plot your table with table Grob in the library(gridExtra)
#    ss <- tableGrob(x)
#
#    #Make a scatterplot of your data
#    k <- ggplot(x,aes(x=x$"Value 1",y=x$"Value 2")) +
#    geom_point()
```

```
#
#    #Arrange them as you want with grid.arrange
#    grid.arrange(k,ss)

# to save as png use png()
# library(gridExtra)
# pdf("mypdf.pdf", height=6, width=4)
# grid.table(x)
# dev.off()
```

I didn't get the kind of response I wanted. There is slightly more uncertainty in the population dynamics than there is in the random walk. It would take a greater channel capacity to encode the information in the population signal. From this analysis, I don't think the simulation has an underlying deterministic chaotic dynamics. I'm really unsure of the accuracy of these results.

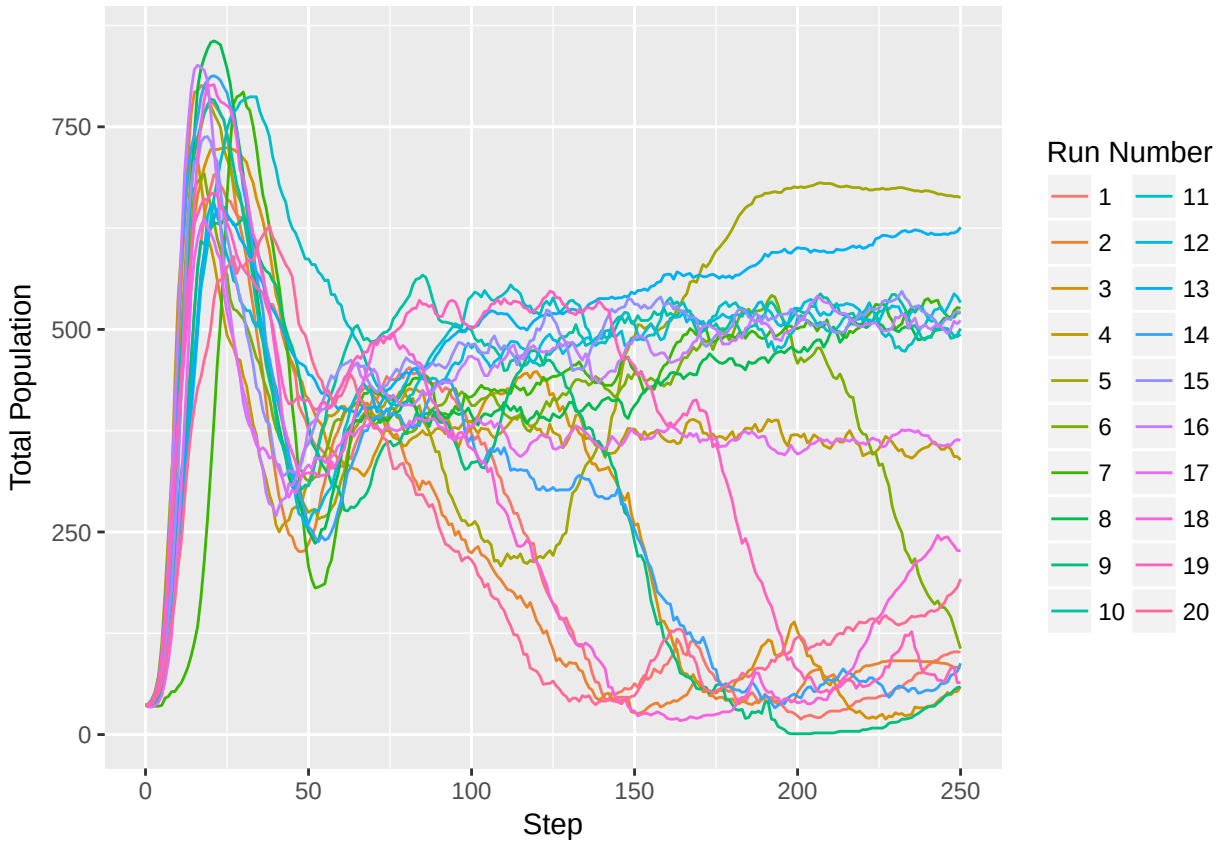
From: Dong, M. A Tutorial on Nonlinear Time-Series Data Mining in Engineering Asset Health and Reliability Prediction: Concepts, Models, and Algorithms. Math. Probl. Eng. 2010, (2010).

They use techniques more related to NLP sequences to model the data. Markov chains up to RNN's. I wonder if using RNN's to find any regularity would work better as long as you were only applying them in a specific setting. I have a feeling the entropy measures work better in general.

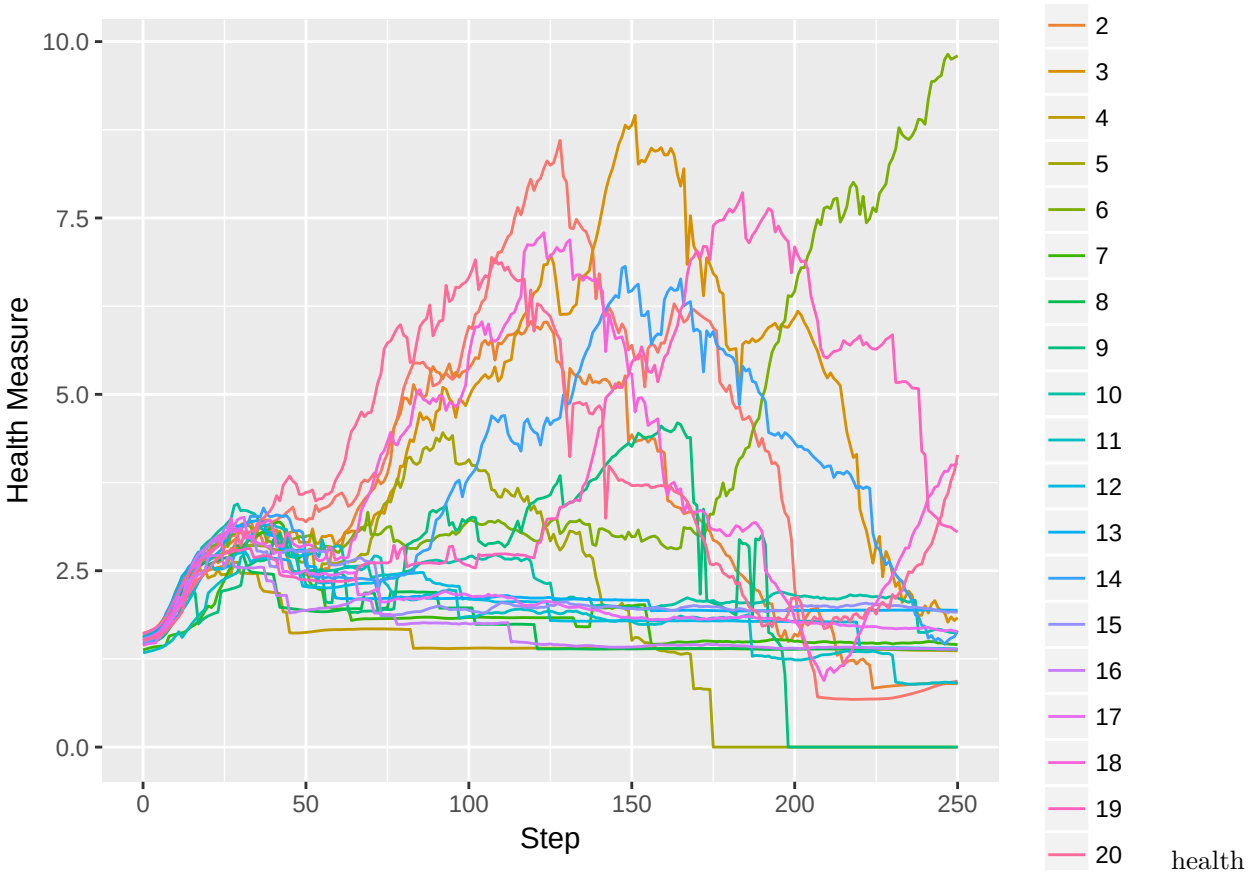
Data analysis

1. Focusing on test because the 20 runs seem enough to illuminate the overall behavior.
2. The population matrix is 20 rows by 252 columns representing 20 runs of 251 timesteps with a rowid column

```
# matrix of run x population with the first var the MultiScale entropy
DtestMSEpop <- matrix(Dtest$count.pills+Dtest$count.pacmen,nrow = max(Dtest$X.run.number.))
df <- as.data.frame(DtestMSEpop)
df$rowid <- 1:max(Dtest$X.run.number.)
df <- cbind(df[,c(252,1:251)]) # reorder


#png(filename = "./images/test.pop")
ggplot(Dtest, aes(Dtest$X.step., Dtest$count.pills+Dtest$count.pacmen, group=factor(Dtest$X.run.number.)
  geom_line(aes(color=factor(Dtest$X.run.number.))) +
  labs(x='Step', y='Total Population', colour='Run Number') +
  guides(col = guide_legend(nrow = 10))
```




```
#dev.off()

# the Health measure for each over same time
ggplot(Dtest, aes(Dtest$X.step., Dtest$healthMeasure, group=factor(Dtest$X.run.number.))) + geom_line(a
  labs(x='Step', y='Health Measure', colour='Run Number')
```

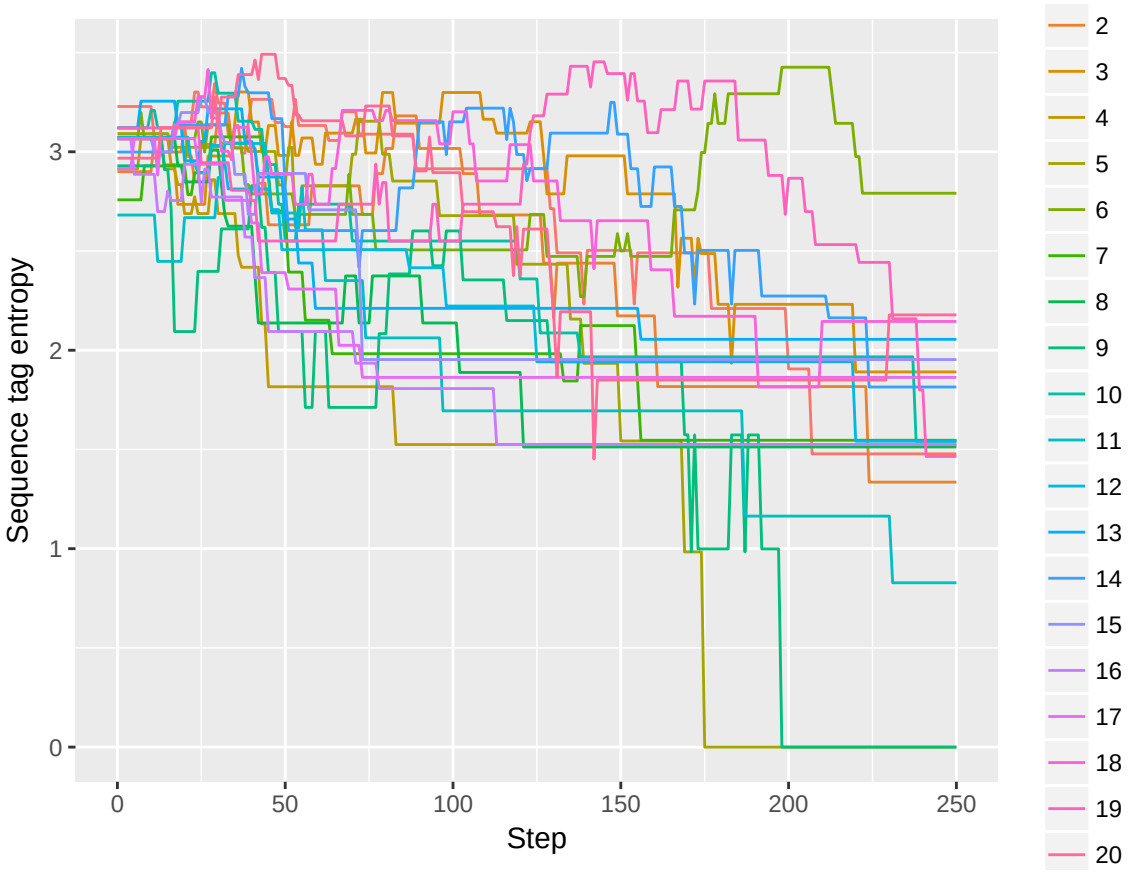


measure

$$E_h[hamming] * H(sequencetag) * \frac{\sum patchSugar}{max(patchSugar)}$$

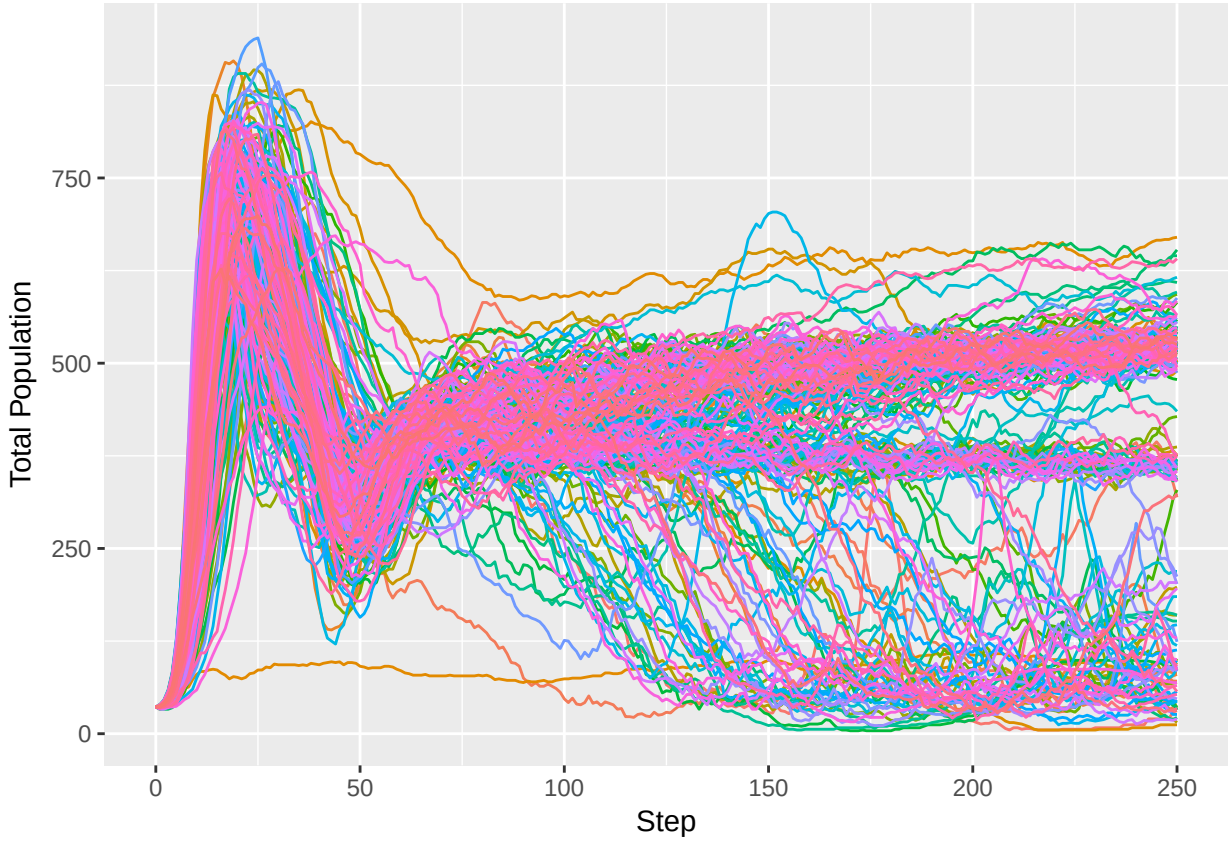
I would rather have a model that did not reach a steady state, but could crash or have periods of innovation. My model kinda does, but it isn't long enough. The crashes happen quickly and I don't know how I could very statistics to keep it in check. Should create a better model in the future. I do get volka terra?? predetor / prey like population explosion and collapse

```
# instantaneous entropy
ggplot(Dtest, aes(Dtest$X.step., Dtest$sequence.tag.entropy, group=factor(Dtest$X.run.number.))) +
  geom_line(aes(color=factor(Dtest$X.run.number.))) +
  labs(x='Step', y='Sequence tag entropy', colour='Run Number')
```

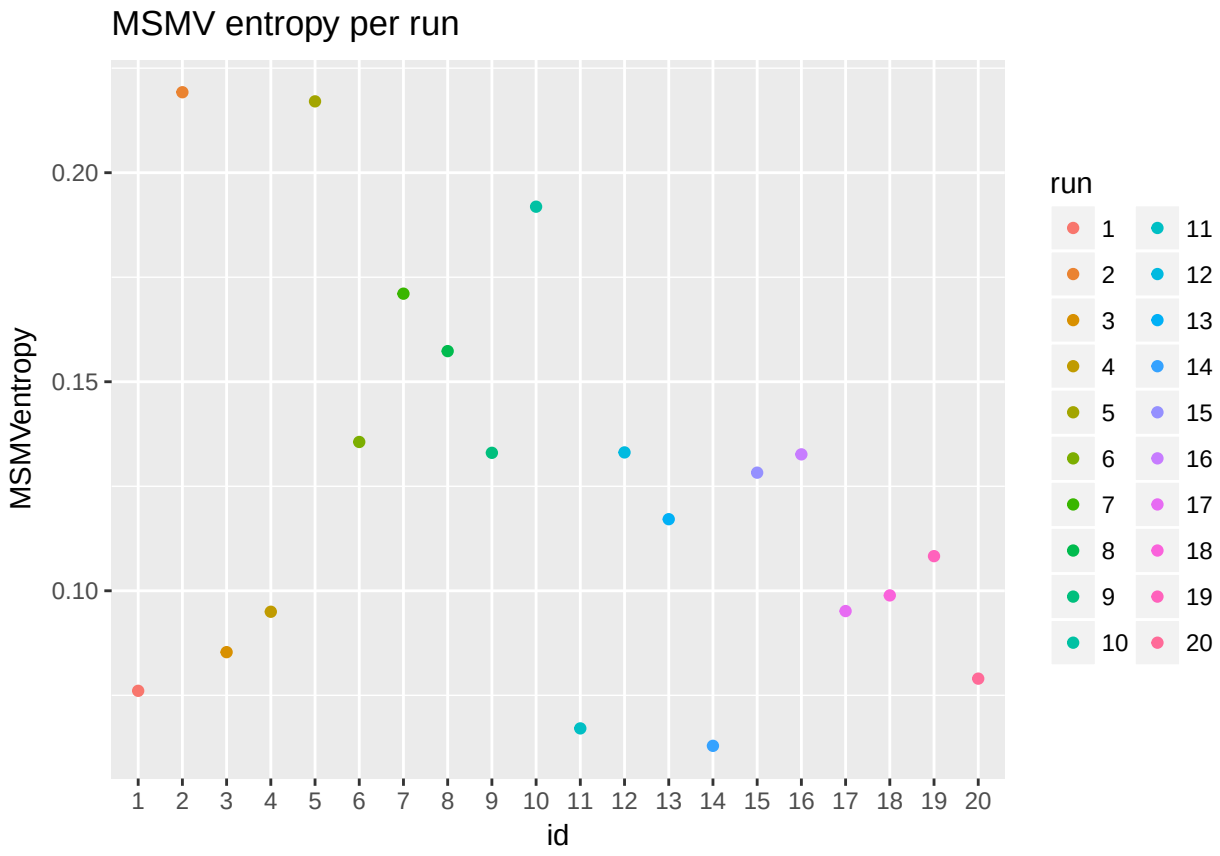


The instantaneous entropy is an entropy of the sequence tag frequency where the sequence tags are the different species. It is a measure of uncertainty in diversity

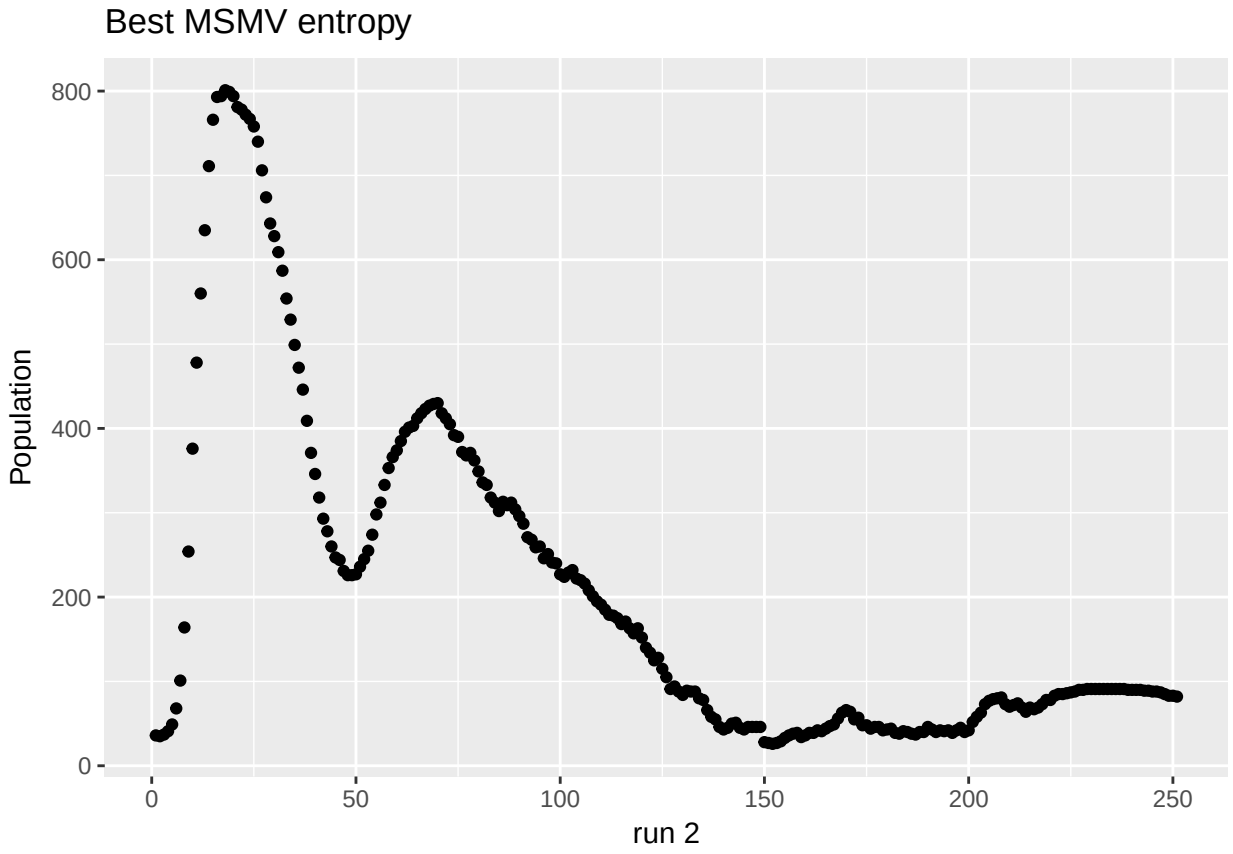
```
ggplot(Dtrain, aes(Dtrain$X.step., Dtrain$count.pills+Dtrain$count.pacmen, group=factor(Dtrain$X.run.nu
  geom_line(aes(color=factor(Dtrain$X.run.number.))) +
  labs(x='Step', y='Total Population', colour='Run Number') +
  theme(legend.position = 'none')
```



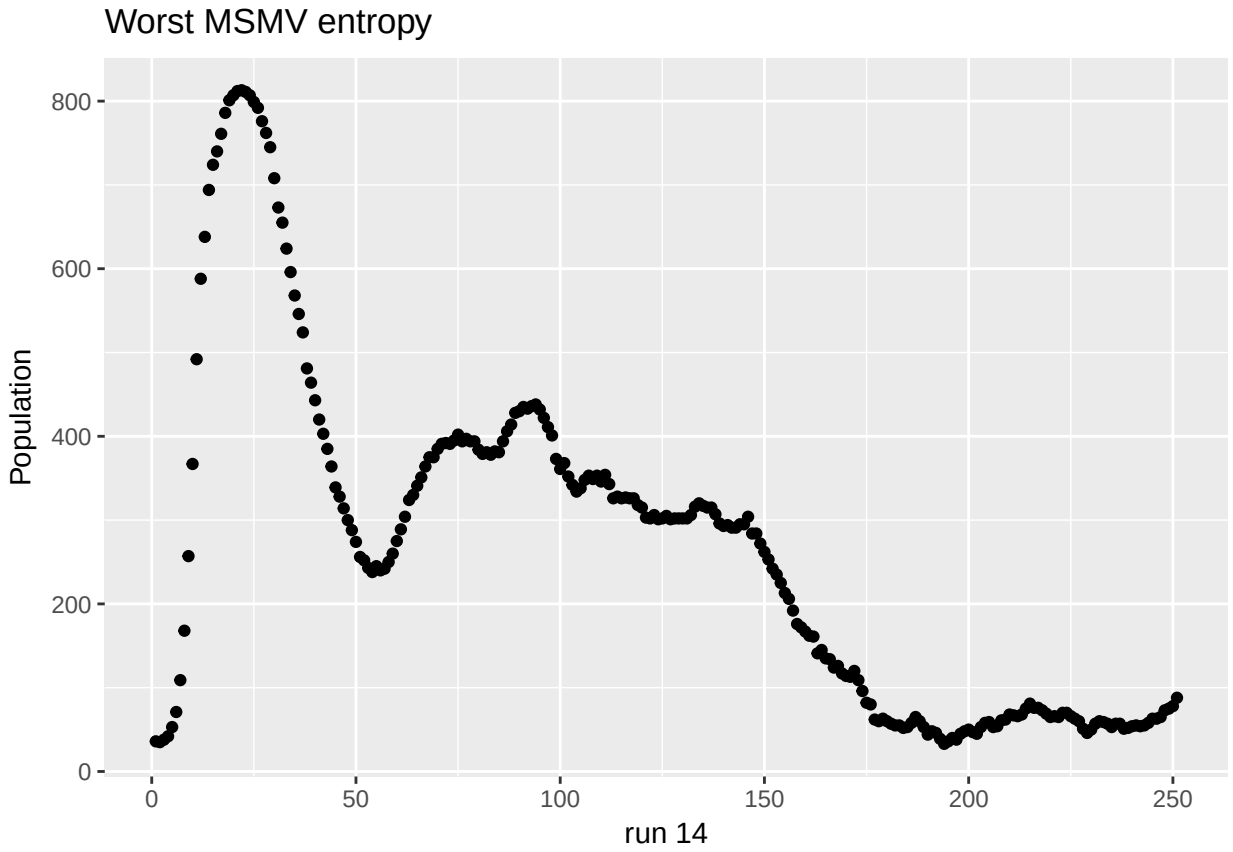

```
# Calculate MSE for each run


# list for each run
# this is only for the test set and it's not multivariate
MSMVEntropy.test = vector(mode = "numeric", length = max(Dtest$X.run.number.))
for(i in 1:max(Dtest$X.run.number.)){
  run <- subset(Dtest, Dtest$X.run.number == i)
  MSMVEntropy.test[i] <- MSMVSampEn(mat = matrix(run$count.pacmen), M = emb.dim,
           tau = sig.tau.ami, r = .5, eps = 1,
           scaleMat = T) # documents say it should be scaled, but I didn't with the last
}
MSMVdf <- data.frame(id = factor(1:length(MSMVEntropy.test)), MSMVentropy = MSMVEntropy.test)

ggplot(MSMVdf, aes(id, MSMVentropy, group=factor(id))) +
  geom_point(aes(color=factor(id))) +
  ggtitle("MSMV entropy per run") +
  scale_color_discrete('run') +
  guides(col = guide_legend(nrow = 10))
```




```
#run 10 highest

qplot(1:251,subset(Dtest, Dtest$X.run.number == MSMVdf[which(MSMVdf$MSMVentropy==max(MSMVdf$MSMVentropy
      ylab = 'Population')
```




```
qplot(1:251,subset(Dtest, Dtest$X.run.number == MSMVdf[which(MSMVdf$MSMVentropy==min(MSMVdf$MSMVentropy
      ylab = 'Population')
```



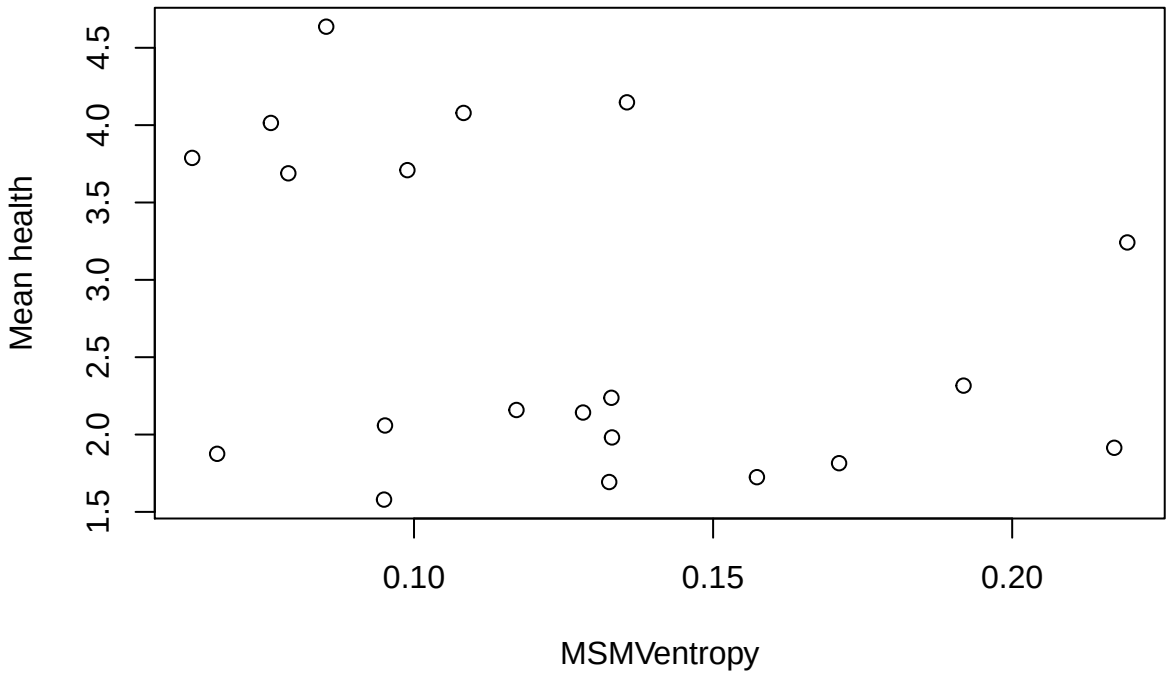
I'm not using the multivariate functionallity. This is really a multiscale entropy measure or MSE.

$$SampEn(m, r, N) = -\ln \frac{A^m(r)}{B^m(r)}$$

Multiscale entropy is based on sample entropy except for the calculation includes different scales. The "coarse graining" is described in the previous nonlinear anaylisis section

There doesn't seem to be any qualitative difference between the worst and best entropies. Intially, I did not standardize and got results I could justify a qualitative difference, but apon further inspection, I don't see any qualitative difference.

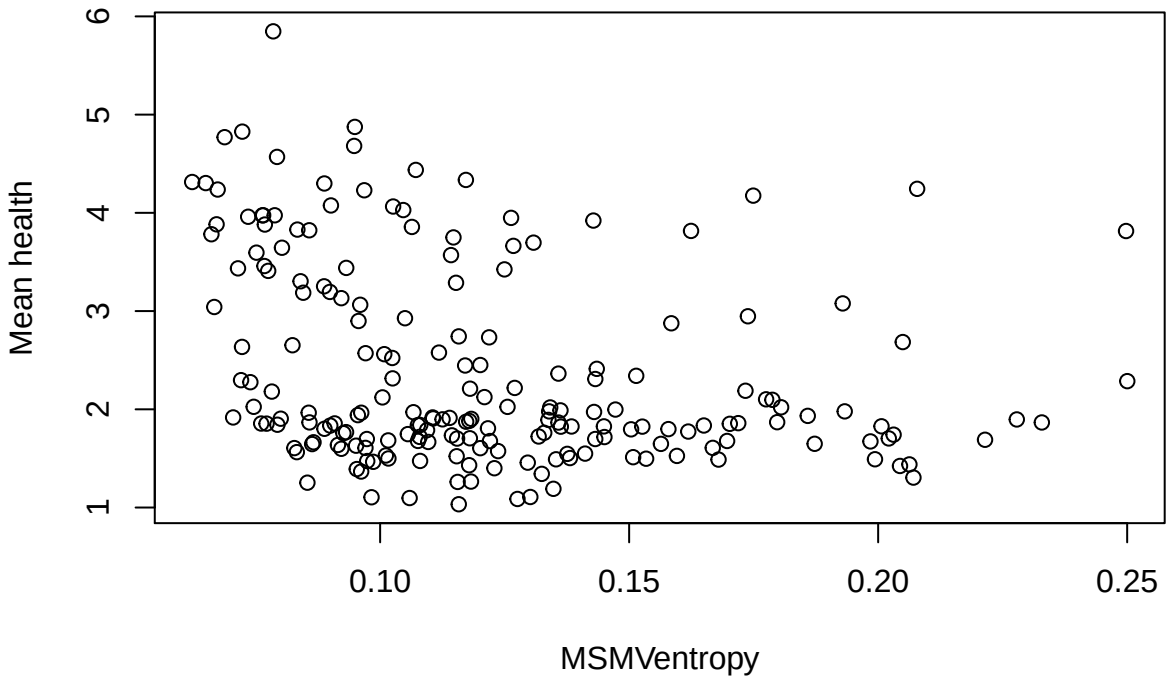
```
avg.health.test <- 0
for(i in 1:max(Dtest$X.run.number.)){
  run <- subset(Dtest, Dtest$X.run.number == i)
  avg.health.test[i] <- mean(run$healthMeasure)
}
MSMVdf$'Mean health' <- avg.health.test
plot(MSMVdf[,-1])
```




```
cor.test(MSMVdf$MSMVentropy,MSMVdf$`Mean health`) # not really linearly correlated
```

```
##
##  Pearson's product-moment correlation
##
## data:  MSMVdf$MSMVentropy and MSMVdf$`Mean health`
## t = -1.599, df = 18, p-value = 0.1272
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.6878474  0.1064588
## sample estimates:
##        cor
## -0.3526767
```

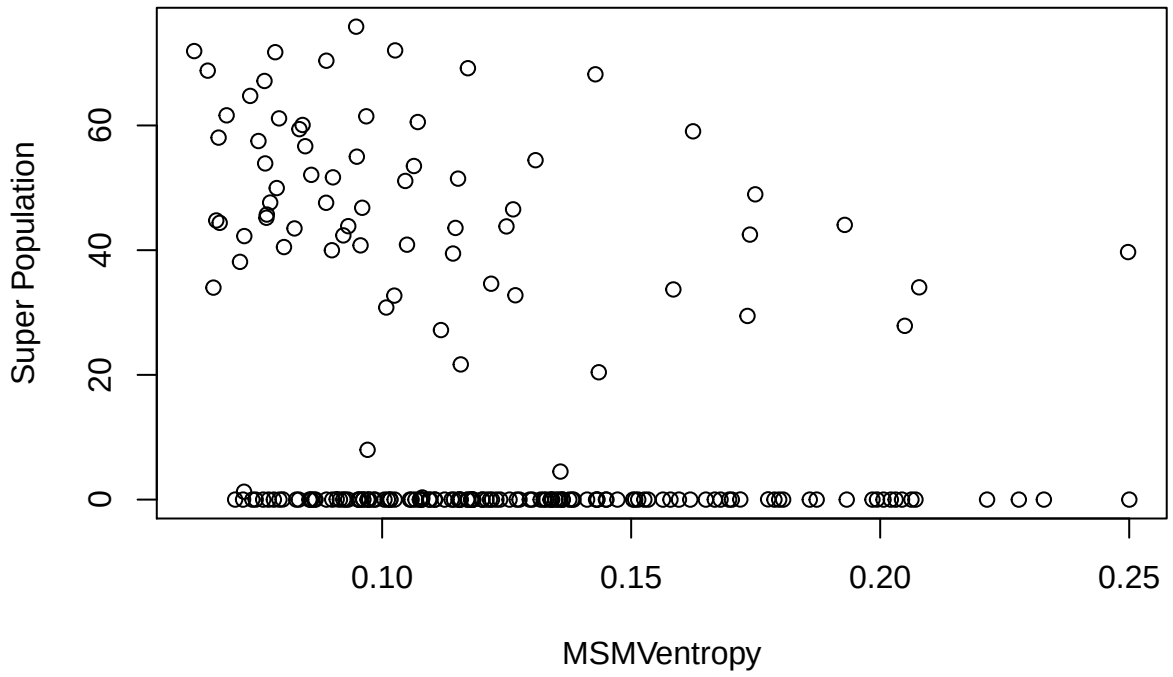
```
# lets do this for the training set too
MSMVEntropy.train = vector(mode = "numeric", length = max(Dtrain$X.run.number.))
for(i in 1:max(Dtrain$X.run.number.)){
  run <- subset(Dtrain, Dtrain$X.run.number == i)
  MSMVEntropy.train[i] <- MSMVSampEn(mat = matrix(run$count.pacmen), M = emb.dim,
           tau = sig.tau.ami, r = .5, eps = 1,
           scaleMat = T) # Previous data is not scaled(problem??)
}
MSMVdf.train <- data.frame(id = factor(1:length(MSMVEntropy.train)), MSMVentropy = MSMVEntropy.train)

avg.health.train <- 0
for(i in 1:max(Dtrain$X.run.number.)){
  run <- subset(Dtrain, Dtrain$X.run.number == i)
  avg.health.train[i] <- mean(run$healthMeasure)
}
MSMVdf.train$'Mean health' <- avg.health.train

# png(filename = './images/enVShealth.png')
plot(MSMVdf.train[,c(2,3)])
```




```
cor.test(MSMVdf.train$MSMVentropy,MSMVdf.train$`Mean health`)
```

```
##
##  Pearson's product-moment correlation
##
## data:  MSMVdf.train$MSMVentropy and MSMVdf.train$`Mean health`
## t = -4.342, df = 198, p-value = 2.251e-05
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.4165565 -0.1627741
## sample estimates:
##        cor
## -0.2948562
```

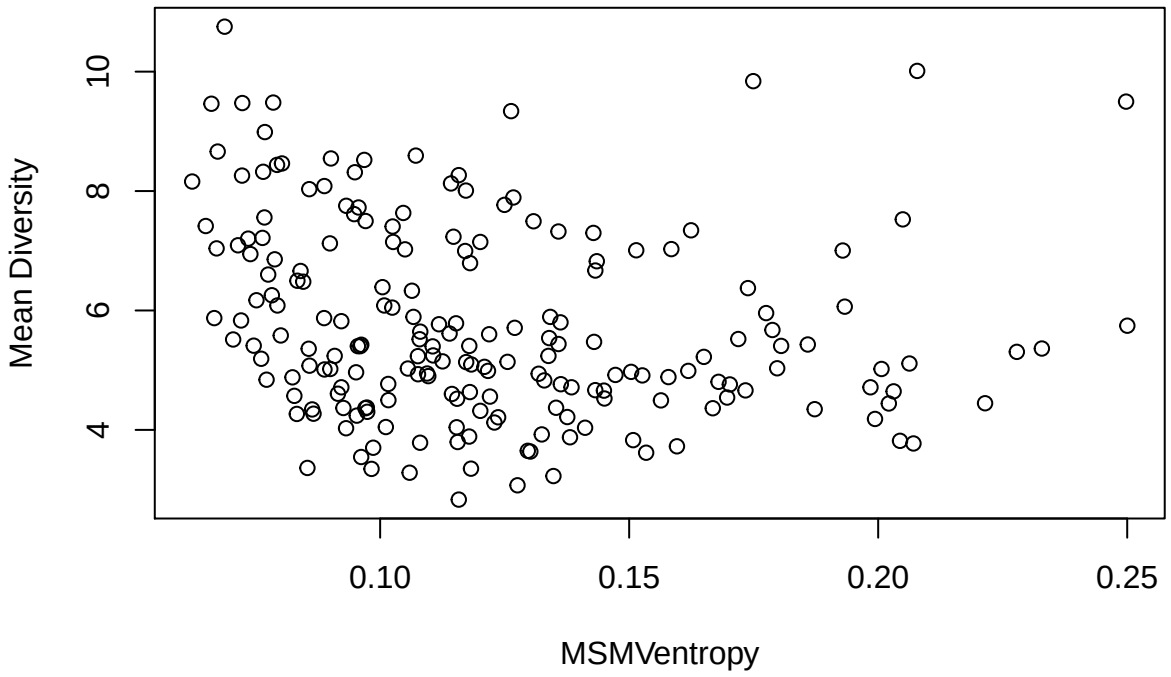
```
# dev.off()
# slightly negatively correlated. Not What I expected ??
# maybe there is a relation to the emergence of super turtles

avg.super.train <- 0
for(i in 1:max(Dtrain$X.run.number.)){
  run <- subset(Dtrain, Dtrain$X.run.number == i)
  avg.super.train[i] <- mean(run$count.supers)
}
MSMVdf.train$'Super Population' <- avg.super.train
plot(MSMVdf.train[,c(-1,-3)])
```




```
cor.test(MSMVdf.train$MSMVentropy,MSMVdf.train$`Super Population`)
```

```
##
##  Pearson's product-moment correlation
##
## data:  MSMVdf.train$MSMVentropy and MSMVdf.train$`Super Population`
## t = -4.597, df = 198, p-value = 7.627e-06
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.4307275 -0.1795390
## sample estimates:
##        cor
## -0.3105445
```

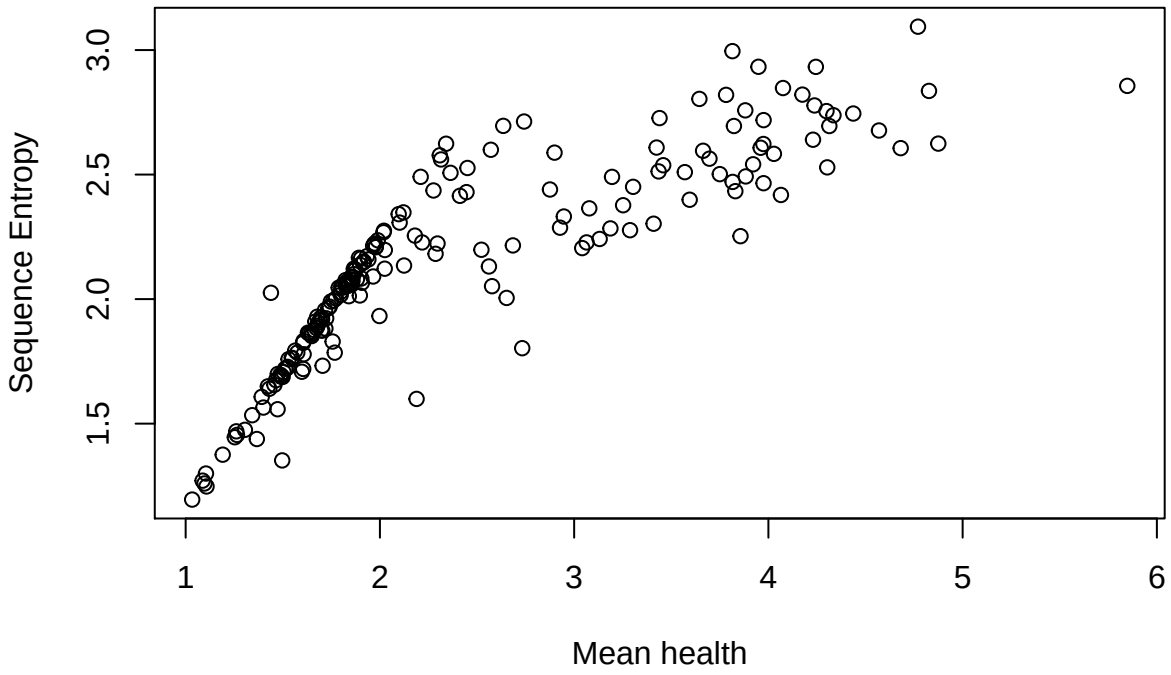
```
# lets try with diversity
avg.diversity.train <- 0
for(i in 1:max(Dtrain$X.run.number.)){
  run <- subset(Dtrain, Dtrain$X.run.number == i)
  avg.diversity.train[i] <- mean(run$diversity)
}
MSMVdf.train$'Mean Diversity' <- avg.diversity.train
plot(MSMVdf.train[,c(-1,-3,-4)])
```




```
cor.test(MSMVdf.train$MSMVentropy,MSMVdf.train$`Mean Diversity`)
```

```
##
##  Pearson's product-moment correlation
##
## data:  MSMVdf.train$MSMVentropy and MSMVdf.train$`Mean Diversity`
## t = -2.9427, df = 198, p-value = 0.003642
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.33395761 -0.06788817
## sample estimates:
##        cor
## -0.2047011
```

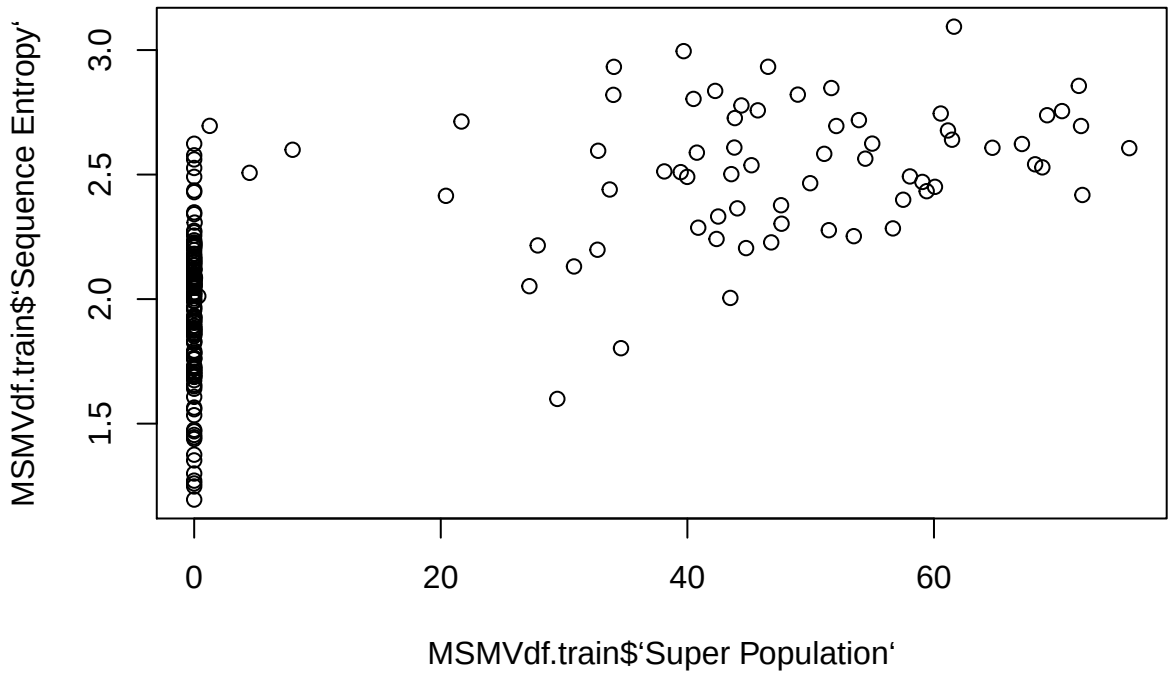
```
## Maybe the average sequence tag entropy has some meaning
avg.seq.entropy.train <- 0
for(i in 1:max(Dtrain$X.run.number.)){
  run <- subset(Dtrain, Dtrain$X.run.number == i)
  avg.seq.entropy.train[i] <- mean(run$sequence.tag.entropy)
}
MSMVdf.train$'Sequence Entropy' <- avg.seq.entropy.train
plot(MSMVdf.train[,c(-1,-2,-4,-5)])
```




```
cor.test(MSMVdf.train$`Mean health`,MSMVdf.train$`Sequence Entropy`)
```

```
##
##  Pearson's product-moment correlation
##
## data:  MSMVdf.train$`Mean health` and MSMVdf.train$`Sequence Entropy`
## t = 23.411, df = 198, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.8153086 0.8900041
## sample estimates:
##       cor
## 0.8570976
```

```
## sequence tag vs super emergence
plot(MSMVdf.train$`Super Population`,MSMVdf.train$`Sequence Entropy`)
```

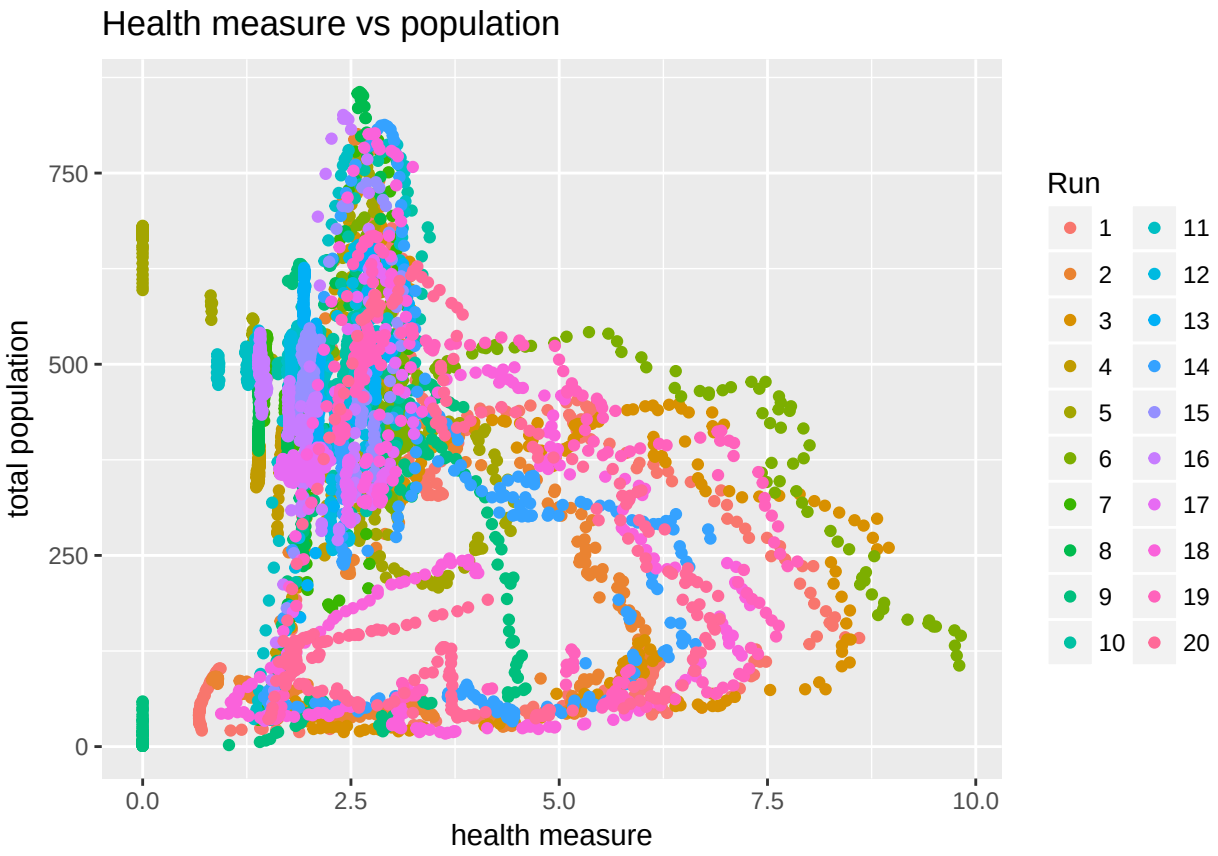



```
# from the plot it doesn't mean anything really
cor.test(MSMVdf.train$`Super Population`,MSMVdf.train$`Sequence Entropy`)
```

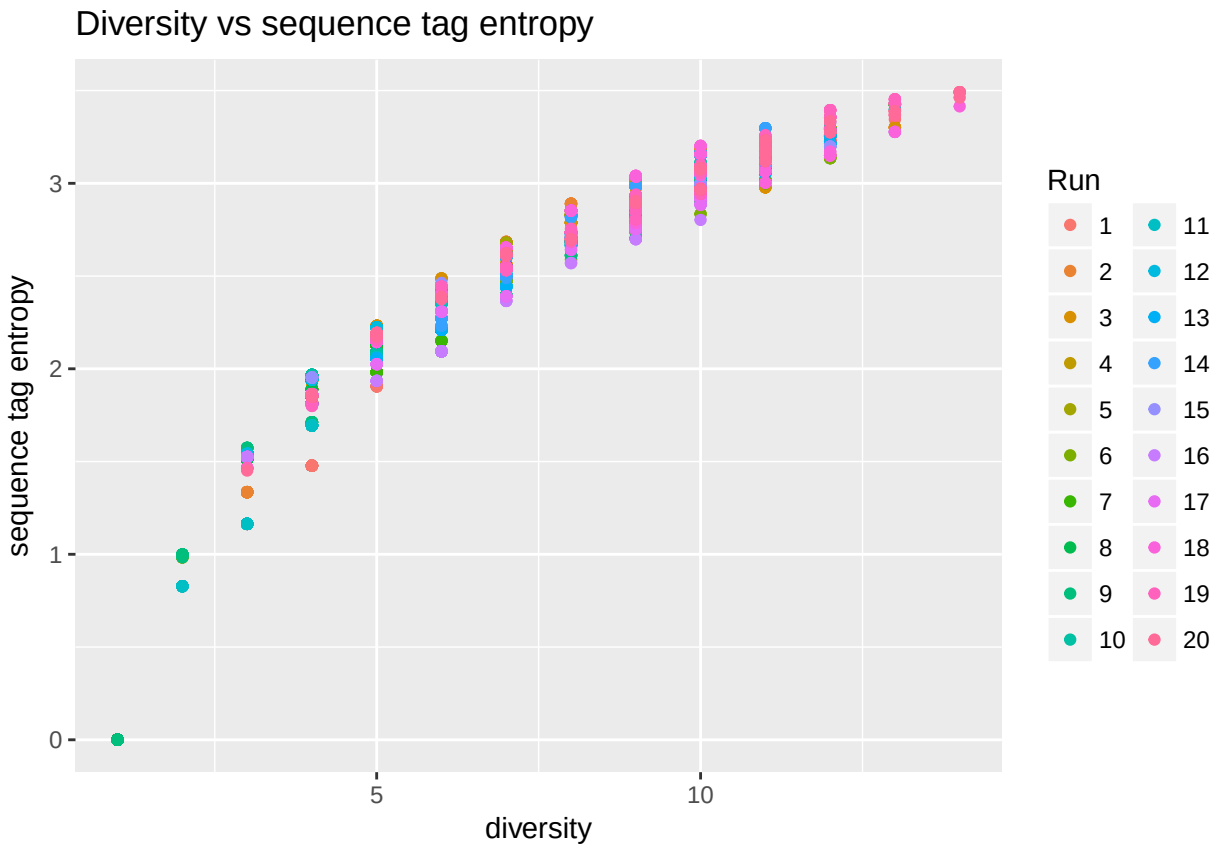
```
##
##  Pearson's product-moment correlation
##
## data:  MSMVdf.train$`Super Population` and MSMVdf.train$`Sequence Entropy`
## t = 13.031, df = 198, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.5969937 0.7477123
## sample estimates:
##      cor
## 0.679457
```

```
# but it is correlated (typically more sequence tag diversity with supers)

## health measure
ggplot(Dtest, aes(Dtest$healthMeasure, Dtest$sequence.tag.entropy, group=factor(Dtest$X.run.number.))) +
  geom_line(aes(color=factor(Dtest$X.run.number.))) +
  ggtitle("Health measure vs sequence tag entropy") +
  xlab("health measure") +
  ylab('sequence tag entropy') +
  guides(col = guide_legend(nrow = 10)) +
  scale_colour_discrete(name = 'Run')
```

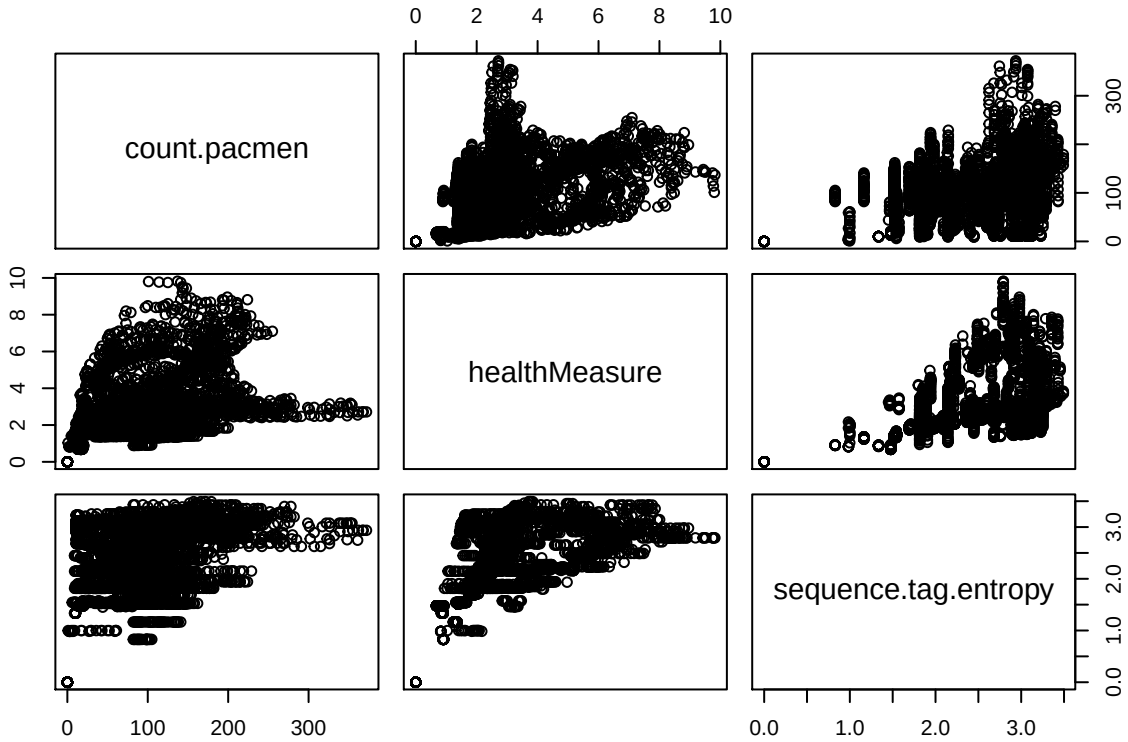

```
ggplot(Dtest, aes(Dtest$healthMeasure, Dtest$count.pills+Dtest$count.pacmen, group=factor(Dtest$X.run.nu
  geom_point(aes(color=factor(Dtest$X.run.number.))) +
  labs(title='Health measure vs population',x='health measure',y='total population') +
  guides(col = guide_legend(nrow = 10)) +
  scale_colour_discrete(name = 'Run')
```




```
ggplot(Dtest, aes(Dtest$diversity, Dtest$sequence.tag.entropy, group=factor(Dtest$X.run.number.))) +
  geom_point(aes(color=factor(Dtest$X.run.number.))) +
  labs(title='Diversity vs sequence tag entropy',x='diversity',y='sequence tag entropy') +
  guides(col = guide_legend(nrow = 10)) +
  scale_colour_discrete(name = 'Run')
```




```
plot(Dtest[,c('count.pacmen','healthMeasure','sequence.tag.entropy')])
```



There is a small negative statstical correlation between the MSMV entropy measure and the health measure

Since the health measure is made up and the entropy is dependent on the radius and scaling, maybe it doesn't

work here.

I get a similar small statistically significant negative correlation for the emergence of supers in the training set. This makes sense considering the depedence of the geneology length and the health measure.

Given just the population for our entropy, the MSMVentropy measure doesn't really give us much.

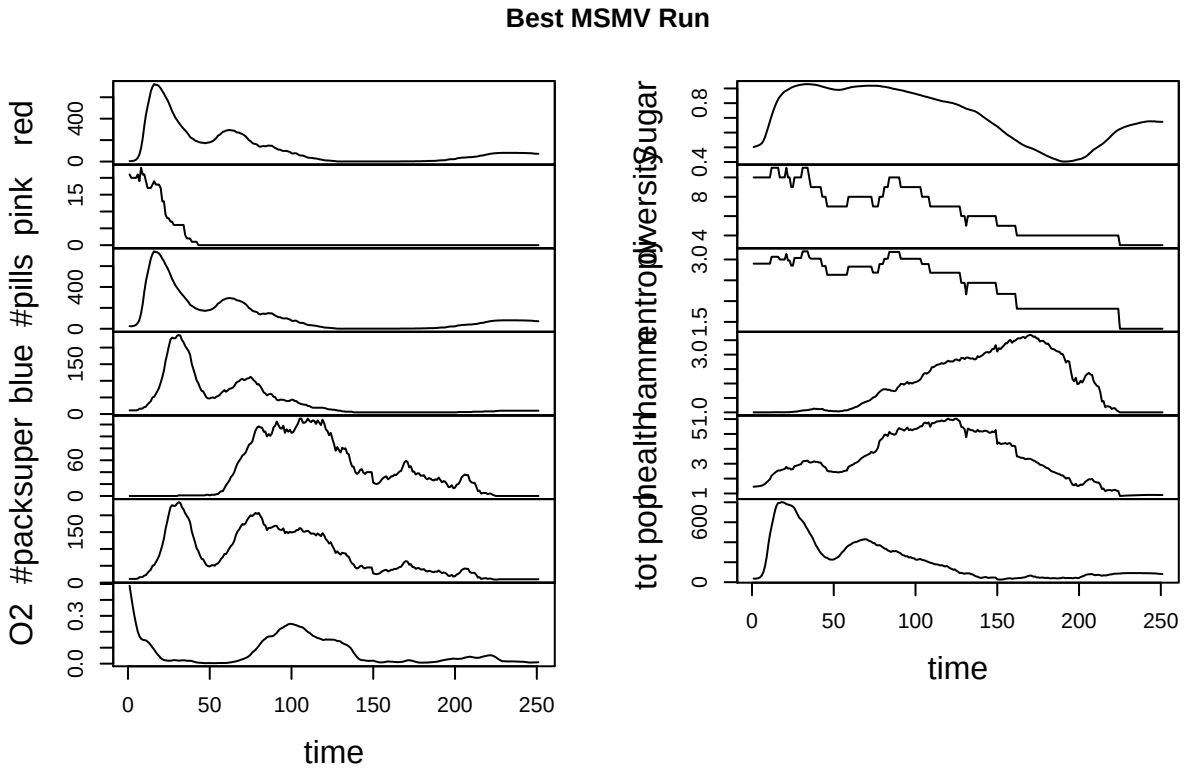
Sequence Tag entropy is correlated with the health measure. It's defined that way as a measure of the organization of the ecosystem. But it is also predictive of Super emergence.

```r
# need better column names and I don't really need the setup
# but I don't have time to go through what I've already written
# to change it

Dtest1 <- Dtest[,c(1,2,14:28)]
Dtrain1 <- Dtrain[,c(1,2,14:28)]

head(Dtest1) # for datatype reference in document
```

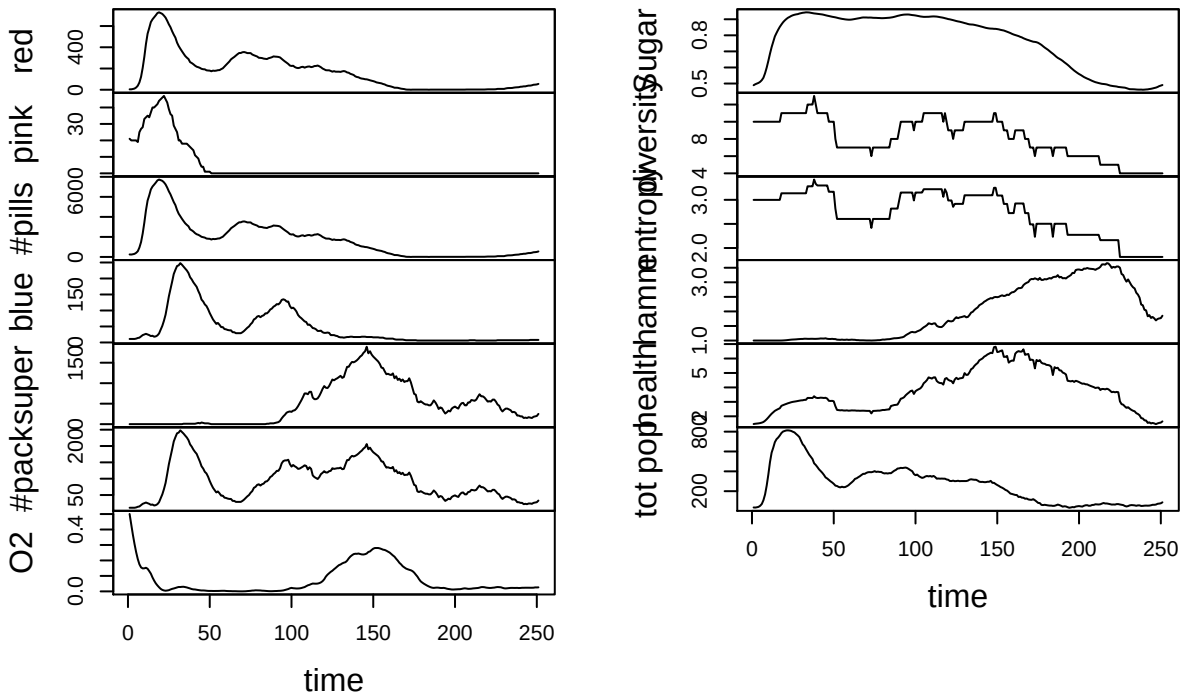
```
##   X.run.number. X.step. count.pills.with..color...red.
## 1             1       0                              4
## 2             1       1                              5
## 3             1       2                              7
## 4             1       3                             11
## 5             1       4                             19
## 6             1       5                             35
##   count.pills.with..color...pink. count.pills
## 1                              21          25
## 2                              21          26
## 3                              21          28
## 4                              20          31
## 5                              20          39
## 6                              23          58
##   count.pacmen.with..color...blue. count.supers count.pacmen
## 1                               11            0           11
## 2                               11            0           11
## 3                               11            0           11
## 4                               12            0           12
## 5                               12            0           12
## 6                               12            0           12
##
## 1
## 2
## 3                                                                                      1, 1,
## 4                                                                              1, 1, 1, 1, 1, 1,
## 5                                                              1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 6 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
##
## 1
## 2
## 3
## 4
## 5                                                                          29, 1, 5, 14, 29, 9, 29,
## 6 13, 29, 29, 29, 1, 29, 14, 5, 13, 29, 29, 29, 13, 29, 29, 6, 2, 29, 29, 9, 29, 29, 2, 5, 9, 10, 29
##   X1....sum..pO2..of.patches...sum..max.pO2..of.patches.
## 1                                              0.5249320
```

```
## 2                                                0.4533815
## 3                                                0.3854929
## 4                                                0.3231241
## 5                                                0.2689042
## 6                                                0.2226440
##   X1...sum..psugar..of.patches...sum..max.psugar..of.patches diversity
## 1                                                  0.4981499        12
## 2                                                  0.5031983        12
## 3                                                  0.5079951        12
## 4                                                  0.5113429        12
## 5                                                  0.5174755        12
## 6                                                  0.5268062        12
##   sequence.tag.entropy mean.hamming healthMeasure turtles
## 1             3.228322            1      1.608188      36
## 2             3.228322            1      1.624486      37
## 3             3.228322            1      1.639972      39
## 4             3.228322            1      1.650779      43
## 5             3.228322            1      1.670577      51
## 6             3.228322            1      1.700700      70
```

```
new.names <- c("run number", "time", "red pills", "pink pills", "total pills",
               "blue packmen", "super packmen", "total packmen", "geneology length",
               "sequence tags", "O2 utilization", "sugar utilization",
               "diversity magnitude", "entropy of unique sequences", "mean hamming",
               "health measure", "total population")

names(Dtest1) <- new.names
names(Dtrain1) <- new.names


# zoo so I can easily plot a run
# look at run with worst and best MSMV entropy
ts.data <- Dtest1[,- c(9,10)]

# best run by MSMV entropy
best.ts <- subset(ts.data, ts.data$`run number` ==
                    MSMVdf[which(MSMVdf$MSMVentropy==max(MSMVdf$MSMVentropy)),1])

worst.ts <- subset(ts.data, ts.data$`run number` ==
                     MSMVdf[which(MSMVdf$MSMVentropy==min(MSMVdf$MSMVentropy)),1])

#ts.plot(best.ts)
ynames <- c('red','pink','#pills','blue','super','#pack',
         'O2','Sugar','diversity','entropy','hamm','health','tot pop')
plot(zoo(best.ts[,-c(1,2)]),xlab = 'time', ylab = ynames,main = "Best MSMV Run")
```




```
#worst
plot(zoo(worst.ts[,-c(1,2)]),xlab = 'time', ylab = ynames,main = "Worst MSMV Run")
```



The plot above shows the relationships for the best MSMVSampEn run for the second set of experiments. The best MSMVSampEn run has relatively low total population, but has a large population of "supers" and is diverse. This could be similar to a healthy ecosystem with large predator species vs a very high population of

much smaller species in an unhealthy ecosystem.

The lowest MSMVSampEn run shows a high population and high utilization, but low diversity and low long term complexity across scales. This is analogous to a pond full of algae and little else

prediction

After further analysis, the entropy measure of the population doesn't seem to be related to the system health. This is the reason I only get positive predictions using the nonlinear classification techniques below.

Using the first 100 time steps to attempt to predict the health at time step 150

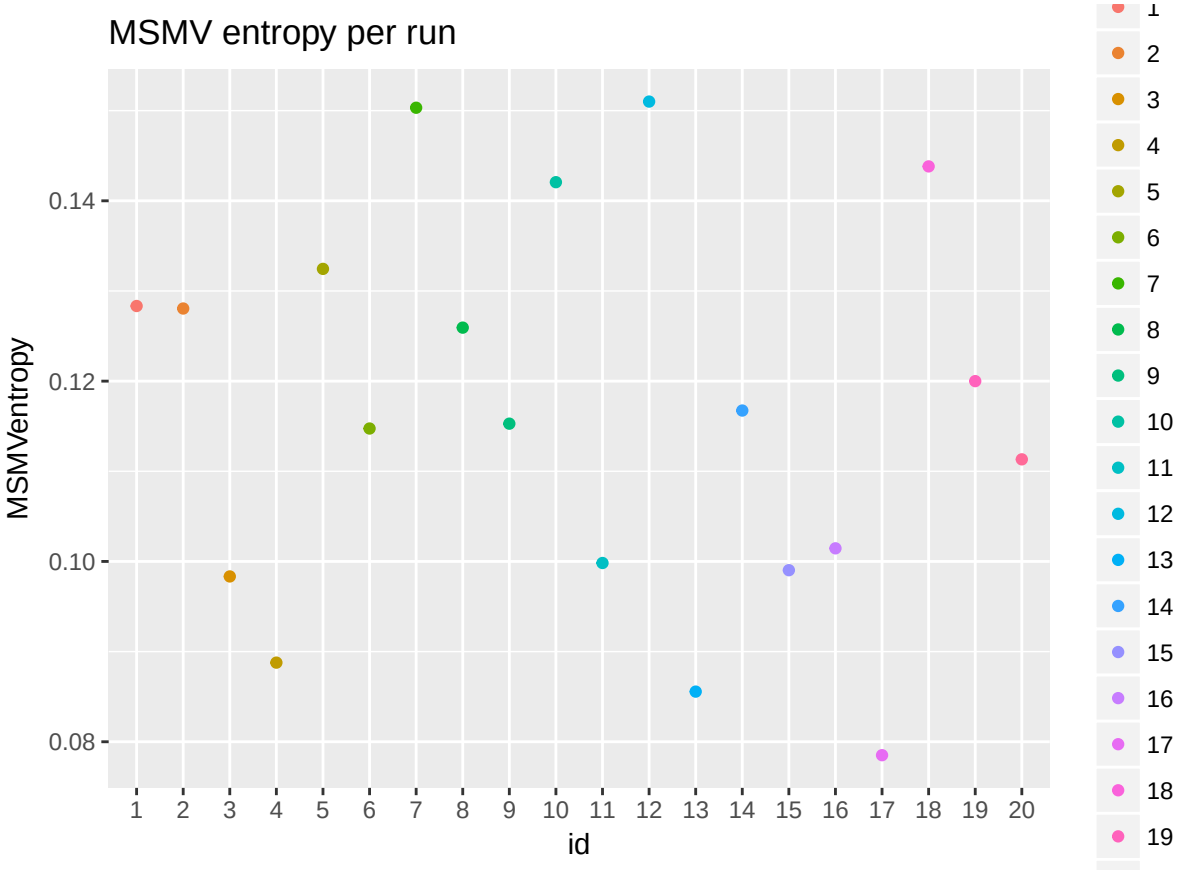
```
# doesn't work because the measure doesn't work
# setting good health at greater than 5 at end of run

Dtest1$'instantaneous health' <- lapply(Dtest1$`health measure`,
                                  FUN = function(x) ifelse(x>3,'good',
                                                           'bad'))

Dtrain1$'instantaneous health' <- lapply(Dtrain1$`health measure`,
                                  FUN = function(x) ifelse(x>3,'good',
                                                           'bad'))
# testing for margins percent == good
a <- subset(Dtest1, Dtest1$`instantaneous health` == 'good')
length(unique(a$`run number`))/20 # %good health
```

```
## [1] 0.8
```

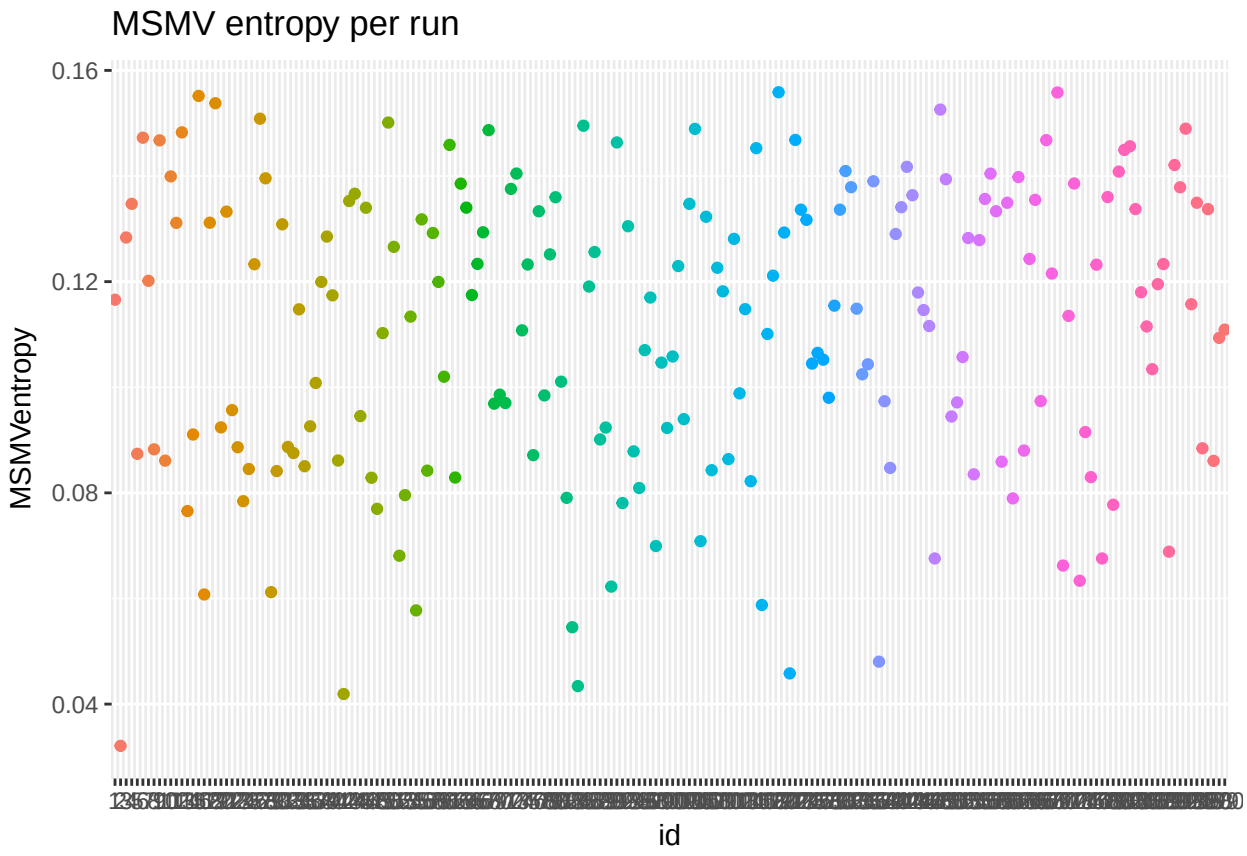
```
a<- subset(Dtrain1, Dtrain1$`instantaneous health` == 'good')
length(unique(a$`run number`))/200 # %good health
```

```
## [1] 0.505
```

```
# MSMV for first 150 time steps
MSMVEntropy.test1 = vector(mode = "numeric", length = max(Dtest1$`run number`))
for(i in 1:max(Dtest1$`run number`)){
  run <- subset(Dtest1, Dtest1$`run number` == i)
  MSMVEntropy.test1[i] <- MSMVSampEn(mat = matrix(run$`total population`[1:100]),
                                     M = c(2), tau = c(1), r = .5, eps = 1,
                                     scaleMat = T)
}
MSMVdf.test1 <- data.frame(id = factor(1:length(MSMVEntropy.test1)), MSMVentropy = MSMVEntropy.test1)

ggplot(MSMVdf.test1, aes(id, MSMVentropy, group=factor(id))) +
  geom_point(aes(color=factor(id))) +
  ggtitle("MSMV entropy per run")
```




```
# adding health
health <- list()
for(i in 1:max(Dtest1$`run number`)){
  run <- subset(Dtest1, Dtest1$`run number` == i)
  health[i] <- run$`health measure`[150]
}
MSMVdf.test1$health <- health

#### same for train
# MSMV for first 100 time steps
MSMVEntropy.train1 = vector(mode = "numeric", length = max(Dtrain1$`run number`))
for(i in 1:max(Dtrain1$`run number`)){
  run <- subset(Dtrain1, Dtrain1$`run number` == i)
  MSMVEntropy.train1[i] <- MSMVSampEn(mat = matrix(run$`total population`[1:100]),
                                      M = c(2), tau = c(1), r = .5, eps = 1,
                                      scaleMat = T)
}
MSMVdf.train1 <- data.frame(id = factor(1:length(MSMVEntropy.train1)), MSMVentropy = MSMVEntropy.train1)

ggplot(MSMVdf.train1, aes(id, MSMVentropy, group=factor(id))) +
  geom_point(aes(color=factor(id))) +
  theme(legend.position = 'none') +
  ggtitle("MSMV entropy per run")
```




```
# adding health
health <- list()
for(i in 1:max(Dtrain1$`run number`)){
  run <- subset(Dtrain1, Dtrain1$`run number` == i)
  health[[i]] <- run$`health measure`[150]
}
# health
MSMVdf.train1$health <- health




print.me = list()
i=1
train.y <- lapply(MSMVdf.train1$health, function (x) ifelse(x>3,1,0)) #0 good; 1 bad
train.x <- data.frame(MSMVentropy = MSMVdf.train1$MSMVentropy)
train.y <- factor(unlist(train.y))

test.y <- lapply(MSMVdf.test1$health, function (x) ifelse(x>3,1,0))
test.x <- data.frame(MSMVentropy = MSMVdf.test1$MSMVentropy)
test.y <- factor(unlist(test.y))

print.me$method[i]="CART"
# selection via complexity parameter
modelCART=train(train.x,train.y,
                method="rpart",
                preProcess=c("center","scale","knnImpute"),
```

```
                trControl=trainControl("cv", number = 10))
```

```
## Loading required package: rpart
```

```
train.predict=predict(modelCART,train.x)
print.me$train.error[i]=sum(train.y != train.predict)/length(train.y)
test.predict=predict(modelCART,test.x)
print.me$test.error[i]=sum(test.y != test.predict)/length(test.y)
print.me
```

```
## $method
## [1] "CART"
##
## $train.error
## [1] 0.225
##
## $test.error
## [1] 0.55
```

```
######## evaluating performance
# I started using plr for penalized logistic regression
# i = 1
# cart.prob = predict(modelCART, test.x, type = "prob")
# test.predict = predict(modelplr, test.x)
# pred = prediction(plr.prob$`1`, test.y)
#
# perf <- performance(pred,'tpr', 'fpr')
#
# print.me$auc[i] <- performance(pred, 'auc')@y.values[[1]]
#
# rocDataFrame <- data.frame(tpr=perf@x.values[[1]],fpr=perf@y.values[[1]],
#                            type=as.factor('Logistic Regression'))

cart.confusion = confusionMatrix(data = test.predict,
                                 reference = test.y, mode = "prec_recall")
cart.confusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
##          0 9 9
##          1 2 0
##
##                Accuracy : 0.45
##                  95% CI : (0.2306, 0.6847)
##     No Information Rate : 0.55
##     P-Value [Acc > NIR] : 0.86924
##
##                   Kappa : -0.1957
##  Mcnemar's Test P-Value : 0.07044
##
##               Precision : 0.5000
##                  Recall : 0.8182
##                      F1 : 0.6207
```

```
##              Prevalence : 0.5500
##          Detection Rate : 0.4500
##    Detection Prevalence : 0.9000
##       Balanced Accuracy : 0.4091
##
##        'Positive' Class : 0
##
```

Conclusion

I learned a lot, but a bit of the mystic of the nonlinear methods for complex regularity has worn off. I think in specific situations with knowledge of the underlying dynamics the measure can be very useful. I'm skeptical of its applicability in real world distributed systems. The methods are essentially used to find patterns that linear approximators would miss, but current statistical learning methods are likely to find the same patterns with less ambiguity in the hyperparameters. However, the statistics could still prove to be useful in preprocessing or feature generation.

After spending more time than I would like to admit, this study is also fundametally flawed. The ABM simulation does not have the type of long term nonlinear regularity we would like to see. It does not provide a good test bed for studying these measures. I would need to develop a model that better simulates real world ecosystems or rely on real data to really test the usefulness of the statistics.