LAB 6 - Empirical Dynamics

ER Deyle

Fall 2024; Marine Semester Block 4

Fisheries Context

Computational Approach

Empirical Dynamic Modeling

The analytical approach will rely on the `rEDM` package, and particularly the basic application of the `Simplex()` and `Smap()` functions as described in the `rEDM-tutorial` vignette. Before moving forward with the Lab analysis, please complete the rEDM tutorial.

```
library('rEDM')
vignette('rEDM-tutorial')
```

Ram Myers Data-base

The data for the lab tasks will come from the Ram Myers Legacy Database. Specifically, we'll extract data on fish stocks on the US East Coast. An overview of the database itself and the science around it are available online:

https://www.ramlegacy.org/

Including specific information on the US East Coast:

https://www.ramlegacy.org/explore-the-database/regions/us-east-coast/

However, the class OneDrive contains the `.Rdata` file that you need (within the "Lab 6" folder). Download a local copy of that data-set to begin the lab!

Data Wrangling

You can load the data as follows:

```
load("DBdata[asmt][v4.495].RData")
```

This data-base is large, and the organization is… interesting. There is a meta-data object, `stock` that we can use as a starting place.

```
stock_US_East <- subset(stock,region=="US East Coast")
NROW(stock_US_East)
```

```
## [1] 58
```

There are lots of fish stocks in here that have come up at various times in class, and others you probably have some degree of familiarity with from other classes and contexts: Atlantic menhaden, Striped bass, Spiny dogfish, Weakfish. If one of these calls to you, you may proceed with an alternative choice.

For illustration, let's look at cod (*Gadus morhua*) and herring (*Clupea harengus*). We can use the `grepl` command to return `TRUE` or `FALSE` depending on if the `commonname` column contains either word. In my own research I tend to use `stringr` commands instead of the base R `grep`, `grepl`, and `regexpr`, but there is a lot of similarity. Note that the `areaid` differs.

```
subset(stock_US_East,grepl("cod",commonname,ignore.case=TRUE))
```

```
##     stockid     tsn scientificname   commonname      areaid
## 216   CODGB  164712   Gadus morhua Atlantic cod USA-NMFS-5Z
## 217  CODGOM  164712   Gadus morhua Atlantic cod USA-NMFS-5Y
##                        stocklong        region primary_country primary_FAOarea
## 216  Atlantic cod Georges Bank US East Coast             USA              21
## 217 Atlantic cod Gulf of Maine US East Coast             USA              21
##     ISO3_code                            GRSF_uuid inmyersdb myersstockid
## 216       USA 71c5f2e5-beaf-3b36-afc7-baf16f0080b7         1        COD5Z
## 217       USA 0fab3985-c6c0-3851-a1c0-12945c1c18aa         0         <NA>
##       state
## 216 Current
## 217 Current
```

```
subset(stock_US_East,grepl("herring",commonname,ignore.case=TRUE))
```

```
##        stockid     tsn  scientificname commonname          areaid
## 484 HERRNWATLC  161722 Clupea harengus    Herring USA-NMFS-NWATLC
##                                stocklong        region primary_country
## 484 Herring Northwestern Atlantic Coast US East Coast             USA
##     primary_FAOarea ISO3_code                            GRSF_uuid inmyersdb
## 484              21       USA fec91721-2a5d-3e06-bb54-d3d191603e08         0
##     myersstockid   state
## 484         <NA> Current
```

We can save just the `stockid` column, which we can use to extract data on these stocks from other objects loaded from the databse.

```
list_stock_id <- subset(stock_US_East,grepl("cod",commonname,ignore.case=TRUE)|grepl("herring",commonnam
```

```
data_LAB_6 <- subset(timeseries,stockid %in% list_stock_id)
```

This is a lot of data rows. In particular there are a large number of different parameters. We can use the `unique` command to get a better idea of what we have.

```
unique(data_LAB_6$tsid)
```

```
##  [1] "BdivBmgtpref-dimensionless"      "BdivBmsypref-dimensionless"
##  [3] "BdivBmsytouse-dimensionless"     "CdivMEANC-ratio"
##  [5] "CdivMSY-ratio"                   "ER-ratio"
##  [7] "ERbest-ratio"                    "F-1/yr"
```

```
##  [9] "FdivFmsy-calc-dimensionless"     "R-E00"
## [11] "SSB-MT"                          "SSBdivSSBmsy-calc-dimensionless"
## [13] "TC-MT"                           "TCbest-MT"
## [15] "UdivUmgtpref-dimensionless"      "UdivUmsypref-dimensionless"
## [17] "UdivUmsytouse-dimensionless"     "survB-index"
## [19] "BdivBmgttouse-dimensionless"     "DiscC-MT"
## [21] "FdivFmgt-calc-dimensionless"     "RecC-MT"
## [23] "SSBdivSSBmgt-calc-dimensionless" "TL-MT"
## [25] "UdivUmgttouse-dimensionless"     "ER-calc-ratio"
## [27] "TB-MT"                           "TBbest-MT"
## [29] "R-MT"
```

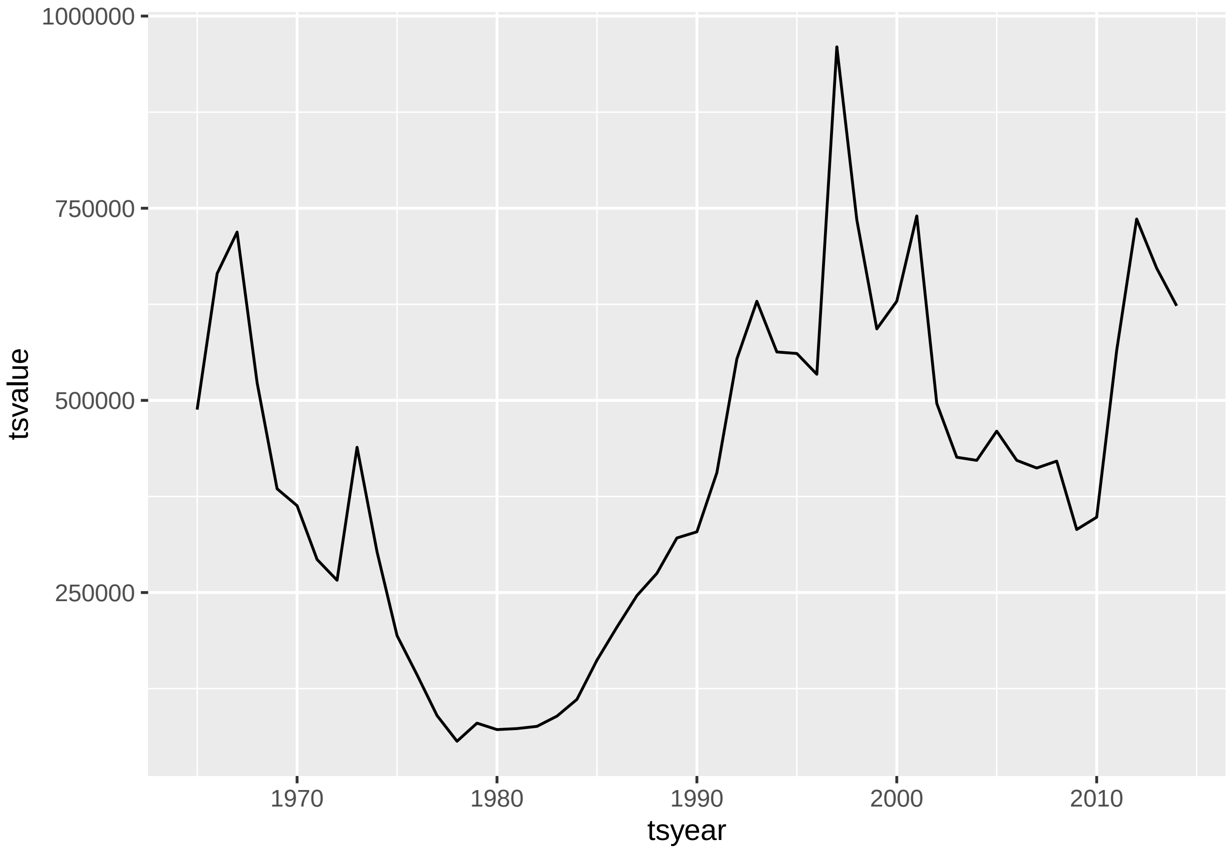
Note there's a variable "R-MT" and a variable "SSB-MT"; these are time series of recruitment and spawning-stock biomass (in metric tonnes).

```
data_LAB_6_R <- subset(data_LAB_6,tsid=="R-E00")
data_LAB_6_SSB <- subset(data_LAB_6,tsid=="SSB-MT")
data_LAB_6_TC <- subset(data_LAB_6,tsid=="TC-MT")
```

EDM analysis for a single time series

That's time series for 3 variables (recruitment, spawning stock biomass, and total catch) of 3 stocks (2 cod, 1 herring). Let's start with just one, the SSB of herring. Note that we really only need the year and value for EDM analysis. HOWEVER, the Ram Myers database is very comprehensive, and we need to pay attention to another column, "assessid". There are actually two timeseries of SSB for the same stockk, "NEFSC-HERRNWATLC-1964-2011-HIVELY" and "NEFSC-HERRNWATLC-1965-2014-SISIMP2016".

```
df_EDM_try <- subset(data_LAB_6_SSB,stockid=="HERRNWATLC" & assessid =="NEFSC-HERRNWATLC-1965-2014-SISI

ggplot(df_EDM_try,aes(x=tsyear,y=tsvalue)) + geom_line()
```

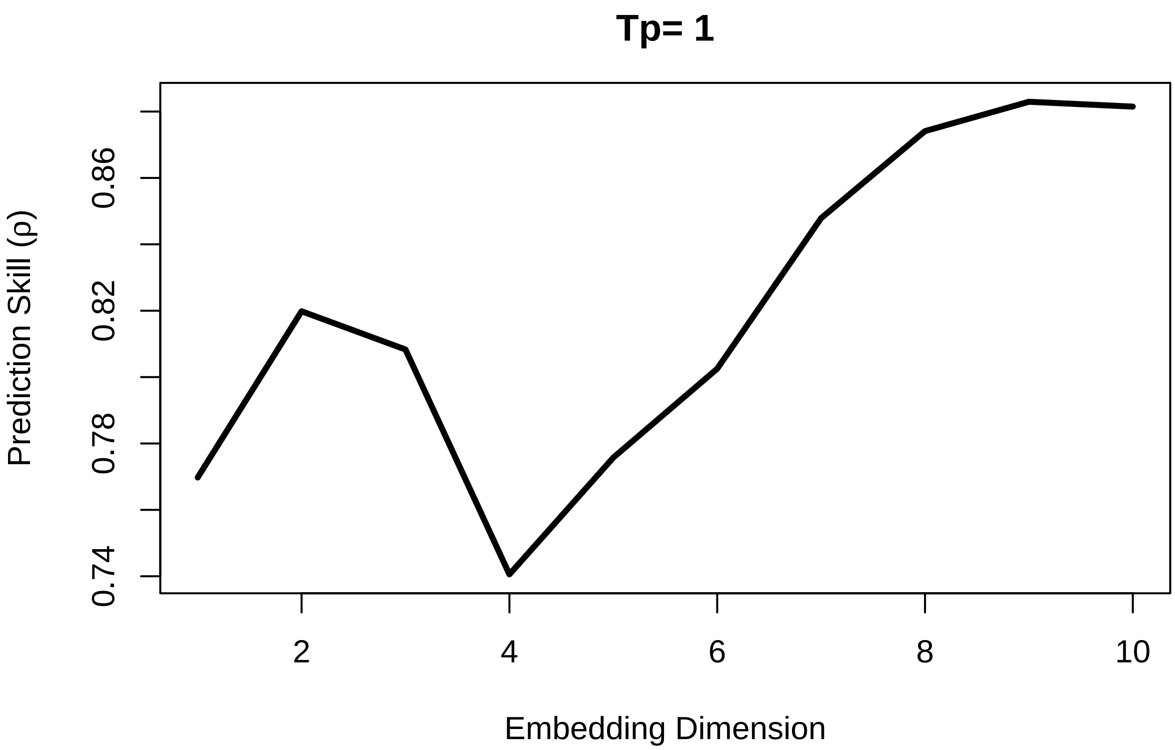


Simplex

```
library('rEDM')
```

Simplex projection is used to assess the basic predictability of the time series (ala Sugihara & May 1990), and also to estimate a functional embedding dimension for prediction.

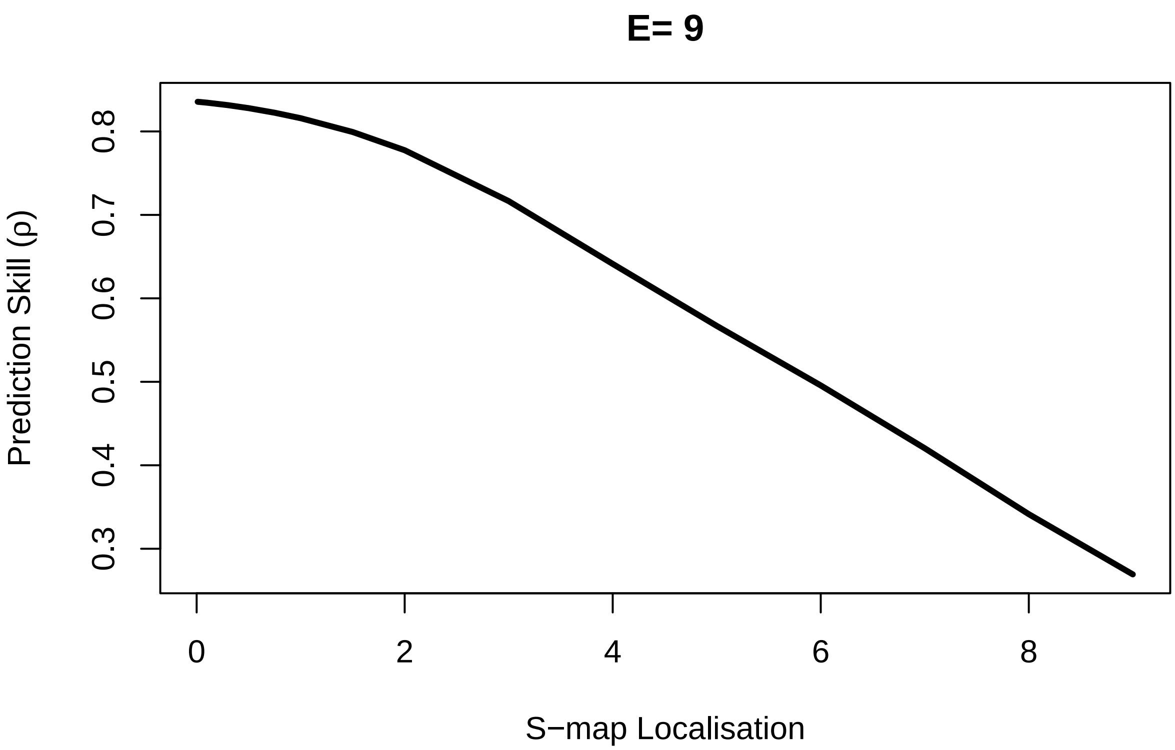
```
out_simplex_EDM_try <- EmbedDimension(dataFrame=df_EDM_try,lib="1 50",pred="1 50",columns="tsvalue",targ
```



The predictability of this time series certainly seems high (prediction skill here being measured as Pearson's correlation between observed and predicted values); the highest prediction skill is also at `E = 9`. That is, if we use 9 time-lag coordinates, then nearest neighbor predictions show the highest skill with leave-one-out cross-validation. We can use this values of `E`, then, to look at nonlinearity with sequentially weighted local linear maps (S-maps).

S-map

```
out_smap_EDM_try <- PredictNonlinear(dataFrame=df_EDM_try,lib="1 50",pred="1 50",columns="tsvalue",targ
```



Here, increasing nonlienar parameter `theta` appears to degrade rather than improve forecast skill. This suggests that the predictive skill we see is due to underlying linear or equilibrium dynamics rather than non-equilibrium dynamics. Of course, recall what Storch et al. and Giron-Nava et al. found about evidence of nonlinearity in the outputs of stock assessments!

Also, note that these functions automatically generate plots. This is handy for exploring time-series by time-series, but can be a problem if you're repeating analysis over many functions at once. To suppress the plot, set the argument `showPlot` to `FALSE`.

```
out_smap_EDM_try <- PredictNonlinear(dataFrame=df_EDM_try,lib="1 50",pred="1 50",columns="tsvalue",targ
```

EDM anlysis of multiple time series

TASK: Analyze stock, total catch, and recruitment data for a single stock.

Surrogate analysis

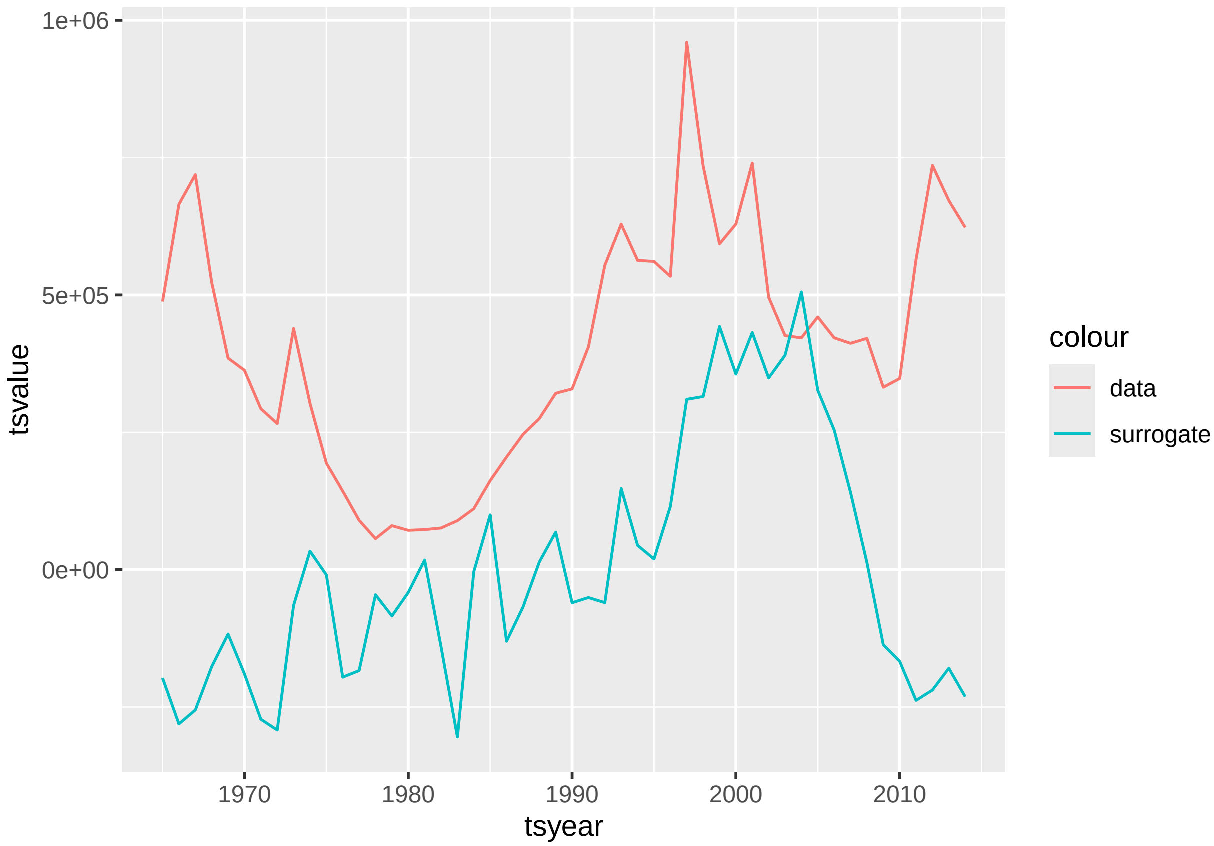
Often in empirical dynamic modeling studies, forecast skill is measured as Pearson's correlation between observed and predicted values. The correlation coefficient has values associated with it based on Fisher's Z transform. However, these values are based on independence between each pair. That's a pretty big assumption for studying dynamics (which has a lot to do with how time series values depend on each other as a sequence through time).

For this reason, more conservative significance estimates are often made using "surrogate methods". This is similar to how we talked about empirical statistical analysis in Lab 1 with bootstrap resampling or repeated simulation. Two types of surrogates have been used for univariate EDM analysis to address how autocorrelation in otherwise random time series can produce higher predictability than just uncorrelated noise. One is a phase-randomization procedure (Ebisuzaki 1999) and the other is an AR(1) procedure.

The phase-randomization procedure is included in the package under the function `SurrogateData`. In the tutorial, the example of surrogates use the `"seasonal"` option, but the syntax is otherwise the same.

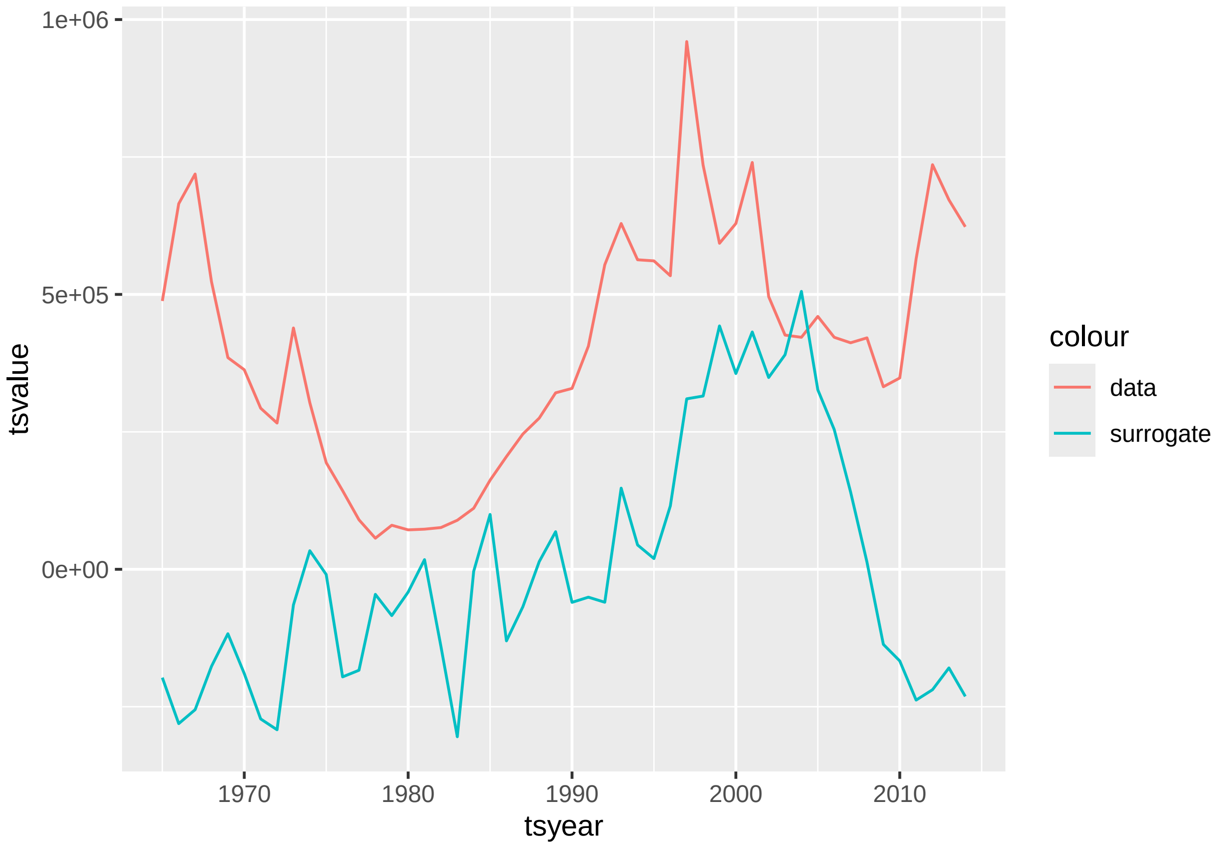
Let's try generating a single surrogate time series with the Ebisuzaki method and plot to see what's going on.

```
set.seed(1773)
ts_surr_test <- SurrogateData(df_EDM_try$tsvalue,method="ebisuzaki",1)
df_surr_test <- data.frame(tsyear=df_EDM_try$tsyear,tssurr=ts_surr_test)

ggplot(df_EDM_try,aes(x=tsyear,y=tsvalue)) +
  geom_line(aes(color="data")) +
  geom_line(data=df_surr_test,aes(y=tssurr,color="surrogate"))
```



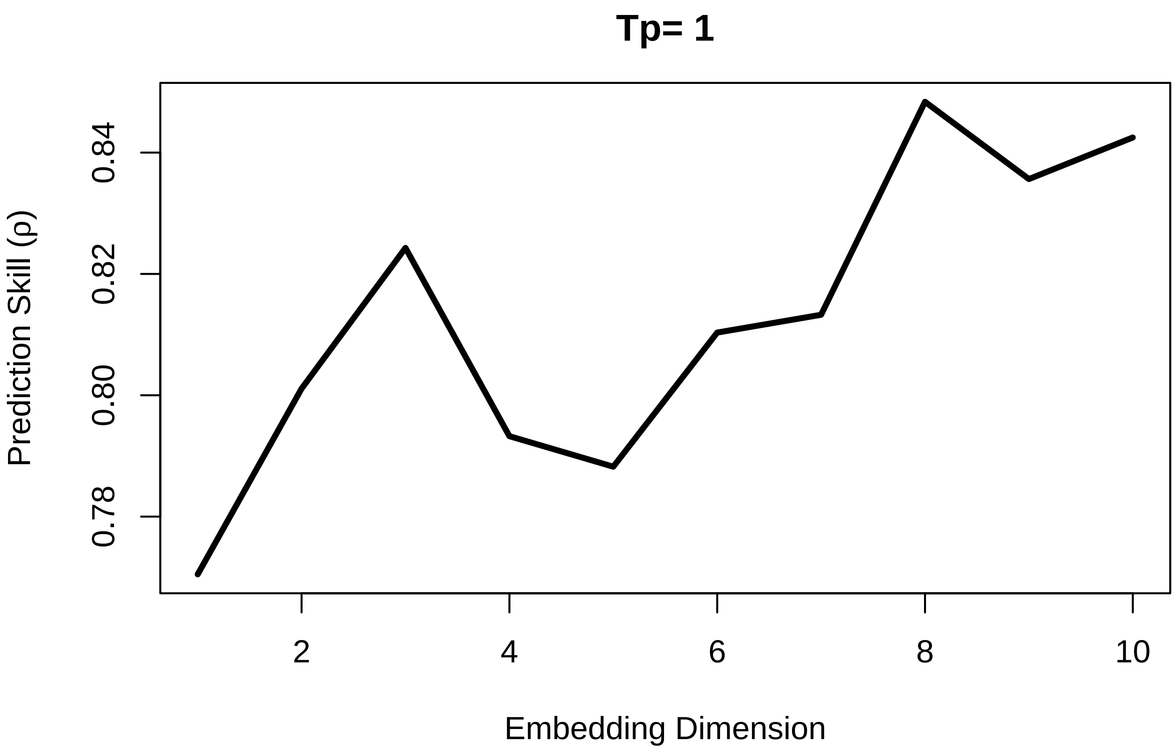
QUESTION: What happens when you run this code again? Did you get the same plot? What's the deal with that? For the purposes of illustration, I'll fix the RNG here and run again.

```
set.seed(1773)
ts_surr_test <- SurrogateData(df_EDM_try$tsvalue,method="ebisuzaki",1)
df_surr_test <- data.frame(tsyear=df_EDM_try$tsyear,tssurr=ts_surr_test)

ggplot(df_EDM_try,aes(x=tsyear,y=tsvalue)) +
  geom_line(aes(color="data")) +
  geom_line(data=df_surr_test,aes(y=tssurr,color="surrogate"))
```



We can apply the same analysis above to these types of surrogate time series. They give us a null expectation of the EDM tests under random, correlated data.

```
EmbedDimension(dataFrame=df_surr_test,lib="1 50",pred="1 50",target="tssurr",columns="tssurr")
```




```
##     E       rho
## 1   1 0.7704784
## 2   2 0.8010961
## 3   3 0.8242936
## 4   4 0.7932556
## 5   5 0.7882261
## 6   6 0.8103371
## 7   7 0.8132672
## 8   8 0.8483749
## 9   9 0.8356371
## 10 10 0.8424913
```

```
# EmbedDimension(dataFrame=df_EDM_try,lib="1 50",pred="1 50",columns="tsvalue",target="tsvalue")
```

This time series has had it Fourier spectrum preserved but the dynamics otherwise destroyed. It shows pretty comparable predictability to the test on the original data. This suggests that the prediction skill we measured is not significantly different from the null expectation of simple periodic dynamics in a time series. Does this mean the time series isn't predictable, though? No. It just means we should be careful to attribute the predictability to underlying nonlinear dynamics. At the same time, we are only visually comparing a two plots. For a formal test, we need to generate the a distribution of test statistics (in this case, the maximum forecast skill from simplex-projection for embedding dimension E=1-10.). This means a `for` loop! Each time in the loop you will need to generate a new realization of the surrogate test.

OPTIONAL TASK: Generate surrogate results for simplex-projection forecast skill for one or more Ram Legacy Database time series. Compare the significance ("p-value") based on the surrogate analysis to the significance based on Fisher's Z transform.

> Plot the distribution of surrogate values for prediction skill and compare to the measurement of the real data. What's the p-value under the surrogate test?

Reflection questions:

Several of our readings have discussed effects of stock-assessment analysis on the dynamics of time series that come out (Storch et al.; Giron-Nava et al.), and several other papers have made arguments about basic ecology or fishing science based on quantitative analysis of these assessment-derived "data". Discuss your results in light of these papers. Is what you found surprising or expected?