COVID-19

COVID-19 chaos model

Chaos model for COVID-19 EU data.

Get data

Get data from European Center for Disease Control (ECDC).

```
data <- read.csv("https://opendata.ecdc.europa.eu/covid19/casedistribution/csv",
                 na.strings = "", fileEncoding = "UTF-8-BOM", stringsAsFactors = F)

# construct "date_reported" <date> column
data$date_reported <- mdy(paste0(data$month,"-",data$day,"-",data$year))
```

Isolate data from Europe, from countries with population at least a million, and select relevant columns only.

```
eu <- data[data$continentExp == "Europe" & data$popData2019 > 1000000,] %>%
  select("date_reported", "cases", "deaths", "countriesAndTerritories", "popData2019")

glimpse(eu)
```

```
## Rows: 12,726
## Columns: 5
## $ date_reported           <date> 2020-12-08, 2020-12-07, 2020-12-06, 2020-1...
## $ cases                   <int> 695, 840, 846, 801, 782, 705, 832, 557, 835...
## $ deaths                  <int> 17, 16, 19, 18, 13, 17, 12, 12, 11, 16, 18,...
## $ countriesAndTerritories <chr> "Albania", "Albania", "Albania", "Albania",...
## $ popData2019             <int> 2862427, 2862427, 2862427, 2862427, 2862427...
```

Transform into timeseries

Get cases and deaths logdiffs per country.

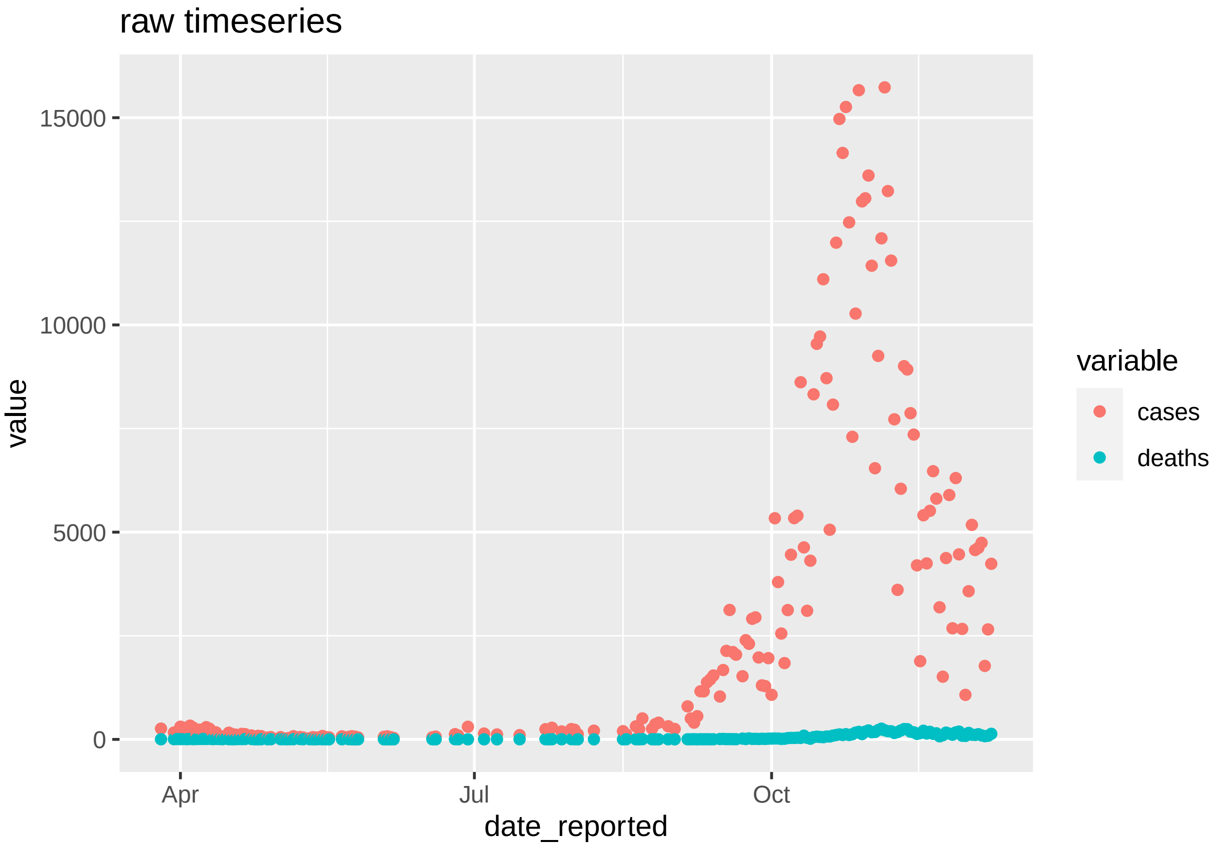
```
isValid <- function(x) {
  !is.nan(x) & is.finite(x) & x != 0
}

euClean <- eu %>%
  arrange(date_reported) %>%
  group_by(countriesAndTerritories) %>%
  mutate(
    cases_growth = cases / lag(cases),
    deaths_growth = deaths / lag(deaths),
    cases_logdiff = log(cases) - log(lag(cases)),
    deaths_logdiff = log(deaths) - log(lag(deaths))) %>%
  tail(-1) %>%
  filter(isValid(cases_logdiff),
         isValid(deaths_logdiff),
         isValid(cases_growth),
         isValid(deaths_growth)) %>%
  group_split()
names(euClean) = unique(eu$countriesAndTerritories)

euWeeklyClean <- eu %>%
  arrange(date_reported) %>%
  group_by(countriesAndTerritories, week = isoweek(date_reported)) %>%
  summarize(cases = sum(cases), deaths = sum(deaths)) %>%
  mutate(
    cases_growth = cases / lag(cases),
    deaths_growth = deaths / lag(deaths)) %>%
  filter(isValid(cases_growth),
         isValid(deaths_growth)) %>%
  group_split()
names(euWeeklyClean) = unique(eu$countriesAndTerritories)
```

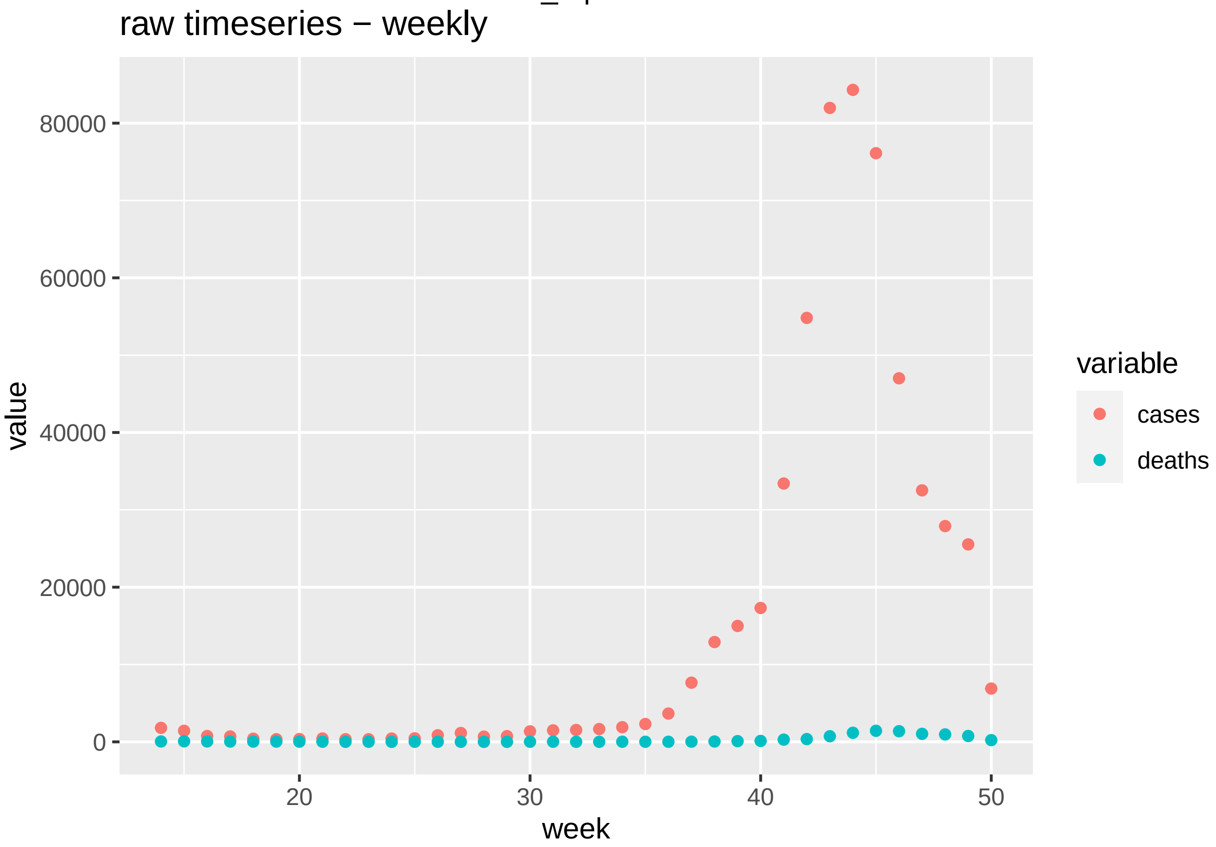
Pick a country for further analysis.

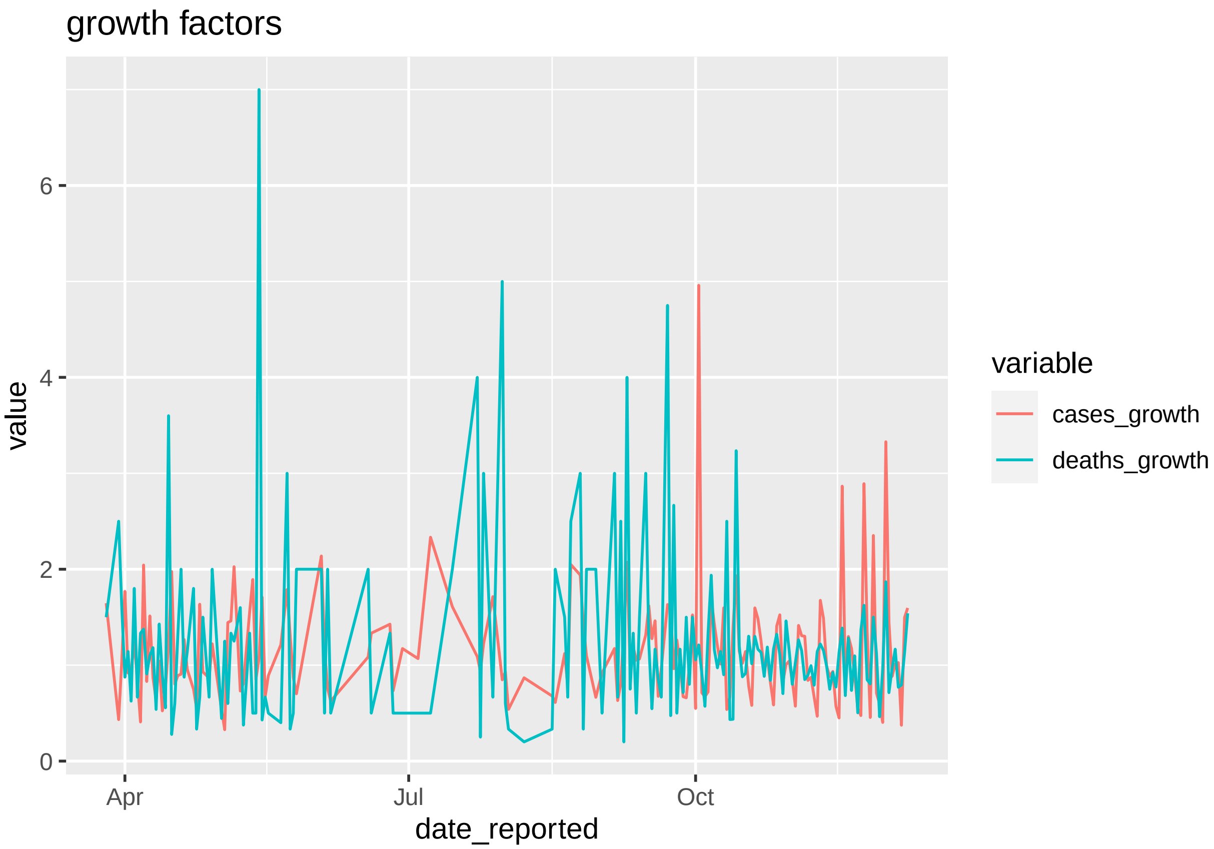
```
country <- euClean$Czechia
countryWeekly <- euWeeklyClean$Czechia

glimpse(country)
```

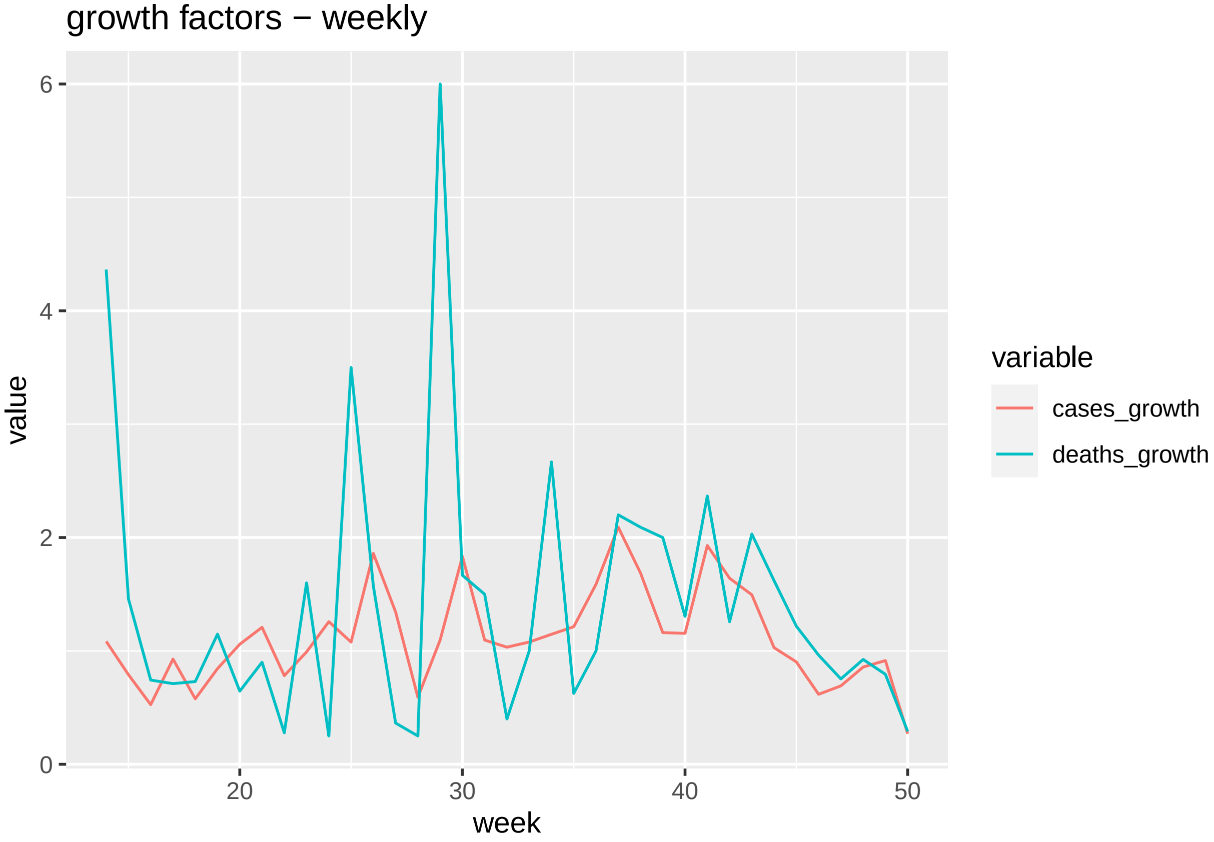
```
## Rows: 172
## Columns: 9
## $ date_reported           <date> 2020-03-26, 2020-03-30, 2020-03-31, 2020-0...
## $ cases                   <int> 260, 166, 173, 306, 281, 269, 332, 282, 115...
## $ deaths                  <int> 3, 5, 8, 7, 8, 5, 9, 6, 8, 11, 10, 11, 13, ...
## $ countriesAndTerritories <chr> "Czechia", "Czechia", "Czechia", "Czechia",...
## $ popData2019             <int> 10649800, 10649800, 10649800, 10649800, 106...
## $ cases_growth            <dbl> 1.6455696, 0.4322917, 1.0421687, 1.7687861,...
## $ deaths_growth           <dbl> 1.5000000, 2.5000000, 1.6000000, 0.8750000,...
## $ cases_logdiff           <dbl> 0.49808660, -0.83865476, 0.04130381, 0.5702...
## $ deaths_logdiff          <dbl> 0.40546511, 0.91629073, 0.47000363, -0.1335...
```

Plot


raw timeseries
value
15000
10000
5000
0
Apr
Jul
Oct
date_reported
variable
● cases
● deaths


raw timeseries − weekly
value
80000
60000
40000
20000
0
20
30
40
50
week
variable
● cases
● deaths


growth factors
value
6
4
2
0
Apr
Jul
Oct
date_reported
variable
cases_growth
deaths_growth

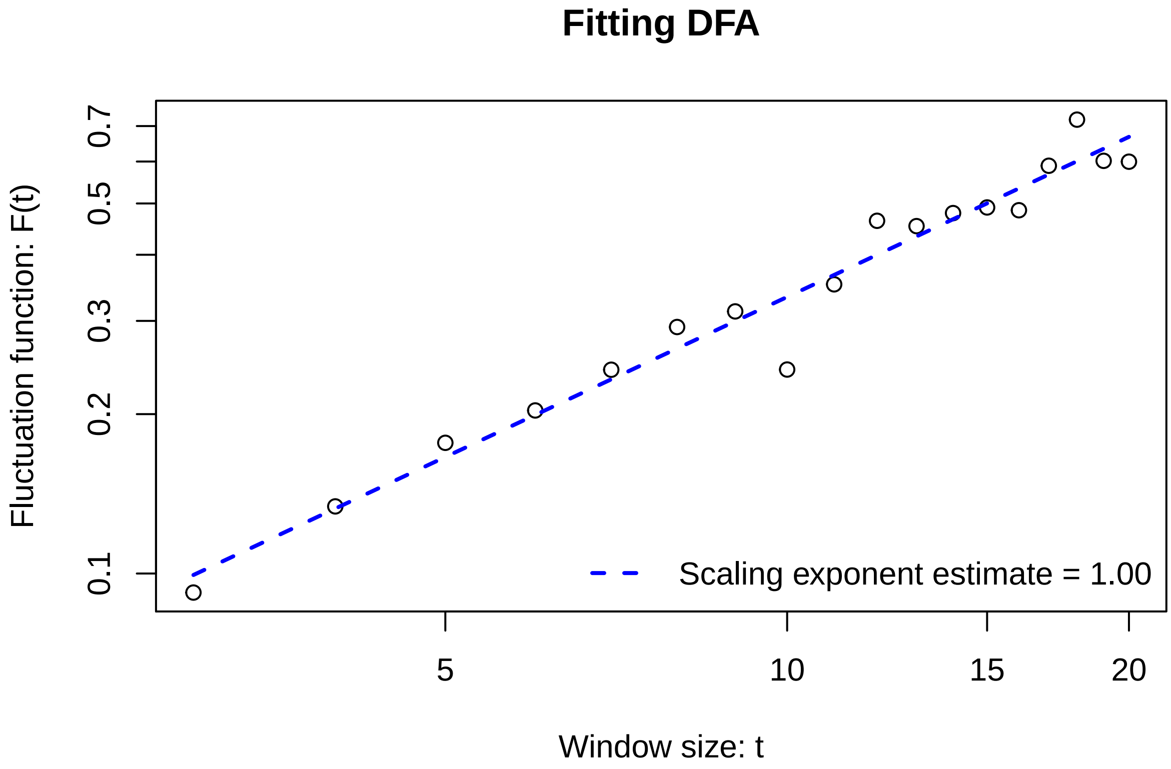

growth factors − weekly
value
6
4
2
0
20
30
40
50
week
variable
cases_growth
deaths_growth

Reconstruct phase space

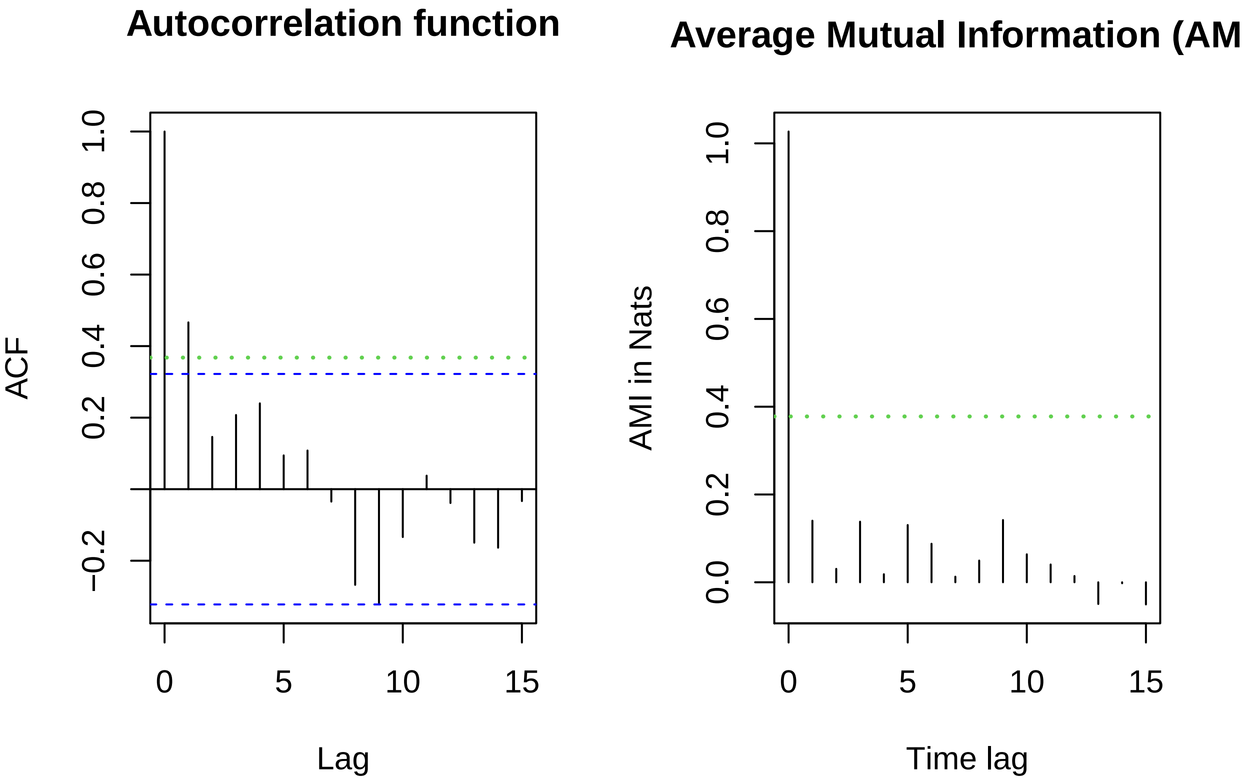
Taken's embedding theorem.

Use weekly sums to avoid weekend fluctuations.

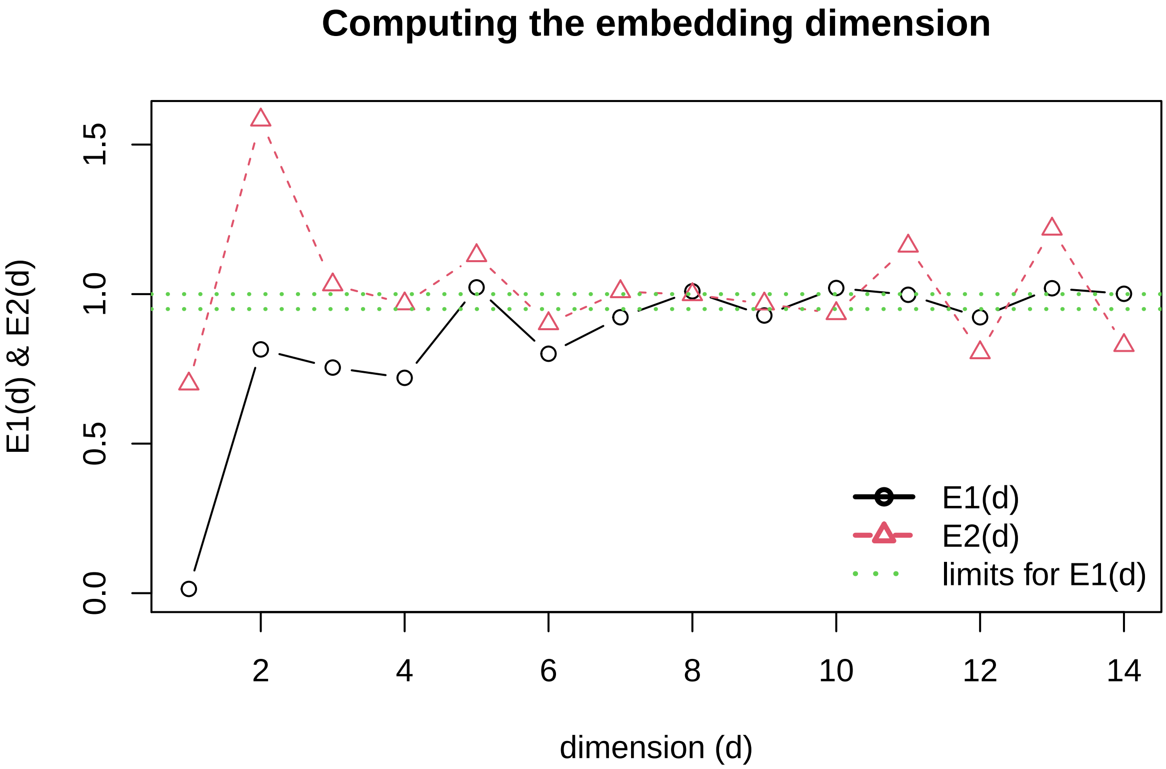
```
ts <- as.ts(zoo(countryWeekly$cases_growth, order.by = countryWeekly$week))
ts <- na.contiguous(ts)

dfa.analysis = dfa(time.series = ts, npoints = 30,
                   window.size.range=c(3,20),
                   do.plot=FALSE)
ts.estimation = estimate(dfa.analysis, do.plot = TRUE,
                         fit.col="blue",fit.lwd=2,fit.lty=2,
                         main="Fitting DFA")
```



```
par(mfrow = c(1, 2))
# tau-delay estimation based on the autocorrelation function
tau.acf = timeLag(ts, technique = "acf", lag.max = 15, do.plot = T)
# tau-delay estimation based on the mutual information function
tau.ami = timeLag(ts, technique = "ami", lag.max = 15, do.plot = T)
```



```
emb.dim = estimateEmbeddingDim(ts, time.lag = tau.ami, max.embedding.dim = 15)
```



```
tak = buildTakens(ts, embedding.dim = emb.dim, time.lag = tau.ami)

scatter3D(tak[,1], tak[,2], tak[,3],
          col = "red", type="o",cex = 0.3)
```