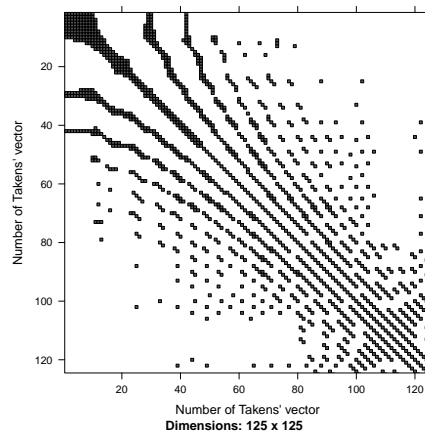


STATISTICAL DATA ANALYSIS: RECURRENCE PLOT

GÜNTHER SAWITZKI



CONTENTS

1. Setup	1
2. Test Cases	2
3. Recurrence States	4
4. Recurrence Plots	6
4.1. Sinus	7
4.2. Uniform random	7
4.3. Chirp Signal	9
5. Case Study: Geyser data	10
5.1. Geyser Eruptions	10
5.2. Geyser Eruptions: Comparison by Dimension	15
5.3. Geyser Waiting	15
6. Case Study: HRV data	16
6.1. RHRV: Comparison by Dimension	20
References	24
Index	25

1. SETUP

Input

Date: 2013-11.

Key words and phrases. data analysis, distribution diagnostics, recurrence plot.

This waste book is a companion to “G. Sawitzki: Statistical Data Analysis”

Typeset, with minor revisions: February 5, 2014 from cvs *Revision* : 1.2

gs@statlab.uni-heidelberg.de .

```
save.RNGseed <- 87149 #.Random.seed
save.RNGkind <- RNGkind()
# save.RNGseed
save.RNGkind
```

Output

```
[1] "Mersenne-Twister" "Inversion"
```

Input

```
set.seed(save.RNGseed, save.RNGkind[1])
```

Input

```
laptime <- function(){
  return(round(structure(proc.time() - chunk.time.start, class = "proc_time")[3],3))
  chunk.time.start <- proc.time()
}
```

Input

```
# install.packages("sintro",repos="http://r-forge.r-project.org",type="source")
library(sintro)
```

We use

Input

```
library(nonlinearTseries)
```

To display state space, we use a variant of pairs().

Input

```
statepairs <- function(states, rank=FALSE){
  main <- paste("Takens states:",deparse(substitute(states)))
  if (rank) {states <- apply(unifrank,2,rank,ties.method="random")}
  main <- paste(main," ranked")}
  pairs(states,
    main=main,
    col=rgb(0,0,0,0.2))
}
```

2. TEST CASES

We set up a small series of test signals.

For convenience, some source code from other libraries is included to make this self-contained.

As a global constant, we set up the length of the series to be used.

Input

```
nsignal <- 128
system.time.start <- proc.time()
```

For representation, we use a common layout.

Input

```
plotsignal <- function (signal) {
  par(mfrow=c(1,2))
  plot(signal, col="blue", pch=20, xlab="t" )

  plot(signal, type="l",
        main=deparse(substitute(signal)), xlab="t")
  points(signal, col="blue", pch=20 )
}
```

Input

```
sin10 <- function(n=nsignal) {sin( (1:n)/n* 2*pi*10)}
plotsignal(sin10())
```

See Figure 1,

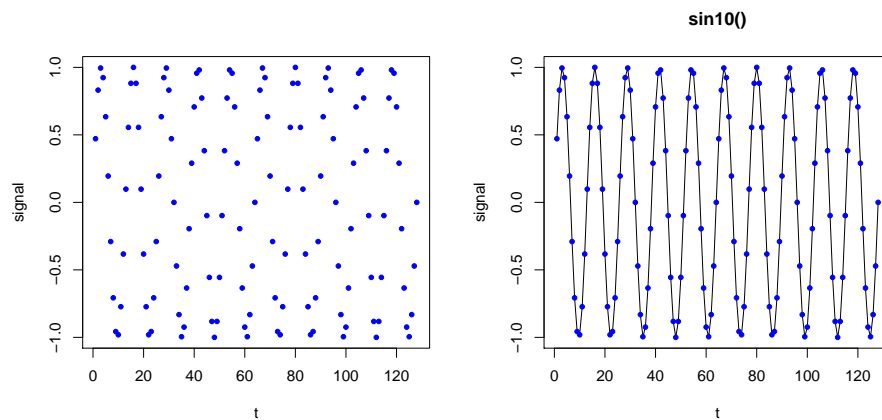


FIGURE 1. Test case: sin10. Signal and linear interpolation.

Input

```
unif <- function(n=nsignal) {runif(n)}
xunif<-unif()
plotsignal(xunif)
```

See Figure 2 on the next page,

Input

```
chirp <- function(n=nsignal) {
  # this is copied from library(signal)
  signal.chirp <- function(t, f0 = 0, t1 = 1, f1 = 100,
                           form = c("linear", "quadratic", "logarithmic"), phase = 0){

    form <- match.arg(form)
    phase <- 2*pi*phase/360

    switch(form,
           "linear" = {
```

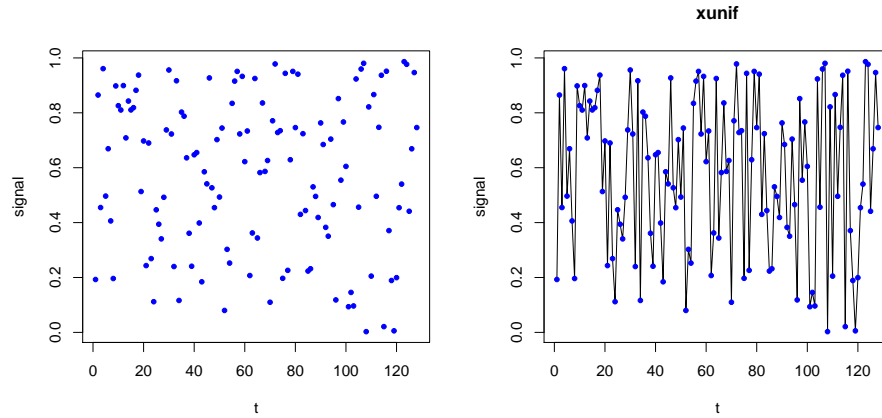


FIGURE 2. Test case: unif - uniform random numbers. Signal and linear interpolation.

```

a <- pi*(f1 - f0)/t1
b <- 2*pi*f0
cos(a*t^2 + b*t + phase)
},
"quadratic" = {
a <- (2/3*pi*(f1-f0)/t1/t1)
b <- 2*pi*f0
cos(a*t^3 + b*t + phase)
},
"logarithmic" = {
a <- 2*pi * t1 / log(f1 - f0)
b <- 2*pi * f0
x <- (f1-f0)^(1/t1)
cos(a*x^t + b*t + phase)
})
}

signal.chirp(seq(0, 0.6, len=nsignal))
}
plotsignal(chirp())

```

ToDo: include See Figure 3 on the facing page,
doppler waveslim

3. RECURRENCE STATES

Recurrence plots have been introduced in an attempt to understand near periodic in hydrodynamics. On the one hand, and extended theory on dynamical systems was available, covering deterministic models. A fundamental concept is that at a certain time a system is in some state, and developing from this. Defining the proper state space is a critical step in modelling.

The other toolkit is that of stochastic processes, in particular Markov models. Classical time series assumes stationarity, and this is obviously not the way to go. A fundamental idea for Markov models is that the system state is seen in a temporal context: you have a Markov process, if you can define a (non-anticipating) state that has sufficient information for prediction: given this state, the future is independent from the past.

Recurrence, coming back to some state, is often a key to understand a near periodic system.

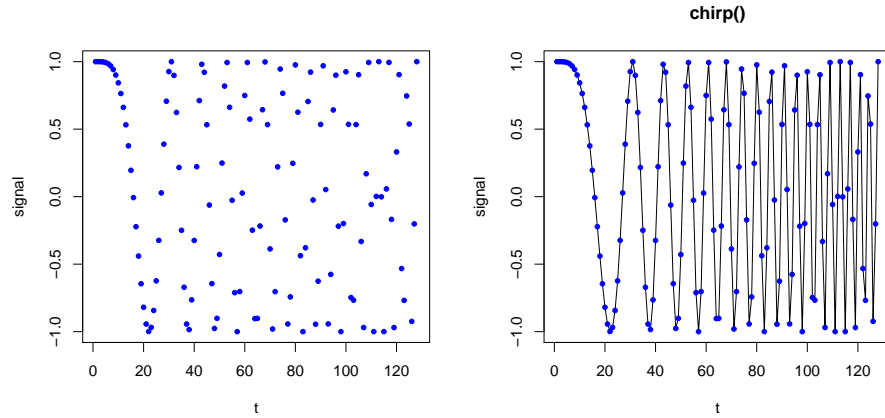


FIGURE 3. Test case: chirp signal. Signal and linear interpolation.

Hydrodynamics is a challenging problem. Understanding planetary motion is a historical challenge, and may be useful as an illustration.

As a simple illustration, let $x = (x_i)$ be a sequence, maybe near periodic. For now, think of i as a time index.

Recurrency plots have two steps. The first was a bold step by Floris Takens. If you do not know the state space of a system, for a choice of “dimension” d , take the sequence of d tuples taken from your data to define the states.

$$u_i = (x_i, \dots, x_{i+d})$$

As a mere technical refinement: you may know that your data are a flattened representation of t dimensional data. So you take

$$u_i = (x_i, \dots, x_{i+d*m}).$$

We ignore this detail here and take $m = 1$.

Conceptually, you define states by observed histories. For classical Markov setup, the state is defined by the previous information x_{i-1} , but for more complex situations you may have to step back in the past. Finding the appropriate d is the challenge. So it may be appropriate to view the Takens states as a family, indexed by the time scope d . The rest is structural information how to arrange items.

Of course it is possible to compress information here, sorting states and removing duplicates. Keeping the original definition as the advantage that we have the index i , so that u_i is the state at index position i .

But the states may have an inherent structure, which we may take into account or ignore. Since for this example, we are just in 4-dimensional space, marginal scatterplots may give enough information.

Input

```
sintakens <- buildTakens( time.series=sin10(), embedding.dim=4, time.lag=1)
statepairs(sintakens)
```

See Figure 4 on the next page.

Input

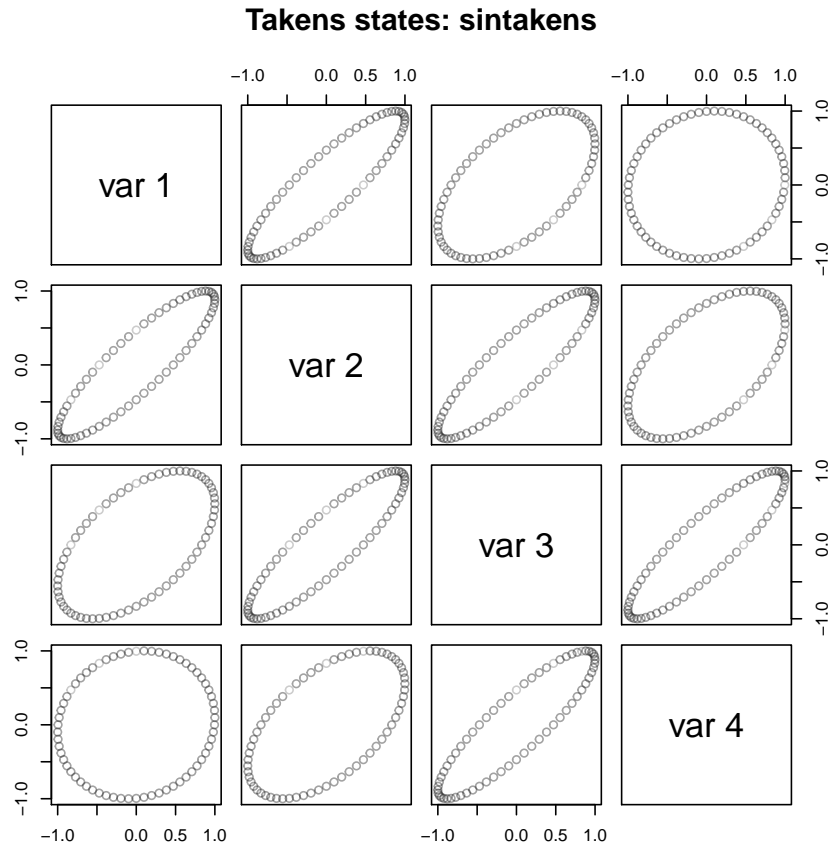


FIGURE 4. Test case: sinus. Note that marginal views of 1-dimensional circles in d space may appear as ellipses. Time used: 0.199 sec.

```
uniftakens <- buildTakens( time.series=xunif,embedding.dim=4,time.lag=1)
statepairs(uniftakens)
```

See Figure 5 on the facing page.

```
statepairs(chirptakens) Input
```

See Figure 6 on page 8

4. RECURRENCE PLOTS

The next step, taken in Eckmann *et al.* [1987] was to use a two dimensional display. Take a scatterplot with the Taken's staes a marginal. Take a sliding window of your process data, and for each i , find the “distance” of u_i from and to any of the collected states. If the distance is below some chosen threshold, mark the point (i, j) for which $u(j)$ is in the ball of radius $r(i)$ centred at $u(i)$.

The original publication Eckmann *et al.* [1987] actually used a nearest neighbourhood environment to cover about 10 data points.

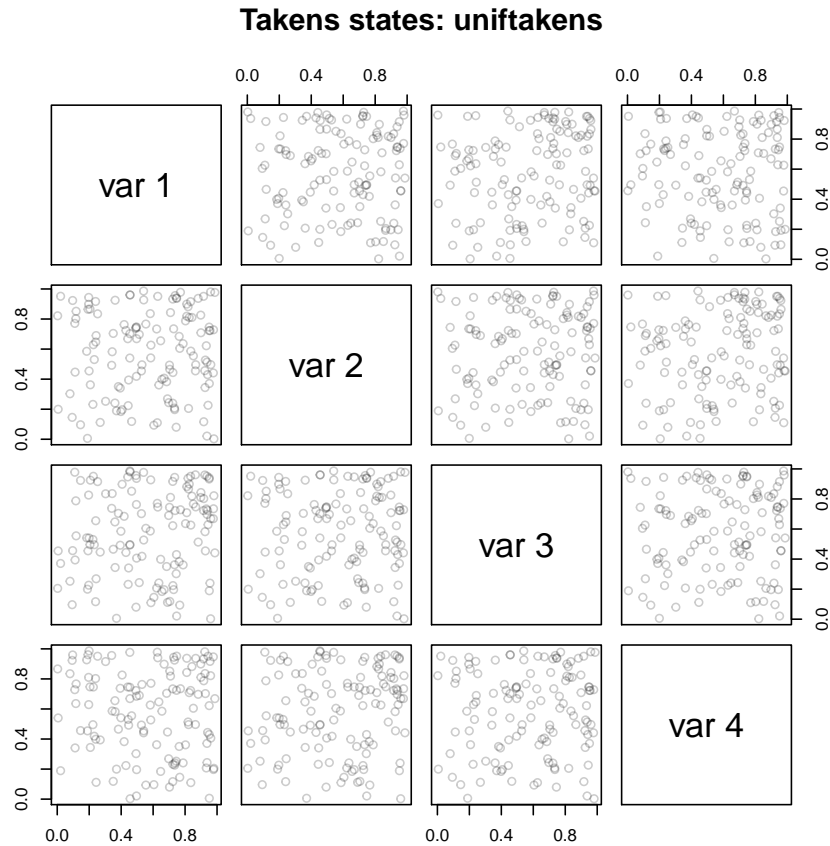


FIGURE 5. Test case: uniform random numbers. Time used: 0.203 sec.

The construction has considerable arbitrary choices. The critical radius may depend on the point i . In practical applications, using a constant radius is a common first step. Using a dichotomous marking was what presumably was necessary when the idea was introduced. With today's technology, we can allow a markup on a finer scale, as has been seen in Orion-1.

We can gain additional freedom by using a correlation view: instead of looking from one axis, we can walk along the diagonal, using two reference axis.

4.1. Sinus.

Input

```
load(file="sin10neighs.RData")
nonlinearTseries::recurrencePlotAux(sin10neighs)
```

4.2. Uniform random.

Input

```
load(file="unifneighs.RData")
nonlinearTseries::recurrencePlotAux(unifneighs)
```

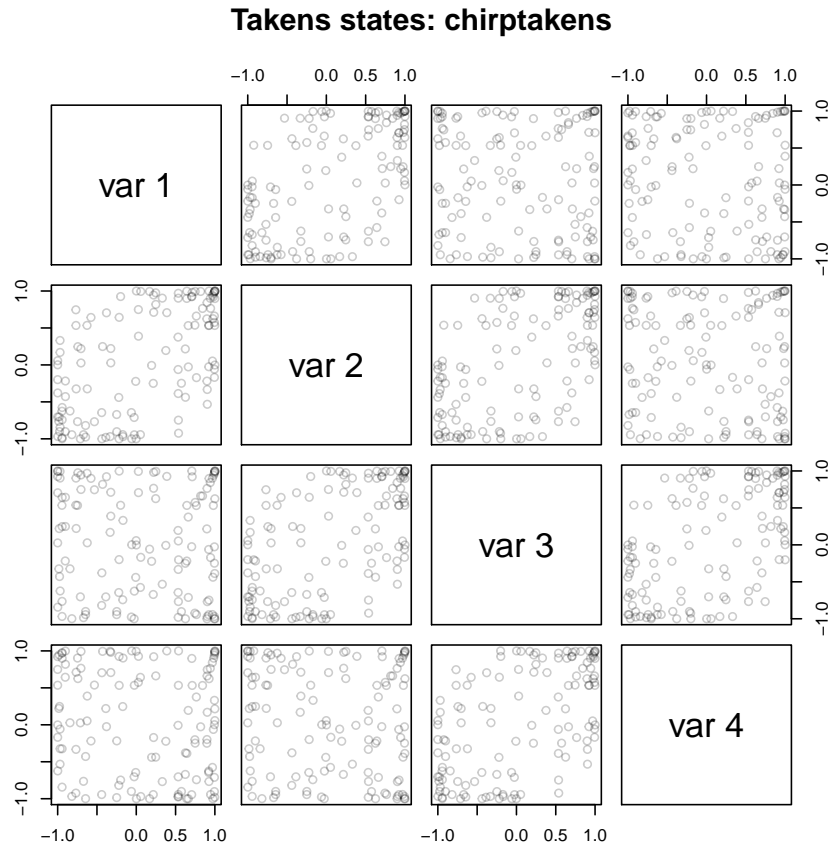


FIGURE 6. Test case: chirp signal. Time used: 0.186 sec.

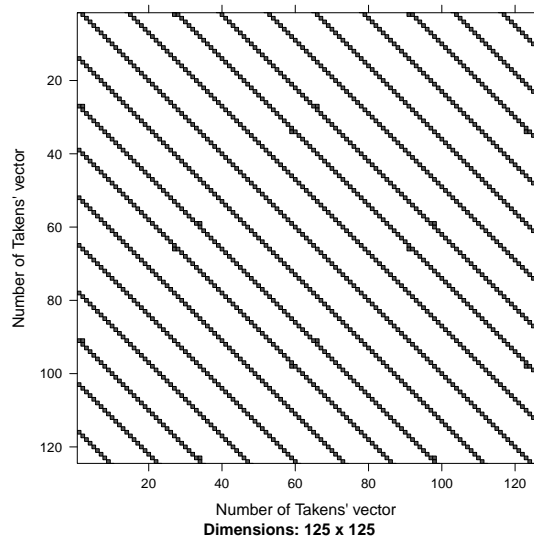


FIGURE 7. Recurrence Plot. Test case: sinus curves. Time used: 4.443 sec.

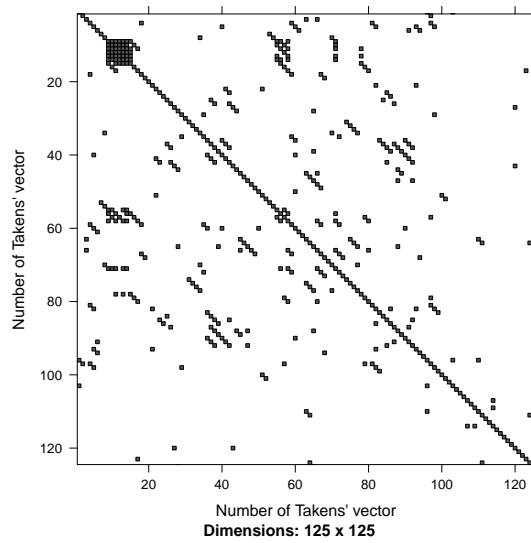


FIGURE 8. Recurrence Plot. Test case: uniform random numbers. Time used: 1.568 sec.

4.3. Chirp Signal.

Input

```
chirpneighs<-nonlinearTseries::findAllNeighbours(chirptakens,radius=0.6)#0.4
save(chirpneighs, file="chirpneighs.RData")
```

Input

```
load(file="chirpneighs.RData")
nonlinearTseries::recurrencePlotAux(chirpneighs)
```

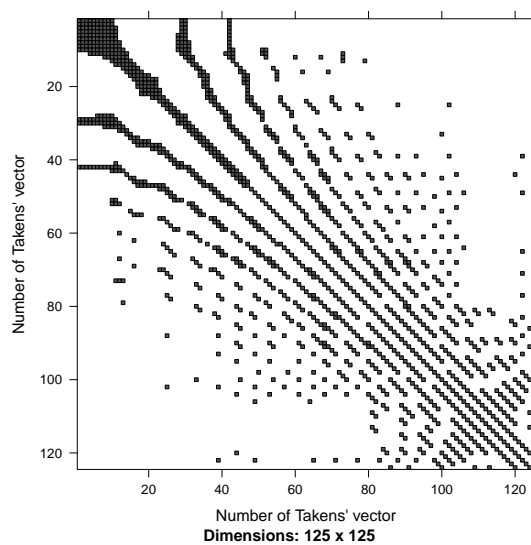


FIGURE 9. Recurrence Plot. Test case: chirp signal. Time used: 5.255 sec.

5. CASE STUDY: GEYSER DATA

ToDo: extend to This is a classical data set with a two dimensional structure, *waiting* and *waiting*.
two-dimensional data

Input

```
library(MASS)
data(faithful)
```

5.1. Geyser Eruptions.

Input

```
plotsignal(faithful$eruptions)
```

See Figure 10,

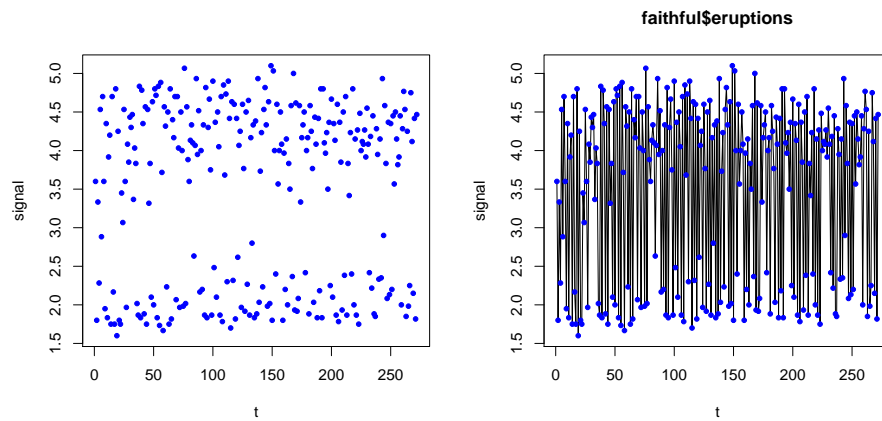


FIGURE 10. Example case: Old Faithful Geyser eruptions. Signal and linear interpolation.

Input

```
eruptionstakens4 <- buildTakens( time.series=faithful$eruptions, embedding.dim=4, time.lag=1)
statepairs(eruptionstakens4)
```

See Figure 11 on the next page

Input

```
eruptionsneighs4<-nonlinearTseries::findAllNeighbours(eruptionstakens4, radius=0.8)
save(eruptionsneighs, file="eruptionsneighs4.RData")
```

Input

```
load(file="eruptionsneighs4.RData")
nonlinearTseries::recurrencePlotAux(eruptionsneighs4)
```

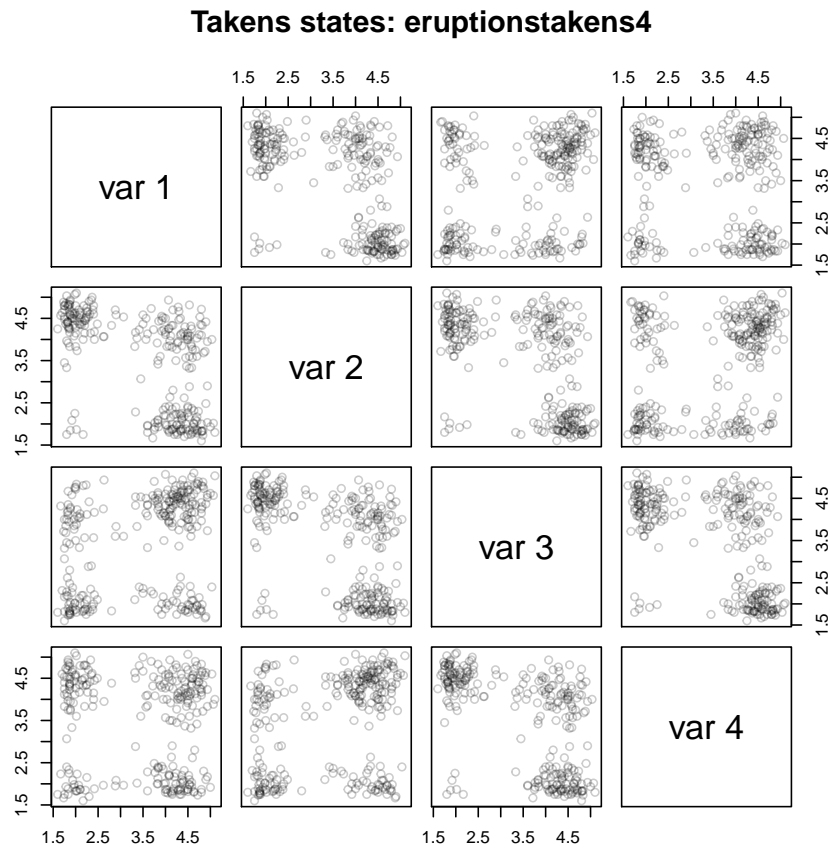


FIGURE 11. Example case: Old Faithful Geyser eruptions. Time used: 0.288 sec.

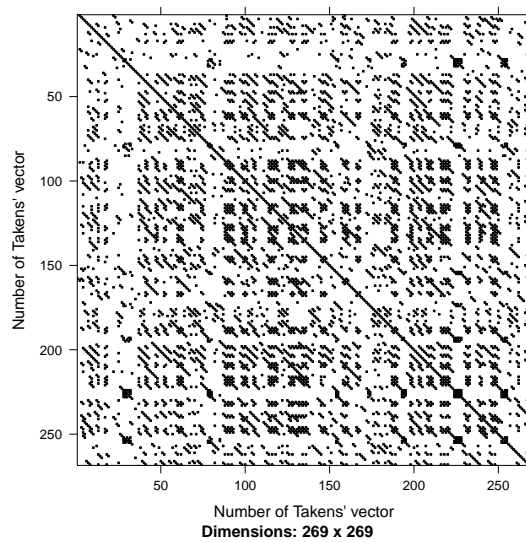


FIGURE 12. Recurrence Plot. Example case: Old Faithful Geyser eruptions. Dim=4. Time used: 23.292 sec.

5.1.1. *Dim=2.*

Input

```
eruptionstakens2 <- buildTakens(time.series=faithful$eruptions, embedding.dim=2, time.lag=1)
statepairs(eruptionstakens2)
```

See Figure 13

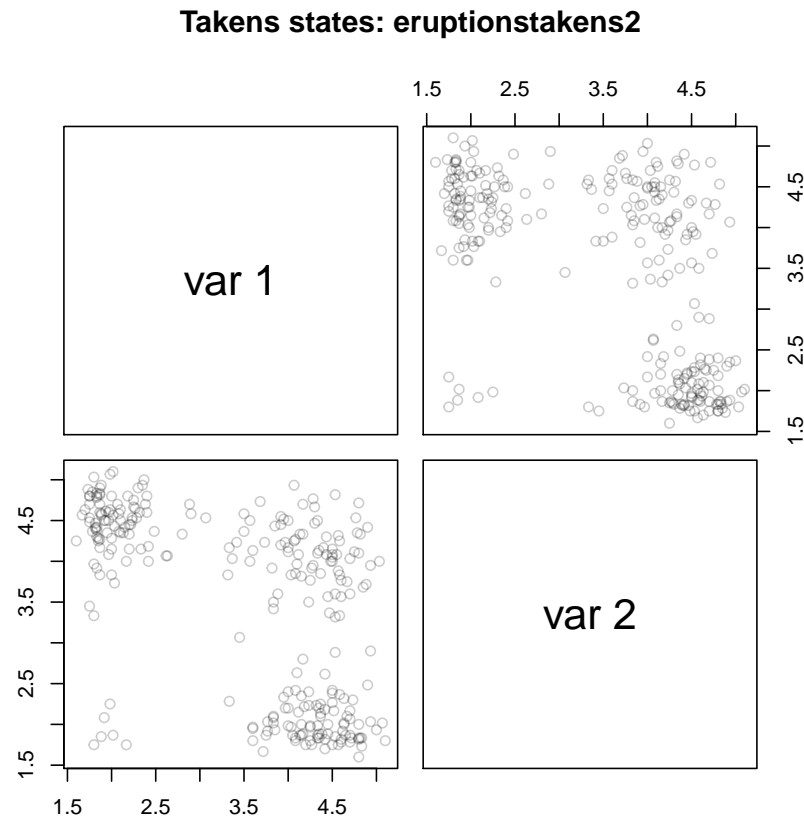


FIGURE 13. Example case: Old Faithful Geyser eruptions. $\text{Dim}=2$. Time used: 0.128 sec.

Input

```
eruptionsneighs2<-nonlinearTseries::findAllNeighbours(eruptionstakens2, radius=0.8)
save(eruptionsneighs2, file="eruptionsneighs2.RData")
```

Input

```
load(file="eruptionsneighs2.RData")
nonlinearTseries::recurrencePlotAux(eruptionsneighs2)
```

5.1.2. *Dim=6.*

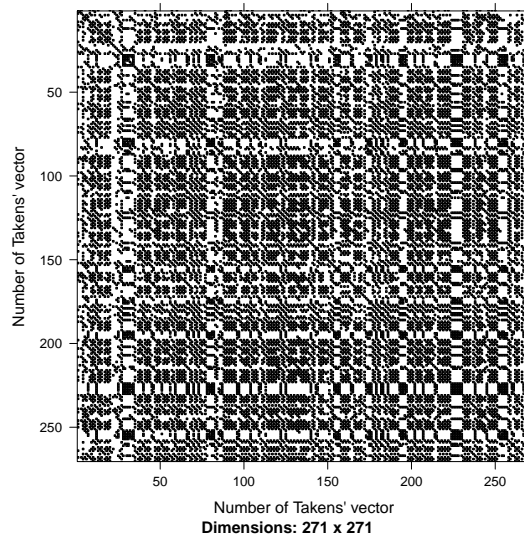


FIGURE 14. Recurrence Plot. Example case: Old Faithful Geyser eruptions. Dim=2. Time used: 74.516 sec.

Input

```
eruptionstakens6 <- buildTakens( time.series=faithful$eruptions,embedding.dim=6,time.lag=1)
statepairs(eruptionstakens6)
```

See Figure 15 on the next page

Input

```
eruptionsneighs6<-nonlinearTseries::findAllNeighbours(eruptionstakens6, radius=0.8)
save(eruptionsneighs6, file="eruptionsneighs6.RData")
```

Input

```
load(file="eruptionsneighs6.RData")
nonlinearTseries::recurrencePlotAux(eruptionsneighs6)
```

5.1.3. *Dim=8.*

Input

```
eruptionstakens8 <- buildTakens( time.series=faithful$eruptions,embedding.dim=8,time.lag=1)
statepairs(eruptionstakens8)
```

See Figure 17 on page 15

Input

```
eruptionsneighs8<-nonlinearTseries::findAllNeighbours(eruptionstakens8, radius=0.8)
save(eruptionsneighs8, file="eruptionsneighs8.RData")
```

Input

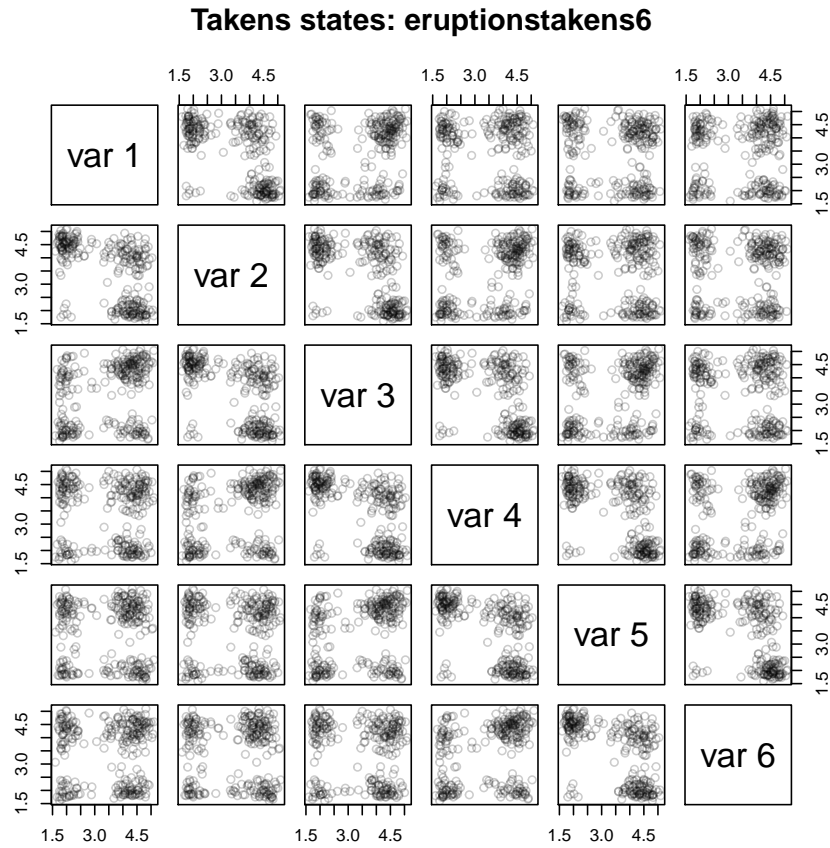


FIGURE 15. Example case: Old Faithful Geyser eruptions. Dim=6. Time used: 0.537 sec.

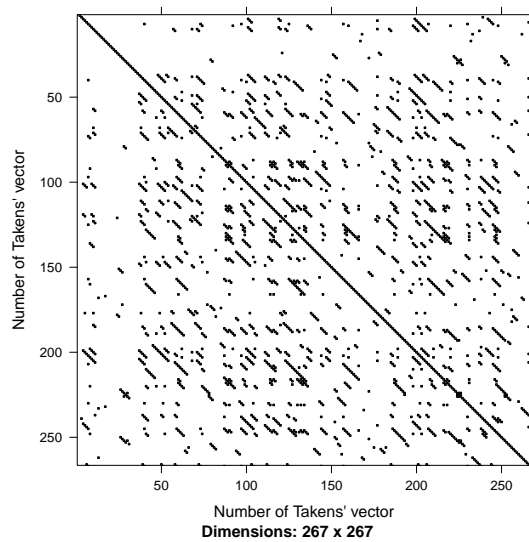


FIGURE 16. Recurrence Plot. Example case: Old Faithful Geyser eruptions. Dim=6. Time used: 7.676 sec.

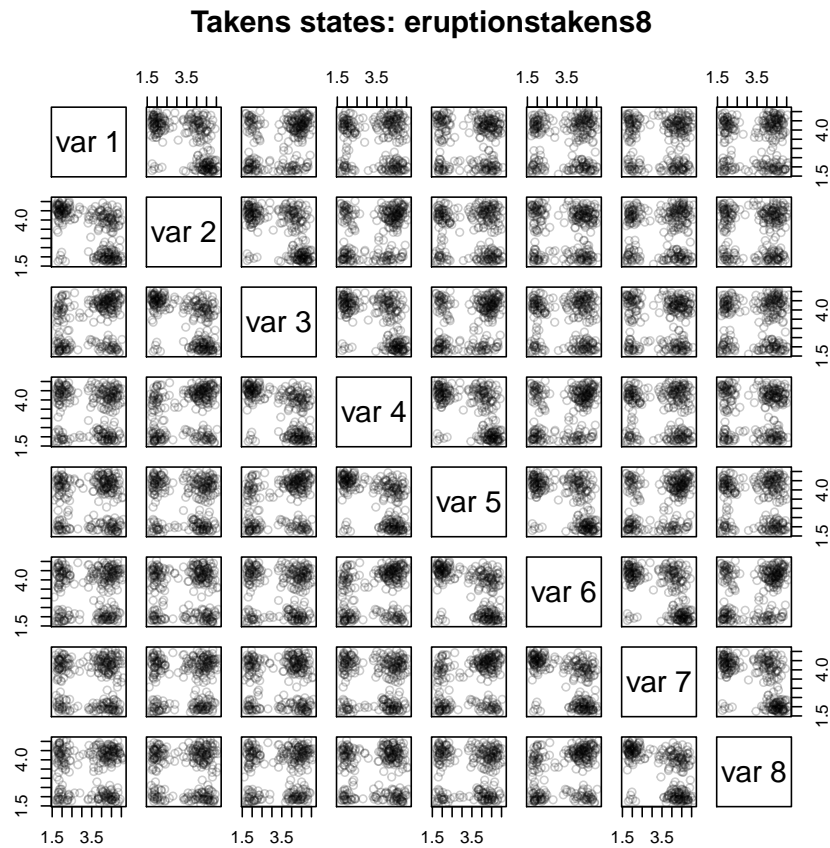


FIGURE 17. Example case: Old Faithful Geyser eruptions. Dim=8. Time used: 0.881 sec.

```
load(file="eruptionsneighs8.RData")
nonlinearTseries::recurrencePlotAux(eruptionsneighs8)
```

5.2. Geyser Eruptions: Comparison by Dimension. For comparison, recurrence plots for the Geyser data with varying dimension are in Figure 19 on page 17

5.3. Geyser Waiting.

```
plotsignal(faithful$waiting) Input
```

See Figure 20 on page 17,

```
waitingtakens <- buildTakens( time.series=faithful$waiting,embedding.dim=4,time.lag=4)
statepairs(waitingtakens) Input
```

See Figure 21 on page 18

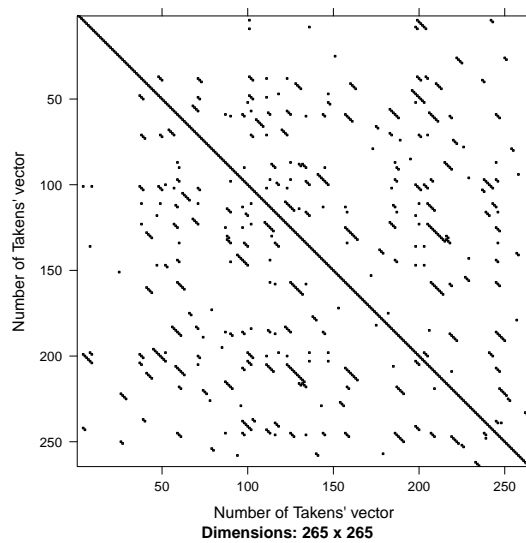


FIGURE 18. Recurrence Plot. Example case: Old Faithful Geyser eruptions. Dim=8. Time used: 2.768 sec.

Input

```
waitingneighs<-nonlinearTseries::findAllNeighbours(waitingtakens, radius=16)
save(waitingneighs, file="waitingneighs.Rdata")
```

Input

```
load(file="waitingneighs.RData")
nonlinearTseries::recurrencePlotAux(waitingneighs)
```

6. CASE STUDY: HRV DATA

Input

```
library(RHRV)
load("/data/pulse/rhrv/pkg/data/HRVData.rda")
load("/data/pulse/rhrv/pkg/data/HRVProcessedData.rda")
#####
### code chunk number 1: creation
#####
hrv.data = CreateHRVData()
hrv.data = SetVerbose(hrv.data, TRUE )
#####
### code chunk number 3: loading
#####
hrv.data = LoadBeatAscii(hrv.data, "example.beats",
  RecordPath = "/data/pulse/rhrv/tutorial/beatsFolder")
```

Output

```
** Loading beats positions for record: example.beats **
Path: /data/pulse/rhrv/tutorial/beatsFolder
Scale: 1
Date: 01/01/1900
```

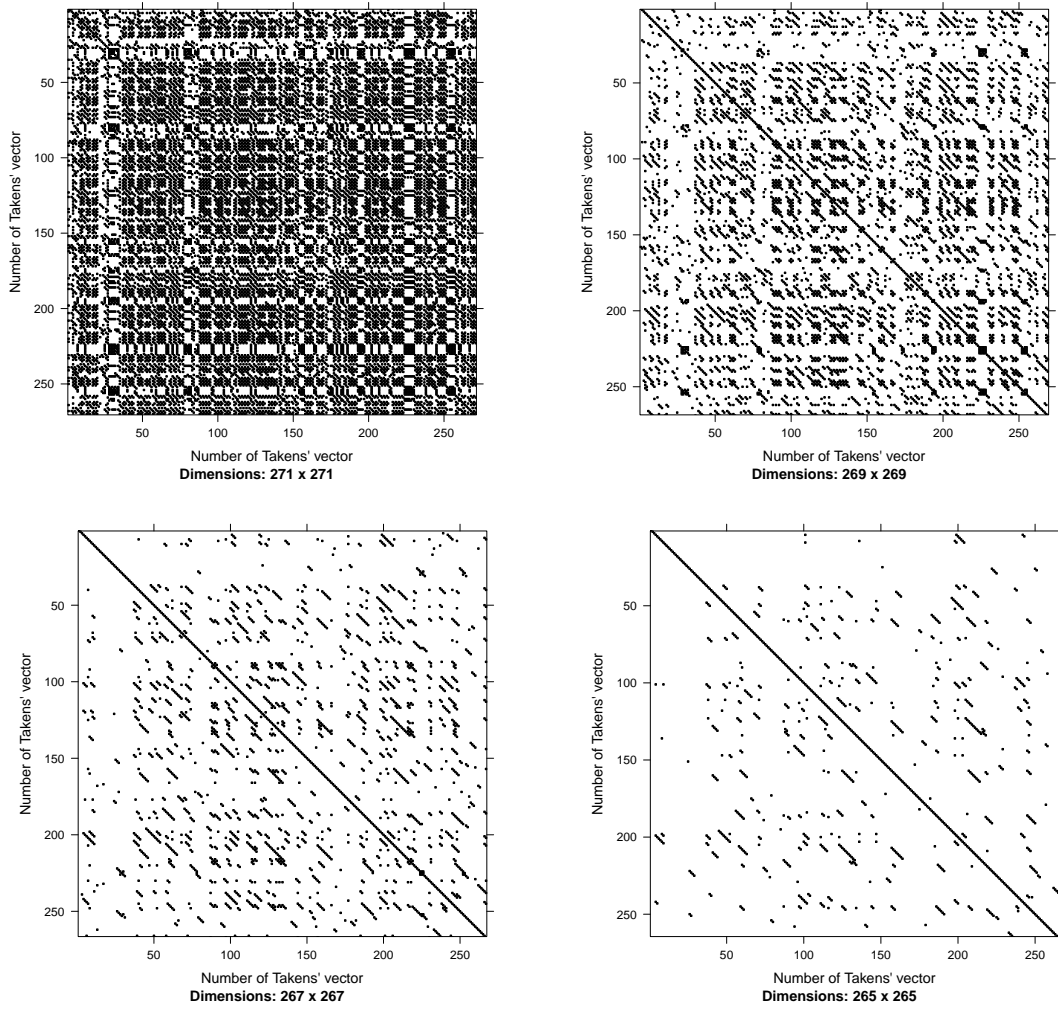



FIGURE 19. Recurrence Plot. Example case: Old Faithful Geyser eruptions. Dim=2, 4, 6, 8.

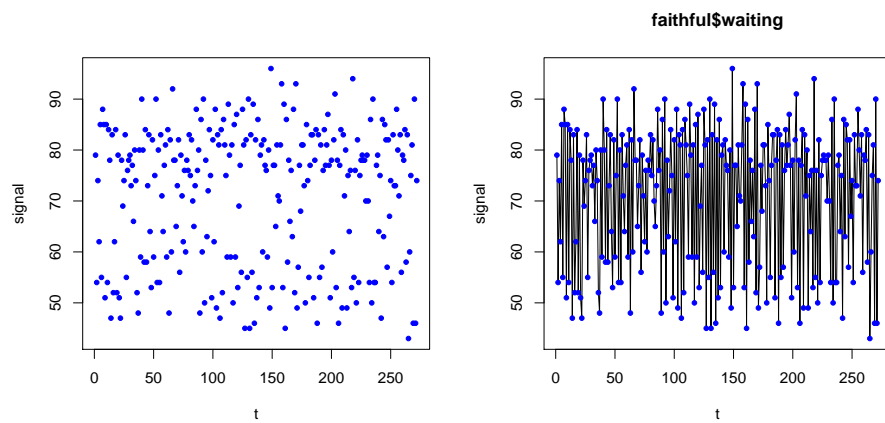


FIGURE 20. Example case: Old Faithful Geyser waiting. Signal and linear interpolation. Time used: 2.99 sec.

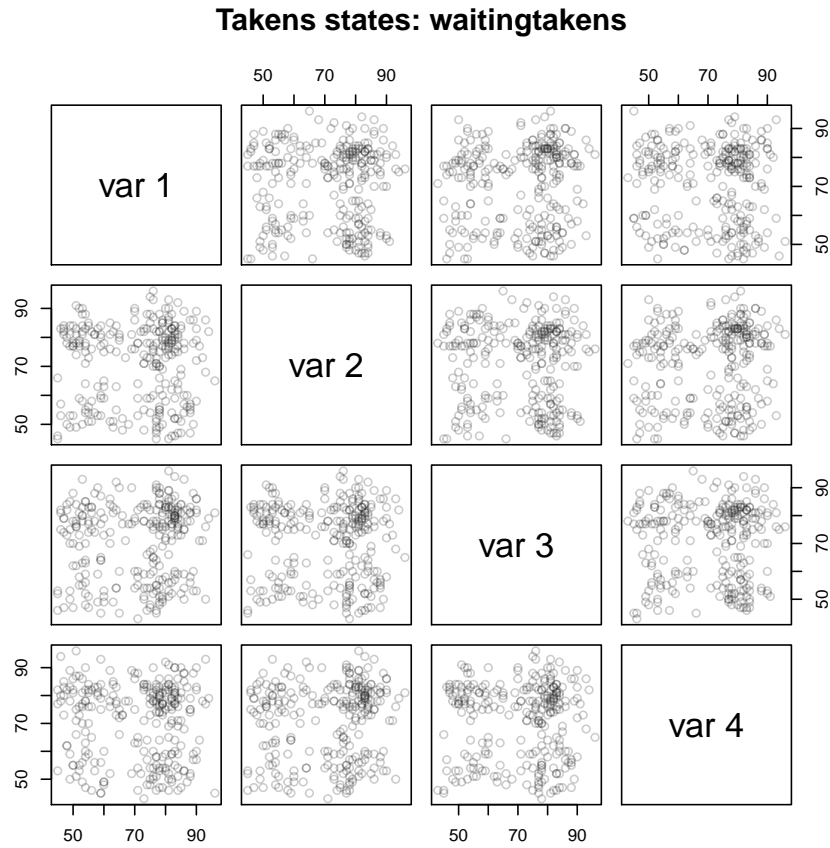


FIGURE 21. Example case: Old Faithful Geyser waiting. Time used: 0.268 sec.

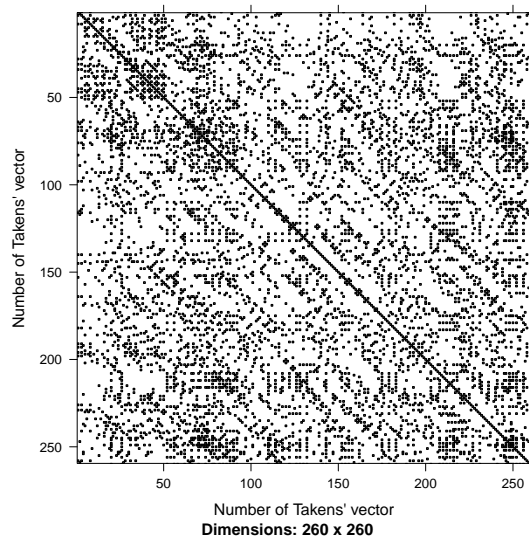


FIGURE 22. Recurrence Plot. Example case: Old Faithful Geyser waiting. Time used: 24.276 sec.

Time: 00:00:00
Number of beats: 17360

```
#      RecordPath = "beatsFolder")      Input

#####
### code chunk number 4: derivating
#####
hrv.data = BuildNIHR(hrv.data)
```

```
** Calculating non-interpolated heart rate **      Output
Number of beats: 17360
```

```
Input

plotsignal(hrv.data$Beat$RR)
```

See Figure 23,

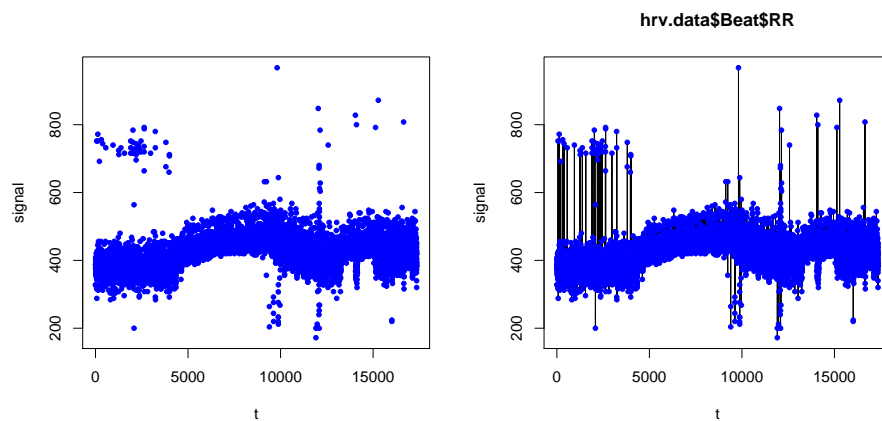


FIGURE 23. Example case: RHRV tutorial. Signal and linear interpolation.

```
Input
hrvRRtakens4 <- buildTakens( time.series=hrv.data$Beat$RR[1:nsignal],embedding.dim=4,time.lag=1)
statepairs(hrvRRtakens4)
```

See Figure 24 on the next page

```
Input
statepairs(hrvRRtakens4, rank=TRUE)
```

ToDo: We have a problem with the signal. It lies at approximately $2 \times RR$. Could be an artefact of the processing, too many in

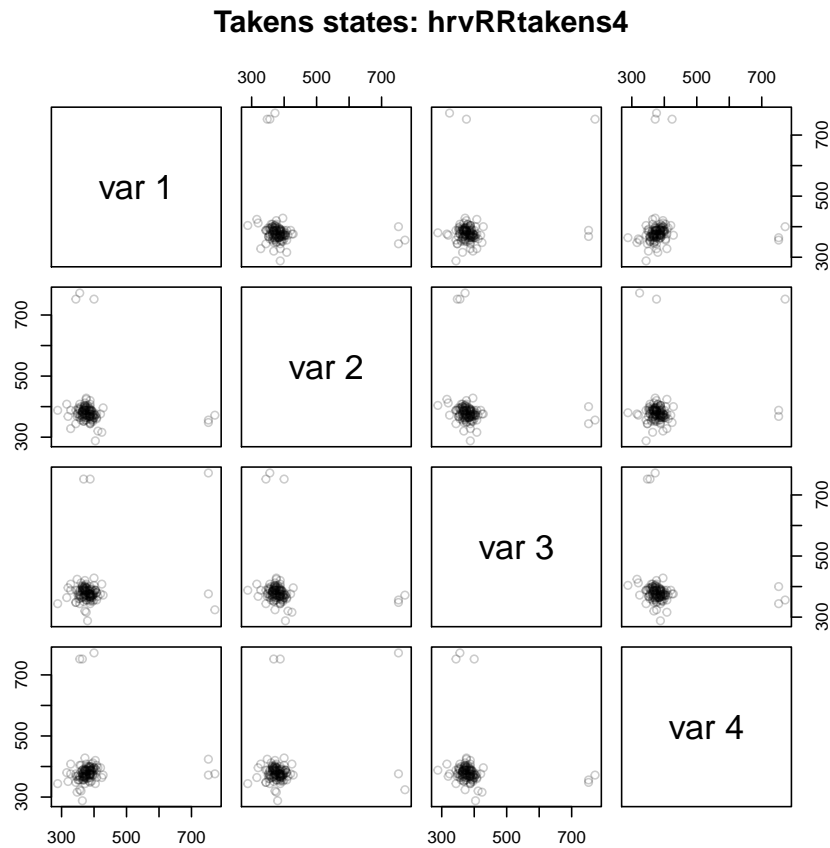


FIGURE 24. Example case: RHRV tutorial. Time used: 0.197 sec.

See Figure 25 on the facing page

Input

```
hrvRRneighs4 <- nonlinearTseries::findAllNeighbours(hrvRRtakens, radius=16)
save(hrvRRneighs4, file="hrvRRneighs4.Rdata")
```

Time used: 0.034 sec.

Input

```
load(file="hrvRRneighs4.Rdata")
nonlinearTseries::recurrencePlotAux(hrvRRneighs4)
```

6.1. RHRV: Comparison by Dimension.

Input

```
hrvRRtakens2 <- buildTakens( time.series=hrv.data$Beat$RR[1:nsignal], embedding.dim=2, time.lag=1)
hrvRRneighs2 <- nonlinearTseries::findAllNeighbours(hrvRRtakens2, radius=16)
save(hrvRRneighs2, file="hrvRRneighs2.Rdata")
```

Time used: 0.053 sec.

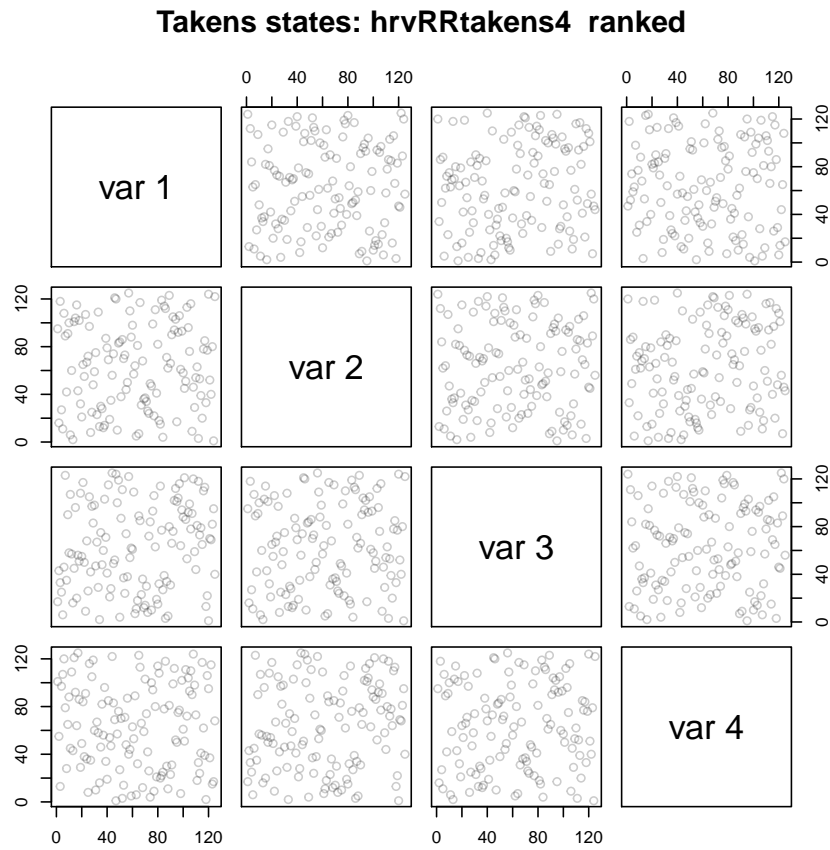


FIGURE 25. Example case: RHRV tutorial. Ranked data. Time used: 0.436 sec.

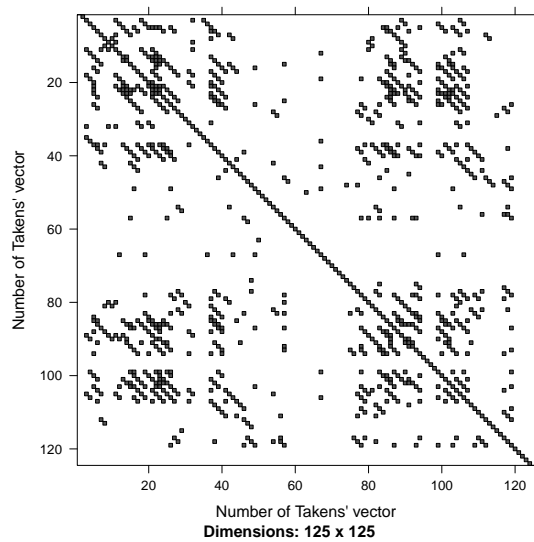


FIGURE 26. Recurrence Plot. Example case: RHRV tutorial. Dim=4. Time used: 3.647 sec.

Input

```
load(file="hrvRRneighs2.RData")
nonlinearTseries::recurrencePlotAux(hrvRRneighs2)
```

Time used: 11.47 sec.

Input

```
hrvRRtakens6 <- buildTakens( time.series=hrv.data$Beat$RR[1:nsignal],embedding.dim=6,time.lag=1)
hrvRRneighs6 <-nonlinearTseries::findAllNeighbours(hrvRRtakens6, radius=16)
save(hrvRRneighs6, file="hrvRRneighs6.Rdata")
```

Time used: 0.047 sec.

Input

```
load(file="hrvRRneighs6.RData")
nonlinearTseries::recurrencePlotAux(hrvRRneighs6)
```

Dim=6. Time used: 1.212 sec.

Input

```
hrvRRtakens8 <- buildTakens( time.series=hrv.data$Beat$RR[1:nsignal],embedding.dim=8,time.lag=1)
hrvRRneighs8 <-nonlinearTseries::findAllNeighbours(hrvRRtakens8, radius=16)
save(hrvRRneighs8, file="hrvRRneighs8.Rdata")
```

Time used: 0.036 sec.

Input

```
load(file="hrvRRneighs8.RData")
nonlinearTseries::recurrencePlotAux(hrvRRneighs8)
```

Dim=8. Time used: 0.539 sec.

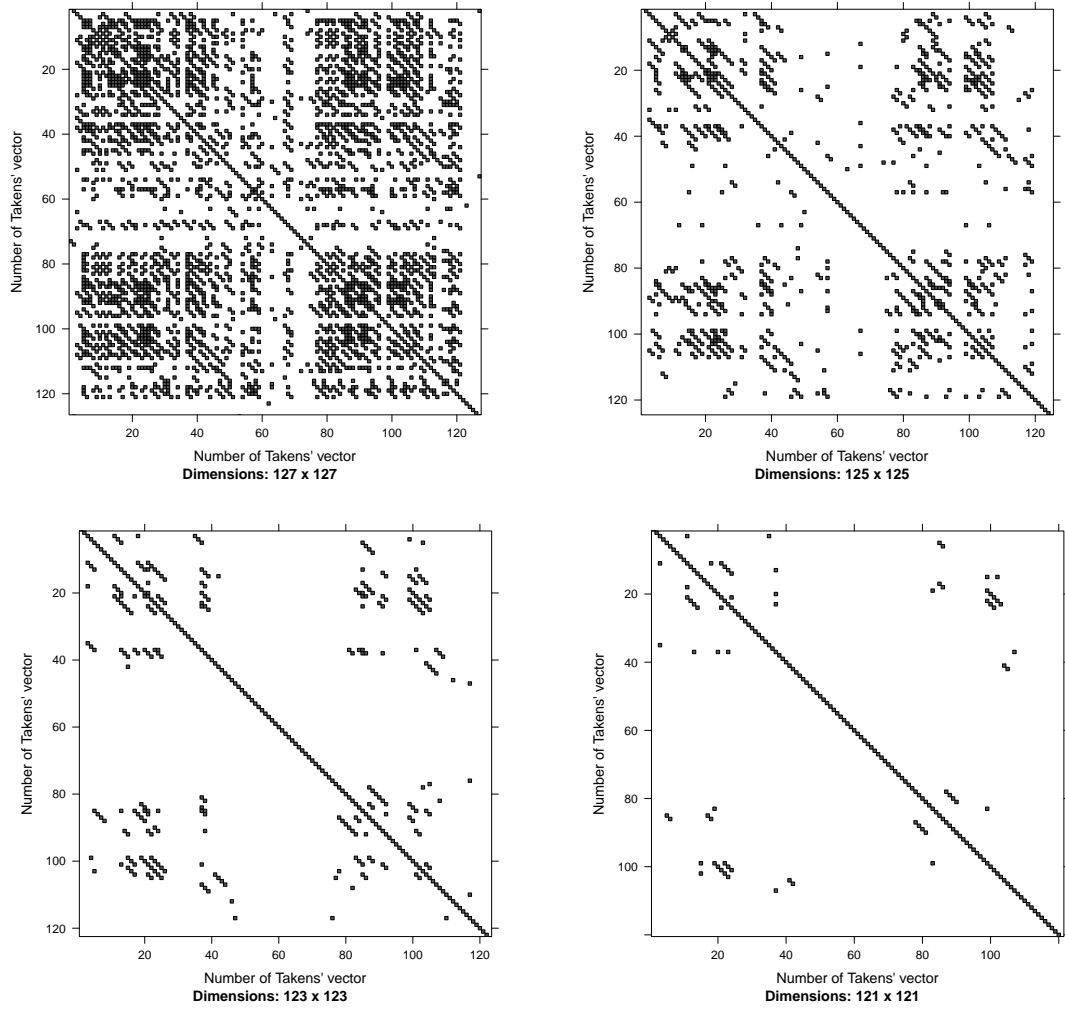


FIGURE 27. Recurrence Plot. Example case: RHRV tutorial. Dim=2, 4, 6, 8. Time used: 0.54 sec.

REFERENCES

- ECKMANN, JEAN-PIERRE, KAMPHORST, S OLIFFSON, & RUELLE, DAVID. 1987. Recurrence plots of dynamical systems. *Europhys. Lett*, **4**(9), 973–977.

INDEX

ToDo

- 2: include doppler waveslim, 4
- 5: extend to two-dimensional data, 10
- 6: We have outliers at approximately $2*RR$. Could this be an artefact of preprocessing, filtering too many impulses?, 19

Geyser, 10

heart rate variation, 16

R session info:

Total Sweave time used: 166.87 sec.

- R version 3.0.2 (2013-09-25), x86_64-apple-darwin10.8.0
- Locale: en_GB.UTF-8/en_GB.UTF-8/en_GB.UTF-8/C/en_GB.UTF-8/en_GB.UTF-8
- Base packages: base, datasets, graphics, grDevices, methods, stats, tcltk, utils
- Other packages: leaps 2.9, locfit 1.5-9.1, MASS 7.3-29, Matrix 1.1-2, mgcv 1.7-27, nlme 3.1-113, nonlinearTseries 0.2, rgl 0.93.996, RHRV 4.0, sintro 0.1-3, tkrplot 0.0-23, TSA 1.01, tseries 0.10-32, waveslim 1.7.3
- Loaded via a namespace (and not attached): grid 3.0.2, lattice 0.20-25, quadprog 1.5-5, tools 3.0.2, zoo 1.7-11

L^AT_EX information:

textwidth: 6.00612in linewidth: 6.00612in
textheight: 9.21922in

CVS/Svn repository information:

```
$Source: /u/math/j40/cvsroot/lectures/src/dataanalysis/Rnw/recurrence.Rnw,v $  
$Revision: 1.2 $  
$Date: 2014/02/05 20:05:07 $  
$Name: $  
$Author: j40 $
```

E-mail address: gs@statlab.uni-heidelberg.de

GÜNTHER SAWITZKI
STATLAB HEIDELBERG
IM NEUENHEIMER FELD 294
D 69120 HEIDELBERG