

Package ‘wccsom’

February 15, 2012

Version 1.2.8

Title SOM networks for comparing patterns with peak shifts

Author Ron Wehrens

Maintainer Ron Wehrens <ron.wehrens@fmach.it>

Description SOM networks for comparing patterns with peak shifts.

License GPL (>= 2)

Depends class, MASS

Repository CRAN

Date/Publication 2012-01-09 11:54:24

R topics documented:

bucket	2
cepha	3
classvec2classmat	3
degelder	4
expand.som	5
plot.wccsom	6
predict.wccsom	8
summary.wccsom	9
unit.distances	11
wcc	12
wccassign	13
wccmap	14
wccsom	15
wccxyf	17

Index	19
--------------	-----------

bucket

Variable averaging (bucketing) for data matrices

Description

Function `bucket` decreases the size (i.e., the number of columns) of a data matrix by averaging variables. Function `debucket` achieves the reverse by linear interpolation.

Usage

```
bucket(x, factor)
debucket(x, nout)
```

Arguments

<code>x</code>	Data matrix: each variable corresponds with a column.
<code>factor</code>	Bucket factor: this number of variables will be averaged.
<code>nout</code>	Required number of variables after debucketing.

Value

Returns a data matrix of the new dimensions.

Author(s)

Ron Wehrens

Examples

```
data(cepha)
gr <- somgrid(3,3, "hexagonal")
set.seed(7)
system.time(x <- wccsom(cepha$patterns, grid=gr, trwidth=20,
                       rlen=500, FineTune=FALSE))

X <- bucket(cepha$patterns, 4)
system.time(x <- wccsom(X, grid=gr, trwidth=5,
                       rlen=500, FineTune=FALSE))
```

cepha

Cephalosporin data set

Description

X-ray powder patterns of 20 cephalosporin / antibiotic complexes.

Usage

```
data(cepha)
```

Format

This yields a list with three components: the first component, "patterns", is a matrix of 20 rows and 425 variables, containing the powder patterns; the second component is "class.names", and gives information on the class of the crystal structure. The final component, "thetas", contains the 2theta values at which intensities have been measured.

References

R. de Gelder, R. Wehrens, and J.A. Hageman. J. Comput. Chem., 22(3), 273-289, 2001.

Examples

```
data(cepha)
plot(cepha$thetas, cepha$patterns[1,], type="l", xlab="2theta", ylab="Counts")
matplot(cepha$thetas, t(cepha$patterns), type="l", xlab="2theta", ylab="Counts",
        col=as.integer(factor(cepha$class.names))+1, lty=1)
```

classvec2classmat

Convert a classification vector into a matrix or the other way around.

Description

Functions toggle between a matrix representation, where class membership is indicated with one '1' and for the rest zeros at each row, and an class vector (maybe integers or class names). The classification matrix contains one column per class. Conversion from a class matrix to a class vector assigns each row to the column with the highest value. An optional argument can be used to assign only those objects that have a probability higher than a certain threshold (default is 0).

Usage

```
classvec2classmat(yvec)
classmat2classvec(yamat, threshold=0)
```

Arguments

yvec class vector. Usually integer values, but other types are also allowed.
ymat class matrix: every column corresponds to a class.
threshold only classify into a class if the probability is larger than this threshold.

Value

classvec2classmat returns the classification matrix, where each column consists of zeros and ones; classmat2classvec returns a class vector (integers).

Author(s)

Ron Wehrens

See Also

[wccxyf](#)

Examples

```
classes <- c(rep(1, 5), rep(2, 7), rep(3, 9))
classmat <- classvec2classmat(classes)
classmat
classmat2classvec(classmat)
```

degelder

Powder pattern data by Rene de Gelder

Description

X-ray powder patterns of 131 crystallographic structures, contributed by Rene de Gelder.

Usage

```
data(degelder)
```

Format

This yields a list with three components: the first component, "patterns", is a matrix of 131 rows and 441 variables, containing the powder patterns; the second component is "thetas", the 2theta values at which intensities have been measured. The final component, "properties", gives information on the crystallographic properties of the structures.

Source

Rene de Gelder, Institute of Molecules and Materials, Radboud University Nijmegen.

Examples

```
data(degelder)
set.seed(1)
geldemap <- wccsom(degelder$patterns, somgrid(6, 6, "hexagonal"))
options(digits = 3)
summary(geldemap, "unit", nr=1, properties = degelder$properties)
```

expand.som

Expand a SOM network

Description

Increases the size of a network by a factor of 2 in both x and y directions; code vectors for new units are interpolated.

Usage

```
expand.som(somnet, plotit=FALSE)
```

Arguments

somnet	Object of class "wccsom", generated by wccsom.
plotit	If TRUE, plots the new network with the original units coloured white, the new ones red.

Details

For hexagonal grids, the six closest units are used for interpolation, weighted by distance; for rectangular grids, the four closest are used.

Value

Returns the new network.

Examples

```
data(cepha)
gr <- somgrid(3,3, "hexagonal")
set.seed(7)
x <- wccsom(cepha$patterns, grid=gr, trwidth=20, rlen=50)
x2 <- expand.som(x, plotit=TRUE)
```

plot.wccsom

Plot wccsom object

Description

Plot self-organising map, obtained from function wccsom. Several types of plots are supported.

Usage

```
## S3 method for class 'wccsom'
plot(x, type = c("changes", "codes", "counts", "mapping",
               "prediction", "property", "quality"),
     classif = NULL, labels = NULL, pchs = NULL, main = NULL,
     palette.name = heat.colors, ncolors, unit.colors,
     unit.bgcol = NULL, zlim = NULL, property = NULL,
     heatkey=TRUE, contin, ...)
```

Arguments

x	SOM object.
type	Type of plot. (Wow!)
classif	classification object, as returned by wccassign. Only needed if type equals "mapping", "quality", and "counts". Also a vector of class numbers may be given. If the network was trained with keep.data equal to TRUE, then the object already contains this information.
labels	(optional) labels to plot when type equals "mapping".
pchs	(optional) plotting symbols to use when type equals "mapping".
main	title of the plot.
palette.name	colors to use as unit background for "codes", "counts", "prediction", "property" and "quality" plotting types.
ncolors	number of colors to use in the palette.
unit.colors	explicit definition of the colors for the units in the "codes", "counts", "prediction", "property" and "quality" plotting types.
unit.bgcol	background color for units if no other color information is to be plotted. For plotting type is "codes", the default is "transparent"; in other cases the default is "gray".
zlim	Optional range for color coding of unit backgrounds.
property	Values to use if the "property" plotting type. Can be used for colouring units in general.
heatkey	Whether or not to generate a heatkey at the left side of the plot in the "quality", "counts", and "property" plotting types.
contin	Whether the heatkey should show a range of values (TRUE) or a series of discrete values. The function tries to make a good guess; in case of strange-looking results it may pay to explicitly provide a value for this argument.

... Other graphical parameters, e.g. colours of labels in the "mapping" plotting type.

Details

Several different types of plots are supported:

"changes" Shows the mean change in similarity to the best matching codebook vector for each epoch. Since codebook vectors become more similar to the data that are mapped to them, the changes should always be positive. Upon convergence, the changes should be very small.

"codes" Shows the codebook vectors.

"counts" Shows the number of objects mapped to the individual units. Empty units are depicted in gray.

"mapping" Show where a set of objects is mapped. It needs a "labels" argument: a string name for each object.

"prediction" Shows predictions for units; if no "property" argument is given, the function will for supervised maps the predictions for every unit; for unsupervised maps, where this information is not available, it gives an error.

"property" Plot a map with the units coloured according to a specific property. The standard application is to precompute the similarities of an object to all units in the map, and plot these with a colour key. Also other quantities may be used to colour the units: see the example of distances below.

"quality" Shows the units coloured according to the mean agreement (WCC) of mapped objects to the unit vector. A colour key is plotted on the left. The variation in the WCCs of the mapped objects is indicated by the blue line: if it is pointing downwards it indicates low variation, if pointing upwards large variation.

Value

If type equals "property", the wcc values for all units are returned.

Author(s)

Ron Wehrens

References

R. Wehrens, W.J. Melssen, L.M.C. Buydens and R. de Gelder. Representing Structural Databases in a Self-Organising Map. Acta Cryst. B61, 548-557, 2005.

See Also

[wccsom](#), [wccxyf](#), [unit.distances](#), [predict.wccsom](#)

Examples

```

data(cepha)
gr <- somgrid(3, 3, "hexagonal")
set.seed(7)
x <- wccsom(cepha$patterns, grid = gr, trwidth = 20, rlen = 100)

par(mfrow = c(3,2))
plot(x, type = "codes", main = "Codebook vectors")
plot(x, type = "changes", main = "Convergence")
plot(x, type = "counts", main = "Mapping counts")
plot(x, type = "quality", main = "Mapping quality")
plot(x, type = "mapping", main = "Mapping",
      labels = cepha$class.names, col = as.integer(factor(cepha$class.names)))
plot(x, type = "mapping", main = "Mapping",
      pchs = as.integer(factor(cepha$class.names)),
      col = as.integer(factor(cepha$class.names)))

par(mfrow=c(1,1))
obj1.wccs <- wccmap(x, cepha$patterns[1,])
plot(x, type = "property", property = obj1.wccs)

```

predict.wccsom

Predict properties from self-organising maps

Description

Function to predict property values for every unit in a supervised or unsupervised SOM. These, in turn, are used to provide predictions for individual objects.

Usage

```

## S3 method for class 'wccsom'
predict(object, newdata,
        trainX, trainY, unit.predictions, ...)

```

Arguments

object	Trained SOM.
newdata	If new objects are supplied (in the form of a matrix), they are mapped to the SOM; predictions for the new data are the predicted values associated with the units to which they are mapped. In order to calculate these the maps should be either supervised, or the training data should be stored in the map, or these data should be provided through arguments <code>"trainX"</code> and <code>"trainY"</code> , or finally, the unit predictions can be explicitly given (<code>"unit.predictions"</code>).
trainX	Training data, only used when they have not been stored in the trained map.
trainY	Dependent values for the training data.
unit.predictions	Alternatively, one can provide predictions for every unit.
...	Not used.

Details

For supervised SOMs, predictions per unit are available after training. For unsupervised SOMs, these predictions can be obtained from the average values of the properties of training set objects mapping to specific units. New objects that are mapped to the SOM will receive the predicted value of the unit to which they are mapped.

Value

The function returns a list with components

```
unit.predictions      Property predictions per unit of the map.
predictions          Property predictions for the new data.
```

Author(s)

R. Wehrens

References

~put references to the literature/web site here ~

See Also

[wccsom](#), [wccxyf](#), [plot.wccsom](#)

Examples

```
data(degelder)
gr <- somgrid(5, 5, "hexagonal")
set.seed(7)
x <- wccxyf(degelder$patterns, degelder$properties[, "cell.vol"],
            grid=gr, trwidth=20, rlen=100)
plot(x, "predict")

predicted.volumes <- predict(x)
plot(degelder$properties[, "cell.vol"], predicted.volumes$predictions,
     xlab="Cell volume", ylab="Predicted cell volume")
abline(0,1, col="gray")
```

summary.wccsom

Summarizing wccsom objects

Description

Summary for objects of class "wccsom"

Usage

```
## S3 method for class 'wccsom'  
summary(object, type = c("unit", "object", "smoothness", "quality"),  
        nr, labels, data = object$data,  
        classif = object$unit.classif,  
        wccs = object$wccs, properties = NULL, ...)
```

Arguments

object	Trained Kohonen map.
type	One of "unit", "object", "smoothness" or "quality".
nr	Number of the unit or object that is to be summarised (only needed for type "unit" and "object").
labels	Labels for all objects (not needed for type equaling "smoothness" and "quality"). Defaults to integer numbers.
data	Data matrix (not needed for type equaling "smoothness"). Default is to take the training objects (if available).
classif	Mapping of all objects (not needed for type equaling "smoothness"). Default is to take the mapping of the training objects (if available). Alternatively, a mapping object (output of wccassign may be given for this argument).
wccs	WCC values of all mapped objects (not needed for type equaling "smoothness"). Default is to take the training objects (if available).
properties	Other properties of the crystals that should be shown in the summary, as e.g. reduced cell parameters.
...	Further arguments. Currently ignored.

Details

Several types of summary are calculated and printed to the screen. If type equals "unit", a summary is given of all objects mapped to that unit. In case type equals "object", the summary indicates the unit to which the object is mapped and shows other objects mapped to it. For a type of "smoothness", the function returns the ratio of WCC values between neighbouring and non-neighbouring units. Values larger than 1 show that neighbouring units are "more alike" than non-neighbouring units. Type "quality" does something similar: it compares all data objects and takes the ratio of WCCs between objects mapped to the same unit and objects mapped to different units.

Author(s)

Ron Wehrens

See Also

[WCCSOM](#)

Examples

```
data(cepha)
gr <- somgrid(3,3, "hexagonal")
set.seed(7)
x <- wccsom(cepha$patterns, grid=gr, trwidth=20, rlen=100)

summary(x, "unit", nr=1, labels=paste(cepha$class.names, 1:20, sep=""))
summary(x, "object", nr=1, labels=paste(cepha$class.names, 1:20, sep=""))
summary(x, type = "smoothness")
summary(x, "quality")
```

unit.distances	<i>Function to calculate distances between units in a SOM</i>
----------------	---

Description

Function calculates Euclidean distances between units in a SOM; if argument "toroidal" is TRUE, the edges of the map are considered to be joined so that the overall shape of the map is a torus. The distances are calculated correspondingly.

Usage

```
unit.distances(grid, toroidal)
```

Arguments

grid	A somgrid object.
toroidal	For toroidal maps, equal to TRUE. Default is FALSE.

Value

Returns a distance matrix.

Author(s)

Ron Wehrens

See Also

[wccsom](#), [wccxyf](#)

Examples

```
gr <- somgrid(3, 3, "hexagonal")
x <- list(grid = gr)
class(x) <- "wccsom"

par(mfrow = c(1,2))
unit.dists <- unit.distances(gr, toroidal = FALSE)
plot(x, type = "property", property = unit.dists[1,],
     main = "Distances to unit 1", zlim = c(0,2.75), contin = TRUE)
unit.dists <- unit.distances(gr, toroidal = TRUE)
plot(x, type = "property", property = unit.dists[1,],
     main = "Toroidal distances to unit 1", zlim = c(0,2.75), contin = TRUE)
```

wcc

Agreement between patterns including peak shifts

Description

Weighted cross correlation and autocorrelation, as described in De Gelder et al. (2001), for assessing similarities in spectra-like data containing peak shifts. Euclidean distances are useless in this situation.

Usage

```
wcc(pattern1, pattern2, trwidth, wghts, acors)
wac(pattern1, trwidth, wghts)
wacmat(patterns, trwidth, wghts, do.transpose = TRUE)
```

Arguments

pattern1	Pattern.
pattern2	Another pattern.
patterns	Pattern matrix: rows correspond with patterns.
trwidth	Triangle width, given in the number of data points.
wghts	Optional weights vector, will be calculated from triangle width if necessary. Sometimes it is more efficient to pre-calculate it and give it as an argument.
acors	Autocorrelation, also optional to speed up calculations.
do.transpose	Internally, columns should correspond with patterns, so normally one should leave this value to its default: TRUE. If a matrix is already in the correct format, one can avoid unnecessary double transpose operations.

Value

Function wcc returns the WCC value, a similarity value between 0 and 1. Functions wac and wacmat return weighted autocorrelations for one pattern and a matrix of patterns, respectively.

Author(s)

Ron Wehrens

References

R. de Gelder, R. Wehrens, and J.A. Hageman. A generalized expression for the similarity spectra: application to powder diffraction pattern classification. *J. Comput. Chem.*, 22(3), 273-289, 2001.

See Also

[wccsom](#), [wccxyf](#)

Examples

```
data(cepha)
wac(cepha$patterns[1,], 20)
wacmat(t(cepha$patterns), 20)
wcc(cepha$patterns[1,], cepha$patterns[2,], 20)
```

wccassign

Assign patterns to nodes in a SOM network by WCC value

Description

KNN assignment of patterns to units in a Kohonen map, with maximal WCC as the criterion.

Usage

```
wccassign(x, data)
```

Arguments

x	Trained Kohonen map
data	Data matrix

Value

Returns a list with components:

classif	Unit numbers to which rows in the data matrix are assigned
wccs	wcc value of rows in the data matrix and the units to which they are assigned.

Author(s)

Ron Wehrens

See Also

[wccsom](#)

Examples

```
data(cepha)
gr <- somgrid(3,3, "hexagonal")
set.seed(7)
x <- wccsom(cepha$patterns, grid = gr, trwidth = 20, rlen = 50,
           FineTune = FALSE, keep.data = FALSE)
sombins <- wccassign(x, cepha$patterns)
```

wccmap

Map one powder pattern to a trained Kohonen map

Description

Function calculates the agreement - as measured by WCC - of a powder pattern to all units in a network (SOM or XYF).

Usage

```
wccmap(x, obj)
```

Arguments

x	Kohonen network, either from wccsom or wccxyf
obj	Vector: new powder pattern

Value

Returns a vector of length equal to the number of units in the network, containing all WCC values, i.e. similarities of the new pattern to every unit.

Author(s)

Ron Wehrens

References

R. Wehrens, W.J. Melssen, L.M.C. Buydens and R. de Gelder. Representing Structural Databases in a Self-Organising Map. Acta Cryst. B61, 548-557, 2005.

See Also

[wccsom](#), [wccxyf](#), [wccassign](#)

Examples

```
data(cepha)
gr <- somgrid(3,3, "hexagonal")
set.seed(7)
x <- wccsom(cepha$patterns, grid=gr, trwidth=20, rlen=100)

wccs1 <- wccmap(x, cepha$patterns[1,])
par(mfrow=c(1,2))
plot(x, "property", property = wccs1, main="Unit similarities to object 1")
plot(x, "property", property = wccs1,
      main="Unit similarities to object 1", zlim=c(0.96, 1))
```

wccsom

Mapping spectra with self-organising maps

Description

Self-organising maps for mapping high-dimensional spectra or patterns to 2D; instead of Euclidean distance, the weighted cross correlation (WCC) similarity measure is used. Modelled after the SOM function in package 'class'. wccsom takes 'continuous' patterns, i.e. datapoints are equidistant.

Usage

```
wccsom(data, grid=somgrid(), rlen = 100, alpha = c(0.05, 0.01),
        radius = quantile(nhbrdist, 0.7), init, nhbrdist, trwidth = 20,
        toroidal = FALSE, FineTune = TRUE, keep.data = TRUE)
```

Arguments

data	Spectra or patterns to be mapped: a matrix, with each row representing a compound.
grid	A grid for the representatives: see 'somgrid'.
rlen	the number of times the complete data set will be presented to the network.
alpha	a vector of two numbers indicating the amount of change. Default is to decline linearly from 0.05 to 0.01 over rlen updates.
radius	the initial radius of the neighbourhood to be used for each update: the decrease is exponential over rlen updates in such a way that after one-third of the updates only the winning unit is updated. The default is to start with a value that covers 2/3 of all units.
init	the initial representatives, represented as a matrix. If missing, chosen (without replacement) randomly from 'data'.
nhbrdist	optionally, the distance matrix for the units.
trwidth	width of the triangle function used in the WCC measure, given in the number of data points.

toroidal	if TRUE, then the edges of the map are joined. Note that in a toroidal hexagonal map, the number of rows must be even.
FineTune	apply kmeans for fine-tuning the codebook vectors.
keep.data	store training data and their mapping in the network.

Value

an object of class "wccsom" with components

grid	the grid, an object of class "somgrid".
changes	vector of mean average deviations from code vectors
codes	a matrix of code vectors.
trwidth	the triangle width used for the WCC measure
acors	autocorrelations of the code vectors.
toroidal	setting of parameter 'toroidal'.
FineTune	setting of parameter 'FineTune'.
unit.classif	mapping of training data: a vector of unit numbers. Only if keep.data equals TRUE.
wccs	WCC values of all training data, compared to the best matching codebook vector. Only if keep.data equals TRUE.
data.acors	WAC values for training data. Only if keep.data equals TRUE.

Author(s)

Ron Wehrens

References

R. Wehrens, W.J. Melssen, L.M.C. Buydens and R. de Gelder. Representing Structural Databases in a Self-Organising Map. Acta Cryst. B61, 548-557, 2005.

See Also

[SOM](#), [plot.wccsom](#), [wccxyf](#), [wcc](#)

Examples

```
data(cepha)
gr <- somgrid(3,3, "hexagonal")
set.seed(7)
x <- wccsom(cepha$patterns, grid=gr, trwidth=20, rlen=100)
```

Description

Supervised self-organising maps for mapping high-dimensional spectra or patterns to 2D; instead of Euclidean distance, the weighted cross correlation (WCC) similarity measure is used. Modelled after the SOM function in package 'class'. wccxyf takes 'continuous' patterns, i.e. datapoints are equidistant.

At this point, no facilities are implemented for growing networks or k-means-like fine-tuning of the maps, such as in function wccsom.

Usage

```
wccxyf(data, Y, grid=somgrid(), rlen = 100, alpha = c(0.05, 0.01),
        radius = quantile(nhbrdist, 0.67), xweight = 0.5, trwidth = 20,
        toroidal = FALSE, keep.data = TRUE)
```

Arguments

data	Spectra or patterns to be mapped: a matrix, with each row representing a compound.
Y	Property for each pattern, either a numerical vector or matrix, or a class matrix. In the latter case, the Tanimoto distance is used for Y; in all other cases (also for combinations of numerical and class properties) the Euclidean distance is used.
grid	A grid for the representatives: see 'somgrid'.
rlen	the number of times the complete data set will be presented to the network.
alpha	a vector of two numbers indicating the amount of change. Default is to decline linearly from 0.05 to 0.01 over rlen updates.
radius	the initial radius of the neighbourhood to be used for each update: the decrease is exponential over rlen updates in such a way that after one-third of the updates only the winning unit is updated. The default is to start with a value that covers 2/3 of all units.
xweight	weight of X matrix in determining the distances of objects to units.
trwidth	width of the triangle function used in the WCC measure, given in the number of data points.
toroidal	if TRUE, then the edges of the map are joined. Note that in a toroidal hexagonal map, the number of rows must be even.
keep.data	store training data and their mapping in the network.

Value

an object of class `"wccsom"` with components

<code>grid</code>	the grid, an object of class <code>"somgrid"</code> .
<code>changes</code>	vector of mean average deviations from code vectors
<code>codes</code>	a matrix of code vectors.
<code>trwidth</code>	the triangle width used for the WCC measure
<code>acors</code>	autocorrelations of the code vectors.
<code>toroidal</code>	setting of parameter <code>'toroidal'</code> .
<code>FineTune</code>	setting of parameter <code>'FineTune'</code> .
<code>unit.classif</code>	mapping of training data: a vector of unit numbers. Only if <code>keep.data</code> equals <code>TRUE</code> .
<code>wccs</code>	WCC values of all training data, compared to the best matching codebook vector. Only if <code>keep.data</code> equals <code>TRUE</code> .
<code>data.acors</code>	WAC values for training data. Only if <code>keep.data</code> equals <code>TRUE</code> .

Author(s)

Ron Wehrens

References

FIXME: this page is a copy of `wccsom`, should be edited further

See Also

[SOM](#), [plot.wccsom](#), [wccsom](#), [wcc](#)

Examples

```
data(degelder)
gr <- somgrid(5, 5, "hexagonal")
set.seed(7)
x <- wccxyf(degelder$patterns, degelder$properties[, "cell.vol"],
            grid=gr, trwidth=20, rlen=100)
```

Index

*Topic **classif**

- bucket, 2
- classvec2classmat, 3
- expand.som, 5
- plot.wccsom, 6
- predict.wccsom, 8
- summary.wccsom, 9
- unit.distances, 11
- wcc, 12
- wccassign, 13
- wccmap, 14
- wccsom, 15
- wccxyf, 17

*Topic **datasets**

- cepha, 3
- degelder, 4

bucket, 2

cepha, 3

classmat2classvec (classvec2classmat), 3

classvec2classmat, 3

debucket (bucket), 2

degelder, 4

expand.som, 5

plot.heatkey (plot.wccsom), 6

plot.wccchanges (plot.wccsom), 6

plot.wcccodes (plot.wccsom), 6

plot.wcccounts (plot.wccsom), 6

plot.wccmapping (plot.wccsom), 6

plot.wccpred (plot.wccsom), 6

plot.wccprop (plot.wccsom), 6

plot.wccquality (plot.wccsom), 6

plot.wccsom, 6, 9, 16, 18

predict.wccsom, 7, 8

SOM, 16, 18

summary.wccsom, 9

unit.distances, 7, 11

wac (wcc), 12

wacmat (wcc), 12

wcc, 12, 16, 18

wccassign, 13, 14

wccmap, 14

wccsom, 7, 9–11, 13, 14, 15, 18

wccxyf, 4, 7, 9, 11, 13, 14, 16, 17