Clustering time series data with tscR

A R package to cluster time series data, base on slope and Frechet distance

Pérez-Sanz, Fernando. Murcian Institute of biomedical research
Riquelme-Pérez, Miriam. CNRS - CEA, Univ. Paris-Saclay. MIRCen

Contents

Overview

El clustering es una técnica de aprendizaje no supervisado, ampliamente empleado en diversas áreas tales como machine learning, bioinformatics, image analysis o pattern recognition. El clustering en gene expression data es útil para encontrar grupos de genes con comportamiento similar y puede proporcionar información muy útil para el entendimiento de determinados procesos biológicos.

Este clustering de gene expression data puede realizarse sobre genes, muestras o sobre la variable tiempo. En este último caso, denominado time series genes expression, es posible identificar genes con similar dinámica y obtener conjuntos de respuestas ante determinadas situaciones. Esto ha permitido resaltar (insight) respuesta a stress ambiental, ritmos circadianos, respuesta a algún tratamiento, etc.

Existen en R algunos paquetes genéricos para el análisis del clustering de series temporales – *kmlShape* , *dtwclust*, *clue*, *TSclust* – sin embargo, no hay un único algoritmo de clustering que resuelva satisfactoriamente todos los problemas. Cada problema de clustering requiere una aproximación específica. Existen paquetes orientados al análsis de time series gene expression como *TSMixclust*, *ctsGE*, *MFuzz*. *Mfuzz* minimiza la alta sensibilidad al ruido de otros algoritmos mediante un fuzzy clustering, por su parte *TSmixClust* es un soft-clustering que emplea mixed-effects models with nonparametric smoothing spline fitting. *ctsGE* busca minimizar noisy data a traves de 2 pasos, primero se define grupos en base a sus perfiles de expresión (expression index) y posteriormente, los genes del mismo index son clustered aplicando kmeans. A pesar de que estos métodos buscan resolver la importante tarea de minimizar el efecto del ruido en el clustering, generalmente genes con similar expresión (en magnitud) terminan en los mismos grupos.

En ocasiones, sin embargo, el mayor interés del investigador es encontrar genes con patrones similares (similares trayectorias) aunque estas ocurran a niveles de expresión diferentes. Se busca por tanto genes con una evolución temporal similar independientemente de la magnitud de la expresión.

Supongamos que sometemos a un individuo a un tratamiento, y medimos la expresión de tres (o diez mil) de sus genes en el momento en que comenzamos a suministrarle el tratamiento, a la semana, a las dos semanas y a las 3 semanas (Fig. 1).

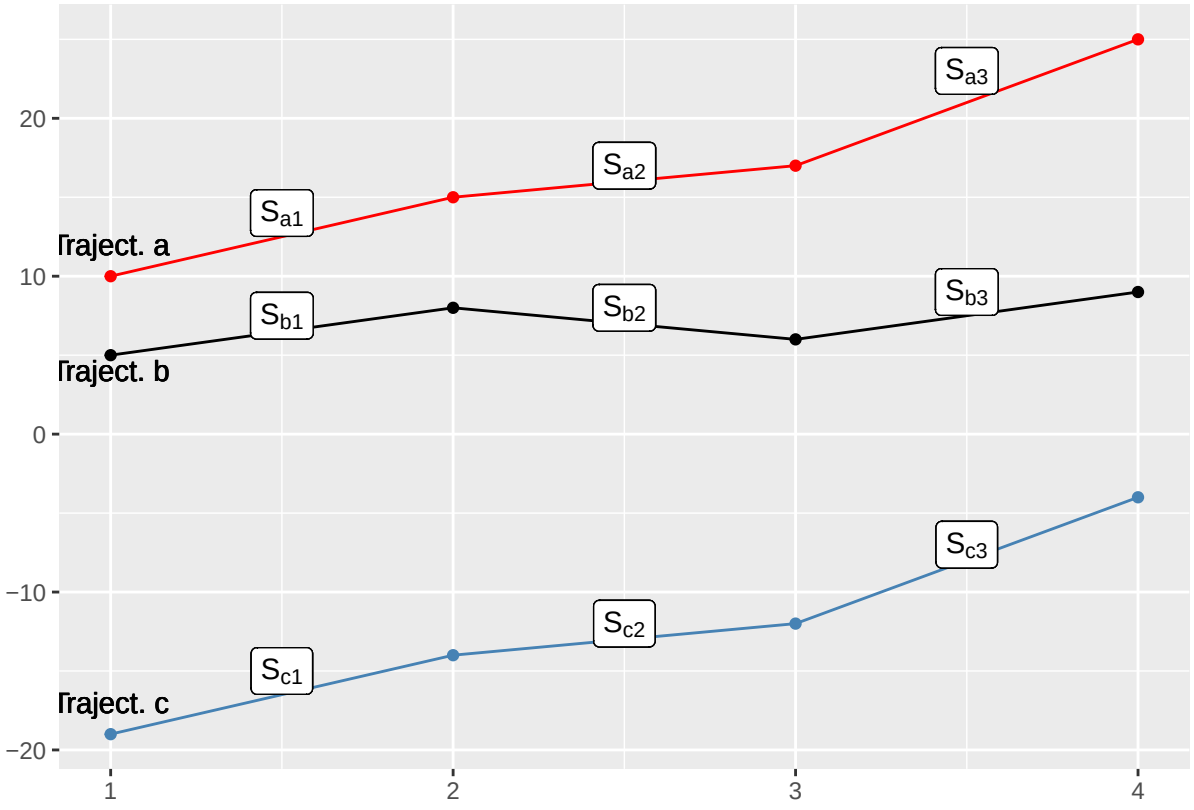


Figure 1: Figure 1

Las lineas de la figura 1 (a,b,c) representan 3 trajectorias (p.e la expresión de 3 genes en 4 momentos diferentes). Las tayectorias T_a y T_c son idénticas desde el punto de vista de sus pendientes, solo difieren en la magnitud de si expresión. Por otro lado la trayectoria T_b desde el punto de vista de la magnitud de expresión estaría más cerca de la trayectoria T_a que de la T_c.

Intuitivamente, si tuviésemos que decidir la distancia entre estas trayectorias, pensaríamos que T_a está más cerca de T_b que de T_c. A su vez, T_c está más cerca de T_b que de T_a. Se podrían cuantificar esas distancias con distintas métricas:

- Distancia euclídea

	a	b
b	21.24	
c	54.00	39.41

Otras medidas de distancia basadas más específicamente en series temporales como las distancias de Frechet o DTW arrojan resultados similares:

- Distancia de Frechet

	a	b
b	16	
c	29	24

- Distancia DTW

	a	b
b	42	
c	158	104

Sin embargo, si el interés es conocer qué trayectorias tienen similar comportamiento a lo largo del tiempo independientemente de la magnitud de su expresión, hay que recurrir a criterios puramente geométricos y buscar similitud basada es sus pendientes. En este caso la matriz de distancias quedaría de la siguiente manera:

	a	b
b	6.71	
c	0.00	6.71

Donde se puede ver que la distancia entre la trayectoria a y c es cero pues son dos líneas con idénticas pendientes y la distancia de ambas a b es la misma.

Por tanto nos podemos encontrar que un investigador esté interesado en identificar y agrupar conjuntos de genes con comportamientos similares desde diferentes puntos de vista:

- Genes agrupados con niveles de expresión similares, es decir genes cuyas trayectorias estén próximas en términos de distancia física (fig.2 B).
- Genes agrupados con similar evolución independientemente de la "distancia física" a la que se encuentren. Se Trata en este segundo caso de genes que responden de manera similar, pero con diferente intensidad (fig.2 C).
- Además podría ser de interés agrupar genes en función de ambos factores (distancia + tendencia) (fig.2 D).

Con este paquete proponemos una metodología que permite agrupar esas trayectorias basándonos en distancias físicas empleando métrica de distancia adaptada a series temporales (distancia de Frechet) y una nueva aproximación basada en similitudes de pendientes de las trayectorias, donde éstas se agrupan en función de dicha similitud independientemente de la distancia a la que se encuentren..

Además se da la opción de combinar ambas clasificaciones de manera que el resultado final sería el de trayectorias agrupadas en base a sus valores y a sus evoluciones.

Puesto que en gran cantidad de estudios (especialmente en diferentes disciplinas ómicas), el número de trayectorias puede ser muy grande (miles o decenas de miles), se ha desarrollado una metodología basada en la existente en el paquete *kmlShape*, en la que mediante un preagrupamiento, se obtiene una serie de "representantes" de las trayectorias, denominados senators, estos senators son entonces clasificados mediante alguno de los algoritmos propuestos. Finalmente el conjunto de trayectorias es asignado al cluster al que su senator ha sido asignado. De esta manera se reduce el coste computacional y el riesgo de desbordar la memoria.

Getting started

Install with `install_github`.

```
devtools::install_github("fpsanz/tscR")
```

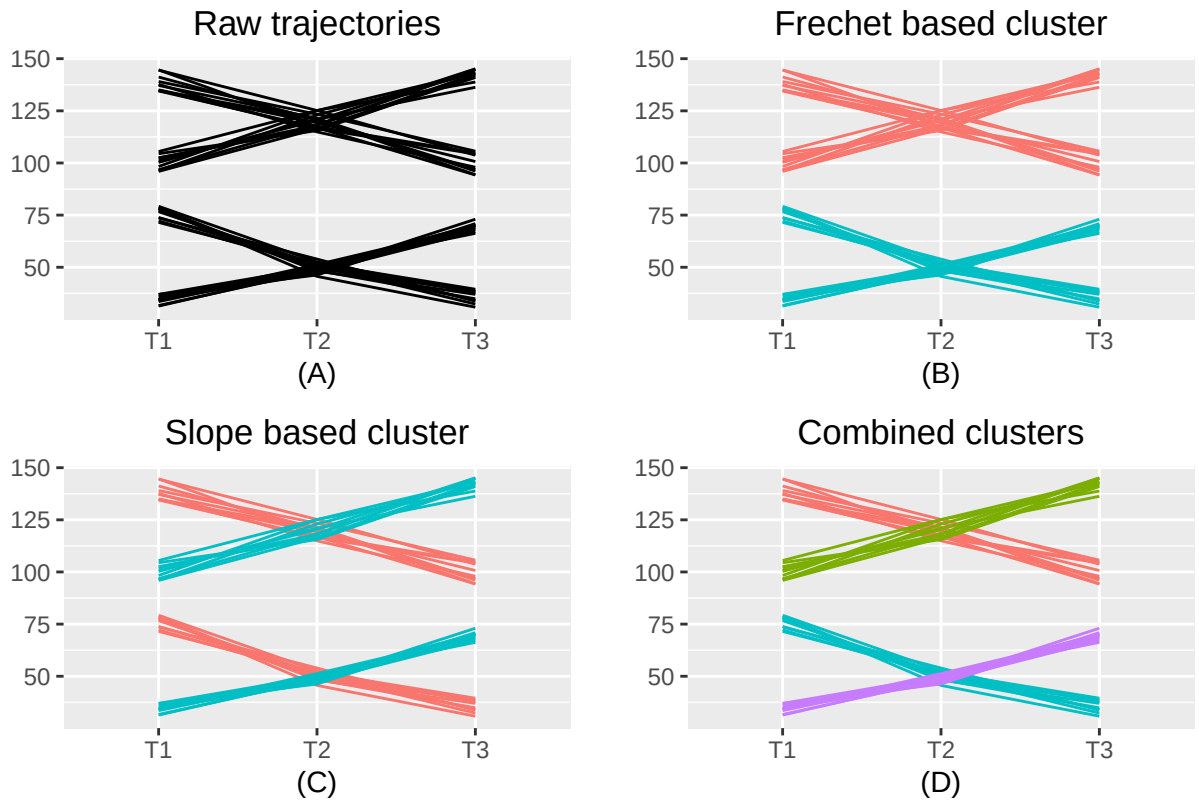


Fig. 1

Figure 2: Figure 2

Read the vignette (this document):

```
library(tscR)
```

```
browseVignettes("tscR")
```

Simple clustering

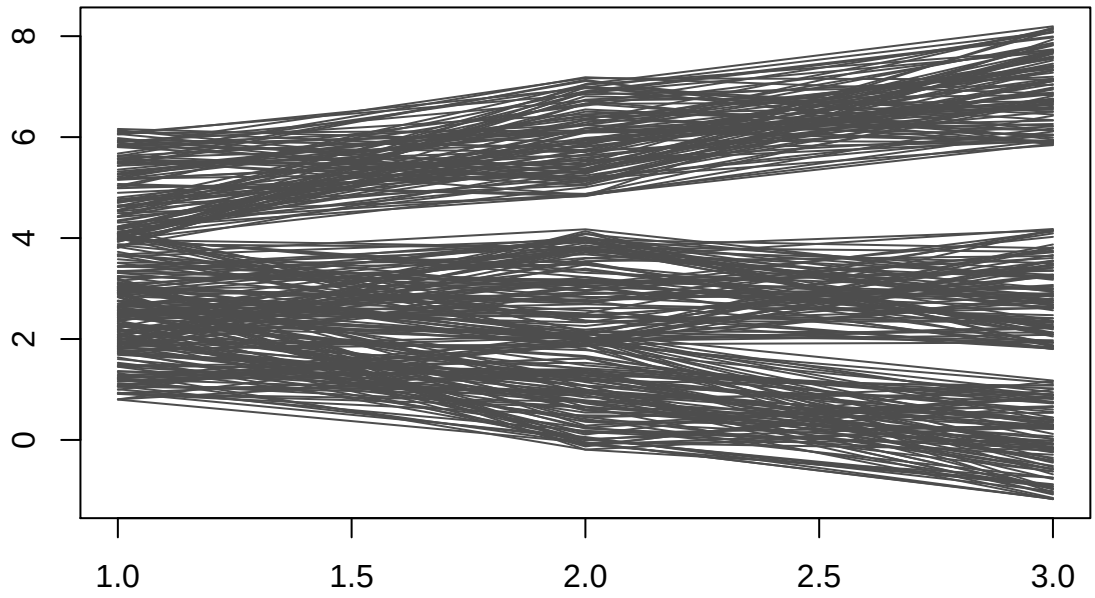
Based on slope distance

Los datos de entrada deben ser un dataframe o matrix donde las filas serán cada una de las trayectorias y las columnas los momentos en el tiempo (Table 1).

En primer lugar vamos a cargar un conjunto de trayectorias de ejemplo incluidas en la librería

```
data(tscR)
df <- tscR
head(df)
#>         T1        T2         T3
#> 1 6.161502 5.9001500  6.6022870
#> 2 3.344366 3.6755408  2.9476736
#> 3 1.058245 0.8805583 -0.5132939
#> 4 4.416688 6.7408719  7.1964777
#> 5 2.409223 3.7986534  2.1898294
#> 6 1.056814 1.8803695 -1.0379369
```

tscR contiene 300 observaciones (trayectorias) con datos tomados en 3 puntos temporales regulares (day 1, day 2, day 3).



A continuación se calcula la matrix de similitudes que será de tamaño n x n, siendo n el número de filas de la matriz de entrada. Esta matrix es un objeto de la clase `dist` y contiene las similitudes entre trayectorias basado en pendientes similares.

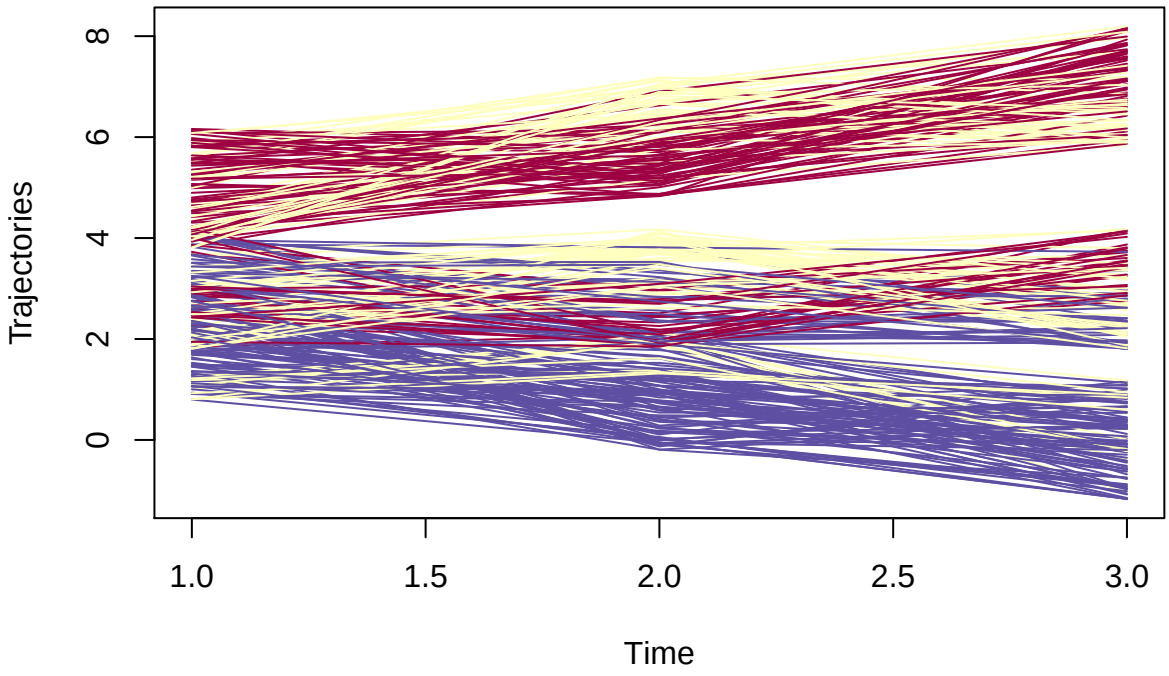
```
time <- c(1,2,3)
sDist <- slopeDist(df, time)
```

El siguiente paso sería agrupar las trayectorias basándonos en la similitud de sus pendientes independientemente de la distancia a la que se encuentren (meter referencia al paper).

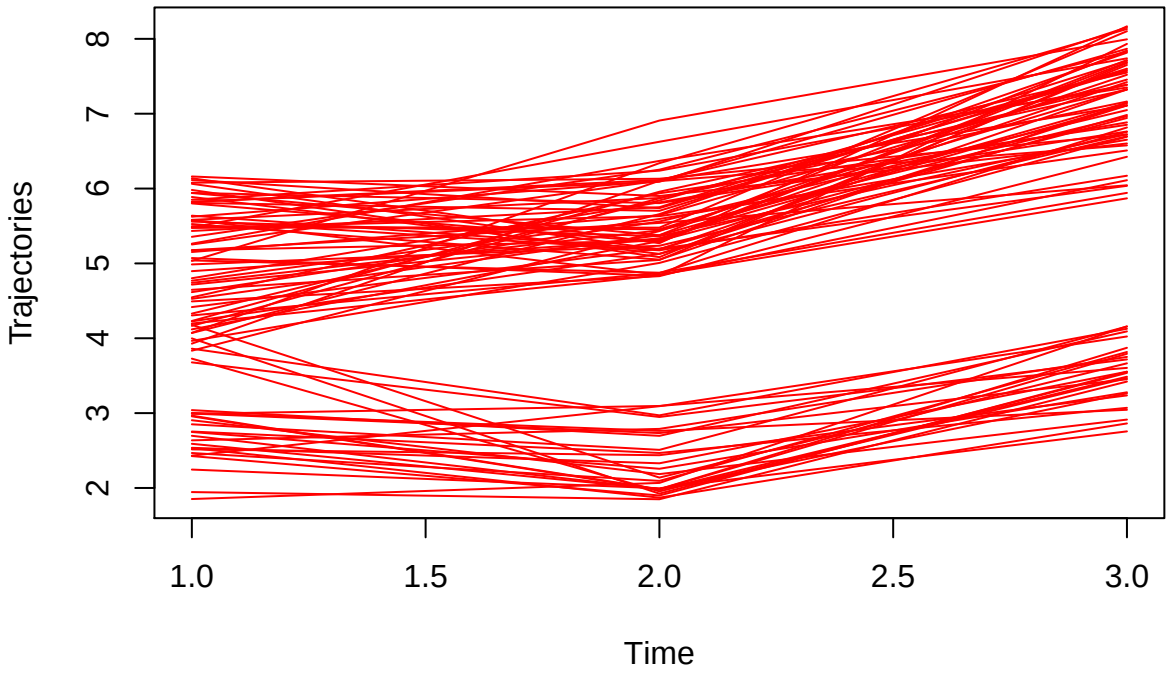
```
sclust <- getClusters(sDist, k = 3)
```

El resultado puede ser visualizado con la función `plotCluster`. Esta función toma como parámetros los datos originales (data), el objeto resultante de `getClusters` (cluster) y los cluster que se desean visualizar: "all" para visulizarlos todos en un gráfico único, un entero para visulizar las trayectorias de ese cluster o un vector definiendo que clusters se desean visualizar (uno por subplot)

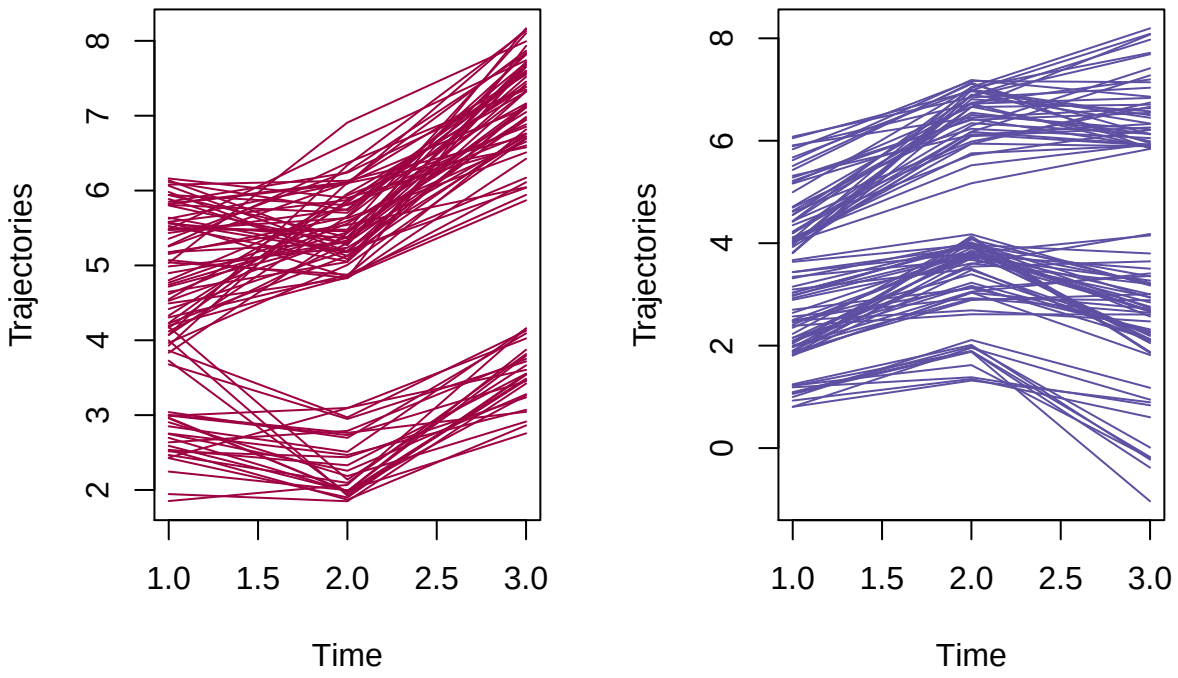
```
plotCluster(data = df, clust = sclust, ncluster = "all")
```




```
plotCluster(df, sclust, 1)
```




```
plotCluster(df, sclust, c(1:2))
```



Como puede observarse en este último gráfico, las trayectorias con evolución descedente-ascendente se han agrupado por un lado, independientemente de su distancia (plot izquierdo) y las de trayectoria ascendente-descentente por otro (plot derecho).

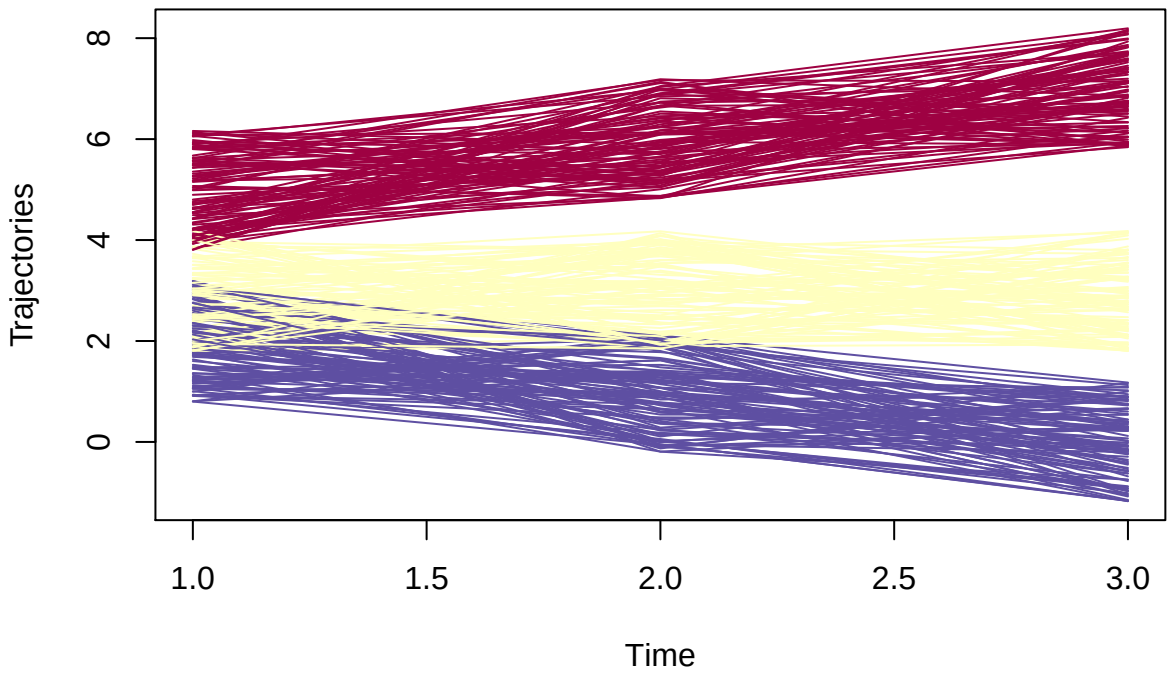
Based on Frechet distance

Es una medida de similitud entre curvas que tiene en cuenta la localización y orden de los puntos a lo largo de la curva (TODO: meter alguna referencia)

El procedimiento sería similar al caso anterior:

- Calcular la matriz de distancia.
- Obtener los clusters.
- Visualizar los resultados.

```
fdist <- frechetDistC(df, time)
fclust <- getClusters(fdist, 3)
plotCluster(df, fclust, "all")
```



Se observa que el agrupamiento tiene más que ver con la distancia (en términos euclídeos) entre trayectorias que con las pendientes en si mismas tal como se ilustra en el overview. En ciertas circustancias puede ser esta la clasificación que interese.

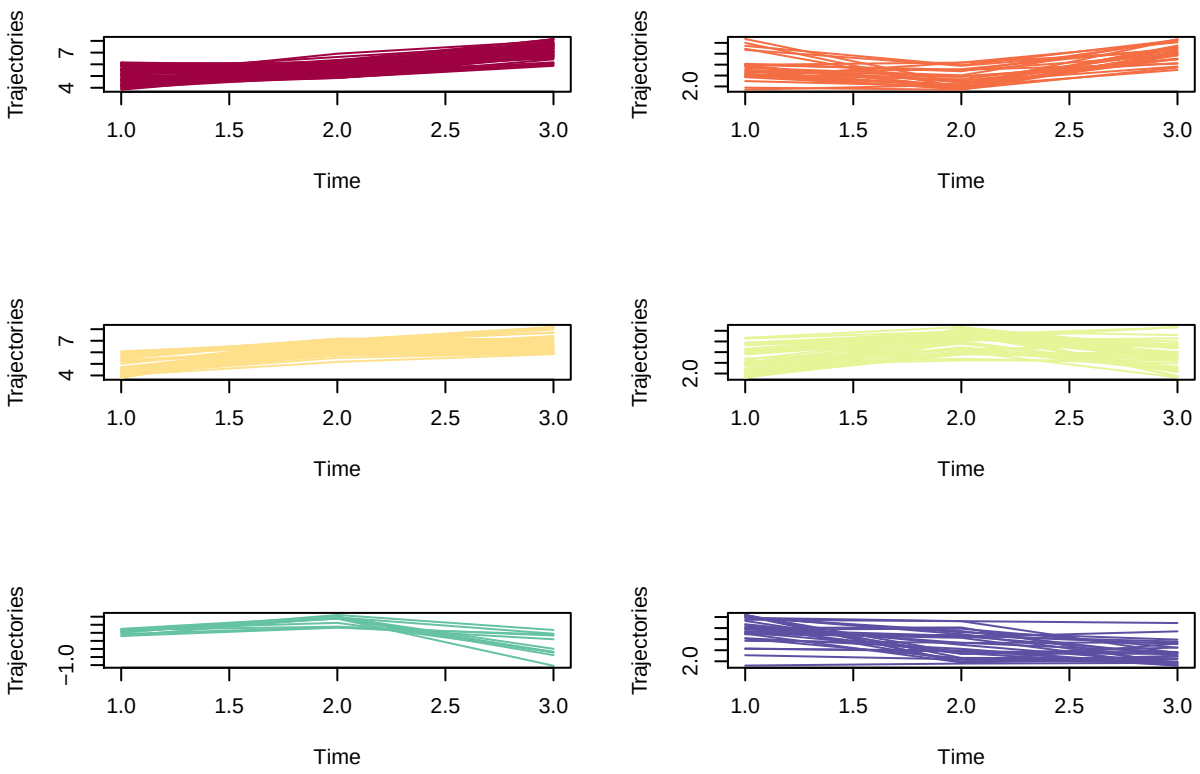
Combined clusters

Una tercera opción es combinar los resultados de ambos agrupamientos de manera que se obtendran conjuntos de trayectorias agrupados en función tanto de su distancia como de la similitud de sus pendientes.

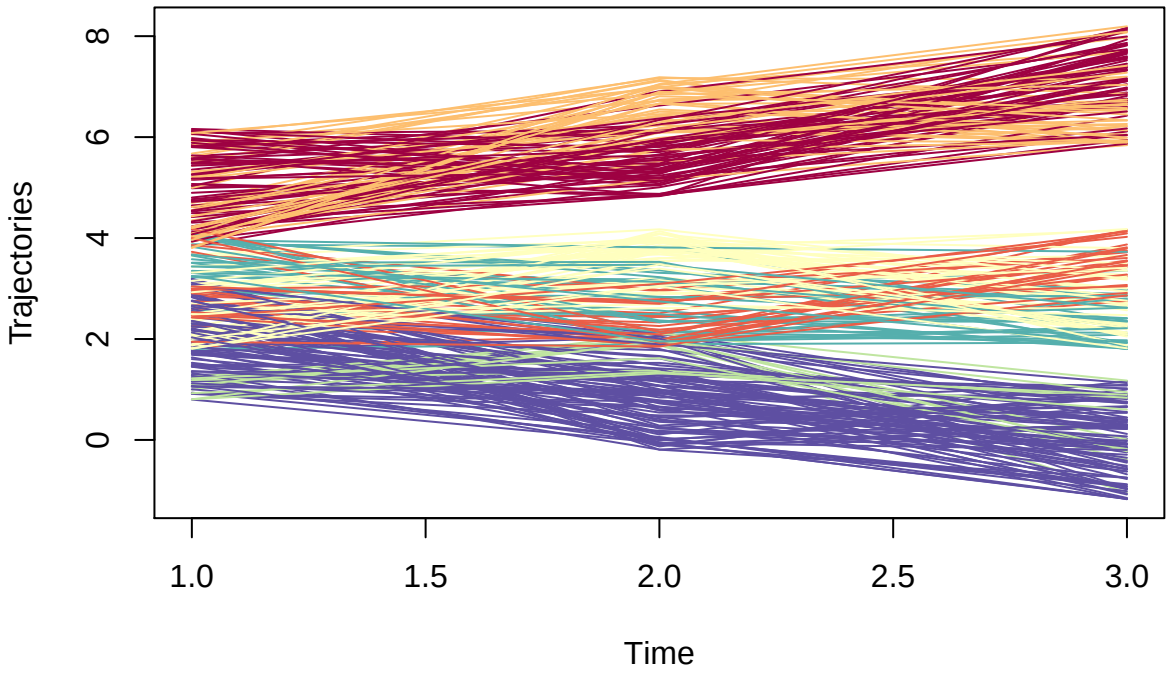
Para ello, la función `combineCluster` toma como entrada los objetos generados por getClusters generando una clasificación combinada.

De esta manera trayectorias muy cercanas pero con pendientes diferentes serán clasificadas en grupos diferentes y trayectorias con pendientes similares pero alejadas también se claficicarán en grupos diferentes. Así se obtiene una clasificación más fina (precisa) de las trayectorias.

```
ccluster <- combineCluster(sclust, fclust)
plotCluster(df, ccluster, c(1:6))
```




```
plotCluster(df, ccluster, "all")
```



Clustering large data

En los casos en los que se dispone de una cantidad de trayectorias elevadas (>1000), el coste computacional y de memoria puede ser muy elevado, principalmente cuando se trata de calcular las matrices de distancia. Nosotros proponemos hacer un clustering previo con la función `clara`, implementada dentro de `imputeSenators`, con un número de clusters elevado (p.e 100), esto generará 100 centroides, asumiendo que esos 100 centroides pueden representar al conjunto de formas de todas las trayectorias (de ahí el nombre de senators). Una vez obtenidos estos senators, se puede proceder con ellos como en el apartado anterior, agrupando en base distancia de Frechet, pendientes o ambas y generando el número de cluster deseados. Una vez obtenidos los clusters, como se ha guardado a que senator pertenecia cada trayectoria original, mediante `imputeSenatorToData` se asigna cada trayectoria a los clusters finales. Esta metodología permite clasificar un número muy elevado de trayectorias a un bajo coste computacional y de memoria.

El procedimiento, sería el siguiente:

En primer lugar obtener los senators

```
data( "tscR" )
bigDF <- tscR
senators <- imputeSenators( bigDF, k = 100 )
#> Setting k to 30 . 10% total of data
```

El objeto generado (`senators`) es una lista de 3 elementos

- $data: dataframe con los datos originales

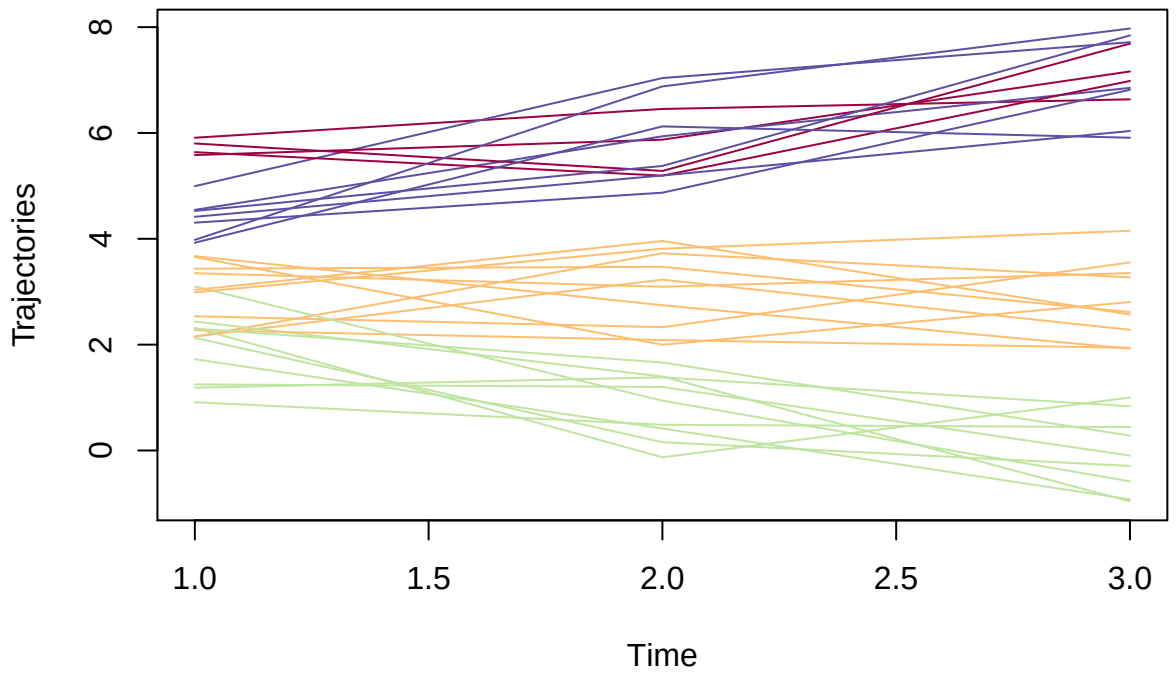
- $senatorData: es una matrix con las trajectorias de los senators
- $senatorCluster: vector con los clusters de los senators

Como ejemplo clasificaremos en función de la distancia basada en pendientes, aunque como se ha mencionado anteriormente se podría hacer en base cualquiera de las distancias e incluso combinarlas.

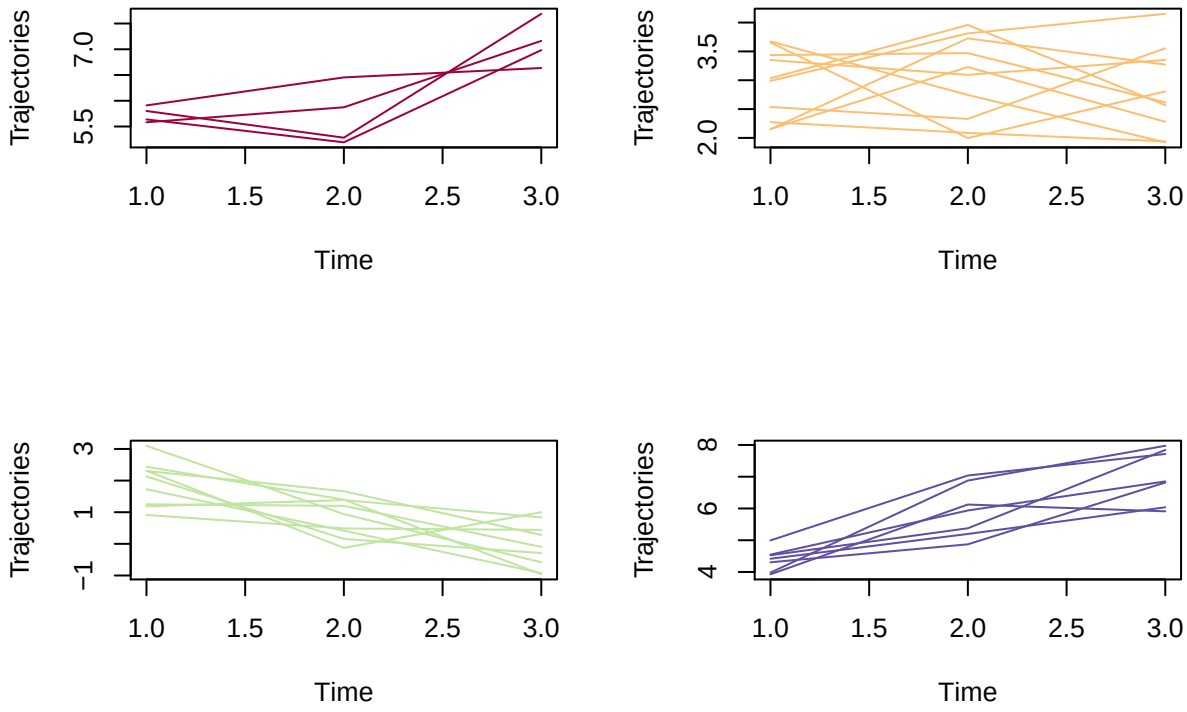
```
sdistSen <- frechetDistC( senators$senatorData, time = c( 1, 2, 3 ) )
cSenators <- getClusters( sdistSen, k = 4 )
```

Se podría visualizar la clasificación realizada a los senators

```
plotCluster(senators$senatorData, cSenators, "all")
```




```
plotCluster(senators$senatorData, cSenators, c(1,2,3,4))
```



Finalmente, hay que asignar los datos originales a los nuevos cluster

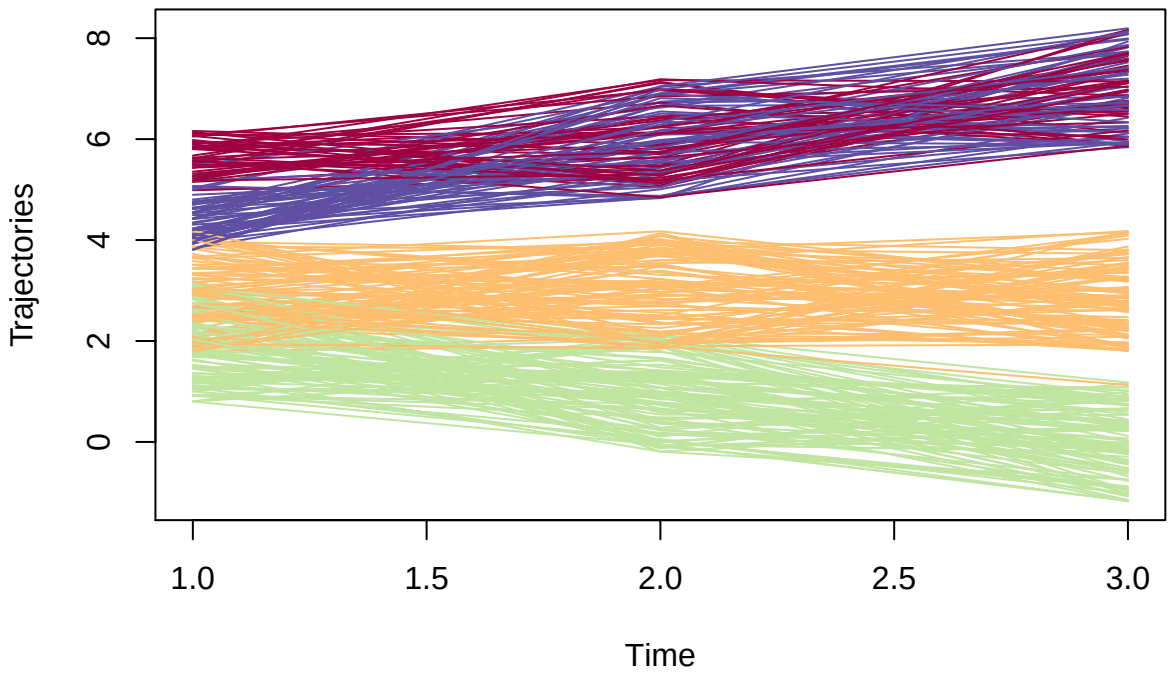
```
endCluster <- imputeSenatorToData(senators, cSenators)
```

Esto crea un objeto de la clase imputeSenator con 3 slots

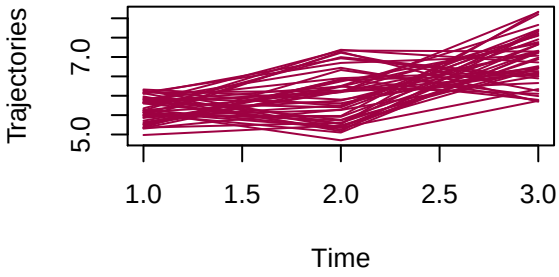
- @data: contiene el data frame con los datos originales
- @senators: identifica cada dato de data a que senator pertenece
- @endcluster: contiene los clusters finales a los que han sido asignados los datos

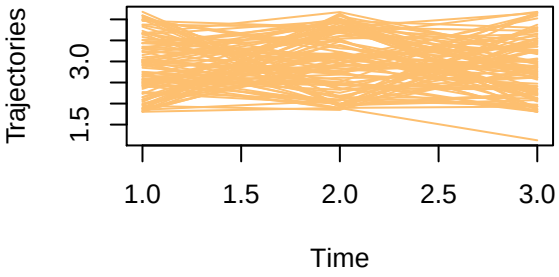
Y la visualización

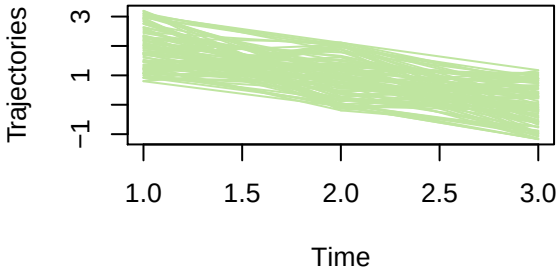
```
plotClusterSenator(endCluster, "all")
```

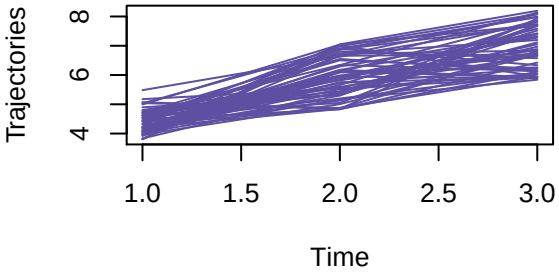



```
plotClusterSenator(endCluster, c(1,2,3,4))
```


Trajectories
5.0
7.0
1.0
1.5
2.0
2.5
3.0
Time


Trajectories
1.5
3.0
1.0
1.5
2.0
2.5
3.0
Time


Trajectories
−1
1
3
1.0
1.5
2.0
2.5
3.0
Time


Trajectories
4
6
8
1.0
1.5
2.0
2.5
3.0
Time