Trajectory Modeling

Charlotte Schenk

2024-09-22

Contents

The analyses are based on Ahrens et al. (in press) and have been adapted for this workshop.

Ahrens, K.F., Schenk, C., Kollmann, B.,… , Reif, A., Kalisch, R, & Plichta, M. M. (*in press*). Resilience to Major Life Events: Advancing Trajectory Modeling and Resilience Factor Identification by Controlling for Background Stressor Exposure. *American Psychologist.* https://doi.org/10.1037/amp0001315

Setups

Default chunk options

clear workspace

```r
rm(list = ls())
```

load packages

```r
library(dplyr)        # for data management
library(tidyr)        # for data management
library(kml)          # for k-mean clustering for longitudinal data
library(ggplot2)      # for plotting trajectories
library(knitr)        # for nice tables
library(tidyverse)    # for statistics
```

Data Preparation

Set Path

Please use individual path!

```
# path <- "/home/user/file/"
```

Read Data

```
raw_data <- read.csv(file = paste0(path, "ressymp_workshop.csv"), head = TRUE)
```

Select Relevant Data

```
data_long <- raw_data %>% select(subject.ID,         # ID
                                 beep,               # Time
                                 SR,                 # Stressor Reactivity
                                 starts_with("GE_")) # General events
```

Check Data Structure

```
kable(head(data_long[, 1:8]))    # column 1 to 8
```

subject.ID	beep	SR	GE_01	GE_02	GE_03	GE_04	GE_05
9	1	-0.8355115	1	0	0	0	1
9	2	-1.0140440	1	0	0	0	0
9	4	-0.9148592	1	0	0	0	0
9	5	-0.2582584	0	0	0	0	0
9	6	-0.7760006	1	0	0	0	0
11	1	-0.1174180	3	3	0	0	0

Stressor Lock

Reshape Dataframe (long to wide)

```r
data_wide <- data_long %>%
  pivot_wider(names_from  = beep,
              values_from = setdiff(names(data_long), c("beep", "subject.ID")))
                # or all columns except "beep" and "subject.ID"
```

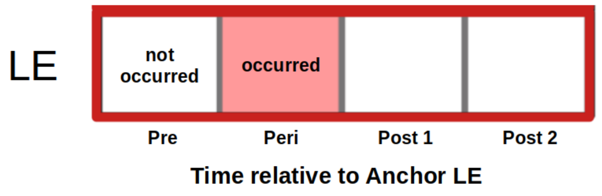
Check Data Structure

```r
kable(head(data_wide[,1:8]))  # column 1 to 8
```

subject.ID	SR__1	SR__2	SR__4	SR__5	SR__6	SR__3	GE__01__1
9	-0.8355115	-1.0140440	-0.9148592	-0.2582584	-0.7760006	NA	1
11	-0.1174180	-0.6351603	-1.0140440	-1.0338809	NA	-0.6093716	3
12	-0.4526643	0.8010267	-0.6470637	-0.0222032	0.7930932	0.9260000	3
15	0.0055672	0.3904026	0.6224942	1.1124660	-0.0420402	0.1642627	5
16	-1.3096163	-0.5935046	NA	-0.6073898	1.4358088	NA	5
17	-0.2007357	0.5828203	1.0668404	0.3170065	0.9934444	0.9874926	4

Find Anchor Experience

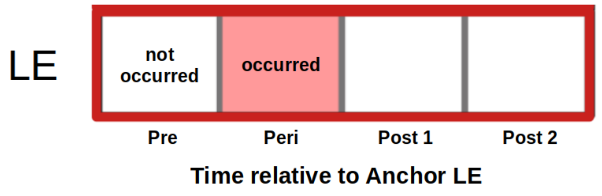
It is proposed to use only binary-coded life events in this approach (occurred: Yes = 1, No = 0). In the Stressor Lock approach, the anchor life event must have occurred at the anchor time point, but not before.



However, the dataset used only captures the level of burden, not the occurrence:

- 0 = *This situation did not happen*
- 1 = *Not at all burdensome*
- 2 = *Hardly burdensome*
- 3 = *Somewhat burdensome*
- 4 = *Quite burdensome*
- 5 = *Very burdensome*

Therefore, the anchor life event is selected if it was rated as 'Quite burdensome' or 'Very burdensome' at the anchor time point, and was rated as less than 'Quite burdensome' at the previous time point. This approach is a compromise to make the data usable for this method. As a result, the increase in stressor reactivity to the anchor life event appears smaller, but the method can still be demonstrated.



Stressor Lock Loop

```r
data_wide$anchor_time <- NA               # represents the anchor time point
data_wide$anchor_GE   <- NA               # represents the experience at the anchor time point
(GE_number   <- sprintf("%02d", 1:11))    # numbering of the general experiences
```

```
##  [1] "01" "02" "03" "04" "05" "06" "07" "08" "09" "10" "11"
```

```r
for(i in 1:nrow(data_wide)){          # for every participant
  for(j in 1:3){                      # for time 1 to time 3
    for(k in 1:length(GE_number)){    # for every general event

      if(!is.na(data_wide[i, paste0("GE_", GE_number[k], "_", j + 1)]) &
                data_wide[i, paste0("GE_", GE_number[k], "_", j + 1)] > 3 &
        # if at time j + 1 a GE is not missing AND > 3 (> "Somewhat burdensome")

        !is.na(data_wide[i, paste0("GE_", GE_number[k], "_", j)]) &
                data_wide[i, paste0("GE_", GE_number[k], "_", j)] <= 3){
        # AND if at time j this GE is not missing AND <= 3 (<= "Somewhat burdensome")

        data_wide$anchor_time[i] <- j + 1  # time j + 1 is the anchor time point
        data_wide$anchor_GE[i]   <- paste0("GE_", GE_number[k])
                                           # and this GE is a potential anchor event
      }
    }
  }
}
```

Frequencies of Anchor Events and Anchor Time Points

```r
kable(table(data_wide$anchor_GE),
      caption   = "Frequency of Anchor Events",
      col.names = c("General Experience", "Frequency"))
```

Table 3: Frequency of Anchor Events

General Experience	Frequency
GE_01	13
GE_02	17
GE_03	4
GE_04	10
GE_05	21
GE_06	9
GE_07	27
GE_08	9
GE_09	6
GE_10	10
GE_11	1

```r
kable(table(data_wide$anchor_time),
      caption   = "Frequency of Anchor Time Points",
      col.names = c("Time", "Frequency"))
```

Table 4: Frequency of Anchor Time Points

Time	Frequency
2	25
3	40
4	62

Match SR Score and Anchor Time Frame

```r
data_wide$SR_pre   <- NA
data_wide$SR_peri  <- NA
data_wide$SR_post1 <- NA
data_wide$SR_post2 <- NA

for(i in 1:nrow(data_wide)){
  for(t in 1:3){
    if(!is.na(data_wide$anchor_time[i]) &
       (data_wide$anchor_time[i] == t + 1)){
      data_wide$SR_pre[i]   <- as.numeric(data_wide[i, paste0("SR_", t)])
      data_wide$SR_peri[i]  <- as.numeric(data_wide[i, paste0("SR_", t + 1)])
      data_wide$SR_post1[i] <- as.numeric(data_wide[i, paste0("SR_", t + 2)])
      data_wide$SR_post2[i] <- as.numeric(data_wide[i, paste0("SR_", t + 3)])
    }
  }
}
```

Reduce Dataset to the Relevant Sample and Variables

```r
# Only if an anchor time point has been identified
data_wide_anchor <- data_wide[complete.cases(data_wide[, c("anchor_time")]), ]

# Only the following variables
data_wide_SR <- data_wide_anchor %>%
  select("subject.ID"        |
         starts_with("anchor_") |
         starts_with("SR_p"))
```

Check Data Structure

```r
kable(head(data_wide_SR))
```

subject.ID	anchor_time	anchor_GE	SR_pre	SR_peri	SR_post1	SR_post2
12	4	GE_10	0.9260000	-0.6470637	-0.0222032	0.7930932
15	4	GE_10	0.1642627	0.6224942	1.1124660	-0.0420402
18	4	GE_09	-0.1967658	-0.4625795	-0.7879041	1.1898320
21	3	GE_05	2.5645263	3.3222937	NA	2.9255548
26	4	GE_07	-0.7264115	-0.2027175	0.6800232	-0.8732036
37	4	GE_05	-0.4010869	0.1979849	-1.2044799	NA

Clustering Trajectories

We're using the `kml` package. For more information, please refer to the following links:

1. https://cran.r-project.org/web/packages/kml/kml.pdf
2. https://cran.r-project.org/web/packages/longitudinalData/longitudinalData.pdf
3. https://www.jstatsoft.org/article/view/v065i04

Data Preparation

`clusterLongData` (or `cld` in short) is the constructor for a object of class ClusterLongData.

Arguments:

- `traj [matrix(numeric)] or [data.frame]`: structure containning the trajectories. Each line is the trajectory of an individual. The columns refer to the time during which measures were made.
- `idAll [vector(character)]`: single identifier for each trajectory (ie each 'individual).
- `timeInData [vector(numeric)]`: precise the column containing the trajectories.

```
cldSDQ <- cld(traj       = data_wide_SR[, c("SR_pre", "SR_peri",
                                            "SR_post1", "SR_post2")],
              idAll      = data_wide_SR$subject.ID,
              timeInData = 1:4)
```

Building partition with kml

`kml` is a implementation of k-means for longitudinal data (or trajectories).

Arguments:

- `object [ClusterLongData]` (see above)
- `nbClusters [vector(numeric)]`: Vector containing the number of clusters with which kml must work. (Default is 2:6 and maximum number of cluster is 26)
- `nbRedrawing [numeric]` Sets the number of time that k-means must be re-run (with different starting conditions) for each number of clusters.
- `toPlot [character]`: either 'traj' for plotting trajectories alone, 'criterion' for plotting criterion alone, 'both' for plotting both or 'none' for not display anything

```
kml(object      = cldSDQ,
    nbClusters  = 2:5,
    nbRedrawing = 100,
    toPlot      = "none")
```

Save Cluster

```
data_wide_SR$cluster2 <- getClusters(cldSDQ, 2)
data_wide_SR$cluster3 <- getClusters(cldSDQ, 3)
data_wide_SR$cluster4 <- getClusters(cldSDQ, 4)
data_wide_SR$cluster5 <- getClusters(cldSDQ, 5)
```

Check the best cluster number

```r
criterions <- (rbind(qualityCriterion(cldSDQ@traj, getClusters(cldSDQ,2))$criters,
                     qualityCriterion(cldSDQ@traj, getClusters(cldSDQ,3))$criters,
                     qualityCriterion(cldSDQ@traj, getClusters(cldSDQ,4))$criters,
                     qualityCriterion(cldSDQ@traj, getClusters(cldSDQ,5))$criters))
colnames(criterions) <- c("CH", "CH2", "CH3", "RT", "DB", "BIC", "BIC2",
                          "AIC", "AICc", "AICc2", "pPG", "random")
rownames(criterions) <- c("2", "3", "4", "5")
kable(criterions, digits = 2)
```

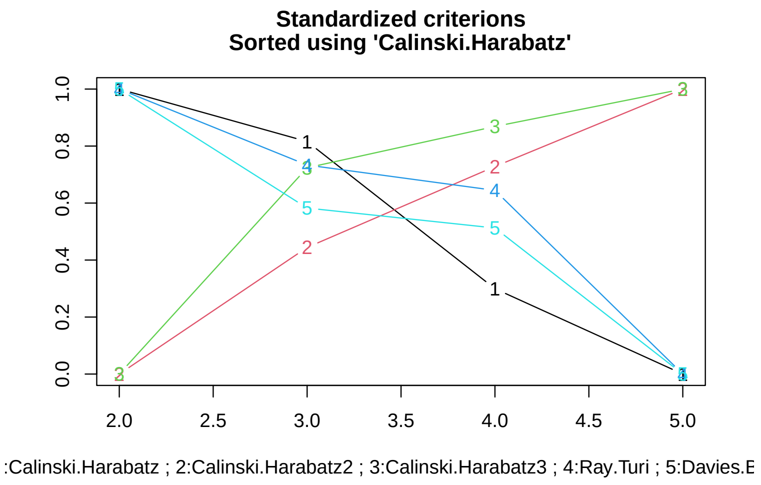
	CH	CH2	CH3	RT	DB	BIC	BIC2	AIC	AICc	AICc2	pPG	random
2	124.80	1.01	124.80	-0.05	-1.10	-1193.03	-1205.50	-1167.43	-1168.97	-1167.79	0.95	-0.47
3	118.81	1.95	168.02	-0.08	-1.18	-1140.35	-1158.37	-1103.37	-1106.59	-1104.11	0.93	-1.27
4	102.00	2.55	176.68	-0.09	-1.19	-1131.46	-1155.03	-1083.11	-1088.72	-1084.36	0.90	0.27
5	92.26	3.12	184.52	-0.17	-1.29	-1143.82	-1172.93	-1084.09	-1092.89	-1085.99	0.86	-0.45

> "IMPORTANT NOTE: Some criterion should be maximized, some other should be minimized. This might be confusing for the non expert. In order to simplify the comparison of the criterion, `qualityCriterion` compute the OPPOSITE of the criterion that should be minimized (Ray & Bouldin, Davies & Turi, BIC and AIC). **Thus, all the criterion computed by this function should be maximized.**"

See: https://cran.r-project.org/web/packages/longitudinalData/longitudinalData.pdf

Displayed graphically and standardized (range 0 to 1)

```r
plotAllCriterion(cldSDQ)
```



Plot Cluster

Reshape wide to long

```r
data_long_cluster <- data_wide_SR %>%
  pivot_longer(
    cols = starts_with("SR_"),
    names_to = c(".value", "time"),
    names_pattern = "(SR)_(.*)")
```

Frequencies within the clusters

```r
tab <- data_long_cluster %>%
  count(cluster2) %>%
  mutate(Percent = n / sum(n) * 100) %>%
  rename(N = n)

tab <- tab %>%
  select(cluster2, N, Percent)

kable(tab, col.names = c("Cluster", "Absolute Frequency (N)", "Relative Frequency (%)"))
```

Cluster	Absolute Frequency (N)	Relative Frequency (%)
A	356	70.07874
B	152	29.92126

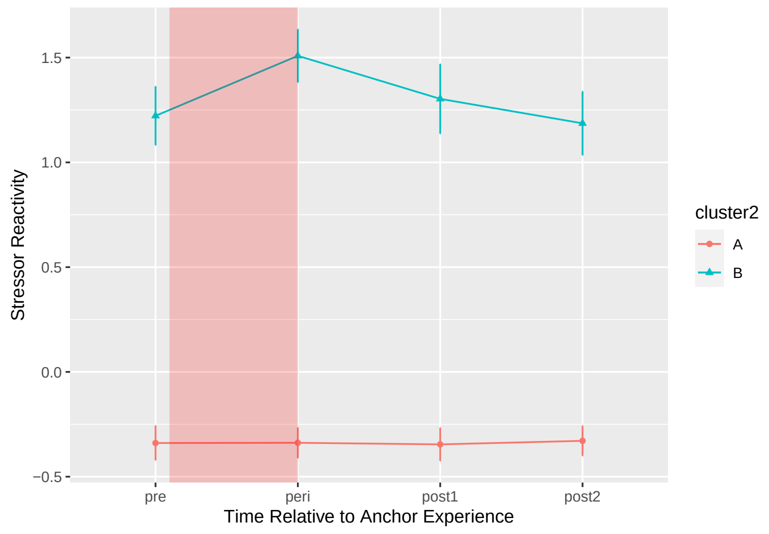
Cluster trajectories

```
data_long_cluster$time <- factor(data_long_cluster$time,
                                 levels = unique(data_long_cluster$time))

stats_cluster2 <- data_long_cluster %>%
  group_by(time, cluster2) %>%
  summarise(SR_mean = mean(SR, na.rm = TRUE),
            SR_se   = sd(SR, na.rm = TRUE) / sqrt(length(SR)))
```

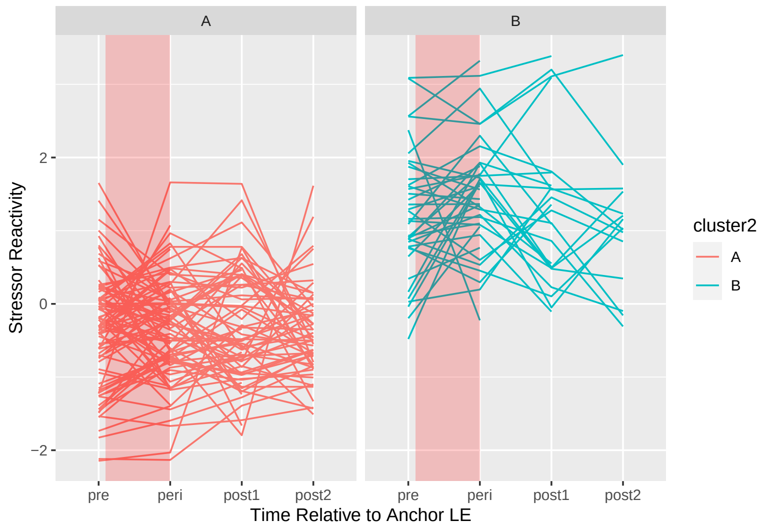
```
## `summarise()` has grouped output by 'time'. You can override using the
## `.groups` argument.
```

```
(cluster_trajectories <- ggplot(stats_cluster2,
             aes(x     = factor(time),
                 y     = SR_mean,
                 color = cluster2,
                 shape = cluster2)) +
  geom_line(aes(group  = cluster2)) +
  geom_point(aes(group  = cluster2)) +
  ylab("Stressor Reactivity") +
  xlab("Time Relative to Anchor Experience") +
  geom_errorbar(aes(ymin = SR_mean - SR_se, ymax = SR_mean + SR_se), width= .0) +
  annotate("rect",
           xmin = c(1.1),
           xmax = c(2),
           ymin = -Inf, ymax = Inf,
           fill = "red", alpha = .2))
```



Individual trajectories

```r
(individual_trajectories <- ggplot(data = data_long_cluster ,
            aes(x     = time,
                y     = SR,
                group = subject.ID,
                shape = cluster2)) +
  geom_line(aes(col = cluster2)) +
  facet_grid(. ~ cluster2) +
  ylab("Stressor Reactivity") +
  xlab("Time Relative to Anchor LE") +
  annotate("rect",
           xmin = c(1.1),
           xmax = c(2),
           ymin = -Inf, ymax = Inf,
           fill = "red", alpha = .2))
```



More Advanced

Imputation of missing values using the `copyMean` method

You can find more information at the following link: https://cran.r-project.org/web/packages/longitudinalData/longitudinalData.pdf

```
impTrajs <- imputation(cldSDQ, method="copyMean")@traj
```