Untitled

Willem Feijen

16 februari 2018

library(bupaR)

## Loading required package: edeaR

## Loading required package: eventdataR

## Loading required package: processmapR

## Loading required package: xesreadR

##   
## Attaching package: 'bupaR'

## The following object is masked from 'package:stats':  
##   
## filter

## The following object is masked from 'package:utils':  
##   
## timestamp

library(edeaR)  
library(processmonitR)  
library(plotluck)  
library(DiagrammeR)  
library(data.table)  
#library(DiagrammeRsvg)  
  
csvLog<-read.table("W:/Uitbreiden\_ProcessMap/RefundProcessSelectie.csv",header = TRUE, sep = ";",na = c("n.v.t.","onbekend"))  
csvLog$status = "complete"  
csvLog$activity\_instance = 1:nrow(csvLog)  
csvLog$startmoment<-as.POSIXct(strptime(csvLog$startmoment, "%d-%m-%Y %H:%M"))  
csvLog$eindmoment<-as.POSIXct(strptime(csvLog$eindmoment, "%d-%m-%Y %H:%M"))  
  
  
lengteLogIngelezen<-nrow(csvLog)  
kolomNamen<-colnames(csvLog)  
csvLog$activity\_instance = 1:nrow(csvLog)  
csvLogStart<-data.table(csvLog)  
csvLogStart$timeStamp<-csvLogStart$startmoment  
csvLogStart$status<-"start"  
csvLogEind<-csvLog  
csvLogEind$timeStamp<-csvLogEind$eindmoment  
csvLogEind$status<-"eind"  
csvLog<-rbind(csvLogStart,csvLogEind)  
rm(csvLogStart,csvLogEind)  
  
csvLog<-na.omit(csvLog,cols=c("OrderNr","Status","timeStamp"))

#Gevonden kolomkoppen:

## [1] "OrderNr" "Status" "Employee"   
## [4] "startmoment" "Channel" "RandomWaarde1"   
## [7] "RandomWaarde2" "eindmoment" "status"   
## [10] "activity\_instance" "timeStamp"

#Omzetten naar een processmining Log

# Map met standaard functie

graphStandaard <- process\_map(eventLog.Base, type = performance(),render = FALSE )  
graphStandaard %>%  
 export\_graph(file\_name = "Z:/Dump/mapMetStandaardFunctie.svg", file\_type = "SVG")



# Map met frequency

fileName <- "frequency"  
agregationInst = list( enriched\_frequency(value = "absolute", columnName = "xxx") )  
map<-enriched\_process\_map(eventLog.Base, aggregationInstructions = agregationInst )

## Joining, by = c("act", "aid", "case")

## Warning in bind\_rows\_(x, .id): binding factor and character vector,  
## coercing into character vector

## Warning in bind\_rows\_(x, .id): binding character and factor vector,  
## coercing into character vector

## Warning in min(nodes\_df$color\_level): no non-missing arguments to min;  
## returning Inf

## Warning in max(nodes\_df$color\_level[nodes\_df$color\_level < Inf]): no non-  
## missing arguments to max; returning -Inf

map <- map %>%  
 processmapR::color\_activities( column = "xxx",  
 colorCaption = "black",  
 colorUpperbound = "deeppink",  
 colorLowerbound = "green1") %>%  
 label\_activities(columns =c("activity\_name","xxx")) %>%  
 color\_transitions(column = "xxx",  
 colorCaption = "black",  
 colorUpperbound = "deeppink",  
 colorLowerbound = "darkgreen") %>%  
 label\_transitions(columns = c("xxx"))

## `select\_nodes()` INFO: created a new selection of 1 node

## `clear\_selection()` INFO: cleared an existing selection of 1 node

## `select\_nodes()` INFO: created a new selection of 1 node

map %>% export\_graph(file\_name = "Z:/Dump/Map met frequency.svg", file\_type = "SVG")



# Map met performance

agregationInst = list( enriched\_performance(FUN = mean, columnName = "xxx"))  
rm(map)  
map<-enriched\_process\_map(eventLog.Base, aggregationInstructions = agregationInst )

## Joining, by = c("act", "aid", "case")

## Warning in bind\_rows\_(x, .id): binding factor and character vector,  
## coercing into character vector

## Warning in bind\_rows\_(x, .id): binding character and factor vector,  
## coercing into character vector

## Warning in min(nodes\_df$color\_level): no non-missing arguments to min;  
## returning Inf

## Warning in max(nodes\_df$color\_level[nodes\_df$color\_level < Inf]): no non-  
## missing arguments to max; returning -Inf

map <- map %>%  
 processmapR::color\_activities( column = "xxx",  
 colorCaption = "black",  
 colorUpperbound = "deeppink",  
 colorLowerbound = "green1") %>%  
 label\_activities(columns =c("activity\_name","xxx")) %>%  
 color\_transitions(column = "xxx",  
 colorCaption = "black",  
 colorUpperbound = "deeppink",  
 colorLowerbound = "darkgreen") %>%  
 label\_transitions(columns = c("xxx"))

## `select\_nodes()` INFO: created a new selection of 1 node

## `clear\_selection()` INFO: cleared an existing selection of 1 node

## `select\_nodes()` INFO: created a new selection of 1 node

map %>%  
 export\_graph(file\_name = "Z:/Dump/Map met performance.svg", file\_type = "SVG")



# Map met aggregate en perf

agregationInst = list(enriched\_column\_aggregate( FUN = mean, columnNameIn = "RandomWaarde1", columnNameOut = "xxx1", edgeOperation = "to"),enriched\_column\_aggregate( FUN = mean, columnNameIn = "RandomWaarde2", columnNameOut = "xxx2", edgeOperation = "from"),enriched\_performance(FUN = mean, columnName = "perf")  
 )  
map<-enriched\_process\_map(eventLog.Base, aggregationInstructions = agregationInst )

## Joining, by = c("act", "aid", "case")

## Warning in bind\_rows\_(x, .id): binding factor and character vector,  
## coercing into character vector

## Warning in bind\_rows\_(x, .id): binding character and factor vector,  
## coercing into character vector

## Warning in min(nodes\_df$color\_level): no non-missing arguments to min;  
## returning Inf

## Warning in max(nodes\_df$color\_level[nodes\_df$color\_level < Inf]): no non-  
## missing arguments to max; returning -Inf

## Adding missing grouping variables: `case`

## Adding missing grouping variables: `case`

# Map met xxx2

map <- map %>%  
 processmapR::color\_activities( column = "xxx2",  
 colorCaption = "black",  
 colorUpperbound = "gray50",  
 colorLowerbound = "gray100") %>%  
 label\_activities(columns =c("activity\_name","xxx2")) %>%  
 color\_transitions(column = "xxx2",  
 colorCaption = "black",  
 colorUpperbound = "gray0",  
 colorLowerbound = "gray50") %>%  
 label\_transitions(columns = c("xxx2"))

## `select\_nodes()` INFO: created a new selection of 1 node

## `clear\_selection()` INFO: cleared an existing selection of 1 node

## `select\_nodes()` INFO: created a new selection of 1 node

map %>%  
 export\_graph(file\_name = "Z:/Dump/Map met xxx2.svg", file\_type = "SVG")



# xxx1xxx2activiteit

map <- map %>%  
 color\_activities( column = "xxx2",  
 colorCaption = "black",  
 colorUpperbound = "lightblue",  
 colorLowerbound = "green1") %>%  
 label\_activities(columns =c("activity\_name","xxx1","xxx2")) %>%  
 color\_transitions(column = "xxx2",  
 colorCaption = "black",  
 colorUpperbound = "deeppink",  
 colorLowerbound = "darkgreen") %>%  
 label\_transitions(columns = c("xxx1","xxx2"))

## `select\_nodes()` INFO: modified an existing selection of 1 node:  
## \* 2 nodes are now in the active selection  
## \* used the `union` set operation

## `clear\_selection()` INFO: cleared an existing selection of 2 nodes

## `select\_nodes()` INFO: created a new selection of 1 node

# Map met max en perf

agregationInst = list(enriched\_column\_aggregate( FUN = sum, columnNameIn = "RandomWaarde1", columnNameOut = "xxx1", edgeOperation = "to"),enriched\_column\_aggregate( FUN = mean, columnNameIn = "RandomWaarde2", columnNameOut = "xxx2", edgeOperation = "mean"),enriched\_performance(FUN = mean, columnName = "perf")  
 )  
map<-enriched\_process\_map(eventLog.Base, aggregationInstructions = agregationInst )

## Joining, by = c("act", "aid", "case")

## Warning in bind\_rows\_(x, .id): binding factor and character vector,  
## coercing into character vector

## Warning in bind\_rows\_(x, .id): binding character and factor vector,  
## coercing into character vector

## Warning in min(nodes\_df$color\_level): no non-missing arguments to min;  
## returning Inf

## Warning in max(nodes\_df$color\_level[nodes\_df$color\_level < Inf]): no non-  
## missing arguments to max; returning -Inf

## Adding missing grouping variables: `case`

## Adding missing grouping variables: `case`

# Map met xxx2

map <- map %>%  
 color\_activities( column = "xxx2",  
 colorCaption = "black",  
 colorUpperbound = "gray50",  
 colorLowerbound = "gray100") %>%  
 label\_activities(columns =c("activity\_name","xxx1","xxx2")) %>%  
 color\_transitions(column = "xxx2",  
 colorCaption = "black",  
 colorUpperbound = "gray0",  
 colorLowerbound = "gray50") %>%  
 label\_transitions(columns = c("xxx2"))

## `select\_nodes()` INFO: created a new selection of 1 node

## `clear\_selection()` INFO: cleared an existing selection of 1 node

## `select\_nodes()` INFO: created a new selection of 1 node

map %>%  
 export\_graph(file\_name = "Z:/Dump/Map met xxx2.svg", file\_type = "SVG")



# xxx1xxx2activiteit

map <- map %>%  
 color\_activities( column = "xxx2",  
 colorCaption = "black",  
 colorUpperbound = "lightblue",  
 colorLowerbound = "green1") %>%  
 label\_activities(columns =c("activity\_name","xxx1","xxx2")) %>%  
 color\_transitions(column = "xxx2",  
 colorCaption = "black",  
 colorUpperbound = "deeppink",  
 colorLowerbound = "darkgreen") %>%  
 label\_transitions(columns = c("xxx1","xxx2"))

## `select\_nodes()` INFO: modified an existing selection of 1 node:  
## \* 2 nodes are now in the active selection  
## \* used the `union` set operation

## `clear\_selection()` INFO: cleared an existing selection of 2 nodes

## `select\_nodes()` INFO: created a new selection of 1 node

map %>% export\_graph(file\_name = "Z:/Dump/xxx1xxx2activiteit.svg", file\_type = "SVG")



# xxx2 blind

map <- map %>%  
 color\_activities( column = "xxx2",  
 colorCaption = "black",  
 colorUpperbound = "lightblue",  
 colorLowerbound = "green1") %>%  
 label\_activities(columns =c()) %>%  
 color\_transitions(column = "xxx2",  
 colorCaption = "black",  
 colorUpperbound = "deeppink",  
 colorLowerbound = "darkgreen") %>%  
 label\_transitions(columns = c())

## `select\_nodes()` INFO: modified an existing selection of 1 node:  
## \* 2 nodes are now in the active selection  
## \* used the `union` set operation

## `clear\_selection()` INFO: cleared an existing selection of 2 nodes

## `select\_nodes()` INFO: created a new selection of 1 node

map %>% export\_graph(file\_name = "Z:/Dump/xxx2blind.svg", file\_type = "SVG")

