**Intro**

A week ago, [Havard Business Review](https://hbr.org/2019/04/what-process-mining-is-and-why-companies-should-do-it) published an article on process mining and provided reasons for companies to adopt it. If you need a refresher on the concepts of process mining, you can refer to my [first post](https://notast.netlify.com/post/process-mining-part-1-3-introduction-to-bupar-package/). Conducting process mining is easy with R’s [bupaR](https://www.bupar.net/) package. bupaR allows you to create a variety of visualizations as you analyse event logs. It includes [visualizations of workflow on the ground](https://notast.netlify.com/post/process-mining-part-2-3-more-on-bupar-package/) which you can then compare them against theoretical models to discover deviations. However, there are some analysis and visualizations which are not included in bupaR. I will cover these in this last post on process mining. *(The datasets used in this post are healthcare related.)*

**Interpruption Index**

The interruption index measures how much does a resource have to toggle between cases before completing the case instead of completing all the activities for a case before proceeding to the next case. The toggling between incomplete cases could be due to many reasons. However, if the reason is due to disruptions in the workflow, it will result in inefficiency and lost in productivity. *(If there is a proper term for what I’m doing, please let me know in the comments below)*

**Index components**

The interruption index is a ratio of two parameters, time block and case load.

Time block is where all the activity instances executed by a specific resource are arranged in chronological order within a defined time. For this post, it will be a day. The consecutive activity instances for a particular case are grouped together to form a time block. In the example below, there are four time blocks.

| **Activity** | **Resource** | **Time** | **Day** | **Case** | **Time Block** |
| --- | --- | --- | --- | --- | --- |
| Admission | Z | 10:40 | 1 | AA | 1 |
| Admission | Z | 11:11 | 1 | BB | 2 |
| X-ray | Z | 11:30 | 1 | BB | 2 |
| X-ray | Z | 12:00 | 1 | AA | 3 |
| Scans | Z | 12:44 | 1 | AA | 3 |
| Pre-op | Z | 13:59 | 1 | BB | 4 |

The caseload component in the ratio refers to the maximum number of cases seen by a specific resource on a particular day. In the example above, there is a maximum of 2 cases. Hence, the ratio is 2 (4/2= 2). The lowest score is 1 which means there is no toggling between cases and the day is the least interrupted.

**Calculating the index**

I will be using the sepsis eventlog from the bupaR package to illustrate the interprution index.

There are 26 resources providers in the dataset but we will only compare the index between 2 resources for simplification purposes.

# library

library(plyr)

library(tidyverse)

library(bupaR)

theme\_set(theme\_light())

n\_resources(sepsis)

## [1] 26

Let’s do some wrangling to derive a dataframe which we will work with.

#derive desired df

sepsis\_df<-sepsis %>% filter\_resource(c("A", "B")) %>% # filter 2 resources for our example

data.frame() %>% # convert event log object

select(case\_id, activity, lifecycle, resource, timestamp) %>% # select relevant variables

drop\_na() #drop na observation

In this post the timeframe for the index is a day so we will create a day identifier, ID\_day. The index requires us to calculate the caseload for each day for each resource.

sepsis\_df<-sepsis\_df %>% mutate(Date= as.Date(timestamp)) %>% mutate("ID\_day" = group\_indices\_(., .dots = c("resource","Date"))) %>% # add ID\_day

group\_by(ID\_day) %>% arrange(ID\_day, timestamp) %>% select(resource, ID\_day, case\_id) %>%

mutate(caseload = n\_distinct(case\_id)) %>% # caseload

ungroup()

sepsis\_df %>% head()

## # A tibble: 6 x 4

## resource ID\_day case\_id caseload

##

## 1 A 1 XJ 1

## 2 A 1 XJ 1

## 3 A 1 XJ 1

## 4 A 1 XJ 1

## 5 A 2 WEA 1

## 6 A 2 WEA 1

We now calculate the timeblock and then the interuption index.

#remove duplicate rows

ix <- c(TRUE, rowSums(tail(sepsis\_df, -1) == head(sepsis\_df, -1)) != ncol(sepsis\_df))

sepsis\_df<-sepsis\_df[ix,]

#transpose case\_id column

sepsis\_df <- ddply(sepsis\_df, .(ID\_day), transform, idx = paste("TB", 1:length(case\_id), sep = "")) %>% spread(idx, case\_id)

#calculate timeblocks

sepsis\_df<-sepsis\_df %>% mutate(timeblock = rowSums(!is.na(select(.,starts\_with("TB"))))) %>% select(-starts\_with("TB")) # remove reduntanct TB variables

#calculate index

sepsis\_df$interupt\_index<-sepsis\_df$timeblock/ sepsis\_df$caseload

# sample size of index for each resource

sepsis\_df<-sepsis\_df %>% add\_count(resource, interupt\_index)

head(sepsis\_df)

## # A tibble: 6 x 6

## resource ID\_day caseload timeblock interupt\_index n

##

## 1 A 1 1 1 1 282

## 2 A 2 1 1 1 282

## 3 A 3 1 1 1 282

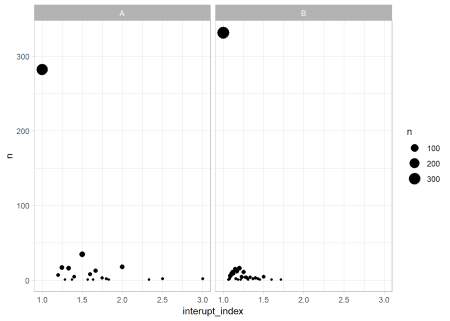
## 4 A 4 1 1 1 282

## 5 A 5 2 5 2.5 2

## 6 A 6 1 1 1 282

**Visualizing the index**

sepsis\_df %>% ggplot(aes (interupt\_index, n, size=n)) + geom\_point() + facet\_grid(~resource)

Let’s plot the interruption index for Resource A and Resource B. Resource A and Resource B experienced an interruption index of 1 on most days. In other words, they completed all the activities for a specific case before moving to the next case. Resource B has more days with an index of 1 than Resource A. There is a greater variation of the interruption index for Resource A, with a maximum ratio of 3.

**Timeseries Heatmap**

bupaR has many fantastic built in functions to create various visualizations to address different questions on workflow. Unfortunately, bupaR does not have a function to create a time series heat map. I find time series heat map useful to identify peak and lull periods for each activity. Discovering peak periods allow managers to allocate more resources to prevent accumulation of bottlenecks. Understanding lull periods can free up excess capacity.

**Data wrangling**

We will use a different dataset, patients, to demonstrate how to create a time series heat map from an event log.

The patients dataset is an event log of a series of activates conducted when patients are admitted till they are discharged. However, the current factor levelling for the activity variable, handling is not reflective of an expected workflow when patients are admitted. For instance, “Blood test” is the first activity and “Registration” is the fifth activity.

patients\_df<-data.frame(patients) #convert the `eventlog` object to a `dataframe` object

levels(patients\_df$handling)

## [1] "Blood test" "Check-out" "Discuss Results"

## [4] "MRI SCAN" "Registration" "Triage and Assessment"

## [7] "X-Ray"

We’ll need to relevel the activities in handling to a sequence likely seen in a hospital admission.

patients\_df<-patients\_df %>% mutate(handling= fct\_relevel(handling, "Registration", "Triage and Assessment", "Blood test", "X-Ray", "MRI SCAN", "Discuss Results", "Check-out"))

levels(patients\_df$handling)

## [1] "Registration" "Triage and Assessment" "Blood test"

## [4] "X-Ray" "MRI SCAN" "Discuss Results"

## [7] "Check-out"

**Ploting the heatmap**

Traditionally, I will use the plasma colour palette in scale\_fill\_viridis\_c for heatmaps. However, since I encountered this [TidyTuesday tweet](https://twitter.com/noccaea/status/1095735292206739456/photo/1?ref_src=twsrc%5Etfw%7Ctwcamp%5Etweetembed%7Ctwterm%5E1095735292206739456&ref_url=https%3A%2F%2Fnsgrantham.shinyapps.io%2Ftidytuesdayrocks%2F), I have adopted the Zissou1 colour palette from the wesanderson package for my heatmaps.

patients\_df %>% dplyr::mutate(

time= format(time, format = "%H:%M:%S") %>% as.POSIXct(format = "%H:%M:%S"), #standardized the date for ploting

hour= lubridate::floor\_date(time, "hour")) %>% # round down time to nearest hour

count(handling, hour) %>% # total instances of each activity at each hour

add\_count(handling, wt=n) %>% # total instances of each activity

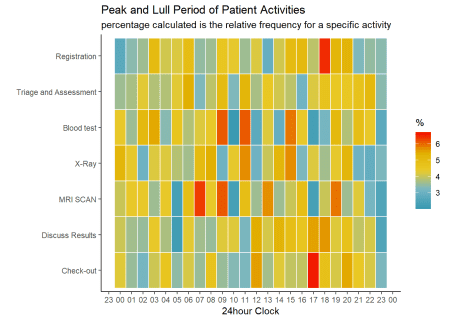
mutate(percent= ((n/nn)\*100)) %>% #relative freq for each activity

ggplot(aes(hour, handling, fill=percent)) + geom\_tile(size=.5, color="white") + scale\_fill\_gradientn(colours = wesanderson::wes\_palette("Zissou1", 20 ,type = "continuous"))+

theme\_classic() +

labs(x="24hour Clock", y="", title= "Peak and Lull Period of Patient Activities", subtitle= "percentage calculated is the relative frequency for a specific activity", fill="%") + scale\_y\_discrete(limits = rev(levels(patients\_df$handling)))+ # reverse display of y-axis varaibles

scale\_x\_datetime(date\_breaks = ("1 hour"), date\_labels = "%H") #display only 24H clock values

The heatmap reveals that the most common time to check out is 1700hours and the most common time to register is 1800hours. This makes logical sense, as existing patients need to be discharged first before the hospital can admit new patients. There is consecutive alternation between warm and cool colours for the blood test activity from 0900 hours to 1600hours. I hypothesize that the hourly fluctuation is due to insufficient machinery/manpower during office hours. During the lull periods, manpower and machinery are operating at maximum capacity to process blood collected from the previous peak period. There is inadequate machinery/manpower to accommodate more blood test thus less blood test are conducted.

**To Sum Up**

In this last post on process mining, we look at analysis and visualizations not included in the bupaR package but are still useful in understanding the workflow of business operations. We calculated the interruption index to examine the extend of disruption when a resource attends to a specific case. We also plotted a heat map to visualize busy and quiet periods of the various activities in an event log.