Dataset generation: commented code

Parameters

```
### Sales parameters ---------
sales_A <- 3000
startdate_A <- as.Date("2017/1/1")
enddate_A <- as.Date("2019/12/31")

sales_B <- 2000
startdate_B <- as.Date("2018/1/1")
enddate_B <- as.Date("2019/12/31")

### Device parts parameters ---------

parts_A <- c("K01", "S01")
parts_B <- c("K02", "S01")
```

Given these parameters, we can build the dataset that will serve as a basis for the remainder of this notebook. Most of what follows will consists in looping over this basic structure to enrich it with sales and failures data.

```
(main_df <- tibble(
  terminal_model = c("A", "B"),
  total_sales = c(sales_A, sales_B),
  from = c(startdate_A, startdate_B),
  to = c(enddate_A, enddate_B)
))
```

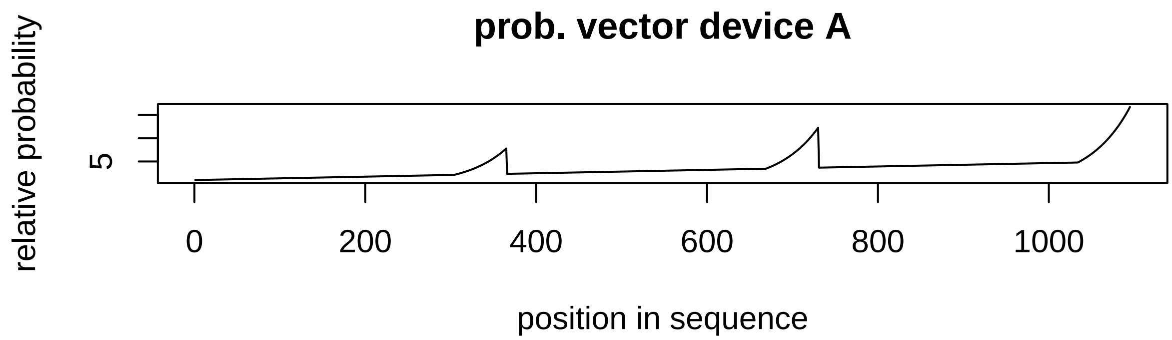
```
## # A tibble: 2 x 4
##   terminal_model total_sales from       to
##   <chr>                <dbl> <date>     <date>
## 1 A                     3000 2017-01-01 2019-12-31
## 2 B                     2000 2018-01-01 2019-12-31
```

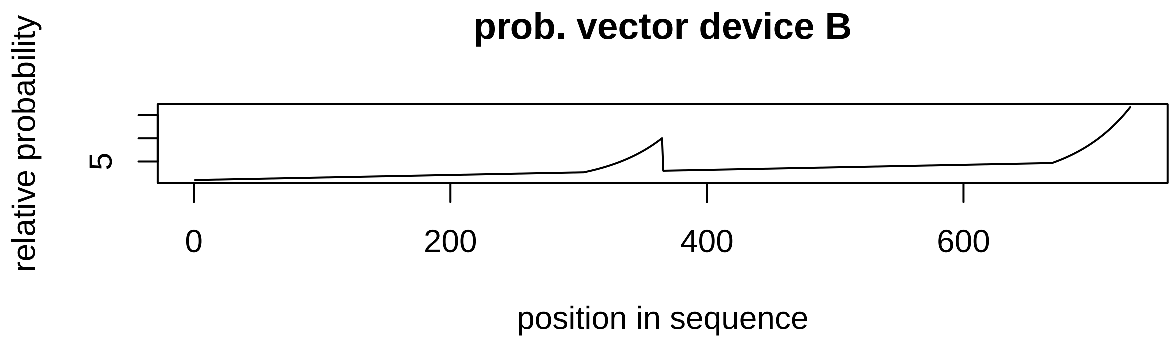
Generate sales

For each device model $\mathcal{M} \in \{A, B\}$, a sequence of consecutive dates is generated from $startdate_M$ to $enddate_M$. Then, a sample of size $sales_M$ is randomly drawn with replacement from that sequence. Some trend and seasonality is deliberately created by the fact that at each stochastic iteration, the relative probability for a given date in the sequence to be picked depends on its position in the calendar year.

```
sales_period_A <- as.numeric(enddate_A - startdate_A) + 1
sales_period_B <- as.numeric(enddate_B - startdate_B) + 1

# for simplicity, let's assume A and B have the same total sales growth over their respective sales per
probability_vector_A <-
  rep(c(rep(1,304),cumprod(rep(1.02, 61))), sales_period_A/365) * seq(from = 1, to = 5, length.out = sal

probability_vector_B <-
  rep(c(rep(1,304),cumprod(rep(1.02, 61))), sales_period_B/365) * seq(from = 1, to = 5, length.out = sal

par(mfrow=c(2,1))
plot.ts(probability_vector_A, ylab = "relative probability", xlab = "position in sequence", main = "prob
```

```
plot.ts(probability_vector_B, ylab = "relative probability", xlab = "position in sequence", main = "prol
```





Now I create and apply the function that generate the sales based on that logic :

```
generate_sales <- function(totalsales, startdate, enddate){

  sales_period = as.numeric(enddate - startdate) + 1
  probability_vector = rep(c(rep(1,304),cumprod(rep(1.02, 61))), sales_period/365) * seq(from = 1, to =
  (sales <- sample(seq(from = startdate, to = enddate, by = "day"), size = totalsales, prob = probabili
}

(main_df <- main_df %>%
  mutate(date_sold = pmap(list(total_sales, from, to), generate_sales)) %>%
  select(-c(from, to)))
```

```
## # A tibble: 2 x 3
##   terminal_model total_sales date_sold
##   <chr>                <dbl> <list>
## 1 A                     3000 <date [3,000]>
## 2 B                     2000 <date [2,000]>
```

Generate failures

In generating failures, I make the assumption that no part can live forever. Hence, each part j of type $k \in \{"K01", "K02", "S01"\}$ has a theoretical time-to-failure $x_j^{theoretical}$ that is randomly drawn from a part-specific life length distribution $\mathcal{D}_k$. Yet, this failure might not occur for two reasons:

- A failure occured due to another part of the same device
- The failure date has not been reached yet

In this setting, the *observed time-to-failure* can be generated for each device i that contains a set of parts J_i as:

$$x_i^{obs} = \begin{cases} x_i = \min_{j \in J_i}(x_j^{theoretical}) & \text{if } date\_sold_i + x_i \leq today \\ \emptyset & \text{otherwise} \end{cases}$$

where $x_j^{theoretical} \sim \mathcal{D}_k$.

Then, the *reparation date* field is filled as follows:

$$reparation\_date_i = \begin{cases} date\_sold_i + x_i^{obs} & \text{if } \exists x_i^{obs} \\ \emptyset & \text{otherwise} \end{cases}$$

I obtain the part-specific distributions $\mathcal{D}_k$ for $k \in \{"K01", "K02", "S01"\}$ by combining together skewed normal, truncated normal and uniform distributions. The goals here is to make these distributions heterogeneous and "unpure" enough so that methods based on over-simplifying assumptions are not misleadingly validated during the analysis phase.

This process also brings in the realistic challenge of right-censoring : we are less likely to observe the time-to-failure of longer-living devices. If not tackeled carefully, this could easily be a source of bias in the analysis.

```
# Build function able to loop on part_codes and draw individual theoretical times-to-failure

generate_parts_failures <- function(part_code){
  if (part_code == "K01"){
    df = data.frame(
      failure_description = "Key not responding",
      timetofailure_theoretical = sample(
          c(rsn(n = 1, omega = 60, alpha = 10, tau = 0),  #Skewed normal
            rsn(n = 1, omega = -100, alpha = 5, tau = 10) + 400, #Skewed normal
            runif(1, min = 1, max = 5000)), # Uniform
            size = 1,
            prob = c(0.15,0.05, 0.8)
          )
    )
  }
  if (part_code == "K02"){
    df = data.frame(
      failure_description = "Key not responding",
      timetofailure_theoretical = sample(
          c(rsn(n = 1, omega = -100, alpha = 5, tau = 10) + 400, #Skewed normal
            runif(1, min = 1, max = 7000)), # Uniform
            size = 1,
            prob = c(0.05, 0.95)
          )
    )
  }
  if (part_code == "S01"){
    # might suffer from two types of failures
    if (sample(c("type1", "type2"), 1) == "type1"){
      df = data.frame(
            failure_description = "Screen flickering",
            timetofailure_theoretical = sample(
                c(rsn(n = 1, omega = -150, alpha = 10000, tau = .2) + 800, #Skewed normal
```

```
                  rtruncnorm(n = 1, mean = 800, sd = 600, a = 800)), #Truncated normal (only variates a
                  size = 1,
                  prob = c(0.25, 0.75)
                )
            )
    }
    else{
      df = data.frame(
            failure_description = "Screen shutdown",
            timetofailure_theoretical = runif(1, min = 1, max = 2500) # Uniform
      )
    }
  }
  return(df %>% filter(timetofailure_theoretical >=0))
}



# Build function that, applied to main_df unnested:
# 1. splits a device in its parts;
# 2. applies the function generate_parts_failures() for each part;
# 3. computes the observed time-to-failure
# 4. computes the reparation date (when)

parts_df <- data.frame(
  terminal_model = c(rep("A", length(parts_A)), rep("B", length(parts_B))),
  part_code = c(parts_A, parts_B)
)

generate_reparation_dates <- function(model, date_sold){

  temporary_df<- filter(parts_df, terminal_model == model) %>%
  mutate(date_sold = date_sold)

  observed_failure <- temporary_df %>%
  mutate(timetofailure_theoretical = map(part_code, generate_parts_failures)) %>%
  unnest() %>%
  top_n(-1, timetofailure_theoretical) %>%
  mutate(failure_date = as.Date(
    ifelse(date_sold + ddays(timetofailure_theoretical) < Sys.time(),
           as.Date(date_sold + ddays(timetofailure_theoretical)),
           NA),
    origin="1970-01-01")) %>%
  mutate(failure_description = ifelse(is.na(failure_date), NA, failure_description)) %>%
  select(failure_description, failure_date)

  return(observed_failure)
}

## Apply the function:

source_df <- main_df %>%
  unnest() %>%
```

```r
  mutate(reparation_date = map2(terminal_model, date_sold, generate_reparation_dates)) %>%
  unnest() %>%
  select(-total_sales)

source_df %>%
  head(100)
```

```
## # A tibble: 100 x 4
##    terminal_model date_sold  failure_description failure_date
##    <chr>          <date>     <chr>               <date>
##  1 A              2019-07-13 <NA>                NA
##  2 A              2019-08-26 Key not responding  2019-10-02
##  3 A              2018-05-19 Screen shutdown     2018-08-09
##  4 A              2019-12-25 Key not responding  2020-01-03
##  5 A              2018-09-19 <NA>                NA
##  6 A              2017-06-04 <NA>                NA
##  7 A              2018-12-02 <NA>                NA
##  8 A              2018-12-16 <NA>                NA
##  9 A              2019-08-28 <NA>                NA
## 10 A              2018-09-22 <NA>                NA
## # ... with 90 more rows
```

Here is a summary of the observed failures in the obtained dataset.

```r
source_df %>%
  group_by(failure_description) %>%
  summarize(n = n())
```

```
## # A tibble: 4 x 2
##   failure_description     n
##   <chr>               <int>
## 1 Key not responding    734
## 2 Screen flickering      70
## 3 Screen shutdown       300
## 4 <NA>                 3896
```