Projet-RM04

Baptiste Toussaint

2024-05-29

Contents

RM04 - Projet n°2

Baptiste Toussaint

```
# thème personnel pour mes graphiques
custom_theme <- theme(
        panel.background = element_rect(fill = "#D9E8F1",
                                        colour = "#6D9EC1",
                                        size = 1, linetype = "solid")
)
```

```
# récupération des données depuis le document excel
df <- readxl::read_xlsx("data/Historique panne.xlsx")
# le symbole '%>%' est nommé 'pipe' et signifie : passer en argument
# ici je passe le dataframe df en argument à la fonction `head()` pour
# afficher les premières colonnes.
df %>% head()
```

```
## # A tibble: 6 x 5
##   Date                `temps d'arrêt` Nature du travail/déf~1 Désignation Repère
##   <dttm>              <chr>           <chr>                   <chr>       <chr>
## 1 2003-11-18 00:00:00 20 min          "Mauvaise trajectoire ~ Nez robot   E
```

```
## 2 2003-11-22 00:00:00 45 min          "Départ cycle défailla~ Nez robot   E
## 3 2003-01-13 00:00:00 25 min          "Avance du bras saccad~ Nez robot   E
## 4 2004-01-18 00:00:00 94 min          "Mauvaise trajectoire ~ Carte DH    I
## 5 2004-01-18 00:00:00 10 min          "Avant par saccade (ca~ Carte (s) ~ J
## 6 2004-01-27 00:00:00 10 min          "Pas de départ cycle \~ <NA>        <NA>
## # i abbreviated name: 1: `Nature du travail/défaut`
```

Projet : Maintenance d'un robot de peinture

Dans ce projet on étudie la maintenance d'un robot permettant de positionner un pistolet de peinture.

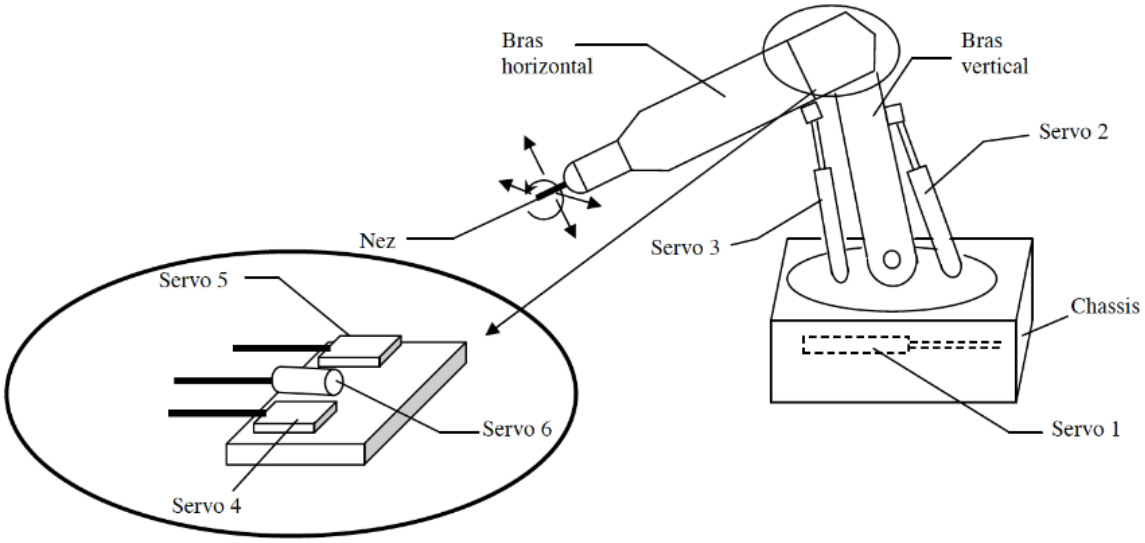


FIGURE 1 – Plan schématique d'un robot de peinture

Figure 1: Plan schématique d'un robot de peinture

Présentation du robot

Le robot à pour mission de positionner dans l'espace un pistolet à peinture. Le pistolet est actionné par un autre système qui n'est pas étudié ici.

Le robot doit donc positionner avec précision et au bon moment le pistolet pour permettre ensuite au pistolet d'appliquer la peinture.

Le robot est composé des éléments suivants :

- (A) l'électrovanne du pistolet ;
- (B) Vérins ;
- (D) Poignets de programmation ;
- (E) Nez robot ;
- (F) Fin de course support bras ;
- (I) Carte DH ;
- (J) Carte Servo.

Objectif de l'analyse

L'objectif est de proposer pour l'entreprise un nouveau plan de maintenance visant à minimiser les coûts associés à la maintenance de ce robot.

Nous disposons pour cela de deux sources de données :

- un historique de durée inter-panne en heures de fonctionnement pour chaque composant du robot ;
- Un retour d'expérience détaillant certaines pannes pour les différents composants du robot. Ce retour d'expérience prend la forme d'un tableau excel dans lequel on retrouve, pour chaque panne renseignée.

Dans un premier temps, on cherchera à classer les différents types de pannes selon la méthode ABC pour identifier quelles pannes sont prépondérantes dans les coûts totaux liés à la maintenance.

On essaiera ensuite de déterminer les paramètres des lois de durée de vie des composants en supposant qu'il s'agit de loi de Weibull. On estimera aussi le paramètre de la loi exponentielle associée au changement correctif visant à remplacer les éléments.

Enfin nous pourrons optimiser les politiques de maintenance pour les pannes de la classe A et B identifiées en début d'analyse.

Classification des pannes par la méthode ABC.

Le principe de Pareto stipule qu'environ 80 % des effets sont le produit de 20 % des causes (Principe de Pareto, Wikipédia). Ce principe général à été observé par l'italien Vilfredo Pareto et on peut l'appliquer à différents phénomène observables.

Ce principe ne doit pas être appliqué à la lettre mais est une première approche souvent pertinente pour identifier les éléments prépondérant dans un phénomène et prioriser les actions.

La méthode ABC s'appuie sur ce principe de Pareto, en répartissant les éléments observés en trois classes : A, B et C, selon leurs effets pour le phénomène observé.

Appliqué au problématique de maintenance, on peut donc imaginer que 80% des coûts de maintenances sont issus de 20% des pannes, que l'on classera alors dans la catégorie A. Les 15% des coûts suivants sont le fruit de 30% des défaillances : la classe B. Enfin, la majorité des défaillance, soit les 50% restants, ne jouent que sur 5% des coût totaux de maintenance. Ils constituent alors la classe C.

Pour chaque panne identifiées par le retour d'expérience, on va calculer le coût associé.

On pourra alors calculer un coût total de maintenance et observer quelles pannes contribuent le plus à ce coût total. Ces pannes seront alors prioritaires dans nos améliorations de politiques de maintenance.

Calcul du coût d'une panne.

L'analyse des coût a permis d'identifier deux coûts associés à une panne :

- le coût de remplacement (coût de changement) d'une pièce estimé à 30£ ;
- le coût d'inactivité estimé à $20£/min$.

On constate immédiatement que le coût immobilisation est prépondérant dans. le coût total de la politique de maintenance : deux minutes d'arrêts sont plus coûteuses qu'un changement de pièce.

Le coût de remplacement étant fixe et identique pour chaque composant du robot, on peut se concentrer sur le coût d'immobilisation dans notre classification.

On va donc construire un diagramme ABC basée sur le temps d'arrêt des pannes associé à chaque élément pour identifier les éléments dont les pannes immobilisent le plus le robot.

Construction du graphique ABC

Le retour d'expérience est composé des données suivantes :

- `Date` : la date d'observation de la défaillance ;
- `temps d'arrêt` : temps d'arrêt observé pour cette défaillance ;
- `Nature du travail/défaut` décrit la défaillance et les opérations associées ;
- `Désignation` : le nom du composant défaillant ;
- `Repère` : le repère du composant défaillant.

Dans notre cas, ce sont les attributs : `temps d'arrêt` et `Repère` qui nous intéressent le plus.

```
# Récupère dans `df_abc` que les colonnes utiles pour cette partie du projet
df_abc <- df %>% select("temps d'arrêt","Repère")
df_abc
```

```
## # A tibble: 40 x 2
##    `temps d'arrêt` Repère
##    <chr>           <chr>
##  1 20 min          E
##  2 45 min          E
##  3 25 min          E
##  4 94 min          I
##  5 10 min          J
##  6 10 min          <NA>
##  7 30 min          H
##  8 30 min          A
##  9 10 min          G
## 10 15 min          B
## # i 30 more rows
```

Le temps d'arrêt étant stocké en tant que numérique, il faut les convertir en valeur numérique.

```
# convertit la colonne `temps d'arrêt` (char) en une colonne `temps_arret_min`
# en numérique (double)
df_abc <- df_abc %>% mutate(
    temps_arret_min = as.numeric(
        str_sub(`temps d'arrêt`,start = 1, end = -5)
    )
)
df_abc
```

```
## # A tibble: 40 x 3
##    `temps d'arrêt` Repère temps_arret_min
##    <chr>           <chr>            <dbl>
##  1 20 min          E                   20
##  2 45 min          E                   45
##  3 25 min          E                   25
##  4 94 min          I                   94
##  5 10 min          J                   10
##  6 10 min          <NA>                10
##  7 30 min          H                   30
##  8 30 min          A                   30
##  9 10 min          G                   10
## 10 15 min          B                   15
## # i 30 more rows
```

Deux mesures ne possèdent pas de repère : cela signifie que la défaillance, selon les opérateurs chargés de

répertorier les défaillances, ne concernent pas une partie spécifique du robot.

Dans un premier temps je propose d'ignorer ces défaillances que je qualifierai de "non attribuées". On discutera par la suite de la manière de les intégrer à l'analyse.

```
# Retire les lignes contenant des NA
df_abc_noNA <- df_abc[!is.na(df_abc$Repère),]
df_abc_noNA
```

```
## # A tibble: 38 x 3
##    `temps d'arrêt` Repère temps_arret_min
##    <chr>           <chr>            <dbl>
##  1 20 min          E                   20
##  2 45 min          E                   45
##  3 25 min          E                   25
##  4 94 min          I                   94
##  5 10 min          J                   10
##  6 30 min          H                   30
##  7 30 min          A                   30
##  8 10 min          G                   10
##  9 15 min          B                   15
## 10 15 min          A                   15
## # i 28 more rows
```

```
temps_arret_total <- sum(df_abc_noNA$temps_arret_min)
temps_arret_total
```

```
## [1] 1959
```

```
# Effectue le calcul du temps total d'arrêt par repère puis
# classe par ordre décroissant
somme_temps_arret_par_repere <- aggregate(temps_arret_min~Repère,
                                          data = df_abc_noNA, sum) %>%
    arrange(desc(temps_arret_min))

somme_temps_arret_par_repere
```

```
##   Repère temps_arret_min
## 1      E             925
## 2      D             530
## 3      A             185
## 4      F             160
## 5      I              94
## 6      H              30
## 7      B              15
## 8      G              10
## 9      J              10
```

```
# Ajoute un point fictif en (0,0) pour le graphique suivant
x_set <- c("", somme_temps_arret_par_repere$Repère); x_set
```

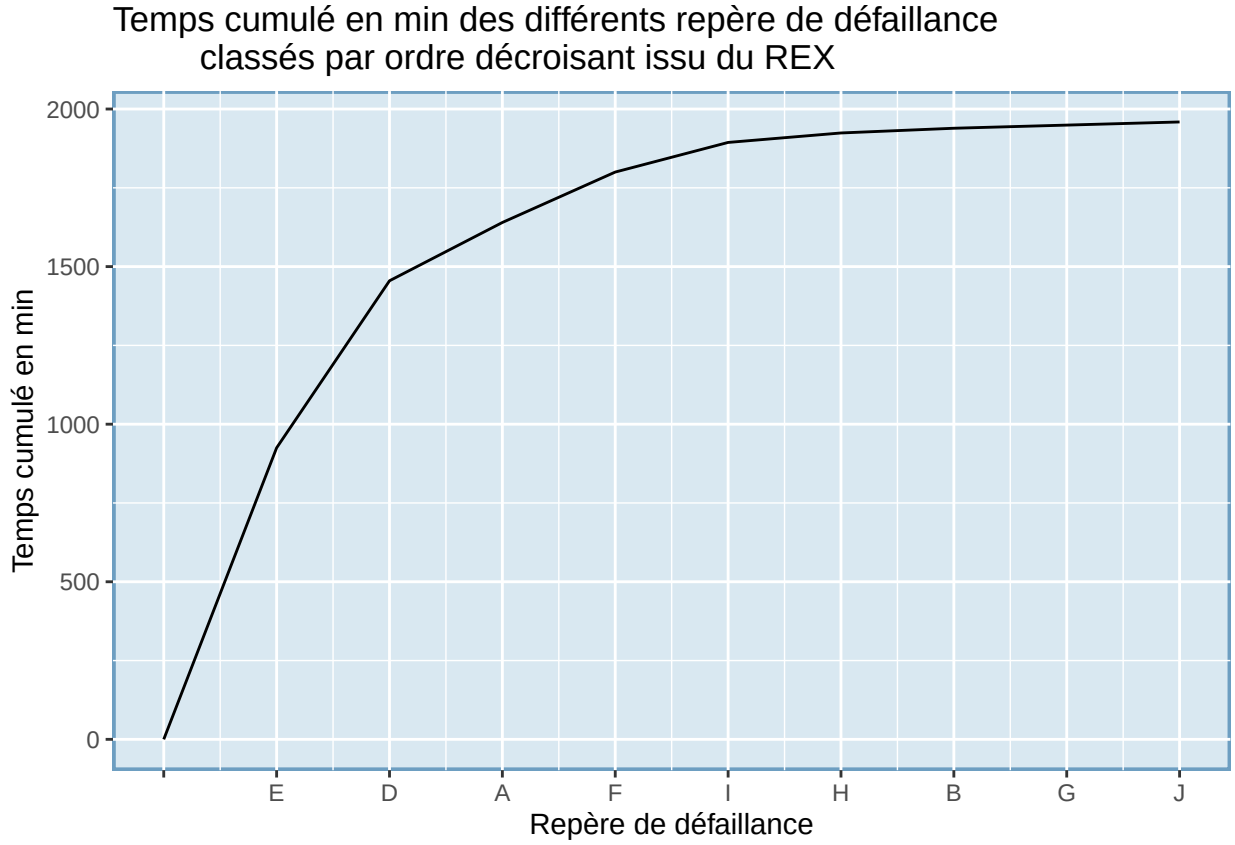
```
##  [1] ""  "E" "D" "A" "F" "I" "H" "B" "G" "J"
```

```
y_set <- c(0, cumsum(somme_temps_arret_par_repere$temps_arret_min)); y_set
```

```
##  [1]    0  925 1455 1640 1800 1894 1924 1939 1949 1959
```

À noter que les défaillances avec pour repère H et G sont des défaillances ne se rapportant pas directement au robot : - H : Disquette - G : Manque de pression

```r
# ggplot permet de créer de plus jolis graphiques et est facilement
# utilisable pour un résultat proche de ce que l'on souhaite avoir
ggplot(mapping = aes(x = 0:9, y = y_set))+
    geom_line() +
    scale_x_continuous(breaks = 0:9,
                       labels = x_set) +
    labs(x = "Repère de défaillance",
         y = "Temps cumulé en min",
         title = "Temps cumulé en min des différents repère de défaillance
         classés par ordre décroisant issu du REX") +
    custom_theme
```

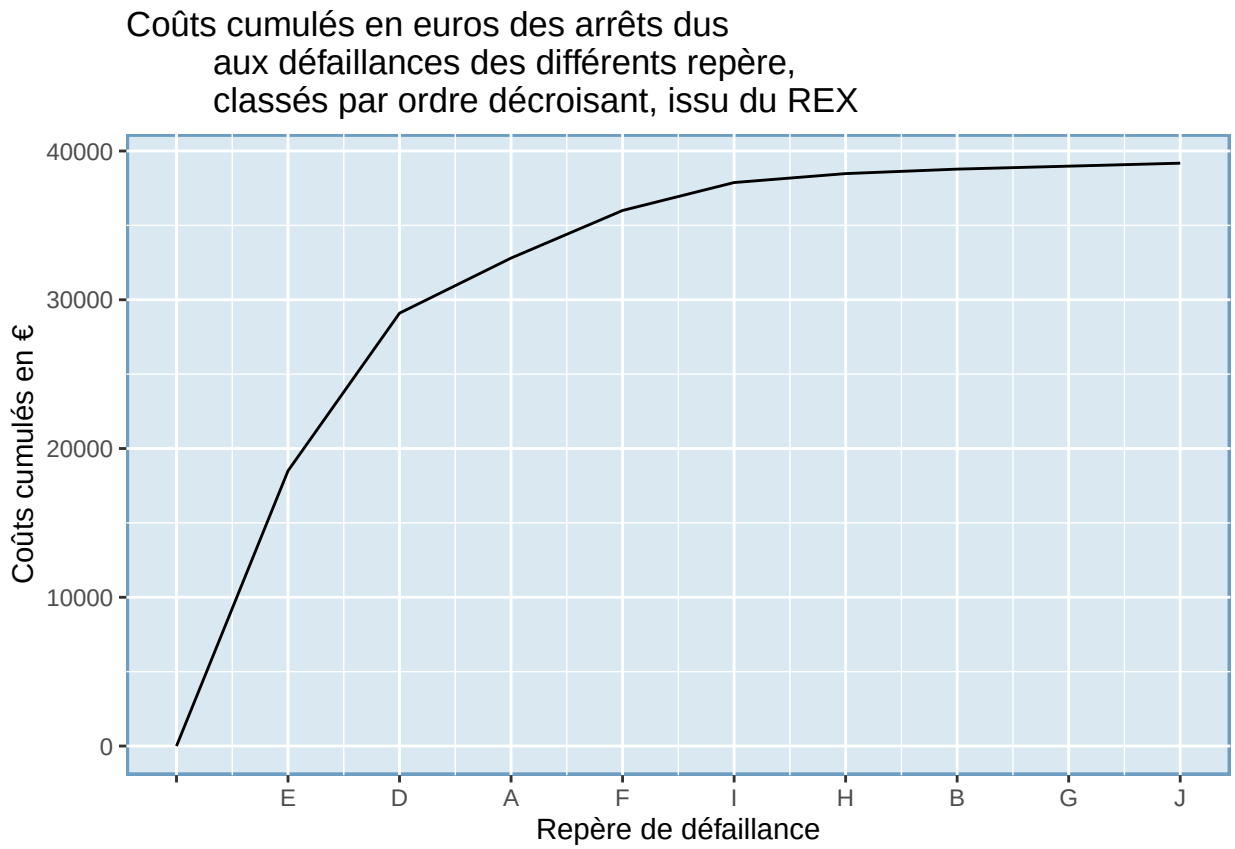


Avec ce graphique, on peut facilement voir par exemple que le temps d'arrêt cumulé des défaillances sur les repères E et D avoisine les 1500 minutes.

Comme on connait le coût d'arrêt par minute qui est de $ci = 20£/min$, on peut facilement produire le même graphique mais cette fois en € :

```r
# cout d'arret par minute en euro
ci = 20
ggplot(mapping = aes(x = 0:9, y = y_set * ci))+
    geom_line() +
    scale_x_continuous(breaks = 0:9,
                       labels = x_set) +
    labs(x = "Repère de défaillance",
         y = "Coûts cumulés en €",
         title = "Coûts cumulés en euros des arrêts dus
```

```
        aux défaillances des différents repère,
        classés par ordre décroisant, issu du REX") +
    custom_theme
```



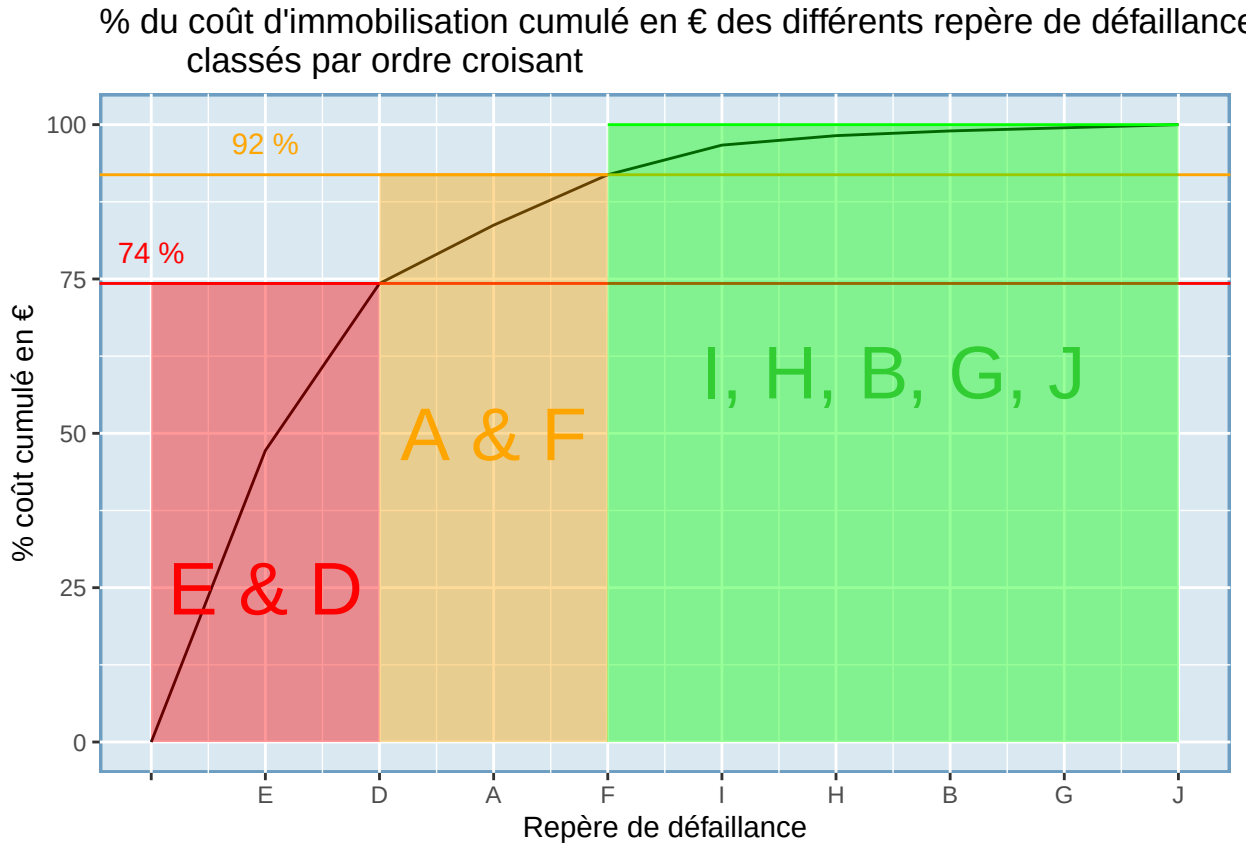
Pour simplifier la lecture, et distinguer nos classes A, B et C, je propose de retourner sur l'analyse des temps d'arrêt (équivalent au coût par un simple produit) et de raisonner en % du temps total d'arrêt mesuré.

On va séparer les repères dans les classes suivantes par ordre de contribution au temps d'arrêt total :

- Classe A : Repères E et D soit "Nez robot" et "Poignées de programmation"
- Classe B : "Electrovanne pistolet" et "Fin de course du support bras"
- Classe C : le reste.

```
ggplot(mapping = aes(x = 0:9, y = 100*y_set/temps_arret_total))+
    geom_line() +
    ylim(c(0,100)) +
    scale_x_continuous(breaks = 0:9,
                       labels = x_set) +
    labs(x = "Repère de défaillance",
         y = "% coût cumulé en €",
         title = "% du coût d'immobilisation cumulé en € des différents repère de défaillance
         classés par ordre croisant") +
    geom_hline(yintercept = 100*y_set[3]/temps_arret_total,
               color = 'red') +
    annotate(geom = "text",
             x = 0,
             y = 100*y_set[3]/temps_arret_total+5,
```

```
          label = paste(
              round(100*y_set[3]/temps_arret_total),"%"),
          color = 'red') +
  geom_area(aes(x = 0:2, y = 100*y_set[3]/temps_arret_total),
           fill = 'red',
           alpha = 0.4,
           color = 'red') +
  geom_hline(yintercept = 100*y_set[5]/temps_arret_total,
            color = 'orange') +
  annotate(geom = "text",
          x = 1,
          y = 100*y_set[5]/temps_arret_total+5,
          label = paste(
              round(100*y_set[5]/temps_arret_total),"%"),
          color = 'orange') +
  geom_area(aes(x = 2:4, y = 100*y_set[5]/temps_arret_total),
           fill = 'orange',
           alpha = 0.4,
           color = 'orange') +
  geom_area(aes(x = 4:9, y = 100),
           fill = 'green',
           alpha = 0.4,
           color = 'green') +
  annotate(geom = "text", x = 1, y = 25, label = "E & D", color = 'red',
          size = 10) +
  annotate(geom = "text", x = 3, y = 50, label = "A & F", color = 'orange',
          size = 10) +
  annotate(geom = "text", x = 6.5, y = 60, label = "I, H, B, G, J", color = 'limegreen',
          size = 10) +
  custom_theme
```



Sur le graphique on peut alors observer que les défaillances liées aux repères E et D (la classe A), soit le "Nez robot" et la "Poignées de programmation" représentent à elles seules 74% du coût d'immobilisation (puisque ces défaillances représentent 74% du temps d'immobilisation total).

La classe B représente 18% du coût total d'immobilisation.

Les autres défaillances (classe C) représentent donc 8% du temps total d'immobilisation.

La priorité devra donc être mise sur la politique de maintenance des composants des classes A et B (les repères E, D, A, F).

Estimation des durées de vie de chaque élément.

On cherche à présent à définir la durée de vie de chaque élément du robot. Cette information sera utile pour ensuite pour créer un modèle permettant l'optimisation de la politique de maintenance.

Modèle mathématique

On suppose que la durée de bon fonctionnement avant la panne de chaque composant ω, avec $\omega \in [A, B, D, E, F, G, H, I, J]$ est une variable aléatoire notée T_ω à valeur dans $\mathbb{R}_+^*$.

On suppose que les T_ω suivent une loi de Weibull de paramètre d'échelle α et de forme β. On note alors $T_\omega \sim \mathbf{W}(\alpha_\omega, \beta_\omega)$.

On cherche alors a estimer les paramètres $(\alpha_\omega, \beta_\omega)$, pour avoir une idée plus précise du comportement des composants.

Méthode d'estimation par maximum de vraissemblance

Wikipédia définit la vraisemblance comme : "fonction des paramètres d'un modèle statistique calculée à partir de données observées".

Sous l'hypothèse d'un modèle connu, la fonction de vraisemblance (notée L permet de calculer la probabilité d'obtenir une un tirage précis d'observation. C'est en somme la probabilité d'obtenir ce résultat précis de tirage en supposant un modèle précis.

On note la vraisemblance $L(\vec{\theta}, \vec{x})$, avec $\vec{\theta}$ les paramètre de notre modèle et $\vec{\curvearrowleft}$ le vecteur des observations issues de n variables aléatoires X_i

On a lors : $L(\vec{\theta}, \vec{x}) = \prod_{i=1}^{n} f_i(\vec{\theta}, x_i)$.

L'estimation par maximum de vraisemblance, revient à chercher, pour un jeu d'observation donné $\vec{x}$, les paramètres $\vec{\theta^*}$ qui maximisent la fonction de vraisemblance.

Cela revient à chercher : avec les observations obtenues, quels sont les valeurs prises par mon modèle les plus "vraisemblables".

La maximisation passe par l'annulation dérivée (ou les dérivées partielles) suivant les paramètres.

Application au cas de la loi de Weibull

On se place dans un cas de maintenance parfaite (chaque maintenance remet à neuf le système en remplaçant les composants) suivant une loi de Weibull.

Pour rappel la loi de Weibull possède la densité suivante :

$$f_i(\alpha, \beta, x_i) = \left(\frac{\beta}{\alpha}\right)\left(\frac{x_i}{\alpha}\right)^{\beta-1} e^{-\left(\frac{x_i}{\alpha}\right)^{\beta}}$$

Le calcul de la vraisemblance est donc relativement complexe. Si l'on souhaite une solution analytique au problème, il est préférable de passer la log-vraissemblance.

En effet, la fonction logarithme étant croissante, maximiser la log-vraissemblance revient à maximiser la vraisemblance.

Après un développement, on obtient alors le système :

$$\begin{cases} \alpha^* = \left(\frac{1}{n}\sum_{i=1}^{n} x_i^{\beta^*}\right)^{\frac{1}{\beta^*}} \\ \frac{1}{\beta^*} \frac{\sum_{i=1}^{n} ln(x_i)}{n} - \frac{\sum_{i=1}^{n} x_i^{\beta^*} ln(x_i)}{\sum_{i=1}^{n} x_i^{\beta^*}} = 0 \end{cases}$$

que l'on doit résoudre pour obtenir β^* et α^* qui sont alors les estimateurs du maximum de vraisemblance que l'on notera $\hat{\alpha}_{MV}$ et $\hat{\beta}_{MV}$.

Plusieurs solutions existent pour obtenir ses estimateurs sans passer par la résolution de ce système.

Construction des données

Pour estimer les paramètres des lois, on utilise les données d'analyse de durées de défaillance entre deux pannes (en heures).

```
durees_inter_panne <- list(
    A = c(100,150,30,45,170,195,200,250,340,60),
    B = c(250,400,430,670,1000,1500,1200,1050,480),
    D = c(55,40,70,120,150,270,200,190),
    E = c(110,208,170,190,155,230,340,150,160,195,280,250),
    F = c(45,60,72,68,95,12,18,40,49),
```

```
    I = c(111,70,50,60,80,904,100,75,67,71,110),
    J = c(130,150,117,200,180,155,140,130,81,75)
)
```

Ici la liste `durees_inter_panne` contient les durées inter-panne en heure. Cependant, nous avions stocké les temps d'arrêt

Estimations

Utilisation de `nlm` La fonction `nlm` permet de minimiser une fonction dans R suivant un jeu de paramètre.

On peut alors utiliser cette fonction pour minimiser la fonction -log-vraisemblance, ce qui revient à maximiser la vraisemblance.

La log-vraisemblance dans le cas d'un loi de Weibull s'écrit :

$$lnL(\alpha, \beta, \vec{x}) = nln(\beta) - n\beta ln(\alpha) + (\beta - 1)\sum_{i=1}^{n} ln(x_i) - \frac{1}{\alpha^\beta}\sum_{i=1}^{n}(x_i)^\beta$$

Donc on va minimiser la fonction :

$$-lnL(\alpha, \beta, \vec{x}) = -nln(\beta) + n\beta ln(\alpha) - (\beta - 1)\sum_{i=1}^{n} ln(x_i) + \frac{1}{\alpha^\beta}\sum_{i=1}^{n}(x_i)^\beta$$

```
neg_log_likelihood <- function(param, xi) {

    # la fonction nlm ne prenant qu'un seul paramètre : les paramètres d'optimisation,
    # il faut donc utiliser un tableau 'param' dans lequel :
    # param[1] = alpha
    # param[2] = beta

    n <- length(xi)

    -n * log(param[2]) +
        n * param[2] * log(param[1]) -
        (param[2]-1) * sum(log(xi)) +
        (1/(param[1]^(param[2]))) * sum((xi)^param[2]) %>%
        return()

}
```

```
estim_max_likelihood <- c()
for (xi in durees_inter_panne) {
    ans <- nlm(f = neg_log_likelihood, p = c(1,1), xi)
    # print(ans$estimate)
    estim_max_likelihood <- c(estim_max_likelihood, ans$estimate)
}

estim_max_likelihood <- tibble::as_data_frame(
    matrix(estim_max_likelihood, ncol = 2, byrow = TRUE)
)
estim_max_likelihood$Repère <- c("A", "B", "D", "E", "F", "I", "J")
colnames(estim_max_likelihood) <- c("Alpha", "Beta", "Repère")
estim_max_likelihood
```

```
## # A tibble: 7 x 3
##   Alpha  Beta Repère
##   <dbl> <dbl> <chr>
## 1 173.  1.68  A
## 2 880.  2.06  B
## 3 155.  1.91  D
## 4 226.  3.50  E
## 5  57.5 2.17  F
## 6 150.  0.954 I
## 7 150.  4.15  J
```

La fonction `neg_log_likelihood` calcul la -log-vraisemblance tel que décrite dans l'équation plus haut. Cependant il tout à fait possible d'optimiser avec la fonction `nlm` en partant d'une écrite plus simple de la fonction :

$$-lnL(\alpha, \beta, \vec{x}) = \sum_{i=1}^{n} ln(f_i(\alpha, \beta, x_i))$$

```
LL_Wei <- function(param, xi) {
    vec <- dweibull(xi, scale = param[1], shape = param[2])
    -sum(log(vec)) %>% return()
}
for (xi in durees_inter_panne) {
    print(nlm(LL_Wei,c(mean(xi),1),xi)$estimate)
}
```

```
## [1] 172.575642   1.684521
## [1] 879.851715   2.062461
## [1] 154.800522   1.913595
## [1] 225.613017   3.503191
## [1] 57.517548  2.173341
## [1] 149.8722625   0.9538739
## [1] 149.725384   4.151666
```

On a donc pour chaque composant du robot, l'estimations des paramètres de la durée de vie suivant une loi de Weibull.

Fonction `eweibull` Cette fonction du package `EnvStats` effectue l'estimation des paramètre d'une loi de Weibull suivant la méthode du maximum de vraisemblance directement à partir des données

```
library(EnvStats)
ewebull_estim <- c()
for (xi in durees_inter_panne) {
    ewebull_estim <- c(ewebull_estim, eweibull(xi, method = "mle")$parameters)
}
```

```
## Warning in nlminb(start = 1, objective = mcf, lower = .Machine$double.eps, :
## NA/NaN function evaluation
```

```
ewebull_estim <- tibble::as_data_frame(
    matrix(ewebull_estim, ncol = 2, byrow = TRUE)
)
ewebull_estim$Repère <- c("A", "B", "D", "E", "F", "I", "J")
colnames(ewebull_estim) <- c("Beta", "Alpha", "Repère")
ewebull_estim
```

```
## # A tibble: 7 x 3
##    Beta Alpha Repère
##   <dbl> <dbl> <chr>
## 1 1.68  173.  A
## 2 2.06  880.  B
## 3 1.91  155.  D
## 4 3.50  226.  E
## 5 2.17   57.5 F
## 6 0.954 150.  I
## 7 4.15  150.  J
```

```
ewebull_estim
```

```
## # A tibble: 7 x 3
##    Beta Alpha Repère
##   <dbl> <dbl> <chr>
## 1 1.68  173.  A
## 2 2.06  880.  B
## 3 1.91  155.  D
## 4 3.50  226.  E
## 5 2.17   57.5 F
## 6 0.954 150.  I
## 7 4.15  150.  J
```

```
estim_max_likelihood
```

```
## # A tibble: 7 x 3
##   Alpha  Beta Repère
##   <dbl> <dbl> <chr>
## 1 173.  1.68  A
## 2 880.  2.06  B
## 3 155.  1.91  D
## 4 226.  3.50  E
## 5  57.5 2.17  F
## 6 150.  0.954 I
## 7 150.  4.15  J
```

Malgré de très légères différences, les deux solutions permettent d'obtenir les mêmes estimateurs.

Je décide de conserver les résultat obtenus à l'aide de la première méthode contenus dans le vecteur `estim_max_likelihood`.

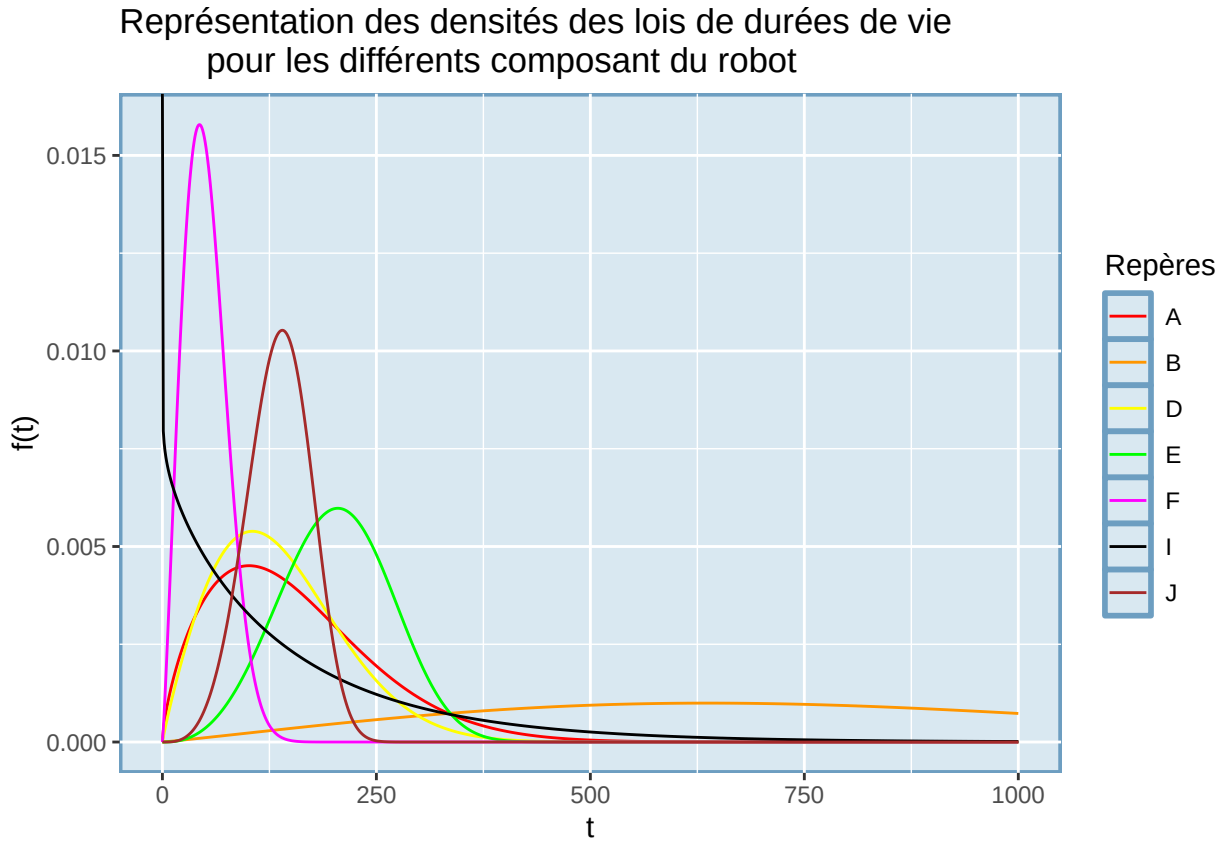
```
# Place la colonne Repère en premier dans on dataframe
estim_max_likelihood <- estim_max_likelihood %>% relocate(Repère)
```

On peut ensuite tracer les densités associées et donner le $MTTF$ pour mieux se représenter les comportement des composants.

```
composants <- c("Electrovanne pistolet",
                "Vérins ",
                "Bras horizontal - Poignets de programmation",
                "Nez robot",
                "Fin de course support bras",
                "Carte DH",
                "Carte Servo")
reperes <- y_set

t_div <- seq(0,1000,1)
```

```
ggplot() +
    geom_line(aes(x = t_div,
                  y = dweibull(t_div,
                               scale = ewebull_estim$Alpha[1],
                               shape = ewebull_estim$Beta[1]),
              colour = "A")) +
    geom_line(aes(x = t_div,
                  y = dweibull(t_div,
                               scale = ewebull_estim$Alpha[2],
                               shape = ewebull_estim$Beta[2]),
                  colour = "B")) +
    geom_line(aes(x = t_div,
                  y = dweibull(t_div,
                               scale = ewebull_estim$Alpha[3],
                               shape = ewebull_estim$Beta[3]),
              colour = "D")) +
    geom_line(aes(x = t_div,
                  y = dweibull(t_div,
                               scale = ewebull_estim$Alpha[4],
                               shape = ewebull_estim$Beta[4]),
              colour = "E")) +
    geom_line(aes(x = t_div,
                  y = dweibull(t_div,
                               scale = ewebull_estim$Alpha[5],
                               shape = ewebull_estim$Beta[5]),
              colour = "F")) +
    geom_line(aes(x = t_div,
                  y = dweibull(t_div,
                               scale = ewebull_estim$Alpha[6],
                               shape = ewebull_estim$Beta[6]),
                    colour = "I")) +
    geom_line(aes(x = t_div,
                  y = dweibull(t_div,
                               scale = ewebull_estim$Alpha[7],
                               shape = ewebull_estim$Beta[7]),
                  colour = "J")) +
    scale_color_manual(name = "Repères",
                       values = c("A" = 'red',
                                  "B" = "#FF9600",
                                  "D" = 'yellow',
                                  "E" = 'green',
                                  "F" = 'magenta',
                                  "I" = 'black',
                                  "J" = 'brown')) +
    labs(title = "Représentation des densités des lois de durées de vie
         pour les différents composant du robot",
         x = "t",
         y = "f(t)") +
    custom_theme
```



On peut ensuite calculer le $MTTF$. Pour une loi de Weibull, on a $E[X_i] = \alpha\Gamma(1 + \frac{1}{\beta})$

```
estim_max_likelihood$MTTF <- as.double(1:7)

for (i in 1:7) {
    estim_max_likelihood[i,'MTTF'] <- ewebull_estim$Alpha[i] *
        gamma(1+1/ewebull_estim$Beta[i])
}
estim_max_likelihood
```

```
## # A tibble: 7 x 4
##   Repère Alpha  Beta  MTTF
##   <chr>  <dbl> <dbl> <dbl>
## 1 A      173.  1.68  154.
## 2 B      880.  2.06  779.
## 3 D      155.  1.91  137.
## 4 E      226.  3.50  203.
## 5 F       57.5 2.17   50.9
## 6 I      150.  0.954 153.
## 7 J      150.  4.15  136.
```

Toutes les valeurs manipulées sont en heures de fonctionnement.

Je propose de les convertir en minutes.

```
estim_max_likelihood <- estim_max_likelihood %>%
  rename(Alpha_h = Alpha) %>%
  rename(MTTF_h = MTTF) %>%
```

```
  mutate(Alpha_m = Alpha_h * 60) %>%
  mutate(MTTF_m = Alpha_m * gamma(1+1/Beta)) %>%
  relocate(Alpha_m, .after = Alpha_h)
estim_max_likelihood
```

```
## # A tibble: 7 x 6
##   Repère Alpha_h Alpha_m  Beta MTTF_h MTTF_m
##   <chr>    <dbl>   <dbl> <dbl>  <dbl>  <dbl>
## 1 A        173.   10355. 1.68   154.   9245.
## 2 B        880.   52791. 2.06   779.  46764.
## 3 D        155.    9288. 1.91   137.   8240.
## 4 E        226.   13537. 3.50   203.  12180.
## 5 F         57.5   3451. 2.17    50.9  3056.
## 6 I        150.    8992. 0.954  153.   9185.
## 7 J        150.    8984. 4.15   136.   8160.
```

Estimation des distributions des durées de changement correctif pour chaque élément.

Après avoir estimé les les paramètres des lois de Weibull des durées inter-panne (loi de durée de vie) des composants du robot, on va maintenant chercher à estimer les temps de remplacement pour chacun des ces derniers.

On va exploiter, pour cela, les données issues du fichier de retour d'expérience.

On peut réutiliser la même démarche que précédemment, en cherchant les estimateurs du maximum de vraisemblance. Cette fois, on suppose que les durées suivent des lois exponentielles.

Cette loi étant plus simple que la loi de Weibull, il est possible d'obtenir l'estimateur du maximum de vraisemblance analytiquement.

$$\hat{\lambda}_{MV} = \frac{n}{\sum_{i=1}^{n}(x_i)} = \frac{1}{\hat{m}}$$

Soit l'inverse de la moyenne empirique de l'échantillon mesurée $\hat{m}$.

Les durée de temps d'arrêt stockée dans `df_abc_noNA` étant en minutes, je propse de les convertir en heures pour garder la cohérence avec les durée de fonctionnement estimées précédement.

```
df_abc_noNA <- df_abc %>%
  mutate(temps_arret_h = temps_arret_min * (1/60))
df_abc_noNA <- df_abc_noNA %>% relocate()
df_abc_noNA
```

```
## # A tibble: 40 x 4
##    `temps d'arrêt` Repère temps_arret_min temps_arret_h
##    <chr>           <chr>            <dbl>         <dbl>
##  1 20 min          E                   20         0.333
##  2 45 min          E                   45         0.75
##  3 25 min          E                   25         0.417
##  4 94 min          I                   94         1.57
##  5 10 min          J                   10         0.167
##  6 10 min          <NA>                10         0.167
##  7 30 min          H                   30         0.5
##  8 30 min          A                   30         0.5
##  9 10 min          G                   10         0.167
```

```
## 10 15 min          B                   15         0.25
## # i 30 more rows
```

```
estim_mv_expo <- function(xi) {
    1/mean(xi) %>% return()
}
# Effectue le calcul de l'inverse de la moyenne par repère
estim_param_duree_remplacement <- aggregate(
  temps_arret_h~Repère,
  data = df_abc_noNA,
  estim_mv_expo
)
estim_param_duree_remplacement <- estim_param_duree_remplacement %>%
  rename(lambda_h = temps_arret_h)
estim_param_duree_remplacement
```

```
##   Repère  lambda_h
## 1      A 1.6216216
## 2      B 4.0000000
## 3      D 0.6792453
## 4      E 0.8432432
## 5      F 3.3750000
## 6      G 6.0000000
## 7      H 2.0000000
## 8      I 0.6382979
## 9      J 6.0000000
```

On peut ensuite calculer le temps moyen de remplacement :

```
estim_param_duree_remplacement$temps_arret_moyen_h <- 1/estim_param_duree_remplacement$lambda_h
estim_param_duree_remplacement$lambda_m <- estim_param_duree_remplacement$lambda_h / 60
estim_param_duree_remplacement$temps_arret_moyen_m <- 1/estim_param_duree_remplacement$lambda_m
estim_param_duree_remplacement
```

```
##   Repère  lambda_h temps_arret_moyen_h   lambda_m temps_arret_moyen_m
## 1      A 1.6216216           0.6166667 0.02702703            37.00000
## 2      B 4.0000000           0.2500000 0.06666667            15.00000
## 3      D 0.6792453           1.4722222 0.01132075            88.33333
## 4      E 0.8432432           1.1858974 0.01405405            71.15385
## 5      F 3.3750000           0.2962963 0.05625000            17.77778
## 6      G 6.0000000           0.1666667 0.10000000            10.00000
## 7      H 2.0000000           0.5000000 0.03333333            30.00000
## 8      I 0.6382979           1.5666667 0.01063830            94.00000
## 9      J 6.0000000           0.1666667 0.10000000            10.00000
```

On a donc a présent pour les composants du robot : - un estimation de la durée de vie du composant (loi de Weibull et les paramètres associés) - une estimation du temps de remplacement du composant (temps d'immobilisation du robot engendré. Loi exponentielles et le paramètre associé).

Discussions des résultats

Les résultats obtenus peuvent être critiqués et doivent être mit en perspective avec le contexte général de l'analyse.

Premièrement, nous avons écarté deux défaillance de la classification de Pareto.

Puisque que ces défaillances ne touchent pas directement une pièce du robot et qu'elles sont rare et peu impactant sur le temps d'arrêt (10 et 35 minutes), ces deux défaillances auraient été incluses dans la classe

C. Nous n'aurions donc dans tous les cas pas concentrer nos efforts sur l'optimisation de la politique de maintenance de ces éléments.

Pour les estimations effectués, plusieurs éléments doivent être prit en compte :

- nous avons supposé les lois de probabilité (Weibull et Exponentielle) sans pour autant avoir testé l'adéquation de ces lois aux données.
- nous disposons de très peu de donnée en général (parfois une seule panne référencée pour certain composants).

Les résultats obtenus permettent donc une première analyses, mais devraient être améliorés si nous avions l'opportunité de récupérer plus de mesures.