RM04 - Projet n°2

Baptiste Toussaint

2024-06-22

Contents

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

```r
# 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.

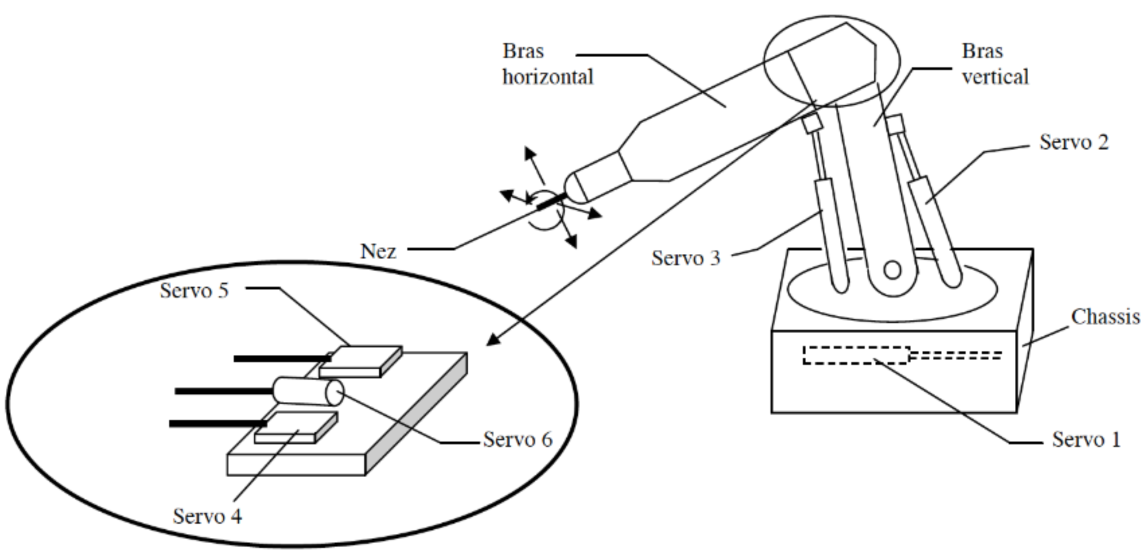


FIGURE 1 – Plan schématique d'un robot de peinture

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.

Classification

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.

```r
# 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.

```r
# 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.

```r
# 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
```

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

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

```r
# 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
```

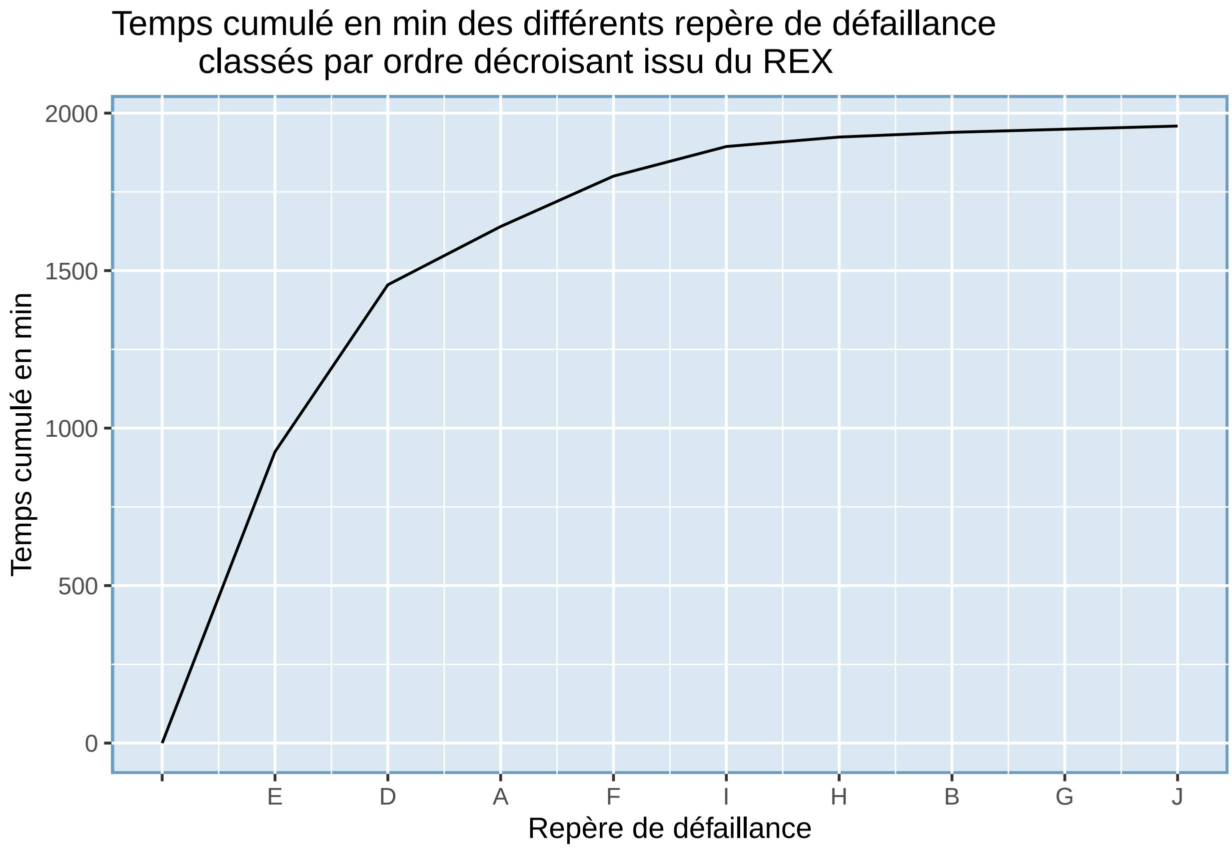
```
##  [1] ""  "E" "D" "A" "F" "I" "H" "B" "G" "J"
```

```
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

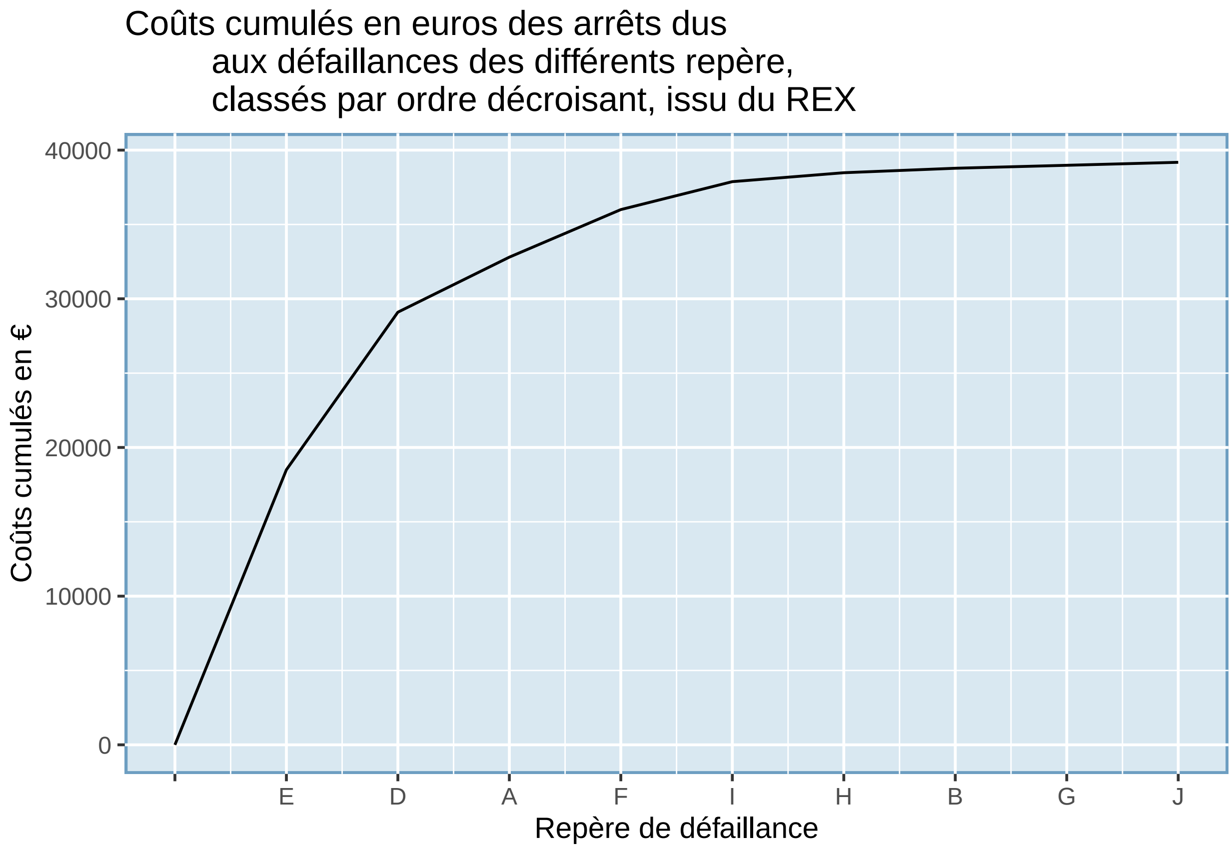
```
# ggplot permet de créer de plus jolis graphiques et est facilement
# utilisable pour un résultat proche de ce que l'on souhaite avoir
ggplot(mapping = aes(x = 0:9, y = y_set))+
    geom_line() +
    scale_x_continuous(breaks = 0:9,
                     labels = x_set) +
    labs(x = "Repère de défaillance",
         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 € :

```
# 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
```



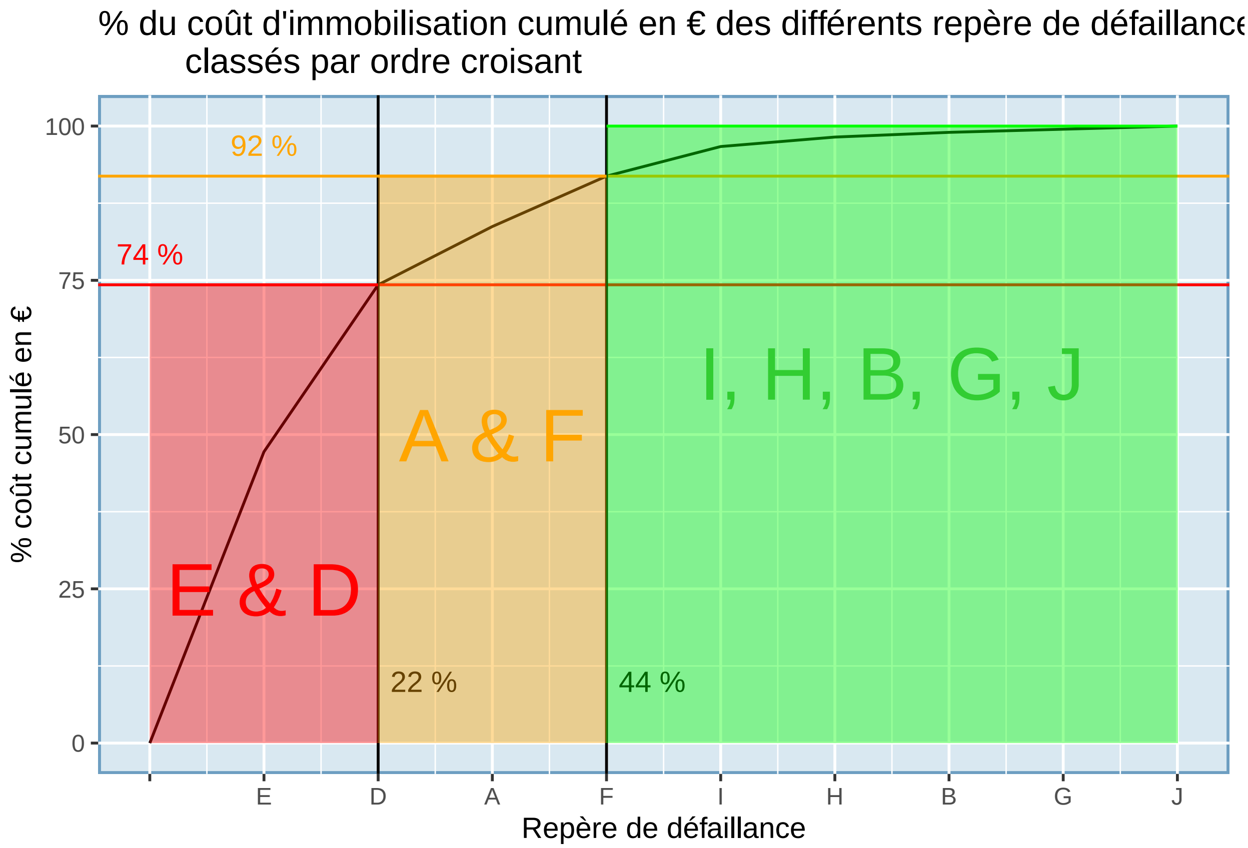
Pour simplifier la lecture, et distinguer nos classes A, B et C, je propose de retourner sur l'analyse des temps d'arrêt (équivalent au coût par un simple produit) et de raisonner en % du temps total d'arrêt mesuré.

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 croissant") +
    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_vline(xintercept = 2, color = 'black') +
    annotate(geom = "text",
```

```r
           x = 2+0.4,
           y = 10,
           label = paste(round((2/9)*100),"%")) +
  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_vline(xintercept = 4, color = 'black') +
  annotate(geom = "text",
           x = 4+0.4,
           y = 10,
           label = paste(round((4/9)*100),"%")) +
  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
```



Cette premièr manière de classifier les éléments est cohérente si l'on observe les données :

```
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
```

On voit bien qu'il y a un changement de comportement visible entre le composant D et le composant A, de même qu'entre le composant F et le composant I (on passe de 530 à 185 puis de 160 à 94).

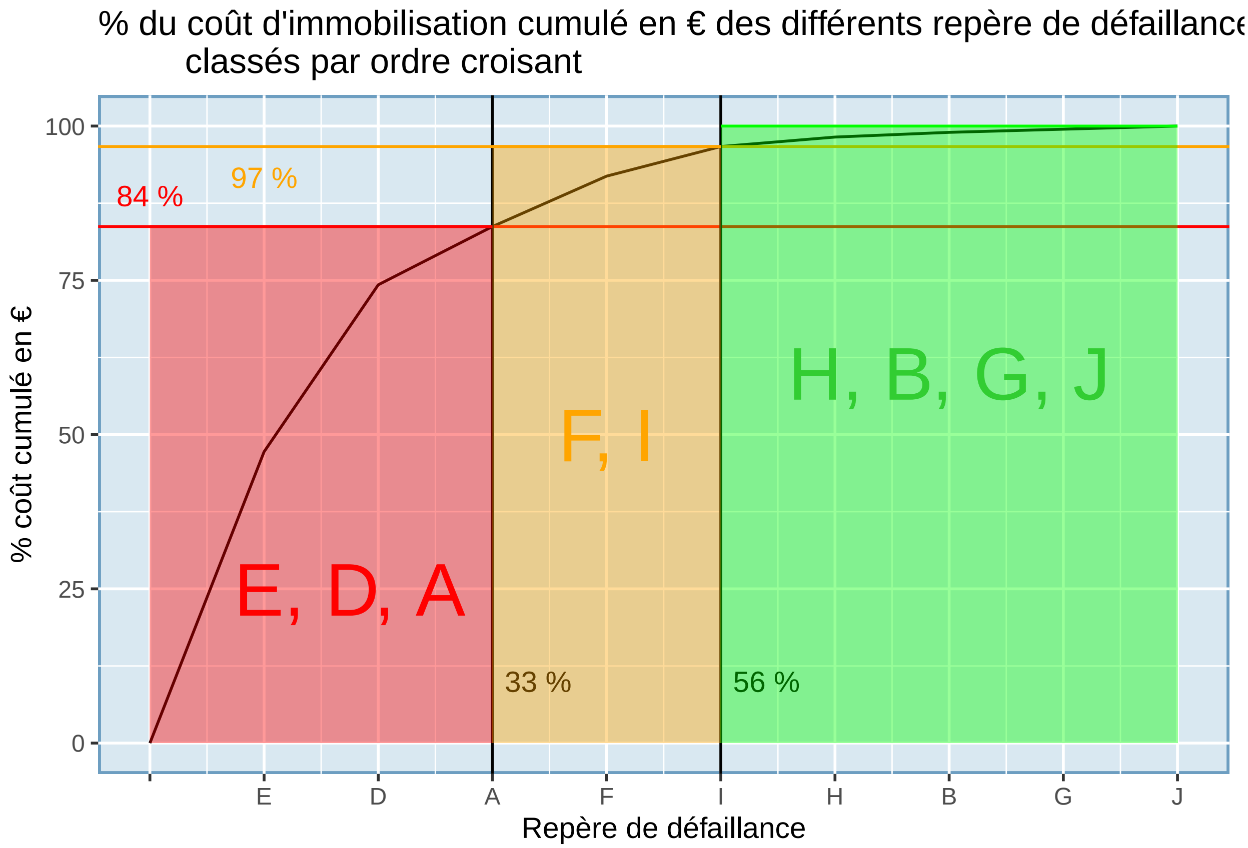
Autre classification possible

Notre classe A représente bien environ 20% des données mais seulement 74% des effets.

On peut aussi décider d'inclure le composant A dans la classe A et le composant I à la classe B, ce qui donne :

```
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) +
```

```r
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[4]/temps_arret_total,
           color = 'red') +
annotate(geom = "text",
         x = 0,
         y = 100*y_set[4]/temps_arret_total+5,
         label = paste(
             round(100*y_set[4]/temps_arret_total),"%"),
         color = 'red') +
geom_vline(xintercept = 3, color = 'black') +
annotate(geom = "text",
         x = 3+0.4,
         y = 10,
         label = paste(round((3/9)*100),"%")) +
geom_area(aes(x = 0:3, y = 100*y_set[4]/temps_arret_total),
          fill = 'red',
          alpha = 0.4,
          color = 'red') +
geom_hline(yintercept = 100*y_set[6]/temps_arret_total,
           color = 'orange') +
annotate(geom = "text",
         x = 1,
         y = 100*y_set[6]/temps_arret_total-5,
         label = paste(
             round(100*y_set[6]/temps_arret_total),"%"),
         color = 'orange') +
geom_vline(xintercept = 5, color = 'black') +
annotate(geom = "text",
         x = 5+0.4,
         y = 10,
         label = paste(round((5/9)*100),"%")) +
geom_area(aes(x = 3:5, y = 100*y_set[6]/temps_arret_total),
          fill = 'orange',
          alpha = 0.4,
          color = 'orange') +
geom_area(aes(x = 5:9, y = 100),
          fill = 'green',
          alpha = 0.4,
          color = 'green') +
annotate(geom = "text", x = 1.75, y = 25, label = "E, D, A", color = 'red',
         size = 10) +
annotate(geom = "text", x = 4, y = 50, label = "F, I", color = 'orange',
         size = 10) +
annotate(geom = "text", x = 7, y = 60, label = "H, B, G, J", color = 'limegreen',
         size = 10) +
custom_theme
```



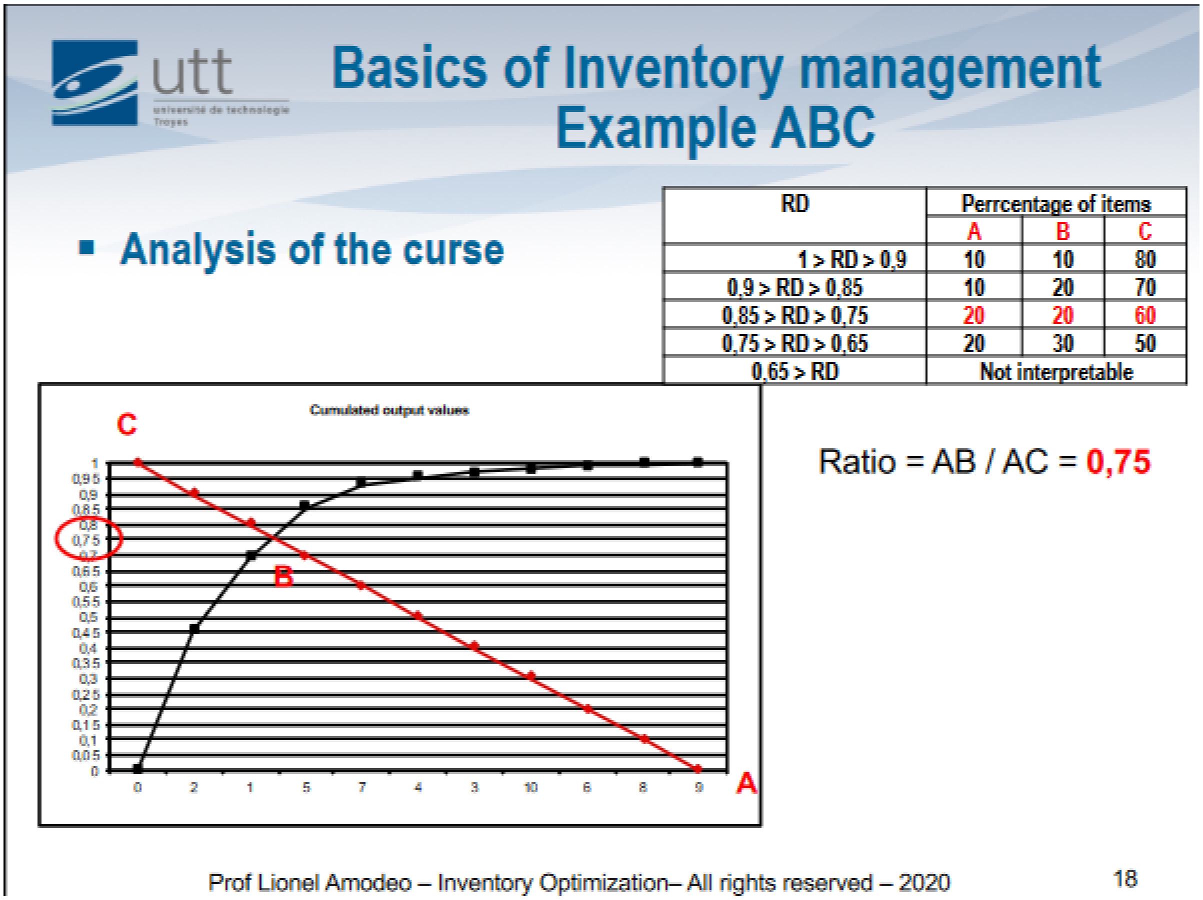
Avec cette autre proposition de classification, on inclue le composant A à la classe A, prioritaire.

Avec les éléments disponibles, je ne suis pas en mesure de trancher sur la classification la plus pertienente pour l'entreprise.

Le choix peut se faire par exemple à partir du retour d'expérience des opérateur qui jugeront si le composant A est effectivement aussi prioritaire que les composants E et D

Analsye du ratio discriminant

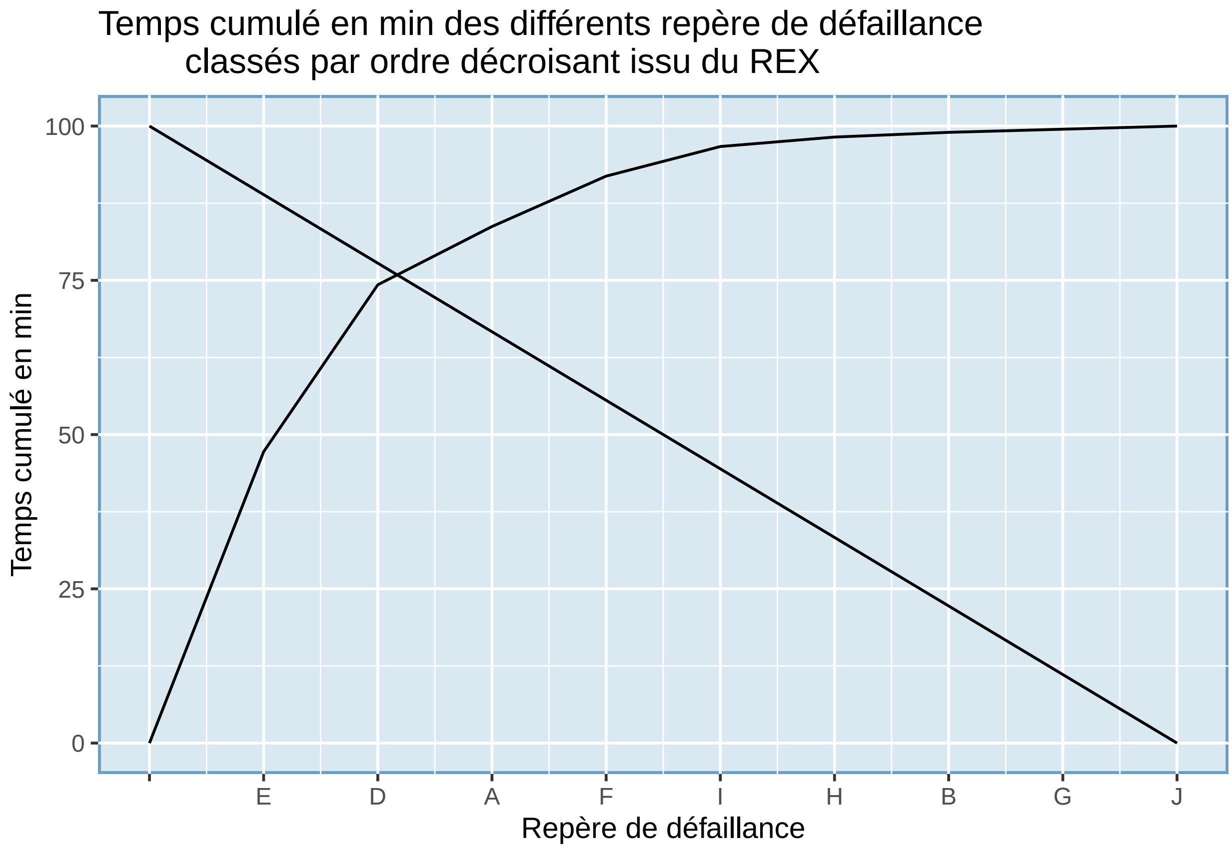
Pour trancher entre les deux méthodes, je propose de réexploiter la méthode présentée par Pr. Amodeo dans l'UE GP27 : gestion des stocks.



Dans cette méthode, on définit le pourcentage d'élements présents dans chaque classe en fonction du calcul d'un ratio qui peut se déduire graphiquement par le point d'intersection entre la droite d'équation : $y = 100 - \frac{100}{\text{Nombre d'éléments considérés}} x$ et la courbe.

On utilise ensuite le tableau pour déduire les pourcentages déléments présents dans chaque classe en fonction de ce ratio RD.

```
# 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*100/max(y_set)))+
    geom_line() +
    geom_line(aes(x = 0:9, y = -(100/9)*(0:9)+100)) +
    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
```



Par lecture graphique on en conclut que $0.85 > RD > 0.75$. Si l'on reprend le tableau présenté plus haut, on en conclut que :
- la classe A doit représenter 20
- la classe B doit représenter 20
- la classe C doit représenter 60

Ce résultat conforte la première classification proposée avec :
- classe A : composants E et D (22
- classe B : composants A et F (22
- classe C : le reste des composants (56

Conclusion de la classification

Nous avons donc identifié, à l'aide de la classification de Pareto augmentée par analyse du ratio discriminant, les composants prioritaires : E et D, qui sont rangés dans ce qu'on nommera à présent la "Classe A".

Les composants A et F, rangés dans la "Classe B" sont des composants jugés non prioritaires mais dont la maintenance peut être optimisée si cette optimisation peut se faire à moindre coûts.

Estimation des modèles de comportement des composants

Pour proposer une optimisation de la politique de maintenance, il est essentiel de d'abord définir les modèles de durée de vie des composants et les modèles de temps de réparation.

Notre optimisation reposant sur des simulations du systèmes, nous devons définir le comportant de nos composants à travers ces modèles de maintenance et de durée de vie.

Modèle mathématique

Classiquement, les durée de vie des composants et les temps de réparation sont modélisés par des variables aléatoires.

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 $\mathbb{R}_+^*$.

> Cela signifie que le temps entre la mise en service du composant et sa panne est une valeur T_ω aléatoire (que l'on ne peut prédire) positive plus grande que 0

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

Cette hypothèse est classique dans les différents modèles de maintenance et semble avoir été confirmé par les experts de l'entreprise.

On considère cette hypothèse comme vraie tout en gardant à l'esprit qu'il serait préférable de la confirmer avec un plan d'expérience supplémentaire.

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

Dans une loi de Weibull, le paramètre α_ω définit l'ordre de grandeur de la valeur T_ω (plus α_ω est grand plus T_ω est grand). β_ω quand à lui sert à décrire le vieillissement du composant.

En fonction des contextes il existe plusieurs familles de méthodes d'estimation possible :

- les estimations paramétriques, qui sont utilisée quand on a une hypothèse de modèle (c'est notre cas ici).
- les estimations non paramétriques, en l'absence de modèle
- les estimations bayésiennes, pouvant intégrer des avis d'experts (utiles avec peu de données).

J'ai décidé d'utiliser une estimation paramétrique par méthode du maximum de vraisemblance.

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 $\vec{x}$ 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 par 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 qui est très calculatoire.

Estimations des lois de panne

Construction des données

Préalablement aux calcul numérique de nos estimateurs, nous devons préparer nos données.

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

```r
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.

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 :

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

```r
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()

}
```

```r
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
```

Les deux solutions obtiennent des résultats très proches, ce qui nous conforte dans nos estimations.

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

On peut ensuite tracer les densités associées se représenter le comportement des composants.

Plusieurs analyse peuvent être faite rien qu'avec ce graphique :

- le composant I (coubre noire) (Classe C non prioritaire), suit en réalité une loi exponentielle (que l'on reconnait à la forme de la densité). On a donc : $\hat{\beta}_{I_{MV}} \simeq 1$.
- les estimations ne sont pas effectuées avec le même nombre de mesures pour chaque composant, ces résultats doivent être interprétés avec précaution.
- le composant E (courbe verte), bien que présent dans la classe A, ne semble pas avoir la valeur moyenne (proche du point le plus haut de la courbe) le plus faible : ce n'est donc pas le composant qui tombe le plus vite en panne. Pour rappel, la classification est faite sur le temps d'inactivité (donc le temps de réparation) et non la durée de vie des composants. Si un composant tombe souvent en panne mais prend quelques minutes à remplacer, il ne sera pas prioritaire car il n'engendrera pas beaucoup de coûts d'inactivité.

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 pour pouvoir les manipuler avec les temps de remplacements (eux en minutes).

```r
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.
```

On peut à ce stade conclure par exemple que, d'après notre étude statistique basée sur les données récoltées le composant E tombe en panne au bout de d'environ $203h$ de fonctionnement.

Estimation des lois de réparation des composants

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 propose de les convertir en heures pour garder la cohérence et conserver les deux options pour plus tard.

```r
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
```

```r
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 :

```r
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
```

Conclusion et discussions des résultats

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).

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.

Optimisation de la politique de maintenance pour les éléments de la classe A

Nous sommes maintenant en mesure de procéder à l'optimisation des éléments de la classe A.

L'optimisation s'effectuera en deux temps. On cherchera abord à construire une méthode pour simuler le coût moyen par unité de temps d'une politique de maintenance avec une période de visite préventive quelconque T.

On cherchera ensuite les périodes de remplacement périodiques optimales $T*$ pour ces composants.

La simulation est très souvent la seule solution abordable pour optimiser une politique de maintenance.

Pour ce type de problèmes avec plusieurs variables aléatoires dépendantes, le calcul formel des résultats optimaux sont bien souvent des entreprises très longues et complexe.

Quelques hypothèses doivent être précisées.

Je considère une politique de maintenance basée sur l'âge. Quand le système tombe en panne, un opérateur est immédiatement informé de la panne et commence immédiatement la réparation. On considère donc négligeable le temps de déplacement de l'opérateur au robot, et on considère qu'il possède immédiatement à porté les outils et pièces de rechange nécessaires à la maintenance.

Détermination du coût de maintenance moyen par unité de temps en fonction de la politique de maintenance pour tout T.

On cherche à simuler le coût moyen par unité de temps de la maintenance en fonction de la politique de maintenance.

Deux coûts se rapportent à la maintenance d'un composant :

- le coût d'immobilisation du robot $ci = 20€/min$.
- le coût de remplacement, fixe, $cc = 30€$.

Si l'on note T_r la date de remise en service d'un composant après la panne, et CC_r le coût total de la maintenance à la date T_r, le coût par unité de temps de la maintenance, noté CUT sera alors égal à :

$$CUT = \frac{CC_r}{T_r}$$

Toutes ces grandeurs sont dépendante de la période d'inspection T, on peut alors noter :

$$CUT(T) = \frac{CC_r(T)}{T_r(T)}$$

Cette grandeur est une variable aléatoire, hors pour comparer les performances d'une politique de maintenance, il faut raisonner en moyenne, on cherche donc à calculer l'espérance du coût moyen par unité de temps, soit :

$$\mathbb{E}[CUT(T)] = \frac{\mathbb{E}[CC_r(T)]}{\mathbb{E}[T_r(T)]}$$

Détermination de la loi de remise en service $T_r(T)$

On considère une politique de remplacement basée sut âge de élément. A partir de la mise en service du robot (soit la mise en service initiale, soit après le remplacement d'une pièce), on note $T_r(T)$ la durée aléatoire du cycle jusqu'à la prochaine remise en service (après une panne ou une maintenance préventive). On a donc :

- T la période de remplacement préventif, déterministe, dont on cherchera à optimiser la valeur pour réduire les coûts de la politique de maintenance.
- T_f la date aléatoire de la prochaine défaillance, qui dépend de la fiabilité du composant. Si on effectue un remplacement préventif, le composant est remit à neuf.
- la durée d'inactivité tp due à un remplacement préventif
- la durée d'inactivité due à un remplacement correctif D.

T_r est donc une fonction de ces différents éléments, que l'on cherche à déterminer.

La remise en service à une date T_r est forcément consécutive à un remplacement correctif ou préventif. Les deux types de remplacement sont disjoint, on peut donc conditionner la date T_r au type de remplacement effectué.

La date T de remplacement périodique étant fixé après un temps T de fonctionnement sans défaillance, le type de remplacement dépend donc de la variable aléatoire T_f la date de prochaine défaillance.

On a donc :

$$\begin{cases} \text{Remplacement correctif si : } T_f < T \\ \text{Remplacement préventif si : } T_f > T \end{cases}$$

Si la maintenance est préventive pour un élément de la classe A noté ω, on a :

$$P(\text{Remplacement préventif } |\omega) = P(T_f > T|\omega) = 1 - F_\omega(T)$$

Puisque T dépend de notre politique de maintenance, et que l'on a déterminé les paramètres des lois pour tous les composants, alors cette probabilité est fixe.

Dans le cas d'un remplacement préventif, on est certain que la date de remise en service sera égal à $T + tp$ puisque que le temps de changement est fixe et dure tp minutes, et qu'il a lieu après un temps T sans défaillance.

Donc :

$$T_r|\text{Préventif} = T + p$$

Donc le temps moyen de remise en service si le remplacement est préventif est aussi de $T + tp$ puisque l'espérance d'une constante est égal à cette constante.

$$\mathbb{E}[T_r|\text{Préventif}] = \mathbb{E}[T + tp] = T + tp$$

Si la maintenance est corrective pour un élément ω de la classe A:

$$P(\text{Remplacement correctif }|\omega) = P(T_f < T|\omega) = F_\omega(T) = 1 - R_\omega(T) = 1 - R_{\mathbf{W}(\alpha_\omega,\beta_\omega)}(T) = F_{\mathbf{W}(\alpha_\omega,\beta_\omega)}(T)$$

Et :

$$T_r|\text{Correctif} = T_f + D$$

Et :

$$\mathbb{E}[T_r|\text{Correctif}] = \mathbb{E}[T_f] + \mathbb{E}[D]$$

On peut alors donner une expression de T_r :

$$T_r = \begin{cases} T_f + D & \text{si } T_f \leq T \\ T + tp & \text{si } T_f > T \end{cases}$$

Ce résultat suppose que ma maintenance est auto-détectable ce qui colle avec la politique de maintenance demandée par le client.

Le calcul explicite de $\mathbb{E}[T_r]$ est complexe. On passe donc par la simulation pour trouver la valeur de $\mathbb{E}[T_r]$ pour chaque composants.

Création d'une fonction de simulation pour $\mathbb{E}[T_r]$

Je propose de créer une fonction `simulation_esp_Tr` qui peut simuler $\mathbb{E}[T_r]$ en fonction : - du composant étudié - des paramètres du système

Attention aux unités utilisées. Nos informations utiles sont stockées dans les structures : - `estim_param_duree_remplacement` pour les paramètres des durées de remplacement, en heures ou minutes. - `estim_max_likelihood` pour les lois de durées de vie.

```
# paramètres au format
# param = list(alpha = X, beta = X, lambda = X, T = X, tp = X)
simulation_esp_Tr <- function(param, p = 1000) {
  # MTTF du composant
  esp_Tf <- param$alpha * gamma(1 + 1/param$beta)
  # Durée moyenne de remplacement périodique du composant
  esp_D <- 1/param$lambda

  # Routine pour calculer par simulation l'espérance de la date de remise en
  # service E[Tr]
  # Simuler une trajectoire revient à simuler le système et récupérer la date
  # de remise en service Tr
  # Pour obtenir la moyenne, on recommence le processus un grand nombre de fois et
  # on prend la moyenne
```

```
  # contient les trajectoires du système.
  list_traj <- c()
  for (i in 1:p) {
      # simule une date de panne aléatoire
      Tf <- rweibull(1, shape = param$beta, scale = param$alpha)
      # simule un temps de réparation corrective aléatoire
      D <- rexp(1, rate = param$lambda)
      if (Tf <= T) {
          # Dans ce cas maintenance corrective
          list_traj <- c(list_traj, Tf + D)
      } else {
          # Dans ce cas maintenance préventive
          list_traj <- c(list_traj, T + param$tp)
      }
  }

  return(mean(list_traj))
}
```

On peut tester notre fonction pour le composant E et un $T = 200h$.

```
parametres = list(
  alpha = estim_max_likelihood[estim_max_likelihood$Repère == "E",]$Alpha_h,
  beta = estim_max_likelihood[estim_max_likelihood$Repère == "E",]$Beta,
  lambda = estim_param_duree_remplacement[estim_param_duree_remplacement$Repère == "E",]$lambda_h,
  tp = 10 / 60, # en heures
  T = 200 # en heures
)
simulation_esp_Tr(parametres, p = 10000)
```

```
## [1] 1.166667
```

Cette fonction permet donc de simuler le système avec une politique de maintenance pour T et d'observer le temps moyen avant la panne et remise en service (cycle) du composant E.

Détermination du coût cumulé jusqu'à la remise en service

Pour obtenir cette information, on va procéder de la même façon que pour la loi de remise en service.

Le coût cumulé de maintenance à la remise en service, noté CC_r est une variable aléatoire dépendant du type de maintenance. Elle est composée :

- du coût de remplacement fixe : $cc = 30$€.
- du coût d'inactivité : $ci = 20$€.

On peut alors écrire : $CC_r = cc + \text{Temps à l'arrêt en minutes } * ci$.

Si la maintenance est corrective (donc si $T_f < T$) alors on a : $\text{Temps à l'arrêt en minutes } = D$ qui dépend du type de composant considéré. Dans ce cas si $T_f < T$ alors $CC_r = cc + D \cdot ci = 30 + 20D(€)$

Si la maintenance est préventive (donc si $T_f > T$) alors on a : $\text{Temps à l'arrêt en minutes} = tp = 10 minutes$.

Dans ce cas si $T_f > T$ alors $CC_r = cc + tp \times ci = 30 + 10 \times 20 = 230$€

$$CC_r = \begin{cases} cc + D \cdot ci & \text{si } T_f \leq T \\ cc + tp \cdot ci & \text{si } T_f > T \end{cases}$$

Comme T_f et D sont deux variables indépendantes on peut alors calculer facilement l'espérance du coût de maintenance total pour T :

$$\mathbb{E}[CC_r(T)] = F_{\mathbf{W}(\alpha_\omega,\beta_\omega)}(T) \cdot (cc + \mathbb{E}[D] \cdot ci) + (1 - F_{\mathbf{W}(\alpha_\omega,\beta_\omega)}(T)) \cdot (cc + tp \cdot ci)$$

Le coût final à T_r ne dépend donc que de T en moyenne.

Création d'une fonction de simulation pour $\mathbb{E}[CC_r]$

On peut à la fois utiliser la formule littérale ou une simulation pour calculer ce coût pour en fonction des paramètres du systèmes (composant ou T).

```r
# param = list(alpha = X, beta = X, lambda = X, T = X, tp = X,
# cc = X, ci = X, tp = X)
calcul_esp_CCr <- function(param) {
  esp_D <- 1/param$lambda
  FT <- pweibull(param$T, shape = param$beta, scale = param$alpha)
  return(
    FT * (param$cc + esp_D * param$ci) + (1-FT) * (param$cc + param$tp * param$ci)
  )
}

simulation_esp_CCr <- function(param, p = 1000) {

  # Routine pour calculer par simulation l'espérance du coût à date de remise
  # en service E[CCr]
  # Simuler une trajectoire revient à simuler le système et récupérer le coût total
  # à date de remplacement
  # Pour obtenir la moyenne, on recommence le processus un grand nombre
  # de fois et on prend la moyenne

  # contient les trajectoires du système.
  list_traj <- c()
  for (i in 1:p) {
      # Simule une date de panne
      Tf <- rweibull(1, shape = param$beta, scale = param$alpha)
      # Simule une durée de réparation
      D <- rexp(1, rate = param$lambda)
      if (Tf <= param$T) {
          # Maintenance corrective
          list_traj <- c(list_traj, param$cc + D * param$ci)
      } else {
          # Maintenance préventive
          list_traj <- c(list_traj, param$cc + param$tp * param$ci)
      }
  }
  return(mean(list_traj))
}
```

On peut tester ces deux méthodes :

```r
parametres = list(
  alpha = estim_max_likelihood[estim_max_likelihood$Repère == "E",]$Alpha_h,
  beta = estim_max_likelihood[estim_max_likelihood$Repère == "E",]$Beta,
  lambda = estim_param_duree_remplacement[
```

```r
    estim_param_duree_remplacement$Repère == "E",
  ]$lambda_h,
  tp = 10 / 60, # en heures
  T = 200, # en heures
  # Temps de remplacement périodique en heures
  tp = 10 / 60,
  # coût de changement fixe
  cc = 30,
  # coût d'indisponibilité par heures
  ci = 20 * 60
)

calcul_esp_CCr(param = parametres)
```

```
## [1] 818.1638
```

```r
simulation_esp_CCr(param = parametres, p = 50000)
```

```
## [1] 805.238
```

Évaluation de l'espérance du coût moyen par unité de temps de la maintenance

Nous disposons à présent de tous les éléments pour calculer le coût moyen par unité de temps en fonction des paramètres (T, composant) du système :

$$\text{Coût moyen par unité de temps de la maintenance} = \mathbb{E}[CUT(T)] = \frac{\mathbb{E}[CC_r(T)]}{\mathbb{E}[T_r(T)]}$$

On peut écrire une fonction qui effectue le calcul, cependant on ne peut pas juste de contenter de réutiliser les fonctions crées précédemment.

En effet, le calcul de $\mathbb{E}[CC_r]$ et de $\mathbb{E}[T_r]$ doit se faire sur les mêmes tirages alétoires.

Si l'on se contente de faire le quotient des deux fonction, le résultat final n'aurait pas de sens puisque l'on comparerai de grandeurs calculées à partir de données différentes.

Cependant, avec un très grand nombre de simulation, les deux méthodes doivent converger.

```r
# param = list(alpha = X, beta = X, lambda = X, T = X, tp = X,
# cc = X, ci = X, tp = X)
simulation_esp_cut <- function(param, p = 1000) {

  # Routine pour calculer par simulation l'espérance du coût à date de remise en
  # service E[CCr]
  # Simuler une trajectoire revient à simuler le système et récupérer le coût
  # total à date de remplacement
  # Pour obtenir la moyenne, on recommence le processus un grand nombre de fois
  # et on prend la moyenne

  # contient les trajectoires du système.
  list_traj <- list(temps = c(), cout = c())
  for (i in 1:p) {
      # Simule une date de panne
      Tf <- rweibull(1, shape = param$beta, scale = param$alpha)
      # Simule une durée de réparation
      D <- rexp(1, rate = param$lambda)
```

```
      if (Tf <= param$T) {
          # Maintenance corrective
          list_traj$temps = c(list_traj$temps, Tf + D)
          list_traj$cout = c(list_traj$cout, param$cc + D * param$ci)
      } else {
          # Maintenance préventive
          list_traj$temps = c(list_traj$temps, param$T + param$tp)
          list_traj$cout = c(list_traj$cout,
                             param$cc + param$tp * param$ci)
      }
  }

  return(mean(list_traj$cout)/mean(list_traj$temps))
}
```

Si on regarde pour le composant E :

```
parametres = list(
  alpha = estim_max_likelihood[estim_max_likelihood$Repère == "E",]$Alpha_h,
  beta = estim_max_likelihood[estim_max_likelihood$Repère == "E",]$Beta,
  lambda = estim_param_duree_remplacement[
    estim_param_duree_remplacement$Repère == "E",
  ]$lambda_h,
  tp = 10 / 60, # en heures
  T = 150, # en heures
  # Temps de remplacement périodique en heures
  tp = 10 / 60,
  # coût de changement fixe
  cc = 30,
  # coût d'indisponibilité par heures
  ci = 20 * 60
)

simulation_esp_cut(param = parametres, p = 5000)
```

```
## [1] 3.496034
```

Pour le composant E, pour ces paramètres de simulation : $T = 150$ on obtient un coût de 3.408627€.h^{-1}.

On a donc crée des fonctions permettant d'évaluer, pour un composant donné, une politique de maintenance.

On peut à présent chercher une méthode pour déterminer la période d'inspection T^*_ω pour chaque composant ω.

Optimisation de la politique de maintenance pour le composant E

Je propose de passer par une recherche graphique de l'optimum T^*_ω.

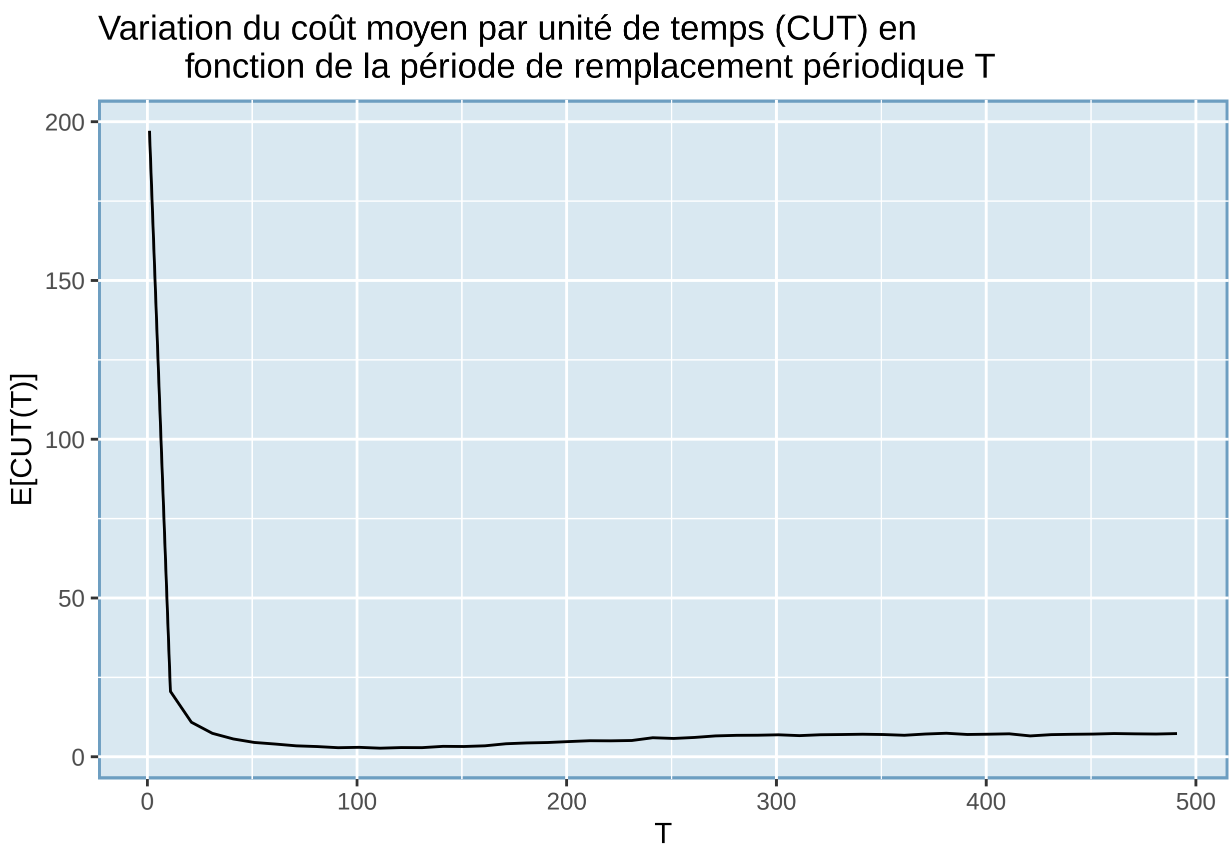
Cette méthode consiste à tracer l'évolution de $\mathbb{E}[CUT(T)]$ en fonction de T et d'identifier graphiquement le minimum.

```
# Fonction qui va afficher le graphique de l'évolution de E[CUT(T)] en fonction
# de T
# Prend trois arguments :
# - un vecteur T_set : une subdivision des valeurs de T
# - les paramètres, au format identiques aux précédents
# - un nombre de réplication.
```

```r
afficher_variation_cut <- function(T_set, param, p = 1000) {

    # vecteur stockant les valeurs successives de E[CUT(T)]
    res_list <- c()

    for (t in T_set) {
        # met à jour les paramètres en fonction de T étudié
        param$T <- t
        res_list <- c(res_list, simulation_esp_cut(param, p))
    }

    p<- ggplot() +
        geom_line(aes(x = T_set, y = res_list)) +
        labs(x = "T", y = "E[CUT(T)]",
             title = "Variation du coût moyen par unité de temps (CUT) en
        fonction de la période de remplacement périodique T") +
        custom_theme
    return(p)
}
```

On teste le résultat de cette fonction pour le composant E

```r
afficher_variation_cut(seq(1,500,10), parametres, p = 1000)
```

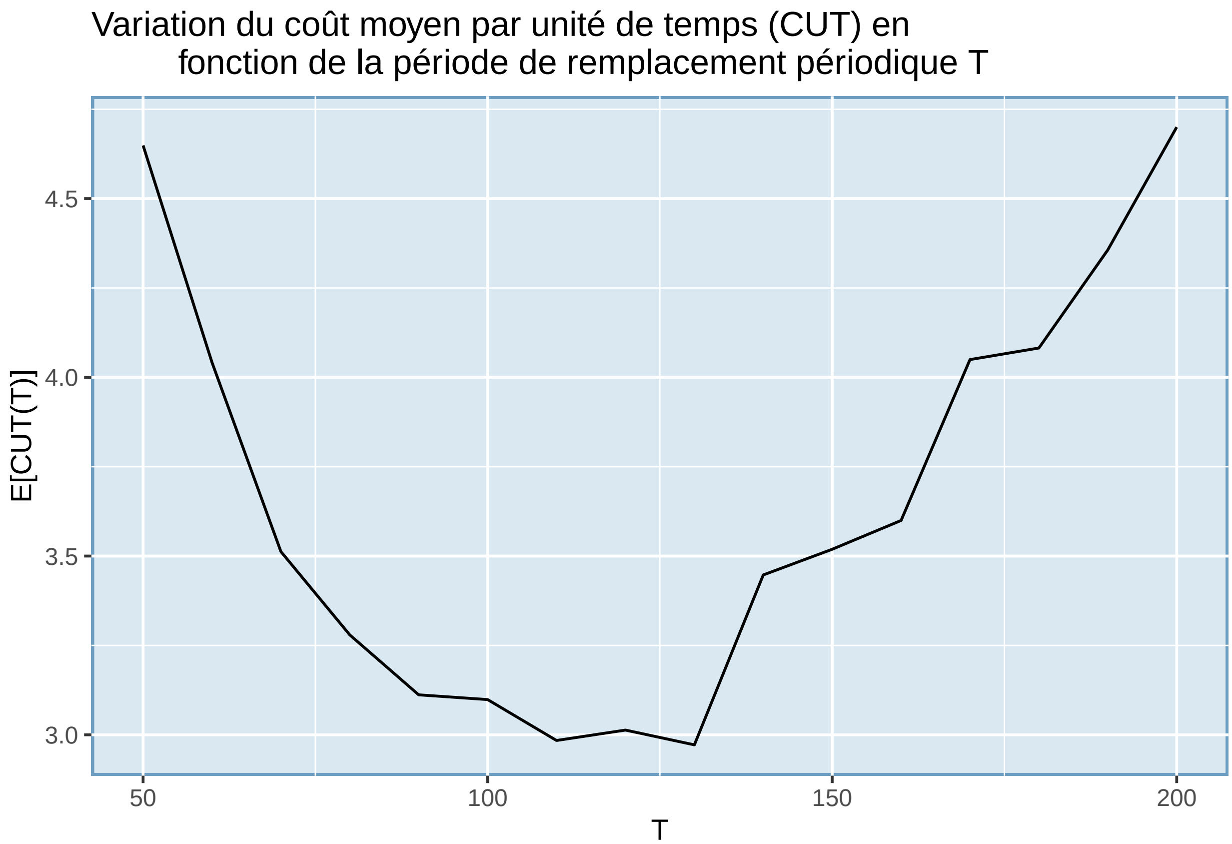


Il semble que le T optimal pour le composant E, notons le T_E^* est proche de $T = 100$.

On va ensuite effectuer un raffinement successif de intervalle étudié jusqu'à trouver une valeur de T_E^*

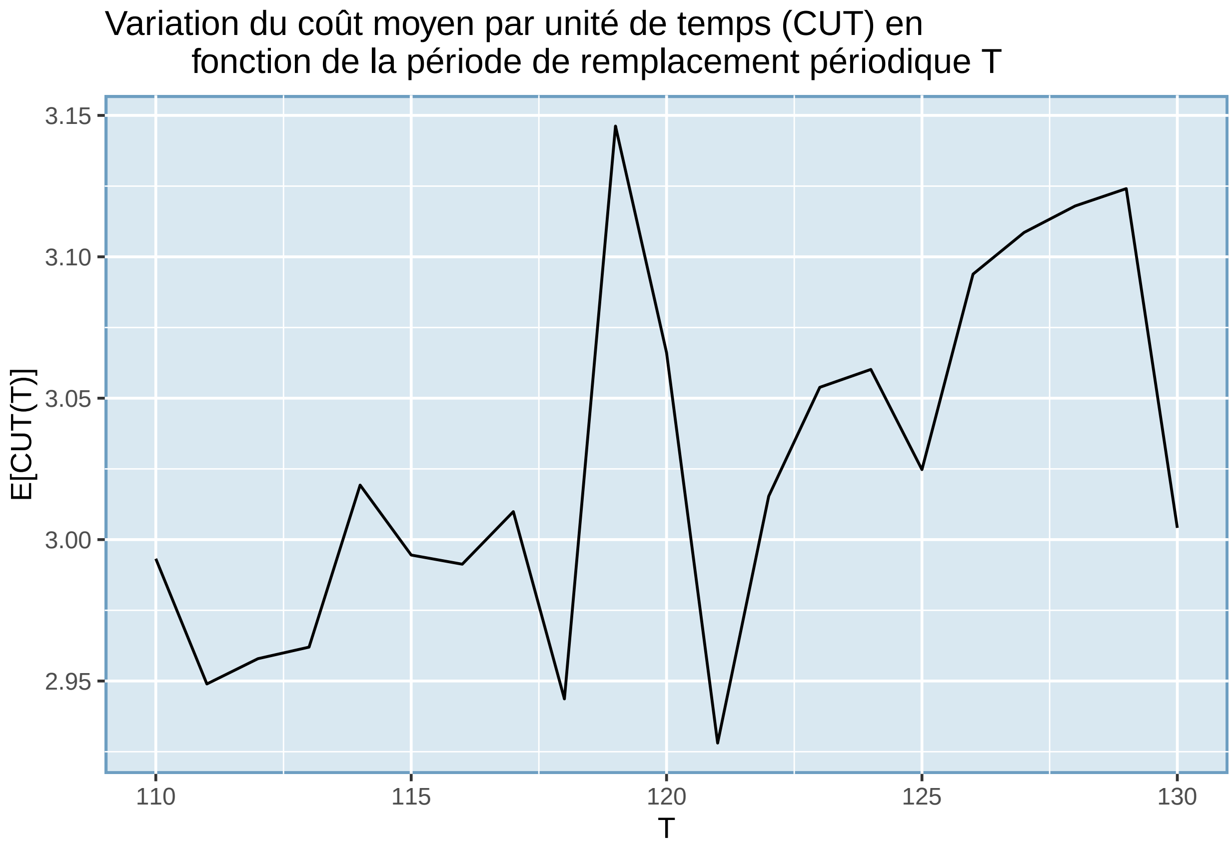
suffisamment précise.

```r
afficher_variation_cut(seq(50,200,10), parametres, p = 5000)
```



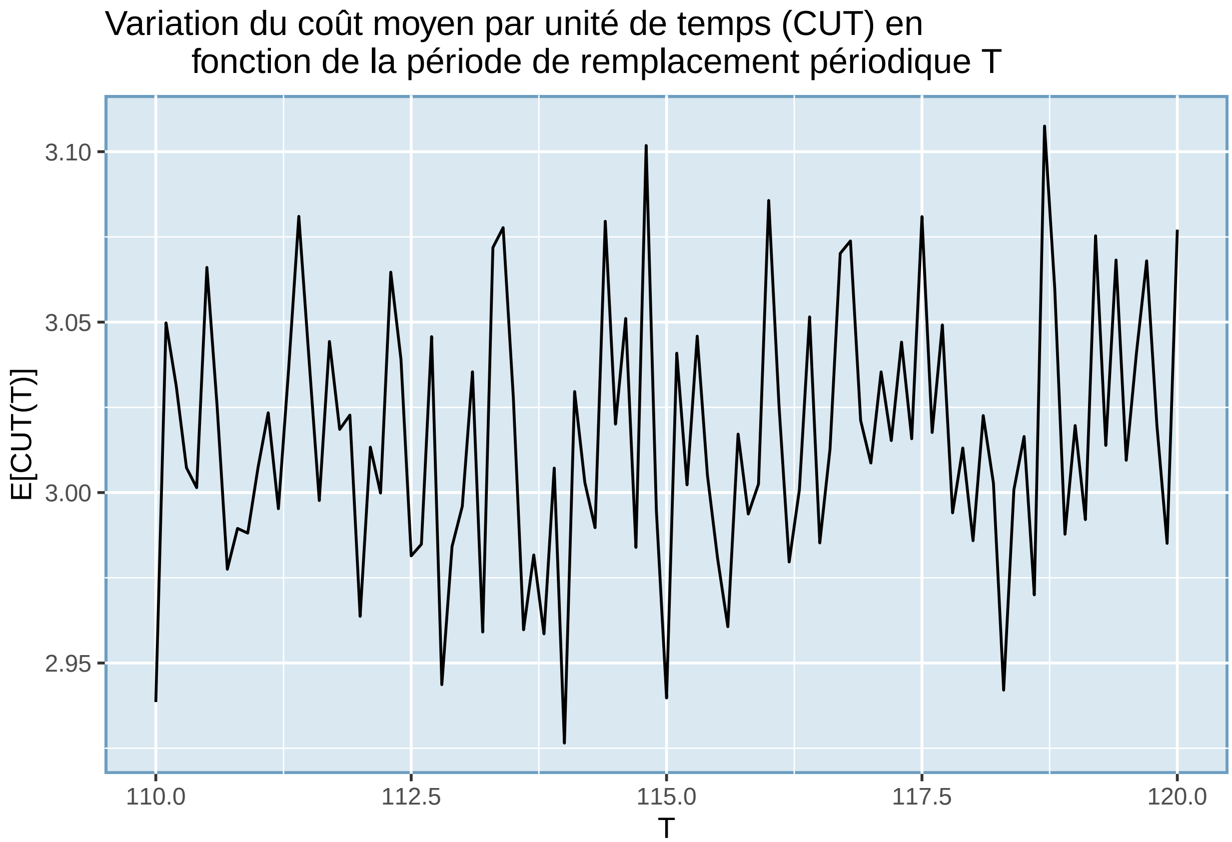
On peut augmenter le nombre de réplication pour réduire l'incertitude liée à la simulation.

```r
afficher_variation_cut(seq(110,130,1), parametres, p = 10000)
```



On peut essayer de doubler le nombre de réplications en réduisant le pas pour T :

```
afficher_variation_cut(seq(110,120,0.1), parametres, p = 20000)
```



En l'état, il semble difficile d'améliorer le résultat.

Il semble donc que la période optimale pour le composant E se trouve entre $T = 110$ et $T = 120$, avec un coût unitaire moyen proche de $3€/h$.

Avec cette politique de maintenance, on va dépenser : - $3€/h$ - $72€/j$ - $504€/semaines$ - $26280€/ans$ environ, avec une période de $T_E^* \simeq [110; 120]$

On peut donc effectuer ce travail de recherche pour tous les composants.

Automatisation de la recherche

Cette méthode graphique prend un temps important. Pour optimiser la recherche, je propose d'écrire un script qui calcul le minimum pour les composants de la classe A.

```
repere <- c("E", "D")
t_opti <- c()
cut_opti <- c()
for (w in repere) {

    # Les paramètres
    alpha = estim_max_likelihood[estim_max_likelihood$Repère == w,]$Alpha_h
    beta = estim_max_likelihood[estim_max_likelihood$Repère == w,]$Beta
    lambda = estim_param_duree_remplacement[
        estim_param_duree_remplacement$Repère == w,
    ]$lambda_h
    tp = 10 / 60 # en heures
    T = 0 # en heures
```

```r
    # Temps de remplacement périodique en heures
    tp = 10 / 60
    # coût de changement fixe
    cc = 30
    # coût d'indisponibilité par heures
    ci = 20 * 60

    # nombre de simulation
    p <- 20000
    # intervalle de recherche du T optimal
    T_set <- seq(50,400,1)
    n <- length(T_set)
    res_list <- c()
    for (t in T_set) {
        sumT <- 0
        sumC <- 0
        for (i in 1:p) {
            Tf <- rweibull(1, shape = beta, scale = alpha)
            D <- rexp(1, rate = lambda)
            if (Tf <= t) {
                # Maintenance corrective
                sumT = sumT + Tf + D
                sumC = sumC + cc + D * ci
            } else {
                # Maintenance préventive
                sumT = sumT + t + tp
                sumC = sumC + cc + tp * ci
            }
        }
        # calcul du coût moyen par unité de temps
        res_list <- append(res_list, sumC/sumT)
    }

    # Calcul de la moyenne mobile
    moyenne_mobile <- c()
    for (i in 3:(n-2)) {
        moyenne_mobile <- c(moyenne_mobile,
                            mean(res_list[seq(i-2,i+2,1)]))
    }

    t_opti <- c(t_opti, T_set[which(moyenne_mobile == min(moyenne_mobile))])
    cut_opti <- c(cut_opti, min(moyenne_mobile))
}
politique_maintenance <- tibble(repere,t_opti,cut_opti)
politique_maintenance
```

```
## # A tibble: 2 x 3
##   repere t_opti cut_opti
##   <chr>   <dbl>    <dbl>
## 1 E         110     2.98
## 2 D          59     8.05
```

Ce script va simuler une valeur de coût moyen par unité de temps pour plusieurs valeurs de T, à partir de p simulations.

Le script calcul ensuite ensuite une moyenne mobile sur ces valeurs, ce qui permet de légèrement lisser sauts aléatoires dûs à la simulation.

Elle sort ensuite la meilleure valeure trouvée pour T^*_ω et le coût moyen associé.

On va donc mettre en place les politiques suivantes :
- Composant E : $T^*_E = 104$ et $\mathbb{E}[CUT(T^*_E)] \simeq 2.980781€.h^{-1}$ contre un coût de $\mathbb{E}[CUT(\infty)] \simeq 7€.h^{-1}$ si l'on ne fait que de la maintenance corrective. Celà représente une diminution de près de 57,12
- Composant D : $T^*_D = 59$ et $\mathbb{E}[CUT(T^*_D)] \simeq 8.116167€.h^{-1}$ contre un coût de $\mathbb{E}[CUT(\infty)] \simeq 13€.h^{-1}$ si l'on ne fait que de la maintenance corrective. Celà représente une division de près de 37.57

Si l'on cumule les coûts de maintenance associé à E et à D, on obtient alors un coût de $11.09695€.h^{-1}$.

Si on cherche à regrouper les maintenances (pour simplifier l'organisation du service ou pour réduire d'éventuels coûts liés à la mobilisation du service de maintenance), on va alors prendre une période de remplacement périodique de $t1 = 60$ (je me permet d'arrondir 59h à 60h pour simplifier la mise en place).

```r
parametres = list(
  alpha = estim_max_likelihood[estim_max_likelihood$Repère == "E",]$Alpha_h,
  beta = estim_max_likelihood[estim_max_likelihood$Repère == "E",]$Beta,
  lambda = estim_param_duree_remplacement[
    estim_param_duree_remplacement$Repère == "E",
  ]$lambda_h,
  tp = 10 / 60, # en heures
  T = 60, # en heures
  # Temps de remplacement périodique en heures
  tp = 10 / 60,
  # coût de changement fixe
  cc = 30,
  # coût d'indisponibilité par heures
  ci = 20 * 60
)
a <- simulation_esp_cut(param = parametres, p = 10000)
parametres = list(
  alpha = estim_max_likelihood[estim_max_likelihood$Repère == "D",]$Alpha_h,
  beta = estim_max_likelihood[estim_max_likelihood$Repère == "D",]$Beta,
  lambda = estim_param_duree_remplacement[
    estim_param_duree_remplacement$Repère == "D",
  ]$lambda_h,
  tp = 10 / 60, # en heures
  T = 60, # en heures
  # Temps de remplacement périodique en heures
  tp = 10 / 60,
  # coût de changement fixe
  cc = 30,
  # coût d'indisponibilité par heures
  ci = 20 * 60
)
b <- simulation_esp_cut(param = parametres, p = 10000)
```

```r
print(paste("Coût maintenance groupée à T = 60 : ", round(a + b, 4), "€/h"))
```

```
## [1] "Coût maintenance groupée à T = 60 :  12.0911 €/h"
```

On a donc une légère augmentation du coût moyen par unité de temps, mais qui simplifie grandement la gestion de la politique de maintenance basée sur l'age groupée

Je propose donc à l'entreprise de mettre en place une maintenance

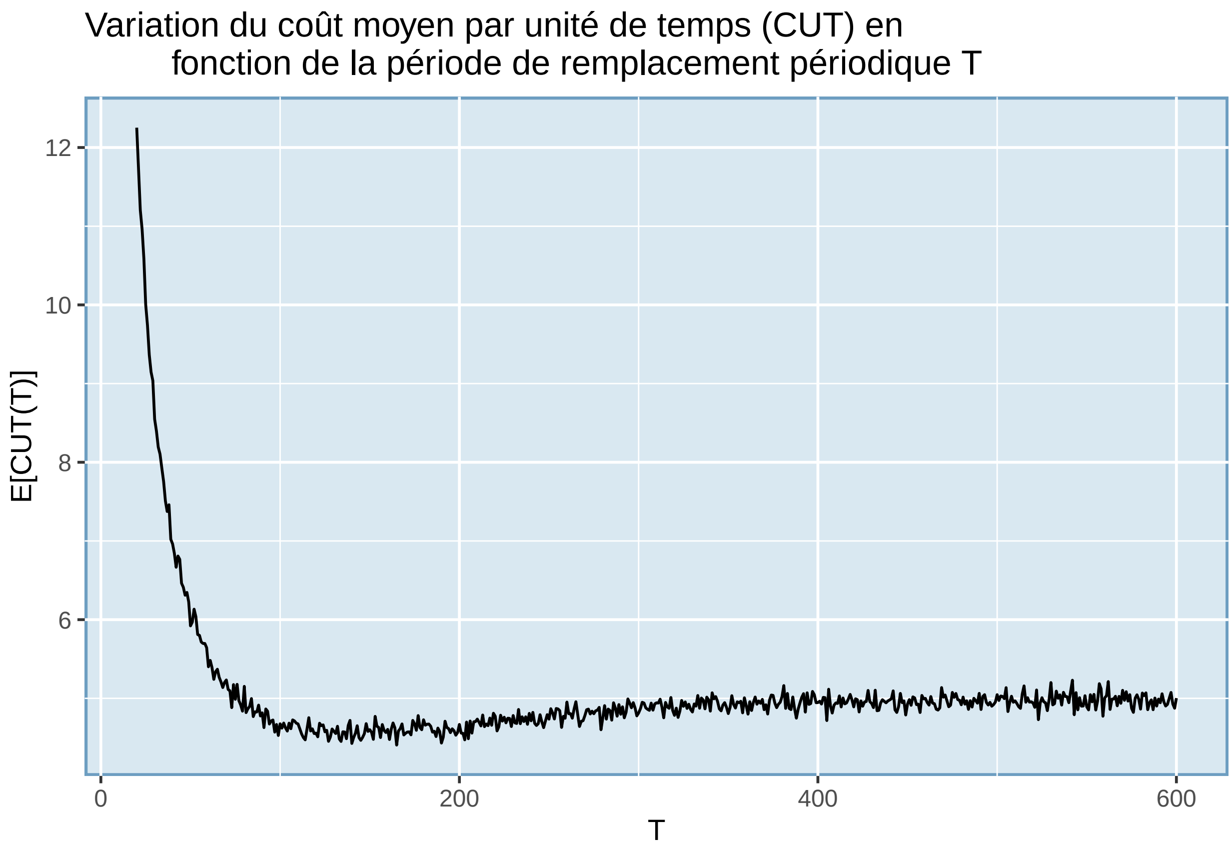
Optimisation de la politique de maintenance pour les éléments de la classe B, méthode ABAC et ABAD

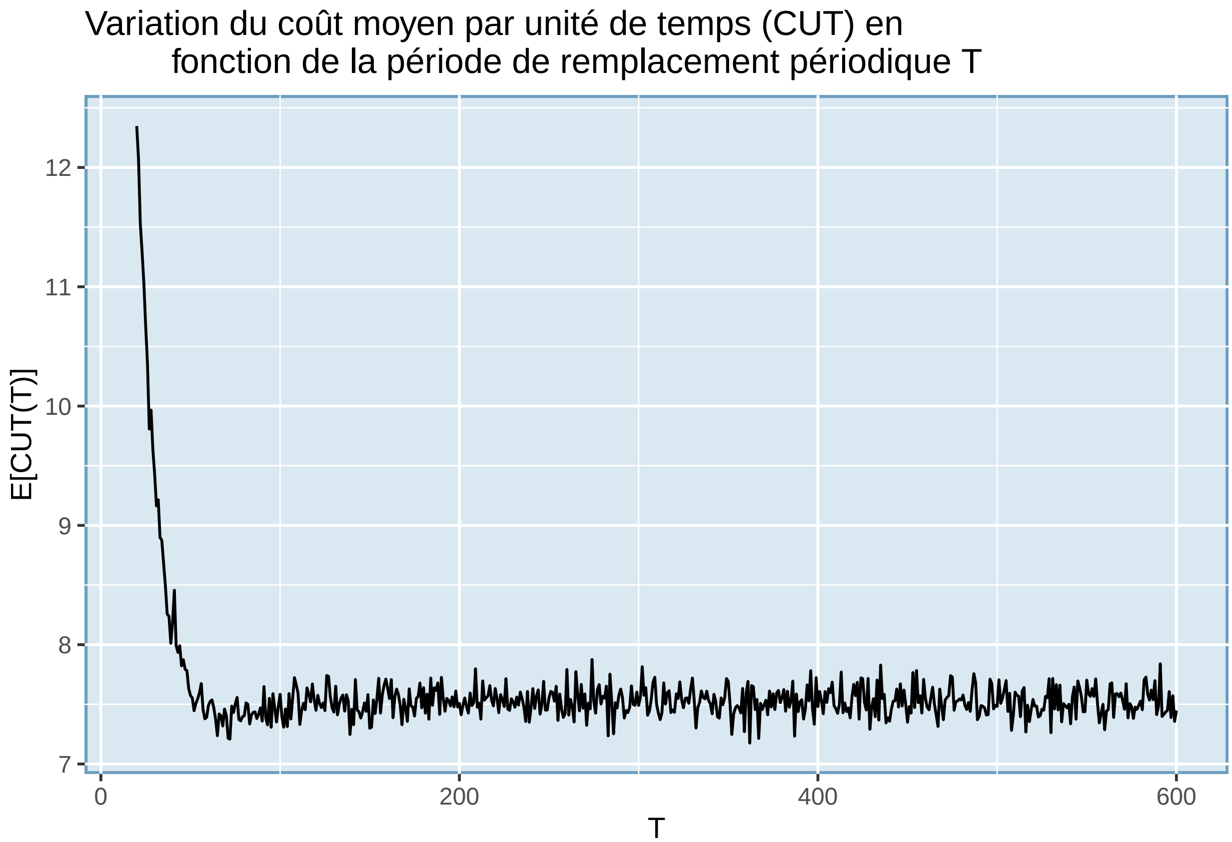
Nous avons défini une période d'inspection $t1 = 60h$ pour les éléments de classe A.

Les éléments de la classe B seront alors optimisé à partir d'une politique similaire avec $t2 = 2t1 = 120$.

On va comparer cette politique avec : - une politique basée uniquement sur les maintenances correctives - une politique optimale pour chaque élément

Coût d'une politique uniquement corrective





Pour le composant "A", on peut observer un minimum autour de $110 \sim 120$. Pour le composant "F" en revanche, il semble que la politique préventive à un effet limité. On pourrait presque conseiller de ne pas l'utiliser et de préférer une politique de maintenance corrective.

Pour le composant A, une politique uniquement corrective semble mener à un coût d'environ 5€.h^{-1}.

Pour le composant F, une politique uniquement corrective semble mener à un coût d'environ 7.5€.h^{-1}.

Coût optimal par composant pour A et F

```
## # A tibble: 2 x 3
##   reperes t_opti cut_opti
##   <chr>    <dbl>    <dbl>
## 1 A          149     4.52
## 2 F           60     7.36
```

On observe dans ce cas :

- $T_A^* = 134$ et $\mathbb{E}[CUT(T_A^*)] = 4.534623€.h^{-1}$
- $T_F^* = 134$ et $\mathbb{E}[CUT(T_F^*)] = 7.356775€.h^{-1}$

Cela confirme que pour le composant F, par rapport à une politique uniquement corrective, l'amélioration est faible.

Politique ABAC ABAD avec $t2 = 2t1$

Dans ce cas on prend une politique groupée pour A et F avec

```r
print(paste("Coût moyen par heure pour t2 = 120 pour le composant A :", a))
```

```
## [1] "Coût moyen par heure pour t2 = 120 pour le composant A : 4.58123500005969"
```

```r
print(paste("Coût moyen par heure pour t2 = 120 pour le composant F :", b))
```

```
## [1] "Coût moyen par heure pour t2 = 120 pour le composant F : 7.49902305034819"
```

```r
print(paste("Coût maintenance groupée à t2 = 120 : ", round(a + b, 4), "€/h"))
```

```
## [1] "Coût maintenance groupée à t2 = 120 :  12.0803 €/h"
```

Avec cette méthode, on remarque que les composants de la classe A et B ont finalement le même coût de maintenance par heure, ce qui montre que l'optimisation est efficace pour la classe A.

Conclusion

Pour une politique corrective uniquement, comme celle appliquée actuellement par l'entreprise on a :
- pour E : 7€.h^{-1}
- pour D : 13€.h^{-1}
- pour A : 5€.h^{-1}
- pour F : 7.5€.h^{-1}

Soit un coût total d'environ : 32.5€.h^{-1} soit environ 283920€/ans.
Si l'on applique les deux politiques groupées pour la classe A et B avec $t1 = 60h$ et $t2 = 2t1 = 120h$, on a : 24.2558€.h^{-1} soit environ 211898.7€/ans.
On a donc d'une amélioration de 25.37 %.

Je conseille donc à l'entreprise de mettre en place cette politique de maintenance.

Attention cependant, les résultats proposés reposent sur une analyse obtenue avec très peu de données.