Projet Séries chronologiques

Université de Paris Nanterre

Anissa Goulif, Radwan Sarmini et Sirine Gabriel

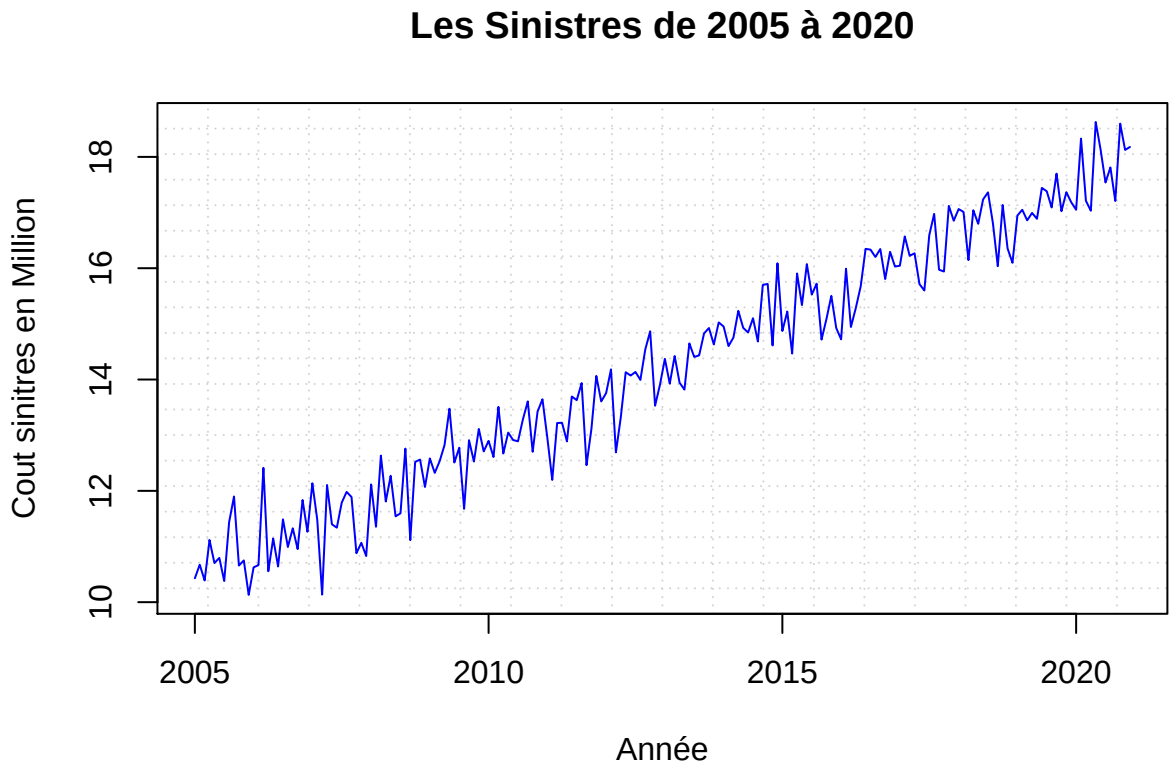
March 11, 2021

Contents

1 Sinistres

1.1 Question 1

```
#setwd("~/School/IESFAR NANTERRE/Données TP - Sinistres - Primes")
load("Sinistres.Rdata")
plot(Sinistres,
     type = 'l',col='blue',
     main="Les Sinistres de 2005 à 2020"
     ,ylab="Cout sinitres en Million",xlab="Année",
     panel.first = grid(20))
```



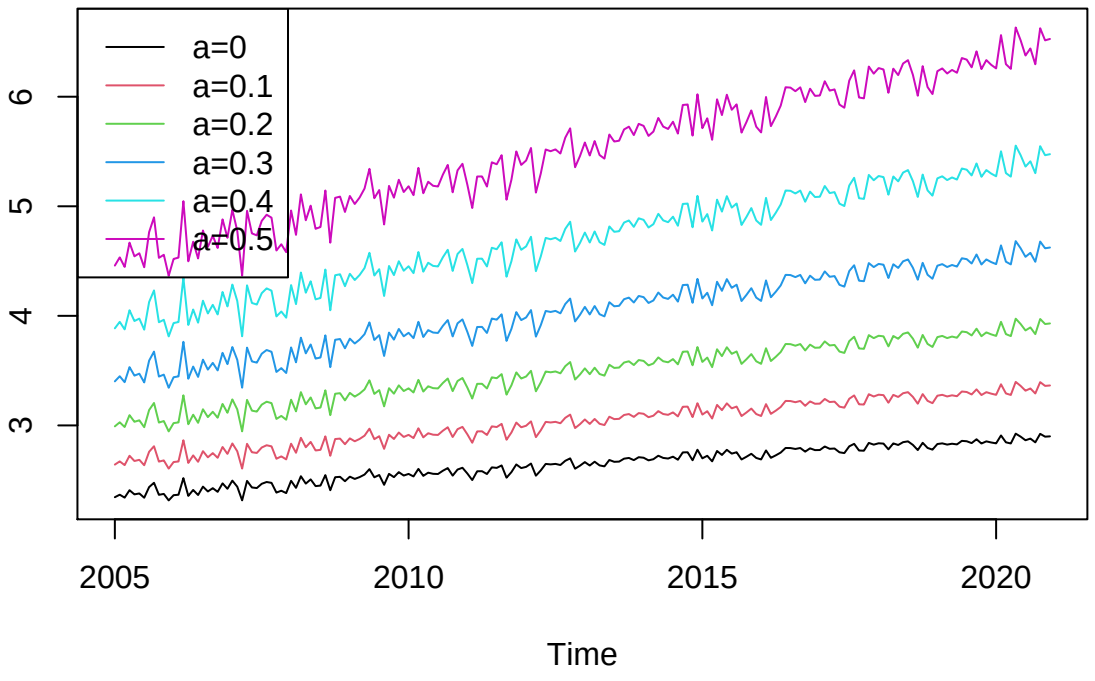
On remarque une tendance, mais pas de saisonnalité, le modele semble additif (amplitude 'constante', en effet la composante tendancielle et le bruit semblent rester constante au cours du temps) et donc pertinent la prediction sera plus simple.

1.2 Question 2

Pour cette question, nous avons crée une fonction qui prends en argument une série chronologique "x" un parametre "a" et retourner la série chronologique transformée. Nous avons ensuite essayer quelques transformations pour se rapprocher d'un modèle additif.

```
source("boxcox.r")
a = 0.2
```

```r
Sinistres02 = boxcox(Sinistres,a)

b = 0.5
Sinistres05 = boxcox(Sinistres,b)

A = seq(0,0.5,0.1)
n = length(Sinistres)
M = matrix(0,n,length(A))

for (i in (1:length(A))){
  M[,i] = boxcox(Sinistres,A[i])
}

M = ts(M,start=start(Sinistres),frequency=frequency(Sinistres))

#visualisation de toutes les transformations
ts.plot(M,col=(1:length(A)))
legend('topleft',legend=paste("a=",as.character(A),sep=""),col=(1:length(A)),lty=1)
```

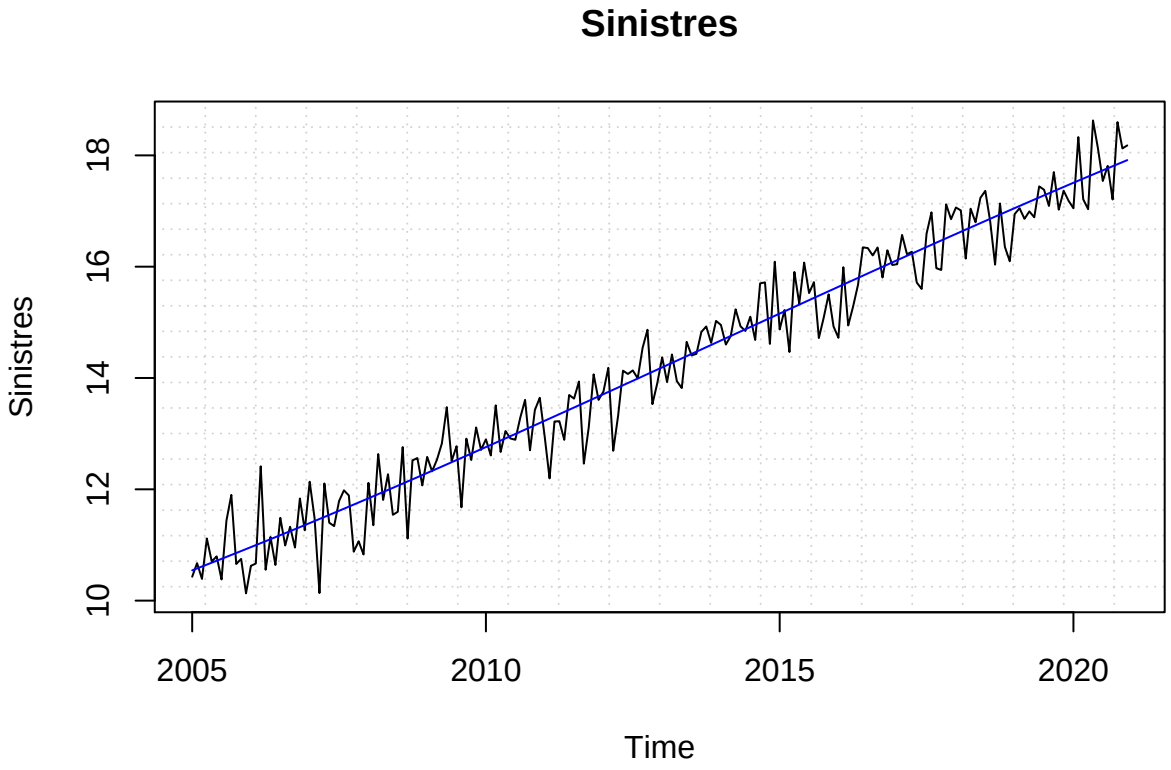


Nous avons voulu faire une transformation additive mais il n'y a pas eu de réel changement observé. Il n'y a pas de difference entre BOXCOX a=0.1 et a=0.9. Nous ne gardons pas la transformation, et nous travaillerons sur la série telle quelle.

Nous avons ensuite effectué l'ajustement de la tendance polynomiale et visualiser la série avec cette tendance ajustée :

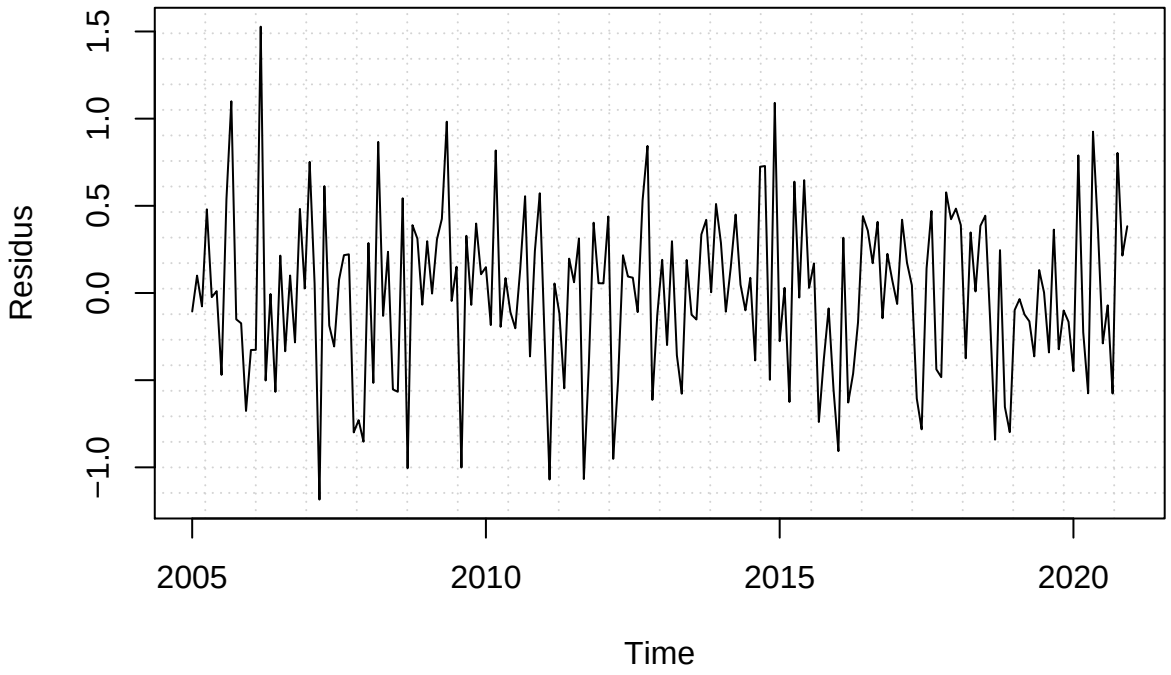
```r
lm.out2 = lm(Sinistres ~ I(time(Sinistres)) + I(time(Sinistres)^2) + I(time(Sinistres)^6), data
```

```r
plot(Sinistres,main="Sinistres",panel.first = grid(20))
lines(as.vector(time(Sinistres)),lm.out2$fitted.values,col="blue")
```

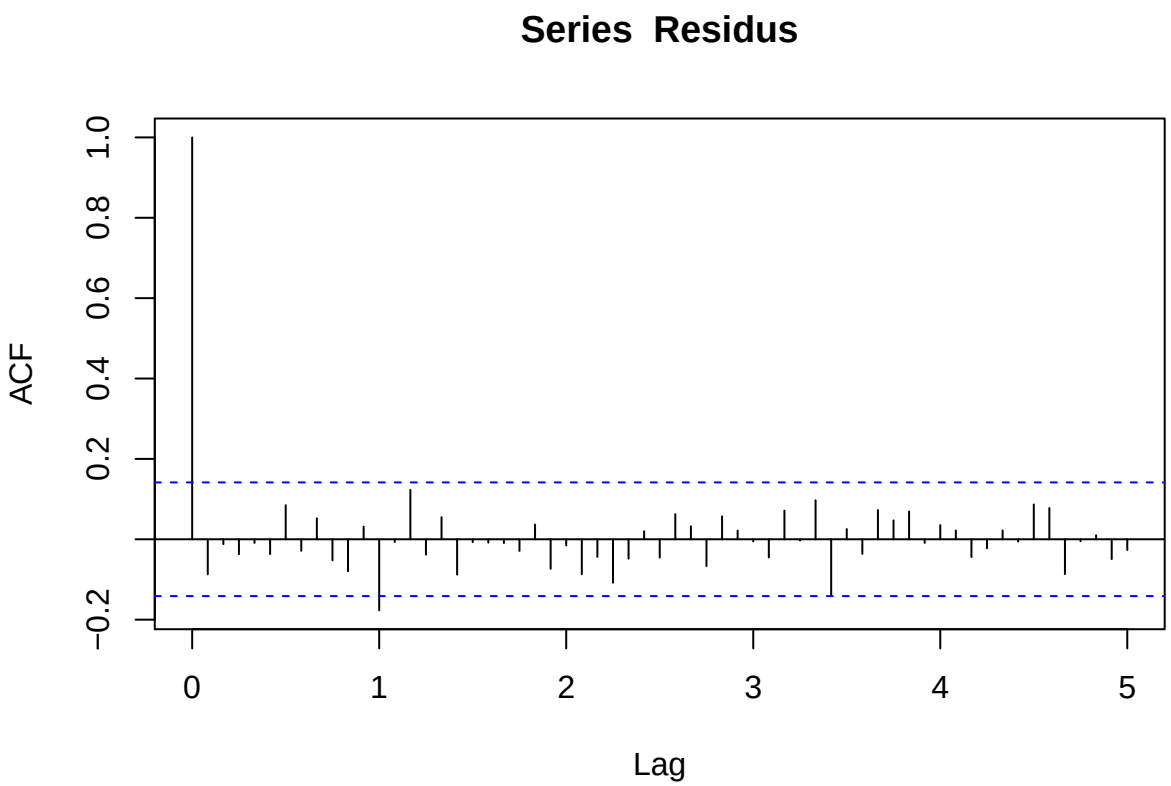


1.3 Question 3

```r
SinistresDetend = Sinistres-lm.out2$fitted.values

source("SaisEst.r")

p       = 12
List    = SaisEst(SinistresDetend,p)
motif   = List$motif
SEst    = List$serie

Residus = SinistresDetend-SaisEst(SinistresDetend,p)$serie

par(mfrow=c(1,1))
plot(Residus,panel.first = grid(20))
```

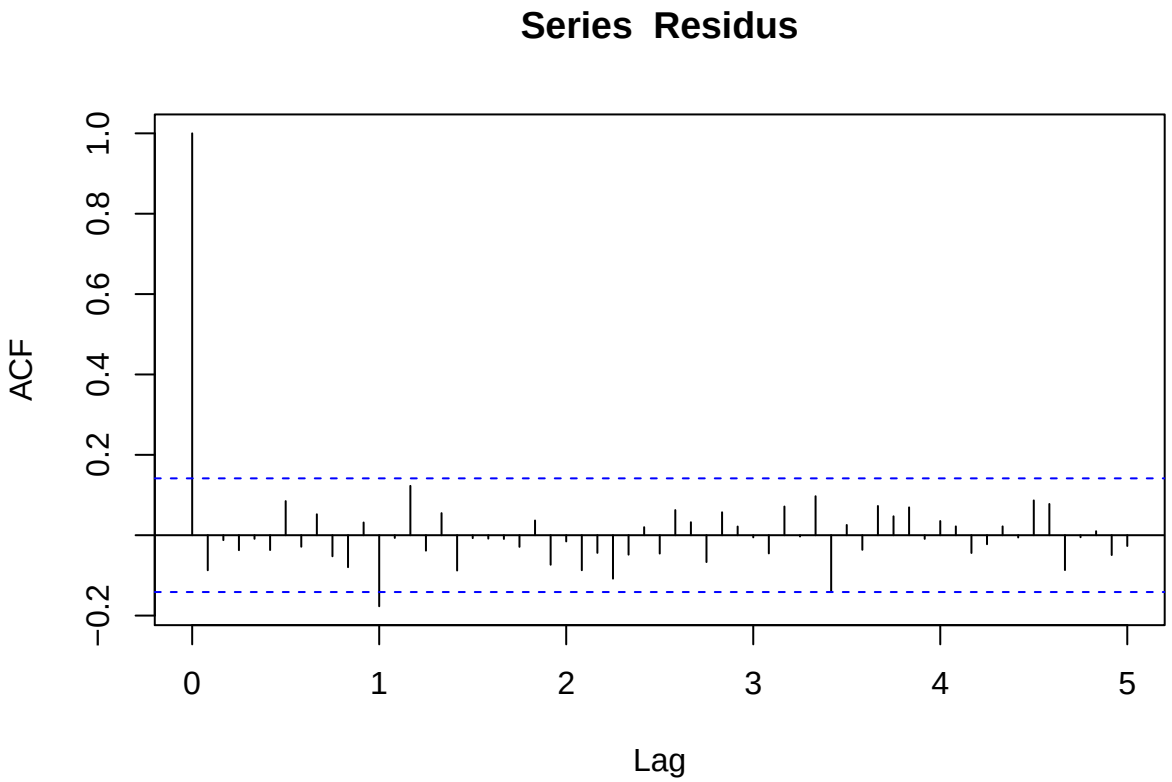



```
acf(Residus,lag = 60)
```



Pour cette question, nous avons d'abord crée "SinistresDetend" qui contient les résidus de la série. En effet, nous avons supposé qu'il n'y avait pas de saisonnalité. Nous avons ensuite utilisé la fonction "saisest". En effet, cette fonction permet d'extraire la saisonnalité de la série. Nous sommes passés par cette étape pour être sur de ne pas nous tromper sur nos suppositions.

```r
p       = 12
List    = SaisEst(SinistresDetend,p)
motif   = List$motif
SEst    = List$serie

Residus = SinistresDetend-SaisEst(SinistresDetend,p)$serie

par(mfrow=c(1,1))

acf(Residus,lag = 60)
```



On remarque ici que plus le lag est grand plus il y a de la corrélation.

```r
h = 50
Box.test(Residus,lag=h,type ="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  Residus
## X-squared = 42.585, df = 50, p-value = 0.7624
```

On remarque ici que la p-value est supérieure à 5% : On ne rejette pas l'hypothese d'indépendance des résidus.

1.4 Question 4

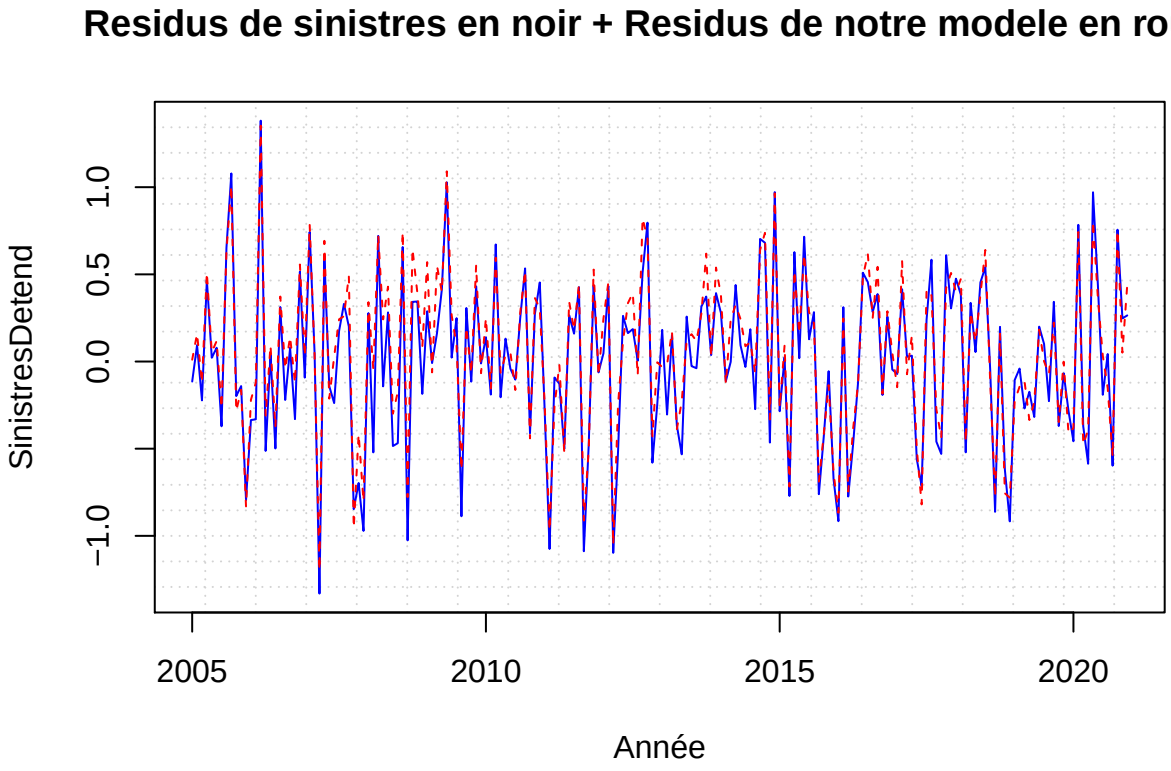
Afin de répondre à cette question, nous avons choisis d'utiliser l'arima et pour se faire nous devions choisir les ordres. Nous avons tout d'abord choisis de crée une fonction "min_aic". En effet, la clé principale pour bien modéliser est L'AIC et que plus l'AIC est petit plus le modèle sera bon. Alors on doit minimiser l'AIC dans un modèle pour choisir le bon ordre de q et d où d est l'opérateur différence. Bien sur, on peut choisir q à partir de son autocorrélogramme alors c'est bien lié à notre observation et notre goût statistique, cependant nous avons laissé cette méthode de modélisation pour la section suivante autorégressif. Enfin, pour minimiser l'AIC on peut créer deux boucles pour d et q et laisser le code tourner pour tester plusieurs ordres jusqu'à ce qu'elle trouve le plus petit AIC.

```
min_aic<-function(X,P,D,Q) { #X = donnes, P = ordre de la autoregressif, D = differention rend
  min_p=P[1] #vecteur pour autoregressif
  min_d=D[1] #vecteur pour diff
  min_q=Q[1] #vecteur pour moyenne mobile
  cpt=0
  for(d in D[1]:D[2]) {
    for(p in P[1]:P[2]) {
      for(q in Q[1]:Q[2] ) {
        tryCatch( { #eviter l'errure
          Mod<- arima(X,order = c(p,d,q),method='ML') #Modiliser avec un combinaision
          Mod____<<-Mod
          aic=Mod$aic  #stocker le AIC
        },error=function(e){})
        if(cpt==0){ #essyer la premire combinasion
          aic_min=aic # et stocker le premier AIC
          print(aic_min)
          print(sprintf
                ('Tour %s et les orders sont : %s , %s , %s et aic est %1f :'
                  ,cpt,min_p,min_d,min_q,aic_min))}
        else{
          if(aic<aic_min){ #si la nouvelle combinasion a un AIC plus petite que la precednt
            aic_min=aic #alors remblase L'AIC
            min_p=p # et remblase p
            min_d=d # et remblase d
            min_q=q # et remblase q
            print(sprintf
                  ('Tour %s et les orders sont : %s , %s , %s et aic est %1f :'
                    ,cpt,min_p,min_d,min_q,aic_min))
          }}
        cpt = cpt+1
      }
    }
    print("Stil running")
  }
  aic_<<-aic_min
  print("Done")
```

```
  return(c(min_p,min_d,min_q))
}
```

D'apres L'AIC on trouve que les ordre 4 1 4 sont les meilleurs. On fait maintenant tourné l'arima et on vérifie si les résidus prédits correspondent à nos résidus originaux.

```
model_sini <-arima(Sinistres,order = c(4,1,4),method='ML')

plot(SinistresDetend,
     type = 'l',col='blue',
     main="Residus de sinistres en noir + Residus de notre modele en rouge"
     ,xlab="Année",
     panel.first = grid(20))
lines(model_sini$residuals,col='red',lty = 2)
```

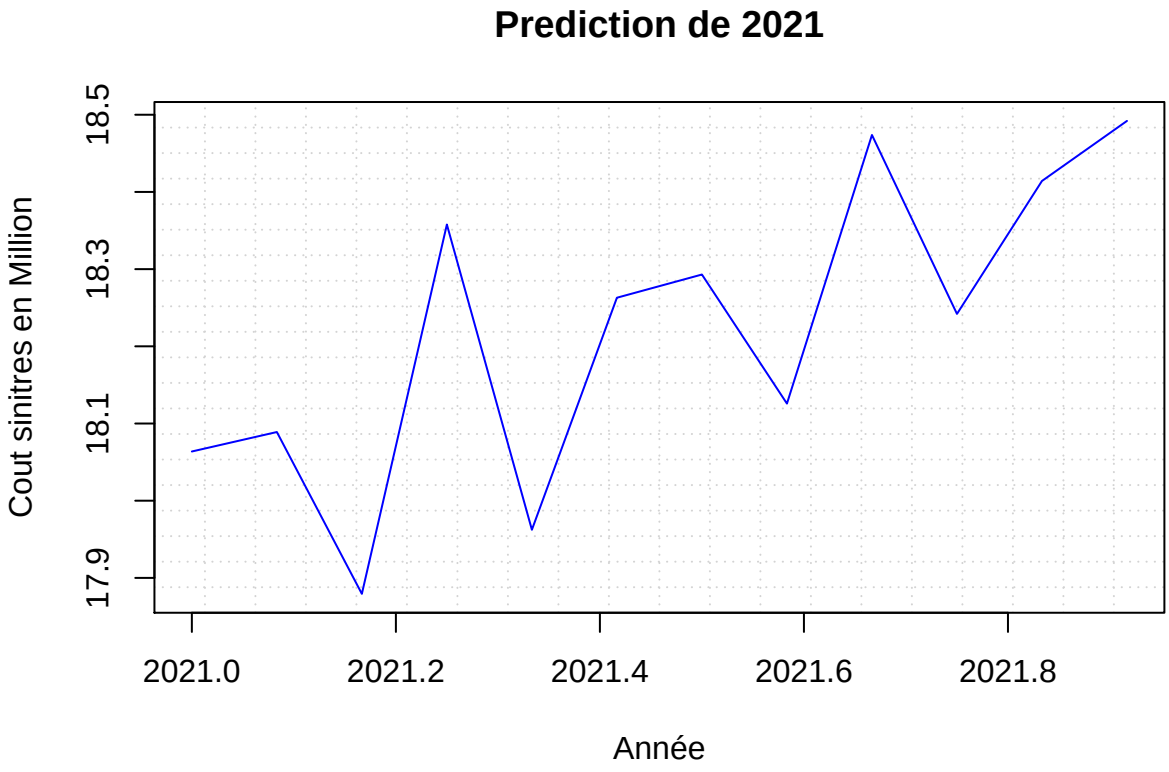


Ici, on voit les residus de sinistres en noir ainsi que les residus de notre modèle en rouge. Notre conclusion est donc que c'est presque proche des residus originaux.

Passons maintenant à la prédiction.

```
pred_sini<-predict(model_sini,12) #utilisation de notre modèle

sini_pred <- pred_sini$pred

plot(sini_pred,
     type = 'l',col='blue',
```

```
     main="Prediction de 2021"
     ,ylab="Cout sinitres en Million",xlab="Année",
     panel.first = grid(20))
```

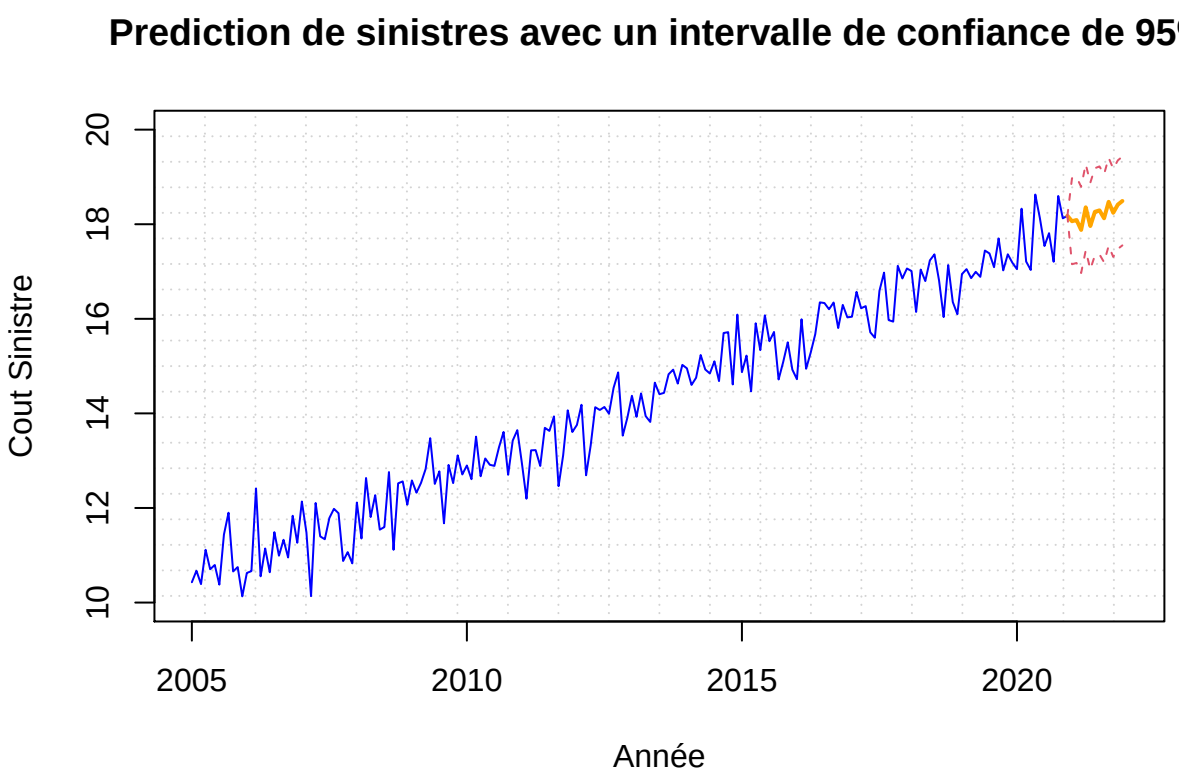


1.5 Question 5

Afin de répresenter le graphe final, nous avons crée/ressorti plusieurs éléments : l'intervalle de variation à 95%, les 12 dernières valeurs de Sinistres, les valeur predites, les valeurs predites supérieures, les valeurs predites inférieures.

```
#intervalle de variations à 95% :
sini_se <- pred_sini$se

#les 12 dernières valeurs de Sinistres :
Sini_last = ts(tail(Sinistres,12),end=end(Sinistres),frequency = frequency(Sinistres))

#valeur predites :
sini_val<-ts(c(tail(Sinistres,1),sini_pred),
             end =end(sini_pred),frequency = frequency(sini_pred))

#valeur predites supérieures :
sini_up<-ts(c(tail(Sinistres,1),sini_pred + qnorm(0.975)*sini_se),
            end =end(sini_pred),frequency = frequency(sini_pred))
```

```r
#valeur predites inférieures :
sini_down<-ts(c(tail(Sinistres,1),sini_pred -qnorm(0.975)*sini_se),
              end =end(sini_pred),frequency = frequency(sini_pred))

# Graphe :
plot(Sinistres,
     type = 'l',col='blue',
     main="Prediction de sinistres avec un intervalle de confiance de 95%"
     ,ylab="Cout Sinistre",xlab="Année",
     panel.first = grid(20),
    xlim=c(start(Sinistres)[1],end(Sinistres)[1]+2),
    ylim=c(10,20))
lines(sini_val,type = 'l',col='orange',lty=1,lwd=2)
points(sini_up, type = "l", col = 2, lty = 2)
points(sini_down, type = "l", col = 2, lty = 2)
```



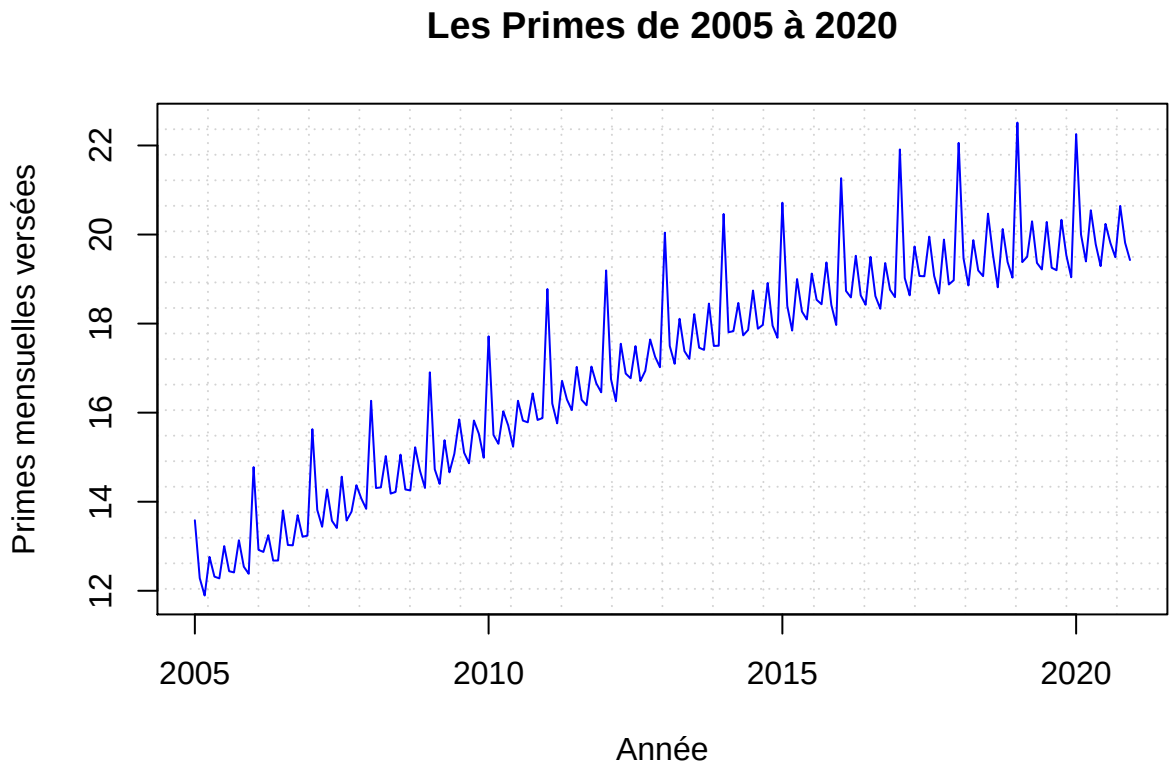
Enfin, nous obtenons ce graphe représentant les sinistres ainsi que leurs prédictions grace à un plot où nous "plaçons" les éléments créé précédemment.

2 Primes

2.1 Question 1

```r
#total des primes mensuelles versés par les assurés.
load("Primes.Rdata")
#Question 1
#Primes
#str(Primes)
#visualisation
#plot(Primes)
#time(Primes)
#12 mois de 2005 à 2020, pareil que sinistres (logique)
```

```r
#meilleure visualisation avec ggplot
plot(Primes,
     type = 'l',col='blue',
     main="Les Primes de 2005 à 2020"
     ,ylab="Primes mensuelles versées",xlab="Année",
     panel.first = grid(20))
```

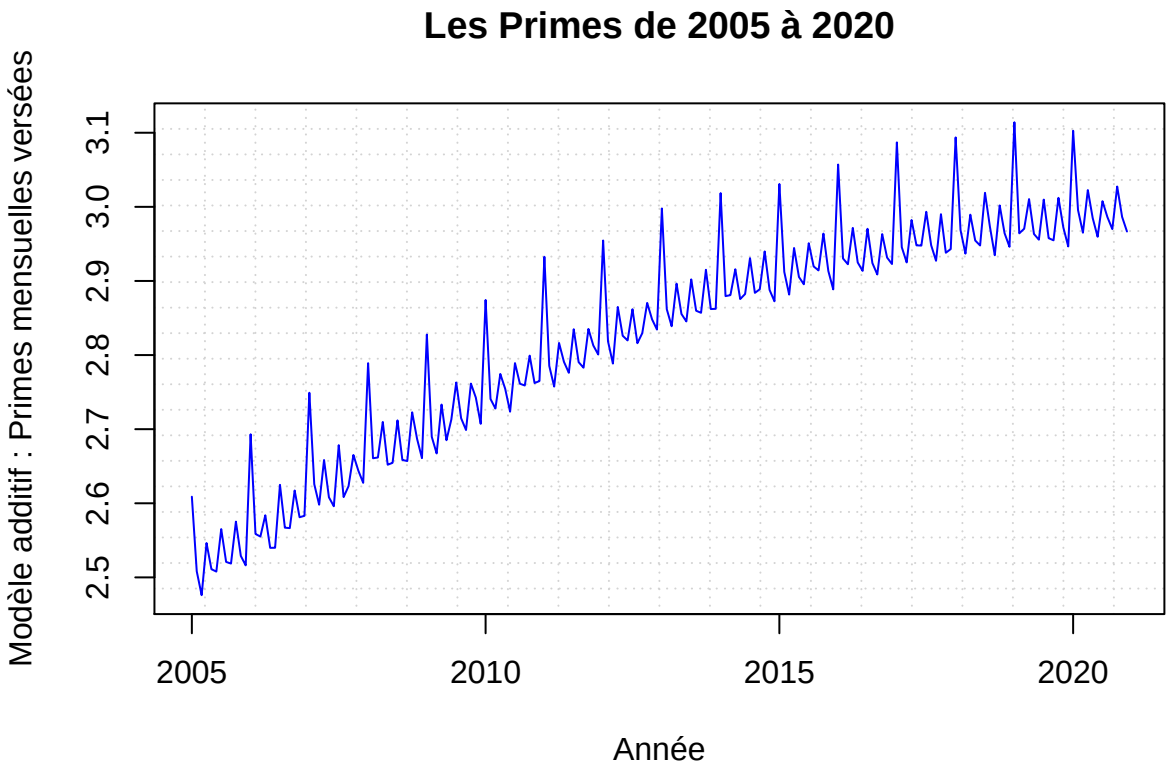


On remarque une meilleure visualisatiob avec “ggplot”. En effet, on constate qu’il y a une tendance, une saisonnalité de période 1 an (même motif se repetant). Cependant, il n’y a pas la même amplitude, elle varie entre 2005 et 2020 (contrairement à ce qu’on a constaté pour les sinistres). Le modèle ne semble donc pas être un modèle additif.

2.2 Question 2

On rend le modèle additif avec la fonction log. Remarque : on aurait pu utiliser la fonction boxcox avec a = 0. Enfin, on visualise la transformation.

```
logPrimes=log(Primes)
#Visualisation de la transformation
plot(logPrimes,
     type = 'l',col='blue',
     main="Les Primes de 2005 à 2020"
     ,ylab="Modèle additif : Primes mensuelles versées",xlab="Année",
     panel.first = grid(20))
```



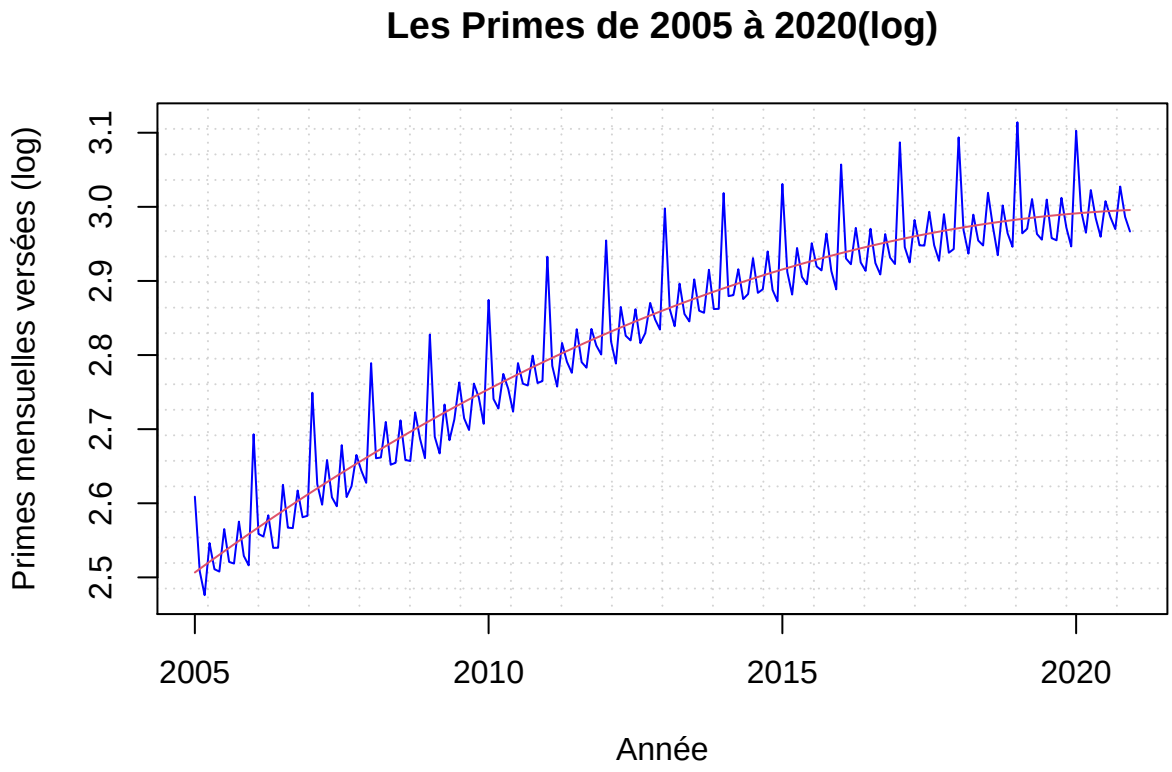
Il semble donc ici y avoir la meme amplitude en 2005 comme en 2020. Le modèle additif sera donc pertinent notamment pour les prédictions.

2.3 Question 3

Ajustons maintenant la tendance polynomiale avec un polynome de degres 2. Nous choisissons le degré 2 car la repartition des données semble se rapprocher d'un polynome de degre 2.

```
T = time(logPrimes)
lm.primes = lm(logPrimes ~I(T) + I(T^2), data = logPrimes)
```

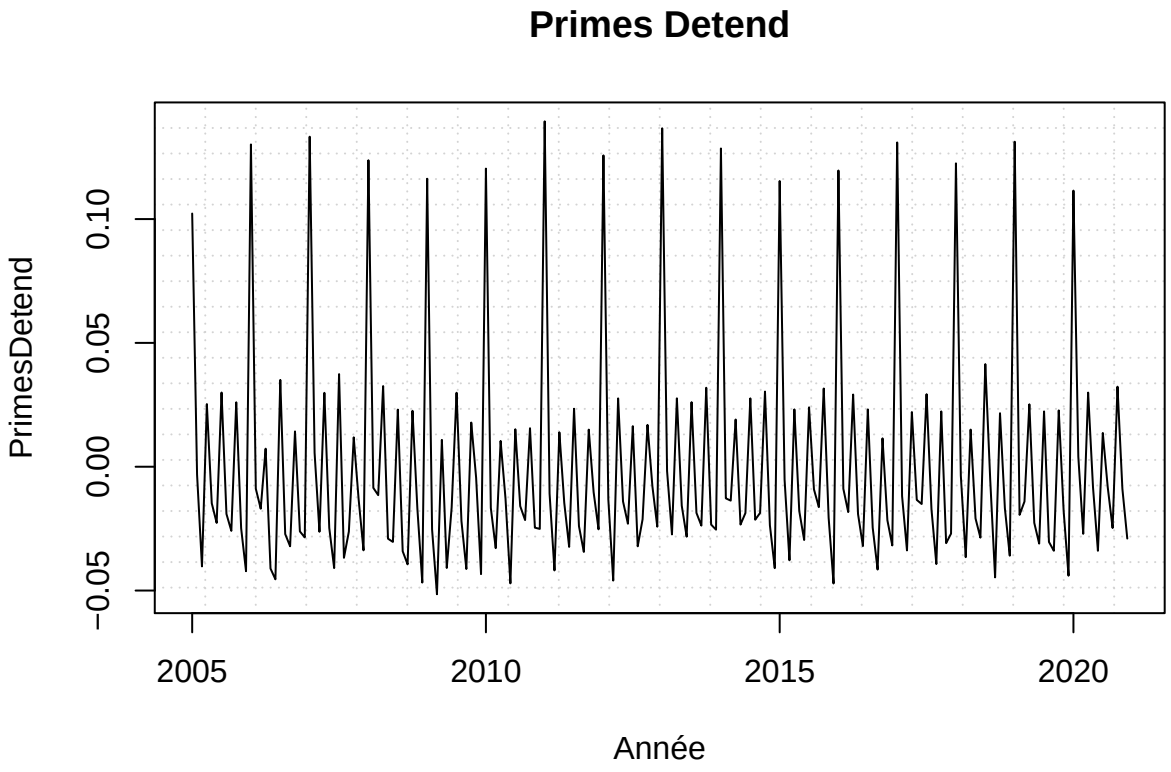
```r
plot(logPrimes,
     type = 'l',col='blue',
     main="Les Primes de 2005 à 2020(log)"
     ,ylab="Primes mensuelles versées (log)",xlab="Année",
     panel.first = grid(20))
lines( ts(lm.primes$fitted.values,frequency = frequency(logPrimes),start=start(logPrimes)),col=
```



Nous visualisons donc logPrimes ainsi que la tendance ajustée. L'ajustement semble correct.

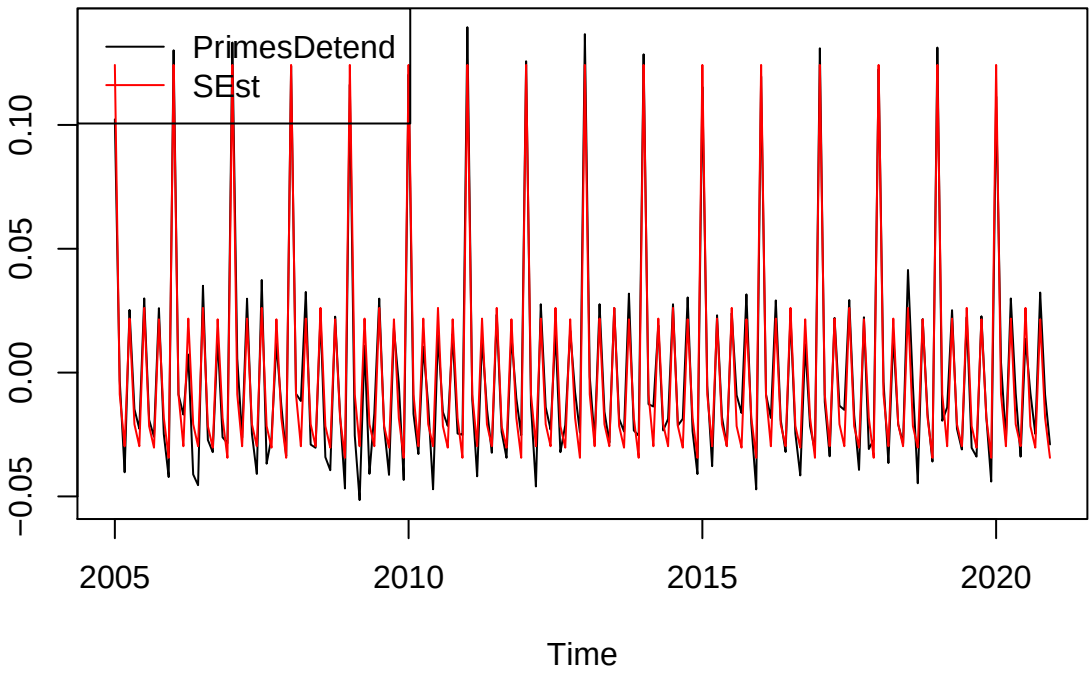
2.4 Question 4

```r
PrimesDetend = logPrimes-lm.primes$fitted.values

#visualisation de logprimes sans la tendance estimée
plot(PrimesDetend,
     type = 'l',col='black',
     main="Primes Detend"
     ,xlab="Année",
     panel.first = grid(20))
```




```r
p= 12 #car la saisonnalité est de période 1an soit 12 mois
saison= SaisEst(PrimesDetend,p)
motif   = saison$motif
SEst    = saison$serie

#Visualisation
plot(motif,type="l",main ="motif")
```

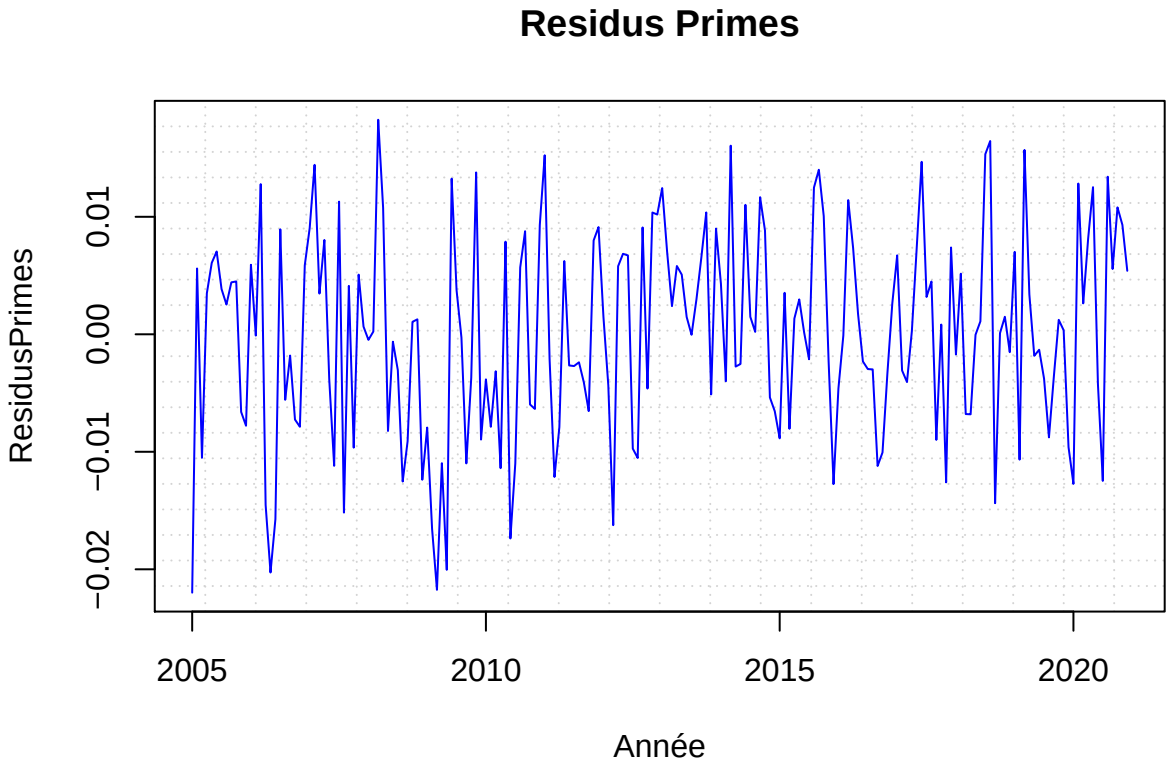

```
ts.plot(PrimesDetend,SEst,col=c("black","red"))
legend('topleft',legend=c("PrimesDetend","SEst"),col=c("black","red"),lty=1)
```



2.5 Question 5

On crée ici "ResidusPrimes", les residus comme son nom l'indique. Pour se faire on reprend notre logPrimes (sans la tendance) dans lequel on enlève la saisonnalité (il ne reste que les résidus). Ensuite, nous visualisons à l'aide d'un ggplot ces résidus :

```
ResidusPrimes = PrimesDetend-SEst


plot(ResidusPrimes,
     type = 'l',col='blue',
     main="Residus Primes"
     ,xlab="Année",
     panel.first = grid(20))
```




```
#autocorrelation : correlogramme
#acf(ResidusPrimes,lag = 10)
#acf(ResidusPrimes,lag = 20)
#acf(ResidusPrimes,lag = 30)
```

Nous réalisons ensuite un test de non stationnarité de ces résidus (nous avons utliser un test de kpss). L'hypothèse H0 de ce test est la stationnarité des résidus.

```
library(tseries)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
kpss.test(ResidusPrimes)
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  ResidusPrimes
## KPSS Level = 0.4238, Truncation lag parameter = 4, p-value = 0.0669
```

On remarque que la p-value est égale à $0.0669 > 0.05$: on ne rejette pas l'hypothèse de stationnarité au seuil 5%, on peut dire que les residus sont stationnaires.

2.6 Question 6

Pour cette question nous réalisons un box.test. Pour bien choisir le lag, nous avons effectué à part quelques acf pour nous permettre de choisir au mieux le lag.

```
h = 20
Box.test(ResidusPrimes,lag=h,type ="Ljung-Box")
```

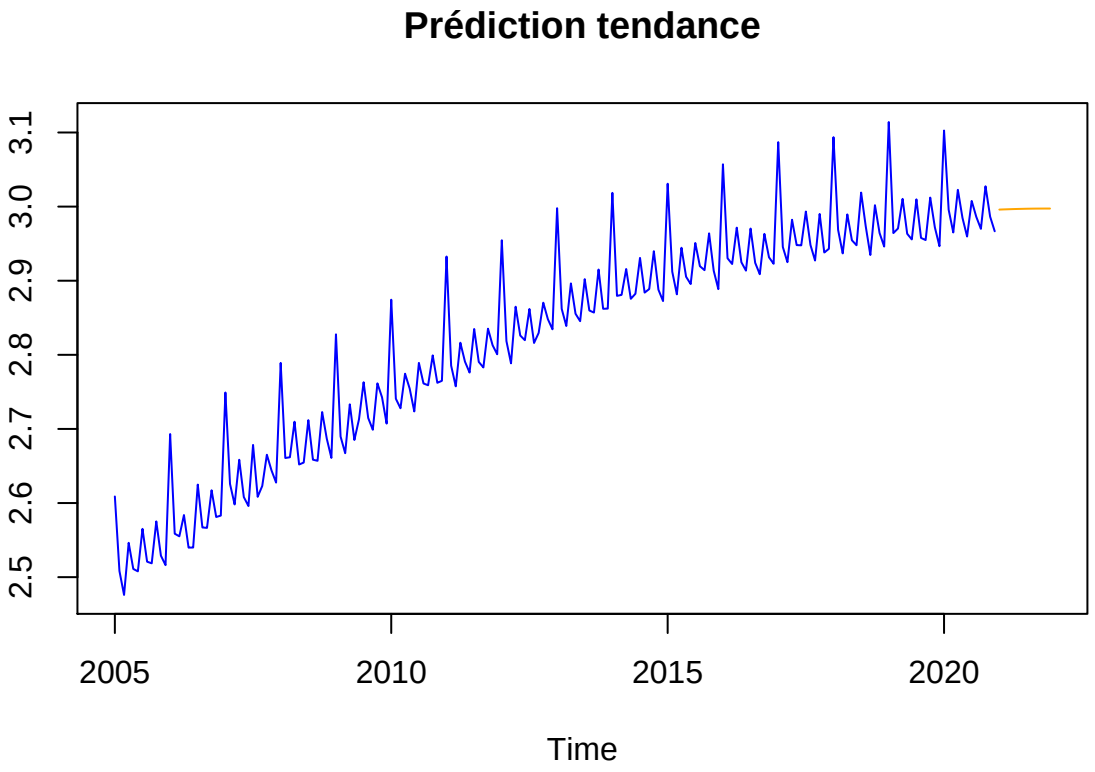
```
##
##  Box-Ljung test
##
## data:  ResidusPrimes
## X-squared = 16.828, df = 20, p-value = 0.6641
```

On trouve ici un p-value supérieure à 5% : On ne rejette pas l'hypothese d'indépendance des résidus.

2.7 Question 7

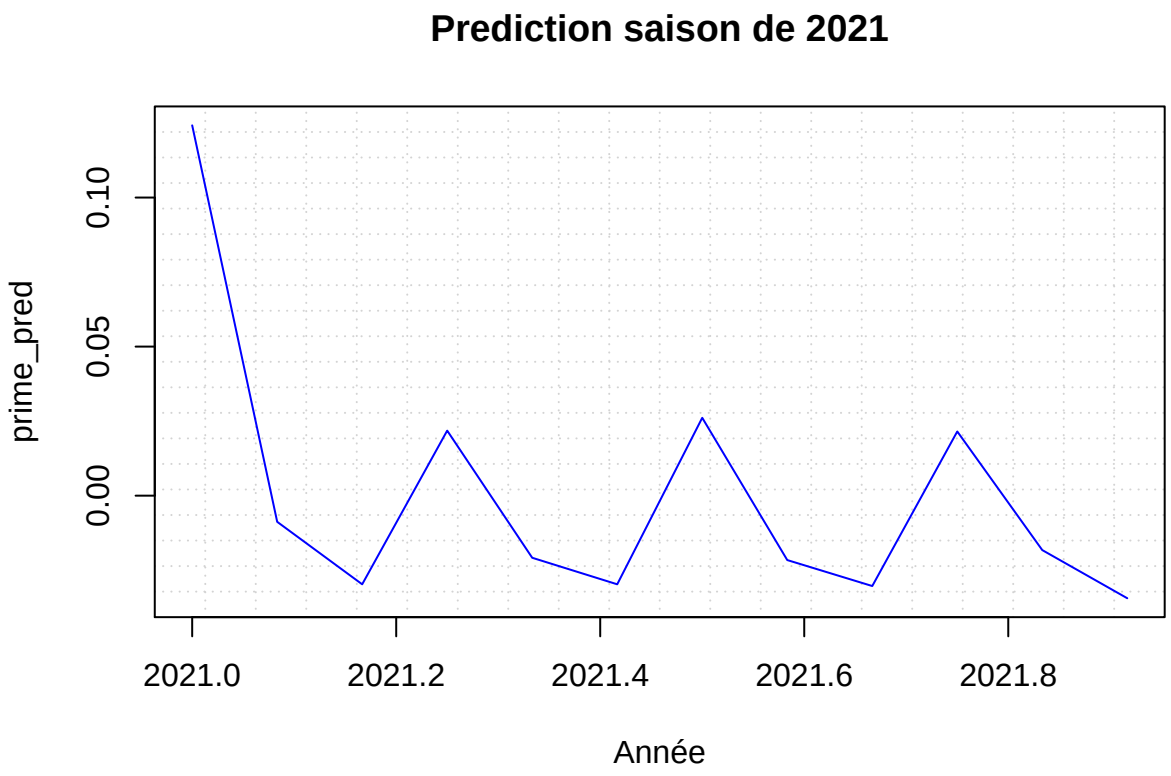
Afin de prédire la tendance, nous créons un vecteur contenant les composantes des 12 prochains mois.

```
#Prediction
#Tendance :
newT = seq(2021,2022,(1/12))[-13] #création d'un vecteur contenant les composantes temps des 1
pred_tend=predict(lm.primes,   newdata=data.frame(T=newT))
pred_tend_ts= ts(pred_tend,start=c(2021,1),frequency = 12)
ts.plot(logPrimes,pred_tend_ts,col=c("blue","orange"),main="Prédiction tendance")
```

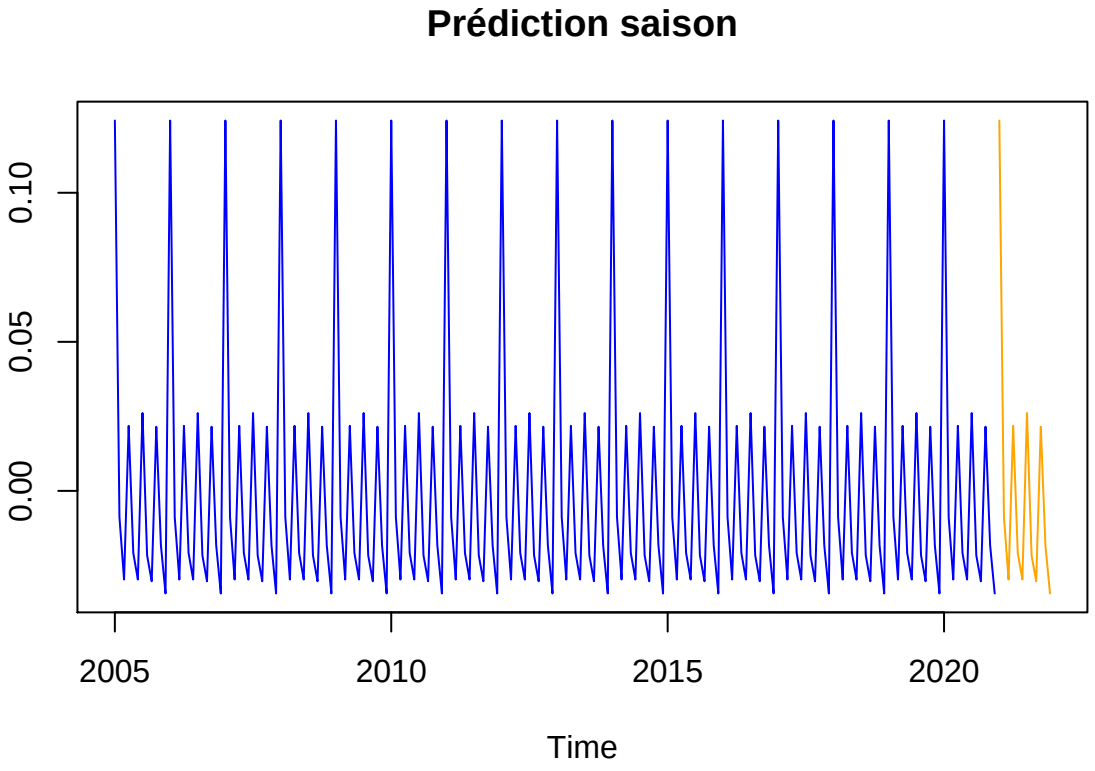


Afin maintenant de prédire la saison, nous utilisons la fonction min_aic définie précédement pour obtenir le meilleur modèle. Nous trouvons que les ordres donnes par l'algo sont 11,1,5 mais l'arima est meilleure avec les ordres 12,1,5.

```
#Saison
#min_aic(SEst,c(0,12),c(1,1),c(0,12))

model_prime <-arima(SEst,order = c(12,1,5), method ='ML')

pred_prime<-predict(model_prime,12)

prime_pred <- pred_prime$pred
plot(prime_pred,
     type = 'l',col='blue',
     main="Prediction saison de 2021"
     ,xlab="Année",
     panel.first = grid(20))
```

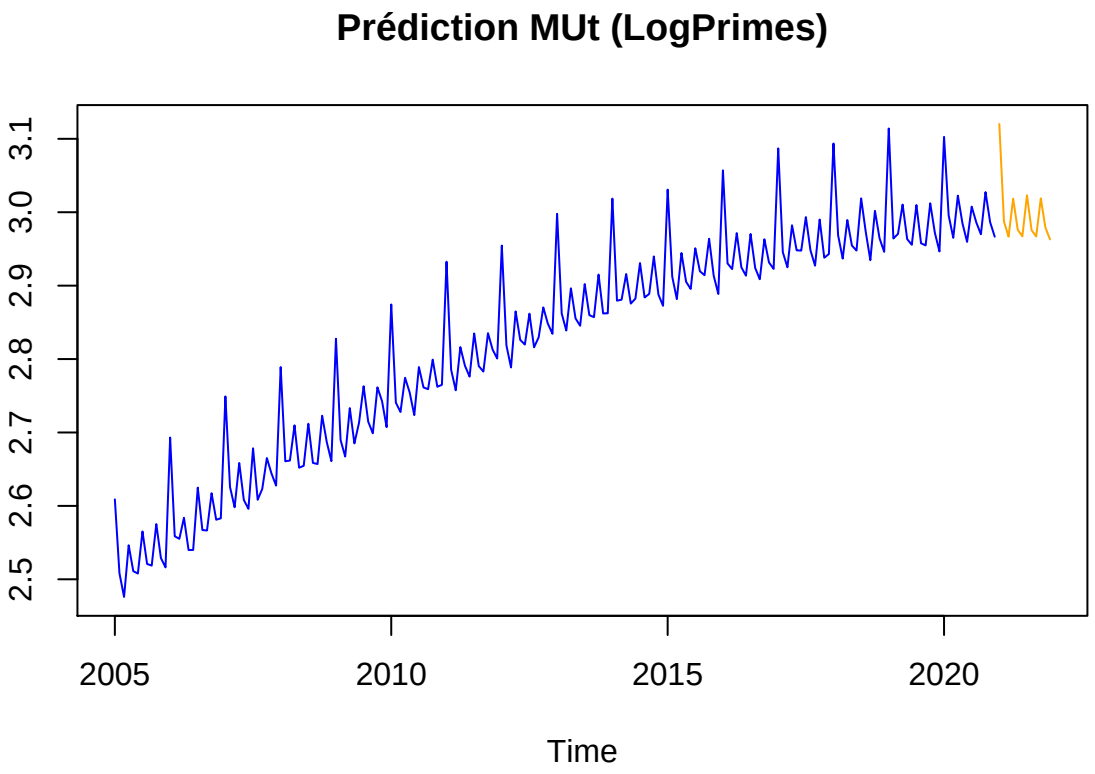



```
prime_pred_ts= ts(prime_pred,start=c(2021,1),frequency = 12)
ts.plot(SEst,prime_pred_ts,col=c("blue","orange"),main="Prédiction saison")
```



Enfin, nous réprésentons notre modèle prédit avec toutes les prédictions faites précedemment.

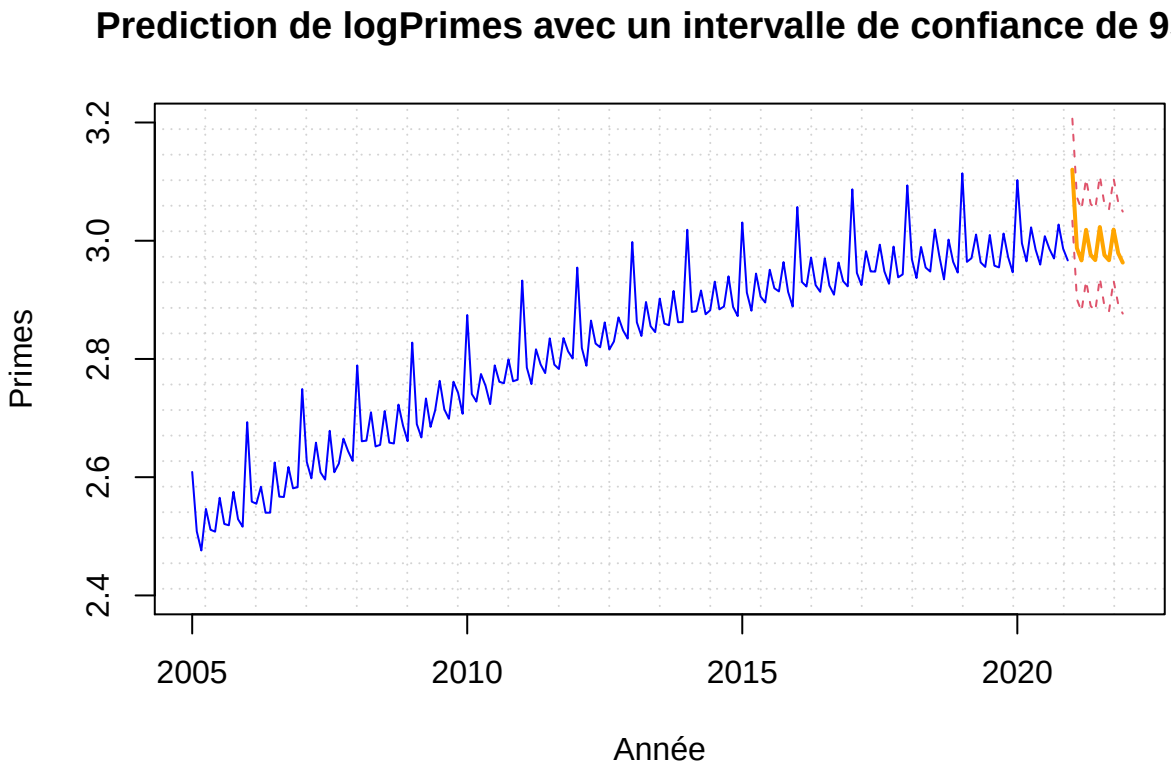
```r
#Tendance + Saison (predits)
#MUt=Mpt + St
MUt = pred_tend +  prime_pred
ts.plot(logPrimes,MUt  ,col=c("blue","orange"),main="Prédiction MUt (LogPrimes)")
```



2.8 Question 8

```r
#valeur predites :
prim_val<-ts(MUt,end =end(MUt),frequency = frequency(MUt))

#valeur predites supérieures :
prim_up<-ts(MUt + qnorm(0.975)*sd(PrimesDetend),end =end(MUt),frequency = frequency(MUt))

#valeur predites inférieures :
prim_down<-ts(MUt-qnorm(0.975)*sd(PrimesDetend),end =end(MUt),frequency = frequency(MUt))

# Graphe :
plot(logPrimes,
     type = 'l',col='blue',
     main="Prediction de logPrimes avec un intervalle de confiance de 95%"
     ,ylab="Primes",xlab="Année",panel.first = grid(20),
     xlim=c(start(logPrimes)[1],end(logPrimes)[1]+2),
    ylim=c(2.4,3.2))
```

```r
lines(prim_val,type = 'l',col='orange',lty=1,lwd=2)
points(prim_up, type = "l", col = 2, lty = 2)
points(prim_down, type = "l", col = 2, lty = 2)
```



Nous comme pour les sinistres, nous avons crée/ressorti plusieurs éléments : les valeurs predites des primes , les valeurs predites supérieures, les valeurs predites inférieures. Nous obtenons donc ce graphe en représentant les primes ainsi que leurs prédictions grace à un plot où nous "plaçons" les éléments créé précédemment.

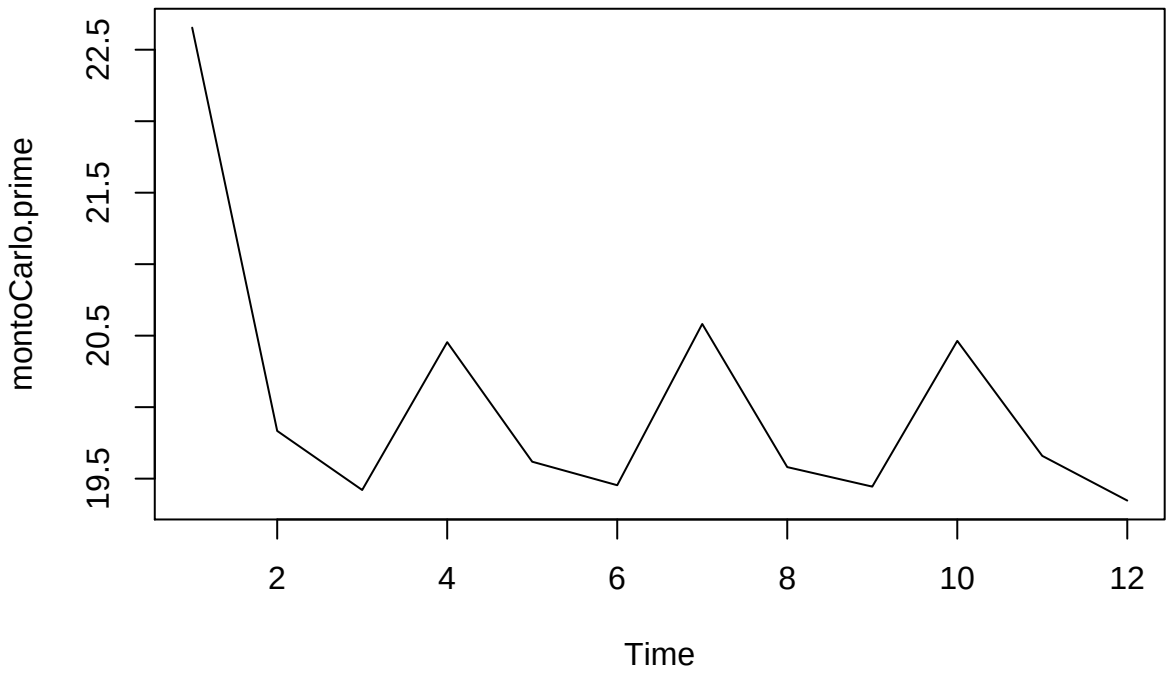
2.9 Question 9

Initialement : Primes = MtStZt avec Mt la tendance, St la saisonnalité, Zt les residus. Nous avons appliqué le log à la série chronologique Primes afin d'avoir un modèle additifs alors nous avons : logPrimes=lm.primes+SEst+PrimesDetend E(Primes)=E(exp(logPrimes)) =E(exp(lm.primes+SEst+PrimesDetend)) =E(ext(lm.primes))*E(exp(SEst))*E(exp(PrimesDetend)) par indépendance.

2.10 Question 10

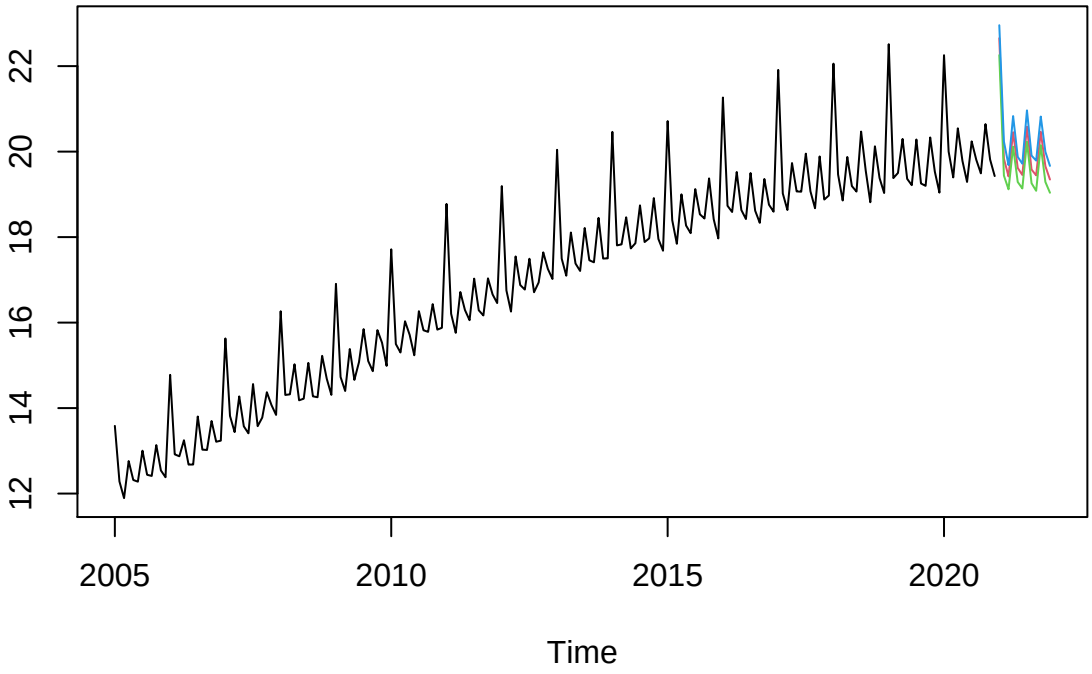
Pour les questions a, b, c et d, après avoir simuler et stocker 100 trajectoires de Z˜t de longueur 12 dans une matrice ZPmat de taille 12x100 et Ajouter à chacune des trajectoires (à chaque colonne) l'espérance de logPrimes, on utilise la fonction replicate pour simule 100 fois les 12 mois. Ensuite on repete 100 fois les 12 valeurs predites. Enfin, on ajoute à chacune des trajectoires (à chaque colonne) l'espérance de primes et on moyenne chaque ligne k pour obtenir un estimateur de l'espérance de chaque valeur prédite (grace à la méthode Monte-Carlo). Nous obtenons donc :

```r
#a
n=12
m=100
var_res=sd(PrimesDetend)
Mat <- matrix(rnorm(100*12,mean=0,var_res),n,m)

#b
#Mat+matrix(MUt,n,m)
sum_mat=ts(Mat+matrix(MUt,n,m),start=c(2021,1),frequency = 12)
#ts.plot(sum_mat)

#c

ZSMAT.prime = replicate(100,expr=rnorm(12,0,sd = sd(ResidusPrimes))  )

rep.prime<- replicate(100,as.vector(MUt))


trajecto.prime.BB <- exp(ZSMAT.prime+rep.prime)

#d

montoCarlo.prime<- apply(trajecto.prime.BB,1,mean)
ts.plot(montoCarlo.prime)
```



Pour la suite nous avons utilisé une fonction “put_in_time” qui permet de transformer la formule en série chronologique.

```r
#e

montoCarlo.prime.all = apply(trajecto.prime.BB, 1, function(yy) c(mean = mean(yy),quantile(yy,

put_in_time<- function(data){
  return(ts(data,start=c(2021.000,1),frequency = 12))
}

MoCar.prime_m<- put_in_time(montoCarlo.prime.all["mean",])
MoCar.prime_l <- put_in_time(montoCarlo.prime.all["2.5%",])
MoCar.prime_u<- put_in_time(montoCarlo.prime.all["97.5%",])
ts.plot(Primes,MoCar.prime_m,MoCar.prime_l,MoCar.prime_u,col=1:4)
```



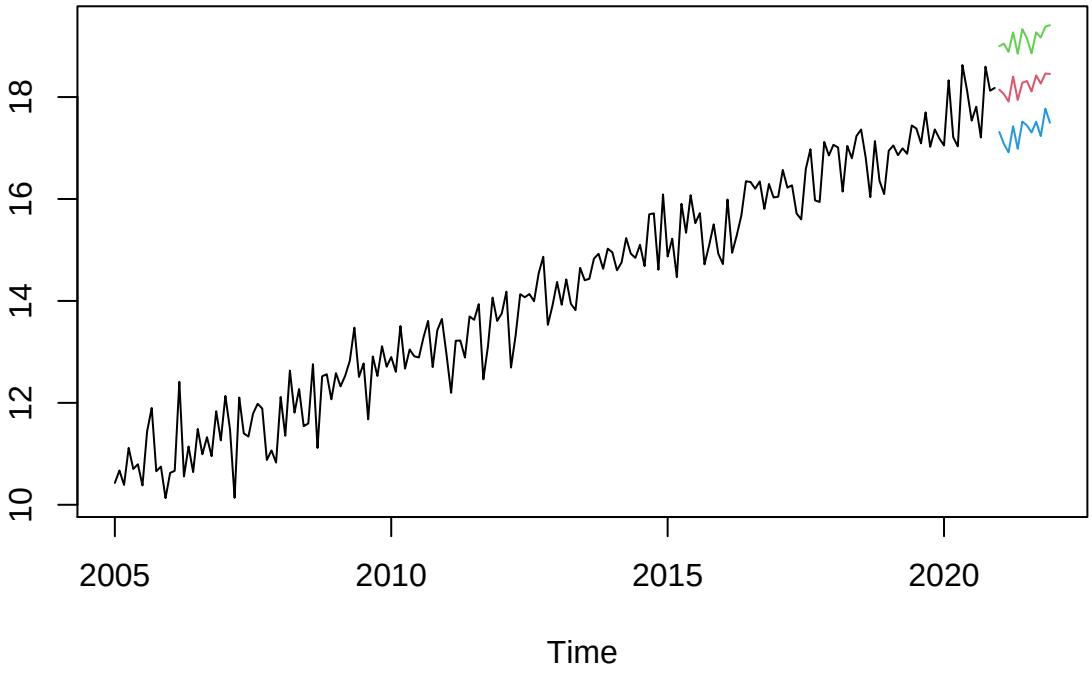
3 Prédiction de R = S/P par méthode de Monte-Carlo

3.1 Question 1

Pour les sinistres, on simule tout d'abord des valeurs de Sinistres par la methode de Monte-Carlo comme on a fait pour les Primes pour simuler 100 trajectoires des 12 valeurs futures de Sinistres. On utilise la fonction replicate pour simuler 100 fois les 12 mois ensuite on repete 100 fois les 12 valures predite de sinistres et on ajoute à chacune des trajectoires (à chaque colonne) l'espérance de sinistres. On moyenne ensuite chaque ligne k = 1, . . . , 12 pour obtenir un estimateur de l'espérance de chaque valeur prédite. Enfin, on applique la fonction quantile à chaque ligne k = 1,.., 12 pour obtenir une estimation du quantile d'ordre 0,975 et 0,025. Nous obtenons donc pour finir :

on utilise la fonction replicate pour simuler 100 fois les 12 mois

```
ZSMAT.sini = replicate(100,expr=rnorm(12,0,sd = sqrt(var(SinistresDetend )))  )

rep.sini<- replicate(100,as.vector(sini_pred))


trajecto.sini.BB <- ZSMAT.sini+rep.sini


montoCarlo.sini<- apply(trajecto.sini.BB,1,mean)


montoCarlo.sini.all = apply(trajecto.sini.BB, 1, function(yy) c(mean = mean(yy),quantile(yy,c(0

MonCar.sini_mean<- put_in_time(montoCarlo.sini.all["mean",])
MonCar.sini_down <- put_in_time(montoCarlo.sini.all["2.5%",])
MonCar.sini_up<- put_in_time(montoCarlo.sini.all["97.5%",])
ts.plot(Sinistres,MonCar.sini_mean,MonCar.sini_up,MonCar.sini_down,col=1:4)
```



3.2 Question 2

Pour cette question nous avons trouvé qu'il faudrait augmenter les primes d'un facteur pour que les deux conditions soient vérifiées. Nous ramenons alors le quantile à 2,5% de R à 60%. Ce facteur serait : facteur = R2021.pred.low/0.6. Nous trouvons que l'augmentation maximale serait de 1.499754 = 149%.

```r
#Pour chacune des 100 trajectoires i = 1, . . . , 100, on calcule le Rvec=S/P

Rvec <- apply(trajecto.sini.BB,2,sum)  / apply(trajecto.prime.BB,2,sum)

R2021.pred      = mean(Rvec)
R2021.pred.up   = quantile(Rvec,0.975)
R2021.pred.low  = quantile(Rvec,0.025)


# R2021.pred.low/facteur = 0.6
# facteur = R2021.pred.low/0.6
facteur = R2021.pred.low/0.6
```

L'augmentation maximale serait de $1.499754 = 149\%$