Projet - MST

Thibaut MILHAUD & Thomas KOWALSKI

6 mai 2018

Statistiques descriptives

Comparaison hommes/femmes

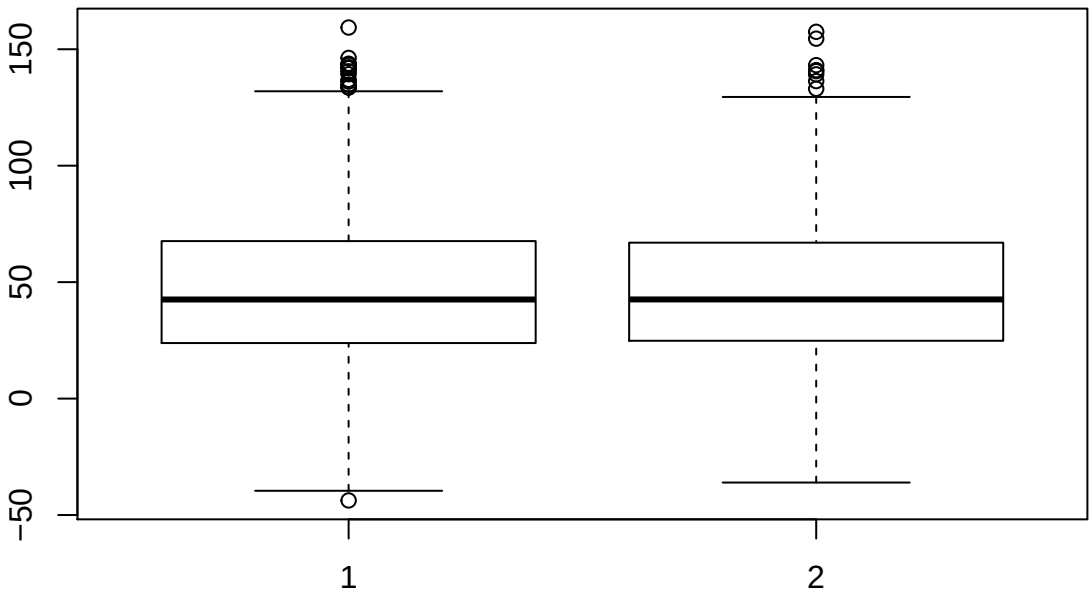
```
data <- read.csv(file = "DB_binome_2.csv");
n <- nrow(data);
mandata <- c();
womandata <- c();
for (i in 1:n)
{
  if(data[i, 'Sexe'] == 0)
  {
    mandata <- c(mandata, data[i, 'Peche'])
  }
  else
  {
    womandata <- c(womandata, data[i, 'Peche'])
  }
}
print(mean(womandata))
```

```
## [1] 47.73338
```

```
print(mean(mandata))
```

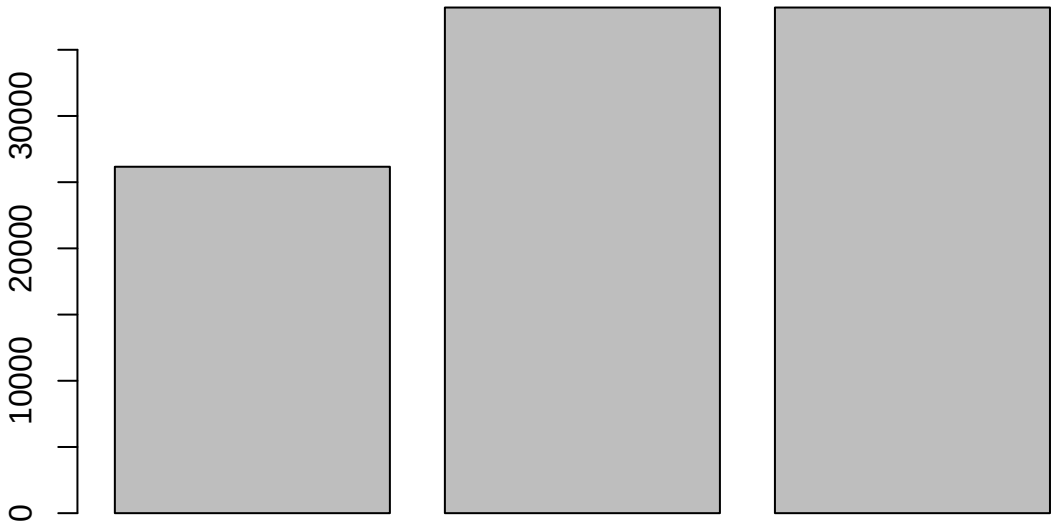
```
## [1] 47.93575
```

```
boxplot(mandata, womandata)
```



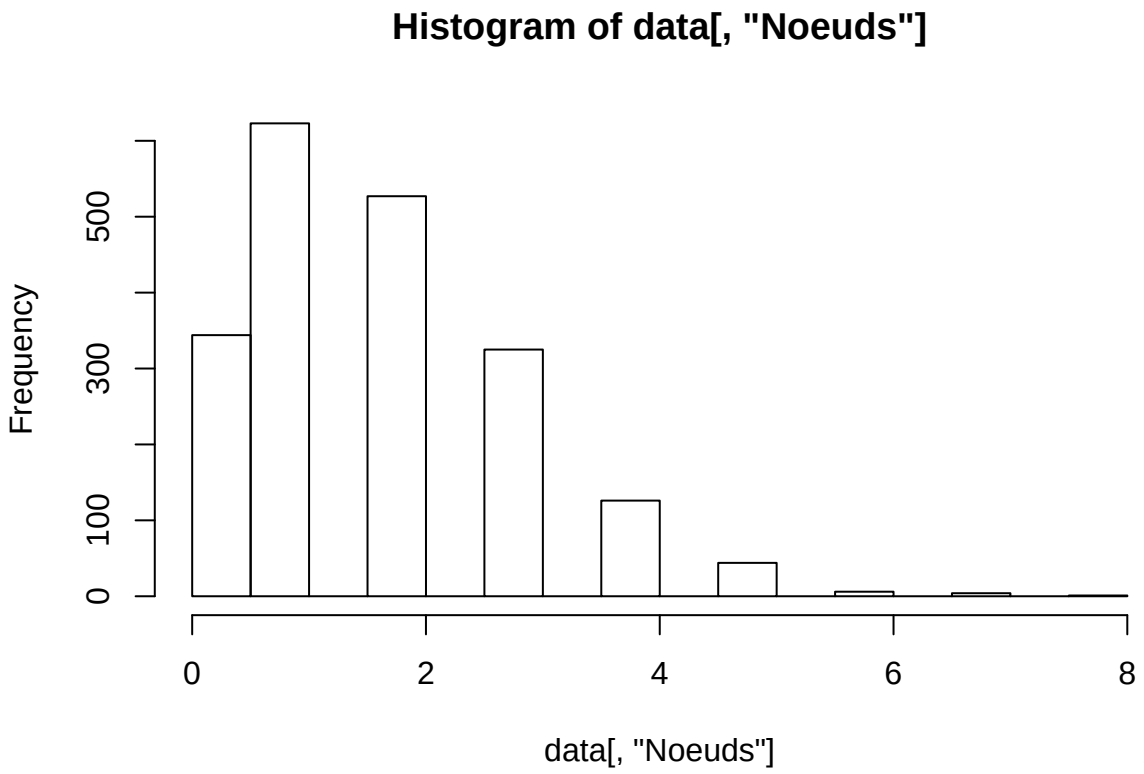
Distribution de la pêche en fonction de la tranche d'âge

```
tranches = c(0, 0, 0)
for (i in seq(1, n)) {
    tranche = data[i, "Age"]
    tranches[tranche - 1] = tranches[tranche - 1] + data[i, "Peche"]
}
barplot(tranches)
```



Intensité du vent

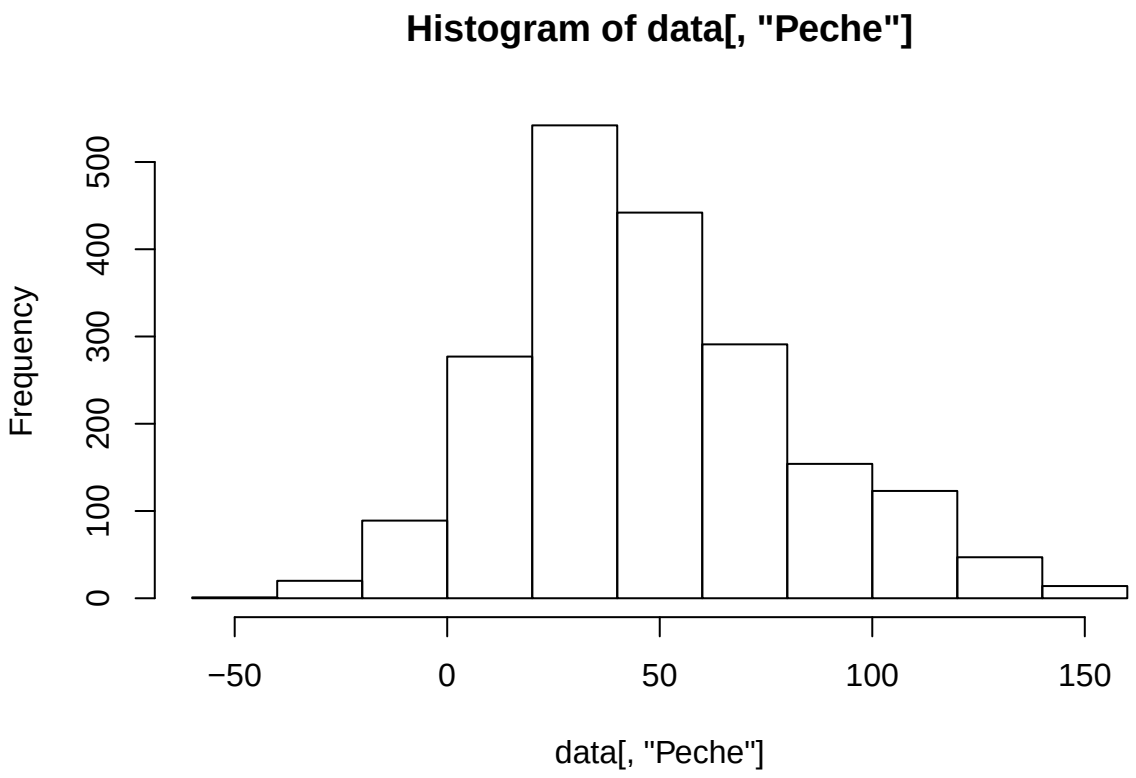
```
hist(data[,'Noeuds'])
```



On remarque que l'intensité du vent est entière, de plus, l'allure de cet histograme rappelle un loi de poisson.

Quantité de pêche

```
hist(data[,'Peche'])
```



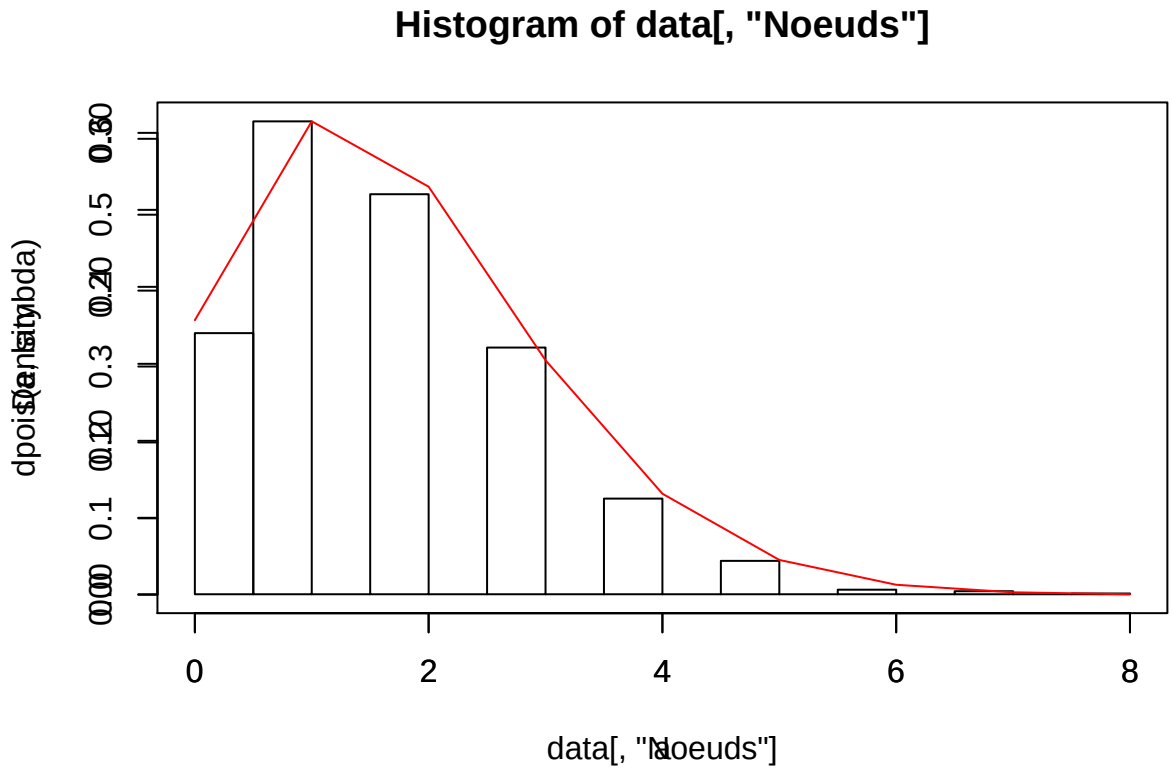
On dirait une loi Normale.

Statistiques Inférentielles

Etude du modèle suivi par le vent

Cohérence avec la loi de poisson

```
#On regarde la cohérence par rapport à la loi de poisson
a <- seq(0,8,1)
lambda = mean(data[,'Noeuds'])

hist(data[,'Noeuds'],freq=FALSE,breaks = seq(0,8,0.5))
par(new=TRUE)
plot(a,dpois(a,lambda),"l",col="red")
```



Vraisemblance

Soit X un echantillon de taille n suivant une loi de poisson de paramêtre λ, alors sa vraisemblance vaut :

$$L_\lambda(X) = \prod_{i=1}^{n} \exp(-\lambda)\frac{\lambda^{x_i}}{x_i!} = \exp(-n\lambda)\frac{\lambda^{\sum x_i}}{\prod x_i!}$$

d'où,

$$\mathcal{L}_\lambda(X) = \log(L_\lambda(X)) = -n\lambda + \log\lambda \sum x_i - \sum \log x_i!$$

Ainsi en dérivant $\mathcal{L}_\lambda(X)$ par rapport à λ, on obtient :

$$\frac{\partial \mathcal{L}_\lambda(X)}{\partial \lambda} = -n + \frac{\sum x_i}{\lambda}$$

et

$$\frac{\partial^2 \mathcal{L}_\lambda(X)}{\partial \lambda^2} = -\frac{\sum x_i}{\lambda^2} \leq 0$$

La log-vraisemblance est donc concave ce qui signifie que les points où la dérivée s'annule sont des maximums globaux. Ainsi,

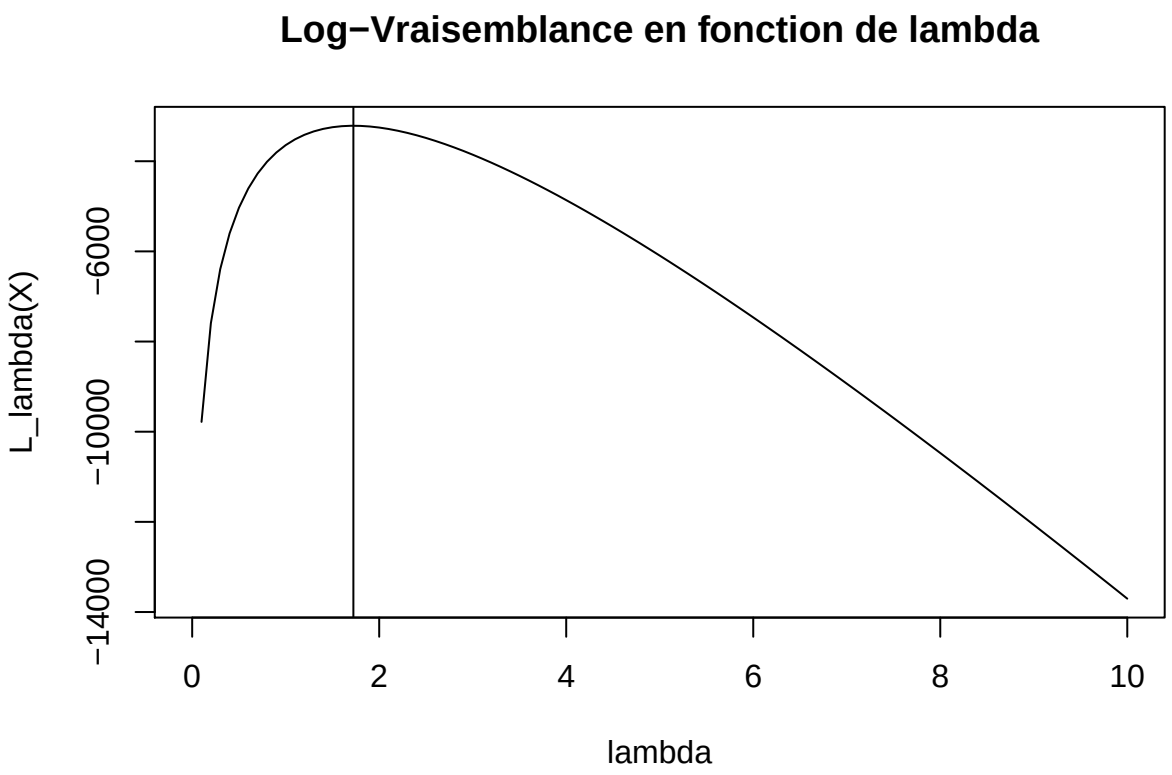
$$\lambda_{\max} = \frac{\sum_{i=1}^{n} x_i}{n}$$

```
log_L = function(x, l) {
  s = 0
  for(i in seq(1, length(x)))
  {
    s = s + log(factorial(x[i]))
  }
  return(-1 * length(x) * l + log(l) * sum(x) - s)
```

```
}

x = seq(0, 10, 0.1)
plot(x = x,
     y = log_L(data[, "Noeuds"], x),
     main = "Log-Vraisemblance en fonction de lambda",
     xlab = "lambda",
     ylab = "L_lambda(X)",
     type = "l")
lambda = mean(data[,'Noeuds'])
print(lambda)
```

```
## [1] 1.724
```

```
par(new = TRUE)
abline(v = lambda)
```



Estimation du paramètre sigma

Formules des probabiltés totales

D'après la formule des probabilités totales, on obtient

$$P(\text{ pêche} = x) = \sum_{i=0}^{\infty} P(\text{ vent} = i) \times \mathbb{P}(\text{ pêche} = x | \text{ vent} = i)$$

c'est-à-dire que la densité de la loi de X est la suivante :

$$f_X(x) = e^{-\lambda} \frac{1}{\sqrt{2\pi}\sigma} \sum_{i=0}^{\infty} \frac{\lambda^i}{i!} \exp\left(-\frac{(x - \frac{100}{1+i})^2}{2\sigma^2}\right)$$

On peut alors calculer la vraisemblance en faisant le produit des densités evaluées en x_i :

$$L_\sigma(x) = \prod_{i=1}^{n} f_X(x_i)$$

Cependant l'expression de ce produit ne parait pas se simplifier et on va donc devoir calculer la vraisemblance numériquement puis de trouver son maximum grâce à la fonction \verb{optimize}. Pour nos applications numériques, on veut trouver i tel que $\mathbb{P}(vent = i) < 0.00001$ (sachant qu'on utilise le λ de max vraisemblance trouvé précédemment.) afin de travailler avec des sommes finies.

```
print(lambda)
```

```
## [1] 1.724
```

```
for(i in seq(1, 100)) {
  p = exp(-1 * lambda) * lambda ^ i / factorial(i)
  if (p < 0.00001)
  {
    break
  }
}
print(i)
```

```
## [1] 11
```

On trouve donc que la valeur de vent dont la probabilité est inférieure à 0.0000001 est **11**.

Pour déterminer le σ de maximum vraisemblance on utilise la fonction `optimize` avec une fonction de vraisemblance que l'on définit.

```
logvraiss = function(sigma, lambda, X, imax) {
  #initalisation du tableau des puissances de lambda et des i factoriel de 0 à imax
  powers = c(1)
  facts = c(1)
  for(i in 2:(imax+1))
  {
    powers <- c(powers, lambda*powers[i-1])
    facts <- c(facts, (i-1)*facts[i-1])
  }
  logproduit = 0
  for (xi in X) {
    somme = 0
    for (i in i:(imax+1)){
      somme <- somme + powers[i]/facts[i]*exp(-(xi - 100/(i))**2/(2*sigma**2))
    }
    logproduit <- -lambda -log((2*pi)**0.5*sigma) + logproduit + log(somme)
  }
  return(logproduit)
}
```

```
vec = data[, "Peche"]
imax = 11
lambda = lambda # on garde l'ancien
sig = optimise(logvraiss, lambda = lambda, X = vec, imax = imax, lower = 0, upper = 100, maximum = TRUE)
print(sig)
```

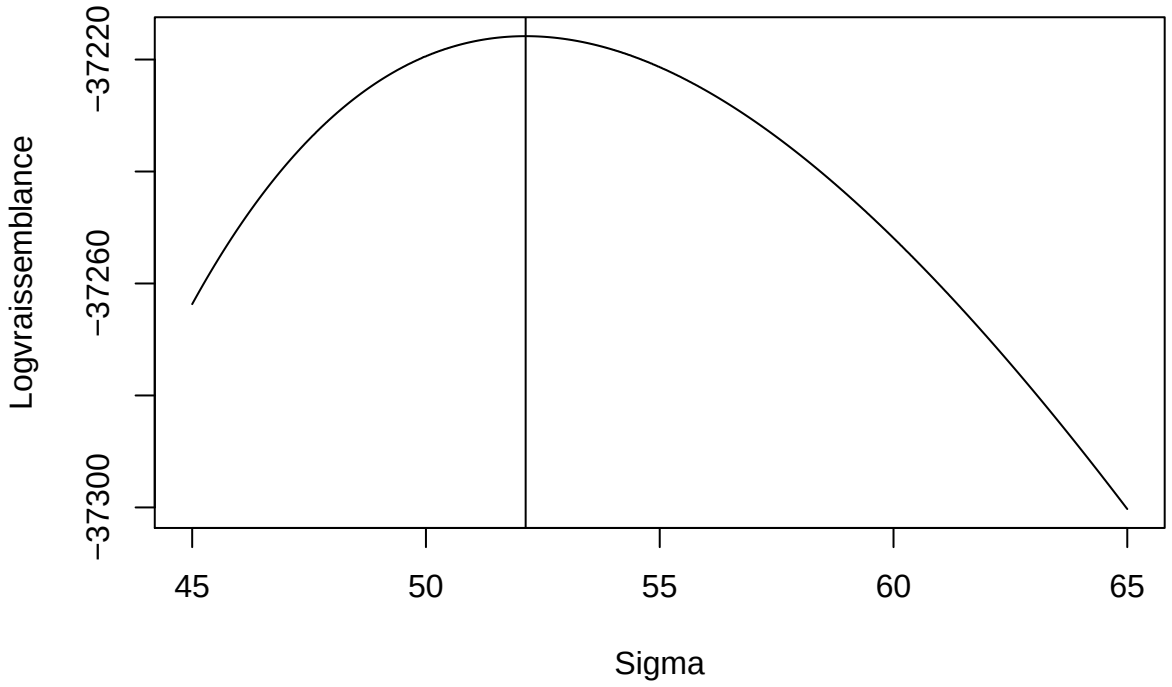
```
## $maximum
## [1] 52.13232
```

```
##
## $objective
## [1] -37215.82
```

```
sigma1 = sig$maximum
```

On obtient donc une valeur de $\sigma = 52.1$ en utilisant notre estimateur de vraisemblance approché.

```
Sigma = seq(45,65,0.001)
Logvraissemblance = logvraiss(Sigma,lambda,vec,imax)
plot(x = Sigma, y = Logvraissemblance, type = "l")
par(new = TRUE)
abline(v = sigma1)
```



Utilisation du T.C.L pour se ramener à une seule loi

On d'après le TCL on peut se ramener à une seule loi normale $\mathcal{N}(\mu, \sigma^2)$ on va donc utiliser la moyenne empirique pour estimer μ. D'où notre échantillon va suivre la loi $\mathcal{N}(\overline{X}, \sigma^2)$.

calcul de la vraisemblance :

$$L = (\frac{1}{\sqrt{2\pi}\sigma})^n \exp(-\frac{\sum (x_i - \overline{X})^2}{2\sigma^2})$$

D'où la log-vraisemblance vaut :

$$\mathcal{L}(\sigma) = -n \log(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2} \sum (x_i - \overline{X})^2$$

Et

$$\frac{\partial \mathcal{L}(\sigma)}{\partial \sigma} = \frac{-n}{\sigma} + \frac{1}{\sigma^3} \sum (x_i - \overline{X})^2$$

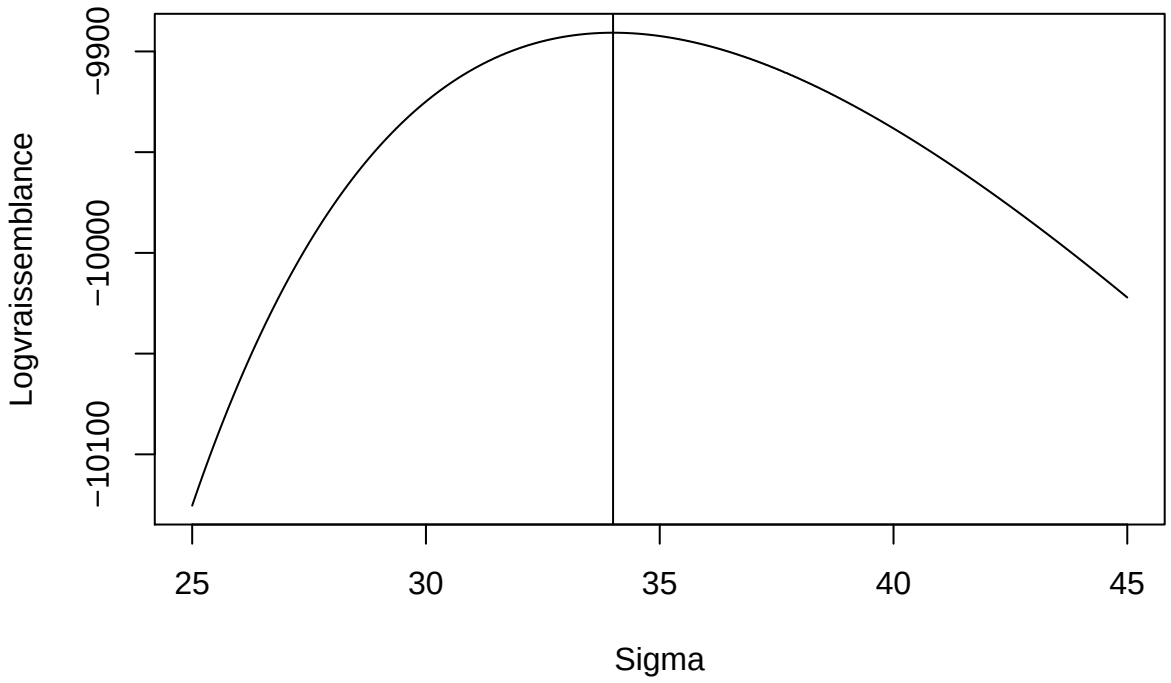
et en cherchant le point où la dérivée s'annule :

$$\sigma_{\max} = \sqrt{\frac{\sum (x_i - \overline{X})^2}{n}}$$

```
ecart_type = function (X)
{
  moy = mean(X)
  sum = 0
  for(x in X)
  {
    sum <- sum + (x - moy)**2
  }

  return((sum/length(X))**0.5)
}

logvraiss2 <- function(sigma,X)
{
  n = length(X)
  return(-n*log((2*pi)**0.5*sigma) - ecart_type(X)**2*n/(2*sigma**2))
}

sigma2 = ecart_type(vec)
print(sigma2)
```

```
## [1] 34.00199
```

```
Sigma = seq(25,45,0.001)
Logvraissemblance = logvraiss2(Sigma,vec)

plot(x = Sigma, y = Logvraissemblance, type = "l")
par(new = TRUE)
abline(v = sigma2)
```



Intervalles de confiance

Calculs théoriques

On sait que :

$$\Lambda_n = \sqrt{n}\frac{\overline{X} - \mu}{S_n} \sim T_{n-1}$$

où $\overline{X}$ correspond à la moyenne empirique, μ à l'espérance et $S_n = \frac{1}{n-1}\sum_{i=0}^{n}(x_i - \overline{X})^2$. D'où, dans notre cas,

$$\Lambda_n = \sqrt{n}\frac{\overline{X} - \lambda}{S_n} \sim T_{n-1}$$

Ainsi,

$$\begin{aligned} \mathbb{P}(a < \Lambda_n < b) = 1 - \alpha \quad &\iff \quad F^{-1}_{T_{n-1}}(\tfrac{\alpha}{2}) < \Lambda_n < F^{-1}_{T_{n-1}}(1 - \tfrac{\alpha}{2}) \\ &\iff \quad \overline{X} - \tfrac{S_n}{\sqrt{n}}F^{-1}_{T_{n-1}}(1 - \tfrac{\alpha}{2}) < \lambda < \overline{X} + \tfrac{S_n}{\sqrt{n}}F^{-1}_{T_{n-1}}(1 - \tfrac{\alpha}{2}) \end{aligned}$$

Application numérique

```
confiance <- function(X, alpha=0.05)
{
  n = length(X)
  q = qt(1-alpha/2,n)
  moy = 0
  for (xi in X)
  {
    moy <- moy + xi
  }
  moy <- moy/n

  sn = 0
  for (xi in X)
  {
    sn <- sn + (xi-moy)^2
  }
  sn <- sn/(n-1)

  q <- sn * q / n**0.5
  return(c(moy-q, moy+q))
}

confiance(data[,"Noeuds"])
```

```
## [1] 1.651527 1.796473
```

On obtient que λ a 95% de chances d'appartenir à l'intervalle $I = [1.651, 1.796]$.

Tests

Ultime question

On veut vérifier que lorsque $\lambda = 3$, la QUANTITE DE PECHE suit une loi $\mathcal{N}(\frac{100}{1+\lambda}, \sigma^2)$ avec $\sigma = 20$. On veut savoir pour quel α on ne peut rejeter cette hypothèse.

On utilise le test χ^2.

```
peche = c()
for (i in seq(1, n)) {
  if (data[i, "Noeuds"] == 3)
  {
    peche = c(peche, data[i, "Peche"])
  }
}


k = 15
m = min(peche)
M = max(peche)

intervalles = seq(m, M, length.out = k)
print(intervalles)
```

```
##  [1] -36.0529553 -26.9104386 -17.7679219  -8.6254052   0.5171115
##  [6]   9.6596282  18.8021449  27.9446616  37.0871783  46.2296950
## [11]  55.3722117  64.5147284  73.6572450  82.7997617  91.9422784
```

```
muChapeau = 0
for(i in seq(1, k - 1))
{
  ci = (intervalles[i] + intervalles[i + 1])
  ni = 0
  for (j in seq(1, length(peche)))
  {
    if (peche[j] >= intervalles[i])
    {
      if(peche[j] < intervalles[i + 1])
      {
        ni = ni + 1
      }
    }
  }
  muChapeau = muChapeau + ci * ni
}
muChapeau = muChapeau / length(peche)

sigmaChapeau = 0
for(i in seq(1, k - 1))
{
  ci = (intervalles[i] + intervalles[i + 1])
  ni = 0
  for (j in seq(1, length(peche)))
  {
    if (peche[j] >= intervalles[i])
    {
      if(peche[j] < intervalles[i + 1])
      {
        ni = ni + 1
      }
    }
  }
```

```r
  sigmaChapeau = sigmaChapeau + ni * (ci - muChapeau) * (ci - muChapeau)
}
sigmaChapeau = sigmaChapeau / length(peche)
sigmaChapeau = sqrt(sigmaChapeau)

print(muChapeau)
```

```
## [1] 46.49045
```

```r
print(sigmaChapeau)
```

```
## [1] 44.0246
```