TP no 3 - Bootstrap

Master parcours SSD - UE Statistique Computationnelle

Septembre 2019

1 Exercice 1

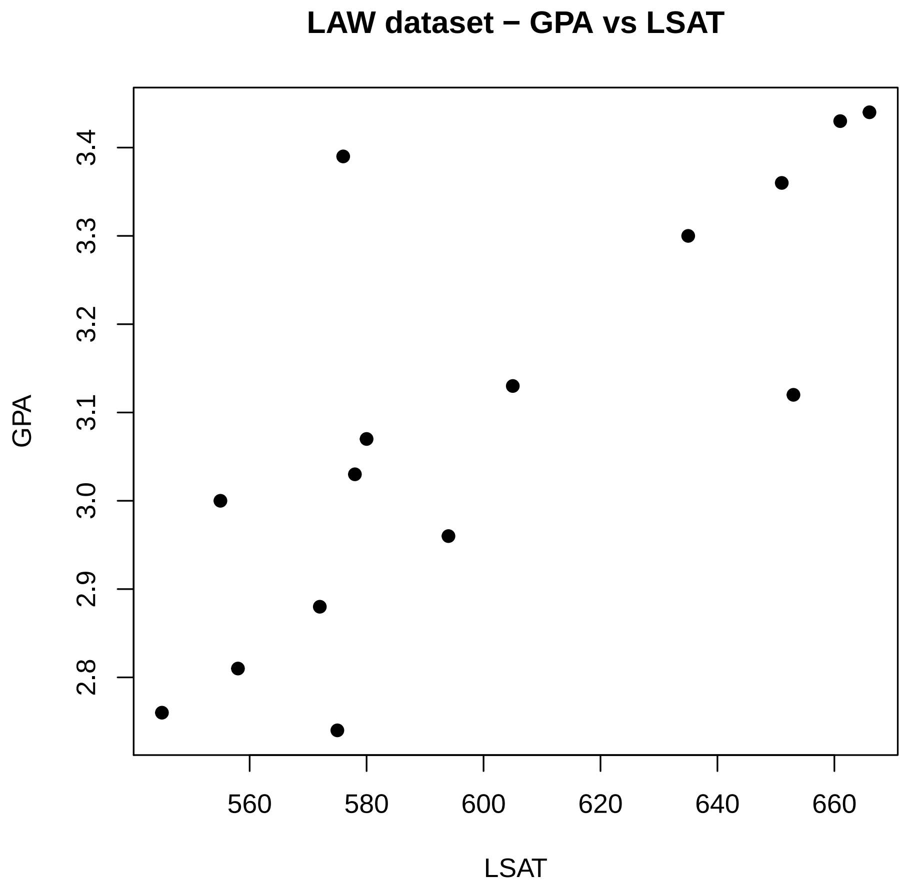
Dans cet exercice nous allons illustrer la technique du bootstrap sur un cas d'école. Pour cela, commencez par installer le package `bootstrap` pour avoir accès au jeu de données `law` que l'on va utiliser. Ce jeu de données contient deux variables : `LSAT` et `GPA` dont on veut estimer la corrélation. Ces deux variables correspondent à des notes obtenues par des étudiants lors de leur examen d'admission à l'université et tout au long de leurs études secondaires. Une fois chargé le package `bootstrap`, on y accède via `law£GPA` et `law£LSAT`.

1. *Calculer le biais et l'erreur type du coefficient de corrélation empirique par une procédure bootstrap de $B = 200$ répétitions.*
2. *Calculer les intervalles de confiance à 95% obtenus par les deux approches décrites dans le cours (approche "basique" et approche par percentiles)*
3. *Proposer une représentation graphique des résultats obtenus.*
4. *Le jeu de données `law` est en fait un sous-échantillon du jeu `law82`. Les intervalles de confiance obtenues sont-ils en accord avec la corrélation estimée à partir du jeu global ?*

Au préalable il faut donc installer le package `bootstrap` par la fonction `install.packages(boostrap)`. Commençons par visualiser le jeu de données et par calculer le coefficient de corrélation empirique.

```
> ##############################
> ####### STARTING EXO 1 ######
> ##############################
> library(bootstrap)
> set.seed(27)
> # on trace les variables
> x = law$LSAT
> y = law$GPA
> n = length(x)
> plot(x, y, pch = 19, xlab = "LSAT", ylab = "GPA", main = "LAW dataset - GPA vs LSAT")
> # on calcule la corrélation sur l'échantillon
> theta = cor(x,y)
> cat("*** sample correlation =", theta, "***\n")
```

```
*** sample correlation = 0.7763745 ***
```

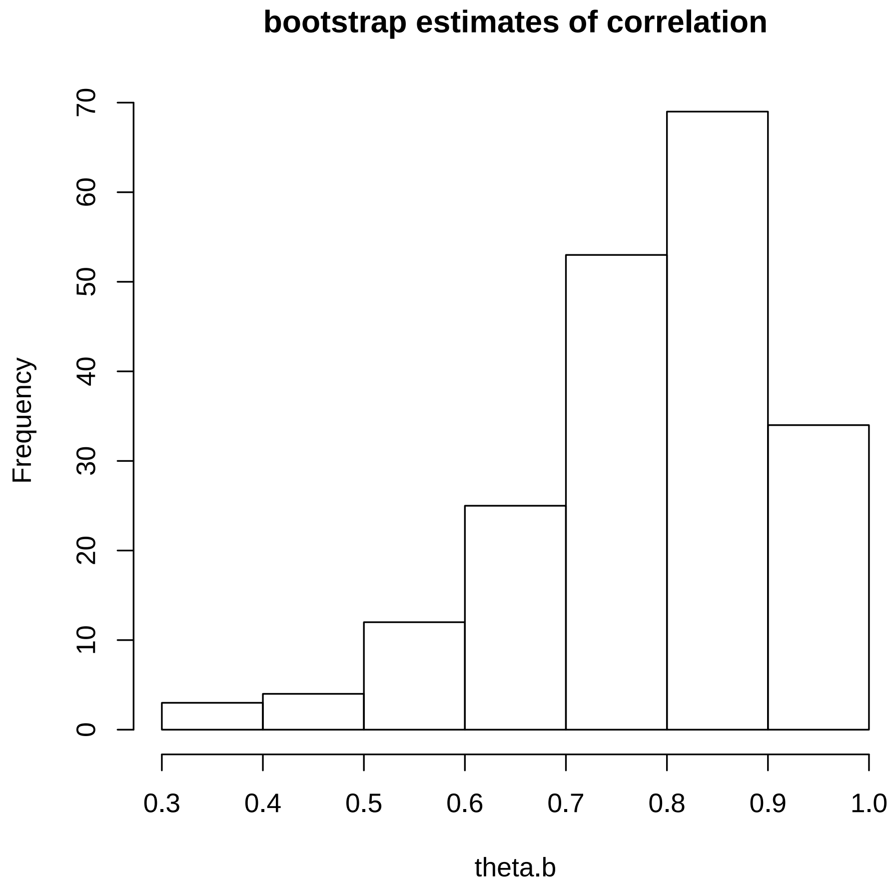


Appliquons ensuite la procédure bootstrap classique pour estimer le biais et l'erreur type du coefficient de corrélation empirique. On en profite pour tracer la distribution d'échantillonnage obtenue par bootstrap, et on note qu'elle n'est pas normale.

```
> # proc\'edure bootstrap
> B = 200
> theta.b = numeric(B)
> for(b in 1:B){
+     ind = sample(c(1:n), n, replace = TRUE)
+     theta.b[b] = cor(x[ind],y[ind])
+ }
> # compute bias & SE
> bias = mean(theta.b) - theta
> se = sd(theta.b)
> cat("** bias =", round(bias, digits=5), "standard-error =", round(se, digits = 3), "**\n*"
```

```
** bias = 0.00618 standard-error = 0.127 **
*
```

```
> # show distribution
> hist(theta.b, main = "bootstrap estimates of correlation")
```



On peut ensuite calculer les intervalles de confiance par les méthodes "basique" et percentiles à partir des quantiles d'ordre $\alpha/2$ et $1-\alpha/2$ de la distribution obtenue par bootstrap.

```
> # compute confidence intervals
> alpha = 0.05
>     # percentile method
> q1 = quantile(theta.b, alpha/2)
> q2 = quantile(theta.b, 1-alpha/2)
> Iperc = c(q1,q2)
> cat("percentile confidence interval = [", Iperc[1], ";", Iperc[2], "]\n")
```

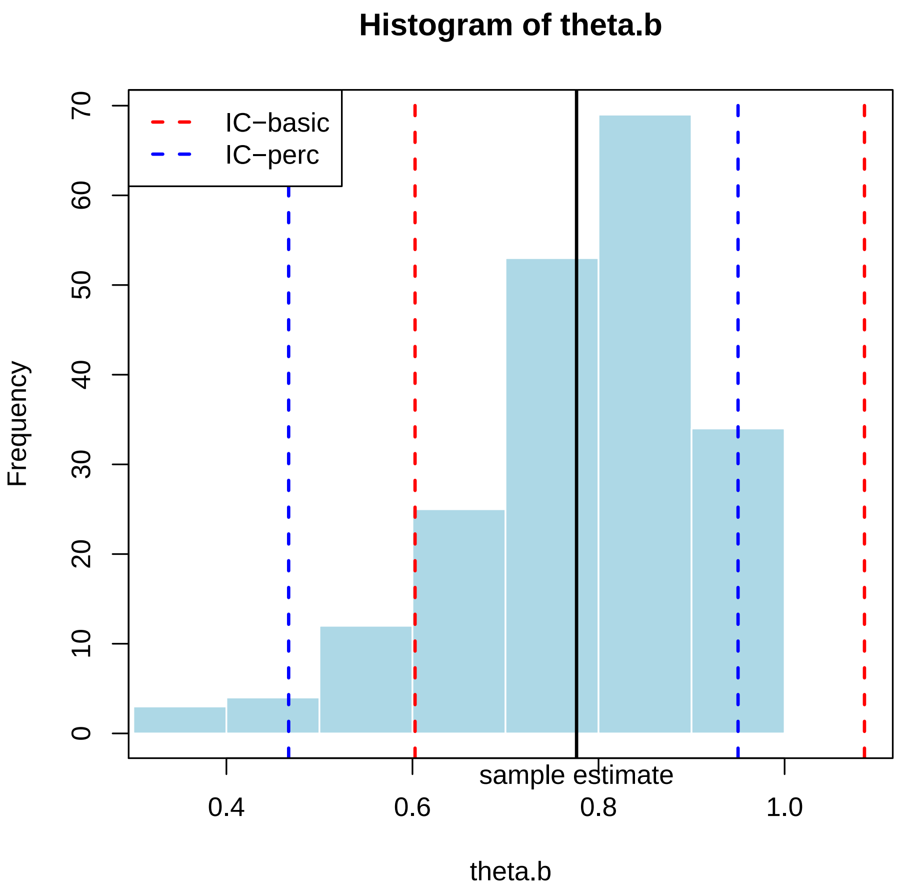
```
percentile confidence interval = [ 0.4669003 ; 0.9499815 ]
```

```
> # basic method
> I1 = 2*theta - q2
> I2 = 2*theta - q1
> Ibasic = c(I1,I2)
> cat("basic confidence interval = [", Ibasic[1], ";", Ibasic[2], "]\n")
```

```
basic confidence interval = [ 0.6027675 ; 1.085849 ]
```

Enfin, pour résumer les résultats, on peut superposer ces intervalles de confiance et la distribution des estimations obtenues par bootstrap.

```
> # plot
> hist(theta.b, col = "lightblue", border = "white", xlim = range(c(theta.b,Iperc,Ibasic)))
> abline(v = Ibasic,lty = 2, lwd = 2, col = "red")
> abline(v = Iperc,lty = 2, lwd = 2, col = "blue")
> legend("topleft", c("red","blue"), c("IC-basic","IC-perc"), col = c("red","blue"),lty =2,
> abline(v = theta, lwd = 2)
> mtext("sample estimate",  side = 1, at = theta)
> box()
```



2 Exercice 2

Reproduire cette analyse en utilisant les fonctions `boot` et `boot.ci` du package `boot`. On rappelle qu'il s'agit ici d'une procédure de bootstrap "ordinaire", et on suggère de porter une attention particulière à l'option `statistic`.

Au préalable, il faut donc installer le package `boot`. La fonction `boot` (du package du même nom) permet de réaliser très simplement l'analyse de la section 2. Pour cela il suffit d'implémenter une fonction calculant la statistique d'intérêt. Cette fonction doit prendre deux arguments : le jeu de données global, et un vecteur d'indices (compris entre 1 et n si on dispose de n observations) spécifiant les observations sur lesquelles doit être calculée la statistique. Dans notre cas, elle est particulièrement simple :

```
> ##############################
> ####### STARTING EXO 2 ######
> ##############################
> library(boot)
```

```
> # define a function to compute the statistic
> my.cor = function(x,ind){
+   return( cor(x[ind,1],x[ind,2]) )
+ }
```

Il suffit alors d'appeler la fonction `boot` en donnant le nom de notre fonction à l'argument `statistic` :

```
> # call the bootstrap procedure
> X = cbind(x,y)
> res = boot(data = X, statistic = my.cor, R = B)
```

On peut alors afficher directement les résultats à l'écran :

```
> # print  results
> print(res)


ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = X, statistic = my.cor, R = B)


Bootstrap Statistics :
     original       bias    std. error
t1* 0.7763745 -0.02283905   0.1409826
```

L'objet renvoyé par la fonction `boot` peut ensuite être donné comme entrée à la fonction `boot.ci` pour calculer les intervalles de confiance correspondants. Notons que cette fonction propose également d'autres défintions que celles décrites dans le cours.

```
> # compute confidence intervals
> res.ci= boot.ci(res, type = c("basic","perc","norm"))
> # print results
> print(res.ci)

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 200 bootstrap replicates

CALL :
boot.ci(boot.out = res, type = c("basic", "perc", "norm"))

Intervals :
Level      Normal              Basic              Percentile
```

```
95%   ( 0.5229,  1.0755 )   ( 0.5838,  1.1067 )   ( 0.4461,  0.9689 )
Calculations and Intervals on Original Scale
Some basic intervals may be unstable
Some percentile intervals may be unstable
```

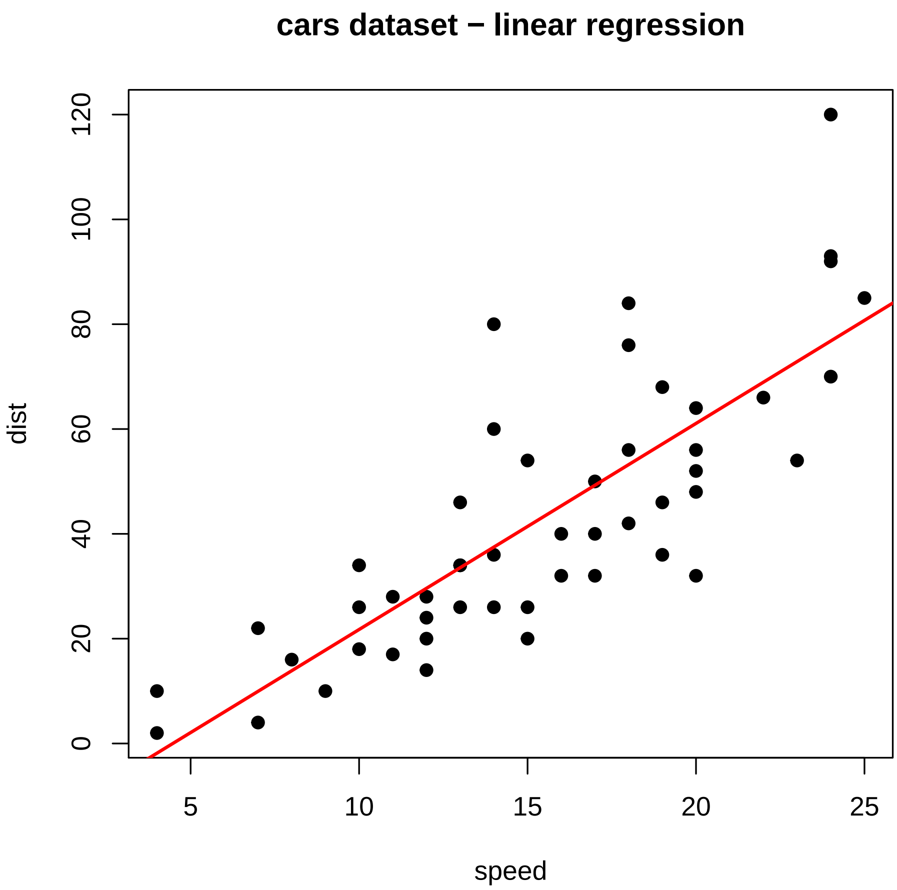
3 Exercice 3

Nous allons travailler sur le jeu de données `cars` *qui contient deux variables : la vitesse de voitures des années 1920 et les distance parcourues pour qu'elles s'arr<ea>tent.*

1. *Visualiser la distance en fonction de la vitesse et superposer le résultat d'une régression linéaire classique (i.e., avec intercept).*
   — *NB : le jeu de données* `cars` *fait "nativement" partie de* `R`*. On accède aux deux variables mentionnées ci-dessus via* `cars£speed` *et* `cars£dist`*.*
2. *Effectuer une régression "bootstrappée" par les approches "par paires" et "par résidus". Considérer* $B = 500$ *et enregistrer les coefficients obtenus.*
   — *On pourra en profiter pour visualiser la variabilité obtenue en superposant les droites de régression.*
3. *Calculer les intervalles de confiance à* 95% *des coefficients du modèle par la méthode des quantiles.*
4. *Comparer ces intervalles avec ceux obtenus par l'approche paramétrique classique reposant sur un modèle gaussien. Comment interpréter ces résultats ?*
   — *On peut (par exemple) obtenir cet intervalle de confiance via la fonction* `confint()`*.*
5. *Proposer une représentation graphique permettant de résumer ces résultats.*
6. *Pour aller plus loin :*
   (a) *Appliquer une telle procédure bootstrap ("par paires" ou "par résidus", au choix) pour caractériser l'incertitude de prédiction obtenue pour une vitesse égale à 21.*
   (b) *Comparer la variabilité obtenue par cette procédure bootstrap aux intervalles de confiance proposés par la fonction* `predict`*, appliquée au modèle de régression linéaire global de la question 1.*
   — *Pour obtenir ces intervalles de confiance, on utilise la commande suivante :*
   `I = predict(global.fit, interval="confidence",`
   `level=0.95, newdata=data.frame(speed=0:30))`

Commencons par visualiser le jeu de données et le résultat d'une régression linéaire classique.

```
> ##############################
> ####### STARTING EXO 3 ######
> ##############################
> # show dataset and fit global linear model
> fit.global = lm(dist~speed, data = cars)
> plot(cars, pch = 19, main = "cars dataset - linear regression")
> abline(fit.global, col = "red", lwd = 2)
```



Réalisons ensuite une procédure de boostrap. A chaque répétition on procède ainsi :

— on génère un échantillon de la manière suivante :

  — pour l'appproche "par paires" : on tire n observations avec remise dans notre échantillon

  — pour l'approche "par résidus" : on tire n valeurs avec remise dans le vecteur des résidus obtenus par le modèle global, et on génère de nouvelles réponses pour chacune des observations en ajoutant ces résidus aux valeurs estimées par le modèle global.

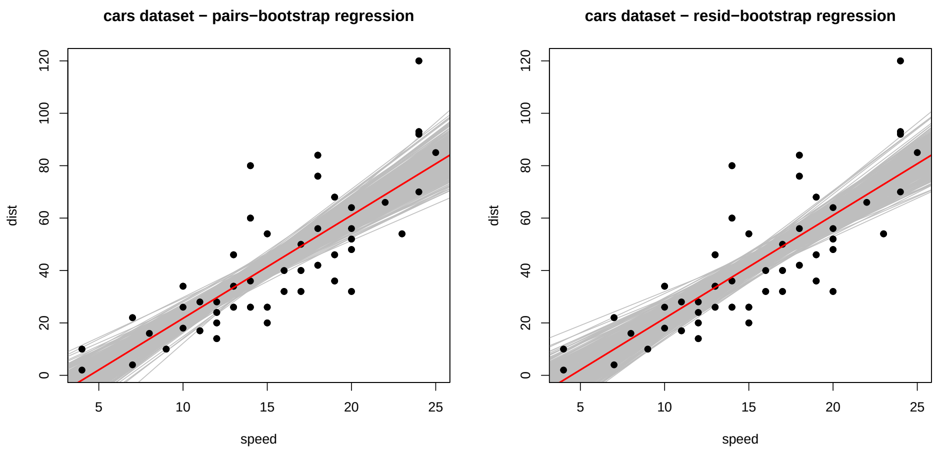
— on réalise une régression linéairee sur ces échantillons bootstrap

— on stocke les coefficients (intercept et pente) obtenus.

Ce faisant, on affiche (sur une même figure) les droites de régression obtenus pour visualiser la variabilité de l'estimation. La figure suivante représente les résultats obtenus.

```
> # carry out bootstrap procedure #
> #-------------------------------#
> B = 500
> n = nrow(cars)
> par(mfrow = c(1,2))
```

```
> # 1) from pairs
> plot(cars, main = "cars dataset - pairs-bootstrap regression")
> beta.pairs = c()
> for(b in 1:B){
+   # get model coefficients
+   ind = sample(c(1:n), n, replace = TRUE)
+   x = cars$speed[ind]
+   y = cars$dist[ind]
+   fit.b = lm(y~x)
+   # store coefficients
+   beta.pairs = rbind(beta.pairs, fit.b$coefficients)
+   # show fit for illustration purposes
+   abline(fit.b, col = "grey")
+ }
> points(cars, pch = 19)
> abline(fit.global, col = "red", lwd = 2)
> box()
> # 2) from residuals
> x = cars$speed
> resid = fit.global$residuals
> fitted = fit.global$fitted
> plot(cars, main = "cars dataset - resid-bootstrap regression")
> beta.resid = c()
> for(b in 1:B){
+   # pick residuals
+   ind = sample(c(1:n), n, replace = TRUE)
+   y = fitted + resid[ind]
+   # fit model
+   fit.b = lm(y~x)
+   # store coefficients
+   beta.resid = rbind(beta.resid, fit.b$coefficients)
+   # show fit for illustration purposes
+   abline(fit.b, col = "grey")
+ }
> points(cars, pch = 19)
> abline(fit.global, col = "red", lwd = 2)
> box()
```



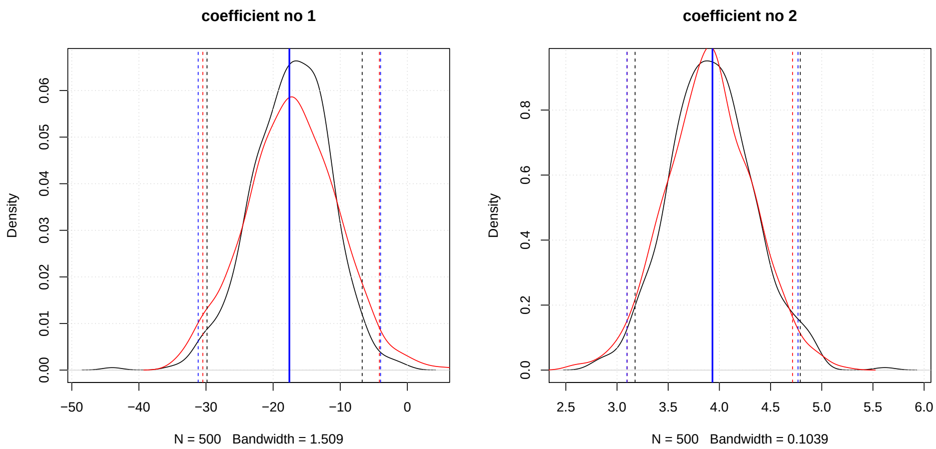
Une fois les B estimations bootstrap obtenues, on peut définir des intervalles de confiance de différentes manières, tel qu'évoqué en cours. Nous appliquons ici la procédure des quantiles.

```
> # get confidence intervals #
> #--------------------------#
> # boostrap approaches
> ci.pairs = apply(beta.pairs, 2, quantile, probs = c(0.025,0.975))
> ci.resid = apply(beta.resid, 2, quantile, probs = c(0.025,0.975))
> # get "standard" confidence intervals
> ci.lm = confint(fit.global, level = .95)
```

La figure suivante représente graphiquement les résultats pour l'intercept (à gauche) et la pente (à droite) :

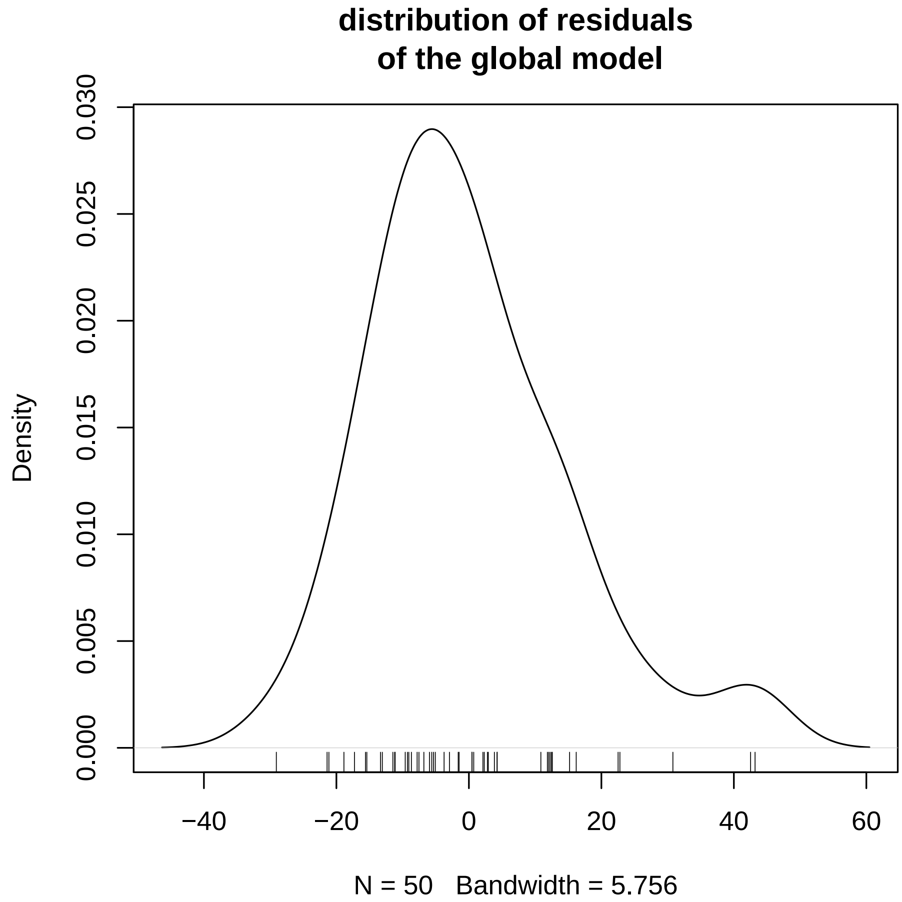
— les courbes noires et rouges représentent respectivement la distribution des estimations obtenues par bootstrap pour l'approche "par paires" et l'approche "par résidus".
— les courbes verticales en pointillés rouges et noirs représentent les intervalles de confiance correpondant.
— en bleu sont représentés l'estimation faite par le modèle global, et l'intervalle de confiance correspondant.

```
> par(mfrow = c(1,2))
> # intercept
> for(i in 1:2){
+   plot(density(beta.pairs[,i]), main = paste("coefficient no", i))
+   grid()
+   lines(density(beta.resid[,i]), col  = 2)
+   abline(v = fit.global$coefficients[i], col = "blue", lwd = 2)
+   abline(v = ci.pairs[,i], col = 1, lty = 2)
+   abline(v = ci.resid[,i], col = 2, lty = 2)
+   abline(v = ci.lm[i,], col = "blue", lty = 2)
+ }
```



On note (1) que les deux approche bootstrap donnent, sur cet exemple, des résultats très similaires, et (2) que les résultats sont également très proches de l'approche "standard" reposant sur un modèle de résidus gaussiens. Ce n'est pas surprenant car si on regarde la distribution des résidus, on note qu'elle est effectivement plutôt gaussienne (avec néanmoins une queue de distribution légèrement plus forte dans les valeurs positives). Ce modèle paramétrique est donc tout à fait adapté ici.

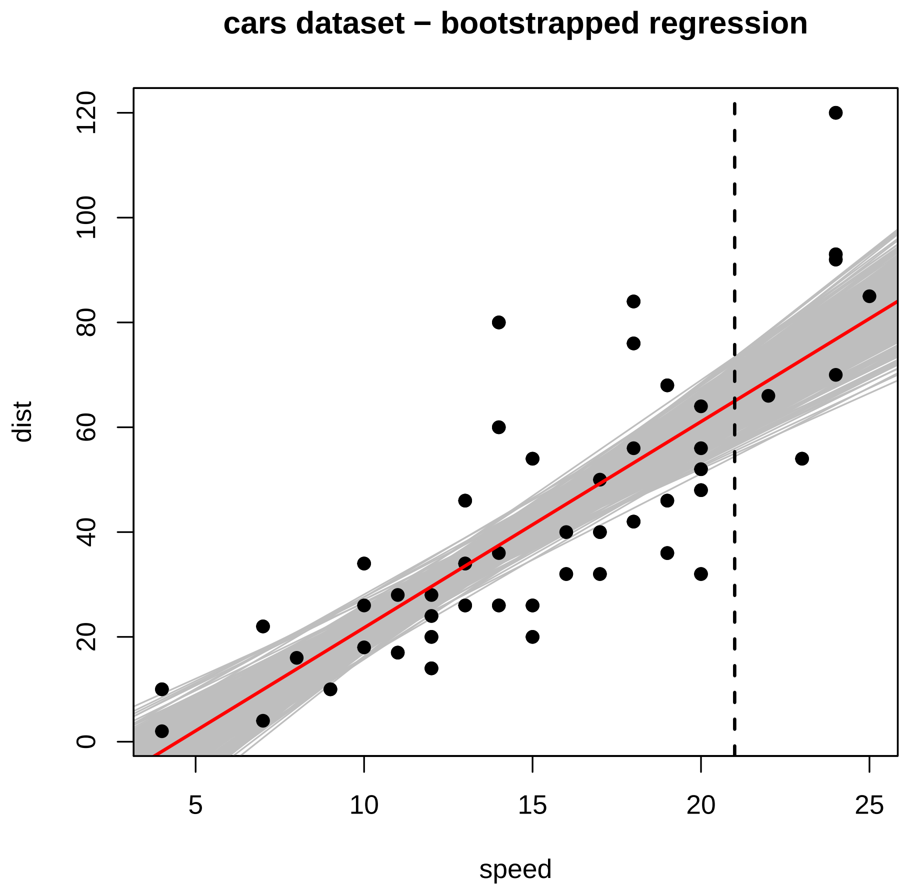
```
> plot(density(resid), main = "distribution of residuals\n of the global model")
> rug(resid)
```



Enfin, on peut appliquer une procédure bootstrap pour caractériser l'incertitude de prédiction en un point donné, à l'instar d'une procédure de "bagging" (si on applique un bootstrap "par paires"). Le code ci-dessous réalise cette analyse pour une vitesse égale à 21.

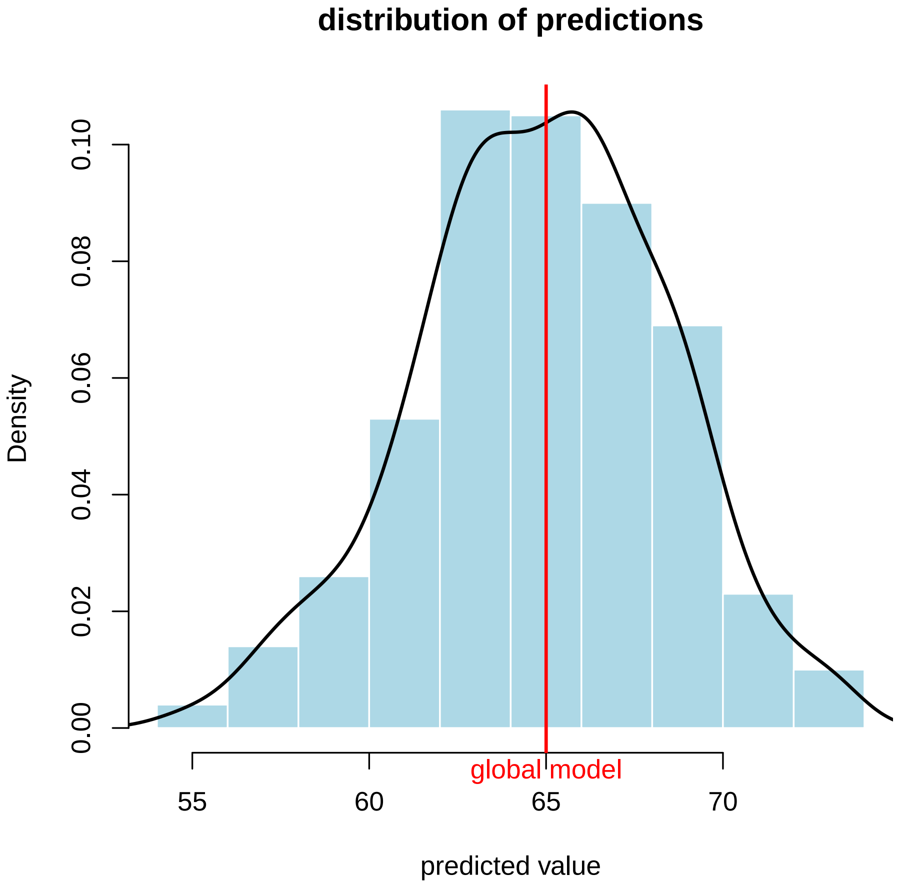
```
> # boostrap  procedure
> B = 500
> x.test = 21
> preds = numeric(B)
> n = nrow(cars)
> plot(cars, main = "cars dataset - bootstrapped regression")
> for(b in 1:B){
+   ind = sample(c(1:n), n, replace = TRUE)
+   x = cars$speed[ind]
+   y = cars$dist[ind]
+   fit.b = lm(y~x)
+   abline(fit.b, col = "grey")
+   # compute prediction
```

```
+   preds[b] = fit.b$coefficients[1] + x.test*fit.b$coefficients[2]
+ }
> points(cars, pch = 19)
> abline(fit.global, col = "red", lwd = 2)
> box()
> abline(v = x.test, lty = 2, lwd = 2)
> # compute global prediction
> pred.global = fit.global$coefficients[1] + x.test*fit.global$coefficients[2]
```



On peut alors visualiser la variabilité des prédictions obtenues pour une vitesse de 21 via un histogramme et/ou une densité :

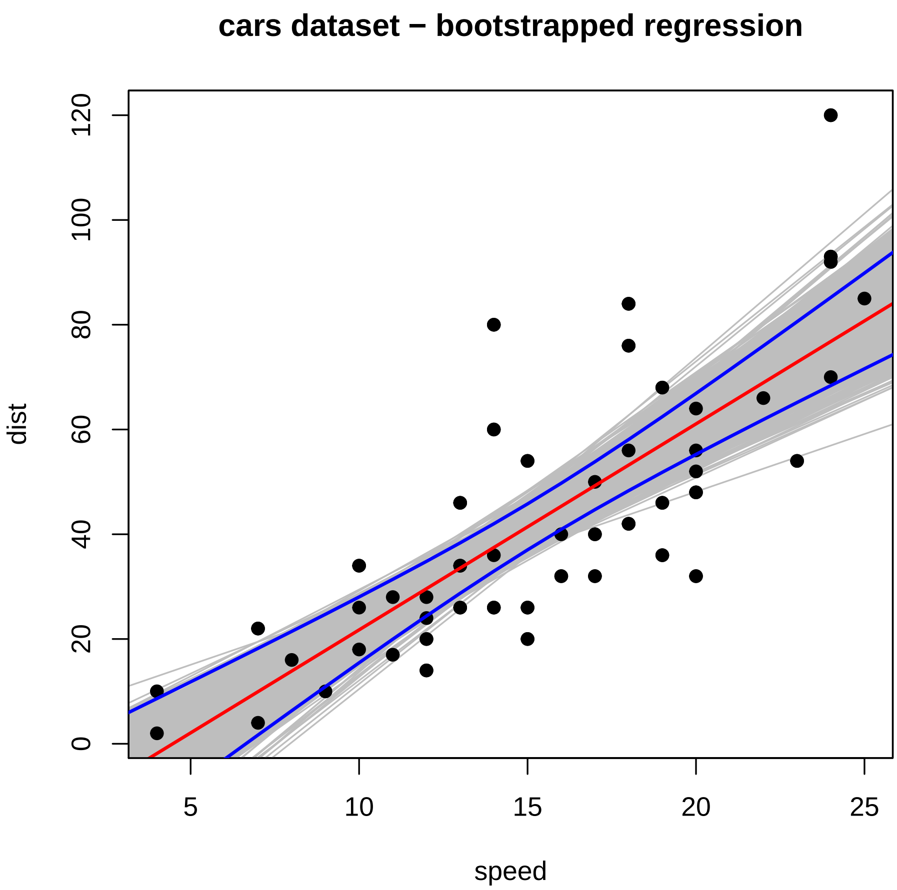
```
> hist(preds, prob = TRUE, main = "distribution of predictions", xlab = "predicted value", c
> lines(density(preds), lwd = 2)
> abline(v = pred.global, col = 2, lwd = 2)
> mtext("global model",  side = 1, at = pred.global, col = 2)
```



Enfin, on peut comparer la variabilité des résultats obtenus par bootstrap aux intervalles de confiance proposés par la fonction `predict.lm`.

```
> # same procedure with standard CI
> B = 2000
> x.test = 21
> n = nrow(cars)
> plot(cars, main = "cars dataset - bootstrapped regression")
> for(b in 1:B){
+   ind = sample(c(1:n), n, replace = TRUE)
+   x = cars$speed[ind]
+   y = cars$dist[ind]
+   fit.b = lm(y~x)
+   abline(fit.b, col = "grey")
+ }
> points(cars, pch = 19)
> abline(fit.global, col = "red", lwd = 2)
```

```
> box()
> I = predict(fit.global, interval = "confidence", level = 0.95, newdata = data.frame(speed=
> lines(0:30,I[,2], col = "blue", lwd = 2)
> lines(0:30,I[,3], col = "blue", lwd = 2)
```



Pour aller plus loin, on pourrait effectuer plus finement cette comparaison en un point donné, en comparant ces intervalles de confiance et les quantiles d'ordre $\alpha/2$ et $1-\alpha/2$ de la distribution de prédictions obtenues par bootstrap.