Statistiques non-paramétriques
Ch. 2. Bootstrap 2018-19
M2 CEE

Pr. Philippe Polomé, Université Lumière Lyon 2

Sommaire

Motivation

- En régression np, on n'a pas
  - d'hyp. de normalité
  - de coefficient dont on pourrait tester la significativité
- Par contre, il faut pouvoir inférer
  - L'effet d'un régresseur x sur une dépendante y, $\partial y/\partial x$, est-il significatif ?
    - Peut-on calculer un IC ?
  - Toute autre hyp moins simple
- Dans les régressions np au ch suivant, bootstrap est la seule technique d'inférence

Bootstrap

- Hyp. "**Principe de médiocrité**" :
  - Si on pouvait **ré-échantillonner** la pop.
    - dans les mêmes conditions
    - on obtiendrait un échantillon semblable à celui qu'on a déjà
  - On ne peut pas ré-échantillonner
  - Pas la même chose que représentativité
- Principe : Traiter l'éch. comme une pop.
  - Échantillonner l'échantillon original **avec remplacement**
  - Si on a une taille n au départ, on fait n tirages avec remplacement
    - Chaque i a une probabilité $1/n$ de sortir à chaque tirage
    - On obtient un **échantillon Bootstrap** (de taille n)
    - ou **pseudo-échantillon**
    - Certaines obs. sont tirées pls fois, d'autres aucune
    - **Hyp**. : ce pseudo-échantillon est semblable à ce qu'on aurait obtenu en ré-échantillonnant la pop.

Exemple notionel

- Dans un tableur, plus intuitif
- Soit un éch. de taille n d'une variable z
  - On numérote chaque obs. de 1 à n
  - On crée une plage de n lignes x 1 col
    - Chaque cellule est remplie de la fonction randbetween(1 ;n)
    - Tire un naturel entre 1 et n
  - On obtient n chiffres entre 1 et n
    - certains sont répétés
    - d'autres absents
  - un pseudo-échantillon est généré en
    - créant une nouvelle plage de n lignes x 1 col
    - dans chaque cellule on associe l'observation original au chiffre trouvé

Exemple

Echantillon original			Bootstrap # 1		Bootstrap # 2	
nbr	z		randbetween	z	randbetween	z
1	13.3		17	5	10	7.2
2	17.4		4	13.8	24	7.5
3	6.7		8	10.3	29	18
4	13.8		15	3.1	10	7.2
5	7.3		20	10.1	19	5.1
6	12.5		20	10.1	16	11.7
7	8.2		10	7.2	2	17.4
8	10.3		5	7.3	17	5
9	8.9		31	12	23	5.7

Fichier “bootstrap.ods” sur page web

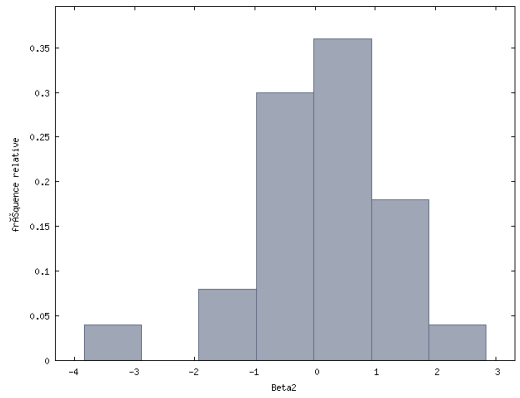
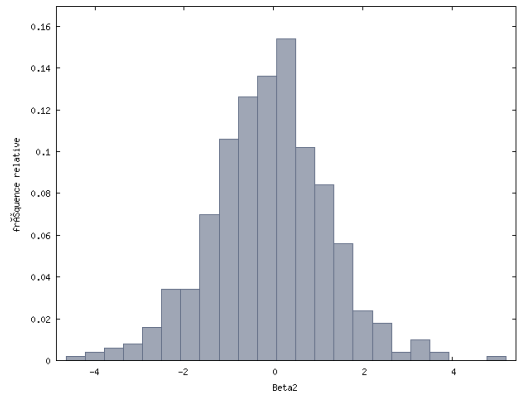
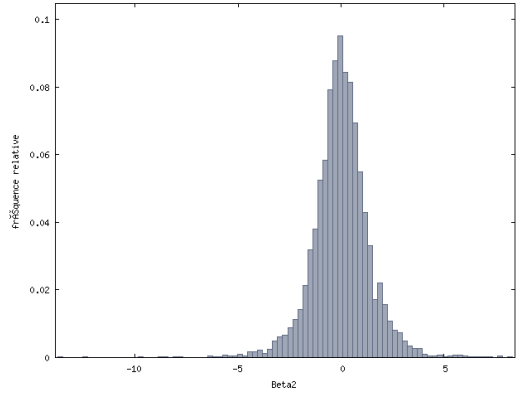
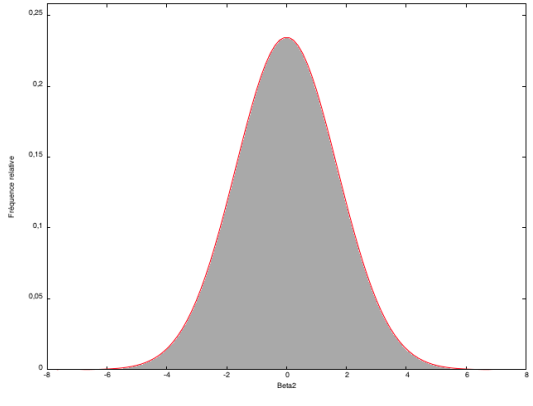
Exemple MCO

- Échantillon original $< Y, X >$, Xpeut être une matrice
- B **pseudo-échantillons** différents $< Y_b, X_b >$
  - S'appelle "Bootstrap par paire" car y et X sont tirés ensemble
  - N'est pas la seule façon
    - p.e. on peut se baser sur les résidus
- Pour chaque pseudo-échantillon
  - On prend le MRL $Y = X\beta + \epsilon$
    - On estime dans chaque pseudo-échantillon

$$\hat{\beta}_b = \left(X_b' X_b\right)^{-1} X_b' Y_b$$

  - Au total, une matrice $\hat{\beta}_B$ de B vecteurs estimées $\hat{\beta}_b$

- On prend un élément de β, soit β_i
  - On a B estimations $\hat{\beta}_{ib}$, on les mets en histogramme

50 échantillons	500 échantillons
	
5000 échantillons	**∞ échantillons**
	

Exemple MCO et distribution empirique

- La distribution empirique est représentée par les histogrammes
  - Lorsque $B \to \infty$, et que les hypothèses bootstrap sont vérifiées
    - alors la distribution empirique converge à la vraie distribution
    - qui est normale pour un $\hat{\beta}_{MCO}$ (quand n est grand), mais ça n'est pas vrai pour p.e. des stats de test
- Deux moments de cette distribution empirique "bootstrap"
  - $\bar{\hat{\beta}}_i = \frac{1}{B}\sum_{b=1}^{B}\hat{\beta}_{ib} = \widehat{E\left(\hat{\beta}_{ib}\right)} \approx \hat{\beta}_i$
  - $\widehat{var\left(\hat{\beta}_{ib}\right)} = \frac{1}{B-1}\sum_{b=1}^{B}\left(\hat{\beta}_{ib} - \bar{\hat{\beta}}_i\right)^2$
- Intervalle de confiance, p.e. avec $B = 1000$
  - On ordonne ces 1000 estimations de la plus petite à la plus grande
  - Alors les estimations numéro 25 et 975 sont les bornes inf et sup, respectivement, de l'intervalle de confiance à 95% de β

Pourquoi est-ce intéressant ?

1. Pas d'hypothèse sur la distribution des erreurs
   - 1.1 Mais il ne peut y avoir de corrélation entre observations
     - Le ré-échantillonnage casserait cette corrélation
     - En panels, on ré-échantillonne seulement sur i en utilisant **toutes** les T périodes de chaque i sélectionné
   - 1.2 Le bootstrap par paires $< Y_b, X_b >$ devrait produire des écarts types robustes à l'hétéroscédasticité
2. On peut calculer des intervales de confiance
   - 2.1 pour toute fonction des paramètres estimés, y-compris non-linéaire
   - 2.2 pour des paramètres estimés de modèles sans propriétés d'échantillons finis connues
     - comme semi-np

Sommaire

Exemple du package AER, *Journals*

- On veut calculer des écarts-types & des intervales de confiance
- data("Journals")
- journals <- Journals[, c("subs", "price")]
- journals$citeprice <- Journals$price/Journals$citations
- jour_lm <- lm(log(subs) ~ log(citeprice), data = journals)
- Voir commandes bootstrap dans np2018.R

La commande boot( )

- Le bootstrap dans R
  - utilise la commande boot( ) du package boot
  - Elle accepte pls arguments,
    - desquels 3 sont requis :
- Data : les données
- Statistic : une fonction (*function*) à définir qui renvoie la stat à "bootstrapper"
  - avec 2 arguments
    - Les données (une nouvelle fois !)
  - Un vecteur index qui donne les indices des obs à inclure dans l'échantillon bootstrap
- R : le nombre de réplications
  - B dans la présentation théorique

Construction de la fonction pour l'arg. "statistic" de boot( )

refit <- function(data, i)

- Dans une ***function***, il ne faut qu'énoncer les arguments
  - Ce que la fonction fait est décrit en dessous
  - refit est définie juste pour les besoins de boot( )
    - sur les données et sur un index i des données
- coef(lm(log(subs) ~ log(citeprice), data = data[i,]))
  - Ici refit renvoie les coefficients MCO de log(subs) ~ log(citeprice)
    - et utilise comme index *i* le num. de la ligne des données
    - pas MCO sur la ligne *i*
  - Donc, la fonction boot( ) prend *i* comme index du bootstrap
    - Dans chaque réplication bootstrap, un nouvel éch. extrait des lignes de *data*

Tout ça n'est pas très intuitif,

- mais c'est le format de boot( )

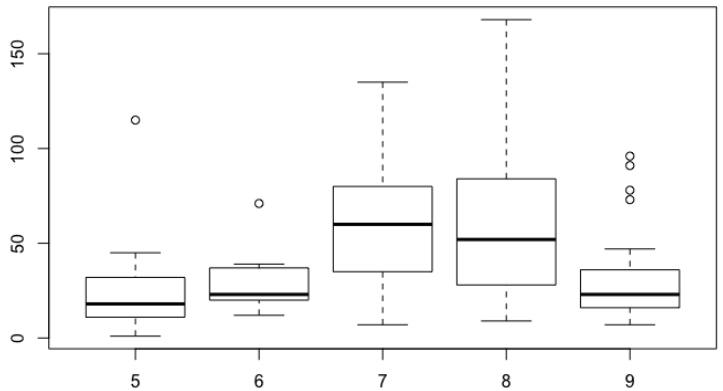
Appeler boot( )

- library("boot")
  - boot est le package recommandé pour le bootstrap
- set.seed(123)
- jour_boot <- boot(journals, refit, R = 999)
  - boot : 3 arg – données, fonction, B
- jour_boot montre comme résultats :
  - Les coef du lm original
  - La différence avec la moyenne des coeff issus du bootstrap
    - appelé “bias”
  - Les écarts-types de ces derniers
    - Pour calculer des t-stats bootstrappés on peut extraire les t-stats de lm plutôt que les coef
    - voir np2017.R
    - Qu’est-ce qui vaut mieux ?
- Peu de différence avec la sortie standard coeftest(jour_lm)

Autres éléments de boot

- t0 est la stat originale sur laquelle on réalise le bootstrap
  - jour_boot$t0 pour l'extraire
- t l'ensemble des R stat bootstrappées
  - Ici, intercept (col 1) et pente (col 2)
  - On peut calculer directement un IC à partir de t avec la fonction quantile
    - quantile(jour_boot$t[,2], probs = c(.025,.975))
    - on prend [,2] pour le 2nd élément des coef = pente de la régression
- Pour calculer un IC des t-stat, voir np2017.R
- Exercice : IC du R^2 de cette régression
  - Le R^2 peut être extrait de summary

Exemple test de Kruskal Walis



- kruskal.test(Ozone ~ Month, data = airquality)
- retest <- function(data, i)
  - (kruskal.test(Ozone ~ Month, data = data[i,]))$statistic
- air_boot <- boot(airquality, retest, R = 999)
- Exercice :
  - Donnez un IC pour la p-valeur en utilisant la fonction boot.ci

Sommaire

Commande sample

- sample(x, size, replace = FALSE)
- Permet de prendre un sous-échantillon de x de taille size
  - avec ou sans remplacement
- En prenant un sous-éch. de la même taille que l'éch.
  - avec remplacement
  - on fait un éch. bootstrap
- En mettant ça dans une boucle R fois
  - et en pratiquant l'analyse désirée à chaque fois
  - on obtient une alternative à la commande boot
- Quel intérêt ? permet de spécifier des groupes
  - x est soit une variable dans le dataframe
  - ou un index compatible
    - p.e. en panel, on pourrait avoir i=pays et t=année
    - on veut faire un bootstrap sur les seuls pays

Bootstrap avec sample

- package plm
- Estimation panel effet fixe
  - grun.fe <- plm(inv~value+capital, data = Grunfeld, model = "within")
  - En panel, svt het. & autocor. => calculer le t-stat par bootstrap
- Ensuite, il faut définir une "function" pour définir un éch. bootstrap
  - Puis une boucle "for" pour appeler cette function R fois

FUN

- myfunc <- function(n,df) {
  - définit function (n et df sont dans l'appel de la fonction + bas)
    - n est Grunfeld$firm lors de l'appel de myfunc
    - df est le dataframe
  - Grunfeld unique_firm <- unique(n)
    - firme unique (help : unique renvoie un vecteur ou data frame comme son arg x mais en enlevant les lignes doublons, dans ce cas définies sur firm)
  - sample_firm <- sample(unique_firm, size=length(unique_firm), replace=TRUE)
    - choisit sur firm unique, au hasard avec remplacement, de la taille de "size" (help sample)
  - new_df <- do.call(rbind, lapply(sample_firm, function(x) df[df$firm==x,]))
    - va chercher toutes les années de chaque firme choisie aléatoirement et rbind c'est-à-dire empile en ligne
  - }

Bootstrap avec sample

- do.call
  - construit & exécute une fonction : do.call(what, args)
    - what : soit une fonction ou une chaîne de caractères nommant une fonction à appeler - rbind "row bind" lier des lignes
    - args : lapply(X, FUN, ...) : Appliquer une Function FUN sur un vecteur ou une **L**iste X et renvoie une liste de même longueur que X, chaque élément de cette liste est le résultat d'appliquer FUN à l'élément correspondant de X
    - Donc création d'une 2ème function (non-nommée) à l'intérieur de lapply qui va chercher les lignes correspondantes aux firmes dans "sample_firm"
- a <- myfunc(Grunfeld$firm, Grunfeld)
  - exécute la function, on voit que le résultat a la même taille que Grunfeld
  - Mais certaines firmes sont dupliquées, d'où message d'erreur de plm
  - pour éviter ca, re-créer un index des firmes
    - On sait qu'on a 20 obs par firme, 10 firmes
    - On va créer une séquence a$E <- rep(1 :10, each = 20)
    - On teste : coef(plm(inv~value+capital, data = a, index = c("E","year"), model = "within")) ok !

Boucle for de 1 à R

- R=99
- c <- coef(plm(inv~value+capital, data = Grunfeld, model = "within"))
  - Estimation initiale (peut être part du bootstrap), coef est un vector
- for(i in 1 :R) {
  - a <- myfunc(Grunfeld$firm, Grunfeld) # À chaque i on relance le bootstrap & les estimations
  - a$E <- rep(1 :10, each = 20)
  - c <- cbind(c,coef(plm(inv~value+capital, data = a, index = c("E","year"), model = "within")))
    - crée une matrice avec tous les 1ers coefs sur la 1ère ligne, tous les 2nds sur la seconde et ainsi de suite pour l'ensemble des coef du modele (ici 2)
  - }

Sorties

- colMeans(t(c))
  - moyenne sur chaque coef
- sqrt(apply(t(c), MARGIN = 2, var))
  - écarts-types par coef
  - MARGIN = 2 indique à apply de calculer var sur les col de t(c) (transposée de c)
    - MARGIN = 1 indique les rows
- apply(t(c), MARGIN = 2, quantile, probs=c(.025, .975))
  - 95% CI over coef value

Exercice. Répliquer cet exemple en groupant par year au lieu de firm

Devoir # 2

- Les 3 exercices
  1. IC du R^2 de la régression de *journals*
  2. Kruskal-Walis : IC pour la p-valeur en utilisant la fonction boot.ci
  3. Bootstrap avec la cmd sample, Grunfeld data, group by year