STT3030 - Cours #4

Arthur Charpentier

Automne 2024

Les fondations

Dans cette première partie du cours, il est question de techniques de base en **apprentissage supervisé** ("*supervised learning*").

Il s'agit du problème de prédire/estimer la relation entre deux ensembles de variables.

On veut prédire la **réponse** Y avec les **prédicteurs** $\mathbf{X}$.

$\mathbf{X} = \{X_1, ..., X_p\}$ forment une collection de prédicteurs, variables explicatives, covariables, entrées, etc.. (*predictors, features or input*).

On dit que $\mathbf{X} \in \mathcal{X}_1 \times ... \times \mathcal{X}_p = \mathcal{X}$, où $\mathcal{X}_j$ est le domaine de X_j.

Ce sont les variables que nous utilisons pour prédire.

$Y \in \mathcal{Y}$ est une réponse, variable expliquée, étiquette (si catégoriel) ou tout simplement la "sortie".

C'est ce que nous cherchons à prédire (à l'aide des prédicteurs).

Supposons qu'il s'agit d'une variable continue pour l'instant.

Processus génératif et données d'entraimenet

Comment les données sont-elles générées ? (dans la **nature**, dans le vrai monde) On suppose l'existence d'une fonction $Y = f(\boldsymbol{X}) + \varepsilon$, où ε est une supposée variabilité inexpliquée (erreur de mesure, variabilité naturelle, etc).

En apprentissage supervisé, on veut estimé f pour faire de la prédiction: $\widehat{Y} = \widehat{f}(\boldsymbol{X})$.

Nous avons un ensemble de données

$$\mathcal{D}_n = \{(\boldsymbol{x}_i, y_i) | i \in (1, ..., n)\} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_n, y_n)\}$$

C'est grâce à ces données que nous allons **entrainer** notre modèle, le valider et le tester.

C'est notre **échantillon**. On sait comment apprendre des modèles linéaires pour plusieurs domaine de prédicteurs $\mathcal{X}$ et de réponse $\mathcal{Y}$.

La réponse peut être continue ou catégorielle.

On peut aussi inclure des termes polynomiaux et d'intéractions dans les prédicteurs.

Sélection et régularisation: Pourquoi ?

- Quand $n \gg p$ il n'y a pas trop de problème.
- Lorsqu'on veut intégrer beaucoup de prédicteurs et que p grossit plusieurs problèmes apparaissent.
- Une grande instabilité s'installe. (Mauvaise prédictions, mauvaise inférence)
- On fait du surraprentissage. (Mauvaise prédictions, mauvaise inférence)
- Le modèle devient très difficile à interpreter. (Mauvaise inférence)
- Lorsque $p > n$ le modèle brise, pas de solution exacte et unique.
  (On ne peut calculer plus $(\boldsymbol{X}^\top \boldsymbol{X})^{-1}$)

Les régressions Ridge et Lasso

Si on dispose d'une seule variable explicative x,

$$\min_{\beta_0,\beta_1}\left\{\frac{1}{n}\sum_{i=1}^{n}(y_i-\beta_0-\beta_1 x_i)^2\right\} \text{ : Régression par moindres carrés}$$

$$\min_{\beta_0,\beta_1}\left\{\frac{1}{n}\sum_{i=1}^{n}(y_i-\beta_0-\beta_1 x_i)^2+\lambda\beta_1^2\right\} \text{ : Ridge}$$

$$\min_{\beta_0,\beta_1}\left\{\frac{1}{n}\sum_{i=1}^{n}(y_i-\beta_0-\beta_1 x_i)^2+\lambda|\beta_1|\right\} \text{ : Lasso}$$

Moindres carrés, $\text{SCR}=\sum_{i=1}^{n}(y_i-\beta_0-\beta_1 x_i)^2$

$$\frac{\partial \text{SCR}}{\partial \beta_0}=-2\sum_{i=1}^{n}(y_i-\beta_0-\beta_1 x_i) \text{ et } \frac{\partial \text{SCR}}{\partial \beta_1}=-2\sum_{i=1}^{n}x_i(y_i-\beta_0-\beta_1 x_i)$$

Les régressions Ridge et Lasso

i.e.

$$\mathbf{1}^\top \boldsymbol{y} - \widehat{\beta}_0 \mathbf{1}^\top \mathbf{1} - \widehat{\beta}_1 \mathbf{1}^\top \boldsymbol{x} = 0 \text{ et } \boldsymbol{x}^\top \boldsymbol{y} - \widehat{\beta}_0 \boldsymbol{x}^\top \mathbf{1} - \widehat{\beta}_1 \boldsymbol{x}^\top \boldsymbol{x} = 0$$

$$\overline{y} - \widehat{\beta}_0 - \widehat{\beta}_1 \overline{x} = 0 \text{ et } \widehat{\beta}_1 = \frac{(\boldsymbol{x} - \overline{x})^\top (\boldsymbol{y} - \overline{y})}{(\boldsymbol{x} - \overline{x})^\top (\boldsymbol{x} - \overline{x})}$$

Si on considère des variables centrées, $\overline{y} = \overline{x} = 0$

$$\min_\beta \left\{ \frac{1}{n} \sum_{i=1}^n (y_i - \beta x_i)^2 \right\} \text{ : Régression par moindres carrés}$$

$$\min_\beta \left\{ \frac{1}{n} \sum_{i=1}^n (y_i - \beta x_i)^2 + \lambda \beta^2 \right\} \text{ : Ridge}$$

$$\min_\beta \left\{ \frac{1}{n} \sum_{i=1}^n (y_i - \beta x_i)^2 + \lambda |\beta| \right\} \text{ : Lasso}$$

Les régressions Ridge et Lasso

Moindres carrés, $\text{SCR} = \sum_{i=1}^{n}(y_i - \beta x_i)^2 = \boldsymbol{y}^\top \boldsymbol{y} - 2\boldsymbol{x}^\top \boldsymbol{y}\beta + \beta \boldsymbol{x}^\top \boldsymbol{x}\beta$

$$\frac{\partial \text{SCR}}{\partial \beta} = -2\sum_{i=1}^{n} x_i(y_i - \beta x_i) = -2\boldsymbol{x}^\top(\boldsymbol{y} - \beta \boldsymbol{x}) \text{ and } \widehat{\beta}^{\text{ols}} = \frac{\boldsymbol{x}^\top \boldsymbol{y}}{\boldsymbol{x}^\top \boldsymbol{x}} = (\boldsymbol{x}^\top \boldsymbol{x})^{-1}\boldsymbol{x}^\top \boldsymbol{y}$$

Ridge, $\text{PSCR} = \boldsymbol{y}^\top \boldsymbol{y} - 2\boldsymbol{x}^\top \boldsymbol{y}\beta + \beta \boldsymbol{x}^\top \boldsymbol{x}\beta + \lambda\beta^2$

$$\frac{\partial \text{PSCR}}{\partial \beta} = -2\boldsymbol{y}^\top \boldsymbol{x} + 2\boldsymbol{x}^\top \boldsymbol{x}\beta + 2\beta\lambda \text{ and } \widehat{\beta}_\lambda^{\text{ridge}} = \frac{\boldsymbol{x}^\top \boldsymbol{y}}{\boldsymbol{x}^\top \boldsymbol{x}+\lambda} = (\boldsymbol{x}^\top \boldsymbol{x}+\lambda)^{-1}\boldsymbol{x}^\top \boldsymbol{y}$$

Lasso, $\text{PSCR} = \boldsymbol{y}^\top \boldsymbol{y} - 2\boldsymbol{x}^\top \boldsymbol{y}\beta + \beta \boldsymbol{x}^\top \boldsymbol{x}\beta + \lambda|\beta|$

$$\frac{\partial \text{PSCR}}{\partial \beta} = -2\boldsymbol{y}^\top \boldsymbol{x} + 2\boldsymbol{x}^\top \boldsymbol{x}\widehat{\beta} \pm \lambda$$

Les régressions Ridge et Lasso

où le signe de $\pm$ est le signe de $\widehat{\beta}$.

Supposons que l'estimateur par moindres carrés (obtenu quand $\lambda = 0$) est (strictement) positif, i.e. $\boldsymbol{x}^\top \boldsymbol{y} > 0$.

Si λ n'est pas trop grand $\widehat{\beta}_\lambda$ et $\widehat{\beta}^{\text{ols}}$ sont de même signe, et

$$-2\boldsymbol{y}^\top \boldsymbol{x} + 2\boldsymbol{x}^\top \boldsymbol{x}\widehat{\beta} + \lambda = 0, \text{ soit } \widehat{\beta}_\lambda^{\text{lasso}} = \frac{\boldsymbol{y}^\top \boldsymbol{x} - \lambda/2}{\boldsymbol{x}^\top \boldsymbol{x}}$$

On augmente alors λ jusqu'à avoir $\widehat{\beta}_\lambda = 0$.

En augmentant encore un peu $\widehat{\beta}_\lambda$ ne peut pas devenir négatif, et donc

$$-2\boldsymbol{y}^\top \boldsymbol{x} + 2\boldsymbol{x}^\top \boldsymbol{x}\widehat{\beta} - \lambda = 0, \text{ soit } \widehat{\beta}_\lambda^{\text{lasso}} = \frac{\boldsymbol{y}^\top \boldsymbol{x} + \lambda/2}{\boldsymbol{x}^\top \boldsymbol{x}}$$

Mais cette solution est positive ($\boldsymbol{y}^\top \boldsymbol{x} > 0$) donc on doit avoir $\widehat{\beta}_\lambda < 0$.

Les régressions Ridge et Lasso

La seule solution possible est alors $\widehat{\beta}_\lambda = 0$.
Si x est centrée ($\bar{x} = 0$) et réduite ($\boldsymbol{x}^\top \boldsymbol{x} = n$),

$$\widehat{\beta}^{\text{ols}} = \frac{1}{n} \boldsymbol{x}^\top \boldsymbol{y}$$

$$\widehat{\beta}_\lambda^{\text{ridge}} = \frac{\boldsymbol{x}^\top \boldsymbol{y}}{n+\lambda} = \frac{n}{n+\lambda} \cdot \frac{1}{n} \boldsymbol{x}^\top \boldsymbol{y} = \frac{n}{n+\lambda} \cdot \widehat{\beta}^{\text{ols}}$$

$$\widehat{\beta}_\lambda^{\text{lasso}} = \frac{\boldsymbol{x}^\top \boldsymbol{y} \pm \lambda/2}{n} = \frac{1}{n} \boldsymbol{x}^\top \boldsymbol{y} \pm \frac{\lambda}{2n} = \widehat{\beta}^{\text{ols}} \pm \frac{\lambda}{2n}$$

Ridge Regression

La fonction cible pour la **régression Ridge** s'écrit

$$\mathcal{L}_\lambda(\boldsymbol{\beta}) = \sum_{i=1}^{n}(y_i - \beta_0 - \boldsymbol{x}_i^\top\boldsymbol{\beta})^2 + \lambda\sum_{j=1}^{p}\beta_j^2$$

$$\frac{\partial\mathcal{L}_\lambda(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}} = -2\boldsymbol{X}^\top\boldsymbol{y} + 2(\boldsymbol{X}^\top\boldsymbol{X} + \lambda\mathbb{I})\boldsymbol{\beta}$$

$$\frac{\partial^2\mathcal{L}_\lambda(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^\top} = 2(\boldsymbol{X}^\top\boldsymbol{X} + \lambda\mathbb{I})$$

où $\boldsymbol{X}^\top\boldsymbol{X}$ est une matrice semi-positive définie, et $\lambda\mathbb{I}$ est une matrice définie positive, et

$$\widehat{\boldsymbol{\beta}}_\lambda^{\text{ridge}} = (\boldsymbol{X}^\top\boldsymbol{X} + \lambda\mathbb{I})^{-1}\boldsymbol{X}^\top\boldsymbol{y}$$

La régression Ridge

La **régression Ridge** est la solution du problème de minimisation suivant:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{i,j})^2 + \lambda\sum_{j=1}^{p}\beta_j^2$$

Plus λ est grand, plus on pénalise fort de grands coefficients β donc plus on aura une solution simple, près de $\widehat{y} = \bar{y}$, avec de petits coefficients β.

Plus λ est petit, plus nous sommes près de la régression standard (moindres carrés). Si $\lambda = 0$ alors nous retrouve exactement la régression standard.

λ mène a un meilleur compromis biais-variance, en réduisant la variance. A vrai dire, λ contrôle directement ce compromis.

La régression Lasso

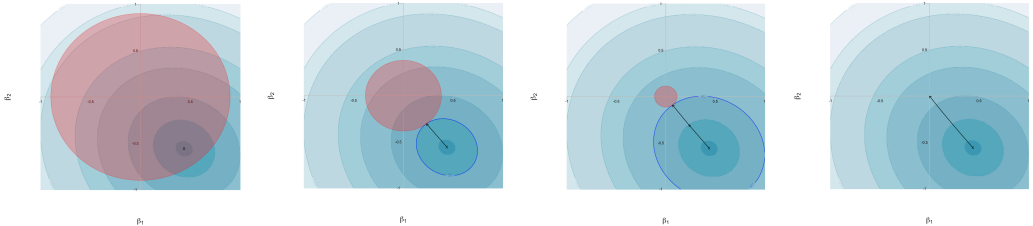
La **régression Lasso** (pour **least absolute shrinkage and selection operator**) est la solution du problème de minimisation suivant:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{i,j})^2+\lambda\sum_{j=1}^{p}|\beta_j|$$

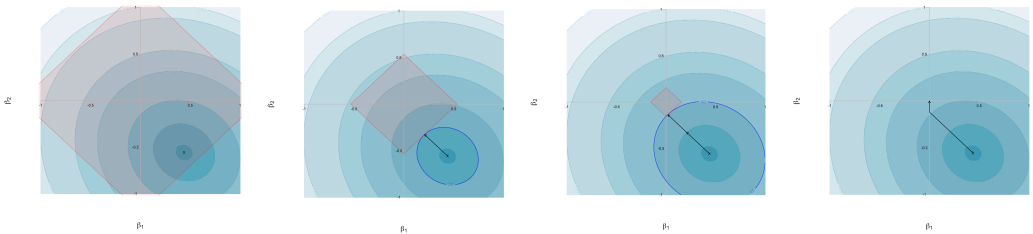
- Cette différence de pénalisation/régularisation pousse certains coefficients à 0.
- Ça nous permet donc de faire la sélection de variables ET l'estimation de coefficients en simultané pour un λ fixé.

Lasso et Ridge

- régression Ridge



- régression Lasso



Lasso et Ridge: comment choisir λ ?

- Plus λ est petit, plus nous somme près de la régression standard. Si $\lambda = 0$ alors nous retrouve la modèle complet; la régression standard.
- Une question qui découle naturellement de ce que nous venons de faire est: comment choisir l'hyper-paramètre λ ?
- Ça ressemble à k dans le modèle du plus proche voisinage.
- Ou bien le degré d en régression polynomial.
- Ces paramètres sont des boutons d'ajustement biais-variance!

Paramètres et hyperparamètres.

- Jusqu'à présent, on a parlé de paramètres et d'hyperparamètres.
- On dit qu'on apprend les paramètres avec les données d'entrainement
- et qu'on choisi les hyper-paramètres, disons λ pour la ridge/lasso, avec les donées tests.
- Donc en quelque sorte... on *apprend* les hyperparamètres ?
- Mais apprendre sur les données tests, c'est tricher!

C'est quoi la différence entre paramètres et hyperparamètres ?

- On dit que les paramètres (du modèle) sont appris lors de l'apprentissage
- alors que les hyperparametres contrôlent la forme du modèle et le processus d'apprentissage

Ensemble de test

En classe, j'ai beaucoup parler d'ensemble test et entrainement, mais nous n'avons pas parlé de comment gérer notre échantillon $\mathcal{D}_n$ rigoureusement.

Comme vous le savez maintenant, si on entraine un modèle flexible sur l'échantillon $\mathcal{D}_n$ et puis qu'on le test sur $\mathcal{D}_n$ il performera bien (c'est fait pour).

Par contre, il se peut que ce modèle ne performe pas bien sur de toutes nouvelles observations.

On apelle ce phénomène le **surrapprentissage**.

Ensemble de test

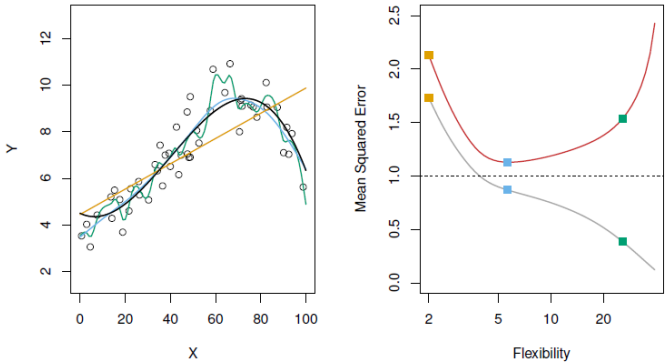


FIGURE 2.9. Left: *Data simulated from f, shown in black. Three estimates of f are shown: the linear regression line (orange curve), and two smoothing spline fits (blue and green curves).* Right: *Training MSE (grey curve), test MSE (red curve), and minimum possible test MSE over all methods (dashed line). Squares represent the training and test MSEs for the three fits shown in the left-hand panel.*

Figure 1: Figure 2.9 du livre James et al. (2013)

Ensemble de test

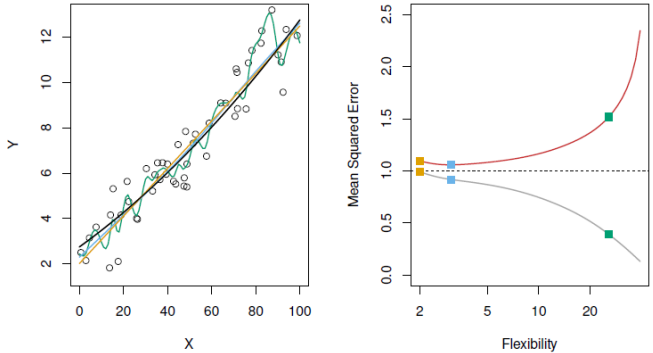


FIGURE 2.10. *Details are as in Figure 2.9, using a different true f that is much closer to linear. In this setting, linear regression provides a very good fit to the data.*

Figure 2: Figure 2.10 du livre James et al. (2013)

Ensemble de test

Comme il est couteux d'échantillonner $\mathcal{D}_n$ tout d'abord et puis de refaire une collecte de données pour obtenir un échantillon test, il est commun de diviser $\mathcal{D}_n$ en plusieurs sous-échantillons.

La première étape est de prendre $\mathcal{D}_n$ et d'en soustraire une certaine fraction (disons $p = 0.2$), **aléatoirement** pour former un ensemble test $\mathcal{D}_{test}$, le reste, $\mathcal{D}_{ent}$ fut utiliser comme ensemble d'entrainement avant aujourd'hui,

$$\mathcal{D}_n = \underbrace{\mathcal{D}_{ent}}_{\mathcal{D}^\star} \cup \mathcal{D}_{test}.$$

Ensemble de test

1) on peut tirer au hasard les indices de la base d'apprentissage

```
> p_train <- 0.75
> n_train <- floor(  p_train * nrow(base))
> set.seed(123)
> train_ind <- sample( 1:nrow(base), size = n_train, replace = FALSE)
> train_base <- base[train_ind, ]
> test_base <-  base[-train_ind, ]
```

2) on peut brasser les données, et garder les premiers p% pour apprendre

```
> set.seed(123)
> base_shuffle = base[sample( 1:nrow(base), replace = FALSE), ]
> train_base <- base_shuffle[1:n_train, ]
> test_base <-  base_shuffle[(n_train+1):nrow(base), ]
```

Ensemble de validation

Pour éviter d'apprendre les hyperparamètres sur les données test et donc tricher. La méthode la plus simple est l'utilisation d'un ensemble de validation.

- On subdivise **aléatoirement** les données $\mathcal{D}^\star$ en deux, soit $\mathcal{D}_{ent}$ et $\mathcal{D}_{val}$.
- Alternativement on divise $\mathcal{D}_n$ en

$$\mathcal{D}_n = \mathcal{D}_{ent} \cup \mathcal{D}_{val} \cup \mathcal{D}_{test}$$

  avec $\mathcal{D}_{ent} \cap \mathcal{D}_{val} = \mathcal{D}_{ent} \cap \mathcal{D}_{test} = \mathcal{D}_{val} \cap \mathcal{D}_{test} = \emptyset$ (on parle de partition)
- On entraine les modèles (estimer les paramètres) sur $\mathcal{D}_{ent}$.
- On compare les modèles et choisi les hyperparamètres sur $\mathcal{D}_{val}$.
- Finalement on peut évaluer le modèle **final** sur $\mathcal{D}_{test}$.

Ensemble de validation

```
> p_train <- 0.5
> p_val   <- 0.2
> n_train <- floor(  p_train * nrow(base))
> n_val <- floor(  (p_train+p_val) * nrow(base))
> set.seed(123)
> base_shuffle = base[sample( 1:nrow(base), replace = FALSE), ]
> train_base <- base_shuffle[1:n_train, ]
> val_base  <-  base_shuffle[(n_train+1):n_val, ]
> test_base <-  base_shuffle[(n_val+1):nrow(base), ]
```

Ensemble de validation: régression polynomial

Rappel: On a introduit la régression polynomial au cours # 2.

Soit $\boldsymbol{X}$ notre prédicteur alors on veut apprendre le modèle suivant:

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + ... + \beta_d X^d$$

Ici, le degré du polynôme d est un **hyperparamètre** (doit être pré-déterminé avant d'utiliser une fonction "*fit*" sur R/Python), un paramètre qui affecte le forme du modèle.

Les coefficient β sont les **paramètres** du modèle.

Ensemble de validation: régression polynomial

Procédure:

- Trier **aléatoirement** les données $\mathcal{D}_n$
- Prendre toute nos données $\mathcal{D}_n$ et diviser celles-ci en trois sous-ensemble: $\mathcal{D}_{ent}$, $\mathcal{D}_{val}$ et $\mathcal{D}_{test}$. (Disons, 70%, 15%, 15% souvent un peu arbitrairement)
- Choisir la liste de degré considéré disons $d = (1, 2, \cdots, 10)$
- Entrainer (estimer $\widehat{\beta}$) les 10 modèles polynomiaux sur $\mathcal{D}_{ent}$.
- Comparer la performance (EQM/Précision) des 10 modèles sur $\mathcal{D}_{val}$ et choisir d qui minimise la mesure sur $\mathcal{D}_{val}$.
- Raporter finalement la performance du modèle choisi sur $\mathcal{D}_{test}$.

On utilise donc $\mathcal{D}_{val}$ pour apprendre la valeur de l'hyperparamètre. On utilise chacun des trois ensembles pour une seule tâche précise.

Ensemble de validation: régression Ridge ou Lasso

Procédure:

- Trier **aléatoirement** les données $\mathcal{D}_n$
- Prendre toute nos données $\mathcal{D}_n$ et diviser celles-ci en trois sous-ensemble: $\mathcal{D}_{ent}$, $\mathcal{D}_{val}$ et $\mathcal{D}_{test}$.
- Choisir la liste des valeurs λ à considérer.
- Entrainer (estimer $\widehat{\beta}$) les modèles ridge sur $\mathcal{D}_{ent}$.
- Comparer la performance (EQM/Précision) des modèles sur $\mathcal{D}_{val}$ pour choisir λ.
- Raporter finalement la performance sur $\mathcal{D}_{test}$.

On utilise donc $\mathcal{D}_{val}$ pour apprendre la valeur de l'hyperparamètre λ. On utilise chaque ensemble une seule fois.

Ensemble de validation

En d'autres mots:

- On apprend les paramètres sur $\mathcal{D}_{ent}$.
- On apprend les hyperparamètres ensuite sur $\mathcal{D}_{val}$
- On test la performance combiné paramètres et hyperparamètres (le modèle final) sur $\mathcal{D}_{test}$.

Cette simple approche fonctionne plutôt bien; en application ça vous protège des critiques et vous donne un bon cadre de travail.

Ensemble de validation et la variabilité

Peut-on faire mieux ?

- Supposons que les modèles sont variables (grandes variances/flexibilité)
- Peut-être qu'une partition de $\mathcal{D}^*$ en $\mathcal{D}_{ent}$ et $\mathcal{D}_{val}$ différente peut mener à différents modèles.
- On veut donc *stabiliser* la procédure de sélection d'hyperparamètres proposé. On veut être sûre de notre choix de modèle.
- La validation croisée permet de répéter ce processus et donc de faire une moyenne de la performance des modèles sur plusieurs $\mathcal{D}_{val}$ avant d'en faire le choix.

Ensemble de validation et la variabilité

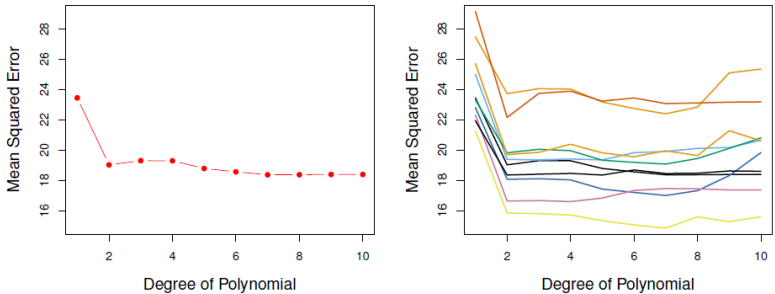


FIGURE 5.2. *The validation set approach was used on the* `Auto` *data set in order to estimate the test error that results from predicting* `mpg` *using polynomial functions of* `horsepower`. Left: *Validation error estimates for a single split into training and validation data sets.* Right: *The validation method was repeated ten times, each time using a different random split of the observations into a training set and a validation set. This illustrates the variability in the estimated test MSE that results from this approach.*

Figure 3: Extrait de la figure 5.2 du livre James et al. (2013)

Validation croisée

- Quand on reçoit $\mathcal{D}_n$, notre échantillon, on détermine toute de suite des données test $\mathcal{D}_{test}$ puis on n'y touche plus (on ne veut pas tricher).
- Ensuite appelons le reste $\mathcal{D}^* = \mathcal{D}_n \backslash \mathcal{D}_{test}$
- Puis on va diviser $\mathcal{D}^*$ en k "plies" (k-**folds**), k sous-ensembles aléatoires de même taille.
- On entraine le modèle sur $k-1$ plies et on valide sur l'autre plie.
- En répétant cette produre sur chacunes des combinaisons possibles de plies, on obtient k estimateurs de l'EQM sur des données validés et on peut en prendre la moyenne!

Ensemble de validation

FIGURE 5.1. *A schematic display of the validation set approach. A set of n observations are randomly split into a training set (shown in blue, containing observations 7, 22, and 13, among others) and a validation set (shown in beige, and containing observation 91, among others). The statistical learning method is fit on the training set, and its performance is evaluated on the validation set.*

Figure 4: Extrait de la figure 5.1 du livre James et al. (2013)

Validation croisée

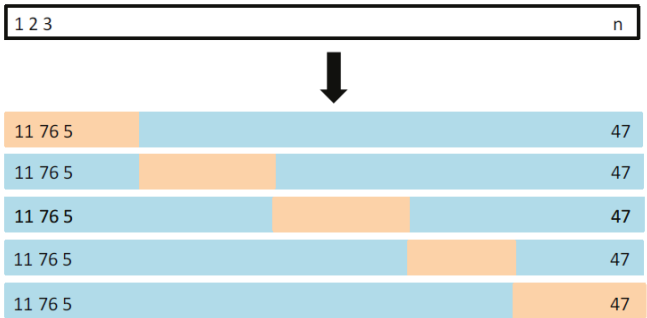


FIGURE 5.5. *A schematic display of 5-fold CV. A set of n observations is randomly split into five non-overlapping groups. Each of these fifths acts as a validation set (shown in beige), and the remainder as a training set (shown in blue). The test error is estimated by averaging the five resulting MSE estimates.*

Figure 5: Extrait de la figure 5.5 du livre James et al. (2013)

Ensemble de validation

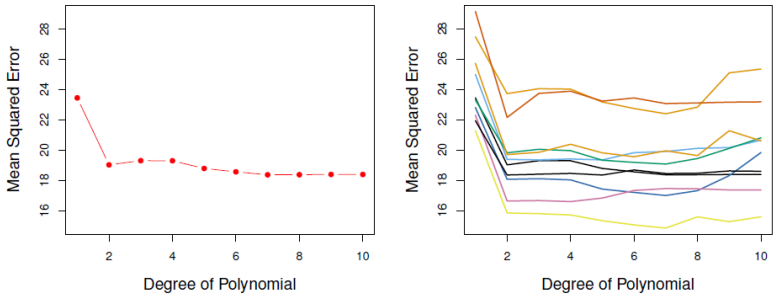


FIGURE 5.2. *The validation set approach was used on the* `Auto` *data set in order to estimate the test error that results from predicting* `mpg` *using polynomial functions of* `horsepower`*. Left: Validation error estimates for a single split into training and validation data sets. Right: The validation method was repeated ten times, each time using a different random split of the observations into a training set and a validation set. This illustrates the variability in the estimated test MSE that results from this approach.*

Figure 6: Extrait de la figure 5.2 du livre James et al. (2013)

Validation croisée

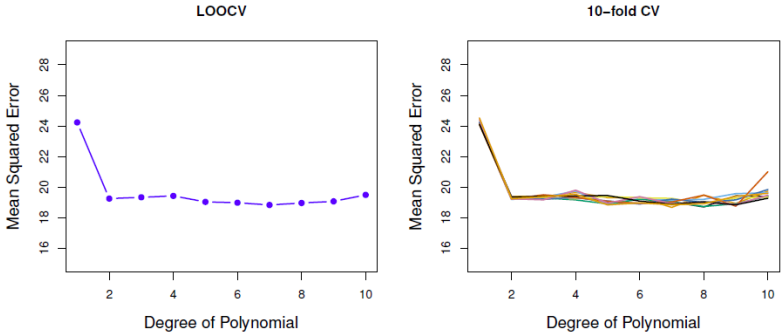


FIGURE 5.4. *Cross-validation was used on the* `Auto` *data set in order to estimate the test error that results from predicting* `mpg` *using polynomial functions of* `horsepower`. Left: *The LOOCV error curve.* Right: *10-fold CV was run nine separate times, each with a different random split of the data into ten parts. The figure shows the nine slightly different CV error curves.*

Figure 7: Extrait de la figure 5.4 du livre James et al. (2013)

On en laisse juste un de côté

Une configuration populaire de la validation croisée est celle ou $k = n$; on entraine sur $n - 1$ observations et compare les modèles sur l'observation restante.

Ça nous donne une estimation vraiment stable (on prend la moyenne sur n EQM pour chaque hyperparamètre), mais ça prend beaucoup de temps puisqu'on entraine le modèle n fois pour chaque valeur des hyperparamètres.

C'est un peu un idéal à atteindre; on espère que la k-fold validation croiséee fait un travail aussi bon.

Validation croisée

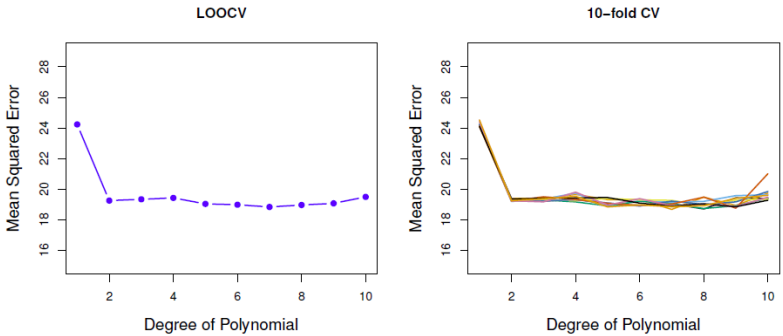


FIGURE 5.4. *Cross-validation was used on the* `Auto` *data set in order to estimate the test error that results from predicting* `mpg` *using polynomial functions of* `horsepower`. Left: *The LOOCV error curve.* Right: *10-fold CV was run nine separate times, each with a different random split of the data into ten parts. The figure shows the nine slightly different CV error curves.*

Figure 8: Extrait de la figure 5.4 du livre James et al. (2013)

Ensemble de validation

FIGURE 5.1. *A schematic display of the validation set approach. A set of n observations are randomly split into a training set (shown in blue, containing observations 7, 22, and 13, among others) and a validation set (shown in beige, and containing observation 91, among others). The statistical learning method is fit on the training set, and its performance is evaluated on the validation set.*

Figure 9: Extrait de la figure 5.1 du livre James et al. (2013)

k-fold Validation croisée

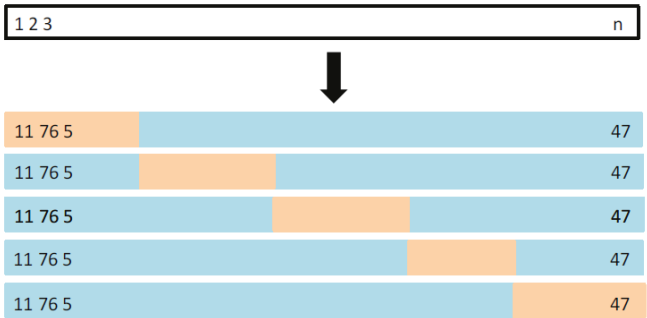


FIGURE 5.5. *A schematic display of 5-fold CV. A set of n observations is randomly split into five non-overlapping groups. Each of these fifths acts as a validation set (shown in beige), and the remainder as a training set (shown in blue). The test error is estimated by averaging the five resulting MSE estimates.*

Figure 10: Extrait de la figure 5.5 du livre James et al. (2013)

Validation croisée Leave-one-out LOOCV

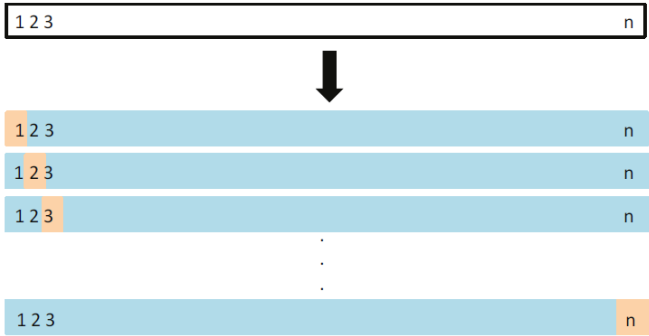


FIGURE 5.3. *A schematic display of LOOCV. A set of n data points is repeatedly split into a training set (shown in blue) containing all but one observation, and a validation set that contains only that observation (shown in beige). The test error is then estimated by averaging the n resulting MSE's. The first training set contains all but observation 1, the second training set contains all but observation 2, and so forth.*

Validation croisée

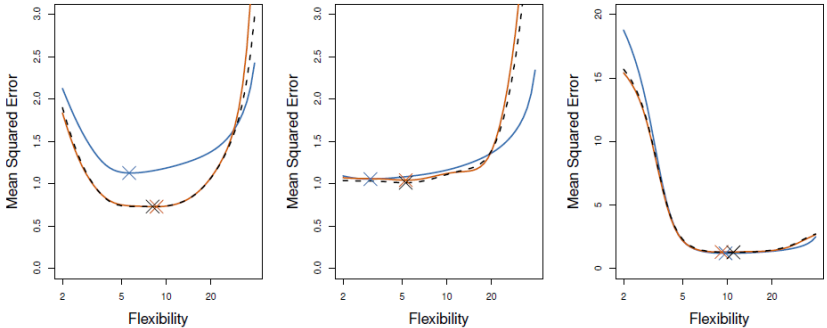


FIGURE 5.6. *True and estimated test MSE for the simulated data sets in Figures 2.9 (*left*), 2.10 (*center*), and 2.11 (*right*). The true test MSE is shown in blue, the LOOCV estimate is shown as a black dashed line, and the 10-fold CV estimate is shown in orange. The crosses indicate the minimum of each of the MSE curves.*

Figure 12: Extrait de la figure 5.6 du livre James et al. (2013)

Validation croisée

- La validation est encore l'approche numéro #1 pour la sélection d'hyperparamètre
- C'est simple mais c'est important d'être à l'aise avec ce concept (ça revient toujours)
- Les modèles modernes d'AI (Réseau de neuronne etc) ont beaucoup d'hyperparamètres, il faut donc penser concrètement à comment faire la validation.

```
> set.seed(123)
> base_shuffle <- base[sample( 1:nrow(base), replace = FALSE), ]
> k <- 5
> k_fold <- rep(1:k, ceiling(nrow(base)/k)[1:nrow(base)]
  # step j
> train_base <- base_shuffle[k_fold != j, ]
> test_base <-  base_shuffle[k_fold == j, ]
```

Validation croisée

En régression, on avait déjà l'habitude de faire du "leave one-out".
Pour $i = 1, \cdots, n$, on définit:

- $\boldsymbol{X}(i)$: la matrice de design $\boldsymbol{X}$ sans la i-ème ligne.
- $\widehat{\beta}_{(i)}$ et $\widehat{\sigma}^2_{(i)}$: les estimateurs de β et σ^2 basés sur toutes les observations hormis celle du i-ème individu.
- on note $\widehat{\varepsilon}_i^{(i)} = Y_i - \widehat{Y}_{(i)} = Y_i - \boldsymbol{X}_{(i)}\beta_{(i)}$ = erreur de prédiction de la i-ème variable Y_i sans utiliser l'information du i-ème individu.
- On peut démontrer que si la suppression de la i-ème ligne ne modifie pas le rang de $\boldsymbol{X}$, alors

$$\widehat{\varepsilon}_i^{(i)} = \frac{\widehat{\varepsilon}_i}{1 - \boldsymbol{H}_{ii}}, \text{ où } \boldsymbol{H} = \boldsymbol{X}\left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top.$$

Rééchantillonage

Dans ce cours, on a vu qu'on pouvait utiliser l'ensemble de validation et l'ensemble test pour **simuler** de nouveaux échantillons.

Jusqu'à présent, on emploie **ces nouveaux échantillons** pour valider ou bien tester nos modèles.

Par contre, trouver des manière de simuler avoir **de nouveaux échantillons** peut s'appliquer à de nombreux problèmes.

Par exemple un interval de confiance non paramétrique pour $\widehat{\beta}$.

Avant d'aller au laboratoire, on va parler de rééchantillonage et on va surtout l'appliquer à la sélection de modèle.

Bootstrap

Débutons par le rééchantillonage bootstrap.

Supposons que nos estimés d'EQM sont instable malgré la validation croisée.

On pourrait vouloir un estimateur Monte Carlo (moyenne empirique) avec 100 observations et ce n'est pas possible parce que $n^* = 50$ et donc même le leave-one-out ne produit que 50 estimés.

On pourrait **tirer avec remise** des échantillon des tailles n^*, on peut en tirer 1000.

On appelle ces échantillons, des échantillons bootstrap, disons $\mathcal{D}^b$.

```
> set.seed(123)
> base_boot <- base[sample( 1:nrow(base), replace = TRUE), ]
```

À chaque fois il restera quelques observations pas dans $\mathcal{D}^b$ (à cause du avec remise, la probabilité que les n observations y soit est presque nul.)

Bootstrap

On peut ensuite entrainer des modèles sur ces échantillons et estimer l'EQM sur les autres observations.

On peut aussi *construire* des intervales des confiance sur nos paramètres en quelque sorte: regarder les 1000 valeurs de $\widehat{\beta}_1$ sur nos 1000 échantillons bootstrap et faire des intervales empiriques.

Bref, on peut utiliser la ré-échantillonage pour générer plusieurs échantillons et évaluer la variabilité des modèles ou pour stabiliser notre choix d'hyperparamètre.

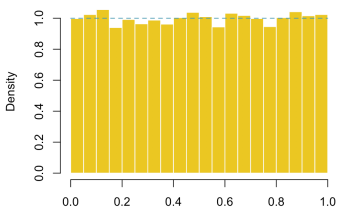
Tirage au hasard et bootstrap

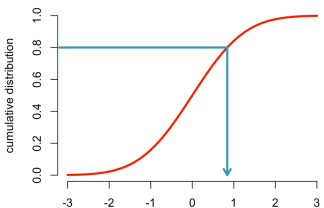
On sait que si F est une fonction de répartition $\mathbb{R} \to [0,1]$, et si $U \sim \mathcal{U}([0,1])$, alors $X = F^{-1}(U)$ a pour fonction de répartition F.

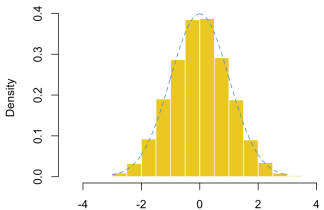
En effet, si $x \in \mathbb{R}$, $\mathbb{P}[X \leq x]$ vérifie

$$\mathbb{P}[X \leq x] = \mathbb{P}[F^{-1}(U) \leq x] = \mathbb{P}[F(F^{-1}(U)) \leq F(x)] = \mathbb{P}[U \leq F(x)] = F(x),$$

où $F^{-1}(u) = \inf\{x \mid F(x) \geq u\}$ pour tout $u \in (0,1)$.







Tirage au hasard et bootstrap

```
> U <- runif(100)
  [1] 0.29 0.79 0.41 0.88 0.94 0.05 0.53 0.89 0.55 0.46 0.96 0.45 0.68
 [14] 0.57 0.10 0.90 0.25 0.04 0.33 0.95 0.89 0.69 0.64 0.99 0.66 0.71
 [27] 0.54 0.59 0.29 0.15 0.96 0.90 0.69 0.80 0.02 0.48 0.76 0.22 0.32
 [40] 0.23 0.14 0.41 0.41 0.37 0.15 0.14 0.23 0.47 0.27 0.86 0.05 0.44
 [53] 0.80 0.12 0.56 0.21 0.13 0.75 0.90 0.37 0.67 0.09 0.38 0.27 0.81
 [66] 0.45 0.81 0.81 0.79 0.44 0.75 0.63 0.71 0.00 0.48 0.22 0.38 0.61
 [79] 0.35 0.11 0.24 0.67 0.42 0.79 0.10 0.43 0.98 0.89 0.89 0.18 0.13
 [92] 0.65 0.34 0.66 0.32 0.19 0.78 0.09 0.47 0.51
```

```
> qnorm(U)
  [1] -0.56  0.80 -0.23  1.19  1.56 -1.69  0.07  1.24  0.13 -0.11  1.72
 [12] -0.12  0.46  0.18 -1.27  1.28 -0.69 -1.73 -0.45  1.69  1.22  0.50
 [23]  0.36  2.53  0.40  0.55  0.11  0.24 -0.56 -1.05  1.79  1.29  0.50
 [34]  0.83 -1.97 -0.06  0.70 -0.78 -0.47 -0.73 -1.07 -0.22 -0.22 -0.33
 [45] -1.03 -1.09 -0.73 -0.09 -0.63  1.07 -1.69 -0.15  0.84 -1.17  0.15
 [56] -0.82 -1.14  0.68  1.25 -0.32  0.43 -1.31 -0.30 -0.60  0.90 -0.13
 [67]  0.88  0.89  0.82 -0.15  0.69  0.33  0.55 -3.23 -0.06 -0.77 -0.31
```

Tirage au hasard et bootstrap

Considérons un échantillon $\{x_1, \cdots, x_n\}$ i.i.d.

$$\text{de loi (théorique) } F(x) = \mathbb{P}[X_i \leq x],$$

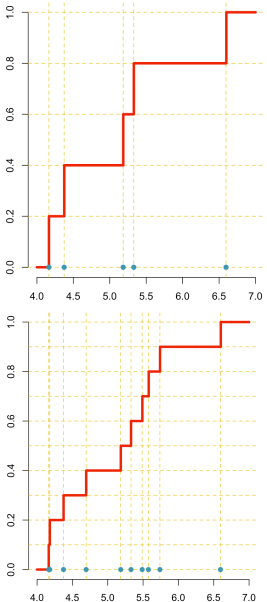
La **fonction de répartition empirique** est

$$\widehat{F}_n(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(x_i \leq x), \ x \in \mathbb{R}$$

Théorème de **Glivenko-Cantelli**: $\widehat{F}_n \to F$ lorsque $n \to \infty$, ou plus précisément

$$\|\widehat{F}_n - F\|_\infty = \sup_{x \in \mathbb{R}} |\widehat{F}_n(x) - F(x)| \longrightarrow 0$$

lorsque $n \to \infty$.



Tirage au hasard et bootstrap

Tirer suivant $\widehat{F}_n^{-1}$ revient à rééchantillonner dans $\{x_1, \cdots, x_n\}$ avec probabilité $1/n$ (ou avec remplacement)

```
> x
[1] 4.164 4.374 5.184 5.330 6.595
> Qemp = Vectorize(function(u) sort(U)[ceiling(length(x)*u)])
> Qemp(U)
  [1] 4.37 5.33 5.18 6.60 6.60 4.16 5.18 6.60 5.18 5.18 6.60 5.18 5.33
 [14] 5.18 4.16 6.60 4.37 4.16 4.37 6.60 6.60 5.33 5.33 6.60 5.33 5.33
 [27] 5.18 5.18 4.37 4.16 6.60 6.60 5.33 5.33 4.16 5.18 5.33 4.37 4.37
 [40] 4.37 4.16 5.18 5.18 4.37 4.16 4.16 4.37 5.18 4.37 6.60 4.16 5.18
 [53] 5.33 4.16 5.18 4.37 4.16 5.33 6.60 4.37 5.33 4.16 4.37 4.37 6.60
 [66] 5.18 6.60 6.60 5.33 5.18 5.33 5.33 5.33 4.16 5.18 4.37 4.37 5.33
 [79] 4.37 4.16 4.37 5.33 5.18 5.33 4.16 5.18 6.60 6.60 6.60 4.16 4.16
 [92] 5.33 4.37 5.33 4.37 4.16 5.33 4.16 5.18 5.18
```

voir aussi Davison and Hinkley (1997)

Régression Linéaire & Bootstrap (1)

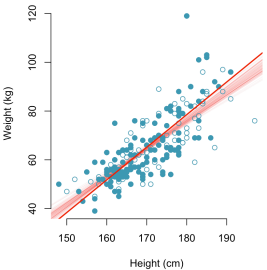
Données $\mathcal{D}_n = \{\boldsymbol{z}_i = (y_i, \boldsymbol{x}_i)\}$, $i = 1, \cdots, n$.

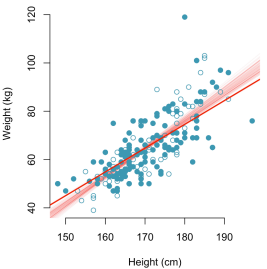
1) approche par **paired sampling** ou échantillonnage des observations, i.e. $\{\boldsymbol{z}_1^\star, \cdots, \boldsymbol{z}_n^\star\}$.

1. tirer $\{i_1^{(b)}, \cdots, i_n^{(b)}\}$ aléatoirement (avec remise) dans $\{1, 2, \cdots, n\}$
2. définir la base $(\boldsymbol{x}_i^{(b)}, y_i^{(b)}) = (\boldsymbol{x}_{i^{(b)}}, y_{i^{(b)}})$'s, et estimer un modèle (linéaire)
3. soit $\widehat{\boldsymbol{\beta}}^{(b)}$ l'estimation du paramètre, ou $\widehat{y}_{n+1}^{(b)}$ une prévision (associée à $\boldsymbol{x}_{n+1}$) (itérer pour $b = 1, \cdots, B$)

Régression Linéaire & Bootstrap (1)

```
> Davis = read.table("http://freakonometrics.free.fr
    /Davis.txt")
> BETA = matrix(NA,1000,2)
> for(s in 1:1000){
+  idx = sample(1:nrow(Davis),nrow(Davis),
+        replace=TRUE)
+  reg_sim = lm(weight~height, data=Davis[idx,])
+  BETA[s,] = reg_sim$coefficients
+ }
> hist(BETA[,1])
> hist(BETA[,2])
```

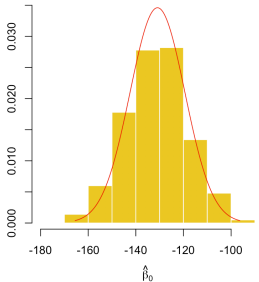


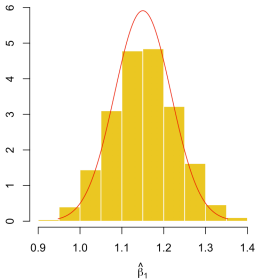


Régression Linéaire & Bootstrap (1)

On peut aussi utiliser le code suivant

```
> library(boot)
> coef = function(formula, data, indices) {
+  d = data[indices,]
+  fit = lm(formula, data=d)
+  return(coef(fit))
+ }
> results = boot(data=Davis, statistic=coef, R=1000,
+           formula=weight~height)
> plot(results, index=1)
> plot(results, index=2)
```





Régression Linéaire & Bootstrap (2)

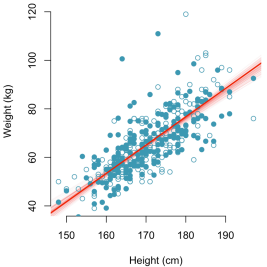
2) on peut aussi faire du **model-based resampling**

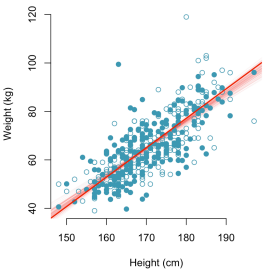
1. estimer un modèle linéaire, et obtenir les prédictions $\widehat{y}_i$ et les résidus $\widehat{\varepsilon}_i$
2. échantillonner $\widehat{\varepsilon}_1^{(b)}, \cdots, \widehat{\varepsilon}_n^{(b)}$ à partir de $\{\widehat{\varepsilon}_1, \widehat{\varepsilon}_2, \cdots, \widehat{\varepsilon}_n\}$
3. poser $y_i^{(b)} = \widehat{y}_i + \widehat{\varepsilon}_i^{(b)}$ (à partir du modèle)
4. à partir de la base $(\boldsymbol{x}, y^{(b)}) = (\boldsymbol{x}_i, y_i^{(b)})$, estimer un modèle linéaire
5. noter $\widehat{\boldsymbol{\beta}}^{(b)}$ l'estimateur de $\boldsymbol{\beta}$, ou $\widehat{y}_{n+1}^{(b)}$ une prévision (associée à $\boldsymbol{x}_{n+1}$)

Note pour une régression simple, $\widehat{\beta}_1^{(b)} = \dfrac{\sum[x_i - \overline{x}] \cdot y_i^{(b)}}{\sum[x_i - \overline{x}]^2} = \widehat{\beta}_1 + \dfrac{\sum[x_i - \overline{x}] \cdot \widehat{\varepsilon}_i^{(b)}}{\sum[x_i - \overline{x}]^2}$ et donc $\mathbb{E}[\widehat{\beta}_1^{(b)}] = \widehat{\beta}_1$, alors que $\text{Var}[\widehat{\beta}_1^{(b)}] = \dfrac{\sum[x_i - \overline{x}]^2 \cdot \text{Var}[\widehat{\varepsilon}_i^{(b)}]}{\left(\sum[x_i - \overline{x}]^2\right)^2} \sim \dfrac{\sigma^2}{\sum[x_i - \overline{x}]^2}$.

Régression Linéaire & Bootstrap (2)

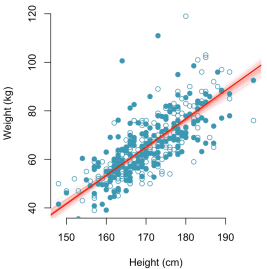
```
> BETA = matrix(NA,1000,2)
> reg = lm(weight~height, data=Davis)
> epsilon = residuals(reg)
> for(s in 1:1000){
+   eps = sample(epsilon,nrow(Davis),replace=TRUE)
+   Davis_s = data.frame(height = Davis$height,
+                        weight = predict(reg)+eps)
+   reg_sim = lm(weight~height, data=Davis_s)
+   BETA[s,] = reg_sim$coefficients
+ }
> hist(BETA[,1])
> hist(BETA[,2])
```

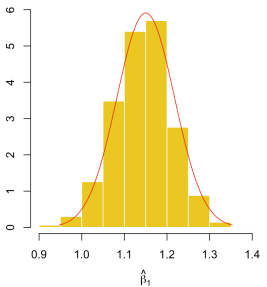




Régression Linéaire & Bootstrap (2)

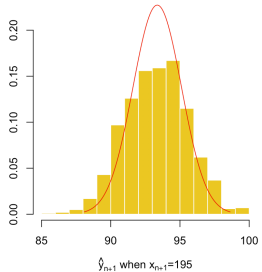
```
> BETA = matrix(NA,1000,2)
> reg = lm(weight~height, data=Davis)
> epsilon = residuals(reg)
> for(s in 1:1000){
+   eps = sample(epsilon,nrow(Davis),replace=TRUE)
+   Davis_s = data.frame(height = Davis$height,
+                        weight = predict(reg)+eps)
+   reg_sim = lm(weight~height, data=Davis_s)
+   BETA[s,] = reg_sim$coefficients
+ }
> hist(BETA[,1])
> hist(BETA[,2])
```

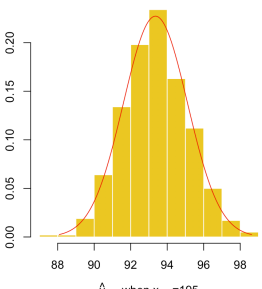




Régression Linéaire & Bootstrap (1/2)

```
> PRED = matrix(NA,1000,2)
> nwDavis = data.frame(height = 195)
> for(s in 1:1000){
+   idx = sample(1:n,n,replace=TRUE)
+   reg_sim = lm(weight~height, data=Davis[idx,])
+   PRED[s,1] = predict(reg_sim, newdata=nwDavis)
+   eps = sample(epsilon,nrow(Davis),replace=TRUE)
+   Davis_s = data.frame(height = Davis$height,
+                        weight = predict(reg)+eps)
+   reg_sim = lm(weight~height, data=Davis_s)
+   PRED[s,2] = predict(reg_sim, newdata=nwDavis)
+ }
```

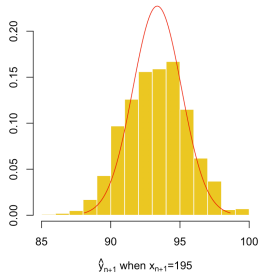


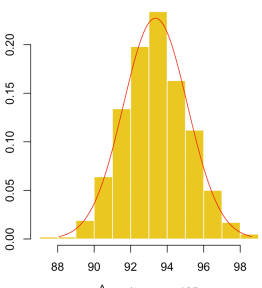


Régression Linéaire & Bootstrap (2/2)

```
> apply(PRED,2,function(x) quantile(x,.025))
[1] 89.04203 89.63030
> apply(PRED,2,function(x) quantile(x,.975))
[1] 97.60345 97.02423
> predict(lm(weight~height, data=Davis),
+         newdata=nwDavis,interval="confidence",se.
    fit = TRUE)
$fit
       fit      lwr      upr
1 93.35749 89.89571 96.81927

$se.fit
[1] 1.755451
```





Sous-échantillonnage

- D'habitude le bootstrap est un échantillon avec remises.
- La duplicité de certaines observations dans un bootstrap peut causer des problèmes.
- On peut aussi faire du sans-remise en prenant un sous-ensemble ($n^* = 50$ et on rééchantillonne des échantillons de taille 35, on peut valider sur les 15 restantes!).
- On apelle ça du **sous-échantillonnage**
- Dans le cours je vais utiliser sous-échantillonnage et bootstrap de manière interchangeable, ce n'est pas toujours le cas dans la littérature. Ce sont deux approches différentes de rééchantillonage.

Rééchantillonage: choisir un hyperparamètre

Procédure:

- On divise $\mathcal{D}_n$ en $\mathcal{D}_{test}$ et $\mathcal{D}^*$ deux ensembles disjoints.
- On échantillonne parmi $\mathcal{D}^*$ (disons 80% des points sans remises) et on entraine tous nos modèle sur ces données ($\mathcal{D}^b_{ent}$).
- On calcule l'EQM sur les 20% de points restants ($\mathcal{D}^b_{val}$).
- On repète le processus pour B échantillons.
- On calcule l'EQM moyen (moyenne sur les B échantillons) pour chaque modèle.

Rééchantillonage

- On génère autant d'échantillons qu'on a besoin; c'est très utile de pouvoir faire ça.
- Plusieurs applications: intervalles de confiance, modèles ensemblistes et sélection d'hyperparamètres.
- Plusieurs méthodes d'échantillonage.
- Laisse place à la créativité.
- On va en reparler constamment.

Validation et rééchantillonage: en pratique

- Extrêmement populaire en pratique, les statisticiens aiment bien la validation croisée, mais il est plus facile (et presque équivalent) de programmer le sous-échantillonage décrit.
- On s'en sert tout le temps à tout moment pour toutes sortes de raisons.
- Quand vous voulez savoir ce qui se passe si on entraine un modèle sur des données *différentes* on peut utiliser ces techniques pour imiter ce processus. C'est moins cher et plus rapide que de collecter de nouvelle données.

Exercises

- Lecture: Chapitre 5 du manuel de référence (James et al. (2013))
- Lab: 5.3 au complet
- Exercises 5.4.3, **5.4.5**

Récapitulons...

Hold-Out Cross Validation

1. Split $\{1, 2, \cdots, n\}$ in $\mathcal{D}_{ent}$ (training) and $\mathcal{D}_{test}$ (test)
2. Estimate $\widehat{m}$ on sample $(y_i, \boldsymbol{x}_i)$, $i \in \mathcal{D}_{ent}$: $\widehat{m}_{ent}$
3. Compute $\dfrac{1}{|\mathcal{D}_{test}|} \displaystyle\sum_{i \in V} \ell(y_i, \widehat{m}_{ent}(\boldsymbol{x}_i))$

Leave-one-Out Cross Validation (LOOCV)

1. Estimate n models : $\widehat{m}_{-j}$ on sample $(y_i, \boldsymbol{x}_i)$, $i \in \{1, \cdots, n\} \backslash \{j\}$
2. Compute $\dfrac{1}{n} \displaystyle\sum_{i=1}^{n} \ell(y_i, \widehat{m}_{-i}(\boldsymbol{x}_i))$

Récapitulons...

K-Fold Cross Validation

1. Split $\{1, 2, \cdots, n\}$ in K groups $V_1, \cdots, V_K$
2. Estimate K models : $\widehat{m}_k$ on sample $(y_i, \boldsymbol{x}_i)$, $i \in \{1, \cdots, n\} \backslash V_k$
3. Compute $\frac{1}{K} \sum_{k=1}^{K} \frac{1}{|V_k|} \sum_{i \in V_k} \ell(y_i, \widehat{m}_k(\boldsymbol{x}_i))$

Bootstrap Cross Validation

1. Generate B boostrap samples from $\{1, 2, \cdots, n\}$, $I_1, \cdots, I_B$
2. Estimate B models : $\widehat{m}_b$ on sample $(y_i, \boldsymbol{x}_i)$, $i \in I_b$
3. Compute $\frac{1}{B} \sum_{b=1}^{B} \frac{1}{n - |I_b|} \sum_{i \notin I_b} \ell(y_i, \widehat{m}_b(\boldsymbol{x}_i))$

Références

Davison, A. C. and Hinkley, D. V. (1997). *Bootstrap methods and their application*. Number 1. Cambridge university press.

James, G., Witten, D., Hastie, T., Tibshirani, R., et al. (2013). *An introduction to statistical learning*, volume 112. Springer.