Apprentissage supervisé - Régression

Mohamed Essaied Hamrita

IHEC - Sousse

Introduction

Introduction

L'apprentissage supervisé est classée dans deux catégories d'algorithmes:

Introduction

L'apprentissage supervisé est classée dans deux catégories d'algorithmes:

- Régression : Un problème de régression se pose lorsque la variable étiquetée (labelled) est une valeur réelle.

Introduction

L'apprentissage supervisé est classée dans deux catégories d'algorithmes:

- Régression : Un problème de régression se pose lorsque la variable étiquetée (labelled) est une valeur réelle.

Dans la littérature, il existe plusieurs types de modèles de régression tels que la régression linéaire, la régression logistique, le SVM (Support Vector Machine), NN (neural network) et KNN (K nearest neighbour), ...

Introduction

L'apprentissage supervisé est classée dans deux catégories d'algorithmes:

- Régression : Un problème de régression se pose lorsque la variable étiquetée (labelled) est une valeur réelle.

Dans la littérature, il existe plusieurs types de modèles de régression tels que la régression linéaire, la régression logistique, le SVM (Support Vector Machine), NN (neural network) et KNN (K nearest neighbour), ...

- Classification : On parle d'un problème de classification lorsque la variable étiquetée est catégorielle (binaire ou multi-classes).

Introduction

L'apprentissage supervisé est classée dans deux catégories d'algorithmes:

- Régression : Un problème de régression se pose lorsque la variable étiquetée (labelled) est une valeur réelle.

Dans la littérature, il existe plusieurs types de modèles de régression tels que la régression linéaire, la régression logistique, le SVM (Support Vector Machine), NN (neural network) et KNN (K nearest neighbour), ...

- Classification : On parle d'un problème de classification lorsque la variable étiquetée est catégorielle (binaire ou multi-classes).

Les modèles de classification comprennent la régression logistique, l'arbre de décision (tree decision), la forêt aléatoire (random forest), KNN et SVM.

La fonction coût (Cost (loss) function)

La fonction coût (Cost (loss) function)

L'apprentissage statistique supervisé repose sur une hypothèse qui minimise le coût (la perte) moyen(ne) dans l'échantillon. Ce qu'on appelle le problème minimisation du coût.

La fonction coût (Cost (loss) function)

L'apprentissage statistique supervisé repose sur une hypothèse qui minimise le coût (la perte) moyen(ne) dans l'échantillon. Ce qu'on appelle le problème minimisation du coût.

La fonction d'un prédicteur f est donné par: $\mathcal{R}(f) = \mathbb{E}\left(\ell\left(f(x)\right), y\right)$.

Le meilleur prédicteur est celui qui minimise cette fonction: $f^* = \min\left\{\mathcal{R}(f)\right\}$.

La fonction coût (Cost (loss) function)

L'apprentissage statistique supervisé repose sur une hypothèse qui minimise le coût (la perte) moyen(ne) dans l'échantillon. Ce qu'on appelle le problème minimisation du coût.

La fonction d'un prédicteur f est donné par: $\mathcal{R}(f) = \mathbb{E}\left(\ell\left(f(x)\right), y\right)$.

Le meilleur prédicteur est celui qui minimise cette fonction: $f^* = \min\left\{\mathcal{R}(f)\right\}$.

L'estimateur de la fonction coût est:

$$\mathcal{R}_n(f) = \frac{1}{n}\sum_i \ell\left(f(x_i), y_i\right)$$

La régression linéaire

La régression linéaire

La régression linéaire consiste à estimer $(k+1)$ paramètres dans une relation **linéaire** entre une variable dépendante (target) et k variables indépendantes (features):

$$y_i = \beta_0 + \beta_1 x_{1i} + \ldots + \beta_k x_{ki} + \varepsilon_i \qquad (Y = X\beta + \varepsilon)$$

- f est la relation linéaire entre y_i et x_{ki};
- ℓ peut être le coût de l'erreur quadratique:

$$\ell(f(x), y) = (f(x) - y)^2 = \frac{1}{n} \sum_i \left(x_i' \beta - y_i\right)^2$$

- Le meilleur prédicteur s'obtient en minimisant la fonction coût

$$\widehat{f^*} = \min_\beta (f(x) - y)^2 = \frac{1}{n} \sum_i \left(x_i' \beta - y_i\right)^2$$

Implémentation sous R:

- Dataset and exploration: La bibliothèque `MASS` contient l'ensemble de données de `Boston`, qui enregistre la `medv` (valeur mediane des maisons) pour 506 quartiers de Boston. Nous chercherons à prédire `medv` à l'aide de 13 prédicteurs tels que `rm` (nombre moyen de pièces par maison), age (âge moyen des maisons) et `lstat` (pourcentage de ménages à faible revenu).

```
library(MASS)
names(Boston)      # Les noms des variables
```

```
 [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"
 [8] "dis"     "rad"     "tax"     "ptratio" "black"   "lstat"   "medv"
```

```
any(is.na(Boston)) # y a t-il des valeurs manquantes?
```

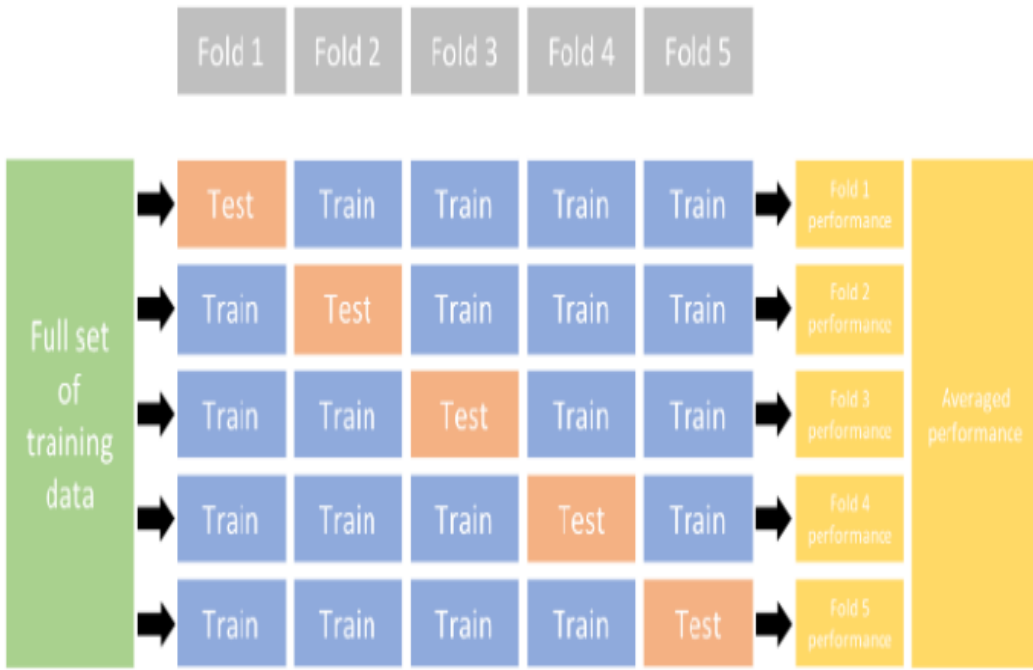
```
[1] FALSE
```

- Model training and evaluation

Le problème fondamental de sur-estimation (overfitting) est que le risque empirique, est biaisé à la baisse par rapport au risque de la population. Nous pouvons éliminer ce biais de deux manières : (a) des approches de ré-échantillonnage purement algorithmiques (train/test (échantillonnage aléatoire simple (EAS)) et k-fold cross validation (échantillonnage stratifié)) et (b) des estimateurs fondés sur la théorie.

- Model training and evaluation

Le problème fondamental de sur-estimation (overfitting) est que le risque empirique, est biaisé à la baisse par rapport au risque de la population. Nous pouvons éliminer ce biais de deux manières : (a) des approches de ré-échantillonnage purement algorithmiques (train/test (échantillonnage aléatoire simple (EAS)) et k-fold cross validation (échantillonnage stratifié)) et (b) des estimateurs fondés sur la théorie.



- Spit data

```
set.seed(0123); n=nrow(Boston)
i_train=sample(1:n, round(0.7*n))
train=Boston[i_train,] ; test=Boston[-i_train,]
```

où encore, avec la bibliothèque caret

```
library(caret); set.seed(0123)
i_tr=createDataPartition(Boston[,1], p=0.7, list=F)
train2=Boston[i_tr,] ; test2=Boston[-i_tr,]
```

- Estimation

```
mod1=lm(medv ~ rm , data=train); summary(mod1)
```

```
Call:
lm(formula = medv ~ rm, data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-19.325  -2.586   0.028   2.829  39.073

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -32.6774     3.3137  -9.861   <2e-16 ***
rm            8.7735     0.5256  16.691   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.803 on 352 degrees of freedom
Multiple R-squared:  0.4418,    Adjusted R-squared:  0.4402
F-statistic: 278.6 on 1 and 352 DF,  p-value: < 2.2e-16
```

```r
mod2=update(mod1, .~. + lstat , data=train)
summary(mod2)
```

```
Call:
lm(formula = medv ~ rm + lstat, data = train)

Residuals:
    Min      1Q  Median      3Q     Max 
-12.468  -3.470  -1.154   1.753  27.410 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1.98609    3.82193    0.52    0.604    
rm           4.58297    0.54042    8.48 6.33e-16 ***
lstat       -0.66666    0.05145  -12.96  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.603 on 351 degrees of freedom
Multiple R-squared:  0.6224,    Adjusted R-squared:  0.6202 
F-statistic: 289.3 on 2 and 351 DF,  p-value: < 2.2e-16
```

```r
mod3=update(mod2, .~. + age, data=train)
summary(mod3)
```

```
Call:
lm(formula = medv ~ rm + lstat + age, data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-12.928  -3.442  -1.175   1.915  26.430

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.96544    3.81823   0.515    0.607
rm           4.49245    0.54437   8.253  3.2e-15 ***
lstat       -0.71310    0.06261 -11.389  < 2e-16 ***
age          0.01723    0.01327   1.299    0.195
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.598 on 350 degrees of freedom
Multiple R-squared:  0.6242,    Adjusted R-squared:  0.621
F-statistic: 193.8 on 3 and 350 DF,  p-value: < 2.2e-16
```

On a estimé 3 modèles. La question qui se pose: quel est le meilleur modèle ?

On a estimé 3 modèles. La question qui se pose: quel est le meilleur modèle ?

Pour répondre à cette question, on doit définir ce que veut dire “meilleur”. Dans ce cas, on utilisera RMSE comme métrique et la validation croisée comme approche de re-échantillonnage.

```
set.seed(123)
(cv_mod1 <- train(form = medv ~ rm,
  data = train, method = "lm",
  trControl = trainControl(method = "cv", number = 10)))
```

```
Linear Regression

354 samples
  1 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 318, 319, 318, 319, 319, 319, ...
Resampling results:

  RMSE      Rsquared   MAE
  6.568065  0.4999459  4.517952

Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
set.seed(123)
(cv_mod2 <- train(form = medv ~ rm + lstat,
  data = train, method = "lm",
  trControl = trainControl(method = "cv", number = 10)))
```

```
Linear Regression

354 samples
  2 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 318, 319, 318, 319, 319, 319, ...
Resampling results:

  RMSE     Rsquared   MAE
  5.50875  0.6357689  4.0166

Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
set.seed(123)
(cv_mod3 <- train(form = medv ~ rm + lstat + age,
  data = train, method = "lm",
  trControl = trainControl(method = "cv", number = 10)))
```

```
Linear Regression

354 samples
  3 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 318, 319, 318, 319, 319, 319, ...
Resampling results:

  RMSE      Rsquared   MAE
  5.577216  0.6290893  4.048824

Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
# Extract out of sample performance measures
summary(resamples(list(model1 = cv_mod1, model2 = cv_mod2,
                       model3 = cv_mod3)))
```

```
Call:
summary.resamples(object = resamples(list(model1 = cv_mod1, model2 =
 cv_mod2, model3 = cv_mod3)))

Models: model1, model2, model3
Number of resamples: 10

MAE
           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
model1 3.679413 3.789617 4.171966 4.517952 4.755100 7.561154    0
model2 2.666590 3.587005 3.803046 4.016600 4.392750 6.279433    0
model3 2.810045 3.649151 3.806138 4.048824 4.410426 6.322432    0

RMSE
           Min.  1st Qu.   Median     Mean  3rd Qu.      Max. NA's
model1 4.994818 5.231666 6.008484 6.568065 7.007573 12.070129    0
model2 3.132182 4.933731 5.178013 5.508750 6.024353  9.429550    0
model3 3.311719 4.892822 5.251309 5.577216 6.236043  9.544912    0

Rsquared
             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
model1 0.04276741 0.3401524 0.5536225 0.4999459 0.6888395 0.7678734    0
model2 0.32808065 0.5486184 0.6440678 0.6357689 0.7665443 0.8070292    0
model3 0.31286706 0.5440675 0.6521460 0.6290893 0.7546747 0.7883733    0
```

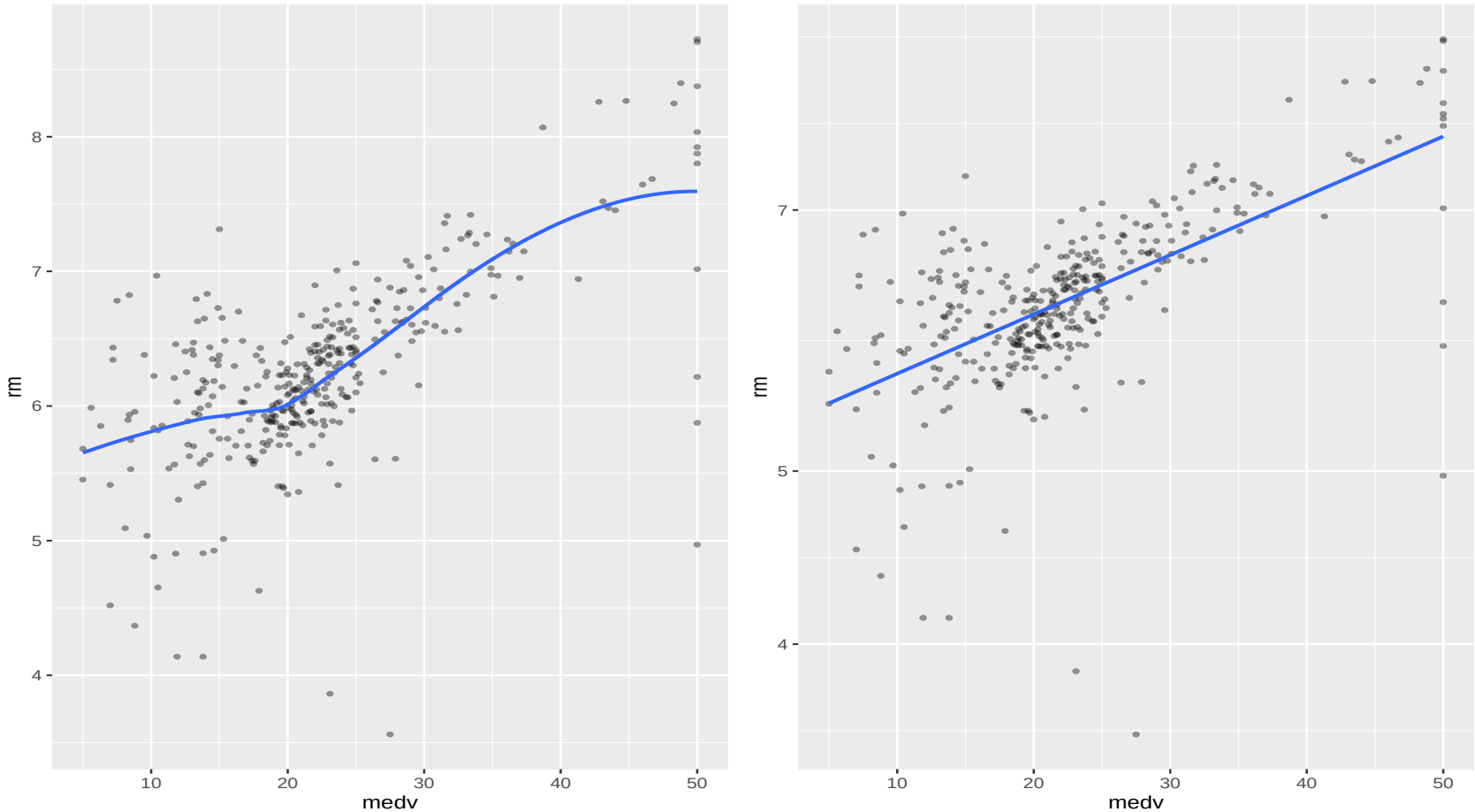
L'estimation du modèle linéaire se repose sur le respect de quelques hypothèses (linéarité, variance constante, erreurs non corrolées, . . .)

L'estimation du modèle linéaire se repose sur le respect de quelques hypothèses (linéarité, variance constante, erreurs non corrolées, . . .)

Donc, l'étape de la validation consiste à vérifier si ces hypothèses sont bien remplies à travers des graphiques et des tests statistiques.

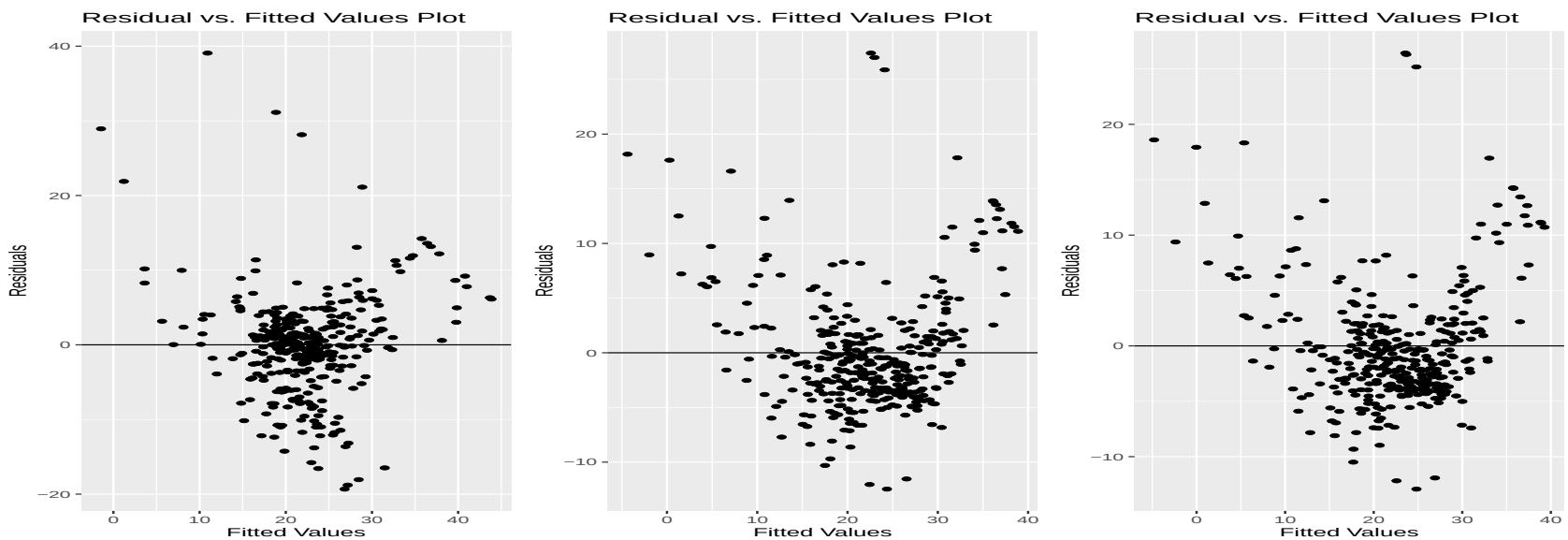
```
p1=ggplot(train,aes(medv, rm))+geom_point(size=1, alpha=0.4)+geom_smooth(se=F)

p2=ggplot(train,aes(medv, rm))+geom_point(size=1, alpha=0.4)+geom_smooth(method="lm",se=F)+
  scale_y_log10("rm")
```

```
gridExtra::grid.arrange(p1,p2, nrow=1)
```



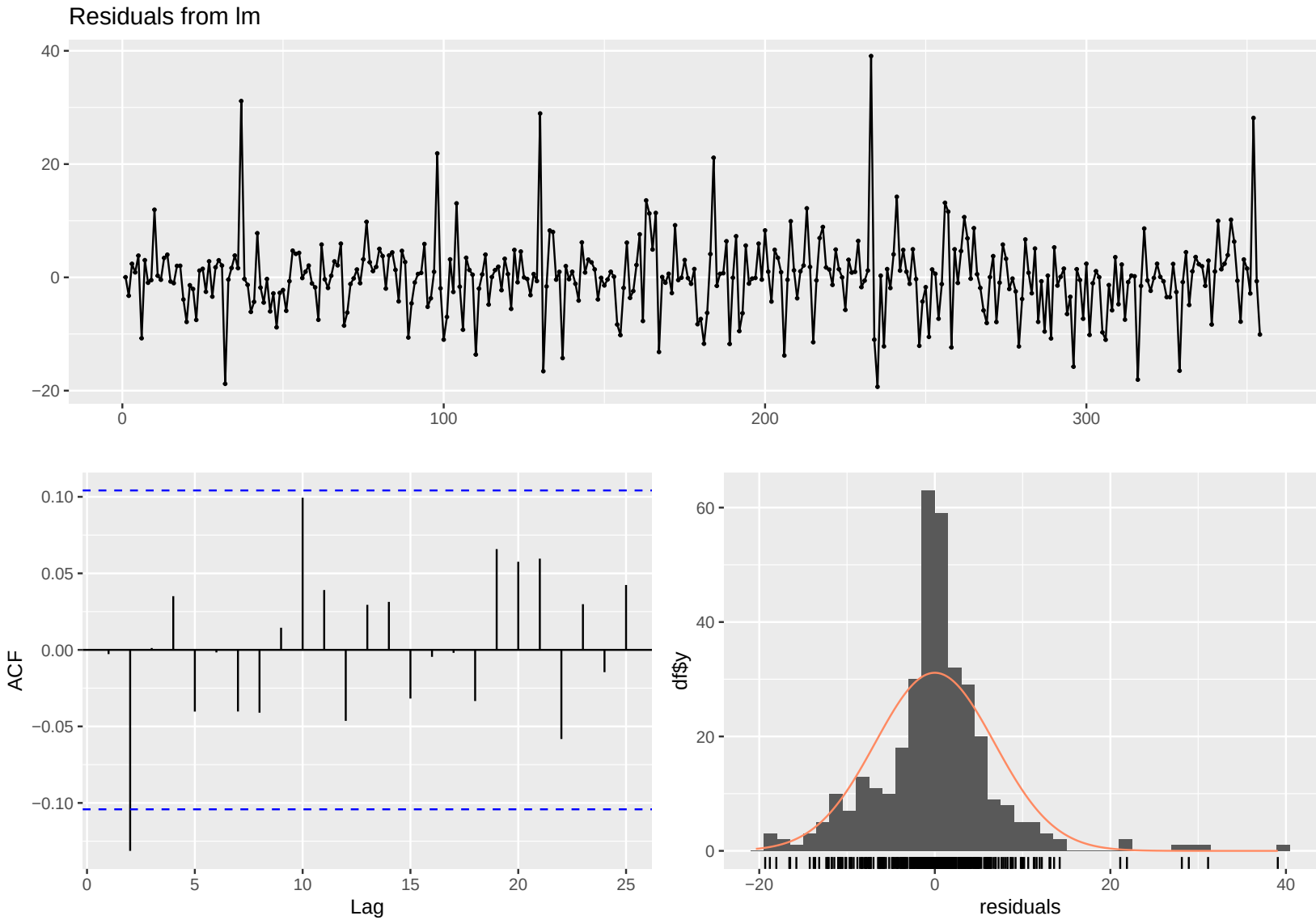
L'homoscédasticité:

```
p11=ggplot(cv_mod1$finalModel, aes(x = .fitted, y = .resid))+ geom_point()+
  geom_hline(yintercept = 0) +  labs(title='Residual vs. Fitted Values Plot', x='Fitted Values',
                                     y='Residuals')

p12=ggplot(cv_mod2$finalModel, aes(x = .fitted, y = .resid)) +  geom_point() +
 geom_hline(yintercept = 0) +   labs(title='Residual vs. Fitted Values Plot', x='Fitted Values',
                                     y='Residuals')

p13=ggplot(cv_mod3$finalModel, aes(x = .fitted, y = .resid)) +geom_point() +
  geom_hline(yintercept = 0) +  labs(title='Residual vs. Fitted Values Plot', x='Fitted Values',
                                     y='Residuals')

gridExtra::grid.arrange(p11,p12,p13,nrow=1)
```



L'autocorrelation: pour tester graphiquement si les erreurs sont indépendantes, on peut faire recours à la fonction `checkresiduals()`de la bibliothèque `forecast`.

```
forecast::checkresiduals(cv_mod1)
```



Une fois, le modèle est validé, on passe à tester la précision des prévisions du modèle. Ceci se fait en déterminant la prévision sur la partie `test` et la comparer par la vrai valeur.

```
pred=predict(cv_mod1, newdata = test)
rmse=sqrt(mean((test$medv-pred)^2)); rmse
```

```
[1] 6.177801
```

```
rmae=sqrt(mean(abs(test$medv-pred))); rmae
```

```
[1] 2.103623
```

```
dd=data.frame(TrueVal=test$medv,PredVal=pred)
p3=ggplot(dd, aes(x=1:nrow(dd)))+
  geom_line(aes(y=TrueVal))+
  geom_line(aes(y=PredVal), color="darkred")+
  labs(x="", y="")
p3
```