TP_économétrie_fish_dataset

Aymeric COUPRIE

08/01/2022

https://github.com/aymericCOUPRIE/Fish_econometrie.git
(https://github.com/aymericCOUPRIE/Fish_econometrie.git)

Peut-on prédire si les poissons appartiennent à l'espèce étudiée en fonction de leur mensuration ?

Choix des technologies

Pour faire ce sujet nous avions la possibilité de choisir le langage de programmation souhaité. J'ai donc choisis d'utiliser R afin de mettre en pratique ce que nous avons vu en cours.

Récupération des données

Ici, on a un jeu de données sous format csv, qu'on va lire et importer en R, en utilisant le délimiteur ";" et en gardant les en-tête des colonnes.

```
# Lecture du csv
fish_dataset <- read.table("Fish.csv", header = TRUE, sep = ";")
```

Représentation des données

Cette phase permet d'avoir quelques informations sur le dataset que l'on vient d'importer. On a donc le nombre de variables contenues dans le dataset. Dans un deuxième temps on affiche les n premières lignes (5 par défaut) du dataset afin d'avoir un aperçu générales des données contenues.

```
str(fish_dataset)
```

```
## 'data.frame':    111 obs. of  4 variables:
##  $ Species: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Weight : num  242 290 340 363 430 450 500 390 450 500 ...
##  $ Height : num  11.5 12.5 12.4 12.7 12.4 ...
##  $ Width  : num  4.02 4.31 4.7 4.46 5.13 ...
```

```
head(fish_dataset)
```

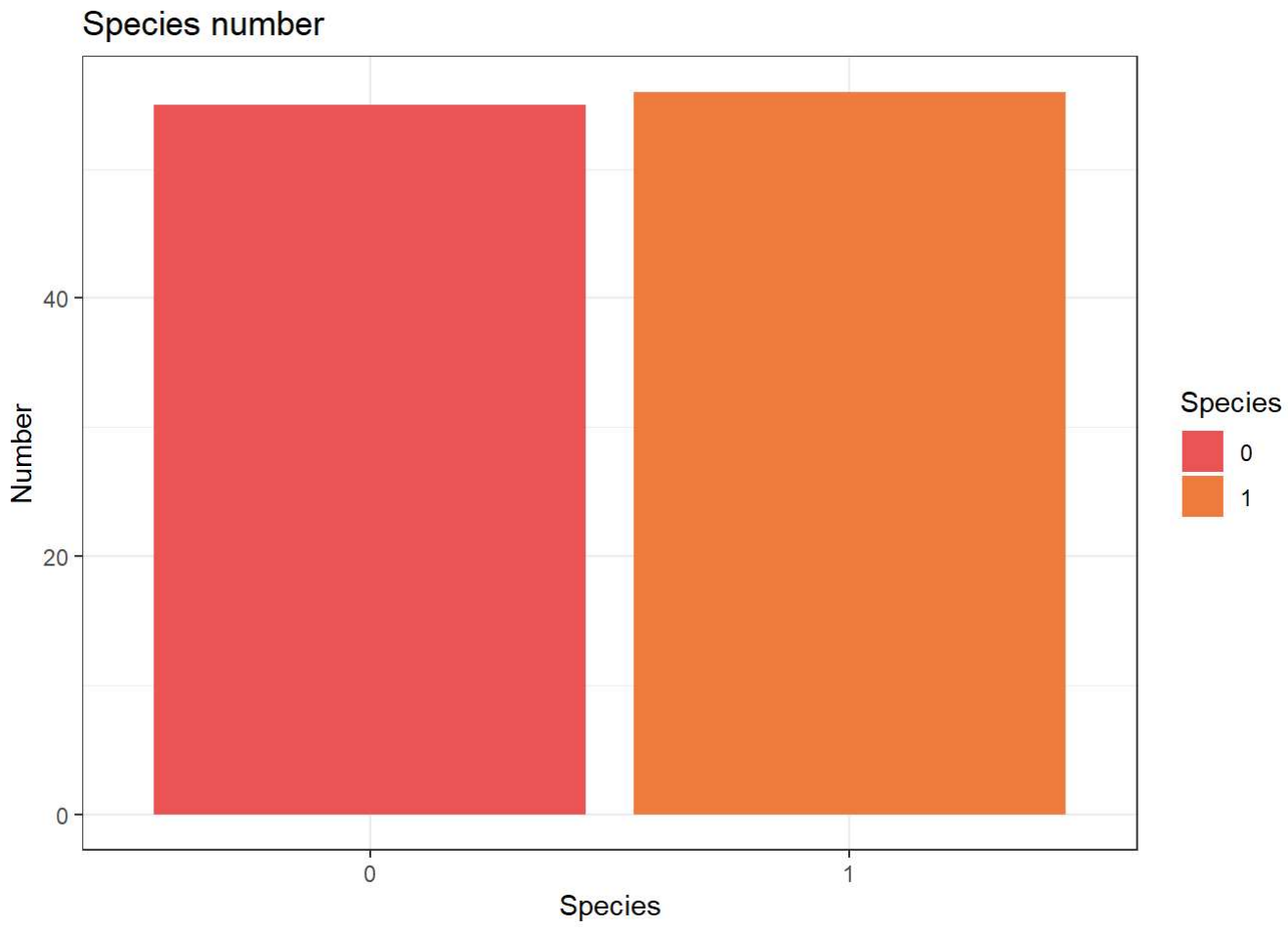
	Species <int>	Weight <dbl>	Height <dbl>	Width <dbl>
1	0	242	11.5200	4.0200

	Species <int>	Weight <dbl>	Height <dbl>	Width <dbl>
2	0	290	12.4800	4.3056
3	0	340	12.3778	4.6961
4	0	363	12.7300	4.4555
5	0	430	12.4440	5.1340
6	0	450	13.6024	4.9274

6 rows

Pour continuer la visualisation rapide des données on va afficher un graphe qui permet de séparer les 2 espèces et d'avoir le nombre de poissons pour chacune.

```
# Graphe répartion des effectifs des espèces
ggplot(fish_dataset, aes(x = as.factor(Species))) +
  geom_bar(aes(fill = as.factor(Species))) +
  scale_fill_manual(values = colors) +
  xlab("Species") +
  ylab("Number") +
  ggtitle("Species number") +
  labs(fill = "Species")
```



Séparation des données

Lors de cette phase on sépare le dataset en 2 : On décompose les données en un échantillon d'apprentissage utilisé pour apprendre le modèle contenant 70% des données et un échantillon de test tester les performances en prédiction du modèle (et sa capacité de généralisation) comprenant les 30% des données restantes.

```
# taille de l'échantillon
n <- nrow(fish_dataset)

train_index <- sample(x = 1:n, size = round(0.7 * n), replace = FALSE)

# Répartion du dataset de base
train_dataset <- fish_dataset[train_index,]
test_dataset <- fish_dataset[-train_index,]
```

Training du model

Pour la prochaine étape, on va essayer de prédire à quelle espèce le poisson étudiée appartient.

Régression backward

Ici on tente d'améliorer le modèle en partant du modèle complet puis en essayant de retirer des colonnes qui pourraient fausser la précision du modèle.

```
# Apprentissage
log_reg2 <- glm(Species ~ ., data = train_dataset, family="binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
log_reg2 <- step(log_reg2, direction="backward")
```

```
## Start:  AIC=26.4
## Species ~ Weight + Height + Width
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##          Df Deviance     AIC
## - Width   1   18.641  24.641
## <none>        18.405  26.405
## - Weight  1   34.044  40.044
## - Height  1   97.420 103.420
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##
## Step:  AIC=24.64
## Species ~ Weight + Height
##
##          Df Deviance     AIC
## <none>        18.641  24.641
## - Weight  1   72.383  76.383
## - Height  1  102.727 106.727
```

```
summary(log_reg2)
```

```
##
## Call:
## glm(formula = Species ~ Weight + Height, family = "binomial",
##     data = train_dataset)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.85115   0.00000   0.00000   0.02486   1.68939
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  54.36631   22.72302   2.393   0.0167 *
## Weight        0.14637    0.06083   2.406   0.0161 *
## Height      -11.67713    4.87573  -2.395   0.0166 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 108.080  on 77  degrees of freedom
## Residual deviance:  18.641  on 75  degrees of freedom
## AIC: 24.641
##
## Number of Fisher Scoring iterations: 10
```

```
# Prédiction
hat_pi2 <- predict(log_reg2, newdata = test_dataset, type = "response")
hat_y2 <- as.integer(hat_pi2 > 0.5)
```

Régression forward

C'est le même principe que la sélection backward sauf que cette fois ci. On part du modèle vide et on ajoute les colonnes une à une afin de terminer avec le modèle complet et ensuite on regarde quel combinaison de colonnes offre le modèle le plus performant.

```
# Apprenstissage
log_reg3 <- glm(Species ~ 1, data = train_dataset, family="binomial")
log_reg3 <- step(log_reg3, direction="forward", scope=list(lower=log_reg3, upper=~Weight+Height+
Width))
```

```
## Start:  AIC=110.08
## Species ~ 1
##
##          Df Deviance     AIC
## + Height  1   72.383  76.383
## + Weight  1  102.727 106.727
## + Width   1  105.985 109.985
## <none>       108.080 110.080
##
## Step:  AIC=76.38
## Species ~ Height
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##          Df Deviance    AIC
## + Weight  1   18.641 24.641
## + Width   1   34.044 40.044
## <none>        72.383 76.383
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##
## Step:  AIC=24.64
## Species ~ Height + Weight
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##         Df Deviance    AIC
## <none>       18.641 24.641
## + Width  1   18.405 26.405
```

```
summary(log_reg3)
```

```
##
## Call:
## glm(formula = Species ~ Height + Weight, family = "binomial",
##     data = train_dataset)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.85115   0.00000   0.00000   0.02486   1.68939
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  54.36631   22.72302   2.393   0.0167 *
## Height      -11.67713    4.87573  -2.395   0.0166 *
## Weight        0.14637    0.06083   2.406   0.0161 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 108.080  on 77  degrees of freedom
## Residual deviance:  18.641  on 75  degrees of freedom
## AIC: 24.641
##
## Number of Fisher Scoring iterations: 10
```

```
# Prédiction
hat_pi3 <- predict(log_reg3, newdata = test_dataset, type = "response")
hat_y3 <- as.integer(hat_pi3 > 0.5)
```

On peut voir que la sélection backward et forward donnent le même résultat. Afin d'obtenir un meilleur score il ne faut sélectionner que les colonnes Height & Weight.

Matrices de confusion

Matrice de confusion pour le système de régression avec sélection des bonnes colonnes. Ici on se sert du résultat de la sélection backward mais utiliser la sélection forward aurait abouti au même résultat

```
# Matrice de confusion
confusionMatrix(data = as.factor(hat_y2), reference = as.factor(test_dataset$Species), positive
 = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 10  1
##          1  5 17
##
##                Accuracy : 0.8182
##                  95% CI : (0.6454, 0.9302)
##     No Information Rate : 0.5455
##     P-Value [Acc > NIR] : 0.001007
##
##                   Kappa : 0.625
##
##  Mcnemar's Test P-Value : 0.220671
##
##             Sensitivity : 0.9444
##             Specificity : 0.6667
##          Pos Pred Value : 0.7727
##          Neg Pred Value : 0.9091
##              Prevalence : 0.5455
##          Detection Rate : 0.5152
##    Detection Prevalence : 0.6667
##       Balanced Accuracy : 0.8056
##
##        'Positive' Class : 1
##
```

Grâce à ce modèle on trouve une accuracy de **0.8182**.
- Le nombre supérieur gauche : vrais positifs (espèce 1, classé 1)
- Le nombre supérieur droit : faux positifs (espèce 0, classé 1)
- Le nombre en bas à gauche : faux négatifs (espèce 1, classé 0)
- Le nombre en bas à droite : vrais négatifs (espèce 0, classé 0)

On peut voir que le ratio de positifs est plutot bon, de meme qu'il y a peu de négatifs. Le modèle semble donc bien fonctionner

Remarque

En raison du nombre de données assez faible, on peut avoir des résultats qui varient selon la répartition des données entre le dataset de train et de test. En effet l'accuracy finale peut varier énormément en raison d'une approche parfois différente, selon la répartition des données, il peut notement s'avérer que la colonne Weight soit utile.