Thème : Arbres de décision TP 3

Prenez soin de bien comprendre les fonctions et commandes de **R** utilisées dans ce travail pratique et ne vous contentez pas d'effectuer une simple copie dans votre session de **R** des commandes se trouvant dans l'énoncé.

Ce travail pratique peut être réalisé par groupes de deux étudiant·e·s. Dans vos recherches bibliographiques, indiquez clairement et précisément vos sources et ne vous limitez pas à Wikipédia, l'encyclopédie libre.

Par rapport au rendu du travail pratique précédent, étoffez celui de ce travail pratique par de nouvelles possibilités que vous offre **rmarkdown**.

Exercice 1

Dans cet exercice, nous introduirons les arbres de classification implémentés par T. M. Therneau et E. J. Atkinson en 1997.

Les laboratoires de Hewlett–Packard ont collecté 4601 messages électroniques dans lesquels figurent 1813 messages non désirés (spams). Parmi les 57 variables permettant de mieux connaître les caractéristiques d'un message spam figurent la longueur totale des mots écrits en majuscules dans un message électronique (**crl.tot**), le pourcentage de $ (**dollar**), le pourcentage de ! (**bang**), le pourcentage de "money" (**money**), le pourcentage de la chaîne de caractères "000" (**n000**) ainsi que le pourcentage de "make" (**make**). Les pourcentages pour les symboles $ et ! sont calculés par rapport au nombre total de caractères et ceux pour les mots ou symboles "money", "000" et "make" ont été déterminés par rapport au nombre total de mots relevés. On attribue un **y** à la variable **yesno** si le message électronique est un spam et un **n** sinon.

On se propose de construire un arbre de classification en prenant la variable **yesno** comme variable réponse et les caractéristiques des messages électroniques comme variables explicatives.

a) Lire les pages 102 à 108 du chapitre 11 "Introduction au data mining orienté vers le business" du support de cours de "Probabilités et Statistique" vous présentant la classification et les arbres de classification.

b) Les arbres de classification introduits par T. M. Therneau et E. J. Atkinson sont disponibles dans la librairie **rpart** de **R**. Il faut d'abord installer la librairie puis la charger dans votre session.

c) Les données se trouvent dans la librairie **DAAG**. Il convient dans un premier temps de télécharger la librairie puis de la charger dans votre session. Dans un second temps, les données sont à charger à leur tour dans votre session. Elles se trouvent dans l'objet **spam7**.

d) On se propose maintenant de construire un arbre de classification à l'aide de la librairie **rpart** par les commandes

```
set.seed(010666)
spam.ct<-rpart(formula=yesno~crl.tot+dollar+bang+money+n000+make,
               method="class", data=spam7, cp=0.001)
```

e) Pour en obtenir un résumé succinct, taper la commande

```
print(spam.ct)
```

```
## n= 4601
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 4601 1813 n (0.60595523 0.39404477)
##     2) dollar< 0.0555 3471  816 n (0.76490925 0.23509075)
##       4) bang< 0.0915 2420  246 n (0.89834711 0.10165289)
##         8) n000< 0.26 2399  232 n (0.90329304 0.09670696) *
##         9) n000>=0.26 21    7 y (0.33333333 0.66666667)
##          18) crl.tot< 83 7    2 n (0.71428571 0.28571429) *
##          19) crl.tot>=83 14    2 y (0.14285714 0.85714286) *
##       5) bang>=0.0915 1051  481 y (0.45765937 0.54234063)
##        10) crl.tot< 85.5 535  175 n (0.67289720 0.32710280)
##          20) bang< 0.7735 418  106 n (0.74641148 0.25358852)
##            40) money< 0.835 408   97 n (0.76225490 0.23774510)
##              80) crl.tot< 51.5 255   40 n (0.84313725 0.15686275) *
##              81) crl.tot>=51.5 153   57 n (0.62745098 0.37254902)
##               162) bang< 0.4065 115   31 n (0.73043478 0.26956522) *
##               163) bang>=0.4065 38   12 y (0.31578947 0.68421053) *
##            41) money>=0.835 10    1 y (0.10000000 0.90000000) *
##          21) bang>=0.7735 117   48 y (0.41025641 0.58974359)
##            42) crl.tot< 17 43   12 n (0.72093023 0.27906977)
##              84) bang< 3.959 35    6 n (0.82857143 0.17142857) *
##              85) bang>=3.959 8    2 y (0.25000000 0.75000000) *
##            43) crl.tot>=17 74   17 y (0.22972973 0.77027027)
##              86) bang>=4.1605 7    1 n (0.85714286 0.14285714) *
##              87) bang< 4.1605 67   11 y (0.16417910 0.83582090) *
##        11) crl.tot>=85.5 516  121 y (0.23449612 0.76550388)
##          22) bang< 0.1955 152   71 y (0.46710526 0.53289474)
##            44) money< 0.04 117   47 n (0.59829060 0.40170940)
##              88) dollar< 0.0085 106   38 n (0.64150943 0.35849057)
##               176) bang< 0.108 15    0 n (1.00000000 0.00000000) *
##               177) bang>=0.108 91   38 n (0.58241758 0.41758242)
##                 354) crl.tot>=125 58   18 n (0.68965517 0.31034483) *
##                 355) crl.tot< 125 33   13 y (0.39393939 0.60606061) *
##              89) dollar>=0.0085 11    2 y (0.18181818 0.81818182) *
##            45) money>=0.04 35    1 y (0.02857143 0.97142857) *
##          23) bang>=0.1955 364   50 y (0.13736264 0.86263736) *
##     3) dollar>=0.0555 1130  133 y (0.11769912 0.88230088)
```

```
##       6) bang< 0.0495 235   89 y (0.37872340 0.62127660)
##        12) money< 0.025 141   59 n (0.58156028 0.41843972)
##          24) n000< 0.465 126   44 n (0.65079365 0.34920635)
##            48) dollar< 0.1665 70   11 n (0.84285714 0.15714286) *
##            49) dollar>=0.1665 56   23 y (0.41071429 0.58928571)
##              98) dollar>=0.1925 45   22 n (0.51111111 0.48888889)
##               196) dollar< 1.012 38   16 n (0.57894737 0.42105263)
##                 392) dollar>=0.3595 17    4 n (0.76470588 0.23529412) *
##                 393) dollar< 0.3595 21    9 y (0.42857143 0.57142857)
##                   786) crl.tot< 54.5 7    2 n (0.71428571 0.28571429) *
##                   787) crl.tot>=54.5 14    4 y (0.28571429 0.71428571) *
##               197) dollar>=1.012 7    1 y (0.14285714 0.85714286) *
##              99) dollar< 0.1925 11    0 y (0.00000000 1.00000000) *
##          25) n000>=0.465 15    0 y (0.00000000 1.00000000) *
##        13) money>=0.025 94    7 y (0.07446809 0.92553191) *
##       7) bang>=0.0495 895   44 y (0.04916201 0.95083799) *
```

f) Un résumé plus détaillé de l'arbre de classification peut être obtenu à l'aide de la commande

```
summary(spam.ct)
```

```
## Call:
## rpart(formula = yesno ~ crl.tot + dollar + bang + money + n000 +
##     make, data = spam7, method = "class", cp = 0.001)
##   n= 4601
##
##             CP nsplit rel error    xerror       xstd
## 1  0.476558191      0 1.0000000 1.0000000 0.01828190
## 2  0.075565361      1 0.5234418 0.5515720 0.01543077
## 3  0.011583012      3 0.3723111 0.3877551 0.01346092
## 4  0.010479868      4 0.3607281 0.3844457 0.01341366
## 5  0.006343078      5 0.3502482 0.3723111 0.01323741
## 6  0.005515720     10 0.3166023 0.3541092 0.01296392
## 7  0.004412576     11 0.3110866 0.3502482 0.01290444
## 8  0.003861004     12 0.3066740 0.3353558 0.01267003
## 9  0.002757860     16 0.2912300 0.3270822 0.01253624
## 10 0.002206288     17 0.2884721 0.3166023 0.01236297
## 11 0.001930502     18 0.2862659 0.3243243 0.01249106
## 12 0.001654716     20 0.2824049 0.3215665 0.01244559
## 13 0.001000000     25 0.2741313 0.3215665 0.01244559
##
## Variable importance
##  dollar    bang   money    n000 crl.tot    make
##      36      20      16      15      12       2
##
## Node number 1: 4601 observations,    complexity param=0.4765582
##   predicted class=n  expected loss=0.3940448  P(node) =1
##     class counts:  2788  1813
##    probabilities: 0.606 0.394
##   left son=2 (3471 obs) right son=3 (1130 obs)
```

```
##   Primary splits:
##       dollar  < 0.0555 to the left,  improve=714.1697, (0 missing)
##       bang    < 0.0795 to the left,  improve=711.9638, (0 missing)
##       money   < 0.01   to the left,  improve=496.0482, (0 missing)
##       n000    < 0.125  to the left,  improve=398.2549, (0 missing)
##       crl.tot < 71.5   to the left,  improve=347.1149, (0 missing)
##   Surrogate splits:
##       n000    < 0.055  to the left,  agree=0.839, adj=0.346, (0 split)
##       money   < 0.045  to the left,  agree=0.833, adj=0.321, (0 split)
##       crl.tot < 693.5  to the left,  agree=0.790, adj=0.143, (0 split)
##       make    < 0.315  to the left,  agree=0.762, adj=0.030, (0 split)
##
## Node number 2: 3471 observations,    complexity param=0.07556536
##   predicted class=n  expected loss=0.2350908  P(node) =0.7544012
##     class counts:  2655   816
##    probabilities: 0.765 0.235
##   left son=4 (2420 obs) right son=5 (1051 obs)
##   Primary splits:
##       bang    < 0.0915 to the left,  improve=284.61340, (0 missing)
##       money   < 0.075  to the left,  improve=123.58970, (0 missing)
##       crl.tot < 77.5   to the left,  improve= 90.53332, (0 missing)
##       n000    < 0.26   to the left,  improve= 78.08758, (0 missing)
##       dollar  < 0.0125 to the left,  improve= 24.73518, (0 missing)
##   Surrogate splits:
##       money < 0.075  to the left,  agree=0.723, adj=0.086, (0 split)
##       n000  < 0.26   to the left,  agree=0.710, adj=0.043, (0 split)
##       make  < 3.97   to the left,  agree=0.697, adj=0.001, (0 split)
##
## Node number 3: 1130 observations,    complexity param=0.006343078
##   predicted class=y  expected loss=0.1176991  P(node) =0.2455988
##     class counts:   133   997
##    probabilities: 0.118 0.882
##   left son=6 (235 obs) right son=7 (895 obs)
##   Primary splits:
##       bang    < 0.0495 to the left,  improve=40.431060, (0 missing)
##       money   < 0.02   to the left,  improve=17.281430, (0 missing)
##       n000    < 0.235  to the left,  improve=12.261890, (0 missing)
##       make    < 0.065  to the left,  improve=10.767510, (0 missing)
##       crl.tot < 112.5  to the left,  improve= 9.826864, (0 missing)
##   Surrogate splits:
##       crl.tot < 31.5   to the left,  agree=0.801, adj=0.043, (0 split)
##
## Node number 4: 2420 observations,    complexity param=0.003861004
##   predicted class=n  expected loss=0.1016529  P(node) =0.5259726
##     class counts:  2174   246
##    probabilities: 0.898 0.102
##   left son=8 (2399 obs) right son=9 (21 obs)
##   Primary splits:
##       n000    < 0.26   to the left,  improve=13.525470, (0 missing)
##       money   < 0.01   to the left,  improve=12.980300, (0 missing)
##       crl.tot < 29.5   to the left,  improve= 9.289818, (0 missing)
##       bang    < 0.0285 to the left,  improve= 7.974710, (0 missing)
```

```
##       dollar  < 0.0395 to the left,  improve= 3.709035, (0 missing)
##
## Node number 5: 1051 observations,    complexity param=0.07556536
##   predicted class=y  expected loss=0.4576594  P(node) =0.2284286
##     class counts:   481   570
##    probabilities: 0.458 0.542
##   left son=10 (535 obs) right son=11 (516 obs)
##   Primary splits:
##       crl.tot < 85.5   to the left,  improve=100.96570, (0 missing)
##       money   < 0.04   to the left,  improve= 46.83813, (0 missing)
##       bang    < 0.4765 to the left,  improve= 39.90369, (0 missing)
##       dollar  < 0.0065 to the left,  improve= 30.70002, (0 missing)
##       n000    < 0.025  to the left,  improve= 28.65365, (0 missing)
##   Surrogate splits:
##       money  < 0.04   to the left,  agree=0.597, adj=0.178, (0 split)
##       dollar < 0.0065 to the left,  agree=0.589, adj=0.163, (0 split)
##       make   < 0.01   to the left,  agree=0.584, adj=0.153, (0 split)
##       n000   < 0.025  to the left,  agree=0.568, adj=0.120, (0 split)
##       bang   < 0.1425 to the right, agree=0.559, adj=0.101, (0 split)
##
## Node number 6: 235 observations,    complexity param=0.006343078
##   predicted class=y  expected loss=0.3787234  P(node) =0.05107585
##     class counts:    89   146
##    probabilities: 0.379 0.621
##   left son=12 (141 obs) right son=13 (94 obs)
##   Primary splits:
##       money   < 0.025  to the left,  improve=29.005670, (0 missing)
##       dollar  < 0.1485 to the left,  improve=19.625870, (0 missing)
##       n000    < 0.435  to the left,  improve=14.670140, (0 missing)
##       crl.tot < 545.5  to the right, improve= 6.372180, (0 missing)
##       make    < 0.135  to the left,  improve= 2.976973, (0 missing)
##   Surrogate splits:
##       n000    < 0.735  to the left,  agree=0.660, adj=0.149, (0 split)
##       dollar  < 0.1505 to the left,  agree=0.638, adj=0.096, (0 split)
##       make    < 0.11   to the left,  agree=0.638, adj=0.096, (0 split)
##       bang    < 0.004  to the left,  agree=0.617, adj=0.043, (0 split)
##       crl.tot < 138    to the left,  agree=0.609, adj=0.021, (0 split)
##
## Node number 7: 895 observations
##   predicted class=y  expected loss=0.04916201  P(node) =0.1945229
##     class counts:    44   851
##    probabilities: 0.049 0.951
##
## Node number 8: 2399 observations
##   predicted class=n  expected loss=0.09670696  P(node) =0.5214084
##     class counts:  2167   232
##    probabilities: 0.903 0.097
##
## Node number 9: 21 observations,    complexity param=0.001654716
##   predicted class=y  expected loss=0.3333333  P(node) =0.004564225
##     class counts:     7    14
##    probabilities: 0.333 0.667
```

```
##   left son=18 (7 obs) right son=19 (14 obs)
##   Primary splits:
##       crl.tot < 83     to the left,  improve=3.0476190, (0 missing)
##       n000    < 0.5    to the right, improve=3.0476190, (0 missing)
##       make    < 0.08   to the right, improve=0.7179487, (0 missing)
##       dollar  < 0.0065 to the left,  improve=0.1794872, (0 missing)
##   Surrogate splits:
##       n000   < 0.5    to the right, agree=0.905, adj=0.714, (0 split)
##       make   < 0.345  to the right, agree=0.810, adj=0.429, (0 split)
##       dollar < 0.0065 to the left,  agree=0.714, adj=0.143, (0 split)
##
## Node number 10: 535 observations,    complexity param=0.01158301
##   predicted class=n  expected loss=0.3271028  P(node) =0.1162791
##     class counts:   360   175
##    probabilities: 0.673 0.327
##   left son=20 (418 obs) right son=21 (117 obs)
##   Primary splits:
##       bang    < 0.7735 to the left,  improve=20.6594000, (0 missing)
##       crl.tot < 51.5   to the left,  improve=10.9080700, (0 missing)
##       money   < 0.835  to the left,  improve= 8.5334580, (0 missing)
##       n000    < 0.36   to the left,  improve= 6.4231100, (0 missing)
##       make    < 1.515  to the left,  improve= 0.8468109, (0 missing)
##   Surrogate splits:
##       crl.tot < 9.5    to the right, agree=0.807, adj=0.120, (0 split)
##       n000    < 0.75   to the left,  agree=0.785, adj=0.017, (0 split)
##
## Node number 11: 516 observations,    complexity param=0.006343078
##   predicted class=y  expected loss=0.2344961  P(node) =0.1121495
##     class counts:   121   395
##    probabilities: 0.234 0.766
##   left son=22 (152 obs) right son=23 (364 obs)
##   Primary splits:
##       bang    < 0.1955 to the left,  improve=23.31715, (0 missing)
##       money   < 0.04   to the left,  improve=13.60065, (0 missing)
##       crl.tot < 185    to the left,  improve= 8.61924, (0 missing)
##       dollar  < 0.0065 to the left,  improve= 7.00789, (0 missing)
##       n000    < 0.115  to the left,  improve= 6.03623, (0 missing)
##   Surrogate splits:
##       crl.tot < 1540.5 to the right, agree=0.713, adj=0.026, (0 split)
##       money   < 1.695  to the right, agree=0.707, adj=0.007, (0 split)
##
## Node number 12: 141 observations,    complexity param=0.006343078
##   predicted class=n  expected loss=0.4184397  P(node) =0.03064551
##     class counts:    82    59
##    probabilities: 0.582 0.418
##   left son=24 (126 obs) right son=25 (15 obs)
##   Primary splits:
##       n000    < 0.465  to the left,  improve=11.3542700, (0 missing)
##       dollar  < 0.1665 to the left,  improve= 8.9933760, (0 missing)
##       crl.tot < 396    to the right, improve= 3.3567280, (0 missing)
##       bang    < 0.015  to the left,  improve= 0.5479230, (0 missing)
##       make    < 0.23   to the left,  improve= 0.3614872, (0 missing)
```

```
##
## Node number 13: 94 observations
##   predicted class=y  expected loss=0.07446809  P(node) =0.02043034
##     class counts:     7    87
##    probabilities: 0.074 0.926
##
## Node number 18: 7 observations
##   predicted class=n  expected loss=0.2857143  P(node) =0.001521408
##     class counts:     5     2
##    probabilities: 0.714 0.286
##
## Node number 19: 14 observations
##   predicted class=y  expected loss=0.1428571  P(node) =0.003042817
##     class counts:     2    12
##    probabilities: 0.143 0.857
##
## Node number 20: 418 observations,    complexity param=0.004412576
##   predicted class=n  expected loss=0.2535885  P(node) =0.09084982
##     class counts:   312   106
##    probabilities: 0.746 0.254
##   left son=40 (408 obs) right son=41 (10 obs)
##   Primary splits:
##       money   < 0.835  to the left,  improve=8.5617830, (0 missing)
##       crl.tot < 51.5   to the left,  improve=8.5457520, (0 missing)
##       bang    < 0.191  to the left,  improve=4.9108310, (0 missing)
##       make    < 0.935  to the right, improve=0.4507922, (0 missing)
##
## Node number 21: 117 observations,    complexity param=0.01047987
##   predicted class=y  expected loss=0.4102564  P(node) =0.02542925
##     class counts:    48    69
##    probabilities: 0.410 0.590
##   left son=42 (43 obs) right son=43 (74 obs)
##   Primary splits:
##       crl.tot < 17     to the left,  improve=13.123870, (0 missing)
##       bang    < 5.027  to the right, improve= 2.440427, (0 missing)
##       make    < 0.14   to the left,  improve= 0.230969, (0 missing)
##   Surrogate splits:
##       bang < 2.9455 to the right, agree=0.641, adj=0.023, (0 split)
##
## Node number 22: 152 observations,    complexity param=0.006343078
##   predicted class=y  expected loss=0.4671053  P(node) =0.0330363
##     class counts:    71    81
##    probabilities: 0.467 0.533
##   left son=44 (117 obs) right son=45 (35 obs)
##   Primary splits:
##       money   < 0.04   to the left,  improve=17.488880, (0 missing)
##       dollar  < 0.0065 to the left,  improve= 7.280843, (0 missing)
##       n000    < 0.335  to the left,  improve= 6.729024, (0 missing)
##       crl.tot < 316    to the left,  improve= 3.592995, (0 missing)
##       bang    < 0.0955 to the right, improve= 1.588042, (0 missing)
##   Surrogate splits:
##       bang < 0.0955 to the right, agree=0.803, adj=0.143, (0 split)
```

```
##       n000 < 0.335  to the left,  agree=0.796, adj=0.114, (0 split)
##
## Node number 23: 364 observations
##   predicted class=y  expected loss=0.1373626  P(node) =0.07911324
##     class counts:    50   314
##    probabilities: 0.137 0.863
##
## Node number 24: 126 observations,    complexity param=0.00551572
##   predicted class=n  expected loss=0.3492063  P(node) =0.02738535
##     class counts:    82    44
##    probabilities: 0.651 0.349
##   left son=48 (70 obs) right son=49 (56 obs)
##   Primary splits:
##       dollar  < 0.1665 to the left,  improve=11.6198400, (0 missing)
##       crl.tot < 302.5  to the right, improve= 3.3587300, (0 missing)
##       n000    < 0.025  to the right, improve= 1.8751040, (0 missing)
##       bang    < 0.001  to the right, improve= 0.9584776, (0 missing)
##       make    < 0.23   to the left,  improve= 0.8253968, (0 missing)
##   Surrogate splits:
##       crl.tot < 91.5   to the right, agree=0.746, adj=0.429, (0 split)
##
## Node number 25: 15 observations
##   predicted class=y  expected loss=0  P(node) =0.003260161
##     class counts:     0    15
##    probabilities: 0.000 1.000
##
## Node number 40: 408 observations,    complexity param=0.003861004
##   predicted class=n  expected loss=0.2377451  P(node) =0.08867637
##     class counts:   311    97
##    probabilities: 0.762 0.238
##   left son=80 (255 obs) right son=81 (153 obs)
##   Primary splits:
##       crl.tot < 51.5   to the left,  improve=8.8970590, (0 missing)
##       bang    < 0.191  to the left,  improve=3.7995550, (0 missing)
##       make    < 0.73   to the right, improve=0.6482843, (0 missing)
##       money   < 0.115  to the left,  improve=0.3074510, (0 missing)
##   Surrogate splits:
##       money < 0.545  to the left,  agree=0.637, adj=0.033, (0 split)
##       bang  < 0.109  to the right, agree=0.635, adj=0.026, (0 split)
##       n000  < 0.36   to the left,  agree=0.627, adj=0.007, (0 split)
##
## Node number 41: 10 observations
##   predicted class=y  expected loss=0.1  P(node) =0.002173441
##     class counts:     1     9
##    probabilities: 0.100 0.900
##
## Node number 42: 43 observations,    complexity param=0.002206288
##   predicted class=n  expected loss=0.2790698  P(node) =0.009345794
##     class counts:    31    12
##    probabilities: 0.721 0.279
##   left son=84 (35 obs) right son=85 (8 obs)
##   Primary splits:
```

```
##       bang    < 3.959  to the left,  improve=4.359468, (0 missing)
##       crl.tot < 14.5   to the left,  improve=0.622587, (0 missing)
##   Surrogate splits:
##       crl.tot < 2.5    to the right, agree=0.86, adj=0.25, (0 split)
##
## Node number 43: 74 observations,    complexity param=0.00275786
##   predicted class=y  expected loss=0.2297297  P(node) =0.01608346
##     class counts:    17    57
##    probabilities: 0.230 0.770
##   left son=86 (7 obs) right son=87 (67 obs)
##   Primary splits:
##       bang    < 4.1605 to the right, improve=6.0868440, (0 missing)
##       crl.tot < 35     to the left,  improve=0.9834476, (0 missing)
##
## Node number 44: 117 observations,    complexity param=0.003861004
##   predicted class=n  expected loss=0.4017094  P(node) =0.02542925
##     class counts:    70    47
##    probabilities: 0.598 0.402
##   left son=88 (106 obs) right son=89 (11 obs)
##   Primary splits:
##       dollar  < 0.0085 to the left,  improve=4.211872, (0 missing)
##       bang    < 0.1065 to the left,  improve=2.838326, (0 missing)
##       crl.tot < 838    to the left,  improve=2.757835, (0 missing)
##       n000    < 0.05   to the left,  improve=1.454901, (0 missing)
##       make    < 0.01   to the right, improve=1.389854, (0 missing)
##   Surrogate splits:
##       crl.tot < 2388   to the left,  agree=0.94, adj=0.364, (0 split)
##
## Node number 45: 35 observations
##   predicted class=y  expected loss=0.02857143  P(node) =0.007607042
##     class counts:     1    34
##    probabilities: 0.029 0.971
##
## Node number 48: 70 observations
##   predicted class=n  expected loss=0.1571429  P(node) =0.01521408
##     class counts:    59    11
##    probabilities: 0.843 0.157
##
## Node number 49: 56 observations,    complexity param=0.001654716
##   predicted class=y  expected loss=0.4107143  P(node) =0.01217127
##     class counts:    23    33
##    probabilities: 0.411 0.589
##   left son=98 (45 obs) right son=99 (11 obs)
##   Primary splits:
##       dollar  < 0.1925 to the right, improve=4.618254, (0 missing)
##       crl.tot < 69     to the left,  improve=2.012949, (0 missing)
##
## Node number 80: 255 observations
##   predicted class=n  expected loss=0.1568627  P(node) =0.05542273
##     class counts:   215    40
##    probabilities: 0.843 0.157
##
```

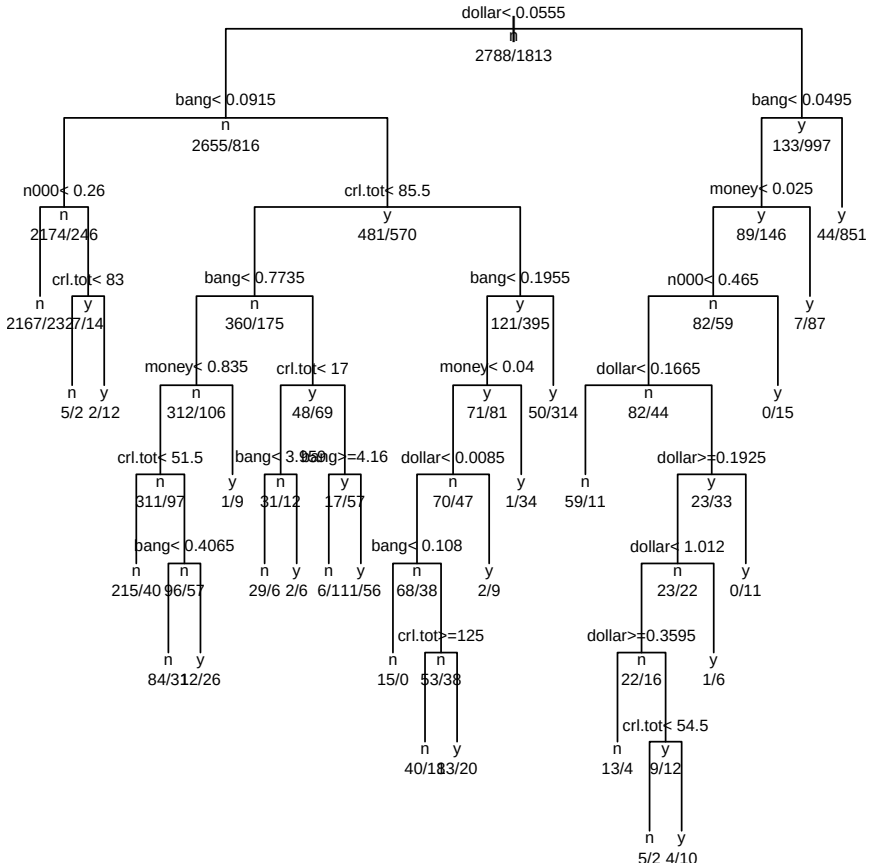
```
## Node number 81: 153 observations,    complexity param=0.003861004
##   predicted class=n  expected loss=0.372549  P(node) =0.03325364
##     class counts:    96    57
##    probabilities: 0.627 0.373
##   left son=162 (115 obs) right son=163 (38 obs)
##   Primary splits:
##       bang    < 0.4065 to the left,  improve=9.821403, (0 missing)
##       make    < 0.255  to the right, improve=1.999168, (0 missing)
##       crl.tot < 81.5   to the right, improve=1.589552, (0 missing)
##
## Node number 84: 35 observations
##   predicted class=n  expected loss=0.1714286  P(node) =0.007607042
##     class counts:    29     6
##    probabilities: 0.829 0.171
##
## Node number 85: 8 observations
##   predicted class=y  expected loss=0.25  P(node) =0.001738752
##     class counts:     2     6
##    probabilities: 0.250 0.750
##
## Node number 86: 7 observations
##   predicted class=n  expected loss=0.1428571  P(node) =0.001521408
##     class counts:     6     1
##    probabilities: 0.857 0.143
##
## Node number 87: 67 observations
##   predicted class=y  expected loss=0.1641791  P(node) =0.01456205
##     class counts:    11    56
##    probabilities: 0.164 0.836
##
## Node number 88: 106 observations,    complexity param=0.001930502
##   predicted class=n  expected loss=0.3584906  P(node) =0.02303847
##     class counts:    68    38
##    probabilities: 0.642 0.358
##   left son=176 (15 obs) right son=177 (91 obs)
##   Primary splits:
##       bang    < 0.108  to the left,  improve=4.490981, (0 missing)
##       crl.tot < 125    to the right, improve=2.939179, (0 missing)
##       make    < 0.01   to the right, improve=1.169594, (0 missing)
##
## Node number 89: 11 observations
##   predicted class=y  expected loss=0.1818182  P(node) =0.002390785
##     class counts:     2     9
##    probabilities: 0.182 0.818
##
## Node number 98: 45 observations,    complexity param=0.001654716
##   predicted class=n  expected loss=0.4888889  P(node) =0.009780483
##     class counts:    23    22
##    probabilities: 0.511 0.489
##   left son=196 (38 obs) right son=197 (7 obs)
##   Primary splits:
##       dollar  < 1.012  to the left,  improve=2.2482870, (0 missing)
```

```
##       crl.tot < 81     to the left,  improve=0.8962963, (0 missing)
##   Surrogate splits:
##       crl.tot < 398.5  to the left,  agree=0.911, adj=0.429, (0 split)
##
## Node number 99: 11 observations
##   predicted class=y  expected loss=0  P(node) =0.002390785
##     class counts:     0    11
##    probabilities: 0.000 1.000
##
## Node number 162: 115 observations
##   predicted class=n  expected loss=0.2695652  P(node) =0.02499457
##     class counts:    84    31
##    probabilities: 0.730 0.270
##
## Node number 163: 38 observations
##   predicted class=y  expected loss=0.3157895  P(node) =0.008259074
##     class counts:    12    26
##    probabilities: 0.316 0.684
##
## Node number 176: 15 observations
##   predicted class=n  expected loss=0  P(node) =0.003260161
##     class counts:    15     0
##    probabilities: 1.000 0.000
##
## Node number 177: 91 observations,    complexity param=0.001930502
##   predicted class=n  expected loss=0.4175824  P(node) =0.01977831
##     class counts:    53    38
##    probabilities: 0.582 0.418
##   left son=354 (58 obs) right son=355 (33 obs)
##   Primary splits:
##       crl.tot < 125    to the right, improve=3.6785740, (0 missing)
##       make    < 0.01   to the right, improve=1.1453150, (0 missing)
##       bang    < 0.1805 to the right, improve=0.7763945, (0 missing)
##   Surrogate splits:
##       bang < 0.12   to the right, agree=0.714, adj=0.212, (0 split)
##       n000 < 0.215  to the left,  agree=0.648, adj=0.030, (0 split)
##
## Node number 196: 38 observations,    complexity param=0.001654716
##   predicted class=n  expected loss=0.4210526  P(node) =0.008259074
##     class counts:    22    16
##    probabilities: 0.579 0.421
##   left son=392 (17 obs) right son=393 (21 obs)
##   Primary splits:
##       dollar  < 0.3595 to the right, improve=2.122954, (0 missing)
##       crl.tot < 81     to the left,  improve=1.002506, (0 missing)
##   Surrogate splits:
##       crl.tot < 27     to the left,  agree=0.632, adj=0.176, (0 split)
##
## Node number 197: 7 observations
##   predicted class=y  expected loss=0.1428571  P(node) =0.001521408
##     class counts:     1     6
##    probabilities: 0.143 0.857
```

```
##
## Node number 354: 58 observations
##   predicted class=n  expected loss=0.3103448  P(node) =0.01260596
##     class counts:    40    18
##    probabilities: 0.690 0.310
##
## Node number 355: 33 observations
##   predicted class=y  expected loss=0.3939394  P(node) =0.007172354
##     class counts:    13    20
##    probabilities: 0.394 0.606
##
## Node number 392: 17 observations
##   predicted class=n  expected loss=0.2352941  P(node) =0.003694849
##     class counts:    13     4
##    probabilities: 0.765 0.235
##
## Node number 393: 21 observations,    complexity param=0.001654716
##   predicted class=y  expected loss=0.4285714  P(node) =0.004564225
##     class counts:     9    12
##    probabilities: 0.429 0.571
##   left son=786 (7 obs) right son=787 (14 obs)
##   Primary splits:
##       crl.tot < 54.5   to the left,  improve=1.714286, (0 missing)
##       dollar  < 0.2895 to the left,  improve=1.714286, (0 missing)
##   Surrogate splits:
##       dollar < 0.2115 to the left,  agree=0.81, adj=0.429, (0 split)
##
## Node number 786: 7 observations
##   predicted class=n  expected loss=0.2857143  P(node) =0.001521408
##     class counts:     5     2
##    probabilities: 0.714 0.286
##
## Node number 787: 14 observations
##   predicted class=y  expected loss=0.2857143  P(node) =0.003042817
##     class counts:     4    10
##    probabilities: 0.286 0.714
```

g) Pour construire graphiquement l'arbre de classification, il suffit d'utiliser les commandes

```
par(pty="s")
plot(spam.ct, uniform=TRUE)
text(spam.ct, use.n=TRUE, all=TRUE, cex=0.6)
```



h) Déterminer le type espéré (spam ou non spam) des deux nouveaux messages électroniques :

	crl.tot	dollar	bang	money	n000	make
1	1257	0.025	0.181	0.15	0.00	0.15
2	112	0.054	0.164	0.00	0.00	0.00

i) Il est aussi possible de connaître la classe espérée des deux messages électroniques en utilisant les commandes suivantes

```
new<-data.frame(crl.tot=c(1257,112), dollar=c(0.025,0.054), bang=c(0.181,0.164),
                money=c(0.15,0.00), n000=c(0.00,0.00), make=c(0.15,0.00))
predict(spam.ct, newdata=new, type="class")

## 1 2
## y y
## Levels: n y
```

Les résultats fournis par **R** sont-ils en accord avec ceux déterminés en h) ?

j) Il nous reste à élaguer l'arbre de classification et le rendre optimal. Le paramètre de complexité (**CP**), valeur comprise entre 0 et 1, a été calculé à chaque division de l'arbre. Il s'agit d'une mesure de la qualité de l'arbre "parfait" par rapport à un arbre plus complexe avec davantage de nœuds. À titre indicatif, le paramètre de complexité est dans notre cas celui qui minimise l'erreur de validation croisée. L'erreur de validation croisée peut être considérée comme une erreur de prédiction. Elle se trouve dans la rubrique **xerror** du tableau affiché à l'écran par les commandes

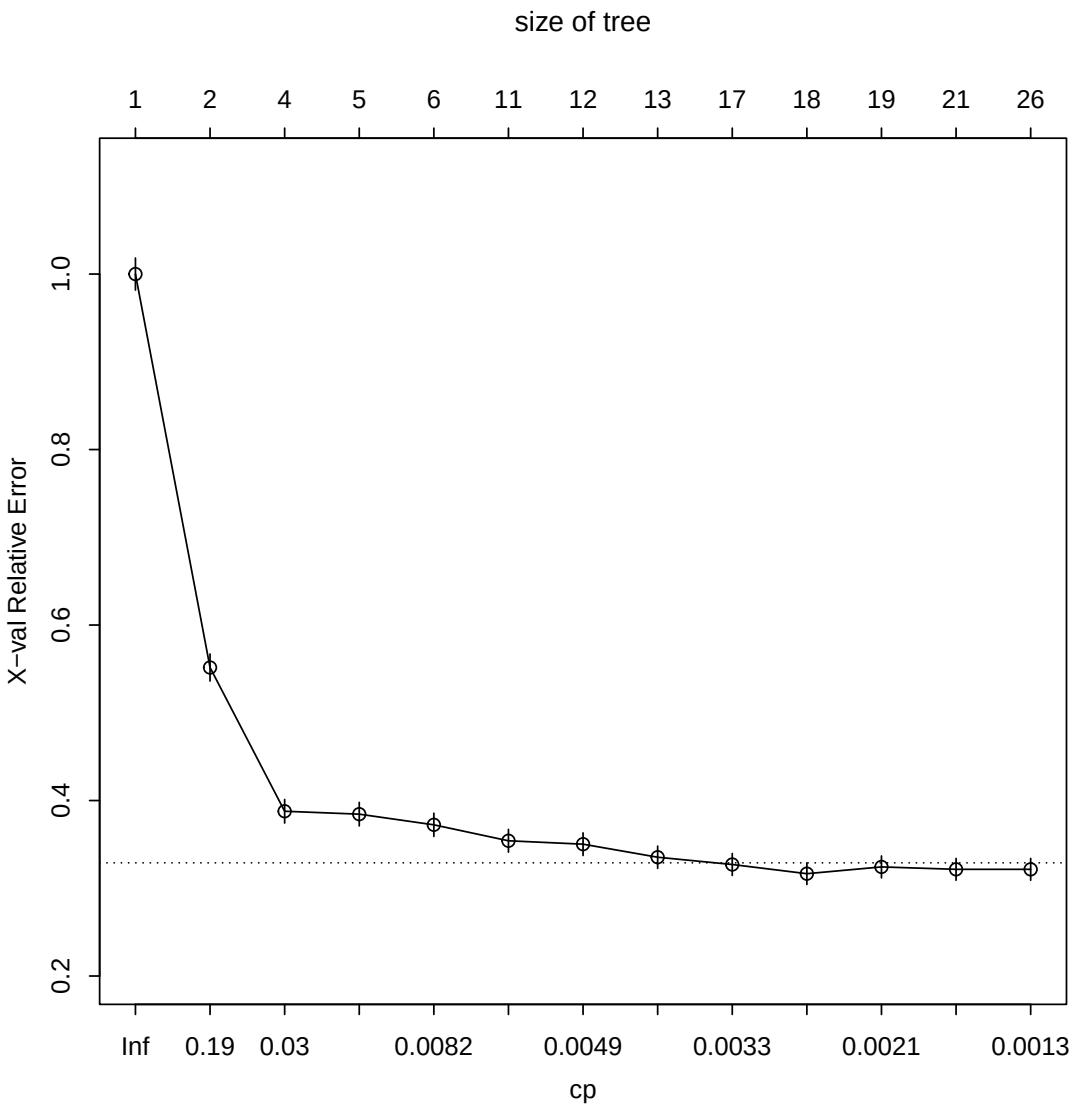
```
options(digits=5)
printcp(spam.ct)

##
## Classification tree:
## rpart(formula = yesno ~ crl.tot + dollar + bang + money + n000 +
##     make, data = spam7, method = "class", cp = 0.001)
##
## Variables actually used in tree construction:
## [1] bang    crl.tot dollar  money   n000
##
## Root node error: 1813/4601 = 0.394
##
## n= 4601
##
##         CP nsplit rel error xerror   xstd
## 1  0.47656      0     1.000  1.000 0.0183
## 2  0.07557      1     0.523  0.552 0.0154
## 3  0.01158      3     0.372  0.388 0.0135
## 4  0.01048      4     0.361  0.384 0.0134
## 5  0.00634      5     0.350  0.372 0.0132
## 6  0.00552     10     0.317  0.354 0.0130
## 7  0.00441     11     0.311  0.350 0.0129
## 8  0.00386     12     0.307  0.335 0.0127
## 9  0.00276     16     0.291  0.327 0.0125
## 10 0.00221     17     0.288  0.317 0.0124
## 11 0.00193     18     0.286  0.324 0.0125
## 12 0.00165     20     0.282  0.322 0.0124
## 13 0.00100     25     0.274  0.322 0.0124
```

Pour rendre l'arbre de classification optimal, on peut choisir le paramètre de complexité qui minimise l'erreur de validation croisée **xerror**. Une procédure alternative consiste à déterminer la plus grande valeur du paramètre de complexité pour faire en sorte que l'erreur de validation croisée correspondante se trouve encore dans un écart-type du minimum. En pratique, il faut déterminer la valeur minimale **xerror** à laquelle on ajoute son écart-type **xstd**. Il faut ensuite déterminer la plus grande valeur du paramètre de complexité telle que son erreur de validation croisée reste inférieure à la valeur minimale **xerror** plus son écart-type. Ce critère s'appelle la règle du "un écart-type". En utilisant cette règle,

1. déterminer le nombre nécessaire de divisions (**nsplit**) pour obtenir un arbre optimal ;
2. en déduire la taille (nombre total de feuilles) de cet arbre.

k) On peut visualiser graphiquement l'application de la règle du "un écart-type" par la commande

```
plotcp(spam.ct)
```



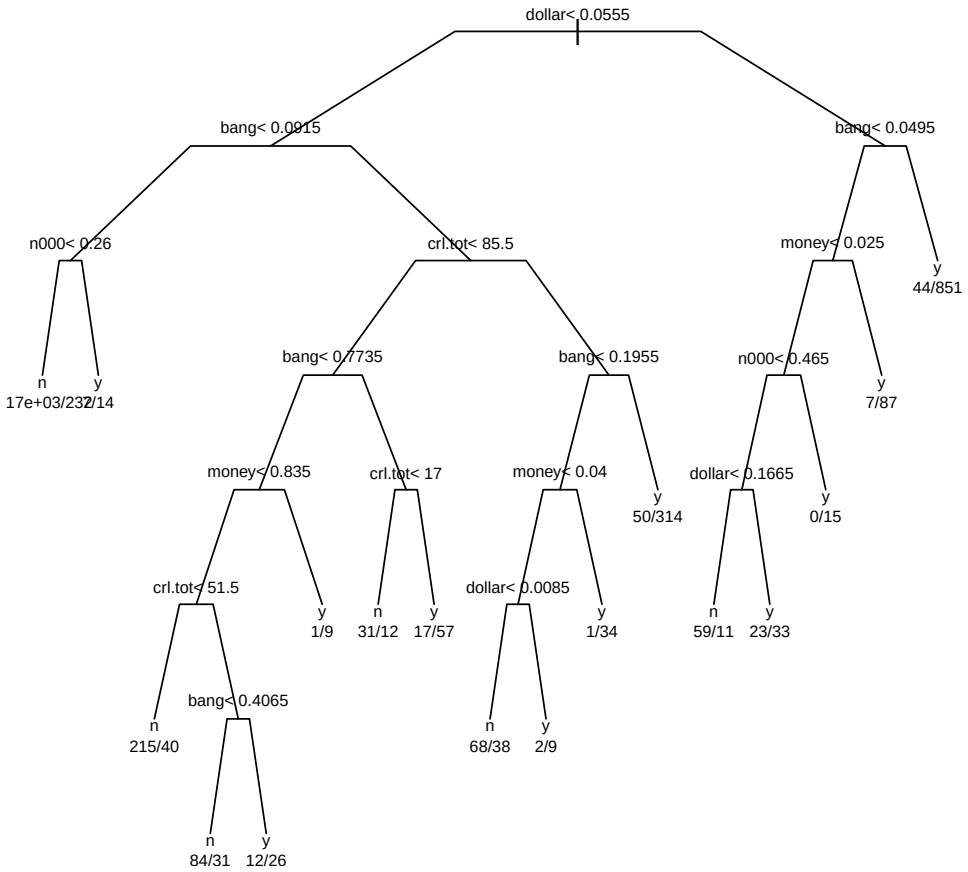
l) Commenter brièvement la qualité de la classification par l'arbre à l'aide des résultats numériques et du tableau de classification obtenus par les commandes suivantes

```
spam.ct1<-prune(spam.ct, cp=0.003)
x<-factor(predict(spam.ct1, type="class"))
table(true=spam7$yesno, predicted=x)

##      predicted
## true    n    y
##    n 2624  164
##    y  364 1449
```

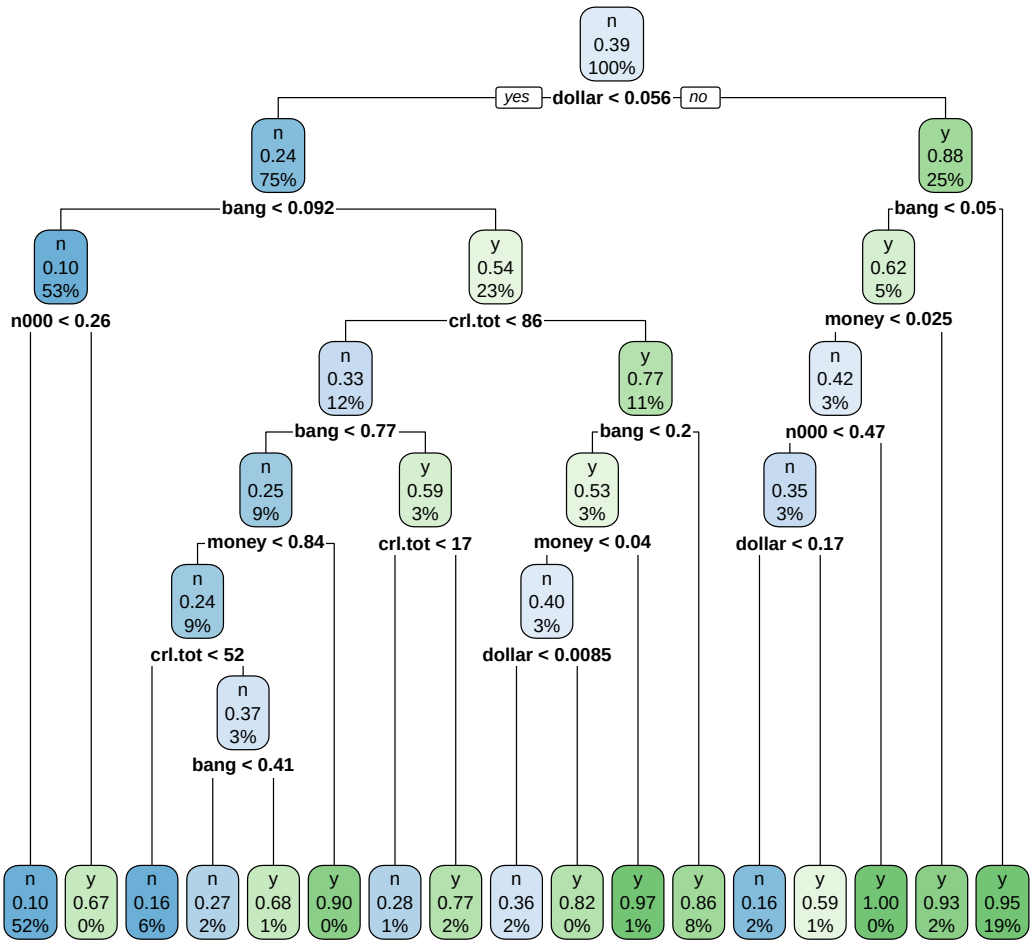
Commenter la matrice de classification.

m) Pour tracer l'arbre de décision final, installer la librairie **rpart.plot**, la charger dans votre session de **R** et construire l'arbre à l'aide des commandes

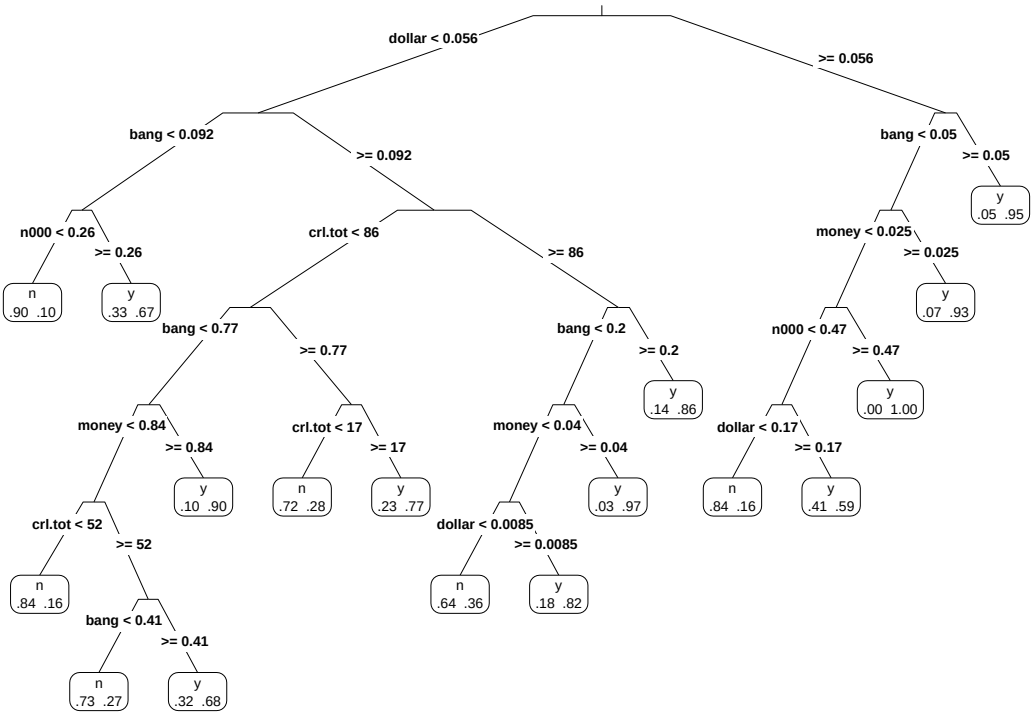
```
plot(spam.ct1, branch=0.4, uniform=TRUE)
text(spam.ct1, digits=3, use.n=TRUE, cex=0.6)
```



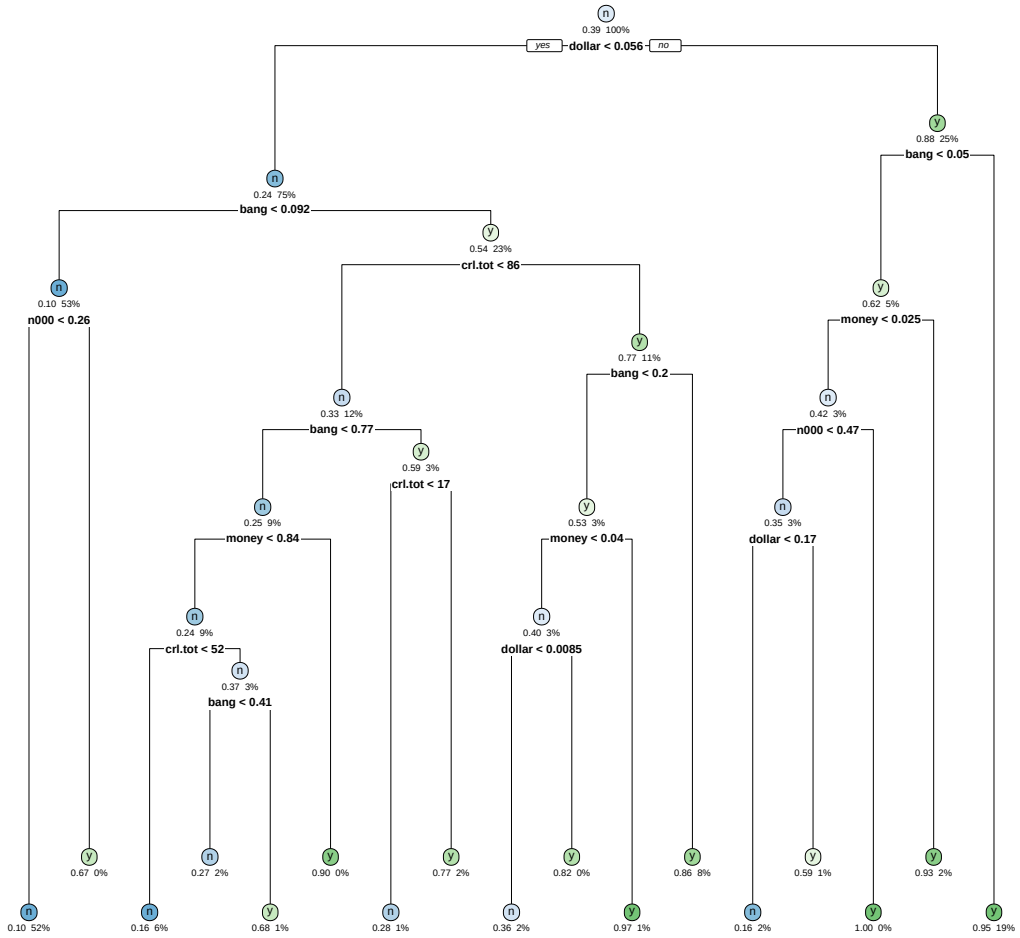
```
library(rpart.plot)
rpart.plot(spam.ct1, main="")
```



```
prp(spam.ct1, type=3, extra=4, faclen=0)
```



```
rpart.plot(spam.ct1, main="", extra=106, under=TRUE, faclen=0)
```



Exercice 2

Pour comprendre leurs performances, les valeurs de plusieurs caractéristiques de 209 processeurs d'ordinateurs ont été relevées. Les caractéristiques sont la période de l'horloge en nanosecondes (**syct**), la mémoire vive minimale en kilo-octets (**mmin**), la mémoire vive maximale en kilo-octets (**mmax**), la mémoire cache en kilo-octets (**cach**), le nombre minimal de canaux (**chmin**) et le nombre maximal de canaux (**chmax**).

On se propose de construire un arbre de régression à l'aide de la librairie **rpart** implémentée par Therneau et Atkinson en prenant la performance relative à un processeur particulier comme variable réponse et les caractéristiques des processeurs comme variables explicatives. Pour des raisons pratiques, la performance relative est donnée sous sa forme logarithmique en base 10.

a) Les données se trouvent dans l'objet **cpus** de la librairie **MASS** qui doit être installée puis chargée dans votre session de **R**.

Pour obtenir les données ainsi que des informations sur elles, utiliser les commandes

```
data(cpus)
?cpus
```

b) Pour construire l'arbre de régression à l'aide de la librairie **rpart** et en obtenir un résumé succinct partiel (paramètre de complexité fixé à 0.001), utiliser les commandes

```
library(rpart)
set.seed(123)
cpus.rt<-rpart(log10(perf)~., cpus[,2:8], cp=0.001)
print(cpus.rt, cp=0.001)
```

c) Afficher un résumé plus détaillé de l'arbre de régression.

d) Représenter graphiquement l'arbre de régression complet.

e) Il nous reste à élaguer l'arbre de régression et le rendre optimal par la règle du "un écart-type" en partant d'un paramètre de complexité **CP** fixé à 0.001. À l'aide de **R**,

1. déterminer le nombre nécessaire de divisions (**nsplit**) pour obtenir un arbre optimal ;
2. en déduire la taille (nombre total de feuilles) de cet arbre ;
3. visualiser l'application de la règle du "un écart-type" par la fonction **plotcp** ;
4. tracer l'arbre de régression final.

Bout de code qui peut être utile :

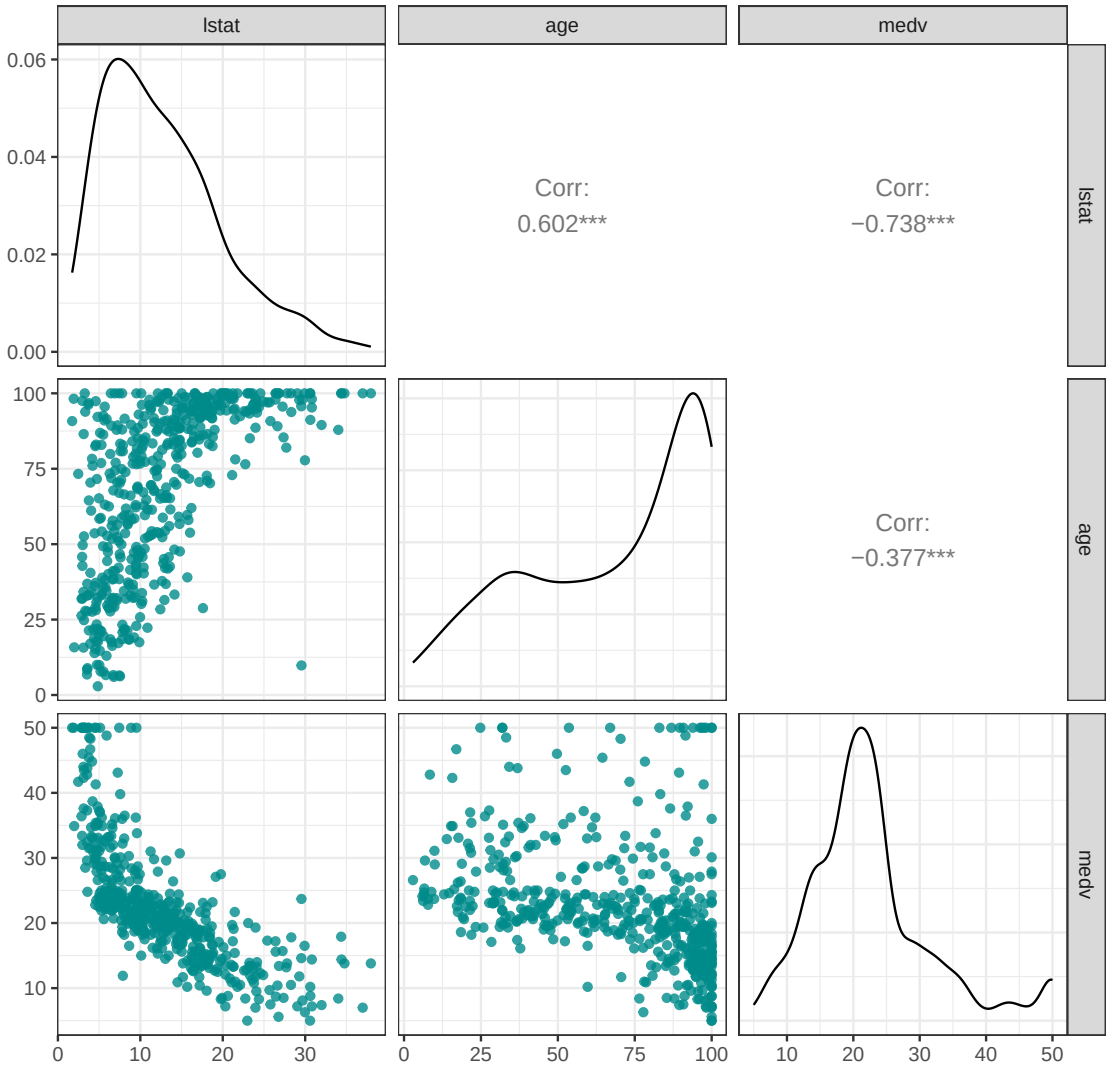
```
cp<-cpus.rt$cptable
opt<-which.min(cpus.rt$cptable[,"xerror"])
r<-cp[, 4][opt] + cp[, 5][opt]
rmin<-min(seq(1:dim(cp)[1])[cp[, 4] < r])
cp0<-cp[rmin,1]
cp0
cat("size chosen was", cp[rmin,2]+1, "\n")
```

Exercice 3

Les caractéristiques de 506 logements situés dans les banlieues de Boston ont été relevées. Parmi elles figurent le pourcentage de ménages à faible status socio-éconmique qui résident dans la banlieue considérée (**lstat**), la proportion de logements construits avant 1940 (**age**) ainsi que la médiane de la valeur des logements en milliers de dollars (**medv**). Pour les deux dernières variables, les logements sont occupés par leur propriétaire.

On se propose d'étudier la relation qui peut exister entre la médiane de la valeur des logements (variable de réponse) et l'âge du logement et la proportion de ménages à bas statut socio-économique qui vivent dans les banlieues de Boston (variables explicatives). Les données se trouvent dans l'objet **Boston** la librairie **ISLR2** de **R**.

a) Tracer le graphique des corrélations et des nuages de points se trouvant ci-dessous à l'aide de la fonction **ggpairs()** de la librairie **GGally**.



b) Existe-t-il une relation entre les deux variables explicatives (**lstat** et **age**) ?

c) Qu'en est-il de la relation entre la variables de réponse (**medv**) et les deux variables explicatives considérées l'une après l'autre ?

d) Répondre aux mêmes questions en utilisant la librairie **rgl** de **R** et les commandes qui suivent.

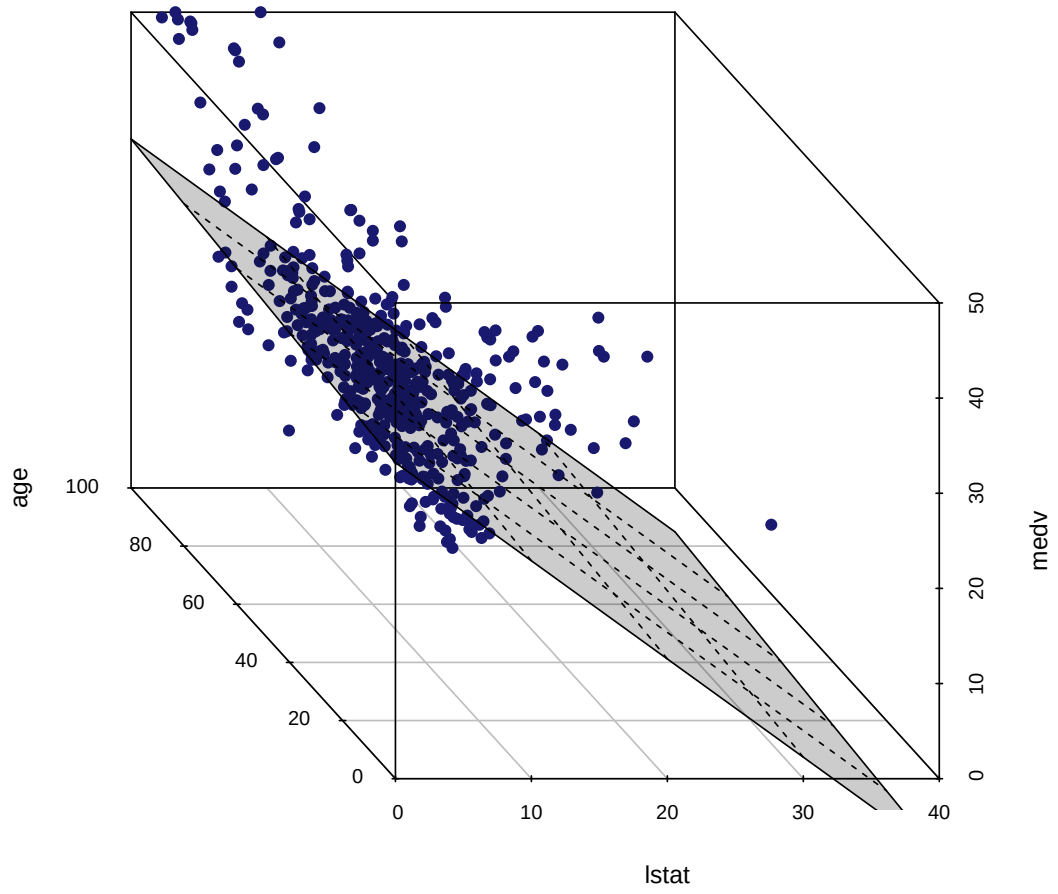
```
library(rgl)
plotids<-with(Boston, plot3d(lstat, age, medv, type="s", col="blue"))
rglwidget(elementId="plot3drgl")
```

e) Déterminer l'équation du modèle de régression linéaire multiple obtenue par la méthode des moindres carrés.

f) Dresser la table qui résume l'ajustement du modèle.

Tester séparément la significativité de chaque variable explicative par rapport au modèle complet. Quelles variables explicatives sont significatives à un seuil de confiance de 95 % ? Que peut-t-on conclure en considérant également les graphiques des corrélations et des nuages de points tracés ci-dessus ?

g) Tracer le graphique du nuage de points ci-dessous.



h) Déterminer le coefficient de détermination R^2 et le coefficient de détermination ajusté R^2_{adj} associés au modèle.

Que peut-on conclure ?

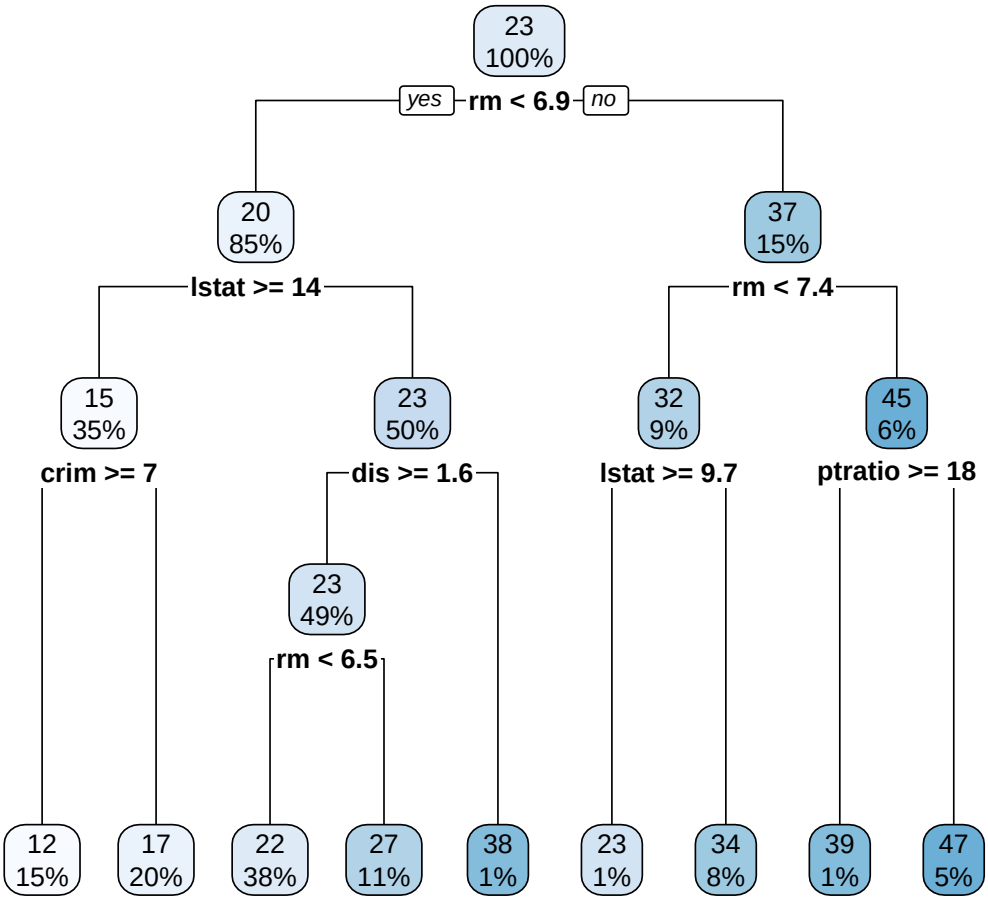
i) Effectuer une vérification des hypothèses inhérentes au modèle à l'aide des graphiques usuels.

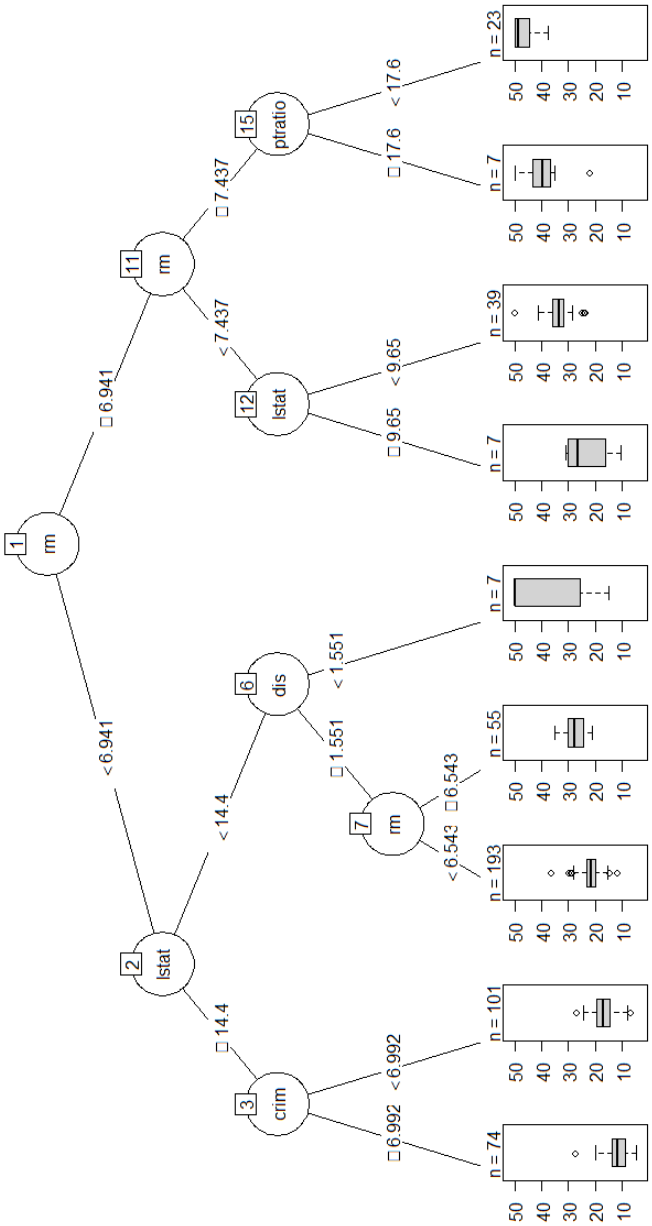
À votre avis, que peut-on faire pour améliorer la qualité du modèle ajusté ?

j) Ajuster le modèle de régression linéaire en considérant toutes les caractéristiques des logements des banlieues de Boston.

Que peut-on conclure ?

k) Construire à l'aide de la librairie **rpart** un arbre de régression optimal en utilisant toutes les variables explicatives et un coefficient de complexité de 0.001.





l) En utilisant l'arbre de régression élagué, quelles sont les variables explicatives les plus importantes ?

m) Quelles sont les caractéristiques des maisons les plus chères dans les banlieues de Boston ?

The R-Files

The truth is in the data