# データマイニング

## 第9回　ツリーモデル

2023年春学期

宮津和弘

# 本日の講義・演習

| 日付 | 講義・演習内容 |
|---|---|
| 04/14/23 | (1) イントロダクション |
| 04/21/23 | (2) ビジネスシミュレーション |
| 04/28/23 | (3) ID-POSデータ分析 |
| 05/12/23 | (4) 対応分析 |
| 05/19/23 | (5) クラスター分析 |
| 05/26/23 | (6) 自己組織化マップ |
| 06/02/23 | (7) 線形判別分析 |
| 06/09/23 | (8) 非線形判別分析 |
| **06/16/23** | **(9) ツリーモデル** |
| 06/23/23 | (10) 集団学習 |
| 06/30/23 | (11) サポートベクターマシン |
| 07/04/23 | (12) ネットワーク分析 |
| 07/14/23 | (13) 共分散構造分析 |
| 07/21/23 | (14) テキスト分析 |
| 07/28/23 | (15) まとめ |

# 機械学習の手法

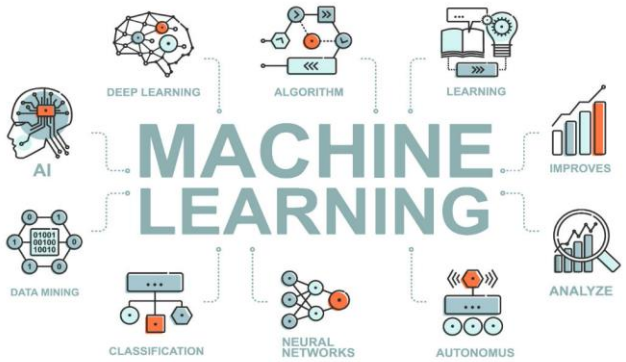
## 教師データあり

- ✓ 線形回帰
- ✓ ロジスティック回帰
- サポートベクターマシーン
- **分類木** ← 本日の演習はこれ！ **“決定木”**
- **回帰木**
- ランダムフォレスト
- 勾配ブースティング木
- ✓ ニューラルネットワーク
- 畳み込みニューラルネットワーク
- 再起型ニューラルネットワーク
- ✓ ナイーブベイズ
- k近傍法ブースティング
- バギング

## 教師データなし

- ✓ 階層型クラスタリング(ウォード法など)
- ✓ 非階層型クラスタリング（k-meansなど)
- トピックモデル（LDAなど)
- ✓ 自己組織化マップ
- ✓ アソシエーション分析（*）
- ✓ 協調フィルタリング（*）
- ✓ ベイジアンネットワーク（*）

＊ データサイエンス演習 1



# 本日の演習概要とポイント

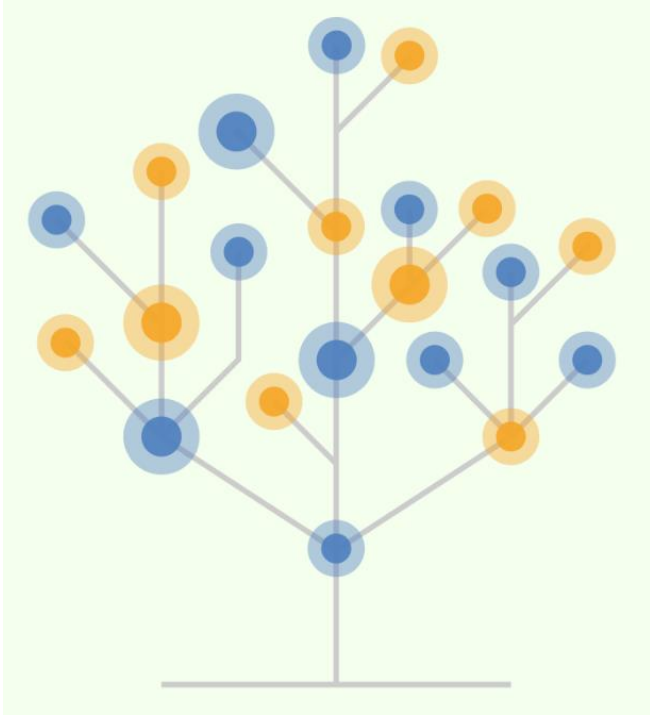
- **分類木と回帰木の考え方**

  → カイ二乗統計量、ジニ係数、エントロピー

- **決定木を用いたデータ演習**

  → 偽札データ、IRISデータ、車速度＆距離データ

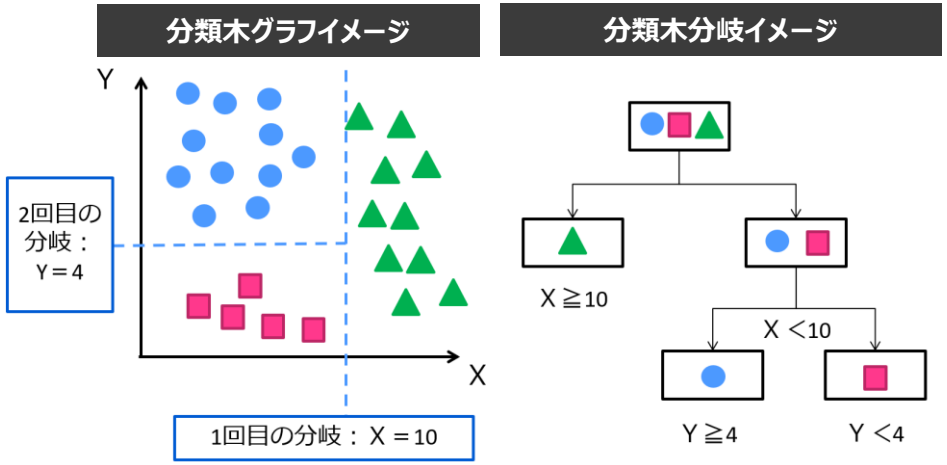
  ボストン住宅価格データ



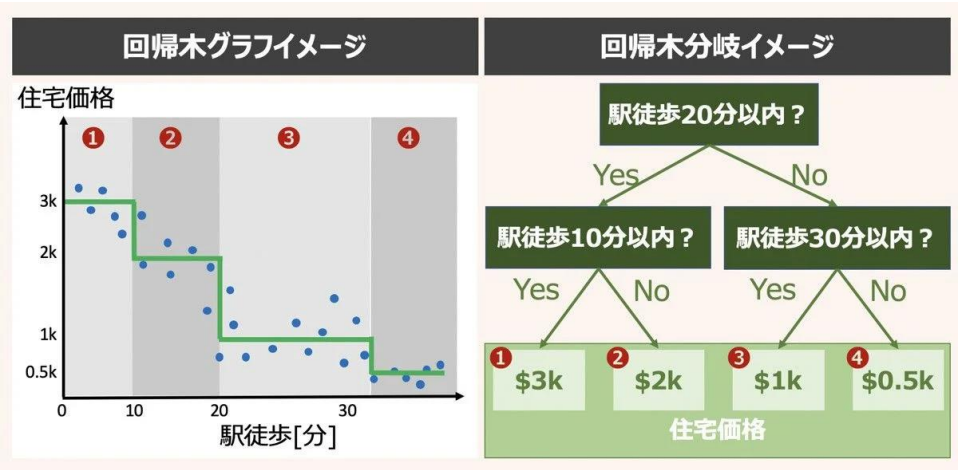
# 決定木（ディシジョンツリー）

# 決定木（ディシジョンツリー）分析とは？

ある目的変数に対して、説明変数を関連性の強い項目から分岐させて、**ツリー状**に表して分析する手法のこと。区分を分類するとき「**分類木**」、数値を予測するとき「**回帰木**」とよばれる。




https://toukei-lab.com/light-gbm




https://di-acc2.com/programming/python/23564/

# 分岐基準の考え方

決定木では、以下のような基準にもとづいて情報(データ)を分岐させる。

- **カイ二乗統計量**
  → 観測値と期待値の各分布の相違度合いを表す
- **ジニ係数**
  → もともとは社会における所得の不平等さを測る指標で、データの偏りを表す
- **エントロピー**
  → 情報理論におけるエントロピーでは、ある事象の起こりにくさを表す

**決定木**では、バラつきや偏りが生じていないようにサンプルを振り分ける

# カイ二乗値

**期待度数**$(E_{ij})$に対して、実際の**観測度数**$(n_{ij})$が統計的有意に異なるかを検定する

⇒ 以下では、k × $m$ の二元配置における独立性を考える

自由度$(k-1)(m-1)$の**カイ二乗分布**に従う

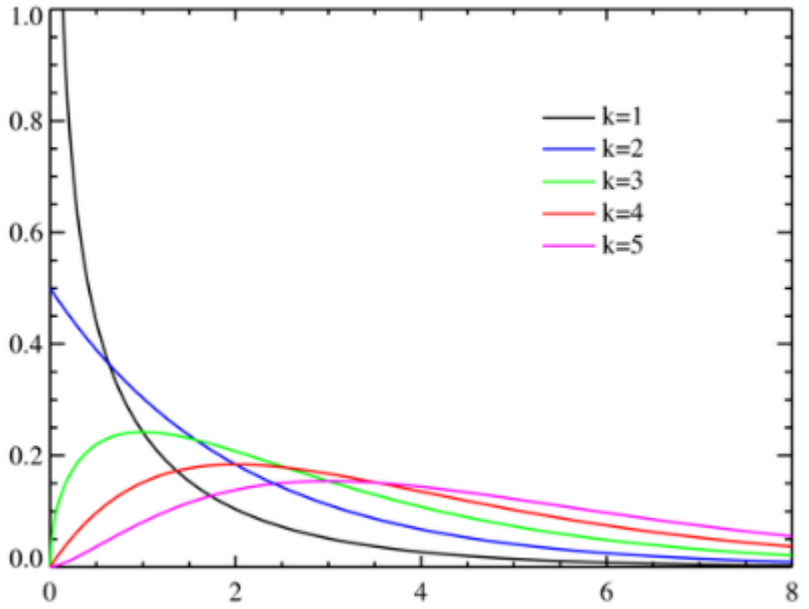
$$\sum_{i=1}^{k}\sum_{j=1}^{m}\frac{(n_{ij}-E_{ij})^2}{E_{ij}} \sim \chi^2(k-1)(m-1)$$

カイ二乗分布の確率密度関数は $x \geq 0$ に対し

$$f(x;k) = \frac{1}{2^{k/2}\Gamma(k/2)} x^{k/2-1} e^{-x/2}$$

また $x \leq 0$ に対し $f_k(x) = 0$ という形をとる。ここで $\Gamma$ はガンマ関数である。

PDF of chi-square distribution



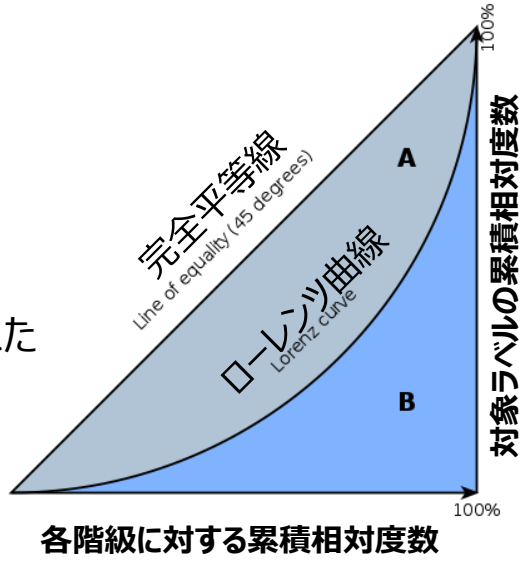

https://ja.wikipedia.org/wiki

# ジニ係数

**ジニ係数**とは右グラフにおいて、完全平等線とローレンツ曲線で囲まれた面積と三角形全体の面積(1/2)の比で表す。０のときは完全平等で、１のときは完全不平等を示す。

$$GI = 1 - \sum_{i=1}^{c} |p(i|t)|^2$$

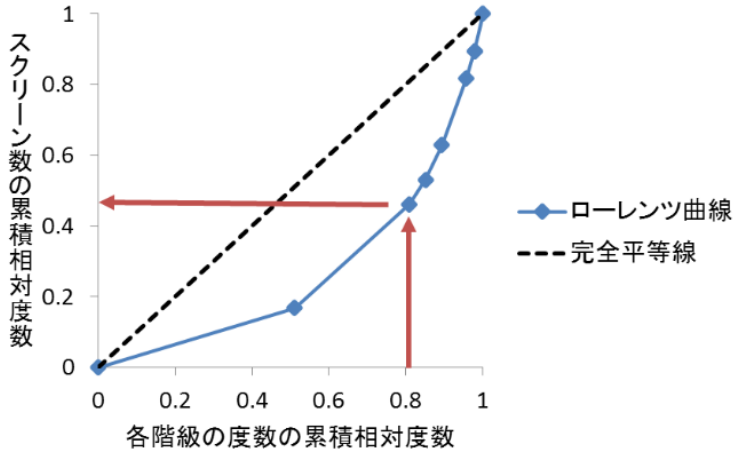
tはノード、iはクラス、pは分割された個体がクラスに属する比率

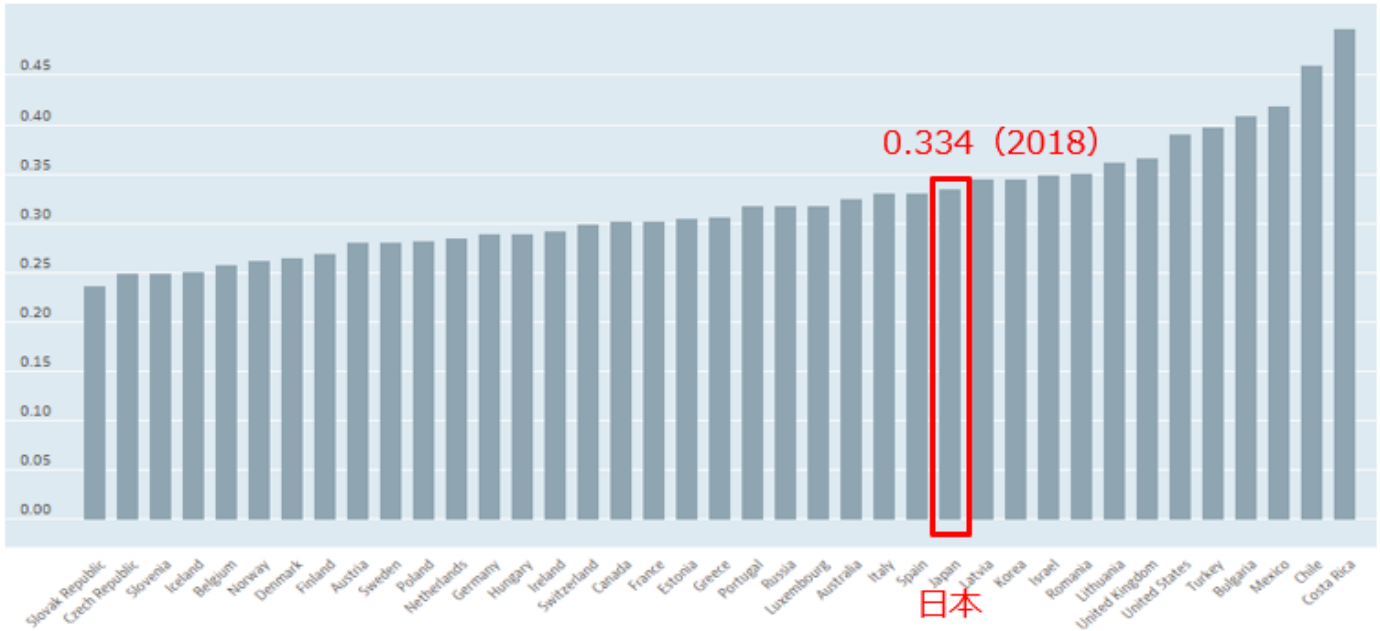


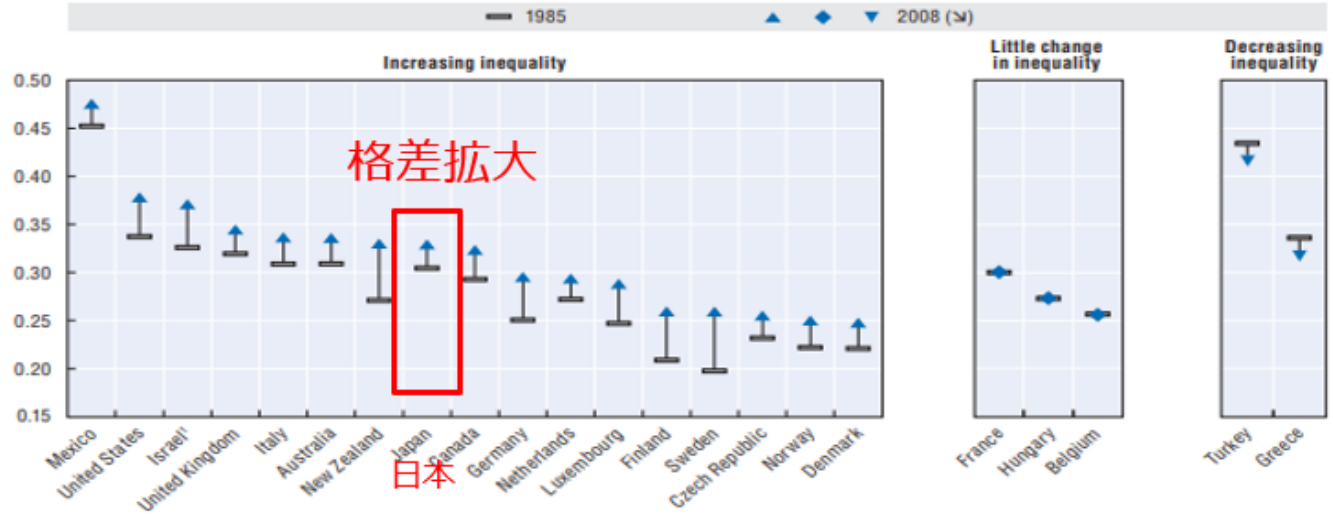

ローレンツ曲線の例

都道府県における映画館のスクリーン数を少ない順に並べて累積度数を表す。80%の都道県併せてスクリーン数は全体の40%強、残り20%で全体の60%弱を占めている。

→スクリーン数には格差がある！






（OECD）https://data.oecd.org/inequality/income-inequality.htm

# ジニ係数による経済格差

各国における経済格差を表す指標：
0 (完全平等)－1 (完全不平等)

- 旧共産圏国で小さい （～0.2）
- 格差で社会が不安定 （～0.4）
- 社会的な不満が激増 （～0.5）

1990年代に「一億総中流社会」と言われた日本は、0.249(1993)は、0.379(2011)と大きく上昇している

# エントロピー

情報量は生起する確率をpとして
以下のように定義する

$$log_2 \frac{1}{p} = -log_2\, p$$

情報源が生起する確率をそれぞれ
p(1),p(2),・・・,p(c)として、
**エントロピー**は平均情報量として
以下のように表す

$$entropy = -\sum_{i=1}^{c} p(i|t) log_2\, p(i|t)$$

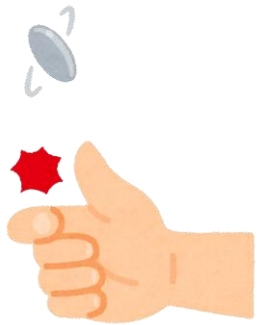
情報減が２つの例

例えば、理想的なコイン投げを考えるとき、
この場合のエントロピーは１となる

$$-\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2} = 1$$

ここで、表が1/4、裏が3/4で生起するコイン
の場合、エントロピーは0.811と減少する

$$-\frac{1}{2}\log\frac{1}{4} - \frac{1}{2}\log\frac{3}{4} = 0.811$$



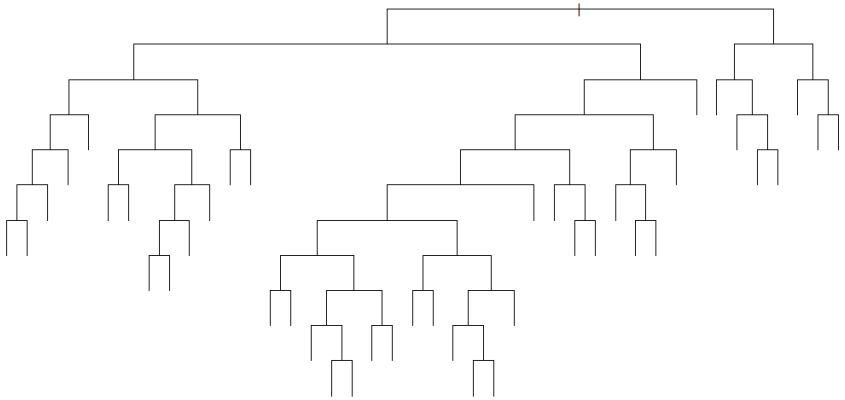
つまり、裏の塊（ラベル）が出やすくなると
**エントロピーは減少**する

→ エントロピー増大の原理とは正反対！

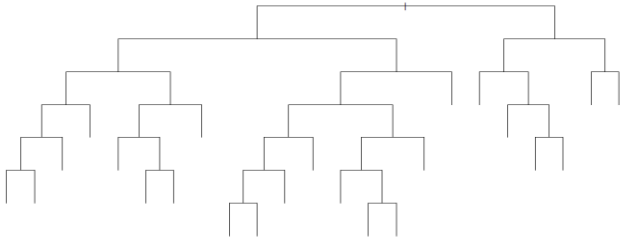
# 分類回帰樹木法(CART)とは？

**予測・分類** を目的とした教師あり機械学習の一つで、段階的にデータを不均一性が増加するように樹木の枝葉の様に分岐させる。CARTでは全データを用いて樹木を成長させてから剪定し、最適な樹木を構築する。

## ① 樹木の成長



## ② 樹木の刈込み



## ③ 最適な樹木の決定

※ 例えば、全データを10分割して、9つのデータセットで樹木を成長させ、残りの１つのデータセットを用いて検証しながら樹木を剪定する。

# 偽札データの読込み

① Rstudio起動する

② > library(Rcmdr)　※コマンドラインから Rコマンダー を起動する

③ 演習ファイル “sbnote.csv” を読み込む

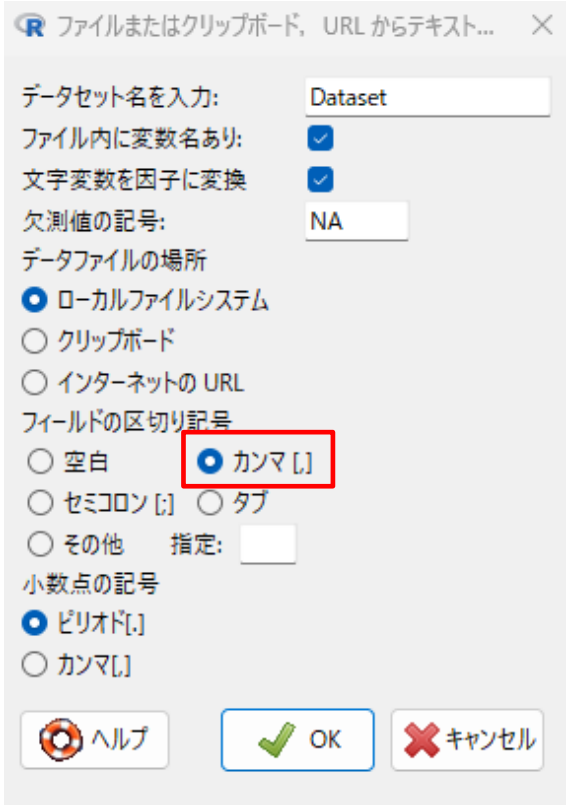
- Rstudio　> Dataset<-read.csv(“sbnote.csv”)

  又は

- Rコマンダー　(データ) → (データインポート) → (テキストファイルまたはクリップボード・・・) →

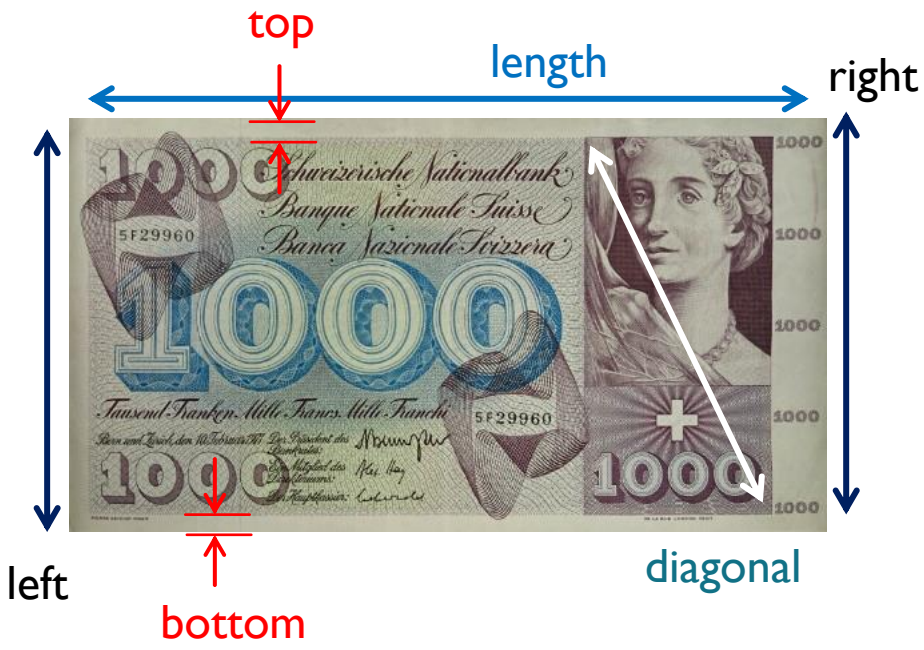
  ✔OKを選択して、sbnote.csv を指定する

④ 演習データが Dataset に読込まれる



# スイス銀行偽札データ

**1000フラン紙幣の尺度指標**



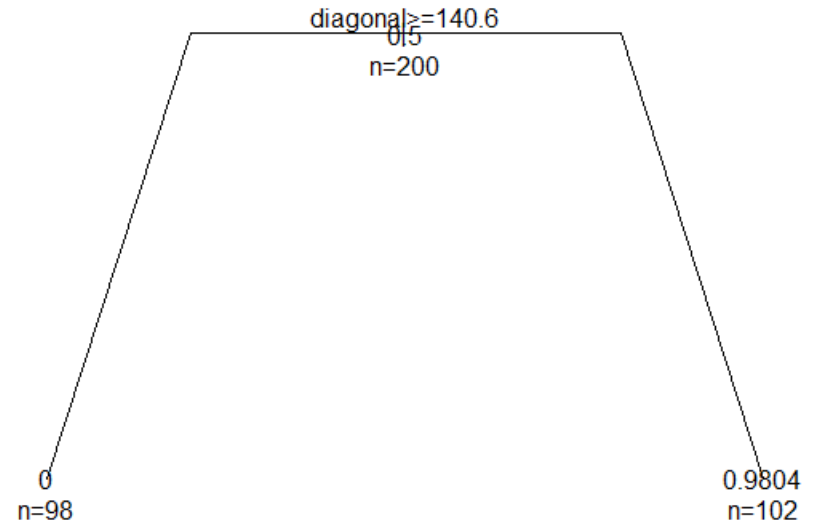

| length | left | right | bottom | top | diagonal | | class |
|---|---|---|---|---|---|---|---|
| 214.8 | 131 | 131.1 | 9 | 9.7 | 141 | | 0 |
| 214.6 | 129.7 | 129.7 | 8.1 | 9.5 | 141.7 | | 0 |
| 214.8 | 129.7 | 129.7 | 8.7 | 9.6 | 142.2 | | 0 |
| 214.8 | 129.7 | 129.6 | 7.5 | 10.4 | 142 | | 0 |
| 215 | 129.6 | 129.7 | 10.4 | 7.7 | 141.8 | | 0 |
| 215.7 | 130.8 | 130.5 | 9 | 10.1 | 141.4 | | 0 |
| 215.5 | 129.5 | 129.7 | 7.9 | 9.6 | 141.6 | | 0 |
| 214.5 | 129.6 | 129.2 | 7.2 | 10.7 | 141.7 | | 0 |
| 214.9 | 129.4 | 129.7 | 8.2 | 11 | 141.9 | | 0 |
| 215.2 | 130.4 | 130.3 | 9.2 | 10 | 140.7 | | 0 |
| 215.3 | 130.4 | 130.3 | 7.9 | 11.7 | 141.8 | | 0 |
| 215.1 | 129.5 | 129.6 | 7.7 | 10.5 | 142.2 | | 0 |
| 215.2 | 130.8 | 129.6 | 7.9 | 10.8 | 141.4 | | 0 |
| 214.7 | 129.7 | 129.7 | 7.7 | 10.9 | 141.7 | | 0 |
| 215.1 | 129.9 | 129.7 | 7.7 | 10.8 | 141.8 | | 0 |
| 214.5 | 129.8 | 129.8 | 9.3 | 8.5 | 141.6 | | 0 |
| 214.6 | 129.9 | 130.1 | 8.2 | 9.8 | 141.7 | | 0 |
| 215 | 129.9 | 129.7 | 9 | 9 | 141.9 | | 0 |
| 215.2 | 129.6 | 129.6 | 7.4 | 11.5 | 141.5 | | 0 |
| 214.7 | 130.2 | 129.9 | 8.6 | 10 | 141.9 | | 0 |

真偽判定：1（**偽札**）、0（**真札**）

# 偽札データの決定木

1000フラン紙幣の真贋は diagonal を測れば、ほとんど見分けられる！

```
> library(rpart)
> note.rp<-rpart(class~.,data=Dataset)
> plot(note.rp,uniform=T,branch=0.6,margin=0.05)
> text(note.rp,use.n=T,all=T)
> print(note.rp)
n= 200

node), split, n, deviance, yval
      * denotes terminal node

1) root 200 50.000000 0.5000000
  2) diagonal>=140.65 98  0.000000 0.0000000 *
  3) diagonal< 140.65 102  1.960784 0.9803922 *
```




真札　　偽札

# IRISデータの読込み

Rコマンダーから 【データ】 → 【パッケージ内のデータ】 → 【アタッチされたパッケージからデータセットを…】


データセット名に“**iris**”入力してOKすると、
irisデータが読み込まれるのを確認（↓）

行の追加　列の追加

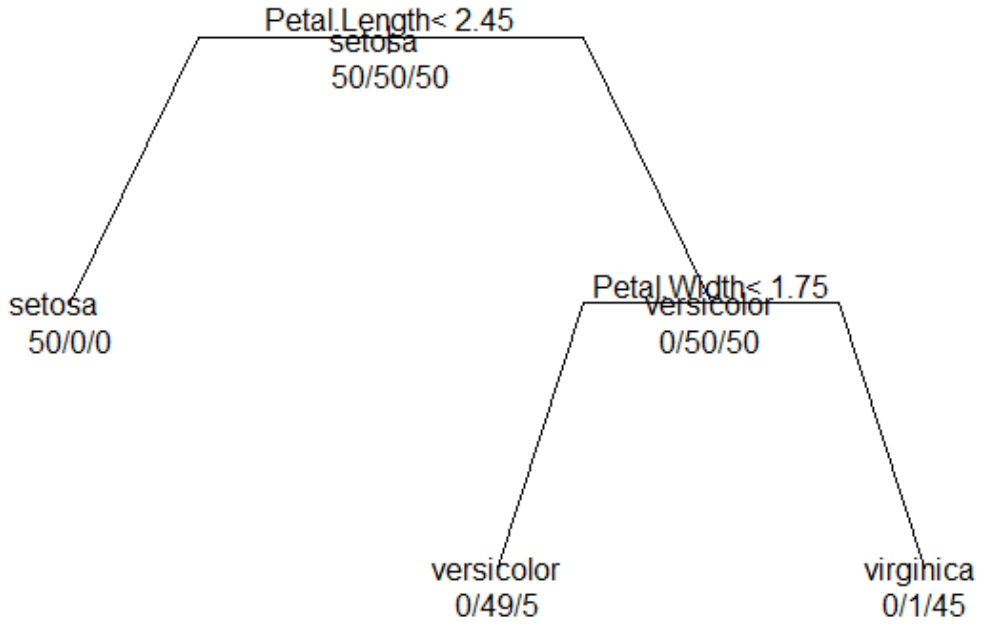
| | | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| | rowname | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
| 1 | 1 | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 2 | 2 | 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| 3 | 3 | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4 | 4 | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5 | 5 | 5.0 | 3.6 | 1.4 | 0.2 | setosa |
| 6 | 6 | 5.4 | 3.9 | 1.7 | 0.4 | setosa |
| 7 | 7 | 4.6 | 3.4 | 1.4 | 0.3 | setosa |
| 8 | 8 | 5.0 | 3.4 | 1.5 | 0.2 | setosa |
| 9 | 9 | 4.4 | 2.9 | 1.4 | 0.2 | setosa |
| 10 | 10 | 4.9 | 3.1 | 1.5 | 0.1 | setosa |
| 11 | 11 | 5.4 | 3.7 | 1.5 | 0.2 | setosa |
| 12 | 12 | 4.8 | 3.4 | 1.6 | 0.2 | setosa |

または、コマンドラインから直接入力

```
> data(iris)
```

# IRISデータの決定木モデル結果

アヤメの種類はPetal(花弁)の大きさだけで、
ほとんど分類できる




```
> iris.rp<-rpart(Species~.,data=iris)
> print(iris.rp)
n= 150

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 150 100 setosa (0.33333333 0.33333333 0.33333333)
  2) Petal.Length< 2.45 50   0 setosa (1.00000000 0.00000000 0.00000000) *
  3) Petal.Length>=2.45 100  50 versicolor (0.00000000 0.50000000 0.50000000)
    6) Petal.Width< 1.75 54   5 versicolor (0.00000000 0.90740741 0.09259259) *
    7) Petal.Width>=1.75 46   1 virginica (0.00000000 0.02173913 0.97826087) *
> plot(iris.rp,uniform=T,branch=0.6,margin=0.05)
> text(iris.rp)
```

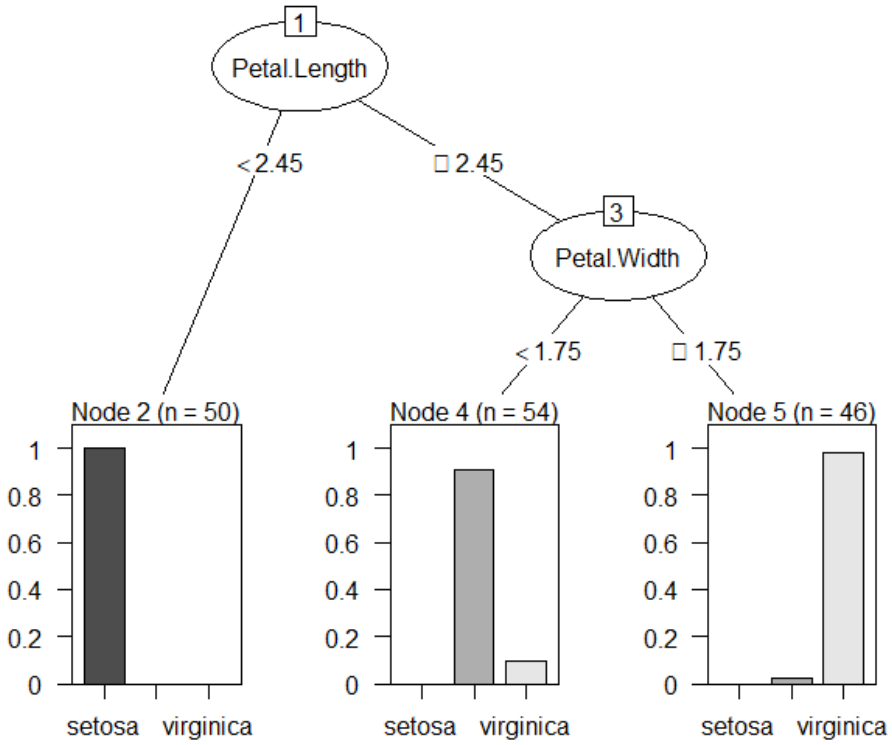
# IRISデータの決定木モデルの検証

```
> even.n<-2*(1:75)-1
> iris.train<-iris[even.n,]
> iris.test<-iris[-even.n,]
> iris.rp2<-rpart(Species~.,iris.train)
> iris.rp3<-predict(iris.rp2,iris.test[,-5],type="class")
> table(iris.test[,5],iris.rp3)
            iris.rp3
             setosa versicolor virginica
  setosa         25          0         0
  versicolor      0         24         1
  virginica       0          3        22
```

→ 検証データの誤り率は4%

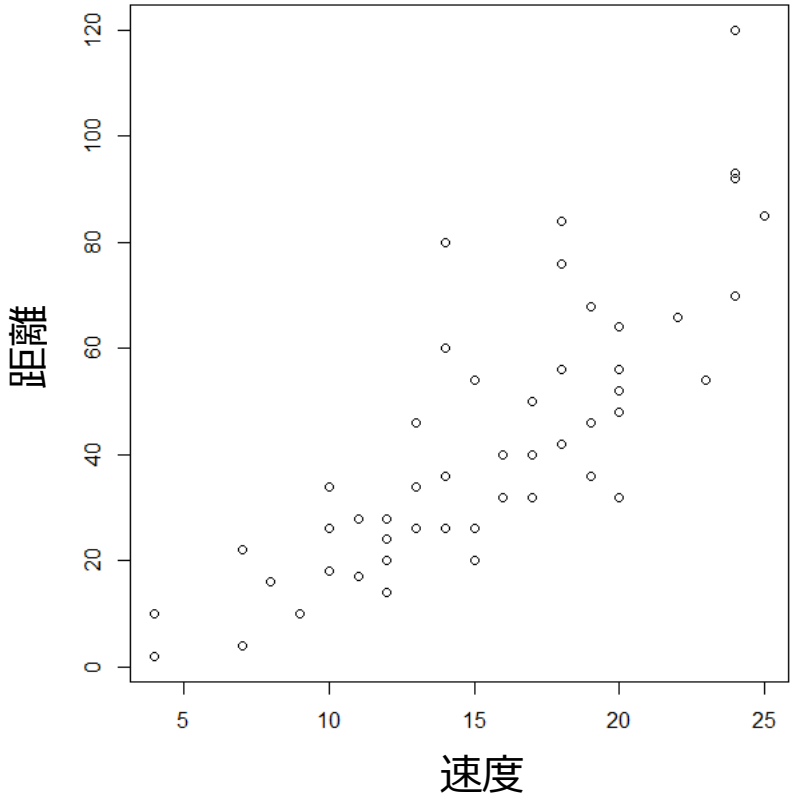
“partykit”をインストールして、
library(partykit)で立ち上げた後

```
> res<-as.party(iris.rp)
> plot(res)
```



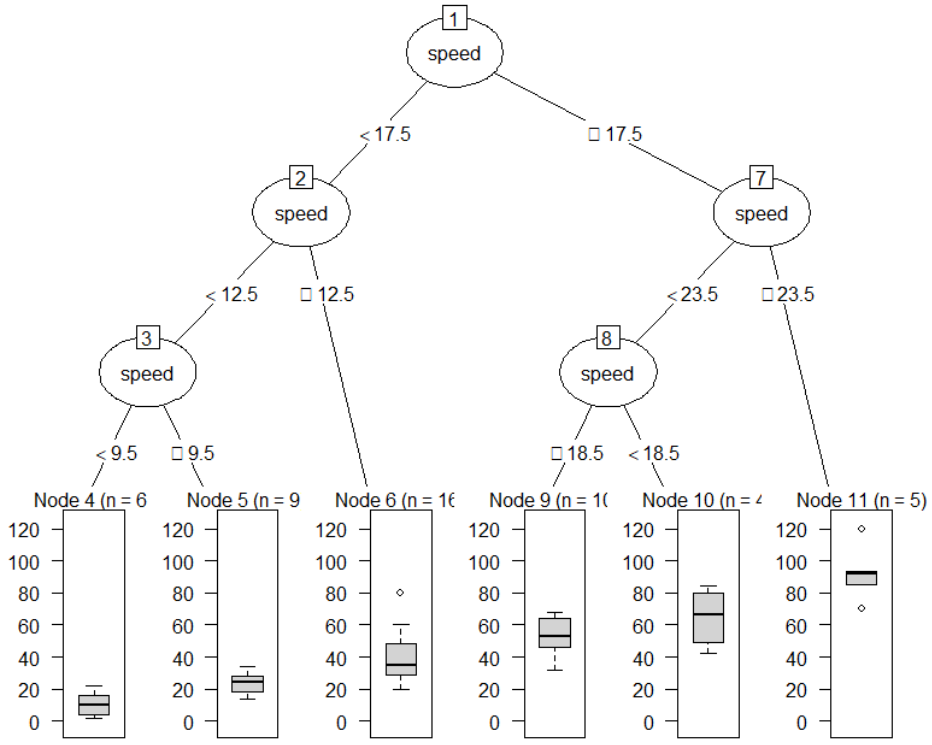
# CARSデータの読込み

```
> data("cars")
> set.seed(0)
> cars.rp<-rpart(dist~speed,data=cars,minsplit=3)
> printcp(cars.rp)

Regression tree:
rpart(formula = dist ~ speed, data = cars, minsplit = 3)

Variables actually used in tree construction:
[1] speed

Root node error: 32539/50 = 650.78

n= 50

        CP nsplit rel error  xerror     xstd
1 0.467640      0   1.00000 1.02866 0.214220
2 0.149077      1   0.53236 0.79834 0.153005
3 0.110494      2   0.38328 0.65919 0.144586
4 0.017441      3   0.27279 0.42660 0.092488
5 0.013284      4   0.25535 0.46957 0.086421
6 0.010000      5   0.24206 0.50496 0.092792
```

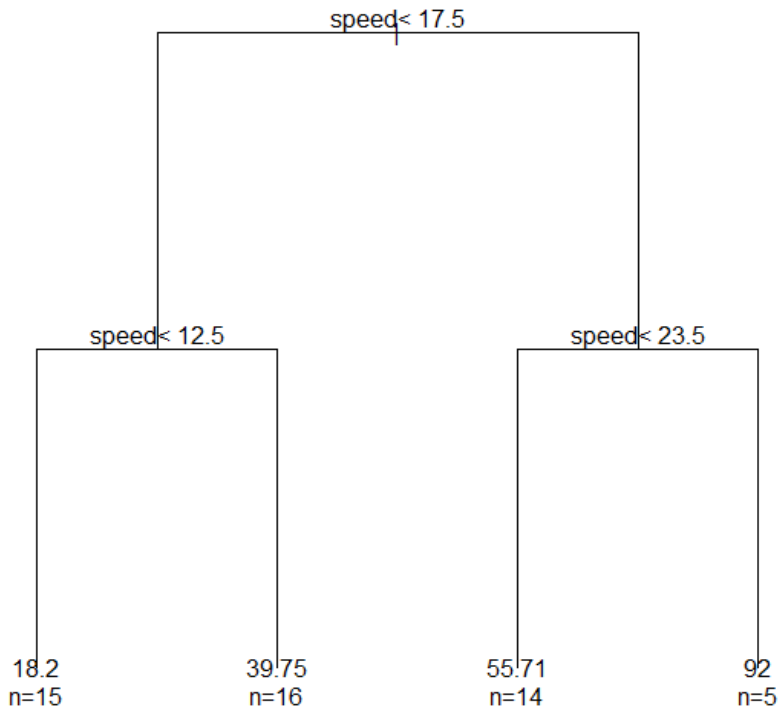


# CARSモデル検証の結果表示

```
> res<-as.party(cars.rp)
> plot(res)
```

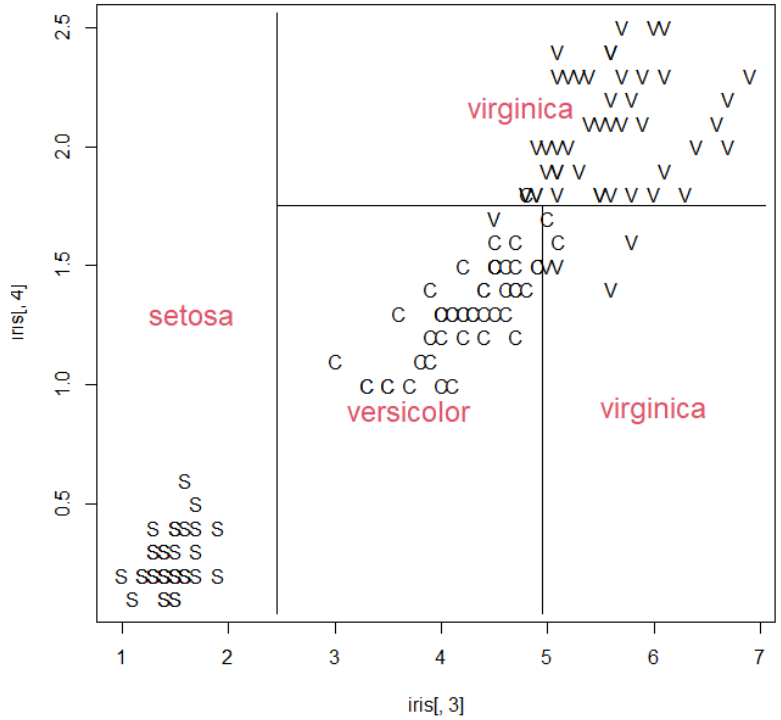



```
> cars.rp1<-prune(cars.rp,cp=0.044)
> plot(cars.rp1,uniform=T,margin=0.05)
> text(cars.rp1,use.n=T)
```

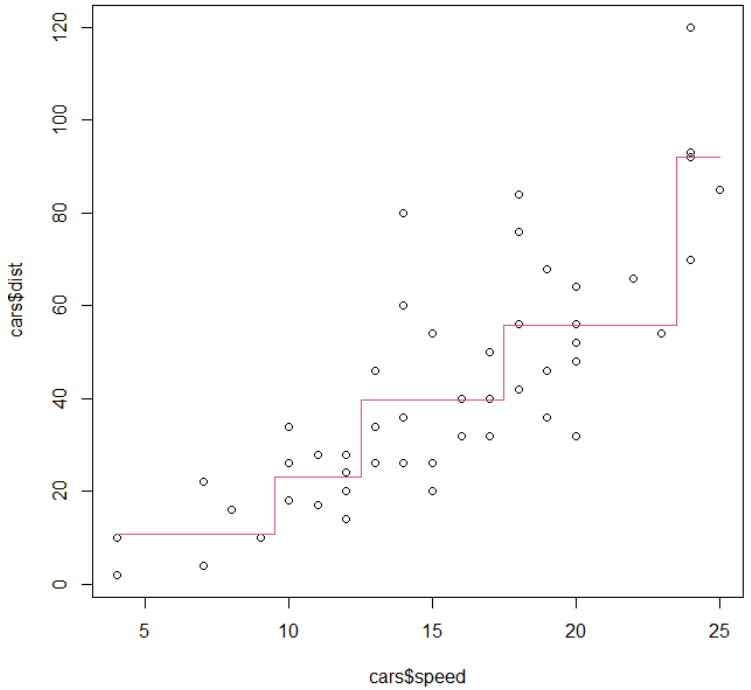


# 検証結果のパーティショニング

```
> iris.tr<-tree(Species~.,data=iris)
> iris.tr1<-snip.tree(iris.tr,nodes=c(12,7))
> iris.labels<-c("S","C","V")[iris[,5]]
> plot(iris[,3],iris[,4],type="n")
> text(iris[,3],iris[,4],labels=iris.labels)
> partition.tree(iris.tr1,add=T,col=2,cex=1.5)
```

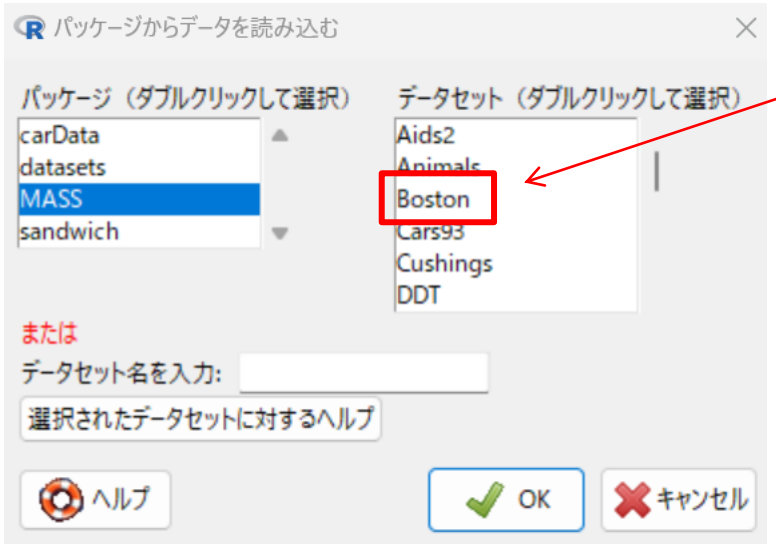



```
> cars.tr<-tree(dist~speed,data=cars)
> plot(cars$speed,cars$dist)
> partition.tree(cars.tr,add=T,col=2)
```



# BOSTON住宅価格データの読込み

Rコマンダーから 【データ】 → 【パッケージ内のデータ】 → 【アタッチされたパッケージからデータセットを…】




MASSパッケージ選択してBostonデータが
読み込まれるのを確認（↓）

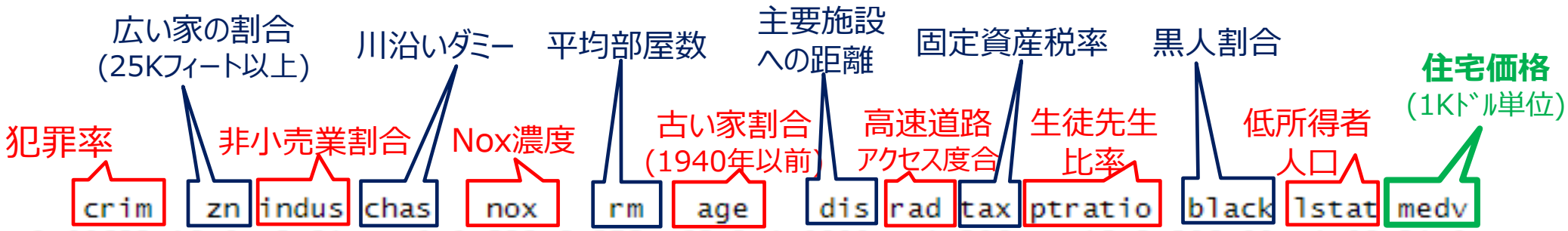
| crim | zn | indus | chas | nox | rm | age | dis | rad | tax | ptratio | black | lstat | medv |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.00632 | 18 | 2.31 | 0 | 0.538 | 6.575 | 65.2 | 4.09 | 1 | 296 | 15.3 | 396.9 | 4.98 | 24 |
| 0.02731 | 0 | 7.07 | 0 | 0.469 | 6.421 | 78.9 | 4.9671 | 2 | 242 | 17.8 | 396.9 | 9.14 | 21.6 |
| 0.02729 | 0 | 7.07 | 0 | 0.469 | 7.185 | 61.1 | 4.9671 | 2 | 242 | 17.8 | 392.83 | 4.03 | 34.7 |
| 0.03237 | 0 | 2.18 | 0 | 0.458 | 6.998 | 45.8 | 6.0622 | 3 | 222 | 18.7 | 394.63 | 2.94 | 33.4 |
| 0.06905 | 0 | 2.18 | 0 | 0.458 | 7.147 | 54.2 | 6.0622 | 3 | 222 | 18.7 | 396.9 | 5.33 | 36.2 |
| 0.02985 | 0 | 2.18 | 0 | 0.458 | 6.43 | 58.7 | 6.0622 | 3 | 222 | 18.7 | 394.12 | 5.21 | 28.7 |
| 0.08829 | 12.5 | 7.87 | 0 | 0.524 | 6.012 | 66.6 | 5.5605 | 5 | 311 | 15.2 | 395.6 | 12.43 | 22.9 |
| 0.14455 | 12.5 | 7.87 | 0 | 0.524 | 6.172 | 96.1 | 5.9505 | 5 | 311 | 15.2 | 396.9 | 19.15 | 27.1 |
| 0.21124 | 12.5 | 7.87 | 0 | 0.524 | 5.631 | 100 | 6.0821 | 5 | 311 | 15.2 | 386.63 | 29.93 | 16.5 |
| 0.17004 | 12.5 | 7.87 | 0 | 0.524 | 6.004 | 85.9 | 6.5921 | 5 | 311 | 15.2 | 386.71 | 17.1 | 18.9 |
| 0.22489 | 12.5 | 7.87 | 0 | 0.524 | 6.377 | 94.3 | 6.3467 | 5 | 311 | 15.2 | 392.52 | 20.45 | 15 |
| 0.11747 | 12.5 | 7.87 | 0 | 0.524 | 6.009 | 82.9 | 6.2267 | 5 | 311 | 15.2 | 396.9 | 13.27 | 18.9 |
| 0.09378 | 12.5 | 7.87 | 0 | 0.524 | 5.889 | 39 | 5.4509 | 5 | 311 | 15.2 | 390.5 | 15.71 | 21.7 |
| 0.62976 | 0 | 8.14 | 0 | 0.538 | 5.949 | 61.8 | 4.7075 | 4 | 307 | 21 | 396.9 | 8.26 | 20.4 |
| 0.63796 | 0 | 8.14 | 0 | 0.538 | 6.096 | 84.5 | 4.4619 | 4 | 307 | 21 | 380.02 | 10.26 | 18.2 |
| 0.62739 | 0 | 8.14 | 0 | 0.538 | 5.834 | 56.5 | 4.4986 | 4 | 307 | 21 | 395.62 | 8.47 | 19.9 |
| 1.05393 | 0 | 8.14 | 0 | 0.538 | 5.935 | 29.3 | 4.4986 | 4 | 307 | 21 | 386.85 | 6.58 | 23.1 |

または、コマンドラインから直接入力

> data(“Boston”)

※　まずは、**MASS**パッケージがインストールされていることを確認する！

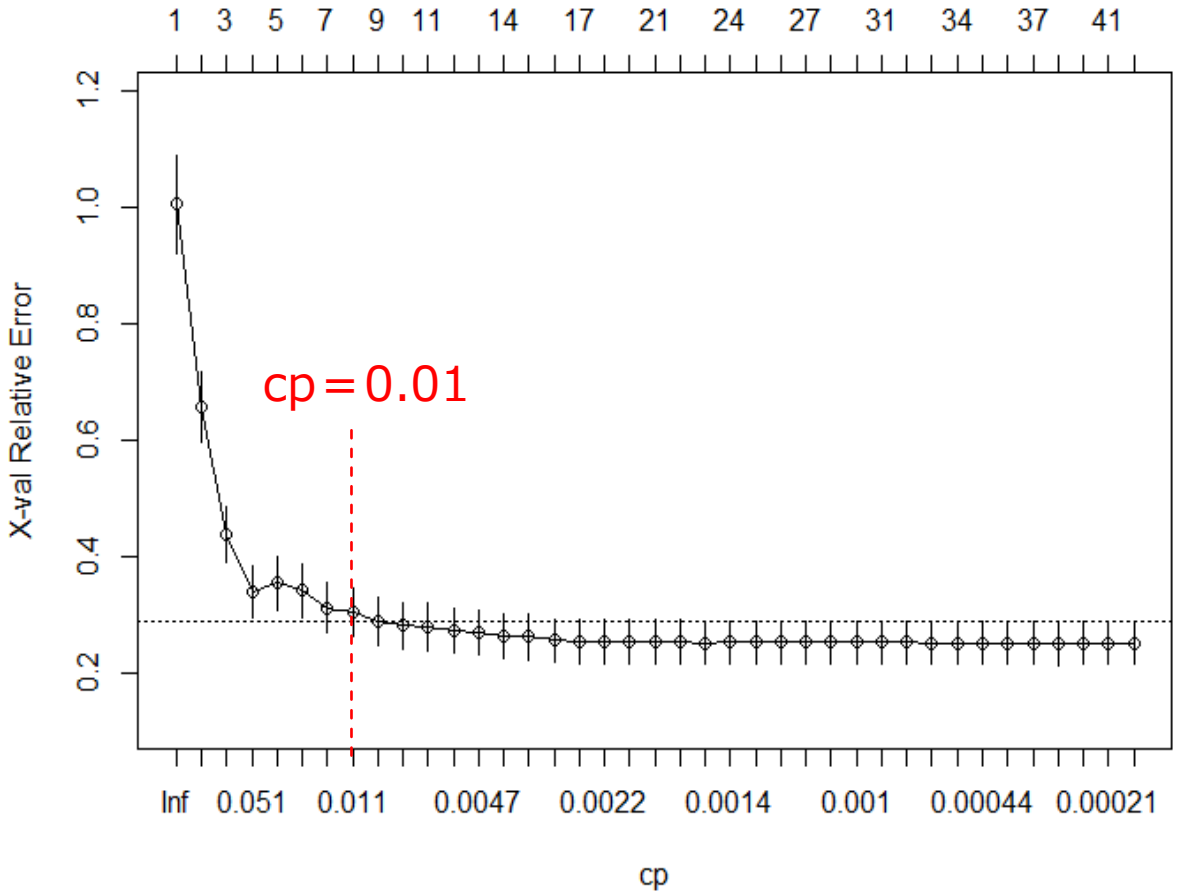
# BOSTON住宅価格データ




| crim | zn | indus | chas | nox | rm | age | dis | rad | tax | ptratio | black | lstat | medv |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.00632 | 18.0 | 2.31 | 0 | 0.538 | 6.575 | 65.2 | 4.0900 | 1 | 296 | 15.3 | 396.90 | 4.98 | 24.0 |
| 0.02731 | 0.0 | 7.07 | 0 | 0.469 | 6.421 | 78.9 | 4.9671 | 2 | 242 | 17.8 | 396.90 | 9.14 | 21.6 |
| 0.02729 | 0.0 | 7.07 | 0 | 0.469 | 7.185 | 61.1 | 4.9671 | 2 | 242 | 17.8 | 392.83 | 4.03 | 34.7 |
| 0.03237 | 0.0 | 2.18 | 0 | 0.458 | 6.998 | 45.8 | 6.0622 | 3 | 222 | 18.7 | 394.63 | 2.94 | 33.4 |
| 0.06905 | 0.0 | 2.18 | 0 | 0.458 | 7.147 | 54.2 | 6.0622 | 3 | 222 | 18.7 | 396.90 | 5.33 | 36.2 |
| 0.02985 | 0.0 | 2.18 | 0 | 0.458 | 6.430 | 58.7 | 6.0622 | 3 | 222 | 18.7 | 394.12 | 5.21 | 28.7 |
| 0.08829 | 12.5 | 7.87 | 0 | 0.524 | 6.012 | 66.6 | 5.5605 | 5 | 311 | 15.2 | 395.60 | 12.43 | 22.9 |
| 0.14455 | 12.5 | 7.87 | 0 | 0.524 | 6.172 | 96.1 | 5.9505 | 5 | 311 | 15.2 | 396.90 | 19.15 | 27.1 |
| 0.21124 | 12.5 | 7.87 | 0 | 0.524 | 5.631 | 100.0 | 6.0821 | 5 | 311 | 15.2 | 386.63 | 29.93 | 16.5 |
| 0.17004 | 12.5 | 7.87 | 0 | 0.524 | 6.004 | 85.9 | 6.5921 | 5 | 311 | 15.2 | 386.71 | 17.10 | 18.9 |

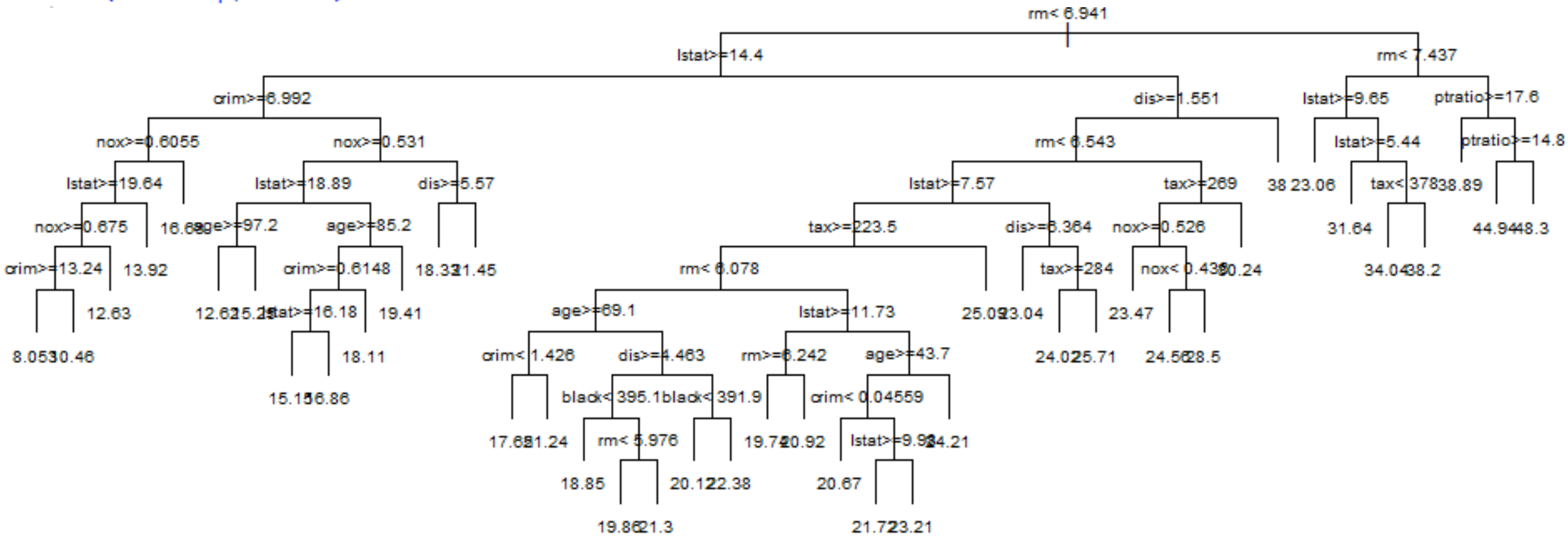
# BOSTON住宅価格モデルの結果（刈込み前）

```
> library(rpart)
> boston.rp<-rpart(medv~.,data=Boston,control=rpart.control(cp=0))
> boston.rp
n= 506

node), split, n, deviance, yval
      * denotes terminal node

   1) root 506 42716.30000 22.532810
     2) rm< 6.941 430 17317.32000 19.933720
       4) lstat>=14.4 175  3373.25100 14.956000
         8) crim>=6.99237 74  1085.90500 11.978380
          16) nox>=0.6055 62   552.28840 11.077420
            32) lstat>=19.645 44   271.79180  9.913636
              64) nox>=0.675 34   160.86260  9.114706
               128) crim>=13.2402 19    52.44737  8.052632 *
               129) crim< 13.2402 15    59.83600 10.460000 *
              65) nox< 0.675 10    15.44100 12.630000 *
            33) lstat< 19.645 18    75.23111 13.922220 *
          17) nox< 0.6055 12   223.26670 16.633330 *
                    ⋮
```



# BOSTON住宅価格樹木（刈込み前）

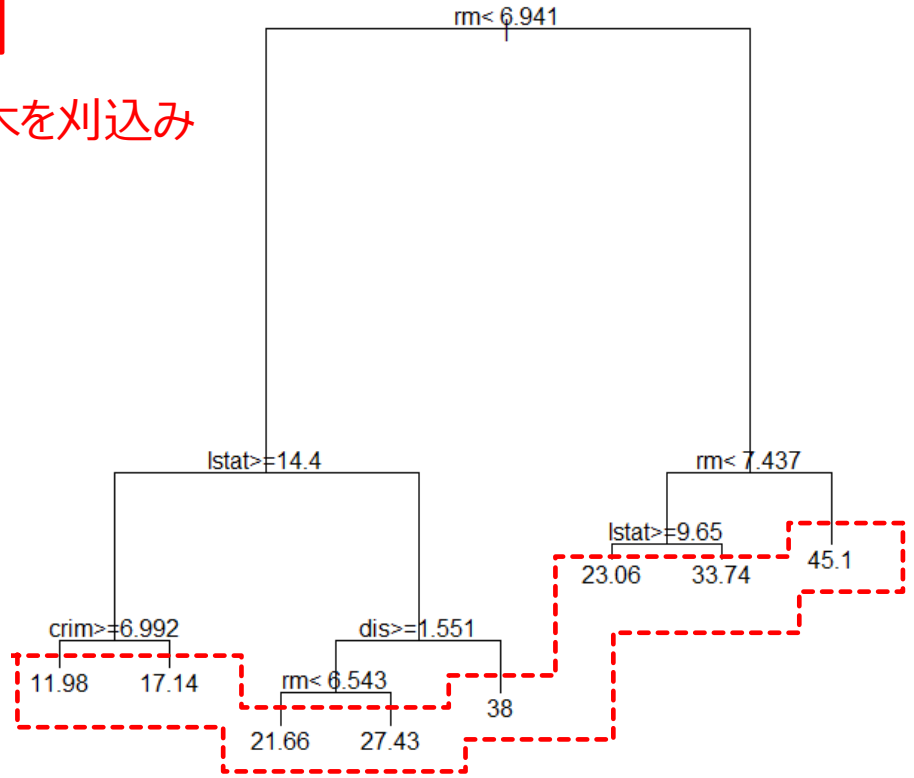
```
> plot(boston.rp,uniform=T,margin=0.1)
> text(boston.rp,cex=0.6)
```



# BOSTON住宅価格モデルの結果（刈込み後）

```
> boston.rp<-rpart(medv~.,data=Boston,control=rpart.control(cp=0.01))
> plot(boston.rp)
> text(boston.rp)
> print(boston.rp)
n= 506

node), split, n, deviance, yval
      * denotes terminal node

 1) root 506 42716.3000 22.53281
   2) rm< 6.941 430 17317.3200 19.93372
     4) lstat>=14.4 175  3373.2510 14.95600
       8) crim>=6.99237 74  1085.9050 11.97838 *
       9) crim< 6.99237 101  1150.5370 17.13762 *
     5) lstat< 14.4 255  6632.2170 23.34980
      10) dis>=1.5511 248  3658.3930 22.93629
        20) rm< 6.543 193  1589.8140 21.65648 *
        21) rm>=6.543 55  643.1691 27.42727 *
      11) dis< 1.5511 7  1429.0200 38.00000 *
   3) rm>=6.941 76  6059.4190 37.23816
     6) rm< 7.437 46  1899.6120 32.11304
      12) lstat>=9.65 7  432.9971 23.05714 *
      13) lstat< 9.65 39  789.5123 33.73846 *
     7) rm>=7.437 30  1098.8500 45.09667 *
```

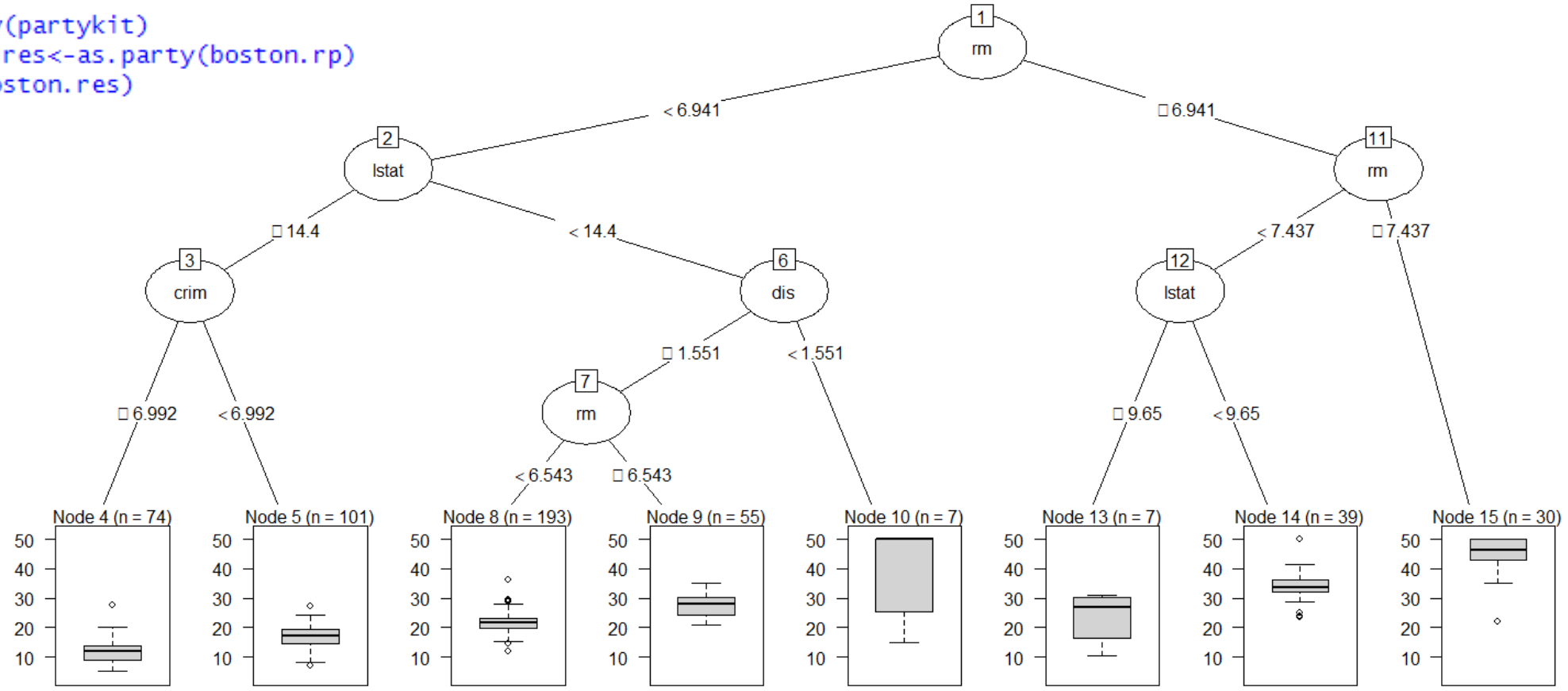
cp=0.01で樹木を刈込み




Boston地域の住宅価格帯を分類

# BOSTON住宅価格モデルの結果表示

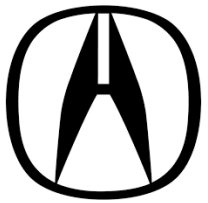
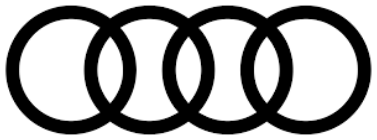
```
> library(partykit)
> boston.res<-as.party(boston.rp)
> plot(boston.res)
```



# 課題：自動車データの決定木分析　auto.csv

自動車に関するデータ（”auto.csv”）を用いて、自動車価格(Price)に対する決定木モデルを構築して、価格帯に対する自動車の特徴を議論しなさい。

| Manufacturer | Price | Cylinders | EngineSize | Horsepower | RPM | Rev.per.mile | Fuel.tank.capacity | Passengers | Length | Wheelbase | Width | Turn.circle | Weight |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Acura | 15.9 | 4 | 1.8 | 140 | 6300 | 2890 | 13.2 | 5 | 177 | 102 | 68 | 37 | 2705 |
| Acura | 33.9 | 6 | 3.2 | 200 | 5500 | 2335 | 18.0 | 5 | 195 | 115 | 71 | 38 | 3560 |
| Audi | 29.1 | 6 | 2.8 | 172 | 5500 | 2280 | 16.9 | 5 | 180 | 102 | 67 | 37 | 3375 |
| Audi | 37.7 | 6 | 2.8 | 172 | 5500 | 2535 | 21.1 | 6 | 193 | 106 | 70 | 37 | 3405 |
| BMW | 30.0 | 4 | 3.5 | 208 | 5700 | 2545 | 21.1 | 4 | 186 | 109 | 69 | 39 | 3640 |
| Buick | 15.7 | 4 | 2.2 | 110 | 5200 | 2565 | 16.4 | 6 | 189 | 105 | 69 | 41 | 2880 |
| Buick | 20.8 | 6 | 3.8 | 170 | 4800 | 1570 | 18.0 | 6 | 200 | 111 | 74 | 42 | 3470 |
| Buick | 23.7 | 6 | 5.7 | 180 | 4000 | 1320 | 23.0 | 6 | 216 | 116 | 78 | 45 | 4105 |
| Buick | 26.3 | 6 | 3.8 | 170 | 4800 | 1690 | 18.8 | 5 | 198 | 108 | 73 | 41 | 3495 |
| Cadillac | 34.7 | 8 | 4.9 | 200 | 4100 | 1510 | 18.0 | 6 | 206 | 114 | 73 | 43 | 3620 |
| Cadillac | 40.1 | 8 | 4.6 | 295 | 6000 | 1985 | 20.0 | 5 | 204 | 111 | 74 | 44 | 3935 |
| Chevrolet | 13.4 | 4 | 2.2 | 110 | 5200 | 2380 | 15.2 | 5 | 182 | 101 | 66 | 38 | 2490 |




データマイニングを楽しもう！