Baayen, R. H. (2008). *Analyzing Linguistic Data: A practical introduction to statistics using R.* Cambridge: Cambridge University Press.

金田　拓

5.2　Classification

cluster ではなく classification に注目。
cluster を色やフォントで分けたが、これは後付けによるもので、今度は分類に着目する。

5.2.1　Classification trees　　pp. 148 ~ 154

- dative alternation のデータを使う。与格位置が NP になるか PP になるか？
- 動詞 3263 回分について、15 の変数を扱って分類する。具体的にはどのような変数が格納されているか？

```
> colnames(dative)
> ?dative
```

✓ これらの変数から、与格が前置詞句になるか名詞句になるかを予測することができないか？
✓ CART 分析、Classification And Regression Trees（回帰ツリー）を用いる。
✓ 分類ツリーに限定して論じる（連続量を扱う場合、結果出力は回帰樹となる）

- Figure 5.14 の読み方：まず AccessOfRec が given の値を持っているかで分岐。正しければ左、違うなら右へ。分岐を通っていくと、葉 (leaf node)にたどり着く。例えば given, given, nonpronominal（左、右、左）で、与格が NP なのは 119 回中 17 回、というように解釈。

- 項目の重複をせずに分割するところから、CART 分析は再起分割(recursive partitioning)とも呼ばれる。それぞれの条件につき枝分かれしていき、停止基準を設けない限りあらゆる組み合わせが試され細分化される。そのようにしてできた細かい葉を pure であるという。が、trivially pure であるため細かすぎて一般化ができなくなる。
- 従って、デフォルトでは観測数が 20 になるとそれ以上細かくならず、止まるようになっている。また、極端な値が出ない条件は意味をなさないため、
- ノードの不純度(node impurity)を計算し、娘ノード(daughter node)での純度が上がるようにツリーを作る。

➢ rpart パッケージの rpart()関数を使って Figure 5.14 のような樹形図を描ける。

```
> library(rpart)
> dative.rp = rpart(RealizationOfRecipient ~ ., data = dative[ , -c(1, 3)])
```

✓ ドットは、従属変数以外の変数全てを示す

➢ 回帰樹をプロットする。

```
> plot(dative.rp, compress = T, branch =1, margin = 0.1)
> text(dative.rp, use.n = T, pretty = 0)
```

✓ use.n を T に設定すると、頻度が記入される。pretty を 0 にするのは、変数名を表示するため。

✓ まだ無駄が多いので、コスト複雑度枝刈り(cost-complexity pruning)を使って剪定を行う。

・ cost-complexity パラメーターである cp を計算する。値が大きいほど多くの枝をはらえて、根と切り株が残る。小さいと枝は切れない。

・ 正確さと複雑さ（葉の数で考える）を評価するために、*n* 重交差確認(*n*-fold cross-validation)を行う。交差確認法では、データセットの標本全体を *n* 等分に分割し、そのうちの 1 等分をテストデータ、それ以外を学習データとして、重複しない組み合わせで *n* 回、モデルの構築とテスト（確認・検証）を行い、その *n* 回のテスト結果の平均を全体の評価に用いる。これを *n* 重交差確認と呼ぶ。

・ この場合、連続値の cp と樹形図の大きさのため、無作為に 10 個に等分する。そのうち 1 つを除き、残りの 9 つを基にモデルを構築し、残る 1 つを予想するテストを行う。

・ 誤って分類する確率と、元の樹形図の誤って分類する確率を比較する。

・ 本当は、新しいデータにモデルがどの程度適用できるか、ということを検証するのだが、新しいデータ無しで行う確認法である。

```
> plotcp(dative.rp)
```

・ 横軸が、枝が切られる地点の cp 値を表す。切られるツリーのサイズは上の値。縦軸が cross-validation でのエラーの値を示す。

・ 各点から伸びる短い縦線が、平均より ±1 標準誤差の値で、点線が平均より 1 標準誤差上で、グラフ中最も低い点を示す。

・ cost-complexity pruning での一般的な枝の刈り方は、点線より下で最も左の点を剪定すること。この場合は 1 点しかないので一番左でもあり右でもあるわけだが。

・ 無難に cp = 0.41 を基準として prune ( )で刈って、6 つの葉を残すことにする。

```
> dative.rpl = prune(dative.rp, cp = 0.041)
> plot(dative.rpl, compress = T, branch = 1, margin = 0.1)
> text(dative.rpl, use.n = T, pretty = 0)
```

・ ちなみに出力される値は、 *t* 、*F*, χ2 乗分布に従う古典的統計の *p* 値的に有意である。

```
> dative.rpl
```

– それぞれ、1 行目はデータ数、2 行目は残ったものの説明で、行番号、分類の基準、ノード下の観測値、純度減、NP と PP の比率である。

➢ どれくらい正確に予測ができているだろうか。CART の予測値と観測値を比較する。predict( )を使い、dative.rpl に格納されているモデルで予想。

```
> head(predict(dative.rpl))
```

・ 入力した各行は確率に沿って振り分けられる。この場合は NP と PP の 2 通りがある。

➢ 基準となるのは大きい値の NP。0.5 より大きければ TRUE(NP),小さければ FALSE(PP)。

```
> choiceIsNP = predict(dative.rpl)[ ,1] >= 0.5
> choiceIsNP[1:6]
```

➢ 実際の値 NP, PP と、TRUE, FALSE が合致するか、得られたベクトルを合成する。

```
> preds = data.frame(obs = dative$RealizationOfRecipient, choiceIsNP)
> head(preds)
```

➢ クロス集計表にする

```
> xtabs ( ~ obs + choiceIsNP, data = preds)
```

✓ 269 + 177 = 446 件、13.7%が誤分類された。

➢ もともとの比率と比べてみる

```
> xtabs ( ~ RealizationOfRecipient, dative)
```

☆ 3263 件を全て NP に分類したときの 849/3263 の比率 26%より、このモデルを用いた方が精度は上がっている。

・ CART の回帰樹の強みとして、相互作用を非常に上手く扱えている点が挙げられる。
・ 例えば SemanticClass は右にのみ現れ、右側の要素と交互作用があることが見て取れる。右と右、左は左で相互作用している。
・ 回帰モデルでは複雑に見える相互作用も、回帰ツリーでは把握しやすい。