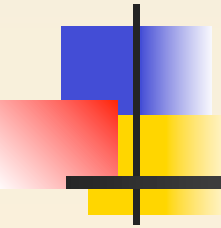
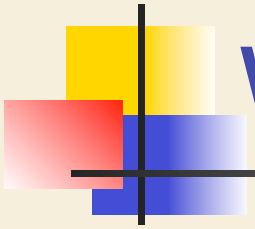


Classification and Regression

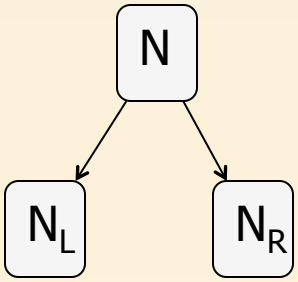


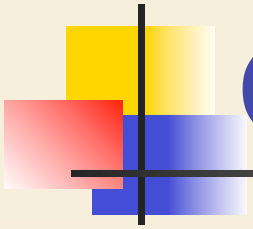
When?

- Classification tree models
  - Dependent variable is categorical variable
- Regression tree models
  - Dependent variable is continuous variable
- Independent variables can be either one: categorical and/or continuous
- Robust with respect to outliers

Classification Trees: Characteristics

- No assumption on the data distribution (i.e., non parametric approach)
- Goal: each terminal node is of one class
- Idea:
  - Split the initial data using one variable that best differentiate the data with respect to the dependent variable
  - Keep splitting recursively each node until…

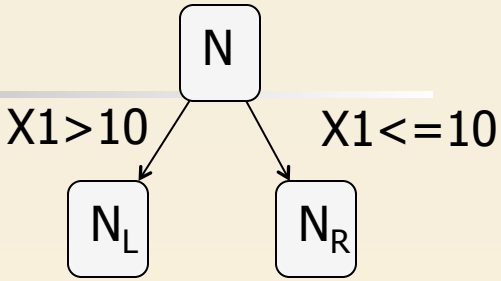




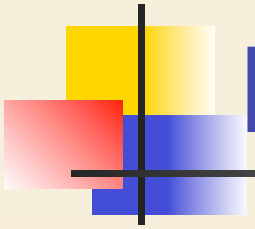
Common stop conditions

- A node is pure!
- The number of observations in the current node is < a certain minimum
- Maximum number of nodes is reached
- In the current node the number of observations of a same class is higher than a certain amount
- Minimum improvement

Split criteria



- Univariate vs multivariate
- Maximize the decrease of risk at each level:
  - $p(N_L)I(N_L) + p(N_R)I(N_R) \leq p(N)I(N)$
  - $\Delta I = p(N)I(N) - p(N_L)I(N_L) - p(N_R)I(N_R)$
- Impurity: $I(N) = \sum_{i=1}^{C} f(p_{iN})$, where
  - C is the number of classes
  - p_{iN} is the proportion of those of class i



Measures of impurity

- Expectations from an impurity function
  - Maximum: when a node contains equal number of observations of each class
  - Minimum: all observations belong to one class
- Examples
  - Information index: $f(p) = -\sum_i p_i \log(p_i)$
  - Gini index: $f(p) = 1 - \sum_j p_j^2$

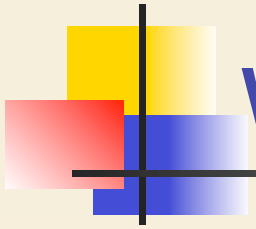
Extending impurity criteria by introducing a loss function

- Generalized Gini index:

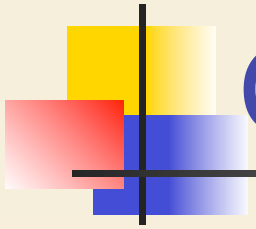
$$G(p) = \sum_i \sum_j L(i,j) p_i p_j$$

  - Where L(i,j) is the loss of assigning j to an object of class i, p_i is the proportion of objects belonging to i
- Problem: Costs are symmetric

Extending impurity criteria with altered priors

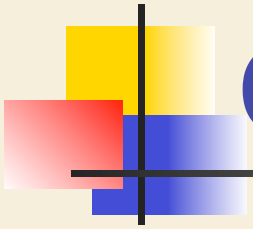


- Maximum impurity is reached when $p_i=1/C$, C is the number of classes
  - Ex: Colour of an apple [green, red, yellow]
- Altered priors allow to weight different misclassifications differently and decide on the worst case scenario
  - Ex: Patient risk for heart attack [not at risk, medium risk, high risk]



Growing a tree

- Two approaches
  - Build the maximum tree, i.e., split as much as possible. Prune the tree based on selected criteria, e.g., error
  - Impose more conservative splitting criteria and avoid computation
    - Possible problem: overprunning



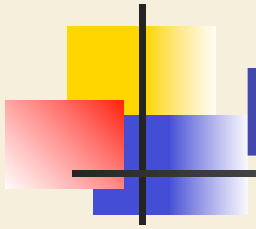
CART with R and rpart

- Implement most of the ideas of CART
- Load the library: library(rpart)
- Build a model:

  rpart(formula, method,…)

- Can specify parameters for the stop conditions

rpart: some of the default parameters

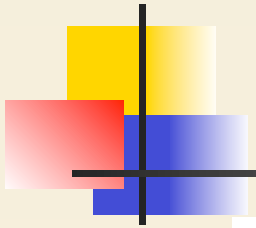


- Min objects (minsplit) = 20
- Min objects in terminal nodes (minbucket): round(minsplit/3)
- The minimal decrease of risk, i.e., complexity parameter (cp): 0.01
- Maximum tree depth (maxdepth): 30

Example: Stage C prostate cancer

- Load and have a look on the data

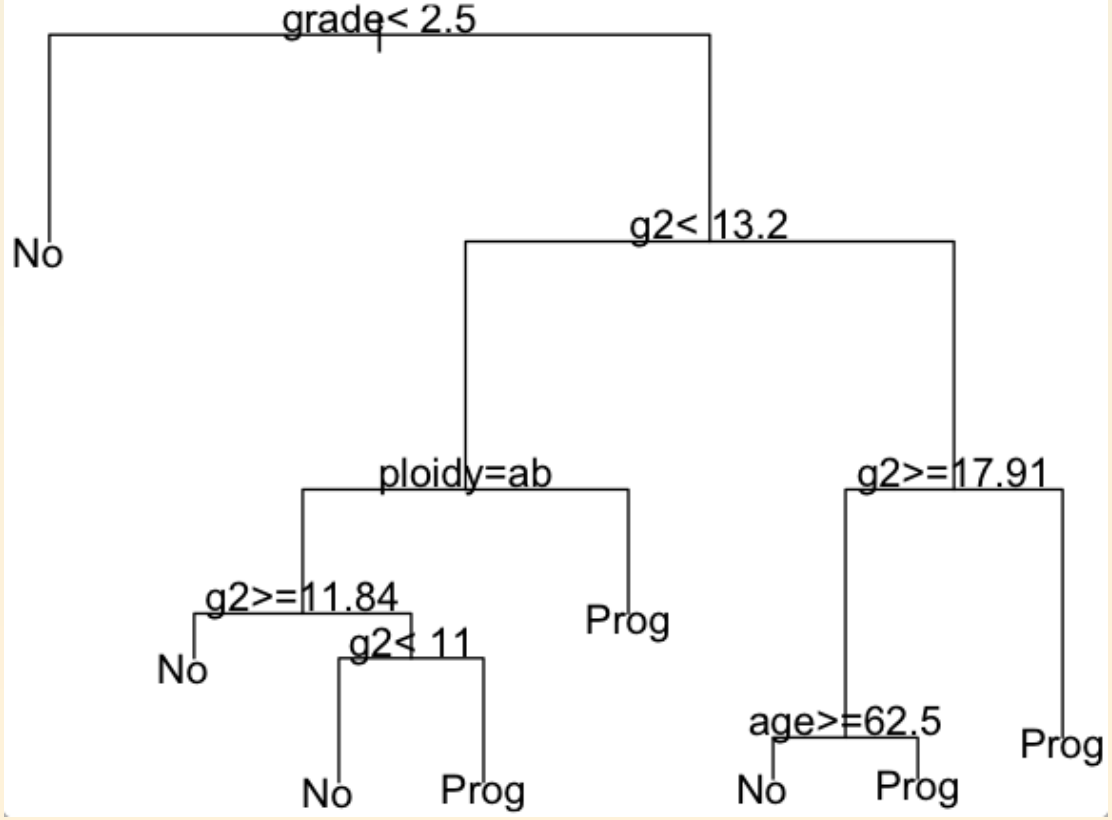
```
> library(rpart)
> data(stagec)
> str(stagec)
'data.frame':   146 obs. of  8 variables:
 $ pgtime : num  6.1 9.4 5.2 3.2 1.9 4.8 5.8 7.3 3.7 15.9 ...
 $ pgstat : int  0 0 1 1 1 0 0 0 1 0 ...
 $ age    : int  64 62 59 62 64 69 75 71 73 64 ...
 $ eet    : int  2 1 2 2 2 1 2 2 2 2 ...
 $ g2     : num  10.26 NA 9.99 3.57 22.56 ...
 $ grade  : int  2 3 3 2 4 3 2 3 3 3 ...
 $ gleason: int  4 8 7 4 8 7 NA 7 6 7 ...
 $ ploidy : Factor w/ 3 levels "diploid","tetraploid",..: 1 3 1 1 2 1 2 3 1 2 ...
> progstat <- factor(stagec$pgstat, levels = 0:1, labels = c("No", "Prog"))
> str(progstat)
 Factor w/ 2 levels "No","Prog": 1 1 2 2 2 1 1 1 2 1 ...
```

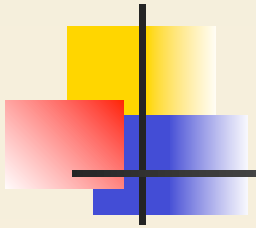


```
> table(progstat)
progstat
  No Prog
  92   54
> summary(stagec)
     pgtime           pgstat             age             eet              g2
 Min.   : 0.300   Min.   :0.0000   Min.   :47   Min.   :1.00   Min.   : 2.400
 1st Qu.: 3.700   1st Qu.:0.0000   1st Qu.:59   1st Qu.:1.75   1st Qu.: 9.215
 Median : 5.900   Median :0.0000   Median :63   Median :2.00   Median :13.010
 Mean   : 6.324   Mean   :0.3699   Mean   :63   Mean   :1.75   Mean   :14.275
 3rd Qu.: 7.900   3rd Qu.:1.0000   3rd Qu.:67   3rd Qu.:2.00   3rd Qu.:16.715
 Max.   :17.700   Max.   :1.0000   Max.   :75   Max.   :2.00   Max.   :54.930
                                               NA's   :2      NA's   :7

     grade          gleason             ploidy
 Min.   :1.00   Min.   : 3.00   diploid   :67
 1st Qu.:2.00   1st Qu.: 5.00   tetraploid:68
 Median :3.00   Median : 6.00   aneuploid :11
 Mean   :2.61   Mean   : 6.35
 3rd Qu.:3.00   3rd Qu.: 7.00
 Max.   :4.00   Max.   :10.00
                NA's   :3
```

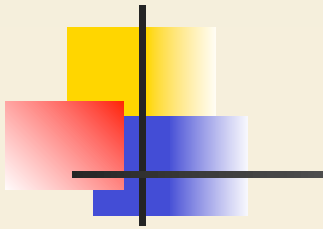
Build the model and display the tree

```
> cfit <- rpart(progstat ~ age + eet + g2 + grade + gleason + ploidy, data = stagec, method = 'class')
> plot(cfit)
> text(cfit)
```





```
> print(cfit)
n= 146

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

 1) root 146 54 No (0.6301370 0.3698630)
   2) grade< 2.5 61  9 No (0.8524590 0.1475410) *
   3) grade>=2.5 85 40 Prog (0.4705882 0.5294118)
     6) g2< 13.2 40 17 No (0.5750000 0.4250000)
      12) ploidy=diploid,tetraploid 31 11 No (0.6451613 0.3548387)
        24) g2>=11.845 7  1 No (0.8571429 0.1428571) *
        25) g2< 11.845 24 10 No (0.5833333 0.4166667)
          50) g2< 11.005 17  5 No (0.7058824 0.2941176) *
          51) g2>=11.005 7  2 Prog (0.2857143 0.7142857) *
      13) ploidy=aneuploid 9  3 Prog (0.3333333 0.6666667) *
     7) g2>=13.2 45 17 Prog (0.3777778 0.6222222)
      14) g2>=17.91 22  8 No (0.6363636 0.3636364)
        28) age>=62.5 15  4 No (0.7333333 0.2666667) *
        29) age< 62.5 7  3 Prog (0.4285714 0.5714286) *
      15) g2< 17.91 23  3 Prog (0.1304348 0.8695652) *
```



```
> summary(cfit)
Call:
rpart(formula = progstat ~ age + eet + g2 + grade + gleason +
    ploidy, data = stagec, method = "class")
  n= 146

          CP nsplit rel error   xerror      xstd
1 0.10493827      0 1.0000000 1.000000 0.1080241
2 0.05555556      3 0.6851852 1.092593 0.1098033
3 0.02777778      4 0.6296296 1.037037 0.1088041
4 0.01851852      6 0.5740741 1.037037 0.1088041
5 0.01000000      7 0.5555556 0.962963 0.1071508

Variable importance
     g2   grade gleason  ploidy     age     eet
     30      28      20      13       7       2

Node number 1: 146 observations,    complexity param=0.1049383
  predicted class=No    expected loss=0.369863  P(node) =1
    class counts:    92    54
   probabilities: 0.630 0.370
  left son=2 (61 obs) right son=3 (85 obs)
  Primary splits:
      grade   < 2.5    to the left,  improve=10.357590, (0 missing)
      gleason < 5.5    to the left,  improve= 8.399574, (3 missing)
      ploidy  splits as  LRR,        improve= 7.656533, (0 missing)
      g2      < 13.2   to the left,  improve= 7.186766, (7 missing)
      age     < 58.5   to the right, improve= 1.388128, (0 missing)
  Surrogate splits:
      gleason < 5.5    to the left,  agree=0.863, adj=0.672, (0 split)
      ploidy  splits as  LRR,        agree=0.644, adj=0.148, (0 split)
      g2      < 9.945  to the left,  agree=0.630, adj=0.115, (0 split)
      age     < 66.5   to the right, agree=0.589, adj=0.016, (0 split)
```

Model summary (cont.)

```
Node number 2: 61 observations
  predicted class=No    expected loss=0.147541  P(node) =0.4178082
    class counts:    52     9
   probabilities: 0.852 0.148

Node number 3: 85 observations,    complexity param=0.1049383
  predicted class=Prog  expected loss=0.4705882  P(node) =0.5821918
    class counts:    40    45
   probabilities: 0.471 0.529
  left son=6 (40 obs) right son=7 (45 obs)
  Primary splits:
      g2      < 13.2   to the left,  improve=2.1781360, (6 missing)
      ploidy  splits as  LRR,        improve=1.9834830, (0 missing)
      age     < 56.5   to the right, improve=1.6596080, (0 missing)
      gleason < 8.5    to the left,  improve=1.6386550, (0 missing)
      eet     < 1.5    to the right, improve=0.1086108, (1 missing)
  Surrogate splits:
      ploidy  splits as  LRL,        agree=0.962, adj=0.914, (6 split)
      age     < 68.5   to the right, agree=0.608, adj=0.114, (0 split)
      gleason < 6.5    to the left,  agree=0.582, adj=0.057, (0 split)
```

...

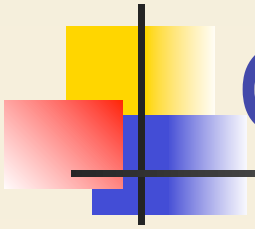
How good is the model?

```
> pred<-predict(cfit, newdata=stagec, type="class")
> confusion.mat<-table(progstat,pred)
> confusion.mat
        pred
progstat No Prog
    No   81   11
    Prog 19   35
> resubst.error<-1-sum(diag(confusion.mat))/sum(confusion.mat)
> resubst.error
[1] 0.2054795
```

Testing on the training data

```
> res.cv<-r.cv.cart.folds(formula="progstat ~ age + eet + g2 + grade
+ + gleason + ploidy",data = stagec)
> confusion.mat2<-table(progstat,res.cv)
> confusion.mat2
        res.cv
progstat No Prog
    No   71   21
    Prog 33   21
> error<-1-sum(diag(confusion.mat2))/sum(confusion.mat2)
> error
[1] 0.369863
```

Testing using cross-validation



Cross-validation

- Split the data into n folds (ex. n=10) of equal size blocks (random selection)
- For all folds do
  - Learn the model on all except the current fold
  - Predict on the current fold
- On our example:
  - 10-fold cross validation?

* Default parameter for cross-validation in rpart is set to 10

Regression trees: Car example

- Data exploration and model building

```
> str(cu.summary)
'data.frame':   117 obs. of  5 variables:
 $ Price      : num  11950 6851 6995 8895 7402 ...
 $ Country    : Factor w/ 10 levels "Brazil","England",..: 5 5 10 10 10 7 5 6 6 7 ...
 $ Reliability: Ord.factor w/ 5 levels "Much worse"<"worse"<..: 5 NA 1 4 2 4 NA 5 5 2 ...
 $ Mileage    : num  NA NA NA 33 33 37 NA NA 32 NA ...
 $ Type       : Factor w/ 6 levels "Compact","Large",..: 4 4 4 4 4 4 4 4 4 4 ...
> summary(cu.summary)
     Price            Country        Reliability    Mileage          Type
 Min.   : 5866   USA       :49   Much worse :18   Min.   :18.00   Compact:22
 1st Qu.:10125   Japan     :31   worse      :12   1st Qu.:21.00   Large  : 7
 Median :13150   Germany   :11   average    :26   Median :23.00   Medium :30
 Mean   :15743   Japan/USA: 9   better     : 8   Mean   :24.58   Small  :22
 3rd Qu.:18900   Korea     : 5   Much better:21   3rd Qu.:27.00   Sporty :26
 Max.   :41990   Sweden    : 5   NA's       :32   Max.   :37.00   Van    :10
                 (Other)   : 7                    NA's   :57
> rfit <- rpart(Mileage~Price + Country + Reliability + Type, method="anova", data=cu.summary)
```

Result

- The tree

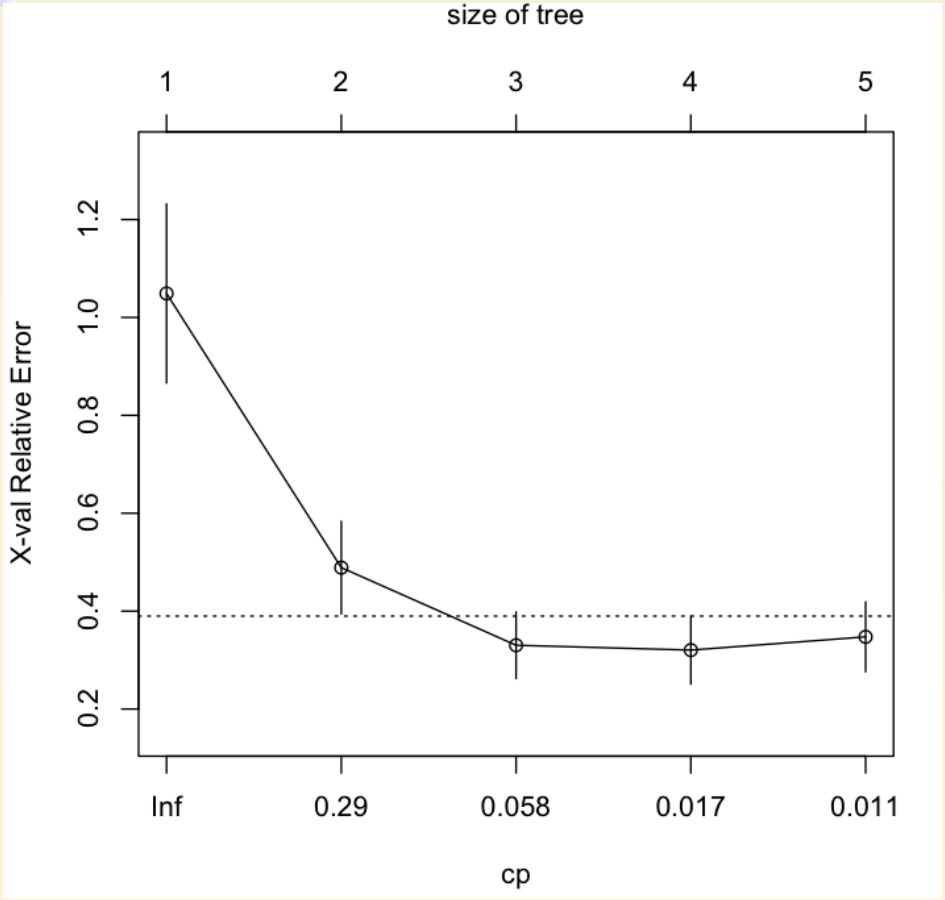
```
> print(rfit)
n=60 (57 observations deleted due to missingness)

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

 1) root 60 1354.58300 24.58333
   2) Price>=9446.5 48  407.91670 22.70833
     4) Type=Large,Medium,Van 23   66.86957 20.69565
       8) Type=Large,Van 10   22.10000 19.30000 *
       9) Type=Medium 13   10.30769 21.76923 *
     5) Type=Compact,Small,Sporty 25  162.16000 24.56000
      10) Price>=11484.5 14  107.71430 23.85714 *
      11) Price< 11484.5 11   38.72727 25.45455 *
   3) Price< 9446.5 12  102.91670 32.08333 *
```

Result

- Partial summary of the model

```
> summary(rfit)
Call:
rpart(formula = Mileage ~ Price + Country + Reliability + Type,
    data = cu.summary, method = "anova")
  n=60 (57 observations deleted due to missingness)

          CP nsplit rel error    xerror       xstd
1 0.62288527      0 1.0000000 1.0489627 0.18289799
2 0.13206061      1 0.3771147 0.4888716 0.09458470
3 0.02544094      2 0.2450541 0.3303054 0.06804181
4 0.01160389      3 0.2196132 0.3204023 0.06944997
5 0.01000000      4 0.2080093 0.3474435 0.07137415

Variable importance
  Price    Type Country
     48      42      10
```

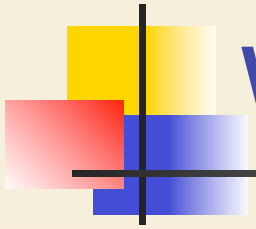
The relative error based on cp using 'plotcp'



Prune the tree

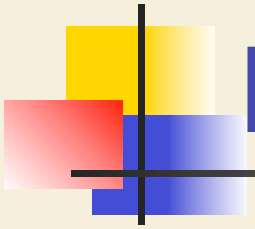
- Using the complexity factor

```
> prfit<- prune(rfit, cp=0.018)
> summary(prfit)
Call:
rpart(formula = Mileage ~ Price + Country + Reliability + Type,
    data = cu.summary, method = "anova")
  n=60 (57 observations deleted due to missingness)

          CP nsplit rel error    xerror       xstd
1 0.62288527      0 1.0000000 1.0489627 0.18289799
2 0.13206061      1 0.3771147 0.4888716 0.09458470
3 0.02544094      2 0.2450541 0.3303054 0.06804181
4 0.01800000      3 0.2196132 0.3204023 0.06944997

Variable importance
  Price    Type Country
     48      43       9
```



Warning

- Modelling ≠ R
  - Another tool is WEKA
- Tree ≠ CART
  - Random Forest (in R package randomForest)
- CART ≠ rpart
  - See package tree



References

- Breiman L., Friedman J. H., Olshen R. A., and Stone, C. J. (1984) Classification and Regression Trees. Wadsworth.
- Therneau, T. M., and Atkinson, E. J., "An Introduction to Recursive Partitioning Using the RPART Routines”
- Applied Statistics