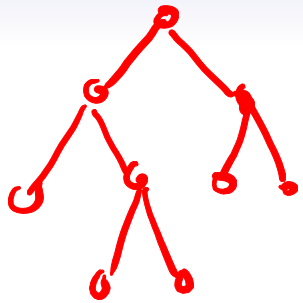
Stat 435 Intro to Statistical Machine Learning

Week 9: Trees (one more week to go, hang in there!)

Richard Li

May 24, 2017

Review



Do they still make sense to you?

- Regression tree
- Classification tree

k terminal nodes

k-1 internal nodes.

- terminal nodes / leaves

$R_1 = \{x \mid x_1 < 10\}$ & $R_2 = \{x \mid x_1 > 10\}$

- Growing a tree / Recursive binary splitting
- Pruning a tree / Cost complexity pruning

Find Best $\{j, s\}$

$\{x \mid x_j < s\}$ & $\{x \mid x_j > s\}$

$$\sum_{m=1}^{|T|} RSS_m + \alpha |T|$$

Regression tree (Alg. 8.1)

- Select $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_M\}$
- Divide data into K folds
- For each training fold
  - Grow a tree until { each node < n obs. / each node has MSE < C
  - Prun the tree with each of the α
    i.e. Find T s.t $\sum_{m}^{|T|} RSS_m + \alpha |T|$ is minimized
  - Calculate test MSE for each α
- Average K test MSE for each α
- Find the α^* that minimizes average test MSE

Classification tree

RSS_m

$\Downarrow$

$\sum_{m}^{|T|} N_m G_m$

$G_m = \begin{cases} 1. & \text{Error rate} \\ 2. & \text{Gini index} \\ 3. & \text{Cross-entropy} \end{cases}$

(400, 400)

(300, 100) (100, 300)

Build: Gini index / Cross entropy

Prun: Depends

(200, 400)

(200, 0)

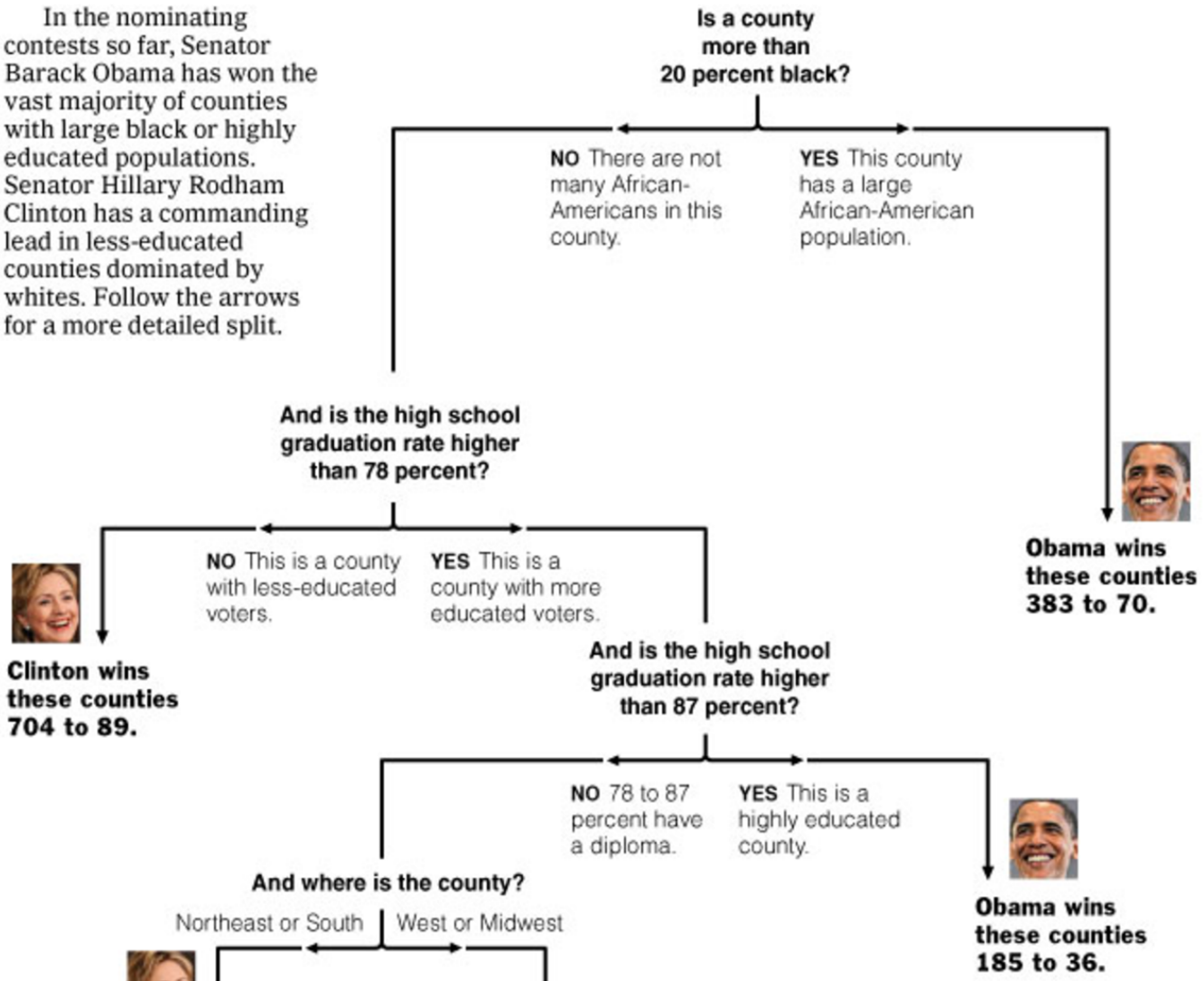
pure

Example

The New York Times — April 16, 2008

Decision Tree: The Obama-Clinton Divide

In the nominating contests so far, Senator Barack Obama has won the vast majority of counties with large black or highly educated populations. Senator Hillary Rodham Clinton has a commanding lead in less-educated counties dominated by whites. Follow the arrows for a more detailed split.



OK, hands-on example (California housing price)

```
calif <- read.table("https://raw.githubusercontent.com/jbryer/CompStats/master/Data/cadata.dat", header = TRUE)
dim(calif)
```

```
## [1] 20640     9
```

```
head(calif)
```

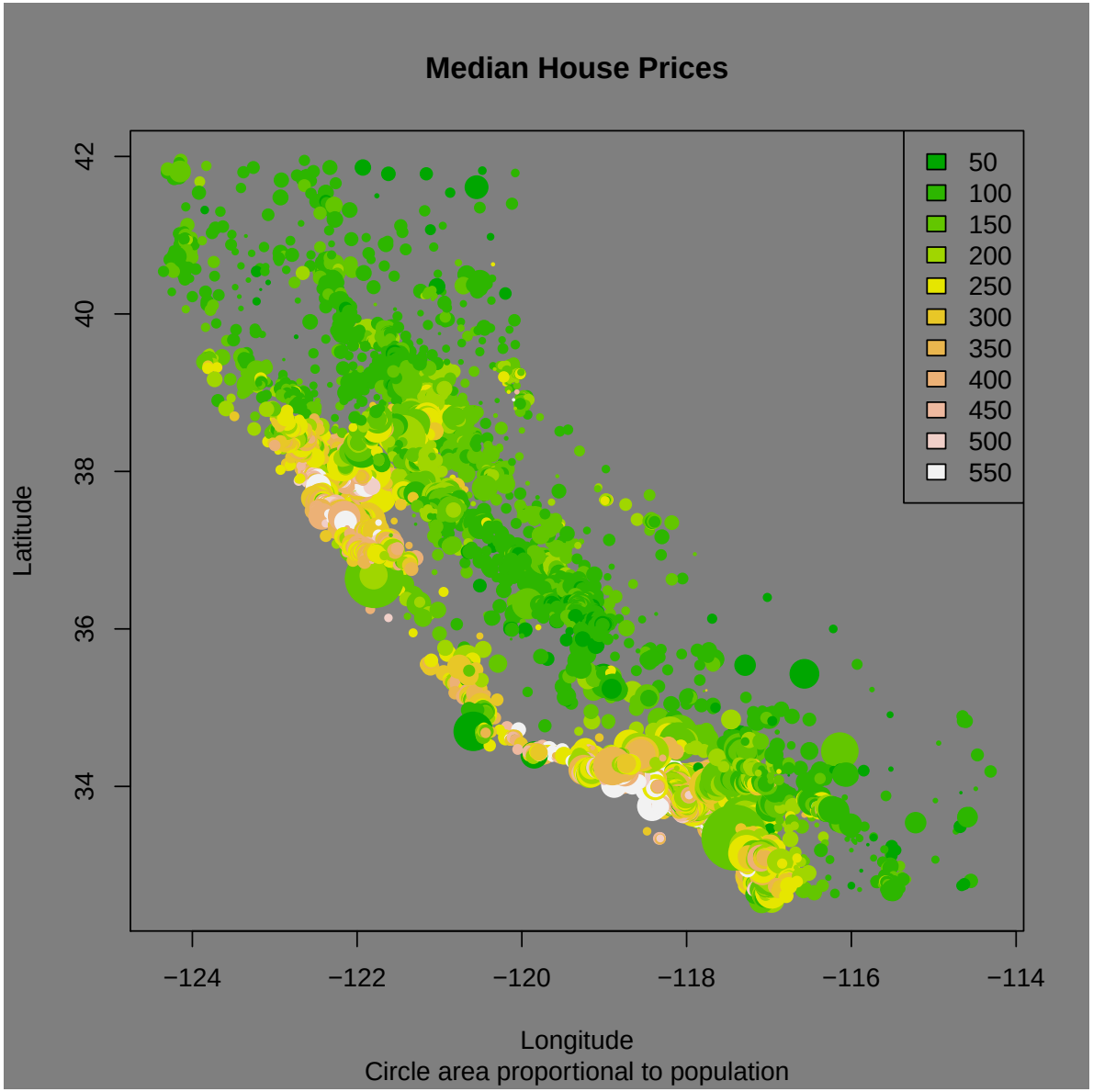
```
##   MedianHouseValue MedianIncome MedianHouseAge TotalRooms TotalBedrooms Population
## 1           452600       8.3252             41        880           129        322
## 2           358500       8.3014             21       7099          1106       2401
## 3           352100       7.2574             52       1467           190        496
## 4           341300       5.6431             52       1274           235        558
## 5           342200       3.8462             52       1627           280        565
## 6           269700       4.0368             52        919           213        413
##   Households Latitude Longitude
## 1        126    37.88   -122.23
## 2       1138    37.86   -122.22
## 3        177    37.85   -122.24
## 4        219    37.85   -122.25
## 5        259    37.85   -122.25
## 6        193    37.85   -122.25
```

Visualization

(The codes are just for your curiosity)

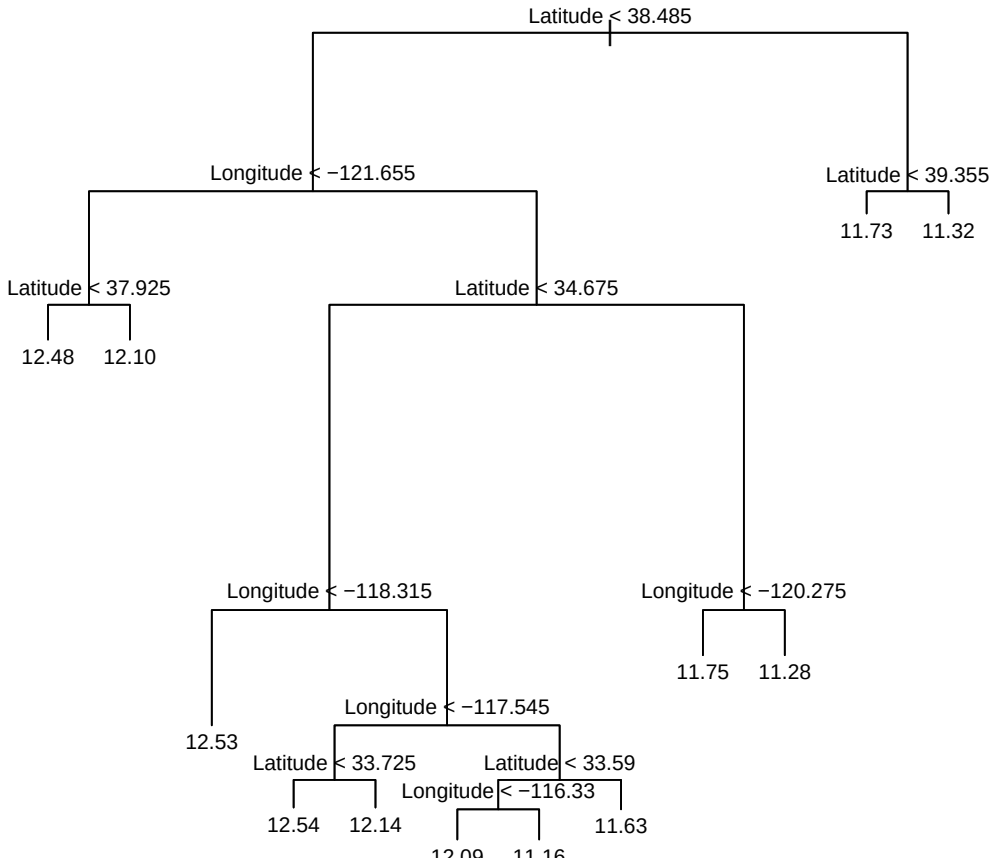
```
par(bg = 'gray50')
plot(calif$Longitude,calif$Latitude,pch=21,
        col=terrain.colors(11)[1+floor(calif$MedianHouseValue/50e3)],
        bg=terrain.colors(11)[1+floor(calif$MedianHouseValue/50e3)],
        cex=sqrt(calif$Population/median(calif$Population)),
        xlab="Longitude",ylab="Latitude",main="Median House Prices",
        sub="Circle area proportional to population")
legend(x="topright",legend=(50*(1:11)),fill=terrain.colors(11))
```

Visualization



Fitting a tree with two predictors

```
library(tree)
treefit <- tree(log(MedianHouseValue) ~ Longitude+Latitude,data=calif)
par(bg = 'white')
plot(treefit)
text(treefit,cex=0.75)
```

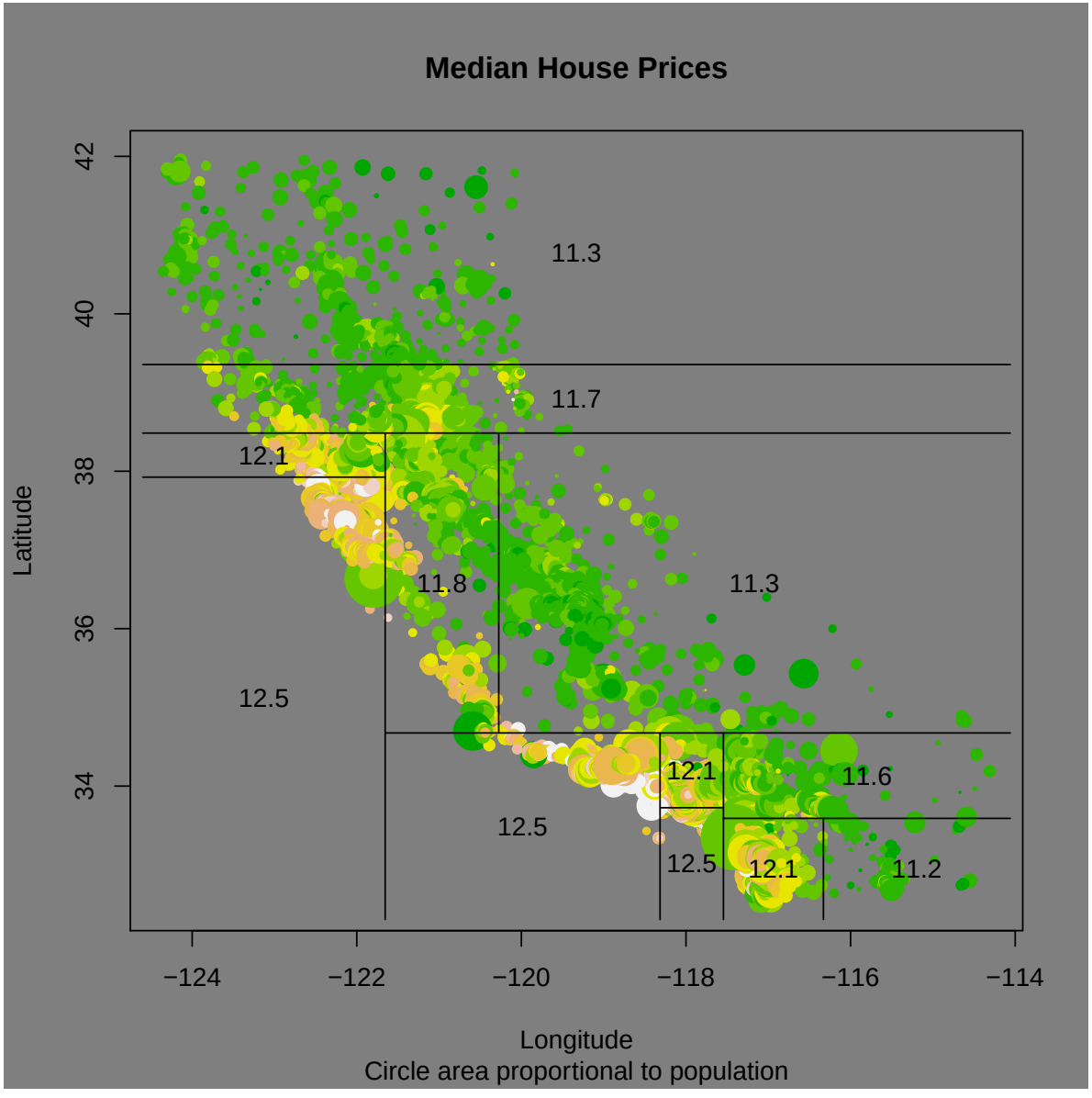


Show this tree on map

(The codes are just for your curiosity)

```
par(bg = 'gray50')
plot(calif$Longitude,calif$Latitude,pch=21,
        col=terrain.colors(11)[1+floor(calif$MedianHouseValue/50e3)],
        bg=terrain.colors(11)[1+floor(calif$MedianHouseValue/50e3)],
        cex=sqrt(calif$Population/median(calif$Population)),
        xlab="Longitude",ylab="Latitude",main="Median House Prices",
        sub="Circle area proportional to population")
partition.tree(treefit,ordvars=c("Longitude","Latitude"),add=TRUE)
```

Show this tree on map

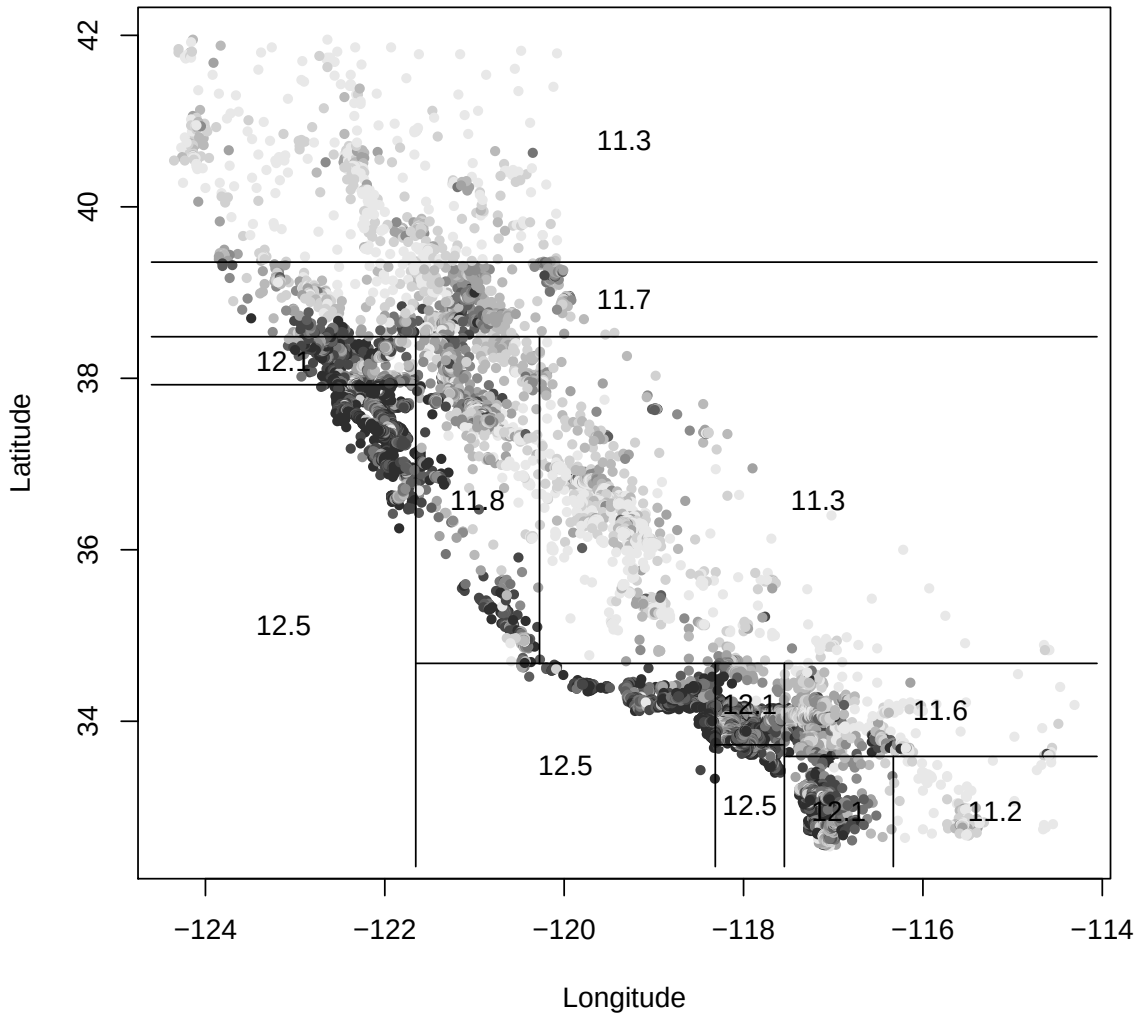


Another visualization

(The codes are just for your curiosity)

```
par(bg = 'white')
price.deciles <- quantile(calif$MedianHouseValue,0:10/10)
cut.prices <- cut(calif$MedianHouseValue,price.deciles,include.lowest=TRUE)
plot(calif$Longitude,calif$Latitude,col=grey(10:2/11)[cut.prices],pch=20,
        xlab="Longitude",ylab="Latitude")
partition.tree(treefit,ordvars=c("Longitude","Latitude"),add=TRUE)
```

Another visualization



Summary of the tree

```
summary(treefit)

##
## Regression tree:
## tree(formula = log(MedianHouseValue) ~ Longitude + Latitude,
##     data = calif)
## Number of terminal nodes:  12
## Residual mean deviance:  0.1662 = 3429 / 20630
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.75900 -0.26080 -0.01359  0.00000  0.26310  1.84100
```

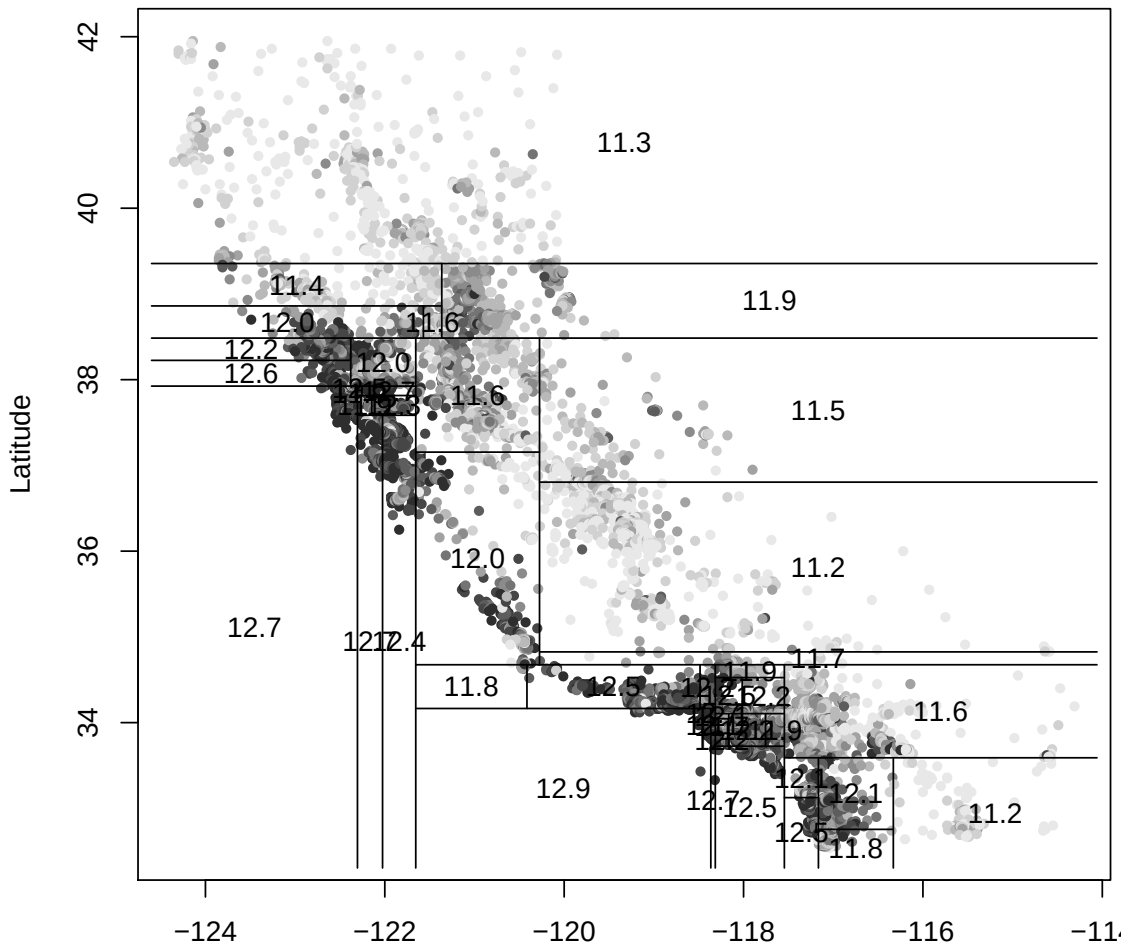
Fit a bigger tree

```
treefit2 <- tree(log(MedianHouseValue) ~ Longitude+Latitude,data=calif,
                                 mindev = 0.002)
summary(treefit2)

##
## Regression tree:
## tree(formula = log(MedianHouseValue) ~ Longitude + Latitude,
##     data = calif, mindev = 0.002)
## Number of terminal nodes:  43
## Residual mean deviance:  0.1163 = 2396 / 20600
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.94700 -0.21970 -0.01628  0.00000  0.22110  1.48800
```

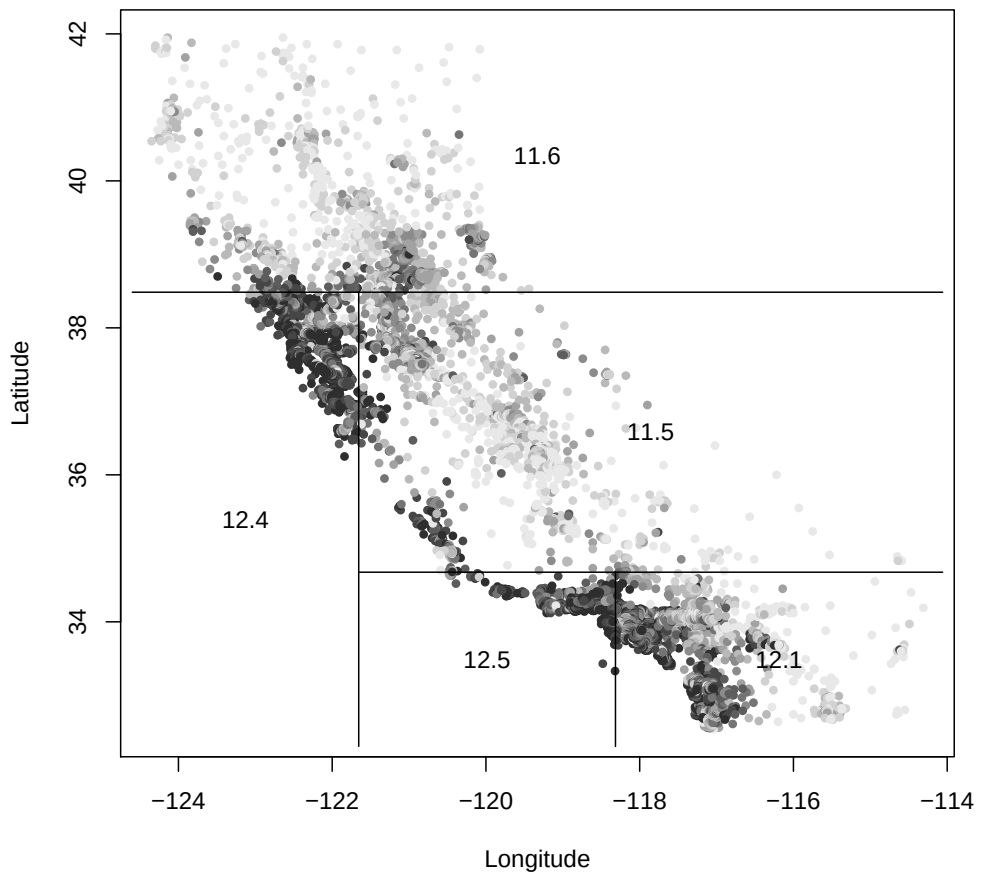
Look at this bigger tree

```
plot(calif$Longitude,calif$Latitude,col=grey(10:2/11)[cut.prices],pch=20,
        xlab="Longitude",ylab="Latitude")
partition.tree(treefit2,ordvars=c("Longitude","Latitude"),add=TRUE)
```



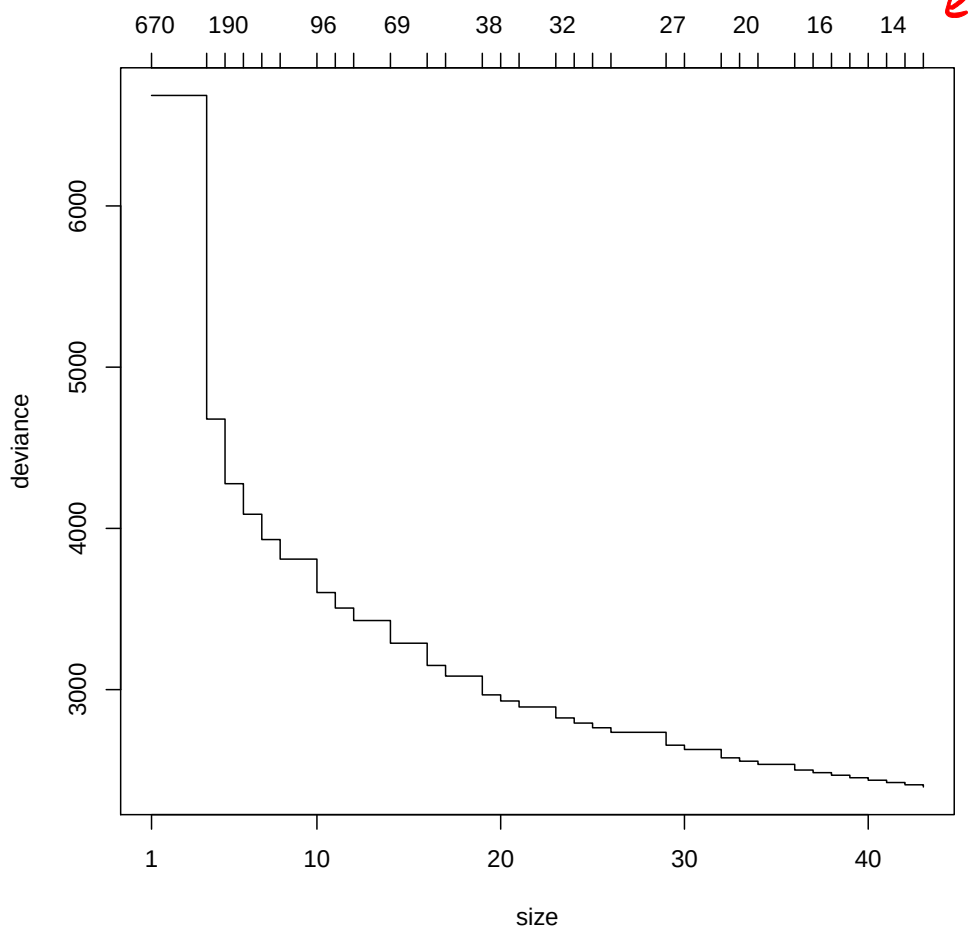
Prune the tree to a fixed size

```
treeprun5 <- prune.tree(treefit2, best=5)
plot(calif$Longitude,calif$Latitude,col=grey(10:2/11)[cut.prices],pch=20,
        xlab="Longitude",ylab="Latitude")
partition.tree(treeprun5,ordvars=c("Longitude","Latitude"),add=TRUE)
```



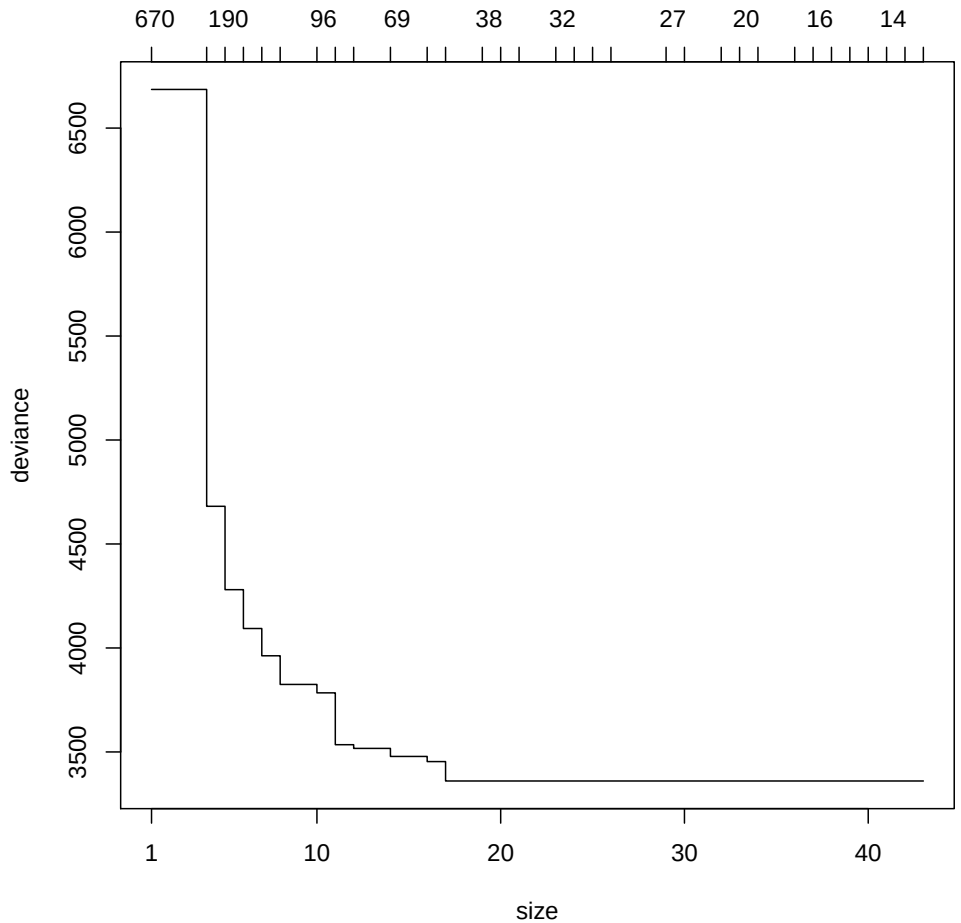
Prune the tree

```
treeprun <- prune.tree(treefit2)
plot(treeprun)
```



Cross-validation on trees (regression tree algorithm 8.1)

```
treecv <- cv.tree(treefit2)
plot(treecv)
```



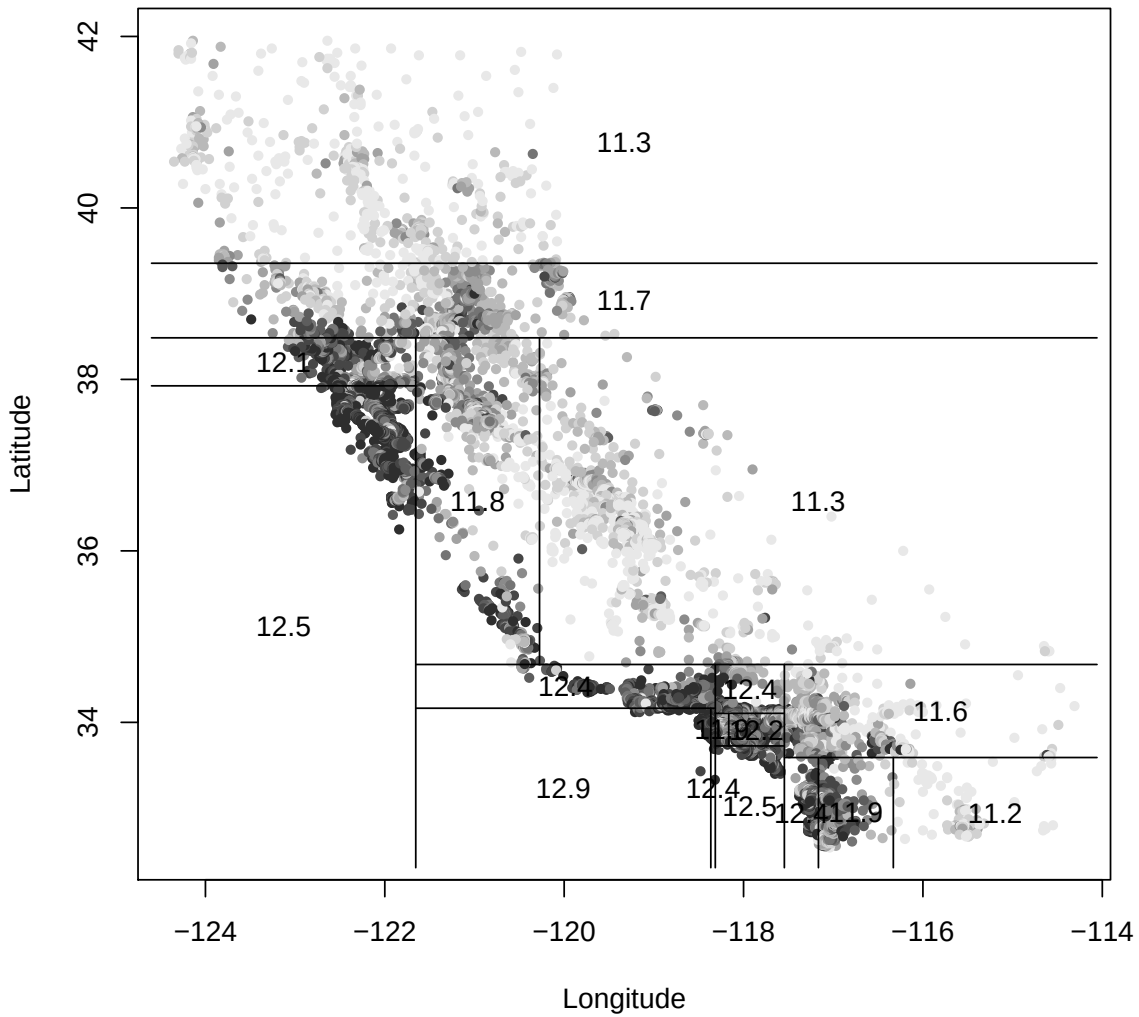
Optimal tree from cross-validation

```
opt.trees <- which(treecv$dev == min(treecv$dev))
best.leaves <- min(treecv$size[opt.trees])
best.leaves
treefit3 <- prune.tree(treefit2, best=best.leaves)

plot(calif$Longitude,calif$Latitude,col=grey(10:2/11)[cut.prices],pch=20,
        xlab="Longitude",ylab="Latitude")
partition.tree(treefit3,ordvars=c("Longitude","Latitude"),add=TRUE)
```

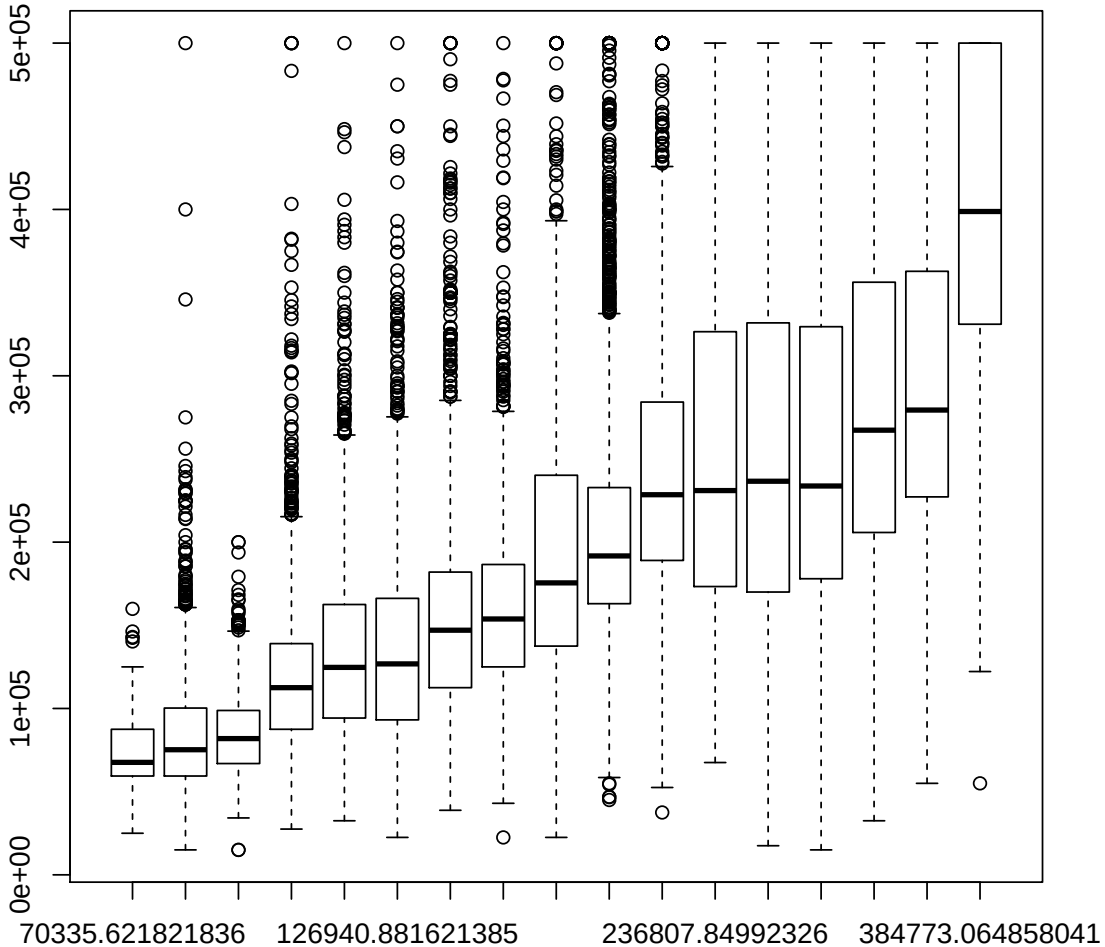
Optimal tree from cross-validation

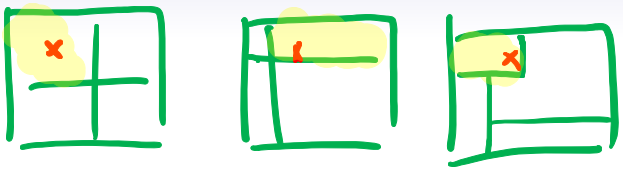
```
## [1] 17
```



Fitted v.s. true median house values

```
boxplot(calif$MedianHouseValue ~ exp(predict(treefit3, newdata = calif)))
```





Review

Ensemble method
* predictive ✓
* descriptive ✗

Do they still make sense to you?

- Bagging / Bootstrap aggregation
- Boosting
- Out-of-bag error → P_r(i-th obs in a bootstrap sample) → $\approx 2/3$
- Random Forest — random version of bagging
- What is random about random forest
  1. rows
  2. columns.

Fit regression tree on half of the data

```
library(randomForest)
set.seed(1)
train <- sample(1:nrow(calif), nrow(calif)/2)
```

```
treefit0 <- tree(log(MedianHouseValue) ~ ., data=calif, subset = train)
treecv <-  cv.tree(treefit0)
opt.trees <- which(treecv$dev == min(treecv$dev))
best.leaves <- min(treecv$size[opt.trees])
treefit <- prune.tree(treefit0, best=best.leaves)
```

Bagging and random forest

```
bagfit <- randomForest(log(MedianHouseValue) ~ ., data=calif, subset = train,
                                                mtry = ncol(calif)-1)
bagfit

##
## Call:
##  randomForest(formula = log(MedianHouseValue) ~ ., data = calif,      mtry = ncol(calif)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 8
##
##           Mean of squared residuals: 0.05691351
##                     % Var explained: 82.41

rffit <- randomForest(log(MedianHouseValue) ~ ., data=calif, subset = train,
                                                mtry = 3)
rffit

##
## Call:
##  randomForest(formula = log(MedianHouseValue) ~ ., data = calif,      mtry = 3, subset =
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 0.06044418
##                     % Var explained: 81.31
```

Test MSE

```
y <- log(calif[-train, "MedianHouseValue"])
yhat.tree <- predict(treefit, newdata=calif[-train ,])
yhat.bag <- predict(bagfit, newdata=calif[-train ,])
yhat.rf <- predict(rffit, newdata=calif[-train ,])
c(mean((yhat.tree-y)^2), mean((yhat.bag-y)^2), mean((yhat.rf-y)^2))

## [1] 0.13907807 0.05495472 0.05870952
```

There's a lot more to trees and forest if you are interested

Three-Eyed Raven: *You won't be here forever. You won't be an old man in a tree. But before you leave, you must learn.*

Bran Stark: *Learn what?*

Three-Eyed Raven: *Everything.*

(Script from Game of Thrones: Oathbreaker (#6.3))

You will receive course evaluation email tomorrow/soon