10 fold Cross Validation and 1 SE Rule

CART

Based on Chew C. H. (2020) textbook: AI, Analytics and Data Science. Vol 1., Chap 8.

CP Table shows min CV error and its 1SE.

```
31  # prints out the pruning sequence and 10-fold CV errors, as a table.
32  printcp(m2)
```

```
Root node error: 13/31 = 0.41935

n= 31

        CP nsplit rel error  xerror    xstd
1 0.615385      0   1.00000 1.00000 0.21134
2 0.051282      1   0.38462 0.69231 0.19441
3 0.038462      4   0.23077 0.76923 0.20021
4 0.000000     10   0.00000 0.84615 0.20492
```

Geometric mean CP of the 2nd Tree = sqrt(0.0512 * 0.615) ≈ 0.18

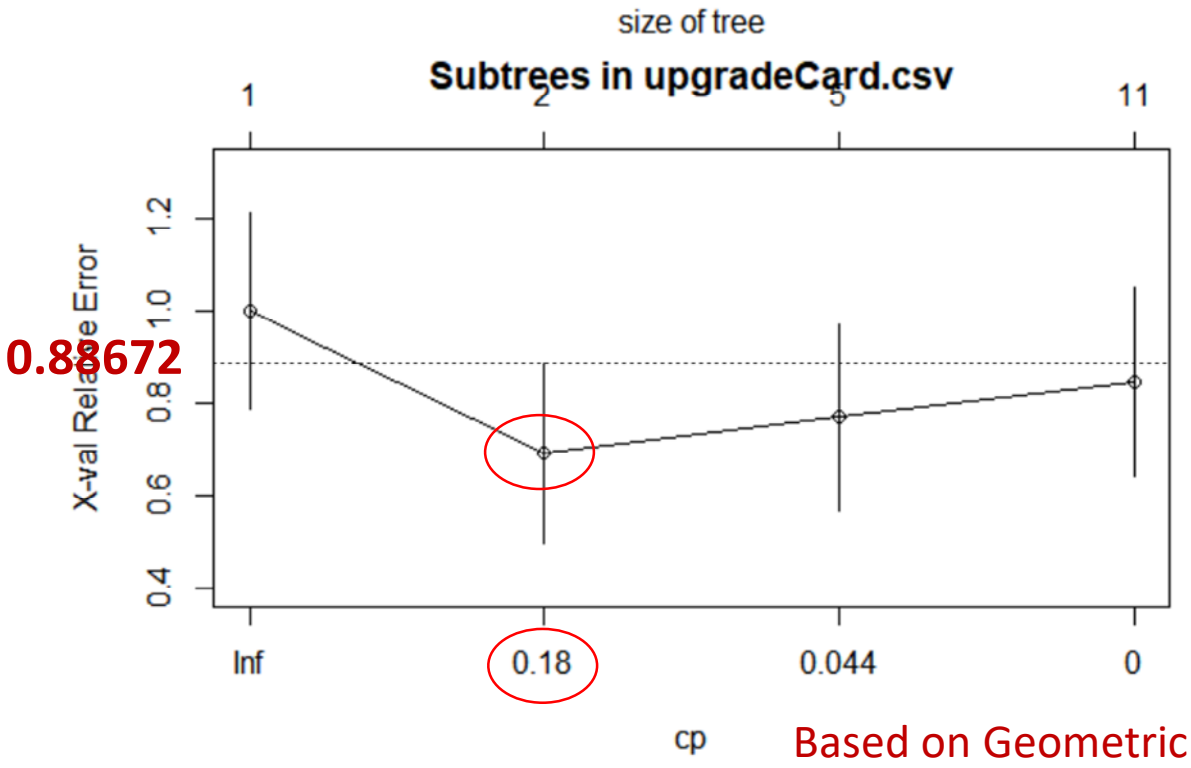
Min CV Error = 0.69231

1SE = 0.19441

CV Error Cap = 0.69231 + 0.19441 = **0.88672**

CV error cap is displayed in plotcp()

```
34  # Display the pruning sequence and 10-fold CV errors, as a chart.
35  plotcp(m2, main = "Subtrees in upgradeCard.csv")
```

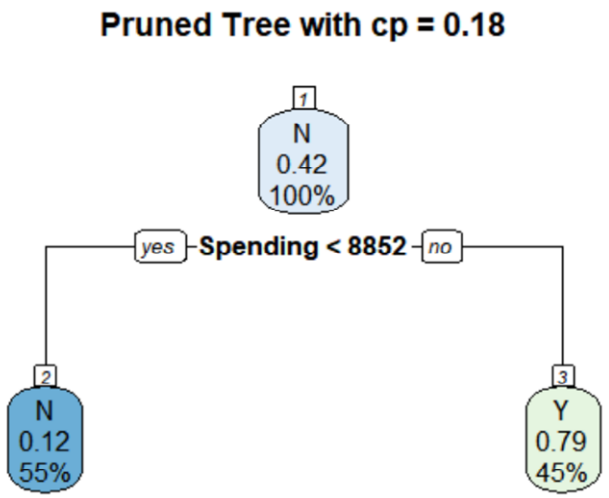


Based on Geometric mean

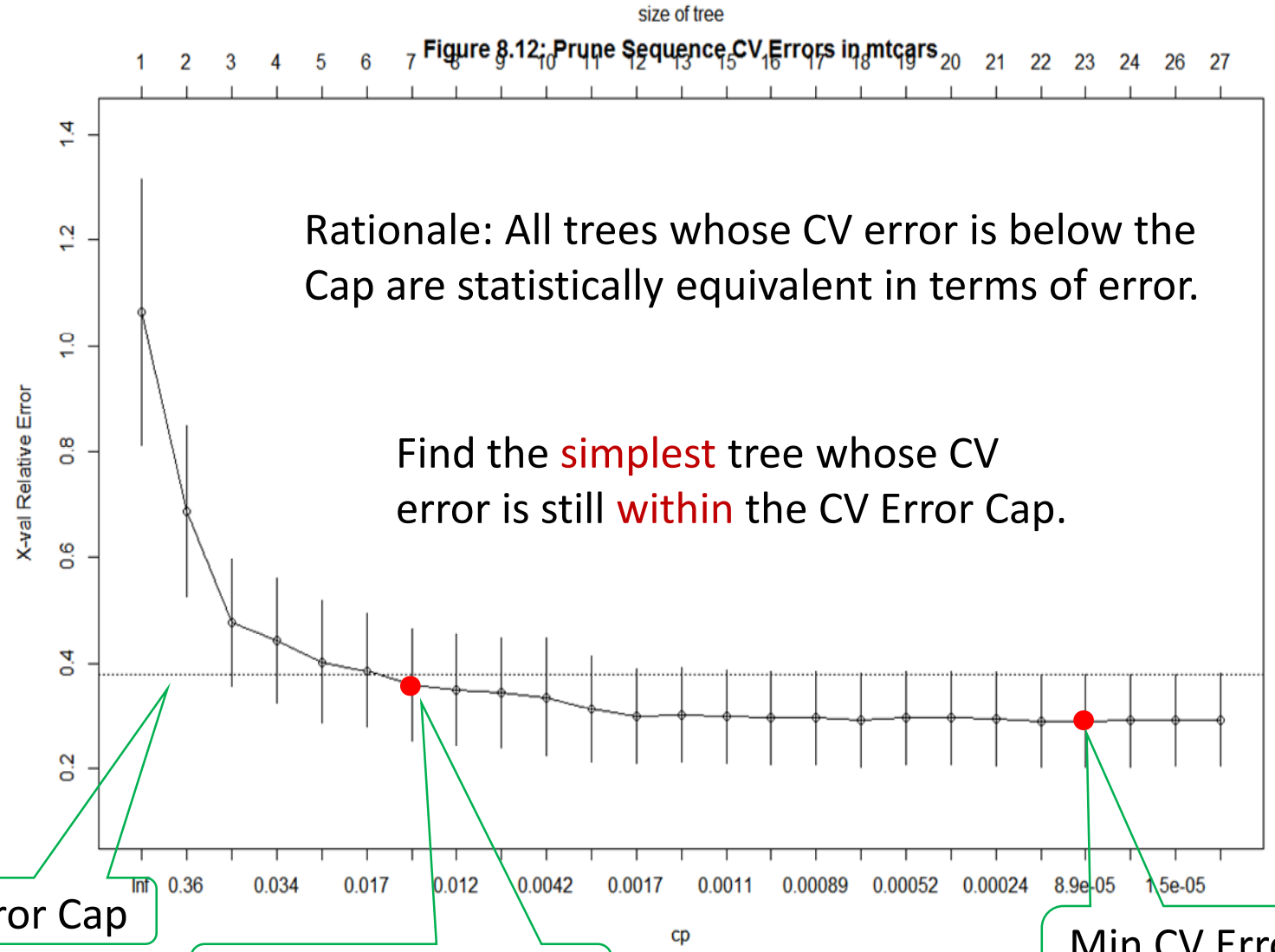
Get a specific subtree via prune() by pruning the maximal tree m2 with a specific value of cp

```
41  cp1 <- 0.18
42
43  m3 <- prune(m2, cp = cp1)
44
45  printcp(m3)
46
47  # plots the tree m3 pruned using cp1.
48  rpart.plot(m3, nn= T, main = "Pruned Tree with cp = 0.18")
```

```
Root node error: 13/31 = 0.41935

n= 31

        CP nsplit rel error  xerror    xstd
1 0.61538      0   1.00000 1.00000 0.21134
2 0.18000      1   0.38462 0.69231 0.19441
```



CART on mtcars dataset shows 25 trees



Rationale: All trees whose CV error is below the Cap are statistically equivalent in terms of error.

Find the simplest tree whose CV error is still within the CV Error Cap.

CV Error Cap

Optimal Tree #7

Min CV Error
Tree #22

Trainset error vs 10 fold Cross Validation (CV) error

```
31  # prints out the pruning sequence and 10-fold CV errors, as a table.
32  printcp(m2)
```

```
Root node error: 13/31 = 0.41935

n= 31

        CP nsplit rel error  xerror     xstd
1 0.615385      0   1.00000 1.00000 0.21134
2 0.051282      1   0.38462 0.69231 0.19441
3 0.038462      4   0.23077 0.76923 0.20021
4 0.000000     10   0.00000 0.84615 0.20492
```

Trainset error

10-fold CV error

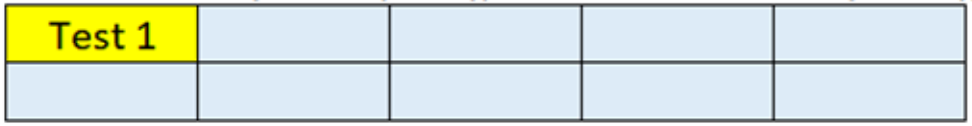
Plotted as the y-coor in plotcp() chart.

What is 10-fold Cross Validation Error?

That's why we need to set.seed() before executing rpart.

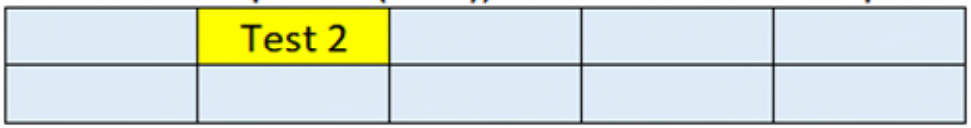
0. Randomly split the data into 10 subsets.

1. Train on 9 pieces (blue), test on unseen 1st piece (yellow).



Trainset error 1,
Testset error 1.

2. Train on 9 pieces (blue), test on unseen 2nd piece (yellow).



Trainset error 2,
Testset error 2.

10. Train on 9 pieces (blue), test on unseen 10th piece (yellow).

Trainset error 10,
Testset error 10.

1 SE Rule is just a guideline to select the optimal tree.

- Min CV error tree is an unstable solution.
  - A small change in data could lead to a different solution.
  - Depends on the random subsets in 10 fold CV.
- 1 SE rule is more stable.
  - Many trees are statistically equivalent in terms of errors.
  - Choose the simplest tree that still perform well.

Next Video: CART for Continuous Y

- Continuous Y:
  - How to choose the best split?
  - How to evaluate node error and overall Tree error.
  - Variable Importance.
    - How important are each of those X variables?

CART for Continuous Y

Rscript: mtcars CART.R

CART

Based on Chew C. H. (2020) textbook: AI, Analytics and Data Science. Vol 1., Chap 8.

Base R dataset: mtcars

32 cases, 11 columns.

Continuous Outcome variable Y: mpg

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1

Continuous Y: method = 'anova'

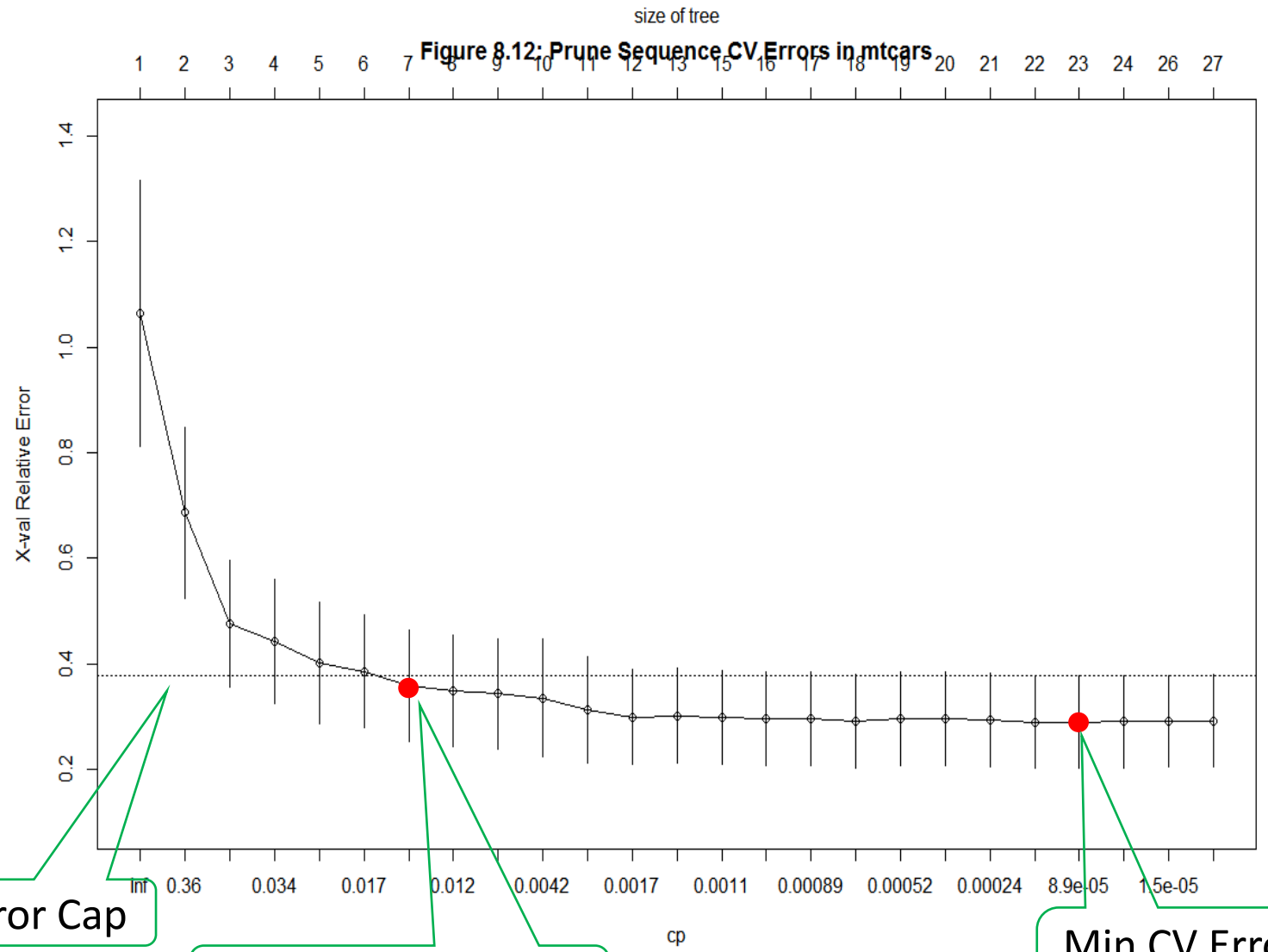
```r
data(mtcars)

library(rpart)
library(rpart.plot)     # For Enhanced tree plots

set.seed(2014)

# Continuous Y: Set method = 'anova'
cart1 <- rpart(mpg ~ ., data = mtcars, method = 'anova', control = rpart.control(minsplit = 2, cp = 0))

printcp(cart1)
## Caution: printcp() shows that if you forgot to change the default CP from 0.01 to 0,
## It would have stopped the tree growing process too early. A lot of further growth at CP < 0.01.

plotcp(cart1)
```

CP Table shows many trees with cp < 0.01

```
Root node error: 1126/32 = 35.189

n= 32

           CP nsplit  rel error  xerror     xstd
1  6.5266e-01      0 1.0000e+00 1.06389 0.252198
2  1.9470e-01      1 3.4734e-01 0.68629 0.160870
3  4.5774e-02      2 1.5264e-01 0.47604 0.119551
4  2.5328e-02      3 1.0686e-01 0.44324 0.117846
5  2.3250e-02      4 8.1534e-02 0.40281 0.115329
6  1.2488e-02      5 5.8285e-02 0.38559 0.106955
7  1.2149e-02      6 4.5796e-02 0.35818 0.105197
8  1.1647e-02      7 3.3648e-02 0.34943 0.105751
9  9.6700e-03      8 2.2000e-02 0.34357 0.104871
10 1.8010e-03      9 1.2330e-02 0.33605 0.112381
11 1.8010e-03     10 1.0529e-02 0.31304 0.099915
12 1.5156e-03     11 8.7282e-03 0.29965 0.090460
13 1.2868e-03     12 7.2125e-03 0.30216 0.090476
14 9.9907e-04     14 4.6389e-03 0.29853 0.089054
15 9.2506e-04     15 3.6399e-03 0.29704 0.089144
16 8.5254e-04     16 2.7148e-03 0.29628 0.089205
17 7.5041e-04     17 1.8623e-03 0.29221 0.089117
18 3.5967e-04     18 1.1119e-03 0.29642 0.088779
19 2.8418e-04     19 7.5219e-04 0.29642 0.088779
20 2.0011e-04     20 4.6801e-04 0.29479 0.088862
21 1.1101e-04     21 2.6790e-04 0.29055 0.086900
22 7.1045e-05     22 1.5689e-04 0.29013 0.086937
23 3.9963e-05     23 8.5846e-05 0.29080 0.086896
24 5.9204e-06     25 5.9204e-06 0.29159 0.086831
25 0.0000e+00     26 0.0000e+00 0.29237 0.087231
```

Min CV Error Tree

plotcp() on mtcars dataset shows CV Error Cap



CV Error Cap

Optimal Tree #7

Min CV Error Tree #22

To get the optimal tree from the list of 25 trees, use prune() with a specific value of cp.

```
Root node error: 1126/32 = 35.189

n= 32

           CP nsplit  rel error  xerror     xstd
1  6.5266e-01      0 1.0000e+00 1.06389 0.252198
2  1.9470e-01      1 3.4734e-01 0.68629 0.160870
3  4.5774e-02      2 1.5264e-01 0.47604 0.119551
4  2.5328e-02      3 1.0686e-01 0.44324 0.117846
5  2.3250e-02      4 8.1534e-02 0.40281 0.115329
6  1.2488e-02      5 5.8285e-02 0.38559 0.106955
7  1.2149e-02      6 4.5796e-02 0.35818 0.105197
8  1.1647e-02      7 3.3648e-02 0.34943 0.105751
```

```
31  # 7th tree is optimal. Choose any CP value betw the 6th and 7th tree CP values.
32  cp1 <- sqrt(1.2149e-02*1.2488e-02)
```

```
cp1                    0.012317333802410 3
```

```
53  # Prune the max tree using a particular CP value
54  cart2 <- prune(cart1, cp = cp1)
```

Get optimal tree (based on 1 SE rule) CV error

```
53  # Prune the max tree using a particular CP value
54  cart2 <- prune(cart1, cp = cp1)
55  printcp(cart2, digits = 3)
56  ## --- Trainset Error & CV Error ------------------------
57  ## Root node error: 1126/32 = 35.2
58  ## cart2 trainset MSE = 0.0458 * 35.2 = 1.6
59  ## cart2 CV MSE = 0.358 * 35.2 = 12.6
```

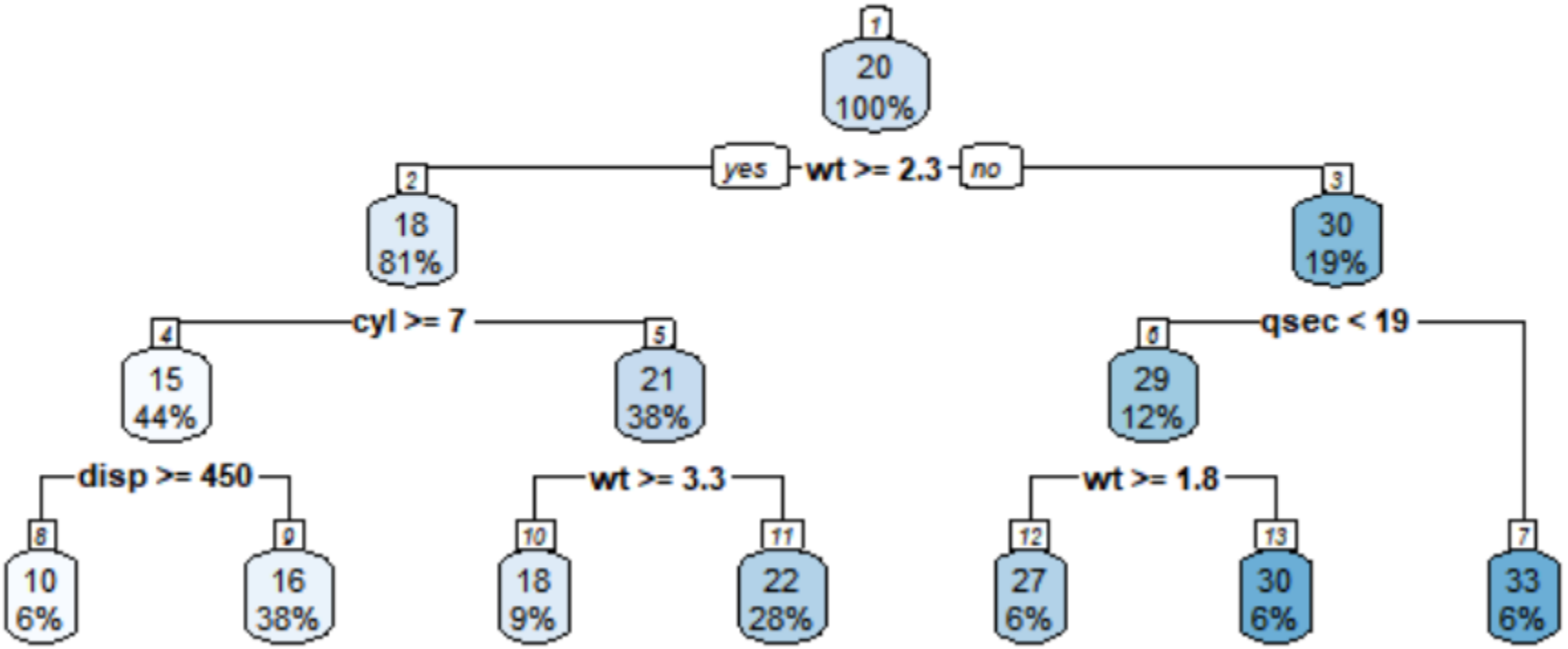
```
Root node error: 1126/32 = 35.2

n= 32

        CP nsplit rel error xerror  xstd
1 0.6527      0    1.0000  1.064 0.252
2 0.1947      1    0.3473  0.686 0.161
3 0.0458      2    0.1526  0.476 0.120
4 0.0253      3    0.1069  0.443 0.118
5 0.0232      4    0.0815  0.403 0.115
6 0.0125      5    0.0583  0.386 0.107
7 0.0123      6    0.0458  0.358 0.105
```

View tree structure on console with print()

```
61  print(cart2)
```

```
n= 32

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

 1) root 32 1126.047000 20.09062
   2) wt>=2.26 26  346.566500 17.78846
     4) cyl>=7 14   85.200000 15.10000
       8) disp>=450 2    0.000000 10.40000 *
       9) disp< 450 12   33.656670 15.88333 *
     5) cyl< 7 12   42.122500 20.92500
      10) wt>=3.3275 3    1.086667 18.36667 *
      11) wt< 3.3275 9   14.855560 21.77778 *
   3) wt< 2.26 6   44.553330 30.06667
     6) qsec< 19.185 4   14.907500 28.52500
      12) wt>=1.775 2    0.845000 26.65000 *
      13) wt< 1.775 2    0.000000 30.40000 *
     7) qsec>=19.185 2    1.125000 33.15000 *
```

```r
rpart.plot(cart2, nn = T, main = "Optimal Tree in mtcars")
## The number inside each node represent the mean value of Y.
```



Variable Importance

```
66  cart2$variable.importance
67  ## Weight has the highest importance, disp is second impt.
```

```
> cart2$variable.importance
       wt      disp        hp      drat       cyl      qsec        vs      carb
965.37479 914.94074 699.65200 393.23532 341.73192 218.72553 164.43303  14.26042
```

```
71  summary(cart2)
```

```
Variable importance
  wt disp   hp drat  cyl qsec   vs
  26   25   19   11    9    6    4
```

Next Video: Surrogates

- How CART handles missing values automatically.

Surrogates

Rscript: mtcars CART.R

CART

Based on Chew C. H. (2020) textbook: AI, Analytics and Data Science. Vol 1., Chap 8.

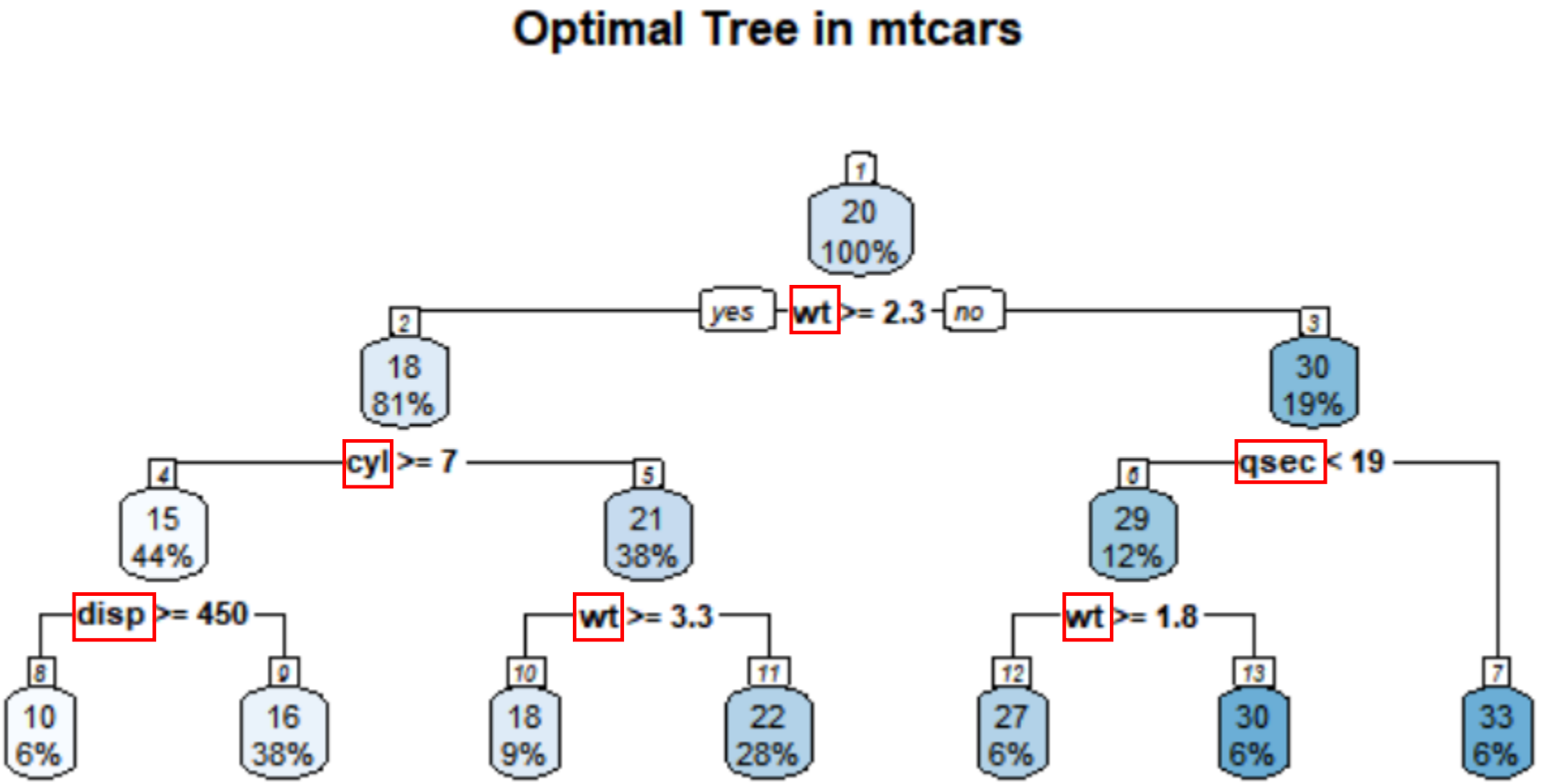
Base R dataset: mtcars

32 cases, 11 columns.

Continuous Outcome variable Y: mpg

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1

CART requires the splitting variable value to be available



Displays surrogates at each internal node via summary(). Example: At node 1 i.e. root node

```
70 # Surrogates shown in summary()
71 summary(cart2)
```

Min CP required to prune at Node 1

```
Node number 1: 32 observations,    complexity param=0.6526612
  mean=20.09062, MSE=35.18897
  left son=2 (26 obs) right son=3 (6 obs)
  Primary splits:
      wt   < 2.26   to the right, improve=0.6526612, (0 missing)
      cyl  < 5      to the right, improve=0.6431252, (0 missing)
      disp < 163.8  to the right, improve=0.6130502, (0 missing)
      hp   < 118    to the right, improve=0.6010712, (0 missing)
      vs   < 0.5    to the left,  improve=0.4409477, (0 missing)
  Surrogate splits:
      disp < 101.55 to the right, agree=0.969, adj=0.833, (0 split)
      hp   < 92     to the right, agree=0.938, adj=0.667, (0 split)
      drat < 4      to the left,  agree=0.906, adj=0.500, (0 split)
      cyl  < 5      to the right, agree=0.844, adj=0.167, (0 split)
```

This surrogate split is 96.9% similar to the best Primary split.

This surrogate split is 83.3% better than using majority rule to send NAs to left or right child.

Surrogates are only activated when there are missing values

- The dataset mtcars has no missing values.

1. Copy the data to be mtcars2.
2. Delete the wt value in first row of mtcars2.
3. Delete wt and disp values in second row of mtcars2.
4. Repeat the CART model on mtcars2.
5. View surrogates activated in node 1.

```
70 # Surrogates shown in summary() ----------------------------
71 summary(cart2)
72
73 # Create missing values in first two rows
74 mtcars2 <- mtcars
75 mtcars2[1,6] <- NA   # first row, 6th col. wt is the 6th col.
76 mtcars2[2,6] <- NA
77 mtcars2[2,3] <- NA   # 3rd column is disp.
```

Check the NAs in rows 1 and 2

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	*NA*	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	*NA*	110	3.90	*NA*	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1

- Case 1 has one missing value in wt.
- Case 2 has two missing values:
  - wt
  - disp

```
79  cart3 <- rpart(mpg ~ ., data = mtcars2, method = 'anova', control = rpart.control(minsplit = 2, cp = 0))
80
81  summary(cart3)
```

```
Node number 1: 32 observations,    complexity param=0.6526612
  mean=20.09062, MSE=35.18897
  left son=2 (26 obs) right son=3 (6 obs)
  Primary splits:
      wt   < 2.26   to the right, improve=0.6709400, (2 missing)
      cyl  < 5      to the right, improve=0.6431252, (0 missing)
      disp < 153.35 to the right, improve=0.6305513, (1 missing)
      hp   < 118    to the right, improve=0.6010712, (0 missing)
      vs   < 0.5    to the left,  improve=0.4409477, (0 missing)
  Surrogate splits:
      disp < 101.55 to the right, agree=0.967, adj=0.833, (1 split)
      hp   < 92     to the right, agree=0.933, adj=0.667, (1 split)
      drat < 4      to the left,  agree=0.900, adj=0.500, (0 split)
      cyl  < 5      to the right, agree=0.833, adj=0.167, (0 split)
      am   < 0.5    to the left,  agree=0.833, adj=0.167, (0 split)
```

- Case 1 used disp as surrogate.
- Case 2 used hp as surrogate.

Enhanced version of CART: Random Forest

- Ensemble of 500 CARTs.
- More stable than 1 CART model.
- More accurate than 1 CART model.
- Will not overfit.
- Taught in BC2407 Analytics II.

Summary: Categorical Y vs Continuous Y

	Categorical Y	Continuous Y
rpart() parameter	method = ‘class’	method = ‘anova’
Model Prediction	Majority Category	Mean value of Y
Metric to determine Best Split	Gini index	SSE
Metric to evaluate model performance	Misclassification Error	MSE

Summary of CART

- Phrase 1: Grow Tree to max.
- Phrase 2: Prune Tree to min.
- Optimal Tree selection via 10 fold CV with 1 SE rule.
- Extract Decision Rules (i.e. Predictions) at terminal nodes.
- Variable Importance.
- CART handles missing values automatically via Surrogates.

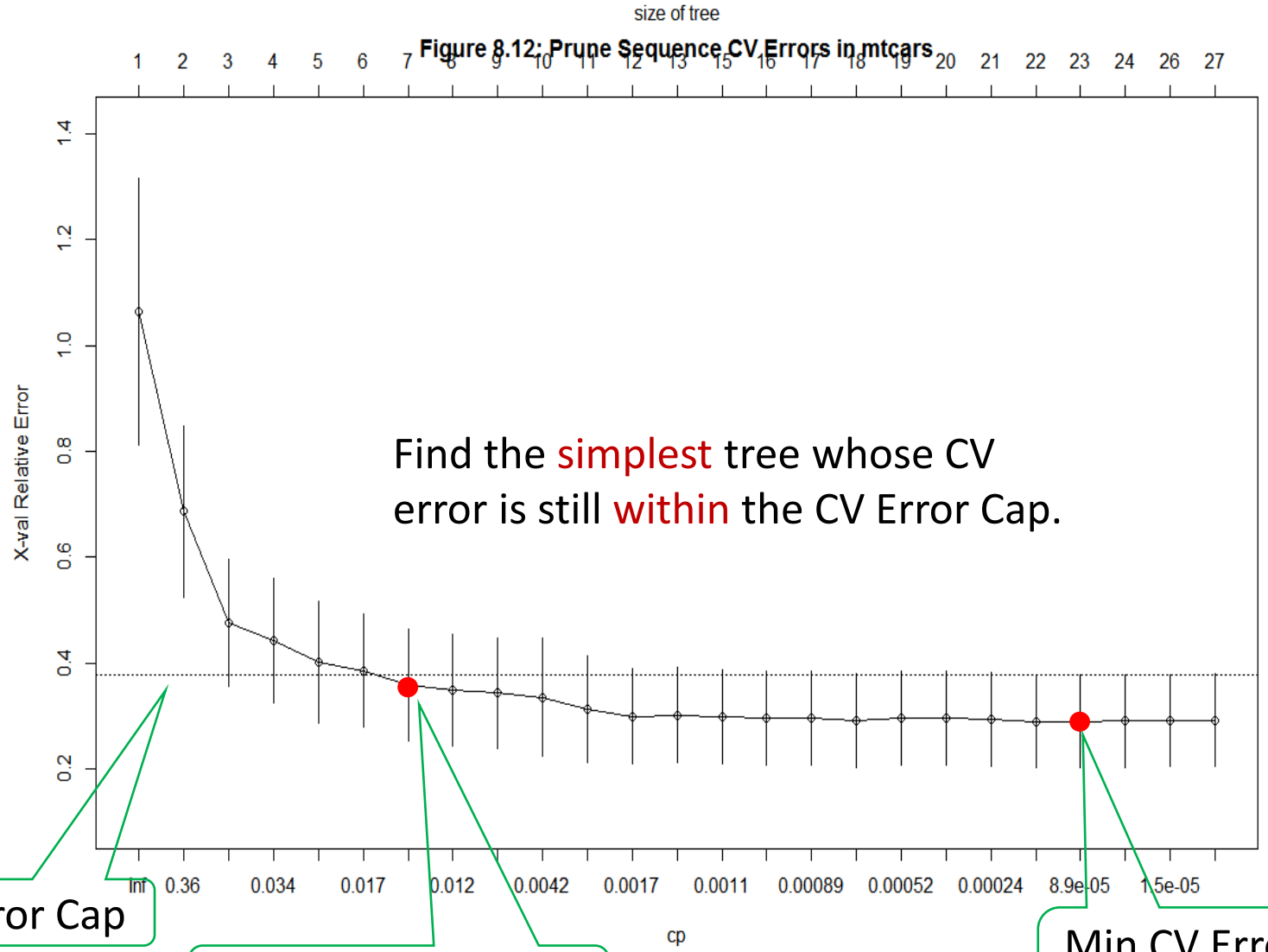
(Optional) Automating the Search for Optimal Tree

Rscript: mtcars CART.R

CART

Based on Chew C. H. (2020) textbook: AI, Analytics and Data Science. Vol 1., Chap 8.

CART on mtcars dataset shows 25 trees
[32 cases, 10 X variables, continuous Y variable]



Find the simplest tree whose CV error is still within the CV Error Cap.

CV Error Cap

Optimal Tree #7

Min CV Error Tree #22

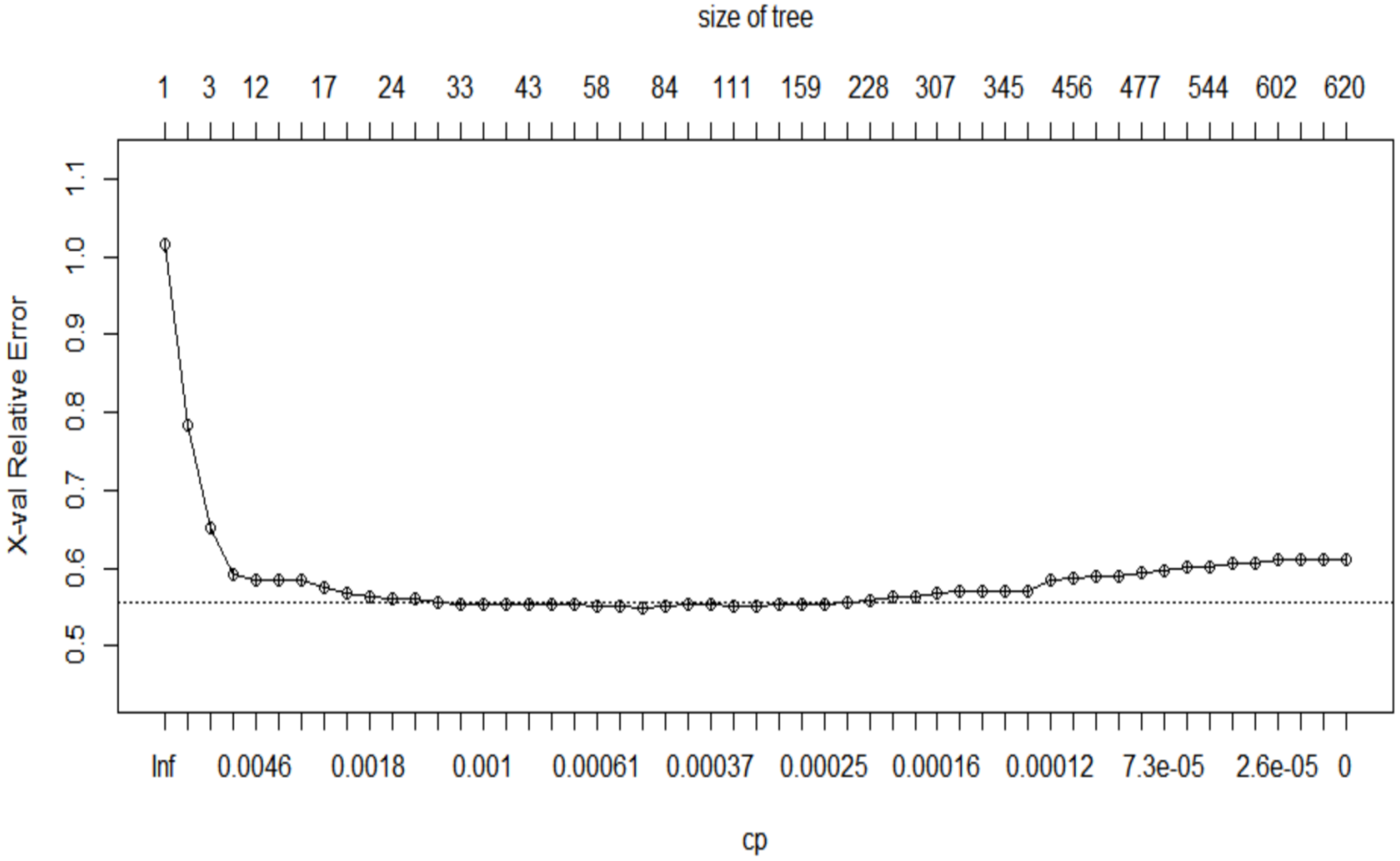
CP Table can be used to search for Optimal Tree too

```
Root node error: 1126/32 = 35.189

n= 32

           CP nsplit  rel error  xerror     xstd
1  6.5266e-01      0 1.0000e+00 1.06389 0.252198
2  1.9470e-01      1 3.4734e-01 0.68629 0.160870
3  4.5774e-02      2 1.5264e-01 0.47604 0.119551
4  2.5328e-02      3 1.0686e-01 0.44324 0.117846
5  2.3250e-02      4 8.1534e-02 0.40281 0.115329
6  1.2488e-02      5 5.8285e-02 0.38559 0.106955
7  1.2149e-02      6 4.5796e-02 0.35818
8  1.1647e-02      7 3.3648e-02 0.34943 0.105751
9  9.6700e-03      8 2.2000e-02 0.34357 0.104871
10 1.8010e-03      9 1.2330e-02 0.33605 0.112381
11 1.8010e-03     10 1.0529e-02 0.31304 0.099915
12 1.5156e-03     11 8.7282e-03 0.29965 0.090460
13 1.2868e-03     12 7.2125e-03 0.30216 0.090476
14 9.9907e-04     14 4.6389e-03 0.29853 0.089054
15 9.2506e-04     15 3.6399e-03 0.29704 0.089144
16 8.5254e-04     16 2.7148e-03 0.29628 0.089205
17 7.5041e-04     17 1.8623e-03 0.29221 0.089117
18 3.5967e-04     18 1.1119e-03 0.29642 0.088779
19 2.8418e-04     19 7.5219e-04 0.29642 0.088779
20 2.0011e-04     20 4.6801e-04 0.29479 0.088862
21 1.1101e-04     21 2.6790e-04 0.29055 0.086900
22 7.1045e-05     22 1.5689e-04 0.29013 0.086937
23 3.9963e-05     23 8.5846e-05 0.29080 0.086896
24 5.9204e-06     25 5.9204e-06 0.29159 0.086831
25 0.0000e+00     26 0.0000e+00 0.29237 0.087231
```

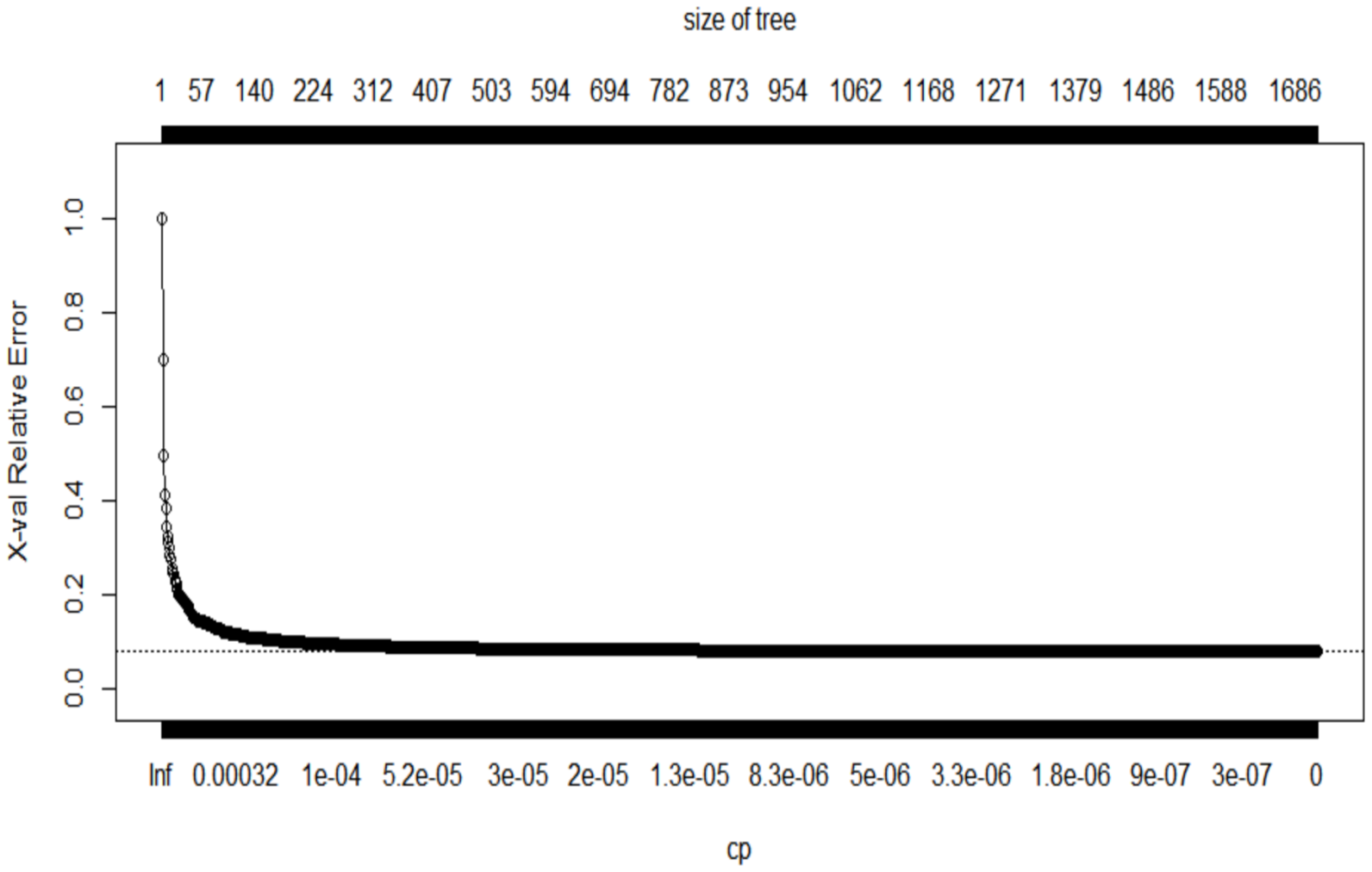
```
> 0.29013 + 0.086937
[1] 0.377067
```

Min CV Error Tree

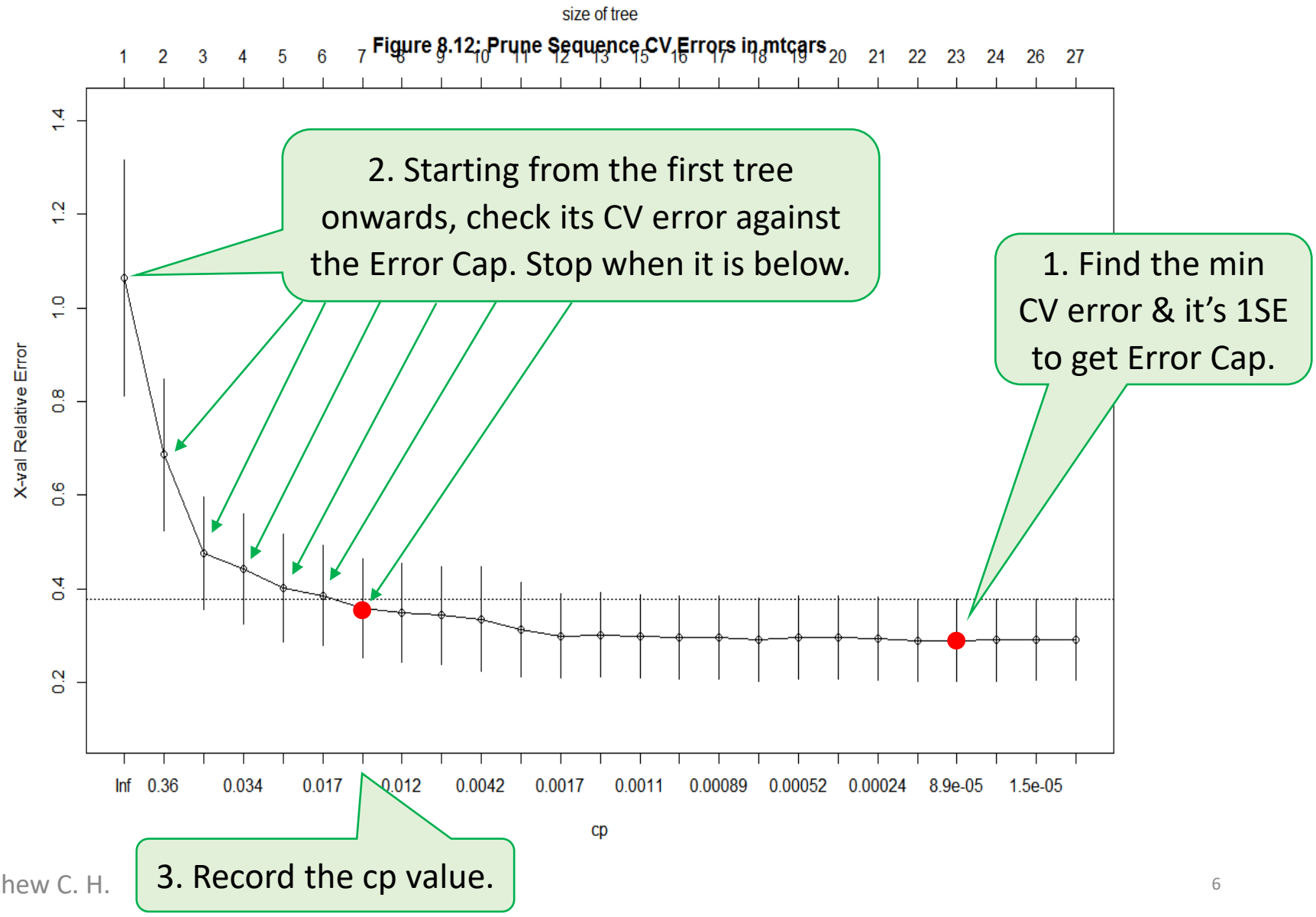
Census data sample: 17,296 rows. Categorical Y. Easy to find optimal tree?



HDB Resale Flat data: 22,204 cases, Continuous Y. Easy to find optimal tree?



How did we identify the optimal tree's CP value?



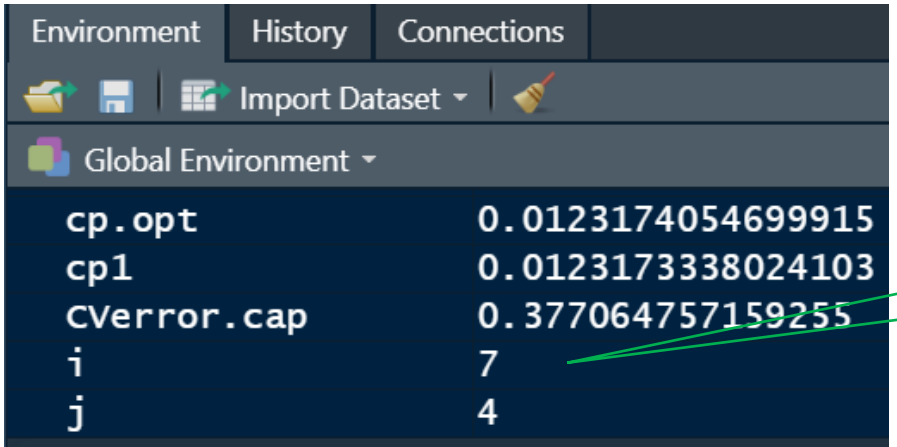
Rscript to Automate the Search for Optimal Tree

2. Starting from the first tree onwards, check its CV error against the Error Cap. Stop when it is below.

1. Find the min CV error & it's 1SE to get Error Cap.

```
35  # [Optional] Extract the Optimal Tree via code instead of eye power ----------
36  # Compute min CVerror + 1SE in maximal tree cart1.
37  CVerror.cap <- cart1$cptable[which.min(cart1$cptable[,"xerror"]), "xerror"] +
38    cart1$cptable[which.min(cart1$cptable[,"xerror"]), "xstd"]
39
40  # Find the optimal CP region whose CV error is just below CVerror.cap in maximal tree cart1.
41  i <- 1; j<- 4
42  while (cart1$cptable[i,j] > CVerror.cap) {
43    i <- i + 1
44  }
45
46  # Get geometric mean of the two identified CP values in the optimal region
47  # if optimal tree has at least one split.
48  cp.opt = ifelse(i > 1, sqrt(cart1$cptable[i,1] * cart1$cptable[i-1,1]), 1)
49  # ------------------------------------------------------------------------------
```

3. Record the cp value.



The 7th tree is optimal.

Important: Remember to adjust the Rscript to use the name of your maximal CART model

```
35  # [Optional] Extract the Optimal Tree via code instead of eye power ------------
36  # Compute min CVerror + 1SE in maximal tree cart1.
37  CVerror.cap <- cart1$cptable[which.min(cart1$cptable[,"xerror"]), "xerror"] +
38    cart1$cptable[which.min(cart1$cptable[,"xerror"]), "xstd"]
39
40  # Find the optimal CP region whose CV error is just below CVerror.cap in maximal tree cart1.
41  i <- 1; j<- 4
42  while (cart1$cptable[i,j] > CVerror.cap) {
43    i <- i + 1
44  }
45
46  # Get geometric mean of the two identified CP values in the optimal region
47  # if optimal tree has at least one split.
48  cp.opt = ifelse(i > 1, sqrt(cart1$cptable[i,1] * cart1$cptable[i-1,1]), 1)
49  # ---------------------------------------------------------------------------
```

- The maximal tree in this script was named cart1.
- Replace it with the name of your maximal tree.
- Select the 3 blocks of code and use <CTRL> + <F> on windows machines to find and replace all occurrence of cart1.

Advice on using this automation

- Use plotcp() chart if there are not too many trees and the optimal cp can be obviously identified from the chart.
- Else, use my R script to extract the cp value of the optimal tree.
- Remember to correct the name of the maximal tree before use.