GBM On mtcars Dataset

Greg Sollenberger

27 October, 2020

```
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.1
## v tidyr   1.1.1     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
mtcars
```

```
##                     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
## Duster 360         14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D          24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230           22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280           19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C          17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SE         16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL         17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC        15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood 10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4  8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial  14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Fiat 128           32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic        30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
```

```
## Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

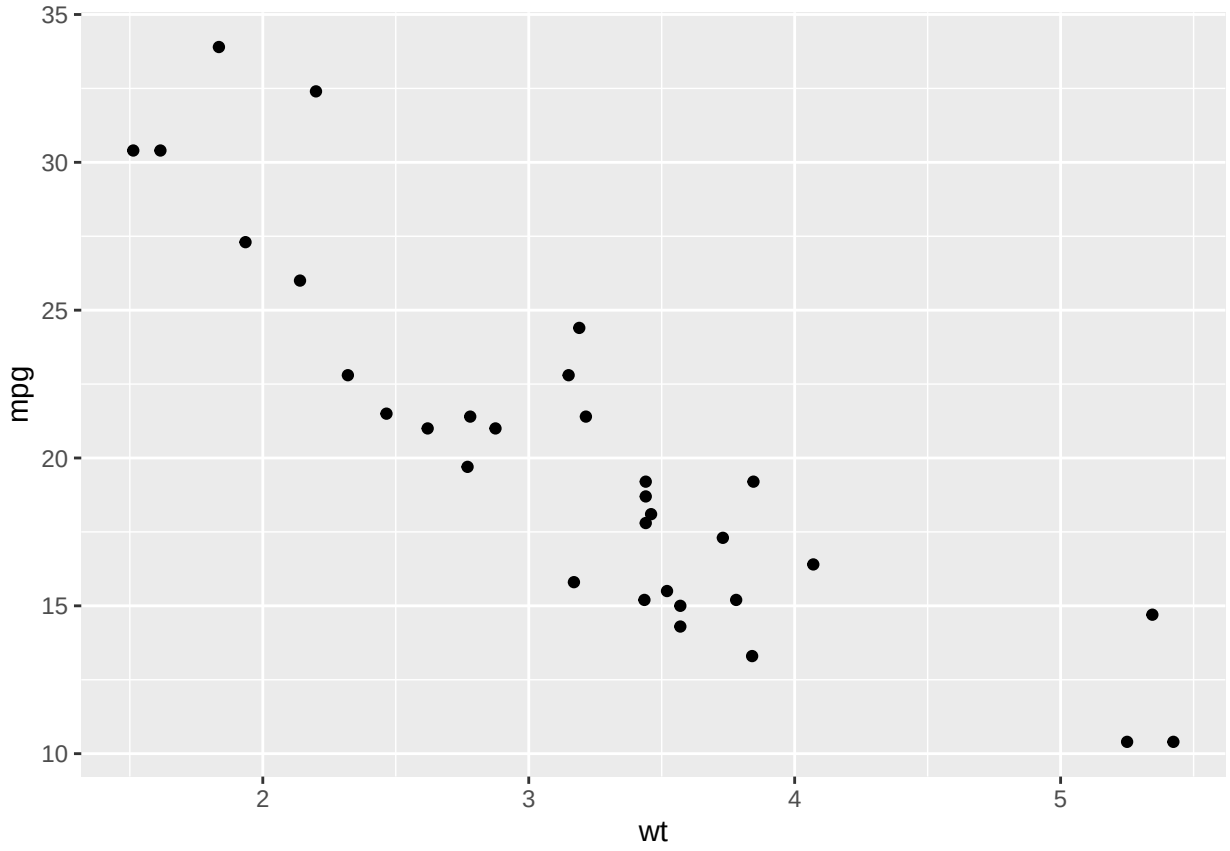
```
glimpse(mtcars)
```

```
## Rows: 32
## Columns: 11
## $ mpg  <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19.2, 17...
## $ cyl  <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4, 4, 4, 4,...
## $ disp <dbl> 160.0, 160.0, 108.0, 258.0, 360.0, 225.0, 360.0, 146.7, 140.8,...
## $ hp   <dbl> 110, 110, 93, 110, 175, 105, 245, 62, 95, 123, 123, 180, 180, ...
## $ drat <dbl> 3.90, 3.90, 3.85, 3.08, 3.15, 2.76, 3.21, 3.69, 3.92, 3.92, 3....
## $ wt   <dbl> 2.620, 2.875, 2.320, 3.215, 3.440, 3.460, 3.570, 3.190, 3.150,...
## $ qsec <dbl> 16.46, 17.02, 18.61, 19.44, 17.02, 20.22, 15.84, 20.00, 22.90,...
## $ vs   <dbl> 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,...
## $ am   <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0,...
## $ gear <dbl> 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3,...
## $ carb <dbl> 4, 4, 1, 1, 2, 1, 4, 2, 2, 4, 4, 3, 3, 3, 4, 4, 4, 1, 2, 1, 1,...
```

```
summary(mtcars)
```

```
##       mpg             cyl             disp             hp
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0
##       drat             wt             qsec             vs
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000
##  Median :3.695   Median :3.325   Median :17.71   Median :0.0000
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000
##  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000
##        am              gear            carb
##  Min.   :0.0000   Min.   :3.000   Min.   :1.000
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000
##  Median :0.0000   Median :4.000   Median :2.000
##  Mean   :0.4062   Mean   :3.688   Mean   :2.812
##  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000
##  Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

```
ggplot(mtcars,aes(x = wt, y = mpg)) + geom_point()
```



Next we build a gbm with 300 trees and examine some of the output.

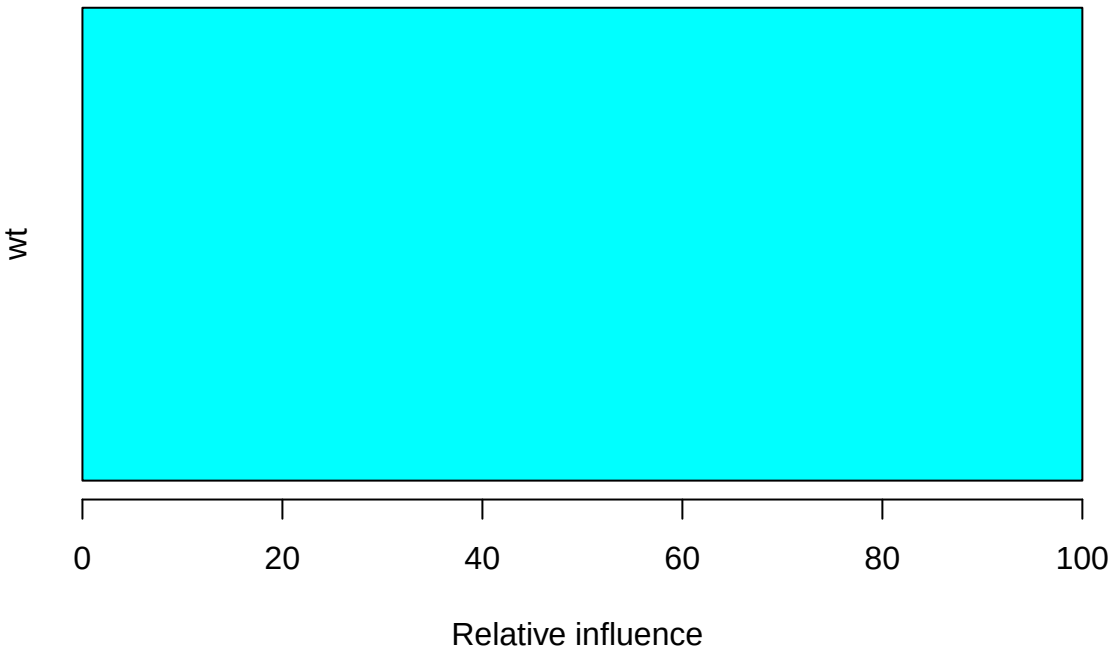
```
gbm1 <- gbm(mpg~wt, #formula input. mpg as a function of wt
            data = mtcars, #data input
            distribution = "gaussian", #the distribution/error function
            n.trees = 300, #How many trees?
            shrinkage = 0.02, #shrinkage factor, aka learning rate
            interaction.depth = 1, #interaction depth, aka number of terminal nodes - 1
            bag.fraction = 0.5, #fraction of data to use at each iteration
            n.minobsinnode=0, #minimum observations at each terminal node - small dataset so I set to z
            train.fraction = 0.7, #use 70%/30% train/test split
            verbose = TRUE) #verbose = TRUE, so progress gets output to the terminal
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1       36.2548         30.3817     0.0200    0.9960
##      2       35.2908         29.6142     0.0200    0.8338
##      3       34.6601         29.1170     0.0200    0.7765
##      4       33.6973         27.8036     0.0200    0.6074
##      5       32.9055         26.7177     0.0200    1.0607
##      6       31.9744         25.5873     0.0200    0.3156
##      7       31.0074         24.3903     0.0200    0.1867
##      8       30.6163         24.2138     0.0200    0.4177
##      9       30.0546         23.0949     0.0200    0.4452
```

```
##      10       29.3833        22.5963     0.0200    0.7257
##      20       22.9338        15.8696     0.0200    0.1721
##      40       14.9894         9.5997     0.0200   -0.0248
##      60        9.9754         7.6099     0.0200    0.1419
##      80        7.0286         7.1630     0.0200    0.0518
##     100        5.0802         7.4169     0.0200   -0.0122
##     120        4.0473         7.8310     0.0200    0.0053
##     140        3.2964         8.4808     0.0200    0.0298
##     160        2.8039         9.2930     0.0200    0.0151
##     180        2.5410         9.9416     0.0200   -0.0439
##     200        2.3394        10.9113     0.0200   -0.0111
##     220        2.2128        11.2548     0.0200   -0.0240
##     240        2.1095        11.7908     0.0200   -0.0073
##     260        2.0257        12.3474     0.0200   -0.0241
##     280        1.9607        12.7578     0.0200   -0.0116
##     300        1.9209        13.0201     0.0200   -0.0034
```

Take a look at the model's performance. This should tell us approximately how many trees to use. The OOB method is telling us to use ~140 trees, the test method is telling us to use ~75.

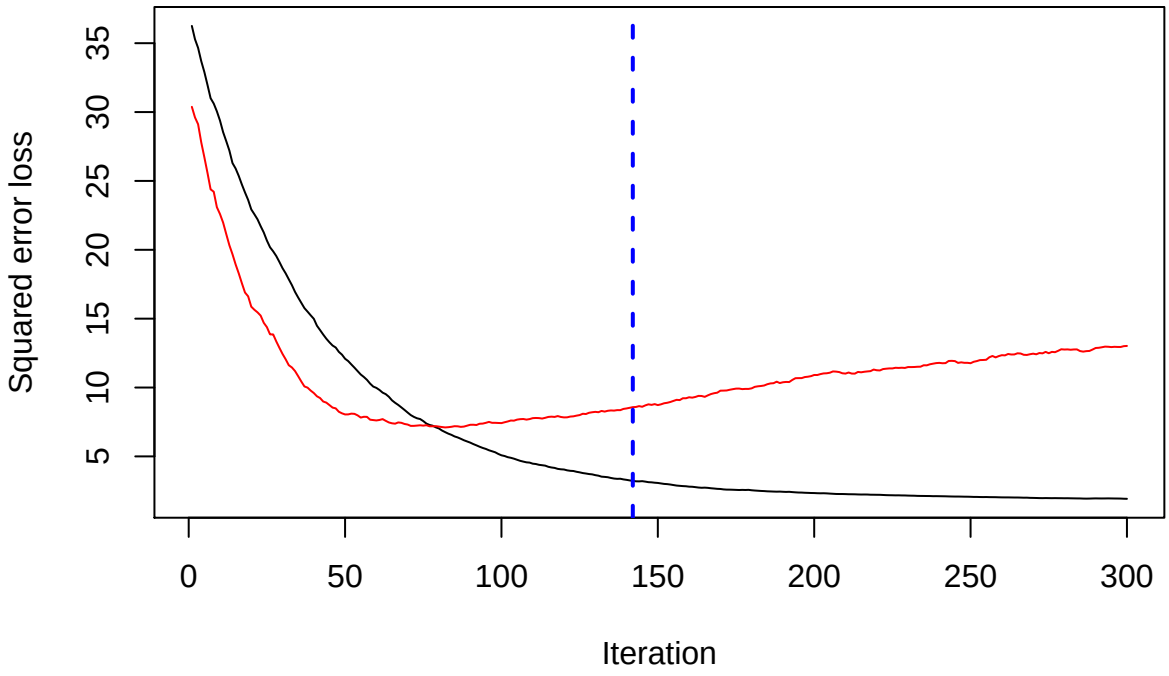
```
summary(gbm1)
```




```
##    var rel.inf
## wt  wt     100
```

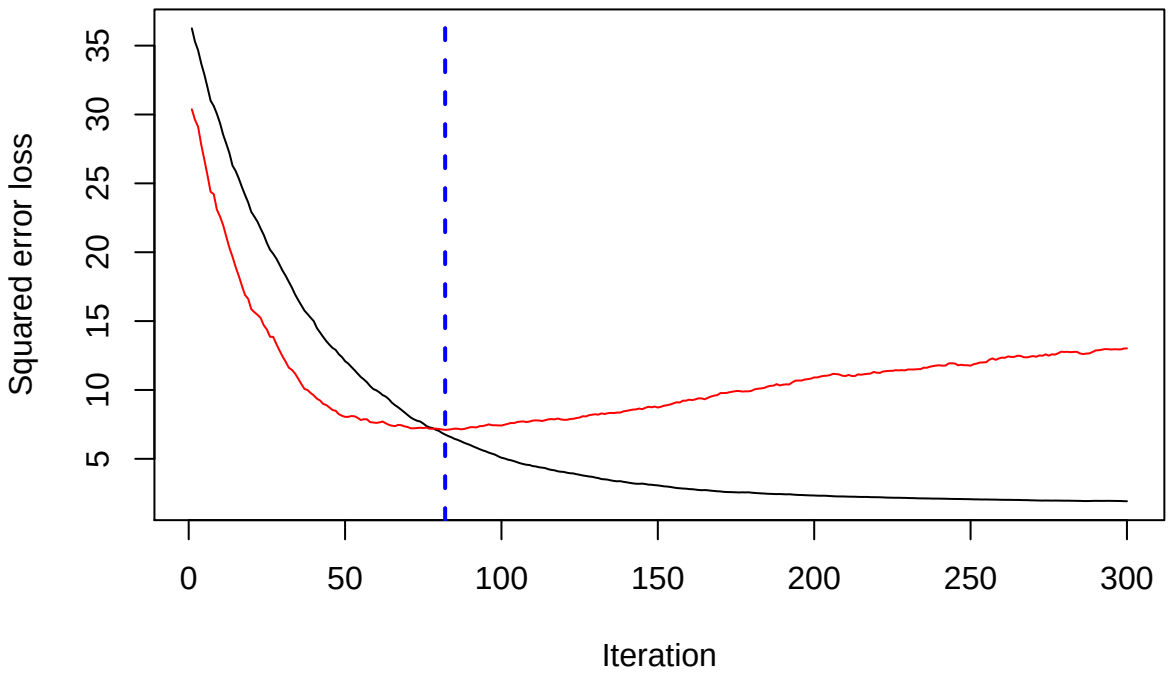
```r
gbm.perf(gbm1,
         method = "OOB")
```

```
## OOB generally underestimates the optimal number of iterations although predictive performance is reas
```




```
## [1] 142
## attr(,"smoother")
## Call:
## loess(formula = object$oobag.improve ~ x, enp.target = min(max(4,
##     length(x)/10), 50))
##
## Number of Observations: 300
## Equivalent Number of Parameters: 25.36
## Residual Standard Error: 0.08482
```

```r
gbm.perf(gbm1,
         method = "test")
```

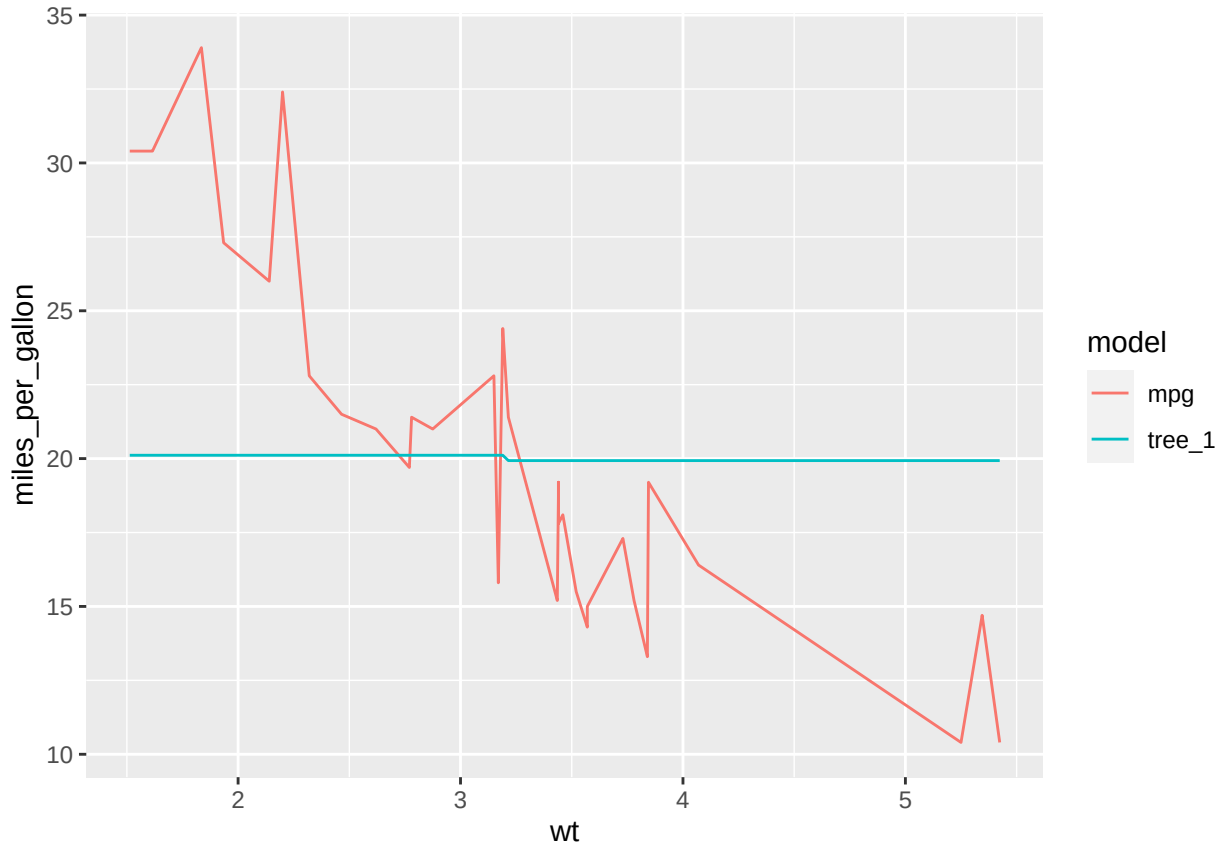


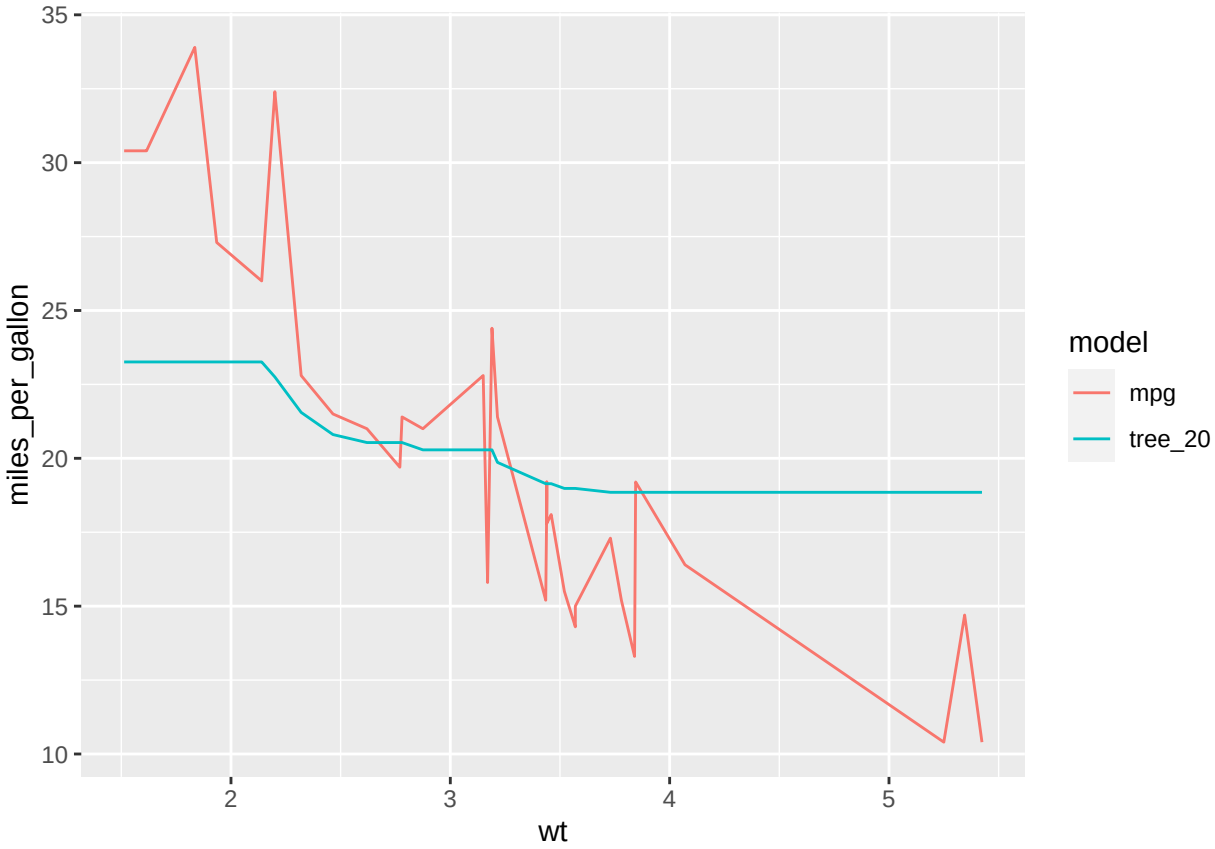

```
## [1] 82
```

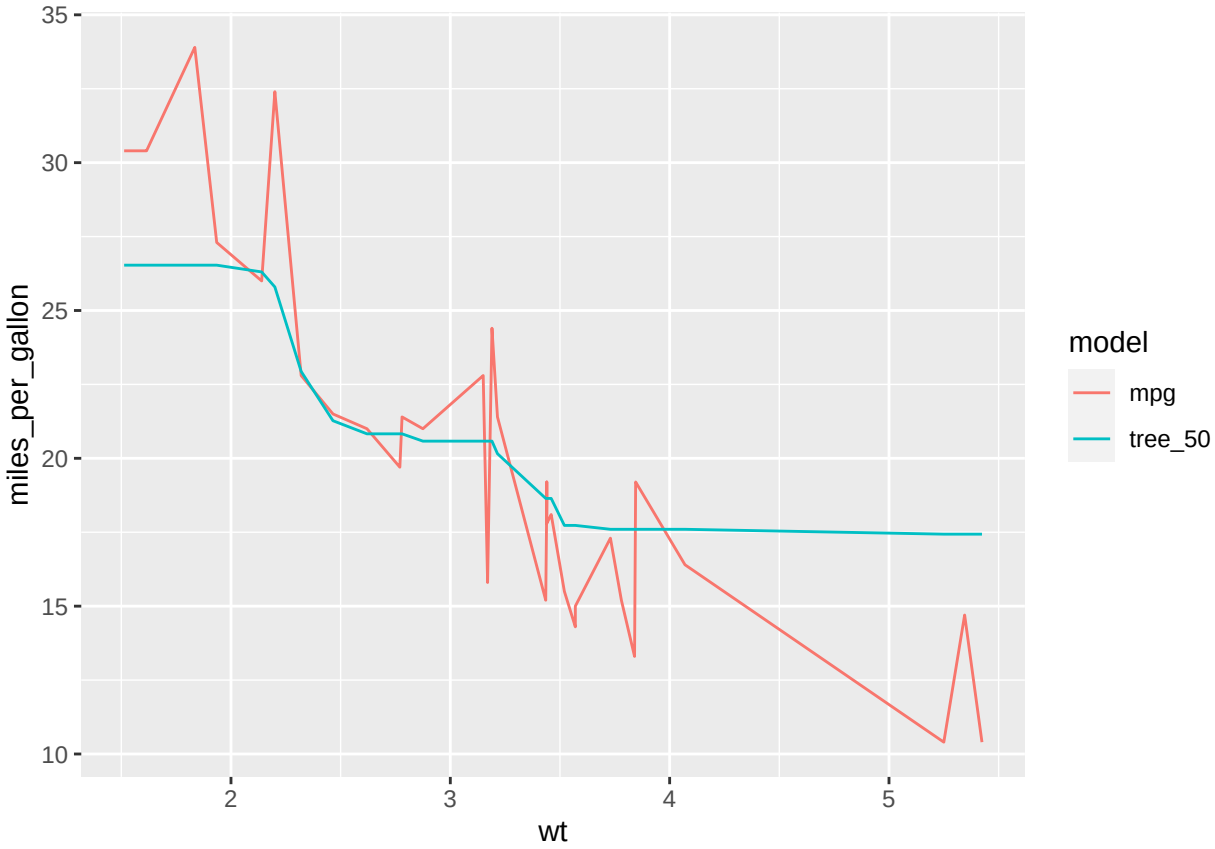
Next take a look at some plots showing how well the model matches the observations. Note how larger numbers of trees begin to over-fit and start "chasing" random noise.

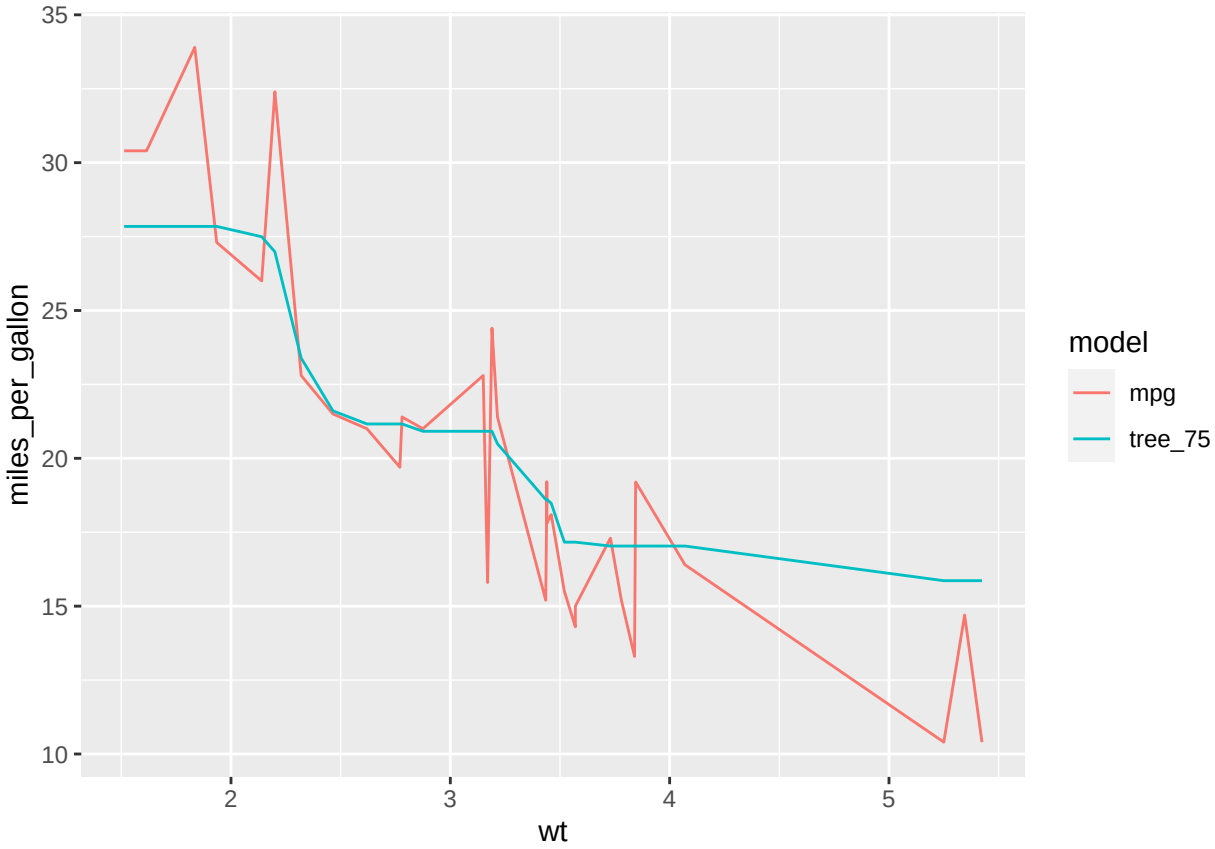
```
mtcars_copy <- mtcars
for(i in 1:300 ) {
  column_name <- paste("tree_",i,sep = "")
  mtcars_copy[,column_name] <- predict(gbm1,mtcars,n.trees = i,type = "response")
  #print(column_name)
  #print(mtcars_copy[1:3,column_name])
  #ggplot(mtcars,aes_string(x="wt",y=column_name)) + geom_point()
  #Sys.sleep(2)
}

for(i in c(1,20,50, 75, 100, 150, 200, 250)) {
  column_name <- paste("tree_",i,sep = "")
  print(mtcars_copy %>%
          select(wt,mpg,tree_1:tree_300) %>%
          gather(key = "model",
                 value = "miles_per_gallon",
                 c("mpg",column_name)) %>%
    ggplot(aes(x = wt, y = miles_per_gallon, col = model)) +
      geom_line())
}
```

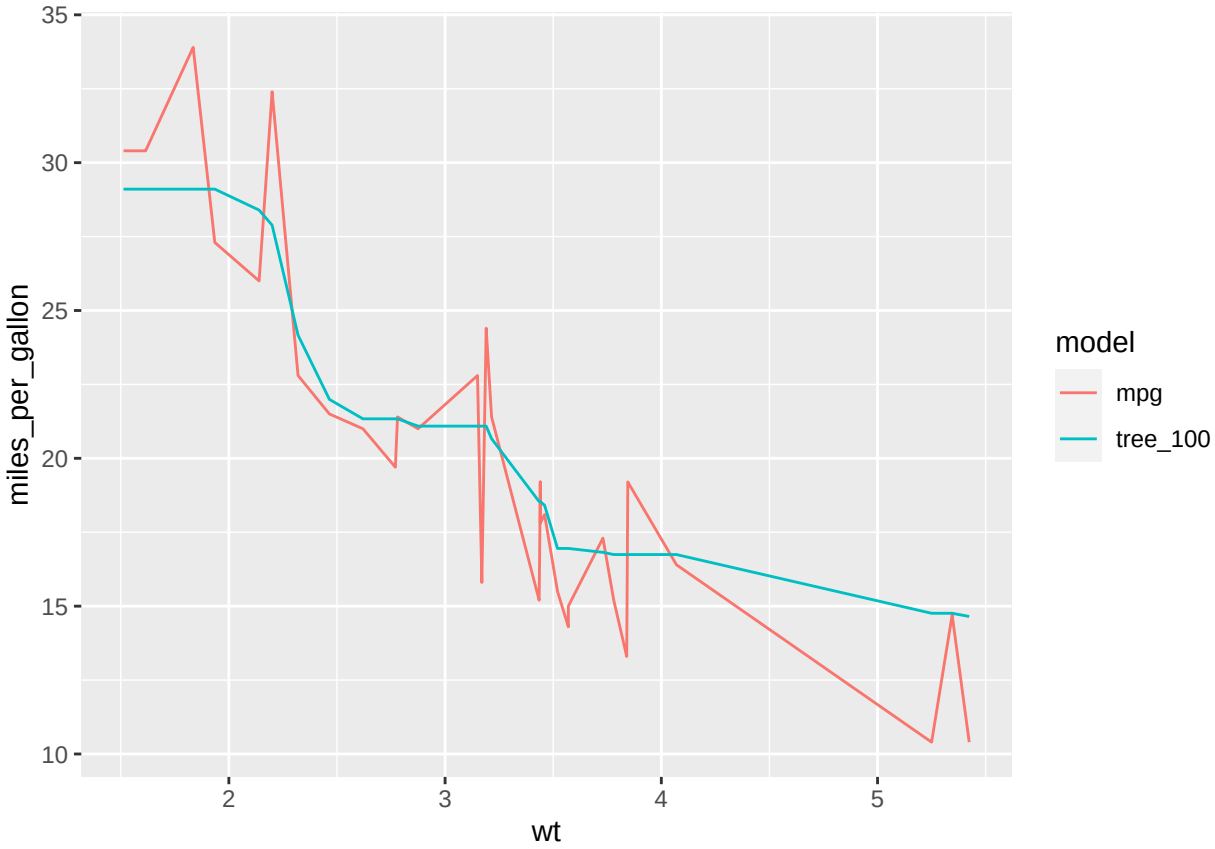
```
## Note: Using an external vector in selections is ambiguous.
## i Use ‘all_of(column_name)‘ instead of ‘column_name‘ to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

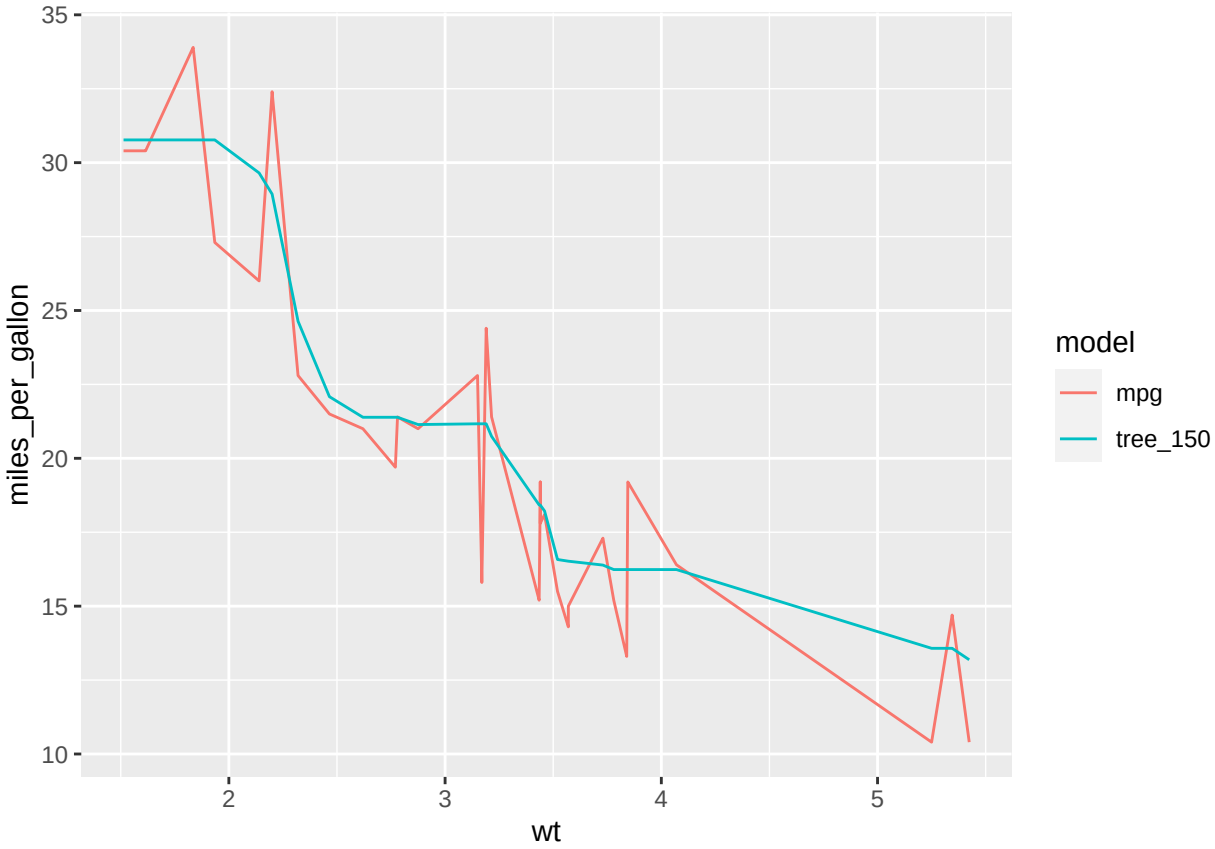


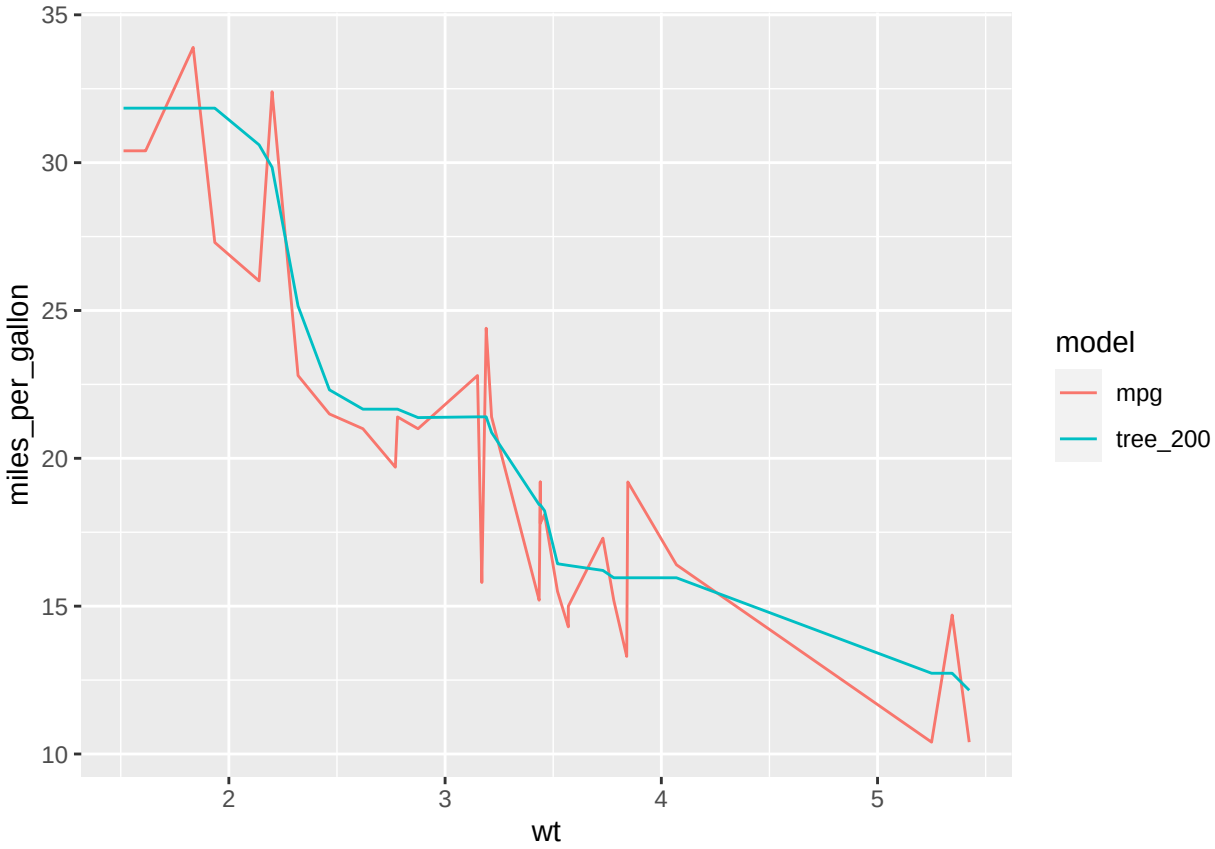

35
30
25
20
15
10
miles_per_gallon
2
3
4
5
wt
model
mpg
tree_20

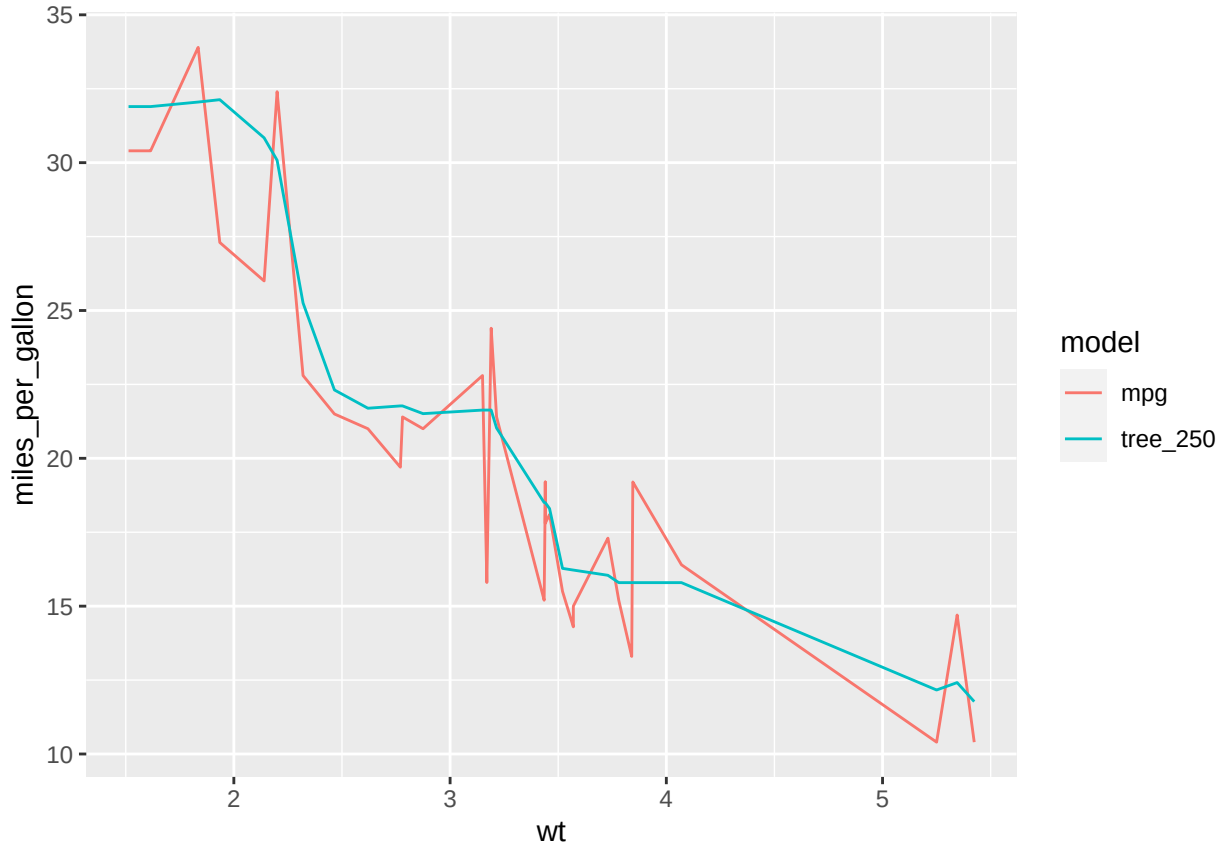

35
30
25
20
15
10
miles_per_gallon
2
3
4
5
wt
model
mpg
tree_50


35
30
25
20
15
10
miles_per_gallon
2
3
4
5
wt
model
mpg
tree_75


35
30
25
20
15
10
miles_per_gallon
2
3
4
5
wt
model
mpg
tree_100


35
30
25
20
15
10
miles_per_gallon
2
3
4
5
wt
model
mpg
tree_150


35
30
25
20
15
10
miles_per_gallon
2
3
4
5
wt
model
mpg
tree_200



Look at the partial dependence plot for weight. Note that we can return a table of values. This suggests a way to implement a gbm as a rating table, the same way we do for glms. In both cases, a factor table is a simple translation of the model's true form into something more traditional to insurance pricing.

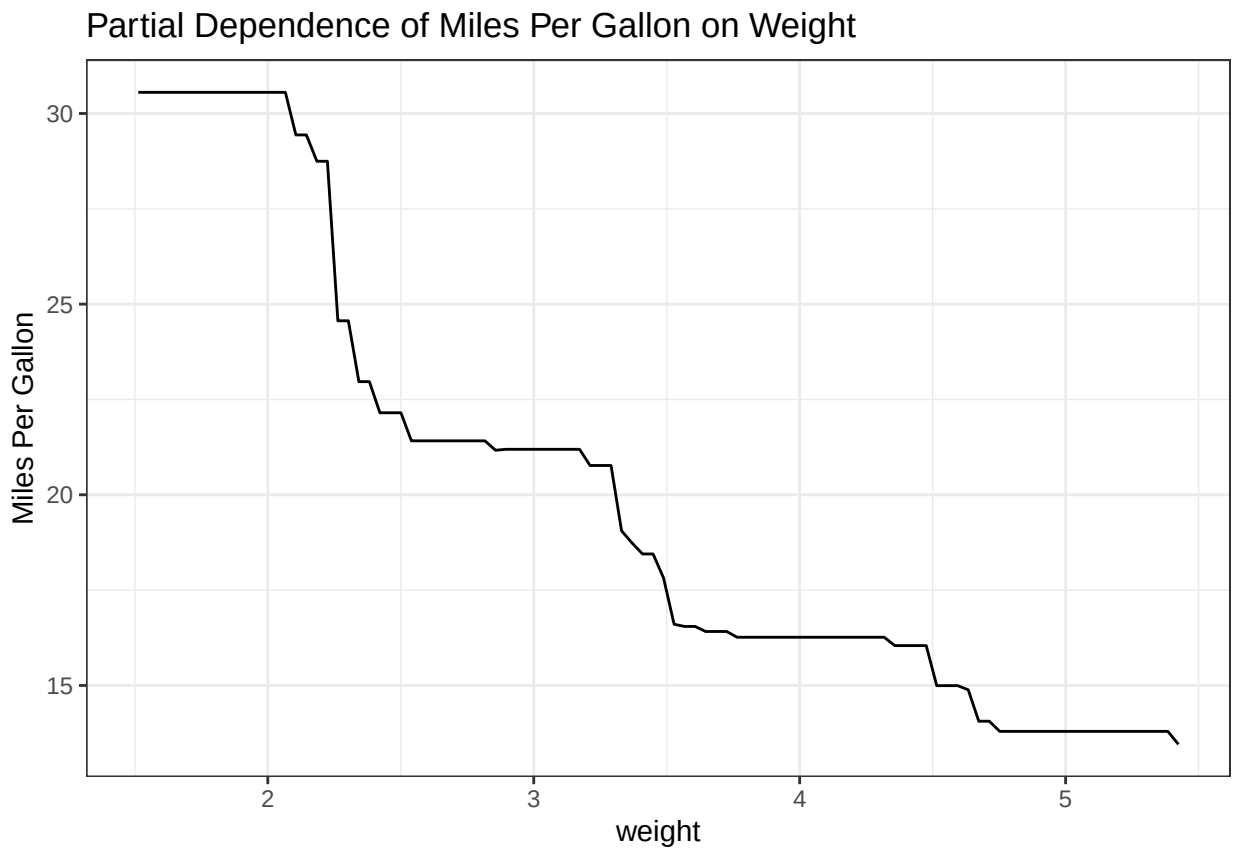
```
# plot.gbm(gbm1,
#          i.var = 1,
#          n.trees = 140)

partial_table <-
plot.gbm(gbm1,
         i.var = 1,
         n.trees = 140,
         return.grid = TRUE) %>%
  rename(mileage = y)

partial_table
```

```
##           wt  mileage
## 1   1.513000 30.55468
## 2   1.552505 30.55468
## 3   1.592010 30.55468
## 4   1.631515 30.55468
## 5   1.671020 30.55468
## 6   1.710525 30.55468
## 7   1.750030 30.55468
## 8   1.789535 30.55468
```

```
## 9   1.829040 30.55468
## 10  1.868545 30.55468
## 11  1.908051 30.55468
## 12  1.947556 30.55468
## 13  1.987061 30.55468
## 14  2.026566 30.55468
## 15  2.066071 30.55468
## 16  2.105576 29.43764
## 17  2.145081 29.43764
## 18  2.184586 28.74655
## 19  2.224091 28.74655
## 20  2.263596 24.56332
## 21  2.303101 24.56332
## 22  2.342606 22.96746
## 23  2.382111 22.96746
## 24  2.421616 22.15116
## 25  2.461121 22.15116
## 26  2.500626 22.15116
## 27  2.540131 21.41454
## 28  2.579636 21.41454
## 29  2.619141 21.41454
## 30  2.658646 21.41454
## 31  2.698152 21.41454
## 32  2.737657 21.41454
## 33  2.777162 21.41454
## 34  2.816667 21.41454
## 35  2.856172 21.16773
## 36  2.895677 21.19308
## 37  2.935182 21.19308
## 38  2.974687 21.19308
## 39  3.014192 21.19308
## 40  3.053697 21.19308
## 41  3.093202 21.19308
## 42  3.132707 21.19308
## 43  3.172212 21.19308
## 44  3.211717 20.76869
## 45  3.251222 20.76869
## 46  3.290727 20.76869
## 47  3.330232 19.05495
## 48  3.369737 18.73597
## 49  3.409242 18.44806
## 50  3.448747 18.44806
## 51  3.488253 17.81606
## 52  3.527758 16.60386
## 53  3.567263 16.54714
## 54  3.606768 16.54714
## 55  3.646273 16.41539
## 56  3.685778 16.41539
## 57  3.725283 16.41539
## 58  3.764788 16.26398
## 59  3.804293 16.26398
## 60  3.843798 16.26398
## 61  3.883303 16.26398
## 62  3.922808 16.26398
```

```
## 63  3.962313 16.26398
## 64  4.001818 16.26398
## 65  4.041323 16.26398
## 66  4.080828 16.26398
## 67  4.120333 16.26398
## 68  4.159838 16.26398
## 69  4.199343 16.26398
## 70  4.238848 16.26398
## 71  4.278354 16.26398
## 72  4.317859 16.26398
## 73  4.357364 16.04521
## 74  4.396869 16.04521
## 75  4.436374 16.04521
## 76  4.475879 16.04521
## 77  4.515384 14.99480
## 78  4.554889 14.99480
## 79  4.594394 14.99480
## 80  4.633899 14.88609
## 81  4.673404 14.06182
## 82  4.712909 14.06182
## 83  4.752414 13.79520
## 84  4.791919 13.79520
## 85  4.831424 13.79520
## 86  4.870929 13.79520
## 87  4.910434 13.79520
## 88  4.949939 13.79520
## 89  4.989444 13.79520
## 90  5.028949 13.79520
## 91  5.068455 13.79520
## 92  5.107960 13.79520
## 93  5.147465 13.79520
## 94  5.186970 13.79520
## 95  5.226475 13.79520
## 96  5.265980 13.79520
## 97  5.305485 13.79520
## 98  5.344990 13.79520
## 99  5.384495 13.79520
## 100 5.424000 13.45670
```

```
partial_table %>%
  ggplot(aes(x = wt, y = mileage)) +
  geom_line() +
  theme_bw() +
  labs(x = "weight",
       y = "Miles Per Gallon",
       title = "Partial Dependence of Miles Per Gallon on Weight")
```



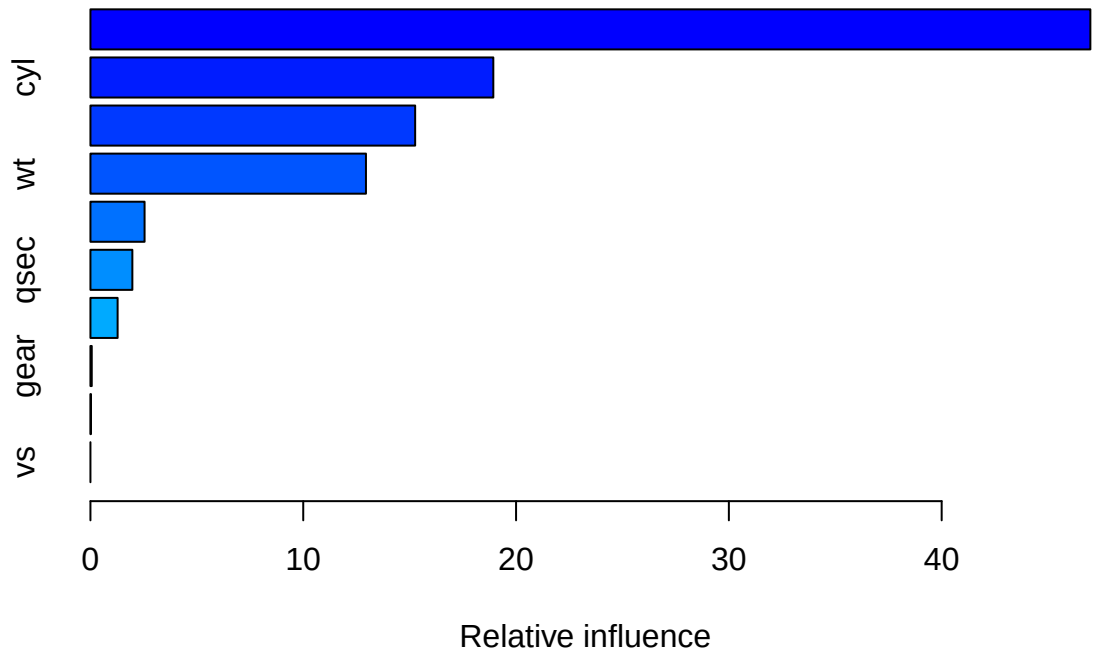
I was looking at just a single variable for illustrative purposes. Obviously we'd want to use more variables for a real model. In this case we're including a lot more variables and deeper interaction depth.

```
gbm2 <- gbm(mpg~.,
            data = mtcars,
            distribution = "gaussian",
            n.trees = 1000,
            shrinkage = 0.02,
            train.fraction = 0.7,
            bag.fraction = 0.5,
            n.minobsinnode = 0,
            interaction.depth = 3,
            verbose = TRUE)
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1       36.2650         30.5626     0.0200    1.0185
##      2       35.0286         29.8025     0.0200    1.1575
##      3       33.8797         28.5792     0.0200    0.4867
##      4       32.5441         27.5176     0.0200    1.3250
##      5       31.1710         26.7010     0.0200    1.4949
##      6       30.2940         25.9952     0.0200    0.7755
##      7       29.3677         25.2726     0.0200    1.2370
##      8       28.2511         24.4056     0.0200    1.0336
##      9       27.1951         23.5830     0.0200    0.9833
##     10       26.1232         23.0027     0.0200    0.7561
##     20       18.6828         18.4839     0.0200    0.2477
```

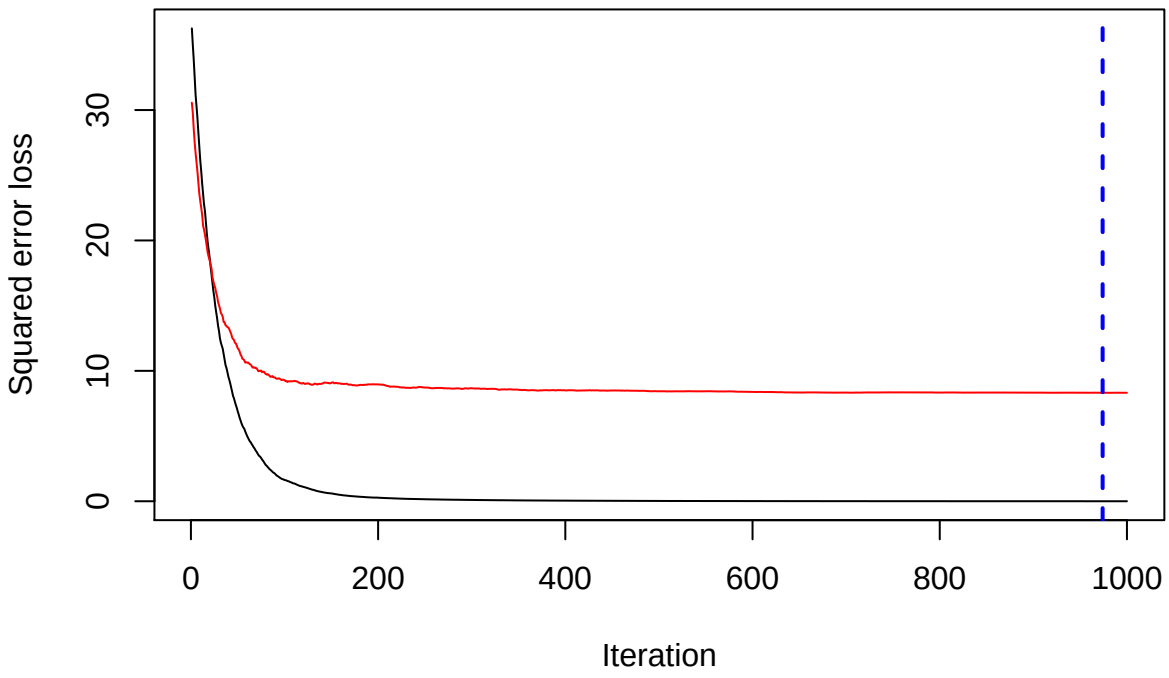
```
##     40        9.5540         13.3226     0.0200    0.3030
##     60        5.0109         10.6257     0.0200    0.1456
##     80        2.7744          9.7537     0.0200    0.0747
##    100        1.6291          9.2835     0.0200    0.0264
##    120        1.1084          9.0773     0.0200   -0.0112
##    140        0.7049          9.0194     0.0200    0.0101
##    160        0.4854          9.0140     0.0200   -0.0009
##    180        0.3464          8.9088     0.0200   -0.0022
##    200        0.2739          8.9514     0.0200    0.0024
##    220        0.2118          8.7782     0.0200   -0.0022
##    240        0.1707          8.7095     0.0200   -0.0016
##    260        0.1405          8.6777     0.0200   -0.0031
##    280        0.1179          8.6407     0.0200   -0.0026
##    300        0.1005          8.6680     0.0200   -0.0020
##    320        0.0849          8.6252     0.0200   -0.0007
##    340        0.0734          8.5858     0.0200   -0.0002
##    360        0.0618          8.5119     0.0200   -0.0012
##    380        0.0545          8.5155     0.0200   -0.0008
##    400        0.0476          8.5017     0.0200   -0.0005
##    420        0.0414          8.4957     0.0200   -0.0000
##    440        0.0364          8.4930     0.0200   -0.0009
##    460        0.0316          8.4823     0.0200   -0.0004
##    480        0.0277          8.4675     0.0200   -0.0005
##    500        0.0238          8.4348     0.0200   -0.0002
##    520        0.0204          8.4321     0.0200    0.0000
##    540        0.0183          8.4281     0.0200   -0.0003
##    560        0.0165          8.4263     0.0200   -0.0002
##    580        0.0146          8.4158     0.0200   -0.0004
##    600        0.0125          8.3837     0.0200   -0.0002
##    620        0.0108          8.3759     0.0200   -0.0001
##    640        0.0092          8.3445     0.0200   -0.0001
##    660        0.0083          8.3460     0.0200   -0.0001
##    680        0.0073          8.3313     0.0200   -0.0001
##    700        0.0066          8.3344     0.0200   -0.0001
##    720        0.0057          8.3402     0.0200   -0.0001
##    740        0.0053          8.3453     0.0200   -0.0002
##    760        0.0046          8.3499     0.0200   -0.0000
##    780        0.0041          8.3439     0.0200   -0.0000
##    800        0.0037          8.3350     0.0200   -0.0000
##    820        0.0032          8.3349     0.0200   -0.0000
##    840        0.0029          8.3346     0.0200   -0.0000
##    860        0.0026          8.3343     0.0200   -0.0000
##    880        0.0023          8.3330     0.0200   -0.0001
##    900        0.0020          8.3284     0.0200   -0.0000
##    920        0.0018          8.3214     0.0200   -0.0000
##    940        0.0016          8.3254     0.0200   -0.0000
##    960        0.0014          8.3209     0.0200   -0.0000
##    980        0.0012          8.3175     0.0200   -0.0000
##   1000        0.0011          8.3204     0.0200   -0.0000
```

```
summary(gbm2)
```




```
##       var     rel.inf
## disp disp 46.98696153
## cyl   cyl 18.93126277
## hp     hp 15.25859213
## wt     wt 12.94714820
## drat drat  2.54409373
## qsec qsec  1.96926151
## carb carb  1.27667209
## gear gear  0.06159796
## am     am  0.02441009
## vs     vs  0.00000000
```

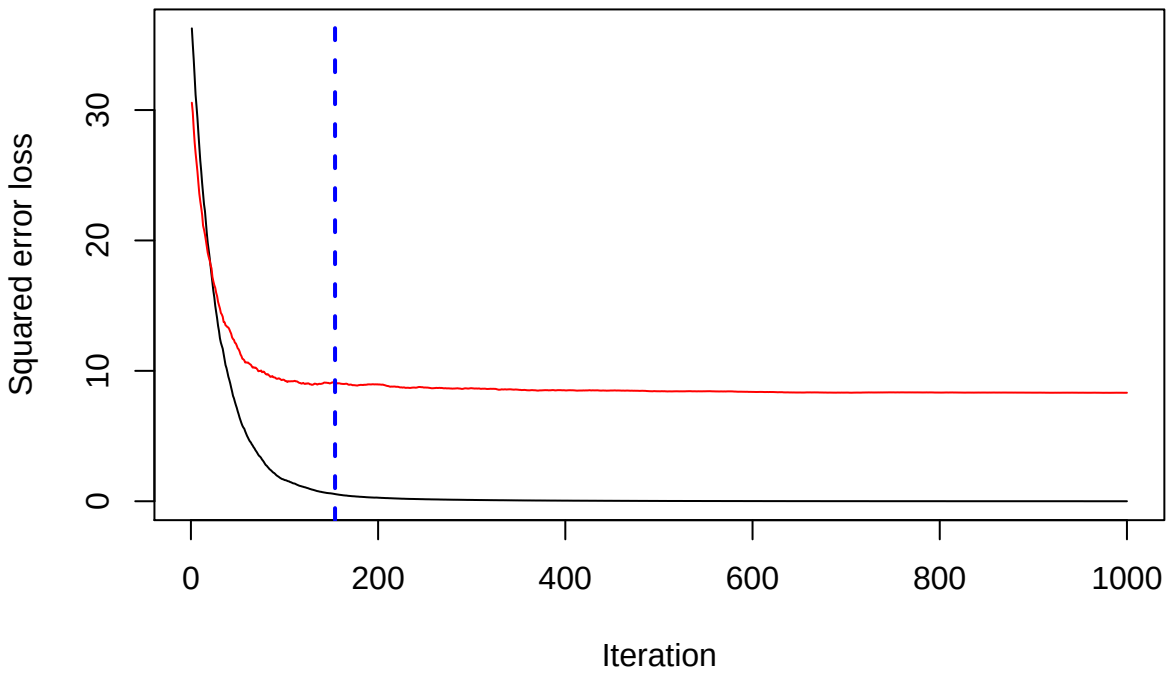
```
gbm.perf(gbm2,
         method = "test")
```




```
## [1] 974
```

```
gbm.perf(gbm2,
         method = "OOB")
```

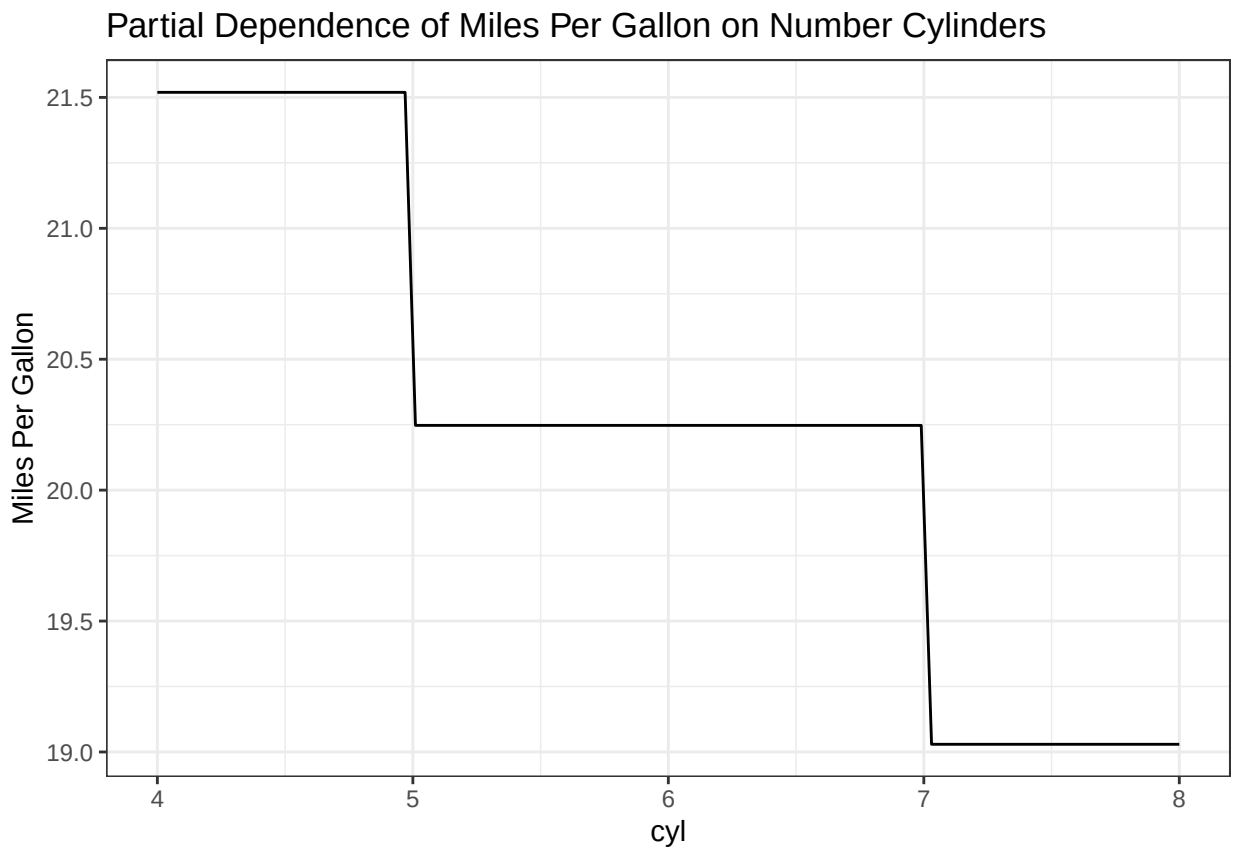
```
## OOB generally underestimates the optimal number of iterations although predictive performance is reas
```



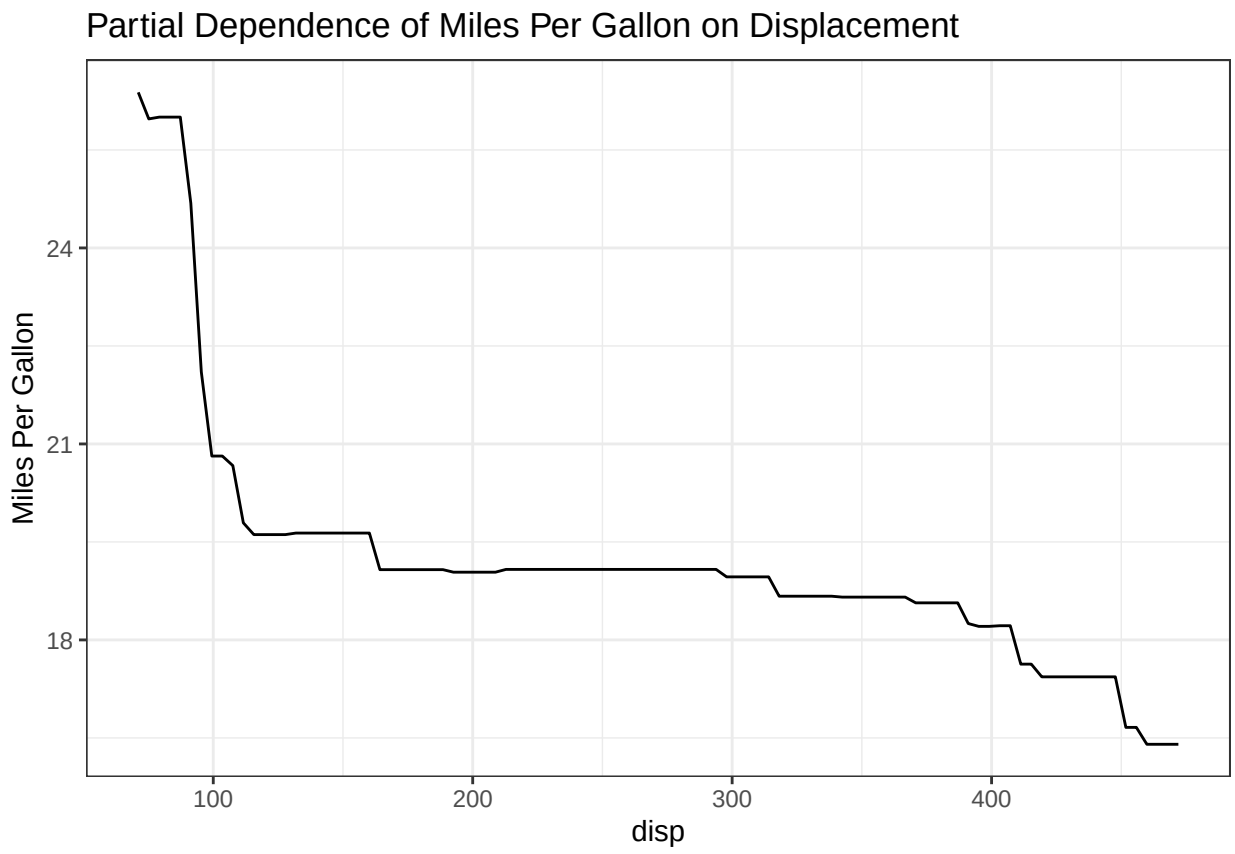

```
## [1] 154
## attr(,"smoother")
## Call:
## loess(formula = object$oobag.improve ~ x, enp.target = min(max(4,
##     length(x)/10), 50))
##
## Number of Observations: 1000
## Equivalent Number of Parameters: 40
## Residual Standard Error: 0.04626
```

Two-way partial dependence can be much more difficult to understand. It's tricky to graph this in a way that is expressive of the relationship. (There are many packages for trying to do so.)

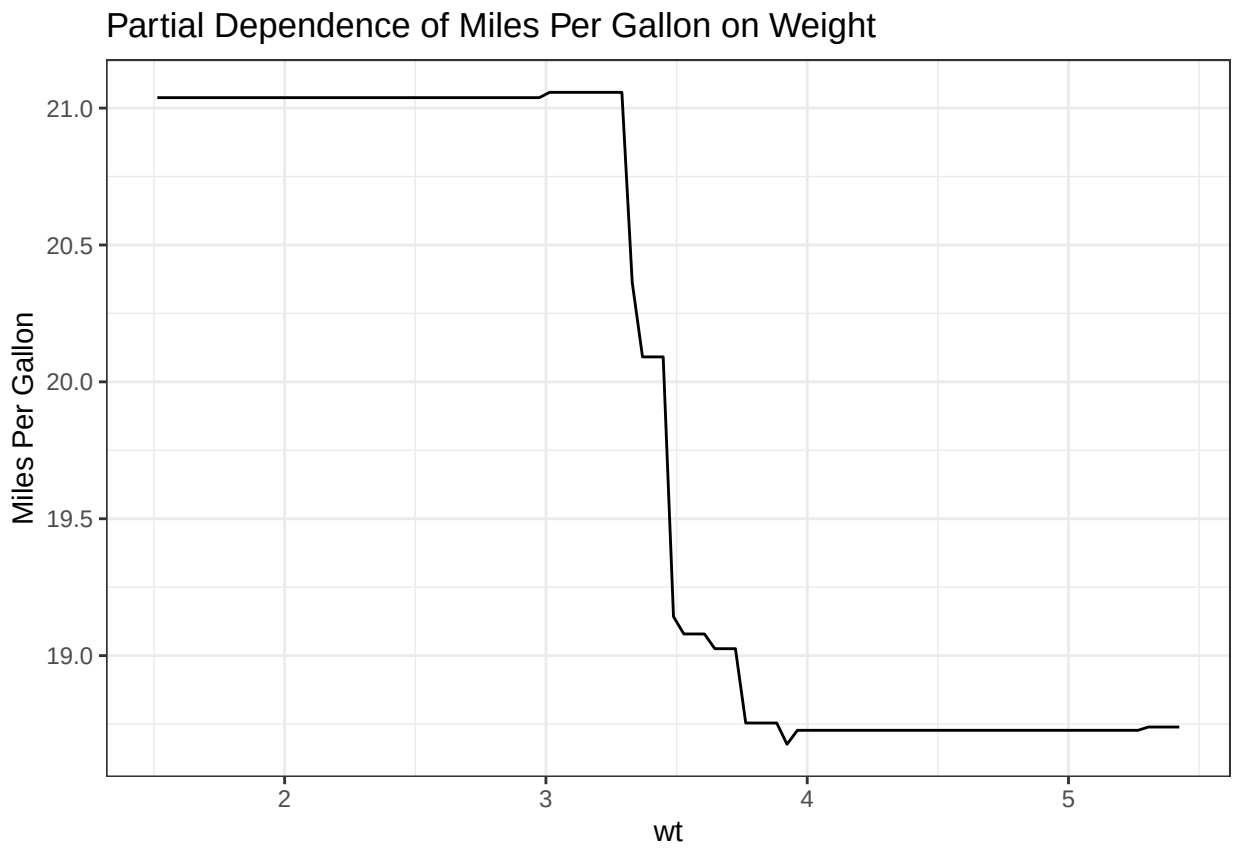
```
plot.gbm(gbm2,
         i.var = 'cyl',
         n.trees = 140,
         return.grid = TRUE) %>%
  rename(mileage = y) %>%
  ggplot(aes(x = cyl, y = mileage)) +
  geom_line() +
  theme_bw() +
  labs(x = "cyl",
       y = "Miles Per Gallon",
       title = "Partial Dependence of Miles Per Gallon on Number Cylinders")
```



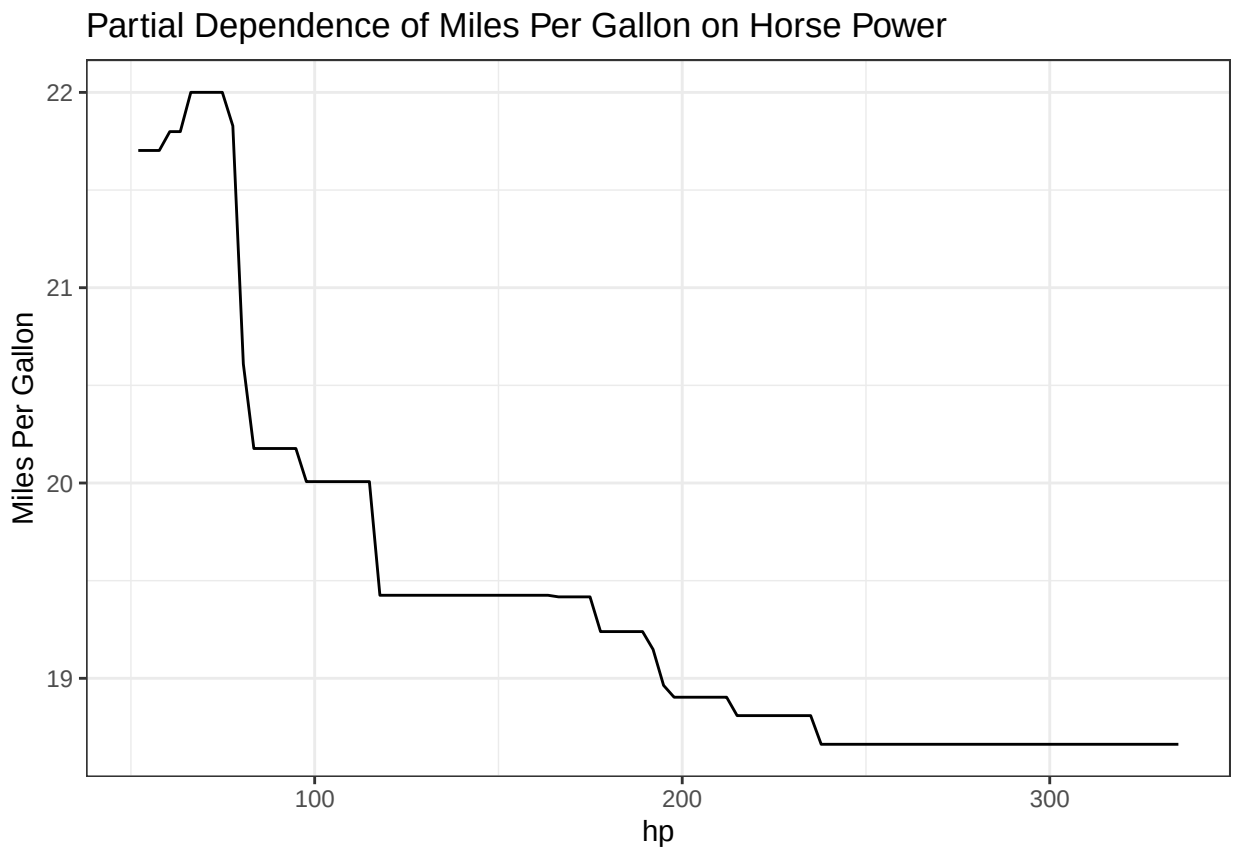

```
plot.gbm(gbm2,
         i.var = 'disp',
         n.trees = 140,
         return.grid = TRUE) %>%
  rename(mileage = y) %>%
  ggplot(aes(x = disp, y = mileage)) +
  geom_line() +
  theme_bw() +
  labs(x = "disp",
       y = "Miles Per Gallon",
       title = "Partial Dependence of Miles Per Gallon on Displacement")
```



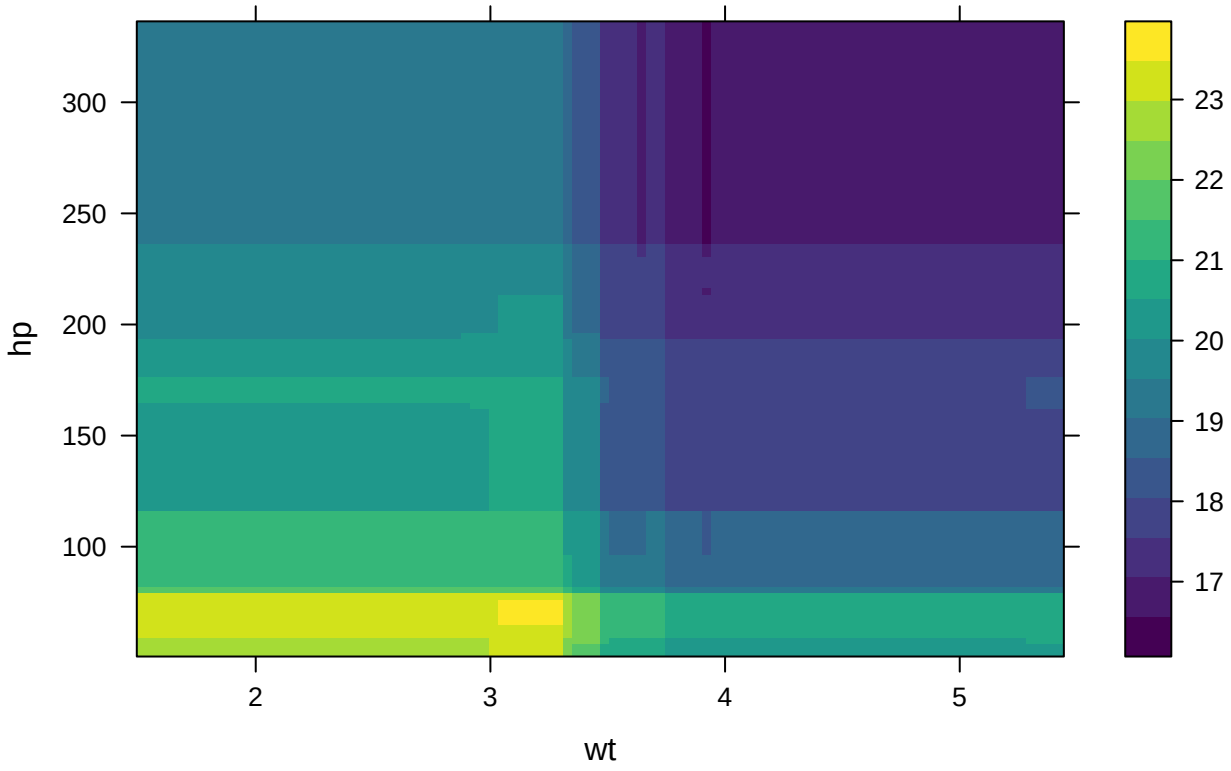

```
plot.gbm(gbm2,
         i.var = 'wt',
         n.trees = 140,
         return.grid = TRUE) %>%
  rename(mileage = y) %>%
  ggplot(aes(x = wt, y = mileage)) +
  geom_line() +
  theme_bw() +
  labs(x = "wt",
       y = "Miles Per Gallon",
       title = "Partial Dependence of Miles Per Gallon on Weight")
```




```
plot.gbm(gbm2,
         i.var = 'hp',
         n.trees = 140,
         return.grid = TRUE) %>%
  rename(mileage = y) %>%
  ggplot(aes(x = hp, y = mileage)) +
  geom_line() +
  theme_bw() +
  labs(x = "hp",
       y = "Miles Per Gallon",
       title = "Partial Dependence of Miles Per Gallon on Horse Power")
```




```
plot.gbm(gbm2,
         i.var = c("wt","hp"),
         n.trees = 500,
         return.grid = FALSE)
```


hp
300
250
200
150
100
2
3
4
5
wt
23
22
21
20
19
18
17