Untitled

Yufei Liu

2024-05-26

```
setwd("~/Documents/Document/U.S/class/spring2024/Big Data/Final_project")
game <- read.csv("game2.csv")

game$log_activePlayers <- log(game$activePlayers + 1)
game$series <- sapply(str_split(game$Title, "\\s+"), function(words) {
  paste(words[1:min(length(words), 2)], collapse = " ")
})

series_counts <- game %>%
  group_by(series, publisher) %>%
  summarise(Count = n(), .groups = 'drop')

game <- game %>%
  left_join(series_counts, by = c("series", "publisher"))

game <- game %>%
  mutate(IP_Type = case_when(
    Count >= 6 ~ "Big IP",
    Count >= 4 ~ "Medium IP",
    Count >= 2 ~ "Small IP",
    TRUE ~ "Not IP"
  ))

game <- game %>%
  filter(!is.na(activePlayers) & !is.na(genre) & !is.na(IP_Type))

game$IP_Type <- factor(game$IP_Type, levels = c("Big IP", "Medium IP", "Small IP", "Not IP"))

game <- game %>%
  drop_na(activePlayers, genre, IP_Type)

game$genre <- factor(game$genre)
levels_genre <- levels(game$genre)

set.seed(123)
trainIndex <- createDataPartition(game$IP_Type, p = .8,
                                  list = FALSE,
                                  times = 1)
gameTrain <- game[ trainIndex,]
gameTest  <- game[-trainIndex,]

gameTrain$genre <- factor(gameTrain$genre, levels = levels_genre)
```

```r
gameTest$genre <- factor(gameTest$genre, levels = levels_genre)

gameTrain_balanced <- upSample(x = gameTrain[, c("activePlayers", "genre")], y = gameTrain$IP_Type)

tree_model <- rpart(Class ~ activePlayers + genre, data = gameTrain_balanced, method = "class")

summary(tree_model)
```

```
## Call:
## rpart(formula = Class ~ activePlayers + genre, data = gameTrain_balanced,
##     method = "class")
##   n= 1528
##
##            CP nsplit rel error    xerror       xstd
## 1  0.17452007      0 1.0000000 1.0488656 0.01397381
## 2  0.03228621      1 0.8254799 0.8254799 0.01656383
## 3  0.02006981      2 0.7931937 0.8324607 0.01651900
## 4  0.01832461      3 0.7731239 0.8158813 0.01662214
## 5  0.01788831      7 0.6998255 0.7914485 0.01675341
## 6  0.01657941      9 0.6640489 0.7617801 0.01688035
## 7  0.01483421     11 0.6308901 0.7347295 0.01696570
## 8  0.01308901     12 0.6160558 0.7059337 0.01702521
## 9  0.01134380     13 0.6029668 0.6701571 0.01705458
## 10 0.01000000     14 0.5916230 0.6439791 0.01704494
##
## Variable importance
## activePlayers         genre
##            56            44
##
## Node number 1: 1528 observations,    complexity param=0.1745201
##   predicted class=Big IP     expected loss=0.75  P(node) =1
##     class counts:   382   382   382   382
##    probabilities: 0.250 0.250 0.250 0.250
##   left son=2 (752 obs) right son=3 (776 obs)
##   Primary splits:
##       genre         splits as  RLLRRL-LLLRRLRLRL, improve=63.27215, (0 missing)
##       activePlayers < 519000 to the left,  improve=17.58418, (0 missing)
##   Surrogate splits:
##       activePlayers < 11600  to the left,  agree=0.537, adj=0.059, (0 split)
##
## Node number 2: 752 observations,    complexity param=0.03228621
##   predicted class=Big IP     expected loss=0.5864362  P(node) =0.4921466
##     class counts:   311   111   133   197
##    probabilities: 0.414 0.148 0.177 0.262
##   left son=4 (650 obs) right son=5 (102 obs)
##   Primary splits:
##       genre         splits as  -LR--L-LLR--L-R-R, improve=20.14201, (0 missing)
##       activePlayers < 377950 to the right, improve=13.41335, (0 missing)
##   Surrogate splits:
##       activePlayers < 6300   to the right, agree=0.866, adj=0.01, (0 split)
##
## Node number 3: 776 observations,    complexity param=0.02006981
##   predicted class=Medium IP  expected loss=0.6507732  P(node) =0.5078534
##     class counts:    71   271   249   185
```

```
##    probabilities: 0.091 0.349 0.321 0.238
##   left son=6 (48 obs) right son=7 (728 obs)
##   Primary splits:
##       genre         splits as  R--RR-----LR-R-R-, improve=24.98582, (0 missing)
##       activePlayers < 519450 to the left,  improve=12.31947, (0 missing)
##
## Node number 4: 650 observations,    complexity param=0.01788831
##   predicted class=Big IP     expected loss=0.56  P(node) =0.4253927
##     class counts:   286   111   118   135
##    probabilities: 0.440 0.171 0.182 0.208
##   left son=8 (112 obs) right son=9 (538 obs)
##   Primary splits:
##       activePlayers < 377950 to the right, improve=14.61895, (0 missing)
##       genre         splits as  -R---L-LL---R----, improve=13.17963, (0 missing)
##
## Node number 5: 102 observations
##   predicted class=Not IP     expected loss=0.3921569  P(node) =0.06675393
##     class counts:    25     0    15    62
##    probabilities: 0.245 0.000 0.147 0.608
##
## Node number 6: 48 observations
##   predicted class=Medium IP  expected loss=0.0625  P(node) =0.03141361
##     class counts:     0    45     0     3
##    probabilities: 0.000 0.938 0.000 0.062
##
## Node number 7: 728 observations,    complexity param=0.01832461
##   predicted class=Small IP   expected loss=0.657967  P(node) =0.4764398
##     class counts:    71   226   249   182
##    probabilities: 0.098 0.310 0.342 0.250
##   left son=14 (663 obs) right son=15 (65 obs)
##   Primary splits:
##       activePlayers < 519450 to the left,  improve=9.618929, (0 missing)
##       genre         splits as  L--LR------L-R-L-, improve=6.923770, (0 missing)
##
## Node number 8: 112 observations,    complexity param=0.01308901
##   predicted class=Big IP     expected loss=0.3660714  P(node) =0.07329843
##     class counts:    71     0     3    38
##    probabilities: 0.634 0.000 0.027 0.339
##   left son=16 (94 obs) right son=17 (18 obs)
##   Primary splits:
##       activePlayers < 742500 to the left,  improve=14.27318, (0 missing)
##       genre         splits as  -L---R-RL---R----, improve=11.27786, (0 missing)
##
## Node number 9: 538 observations,    complexity param=0.01788831
##   predicted class=Big IP     expected loss=0.6003717  P(node) =0.3520942
##     class counts:   215   111   115    97
##    probabilities: 0.400 0.206 0.214 0.180
##   left son=18 (441 obs) right son=19 (97 obs)
##   Primary splits:
##       activePlayers < 198000 to the left,  improve=26.43506, (0 missing)
##       genre         splits as  -R---L-LL---L----, improve=17.62809, (0 missing)
##
## Node number 14: 663 observations,    complexity param=0.01832461
##   predicted class=Medium IP  expected loss=0.6591252  P(node) =0.4339005
```

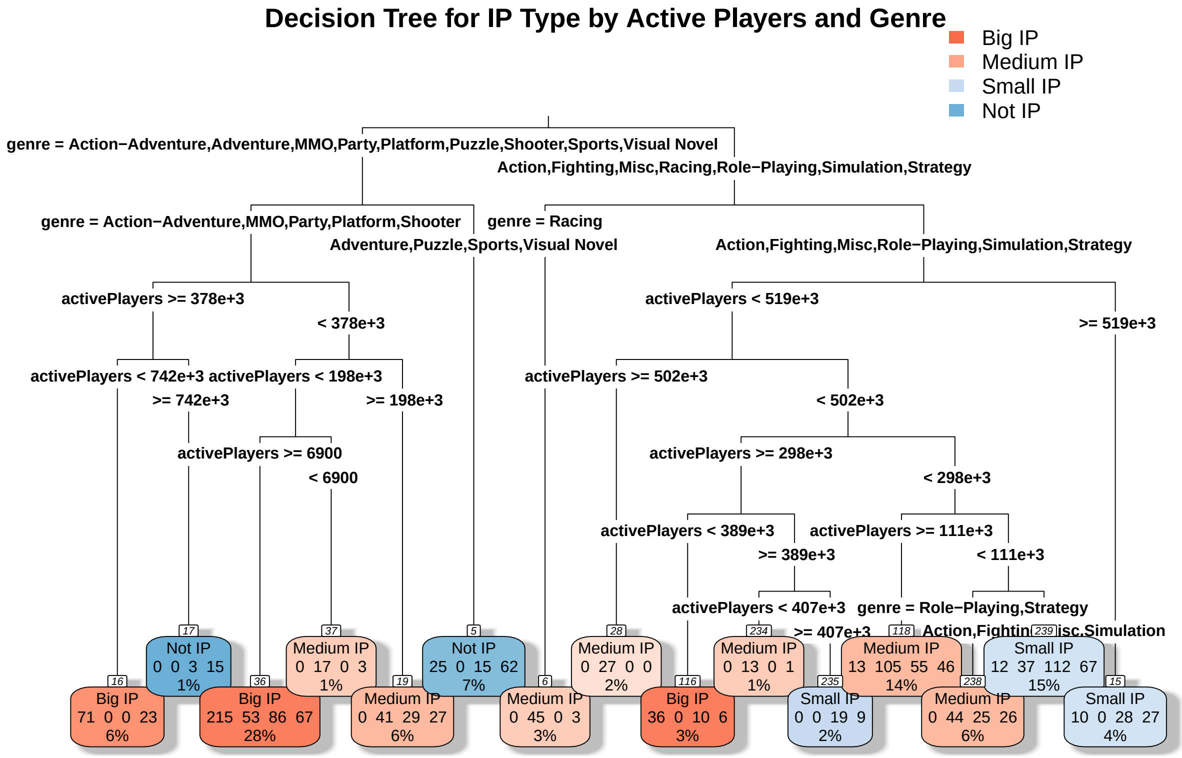
```
##     class counts:    61   226   221   155
##    probabilities: 0.092 0.341 0.333 0.234
##   left son=28 (27 obs) right son=29 (636 obs)
##   Primary splits:
##       activePlayers < 502000 to the right, improve=17.131990, (0 missing)
##       genre         splits as  L--LR------L-R-L-, improve= 7.691782, (0 missing)
##
## Node number 15: 65 observations
##   predicted class=Small IP   expected loss=0.5692308  P(node) =0.04253927
##     class counts:    10     0    28    27
##    probabilities: 0.154 0.000 0.431 0.415
##
## Node number 16: 94 observations
##   predicted class=Big IP     expected loss=0.2446809  P(node) =0.06151832
##     class counts:    71     0     0    23
##    probabilities: 0.755 0.000 0.000 0.245
##
## Node number 17: 18 observations
##   predicted class=Not IP     expected loss=0.1666667  P(node) =0.0117801
##     class counts:     0     0     3    15
##    probabilities: 0.000 0.000 0.167 0.833
##
## Node number 18: 441 observations,    complexity param=0.01483421
##   predicted class=Big IP     expected loss=0.5124717  P(node) =0.2886126
##     class counts:   215    70    86    70
##    probabilities: 0.488 0.159 0.195 0.159
##   left son=36 (421 obs) right son=37 (20 obs)
##   Primary splits:
##       activePlayers < 6900   to the right, improve=15.78892, (0 missing)
##       genre         splits as  -R---L-LR---R----, improve=12.29881, (0 missing)
##
## Node number 19: 97 observations
##   predicted class=Medium IP  expected loss=0.5773196  P(node) =0.06348168
##     class counts:     0    41    29    27
##    probabilities: 0.000 0.423 0.299 0.278
##
## Node number 28: 27 observations
##   predicted class=Medium IP  expected loss=0  P(node) =0.01767016
##     class counts:     0    27     0     0
##    probabilities: 0.000 1.000 0.000 0.000
##
## Node number 29: 636 observations,    complexity param=0.01832461
##   predicted class=Small IP   expected loss=0.6525157  P(node) =0.4162304
##     class counts:    61   199   221   155
##    probabilities: 0.096 0.313 0.347 0.244
##   left son=58 (94 obs) right son=59 (542 obs)
##   Primary splits:
##       activePlayers < 297950 to the right, improve=13.215540, (0 missing)
##       genre         splits as  R--RR------L-R-R-, improve= 8.220527, (0 missing)
##
## Node number 36: 421 observations
##   predicted class=Big IP     expected loss=0.4893112  P(node) =0.2755236
##     class counts:   215    53    86    67
##    probabilities: 0.511 0.126 0.204 0.159
```

```
##
## Node number 37: 20 observations
##   predicted class=Medium IP  expected loss=0.15  P(node) =0.01308901
##     class counts:     0    17     0     3
##    probabilities: 0.000 0.850 0.000 0.150
##
## Node number 58: 94 observations,    complexity param=0.01657941
##   predicted class=Big IP     expected loss=0.6170213  P(node) =0.06151832
##     class counts:    36    13    29    16
##    probabilities: 0.383 0.138 0.309 0.170
##   left son=116 (52 obs) right son=117 (42 obs)
##   Primary splits:
##       activePlayers < 389100 to the left,  improve=15.283140, (0 missing)
##       genre         splits as  R--L-------L-L-L-, improve= 3.609572, (0 missing)
##   Surrogate splits:
##       genre splits as  L--L-------R-R-L-, agree=0.628, adj=0.167, (0 split)
##
## Node number 59: 542 observations,    complexity param=0.01832461
##   predicted class=Small IP   expected loss=0.6457565  P(node) =0.354712
##     class counts:    25   186   192   139
##    probabilities: 0.046 0.343 0.354 0.256
##   left son=118 (219 obs) right son=119 (323 obs)
##   Primary splits:
##       activePlayers < 110700 to the right, improve=11.587250, (0 missing)
##       genre         splits as  R--RR------L-R-R-, improve= 8.214843, (0 missing)
##   Surrogate splits:
##       genre splits as  R--RR------L-R-R-, agree=0.627, adj=0.078, (0 split)
##
## Node number 116: 52 observations
##   predicted class=Big IP     expected loss=0.3076923  P(node) =0.03403141
##     class counts:    36     0    10     6
##    probabilities: 0.692 0.000 0.192 0.115
##
## Node number 117: 42 observations,    complexity param=0.0113438
##   predicted class=Small IP   expected loss=0.547619  P(node) =0.02748691
##     class counts:     0    13    19    10
##    probabilities: 0.000 0.310 0.452 0.238
##   left son=234 (14 obs) right son=235 (28 obs)
##   Primary splits:
##       activePlayers < 406600 to the left,  improve=12.92857, (0 missing)
##       genre         splits as  R----------L-L---, improve= 2.86250, (0 missing)
##
## Node number 118: 219 observations
##   predicted class=Medium IP  expected loss=0.5205479  P(node) =0.1433246
##     class counts:    13   105    55    46
##    probabilities: 0.059 0.479 0.251 0.210
##
## Node number 119: 323 observations,    complexity param=0.01657941
##   predicted class=Small IP   expected loss=0.5758514  P(node) =0.2113874
##     class counts:    12    81   137    93
##    probabilities: 0.037 0.251 0.424 0.288
##   left son=238 (95 obs) right son=239 (228 obs)
##   Primary splits:
##       genre         splits as  R--RR------L-R-L-, improve=9.771827, (0 missing)
```

```
##       activePlayers < 24000  to the left,  improve=9.226444, (0 missing)
##   Surrogate splits:
##       activePlayers < 15200  to the left,  agree=0.734, adj=0.095, (0 split)
##
## Node number 234: 14 observations
##   predicted class=Medium IP  expected loss=0.07142857  P(node) =0.009162304
##     class counts:     0    13     0     1
##    probabilities: 0.000 0.929 0.000 0.071
##
## Node number 235: 28 observations
##   predicted class=Small IP   expected loss=0.3214286  P(node) =0.01832461
##     class counts:     0     0    19     9
##    probabilities: 0.000 0.000 0.679 0.321
##
## Node number 238: 95 observations
##   predicted class=Medium IP  expected loss=0.5368421  P(node) =0.06217277
##     class counts:     0    44    25    26
##    probabilities: 0.000 0.463 0.263 0.274
##
## Node number 239: 228 observations
##   predicted class=Small IP   expected loss=0.5087719  P(node) =0.1492147
##     class counts:    12    37   112    67
##    probabilities: 0.053 0.162 0.491 0.294
```

```r
rpart.plot(tree_model, type = 3, extra = 101, fallen.leaves = TRUE,
           main = "Decision Tree for IP Type by Active Players and Genre",
           cex = 0.4,
           tweak = 1.2,
           box.palette = "RdBu", shadow.col = "gray", nn = TRUE)
```

```
## Warning: cex and tweak both specified, applying both
```




```r
train_pred <- predict(tree_model, newdata = gameTrain, type = "class")

# Test set prediction
test_pred <- predict(tree_model, newdata = gameTest, type = "class")

# Calculating training set accuracy
train_accuracy <- mean(train_pred == gameTrain$IP_Type)
test_accuracy <- mean(test_pred == gameTest$IP_Type)

# Output results
print(paste("Training set accuracy:", train_accuracy))
```

```
## [1] "Training set accuracy: 0.3"
```

```r
print(paste("Test set accuracy:", test_accuracy))
```

```
## [1] "Test set accuracy: 0.203125"
```

```r
# Calculating training set and test set R²
train_R2 <- caret::postResample(as.numeric(train_pred), as.numeric(gameTrain$IP_Type))["Rsquared"]
test_R2 <- caret::postResample(as.numeric(test_pred), as.numeric(gameTest$IP_Type))["Rsquared"]

# Output results
print(paste("Training set R²:", train_R2))
```

```
## [1] "Training set R²: 0.0548582100060946"
```

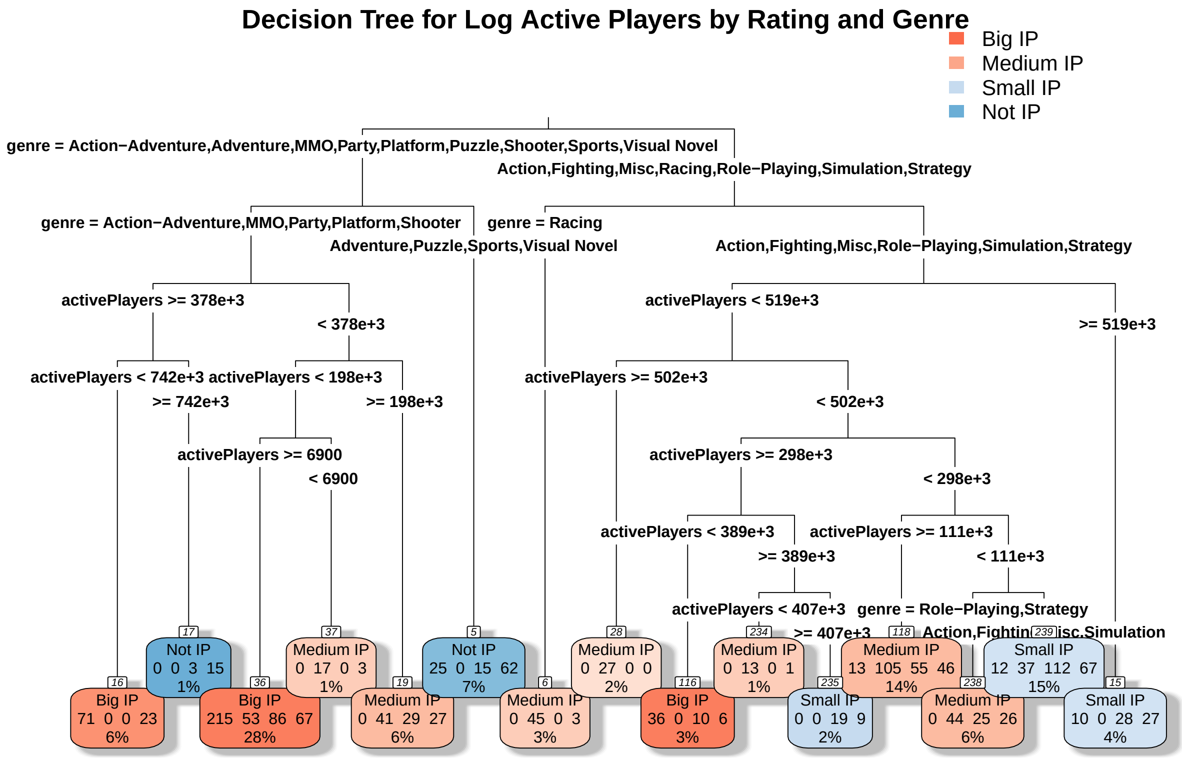
```r
print(paste("Test set R²:", test_R2))
```

```
## [1] "Test set R²: 0.0049286465494176"
```

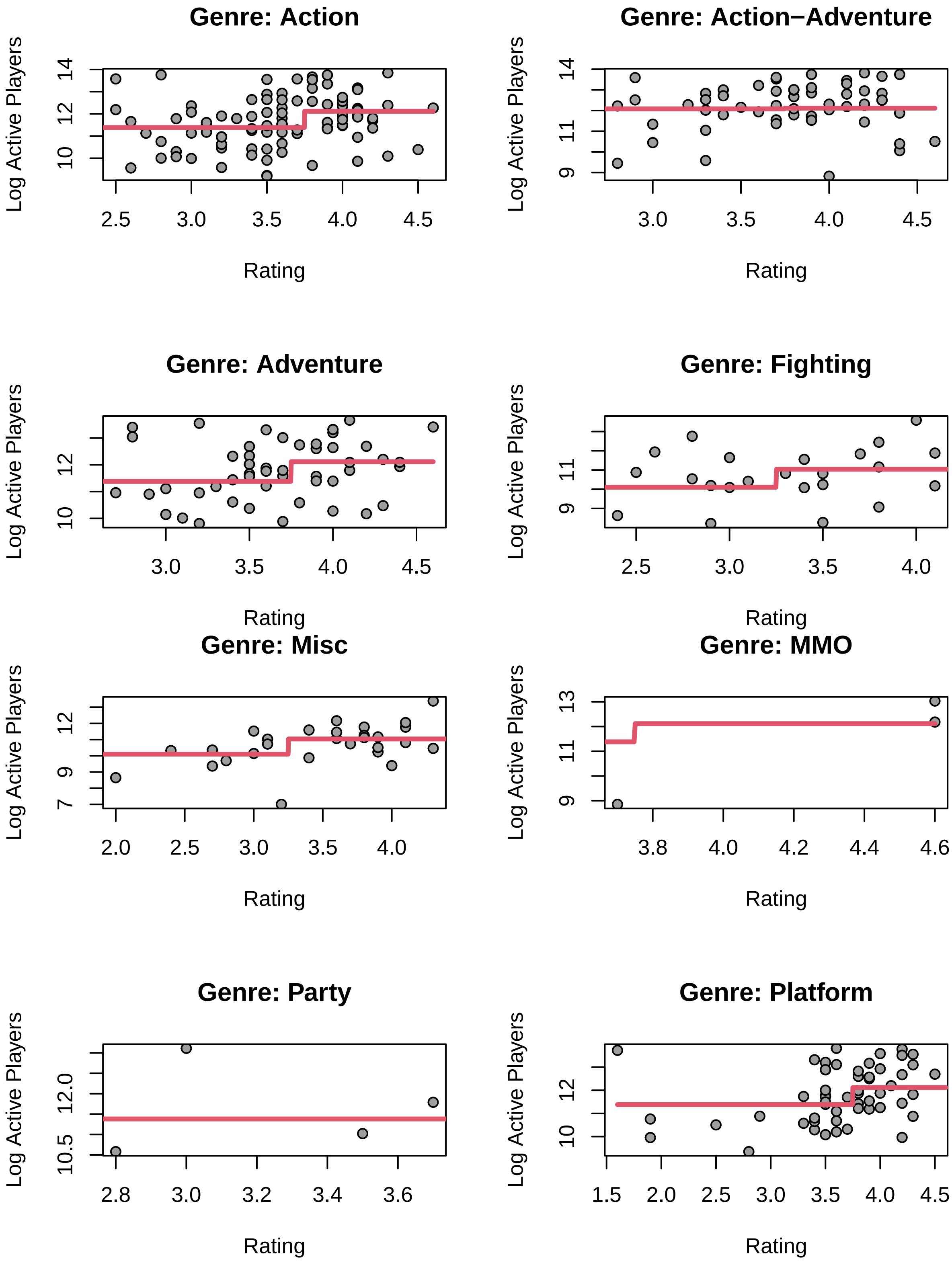
```r
library(tree)
game_tree <- tree(log_activePlayers ~ Rating + genre, data=gameTrain)
```

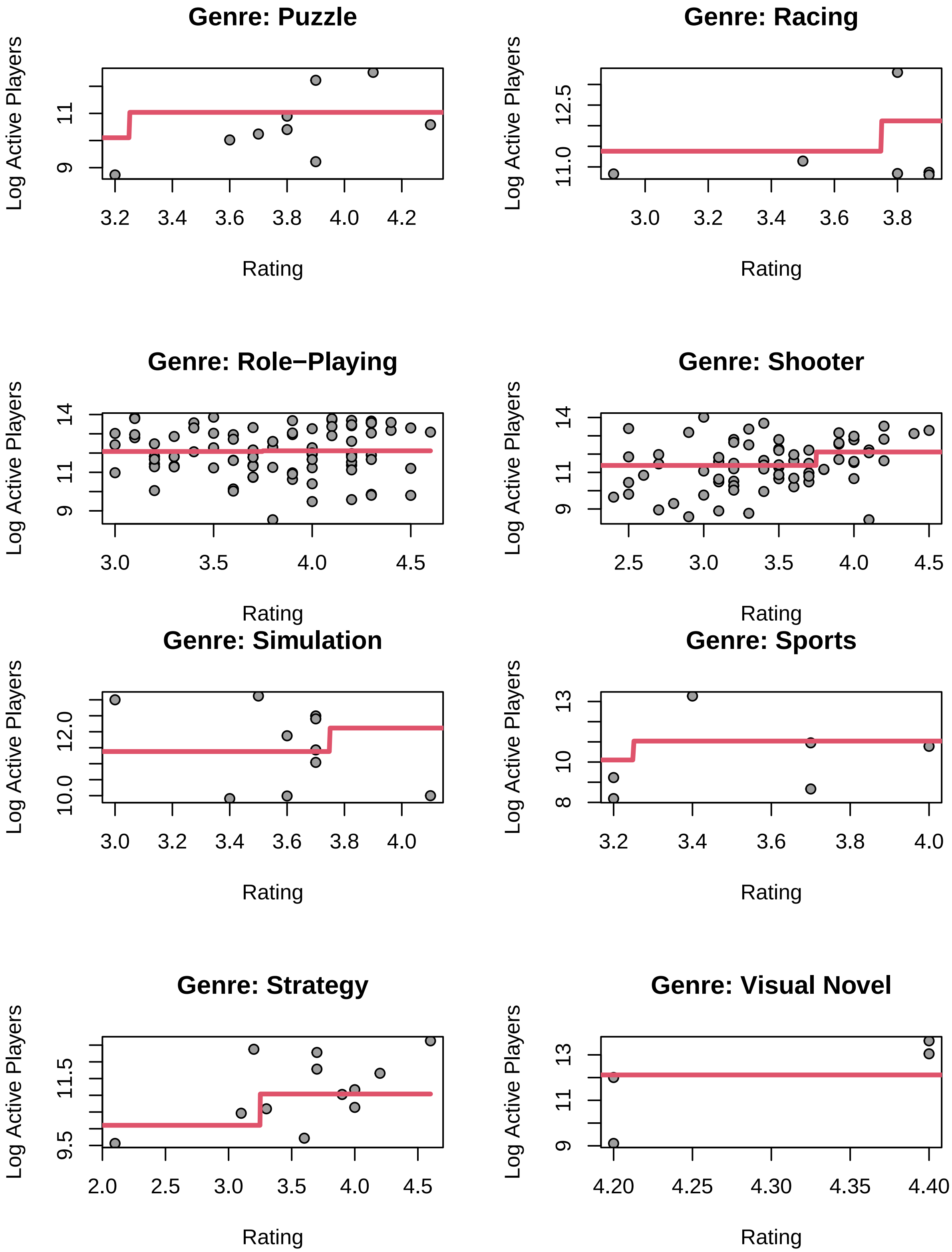
```r
rpart.plot(tree_model, type = 3, extra = 101, fallen.leaves = TRUE,
           main = "Decision Tree for Log Active Players by Rating and Genre",
           cex = 0.4,
           tweak = 1.2,
           box.palette = "RdBu", shadow.col = "gray", nn = TRUE)
```

```
## Warning: cex and tweak both specified, applying both
```



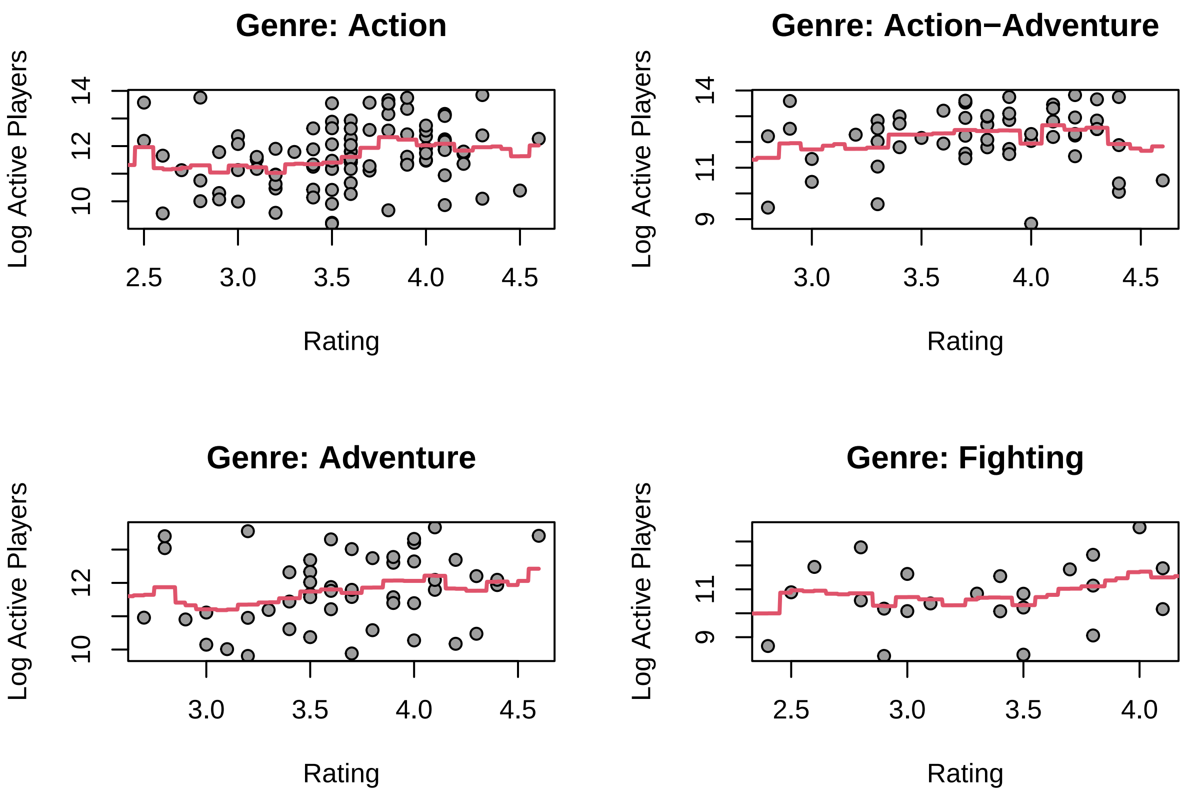

```r
par(mfrow=c(2,2))
rating_grid <- expand.grid(Rating = seq(min(game$Rating), max(game$Rating), length = 1000),
                           genre = levels_genre)

for (g in levels_genre) {
  genre_grid <- rating_grid %>% filter(genre == g)
  train_data <- gameTrain %>% filter(genre == g)

  if (nrow(train_data) > 0) {
    plot(train_data$Rating, train_data$log_activePlayers,
         pch=21, bg=8, xlab="Rating", ylab="Log Active Players", main=paste("Genre:", g))
    lines(genre_grid$Rating, predict(game_tree, newdata=genre_grid), col=2, lwd=3)
  }
}
```

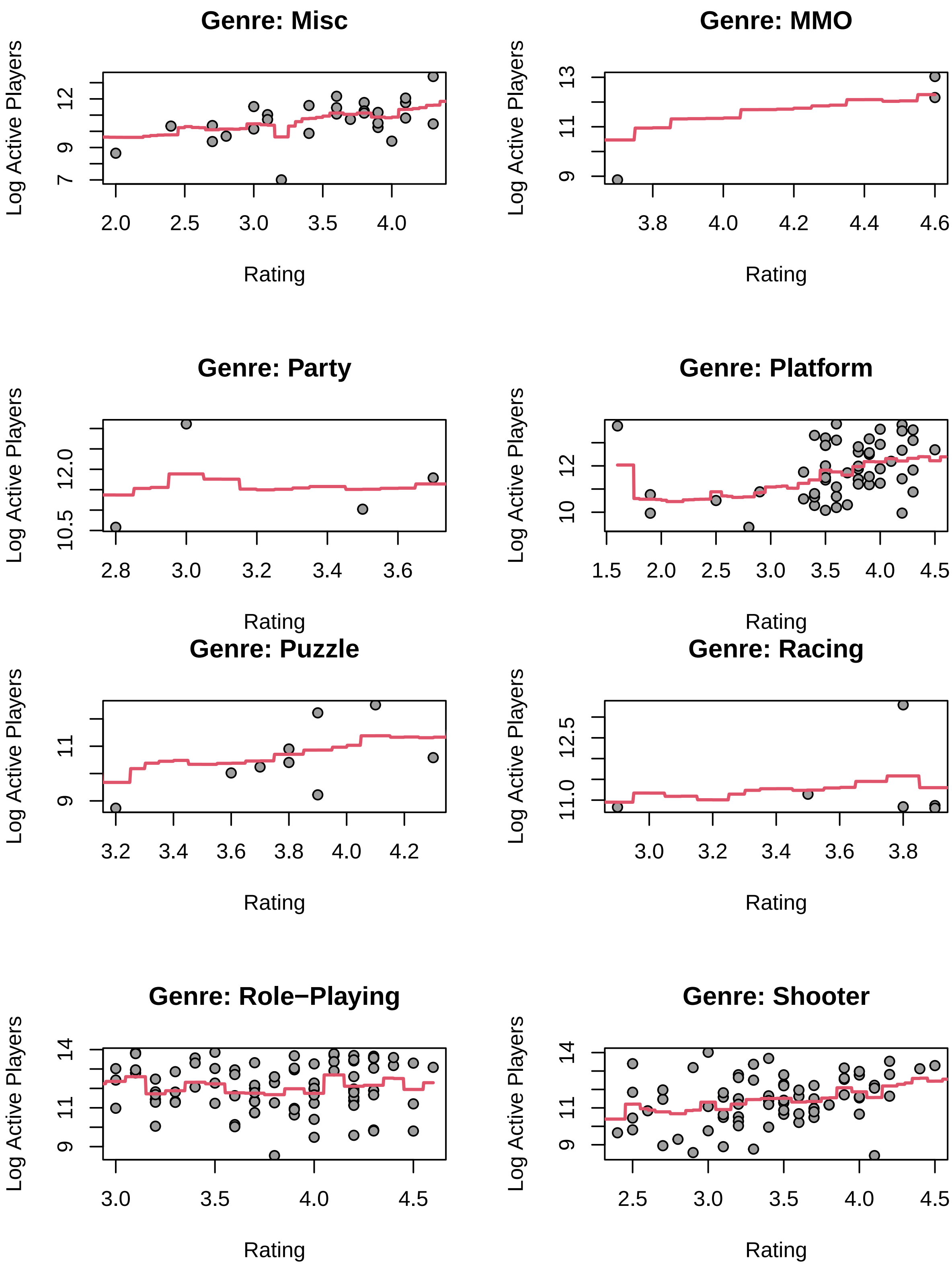

Genre: Action
Genre: Action−Adventure
Genre: Adventure
Genre: Fighting
Genre: Misc
Genre: MMO
Genre: Party
Genre: Platform
Log Active Players
Rating

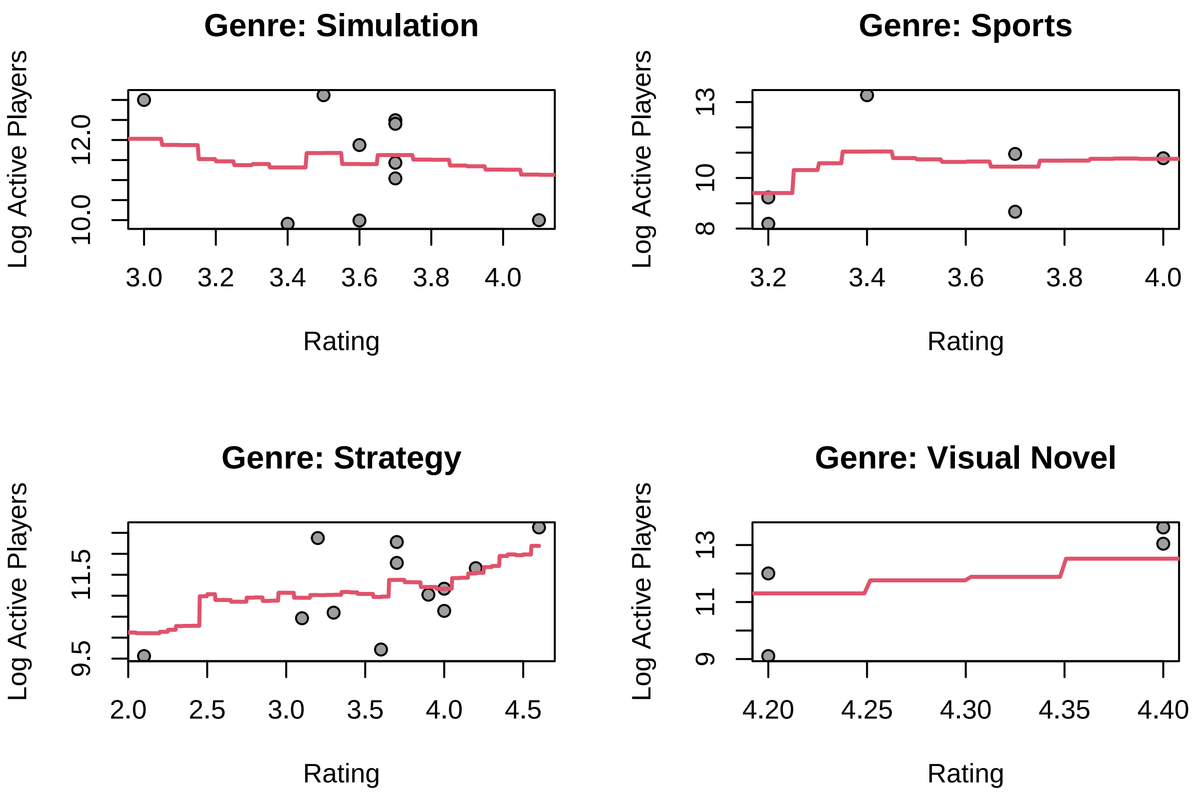



```r
rf_game <- randomForest(log_activePlayers ~ Rating + genre, data=gameTrain, nodesize=10)

for (g in levels_genre) {
  genre_grid <- rating_grid %>% filter(genre == g)
```

```
  train_data <- gameTrain %>% filter(genre == g)

  if (nrow(train_data) > 0) {
    pred_rf_game <- predict(rf_game, genre_grid)
    plot(train_data$Rating, train_data$log_activePlayers,
         pch=21, bg=8, xlab="Rating", ylab="Log Active Players", main=paste("Genre:", g))
    lines(genre_grid$Rating, pred_rf_game, col=2, lwd=2)
  }
}
```




Genre: Misc
Log Active Players
7 9 12
2.0 2.5 3.0 3.5 4.0
Rating
Genre: MMO
Log Active Players
9 11 13
3.8 4.0 4.2 4.4 4.6
Rating
Genre: Party
Log Active Players
10.5 12.0
2.8 3.0 3.2 3.4 3.6
Rating
Genre: Platform
Log Active Players
10 12
1.5 2.0 2.5 3.0 3.5 4.0 4.5
Rating
Genre: Puzzle
Log Active Players
9 11
3.2 3.4 3.6 3.8 4.0 4.2
Rating
Genre: Racing
Log Active Players
11.0 12.5
3.0 3.2 3.4 3.6 3.8
Rating
Genre: Role−Playing
Log Active Players
9 11 14
3.0 3.5 4.0 4.5
Rating
Genre: Shooter
Log Active Players
9 11 14
2.5 3.0 3.5 4.0 4.5
Rating




```r
plot(rf_game)
```