Predicting household composition by TV viewing behavior

Rafael Lüchinger

2019-02-11

Data

```
load("~/diplom/data/data_predictors.RData")

# all(hh.composition$hh == predictors$hh)
d <- cbind(hhsize = hh.composition$hhsize, predictors[,-1])

# classification or regression, target as.factor or as.integer?
d$hhsize <- factor(d$hhsize)
levels(d$hhsize) <- paste0("hhsize", levels(d$hhsize))

d[1:5, 1:4]
```

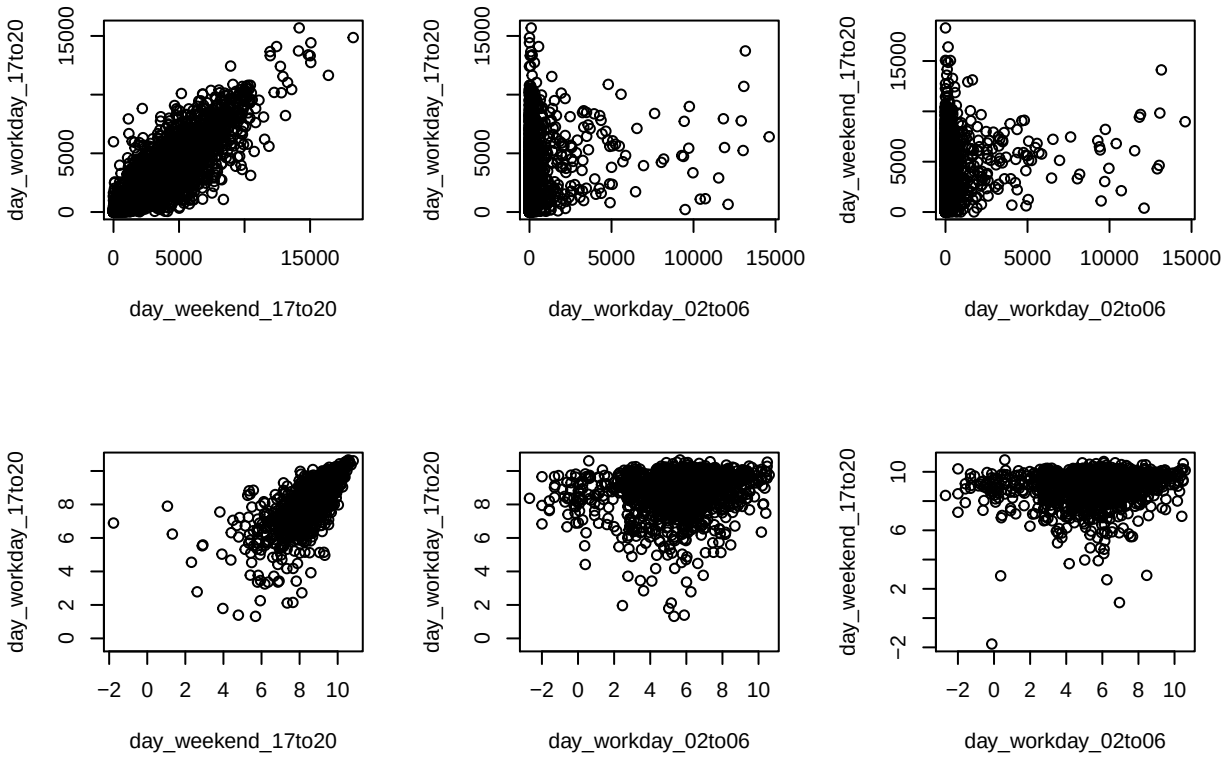
```
##    hhsize day_weekend_02to06 day_weekend_06to08 day_weekend_08to11
## 1 hhsize2             0.0000             0.0000           372.8750
## 2 hhsize4            88.3125            20.7500           621.5000
## 3 hhsize2           328.1250            39.2500            12.0000
## 4 hhsize2          1019.6667           555.1333           824.4667
## 5 hhsize2           607.3750           917.2500          3143.5000
```

```
matrix(names(d), ncol = 3)
```

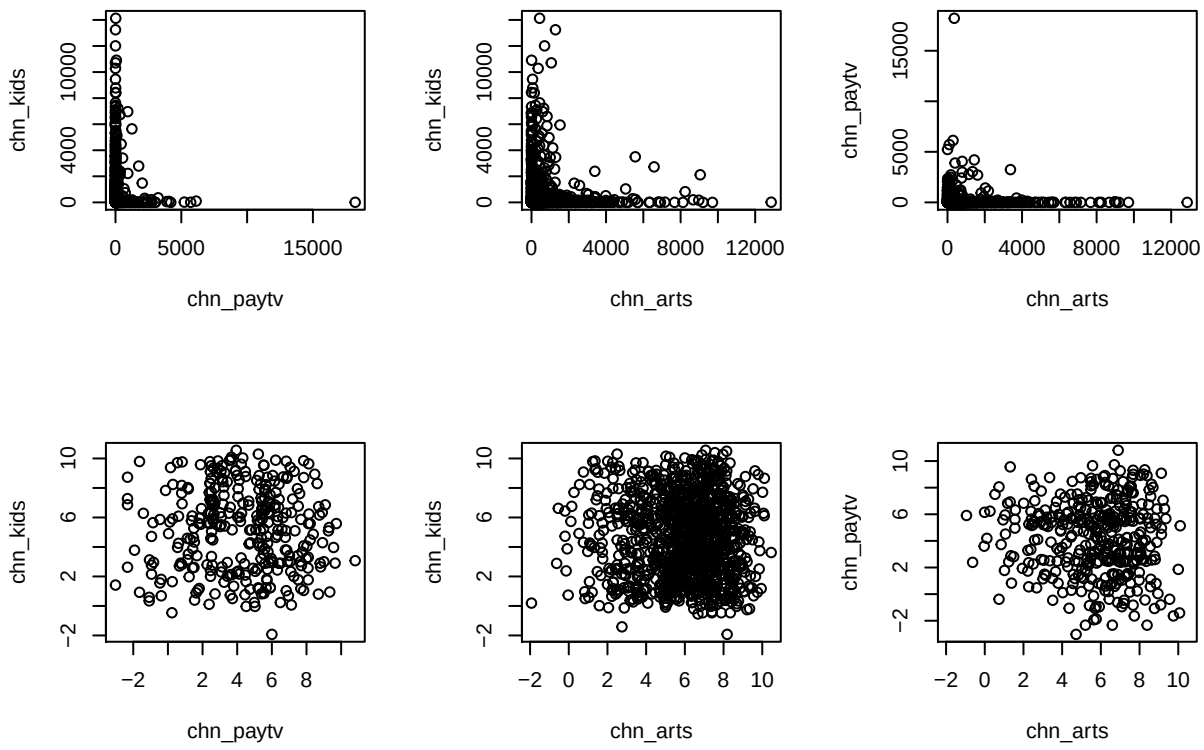
```
##       [,1]                [,2]                     [,3]
##  [1,] "hhsize"            "day_workday_24to02"     "chn_french"
##  [2,] "day_weekend_02to06" "chn_arts"               "chn_german"
##  [3,] "day_weekend_06to08" "chn_generalistprivate"  "chn_italian"
##  [4,] "day_weekend_08to11" "chn_generalistpublic"   "chn_other"
##  [5,] "day_weekend_11to13" "chn_kids"               "prg_commercial"
##  [6,] "day_weekend_13to17" "chn_livestileindoor"    "prg_info"
##  [7,] "day_weekend_17to20" "chn_livestileoutdoor"   "prg_kids"
##  [8,] "day_weekend_20to22" "chn_local"              "prg_missing"
##  [9,] "day_weekend_22to24" "chn_movieseries"        "prg_movie"
## [10,] "day_weekend_24to02" "chn_music"              "prg_music"
## [11,] "day_workday_02to06" "chn_nature"             "prg_news"
## [12,] "day_workday_06to08" "chn_news"               "prg_other"
## [13,] "day_workday_08to11" "chn_paytv"              "prg_series"
## [14,] "day_workday_11to13" "chn_religion"           "prg_service"
## [15,] "day_workday_13to17" "chn_sport"              "prg_show"
## [16,] "day_workday_17to20" "chn_foreign"            "prg_sport"
## [17,] "day_workday_20to22" "chn_swiss"              "prg_talk"
## [18,] "day_workday_22to24" "chn_english"            "prg_trailer"
```

data log tranformation

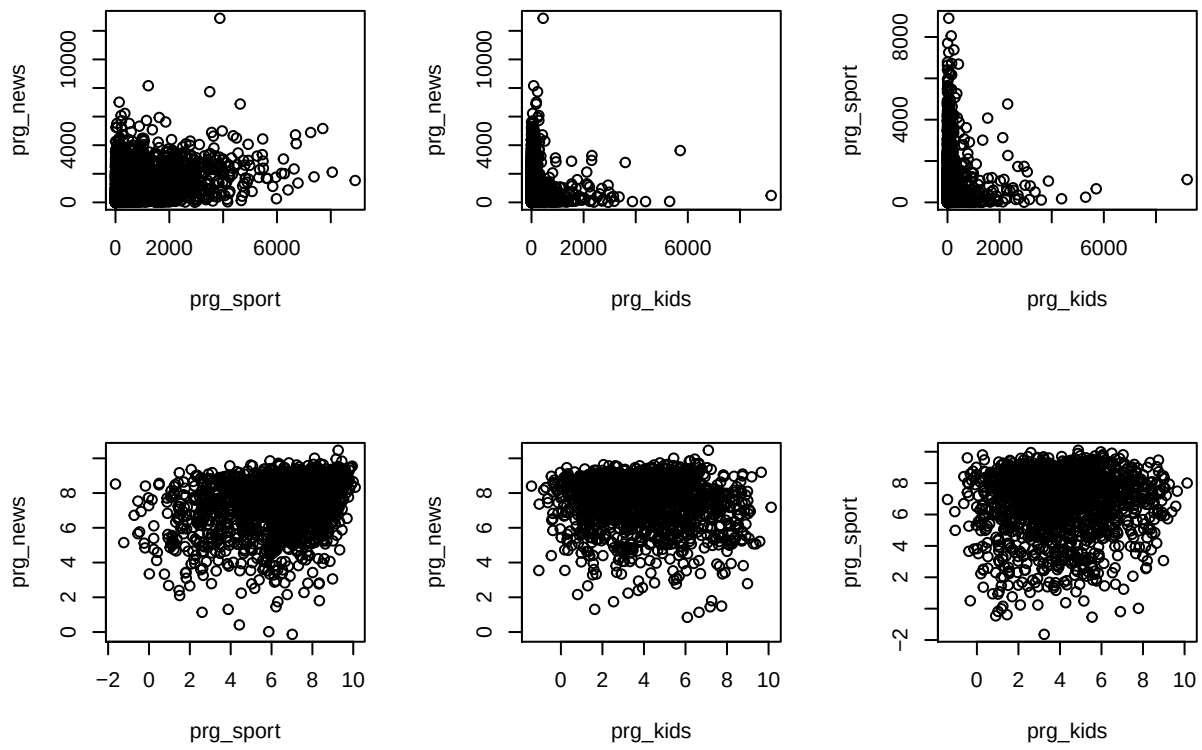
```
par(mfrow = c(2,3))
cols <- c('day_workday_02to06','day_weekend_17to20','day_workday_17to20')
for(i in seq_along(cols)) plot(d[, cols[-i]])
for(i in seq_along(cols)) plot(log(d[, cols[-i]]) + 1)
```




```
par(mfrow = c(2,3))
cols <- c('chn_arts','chn_paytv', 'chn_kids')
for(i in seq_along(cols)) plot(d[, cols[-i]])
for(i in seq_along(cols)) plot(log(d[, cols[-i]]) + 1)
```



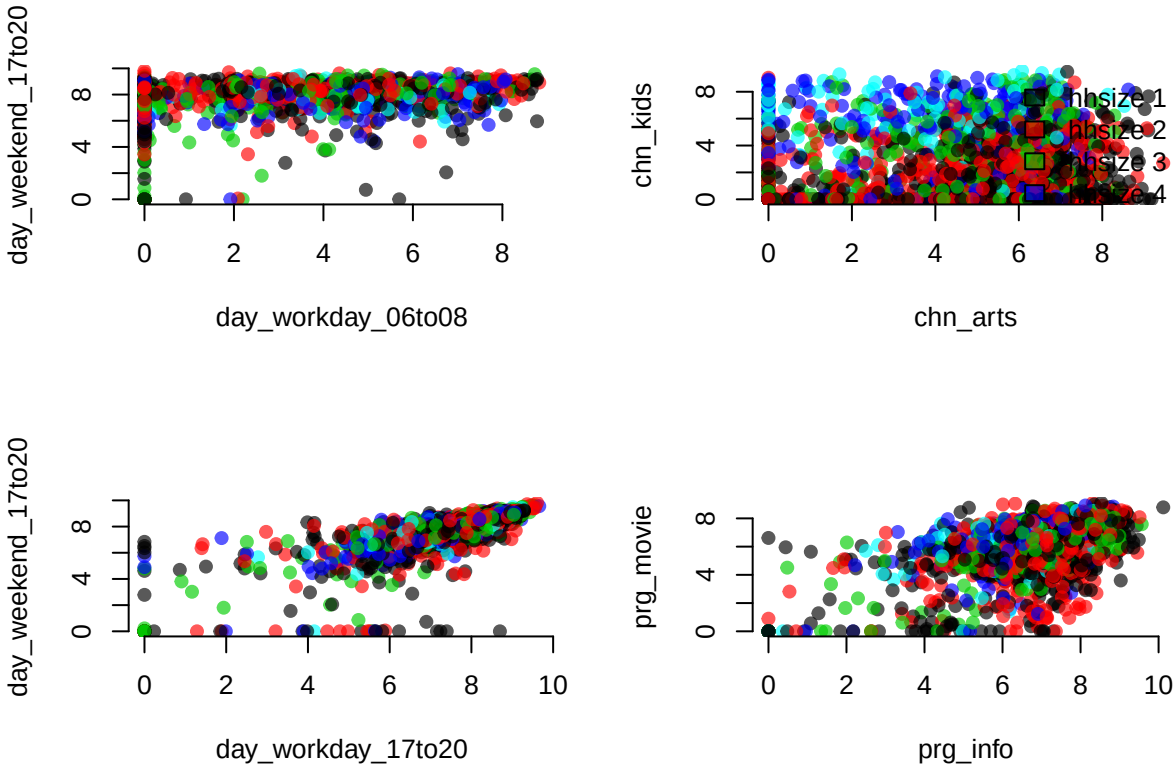

```
par(mfrow = c(2,3))
cols <-  c('prg_kids','prg_sport','prg_news')
for(i in seq_along(cols)) plot(d[, cols[-i]])
for(i in seq_along(cols)) plot(log(d[, cols[-i]]) + 1)
```




```
# log transformation log(x + 1)
d[, -1] <- lapply(d[, -1] + 1, log)
```

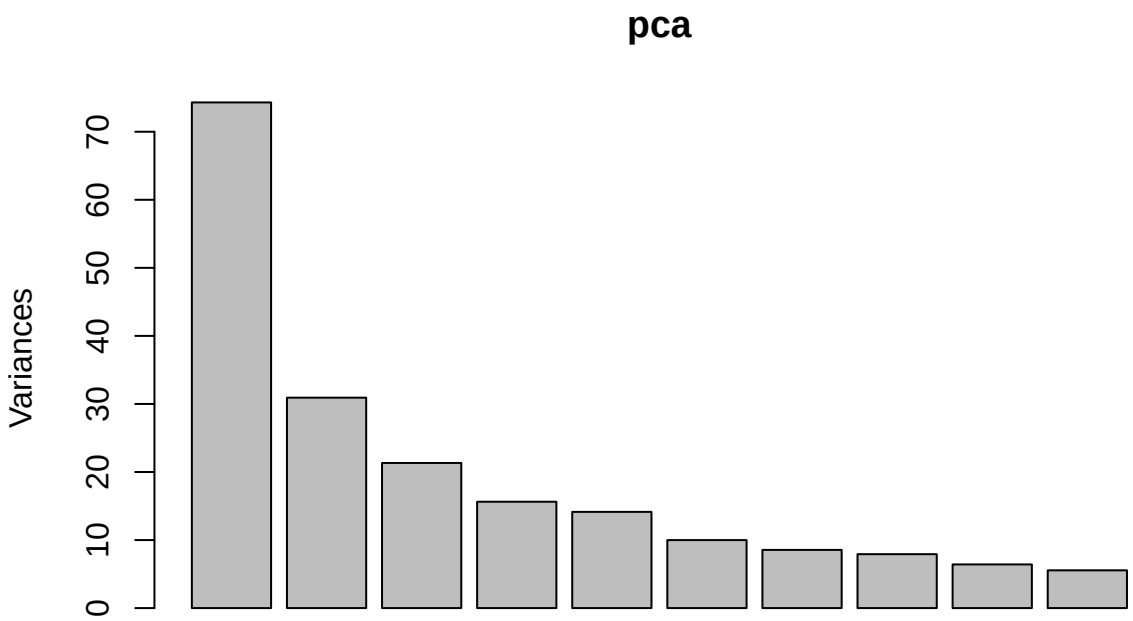
exploring clustering by hhsize

```
fun <- function(x, cols, col = adjustcolor(as.integer(x$hhsize), .65))
  plot(x[, cols], bg = col, pch = 21, cex = 1, col = col, frame.plot = FALSE)

par(mfcol = c(2,2))
cols <- c('day_workday_06to08','day_weekend_17to20')
fun(d, cols)
cols <- c('day_workday_17to20','day_weekend_17to20')
fun(d, cols)
cols <- c('chn_arts','chn_kids')
fun(d, cols)
legend('topright', paste('hhsize', 1:5), bty = 'n', fill = adjustcolor(1:5, .65))
cols <- c('prg_info','prg_movie')
fun(d, cols)
```

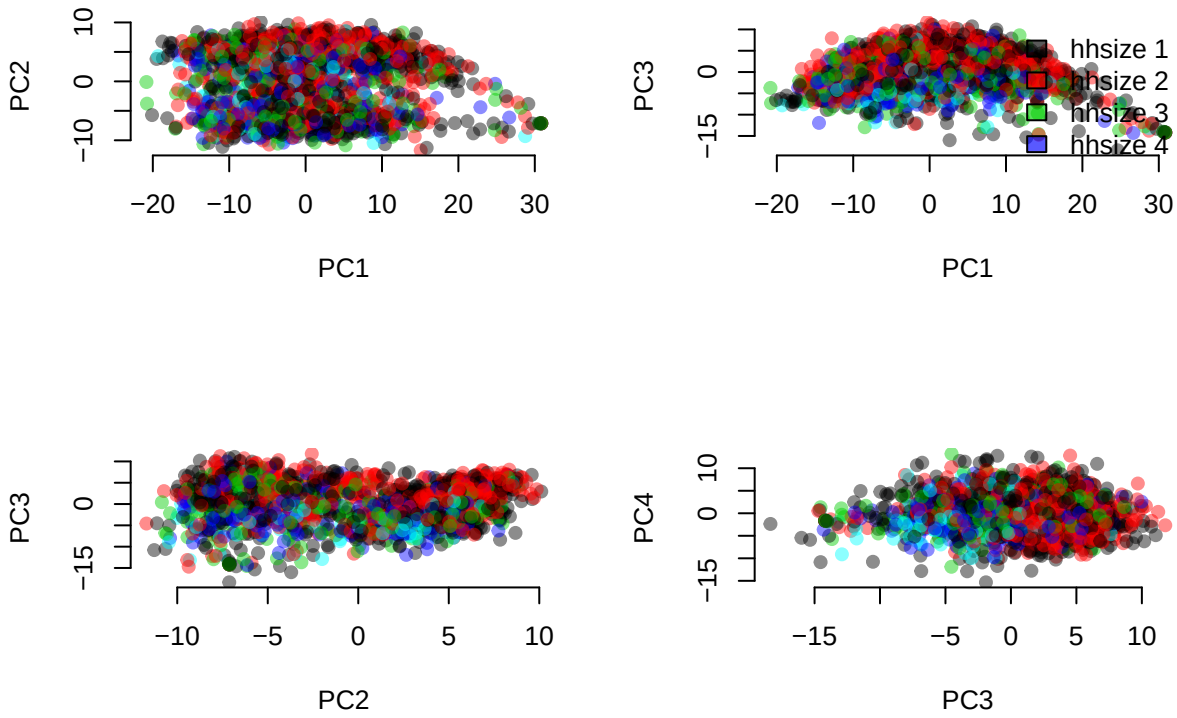


Principal Component Analysis

```
pca <- prcomp(d[,-1])
par(mfcol = c(1,1))
plot(pca)
```




```
par(mfrow = c(2,2))
fun(pca$x, 1:2, col = adjustcolor(as.integer(d$hhsize), .45))
fun(pca$x, c(1,3), col = adjustcolor(as.integer(d$hhsize), .45))
legend('topright', paste('hhsize', 1:5), bty = 'n', fill = adjustcolor(1:5, .65))
fun(pca$x, 2:3, col = adjustcolor(as.integer(d$hhsize), .45))
fun(pca$x, 3:4, col = adjustcolor(as.integer(d$hhsize), .45))
```



Split data into training & test

```
set.seed(1)
d <- setNames(split(d, runif(nrow(d)) > .6), c("train","test"))

# summary
tbl <- function(x){
  x <- table(x$hhsize)
  x <- rbind(n = x, `%` = round(prop.table(x) * 100, 2))
  t(cbind(x, total = round(rowSums(x))))
}
lapply(d, tbl)
```

```
## $train
##              n      %
## hhsize1  417  34.18
## hhsize2  409  33.52
## hhsize3  164  13.44
## hhsize4  163  13.36
## hhsize5   67   5.49
## total   1220 100.00
##
## $test
```

```
##           n      %
## hhsize1 276  35.11
## hhsize2 247  31.42
## hhsize3 111  14.12
## hhsize4 105  13.36
## hhsize5  47   5.98
## total   786 100.00
```

```r
# chance is 1/5
mean(hh.composition$hhsize == sample(1:5, nrow(hh.composition), replace = TRUE))
```

```
## [1] 0.2043868
```

Support Vector Machine

Linear Kernel

```r
library(e1071)
```

```r
# linear
# svm.linear.tune <- tune.svm(
#   hhsize ~ ., data = d$train, kernel = "linear",
#   cost  = seq(0.01, 1, length.out = 5) # seq(0.01, 10, length.out = 3) => 5
#   )
# summary(svm.linear.tune)
# plot(svm.linear.tune)

svm.linear <- svm(
  hhsize ~ ., data = d$train, kernel = "linear",
  cost = 1 # 7.8 # svm.linear.tune$best.parameters$cost
  )

performance.svm.linear <- list(
  train = list(
    accuracy = mean(d$train$hhsize == predict(svm.linear)),
    confusion = table(true = d$train$hhsize, predict = predict(svm.linear))
    ),
  test = list(
    accuracy = mean(d$test$hhsize == predict(svm.linear, d$test)),
    confusion = table(true = d$test$hhsize, predict = predict(svm.linear, d$test))
  )
)
performance.svm.linear
```

```
## $train
## $train$accuracy
## [1] 0.5655738
##
## $train$confusion
##          predict
```

```
## true      hhsize1 hhsize2 hhsize3 hhsize4 hhsize5
##   hhsize1     255     132      13      17       0
##   hhsize2      97     286      11      13       2
##   hhsize3      24      68      49      19       4
##   hhsize4      13      47      18      80       5
##   hhsize5       6       7      10      24      20
##
##
## $test
## $test$accuracy
## [1] 0.4516539
##
## $test$confusion
##          predict
## true      hhsize1 hhsize2 hhsize3 hhsize4 hhsize5
##   hhsize1     125     123      16      12       0
##   hhsize2      71     154      12       8       2
##   hhsize3      20      41      23      21       6
##   hhsize4       6      31       8      45      15
##   hhsize5       5       7       6      21       8
```

Radial Kernel

```
# svm.radial.tune <- tune.svm(
#   hhsize ~ ., data = d$train, kernel = "radial",
#   cost  = seq(0.01, 20, length.out = 7),
#   gamma = seq(0.01, 20, length.out = 7)
# )
# summary(svm.radial.tune)
# plot(svm.radial.tune)

svm.radial <- svm(
  hhsize ~ ., data = d$train, kernel = "radial",
  gamma = 0.01, # svm.radial.tune$best.parameters$gamma
  cost = 15     # svm.radial.tune$best.parameters$cost
)

performance.svm.radial <- list(
  train = list(
    accuracy = mean(d$train$hhsize == predict(svm.radial)),
    confusion = table(true = d$train$hhsize, predict = predict(svm.radial))
  ),
  test = list(
    accuracy = mean(d$test$hhsize == predict(svm.radial, d$test)),
    confusion = table(true = d$test$hhsize, predict = predict(svm.radial, d$test))
  )
)
performance.svm.radial
```

```
## $train
## $train$accuracy
## [1] 0.9434426
```

```
##
## $train$confusion
##          predict
## true      hhsize1 hhsize2 hhsize3 hhsize4 hhsize5
##   hhsize1     400      15       0       2       0
##   hhsize2      15     392       1       1       0
##   hhsize3       6      10     146       2       0
##   hhsize4       1       9       2     151       0
##   hhsize5       1       4       0       0      62
##
##
## $test
## $test$accuracy
## [1] 0.4465649
##
## $test$confusion
##          predict
## true      hhsize1 hhsize2 hhsize3 hhsize4 hhsize5
##   hhsize1     154     100      14       7       1
##   hhsize2      88     133      16       7       3
##   hhsize3      16      45      16      24      10
##   hhsize4      12      24      16      40      13
##   hhsize5       4       8       4      23       8
```

```
# library('psych')
# cohen.kappa(performance.svm.radial$train$confusion)
# cohen.kappa(performance.svm.radial$test$confusion)
```

Random Forest

```
library(randomForest)
```

```
## randomForest 4.6-14
```

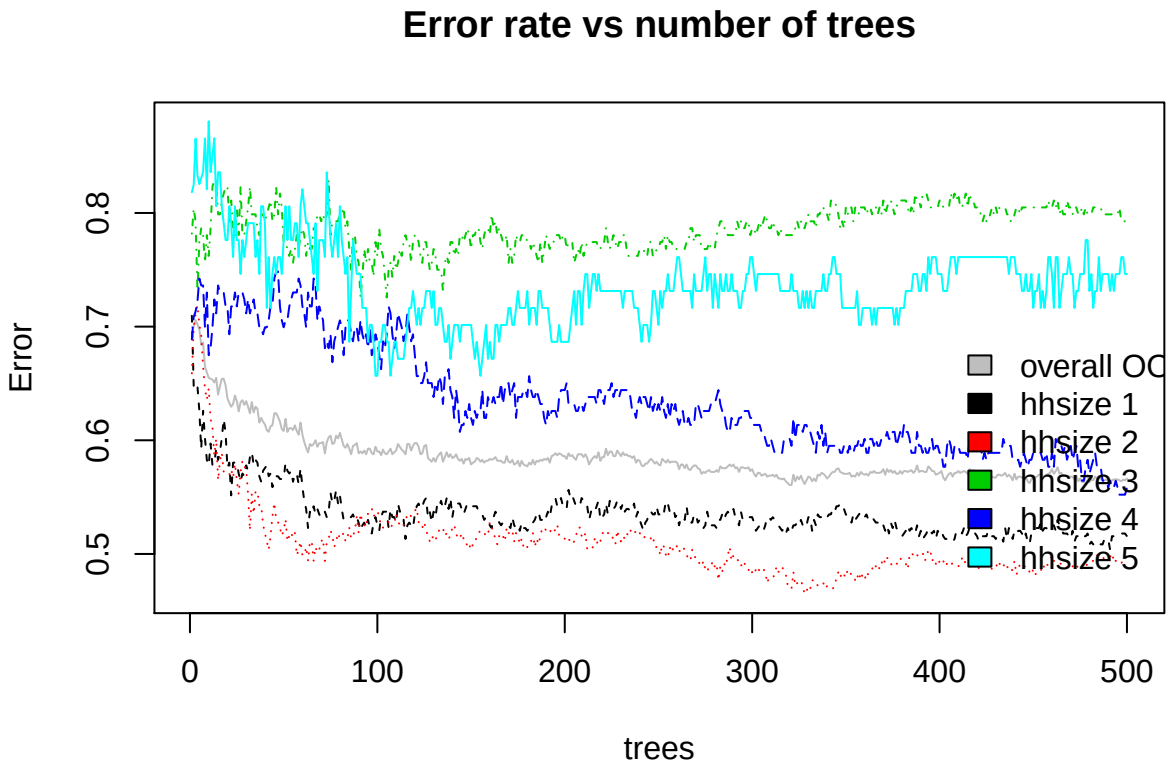
```
## Type rfNews() to see new features/changes/bug fixes.
```

```
rf <- randomForest(
  hhsize ~ ., data = d$train, imortance = TRUE,
  strata = d$train$hhsize, sampsize = rep(min(table(d$train$hhsize)), 5) # 67
)

performance.rf <- list(
  train = list(
    accuracy = mean(d$train$hhsize == predict(rf)),
    # confusion = table(true = d$train$hhsize, predict = predict(rf))
    confusion = rf$confusion # the same
  ),
  test = list(
    accuracy = mean(d$test$hhsize == predict(rf, d$test)),
    confusion = table(true = d$test$hhsize, predict = predict(rf, d$test))
```

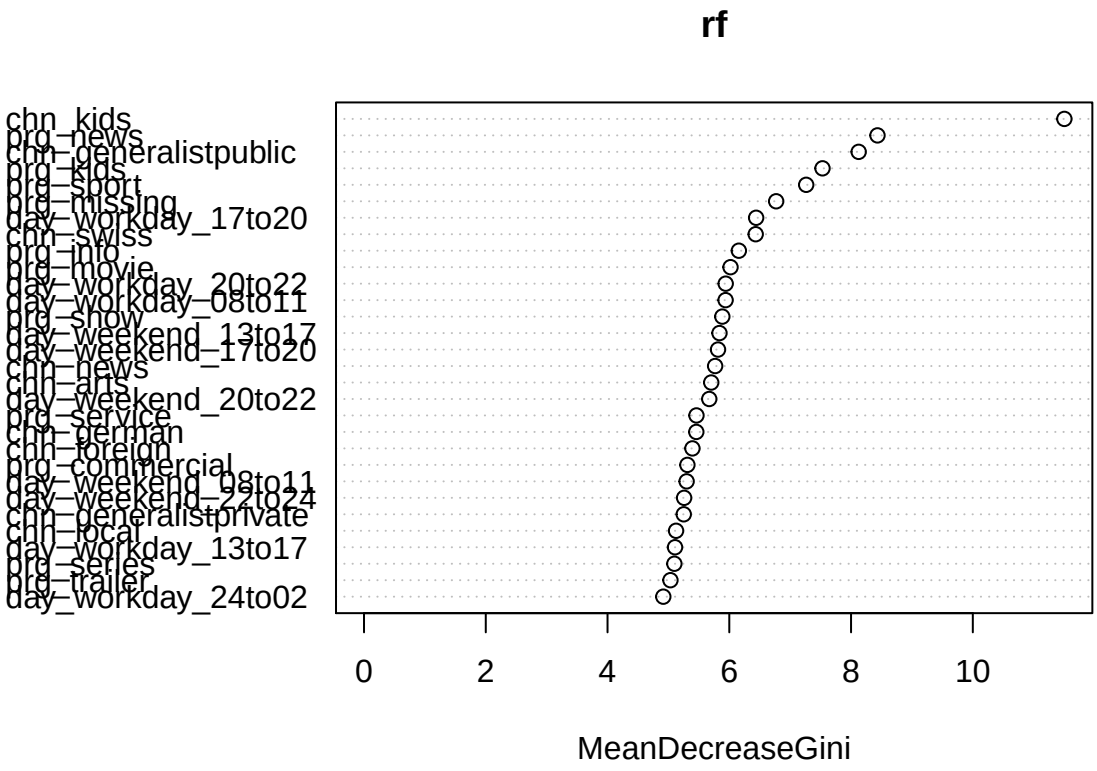
```
  )
)
performance.rf
```

```
## $train
## $train$accuracy
## [1] 0.4336066
##
## $train$confusion
##         hhsize1 hhsize2 hhsize3 hhsize4 hhsize5 class.error
## hhsize1     202     121      54      30      10   0.5155875
## hhsize2     106     205      47      42       9   0.4987775
## hhsize3      21      43      33      44      23   0.7987805
## hhsize4      13      24      31      72      23   0.5582822
## hhsize5       2       6      16      26      17   0.7462687
##
##
## $test
## $test$accuracy
## [1] 0.4351145
##
## $test$confusion
##          predict
## true      hhsize1 hhsize2 hhsize3 hhsize4 hhsize5
##   hhsize1     136      86      33      18       3
##   hhsize2      65     125      31      20       6
##   hhsize3      14      26      22      31      18
##   hhsize4       5      19      18      45      18
##   hhsize5       2       4       4      25      12
```

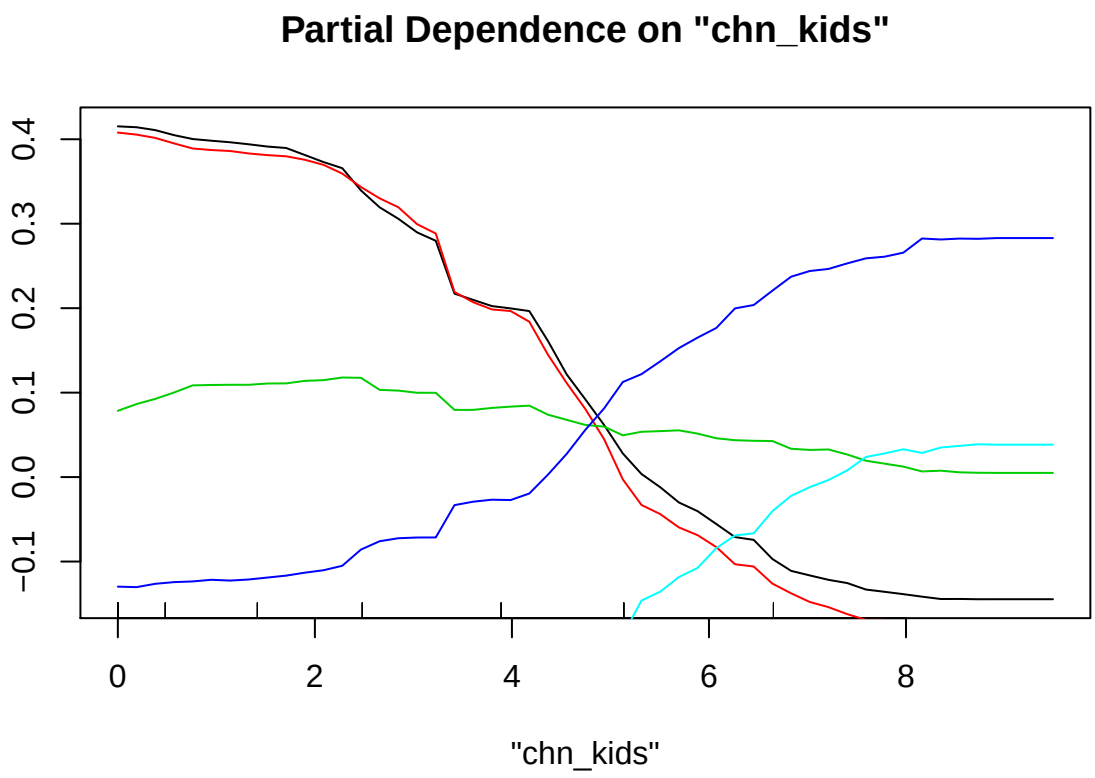
```
# cohen.kappa(performance.rf$train$confusion[,-6])
# cohen.kappa(performance.rf$test$confusion)

plot(rf, main = 'Error rate vs number of trees', col = c(8,1:5))
legend(400, 0.7, c('overall OOB', paste('hhsize', 1:5)), bty = 'n',
       fill = c(8,1:5))
```



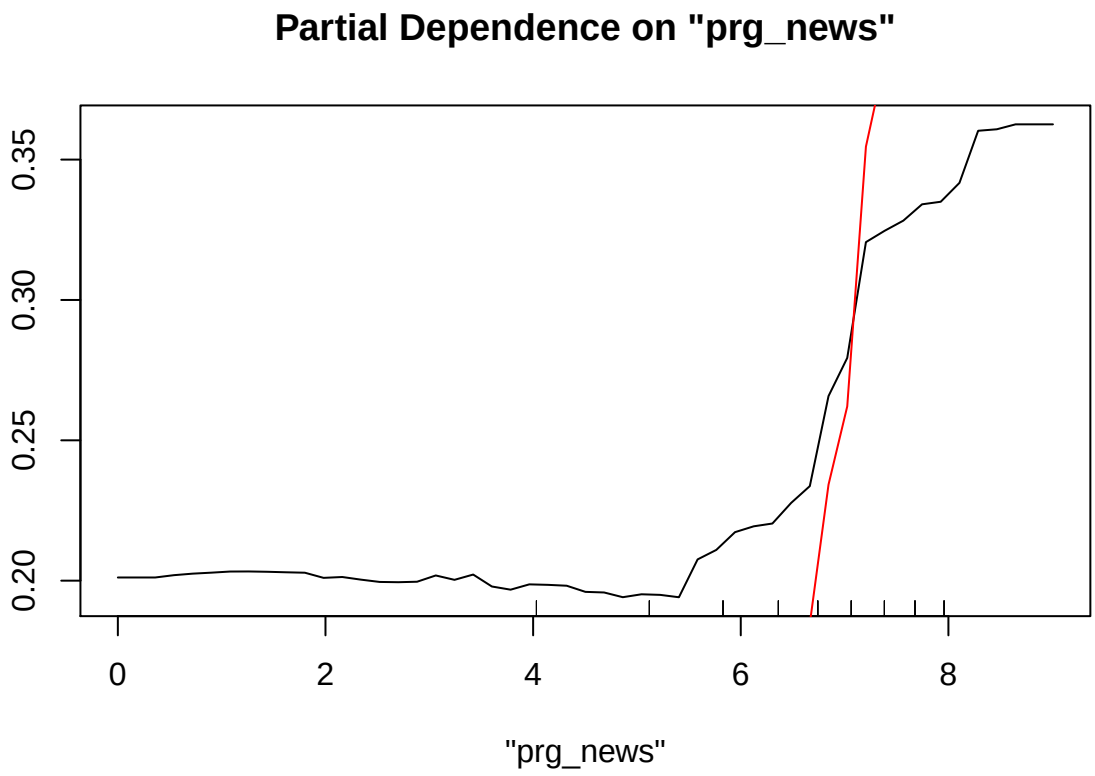

```
varImpPlot(rf)
```




```r
partialPlot(rf, d$train, x.var = "chn_kids")
for (i in 2:5)
  partialPlot(rf, d$train, x.var = "chn_kids", add = TRUE, col = i,
              which.class = levels(d$train$hhsize)[i]
  )
```




```
partialPlot(rf, d$train, x.var = "prg_news")
for (i in 2:5)
  partialPlot(rf, d$train, x.var = "prg_news", add = TRUE, col = i,
              which.class = levels(d$train$hhsize)[i]
  )
```


Partial Dependence on "prg_news"
0.20
0.25
0.30
0.35
0
2
4
6
8
"prg_news"