Algorithm

```r
library(tidyverse)
library(haven)
library(palmerpenguins)
library(gtsummary)
library(caret)
library(finalfit)
library(ranger)
library(kernlab)
library(MLeval)
library(bangladesh)
library(viridisLite)
library(viridis)
```

Import dataset

```r
#PR <- read_spss("BDPR7RFL.SAV")
hr <- read_sav("BDHR7RFL.SAV")

#PR_df <-PR |>
# select(HV226, HV206, HV208, HV243A, HV221, HV209, HV242, HV025, HV220, HV219, HV106,
# rename(fuel= HV226, Electricity = HV206,
      #   Television = HV208, Mobile.phone = HV243A, Landline = HV221,
       #  Refrigerator = HV209, separate.kitchen = HV242, residence = HV025, age = HV220,
        # sex = HV219, education = HV106, marital.status = HV115, work.status = SH13,
         #mutate(Cooking.fuel = cut(fuel,
         #                          breaks = c(1,5,10),
         #                          labels = c("Clean Fuel", "Not Clean"),
         #                         right = TRUE))


hr_df <- hr |>
  select(HV226, HV206, HV208, HV243A, HV221, HV209, HV242,HV241, HV025, HV220,
         HV219, `HV106$01`, HV024, `HV115$01`, `SH13$01`, HV270, HV009) |>
  ## Renaming Variable
  rename(fuel= HV226, Electricity = HV206, Television = HV208,
         Mobile.phone = HV243A, Landline = HV221, Refrigerator = HV209,
         separate.kitchen = HV242,  Kitchen = HV241, residence = HV025,
```

```r
         age = HV220, Division = HV024,
         sex = HV219, education = `HV106$01`, marital.status = `HV115$01`,
         work.status = `SH13$01`, Wealth.index = HV270, Fsize = HV009) |>

        mutate(cooking.fuel = case_when(fuel <= 5 ~ 1,
                                        ## Categories fuel into two categories
                                        fuel == 6 ~ 0,
                                        ## 1 = Clean, 0 = Unclean
                                        fuel == 7 ~ 0,
                                        fuel == 8 ~ 0,
                                        fuel == 9 ~ 0,
                                        fuel == 10 ~ 0,
                                        fuel == 11 ~ 0,
                                        TRUE ~ NA),
               sex = case_when(sex == 2 ~ 0,
                               sex == 1 ~ 1),
               residence = case_when(residence == 1 ~ 1,
                                     # 1 = Urban 0 = Rural
                                     residence ==2 ~ 0),
               marital.status = case_when(marital.status == 1 ~ 1,
                                          marital.status == 2 ~ 1,
                                          # 1 = Yes
                                          marital.status == 0 ~ 0,
                                          marital.status == 3 ~ 0,
                                          marital.status == 4 ~ 0,
                                          marital.status == 5 ~ 0),
                                          # 0 = No

               separate.kitchen = if_else(separate.kitchen == 1, 1, 0, missing = 1),
                                          # 0 = No 1 = Yes

               tele.communication = case_when(Landline == 1 | Mobile.phone == 1 ~ 1,
                                              TRUE ~ 0),
                                          # 1 = Yes 0 = NO
               Family.size = case_when(Fsize < 3 ~ 0,
                                       TRUE ~ 1)
                                          # 1 = Large Family 0 = Small family

        )

table(hr_df$separate.kitchen)
```

```

  0     1
387 19070
```

```r
table(hr_df$tele.communication)
```

```
   0     1
 1042 18415
```

```
head(hr_df)
```

```
# A tibble: 6 x 20
  fuel                    Electricity Television Mobile.phone Landline Refrigerator
  <dbl+lbl>               <dbl+lbl>   <dbl+lbl>  <dbl+lbl>    <dbl+lb> <dbl+lbl>
1  8 [Wood]               0 [No]      0 [No]     1 [Yes]      0 [No]   0 [No]
2  8 [Wood]               0 [No]      0 [No]     1 [Yes]      0 [No]   0 [No]
3 11 [Animal dung]        0 [No]      0 [No]     1 [Yes]      0 [No]   0 [No]
4  8 [Wood]               0 [No]      0 [No]     1 [Yes]      0 [No]   0 [No]
5  8 [Wood]               0 [No]      0 [No]     1 [Yes]      0 [No]   0 [No]
6 10 [Agricultural cr~ 0 [No]      0 [No]     1 [Yes]      0 [No]   0 [No]
# i 14 more variables: separate.kitchen <dbl>, Kitchen <dbl+lbl>,
#   residence <dbl>, age <dbl+lbl>, sex <dbl>, education <dbl+lbl>,
#   Division <dbl+lbl>, marital.status <dbl>, work.status <dbl+lbl>,
#   Wealth.index <dbl+lbl>, Fsize <dbl>, cooking.fuel <dbl>,
#   tele.communication <dbl>, Family.size <dbl>
```

Multidimentional Energy Poverty Index:

```
hr_ep <-  hr_df |>
  select(cooking.fuel, Electricity, Television, tele.communication ,
         Refrigerator, separate.kitchen,residence, age, sex, education,
         marital.status, work.status, Wealth.index, Family.size, Division) |>
  mutate(cooking.fuel = case_when( cooking.fuel == 0 ~ 1,
                                   cooking.fuel == 1 ~ 0,
                                   # 1 = Do not use clean fuel
                                   TRUE ~ NA),
         Electricity = case_when( Electricity == 0 ~ 1,
                                  # 1 = Do not have Electricity
                                  Electricity == 1 ~ 0),

         Television = case_when( Television == 0 ~ 1,
                                 # 1 = Do not have Television
                                 Television == 1 ~ 0),

         tele.communication = case_when( tele.communication == 0 ~ 1,
                                 # 1 = Do not have a landline or mobile phone
                                         tele.communication == 1 ~ 0),

         Refrigerator = case_when( Refrigerator == 0 ~ 1,
                                   # 1 = Do not have Refrigerator
                                   Refrigerator == 1 ~ 0),

         separate.kitchen = case_when( separate.kitchen == 0 ~ 1,
```

```r
                                                    # 1 = Do not have separate.kitchen
                                                    separate.kitchen == 1 ~ 0),

            ) |>
    na.omit()

head(hr_ep)
```

```
# A tibble: 6 x 15
  cooking.fuel Electricity Television tele.communication Refrigerator
         <dbl>       <dbl>      <dbl>              <dbl>        <dbl>
1            1           1          1                  0            1
2            1           1          1                  0            1
3            1           1          1                  0            1
4            1           1          1                  0            1
5            1           1          1                  0            1
6            1           1          1                  0            1
# i 10 more variables: separate.kitchen <dbl>, residence <dbl>, age <dbl+lbl>,
#   sex <dbl>, education <dbl+lbl>, marital.status <dbl>,
#   work.status <dbl+lbl>, Wealth.index <dbl+lbl>, Family.size <dbl>,
#   Division <dbl+lbl>
```

```r
table(hr_ep$Family.size)
```

```

    0     1 
 2223 17194 
```

```r
table(hr_ep$cooking.fuel)
```

```

    0     1 
 3983 15434 
```

```r
table(hr_df$cooking.fuel)
```

```

    0     1 
15435  3983 
```

```r
table(hr_ep$Electricity)
```

```

    0     1 
15779  3638 
```

```r
table(hr_df$Electricity)
```

```
   0     1 
 3643 15814 
```

```r
table(hr_ep$Television)
```

```
   0     1 
 9218 10199 
```

```r
table(hr_df$Television)
```

```
    0     1 
10223  9234 
```

```r
table(hr_ep$tele.communication)
```

```
    0     1 
18379  1038 
```

```r
table(hr_df$tele.communication)
```

```
   0     1 
 1042 18415 
```

```r
table(hr_ep$Refrigerator)
```

```
   0     1 
 5734 13683 
```

```r
table(hr_df$Refrigerator)
```

```
    0     1 
13711  5746 
```

```r
table(hr_ep$separate.kitchen)
```

```
    0     1 
19031   386 
```

```r
table(hr_df$separate.kitchen)
```

```
    0     1 
  387 19070 
```

```r
w <-c(0.2, 0.2, 0.15, 0.15, 0.15, 0.15)

y1 = as.matrix(hr_ep$cooking.fuel)*(w[1])
y2 = as.matrix(hr_ep$Electricity)*w[2]
y3 = as.matrix(hr_ep$Television)*w[3]
y4 = as.matrix(hr_ep$tele.communication)*w[4]
y5 = as.matrix(hr_ep$Refrigerator)*w[5]
y6 = as.matrix(hr_ep$separate.kitchen)*w[6]

Y = as.matrix(cbind(y1,y2,y3,y4,y5,y6),ncol = 6)

head(Y)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]  0.2  0.2 0.15    0 0.15    0
[2,]  0.2  0.2 0.15    0 0.15    0
[3,]  0.2  0.2 0.15    0 0.15    0
[4,]  0.2  0.2 0.15    0 0.15    0
[5,]  0.2  0.2 0.15    0 0.15    0
[6,]  0.2  0.2 0.15    0 0.15    0
```

```r
#C = Y * as.vector(w)
C = Y

Energy <- C %>% as_tibble() %>%
   mutate(deprivation_score = rowSums(across(where(is.numeric))),

          deprived = case_when( deprivation_score >= 0.35 ~ deprivation_score,
                                deprivation_score < 0.35 ~ 0),

          energy_poor = case_when(deprived == 0 ~ 0,
                             TRUE ~ 1))
```

```
Warning: The `x` argument of `as_tibble.matrix()` must have unique column names if
`.name_repair` is omitted as of tibble 2.0.0.
i Using compatibility `.name_repair`.
```

```r
head(C)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]  0.2  0.2 0.15    0 0.15    0
[2,]  0.2  0.2 0.15    0 0.15    0
[3,]  0.2  0.2 0.15    0 0.15    0
[4,]  0.2  0.2 0.15    0 0.15    0
[5,]  0.2  0.2 0.15    0 0.15    0
[6,]  0.2  0.2 0.15    0 0.15    0
```

```
head(Y)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]  0.2  0.2 0.15    0 0.15    0
[2,]  0.2  0.2 0.15    0 0.15    0
[3,]  0.2  0.2 0.15    0 0.15    0
[4,]  0.2  0.2 0.15    0 0.15    0
[5,]  0.2  0.2 0.15    0 0.15    0
[6,]  0.2  0.2 0.15    0 0.15    0
```

```
head(Energy)
```

```
# A tibble: 6 x 9
     V1    V2    V3    V4    V5    V6 deprivation_score deprived energy_poor
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>             <dbl>    <dbl>       <dbl>
1   0.2   0.2  0.15     0  0.15     0               0.7      0.7           1
2   0.2   0.2  0.15     0  0.15     0               0.7      0.7           1
3   0.2   0.2  0.15     0  0.15     0               0.7      0.7           1
4   0.2   0.2  0.15     0  0.15     0               0.7      0.7           1
5   0.2   0.2  0.15     0  0.15     0               0.7      0.7           1
6   0.2   0.2  0.15     0  0.15     0               0.7      0.7           1
```

```
head.count = sum(Energy$energy_poor==1)/length(Energy$energy_poor)
head.count
```

```
[1] 0.675336
```

```
intensity = sum(Energy$deprived)/sum(Energy$energy_poor==1);intensity
```

```
[1] 0.5199115
```

```
MEPI = head.count * intensity;MEPI
```

```
[1] 0.351115
```

```r
table(Energy$energy_poor)
```

```
   0     1
6304 13113
```

Multidimensional Energy Poverty Index by Division: Barisal

```r
hr_barisal <- hr_ep |>
  select(-c(residence, age, sex, education,
         marital.status, work.status, Wealth.index, Family.size)) |>
  filter(Division == 1) |>
  select(-Division)

w1 <- c(0.2, 0.2, 0.15, 0.15, 0.15, 0.15)

z = as.matrix(hr_barisal)


## Using for loop for creating deprivation score matrix

for (i in 1:nrow(z)) {
 Y_barisal = matrix(nrow = nrow(z), ncol = ncol(z))
for (j in 1:length(w1)) {

  Y_barisal[,j] = z[,j] * w1[j]

}
}

head(Y_barisal)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]  0.2  0.2 0.15    0 0.15    0
[2,]  0.2  0.2 0.15    0 0.15    0
[3,]  0.2  0.2 0.15    0 0.15    0
[4,]  0.2  0.2 0.15    0 0.15    0
[5,]  0.2  0.2 0.15    0 0.15    0
[6,]  0.2  0.2 0.15    0 0.15    0
```

```r
Energy_barisal <- Y_barisal %>% as_tibble() %>%
   mutate(deprivation_score = rowSums(across(where(is.numeric))),
          deprived = case_when(deprivation_score >= 0.35 ~ deprivation_score,
                               deprivation_score < 0.35 ~ 0),

          energy_poor = case_when(deprived == 0 ~ 0,
```

```
                                              TRUE ~ 1))
head.count.barisal = sum(Energy_barisal$energy_poor==1)/length(Energy_barisal$energy_poor)
head.count.barisal
```

```
[1] 0.7864838
```

```
intensity.barisal = sum(Energy_barisal$deprived)/sum(Energy_barisal$energy_poor==1)
intensity.barisal
```

```
[1] 0.5641034
```

```
MEPI.barisal = head.count.barisal * intensity.barisal
MEPI.barisal
```

```
[1] 0.4436582
```

Multidimensional Energy Poverty Index by Division: Chittagong

```
hr_Chittagong <- hr_ep |>
  select(-c(residence, age, sex, education,
         marital.status, work.status, Wealth.index, Family.size)) |>
  filter(Division == 2) |>
  select(-Division)

w2 <- c(0.2, 0.2, 0.15, 0.15, 0.15, 0.15)

z1 = as.matrix(hr_Chittagong)


## Using for loop for creating deprivation score matrix

for (i in 1:nrow(z1)) {
 Y_Chittagong = matrix(nrow = nrow(z1), ncol = ncol(z1))
for (j in 1:length(w2)) {

  Y_Chittagong[,j] = z1[,j] * w2[j]

}
}

head(Y_Chittagong)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]  0.2  0.2 0.15 0.15 0.15    0
[2,]  0.2  0.2 0.15 0.15 0.15    0
[3,]  0.2  0.2 0.15 0.00 0.15    0
[4,]  0.2  0.2 0.15 0.00 0.15    0
[5,]  0.0  0.0 0.00 0.00 0.00    0
[6,]  0.2  0.0 0.15 0.00 0.00    0
```

```
Energy_Chittagong <- Y_Chittagong %>% as_tibble() %>%
   mutate(deprivation_score = rowSums(across(where(is.numeric))),
          deprived = case_when(deprivation_score >= 0.35 ~ deprivation_score,
                               deprivation_score < 0.35 ~ 0),

          energy_poor = case_when(deprived == 0 ~ 0,
                                              TRUE ~ 1))
hc.Chittagong = sum(Energy_Chittagong$energy_poor==1)/length(Energy_Chittagong$energy_poor)
hc.Chittagong
```

```
[1] 0.5543355
```

```
intensity.Chittagong = sum(Energy_Chittagong$deprived)/sum(Energy_Chittagong$energy_poor==1)
intensity.Chittagong
```

```
[1] 0.5197404
```

```
MEPI.Chittagong = hc.Chittagong * intensity.Chittagong
MEPI.Chittagong
```

```
[1] 0.2881106
```

Multidimensional Energy Poverty Index by Division: Dhaka

```
hr_Dhaka <- hr_ep |>
  select(-c(residence, age, sex, education,
         marital.status, work.status, Wealth.index, Family.size)) |>
  filter(Division == 3) |>
  select(-Division)

w3 <- c(0.2, 0.2, 0.15, 0.15, 0.15, 0.15)

z2 = as.matrix(hr_Dhaka)
```

```r
## Using for loop for creating deprivation score matrix

for (i in 1:nrow(z2)) {
 Y_Dhaka = matrix(nrow = nrow(z2), ncol = ncol(z2))
for (j in 1:length(w3)) {

  Y_Dhaka[,j] = z2[,j] * w3[j]

}
}

head(Y_Dhaka)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]  0.2    0 0.15    0    0 0.00
[2,]  0.0    0 0.00    0    0 0.15
[3,]  0.2    0 0.00    0    0 0.00
[4,]  0.2    0 0.00    0    0 0.00
[5,]  0.2    0 0.00    0    0 0.00
[6,]  0.2    0 0.00    0    0 0.00
```

```r
Energy_Dhaka <- Y_Dhaka %>% as_tibble() %>%
   mutate(deprivation_score = rowSums(across(where(is.numeric))),
          deprived = case_when(deprivation_score >= 0.35 ~ deprivation_score,
                               deprivation_score < 0.35 ~ 0),

          energy_poor = case_when(deprived == 0 ~ 0,
                                            TRUE ~ 1))
hc.Dhaka = sum(Energy_Dhaka$energy_poor==1)/length(Energy_Dhaka$energy_poor)
hc.Dhaka
```

```
[1] 0.3560606
```

```r
intensity.Dhaka = sum(Energy_Dhaka$deprived)/sum(Energy_Dhaka$energy_poor==1)
intensity.Dhaka
```

```
[1] 0.4955996
```

```r
MEPI.Dhaka = hc.Dhaka * intensity.Dhaka
MEPI.Dhaka
```

```
[1] 0.1764635
```

Multidimensional Energy Poverty Index by Division: Khulna

```r
hr_Khulna <- hr_ep |>
  select(-c(residence, age, sex, education,
         marital.status, work.status, Wealth.index, Family.size)) |>
  filter(Division == 4) |>
  select(-Division)

w4 <- c(0.2, 0.2, 0.15, 0.15, 0.15, 0.15)

z3 = as.matrix(hr_Khulna)


## Using for loop for creating deprivation score matrix

for (i in 1:nrow(z3)) {
 Y_Khulna = matrix(nrow = nrow(z3), ncol = ncol(z3))
for (j in 1:length(w4)) {

  Y_Khulna[,j] = z3[,j] * w4[j]

}
}

head(Y_Khulna)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]  0.2    0 0.00    0 0.15 0.00
[2,]  0.2    0 0.00    0 0.15 0.00
[3,]  0.2    0 0.00    0 0.15 0.00
[4,]  0.0    0 0.00    0 0.00 0.15
[5,]  0.2    0 0.15    0 0.15 0.00
[6,]  0.2    0 0.00    0 0.15 0.00
```

```r
Energy_Khulna <- Y_Khulna %>% as_tibble() %>%
   mutate(deprivation_score = rowSums(across(where(is.numeric))),
          deprived = case_when(deprivation_score >= 0.35 ~ deprivation_score,
                               deprivation_score < 0.35 ~ 0),

          energy_poor = case_when(deprived == 0 ~ 0,
                                  TRUE ~ 1))
hc.Khulna = sum(Energy_Khulna$energy_poor==1)/length(Energy_Khulna$energy_poor)
hc.Khulna
```

```
[1] 0.7399442
```

```r
intensity.Khulna = sum(Energy_Khulna$deprived)/sum(Energy_Khulna$energy_poor==1)
intensity.Khulna
```

```
[1] 0.4823197
```

```
MEPI.Khulna = hc.Khulna * intensity.Khulna
MEPI.Khulna
```

```
[1] 0.3568897
```

Multidimensional Energy Poverty Index by Division: Mymensingh

```
hr_Mym <- hr_ep |>
  select(-c(residence, age, sex, education,
         marital.status, work.status, Wealth.index, Family.size)) |>
  filter(Division == 5) |>
  select(-Division)

w5 <- c(0.2, 0.2, 0.15, 0.15, 0.15, 0.15)

z4 = as.matrix(hr_Mym)


## Using for loop for creating deprivation score matrix

for (i in 1:nrow(z4)) {
 Y_Mym = matrix(nrow = nrow(z4), ncol = ncol(z4))
for (j in 1:length(w5)) {

  Y_Mym[,j] = z4[,j] * w5[j]

}
}

head(Y_Mym)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]  0.2  0.2 0.15 0.00 0.15    0
[2,]  0.2  0.2 0.15 0.00 0.15    0
[3,]  0.2  0.2 0.15 0.00 0.15    0
[4,]  0.2  0.2 0.15 0.15 0.15    0
[5,]  0.2  0.2 0.15 0.00 0.15    0
[6,]  0.2  0.2 0.15 0.00 0.15    0
```

```
Energy_Mym <- Y_Mym %>% as_tibble() %>%
   mutate(deprivation_score = rowSums(across(where(is.numeric))),
          deprived = case_when(deprivation_score >= 0.35 ~ deprivation_score,
```

```
                                  deprivation_score < 0.35 ~ 0),

          energy_poor = case_when(deprived == 0 ~ 0,
                                              TRUE ~ 1))
hc.Mym = sum(Energy_Mym$energy_poor==1)/length(Energy_Mym$energy_poor)
hc.Mym
```

```
[1] 0.7811935
```

```
intensity.Mym = sum(Energy_Mym$deprived)/sum(Energy_Mym$energy_poor==1)
intensity.Mym
```

```
[1] 0.529919
```

```
MEPI.Mym = hc.Mym * intensity.Mym
MEPI.Mym
```

```
[1] 0.4139693
```

Multidimensional Energy Poverty Index by Division: Rajshahi

```
hr_Rajshahi <- hr_ep |>
  select(-c(residence, age, sex, education,
         marital.status, work.status, Wealth.index, Family.size)) |>
  filter(Division == 6) |>
  select(-Division)

w6 <- c(0.2, 0.2, 0.15, 0.15, 0.15, 0.15)

z5 = as.matrix(hr_Rajshahi)


## Using for loop for creating deprivation score matrix

for (i in 1:nrow(z5)) {
 Y_Rajshahi = matrix(nrow = nrow(z5), ncol = ncol(z5))
for (j in 1:length(w6)) {

  Y_Rajshahi[,j] = z5[,j] * w6[j]

}
}
```

```
head(Y_Rajshahi)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]  0.2    0    0 0.00 0.00 0.00
[2,]  0.2    0    0 0.15 0.15 0.00
[3,]  0.2    0    0 0.00 0.15 0.00
[4,]  0.2    0    0 0.00 0.15 0.00
[5,]  0.2    0    0 0.00 0.15 0.00
[6,]  0.0    0    0 0.00 0.15 0.15
```

```
Energy_Rajshahi <- Y_Rajshahi %>% as_tibble() %>%
   mutate(deprivation_score = rowSums(across(where(is.numeric))),
          deprived = case_when(deprivation_score >= 0.35 ~ deprivation_score,
                               deprivation_score < 0.35 ~ 0),

          energy_poor = case_when(deprived == 0 ~ 0,
                                  TRUE ~ 1))
hc.Rajshahi = sum(Energy_Rajshahi$energy_poor==1)/length(Energy_Rajshahi$energy_poor)
hc.Rajshahi
```

```
[1] 0.7329168
```

```
intensity.Rajshahi = sum(Energy_Rajshahi$deprived)/sum(Energy_Rajshahi$energy_poor==1)
intensity.Rajshahi
```

```
[1] 0.4838039
```

```
MEPI.Rajshahi = hc.Rajshahi * intensity.Rajshahi
MEPI.Rajshahi
```

```
[1] 0.3545881
```

Multidimensional Energy Poverty Index by Division: Rangpur

```
hr_Rangpur <- hr_ep |>
  select(-c(residence, age, sex, education,
         marital.status, work.status, Wealth.index, Family.size)) |>
  filter(Division == 7) |>
  select(-Division)

w7 <- c(0.2, 0.2, 0.15, 0.15, 0.15, 0.15)
```

```r
z6 = as.matrix(hr_Rangpur)


## Using for loop for creating deprivation score matrix

for (i in 1:nrow(z6)) {
 Y_Rangpur = matrix(nrow = nrow(z6), ncol = ncol(z6))
for (j in 1:length(w7)) {

  Y_Rangpur[,j] = z6[,j] * w7[j]

}
}

head(Y_Rangpur)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]  0.2  0.2 0.15    0 0.15    0
[2,]  0.2  0.2 0.15    0 0.15    0
[3,]  0.2  0.2 0.15    0 0.15    0
[4,]  0.2  0.2 0.15    0 0.15    0
[5,]  0.2  0.2 0.15    0 0.15    0
[6,]  0.2  0.2 0.15    0 0.15    0
```

```r
Energy_Rangpur <- Y_Rangpur %>% as_tibble() %>%
   mutate(deprivation_score = rowSums(across(where(is.numeric))),
          deprived = case_when(deprivation_score >= 0.35 ~ deprivation_score,
                               deprivation_score < 0.35 ~ 0),

          energy_poor = case_when(deprived == 0 ~ 0,
                                           TRUE ~ 1))
hc.Rangpur = sum(Energy_Rangpur$energy_poor==1)/length(Energy_Rangpur$energy_poor)
hc.Rangpur
```

```
[1] 0.8505887
```

```r
intensity.Rangpur = sum(Energy_Rangpur$deprived)/sum(Energy_Rangpur$energy_poor==1)
intensity.Rangpur
```

```
[1] 0.5452745
```

```r
MEPI.Rangpur = hc.Rangpur * intensity.Rangpur
MEPI.Rangpur
```

```
[1] 0.4638043
```

Multidimensional Energy Poverty Index by Division: Sylhet

```
hr_Sylhet <- hr_ep |>
  select(-c(residence, age, sex, education,
         marital.status, work.status, Wealth.index, Family.size)) |>
  filter(Division == 8) |>
  select(-Division)

w8 <- c(0.2, 0.2, 0.15, 0.15, 0.15, 0.15)

z7 = as.matrix(hr_Sylhet)


## Using for loop for creating deprivation score matrix

for (i in 1:nrow(z7)) {
 Y_Sylhet = matrix(nrow = nrow(z7), ncol = ncol(z7))
for (j in 1:length(w8)) {

  Y_Sylhet[,j] = z7[,j] * w8[j]

}
}

head(Y_Sylhet)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]  0.2    0 0.00    0 0.00    0
[2,]  0.2    0 0.15    0 0.00    0
[3,]  0.2    0 0.00    0 0.00    0
[4,]  0.2    0 0.00    0 0.15    0
[5,]  0.2    0 0.15    0 0.15    0
[6,]  0.2    0 0.15    0 0.15    0
```

```
Energy_Sylhet <- Y_Sylhet %>% as_tibble() %>%
   mutate(deprivation_score = rowSums(across(where(is.numeric))),
          deprived = case_when(deprivation_score >= 0.35 ~ deprivation_score,
                               deprivation_score < 0.35 ~ 0),

          energy_poor = case_when(deprived == 0 ~ 0,
                                          TRUE ~ 1))
hc.Sylhet = sum(Energy_Sylhet$energy_poor==1)/length(Energy_Sylhet$energy_poor)
hc.Sylhet
```

```
[1] 0.6965915
```

```r
intensity.Sylhet = sum(Energy_Sylhet$deprived)/sum(Energy_Sylhet$energy_poor==1)
intensity.Sylhet
```

```
[1] 0.5348036
```

```r
MEPI.Sylhet = hc.Sylhet * intensity.Sylhet
MEPI.Sylhet
```

```
[1] 0.3725396
```

MEPI by Division

```r
MEPI.Value <- c(MEPI.barisal, MEPI.Chittagong, MEPI.Dhaka,
                            MEPI.Khulna, MEPI.Mym, MEPI.Rajshahi,
                            MEPI.Rangpur, MEPI.Sylhet)
Division <- c("Barisal","Chittagong", "Dhaka", "Khulna", "Mymensingh",
              "Rajshahi", "Rangpur", "Sylhet")
MEPI.Division <- data.frame(Division, MEPI.Value = round(MEPI.Value,2))
MEPI.Division <- as_tibble(MEPI.Division)
MEPI.Division
```

```
# A tibble: 8 x 2
  Division   MEPI.Value
  <chr>           <dbl>
1 Barisal          0.44
2 Chittagong       0.29
3 Dhaka            0.18
4 Khulna           0.36
5 Mymensingh       0.41
6 Rajshahi         0.35
7 Rangpur          0.46
8 Sylhet           0.37
```

```r
## For loop Experiment

a <- matrix(rbinom(42,1,.5), ncol = 6)

b <- c(0.2, 0.2, 0.15, 0.15, 0.15, 0.15)


for (i in 1:nrow(a)) {
 c = matrix(nrow = nrow(a), ncol = ncol(a))
for (j in 1:length(b)) {
```

```
    c[,j] = a[,j] * b[j]
  
  }
  }
  c
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]  0.0  0.0 0.00 0.15 0.00 0.15
[2,]  0.2  0.0 0.15 0.00 0.15 0.15
[3,]  0.2  0.2 0.15 0.15 0.15 0.15
[4,]  0.2  0.0 0.15 0.15 0.15 0.00
[5,]  0.0  0.2 0.00 0.15 0.00 0.00
[6,]  0.2  0.0 0.00 0.00 0.15 0.15
[7,]  0.0  0.2 0.15 0.00 0.15 0.00
```

```
nrow(a)
```

```
[1] 7
```

```
class(b)
```

```
[1] "numeric"
```

Graphical Representation

```
hr_ml <-hr_ep |>
  select(age, Family.size, residence, sex, education, marital.status, work.status, Wealth.index, Di
  na.omit()

hr_ml$poverty <- cbind(Energy$energy_poor)

head(hr_ml)
```

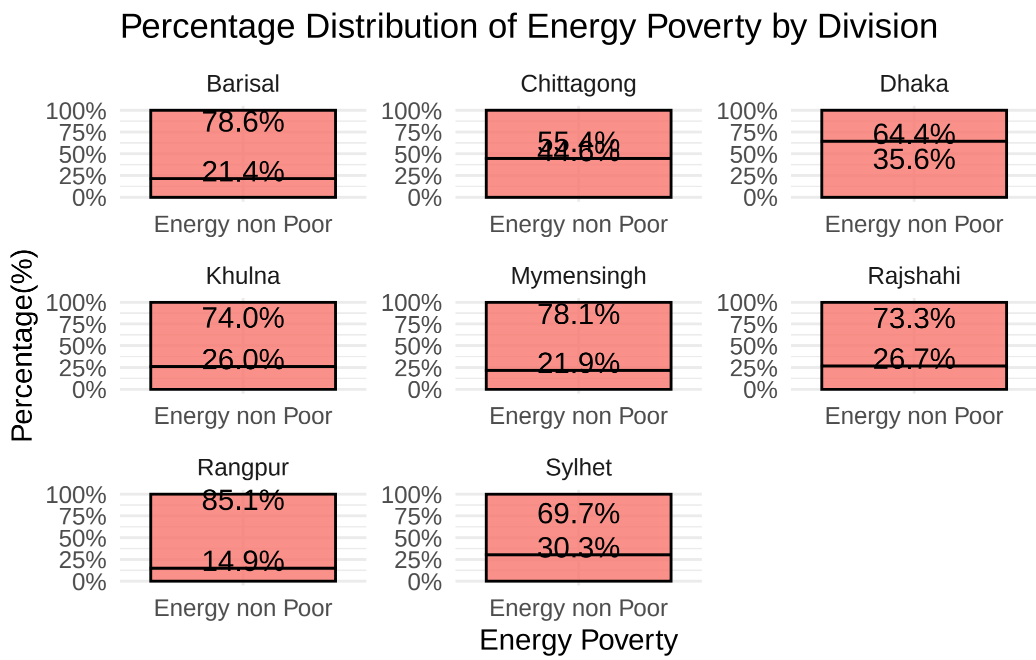
```
# A tibble: 6 x 10
  age       Family.size residence   sex education   marital.status work.status
  <dbl+lbl>       <dbl>     <dbl> <dbl> <dbl+lbl>            <dbl> <dbl+lbl>
1 45                  1         0     1 1 [Primary]              1 1 [Yes]
2 65                  1         0     1 1 [Primary]              1 1 [Yes]
3 65                  1         0     1 1 [Primary]              1 1 [Yes]
4 68                  1         0     1 1 [Primary]              1 1 [Yes]
5 42                  1         0     0 1 [Primary]              1 1 [Yes]
```

```
6 42                  1         0     1 1 [Primary]               1 1 [Yes]
# i 3 more variables: Wealth.index <dbl+lbl>, Division <dbl+lbl>,
#   poverty <dbl[,1]>
```

```r
table(hr_ml$poverty)
```

```
   0     1
 6304 13113
```

```r
hrml1 <- hr_ml |>
  mutate_at(factor, .vars = vars(Family.size:poverty)) |>
  mutate(poverty = factor(case_when(poverty == 1 ~ "Yes",
                            TRUE ~ "No")))


hrml1 |>
  pivot_longer(poverty) |>
  mutate(Ep = case_when( value == 1 ~ "Energy Poor",
                         TRUE ~ "Energy non Poor"),
         Division = case_when(Division == 1 ~ "Barisal",
                              Division == 2 ~  "Chittagong",
                              Division == 3 ~ "Dhaka",
                              Division == 4 ~ "Khulna",
                              Division == 5 ~ "Mymensingh",
                              Division == 6 ~ "Rajshahi",
                              Division == 7 ~ "Rangpur",
                              Division == 8 ~ "Sylhet"
                              )) |>
  group_by(Division)|>
  count(Division,value,Ep) |>
  mutate(prop = prop.table(n),
         label=scales::percent(prop,accuracy = 0.1)) |>
  ungroup()|>
  ggplot(aes(x = as.factor(Ep), y = prop, label = label)) +
  geom_col(aes(fill= as.factor(Ep)),alpha = 0.8, show.legend = F, col="black") +
  facet_wrap(~ Division,scales = "free") +
  geom_text(nudge_y = 0.09) +
  scale_y_continuous(labels = scales::percent_format()) +
  labs(title = "Percentage Distribution of Energy Poverty by Division",
       x = "Energy Poverty", y = "Percentage(%)") +
  theme_minimal()
```




```r
library(tmap)
```

```
Breaking News: tmap 3.x is retiring. Please test v4, e.g. with
remotes::install_github('r-tmap/tmap')
```

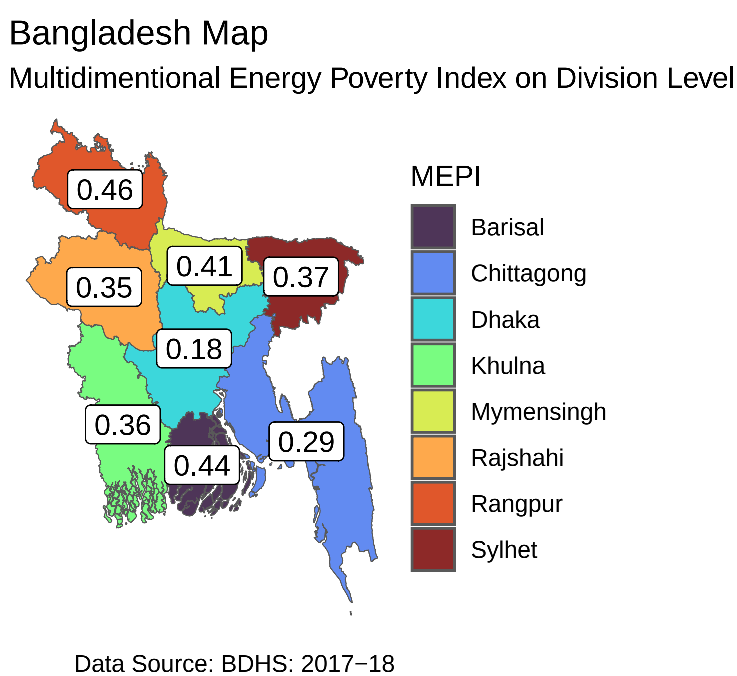
```r
MEPI.Division
```

```
# A tibble: 8 x 2
  Division   MEPI.Value
  <chr>           <dbl>
1 Barisal          0.44
2 Chittagong       0.29
3 Dhaka            0.18
4 Khulna           0.36
5 Mymensingh       0.41
6 Rajshahi         0.35
7 Rangpur          0.46
8 Sylhet           0.37
```

```r
division <- get_map("division")
division_centroids <- bangladesh::get_coordinates(level = "division")
knitr::kable(division_centroids, format = "html")
```

Division	lat	lon
Barisal	22.41889	90.34684
Chittagong	22.70692	91.73546
Dhaka	23.83870	90.24064
Khulna	22.91367	89.29437
Mymensingh	24.84675	90.38088
Rajshahi	24.58846	89.04540
Rangpur	25.77920	89.05685
Sylhet	24.71515	91.66400

```r
map_data <- dplyr::left_join(division, MEPI.Division, by = c("Division" = "Division"))

ggplot(data = map_data, aes(fill = Division,
                            label = MEPI.Value)) +
  geom_sf()+
 #geom_sf_label(aes(label = Division)) +
  geom_sf_label(fill = "white",  # override the fill from aes()
               fun.geometry = sf::st_centroid ) +
  theme_void() +
  scale_fill_viridis(name ="MEPI", discrete = TRUE, option = "H", alpha = 0.85) +
  labs(
    title = "Bangladesh Map",
    subtitle = "Multidimentional Energy Poverty Index on Division Level",
    caption = "Data Source: BDHS: 2017-18"
  )
```



```r
hrml1 |>
   tbl_summary(by = poverty) |>
  add_p()
```

Characteristic	**No**, N = 6,304	**Yes**, N = 13,113	**p-value**
Age of head of household	44 (35, 55)	45 (35, 56)	0.001
Family.size			0.001
0	655 (10%)	1,568 (12%)	
1	5,649 (90%)	11,545 (88%)	
residence			<0.001
0	2,154 (34%)	10,179 (78%)	
1	4,150 (66%)	2,934 (22%)	
sex			0.5
0	986 (16%)	1,997 (15%)	
1	5,318 (84%)	11,116 (85%)	
education			<0.001
0	858 (14%)	4,550 (35%)	
1	1,461 (23%)	4,842 (37%)	
2	2,064 (33%)	2,932 (22%)	
3	1,915 (30%)	779 (5.9%)	
8	6 (<0.1%)	10 (<0.1%)	
marital.status			<0.001
0	492 (7.8%)	1,325 (10%)	
1	5,812 (92%)	11,788 (90%)	
work.status			<0.001
0	1,181 (19%)	1,633 (12%)	
1	5,123 (81%)	11,480 (88%)	
Wealth.index			<0.001
1	0 (0%)	4,072 (31%)	
2	16 (0.3%)	3,817 (29%)	
3	582 (9.2%)	3,046 (23%)	
4	1,930 (31%)	1,859 (14%)	
5	3,776 (60%)	319 (2.4%)	
Division			<0.001
1	436 (6.9%)	1,606 (12%)	
2	1,177 (19%)	1,464 (11%)	
3	1,870 (30%)	1,034 (7.9%)	
4	653 (10%)	1,858 (14%)	
5	484 (7.7%)	1,728 (13%)	
6	684 (11%)	1,877 (14%)	
7	368 (5.8%)	2,095 (16%)	
8	632 (10%)	1,451 (11%)	

```r
names(hrml1)
```

```
 [1] "age"            "Family.size"     "residence"       "sex"
```

```
[5] "education"      "marital.status" "work.status"    "Wealth.index"
[9] "Division"       "poverty"
```

```r
dependent = names(hrml1)[10]
explanatory = names(hrml1)[-c(10)]

T <-hrml1 |>
  finalfit(dependent,explanatory, metrics = FALSE,p = TRUE,estimate_name = "Odds ratio",digits = c(3


knitr::kable(T,
             caption = "Logistic regression results predicting likelihood of Energy Poverty")
```

Table 3: Logistic regression results predicting likelihood of Energy Poverty

	Dependent: poverty		No	Yes	Odds ratio (univariable)	Odds ratio (multivariable)
1	Age of head of household	Mean (SD)	45.1 (13.7)	46.0 (14.6)	1.004 (1.002 to 1.006, TRUE=0.0001)	1.008 (1.004 to 1.011, TRUE=0.0002)
15	Family.size	0	655 (29.5)	1568 (70.5)	-	-
16		1	5649 (32.9)	11545 (67.1)	0.854 (0.775 to 0.940, TRUE=0.0013)	0.883 (0.747 to 1.044, TRUE=0.1462)
19	residence	0	2154 (17.5)	10179 (82.5)	-	-
20		1	4150 (58.6)	2934 (41.4)	0.150 (0.140 to 0.160, TRUE<0.0001)	0.558 (0.503 to 0.620, TRUE<0.0001)
21	sex	0	986 (33.1)	1997 (66.9)	-	-
22		1	5318 (32.4)	11116 (67.6)	1.032 (0.950 to 1.121, TRUE=0.4563)	1.125 (0.942 to 1.344, TRUE=0.1922)
10	education	0	858 (15.9)	4550 (84.1)	-	-
11		1	1461 (23.2)	4842 (76.8)	0.625 (0.569 to 0.686, TRUE<0.0001)	0.901 (0.778 to 1.044, TRUE=0.1657)
12		2	2064 (41.3)	2932 (58.7)	0.268 (0.244 to 0.294, TRUE<0.0001)	0.806 (0.693 to 0.936, TRUE=0.0047)
13		3	1915 (71.1)	779 (28.9)	0.077 (0.069 to 0.086, TRUE<0.0001)	0.577 (0.482 to 0.691, TRUE<0.0001)
14		8	6 (37.5)	10 (62.5)	0.314 (0.116 to 0.926, TRUE=0.0254)	0.593 (0.038 to 3.944, TRUE=0.6373)
17	marital.status	0	492 (27.1)	1325 (72.9)	-	-
18		1	5812 (33.0)	11788 (67.0)	0.753 (0.675 to 0.839, TRUE<0.0001)	0.764 (0.626 to 0.931, TRUE=0.0078)
28	work.status	0	1181 (42.0)	1633 (58.0)	-	-

	Dependent: poverty		No	Yes	Odds ratio (univariable)	Odds ratio (multivariable)
29		1	5123 (30.9)	11480 (69.1)	1.621 (1.493 to 1.759, TRUE<0.0001)	1.159 (0.988 to 1.359, TRUE=0.0695)
23	Wealth.index	1	0 (0.0)	4072 (100.0)	-	-
24		2	16 (0.4)	3817 (99.6)	0.000 (0.000 to 0.000, TRUE=0.9334)	0.000 (0.000 to 0.000, TRUE=0.9321)
25		3	582 (16.0)	3046 (84.0)	0.000 (0.000 to 0.000, TRUE=0.9154)	0.000 (0.000 to 0.000, TRUE=0.9141)
26		4	1930 (50.9)	1859 (49.1)	0.000 (0.000 to 0.000, TRUE=0.9074)	0.000 (0.000 to 0.000, TRUE=0.9065)
27		5	3776 (92.2)	319 (7.8)	0.000 (0.000 to 0.000, TRUE=0.8960)	0.000 (0.000 to 0.000, TRUE=0.8952)
2	Division	1	436 (21.4)	1606 (78.6)	-	-
3		2	1177 (44.6)	1464 (55.4)	0.338 (0.296 to 0.385, TRUE<0.0001)	0.537 (0.437 to 0.658, TRUE<0.0001)
4		3	1870 (64.4)	1034 (35.6)	0.150 (0.132 to 0.171, TRUE<0.0001)	0.282 (0.230 to 0.346, TRUE<0.0001)
5		4	653 (26.0)	1858 (74.0)	0.772 (0.672 to 0.887, TRUE=0.0003)	2.124 (1.718 to 2.625, TRUE<0.0001)
6		5	484 (21.9)	1728 (78.1)	0.969 (0.837 to 1.122, TRUE=0.6754)	0.790 (0.629 to 0.991, TRUE=0.0420)
7		6	684 (26.7)	1877 (73.3)	0.745 (0.649 to 0.854, TRUE<0.0001)	0.967 (0.783 to 1.193, TRUE=0.7549)
8		7	368 (14.9)	2095 (85.1)	1.546 (1.326 to 1.802, TRUE<0.0001)	2.307 (1.812 to 2.940, TRUE<0.0001)
9		8	632 (30.3)	1451 (69.7)	0.623 (0.541 to 0.718, TRUE<0.0001)	1.297 (1.036 to 1.623, TRUE=0.0232)

Model Building

```r
set.seed(123)
split <-  createDataPartition(hrml1$poverty, p=3/4, list = FALSE)
training <- hrml1[split,]
testing <- hrml1[-split,]
```

```r
set.seed(12345)
model <- train(poverty ~ ., data =training,
method = "svmLinear",
na.action = na.omit,
preProcess = c("scale","center"),
trControl = trainControl(method = "none"),
tune_grid = data.frame(degree=1, scale = 1, C=1))
```

```r
model.cv <- train(poverty ~ ., data = training,
method = "svmLinear",
na.action = na.omit,
preProcess = c("scale","center"),
trControl = trainControl(method = "cv",number = 10),
tune_Grid = data.frame(degree=1, scale = 1, C=1))

model.Rf <- train(poverty ~ ., data = training,
method = 'ranger',
na.action = na.omit,
preProcess = c("scale","center"),
trControl = trainControl(method = "cv",number = 10))

model.knn <- train(poverty ~ ., data = training,
method = "knn",
na.action = na.omit,
preProcess = c("scale","center"),
trControl = trainControl(method = "cv",number = 10))

model.glm <- train(poverty ~ ., data = training,
method = "glm",
na.action = na.omit,
preProcess = c("scale","center"),
family = "binomial",
trControl = trainControl(method = "cv",number = 10))
```

ROC Curve

```r
crlt <- trainControl(method = "cv",
                     number = 10,
                     classProbs = TRUE,
                     savePredictions = TRUE,
                     summaryFunction = twoClassSummary)


model.glm1 <- train(poverty ~ ., data = training,
method = "glm",
na.action = na.omit,
preProcess = c("scale","center"),
family = "binomial",
trControl = crlt,
metric = "ROC")

model.knn1 <- train(poverty ~ ., data = training,
method = "knn",
na.action = na.omit,
```

```r
preProcess = c("scale","center"),
trControl = crlt,
metric = "ROC")



model.svm <- train(poverty ~ ., data = training,
method = "svmLinear",
na.action = na.omit,
preProcess = c("scale","center"),
trControl = crlt,
metric = "ROC")

model.rf <- train(poverty ~ ., data = training,
method = 'rf',
na.action = na.omit,
preProcess = c("scale","center"),
trControl = crlt,
metric = "ROC")

model.knn1
```

```
k-Nearest Neighbors

14563 samples
    9 predictor
    2 classes: 'No', 'Yes'

Pre-processing: scaled (21), centered (21)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 13106, 13107, 13106, 13107, 13107, 13106, ...
Resampling results across tuning parameters:

  k  ROC        Sens       Spec
  5  0.9239941  0.7779191  0.9130659
  7  0.9319485  0.7789762  0.9168283
  9  0.9356713  0.7768629  0.9190658

ROC was used to select the optimal model using the largest value.
The final value used for the model was k = 9.
```

```r
model.glm1
```

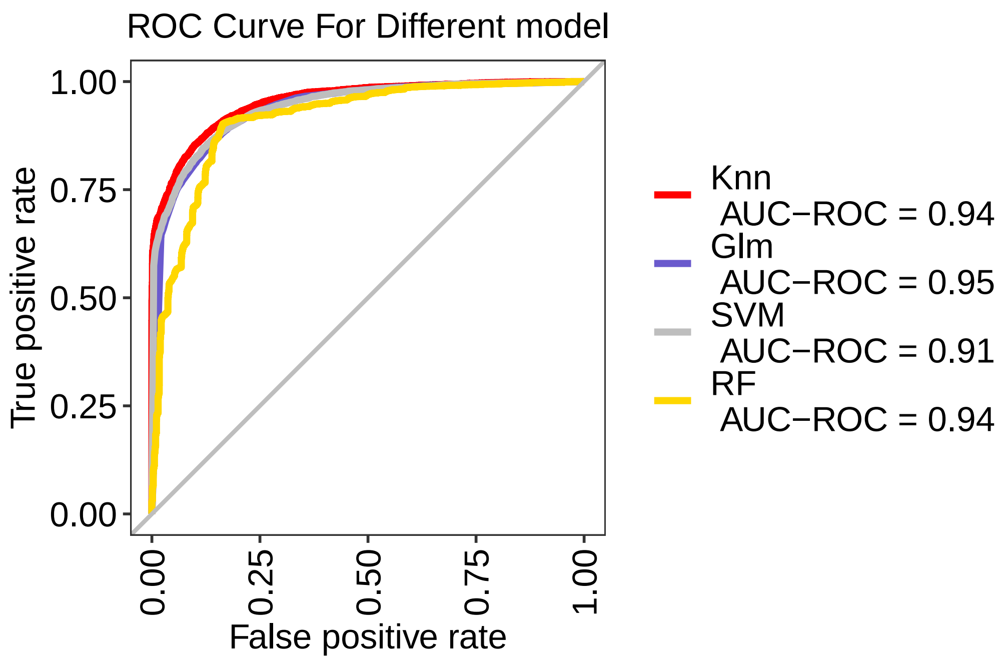
```
Generalized Linear Model

14563 samples
    9 predictor
```

```
    2 classes: 'No', 'Yes'

Pre-processing: scaled (21), centered (21)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 13108, 13106, 13107, 13106, 13107, 13106, ...
Resampling results:

  ROC        Sens     Spec
  0.9534742  0.80034  0.9272998
```

```r
res <- evalm(list(model.knn1,model.glm1, model.svm, model.rf),gnames=c('Knn','Glm', "SVM", "RF"),
             plots = "r", title = "ROC Curve For Different model")
```

```
***MLeval: Machine Learning Model Evaluation***

Input: caret train function object

Not averaging probs.

Group 1 type: cv

Group 2 type: cv

Group 3 type: cv

Group 4 type: cv

Observations: 58252

Number of groups: 4

Observations per group: 14563

Positive: Yes

Negative: No

Group: Knn

Positive: 9835

Negative: 4728
```

```
Group: Glm

Positive: 9835

Negative: 4728

Group: SVM

Positive: 9835

Negative: 4728

Group: RF

Positive: 9835

Negative: 4728

***Performance Metrics***

Knn Optimal Informedness = 0.718271079409237

Glm Optimal Informedness = 0.754733646624465

SVM Optimal Informedness = 0.737658914388596

RF Optimal Informedness = 0.727632974536708

Knn AUC-ROC = 0.94

Glm AUC-ROC = 0.95

SVM AUC-ROC = 0.91

RF AUC-ROC = 0.94
```




```r
varImp(model.glm1)
```

```
glm variable importance

  only 20 most important variables shown (out of 21)

                 Overall
Division3       100.0000
residence1       79.6727
Division2        51.0364
education3       48.1491
Division7        46.0223
Division4        42.8228
age              30.9640
Division5        23.6128
education2       20.4930
marital.status1  18.8543
work.status1     14.2524
Division8        12.4403
education1       10.1480
sex1              7.7414
Family.size1      7.6139
Division6         7.0356
education8        2.6784
Wealth.index5     0.3599
Wealth.index4     0.2550
Wealth.index3     0.1838
```

Apply model for prediction

```r
model.train <- predict(model, training)
model.test <- predict(model, testing)
model.cross <- predict(model.cv,training)
model.cross.test <-  predict(model.cv, testing)
model.random.forest <- predict(model.Rf,training)
model.random.forest.test <- predict(model.Rf, testing)
model.kNN <- predict(model.knn,training)
model.kNN.test <-  predict(model.knn, testing)
model.lr <- predict(model.glm,training)
model.lr.test <-  predict(model.glm, testing)
```

Display confusion matrix

```r
model.train.confusion <- confusionMatrix(model.train, training$poverty)
print(model.train.confusion)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  3836  771
       Yes  892 9064

               Accuracy : 0.8858
                 95% CI : (0.8805, 0.8909)
    No Information Rate : 0.6753
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7378

 Mcnemar's Test P-Value : 0.003254

            Sensitivity : 0.8113
            Specificity : 0.9216
         Pos Pred Value : 0.8326
         Neg Pred Value : 0.9104
             Prevalence : 0.3247
         Detection Rate : 0.2634
   Detection Prevalence : 0.3163
      Balanced Accuracy : 0.8665

       'Positive' Class : No
```

```r
model.test.confusion <- confusionMatrix(model.test, testing$poverty)
print(model.test.confusion)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  1255  235
       Yes  321 3043

               Accuracy : 0.8855
                 95% CI : (0.8762, 0.8943)
    No Information Rate : 0.6753
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.735

 Mcnemar's Test P-Value : 0.0003124

            Sensitivity : 0.7963
            Specificity : 0.9283
         Pos Pred Value : 0.8423
         Neg Pred Value : 0.9046
             Prevalence : 0.3247
         Detection Rate : 0.2585
   Detection Prevalence : 0.3070
      Balanced Accuracy : 0.8623

       'Positive' Class : No
```

```r
model.cv.confusion <- confusionMatrix(model.cross, training$poverty)
model.cv.confusion1 <- confusionMatrix(model.cross.test, testing$poverty)
print(model.cv.confusion)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  3836  771
       Yes  892 9064

               Accuracy : 0.8858
                 95% CI : (0.8805, 0.8909)
    No Information Rate : 0.6753
    P-Value [Acc > NIR] : < 2.2e-16
```

```
                 Kappa : 0.7378

 Mcnemar's Test P-Value : 0.003254

            Sensitivity : 0.8113
            Specificity : 0.9216
         Pos Pred Value : 0.8326
         Neg Pred Value : 0.9104
             Prevalence : 0.3247
         Detection Rate : 0.2634
   Detection Prevalence : 0.3163
      Balanced Accuracy : 0.8665

       'Positive' Class : No
```

```r
print(model.cv.confusion1)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  1255  235
       Yes  321 3043

               Accuracy : 0.8855
                 95% CI : (0.8762, 0.8943)
    No Information Rate : 0.6753
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.735

 Mcnemar's Test P-Value : 0.0003124

            Sensitivity : 0.7963
            Specificity : 0.9283
         Pos Pred Value : 0.8423
         Neg Pred Value : 0.9046
             Prevalence : 0.3247
         Detection Rate : 0.2585
   Detection Prevalence : 0.3070
      Balanced Accuracy : 0.8623

       'Positive' Class : No
```

```r
model.rf.confusion <- confusionMatrix(model.random.forest,training$poverty)
model.rf.confusion1 <- confusionMatrix(model.random.forest.test, testing$poverty)
print(model.rf.confusion)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  3433  401
       Yes 1295 9434

               Accuracy : 0.8835
                 95% CI : (0.8782, 0.8887)
    No Information Rate : 0.6753
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7207

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.7261
            Specificity : 0.9592
         Pos Pred Value : 0.8954
         Neg Pred Value : 0.8793
             Prevalence : 0.3247
         Detection Rate : 0.2357
   Detection Prevalence : 0.2633
      Balanced Accuracy : 0.8427

       'Positive' Class : No
```

```r
print(model.rf.confusion1)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  1148  139
       Yes  428 3139

               Accuracy : 0.8832
                 95% CI : (0.8738, 0.8921)
    No Information Rate : 0.6753
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7203
```

```
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.7284
            Specificity : 0.9576
         Pos Pred Value : 0.8920
         Neg Pred Value : 0.8800
             Prevalence : 0.3247
         Detection Rate : 0.2365
   Detection Prevalence : 0.2651
      Balanced Accuracy : 0.8430

       'Positive' Class : No
```

```r
model.knn.confusion <- confusionMatrix(model.kNN, training$poverty)
model.knn.confusion1 <- confusionMatrix(model.kNN.test, testing$poverty)
print(model.knn.confusion)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  3828  660
       Yes  900 9175

               Accuracy : 0.8929
                 95% CI : (0.8877, 0.8979)
    No Information Rate : 0.6753
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7525

 Mcnemar's Test P-Value : 1.438e-09

            Sensitivity : 0.8096
            Specificity : 0.9329
         Pos Pred Value : 0.8529
         Neg Pred Value : 0.9107
             Prevalence : 0.3247
         Detection Rate : 0.2629
   Detection Prevalence : 0.3082
      Balanced Accuracy : 0.8713

       'Positive' Class : No
```

```r
print(model.knn.confusion1)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  1205  253
       Yes  371 3025

               Accuracy : 0.8714
                 95% CI : (0.8617, 0.8807)
    No Information Rate : 0.6753
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.701

 Mcnemar's Test P-Value : 2.817e-06

            Sensitivity : 0.7646
            Specificity : 0.9228
         Pos Pred Value : 0.8265
         Neg Pred Value : 0.8908
             Prevalence : 0.3247
         Detection Rate : 0.2482
   Detection Prevalence : 0.3004
      Balanced Accuracy : 0.8437

       'Positive' Class : No
```

```r
model.glm.confusion <- confusionMatrix(model.lr,training$poverty)
model.glm.confusion1 <- confusionMatrix(model.lr.test,testing$poverty)
print(model.glm.confusion)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  3789  716
       Yes  939 9119

               Accuracy : 0.8864
                 95% CI : (0.8811, 0.8915)
    No Information Rate : 0.6753
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7376
```

```
 Mcnemar's Test P-Value : 4.842e-08

            Sensitivity : 0.8014
            Specificity : 0.9272
         Pos Pred Value : 0.8411
         Neg Pred Value : 0.9066
             Prevalence : 0.3247
         Detection Rate : 0.2602
   Detection Prevalence : 0.3093
      Balanced Accuracy : 0.8643

       'Positive' Class : No
```

```r
print(model.glm.confusion1)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  1236  232
       Yes  340 3046

               Accuracy : 0.8822
                 95% CI : (0.8728, 0.8911)
    No Information Rate : 0.6753
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7264

 Mcnemar's Test P-Value : 7.681e-06

            Sensitivity : 0.7843
            Specificity : 0.9292
         Pos Pred Value : 0.8420
         Neg Pred Value : 0.8996
             Prevalence : 0.3247
         Detection Rate : 0.2546
   Detection Prevalence : 0.3024
      Balanced Accuracy : 0.8567

       'Positive' Class : No
```

```r
library(caret)

data(Sonar)
```

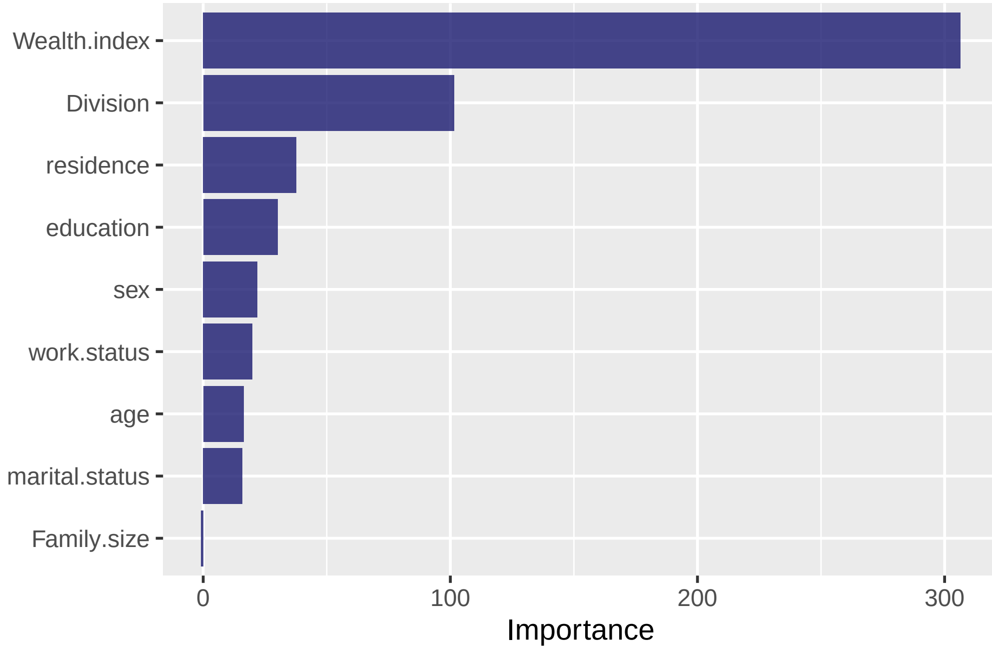
```r
ctrl <- trainControl(method="cv",
  classProbs=TRUE,savePredictions = "all")
rfFit <- train(Class ~ ., data=Sonar,
  method="rf", preProc=c("center", "scale"),
  trControl=ctrl,
  importance=TRUE)

library(randomForest)
```

```
randomForest 4.7-1.1

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


Attaching package: 'randomForest'


The following object is masked from 'package:ranger':

    importance


The following object is masked from 'package:dplyr':

    combine


The following object is masked from 'package:ggplot2':

    margin
```

```r
rfo <-  randomForest(factor(poverty) ~. , data = training, importance = TRUE)

library(vip)
```

```
Attaching package: 'vip'


The following object is masked from 'package:utils':

    vi
```

```r
vip(rfo,aesthetics = list(alpha = 0.8, fill = "midnightblue"))
```




```r
varImp(rfFit)
```

```
rf variable importance

  only 20 most important variables shown (out of 60)

    Importance
V11     100.00
V12      88.50
V9       78.58
V10      68.16
V49      65.71
V48      61.04
V36      57.18
V13      56.53
V45      55.33
V21      54.54
V37      52.90
V28      51.24
V44      49.61
V20      48.08
V46      47.10
V47      43.73
V16      42.60
V1       40.16
V5       40.10
V4       39.59
```

```r
rfFit$variable.importance
```

```
NULL
```

```r
rfFit
```

```
Random Forest

208 samples
 60 predictor
  2 classes: 'M', 'R'

Pre-processing: centered (60), scaled (60)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 188, 188, 187, 187, 187, 187, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.8557143  0.7057720
  31    0.8126190  0.6199729
  60    0.8126190  0.6204268

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 2.
```

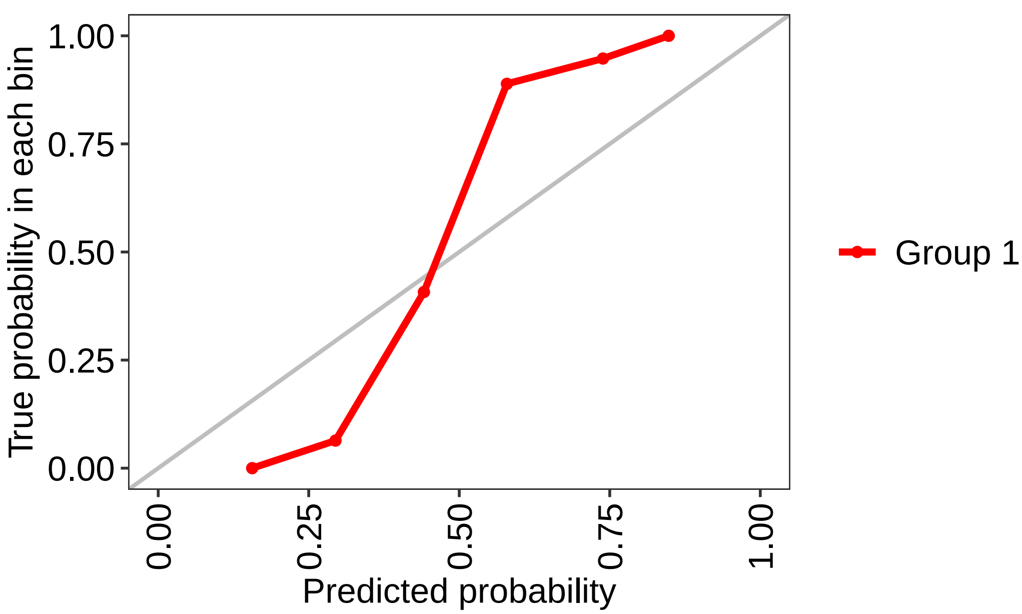
```r
str(Sonar)
```

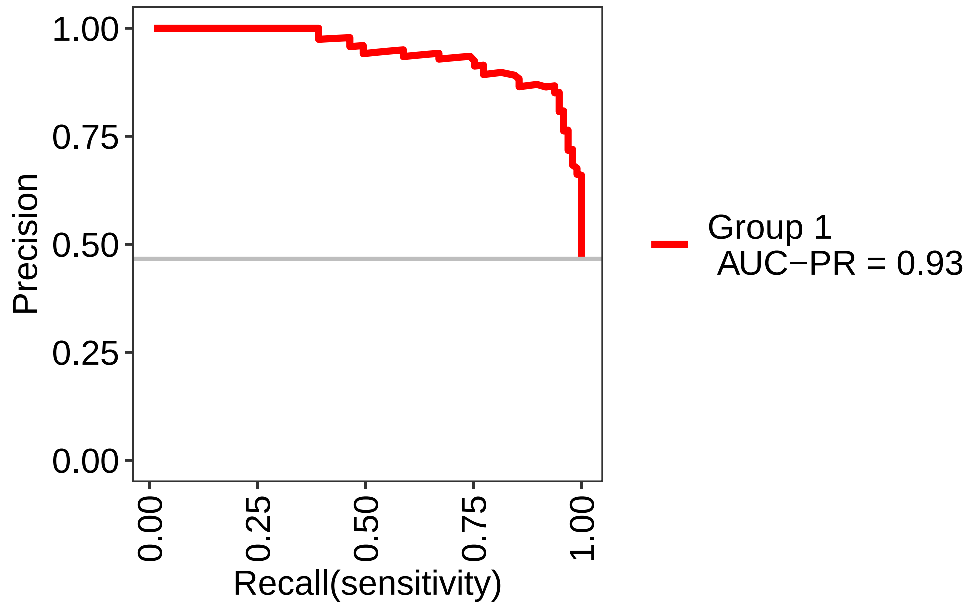
```
'data.frame':   208 obs. of  61 variables:
 $ V1   : num  0.02 0.0453 0.0262 0.01 0.0762 0.0286 0.0317 0.0519 0.0223 0.0164 ...
 $ V2   : num  0.0371 0.0523 0.0582 0.0171 0.0666 0.0453 0.0956 0.0548 0.0375 0.0173 ...
 $ V3   : num  0.0428 0.0843 0.1099 0.0623 0.0481 ...
 $ V4   : num  0.0207 0.0689 0.1083 0.0205 0.0394 ...
 $ V5   : num  0.0954 0.1183 0.0974 0.0205 0.059 ...
 $ V6   : num  0.0986 0.2583 0.228 0.0368 0.0649 ...
 $ V7   : num  0.154 0.216 0.243 0.11 0.121 ...
 $ V8   : num  0.16 0.348 0.377 0.128 0.247 ...
 $ V9   : num  0.3109 0.3337 0.5598 0.0598 0.3564 ...
 $ V10  : num  0.211 0.287 0.619 0.126 0.446 ...
 $ V11  : num  0.1609 0.4918 0.6333 0.0881 0.4152 ...
 $ V12  : num  0.158 0.655 0.706 0.199 0.395 ...
 $ V13  : num  0.2238 0.6919 0.5544 0.0184 0.4256 ...
 $ V14  : num  0.0645 0.7797 0.532 0.2261 0.4135 ...
 $ V15  : num  0.066 0.746 0.648 0.173 0.453 ...
 $ V16  : num  0.227 0.944 0.693 0.213 0.533 ...
 $ V17  : num  0.31 1 0.6759 0.0693 0.7306 ...
```

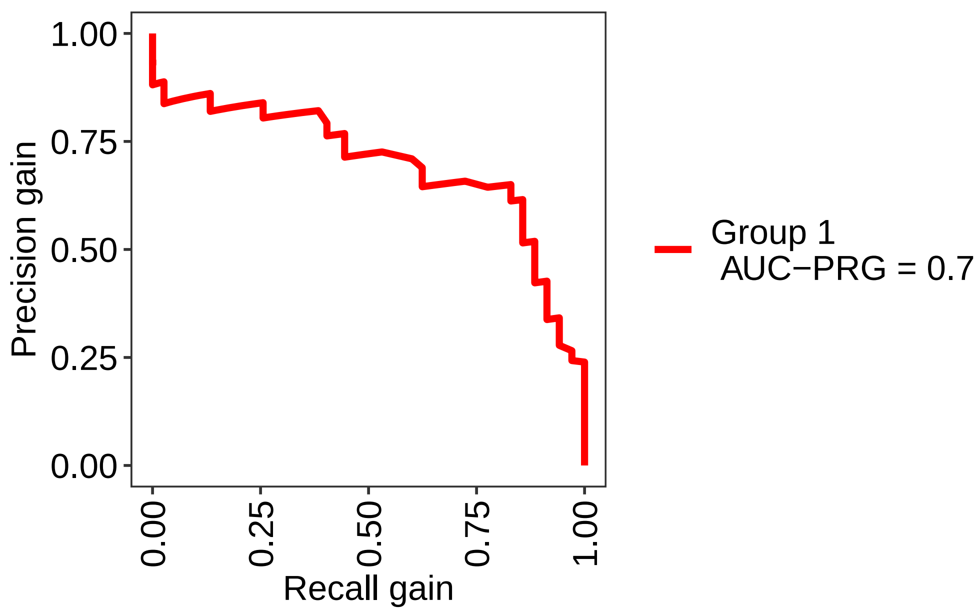
```
$ V18  : num  0.3 0.887 0.755 0.228 0.619 ...
$ V19  : num  0.508 0.802 0.893 0.406 0.203 ...
$ V20  : num  0.48 0.782 0.862 0.397 0.464 ...
$ V21  : num  0.578 0.521 0.797 0.274 0.415 ...
$ V22  : num  0.507 0.405 0.674 0.369 0.429 ...
$ V23  : num  0.433 0.396 0.429 0.556 0.573 ...
$ V24  : num  0.555 0.391 0.365 0.485 0.54 ...
$ V25  : num  0.671 0.325 0.533 0.314 0.316 ...
$ V26  : num  0.641 0.32 0.241 0.533 0.229 ...
$ V27  : num  0.71 0.327 0.507 0.526 0.7 ...
$ V28  : num  0.808 0.277 0.853 0.252 1 ...
$ V29  : num  0.679 0.442 0.604 0.209 0.726 ...
$ V30  : num  0.386 0.203 0.851 0.356 0.472 ...
$ V31  : num  0.131 0.379 0.851 0.626 0.51 ...
$ V32  : num  0.26 0.295 0.504 0.734 0.546 ...
$ V33  : num  0.512 0.198 0.186 0.612 0.288 ...
$ V34  : num  0.7547 0.2341 0.2709 0.3497 0.0981 ...
$ V35  : num  0.854 0.131 0.423 0.395 0.195 ...
$ V36  : num  0.851 0.418 0.304 0.301 0.418 ...
$ V37  : num  0.669 0.384 0.612 0.541 0.46 ...
$ V38  : num  0.61 0.106 0.676 0.881 0.322 ...
$ V39  : num  0.494 0.184 0.537 0.986 0.283 ...
$ V40  : num  0.274 0.197 0.472 0.917 0.243 ...
$ V41  : num  0.051 0.167 0.465 0.612 0.198 ...
$ V42  : num  0.2834 0.0583 0.2587 0.5006 0.2444 ...
$ V43  : num  0.282 0.14 0.213 0.321 0.185 ...
$ V44  : num  0.4256 0.1628 0.2222 0.3202 0.0841 ...
$ V45  : num  0.2641 0.0621 0.2111 0.4295 0.0692 ...
$ V46  : num  0.1386 0.0203 0.0176 0.3654 0.0528 ...
$ V47  : num  0.1051 0.053 0.1348 0.2655 0.0357 ...
$ V48  : num  0.1343 0.0742 0.0744 0.1576 0.0085 ...
$ V49  : num  0.0383 0.0409 0.013 0.0681 0.023 0.0264 0.0507 0.0285 0.0777 0.0092 ...
$ V50  : num  0.0324 0.0061 0.0106 0.0294 0.0046 0.0081 0.0159 0.0178 0.0439 0.0198 ...
$ V51  : num  0.0232 0.0125 0.0033 0.0241 0.0156 0.0104 0.0195 0.0052 0.0061 0.0118 ...
$ V52  : num  0.0027 0.0084 0.0232 0.0121 0.0031 0.0045 0.0201 0.0081 0.0145 0.009 ...
$ V53  : num  0.0065 0.0089 0.0166 0.0036 0.0054 0.0014 0.0248 0.012 0.0128 0.0223 ...
$ V54  : num  0.0159 0.0048 0.0095 0.015 0.0105 0.0038 0.0131 0.0045 0.0145 0.0179 ...
$ V55  : num  0.0072 0.0094 0.018 0.0085 0.011 0.0013 0.007 0.0121 0.0058 0.0084 ...
$ V56  : num  0.0167 0.0191 0.0244 0.0073 0.0015 0.0089 0.0138 0.0097 0.0049 0.0068 ...
$ V57  : num  0.018 0.014 0.0316 0.005 0.0072 0.0057 0.0092 0.0085 0.0065 0.0032 ...
$ V58  : num  0.0084 0.0049 0.0164 0.0044 0.0048 0.0027 0.0143 0.0047 0.0093 0.0035 ...
$ V59  : num  0.009 0.0052 0.0095 0.004 0.0107 0.0051 0.0036 0.0048 0.0059 0.0056 ...
$ V60  : num  0.0032 0.0044 0.0078 0.0117 0.0094 0.0062 0.0103 0.0053 0.0022 0.004 ...
$ Class: Factor w/ 2 levels "M","R": 2 2 2 2 2 2 2 2 2 2 ...
```

```r
classprobs <- predict(rfFit, newdata = Sonar, type = "prob")

res <- evalm(rfFit)
```

```
***MLeval: Machine Learning Model Evaluation***

Input: caret train function object

Not averaging probs.

Group 1 type: cv

Observations: 208

Number of groups: 1

Observations per group: 208

Positive: R

Negative: M

Group: Group 1

Positive: 97

Negative: 111

***Performance Metrics***
```

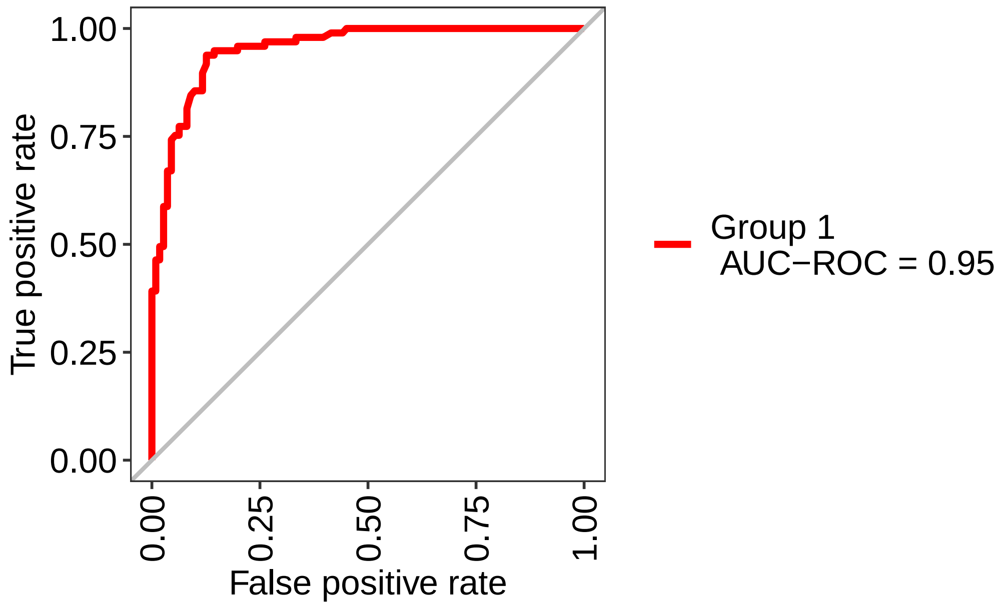




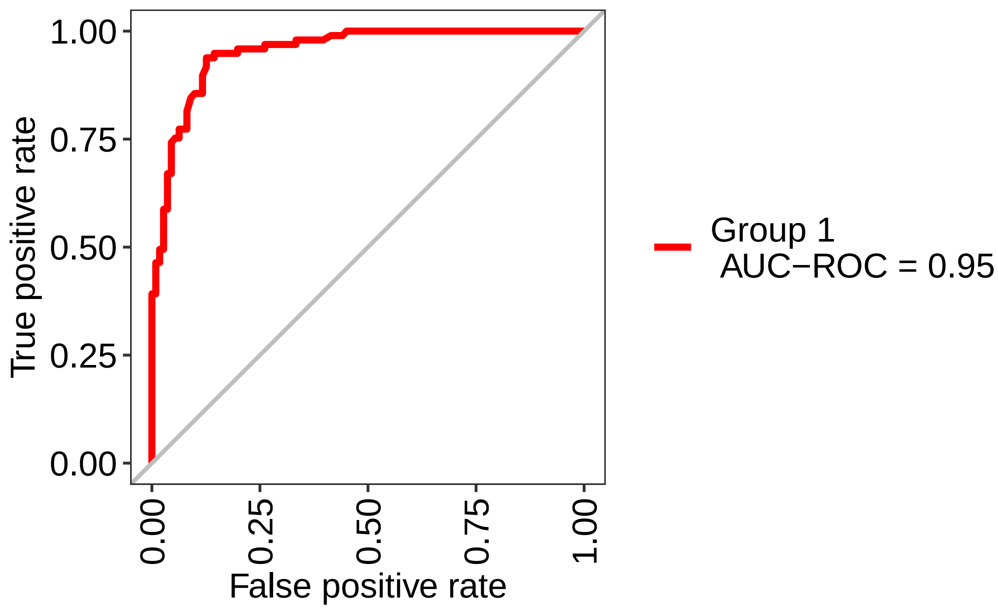



```
Group 1 Optimal Informedness = 0.812018203770781
```

```
Group 1 AUC-ROC = 0.95
```

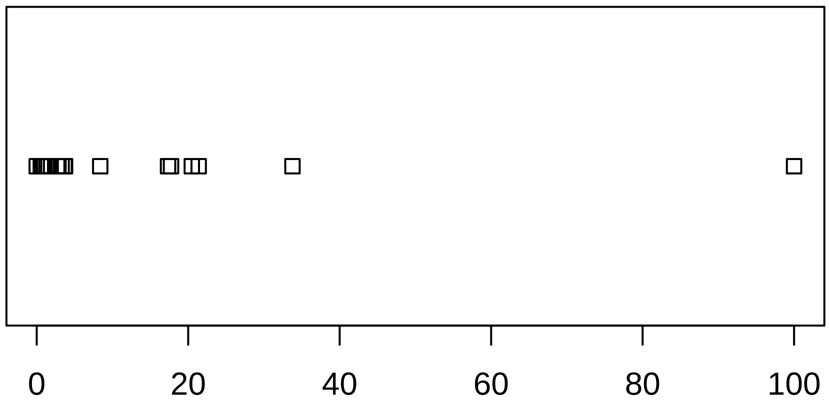



```
res$roc
```

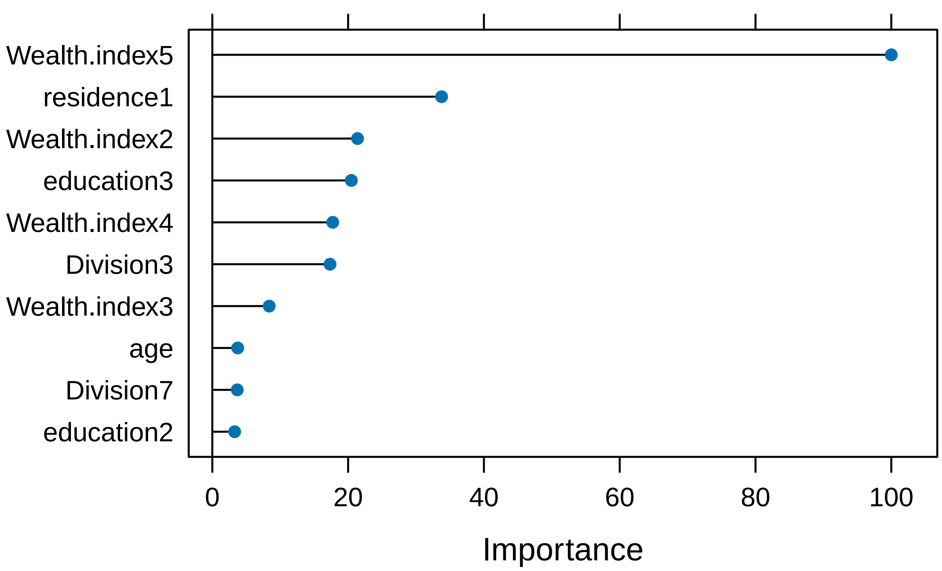



```
model.Rf <- train(poverty ~ ., data = training,
method = 'rf',
na.action = na.omit,
preProcess = c("scale","center"),
trControl = trainControl(method = "cv",number = 10))
```

```r
v <- varImp(model.Rf,scale = TRUE)[["importance"]]

plot(v)
```



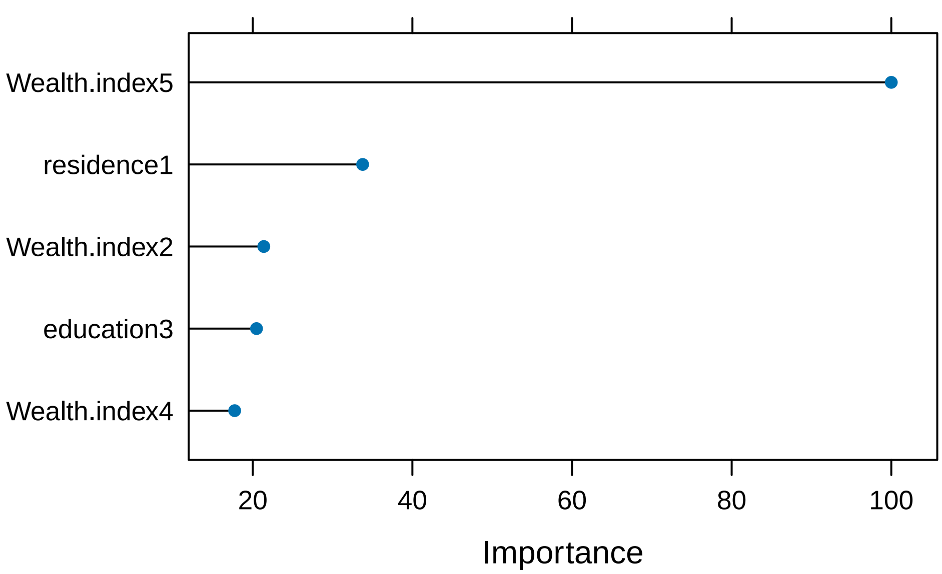

```r
R <- varImp(model.Rf)

plot(R, top = 10)
```



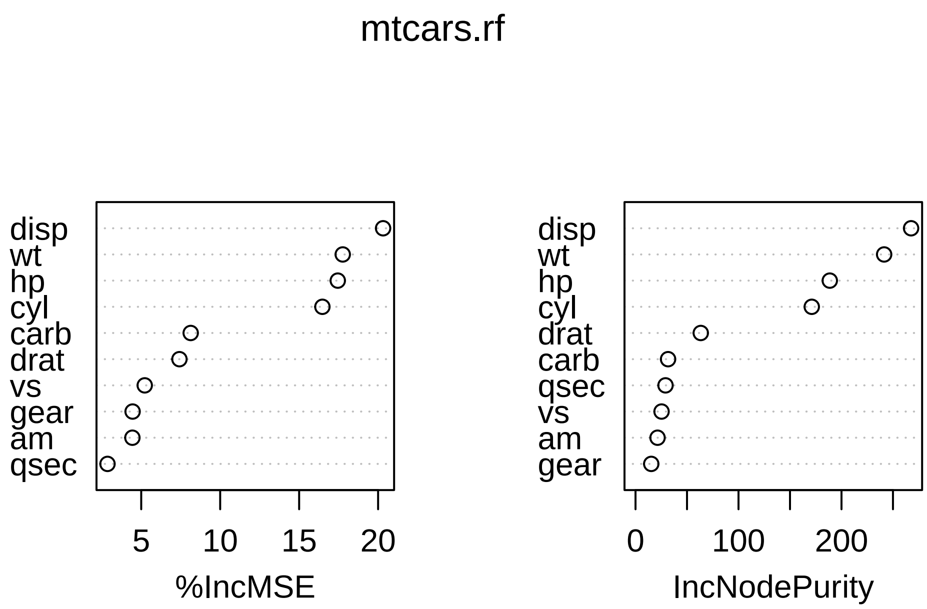
```
v |>
  arrange(desc(Overall))
```

```
                    Overall
Wealth.index5   100.0000000
residence1       33.7681623
Wealth.index2    21.3855939
education3       20.4741640
Wealth.index4    17.7339306
Division3        17.3363117
Wealth.index3     8.3820476
age               3.7401379
Division7         3.6729697
education2        3.2900103
Division2         2.8746793
education1        2.7000639
work.status1      1.5504481
Division4         1.2873140
Division5         1.0047847
Division6         0.7825403
sex1              0.6911694
marital.status1   0.6699631
Family.size1      0.6447404
Division8         0.5378978
education8        0.0000000
```

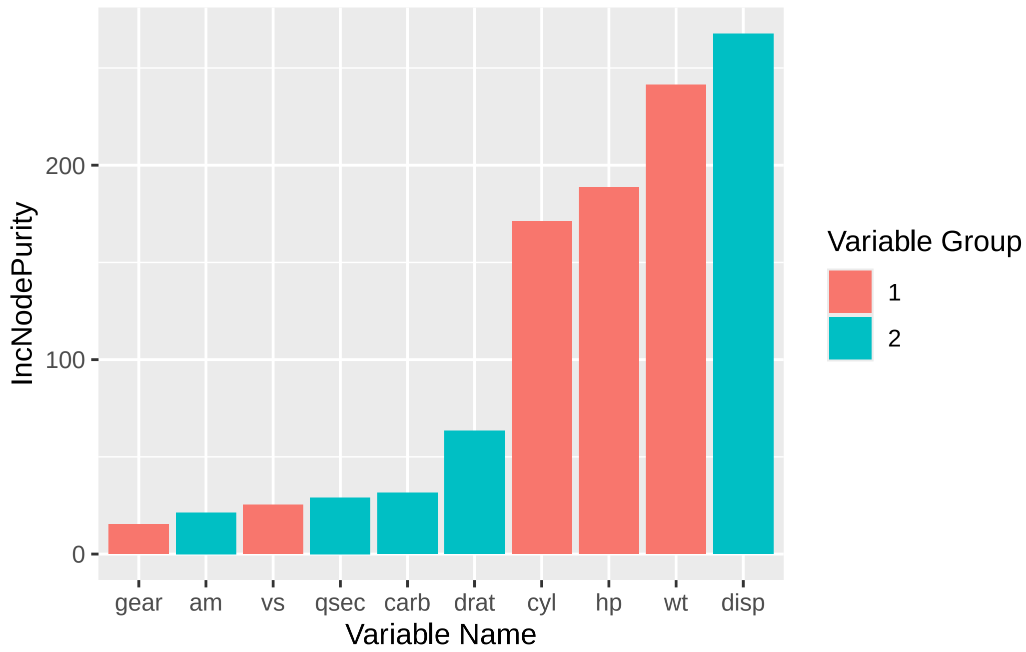
```
plot(varImp(model.Rf),top=5)
```



```
set.seed(4543)
data(mtcars)

library(randomForest)
mtcars.rf <- randomForest(mpg ~ ., data=mtcars, ntree=1000, keep.forest=FALSE,
                          importance=TRUE)
imp <- varImpPlot(mtcars.rf) # let's save the varImp object
```




```
# this part just creates the data.frame for the plot part
library(dplyr)
imp <- as.data.frame(imp)
imp$varnames <- rownames(imp) # row names to column
rownames(imp) <- NULL
imp$var_categ <- rep(1:2, 5) # random var category

# this is the plot part, be sure to use reorder with the correct measure name
library(ggplot2)
ggplot(imp, aes(x=reorder(varnames, IncNodePurity), weight=IncNodePurity, fill=as.factor(var_categ))
  geom_bar() +
  scale_fill_discrete(name="Variable Group") +
  ylab("IncNodePurity") +
  xlab("Variable Name")
```




```r
library(pROC)
```

```
Type 'citation("pROC")' for a citation.

Attaching package: 'pROC'

The following objects are masked from 'package:stats':

    cov, smooth, var
```

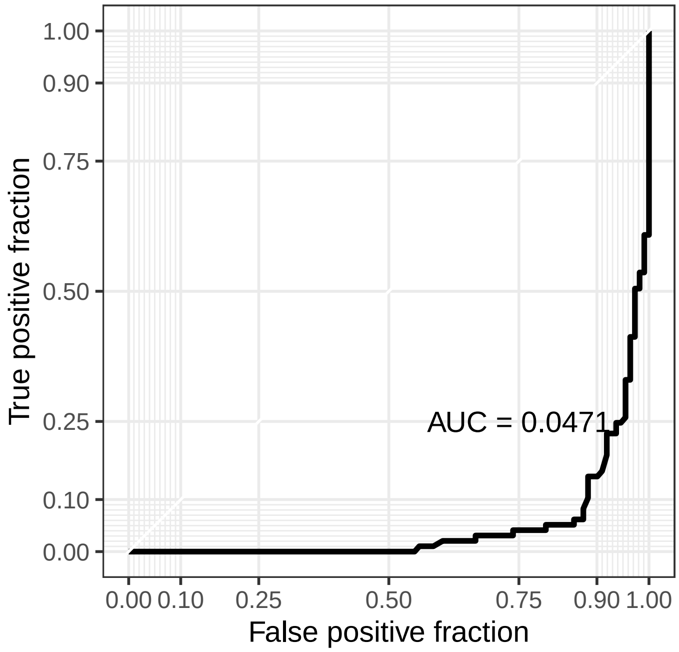
```r
library(plotROC)
```

```
Attaching package: 'plotROC'

The following object is masked from 'package:pROC':

    ggroc
```

```r
selectedIndices <- rfFit$pred$mtry == 2

g <- ggplot(rfFit$pred[selectedIndices, ], aes(m=M, d=factor(obs, levels = c("R", "M")))) +
  geom_roc(n.cuts=0) +
  coord_equal() +
  style_roc()
```

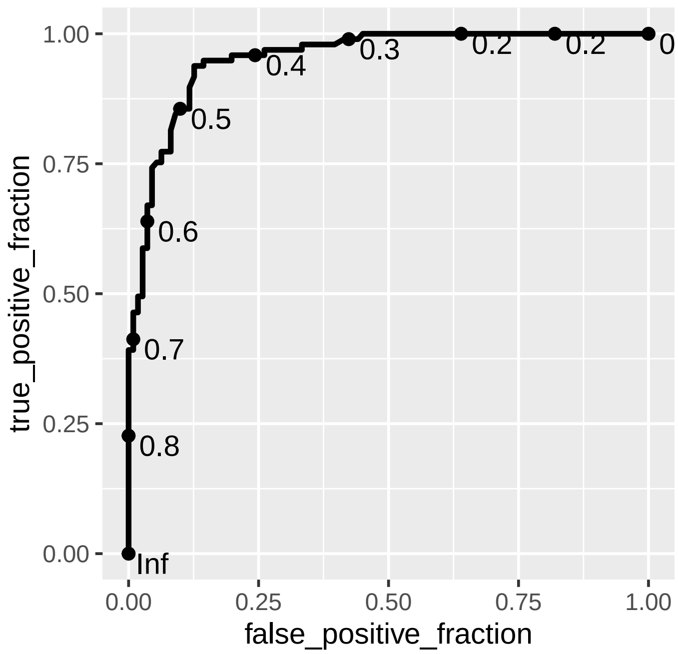
```r
g + annotate("text", x=0.75, y=0.25, label=paste("AUC =", round((calc_auc(g))$AUC, 4)))
```

```
Warning in verify_d(data$d): D not labeled 0/1, assuming M = 0 and R = 1!

Warning in verify_d(data$d): D not labeled 0/1, assuming M = 0 and R = 1!
```



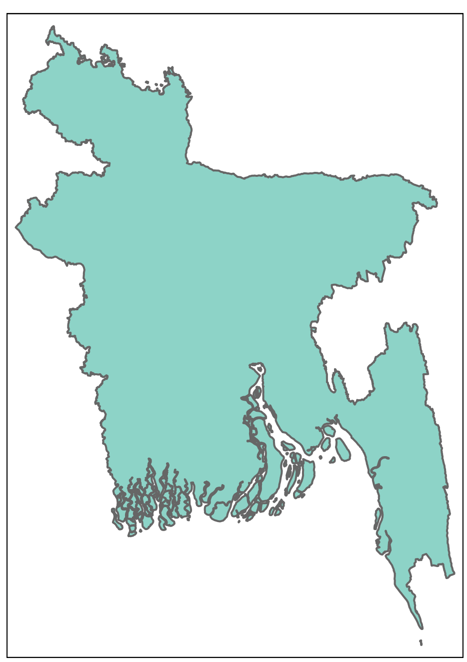

```r
ggplot(rfFit$pred[selectedIndices, ],
       aes(m = R, d = factor(obs, levels = c("R", "M")))) +
    geom_roc(hjust = -0.4, vjust = 1.5) + coord_equal()
```

```
Warning in verify_d(data$d): D not labeled 0/1, assuming M = 0 and R = 1!
```



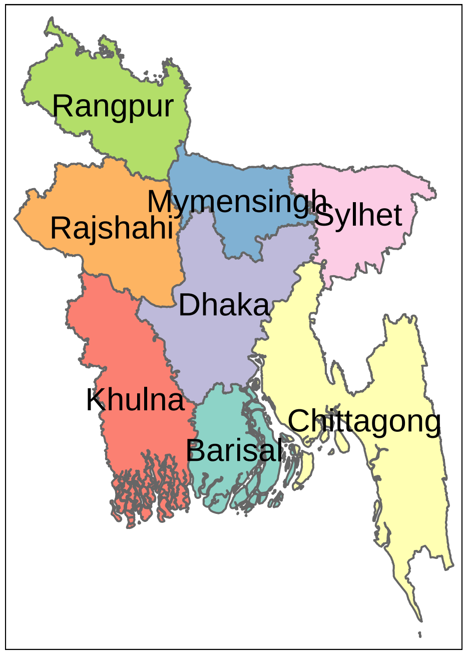

```r
country <- get_map("country")
division <- get_map("division")
district <- get_map("district")
upazila <- get_map("upazila")
union <- get_map("union")

bd_plot("country")
```

```
tmap mode set to plotting
```



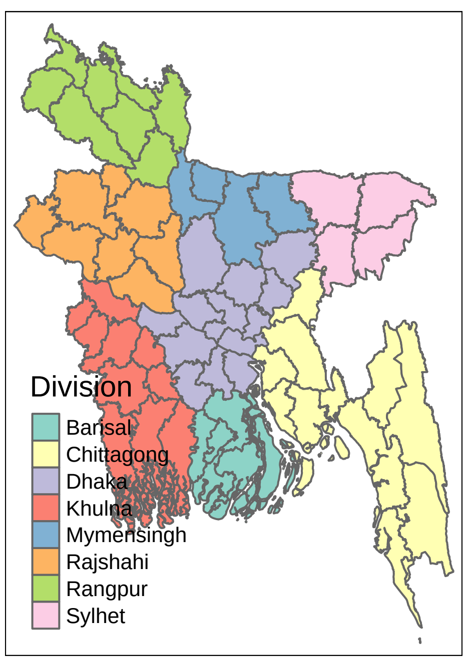
```r
bd_plot("division")
```

```
tmap mode set to plotting
```

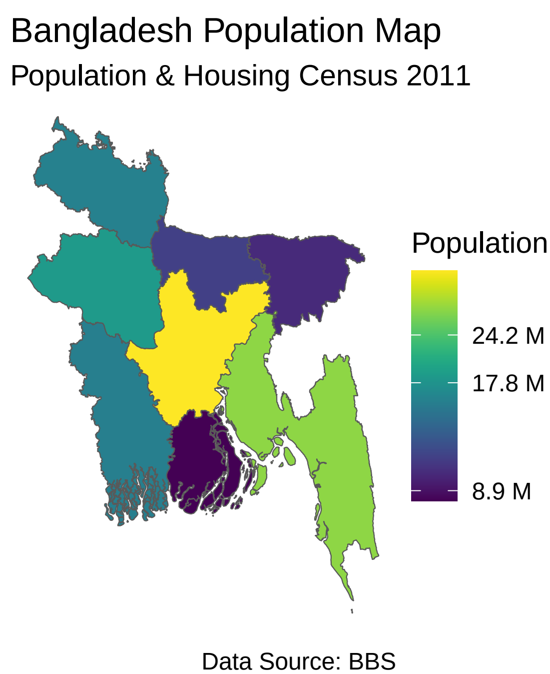



```r
bd_plot("district")
```

```
tmap mode set to plotting
```



```r
library(tmap)
population <- bangladesh::pop_division_2011[, c("division", "population")]
district <- get_map("district")
division <- get_map("division")
map_data <- dplyr::left_join(division, population, by = c("Division" = "division"))

ggplot(data = map_data) +
  geom_sf(aes(fill = population))+
  theme_void() +
  viridis::scale_fill_viridis(trans = "log", name="Population", labels = scales::unit_format(unit =
  labs(
    title = "Bangladesh Population Map",
    subtitle = "Population & Housing Census 2011",
    caption = "Data Source: BBS"
  )
```



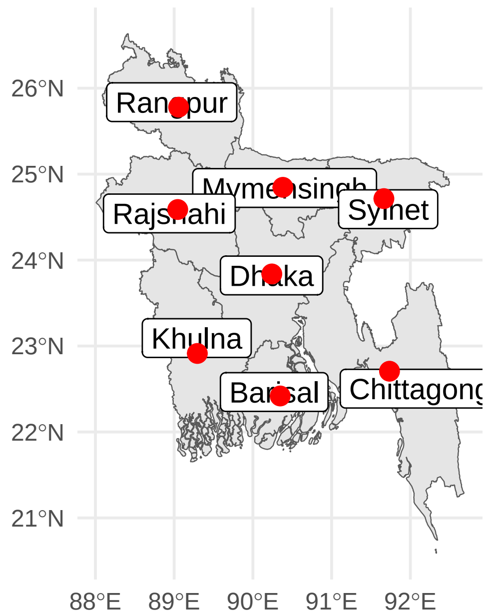

```r
division_map <- get_map("division")
division_centroids <- bangladesh::get_coordinates(level = "division")
knitr::kable(division_centroids, format = "html")
```

Division	lat	lon
Barisal	22.41889	90.34684
Chittagong	22.70692	91.73546
Dhaka	23.83870	90.24064
Khulna	22.91367	89.29437
Mymensingh	24.84675	90.38088
Rajshahi	24.58846	89.04540

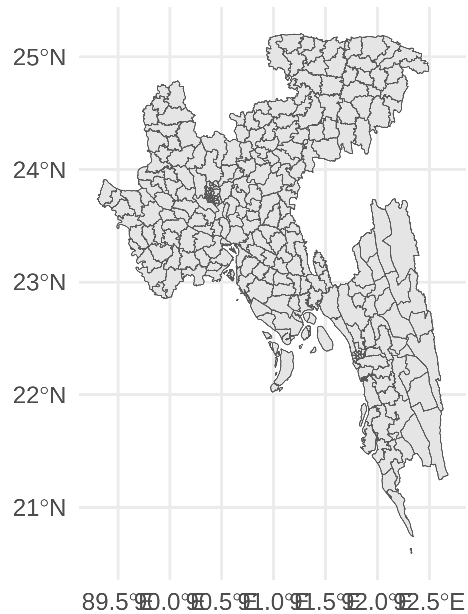
Division	lat	lon
Rangpur	25.77920	89.05685
Sylhet	24.71515	91.66400

```
ggplot(data = division_map) +
  geom_sf() +
  theme_void()+
  geom_sf_label(aes(label = Division))+
  geom_point(data = division_centroids, x = division_centroids$lon, y = division_centroids$lat, col
  xlab("")+ ylab("")+
  theme_minimal()
```

```
Warning in st_point_on_surface.sfc(sf::st_zm(x)): st_point_on_surface may not
give correct results for longitude/latitude data
```



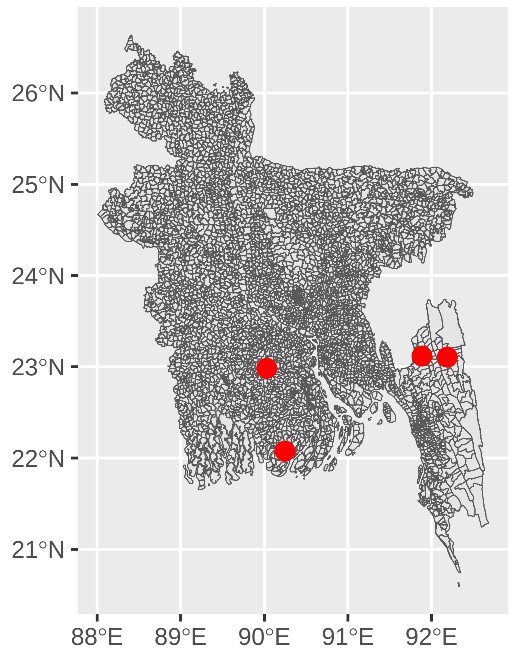

```
sylhet_chittagong_dhaka <- get_divisions(divisions = c("Sylhet", "Chittagong", "Dhaka"),level =  "up
ggplot(data = sylhet_chittagong_dhaka) +
  geom_sf() +
  xlab("")+ ylab("")+
  theme_minimal()
```




```
amtali <- bd_search("amtali", level = "union", as.is = TRUE, coordinates = TRUE)
knitr::kable(amtali, format = "html")
```

Division	District	Upazila	Union	lat	lon
Barisal	Barguna	Amtali	Amtali	22.07556	90.24699
Chittagong	Rangamati	Baghai Chhari	Amtali	23.10559	92.18832
Dhaka	Gopalganj	Kotali Para	Amtali	22.98264	90.03070
Chittagong	Khagrachhari	Matiranga	Amtali	23.11701	91.88545

```
ggplot(bangladesh::map_union) +
  geom_sf() +
  geom_point(data = amtali, x = amtali$lon, y = amtali$lat, col = "red", size = 3)
```



Univariate Analysis

```
hr_a <- hr_df |>
  select(cooking.fuel,Electricity, Television, Mobile.phone, Landline, Refrigerator, separate.kitche
  mutate_all(as.numeric, as.factor) |>
  mutate(across(1:7,as.factor)) |>
  tbl_summary()


skimr::skim(hrml1) %>%
select(-c( n_missing,complete_rate)) %>%
filter(skim_variable != "poverty")
```

Table 6: Data summary

Name	hrml1
Number of rows	19417
Number of columns	10
Column type frequency:	
factor	8
numeric	1
Group variables	None

Variable type: factor

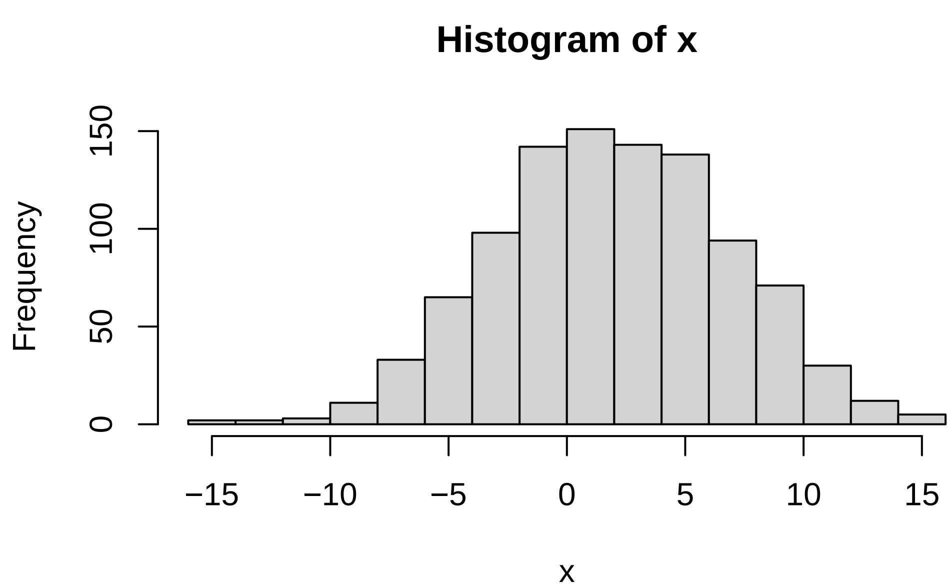
skim_variable	ordered	n_unique	top_counts
Family.size	FALSE	2	1: 17194, 0: 2223
residence	FALSE	2	0: 12333, 1: 7084
sex	FALSE	2	1: 16434, 0: 2983
education	FALSE	5	1: 6303, 0: 5408, 2: 4996, 3: 2694
marital.status	FALSE	2	1: 17600, 0: 1817
work.status	FALSE	2	1: 16603, 0: 2814
Wealth.index	FALSE	5	5: 4095, 1: 4072, 2: 3833, 4: 3789
Division	FALSE	8	3: 2904, 2: 2641, 6: 2561, 4: 2511

Variable type: numeric

skim_variable	mean	sd	p0	p25	p50	p75	p100	hist
age	45.72	14.33	15	35	45	55	95	

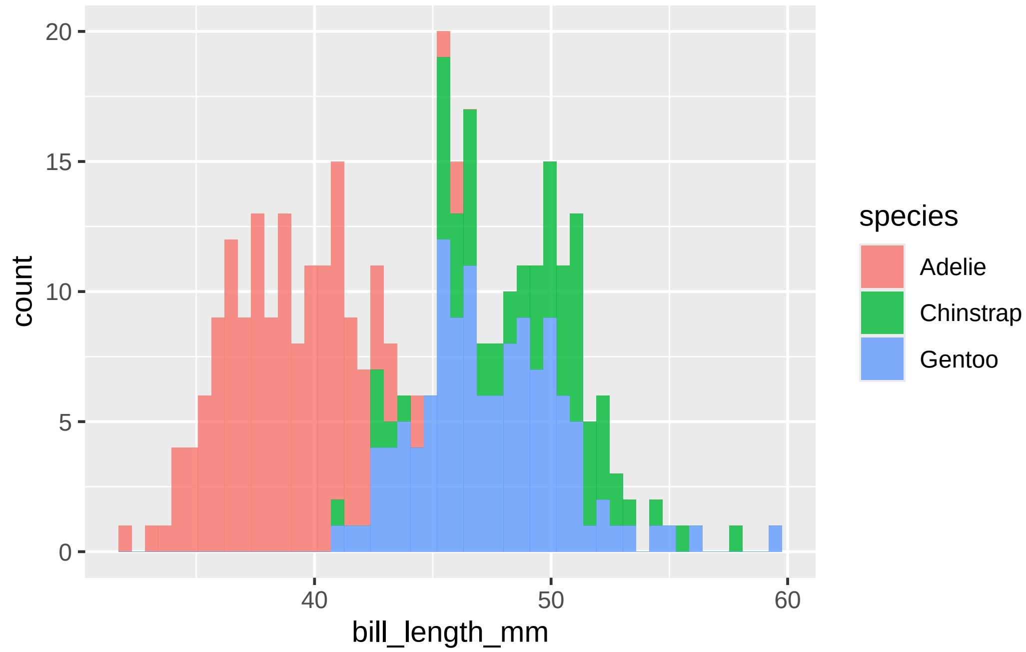
Generate data from Normal Distribution

```r
x <- rnorm(1000,2,5)
hist(x)
```




```r
penguins |>
  ggplot(aes(x= bill_length_mm, fill = species))+
  geom_histogram(bins = 50, alpha=0.8)
```

```
Warning: Removed 2 rows containing non-finite outside the scale range
(`stat_bin()`).
```

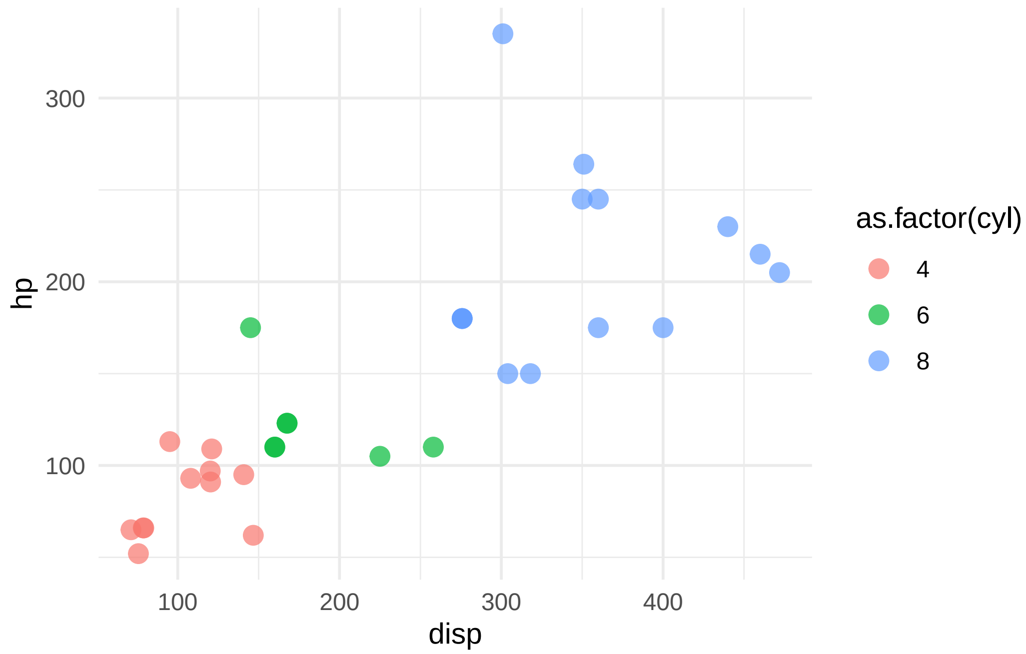


Data Cleaning

```r
head(mtcars)
```

```
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

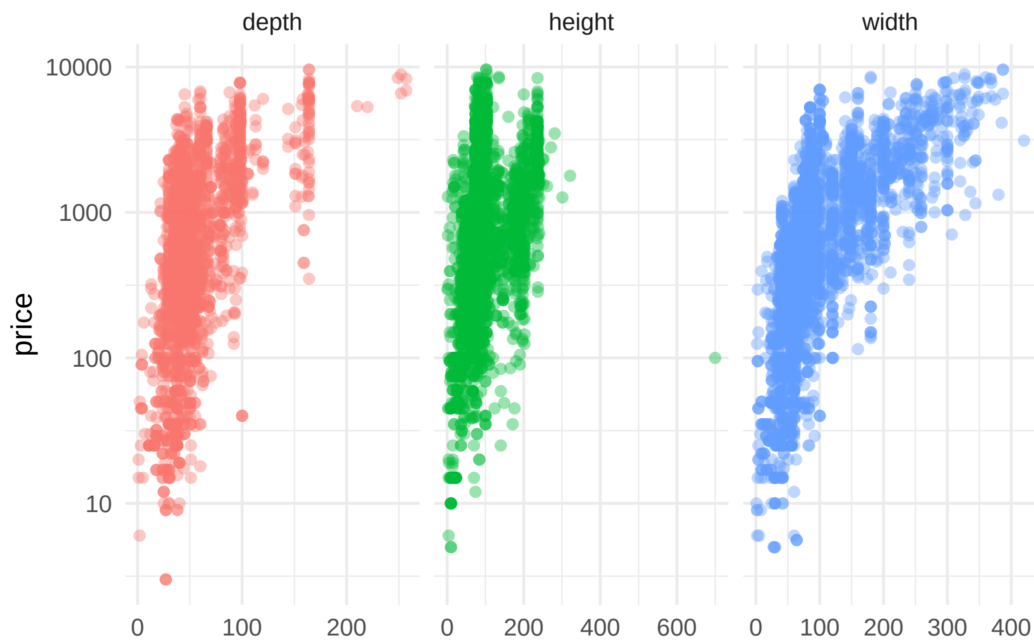
```r
ggplot(mtcars,aes(x= disp,y=hp,col=as.factor(cyl)))+
  geom_point(alpha=0.7,size=3)+
  theme_minimal()
```




```
library(tidyverse)
ikea <- read_csv("https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2020/202
```

```
New names:
Rows: 3694 Columns: 14
-- Column specification
-------------------------------------------------------- Delimiter: "," chr
(7): name, category, old_price, link, other_colors, short_description, d... dbl
(6): ...1, item_id, price, depth, height, width lgl (1): sellable_online
i Use `spec()` to retrieve the full column specification for this data. i
Specify the column types or set `show_col_types = FALSE` to quiet this message.
* `` -> `...1`
```

```
ikea <- rename(ikea, id = ...1)

ikea %>%
  select(id, price, depth:width) %>%
  pivot_longer(depth:width, names_to = "dim") %>%
  ggplot(aes(value, price, color = dim)) +
  geom_point(alpha = 0.4, show.legend = FALSE) +
  scale_y_log10() +
  facet_wrap(~dim, scales = "free_x") +
  labs(x = NULL) +
  theme_minimal()
```




```r
ikea_df <- ikea %>%
  select(price, name, category, depth, height, width) %>%
  mutate(price = log10(price)) %>%
  mutate_if(is.character, factor)

ikea_df
```

```
# A tibble: 3,694 x 6
   price name                    category      depth height width
   <dbl> <fct>                   <fct>         <dbl>  <dbl> <dbl>
 1  2.42 FREKVENS                Bar furniture    NA     99    51
 2  3.00 NORDVIKEN               Bar furniture    NA    105    80
 3  3.32 NORDVIKEN / NORDVIKEN Bar furniture    NA     NA    NA
 4  1.84 STIG                    Bar furniture    50    100    60
 5  2.35 NORBERG                 Bar furniture    60     43    74
 6  2.54 INGOLF                  Bar furniture    45     91    40
 7  2.11 FRANKLIN                Bar furniture    44     95    50
 8  2.29 DALFRED                 Bar furniture    50     NA    50
 9  2.11 FRANKLIN                Bar furniture    44     95    50
10  3.34 EKEDALEN / EKEDALEN     Bar furniture    NA     NA    NA
# i 3,684 more rows
```

#Building Model

```r
## Build Model


library(tidymodels)
```

```
-- Attaching packages -------------------------------------- tidymodels 1.1.1 --

v broom        1.0.5     v rsample      1.2.0
v dials        1.2.0     v tune         1.1.2
v infer        1.0.5     v workflows    1.1.3
v modeldata    1.2.0     v workflowsets 1.0.1
v parsnip      1.1.1     v yardstick    1.2.0
v recipes      1.0.8

-- Conflicts ----------------------------------------- tidymodels_conflicts() --
x recipes::all_double()    masks gtsummary::all_double()
x recipes::all_factor()    masks gtsummary::all_factor()
x recipes::all_integer()   masks gtsummary::all_integer()
x recipes::all_logical()   masks gtsummary::all_logical()
x recipes::all_numeric()   masks gtsummary::all_numeric()
x scales::alpha()          masks kernlab::alpha(), ggplot2::alpha()
x randomForest::combine()  masks dplyr::combine()
x kernlab::cross()         masks purrr::cross()
x scales::discard()        masks purrr::discard()
x dplyr::filter()          masks stats::filter()
x recipes::fixed()         masks stringr::fixed()
x dplyr::lag()             masks stats::lag()
x caret::lift()            masks purrr::lift()
x randomForest::margin()   masks ggplot2::margin()
x yardstick::precision()   masks caret::precision()
x yardstick::recall()      masks caret::recall()
x yardstick::sensitivity() masks caret::sensitivity()
x yardstick::spec()        masks readr::spec()
x yardstick::specificity() masks caret::specificity()
x recipes::step()          masks stats::step()
* Search for functions across packages at https://www.tidymodels.org/find/
```

```r
set.seed(123)
ikea_split <- initial_split(ikea_df, strata = price)
ikea_train <- training(ikea_split)
ikea_test <- testing(ikea_split)

set.seed(234)
ikea_folds <- bootstraps(ikea_train, strata = price)
ikea_folds
```

```
# Bootstrap sampling using stratification
# A tibble: 25 x 2
   splits              id
   <list>              <chr>
 1 <split [2770/994]>  Bootstrap01
 2 <split [2770/1003]> Bootstrap02
```

```
 3 <split [2770/1037]> Bootstrap03
 4 <split [2770/1010]> Bootstrap04
 5 <split [2770/1014]> Bootstrap05
 6 <split [2770/1007]> Bootstrap06
 7 <split [2770/1036]> Bootstrap07
 8 <split [2770/1016]> Bootstrap08
 9 <split [2770/1021]> Bootstrap09
10 <split [2770/1043]> Bootstrap10
# i 15 more rows
```

```r
library(usemodels)
use_ranger(price ~ ., data = ikea_train)
```

```
ranger_recipe <-
  recipe(formula = price ~ ., data = ikea_train)

ranger_spec <-
  rand_forest(mtry = tune(), min_n = tune(), trees = 1000) %>%
  set_mode("classification") %>%
  set_engine("ranger")

ranger_workflow <-
  workflow() %>%
  add_recipe(ranger_recipe) %>%
  add_model(ranger_spec)

set.seed(67013)
ranger_tune <-
  tune_grid(ranger_workflow, resamples = stop("add your rsample object"), grid = stop("add number of c
```

```r
## lots of options, like use_xgboost, use_glmnet, etc


library(textrecipes)
ranger_recipe <-
  recipe(formula = price ~ ., data = ikea_train) %>%
  step_other(name, category, threshold = 0.01) %>%
  step_clean_levels(name, category) %>%
  step_impute_knn(depth, height, width)

ranger_spec <-
  rand_forest(mtry = tune(), min_n = tune(), trees = 1000) %>%
  set_mode("regression") %>%
  set_engine("ranger")

ranger_workflow <-
  workflow() %>%
```

```r
  add_recipe(ranger_recipe) %>%
  add_model(ranger_spec)

set.seed(8577)
doParallel::registerDoParallel()
ranger_tune <-
  tune_grid(ranger_workflow,
    resamples = ikea_folds,
    grid = 11
  )
```

```
i Creating pre-processing data to finalize unknown parameter: mtry
```

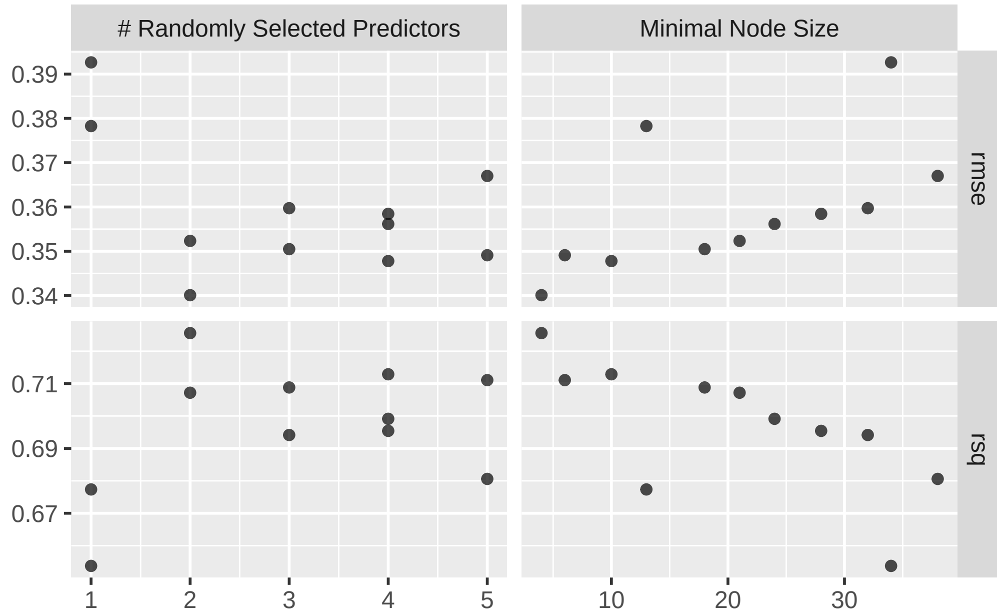
```r
show_best(ranger_tune, metric = "rmse")
```

```
# A tibble: 5 x 8
   mtry min_n .metric .estimator  mean     n std_err .config
  <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1     2     4 rmse    standard   0.340    25 0.00203 Preprocessor1_Model10
2     4    10 rmse    standard   0.348    25 0.00226 Preprocessor1_Model05
3     5     6 rmse    standard   0.349    25 0.00235 Preprocessor1_Model06
4     3    18 rmse    standard   0.350    25 0.00218 Preprocessor1_Model01
5     2    21 rmse    standard   0.352    25 0.00200 Preprocessor1_Model08
```

```r
show_best(ranger_tune, metric = "rsq")
```

```
# A tibble: 5 x 8
   mtry min_n .metric .estimator  mean     n std_err .config
  <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1     2     4 rsq     standard   0.726    25 0.00332 Preprocessor1_Model10
2     4    10 rsq     standard   0.713    25 0.00372 Preprocessor1_Model05
3     5     6 rsq     standard   0.711    25 0.00385 Preprocessor1_Model06
4     3    18 rsq     standard   0.709    25 0.00368 Preprocessor1_Model01
5     2    21 rsq     standard   0.707    25 0.00347 Preprocessor1_Model08
```

```r
autoplot(ranger_tune)
```




```r
final_rf <- ranger_workflow %>%
  finalize_workflow(select_best(ranger_tune))
```

```
Warning: No value of `metric` was given; metric 'rmse' will be used.
```

```r
final_rf
```

```
== Workflow ====================================================================
Preprocessor: Recipe
Model: rand_forest()

-- Preprocessor ----------------------------------------------------------------
3 Recipe Steps

* step_other()
* step_clean_levels()
* step_impute_knn()

-- Model -----------------------------------------------------------------------
Random Forest Model Specification (regression)

Main Arguments:
  mtry = 2
  trees = 1000
  min_n = 4

Computational engine: ranger
```

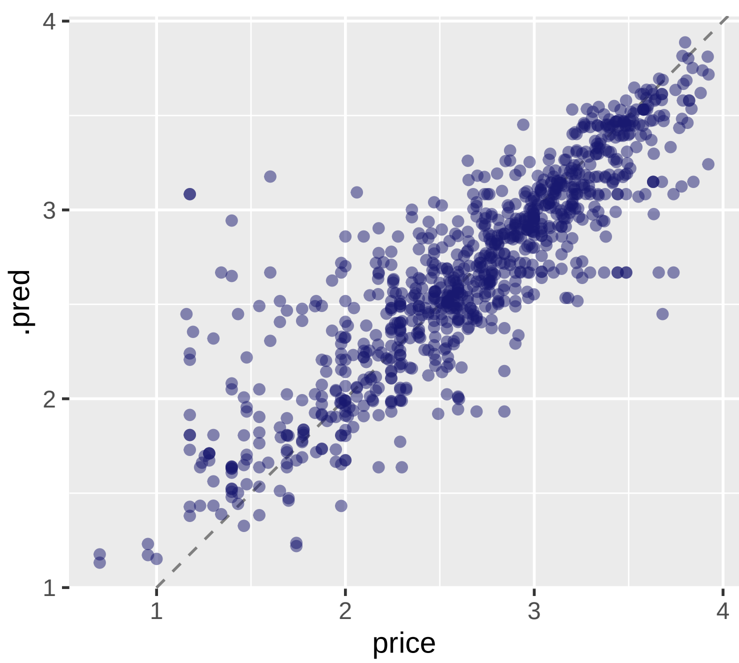
```r
ikea_fit <- last_fit(final_rf, ikea_split)
ikea_fit
```

```
# Resampling results
# Manual resampling
# A tibble: 1 x 6
  splits             id               .metrics .notes   .predictions .workflow
  <list>             <chr>            <list>   <list>   <list>       <list>
1 <split [2770/924]> train/test split <tibble> <tibble> <tibble>     <workflow>
```

```r
collect_metrics(ikea_fit)
```

```
# A tibble: 2 x 4
  .metric .estimator .estimate .config
  <chr>   <chr>          <dbl> <chr>
1 rmse    standard       0.318 Preprocessor1_Model1
2 rsq     standard       0.753 Preprocessor1_Model1
```

```r
collect_predictions(ikea_fit) %>%
  ggplot(aes(price, .pred)) +
  geom_abline(lty = 2, color = "gray50") +
  geom_point(alpha = 0.5, color = "midnightblue") +
  coord_fixed()
```




```r
predict(ikea_fit$.workflow[[1]], ikea_test[15, ])
```

```
# A tibble: 1 x 1
  .pred
  <dbl>
1  2.42
```

```
library(vip)

imp_spec <- ranger_spec %>%
  finalize_model(select_best(ranger_tune)) %>%
  set_engine("ranger", importance = "permutation")
```

```
Warning: No value of `metric` was given; metric 'rmse' will be used.
```

```
workflow() %>%
  add_recipe(ranger_recipe) %>%
  add_model(imp_spec) %>%
  fit(ikea_train) %>%
  pull_workflow_fit() %>%
  vip(aesthetics = list(alpha = 0.8, fill = "midnightblue"))
```

```
Warning: `pull_workflow_fit()` was deprecated in workflows 0.2.3.
i Please use `extract_fit_parsnip()` instead.
```