bikeShare

Alan n. Inglis

2022-08-26

Load libraries:

```
library(lubridate) # for data manipulation
library(BART) # for model
library(bartMan) # for visualizations
library(dplyr) # for data manipulation
library(ggplot2) # for visualizations
```

Read in and setup data:

```
# read data
sbd <- read.csv('/Users/alaninglis/Desktop/SeoulBikeData.csv', sep = ',', check.names = F)


# data manipulation
sbd <- sbd |>
  mutate(dates = dmy(Date))

dd <- sbd |>
  group_by(dates, Season, Holiday) |>
  summarise(Count = sum(Count),
            Temp = mean(Temp),
            Humidity = mean(Humidity),
            Wind.Spd = mean(Wind.Spd),
            Visibility = mean(Visibility),
            Dew.Pt = mean(Dew.Pt),
            Solar.R = mean(Solar.R),
            Rainfall = mean(Rainfall),
            Snowfall = mean(Snowfall)
  ) |>
  mutate(Day = day(dates), Month = month(dates), Year = year(dates))

dd$Month <- month.abb[dd$Month]
dd$Day <- weekdays(as.Date(dd$dates,'%d-%m-%Y'))

# create weekend column
wkday <- c("Monday", "Tuesday" ,  "Wednesday" ,"Thursday", 'Friday' )
wke <- c( "Saturday",  "Sunday" )

dd <- transform(dd, Wkend = ifelse(dd$Day %in% wke, 'wkend', 'wkday'))
```

```r
# make factor variables
dd$Season  <- as.factor(dd$Season)
dd$Holiday  <- as.factor(dd$Holiday)
dd$Month <- as.factor(dd$Month)
dd$Day <- as.factor(dd$Day)
dd$Year <- as.factor(dd$Year)
dd$Wkend <- as.factor(dd$Wkend)

# remove unnessesery columns
dd <- dd[,-c(1)]
dd <- dd[,-12]


# remove zero counts
zeroC <- which(dd$Count == 0)
dd <- dd[-zeroC, ]

# transform response
transCols <- c('Count')
dd[transCols] <- (dd[transCols] + 1)^(1/3)
```

Build models

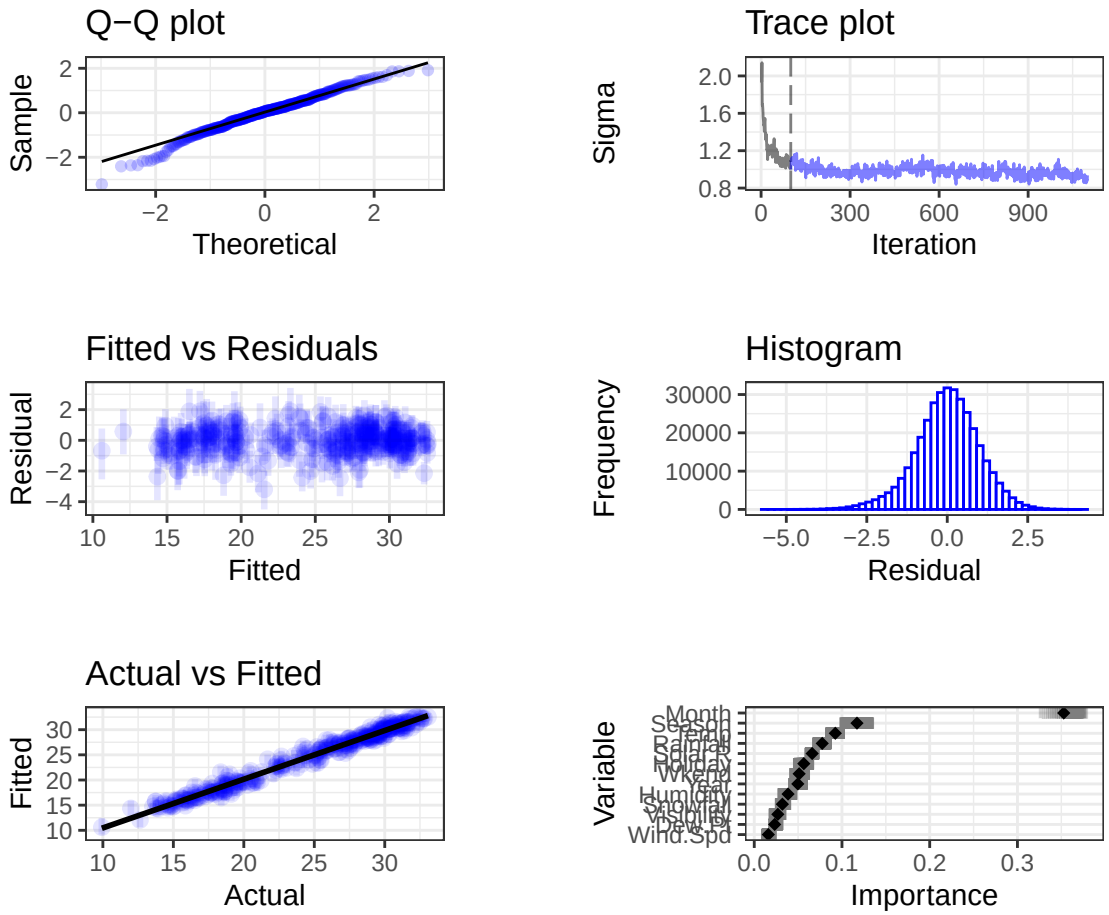
```r
dd <- as.data.frame(dd)
# BART model
xData <- dd[,-3]
yData <- dd[, 3]

set.seed(8642)
bt <- wbart(x.train = xData,
            y.train = yData,
            nskip = 100,
            ndpost = 1000, # MCMC iters
            nkeeptreedraws = 1000,
            ntree = 100
)
```

```
## *****Into main of wbart
## *****Data:
## data:n,p,np: 353, 30, 0
## y1,yn: -3.521673, 0.623425
## x1,x[n*p]: 0.000000, 0.000000
## *****Number of Trees: 100
## *****Number of Cut Points: 1 ... 1
## *****burn and ndpost: 100, 1000
## *****Prior:beta,alpha,tau,nu,lambda: 2.000000,0.950000,0.578474,3.000000,0.566823
## *****sigma: 1.705842
## *****w (weights): 1.000000 ... 1.000000
## *****Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,30,0
## *****nkeeptrain,nkeeptest,nkeeptestme,nkeeptreedraws: 1000,1000,1000,1000
## *****printevery: 100
## *****skiptr,skipte,skipteme,skiptreedraws: 1,1,1,1
```

```
##
## MCMC
## done 0 (out of 1100)
## done 100 (out of 1100)
## done 200 (out of 1100)
## done 300 (out of 1100)
## done 400 (out of 1100)
## done 500 (out of 1100)
## done 600 (out of 1100)
## done 700 (out of 1100)
## done 800 (out of 1100)
## done 900 (out of 1100)
## done 1000 (out of 1100)
## time: 2s
## check counts
## trcnt,tecnt,temecnt,treedrawscnt: 1000,0,0,1000
```

Figure 12:

```
# diagnostic plot
bartDiag(model = bt,
         response = dd$Count,
         burnIn = 100,
         combineFact = T,
         data = dd)
```



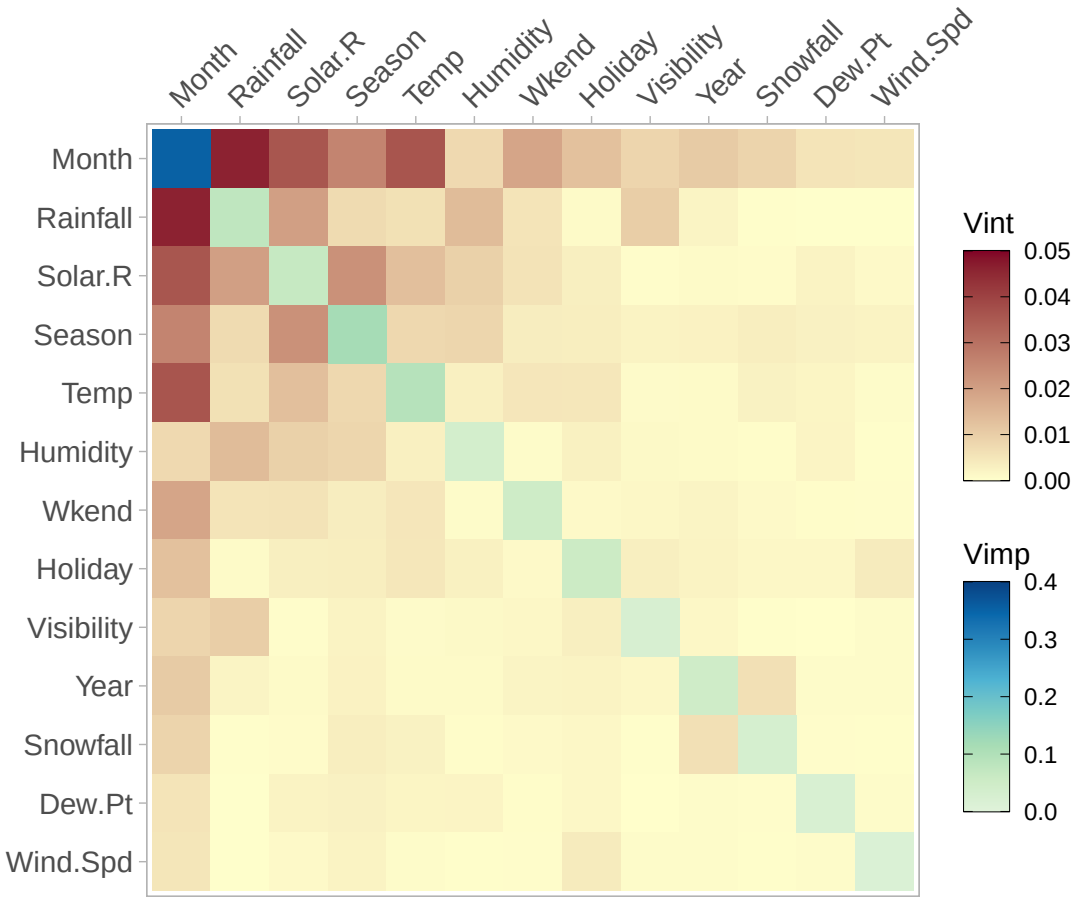
Create dataframe of trees

```
# tree df
btDF <- extractTreeData(model = bt, data = dd)
```

Figure 13:

```
myMat <- viviBartMatrix(btDF,
                        type = 'vsup',
                        metric = 'propMean',
                        metricError = "CV",
                        combineFact = T,
                        reorder = F)



colors <- scales::colour_ramp(
  colors = c(blue = '#FFFFCC', red = '#800026')
)((0:7)/7)
```

```r
newCols <- RColorBrewer::brewer.pal(9, 'GnBu')
colors2 <- newCols[-1]



# regular vivi
vivid::viviHeatmap(myMat$actualMatrix,
                   intPal = colors,
                   impPal = colors2,
                   angle = 45
)


# vsup
viviBartPlot(myMat,
             intPal = colors,
             impPal = colors2,
             max_desat = 1,
             pow_desat = 0.2,
             max_light = 0.6,
             pow_light = 1,
             label = 'CV'
) +
  theme(axis.text.x = element_text(hjust = 0, angle = 45))
```



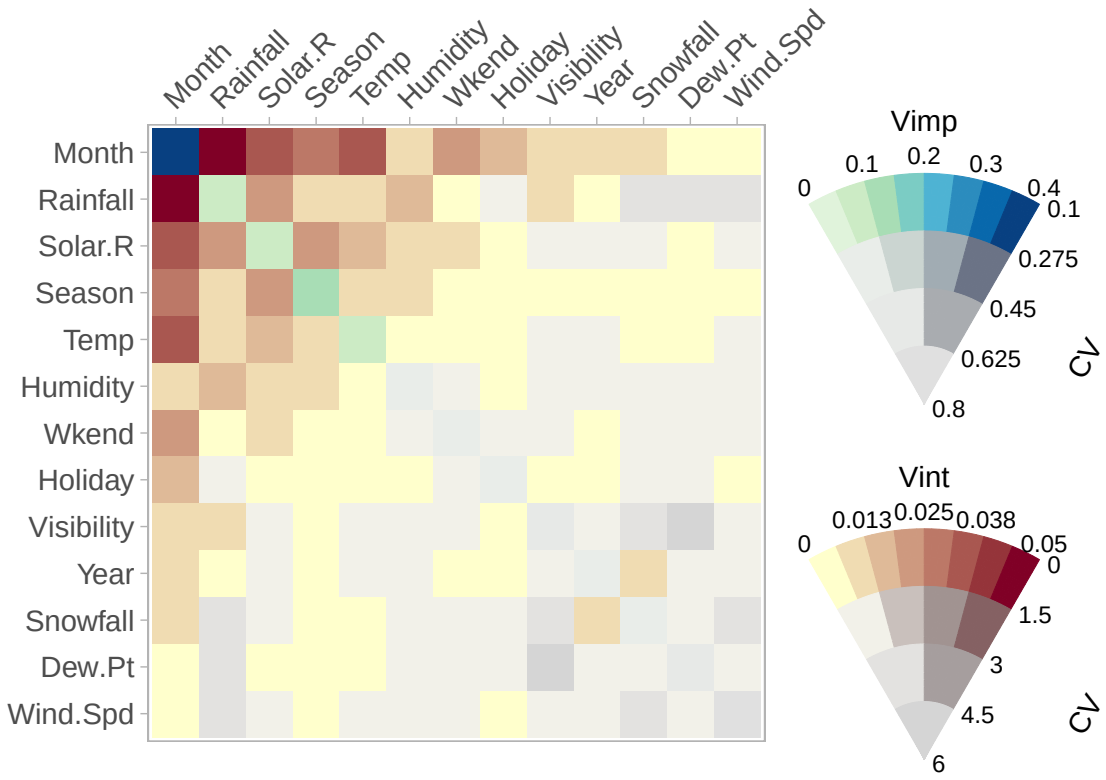


Figure 14:

```
# finding iteration with lowest residual sd
btPost <- bt$yhat.train

resid = NULL
for(i in 1:1000){
  resid[[i]] <- dd$Count - btPost[i,]
}
finalRes <- lapply(resid, sd)
lowRes <- which.min(finalRes)

# plot
plotAllTrees(btDF,
             iter = lowRes,
             selectedVars = c(15:26, 1:4, 7, 12, 13),
             cluster = 'var',
             removeStump = TRUE,
             combineFact = T)
```

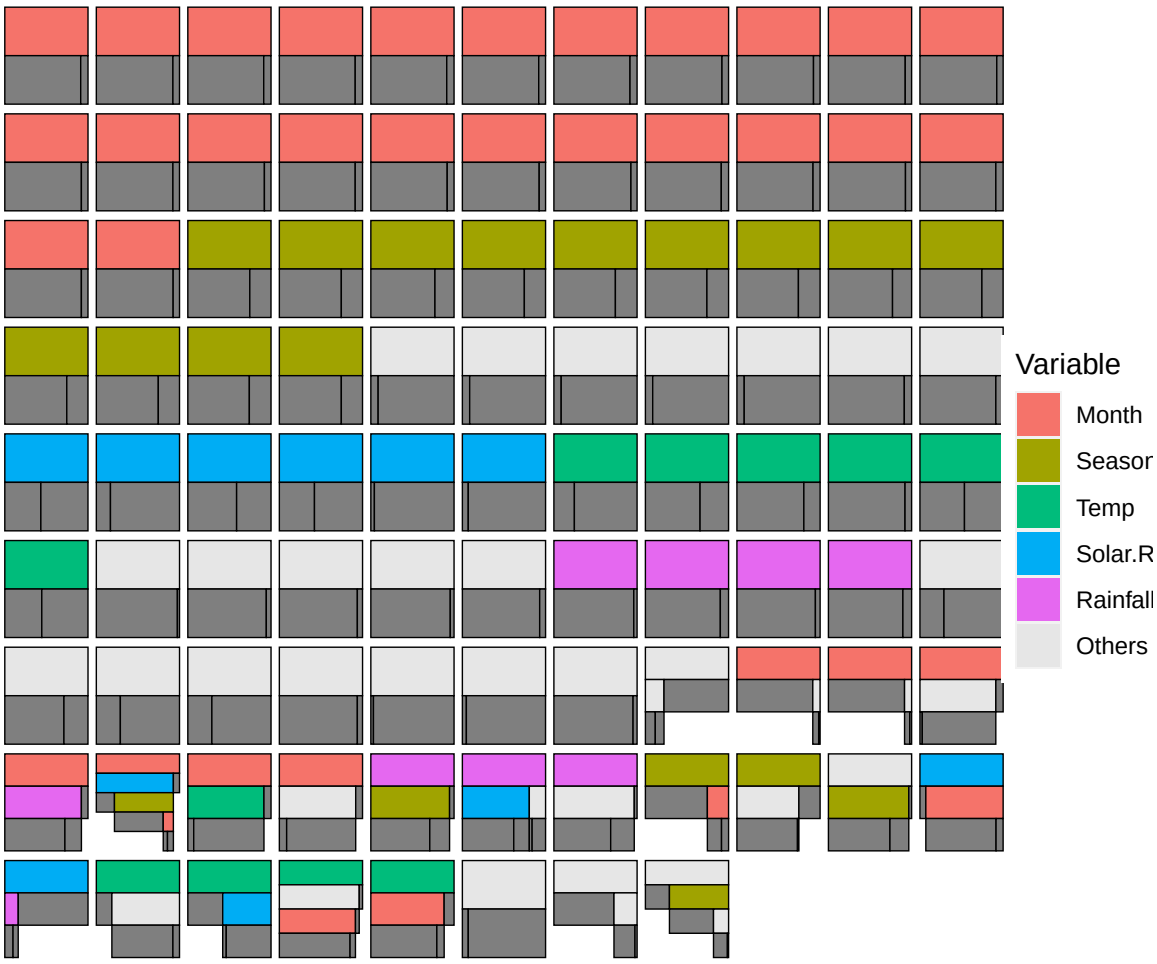


Figure 15:

```
# finding iteration with lowest residual sd
btPost <- bt$yhat.train

resid = NULL
for(i in 1:1000){
  resid[[i]] <- dd$Count - btPost[i,]
}
finalRes <- lapply(resid, sd)
lowRes <- which.min(finalRes)

# select target proximity matrix
bmProx <- proximityMatrix(btDF,
                          dd,
                          reorder = T,
                          normalize = T,
                          iter = lowRes)

# mds plot
mdsBart(treeData = btDF,
```

```
        data = dd,
        target = bmProx,
        plotType = 'interactive')
```