paige r markdown

2024-02-28

data combination from jorge

```
getwd()
```

```
## [1] "C:/Users/paige/OneDrive/Documents/STAT 472/Team-Koopa"
```

```
data1 <- read.csv("Criminal_Offenses_On_campus.csv") |>
  mutate(unique_id = paste0(OPEID, "_", Campus.ID)) |>
  rename_with(~ paste0(.x,"_all_campus"), recycle0 = TRUE) |>
  rename(Survey.year = Survey.year_all_campus, unique_id = unique_id_all_campus)

data2 <- read.csv("Criminal_Offenses_On_campus_Student_Housing_Facilities.csv") |>
  mutate(unique_id = paste0(OPEID, "_", Campus.ID)) |>
  rename_with(~ paste0(.x,"_student_housing"), recycle0 = TRUE) |>
  rename(Survey.year = Survey.year_student_housing, unique_id = unique_id_student_housing)

data3 <- read.csv("Criminal_Offenses_Noncampus.csv") |>
  mutate(unique_id = paste0(OPEID, "_", Campus.ID)) |>
  rename_with(~ paste0(.x, "_crim_offense_noncampus"), recycle0 = TRUE) |>
  rename(Survey.year = Survey.year_crim_offense_noncampus, unique_id = unique_id_crim_offense_noncampus)

data4 <- read.csv("Criminal_Offenses_Public_property.csv") |>
   mutate(unique_id = paste0(OPEID, "_", Campus.ID)) |>
  rename_with(~ paste0(.x, "_crim_offense_public"), recycle0 = TRUE) |>
  rename(Survey.year = Survey.year_crim_offense_public, unique_id = unique_id_crim_offense_public)

data5 <- read.csv("Arrests_On_campus.csv") |>
  mutate(unique_id = paste0(OPEID, "_", Campus.ID)) |>
  rename_with(~ paste0(.x, "_arrests_campus"), recycle0 = TRUE) |>
  rename(Survey.year = Survey.year_arrests_campus, unique_id = unique_id_arrests_campus)

data6 <- read.csv("Arrests_On_campus_Student_Housing_Facilities.csv") |>
  mutate(unique_id = paste0(OPEID, "_", Campus.ID)) |>
  rename_with(~ paste0(.x, "_arrests_stuhousing"), recycle0 = TRUE) |>
  rename(Survey.year = Survey.year_arrests_stuhousing, unique_id = unique_id_arrests_stuhousing)

data7 <- read.csv("Arrests_Noncampus.csv") |>
  mutate(unique_id = paste0(OPEID, "_", Campus.ID)) |>
  rename_with(~ paste0(.x, "_arrests_noncampus"), recycle0 = TRUE) |>
  rename(Survey.year = Survey.year_arrests_noncampus, unique_id = unique_id_arrests_noncampus)

data8 <- read.csv("Arrests_Public_Property.csv") |>
```

```
  mutate(unique_id = paste0(OPEID, "_", Campus.ID)) |>
  rename_with(~ paste0(.x, "_arrests_public"), recycle0 = TRUE) |>
  rename(Survey.year = Survey.year_arrests_public, unique_id = unique_id_arrests_public)

data9 <- read.csv("Disciplinary_Actions_On_campus.csv") |>
  mutate(unique_id = paste0(OPEID, "_", Campus.ID)) |>
  rename_with(~ paste0(.x, "_disciplinary_campus"), recycle0 = TRUE) |>
  rename(Survey.year = Survey.year_disciplinary_campus, unique_id = unique_id_disciplinary_campus)

data10 <- read.csv("Disciplinary_Actions_Student_Housing_Facilities.csv") |>
  mutate(unique_id = paste0(OPEID, "_", Campus.ID)) |>
  rename_with(~ paste0(.x, "_disciplinary_housing"), recycle0 = TRUE) |>
  rename(Survey.year = Survey.year_disciplinary_housing, unique_id = unique_id_disciplinary_housing)

data11 <- read.csv("Disciplinary_Actions_Noncampus.csv") |>
  mutate(unique_id = paste0(OPEID, "_", Campus.ID)) |>
  rename_with(~ paste0(.x, "_disciplinary_noncampus"), recycle0 = TRUE) |>
  rename(Survey.year = Survey.year_disciplinary_noncampus, unique_id = unique_id_disciplinary_noncampus

data12 <- read.csv("Disciplinary_Actions_Public_Property.csv") |>
  mutate(unique_id = paste0(OPEID, "_", Campus.ID)) |>
  rename_with(~ paste0(.x, "_disciplinary_public"), recycle0 = TRUE) |>
  rename(Survey.year = Survey.year_disciplinary_public, unique_id = unique_id_disciplinary_public)

# This is our datasets being joined into one
dataset <- data1 |> left_join(data2) |>
  left_join(data3) |>
  left_join(data4) |>
  left_join(data5) |>
  left_join(data6) |>
  left_join(data7) |>
  left_join(data8) |>
  left_join(data9) |>
  left_join(data10) |>
  left_join(data11) |>
  left_join(data12)
```

```
## Joining with ‘by = join_by(Survey.year, unique_id)‘
## Joining with ‘by = join_by(Survey.year, unique_id)‘
## Joining with ‘by = join_by(Survey.year, unique_id)‘
## Joining with ‘by = join_by(Survey.year, unique_id)‘
## Joining with ‘by = join_by(Survey.year, unique_id)‘
## Joining with ‘by = join_by(Survey.year, unique_id)‘
## Joining with ‘by = join_by(Survey.year, unique_id)‘
## Joining with ‘by = join_by(Survey.year, unique_id)‘
## Joining with ‘by = join_by(Survey.year, unique_id)‘
## Joining with ‘by = join_by(Survey.year, unique_id)‘
## Joining with ‘by = join_by(Survey.year, unique_id)‘
```

data cleaning

removing NA values, removing useless columns

```
#remove NAs
dataset[is.na(dataset)] <- 0

#remove repeated columns (like unitid repeating for each xcel file)
#(3/4/24) just fixed some problems w this

cols_to_remove <- c("Unitid_student_housing", "Institution.name_student_housing", "OPEID_student_housin
cleaned_data <- dataset[, !names(dataset) %in% cols_to_remove]
```

tests

```
mean(cleaned_data$Institution.Size_all_campus)
```

```
## [1] 10839.74
```

```
cleaned_data |> mutate(Institution.name_all_campus = as.factor(Institution.name_all_campus)) |> group_b
  summarize(num = length(unique(Campus.Name_all_campus)))
```

```
## ‘summarise()‘ has grouped output by ’Institution.name_all_campus’. You can
## override using the ‘.groups‘ argument.
```

```
## # A tibble: 915 x 3
## # Groups:   Institution.name_all_campus [86]
##    Institution.name_all_campus    Survey.year   num
##    <fct>                                <int> <int>
##  1 Academy of Natural Therapy Inc        2011     1
##  2 Academy of Natural Therapy Inc        2012     1
##  3 Academy of Natural Therapy Inc        2013     1
##  4 Academy of Natural Therapy Inc        2014     1
##  5 Academy of Natural Therapy Inc        2015     1
##  6 Academy of Natural Therapy Inc        2016     1
##  7 Academy of Natural Therapy Inc        2017     1
##  8 Academy of Natural Therapy Inc        2018     1
##  9 Academy of Natural Therapy Inc        2019     1
## 10 Academy of Natural Therapy Inc        2020     1
## # i 905 more rows
```

summary stats

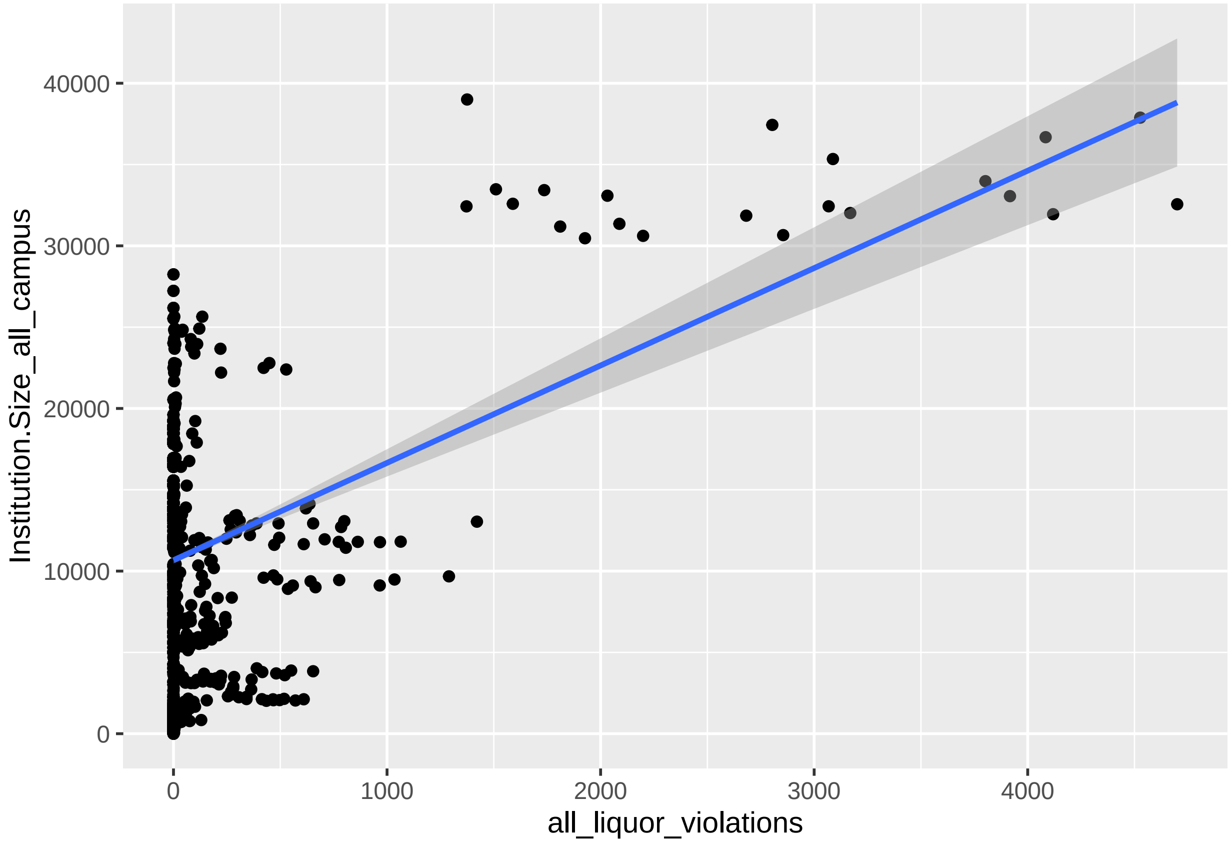
```
#new column combining liquor law violations across disciplinary, arrests and location (public, stuhousi
cleaned_data$all_liquor_violations <- cleaned_data$Liquor.law.violations_arrests_campus + cleaned_data$

numeric_data <- select(cleaned_data, where(is.numeric))

# figure margins too large
#pairs(numeric_data)

ggplot(cleaned_data, aes(y=Institution.Size_all_campus, x=all_liquor_violations)) +
  geom_point() +
  geom_smooth(method = "lm")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

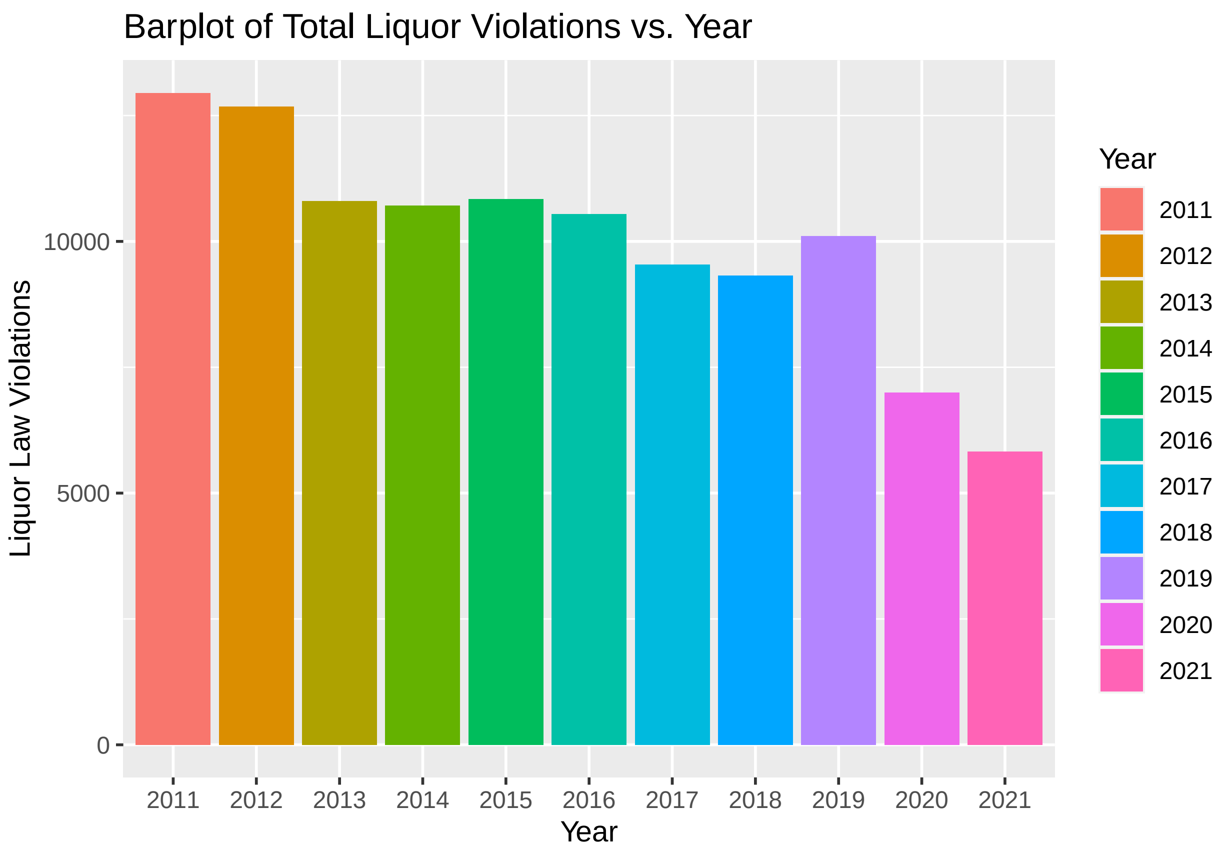


```
#cleaned_data$all_liquor_violations[16]

year_factor <- as.factor(cleaned_data$Survey.year)

#ggplot(cleaned_data, aes(x=year_factor, y=all_liquor_violations)) +
 # geom_bar(stat= "identity", aes(fill=year_factor)) +
  #xlab("Year") +
  #ylab("Liquor Law Violations") +
  #ggtitle("Barplot of Total Liquor Violations v. Year")

ggplot(cleaned_data, aes(x = year_factor, y = all_liquor_violations, fill = year_factor)) +
  geom_bar(stat = "identity") +
  labs(x = "Year", y = "Liquor Law Violations", fill = "Year") +
  ggtitle("Barplot of Total Liquor Violations vs. Year")
```



```
#ggplot(cleaned_data, aes(Institution.Size_all_campus)) +
 # geom_histogram() +
 # xlab("Institution Enrollment") +
 # ylab("Count of Campuses") +
 # ggtitle("Histogram of Institution Size")

#small_inst <- cleaned_data$Institution.Size_all_campus <= 15000
#large_inst <- cleaned_data$Institution.Size_all_campus >= 15001
#large_inst <- cleaned_data$Institution.Size_all_campus >= 20000

#size_data <- data.frame()

#size_table <- data.frame(
 # Small = sum(small_inst),
 # Large = sum(large_inst)
#)

#kable(size_table, caption = "Institution Sizes")

#ggplot(cleaned_data, aes(y=all_liquor_violations, x=unique_id)) +
  #geom_boxplot()

#summary(lm(all_liquor_violations ~ ., data= cleaned_data)) +
#knitr::kable(digits=c(3,3,3,3),
#caption ="Simple Linear Regression Model Estimating Liquor Violations from All Predictors")
```

split data

```
set.seed(4242)

## split cleaned data into 25/75
smp_size <- floor(0.75 * nrow(cleaned_data))

train_split <- sample(seq_len(nrow(cleaned_data)), size = smp_size)

# create train = 75% and test = 25% set
train <- cleaned_data[train_split,] |> as_tibble() |> mutate(train = TRUE)
test <- cleaned_data[-train_split,] |> as_tibble() |> mutate(train = FALSE)


## check split to ensure nothing got screwed up

# create df of training data means and sd of each column
train_means_sd <- sapply(train[,c(7:20, 22:86)],
                         function(x) c(mean(x, na.rm = TRUE),
                                       sd(x, na.rm=TRUE)),
                         simplify = FALSE) |> bind_rows()
# transpose so table is legible
ttrain_means_sd <- t(train_means_sd)
# create kable table
#knitr::kable(ttrain_means_sd, digits = 5, caption = "Training Data, metrics to compare to test", col.n

# create df of testing data means and sd of each column
test_means_sd <- sapply(test[,c(7:20, 22:86)],
                         function(x) c(mean(x, na.rm = TRUE),
                                       sd(x, na.rm=TRUE)),
                         simplify = FALSE) |> bind_rows()
ttest_means_sd <- t(test_means_sd)
#knitr::kable(ttest_means_sd, digits = 5, caption = "Test Data, metrics to compare to training", col.na


## kable tables for hw 5

train_means <- round(c(mean(train$Negligent.manslaughter_all_campus),
           mean(train$Sex.offenses...Forcible_all_campus),
           mean(train$Rape_all_campus),
           mean(train$Fondling_all_campus),
           mean(train$Sex.offenses...Non.forcible_all_campus),
           mean(train$Incest_all_campus),
           mean(train$Statutory.rape_all_campus),
           mean(train$Robbery_all_campus),
           mean(train$Burglary_all_campus),
           mean(train$Motor.vehicle.theft_all_campus),
           mean(train$Arson_all_campus)), 3)

train_sds <- round(c(
  sd(train$Negligent.manslaughter_all_campus),
  sd(train$Sex.offenses...Forcible_all_campus),
  sd(train$Rape_all_campus),
  sd(train$Fondling_all_campus),
  sd(train$Sex.offenses...Non.forcible_all_campus),
  sd(train$Incest_all_campus),
  sd(train$Statutory.rape_all_campus),
  sd(train$Robbery_all_campus),
  sd(train$Burglary_all_campus),
  sd(train$Motor.vehicle.theft_all_campus),
  sd(train$Arson_all_campus)
), 3)

train_pres <- data.frame(
  Variable  = c("Negligent Manslaughter", "Sex Offenses (Forcible)", "Rape",
               "Fondling", "Sex Offenses (Non-forcible)", "Incest",
               "Statutory Rape", "Robbery", "Burglary", "Motor Vehicle Theft",
               "Arson"),
  Mean = train_means,
  StandardDeviation = train_sds
)

knitr::kable(train_pres, caption = "Training Data", col.names = c("Variable", "Mean", "SD"))
```

Table 1: Training Data

Variable	Mean	SD
Negligent Manslaughter	0.000	0.000
Sex Offenses (Forcible)	0.056	0.630
Rape	0.182	1.250
Fondling	0.118	0.829
Sex Offenses (Non-forcible)	0.000	0.000
Incest	0.000	0.000
Statutory Rape	0.001	0.027
Robbery	0.045	0.323
Burglary	0.591	3.170
Motor Vehicle Theft	0.293	2.026
Arson	0.040	0.385

```
test_means <- round(c(mean(test$Negligent.manslaughter_all_campus),
           mean(test$Sex.offenses...Forcible_all_campus),
           mean(test$Rape_all_campus),
           mean(test$Fondling_all_campus),
           mean(test$Sex.offenses...Non.forcible_all_campus),
           mean(test$Incest_all_campus),
           mean(test$Statutory.rape_all_campus),
           mean(test$Robbery_all_campus),
           mean(test$Burglary_all_campus),
           mean(test$Motor.vehicle.theft_all_campus),
           mean(test$Arson_all_campus)), 3)

test_sds <- round(c(
  sd(test$Negligent.manslaughter_all_campus),
  sd(test$Sex.offenses...Forcible_all_campus),
  sd(test$Rape_all_campus),
```

```r
  sd(test$Fondling_all_campus),
  sd(test$Sex.offenses...Non.forcible_all_campus),
  sd(test$Incest_all_campus),
  sd(test$Statutory.rape_all_campus),
  sd(test$Robbery_all_campus),
  sd(test$Burglary_all_campus),
  sd(test$Motor.vehicle.theft_all_campus),
  sd(test$Arson_all_campus)
), 3)

test_pres <- data.frame(
  Variable  = c("Negligent Manslaughter", "Sex Offenses (Forcible)", "Rape",
               "Fondling", "Sex Offenses (Non-forcible)", "Incest",
               "Statutory Rape", "Robbery", "Burglary", "Motor Vehicle Theft",
               "Arson"),
  Mean = test_means,
  StandardDeviation = test_sds
)

knitr::kable(test_pres, caption = "Test Data", col.names = c("Variable", "Mean", "SD"))
```

Table 2: Test Data

Variable	Mean	SD
Negligent Manslaughter	0.000	0.000
Sex Offenses (Forcible)	0.037	0.465
Rape	0.218	1.389
Fondling	0.150	1.026
Sex Offenses (Non-forcible)	0.001	0.033
Incest	0.000	0.000
Statutory Rape	0.001	0.033
Robbery	0.049	0.377
Burglary	0.631	3.531
Motor Vehicle Theft	0.307	1.754
Arson	0.044	0.406

neural network

plot.nnet

```r
## for plot.nnet()
#install.packages("devtools")
library(devtools)
```

```
## Warning: package 'devtools' was built under R version 4.3.3
```

```
## Loading required package: usethis
```

```
## Warning: package 'usethis' was built under R version 4.3.3
```

```
#install.packages("reshape")
library(reshape)
```

```
## Warning: package 'reshape' was built under R version 4.3.3

##
## Attaching package: 'reshape'

## The following object is masked from 'package:lubridate':
##
##     stamp

## The following objects are masked from 'package:tidyr':
##
##     expand, smiths

## The following object is masked from 'package:dplyr':
##
##     rename
```

```
# import plot nnet function from github
plot.nnet <- function(mod.in,nid=T,all.out=T,all.in=T,bias=T,wts.only=F,rel.rsc=5,circle.cex=5,
                    node.labs=T,var.labs=T,x.lab=NULL,y.lab=NULL,line.stag=NULL,struct=NULL,cex.val=1,
                    alpha.val=1,circle.col='lightblue',pos.col='black',neg.col='grey', max.sp = F, ...)

  require(scales)

  #sanity checks
  if('mlp' %in% class(mod.in)) warning('Bias layer not applicable for rsnns object')
  if('numeric' %in% class(mod.in)){
    if(is.null(struct)) stop('Three-element vector required for struct')
    if(length(mod.in) != ((struct[1]*struct[2]+struct[2]*struct[3])+(struct[3]+struct[2])))
      stop('Incorrect length of weight matrix for given network structure')
  }
  if('train' %in% class(mod.in)){
    if('nnet' %in% class(mod.in$finalModel)){
      mod.in<-mod.in$finalModel
      warning('Using best nnet model from train output')
    }
    else stop('Only nnet method can be used with train object')
  }

  #gets weights for neural network, output is list
  #if rescaled argument is true, weights are returned but rescaled based on abs value
  nnet.vals<-function(mod.in,nid,rel.rsc,struct.out=struct){

    require(scales)
    require(reshape)

    if('numeric' %in% class(mod.in)){
      struct.out<-struct
      wts<-mod.in
```

```r
}

#neuralnet package
if('nn' %in% class(mod.in)){
  struct.out<-unlist(lapply(mod.in$weights[[1]],ncol))
    struct.out<-struct.out[-length(struct.out)]
    struct.out<-c(
        length(mod.in$model.list$variables),
        struct.out,
        length(mod.in$model.list$response)
        )
  wts<-unlist(mod.in$weights[[1]])
}

#nnet package
if('nnet' %in% class(mod.in)){
  struct.out<-mod.in$n
  wts<-mod.in$wts
}

#RSNNS package
if('mlp' %in% class(mod.in)){
  struct.out<-c(mod.in$nInputs,mod.in$archParams$size,mod.in$nOutputs)
  hid.num<-length(struct.out)-2
  wts<-mod.in$snnsObject$getCompleteWeightMatrix()

  #get all input-hidden and hidden-hidden wts
  inps<-wts[grep('Input',row.names(wts)),grep('Hidden_2',colnames(wts)),drop=F]
  inps<-melt(rbind(rep(NA,ncol(inps)),inps))$value
  uni.hids<-paste0('Hidden_',1+seq(1,hid.num))
  for(i in 1:length(uni.hids)){
    if(is.na(uni.hids[i+1])) break
    tmp<-wts[grep(uni.hids[i],rownames(wts)),grep(uni.hids[i+1],colnames(wts)),drop=F]
    inps<-c(inps,melt(rbind(rep(NA,ncol(tmp)),tmp))$value)
    }

  #get connections from last hidden to output layers
  outs<-wts[grep(paste0('Hidden_',hid.num+1),row.names(wts)),grep('Output',colnames(wts)),drop=F]
  outs<-rbind(rep(NA,ncol(outs)),outs)

  #weight vector for all
  wts<-c(inps,melt(outs)$value)
  assign('bias',F,envir=environment(nnet.vals))
  }

if(nid) wts<-rescale(abs(wts),c(1,rel.rsc))

#convert wts to list with appropriate names
hid.struct<-struct.out[-c(length(struct.out))]
row.nms<-NULL
for(i in 1:length(hid.struct)){
  if(is.na(hid.struct[i+1])) break
  row.nms<-c(row.nms,rep(paste('hidden',i,seq(1:hid.struct[i+1])),each=1+hid.struct[i]))
```

```
    }
    row.nms<-c(
      row.nms,
      rep(paste('out',seq(1:struct.out[length(struct.out)])),each=1+struct.out[length(struct.out)-1])
      )
    out.ls<-data.frame(wts,row.nms)
    out.ls$row.nms<-factor(row.nms,levels=unique(row.nms),labels=unique(row.nms))
    out.ls<-split(out.ls$wts,f=out.ls$row.nms)

    assign('struct',struct.out,envir=environment(nnet.vals))

    out.ls

    }

  wts<-nnet.vals(mod.in,nid=F)

  if(wts.only) return(wts)

  #circle colors for input, if desired, must be two-vector list, first vector is for input layer
  if(is.list(circle.col)){
                    circle.col.inp<-circle.col[[1]]
                    circle.col<-circle.col[[2]]
                    }
  else circle.col.inp<-circle.col

  #initiate plotting
  x.range<-c(0,100)
  y.range<-c(0,100)
  #these are all proportions from 0-1
  if(is.null(line.stag)) line.stag<-0.011*circle.cex/2
  layer.x<-seq(0.17,0.9,length=length(struct))
  bias.x<-layer.x[-length(layer.x)]+diff(layer.x)/2
  bias.y<-0.95
  circle.cex<-circle.cex

  #get variable names from mod.in object
  #change to user input if supplied
  if('numeric' %in% class(mod.in)){
    x.names<-paste0(rep('X',struct[1]),seq(1:struct[1]))
    y.names<-paste0(rep('Y',struct[3]),seq(1:struct[3]))
  }
  if('mlp' %in% class(mod.in)){
    all.names<-mod.in$snnsObject$getUnitDefinitions()
    x.names<-all.names[grep('Input',all.names$unitName),'unitName']
    y.names<-all.names[grep('Output',all.names$unitName),'unitName']
  }
  if('nn' %in% class(mod.in)){
    x.names<-mod.in$model.list$variables
    y.names<-mod.in$model.list$respons
  }
  if('xNames' %in% names(mod.in)){
    x.names<-mod.in$xNames
    y.names<-attr(terms(mod.in),'factor')
    y.names<-row.names(y.names)[!row.names(y.names) %in% x.names]
  }
  if(!'xNames' %in% names(mod.in) & 'nnet' %in% class(mod.in)){
    if(is.null(mod.in$call$formula)){
      x.names<-colnames(eval(mod.in$call$x))
      y.names<-colnames(eval(mod.in$call$y))
    }
    else{
      forms<-eval(mod.in$call$formula)
      x.names<-mod.in$coefnames
      facts<-attr(terms(mod.in),'factors')
      y.check<-mod.in$fitted
      if(ncol(y.check)>1) y.names<-colnames(y.check)
      else y.names<-as.character(forms)[2]
    }
  }
  #change variables names to user sub
  if(!is.null(x.lab)){
    if(length(x.names) != length(x.lab)) stop('x.lab length not equal to number of input variables')
    else x.names<-x.lab
  }
  if(!is.null(y.lab)){
    if(length(y.names) != length(y.lab)) stop('y.lab length not equal to number of output variables')
    else y.names<-y.lab
  }

  #initiate plot
  plot(x.range,y.range,type='n',axes=F,ylab='',xlab='',...)

  #function for getting y locations for input, hidden, output layers
  #input is integer value from 'struct'
  get.ys<-function(lyr, max_space = max.sp){
    if(max_space){
        spacing <- diff(c(0*diff(y.range),0.9*diff(y.range)))/lyr
    } else {
        spacing<-diff(c(0*diff(y.range),0.9*diff(y.range)))/max(struct)
    }

        seq(0.5*(diff(y.range)+spacing*(lyr-1)),0.5*(diff(y.range)-spacing*(lyr-1)),
        length=lyr)
  }

  #function for plotting nodes
  #'layer' specifies which layer, integer from 'struct'
  #'x.loc' indicates x location for layer, integer from 'layer.x'
  #'layer.name' is string indicating text to put in node
  layer.points<-function(layer,x.loc,layer.name,cex=cex.val){
    x<-rep(x.loc*diff(x.range),layer)
    y<-get.ys(layer)
    points(x,y,pch=21,cex=circle.cex,col=in.col,bg=bord.col)
    if(node.labs) text(x,y,paste(layer.name,1:layer,sep=''),cex=cex.val)
    if(layer.name=='I' & var.labs) text(x-line.stag*diff(x.range),y,x.names,pos=2,cex=cex.val)
```

```r
  if(layer.name=='O' & var.labs) text(x+line.stag*diff(x.range),y,y.names,pos=4,cex=cex.val)
}

#function for plotting bias points
#'bias.x' is vector of values for x locations
#'bias.y' is vector for y location
#'layer.name' is  string indicating text to put in node
bias.points<-function(bias.x,bias.y,layer.name,cex,...){
  for(val in 1:length(bias.x)){
    points(
      diff(x.range)*bias.x[val],
      bias.y*diff(y.range),
      pch=21,col=in.col,bg=bord.col,cex=circle.cex
    )
    if(node.labs)
      text(
        diff(x.range)*bias.x[val],
        bias.y*diff(y.range),
        paste(layer.name,val,sep=''),
        cex=cex.val
      )
  }
}

#function creates lines colored by direction and width as proportion of magnitude
#use 'all.in' argument if you want to plot connection lines for only a single input node
layer.lines<-function(mod.in,h.layer,layer1=1,layer2=2,out.layer=F,nid,rel.rsc,all.in,pos.col,
                      neg.col,...){

  x0<-rep(layer.x[layer1]*diff(x.range)+line.stag*diff(x.range),struct[layer1])
  x1<-rep(layer.x[layer2]*diff(x.range)-line.stag*diff(x.range),struct[layer1])

  if(out.layer==T){

    y0<-get.ys(struct[layer1])
    y1<-rep(get.ys(struct[layer2])[h.layer],struct[layer1])
    src.str<-paste('out',h.layer)

    wts<-nnet.vals(mod.in,nid=F,rel.rsc)
    wts<-wts[grep(src.str,names(wts))][[1]][-1]
    wts.rs<-nnet.vals(mod.in,nid=T,rel.rsc)
    wts.rs<-wts.rs[grep(src.str,names(wts.rs))][[1]][-1]

    cols<-rep(pos.col,struct[layer1])
    cols[wts<0]<-neg.col

    if(nid) segments(x0,y0,x1,y1,col=cols,lwd=wts.rs)
    else segments(x0,y0,x1,y1)

  }

  else{
    if(is.logical(all.in)) all.in<-h.layer
    else all.in<-which(x.names==all.in)

    y0<-rep(get.ys(struct[layer1])[all.in],struct[2])
    y1<-get.ys(struct[layer2])
    src.str<-paste('hidden',layer1)

    wts<-nnet.vals(mod.in,nid=F,rel.rsc)
    wts<-unlist(lapply(wts[grep(src.str,names(wts))],function(x) x[all.in+1]))
    wts.rs<-nnet.vals(mod.in,nid=T,rel.rsc)
    wts.rs<-unlist(lapply(wts.rs[grep(src.str,names(wts.rs))],function(x) x[all.in+1]))

    cols<-rep(pos.col,struct[layer2])
    cols[wts<0]<-neg.col

    if(nid) segments(x0,y0,x1,y1,col=cols,lwd=wts.rs)
    else segments(x0,y0,x1,y1)

  }

}

bias.lines<-function(bias.x,mod.in,nid,rel.rsc,all.out,pos.col,neg.col,...){

  if(is.logical(all.out)) all.out<-1:struct[length(struct)]
  else all.out<-which(y.names==all.out)

  for(val in 1:length(bias.x)){

    wts<-nnet.vals(mod.in,nid=F,rel.rsc)
    wts.rs<-nnet.vals(mod.in,nid=T,rel.rsc)

      if(val != length(bias.x)){
      wts<-wts[grep('out',names(wts),invert=T)]
      wts.rs<-wts.rs[grep('out',names(wts.rs),invert=T)]
          sel.val<-grep(val,substr(names(wts.rs),8,8))
          wts<-wts[sel.val]
          wts.rs<-wts.rs[sel.val]
          }

      else{
      wts<-wts[grep('out',names(wts))]
      wts.rs<-wts.rs[grep('out',names(wts.rs))]
      }

    cols<-rep(pos.col,length(wts))
    cols[unlist(lapply(wts,function(x) x[1]))<0]<-neg.col
    wts.rs<-unlist(lapply(wts.rs,function(x) x[1]))

    if(nid==F){
      wts.rs<-rep(1,struct[val+1])
      cols<-rep('black',struct[val+1])
    }
    if(val != length(bias.x)){
      segments(
        rep(diff(x.range)*bias.x[val]+diff(x.range)*line.stag,struct[val+1]),
        rep(bias.y*diff(y.range),struct[val+1]),
        rep(diff(x.range)*layer.x[val+1]-diff(x.range)*line.stag,struct[val+1]),
        get.ys(struct[val+1]),
        lwd=wts.rs,
        col=cols
      )
    }

    else{
      segments(
        rep(diff(x.range)*bias.x[val]+diff(x.range)*line.stag,struct[val+1]),
        rep(bias.y*diff(y.range),struct[val+1]),
        rep(diff(x.range)*layer.x[val+1]-diff(x.range)*line.stag,struct[val+1]),
        get.ys(struct[val+1])[all.out],
        lwd=wts.rs[all.out],
        col=cols[all.out]
      )
    }

  }
}

#use functions to plot connections between layers
#bias lines
if(bias) bias.lines(bias.x,mod.in,nid=nid,rel.rsc=rel.rsc,all.out=all.out,pos.col=alpha(pos.col,alpha
                    neg.col=alpha(neg.col,alpha.val))

#layer lines, makes use of arguments to plot all or for individual layers
#starts with input-hidden
#uses 'all.in' argument to plot connection lines for all input nodes or a single node
if(is.logical(all.in)){
  mapply(
    function(x) layer.lines(mod.in,x,layer1=1,layer2=2,nid=nid,rel.rsc=rel.rsc,
      all.in=all.in,pos.col=alpha(pos.col,alpha.val),neg.col=alpha(neg.col,alpha.val)),
    1:struct[1]
  )
}
else{
  node.in<-which(x.names==all.in)
  layer.lines(mod.in,node.in,layer1=1,layer2=2,nid=nid,rel.rsc=rel.rsc,all.in=all.in,
              pos.col=alpha(pos.col,alpha.val),neg.col=alpha(neg.col,alpha.val))
}
#connections between hidden layers
lays<-split(c(1,rep(2:(length(struct)-1),each=2),length(struct)),
            f=rep(1:(length(struct)-1),each=2))
lays<-lays[-c(1,(length(struct)-1))]
for(lay in lays){
  for(node in 1:struct[lay[1]]){
    layer.lines(mod.in,node,layer1=lay[1],layer2=lay[2],nid=nid,rel.rsc=rel.rsc,all.in=T,
                pos.col=alpha(pos.col,alpha.val),neg.col=alpha(neg.col,alpha.val))
    }
  }
  #lines for hidden-output
  #uses 'all.out' argument to plot connection lines for all output nodes or a single node
  if(is.logical(all.out))
    mapply(
      function(x) layer.lines(mod.in,x,layer1=length(struct)-1,layer2=length(struct),out.layer=T,nid=ni
                              all.in=all.in,pos.col=alpha(pos.col,alpha.val),neg.col=alpha(neg.col,alpha
      1:struct[length(struct)]
      )
  else{
    node.in<-which(y.names==all.out)
    layer.lines(mod.in,node.in,layer1=length(struct)-1,layer2=length(struct),out.layer=T,nid=nid,rel.rs
                pos.col=pos.col,neg.col=neg.col,all.out=all.out)
  }

  #use functions to plot nodes
  for(i in 1:length(struct)){
    in.col<-bord.col<-circle.col
    layer.name<-'H'
    if(i==1) { layer.name<-'I'; in.col<-bord.col<-circle.col.inp}
    if(i==length(struct)) layer.name<-'O'
    layer.points(struct[i],layer.x[i],layer.name)
    }

  if(bias) bias.points(bias.x,bias.y,'B')

}
```

fit lasso for predictors

```
set.seed(4242)

#for lasso
#install.packages("glmnet")
library(glmnet)

train_num <- dplyr::select_if(train, is.numeric)

#specify y
y <- train_num$all_liquor_violations

train$Liquor
```

```
## Warning: Unknown or uninitialised column: `Liquor`.
```

```
## NULL
```

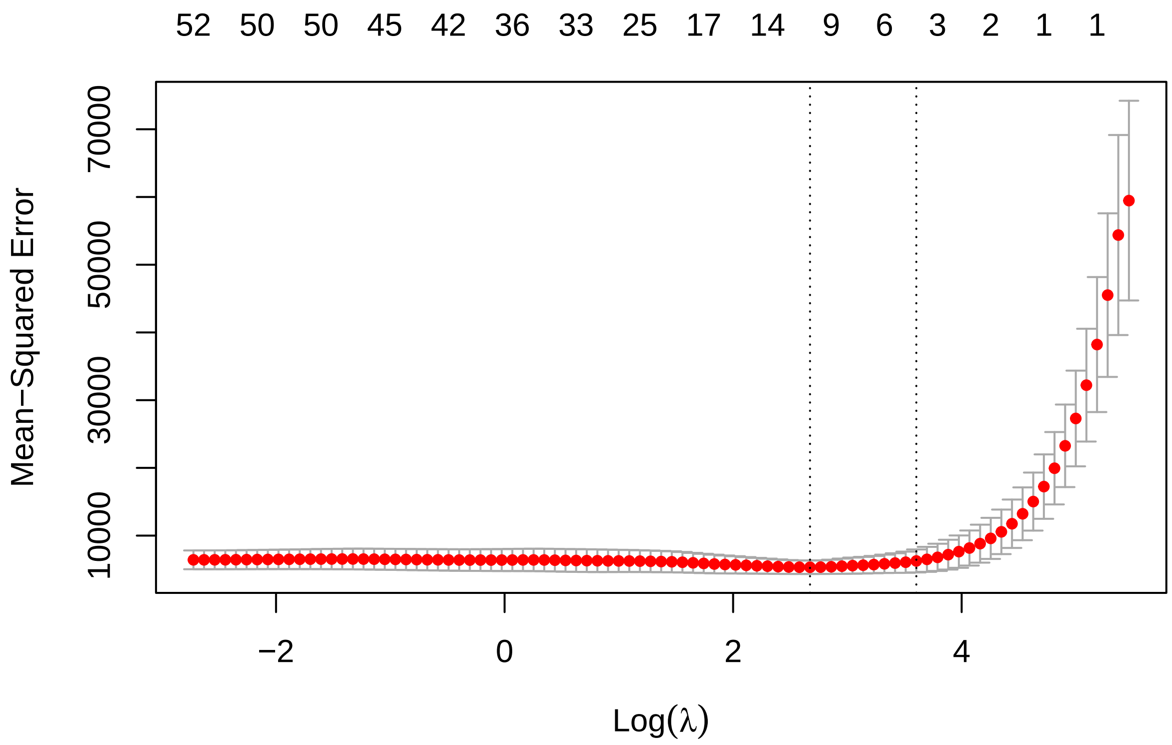
```
exclude_columns <- c("Unitid_all_campus", "OPEID_all_campus",
                    "Campus.ID_all_campus", "all_liquor_violations",
                    "Liquor.law.violations_arrests_campus",
                    "Liquor.law.violations_arrests_public",
                    "Liquor.law.violations_arrests_noncampus",
                    "Liquor.law.violations_arrests_stuhousing",
                    "Liquor.law.violations_disciplinary_campus",
                    "Liquor.law.violations_disciplinary_noncampus",
                    "Liquor.law.violations_disciplinary_public",
                    "Liquor.law.violations_disciplinary_housing",
                    "new_column")

train_finalset <- train_num[, !names(train_num) %in% exclude_columns]

#specify x
x <- data.matrix(train_finalset)


# k fold cv for lambda
cv_model <- cv.glmnet(x,y,alpha = 1)
best_lambda <- cv_model$lambda.min
#best_lambda

plot(cv_model)
```



```
#find optimal lasso model
best_lasso <- glmnet(x, y, alpha = 1, lambda = best_lambda)
```

```r
#coefficients from lasso model
lasso_coef <- coef(best_lasso)

#lasso_coef

#make coefficients matrix
lc_mat <- as.matrix(lasso_coef)

#make coefficients dataframe
lc_df <- as.data.frame(lc_mat)

#filter out coefficients that are 0
rows_to_keep <- apply(lc_mat, 1, function(row) any(row > 0))

lc_df_filtered <- lc_df[rows_to_keep,]

#remove intercept
lc_df_clean <- lc_df_filtered[-1]

#lc_df_clean

lc_table_df <- data.frame(
  Variable = c("Rape (student housing)", "Fondling (student housing)", "Aggravated Assault (student hou
  Coefficients = lc_df_clean)

#table of lasso coefficients
knitr::kable(lc_table_df, caption = "LASSO Coefficients", digits = 3)
```

Table 3: LASSO Coefficients

Variable	Coefficients
Rape (student housing)	15.049
Fondling (student housing)	1.229
Aggravated Assault (student housing)	6.897
Burglary (student housing)	16.604
Arson (student housing)	40.319
Burglary (noncampus)	6.201
Drug Law Violations (arrests, campus)	0.995
Drug Law Violations (arrests, noncampus)	6.513
Drug Law Violations (disciplinary, campus)	0.772
Drug Law Violations (disciplinary, housing)	2.709

best_lambda = 17.44531 lasso coefficients found are liquor.law.violations… -> must remove these cols

Coeffs found: rape_student_housing, burglary_student_housing, arson_student_housing, burglarly_crim_offense_noncampus, drug.law.violation_arrests_campus, drug.law.violations_arrests_noncampus, drug.law.violations_disciplinary_campus, drug.law.violations_disciplinary_housing

fit nets

```r
## potential libraries

#install.packages("keras")
library(keras)
library(tensorflow)
library(nnet)

#install.packages("neuralnet")

#compute object is masked from package:dplyr
library(neuralnet)

#get plots side by side, grid.arrange()
#install.packages("gridExtra")
library(gridExtra)

#for dredge()
#install.packages("MuMIn")
library(MuMIn)

# set seed for reproducibility
set.seed(4242)

snn_1 <- neuralnet(all_liquor_violations ~ Institution.Size_all_campus + Survey.year, data = train, hidd

#plot(snn_1, rep= "best")

plot.nnet(snn_1, x.lab = c("Size", "Year"), y.lab = "TLV")
```

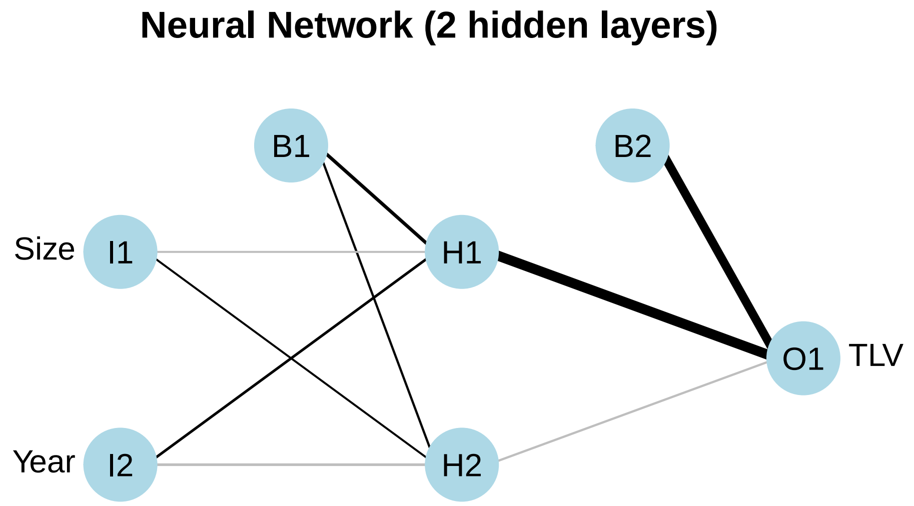
```
## Loading required package: scales

##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor
```

```r
title("Neural Network (2 hidden layers)")
```


Neural Network (2 hidden layers)

```
#snn_3 <- neuralnet(all_liquor_violations ~ Institution.Size_all_campus + Survey.year , data = train, h

#plot(snn_3)

#par(mfrow = c(1,2))
#plot1 <- plot(snn_3, rep = "best")
#plot2 <- plot.nnet(snn_3)
#grid.arrange(plot1, plot2, nrow = 1)

#snn_2 <- neuralnet(all_liquor_violations ~. - Unitid_all_campus - OPEID_all_campus - Campus.ID_all_cam

#plot(snn_2)

#plot.nnet(snn_2)

#train_num_nomiss <- train_num %>%
 # mutate(train_num = if_else(is.na(train_num), "Other", train_num)) %>%
  #drop_na()


#train_num <- na.omit(train_num)

# AIC trials -- failed bc max pred in dredge is 31
#options(na.action = "na.fail")
#full_model <- lm(all_liquor_violations ~ . - Unitid_all_campus - OPEID_all_campus - Campus.ID_all_camp
#model_dredge <- dredge(full_model, rank = "AIC", extra = c("R^2"))
```

```
#NN_model <- neuralnet(all_liquor_violations ~ Rape_student_housing + Burglary_student_housing + Arson_

#w <- NN_model$weights
#w

#plot(NN_model)
#plot.nnet(NN_model)
```

Interpreting plot.nn() output: gray lines indicate negative weight, black: pos weight. the thicker the lines, the greater the |weight|.

this code is useless

```
#read CSV files

#arrests_local_state_police <- read.csv("C:/Users/paige/Downloads/OPE CSS Custom Data 2024-02-28 171746/

#arrests_noncampus <- read.csv("C:/Users/paige/Downloads/OPE CSS Custom Data 2024-02-28 171746/Arrests_

#arrests_oncampus <- read.csv("C:/Users/paige/Downloads/OPE CSS Custom Data 2024-02-28 171746/Arrests_O

##problem: need to combine based on university and campus - make new id

# Create a new column by combining "userID" and "campusID"
#arrests_local_state_police$new_ID <- paste(arrests_local_state_police$Unitid, arrests_local_state_poli

#arrests_noncampus$new_ID <- paste(arrests_noncampus$Unitid, arrests_noncampus$Campus.ID, sep = "-")

# Add source column

#arrests_local_state_police$source <- "Local_State_Police"
#arrests_noncampus$source <- "Noncampus"

# combine data frames

#arrests_combined_1 <- rbind(arrests_local_state_police, arrests_noncampus)

#arrests_combined_1 <- bind_rows(arrests_local_state_police, arrests_noncampus)

# write combined data to CSV

#write.csv(arrests_combined_1, "arrests_combined_1.csv", row.names=FALSE)
```