Analysis of car accidents in Mexico City

2020-03-22

1 Intro

```
rm(list = ls())
#Load packages
library(dplyr) #v. 0.8.5
library(htmlwidgets) #v. 1.5.1
library(mgcv) #v. 1.8-28
library(plotmo) #v. 3.5.6
library(randomForest) #v. 4.6-14
library(ranger) #v. 0.12.1
library(RColorBrewer) #v. 1.1-2
# devtools::install_github("hrbrmstr/streamgraph")
library(streamgraph) #v. 0.9.0
library(xtable) #v. 1.8-4
```

Some colors to start with

```
COL <- c("black",
         rgb(100, 38, 33, maxColorValue = 100), #red
         rgb(0, 65, 55, maxColorValue = 100), #green
         rgb(28, 24, 61, maxColorValue = 100), #blue
         rgb(76, 32, 72, maxColorValue = 100), #purple
         rgb(21, 75, 87, maxColorValue = 100), #cyan
         rgb(0, 47, 59, maxColorValue = 100) #dark cyan
)
```

2 Data description

```
#load data
load("./dataderived/image_preprocessBoth.RData")

#load accidents data again to see the types of accidents
DA <- read.csv("./dataraw/accidents.csv", nrows = 155466)
names(DA)[c(2, 3, 7, 11)] <- c("Month", "Year", "Day", "Type")
#Truncate to the period of analysis (whole years here):
DA <- DA[(DA$Year >= 2001) & (DA$Year <= 2015),]
#DA <- DA[!(DA£Year == 2015 & DA£Month == 12),] #if need to remove Dec 2015
```

Number of accidents by year and type

The codes for the types of accidents:

1. Collision with other vehicle
2. Collision with pedestrian

3. Collision with animal
4. Collision with fixed object
5. Flip
6. Passenger fall off
7. Drive to ditch
8. Fire
9. Collision with train
10. Collision with motorcycle
11. Collision with bicycle
12. Other

Count accidents by year and type

```
DA$Count <- 1L
da <- aggregate(DA$Count, by = list(DA$Year, DA$Type), FUN = sum)
names(da) <- c("Year", "Type", "Count")
```

Number of different types of accidents

```
length(unique(da$Type))

## [1] 12
```

Figure 2: Number of car accidents per year in Mexico City

```
tmp <- brewer.pal(11, name = "Spectral")
#Add 1 more color to this palette of 11 colors and rearrange for a better look
tmp[5] <- "black"
tmp[7] <- tmp[1]
COL2 <- c(tmp[-1], COL[c(2, 5)])
pp = streamgraph(da, "Type", "Count", "Year",
                 offset = "zero", order = "asis",
                 interactive = TRUE) %>%
    sg_axis_x(1, "Year", "%Y") %>%
    sg_fill_manual(COL2) # sg_fill_brewer("Spectral")
#Save the widget then print in PDF and edit for the paper
saveWidget(pp, file = "AccidTypes.html")
```

Percentage by type

```
typeA <- table(DA$Type)
round(typeA *100 / sum(typeA), 1)

##
##    1    2    3    4    5    6    7    8    9   10   11   12
## 65.6  5.5  0.3  8.9  4.8  0.5  4.0  0.1  0.0  8.0  1.3  1.0
```

Number by year

```
tmp = table(DA$Year)
tmp

##
##  2001  2002  2003  2004  2005  2006  2007  2008  2009  2010  2011  2012  2013  2014  2015
##  7473  8024  8057  8858  8731  9396  7611 10214 10197  8711  8523  9431 10814  9502  9294
```

Average (percentage) increase by year

```
mean(diff(tmp)) #average increase

## [1] 130
```

```
MeanRelChange = (tmp[length(tmp)] / tmp[1]) ^ (1 / (length(tmp) - 1) )
MeanRelChange*100 - 100 #average percentage increase
```

```
## 2015
## 1.57
```

```
#check should be close to 0:
tmp[1] * MeanRelChange^((length(tmp) - 1)) - tmp[length(tmp)]
```

```
##      2001
## -1.09e-11
```

Collisions with motorcycles

```
tmp = da$Count[da$Type == 10] #select by type
tmp = tmp[c(1, length(tmp))] #select 1st and last years
tmp
```

```
## [1]  378 1320
```

```
tmp[2]/tmp[1] #increase times
```

```
## [1] 3.49
```

Collisions with pedestrians

```
tmp = da$Count[da$Type == 2] #select by type
tmp = tmp[c(1, length(tmp))] #select 1st and last years
tmp
```

```
## [1] 1455  377
```

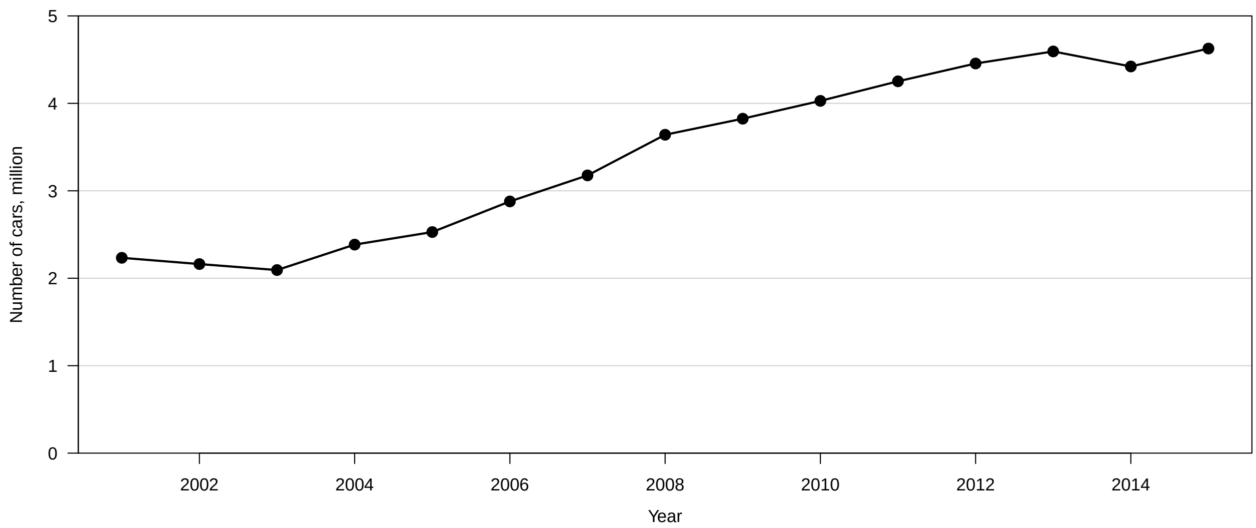
```
tmp[1]/tmp[2] #decrease times
```

```
## [1] 3.86
```

Number of cars registered

Figure 3: Number of cars registered in Mexico City, 2001–2015

```
CR <- CR[CR$Year >= 2001 & CR$Year <= 2015, ]
for(i in 1:2){
    if(i == 1) pdf("./figures/tsCarsReg.pdf", width = 8, height = 3.7)
    par(mar = c(3.5, 3.5, 0.5, 0.1), mgp = c(2.5, 1, 0))
    plot(CR$Year, CR$CarsReg/1000000,
         xlab = "Year", ylab = "Number of cars, million",
         ylim = c(0, 5), yaxs="i",
         type = "o", pch = 16, cex = 1.5,
         panel.first = grid(nx = 0, ny = 5, lty = 1),
         col = COL[1], lty = 1, las = 1, lwd = 2)
    if(i == 1) dev.off()
}
```



Average (percentage) increase by year

```
tmp = CR$CarsReg/1000000 #cars registered, million
tmp[c(1, length(tmp))] #select 1st and last years

## [1] 2.23 4.63

mean(diff(tmp)) #average increase

## [1] 0.171

MeanRelChange = (tmp[length(tmp)] / tmp[1]) ^ (1 / (length(tmp) - 1) )
MeanRelChange*100 - 100 #average percentage increase

## [1] 5.34

#check should be close to 0:
tmp[1] * MeanRelChange^((length(tmp) - 1)) - tmp[length(tmp)]

## [1] 8.88e-16
```

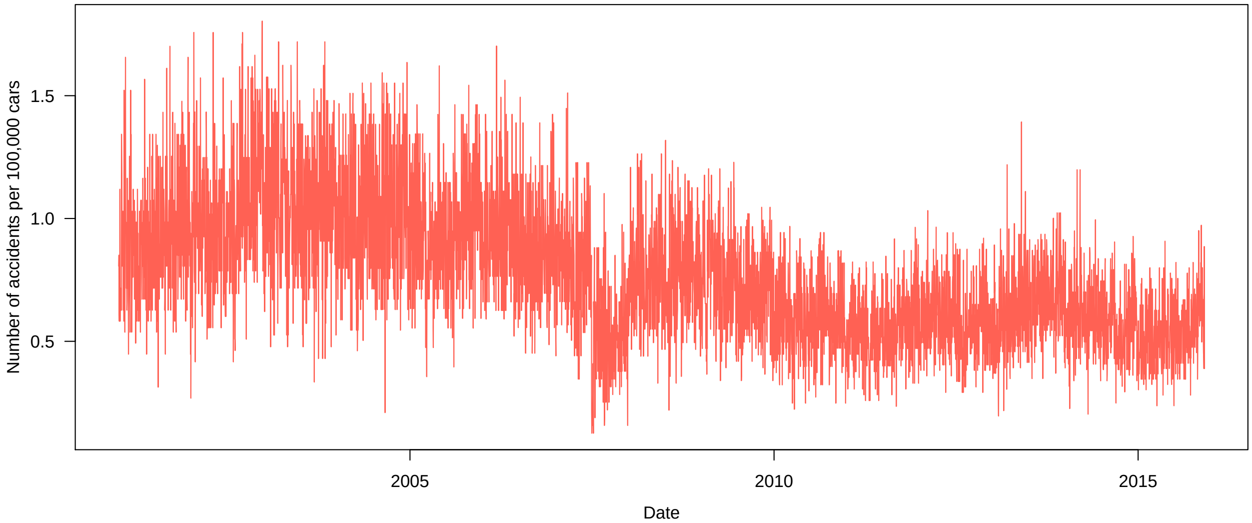
Percent missing values

```
tmp = apply(is.na(DataHour), 2, mean)
max(tmp * 100)

## [1] 0.0635
```

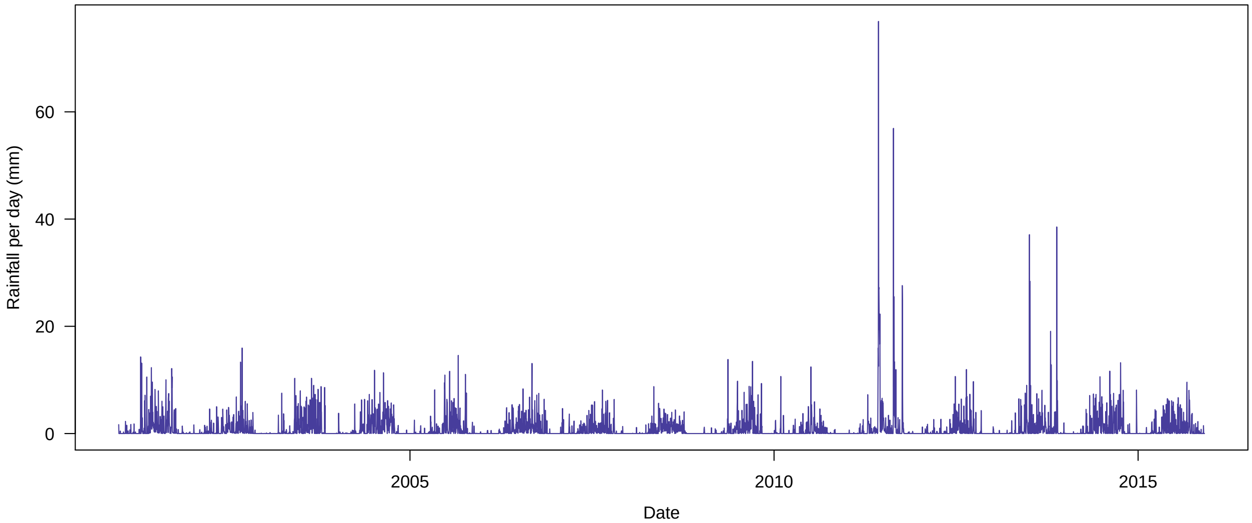
Weather

Figure 4: Time series plots of daily accident rate, total rainfall, and average air temperature

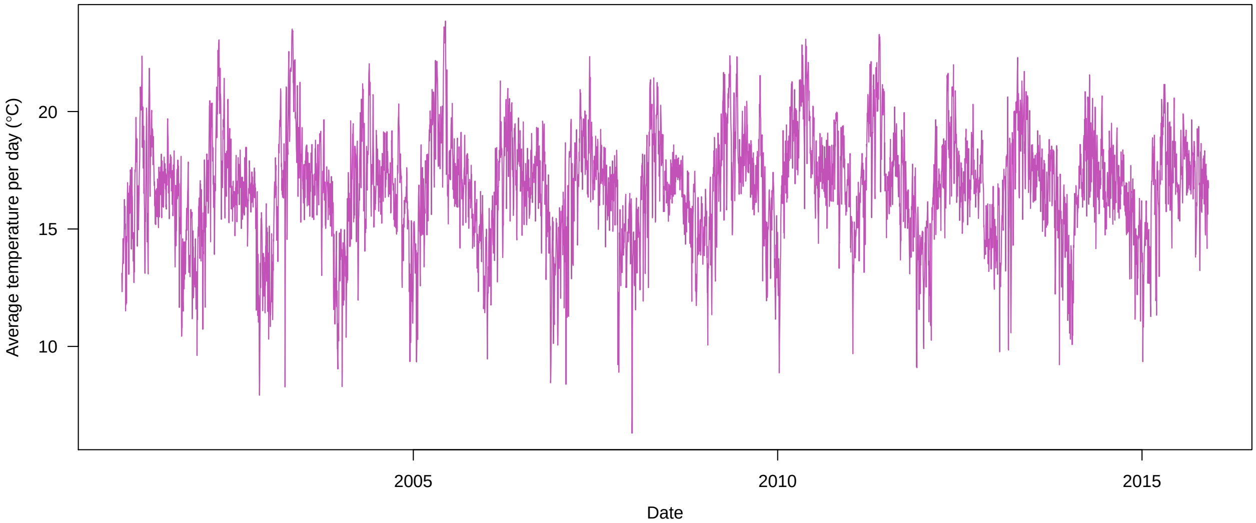
```
D <- DataDay
for(i in 1:2){
    if(i == 1) pdf("./figures/tsNaccid.pdf", width = 8, height = 3.7)
    par(mar = c(3.5, 3.5, 0.1, 0.1), mgp = c(2.5, 1, 0))
    plot(D$Date, D$NAccidPer100000,
         xlab = "Date",
         ylab = "Number of accidents per 100,000 cars",
         col = COL[2], type = "l", lty = 1, las = 1)
    if(i == 1) dev.off()
}
```




```
for(i in 1:2){
    if(i == 1) pdf("./figures/tsRain.pdf", width = 8, height = 3.7)
    par(mar = c(3.5, 3.5, 0.1, 0.1), mgp = c(2.5, 1, 0))
    plot(D$Date, D$Rain,
         xlab = "Date",
         ylab = "Rainfall per day (mm)",
         col = COL[4], type = "l", lty = 1, las = 1)
    if(i == 1) dev.off()
}
```



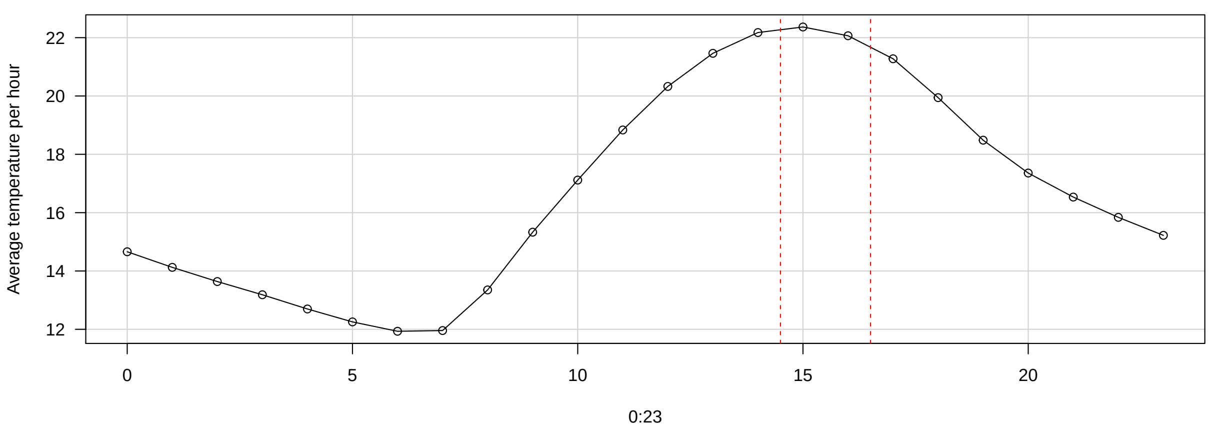

```
for(i in 1:2){
    if(i == 1) pdf("./figures/tsTemp.pdf", width = 8, height = 3.7)
    par(mar = c(3.5, 3.5, 0.1, 0.1), mgp = c(2.5, 1, 0))
    plot(D$Date, D$Temperature,
         xlab = "Date",
         ylab = expression(paste("Average temperature per day (",degree,"C)")),
         col = COL[5], type = "l", lty = 1, las = 1)
    if(i == 1) dev.off()
}
```

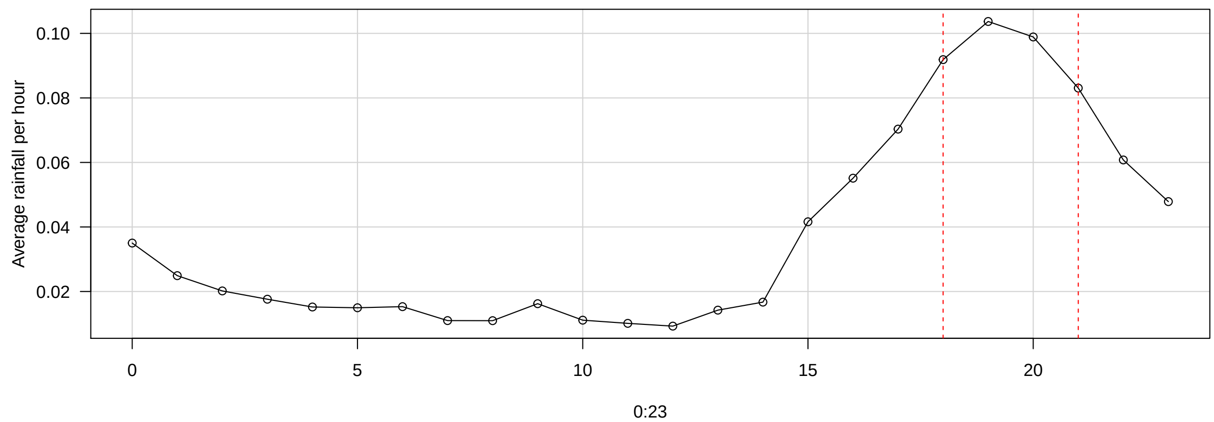


Weather – for conclusions

Weather by hour

```
tmp <- tapply(DataHour$Temperature, DataHour$Hour, mean, na.rm = TRUE)
plot(tmp, x = 0:23, type = "o", panel.first = grid(lty = 1),
     ylab = "Average temperature per hour",
     las = 1)
#add lines for 'busiest' hours
abline(v = c(14.5, 16.5), col = 2, lty = 2)
tmp <- tapply(DataHour$Rain, DataHour$Hour, mean, na.rm = TRUE)
plot(tmp, x = 0:23, type = "o", panel.first = grid(lty = 1),
     ylab = "Average rainfall per hour",
     las = 1)
#most rainy hours
abline(v = c(18, 21), col = 2, lty = 2)
```





Rainy months (apply it to full years only, i.e., before 2015)

```
tapply(DataDay$Rain[DataDay$Year < 2015], DataDay$Month[DataDay$Year < 2015], mean) *
    length(unique(DataDay$Year[DataDay$Year < 2015]))

##      1      2      3      4      5      6      7      8      9     10     11     12
##  1.086  1.915  2.318  6.377 10.359 30.877 25.009 27.180 26.418 13.738  4.500  0.587
```

3 Methods

4 Results

Table 1: Quartile summaries (daily data)

The quartiles table with std dev. for the mean (divide by $\sqrt{n}$)

```
D <- DataDay
labs = seq(0.25, 1, by = 0.25)
#temperature quartiles
tq = quantile(D$Temperature, probs = c(0, 0.25, 0.50, 0.75, 1))
tq

##    0%   25%   50%   75%  100%
##  6.31 15.41 16.87 18.18 23.85

D$tempInt = cut(D$Temperature, breaks = tq, labels = paste("temp_", labs, sep = ""),
                include.lowest = TRUE, right = FALSE, ordered_result = TRUE)
#rain quartiles (FOR RAINY DAYS!)
rq = quantile(D$Rain[D$Rain > 0], probs = c(0, 0.25, 0.50, 0.75, 1))
rq[1] = 0
rq

##     0%    25%    50%    75%   100%
##  0.000  0.178  0.810  2.200 76.870

D$rainInt = cut(D$Rain, breaks = rq, labels = paste("rain_", labs, sep = ""),
                include.lowest = TRUE, right = FALSE, ordered_result = TRUE)
#Summary per intersection of the quartiles:
magg1 <- tapply(D$NAccidPer100000, list(D$tempInt, D$rainInt), mean)
magg1 <- format(round(magg1, 2), digits = 2)
#Sample size per intersection of the quartiles:
ss <- table(D$tempInt, D$rainInt)
sdagg1 <- tapply(D$NAccidPer100000, list(D$tempInt, D$rainInt), sd)
sdagg1 <- sdagg1 / sqrt(ss)
```

```
sdagg1 <- format(round(sdagg1, 2), digits = 2)
M <- matrix(paste(magg1, " (", sdagg1, ")", sep = ""), nrow = 4)
dimnames(M) <- dimnames(magg1)
#Copy this from R console into latex:
print(xtable(M,
             caption = "Average accident rate, st.dev. in the parentheses",
             label = "tab:TempRain", size = "small"))

## % latex table generated in R 3.6.1 by xtable 1.8-4 package
## % Sun Mar 22 03:52:24 2020
## \begin{table}[ht]
## \centering
## \begin{tabular}{rllll}
##   \hline
##  & rain\_0.25 & rain\_0.5 & rain\_0.75 & rain\_1 \\
##   \hline
## temp\_0.25 & 0.79 (0.01) & 0.80 (0.03) & 0.85 (0.04) & 0.80 (0.03) \\
##   temp\_0.5 & 0.78 (0.01) & 0.79 (0.02) & 0.80 (0.02) & 0.79 (0.02) \\
##   temp\_0.75 & 0.75 (0.01) & 0.74 (0.02) & 0.74 (0.02) & 0.79 (0.02) \\
##   temp\_1 & 0.74 (0.01) & 0.69 (0.02) & 0.72 (0.02) & 0.73 (0.03) \\
##    \hline
## \end{tabular}
## \caption{Average accident rate, st.dev. in the parentheses}
## \label{tab:TempRain}
## \end{table}
```

4.1 GAM

Daily GAM

```
D <- DataDay
D$Month <- factor(D$Month)
# Create train+test data
DtrainDay <- D[D$Year <= 2012,]
DtestDay <- D[D$Year > 2012,]
# summary(DtrainDay)
# summary(DtestDay)
```

Size of the training and testing data

```
nrow(DtrainDay)

## [1] 4383

nrow(DtestDay)

## [1] 1064
```

```
K <- 5
set.seed(140)
gamfit <- gamDay <- mgcv::gam(NAccidPer100000 ~ s(Year, k = K)
                              + Month
                              + Weekday
                              + HNSSaturday + Holiday
                              + te(Rain, Temperature, k = K)
                              , select = TRUE
                              , bs = "cr"
                              , method = "REML"
                              , data = DtrainDay)
```

```
anova(gamfit)

##
## Family: gaussian
## Link function: identity
##
## Formula:
```

```
## NAccidPer100000 ~ s(Year, k = K) + Month + Weekday + HNSSaturday +
##     Holiday + te(Rain, Temperature, k = K)
##
## Parametric Terms:
##             df     F p-value
## Month       11  8.08 3.9e-14
## Weekday      6 93.95 < 2e-16
## HNSSaturday  1  6.27 0.01229
## Holiday      1 12.35 0.00045
##
## Approximate significance of smooth terms:
##                        edf Ref.df      F p-value
## s(Year)               3.89   4.00 667.07 < 2e-16
## te(Rain,Temperature)  4.25  24.00   1.06 3.6e-06
```

```r
summary(gamfit)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## NAccidPer100000 ~ s(Year, k = K) + Month + Weekday + HNSSaturday +
##     Holiday + te(Rain, Temperature, k = K)
##
## Parametric coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.770953   0.014288   53.96  < 2e-16 ***
## Month2        0.046945   0.015805    2.97  0.00299 **
## Month3        0.028194   0.016555    1.70  0.08863 .
## Month4       -0.000206   0.017905   -0.01  0.99083
## Month5        0.039739   0.018114    2.19  0.02830 *
## Month6        0.027002   0.017511    1.54  0.12314
## Month7       -0.066507   0.016828   -3.95  7.9e-05 ***
## Month8       -0.021090   0.017015   -1.24  0.21524
## Month9       -0.000971   0.016895   -0.06  0.95417
## Month10      -0.000934   0.015994   -0.06  0.95344
## Month11       0.027360   0.015187    1.80  0.07168 .
## Month12       0.032281   0.014902    2.17  0.03035 *
## Weekday2     -0.062866   0.011493   -5.47  4.8e-08 ***
## Weekday3     -0.031328   0.011502   -2.72  0.00648 **
## Weekday4     -0.021796   0.011486   -1.90  0.05782 .
## Weekday5      0.063075   0.011480    5.49  4.1e-08 ***
## Weekday6      0.180532   0.013327   13.55  < 2e-16 ***
## Weekday7      0.112156   0.011473    9.78  < 2e-16 ***
## HNSSaturday1 -0.044911   0.017931   -2.50  0.01229 *
## Holiday1     -0.058966   0.016781   -3.51  0.00045 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                       edf Ref.df      F p-value
## s(Year)              3.89      4 667.07 < 2e-16 ***
## te(Rain,Temperature) 4.25     24   1.06 3.6e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.471   Deviance explained = 47.4%
## -REML = -695.15  Scale est. = 0.041158  n = 4383
```

```r
concurvity(gamfit)
```

```
##          para s(Year) te(Rain,Temperature)
## worst    0.959   0.164                0.686
## observed 0.959   0.125                0.313
## estimate 0.959   0.120                0.060
```

```r
# gam.check(gamfit) #commented out because plots are slow to render in PDF
```

```r
# plot(gamfit)
# acf(residuals(gamfit, type = "pearson"), las = 1) #significant but low
```

Hourly GAM

```r
D <- DataHour
D$Temperature.l1 <- dplyr::lag(D$Temperature, 1)
D$Rain.l1 <- dplyr::lag(D$Rain, 1)
D$Month <- factor(D$Month)
# Create train+test data
DtrainHour <- D[D$Year <= 2012,]
DtestHour <- D[D$Year > 2012,]
# summary(DtrainHour)
# summary(DtestHour)
```

Size of the training and testing data

```r
nrow(DtrainHour)

## [1] 105192

nrow(DtestHour)

## [1] 25536
```

```r
K <- 5
set.seed(140000)
gamfit <- gamHour <- mgcv::gam(NAccidPer100000 ~ s(Year, k = K)
                               + Month + Weekday
                               + HNSSaturday + Holiday
                               + s(Hour, k = K)
                               + te(Rain, Temperature, k = K)
                               + te(Rain.l1, Temperature.l1, k = K)
                               , select = TRUE
                               , bs = "cr"
                               , method = "REML"
                               , data = DtrainHour)
```

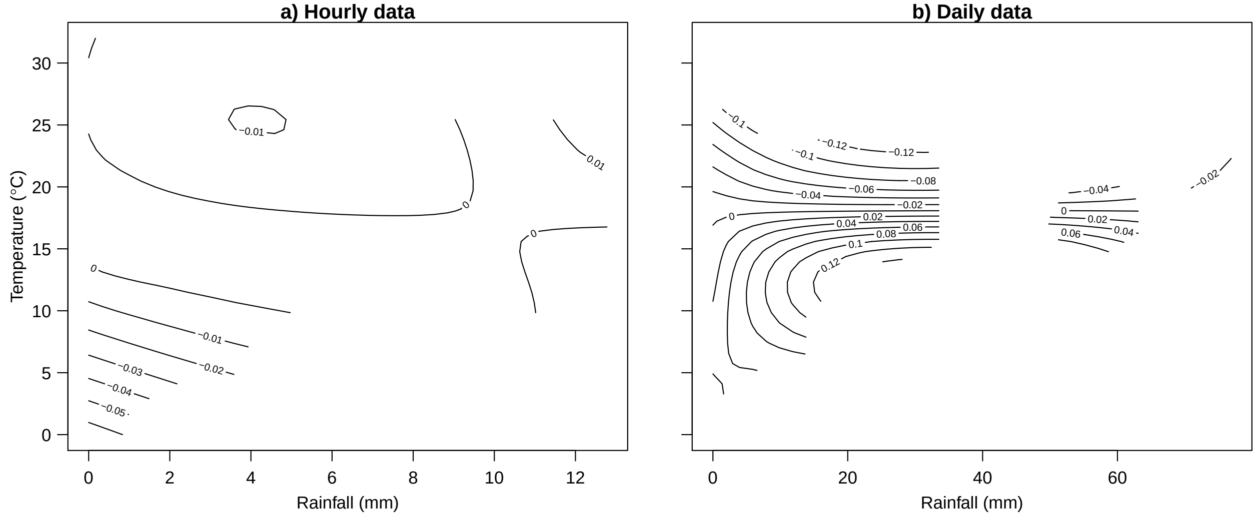
```r
anova(gamfit)

##
## Family: gaussian
## Link function: identity
##
## Formula:
## NAccidPer100000 ~ s(Year, k = K) + Month + Weekday + HNSSaturday +
##     Holiday + s(Hour, k = K) + te(Rain, Temperature, k = K) +
##     te(Rain.l1, Temperature.l1, k = K)
##
## Parametric Terms:
##             df      F p-value
## Month       11  12.32 < 2e-16
## Weekday      6 115.72 < 2e-16
## HNSSaturday  1   8.07  0.0045
## Holiday      1  15.21 9.6e-05
##
## Approximate significance of smooth terms:
##                               edf Ref.df     F p-value
## s(Year)                      3.91   4.00 841.9  <2e-16
## s(Hour)                      3.99   4.00 851.8  <2e-16
## te(Rain,Temperature)         9.12  24.00  18.9  <2e-16
## te(Rain.l1,Temperature.l1)  6.09  24.00  13.9  <2e-16

summary(gamfit)

##
## Family: gaussian
## Link function: identity
```

```
##
## Formula:
## NAccidPer100000 ~ s(Year, k = K) + Month + Weekday + HNSSaturday +
##     Holiday + s(Hour, k = K) + te(Rain, Temperature, k = K) +
##     te(Rain.l1, Temperature.l1, k = K)
##
## Parametric coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.28e-02   5.12e-04   64.06  < 2e-16 ***
## Month2        1.58e-03   5.81e-04    2.71   0.0067 **
## Month3        6.48e-04   5.86e-04    1.11   0.2689
## Month4       -8.68e-04   6.17e-04   -1.41   0.1595
## Month5        5.97e-04   6.20e-04    0.96   0.3356
## Month6        7.33e-05   6.13e-04    0.12   0.9048
## Month7       -3.70e-03   5.95e-04   -6.21  5.2e-10 ***
## Month8       -1.86e-03   5.99e-04   -3.10   0.0019 **
## Month9       -9.72e-04   5.99e-04   -1.62   0.1043
## Month10      -7.75e-04   5.80e-04   -1.34   0.1812
## Month11       7.87e-04   5.67e-04    1.39   0.1652
## Month12       1.17e-03   5.59e-04    2.09   0.0364 *
## Weekday2     -2.58e-03   4.31e-04   -5.97  2.3e-09 ***
## Weekday3     -1.30e-03   4.32e-04   -3.01   0.0026 **
## Weekday4     -9.19e-04   4.31e-04   -2.13   0.0330 *
## Weekday5      2.62e-03   4.31e-04    6.08  1.2e-09 ***
## Weekday6      7.53e-03   5.00e-04   15.06  < 2e-16 ***
## Weekday7      4.70e-03   4.31e-04   10.91  < 2e-16 ***
## HNSSaturday1 -1.91e-03   6.73e-04   -2.84   0.0045 **
## Holiday1     -2.45e-03   6.29e-04   -3.90  9.6e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                              edf Ref.df     F p-value
## s(Year)                     3.91      4 841.9  <2e-16 ***
## s(Hour)                     3.99      4 851.8  <2e-16 ***
## te(Rain,Temperature)        9.12     24  18.9  <2e-16 ***
## te(Rain.l1,Temperature.l1) 6.09     24  13.9  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.124   Deviance explained = 12.4%
## -REML = -1.9638e+05  Scale est. = 0.0013898  n = 105095

concurvity(gamfit)

##          para s(Year) s(Hour) te(Rain,Temperature) te(Rain.l1,Temperature.l1)
## worst    0.95   0.144   0.709                0.965                      0.964
## observed 0.95   0.109   0.660                0.890                      0.908
## estimate 0.95   0.103   0.424                0.731                      0.739

# gam.check(gamfit) #commented out because plots are slow to render in PDF
# plot(gamfit)
# acf(residuals(gamfit, type = "pearson"), las = 1) #significant but low
```

Figure 5: Contour plots of the tensor smooth terms

```
for(i in 1:2){
    if(i == 1) pdf("./figures/teTempRain.pdf", width = 8, height = 3.7)
    par(mar = c(3.3, 3, 1.1, 0.1), mgp = c(2.0, 0.8, 0))
    par(mfrow = c(1, 2))
    plot(gamHour, select = 3, se = FALSE, rug = FALSE, las = 1, ylim = c(0, 32), main = "a) Hourly data",
         xlab = "Rainfall (mm)", ylab = "")
    mtext(expression(paste("Temperature (",degree,"C)")), side = 2, line = 1.8)
    plot(gamDay,  select = 2, se = FALSE, rug = FALSE, las = 1, ylim = c(0, 32), main = "b) Daily data",
         xlab = "Rainfall (mm)",
         yaxt = "n", ylab = "")
    axis(2, labels = NA)
    if(i == 1) dev.off()
```

```
}
```



4.2 Random forest

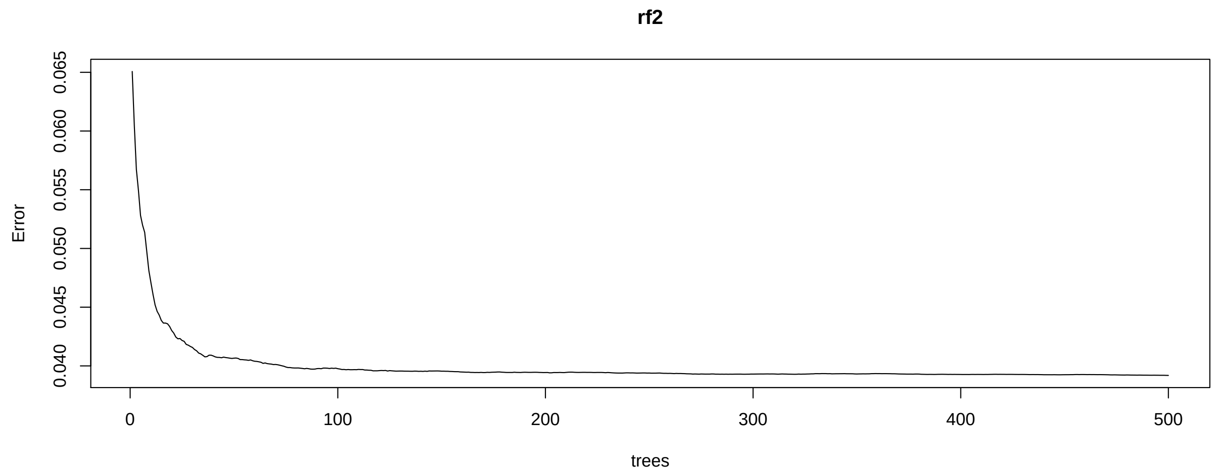
Daily RF

```
RESPONSE <- "NAccidPer100000"
#predictors
v <- c("HNSSaturday", "Holiday", "Month", "Year", "Rain",  "Temperature", "Weekday")
```

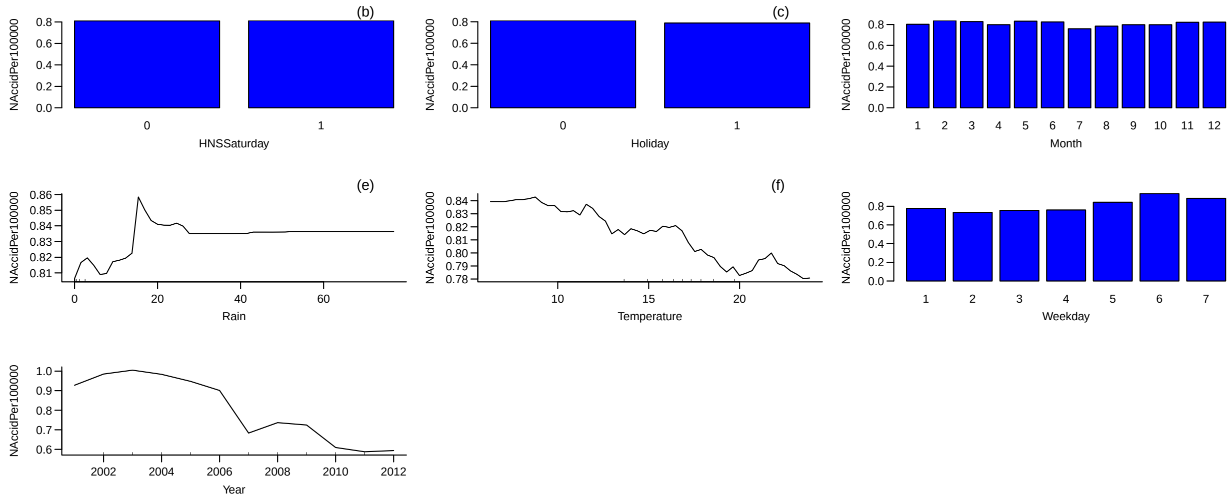
```
DATAnoNA <- na.omit(DtrainDay[,c(RESPONSE, v)])
set.seed(10000)
ran <- RFDay <- ranger(dependent.variable.name = RESPONSE, data = DATAnoNA,
                       importance = 'impurity_corrected',
                       min.node.size = 5, respect.unordered.factors = 'partition',
                       num.trees = 500)
#for predictions
ran2 <- ran2Day <- ranger(dependent.variable.name = RESPONSE, data = DATAnoNA,
                          # importance = 'impurity_corrected',
                          min.node.size = 5, respect.unordered.factors = 'partition',
                          num.trees = 500)
print(ran)

## Ranger result
##
## Call:
##  ranger(dependent.variable.name = RESPONSE, data = DATAnoNA, importance = "impurity_corrected",      min.node.size = 5, respe
##
## Type:                             Regression
## Number of trees:                  500
## Sample size:                      4383
## Number of independent variables:  7
## Mtry:                             2
## Target node size:                 5
## Variable importance mode:         impurity_corrected
## Splitrule:                        variance
## OOB prediction error (MSE):       0.0417
## R squared (OOB):                  0.464

# ranimp <- importance_pvalues(ran, method = "altmann",
#                              num.permutations = 500,
#                              formula = as.formula(paste(RESPONSE, ".", sep = " ~ ")),
#                              data = DATAnoNA)
# ranimp <- ranimp[order(ranimp[,1]),]
```

```r
# ranimp
```

```r
set.seed(300000)
rf2 <- rf2Day <- randomForest(y = DATAnoNA[,RESPONSE],
                              x = DATAnoNA[, v],
                              nodesize = ran$min.node.size,
                              mtry = ran$mtry,
                              ntree = ran$num.trees)
print(rf2)

##
## Call:
##  randomForest(x = DATAnoNA[, v], y = DATAnoNA[, RESPONSE], ntree = ran$num.trees,      mtry = ran$mtry, nodesize = ran$min.nc
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 0.0392
##                     % Var explained: 49.6

plot(rf2)
```




```r
RF <- rf2
preds <- sort(rownames(rf2$importance)) # sort(v)
par(mfrow = c(ceiling(length(preds)/3), 3))
par(bty = "L", mar = c(5, 4, 1, 1) + 0.1, mgp = c(2, 0.7, 0))
for(i in 1:length(preds)) {
    partialPlot(RF, pred.data = DtrainDay, x.var = preds[i],
                las = 1, xlab = preds[i], ylab = "", main = "", xpd = F)
    mtext("NAccidPer100000", side = 2, line = 3, cex = 0.7)
    mtext(paste("(", letters[i], ")", sep = ""), side = 3, line = 0.1, cex = 0.8, adj = -0.37)
}
```



Hourly RF

```r
RESPONSE <- "NAccidPer100000"
#predictors
v <- c("HNSSaturday", "Holiday", "Hour", "Month", "Year", "Rain",  "Temperature",
       "Rain.l1",  "Temperature.l1", "Weekday")
```

```r
DATAnoNA <- na.omit(DtrainHour[,c(RESPONSE, v)])
set.seed(10000)
ran <- RFHour <- ranger(dependent.variable.name = RESPONSE, data = DATAnoNA,
                        importance = 'impurity_corrected',
                        min.node.size = 5, respect.unordered.factors = 'partition',
                        num.trees = 100)

## Growing trees.. Progress: 12%. Estimated remaining time: 4 minutes, 9 seconds.
## Growing trees.. Progress: 26%. Estimated remaining time: 3 minutes, 13 seconds.
## Growing trees.. Progress: 41%. Estimated remaining time: 2 minutes, 23 seconds.
## Growing trees.. Progress: 56%. Estimated remaining time: 1 minute, 43 seconds.
## Growing trees.. Progress: 71%. Estimated remaining time: 1 minute, 6 seconds.
## Growing trees.. Progress: 84%. Estimated remaining time: 38 seconds.
## Growing trees.. Progress: 100%. Estimated remaining time: 0 seconds.

#for predictions
ran2 <- ran2Hour <- ranger(dependent.variable.name = RESPONSE, data = DATAnoNA,
                           # importance = 'impurity_corrected',
                           min.node.size = 5, respect.unordered.factors = 'partition',
                           num.trees = 100)

## Growing trees.. Progress: 18%. Estimated remaining time: 2 minutes, 21 seconds.
## Growing trees.. Progress: 40%. Estimated remaining time: 1 minute, 34 seconds.
## Growing trees.. Progress: 61%. Estimated remaining time: 1 minute, 0 seconds.
## Growing trees.. Progress: 83%. Estimated remaining time: 26 seconds.

print(ran)

## Ranger result
##
## Call:
##  ranger(dependent.variable.name = RESPONSE, data = DATAnoNA, importance = "impurity_corrected",      min.node.size = 5, respe
##
## Type:                             Regression
## Number of trees:                  100
## Sample size:                      105095
## Number of independent variables:  10
## Mtry:                             3
## Target node size:                 5
## Variable importance mode:         impurity_corrected
```

```
## Splitrule:                          variance
## OOB prediction error (MSE):         0.00137
## R squared (OOB):                    0.139
```

```
# ranimp <- importance_pvalues(ran, method = "altmann",
#                               num.permutations = 500,
#                               formula = as.formula(paste(RESPONSE, ".", sep = " ~ ")),
#                               data = DATAnoNA)
# ranimp <- ranimp[order(ranimp[,1]),]
# ranimp
```

```
set.seed(300000)
rf2 <- rf2Hour <- randomForest(y = DATAnoNA[,RESPONSE],
                               x = DATAnoNA[, v],
                               nodesize = ran$min.node.size,
                               mtry = ran$mtry,
                               ntree = ran$num.trees)
print(rf2)
```

```
##
## Call:
##  randomForest(x = DATAnoNA[, v], y = DATAnoNA[, RESPONSE], ntree = ran$num.trees,      mtry = ran$mtry, nodesize = ran$min.nc
##                Type of random forest: regression
##                      Number of trees: 100
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 0.00137
##                     % Var explained: 13.6
```

```
plot(rf2)
```

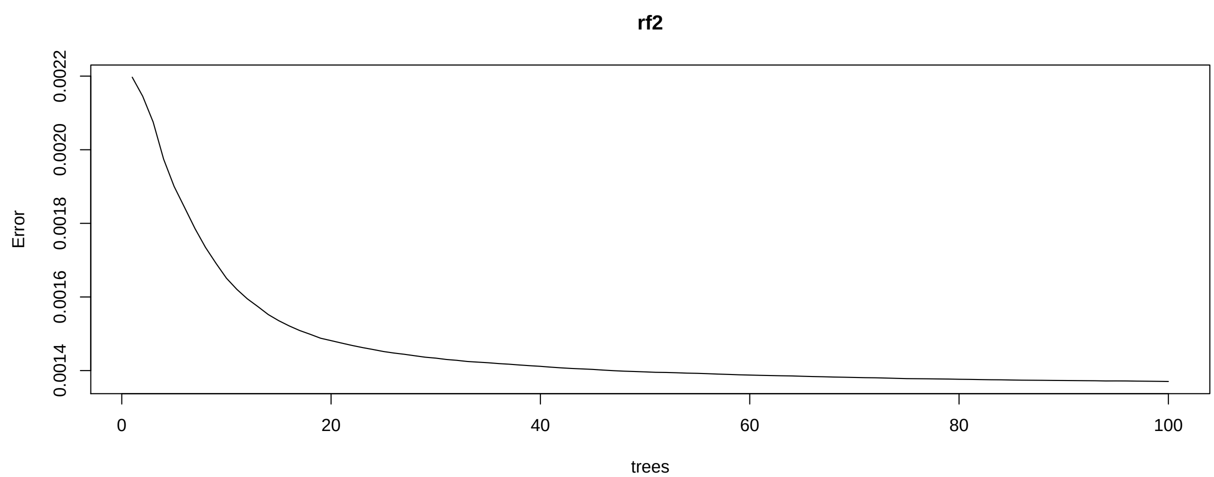


Figure 6: Relative importance of the variables in random forests

```
for(i in 1:2){
    if(i == 1) pdf("./figures/RFimp.pdf", width = 8, height = 3.7)
    par(mar = c(3.3, 5.5, 1.1, 1), mgp = c(2.0, -0.3, 0))
    par(mfrow = c(1, 2))
    #
    tmp <- sort(RFHour$variable.importance)
    names(tmp)[grep("HNS", names(tmp))] <- "HNCS"
    names(tmp)[grep("ture.", names(tmp))] <- "Temperature(t-1)"
    names(tmp)[grep("ain.", names(tmp))] <- "Rainfall(t-1)"
    names(tmp)[names(tmp) == "Rain"] <- "Rainfall"
    barplot(tmp,
            beside = TRUE, las = 1, xlim = c(-2, 10),
            main = "a) Hourly data",
            xlab = "Importance",
```

```
            col = COL[4], border = NA, cex.names = 0.8, xaxt = "n",
            horiz = TRUE)
    par(mgp = c(2.0, 0.8, 0))
    axis(1)
    #
    tmp <- sort(RFDay$variable.importance)
    names(tmp)[grep("HNS", names(tmp))] <- "HNCS"
    names(tmp)[names(tmp) == "Rain"] <- "Rainfall"
    barplot(tmp,
            beside = TRUE, las = 1, xlim = c(0, 100),
            main = "b) Daily data",
            xlab = "Importance",
            col = COL[4], border = NA, cex.names = 0.8,
            horiz = TRUE)
    if(i == 1) dev.off()
}
```

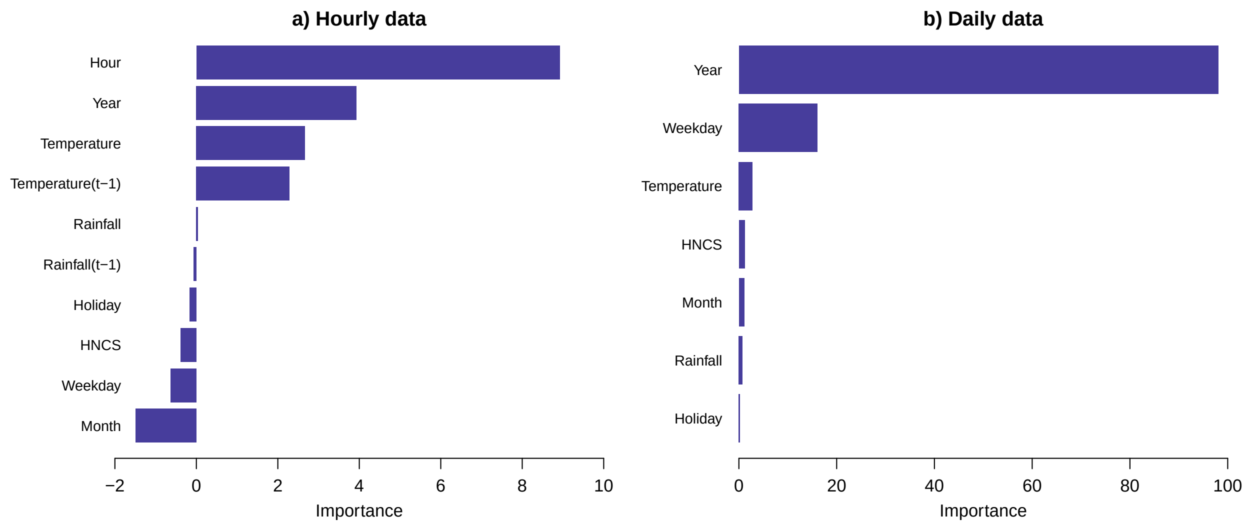


Figure 7: Partial dependence interaction plots from the random forests

```
pdf("./figures/RFinter.pdf", width = 5, height = 4.7)
par(mfrow = c(1, 2))
par(mar = c(3.3, 3, 1.1, 0.1), mgp = c(2.0, 0.8, 0))
v <- c("HNSSaturday", "Holiday", "Hour", "Month", "Year", "Rain",  "Temperature",
       "Rain.l1",  "Temperature.l1", "Weekday")
DATAnoNA <- na.omit(DtrainHour[,c(RESPONSE, v)])
RF <- rf2Hour
plotmo(RF, pmethod = "partdep", ylim = c(0, 32),
       all1 = FALSE,
       degree1 = FALSE,
       degree2 = c("Rain", "Temperature"),
       type2 = "contour",
       caption = "", main = "a) Hourly data",
       all2 = TRUE)

## calculating partdep for Rain:Temperature 01234567890

v <- c("HNSSaturday", "Holiday", "Month", "Year", "Rain",  "Temperature", "Weekday")
DATAnoNA <- na.omit(DtrainDay[,c(RESPONSE, v)])
RF <- rf2Day
plotmo(RF, pmethod = "partdep", ylim = c(0, 32),
       all1 = FALSE,
       degree1 = FALSE,
       degree2 = c("Rain", "Temperature"),
       type2 = "contour",
       caption = "", main = "b) Daily data",
       all2 = TRUE)
```

```
## calculating partdep for Rain:Temperature 01234567890

dev.off()

## pdf
##   2
```

4.3 Performance evaluation

Number of accidents per year in the training set

```
sum(DtrainDay$NAccidPer100000) / length(unique(DtrainDay$Year))

## [1] 296
```

Test daily

```
Dtest <- DtestDay
gamfit <- gamDay
ran2 <- ran2Day
```

```
pred_gam <- predict(gamfit, Dtest)
pred_ran <- predict(ran2, Dtest)$predictions
#PMAE
mean(abs(Dtest[,RESPONSE] - pred_gam))

## [1] 0.189

mean(abs(Dtest[,RESPONSE] - pred_ran))

## [1] 0.113

#PRMSE
sqrt(mean((Dtest[,RESPONSE] - pred_gam)^2))

## [1] 0.223

sqrt(mean((Dtest[,RESPONSE] - pred_ran)^2))

## [1] 0.143

#PMAPE
100*mean(abs((Dtest[,RESPONSE] - pred_gam)/Dtest[,RESPONSE]))

## [1] 30.4

100*mean(abs((Dtest[,RESPONSE] - pred_ran)/Dtest[,RESPONSE]))

## [1] 21.2
```

```
tapply(Dtest[,RESPONSE], Dtest$Year, sum)

## 2013 2014 2015
##  235  215  179

tapply(pred_gam, Dtest$Year, sum)

## 2013 2014 2015
##  174  152  116

tapply(pred_ran, Dtest$Year, sum)

## 2013 2014 2015
##  216  217  198
```

Test hourly

```
Dtest <- DtestHour
gamfit <- gamHour
```

```r
ran2 <- ran2Hour
```

```r
pred_gam <- predict(gamfit, Dtest)
pred_ran <- predict(ran2, Dtest)$predictions
#PMAE
mean(abs(Dtest[,RESPONSE] - pred_gam))
```

```
## [1] 0.0188
```

```r
mean(abs(Dtest[,RESPONSE] - pred_ran))
```

```
## [1] 0.0191
```

```r
#PRMSE
sqrt(mean((Dtest[,RESPONSE] - pred_gam)^2))
```

```
## [1] 0.0257
```

```r
sqrt(mean((Dtest[,RESPONSE] - pred_ran)^2))
```

```
## [1] 0.0246
```

```r
#PMAPE
100*mean(abs((Dtest[,RESPONSE] - pred_gam)/Dtest[,RESPONSE]))
```

```
## [1] Inf
```

```r
100*mean(abs((Dtest[,RESPONSE] - pred_ran)/Dtest[,RESPONSE]))
```

```
## [1] Inf
```

```r
tapply(Dtest[,RESPONSE], Dtest$Year, sum)
```

```
## 2013 2014 2015
##  235  215  179
```

```r
tapply(pred_gam, Dtest$Year, sum)
```

```
## 2013 2014 2015
##  176  155  121
```

```r
tapply(pred_ran, Dtest$Year, sum)
```

```
## 2013 2014 2015
##  218  216  197
```

Average annual accident rate in the training period

```r
tmp = tapply(DtrainDay[,RESPONSE], DtrainDay$Year, sum)
tmp
```

```
## 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012
##  335  371  385  371  345  326  240  281  267  216  200  212
```

```r
mean(tmp)
```

```
## [1] 296
```

Save all objects from the R environment:

```r
save.image(file = "./dataderived/image_MexAnalysis.RData")
```