Project Challenge

Used libraries

```
library(ggfortify)
```

```
## Loading required package: ggplot2
```

```
library(randomForest)
```

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

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(caret)
```

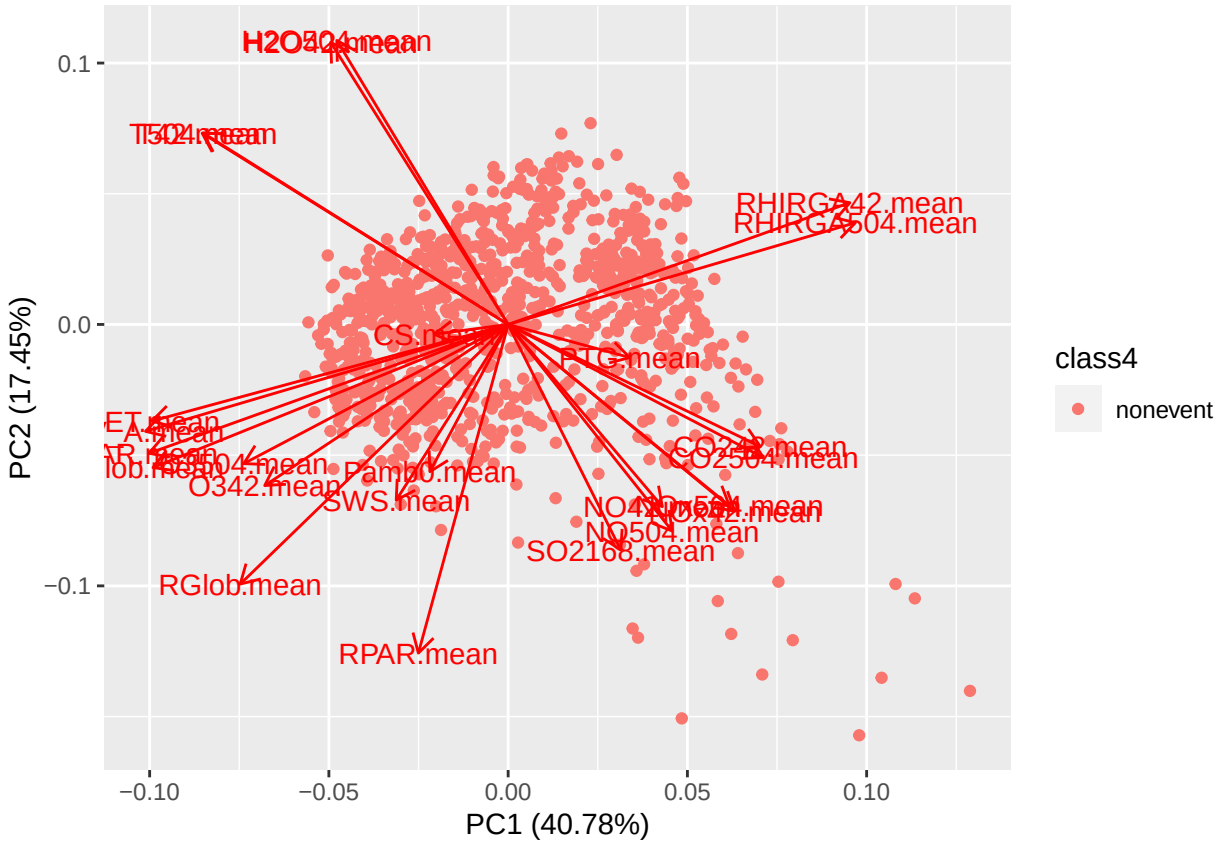
```
## Loading required package: lattice
```

Read data

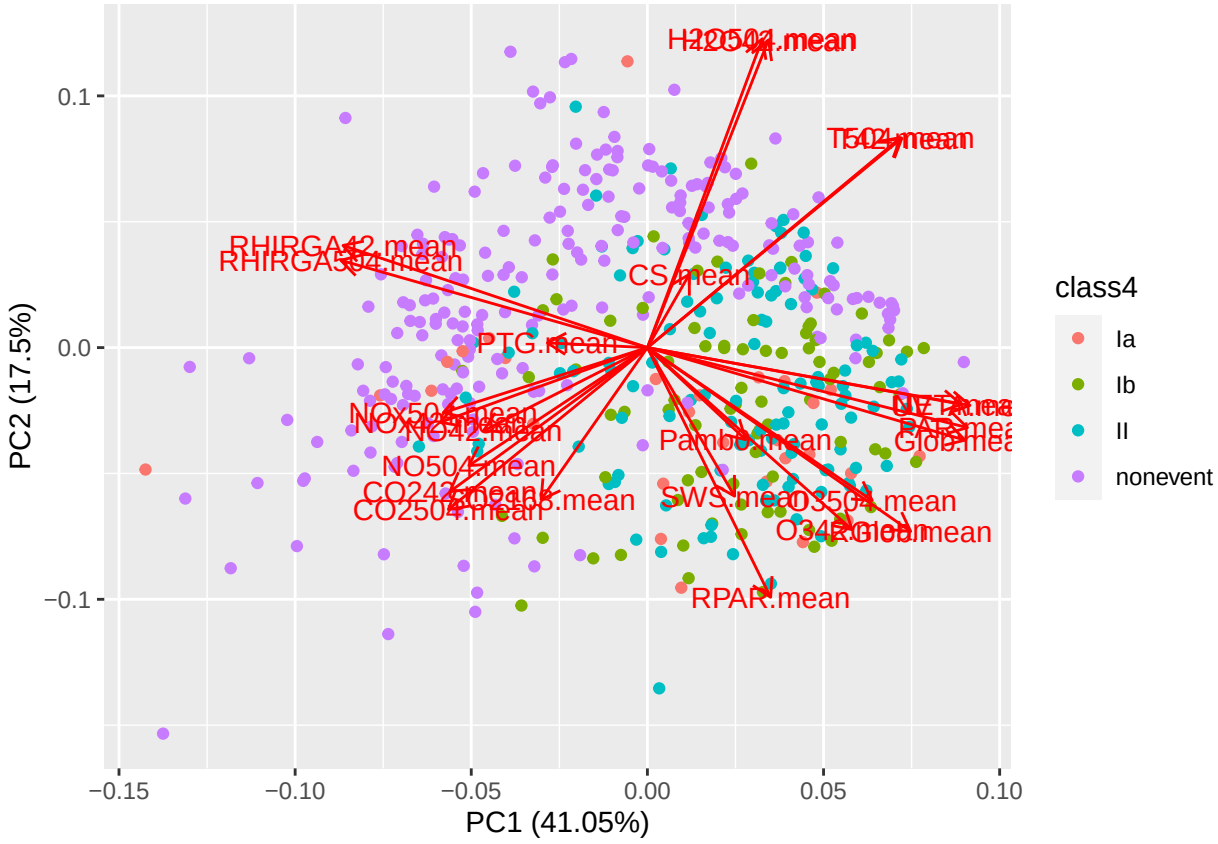
```
#setwd("path/to/folder")
set.seed(42)

npf <- read.csv("npf_train.csv")
npf_test <- read.csv("npf_test_hidden.csv")

rownames(npf) <- npf[,"date"]

npf <- npf[,c(-1,-4)]
npf_test <- npf_test[,c(-1,-2,-4)]

npf <- npf[, c("date",  "class4", "CO242.mean", "CO2504.mean",  "Glob.mean",
               "H2O42.mean",   "H2O504.mean", "NET.mean",  "NO42.mean",    "NO504.mean",
               "NOx42.mean",   "NOx504.mean", "O342.mean", "O3504.mean",   "Pamb0.mean",
               "PAR.mean", "PTG.mean", "RGlob.mean",   "RHIRGA42.mean",    "RHIRGA504.mean",
               "RPAR.mean", "SO2168.mean", "SWS.mean", "T42.mean", "T504.mean",    "UV_A.mean",
               "CS.mean")]
```

```
npf_test <- npf_test[, c("class4", "CO242.mean",    "CO2504.mean",  "Glob.mean",
                "H2O42.mean",   "H2O504.mean", "NET.mean",  "NO42.mean",    "NO504.mean",
                "NOx42.mean",   "NOx504.mean", "O342.mean", "O3504.mean",   "Pamb0.mean",
                "PAR.mean", "PTG.mean", "RGlob.mean",   "RHIRGA42.mean",    "RHIRGA504.mean",
                "RPAR.mean", "SO2168.mean", "SWS.mean", "T42.mean", "T504.mean",    "UV_A.mean",
                "CS.mean")]

npf_test$class2 <- "nonevent"
npf_test$class4 <- "nonevent"
npf$class2 <- factor("event",levels=c("nonevent","event"))
npf$class2[npf$class4=="nonevent"] <- "nonevent"
npf <- npf[,-1]
```

```
vars <- c("CO242.mean", "CO2504.mean",  "Glob.mean",
                "H2O42.mean",   "H2O504.mean", "NET.mean",  "NO42.mean",    "NO504.mean",
                "NOx42.mean",   "NOx504.mean", "O342.mean", "O3504.mean",   "Pamb0.mean",
                "PAR.mean", "PTG.mean", "RGlob.mean",   "RHIRGA42.mean",    "RHIRGA504.mean",
                "RPAR.mean", "SO2168.mean", "SWS.mean", "T42.mean", "T504.mean",    "UV_A.mean",
                "CS.mean")
npf.pcA1 <- prcomp(npf_test[,vars], center=T, scale=T)
npf.pcA2 <- prcomp(npf[,vars], center=T, scale=T)
autoplot(npf.pcA1, data=npf_test, colour="class4", loadings=T, loadings.label=T)
```



```
autoplot(npf.pcA2, data=npf, colour="class4", loadings=T, loadings.label=T)
```



Accuracy functions:

```
accClass4 <-function(p,dataset) {
  true_vals <- 0
  for (i in 1:length(dataset$class4)) {
    if (p$Ia[i] > p$nonevent[i] & p$Ia[i] > p$II[i] &
        p$Ia[i] > p$Ib[i] & dataset$class4[i]=="Ia") {
      true_vals = true_vals + 1
    }
    if (p$Ib[i] > p$nonevent[i] & p$Ib[i] > p$II[i] &
        p$Ib[i] > p$Ia[i] & dataset$class4[i]=="Ib") {
      true_vals = true_vals + 1
    }
    if (p$II[i] > p$nonevent[i] & p$II[i] > p$Ia[i] &
        p$II[i] > p$Ib[i] & dataset$class4[i]=="II") {
      true_vals = true_vals + 1
    }
    if (p$nonevent[i] > p$II[i] & p$nonevent[i] > p$Ia[i] &
        p$nonevent[i] > p$Ib[i] & dataset$class4[i]=="nonevent") {
      true_vals = true_vals + 1
    }
  }
  return(true_vals/length(dataset$class4))
}
```

```
accClass2 <- function(p,dataset) {
  true_vals <- 0
  for (i in 1:length(dataset$class2)) {
    if (p$event[i] >=0.5 & dataset$class2[i]=="event") {
      true_vals = true_vals + 1
    }
    if (p$nonevent[i] >0.5 & dataset$class2[i]=="nonevent") {
      true_vals = true_vals + 1
    }
  }
  return(true_vals/length(dataset$class2))
}

testClass4 <-function(p) {
  ia <- 0
  ib <- 0
  ii <- 0
  nonevent <- 0
  for (i in 1:length(p$nonevent)) {
    if (p$nonevent[i] > p$II[i] & p$nonevent[i] > p$Ia[i] &
        p$nonevent[i] > p$Ib[i]) {
      nonevent = nonevent + 1
    }
    if (p$II[i] > p$nonevent[i] & p$II[i] > p$Ia[i] &
        p$II[i] > p$Ib[i]) {
      ii = ii + 1
    }
    if (p$Ia[i] > p$nonevent[i] & p$Ia[i] > p$II[i] &
        p$Ia[i] > p$Ib[i]) {
      ia = ia + 1
    }
    if (p$Ib[i] > p$nonevent[i] & p$Ib[i] > p$II[i] &
        p$Ib[i] > p$Ia[i]) {
      ib = ib + 1
    }
  }
  return(list(ia, ib, ii, nonevent))
}
testClass2 <-function(p) {
  event <- 0
  nonevent <- 0
  for (i in 1:length(p$event)) {
    if (p$event[i] >=0.5) {
      event = event + 1
    }
    if (p$nonevent[i] >0.5) {
      nonevent = nonevent + 1
    }
  }
  return(list(event, nonevent))
}
```

Binary accuracy on random forest classifier with different sized training/validation sets:

```
set.seed(42)
# Sample rows
nmbr = c(90, 180, 270)
accuracy_2 <- c()
for (i in 1:3) {
  idx <- sample.int(nrow(npf),nmbr[i])

  training_set <- npf[ idx,]
  validation_set <- npf[-idx,]

  # 10-fold Cross-validation is repeated 10 times
  # Class probabilities are collected
  # The predictions for optimal tuning parameters are saved
  # The data is scaled and centered. Principal component analysis is used to
  # find the optimal parameters
  ctrl <- trainControl(method = "repeatedcv",
                       number = 10,
                       repeats = 10,
                       classProbs = TRUE,
                       savePredictions = "final")
  rFClass2 <- train(factor(class2) ~ .,
              method="rf",
              data=training_set,
              trControl=ctrl,
              preProc=c("pca","center", "scale"))
  probs2 <- predict(rFClass2, newdata = validation_set, type = "prob")
  accuracy_2 <- c(accuracy_2, accClass2(probs2, validation_set))
}
accuracy_2
```

```
## [1] 0.9755435 0.9784173 0.9893617
```

Binary accuracy (class 2)

```
ctrl <- trainControl(method = "repeatedcv",
                     number = 10,
                     repeats = 10,
                     classProbs = TRUE,
                     savePredictions = "final")
rFClass2 <- train(factor(class2) ~ .,
            method="rf",
            data=npf,
            trControl=ctrl,
            preProc=c("pca","center", "scale"))

pred_test2 <- predict(rFClass2, newdata = npf_test)
length(which(pred_test2=="nonevent"))
```

```
## [1] 914
```

```
length(which(pred_test2=="event"))
```

```
## [1] 51
```

```
probs_test2 <- predict(rFClass2, newdata = npf_test, type = "prob")
estimates2 <- testClass2(probs_test2)
estimates2
```

```
## [[1]]
## [1] 52
##
## [[2]]
## [1] 913
```

Accuracy of the estimate of accuracy

Perplexity

Multi-class accuracy (class4)

Multiclass accuracy on the random forest classifier:

```
nmbr = c(90, 180, 270)
accuracy_4 <- c()
for (i in 1:3) {
  idx <- sample.int(nrow(npf),nmbr[i])

  training_set <- npf[ idx,]
  validation_set <- npf[-idx,]
  ctrl <- trainControl(method = "repeatedcv",
                       number = 10,
                       repeats = 10,
                       classProbs = TRUE,
                       savePredictions = "final")
  rFClass4 <- train(factor(class4) ~ .,
                    method="rf",
                    data=training_set,
                    trControl=ctrl,
                    preProc=c("pca","center", "scale"))
  probs4 <- predict(rFClass4, newdata = validation_set, type = "prob")
  accuracy_4 <- c(accuracy_4, accClass4(probs4, validation_set))
}
accuracy_4
```

```
## [1] 0.6304348 0.7302158 0.6436170
```

```
# confusionMatrix(factor(validation_set$class4), pred4)
```

```
ctrl <- trainControl(method = "repeatedcv",
                     number = 10,
                     repeats = 10,
                     classProbs = TRUE,
                     savePredictions = "final")
rFClass4 <- train(factor(class4) ~ .,
                    method="rf",
                    data=npf,
                    trControl=ctrl,
                    preProc=c("pca","center", "scale"))

pred_test4 <- predict(rFClass4, newdata = npf_test)
probs_test4 <- predict(rFClass4, newdata = npf_test, type = "prob")
# Both ways give the same answer
testClass4(probs_test4)
```

```
## [[1]]
## [1] 2
##
## [[2]]
## [1] 44
##
## [[3]]
## [1] 106
##
## [[4]]
## [1] 810
```

```
length(which(pred_test4=="nonevent"))
```

```
## [1] 811
```

```
length(which(pred_test4=="Ia"))
```

```
## [1] 2
```

```
length(which(pred_test4=="Ib"))
```

```
## [1] 45
```

```
length(which(pred_test4=="II"))
```

```
## [1] 107
```