# Supplemental

Alan n. Inglis

2021-07-27

## Packages:

```
# install the development version of vivid:
#devtools::install_github("AlanInglis/vivid")

# Load relevant packages:
library("vivid") # for visualisations
library("ranger") # to create model
library("ggplot2") # for visualisations
```

## Create Data:

Here we use Friedman's Benchmark problem 1[1] to create data from:

$$y = 10sin(\pi x1x2) + 20(x3 - 0.5)^2 + 10x4 + 5x5 + \epsilon$$

where, $ x_n  U(0, 1)$ and $\epsilon \sim N(0, 1)$ We simulate 10 variables and 1000 observations and fit our model.

```
genFriedman <- function(noFeatures = 10,
                        noSamples = 100,
                        sigma = 1,
                        bins = NULL,
                        seed = NULL,
                        showTrueY = FALSE) {
  if (!is.null(seed)) {
    set.seed(seed)
  }

  # Set Values
  n <- noSamples # no of rows
  p <- noFeatures # no of variables
  e <- rnorm(n, sd = sigma)


  # Equation:
  # y = 10sin( x1x2) + 20(x3-0.5)^2 + 10x4 + 5x5 +

  xValues <- matrix(runif(n*p, 0, 1), nrow = n)
  colnames(xValues)<- paste0("x",1:p)
```

[1] Friedman, Jerome H. (1991) Multivariate adaptive regression splines. The Annals of Statistics 19 (1), pages 1-67.

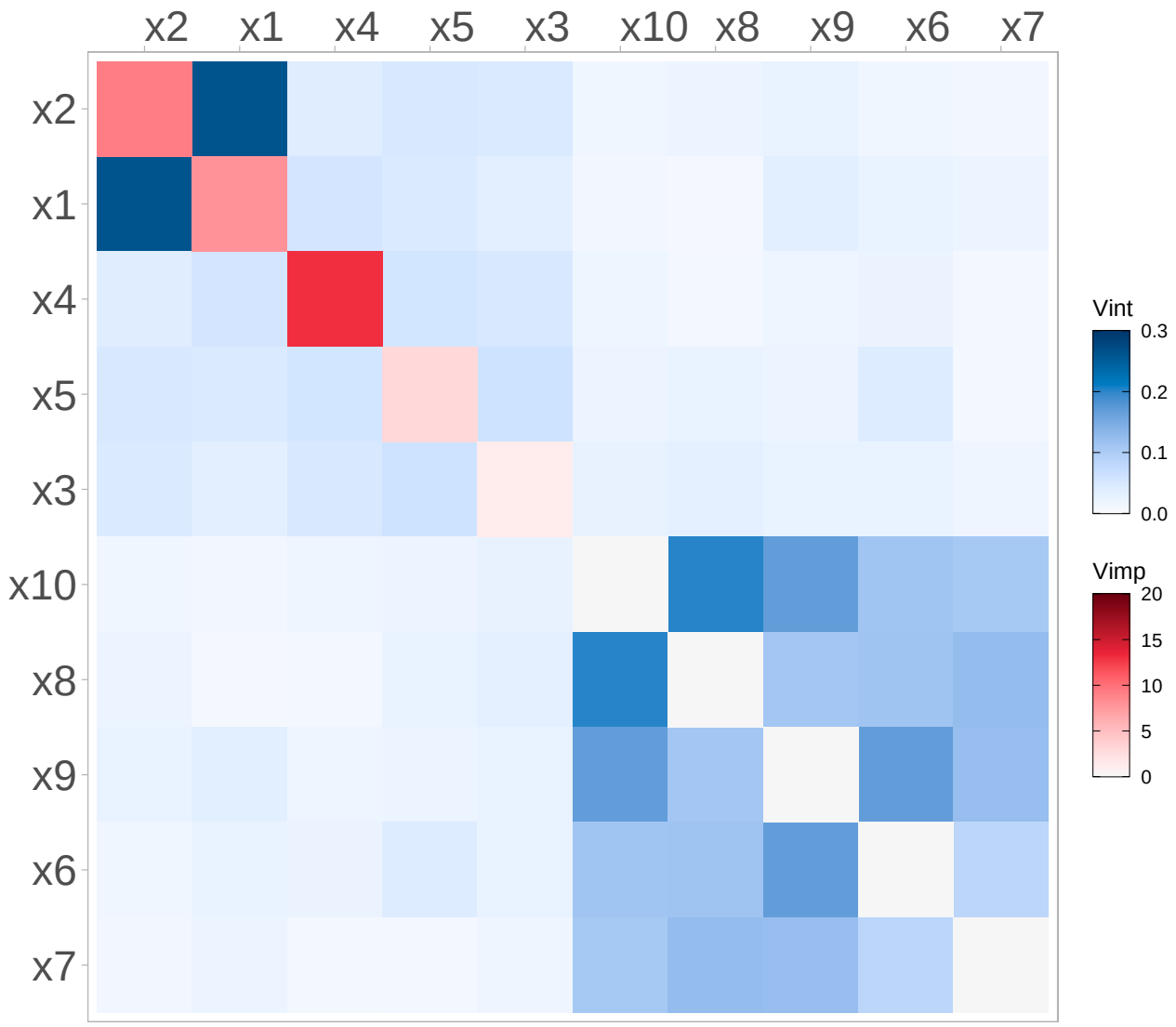
```
  yTrue <- 10 * sin(pi * xValues[, 1] * xValues[, 2]) + 20 * (xValues[, 3] - 0.5)^2 + 10 * xValues[, 4]
  y <- yTrue + e

  if (showTrueY) {
    df <- data.frame(xValues, y, yTrue)
  } else {
    df <- data.frame(xValues, y)
  }

  # Function to bin a numberic vector
  bin <- function(x, bins) {
    x <- df$y
    quantiles <- quantile(x, probs = seq(from = 0, to = 1, length = bins + 1))
    bins <- cut(x, breaks = quantiles, label = FALSE, include.lowest = TRUE)
    as.factor(paste0("class", bins))
  }

  if (!is.null(bins)) {
    bins <- as.integer(bins)
    if (bins < 2) {
      stop("bins should be an integer greater than 1.", call. = FALSE)
    }
    df$y <- bin(df$y, bins = bins)
  }

  df
}

# Data:
fData <- genFriedman(noFeatures = 10, noSamples = 1000, seed = 1701)

# Fit:
set.seed(1701)
fFit <- ranger(y ~ ., data = fData, importance = "permutation")
```

### Create vivid matrix:

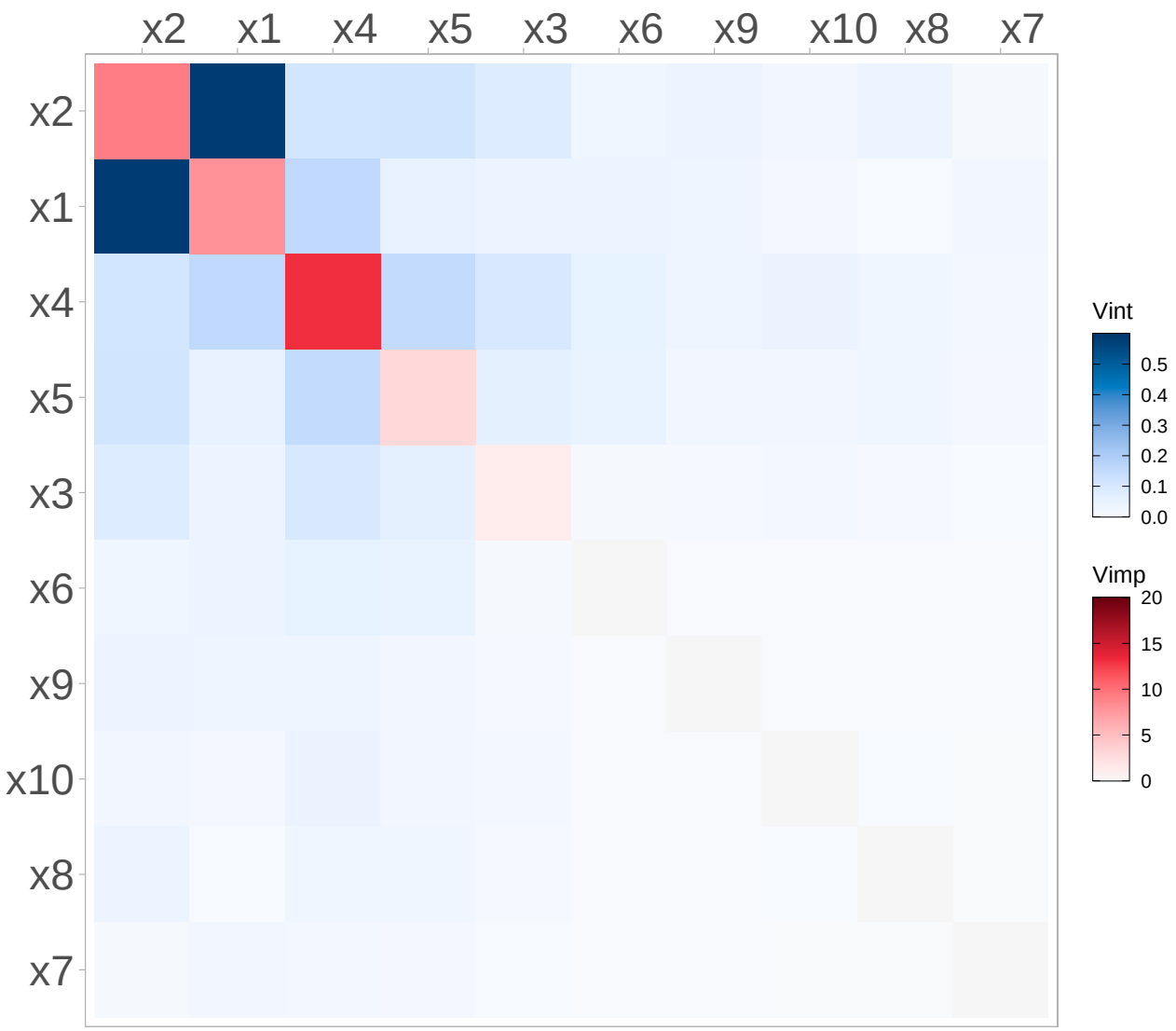
```
# vivi Normalized & Unnormalized:
set.seed(1701)
fNormalT <- vivi(fit = fFit, data = fData, response = "y", normalized = TRUE)
fNormalF <- vivi(fit = fFit, data = fData, response = "y", normalized = FALSE)
```

## Figure 10 (a) & (b)

```
viviHeatmap(fNormalT, impLims = c(0, 20))
```




```
viviHeatmap(fNormalF, intLims = c(0, 0.6), impLims = c(0, 20))
```




Now we simulate data with a correlation between x4 and x5 and refit our model:

```
set.seed(1701)
dfCorr <- fData
dfCorr$x5 <-  0.3 * fData$x5+ 0.7 * fData$x4
# checking correlation
#corrplot::corrplot(cor(dfCorr), type = "lower")

# Create  ranger model ------------------------------------------------
corrFit <- ranger(y~., data = dfCorr, importance = "permutation")
```

## Create VIVI matrix

```
# vivi Normalized & Unnormalized:
set.seed(1701)
corrNormF <- vivi(fit = corrFit, data = dfCorr, response = "y", normalize = FALSE)
```

## Figure 11 (a) & (b)

```
# Heatmap:
viviHeatmap(corrNormF, intLims = c(0, 0.6), impLims = c(0, 20))
```