Supplamental

Alan n. Inglis

2021-07-12

output: pdf_document: default editor_options: chunk_output_type: console

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("MASS") # 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:
```

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

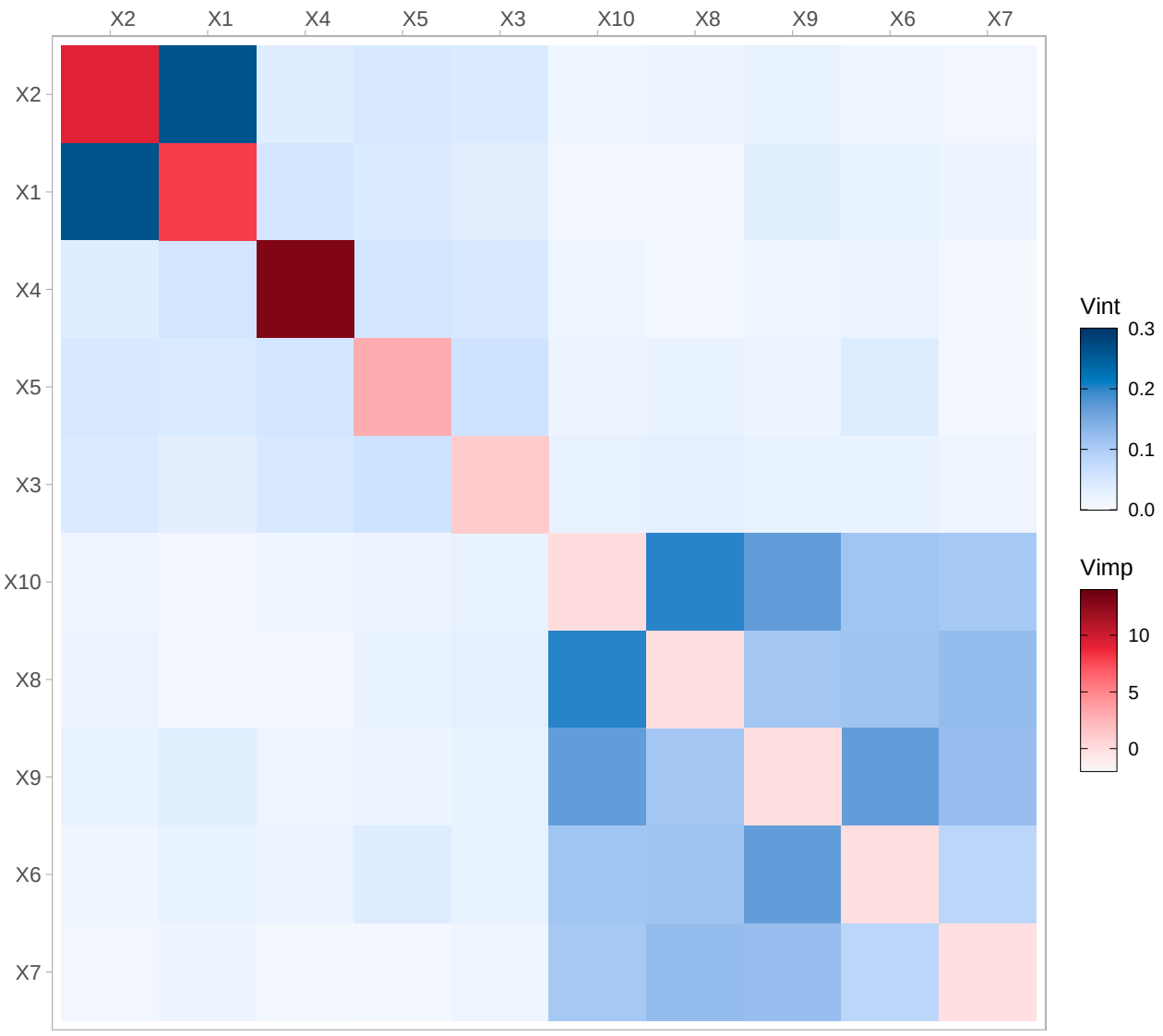
```
  # y = 10sin( x1x2) + 20(x3-0.5)^2 + 10x4 + 5x5 +

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

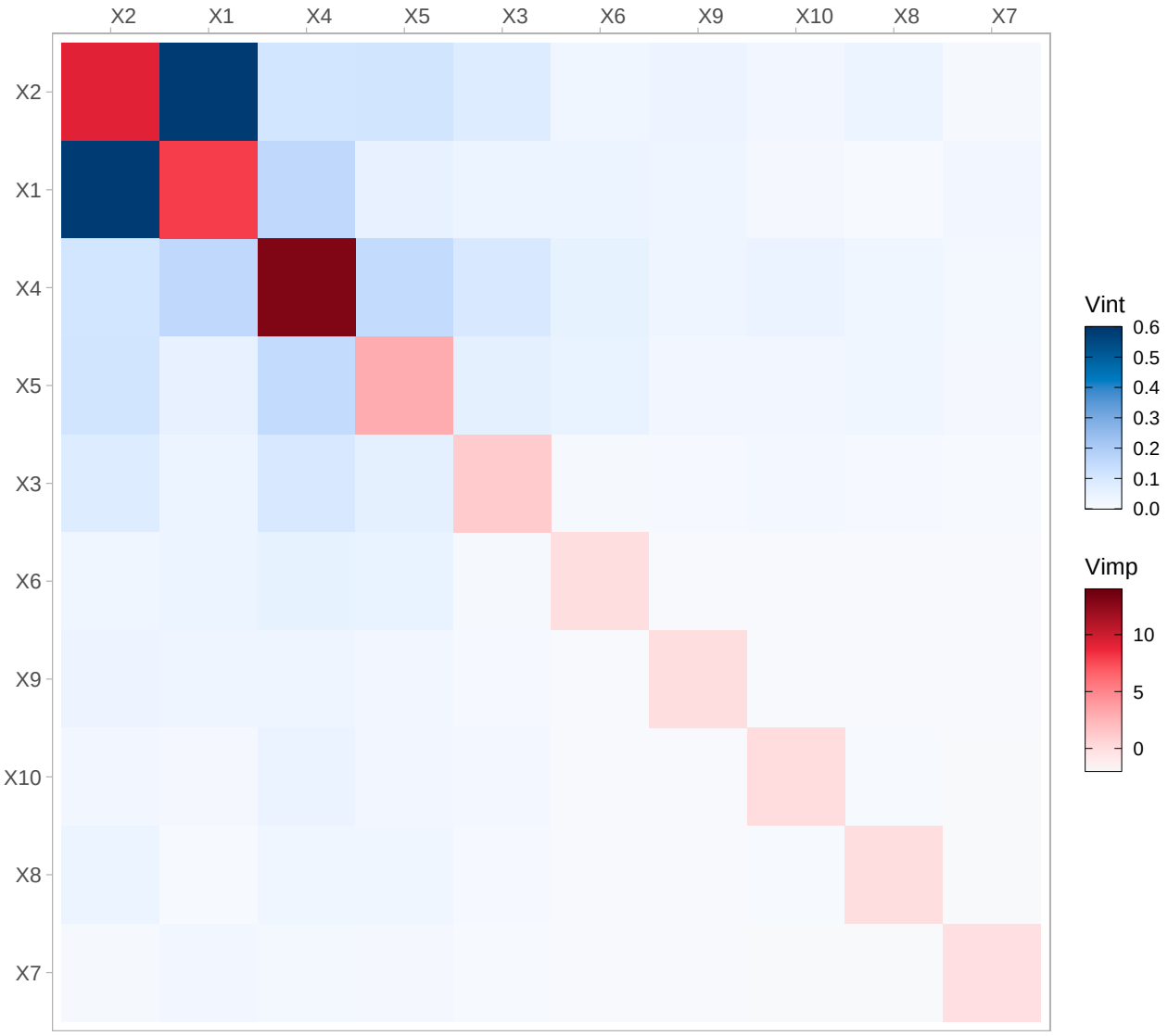
Create vivid matrix:

```
# 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)

```
# Visualisations:
viviHeatmap(fNormalT)
viviHeatmap(fNormalF)
```

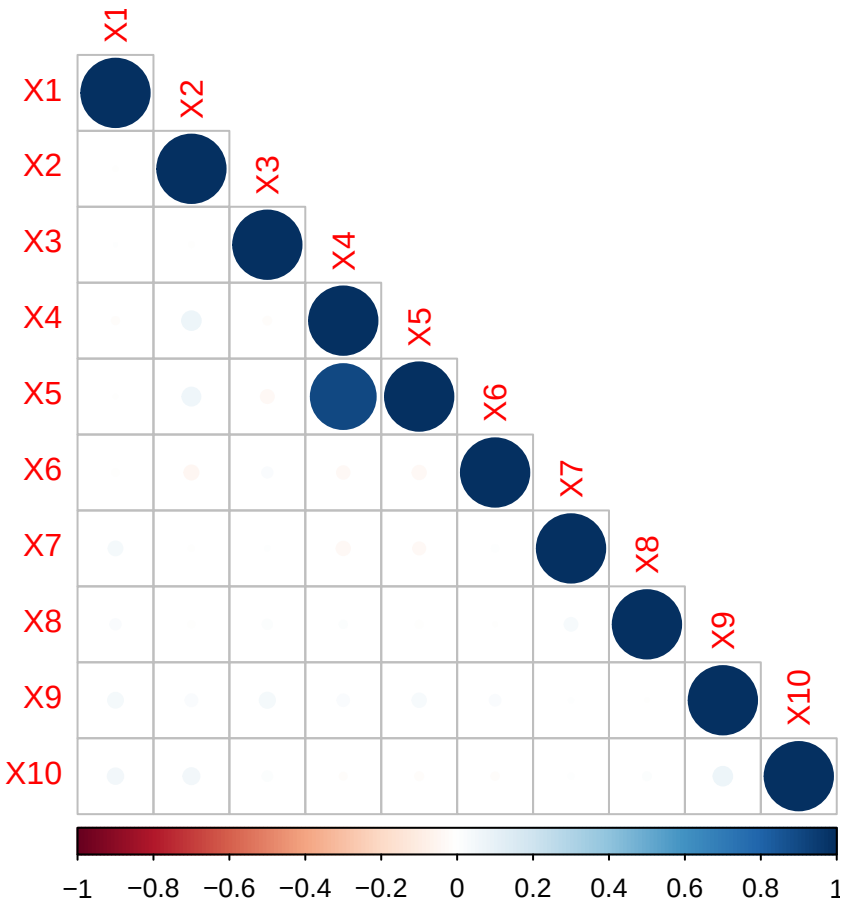

X2
X1
X4
X5
X3
X10
X8
X9
X6
X7
Vint
0.3
0.2
0.1
0.0
Vimp
10
5
0



Now we introduce a correlation between X4 and X5 and refit our model:

```
# Correlated Variables Example -------------------------------------------
set.seed(1701)
samples <- 1000
r <- 0.9 # correlation of 2 variabless will be this value
e <- rnorm(samples)

# Create correlated values:
corVals <- mvrnorm(n = samples, mu = c(0, 0), Sigma = matrix(c(1, r, r, 1), nrow = 2), empirical = TRUE)

# Create dataframe:
mvnData <- data.frame(
  X1 = runif(samples, 0 ,1),
  X2 = runif(samples, 0 ,1),
  X3 = runif(samples, 0, 1),
  X4 = corVals[, 1],
  X5 = corVals[, 2],
  X6 = runif(samples, 0 ,1),
```

```
  X7 = runif(samples, 0 ,1),
  X8 = runif(samples, 0, 1),
  X9 = runif(samples, 0 ,1),
  X10 = runif(samples, 0 ,1)
)

# Check correlations
corrplot::corrplot(cor(mvnData), type = "lower")

# Equation:
y = (10*sin(pi*mvnData$X1*mvnData$X2) + 20 * (mvnData$X3-0.5)^2 + 10 * mvnData$X4 + 5 * mvnData$X5 + e)

mvnData$y <- y

# Fit:
fFitCorr <- ranger(y ~ ., data = mvnData, importance = "permutation")
```

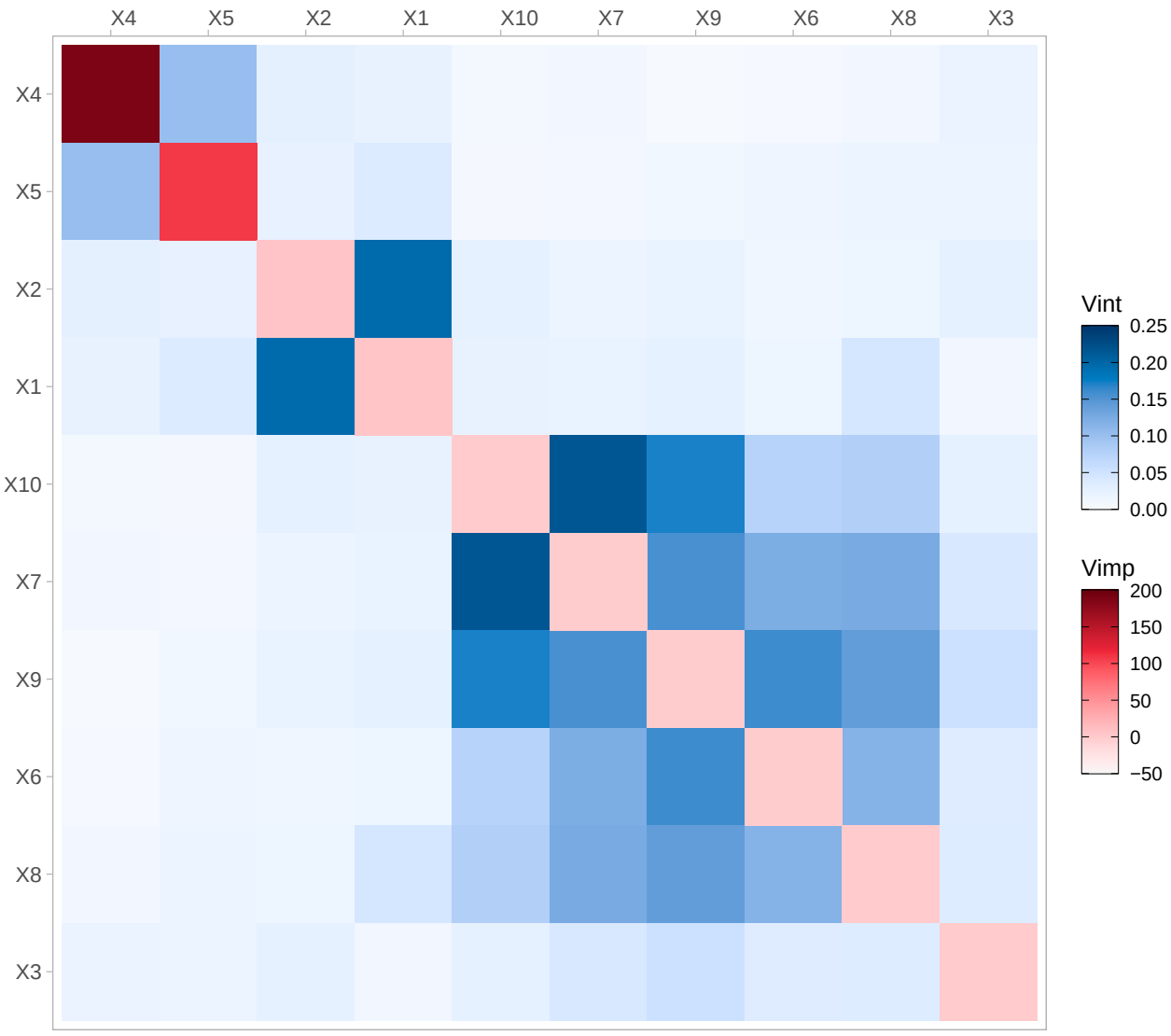


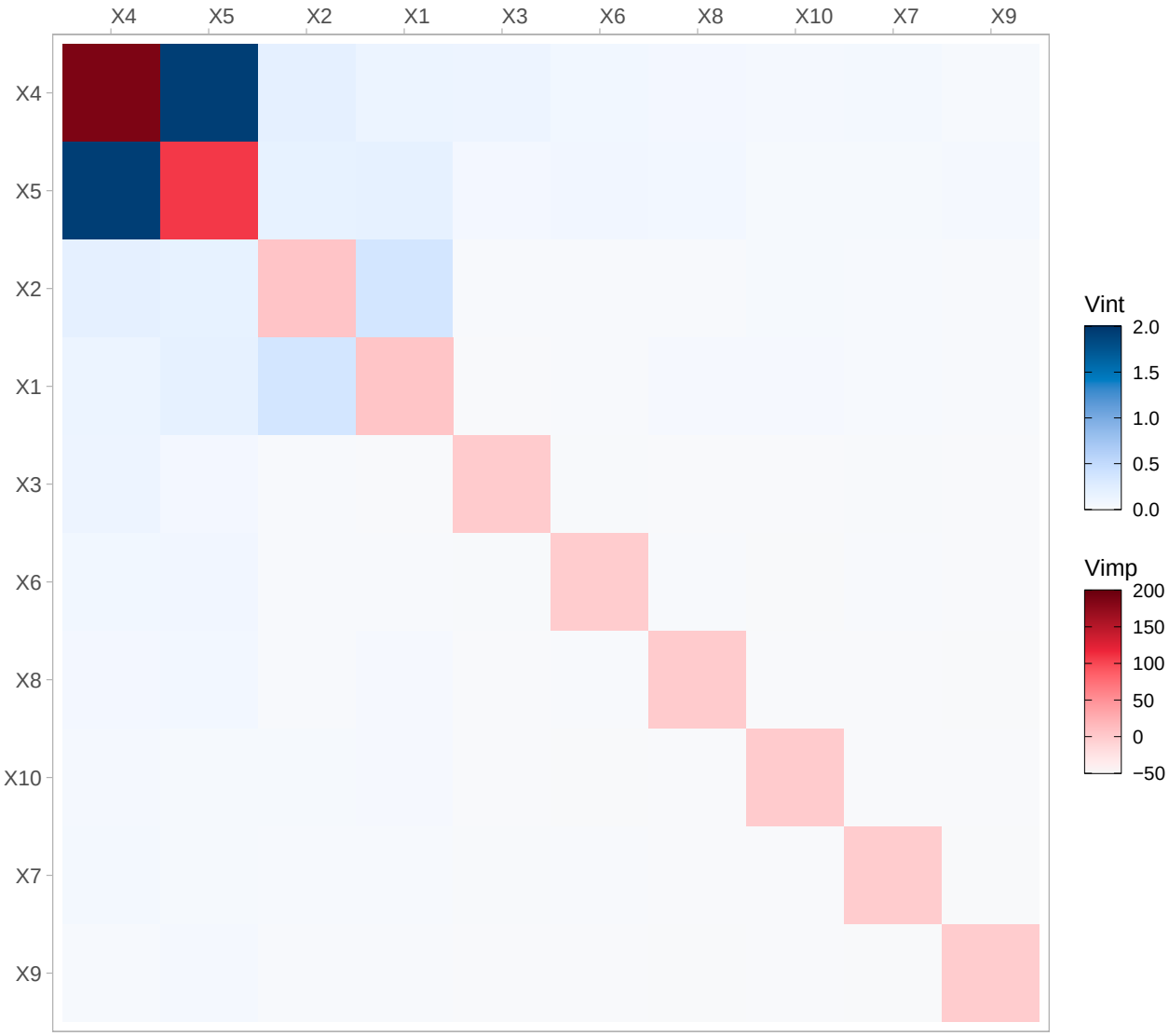
Create VIVI matrix

```
# vivi Normalized & Unnormalized:
set.seed(1701)
fNormalTCorr <- vivi(fit = fFitCorr, data = mvnData, response = "y", normalized = TRUE)
fNormalFCorr <- vivi(fit = fFitCorr, data = mvnData, response = "y", normalized = FALSE)
```

Figure 11 (a) & (b)

```
# Heatmap:
viviHeatmap(fNormalTCorr)
viviHeatmap(fNormalFCorr)
```





Look at individual PDPs

Create models with and without the correlated variables:

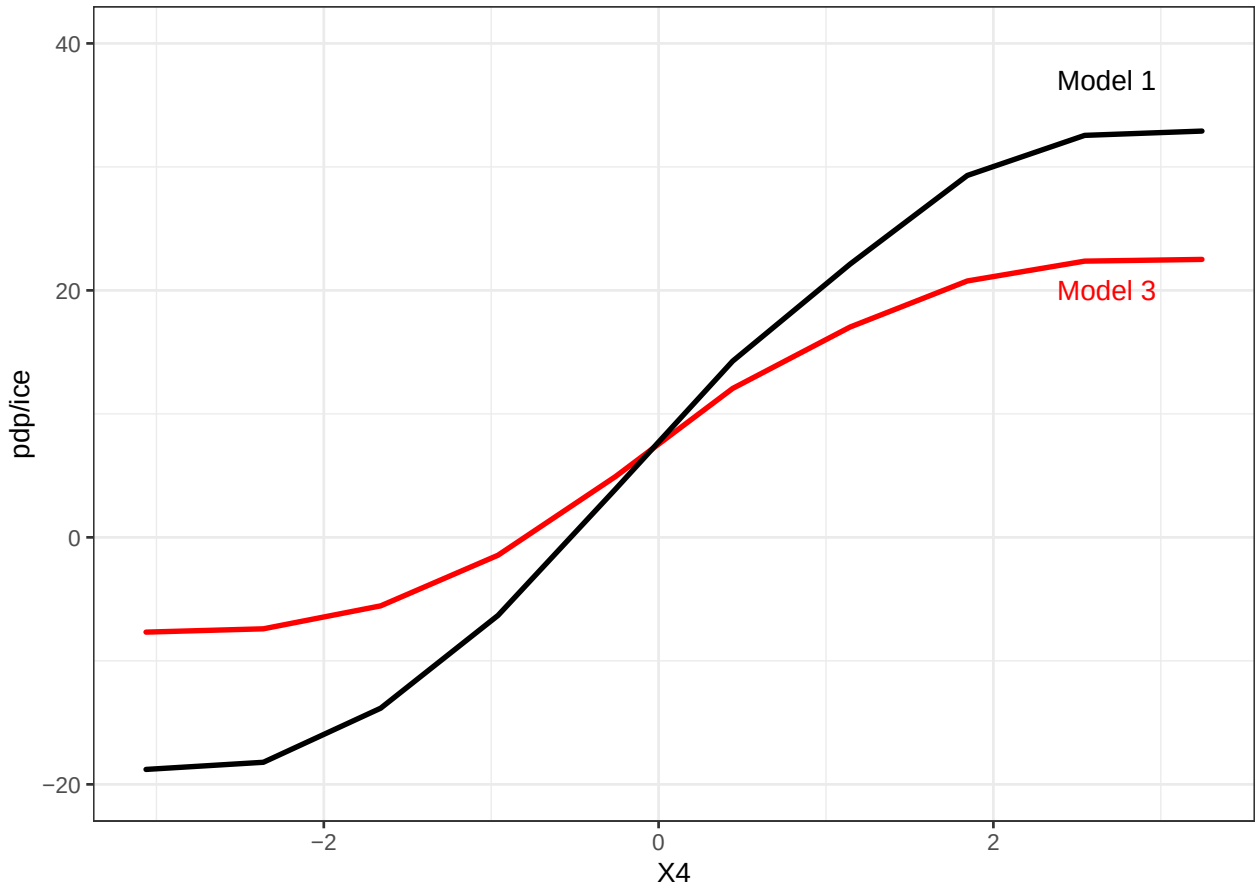
```
# Create models with & without influence of correlated variables:
set.seed(1701)
mvnFit_both <- ranger(y~ X3 + X4 + X5 + X6 + X7 + X8 + X9 + X10, data = mvnData, importance = "permutati
mvnFit_X4 <- ranger(y~ X3 + X4 + X6 + X7 + X8 + X9 + X10, data = mvnData, importance = "permutation")
mvnFit_X5 <- ranger(y~ X3 + X5 + X6 + X7 + X8 + X9 + X10, data = mvnData, importance = "permutation")
```

Create PDPs for each scenario

```r
# Visualise PDPs for each scenario:
set.seed(1701)
bothX4 <- pdpVars(mvnData, mvnFit_both, "y", nIce = 0, vars = c("X4"), limits = c(-20, 40))

bothX5 <- pdpVars(mvnData, mvnFit_both, "y", nIce = 0, vars = c("X5"), limits = c(-20, 40))

justX4 <- pdpVars(mvnData, mvnFit_X4, "y", nIce = 0,   vars = c("X4"), limits = c(-20, 40))

justX5 <- pdpVars(mvnData, mvnFit_X5, "y", nIce = 0,   vars = c("X5"), limits = c(-20, 40))
```

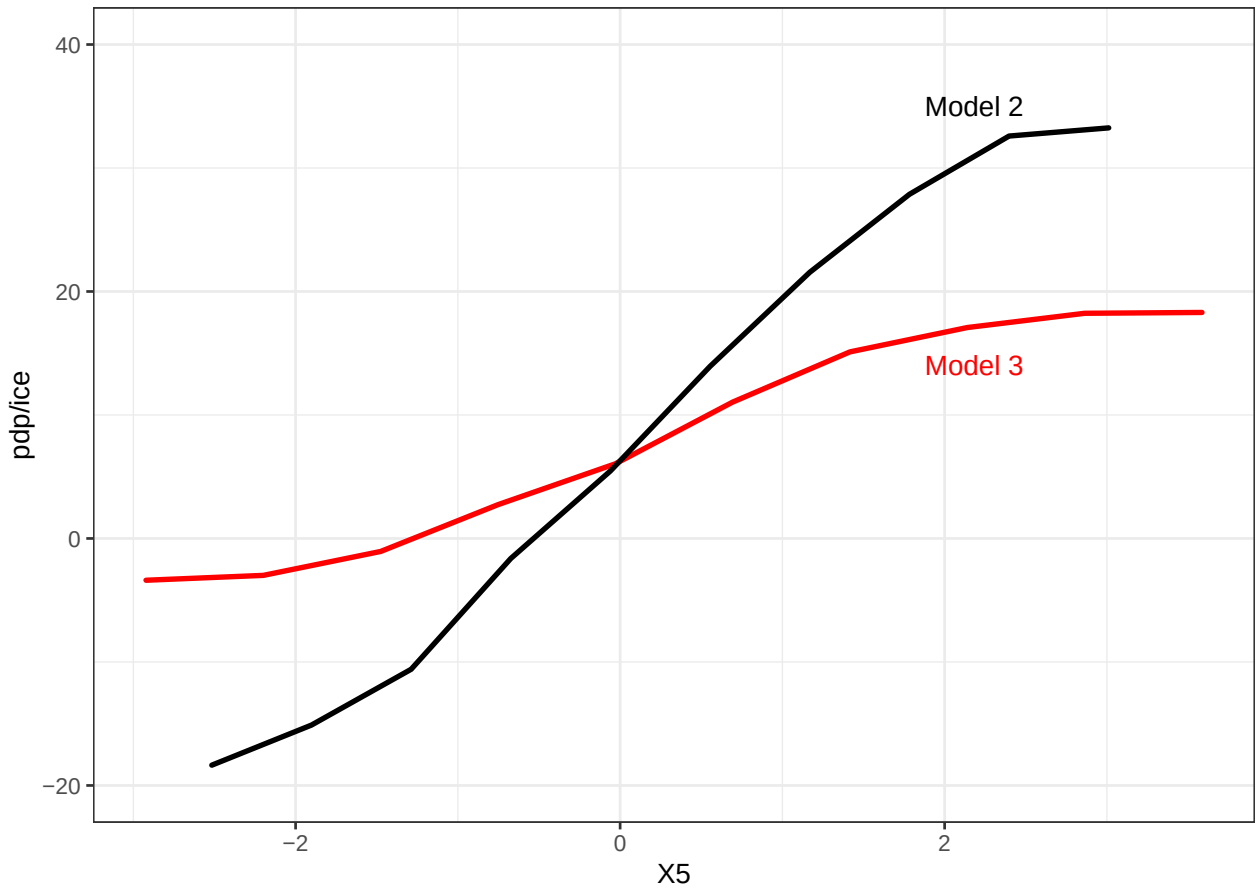
Create overlaid plot for X4:

```r
bothX4[[1]]$layers[[2]]$aes_params$colour <- "red"
pX4 <- bothX4[[1]] + justX4[[1]]$layers
aX4 <- annotate("text", x = 2.5, y = 16, label = "Model 3", vjust = -2, hjust = 0.2, colour = "red")
aX4_1 <- annotate("text", x = 2.5, y = 33, label = "Model 1", vjust = -2, hjust = 0.2, colour = "black")
pX4 + aX4 + aX4_1
```



create overlaid plot for X5:

```
bothX5[[1]]$layers[[2]]$aes_params$colour <- "red"
pX5 <- bothX5[[1]] + justX5[[1]]$layers
aX5 <-  annotate("text", x = 2, y = 10, label = "Model 3", vjust = -2, hjust = 0.2, colour = "red")
aX5_1 <- annotate("text", x = 2, y = 31, label = "Model 2", vjust = -2, hjust = 0.2, colour = "black")
pX5 + aX5 + aX5_1
```



PDPs with interacting variables ————————————————-

Create data and fit model:

```
set.seed(1701)
# Set Values
n <- 1000
p <- 5
e <- rnorm(n)



# Create matrix of values
```

```
xValues <- matrix(runif(n*p, -1, 1), nrow=n)              # Create matrix wt p columns
colnames(xValues)<- paste0("x",1:p)                       # Name columns
intDf <- data.frame(xValues)                              # Create dataframe


# Equation:
y <- 0.2 * intDf$x1 + 20 * (intDf$x2 * intDf$x3) + e
intDf$y <- y

fit <- ranger(y ~ ., data = intDf, importance = "permutation")
```

Create PDP

```
set.seed(1701)
p <- pdpVars(intDf, fit, "y", nIce = 1000, vars = c("x2"), colorVar = "x3")
```

```
p[[1]] + scale_color_gradientn(
  colors = rev(RColorBrewer::brewer.pal(11, "PuOr"))
)
```