Supplamental

Alan n. Inglis

2021-07-13

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.

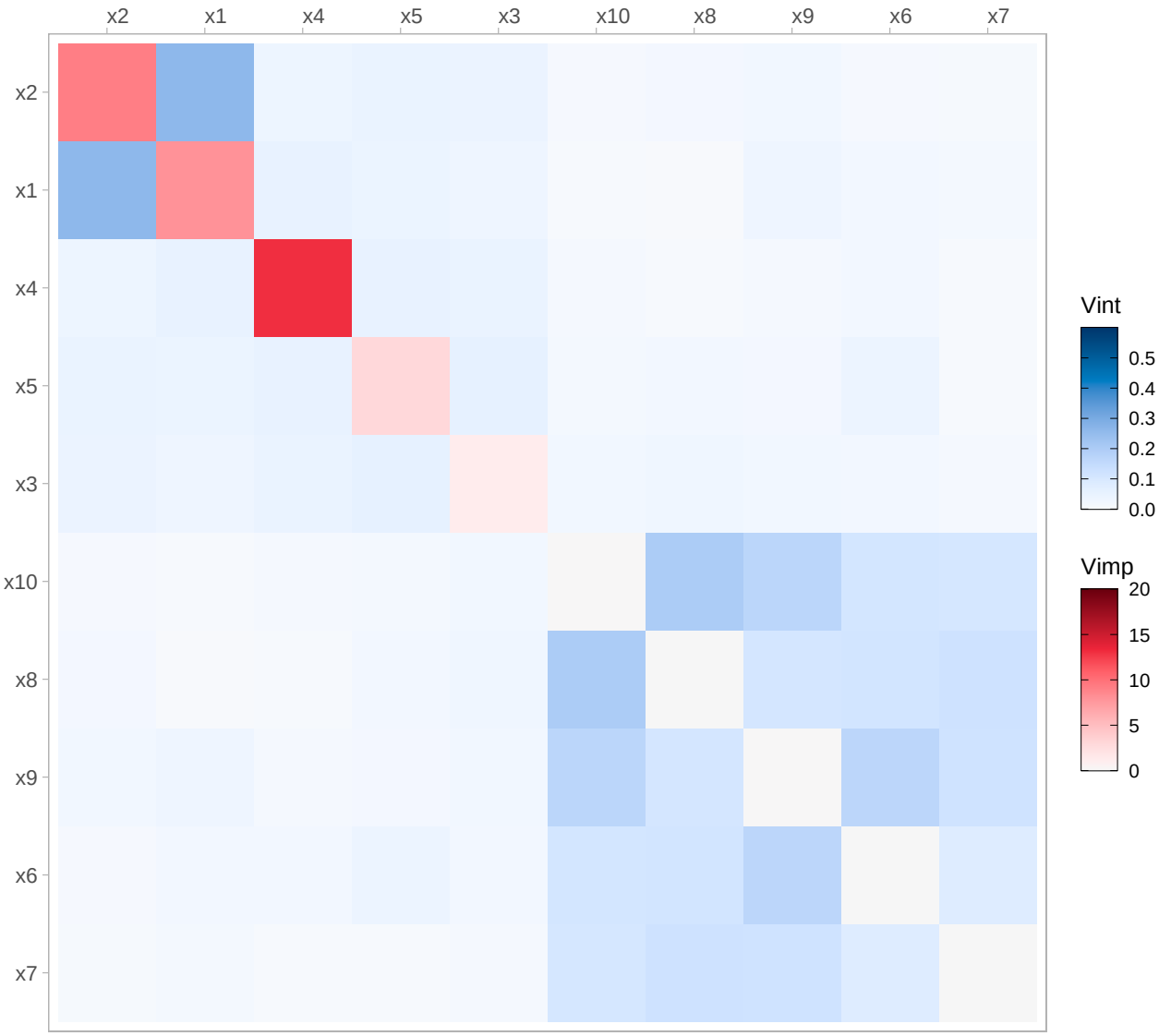
```
  yTrue <- 10 * sin(pi * xValues[, 1] * xValues[, 2]) + 20 * (xValues[, 3] - 0.5)^2 + 10 * xValues[, 4]
  y <- yTrue + e

  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")
```

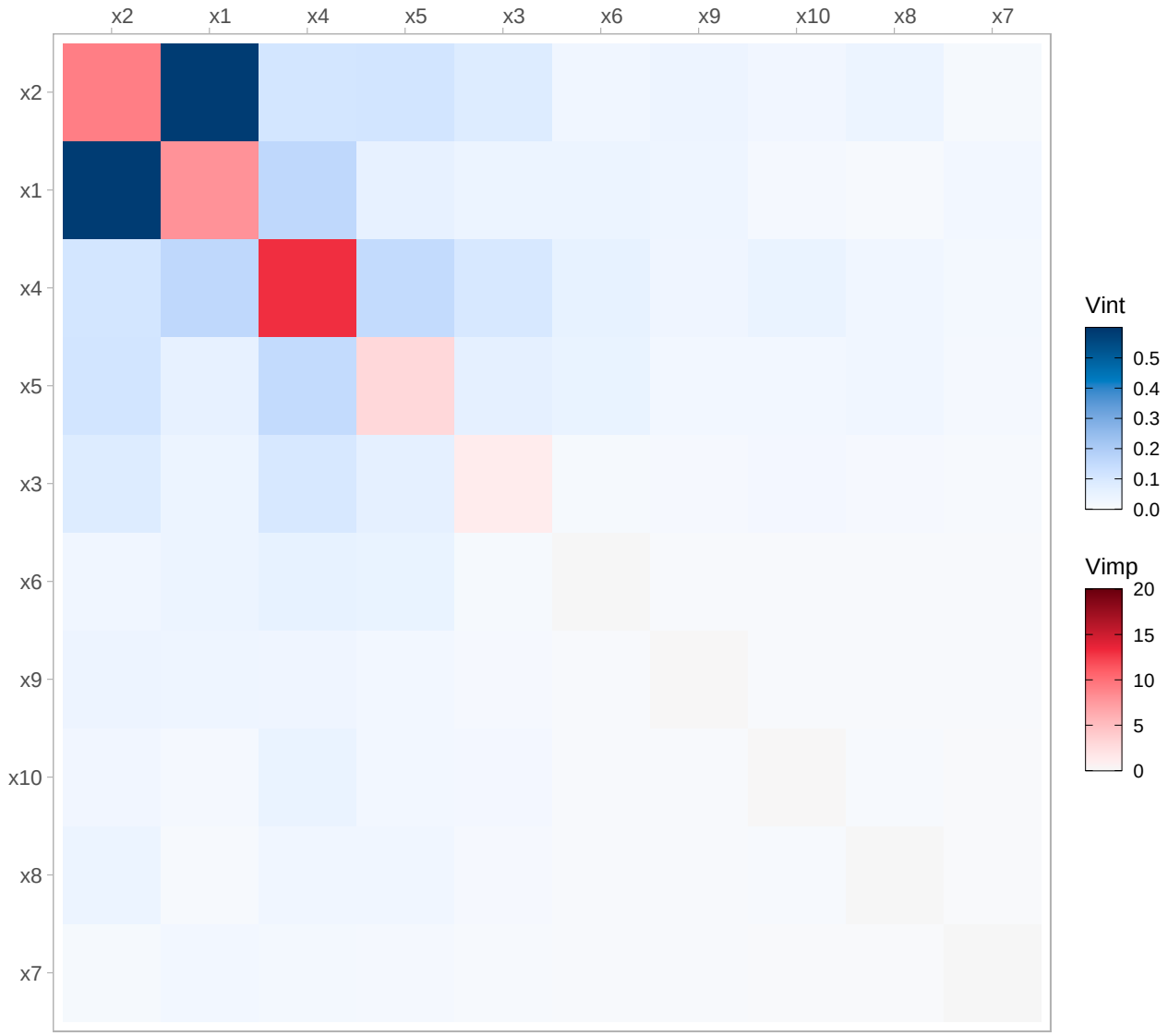
Create vivid matrix:

```
# 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)

```
# Visualisations:
viviHeatmap(fNormalT, intLims = c(0, 0.6), impLims = c(0, 20))
viviHeatmap(fNormalF, intLims = c(0, 0.6), impLims = c(0, 20))
```


x2
x1
x4
x5
x3
x10
x8
x9
x6
x7
Vint
0.5
0.4
0.3
0.2
0.1
0.0
Vimp
20
15
10
5
0



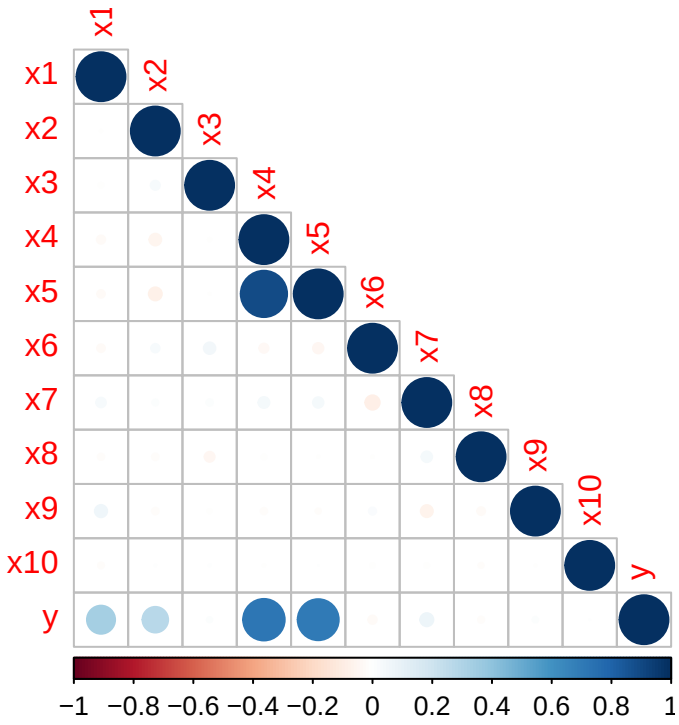
Now we simulate data with a correlation between x4 and x5 and refit our model:

```
set.seed(1701)
# Function to create correlated uniform ----------------------------------
rbvunif <- function(n, rho) {
  x <- runif(n)
  if ((rho > 1.0) || (rho < -1.0)) {
    stop("rbvunif::rho not in [-1,+1]")
  }
  else if (rho == 1.0) {
    xy <- cbind(x, x)
  } else if (rho == -1.0) {
    xy <- cbind(x, 1 - x)
  } else if (rho == 0.0) {
    xy <- cbind(x, runif(n))
  } else {
    a <- (sqrt((49 + rho) / (1 + rho)) - 5) / 2
    u <- rbeta(n, a, 1.0)
    y <- runif(n)
```

```
    y <- ifelse(y < 0.5, abs(u - x), 1 - abs(1 - u - x))
    xy <- cbind(x, y)
  }
  return(xy)
}

z <- rbvunif(1000, 0.9)
print(cor(z[, 1], z[, 2]))
```

```
## [1] 0.8970419
```

```
# Create data ------------------------------------------------------------
samples <- 1000
e <- rnorm(samples, 0, 1)

dfCorr <- data.frame(
  x1 = runif(samples, 0, 1),
  x2 = runif(samples, 0, 1),
  x3 = runif(samples, 0, 1),
  x4 = z[, 1],
  x5 = z[, 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)
)

# y = 10sin( x1x2) + 20(x3-0.5)^2 + 10x4 + 5x5 +
y <- (10*sin(pi*dfCorr$x1*dfCorr$x2) +
        20 * (dfCorr$x3-0.5)^2 + 10 * dfCorr$x4 + 5 * dfCorr$x5 + e)

# Adding y to df
dfCorr$y <- y

# checking correlation
corrplot::corrplot(cor(dfCorr), type = "lower")

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

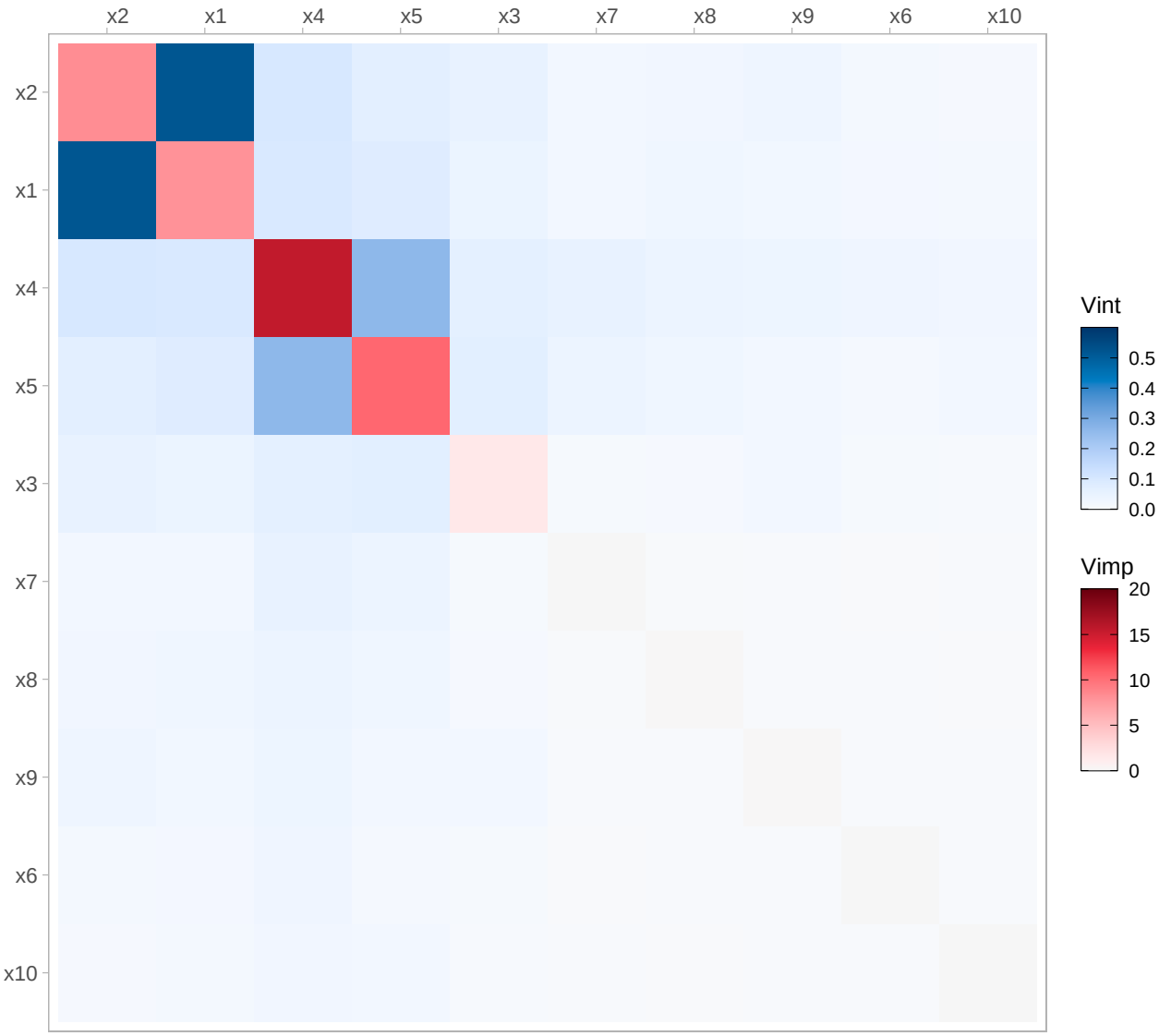


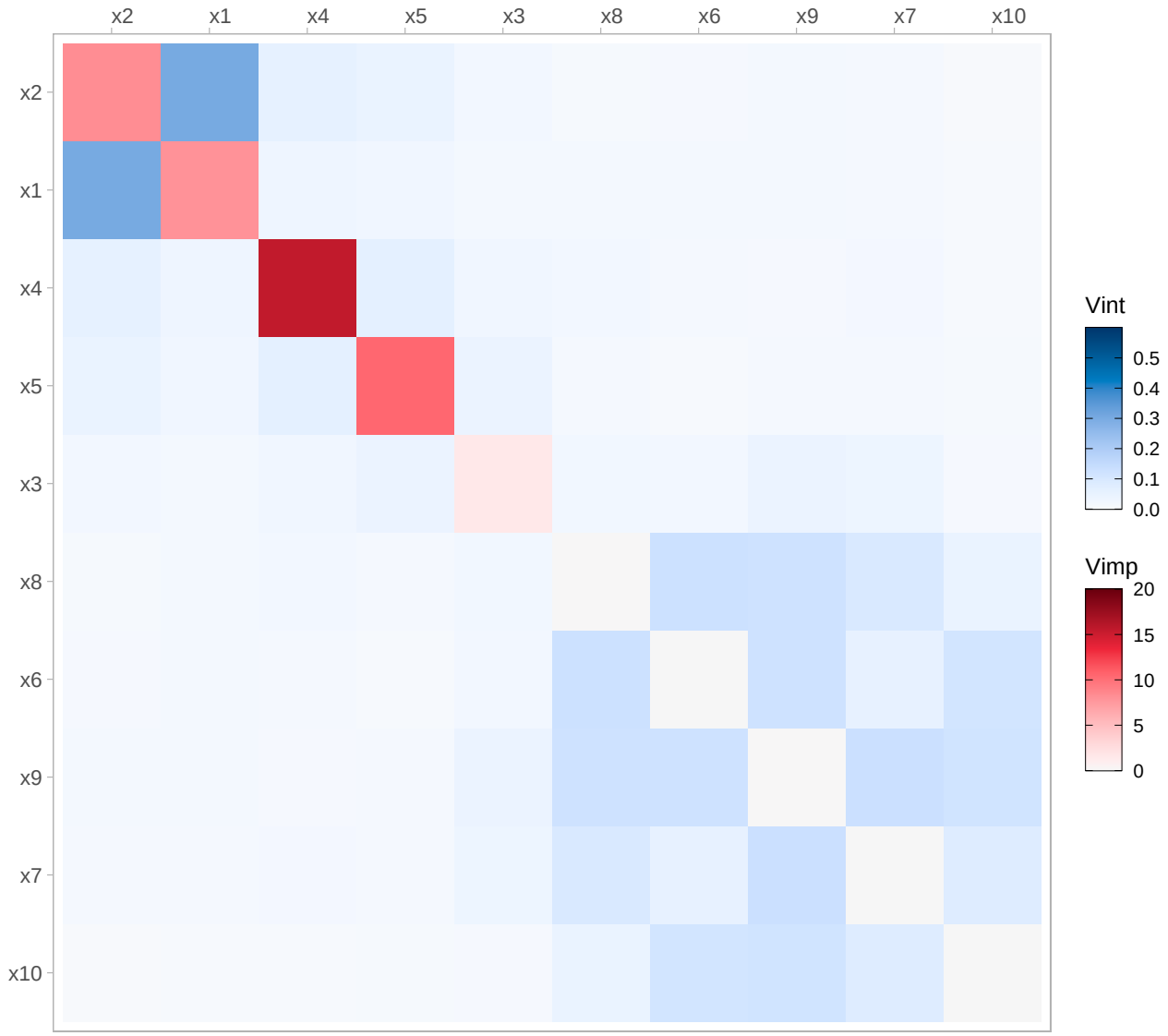
Create VIVI matrix

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

Figure 11 (a) & (b)

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

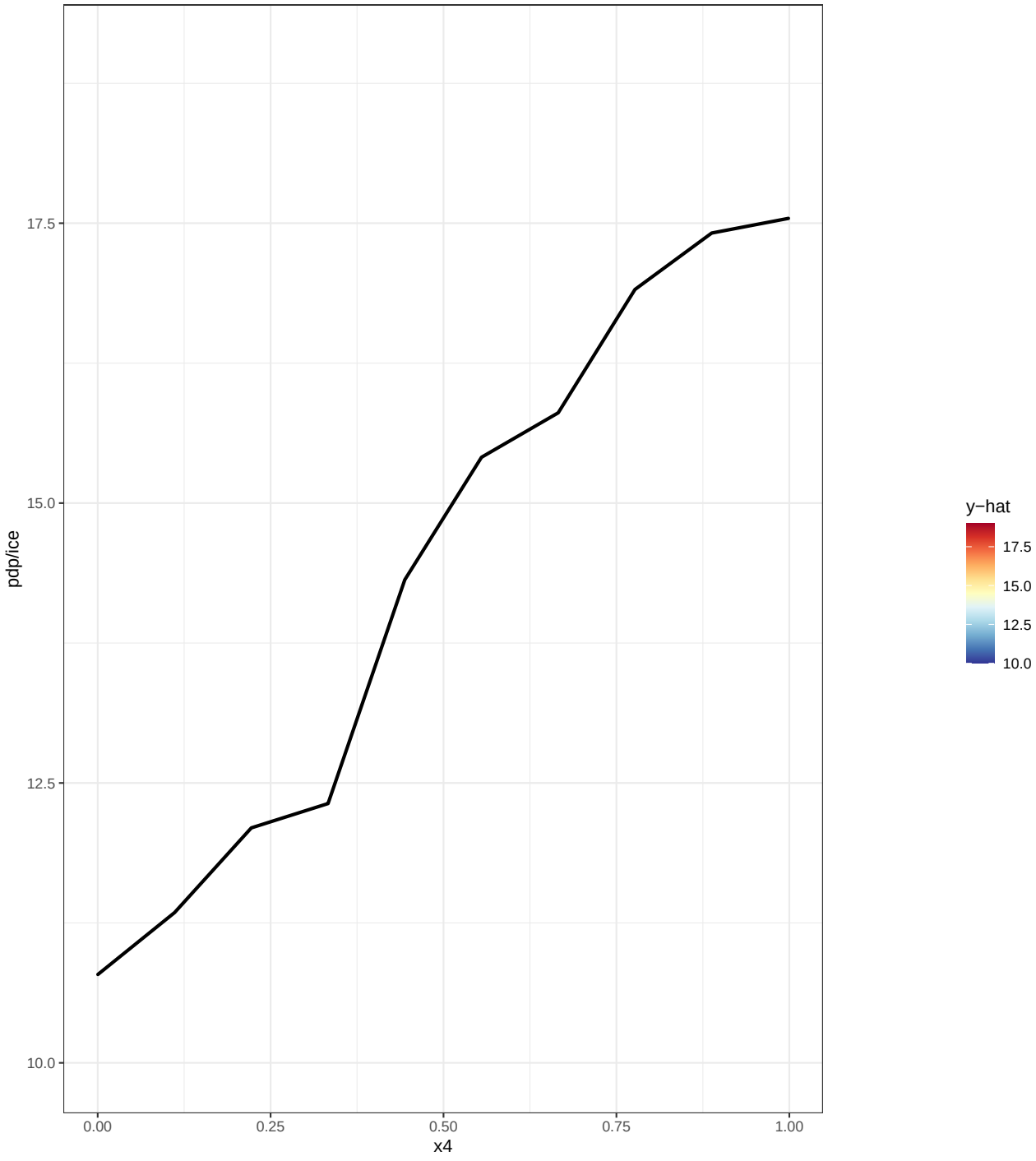

x2
x1
x4
x5
x3
x7
x8
x9
x6
x10
Vint
0.5
0.4
0.3
0.2
0.1
0.0
Vimp
20
15
10
5
0

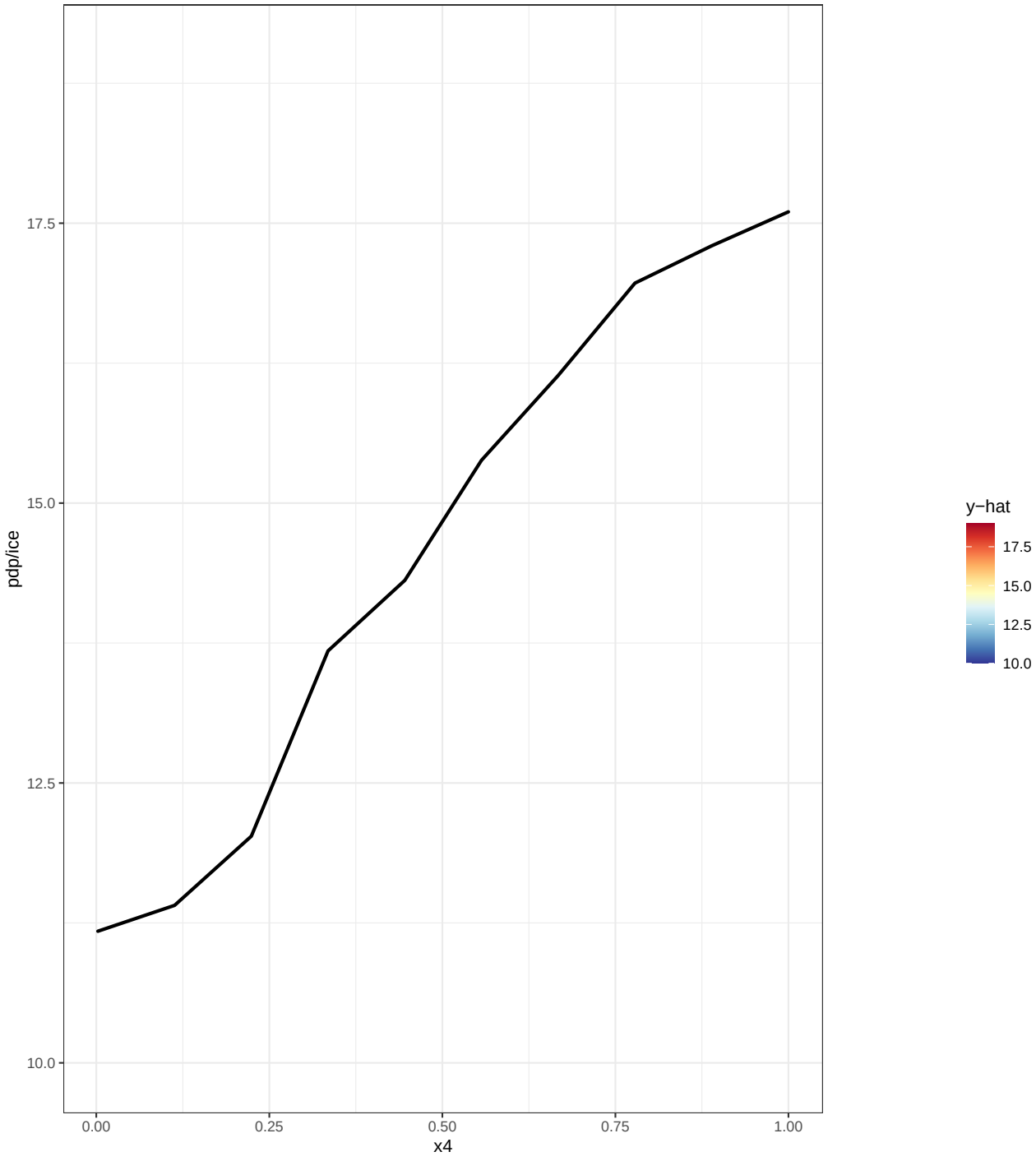


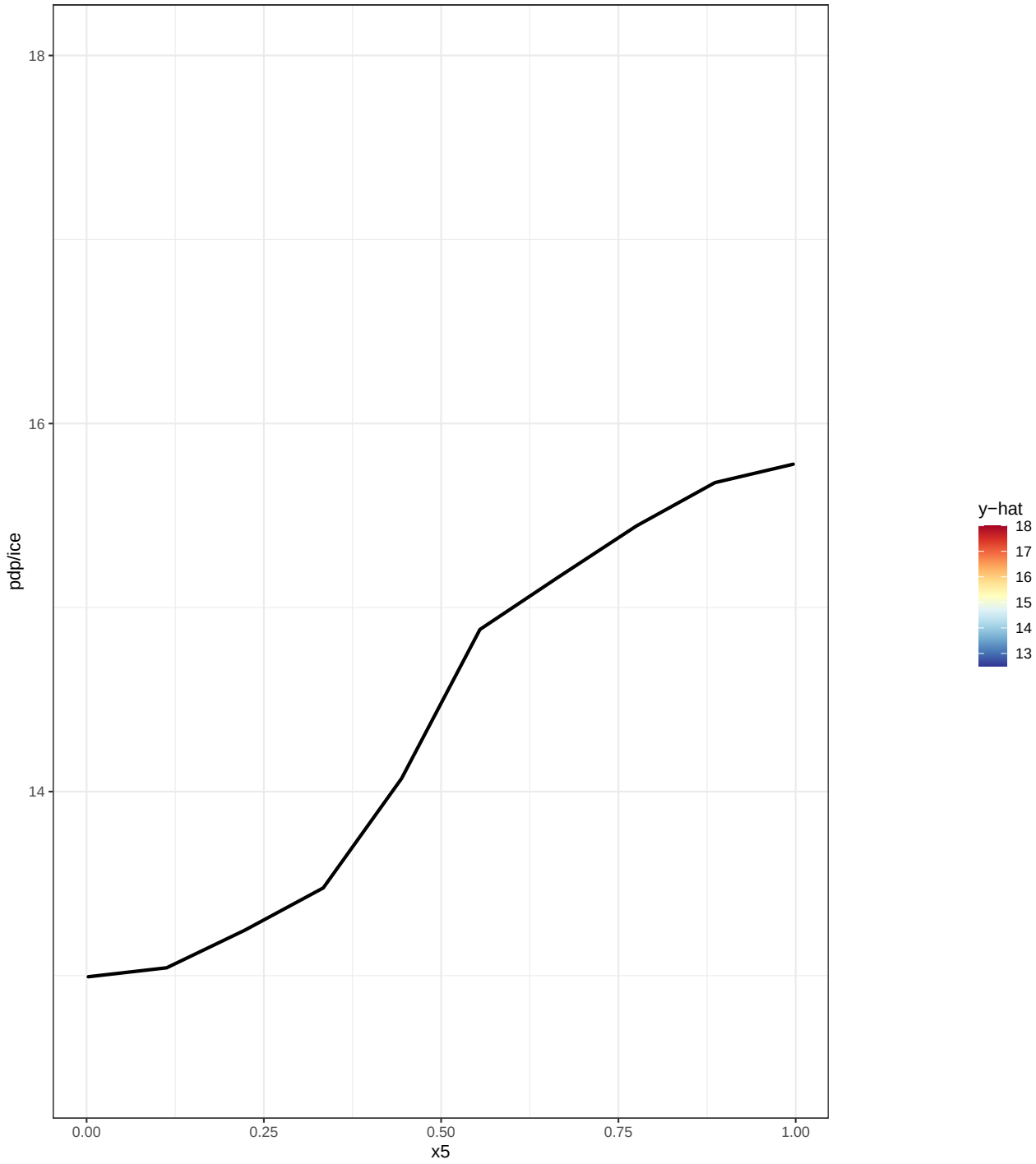
Look at individual PDPs

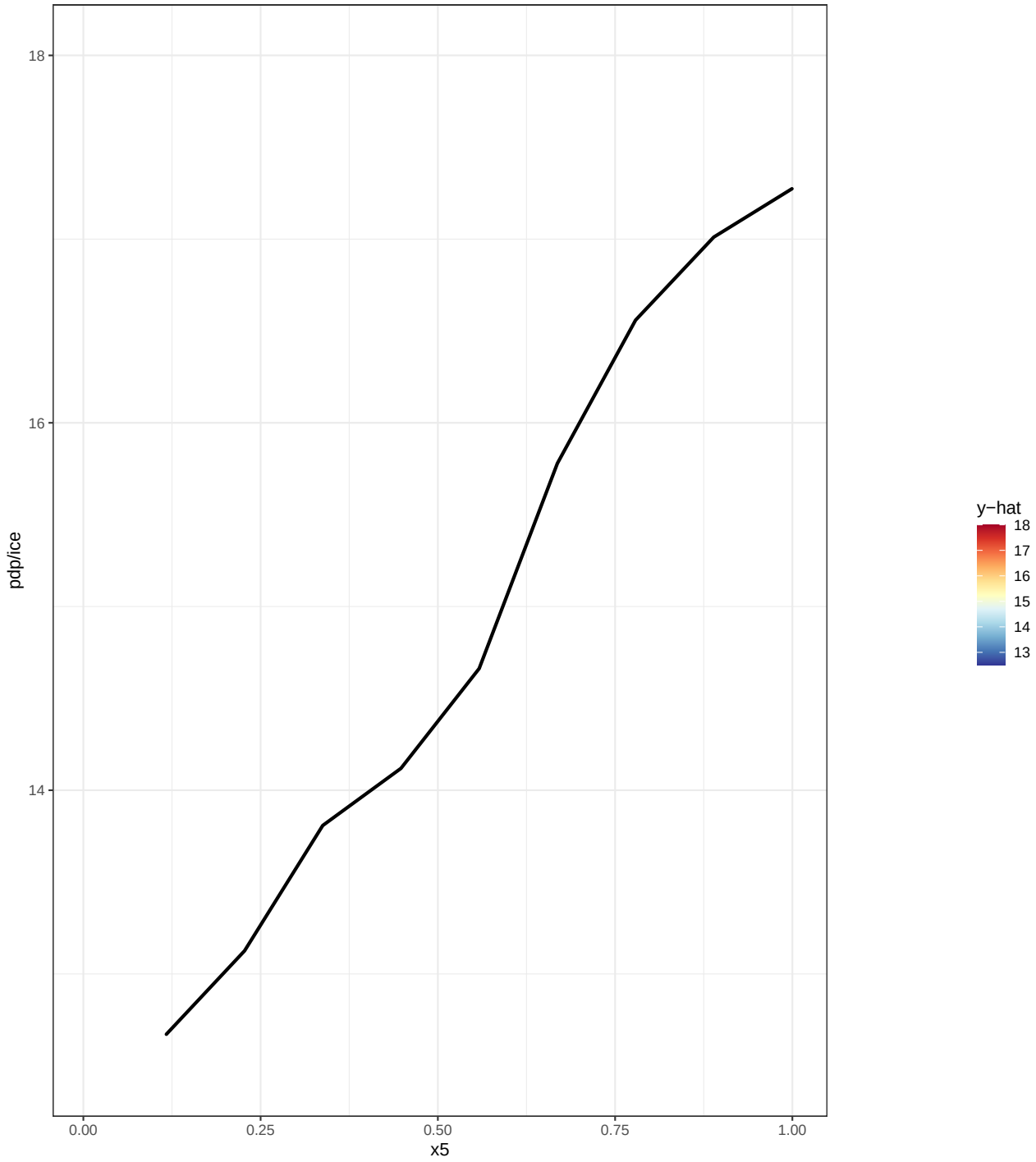
Visualise PDPs for each scenario:

```
x4CorrF <- pdpVars(fData, fFit, "y", nIce = 0, vars = c("x4"), limits = c(10, 19))
x4CorrT <- pdpVars(dfCorr, corrFit, "y", nIce = 0, vars = c("x4"), limits = c(10, 19))

x5CorrF <- pdpVars(fData, fFit, "y", nIce = 0, vars = c("x5"), limits = c(12.5, 18))
x5CorrT <- pdpVars(dfCorr, corrFit, "y", nIce = 0, vars = c("x5"), limits = c(12.5, 18))
```


17.5
15.0
12.5
10.0
pdp/ice
0.00
0.25
0.50
0.75
1.00
x4
y−hat
17.5
15.0
12.5
10.0


pdp/ice
17.5
15.0
12.5
10.0
0.00
0.25
0.50
0.75
1.00
x4
y−hat
17.5
15.0
12.5
10.0

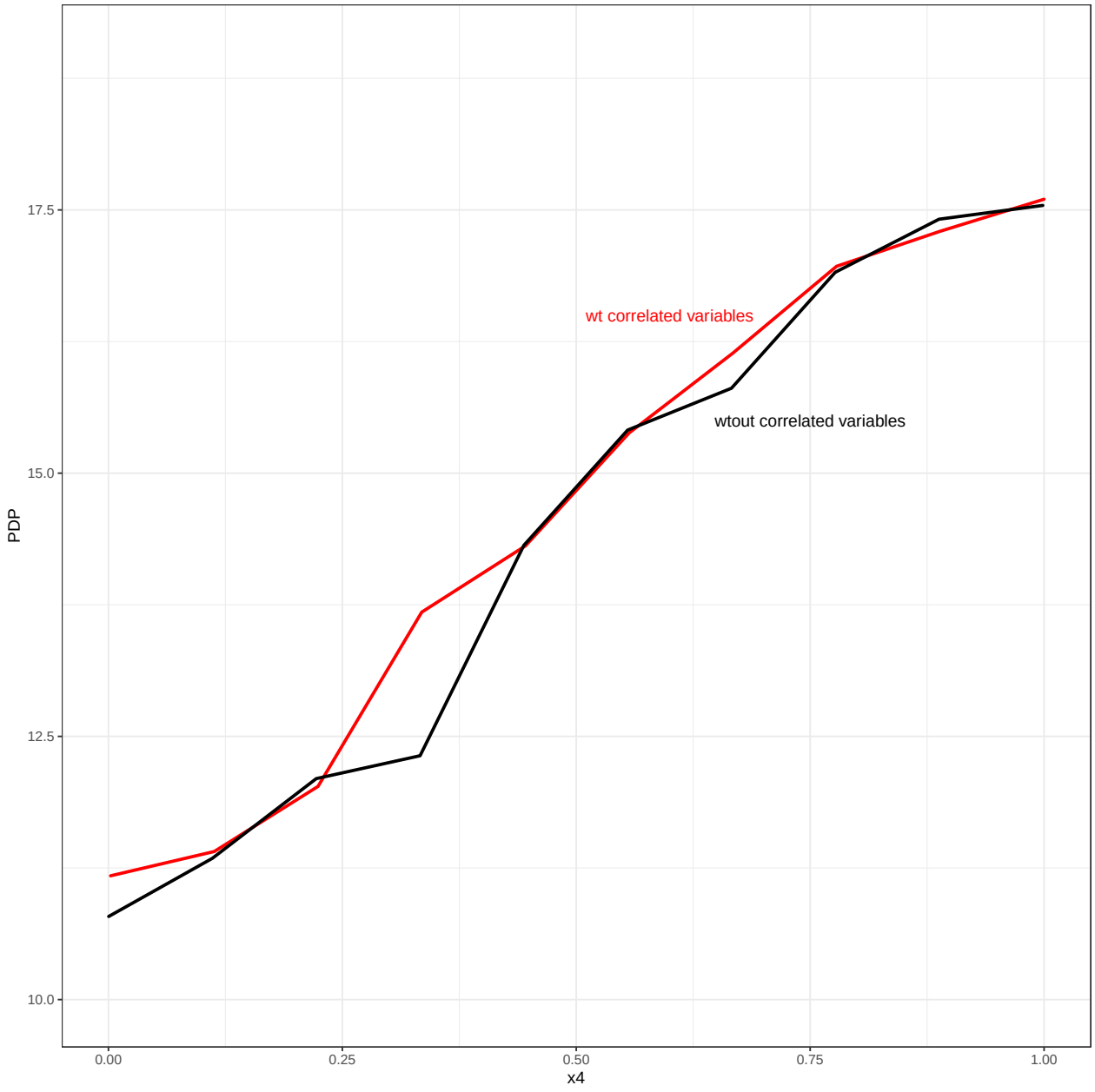

18
16
14
pdp/ice
0.00
0.25
0.50
0.75
1.00
x5
y−hat
18
17
16
15
14
13

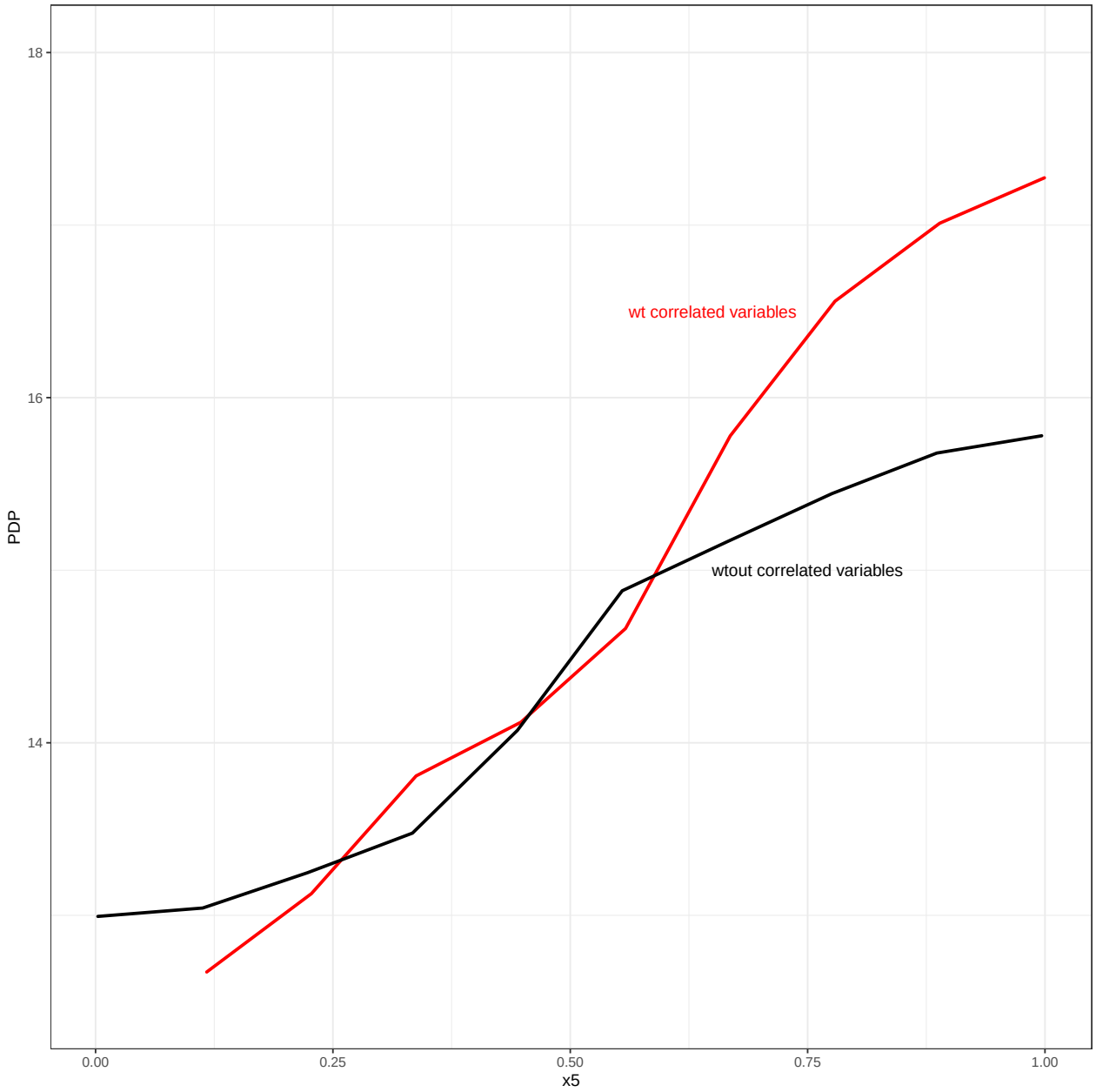


create overlaid plot for X4:

```
x4CorrT[[1]]$layers[[2]]$aes_params$colour <- "red"
ppX4 <- x4CorrT[[1]] + x4CorrF[[1]]$layers
aaX4 <- annotate("text", x = 0.6, y = 16.5, label = "wt correlated variables", colour = "red")
aaX4_1 <- annotate("text", x = 0.75, y = 15.5, label = "wtout correlated variables", colour = "black")
ppX4 + aaX4 + aaX4_1 + ylab("PDP")
```

```r
# create overlaid plot for X5:
x5CorrT[[1]]$layers[[2]]$aes_params$colour <- "red"
ppX5 <- x5CorrT[[1]] + x5CorrF[[1]]$layers
aaX5 <-  annotate("text", x = 0.65, y = 16.5, label = "wt correlated variables", colour = "red")
aaX5_1 <- annotate("text", x = 0.75, y = 15, label = "wtout correlated variables", colour = "black")
ppX5 + aaX5 + aaX5_1 + ylab("PDP")
```





Refitting models

Create models with and without the correlated variables:

```
# Create models with & without influence of correlated variables:
set.seed(1701)
Fit_both <- ranger(y~ x3 + x4 + x5 + x6 + x7 + x8 + x9 + x10, data = dfCorr, importance = "permutation")
Fit_x4 <- ranger(y~ x3 + x4 + x6 + x7 + x8 + x9 + x10, data = dfCorr, importance = "permutation")
Fit_x5 <- ranger(y~ x3 + x5 + x6 + x7 + x8 + x9 + x10, data = dfCorr, importance = "permutation")
```

Create PDPs for each scenario

```
# Visualise PDPs for each scenario:
set.seed(1701)
bothx4 <- pdpVars(dfCorr, Fit_both, "y", nIce = 0, vars = c("x4"), limits = c(7, 22))
```

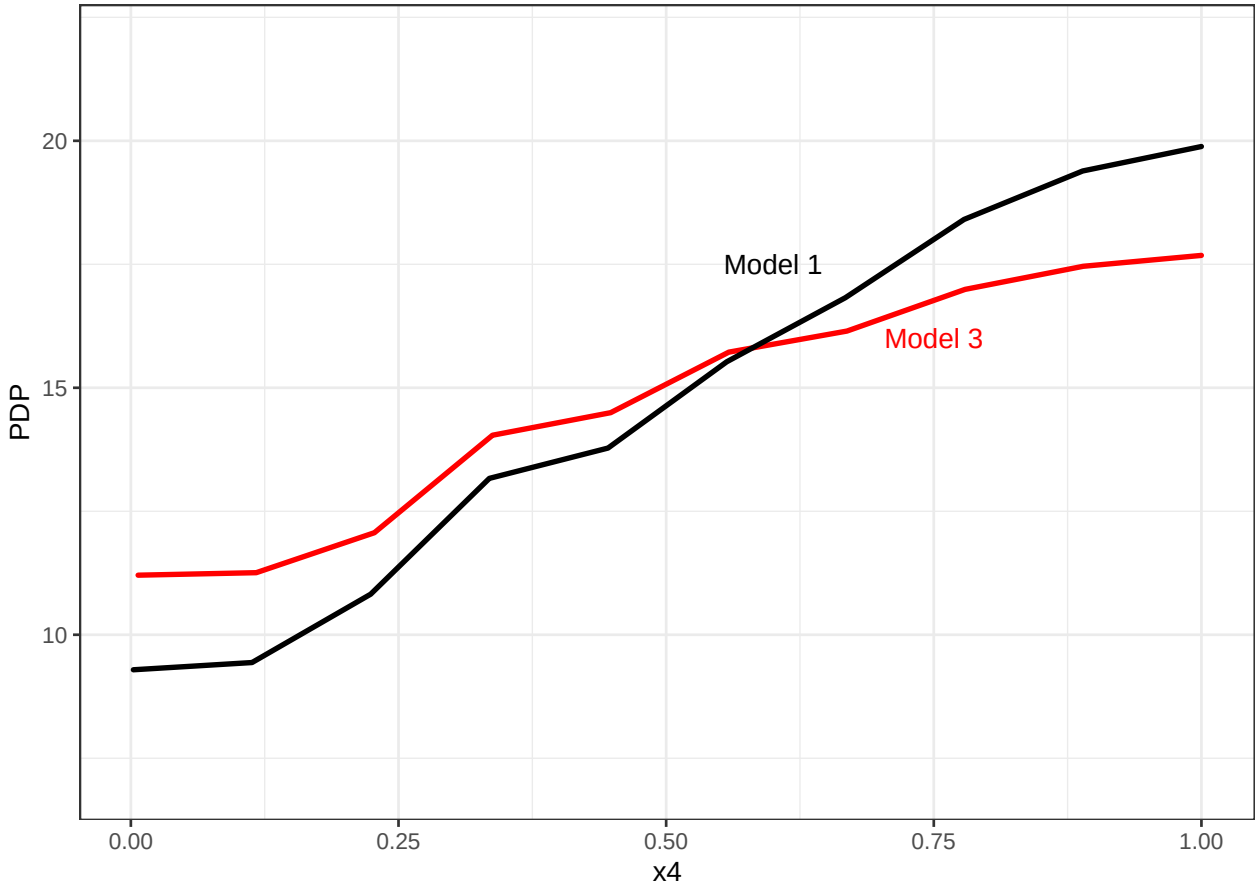
```
bothx5 <- pdpVars(dfCorr, Fit_both, "y", nIce = 0, vars = c("x5"), limits = c(7, 22))
```

```
justx4 <- pdpVars(dfCorr, Fit_x4, "y", nIce = 0,   vars = c("x4"), limits = c(7, 22))
```

```
justx5 <- pdpVars(dfCorr, Fit_x5, "y", nIce = 0,   vars = c("x5"), limits = c(7, 22))
```

Create overlaid plot for x4:

```
bothx4[[1]]$layers[[2]]$aes_params$colour <- "red"
px4 <- bothx4[[1]] + justx4[[1]]$layers
ax4 <- annotate("text", x = 0.75, y = 16, label = "Model 3", colour = "red")
ax4_1 <- annotate("text", x = 0.6, y = 17.5 , label = "Model 1", colour = "black")
px4 + ax4 + ax4_1 + ylab("PDP")
```



Create overlaid plot for x5:

```
bothx5[[1]]$layers[[2]]$aes_params$colour <- "red"
px5 <- bothx5[[1]] + justx5[[1]]$layers
ax5 <-  annotate("text", x = 0.75, y = 15, label = "Model 3",  colour = "red")
ax5_1 <- annotate("text", x = 0.6, y = 17, label = "Model 2", colour = "black")
px5 + ax5 + ax5_1 + ylab("PDP")
```