Multinomial Model

Daniel Polites

Setup

```
# Loads the MNIST dataset, saves as an .RData file if not in WD
if (!(file.exists("mnist_data.RData"))) {

  # ## installs older python version
  # reticulate::install_python("3.10:latest")
  # keras::install_keras(python_version = "3.10")
  # ## re-loads keras
  # library(keras)

  ## get MNIST data
  mnist <- dataset_mnist()
  ## save to WD as .RData
  save(mnist, file = "mnist_data.RData")

} else {
  ## read-in MNIST data
  load(file = "mnist_data.RData")
}

# Access the training and testing sets
x_train <- mnist$train$x
y_train <- mnist$train$y
x_test <- mnist$test$x
y_test <- mnist$test$y

rm(mnist)
```

```
## plot function
plot_mnist_array <- function(plt, main_label = NA, color = FALSE) {

  if (color == TRUE) {
      colfunc <- colorRampPalette(c("red", "white", "blue"))

      min_abs <- -max(abs(range(plt)))
      max_abs <- max(abs(range(plt)))

      col <- colfunc(256)
  } else {
    col <- gray((255:0)/255)
    min_abs <- 0
```

```
    max_abs <- 255
    }



  ## create image
  image(x = 1:28,
        y = 1:28,
        ## image is oriented incorrectly, this fixes it
        z = t(apply(plt, 2, rev)),
        col = col,
        zlim = c(min_abs, max_abs),
        axes = FALSE,
        xlab = NA,
        ylab = NA,
        main = ifelse(is.na(main_label),
                      "",
                      main_label))

  ## create plot border
  rect(xleft = 0.5,
       ybottom = 0.5,
       xright = 28 + 0.5,
       ytop = 28 + 0.5,
       border = "black",
       lwd = 1)

}
```

```
## train data

# initialize matrix
x_train_2 <- matrix(nrow = nrow(x_train),
                    ncol = 28*28)

## likely a faster way to do this in the future
for (i in 1:nrow(x_train)) {
  ## get each layer's matrix image, stretch to 28^2 x 1
  x_train_2[i, ] <- matrix(x_train[i, , ], 1, 28*28)
}

x_train_2 <- x_train_2 %>%
  as.data.frame()

## test data
x_test_2 <- matrix(nrow = nrow(x_test),
                   ncol = 28*28)

for (i in 1:nrow(x_test)) {
  x_test_2[i, ] <- matrix(x_test[i, , ], 1, 28*28)
}

x_test_2 <- x_test_2 %>%
```

```
  as.data.frame()


## re-scale data
x_train_2 <- x_train_2 / 256
x_test_2 <- x_test_2 / 256

## response
# x_test_2$y <- y_test
# x_train_2$y <- y_train
```

Model

train

```
## set training data size
# n <- nrow(x_train_2)
n <- 10000

indices <- sample(x = 1:nrow(x_train_2),
                  size = n)

## init data
x_multi <- x_train_2[indices, ]
y_multi <- y_train[indices]

## drop cols with all 0s
#x_multi <- x_multi[, (colSums(x_multi) > 0)]
```

```
## for the sake of the coefficients viz, setting alpha = 0
init_model <- cv.glmnet(x = x_multi %>% as.matrix,
                        y = y_multi %>% factor,
                        family = "multinomial",
                        alpha = 1)
```

```
## Warning: from glmnet C++ code (error code -58); Convergence for 58th lambda
## value not reached after maxit=100000 iterations; solutions for larger lambdas
## returned

## Warning: from glmnet C++ code (error code -58); Convergence for 58th lambda
## value not reached after maxit=100000 iterations; solutions for larger lambdas
## returned
```

```
multi_model <- predict(init_model,
                       x_multi %>% as.matrix,
                       s = init_model$lambda.min,
                       type = "response")
```

```r
## format results
preds_init <- multi_model[, , 1] %>%
  as.data.frame()

preds <- apply(X = preds_init,
               MARGIN = 1,
               FUN = function(x) names(which.max(x)) %>% as.numeric)

## TRAIN confusion matrix
table(y_multi, preds)
```

```
##        preds
## y_multi    0    1    2    3    4    5    6    7    8    9
##       0  969    1    1    1    1    4    1    2    7    1
##       1    1 1083    6    5    1    2    0    1    7    1
##       2    5    6  914    5    7    5    4    8   15    3
##       3    2    4   14  944    2   25    2    5   13    7
##       4    1    4    4    1  927    1    3    0    6   24
##       5    7    3    5   16    8  832   10    1   16    4
##       6    0    3    4    1    2    8  957    0    2    0
##       7    2    5    8    0    6    2    0 1047    0   18
##       8    2    9    3   23    6   14    6    1  928   11
##       9    5    3    0    6   11    4    0   20    5  920
```

```r
## TRAIN misclassification rate
mean(!(y_multi == preds))
```

```
## [1] 0.0479
```

test

```r
## pre-process data
x_multi_test <- x_test_2 %>%
  select(all_of(names(x_multi)))

## get preds
multi_model_test <- predict(init_model,
                            x_multi_test %>% as.matrix,
                            s = init_model$lambda.min,
                            type = "response")
```

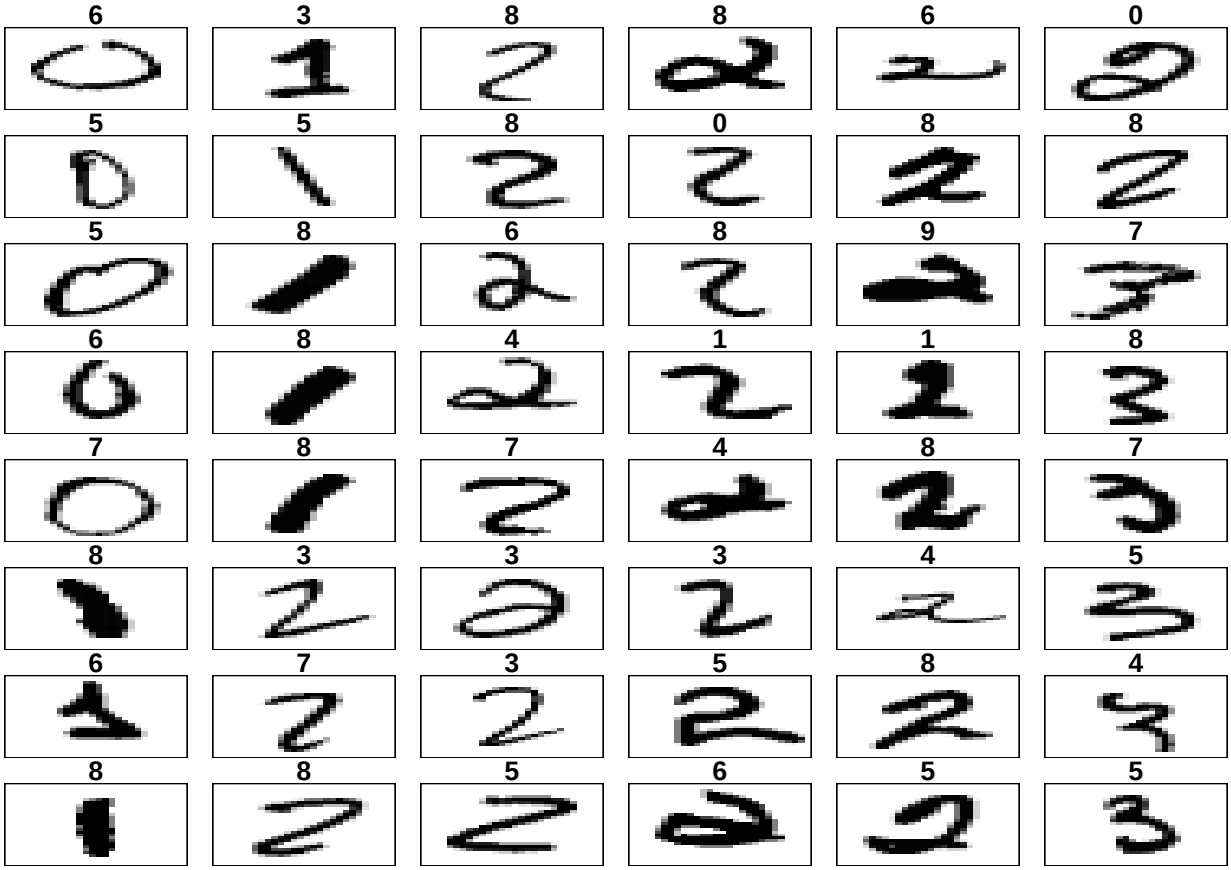
```r
## format results
preds_init_test <- multi_model_test[, , 1] %>%
  as.data.frame()

preds_test <- apply(X = preds_init_test,
               MARGIN = 1,
               FUN = function(x) names(which.max(x)) %>% as.numeric)

## TEST confusion matrix
table(y_test, preds_test)
```

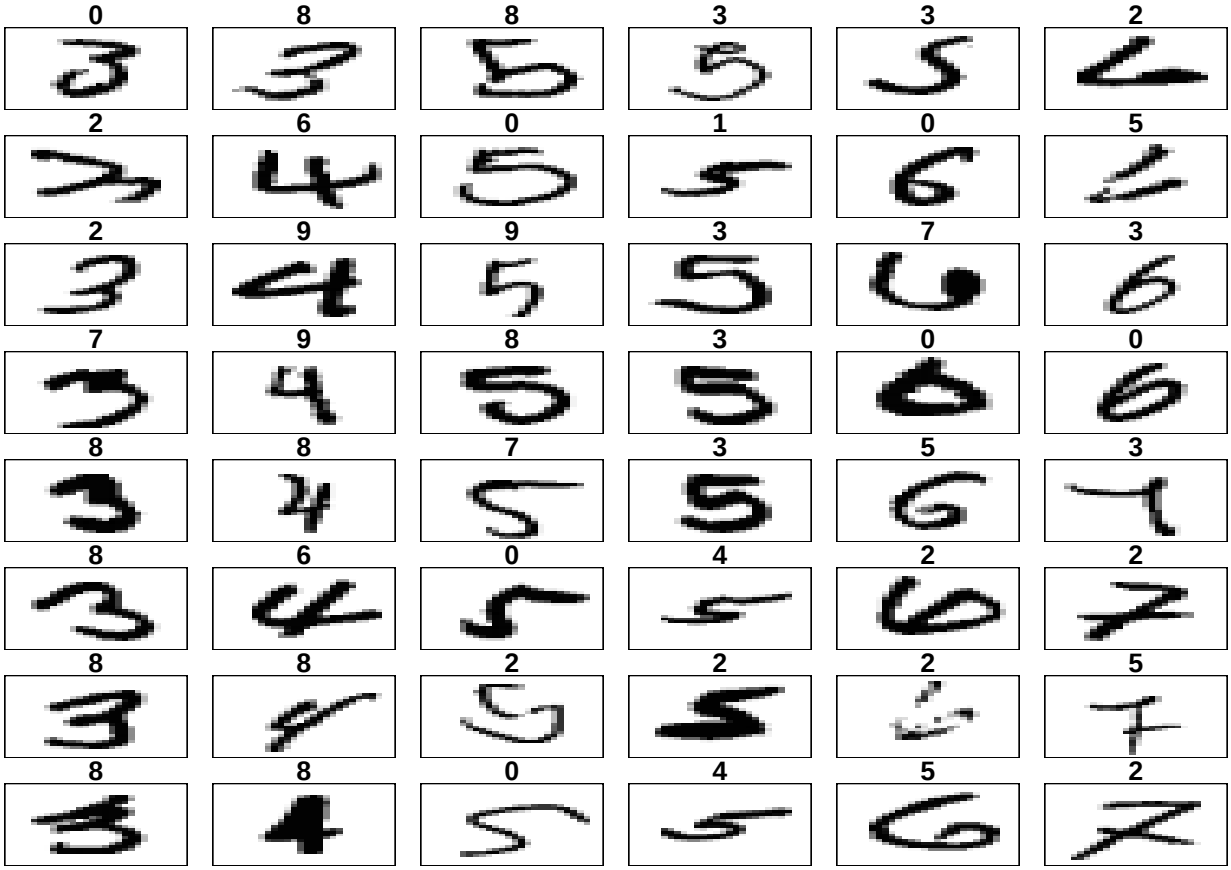
```
##       preds_test
## y_test    0    1    2    3    4    5    6    7    8    9
##      0  948    0    5    1    0    7   15    2    2    0
##      1    0 1110    3    2    1    2    3    2   12    0
##      2   12   14  900   25    9    4   18    7   37    6
##      3    3    2   23  900    2   32    2   11   28    7
##      4    1    3    5    0  913    0   12    6    9   33
##      5   11    4    3   36   12  765   14    8   32    7
##      6   11    3   12    2   10   20  897    1    2    0
##      7    1   12   24   10    9    2    2  933    4   31
##      8    6   13    8   26   12   35   14   12  831   17
##      9   13    6    3   11   36    8    0   27   10  895
```

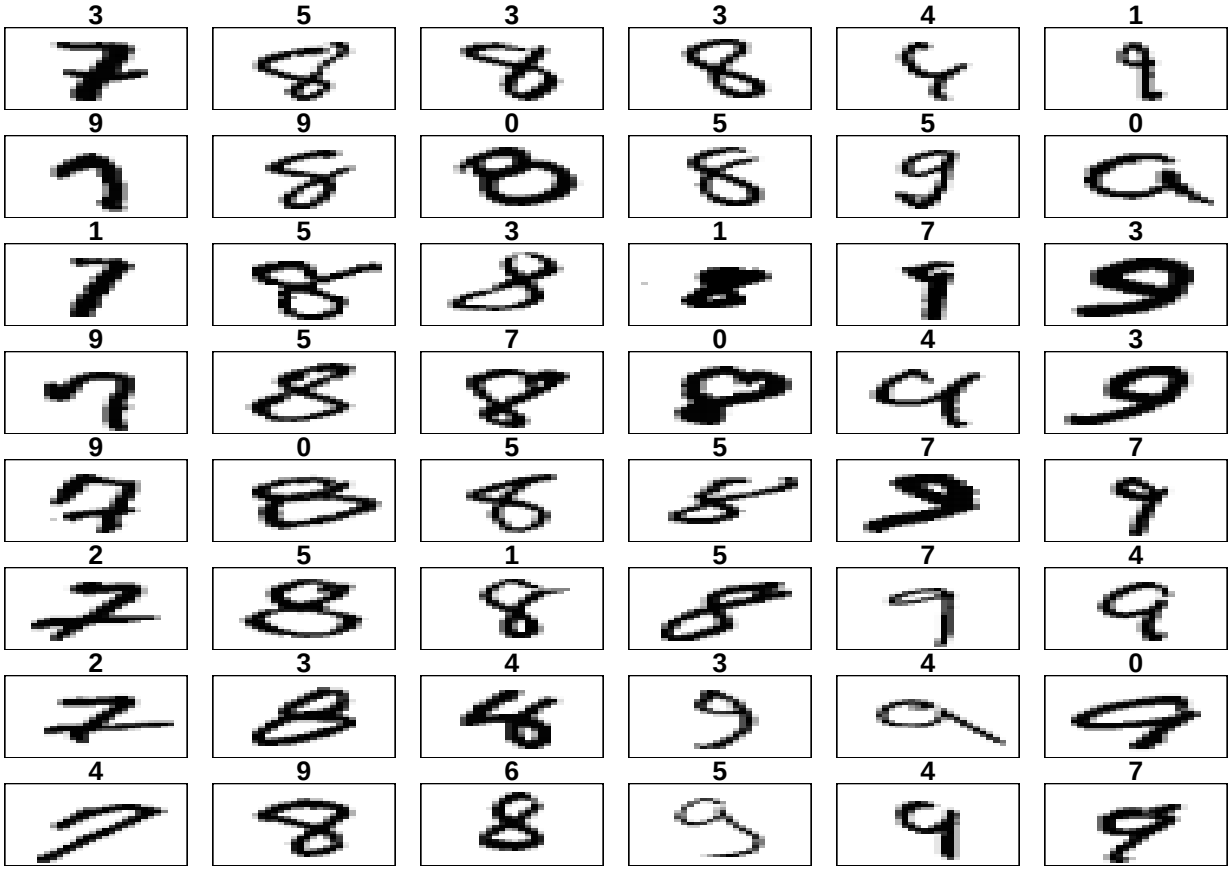
```
## TEST misclassification rate
mean(!(y_test == preds_test))
```

```
## [1] 0.0908
```

```
## sort vectors so outputs are grouped
x_test_sort <- x_test[order(y_test), , ]
y_test_sort <- y_test[order(y_test)]
preds_test_sort <- preds_test[order(y_test)]

## get misclassified obs
wrong <- which(!(y_test_sort == preds_test_sort))

## plot a sample of misclassified obs
plot_wrong <- wrong[sample(x = 1:length(wrong), size = 3*8*6)] %>%
  sort()

## plot params
par(mfcol = c(8, 6))
par(mar = c(0, 0.5, 1, 0.5))

for (i in plot_wrong) {
  plot_mnist_array(plt = x_test_sort[i, , ],
                   main_label = preds_test_sort[i])
}
```


6 3 8 8 6 0
5 5 8 0 8 8
5 8 6 8 9 7
6 8 4 1 1 8
7 8 7 4 8 7
8 3 3 3 4 5
6 7 3 5 8 4
8 8 5 6 5 5


0 8 8 3 3 2
2 6 0 1 0 5
2 9 9 3 7 3
7 9 8 3 0 0
8 8 7 3 5 3
8 6 0 4 2 2
8 8 2 2 2 5
8 8 0 4 5 2

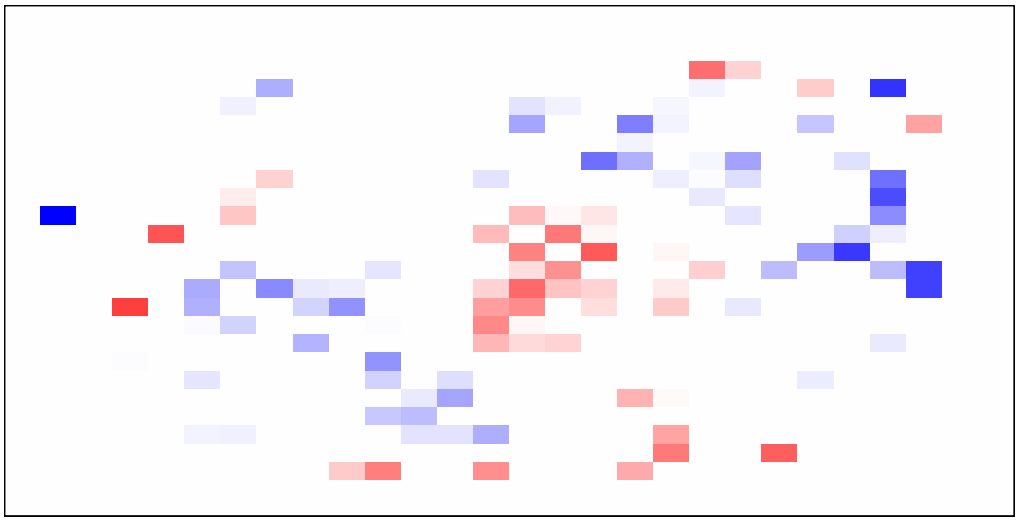



```
par(mfcol = c(1, 1))
```

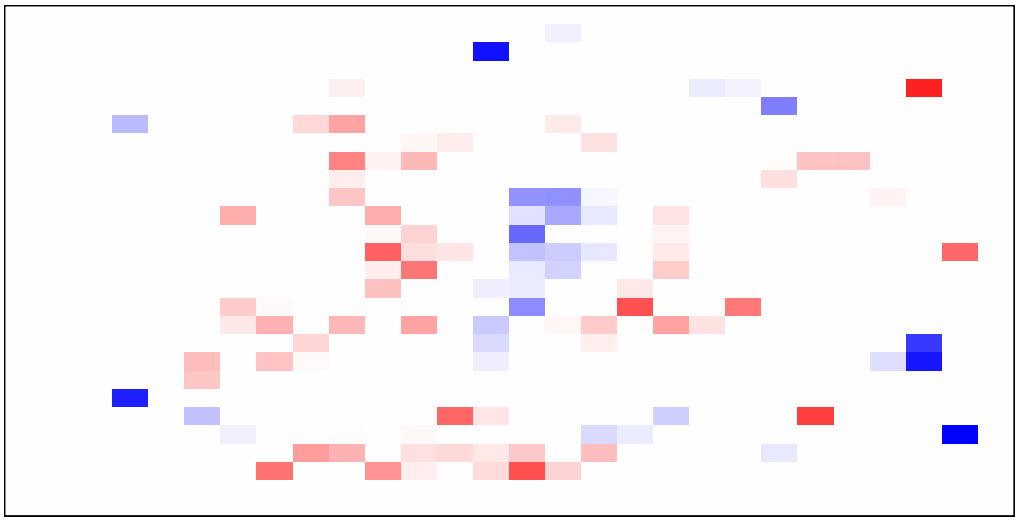
model heatmaps

```
## get coefficients into matrices
model_coef <- coef(init_model, s = init_model$lambda.min) %>%
  lapply(as.matrix) %>%
  lapply(function(x) matrix(x[-1, ], nrow = 28, ncol = 28)) %>%
  ## take sigmoid activation just to help viz
  lapply(function(x) 1 / (1 + exp(-x)) - 0.5)

## plot
# par(mfcol = c(4, 3))
# par(mar = c(0, 0.5, 1, 0.5))

mapply(FUN = plot_mnist_array,
       plt = model_coef,
       main_label = names(model_coef),
       color = TRUE)
```

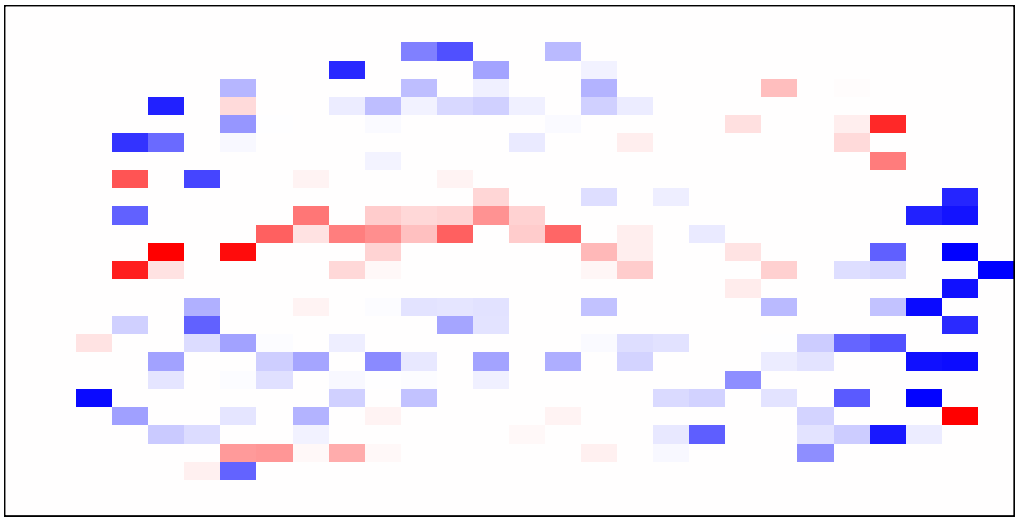
0



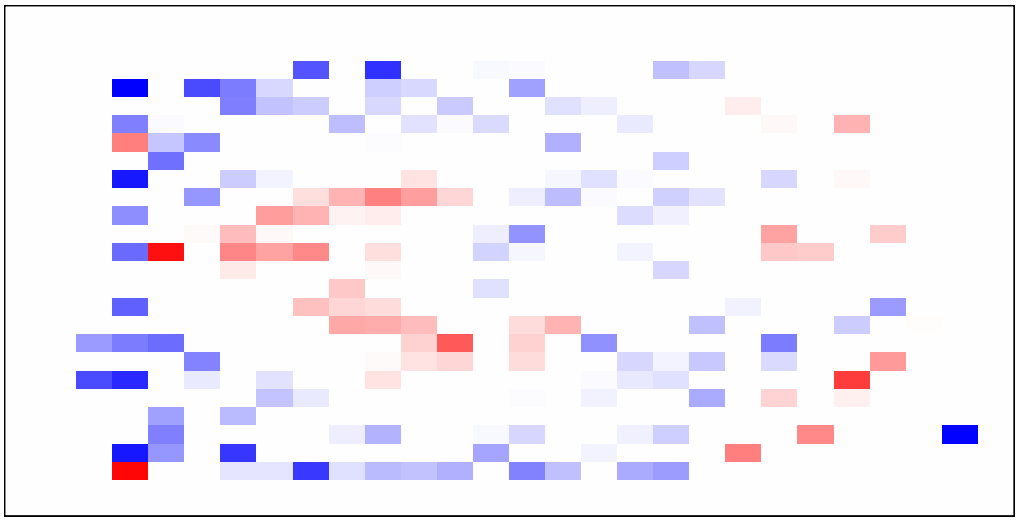
1



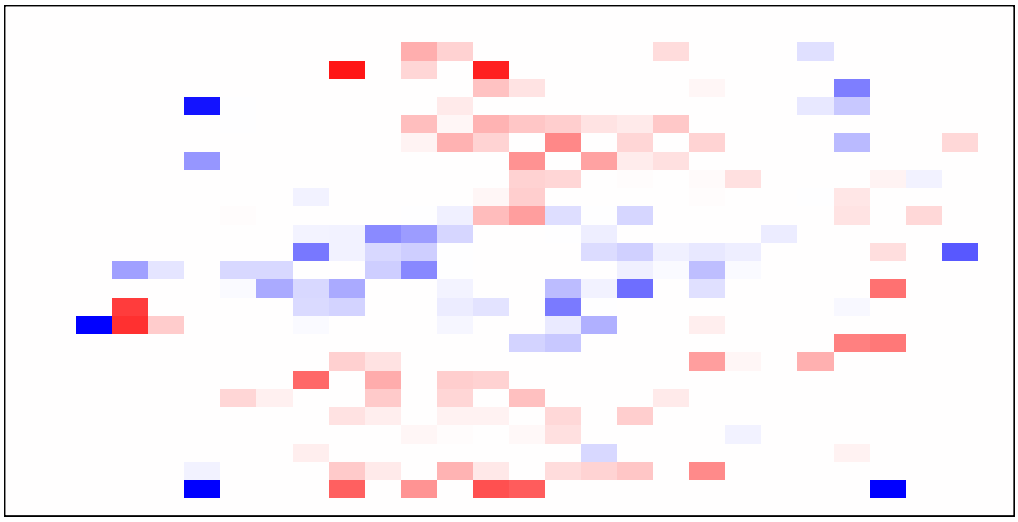
2



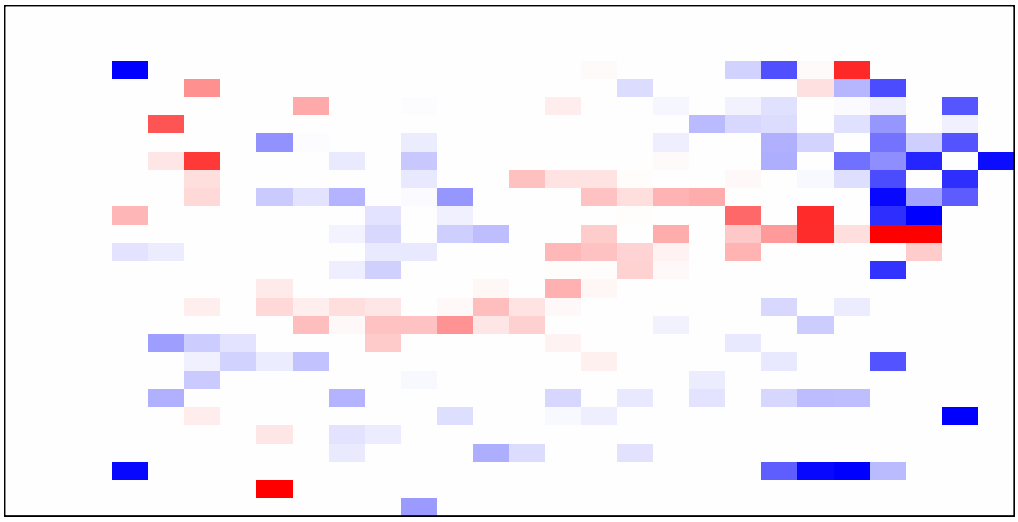
3



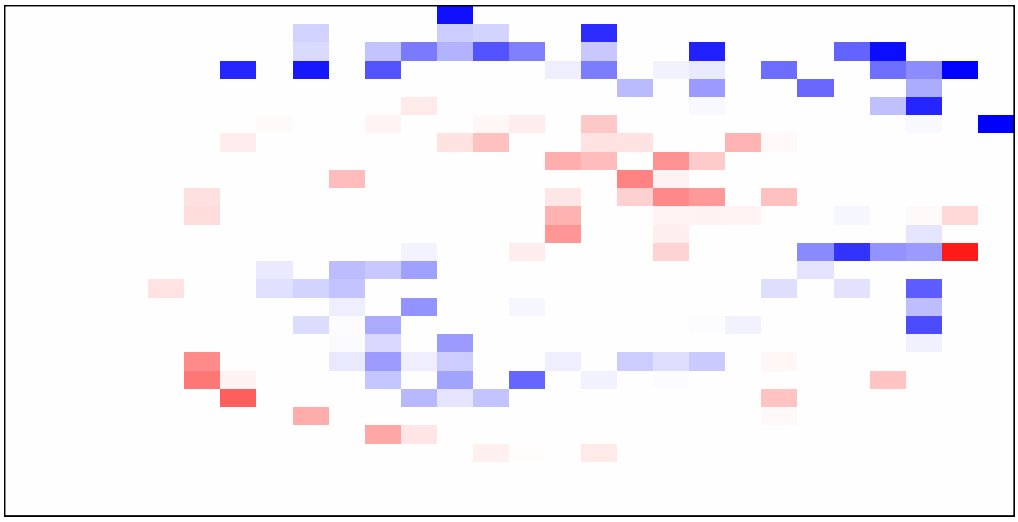
4



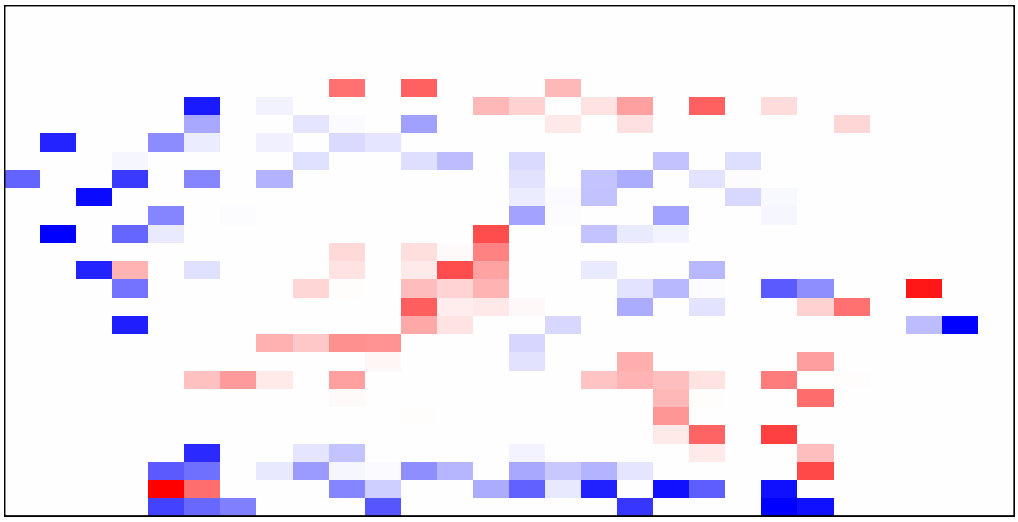
5



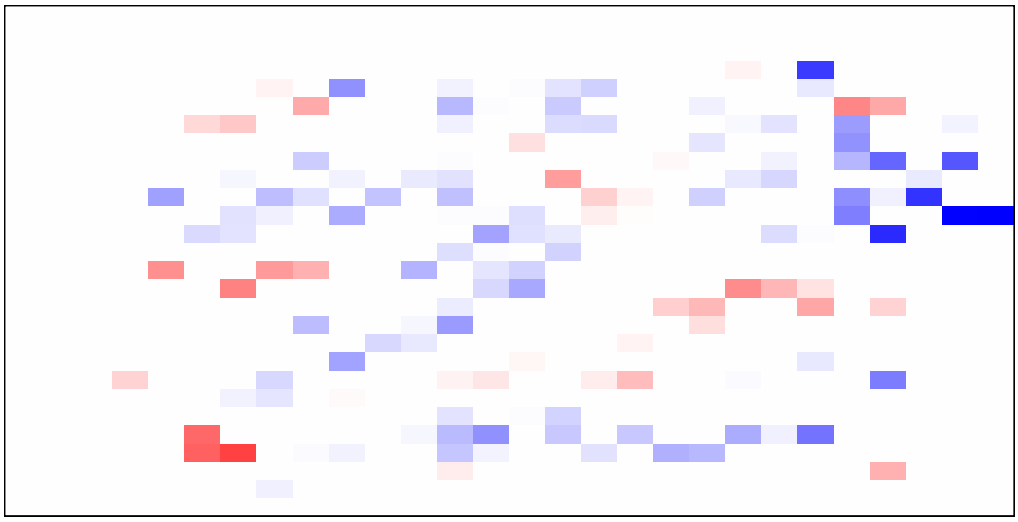
6

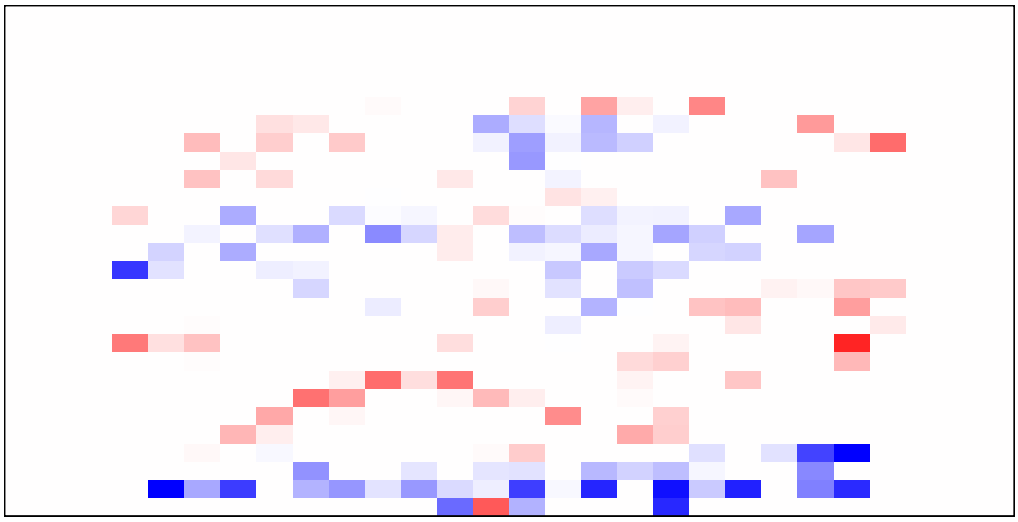


7



8






```
## $`0`
## NULL
##
## $`1`
## NULL
##
## $`2`
## NULL
##
## $`3`
## NULL
##
## $`4`
## NULL
##
## $`5`
## NULL
##
## $`6`
## NULL
##
## $`7`
## NULL
##
## $`8`
## NULL
```

```
##
## $`9`
## NULL
```