Outputs inspection half CIFAR10

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.0.5
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 4.0.5
```

```
library("ggpubr")
```

```
## Warning: package 'ggpubr' was built under R version 4.0.5
```

```
library(LDATS)
```

```
## Warning: package 'LDATS' was built under R version 4.0.5
```

```
library(stringr)
library(reshape2)
```

```
## Warning: package 'reshape2' was built under R version 4.0.3
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
library(reticulate)
```

```
## Warning: package 'reticulate' was built under R version 4.0.5
```

```
np <- import("numpy")

source("utils.R")
```

```
## Warning: package 'hash' was built under R version 4.0.5

## hash-2.2.6.1 provided by Decision Patterns

## Warning: package 'berryFunctions' was built under R version 4.0.5

##
## Attaching package: 'berryFunctions'

## The following object is masked from 'package:dplyr':
##
##     between

## Warning: package 'purrr' was built under R version 4.0.3
```

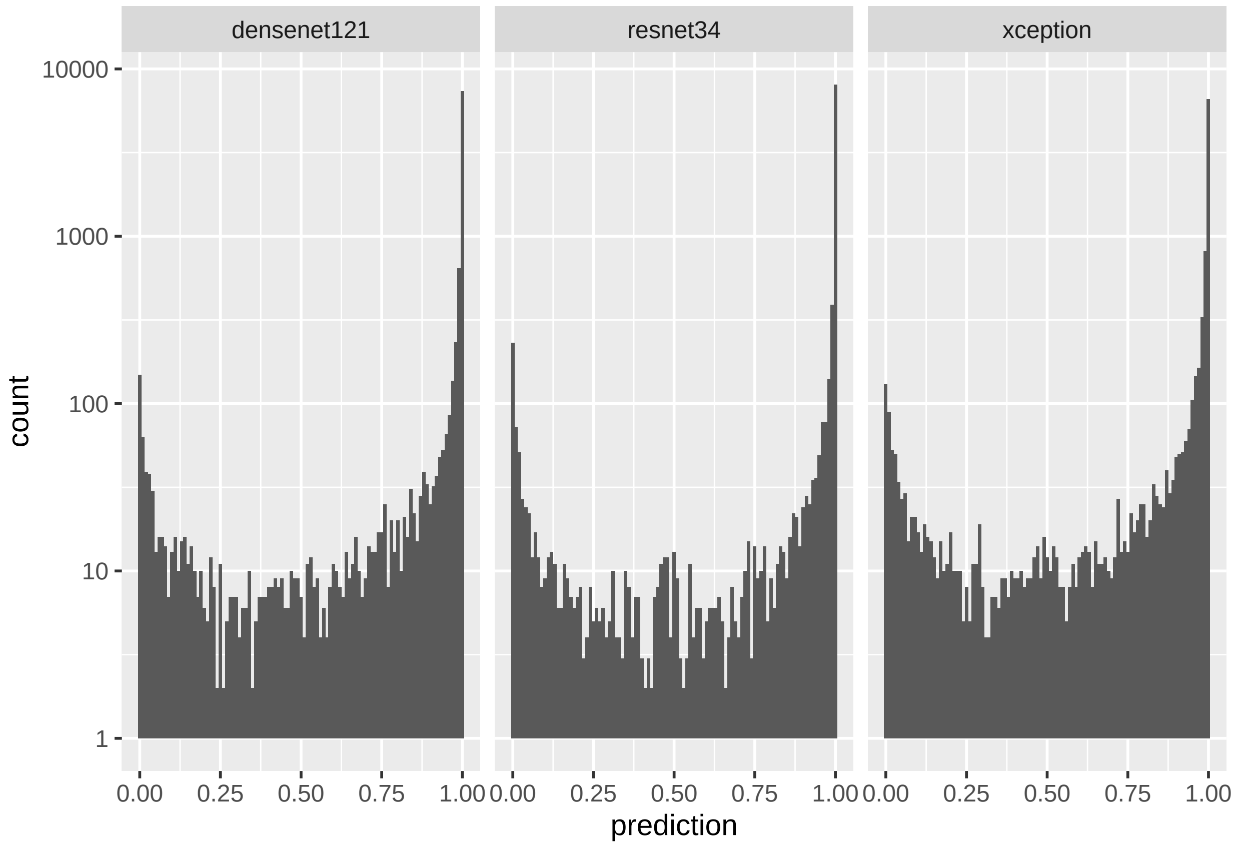
Visualization on CIFAR10. We are using data of three neural networks trained on reduced CIFAR10 training set. Half of the CIFAR10 training set was extracted as a validation set. We then divided both the reduced training set and validation set into 50 disjoint subsets and trained an ensemble on each of them. In this visualization, we are trying to inspect the outputs deeper, mainly to make sense of strange behavior of nll metric for ensemble outputs.

```
base_dir <- "../data/data_train_val_half_c10"
repls <- 0:0
folds <- 0:49
classes <- 10

nets_outputs <- load_network_outputs(base_dir, repls)
ens_outputs <- load_ensemble_outputs(base_dir, repls, folds)
net_results <- read.csv(file.path(base_dir, "net_accuracies.csv"))
ens_results <- read.csv(file.path(base_dir, "ensemble_accuracies.csv"))
```

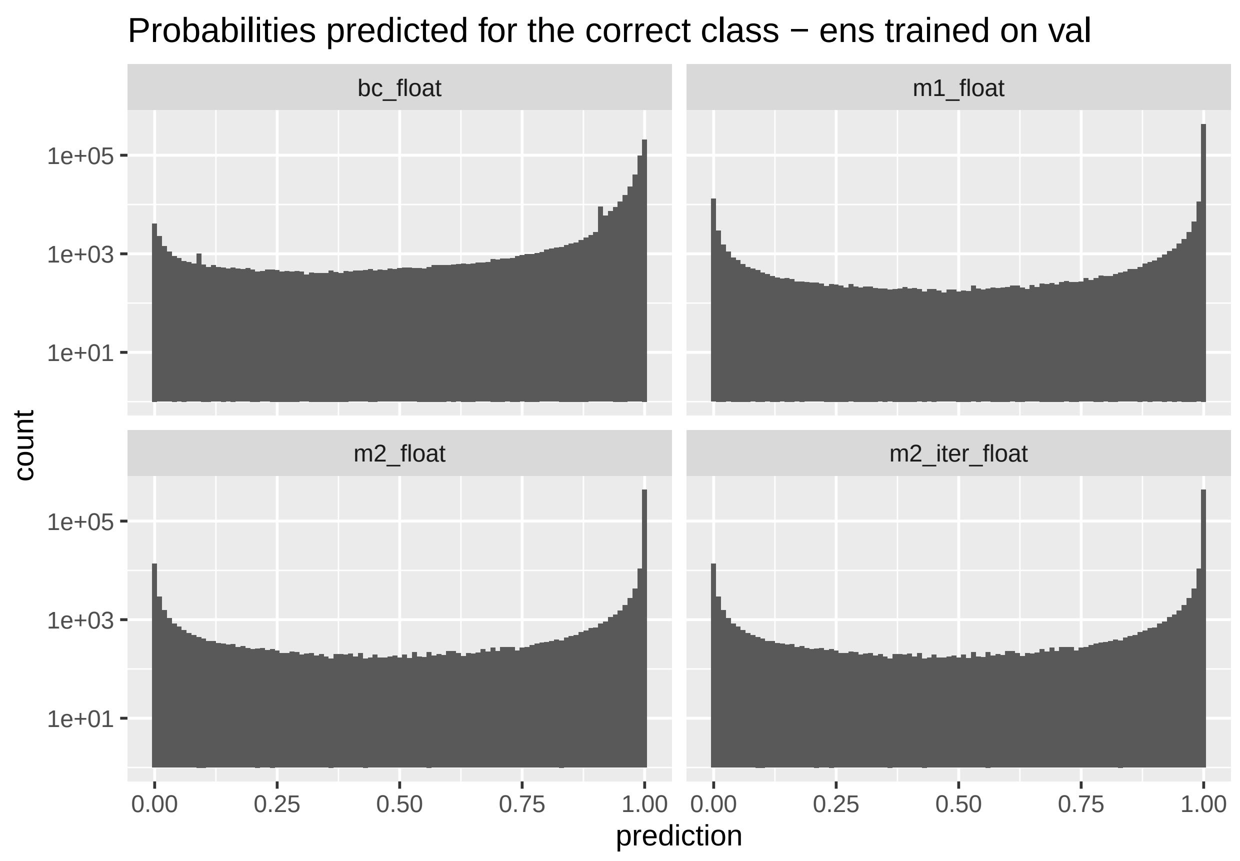
```
preds <- nets_outputs$test_outputs
for (ri in repls + 1)
{
  for (net_i in seq_along(nets_outputs[["networks"]]))
  {
    preds[ri, net_i, ,] <- softmax(preds[ri, net_i, , ])
  }
}
nets_test_cor_probs <- gather(preds, 1 + nets_outputs$test_labels[1, ], 3, 4)
nets_test_cor_probs <- melt(nets_test_cor_probs)
nets_test_cor_probs <- nets_test_cor_probs[, c(-3, -4)]
names(nets_test_cor_probs) <- c("replication", "network", "prediction")
nets_test_cor_probs$network <- as.factor(nets_test_cor_probs$network)
levels(nets_test_cor_probs$network) <- nets_outputs$networks
```

```
nets_cor_preds_histo <- ggplot(data=nets_test_cor_probs) + geom_histogram(mapping=aes(x=prediction), bi
nets_cor_preds_histo
```



```
val_ens_cor_probs <- gather(ens_outputs$val_training, 1 + nets_outputs$test_labels[1, ], 4, 5)
val_ens_cor_probs <- melt(val_ens_cor_probs)
val_ens_cor_probs <- val_ens_cor_probs[, c(-4, -5)]
names(val_ens_cor_probs) <- c("replication", "method", "fold", "prediction")
val_ens_cor_probs$method <- as.factor(val_ens_cor_probs$method)
levels(val_ens_cor_probs$method) <- ens_outputs$methods
```

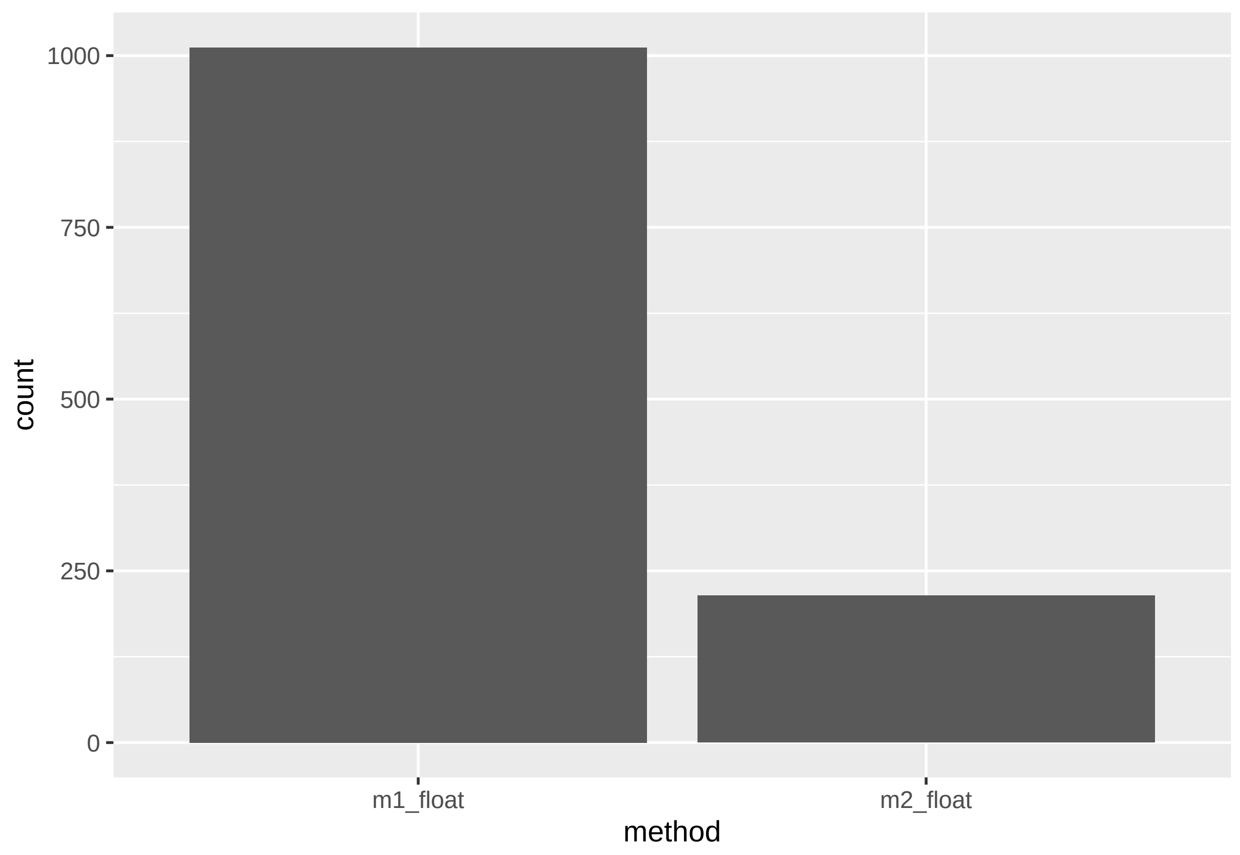
```
val_ens_cor_preds_histo <- ggplot(data=val_ens_cor_probs) + geom_histogram(mapping=aes(x=prediction), bi
val_ens_cor_preds_histo
```



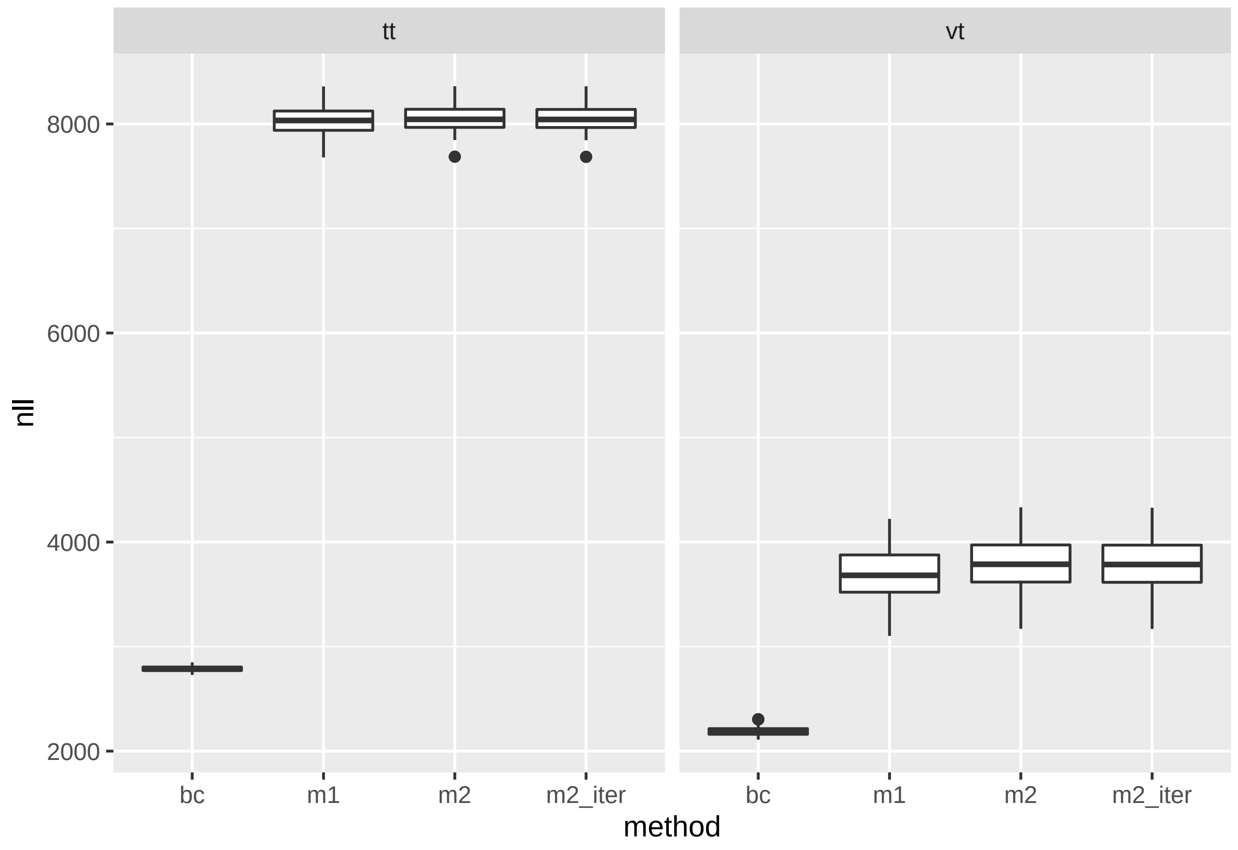
```
val_ens_zero_counts <- ggplot(data=val_ens_cor_probs[val_ens_cor_probs$prediction <= 0, ]) + geom_histo
```

```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```

```
val_ens_zero_counts
```

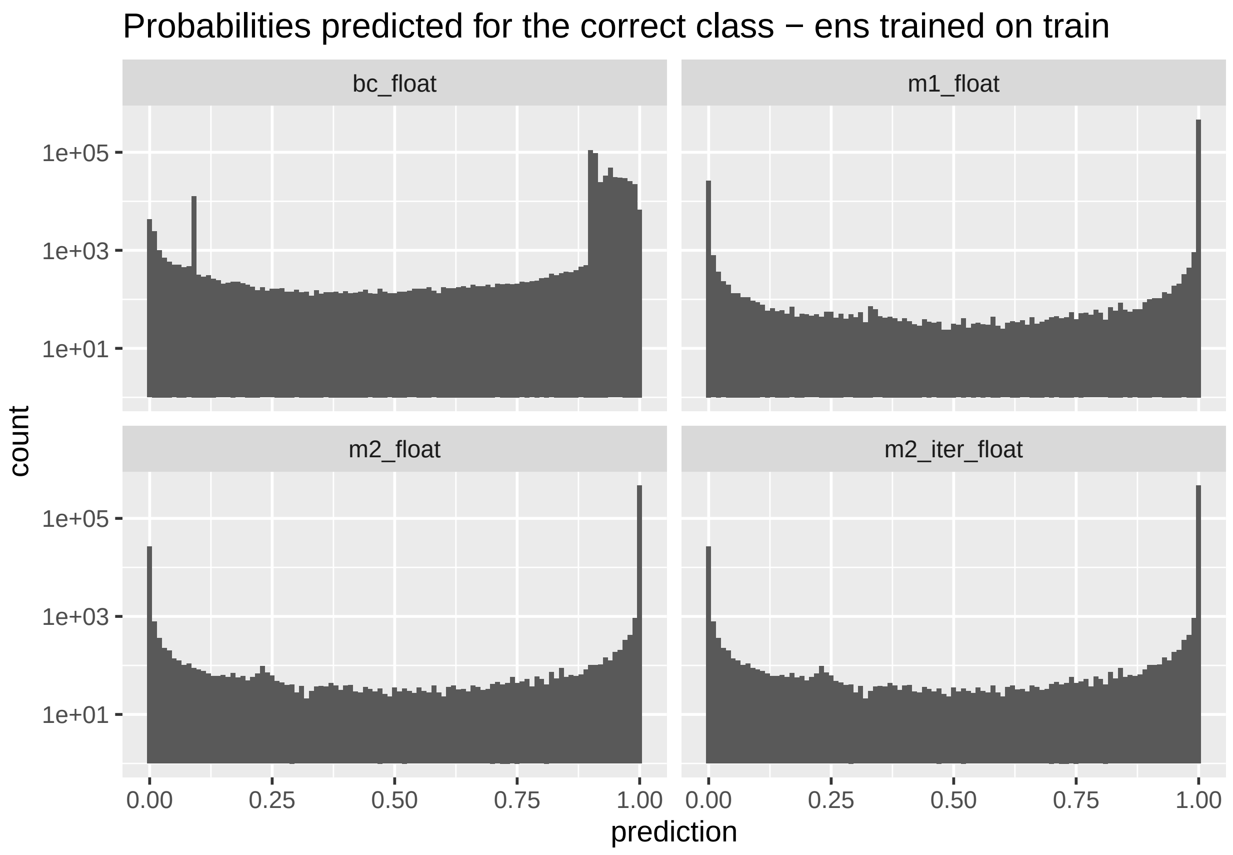


```
val_ens_nll <- ggplot(data=ens_results) + geom_boxplot(mapping=aes(x=method, y=nll)) + facet_wrap(~trai
val_ens_nll
```



```
train_ens_cor_probs <- gather(ens_outputs$train_training, 1 + nets_outputs$test_labels[1, ], 4, 5)
train_ens_cor_probs <- melt(train_ens_cor_probs)
train_ens_cor_probs <- train_ens_cor_probs[, c(-4, -5)]
names(train_ens_cor_probs) <- c("replication", "method", "fold", "prediction")
train_ens_cor_probs$method <- as.factor(train_ens_cor_probs$method)
levels(train_ens_cor_probs$method) <- ens_outputs$methods
```

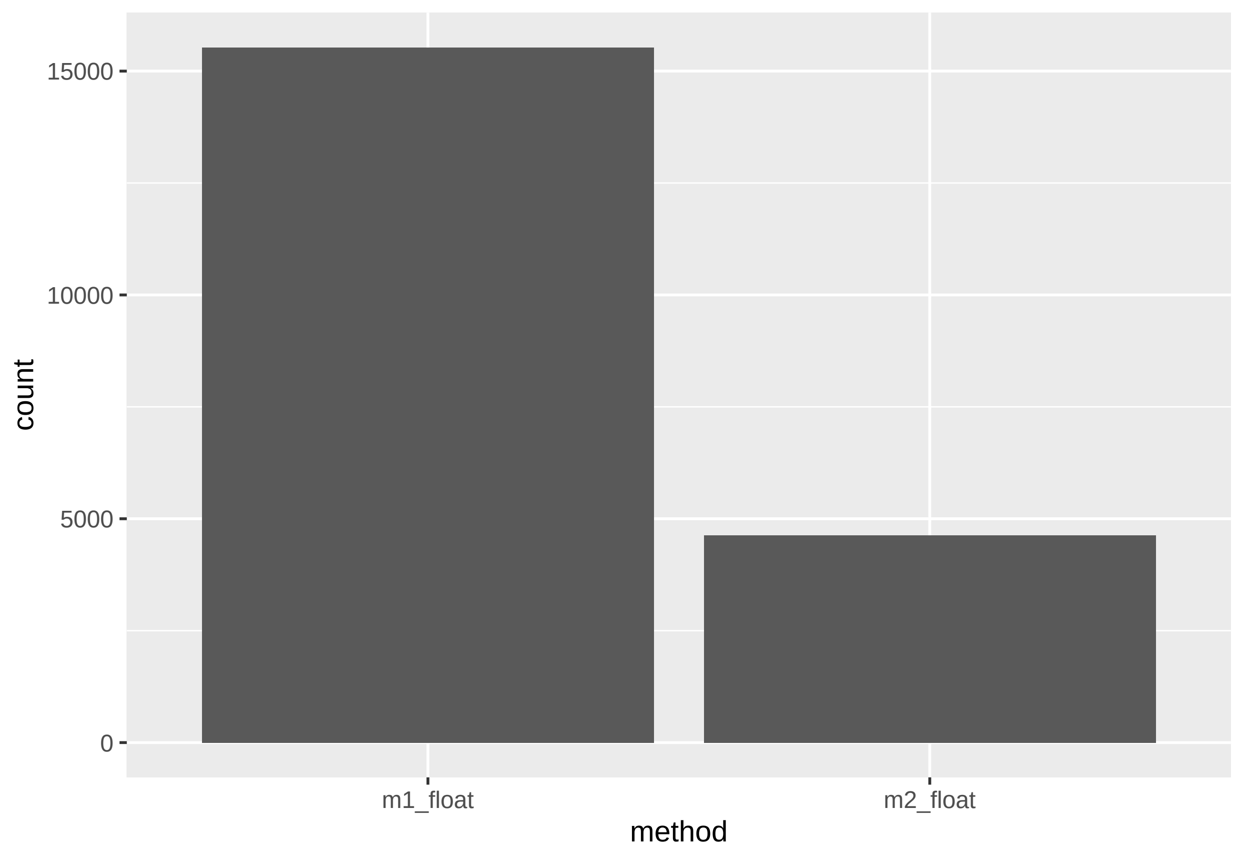
```
train_ens_cor_preds_histo <- ggplot(data=train_ens_cor_probs) + geom_histogram(mapping=aes(x=prediction
train_ens_cor_preds_histo
```



```
train_ens_zero_counts <- ggplot(data=train_ens_cor_probs[train_ens_cor_probs$prediction <= 0, ]) + geom_
```

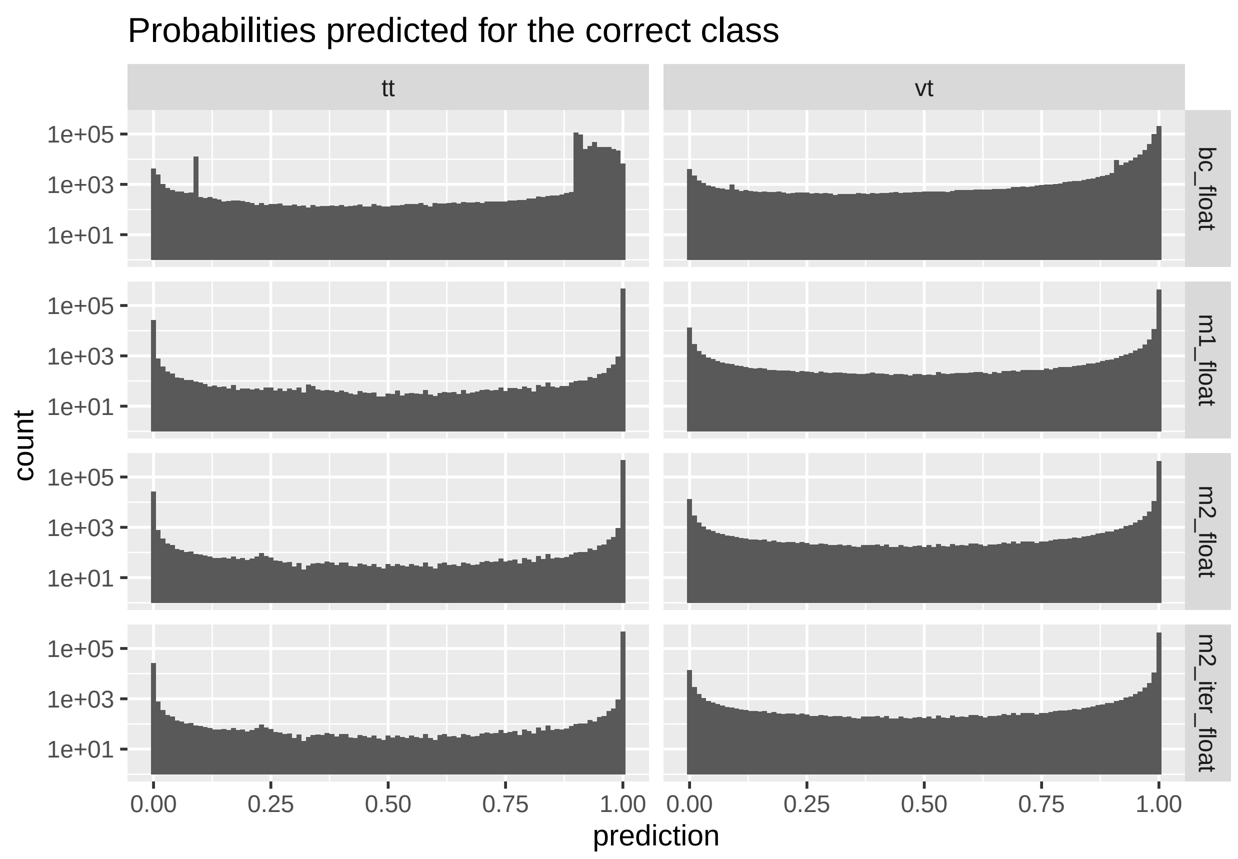
```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```

```
train_ens_zero_counts
```



```
val_ens_cor_probs$train_type <- "vt"
train_ens_cor_probs$train_type <- "tt"
ens_cor_probs <- rbind(val_ens_cor_probs, train_ens_cor_probs)
```

```
ens_cor_preds_histo <- ggplot(data=ens_cor_probs) + geom_histogram(mapping=aes(x=prediction), binwidth=
ens_cor_preds_histo
```



Here we observe similar results to those in vasualizations_ensemble_outputs_CIF10.

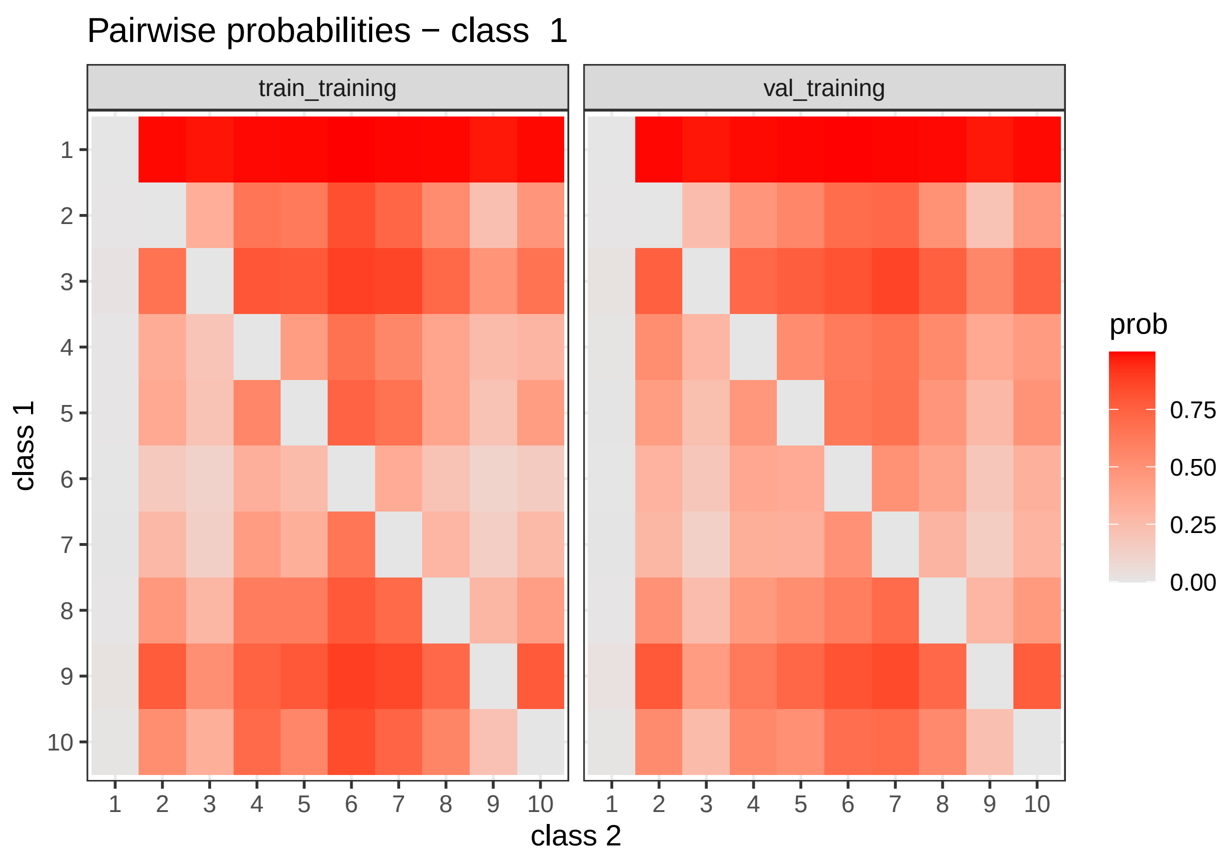
```
aggreg_Rs <- load_class_averaged_R_matrices(base_dir,  nets_outputs$test_labels[1, ], repls, folds=fold
```

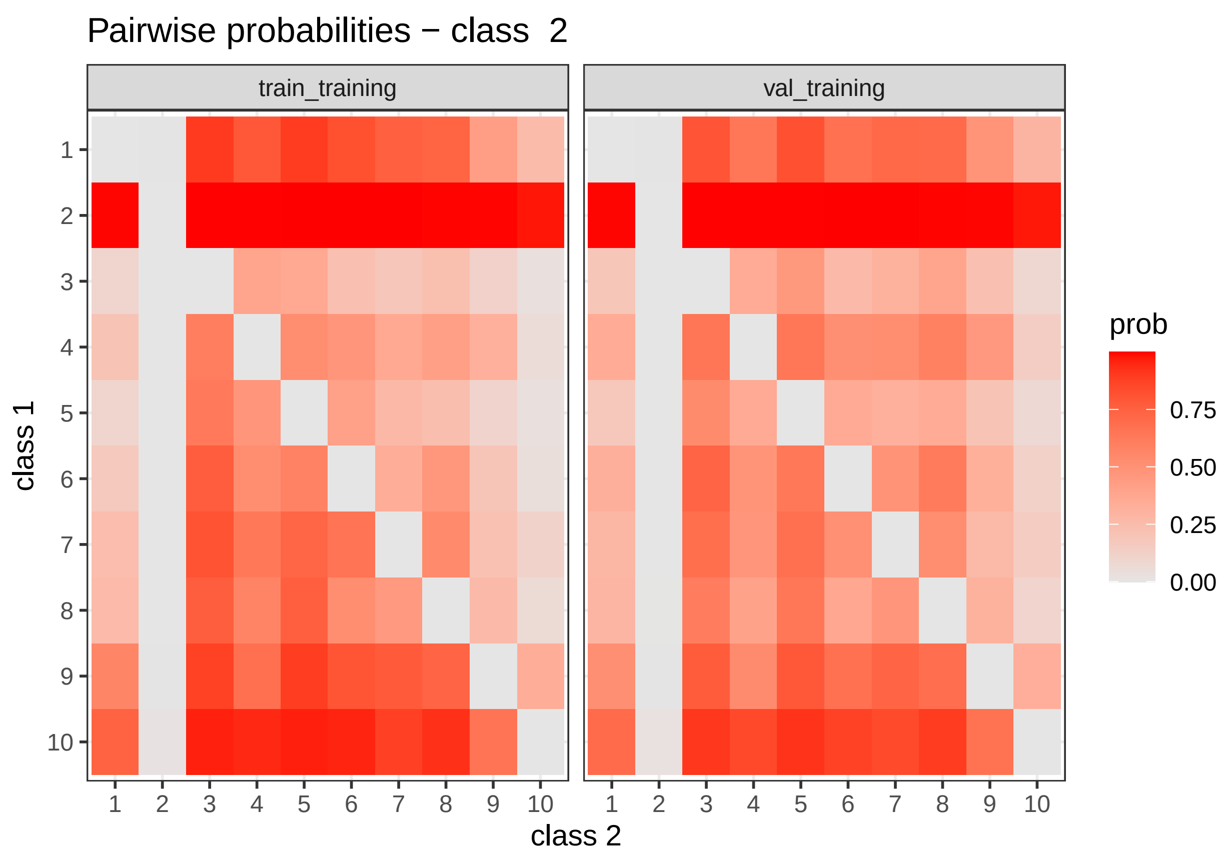
```
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
## `summarise()` has grouped output by 'class1', 'class2'. You can override using the `.groups` argumen
```

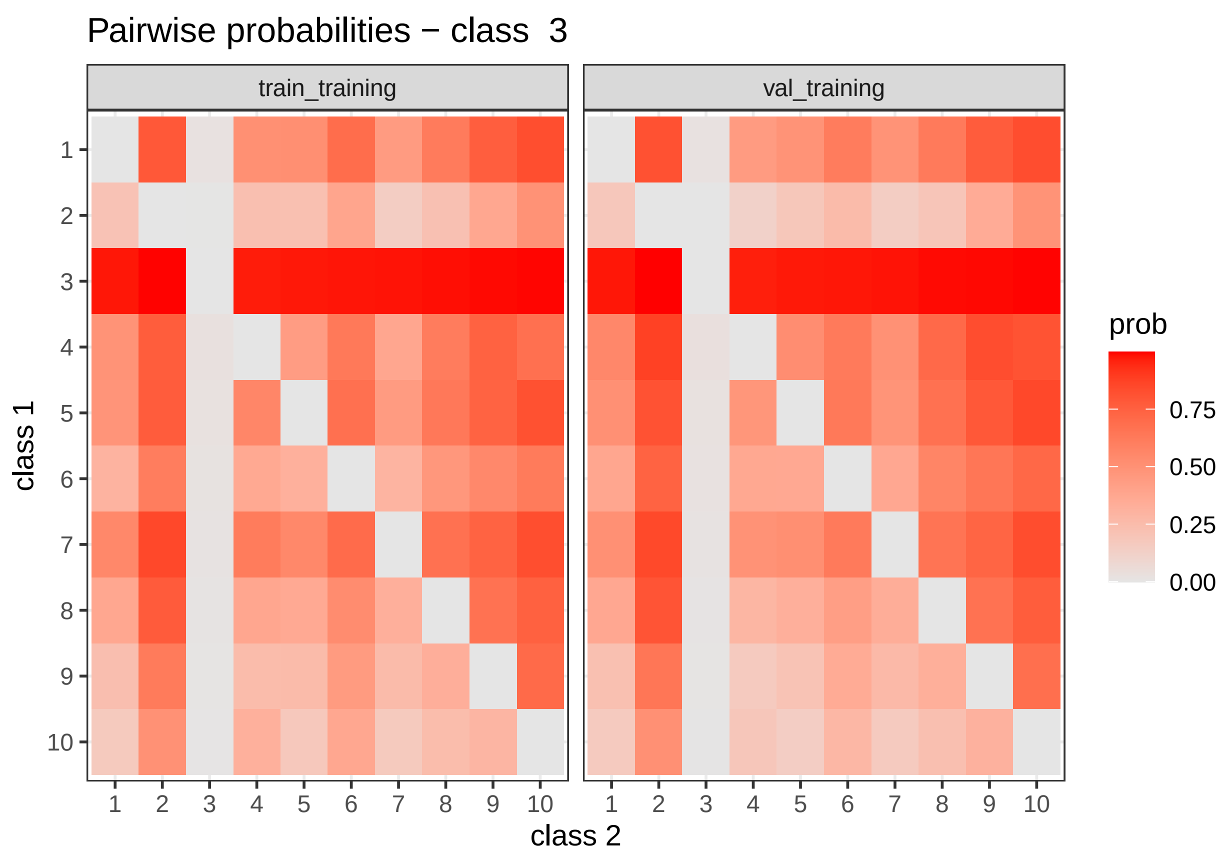
```
## `summarise()` has grouped output by 'precision', 'train_type', 'class1', 'class2'. You can override
```

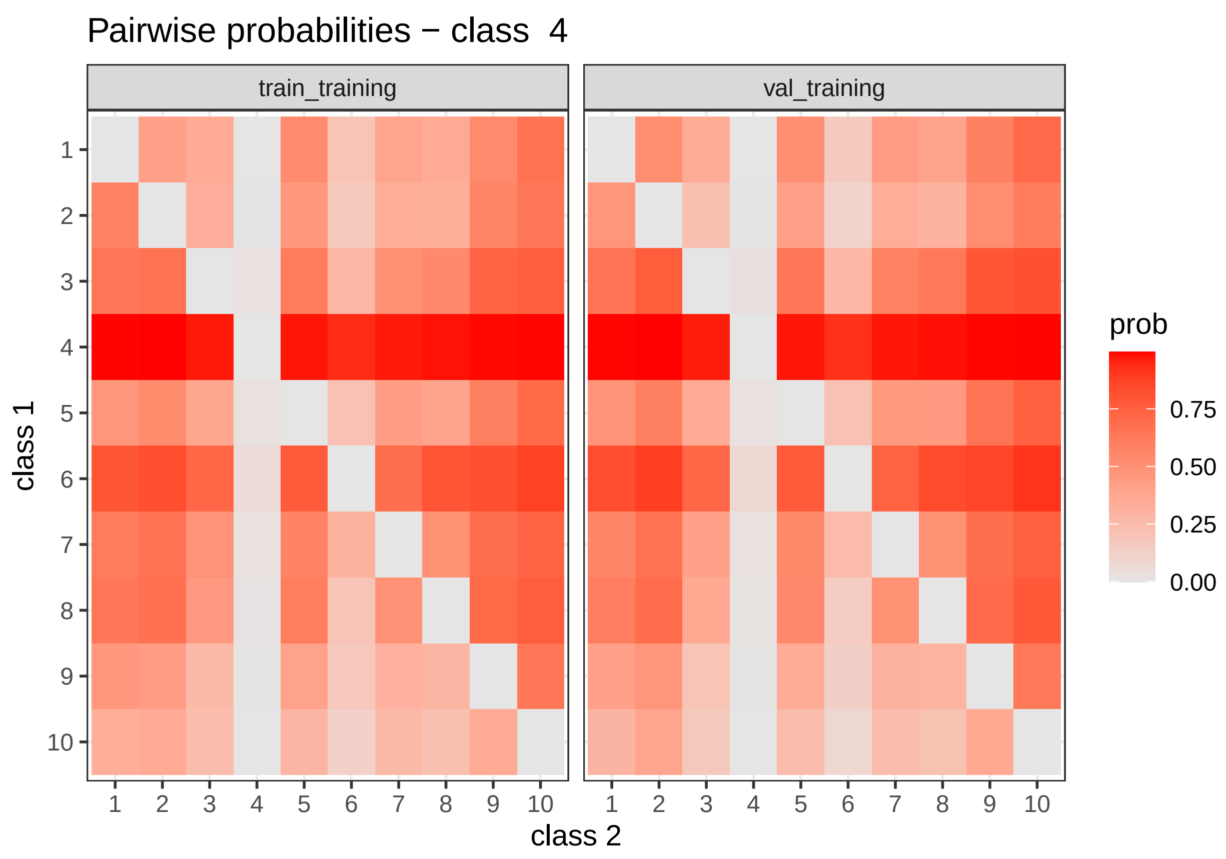
```r
for (cls in 1:classes)
{
  cur_class_Rs <- aggreg_Rs %>% filter(class == cls)
  plot_cls <-  ggplot(cur_class_Rs, aes(x = class2, y = class1)) +
    geom_raster(aes(fill=prob)) +
    facet_wrap(~train_type) +
    scale_fill_gradient(low="grey90", high="red") +
    scale_y_discrete(limits=rev) +
    labs(x="class 2", y="class 1", title=paste("Pairwise probabilities - class ", cls)) +
    theme_bw()

  print(plot_cls)
}
```

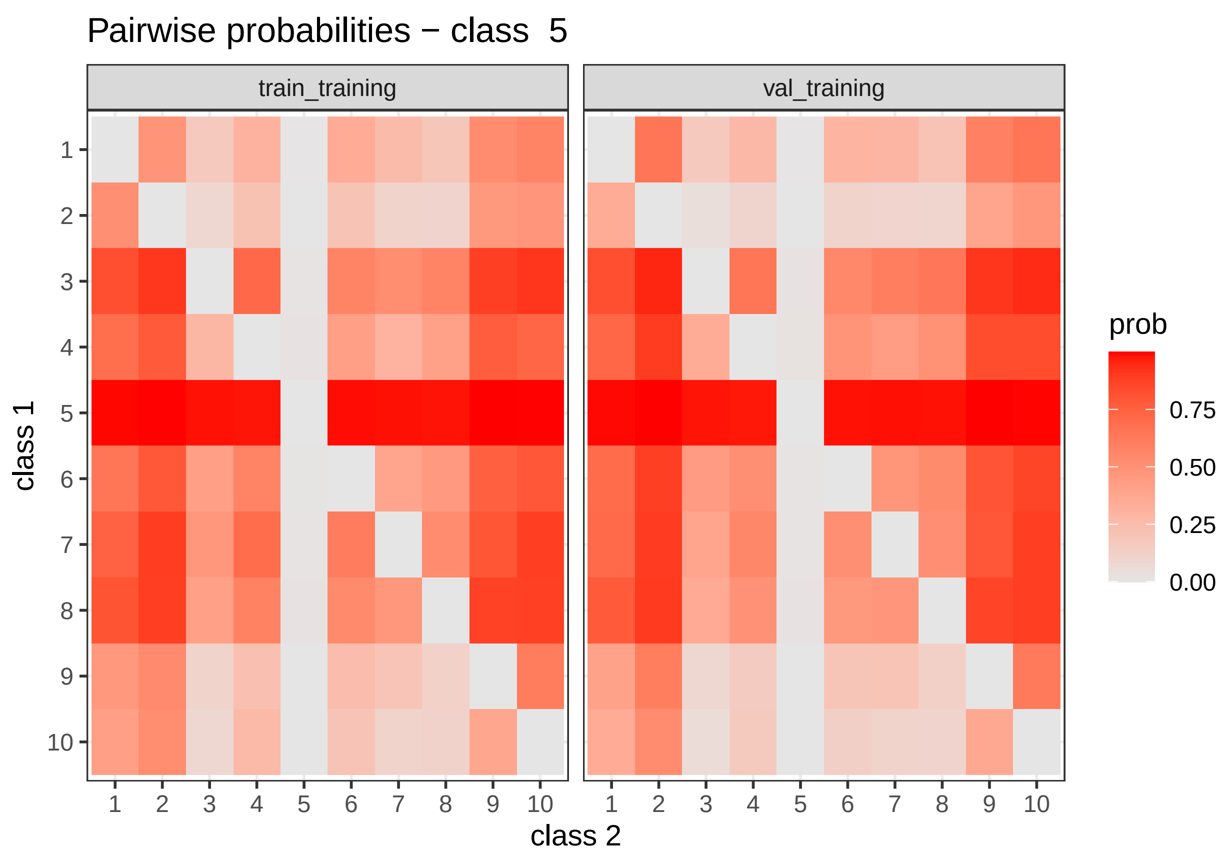

Pairwise probabilities − class 1
train_training
val_training
class 1
class 2
1
2
3
4
5
6
7
8
9
10
prob
0.75
0.50
0.25
0.00

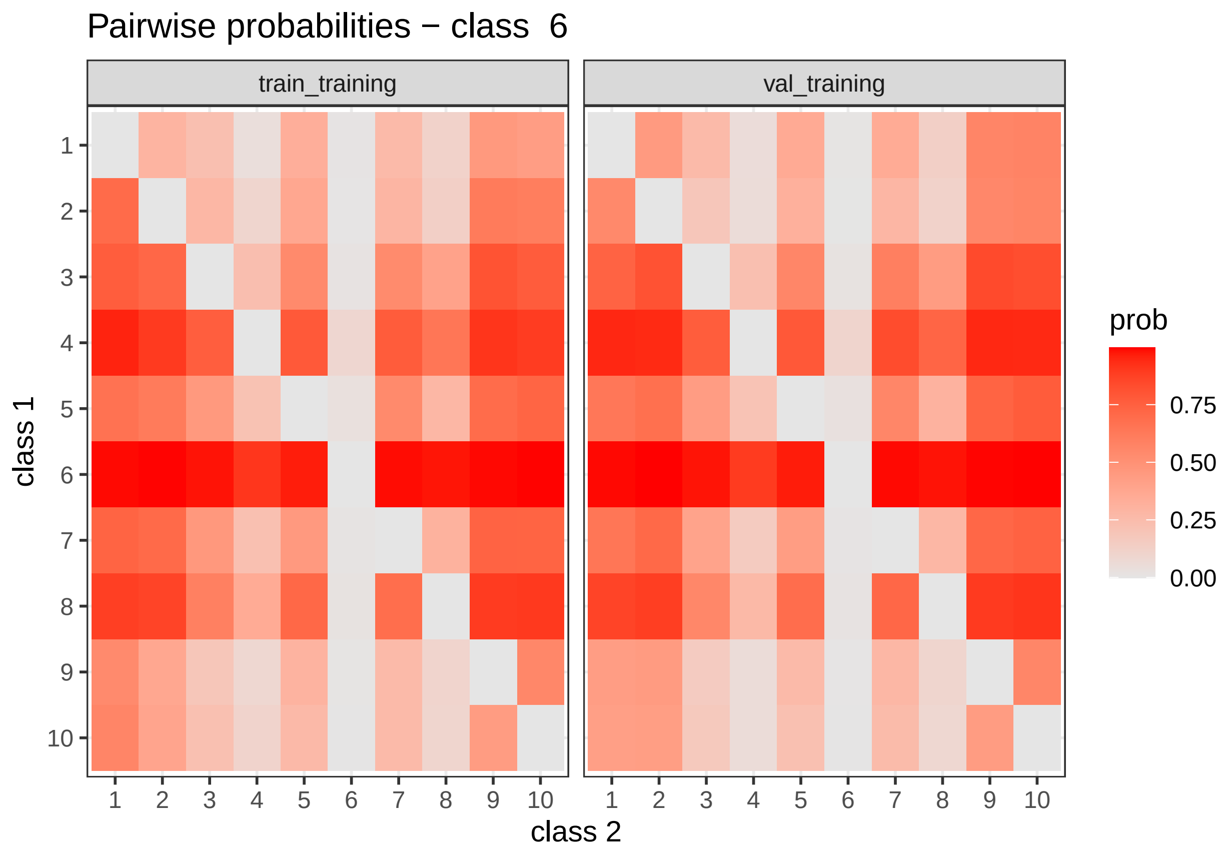

Pairwise probabilities − class 2
train_training
val_training
class 1
class 2
prob
0.75
0.50
0.25
0.00
1
2
3
4
5
6
7
8
9
10

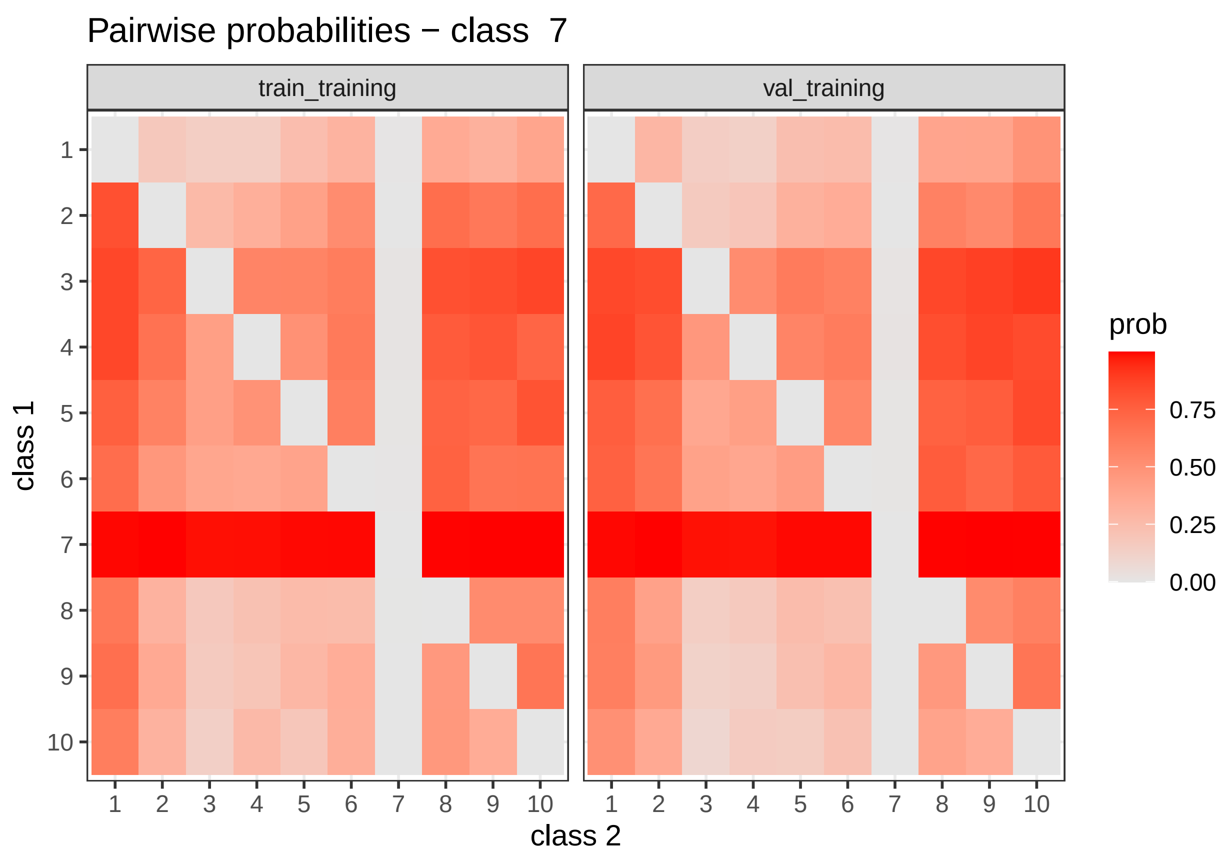

Pairwise probabilities − class 3
train_training
val_training
class 1
class 2
prob
0.75
0.50
0.25
0.00
1
2
3
4
5
6
7
8
9
10

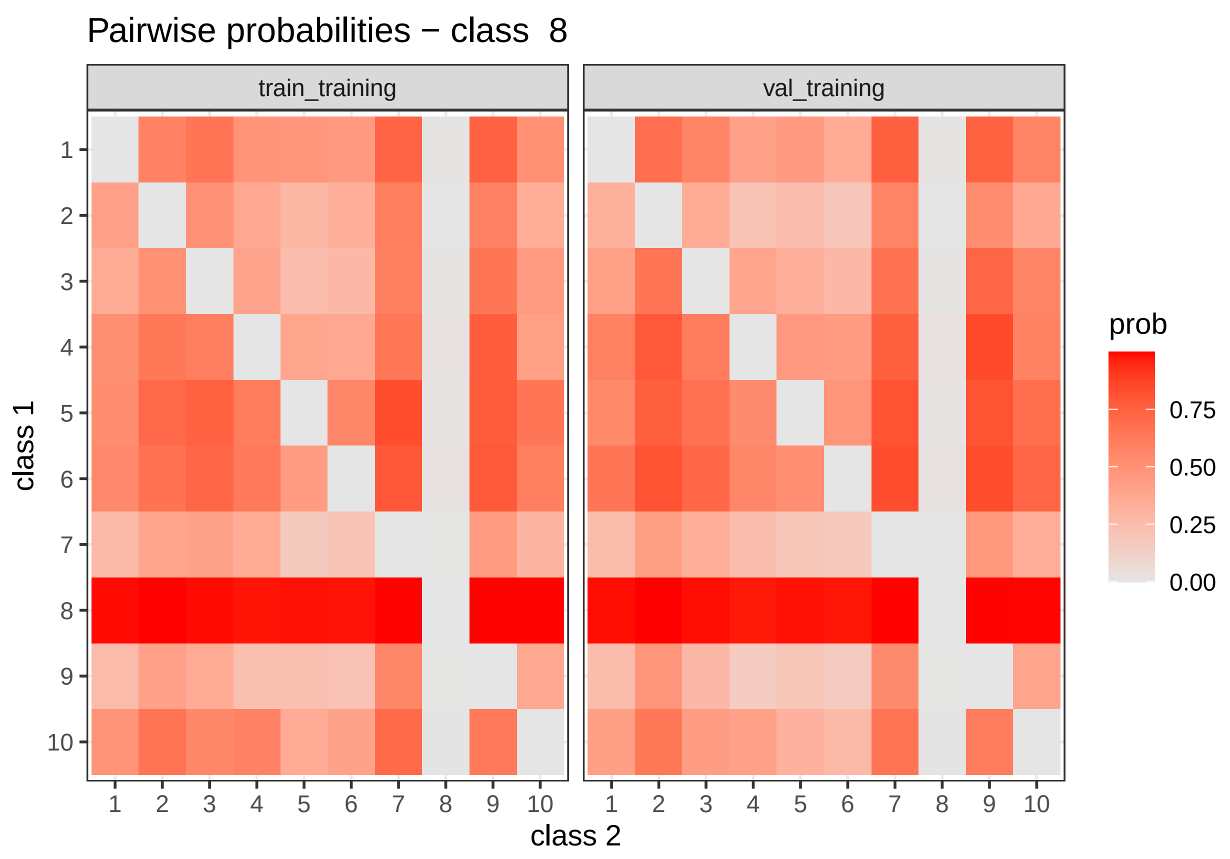


Pairwise probabilities − class 4
train_training
val_training
class 1
class 2
prob
0.75
0.50
0.25
0.00
1
2
3
4
5
6
7
8
9
10

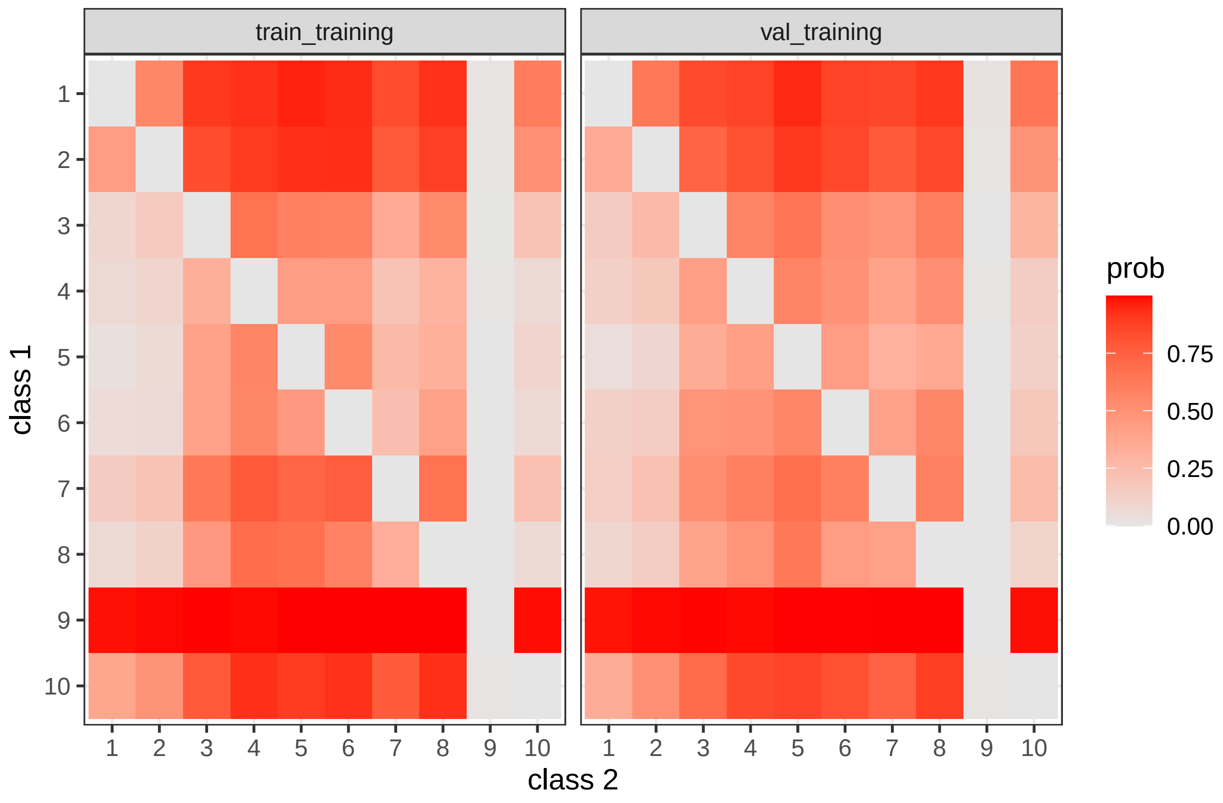

Pairwise probabilities − class 5
train_training
val_training
class 1
class 2
prob
0.75
0.50
0.25
0.00
1
2
3
4
5
6
7
8
9
10

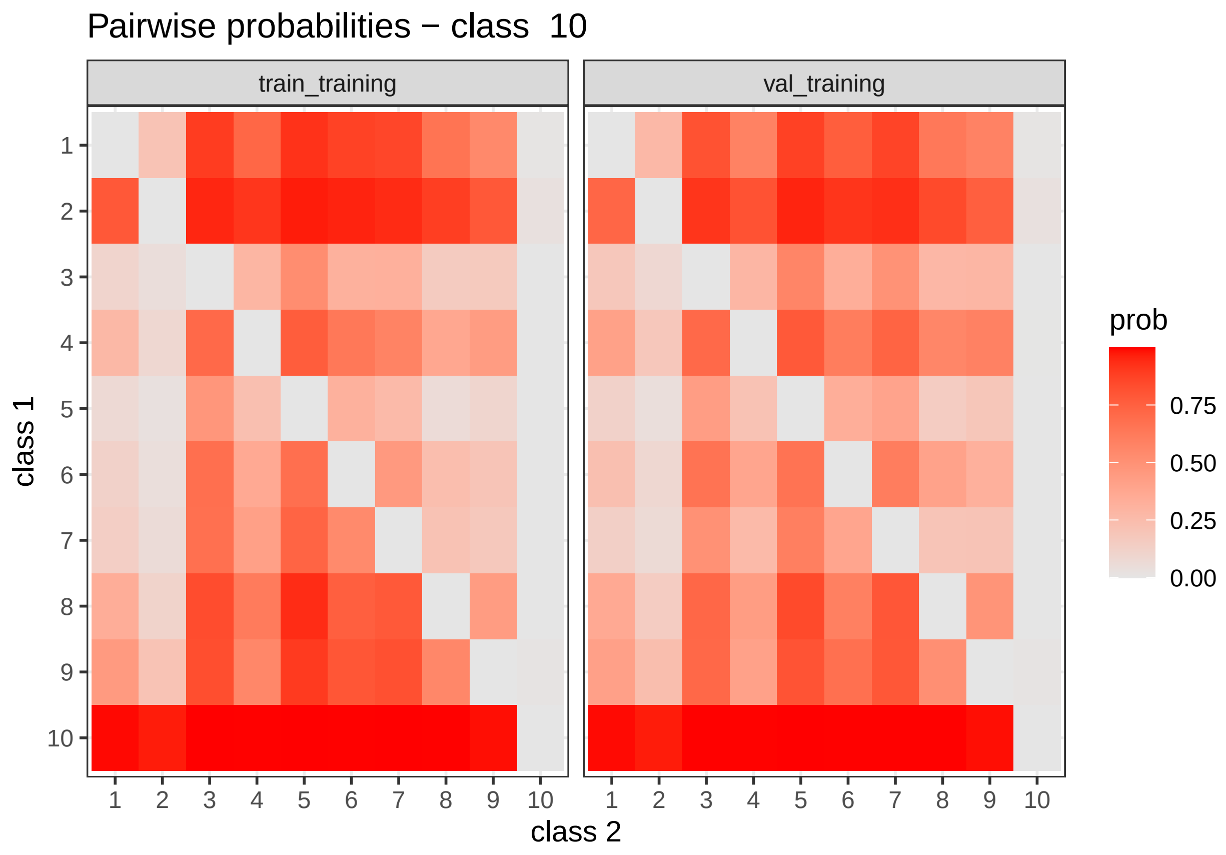

Pairwise probabilities − class 6
train_training
val_training
class 1
class 2
prob
0.75
0.50
0.25
0.00
1
2
3
4
5
6
7
8
9
10


Pairwise probabilities − class 7
train_training
val_training
class 1
class 2
1
2
3
4
5
6
7
8
9
10
prob
0.75
0.50
0.25
0.00


Pairwise probabilities − class 8
train_training
val_training
class 1
class 2
prob
0.75
0.50
0.25
0.00
1
2
3
4
5
6
7
8
9
10

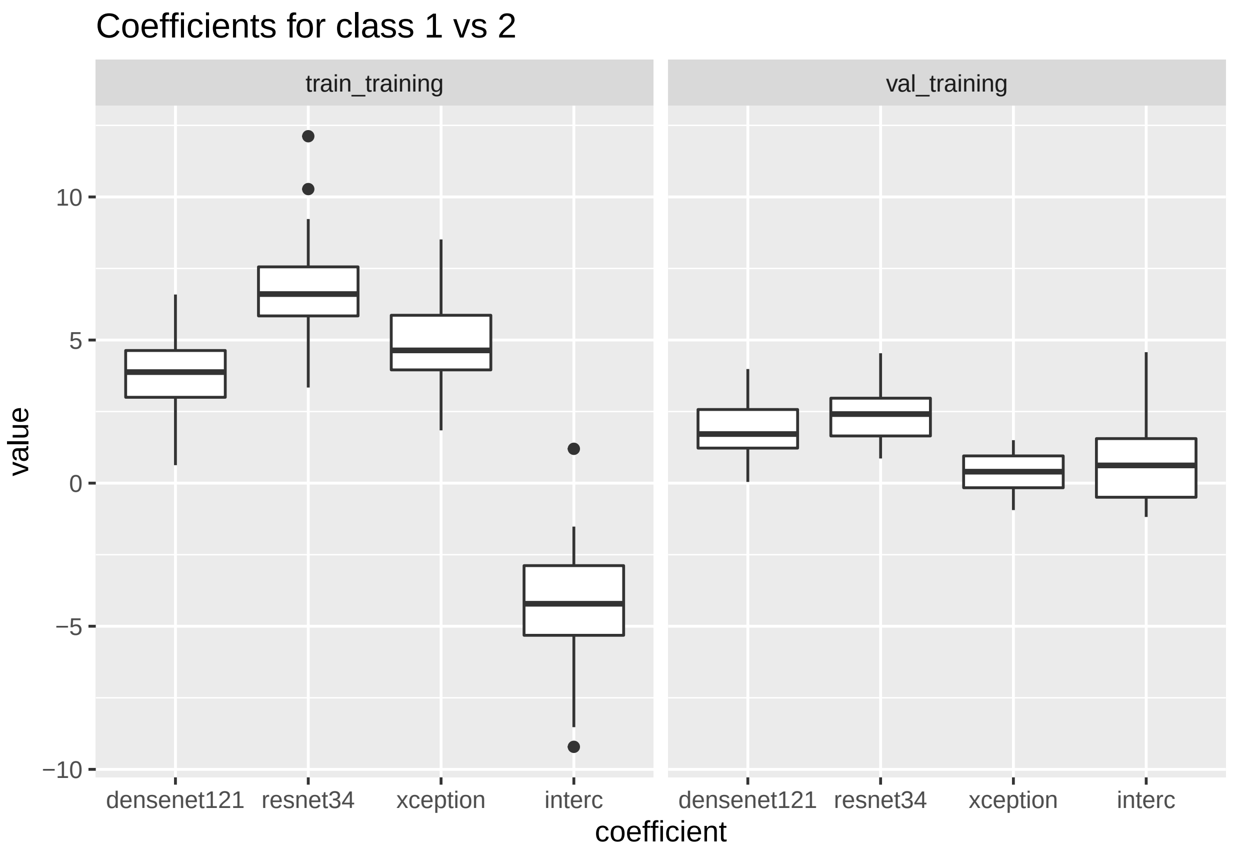
Pairwise probabilities − class 9

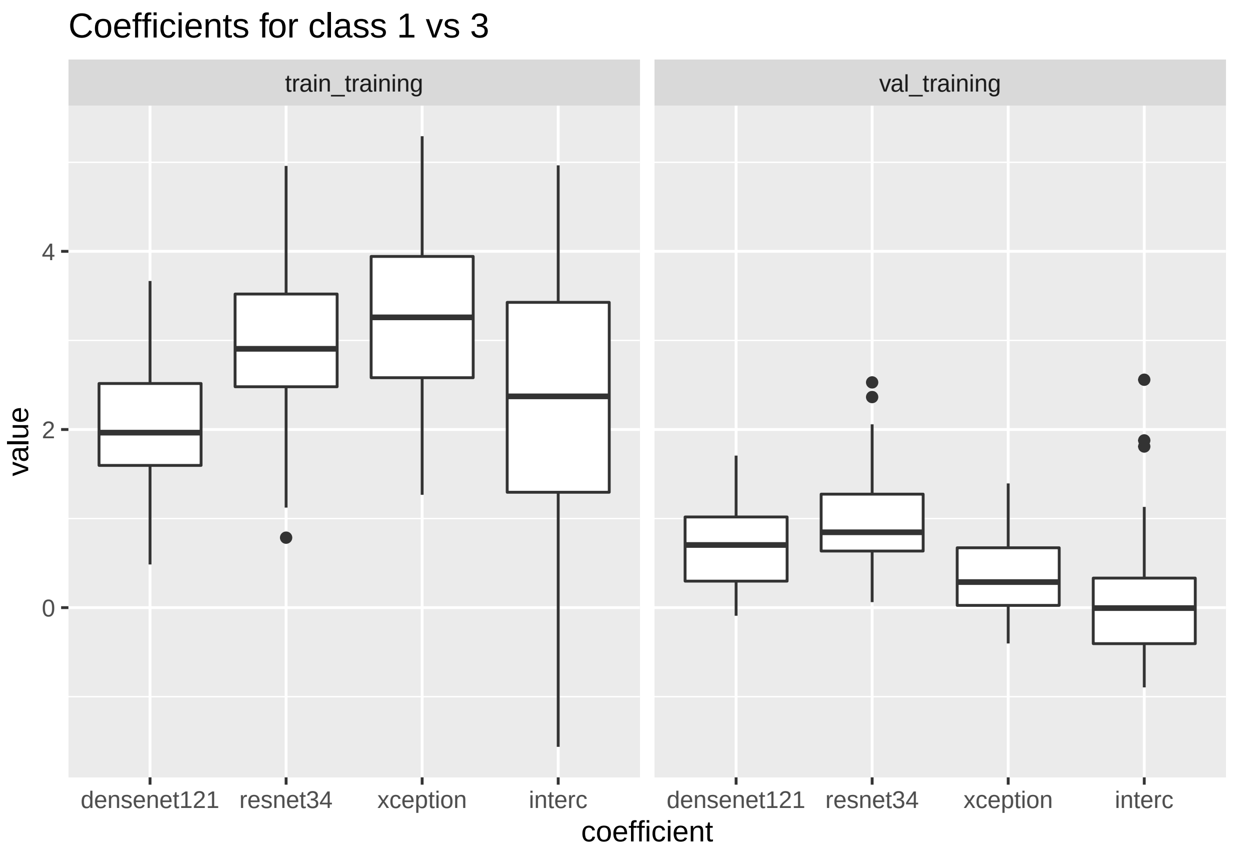


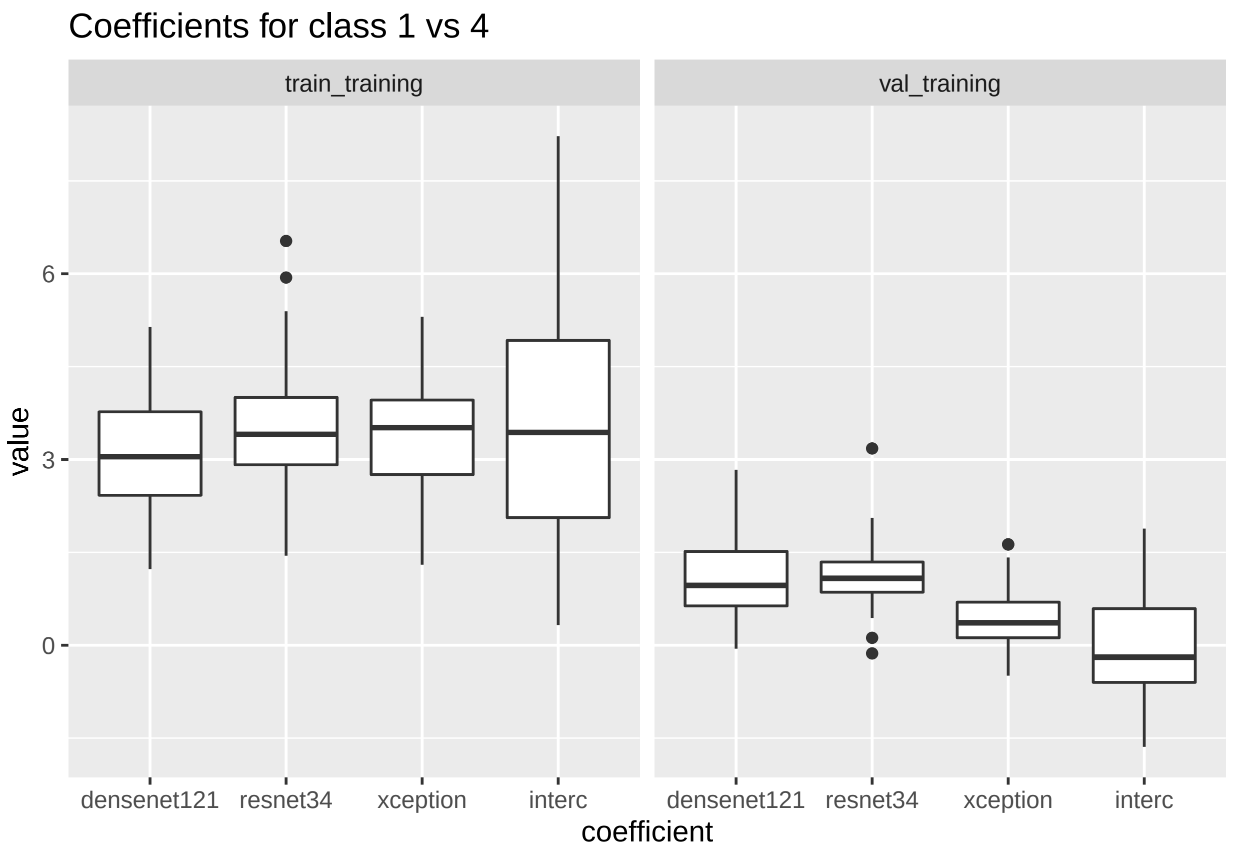


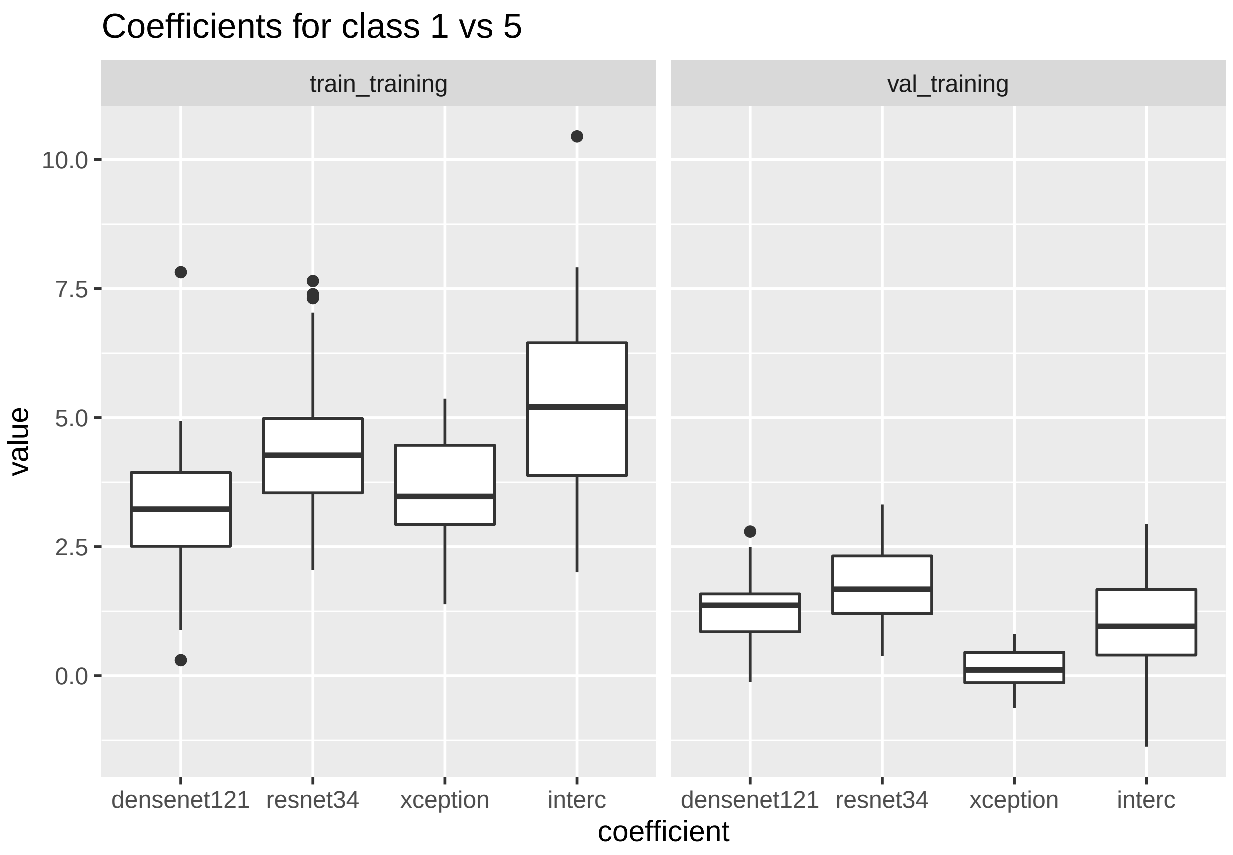
```
lda_coefs <- load_lda_coefs(base_dir, repls, folds)
```

```
for (cl1 in 1:(classes - 1))
{
  for (cl2 in (cl1 + 1):classes)
  {
    cur_plt <- lda_coefs %>% filter(class1 == cl1 & class2 == cl2) %>% ggplot() + geom_boxplot(aes(x=co
      facet_wrap(~train_type) + ggtitle(paste("Coefficients for class", cl1, "vs", cl2))
    print(cur_plt)
  }
}
```

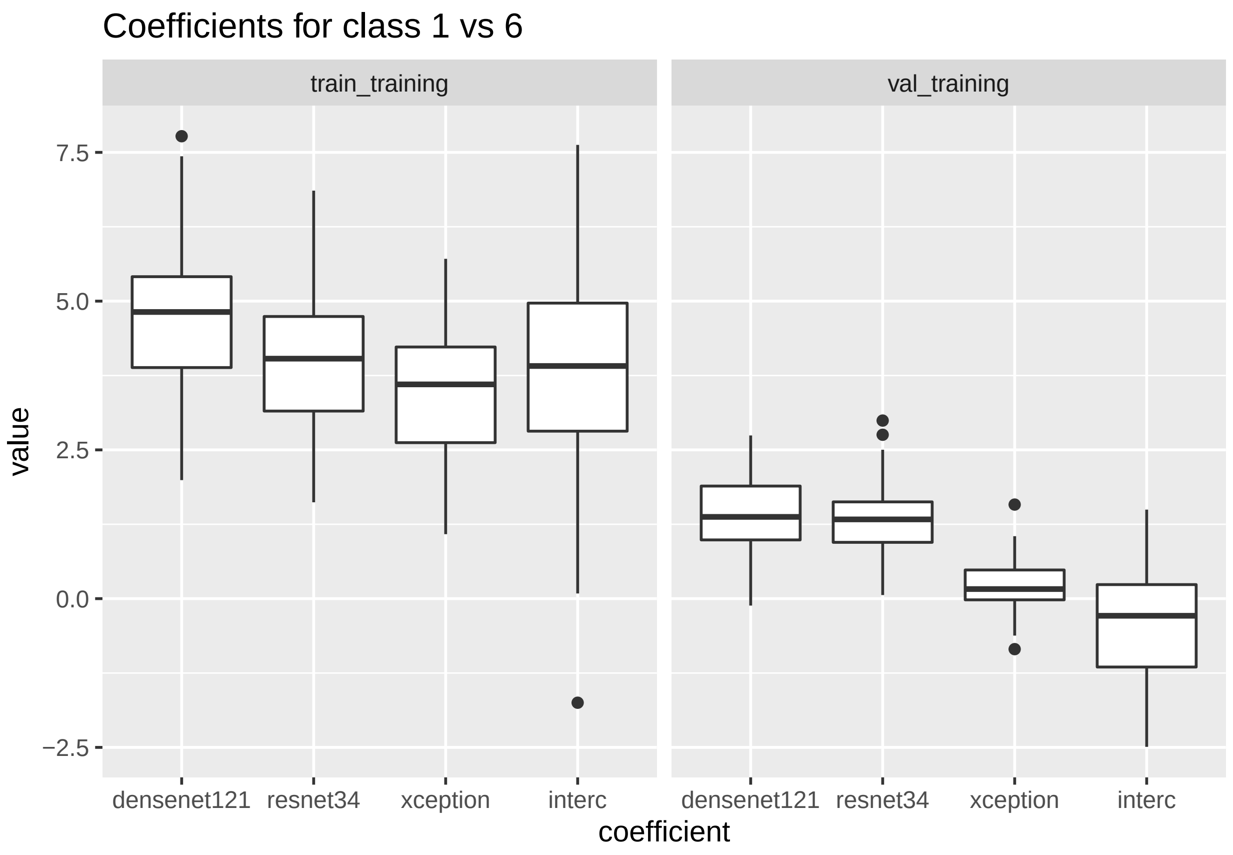

Coefficients for class 1 vs 2
train_training
val_training
value
10
5
0
−5
−10
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

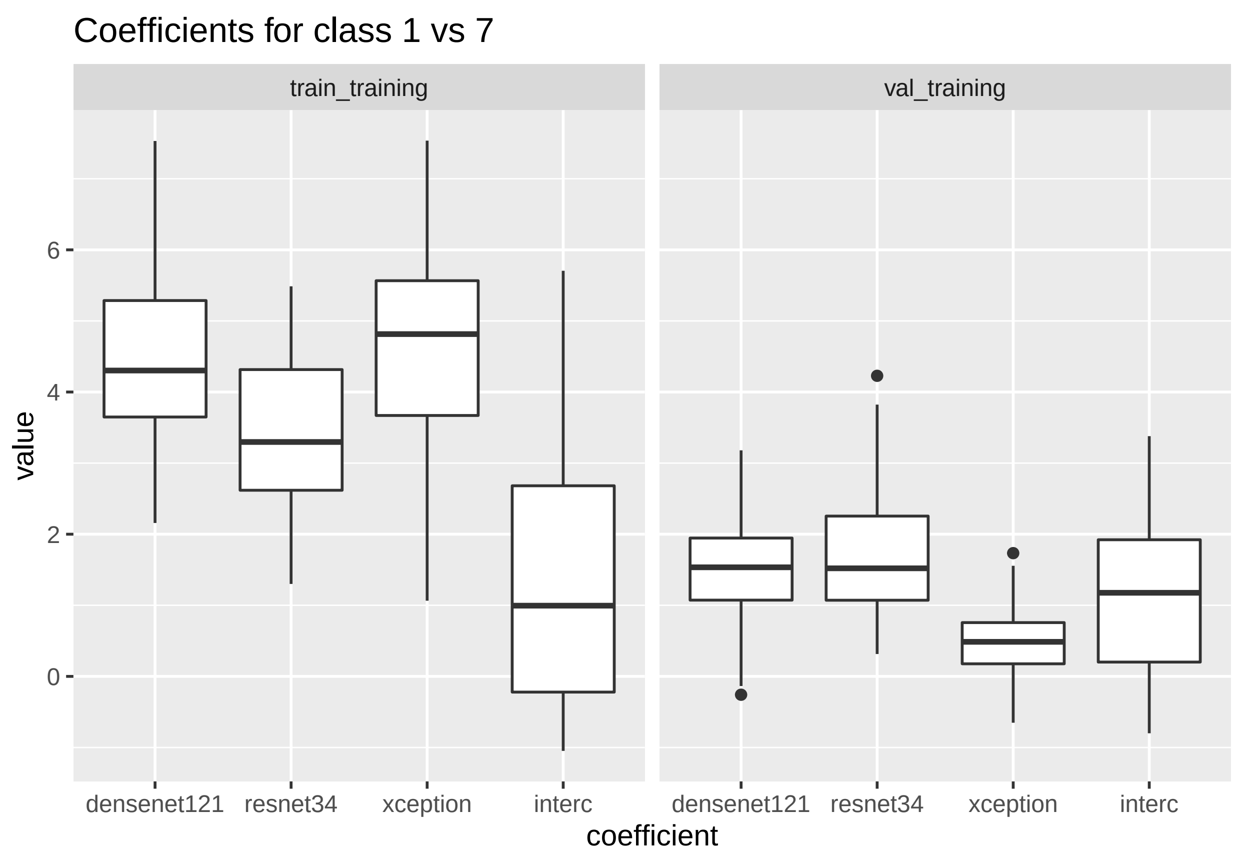

Coefficients for class 1 vs 3
train_training
val_training
value
4
2
0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

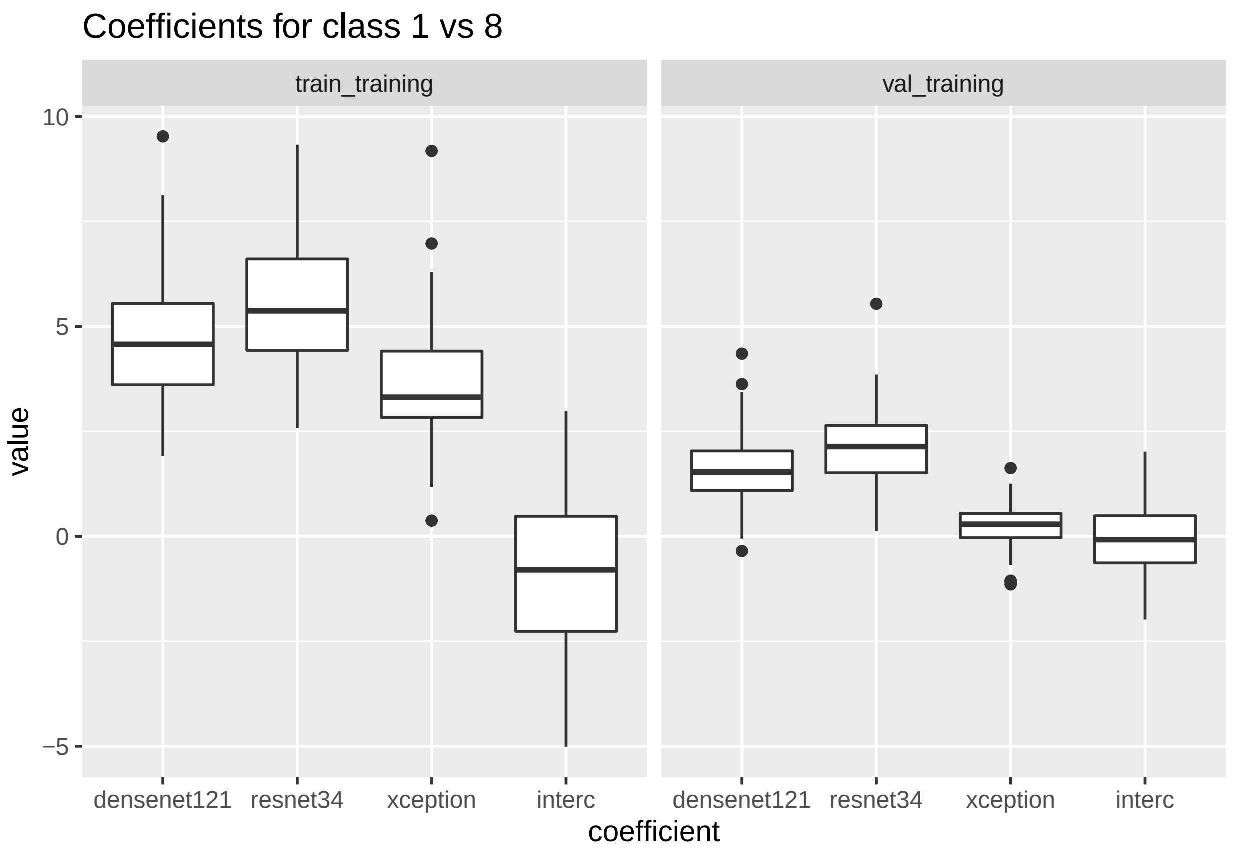

Coefficients for class 1 vs 4
train_training
val_training
value
6
3
0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

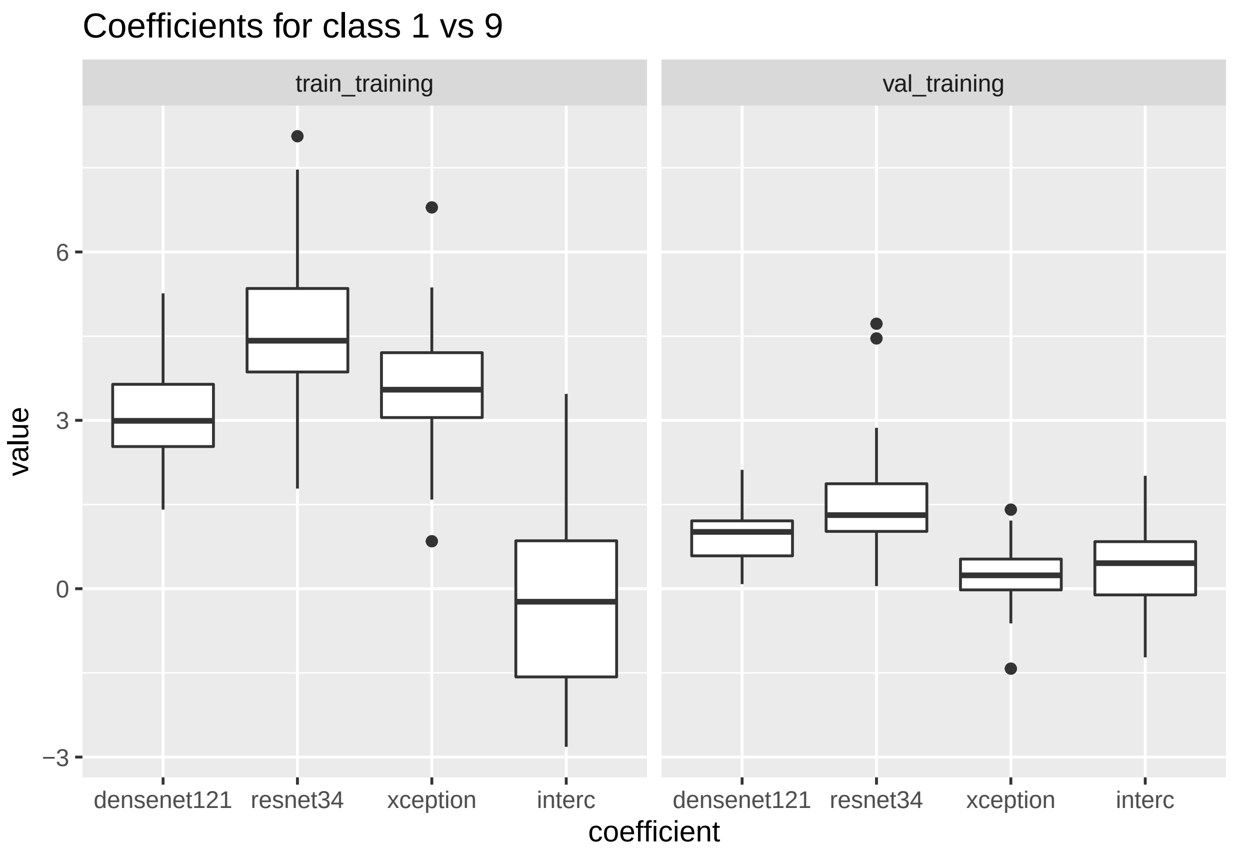


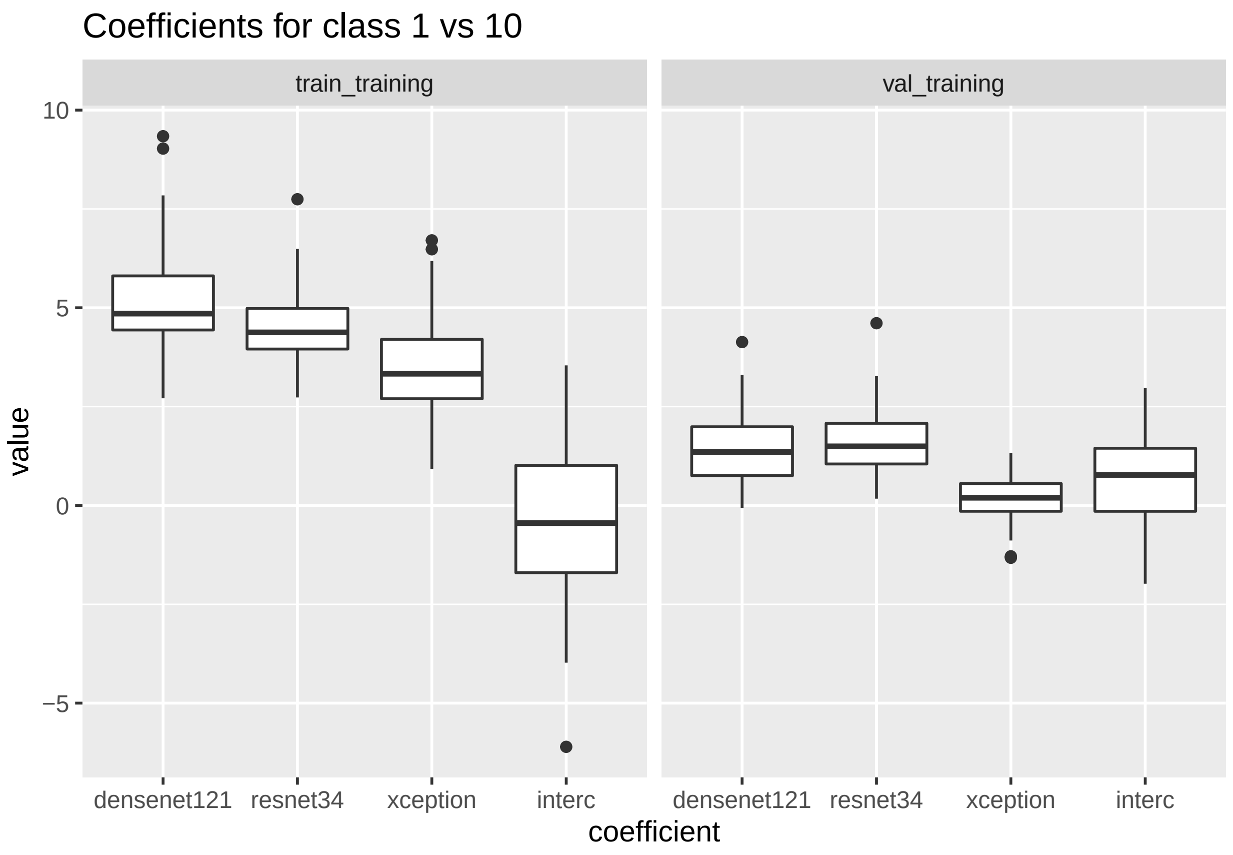
Coefficients for class 1 vs 5
train_training
val_training
10.0
7.5
5.0
2.5
0.0
value
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

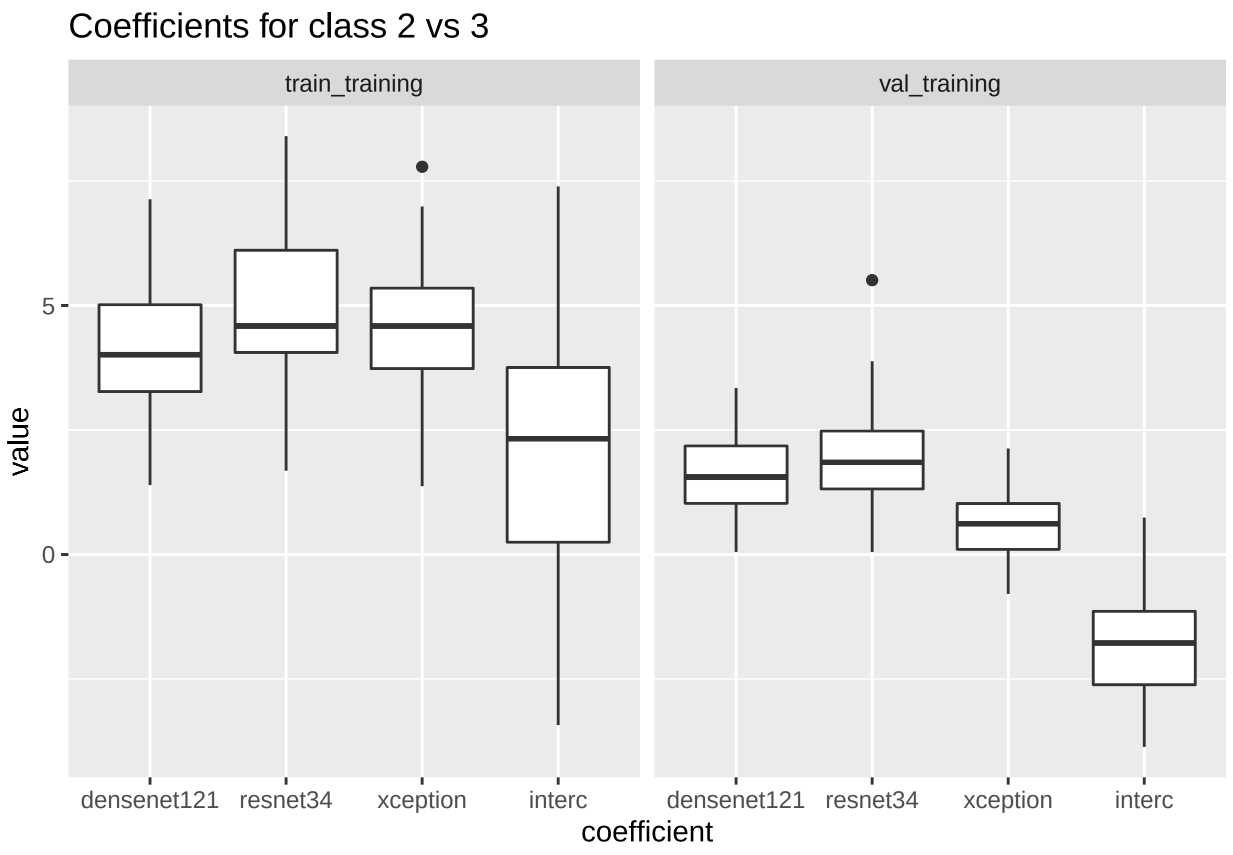

Coefficients for class 1 vs 6
train_training
val_training
value
7.5
5.0
2.5
0.0
−2.5
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

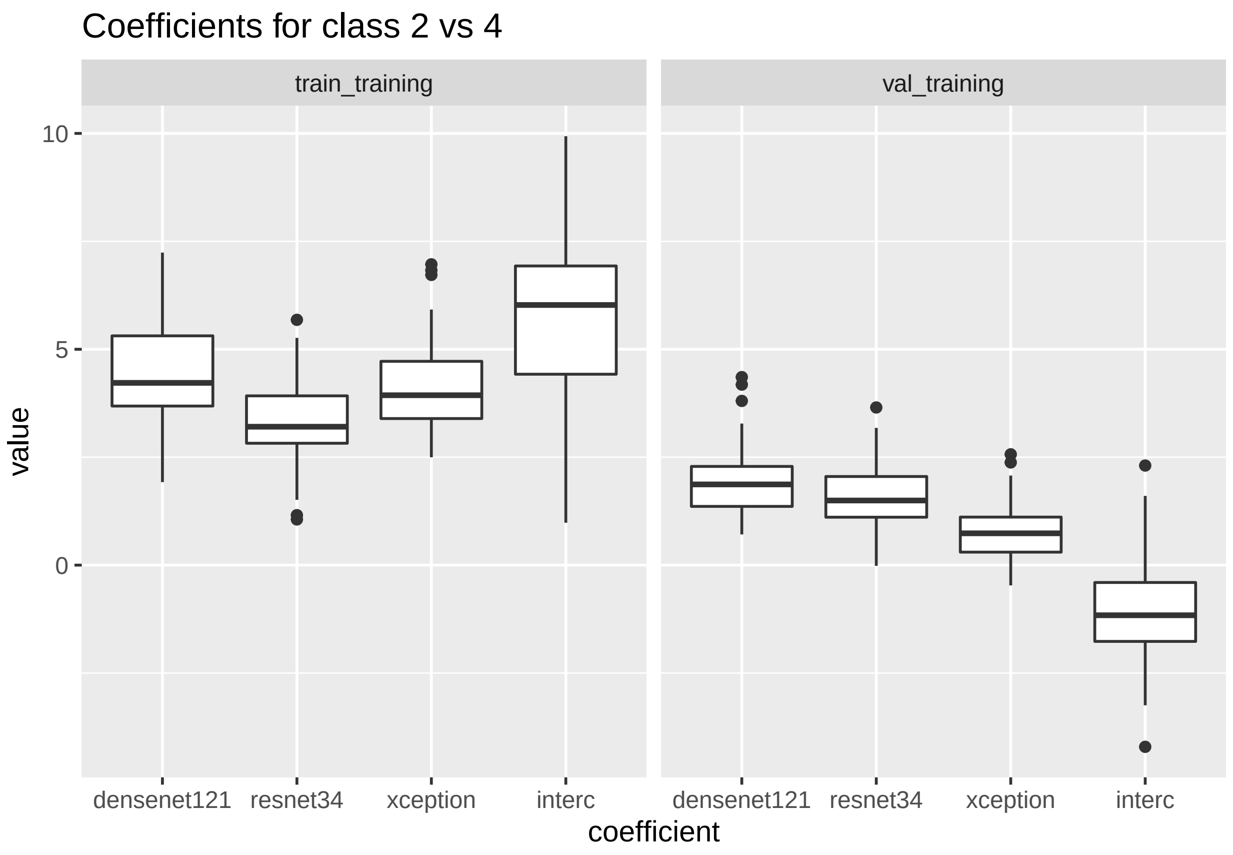

Coefficients for class 1 vs 7
train_training
val_training
value
6
4
2
0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

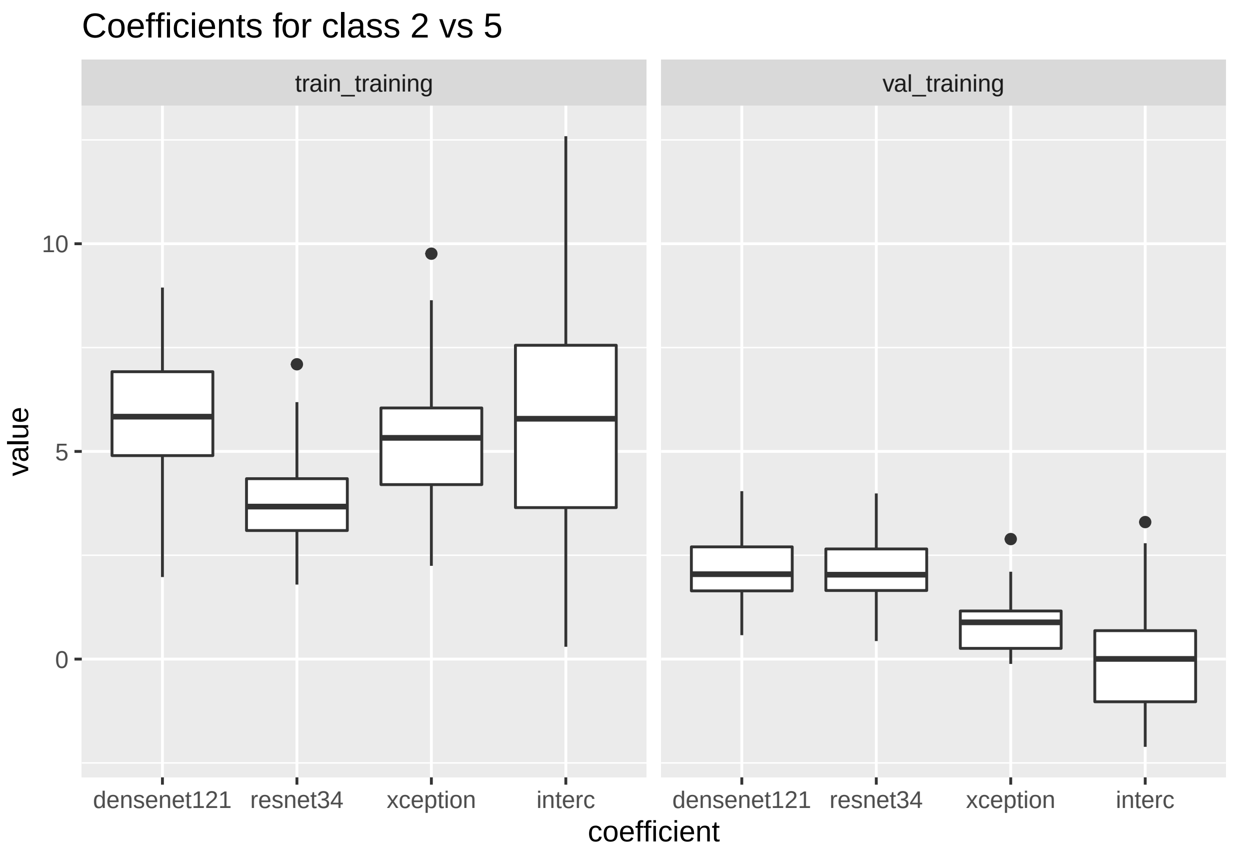

Coefficients for class 1 vs 8
train_training
val_training
value
10
5
0
−5
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

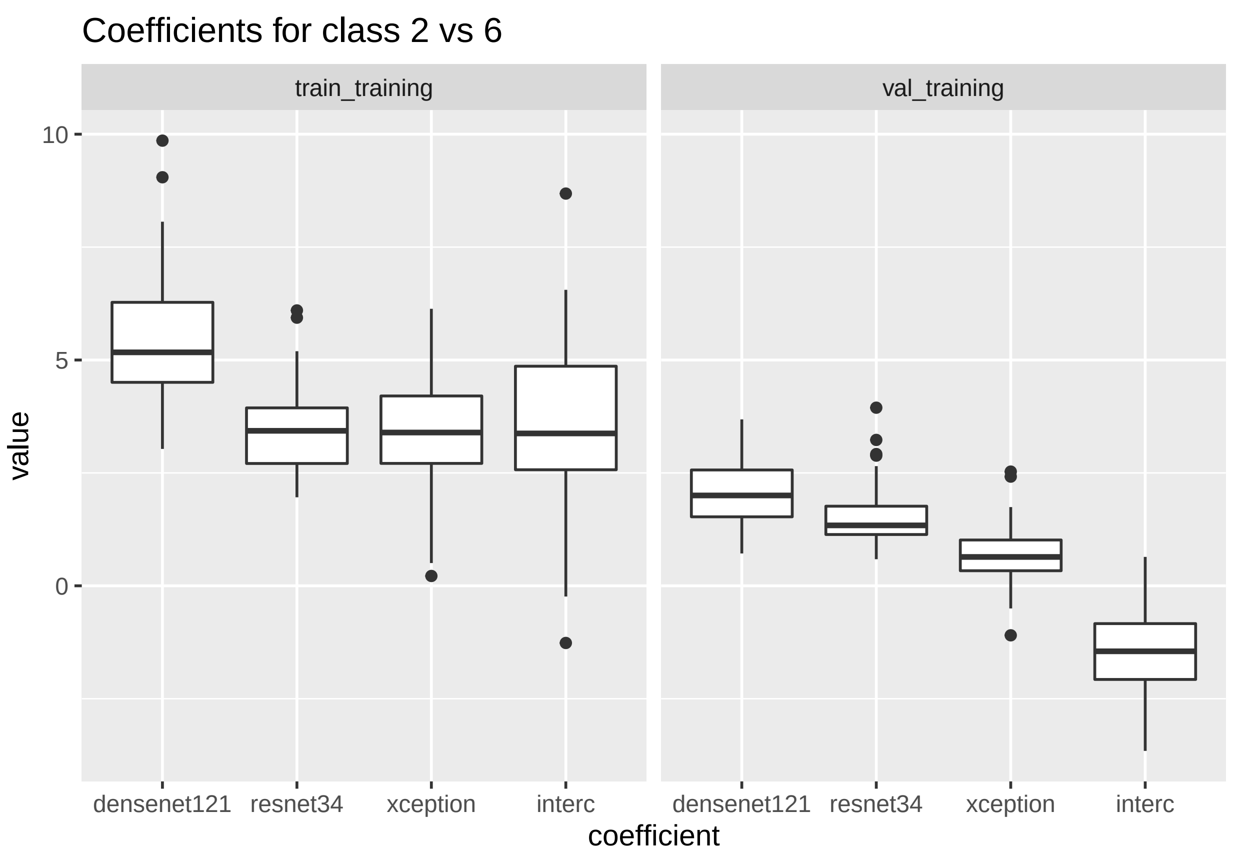

Coefficients for class 1 vs 9
train_training
val_training
value
6
3
0
−3
densenet121
resnet34
xception
interc
coefficient

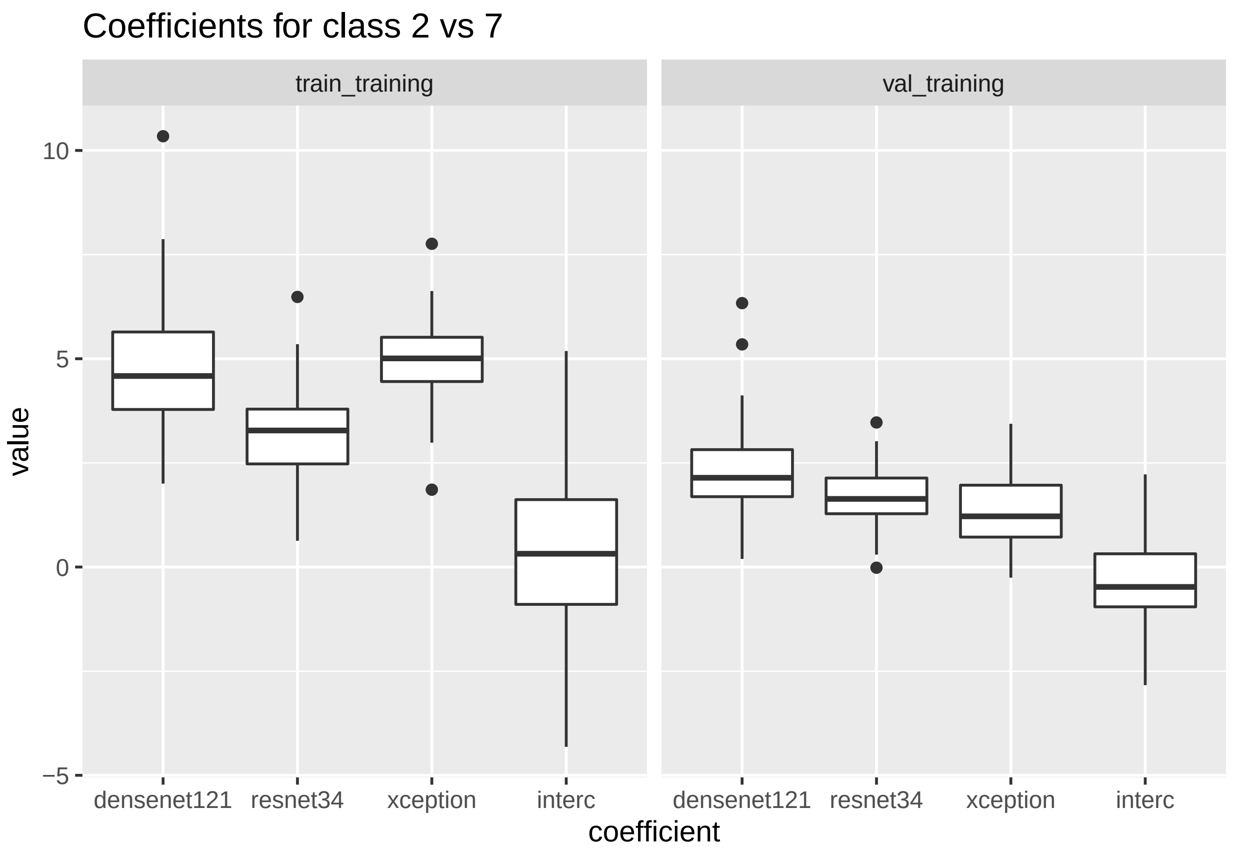

Coefficients for class 1 vs 10
train_training
val_training
value
10
5
0
−5
densenet121
resnet34
xception
interc
coefficient

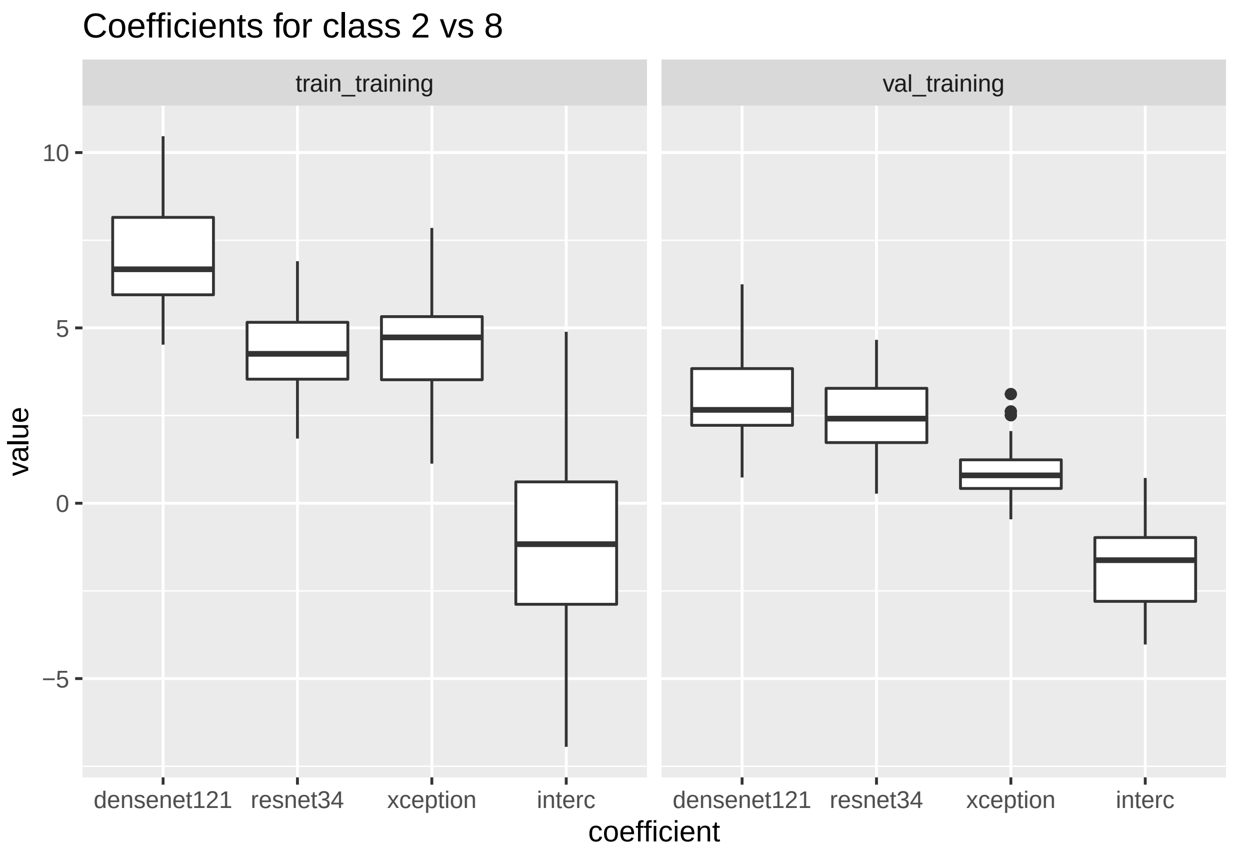

Coefficients for class 2 vs 3
train_training
val_training
value
5
0
densenet121
resnet34
xception
interc
coefficient


Coefficients for class 2 vs 4
train_training
val_training
value
10
5
0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

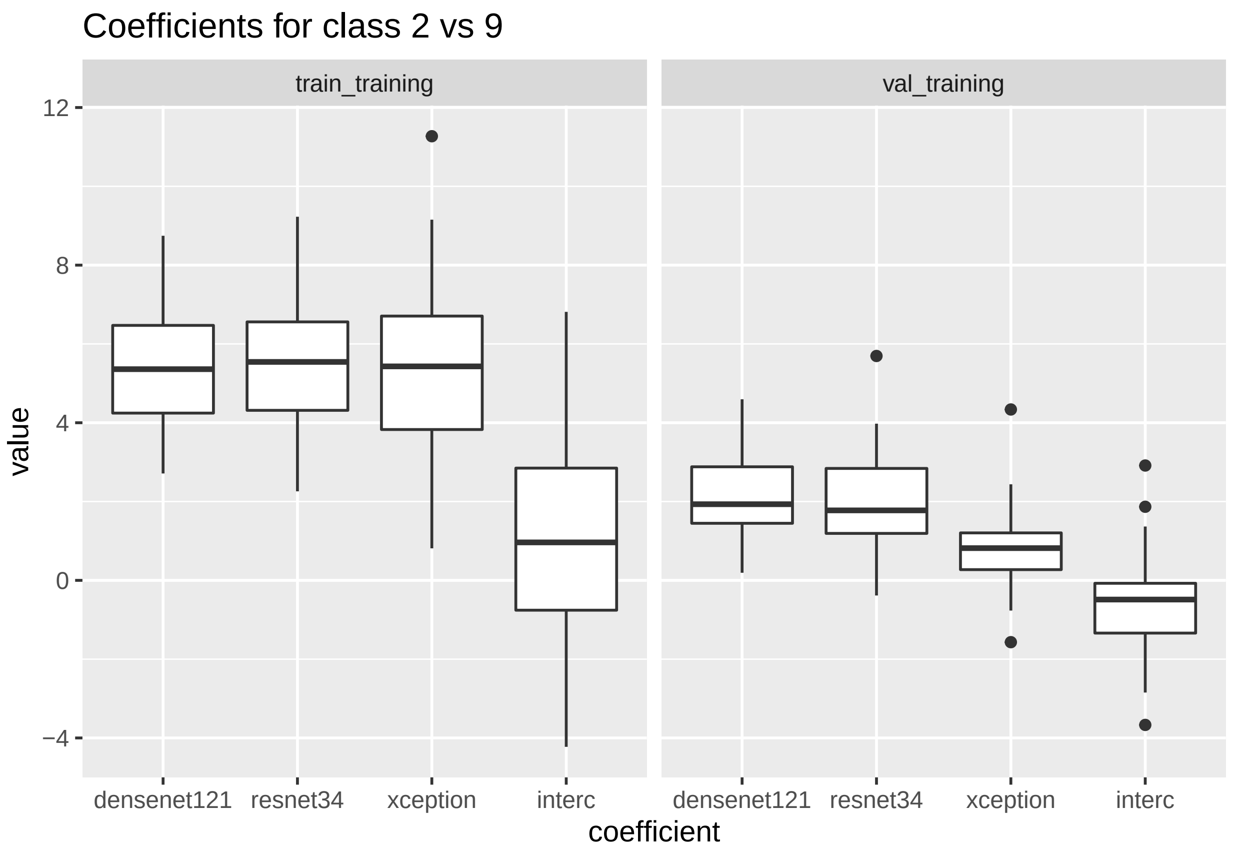

Coefficients for class 2 vs 5
train_training
val_training
value
10
5
0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

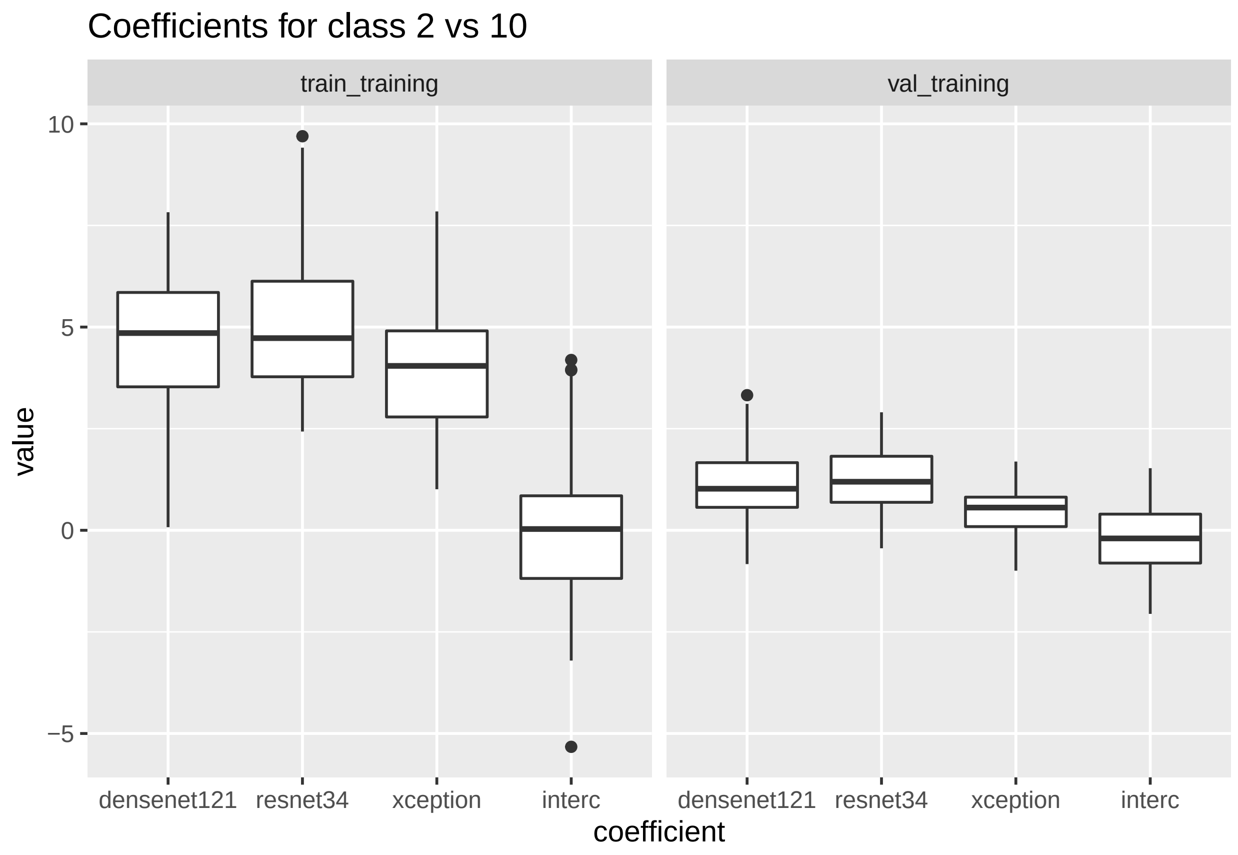

Coefficients for class 2 vs 6
train_training
val_training
10
5
0
value
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

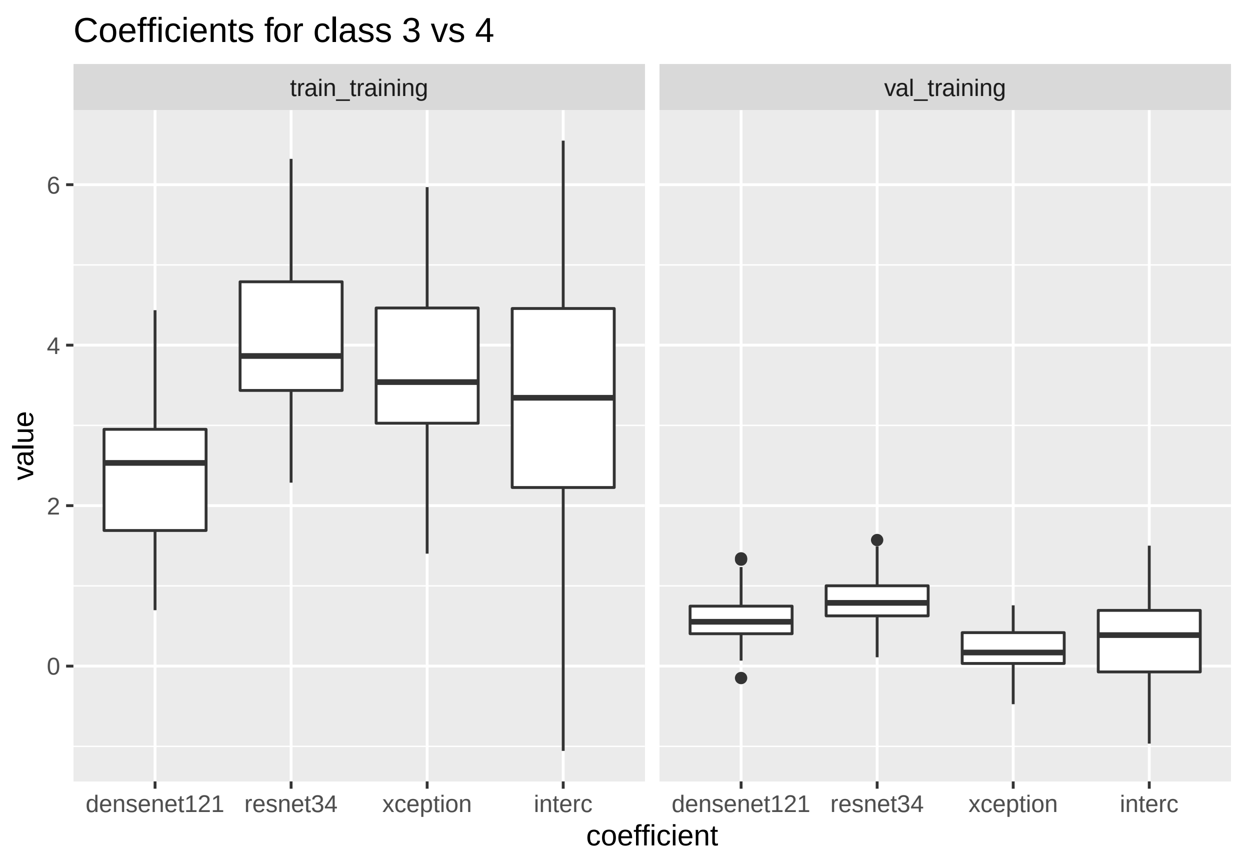

Coefficients for class 2 vs 7
train_training
val_training
value
10
5
0
−5
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

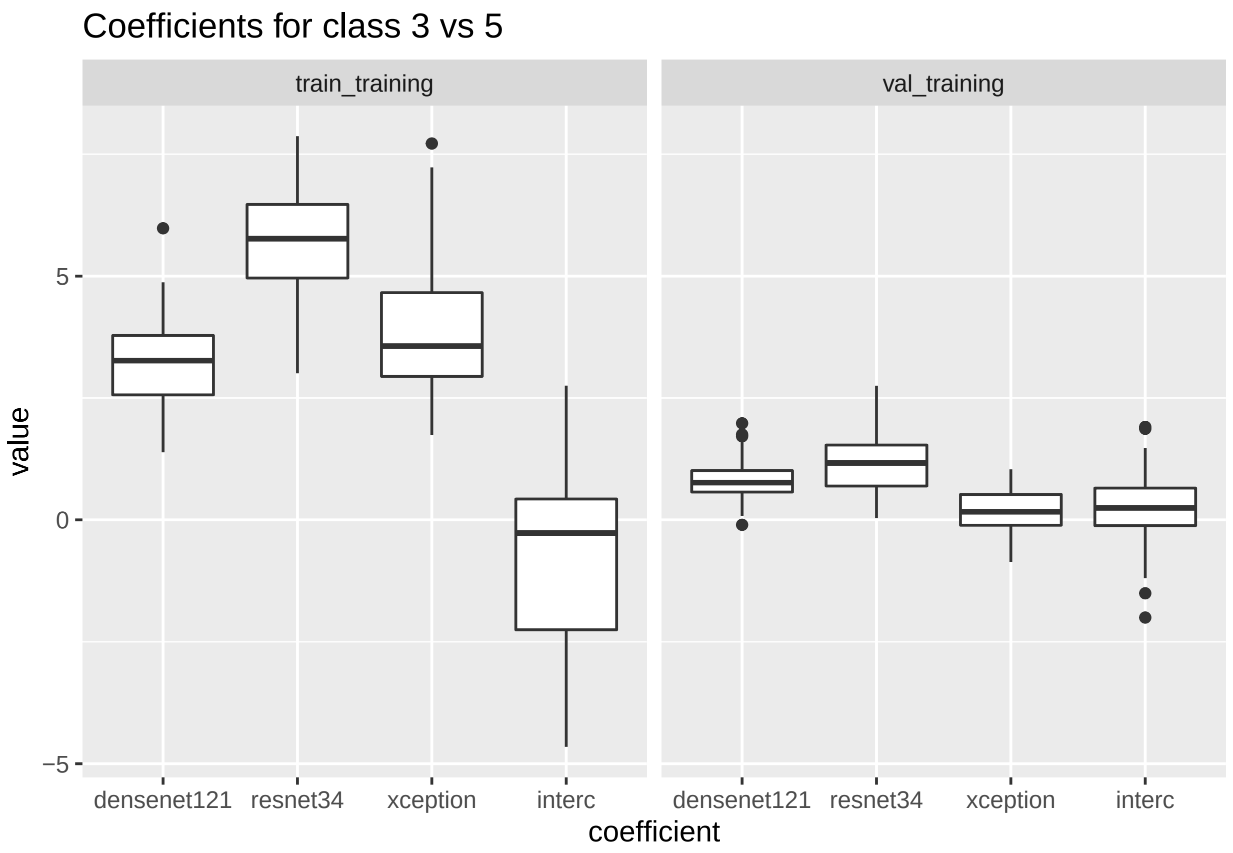


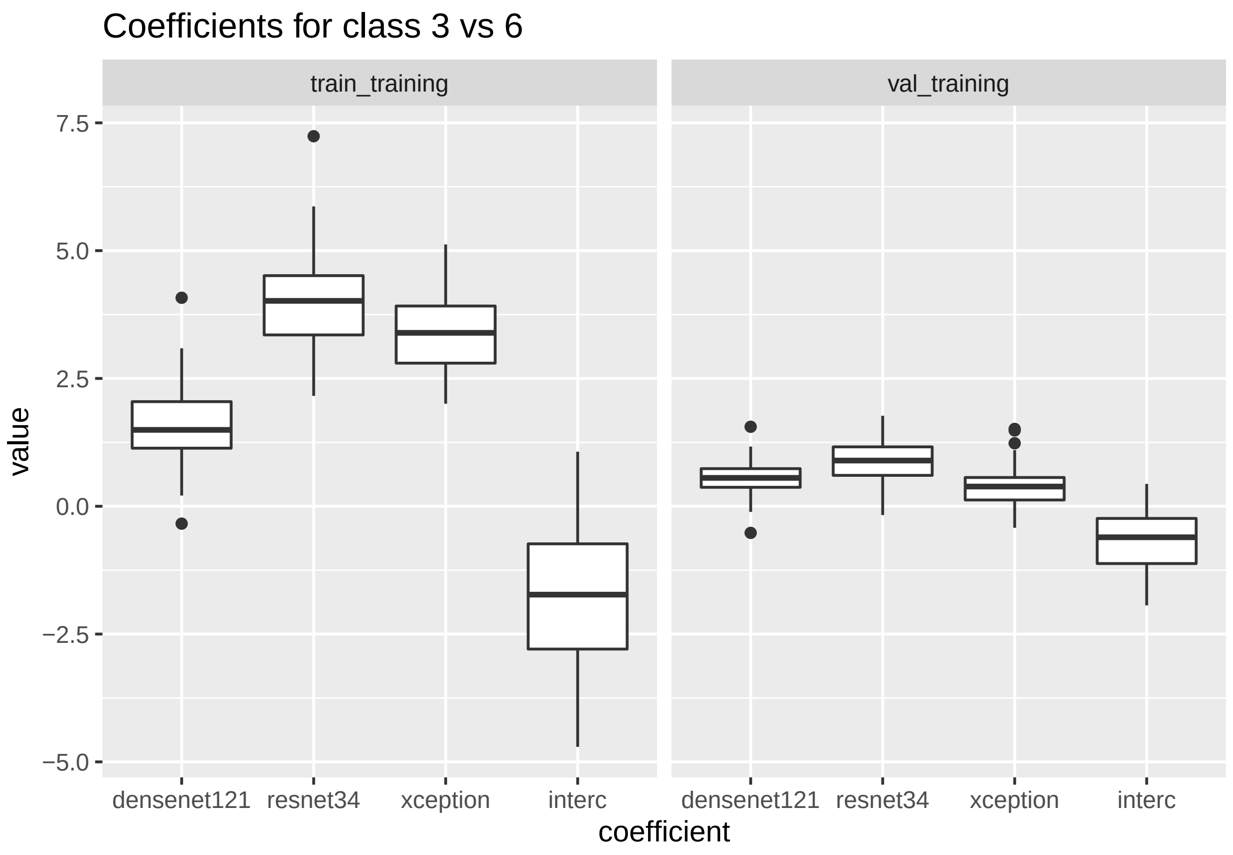
Coefficients for class 2 vs 8
train_training
val_training
value
10
5
0
−5
densenet121
resnet34
xception
interc
coefficient

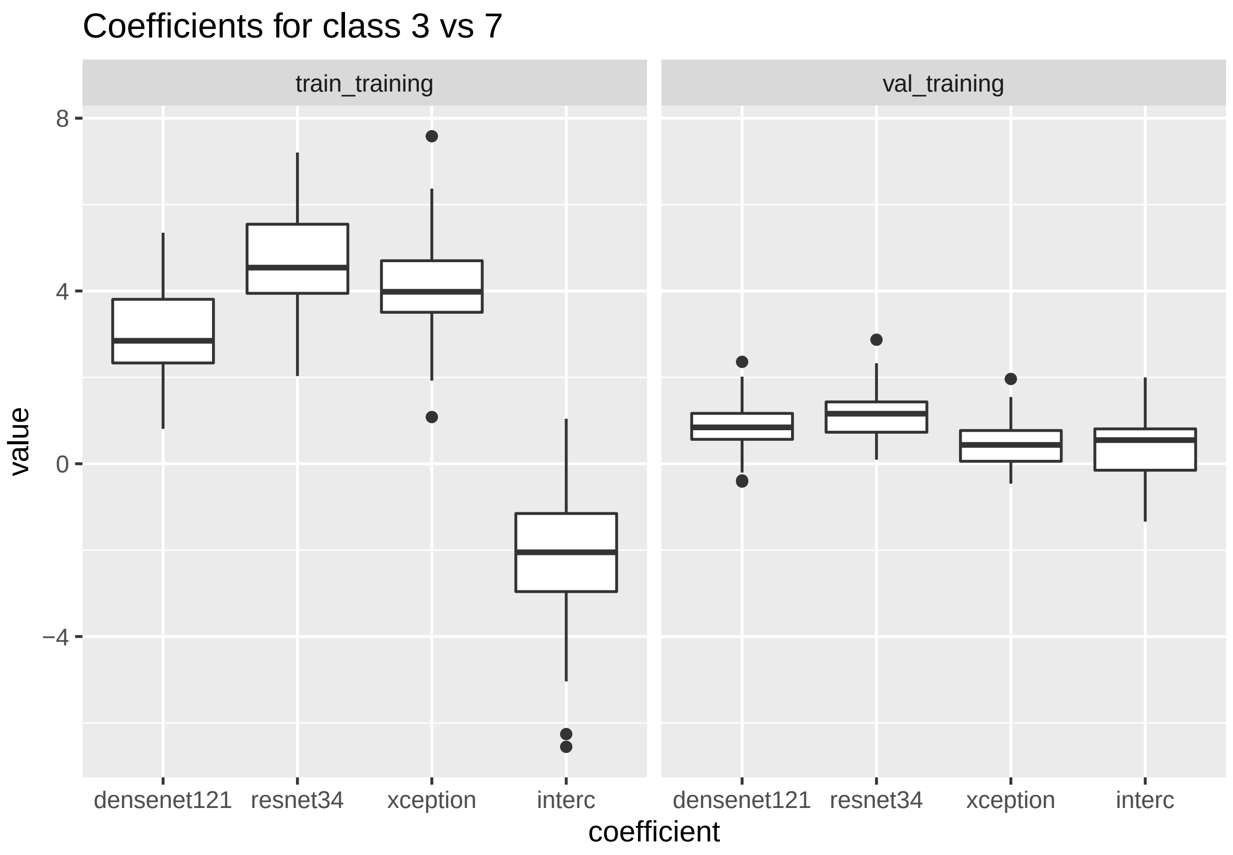

Coefficients for class 2 vs 9
train_training
val_training
value
coefficient
densenet121
resnet34
xception
interc
12
8
4
0
−4

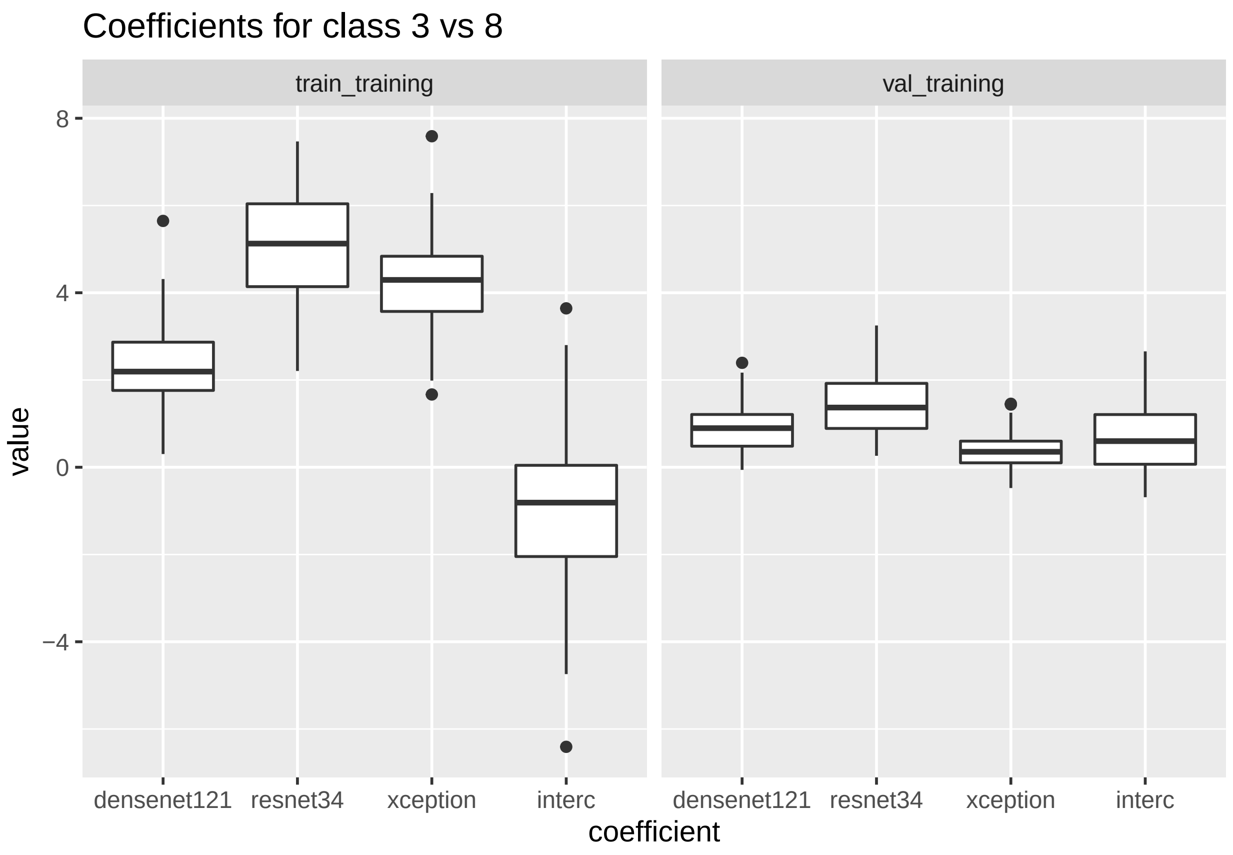

Coefficients for class 2 vs 10
train_training
val_training
value
10
5
0
−5
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

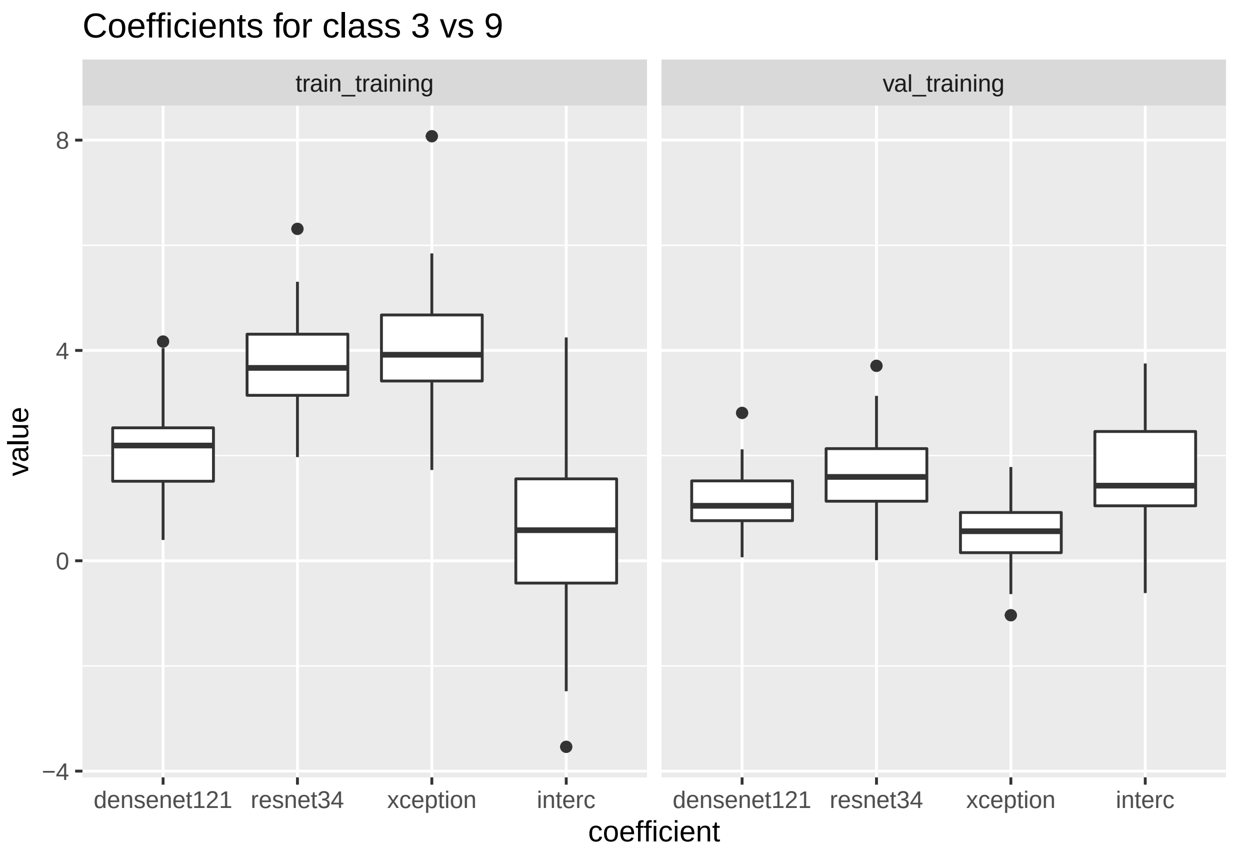

Coefficients for class 3 vs 4
train_training
val_training
value
6
4
2
0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

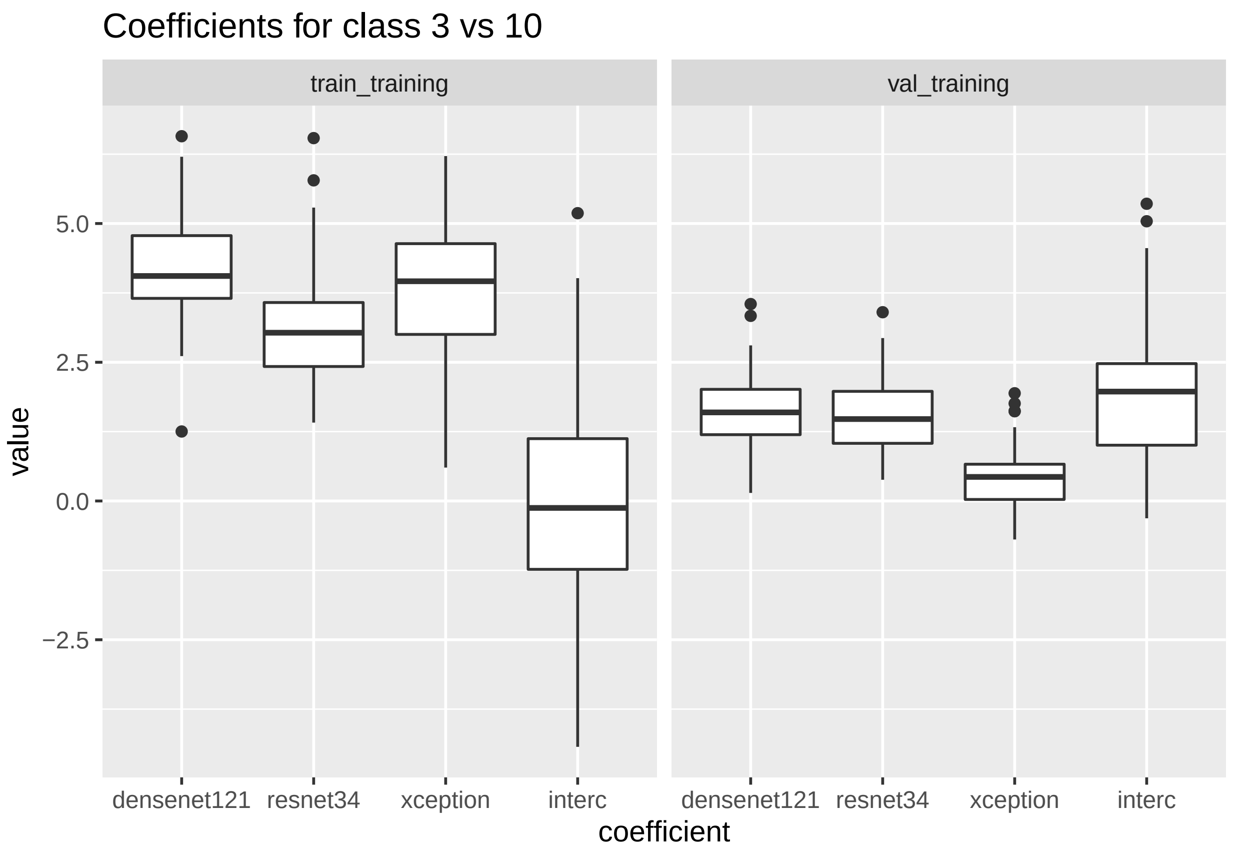

Coefficients for class 3 vs 5
train_training
val_training
value
5
0
−5
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

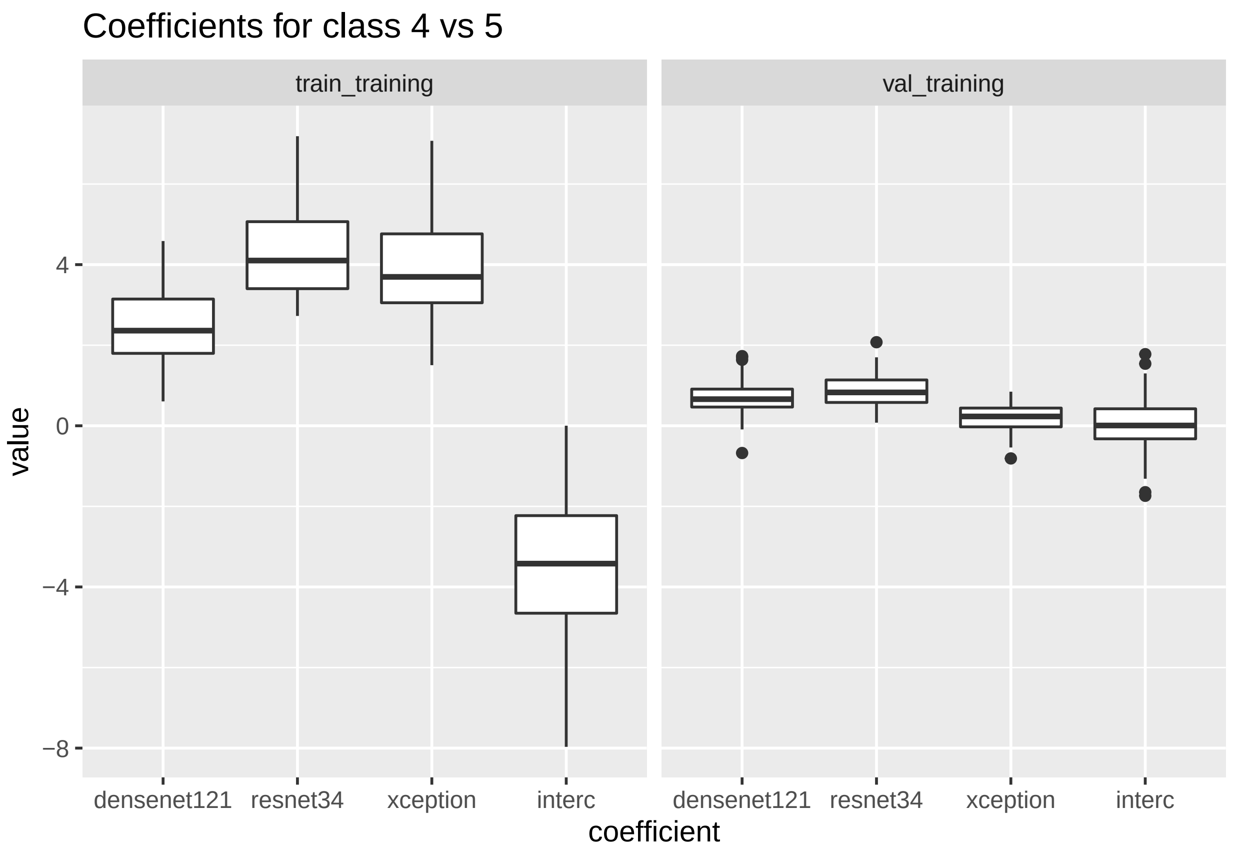

Coefficients for class 3 vs 6
train_training
val_training
value
7.5
5.0
2.5
0.0
−2.5
−5.0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient


Coefficients for class 3 vs 7
train_training
val_training
8
4
0
−4
value
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

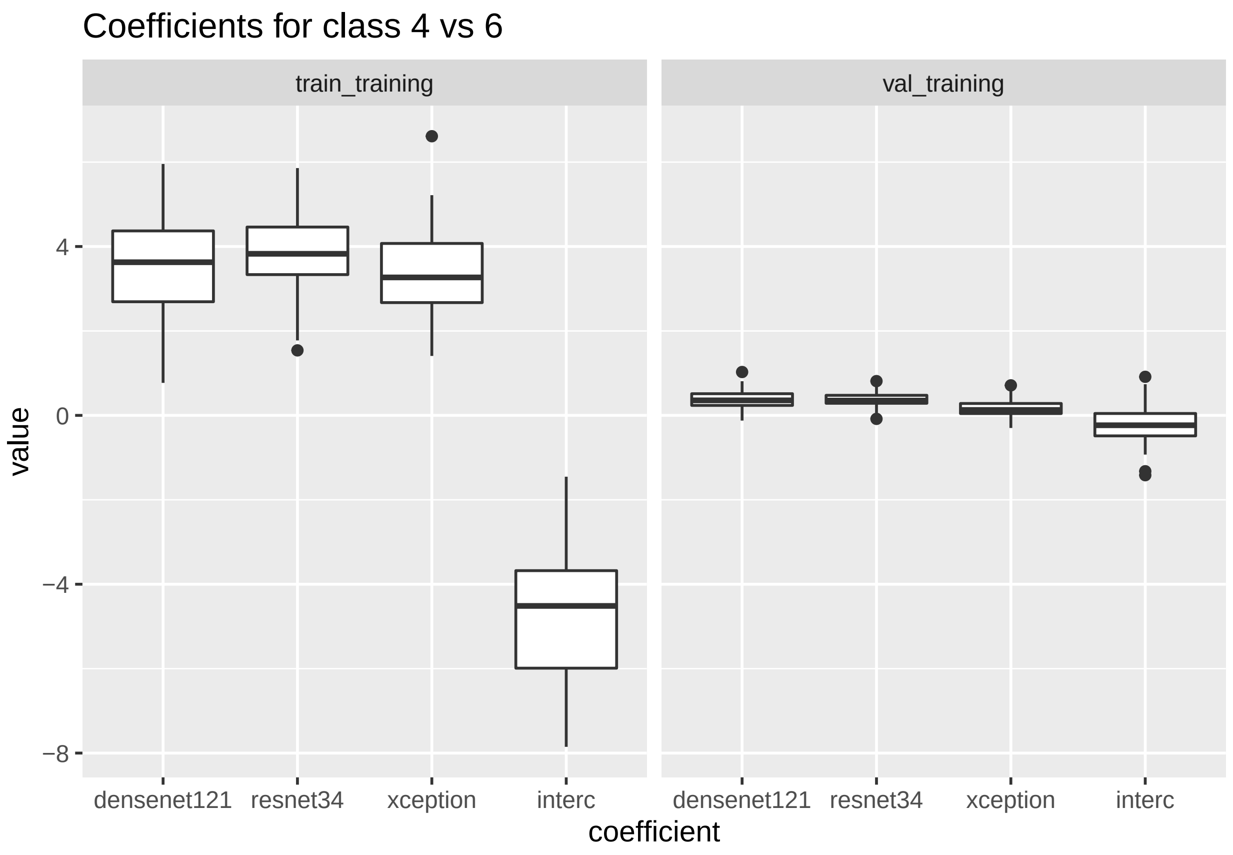

Coefficients for class 3 vs 8
train_training
val_training
8
4
0
−4
value
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

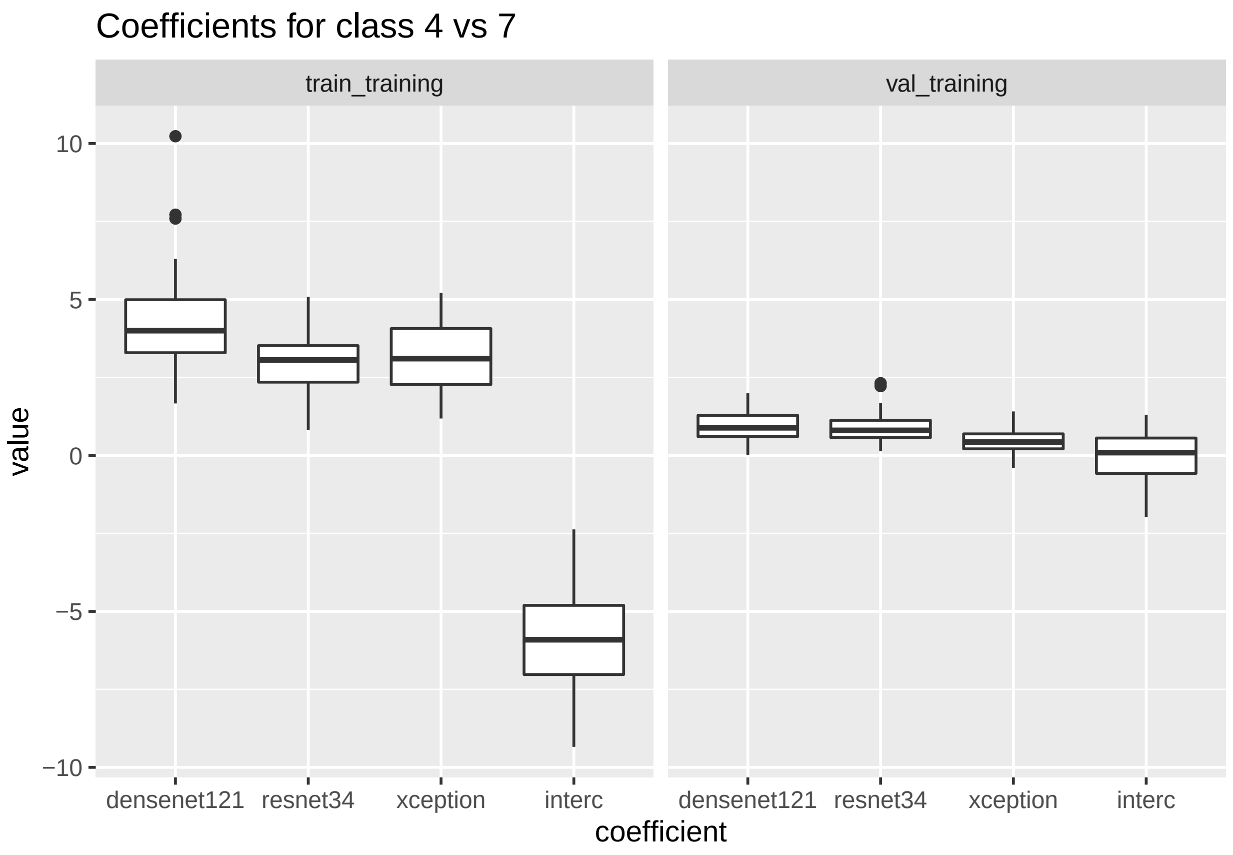

Coefficients for class 3 vs 9
train_training
val_training
value
8
4
0
−4
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

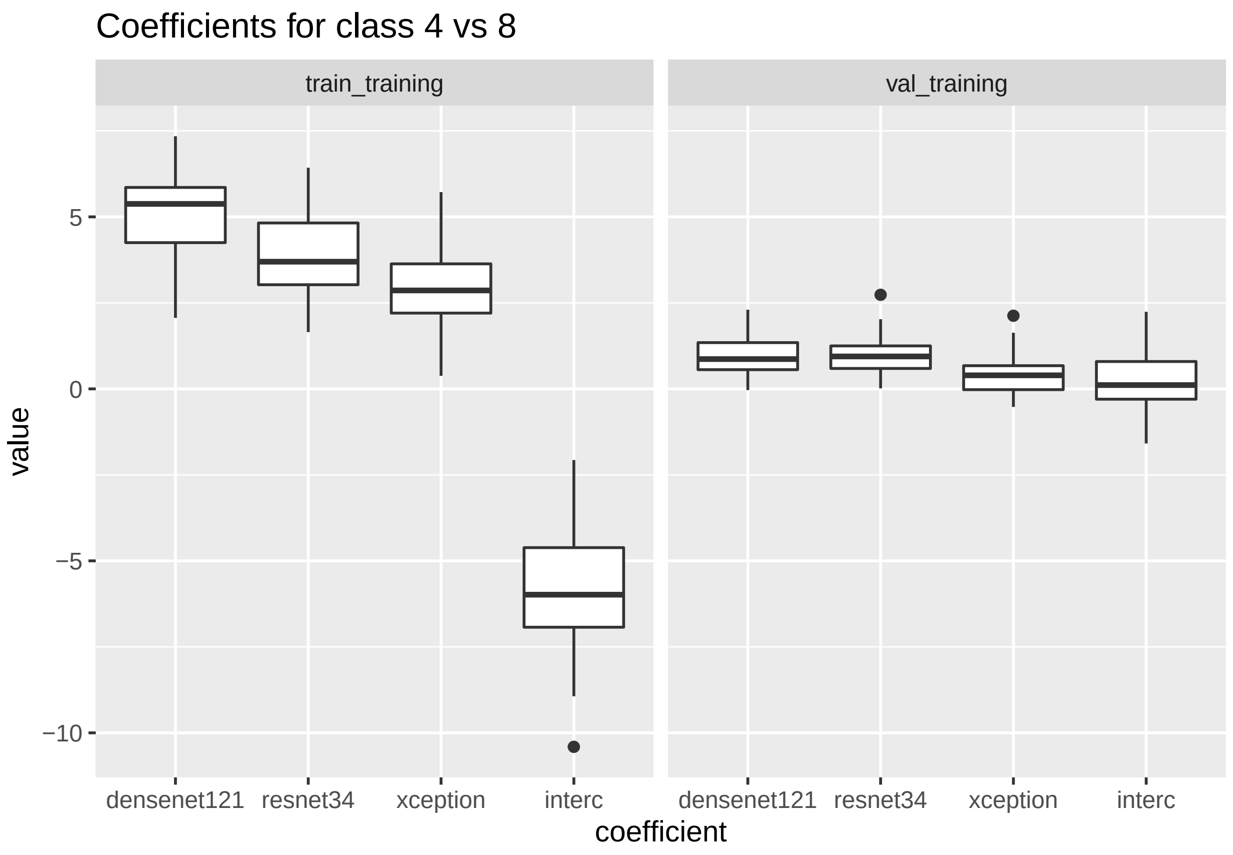

Coefficients for class 3 vs 10
train_training
val_training
value
5.0
2.5
0.0
−2.5
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

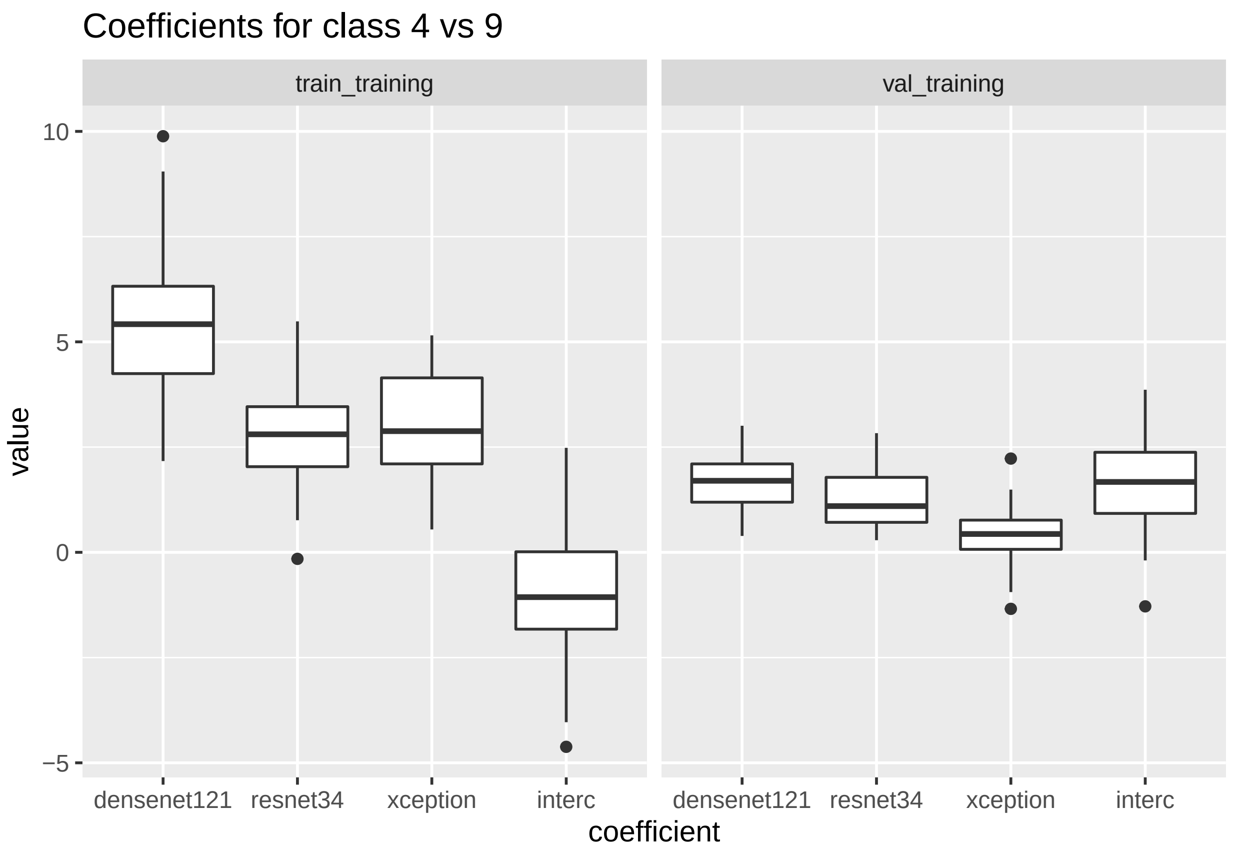


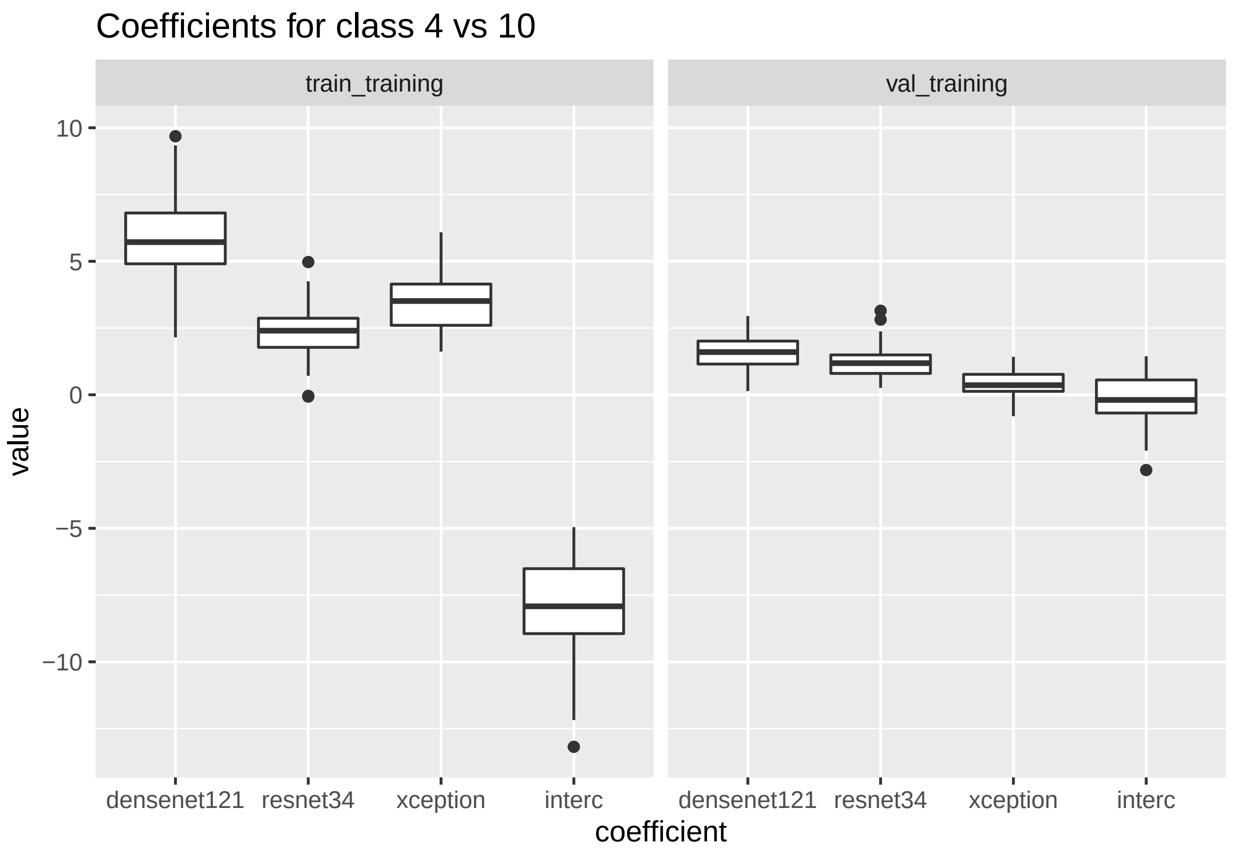
Coefficients for class 4 vs 5
train_training
val_training
value
4
0
−4
−8
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

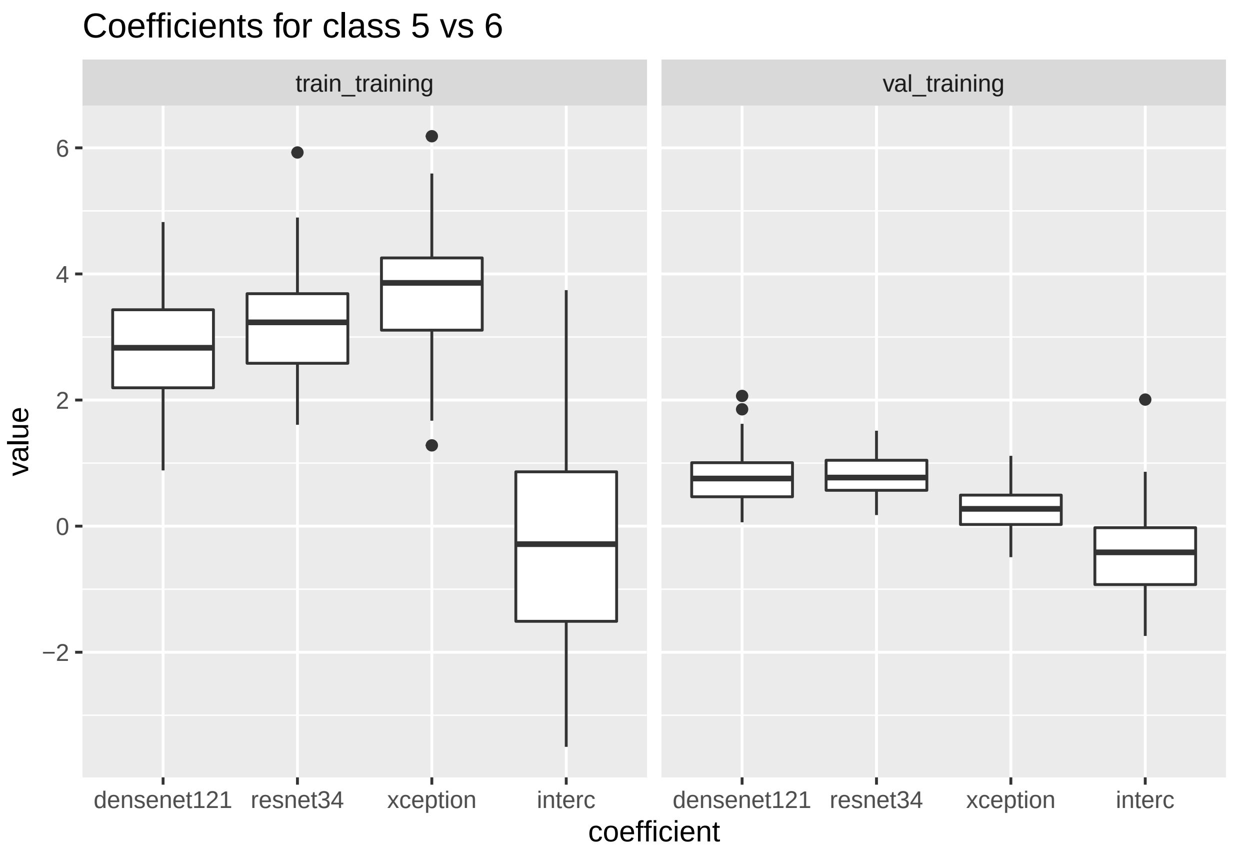

Coefficients for class 4 vs 6
train_training
val_training
value
4
0
−4
−8
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

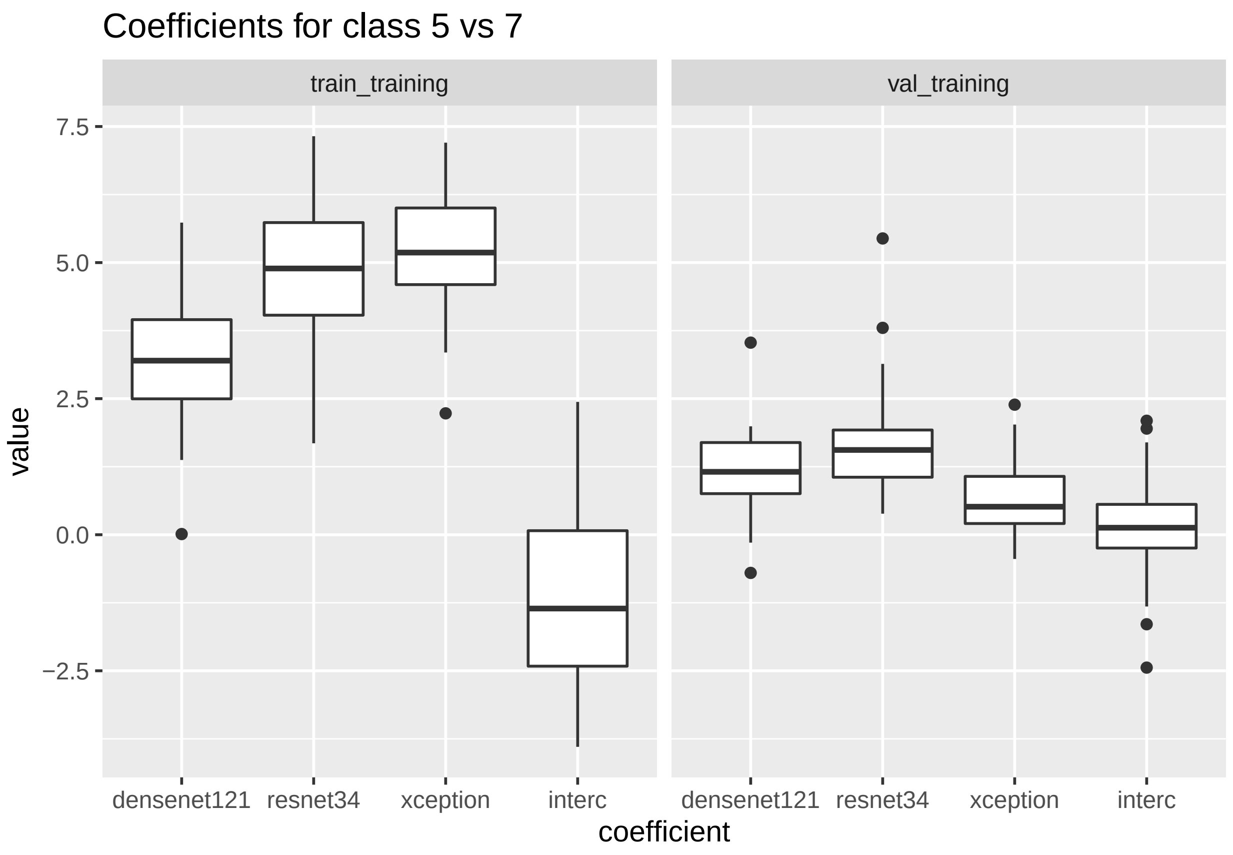

Coefficients for class 4 vs 7
train_training
val_training
value
10
5
0
−5
−10
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

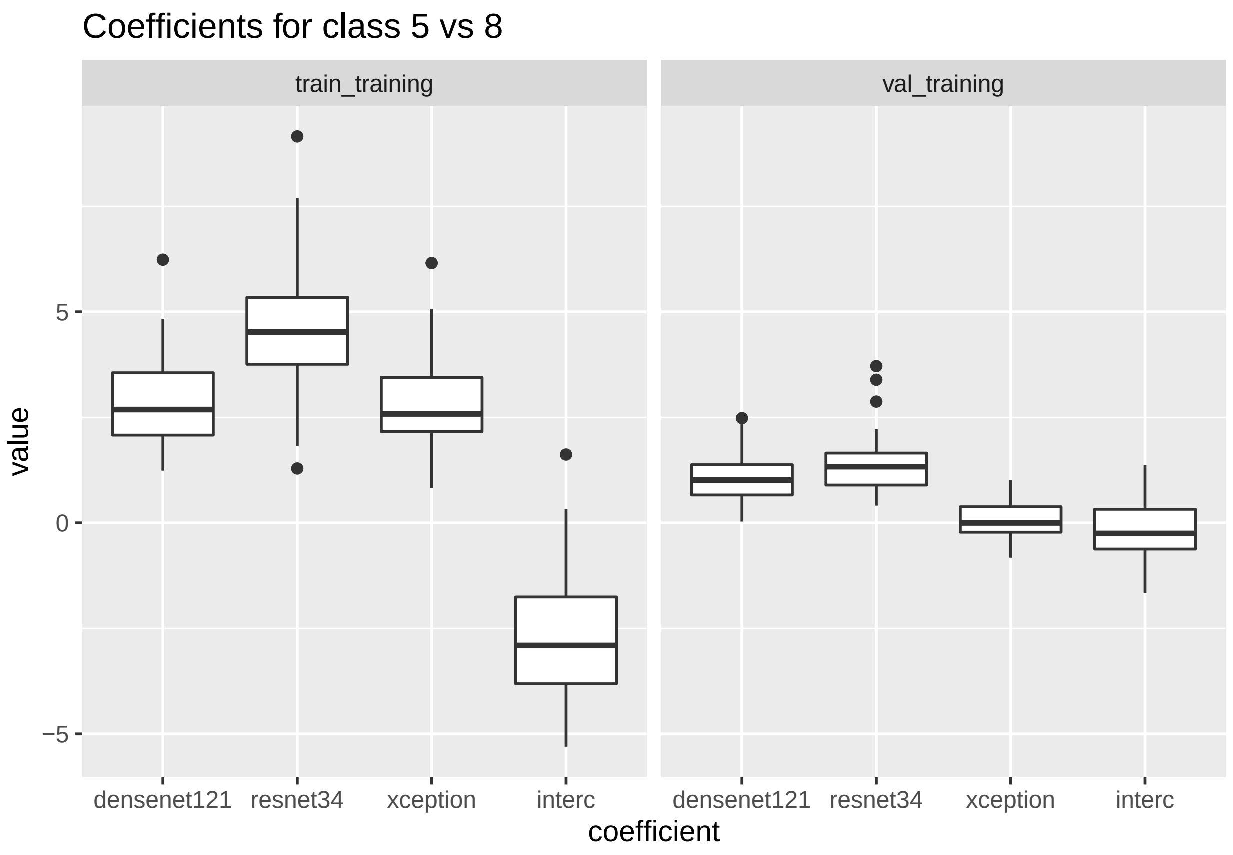

Coefficients for class 4 vs 8
train_training
val_training
value
5
0
−5
−10
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

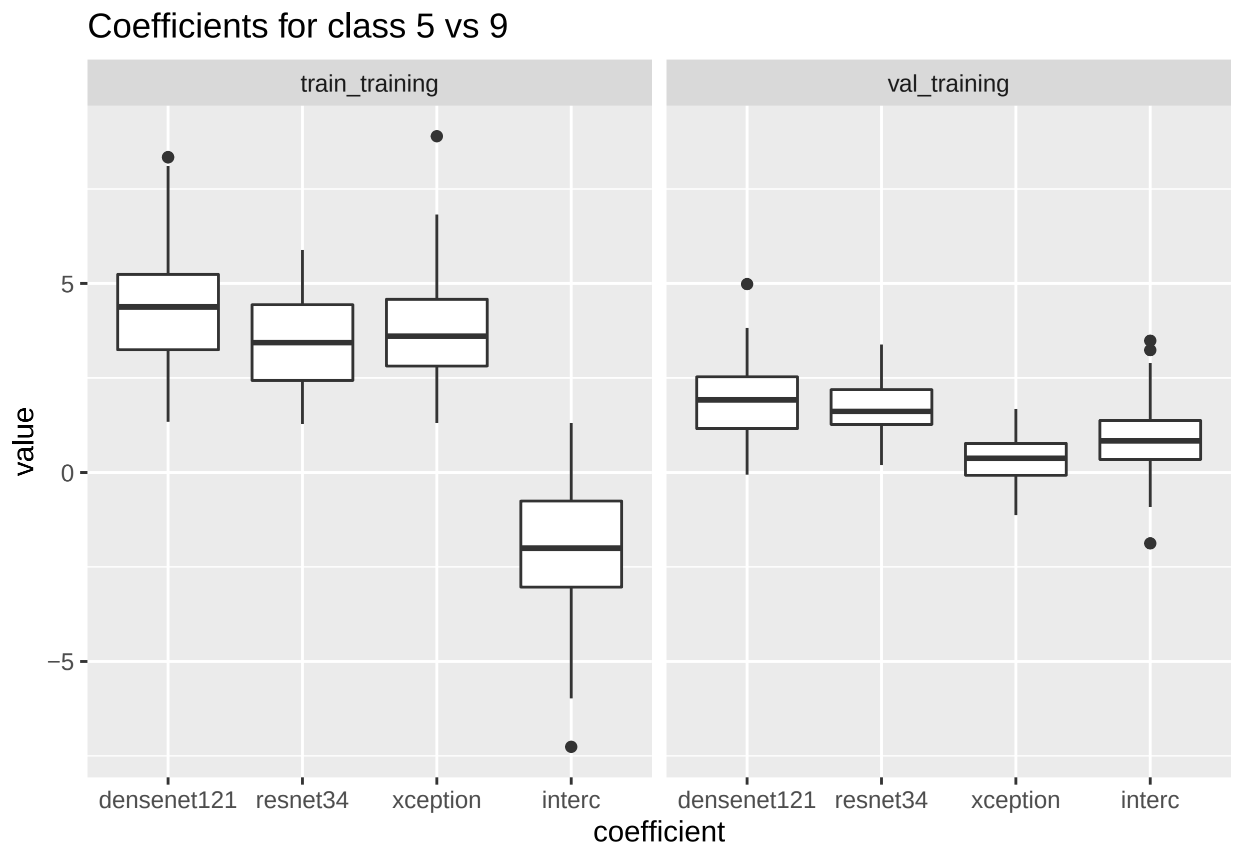

Coefficients for class 4 vs 9
train_training
val_training
10
5
0
−5
value
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

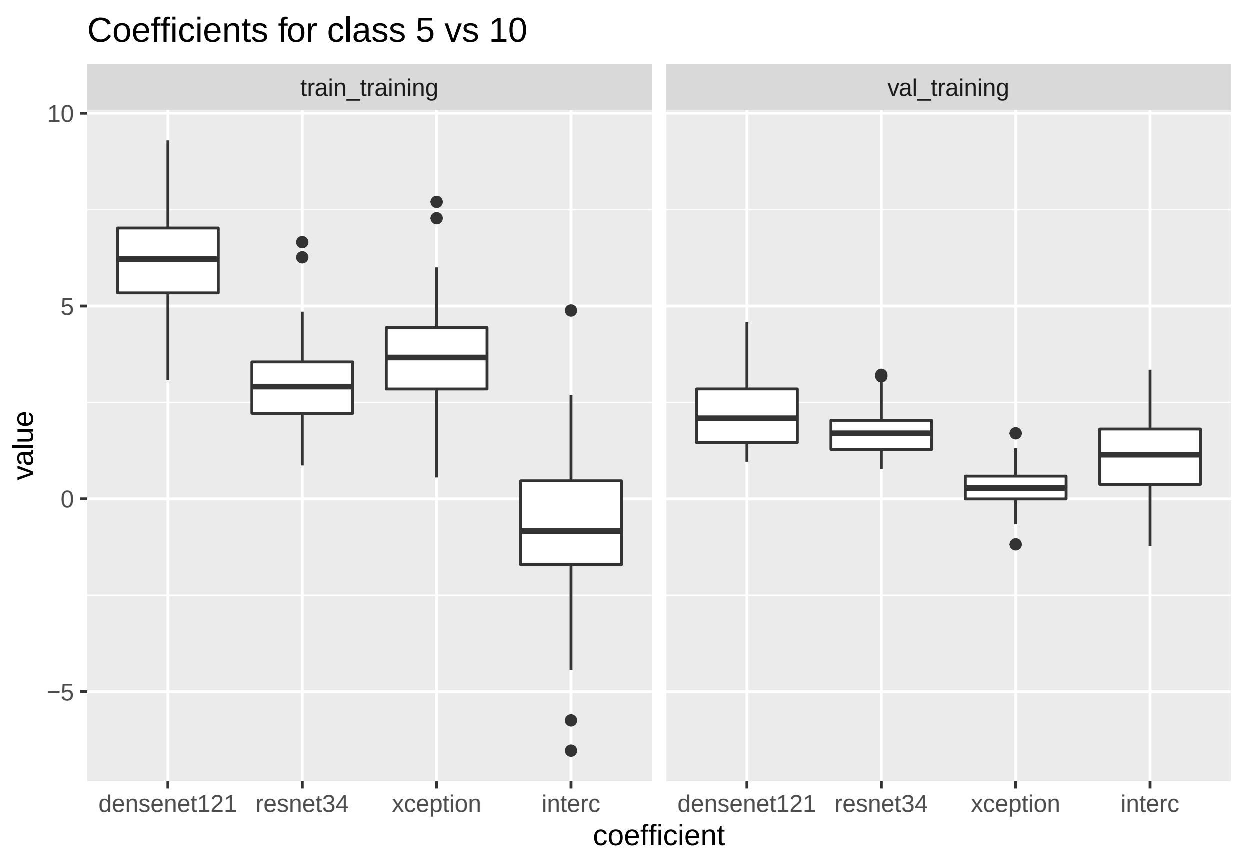

Coefficients for class 4 vs 10
train_training
val_training
10
5
0
−5
−10
value
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

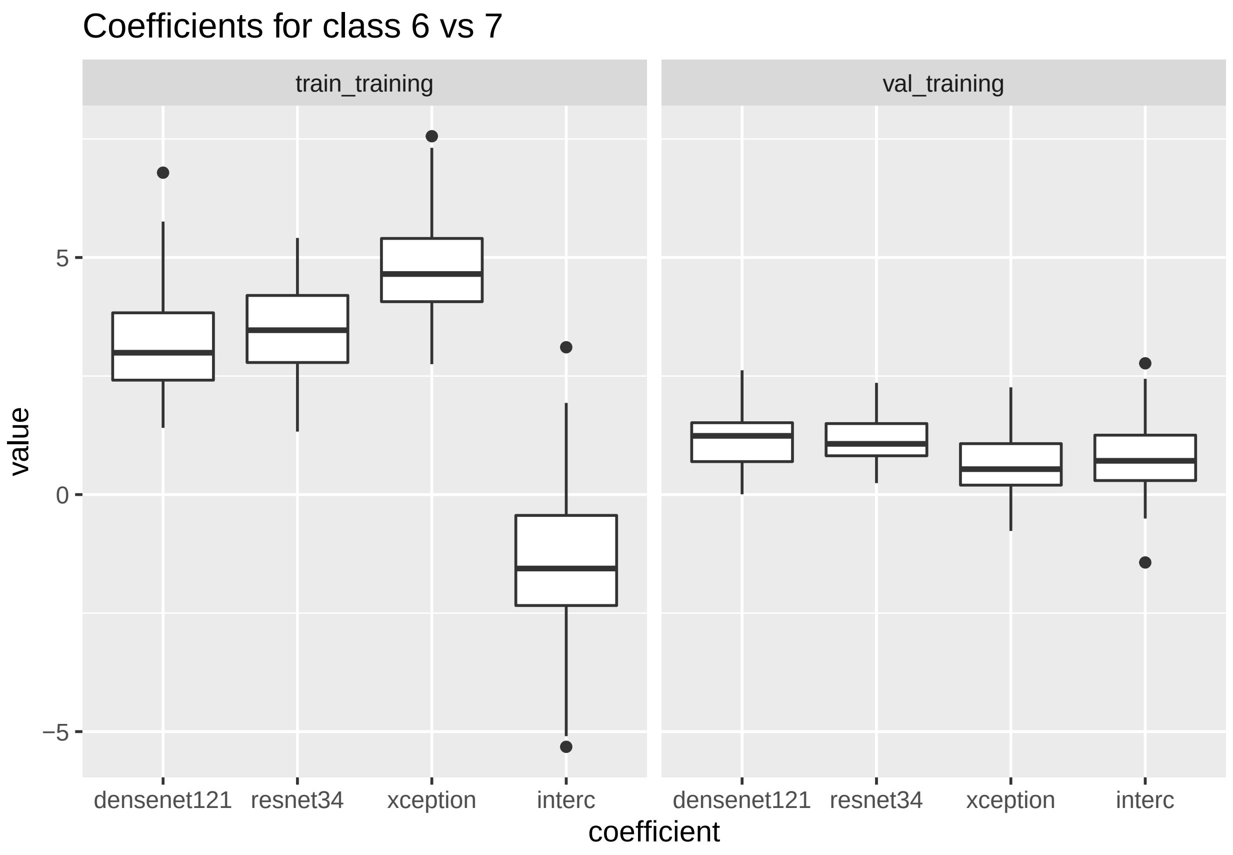

Coefficients for class 5 vs 6
train_training
val_training
value
6
4
2
0
−2
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

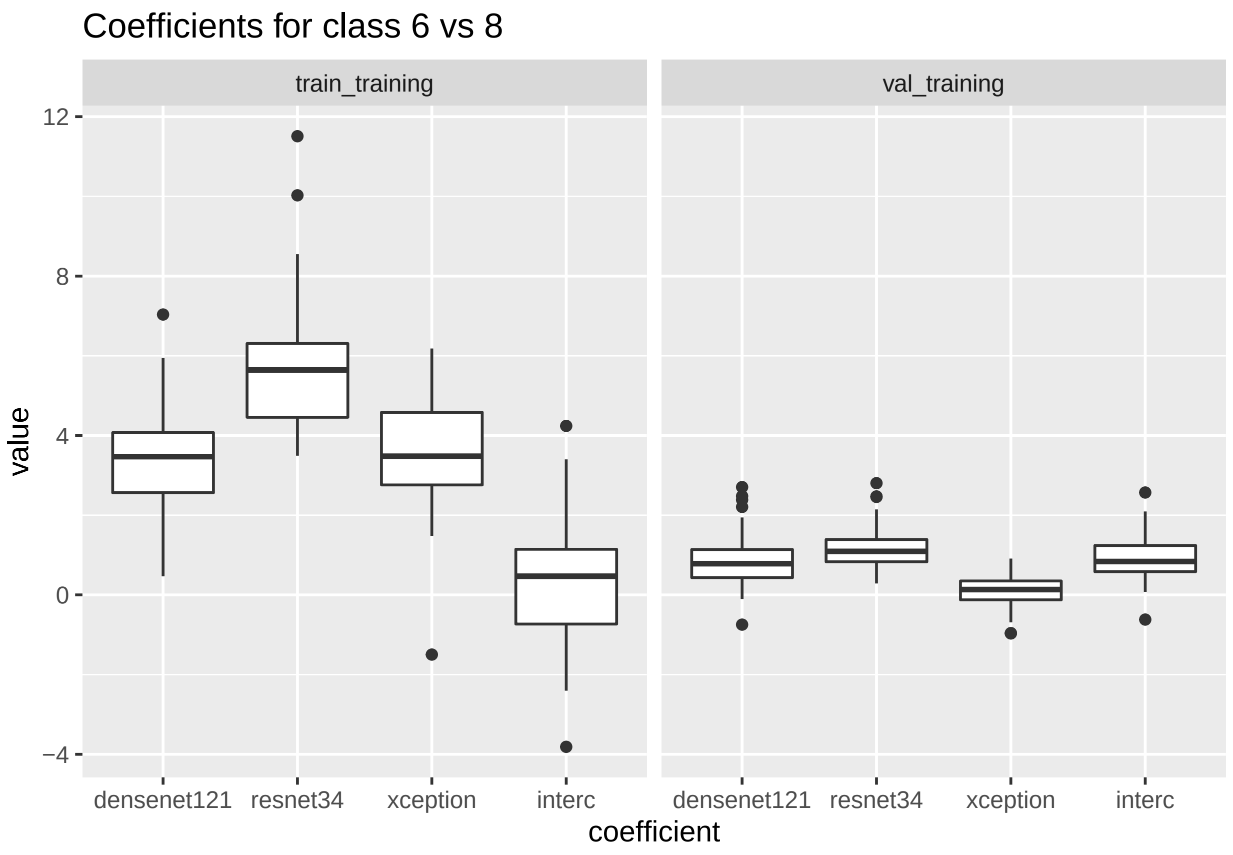

Coefficients for class 5 vs 7
train_training
val_training
value
7.5
5.0
2.5
0.0
−2.5
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient


Coefficients for class 5 vs 8
train_training
val_training
value
5
0
−5
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

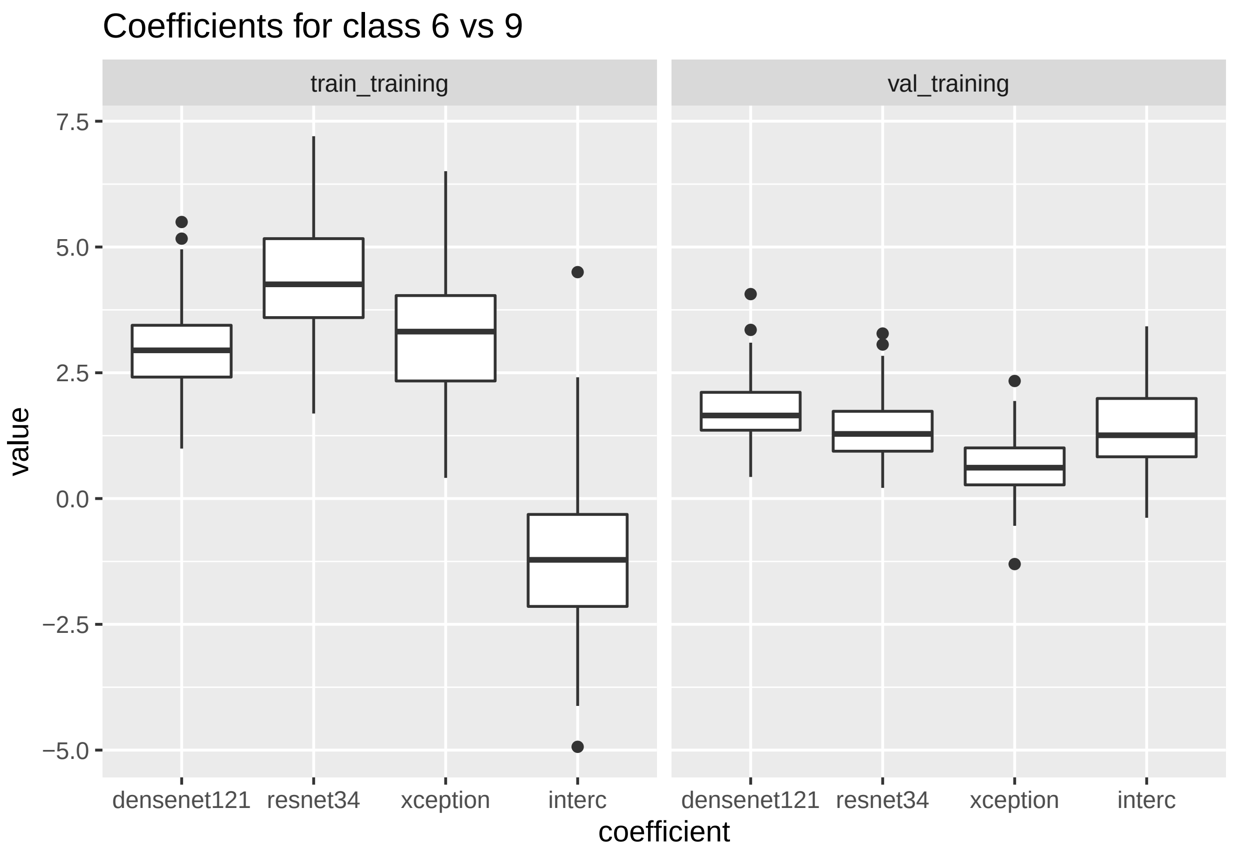

Coefficients for class 5 vs 9
train_training
val_training
value
5
0
−5
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

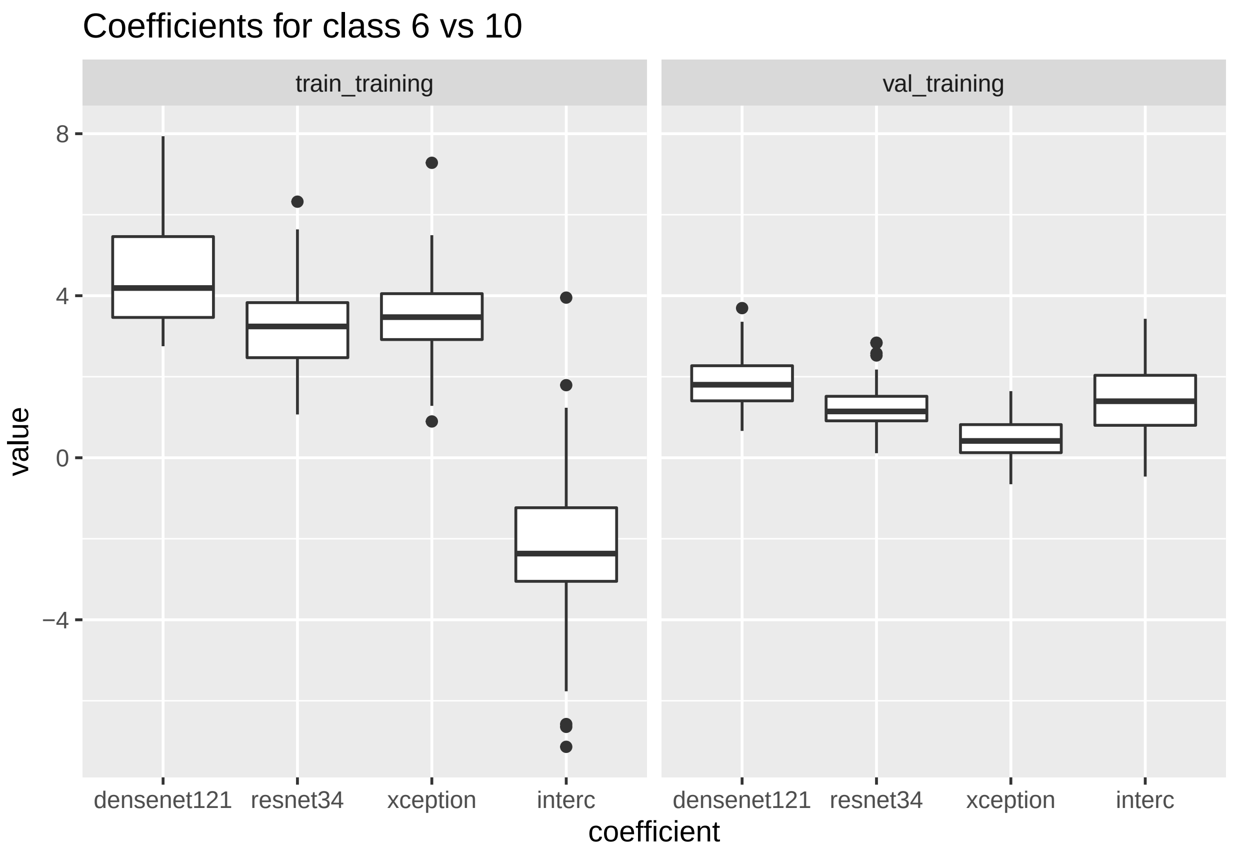

Coefficients for class 5 vs 10
train_training
val_training
value
10
5
0
−5
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient


Coefficients for class 6 vs 7
train_training
val_training
value
5
0
−5
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

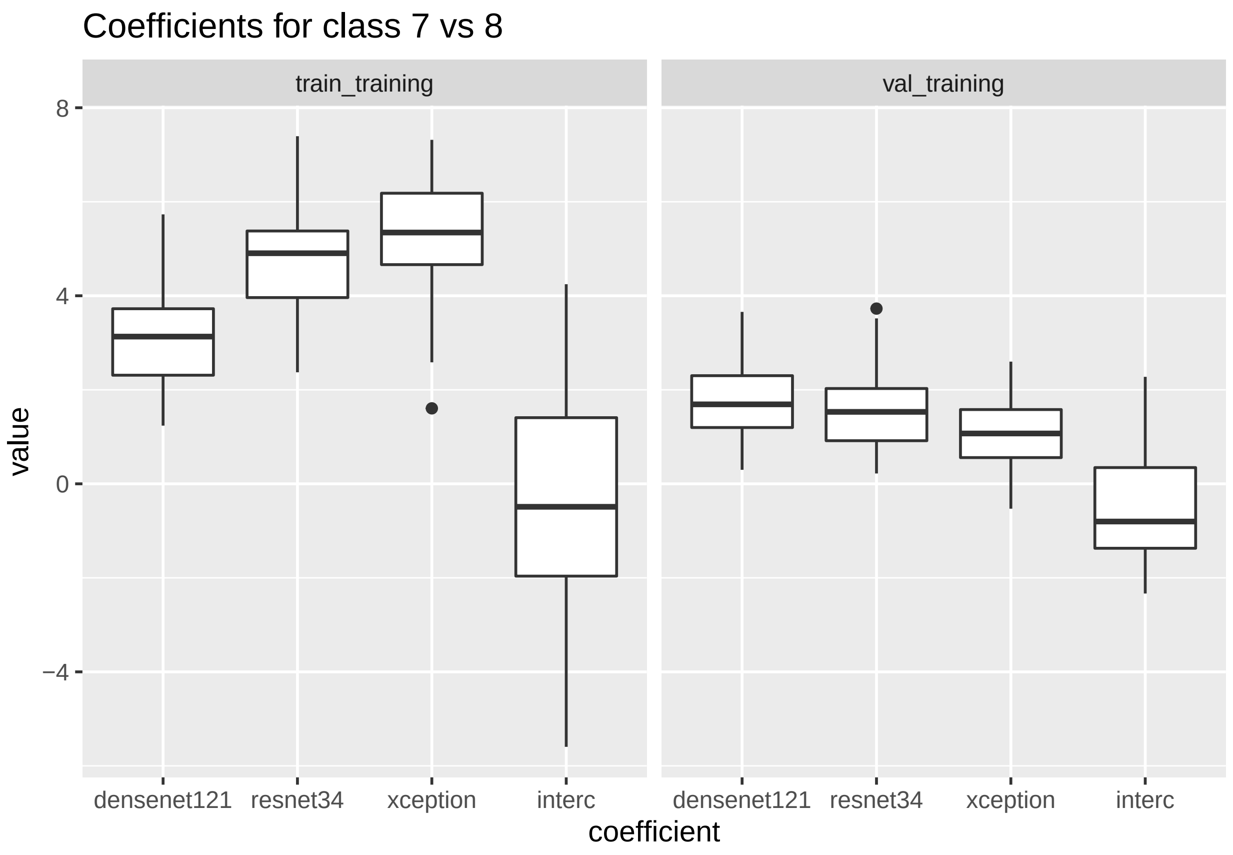


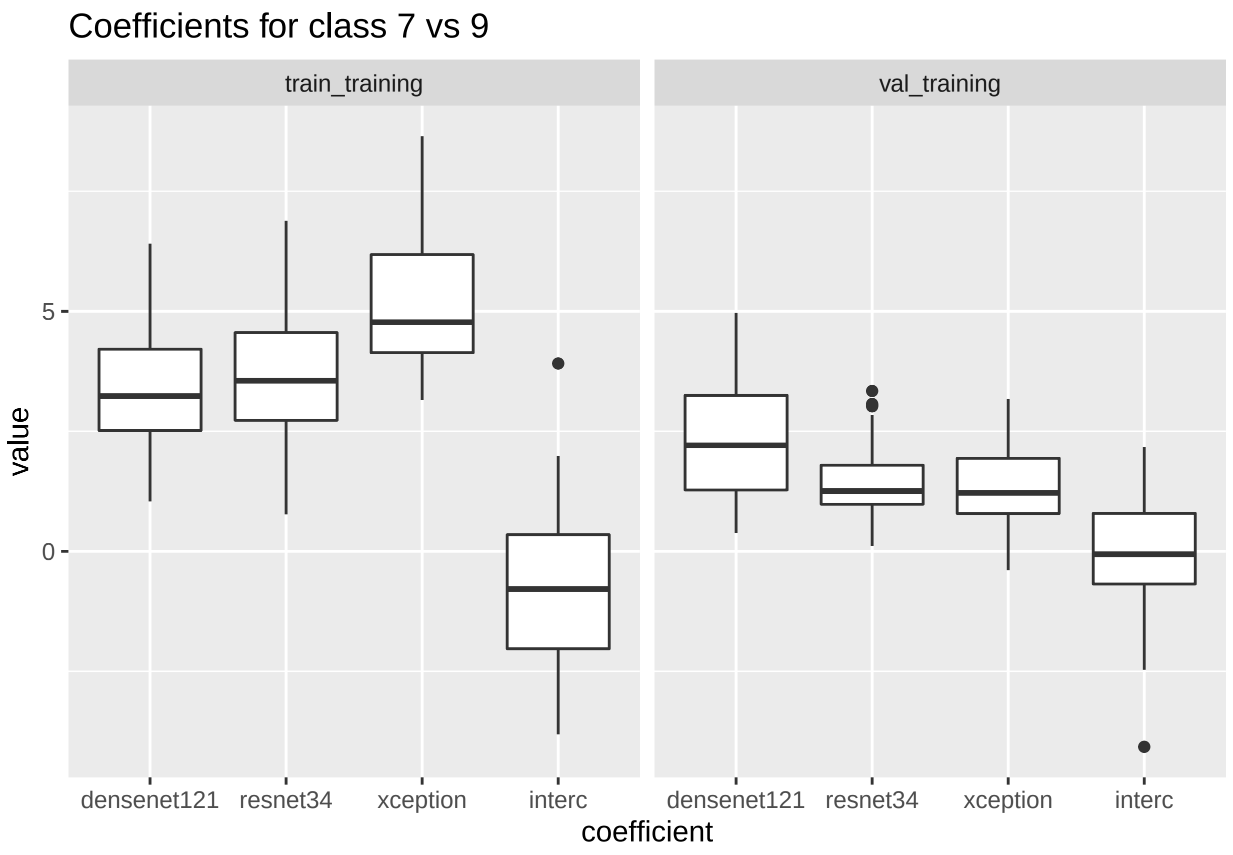
Coefficients for class 6 vs 8
train_training
val_training
12
8
4
0
−4
value
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

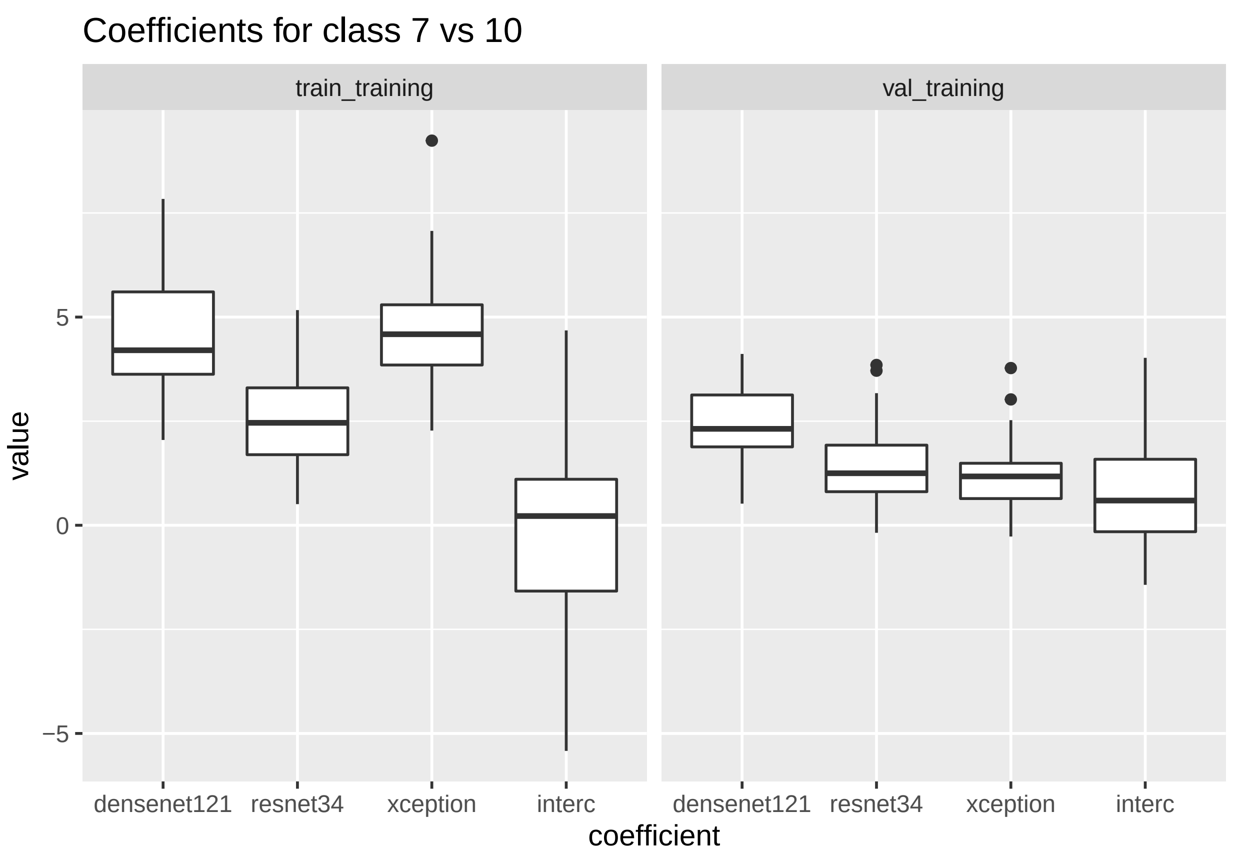

Coefficients for class 6 vs 9
train_training
val_training
value
coefficient
7.5
5.0
2.5
0.0
−2.5
−5.0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc

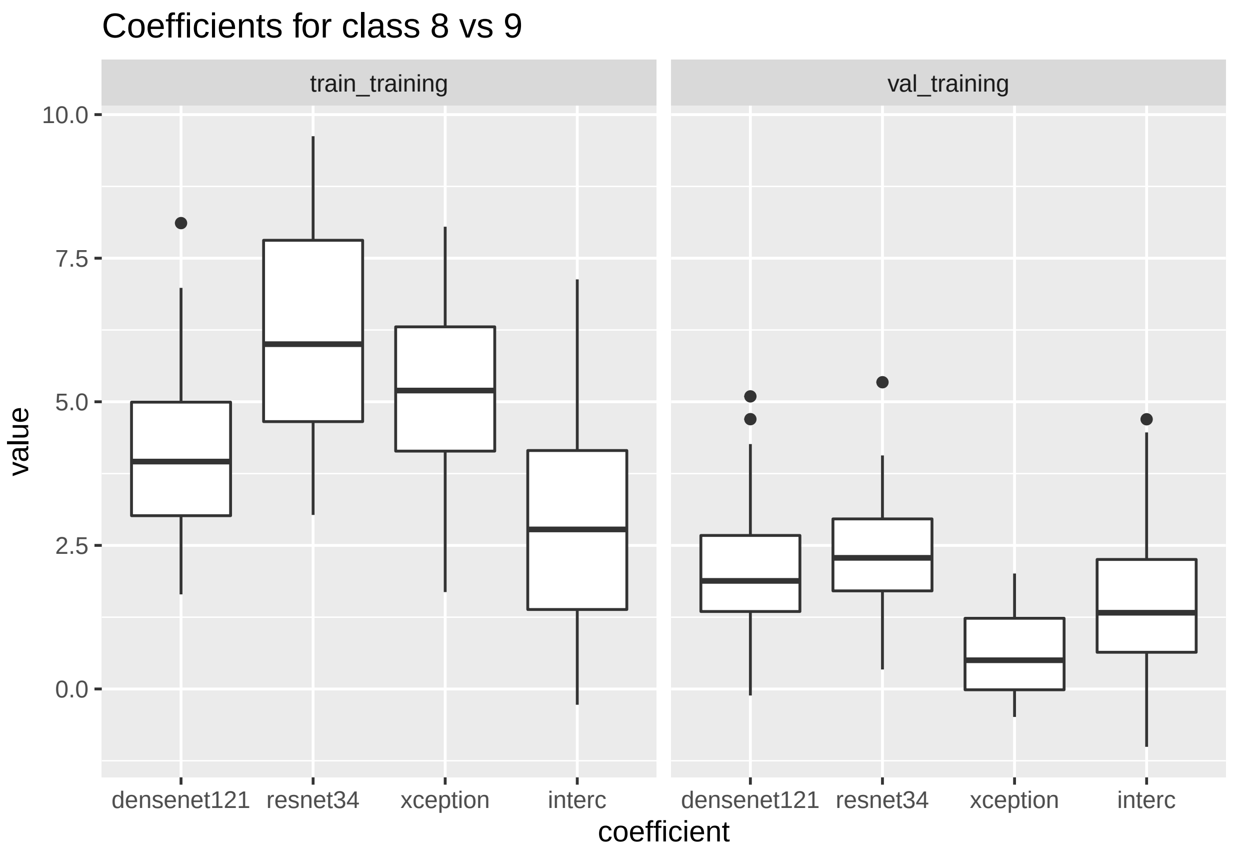


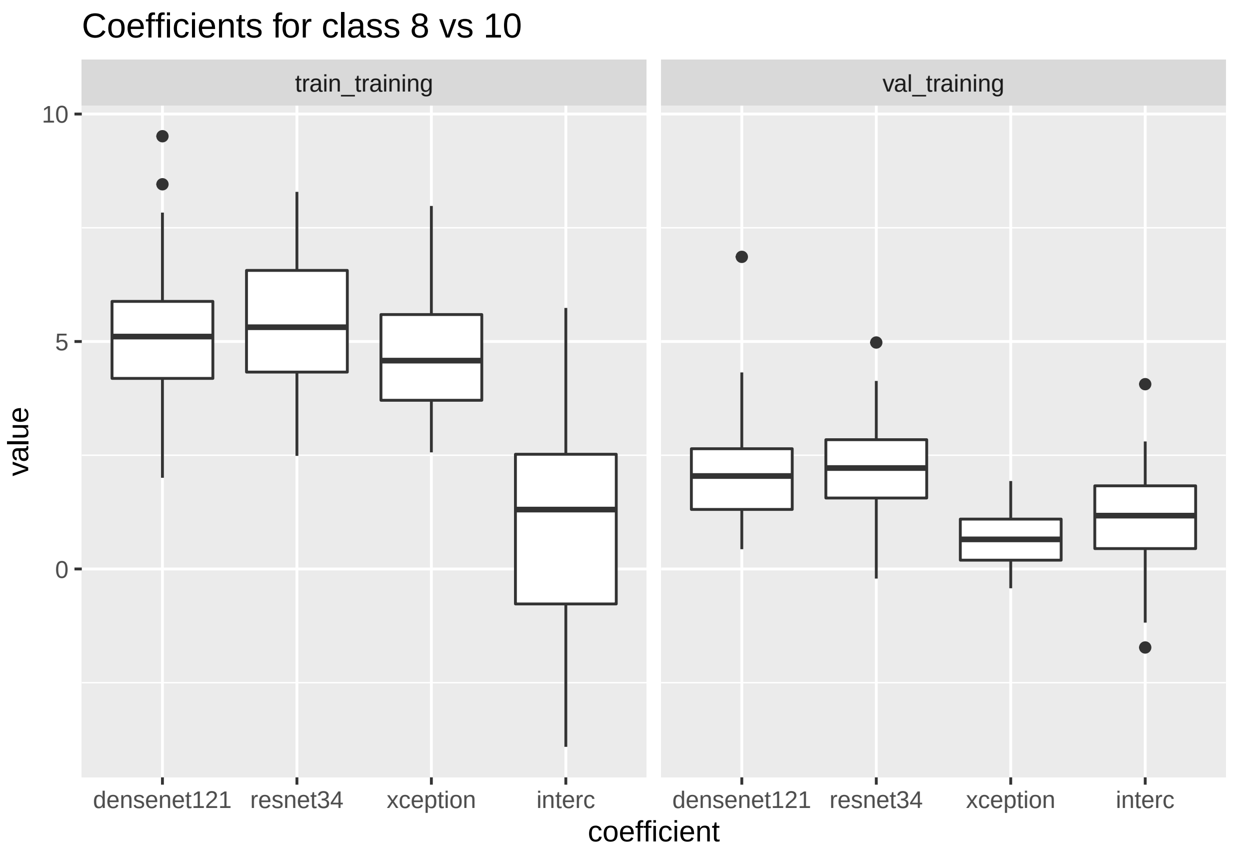
Coefficients for class 6 vs 10
train_training
val_training
8
4
0
−4
value
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient


Coefficients for class 7 vs 8
train_training
val_training
value
8
4
0
−4
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

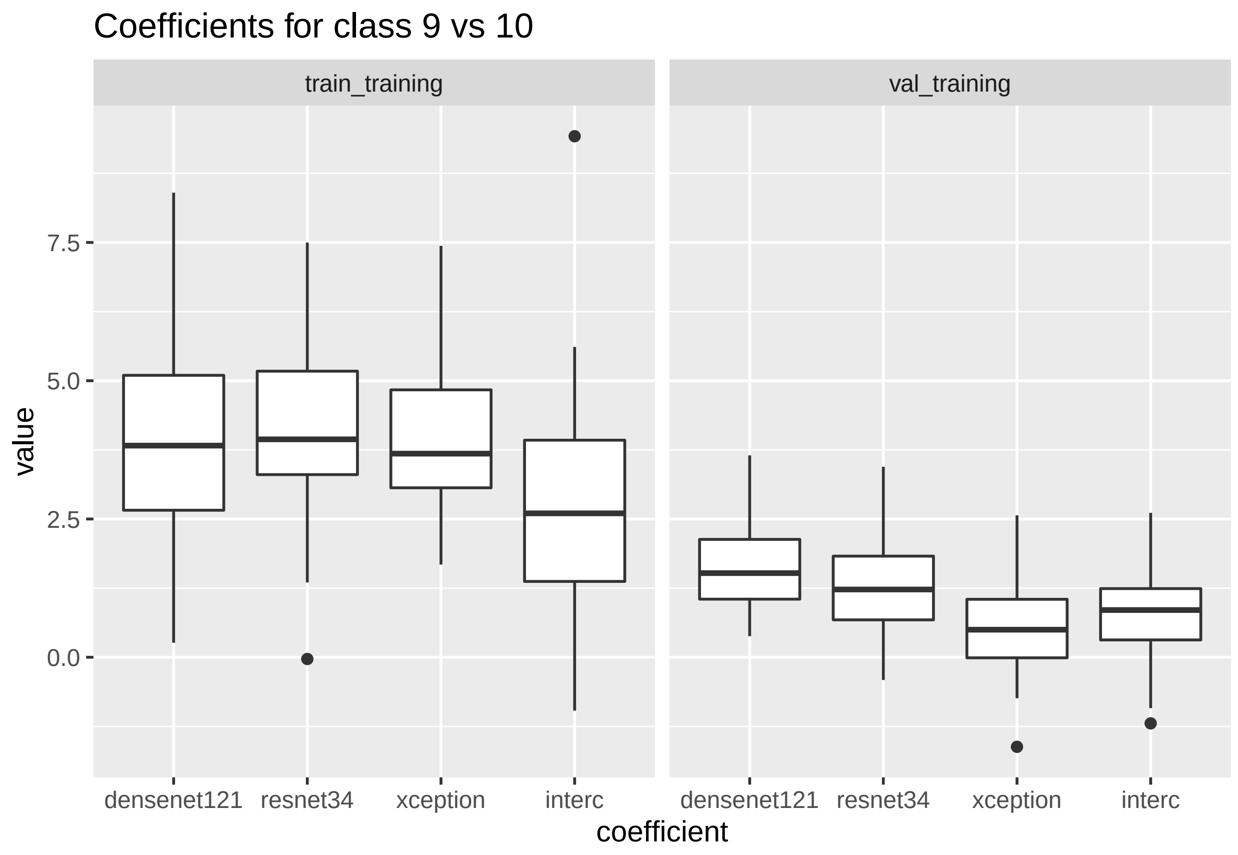

Coefficients for class 7 vs 9
train_training
val_training
value
5
0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient


Coefficients for class 7 vs 10
train_training
val_training
value
5
0
−5
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient


Coefficients for class 8 vs 9
train_training
val_training
10.0
7.5
5.0
2.5
0.0
value
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient



Coefficients for class 8 vs 10
train_training
val_training
10
5
0
value
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

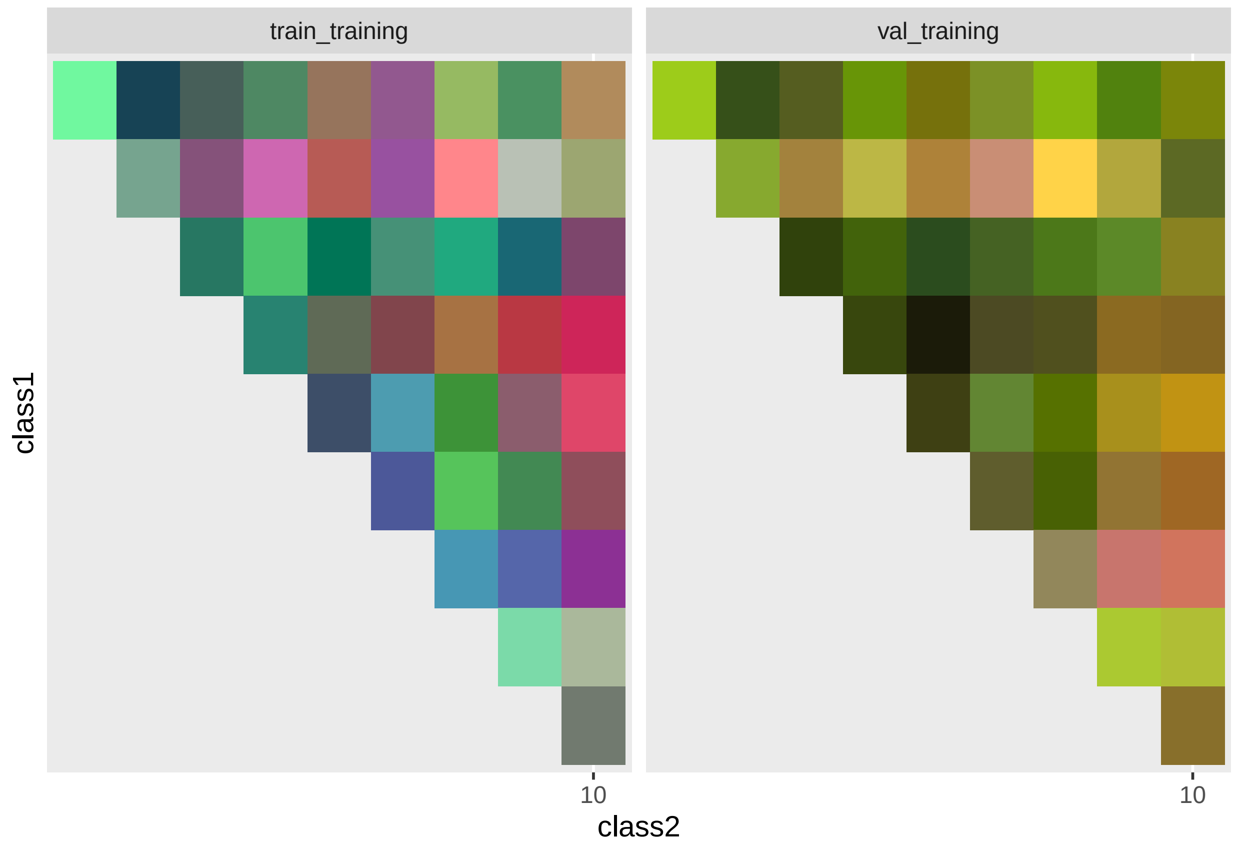


```
avg_lda_coefs <- lda_coefs %>% filter(coefficient != "interc") %>% group_by(class1, class2, precision,
```

```
## ‘summarise()‘ has grouped output by ’class1’, ’class2’, ’precision’, ’train_type’. You can override
```

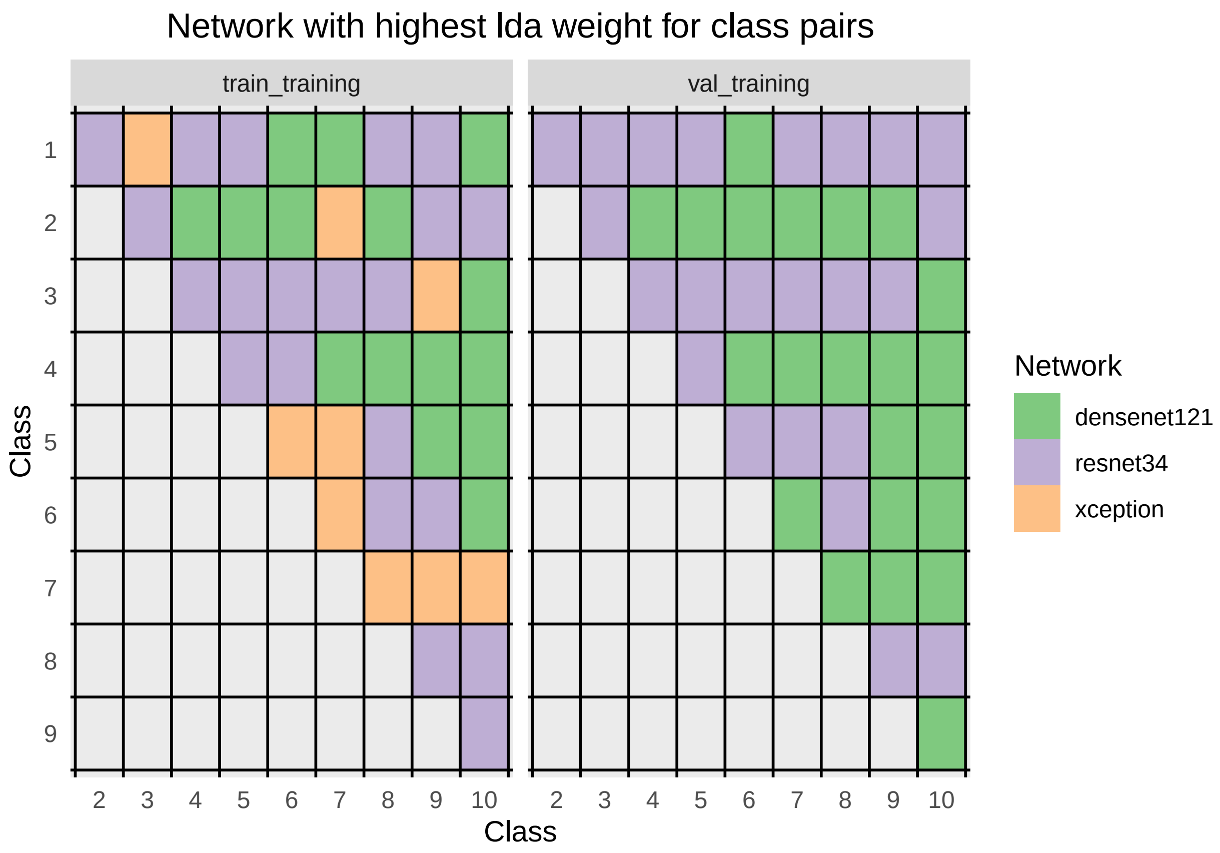
```
avg_lda_coefs_vt <- avg_lda_coefs %>% filter(train_type=="val_training")
avg_lda_coefs_tt <- avg_lda_coefs %>% filter(train_type=="train_training")
avg_lda_coefs_vt$value <- avg_lda_coefs_vt$value - min(avg_lda_coefs_vt$value)
avg_lda_coefs_vt$value <- avg_lda_coefs_vt$value / max(avg_lda_coefs_vt$value)
avg_lda_coefs_tt$value <- avg_lda_coefs_tt$value - min(avg_lda_coefs_tt$value)
avg_lda_coefs_tt$value <- avg_lda_coefs_tt$value / max(avg_lda_coefs_tt$value)
avg_lda_coefs <- rbind(avg_lda_coefs_vt, avg_lda_coefs_tt)
avg_lda_c_w <- pivot_wider(avg_lda_coefs, names_from = coefficient, values_from = value)
avg_lda_c_w[, c("class1", "class2")] <- lapply(avg_lda_c_w[, c("class1", "class2")], as.factor)
avg_lda_c_w$top_net <- factor(c("densenet121", "resnet34", "xception")[max.col(as.matrix(avg_lda_c_w[,
```

```
raster_plot <- ggplot(avg_lda_c_w) +
  geom_tile(aes(x=class2, y=class1, fill=rgb(densenet121, resnet34, xception))) +
  scale_y_discrete(limits=rev, breaks=seq(0,classes, 10)) + scale_x_discrete(breaks=seq(0,classes, 10))
raster_plot
```



```
coefs_grid <- ggplot(avg_lda_c_w, aes(x=class2, y=class1, fill=top_net)) +
  geom_raster() +
  scale_fill_brewer(type="qual") +
  facet_wrap(~train_type) +
  scale_y_discrete(limits=rev) +
  geom_vline(xintercept=seq(-0.5, 9.5, 1.0)) +
  geom_hline(yintercept=seq(-0.5, 9.5, 1.0)) +
  guides(fill=guide_legend(title="Network")) +
  xlab("Class") +
  ylab("Class") +
  ggtitle("Network with highest lda weight for class pairs") +
  theme(plot.title = element_text(hjust = 0.5),
        axis.ticks = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())

coefs_grid
```


Network with highest lda weight for class pairs
train_training
val_training
Class
1
2
3
4
5
6
7
8
9
Class
2
3
4
5
6
7
8
9
10
Network
densenet121
resnet34
xception