Outputs inspection half CIFAR10

```r
library(ggplot2)
```

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

```r
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
```

```r
library(tidyr)
```

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

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

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

```r
library(LDATS)
```

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

```r
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(ggVennDiagram)
```

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

```
library(reticulate)
```

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

```
library(abind)
```

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

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

source("utils.R")
```

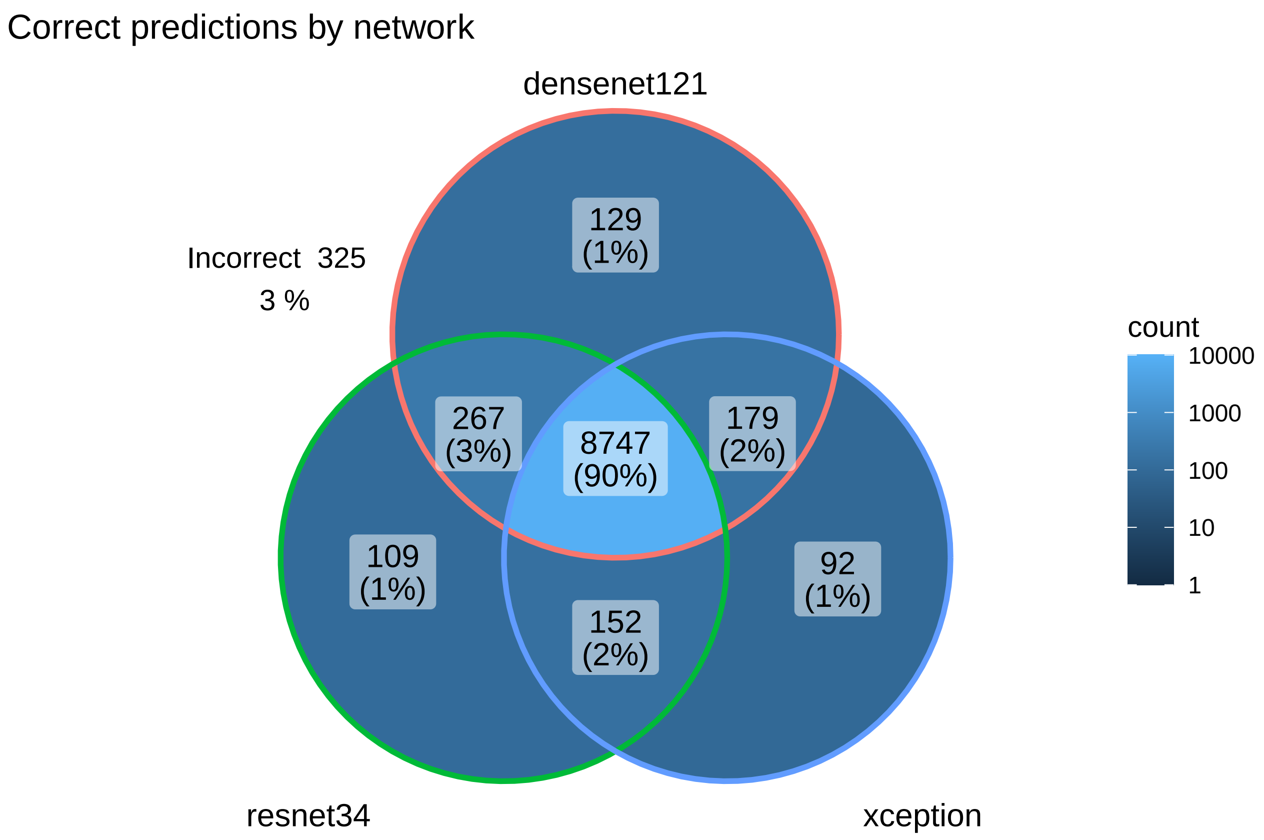
```
## Warning: package 'hash' was built under R version 4.0.5

## 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:ggVennDiagram':
##
##     circle

## 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"))
```

```r
sort_ind <- function(lst)
{
  return(sort(lst, index.return=TRUE, decreasing=TRUE)$ix)
}
nets_test_top_indices <- apply(X=nets_outputs$test_outputs, MARGIN=c(1, 2, 3), FUN=sort_ind)[1, , , ]
r_n <- length(repls)
samples_n <- dim(nets_outputs$test_labels)[2]
nets_n <- length(nets_outputs$networks)
test_labs <- nets_outputs$test_labels + 1
dim(test_labs) <- c(r_n, 1, samples_n)
test_labs <- aperm(abind(array(rep(aperm(test_labs, perm=c(2, 1, 3)), nets_n), c(r_n, samples_n, nets_n
if (r_n == 1)
{ dim(test_labs) <- dim(test_labs)[-1] }
nets_test_cor_preds <- test_labs == nets_test_top_indices
```

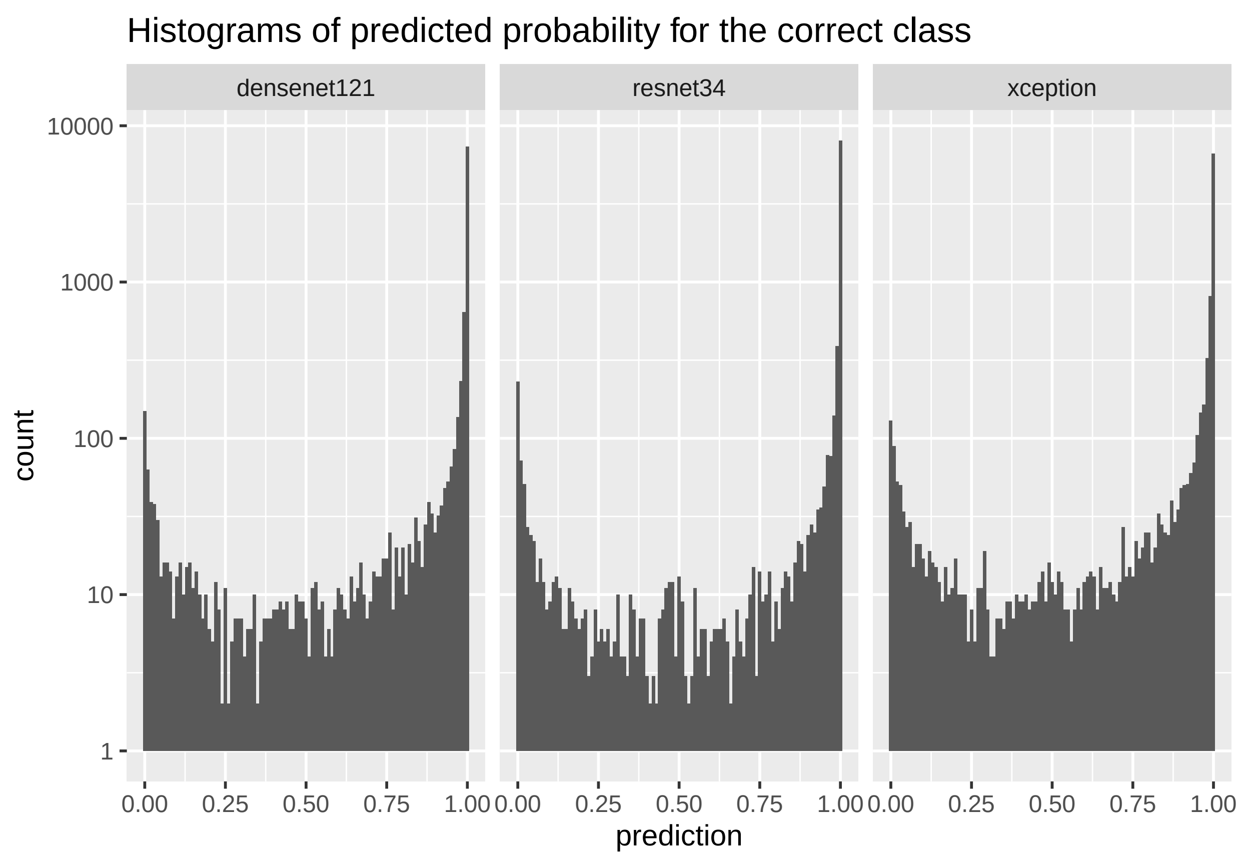
```r
nets_cor_list <- list()
incor <- 1:samples_n
for (ni in 1:nets_n)
{
  cor_list <- which(nets_test_cor_preds[ni, ])
  nets_cor_list[[nets_outputs$networks[ni]]] = cor_list
  incor <- setdiff(incor, cor_list)
}
incor_n <- length(incor)
venn_diag <- ggVennDiagram(nets_cor_list) + scale_fill_gradient(trans="log10", name="count", limits=c(1
  annotate(geom="text", x=-4, y=5, label=paste("Incorrect ", incor_n, "\n", round(incor_n / samples_n *
    ggtitle("Correct predictions by network") +
    scale_x_continuous(limits=c(-8, 10))
print(venn_diag)
```



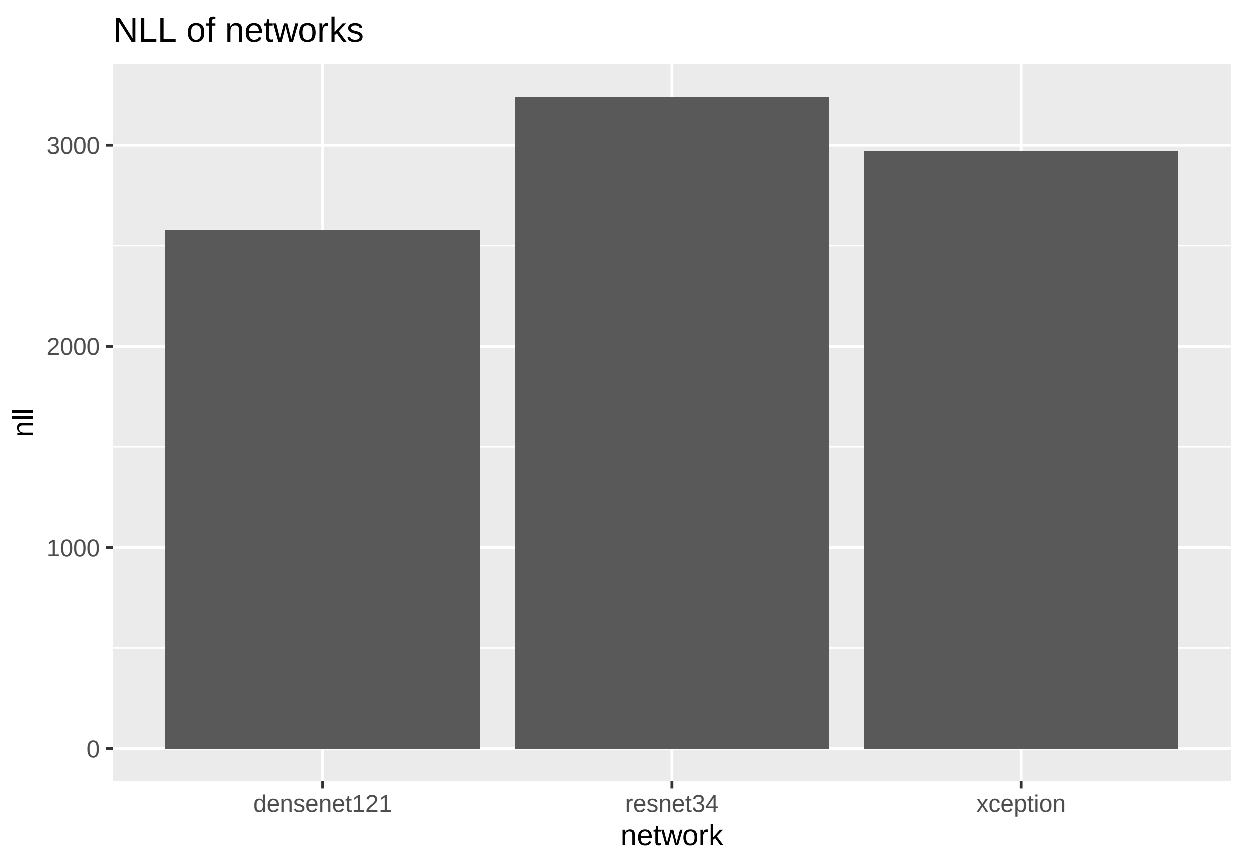
Compared to training the networks on almost complete CIFAR 10 training set, in this case all networks are correct in fewer samples. Otherwise the observations hold.

```
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
  ggtitle("Histograms of predicted probability for the correct class") + facet_wrap(~network) + scale_y
nets_cor_preds_histo
```

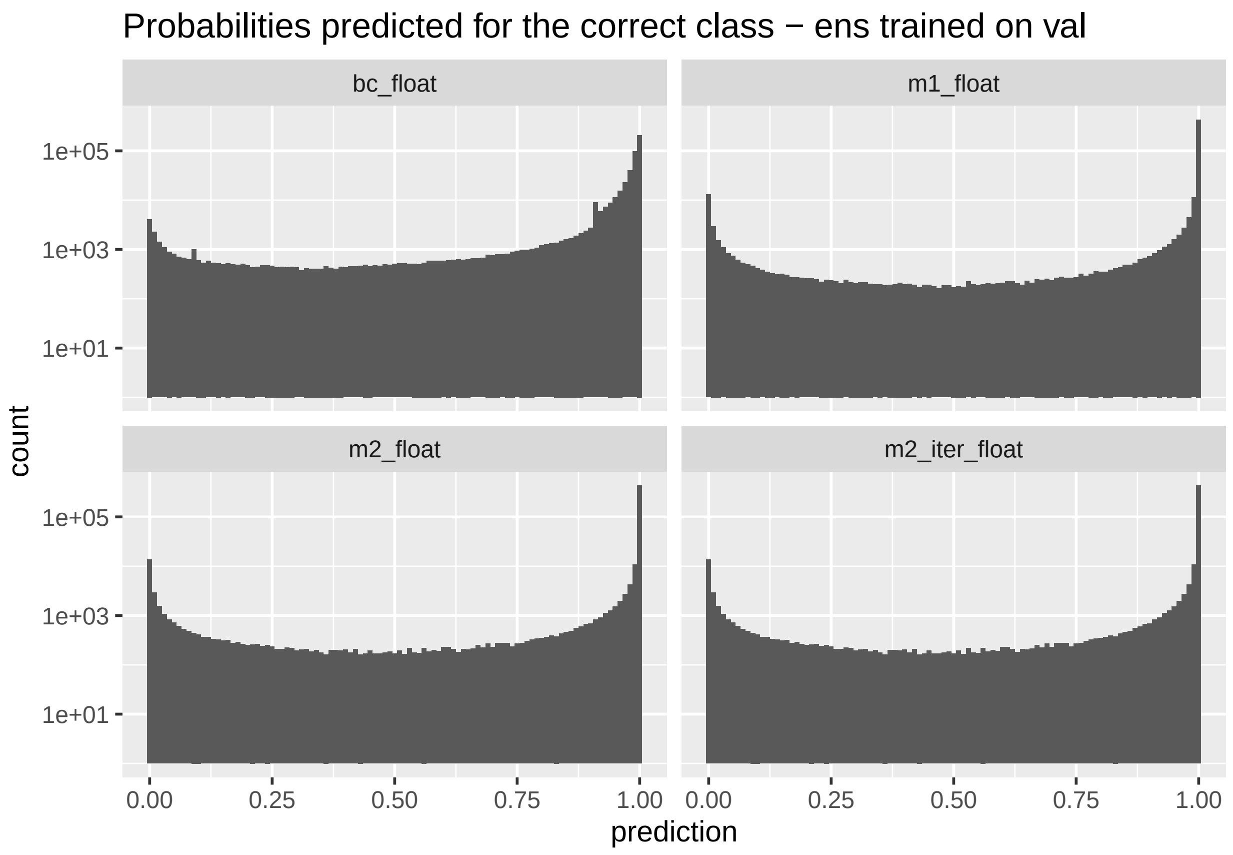


```
networks_nll <- ggplot(data=net_results) + geom_bar(mapping=aes(x=network, y=nll), stat="identity") + gg
networks_nll
```



```
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), b
val_ens_cor_preds_histo
```

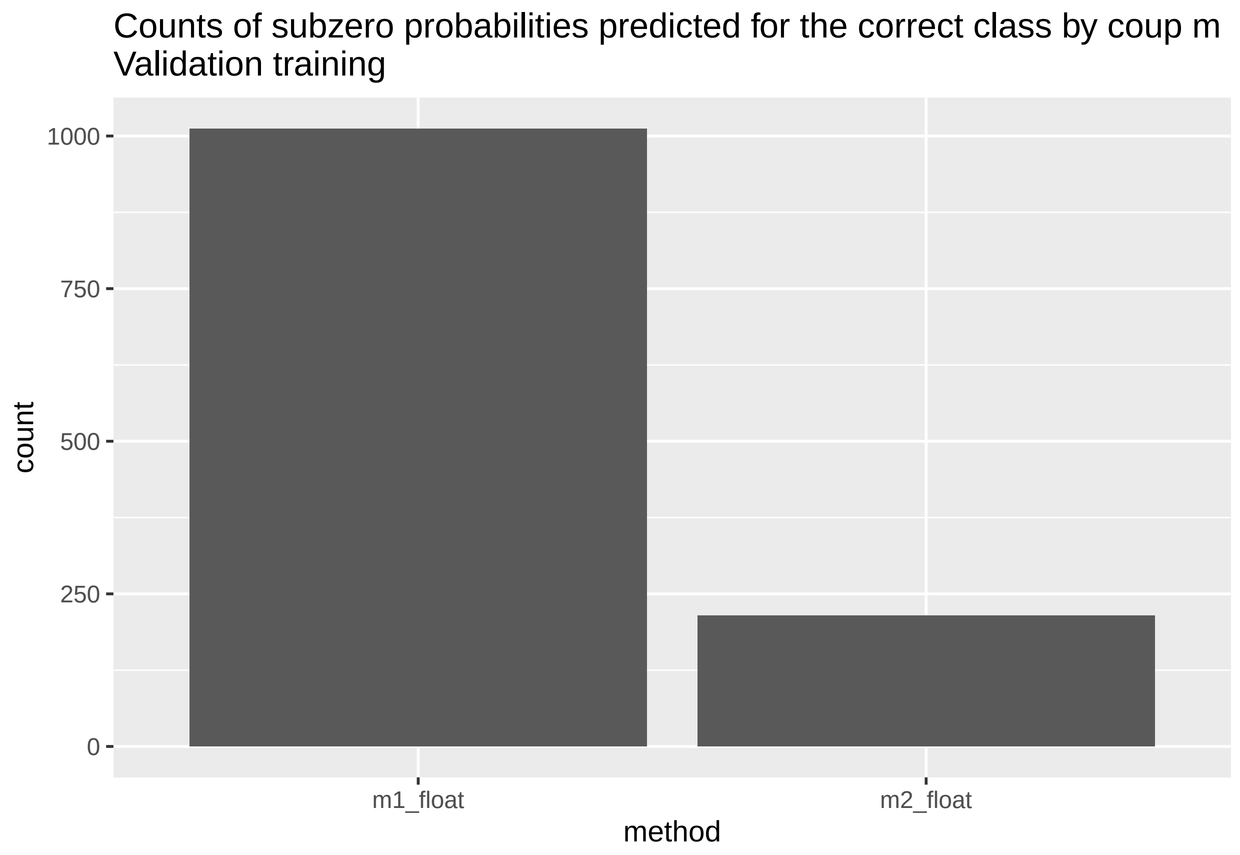


Coupling method bc produces fewer probabilities falling into the lowest bin for the correct class than m1 and m2.

```
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
```

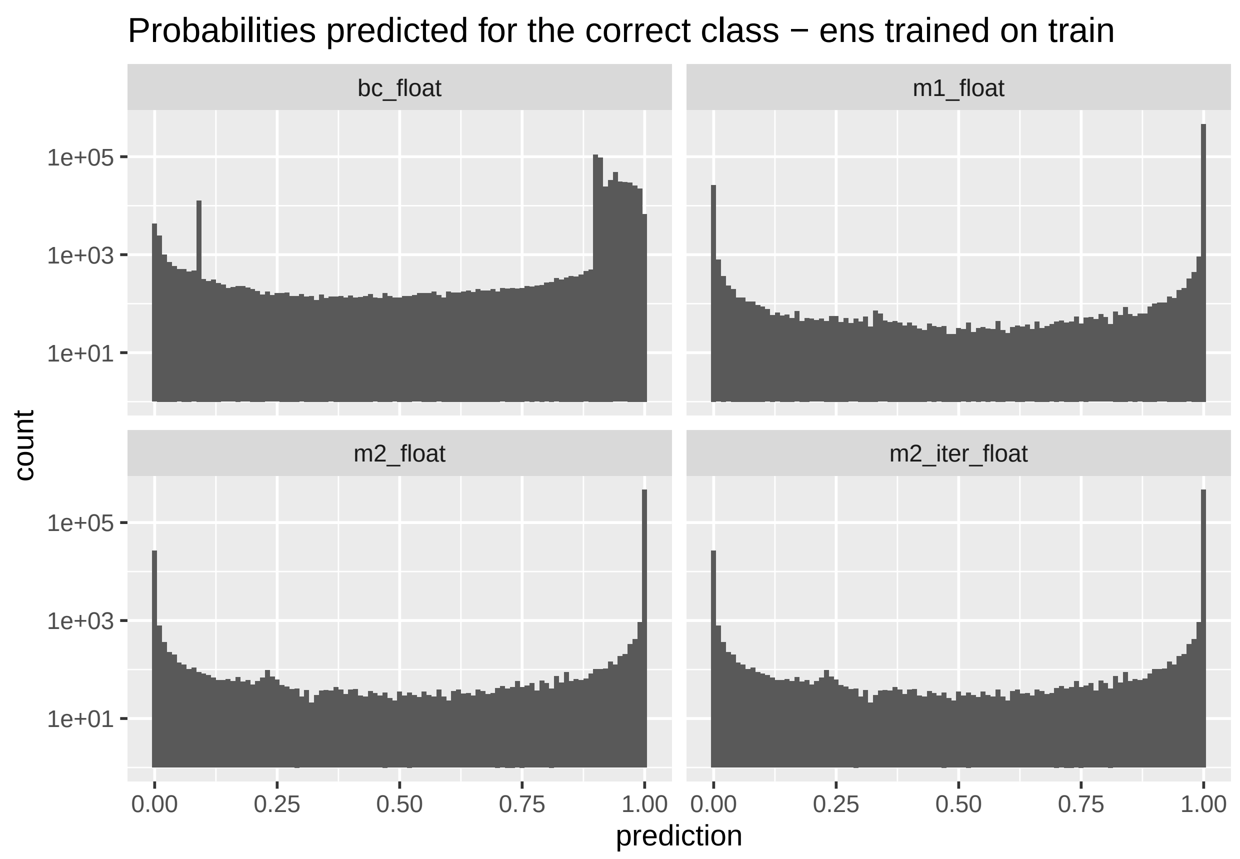
```
val_ens_zero_counts
```



m2_iter and bc didn't produce any zero probability outputs.

```
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
```

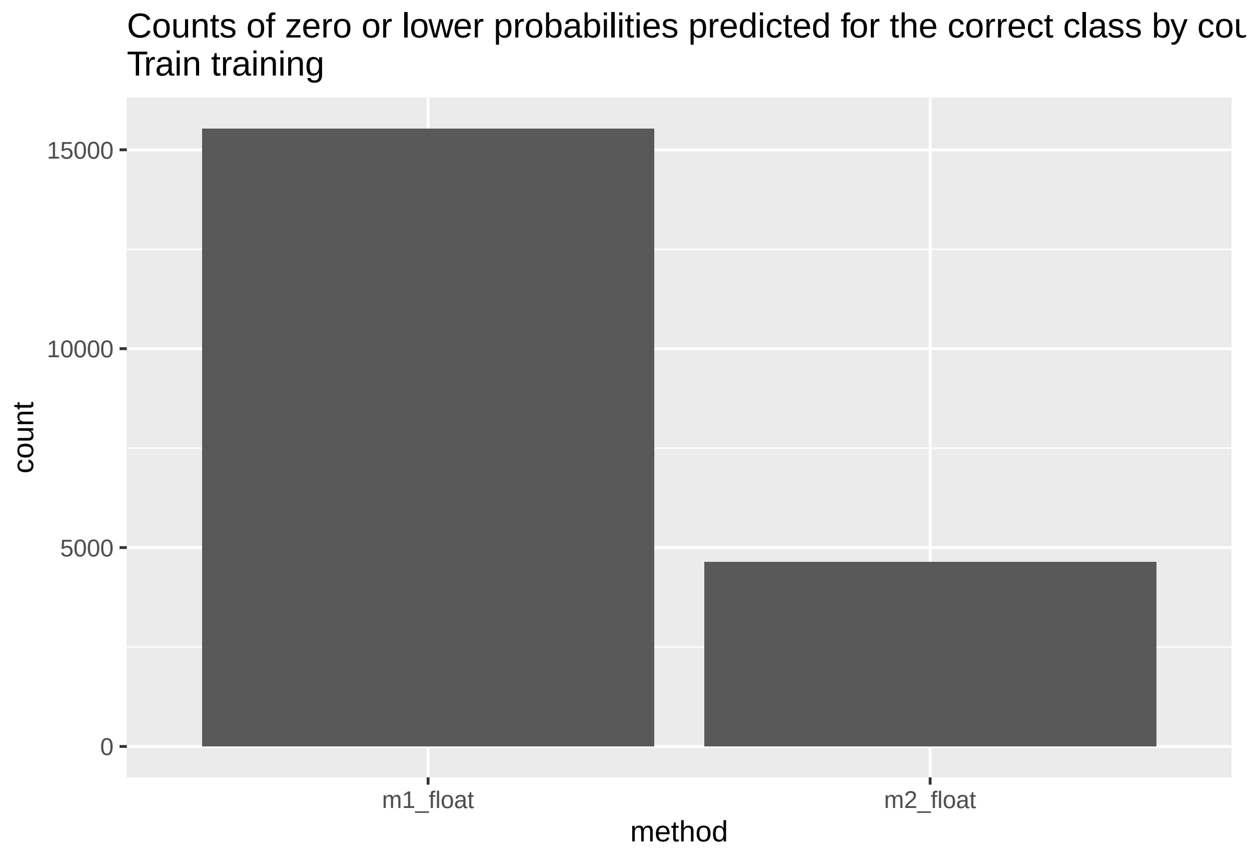


Again, coupling method bc produces fewer probabilities falling into the lowest bin for the correct class than m1 and m2.

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

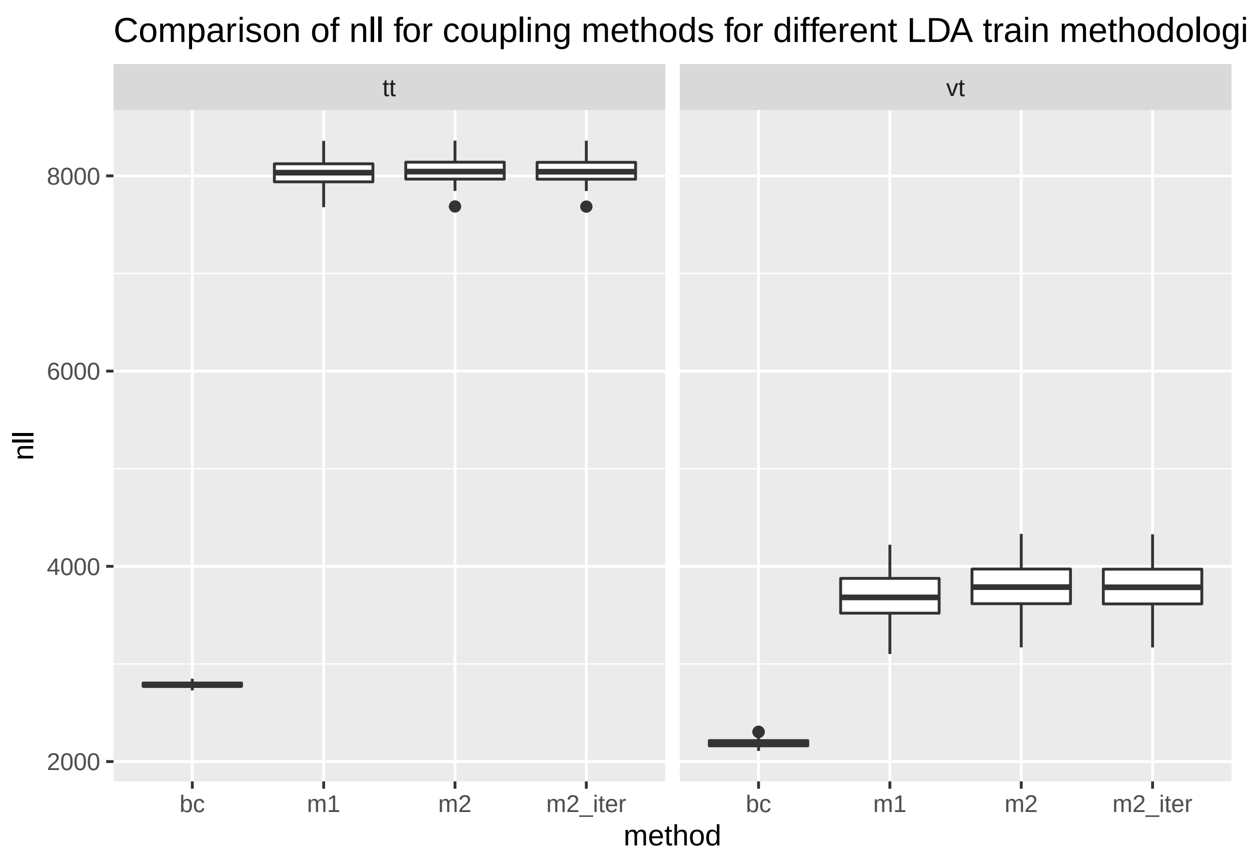
```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```

```
train_ens_zero_counts
```



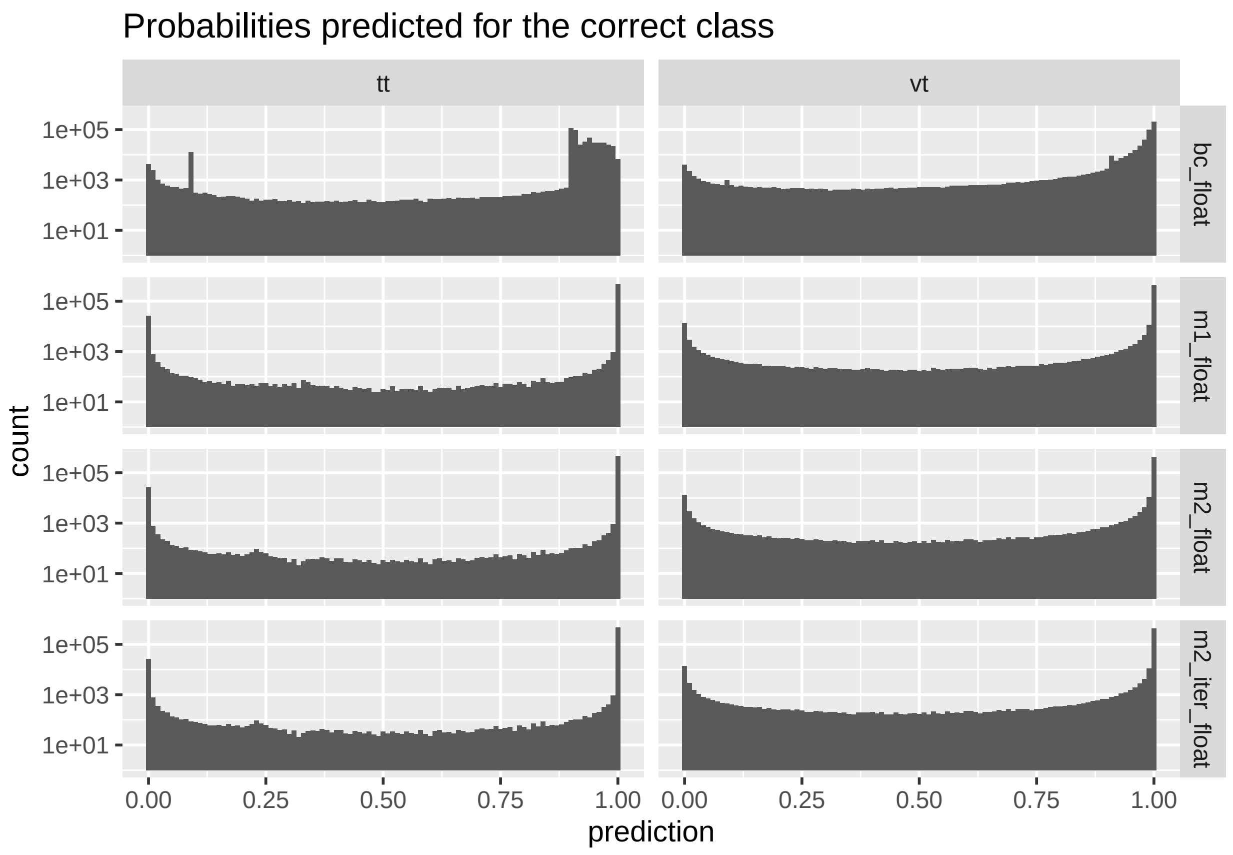
m2_iter and bc didn't produce any zero probability outputs.

```
val_ens_nll <- ggplot(data=ens_results) + geom_boxplot(mapping=aes(x=method, y=nll)) + facet_wrap(~trai
  ggtitle("Comparison of nll for coupling methods for different LDA train methodologies")
val_ens_nll
```



```
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
```



Bayes covariant coupling method produces more uniformly distributed predictions than methods m1 and m2. Also, there is a big difference in each method between ensemble trained on validation and ensemble trained on train set. Ensembles trained on validation set produce generally more uniformly distributed predictions. However, ensembles trained on training set attain statistically significantly higher accuracy. Similar results to those in visualizations_ensemble_outputs_CIF10.

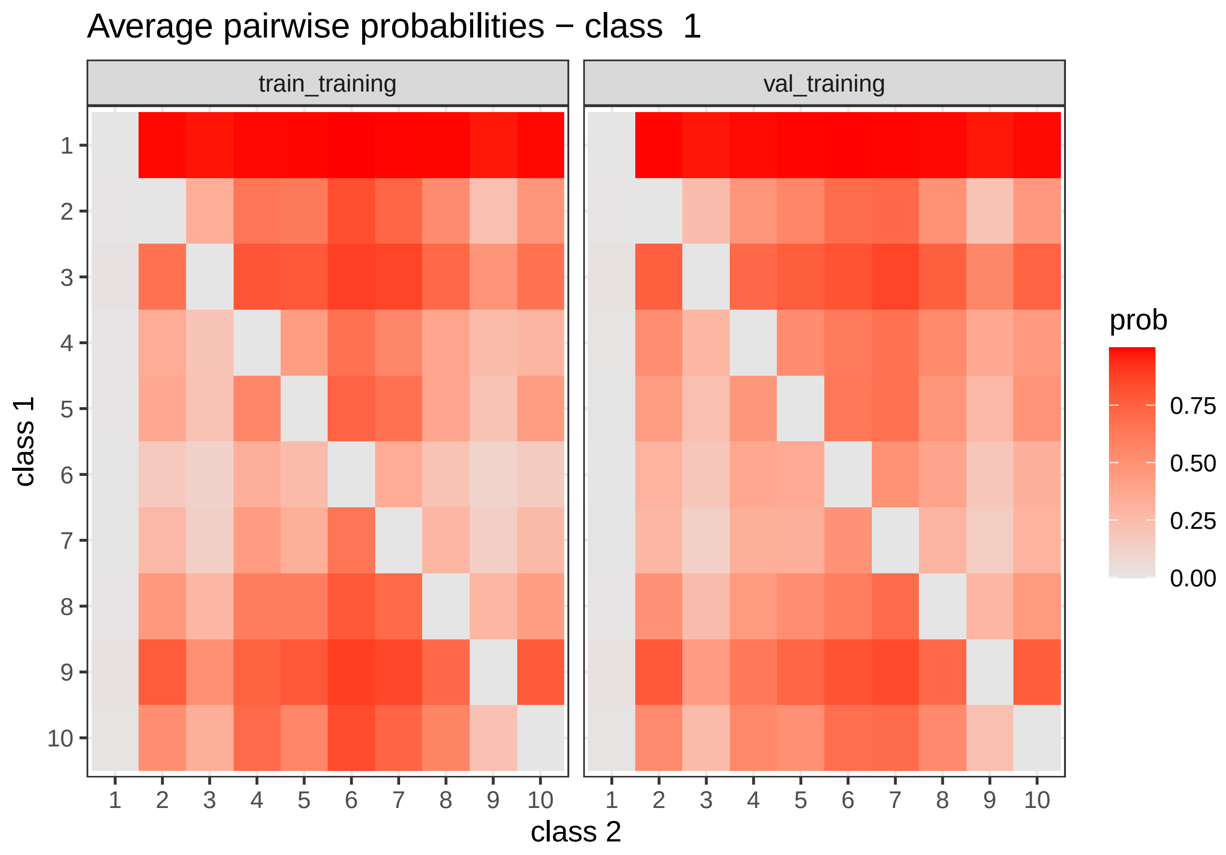
```
aggreg_Rs <- load_class_averaged_R_matrices(base_dir,  nets_outputs$test_labels[1, ], repls, folds=fold
```

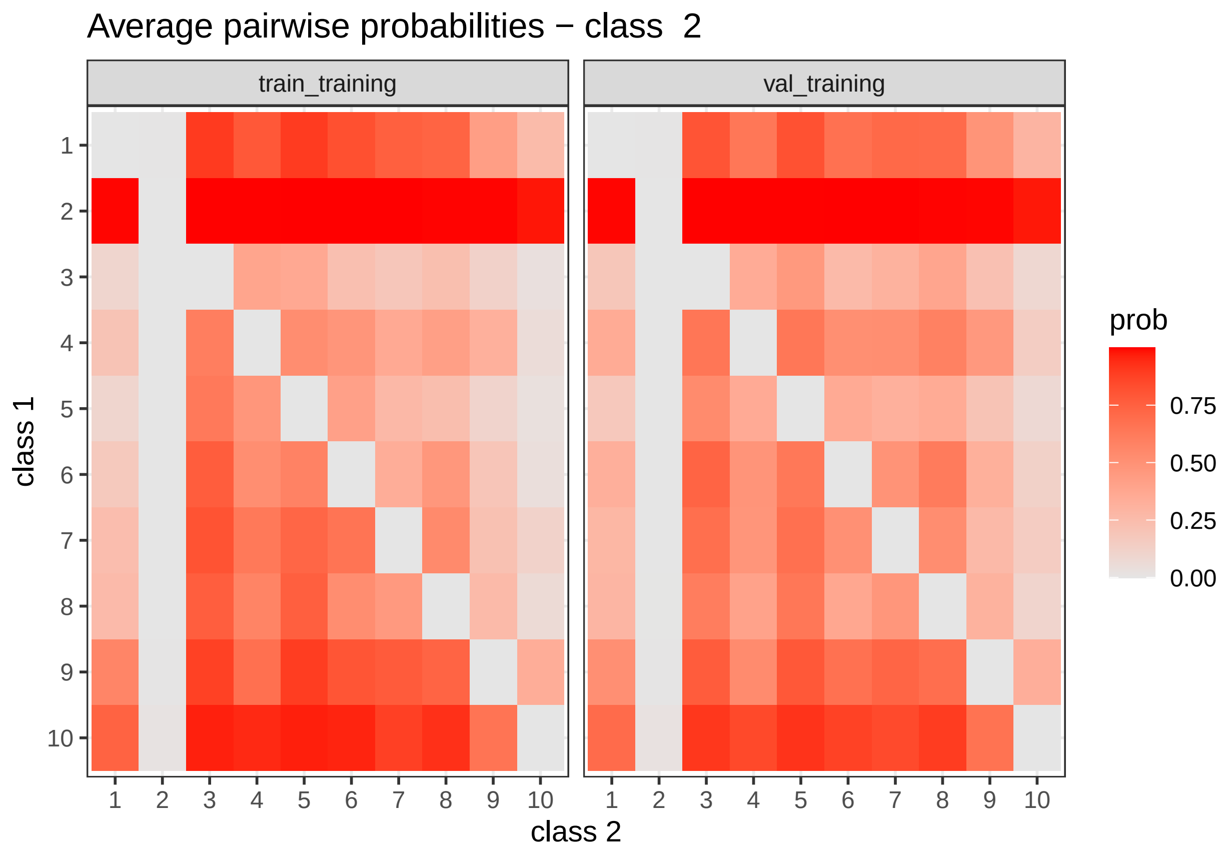
```
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘ argumen
## ‘summarise()‘ has grouped output by ’class1’, ’class2’. You can override using the ‘.groups‘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

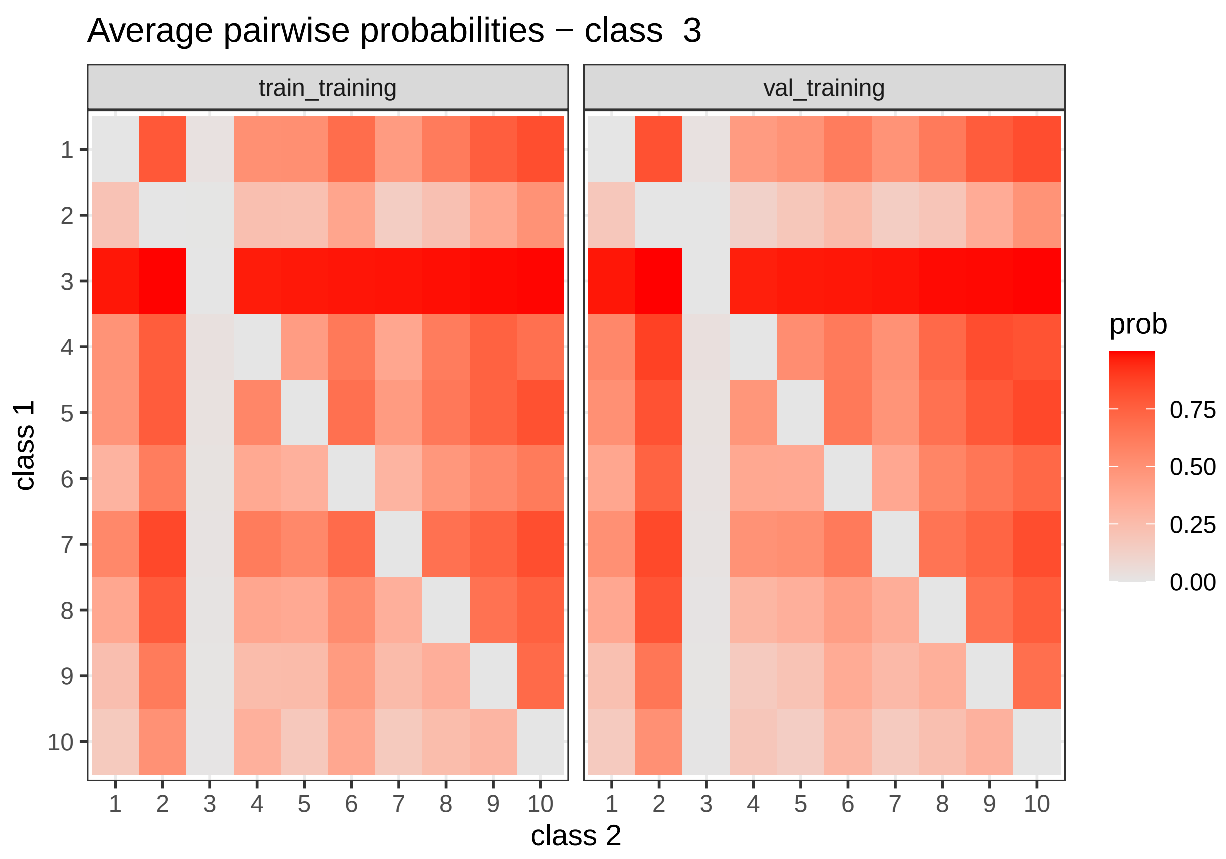
```
## `summarise()` has grouped output by 'precision', 'train_type', 'class1', 'class2'. You can override
```

```r
df_aggr_Rs_diff <- aggreg_Rs %>% pivot_wider(names_from = train_type, values_from = prob) %>% mutate(va
```

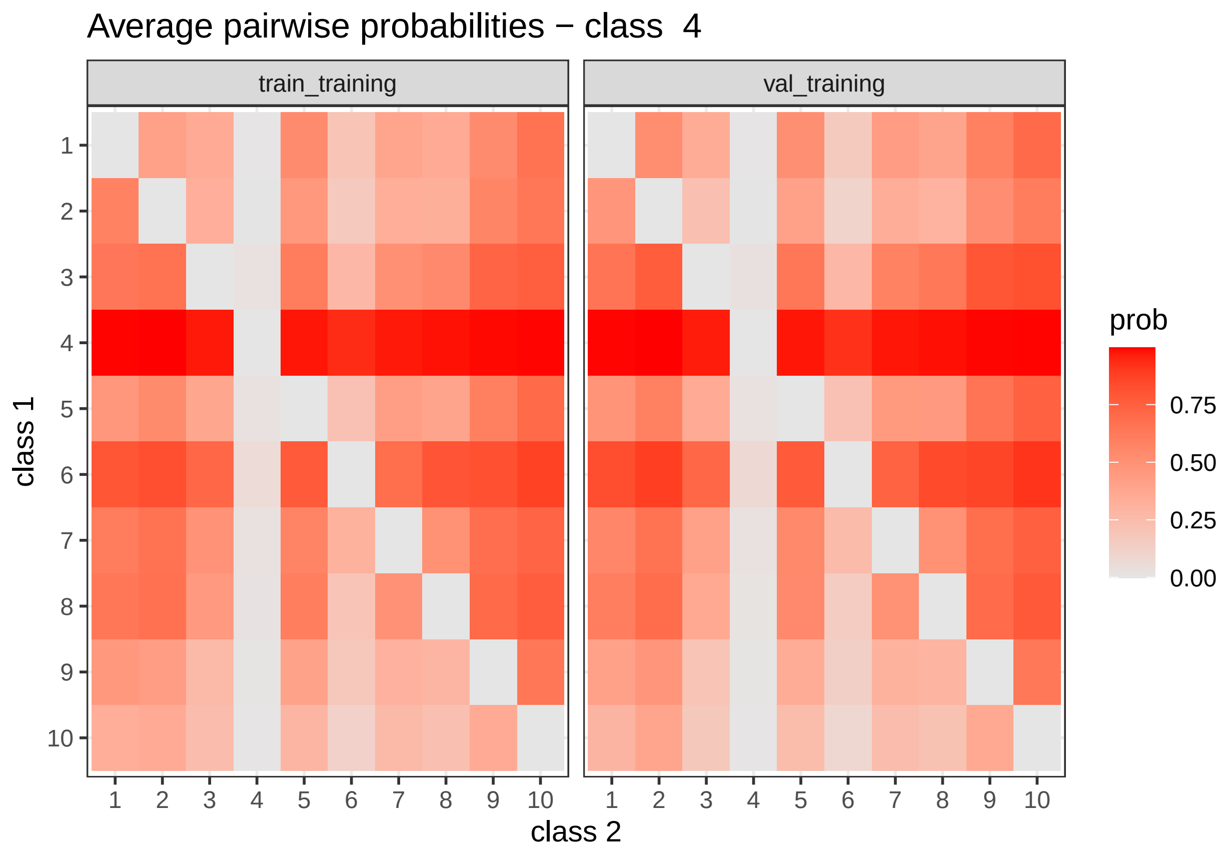
```r
for (cls in 1:classes)
{
  cur_class_Rs <- aggreg_Rs %>% filter(class == cls)
  plot_cls <-  ggplot(cur_class_Rs, aes(x = class2, y = class1)) +
    geom_raster(aes(fill=prob)) +
    facet_wrap(~train_type) +
    scale_fill_gradient(low="grey90", high="red") +
    scale_y_discrete(limits=rev) +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

  print(plot_cls)
}
```

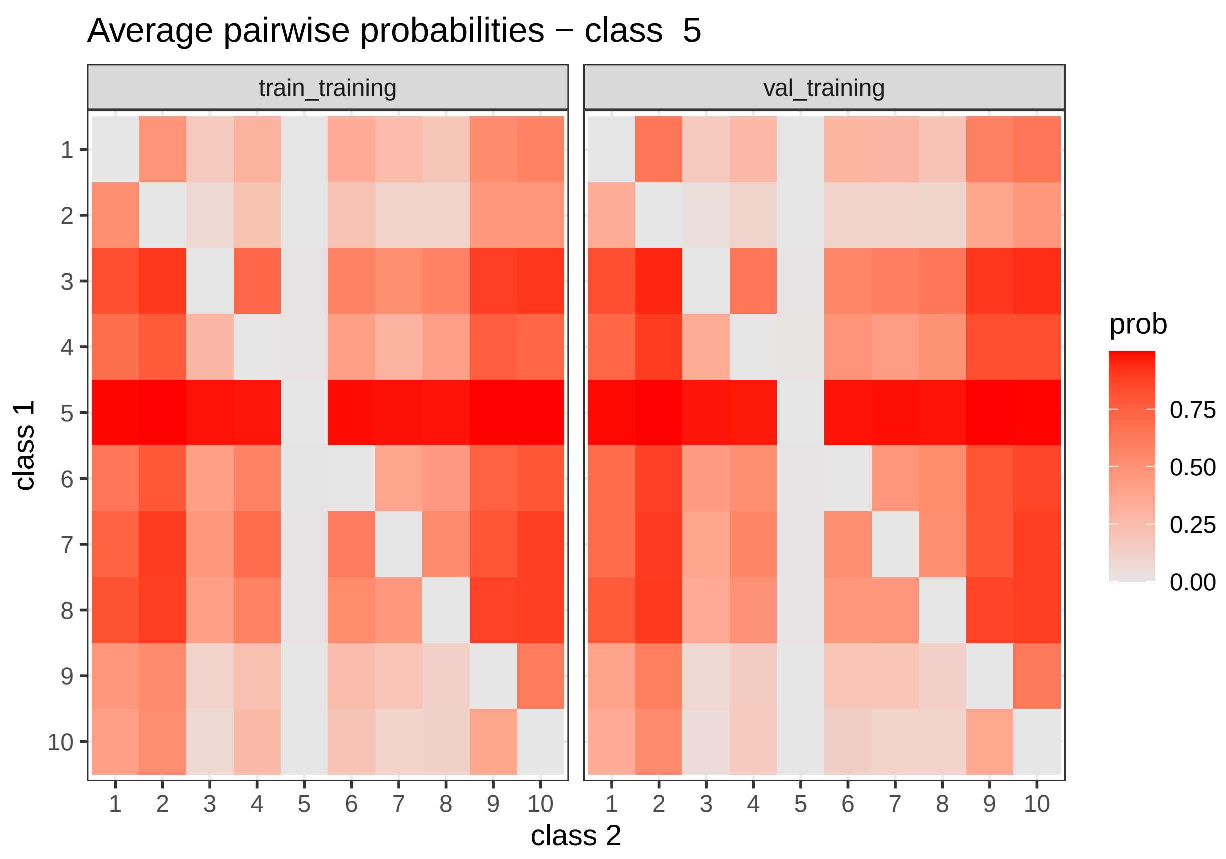

Average pairwise probabilities − class 1
train_training
val_training
class 1
class 2
1
2
3
4
5
6
7
8
9
10
prob
0.75
0.50
0.25
0.00

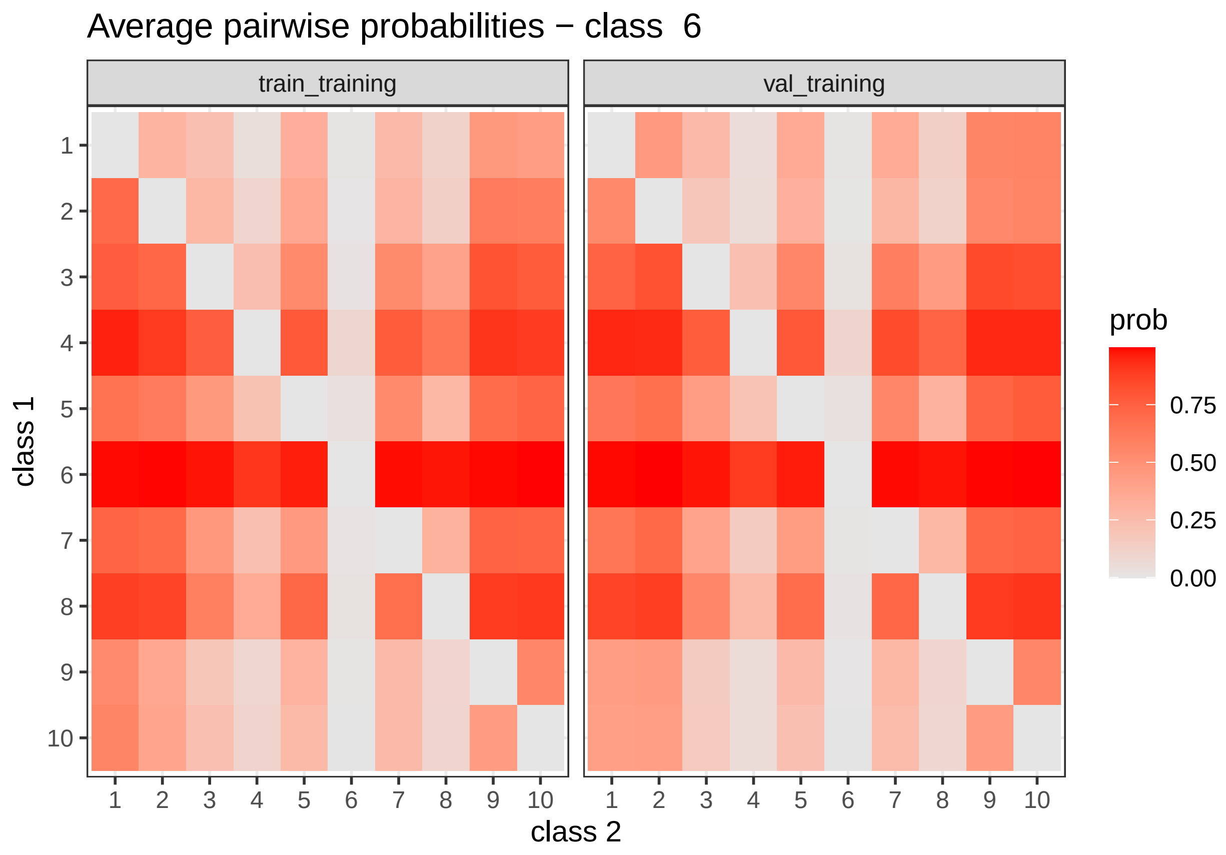

Average pairwise probabilities − class 2
train_training
val_training
class 1
class 2
1
2
3
4
5
6
7
8
9
10
prob
0.75
0.50
0.25
0.00

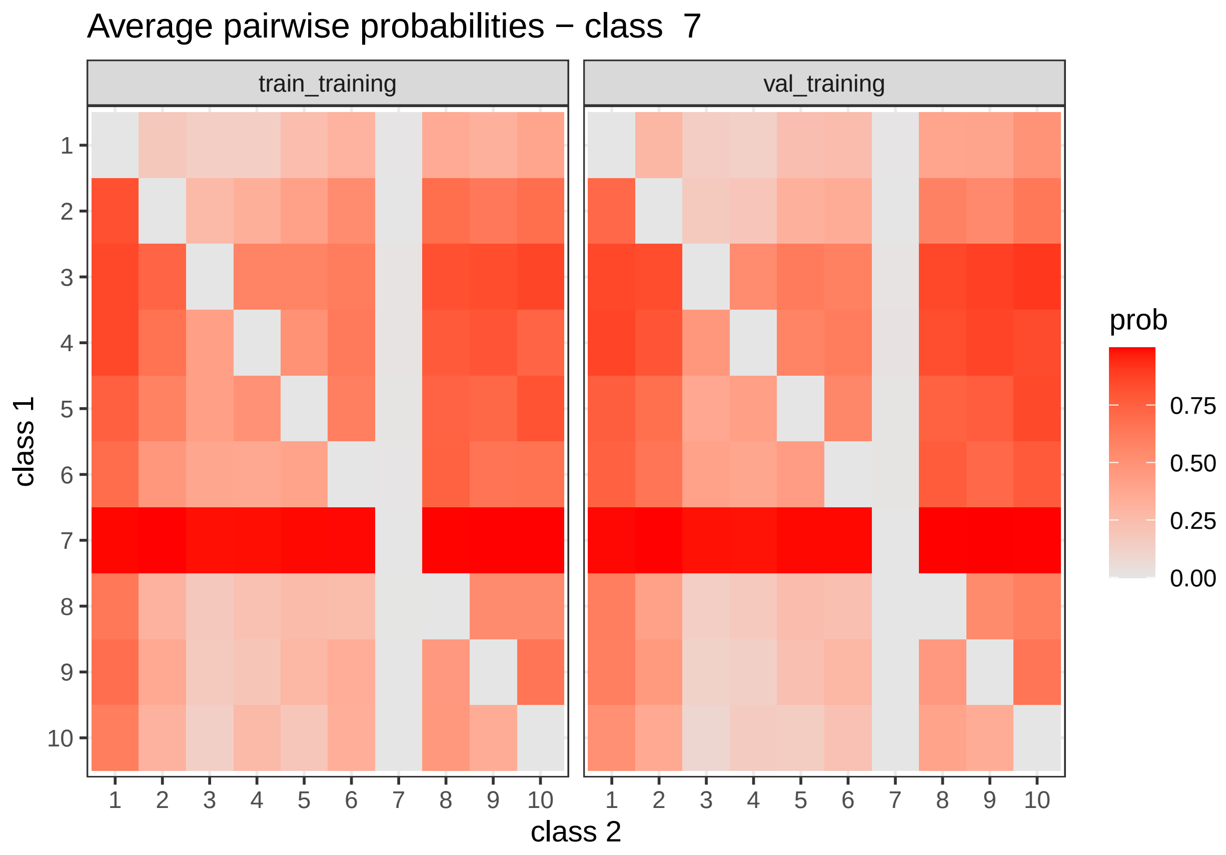


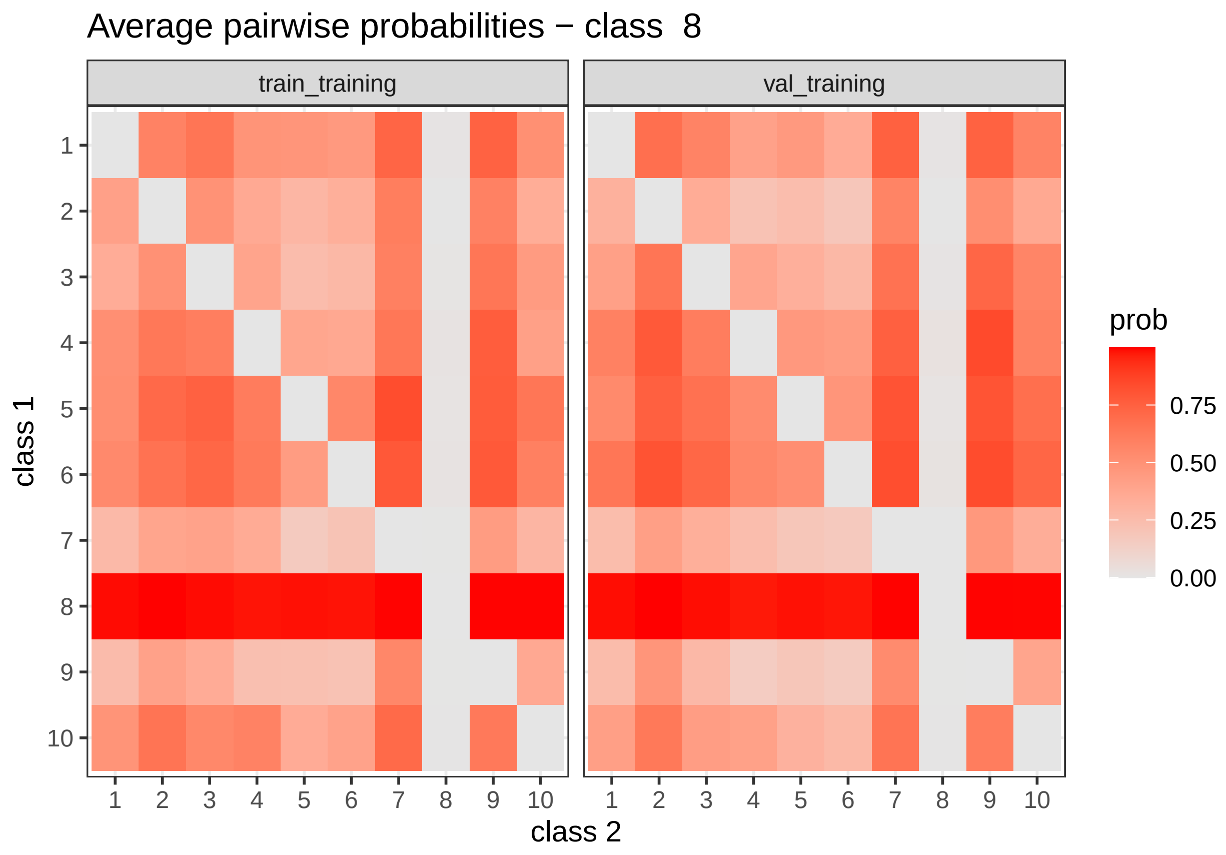
Average pairwise probabilities − class 3
train_training
val_training
class 1
class 2
1
2
3
4
5
6
7
8
9
10
prob
0.75
0.50
0.25
0.00

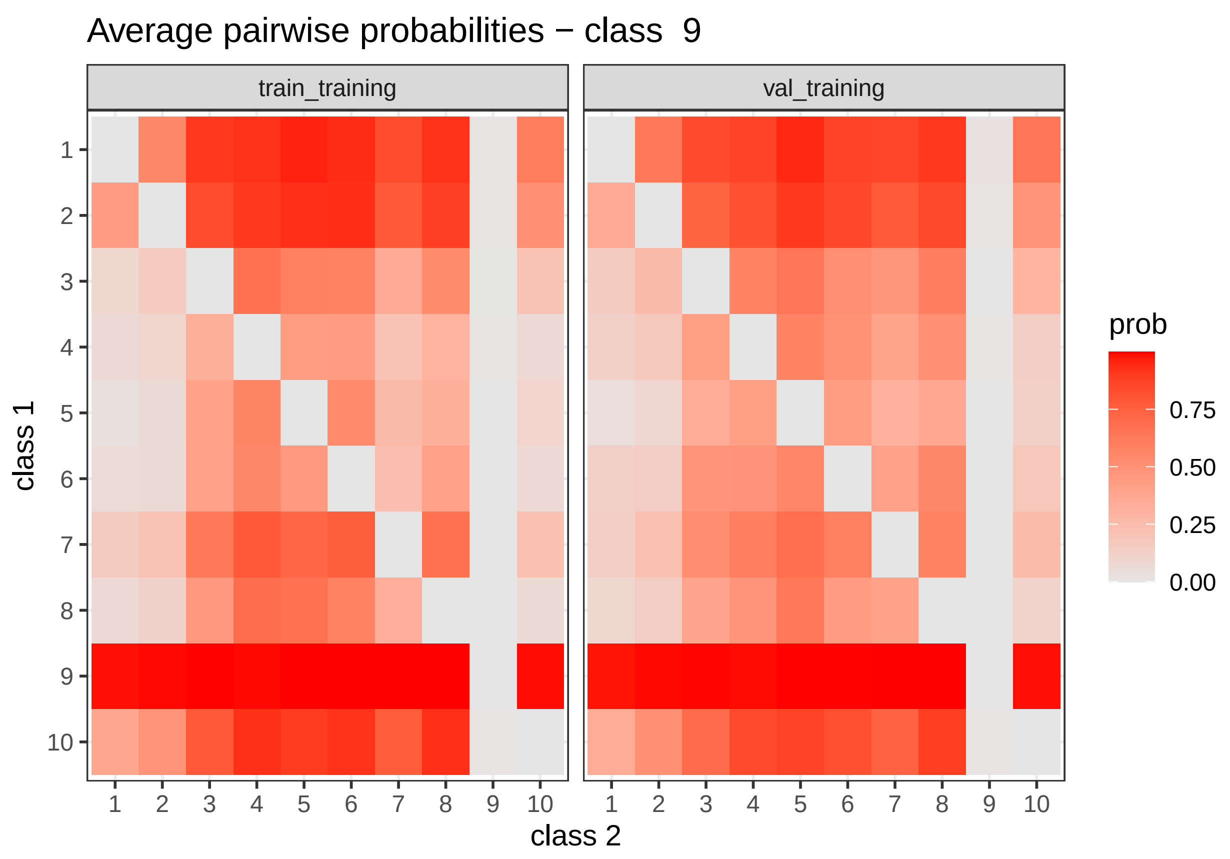

Average pairwise probabilities − class 4
train_training
val_training
prob
0.75
0.50
0.25
0.00
class 1
class 2
1
2
3
4
5
6
7
8
9
10

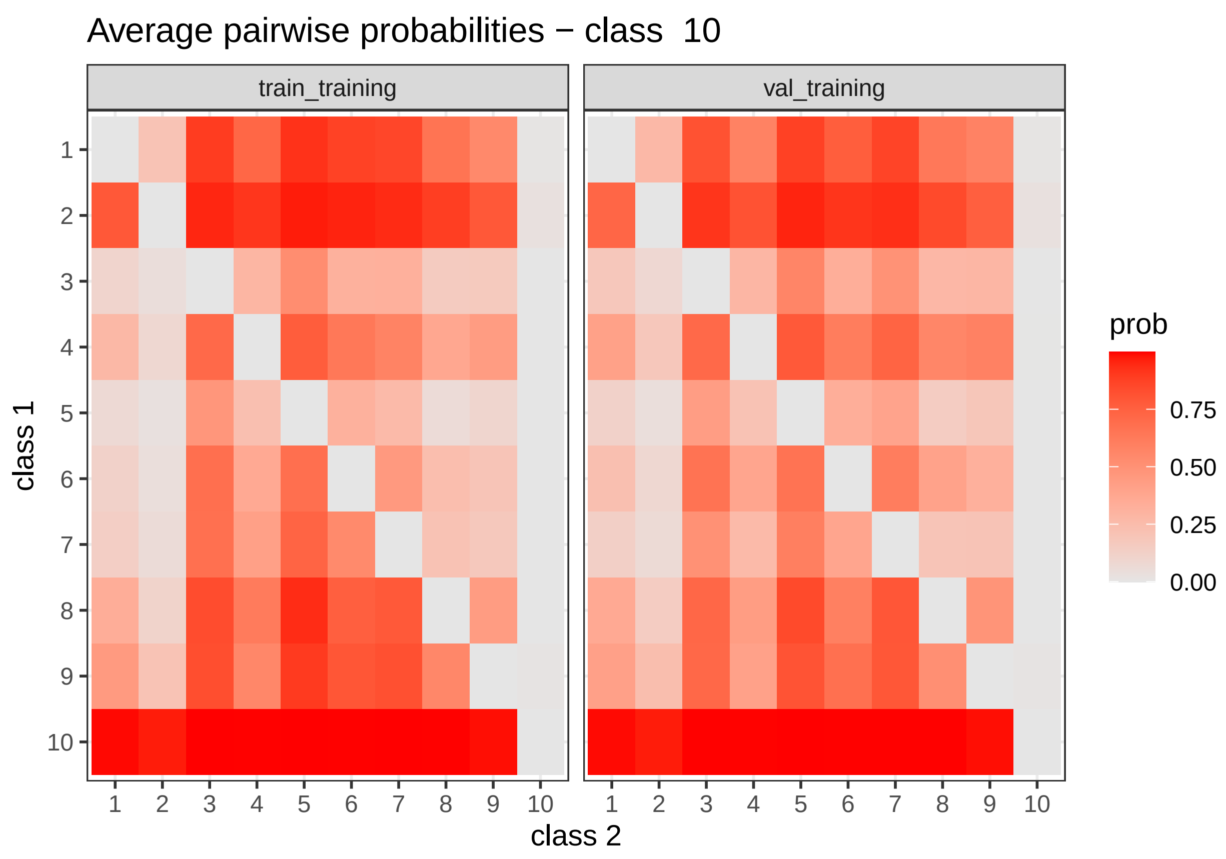

Average pairwise probabilities − class 5
train_training
val_training
class 1
class 2
1
2
3
4
5
6
7
8
9
10
prob
0.75
0.50
0.25
0.00


Average pairwise probabilities − class 6
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

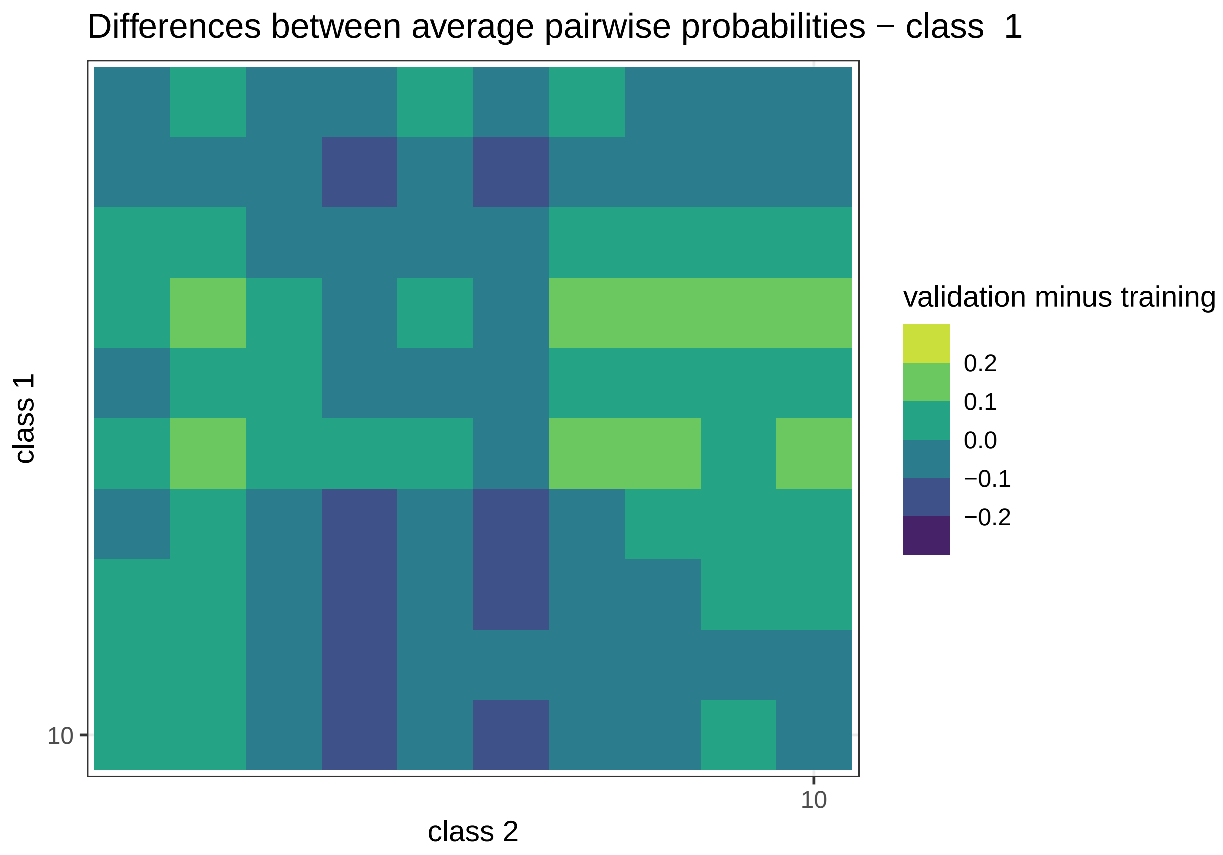

Average pairwise probabilities − class 7
train_training
val_training
class 1
class 2
1
2
3
4
5
6
7
8
9
10
prob
0.75
0.50
0.25
0.00

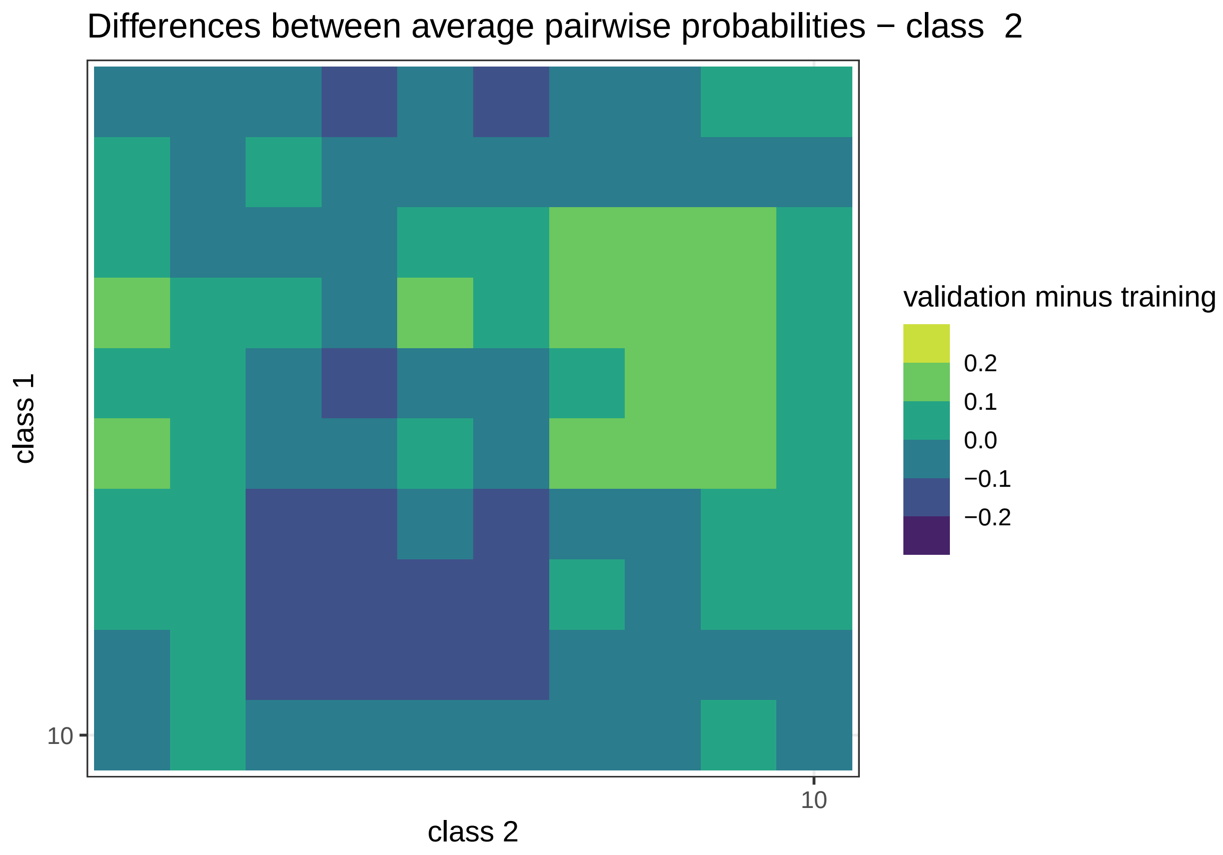

Average pairwise probabilities − class 8
train_training
val_training
class 1
class 2
1
2
3
4
5
6
7
8
9
10
prob
0.75
0.50
0.25
0.00

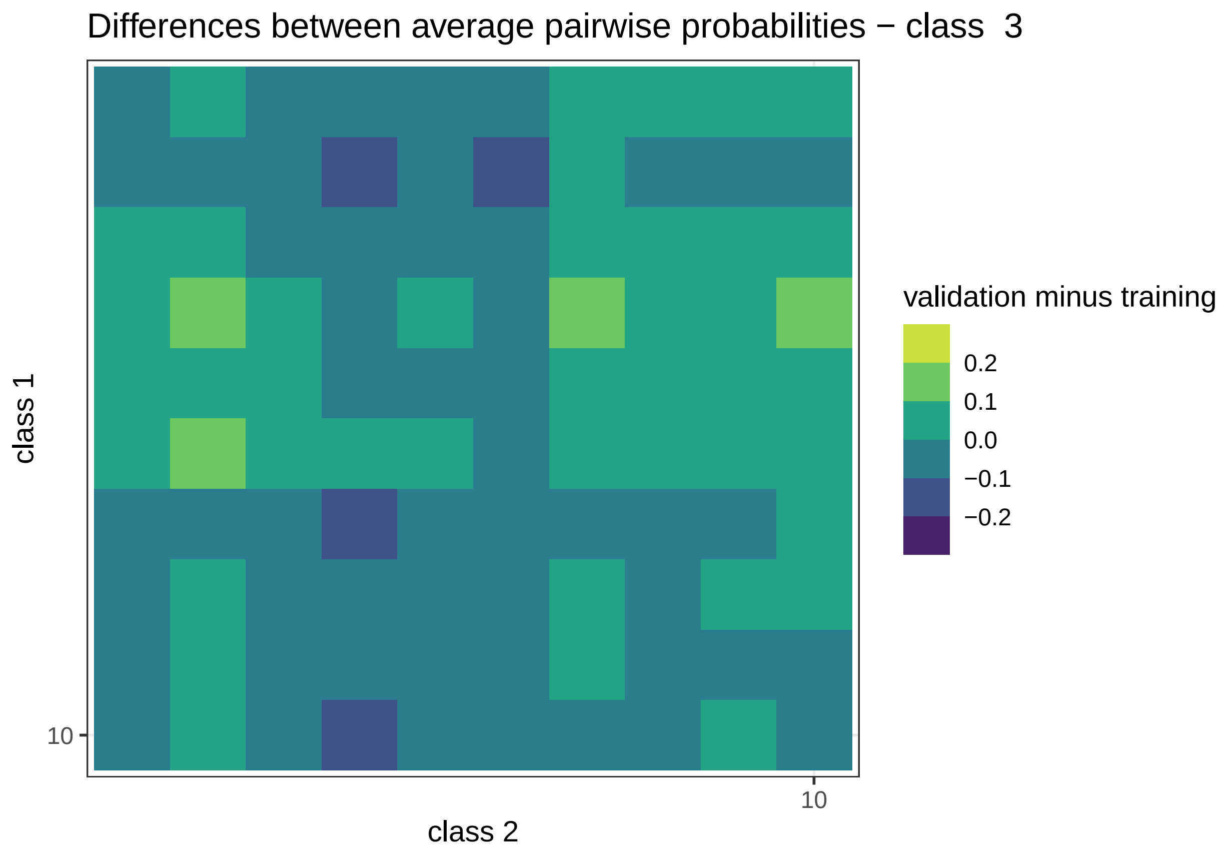

Average pairwise probabilities − class 9
train_training
val_training
class 1
class 2
1
2
3
4
5
6
7
8
9
10
prob
0.75
0.50
0.25
0.00



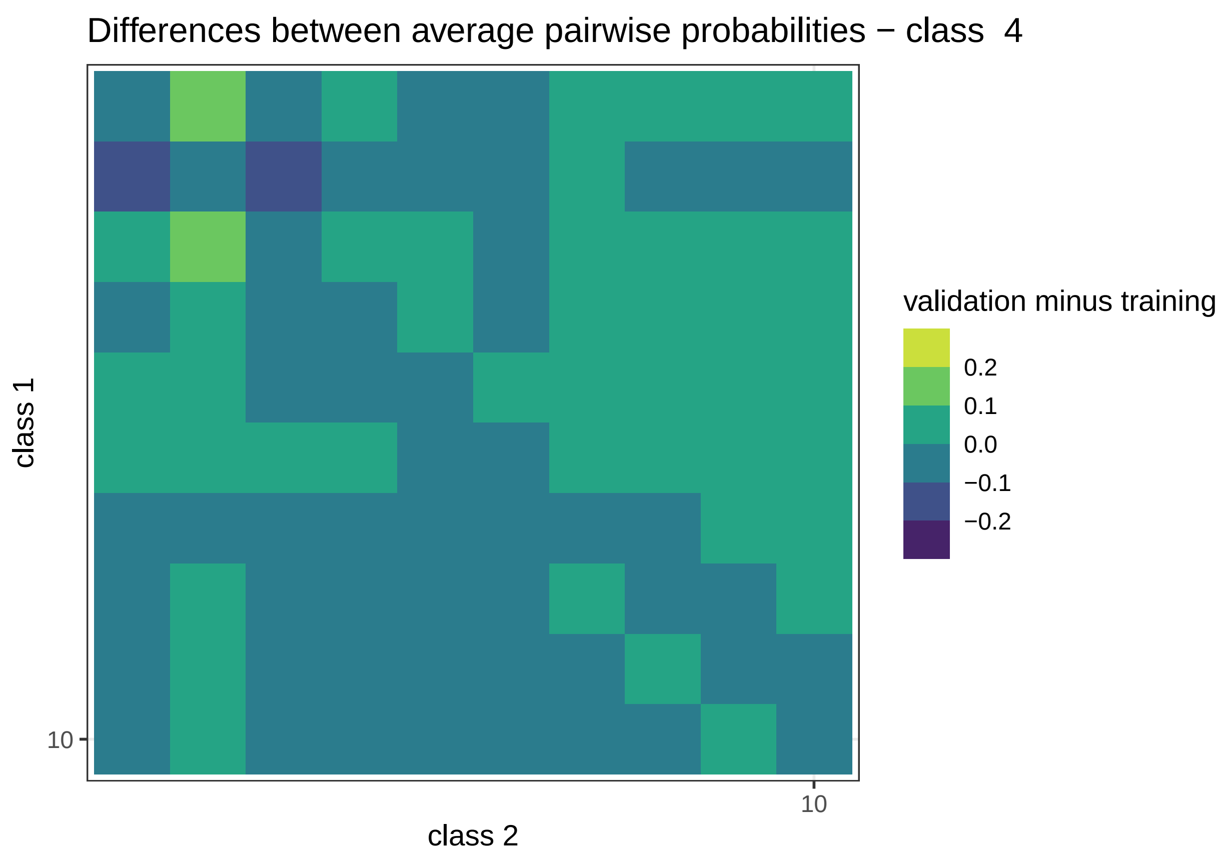
```r
for (cls in 1:classes)
{
  cur_class_Rs <- df_aggr_Rs_diff %>% filter(class == cls)
  plot_cls <-  ggplot(cur_class_Rs, aes(x = class2, y = class1)) +
    geom_raster(aes(fill=val_min_train)) +
    scale_fill_binned(type="viridis", limits=c(-0.3, 0.3), name="validation minus training") +
    scale_y_discrete(limits=rev, breaks=seq(0, classes, 10)) +
    scale_x_discrete(breaks=seq(0, classes, 10)) +
    labs(x="class 2", y="class 1", title=paste("Differences between average pairwise probabilities - cla
    theme_bw()

  print(plot_cls)
}
```

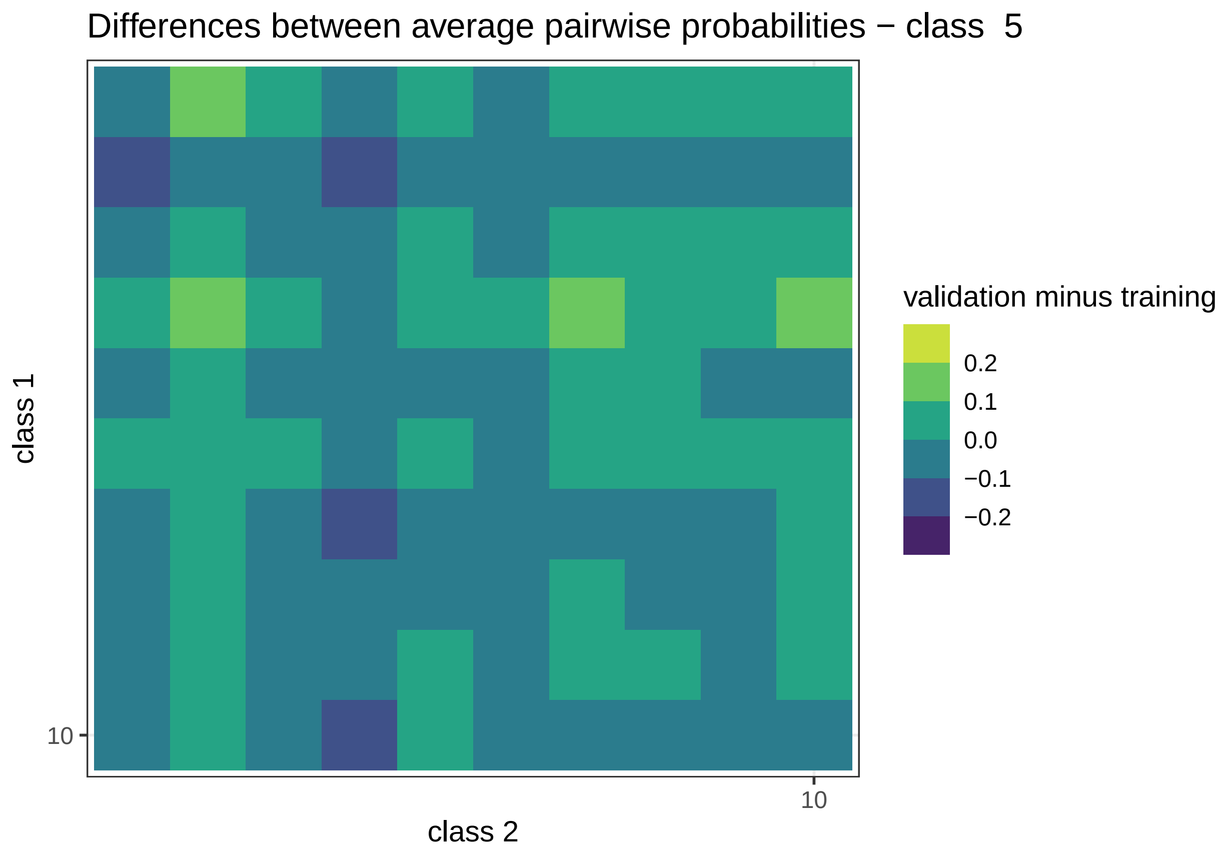

Differences between average pairwise probabilities − class 1
class 1
class 2
10
10
validation minus training
0.2
0.1
0.0
−0.1
−0.2

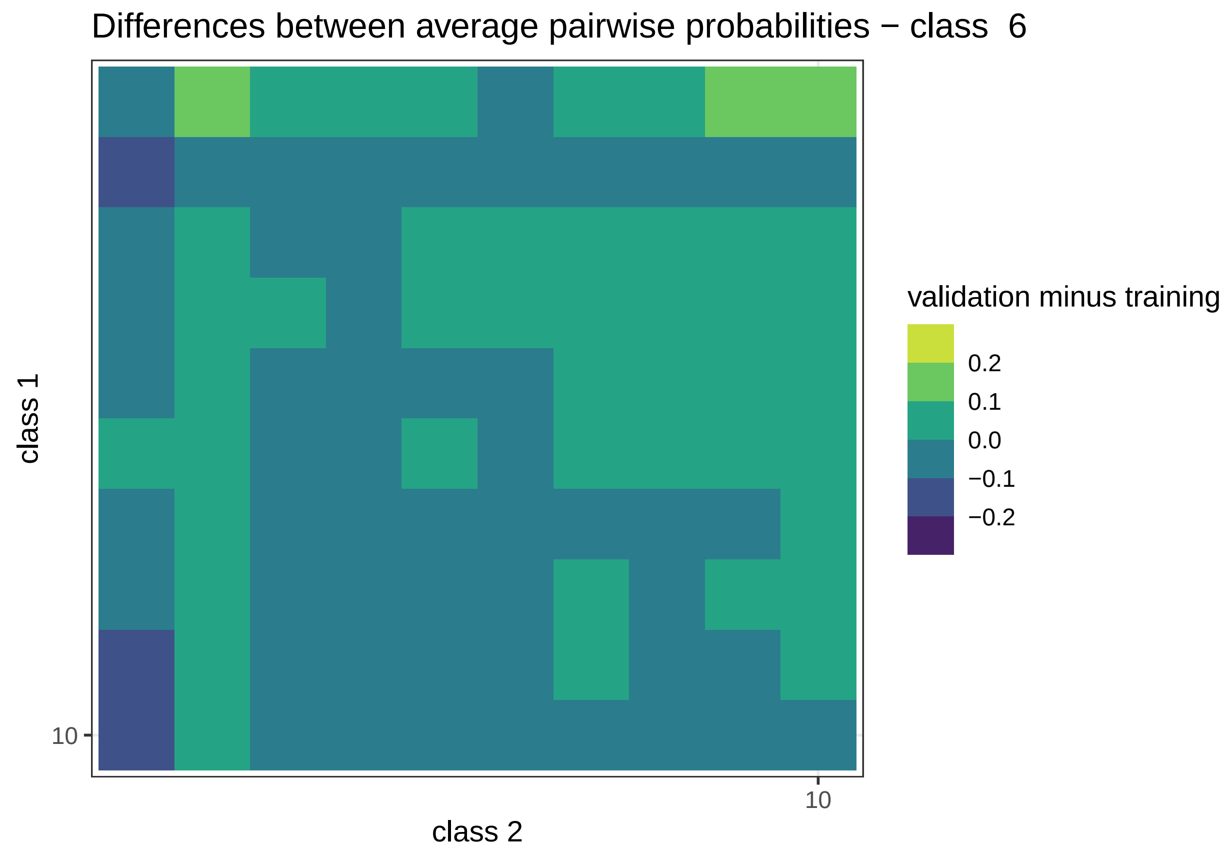

Differences between average pairwise probabilities − class 2
class 1
10
class 2
10
validation minus training
0.2
0.1
0.0
−0.1
−0.2

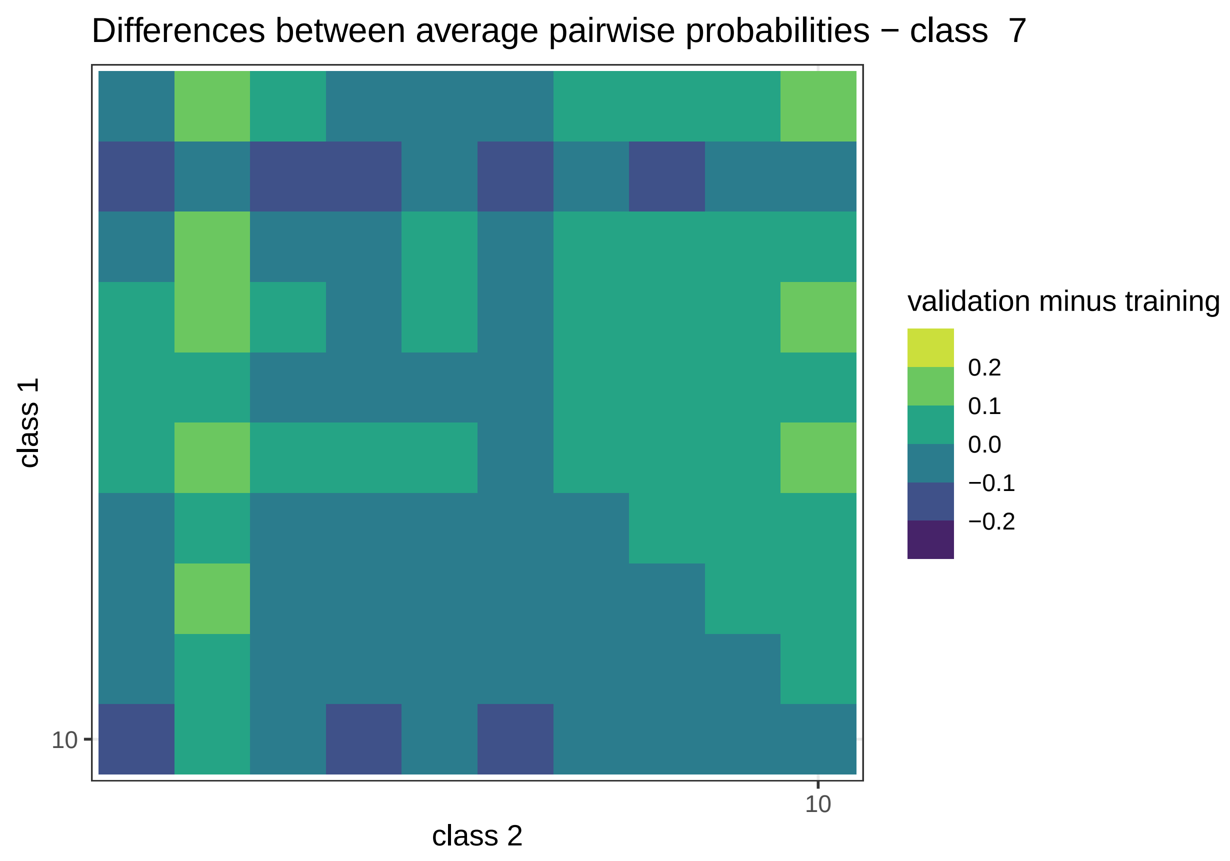


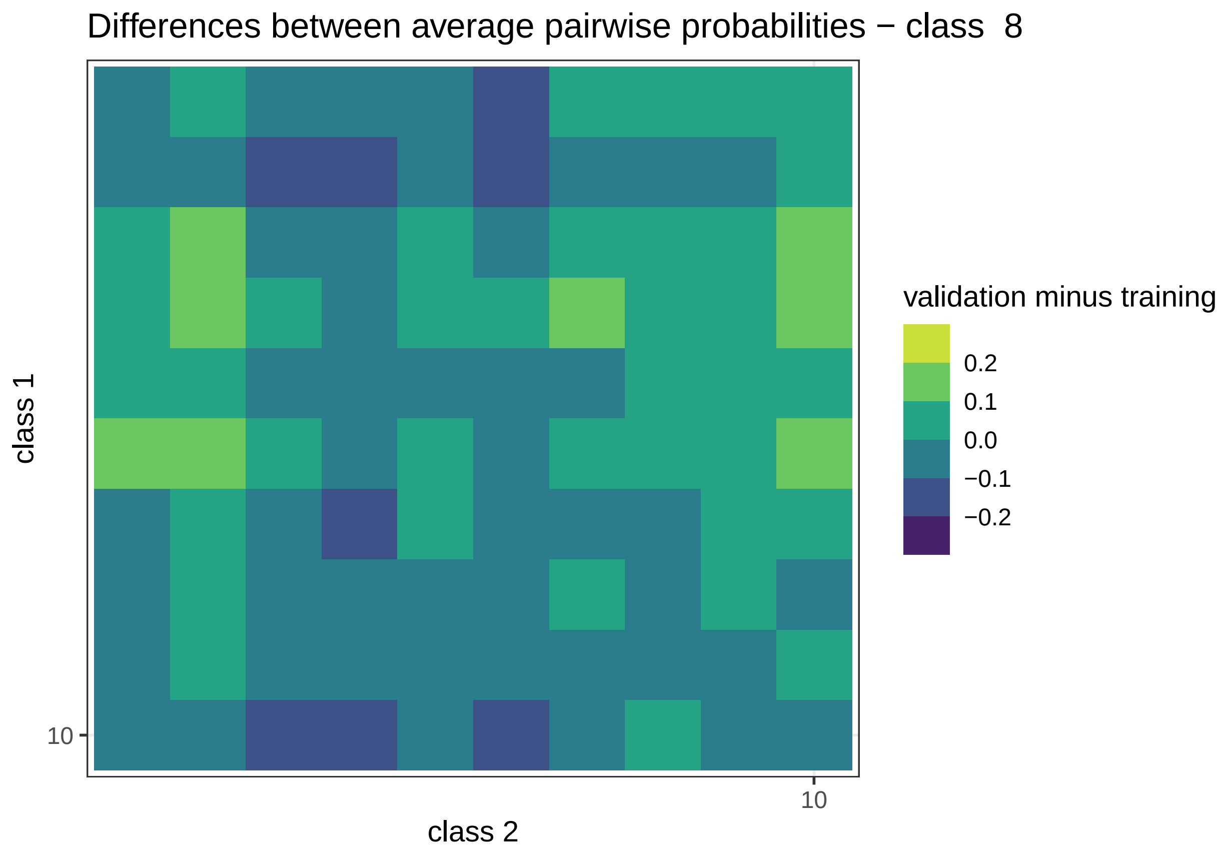
Differences between average pairwise probabilities − class 3
class 1
class 2
10
10
validation minus training
0.2
0.1
0.0
−0.1
−0.2

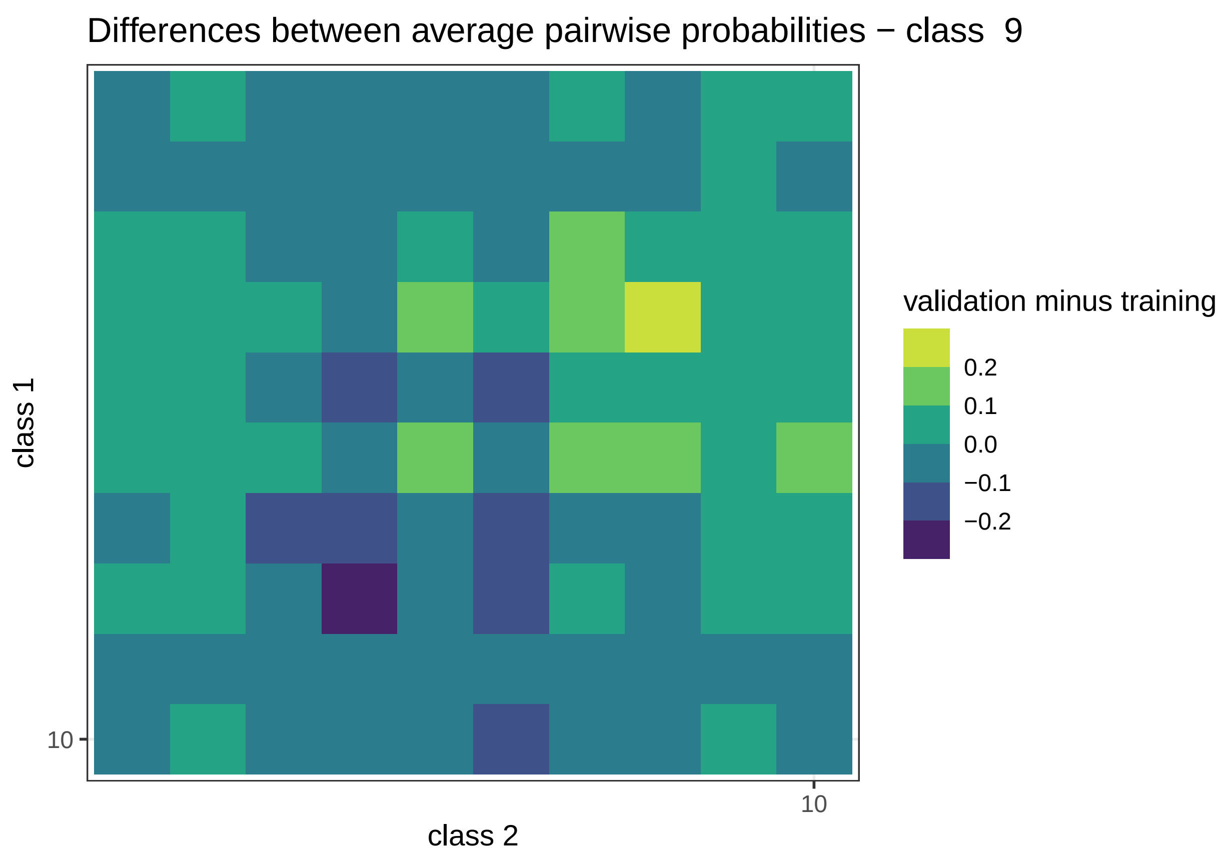

Differences between average pairwise probabilities − class 4
class 1
class 2
10
10
validation minus training
0.2
0.1
0.0
−0.1
−0.2

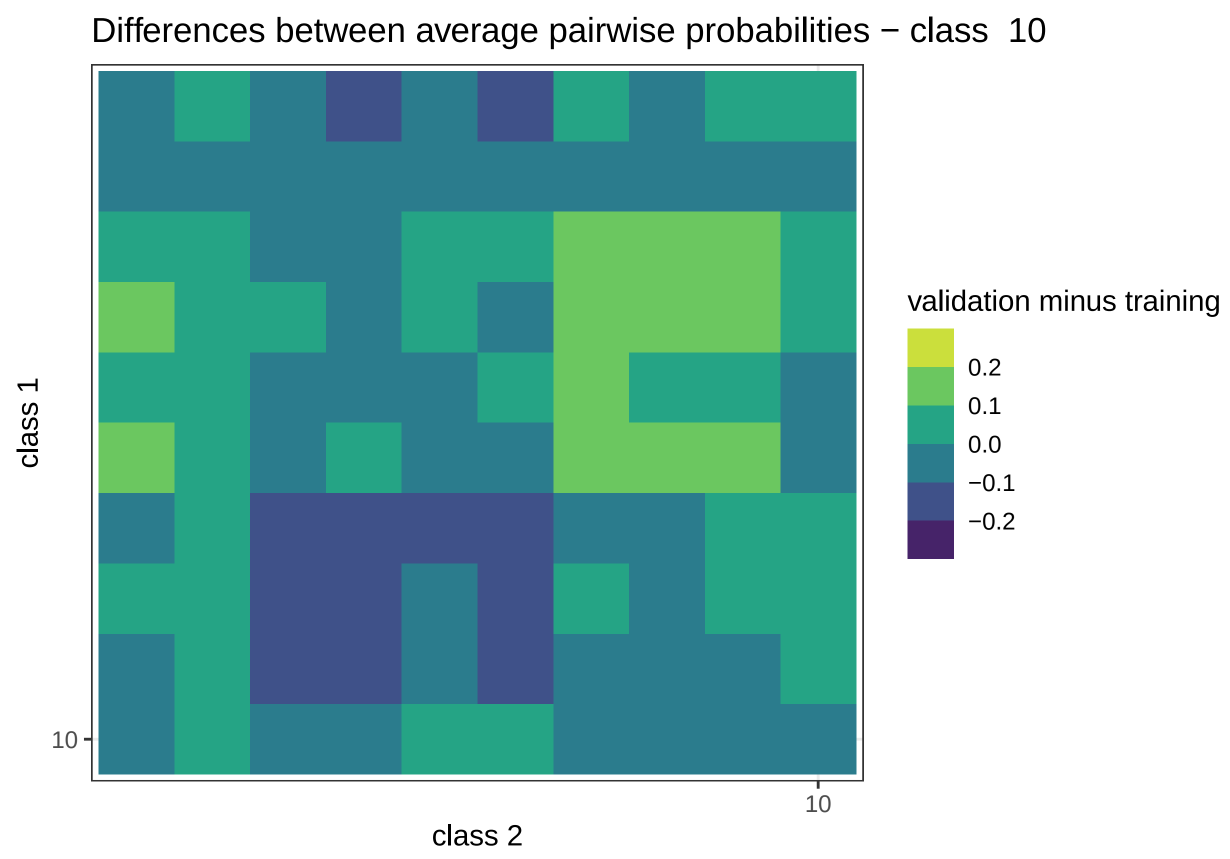

Differences between average pairwise probabilities − class 5
class 1
class 2
10
10
validation minus training
0.2
0.1
0.0
−0.1
−0.2


Differences between average pairwise probabilities − class 6
class 1
class 2
10
10
validation minus training
0.2
0.1
0.0
−0.1
−0.2


Differences between average pairwise probabilities − class 7
class 1
class 2
10
10
validation minus training
0.2
0.1
0.0
−0.1
−0.2

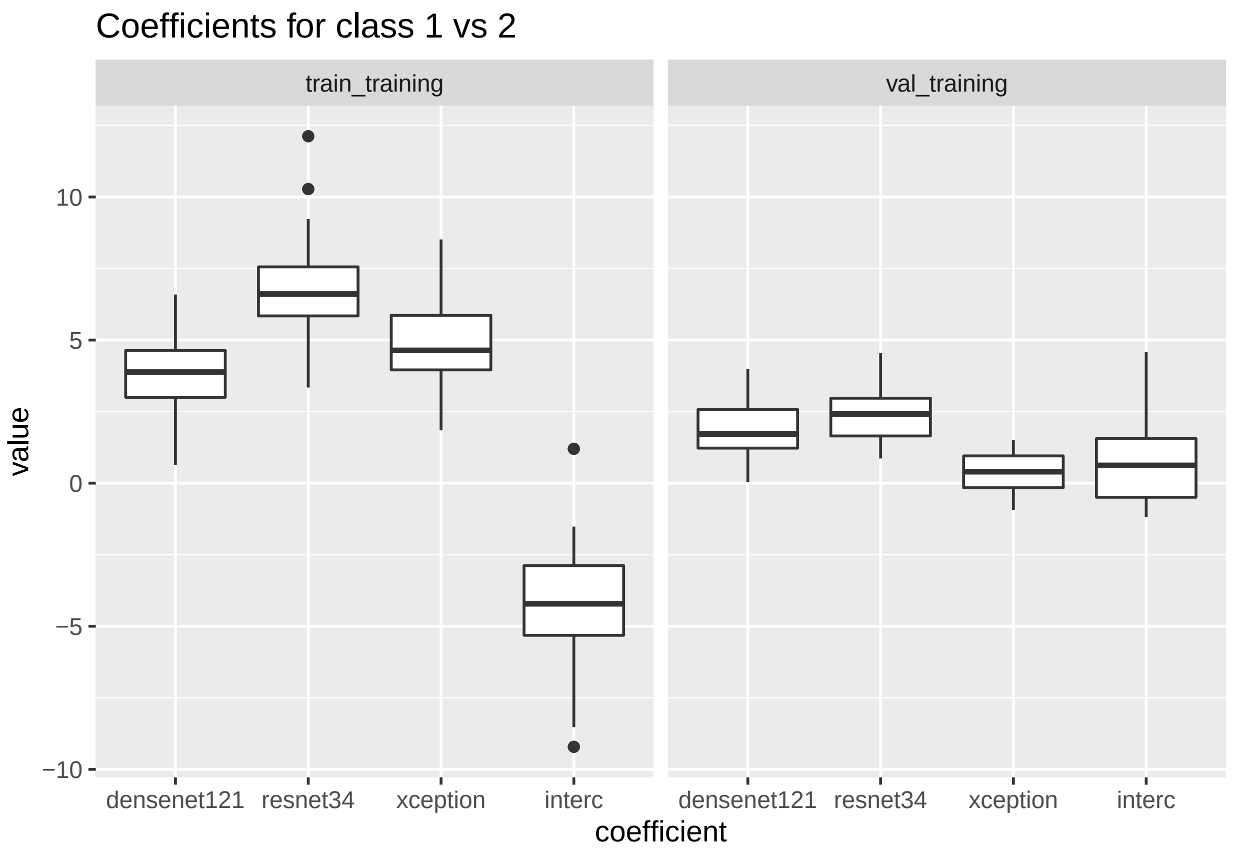

Differences between average pairwise probabilities − class 8
validation minus training
0.2
0.1
0.0
−0.1
−0.2
class 1
10
class 2
10

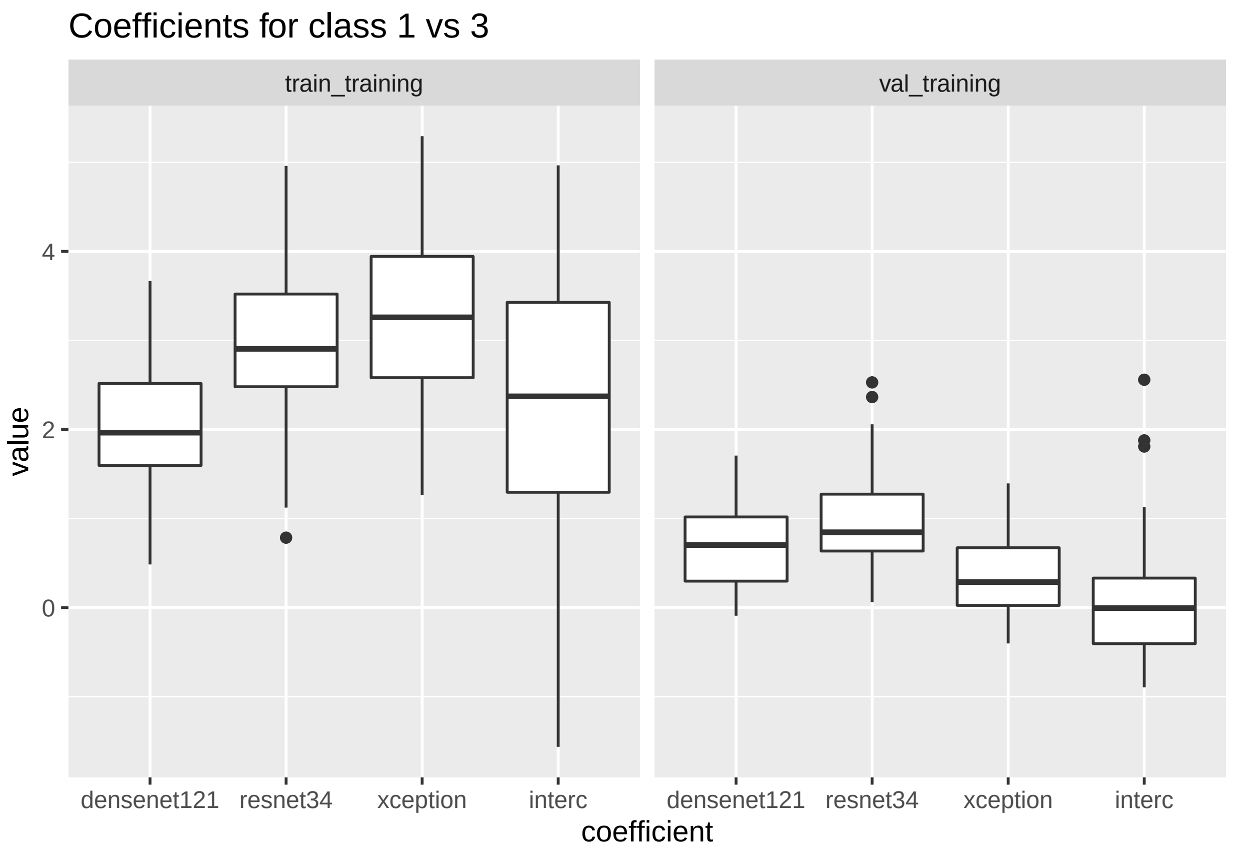

Differences between average pairwise probabilities − class 9
class 1
class 2
10
10
validation minus training
0.2
0.1
0.0
−0.1
−0.2

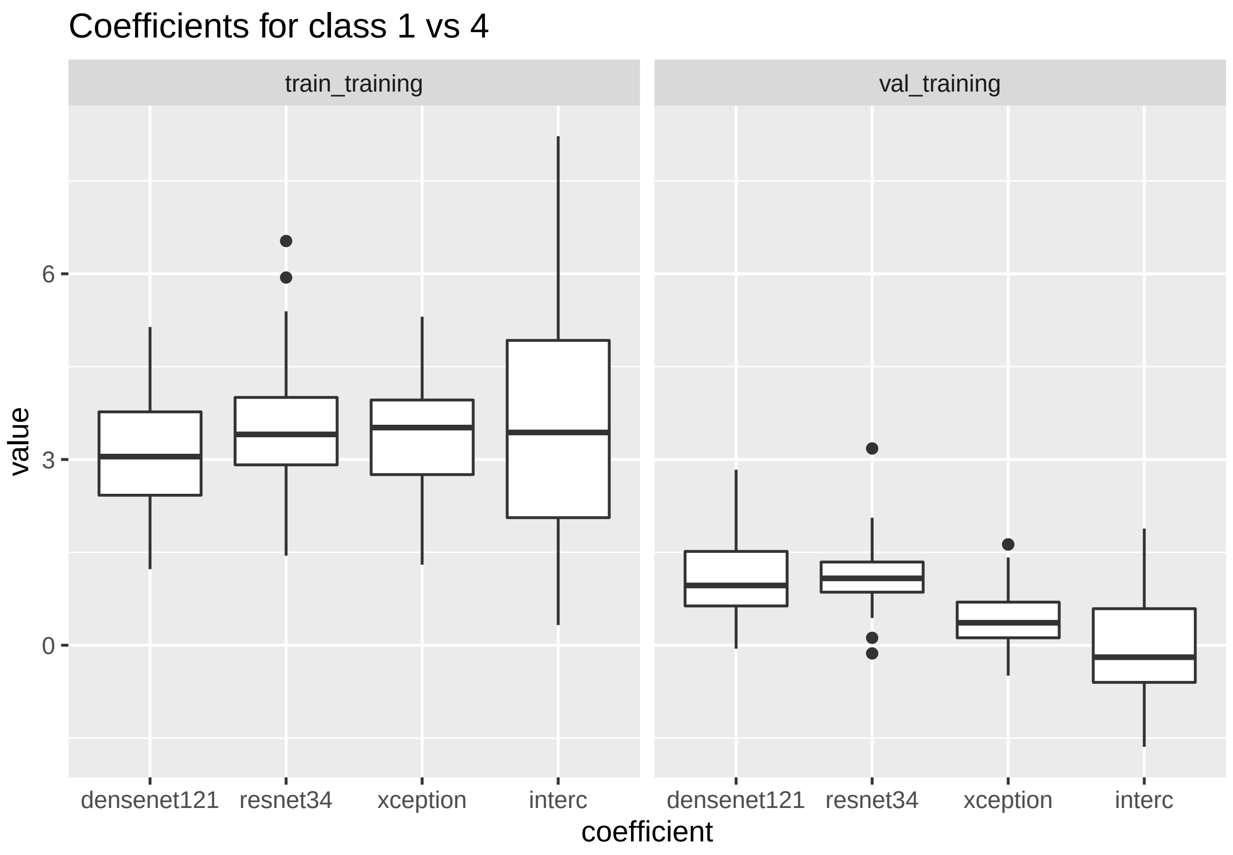


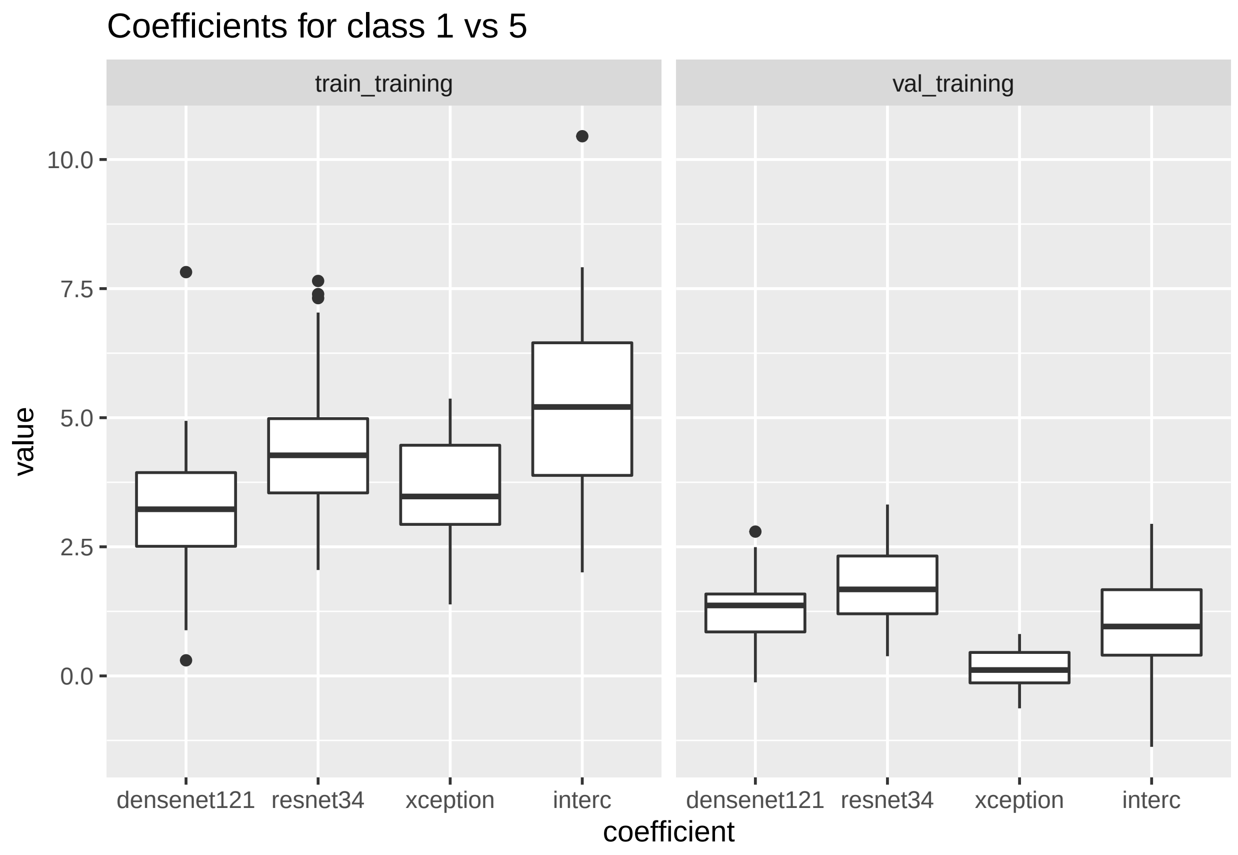
```
lda_coefs <- load_lda_coefs(base_dir, repls, folds)
```

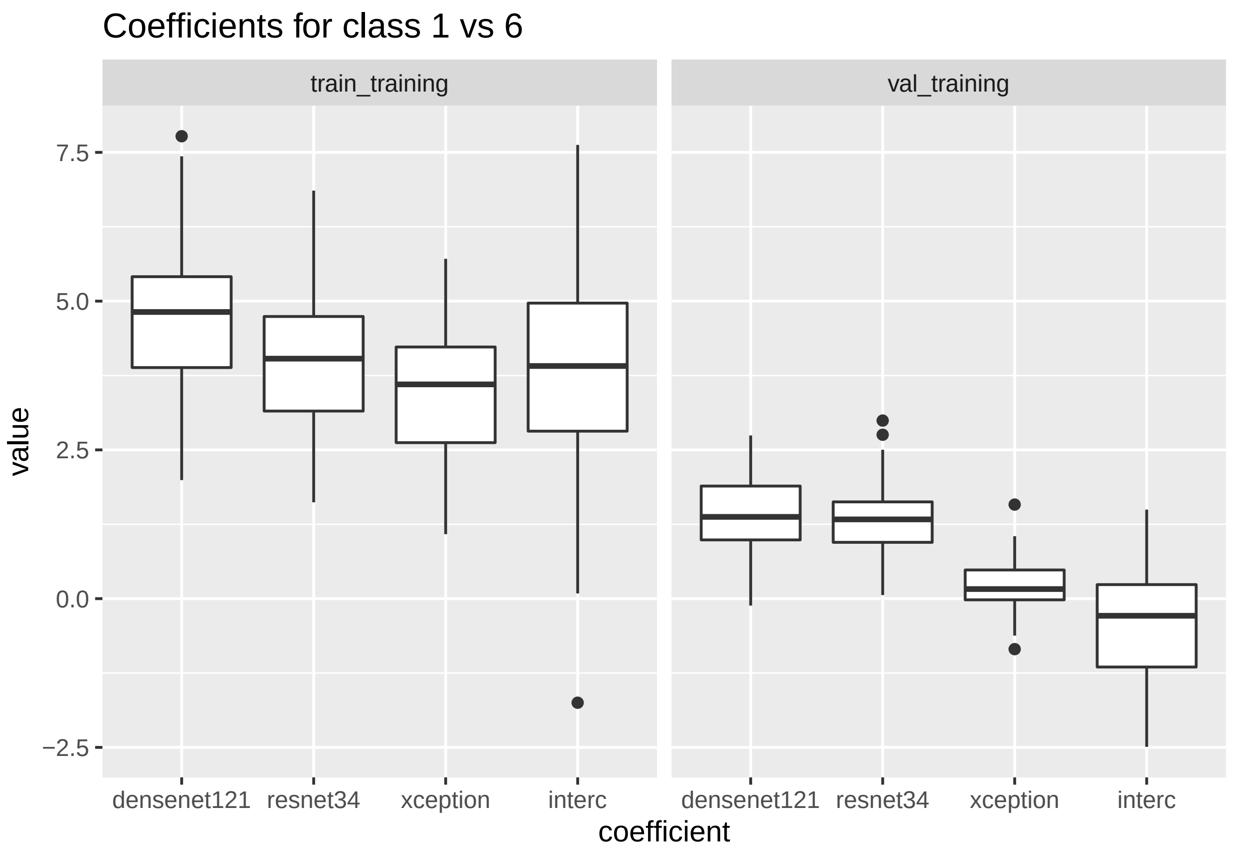
```
for (cl1 in 1:(classes - 1))
{
  for (cl2 in (cl1 + 1):classes)
  {
    cur_plt <- lda_coefs %>% filter(class1 == cl1 & class2 == cl2) %>% ggplot() + geom_boxplot(aes(x=coe
      facet_wrap(~train_type) + ggtitle(paste("Coefficients for class", cl1, "vs", cl2))
    print(cur_plt)
  }
}
```

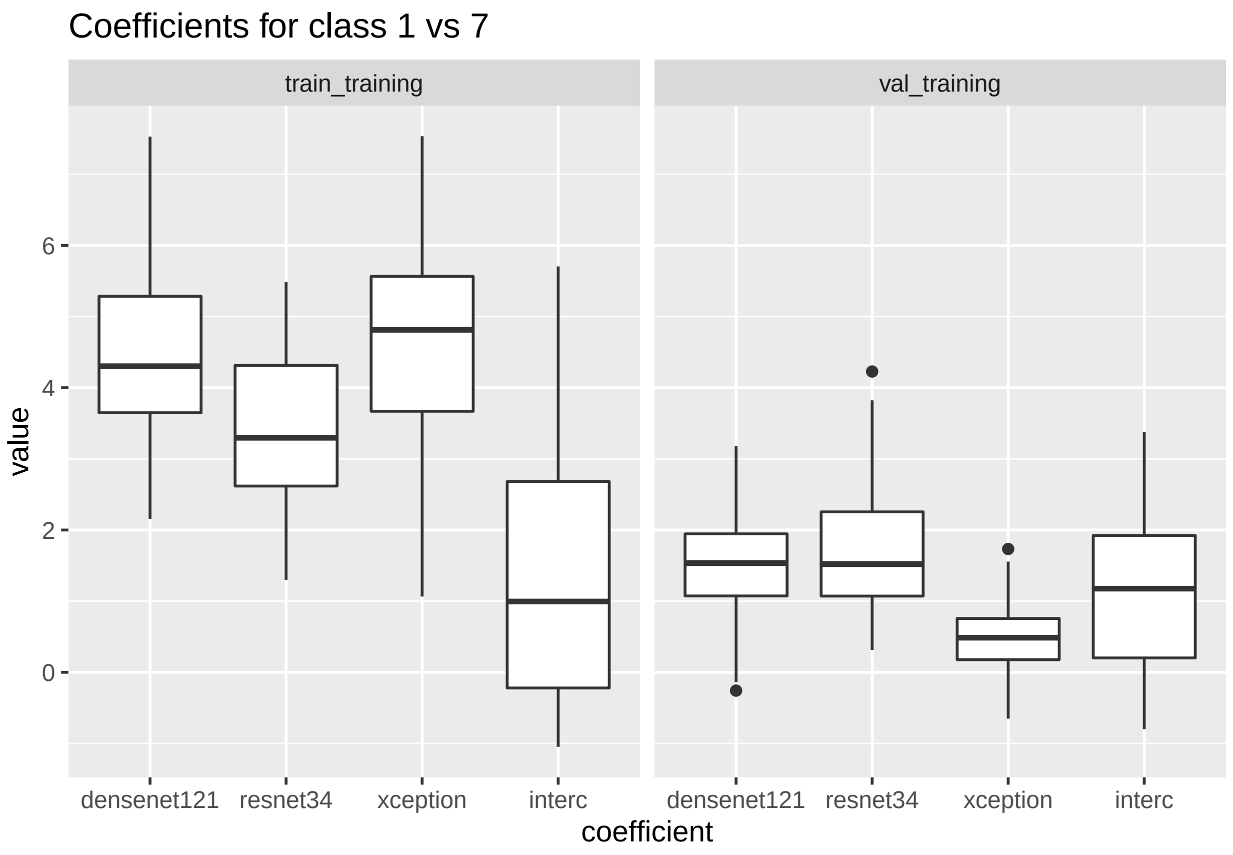

Coefficients for class 1 vs 2
train_training
val_training
value
10
5
0
−5
−10
densenet121
resnet34
xception
interc
coefficient

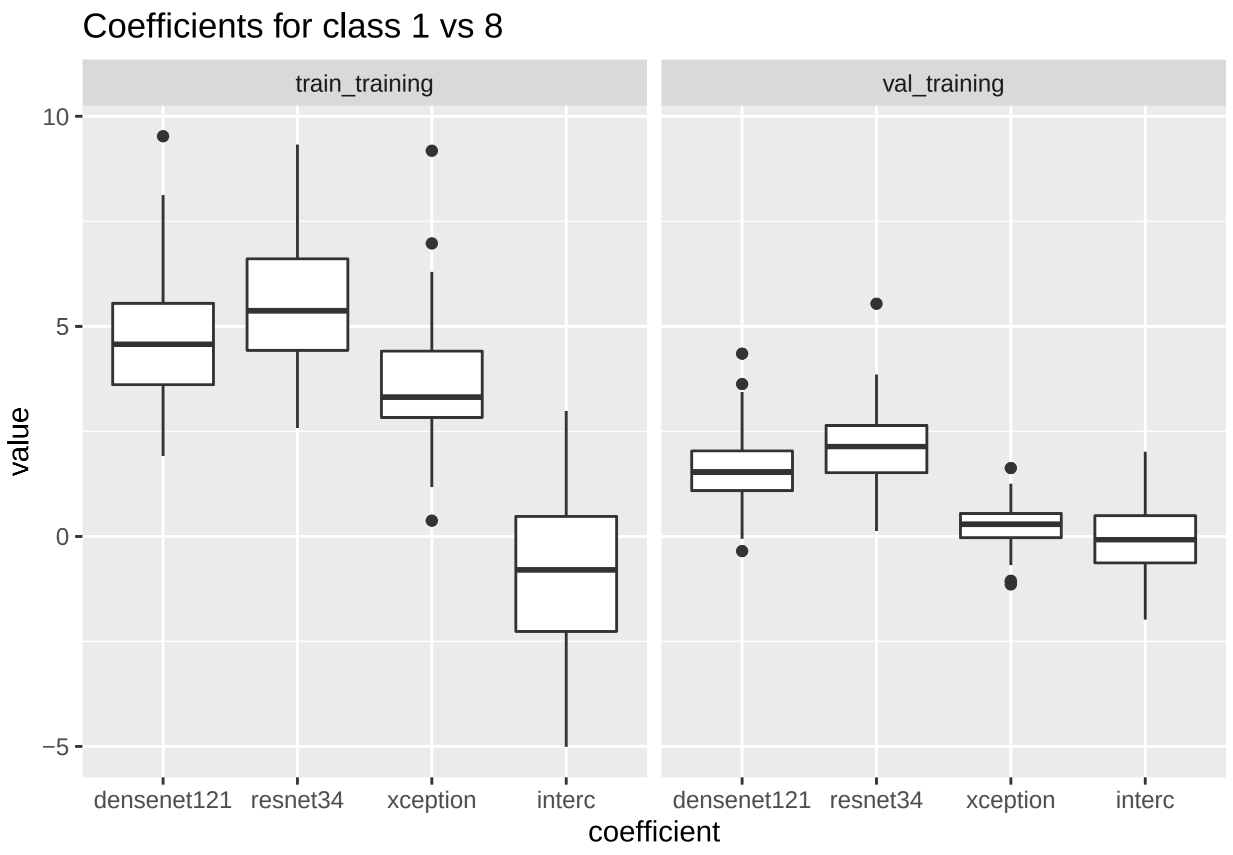

Coefficients for class 1 vs 3
train_training
val_training
value
4
2
0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

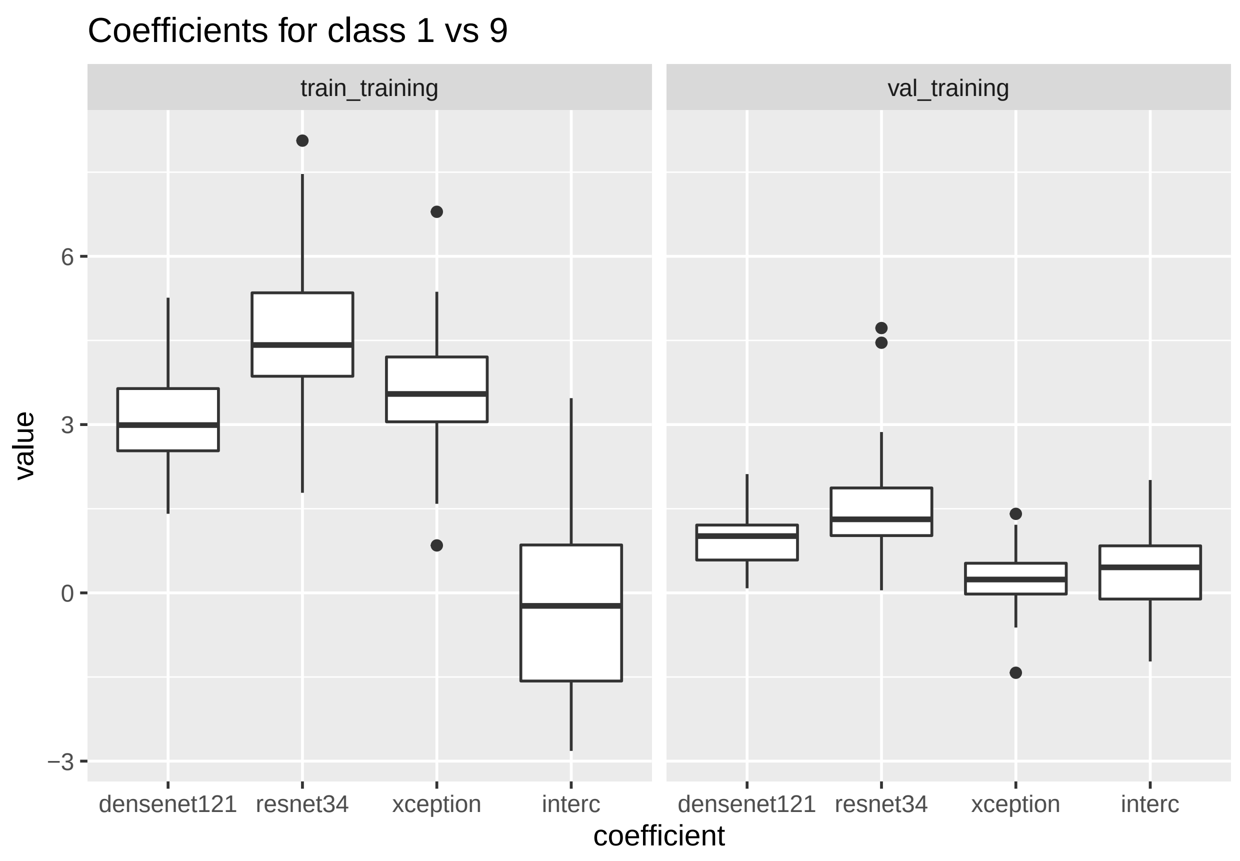

Coefficients for class 1 vs 4
train_training
val_training
value
6
3
0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

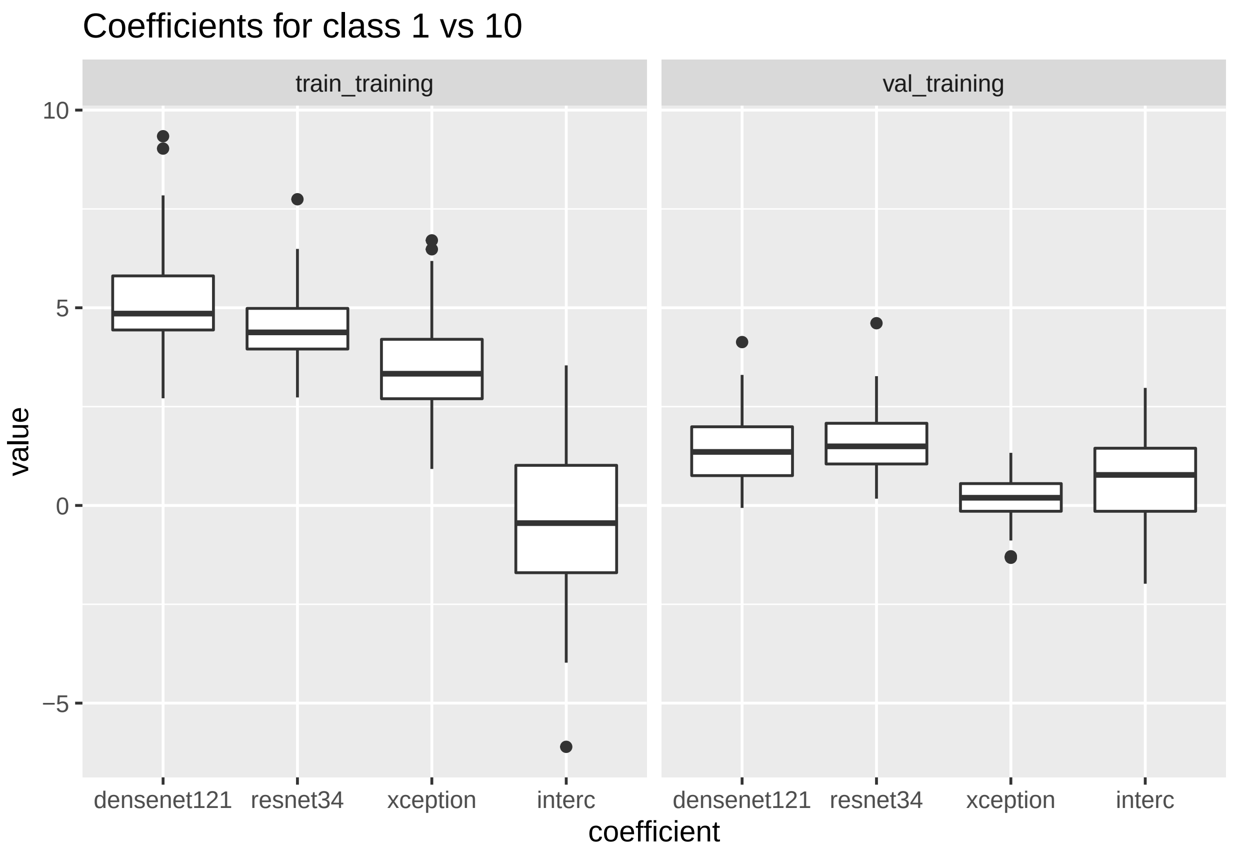

Coefficients for class 1 vs 5
train_training
val_training
value
10.0
7.5
5.0
2.5
0.0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

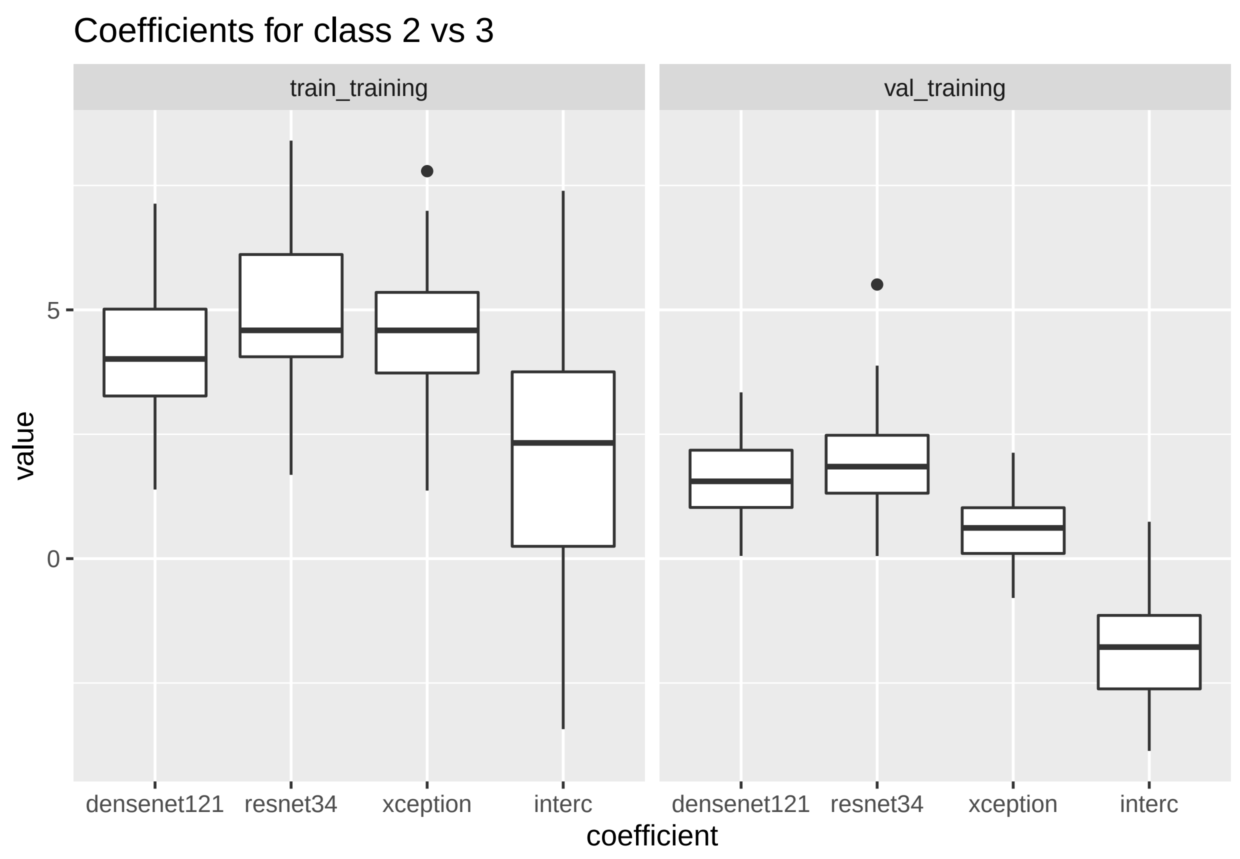

Coefficients for class 1 vs 6
train_training
val_training
value
7.5
5.0
2.5
0.0
−2.5
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient


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

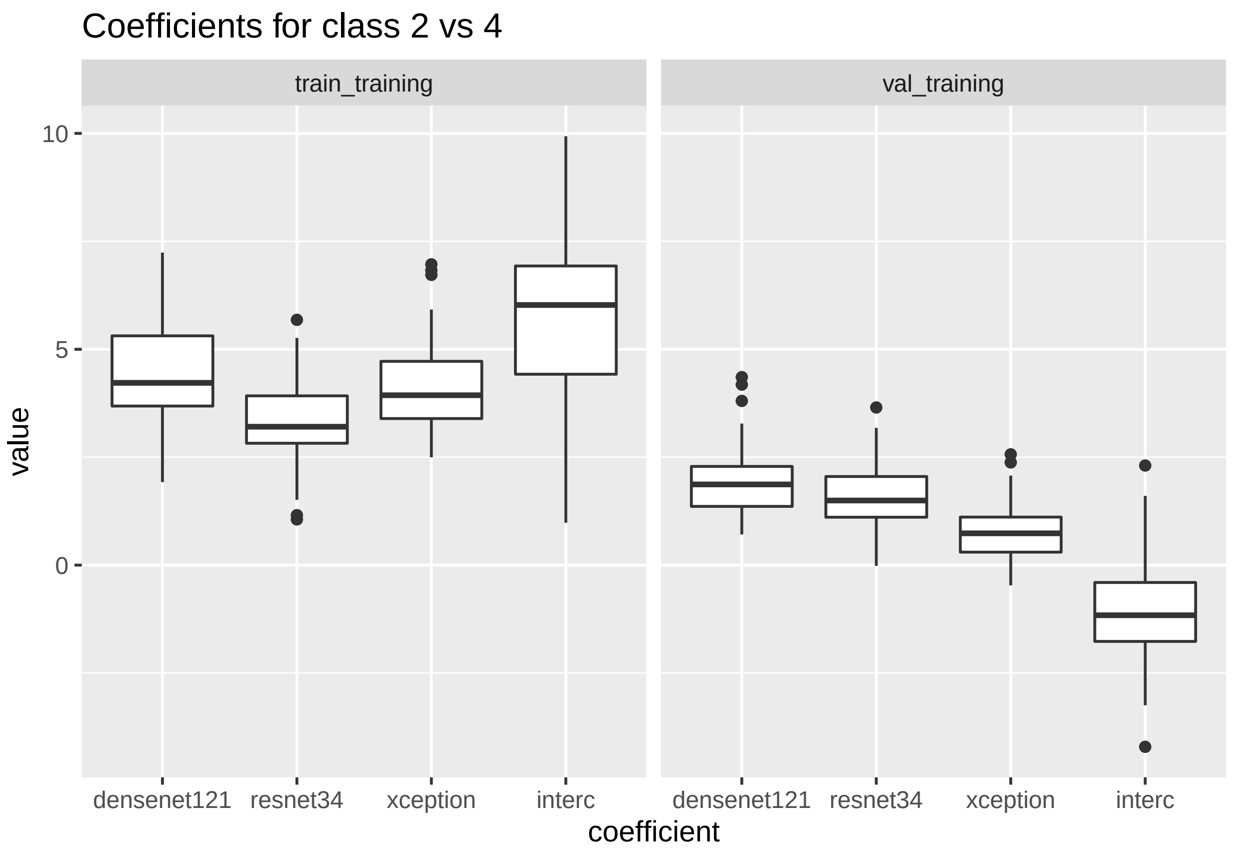

Coefficients for class 1 vs 8
train_training
val_training
value
10
5
0
−5
densenet121
resnet34
xception
interc
coefficient

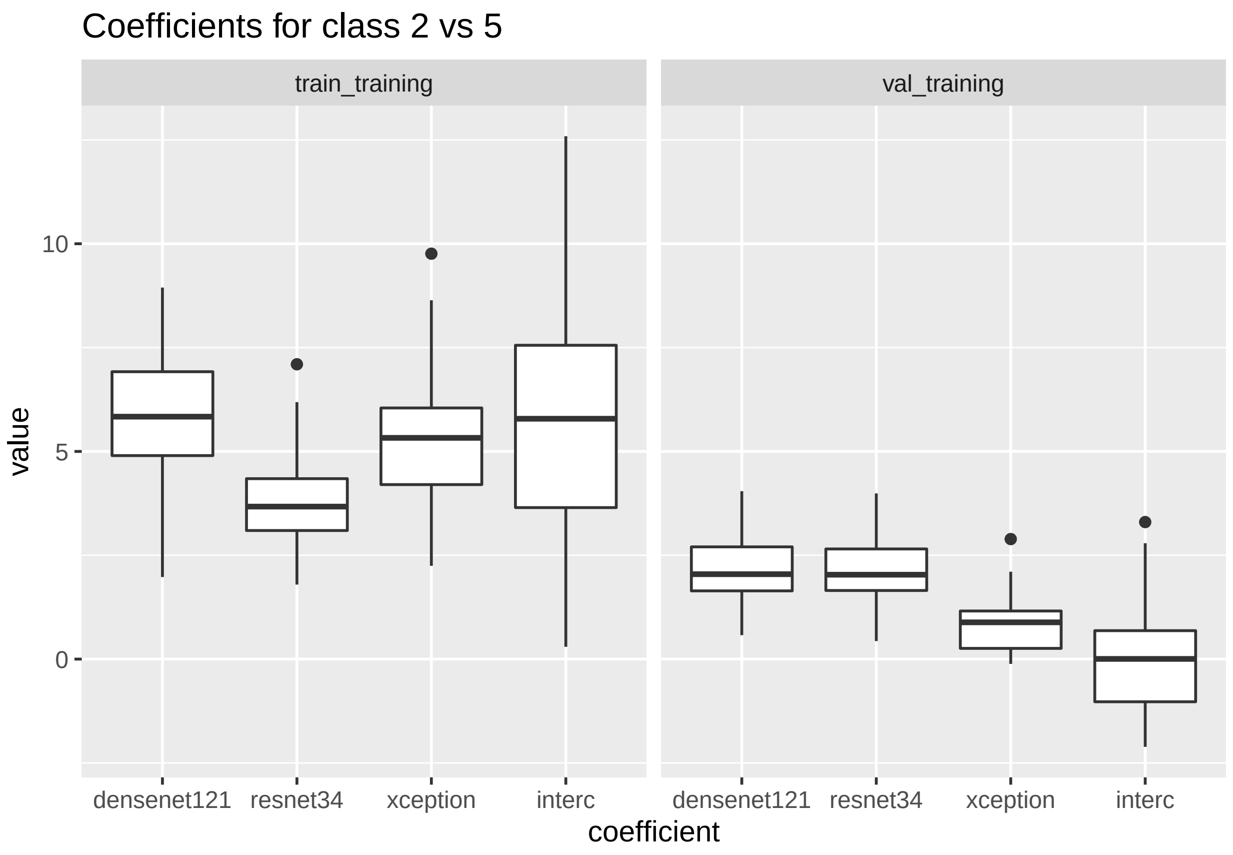

Coefficients for class 1 vs 9
train_training
val_training
value
6
3
0
−3
densenet121
resnet34
xception
interc
coefficient

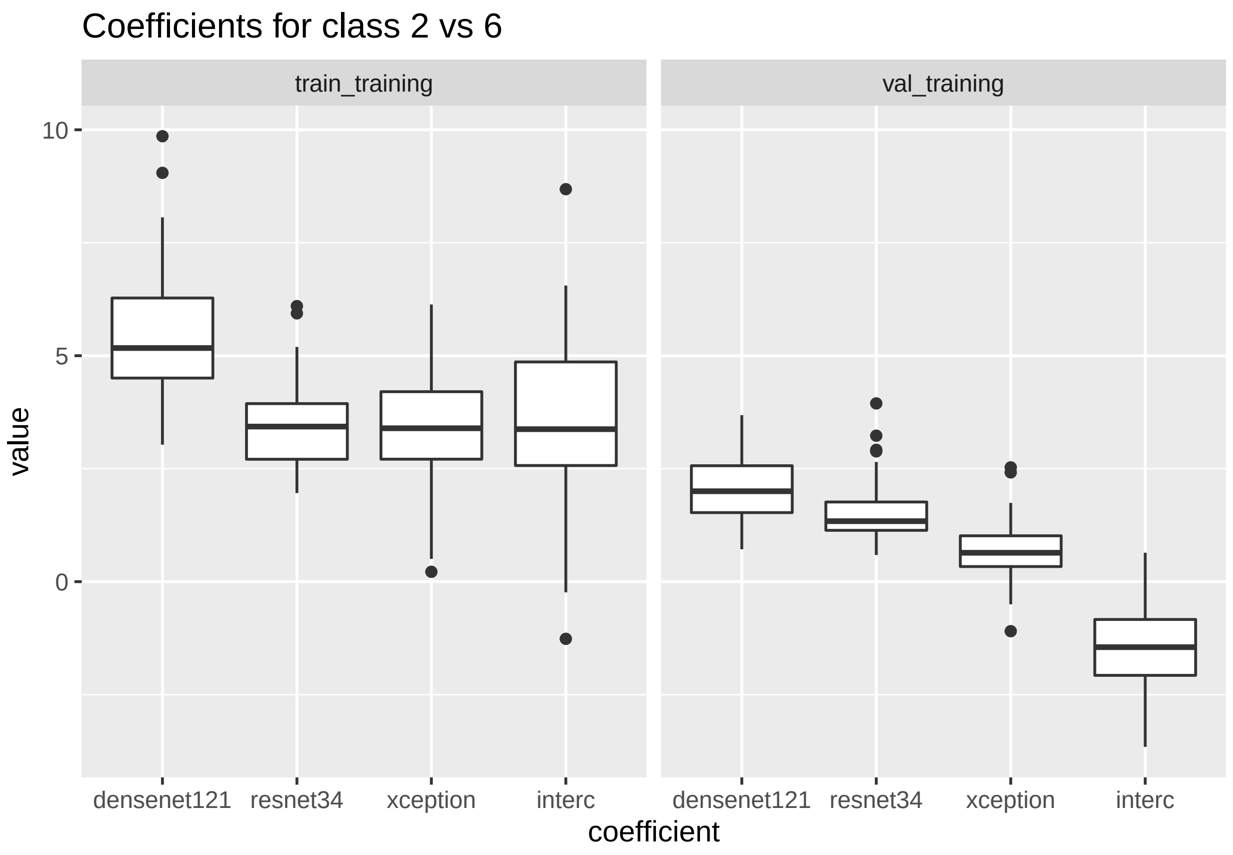

Coefficients for class 1 vs 10
train_training
val_training
value
10
5
0
−5
densenet121
resnet34
xception
interc
coefficient

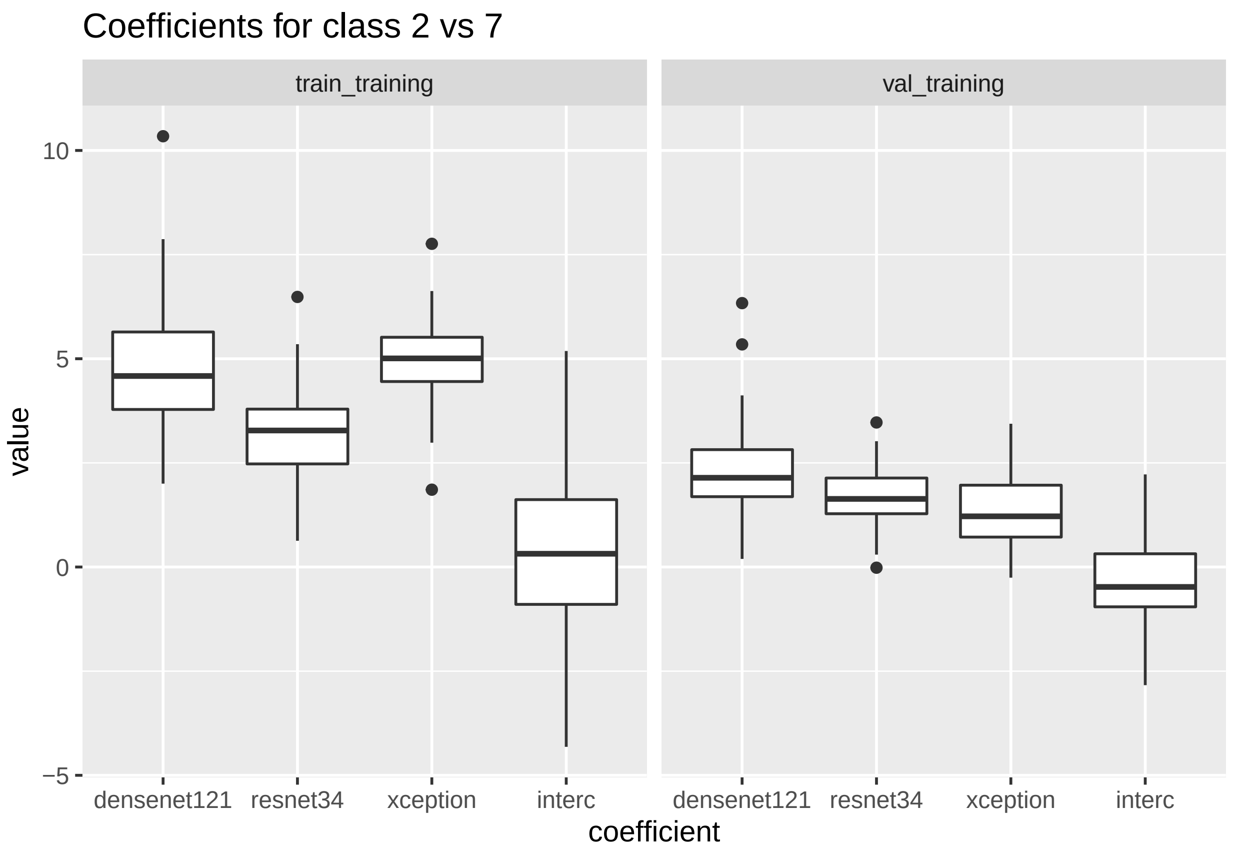


Coefficients for class 2 vs 3
train_training
val_training
value
5
0
densenet121
resnet34
xception
interc
coefficient

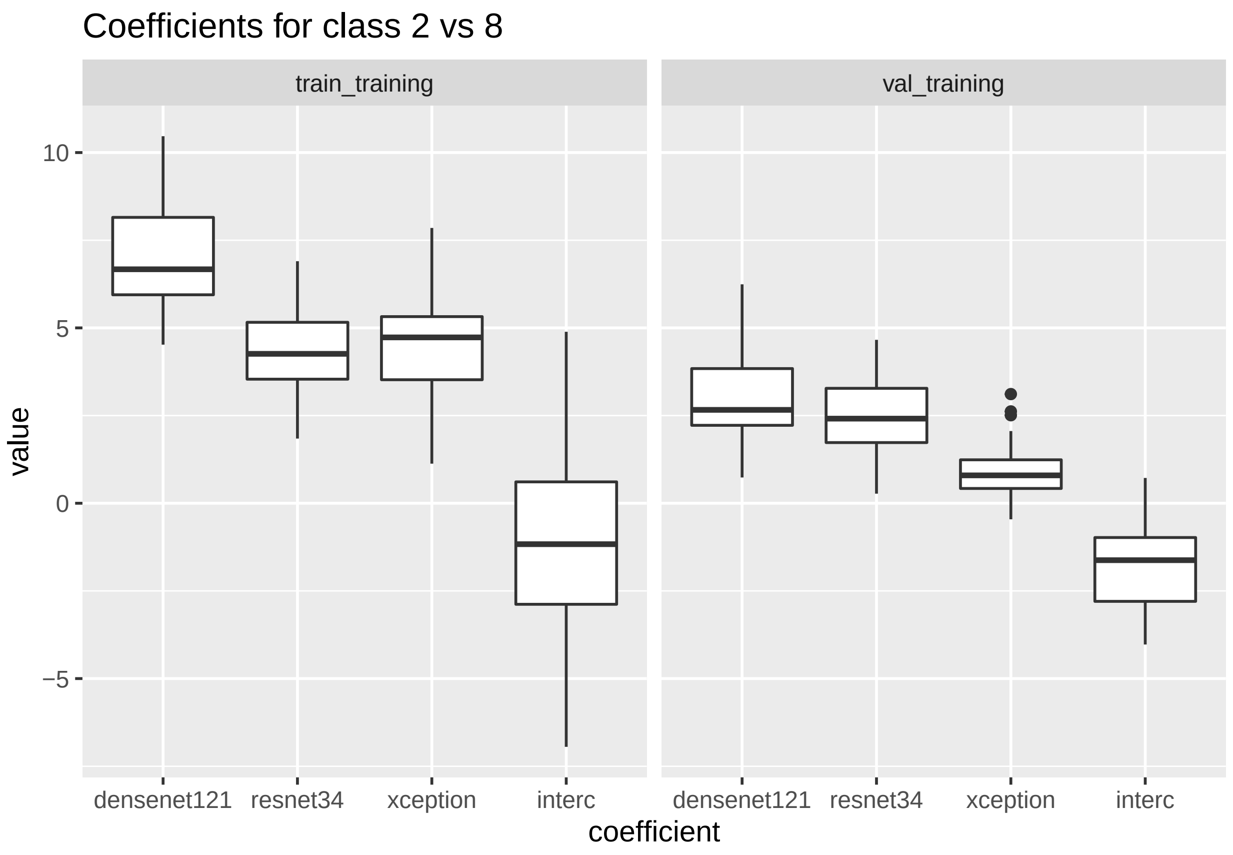

Coefficients for class 2 vs 4
train_training
val_training
10
5
0
value
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

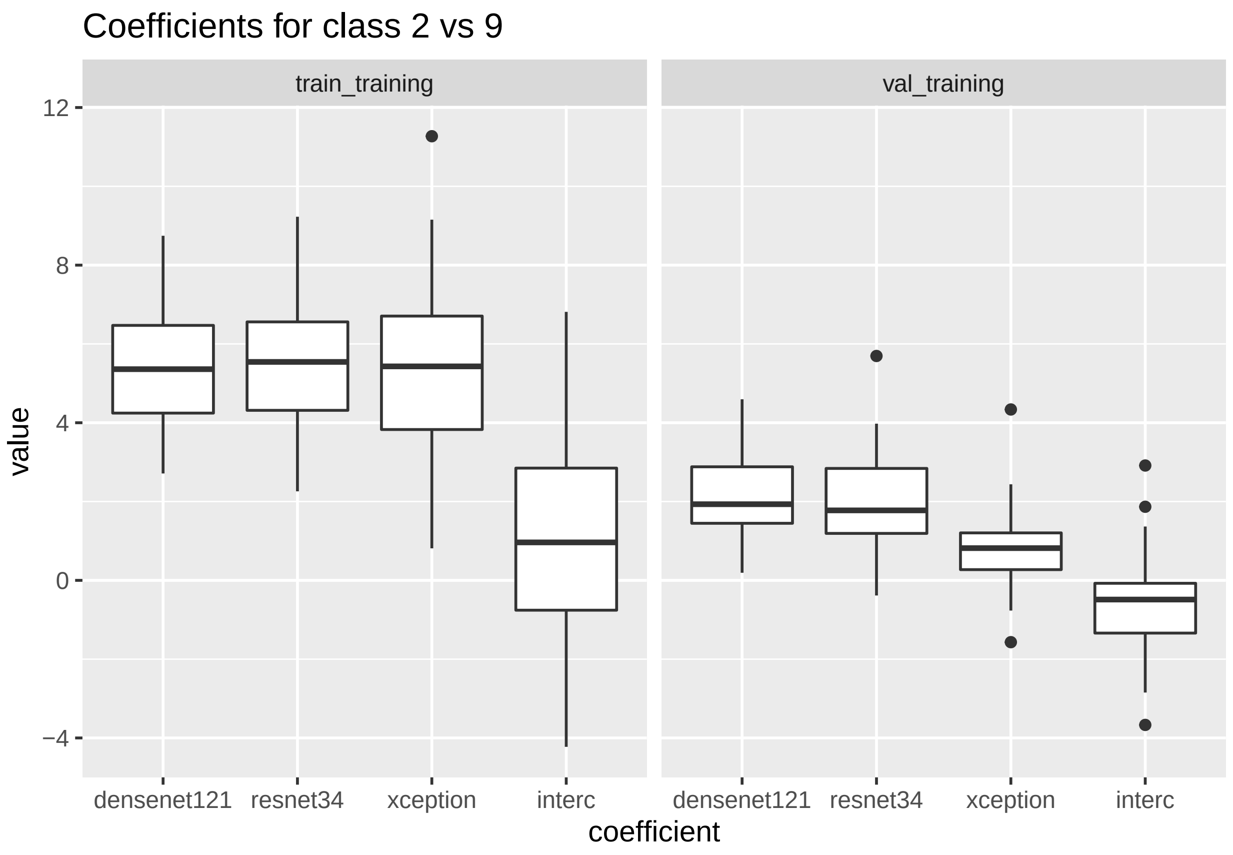

Coefficients for class 2 vs 5
train_training
val_training
value
10
5
0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

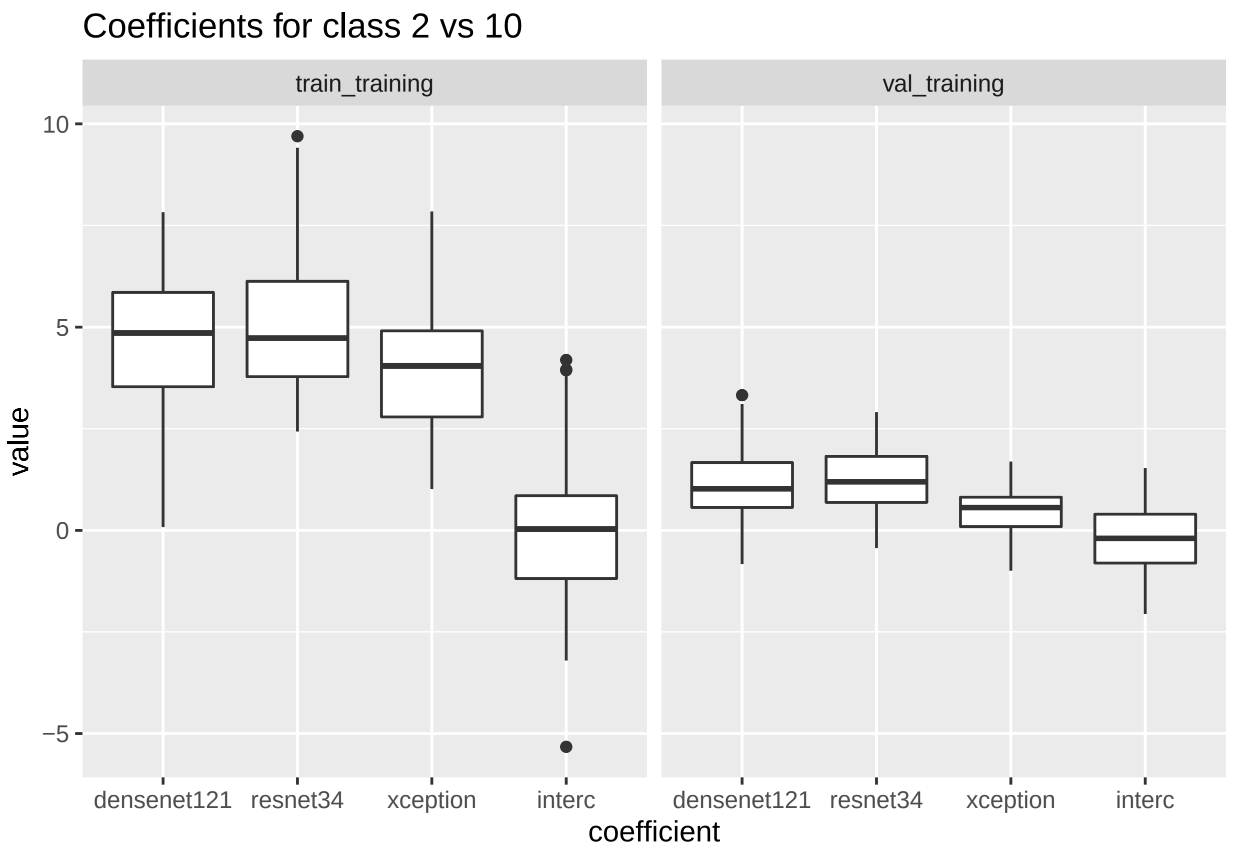

Coefficients for class 2 vs 6
train_training
val_training
10
5
0
value
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

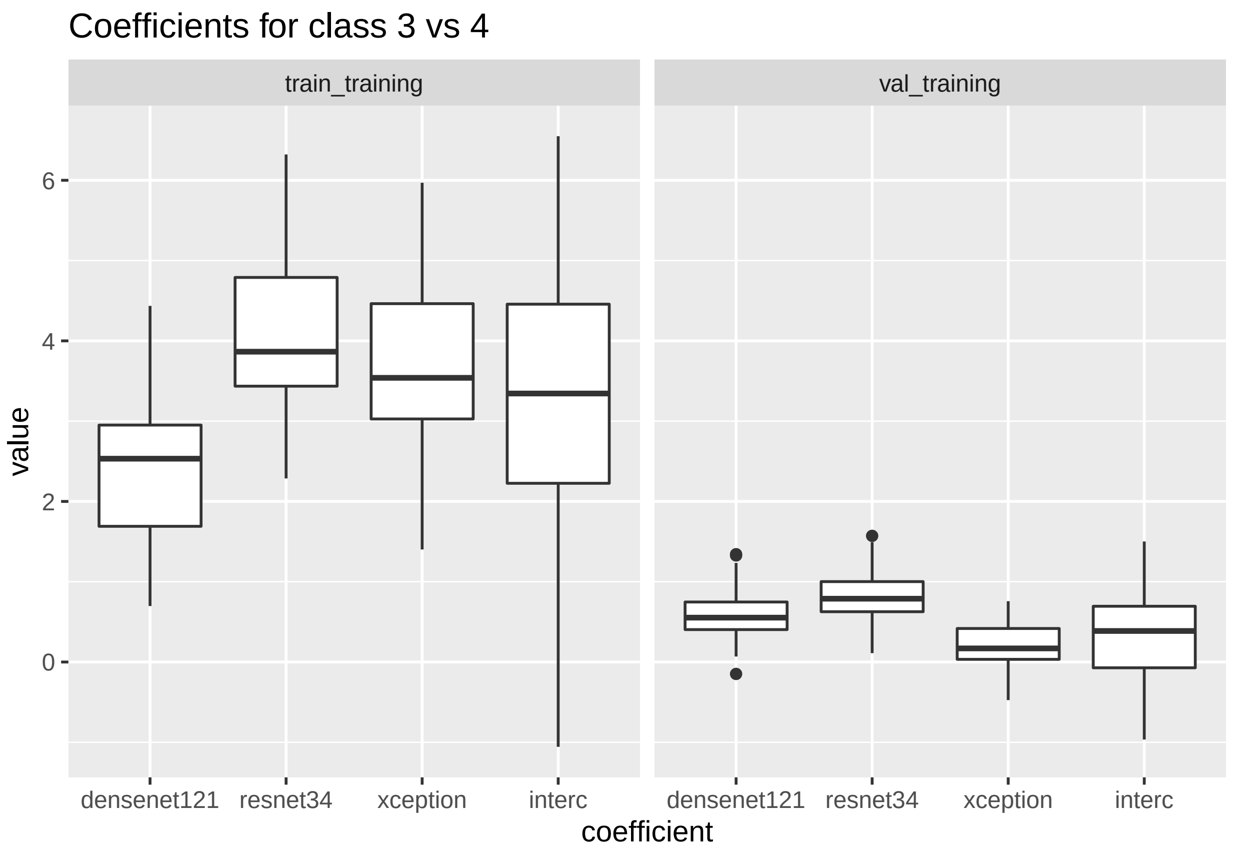


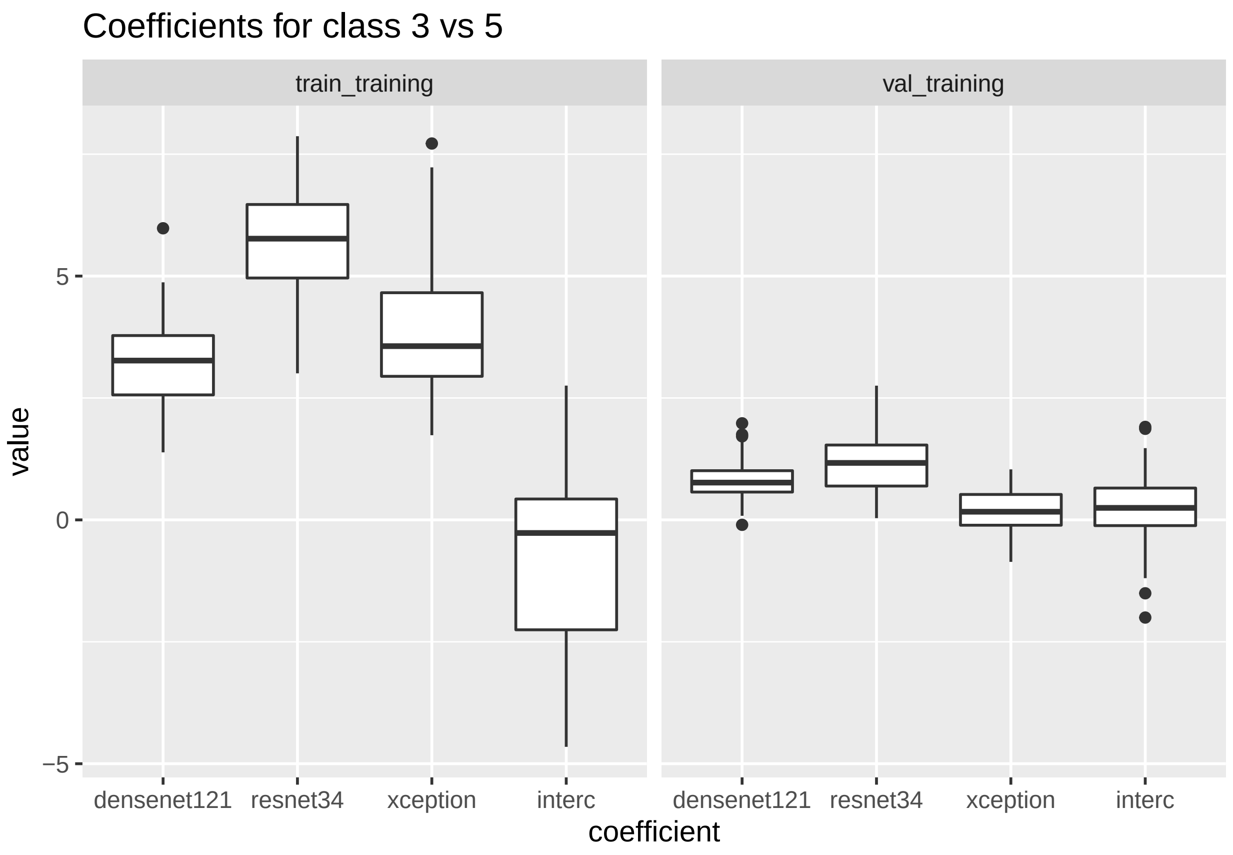
Coefficients for class 2 vs 7
train_training
val_training
value
10
5
0
−5
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

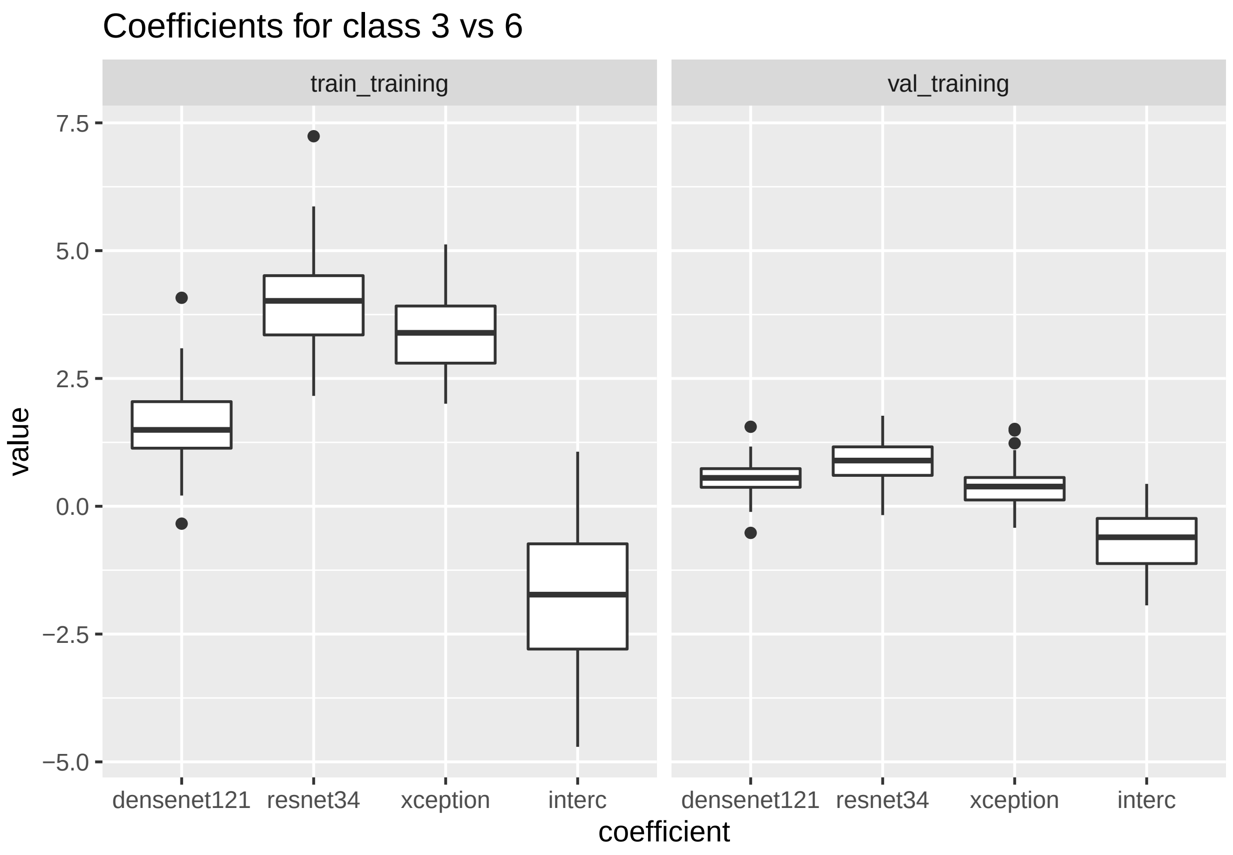

Coefficients for class 2 vs 8
train_training
val_training
value
10
5
0
−5
densenet121
resnet34
xception
interc
coefficient

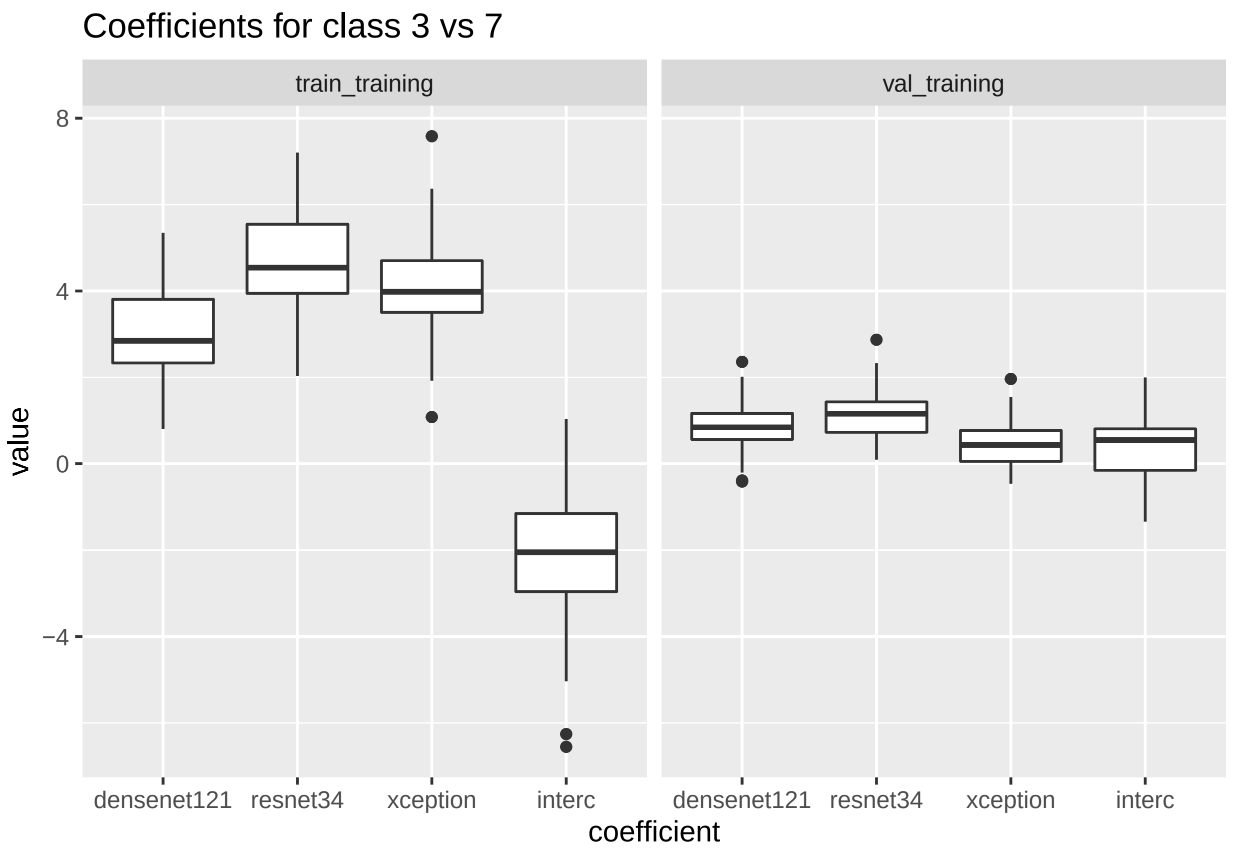

Coefficients for class 2 vs 9
train_training
val_training
value
12
8
4
0
−4
densenet121
resnet34
xception
interc
coefficient

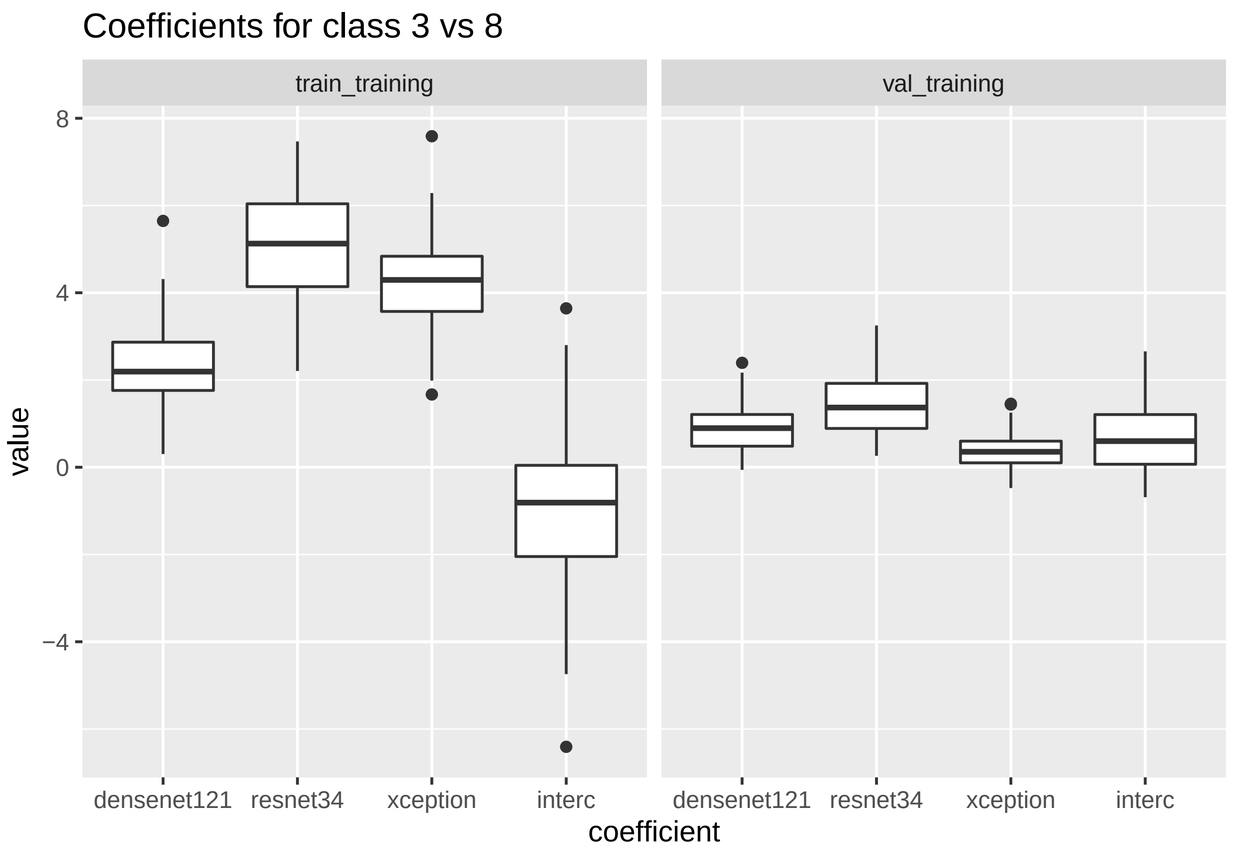

Coefficients for class 2 vs 10
train_training
val_training
10
5
0
−5
value
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

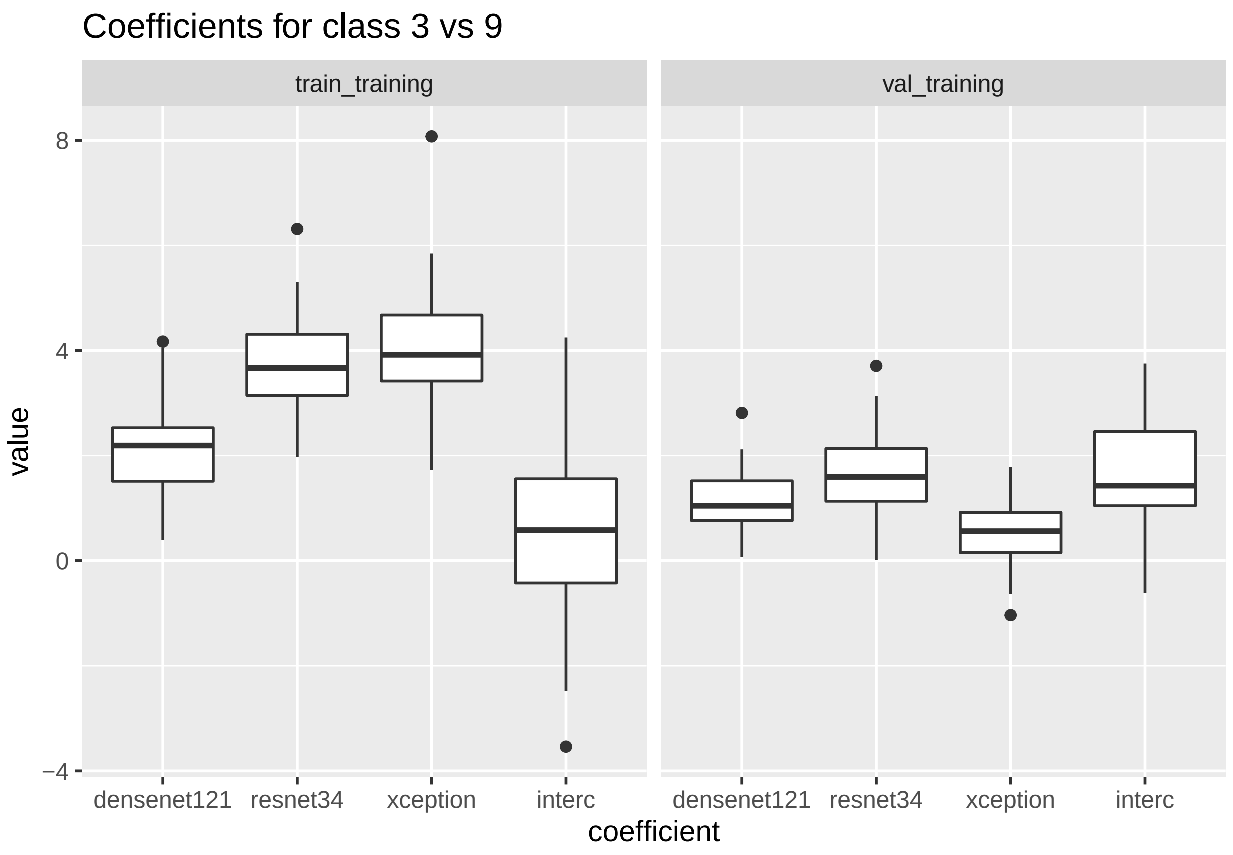

Coefficients for class 3 vs 4
train_training
val_training
value
6
4
2
0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient


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

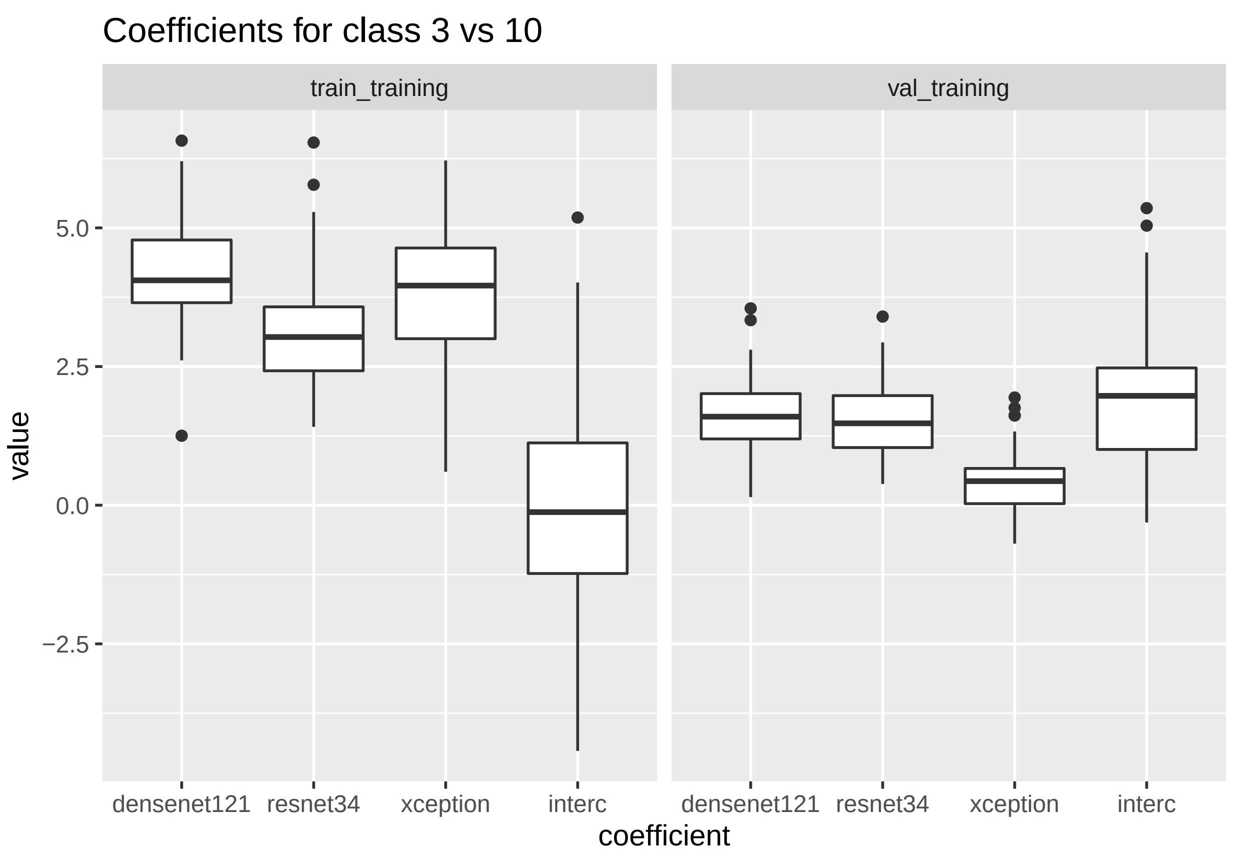

Coefficients for class 3 vs 6
train_training
val_training
value
7.5
5.0
2.5
0.0
−2.5
−5.0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

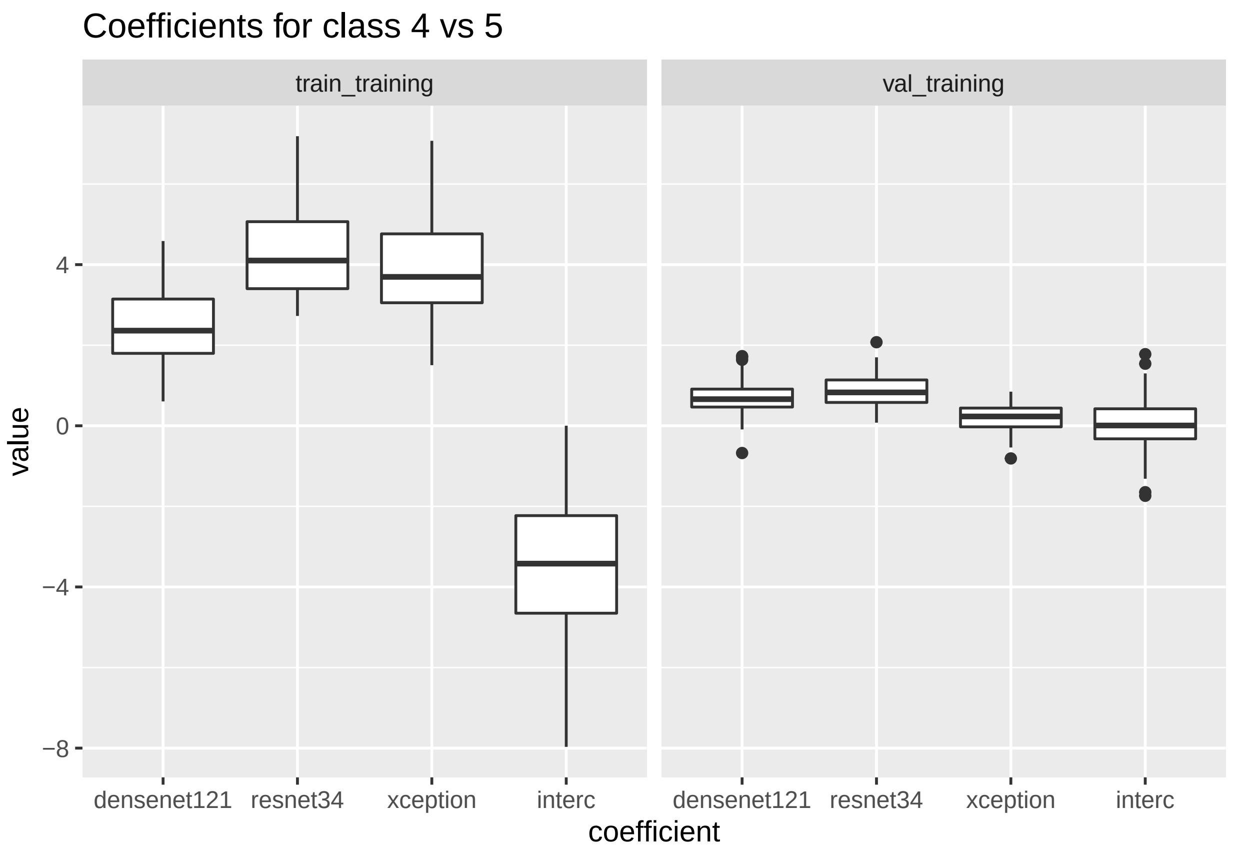

Coefficients for class 3 vs 7
train_training
val_training
value
8
4
0
−4
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

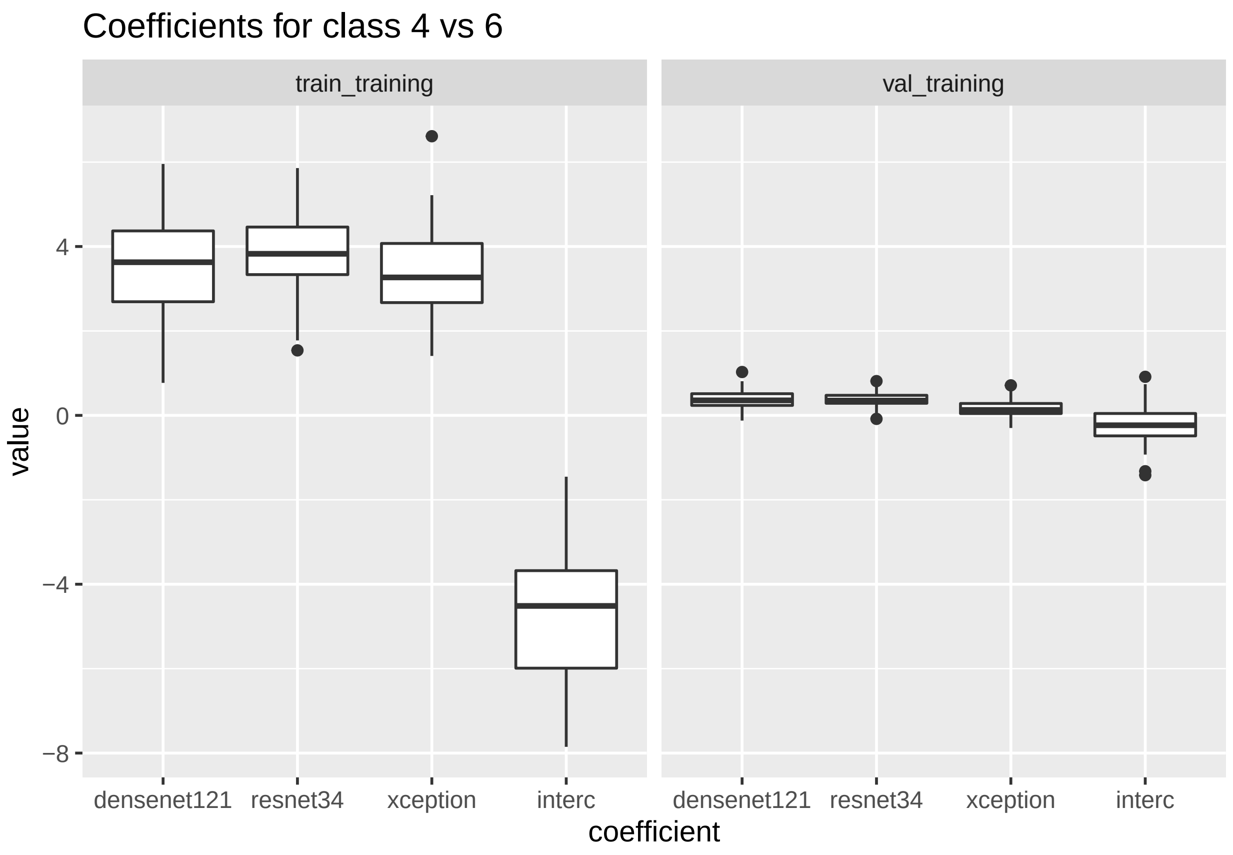

Coefficients for class 3 vs 8
train_training
val_training
value
8
4
0
−4
densenet121
resnet34
xception
interc
coefficient

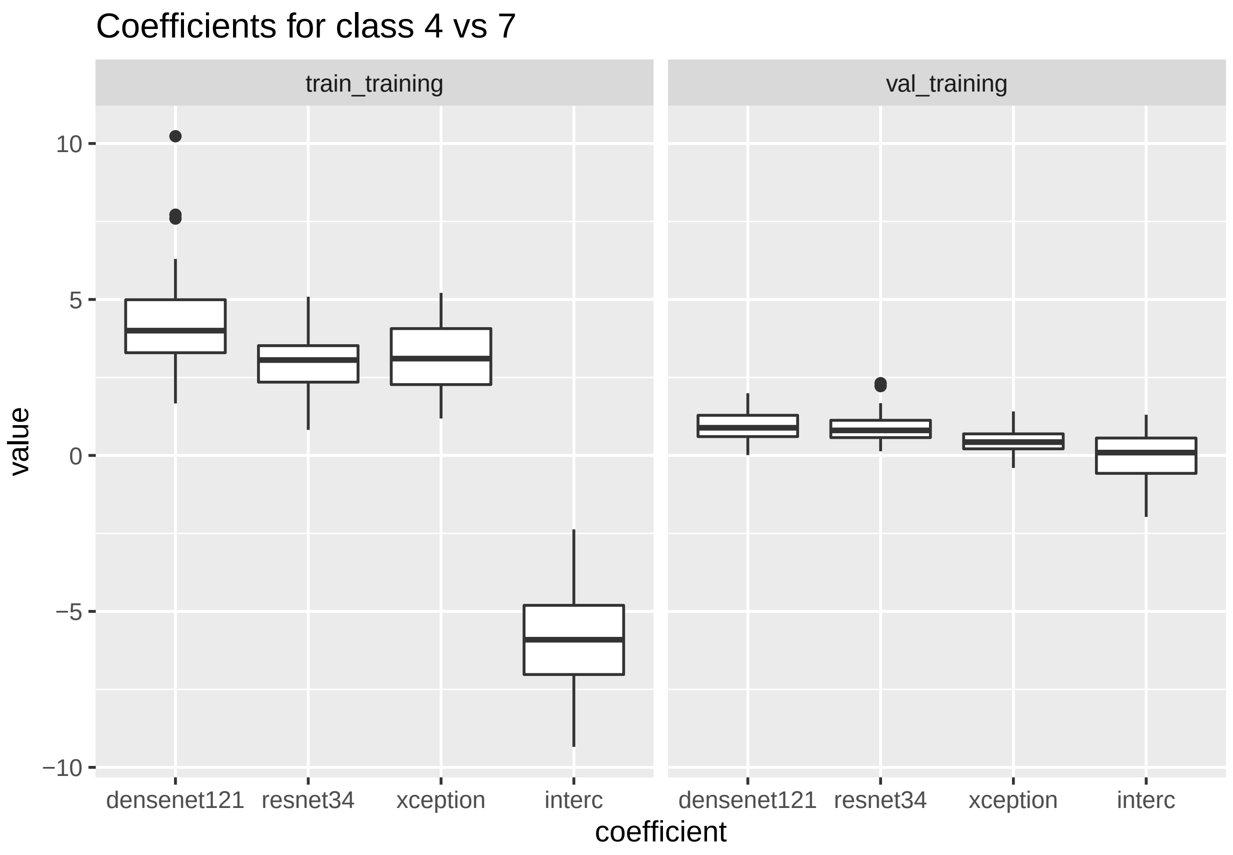


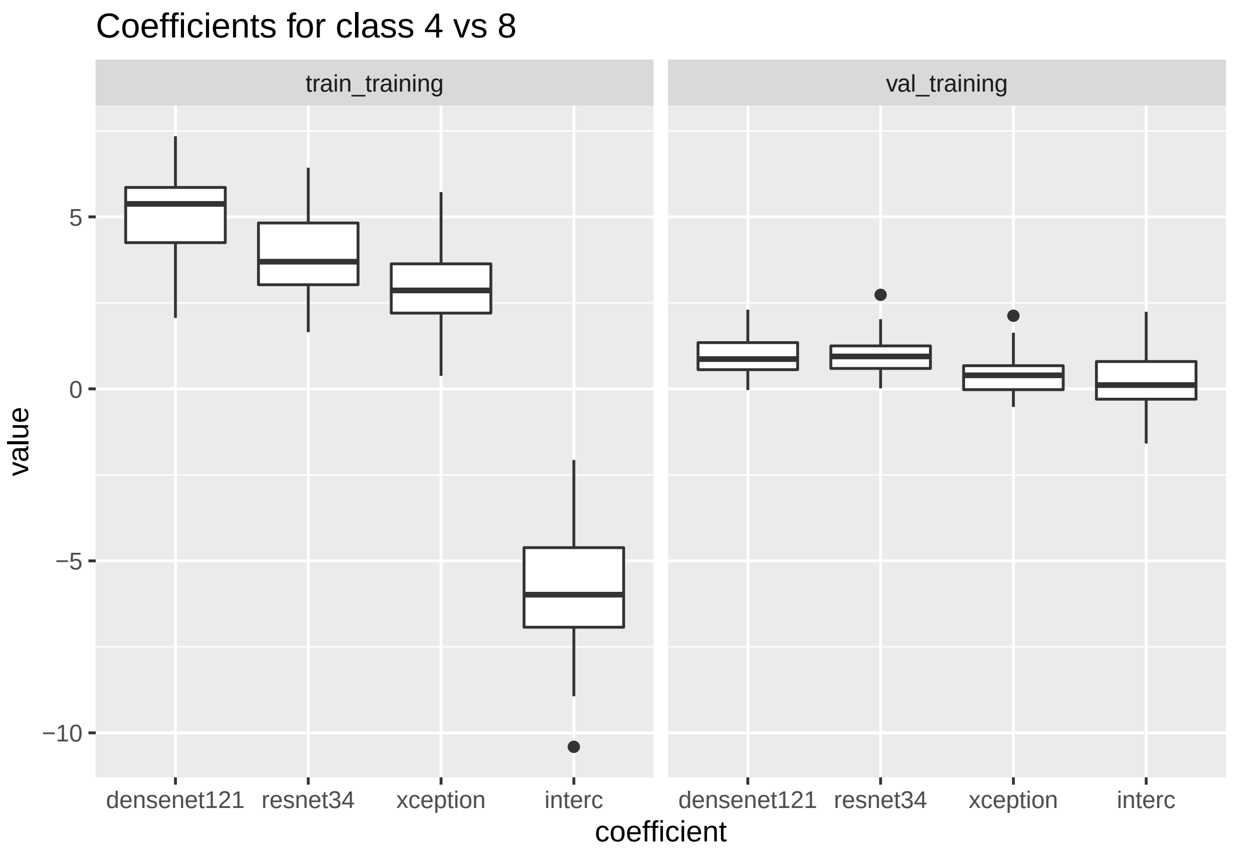
Coefficients for class 3 vs 9
train_training
val_training
value
8
4
0
−4
densenet121
resnet34
xception
interc
coefficient

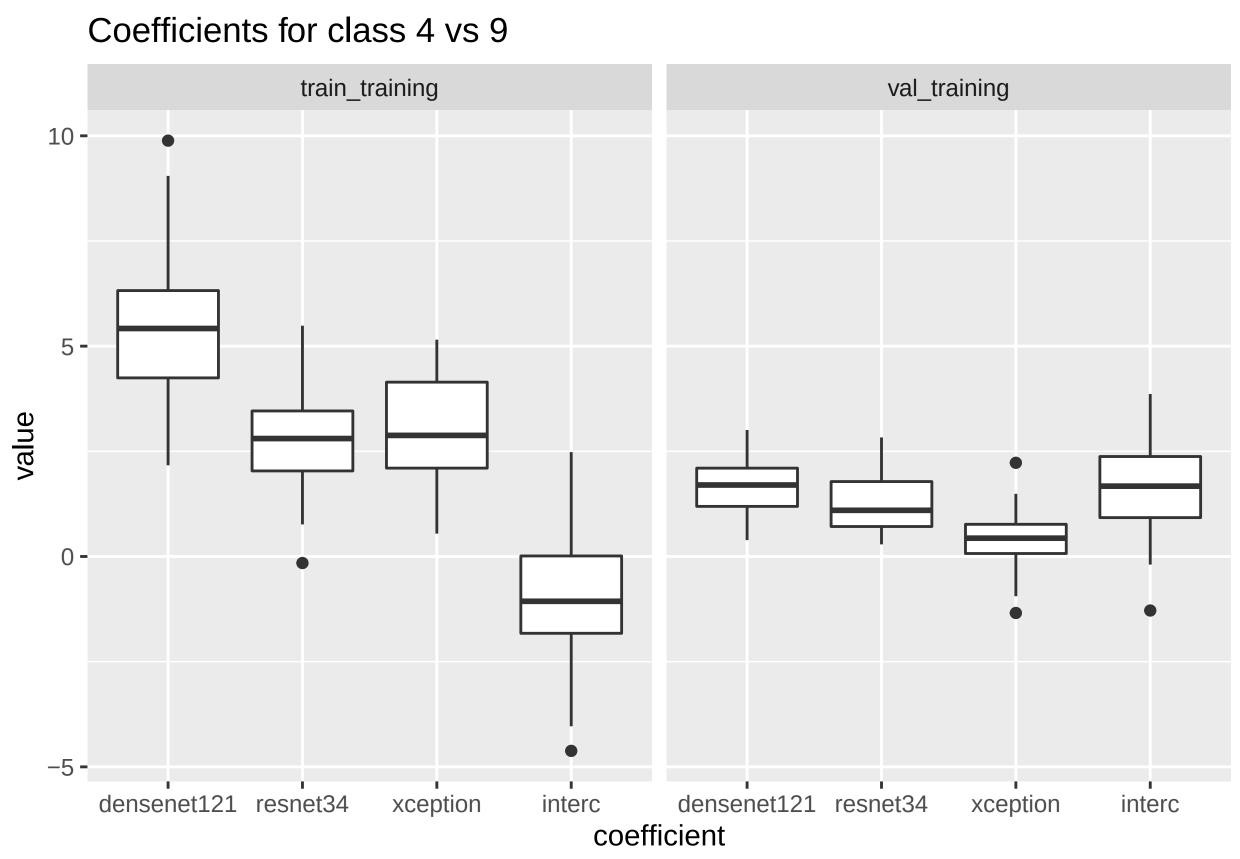

Coefficients for class 3 vs 10
train_training
val_training
value
5.0
2.5
0.0
−2.5
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

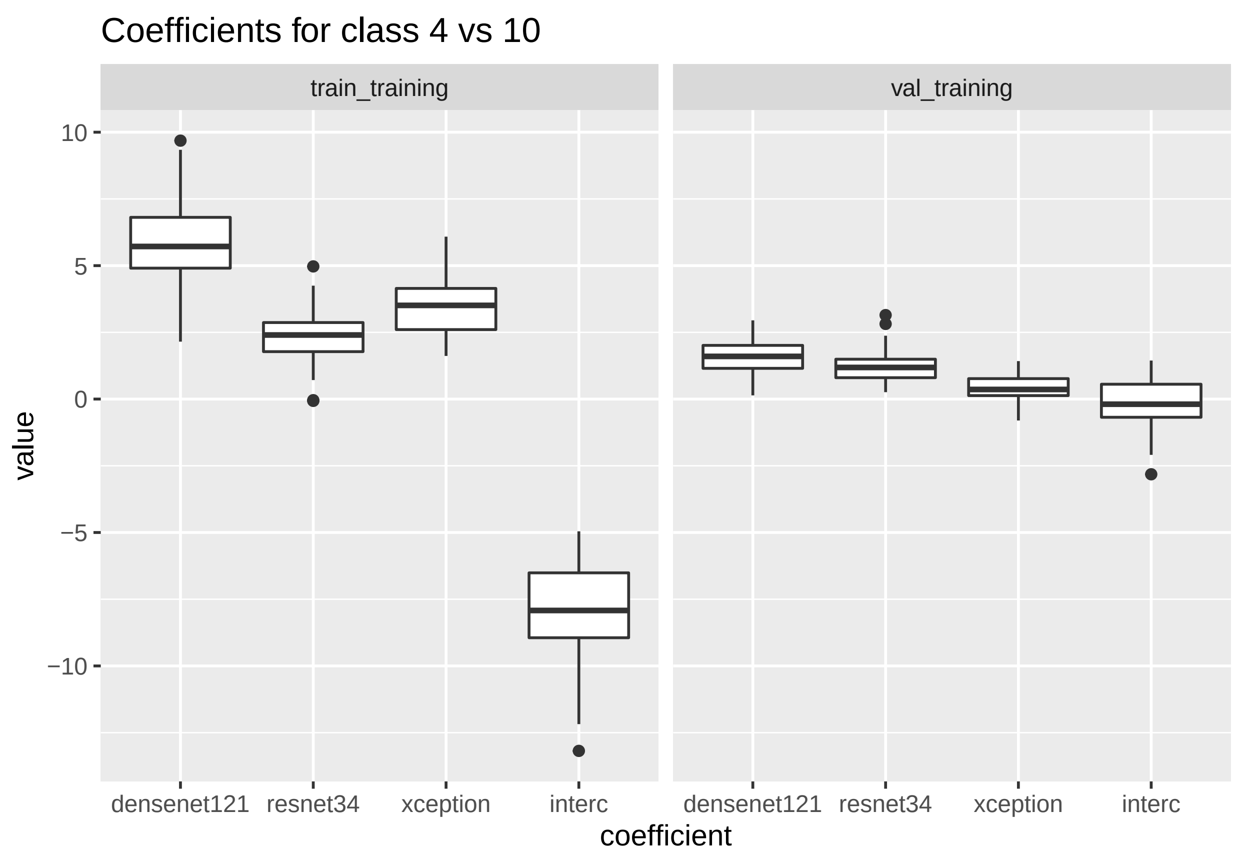

Coefficients for class 4 vs 5
train_training
val_training
value
4
0
−4
−8
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient


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

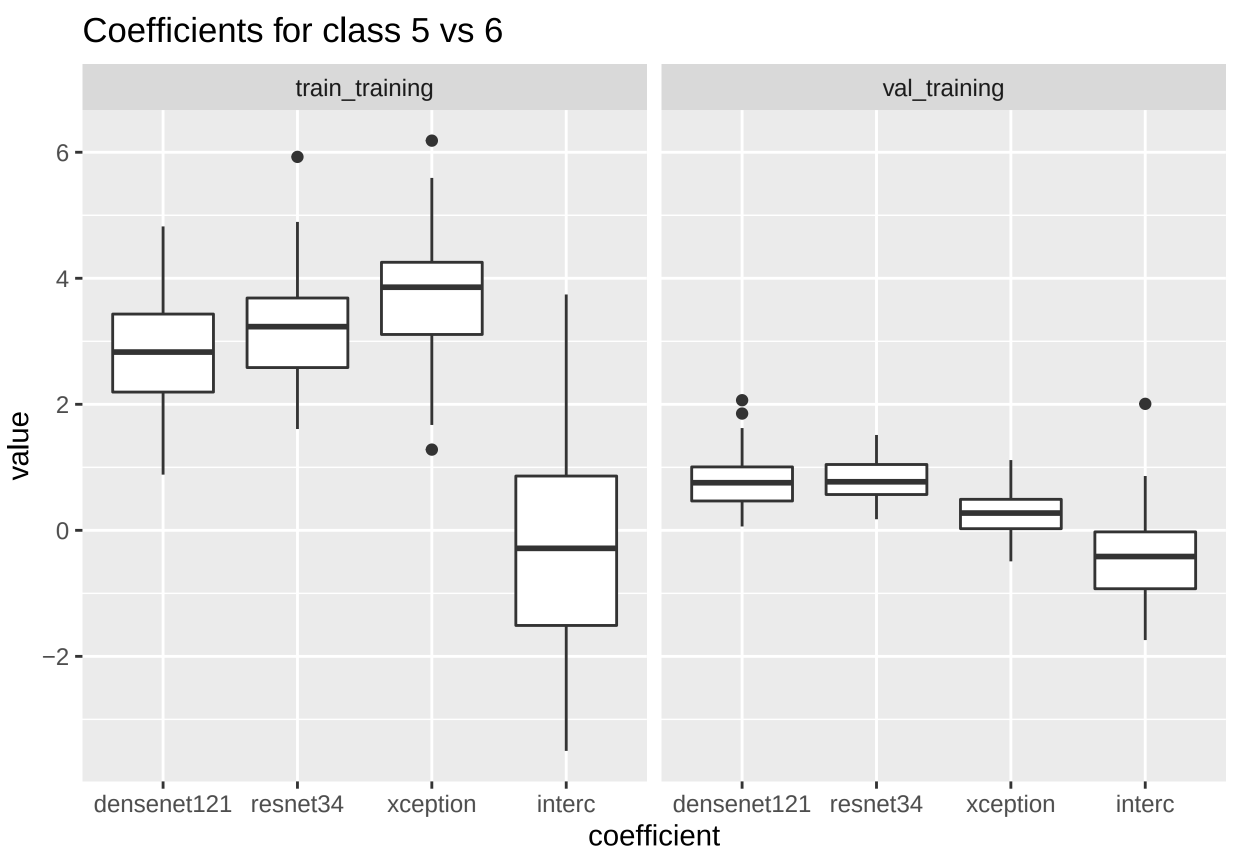

Coefficients for class 4 vs 7
train_training
val_training
value
10
5
0
−5
−10
densenet121
resnet34
xception
interc
coefficient

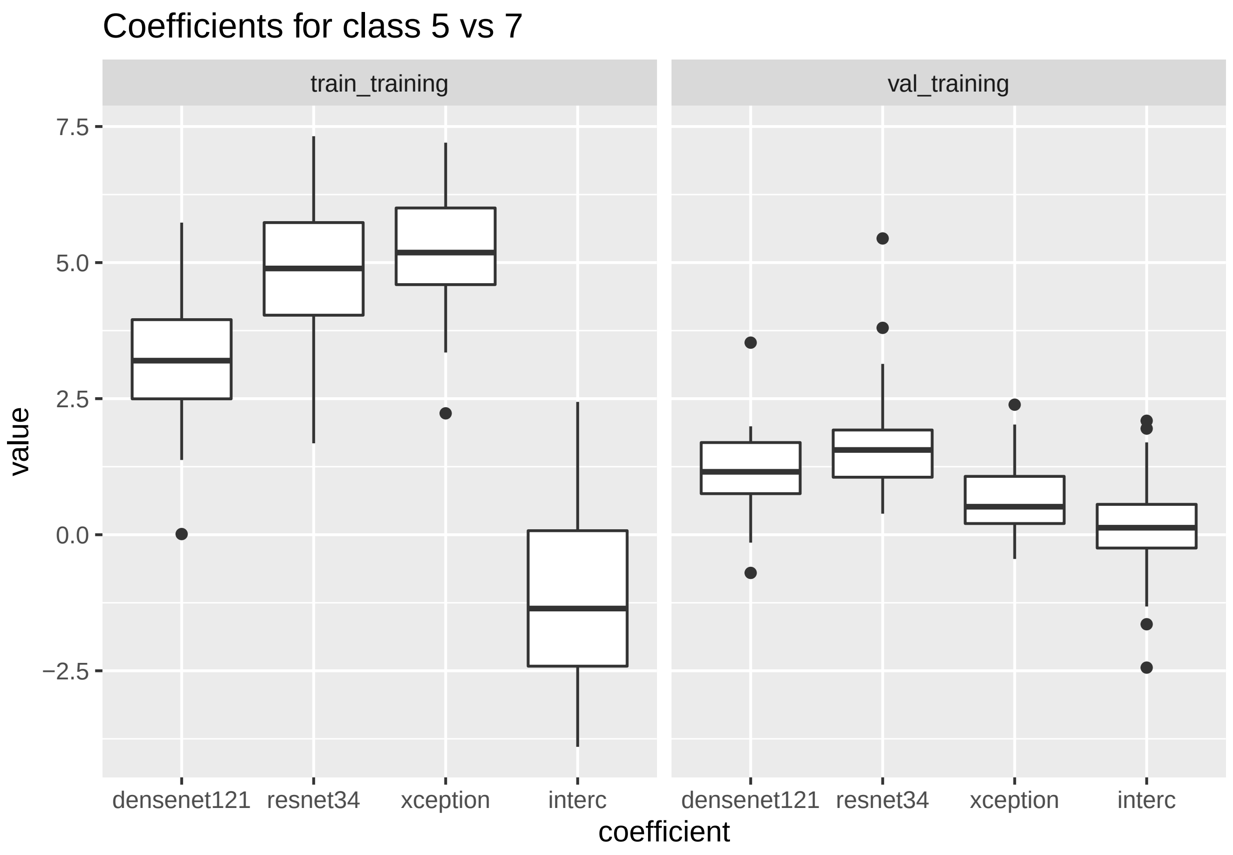

Coefficients for class 4 vs 8
train_training
val_training
value
coefficient
5
0
−5
−10
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc


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

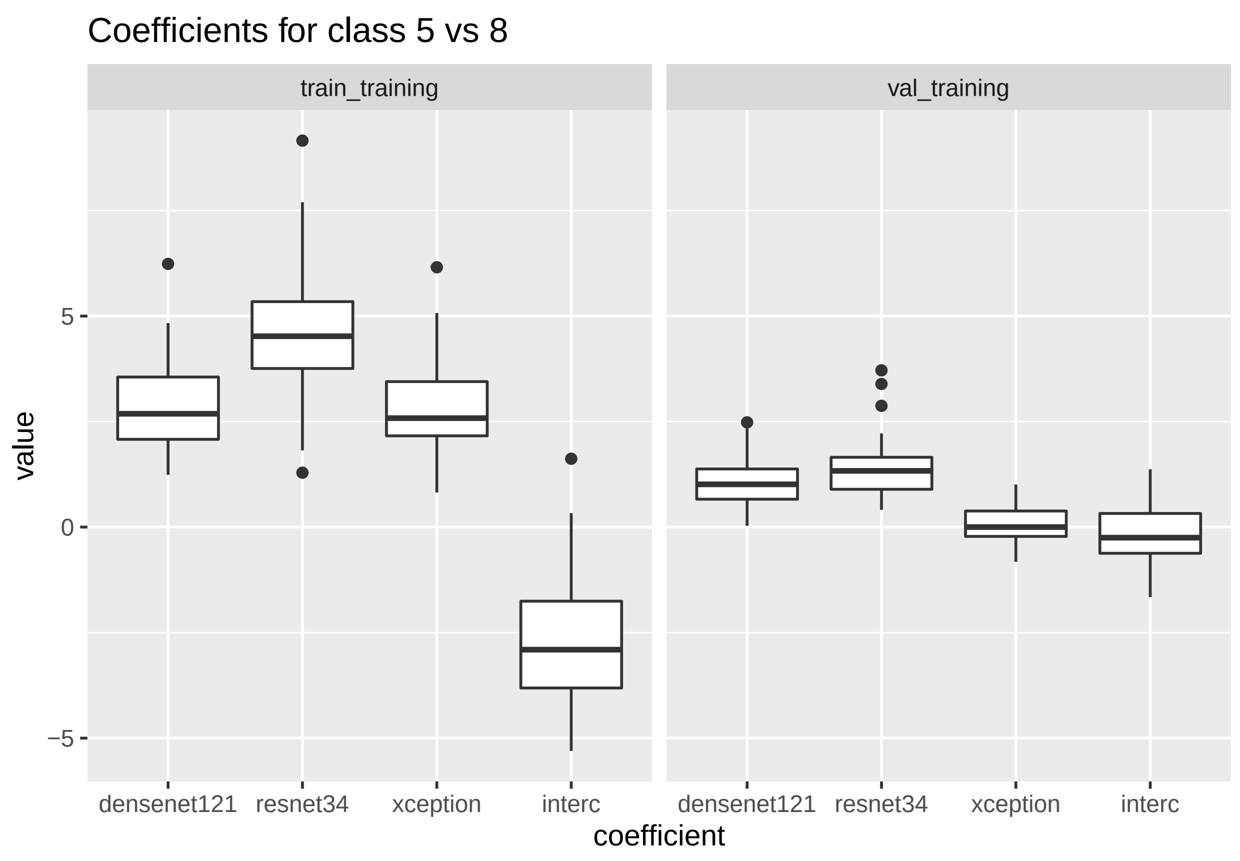


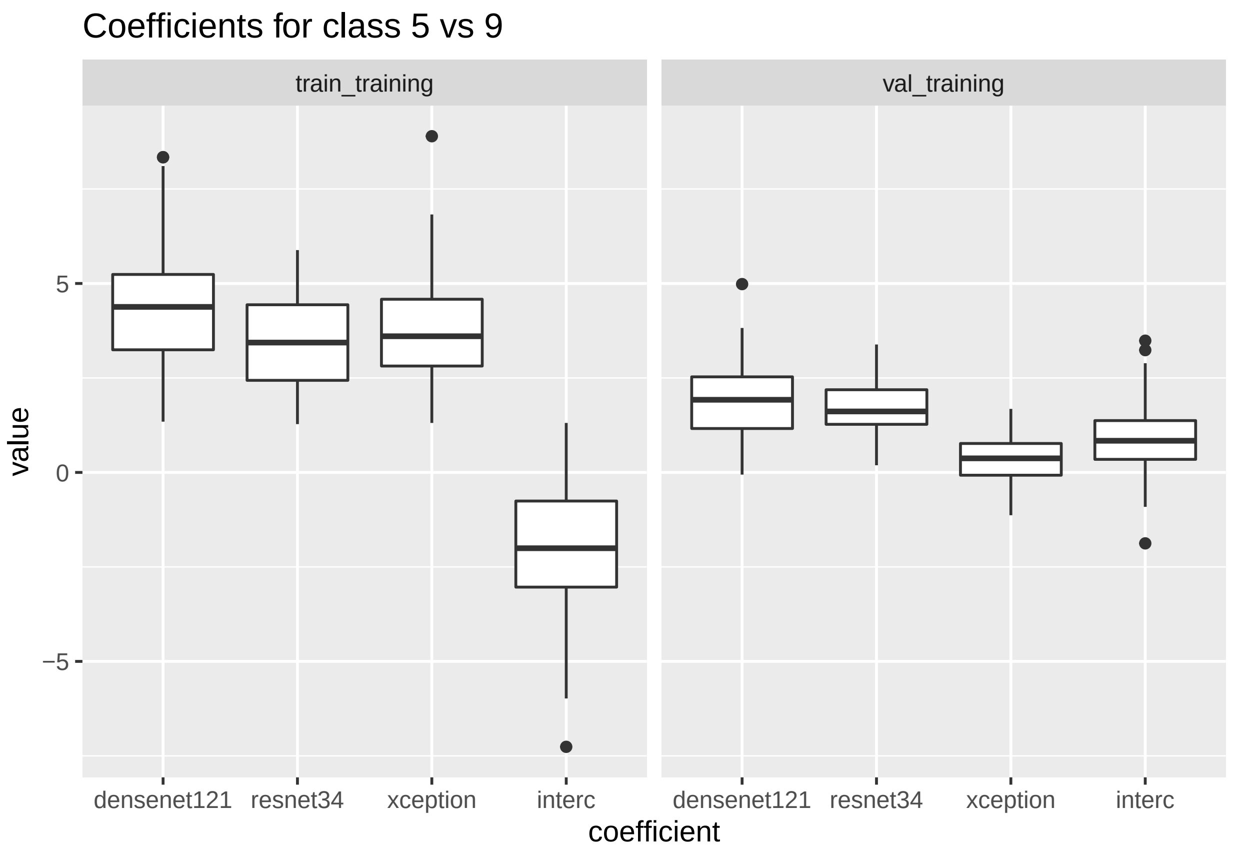
Coefficients for class 4 vs 10
train_training
val_training
10
5
0
−5
−10
value
densenet121
resnet34
xception
interc
coefficient


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

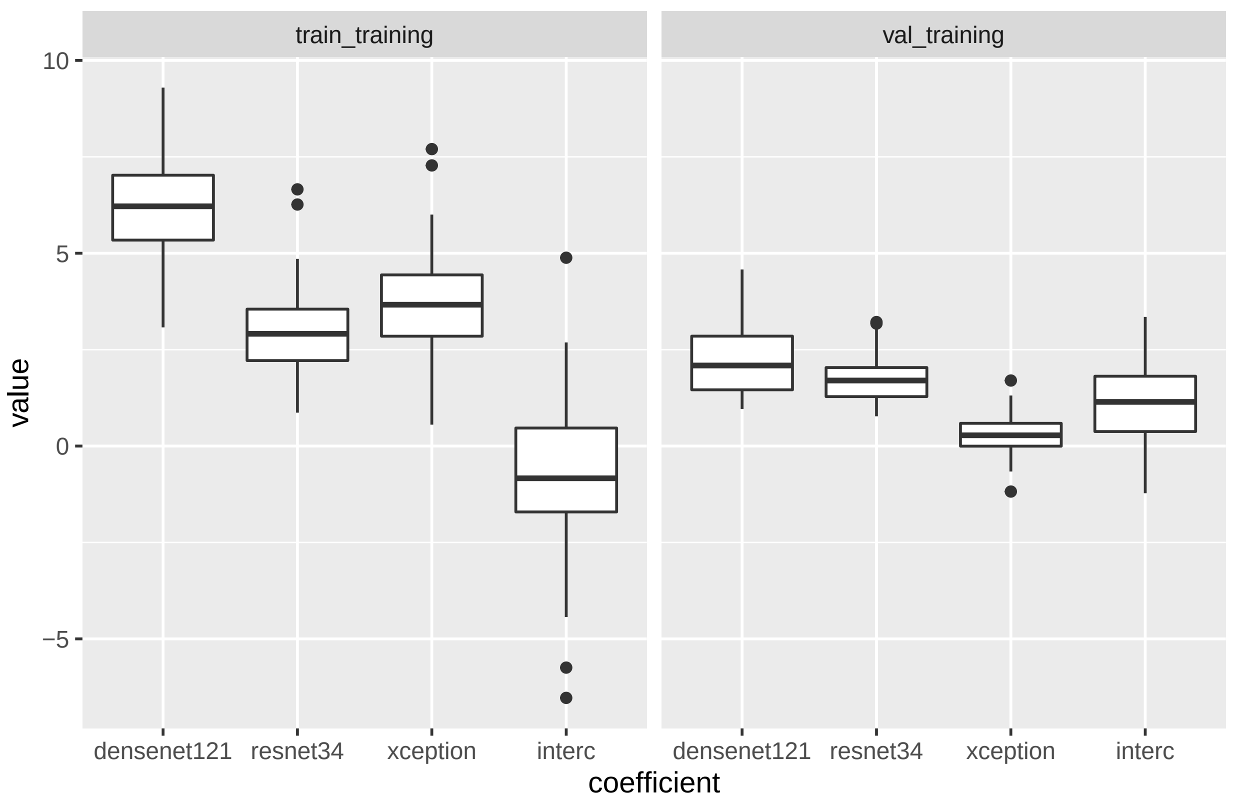


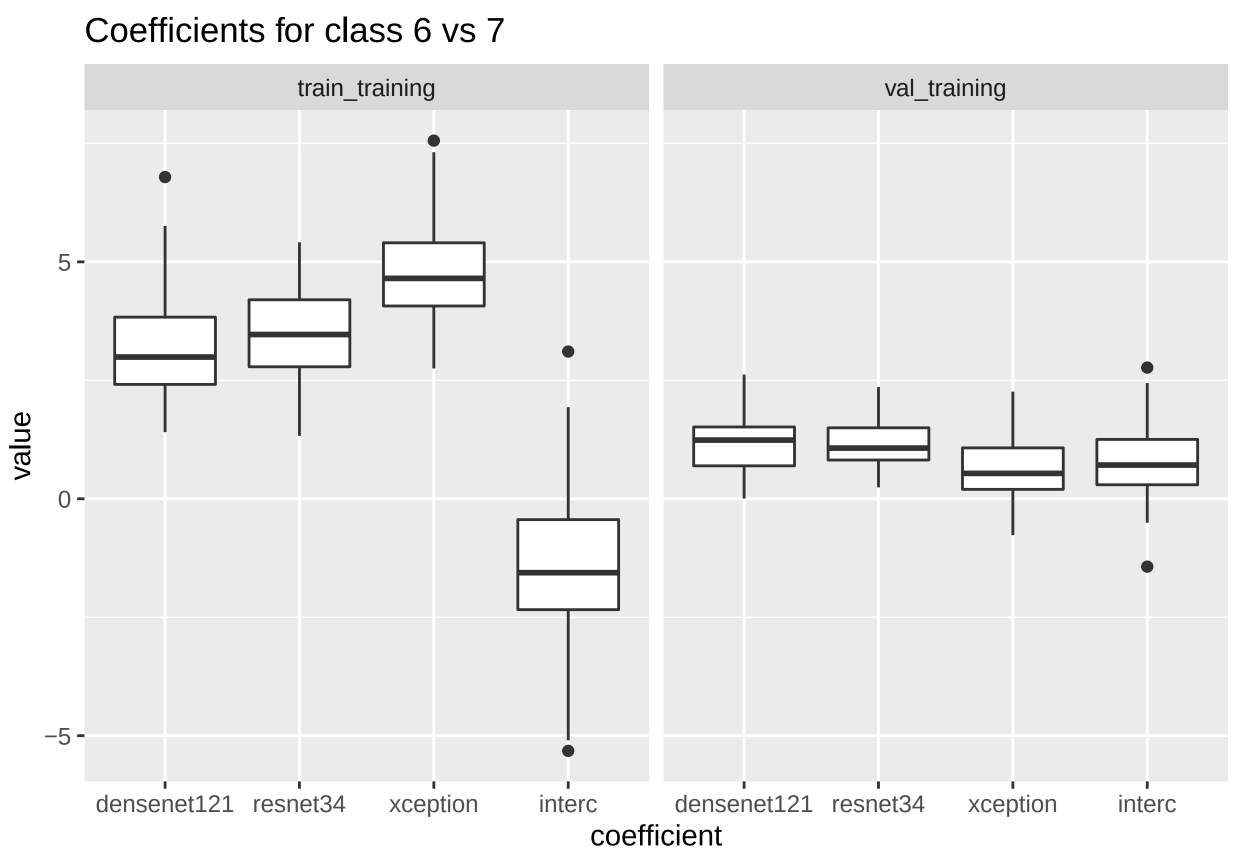
Coefficients for class 5 vs 7
train_training
val_training
value
7.5
5.0
2.5
0.0
−2.5
densenet121
resnet34
xception
interc
coefficient

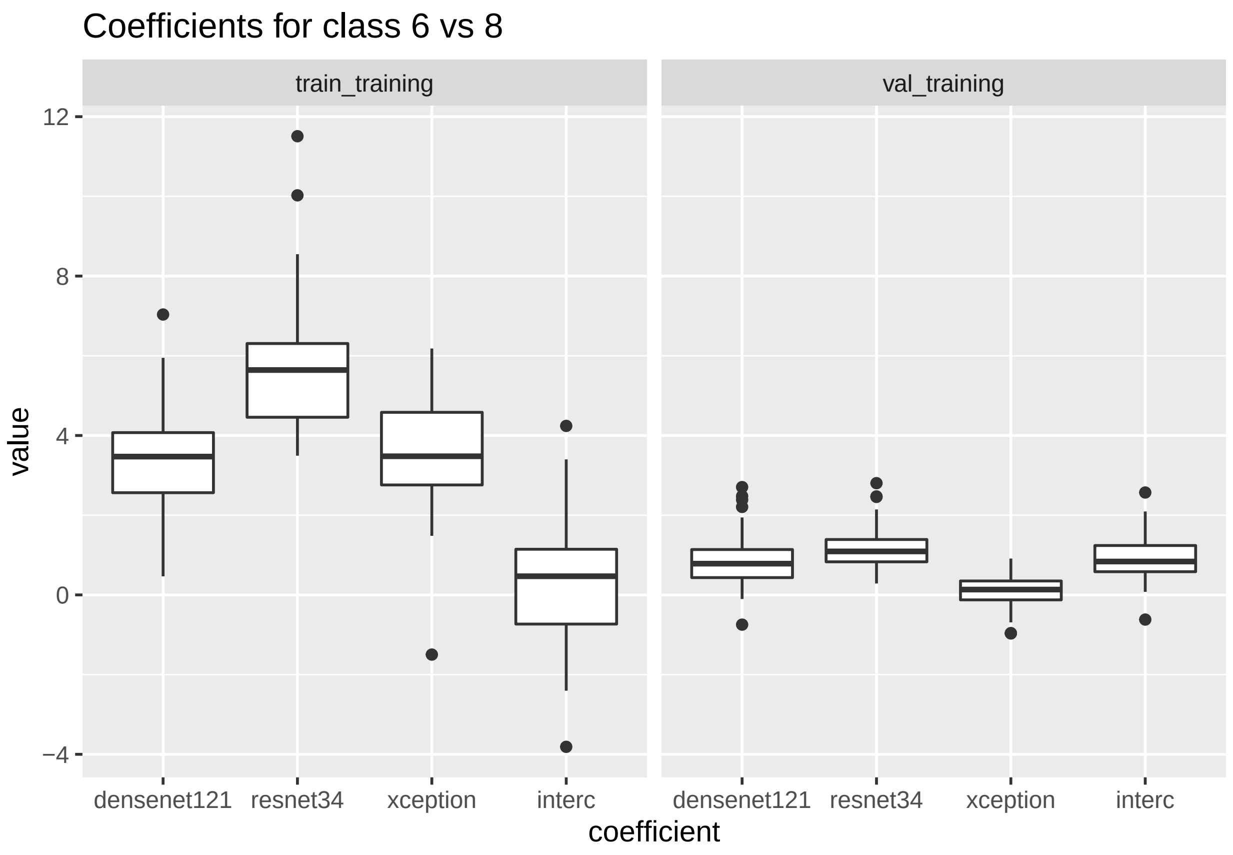

Coefficients for class 5 vs 8
train_training
val_training
value
coefficient
5
0
−5
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc

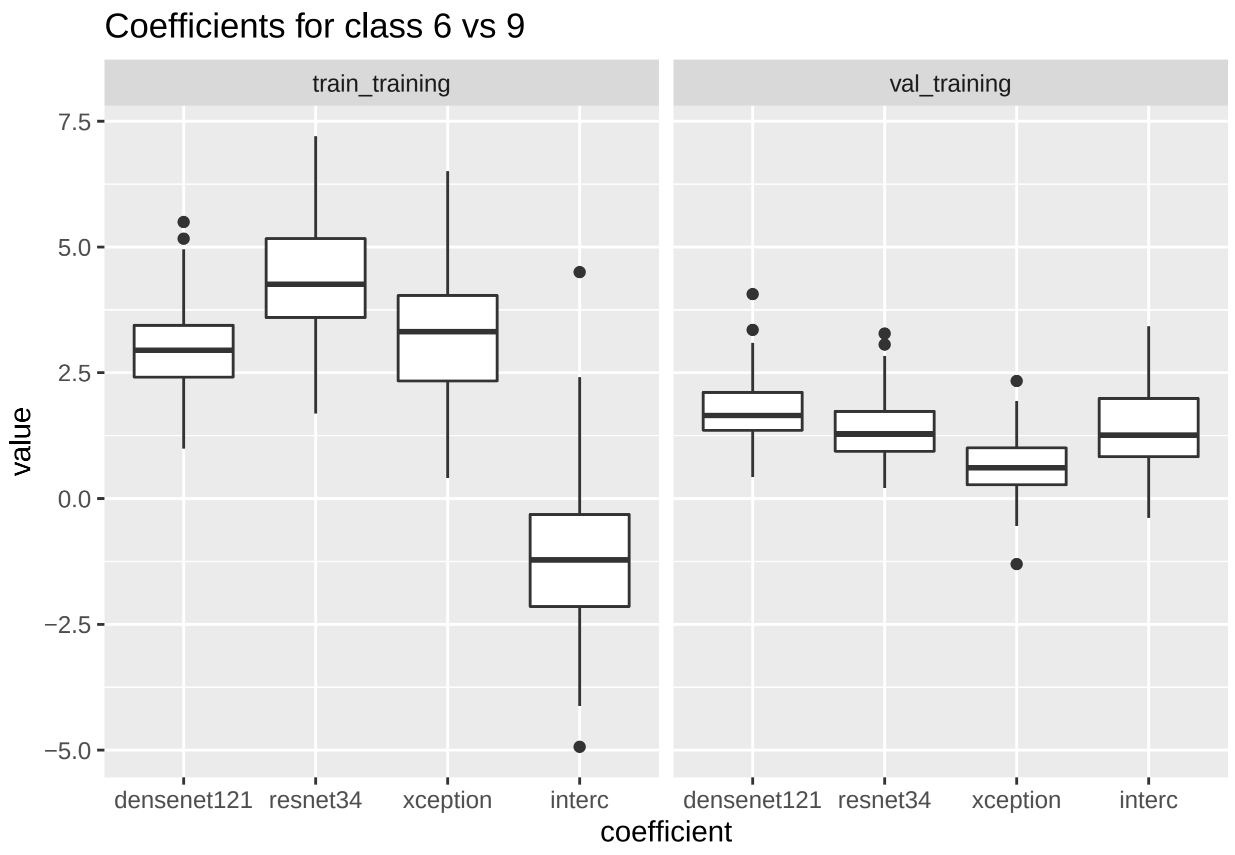

Coefficients for class 5 vs 9
train_training
val_training
value
5
0
−5
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

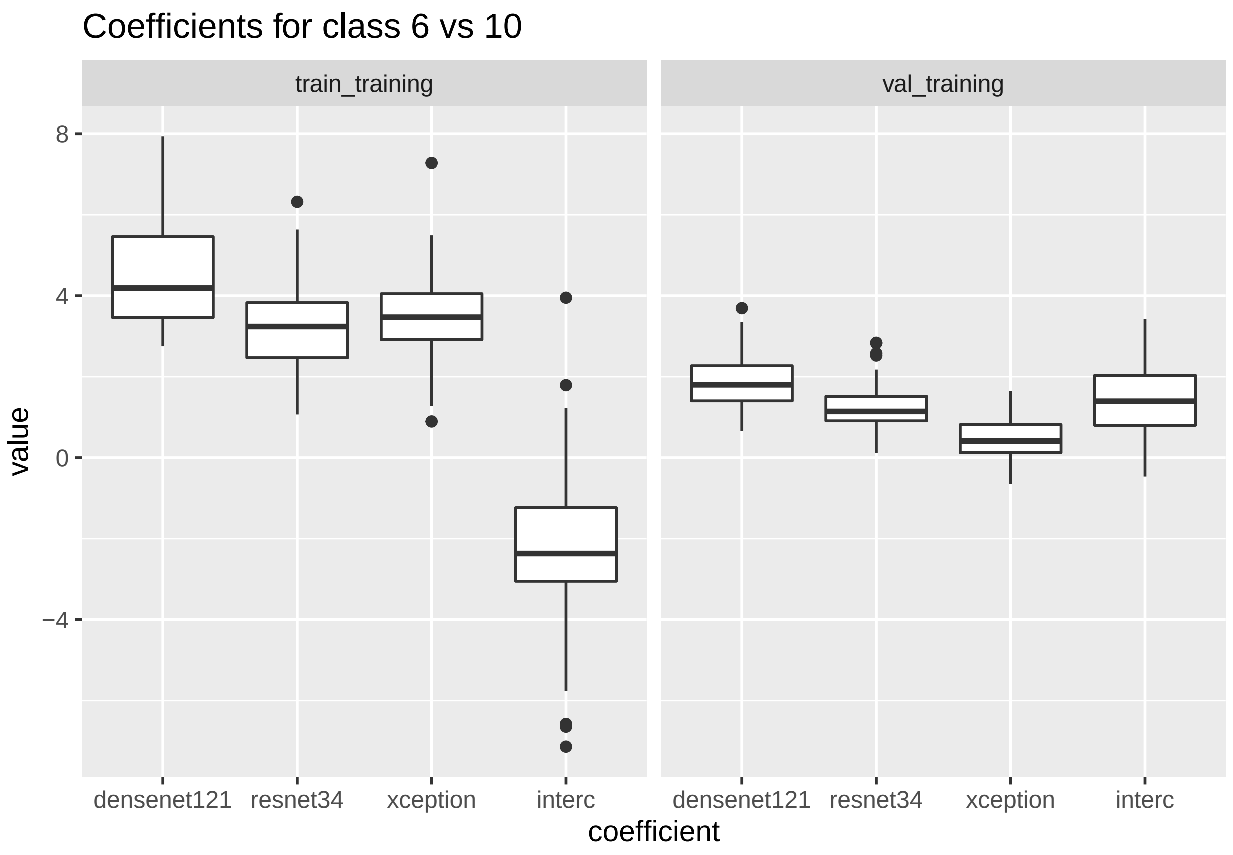
Coefficients for class 5 vs 10

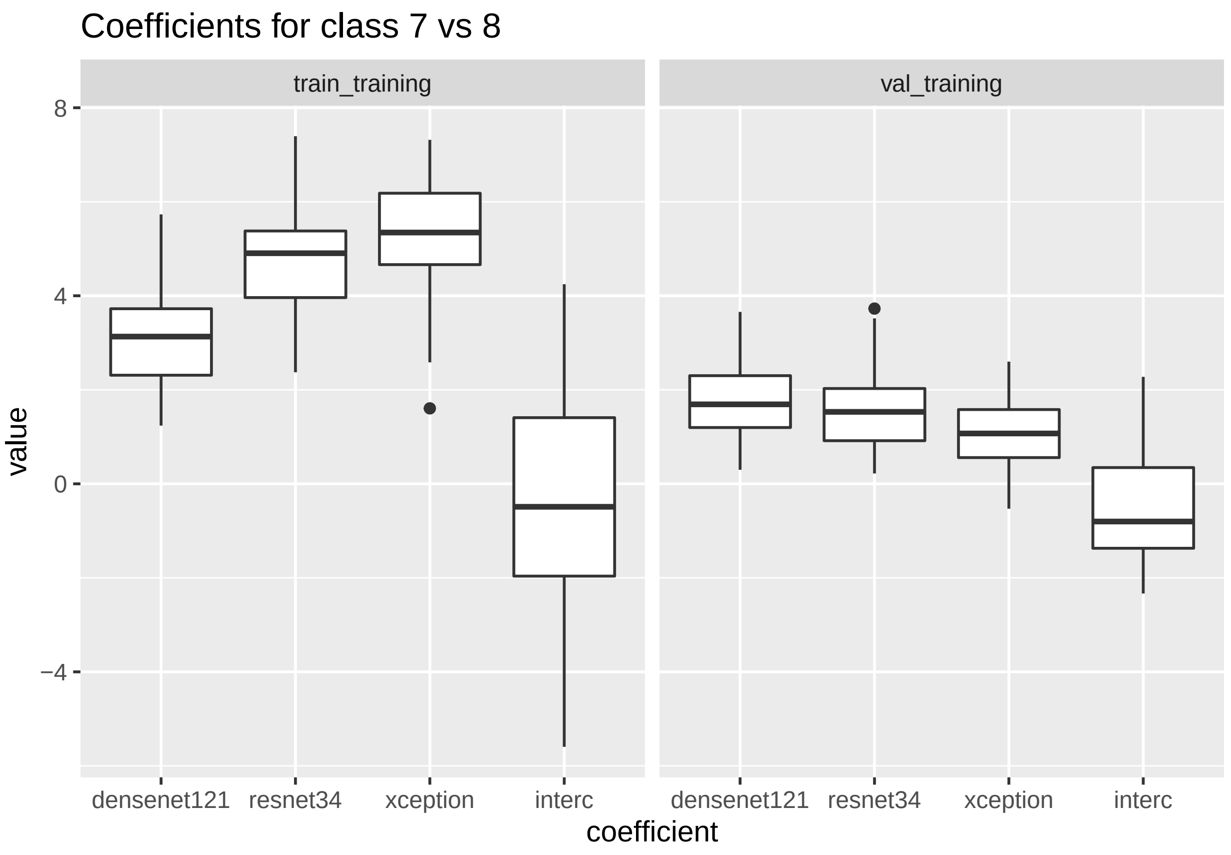


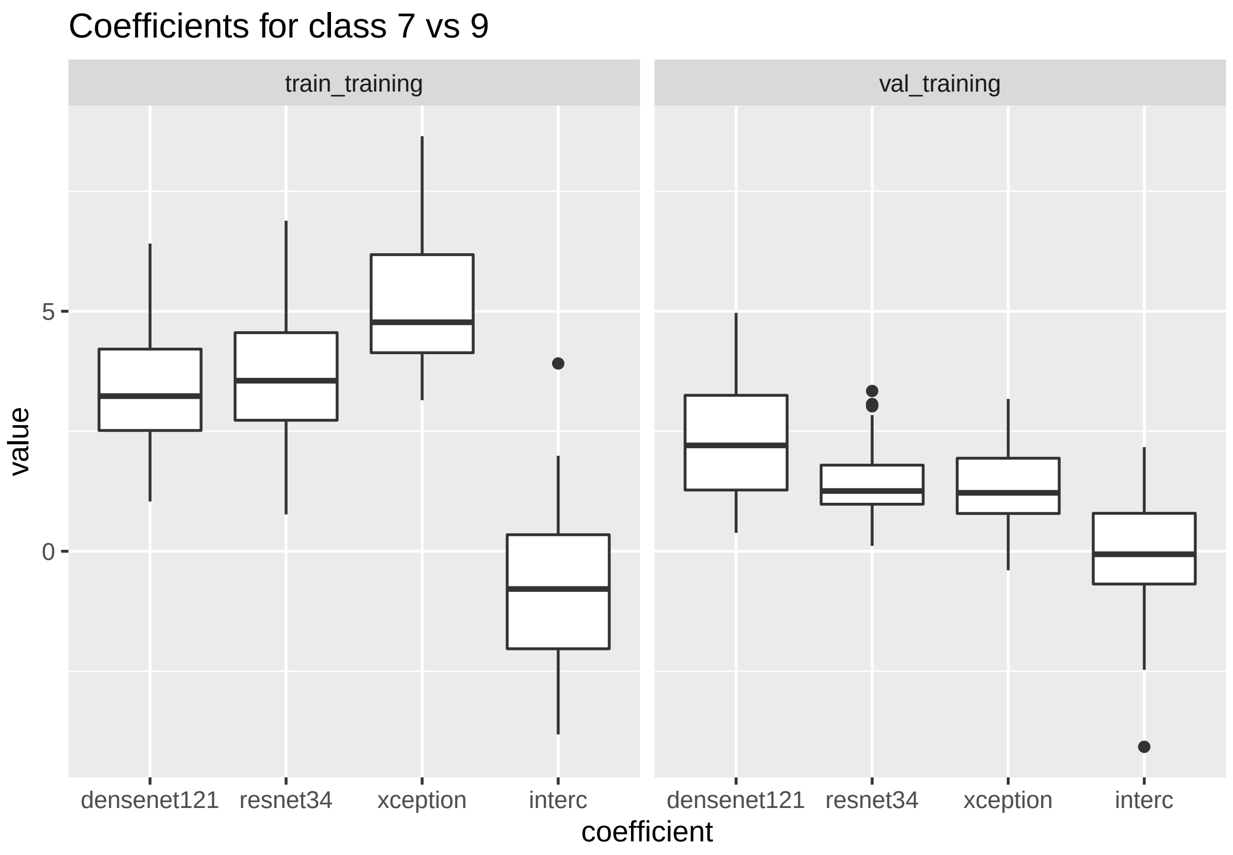

Coefficients for class 6 vs 7
train_training
val_training
value
5
0
−5
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

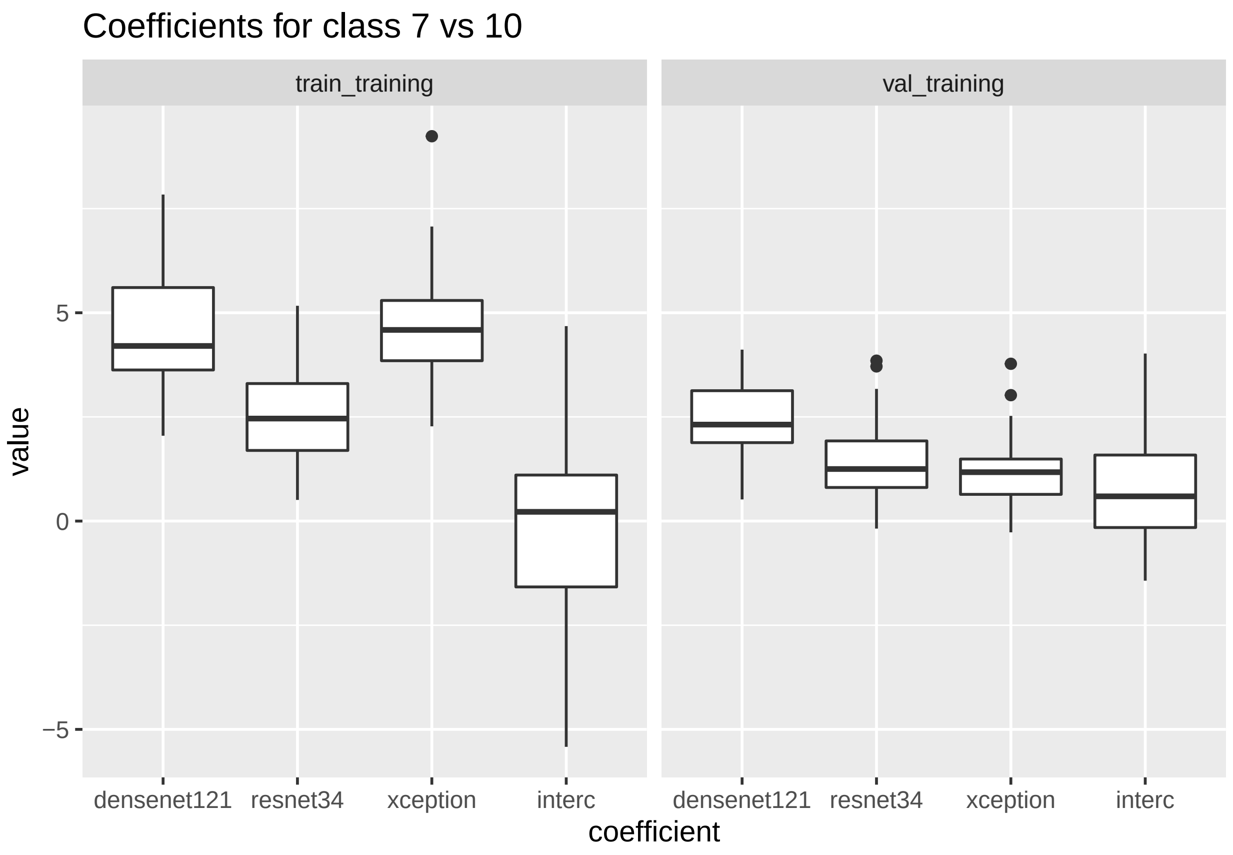

Coefficients for class 6 vs 8
train_training
val_training
value
12
8
4
0
−4
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

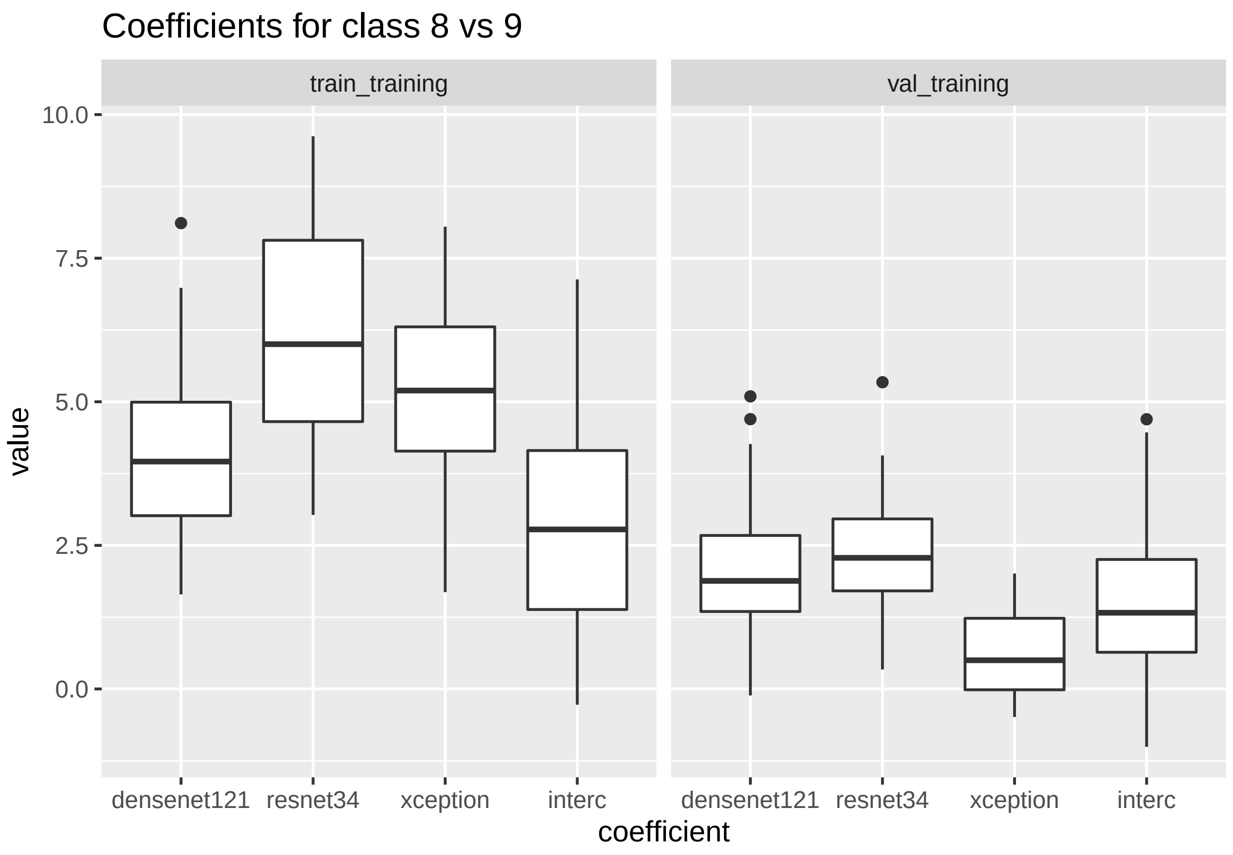

Coefficients for class 6 vs 9
train_training
val_training
value
7.5
5.0
2.5
0.0
−2.5
−5.0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient

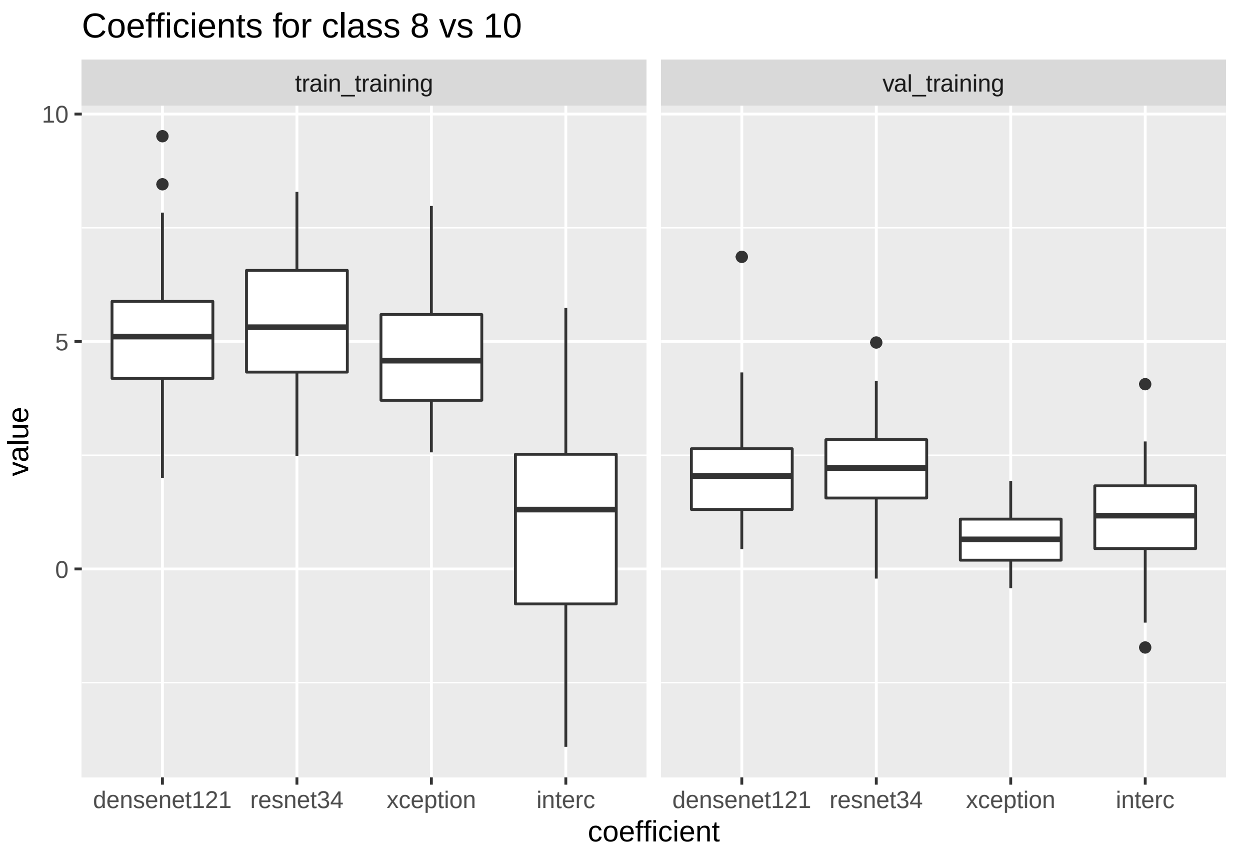

Coefficients for class 6 vs 10
train_training
val_training
value
8
4
0
−4
densenet121
resnet34
xception
interc
coefficient


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

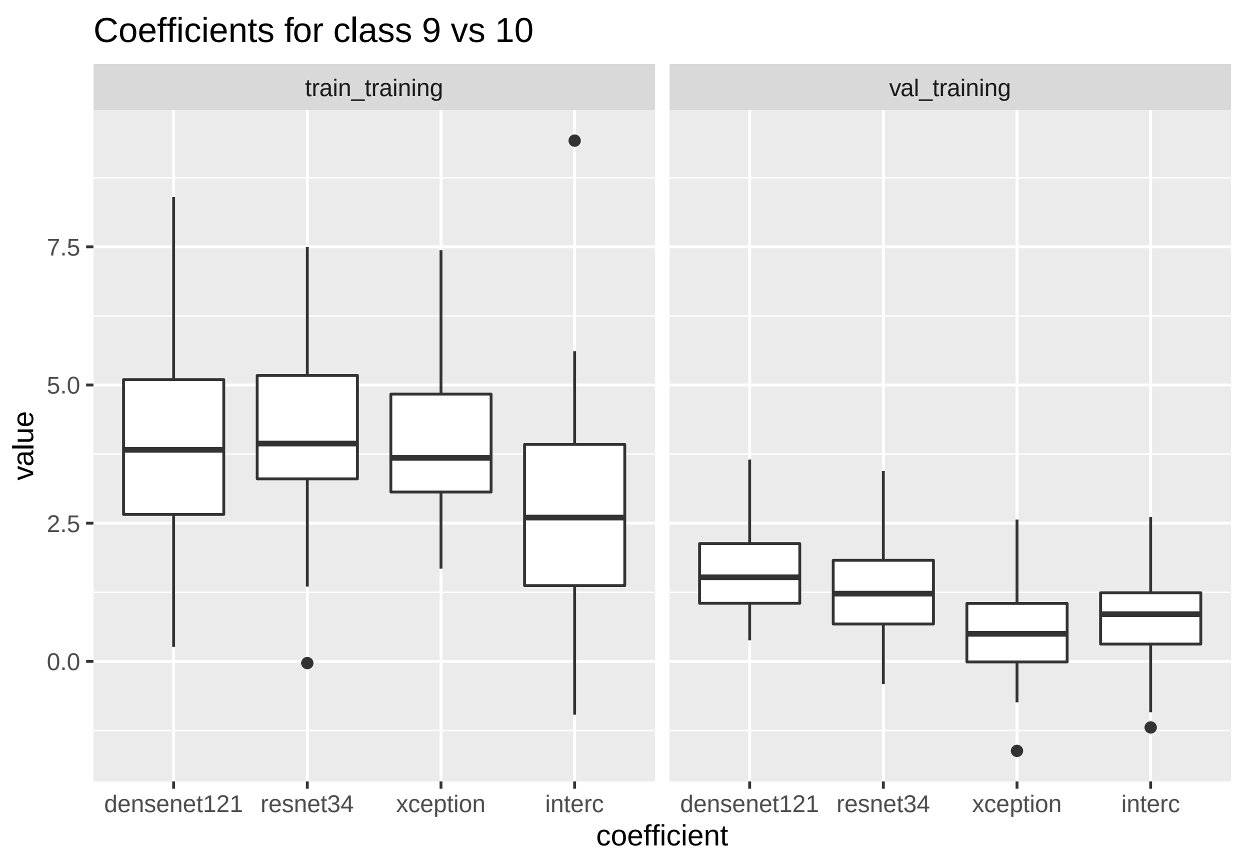

Coefficients for class 7 vs 9
train_training
val_training
value
5
0
densenet121
resnet34
xception
interc
densenet121
resnet34
xception
interc
coefficient


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

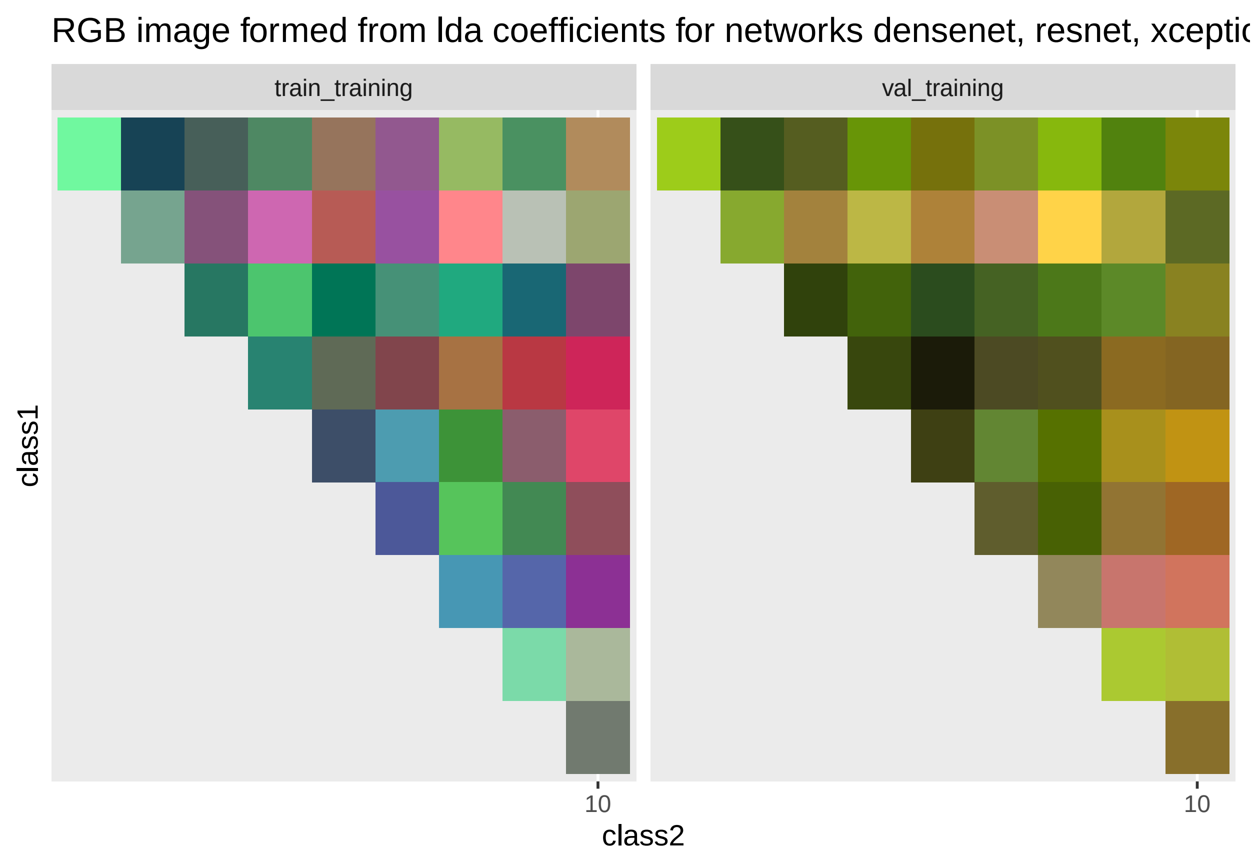


```
avg_lda_coefs <- lda_coefs %>% filter(coefficient != "interc") %>% group_by(class1, class2, precision,
```

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

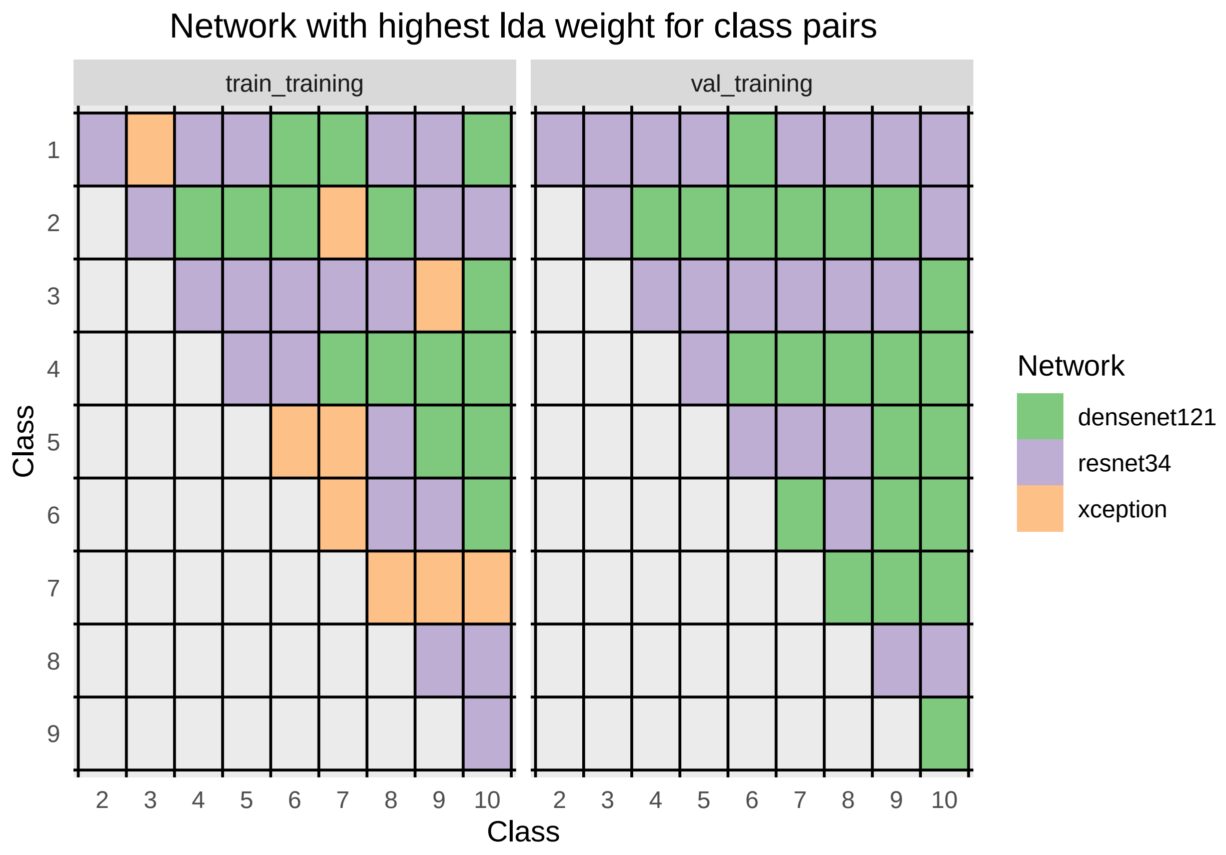
```
avg_lda_coefs_vt <- avg_lda_coefs %>% filter(train_type=="val_training")
avg_lda_coefs_tt <- avg_lda_coefs %>% filter(train_type=="train_training")
avg_lda_coefs_vt$value <- avg_lda_coefs_vt$value - min(avg_lda_coefs_vt$value)
avg_lda_coefs_vt$value <- avg_lda_coefs_vt$value / max(avg_lda_coefs_vt$value)
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
```



Correspondence between colors and networks is red - densenet, green - resnet, blue - xception.

```
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
```



Densenet is far less dominating in this experiment than in visualizations_ensemble_outputs_CIF10. Other networks seem to be more competitive when training is done just on half of CIFAR 10 training set.