

Outputs inspection 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. These networks were trained in 30 replications. In each replication, 500 samples from the training set were randomly extracted and formed validation set. In each replication, we trained two ensembles on the outputs of neural networks. First one was trained on randomly chosen subset, of size 500, of nn training set, second on the extracted validation set. 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_c10"
```

```
repls <- 0:29
```

```
classes <- 10
```

```
nets_outputs <- load_network_outputs(base_dir, repls)
```

```
ens_outputs <- load_ensemble_outputs(base_dir, repls)
```

```
net_results <- read.csv(file.path(base_dir, "net_accuracies.csv"))
```

```
ens_results <- read.csv(file.path(base_dir, "ensemble_accuracies.csv"))
```

For clearer visualization, we will plot just the predicted probability of the correct class for all the methods.

```
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)
```

```

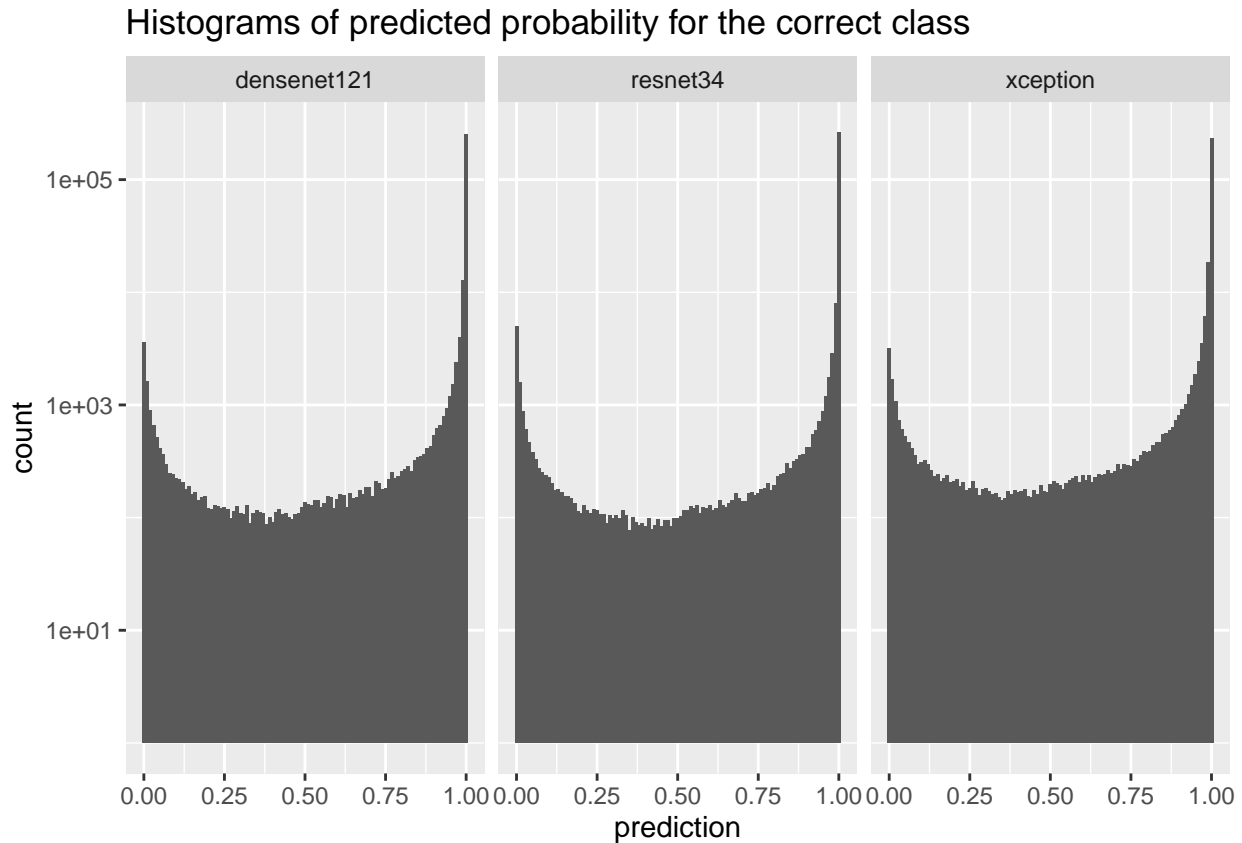
nets_test_cor_probs <- melt(nets_test_cor_probs)
nets_test_cor_probs <- nets_test_cor_probs[, c(-3, -4)]
names(nets_test_cor_probs) <- c("replication", "network", "prediction")
nets_test_cor_probs$network <- as.factor(nets_test_cor_probs$network)
levels(nets_test_cor_probs$network) <- nets_outputs$networks

```

```

nets_cor_preds_histo <- ggplot(data=nets_test_cor_probs) + geom_histogram(mapping=aes(x=prediction), binwidth=0.01) +
  ggtitle("Histograms of predicted probability for the correct class") + facet_wrap(~network) + scale_y_log10()
nets_cor_preds_histo

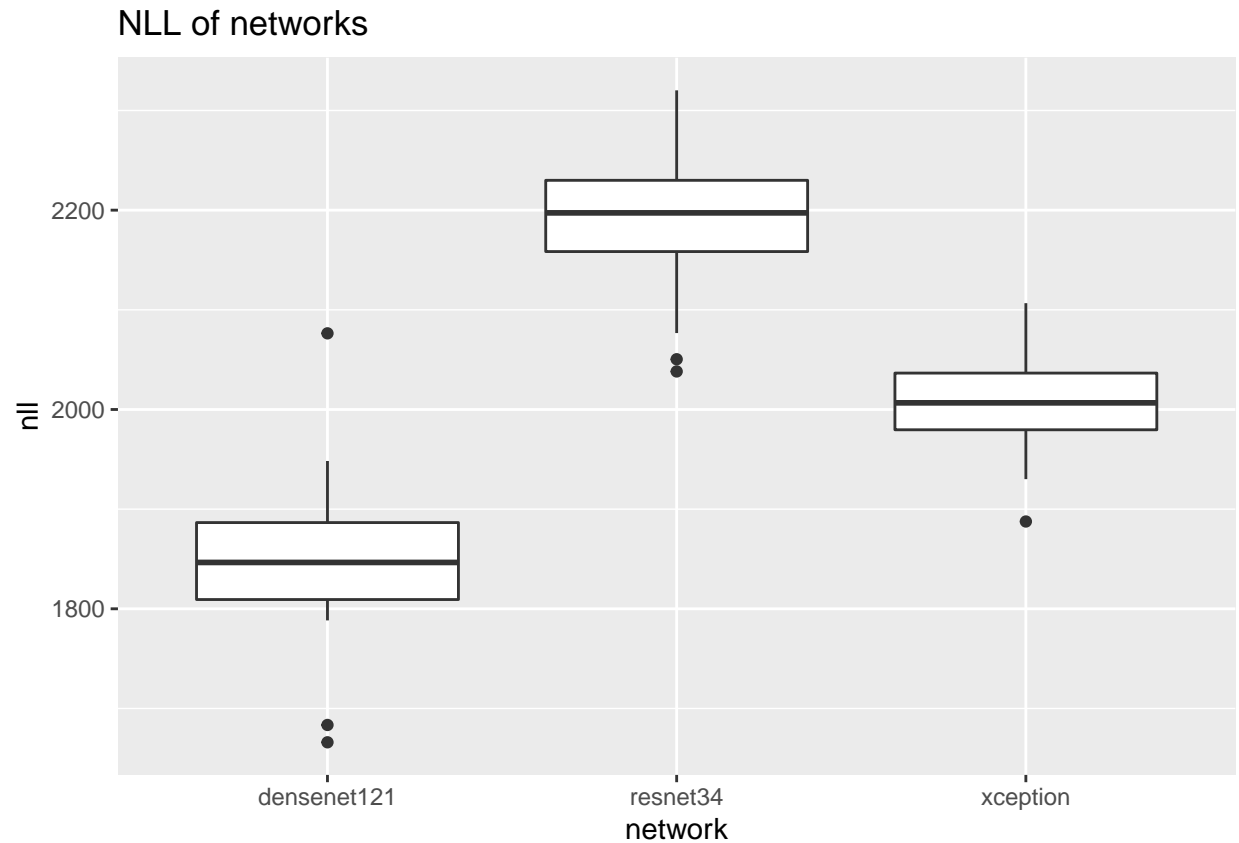
```



```

networks_nll <- ggplot(data=net_results) + geom_boxplot(mapping=aes(x=network, y=nll)) + ggtitle("NLL of predicted probabilities")
networks_nll

```

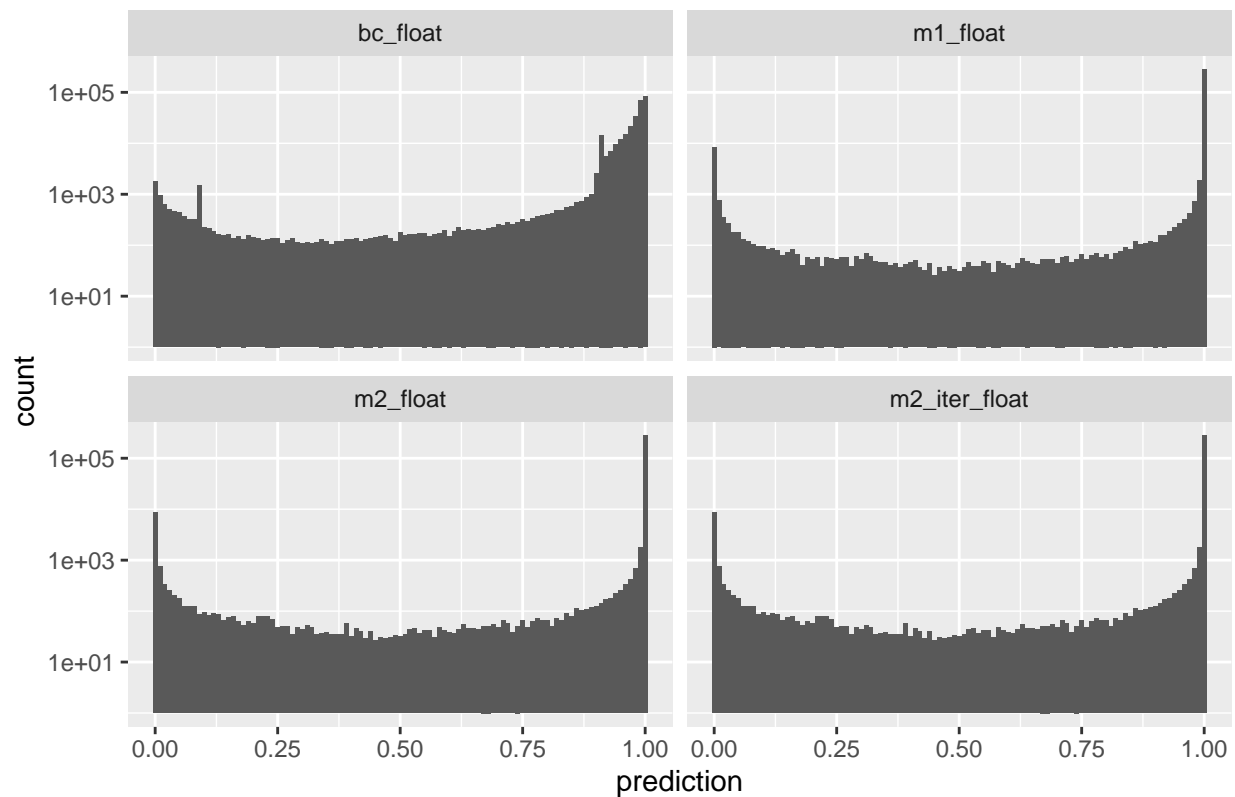


Networks nll seems to have good correspondence with lowest prediction probability column in the previous histograms.

```
val_ens_cor_probs <- gather(ens_outputs$val_training, 1 + nets_outputs$test_labels[1, ], 3, 4)
val_ens_cor_probs <- melt(val_ens_cor_probs)
val_ens_cor_probs <- val_ens_cor_probs[, c(-3, -4)]
names(val_ens_cor_probs) <- c("replication", "method", "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
```

Probabilities predicted for the correct class – ens trained on val



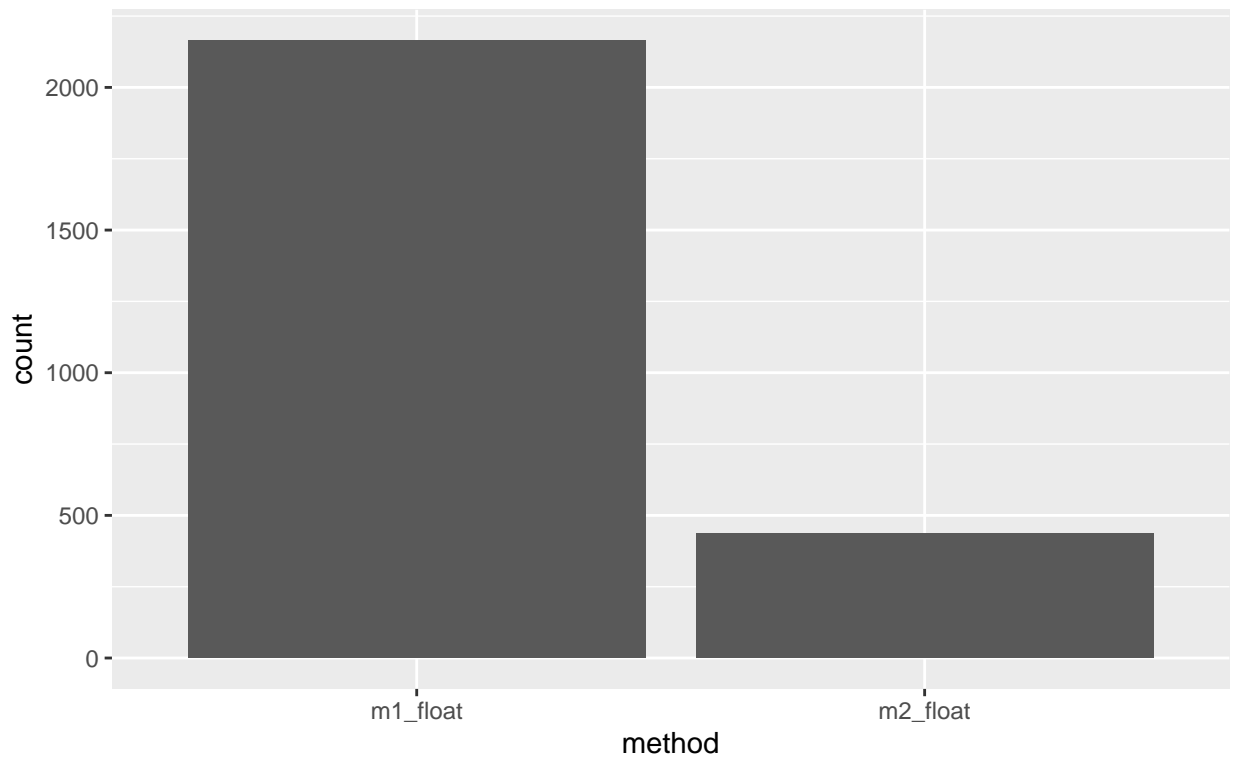
Coupling method bc produces far fewer probabilities falling into the lowest bin for the correct class than m1 and m2.

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

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

```
val_ens_zero_counts
```

Counts of zero or lower probabilities predicted for the correct class by coupling
Validation training

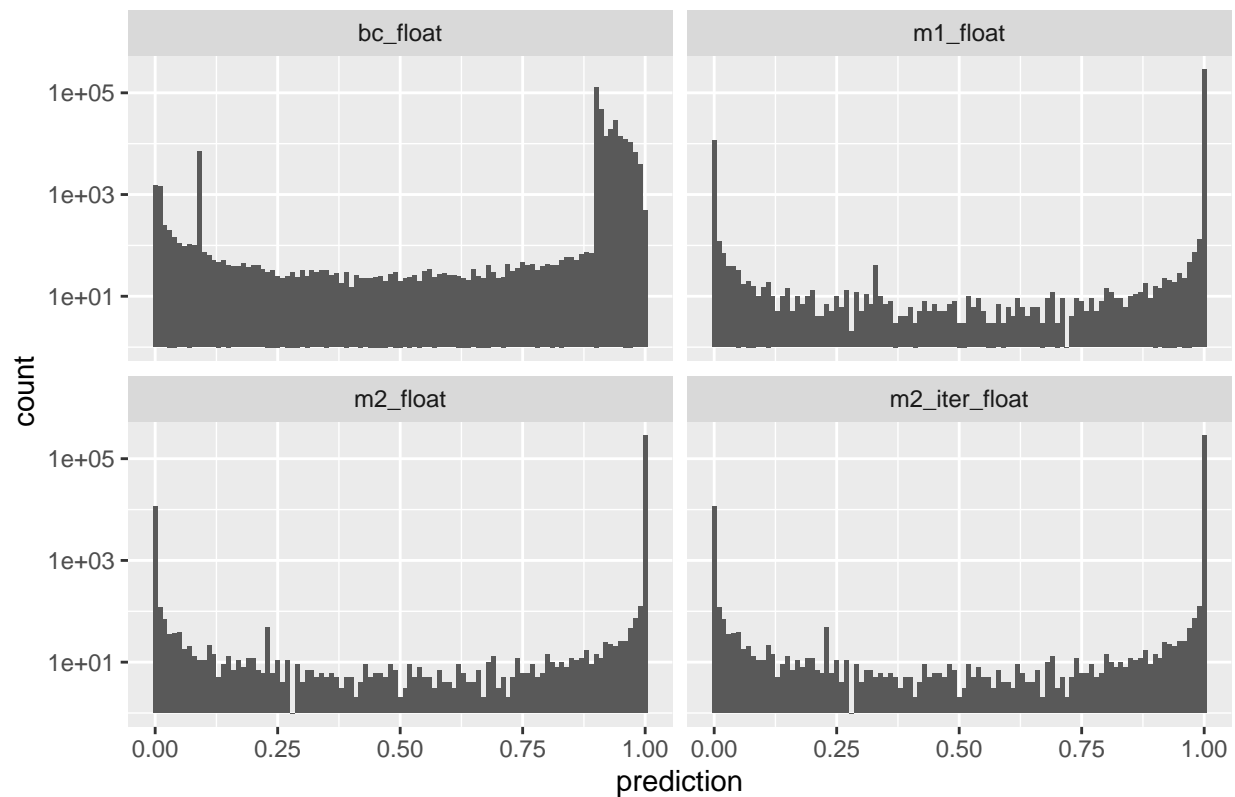


m2_iter and bc didn't produce any zero probability outputs in this replication.

```
train_ens_cor_probs <- gather(ens_outputs$train_training, 1 + nets_outputs$test_labels[1, ], 3, 4)
train_ens_cor_probs <- melt(train_ens_cor_probs)
train_ens_cor_probs <- train_ens_cor_probs[, c(-3, -4)]
names(train_ens_cor_probs) <- c("replication", "method", "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
```

Probabilities predicted for the correct class – ens trained on train



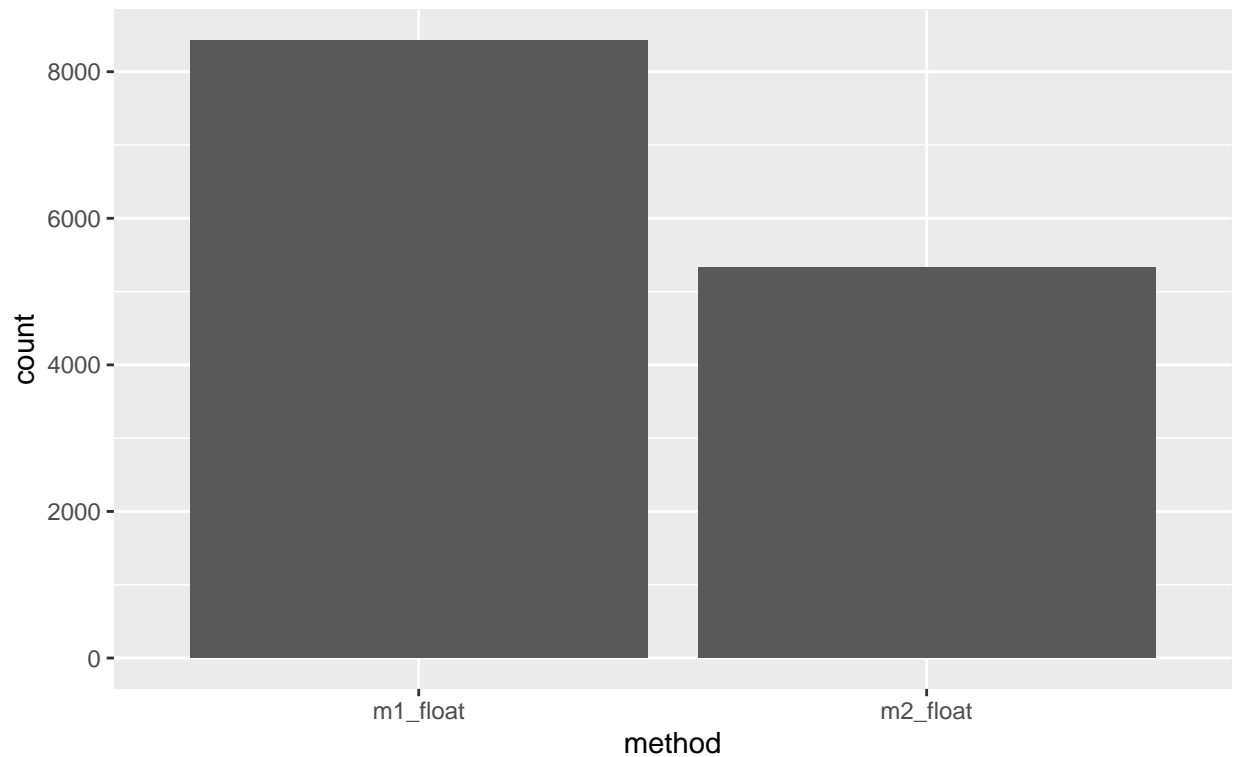
Also in this case, coupling method bc produces far fewer probabilities falling into the lowest bin for the correct class than m1 and m2.

```
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
```

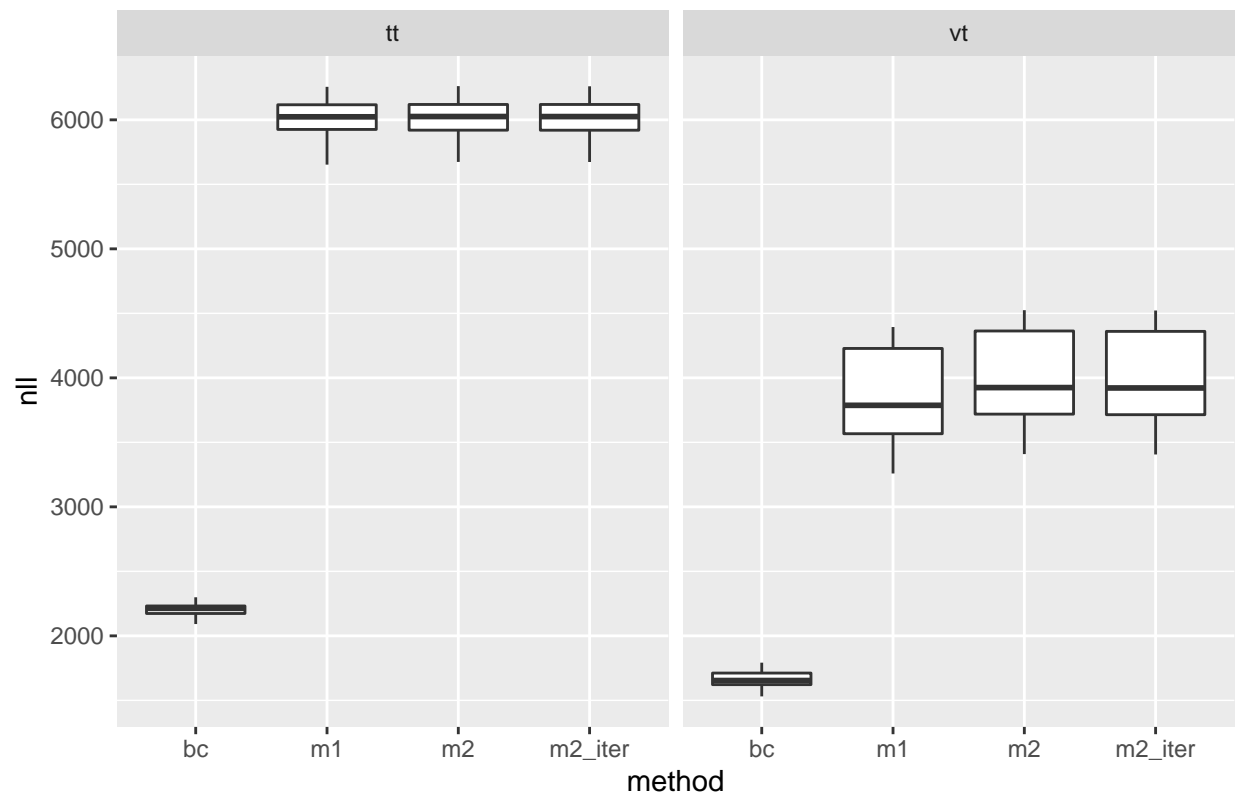
Counts of zero or lower probabilities predicted for the correct class by coupling method
Train training



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

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


Comparison of nll for coupling methods for different LDA train methodology

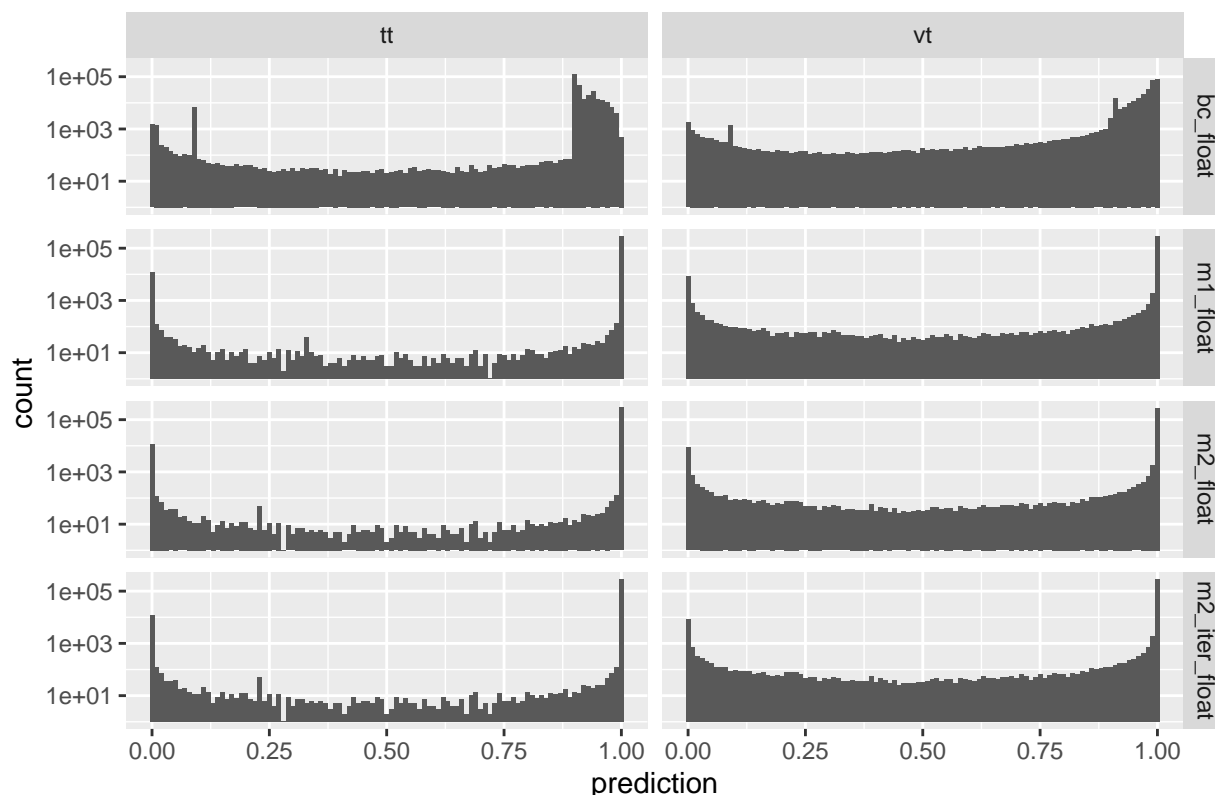


When LDA is trained on training data, bc method has clearly superior nll, however in the case of training LDA on validation data, other coupling methods perform similarly, or better.

```
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
```

Probabilities predicted for the correct class



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.

```
ens_Rs <- load_R_matrices(base_dir, repls)
```

```
df_val_Rs <- melt(ens_Rs$val_training)
names(df_val_Rs) <- c("replication", "precision", "sample", "class1", "class2", "prob")
class <- nets_outputs$test_labels[1, df_val_Rs$sample] + 1
df_val_Rs$class <- as.factor(class)
df_val_Rs[,c("class1", "class2")] <- lapply(df_val_Rs[,c("class1", "class2")], factor)
val_class_mean_Rs <- df_val_Rs %>% group_by(precision, class1, class2, class) %>% summarise(prob=mean(prob))
```

'summarise()' has grouped output by 'precision', 'class1', 'class2'. You can override using the '.groups' argument.

```
df_train_Rs <- melt(ens_Rs$train_training)
names(df_train_Rs) <- c("replication", "precision", "sample", "class1", "class2", "prob")
class <- nets_outputs$test_labels[1, df_train_Rs$sample] + 1
df_train_Rs$class <- as.factor(class)
df_train_Rs[,c("class1", "class2")] <- lapply(df_train_Rs[,c("class1", "class2")], factor)
train_class_mean_Rs <- df_train_Rs %>% group_by(precision, class1, class2, class) %>% summarise(prob=mean(prob))
```

'summarise()' has grouped output by 'precision', 'class1', 'class2'. You can override using the '.groups' argument.

```

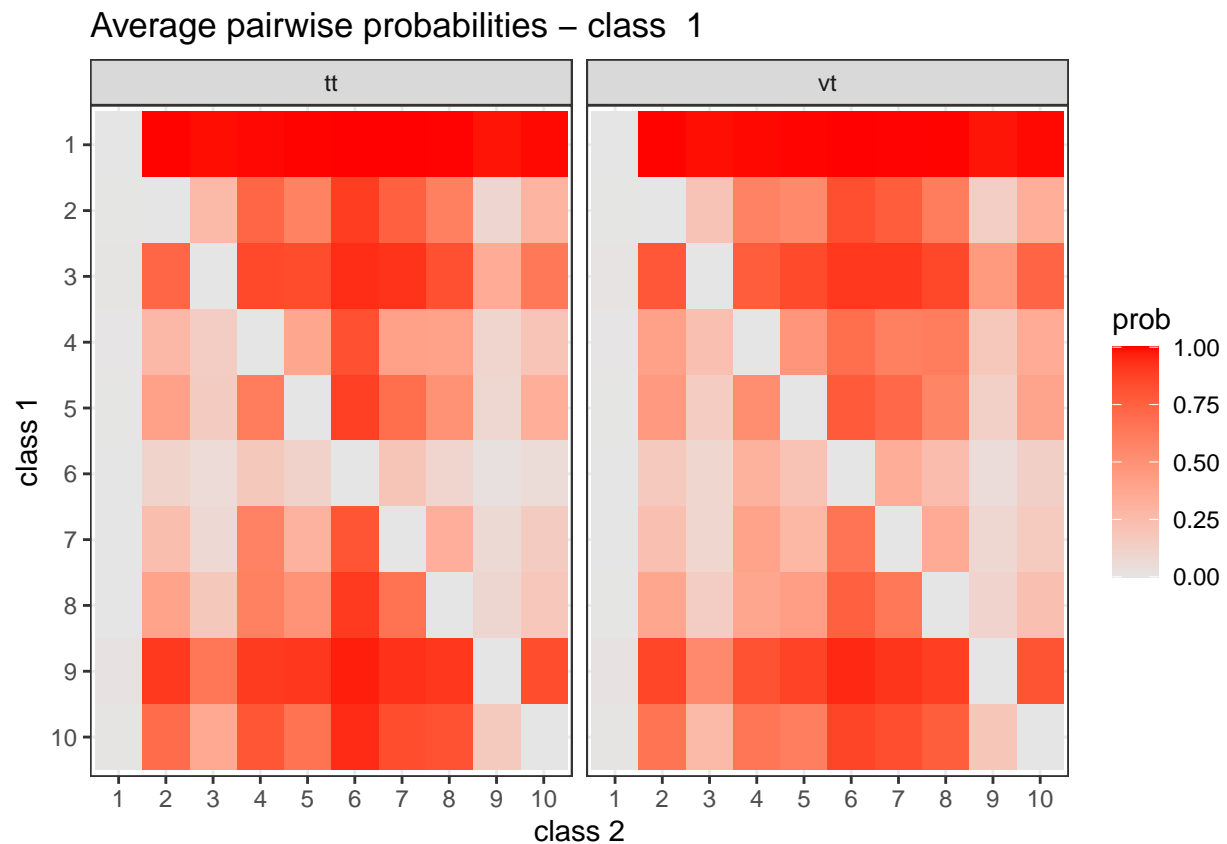
val_class_mean_Rs$train_type <- "vt"
train_class_mean_Rs$train_type <- "tt"
class_mean_Rs <- rbind(val_class_mean_Rs, train_class_mean_Rs)

df_aggr_Rs_diff <- class_mean_Rs %>% pivot_wider(names_from = train_type, values_from = prob) %>% mutate(

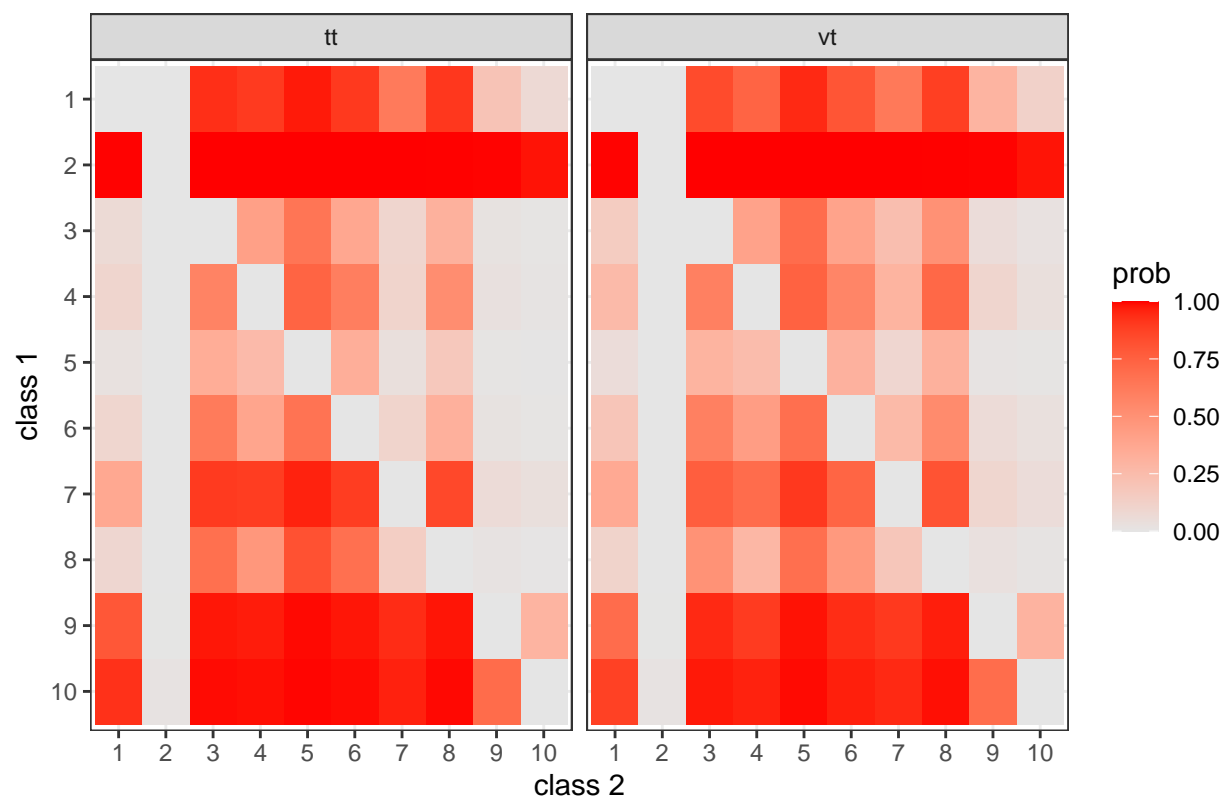
for (cls in 1:classes)
{
  cur_class_Rs <- class_mean_Rs %>% filter(class == cls)
  plot_cls <- ggplot(cur_class_Rs, aes(x = class2, y = class1)) +
    geom_raster(aes(fill=prob)) +
    facet_wrap(~train_type) +
    scale_fill_gradient(low="grey90", high="red", limits=c(0, 1)) +
    scale_y_discrete(limits=rev) +
    labs(x="class 2", y="class 1", title=paste("Average pairwise probabilities - class ", cls)) +
    theme_bw()

  print(plot_cls)
}

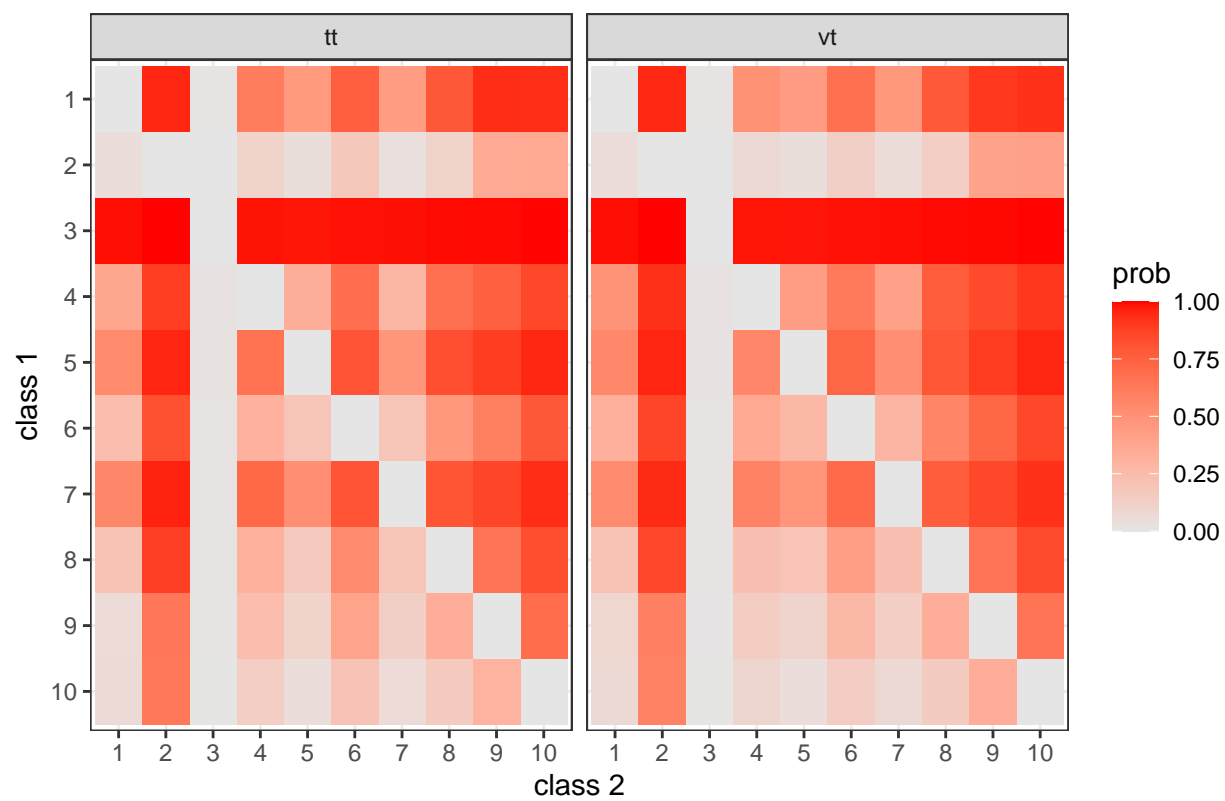
```



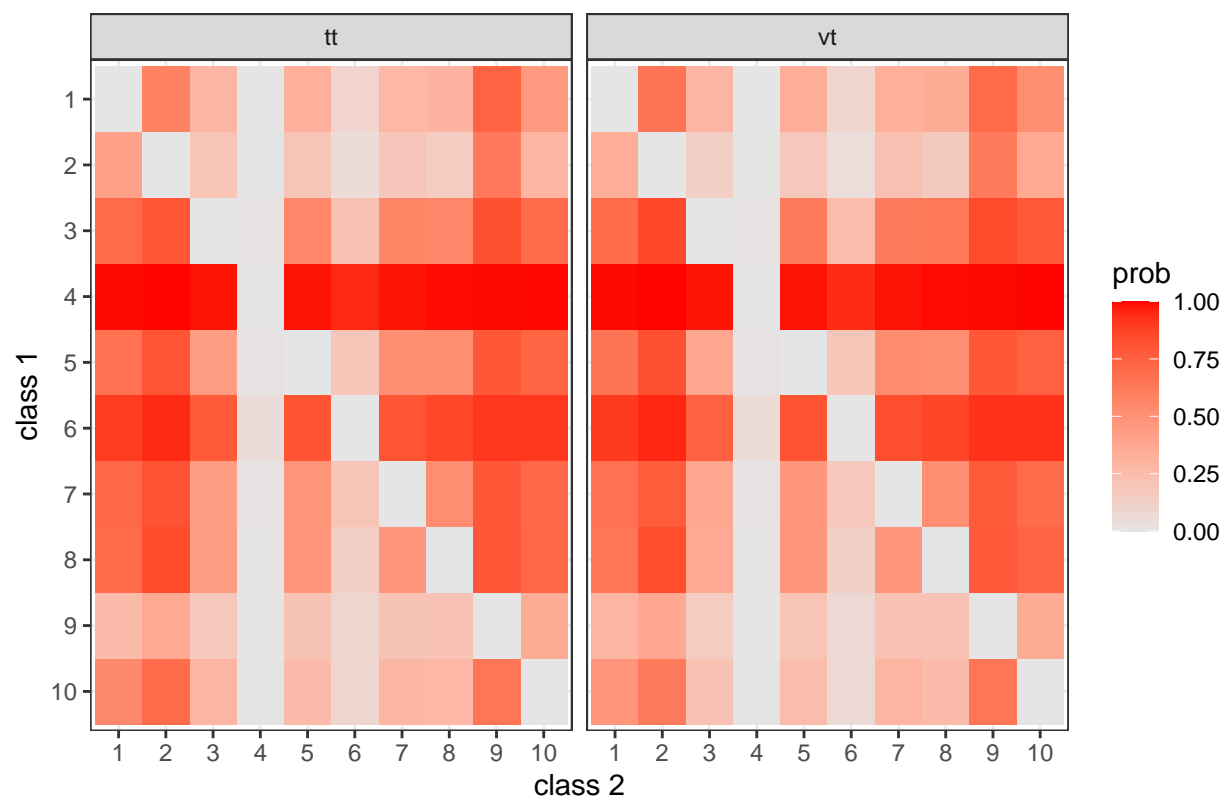
Average pairwise probabilities – class 2



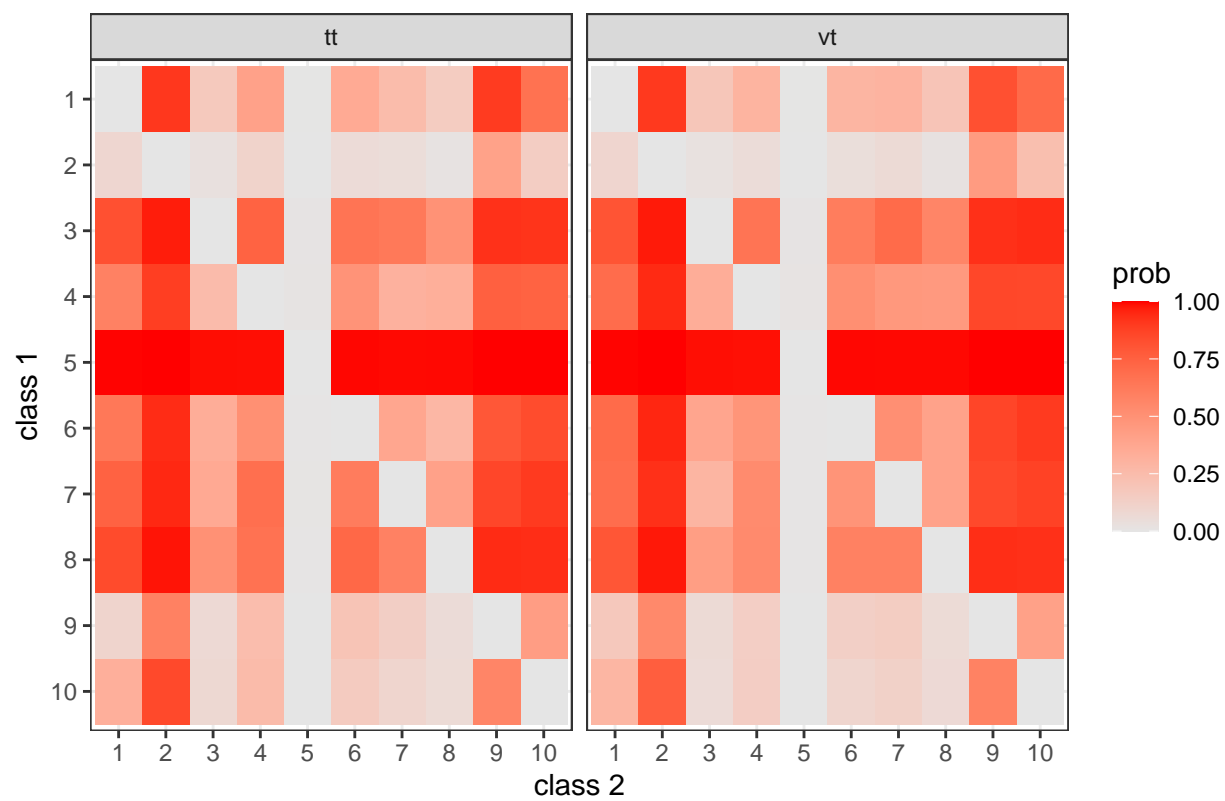
Average pairwise probabilities – class 3



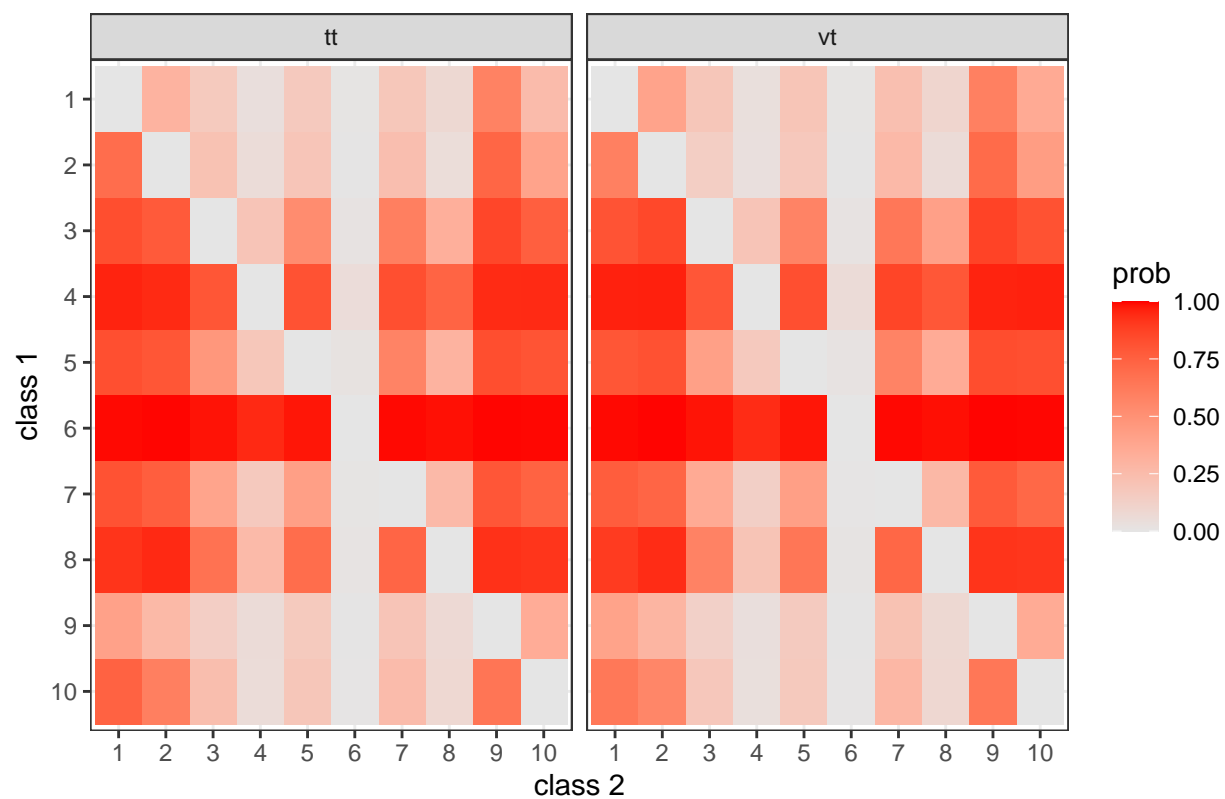
Average pairwise probabilities – class 4



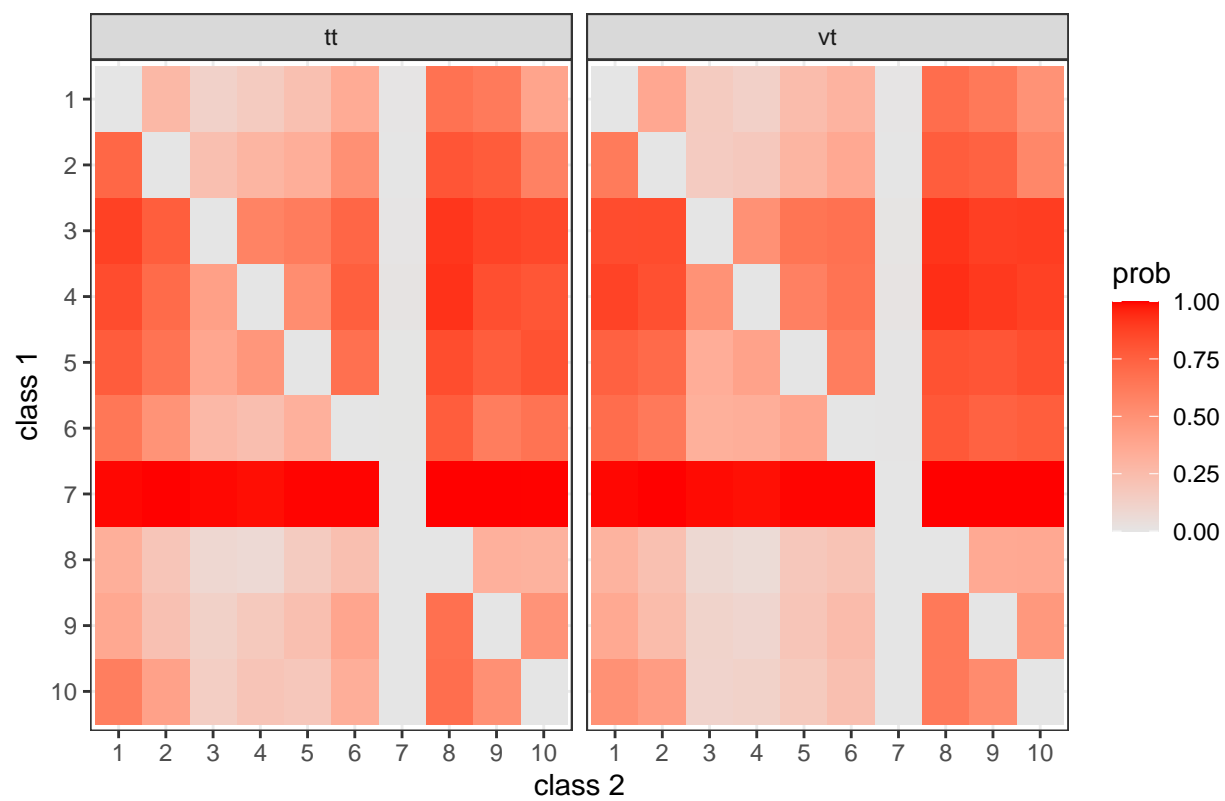
Average pairwise probabilities – class 5



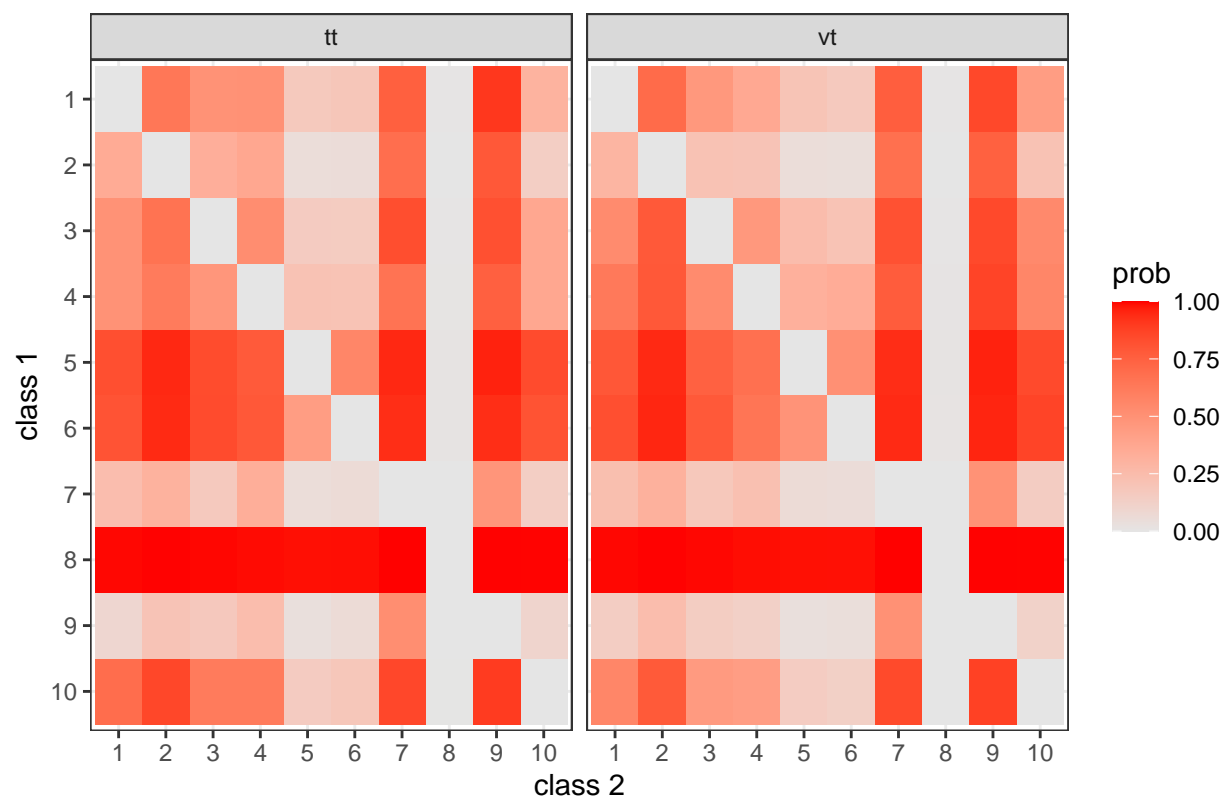
Average pairwise probabilities – class 6



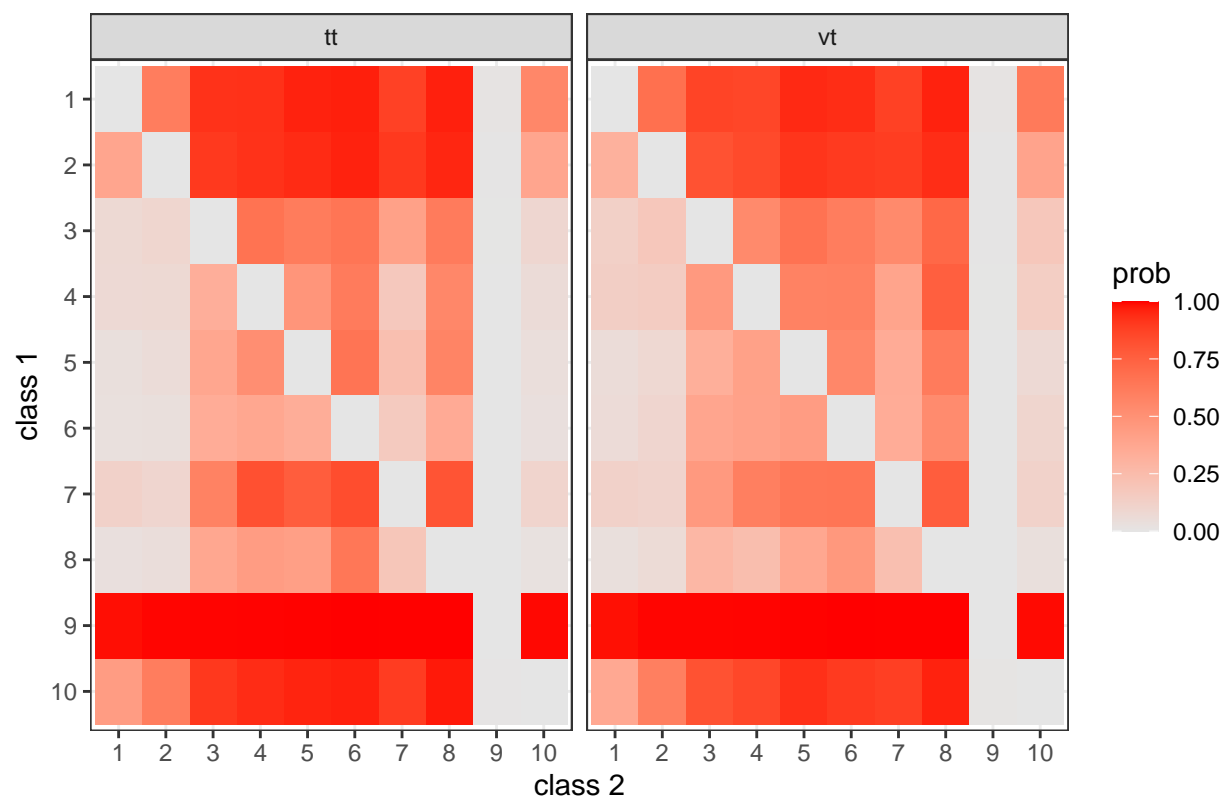
Average pairwise probabilities – class 7



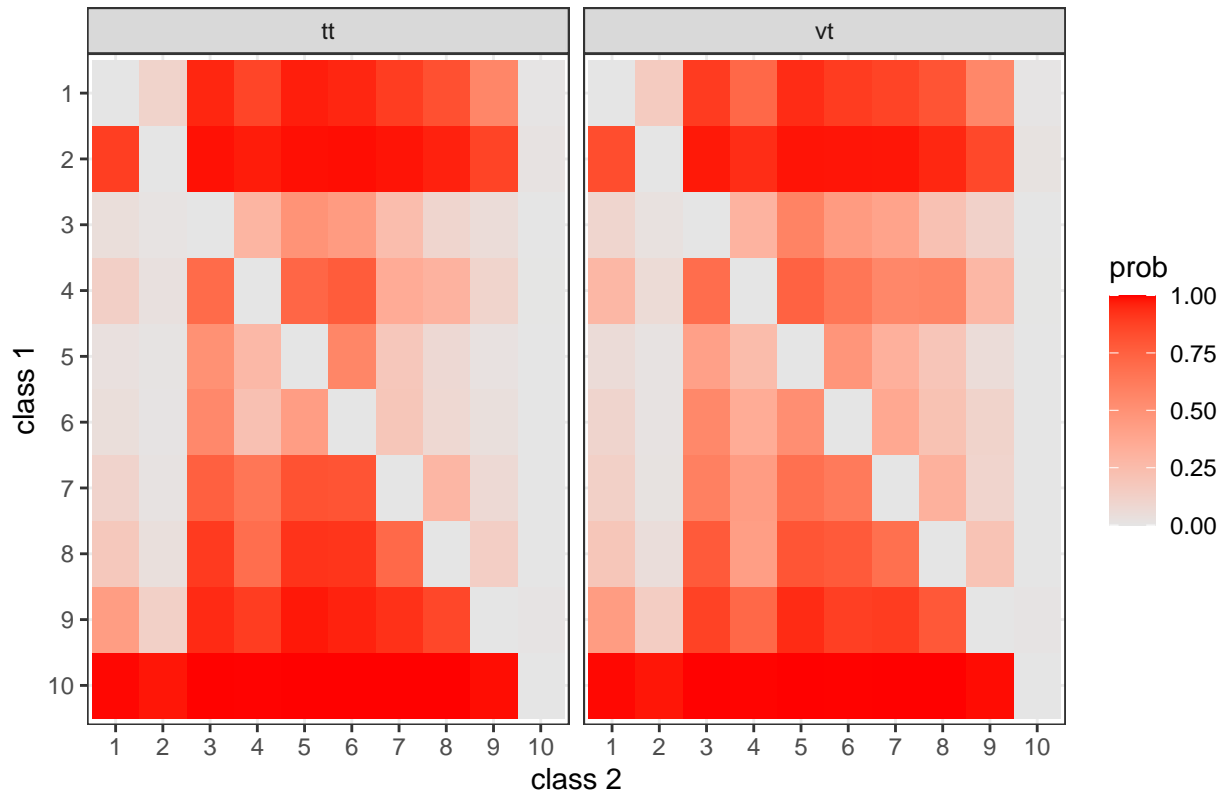
Average pairwise probabilities – class 8



Average pairwise probabilities – class 9



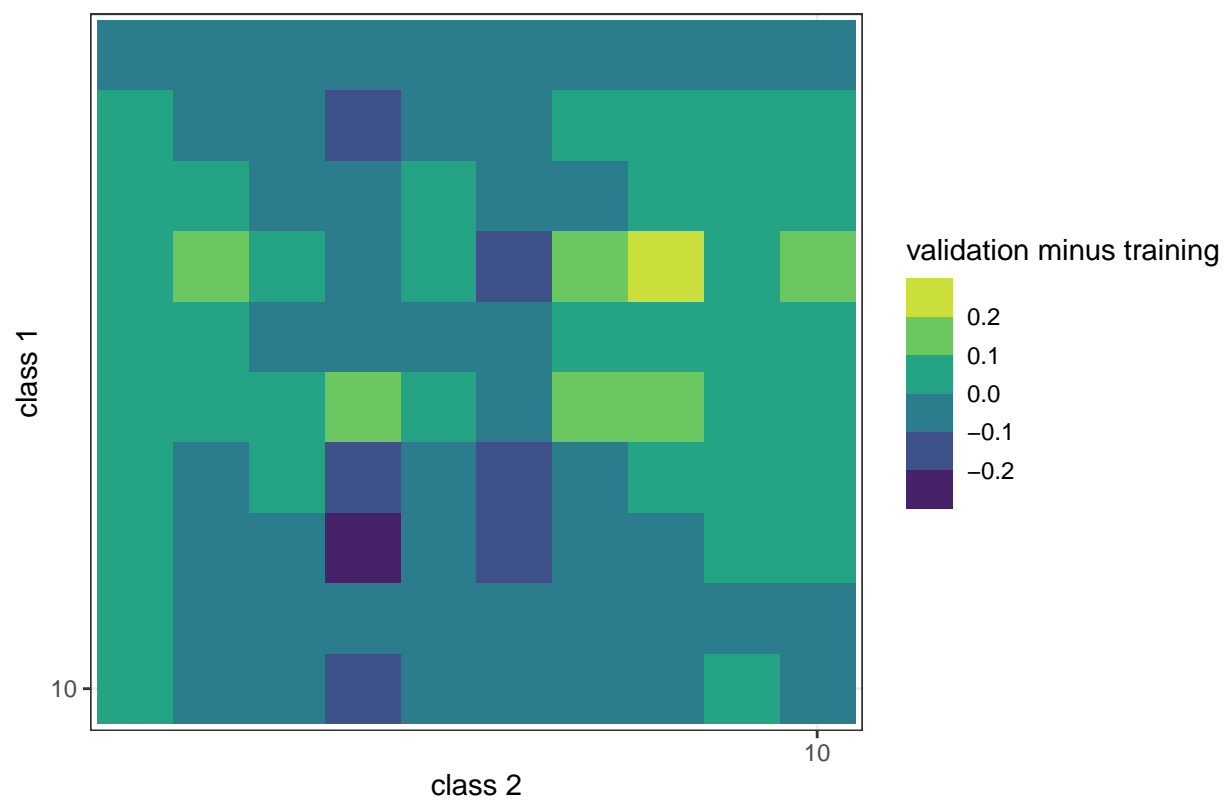
Average pairwise probabilities – class 10



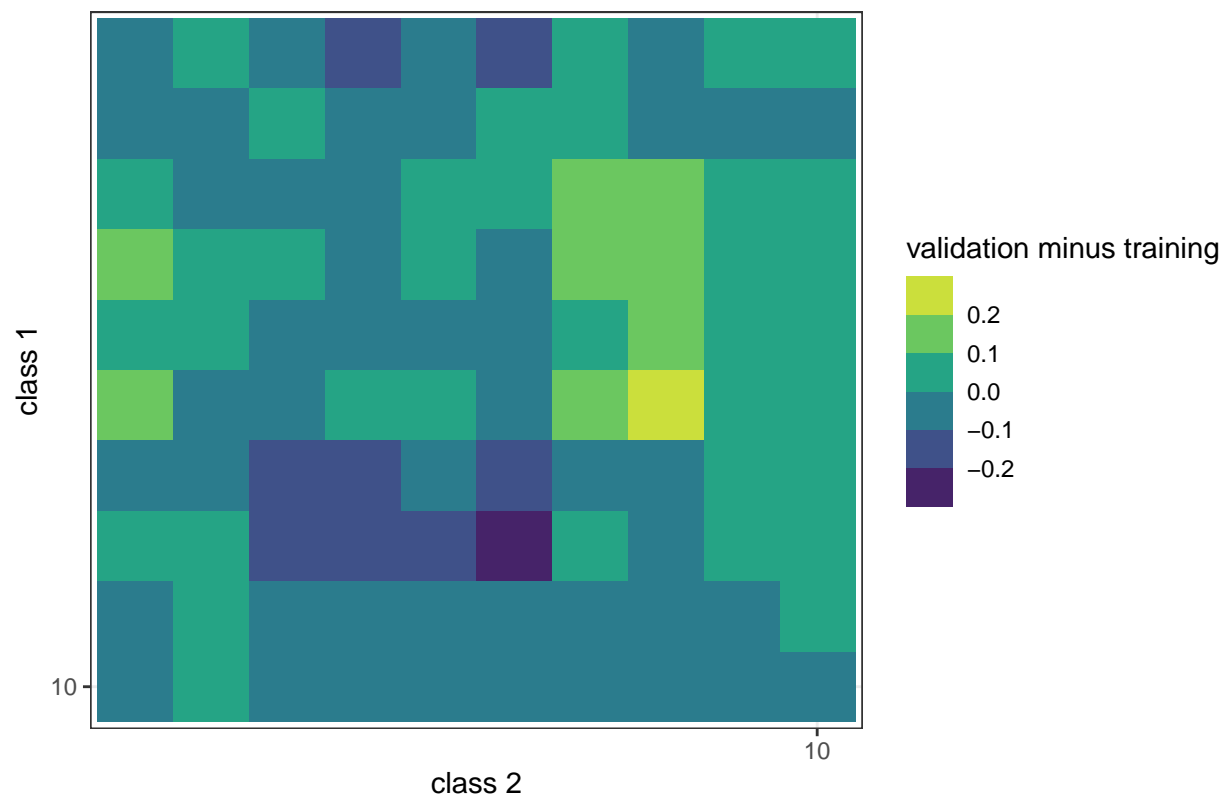
```
for (cls in 1:classes)
{
  cur_class_Rs <- df_aggr_Rs_diff %>% filter(class == cls)
  plot_cls <- ggplot(cur_class_Rs, aes(x = class2, y = class1)) +
    geom_raster(aes(fill=val_min_train)) +
    scale_fill_binned(type="viridis", limits=c(-0.3, 0.3), name="validation minus training") +
    scale_y_discrete(limits=rev, 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 - cl
    theme_bw()

  print(plot_cls)
}
```

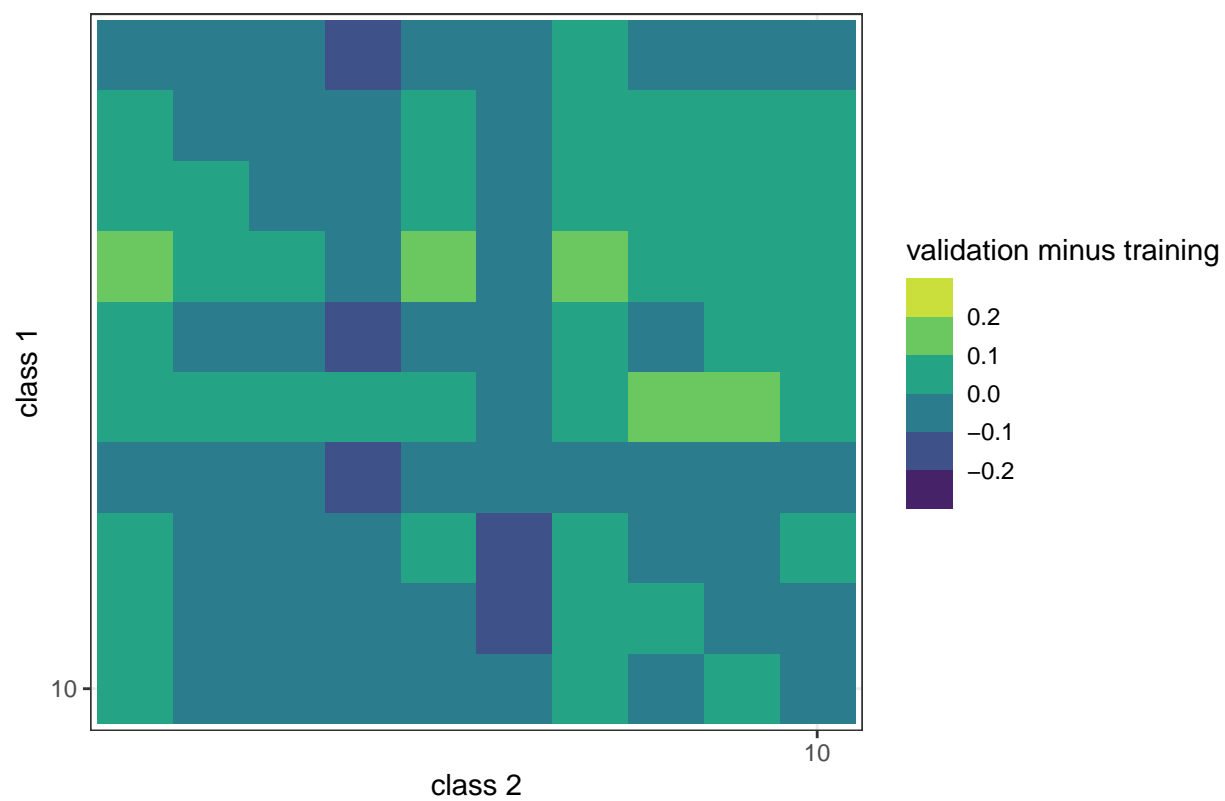
Differences between average pairwise probabilities – class 1



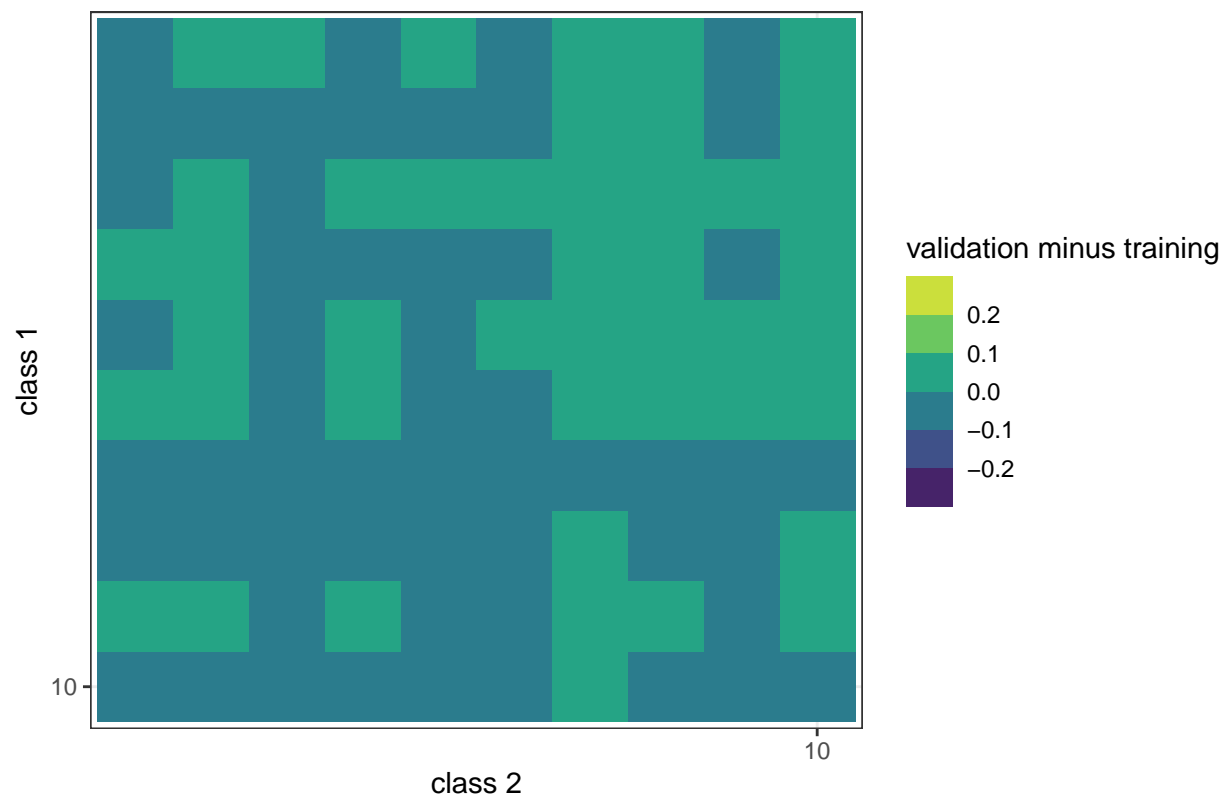
Differences between average pairwise probabilities – class 2



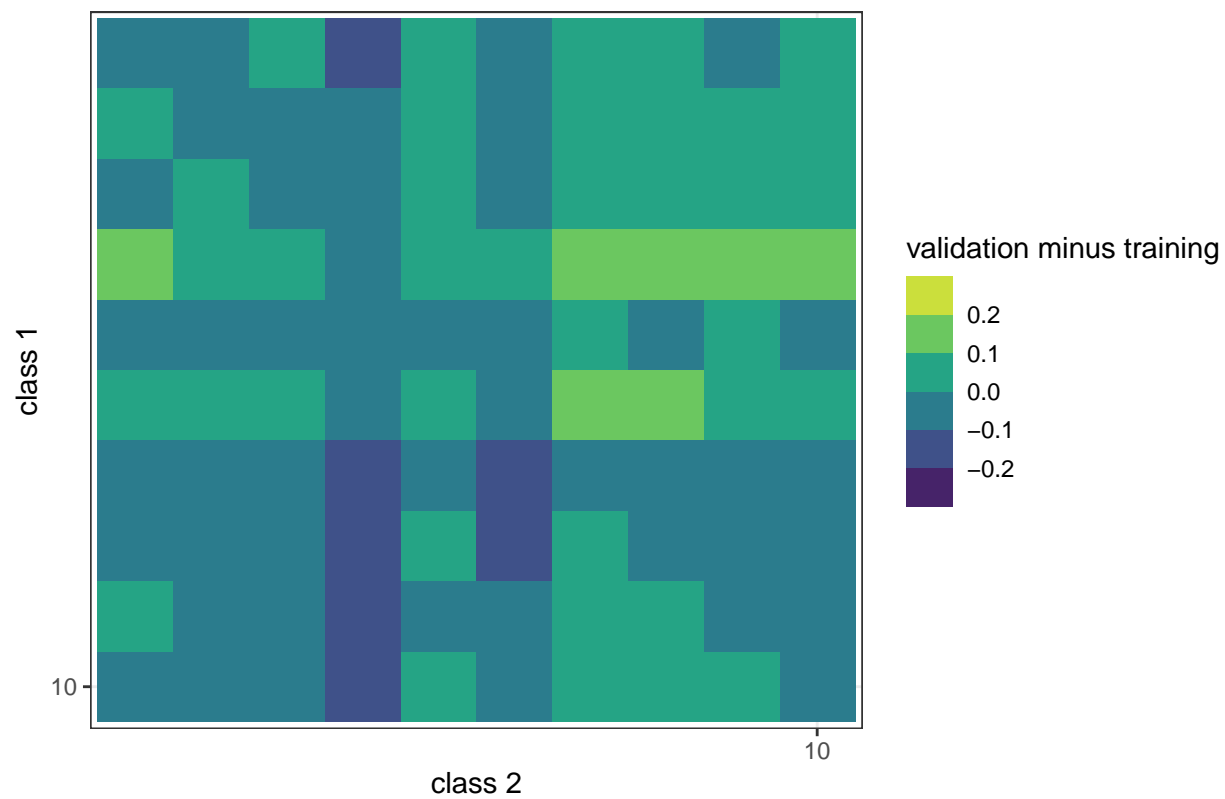
Differences between average pairwise probabilities – class 3



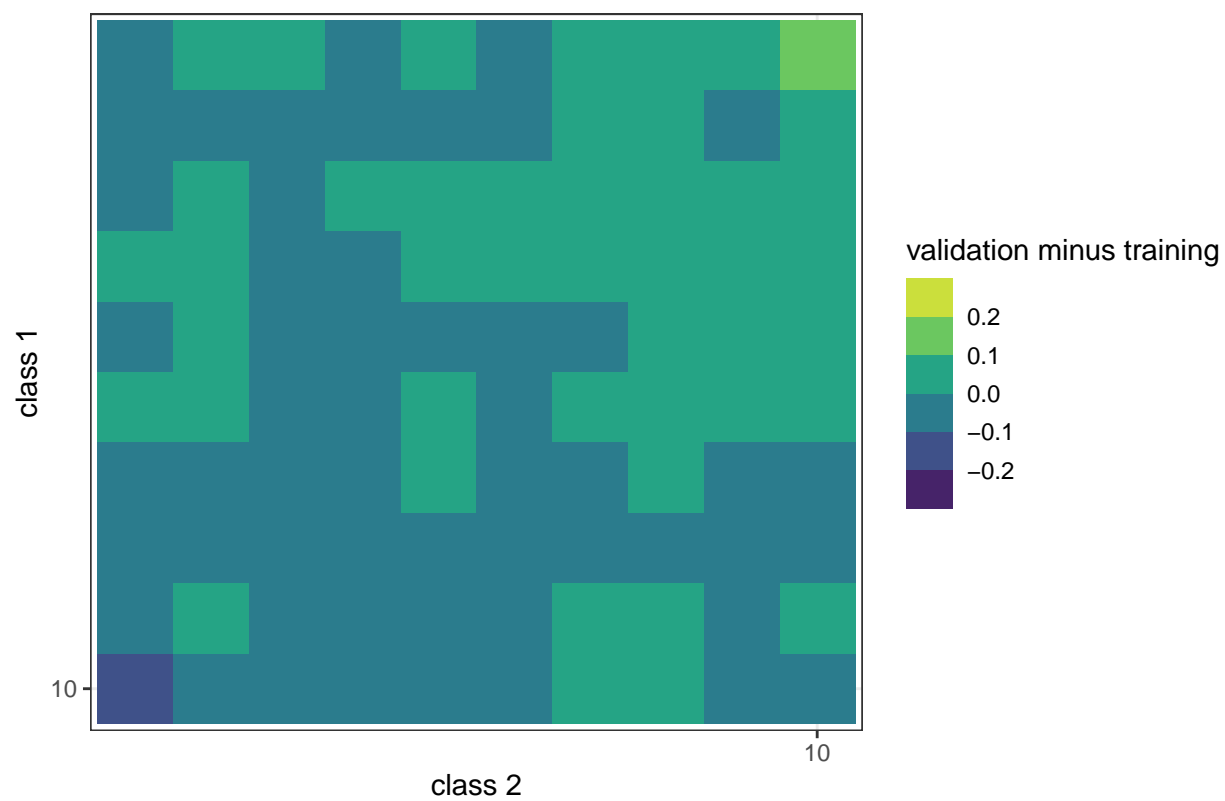
Differences between average pairwise probabilities – class 4



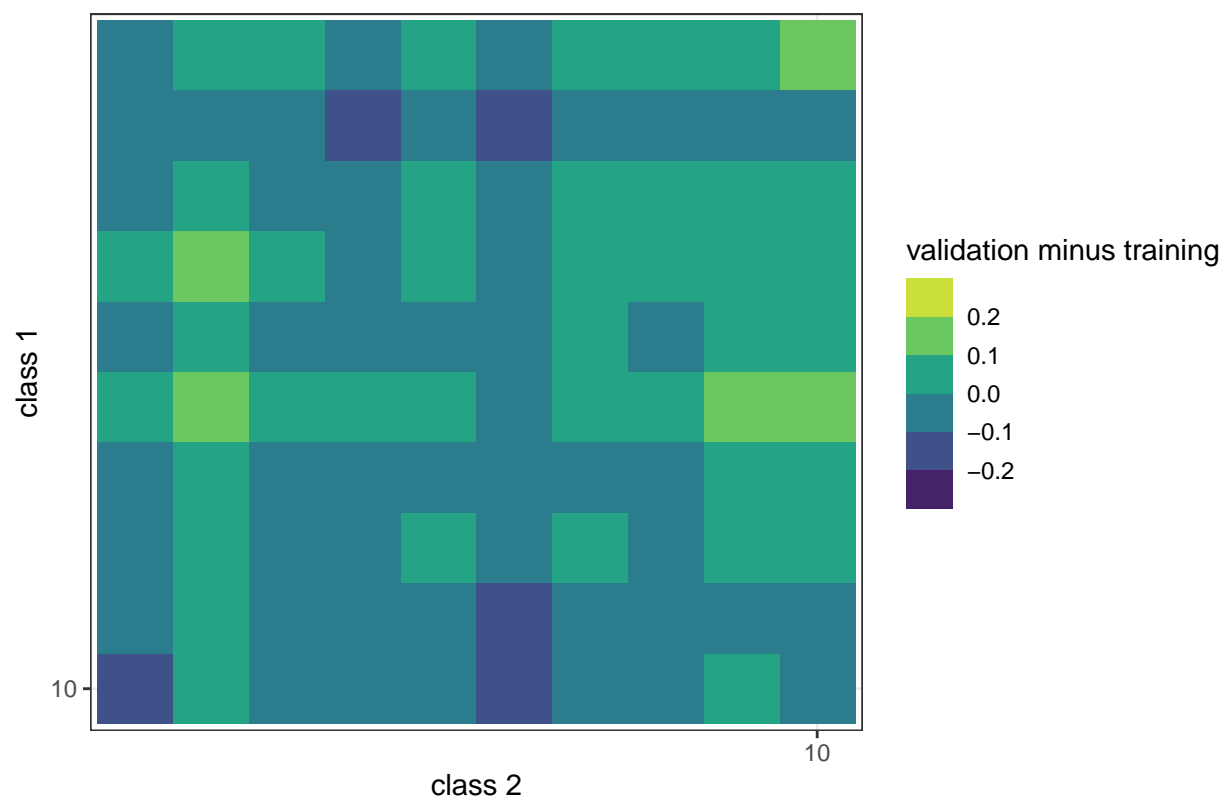
Differences between average pairwise probabilities – class 5



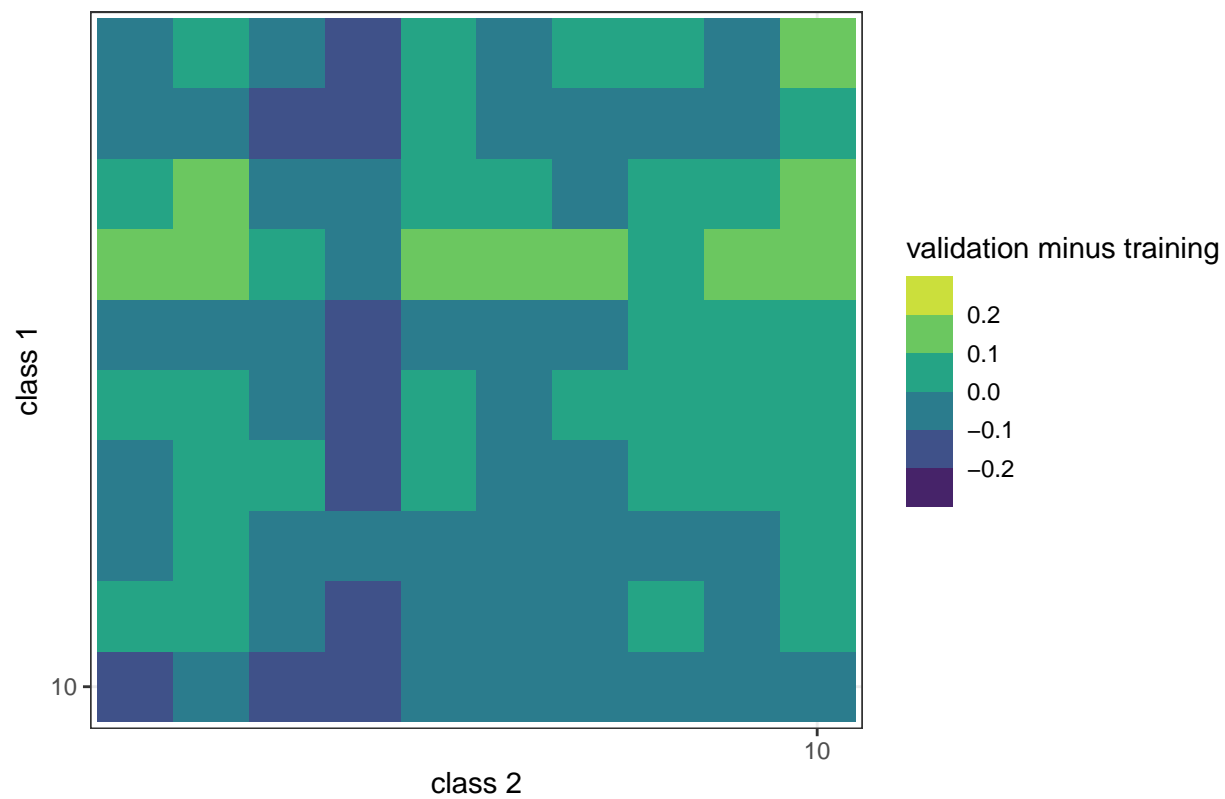
Differences between average pairwise probabilities – class 6



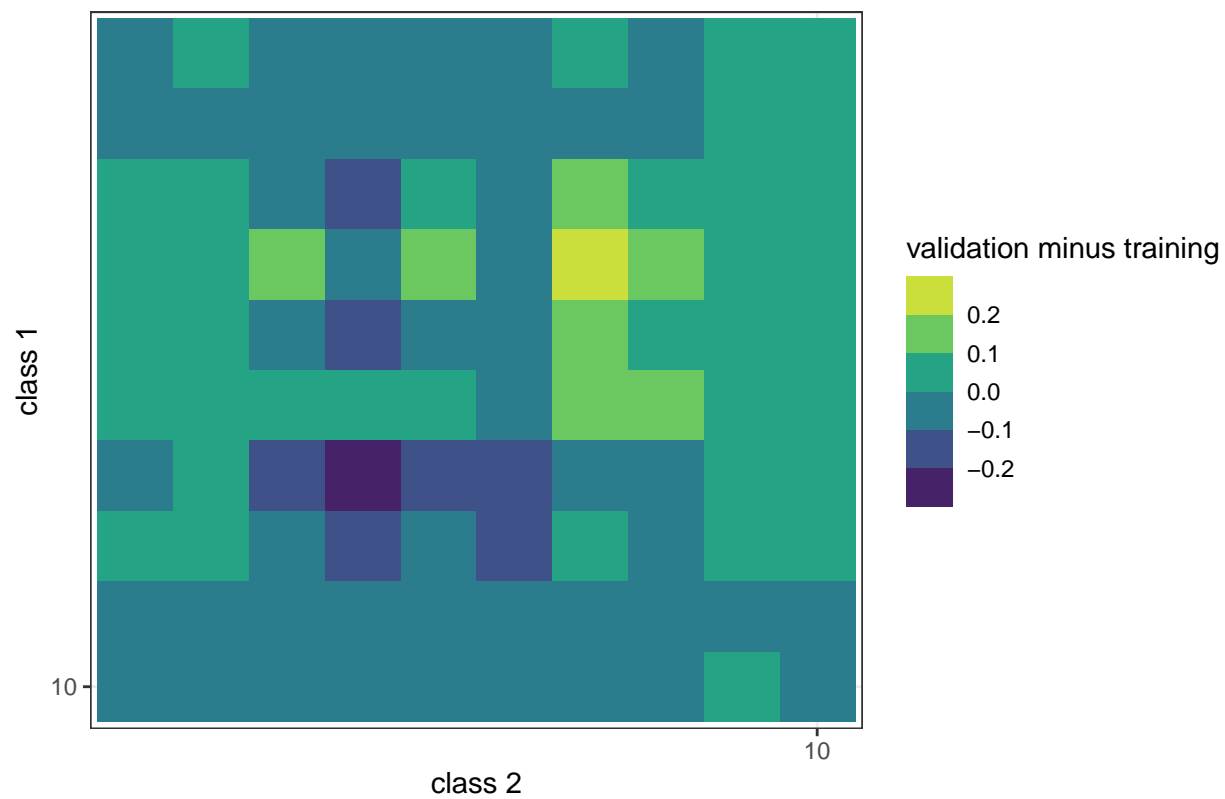
Differences between average pairwise probabilities – class 7



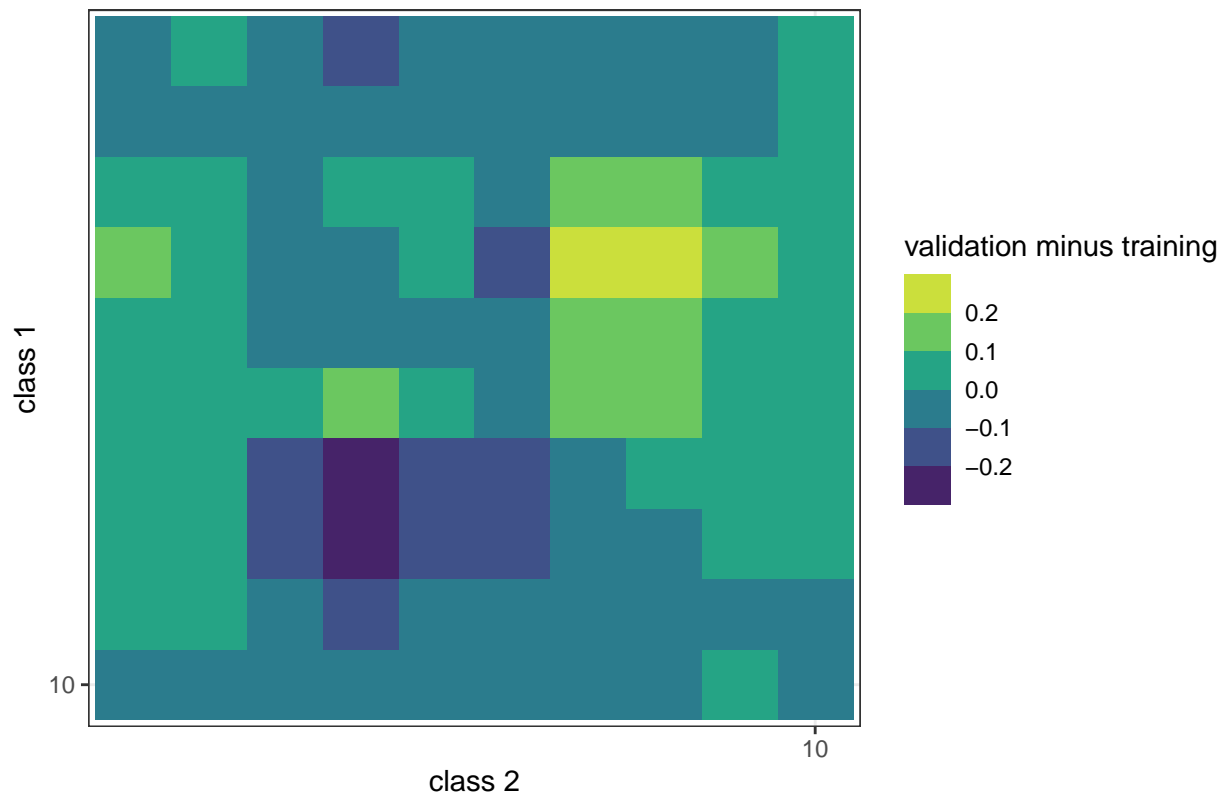
Differences between average pairwise probabilities – class 8



Differences between average pairwise probabilities – class 9



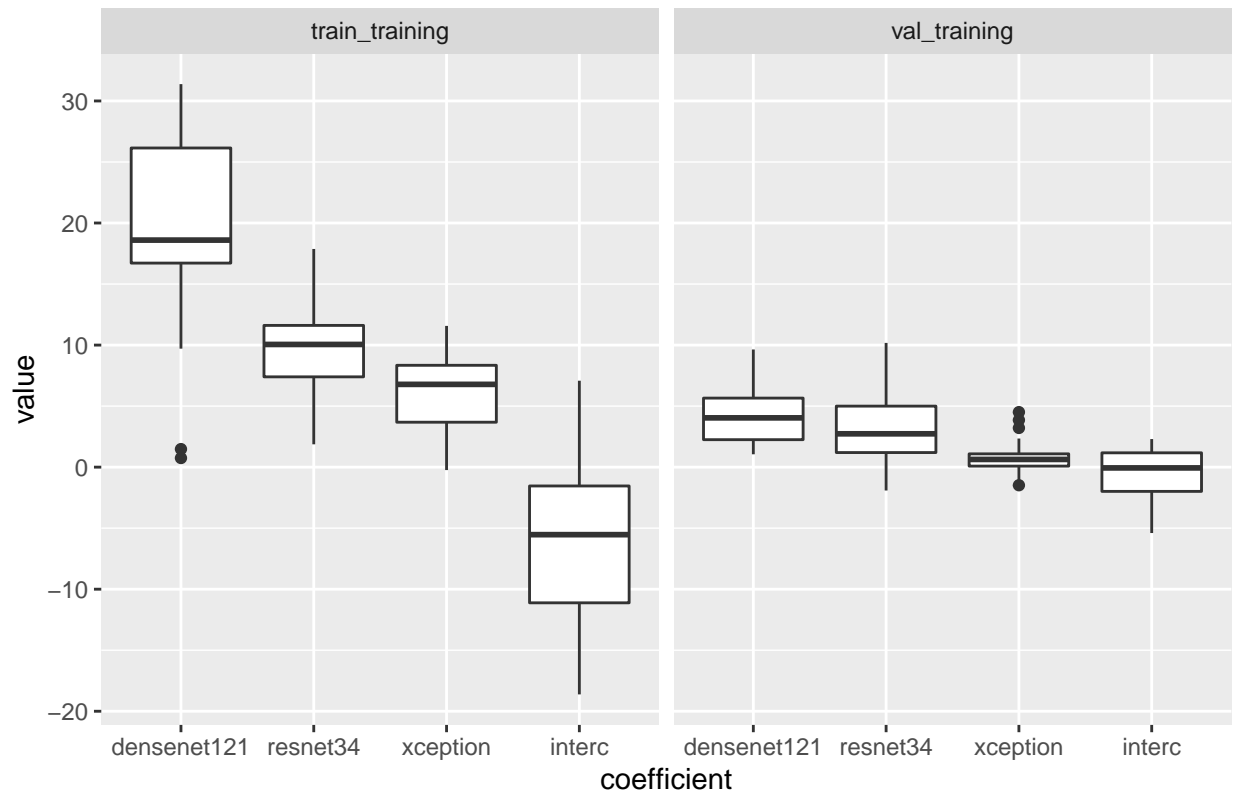
Differences between average pairwise probabilities – class 10



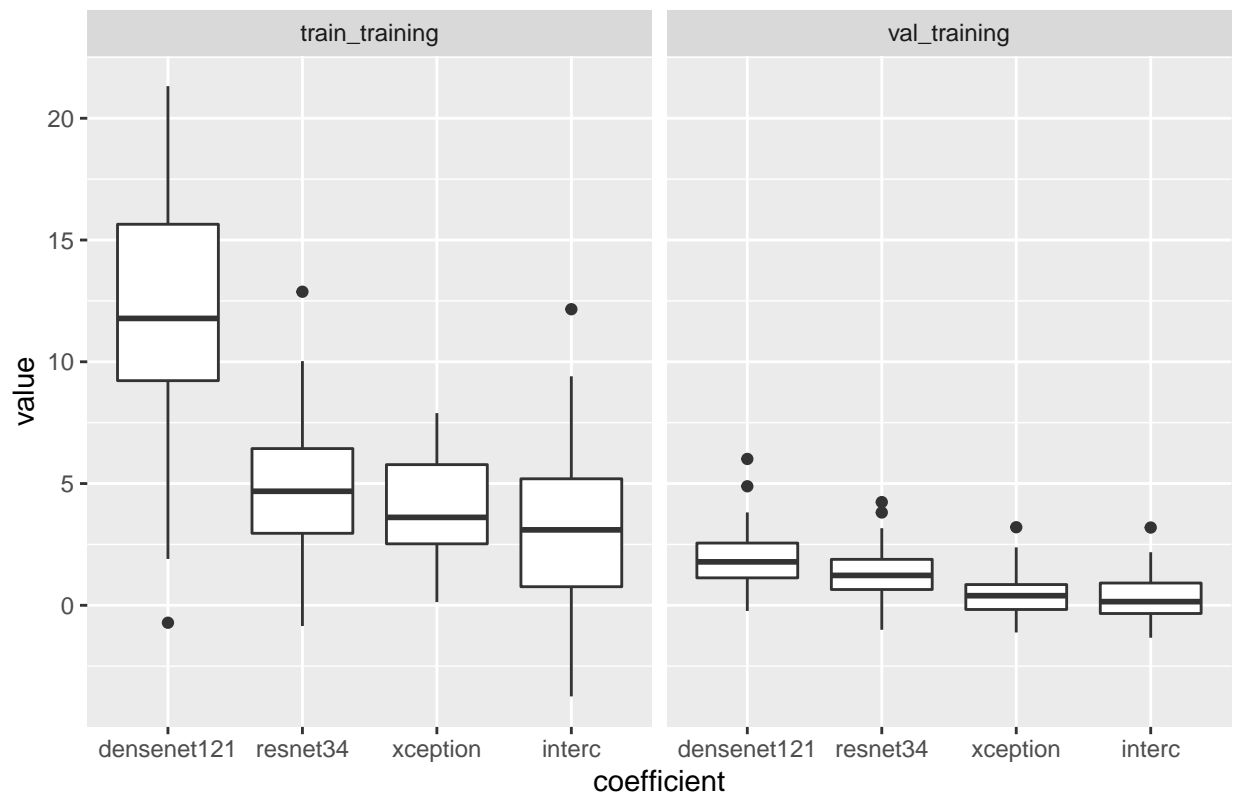
```
lda_coefs <- load_lda_coefs(base_dir, repls)
```

```
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=coefficient))
    facet_wrap(~train_type) + ggtitle(paste("Coefficients for class", cl1, "vs", cl2))
    print(cur_plt)
  }
}
```

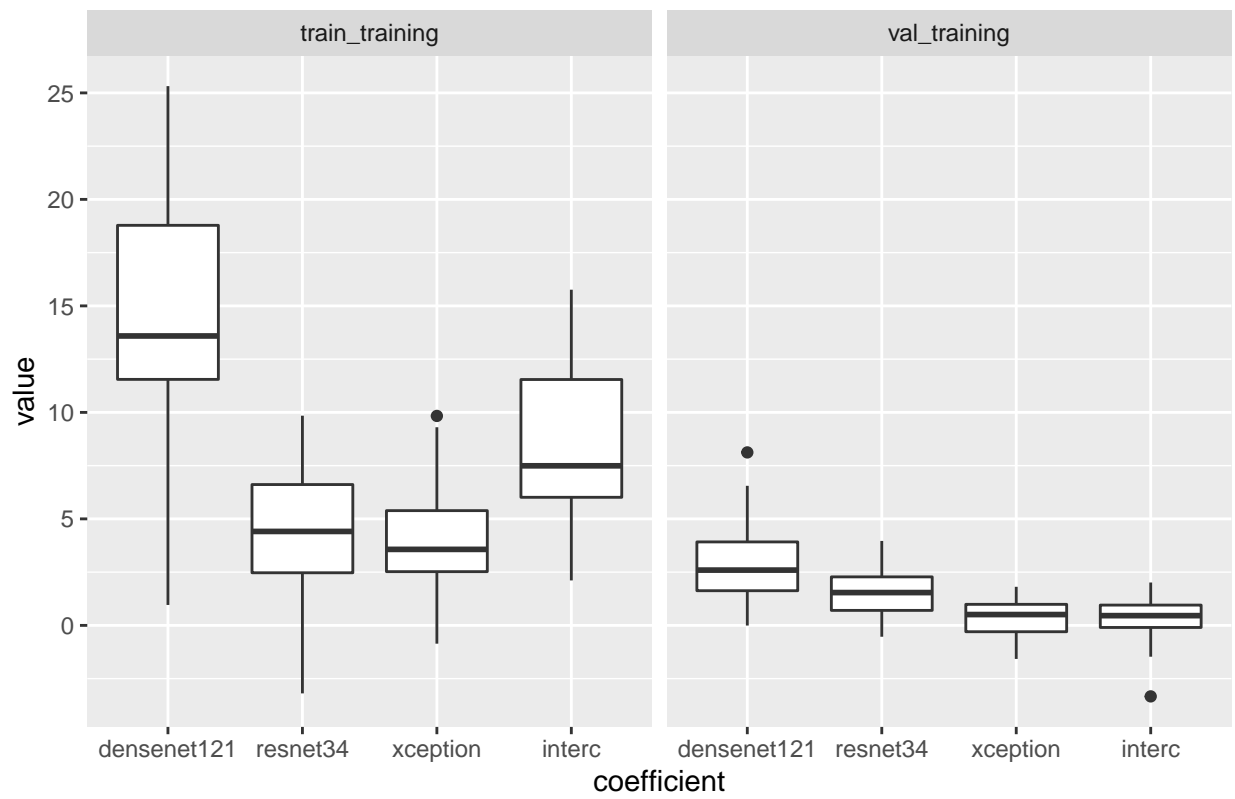
Coefficients for class 1 vs 2



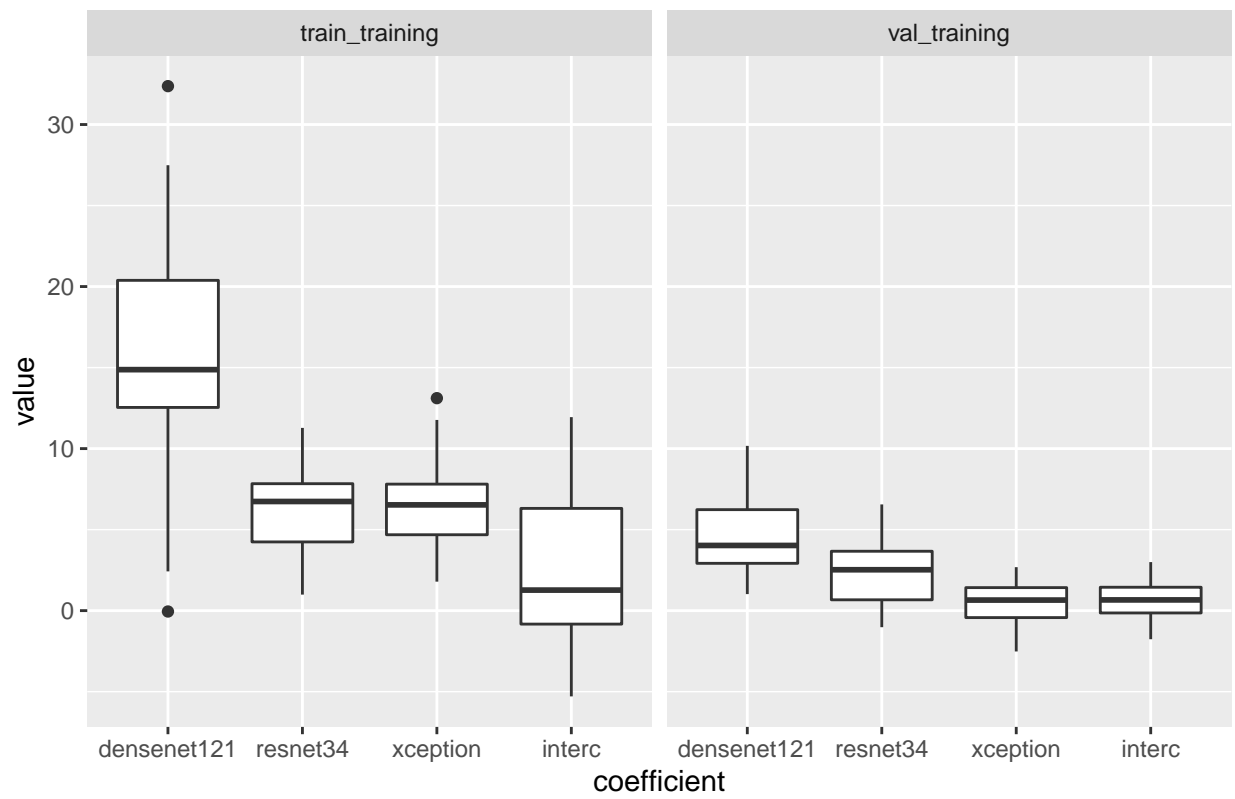
Coefficients for class 1 vs 3



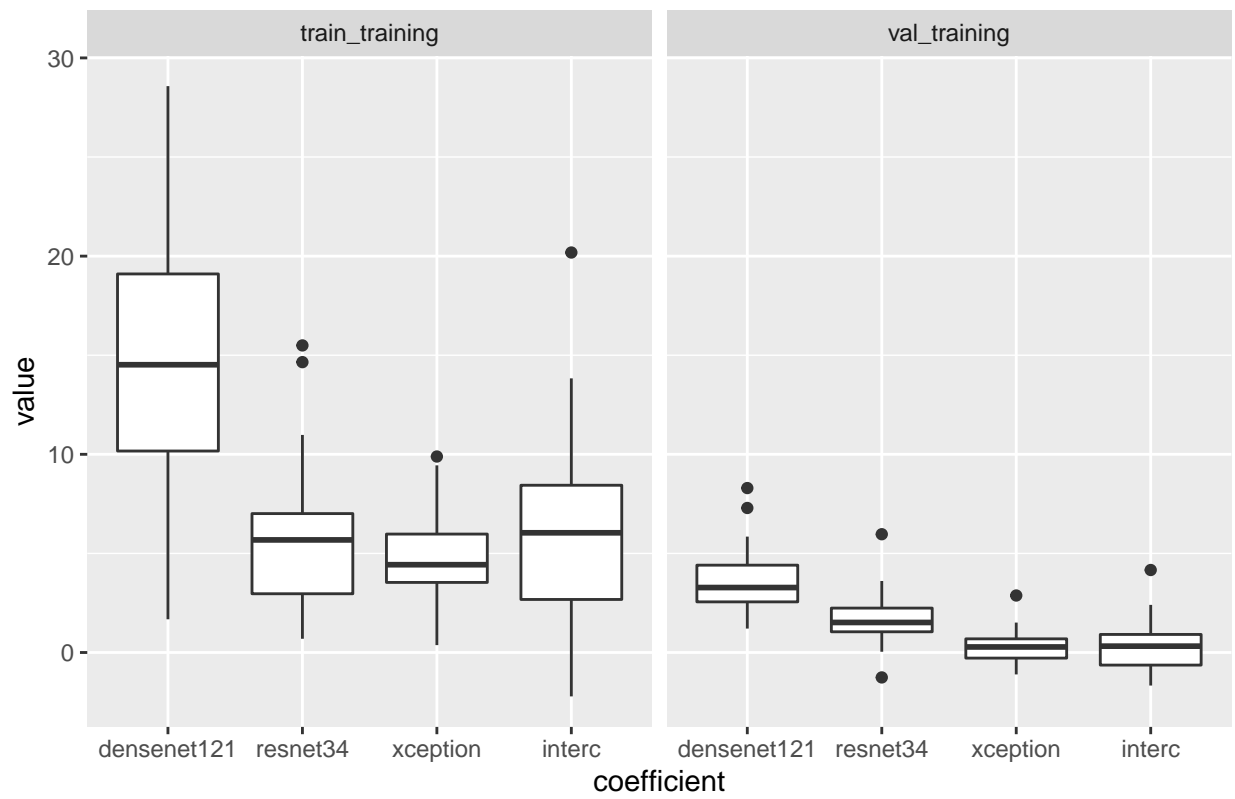
Coefficients for class 1 vs 4



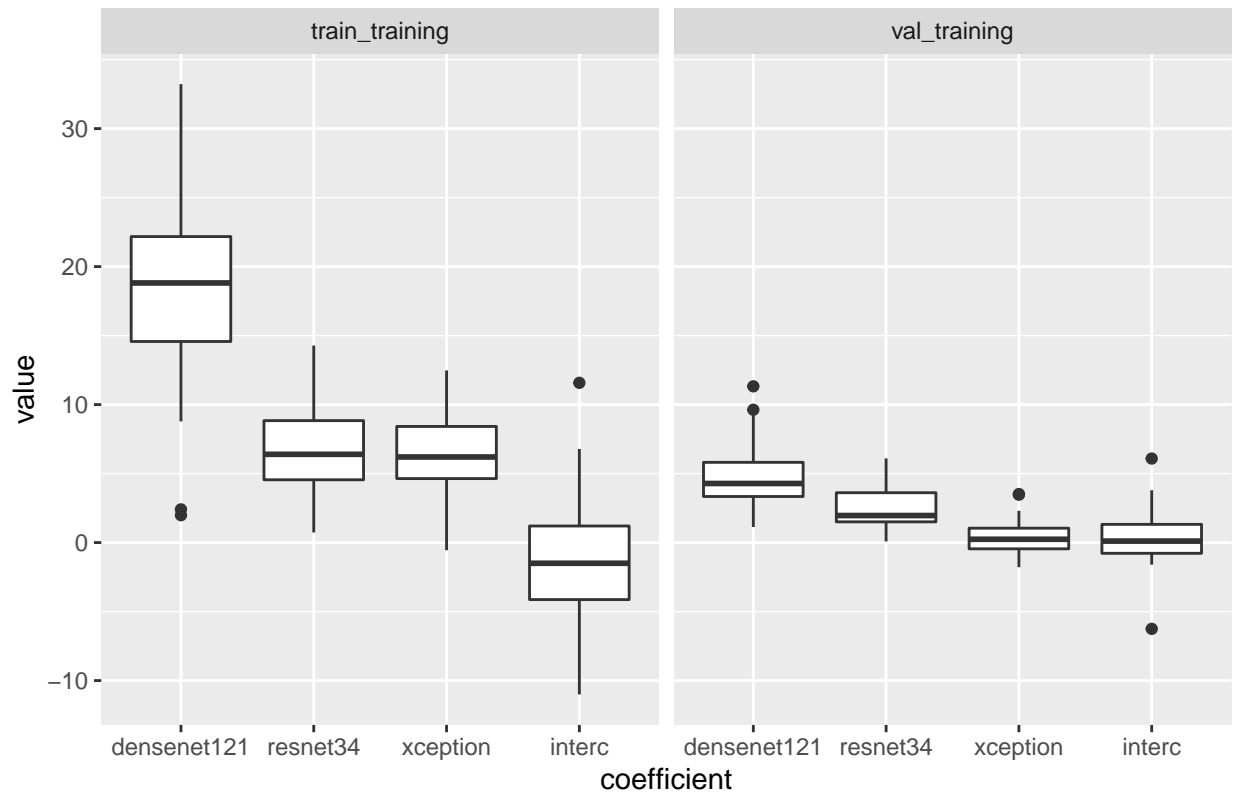
Coefficients for class 1 vs 5



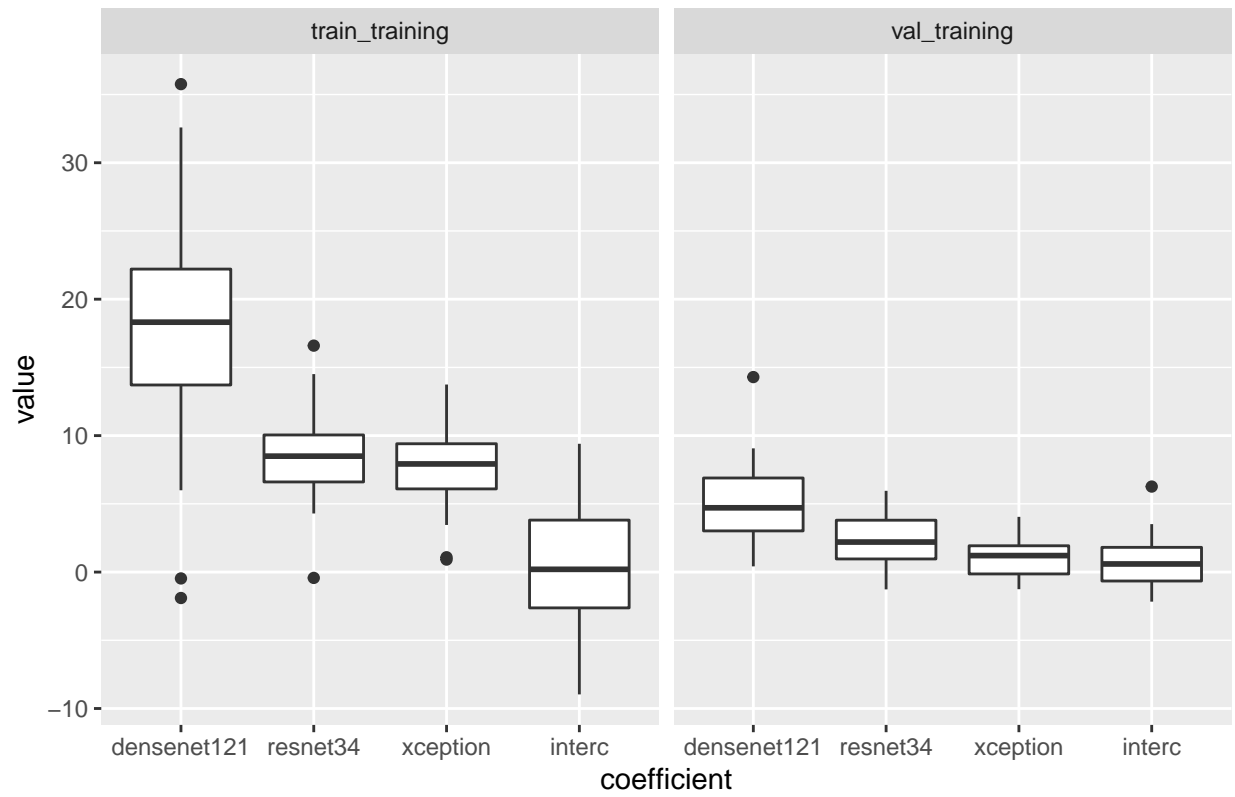
Coefficients for class 1 vs 6



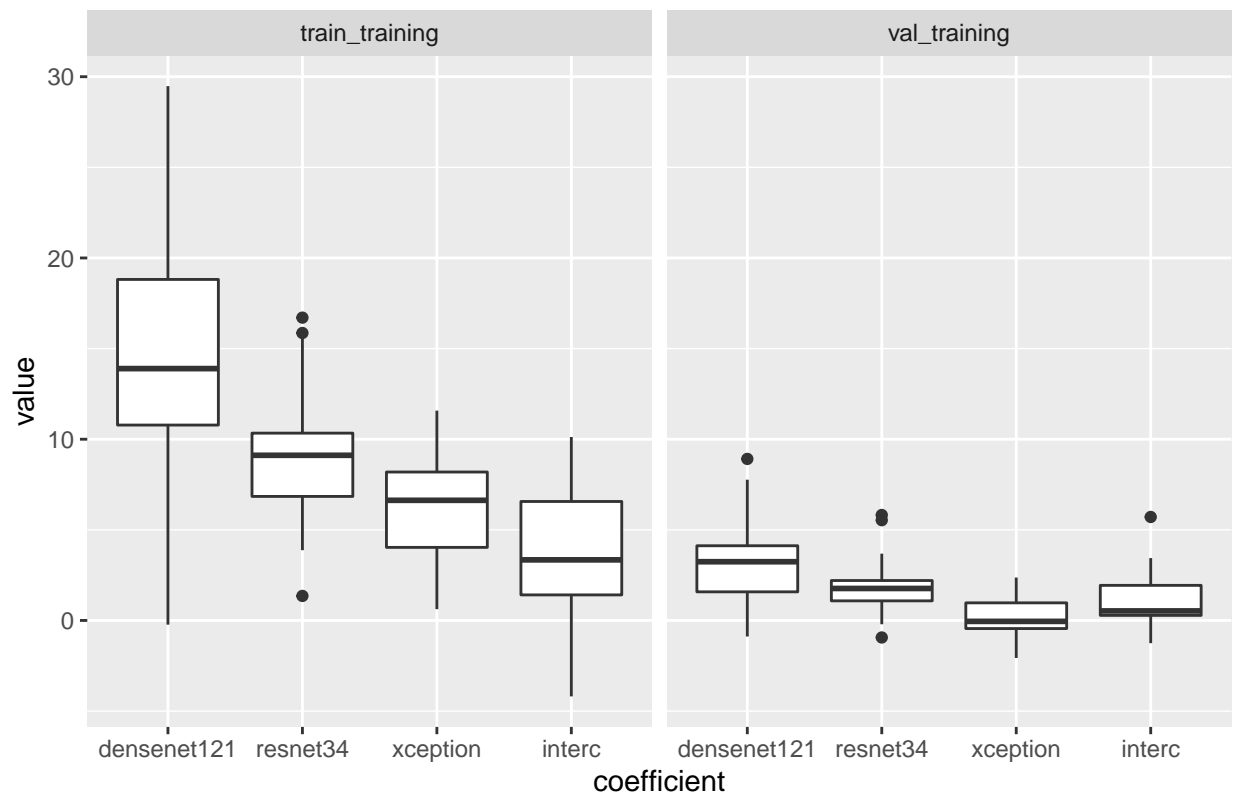
Coefficients for class 1 vs 7



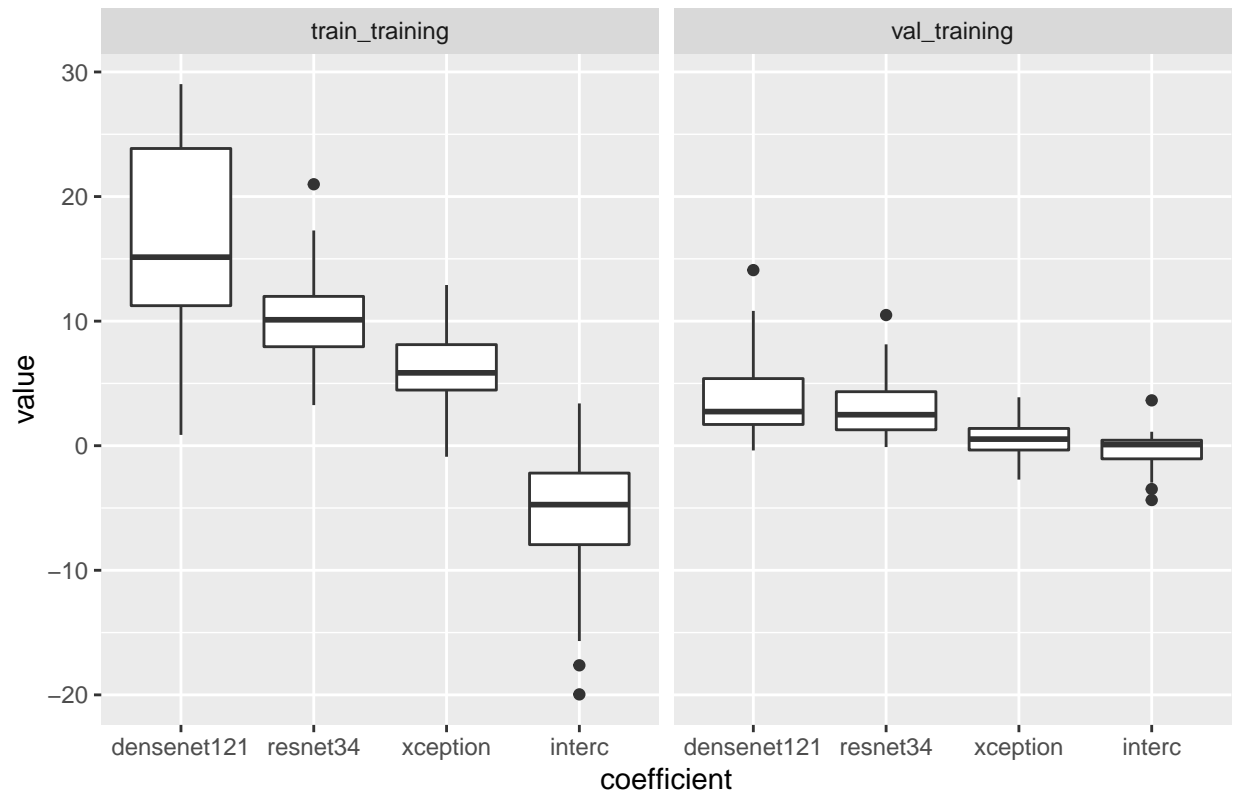
Coefficients for class 1 vs 8



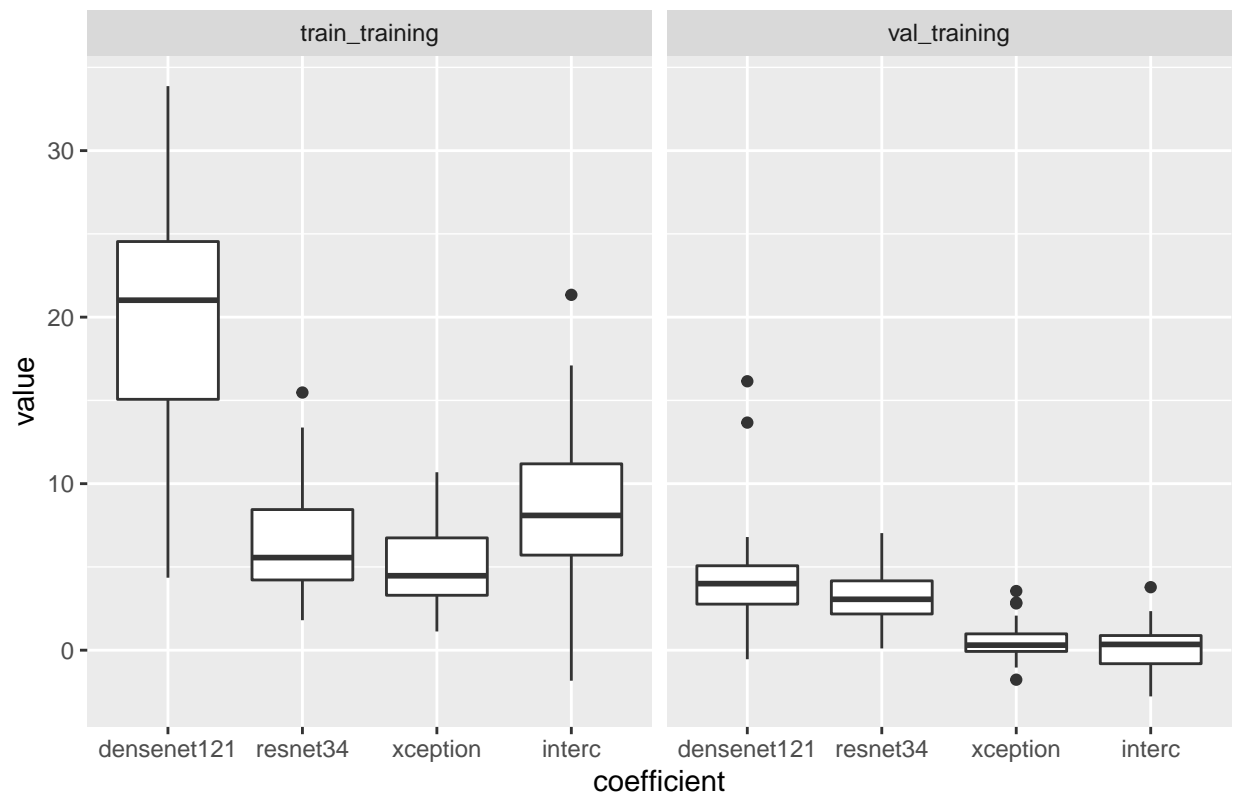
Coefficients for class 1 vs 9



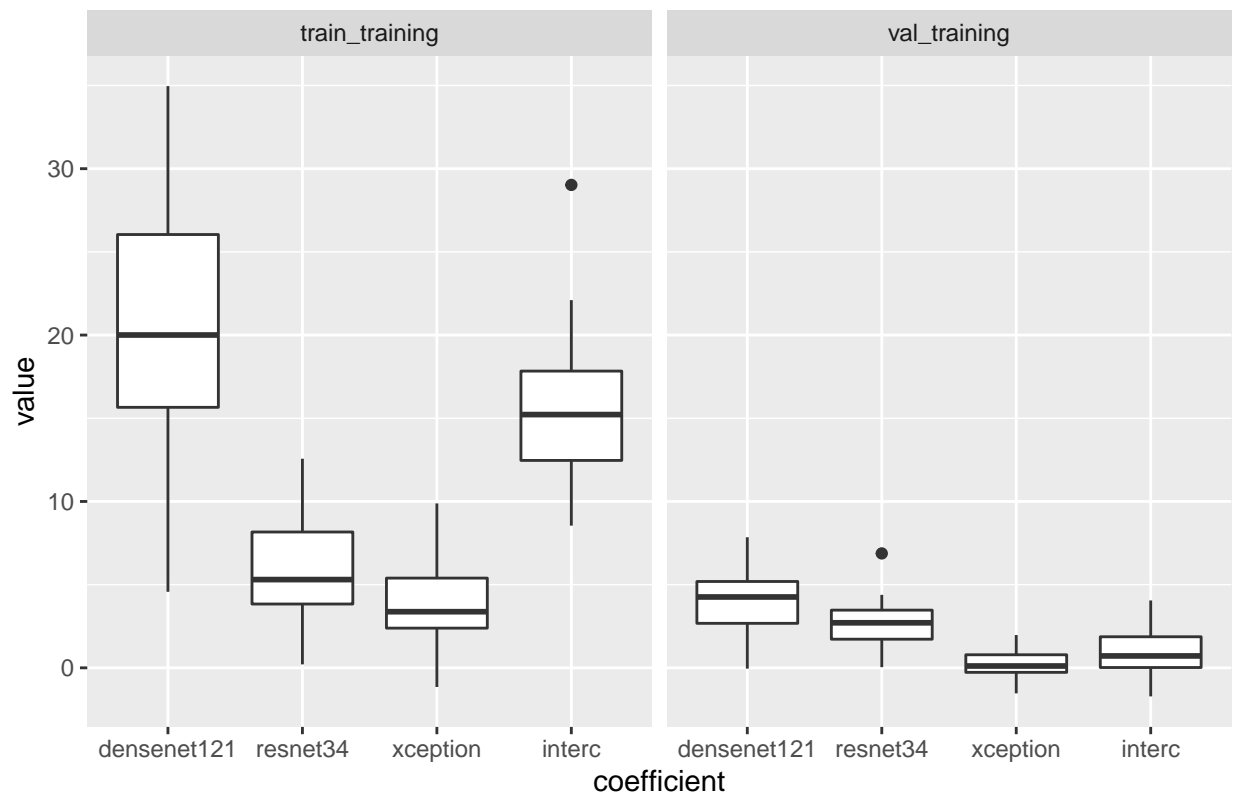
Coefficients for class 1 vs 10



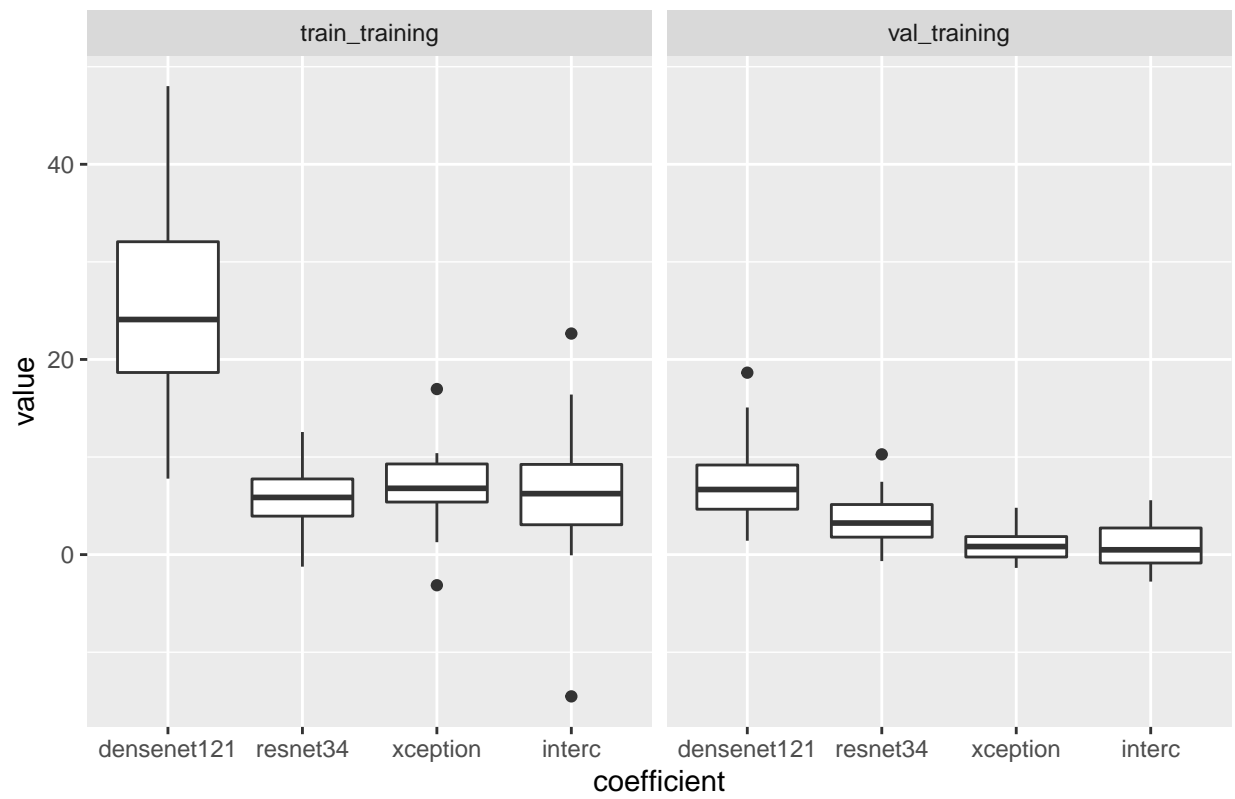
Coefficients for class 2 vs 3



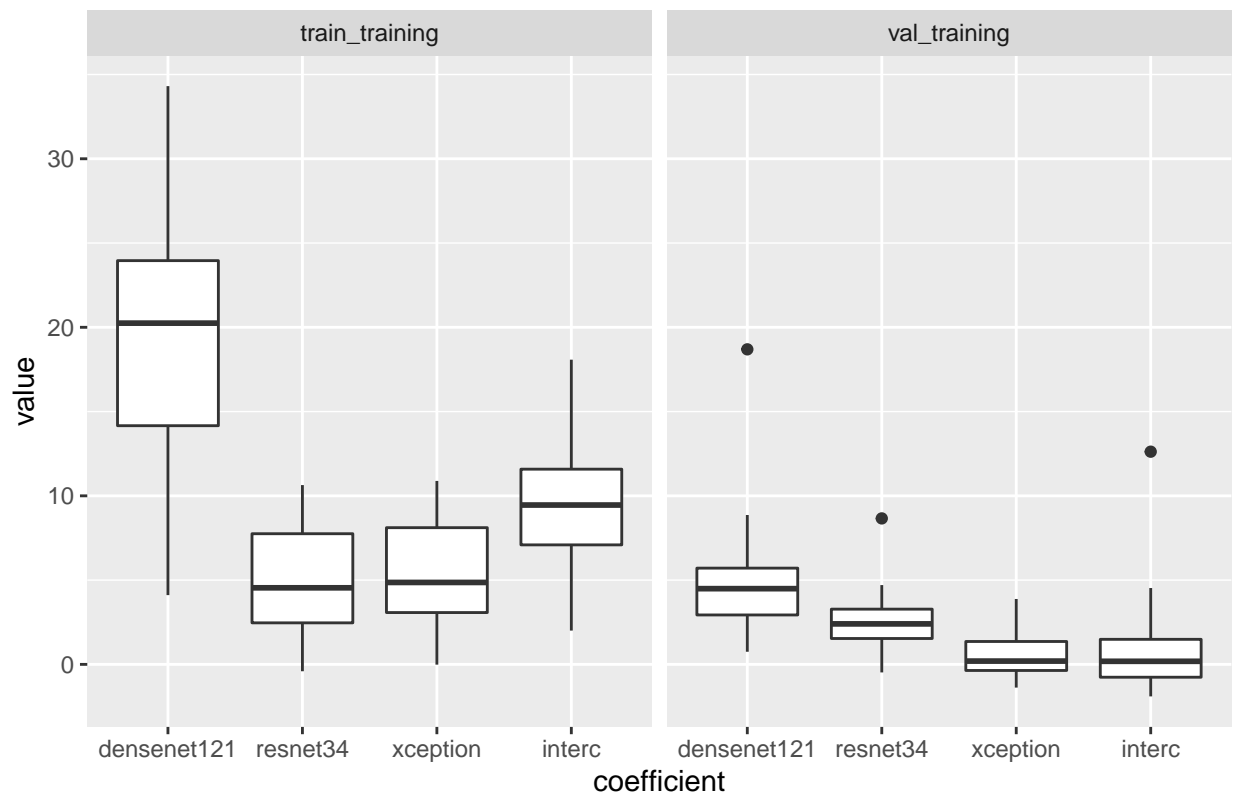
Coefficients for class 2 vs 4



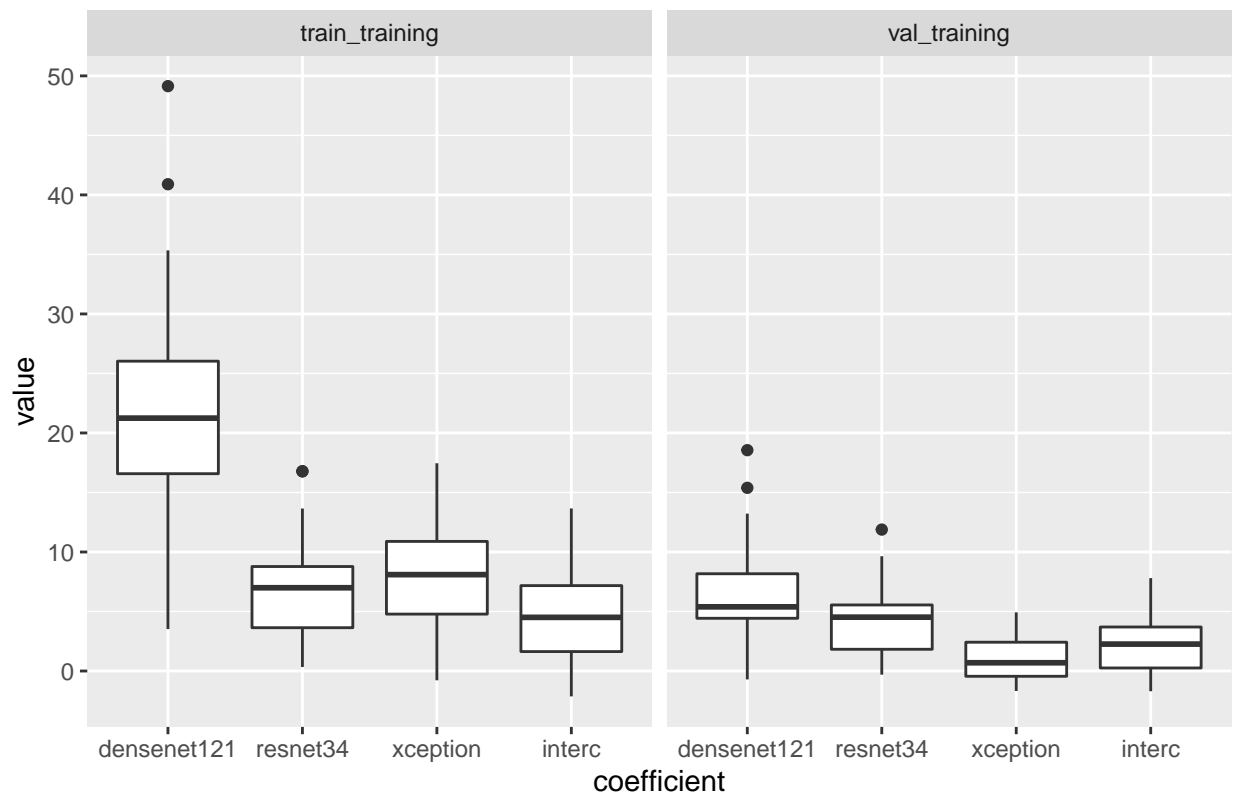
Coefficients for class 2 vs 5



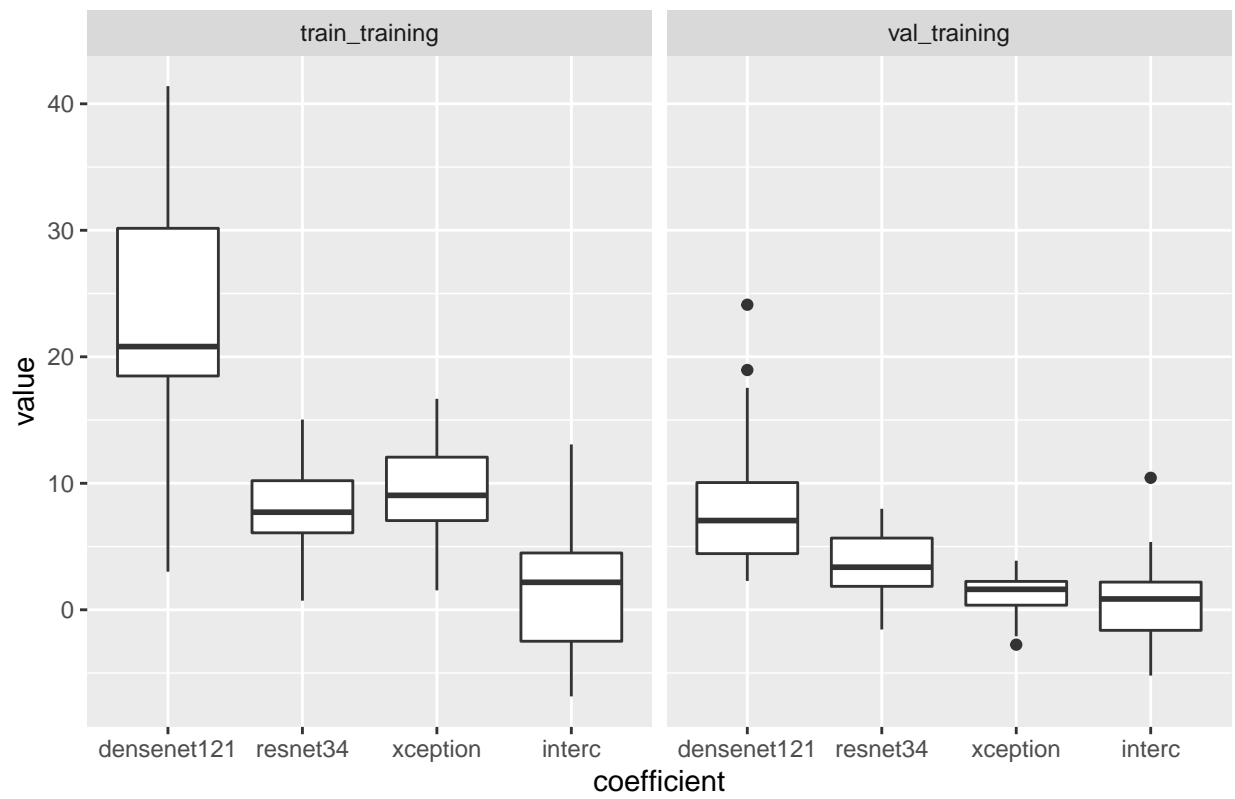
Coefficients for class 2 vs 6



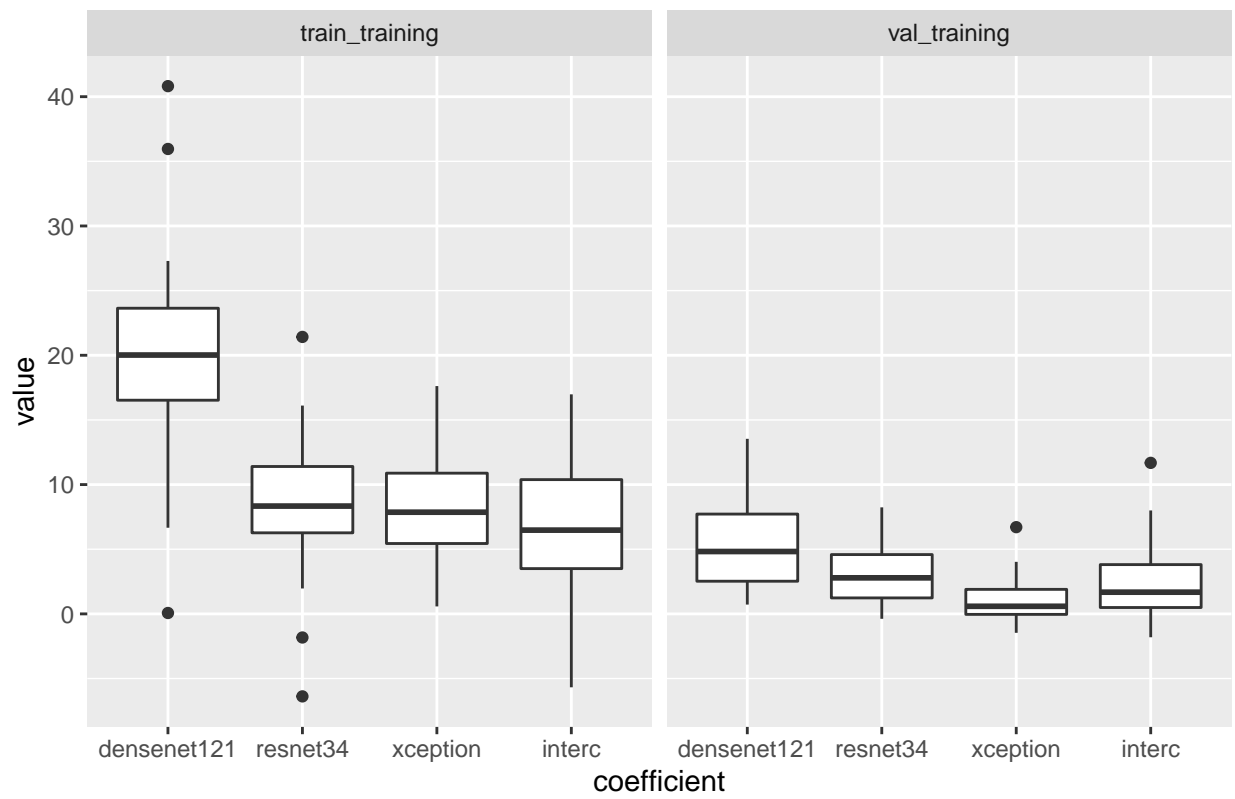
Coefficients for class 2 vs 7



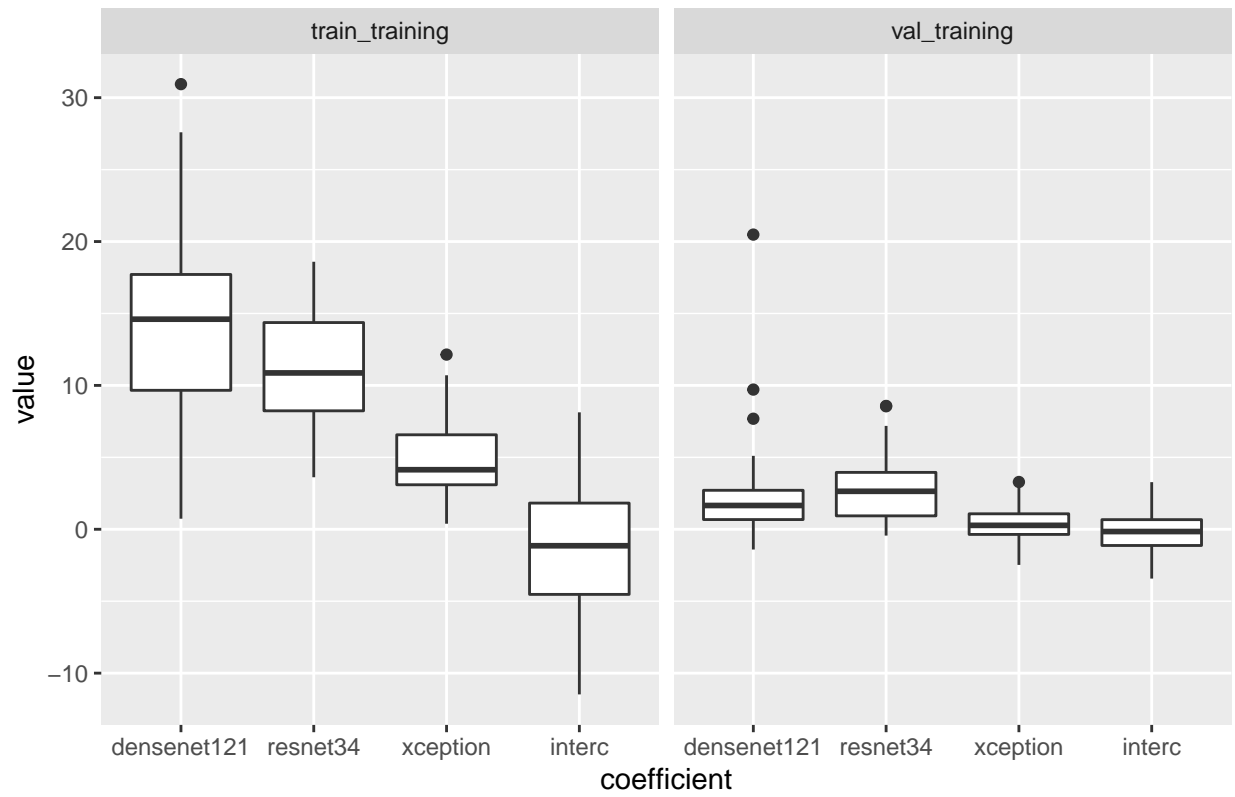
Coefficients for class 2 vs 8



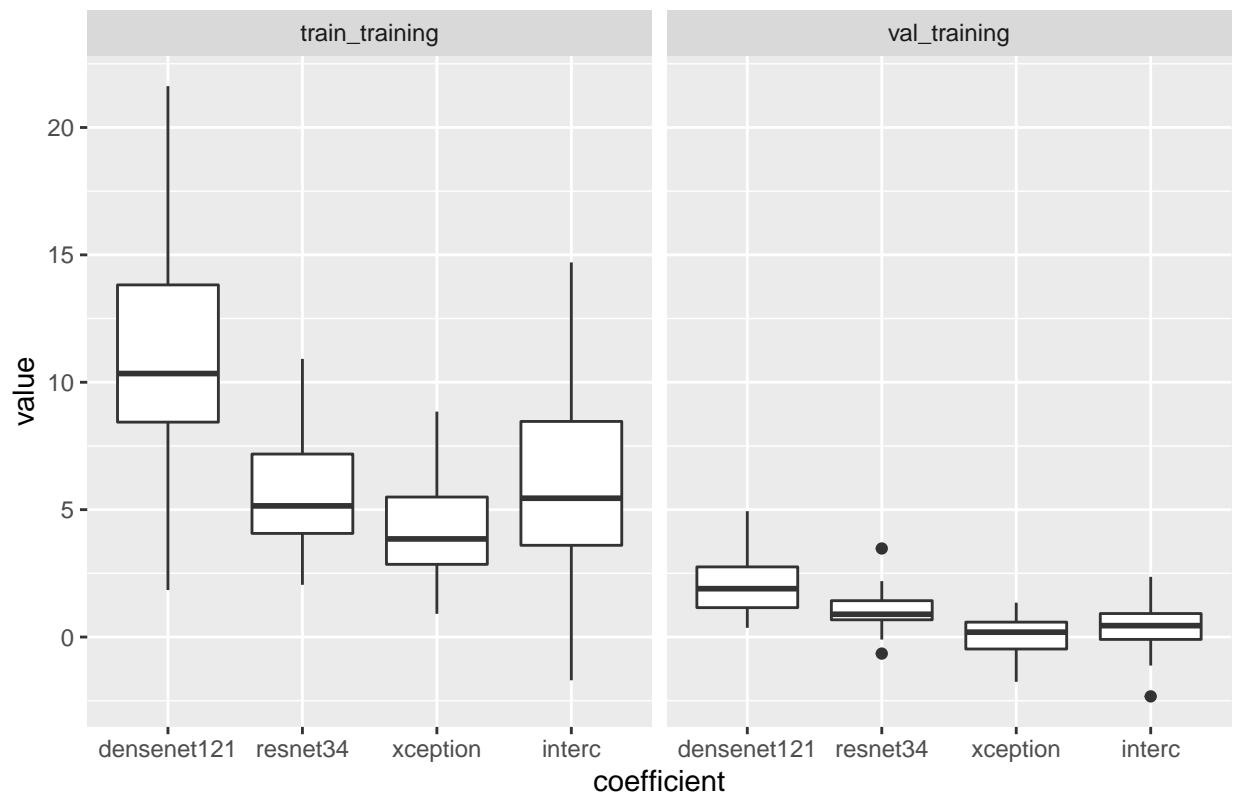
Coefficients for class 2 vs 9



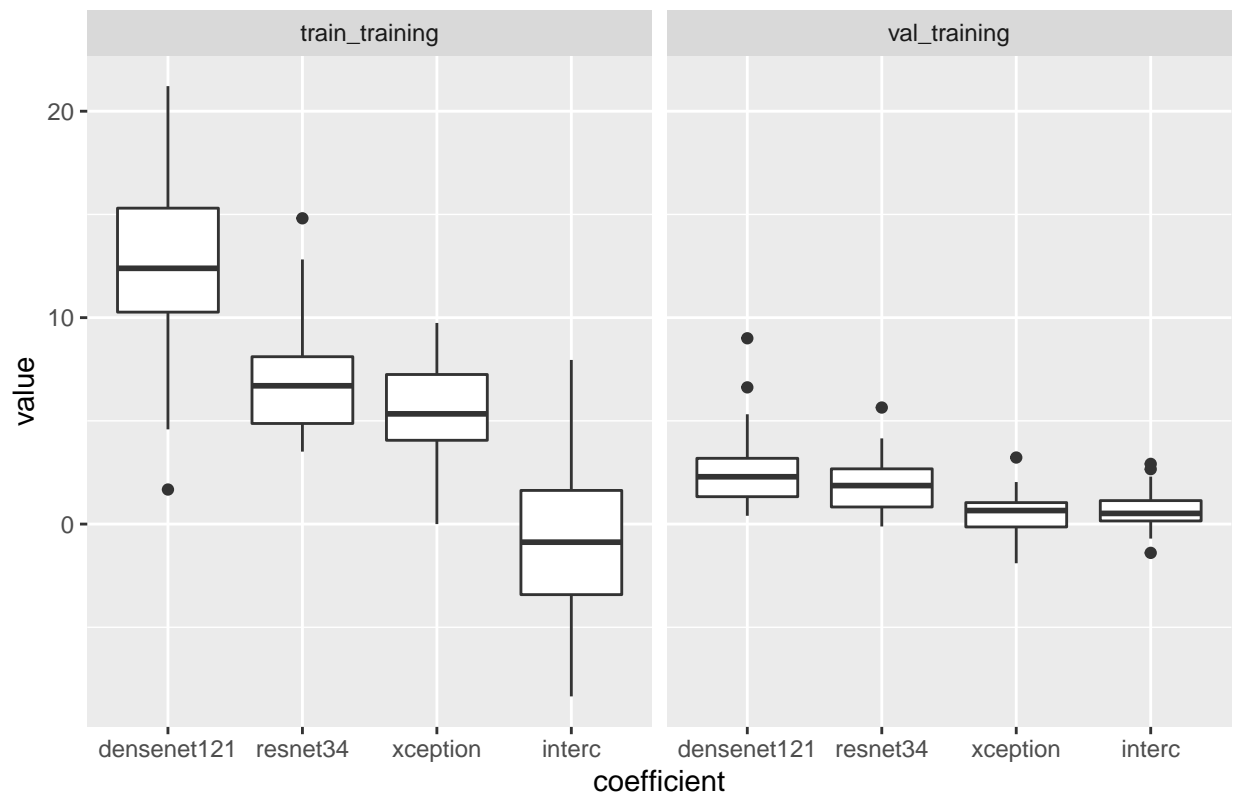
Coefficients for class 2 vs 10



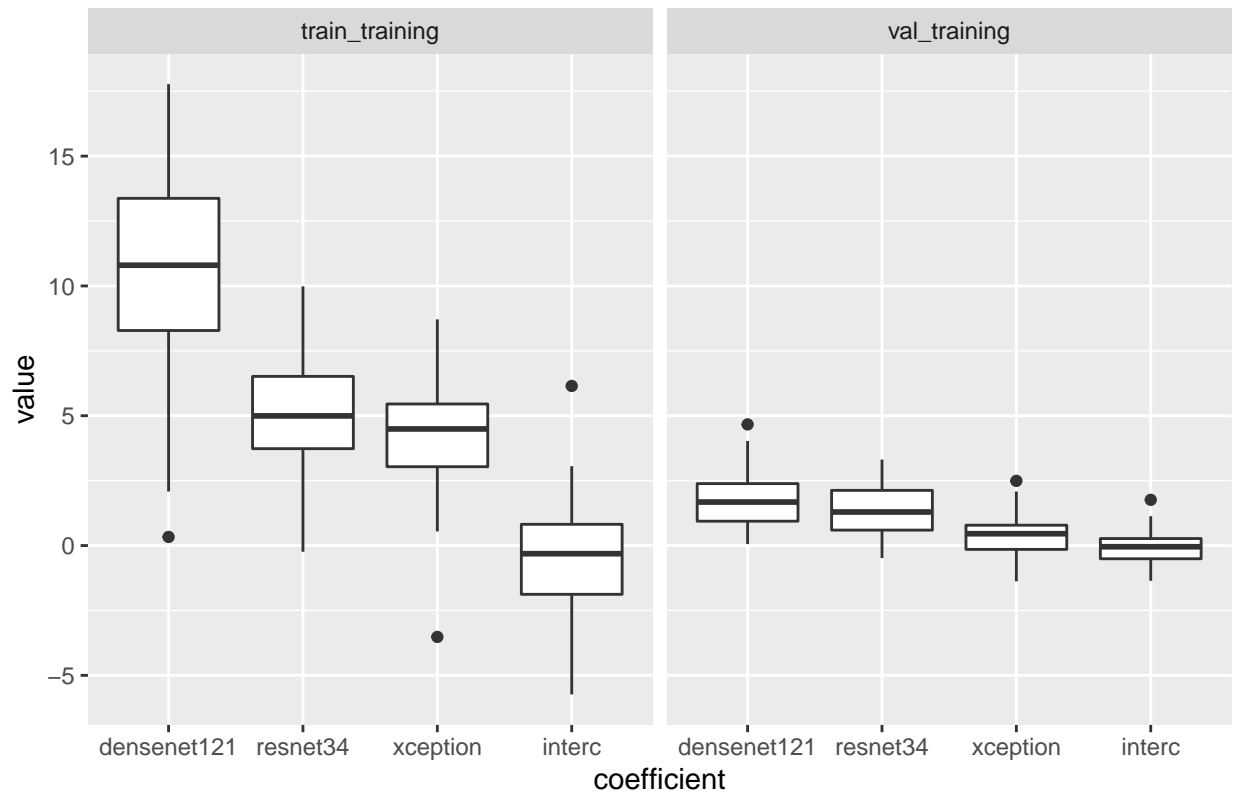
Coefficients for class 3 vs 4



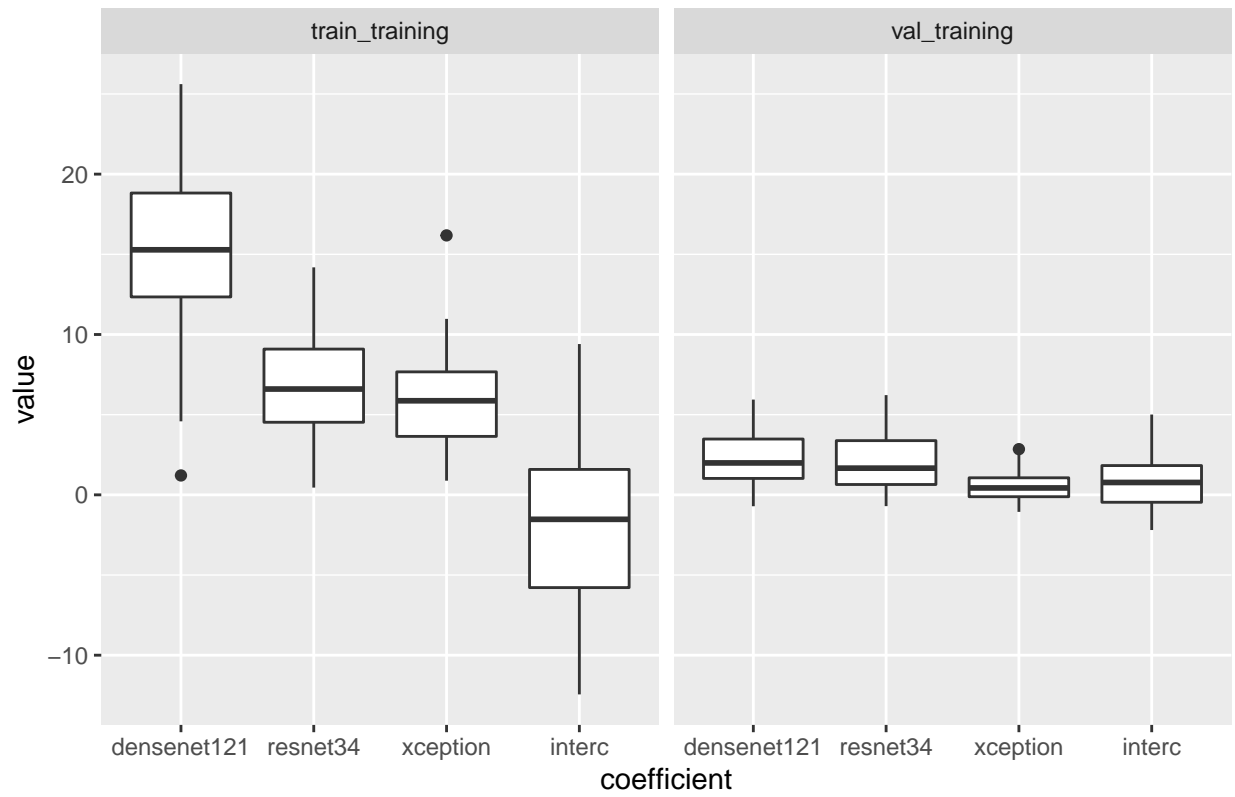
Coefficients for class 3 vs 5



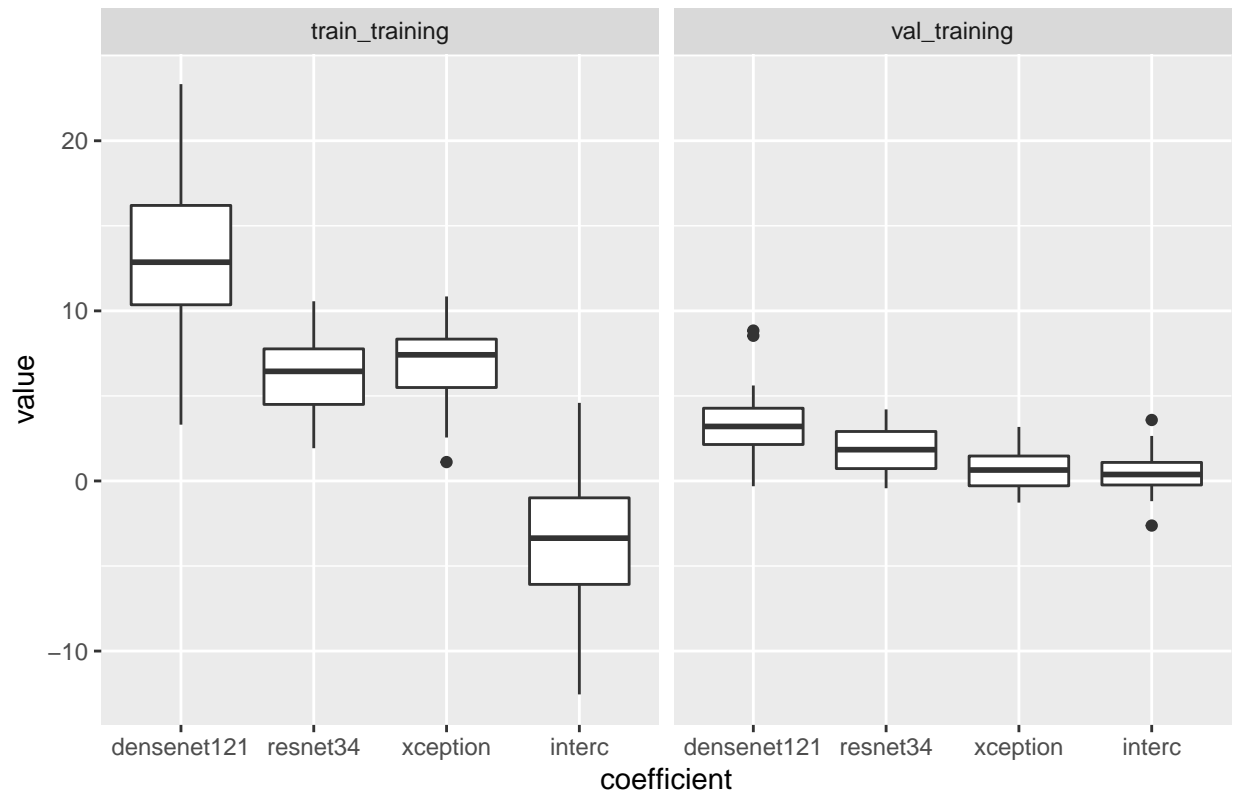
Coefficients for class 3 vs 6



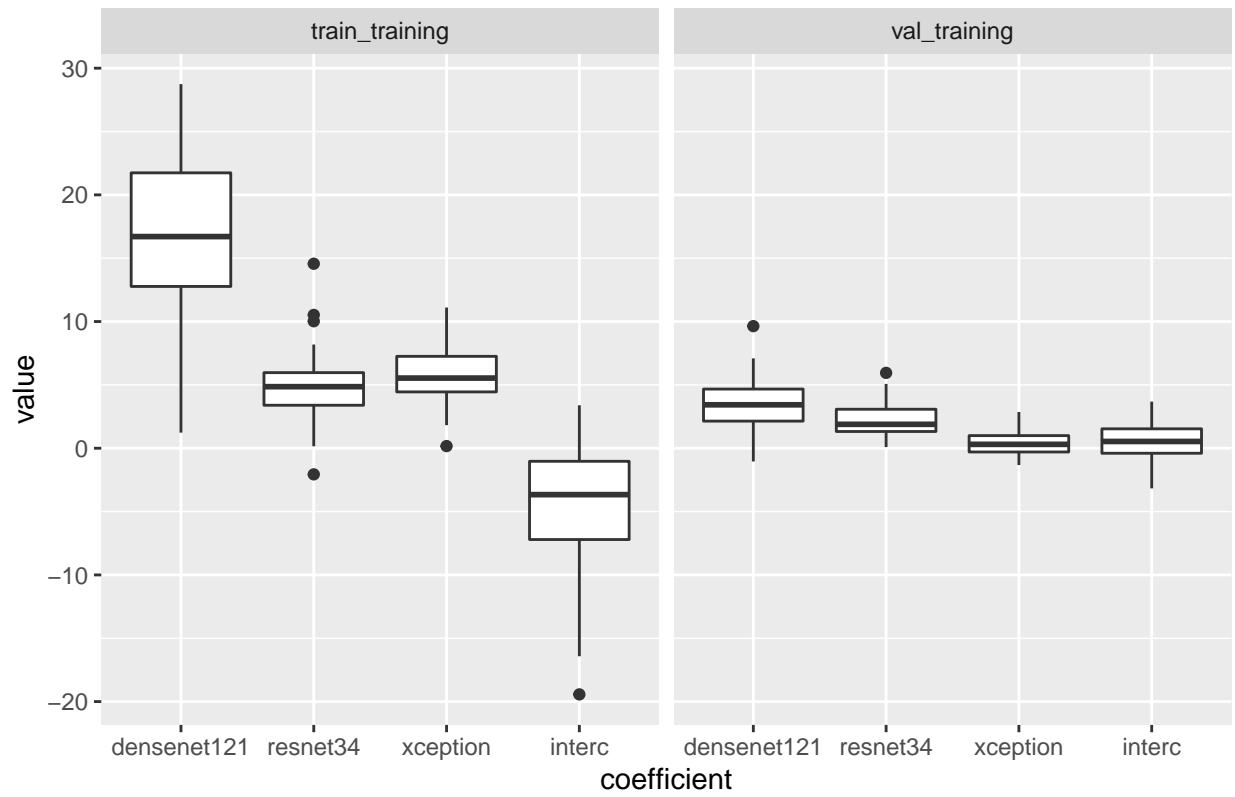
Coefficients for class 3 vs 7



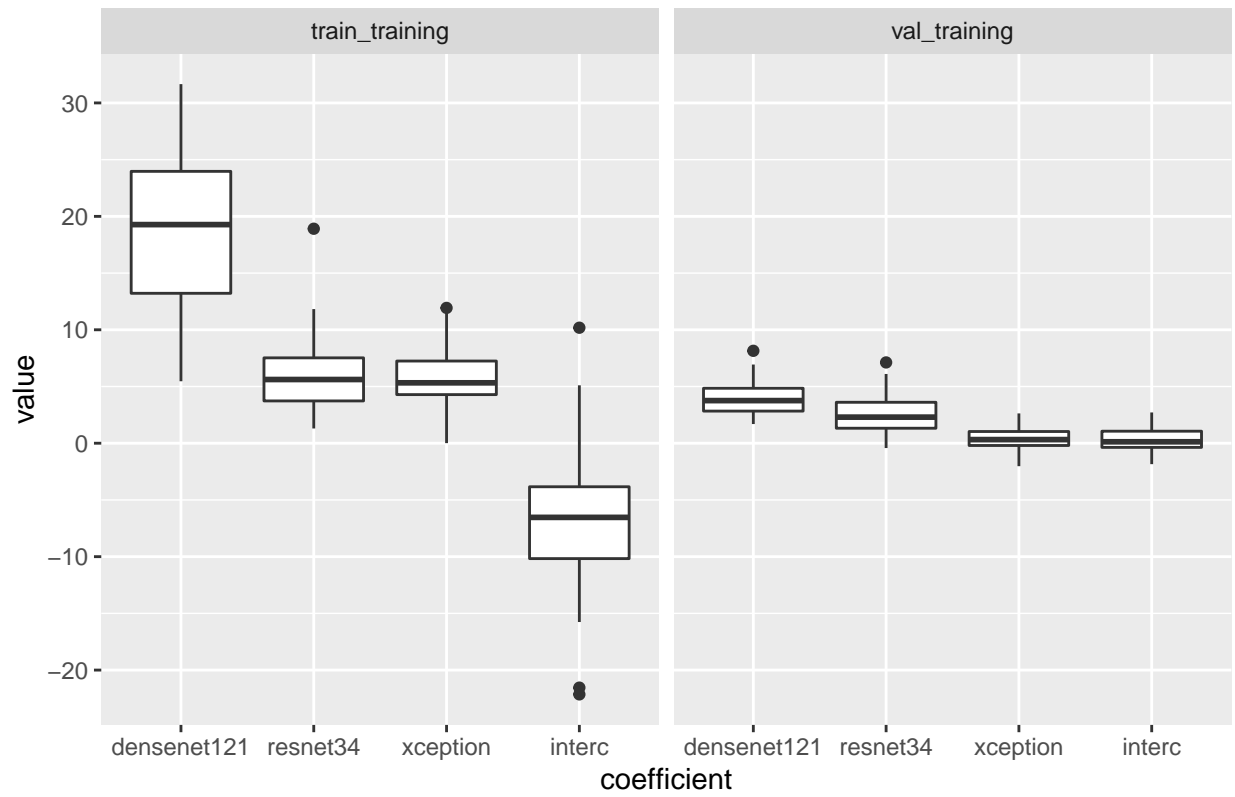
Coefficients for class 3 vs 8



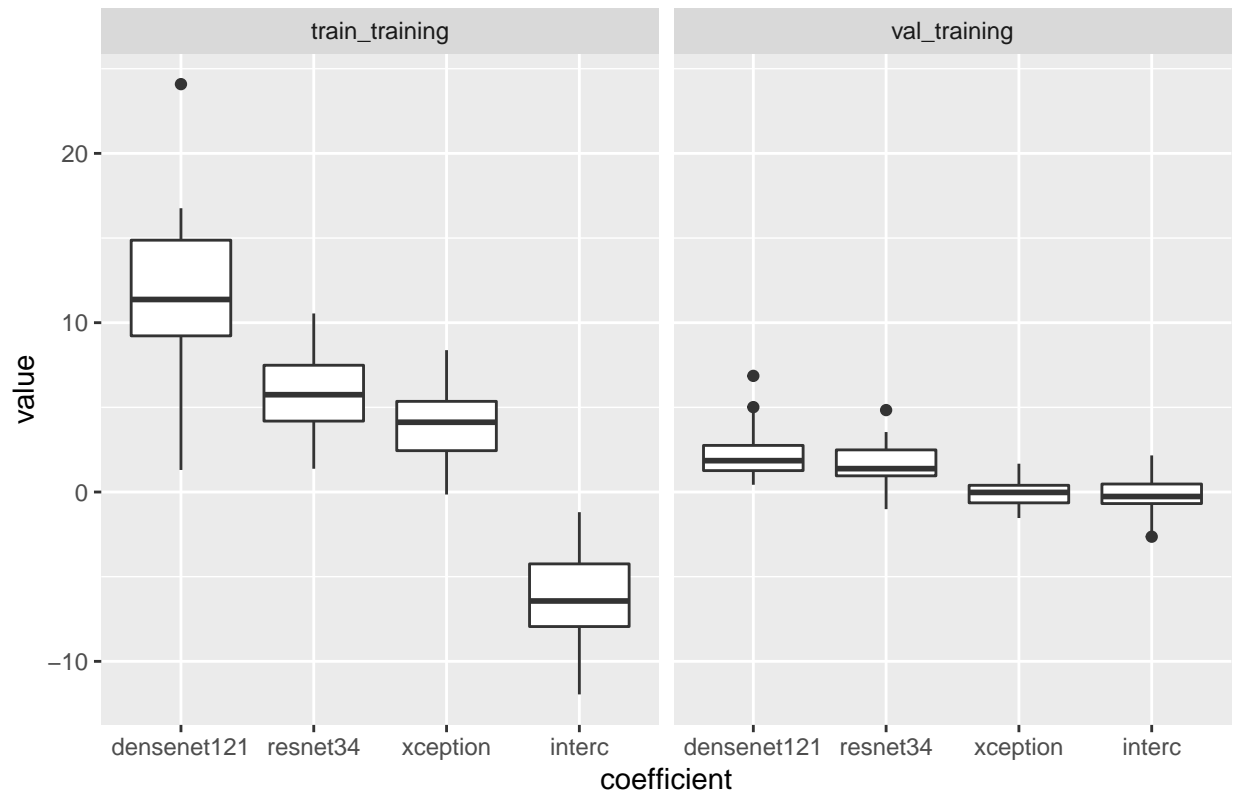
Coefficients for class 3 vs 9



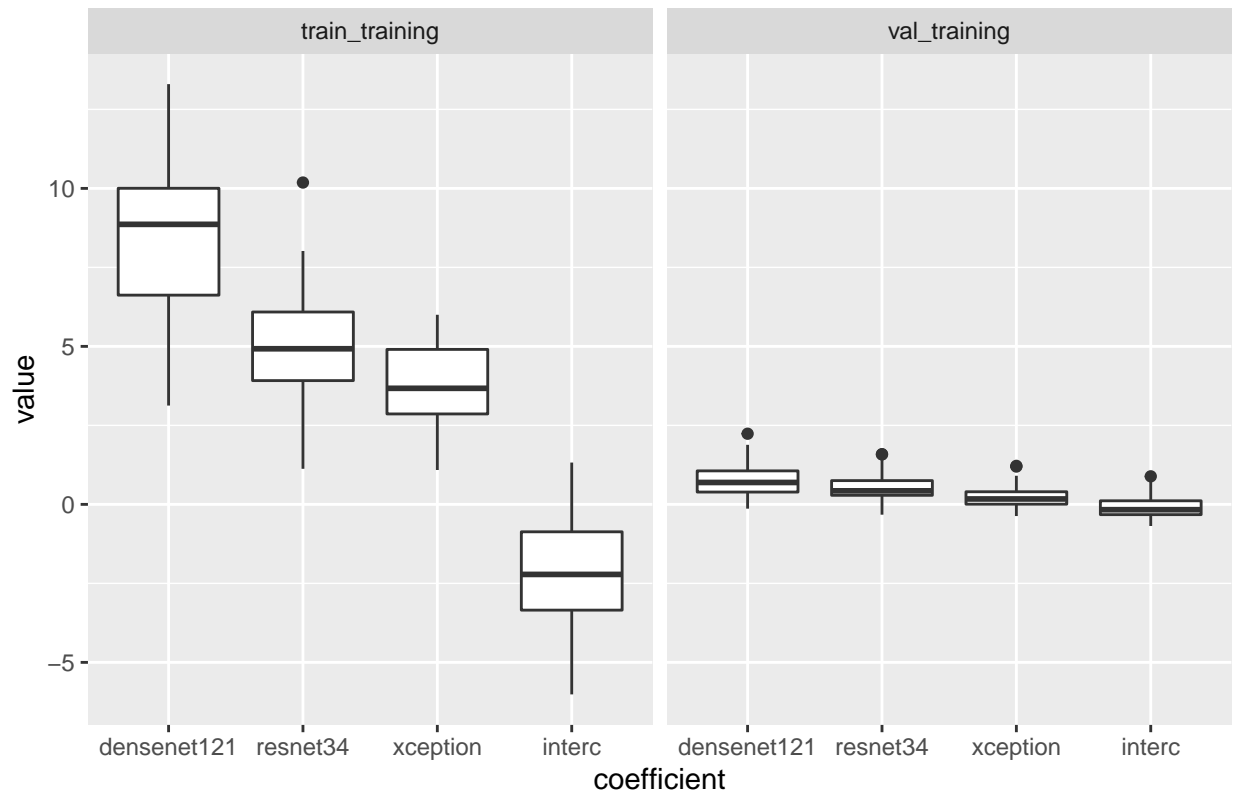
Coefficients for class 3 vs 10



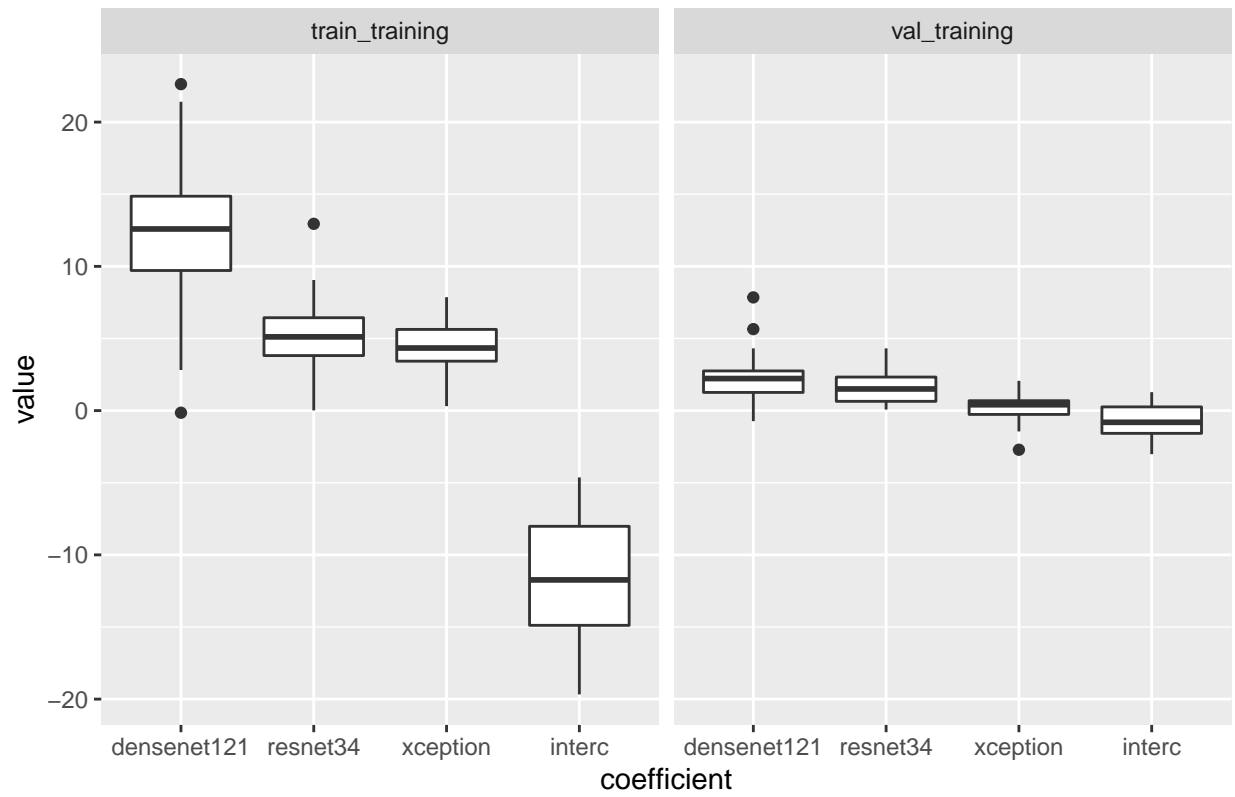
Coefficients for class 4 vs 5



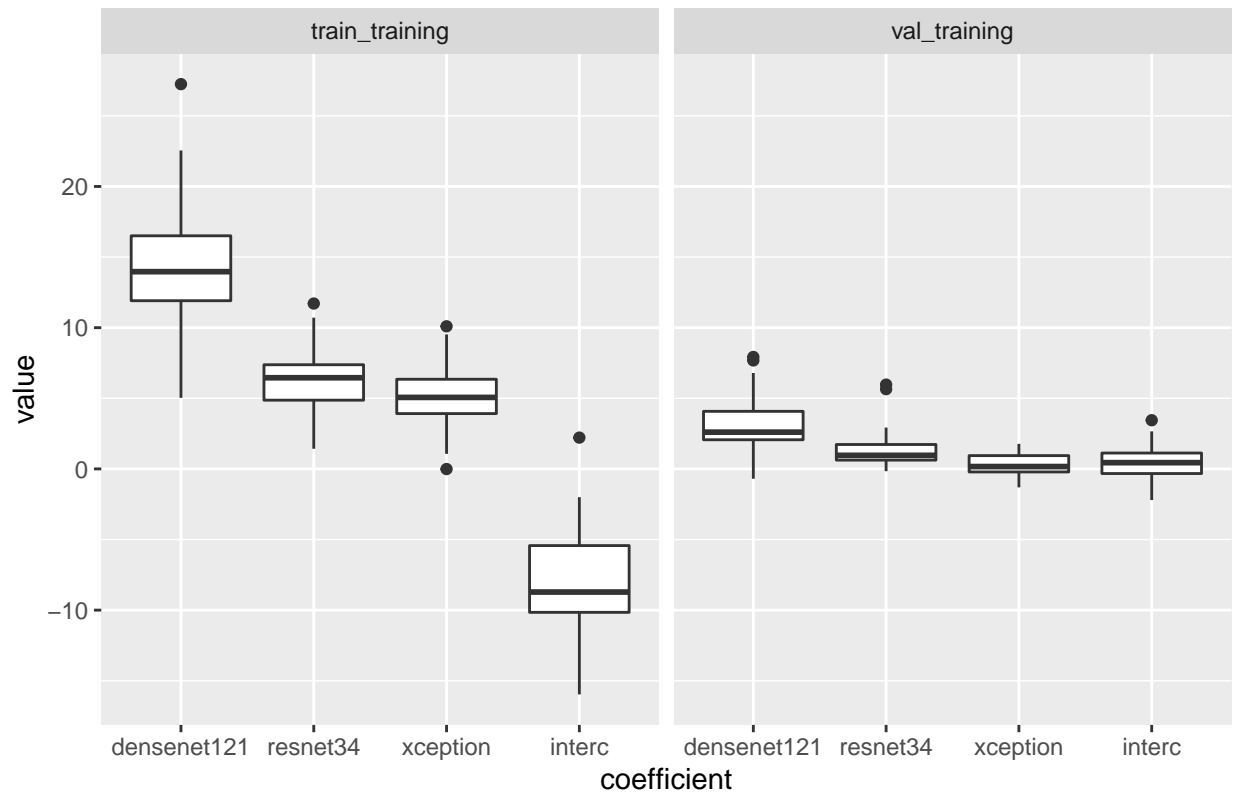
Coefficients for class 4 vs 6



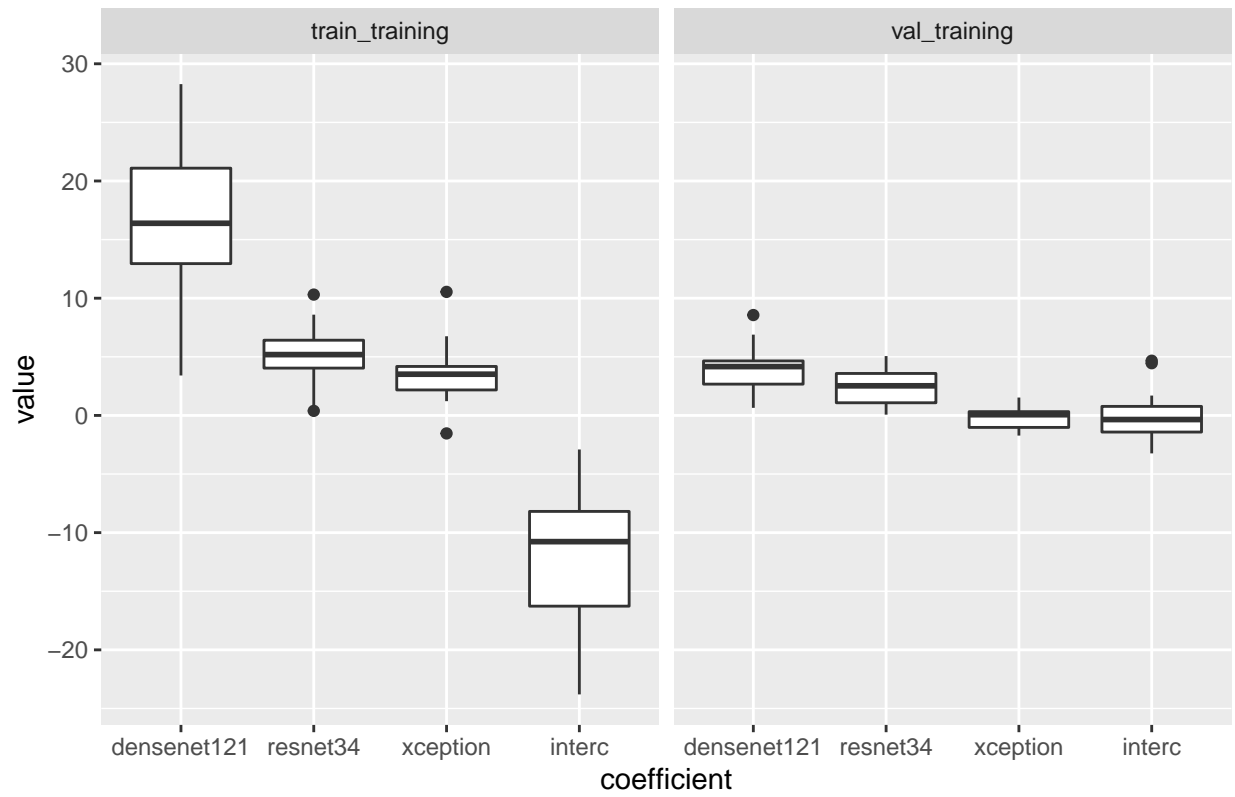
Coefficients for class 4 vs 7



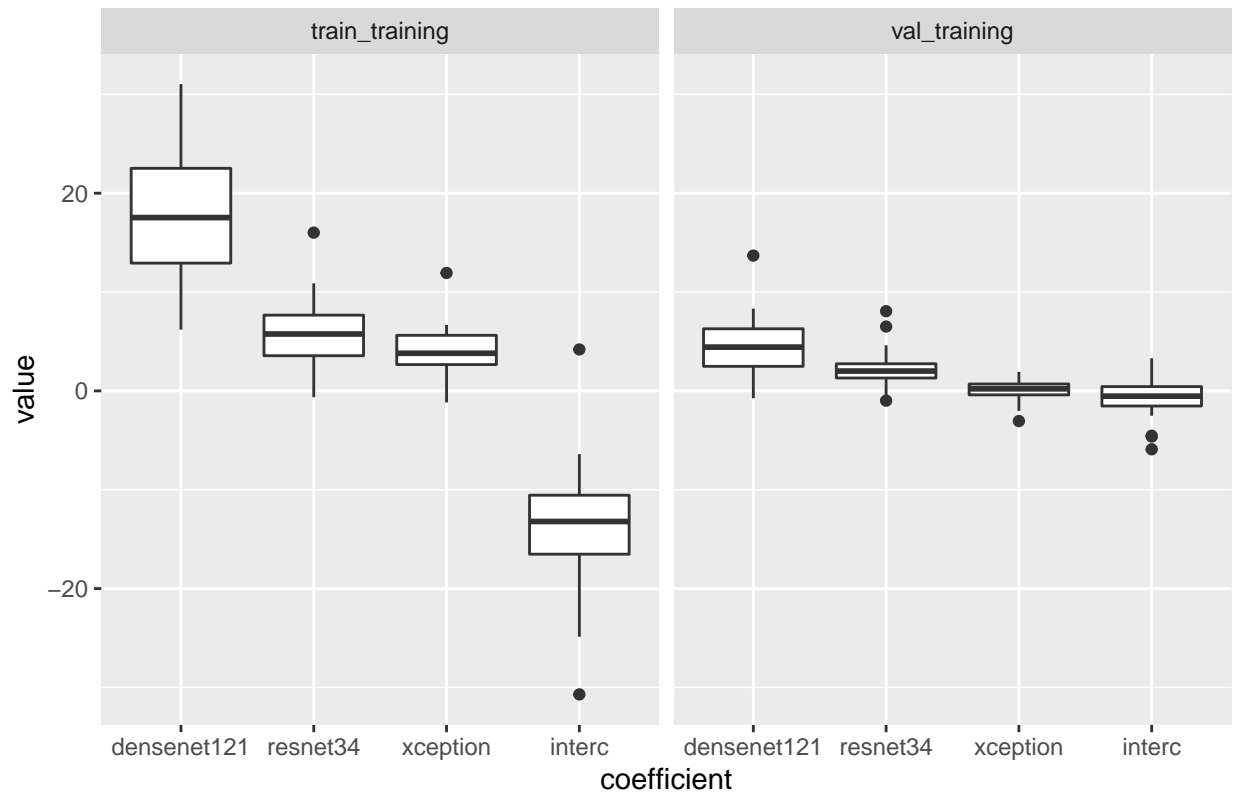
Coefficients for class 4 vs 8



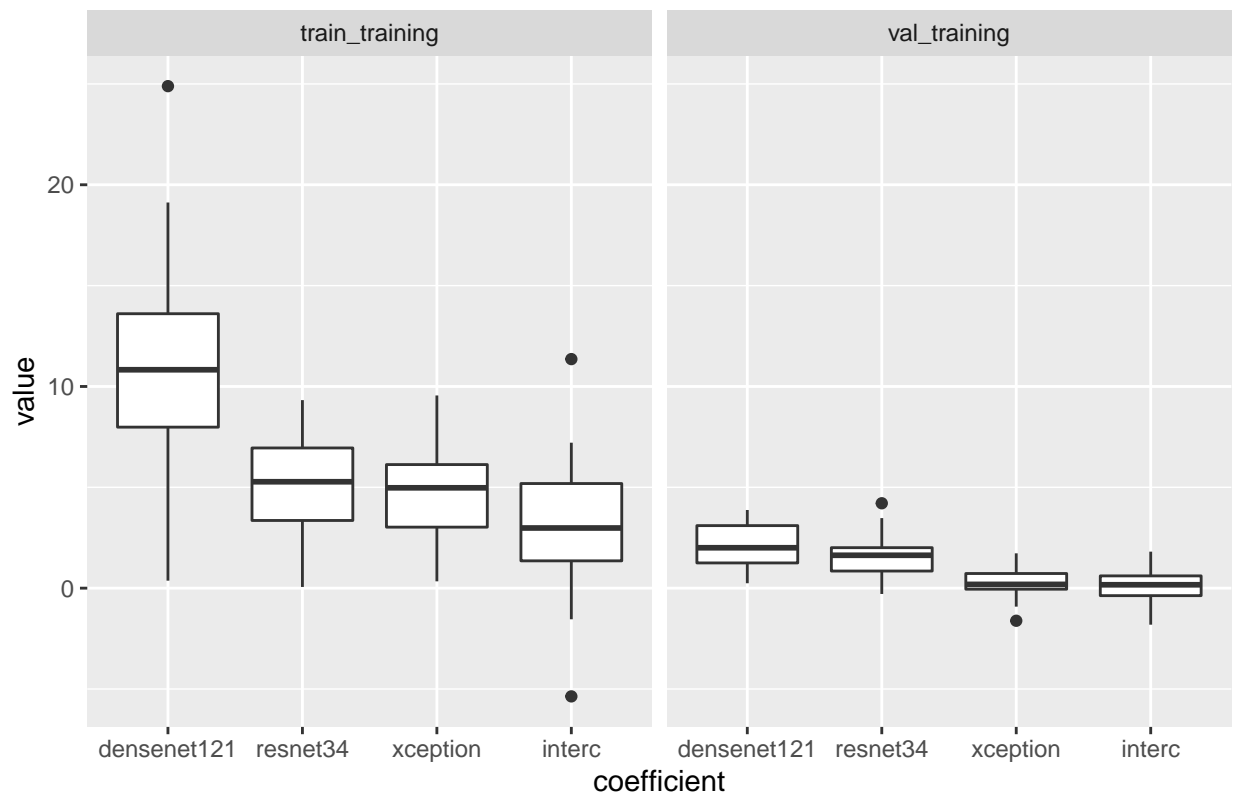
Coefficients for class 4 vs 9



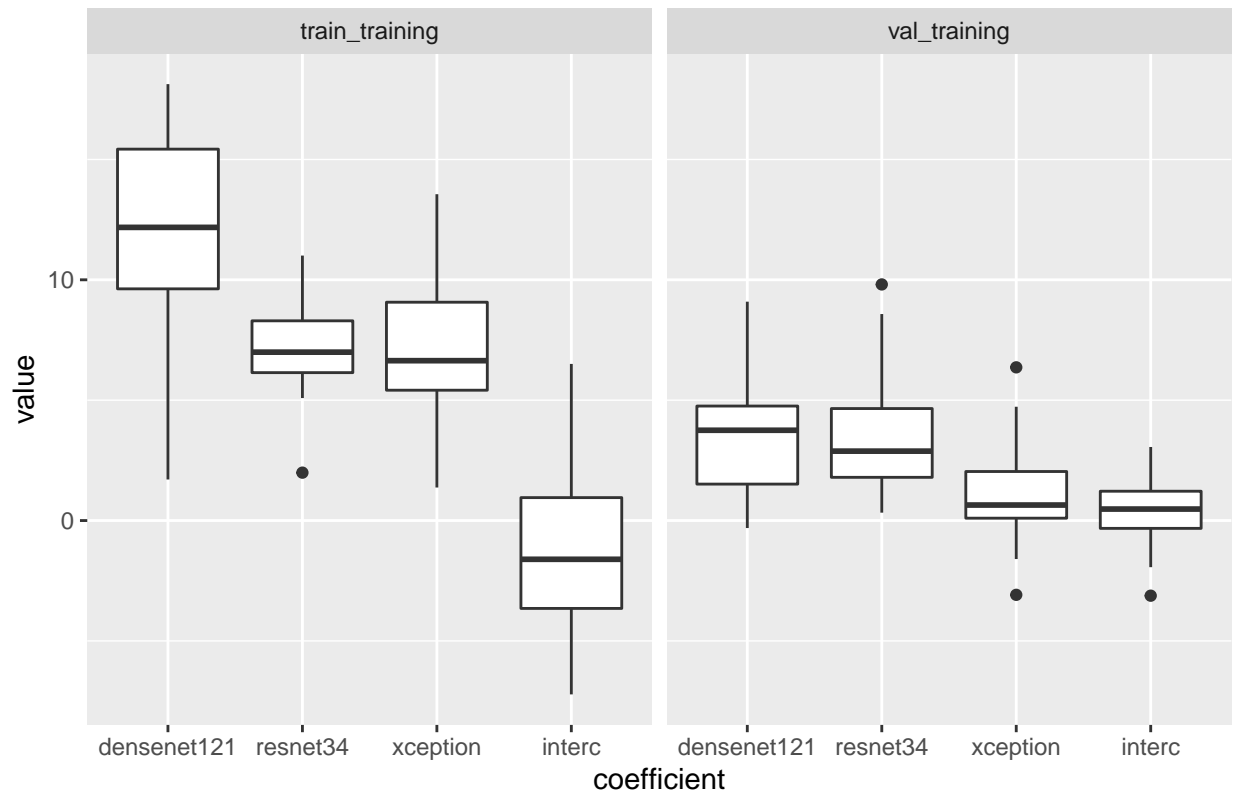
Coefficients for class 4 vs 10



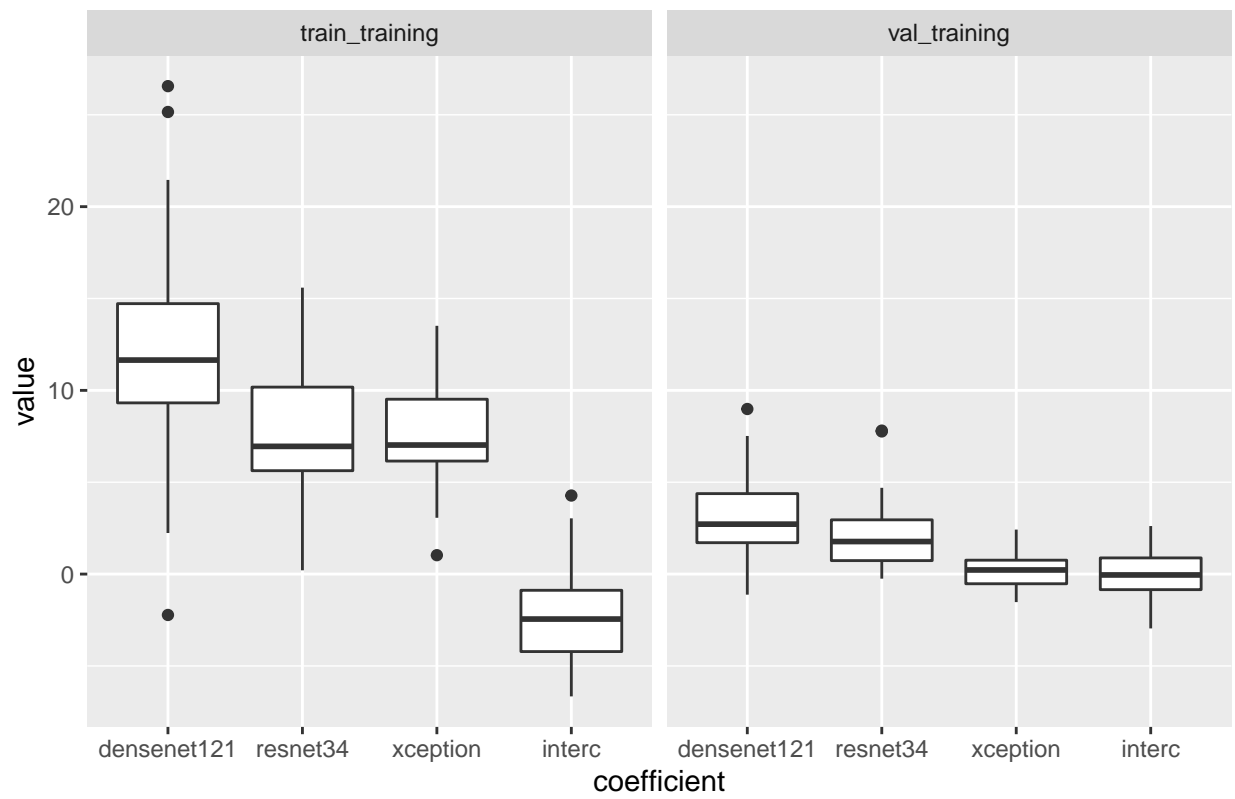
Coefficients for class 5 vs 6



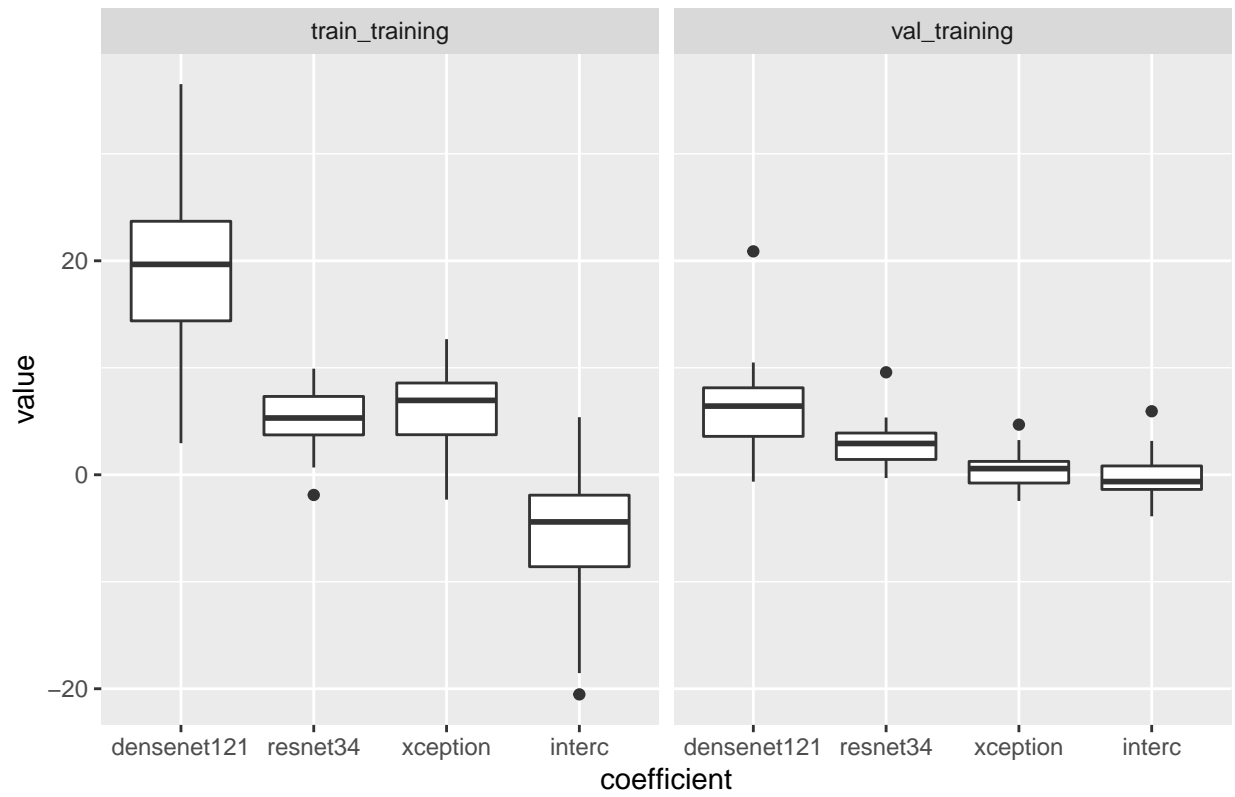
Coefficients for class 5 vs 7



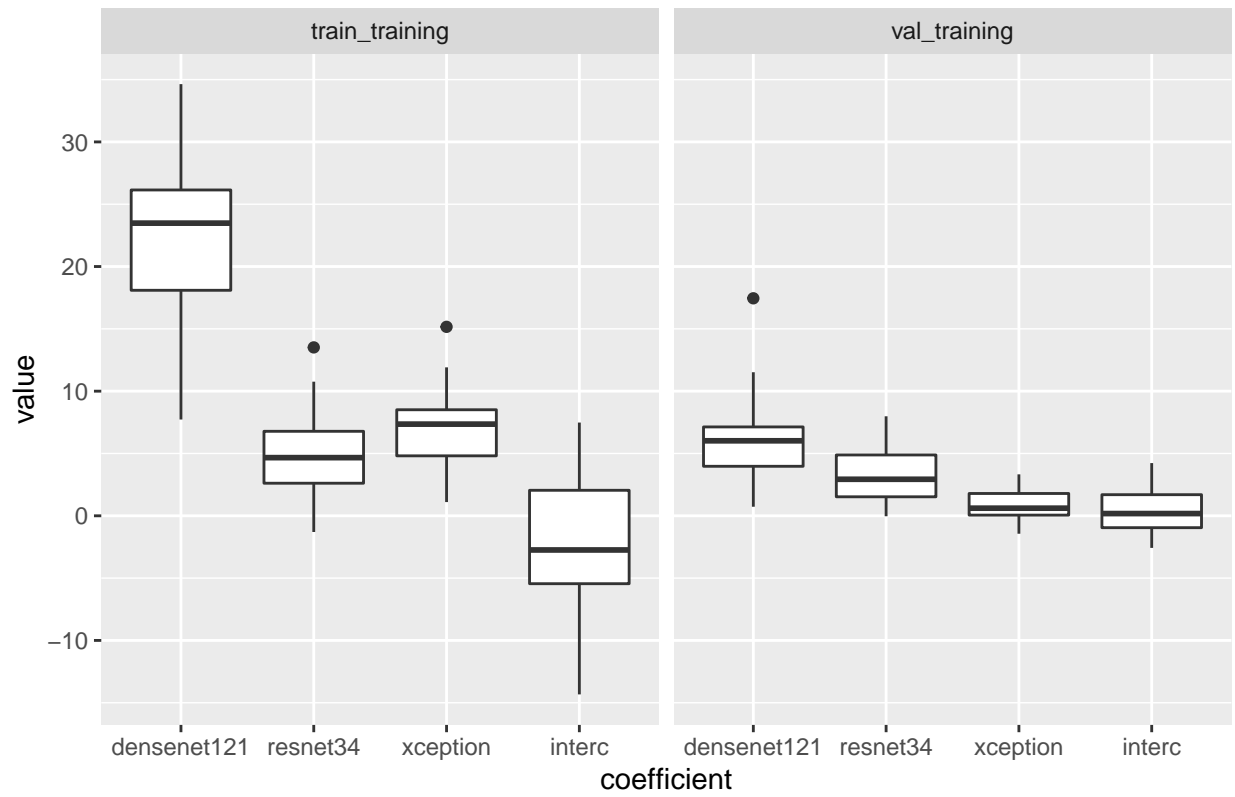
Coefficients for class 5 vs 8



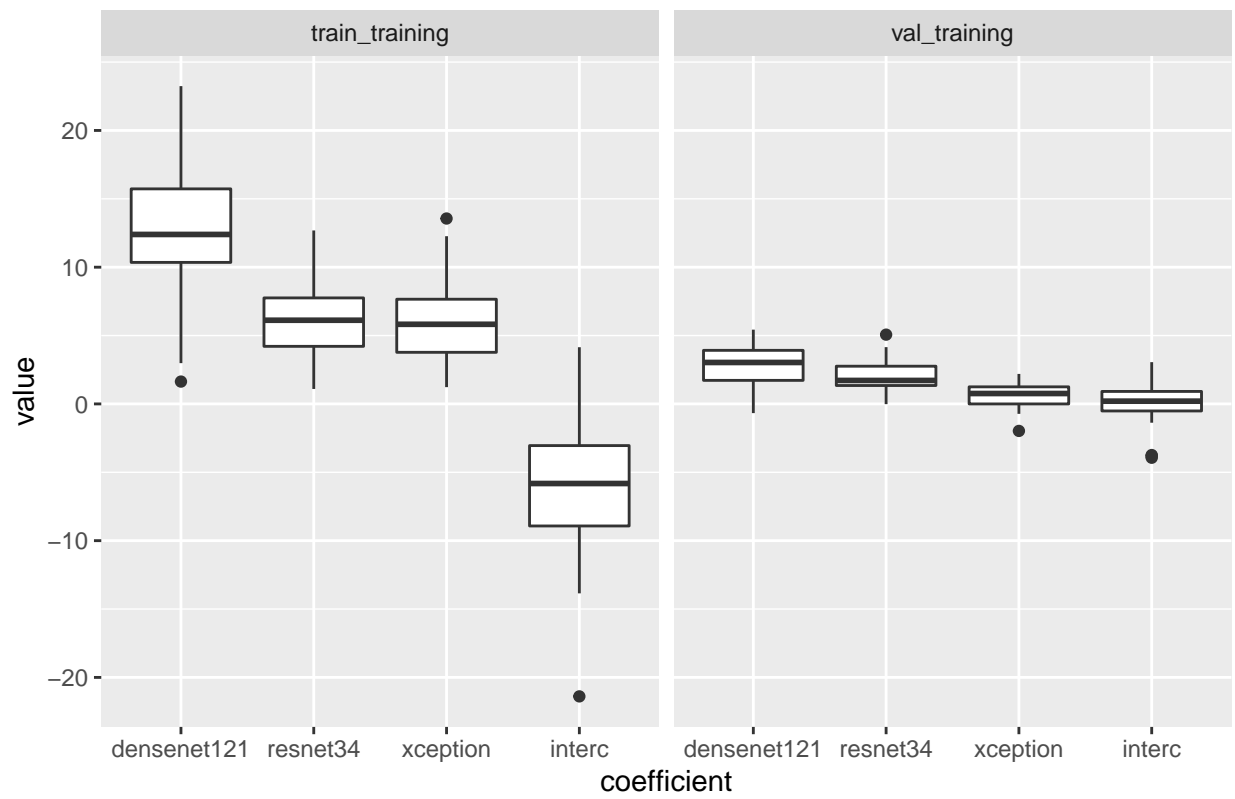
Coefficients for class 5 vs 9



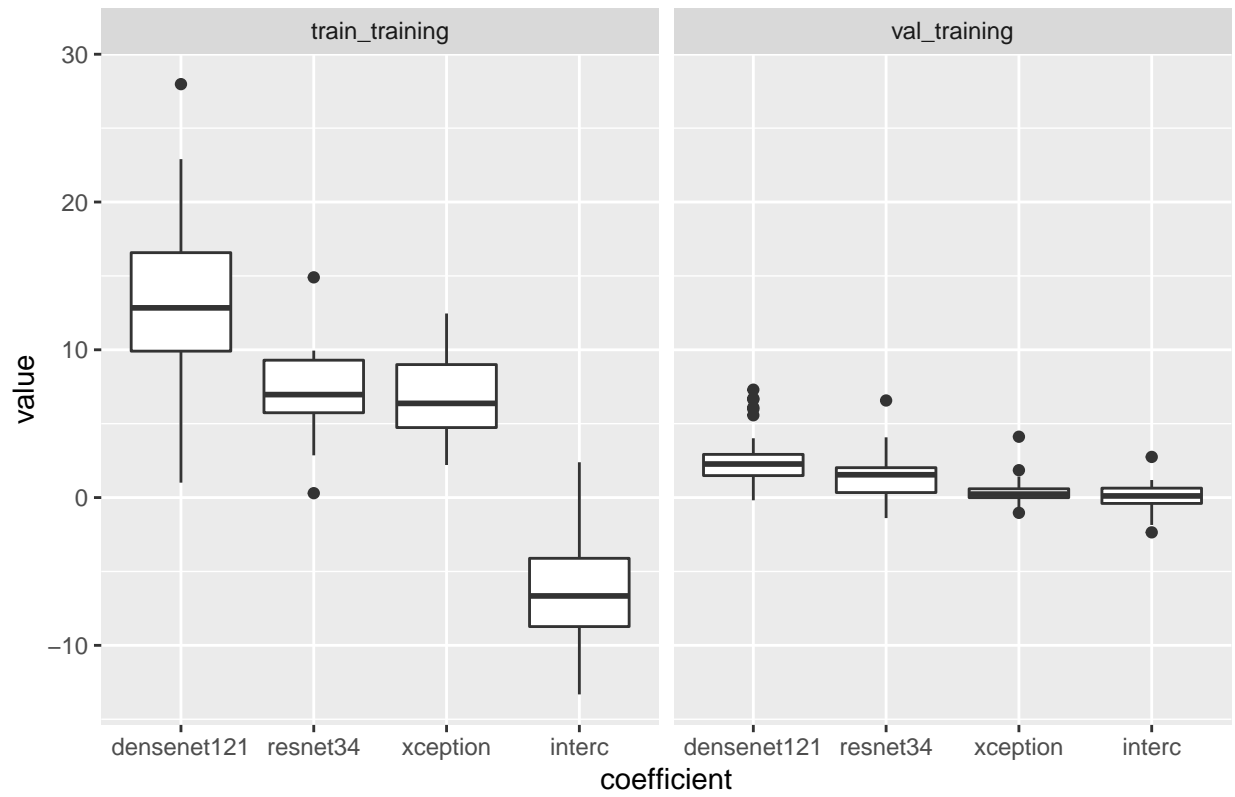
Coefficients for class 5 vs 10



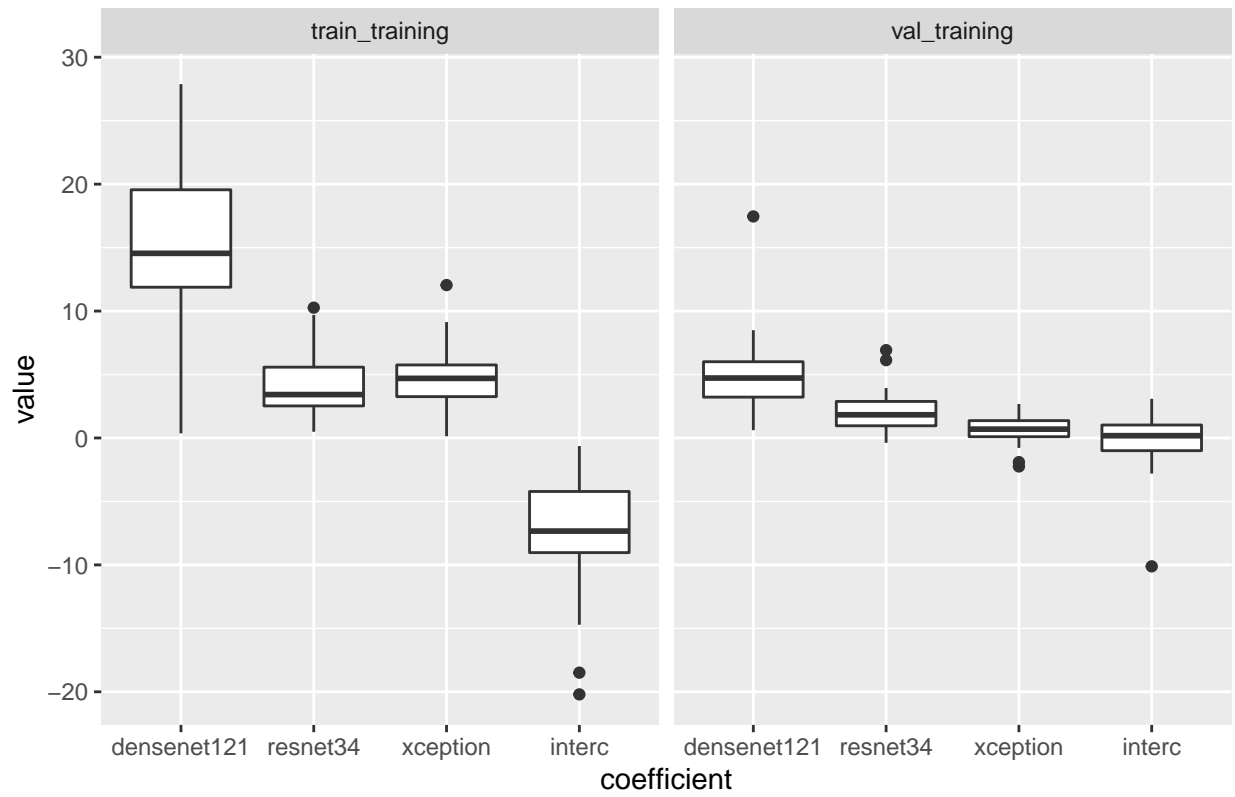
Coefficients for class 6 vs 7



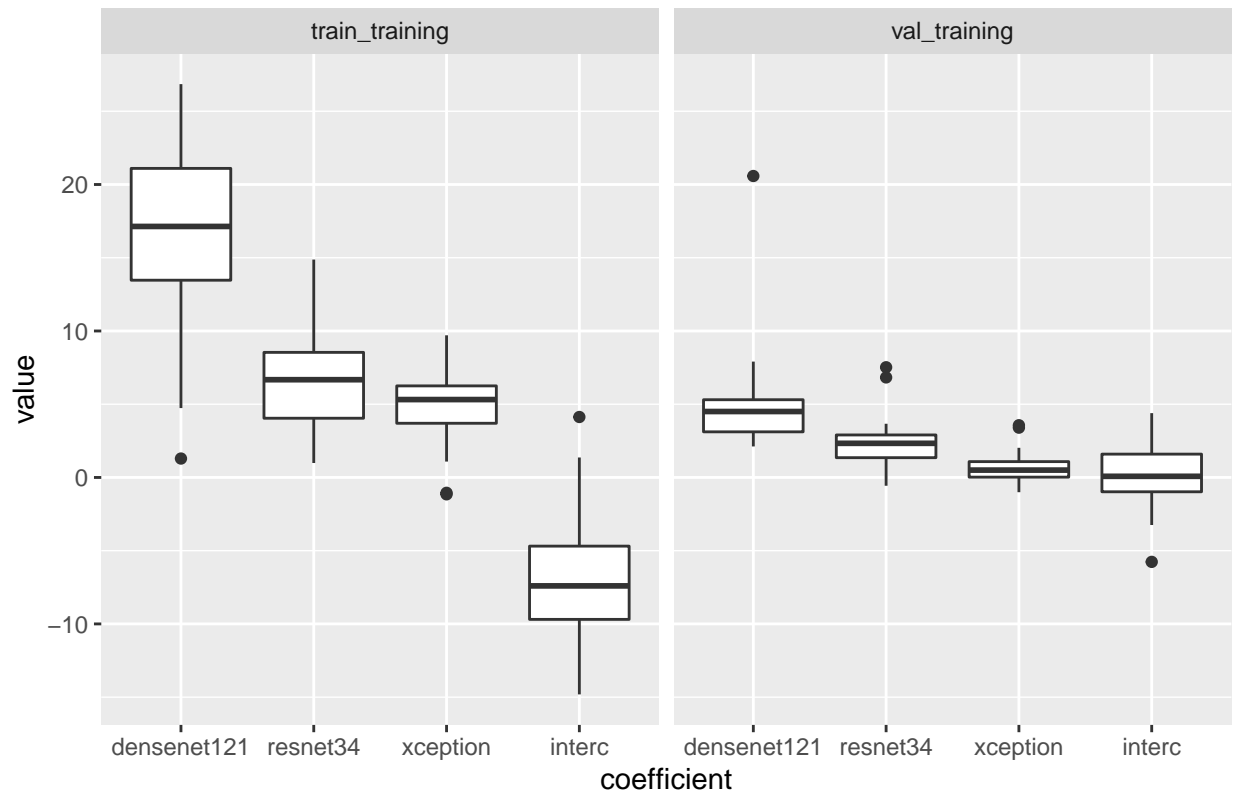
Coefficients for class 6 vs 8



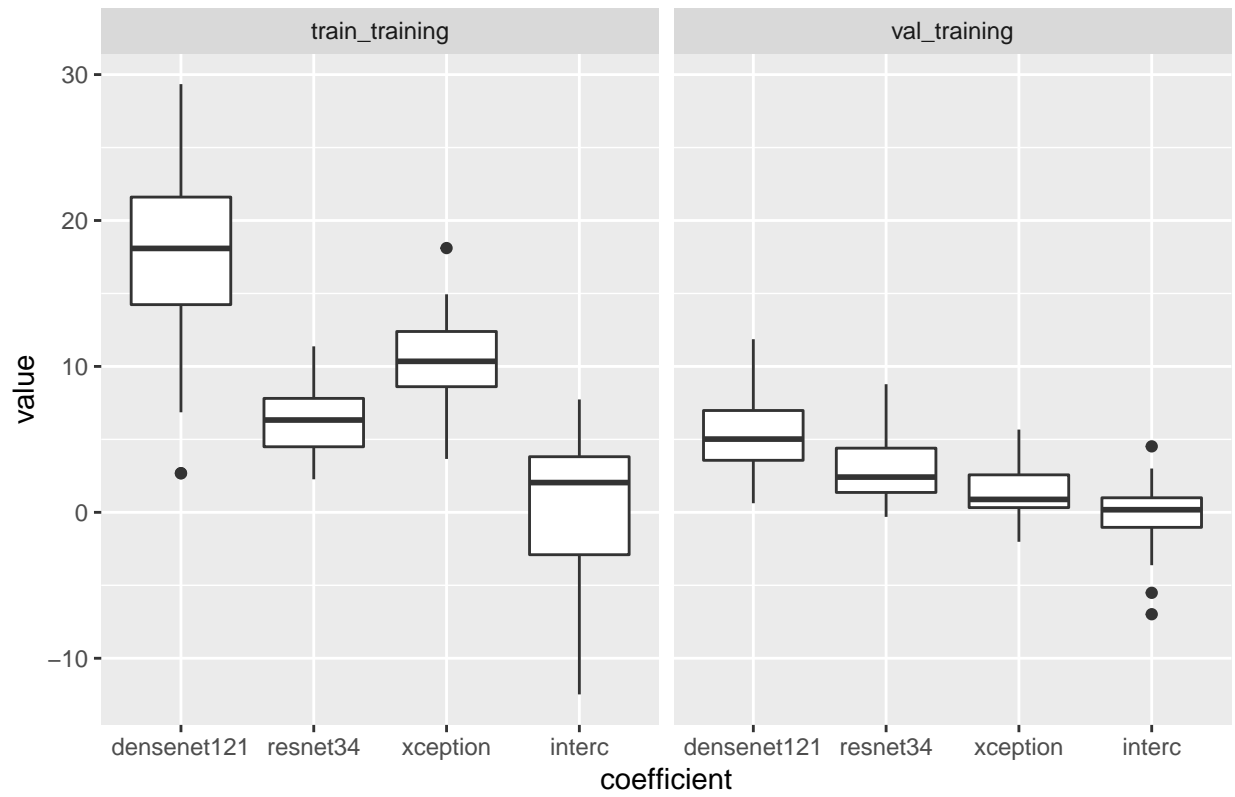
Coefficients for class 6 vs 9



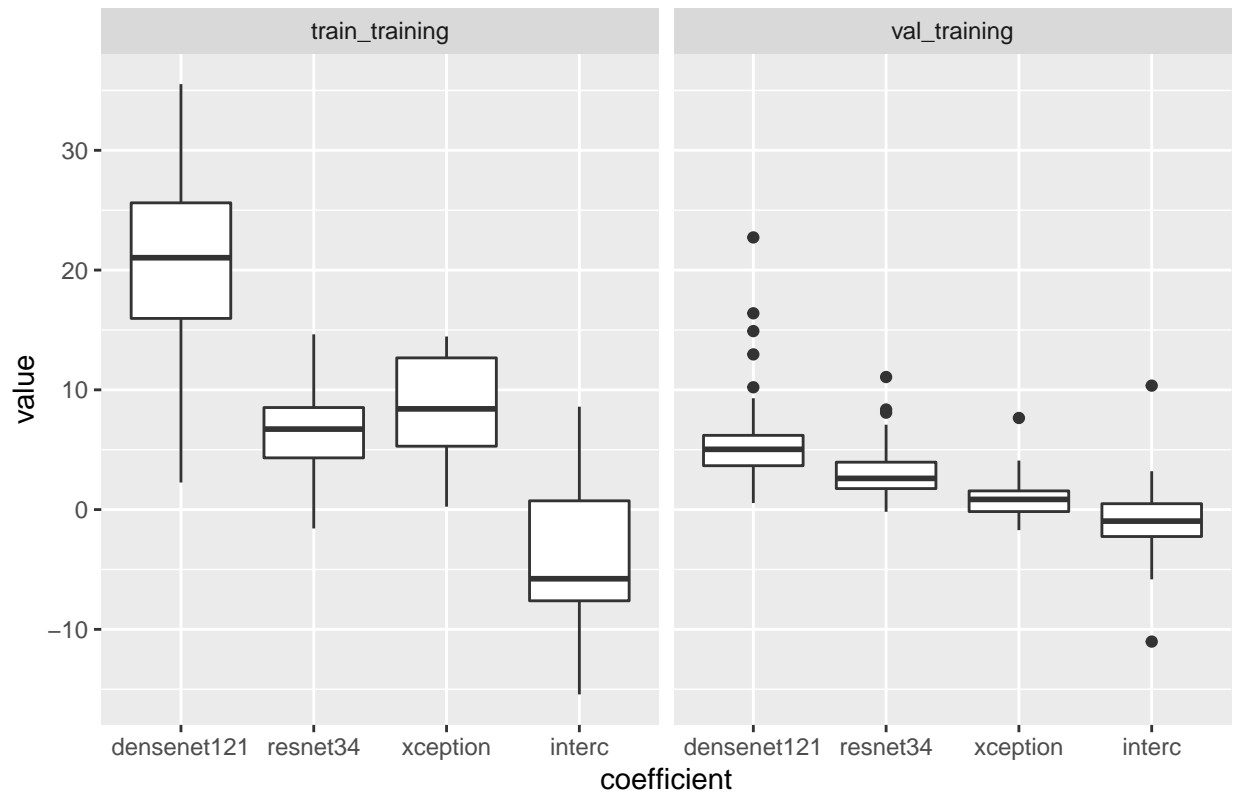
Coefficients for class 6 vs 10



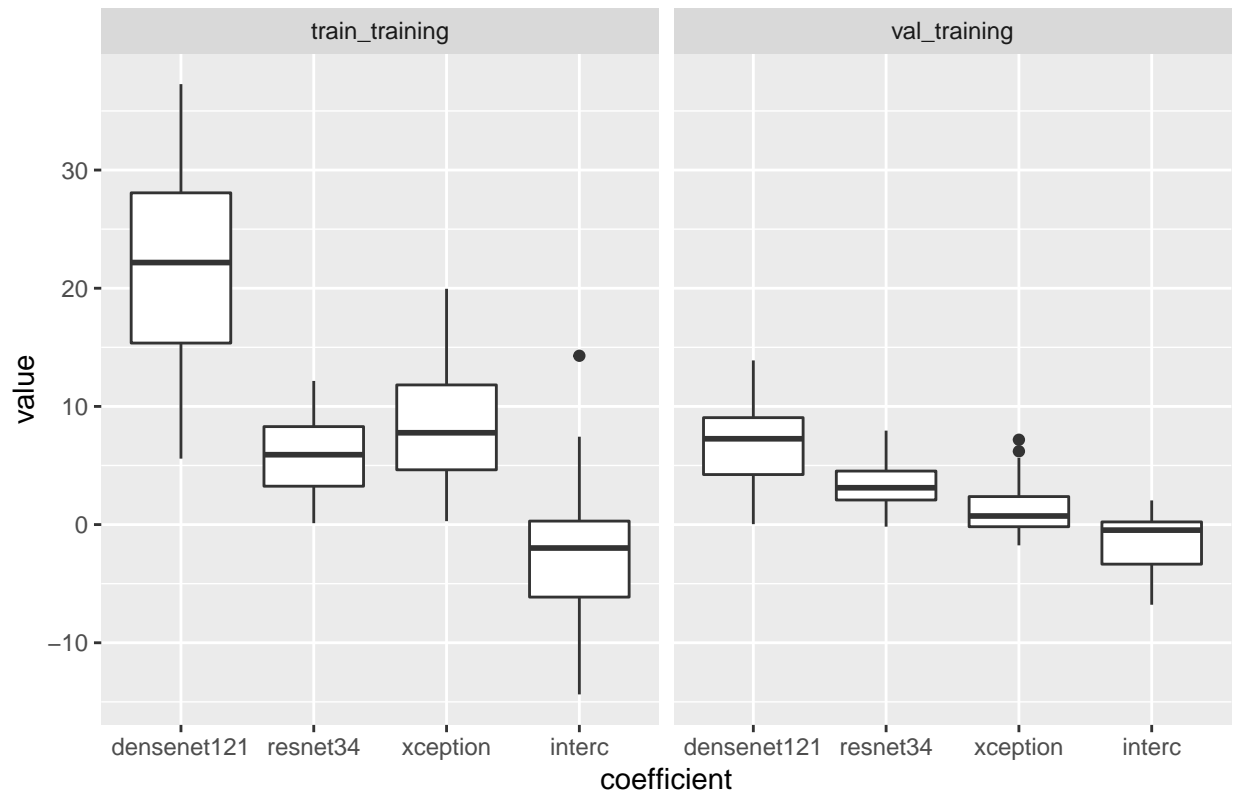
Coefficients for class 7 vs 8



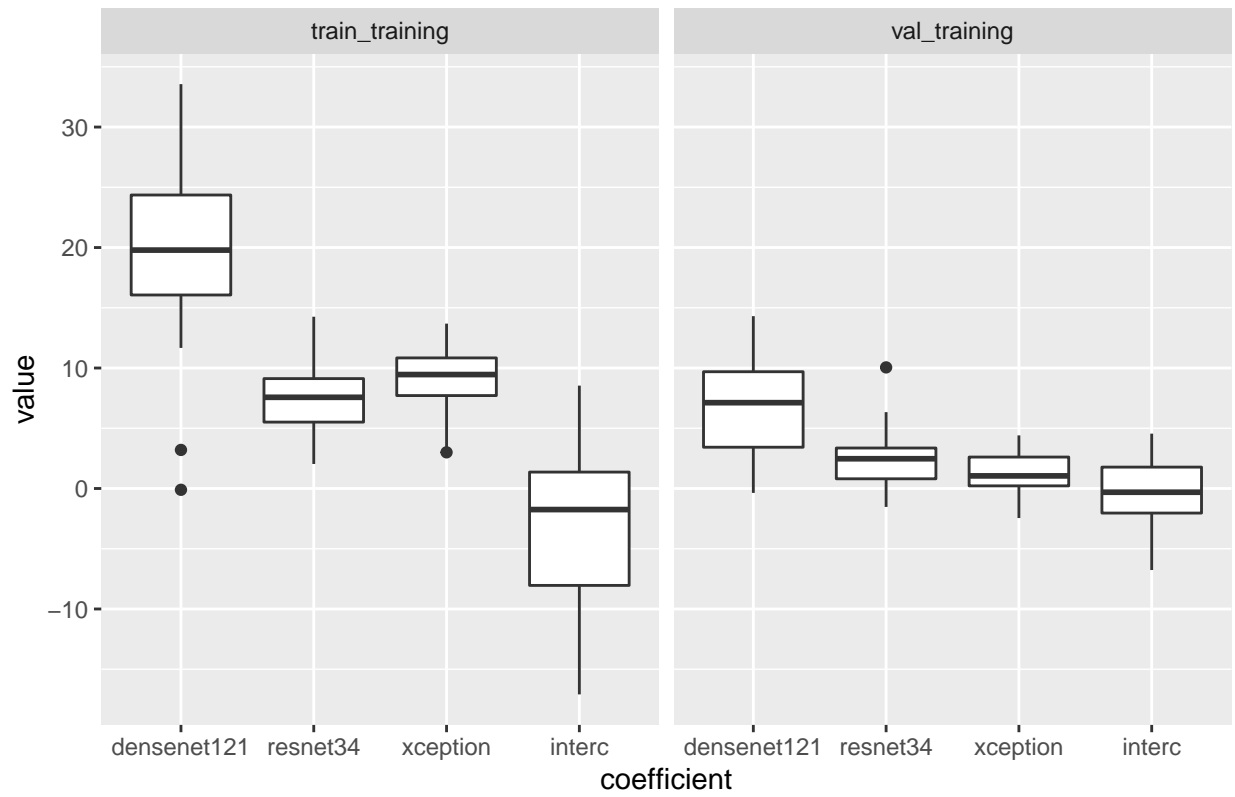
Coefficients for class 7 vs 9



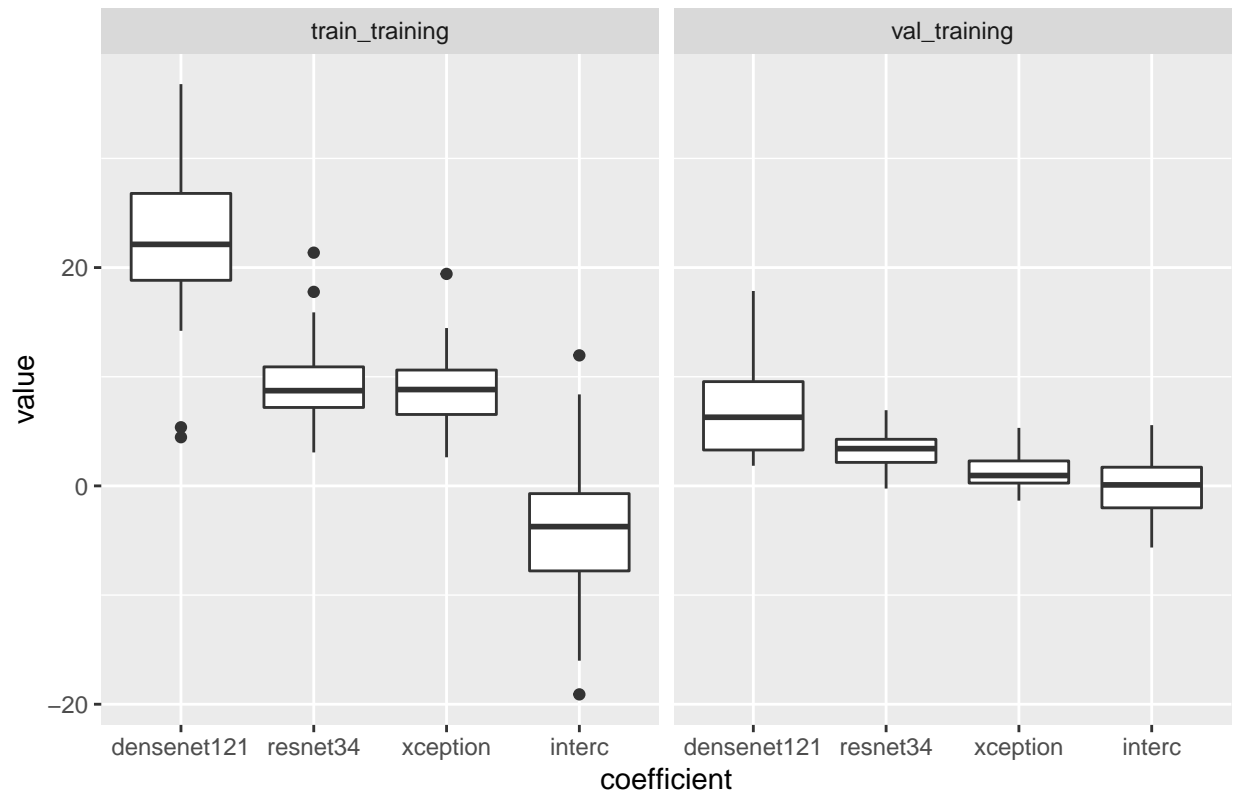
Coefficients for class 7 vs 10



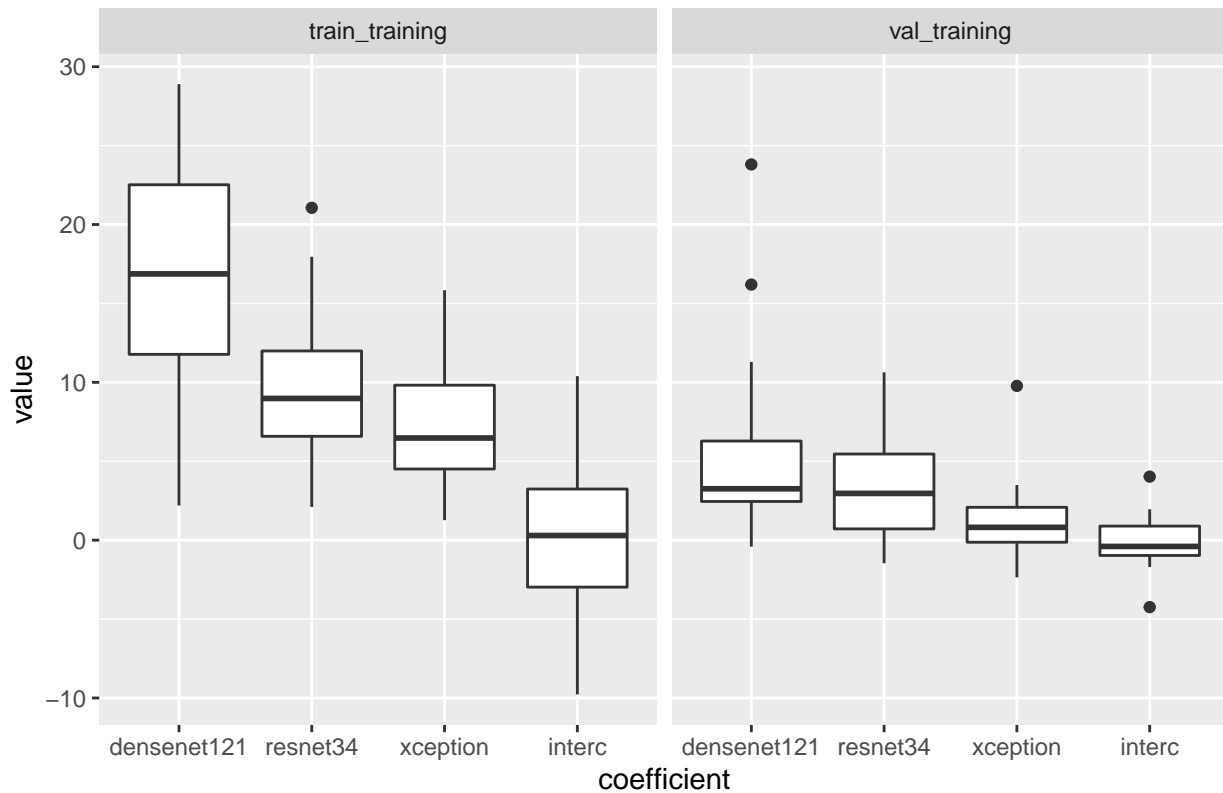
Coefficients for class 8 vs 9



Coefficients for class 8 vs 10



Coefficients for class 9 vs 10



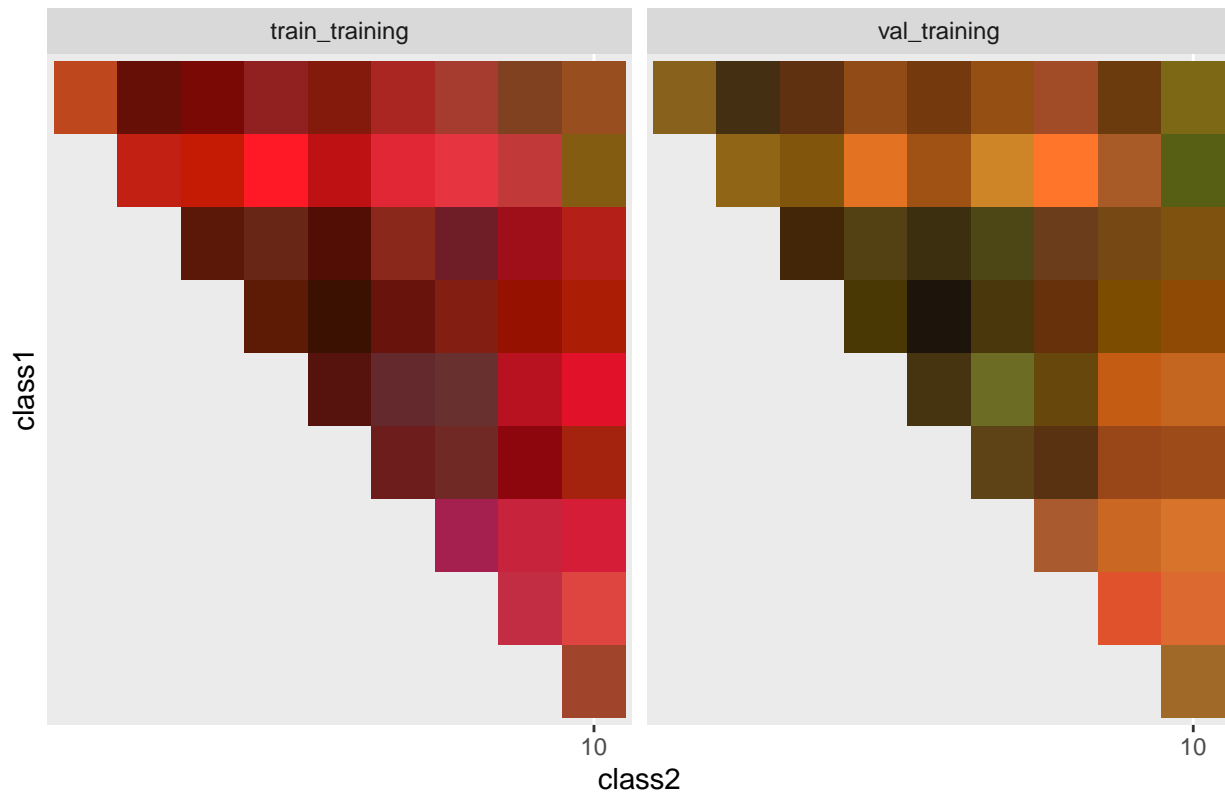
```
avg_lda_coefs <- lda_coefs %>% filter(coefficient != "interc") %>% group_by(class1, class2, precision, train_type)
```

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

```
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[, c("class1", "class2")])])])
```

```
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
```

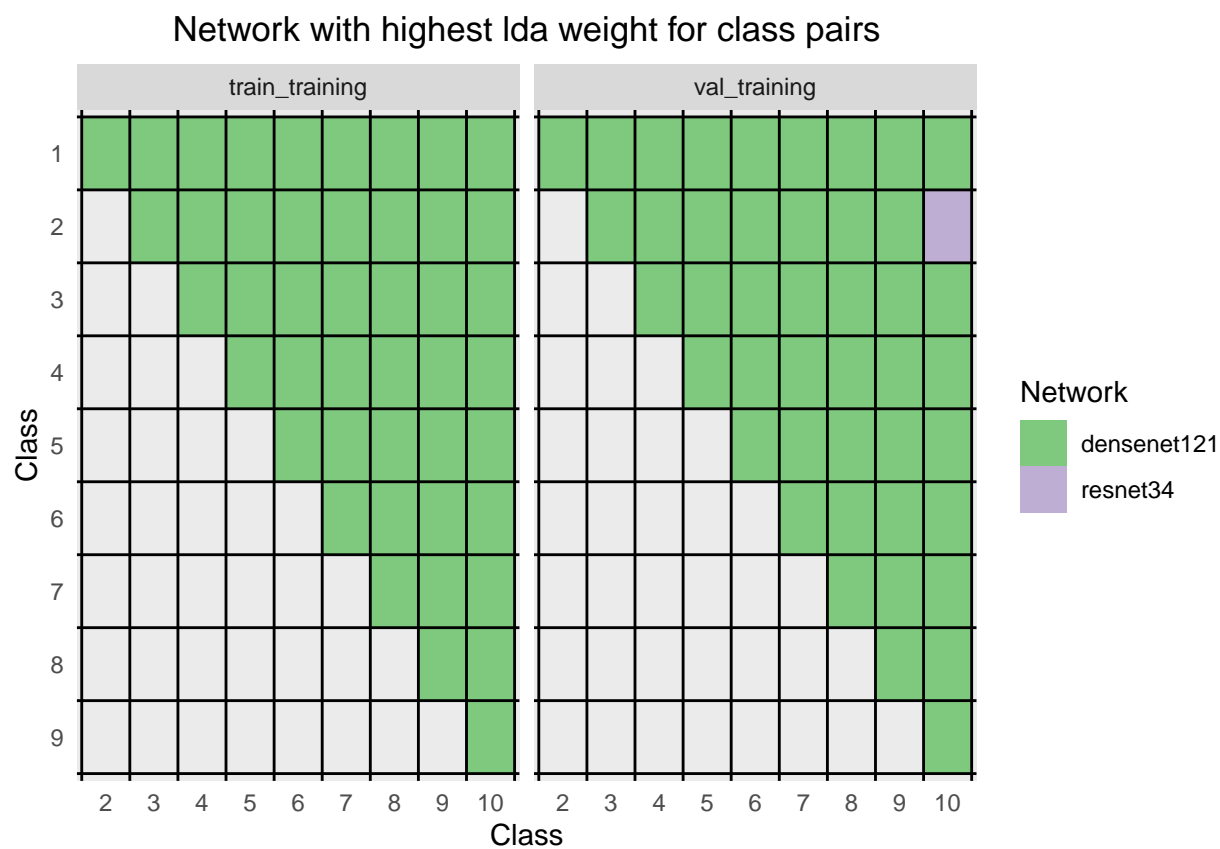
RGB image formed from lda coefficients for networks densenet, resnet, xception



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 clearly dominant for both LDA training methodologies.