Accurate diagnosis of hemoglobinopathies with machine learning based on high-throughput proteomics.

Shaodong Wei

2025-03-07

Read me

This R markdown is to present the code used for generating the figures in the article "Accurate diagnosis of hemoglobinopathies with machine learning based on high-throughput proteomics"

Raw data can not be shared due to GDPR.

fig. 1B

```
library(ggplot2)
library(tidyverse)

# Clear workspace
rm(list = ls())

# Read data
file_path <- './data_in/variants.csv'
df <- read.csv(file_path)

# Update group names
df$group <- recode(df$group, 'BT' = ' -trait', 'CTR' = 'Non-\ncarrier')

# Filter dataset for specific classes
df <- df %>% filter(class %in% c('Development', 'Validation'))

# Calculate sample counts
ct <- df %>%
  select(group, sample, class) %>%
  distinct() %>%
  count(group, class)

# Use custom colors
colors <- readRDS("./data_in/colors.rds")

# Factorize class and group for ordering
ct$class <- factor(ct$class, levels = rev(c('Development', 'Validation')))
ct$group <- factor(ct$group, levels = rev(c('HbC', 'HbD', 'HbE', 'HbS', ' -trait', 'Non-\ncarrier')))
```

```r
# Summarize total sample count by class
sample_summary <- aggregate(n ~ class, data = ct, FUN = sum)

# Create bar plot
ggplot(ct, aes(x = group, y = n, fill = class, color = class)) +
  geom_bar(position = 'stack', stat = 'identity') +
  coord_flip() +
  geom_text(aes(label = n), position = position_stack(vjust = 0.5), color = "white", size = 5) +
  theme_minimal() +
  theme(
    axis.text.x.bottom = element_blank(),
    axis.title = element_blank(),
    text = element_text(size = 15)
  ) +
  scale_fill_manual(
    name = '',
    values = c("Development" = colors[1], "Validation" = colors[2]),
    breaks = c('Development', 'Validation'),
    labels = c('Development\n(n=82)', 'Validation\n(n=45)')
  ) +
  scale_color_manual(
    name = '',
    values = c("Development" = colors[1], "Validation" = colors[2]),
    breaks = c('Development', 'Validation'),
    labels = c('Development\n(n=82)', 'Validation\n(n=45)')
  )
```

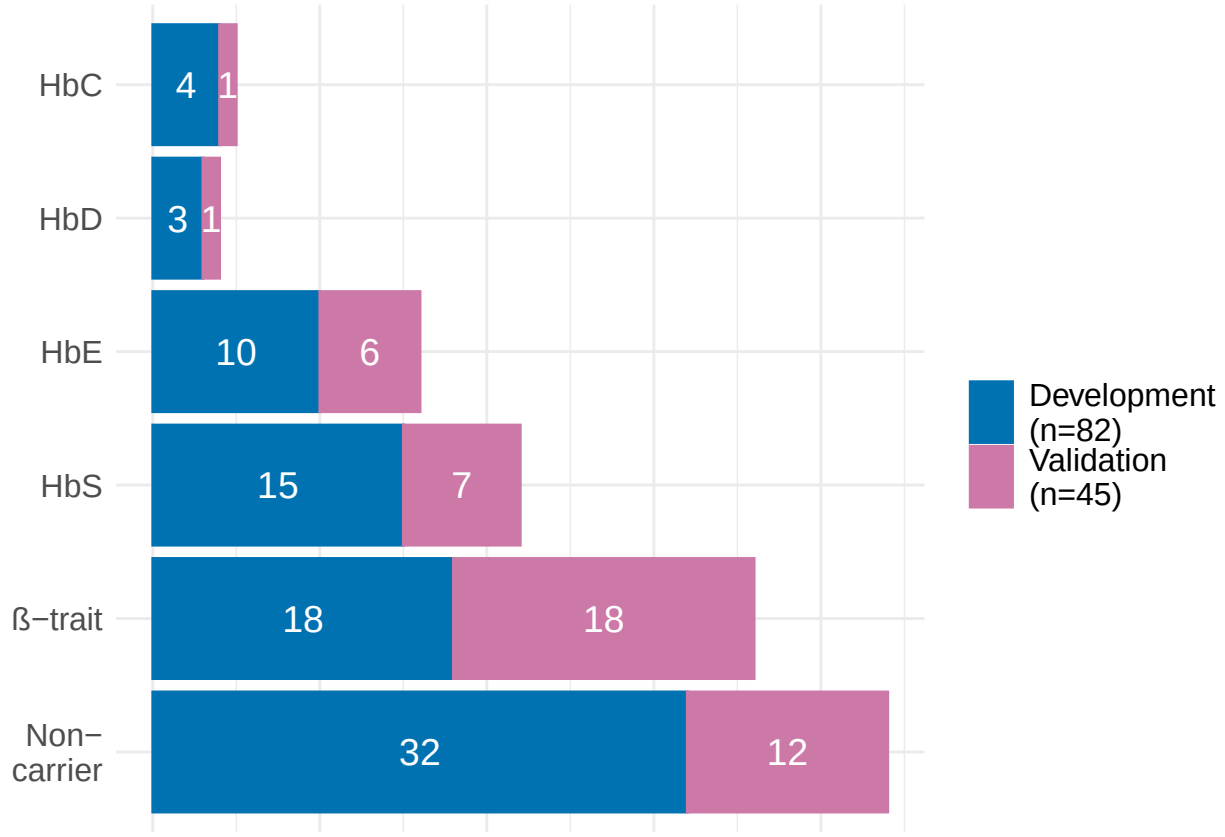


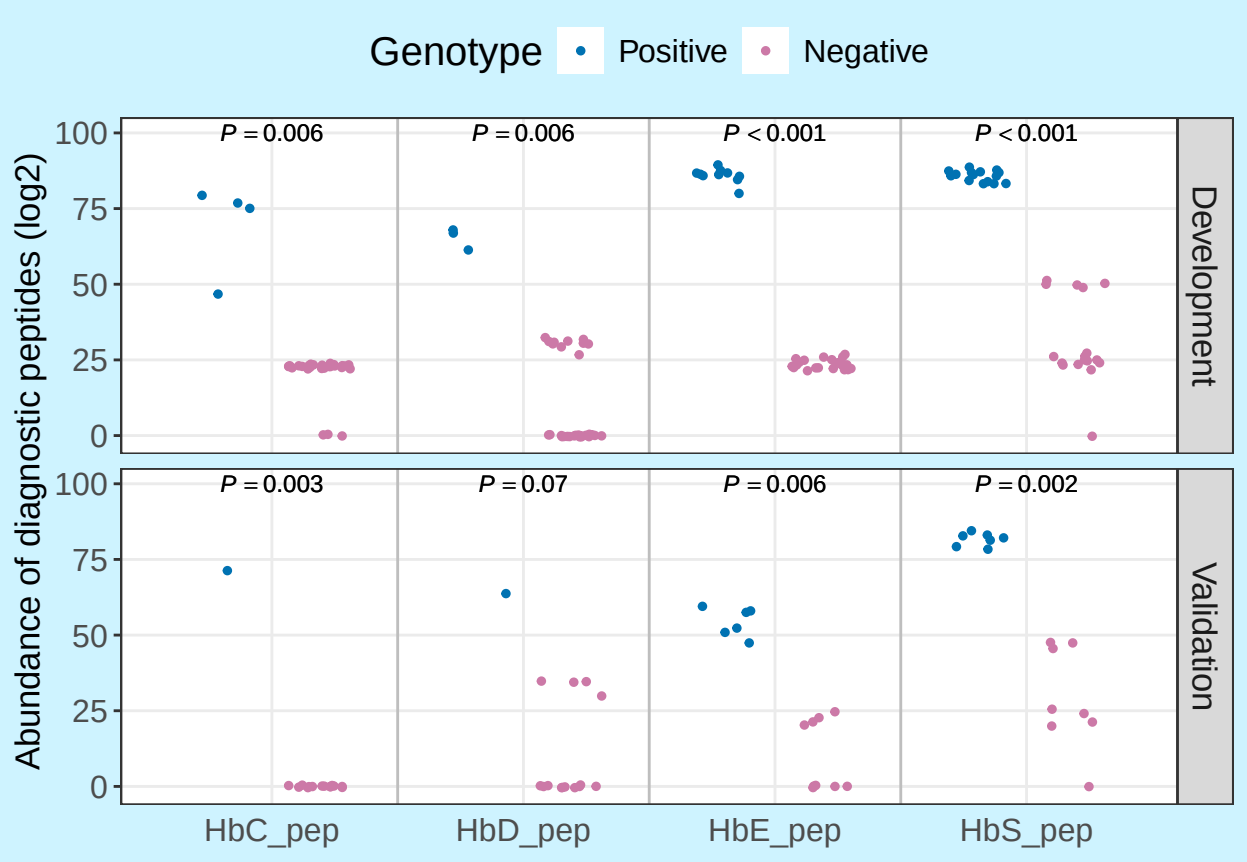
fig. 2A

```r
library(ggplot2)
library(tidyverse)
library(caret)
library(colorspace)
library(ggpubr)

# Clear workspace
rm(list = ls())

# Read data
file_path <- './data_in/variants.csv'
df <- read.csv(file_path)

# Filter out specific groups
df <- df %>% filter(!group %in% c('BT', 'CTR'))

# Rename variants
df <- df %>% mutate(variant = recode(variant,
                                     'HbC' = 'HbC_pep',
                                     'HbD' = 'HbD_pep',
                                     'HbE' = 'HbE_pep',
                                     'HbS' = 'HbS_pep'))
```

```r
# Subset and order groups
df <- df %>% filter(class %in% c('Development', 'Validation'))
df$class <- factor(df$class, levels = c('Development', 'Validation'))
df$group <- factor(df$group, levels = c('HbC', 'HbD', 'HbE', 'HbS'))

# Use custom colors
colors <- readRDS("./data_in/colors.rds")

# Assign xmin and xmax for plotting
df <- df %>%
  group_by(variant) %>%
  mutate(
    xmin = case_when(
      variant == 'HbC_pep' ~ 0.6,
      variant == 'HbD_pep' ~ 1.6,
      variant == 'HbE_pep' ~ 2.6,
      variant == 'HbS_pep' ~ 3.6
    ),
    xmax = case_when(
      variant == 'HbC_pep' ~ 1.4,
      variant == 'HbD_pep' ~ 2.4,
      variant == 'HbE_pep' ~ 3.4,
      variant == 'HbS_pep' ~ 4.4
    )
  )

# Prepare data for comparison
tmp <- df %>%
  mutate(variant2 = sub("_.*", "", variant),
         binary = ifelse(group == variant2, 'Positive', 'Negative'))
tmp$binary <- factor(tmp$binary, levels = c('Positive', 'Negative'))

# Compute p-values
pval <- compare_means(log ~ binary, method = 'wilcox.test', data = tmp, group.by = c('class', 'variant'))
pval$pval <- mapply(function(p) {
  if (p < 0.001) {
    "italic(P) < '0.001'"
  } else if (p < 0.01) {
    sprintf("italic(P) == \"%s\"", formatC(p, format = "f", digits = 3))
  } else {
    sprintf("italic(P) == \"%s\"", formatC(p, format = "f", digits = 2))
  }
}, pval$p.adj)

tmp <- merge(tmp, pval[, c('class', 'variant', 'pval')], by = c('class', 'variant'), all.x = TRUE)


# Generate main plot
f2a <- ggplot(tmp, aes(x = variant, y = log, color = binary)) +
  geom_point(position = position_jitterdodge(dodge.width = 0.75, jitter.width = 0.5,
                                             jitter.height = 0.5), size = 1) +
  facet_grid(class ~ .) +
  scale_color_manual('Genotype', values = colors) +
```

```
  labs(y = 'Abundance of diagnostic peptides (log2)') +
  theme_bw() +
  theme(
    text = element_text(size = 15),
    strip.text = element_text(size = 13),
    axis.title.x = element_blank(),
    panel.grid.minor = element_blank(),
    axis.ticks.x = element_blank(),
    axis.title.y = element_text(size = 13),
    legend.position = 'top',
    plot.background = element_rect(fill = lighten('lightblue', 0.6), color = NA),
    legend.background = element_rect(fill = lighten('lightblue', 0.6), color = NA)
  ) +
  scale_y_continuous(limits = c(-1, 100)) +
  geom_vline(xintercept = c(1.5, 2.5, 3.5), color = 'gray')+
    geom_text(data=unique(tmp[,c('class','variant','pval','binary')]),
              aes(label = pval, y=100),
              color='black',
              size=3,
              parse = TRUE)
```

```
f2a
```




```
# save fig
saveRDS(f2a, file = './data_out/fig.2A.rds')
```

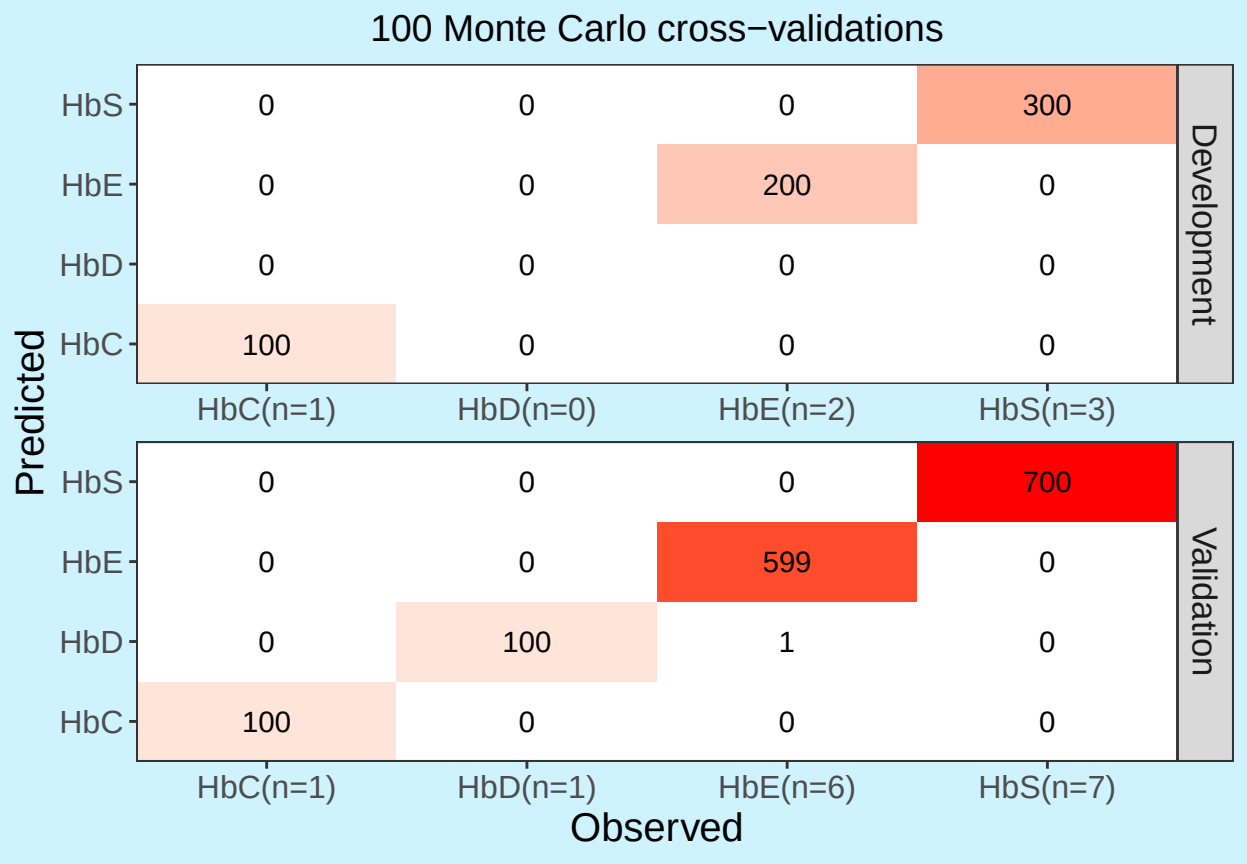
fig. 2B

```r
library(ggplot2)
library(ggthemes)
library(reshape2)
library(pROC)
library(tidyverse)
library(caret)
library(foreach)
library(doParallel)
library(gridExtra)
library(cowplot)
library(mltools)

# Clear workspace
rm(list = ls())

# Read data
file_path <- './data_in/variants.peptides.csv'
df <- read.csv(file_path)

# Filter specific groups
df <- df %>% filter(!group %in% c('BT', 'CTR'))

# Subset data based on class
df.dev <- df %>% filter(class == 'Development')
df.batch <- df %>% filter(class == 'Validation')
df.blood <- df %>% filter(class == 'Whole blood development')
df.plasma <- df %>% filter(class == 'Plasma development')

# Transform data using dcast
dcast_data <- function(df) {
  df %>% dcast(group + sample ~ peptide, value.var = 'value') %>% select(-sample)
}

dev <- dcast_data(df.dev)
dev$group <- factor(dev$group, levels = c('HbC', 'HbD', 'HbE', 'HbS'))

val <- dcast_data(df.batch)
val$group <- factor(val$group, levels = c('HbC', 'HbD', 'HbE', 'HbS'))


# # Check stability of splitting data
# num_cores <- detectCores() - 1  # Use one less than the available cores
# cl <- makeCluster(num_cores)
# registerDoParallel(cl)
#
# results <- foreach(i = 1:100, .combine = rbind, .packages = c("caret", 'tidyverse', 'pROC')) %dopar%
#
#   # Split the data based on the outer fold
#   train_index <- createDataPartition(dev$group, p = 0.75, list = FALSE)
#   trainData <- dev[train_index, ]
#   testData <- dev[-train_index, ]
```

```
#
#   tuneGrid <- expand.grid(mtry = seq(1, 10, 1))  # Adjust mtry values
#   control <- trainControl(method = "repeatedcv", number = 10, repeats = 1)
#
#   # Train the model with inner CV for tuning
#   model <- train(group ~ ., data = trainData,
#                  method = "rf",
#                  tuneGrid = tuneGrid,
#                  trControl = control,
#                  ntree = 5000)
#
#   # Accuracy
#   pred_test <- predict(model, newdata = testData, type = 'raw')
#   acc_value_test <- mean(pred_test == testData$group)
#
#   # AUC
#   probs <- predict(model, newdata = testData, type = 'prob')
#   true_label <- testData$group
#   auc_value_test <- as.numeric(auc(multiclass.roc(true_label, probs)))
#
#   # Save data for test
#   ot_test <- data.frame(pred=pred_test, actual = testData$group, iteration = i,
#                         class = 'Development', acc = acc_value_test,
#                         auc = auc_value_test)
#
#   # Align data
#   vars <- model$trainingData %>%
#     select(-.outcome) %>%
#     colnames()
#   vars <- c(vars, 'group')
#
#   val_tmp <- val
#   missing_cols <- setdiff(vars, colnames(val_tmp))  # Check missing cols
#   val_tmp[missing_cols] <- 0  # Assign missing cols
#   val_tmp <- val_tmp[vars]  # Select and reorder
#
#   # Accuracy on the validation
#   pred_val <- predict(model, newdata = val_tmp, type = 'raw')
#   acc_value_val <- mean(pred_val == val_tmp$group)
#
#   # AUC
#   probs <- predict(model, newdata = val_tmp, type = 'prob')
#   true_label <- val_tmp$group
#   auc_value_val <- as.numeric(auc(multiclass.roc(true_label, probs)))
#
#   # Save data
#   ot_val <- data.frame(pred=pred_val, actual = val_tmp$group, iteration = i,
#                          class = 'Validation', acc = acc_value_val,
#                          auc = auc_value_val)
#
#   # Merge data
#   tmp <- rbind(ot_test, ot_val)
#   return(tmp)
```

```r
# }
#
# stopCluster(cl)
#
# #saveRDS(results, file = './data_out/fig.2B.data.rds')

# Load precomputed results
results <- readRDS('./data_out/fig.2B.data.rds')

# Count predictions
freq <- results %>% group_by(class) %>% count(class, pred, actual)

# Create a base table
categories <- c("HbC", "HbD", "HbE", "HbS")
classes <- c('Development', 'Validation')
tab <- expand.grid(class = classes, pred = categories, actual = categories)
tab <- merge(tab, freq, by = c('class', 'pred', 'actual'), all = TRUE)
tab$n[is.na(tab$n)] <- 0

# Rename x-axis labels
tab$actual <- as.character(tab$actual)
tab <- tab %>% mutate(
  actual = case_when(
    actual == 'HbC' & class == 'Development' ~ 'HbC(n=1)',
    actual == 'HbD' & class == 'Development' ~ 'HbD(n=0)',
    actual == 'HbE' & class == 'Development' ~ 'HbE(n=2)',
    actual == 'HbS' & class == 'Development' ~ 'HbS(n=3)',
    actual == 'HbC' & class == 'Validation' ~ 'HbC(n=1)',
    actual == 'HbD' & class == 'Validation' ~ 'HbD(n=1)',
    actual == 'HbE' & class == 'Validation' ~ 'HbE(n=6)',
    actual == 'HbS' & class == 'Validation' ~ 'HbS(n=7)',
    TRUE ~ actual
  )
)

# Generate confusion matrix plot
f2b <- ggplot(tab, aes(actual, pred, fill = n)) +
  geom_tile() +
  geom_text(aes(label = n)) +
  facet_wrap(class ~ ., scales = 'free', nrow = 2, strip.position = "right") +
  scale_fill_gradient(low = 'white', high = 'red') +
  theme_bw() +
  theme(
    text = element_text(size = 15),
    strip.text = element_text(size = 13),
    legend.position = 'None',
    plot.background = element_rect(fill = lighten('lightblue', 0.6), color = NA),
    plot.title = element_text(hjust = 0.5, size = 14)
  ) +
  labs(x = 'Observed', y = 'Predicted') +
  ggtitle('100 Monte Carlo cross-validations') +
  scale_x_discrete(expand = c(0, 0)) +
  scale_y_discrete(expand = c(0, 0))
```

```r
# Compute classification metrics
evaluate_model <- function(df) {
  conf <- confusionMatrix(factor(df$pred), factor(df$actual))
  acc <- format(round(conf$overall['Accuracy'], 3), nsmall = 3)
  sens <- format(round(mean(conf$byClass[,"Sensitivity"]), 3), nsmall = 3)
  spec <- format(round(mean(conf$byClass[,"Specificity"]), 3), nsmall = 3)
  f1 <- format(round(mean(conf$byClass[,"F1"]), 3), nsmall = 3)
  mcc <- format(round(mltools::mcc(preds = df$pred, actuals = df$actual), 3), nsmall = 3)
  auc <- format(round(mean(df$auc), 3), nsmall = 3)
  data.frame(Metrics = c(paste('AUC=', auc), paste('Accuracy=', acc),
                         paste('Sensitivity=', sens), paste('Specificity=', spec),
                         paste('F1=', f1), paste('MCC=', mcc)))
}

tab.dev <- evaluate_model(results %>% filter(class == 'Development'))
tab.val <- evaluate_model(results %>% filter(class == 'Validation'))

# Generate tables
table_plot <- function(tab) {
  tableGrob(tab, rows = NULL, theme = ttheme_default(
    core = list(fg_params = list(cex = 0.9, hjust = 0, x = 0.15),
                padding = unit(c(10, 2), "mm"),
                bg_params = list(fill = NA)),
    colhead = list(bg_params = list(fill = NA))
  ))
}

f2b_table <- ggdraw() +
  draw_plot(table_plot(tab.dev), x = 0, y = 0.48, height = 0.5) +
  draw_plot(table_plot(tab.val), x = 0, y = 0.06, height = 0.5)

print(f2b)
```




```
# save fig
saveRDS(f2b, file = './data_out/fig.2B.rds')

# save table
saveRDS(f2b_table, file = './data_out/fig.2B.table.rds')

# save table background
f2b_background <- ggplot() +
  annotate("rect", xmin = 0, xmax = 1, ymin = 0, ymax = 1,
           fill = lighten("lightblue", 0.6), color = NA) +  # Background rectangle
  theme_void()  # Remove axes and gridlines
saveRDS(f2b_background, file = './data_out/fig.2B.table.background.rds')
```

fig. 2C

```
rm(list = ls())

library(tidyverse)
library(caret)
library(randomForest)
library(reshape2)
library(pROC)
library(foreach)
library(doParallel)
```

```r
library(ggplot2)
library(colorspace)

# Load dataset
df <- read.csv('./data_in/thalassemia.csv')

# Subset dataset by class
df_dev <- df %>% filter(class == 'Development')
df_val <- df %>% filter(class == 'Validation')


# Transform data to wide format
dev <- dcast(group + sample ~ peptide, data = df_dev, value.var = 'log') %>% select(-sample)
dev$group <- factor(dev$group, levels = c('CTR', 'BT'))

val <- dcast(group + sample ~ peptide, data = df_val, value.var = 'log') %>% select(-sample)
val$group <- factor(val$group, levels = c('CTR', 'BT'))


# # check stability of splitting data
# library(foreach)
# library(doParallel)
#
# #run in parallel
# num_cores <- detectCores() - 1  # Use one less than the available cores
# cl <- makeCluster(num_cores)
# registerDoParallel(cl)
#
# # Paralleled nested cross-validation
# results <- foreach(i = 1:100, .combine = rbind, .packages = c("caret","pROC","tidyverse","reshape2")) %dopar% {
#   tryCatch({
#
#     # Split the data based on the outer fold
#     train_index <- createDataPartition(dev$group, p = 0.75, list = FALSE)
#     trainData <- dev[train_index, ]
#     testData <- dev[-train_index, ]
#
#     tuneGrid <- expand.grid(mtry = seq(2,10,1))
#     control <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
#
#     # Train the model with inner CV for tuning
#     model <- train(group ~ ., data = trainData,
#                    method = "rf",
#                    tuneGrid = tuneGrid,
#                    trControl = control,
#                    ntree = 5000)
#
#     # auc
#     pred_prob_test <- predict(model, testData, type = "prob")
#     roc_test <- roc(testData$group, pred_prob_test[, "BT"])
#     auc_test <- as.numeric(auc(roc_test))
#
#     # accuracy
```

```
#     pred_raw_test <- predict(model, testData, type = 'raw')
#     acc_default_test <- mean(testData$group == pred_raw_test)
#
#     # model calibration using Platt method
#     rf_probs_train <- predict(model, trainData, type = "prob")[, 2]
#     platt_model <- glm(trainData$group ~ rf_probs_train, family = binomial(link = "logit"))
#     rf_probs_test <- predict(model, testData, type = "prob")[, 2]
#     calibrated_probs <- predict(platt_model, newdata = data.frame(rf_probs_train = rf_probs_test), ty
#     threshold <- 0.5
#     pred_calibration_test <- ifelse(calibrated_probs >= threshold, 'BT', 'CTR')
#     acc_calibration_test <- mean(pred_calibration_test == testData$group)
#
#     # f1 score on test
#     f1_macro_test <- yardstick::f_meas_vec(truth=testData$group, estimate = pred_raw_test, estimator=
#     f1_micro_test <- yardstick::f_meas_vec(truth=testData$group, estimate = pred_raw_test, estimator=
#
#     # Collect data
#     df_roc_test <- data.frame(specificity=roc_test$specificities,
#                               sensitivity=roc_test$sensitivities,
#                               iteration=i,
#                               auc=auc_test,
#                               acc_default=acc_default_test,
#                               acc_calibration=acc_calibration_test,
#                               class='Development',
#                               type='roc',
#                               f1_macro=f1_macro_test,
#                               f1_micro=f1_micro_test,
#                               prob=NA,
#                               pred=NA,
#                               actual=NA)
#     df_prob_test <- data.frame(specificity=NA,
#                                sensitivity=NA,
#                                iteration=i,
#                                auc=auc_test,
#                                acc_default=acc_default_test,
#                                acc_calibration=acc_calibration_test,
#                                class='Development',
#                                type='prob',
#                                f1_macro=f1_macro_test,
#                                f1_micro=f1_micro_test,
#                                prob=pred_prob_test[,2],
#                                pred=as.character(pred_calibration_test),
#                                actual=as.character(testData$group))
#
#
#     # Align data
#     var_model <- model$trainingData %>%
#       select(-.outcome) %>%
#       colnames()
#     var_model <- c(var_model, 'group')
#
#     val_tmp <- val
#     missing_cols <- setdiff(var_model, colnames(val)) # find which variable is missing in data
```

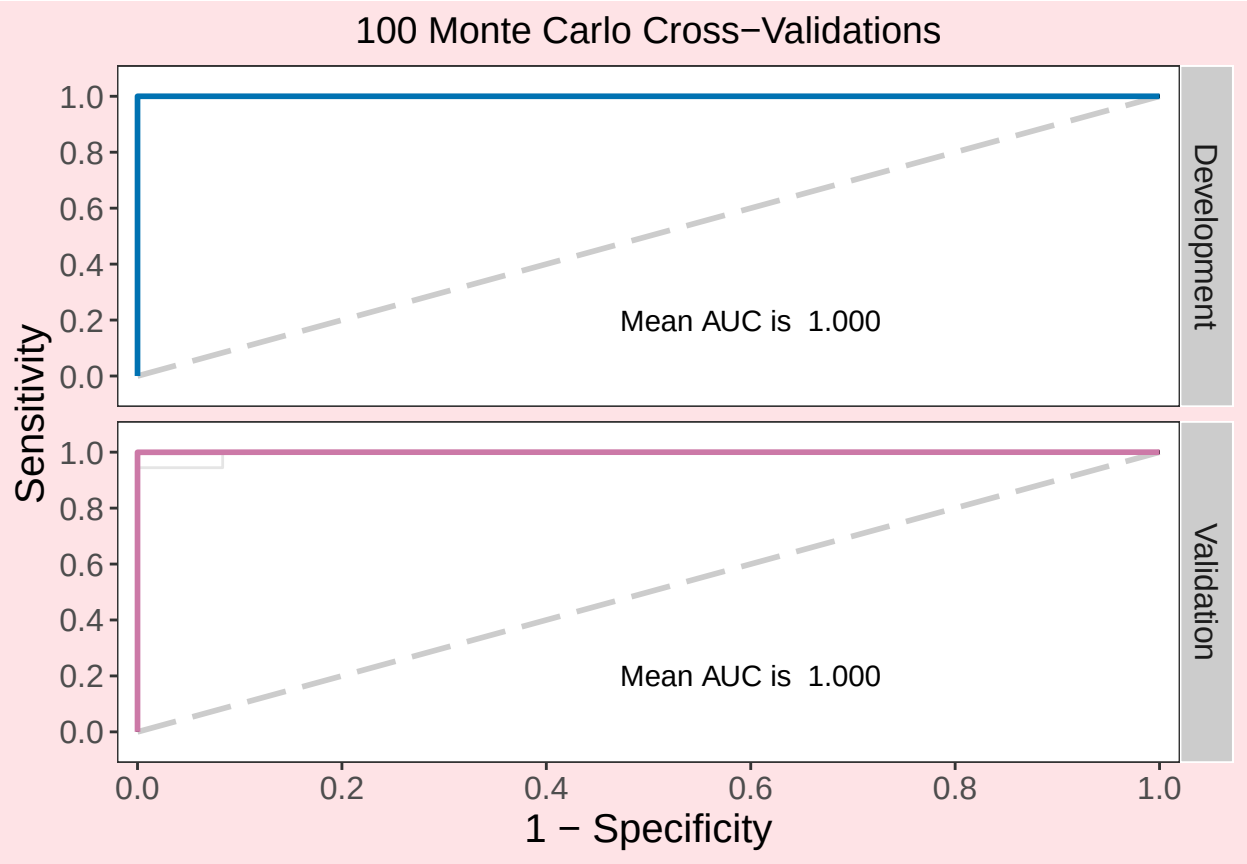
```
#     val_tmp[missing_cols] <- 0 # assign missing columns
#     val_tmp <- val_tmp[var_model] # select columns and reorder
#
#     # auc on validation
#     pred_val <- predict(model, val_tmp, type = "prob")
#     roc_val_batch <- roc(val_tmp$group, pred_val[, "BT"], levels=c('CTR','BT') )
#     auc_val_batch <- as.numeric(roc_val_batch$auc)
#
#     # Default predictions and acc
#     pred_raw_val <- predict(model, val_tmp, type = 'raw')
#     acc_default_val <- mean(val_tmp$group == pred_raw_val)
#
#     # f1 score
#     f1_macro_val <- yardstick::f_meas_vec(truth=val_tmp$group, estimate = pred_raw_val, estimator='ma
#     f1_micro_val <- yardstick::f_meas_vec(truth=val_tmp$group, estimate = pred_raw_val, estimator='mi
#
#     # Model calibration for accuracy
#     rf_probs_train <- predict(model, trainData, type = "prob")[, 2]
#     platt_model <- glm(trainData$group ~ rf_probs_train, family = binomial(link = "logit"))
#     rf_probs_batch <- predict(model, val_tmp, type = "prob")[, 2]
#     calibrated_probs <- predict(platt_model, newdata = data.frame(rf_probs_train = rf_probs_batch), t
#     threshold <- 0.5
#     pred_calibration_batch <- ifelse(calibrated_probs >= threshold, 'BT', 'CTR')
#     acc_calibration_batch <- mean(pred_calibration_batch == val_tmp$group)
#
#     # Save data
#     df_roc_batch <- data.frame(specificity=roc_val_batch$specificities,
#                                sensitivity=roc_val_batch$sensitivities,
#                                iteration=i,
#                                auc=auc_val_batch,
#                                acc_default=acc_default_val,
#                                acc_calibration=acc_calibration_batch,
#                                class='RBC validation',
#                                type='roc',
#                                f1_macro=f1_macro_val,
#                                f1_micro=f1_micro_val,
#                                prob=NA,
#                                pred=NA,
#                                actual=NA)
#     df_prob_batch <- data.frame(specificity=NA,
#                                 sensitivity=NA,
#                                 iteration=i,
#                                 auc=auc_val_batch,
#                                 acc_default=acc_default_val,
#                                 acc_calibration=acc_calibration_batch,
#                                 class='RBC validation',
#                                 type='prob',
#                                 f1_macro=f1_macro_val,
#                                 f1_micro=f1_micro_val,
#                                 prob=pred_val[,2],
#                                 pred=as.character(pred_calibration_batch),
#                                 actual=as.character(val_tmp$group))
#
```

```
#     # Merge data
#     tmp <- NULL
#     tmp <- rbind(df_roc_test,df_prob_test,
#                  df_roc_batch,df_prob_batch
#     )
#     return(tmp)
#
#   }, error=function(e){
#     NULL
#   })
# }
#
# # Stop the parallel cluster
# stopCluster(cl)
#
# # save data
# saveRDS(results, file = './data_out/fig.2CD.data.rds')


## fig. 2C, plot auc

# Load results
results <- readRDS('./data_out/fig.2CD.data.rds')
results$specificity <- 1 - results$specificity
results$class[results$class == 'RBC validation'] <- 'Validation'

# Subset for ROC type
results <- results %>% filter(type == 'roc')

# Define common FPR values for interpolation
common_fpr <- seq(0, 1, length.out = 20)

# Ensure specificity is sorted before computing FPR
results <- results %>%
  mutate(fpr = specificity) %>%
  arrange(iteration, fpr)

# Compute max sensitivity at FPR = 0
max_sens_at_zero <- results %>%
  filter(fpr == 0) %>%
  group_by(iteration, class) %>%
  summarise(max_sens = max(sensitivity, na.rm = TRUE), .groups = "drop")

# Replace sensitivity when specificity is 0
results_fpr_0 <- results %>%
  left_join(max_sens_at_zero, by = c('iteration', 'class')) %>%
  mutate(sensitivity = ifelse(fpr == 0, max_sens, sensitivity)) %>%
  filter(fpr == 0) %>%
  select(-max_sens) %>%
  unique()

results_modified <- results %>%
  filter(fpr != 0) %>%
```

```
  bind_rows(results_fpr_0) %>%
  arrange(class, iteration, fpr)

# Perform interpolation
interp_df <- results_modified %>%
  group_by(iteration, class) %>%
  reframe(
    fpr = common_fpr,
    sens = approx(x = specificity, y = sensitivity, xout = common_fpr, rule = 1, ties = mean)$y
  )

# Compute mean and confidence intervals
summary_df <- interp_df %>%
  group_by(fpr, class) %>%
  summarise(
    mean_sens = mean(sens, na.rm = TRUE),
    .groups = "drop"
  )

# Add (0,0) starting point
start <- data.frame(
  fpr = c(0, 0),
  class = c('Development', 'Validation'),
  mean_sens = c(0, 0)
)
summary_df <- bind_rows(start, summary_df)

# Order specificity and sensitivity
results <- results %>% arrange(fpr, sensitivity)

# Compute mean AUC for each class
auc_summary <- results %>%
  select(iteration, class, auc) %>%
  distinct() %>%
  group_by(class) %>%
  summarise(mean_auc = mean(auc), .groups = "drop") %>%
  mutate(format = format(mean_auc, digits = 3, nsmall = 3))

# Import colors
colors <- readRDS('./data_in/colors.rds')

# Create a diagonal reference line
diagonal_df <- summary_df %>%
  group_by(class) %>%
  summarise(fpr_start = 0, sens_start = 0, fpr_end = 1, sens_end = 1, .groups = "drop")

# Plot ROC Curves
f2c <- ggplot() +
  geom_segment(data = diagonal_df,
               aes(x = fpr_start, y = sens_start, xend = fpr_end, yend = sens_end),
               linetype = "longdash", linewidth = 1, color = "gray80") +
  geom_step(data = results,
            aes(x = fpr, y = sensitivity, group = interaction(iteration, class)),
```

```
            alpha = 0.1, size = 0.5) +
  geom_line(data = summary_df,
            aes(x = fpr, y = mean_sens, color = class),
            size = 1) +
  scale_x_continuous(breaks = seq(0, 1, 0.2), expand = c(0.01, 0.01)) +
  scale_y_continuous(breaks = seq(0, 1, 0.2), expand = c(0.01, 0.1)) +
  theme_bw() +
  theme(
    panel.grid = element_blank(),
    text = element_text(size = 15),
    axis.ticks = element_line(),
    strip.background = element_rect(fill = "gray80", color = 'white'),
    legend.position = 'None',
    plot.title = element_text(hjust = 0.5, size = 14),
    plot.background = element_rect(fill = lighten('lightpink', 0.6), color = NA),
    legend.background = element_rect(fill = lighten('lightpink', 0.6), color = NA)
  ) +
  facet_grid(class ~ .) +
  scale_color_manual('Group', values = colors) +
  scale_fill_manual('Group', values = colors) +
  labs(x = '1 - Specificity', y = 'Sensitivity', title = "100 Monte Carlo Cross-Validations") +
  geom_text(data = auc_summary, aes(x = 0.6, y = 0.2, label = paste("Mean AUC is ", format)))
```

f2c

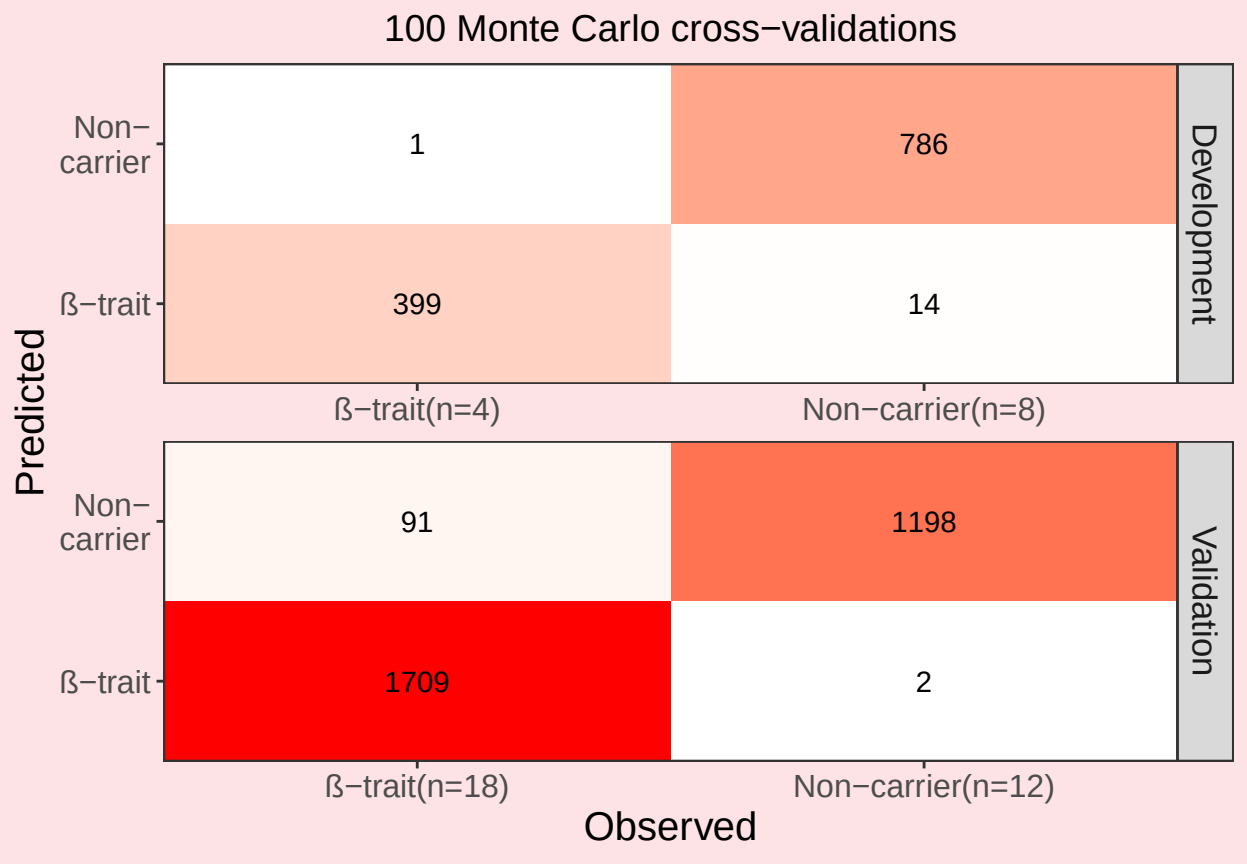


```r
# save fig
saveRDS(f2c, file = "./data_out/fig.2C.rds")
```

fig. 2D

```r
library(ggplot2)
library(tidyverse)
library(colorspace)
library(caret)
library(gridExtra)
library(cowplot)

# Load results
results <- readRDS('./data_out/fig.2CD.data.rds')
results$specificity <- 1 - results$specificity
results$class[results$class == 'RBC validation'] <- 'Validation'

# Adjust prediction and actual labels
results$pred <- recode(results$pred, 'BT' = ' -trait', 'CTR' = 'Non-\ncarrier')
results$actual <- recode(results$actual, 'BT' = ' -trait', 'CTR' = 'Non-\ncarrier')

# Compute frequency counts
freq <- results %>%
  filter(type == 'prob') %>%
  count(class, pred, actual)

# Create base table with all combinations
categories <- c(' -trait', 'Non-\ncarrier')
classes <- c('Development', 'Validation')
tab <- expand.grid(class = classes, pred = categories, actual = categories)
tab <- merge(tab, freq, by = c('class', 'pred', 'actual'), all = TRUE)
tab$n[is.na(tab$n)] <- 0

# Rename x-axis labels
label_map <- list(
  'Development' = c(' -trait' = ' -trait(n=4)', 'Non-\ncarrier' = 'Non-carrier(n=8)'),
  'Validation' = c(' -trait' = ' -trait(n=18)', 'Non-\ncarrier' = 'Non-carrier(n=12)')
)
tab$actual <- mapply(function(cls, val) label_map[[cls]][[val]], tab$class, tab$actual)
tab$actual <- factor(tab$actual, levels = c(" -trait(n=4)", " -trait(n=18)", "Non-carrier(n=8)", "Non-ca

# Plot confusion matrix
f2d <- ggplot(tab, aes(actual, pred, fill = n)) +
  geom_tile() +
  geom_text(aes(label = n)) +
  facet_wrap(class ~ ., scales = 'free', nrow = 2, strip.position = "right") +
  scale_fill_gradient(low = 'white', high = 'red') +
  theme_bw() +
  theme(
    text = element_text(size = 15),
    strip.text = element_text(size = 13),
```

```
    legend.position = 'None',
    plot.background = element_rect(fill = lighten('lightpink', 0.6), color = NA),
    plot.title = element_text(hjust = 0.5, size = 14)
  ) +
  labs(x = 'Observed', y = 'Predicted', title = "100 Monte Carlo cross-validations") +
  scale_x_discrete(expand = c(0, 0)) +
  scale_y_discrete(expand = c(0, 0))

print(f2d)
```




```
# save figure
saveRDS(f2d, file = './data_out/fig.2D.rds')


# Compute classification metrics
compute_metrics <- function(df) {
  conf <- confusionMatrix(factor(df$pred), factor(df$actual), positive = ' -trait')
  acc <- sprintf('Accuracy=%.3f', conf$overall['Accuracy'])
  sensitivity <- sprintf('Sensitivity=%.3f', conf$byClass["Sensitivity"])
  specificity <- sprintf('Specificity=%.3f', conf$byClass["Specificity"])
  f1_score <- sprintf('F1=%.3f', conf$byClass["F1"])
  mcc <- sprintf('MCC=%.3f', mltools::mcc(preds = df$pred, actuals = df$actual))
  data.frame(Metrics = c(acc, sensitivity, specificity, f1_score, mcc))
}
```

```
tab_dev <- compute_metrics(filter(results, class == 'Development'))
tab_val <- compute_metrics(filter(results, class == 'Validation'))

# Generate table plots
theme_table <- ttheme_default(
  core = list(fg_params = list(cex = 0.9, hjust = 0, x = 0.15),
              padding = unit(c(10, 2), "mm"),
              bg_params = list(fill = NA)),
  colhead = list(bg_params = list(fill = NA))
)

table_plot_dev <- tableGrob(tab_dev, rows = NULL, theme = theme_table)
table_plot_val <- tableGrob(tab_val, rows = NULL, theme = theme_table)

# Combine plots
f2d_table <- ggdraw() +
  draw_plot(table_plot_dev, x = 0, y = 0.51, height = 0.5) +
  draw_plot(table_plot_val, x = 0, y = 0.08, height = 0.5)

# save table
saveRDS(f2d_table, file = './data_out/fig.2D.table.rds')

# Background elements
f2d_background <- ggplot() +
  annotate("rect", xmin = 0, xmax = 1, ymin = 0, ymax = 1,
           fill = lighten("lightpink", 0.6), color = NA) +
  theme_void()

# save table background
saveRDS(f2d_background, file = './data_out/fig.2D.table.background.rds')
```

Merge fig. 2A-D

```
library(cowplot)
library(ggplot2)
library(patchwork)
```

```
##
## Attaching package: 'patchwork'

## The following object is masked from 'package:cowplot':
##
##     align_plots
```

```
# load figures
f2a <- readRDS('./data_out/fig.2A.rds')
f2b <- readRDS('./data_out/fig.2b.rds')
f2c <- readRDS('./data_out/fig.2C.rds')
f2d <- readRDS('./data_out/fig.2D.rds')
```

```r
f2b_table <- readRDS('./data_out/fig.2B.table.rds')
f2b_table_background <- readRDS('./data_out/fig.2B.table.background.rds')

f2d_table <- readRDS('./data_out/fig.2D.table.rds')
f2d_table_background <- readRDS('./data_out/fig.2D.table.background.rds')




p=ggdraw()+
  draw_plot(f2a, x=0, y=0.5, width = 0.5, height = 0.5)+
  draw_plot(f2b, x=0.5, y=0.5, width = 0.35, height = 0.5)+
  draw_plot(f2b_table_background, x=0.84, y=0.472, width = 0.16, height = 0.6)+
  draw_plot(f2b_table, x=0.87, y=0.5, width = 0.1, height = 0.5)+
  draw_plot(f2c, x=0, y=0, width = 0.5, height = 0.5)+
  draw_plot(f2d, x=0.5, y=0, width = 0.35, height = 0.5)+
  draw_plot(f2d_table_background, x=0.84, y=-0.025, width = 0.16, height = 0.55)+
  draw_plot(f2d_table, x=0.87, y=0, width = 0.1, height = 0.5)


# Create a side title as a separate plot
side_title1 <- ggplot() +
  annotate("text", x = 0, y = 0, label = "Hb variants classifier",
           angle = 90, size = 5) +
  theme_void()

# Create a side title as a separate plot
side_title2 <- ggplot() +
  annotate("text", x = 0, y = 0, label = " -thalassemia trait classifier",
           angle = 90, size = 5) +
  theme_void()


plot_grid(side_title1 / side_title2, p, rel_widths = c(0.05,1))
```

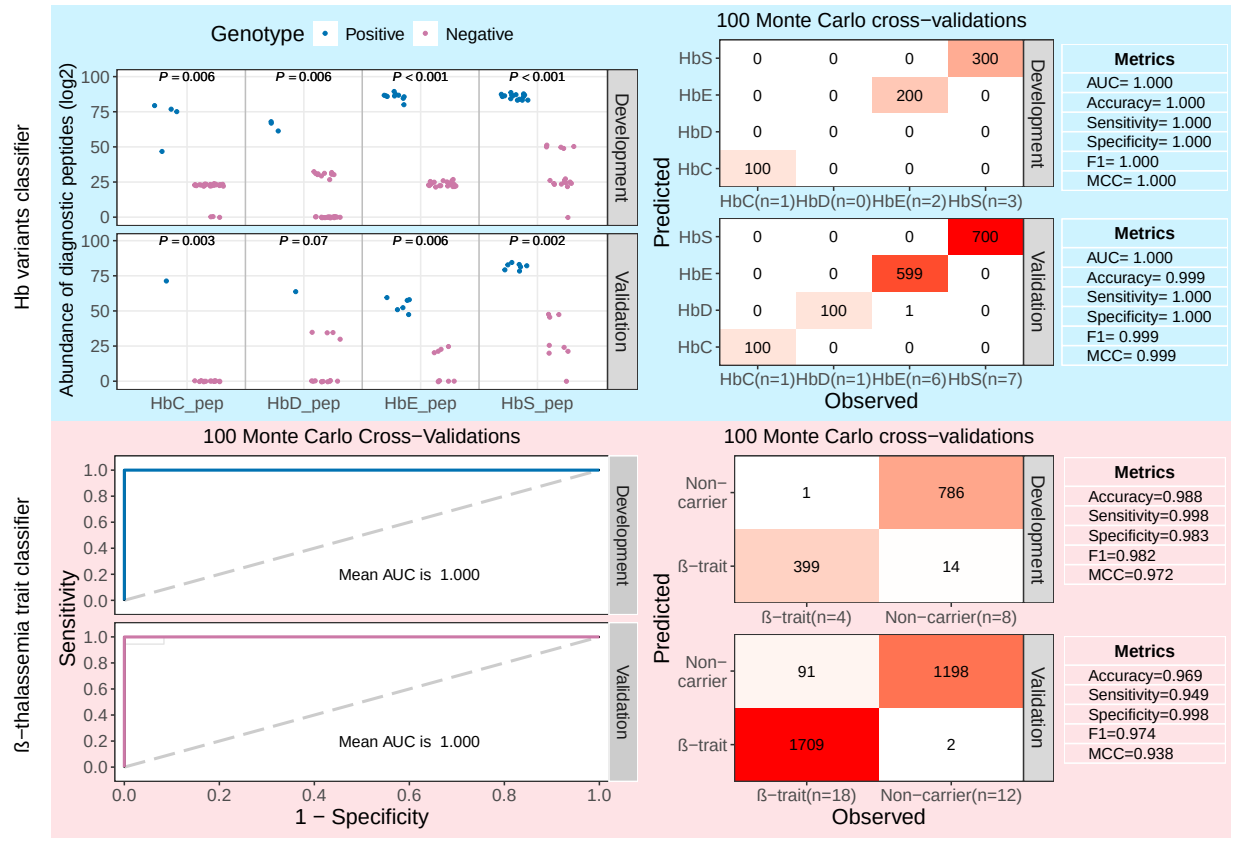


fig. 3

```
library(tidyverse)
library(caret)
library(randomForest)
library(reshape2)
library(pROC)
library(rpart)
library(rpart.plot)
library(doBy)
library(knitr)
library(kableExtra)


# Clear workspace
rm(list = ls())

# Load data
df <- read.csv('./data_in/thalassemia.csv')
df$group[df$group == 'BT'] <- ' -trait'
df$group[df$group == 'CTR'] <- 'Non-carrier'

# Filter development dataset
df <- df[df$class == 'Development', ]
```

```r
# Reshape data
input_dev <- dcast(sample + group ~ peptide, data = df, value.var = 'log')
rownames(input_dev) <- input_dev$sample
input_dev <- input_dev %>% select(-sample)
input_dev$group <- factor(input_dev$group)

# # Fit model and get results
# ot <- NULL
# for (i in 1:500) {
#   trainIndex <- createDataPartition(input_dev$group, p = 0.75, list = FALSE)
#   trainData <- input_dev[trainIndex, ]
#   testData <- input_dev[-trainIndex, ]
#
#   # Train decision tree model
#   tree_model <- rpart(
#     group ~ .,
#     data = trainData,
#     method = "class",
#     control = rpart.control(cp = 0.0001, minsplit = 1, maxdepth = 20, minbucket = 1, xval = 10)
#   )
#
#   # Prune the tree based on best cp (1-SE rule)
#   best_cp <- tree_model$cptable[which.min(tree_model$cptable[, "xerror"]), "CP"]
#   pruned_tree <- prune(tree_model, cp = best_cp)
#
#   # Compute accuracy on test data
#   predictions <- predict(pruned_tree, newdata = testData, type = 'class')
#   conf <- confusionMatrix(predictions, testData$group)
#   acc_dev <- sum(diag(conf$table)) / sum(conf$table)
#
#   # Compute AUC on test data
#   predicted_probs_dev <- predict(pruned_tree, testData, type = "prob")
#   roc_dev <- roc(testData$group, predicted_probs_dev[, "BTT"], levels = c('CTR', 'BTT'))
#   auc_dev <- auc(roc_dev)
#
#   # Load validation data
#   df_val <- read.csv('./manuscript_v2/thalassemia.csv')
#   df_val$group[df_val$group == 'BT'] <- 'BTT'
#   df_val_batch <- df_val[df_val$class == 'Validation', ]
#
#   # Reshape validation data
#   input_val_batch <- dcast(sample + group ~ peptide, data = df_val_batch, value.var = 'log')
#
#   # Align validation data with training variables
#   var_model <- colnames(select(trainData, -group))
#   var_model <- c(var_model, 'group')
#   missing_cols <- setdiff(var_model, colnames(input_val_batch))
#   input_val_batch[missing_cols] <- 0
#   input_val_batch <- input_val_batch[var_model]
#
#   # Compute accuracy on validation data
#   predictions_val <- predict(pruned_tree, newdata = input_val_batch, type = 'class')
#   acc_val_batch <- mean(predictions_val == input_val_batch$group)
```

```
#
#   # Compute AUC on validation data
#   predicted_probs_val <- predict(pruned_tree, input_val_batch, type = "prob")
#   roc_val_batch <- roc(input_val_batch$group, predicted_probs_val[, "BTT"], levels = c('CTR', 'BTT'))
#   auc_val_batch <- as.numeric(auc(roc_val_batch))
#
#   # Store results
#   ot_dev <- data.frame(iteration = i, class = 'Development', auc = auc_dev, acc = acc_dev, var = past
#   ot_val_batch <- data.frame(iteration = i, class = 'Batch', auc = auc_val_batch, acc = acc_val_batch
#   ot <- rbind(ot, ot_dev, ot_val_batch)
# }
#
# # Save results
# saveRDS(ot, file = './data_out/fig.3.decision.tree.rds')

# Load results
ot <- readRDS('./data_out/fig.3.decision.tree.rds')
ot_acc <- NULL

# Compute accuracy statistics
for (i in unique(ot$class)) {
  dc <- dcast(var + iteration ~ class, value.var = 'acc', data = ot[ot$class %in% i, ])
  dc1 <- summaryBy(as.formula(paste(c(i, 'var'), collapse = '~')), data = dc, FUN = c(mean, sd))
  colnames(dc1) <- c('var', 'Mean_accuracy', 'SD_accuracy')
  freq <- data.frame(table(dc$var))
  colnames(freq) <- c('var', 'Freq')
  dc2 <- merge(dc1, freq, by = 'var', all = TRUE)
  dc2$class <- i
  ot_acc <- rbind(ot_acc, dc2)
}

# Prepare final accuracy table
tmp1 <- dcast(var ~ class, data = ot_acc, value.var = 'Mean_accuracy') %>% arrange(Development)
colnames(tmp1)[2:3] <- paste(colnames(tmp1)[2:3], 'accuracy', sep = '_')

tmp2 <- dcast(var ~ class, data = ot_acc, value.var = 'Freq') %>% arrange(Development)
colnames(tmp2)[2:3] <- paste(colnames(tmp2)[2:3], 'freq', sep = '_')

tab <- merge(tmp1, tmp2, by = 'var') %>% arrange(Development_accuracy)
tab <- tab %>%
  select(-Batch_freq) %>%
  rename(Freq = Development_freq, Tree_structure = var, Validation_accuracy = Batch_accuracy) %>%
  select(Tree_structure, Development_accuracy, Validation_accuracy, Freq)
tab$Development_accuracy <- as.numeric(format(round(tab$Development_accuracy, 3), nsmall = 3))
tab$Validation_accuracy <- as.numeric(format(round(tab$Validation_accuracy, 3), nsmall = 3))
tab <- tab %>% rename(Frequency = Freq)

kable(tab, format = "html", digits = 3) %>%
  kable_styling(font_size = 10)
```

Tree_structure

Development_accuracy

Validation_accuracy

Frequency

HBD_LLGNVLVCVLARNFGK,<leaf>,<leaf>

0.583

0.600

1

HBD_EFTPQMQAAYQKVVAGVANALAHK,HBA_FLASVSTVLTSKYR,<leaf>,<leaf>,<leaf>

0.750

0.617

2

HBD_TAVNALWGKVNVDAVGGEALGR,<leaf>,<leaf>

0.801

0.367

18

HBD_EFTPQMQAAYQKVVAGVANALAHK,<leaf>,HBA_LLSHCLLVTLAAHLPAEFTPAVHASLDK,<leaf>,<leaf>

0.833

0.933

1

HBD_EFTPQMQAAYQKVVAGVANALAHK,HBA_LLSHCLLVTLAAHLPAEFTPAVHASLDK,<leaf>,<leaf>,<leaf>

0.833

0.500

2

HBD_EFTPQMQAAYQKVVAGVANALAHK,<leaf>,HBA_AAWGKVGAHAGEYGAEALER,<leaf>,<leaf>

0.891

0.900

16

HBD_EFTPQMQAAYQKVVAGVANALAHK,HBA_LRVDPVNFK,<leaf>,<leaf>,<leaf>

0.917

0.433

26

HBD_EFTPQMQAAYQKVVAGVANALAHK,<leaf>,<leaf>

0.924

0.923

424

HBD_EFTPQMQAAYQKVVAGVANALAHK,<leaf>,HBA_TYFPHFDLSHGSAQVKGHGK,<leaf>,<leaf>

0.992

0.937

10

```r
# Decision tree modeling
vars <- c("HBD_EFTPQMQAAYQKVVAGVANALAHK", "group")
input_dev$group <- factor(as.character(input_dev$group), levels = c(' -trait', 'Non-carrier'))
input_dev <- input_dev[, vars]

tree_model <- rpart(
  group ~ .,
  data = input_dev,
  method = "class",
  parms = list(split = "information"),
  control = rpart.control(cp = 1e-7, minsplit = 2, maxdepth = 2, minbucket = 1, xval = 10)
)

# Prune and plot tree
best_cp <- tree_model$cptable[which.min(tree_model$cptable[, "xerror"]), "CP"]
pruned_tree <- prune(tree_model, cp = best_cp)
rpart.plot(pruned_tree, type = 5, extra = 0, digits = 5)
```

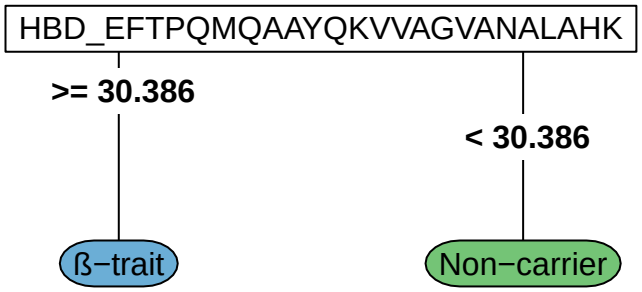
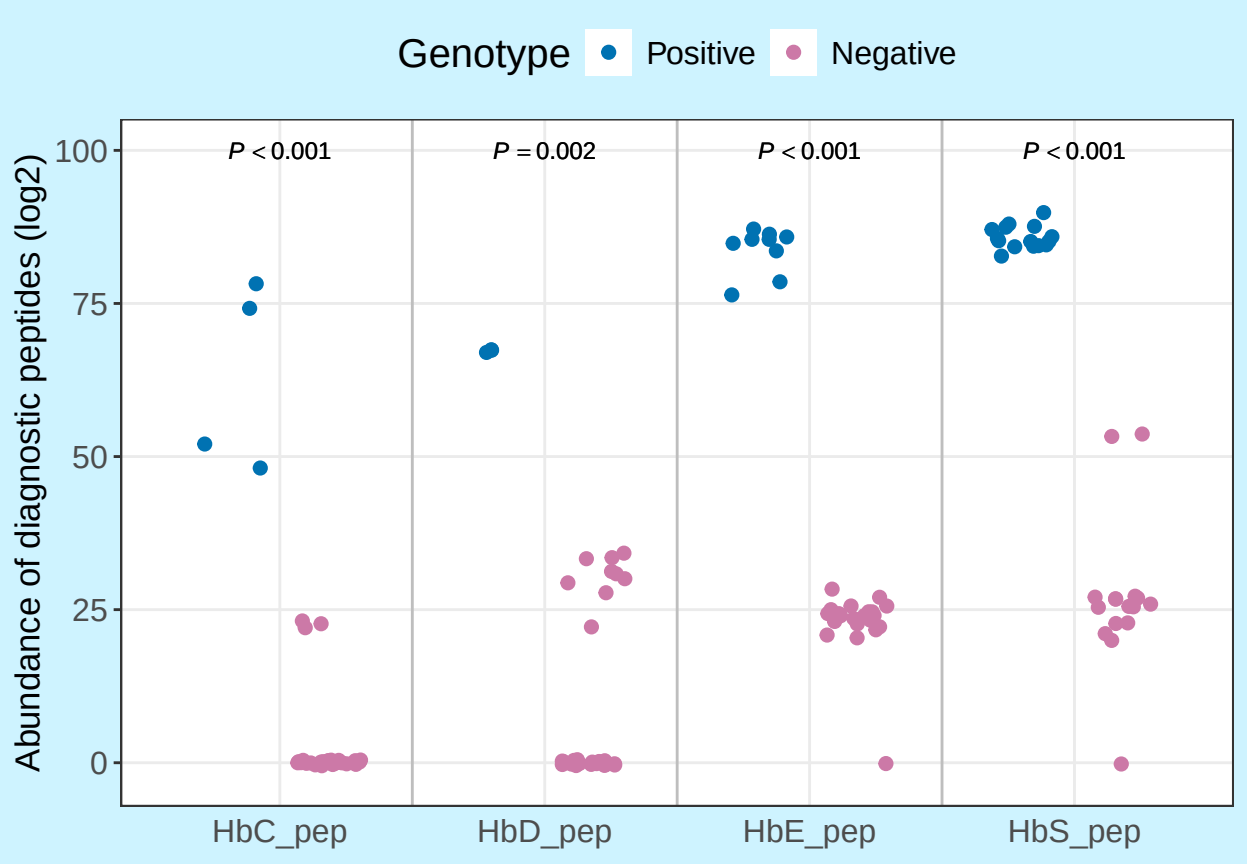


fig. 4A

```r
library(ggplot2)
library(tidyverse)
library(ggthemes)
```

```r
library(ggpubr)
library(colorspace)

# Clear workspace
rm(list = ls())

# Load data
df <- read.csv('./data_in/variants.csv')

# Subset groups and classes
df <- df %>%
  filter(!group %in% c('BT', 'CTR')) %>%
  filter(class == 'Whole blood development')

# Rename variants
df <- df %>%
  mutate(variant = case_when(
    variant == 'HbC' ~ 'HbC_pep',
    variant == 'HbD' ~ 'HbD_pep',
    variant == 'HbE' ~ 'HbE_pep',
    variant == 'HbS' ~ 'HbS_pep',
    TRUE ~ variant
  ))

# Order groups
df$group <- factor(df$group, levels = c('HbC', 'HbD', 'HbE', 'HbS'))

# Define colors
colors <- readRDS('./data_in/colors.rds')

# Assign xmin and xmax
df <- df %>%
  group_by(variant) %>%
  mutate(
    xmin = case_when(
      variant == 'HbC_pep' ~ 0.6,
      variant == 'HbD_pep' ~ 1.6,
      variant == 'HbE_pep' ~ 2.6,
      variant == 'HbS_pep' ~ 3.6
    ),
    xmax = case_when(
      variant == 'HbC_pep' ~ 1.4,
      variant == 'HbD_pep' ~ 2.4,
      variant == 'HbE_pep' ~ 3.4,
      variant == 'HbS_pep' ~ 4.4
    )
  )

# Prepare data for plotting
tmp <- df %>% mutate(
  variant2 = sapply(strsplit(variant, '_'), '[', 1),
  binary = ifelse(group == variant2, 'Positive', 'Negative')
)
```

```r
tmp$binary <- factor(tmp$binary, levels = c('Positive', 'Negative'))

# Compute p-values
pval <- compare_means(log ~ binary, method = 'wilcox.test', data = tmp, group.by = 'variant')
pval$pval <- mapply(function(p) {
  if (p < 0.001) {
    "italic(P) < '0.001'"
  } else if (p < 0.01) {
    sprintf("italic(P) == \"%s\"", formatC(p, format = "f", digits = 3))
  } else if (p < 0.05) {
    sprintf("italic(P) == \"%s\"", formatC(p, format = "f", digits = 2))
  } else {
    "italic(P) > '0.05'"
  }
}, pval$p.adj)

tmp <- merge(tmp, pval[, c('variant', 'pval')], by = 'variant', all.x = TRUE)

# Generate plot
f4a <- ggplot(tmp, aes(variant, log, color = binary)) +
  geom_point(position = position_jitterdodge(dodge.width = 0.75, jitter.width = 0.5, jitter.height = 0.5
  scale_color_manual('Genotype', values = colors) +
  labs(y = 'Abundance of diagnostic peptides (log2)') +
  theme_bw() +
  theme(
    text = element_text(size = 15),
    axis.title.x.bottom = element_blank(),
    panel.grid.minor = element_blank(),
    axis.ticks.x.bottom = element_blank(),
    axis.title.y.left = element_text(size = 13),
    legend.position = 'top',
    plot.background = element_rect(fill = lighten('lightblue', 0.6), color = NA),
    legend.background = element_rect(fill = lighten('lightblue', 0.6), color = NA)
  ) +
  scale_y_continuous(limits = c(-2, 100)) +
  geom_vline(xintercept = c(1.5, 2.5, 3.5), color = 'gray') +
  geom_text(data = unique(tmp[, c('variant', 'pval', 'binary')]),
            aes(label = pval, y = 100),
            color = 'black',
            size = 3,
            parse = TRUE)

f4a
```



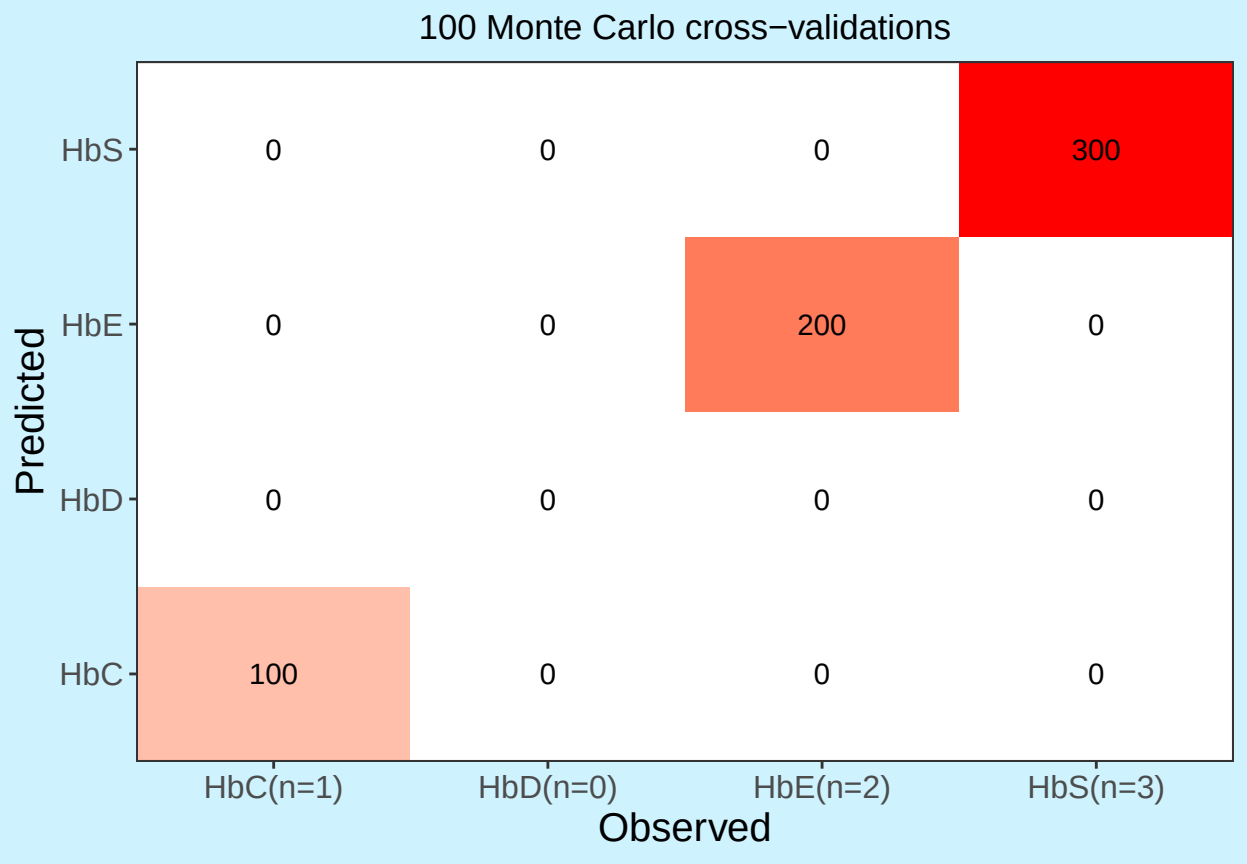

```
# save figure
saveRDS(f4a, file = './data_out/fig.4A.rds')
```

fig.4B

```
library(ggplot2)
library(tidyverse)
library(ggthemes)
library(ggpubr)
library(reshape2)
library(pROC)
library(caret)
library(cowplot)
library(gridExtra)
library(colorspace)

# Clear workspace
rm(list = ls())

# Load data
df <- read.csv('./data_in/variants.peptides.csv')

# Subset groups and classes
df <- df %>%
```

```r
  filter(!group %in% c('BT', 'CTR')) %>%
  filter(class == 'Whole blood development')

# Reshape data
def <- dcast(group + sample ~ peptide, data = df, value.var = 'value') %>%
  select(-sample)
def$group <- factor(def$group, levels = c('HbC', 'HbD', 'HbE', 'HbS'))

# # run in parallel
# num_cores <- detectCores() - 1  # Use one less than the available cores
# cl <- makeCluster(num_cores)
# registerDoParallel(cl)
#
# results <- foreach(i = 1:100, .combine = rbind, .packages = c("caret",'tidyverse','pROC')) %dopar% {
#
#   # Split the data based on the outer fold
#   train_index <- createDataPartition(def$group, p = 0.75, list = FALSE)
#   trainData <- def[train_index, ]
#   testData <- def[-train_index, ]
#
#   tuneGrid <- expand.grid(mtry = seq(1,10,1))
#   control <- trainControl(method = "repeatedcv", number = 10, repeats = 1)
#
#   # Train the model with inner CV for tuning
#   model <- train(group ~ ., data = trainData,
#                  method = "rf",
#                  tuneGrid = tuneGrid,
#                  trControl = control,
#                  ntree = 5000)
#
#   # accuracy by default on the test
#   pred = predict(model, newdata = testData, type = 'raw')
#   acc_value_test = mean(pred == testData$group)
#
#   # auc
#   probs = predict(model, newdata = testData, type = 'prob')
#   true_label = testData$group
#   auc = as.numeric(auc(multiclass.roc(true_label, probs)))
#
#   # save data for test
#   ot_test = data.frame(pred, actual=testData$group, iteration=i,
#                        class='Development', acc=acc_value_test,
#                        auc=auc)
#
#   return(ot_test)
# }
# head(results)
#
# # Save results
# saveRDS(results, file = './data_out/fig.4B.data.rds')

# Load results
results <- readRDS('./data_out/fig.4B.data.rds')
```

```r
# Compute frequency table
freq <- results %>%
  filter(class == 'Whole blood development') %>%
  count(class, pred, actual)

# Create base table
categories <- c("HbC", "HbD", "HbE", "HbS")
tab <- expand.grid(pred = categories, actual = categories)
tab <- merge(tab, freq, by = c('pred', 'actual'), all = TRUE)
tab$n[is.na(tab$n)] <- 0
tab$class[is.na(tab$class)] <- 'Whole blood development'

# Rename x-axis labels
tab$actual <- as.character(tab$actual)
tab$actual[tab$actual == 'HbC' & tab$class=='Whole blood development'] = 'HbC(n=1)'
tab$actual[tab$actual == 'HbD' & tab$class=='Whole blood development'] = 'HbD(n=0)'
tab$actual[tab$actual == 'HbE' & tab$class=='Whole blood development'] = 'HbE(n=2)'
tab$actual[tab$actual == 'HbS' & tab$class=='Whole blood development'] = 'HbS(n=3)'


# Define colors
colors <- readRDS('./data_in/colors.rds')
light_color <- lighten(colors[1], amount = 0.7)

# Generate heatmap
f4b <- ggplot(tab, aes(actual, pred, fill = n)) +
  geom_tile() +
  geom_text(aes(label = n)) +
  scale_fill_gradient(low = 'white', high = 'red') +
  theme_bw() +
  theme(
    text = element_text(size = 15),
    legend.position = 'None',
    plot.background = element_rect(fill = lighten('lightblue', 0.6), color = NA),
    plot.title = element_text(hjust = 0.5, size = 13)
  ) +
  labs(x = 'Observed', y = 'Predicted') +
  ggtitle('100 Monte Carlo cross-validations')+
  scale_x_discrete(expand = c(0,0))+
  scale_y_discrete(expand = c(0,0))

f4b
```




```
# Compute performance metrics
conf <- confusionMatrix(factor(results$pred), factor(results$actual))
acc <- paste('Accuracy=', format(round(conf$overall['Accuracy'], 3), nsmall = 3), sep = '')
sensitivity <- paste('Sensitivity=', format(round(mean(data.frame(conf$byClass)[,'Sensitivity']), 3), ns
specificity <- paste('Specificity=', format(round(mean(data.frame(conf$byClass)[,'Specificity']), 3), ns
f1_score <- paste('F1=', format(round(mean(conf$byClass[,"F1"]), 3), nsmall = 3), sep = '')
mcc <- paste('MCC=', format(round(mltools::mcc(preds = results$pred, actuals = results$actual), 3), nsma
auc <- paste('AUC=', format(round(aggregate(auc ~ class, data = results, FUN = mean)$auc, 3), nsmall = 3
tab_metrics <- data.frame(Metrics = c(auc, acc, sensitivity, specificity, f1_score, mcc))

# Generate table plot
table_plot <- tableGrob(tab_metrics, rows = NULL, theme = ttheme_default(
  core = list(fg_params = list(cex = 1, hjust = 0, x = 0.15), padding = unit(c(10, 2), "mm"), bg_params
  colhead = list(bg_params = list(fill = NA))
))


f4b_table <- ggdraw() + draw_plot(table_plot, x = 0, y = 0.48, height = 0.5)
f4b_background <- ggplot() +
  annotate("rect", xmin = 0, xmax = 1, ymin = 0, ymax = 1,
           fill = lighten("lightblue", 0.6), color = NA) +  # Background rectangle
  theme_void()  # Remove axes and gridlines

# save figure and table and background
saveRDS(f4b, file = './data_out/fig.4B.rds')
saveRDS(f4b_table, file = './data_out/fig.4B.table.rds')
```

```r
saveRDS(f4b_background, file = './data_out/fig.4B.table.background.rds')
```

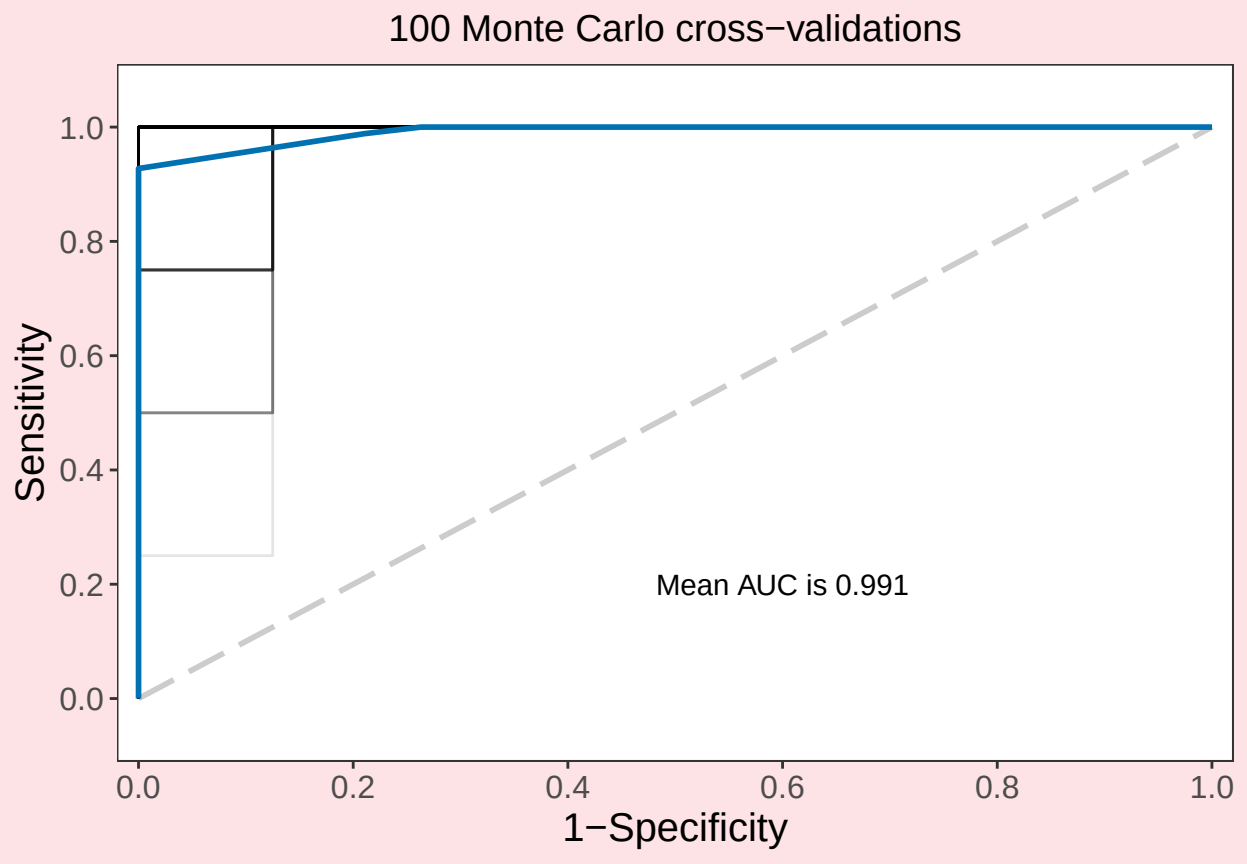
fig.4C

```r
library(ggplot2)
library(ggthemes)
library(reshape2)
library(pROC)
library(tidyverse)
library(caret)
library(foreach)
library(doParallel)

rm(list = ls())

# read data
df <- read.csv('./data_in/thalassemia.csv')

# subset for groups
df <- df[df$group %in% c('BT','CTR'), ]

df.blood <- df%>%
  filter(class == 'Whole blood development')

# make data
def <- dcast(group+sample~peptide, data=df.blood,value.var = 'log') %>%
  select(-sample)
def$group <- factor(def$group, levels = c('CTR','BT'))

# # run in parallel
# num_cores <- detectCores() - 1  # Use one less than the available cores
# cl <- makeCluster(num_cores)
# registerDoParallel(cl)
#
# results <- foreach(i = 1:100, .combine = rbind, .packages = c("caret","pROC")) %dopar% {
#
#   # Split the data based on the outer fold
#   train_index <- createDataPartition(def$group, p = 0.75, list = FALSE)
#   trainData.blood <- def[train_index, ]
#   testData.blood <- def[-train_index, ]
#
#   tuneGrid <- expand.grid(mtry = seq(2,10,1))
#   control <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
#
#   # Train the model with inner CV for tuning
#   model <- train(group ~ ., data = trainData.blood,
#                  method = "rf",
#                  tuneGrid = tuneGrid,
#                  trControl = control,
#                  ntree = 5000)
#
```

```
#   # accuracy
#   pred_raw <- predict(model, newdata = testData.blood, type = 'raw')
#   acc_blood <- mean(pred_raw == testData.blood$group)
#
#   # auc
#   pred_prob <- predict(model, testData.blood, type = "prob")
#   roc <- roc(testData.blood$group, pred_prob[, "BT"])
#   auc <- as.numeric(auc(roc))
#
#   # collect data
#   ot_roc <- data.frame(iteration=i,
#                              class='Whole blood development',
#                              acc=acc,
#                              auc=auc,
#                              specificity=roc$specificities,
#                              sensitivity=roc$sensitivities,
#                              type='roc',
#                              prob=NA,
#                              pred=NA,
#                              actual=NA)
#   ot_prob <- data.frame(iteration=i,
#                               class='Whole blood development',
#                               acc=acc,
#                               auc=auc,
#                               specificity=NA,
#                               sensitivity=NA,
#                               type='prob',
#                               prob=pred_prob[, "BT"],
#                               pred=as.character(pred_label),
#                               actual=as.character(testData.blood$group))
#
#
#
#
#   # return data
#   tmp = rbind(ot_roc, ot_prob)
#
#   return(tmp)
#
# }

#saveRDS(results, file = './data_out/fig.4CD.data.rds')

# Load results
data_path <- './data_out/fig.4CD.data.rds'
results <- readRDS(data_path)
results$specificity <- 1 - results$specificity
results <- results[results$class == 'Whole blood development' & results$type == 'roc',]

# Define common FPR values for interpolation
common_fpr <- seq(0, 1, length.out = 20)

# Make fpr
```

```r
results <- results %>%
  mutate(fpr = specificity)

# Compute max sensitivity at FPR = 0
max_sens_at_zero <- results %>%
  filter(fpr == 0) %>%
  group_by(iteration, class) %>%
  summarise(max_sens = max(sensitivity, na.rm = TRUE), .groups = "drop")

# Replace sensitivity when specificity at 0
results_fpr_0 <- results %>%
  left_join(max_sens_at_zero, by = c('iteration', 'class')) %>%
  mutate(sensitivity = ifelse(fpr == 0, max_sens, sensitivity)) %>%
  filter(fpr == 0) %>%
  select(-max_sens) %>%
  distinct()

# Modify results and interpolate
temp_results <- results %>% filter(fpr != 0) %>% rbind(results_fpr_0) %>% arrange(class, iteration, fpr)
interp_df <- temp_results %>%
  group_by(iteration, class) %>%
  reframe(
    fpr = common_fpr,
    sens = approx(x = specificity, y = sensitivity, xout = common_fpr, rule = 1, ties = mean)$y
  )

# Compute summary statistics
summary_df <- interp_df %>%
  group_by(fpr, class) %>%
  summarise(
    mean_sens = mean(sens, na.rm = TRUE),
    .groups = "drop"
  )

# Add (0,0) starting point
summary_df <- rbind(
  data.frame(fpr = 0, class = 'Whole blood development', mean_sens = 0),
  summary_df
)

# Compute mean AUC
auc_text <- results %>%
  group_by(iteration, class) %>%
  summarise(auc = unique(auc), .groups = "drop") %>%
  group_by(class) %>%
  summarise(mean_auc = mean(auc, na.rm = TRUE), .groups = "drop") %>%
  mutate(format = format(mean_auc, digits = 3, nsmall = 3))

# Load colors
colors <- readRDS('./data_in/colors.rds')

# Create diagonal reference line
diagonal_df <- summary_df %>%
```

```r
  group_by(class) %>%
  summarise(fpr_start = 0, sens_start = 0, fpr_end = 1, sens_end = 1, .groups = "drop")

results = results %>%
  arrange(fpr, sensitivity)

# Generate ROC plot
f4c <- ggplot() +
  geom_segment(data = diagonal_df,
               aes(x = fpr_start, y = sens_start, xend = fpr_end, yend = sens_end),
               linetype = "longdash", size = 1, color = "gray80") +
  geom_step(data = results,
            aes(x = fpr, y = sensitivity, group = interaction(iteration, class)),
            alpha = 0.1, size = 0.5) +
  geom_line(data = summary_df,
            aes(x = fpr, y = mean_sens, color = class),
            size = 1) +
  scale_x_continuous(breaks = seq(0, 1, 0.2), expand = c(0.01, 0.01)) +
  scale_y_continuous(breaks = seq(0, 1, 0.2), expand = c(0.01, 0.1)) +
  theme_bw() +
  theme(panel.grid = element_blank(),
        text = element_text(size = 15),
        axis.ticks = element_line(),
        strip.background = element_rect(fill = "gray80", color = 'white'),
        legend.position = 'None',
        plot.title = element_text(hjust = 0.5, size = 14),
        plot.background = element_rect(fill = lighten('lightpink', 0.6), color = NA),
        legend.background = element_rect(fill = lighten('lightpink', 0.6), color = NA)) +
  guides(color = guide_legend(nrow = 1)) +
  scale_color_manual('Group', values = colors) +
  scale_fill_manual('Group', values = colors) +
  labs(x = '1-Specificity', y = 'Sensitivity', title = "100 Monte Carlo cross-validations") +
  geom_text(data = auc_text, aes(x = 0.6, y = 0.2, label = paste("Mean AUC is", format)))

f4c
```

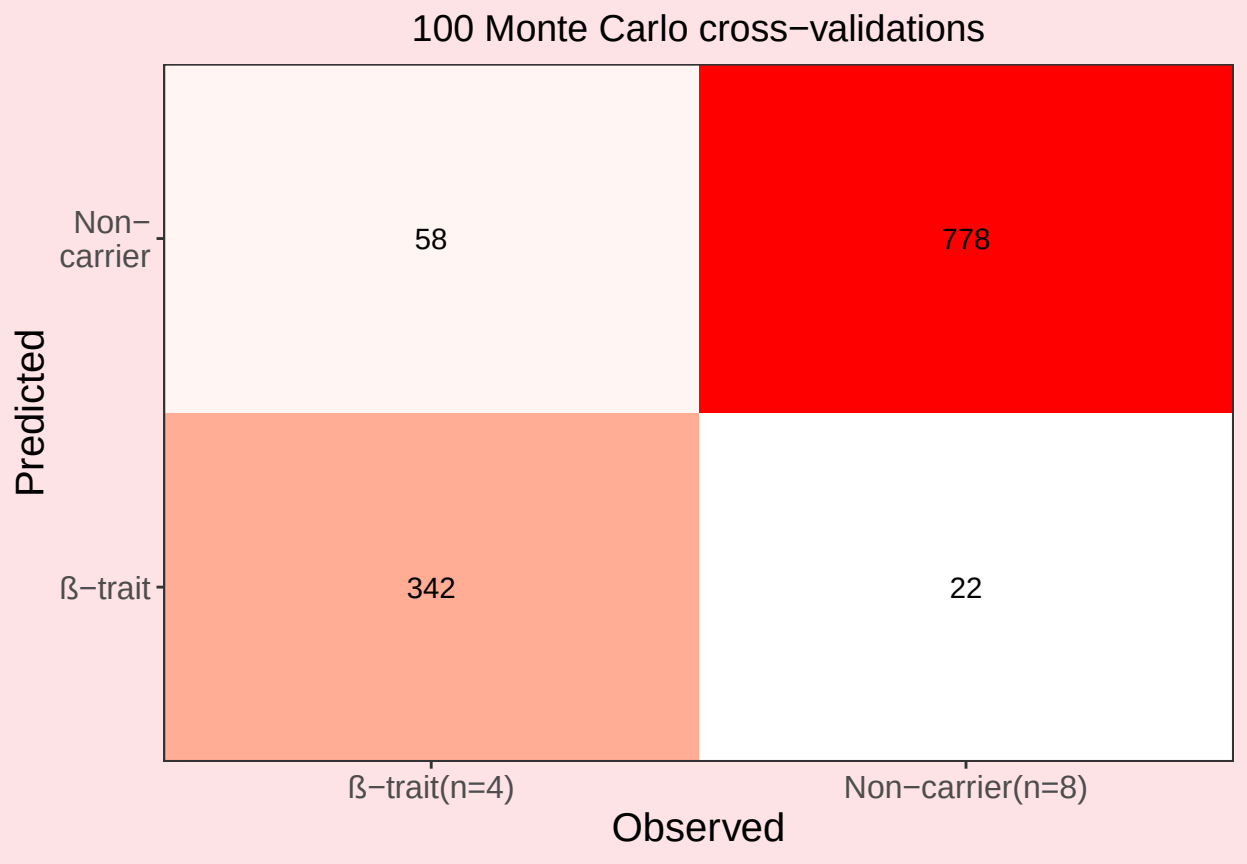



```
# save figure
saveRDS(f4c, file = './data_out/fig.4C.rds')
```

fig.4D

```
# clear environment
rm(list = ls())

# Load libraries
library(ggplot2)
library(tidyverse)
library(colorspace)
library(caret)
library(gridExtra)
library(cowplot)

# Read in data
results <- readRDS('./data_out/fig.4CD.data.rds')
results$pred[results$pred == 'BT'] <- ' -trait'
results$actual[results$actual == 'BT'] <- ' -trait'
results$pred[results$pred == 'CTR'] <- 'Non-\ncarrier'
results$actual[results$actual == 'CTR'] <- 'Non-\ncarrier'

freq <- results %>%
```

```
  filter(type == 'prob' & class == 'Whole blood development') %>%
  count(class, pred, actual)

# Create base table
tab <- expand.grid(pred = c(' -trait', 'Non-\ncarrier'), actual = c(' -trait', 'Non-\ncarrier'))
tab <- merge(tab, freq, by = c('pred', 'actual'), all = TRUE)
tab$n[is.na(tab$n)] <- 0

# rename x axis label
tab$actual <- as.character(tab$actual)
tab$actual[tab$actual == ' -trait'] <- ' -trait(n=4)'
tab$actual[tab$actual == 'Non-\ncarrier'] <- 'Non-carrier(n=8)'

#order
tab$actual <- factor(tab$actual, levels = c(" -trait(n=4)","Non-carrier(n=8)"))


# Generate heatmap
f4d <- ggplot(tab, aes(actual, pred, fill = n)) +
  geom_tile() +
  geom_text(aes(label = n)) +
  scale_fill_gradient(low = 'white', high = 'red') +
  theme_bw() +
  theme(text = element_text(size = 15),
        legend.position = 'None',
        plot.background = element_rect(fill = lighten('lightpink', 0.6), color = NA),
        plot.title = element_text(hjust = 0.5, size = 14)) +
  labs(x = 'Observed', y = 'Predicted', title = "100 Monte Carlo cross-validations")+
  scale_x_discrete(expand = c(0,0))+
  scale_y_discrete(expand = c(0,0))

f4d
```




```
## calculate precision, recall, f1, MCC
# Load results
data_path <- './data_out/fig.4CD.data.rds'
df <- readRDS(data_path)
df <- df %>%
  filter(type == 'prob') %>%
  select(class, pred, actual, iteration)

# Calculate for whole blood
df <- df %>% filter(class == 'Whole blood development')
conf <- confusionMatrix(factor(df$pred), factor(df$actual), positive = 'BT')

# Compute evaluation metrics
acc <- paste('Accuracy=', format(round(conf$overall['Accuracy'], 3), nsmall = 3), sep = '')
sensitivity <- paste('Sensitivity=', format(round(conf$byClass['Sensitivity'], 3), nsmall = 3))
specificity <- paste('Specificity=', format(round(conf$byClass['Specificity'], 3), nsmall = 3))
f1_score <- paste('F1=', format(round(conf$byClass['F1'], 2), nsmall = 3), sep = '')
mcc <- paste('MCC=', format(round(mltools::mcc(preds = df$pred, actuals = df$actual), 2), nsmall = 3), s

# Create results table
tab <- data.frame(Metrics = c(acc, sensitivity, specificity, f1_score, mcc))

# Generate table plot
table_plot <- tableGrob(tab, rows = NULL,
                        theme = ttheme_default(
                          core = list(fg_params = list(cex = 1, hjust = 0, x = 0.15),
```

```
                                    padding = unit(c(10, 2), "mm"),
                                    bg_params = list(fill = NA)),
                        colhead = list(bg_params = list(fill = NA))
                      ))

f4d_table <- ggdraw() + draw_plot(table_plot, x = 0, y = 0.5, height = 0.5)
f4d_background <- ggplot() +
  annotate("rect", xmin = 0, xmax = 1, ymin = 0, ymax = 1,
           fill = lighten("lightpink", 0.6), color = NA) +  # Background rectangle
  theme_void()  # Remove axes and gridlines

# save figure
saveRDS(f4d, file = './data_out/fig.4D.rds')
saveRDS(f4d_table, file = './data_out/fig.4D.table.rds')
saveRDS(f4d_background, file = './data_out/fig.4D.table.background.rds')
```

Merge fig. 4A-D

```
rm(list = ls())

library(cowplot)
library(ggplot2)
library(patchwork)


# load figures
f4a <- readRDS('./data_out/fig.4A.rds')
f4b <- readRDS('./data_out/fig.4B.rds')
f4c <- readRDS('./data_out/fig.4C.rds')
f4d <- readRDS('./data_out/fig.4D.rds')

f4b_table <- readRDS('./data_out/fig.4B.table.rds')
f4b_table_background <- readRDS('./data_out/fig.4B.table.background.rds')

f4d_table <- readRDS('./data_out/fig.4D.table.rds')
f4d_table_background <- readRDS('./data_out/fig.4D.table.background.rds')



p=ggdraw()+
  draw_plot(f4a, x=0, y=0.5, width = 0.5, height = 0.5)+
  draw_plot(f4b, x=0.5, y=0.5, width = 0.35, height = 0.5)+
  draw_plot(f4b_table_background, x=0.84, y=0.472, width = 0.16, height = 0.6)+
  draw_plot(f4b_table, x=0.87, y=0.5, width = 0.1, height = 0.5)+
  draw_plot(f4c, x=0, y=0, width = 0.5, height = 0.5)+
  draw_plot(f4d, x=0.5, y=0, width = 0.35, height = 0.5)+
  draw_plot(f4d_table_background, x=0.84, y=-0.025, width = 0.16, height = 0.55)+
  draw_plot(f4d_table, x=0.87, y=0, width = 0.1, height = 0.5) # save 1200*850
```

```
# Create a side title as a separate plot
side_title1 <- ggplot() +
  annotate("text", x = 0, y = 0, label = "Hb variants classifier",
           angle = 90, size = 5) +
  theme_void()

# Create a side title as a separate plot
side_title2 <- ggplot() +
  annotate("text", x = 0, y = 0, label = " -thalassemia trait classifier",
           angle = 90, size = 5) +
  theme_void()

plot_grid(side_title1 / side_title2, p, rel_widths = c(0.05,1)) # save 1200*800
```

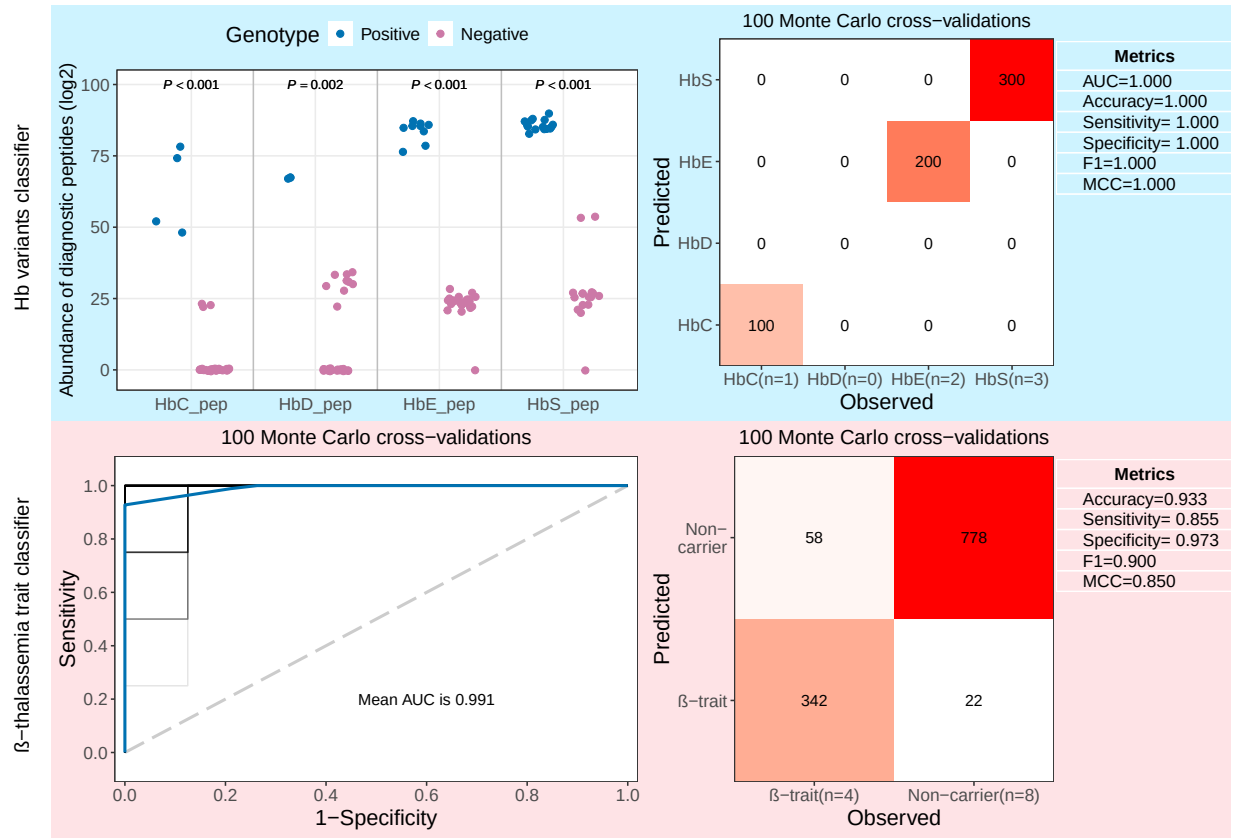
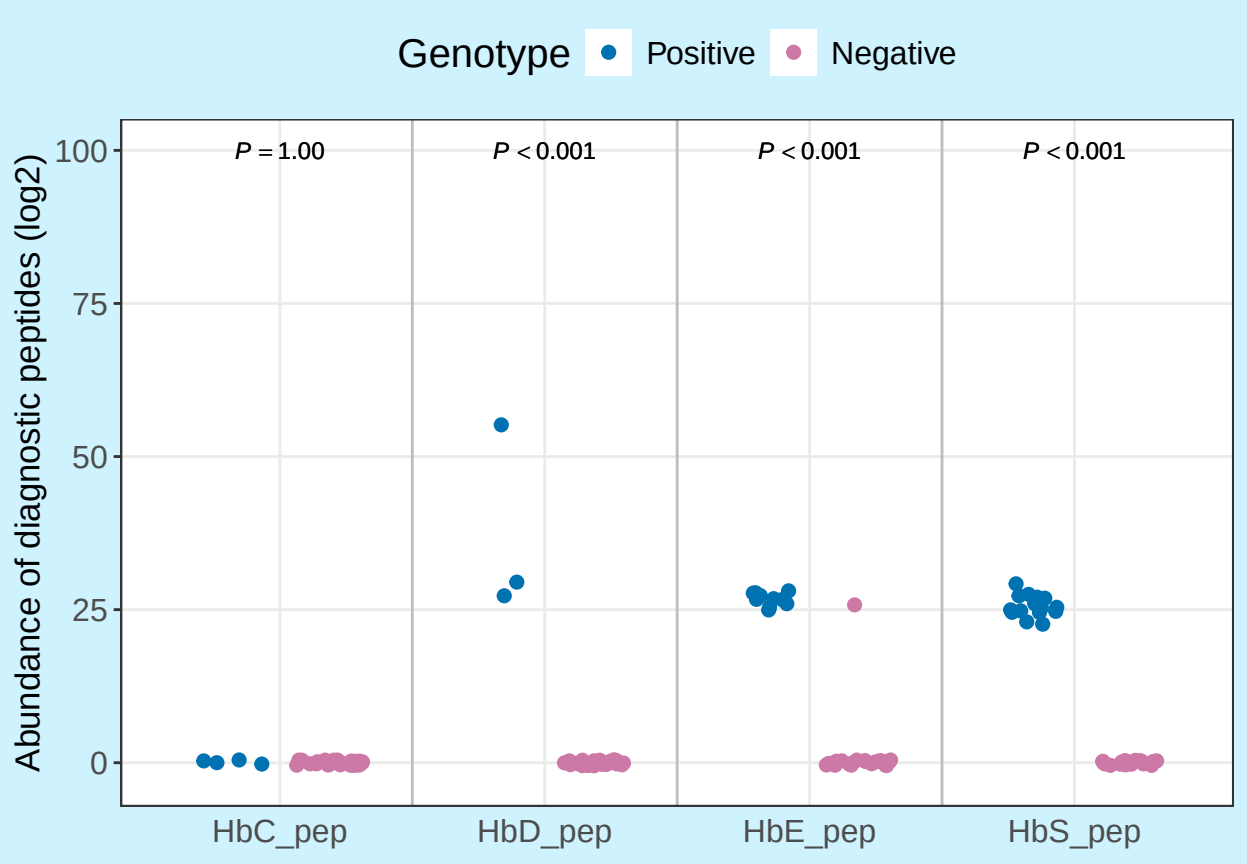


fig. 5A

```
library(ggplot2)
library(tidyverse)
library(ggthemes)
library(ggpubr)
library(colorspace)

# Clear workspace
rm(list = ls())
```

```r
# Load data
df <- read.csv('./data_in/variants.csv')

# Subset groups and classes
df <- df %>%
  filter(!group %in% c('BT', 'CTR')) %>%
  filter(class == 'Plasma development')

# Rename variants
df <- df %>%
  mutate(variant = case_when(
    variant == 'HbC' ~ 'HbC_pep',
    variant == 'HbD' ~ 'HbD_pep',
    variant == 'HbE' ~ 'HbE_pep',
    variant == 'HbS' ~ 'HbS_pep',
    TRUE ~ variant
  ))

# Order groups
df$group <- factor(df$group, levels = c('HbC', 'HbD', 'HbE', 'HbS'))

# Define colors
colors <- readRDS('./data_in/colors.rds')

# Assign xmin and xmax
df <- df %>%
  group_by(variant) %>%
  mutate(
    xmin = case_when(
      variant == 'HbC_pep' ~ 0.6,
      variant == 'HbD_pep' ~ 1.6,
      variant == 'HbE_pep' ~ 2.6,
      variant == 'HbS_pep' ~ 3.6
    ),
    xmax = case_when(
      variant == 'HbC_pep' ~ 1.4,
      variant == 'HbD_pep' ~ 2.4,
      variant == 'HbE_pep' ~ 3.4,
      variant == 'HbS_pep' ~ 4.4
    )
  )

# Prepare data for plotting
tmp <- df %>% mutate(
  variant2 = sapply(strsplit(variant, '_'), '[', 1),
  binary = ifelse(group == variant2, 'Positive', 'Negative')
)
tmp$binary <- factor(tmp$binary, levels = c('Positive', 'Negative'))

# Compute p-values
pval <- compare_means(log ~ binary, method = 'wilcox.test', data = tmp, group.by = 'variant')
pval$pval <- mapply(function(p) {
  if (p < 0.001) {
```

```
    "italic(P) < '0.001'"
  } else if (p < 0.01) {
    sprintf("italic(P) == \"%s\"", formatC(p, format = "f", digits = 3))
  } else if (p < 0.05) {
    sprintf("italic(P) == \"%s\"", formatC(p, format = "f", digits = 2))
  } else {
    "italic(P) > '0.05'"
  }
}, pval$p.adj)

tmp <- merge(tmp, pval[, c('variant', 'pval')], by = 'variant', all.x = TRUE)
tmp[tmp$variant == 'HbC_pep'&tmp$class == 'Plasma development',]$pval = "italic(P) == '1.00'"

# Generate plot
f5a <- ggplot(tmp, aes(variant, log, color = binary)) +
  geom_point(position = position_jitterdodge(dodge.width = 0.75, jitter.width = 0.5, jitter.height = 0.5
  scale_color_manual('Genotype', values = colors) +
  labs(y = 'Abundance of diagnostic peptides (log2)') +
  theme_bw() +
  theme(
    text = element_text(size = 15),
    axis.title.x.bottom = element_blank(),
    panel.grid.minor = element_blank(),
    axis.ticks.x.bottom = element_blank(),
    axis.title.y.left = element_text(size = 13),
    legend.position = 'top',
    plot.background = element_rect(fill = lighten('lightblue', 0.6), color = NA),
    legend.background = element_rect(fill = lighten('lightblue', 0.6), color = NA)
  ) +
  scale_y_continuous(limits = c(-2, 100)) +
  geom_vline(xintercept = c(1.5, 2.5, 3.5), color = 'gray') +
  geom_text(data = unique(tmp[, c('variant', 'pval', 'binary')]),
            aes(label = pval, y = 100),
            color = 'black',
            size = 3,
            parse = TRUE)

f5a
```



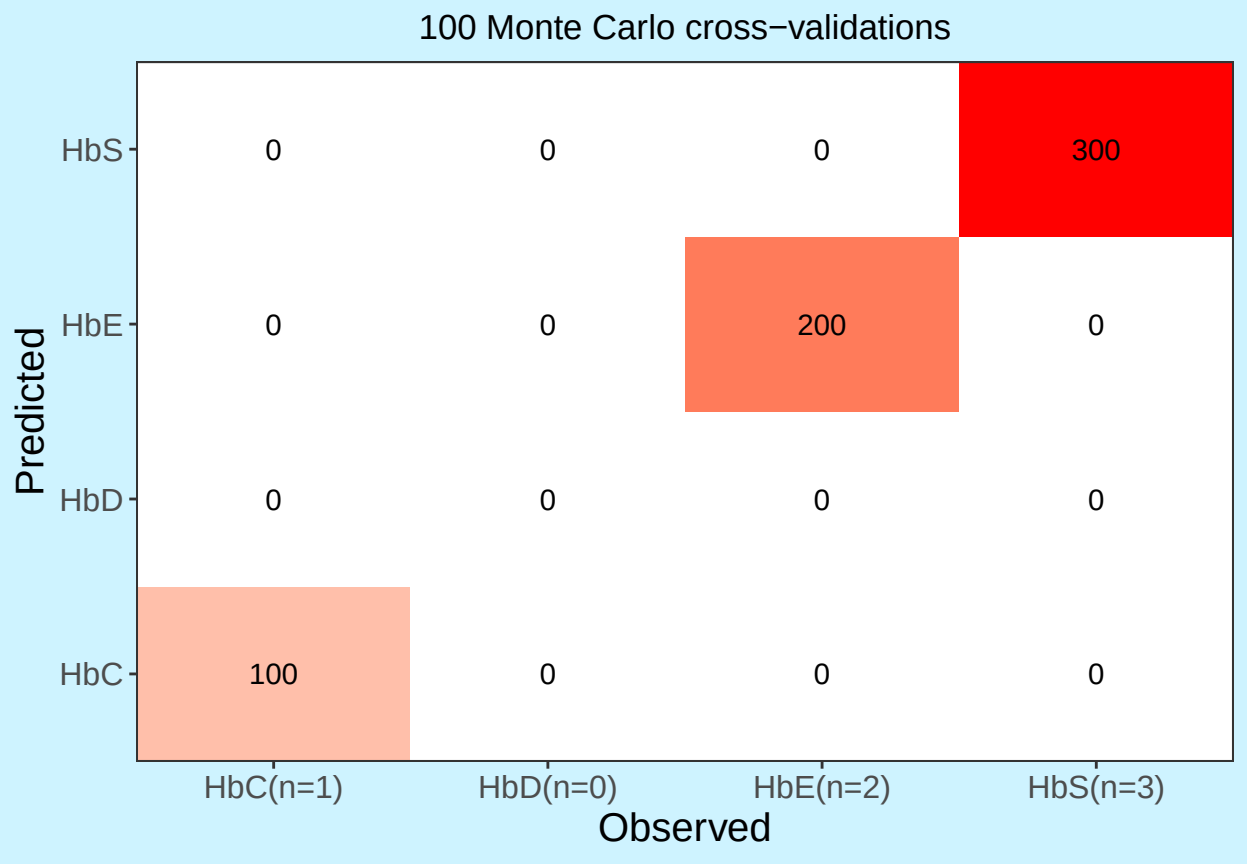

```
# save figure
saveRDS(f5a, file = './data_out/fig.5A.rds')
```

fig.5B

```
library(ggplot2)
library(tidyverse)
library(ggthemes)
library(ggpubr)
library(reshape2)
library(pROC)
library(caret)
library(cowplot)
library(gridExtra)
library(colorspace)

# Clear workspace
rm(list = ls())

# Load data
df <- read.csv('./data_in/variants.peptides.csv')

# Subset groups and classes
df <- df %>%
```

```r
  filter(!group %in% c('BT', 'CTR')) %>%
  filter(class == 'Plasma development')

# Reshape data
def <- dcast(group + sample ~ peptide, data = df, value.var = 'value') %>%
  select(-sample)
def$group <- factor(def$group, levels = c('HbC', 'HbD', 'HbE', 'HbS'))

# # run in parallel
# num_cores <- detectCores() - 1  # Use one less than the available cores
# cl <- makeCluster(num_cores)
# registerDoParallel(cl)
#
# results <- foreach(i = 1:100, .combine = rbind, .packages = c("caret",'tidyverse','pROC')) %dopar% {
#
#   # Split the data based on the outer fold
#   train_index <- createDataPartition(def$group, p = 0.75, list = FALSE)
#   trainData <- def[train_index, ]
#   testData <- def[-train_index, ]
#
#   tuneGrid <- expand.grid(mtry = seq(1,10,1))
#   control <- trainControl(method = "repeatedcv", number = 10, repeats = 1)
#
#   # Train the model with inner CV for tuning
#   model <- train(group ~ ., data = trainData,
#                  method = "rf",
#                  tuneGrid = tuneGrid,
#                  trControl = control,
#                  ntree = 5000)
#
#   # accuracy by default on the test
#   pred = predict(model, newdata = testData, type = 'raw')
#   acc_value_test = mean(pred == testData$group)
#
#   # auc
#   probs = predict(model, newdata = testData, type = 'prob')
#   true_label = testData$group
#   auc = as.numeric(auc(multiclass.roc(true_label, probs)))
#
#   # save data for test
#   ot_test = data.frame(pred, actual=testData$group, iteration=i,
#                        class='Development', acc=acc_value_test,
#                        auc=auc)
#
#   return(ot_test)
# }
# head(results)
#
# # Save results
# saveRDS(results, file = './data_out/fig.5B.data.rds')

# Load results
results <- readRDS('./data_out/fig.5B.data.rds')
```

```r
# Compute frequency table
freq <- results %>%
  filter(class == 'Plasma development') %>%
  count(class, pred, actual)

# Create base table
categories <- c("HbC", "HbD", "HbE", "HbS")
tab <- expand.grid(pred = categories, actual = categories)
tab <- merge(tab, freq, by = c('pred', 'actual'), all = TRUE)
tab$n[is.na(tab$n)] <- 0
tab$class[is.na(tab$class)] <- 'Plasma development'

# Rename x-axis labels
tab$actual <- as.character(tab$actual)
tab$actual[tab$actual == 'HbC'] = 'HbC(n=1)'
tab$actual[tab$actual == 'HbD'] = 'HbD(n=0)'
tab$actual[tab$actual == 'HbE'] = 'HbE(n=2)'
tab$actual[tab$actual == 'HbS'] = 'HbS(n=3)'

# Define colors
colors <- readRDS('./data_in/colors.rds')
light_color <- lighten(colors[1], amount = 0.7)

# Generate heatmap
f5b <- ggplot(tab, aes(actual, pred, fill = n)) +
  geom_tile() +
  geom_text(aes(label = n)) +
  scale_fill_gradient(low = 'white', high = 'red') +
  theme_bw() +
  theme(
    text = element_text(size = 15),
    legend.position = 'None',
    plot.background = element_rect(fill = lighten('lightblue', 0.6), color = NA),
    plot.title = element_text(hjust = 0.5, size = 13)
  ) +
  labs(x = 'Observed', y = 'Predicted') +
  ggtitle('100 Monte Carlo cross-validations')+
  scale_x_discrete(expand = c(0,0))+
  scale_y_discrete(expand = c(0,0))

f5b
```




```r
# Compute performance metrics
conf <- confusionMatrix(factor(results$pred), factor(results$actual))
acc <- paste('Accuracy=', format(round(conf$overall['Accuracy'], 3), nsmall = 3), sep = '')
sensitivity <- paste('Sensitivity=', format(round(mean(data.frame(conf$byClass)[,'Sensitivity']), 3), ns
specificity <- paste('Specificity=', format(round(mean(data.frame(conf$byClass)[,'Specificity']), 3), ns
f1_score <- paste('F1=', format(round(mean(conf$byClass[,"F1"]), 3), nsmall = 3), sep = '')
mcc <- paste('MCC=', format(round(mltools::mcc(preds = results$pred, actuals = results$actual), 3), nsma
auc <- paste('AUC=', format(round(aggregate(auc ~ class, data = results, FUN = mean)$auc, 3), nsmall = 3
tab_metrics <- data.frame(Metrics = c(auc, acc, sensitivity, specificity, f1_score, mcc))

# Generate table plot
table_plot <- tableGrob(tab_metrics, rows = NULL, theme = ttheme_default(
  core = list(fg_params = list(cex = 1, hjust = 0, x = 0.15), padding = unit(c(10, 2), "mm"), bg_params
  colhead = list(bg_params = list(fill = NA))
))


f5b_table <- ggdraw() + draw_plot(table_plot, x = 0, y = 0.48, height = 0.5)
f5b_background <- ggplot() +
  annotate("rect", xmin = 0, xmax = 1, ymin = 0, ymax = 1,
           fill = lighten("lightblue", 0.6), color = NA) +  # Background rectangle
  theme_void()  # Remove axes and gridlines

# save figure and table and background
saveRDS(f5b, file = './data_out/fig.5B.rds')
saveRDS(f5b_table, file = './data_out/fig.5B.table.rds')
```

```r
saveRDS(f5b_background, file = './data_out/fig.5B.table.background.rds')
```

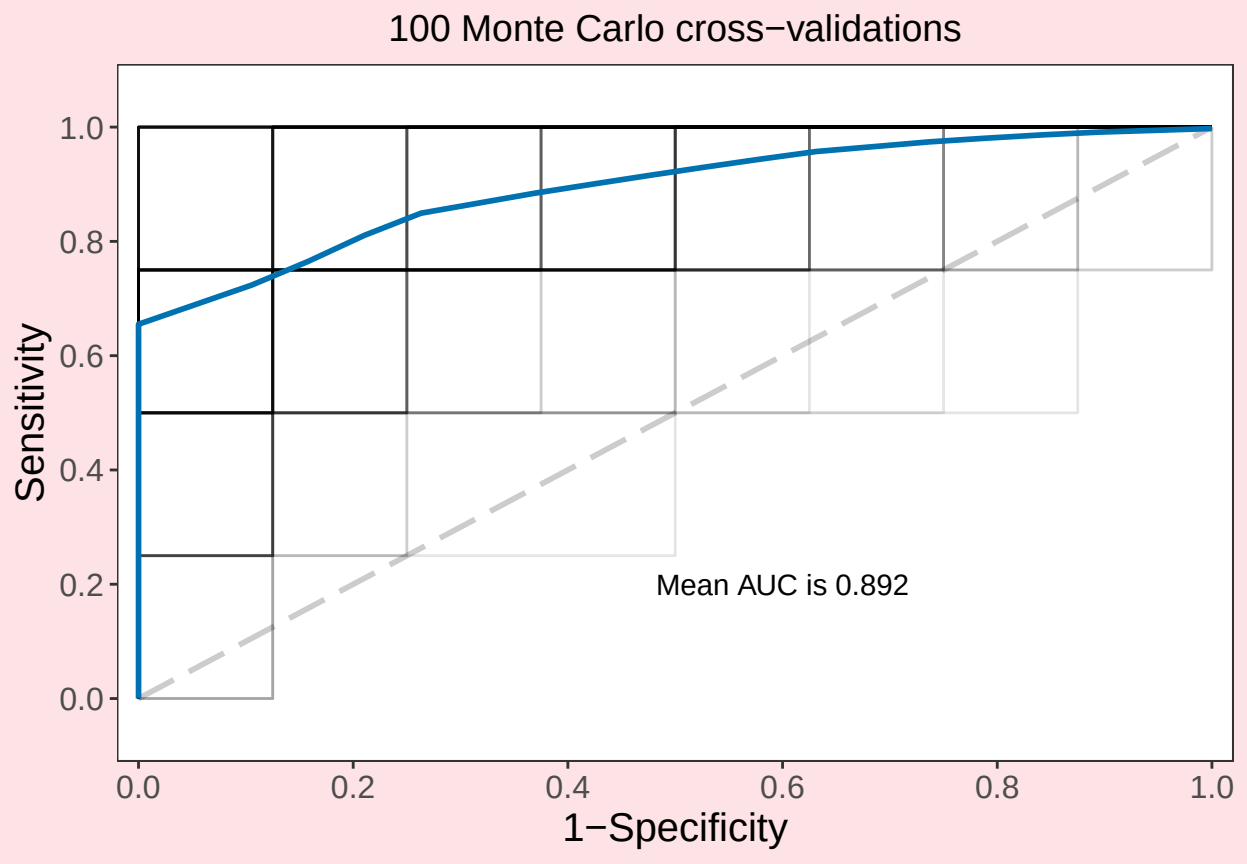
fig.5C

```r
library(ggplot2)
library(ggthemes)
library(reshape2)
library(pROC)
library(tidyverse)
library(caret)
library(foreach)
library(doParallel)

rm(list = ls())

# read data
df <- read.csv('./data_in/thalassemia.csv')

# subset for groups
df <- df[df$group %in% c('BT','CTR'), ]

df.blood <- df%>%
  filter(class == 'Plasma development')

# make data
def <- dcast(group+sample~peptide, data=df.blood,value.var = 'log') %>%
  select(-sample)
def$group <- factor(def$group, levels = c('CTR','BT'))

# # run in parallel
# num_cores <- detectCores() - 1  # Use one less than the available cores
# cl <- makeCluster(num_cores)
# registerDoParallel(cl)
#
# results <- foreach(i = 1:100, .combine = rbind, .packages = c("caret","pROC")) %dopar% {
#
#   # Split the data based on the outer fold
#   train_index <- createDataPartition(def$group, p = 0.75, list = FALSE)
#   trainData.blood <- def[train_index, ]
#   testData.blood <- def[-train_index, ]
#
#   tuneGrid <- expand.grid(mtry = seq(2,10,1))
#   control <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
#
#   # Train the model with inner CV for tuning
#   model <- train(group ~ ., data = trainData.blood,
#                  method = "rf",
#                  tuneGrid = tuneGrid,
#                  trControl = control,
#                  ntree = 5000)
#
```

```
#   # accuracy
#   pred_raw <- predict(model, newdata = testData.blood, type = 'raw')
#   acc_blood <- mean(pred_raw == testData.blood$group)
#
#   # auc
#   pred_prob <- predict(model, testData.blood, type = "prob")
#   roc <- roc(testData.blood$group, pred_prob[, "BT"])
#   auc <- as.numeric(auc(roc))
#
#   # collect data
#   ot_roc <- data.frame(iteration=i,
#                              class='Whole blood development',
#                              acc=acc,
#                              auc=auc,
#                              specificity=roc$specificities,
#                              sensitivity=roc$sensitivities,
#                              type='roc',
#                              prob=NA,
#                              pred=NA,
#                              actual=NA)
#   ot_prob <- data.frame(iteration=i,
#                               class='Whole blood development',
#                               acc=acc,
#                               auc=auc,
#                               specificity=NA,
#                               sensitivity=NA,
#                               type='prob',
#                               prob=pred_prob[, "BT"],
#                               pred=as.character(pred_label),
#                               actual=as.character(testData.blood$group))
#
#
#
#
#   # return data
#   tmp = rbind(ot_roc, ot_prob)
#
#   return(tmp)
#
# }

#saveRDS(results, file = './data_out/fig.5CD.data.rds')

# Load results
data_path <- './data_out/fig.5CD.data.rds'
results <- readRDS(data_path)
results$specificity <- 1 - results$specificity
results <- results[results$class == 'Plasma development' & results$type == 'roc',]

# Define common FPR values for interpolation
common_fpr <- seq(0, 1, length.out = 20)

# Make fpr
```

```r
results <- results %>%
  mutate(fpr = specificity)

# Compute max sensitivity at FPR = 0
max_sens_at_zero <- results %>%
  filter(fpr == 0) %>%
  group_by(iteration, class) %>%
  summarise(max_sens = max(sensitivity, na.rm = TRUE), .groups = "drop")

# Replace sensitivity when specificity at 0
results_fpr_0 <- results %>%
  left_join(max_sens_at_zero, by = c('iteration', 'class')) %>%
  mutate(sensitivity = ifelse(fpr == 0, max_sens, sensitivity)) %>%
  filter(fpr == 0) %>%
  select(-max_sens) %>%
  distinct()

# Modify results and interpolate
temp_results <- results %>% filter(fpr != 0) %>% rbind(results_fpr_0) %>% arrange(class, iteration, fpr
interp_df <- temp_results %>%
  group_by(iteration, class) %>%
  reframe(
    fpr = common_fpr,
    sens = approx(x = specificity, y = sensitivity, xout = common_fpr, rule = 1, ties = mean)$y
  )

# Compute summary statistics
summary_df <- interp_df %>%
  group_by(fpr, class) %>%
  summarise(
    mean_sens = mean(sens, na.rm = TRUE),
    .groups = "drop"
  )

# Add (0,0) starting point
summary_df <- rbind(
  data.frame(fpr = 0, class = 'Plasma development', mean_sens = 0),
  summary_df
)

# Compute mean AUC
auc_text <- results %>%
  group_by(iteration, class) %>%
  summarise(auc = unique(auc), .groups = "drop") %>%
  group_by(class) %>%
  summarise(mean_auc = mean(auc, na.rm = TRUE), .groups = "drop") %>%
  mutate(format = format(mean_auc, digits = 3, nsmall = 3))

# Load colors
colors <- readRDS('./data_in/colors.rds')

# Create diagonal reference line
diagonal_df <- summary_df %>%
```

```
  group_by(class) %>%
  summarise(fpr_start = 0, sens_start = 0, fpr_end = 1, sens_end = 1, .groups = "drop")

results = results %>%
  arrange(fpr, sensitivity)

# Generate ROC plot
f5c <- ggplot() +
  geom_segment(data = diagonal_df,
               aes(x = fpr_start, y = sens_start, xend = fpr_end, yend = sens_end),
               linetype = "longdash", size = 1, color = "gray80") +
  geom_step(data = results,
            aes(x = fpr, y = sensitivity, group = interaction(iteration, class)),
            alpha = 0.1, size = 0.5) +
  geom_line(data = summary_df,
            aes(x = fpr, y = mean_sens, color = class),
            size = 1) +
  scale_x_continuous(breaks = seq(0, 1, 0.2), expand = c(0.01, 0.01)) +
  scale_y_continuous(breaks = seq(0, 1, 0.2), expand = c(0.01, 0.1)) +
  theme_bw() +
  theme(panel.grid = element_blank(),
        text = element_text(size = 15),
        axis.ticks = element_line(),
        strip.background = element_rect(fill = "gray80", color = 'white'),
        legend.position = 'None',
        plot.title = element_text(hjust = 0.5, size = 14),
        plot.background = element_rect(fill = lighten('lightpink', 0.6), color = NA),
        legend.background = element_rect(fill = lighten('lightpink', 0.6), color = NA)) +
  guides(color = guide_legend(nrow = 1)) +
  scale_color_manual('Group', values = colors) +
  scale_fill_manual('Group', values = colors) +
  labs(x = '1-Specificity', y = 'Sensitivity', title = "100 Monte Carlo cross-validations") +
  geom_text(data = auc_text, aes(x = 0.6, y = 0.2, label = paste("Mean AUC is", format)))

f5c
```

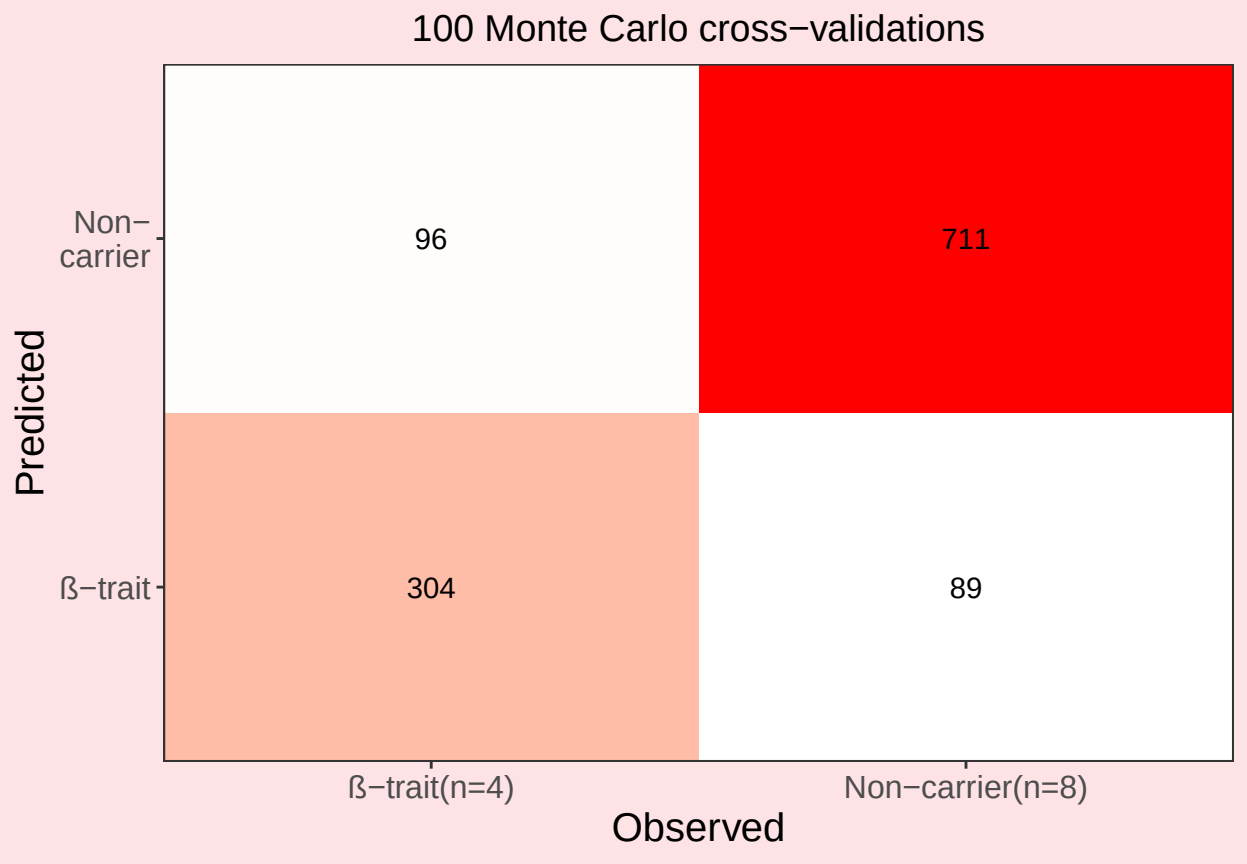



```
# save figure
saveRDS(f5c, file = './data_out/fig.5C.rds')
```

fig.5D

```
# clear environment
rm(list = ls())

# Load libraries
library(ggplot2)
library(tidyverse)
library(colorspace)
library(caret)
library(gridExtra)
library(cowplot)

# Read in data
results <- readRDS('./data_out/fig.5CD.data.rds')
results$pred[results$pred == 'BT'] <- ' -trait'
results$actual[results$actual == 'BT'] <- ' -trait'
results$pred[results$pred == 'CTR'] <- 'Non-\ncarrier'
results$actual[results$actual == 'CTR'] <- 'Non-\ncarrier'

freq <- results %>%
```

```
  filter(type == 'prob' & class == 'Plasma development') %>%
  count(class, pred, actual)

# Create base table
tab <- expand.grid(pred = c(' -trait', 'Non-\ncarrier'), actual = c(' -trait', 'Non-\ncarrier'))
tab <- merge(tab, freq, by = c('pred', 'actual'), all = TRUE)
tab$n[is.na(tab$n)] <- 0

# Rename x-axis labels
tab$actual = as.character(tab$actual)
tab$actual[tab$actual == ' -trait'] = ' -trait(n=4)'
tab$actual[tab$actual == 'Non-\ncarrier'] = 'Non-carrier(n=8)'

#order
tab$actual = factor(tab$actual, levels = c(" -trait(n=4)","Non-carrier(n=8)"))

# Generate heatmap
f5d <- ggplot(tab, aes(actual, pred, fill = n)) +
  geom_tile() +
  geom_text(aes(label = n)) +
  scale_fill_gradient(low = 'white', high = 'red') +
  theme_bw() +
  theme(text = element_text(size = 15),
        legend.position = 'None',
        plot.background = element_rect(fill = lighten('lightpink', 0.6), color = NA),
        plot.title = element_text(hjust = 0.5, size = 14)) +
  labs(x = 'Observed', y = 'Predicted', title = "100 Monte Carlo cross-validations")+
  scale_x_discrete(expand = c(0,0))+
  scale_y_discrete(expand = c(0,0))

f5d
```




```
## calculate precision, recall, f1, MCC
# Load results
data_path <- './data_out/fig.5CD.data.rds'
df <- readRDS(data_path)
df <- df %>%
  filter(type == 'prob') %>%
  select(class, pred, actual, iteration)

# Calculate for whole blood
df <- df %>% filter(class == 'Plasma development')
conf <- confusionMatrix(factor(df$pred), factor(df$actual), positive = 'BT')

# Compute evaluation metrics
acc <- paste('Accuracy=', format(round(conf$overall['Accuracy'], 3), nsmall = 3), sep = '')
sensitivity <- paste('Sensitivity=', format(round(conf$byClass['Sensitivity'], 3), nsmall = 3))
specificity <- paste('Specificity=', format(round(conf$byClass['Specificity'], 3), nsmall = 3))
f1_score <- paste('F1=', format(round(conf$byClass['F1'], 2), nsmall = 3), sep = '')
mcc <- paste('MCC=', format(round(mltools::mcc(preds = df$pred, actuals = df$actual), 2), nsmall = 3), s

# Create results table
tab <- data.frame(Metrics = c(acc, sensitivity, specificity, f1_score, mcc))

# Generate table plot
table_plot <- tableGrob(tab, rows = NULL,
                        theme = ttheme_default(
                          core = list(fg_params = list(cex = 1, hjust = 0, x = 0.15),
```

```
                                    padding = unit(c(10, 2), "mm"),
                                    bg_params = list(fill = NA)),
                        colhead = list(bg_params = list(fill = NA))
                      ))

f5d_table <- ggdraw() + draw_plot(table_plot, x = 0, y = 0.5, height = 0.5)
f5d_background <- ggplot() +
  annotate("rect", xmin = 0, xmax = 1, ymin = 0, ymax = 1,
           fill = lighten("lightpink", 0.6), color = NA) +  # Background rectangle
  theme_void()  # Remove axes and gridlines

# save figure
saveRDS(f5d, file = './data_out/fig.5D.rds')
saveRDS(f5d_table, file = './data_out/fig.5D.table.rds')
saveRDS(f5d_background, file = './data_out/fig.5D.table.background.rds')
```

Merge fig. 5A-D

```
rm(list = ls())

library(cowplot)
library(ggplot2)
library(patchwork)


# load figures
f5a <- readRDS('./data_out/fig.5A.rds')
f5b <- readRDS('./data_out/fig.5B.rds')
f5c <- readRDS('./data_out/fig.5C.rds')
f5d <- readRDS('./data_out/fig.5D.rds')

f5b_table <- readRDS('./data_out/fig.5B.table.rds')
f5b_table_background <- readRDS('./data_out/fig.5B.table.background.rds')

f5d_table <- readRDS('./data_out/fig.5D.table.rds')
f5d_table_background <- readRDS('./data_out/fig.5D.table.background.rds')



p=ggdraw()+
  draw_plot(f5a, x=0, y=0.5, width = 0.5, height = 0.5)+
  draw_plot(f5b, x=0.5, y=0.5, width = 0.35, height = 0.5)+
  draw_plot(f5b_table_background, x=0.84, y=0.472, width = 0.16, height = 0.6)+
  draw_plot(f5b_table, x=0.87, y=0.5, width = 0.1, height = 0.5)+
  draw_plot(f5c, x=0, y=0, width = 0.5, height = 0.5)+
  draw_plot(f5d, x=0.5, y=0, width = 0.35, height = 0.5)+
  draw_plot(f5d_table_background, x=0.84, y=-0.025, width = 0.16, height = 0.55)+
  draw_plot(f5d_table, x=0.87, y=0, width = 0.1, height = 0.5) # save 1200*850
```

```r
# Create a side title as a separate plot
side_title1 <- ggplot() +
  annotate("text", x = 0, y = 0, label = "Hb variants classifier",
           angle = 90, size = 5) +
  theme_void()

# Create a side title as a separate plot
side_title2 <- ggplot() +
  annotate("text", x = 0, y = 0, label = " -thalassemia trait classifier",
           angle = 90, size = 5) +
  theme_void()

plot_grid(side_title1 / side_title2, p, rel_widths = c(0.05,1))
```

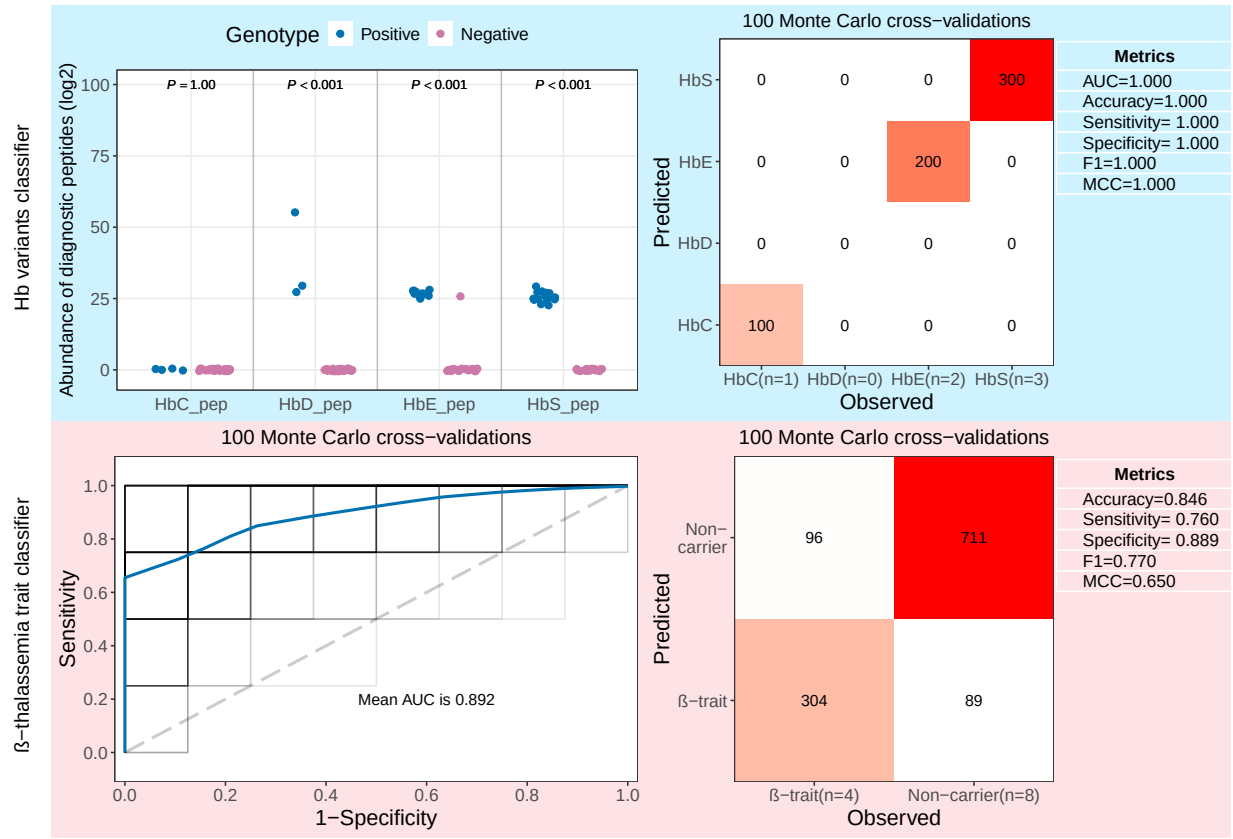


fig. S1

```r
library(ggplot2)
library(tidyverse)
library(ggthemes)
library(ggpubr)
library(reshape2)
library(pROC)
library(caret)
library(gridExtra)
```

```r
library(grid)
library(gtable)
library(cowplot)
library(mltools)

# Clear workspace
rm(list = ls())

# Load data
data_path <- './data_in/variants.peptides.csv'
df <- read.csv(data_path)

# Remove control groups
df <- df %>% filter(!group %in% c('CTR', 'BT'))

# Create binary groups
df$binary <- ifelse(df$group == df$variant, 'Positive', 'Negative')
df$binary <- factor(df$binary, levels = c('Positive','Negative'))

# Define color palette
colors <- readRDS('./data_in/colors.rds')

pos_jitter <- position_jitterdodge(jitter.width = 0.4, jitter.height = 0.1)
point_size <- 1.5

# Remove variants with all zero peptide values
for (i in unique(df$class)) {
  for (j in unique(df$peptide)) {
    sbs <- df %>% filter(class == i & peptide == j)
    if (sum(sbs$value) == 0) {
      df <- df %>% filter(!(class == i & peptide == j))
    }
  }
}

# Generate plots
plot_variants <- function(df_subset, title) {
  ggplot(df_subset, aes(variant, value, color = binary)) +
    geom_point(position = pos_jitter, size = point_size) +
    labs(y = 'Abundance of diagnostic peptides (log2)', x = 'Variants') +
    ggtitle(title) +
    theme_bw() +
    theme(text = element_text(size = 10),
          strip.text = element_text(size = 6),
          axis.text.x = element_blank(),
          axis.ticks.x.bottom = element_blank(),
          axis.title.x.bottom = element_blank(),
          plot.title = element_text(hjust = 0.5),
          panel.grid.minor = element_blank(),
          legend.position = 'None') +
    facet_wrap(variant ~ seq, scale = 'free_x') +
    scale_x_discrete(expand = c(0, 0)) +
    scale_color_manual('Genotype', values = colors)
```

```
}

p1 <- plot_variants(df %>% filter(class == 'Development'), 'RBC development')
p2 <- plot_variants(df %>% filter(class == 'Validation'), 'RBC validation')+
  theme(legend.position = 'right',
        axis.title.y.left = element_blank())
p3 <- plot_variants(df %>% filter(class == 'Whole blood development'), 'Whole blood development')
p4 <- plot_variants(df %>% filter(class == 'Plasma development'), 'Plasma development')+
  theme(legend.position = 'right',
        axis.title.y.left = element_blank())

# Combine plots
plot_grid(p1, p2, p3, p4, nrow = 2)  # Save as 1500x900
```