SAMPLE R SCRIPT

Generation diagnostic plots for IsoSeq3 processed data

Eva Estevan Morió

Contents

First steps

Loading necessary packages

```
library(tidyverse)
library(cowplot)
library(scales)
library(RColorConesa)
library(UpSetR)
library(optparse)
library(funprog)
library(ggplotify)
library(ggrepel)
```

Setting theme parameters

```
sq_theme <- theme_classic(base_family = "Helvetica") +
  theme(plot.title = element_text(lineheight=.4, size=15, hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5)) +
  theme(plot.margin = unit(c(2.5,1,1,1), "cm")) +
  theme(axis.line.x = element_line(color="black", size = 0.4),
        axis.line.y = element_line(color="black", size = 0.4)) +
  theme(axis.title.x = element_text(size=14),
        axis.text.x  = element_text(size=14),
        axis.title.y = element_text(size=14),
        axis.text.y  = element_text(vjust=0.5, size=13) ) +
  theme(legend.text = element_text(size = 12),
        legend.title = element_text(size=14),
        legend.key.size = unit(0.5, "cm"))


theme_set(sq_theme)
```

Setting colors for each filtering combination: FSM FSM_dist RM RM_dist

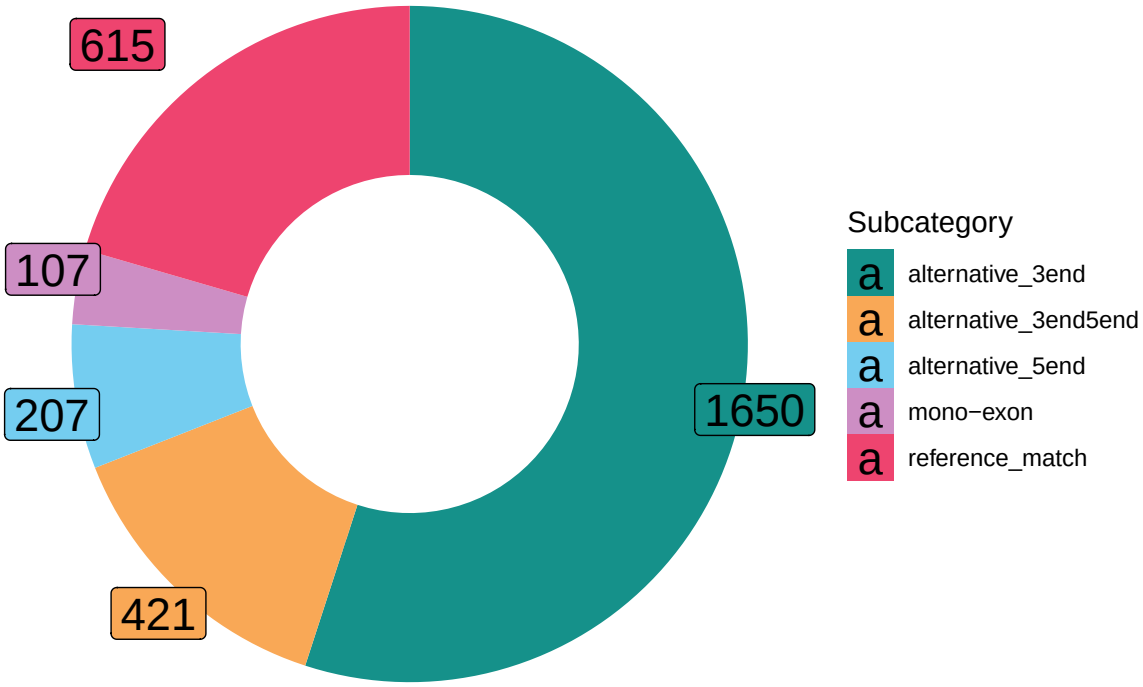
```
filt_palette=c(IsoSeq_RM ="lightslateblue", IsoSeq_FSM="darkorange",IsoSeq_RM_dist="skyblue",
               IsoSeq_FSM_dist="gold",  Common="darkcyan")


filt_palette_un=c(Unique_IsoSeq_RM ="lightslateblue", Unique_IsoSeq_FSM="darkorange"
               ,Unique_IsoSeq_RM_dist="skyblue",
               Unique_IsoSeq_FSM_dist="gold",  Common="darkcyan")
```

Isoseq3 data analysis

TP FSM subcategories

First, the TP FSM set file is loaded to generate a pie plot with the percentages of the subcategories.

```
#load TP FSM file
TP_GM<-read.csv('SQ3_MLresults/GM12878/GM12878_input/TP_FSM_GM12878.txt', header=FALSE)

#select de isoform
colnames(TP_GM)<-'isoform'

#load the input classification file
classif_GM12878<-read_tsv('SQ3_MLresults/GM12878/GM12878_input/GM12878_classification.txt')

#select the columns isoform and subcategory
classif_GM12878<-classif_GM12878 %>% select(isoform,subcategory)

#intersect both tables
TP_GM<-left_join(TP_GM,classif_GM12878)

#calculate percentages of each subcategory
cat<-TP_GM %>% group_by(subcategory) %>% summarize(category_count = n()) %>%
  mutate(percentin=category_count/sum(category_count)) %>%
  mutate(suma=cumsum(category_count))

# Compute percentages
cat$fraction <- cat$category_count / sum(cat$category_count)

# Compute the cumulative percentages (top of each rectangle)
cat$ymax <- cumsum(cat$fraction)

# Compute the bottom of each rectangle
cat$ymin <- c(0, head(cat$ymax, n=-1))

#label positions
cat$labelPosition <- (cat$ymax + cat$ymin) / 2

# Compute a good label
cat$label <- paste0(cat$category_count)

#PIE PLOT
tp_pie<- ggplot(cat, aes(ymax=ymax, ymin=ymin, xmax=4, xmin=3, fill=subcategory)) +
  geom_rect()+
  RColorConesa::scale_fill_conesa(palette = "complete")+
  coord_polar(theta="y") +
  xlim(c(2, 4)) +
  geom_label_repel(data = cat,x=4.5, aes(y=labelPosition, label=label), size=6) +
  guides(fill = guide_legend(title = "Subcategory")) +
  theme_void()
```


615
107
207
421
1650
Subcategory
alternative_3end
alternative_3end5end
alternative_5end
mono−exon
reference_match

Loading output classification files

Then, the four classification files generated by the Machine Learning filter are loaded. For this purpose, each one is given a label.

```
# first output to compare
path1 <- "SQ3_MLresults/GM12878/GM12878_output_withcol"
label1 <- "IsoSeq_RM_dist"
files1 <- dir(path1)

# second output to compare
path2 <- "SQ3_MLresults/GM12878/GM12878_output_FSM_withcol"
label2 <- "IsoSeq_FSM_dist"
files2 <- dir(path2)

# third output to compare
path3 <- "SQ3_MLresults/GM12878/GM12878_output_FSM"
label3 <- "IsoSeq_FSM"
files3 <- dir(path3)

# fourth output to compare
path4 <- "SQ3_MLresults/GM12878/GM12878_output"
label4 <- "IsoSeq_RM"
files4 <- dir(path4)


#Load first classification files
classif1_file <- files1[str_detect(files1, "MLresult_classification")]
classif_1 <- read_tsv(paste0(path1, "/", classif1_file))

classif_1 <- classif_1 %>%
  mutate(structural_category = factor(structural_category) %>%
           fct_infreq() %>%
           fct_recode(ISM = "incomplete-splice_match",
                      FSM = "full-splice_match",
                      NNC = "novel_not_in_catalog",
                      NIC = "novel_in_catalog",
                      Intergenic = "intergenic",
                      Antisense = "antisense",
                      Genic = "genic",
                      Fusion = "fusion",
                      Genic_intron = "genic_intron") %>%
           fct_relevel(c("FSM", "ISM", "NIC", "NNC",
                         "Genic", "Antisense", "Fusion",
                         "Intergenic", "Genic_intron")))


# Load second classification file
classif2_file <- files2[str_detect(files2, "MLresult_classification")]
classif_2 <- read_tsv(paste0( path2, "/", classif2_file))

classif_2 <- classif_2 %>%
  mutate(structural_category = factor(structural_category) %>%
           fct_infreq() %>%
```

```
          fct_recode(ISM = "incomplete-splice_match",
                     FSM = "full-splice_match",
                     NNC = "novel_not_in_catalog",
                     NIC = "novel_in_catalog",
                     Intergenic = "intergenic",
                     Antisense = "antisense",
                     Genic = "genic",
                     Fusion = "fusion",
                     Genic_intron = "genic_intron") %>%
          fct_relevel(c("FSM", "ISM", "NIC", "NNC",
                        "Genic", "Antisense", "Fusion",
                        "Intergenic", "Genic_intron")))


# Load third classification file
classif3_file <- files3[str_detect(files3, "MLresult_classification")]
classif_3 <- read_tsv(paste0(path3, "/", classif3_file))

classif_3 <- classif_3 %>%
  mutate(structural_category = factor(structural_category) %>%
          fct_infreq() %>%
          fct_recode(ISM = "incomplete-splice_match",
                     FSM = "full-splice_match",
                     NNC = "novel_not_in_catalog",
                     NIC = "novel_in_catalog",
                     Intergenic = "intergenic",
                     Antisense = "antisense",
                     Genic = "genic",
                     Fusion = "fusion",
                     Genic_intron = "genic_intron") %>%
          fct_relevel(c("FSM", "ISM", "NIC", "NNC",
                        "Genic", "Antisense", "Fusion",
                        "Intergenic", "Genic_intron")))

# Load fourth classification file
classif4_file <- files4[str_detect(files4, "MLresult_classification")]
classif_4 <- read_tsv(paste0(path4, "/", classif4_file))

classif_4 <- classif_4 %>%
  mutate(structural_category = factor(structural_category) %>%
          fct_infreq() %>%
          fct_recode(ISM = "incomplete-splice_match",
                     FSM = "full-splice_match",
                     NNC = "novel_not_in_catalog",
                     NIC = "novel_in_catalog",
                     Intergenic = "intergenic",
                     Antisense = "antisense",
                     Genic = "genic",
                     Fusion = "fusion",
                     Genic_intron = "genic_intron") %>%
          fct_relevel(c("FSM", "ISM", "NIC", "NNC",
                        "Genic", "Antisense", "Fusion",
                        "Intergenic", "Genic_intron")))
```

Variable importance plot

Importance files (in which the values of importance of each variable are stored), are loaded.

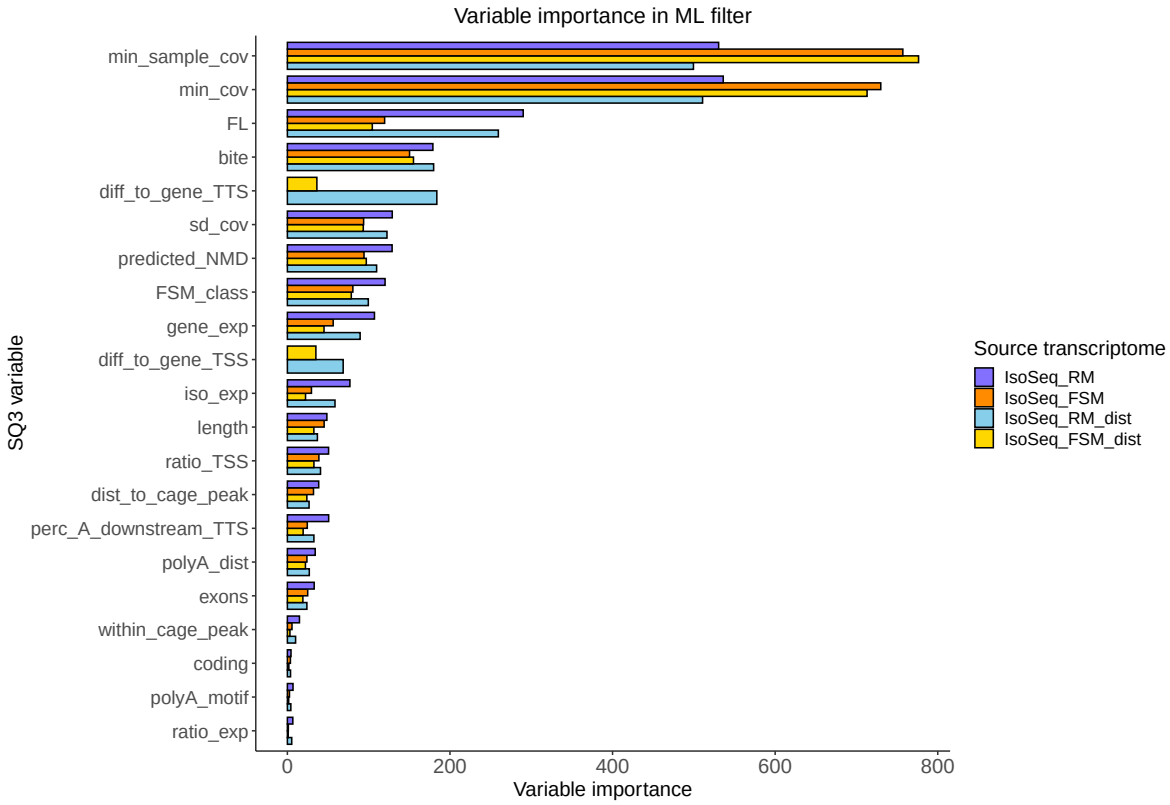
```
imp1_file <- files1[str_detect(files1, "variable-importance")]
imp_1 <- read_tsv(paste0( path1, "/", imp1_file),
                  col_names = c("variable", "importance"))


imp2_file <- files2[str_detect(files2, "variable-importance")]
imp_2 <- read_tsv(paste0( path2, "/", imp2_file),
                  col_names = c("variable", "importance"))

imp3_file <- files3[str_detect(files3, "variable-importance")]
imp_3 <- read_tsv(paste0( path3, "/", imp3_file),
                  col_names = c("variable", "importance"))

imp4_file <- files4[str_detect(files4, "variable-importance")]
imp_4 <- read_tsv(paste0( path4, "/", imp4_file),
                  col_names = c("variable", "importance"))
```

The four importance tables are now combined.

```
imp_combined <- bind_rows(list(imp_1 = imp_1, imp_2 = imp_2,
                               imp_3 = imp_3, imp_4 = imp_4),
                          .id = "source_transcriptome")

imp_combined <- imp_combined %>%
  mutate(variable = fct_reorder(variable, importance),
         source_transcriptome = factor(source_transcriptome,
                                       levels = c("imp_1", "imp_2",
                                                  "imp_3", "imp_4"),
                                       labels = c(label1, label2,
                                                  label3, label4)))
```

And the plot is generated.

```
imp_plot <- ggplot(imp_combined,
                   aes(x = variable, y = importance, fill=source_transcriptome)) +
  ggtitle("Variable importance in ML filter") +
  geom_bar(width = 0.8, color = "black",
           stat = "identity", position = "dodge") +
  labs(x = "SQ3 variable", y = "Variable importance") +
  scale_fill_manual(values = filt_palette[1:4], name="Source transcriptome") +
  coord_flip()
```


Variable importance in ML filter
min_sample_cov
min_cov
FL
bite
diff_to_gene_TTS
sd_cov
predicted_NMD
FSM_class
gene_exp
diff_to_gene_TSS
iso_exp
length
ratio_TSS
dist_to_cage_peak
perc_A_downstream_TTS
polyA_dist
exons
within_cage_peak
coding
polyA_motif
ratio_exp
SQ3 variable
0
200
400
600
800
Variable importance
Source transcriptome
IsoSeq_RM
IsoSeq_FSM
IsoSeq_RM_dist
IsoSeq_FSM_dist

Number of artifacts

First, we filter the transcripts considered artifacts in each classification file and save them in a new table.

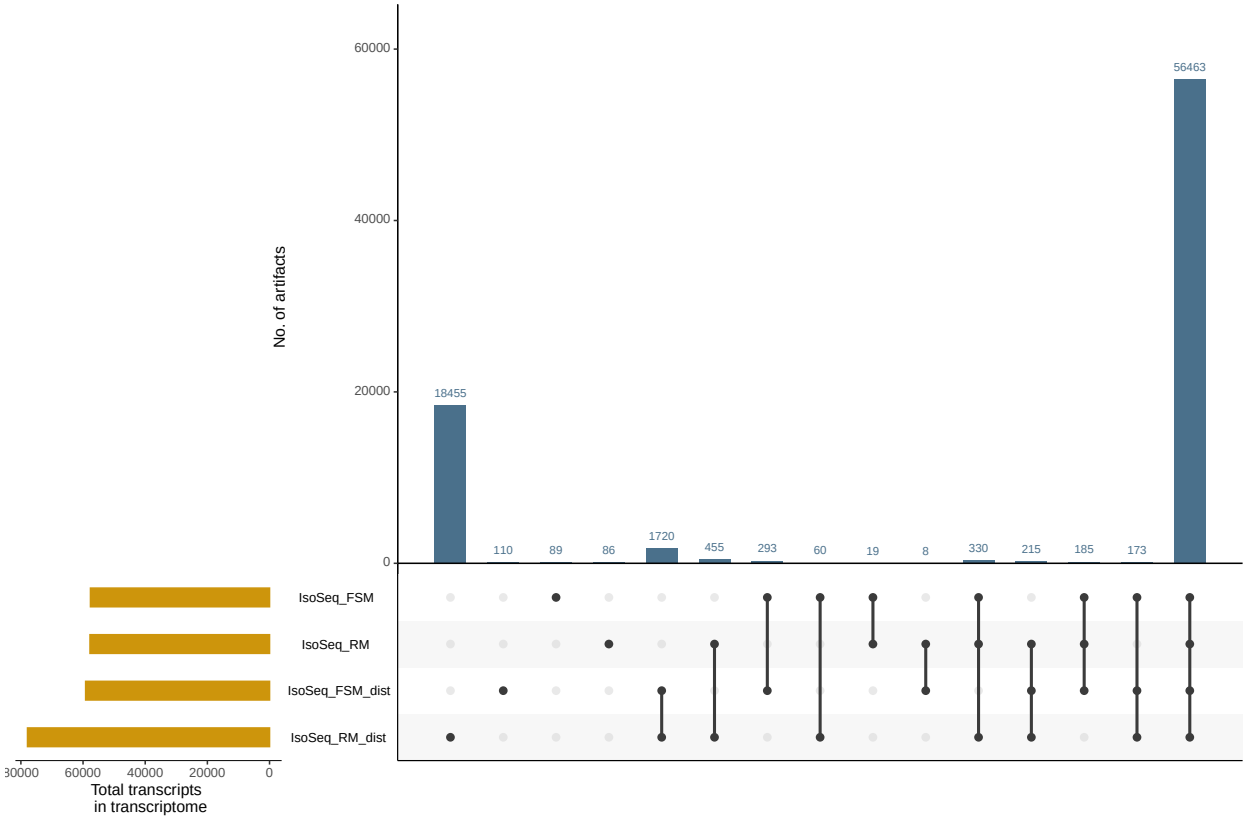
```
artifacts_1 <- filter(classif_1, filter_result == "Artifact")
artifacts_2 <- filter(classif_2, filter_result == "Artifact")
artifacts_3 <- filter(classif_3, filter_result == "Artifact")
artifacts_4 <- filter(classif_4, filter_result == "Artifact")
```

Then, we create a list of these artifacts (iso_list) and add the names of the filtering combination they come from.

```
iso_list <- list(classif_1 = artifacts_1 %>% select(isoform) %>% deframe,
                 classif_2 = artifacts_2 %>% select(isoform) %>% deframe,
                 classif_3 = artifacts_3 %>% select(isoform) %>% deframe,
                 classif_4 = artifacts_4 %>% select(isoform) %>% deframe)

names(iso_list) <- c(label1, label2, label3, label4)
```

Finally, we generate a plot to represent the intersections of the four filtering combination compared.

```
intersections <- fromList(iso_list)

upset <- upset(intersections, main.bar.color = "skyblue4",
      mainbar.y.label = "No. of artifacts",
      sets.bar.color = "darkgoldenrod3",
      sets.x.label = "Total transcripts \n in transcriptome",
      point.size = 2.5, line.size = 1, text.scale = c(1.3, 1.3, 1.3, 1.3, 1.3, 1.3))
```


60000
40000
20000
0
No. of artifacts
18455
110
89
86
1720
455
293
60
19
8
330
215
185
173
56463
IsoSeq_FSM
IsoSeq_RM
IsoSeq_FSM_dist
IsoSeq_RM_dist
30000
60000
40000
20000
0
Total transcripts
in transcriptome

IsoSeq3 first comparison: FSM_dist vs RM_dist

Categories plots

First, we create a list of artifacts of both filtering combinations (artifact_list) and add the name of the source transcriptome.

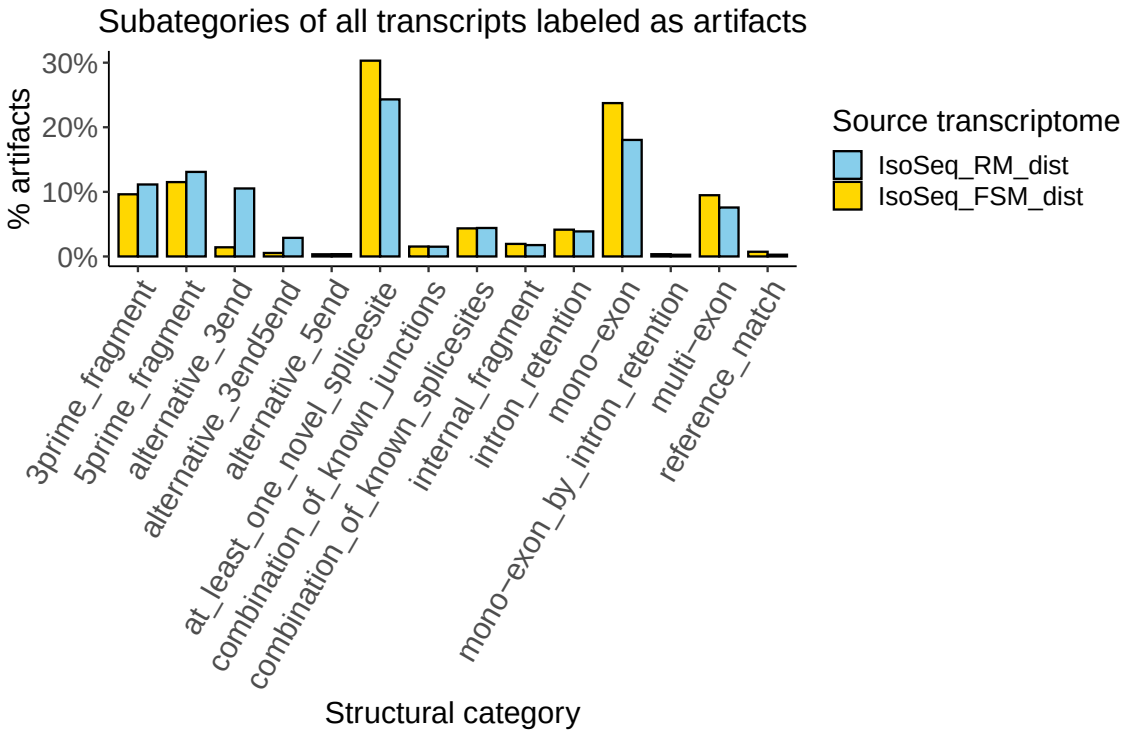
```
artifact_list <- list(artifacts_1, artifacts_2)
names(artifact_list) <- c(label1, label2)

artifacts_all <- bind_rows(artifact_list,
                           .id = "source_transcriptome")
```

Then we calculate the percentage of artifacts filtered by each combination for each subcategory (artifact_summary).

```
artifact_summary <- artifacts_all %>%
  group_by(source_transcriptome, subcategory) %>%
  summarize(category_count = n()) %>%
  mutate(percent = category_count/sum(category_count))
```

And plot it.

```
cat_all <- ggplot(artifact_summary,
                  aes(x = subcategory, y = percent)) +
  ggtitle("Subategories of all transcripts labeled as artifacts") +
  geom_bar(aes(fill = source_transcriptome), stat = "identity", position = "dodge",
           width = 0.8, color = "black") +
  labs(x = "Structural category", y = "% artifacts") +
  scale_y_continuous(labels = scales::percent_format()) +
  scale_fill_manual(values=filt_palette[3:4], name="Source transcriptome")+
  theme(axis.text.x = element_text(angle = 60, hjust = 1))
```



We extract those unique artifacts from each classification file and we make a list with all of them.

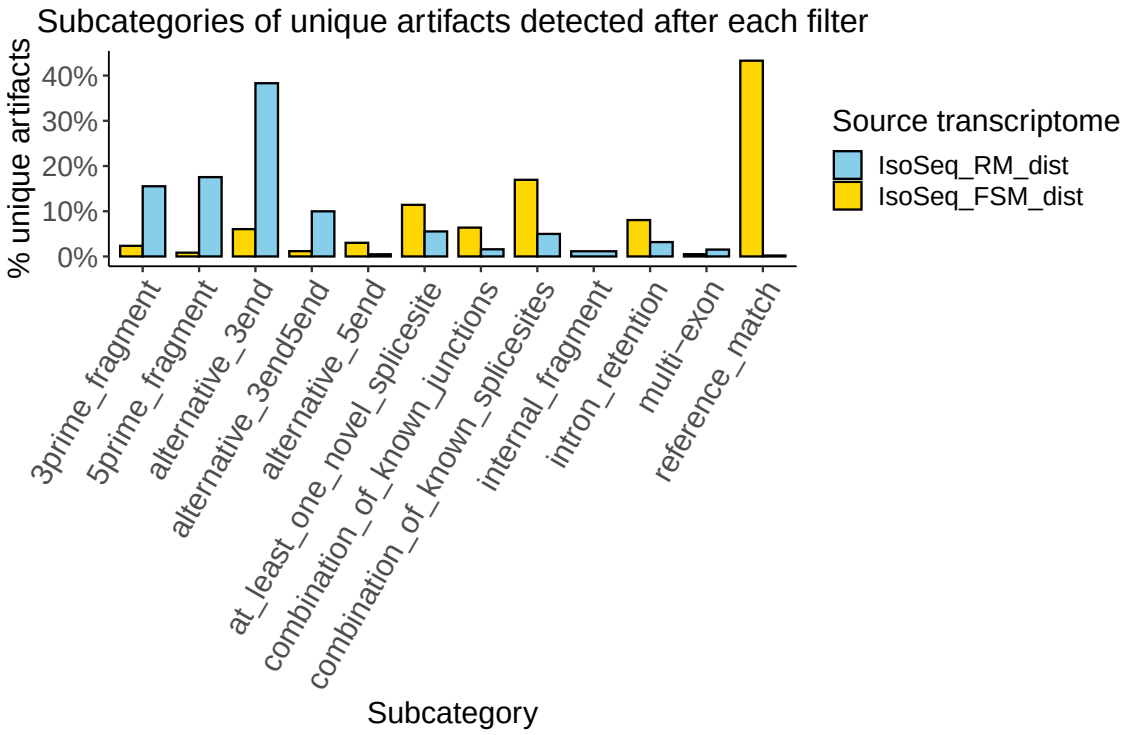
```
unique_1 <- filter(classif_1, filter_result == "Artifact" &
                     !(isoform %in% intersect(iso_list[[1]], iso_list[[2]])))

unique_2 <- filter(classif_2, filter_result == "Artifact" &
                     !(isoform %in% intersect(iso_list[[1]], iso_list[[2]])))

unique_list <- list(unique_1, unique_2)
names(unique_list) <- c(label1, label2)

artifacts_unique <- bind_rows(unique_list,
                              .id = "source_transcriptome")
```

And calculate the percentage of artifacts filtered by each combination (unique_summary).

```
unique_summary <- artifacts_unique %>%
  group_by(source_transcriptome, subcategory) %>%
  summarize(category_count = n()) %>%
  mutate(percent = category_count/sum(category_count))
```

To plot it.

```
cat_un <- ggplot(unique_summary,
                 aes(x = subcategory, y = percent,
                     fill = source_transcriptome)) +
  ggtitle("Subcategories of unique artifacts detected after each filter") +
  geom_bar(stat = "identity", position = "dodge",
           width = 0.8, color = "black") +
  labs(x = "Subcategory", y = "% unique artifacts") +
  scale_y_continuous(labels = percent_format()) +
  scale_fill_manual(values = filt_palette[3:4], name="Source transcriptome")+
    theme(axis.text.x = element_text(angle = 60, hjust = 1))
```



In order to see this information all together, with common and unique artifacts in one plot, we add a column to classification to artifacts_all list including whether the artifact is unique or common.

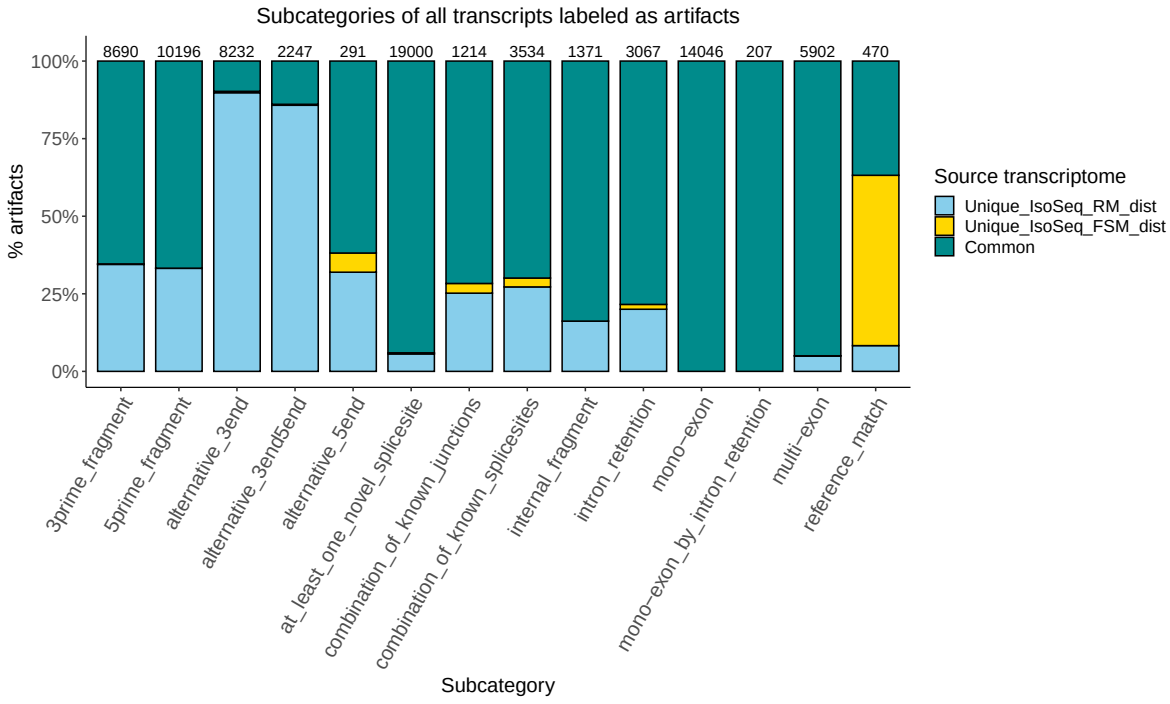
```
artifacts_all <- artifacts_all %>%
  mutate(artifact_type = case_when(isoform %in% artifacts_unique$isoform == TRUE ~ "Unique",
                                   isoform %in% artifacts_unique$isoform == FALSE ~ "Common"),
         artifact_lab = if_else(artifact_type == "Unique",
                                true = paste0(artifact_type, "_", source_transcriptome),
                                false = "Common"))
```

Then, we calculate again the percentages.

```
artifact_sum <- artifacts_all %>%
  dplyr::filter(!(source_transcriptome == label2 &
                    artifact_lab == 'Common')) %>%
  group_by(source_transcriptome,  subcategory, artifact_lab) %>%
  summarize(category_count = n()) %>%
  group_by(subcategory) %>%
  mutate(percent = category_count/sum(category_count),
         suma=cumsum(category_count)) %>%
  arrange(desc(artifact_lab))
```

To finally plot the percentages of categories of all transcripts labeled as artifacts.

```
cat_stack <- ggplot(artifact_sum,
                    aes(x =  subcategory, y = percent)) +
  ggtitle("Subcategories of all transcripts labeled as artifacts") +
  geom_bar(aes( fill =artifact_lab), stat = "identity", position = "stack",
           width = 0.8, color = "black") +
  labs(x = "Subcategory", y = "% artifacts") +
  scale_y_continuous(labels = scales::percent_format()) +
  scale_fill_manual(values = filt_palette_un[3:5], name="Source transcriptome")+
  geom_text(aes(x=subcategory,y=1.03, label=suma),size=4,
            check_overlap=TRUE)+
  theme(axis.text.x = element_text(angle = 60, hjust = 1))
```



Variables comparison plots

For comparing the values of each variable, we make a function (compare_artifacts), that takes the artifacts_all table, each variable and the imp_combined table to return a set of plots.

```
compare_artifacts <- function(classification,
                              var, importance){

  require(ggplot2)
  require(magrittr)

  # Select variable for evaluation plot
  var_df <- classification %>%
    dplyr::select(structural_category,
                  source_transcriptome,
                  artifact_type, artifact_lab,
                  dplyr::all_of(var))

  # Explicitly remove NAs
  var_df <- var_df %>%
    dplyr::filter(!(is.na(var)))

  # Get variable column info (class, name)
  var_type <- purrr::map_chr(var_df, class)
  var_name <- names(var_type[5])

  # Rename variable column to handle during plotting
  var_df <- var_df %>% dplyr::rename(variable = var)

  # Obtain labels
  labels <- importance %>%
    dplyr::filter(variable == var) %>%
    select(source_transcriptome) %>% deframe %>% levels()

  # Obtain and round importance
  imp <- importance %>%
    dplyr::filter(variable == var) %>%
    select(importance) %>% deframe
  imp <- round(imp, 2)

  # Generate plot by type
  if(var_type[5] == "numeric"){

    if(var_name=='min_sample_cov'){

      var_df <- var_df %>%
        dplyr::filter(!is.na(variable),
                      !(source_transcriptome == labels[2] &
                          artifact_lab == 'Common'))%>%
        dplyr::mutate(variable = factor(variable)) %>%
        group_by(source_transcriptome,
                 structural_category, artifact_lab, variable) %>%
        summarize(category_count = n()) %>%
        group_by(source_transcriptome, structural_category, artifact_lab) %>%
```

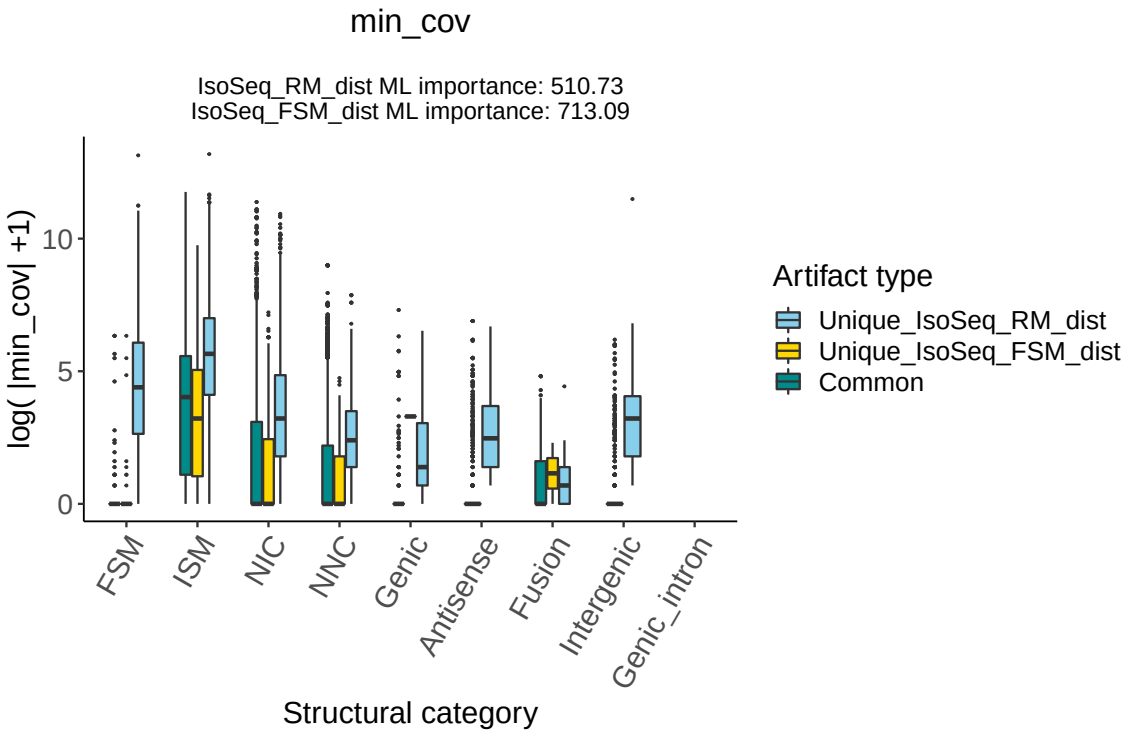
```
      mutate(percentin=category_count/sum(category_count)) %>%
      mutate(suma=cumsum(category_count)) %>%
      arrange(desc(variable))

    p<- ggplot(var_df) +
      ggtitle(paste(var, "\n"),
              subtitle = paste0("\n", labels[1], " ML importance: ", imp[1],
                                "\n", labels[2], " ML importance: ", imp[2])) +
      geom_bar(aes(y=percentin,x = artifact_lab, fill = variable ),
               stat = "identity", width = 0.8, color = "black",
               position = "stack") +
      labs(x = "Artifact type", y = "Percentage") +
      theme(axis.text.x = element_text(angle = 60, hjust = 1)) +
      geom_text(aes(x=artifact_lab,y=1.03, label=suma),size=2,
                check_overlap=TRUE)+
      scale_y_continuous(labels=scales::label_percent())+
              RColorConesa::scale_fill_conesa(paste0(var), palette = "complete",
                                      continuous = FALSE, reverse = FALSE) +
      facet_grid(~structural_category, scales = "free")


    return(p)

  }

  else{
    p <- ggplot(var_df) +
      ggtitle(paste(var),
              subtitle = paste0("\n", labels[1], " ML importance: ", imp[1],
                                "\n", labels[2], " ML importance: ", imp[2])) +
      geom_boxplot(aes(x = structural_category, y = log(abs(variable)+1),
                       fill = artifact_lab),
                   outlier.size = 0.2, width = 0.5) +
      labs(x = "Structural category", y = paste0("log( |", var_name, "| +1)")) +
      scale_fill_manual(values=filt_palette_un[3:5],name="Artifact type")+
      theme(axis.text.x = element_text(angle = 60, hjust = 1))


    return(p)

  }

} else if(var_type[5] == "integer"){

  # Specific plot for exon-related columns (integer variables divided into intervals)
  var_fct <- var_df %>%
    dplyr::filter(artifact_type == "Unique") %>%
    dplyr::mutate(variable = cut(variable, breaks = c(0, 1, 3, 5, 10, max(.$variable)),
                                 labels = c("1", "2-3", "4-5", "6-10", ">10")))

  p <- ggplot(var_fct) +
    ggtitle(paste(var, "\n"),
            subtitle = paste0("\n", labels[1], " ML importance: ", imp[1],
```

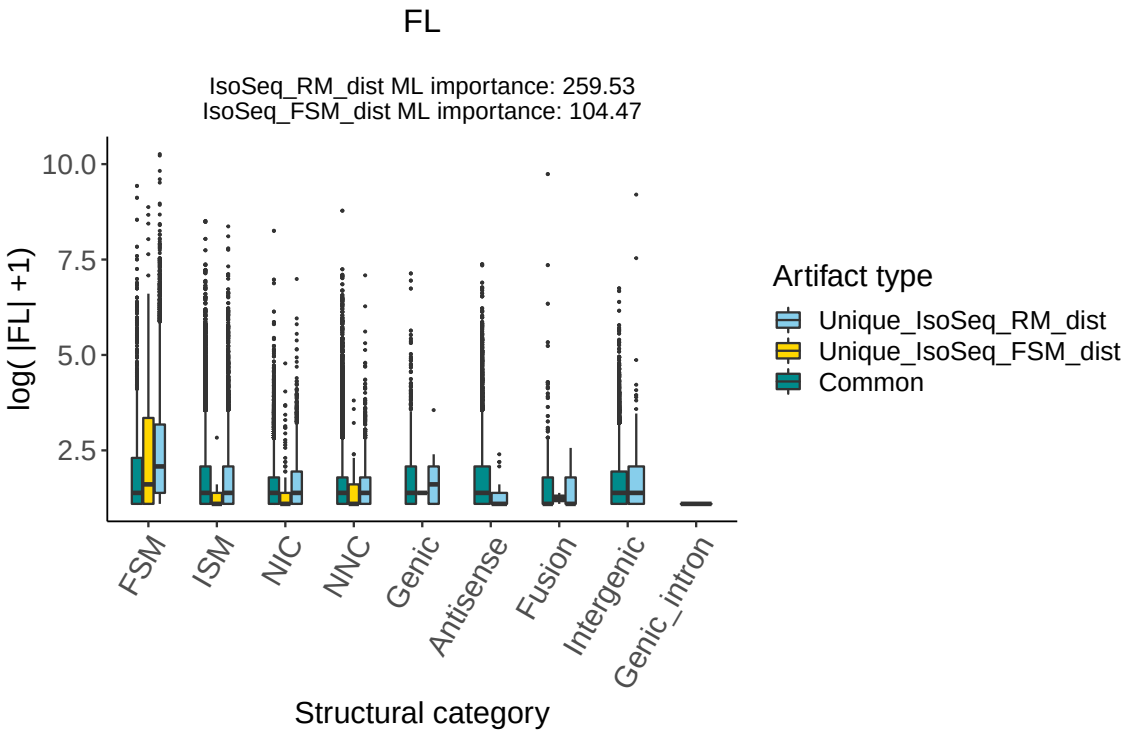
```
                                "\n", labels[2], " ML importance: ", imp[2], "\n")) +
      geom_bar(aes(x = artifact_lab, fill = variable), stat = "count",
               width = 0.8, color = "black", position = "dodge") +
      labs(x = "Artifact type \n(common artifacts not displayed)",
           y = "Transcript no.") +
      theme(axis.text.x = element_text(angle = 60, hjust = 1)) +
      scale_fill_manual(values=filt_palette_un[3:5],name=paste0(var))+
      facet_grid(~structural_category, scales = "free")

    return(p)

  } else{
    var_df <- var_df %>%
      dplyr::filter(!is.na(variable),
                    !(source_transcriptome == labels[2] &
                        artifact_lab == 'Common'))%>%
      dplyr::mutate(variable = factor(variable)) %>%
      group_by(source_transcriptome,
               structural_category, artifact_lab, variable) %>%
      summarize(category_count = n()) %>%
      group_by(source_transcriptome, structural_category, artifact_lab) %>%
      mutate(percentin=category_count/sum(category_count)) %>%
      mutate(suma=cumsum(category_count)) %>%
      arrange(desc(variable))

    p<- ggplot(var_df) +
      ggtitle(paste(var, "\n"),
              subtitle = paste0("\n", labels[1], " ML importance: ", imp[1],
                                "\n", labels[2], " ML importance: ", imp[2])) +
      geom_bar(aes(y=percentin,x = artifact_lab, fill = variable ),
               stat = "identity", width = 0.8, color = "black",
               position = "stack") +
      labs(x = "Artifact type", y = "Percentage") +
      theme(axis.text.x = element_text(angle = 60, hjust = 1)) +
      geom_text(aes(x=artifact_lab,y=1.03, label=suma),size=2,
                check_overlap=TRUE)+
      scale_y_continuous(labels=scales::label_percent())+
      scale_fill_manual(values=filt_palette_un[3:5],name=paste0(var))+
      facet_grid(~structural_category, scales = "free")


    return(p)

  }
}
```

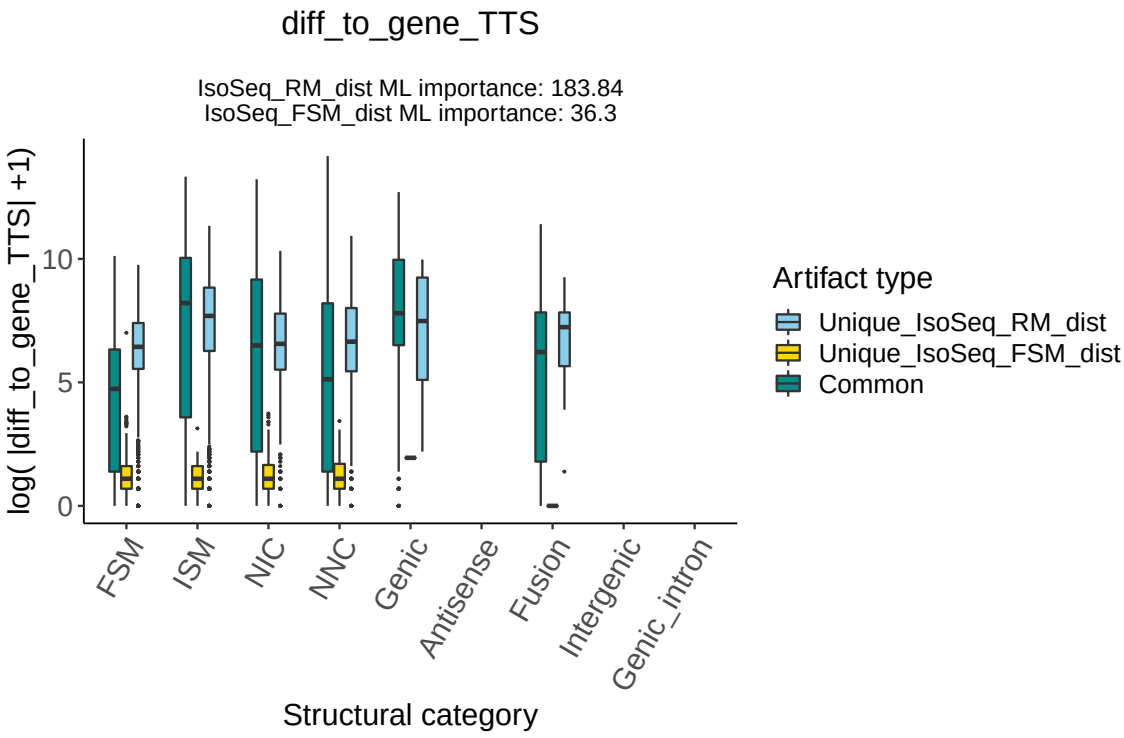
So that, the next code will generate the plots comparing the values of all the variables:

```
art_compare <- purrr::map(imp_combined$variable %>% unique,
                          ~compare_artifacts(artifacts_all, ., imp_combined))
```

As an example, it is shown some of the plots generated:


min_cov
IsoSeq_RM_dist ML importance: 510.73
IsoSeq_FSM_dist ML importance: 713.09
log(|min_cov| +1)
10
5
0
FSM
ISM
NIC
NNC
Genic
Antisense
Fusion
Intergenic
Genic_intron
Structural category
Artifact type
Unique_IsoSeq_RM_dist
Unique_IsoSeq_FSM_dist
Common


FL
IsoSeq_RM_dist ML importance: 259.53
IsoSeq_FSM_dist ML importance: 104.47
log(|FL| +1)
10.0
7.5
5.0
2.5
FSM
ISM
NIC
NNC
Genic
Antisense
Fusion
Intergenic
Genic_intron
Structural category
Artifact type
Unique_IsoSeq_RM_dist
Unique_IsoSeq_FSM_dist
Common


diff_to_gene_TTS
IsoSeq_RM_dist ML importance: 183.84
IsoSeq_FSM_dist ML importance: 36.3
log(|diff_to_gene_TTS| +1)
10
5
0
FSM
ISM
NIC
NNC
Genic
Antisense
Fusion
Intergenic
Genic_intron
Structural category
Artifact type
Unique_IsoSeq_RM_dist
Unique_IsoSeq_FSM_dist
Common

Bidimensional plots (for comparing each variable vs distance variables)

First of all, we store the names of the distance variables columns in distcol.

```
distcol<-colnames(artifacts_all)[str_detect(colnames(artifacts_all), "diff")]
```

And we also store the names of the structural categories in scat.

```
scat<- select(artifacts_all,structural_category) %>% filter(!duplicated(structural_category))
scat<-as.character(scat$structural_category)
```

We create a function to generate bidimensional plots of each numeric variable vs each distance variable (bidimensional_plots_num). The function takes the artifacts_all table, the distance columns names, the variable to compare with, one label of the filtering combination and one name of a structural category.

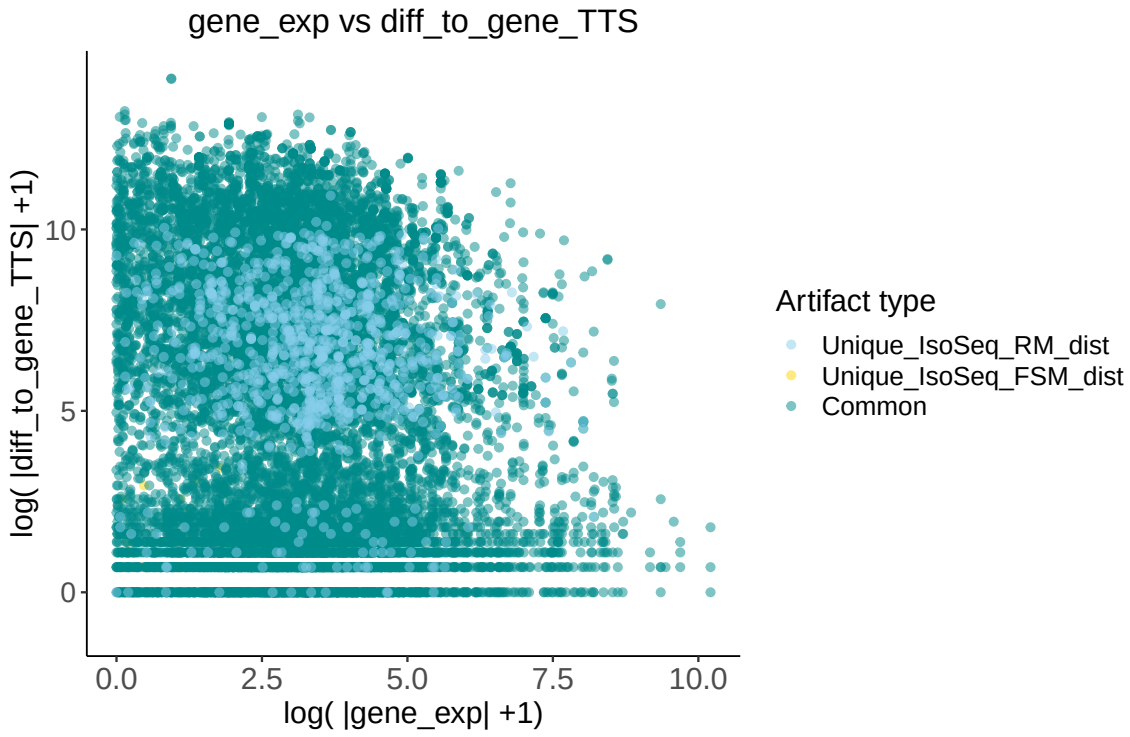
```
bidimensional_plotsnum <- function(classification,
                                   col, var,labl, sc){
  var_df <- classification %>%
    dplyr::select(structural_category,
                  source_transcriptome,
                  artifact_type, artifact_lab,
                  dplyr::all_of(col),
                  dplyr::all_of(var))

  # Rename variable column to handle during plotting
  var_df <- var_df %>% dplyr::rename(variable = var)
  # Rename distance column to handle during plotting
  var_df <- var_df %>% dplyr::rename(distance = col)

  var_df <- var_df %>%
    dplyr::filter(!(is.na(variable))& !(is.na(distance)))


  difvar<-var_df %>%
    dplyr::filter(!(source_transcriptome == labl &
                      artifact_lab == 'Common'))%>%
    dplyr::filter(structural_category==sc) %>%
    dplyr::mutate(variable = factor(variable)) %>%
    group_by(source_transcriptome,
             structural_category, artifact_lab, variable, distance) %>%
    summarize(category_count = n()) %>%
    group_by(source_transcriptome, structural_category, artifact_lab) %>%
    mutate(suma=cumsum(category_count)) %>%
    arrange(structural_category)


  difvar$variable<-as.numeric(as.character(difvar$variable))


  p<- ggplot(difvar) +
    ggtitle(paste0(var," vs ",col))+
    geom_point(aes(x = log(abs(variable)+1) , y = log(abs(distance)+1),
                   colour = factor(artifact_lab)), alpha = 0.5)+
```

```
    ylim(-1,NA)+
    labs(x=paste0("log( |",var,"| +1)"), y = paste0("log( |",col,"| +1)"))+
    scale_color_manual(values=filt_palette_un[3:5],name="Artifact type")

}
```

So that, a code like the following one will store a set of bidimensional plots for each variable desired, in this example, 'gene_exp' for each category, in this example, ISM.

```
plots_expISM<- purrr::map(distcol,
                          ~bidimensional_plotsnum(artifacts_all, .,'gene_exp', label2,'NNC'))
```

As an example, it is shown one of the plots generated:



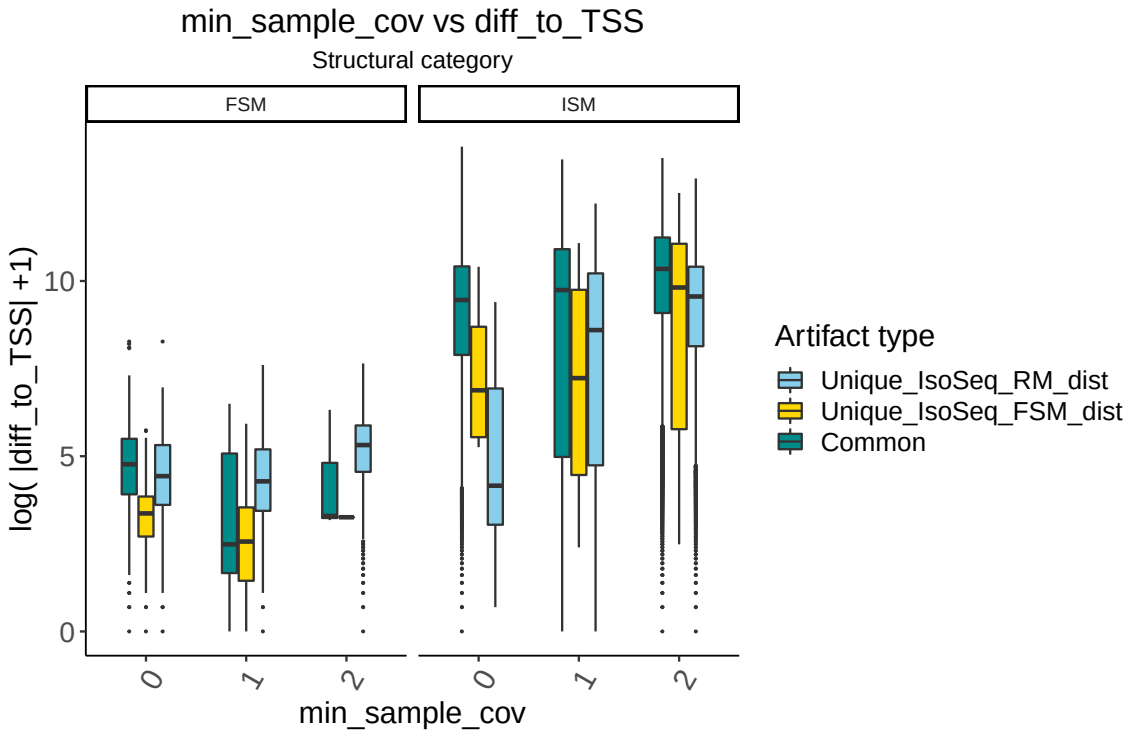
Then, we create another function to plot the rest of variables (non numeric) the same way (bidimensional_plots).

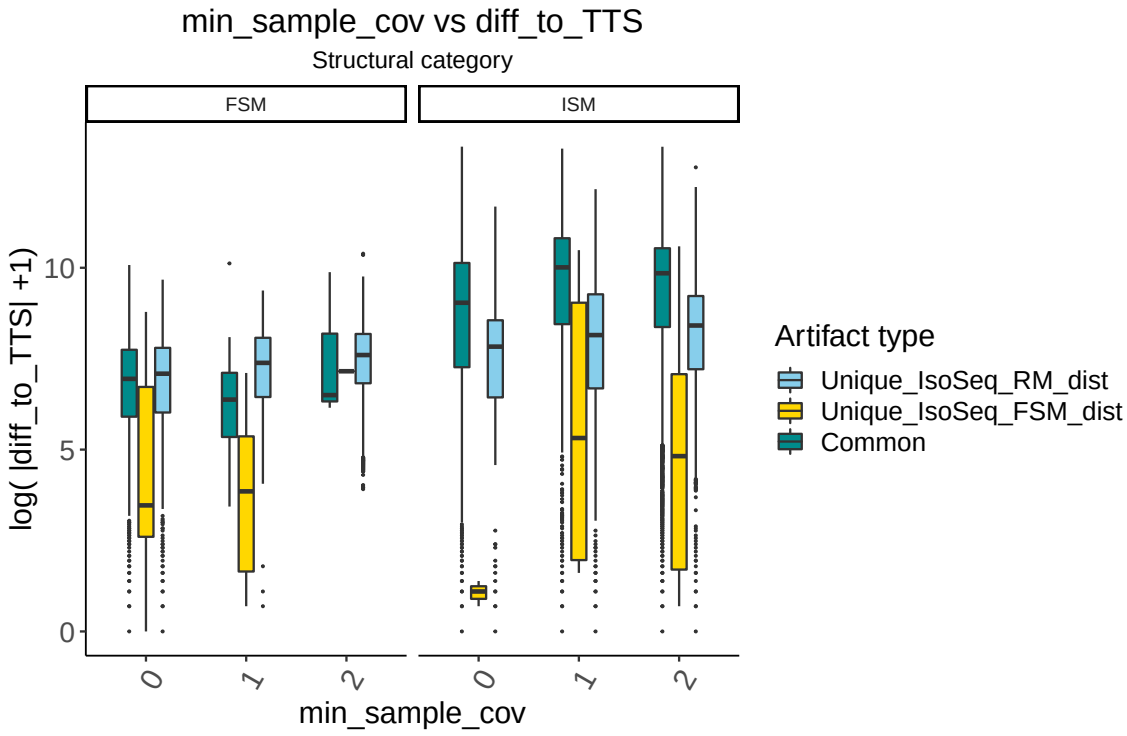
```
bidimensional_plots <- function(classification,
                                col, var,labl){
  var_df <- classification %>%
    dplyr::select(structural_category,
                  source_transcriptome,
                  artifact_type, artifact_lab,
```

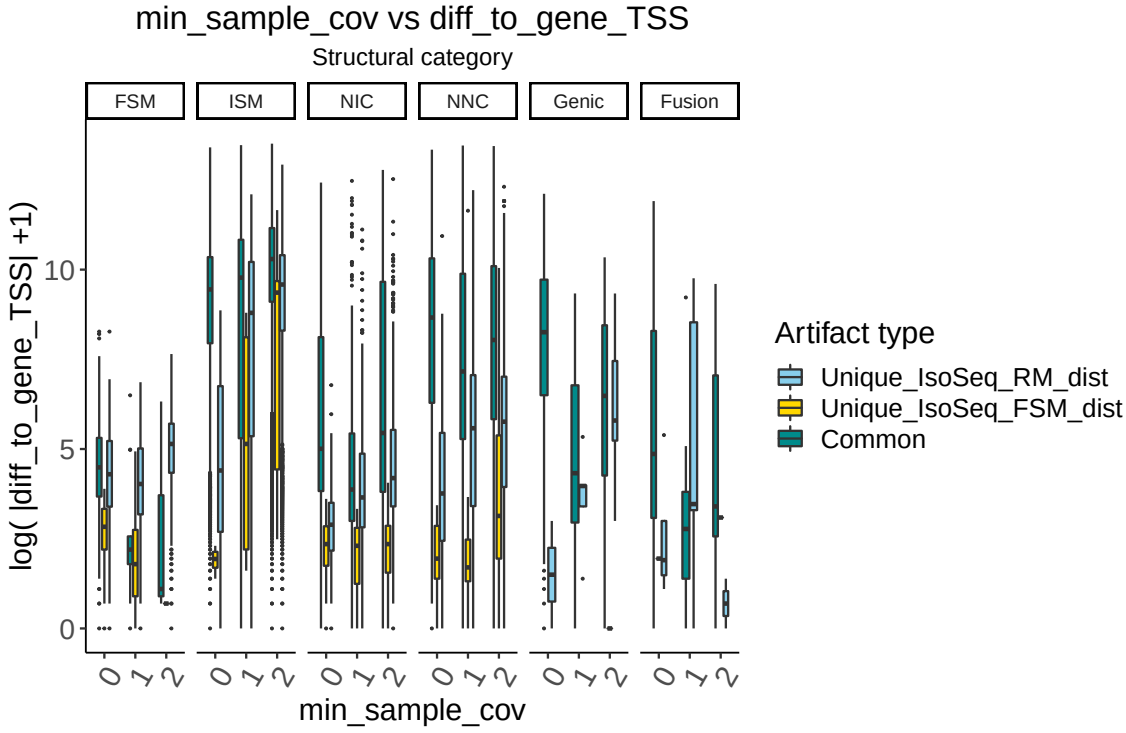
```
                  dplyr::all_of(col),
                  dplyr::all_of(var))

  # Rename variable column to handle during plotting
  var_df <- var_df %>% dplyr::rename(variable = var)
  # Rename distance column to handle during plotting
  var_df <- var_df %>% dplyr::rename(distance = col)

  var_df <- var_df %>%
    dplyr::filter(!(is.na(variable))& !(is.na(distance)))

  #plot


  diffvar<-var_df %>%
    dplyr::filter(!(source_transcriptome == labl &
                      artifact_lab == 'Common'))%>%
    dplyr::mutate(variable = factor(variable)) %>%
    group_by(source_transcriptome,
             structural_category, artifact_lab, variable, distance) %>%
    summarize(category_count = n()) %>%
    group_by(source_transcriptome, structural_category, artifact_lab) %>%
    mutate(suma=cumsum(category_count)) %>%
    arrange(structural_category)

  p<-ggplot(diffvar) +
  ggtitle(paste0(var," vs ",col))+
  geom_boxplot(aes(x = variable, y = log(abs(distance)+1),
                   fill = artifact_lab),
               outlier.size = 0.2, width = 0.5) +
    labs(subtitle= "Structural category", x=var,
         y = paste0("log( |",col,"| +1)")) +
    facet_grid(~structural_category, scales = "free")+
    scale_fill_manual(values=filt_palette_un[3:5],name="Artifact type") +
    theme(axis.text.x = element_text(angle = 60, hjust = 1))
}
```

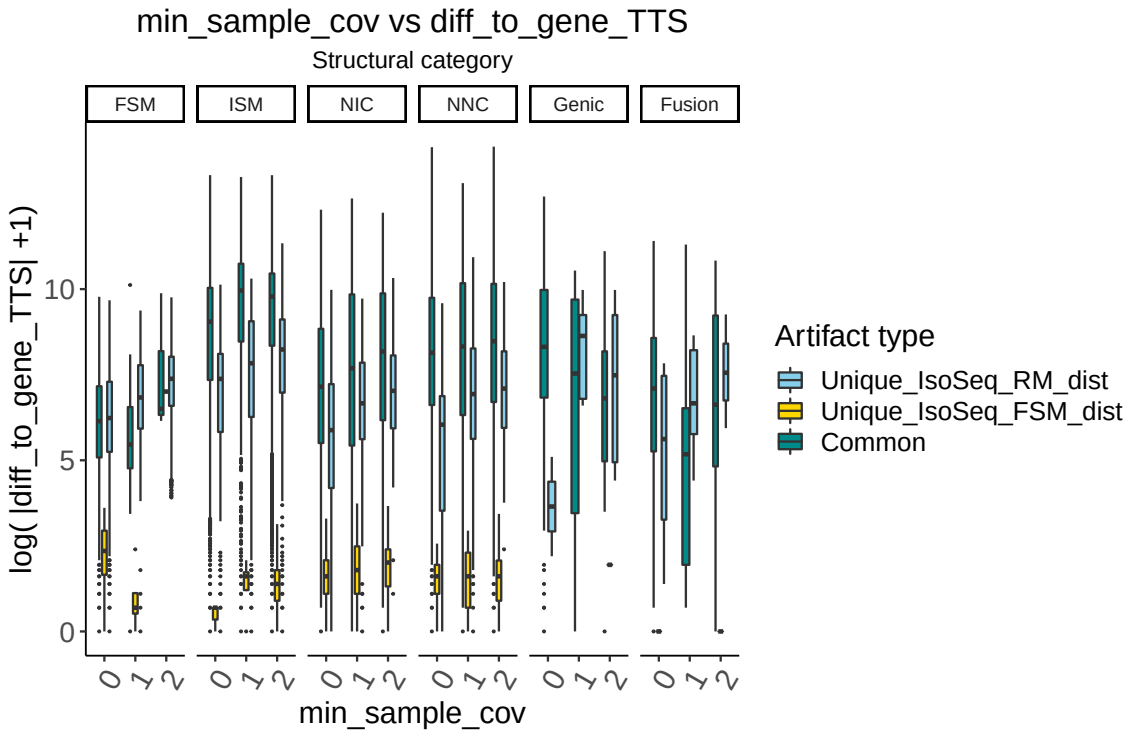
With this code we can obtain all the bidemensional plots of 'min_sample_cov' vs each distance variable:

```
plots_min<- purrr::map(distcol,
                       ~bidimensional_plots(artifacts_all, .,
                                            'min_sample_cov', label2))
```


min_sample_cov vs diff_to_TSS
Structural category
FSM
ISM
log(|diff_to_TSS| +1)
10
5
0
0
1
2
0
1
2
min_sample_cov
Artifact type
Unique_IsoSeq_RM_dist
Unique_IsoSeq_FSM_dist
Common


min_sample_cov vs diff_to_TTS
Structural category
FSM
ISM
log(|diff_to_TTS| +1)
10
5
0
0
1
2
0
1
2
min_sample_cov
Artifact type
Unique_IsoSeq_RM_dist
Unique_IsoSeq_FSM_dist
Common


min_sample_cov vs diff_to_gene_TSS
Structural category
FSM
ISM
NIC
NNC
Genic
Fusion
log(|diff_to_gene_TSS| +1)
10
5
0
0
1
2
min_sample_cov
Artifact type
Unique_IsoSeq_RM_dist
Unique_IsoSeq_FSM_dist
Common


min_sample_cov vs diff_to_gene_TTS
Structural category
FSM
ISM
NIC
NNC
Genic
Fusion
log(|diff_to_gene_TTS| +1)
10
5
0
0
1
2
min_sample_cov
Artifact type
Unique_IsoSeq_RM_dist
Unique_IsoSeq_FSM_dist
Common