Long_Demo

```
library(rnaGinesis)
library(ggplot2)
library(reshape)
library(NMF)
mu <- rnaGinesis::mu
A <- rnaGinesis::A
```

Simulate Default Setting, Four Tissues

```
# Proportions are 0.3 0.4 0.2 0.1
num.sim = 20
true_default_easier <- list()
results_default_easier <- list()
for (i in 1:num.sim)
  {
    writeLines("---------------")
    writeLines(paste("iteration",i))

    simresult <- Complete_simulation(A,
                                     mu,
                                     Samplesize  = 50,
                                     scaleFactor = rep(80, 3),
                                     d.params    = c("Tumor"   = .3,
                                                     "Stromal" = .4,
                                                     "Immune"  = .2,
                                                     "Normal"  = .1),
                                     noise_setting = 1.01,
                                     seed          = i + 1234 )
    data   <- simresult[[1]]

    true <- list()
    true$trueW <- simresult[[2]]
    true$trueH <- simresult[[3]]
    true_default_easier[[i]] <- true

    res <- NMF::nmf(data, rank = 4, seed = i+123456, nrun = 1, .options = "p4")
    results_default_easier[[i]] <- res
}
```

Simulate High Rearrange - more differences in the tissues

```
# Now try high rearrange
num.sim = 20
true_high_rearrange_easier <- list()
```

```
results_high_rearrange_easier <- list()
for (i in 1:num.sim)
  {
    writeLines("---------------")
    writeLines(paste("iteration",i))

    simresult <- Complete_simulation(A,
                                     mu,
                                     Samplesize  = 50, #100,
                                    scaleFactor = rep(4000, 3),
                                    d.params    = c("Tumor"   = .3,
                                                    "Stromal" = .4,
                                                    "Immune"  = .2,
                                                    "Normal"  = .1),
                                    noise_setting = 1.01,
                                    seed          = i + 1234 )
    data   <- simresult[[1]]

    true <- list()
    true$trueW <- simresult[[2]]
    true$trueH <- simresult[[3]]
    true_high_rearrange_easier[[i]] <- true

    res <- NMF::nmf(data, rank = 4, seed = i+123456, nrun = 1, .options = "p4")
    results_high_rearrange_easier[[i]] <- res
}
```

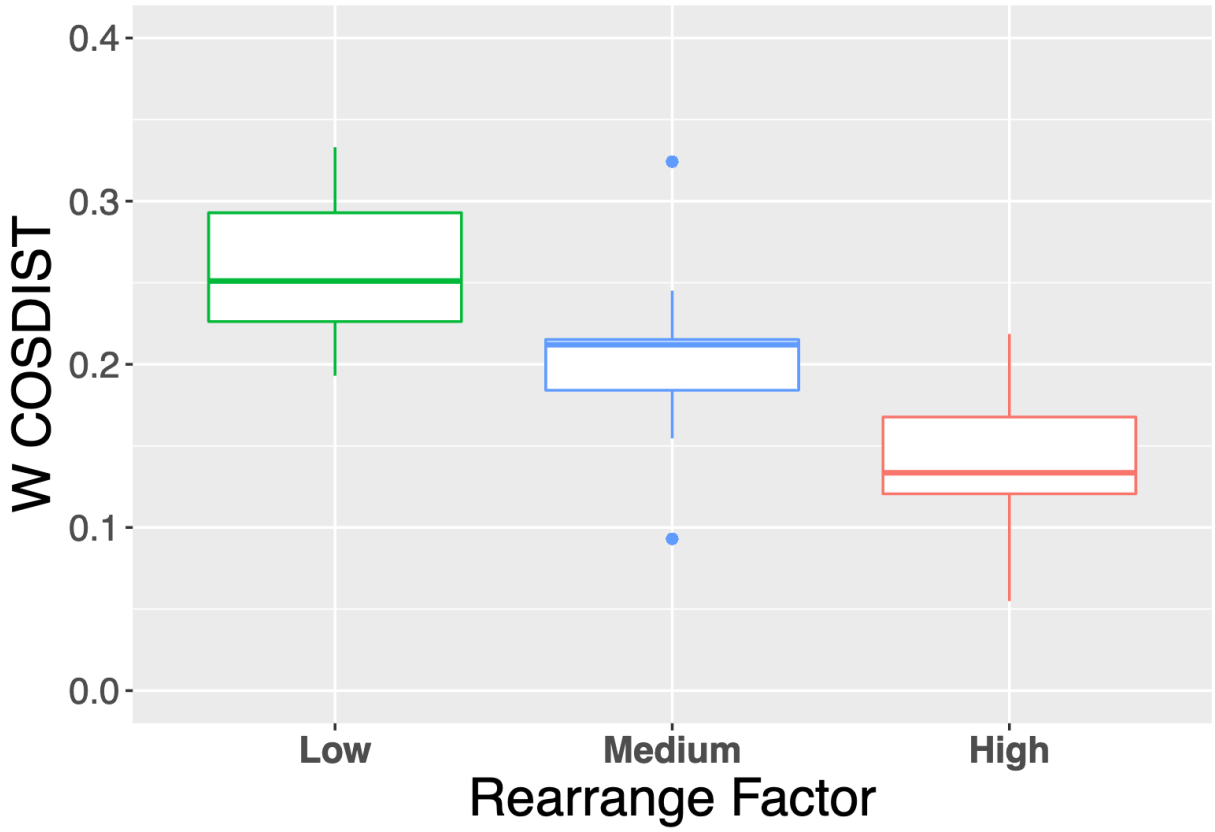
Simulate Low Rearrange - tissues are much more similar

```
# Now try low rearrange
num.sim = 20
true_low_rearrange_easier <- list()
results_low_rearrange_easier <- list()
for (i in 1:num.sim)
  {
    writeLines("---------------")
    writeLines(paste("iteration",i))

    simresult <- Complete_simulation(A,
                                     mu,
                                     Samplesize  = 50,
                                    scaleFactor = rep(2, 3),
                                    d.params    = c("Tumor"   = .3,
                                                    "Stromal" = .4,
                                                    "Immune"  = .2,
                                                    "Normal"  = .1),
                                    noise_setting = 1.01,
                                    seed          = i + 1234 )
    data   <- simresult[[1]]

    true <- list()
    true$trueW <- simresult[[2]]
```

```
    true$trueH <- simresult[[3]]
    true_low_rearrange_easier[[i]] <- true

    res <- NMF::nmf(data, rank = 4, seed = i+123456, nrun = 1, .options = "p4")
    results_low_rearrange_easier[[i]] <- res
}
```

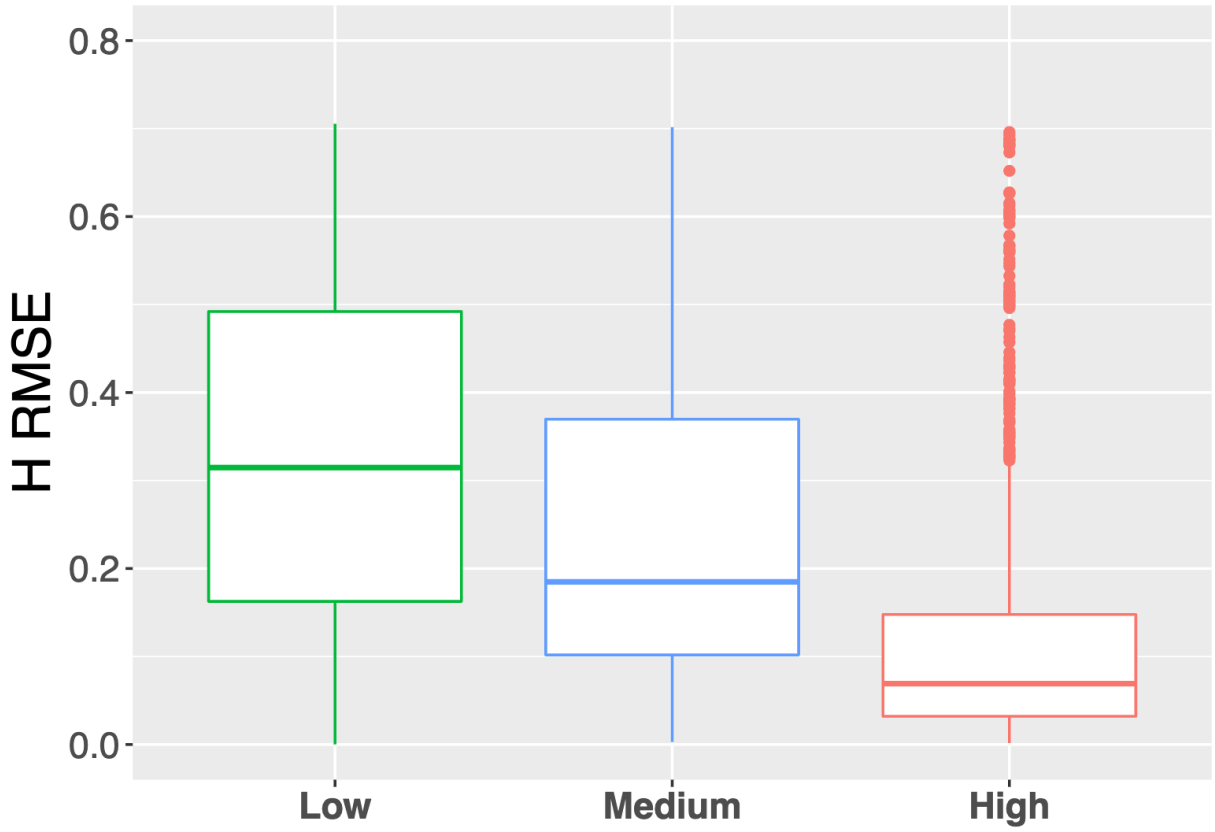
Evaluate the Errors relative to known truth for default setting

```
#  re-arrange relative to the true experiment, and see which one gives a better rearrange

# fill in the h and cosdist
W.COSDIST <- vector() #matrix(ncol = 9)
H.RMSE <- vector() #matrix(ncol = 9)

for (i in 1:20) {

  # get true
  true <- true_default_easier[[i]]
  # make sum to 1
  true$trueH <- apply(true$trueH, 2, FUN = sum_to_1)
  # make into n
  true$trueW <- lapply(seq_len(ncol(true$trueH)), function(X) true$trueW)

  exp <- list()
  exp$resultW <- results_default_easier[[i]]@fit@W
  exp$resultH <- results_default_easier[[i]]@fit@H
  exp$resultW <- lapply(seq_len(ncol(exp$resultH)), function(X) exp$resultW)
  # make sum to 1
  exp$resultH <- apply(exp$resultH, 2, FUN = sum_to_1)
  # re-arrange the results
  exp <-rearrange(exp, true)
  # get the rmse
  rmse <- evalH.RMSE(true$trueH,exp$resultH)
  H.RMSE <- cbind(H.RMSE,rmse)
  # get the cosine distance
  cosdist <- evalW.COSDIST(true$trueW,exp$resultW)
  W.COSDIST <- cbind(W.COSDIST, cosdist)
}

df_default_easier <- reshape2::melt(W.COSDIST)
df_default_easier <- df_default_easier[,-1]
names(df_default_easier) <- c("method", "W_COSDIST")

df_default_easier_h <- reshape2::melt(H.RMSE)
df_default_easier_h <- df_default_easier_h[,-1]
names(df_default_easier_h) <- c("method", "H_RMSE")
```



Plot RMSE for proportions of tissues in each sample

```
boxplot_default_easier_h <- df_default_easier_h
boxplot_default_easier_h[,"method"] <- "Medium"
boxplot_high_rearrange_easier_h <- df_high_rearrange_easier_h
boxplot_high_rearrange_easier_h[,"method"] <- "High"
boxplot_low_rearrange_easier_h <- df_low_rearrange_easier_h
boxplot_low_rearrange_easier_h[,"method"] <- "Low"
boxplot_rmse <- rbind(boxplot_default_easier_h, boxplot_high_rearrange_easier_h, boxplot_low_rearrange_
level_order <- factor(boxplot_rmse$method, level = c("Low", "Medium", "High"))
p4 <-ggplot2::ggplot(boxplot_rmse, ggplot2::aes(x=level_order, y=H_RMSE, color=method)) +
  ggplot2::geom_boxplot() + ylim(0,0.8) +
  ggplot2::theme(
  plot.title = element_blank(),
  axis.title.x = element_blank(),
  axis.title.y = ggplot2::element_text(size = 20), legend.position = "none", axis.text.x = ggplot2::ele
print(p4)
```



Simulate Medium Amount of Noise

```
num.sim = 20
true_med_noise_easier <- list()
results_med_noise_easier <- list()
for (i in 1:num.sim)
  {
    writeLines("---------------")
    writeLines(paste("iteration",i))

    simresult <- Complete_simulation(A,
                                     mu,
                                     Samplesize  = 50, #100,
                                    scaleFactor = rep(80, 3),
                                    d.params    = c("Tumor"   = .3,
                                                    "Stromal" = .4,
                                                    "Immune"  = .2,
                                                    "Normal"  = .1),
                                    noise_setting = 5,
                                    seed          = i + 1234 )
    data   <- simresult[[1]]

    true <- list()
    true$trueW <- simresult[[2]]
    true$trueH <- simresult[[3]]
```

```
    true_med_noise_easier[[i]] <- true

    res <- NMF::nmf(data, rank = 4, seed = i+123456, nrun = 1, .options = "p4")
    results_med_noise_easier[[i]] <- res
}
```

Simulate High amount of noise

```
num.sim = 20
true_high_noise_easier <- list()
results_high_noise_easier <- list()
for (i in 1:num.sim)
  {
    writeLines("---------------")
    writeLines(paste("iteration",i))

    simresult <- Complete_simulation(A,
                                     mu,
                                     Samplesize  = 50, #100,
                                    scaleFactor = rep(80, 3),
                                    d.params    = c("Tumor"   = .3,
                                                    "Stromal" = .4,
                                                    "Immune"  = .2,
                                                    "Normal"  = .1),
                                    noise_setting = 10,
                                    seed          = i + 1234 )
    data   <- simresult[[1]]

    true <- list()
    true$trueW <- simresult[[2]]
    true$trueH <- simresult[[3]]
    true_high_noise_easier[[i]] <- true

    res <- NMF::nmf(data, rank = 4, seed = i+123456, nrun = 1, .options = "p4")
    results_high_noise_easier[[i]] <- res
}
```

Evaluate the Errors relative to known truth for high noise

```
#  re-arrange relative to the true experiment, and see which one gives a better rearrange

# fill in the h and cosdist
W.COSDIST <- vector() #matrix(ncol = 9)
H.RMSE <- vector() #matrix(ncol = 9)

for (i in 1:20) {

  # get true
  true <- true_high_noise_easier[[i]]
  # make sum to 1
  true$trueH <- apply(true$trueH, 2, FUN = sum_to_1)
```

```
  # make into n
  true$trueW <- lapply(seq_len(ncol(true$trueH)), function(X) true$trueW)


  exp <- list()
  exp$resultW <- results_high_noise_easier[[i]]@fit@W
  exp$resultH <- results_high_noise_easier[[i]]@fit@H
  exp$resultW <- lapply(seq_len(ncol(exp$resultH)), function(X) exp$resultW)
  # make sum to 1
  exp$resultH <- apply(exp$resultH, 2, FUN = sum_to_1)
  # re-arrange the results
  exp <-rearrange(exp, true)
  # get the rmse
  rmse <- evalH.RMSE(true$trueH,exp$resultH)
  H.RMSE <- cbind(H.RMSE,rmse)
  # get the cosine distance
  cosdist <- evalW.COSDIST(true$trueW,exp$resultW)
  W.COSDIST <- cbind(W.COSDIST, cosdist)
}

df_high_noise_easier <- reshape2::melt(W.COSDIST)
df_high_noise_easier <- df_high_noise_easier[,-1]
names(df_high_noise_easier) <- c("method", "W_COSDIST")

df_high_noise_easier_h <- reshape2::melt(H.RMSE)
df_high_noise_easier_h <- df_high_noise_easier_h[,-1]
names(df_high_noise_easier_h) <- c("method", "H_RMSE")
```

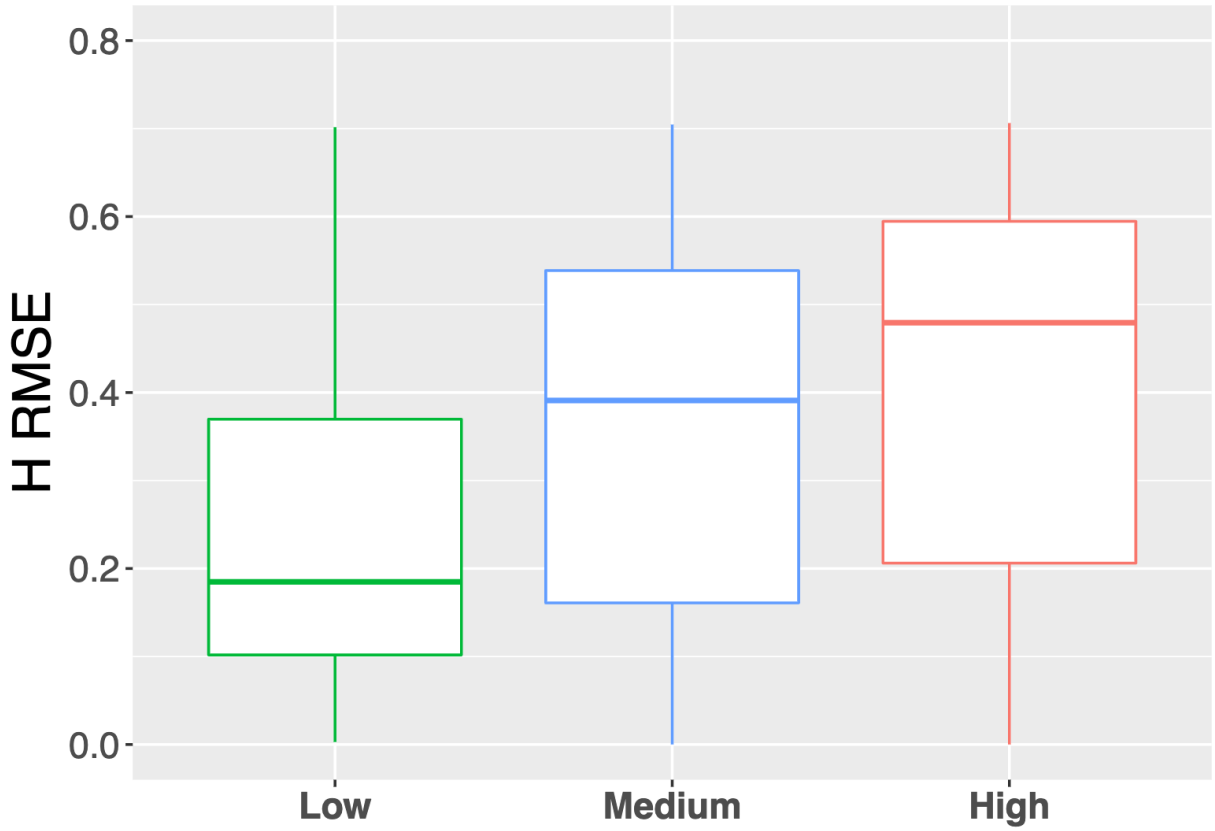
Evaluate the Errors relative to known truth for medium noise

```
#  re-arrange relative to the true experiment, and see which one gives a better rearrange

# fill in the h and cosdist
W.COSDIST <- vector() #matrix(ncol = 9)
H.RMSE <- vector() #matrix(ncol = 9)

for (i in 1:20) {

  # get true
  true <- true_med_noise_easier[[i]]
  # make sum to 1
  true$trueH <- apply(true$trueH, 2, FUN = sum_to_1)
  # make into n
  true$trueW <- lapply(seq_len(ncol(true$trueH)), function(X) true$trueW)


  exp <- list()
  exp$resultW <- results_med_noise_easier[[i]]@fit@W
  exp$resultH <- results_med_noise_easier[[i]]@fit@H
  exp$resultW <- lapply(seq_len(ncol(exp$resultH)), function(X) exp$resultW)
  # make sum to 1
  exp$resultH <- apply(exp$resultH, 2, FUN = sum_to_1)
```

```
  # re-arrange the results
  exp <-rearrange(exp, true)
  # get the rmse
  rmse <- evalH.RMSE(true$trueH,exp$resultH)
  H.RMSE <- cbind(H.RMSE,rmse)
  # get the cosine distance
  cosdist <- evalW.COSDIST(true$trueW,exp$resultW)
  W.COSDIST <- cbind(W.COSDIST, cosdist)
}

df_med_noise_easier <- reshape2::melt(W.COSDIST)
df_med_noise_easier <- df_med_noise_easier[,-1]
names(df_med_noise_easier) <- c("method", "W_COSDIST")

df_med_noise_easier_h <- reshape2::melt(H.RMSE)
df_med_noise_easier_h <- df_med_noise_easier_h[,-1]
names(df_med_noise_easier_h) <- c("method", "H_RMSE")
```

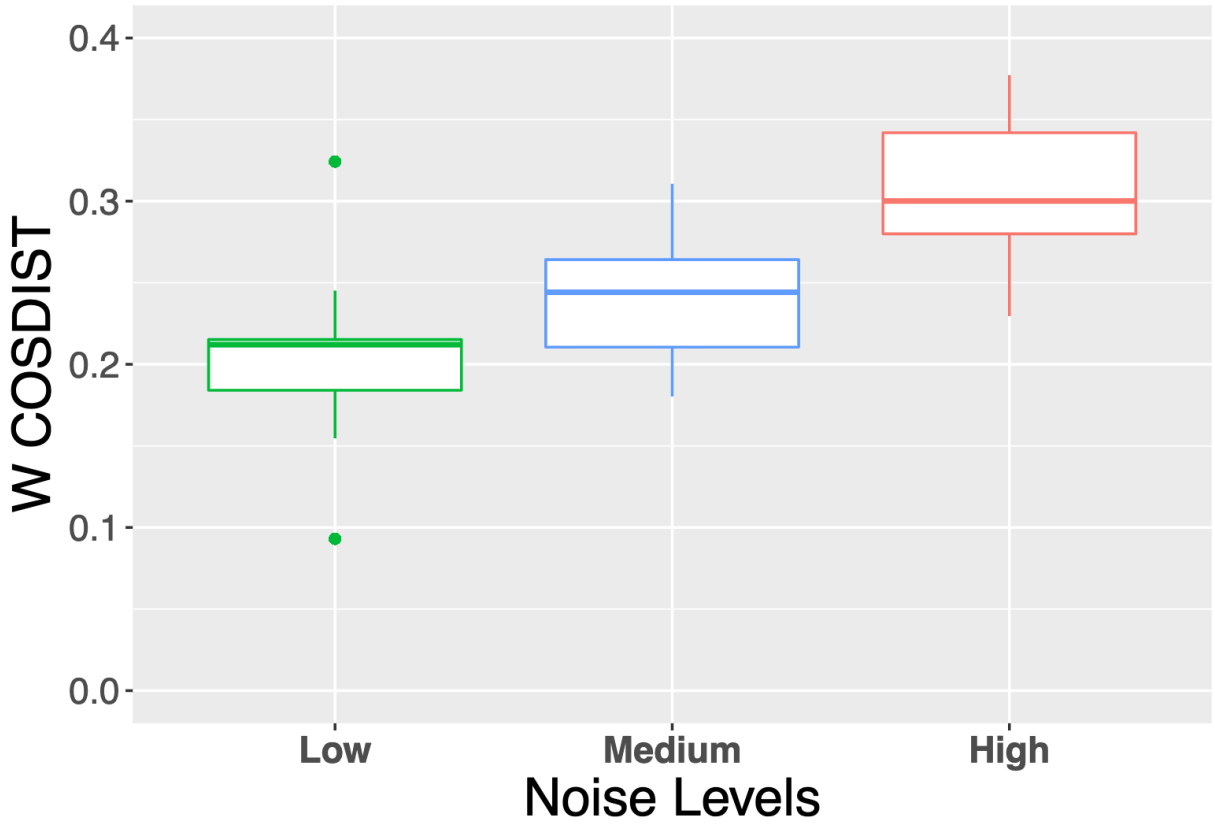
Plot Cosine Distances for gene expression of four tissues

```
boxplot_default_easier_h <- df_default_easier_h
boxplot_default_easier_h[,"method"] <- "Low"
boxplot_high_noise_easier_h <- df_high_noise_easier_h
boxplot_high_noise_easier_h[,"method"] <- "High"
boxplot_med_noise_easier_h <- df_med_noise_easier_h
boxplot_med_noise_easier_h[,"method"] <- "Medium"
boxplot_rmse <- rbind(boxplot_default_easier_h, boxplot_high_noise_easier_h, boxplot_med_noise_easier_h
level_order <- factor(boxplot_rmse$method, level = c("Low", "Medium", "High"))
p4 <-ggplot2::ggplot(boxplot_rmse, ggplot2::aes(x=level_order, y=H_RMSE, color=method)) +
  ggplot2::geom_boxplot() + ylim(0,0.8) +
  ggplot2::theme(
  plot.title = element_blank(),
  axis.title.x = element_blank(),
  axis.title.y = ggplot2::element_text(size = 20), legend.position = "none", axis.text.x = ggplot2::ele
print(p4)
```



Plot RMSE for proportions of tissues in each sample

```
boxplot_default_easier <- df_default_easier
boxplot_default_easier[,"method"] <- "Low"
boxplot_high_noise_easier <- df_high_noise_easier
boxplot_high_noise_easier[,"method"] <- "High"
boxplot_med_noise_easier <- df_med_noise_easier
boxplot_med_noise_easier[,"method"] <- "Medium"
boxplot_rmse <- rbind(boxplot_default_easier, boxplot_high_noise_easier, boxplot_med_noise_easier)
level_order <- factor(boxplot_rmse$method, level = c("Low", "Medium", "High"))
p4 <-ggplot2::ggplot(boxplot_rmse, ggplot2::aes(x=level_order, y=W_COSDIST, color=method)) +
  ggplot2::geom_boxplot()  + ylim(0,0.4) +
ggplot2::theme(
  plot.title = element_blank(),
  axis.title.x = ggplot2::element_text(size = 20),
  axis.title.y = ggplot2::element_text(size = 20), legend.position = "none", axis.text.x = ggplot2::ele
print(p4)
```


0.4
0.3
0.2
0.1
0.0
W COSDIST
Low
Medium
High
Noise Levels