Final Assignment MDGE 610

Dorian Rojas Villalta

Written report.

Dorian Rojas Villalta; Ph.D. Student.

This document responds to assignment theoretical questions and empirical investigation analysis (including executable code). The Github repository containing this pdf-rendered document, a summary `README.md` file, and change tracking information can be found at the following link:

> MDGE610_final_project.

First Section. Conceptual understanding

The EM algorithm.

What is the EM algorithm?

The expectation maximization (EM) algorithm is a general approach for the iteration of computational calculations of maximum likelihood estimates (MLEs) as a function of the observed data (called 'incomplete data'). This allows for the optimization of the log-likelihood objective function on incomplete data, where the calculation of MLEs is complex due to missing variables or unbalanced weights. For instance, linear regression is a statistical approach to understand the linear association of variables into an specific outcome. The effect of each variable is represented by a coefficient that decreases the sum of squares residuals (discrepancy between observed data and model prediction; SSRs). These measures are commonly optimized by least-squares estimations (LSEs). Under EM algorithms, the influence of missing data can be expected based on the observed data (expectation; E-step) and LSEs can be performed using the predicted complete data (maximization, M-step).

What objective function does it maximize?

The expectation maximization algorithm aims to maximize the log-likelihood function; the statistical calculations underlying the maximum likelihood estimation. This is a measure of how the statistical model explains the observed data ($L(\theta) = P(Data|\theta)$). In simpler words, it is the probability of seeing the observed data given the model. This function is commonly used in frequentist statistics for reducing errors such as the least-squares estimation from linear regression modelling.

What are the E-step and M-Step, conceptually?

The expectation step or E-step corresponds to the algorithm section that performs an estimation of sufficient statistics based on the observed data for generating complete data. Sufficient statistics is defined as all the information that can be obtained from a dataset; hence, the E-step aims to predict all the possible information from the observed data to fill missing values on the complete data. Furthermore, the maximization step or M-step conducts a recalculation of the maximum likelihood objective function (log-likelihood) using the predicted complete data instead of the observed data. This way, it allows for maximizing (optimizing) the maximum likelihood estimate with the completed data ignoring the constrains of the missing values.

What constitutes the missing data in kallisto's application of EM?

The missing data of the kallisto software is the equivalent classes of the transcript. An equivalent class is a multi-set of transcripts associated with a read, indicating which transcript the read could have originated from. This provides the identity of the read and their overlap with the reference (E-step), providing count numbers for abundance estimation (M-step).

kallisto's Objective Function

What is the mathematical form of the objective function?

The mathematical function that represents the likelihood function for the abundance quantification of the RNA sequencing data is:

$$L(\alpha) \propto \prod_{f \in F} \sum_{t \in T} y_{f,t} \frac{\alpha_t}{l_t} = \prod_{e \in E} \left(\sum_{t \in e} \frac{\alpha_t}{l_t} \right)^{c_e}$$

Figure 1: likelihood function for RNA-seq (Bray et al. 2016)

Here, F and T represent the total set of fragments (f) and transcripts (t), respectively. l_t is the effective length of transcript t (length with pseudoalignments) and $y_{f,t}$ the compatibility matrix (where 1 is compatibility between f and t and 0 equals no compatibility; meaning chance of pseudoalignment). Finally, α_t is the probability of selecting fragments from transcripts. Overall, this formula indicates the transcript abundance as a function of the product over equivalence classes, with c_e being the number of counts observed for equivalence class e.

How does kallisto's EM implementation maximize it?

The expectation maximization algorithm implemented by kallisto aims to predict the equivalence classes (the transcript from which the read might come from) by maximizing the probability of selecting a fragment from a specific transcript α_t. This means that it optimizes the chances of obtaining a read to be associated to an specific transcript, which would serve as the needed information (counts) for estimation of transcript abundance.

Morever, another advantage of kallisto is that the EM-optimized likelihood function is defined in term of equivalent classes. This allows for a significant reduction in the dataset that needs to be analyze, resulting in a faster sotfware in comparison to other alignment-based tools.

What convergence criteria does kallisto use to decide when to stop iterating? Where are these specified in the code?

The convergence criteria for stopping the iterations is determined by the total expected value of fragment belonging to an specific transcript ($\alpha_t N$; probability times the total number of transcripts). Authors defined the end of the iterations as for when every transcript t, the total expected value is above 0.01 transcript per million (TPM) and the variance between iterations is less than 1%. In simpler words, the EM algorithm will stop when all transcripts have a minimum 0.01 TPM abundance and maximum variance is less than 1% between iterations.

This specification is coded in the source EM algorithm file of the kallisto software, found at the directory `/kallisto/src/EMAlgorithm.h`. The convergence criteria are declared in the lines 100-104:

```
double denom;
const double alpha_limit = 1e-7;
const double alpha_change_limit = 1e-2;
const double alpha_change = 1e-2;
bool finalRound = false;
```

Here, the main convergence criteria are defined using C++ coding language. The overall syntax is `[attributes] <data type> <variable name> = <value>`. Hence, `double` means a double variable (with decimals), `bool` a boolean (`true` or `false`), and `const` is an attribute of constant (not intended to be modified by the function). The variable `alpha_limit` is the overall minimum abundance, `alpha_change_limit` the minimum abundance threshold 0.01 TPM during maximization ($\alpha_t N > 0.01$), and `alpha_change` is the maximum 1% variance between iterations. `denom` is an empty variable for reads counts normalization.

Moreover, the EM algorithm end is declared on `finalRound` which starts as `false` until convergence is tested [lines 173-209, commented code removed]:

```
bool stopEM = false; //!finalRound && (i >= min_rounds); // false initially
      //double maxChange = 0.0;
      int chcount = 0;
      for (int ec = 0; ec < num_trans_; ec++) {
        if (next_alpha[ec] > alpha_change_limit && (std::fabs(next_alpha[ec]
         ↪  - alpha_[ec]) / next_alpha[ec]) > alpha_change) {
          chcount++;
        }

        // reassign alpha_ to next_alpha
        alpha_[ec] = next_alpha[ec];

        // clear all next_alpha values 0 for next iteration
        next_alpha[ec] = 0.0;
      }

      //std::cout << chcount << std::endl;
      if (chcount == 0 && i > min_rounds) {

        stopEM=true;
      }

      if (finalRound) {
        break;
      }
```

This defines the boolean `stopEM` to `false` prior assessing for the convergence and declares integer `chcount` that counts for transcripts over both minimum abundance and iteration variation thresholds, which would deem no convergence. Here, for all equivalence classes (`ec`), the `next_alpha[ec]` (abundance of current iteration; $\alpha_t N$) are tested for minimum abundance threshold of 0.01 TPM and a, abundance variation to previous iteration (`alpha_[ec]`) exceeding 1% (`alpha_change`). If this criteria is met, the `chcount` increases in one unit, else the value remains 0 (indicating convergence). Also, previous iteration (`alpha_[ec]`) stores the values of current iteration before zeroing it for the next iteration (`next_alpha[ec] = 0.0`). Later, convergence and minimum number of iteration (50) are checked (`chcount == 0 && i > min_rounds`). When true, the `stopEM` variable is set to `true`, indicating that this corresponds to the final iteration.

This structure checks for whether the current iteration is the final (`if (finalRound)`), leading to a stopping EM algorithm. The boolean `finalRound` is declared a few lines after the convergence assessment [lines 211-221]:

```
    //std::cout << maxChange << std::endl;
    if (stopEM) {
      finalRound = true;
      alpha_before_zeroes_.resize( alpha_.size() );
      for (int ec = 0; ec < num_trans_; ec++) {
        alpha_before_zeroes_[ec] = alpha_[ec];
        if (alpha_[ec] < alpha_limit/10.0) {
          alpha_[ec] = 0.0;
        }
      }
    }
```

This loop of the EM function defines that when the boolean `stopEM` becomes true, meaning that convergence has been met, the algorithm will stop and estimated abundances will be filtered. Within this control structure, `finalRound` is declared `true`, which will break the EM algorithm in the convergence assessment. Current iteration values are stored on a different variable (`alpha_before_zeroes_`), probably as a backup. Then, each equivalence class values (`alpha_[ec]`) are zero-ed if their value is lower than `alpha_limit/10.0`. Overall, these commands define the end of the algorithm and filter low abundance equivalence classes.

How might these criteria be justified?

Since the criteria is based on a minimum abundance threshold for each equivalence class, it might indicate this is a value considered expected in RNA-seq quantification analysis. In other words, it is expected to find minimum abundance of 0.01 TPM to be reliable in alignment-based quantification and lower abundances could be associated to alignment mismatches. Moreover, the other convergence criteria is the maximum variation threshold of 1% between iterations. This is based on the EM algorithm premise where standard deviation of the likelihood objective function is considered to be constant when likelihood maximum estimates are reached.

Regardless of the fundamental basis for criteria selection, authors do not declare the logic behind defining the specific values 0.01 TPM and 1% variance. In order to assess the rationale, this assignment aims to understand the effect of convergence criteria values into relative differences between simulated data and their ground truth (see Second Section. Empirical investigation).

Local vs. Global Maxima

Examine how kallisto initializes its abundance estimates

The equivalence classes abundance estimation initializes from the pseudoalignment performed by the kallisto software. This approach identifies the transcript from which the read might have been originated without alignment information. For this, a hashing of k-mers (nucleotide sequences of k length) for both reference (transcriptome de Brujin graph) and input reads is conducted. Hashing refers to transforming the proper sequences into numerical values. These values are then used to find the best pairwise match between the reads and reference, which results in reads assigned to an specific path of the reference de Brujin graph. The matched vertices are different read k-mers defined as k-compatibility classes. It is considered that the k-compatibility class of an error-free read coincide with its equivalence class. The equivalence classes are commonly estimated from alignment-based quantification approaches, which are time-consuming and computationally-demanding. However, by conducting a pseudoalignment that estimates an equivalent k-compatibility classes, these can serve as the initial abundance estimate for the EM algorithm.

Do you think initialization matter for this problem? Why or why not?

Yes, the EM algorithm aim is predict missing data (equivalence classes) from the observed data (k-compatibility classes) in order to maximize the maximum likelihood estimates (transcript abundance). Without this initialization process, the algorithm could not be implemented. The novelty of kallisto relies on the implementation of pseudoalignment, faster and computationally-lighter than alignment, with the EM algorithm, accurate objective function optimization, to predict missing information require to obtain reliable transcripts abundance estimates.

Second Section. Empirical investigation

Testing Convergence Criteria

As previously mentioned, this assignment aims to understand the effect of convergence criteria into the relative difference between simulated data and its ground truth. For this, the convergence criteria of minimum abundance threshold (`alpha_change_limit`) and maximum variance between iterations (`alpha_change`) were modified in the `EMAlgorithm.h` file.

The altered software ran on MacBook Air with Apple M4 10 cores and 16GB of RAM local system. The rebuild of each modified binary was conducted using the following code on local terminal, similar to the initial installation (see `README.md`).

```
rm -rf build # Removing previous built
mkdir build && cd build # Re-building with modified EMAlgorithm.h file
CMAKE_POLICY_VERSION_MINIMUM=3.5 cmake .. -DENABLE_AVX2=OFF
 ↪  -DCOMPILATION_ARCH=OFF -DCMAKE_POLICY_VERSION_MINIMUM=3.5

## Build (|| handles a possible race condition on first run)
CMAKE_POLICY_VERSION_MINIMUM=3.5 make -j$(sysctl -n hw.ncpu) || make
 ↪  -j$(sysctl -n hw.ncpu)

## Verification
./src/kallisto version
```

Parameters modification

A total of five alterations per convergence criteria were performed, with specific unit change based on their nature, resulting in a total of 25 trials. Minimum abundance threshold is estimated in transcript per million (TPM) and, due its magnitude, it was modified in the logarithmic scale. The maximum variance between iterations is measured as a percentage; hence, change rate was 0.5 percentage ranges with a minimum of 0.1%.

The five trials for each convergence criteria included the default value and other four unit alteration were defined in a standard deviation basis around the default. Each parameter value was assigned with an alphabet letter (abundance threshold = A-E; variance threshold = Z-V). The overview of the groups used for testing and exact parameter values can be found in Table 1 and Table 2. Additionally, a paired combination of letters was used as an output folder prefix of the kallisto trial for tracking purposes (Table 3). For instance, the quantification output folder with kallisto's default settings is named `/CX-output/`.

Table 1: Minimum expected transcripts abundance ($\alpha_t N$; `alpha_change_limit`) modifications for testing.

A	B	C (default)	D	E
1.0	0.1	0.01	0.001	0.0001

Table 2: Maximum variance between EM algorithm iterations (`alpha_change`) modifications for testing.

Z	Y	X (default)	W	V
0.1	0.5	1.0	1.5	2.0

Table 3: Prefix combinations for convergence criteria analysis.

	A	B	C (default)	D	E
V	AV	BV	CV	DV	EV
W	AW	BW	CW	DW	EW
X (default)	AX	BX	CX	DX	EX
Y	AY	BY	CY	DY	EY
Z	AZ	BZ	CZ	DZ	EZ

All kallisto trials were conducted using the same command for computational consistency.

```
kallisto/build/src/kallisto quant \
  -i empiricalInvestigation/gencode.v44.kidx \
  -o empiricalInvestigation/{prefix}-output \
  empiricalInvestigation/00-raw-data/sim_reads_1.fastq.gz \
  empiricalInvestigation/00-raw-data/sim_reads_2.fastq.gz
```

All results are stored in the `/empiricalInvestigation/` folder available at this github repository. Additionally, each EM algorithm alteration was mirrored into a copy-file present at same folder and was committed with output files for track changes purposes.

The output abundance file were further used for relative difference analysis between the different trials.

Relative difference analysis

The relative difference analysis aims to establish a relationship between each trial parameters and difference between estimated counts to ground truth (true counts). For this, each output abundance files were compared against the true counts provided by the professor. First, the unique transcripts identified in each trial were assessed using simple command line R (see code below).

```
# Loading libraries
library(tidyverse)
library(paletteer)
library(ggpubr)

# Define wd directory (empiricalInvestigation)
dir <- '/Users/dorianrojas-villalta/OneDrive - University of Calgary/2026
↪  Winter term/MDGE 610 - Foundations of
↪  Bioinformatics/MDGE610_final_project/empiricalInvestigation/'

# Uploading files
## Trial files
tsvFiles <- list.files(dir, pattern = '*.tsv', recursive = T, full.names = T)
files <- list()

for (file in tsvFiles) {
  df <- read_tsv(file)
  df <- df[, c('target_id', 'est_counts')]
  df$target_id <- gsub('\\|.*', '', df$target_id) # Extracting only
↪  transcript id
```

```r
  colnames(df)[1] <- 'transcript_id'

  # Keeping only the `<prefix>-abundance` name
  files[[sub('\\.tsv$', '', basename(file))]] <- df
}

## Ground truth file
groundTruth <- read_tsv(paste0(dir, 'sim_true_counts.txt'))

# Compare false positives
falsePos <- data.frame(
  file = names(files),
  counts = sapply(files, function(x) {
    diff <- anti_join(x, groundTruth, by = 'transcript_id')
    diff <- filter(diff, est_counts > 0)
    nrow(diff)
    })
  )

# Adding number of iterations. These were collected manually from the
↪  terminal output; unfortunately, they do not appear in the .json file.
falsePos$iterations <- c(174, 230, 411, 411, 1237, 379, 504, 867, 1280, 5842,
                         481, 683, 1095, 1825, 7828, 584, 1019, 1324, 2230,
                         10000, 687, 1050, 1552, 2615, 10000)

rownames(falsePos) <- NULL # Personal preference
```

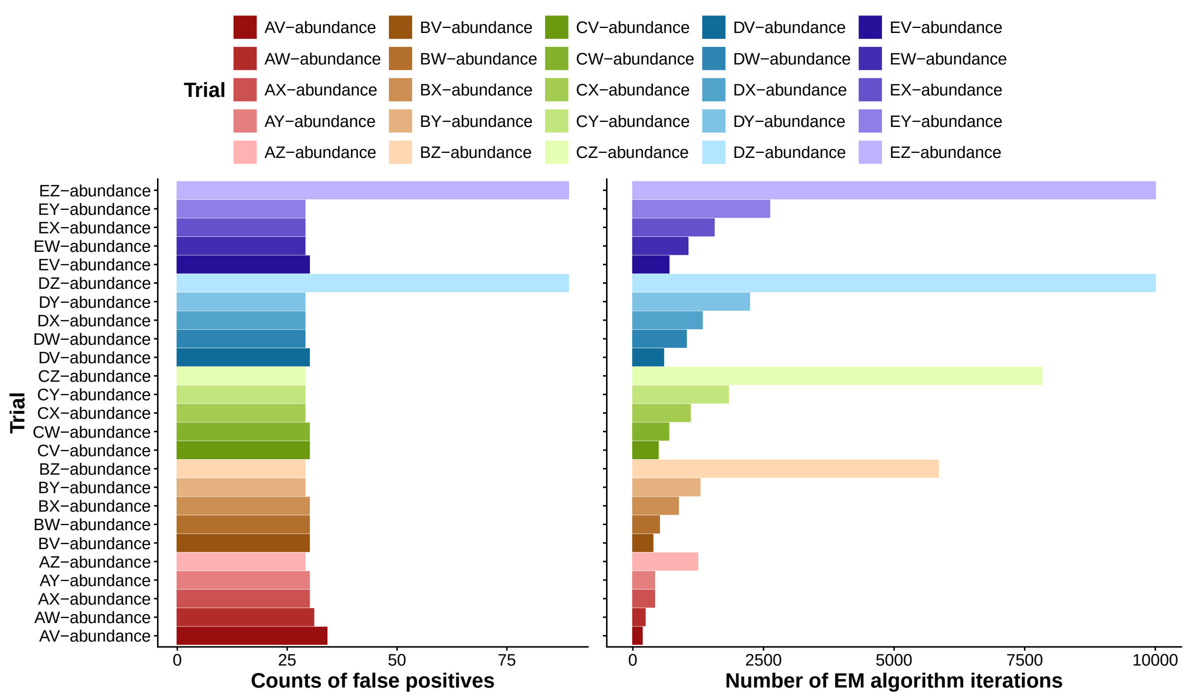
```r
theme <- theme_classic() +
  theme(legend.title = element_text(size = 15, face='bold'),
        legend.text = element_text(size = 12),
        axis.title = element_text(size = 15, face = 'bold'),
        axis.text = element_text(size = 12),
        strip.text = element_text(size = 12, face = 'bold'),
        legend.position = 'top')
set_theme(theme)


# Graphing
g1 <- ggplot(falsePos, aes(file, counts, colour = file, fill = file)) +
  geom_col() +
  labs(x = 'Trial',
       y = 'Counts of false positives',
```

```
      fill = 'Trial',
      colour = 'Trial') +
  scale_fill_paletteer_d('colorBlindness::SteppedSequential5Steps') +
  scale_colour_paletteer_d('colorBlindness::SteppedSequential5Steps') +
  coord_flip()

g2 <- ggplot(falsePos, aes(file, iterations, colour = file, fill = file)) +
  geom_col() +
  labs(x = 'Trial',
      y = 'Number of EM algorithm iterations',
      fill = 'Trial',
      colour = 'Trial') +
    scale_fill_paletteer_d('colorBlindness::SteppedSequential5Steps') +
  scale_colour_paletteer_d('colorBlindness::SteppedSequential5Steps') +
  coord_flip() +
   theme(
    axis.title.y = element_blank(), # Remove x-axis title
    axis.text.y = element_blank())

ggarrange(g1, g2, nrow = 1, common.legend = T)
```



Here, we aimed to identified the number of transcripts that were only found in the trials, but are not part of the true counts, indicating potential false positives. The results showed an overall persistence of false positives transcript identification across all trials, with a maximum of 89 and minimum of 29.

Interestingly, the highest number of false positives were detected in trials related to lower minimum abundance thresholds ($\alpha_t N = 0.001$ and 0.0001 for D and E trials respectively), and lower maximum variance between iterations (0.1% for Z trials). Decreasing the minimum abundance threshold reduces how conservative (stringent) the expectation maximization (EM) algorithm is at defining which transcripts are used for convergence assessment and optimizing the objective function. This consequent loss of EM algorithm's stringiness might be related, in part, to the higher number of false positive. Moreover, these trials reached the maximum number of iterations allowed by kallisto (10,000) before forceably stopping the EM algorithm, which indicates the possibility of not achieved a proper maximization of MLEs, resulting in lack of accuracy.

On the other hand, trials with fewer false positives varied, with at least one from each minimum abundance threshold setting (trials A to E), but were mostly represented by lower maximum variance threshold (1%, 0.5%, and 0.1% for X, Y, and Z trials, respectively). These preliminary results hypothesize that the maximum abundance variance between iteration might be the main driver of accuracy when compared to minimum abundance threshold in the kallisto software.

In order to understand if there is an association between the convergence criteria values and the difference in estimated counts and ground truth, a correlation and root mean square error (RMSE) analyses were conducted.

```
# Correlation and rmse
corrRMSE <- data.frame(
  file = names(files),
  coefficient = sapply(files, function(x) {

    ## Subsetting to remove false positives and normalise df dimensions
    inter = intersect(x$transcript_id, groundTruth$transcript_id)
    fileSubset <- subset(x, transcript_id %in% inter)

    corr <- cor(fileSubset$est_counts, groundTruth$true_counts, method =
↪   'spearman')
  }),
  rmse = sapply(files, function(x) {
    inter = intersect(x$transcript_id, groundTruth$transcript_id)
    fileSubset <- subset(x, transcript_id %in% inter)

    rmse <- sqrt(mean((fileSubset$est_counts - groundTruth$true_counts)^2))
  })
)
```

```r
rownames(corrRMSE) <- NULL # Personal preference

# Merging both sets into a final one for graphing
fullResults <- merge(falsePos, corrRMSE, by = 'file')

# Adding integer parameters
fullResults$minAbundance <- rep(c(1.0, 0.1, 0.01, 0.001, 0.0001), each = 5)
fullResults$maxVariance <- rep(c(2.0, 1.5, 1.0, 0.5, 0.1), 5)
```

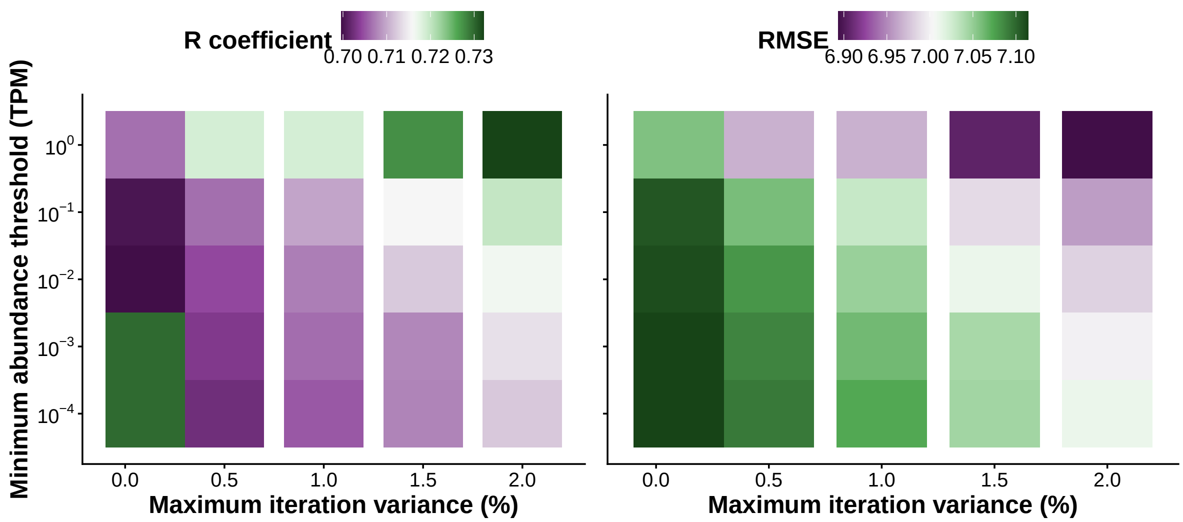
Correlation compares how the estimated transcript counts relate to true counts in a linear relationship. The correlation analysis output is a correlation coefficient (R) that indicates how are predicted and true counts associated. This ranges between 1, being high correlation (either positive [+1] or negative [-1]) and 0, meaning no correlation. This R coefficient can be used for assessing how the well the estimated counts track to the true counts. Moreover, to address the absolute differences between trial and ground truth, the RMSE wass estimated. This is a measure of the magnitude of errors, ranging from 0 to ∞ with values closer to 0 representing less errors (closer to perfect fit).

```r
# Graphing
g3 <- ggplot(fullResults, aes(maxVariance, minAbundance, fill = coefficient))
 ↪  +
  geom_tile() +
  labs(
    x = 'Maximum iteration variance (%)',
    y = 'Minimum abundance threshold (TPM)',
    fill = 'R coefficient') +
  scale_fill_paletteer_c('grDevices::PRGn') +
  scale_y_log10(
    breaks = scales::trans_breaks("log10", function(x) 10^x),
    labels = scales::trans_format("log10", scales::math_format(10^.x)))

g4 <- ggplot(fullResults, aes(maxVariance, minAbundance, fill = rmse)) +
  geom_tile() +
  labs(
    x = 'Maximum iteration variance (%)',
    y = 'Minimum abundance threshold (TPM)',
    fill = 'RMSE') +
  scale_fill_paletteer_c('grDevices::PRGn') +
  scale_y_log10(
    breaks = scales::trans_breaks("log10", function(x) 10^x),
    labels = scales::trans_format("log10", scales::math_format(10^.x))) +
  theme(
```

```r
    axis.title.y = element_blank(), # Remove x-axis title
    axis.text.y = element_blank()) +
  guides(fill = guide_colourbar(barwidth = 8))

ggarrange(g3, g4, legend = 'top')
```



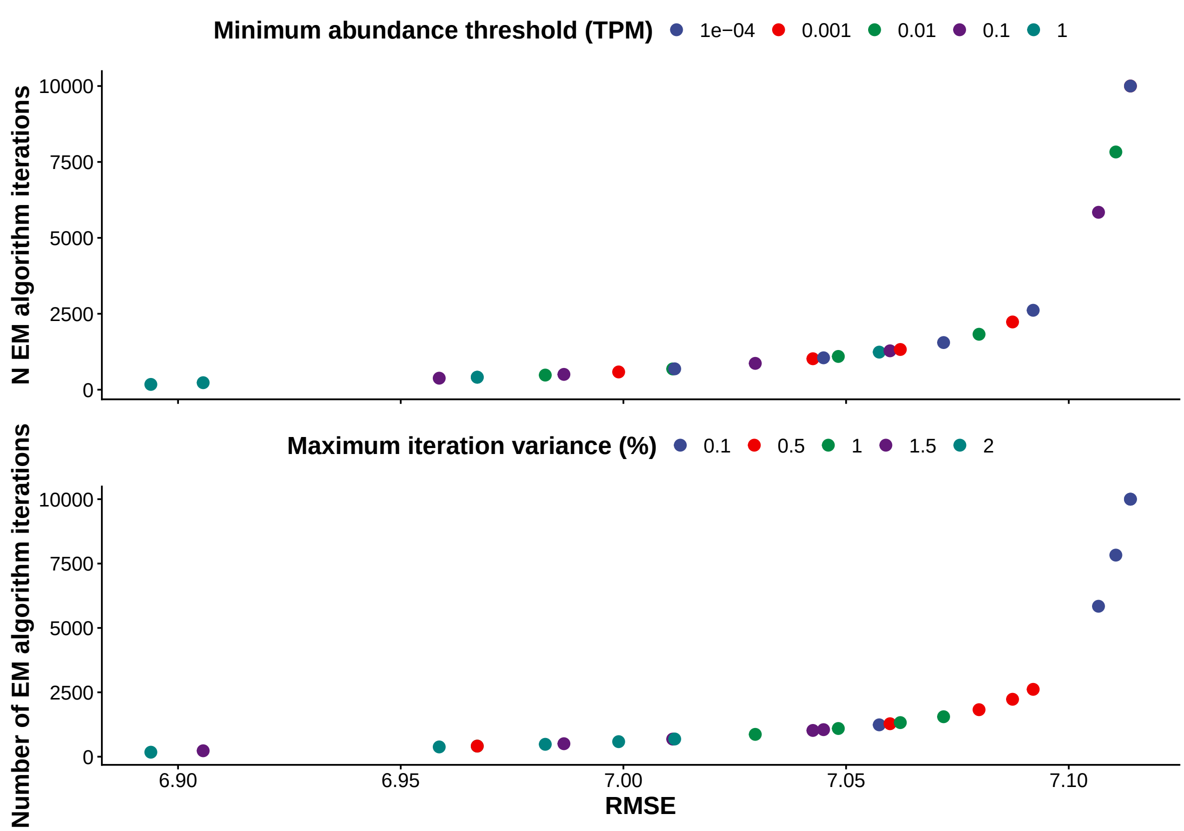
The range of both correlation coefficient and RMSE values is narrow, resulting is almost very similar statistics regardless of trial settings. However, some trends are noticeable in our finding. For instance, there is a high linear relationship among those less stringent trials (maximum iteration variance 1.5 and 2.0, and minimum abundance threshold of 1.0). Interestingly, the settings with higher number of false positives and maximum number of iterations also presented a high R coefficient. When combining this analysis with the RMSE values, it is evidence that less stringent criteria results in less magnitude of errors in comparison with more conservative values.

```r
g5 <- ggplot(fullResults, aes(rmse, iterations,
                        fill = factor(minAbundance),
                        colour = factor(minAbundance))) +
  geom_point(size = 3) +
  labs(
    x = 'RMSE',
    y = 'N EM algorithm iterations',
    fill = 'Minimum abundance threshold (TPM)',
    colour = 'Minimum abundance threshold (TPM)') +
    scale_fill_paletteer_d('ggsci::default_aaas') +
  scale_colour_paletteer_d('ggsci::default_aaas')+
```

```
  theme(
    axis.title.x = element_blank(), # Remove x-axis title
    axis.text.x = element_blank())

g6 <- ggplot(fullResults, aes(rmse, iterations,
                              fill = factor(maxVariance),
                              colour = factor(maxVariance))) +
  geom_point(size = 3) +
  labs(
    x = 'RMSE',
    y = 'Number of EM algorithm iterations',
    fill = 'Maximum iteration variance (%)',
    colour = 'Maximum iteration variance (%)') +
    scale_fill_paletteer_d('ggsci::default_aaas') +
  scale_colour_paletteer_d('ggsci::default_aaas')

ggarrange(g5, g6, nrow = 2)
```



Finally, we address how the number of iterations affect the RMSE values according to the convergence criteria settings. Interestingly, an increase in iteration number of the EM algorithm results in higher error magnitudes. No clear association to the convergence criteria settings is defined besides the previously mentioned decrease in RMSE with less stringent parameters.

Conclusions

- Decreasing the minimum abundance threshold reduces EM-algorithm conservative measures (less stringent), leading to a higher number of false positives. Also, this increases the number of iterations required to achieve convergence in the maximum likelihood estimates.
- Convergence criteria settings here tested resulted in small, almost neglectable, changes in both correlation coefficient and RMSE analysis. Beyond this, less stringent parameters resulted in higher values of these tests.
- Higher number of iteration in the EM-algorithm result in higher RMSE, indicating more difference between estimated values to ground truth.
- Overall, the here tested modifications to EM-algorithm convergence criteria do not seem to have an strong effect in relative differences between estimated data and true values. However, using less stringent setting might result in an increase number of false positives