R-code for ‘GCalignR: An R package for aligning Gas-Chromatography data’

Meinolf Ottensmann, Martin A. Stoffel, Hazel J. Nichols, Joseph I. Hoffman

This document provides all the `R code` used for our manuscript. Both the Rmarkdown file and the data can be accessed directly by downloading the accompanying GitHub repository. Just click on this link and then download a zip archive containing all the files. Make sure you read the instructions in the `README` file on GitHub. For computational reasons we provide the results of time consuming steps in addition to the raw data on GitHub. If you have any questions, do not hesitate to contact: meinolf.ottensmann@web.de

Prerequisites

Most functions that are used in this analysis are part of our package `GCalignR`, while some more functions are provided in form of `R` scripts that are available in the sub-directory `R`. In order to run the code you need to have a sub-directory called `data` that contains the raw datafiles.

- Install `devtools` , `ggplot2`, `plot3D` and `vegan` if these packages are not available.

```
# install devtools
if (!("devtools" %in% rownames(installed.packages()))) {
    install.packages("devtools")
} else if (packageVersion("devtools") < 1.6) {
    install.packages("devtools")
}
# install ggplot2
if (!"ggplot2" %in% rownames(install.packages())) {
    install.packages("ggplot2")
}
# install plot3D
if (!"plot3D" %in% rownames(installed.packages())) {
    install.packages("plot3D")
}
# install ptw
if (!"ptw" %in% rownames(installed.packages())) {
    install.packages("ptw")
}
# install vegan
if (!"vegan" %in% rownames(installed.packages())) {
    install.packages("vegan")
}
```

- Installing `GCalignR`.

```
# install GCalignR
install.packages("GCalignR")
```

- Load packages and source custom functions.

```
library(GCalignR)
library(ggplot2)
library(plot3D)
library(vegan)
```

```r
library(ptw)
# small function to test parameters in
# align_chromatograms
source("R/optimal_params.R")
# calculates errors by matching aligned data to a
# table of known substances
source("R/error_rate.R")
# custom function for simulations based on
# chromatograms
source("R/ChromaSimFunctions.R")
## functions for plotting
source("R/NMDS-Functions.R")
## convert data to be used with ptw
source("R/Convert2ptw.R")
## calculation of descriptive stats for a data frame
source("R/summary_stats.R")
```

Workflow of `GCalignR`

```r
library(GCalignR)  # loads the package
fpath <- system.file(dir = "extdata", file = "peak_data.txt",
    package = "GCalignR")  # path to peak_data.txt
check_input(fpath)  # checks the data
#> All checks passed!
```

In order to begin the alignment procedure, the following code needs to be executed:

```r
# align the chemical data
aligned_peak_data <- align_chromatograms(data = peak_data,
    rt_col_name = "time", max_diff_peak2mean = 0.02,
    min_diff_peak2peak = 0.08, max_linear_shift = 0.05,
    delete_single_peak = TRUE, blanks = c("C2", "C3"))
```

Afterwards, a summary of the alignment process can be retrieved using the printing method, which summarises the function call including defaults that were not altered by the user. This provides all of the relevant information to retrace every step of the alignment procedure.

```r
print(aligned_peak_data)  # verbal summary of the alignment procedure
#> Summary of Peak Alignment running align_chromatograms
#> Input: peak_data
#> Start:  2017-07-19 16:31:47  Finished:  2017-07-19 17:30:47
#>
#> Call:
#>   GCalignR::align_chromatograms(data=peak_data, rt_col_name=time,
#>   max_linear_shift=0.05, max_diff_peak2mean=0.02, min_diff_peak2peak=0.08,
#>   blanks=(C2, C3), delete_single_peak=T, sep=\t, rt_cutoff_low=NULL,
#>   rt_cutoff_high=NULL, reference=NULL)
#>
#> Summary of scored substances:
#>    total   blanks singular retained
#>      494      171       45      278
#>
#> In total 494 substances were identified among all samples. 171 substances were
#>   present in blanks. The corresponding peaks as well as the blanks were removed
#>   from the data. 45 substances were present in just one single sample and were
#>   removed. 278 substances are retained after all filtering steps.
#>
#> Sample overview:
#>   The following 84 samples were aligned to the reference 'P20':
#>   M2, M3, M4, M5, M6, M7, M8, M9, M10, M12, M14, M15, M16, M17, M18, M19, M20,
#>   M21, M23, M24, M25, M26, M27, M28, M29, M30, M31, M33, M35, M36, M37, M38, M39,
#>   M40, M41, M43, M44, M45, M46, M47, M48, P2, P3, P4, P5, P6, P7, P8, P9, P10,
#>   P12, P14, P15, P16, P17, P18, P19, P20, P21, P23, P24, P25, P26, P27, P28, P29,
#>   P30, P31, P33, P35, P36, P37, P38, P39, P40, P41, P43, P44, P45, P46, P47, P48
#>
#> For further details type:
#>   'gc_heatmap(aligned_peak_data)' to retrieve heatmaps
#>   'plot(aligned_peak_data)' to retrieve further diagnostic plots
```

As alignment quality may vary with the parameter values selected by the user, the plot function can be used to output four diagnostic plots. These allow the user to explore how the parameter values affect the resulting alignment and can help flag issues with the raw data.

```
plot(aligned_peak_data)  # Figure 1
```

Additionally, the full alignment can be visualised inspected using a heat map with the function gc_heatmap.

```
gc_heatmap(aligned_peak_data, type = "binary", threshold = 0.05)  # Figure 2
```

Peak normalisation

In order to account for differences in the total concentration of samples, we provide an additional function `norm_peaks` that can be used to normalise peak abundances.

```
scent <- norm_peaks(data = aligned_peak_data, rt_col_name = "time",
    conc_col_name = "area", out = "data.frame")
```

Downstream analyses

The output of GCalignR is compatible with other functionalities in R, thereby providing a seamless transition between packages. For instance, multivariate analyses can be conduced using the package vegan (Oksanen *et al.* 2016). To visualise patterns of chemical similarity within the fur seal dataset in relation to breeding colony membership, we implemented non-metric-multidimensional scaling (NMDS) based on a Bray-Curtis dissimilarity matrix and visualised the outcome using ggplot2 (Wickham 2009).

```
scent <- scent[match(row.names(peak_factors), row.names(scent)),
    ]  # sort data
scent <- log(scent + 1)  # log + 1 transformation
```

```
scent_nmds <- vegan::metaMDS(comm = scent, distance = "bray")  # NMDS
scent_nmds <- as.data.frame(scent_nmds[["points"]])  # extract points
scent_nmds <- cbind(scent_nmds, colony = peak_factors[["colony"]])  # add factors
```

```
# Figure 3
ggplot(data = scent_nmds, aes(MDS1, MDS2, color = colony)) +
    geom_point() + theme_void() + scale_color_manual(values = c("blue",
    "red")) + theme(panel.background = element_rect(colour = "black",
    size = 1.25, fill = NA), aspect.ratio = 1, legend.position = "none")
```

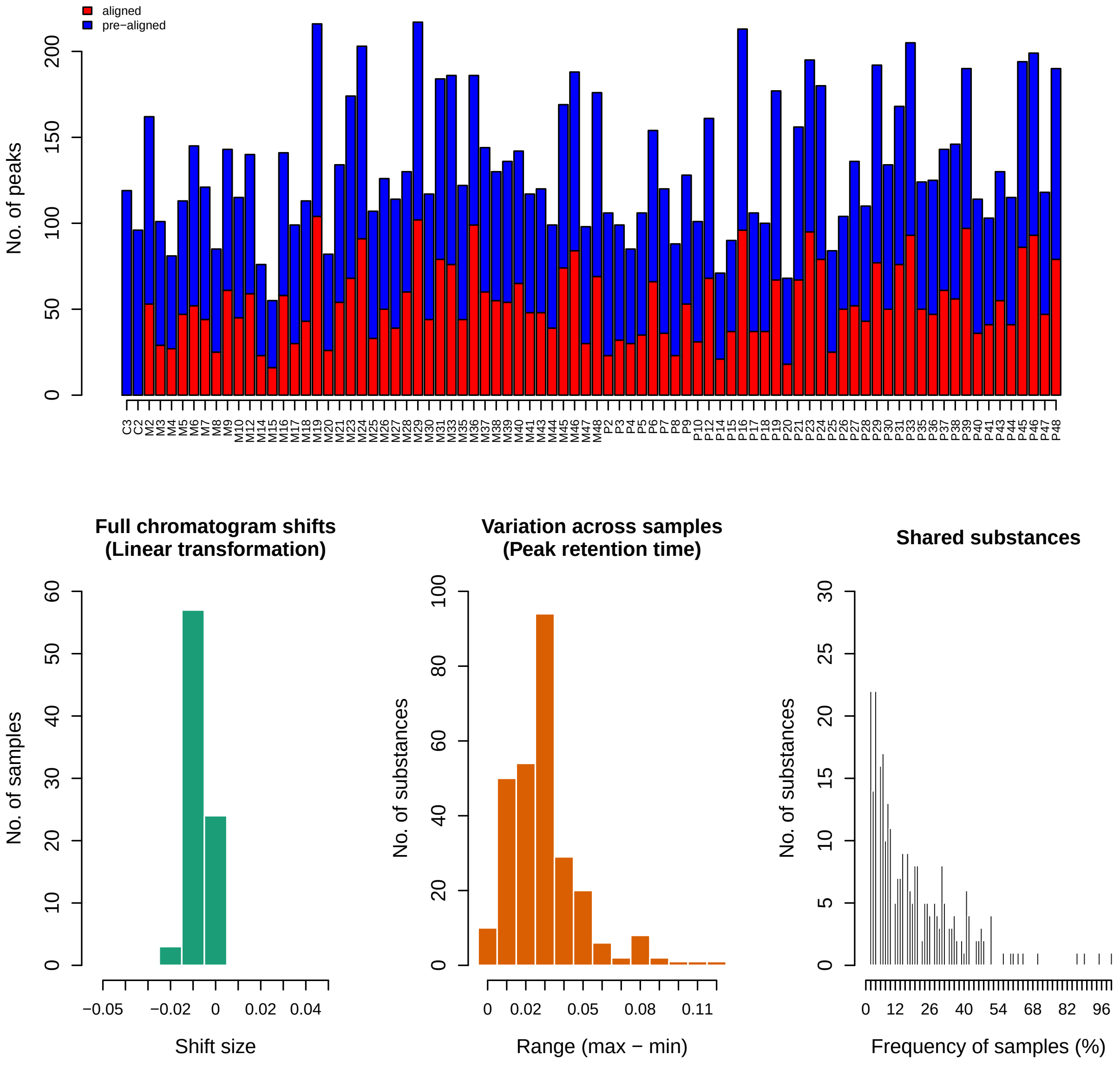


Figure 1: Diagnostic plots summarise aligned datasets

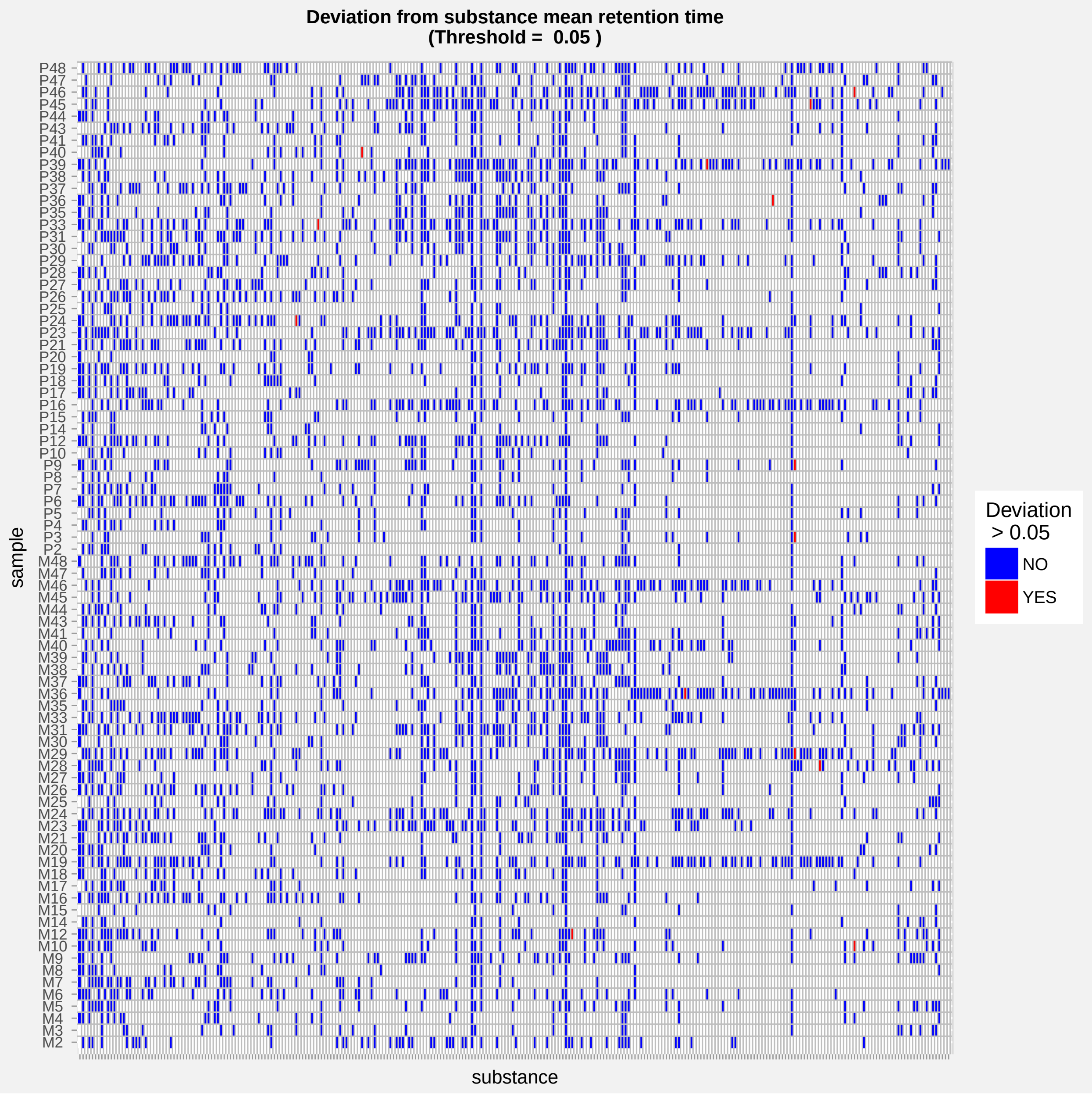


Figure 2: Heatmaps allow to inspect the distribution of substances across samples as well as the variability of their retention times.

Evaluation of the algorithm performance

Aligning the earwig dataset

The earwig dataset (Wong et al. 2014) can be dowloaded on Dryad and is available as a text file on this GitHub repository.

```
earwig_1 <- align_chromatograms(data = "data/earwig.txt",
    rt_col_name = "RT", max_linear_shift = 0.05, max_diff_peak2mean = 0.05,
    min_diff_peak2peak = 0.2)
save(earwig_1, file = "data/earwig_1.RData")
```

```
## load aligning results
load("data/earwig_1.RData")
## inspect the distribution of peaks for the
## complete dataset gc_heatmap(earwig_1) inspect a
## representative subset
a <- gc_heatmap(earwig_1, label = "xy", label_size = 12,
    main_title = "A", show_legend = F, samples_subset = c(1:16,
        21, 167, 174, 285), threshold = 0.2)
a <- a + theme(axis.title.x = element_blank(), axis.title.y = element_blank(),
    plot.title = element_text(hjust = 0), plot.background = element_rect(fill = "white"))
print(a)
```

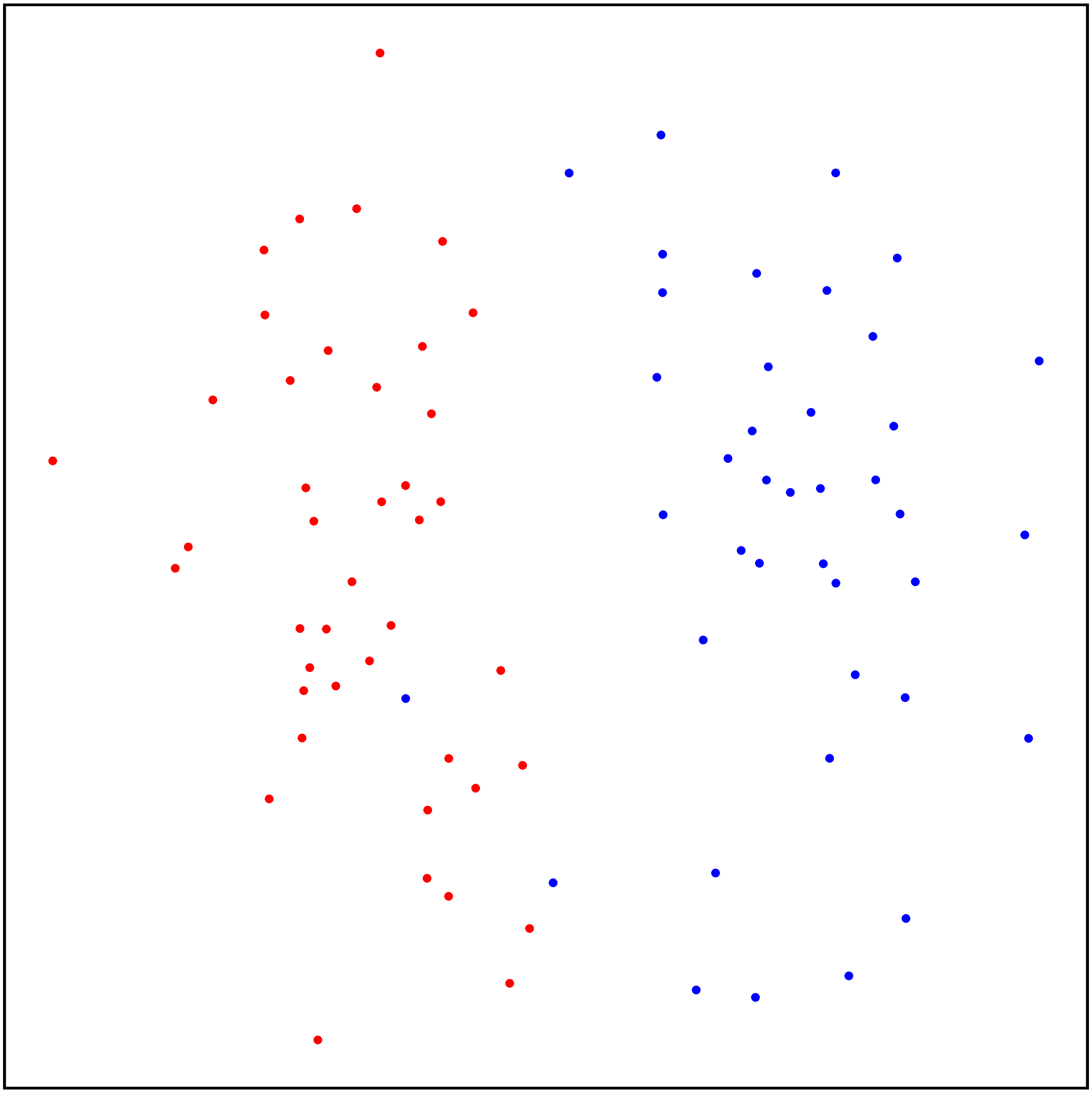
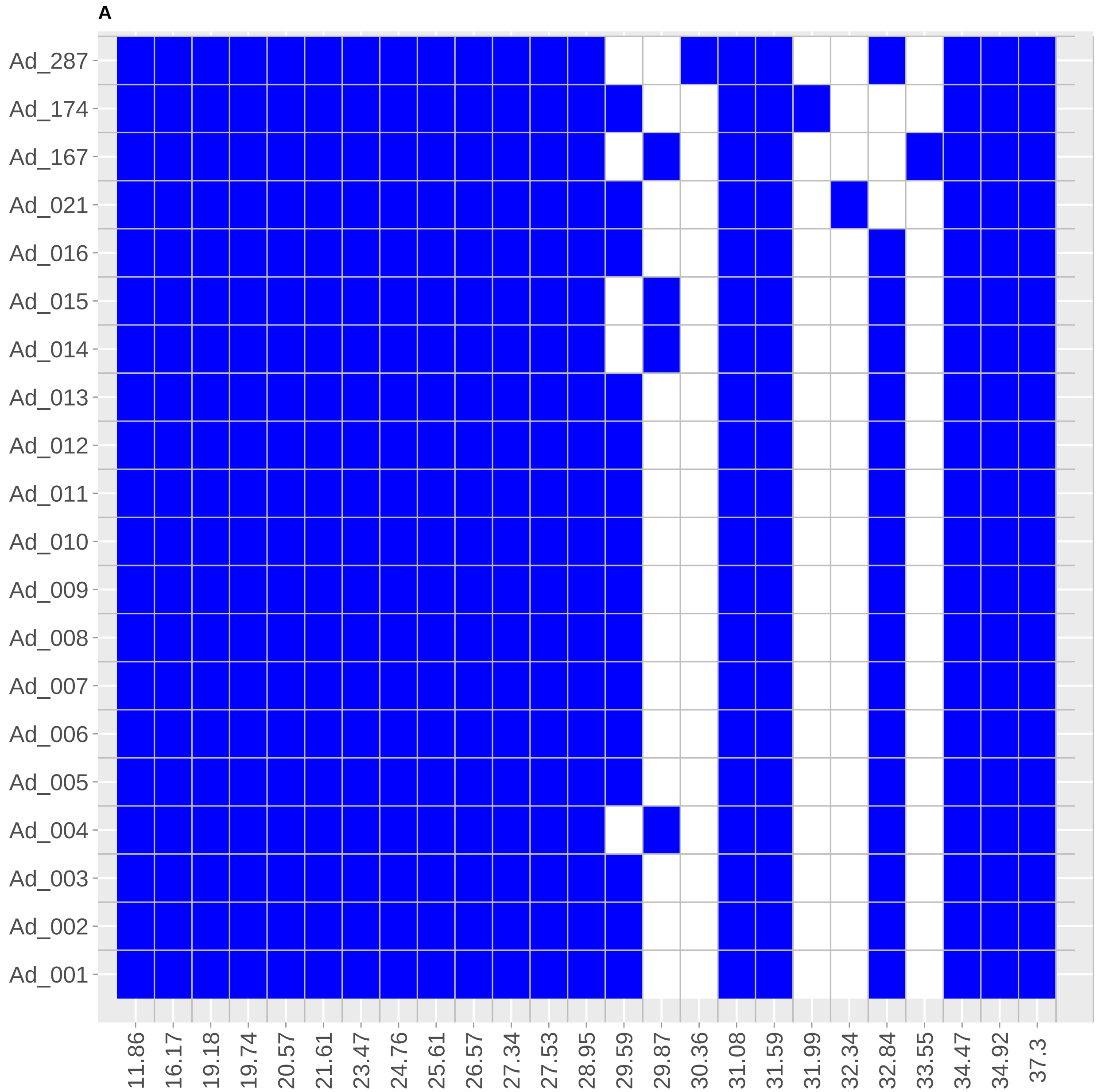


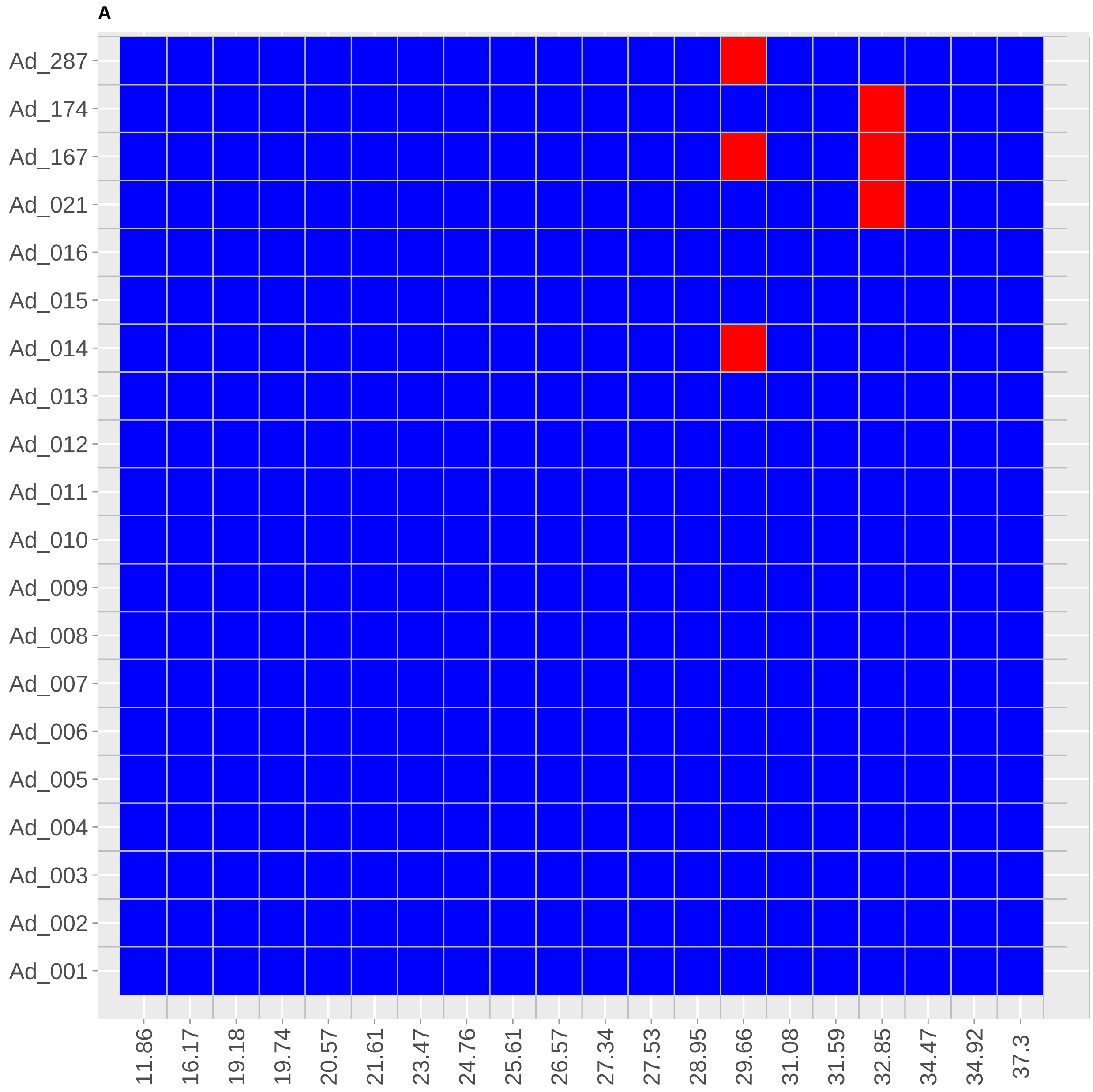
Figure 3: NMDS plot showing the cluserting by colony



The patterns shown by the heatmaps suggest to optimise the alignment by allowing to merge rows with larger deviation in retention times

```
earwig_2 <- align_chromatograms(data = "data/earwig.txt",
    rt_col_name = "RT", max_linear_shift = 0.05, max_diff_peak2mean = 0.05,
    min_diff_peak2peak = 0.75)
save(earwig_2, file = "data/earwig_2.RData")
```

```
## load aligning results
load("data/earwig_2.RData")
## inspect the distribution of peaks for the
## complete dataset gc_heatmap(earwig_2) inspect a
## representative subset
b <- gc_heatmap(earwig_2, label = "xy", label_size = 12,
    main_title = "A", show_legend = F, samples_subset = c(1:16,
        21, 167, 174, 285), threshold = 0.2)
b <- b + theme(axis.title.x = element_blank(), axis.title.y = element_blank(),
    plot.title = element_text(hjust = 0), plot.background = element_rect(fill = "white"))
print(b)
```



Explore the parameter space in `align_chromatograms`

There are two parameters of major importance, namely `max_diff_peak2mean` and `min_diff_peak2peak`. While the first determines the finescale grouping of retention times the latter greatly influences the formation of substances by combining initially separated rows of similar retention times. Here, we evaluate the error rate as a function of the combination of these two parameters. The combinations are tested by iteratively running `aling_chromatograms` 100 parameter combinations.

- Run alignments with all combinations of both parameters

```
# B. flavifrons
results_bfla <- optimal_params(data = "data/bfla.txt",
    rt_col_name = "RT", max_diff_peak2mean = seq(from = 0.01,
        to = 0.1, by = 0.01), min_diff_peak2peak = seq(from = 0.01,
        to = 0.4, by = 0.01))
save(results_bfla, file = "data/results_bfla.RData")

# B. bimaculatus
results_bbim <- optimal_params(data = "data/bbim.txt",
    rt_col_name = "RT", max_diff_peak2mean = seq(from = 0.01,
        to = 0.1, by = 0.01), min_diff_peak2peak = seq(from = 0.01,
        to = 0.4, by = 0.01))
save(results_bbim, file = "data/results_bbim.RData")

# B. ephippiatus
results_beph <- optimal_params(data = "data/beph.txt",
    rt_col_name = "RT", max_diff_peak2mean = seq(from = 0.01,
        to = 0.1, by = 0.01), min_diff_peak2peak = seq(from = 0.01,
        to = 0.4, by = 0.01))
save(results_beph, file = "data/results_beph.RData")
```

- Estimate error rates

Error rate calculations are executed with a custom function `error_rate` that uses a list of annotated substances as a reference. See the code for details.

```
# Load data
load("data/results_bbim.RData")
load("data/results_beph.RData")
load("data/results_bfla.RData")

errors_bbim <- data.frame(p2p = results_bbim[[2]][["p2p"]],
    p2m = results_bbim[[2]][["p2m"]])

errors_bbim[["error"]] <- unlist(lapply(X = results_bbim[[1]],
    error_rate, "data/bbim_ms.txt"))

errors_beph <- data.frame(p2p = results_beph[[2]][["p2p"]],
    p2m = results_beph[[2]][["p2m"]])

errors_beph[["error"]] <- unlist(lapply(X = results_beph[[1]],
    error_rate, "data/beph_ms.txt"))

errors_bfla <- data.frame(p2p = results_bfla[[2]][["p2p"]],
    p2m = results_bfla[[2]][["p2m"]])

errors_bfla[["error"]] <- unlist(lapply(X = results_bfla[[1]],
    error_rate, "data/bfla_ms.txt"))

save(errors_bbim, file = "data/errors_bbim.RData")
save(errors_beph, file = "data/errors_beph.RData")
save(errors_bfla, file = "data/errors_bfla.RData")
```

- Plot results using package `plot3D`

```
# Figure 4
with(errors_bbim, scatter3D(x = p2p, y = p2m, z = error,
    pch = 19, size = 2, theta = 30, phi = 0, ticktype = "detailed",
    main = "", xlab = "min_diff_peak2peak", ylab = "max_diff_peak2mean",
    zlab = "Error rate", bty = "g", colkey = FALSE,
    cex = 1, cex.lab = 1, cex.axis = 1, cex.main = 1.5,
    zlim = c(0, 0.2)))
```

```
# Figure 5
with(errors_beph, scatter3D(x = p2p, y = p2m, z = error,
    pch = 19, size = 2, theta = 30, phi = 0, ticktype = "detailed",
    main = "", xlab = "min_diff_peak2peak", ylab = "max_diff_peak2mean",
    zlab = "Error rate", bty = "g", colkey = FALSE,
    cex = 1, cex.lab = 1, cex.axis = 1, cex.main = 1.5,
    zlim = c(0, 0.2)))
```

```
# Figure 6
with(errors_bfla, scatter3D(x = p2p, y = p2m, z = error,
    pch = 19, size = 2, theta = 30, phi = 0, ticktype = "detailed",
    main = "", xlab = "min_diff_peak2peak", ylab = "max_diff_peak2mean",
    zlab = "Error rate", bty = "g", colkey = FALSE,
    cex = 1, cex.lab = 1, cex.axis = 1, cex.main = 1.5,
    zlim = c(0, 0.2)))
```

Validation based on error rates of known substances from three bumblebee datasets

To further assess the performance of GCalignR, we calculated alignment error rates based on three previously published bumblebee dataset comprising known substances identified using GC-MS (Dellicour & Lecocq 2013). The first dataset comprises 24 *Bombus bimaculatus* individuals characterised for 32 substances (total = 717 retention times). The second comprises 20 *B. ephippiatus* individuals characterised for 42 substances (total = 782 retention times) and the third comprises 11 *B. flavifrons* individuals characterised for 44 substances (total = 457 retention times). We calculated the error rate as the ratio between incorrectly assigned retention times and the total number of retention times (equation (1)).

$$\mathrm{rt_m} > \left(\frac{\sum_{\mathrm{i=1}}^{\mathrm{m-1}} \mathrm{rt_i}}{\mathrm{m-1}}\right) + \mathrm{max\_diff\_peak2mean} \qquad (1)$$

By systematically changing the two parameters `max_diff_peak2mean` and `min_diff_peak2peak`, we explored 100 parameter combinations to investigate how parameter values affect the alignment accuracy.

- We align that untreated dataset in order to extract input retention times

```
bbim_zero <- align_chromatograms(data = "data/bbim.txt",
    rt_col_name = "RT")
save(bbim_zero, file = "data/bbim_zero.RData")
beph_zero <- align_chromatograms(data = "data/beph.txt",
    rt_col_name = "RT")
save(beph_zero, file = "data/beph_zero.RData")
bfla_zero <- align_chromatograms(data = "data/bfla.txt",
    rt_col_name = "RT")
save(bfla_zero, file = "data/bfla_zero.RData")
```

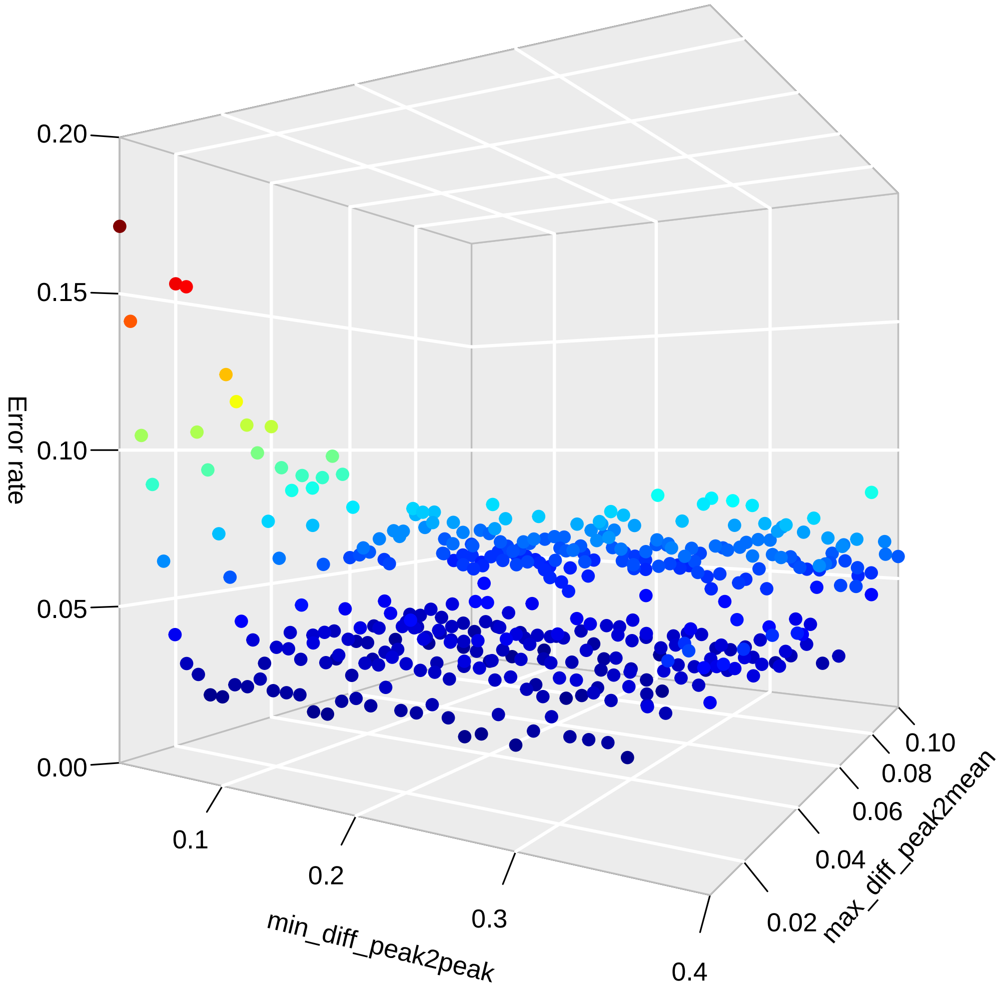


Figure 4: Parameter combinations B. bimaculatus dataset

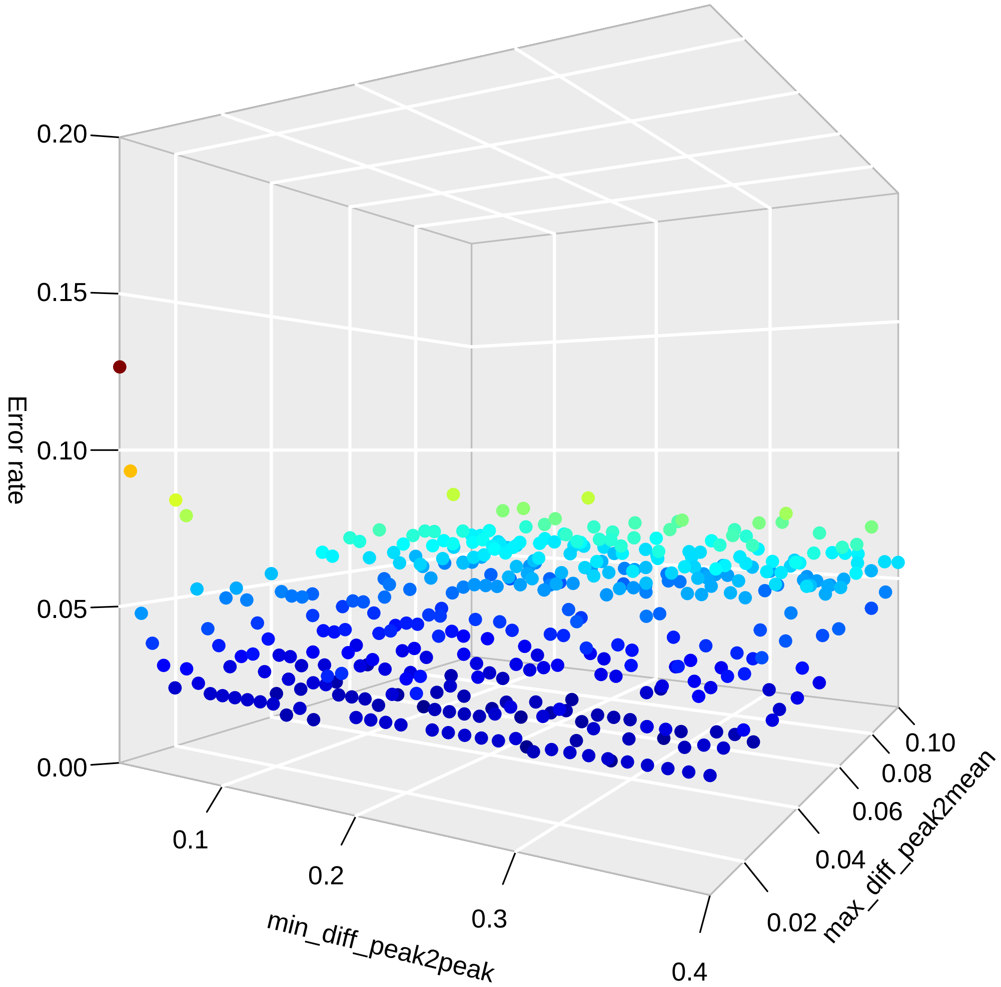


Figure 5: Parameter combinations B. ephippiatus dataset

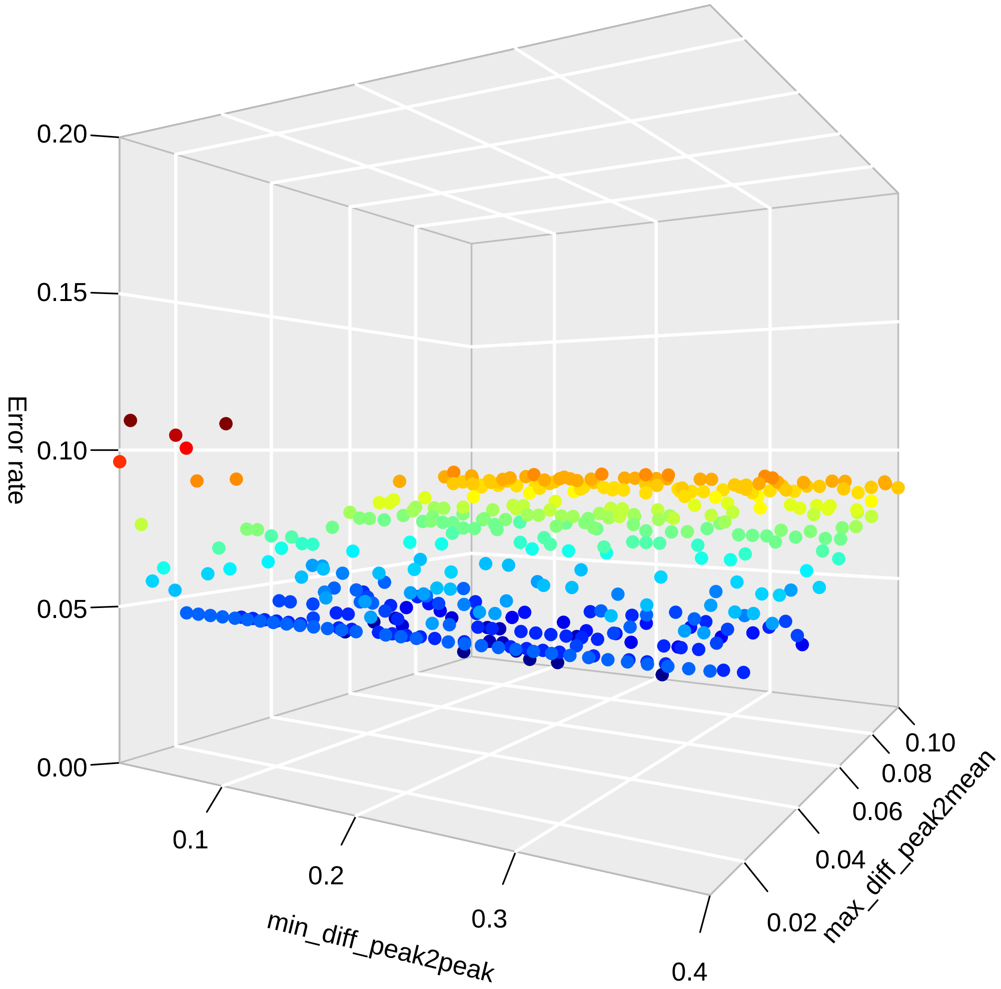


Figure 6: Parameter combinations B. flavifrons dataset

- Prepare the data for simulations

```r
load("data/bbim_zero.RData")
load("data/beph_zero.RData")
load("data/bfla_zero.RData")

bfla_chroma <- lapply(bfla_zero[["input_list"]], na.remove)  # remove NAs
bbim_chroma <- lapply(bbim_zero[["input_list"]], na.remove)  # remove NAs
beph_chroma <- lapply(beph_zero[["input_list"]], na.remove)  # remove NAs
```

```r
# Bombums flavifrons ------------------
bfla_out <- sim_linear_shift(bfla_chroma, rt_col_name = "RT",
    shifts = c(-0.03, 0.03))
bfla_shifted <- bfla_out[["Chromas"]]

p <- rep(seq(from = 0, to = 1, by = 0.1), each = 10)
bfla_data <- list()
names <- character()
for (i in 1:length(p)) {
    # add errors
    temp <- lapply(bfla_shifted, add_peak_error, p = p[i],
        rt_col_name = "RT", conc_col_name = "Area",
        distr = c(-0.02, -0.01, 0.01, 0.02))
    # extract peak list
    temp <- lapply(temp, FUN = function(x) x[["chroma"]])
    aligned <- align_chromatograms(temp, rt_col_name = "RT",
        max_linear_shift = 0.05, min_diff_peak2peak = 0.3)
    # We need the 'true' retention times for
    # referencing purposes
    aligned <- original_rt(org = bfla_chroma, aligned = aligned,
        rt_col_name = "RT")
    bfla_data <- append(bfla_data, list(aligned))
    names <- c(names, paste0("no_", as.character(i),
        "_noise_", as.character(p[i])))
}
names(bfla_data) <- names
bfla_simulations <- list(OptAlign = bfla_zero, SimAlign = bfla_data,
    noise = p)
save(x = bfla_simulations, file = paste0("data/", "bfla_simulations",
    ".RData"))
# ------------------

# Bombus bimaculatus ------------------
bbim_out <- sim_linear_shift(bbim_chroma, rt_col_name = "RT",
    shifts = c(-0.03, 0.03))
bbim_shifted <- bbim_out[["Chromas"]]  # linearly shifted sample

p <- rep(seq(from = 0, to = 1, by = 0.1), each = 10)
bbim_data <- list()
names <- character()
for (i in 1:length(p)) {
    # add errors
    temp <- lapply(bbim_shifted, add_peak_error, p = p[i],
        rt_col_name = "RT", conc_col_name = "Area",
        distr = c(-0.02, -0.01, 0.01, 0.02))
    # extract peak list
    temp <- lapply(temp, FUN = function(x) x[["chroma"]])
    aligned <- align_chromatograms(temp, rt_col_name = "RT",
        max_linear_shift = 0.05, min_diff_peak2peak = 0.3)
    # We need the 'true' retention times for
    # referencing purposes
    aligned <- original_rt(org = bbim_chroma, aligned = aligned,
        rt_col_name = "RT")
    bbim_data <- append(bbim_data, list(aligned))
    names <- c(names, paste0("no_", as.character(i),
        "_noise_", as.character(p[i])))
}
names(bbim_data) <- names
bbim_simulations <- list(OptAlign = bbim_zero, SimAlign = bbim_data,
    noise = p)
save(x = bbim_simulations, file = paste0("data/", "bbim_simulations",
    ".RData"))
# ------------------

# Bombus ephippiatus ------------------

beph_out <- sim_linear_shift(beph_chroma, rt_col_name = "RT",
    shifts = c(-0.03, 0.03))
beph_shifted <- beph_out[["Chromas"]]  # linearly shifted sample

p <- rep(seq(from = 0, to = 1, by = 0.1), each = 10)
beph_data <- list()
names <- character()
for (i in 1:length(p)) {
    # add errors
    temp <- lapply(beph_shifted, add_peak_error, p = p[i],
        rt_col_name = "RT", conc_col_name = "Area",
        distr = c(-0.02, -0.01, 0.01, 0.02))
    # extract peak list
    temp <- lapply(temp, FUN = function(x) x[["chroma"]])
    aligned <- align_chromatograms(temp, rt_col_name = "RT",
        max_linear_shift = 0.05, min_diff_peak2peak = 0.3)
    # We need the 'true' retention times for
    # referencing purposes
    aligned <- original_rt(org = beph_chroma, aligned = aligned,
        rt_col_name = "RT")
    beph_data <- append(beph_data, list(aligned))
    names <- c(names, paste0("no_", as.character(i),
        "_noise_", as.character(p[i])))
}
names(beph_data) <- names
beph_simulations <- list(OptAlign = beph_zero, SimAlign = beph_data,
    noise = p)
save(x = beph_simulations, file = paste0("data/", "beph_simulations",
    ".RData"))
# ------------------
```

- Calculate error rates

```r
load("data/bfla_simulations.RData")
load("data/beph_simulations.RData")
load("data/bbim_simulations.RData")
# set up data frames
bfla <- data.frame(data.frame(noise = bfla_simulations[["noise"]]))
bbim <- data.frame(data.frame(noise = bbim_simulations[["noise"]]))
beph <- data.frame(data.frame(noise = beph_simulations[["noise"]]))
# calculate errors
bfla[["error"]] <- unlist(lapply(X = bfla_simulations[["SimAlign"]],
    error_rate, Reference = "data/bfla_ms.txt", rt_col_name = "RT",
    linshift = FALSE))
bbim[["error"]] <- unlist(lapply(X = bbim_simulations[["SimAlign"]],
    error_rate, Reference = "data/bbim_ms.txt", rt_col_name = "RT",
    linshift = FALSE))
beph[["error"]] <- unlist(lapply(X = beph_simulations[["SimAlign"]],
    error_rate, Reference = "data/beph_ms.txt", rt_col_name = "RT",
    linshift = FALSE))
# Combine data into one data frame
df_bumblebee <- rbind(bbim, bfla, beph)
df_bumblebee[["id"]] <- rep(c("bbim", "bfla", "beph"),
    each = nrow(df)/3)
save(df_bumblebee, file = "data/df_bumblebee.RData")
```

- Plot the results

```r
load("data/df_bumblebee.RData")
df_bumblebee[["id"]] <- plyr::revalue(df_bumblebee[["id"]],
    c(bbim = "Bombus bimaculatus", beph = "B. ephippiatus",
        bfla = "B. flavifrons"))
df_bumblebee[["id"]] <- factor(df_bumblebee[["id"]],
    levels = c("Bombus bimaculatus", "B. ephippiatus",
        "B. flavifrons"))
# Figure 8
ggplot(df_bumblebee, aes(x = noise, y = error, group = id,
    col = id, fill = id)) + facet_wrap(~id) + geom_smooth(size = 1.5,
    se = T, aes(group = id)) + geom_boxplot(alpha = 0.3,
    size = 0.1, weight = 1, aes(group = noise)) + theme_bw(base_size = 14,
    base_family = "sans") + theme(panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(), axis.text.x = element_text(angle = 90,
        vjust = 0.5), aspect.ratio = 1, legend.position = "none",
    strip.text = element_text(face = "italic")) + xlab("Additional noise level") +
    ylab("Error rate") + scale_x_continuous(breaks = seq(0,
    1, 0.1), expand = c(0, 0)) + scale_y_continuous(breaks = seq(0,
    0.3, 0.02), expand = c(0, 0)) + scale_colour_manual(values = c("#1B9E77",
    "#D95F02", "#7570B3")) + scale_fill_manual(values = c("#1B9E77",
    "#D95F02", "#7570B3"), guide = guide_legend(label.theme = element_text(face = "italic",
    angle = 0, size = 12)))
#> `geom_smooth()` using method = 'loess'
```

Testing parametric time warping for the alignment of the earwig dataset

- load the data

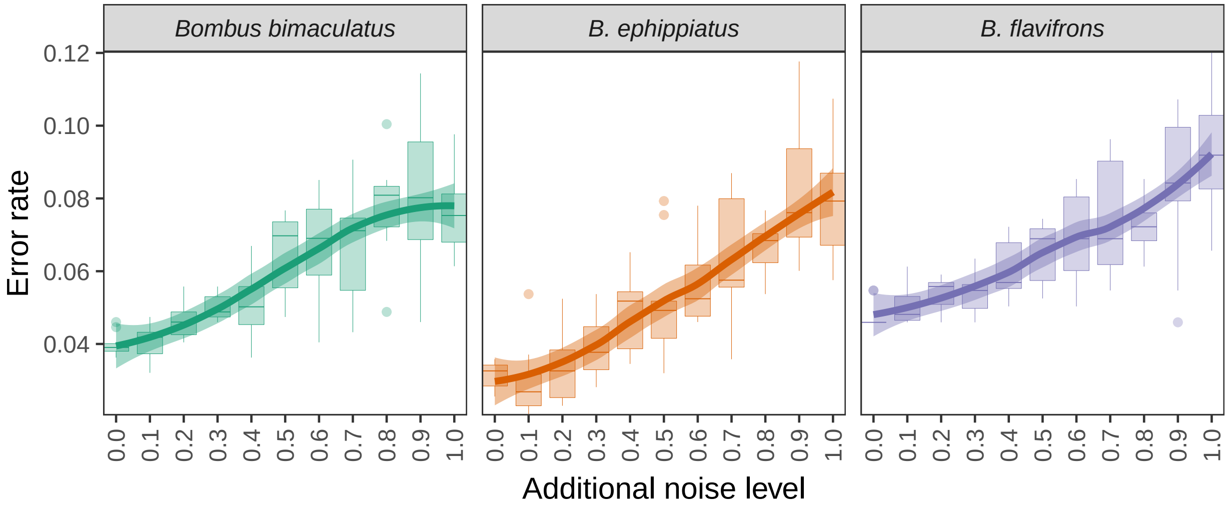


Figure 7: Effects of additional noise on error rates

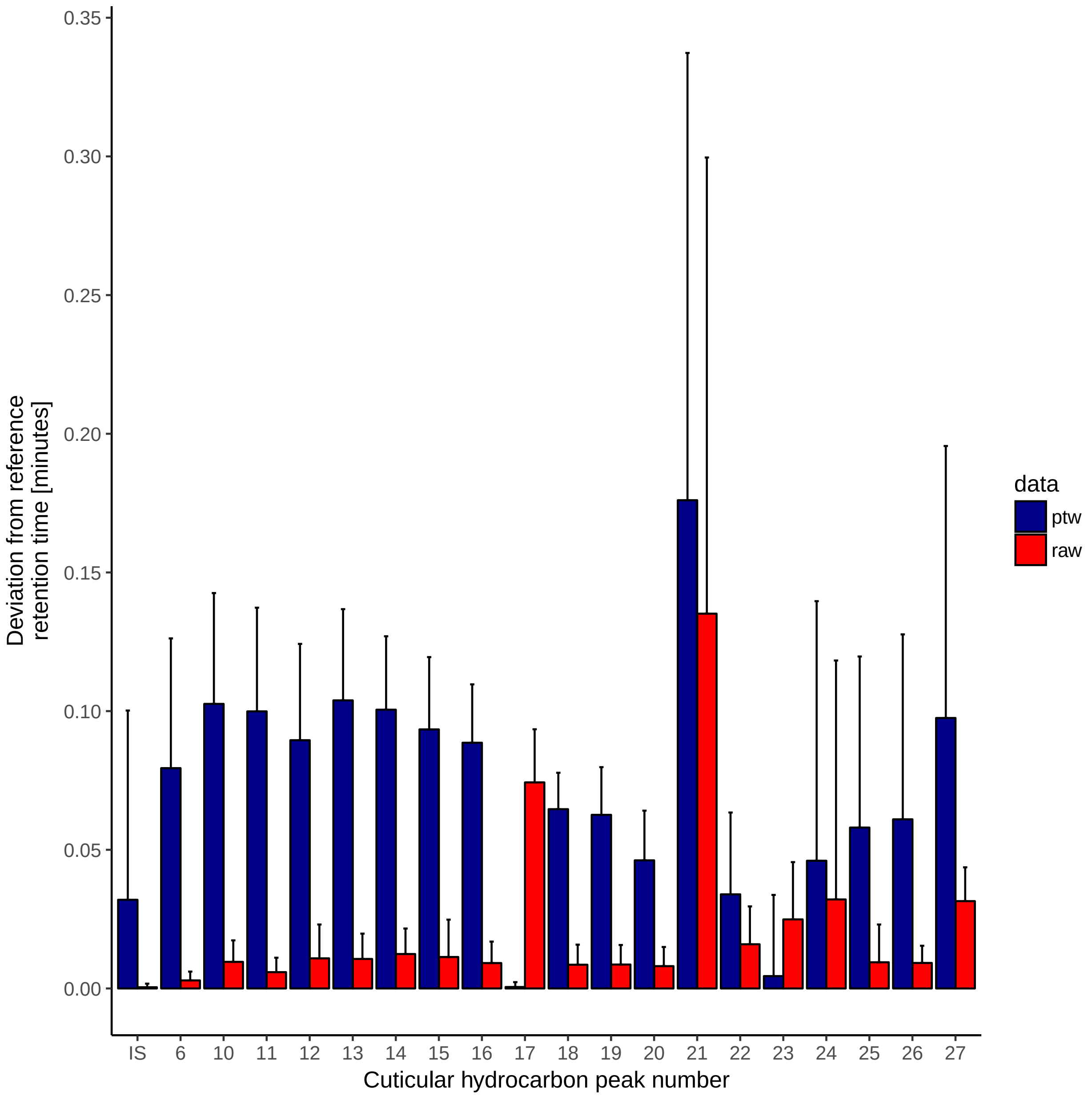
```
## format the data
dat <- convert2ptw(data = "data/earwig.txt", rt_col_name = "RT",
    conc_col_name = "Response")
## dat is a list of samples
class(dat)
#> [1] "list"
## each list contains a matrix
str(dat$Ad_001)
#>  num [1:20, 1:2] 11.9 16.2 19.2 19.7 20.6 ...
#>  - attr(*, "dimnames")=List of 2
#>   ..$ : chr [1:20] "1" "2" "3" "4" ...
#>   ..$ : chr [1:2] "rt" "I"
## specify a reference sample
ref <- "Ad_001"
## get the index of ref in dat
index <- which(names(dat) == ref)
## extract a reference
refst <- dat[index]
## remove the reference from the list of samples
sampst <- dat[-index]
```

- optimise the parameter trwdth Prior to aligning the dataset we optimise the WCC criterion, expressed by the parameter `trwdth`. See `?stptw` for details. Based on the knowledge that all peaks are shared between reference and sample. We can pick the value of `trwdth` that yields to best alignment performance.

```
## preallocate a data frame to store results
opt.crit <- data.frame(wcc = 0, sum_dev = 0)
## set up a vector of values to test
wcc <- c(seq(0.1, 0.9, 0.1), 1:100)
## loop over all values
for (i in 1:length(wcc)) {
    ## conduct the warping
    ptw_out <- stptw(refst, sampst[1], trwdth = wcc[i])
    ## extract results
    opt.crit[i, ] <- c(wcc[i], sum(abs(ptw_out$warped.sample[[1]][,
        1] - ptw_out$reference[[1]][, 1])))
}
## the first row contains the value of wcc yielding
## to the best result
head(opt.crit[order(opt.crit$sum_dev, decreasing = F),
    ])
#>    wcc  sum_dev
#> 11 2.0 1.279437
#> 12 3.0 1.334910
#> 7  0.7 1.445403
#> 5  0.5 1.446737
#> 8  0.8 1.446739
#> 9  0.9 1.451569
```

- Align all samples

```
ptw_out <- lapply(sampst, function(x) stptw(refst,
    list(x), trwdth = 2))

## Estimate deviations between all homologous peaks
## with respect to the reference
aligned <- do.call("rbind", lapply(ptw_out, function(fx) {
    temp <- abs(as.vector(fx$warped.sample[[1]][, 1]) -
        as.vector(fx$reference[[1]][, 1]))
}))

## Obtain deviation from the raw for comparison
input <- do.call("rbind", lapply(ptw_out, function(fx) {
    temp <- abs(as.vector(fx$sample[[1]][, 1]) - as.vector(fx$reference[[1]][,
        1]))
}))

## get peak names for cross-reference to Wong et al.
## 2014
peaks <- c("IS", as.character(read_peak_list(data = "data/earwig.txt",
    rt_col_name = "RT")[[1]][2:20, 1]))

## convert to data frames
df1 <- as.data.frame(aligned)
names(df1) <- peaks
df1 <- suppressMessages(reshape2::melt(df1))
df2 <- as.data.frame(input)
names(df2) <- peaks
df2 <- suppressMessages(reshape2::melt(df2))
df1$data <- "ptw"
df2$data <- "raw"

## merge data frames
df <- rbind(df1, df2)
names(df) <- c("Peak", "diff", "data")
## calculate mean and standard deviations
df <- summary_stats(data = df, measurevar = "diff",
    groupvars = c("Peak", "data"), na.rm = T)

## plot the results
plot <- ggplot(df, aes(x = Peak, y = diff, fill = data)) +
    geom_bar(stat = "identity", color = "black", position = position_dodge()) +
    geom_errorbar(aes(ymin = diff, ymax = diff + sd),
        width = 0.2, position = position_dodge(0.9)) +
    labs(y = "Deviation from reference\nretention time [minutes]",
        x = "Cuticular hydrocarbon peak number") +
    theme_classic(base_size = 12) + scale_fill_manual(values = c("darkblue",
    "red")) + scale_y_continuous(breaks = seq(0, 0.35,
    0.05))
print(plot)
```



References

Dellicour S, Lecocq T. GCALIGNER 1.0: an alignment program to compute a multiple sample comparison data matrix from large eco-chemical dataset obtained by GC. Journal of separation science. 2013;36(19):3206-3209. doi:10.1002/jssc.201300388

Oksanen J, Blanchet FG, Friendly M, Kindt R, Legendre P, McGlinn D, et al.. vegan: Community Ecology Package; 2016. Available from: https://CRAN.R-project.org/package=vegan.

Wickham H. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York; 2009. Available from: http://ggplot2.org.

Wong JWY, Meunier J, Lucas C, Kölliker M. Paternal signature in kin recognition cues of a social insect: concealed in juveniles, revealed in adults. Proceedings of the Royal Society of London B: Biological Sciences.

2014;281(1793):20141236.