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

Meinolf Ottensmann, Martin A. Stoffel, 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 vegan
if (!"vegan" %in% rownames(installed.packages())) {
    install.packages("vegan")
}
```

- Installing the most recent version of `GCalignR` from GitHub.

```
# install GCalignR
devtools::install_github("mastoffel/GCalignR")
```

- Load packages and source custom functions.

```
library(GCalignR)
library(ggplot2)
library(plot3D)
library(vegan)
# small function to test parameters in
# align_chromatograms
source("R/optimal_params.R")
# calculates errors by matching aligned data to a
```

```r
# table of known substances
source("R/error_rate.R")
# custom function for simulations based on
# chromatograms
source("R/ChromaSimFunctions.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:  2017-02-01 18:04:11  Finished:  2017-02-01 18:41:11
#>
#> Call:
#>   GCalignR::align_chromatograms(data=peak_data, rt_col_name=time,
#>   max_linear_shift=0.05, blanks=(C2, C3), sep=\t, rt_cutoff_low=NULL,
#>   rt_cutoff_high=NULL, reference=NULL, max_diff_peak2mean=0.02,
#>   min_diff_peak2peak=0.08, delete_single_peak=FALSE)
#>
#> Summary of scored substances:
#>    total   blanks retained
#>      490      171      319
#>
#> In total 490 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. 319 substances are retained after all filtering steps.
#>
#> Sample overview:
#>   The following 84 samples were aligned to the reference 'P31':
#>   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.

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

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

```r
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 normalise_peaks that can be used to normalise peak abundances.

```r
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).

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

```r
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
```

```r
# 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")
```

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

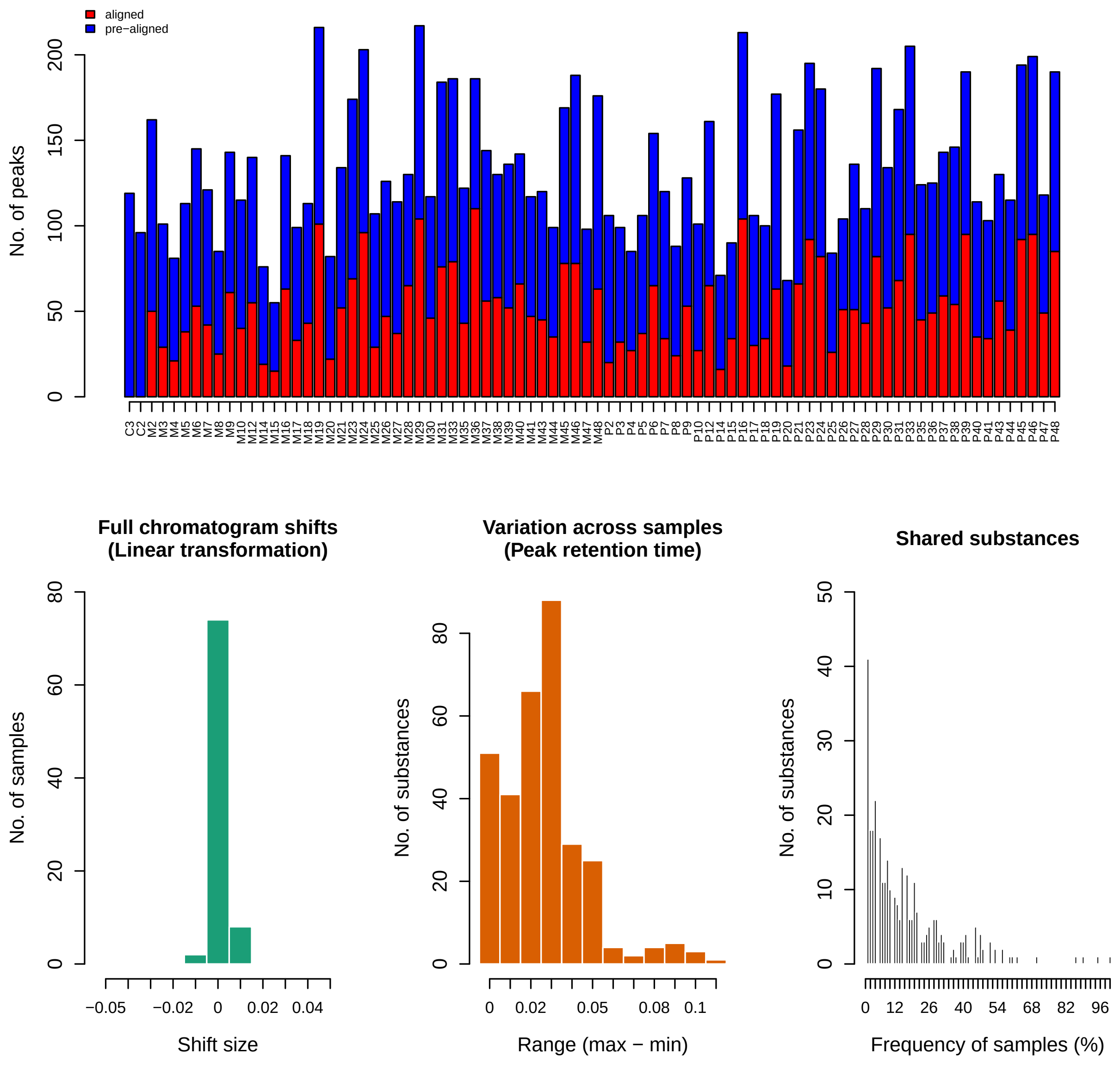

Figure 1: Diagnostic plots summarise aligned datasets

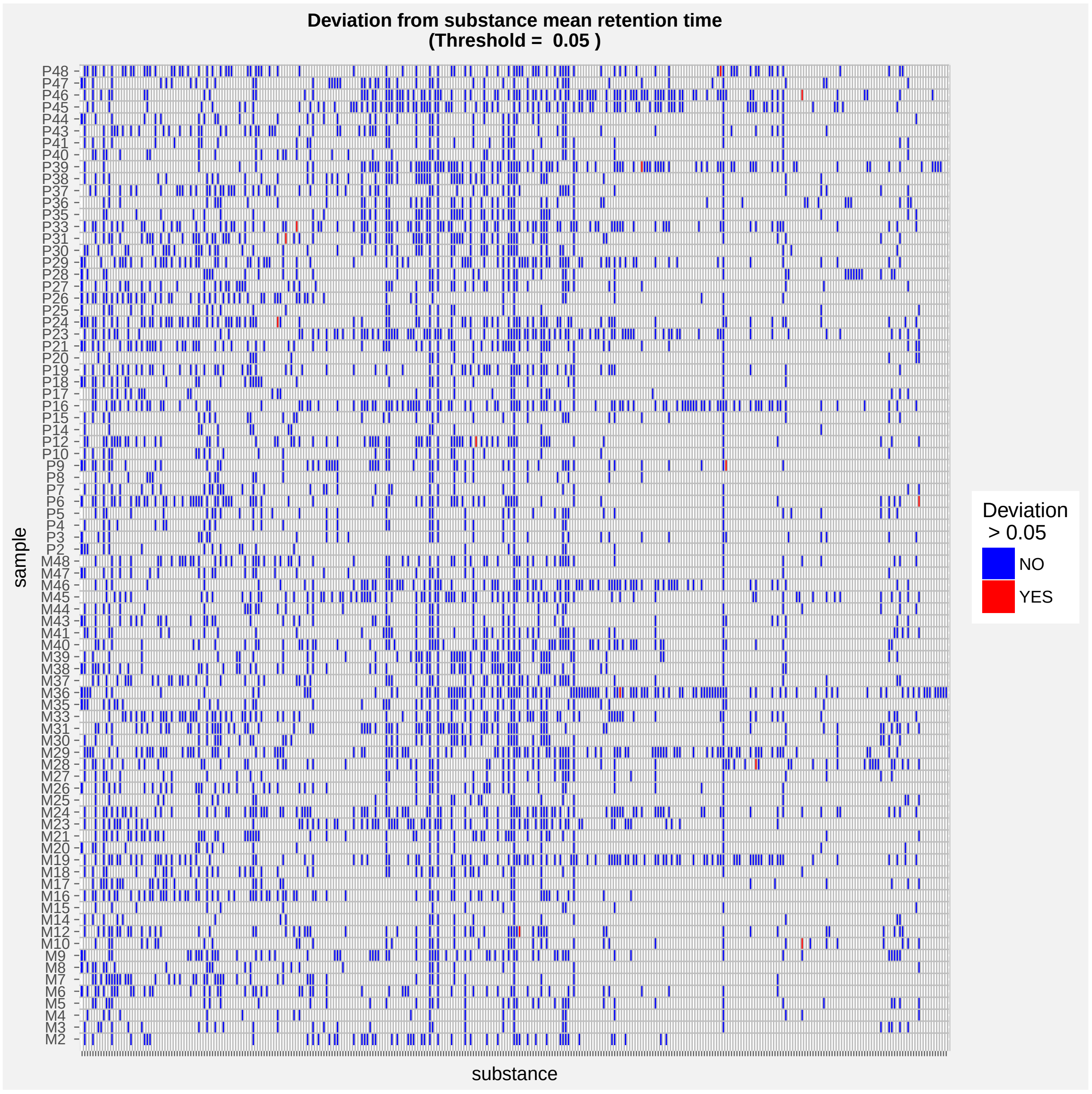

Figure 2: Heatmaps allow to inspect the distribution of substances across samples as well as the variability of their retention times.

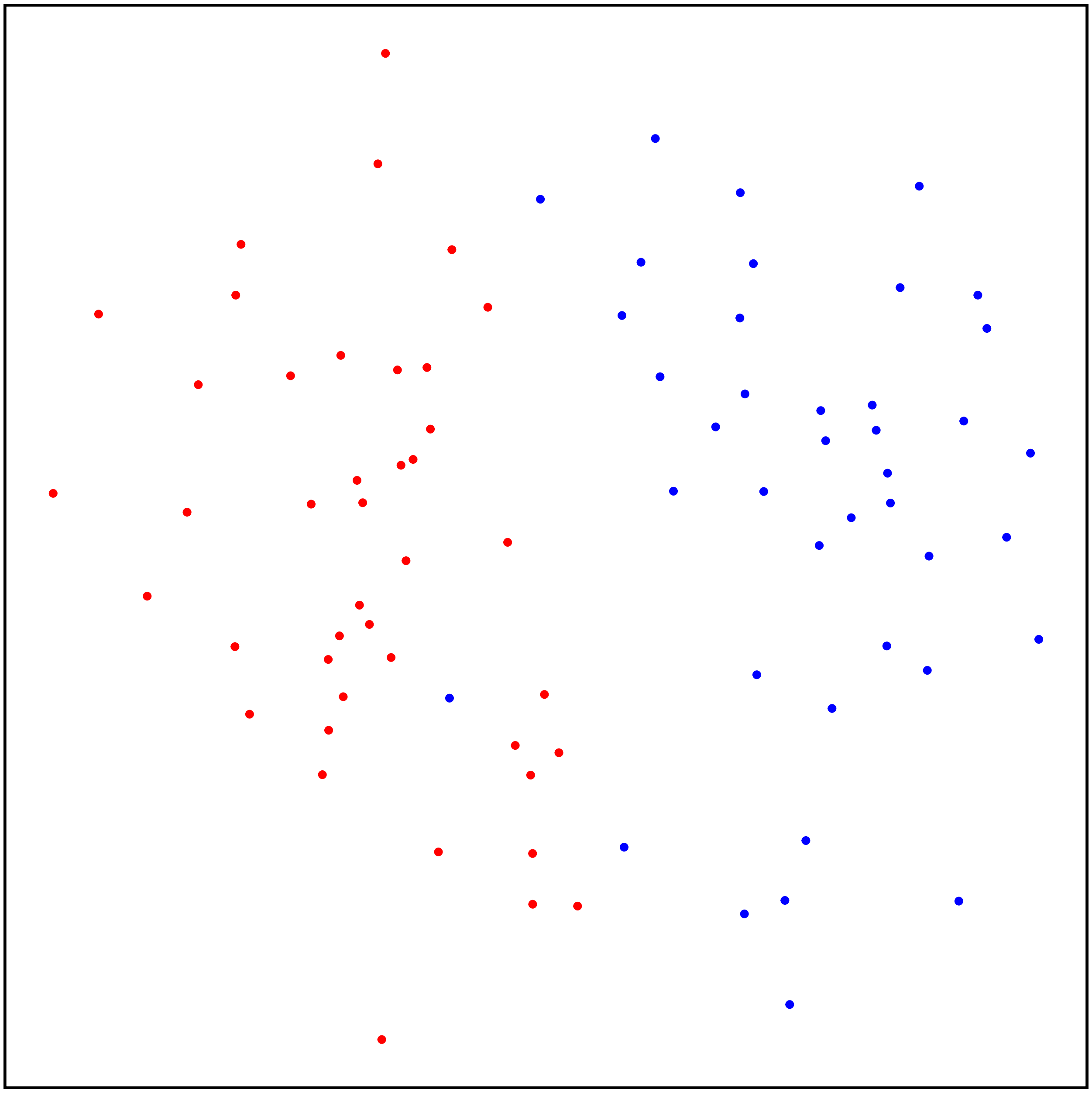

Figure 3: NMDS plot showing the cluserting by colony

```r
# B. flavifrons
results_bfla <- optimal_params(data = "data/bfla.txt",
    rt_col_name = "RT", max_diff_peak2mean = seq(from = 0.01,
        to = 0.05, by = 0.01), min_diff_peak2peak = seq(from = 0.01,
        to = 0.2, 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 = 0.05, by = 0.01), min_diff_peak2peak = seq(from = 0.01,
        to = 0.2, 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 = 0.05, by = 0.01), min_diff_peak2peak = seq(from = 0.01,
        to = 0.2, 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.

```r
# 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"))
```

- Plot results using package `plot3D`

```r
# 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)))
```

```r
# 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)))
```

```r
# 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)))
```

Influence of additional random noise on the detection of patterns in the fur seal dataset

- The input data for this simulation is the raw chemical dataset that is available within `aligned_peak_data.RData`. This file is supplied with the package.

```r
# list of samples
chromas <- aligned_peak_data[["input_list"]]

# vector of additional noise levels between 0 and 1
p <- rep(seq(from = 0, to = 1, by = 0.1), each = 10)
sim_data <- list()
names <- character()

for (i in 1:length(p)) {
    # add errors to the samples using the function
    # add_peak_error
    temp <- lapply(chromas, add_peak_error, p = p[i],
        rt_col_name = "time", conc_col_name = "area",
        distr = c(-0.02, -0.01, 0.01, 0.02))

    # extract peak list with manipulated retention
    # times
    temp <- lapply(temp, FUN = function(x) x[["chroma"]])

    # align the dataset
    aligned <- align_chromatograms(temp, rt_col_name = "time",
        max_linear_shift = 0.05, rt_cutoff_low = 8,
        delete_single_peak = T, blanks = c("C2", "C3"))

    sim_data <- append(sim_data, list(aligned))
    names <- c(names, paste0("no_", as.character(i),
        "_noise_", as.character(p[i])))
}
names(sim_data) <- names
```

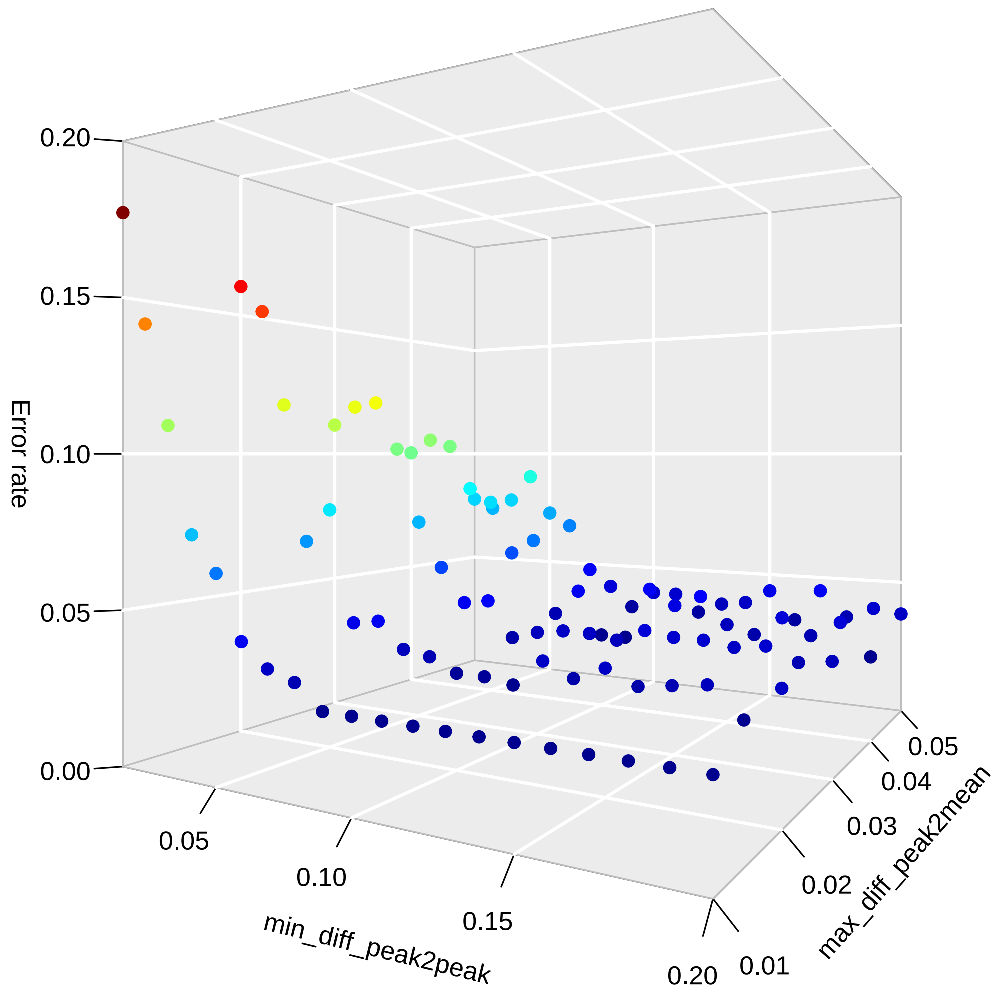

Figure 4: Parameter combinations B. bimaculatus dataset

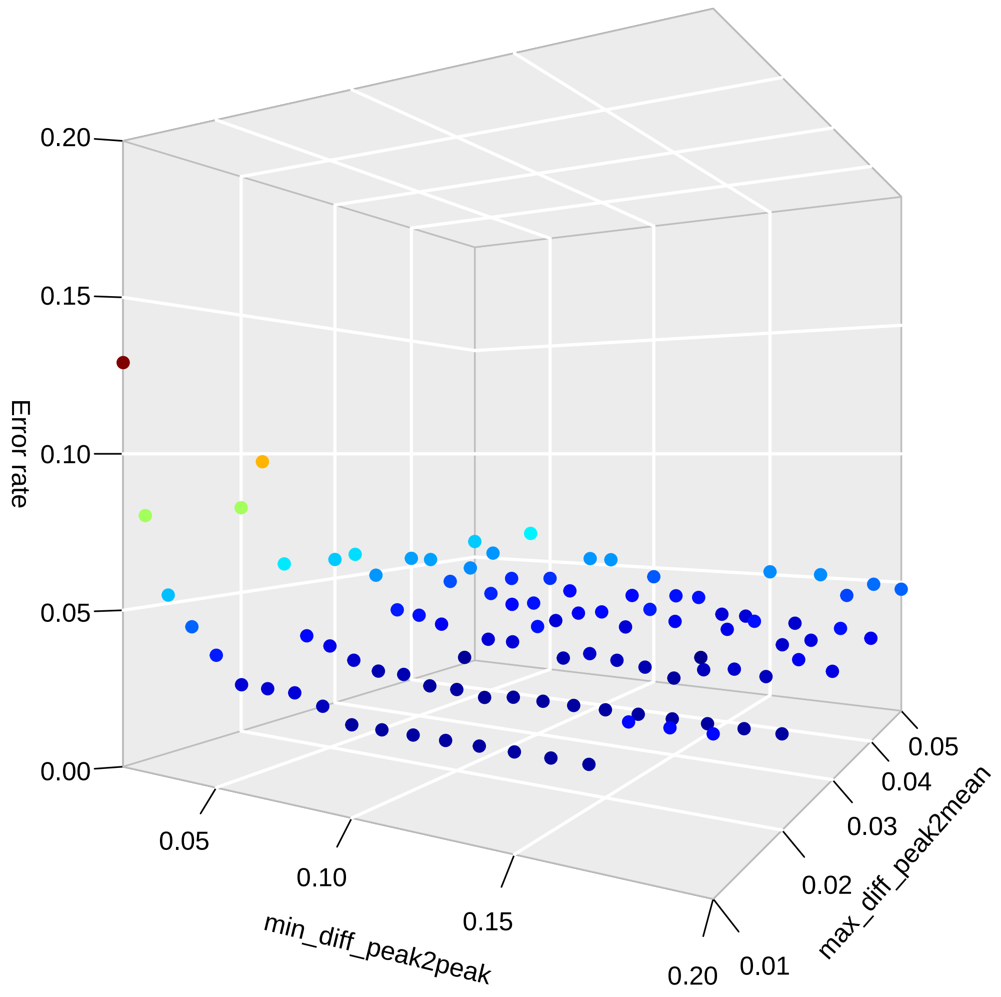

Figure 5: Parameter combinations B. ephippiatus dataset

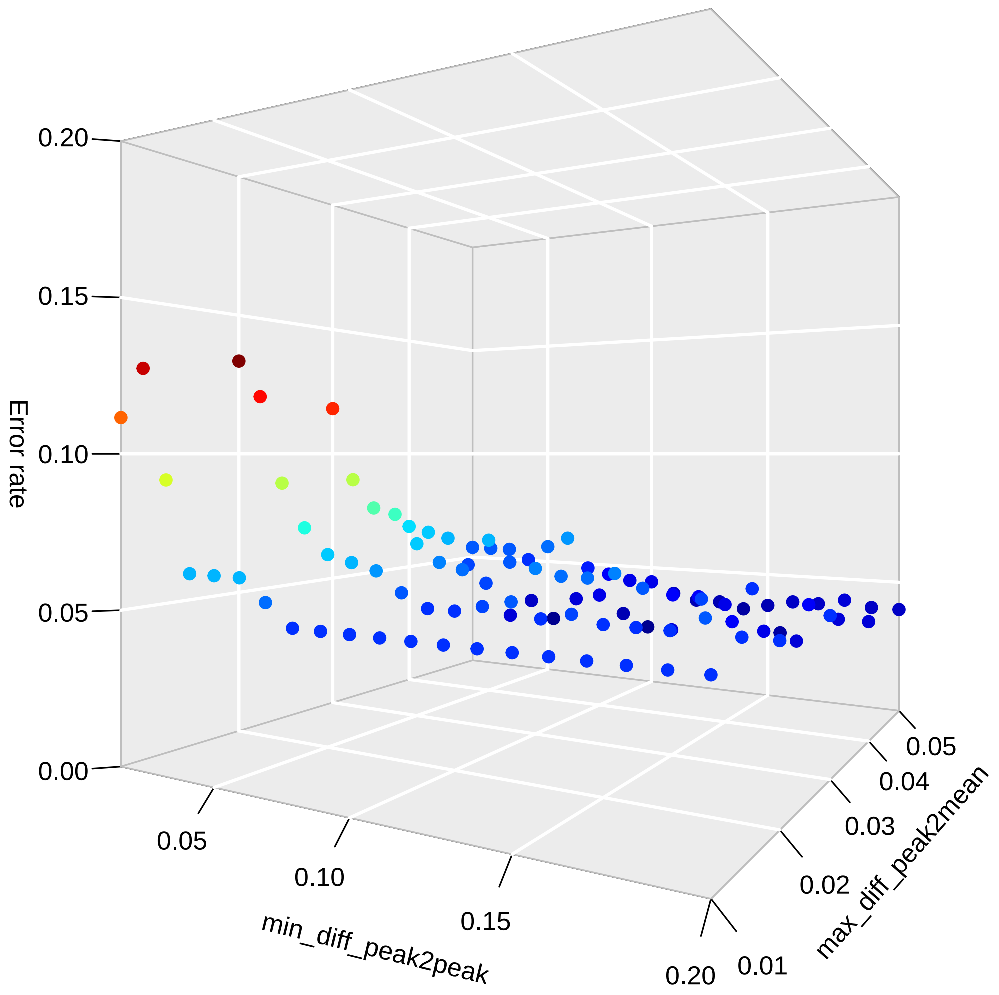

Figure 6: Parameter combinations B. flavifrons dataset

```r
seal_simulations <- list(SimAlign = sim_data, noise = p)
save(x = seal_simulations, file = paste0("data/", "seal_simulations",
    ".RData"))
```

- Load the results of the simulation

```r
# simulated data
load("data/seal_simulations.RData")
# extract data
aligned <- seal_simulations[["SimAlign"]]
# covariates are availabe as a data frame that is
# distributed with GCalignR
data("peak_factors")
covars <- peak_factors
```

- Perform the NMDS

```r
# get the scent, normalised and log+1 transformed
scent <- lapply(aligned, scent_extract, covars = covars)
save(x = scent, file = "data/scent.RData")
# NDMS
scent_mds <- lapply(scent, myMetaMDS, covars)
save(x = scent_mds, file = "data/scent_mds.RData")
```

- Calculate R2 values using `adonis` from package `vegan`

```r
scent_adonis_colony <- lapply(scent, adonis_colony,
    covars)  # calculates the adonis stats
save(x = scent_adonis_colony, file = "data/scent_adonis_colony.RData")
```

```r
load(file = "data/scent_adonis_colony.RData")
load(file = "data/seal_simulations.RData")
noise <- factor(seal_simulations[["noise"]])
r2 <- unlist(lapply(scent_adonis_colony, function(x) x[["aov.tab"]][["R2"]][1]))
p.val <- unlist(lapply(scent_adonis_colony, function(x) x[["aov.tab"]][["Pr(>F)"]][1]))
peaks <- unlist(lapply(seal_simulations[["SimAlign"]],
    function(x) x[["Logfile"]][["Aligned"]][["retained"]]))
df <- data.frame(noise, r2, p.val, peaks)
save(df, file = "data/df_seal_simulations.RData")
```

- Plot the results

```r
load("data/df_seal_simulations.RData")
p1 <- ggplot(df, aes(noise, peaks)) + geom_smooth(size = 1.5,
    se = T, colour = "blue", aes(group = 1)) + geom_boxplot(fill = "blue",
    alpha = 0.3, size = 0.1, weight = 1) + labs(y = "Number of substances") +
    scale_y_continuous(breaks = seq(200, 280, 10)) +
    theme_bw(base_family = "sans", base_size = 14) +
    theme(aspect.ratio = 0.5, axis.text.x = element_blank(),
        axis.ticks.x = element_blank(), axis.title.x = element_blank(),
        axis.title.y = element_text(margin = margin(0,
            13, 0, 0)), panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())

p2 <- ggplot(df, aes(noise, r2)) + geom_smooth(size = 1.5,
    se = T, colour = "red", aes(group = 1)) + geom_boxplot(fill = "red",
    alpha = 0.3, size = 0.1, weight = 1) + labs(y = "Adonis R²",
    x = "Additional noise level") + theme_bw(base_family = "sans",
    base_size = 14) + scale_y_continuous(breaks = seq(0,
    0.2, 0.025)) + theme(aspect.ratio = 0.5, axis.title.y = element_text(margin = margin(0,
    13, 0, 0)), panel.grid.major = element_blank(),
    panel.grid.minor = element_blank())
# Figure 7
grid::grid.draw(rbind(ggplotGrob(p1), ggplotGrob(p2),
    size = "first"))
```

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)).

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

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")
```

- Prepare the data for simulations

```
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
```

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

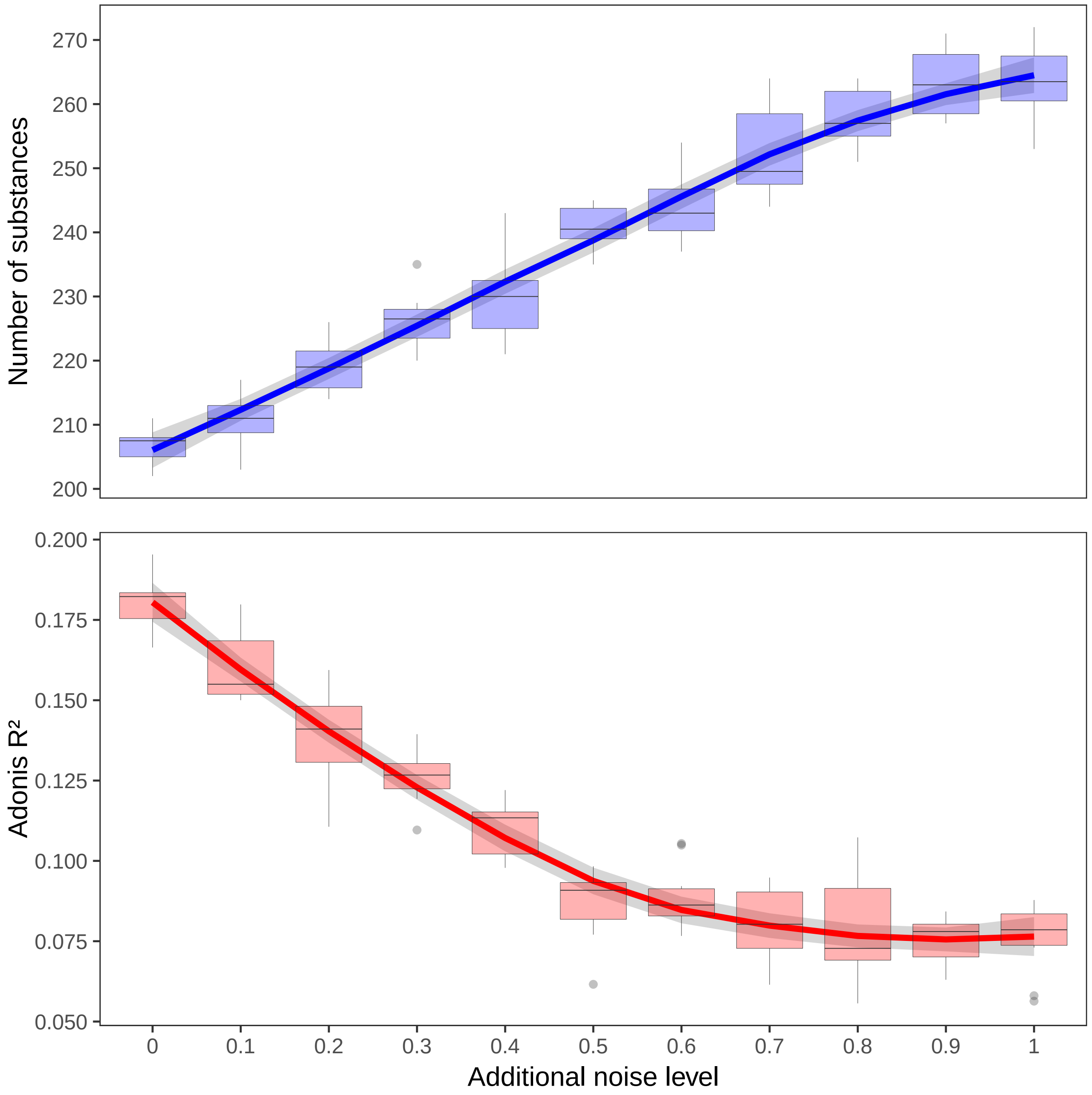

Figure 7: Effects of additional noise on alignments of the fur seal data

```r
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)
    # 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)
    # 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)
    # 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

```
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",
```

```r
    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 <- rbind(bbim, bfla, beph)
df[["id"]] <- rep(c("bbim", "bfla", "beph"), each = nrow(df)/3)
save(df, file = "data/df_bumblebee.RData")
```

- Plot the results

```r
load("data/df_bumblebee.RData")
rm(list = ls())
library(ggplot2)
load("data/df_bumblebee.RData")
df[["id"]] <- plyr::revalue(df[["id"]], c(bbim = "Bombus bimaculatus",
    beph = "B. ephippiatus", bfla = "B. flavifrons"))
df[["id"]] <- factor(df[["id"]], levels = c("Bombus bimaculatus",
    "B. ephippiatus", "B. flavifrons"))
# Figure 8
ggplot(df, 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'
```

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.

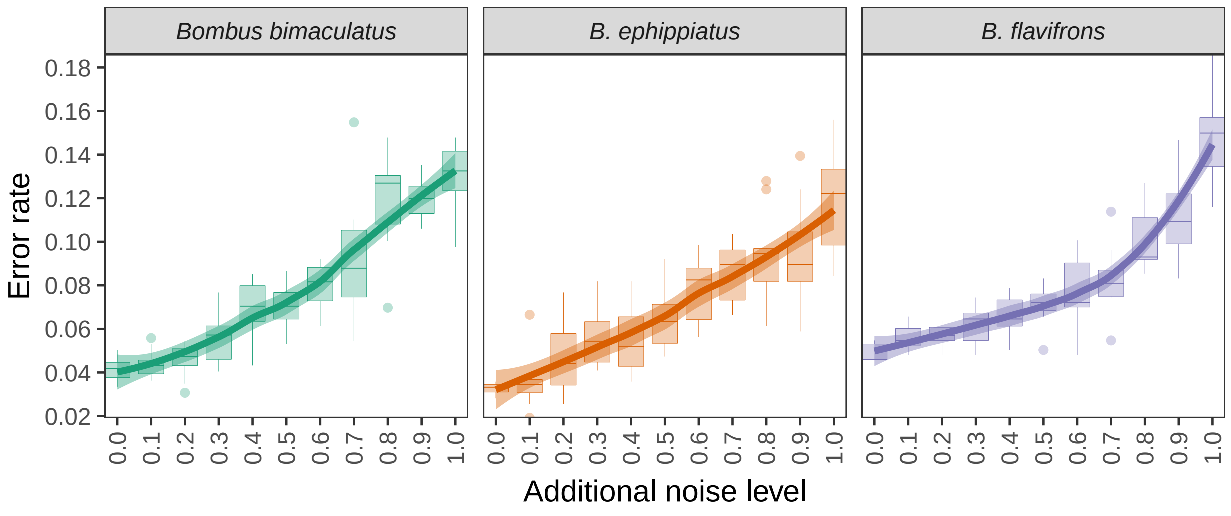

Figure 8: Effects of additional noise on error rates