Preturbation Fur Seal Data

Meinolf Ottensmann

Introduction

This document provides the code that was used to simulate the effect of preturbation in the dataset prior to aligning on the strength of the described pattern in chemical similarites among seals. For this simulation we assume that the aligned chemical dataset is perfect and all peaks are correctly aligned.

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 along with this document in the subdirectory `R`. In order to run the code you need to have a subdirectory called `data` that contains the raw datafiles.

- Install `ggplot2` and `GCalignR` if these packages are not available

```
## install ggplot2
if (!"ggplot2" %in% rownames(install.packages())) {
    install.packages("ggplot2")
}
if (!"GCalignR" %in% rownames(install.packages())) {
    install.packages("GCalignR")
}
if (!"gtable" %in% rownames(install.packages())) {
    install.packages("gtable")
}
```

```
library(GCalignR)
library(ggplot2)
library(gtable)
library(vegan)
#> Loading required package: permute
#> Loading required package: lattice
#> This is vegan 2.4-2
### source functions
source("R/ChromaSimFunctions.R")
source("R/NMDS-Functions.R")
source("R/ggplot_dual_axis.R")
source("R/ggplot_shared_x_axis.R")
```

Prepare the data

Data is extracted from previously aligned data in order to create an datset that is comprised of perfectly aligned retention times. Based on these data error rates are simulated by adding random noise to retention times.

Simualation

```
## input data
chromas <- aligned_peak_data[["input_list"]]

sim_zero <- align_chromatograms(data = chromas, rt_col_name = "time",
    max_linear_shift = 0.02, rt_cutoff_low = 8, blanks = c("C2",
        "C3"))

p <- rep(seq(from = 0, to = 1, by = 0.1), each = 3)
sim_data <- list()
names <- character()
for (i in 1:length(p)) {
    ## add errors
    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
    temp <- lapply(temp, FUN = function(x) x[["chroma"]])
    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"))
    ## We need the 'true' retention times for referencing
    ## purposes
    aligned <- original_rt(org = chromas, aligned = aligned,
        rt_col_name = "time")
    sim_data <- append(sim_data, list(aligned))
    names <- c(names, paste0("no_", as.character(i), "_noise_",
        as.character(p[i])))
}
names(sim_data) <- names
seal_simulations <- list(OptAlign = sim_zero, SimAlign = sim_data,
    noise = p)
save(x = seal_simulations, file = paste0("data/", "seal_simulations",
    ".RData"))
```

- Load simulated data

```
## simulated data
load("data/seal_simulations.RData")
## extract data
aligned <- seal_simulations[["SimAlign"]]
## covariates
data("peak_factors")
covars <- peak_factors
```

```
scent <- lapply(aligned, scent_extract, covars = covars)  # get the scent, normalised and log+1 transfo
save(x = scent, file = "data/scent.RData")
scent_mds <- lapply(scent, myMetaMDS, covars)  # MDS using vegan::metaMDS
save(x = scent_mds, file = "data/scent_mds.RData")
```

```
## load the results of nmds, and the scent data
load(file = "data/scent_mds.RData")
load(file = "data/scent.RData")
```

Do the permutational test

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

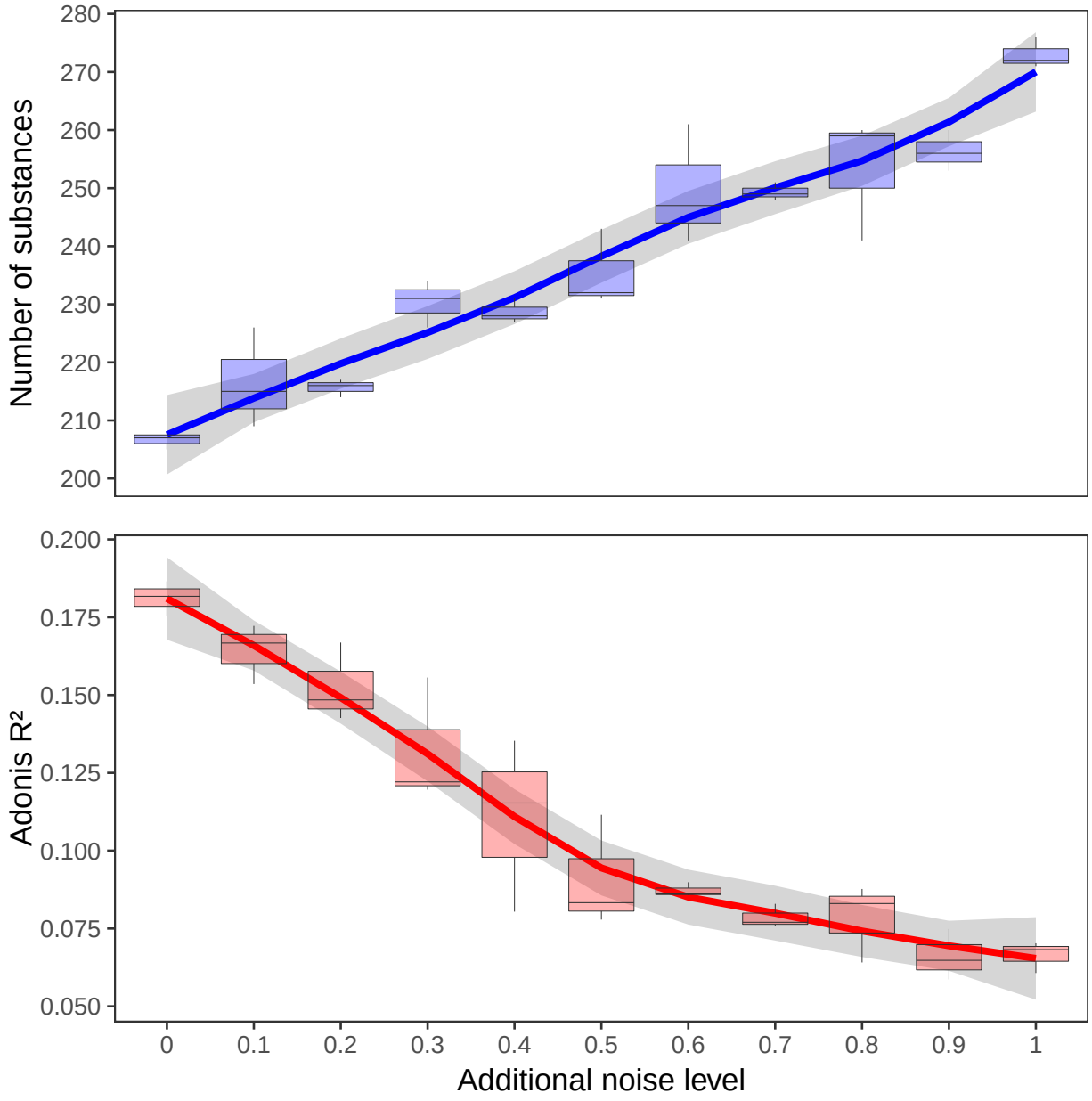
```
load(file = "data/scent_adonis_colony.RData")
load(file = "data/seal_simulations.RData")
```

```
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"]][["retain
df <- data.frame(noise, r2, p.val, peaks)
```

```
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())

grid::grid.draw(rbind(ggplotGrob(p1), ggplotGrob(p2), size = "first"))
#> `geom_smooth()` using method = 'loess'
#> `geom_smooth()` using method = 'loess'
```


Number of substances
280
270
260
250
240
230
220
210
200
Adonis R²
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Additional noise level