Analyse Cleaned Data

This document shows how the different functions of the IOFLOW package are working together and how to generate the figures and tables associated with the manuscript. The cleaned data were generated using process.R as explained in the “Clean_Raw_Data” vignette.

Libraries

```
library(IOFLOW)

# library(here)
library(dplyr) # for pipe
library(stringr) # for str_replace
library(purrr) # for map
# library(reshape2) # for the melt function

# # to plot
library(ggplot2)
library(grid)

# for nice tables
library(kableExtra)
```

Functions

```
devtools::load_all()
#> Loading IOFLOW
```

Dataset

There are three main datasets coming from the **IOFLOW package**:

- Tracking data for density test of 19 water tracers.
- Water and Food tracers data, refered to with “*WF*” suffix or prefix.
- Rigid body data, refered to with “*RB*” suffix or prefix.

The following script is based on functions we can call for a given trial or for all of them. For the analysis, there are:

- 7 sequences of Carps
- 6 sequences of Tilapia

Water tracers density test

Design of water tracers

The water tracer had to fulfil two apparently contradictory requirements: 1) being radio-opaque to be trackable on the x-ray videos
2) being neutrally buoyant to passively follow the water trajectory. We used 0.40 mm-diameter nickel rod on which we thread 1.4 mm closed-cell foam spheres (expanded polystyrene). We cut the nickel rod on both sides of each foam spheres to obtain the water tracers. The lightness of the foam counterbalanced the weight of the inserted nickel.

Test of water tracer density

We dropped 19 water tracers in the middle of a water column, filmed and extracted their trajectory, using ProAnalyst software (XCitex, Cambridge, MA, USA).
We calculated the mean velocity of the tracer until stabilization and calculated the water tracer density. We followed Stoke's law and used the following formula: $\rho_{particle} = (18\mu_{water}V_{mean})/gd^2 + \rho_{water}$ with:

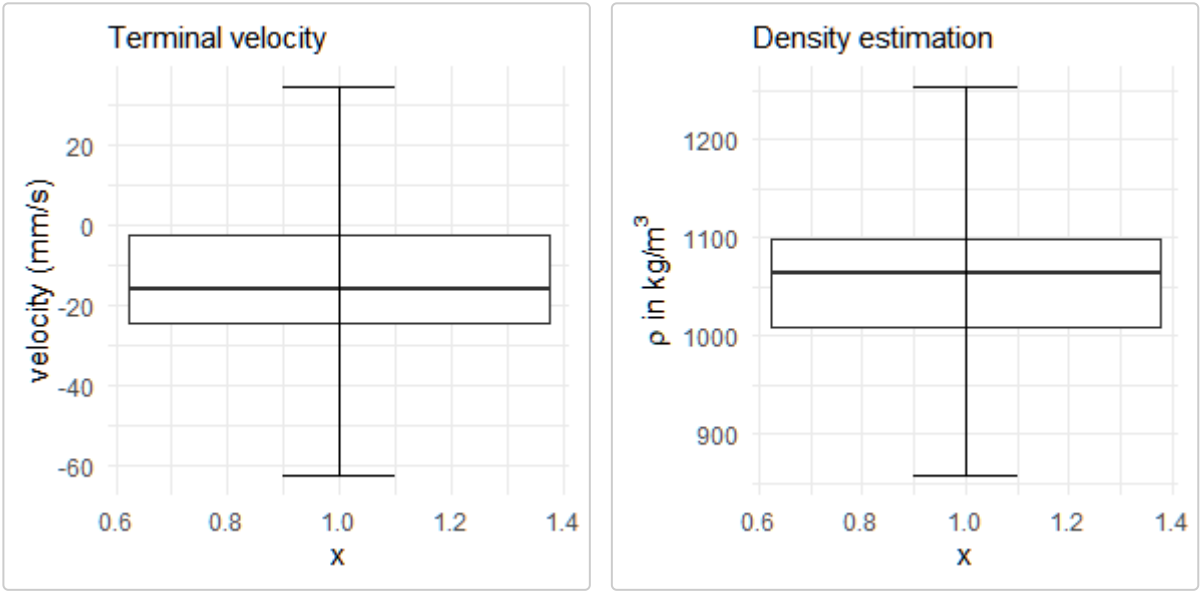
- g = 9.80665 m/s²
- d = 0.00067 m corresponding to the mean diameter of particles (calculated on the day of the experiment)
- ρ_{water} = 998.2 kg/m^3
- μ_{water} = 0.0010016 $kg/(m \cdot s)$ or 1.0016 mPa.s at 20°C, according to IAPWS 2008 reference

```
# DENSITY calculation

# Variable initialization
rho <- vector()
V_Mean <- vector()
V_Sd <- vector()

g <- 9.80665  #in m/s²
d <- 0.0006719836  # mean diameter of particles in m (calculated on the day of the experiment)
rho_water <- 998.2  # in kg/m^3
mu_water <-  0.0010016  # in kg/(m·s) or 1.0016 mPa.s à 20°C  data coming from
         http://www.viscopedia.com/viscosity-tables/substances/water/


# Variable initialization in Density_Data

for (i in 1:length(Density_Data)) {
  # For each water tracer tested
  Density_Data [[i]]$Dist <- NA
  Density_Data [[i]]$Velocity <- NA

  # Distance and Velocity calculation
  for (j in 2:(nrow(Density_Data[[i]]))) {
    Density_Data [[i]][j, 5] <-
      sqrt((Density_Data [[i]][j, 3] - Density_Data [[i]][j - 1, 3]) ^ 2 +
             (Density_Data [[i]][j, 4] - Density_Data [[i]][j - 1, 4]) ^ 2) # Dist
    Density_Data [[i]][j, 6] <- Density_Data[[i]][j, 5] / 0.02 # Velocity
  }

  # Sequence cropping
  A <-  which(Density_Data [[i]][, 1] == 0) + 1
  B <-
    which(Density_Data [[i]][, 6] == 0 &
            Density_Data[[i]][, 1] > 0) [1] - 2

  Density_Data [[i]] <- Density_Data [[i]][A:B,]
```

```
  # Motion orientation
  if (Density_Data[[i]][2, 4] < 0) {
    V_Mean[i] <- mean(Density_Data[[i]][, 6])
  }
  else{
    V_Mean[i] <- -mean(Density_Data[[i]][, 6])
  }

  V_Sd[i] <- sd(Density_Data[[i]][, 6])

  # rho calculation
  rho[i] <- (18 * mu_water *  V_Mean[i]) / (g * d ^ 2) + rho_water
}

Result_density <-
  as.data.frame(
    cbind(
      rho = rho,
      V_Mean = V_Mean,
      V_Sd = V_Sd,
      g = g,
      d = d,
      rho_water = rho_water,
      mu_water = mu_water
    )
  )
```

We obtain a mean value of ρ = 1055.74 $\pm$ 69.66kg/m^3

The distance to ρ_{water} = 57.54 $\pm$ 69.66 kg/m^3 The repartition of the data can be vizualized in the following plot, with the horizontal line corresponding to ρ_{water}.



This plot corresponds to the Figure 6c and 6d in the manuscript.

We considered that the density of water tracers was close enough to the density of water to use them in the experiment.

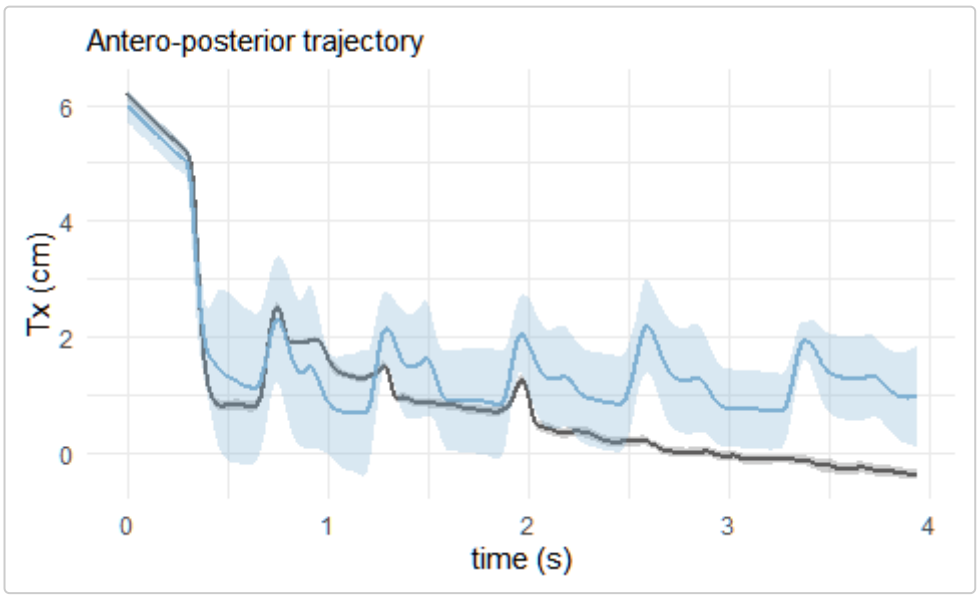
Hydrodynamic mechanism

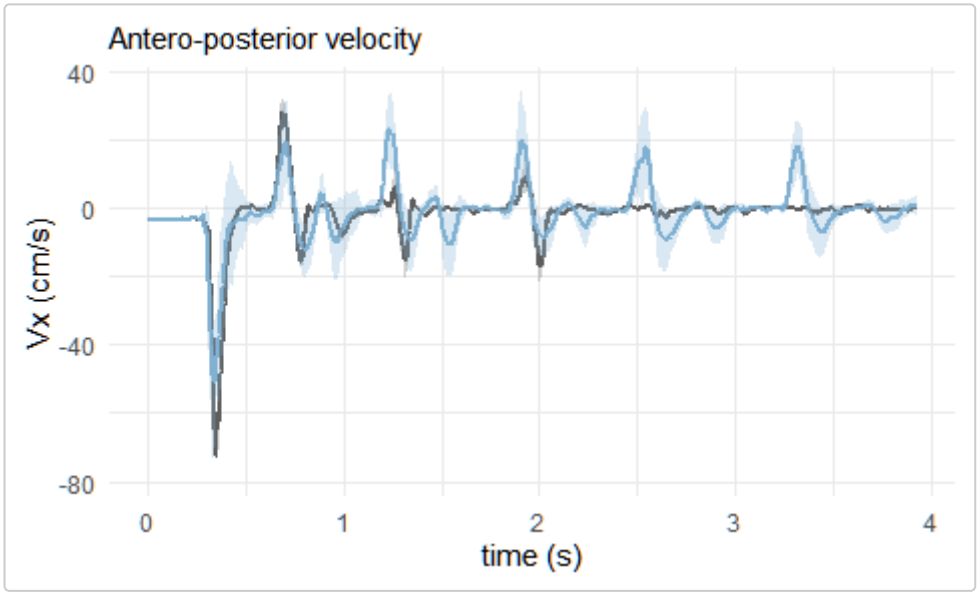
Waterflow trajectory and velocity during the entire sequence of suction feeding

We plot the trajectory and velocity of the food (one item) with the mean and standard deviation of the water tracers over time by calling the plot_mean function. These plots correspond to the Figure 2 in the manuscript.

Example of C2P1T04 (Carp 2, one food item, trial 4):

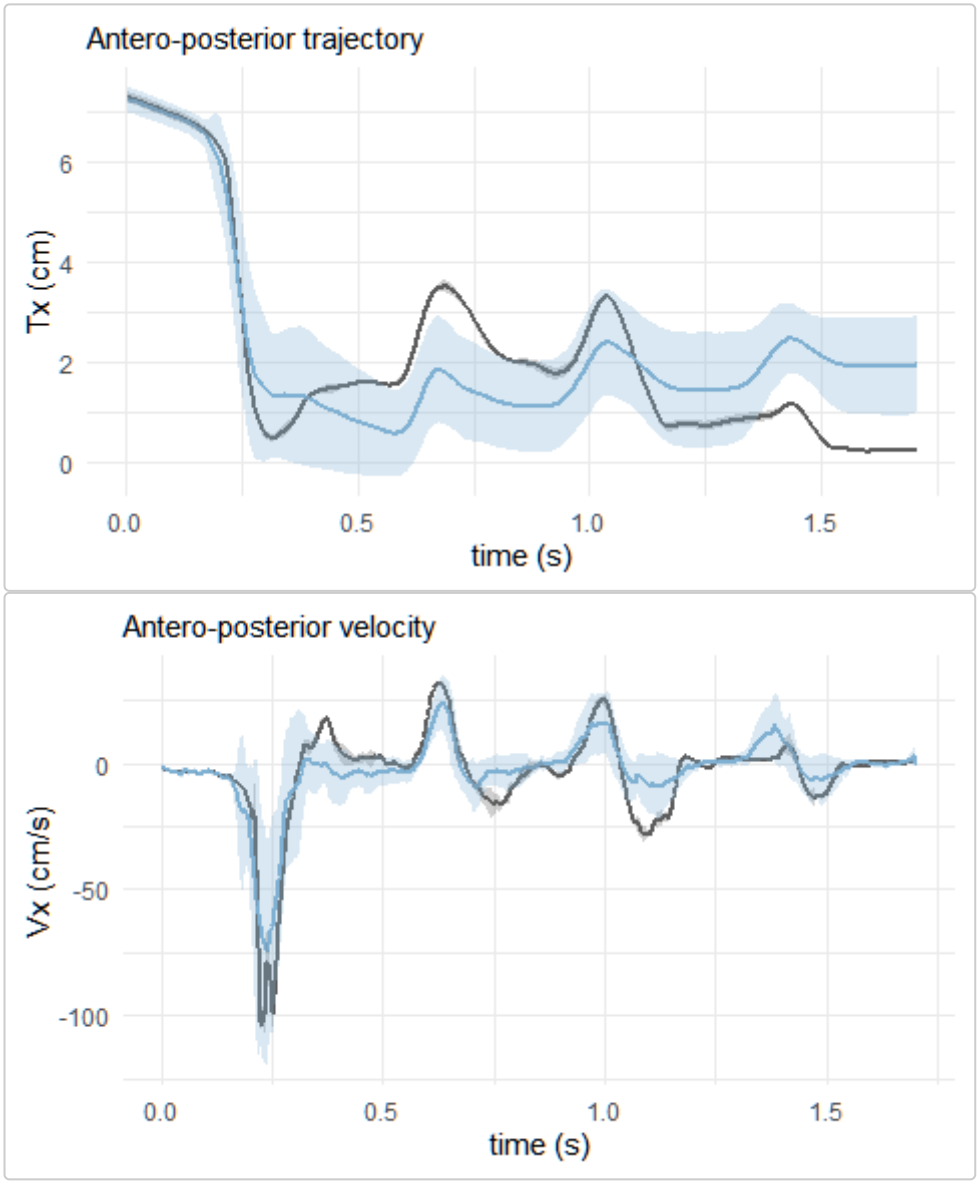
```
# FIGURE 2  Full Sequence: TRAJECTORY and VELOCITY

# C2P1T04: trial 4 of carp 2 (one prey)
trial <- "C2P1T04"
color <- c(palette_IOFLOW[13], # grey
           palette_IOFLOW[5]   # blue
)

# Tx plot
xlab <- "time (s)"
ylab <- "Tx (cm)"
d <- cbind(rbind(Mean_tx_W[[trial]],Mean_tx_F[[trial]]), time = Timing[[trial]]$time)
subtitle <- "Antero-posterior trajectory"

plot.Tx_C2P1T04 <- plot_mean(d, ylab=ylab, xlab=xlab, subtitle = subtitle, color = color)

# Vx plot
xlab <- "time (s)"
ylab <- "Vx (cm/s)"
d <- cbind(rbind(Mean_vx_W[[trial]],Mean_vx_F[[trial]]), time = Timing[[trial]]$time)
subtitle <- "Antero-posterior velocity"

plot.Vx_C2P1T04 <- plot_mean(d, ylab=ylab, xlab=xlab, color = color, subtitle = subtitle)


# plot the combination of the two plots
plot.Tx_C2P1T04
plot.Vx_C2P1T04
```





Example of T1P1T06 (Tilapia 1, one food item, trial 6)

```
# FIGURE 2  Full Sequence: TRAJECTORY and VELOCITY

# T1P1T06
trial <- "T1P1T06"

# Tx plot
xlab <- "time (s)"
ylab <- "Tx (cm)"

d <- cbind(rbind(Mean_tx_W[[trial]],Mean_tx_F[[trial]]), time = Timing[[trial]]$time)
subtitle <- "Antero-posterior trajectory"

plot.Tx_T1P1T06 <- plot_mean(d, ylab=ylab, xlab=xlab, subtitle = subtitle, color = color)


# Vx plot
xlab <- "time (s)"
ylab <- "Vx (cm/s)"

d <- cbind(rbind(Mean_vx_W[[trial]],Mean_vx_F[[trial]]), time = Timing[[trial]]$time)
subtitle <- "Antero-posterior velocity"

plot.Vx_T1P1T06 <- plot_mean(d, ylab=ylab, xlab=xlab, color = color, subtitle = subtitle)

plot.Tx_T1P1T06
plot.Vx_T1P1T06
```



Distance covered by Food vs Water tracers

We calculate the iterative distance covered by the water and food tracers (**WF**) at each frame, over a sequence of suction feeding. We store this in a list we named **Distance_WF**.

```
# Calculate the iterative distance of all tracers for each frame
Distance_WF <-
  map(.x = Trials, .f = ~ sqrt((WF_tx[[.]][] - lag(WF_tx[[.]][])) ^ 2 +
                               (WF_ty[[.]][] - lag(WF_ty[[.]][])) ^2 +
                               (WF_tz[[.]][] - lag(WF_tz[[.]][])) ^2)) %>%
  set_names(names(WF_tx))

# change the name of the variable ".tx" to ".dist"

Distance_WF <- mapply(FUN = data.frame,
                      map(Trials, ~ set_names(
                        x = Distance_WF[[.]][],
                        nm = str_replace(
                          string = names(Distance_WF[[.]][]),
                          pattern = "_data.tx",
                          replacement = ".dist"))),
```

```
                      SIMPLIFY = FALSE) %>%
  set_names(names(WF_tx))
```

Total distance for each tracer, cumulated on the entire sequence

We calculate the total distance covered by each tracer over the entire sequence, by summing the previously computed iterative distances. We name this variable **Sum_dist_WF**.

```
Sum_dist_WF <-
  map(.x = Trials,
      .f = ~ Distance_WF[[.]] %>%
        replace(is.na(.), 0) %>%
        summarise_all(sum)) %>%
  set_names(names(WF_tx))
```

Average of the Total distance cumulated on the entire sequence

We compute the mean and standard deviation of the cumulative distance **Sum_dist_WF** for each species and each type of tracer (food and water).
For the water tracers, the cumulative distance is averaged over all the water tracers of the trial. As there was one to three markers implanted in one food item, the standard deviation for the food tracers is low or corresponds to a NA when only one marker remained.

```
Carps_food     <- compute_species_item(species = "carp",
                       item = "food",
                       name = "Carps_food")
#> [1] Carps_food_mean
#> [1] Carps_food_sd
Tilapias_food  <- compute_species_item(species = "tilapia",
                       item = "food",
                       name = "Tilapias_food")
#> [1] Tilapias_food_mean
#> [1] Tilapias_food_sd

Carps_water    <- compute_species_item(species = "carp",
                       item = "water",
                       name = "Carps_water")
#> [1] Carps_water_mean
#> [1] Carps_water_sd
Tilapias_water <-compute_species_item(species = "tilapia",
                       item = "water",
                       name = "Tilapias_water")
#> [1] Tilapias_water_mean
#> [1] Tilapias_water_sd
```

For example, **Carps_food** is a list with the mean and standard deviation of the cumulative distance of each trial for the food item

```
glimpse(Carps_food[1:2])
#> List of 2
```

```
#>  $ C1P1T08:List of 2
#>   ..$ Carps_food_mean: num 21.7
#>   ..$ Carps_food_sd  : num 0.695
#>  $ C1P1T09:List of 2
#>   ..$ Carps_food_mean: num 22.5
#>   ..$ Carps_food_sd  : num 0.205
```

We combine the food and water average and standard deviation, using the **tidy_data2** function.

```
Carp_names <- names(Sum_dist_WF)[starts_with(match = "C", vars = names(Sum_dist_WF))]
Carps_distance <- tidy_data2(Carp_names, Carps_food, Carps_water)

Tilapia_names <- names(Sum_dist_WF)[starts_with(match = "T", vars = names(Sum_dist_WF))]
Tilapias_distance <- tidy_data2(Tilapia_names, Tilapias_food, Tilapias_water)
```

For example, **Carps_distance** is a list with the mean and standard deviation of the cumulative distance of each trial for both the food and water tracers.

```
glimpse(Carps_distance[1])
#> List of 1
#>  $ C1P1T08:List of 2
#>   ..$ Carps_food :List of 2
#>   .. ..$ Carps_food_mean: num 21.7
#>   .. ..$ Carps_food_sd  : num 0.695
#>   ..$ Carps_water:List of 2
#>   .. ..$ Carps_water_mean: num 28.7
#>   .. ..$ Carps_water_sd  : num 6.74
```

Table combining the distance cumulated on the entire sequence for both types of tracers and each species

We build a table with all the important information about the cumulative distance of food and water tracers.

```
Carps_mean <- data.frame(
  mean = c(map_dbl(.x = Carp_names, .f = ~Carps_food[[.]]$Carps_food_mean ),
           map_dbl(.x = Carp_names, .f = ~Carps_water[[.]]$Carps_water_mean)),
  sd = c(rep(NA, length(Carp_names)),
         map_dbl(.x = Carp_names, .f = ~Carps_water[[.]]$Carps_water_sd)),
  species = 'Carps',
  trials = rep(Carp_names, 2),
  item = c(rep("food", length(Carp_names)), rep("water", length(Carp_names))))

Tilapias_mean <- data.frame(
  mean = c(map_dbl(.x = Tilapia_names, .f = ~Tilapias_food[[.]]$Tilapias_food_mean ),
           map_dbl(.x = Tilapia_names, .f = ~Tilapias_water[[.]]$Tilapias_water_mean)),
  sd = c(rep(NA, length(Tilapia_names)),
         map_dbl(.x = Tilapia_names, .f = ~Tilapias_water[[.]]$Tilapias_water_sd)),
  species = 'Tilapias',
  trials = rep(Tilapia_names, 2),
  item = c(rep("food", length(Tilapia_names)),rep("water", length(Tilapia_names))))
```

Here is what **Carps_mean** looks like

mean	sd	species	trials	item
21.71281	NA	Carps	C1P1T08	food
22.45945	NA	Carps	C1P1T09	food
22.50143	NA	Carps	C1P2T10	food
21.58910	NA	Carps	C1P2T13	food
20.25855	NA	Carps	C2P1T03	food
16.96296	NA	Carps	C2P1T04	food
11.54599	NA	Carps	C2P2T17	food
28.71888	6.739617	Carps	C1P1T08	water
30.49912	7.774822	Carps	C1P1T09	water
32.74554	9.174106	Carps	C1P2T10	water
20.17168	8.550082	Carps	C1P2T13	water
23.67716	4.266879	Carps	C2P1T03	water
29.01463	4.093528	Carps	C2P1T04	water
17.73619	4.072921	Carps	C2P2T17	water

Here is what **Tilapias_mean** looks like

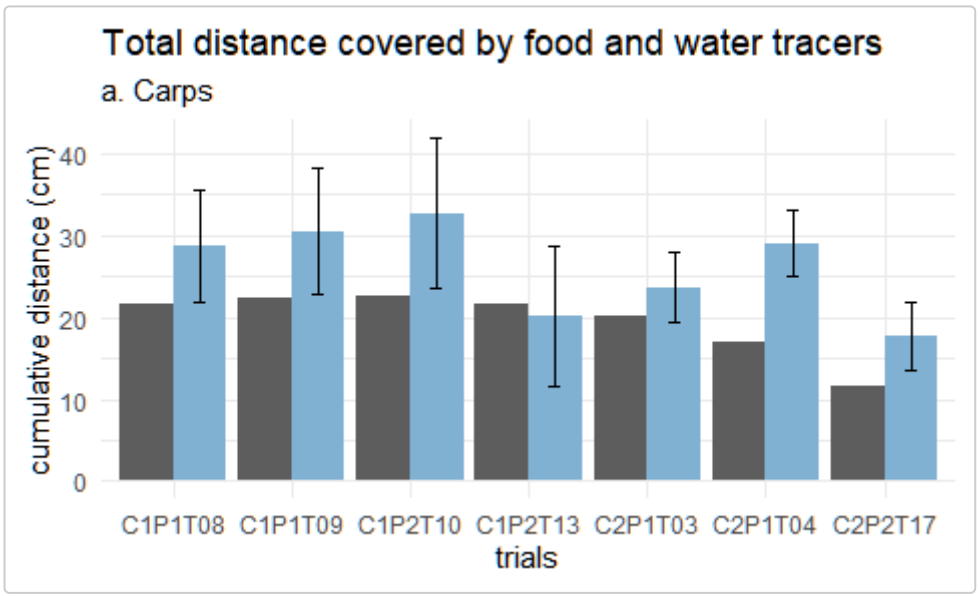
mean	sd	species	trials	item
21.36160	NA	Tilapias	T1P1T06	food
14.59018	NA	Tilapias	T1P1T09	food
17.86253	NA	Tilapias	T1P1T17	food
11.63494	NA	Tilapias	T2P1T10	food
14.93221	NA	Tilapias	T2P1T11	food
12.80737	NA	Tilapias	T2P1T14	food
19.90987	5.499691	Tilapias	T1P1T06	water
17.95164	1.609222	Tilapias	T1P1T09	water
28.10587	3.896228	Tilapias	T1P1T17	water
13.51054	1.174713	Tilapias	T2P1T10	water
17.04378	1.395753	Tilapias	T2P1T11	water
14.30141	1.850642	Tilapias	T2P1T14	water

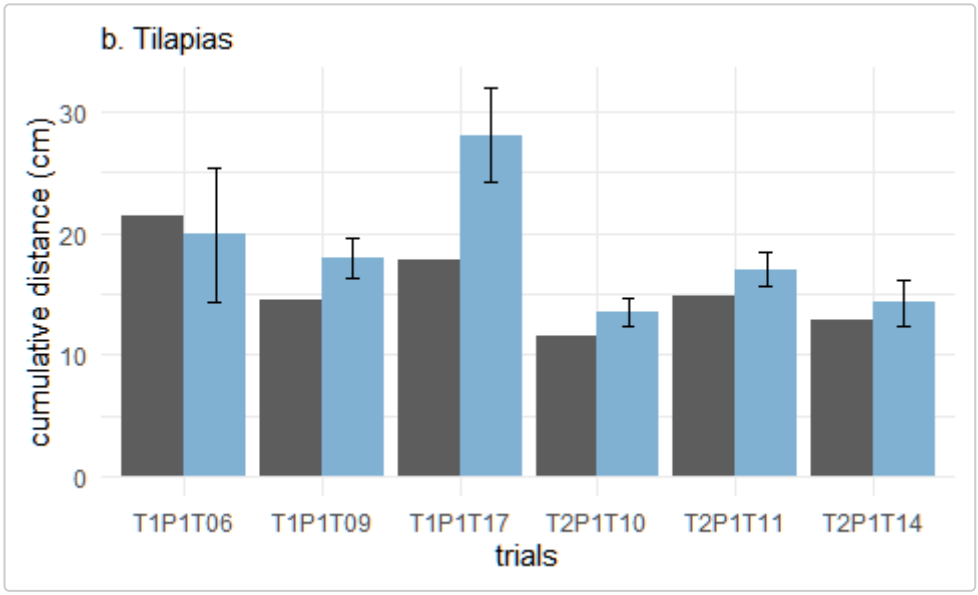
These data correspond to the Table 1 in the manuscript.

Histogram of the total distance covered by food and water tracers

We use the previous table to plot an histogramm with the relevant variables. This histogram corresponds to a visualisation of Table 1.

```
carp_bar <-
  ggplot(data = Carps_mean, aes(x = trials, y = mean, fill = item)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  geom_errorbar(aes(ymin = mean - sd, ymax = mean + sd),
                width = .2,
                position = position_dodge(0.9)) +
  theme_minimal() +
  theme(legend.position = "none") +
  ylab("cumulative distance (cm)") +
  scale_color_manual(values = color) +
  scale_fill_manual(values =  color) +
  labs(title = "Total distance covered by food and water tracers", subtitle =
         "a. Carps")

tilapia_bar <-
  ggplot(data = Tilapias_mean, aes(x = trials, y = mean, fill = item)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  geom_errorbar(aes(ymin = mean - sd, ymax = mean + sd),
                width = .2,
                position = position_dodge(0.9)) +
  theme_minimal() +
  theme(legend.position = "none") +
  ylab("cumulative distance (cm)") +
  scale_color_manual(values = color) +
  scale_fill_manual(values =  color) +
  labs(subtitle = "b. Tilapias")


carp_bar
tilapia_bar
```





The food tracers don't have any standard deviation bars because all the markers were on the same food item.

Phases during a suction feeding sequence

To automatically detect the different phases during a trial, we use a combination of information about the water tracers velocity and trajectory. We focus on the water tracers to determine the different phases.

```
# combine Mean_vx with Mean_tx in Mean_v_t_W
Frame <-  map(.x = Trials,
              .f = ~ seq(from = 1, to = nrow(Mean_vx_W[[.]]))) %>%
  set_names(names(Maya_WF))

vx_names <- paste0(names(Mean_vx_W[[1]][]), "_vx")
Mean_vx_W_2 <- map(.x = Trials,
                   .f = ~ tibble(Mean_vx_W[[.]][]) %>% set_names(vx_names)) %>%
  set_names(names(Mean_vx_W))

tx_names <- paste0(names(Mean_tx_W[[1]][]), "_tx")
Mean_tx_W_2 <- map(.x = Trials,
                   .f = ~ tibble(Mean_tx_W[[.]][]) %>% set_names(tx_names)) %>%
  set_names(names(Mean_tx_W))

Mean_v_t_W <-  mapply(
  FUN = data.frame,
  Mean_vx_W_2,
  Mean_tx_W_2,
  frame = Frame,
  SIMPLIFY = FALSE)

# compute the differences at each row of Mean_v_t_W to obtain a trend (+ or -) and add it to
        Mean_v_t_W

Phase <- map(.x = Trials,
             .f = ~ Mean_v_t_W[[.]] %>%
               mutate(diff_average_vx = average_vx - lag(average_vx)) %>%
               mutate(diff_average_tx = average_tx - lag(average_tx))) %>%
  set_names(names(Mean_v_t_W))
```

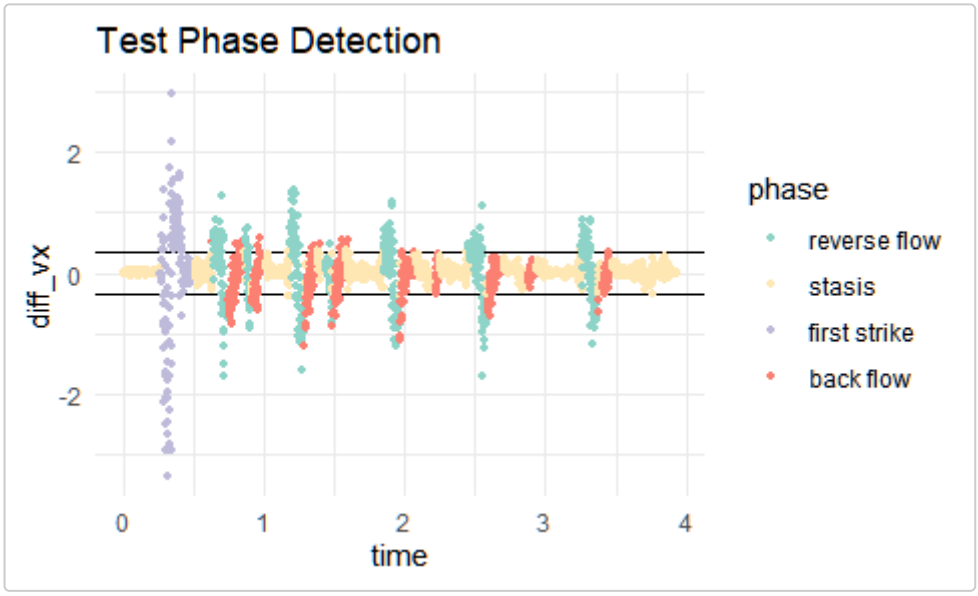
```
# compute a label (stasis, reverse flow, transport) associated with a given combination of value
         for average_vx, and diff_average_vx

Phase <-  map(.x = Trials,
              .f = ~ mutate(.data = Phase[[.]],
                            phase =
                              case_when(
                                frame < Sequence$end_FS[[.]] &
                                  frame > Sequence$beg_FS[[.]] ~ 1, # first strike
                                average_vx < 2 &
                                  average_vx > -4.5 &
                                  diff_average_vx < 0.4 &
                                  diff_average_vx > -0.4 ~ 0,       #stasis
                                average_vx > 0.05 ~ -1,             # reverse flow
                                average_vx < 0.05 ~ 2)              # back flow
                            )) %>%
  set_names(names(Mean_vx_W_2))


# Extract the phase list, corresponding to the previous labels for each trial over time.
phase <-  map(.x = Trials,
              .f = ~ get_sublist(lst = Phase[[.]], group_name = "phase")) %>%
  set_names(names(Phase))
```

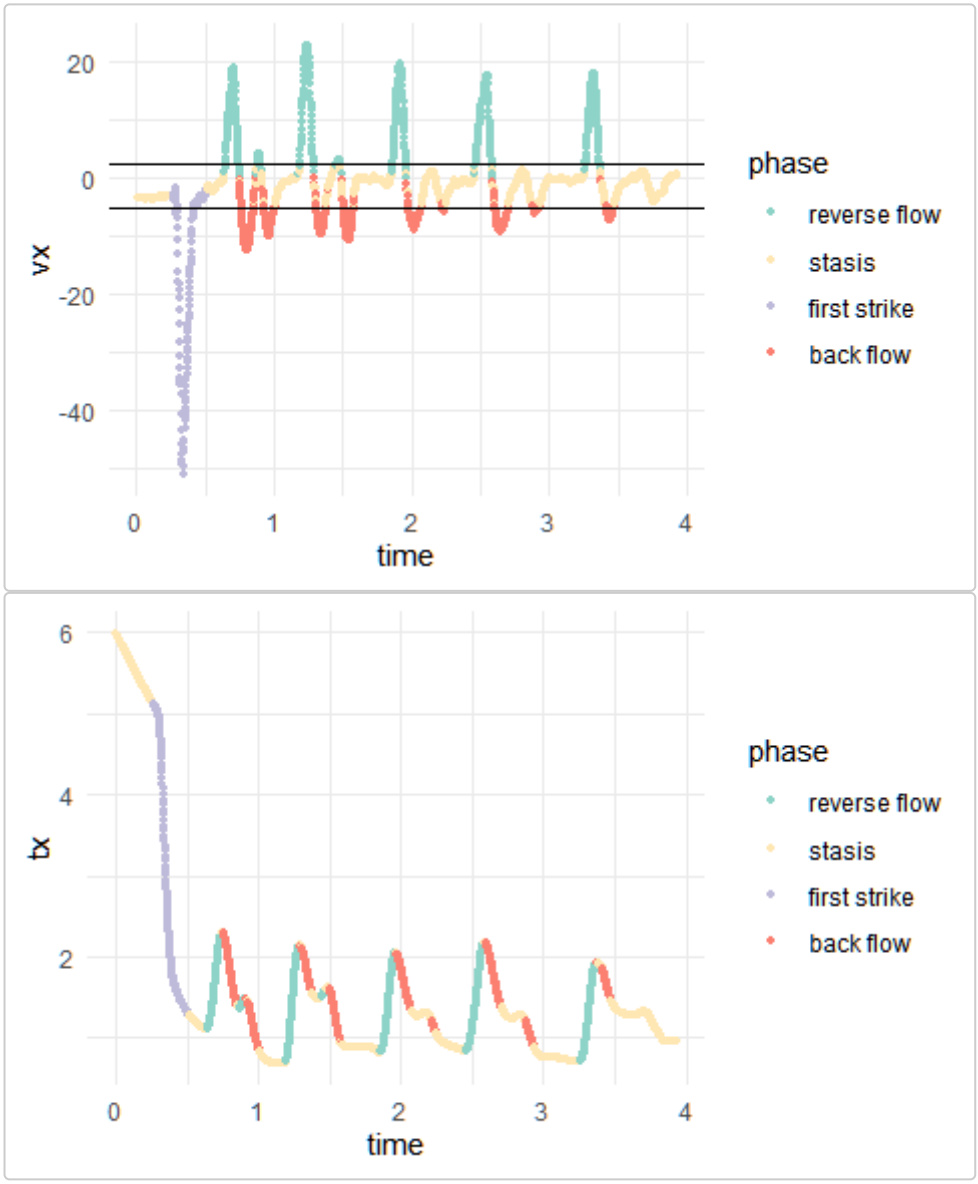
Test for the phase extraction

We test the phase detection on the different trials. Here we chose “T1P1T09”.

```
# Tested trial
trial <- "C2P1T04"

# Plot of the velocity trend (+ or -) over the trial

plot.Velocity_trend <- ggplot(data = Phase[[trial]],
                              aes(x = Timing[[trial]]$time,
                                  y =  diff_average_vx)) +
  geom_hline(yintercept = 0.35) +
  geom_hline(yintercept = -0.35) +
  geom_point(aes(colour = as.factor(phase)), size = 1) +
  ylab("diff_vx") +
  xlab("time") +
  scale_color_manual(
    values = palette_IOFLOW,
    labels = c("reverse flow", "stasis", "first strike", "back flow"),
    name = "phase"
  ) +
  scale_fill_manual(values =  palette_IOFLOW) +
  labs(title = "Test Phase Detection") +
  theme_minimal()


plot.Velocity_mean <- ggplot(data = Phase[[trial]],
                             aes(x = Timing[[trial]]$time,
                                 y = average_vx)) +
  geom_point(aes(colour = as.factor(phase)), size = 1) +
```

```
  ylab("vx") +
  xlab("time") +
  geom_hline(yintercept = -5) +
  geom_hline(yintercept = 2.5) +
  scale_color_manual(
    values = palette_IOFLOW,
    labels = c("reverse flow", "stasis", "first strike", "back flow"),
    name = "phase"
  ) +
  scale_fill_manual(values =  palette_IOFLOW) +
  theme_minimal()


plot.TrajectoryX_mean <- ggplot(data = Phase[[trial]],
                                aes(x = Timing[[trial]]$time,
                                    y = average_tx)) +
  geom_point(aes(colour = as.factor(phase)), size = 1) +
  ylab("tx") +
  xlab("time") +
  scale_color_manual(
    values = palette_IOFLOW,
    labels = c("reverse flow", "stasis", "first strike", "back flow"),
    name = "phase"
  ) +
  scale_fill_manual(values =  palette_IOFLOW) +
  theme_minimal()


plot.Velocity_trend
plot.Velocity_mean
plot.TrajectoryX_mean
```





Periodicity of the phases

To describe the suction feeding waterflow we will compute the periodicity of the different phases. After the first strike, we export the peaks of tx over time

```
# Detection of the transition between each reverse flow and back flow

peaks <-  map(.x = Trials,
              .f = ~ find_peaks(x = Phase[[.]]$average_tx, m = 150)) %>%
  set_names(names(Phase))
```

We want to extract the duration between each transition reverse/back flow:

```
diff_peaks <- map(.x = Trials, .f = ~diff(peaks[[.]])) %>% set_names(names(Phase))

# Mean and sd calculation among the different transitions
mean_peaks <- map(.x = Trials, .f = ~mean(diff_peaks[[.]], na.rm = TRUE)) %>%
         set_names(names(Phase))
sd_peaks <-map(.x = Trials, .f = ~sd(diff_peaks[[.]], na.rm = TRUE)) %>% set_names(names(Phase))

# Mean and sd calculation among the different trials of the same species
mean_peaks_Carps <- c(
```

```
  mean = mean(unlist(mean_peaks)[Carp_names], na.rm = TRUE)/750,
    sd = sd(unlist(mean_peaks)[Carp_names], na.rm = TRUE)/750)

mean_peaks_Tilapias <- c(
  mean = mean(unlist(mean_peaks)[Tilapia_names], na.rm = TRUE)/750,
    sd = sd(unlist(mean_peaks)[Tilapia_names], na.rm = TRUE)/750)
```

For the carps, the transition between a reverse and a backflow happens every 0.58 $\pm$ 0.12 s
For the tilapias, the transition between a reverse and a backflow happens every 0.4 $\pm$ 0.05 s

Velocity amplitudes

To describe the waterflow over time, we need to compute several values, including maximum velocity amplitudes for the different phases in both carps and tilapias.

First strike

We use the minimum of the **average_VX** data to obtain the maximum amplitude the water tracers motion during the first strike as the X axis is pointing toward the back of the buccal cavity.

```
# List of the Water tracers variables
Data_W <- mapply(FUN = data.frame, Timing, phase,
                 map(Trials, ~set_names(x =Mean_tx_W[[.]][2],nm = rep("average_TX", length(.)))),
                 map(Trials, ~set_names(x =Mean_ty_W[[.]][2],nm = rep("average_TY", length(.)))),
                 map(Trials, ~set_names(x =Mean_tz_W[[.]][2],nm = rep("average_TZ", length(.)))),
                 map(Trials, ~set_names(x =Mean_tx_W[[.]][3],nm = rep("sd_TX", length(.)))),
                 map(Trials, ~set_names(x =Mean_ty_W[[.]][3],nm = rep("sd_TY", length(.)))),
                 map(Trials, ~set_names(x =Mean_tz_W[[.]][3],nm = rep("sd_TZ", length(.)))),

                 map(Trials, ~set_names(x =Mean_vx_W[[.]][2],nm = rep("average_VX", length(.)))),
                 map(Trials, ~set_names(x =Mean_vy_W[[.]][2],nm = rep("average_VY", length(.)))),
                 map(Trials, ~set_names(x =Mean_vz_W[[.]][2],nm = rep("average_VZ", length(.)))),
                 map(Trials, ~set_names(x =Mean_vx_W[[.]][3],nm = rep("sd_VX", length(.)))),
                 map(Trials, ~set_names(x =Mean_vy_W[[.]][3],nm = rep("sd_VY", length(.)))),
                 map(Trials, ~set_names(x =Mean_vz_W[[.]][3],nm = rep("sd_VZ", length(.)))),
                 SIMPLIFY=FALSE) %>%
  set_names(names(Timing))



# Compute the min of the average_X
Vx_FS <-  map_dbl(.x = Trials,
                  .f = ~ abs(min(Data_W[[.]]$average_VX, na.rm = TRUE)))

FirstStrike_magnitude <-  data.frame(Trial = names(Maya_WF),
                                     FirstStrike_magnitude = Vx_FS)

FirstStrike_magnitude %>%
  kable() %>%
  kable_styling()
```

Trial	FirstStrike_magnitude
C1P1T08	39.12806

Trial	FirstStrike_magnitude
C1P1T09	108.68672
C1P2T10	126.53022
C1P2T13	206.29392
C2P1T03	56.11972
C2P1T04	51.11313
C2P2T17	56.96054
T1P1T06	74.48523
T1P1T09	82.08532
T1P1T17	86.90078
T2P1T10	99.98013
T2P1T11	104.21908
T2P1T14	139.11654

```
# Compute the mean in a given species
## carps
Vx_FS_Carps <- abs(Vx_FS[1:7])

## tilapias
Vx_FS_Tilapias <- abs(Vx_FS[8:13])
```

- For the carps, the maximum amplitude during the first strike is 92.12 $\pm$ 59.88 cm/s.
- For the tilapias, the maximum amplitude during the first strike is 97.8 $\pm$ 23.08 cm/s.

The high standard deviation for the carps is due to trial C1P2T13, with a huge amplitude compared to the other trials.

Reverse flow

We use the maximum of the **average_VX** data to obtain the maximum amplitude the water tracers motion during the reverse flow because the X axis is pointing toward the back of the buccal cavity.

```
# Compute the min of the average_X
Vx_RF <-  map_dbl(.x = Trials,
                  .f = ~ abs(max(Data_W[[.]]$average_VX, na.rm = TRUE)))

ReverseFlow_magnitude <-  data.frame(Trial = names(Maya_WF),
                                     ReverseFlow_magnitude = Vx_RF)

ReverseFlow_magnitude %>%
  kable() %>%
  kable_styling()
```

Trial	ReverseFlow_magnitude
C1P1T08	34.13381

Trial	ReverseFlow_magnitude
C1P1T09	43.77259
C1P2T10	36.57802
C1P2T13	47.48390
C2P1T03	28.91493
C2P1T04	23.06357
C2P2T17	16.26535
T1P1T06	23.91757
T1P1T09	19.82174
T1P1T17	47.96827
T2P1T10	15.56411
T2P1T11	23.88300
T2P1T14	17.59310

```
# Compute the mean among a given species

## carps
Vx_RF_Carps <- abs(Vx_RF[1:7])

## tilapias
Vx_RF_Tilapias <- abs(Vx_RF[8:13])
```

- For the carps, the maximum velocity amplitude is 32.89 $\pm$ 11.07 cm/s
- For the tilapias, the maximum velocity amplitude is 24.79 $\pm$ 11.84 cm/s

Relative proportion of tx, ty, tz

We want to calculate the relative contribution of each component of the trajectory.

```
Data_Proportion <-  map(.x = Trials,
                        .f = ~ mutate(
                          .data = Data_W[[.]],
                          tr = abs(Data_W[[.]]$average_TX) +
                            abs(Data_W[[.]]$average_TY) +
                            abs(Data_W[[.]]$average_TZ)))

Data_Proportion <-  map(.x = Trials,
                        .f = ~ mutate(
                          .data = Data_Proportion[[.]],
                          tx_prop = abs(Data_Proportion[[.]]$average_TX) /
                            Data_Proportion[[.]]$tr,
                          ty_prop = abs(Data_Proportion[[.]]$average_TY) /
                            Data_Proportion[[.]]$tr,
                          tz_prop = abs(Data_Proportion[[.]]$average_TZ) /
```

```
                            Data_Proportion[[.]]$tr )) %>%
  set_names(names(Sum_dist_WF))
```

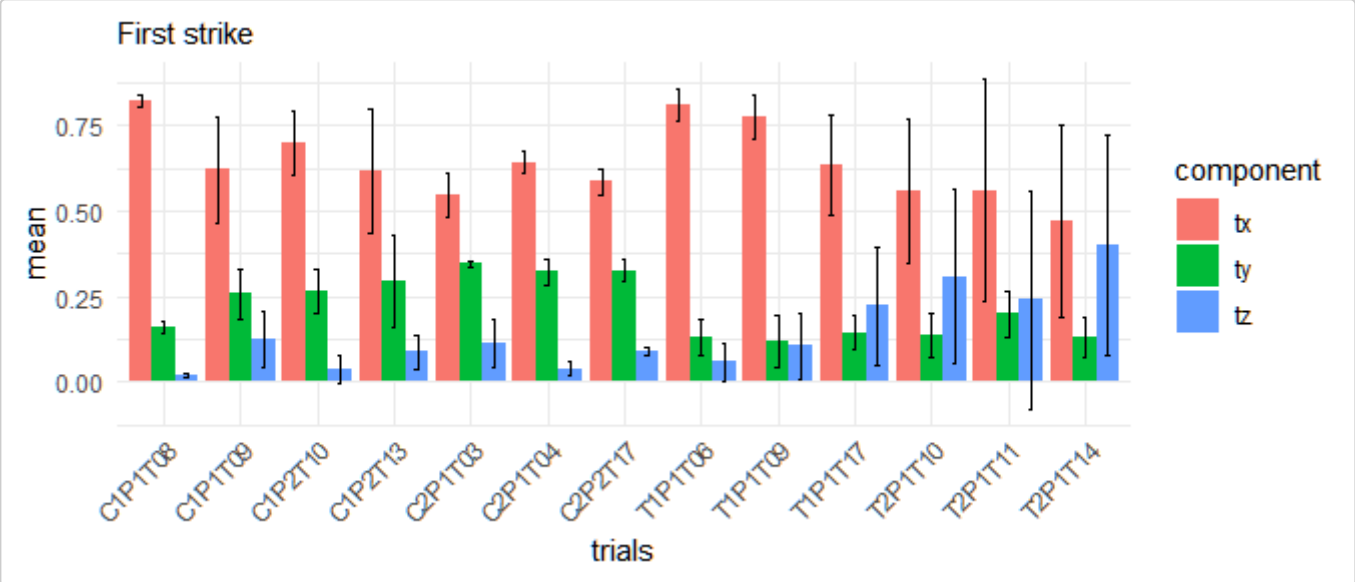
Based on these data, we plot an histogram corresponding to the x, y, and z components contribution for each phase of the suction feeding sequence among the 13 analyzed trials.

First strike

We extract the first strike data, based on the “phase” variable.

```
# reminder
#  1: first strike
#  0: stasis
# -1: reverse flow
#  2: back flow

FirstStrike_comp <- map(.x = Trials,
                        .f = ~ Data_Proportion[[.]][which(Data_Proportion[[.]]$phase == 1), ])
        %>%
  set_names(names(Data_Proportion))


Mean_FirstStrike_comp <-
  map(.x = Trials,
      .f = ~ FirstStrike_comp[[.]] %>%
        summarise(
          mean_tx_prop = mean(tx_prop, na.rm = TRUE),
          mean_ty_prop = mean(ty_prop, na.rm = TRUE),
          mean_tz_prop = mean(tz_prop, na.rm = TRUE),
          sd_tx_prop =  sd(tx_prop, na.rm = TRUE),
          sd_ty_prop =  sd(ty_prop, na.rm = TRUE),
          sd_tz_prop =  sd(tz_prop, na.rm = TRUE))) %>%
  set_names(names(FirstStrike_comp))


Mean_FirstStrike_comp__ <- data.frame(
  mean = c(
    map_dbl(
      .x = Trials,
      .f = ~ Mean_FirstStrike_comp[[.]]$mean_tx_prop
    ),
    map_dbl(
      .x = Trials,
      .f = ~ Mean_FirstStrike_comp[[.]]$mean_ty_prop
    ),
    map_dbl(
      .x = Trials,
      .f = ~ Mean_FirstStrike_comp[[.]]$mean_tz_prop
    )
  ),
  sd = c(
    map_dbl(
      .x = Trials,
      .f = ~ Mean_FirstStrike_comp[[.]]$sd_tx_prop
    ),
    map_dbl(
```

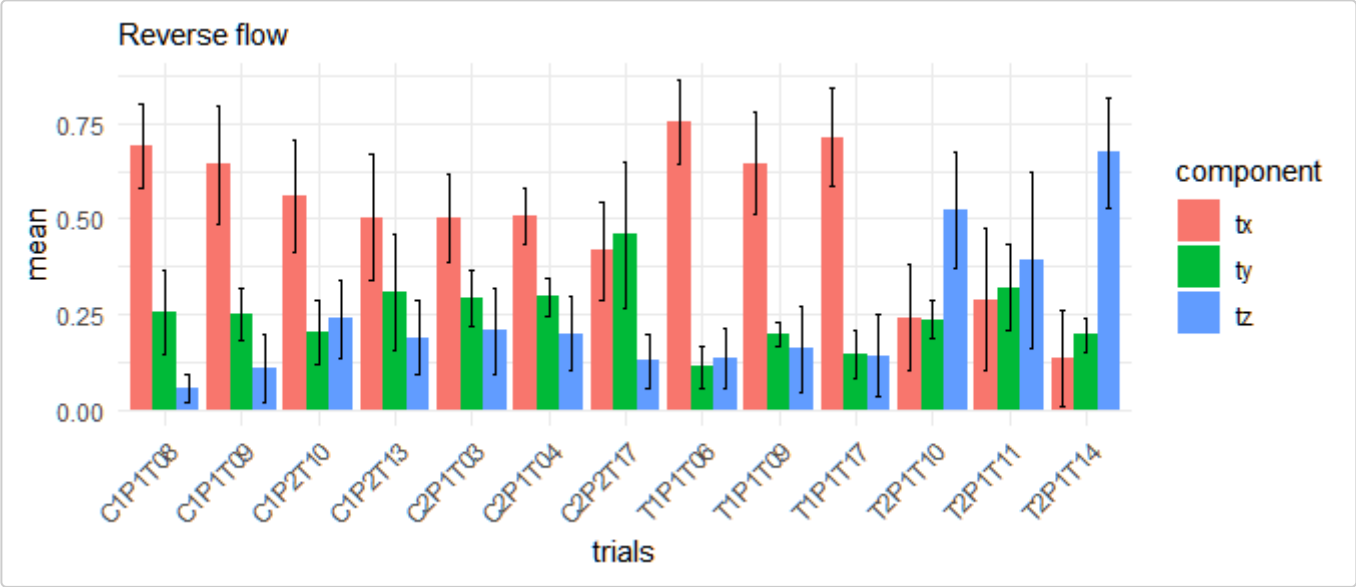
```
      .x = Trials,
      .f = ~ Mean_FirstStrike_comp[[.]]$sd_ty_prop
    ),
    map_dbl(
      .x = Trials,
      .f = ~ Mean_FirstStrike_comp[[.]]$sd_tz_prop
    )
  ),
  species = c(rep("carps", 7), rep("tilapias", 6)),
  trials = names(Mean_FirstStrike_comp),
  component = c(rep("tx", length(Trials)),
                rep("ty", length(Trials)),
                rep("tz", length(Trials)))
)

# plot the First strike data for each trial

ggplot(data = Mean_FirstStrike_comp__,
       aes(x = trials,
           y = mean,
           fill =component)) +
  geom_bar(stat = "identity",
           position = position_dodge()) +
  geom_errorbar(aes(ymin = mean - sd,
                    ymax = mean + sd),
                width = .2,
                position = position_dodge(0.9)) +
  theme_minimal() +
  # theme(legend.position = "none") +
  guides(x =  guide_axis(angle = 45)) +
  labs(subtitle = "First strike")
```



Reverse flow

We extract the reverse flow data, based on the “phase” variable and plot the relative contribution of each variable during the reverse flow phase

```r
# reminder
#  1: first strike
#  0: stasis
# -1: reverse flow
#  2: back flow

RF_comp <-
  map(.x = Trials, ~ Data_Proportion[[.]][which(Data_Proportion[[.]]$phase == -1), ])  %>%
  set_names(names(Data_Proportion))


Mean_RF_comp <- map(
  .x = Trials,
  .f = ~ RF_comp[[.]] %>%
    summarise(
      mean_tx_prop = mean(tx_prop, na.rm = TRUE),
      mean_ty_prop = mean(ty_prop, na.rm = TRUE),
      mean_tz_prop = mean(tz_prop, na.rm = TRUE),
      sd_tx_prop =  sd(tx_prop, na.rm = TRUE),
      sd_ty_prop =  sd(ty_prop, na.rm = TRUE),
      sd_tz_prop =  sd(tz_prop, na.rm = TRUE)
    )
) %>%
  set_names(names(RF_comp))


Mean_RF_comp__ <- data.frame(
  mean = c(
    map_dbl(.x = Trials, .f = ~ Mean_RF_comp[[.]]$mean_tx_prop),
    map_dbl(.x = Trials, .f = ~ Mean_RF_comp[[.]]$mean_ty_prop),
    map_dbl(.x = Trials, .f = ~ Mean_RF_comp[[.]]$mean_tz_prop)
  ),
  sd = c(
    map_dbl(.x = Trials, .f = ~ Mean_RF_comp[[.]]$sd_tx_prop),
    map_dbl(.x = Trials, .f = ~ Mean_RF_comp[[.]]$sd_ty_prop),
    map_dbl(.x = Trials, .f = ~ Mean_RF_comp[[.]]$sd_tz_prop)
  ),
  species = c(rep("carps", 7), rep("tilapias", 6)),
  trials = names(Mean_RF_comp),
  component = c(rep("tx", length(Trials)),
                rep("ty", length(Trials)),
                rep("tz", length(Trials)))
)

# plot the RF data for each trial

ggplot(data = Mean_RF_comp__, aes(x = trials, y = mean, fill = component)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  geom_errorbar(aes(ymin = mean - sd, ymax = mean + sd),
                width = .2,
                position = position_dodge(0.9)) +
  theme_minimal() +
  # theme(legend.position = "none") +
  guides(x =  guide_axis(angle = 45)) +
  labs(subtitle = "Reverse flow")
```

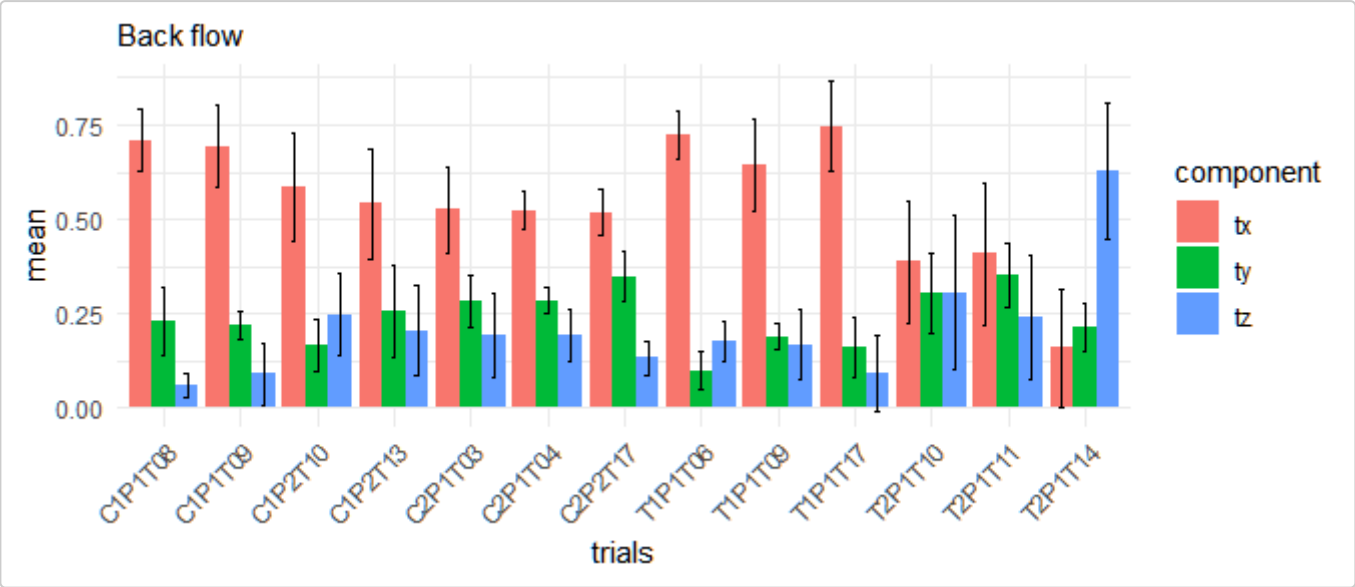


Back Flow

We extract the back flow data, based on the “phase” variable and plot the relative contribution of each variable during the back flow phase.

```
# reminder
#  1: first strike
#  0: stasis
# -1: reverse flow
#  2: back flow

BF_comp <-
  map(.x = Trials, ~ Data_Proportion[[.]][which(Data_Proportion[[.]]$phase == 2), ])  %>%
  set_names(names(Data_Proportion))


Mean_BF_comp <- map(
  .x = Trials,
  .f = ~ BF_comp[[.]] %>%
    summarise(
      mean_tx_prop = mean(tx_prop, na.rm = TRUE),
      mean_ty_prop = mean(ty_prop, na.rm = TRUE),
      mean_tz_prop = mean(tz_prop, na.rm = TRUE),
      sd_tx_prop =  sd(tx_prop, na.rm = TRUE),
      sd_ty_prop =  sd(ty_prop, na.rm = TRUE),
      sd_tz_prop =  sd(tz_prop, na.rm = TRUE)
    )
) %>%
  set_names(names(BF_comp))


Mean_BF_comp__ <- data.frame(
  mean = c(
    map_dbl(.x = Trials, .f = ~ Mean_BF_comp[[.]]$mean_tx_prop),
    map_dbl(.x = Trials, .f = ~ Mean_BF_comp[[.]]$mean_ty_prop),
    map_dbl(.x = Trials, .f = ~ Mean_BF_comp[[.]]$mean_tz_prop)
  ),
  sd = c(
```

```
    map_dbl(.x = Trials, .f = ~ Mean_BF_comp[[.]]$sd_tx_prop),
    map_dbl(.x = Trials, .f = ~ Mean_BF_comp[[.]]$sd_ty_prop),
    map_dbl(.x = Trials, .f = ~ Mean_BF_comp[[.]]$sd_tz_prop)
  ),
  species = c(rep("carps", 7), rep("tilapias", 6)),
  trials = names(Mean_BF_comp),
  component = c(rep("tx", length(Trials)),
                rep("ty", length(Trials)),
                rep("tz", length(Trials)))
)


# plot the Back flow data for each trial

ggplot(data = Mean_BF_comp__, aes(x = trials, y = mean, fill = component)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  geom_errorbar(aes(ymin = mean - sd, ymax = mean + sd),
                width = .2,
                position = position_dodge(0.9)) +
  theme_minimal() +
  # theme(legend.position = "none") +
  guides(x =  guide_axis(angle = 45)) +
  labs(subtitle = "Back flow")
```



Mean Relative proportion of each component during each phase

We average the previous data on all the trials of a given species. The data correspond to Table 2 in the manuscript.

First strike

```
comp <- c("tx", "ty", "tz")
color_components <-
  c(palette_IOFLOW[4], palette_IOFLOW[7], palette_IOFLOW[5])
```

```
Mean_first_strike_Comp <-
  data.frame(
    mean = c(map_dbl(
      .x = comp,
      .f = ~ mean(Mean_FirstStrike_comp__[Mean_FirstStrike_comp__$component == . &
                                           Mean_FirstStrike_comp__$species == "carps" , ]$mean,
         na.rm = TRUE)),
      map_dbl(
        .x = comp,
        .f = ~ mean(Mean_FirstStrike_comp__[Mean_FirstStrike_comp__$component == . &
                                             Mean_FirstStrike_comp__$species == "tilapias",
         ]$mean, na.rm = TRUE))),
    sd = c(map_dbl(
      .x = comp,
      .f = ~ sd(Mean_FirstStrike_comp__[Mean_FirstStrike_comp__$component == . &
                                         Mean_FirstStrike_comp__$species == "carps" , ]$mean)),

    map_dbl(
      .x = comp,
      .f = ~ sd(Mean_FirstStrike_comp__[Mean_FirstStrike_comp__$component == . &
                                         Mean_FirstStrike_comp__$species == "tilapias" , ]$mean,
         na.rm = TRUE))),
    species = c(rep("carps", 3), rep("tilapias", 3)),
    component = rep(c("tx", "ty", "tz"), 2))

hist_FS <-
  ggplot(data = Mean_first_strike_Comp,
         aes(x = component,
             y = mean * 100,
             fill =  component)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  geom_errorbar(
    aes(ymin = mean * 100 - sd * 100,
        ymax = mean * 100 + sd * 100),
    width = .2,
    position = position_dodge(0.9)
  ) +
  theme_minimal() +
  theme(legend.position = "none") +
  ylab("Component contribution") +
  xlab("Relative proportion of each component ") +
  facet_wrap( ~ species, scales = "free") +
  scale_color_manual(values = color_components) +
  scale_fill_manual(values =  color_components) +
  labs(title = "" , subtitle = "a. First strike")
```

Reverse Flow

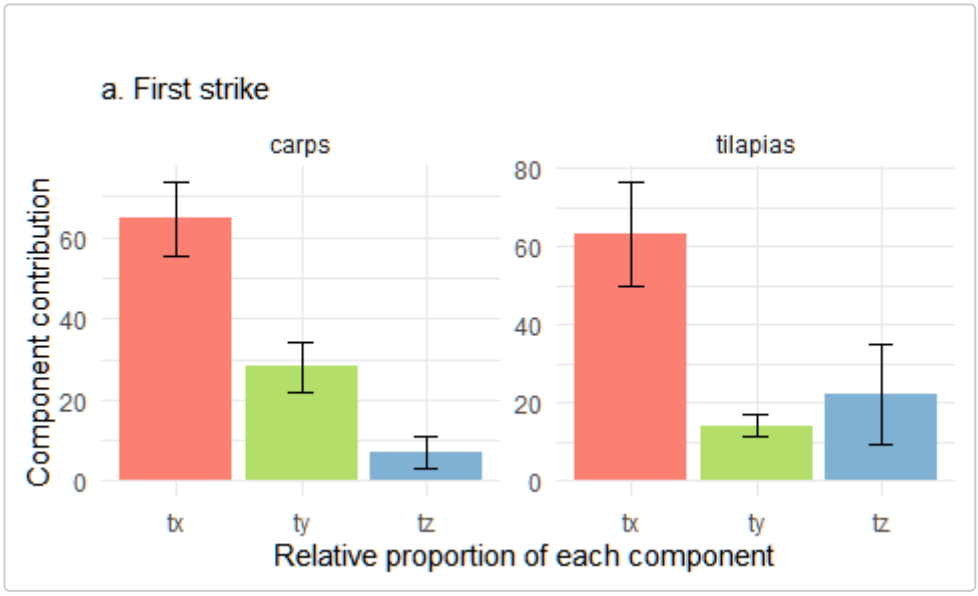
```
Mean_RF_Comp <-  data.frame(
  mean = c(map_dbl(
    .x = comp,
    .f = ~ mean(Mean_RF_comp__[Mean_RF_comp__$component == . &
                                 Mean_RF_comp__$species == "carps" ,]$mean, na.rm = TRUE)
  ),
  map_dbl(
    .x = comp,
```

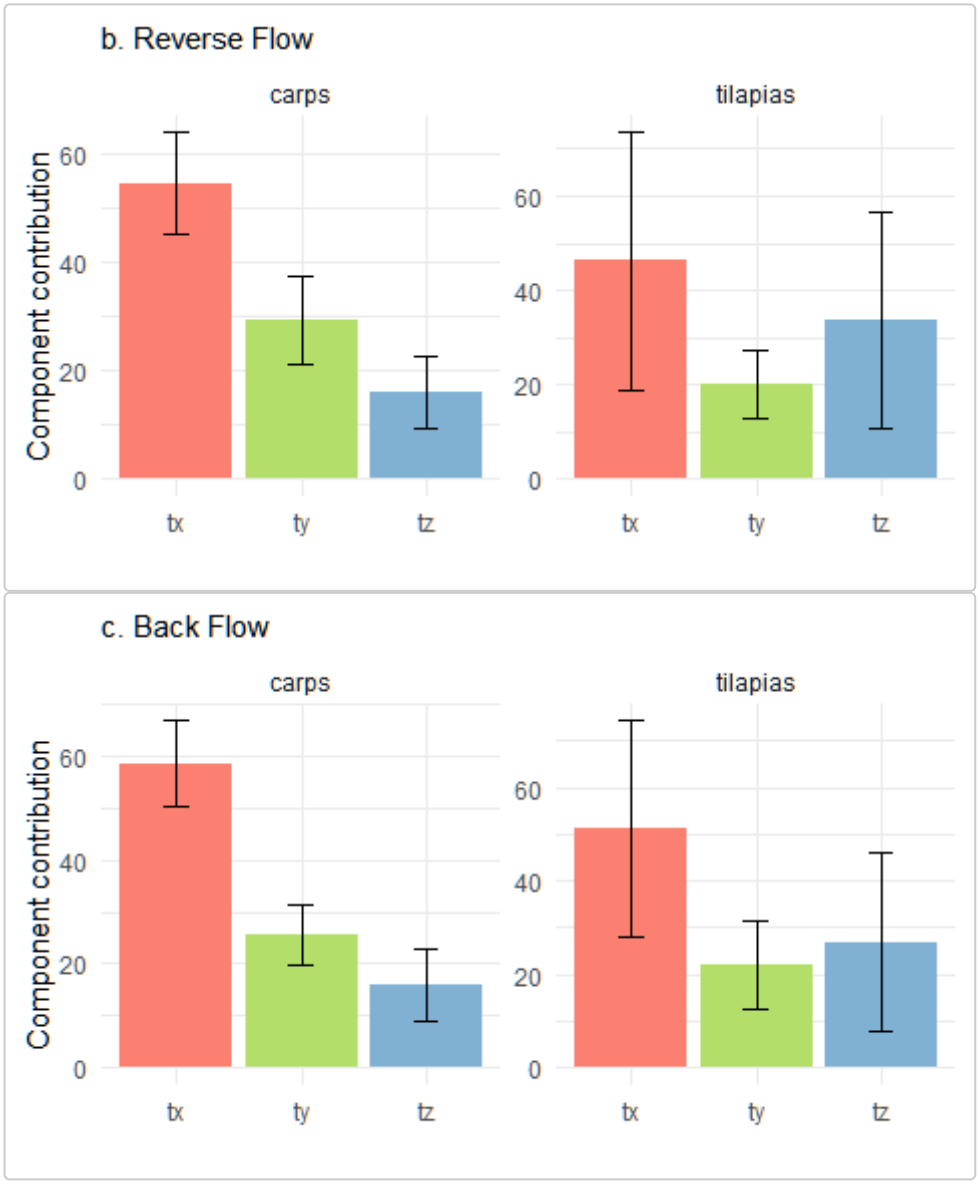
```
    .f = ~ mean(Mean_RF_comp__[Mean_RF_comp__$component == . &
                                 Mean_RF_comp__$species == "tilapias" ,]$mean, na.rm = TRUE)
  )),
  sd = c(map_dbl(
    .x = comp,
    .f = ~ sd(Mean_RF_comp__[Mean_RF_comp__$component ==
                               . &
                               Mean_RF_comp__$species == "carps" ,]$mean)
  ),
  map_dbl(
    .x = comp,
    .f = ~ sd(Mean_RF_comp__[Mean_RF_comp__$component == . &
                               Mean_RF_comp__$species == "tilapias" ,]$mean, na.rm = TRUE)
  )),
  species = c(rep("carps", 3), rep("tilapias", 3)),
  component = rep(c("tx", "ty", "tz"), 2)
)


hist_RF <-
  ggplot(data = Mean_RF_Comp, aes(x = component,
                                  y = mean * 100,
                                  fill = component)) +
  geom_bar(stat = "identity",
           position = position_dodge()) +
  geom_errorbar(
    aes(ymin = mean * 100 - sd * 100, ymax = mean * 100 + sd * 100),
    width = .2,
    position = position_dodge(0.9)) +
  theme_minimal() +
  theme(legend.position = "none") +
  ylab("Component contribution") +
  xlab("") +
  facet_wrap( ~ species, scales = "free") +
  scale_color_manual(values = color_components) +
  scale_fill_manual(values =  color_components) +
  labs(subtitle = "b. Reverse Flow")
```

Back flow

```
Mean_BF_Comp <-
  data.frame(
    mean = c(map_dbl(
      .x = comp,
      .f = ~ mean(Mean_BF_comp__[Mean_BF_comp__$component == . &
                                   Mean_BF_comp__$species == "carps", ]$mean, na.rm = TRUE)
    ), map_dbl(
      .x = comp,
      .f = ~ mean(Mean_BF_comp__[Mean_BF_comp__$component == . &
                                   Mean_BF_comp__$species == "tilapias", ]$mean, na.rm = TRUE)
    )),
    sd = c(map_dbl(
      .x = comp,
      .f = ~ sd(Mean_BF_comp__[Mean_BF_comp__$component == . &
                                 Mean_BF_comp__$species == "carps" , ]$mean, na.rm = TRUE)
```

```
    ), map_dbl(
      .x = comp,
      .f = ~ sd(Mean_BF_comp__[Mean_BF_comp__$component == . &
                                 Mean_BF_comp__$species == "tilapias" , ]$mean, na.rm = TRUE)
    )),
    species = c(rep("carps", 3), rep("tilapias", 3)),
    component = rep(c("tx", "ty", "tz"), 2)
  )


hist_BF <-
  ggplot(data = Mean_BF_Comp, aes(x = component, y = mean * 100, fill = component)) +
  geom_bar(stat = "identity",
           position = position_dodge()) +
  geom_errorbar(
    aes(ymin = mean * 100 - sd * 100,
        ymax = mean * 100 + sd * 100),
    width = .2,
    position = position_dodge(0.9)) +
  theme_minimal() +
  theme(legend.position = "none") +
  ylab("Component contribution") +
  xlab("") +
  facet_wrap( ~ species, scales = "free") +
  scale_color_manual(values = color_components) +
  scale_fill_manual(values =  color_components) +
  labs(subtitle = "c. Back Flow")

hist_FS
hist_RF
hist_BF
```





During each phase, the anterio posterior component is the most important one. The vertical component is more important for the tilapias compared with the carps, especially during the reverse and back flows.

Angle path

We hypothesize that the waterflow, exiting through the opercula during each backflow, causes a passive sorting based on the different mass of the ingested particles. The inertia effect would lead the biggest particles (food item) to stay on the same path, following a midsagittal trajectory towards the esophagus. The cycles reverse flow / back flow put the tracers in suspension in the buccal cavity and increase the probability to be sensed with appropriate sensory organs and trapped (actively) by the pharyngeal jaws if they correspond to a food item.

We explored this hypothesis by detecting the shift in the waterflow trajectory in comparison to the food tracer trajectory.
-> Calculating the angle formed by the position of a given particle during three consecutive frame

Data preparation

We store all the tx, ty, and tz for all the particles of the first strike of each trial in a list.

```
Trajectory <- map(.x = Trials, .f = ~Map(as.data.frame(cbind),
                                         WF_tx = WF_tx [[.]][],
```

```
                                    WF_ty = WF_ty [[.]][],
                                    WF_tz = WF_tz [[.]][],
                                    Time = Phase[[.]][11]/750,
                                    Frame = Phase[[.]][11],
                                    Phase = Phase[[.]][14])) %>% set_names(names(Maya_WF))


Trajectory_df <- list_to_df(list = Trajectory, id.vars = c(
"value.WF_tx",
"value.WF_ty",
"value.WF_tz",
"value.Time",
"value.Frame",
"value.Phase"),
           replace_by = c("value.", ""),
           replace_L1 = "Trials",
           replace_L2 = "Particles")

# we add an item column
Trajectory_df <- Trajectory_df %>% mutate(item = case_when(
str_detect(Trajectory_df$Particles, "F") ~ "W",
str_detect(Trajectory_df$Particles, "A") ~ "F"))

# create a vector with the name of each specimen
Specimens <- levels(factor(substr(names(Maya_WF), start = 1, stop = 2)))


# list all the trials associated with one specimen
Trial_Specimens <- map(.x = Specimens,
                       .f =~levels(factor(Trajectory_df$Trials))[
                         which(substr(levels(factor(Trajectory_df$Trials)), start = 1, stop = 2)
         == .)]
                         ) %>%  set_names(Specimens)

# create a df with the data for the First Strike

FS <-  Trajectory_df  %>%  filter(Phase == "1") %>% mutate(Grp = paste(Particles,Trials,
          sep="_"))

FS_Specimen <- map(.x = c(1:4), .f = ~subset(FS, subset= is.element(Trials,
          Trial_Specimens[[.]]))) %>% set_names(Specimens)

FS_Trial <- map(.x = Trials, .f = ~filter(FS, Trials == names(Maya_WF)[.])) %>%
          set_names(names(Maya_WF))
```

Trajectory cleaning

We want to filter and clean the data to only have the First Strike without and clean the trajectory of the particles that circle back towards the entrance of the mouth. We also add a calculation of the percentage of the anteroposterior position
- 0% corresponds to the entrance of the mouth
- 100% corresponds to the entrance of the esophagus.

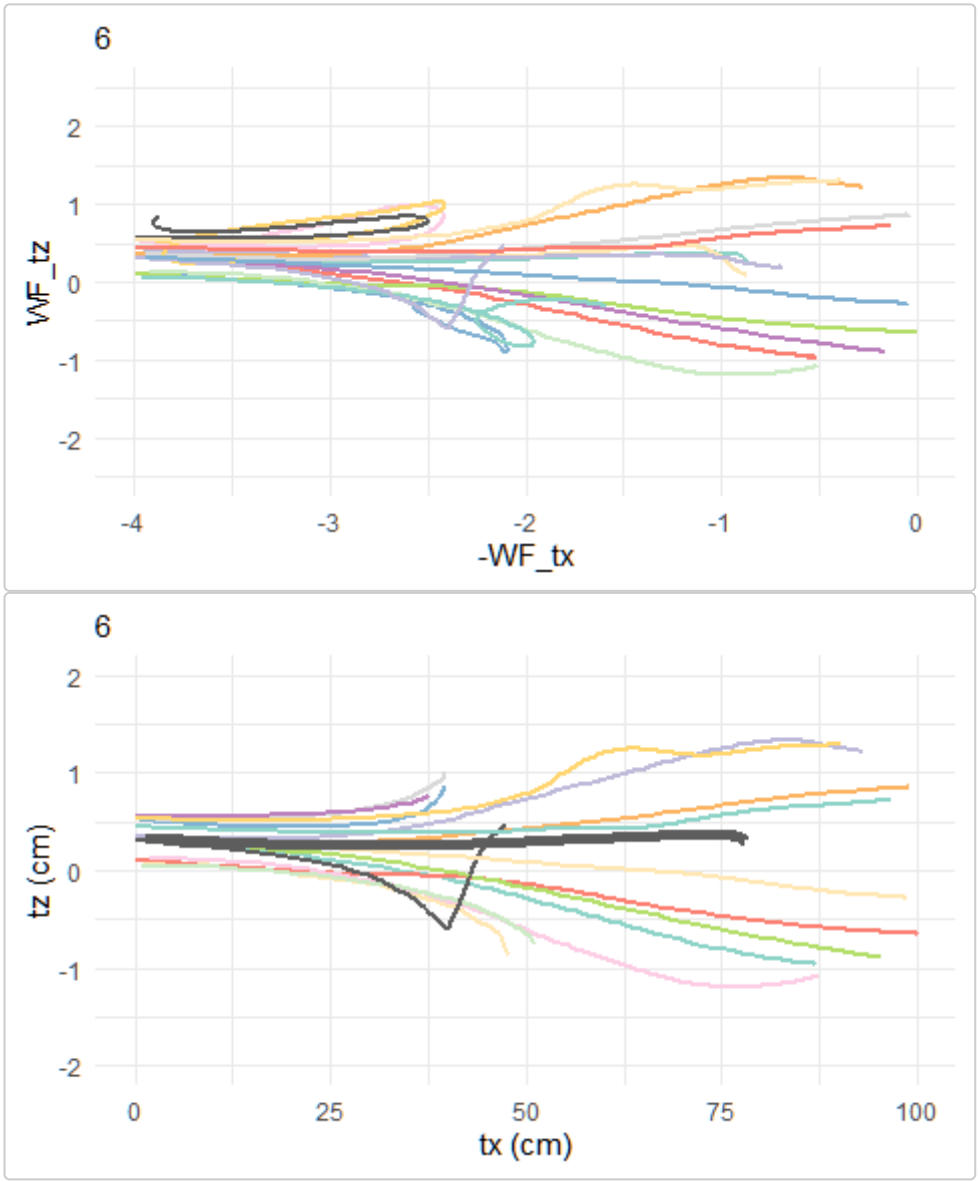
```
FS_Trial_clean <- list()
```

```
for(trial in Trials){
  data <- FS_Trial[[trial]]

  # we start the analysis when the particles passed the entrance of the mouth (different between
      specimens)
  if (str_starts(string = names(Maya_WF)[trial], pattern = "C2")) {
    data_cut <- data %>% filter(WF_tx > 0) %>%
      filter(WF_tx < 4)
  } else {
    data_cut <- data %>% filter(WF_tx > 0) %>%
      filter(WF_tx < 6)
  }

  # We want to clean the particles that circle back towards the entrance of the mouth (using lag)

  Particles <- levels(factor(data_cut$Grp)) # all particles in a given trial of Trial

  Curling <- list()
  data_cut_clean <- 0
  for (i in 1:length(Particles)){
    A <- data_cut[which(data_cut$Grp == Particles[i]),]
    Curling[[i]] <- which(diff(x = A$WF_tx, lag = 1) >0)[1]  # gives the beginning of the curling
    if (is.na(Curling[[i]]) == TRUE) { Curling[[i]] = nrow(A) }

    A <- A[1: Curling[[i]],]
    data_cut_clean <- rbind(data_cut_clean, A)
  }

  Curling <- Curling %>% set_names(Particles)
  data_cut_clean <- data_cut_clean[-1,]

  data_cut_clean$WF_tx_pcent <- 100 - data_cut_clean$WF_tx/(max(data_cut_clean$WF_tx, na.rm=TRUE
      ))*100

  FS_Trial_clean[[trial]] <- data_cut_clean

}

FS_Trial_clean <- FS_Trial_clean %>% set_names(names(Maya_WF)[Trials])
```

We can visualize the cleaning process for a given trial (C2P1T04).

```
# FOR VISUALIZATION
# all the data for a given trial (several trials)

trial <- 6 # trial C2P1T04
data <- FS_Trial[[trial]]

data_cut_example <- data %>% filter(WF_tx > 0) %>% filter(WF_tx < 4)

#plot of the uncleaned path:

color <-c(palette_IOFLOW, palette_IOFLOW)
ggplot(data = data_cut_example,
              aes(
                x = -WF_tx,
                y = WF_tz )) +
```

```
 geom_path(aes(group=Grp, color= Grp), size=0.8)+ ylim(-2.5,2.5)+ labs(subtitle = trial )+
        scale_color_manual(values = color)+
      theme_minimal()+
    theme(legend.position = "none")

# We want to clean the particles that circle back towards the entrance of the mouth (using lag)
Particles <- levels(factor(data_cut_example$Grp)) # all particles in a given trial of Trial
Curling <- list()
data_cut_clean_example <- 0
for (i in 1:length(Particles)){
  A <- data_cut_example[which(data_cut_example$Grp == Particles[i]),]
  Curling[[i]] <- which(diff(x = A$WF_tx, lag = 1) >0)[1]  # gives the beginning of the curling
  if (is.na(Curling[[i]]) == TRUE) { Curling[[i]] = nrow(A) }

  A <- A[1: Curling[[i]],]
  data_cut_clean_example <- rbind(data_cut_clean_example, A)
}
Curling <- Curling %>% set_names(Particles)



data_cut_clean_example <- data_cut_clean_example[-1,]
data_cut_clean_example$WF_tx_pcent <- 100 -
        data_cut_clean_example$WF_tx/(max(data_cut_clean_example$WF_tx, na.rm=TRUE ))*100

#plot of the cleaned path
color <- c(palette_IOFLOW[13], palette_IOFLOW, palette_IOFLOW)


data_cut_clean_Water_example <- data_cut_clean_example %>% filter(item == "W")
data_cut_clean_Food_example <- data_cut_clean_example %>% filter(Grp == "A01_data.tx_C2P1T04")

ggplot(data = data_cut_clean_Water_example,
              aes(
                x = WF_tx_pcent,
                y = WF_tz )) +
 geom_path(aes(group=Grp, color= Particles), size = 0.8)+
    ylim(-2,2)  +labs(subtitle = trial)+
    ylab("tz (cm)") +
    xlab("tx (cm)") +
  geom_path(data = data_cut_clean_Food_example, aes(group=Grp, color= item), size = 2)+
    scale_color_manual(values = color)+
      theme_minimal()+
    theme(legend.position = "none")
```



Path Angle calculation

Now we can calculate the angle made by the trajectory of each particle. We calculate the angle in a the dorsoventral view.

```
Angle <- list()
span <- 1 # for each frame

for(trial in Trials){
  Particles <- levels(factor(FS_Trial_clean[[trial]]$Grp))

  for (i in 1:length(Particles)){
    Temp_particle <- FS_Trial_clean[[trial]][which(
    FS_Trial_clean[[trial]]$Grp == Particles[i]),]

    for (j in (1+span):(nrow(Temp_particle)-(span))){
      A <- unlist(Temp_particle[(j-span),1:3])
      B <- unlist(Temp_particle[j,1:3])
      C <- unlist(Temp_particle[(j+span),1:3])

      A[2] <- 0 # we delete the y coordinate to calculate the angle in the dorsoventral view
      B[2] <- 0
      C[2] <- 0
```

```
      name <- str_replace(string = Particles[i], pattern =  "_data.tx", replacement = "" )

      Angle[[names(Maya_WF)[trial]]][[name]][[paste0(name, "_angle")]][j] <-
         as.vector(angle3D(A,B,C))
      Angle[[names(Maya_WF)[trial]]][[name]][[paste0(name, "_tx_pcent")]][j] <-
         unlist(Temp_particle $ WF_tx_pcent[j])
      Angle[[names(Maya_WF)[trial]]][[name]][[paste0(name, "_tx")]][j] <-
         unlist(Temp_particle[j,1])
      Angle[[names(Maya_WF)[trial]]][[name]][[paste0(name, "_ty")]][j] <-
         unlist(Temp_particle[j,2])
      Angle[[names(Maya_WF)[trial]]][[name]][[paste0(name, "_tz")]][j] <-
         unlist(Temp_particle[j,3])
    }
  }
}
```

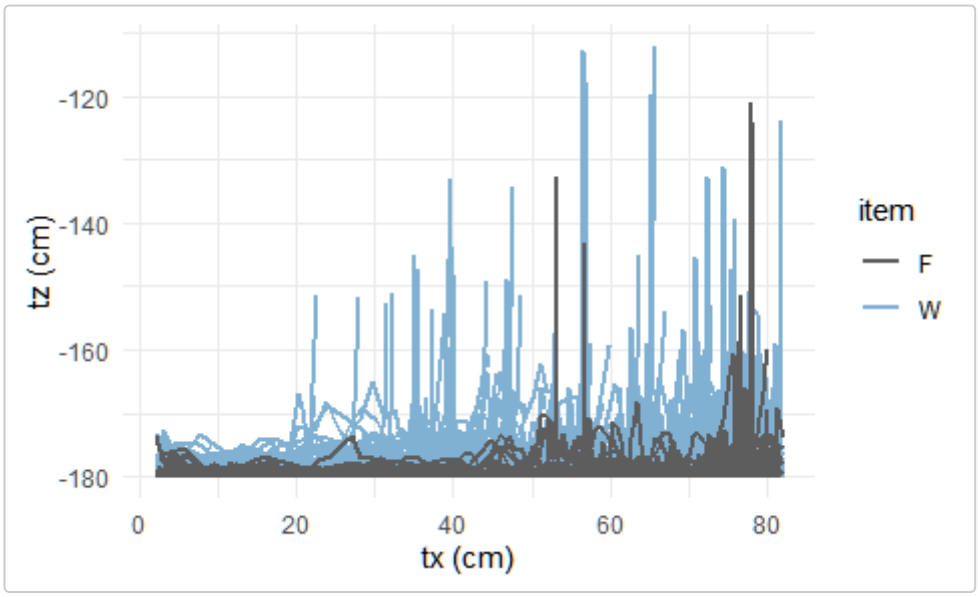
We need to transform the Angle list into a data.frame with the particle name, angle, tx, tx_pcent, and tz.

```
Angle_df_list <- unlist(unlist(Angle, recursive = FALSE), recursive = FALSE)

# we divide between angle, tx, tx_pcent, and tz (same length):
Angle_df_angle <- map(.x = str_which(string = names(Angle_df_list), pattern = "angle"), .f =
         ~get_sublist(lst = Angle_df_list, group_name = names(Angle_df_list)[[.]]))

Angle_df_tx_pcent <- map(.x = str_which(string = names(Angle_df_list), pattern = "_tx_pcent"), .f
         = ~get_sublist(lst = Angle_df_list, group_name = names(Angle_df_list)[[.]]))

Angle_df_tz <- map(.x = str_which(string = names(Angle_df_list), pattern = "_tz"), .f =
         ~get_sublist(lst = Angle_df_list, group_name = names(Angle_df_list)[[.]]))


Angle_df <- data.frame(particle = str_sub(string = melt(Angle_df_angle)[,2],
                          start = 21,
                          end = (nchar(melt(Angle_df_angle)[,2])-6)),
      angle = melt(Angle_df_angle)[,1],
      tx_pcent= melt(Angle_df_tx_pcent)[,1],
      tz= melt(Angle_df_tz)[,1] )

# we add a column with the item category:
Angle_df <- Angle_df %>% mutate(item = case_when(str_detect(string = particle, pattern = "F") ~
         "W",
                              str_detect(string = particle, pattern = "A") ~ "F"))
```

Food vs water path angle Visualisation

```
color <- c(palette_IOFLOW[13], # grey
           palette_IOFLOW[5]   # blue
)

Angle_df %>% filter(item == "W") %>% filter(tx_pcent > 2 & tx_pcent < 82) %>%
ggplot(aes(y=-angle, x=tx_pcent)) +
 geom_path(aes(group=particle, color= item), size = 0.8) +
 geom_path(data = Angle_df %>% filter(item == "F") %>% filter(tx_pcent > 2 & tx_pcent < 82),
         aes(group=particle, color= item), size = 0.8) +
```

```
    ylab("tz (cm)") +
    xlab("tx (cm)")+
    scale_color_manual(values = color)+
      theme_minimal()
```



Average of food and water tracer angle on a small span of tx

Data preparation

```
# split between water and food particles
Angle_df_water <-  Angle_df %>% filter(item == "W") %>% filter(tx_pcent > 2 & tx_pcent < 75)%>%
         arrange(desc(tx_pcent))
Angle_df_food <-  Angle_df %>% filter(item == "F") %>% filter(tx_pcent > 2 & tx_pcent < 75)%>%
         arrange(desc(tx_pcent)) %>% filter(str_detect(string = particle, pattern = "T2P1T10",
         negate = TRUE))

# we cut by the number of intervals we want (5 intervals)

breaks <- seq(from = 0, to = 100, by = 20)

# water -----------------
Angle_interval_water <- Angle_df_water %>%
    mutate(interval = cut(tx_pcent,
                          breaks,
                          include.lowest = TRUE,
                          right = FALSE)) %>% group_by(interval)
Angle_interval_water_mean <-  Angle_interval_water %>%   summarise( mean.angle = mean(angle,
         na.rm=TRUE), sd.angle = sd(angle, na.rm=TRUE) )
#> `summarise()` ungrouping output (override with `.groups` argument)

 # food -----------------
Angle_interval_food <- Angle_df_food %>%
    mutate(interval = cut(tx_pcent,
                          breaks,
                          include.lowest = TRUE,
                          right = FALSE)) %>% group_by(interval)

 Angle_interval_food_mean <- Angle_interval_food %>% summarise( mean.angle = mean(angle,
         na.rm=TRUE) , sd.angle = sd(angle, na.rm=TRUE) )
```

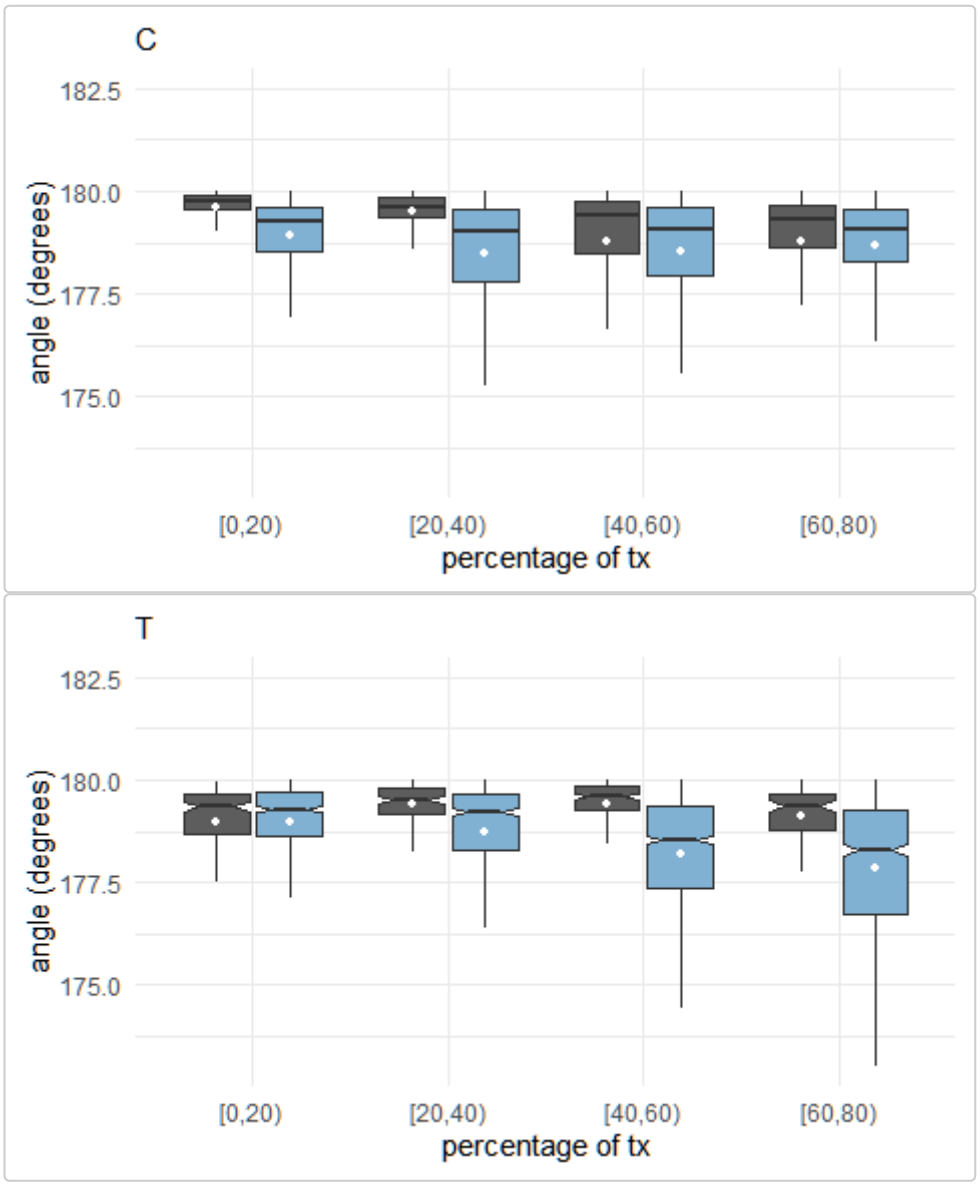
```
#> `summarise()` ungrouping output (override with `.groups` argument)

# merge water and food particles
 Angle_interval<-  rbind(Angle_interval_water, Angle_interval_food) %>% arrange(desc(interval))
        %>%
 mutate(item = case_when(str_detect(string = particle, pattern = "F") ~ "W",
                         str_detect(string = particle, pattern = "A") ~ "F"))

 # # organized by trial
 #  Angle_interval_trial <- map(.x = Trials, .f = ~(filter(.data = Angle_interval,
        str_detect(string = particle, pattern = names(Maya_WF[.])))))) %>%
        set_names(names(Maya_WF[Trials]))
```

Visualisation

Now that we have the angle for intervals of tx for the water and the food items, we organize the data by Species and visualize the difference between each item (Figure 4 in the manuscript).

```
 # Either Carp of Tilapia:
    Species <- c("C", "T")

 Angle_interval_species <- map(.x = Species, .f = ~(filter(.data = Angle_interval,
        str_detect(string =  str_sub(string = particle, start = 4, end = 7 ), pattern = .))))
        %>% set_names(Species)

   # plot the Carp data:
  species <- "C"

  color <- c(palette_IOFLOW[13], # grey
             palette_IOFLOW[5]   # blue
  )



  ggplot(Angle_interval_species[[species]], aes(x=interval, y=angle, fill=item)) +
    geom_boxplot(notch=FALSE
                 , outlier.shape = NA
                 # , outlier.shape=8,  outlier.size=4
    ) +
    labs(x="percentage of tx", y = "angle (degrees)") +
    scale_color_manual(values = color)+
    scale_fill_manual(values =  color)+ labs(subtitle = species )+
    theme_minimal() +
    stat_summary(fun=mean, geom="point", size=1, col = "white", position = position_dodge(0.75))
        +
    theme(legend.position = "none") +
    ylim(173,182.5)
#> Warning: Removed 185 rows containing non-finite values (stat_boxplot).
#> Warning: Removed 185 rows containing non-finite values (stat_summary).


  # plot the Tilapia data:
  species <- "T"

  ggplot(Angle_interval_species[[species]], aes(x=interval, y=angle, fill=item)) +

    geom_boxplot(notch=TRUE
                 , outlier.shape = NA
```

```
                # , outlier.shape=8,  outlier.size=4
    ) +
    labs(x="percentage of tx", y = "angle (degrees)") +
    scale_color_manual(values = color)+
    scale_fill_manual(values =  color)+ labs(subtitle = species )+
    theme_minimal() +
    stat_summary(fun=mean, geom="point", size=1, col = "white", position = position_dodge(0.75))
        +
    theme(legend.position = "none") +
    ylim(173,182.5)
#> Warning: Removed 161 rows containing non-finite values (stat_boxplot).
#> Warning: Removed 161 rows containing non-finite values (stat_summary).
```



Statistical test

We perform a t.test on each interval.

```
# Carps
species <- "C"
Intervals <-levels(Angle_interval_species[[species]]$interval)
Intervals <- Intervals[1:4]
t_test_results_Carps <- map(.x = Intervals,
                            .f = ~(t.test(angle ~ item,
```

```
                                        data = filter(.data =
        Angle_interval_species[[species]], interval== .),
                                        alternative = "two.sided",
                                        mu = 0,
                                        paired = FALSE,
                                        var.equal = TRUE,
                                        conf.level = 0.95))) %>% set_names(Intervals)


t_test_p_value_Carps <-map(.x = Intervals,
                           .f = ~get_sublist(lst = t_test_results_Carps[[.]], group_name =
        "p.value")) %>% set_names(Intervals)


# Tilapias
species <- "T"
t_test_results_Tilapias <- map(.x = Intervals,
                            .f = ~(t.test(angle ~ item,
                                        data = filter(.data =
        Angle_interval_species[[species]], interval== .),
                                        alternative = "two.sided",
                                        mu = 0,
                                        paired = FALSE,
                                        var.equal = TRUE,
                                        conf.level = 0.95))) %>% set_names(Intervals)


t_test_p_value_Tilapias <-map(.x = Intervals,
                           .f = ~get_sublist(lst = t_test_results_Tilapias[[.]], group_name =
        "p.value")) %>% set_names(Intervals)
```

Rigid body kinematics

We want to visualise on the same plot - Food tx - Gape (open close) -> distance inferior vs posterior - Hyoid dorsoventral -> tip of hyoid relative to pharyngeal jaw (ty) - Ope opening/closing -> tip of ope relative to pharyngeal jaw (tz): eloignement de midsagittal

Building Kdata

We want to combine KData (the kinematics data) and MayaData (the tracer trajectory data) and add Time, Frame, Phase.

```
Time <- map(.x = Trials[1:10],
             .f = ~as.data.frame(cbind(
              Time = Trajectory[[.]][[1]]$value.Time,
              Frame = Trajectory[[.]][[1]]$value.Frame,
              Phase = Trajectory[[.]][[1]]$value.Phase
              )))%>%
  set_names(names(Maya_Loc))

Food <- map(.x = Trials[1:10],
            .f = ~(as.data.frame(cbind(
              Mean_tx_F = Mean_tx_F[[.]]$average,
              Mean_ty_F = Mean_ty_F[[.]]$average,
```

```
              Mean_tz_F = Mean_tz_F[[.]]$average
              )))) %>%
  set_names(names(Maya_Loc))


Water <- map(.x = Trials[1:10],
             .f = ~(as.data.frame(cbind(
               Mean_tx_W = Mean_tx_W[[.]]$average,
               Mean_ty_W = Mean_ty_W[[.]]$average,
               Mean_tz_W = Mean_tz_W[[.]]$average
               )))) %>%
  set_names(names(Maya_Loc))


# merging the data:
KData <- mapply(FUN = c, Time, Food, Water, Maya_Loc, SIMPLIFY=FALSE)
```

We calculate the Gape, using the locators placed on the tip of the lower and upper jaws.

```
# Gape calculation

Jaws <- map(.x = Trials[1:10],
             .f = ~ (select(
               KData[[.]]$BonesLocators_relative,
               names(KData[[.]]$BonesLocators_relative)
               [contains(match = "jaw_tip_data.t",
                         vars = names(KData[[.]]$BonesLocators_relative))]
             ))) %>%
  set_names(names(Maya_Loc))


Gape <- map(.x = Trials[1:10],
             .f = ~ (as.data.frame(cbind(
               Gape = distance(
                 df1 = select(Jaws[[.]], names(Jaws[[.]])[contains(match = "lower_jaw", vars =
         names(Jaws[[.]]))]),
                 df2 = select(Jaws[[.]], names(Jaws[[.]])[contains(match = "upper_jaw", vars =
         names(Jaws[[.]]))])))))) %>%
  set_names(names(Maya_Loc))

# merging the data:
KData <- mapply(FUN = c, KData,  Gape, SIMPLIFY=FALSE)
```

Plots kinematics

Carps

We plot the suction feeding sequence of trial C2P1T04 of carp to visualize the kinematics of Gape, Hyoid depression and Opercula opening relative to the water motions.

```
# Carps

trial <- "C2P1T04"
subtime <- 1:1592
```

```r
xlim_inf <- min(subtime)
xlim_sup <- max(subtime)

subtitle <- " "
d <- cbind(Mean_tx_W[[trial]][subtime,], time = Timing[[trial]]$time[subtime])
color <- "#808080"
size <- 1.5

# Plot

## Water
plot.Tx_water <- ggplot(data = d,
                        aes(
                          x = time,
                          y =  average,
                          ymin = (average + std),
                          ymax = (average - std), group=item
                        )) +
  geom_line(aes(colour = item), size= size) +
  geom_ribbon(aes(fill = item), alpha = 0.3, size= size) +
  ylab("") +
  labs(subtitle = "Water Tx (cm)")+
  theme_minimal()+
  theme(legend.position = "none",
        axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank()) +
  scale_x_continuous(breaks = NULL)+
  scale_color_manual(values = palette_IOFLOW[5])+
  scale_fill_manual(values =  palette_IOFLOW[5])

# Gape

KData_df <- as.data.frame(KData[[trial]])
KData_df <- KData_df[subtime,]

KData_Gape <-  ggplot(KData_df, aes(x = Time, y = Gape, colour="#808080")) + #, group="Particles"
  geom_line(aes(group=1), size=size)+
  ylab("") +
  xlab("") +
  labs(subtitle = "Gape (cm)")+
  # xlim(xlim_inf,xlim_sup)+
  theme_minimal()+
  theme(legend.position = "none",
        axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank()) +
  scale_x_continuous(breaks = NULL)+
  scale_color_manual(values = color)+
  scale_fill_manual(values =  color)

## Hyoid

KData_Hyoid <-  ggplot(KData_df, aes(x = Time, y = -BonesLocators_relative.hyoid_tip_data.ty,
         colour="#808080")) +
  geom_line(aes(group=1), size=size)+
  ylab("") +
  labs(subtitle = "Hyoid depression (cm)")+
```

```
  xlab("") +
  # xlim(xlim_inf,xlim_sup)+
  theme_minimal()+
  theme(legend.position = "none",
        axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank()) +
  scale_x_continuous(breaks = NULL)+
  scale_color_manual(values = color)+
  scale_fill_manual(values = color) #+

## Ope

KData_Ope <-  ggplot(KData_df, aes(x = Time, y = BonesLocators_relative.ope_tip_data.tz,
          colour="#808080")) + #, group="Particles"
  geom_line(aes(group=1), size=size)+
  ylab("") +
  labs(subtitle = "Opercula abduction (cm)")+
  # xlim(xlim_inf,xlim_sup)+
  theme_minimal()+
  theme(legend.position = "none") +
  xlab("Time (s)") +
  scale_x_continuous(breaks = NULL)+
  scale_color_manual(values = color)+
  scale_fill_manual(values =  color)


G1 <- rbind(ggplotGrob(plot.Tx_water),
            ggplotGrob(KData_Gape),
            ggplotGrob(KData_Hyoid),
            ggplotGrob(KData_Ope),
            size = "last")
```

Tilapias

We plot the suction feeding sequence of trial T1P1T06 of tilapia to visualize the kinematics of Gape, Hyoid depression and Opercula opening relative to the water motions.

```
# Tilapias

trial <- "T1P1T06"
subtime <- 100:1280
subtitle <- " "
xlim_inf <- min(subtime)
xlim_sup <- max(subtime)
d <- cbind(Mean_tx_W[[trial]][subtime,], time = Timing[[trial]]$time[subtime])

# Plot preparation

## Water
plot.Tx_water <- ggplot(data = d,
                        aes(
                          x = time,
                          y =  average,
                          ymin = (average + std),
```

```
                          ymax = (average - std), group=item
                        )) +
  geom_line(aes(colour = item), size= size) +
  geom_ribbon(aes(fill = item), alpha = 0.3, size= size) +
  ylab("") +
  labs(subtitle = "Water Tx (cm)")+
  theme_minimal()+
  theme(legend.position = "none",
        axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank()) +
  scale_x_continuous(breaks = NULL)+
  scale_color_manual(values = palette_IOFLOW[5])+
  scale_fill_manual(values =  palette_IOFLOW[5])

# Gape
KData_df <- as.data.frame(KData[[trial]])
KData_df <- KData_df[subtime,]

KData_Gape <-  ggplot(KData_df, aes(x = Time, y = Gape, colour="#808080")) + #, group="Particles"
  geom_line(aes(group=1), size=size)+
  ylab("") +
  labs(subtitle = "Gape (cm)")+
  theme_minimal()+
  theme(legend.position = "none",
        axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank()) +
  scale_x_continuous(breaks = NULL)+
  scale_color_manual(values = color)+
  scale_fill_manual(values =  color)

## Hyoid
KData_Hyoid <-  ggplot(KData_df, aes(x = Time, y = -BonesLocators_relative.hyoid_tip_data.ty,
         color="#808080")) +
  geom_line(aes(group=1), size=size)+
  labs(subtitle = "Hyoid depression (cm)")+
  ylab("") +
  theme_minimal()+
  scale_x_continuous(breaks = NULL)+
  theme(legend.position = "none",
        axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank()) +
  scale_color_manual(values = color) +
  scale_fill_manual(values = color)

## Ope
KData_Ope <-  ggplot(KData_df, aes(x = Time, y = -BonesLocators_relative.ope_tip_data.tz,
         colour="#808080")) +     geom_line(aes(group=1), size=size)+
   ylab("") +
  labs(subtitle = "Opercula abduction (cm)")+
  xlim(xlim_inf,xlim_sup)+
  theme_minimal()+
  theme(legend.position = "none") +
  xlab("Time (s)") +
  scale_x_continuous(breaks = NULL)+
  scale_color_manual(values = color)+
```

```
  scale_fill_manual(values =  color)
#> Scale for 'x' is already present. Adding another scale for 'x', which will
#> replace the existing scale.


G2 <- rbind(ggplotGrob(plot.Tx_water),
            ggplotGrob(KData_Gape),
            ggplotGrob(KData_Hyoid),
            ggplotGrob(KData_Ope),
            size = "last")
```

We combine the plots to create Figure 5 of the manuscript

```
# Plots

grid.newpage()

g_ <- cbind(G1, G2, size = "first")
grid.newpage()
grid.draw(g_)
```