Script2: Niche breadth estimation

Angel Rain, Sara Beier & Nicolas Mouquet

11/02/2021

Contents

All figure numbers in this document refer to figures in the manuscript 'Niche breadth affects bacterial transcription patterns along a salinity gradient' by A. Rain-Franco et al.

1 Packages

```
rm(list = ls())
library(ggplot2)
library(dplyr)
library(egg)  #Add lab to panel figures
library(cowplot)  #Formatting multipanel figure
library(xcms)  #Package for selfstarting method ('SSgauss')
library(kableExtra)  #Table formatting
library(e1071)  #Probability distribution characteristics
```

2 Load datasets

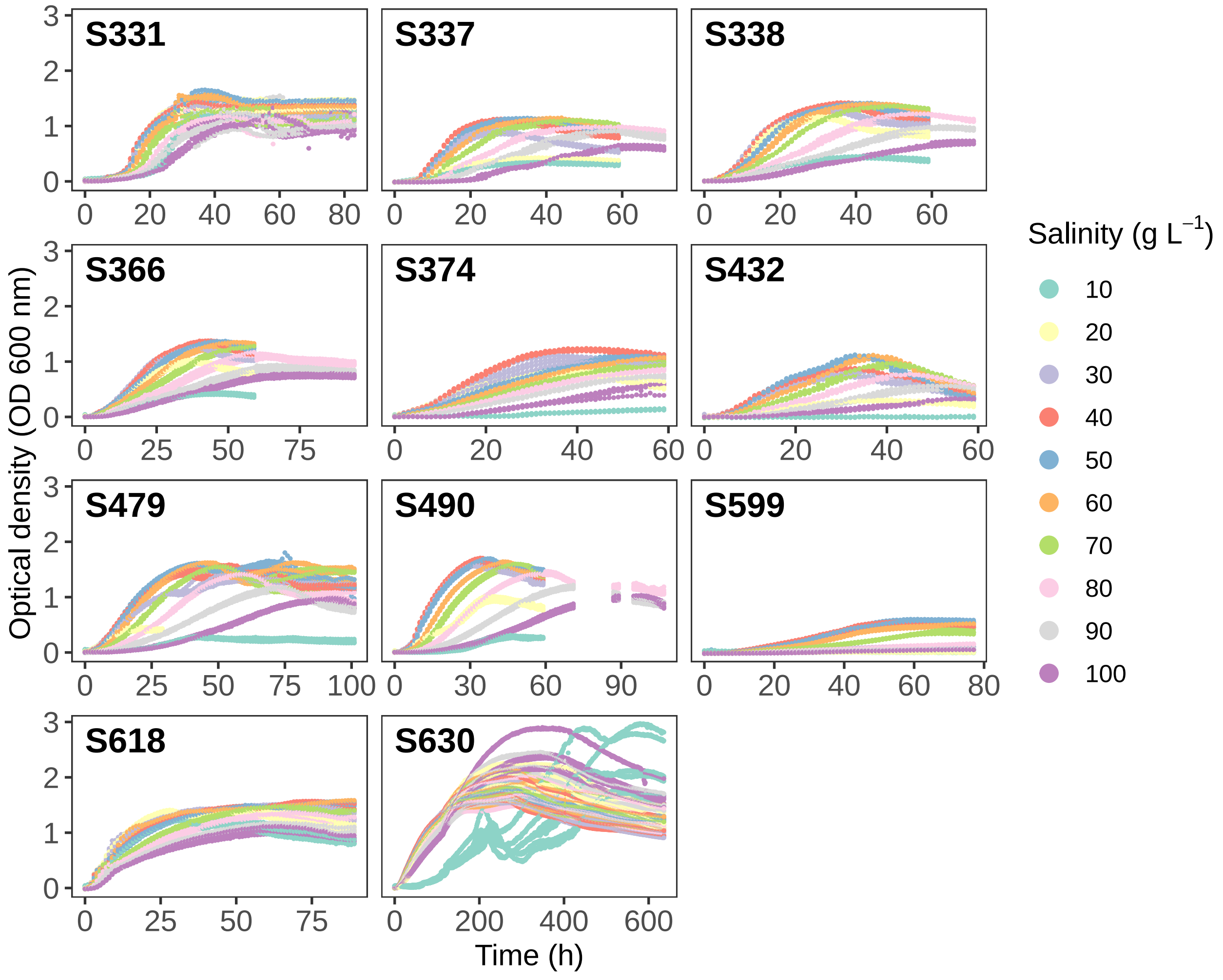
Load output data files from Paradigm Microplate reader growth (OD600) from the individual strains grown under salinity concentrations from 10-100 g L-1 NaCl. OD values from wells that were filled with medium but not inoculated with cells were considered as blanc values and subtracted from OD values of cells that were inoculated.

```
tmp = list()
S = c("331v2", "337v3", "338", "366v3", "374", "432", "479v3", "490v3", "599v2",
    "618v2", "630v4")
for (i in 1:11) {
    fi = paste0("../data_OD/", "S", S[i], ".txt")
    D <- read.table(fi, header = F)
    time <- c(0:(dim(D)[2] - 2))
    label <- D[, 1]
    D <- as.matrix(D[, 2:(dim(D)[2])])
    DM <- D[label != "Blanc", ] - mean(D[label == "Blanc", ], na.rm = T)
    label <- label[label != "Blanc"]
    # Setup dataframe only for plotting
    tmp[[i]] = data.frame(OD = matrix(DM, ncol = 1), Sal = label, time = rep((time),
        each = 60), Strain = S[i])
}
MA = do.call(rbind, tmp)
# Add corrected salinity levels as numerical values
MA$Sal <- seq(10, 100, 10)
MA$Rep = rep(1:6, each = 10)
rm(D, DM)
as_tibble(MA)  #Inspect data
```

```
## # A tibble: 87,480 x 5
##            OD   Sal  time Strain   Rep
##         <dbl> <dbl> <int> <fct>  <int>
##  1  0.0359       10     0 331v2      1
##  2 -0.00186      20     0 331v2      1
##  3 -0.0000614    30     0 331v2      1
##  4  0.00174      40     0 331v2      1
##  5 -0.000161     50     0 331v2      1
##  6  0.00414      60     0 331v2      1
##  7  0.000439     70     0 331v2      1
##  8  0.000839     80     0 331v2      1
##  9  0.00124      90     0 331v2      1
## 10  0.00434     100     0 331v2      1
## # ... with 87,470 more rows
```

3 Growth curves and estimation of fitness parameters

Growth curves of 11 strains at 10 different salt conditions were used to estimate fitness indexes. In the case of strain S630 grown at 10 psu we decided to consider the first maximum reached after roughly 260 hours incubations time to estimate the fitness parameters. We assumed that the later increase in cell densities after extended incubation time that in some replicates surpassed those values measured at salinities with overall highest fitness estimates might have been due to the selection of genetically adapted cells that had evolved under high physiological pressure. The growth curves (Fig.S1) of some strains differed over the salinity gradient mainly in their maximal cell densities, while others differed strongly in the incubation time after which the maximal growth density was reached and therefore in their growth rates.



3.1 Maximum OD (maxOD) estimation

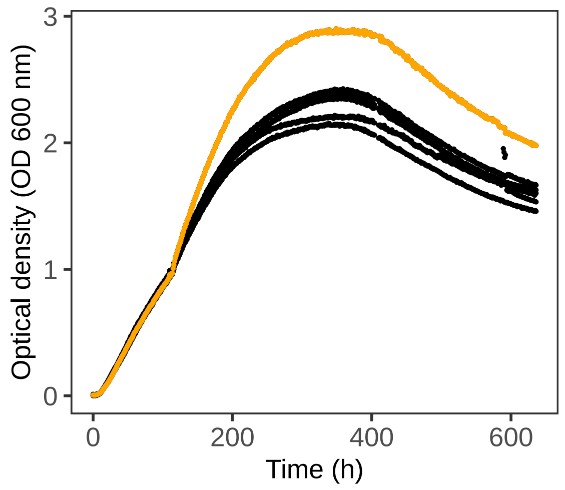
MaxOD values were extracted from each strain under all salt conditions.

```
# Exclude OD values > 263 hrs for strain 630 at 10 psu
MA$OD[which(MA$Strain == "630v4" & MA$Sal == 10 & MA$time > 263)] = NA

# Maximum OD for each replicate incubation per strain
Data_strains = aggregate(OD ~ Sal + Rep + Strain, data = MA, FUN = max)
```

```
names(Data_strains)[4] = "maxOD"  #Renaming the OD output for maximium OD
as_tibble(Data_strains)  #Inspect results
```

```
## # A tibble: 660 x 4
##      Sal   Rep Strain maxOD
##    <dbl> <int> <fct>  <dbl>
##  1    10     1 331v2   1.33
##  2    20     1 331v2   1.33
##  3    30     1 331v2   1.34
##  4    40     1 331v2   1.38
##  5    50     1 331v2   1.40
##  6    60     1 331v2   1.42
##  7    70     1 331v2   1.47
##  8    80     1 331v2   1.40
##  9    90     1 331v2   1.18
## 10   100     1 331v2   1.36
## # ... with 650 more rows
```

After visual inspection of the OD values for the strain 630, we decided to replace the replicate 6 at 100 psu (highlighted in orange) by the average of the treatment for downstream analysis (see figure below)



3.2 Growth rate (GR) estimation

GR were estimated by fitting a logistic growth curve as detailed in García et al., 2018. GR were calculated excluding the lag-phases (considered as the period until initial OD values had doubled) and until the maximal cell densities were reached. Lag phases were excluded as they were likely impacted by an acclimation phase only in those media that differed from the salinity in the pre-culture medium. Values to initiate the model fitting were chosen from preliminary fitting tests to determine values between reasonable limits.

```
# GR for each replicate incubation per strain
tmp = list()
p.phi3 = c(0.3, 0.3, 0.3, 0.3, 0.3, 0.4, 0.22, 0.2, 0.2, 0.3, 0.4)  # Values to initiate model fitting
p.phi1 = c(1.5, 1.5, 1.5, 1.5, 1.5, 1.4, 0.3, 1.5, 0.3, 1.5, 2)  # Values to initiate model fitting
for (i in 1:11) {
    fi = paste0("../data_OD/", "S", S[i], ".txt")
    D <- read.table(fi, header = F)
    label <- D[, 1]
    D <- as.matrix(D[, 2:dim(D)[2]])
    if (i == 11) {
        D[label == "S00", 264:dim(D)[2]] = NA
    }
    # Removing blanc from the OD measurements
    DM <- D[label != "Blanc", ] - mean(D[label == "Blanc", ], na.rm = T)
    label <- label[label != "Blanc"]

    # set parameters
    phi1 <- NA  # K= carrying capacity
    phi2 <- NA  # NO=Initial abundance
    phi3 <- NA  # r= growth rate
    # loop to estimate parameters
    for (j in 1:60) {
        if (mean(DM[j, ], na.rm = T) <= 0.06) {
            # OD threshold to estimate curves parameters. If the growth curves did not
            # surpass 0.060 OD, parameters were replaced by 0.01 instead of 0 to avoid
            # conflicts in the subsequent multplication of maxOD and GR (fitness
            # estimation, see chapter 4). The value 0.060 was used as threshold as it
            # represents the 10% of the max growth of S599 (the strain with lowest cell
            # densities).
            phi1[j] <- 0.01
            phi2[j] <- 0.01
            phi3[j] <- 0.01
        } else {
            start = min(which(DM[j, ] > DM[j, 1] * 2))  #Initial time (after lag-phase)
            end = max(which(DM[j, ] == max(DM[j, ], na.rm = T)))  #Final time (at maxOD)
            timei <- (start):end
            time0 <- timei - start
            M = data.frame(time = time0, gc = DM[j, timei])  #Subset of values for curve fitting
            wilson <- nls(gc ~ phi1/(1 + ((phi1 - phi2)/phi2) * exp(-(phi3 *
                time))), data = M, start = list(phi1 = p.phi1[i], phi2 = 0.027,
                phi3 = p.phi3[i]))  #Fitting logistic growth curve
            phi1[j] <- coef(wilson)[1]  #Assing K
            phi2[j] <- coef(wilson)[2]  #Assing NO
            phi3[j] <- coef(wilson)[3]  #Assing r
        }
    }
    label = label[label != "Blanc"]
    pot = data.frame(GR = matrix(phi3, ncol = 1), Sal = label, Rep = rep(1:6,
        each = 10))  #GR estimation
    tmp[[i]] = pot
}
MA = do.call(rbind, tmp)
Data_strains$GR = MA$GR

# Removing and replacing the outlier by the treatment mean
Data_strains$GR[Data_strains$Strain == "630v4" & Data_strains$Rep == 6 & Data_strains$Sal ==
    100] = NA
Data_strains$GR[Data_strains$Strain == "630v4" & Data_strains$Rep == 6 & Data_strains$Sal ==
    100] = mean(Data_strains$GR[Data_strains$Strain == "630v4" & Data_strains$Sal ==
    100], na.rm = T)
```

```
as_tibble(Data_strains)  #Inspect results
```

```
## # A tibble: 660 x 5
##      Sal   Rep Strain maxOD    GR
##    <dbl> <int> <fct>  <dbl> <dbl>
##  1    10     1 331v2   1.33 0.361
##  2    20     1 331v2   1.33 0.370
##  3    30     1 331v2   1.34 0.337
##  4    40     1 331v2   1.38 0.327
##  5    50     1 331v2   1.40 0.305
##  6    60     1 331v2   1.42 0.263
##  7    70     1 331v2   1.47 0.256
##  8    80     1 331v2   1.40 0.236
##  9    90     1 331v2   1.18 0.207
## 10   100     1 331v2   1.36 0.200
## # ... with 650 more rows
```

4 Niche breath

4.1 Select model to fit fitness curves

To integrate the different aspects of growth curves we created a combined fitness index from the product of maximum cell densities (maxOD) and growth rates (GR) for each strain in a given medium. This combined fitness index was used for all downstream analyses. Fitness curves were fitted by applying either a lognormal or a gaussian model to the fitness values along the salinity gradient. The best model for each strain was selected based on Akaike information criterion.

```
rm(list = setdiff(ls(), c("S", "Data_strains")))
AIC_values = matrix(NA, 11, 2)
Data_strains$fitness <- Data_strains$maxOD * Data_strains$GR  # multiply maxOD x GR
as_tibble(Data_strains)
```

```
## # A tibble: 660 x 6
##      Sal   Rep Strain maxOD    GR fitness
##    <dbl> <int> <fct>  <dbl> <dbl>   <dbl>
##  1    10     1 331v2   1.33 0.361   0.479
##  2    20     1 331v2   1.33 0.370   0.492
##  3    30     1 331v2   1.34 0.337   0.452
##  4    40     1 331v2   1.38 0.327   0.452
##  5    50     1 331v2   1.40 0.305   0.426
##  6    60     1 331v2   1.42 0.263   0.373
##  7    70     1 331v2   1.47 0.256   0.376
##  8    80     1 331v2   1.40 0.236   0.330
##  9    90     1 331v2   1.18 0.207   0.245
## 10   100     1 331v2   1.36 0.200   0.274
## # ... with 650 more rows
```

```
# Fitting measured fitness values to a Gaussian model, here we used an
# self-starting method (see ?getInitial)
for (i in 1:11) {
    # Subsetting data
    tmp = Data_strains[Data_strains$Strain == S[i], ]
    res <- nls(fitness ~ k * exp(-1/2 * (Sal - mu)^2/sigma^2), start = getInitial(fitness ~
        SSgauss(Sal, mu, sigma, k), data = tmp), data = tmp)
    # Saving AIC values
    AIC_values[i, 1] = AIC(res)
}
## Fitting measured fitness values to a Lognormal model. Here we manually
## introduce values to initialize the model that come from preliminary
## fitting tests to setup model parameters between reasonable ranges. Common
## parameters were defined for a and b, while c differed for the strain S599.

for (i in 1:11) {
    c = c(40, 40, 40, 40, 40, 40, 40, 40, 10, 40, 40)
    # Subsetting data
    tmp = Data_strains[Data_strains$Strain == S[i], ]
    res <- nls(fitness ~ exp(-(log10(Sal) - a)^2/(2 * b^2))/(sqrt(2 * pi) *
        b * Sal) * c, start = c(a = 2.1, b = 0.5, c = c[i]), data = tmp)
    # Saving AIC values
    AIC_values[i, 2] = AIC(res)
}
colnames(AIC_values) = c("Gaussian_model", "Lognormal_model")
row.names(AIC_values) = levels(factor(Data_strains$Strain))  #Adding Strains ID
kbl(AIC_values)  # Print Akaike Information criterion values for each model
```

	Gaussian_model	Lognormal_model
331v2	-248.3538	-237.9036
337v3	-189.2549	-254.0235
338	-192.7083	-291.2917
366v3	-242.5447	-424.1100
374	-217.5480	-290.7137
432	-279.0705	-343.7969
479v3	-318.2244	-230.9286
490v3	-185.6913	-238.8411
599v2	-409.8861	-449.0982
618v2	-196.7421	-218.2095
630v4	-395.3968	-465.8087

4.2 Fit fitness curves and estimate niche breadth (NB)

The NB of the strains was calculated after normalizing the fitness curves by dividing through the maximal modeled fitness value. The NB (in g L-1 NaCl) was defined as the salinity range covered by each strains at 50% of its maximal normalized fitness. Due to the asymmetry of the fitness curves, we additionally estimated side-dependent NBs by dividing the full NBs between the right (hyperosmotic) and left (hypoosmotic) sides relative to the modeled optimal salt concentration to obtain a side-dependent hyperosmotic and hypoosmotic NB for each strain. The model for fitting the fitness curves was selected by choosing the model with lower AIC value. The values to initialize the models were set as before.

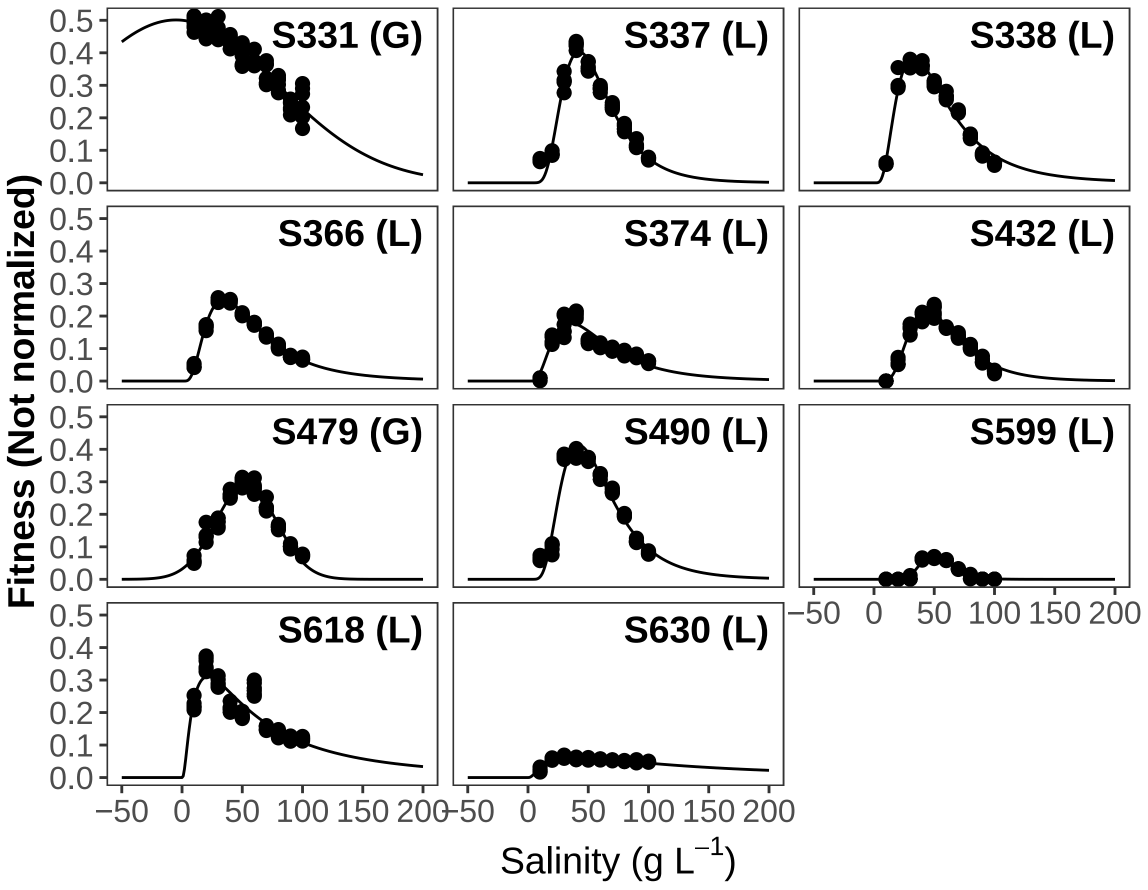
```
# Setting up vector for Niche breadth estimations
id = S
cond = c("331v2", "479v3")  #For these two strains Gaussian model fitting was chosen
```

```
c = c(40, 40, 40, 40, 40, 40, 40, 40, 10, 40, 40)
S2 = c(35, 40, 40, 45, 35, 35, 45, 45, 45, 40, 35)  #Salinity concentrations at sampling point S2
var_niche <- data.frame(strains = c(1:length(id)), niche_breadth = c(1:length(id)),
    max_fit = c(1:length(id)), sal_opt = c(1:length(id)))
Dat = list()
Fig.niche = list()
for (i in 1:11) {
    temp = subset(Data_strains, Strain == id[i])
    datx = subset(temp, temp$fitness > 0)
    datx = datx$Sal
    daty = subset(temp, temp$fitness > 0)
    daty = daty$fitness

    tmp_mod = data.frame(datx = datx, daty = daty)
    if (any(id[i] == cond) == FALSE) {
        # Lognormal model
        model <- nls(daty ~ exp(-(log10(datx) - a)^2/(2 * b^2))/(sqrt(2 * pi) *
            b * datx) * c, start = c(a = 2.1, b = 0.5, c = c[i]))
    } else {
        # Gaussiona model
        model <- nls(daty ~ k * exp(-1/2 * (datx - mu)^2/sigma^2), start = getInitial(daty ~
            SSgauss(datx, mu, sigma, k), data = tmp_mod), data = tmp_mod)
    }
    # Set the salinity range for model prediction
    salt <- seq(-50, 300, 1)
    # Fitness prediction over the chosen salinity range
    fitness <- predict(model, list(datx = salt))
    fitness[is.nan(fitness)] = 0
    # Setup dataframes for not normalized fitness values for Fig. 1
    Fig.niche[[id[i]]] = data.frame(salt = salt, fitness = fitness, Strain = id[i])
    # Model normalization by dividing through maximal predicted fitness
    fitness_norm = fitness/(max(fitness))
    # temporal dataframe
    tempniche <- data.frame(salt = salt, fitness_norm = fitness_norm)
    # Parameters NB estimated as the salinity range covered at >= 50% of maximal
    # fitness
    var_niche$niche_breadth[i] = as.numeric((max(subset(tempniche, tempniche$fitness_norm >=
        0.5)[, 1]) - min(subset(tempniche, tempniche$fitness_norm >= 0.5)[,
        1])))
    var_niche$S2[i] = S2[i]  #Salinity concentration at S2
    var_niche$neg_lim[i] = min(subset(tempniche, tempniche$fitness_norm >= 0.5)[,
        1])  #Lower limit
    var_niche$pos_lim[i] = max(subset(tempniche, tempniche$fitness_norm >= 0.5)[,
        1])  #Upper limit
    var_niche$max_fit[i] = as.numeric(max(temp$fitness))  #Maximum fitness value (not normalized fitnes
    var_niche$sal_opt[i] = as.numeric(mean(subset(tempniche, tempniche$fitness_norm ==
        1)[, 1]))  #Salinity at the fitness optimum
    var_niche$niche_hypo[i] = abs(var_niche$sal_opt[i] - var_niche$neg_lim[i])  #Hypo-osmotic NB
    var_niche$niche_hyper[i] = abs(var_niche$sal_opt[i] - var_niche$pos_lim[i])  #Hyper-osmotic NB
    var_niche$strains[i] = id[i]  #Strain ID

    # Data for Fig.1 (next chunk)
    tempniche$Strain = id[i]
    tempniche$S2 = S2[i]
    tempniche$optimal = as.numeric(var_niche$sal_opt[i])
    Dat[[id[i]]] = tempniche
}
```

strains	niche_breadth	max_fit	sal_opt	S2	neg_lim	pos_lim	niche_hypo	niche_hyper
331v2	143	0.5136630	-5	35	-50	93	45	98
337v3	48	0.4351429	42	40	25	73	17	31
338	53	0.3801369	33	40	16	69	17	36
366v3	56	0.2567095	35	45	17	73	18	38
374	58	0.2155914	35	35	17	75	18	40
432	50	0.2360250	44	35	26	76	18	32
479v3	58	0.3139364	53	45	24	82	29	29
490v3	51	0.4025219	42	45	24	75	18	33
599v2	29	0.0701338	49	45	37	66	12	17
618v2	67	0.3741855	21	40	7	74	14	53
630v4	140	0.0691249	40	35	11	151	29	111

4.3 Raw data and fitted curves

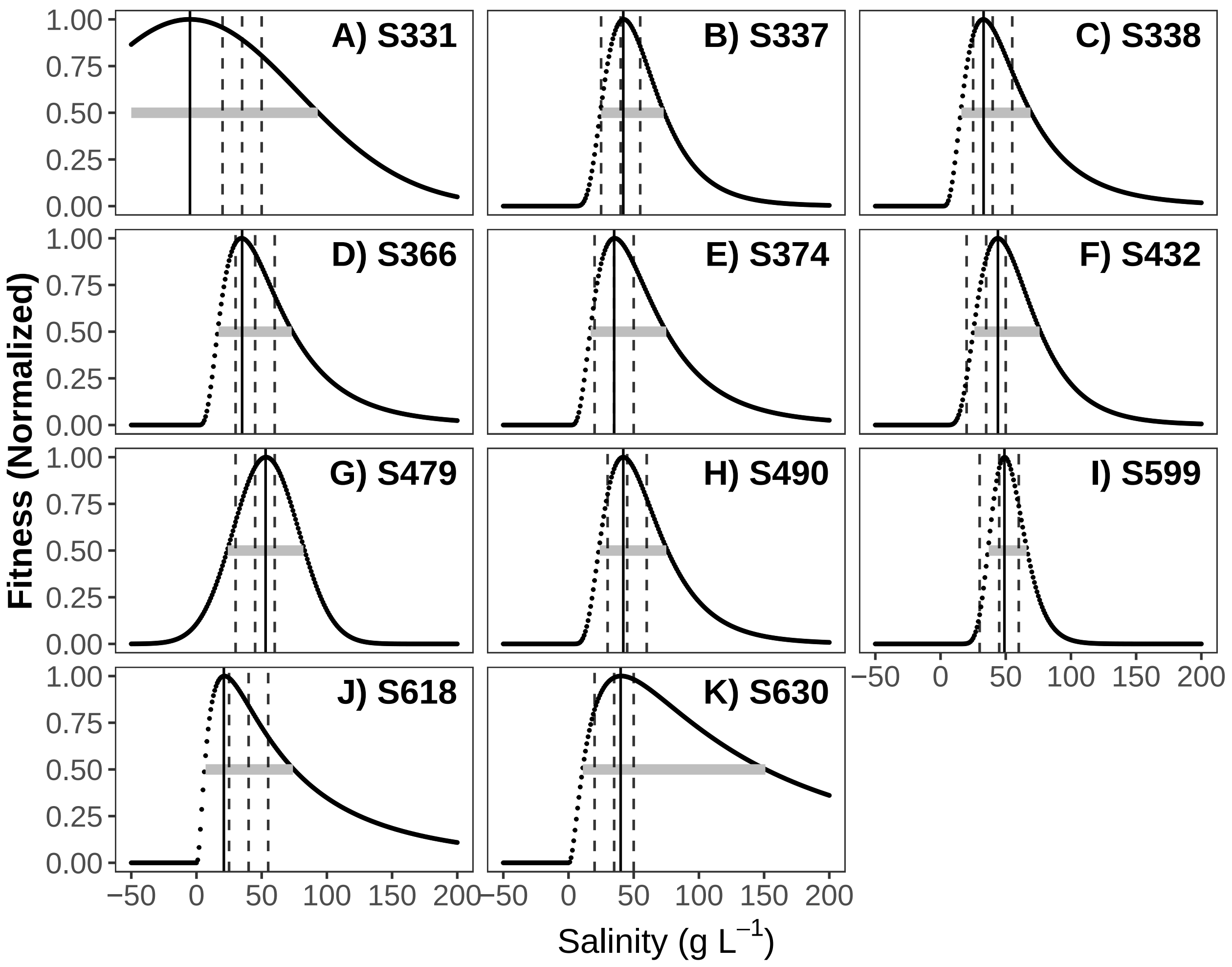
Figure representing the fitness datapoints as well as fitted fitness curves (not normalized) for the 11 strains used in this study (Fig. 2).



4.4 Fitted curves and sampling points

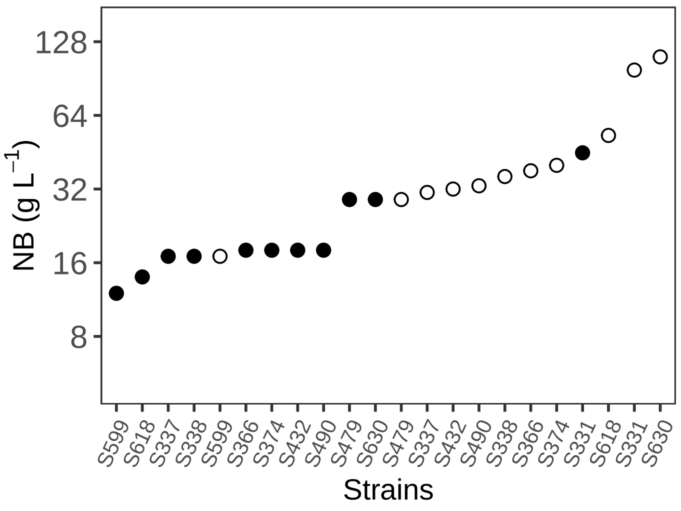
Figure with normalized fitness values. Optimum salinity, the experimental sampling points and the fitness thresholf (0.5) used for NB extimations are indicated. (Fig. 4)

```
## # A tibble: 3,861 x 6
##     salt fitness_norm Strain    S2 optimal exp_S2
##    <dbl>        <dbl> <chr>  <dbl>   <dbl>  <dbl>
##  1   -50        0.866 331v2     35      -5     35
##  2   -49        0.871 331v2     35      -5     35
##  3   -48        0.876 331v2     35      -5     35
##  4   -47        0.882 331v2     35      -5     35
##  5   -46        0.887 331v2     35      -5     35
##  6   -45        0.892 331v2     35      -5     35
##  7   -44        0.897 331v2     35      -5     35
##  8   -43        0.902 331v2     35      -5     35
##  9   -42        0.907 331v2     35      -5     35
## 10   -41        0.912 331v2     35      -5     35
## # ... with 3,851 more rows
```



4.5 Ranking of hypo- and hyper-osmotic NBs

Fig. 5 displays ranked side-dependent NBs



4.6 Filtering side-dependent NBs for downstream statistics

In the downstream statistical analyses we used hyper and hypoosmotic NBs, and only considered side-dependent NBs if the salinity optimum of the respective strain was not passed by more the 4.5 g L-1 NaCl (equivalent to 30% of the total salinity change). With this filtering rule, maximal two values for gene regulation per strain and gene were considered

```r
var_niche$S1 = var_niche$S2 - 15  # Salinity concentration at S1
var_niche = var_niche[, c(1, 2, 3, 4, 6, 7, 8, 9, 10, 5)]  # Reordering level S2 column
var_niche$S3 = var_niche$S2 + 15  # Salinity concentration at S3
var_niche$neighbor_opt = NA  #  For closest neighbor

# 1) Identify the sampling point (S1, S2 or S3) that is closest neighbor to
# the maximal modeled fitness

# The identified closest sampling points are represented by numerical
# indexes for the next loop: S1=1;S2=2;S3=3
for (i in 1:11) {
    var_niche$neighbor_opt[i] = which(abs(var_niche[i, 4] - var_niche[i, 9:11]) ==
        min(abs(var_niche[i, 4] - var_niche[i, 9:11])))
}

# 2) Detect how close is the above selected neighboring sampling point to
# the modeled optimum (30% limit)

# Here the previous index is used to find the position of the closest
# sampling points to calculate the difference to the modelled optimum
# salinity
var_niche$diff_opt = NA  #could you change to diff_opt
for (i in 1:11) {
    # get difference with -/+
    var_niche$diff_opt[i] = (var_niche[i, 4] - var_niche[i, 8 + var_niche$neighbor_opt[i]])
}
```

```r
# 3) Create a filter based on the position of the sampled salinity range
# relative to the modeled optimum

limit = 15 * 0.3
filter <- function(i) {
    # if the sample point S1 is closest to the modeled optimum, either both
    # considered salinity ranger are indeed on the hyperosmostic side, of if the
    # limit is passed only S2:S3 is considered
    if (var_niche$neighbor_opt[i] == 1 & (var_niche$diff_opt[i]) <= limit) {
        c("hyper", "hyper")

    } else if (var_niche$neighbor_opt[i] == 1 & (var_niche$diff_opt[i]) > limit) {
        c("NA", "hyper")
        # if the sample point S2 is closest to the modeled optimum, the considered
        # salinty ranges are placed on the hypo and hyperosmotic side. However, if
        # the limit is passed ether only the hypo- or the hyperosmotic side were
        # considered for downstream statistics
    } else if (var_niche$neighbor_opt[i] == 2 & abs(var_niche$diff_opt[i]) <=
        limit) {
        c("hypo", "hyper")

    } else if (var_niche$neighbor_opt[i] == 2 & (var_niche$diff_opt[i]) < (-limit)) {
        c("NA", "hyper")

    } else if (var_niche$neighbor_opt[i] == 2 & (var_niche$diff_opt[i]) > (limit)) {
        c("hypo", "NA")
        # if the sample point S3 is closest to the modeled optimum, either both
        # considered salinity ranger are indeed on the hypoosmostic side, or if the
        # limit is passed only S2:S1 is considered
    } else if (var_niche$neighbor_opt[i] == 3 & (var_niche$diff_opt[i]) >= (-limit)) {
        c("hypo", "hypo")

    } else {
        c("hypo", "NA")
    }
}
var_niche$filter1 <- sapply(seq(1, 11), filter)[1, ]
var_niche$filter2 <- sapply(seq(1, 11), filter)[2, ]

# 4) Apply the filter to the side dependent NBs for the salinity range S2:S1
# and S2:S3
var_niche$niche_S2S1_filt = ifelse(var_niche$filter1 == "hypo", var_niche$niche_hypo,
    var_niche$niche_hyper)
var_niche$niche_S2S1_filt = ifelse(var_niche$filter1 == "NA", "NA", var_niche$niche_S2S1_filt)
var_niche$niche_S2S3_filt = ifelse(var_niche$filter2 == "hypo", var_niche$niche_hypo,
    var_niche$niche_hyper)
var_niche$niche_S2S3_filt = ifelse(var_niche$filter2 == "NA", "NA", var_niche$niche_S2S3_filt)
```

4.7 Stress exposure

Stress exposure was calculated by building the difference of normalized fitness values between the sampled salinity levels S2:S1,S2:S3 and S1:S3.

```
Dat2 = do.call(rbind, Dat)
Dat2$Strain = factor(Dat2$Strain)
tmp = matrix(NA, 11, 3)
for (i in 1:11) {
    strain = levels(Dat2$Strain)[i]
    t0 = Dat2[Dat2$Strain == strain, ]
    fitness_S1 = t0$fitness_norm[t0$salt == (t0$S2 - 15)]  #Fitness at S1
    fitness_S2 = t0$fitness_norm[t0$salt == (t0$S2)]  #Fitness at S2
    fitness_S3 = t0$fitness_norm[t0$salt == (t0$S2 + 15)]  #Fitness at S3
    tmp[i, ] = c(fitness_S1, fitness_S2, fitness_S3)
}
tmp1 = matrix(NA, 11, 3)
# Subtracting fitness values to obtain parameter for delta fitness (deltaf)
tmp1[, 1] = tmp[, 2] - tmp[, 1]
tmp1[, 2] = tmp[, 2] - tmp[, 3]
tmp1[, 3] = tmp[, 1] - tmp[, 3]
colnames(tmp1) = c("deltaf_S2S1", "deltaf_S2S3", "deltaf_S1S3")
```

4.8 Estimation of weighted NBs

In order to visualize the correlation between NB values (side-dependent NBs as well as full NBs) and stress exposure (Fig. 4, see chapter 4.10), while also including salinity ranges, where the salinity optimum was crossed and that therefore refer partly to the hyper- and hypoosmotic NB the proportional contribution of the hypo- or hyperosmotic NB that was covered by the respectively considered salinity gradient was estimated.

```
tmp = NULL
for (i in 1:11) {
    if (var_niche$sal_opt[i] - (var_niche$S2[i]) < (-15)) {
        tmp[i] = var_niche$niche_hyper[i]
    } else if (var_niche$sal_opt[i] - var_niche$S2[i] >= (-15) & var_niche$sal_opt[i] -
        var_niche$S2[i] < 0) {
        x = abs(var_niche$sal_opt[i] - var_niche$S2[i])/15
        tmp[i] = (var_niche$niche_hyper[i] * x) + (var_niche$niche_hypo[i]) *
            (1 - x)
    } else if (var_niche$sal_opt[i] - (var_niche$S2[i]) >= 0) {
        tmp[i] = var_niche$niche_hypo[i]
    }
}
fractioned_niche = data.frame(niche_S2S1_w = tmp)  # niche_S2S1_w as the corresponding fractioned NB
tmp = NULL
for (i in 1:11) {
    if (var_niche$sal_opt[i] - (var_niche$S2[i]) > (15)) {
        tmp[i] = var_niche$niche_hypo[i]
    } else if (var_niche$sal_opt[i] - var_niche$S2[i] > (0) & var_niche$sal_opt[i] -
        var_niche$S2[i] < 15) {
        x = (abs(var_niche$sal_opt[i] - var_niche$S2[i])/15)
        tmp[i] = var_niche$niche_hypo[i] * x + (var_niche$niche_hyper[i]) *
            (1 - x)
    } else if (var_niche$sal_opt[i] - (var_niche$S2[i]) <= 0) {
        tmp[i] = var_niche$niche_hyper[i]
    }
}
fractioned_niche$niche_S2S3_w = tmp  # niche_S2S3_w fractioned NB
```

4.9 Table with NB and fitness estimations to be used for MLMs (Rscript_03)

```
Niche_table = data.frame(strains.ID = rep(levels(factor(var_niche$strains)),
    3))
Niche_table$direction = rep(c("S2:S1", "S2:S3", "S1:S3"), each = 11)
Niche_table$n_breadth = NA
Niche_table$n_breadth = c(var_niche$niche_hypo, var_niche$niche_hyper, var_niche$niche_breadth)
Niche_table$n_breadth_filtered = NA
Niche_table$n_breadth_filtered[1:22] = c(var_niche$niche_S2S1_filt, var_niche$niche_S2S3_filt)
Niche_table$n_breadth_weighted = NA
Niche_table$n_breadth_weighted[1:22] = c(fractioned_niche$niche_S2S1_w, fractioned_niche$niche_S2S3_w)
Niche_table$delta_fitness = NA
Niche_table$delta_fitness = c(tmp1[, 1], tmp1[, 2], tmp1[, 3])
Niche_table$n_breadth_filtered = as.numeric(Niche_table$n_breadth_filtered)
# Exporting table for MLM analyses
write.table(file = "../niche-performance.txt", sep = "\t", Niche_table)
as_tibble(Niche_table)  #Inspect results
```

```
## # A tibble: 33 x 6
##    strains.ID direction n_breadth n_breadth_filtered n_breadth_weighted
##    <fct>      <chr>         <dbl>              <dbl>              <dbl>
##  1 S331       S2:S1            45                 98               98
##  2 S337       S2:S1            17                 17               17
##  3 S338       S2:S1            17                 NA               25.9
##  4 S366       S2:S1            18                 NA               31.3
##  5 S374       S2:S1            18                 18               18
##  6 S432       S2:S1            18                 18               18
##  7 S479       S2:S1            29                 29               29
##  8 S490       S2:S1            18                 18               21
##  9 S599       S2:S1            12                 12               12
## 10 S618       S2:S1            14                 53               53
## # ... with 23 more rows, and 1 more variable: delta_fitness <dbl>
```

4.10 Regression of NB versus stress

Stress indicates the stress exposure between two salinity concentration and was defined as the absolute value of the difference in fitness between this two salinity concentrations (delta fitness). NBs are negatively correlated to stress if salinity ranges that are used to calculate stress exposure are symmetrically positioned around the modeled salinity optimum of the respective strain (i.e. considering correlation between filtered side-dependent NBs and the respective stress values). This relationship is weakened if including all sampled salinity ranges from our incubation experiment. In order to account for the fact that the considered salinity ranges cross the modeled salinity optima and therefore cover the hypo and hyperosmotic NB at different proportions stress values were plotted against NBs that were weighted by the proportions of the covered hypo and hyperosmitic side (Fig.3).

```
# Formating dataframe delta fitness to stress
Niche_table$stress <- abs(Niche_table$delta_fitness)

# 1) Regression of filtered datapoints
```

```r
Niche_table$stress_filtered <- ifelse(Niche_table$n_breadth_filtered == "NA",
    "NA", Niche_table$stress)

# Fit regression to a exponential decay model, starting points were derived
# from exploratory analysis
fit1 = nls(n_breadth_filtered ~ a * exp(-S * stress_filtered), data = Niche_table,
    start = list(a = 52, S = 1.6))
summary(fit1)
```

```
##
## Formula: n_breadth_filtered ~ a * exp(-S * stress_filtered)
##
## Parameters:
##   Estimate Std. Error t value Pr(>|t|)
## a  163.236     30.871   5.288 9.12e-05 ***
## S    8.184      1.422   5.756 3.80e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.92 on 15 degrees of freedom
##
## Number of iterations to convergence: 10
## Achieved convergence tolerance: 5.817e-06
##   (16 observations deleted due to missingness)
```

```r
# Calculate the determination coefficient (R2)
1 - (sum(residuals(fit1)^2)/((sum(complete.cases(Niche_table$n_breadth_filtered)) -
    1) * var(Niche_table$n_breadth_filtered, na.rm = T)))
```

```
## [1] 0.7270299
```

```r
# Create data points for plot
v <- seq(0, 1, 0.01)
DF2 = data.frame(x = v, delta = predict(fit1, list(stress_filtered = v)))

# 2) Regression of all datapoints

# Create a new column for all the nb values
Niche_table$n_breadth_all = Niche_table$n_breadth_filtered

# Identification of gaps in niche breadth
px = which(is.na(Niche_table$n_breadth_all))

# Fill the gaps with the weigthed niche breadth, case od S2:S1 or S2:S3
Niche_table$n_breadth_all[px] = Niche_table$n_breadth_weighted[px]

# Calculate the average NB for gaps in case of S1:S3
tmp0 = aggregate(n_breadth_weighted ~ strains.ID, data = Niche_table[Niche_table$direction !=
    "S1:S3", ], FUN = mean)

Niche_table$n_breadth_all[is.na(Niche_table$n_breadth_all)] = tmp0$n_breadth_weighted
```

```
# Fit regression to a exponential decay model
fit2 = nls(n_breadth_all ~ a * exp(-S * stress), data = Niche_table, start = list(a = 52,
    S = 1.6))
1 - (sum(residuals(fit2)^2)/((33 - 1) * var(Niche_table$n_breadth_all)))
```

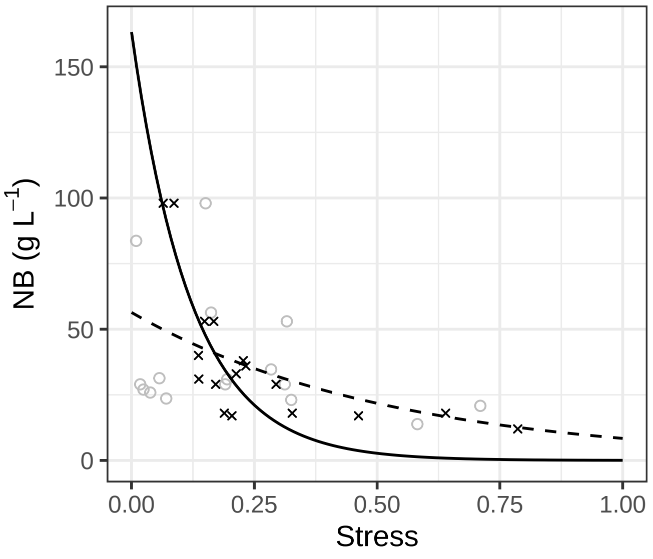
```
## [1] 0.2278207
```

```
summary(fit2)
```

```
##
## Formula: n_breadth_all ~ a * exp(-S * stress)
##
## Parameters:
##   Estimate Std. Error t value Pr(>|t|)
## a  56.3826     8.5004   6.633 2.05e-07 ***
## S   1.9040     0.7852   2.425   0.0213 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.65 on 31 degrees of freedom
##
## Number of iterations to convergence: 3
## Achieved convergence tolerance: 4.839e-06
```

```
# Create data points for plot
v <- seq(0, 1, 0.01)
DF3 = data.frame(x = v, delta = predict(fit2, list(stress = v)))
```

Fig. 3 displays the regression between NB and Stress estimations. The black cross symbols (and solid trend line: R2=0.73, P<0.001) represent stress data points from either the hyper- or hypoosmotic side of the strain's fitness curve without crossing the optimum and the corresponding side dependent NBs that were used downstream for mixed model regression against the NB (see Rscript3_Mixed_models). Gray circles represent stress levels between salinity concentrations where the optimum was crossed. The corresponding NB values were estimated from the proportional contribution of the left or right sided NB that was covered by the respectively considered salinity gradient. The gray trend line was fitted by including all data points (R2=0.23, P=0.021). A pronounced negative correlation was observed for the filtered dataset, while the correlation was less pronounced when all the data were included.



4.11 Moments

Moments (mean, var, skewness, kurtosis) of the modeled fitness curves for 11 strains used in study.

```
Fig.niche = Fig.niche[Fig.niche$salt >= 0, ]
Table = matrix(NA, 11, 4)
for (i in 1:length(levels(Fig.niche$Strain))) {
    x = rep(Fig.niche$salt[Fig.niche$Strain == levels(Fig.niche$Strain)[i]],
        round(Fig.niche$fitness[Fig.niche$Strain == levels(Fig.niche$Strain)[i]] *
            100))
    Table[i, 1] = mean(x)
    Table[i, 2] = sd(x)
    Table[i, 3] = skewness(x)
    Table[i, 4] = kurtosis(x)
}
colnames(Table) = c("Mean", "SD", "Skewness", "Kurtosis")
rownames(Table) = levels(var_niche$strains)
Table[, 1:4] = round(Table[, 1:4], 3)
kbl(Table)  #Inspect results
```

	Mean	SD	Skewness	Kurtosis
S331	64.210	48.783	0.930	0.488
S337	58.135	27.183	1.142	1.485
S338	58.577	37.255	1.442	2.315
S366	61.584	38.411	1.320	1.708
S374	62.084	37.614	1.211	1.253
S432	60.239	27.818	1.016	0.934
S479	54.266	23.879	0.146	-0.348
S490	60.565	30.550	1.215	1.686
S599	53.693	12.921	0.485	-0.214
S618	80.863	66.719	1.233	0.879
S630	115.158	75.424	0.609	-0.658