Intro to R: Assignment 4

Abhilesh Dhawanjewar

February 8 th 2017

Citation:

Lee-Yaw JA, Jacobs CGC, Irwin DE (2014) **Individual performance in relation to cytonuclear discordance in a northern contact zone between long-toed salamander (*Ambystoma macrodactylum*) lineages.** Molecular Ecology 23(18): 4590-4602. http://dx.doi.org/10.1111/mec.12878

Data Repository:

Lee-Yaw JA, Jacobs CGC, Irwin DE (2014) Data from: **Individual performance in relation to cytonuclear discordance in a northern contact zone between long-toed salamander (*Ambystoma macrodactylum*) lineages.** Dryad Digital Repository. http://dx.doi.org/10.5061/dryad.q8473

Synopsis:

When individuals from different species or populations come in contact and hybridize, the patterns of gene flow elucidated by the nuclear vs the mitochondrial markers are often found to be discordant. Two lineages of the long-toed salamander (*Ambystoma macrodactylum*) in Western Canada demonstrate a potential case of such cytonuclear discordance. In this present study, Lee-Yaw et. al. aim to map the extent of this cytonuclear discordance using additional genetic markers (AFLP and SNP data) and samples. They further aim to specifically investigate two hypotheses that could explain the observed patterns, which are adaptive introgression of mtDNA or a neutral wake of mtDNA left behind following hybrid zone movement. Feeding performance of individuals in a common environment was also assayed to test for associations between mitotypes and nuclear backgrounds suggesting coevolution of mito-nuclear haplotypes. Their results confirm a general pattern of cytonuclear discordance with limited introgression of a diagnostic nuclear marker. The lowest performance in the feeding performance assays were of individuals with the greatest degree of mismatch between the nuclear background and mitotype suggesting mito-nuclear coevolution at play.

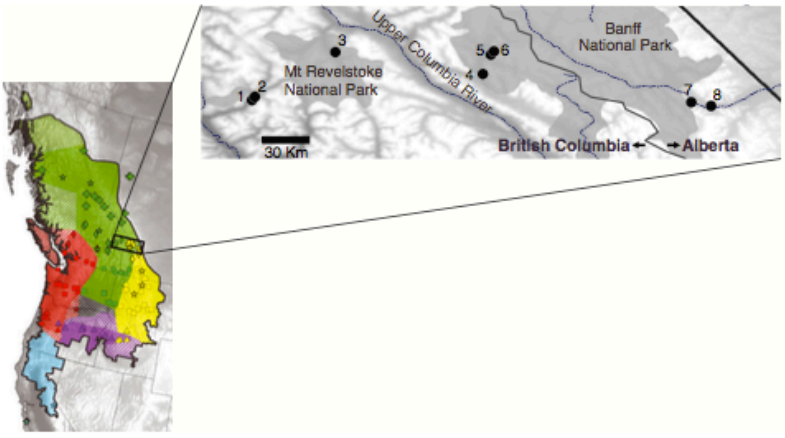

Fig. 1 Breeding ponds from which adult long-toed salamanders were collected and position of sampling with respect to the distribution of the major genetic lineages identified by Lee-Yaw & Irwin (2012). The boundaries between lineages in that study were determined by AFLPs (groups denoted by different coloured shading) and mtDNA (groups denoted by different coloured symbols).

We will be recreating Fig 3. from the publication:

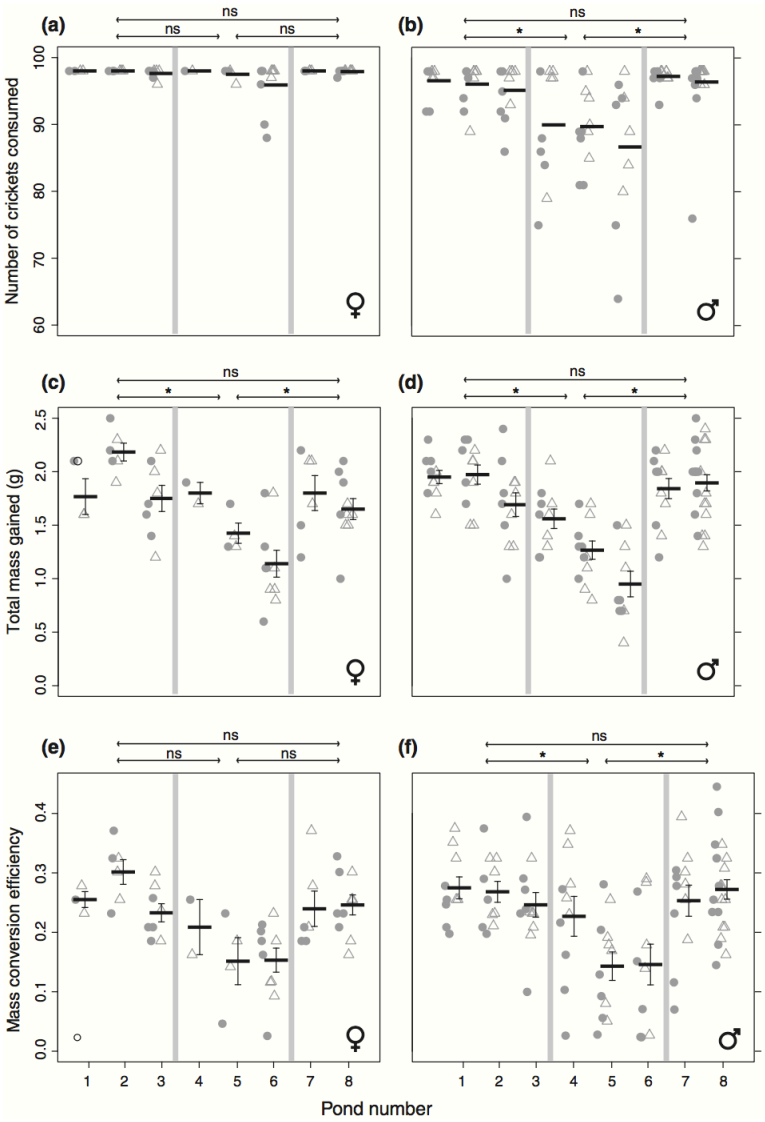

Fig. 3 Long-toed salamander performance in terms of food intake (top panels), total mass gained during the summer (middle panels) and mass conversion efficiency (bottom panels) for females (left panels) and males (right panels) across a northern contact zone between NC and RM. Overall pond means are shown as thick black bars (with standard errors in the four bottom panels). Grey symbols denote the values for individuals with symbol shape indicating temperature treatment (circles = 19 °C, triangles = 25 °C). The one open circle in Pond 1 in panels c and e represents a single female who was an outlier with respect to growth trajectory (the pond means shown do not include this individual but the inclusion or exclusion of this individual did not qualitatively influence our results). Significant differences ($\alpha = 0.05$) based on Tukey's tests between groups of ponds with similar levels of introgression (separated by thick lines) are indicated with an asterisk above the plots.

For easier manipulation of the data, we will be using the `dplyr` library (https://github.com/hadley/dplyr). This piece of code checks if `dplyr` is installed, if not, it installs it before loading.

```
# Install if required packages not installed and load them
if (!require(dplyr)) {
    install.packages("dplyr")
    library(dplyr)
} else {
    library(dplyr)
}
```

The data:

Reading in the data from the csv file "Feeding_and_SNP_data.csv" and store it in the variable 'data'

```
# Read in the data from the csv file
data = read.csv("Lee-Yaw_Jacobs_Irwin_MEC2014/Feeding_and_SNP_data.csv",
    header = TRUE)
```

Description of the data:

The 'data frame' includes data from the feeding performance assays for 149 individuals from 3 different sites over a period of 7 weeks.

- **Individual** - Identifier for the individual salamanders
- **Site** - Pond where the individual was collected
- **Sex** - Sex of the individual
- **mitotype** - The mitochondrial haplotype for the individual
- **G2116_num_RM_alleles** - Number of diagnostic G2116 SNPs
- **HOX_num_RM_alleles** - Number of dignostic HOX SNPs
- **Co1_num_RM_alleles** - Number of diagnostic COI SNPs
- **Final_Genetic_Group** - Final genetic group assigned to the individual
- **Temperature_Treatment** - The temperature at which the feeding assays were conducted
- **Week** - Week of the assay
- **Mass.g** - Mass of the individual recorded in grams
- **MassChangeSinceWeek0** - Running Total of Mass change in individual since Week 0
- **Total_Crickets_Eaten** - Total number of crickets eaten by the individual
- **NumCrickets.AveCricketMass** - Average mass of the total crickets eaten
- **SevenWeekGain** - Mass Gained over a period of 7 weeks
- **MCE** - Mean Conversion Efficiency

Data Format:

```
str(data)
```

```
## 'data.frame':    1192 obs. of  16 variables:
## $ Individual           : Factor w/ 149 levels "BV10-171","BV10-172",..: 136 136 136 136 136 136 136 136 13
##  $ Site                      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Sex                       : Factor w/ 2 levels "F","M": 2 2 2 2 2 2 2 2 2 2 ...
##  $ mitotype                  : Factor w/ 2 levels "NC","RM": 1 1 1 1 1 1 1 1 1 1 ...
##  $ G2116_num_RM_alleles      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ HOX_num_RM_alleles        : Factor w/ 3 levels "0","1","na": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Col_num_RM_alleles       : Factor w/ 4 levels "0","1","2","na": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Final_Genetic_Group       : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Temperature_Treatment     : int  25 25 25 25 25 25 25 25 25 25 ...
##  $ Week                      : int  0 1 2 3 4 5 6 7 0 1 ...
##  $ Mass.g.                   : num  2.5 2.9 3.2 3.4 3.5 4.1 4.3 4.4 3.7 4.1 ...
##  $ MassChangeSinceWeek0      : num  0 0.4 0.7 0.9 1 1.6 1.8 1.9 0 0.4 ...
##  $ Total_Crickets_Eaten      : int  97 97 97 97 97 97 97 97 97 97 ...
##  $ NumCrickets.AveCricketMass: num  4.27 4.27 4.27 4.27 4.27 ...
##  $ SevenWeekGain             : num  1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.6 1.6 ...
##  $ MCE                       : num  0.351 0.351 0.351 0.351 0.351 ...
```

The Code:

1. Subsetting and Filtering the data:

```
# Subsetting data to get unique rows
data_trun = subset(data, Week == 7)

# Filter out outlier
outlier = filter(data_trun, Site == 1 & MassChangeSinceWeek0 > 2 &
    MCE < 0.03)

# Remove outlier from plotting data
data_trun = anti_join(data_trun, outlier, by = "Individual")

# Subsetting data into females and males
data_f = subset(data_trun, Sex == "F")
data_m = subset(data_trun, Sex == "M")
```

2. Computing Summary Stats

Defining functions for standard error and applying a function on a specific subset of data

```
# Function to calculate standard error
std_err = function(x) {
    sqrt(var(x)/length(x))
}

# Function to compute summary statistics for specific columns
# grouped by Sex and Site
summ_stats = function(vars, func) {
    lapply(list(F = data_f, M = data_m), function(j) apply(j[, vars],
        2, function(i) tapply(i, j[, "Site"], func)))
}
```

Computing Mean and Standard Error

```
# Calculate standard error for variables grouped by sex and site
# number
se_results = summ_stats(c("MassChangeSinceWeek0", "MCE"), std_err)

# Calculate mean for variables grouped by sex and site number
mean_results = summ_stats(c("Total_Crickets_Eaten", "MassChangeSinceWeek0",
    "MCE"), mean)
```

3. Defining the plotting subroutines

The main plotting subroutine:

```
# Main routine for plotting each subfigure
plot_figure = function(y, sex, pos, ...) {

    # Parameters for the left-sided panels (the female ones)
    if (sex == "F") {
        dat = data_f
        unicd_sym = " "
        if (pos %in% c(5, 6)) {
            par(mar = c(4, 4, 2, 2))
            flag_yaxt = "s"
        } else {
            par(mar = c(2, 4, 2, 2))
            flag_yaxt = "s"
        }
        # Parameters for the right-sided panels (the male ones)
    } else {
        dat = data_m
        unicd_sym = " "
        if (pos %in% c(5, 6)) {
            par(mar = c(4, 1.2, 2, 4))
            flag_yaxt = "n"
        } else {
            par(mar = c(2, 1.2, 2, 4))
            flag_yaxt = "n"
        }
    }
    # Main plot layer, x-values jittered to avoid overlap
    plot(jitter(dat[, "Site"]), dat[, y], pch = c(16, 2)[as.numeric(as.factor(dat[,
        "Temperature_Treatment"]))], col = "grey", xaxt = "n", yaxt = flag_yaxt,
        xlab = "", xlim = c(0, 9), ...)
    # Add the right-side y-axis ticks for male panels
    if (sex == "M") {
        axis(4, labels = FALSE)
    }
    # Add common x-axis
    if (pos %in% c(5, 6)) {
        axis(1, at = seq(1, 8, by = 1))
    }
    # Add lines geographically demarcating the sites
    abline(v = 3.5, col = "grey", lwd = 3)
    abline(v = 6.5, col = "grey", lwd = 3)
    # Add sub_labels to all panels a-f
    mtext(paste("(", letters[pos], ")"), side = 3, adj = 0, line = 1.1,
        font = 2)
    # Add venus/mars symbol bottom-right corner
    mtext(unicd_sym, side = 1, adj = 0.96, line = -2.2, font = 2,
        cex = 1.8)
}
```

Subroutine to add significance bars:

```
# Routine to add significance bars outside plotting area
add_sig_segments = function(y1, y2, d) {

    x1 = 2
    x2 = 4.5
    x3 = 5
    x4 = 7.5

    # Outer segment
    segments(x1, y1, x4, y1)
    segments(x1, y1 - d, x1, y1 + d)
    segments(x4, y1 - d, x4, y1 + d)
    # Inner segment 1
    segments(x1, y2, x2, y2)
    segments(x1, y2 - d, x1, y2 + d)
    segments(x2, y2 - d, x2, y2 + d)
    # Inner segment 2
    segments(x3, y2, x4, y2)
    segments(x3, y2 - d, x3, y2 + d)
    segments(x4, y2 - d, x4, y2 + d)
}
```

Subroutine to add significance labels:

```
# Routine to add significance labels outside plotting area
add_sig_labels = function(lab_list) {
    mtext(lab_list[1], side = 3, adj = 0.525, line = 2, cex = 0.7)
    mtext(lab_list[2], side = 3, adj = 0.375, line = 1.03, cex = 0.7)
    mtext(lab_list[3], side = 3, adj = 0.685, line = 1.03, cex = 0.7)

}
```

4. Plotting the graph to a PDF

```
# Open the pdf device (cairo_pdf to enable writing of unicode
# characters)
cairo_pdf("Lee-Yaw_et_al_Fig_3.pdf", 8.5, 11, family = "ArialUnicodeMS")

# Split the graph into 6 panels
layout(matrix(seq(1, 6, 1), nrow = 3, ncol = 2, byrow = TRUE))

# Define the outer margins of the plot
par(oma = c(3, 2, 1.3, 2))

# Define parameters for the plotting routine
y_dat = rep(c("Total_Crickets_Eaten", "MassChangeSinceWeek0", "MCE"),
    each = 2)
sex_dat = rep(c("F", "M"), times = 3)
y_lims = rep(list(c(60, 101), c(0, 2.5), c(0, 0.45)), each = 2)
y_labs = c("Number of crickets consumed", "", "Total mass gained(g)",
    "", "Mass conversion efficiency", "")
sig_lab_list = list(c("ns", "ns", "ns"), c("ns", " ", " "), c("ns",
    " ", " "), c("ns", " ", " "), c("ns", "ns", "ns"), c("ns",
    " ", " "))
seg_coords = rep(list(c(106.6, 104.4, 0.3), c(2.85, 2.71, 0.02), c(0.515,
    0.49, 0.004)), each = 2)

# Plot the six subfigures
for (i in seq(1:6)) {

    plot_figure(y_dat[i], sex_dat[i], i, ylim = y_lims[i][[1]], ylab = y_labs[i])

    # Plot the outlier point
    if (i == 3) {
        points(outlier[1, ]$Site, outlier[1, ]$MassChangeSinceWeek0,
            pch = 1, col = "grey")
    }
    if (i == 5) {
        points(outlier[1, ]$Site, outlier[1, ]$MCE, pch = 1, col = "grey")
    }

    # Add the horizontal black bars for mean values
    for (j in seq(1:8)) {
        x_mean_low = j - 0.25
        x_mean_high = j + 0.25
        y_mean = mean_results[[sex_dat[i]]][j, y_dat[i]]
        segments(x_mean_low, y_mean, x1 = x_mean_high, lwd = 3)

        # Add standard error arrows for panels c-f
        if (i %in% seq(3, 6)) {
            se = se_results[[sex_dat[i]]][j, y_dat[i]]
            arrows(j, y_mean + se, j, y_mean - se, code = 3, angle = 90,
                length = 0.05)
        }
    }
```

```
    # Set the Graphical parameters to allow plotting outside the
    # subfigure
    par(xpd = NA)

    # Add line segments for statistical significane levels on the
    # subfigures
    add_sig_segments(seg_coords[i][[1]][[1]], seg_coords[i][[1]][[2]],
        seg_coords[i][[1]][[3]])

    # Add labels for statistical significane on the subfigures
    add_sig_labels(sig_lab_list[i][[1]])

    # Set the Graphical parameters to default values
    par(mar = c(5, 4, 4, 2), xpd = FALSE)
}

# Label the x-axis
mtext(text = "Pond number", side = 1, line = -1, outer = TRUE, cex = 0.7)

# Close the plotting device
dev.off()
```

And, finally the recreated plot (Ta-Da!):

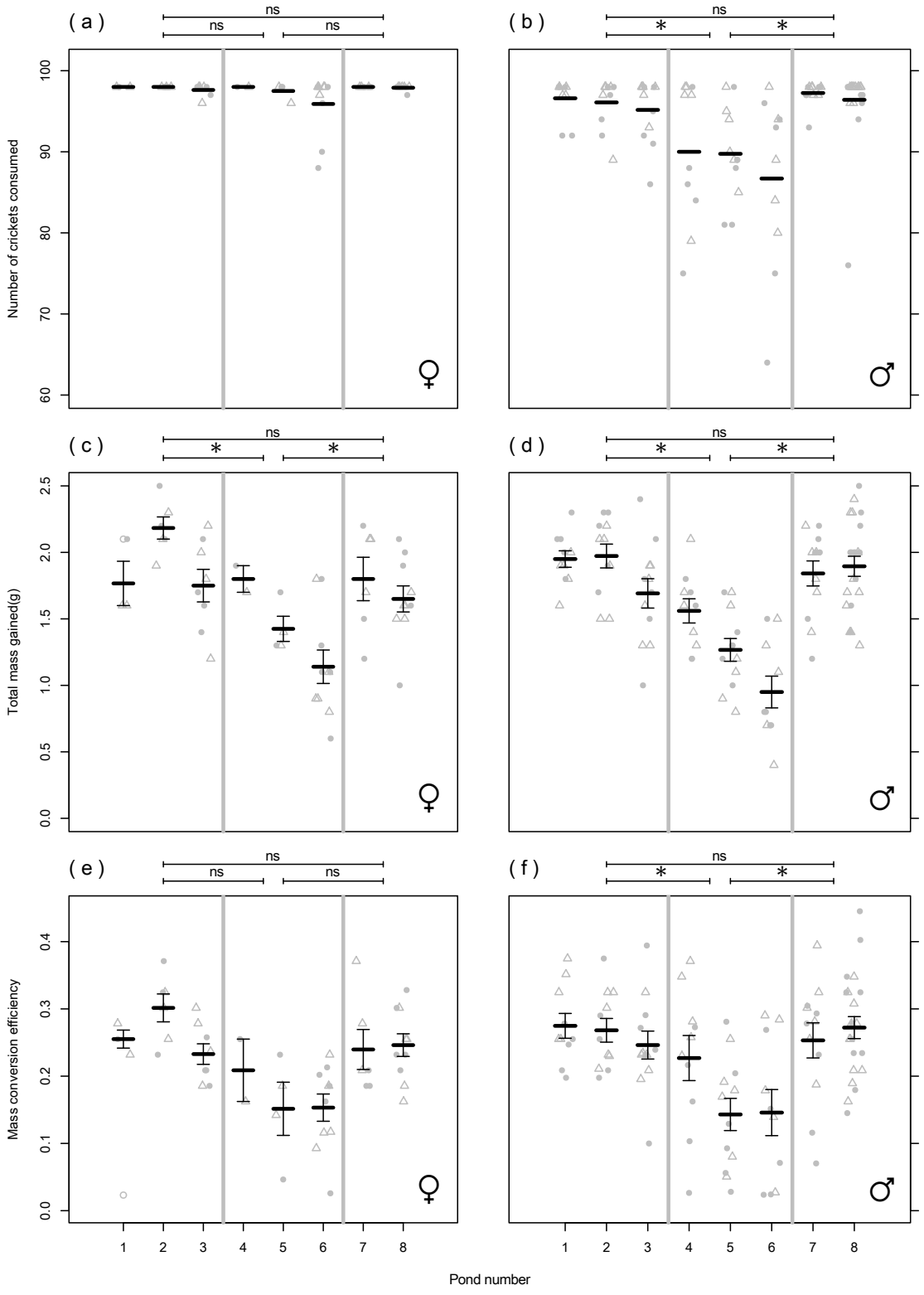

Figure 1: Recreated Fig 3. from Lee-Yaw et. al.