Kasdin et al. (2025) show that dopamine in the brains of young zebra finches acts as a learning signal, increasing when they sing closer to their adult song and decreasing when they sing further away, effectively guiding their vocal development through trial-and-error. This suggests that complex natural behaviors, like learning to sing, are shaped by dopamine-driven reinforcement learning, similar to how artificial intelligence learns. You can find the paper at this link: https://www.nature.com/articles/s41586-025-08729-1..

Note they measure dopamine using fibre photometry, changes in the fluorescence indicate dopamine changes in realtime. Their specific measurement considers changes in flourescence in 100-ms windows between 200 and 300 ms from the start of singing, averaged across development.

1. Using the `pwr` package for `R` (Champely, 2020), conduct a power analysis. How many observations would the researchers need to detect a moderate-to-large effect ($d = 0.65$) when using $\alpha = 0.05$ and default power (0.80) for a two-sided one sample t test.

```
#Power analysis
pwr.t.test(d=0.65,
           sig.level=0.05,
           type="one.sample",
           alternative = "two.sided",
           power=0.8)


##
##      One-sample t test power calculation
##
##               n = 20.58039
##               d = 0.65
##       sig.level = 0.05
##           power = 0.8
##     alternative = two.sided


#n=20.58 - need at least 21 observations
```

2. Click the link to go to the paper. Find the source data for Figure 2. Download the Excel file. Describe what you needed to do to collect the data for Figure 2(g). Note that you only need the `closer_vals` and `further_vals`. Ensure to `mutate()` the data to get a difference (e.g., `closer_vals - further_vals`).

```
#Pulling data from figure 2
fig2_further = read_csv("further_data.csv", col_names = "Further")


## Rows:  25 Columns:  1
## -- Column specification ---------------------------------------------------------
## Delimiter:  ","
## dbl (1):  Further
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.


fig2_closer = read_csv("closer_data.csv", col_names = "Closer")


## Rows:  25 Columns:  1
## -- Column specification ---------------------------------------------------------
## Delimiter:  ","
## dbl (1):  Closer
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.


#Combining 2 values into a single tibble
fig2_tibble = bind_cols(fig2_further, fig2_closer)

#Mutating a new column to show the difference between columns
fig2_tibble = fig2_tibble |>
  mutate(Difference = Closer-Further)
```

To collect the data, I had to first download the source data from the research paper. I could then save the desired variables as .csv files (further and closer), before combining them all into a single tibble. Generating the difference between further and closer just required using `mutate()`. With these variables, it would be possible to generate Figure 2(g) - the closer values would be on the left, further values on the right, and the difference would represent the lines between them.

3. Summarize the data.

   (a) Summarize the further data. Do the data suggest that dopamine in the brains of young zebra finches decreases when they sing further away?

   (b) Summarize the closer data. Do the data suggest that dopamine in the brains of young zebra finches increases when they sing closer to their adult song?

   (c) Summarize the paired differences. Do the data suggest that there is a difference between dopamine in the brains of young zebra finches when they sing further away compared to closer to their adult song?

   (d) **Optional Challenge:** Can you reproduce Figure 2(g)? Note that the you can use `geom_errorbar()` to plot the range created by adding the mean ± one standard deviation.

4. Conduct the inferences they do in the paper. Make sure to report the results a little more comprehensively – that is your parenthetical should look something like: ($t = 23.99$, $p < 0.0001$; $g = 1.34$; 95% CI: 4.43, 4.60).
   Note: Your numbers may vary slightly as they performed some unclear correction of their p-values. I'm waiting to hear back from them via email!

   (a) "The close responses differed significantly from 0 ($p = 1.63 \times 10^{-8}$)."

```
mu0 <- 0
#Calculating for closer (right-sided)
#Manually calculating statistics
x <- fig2_tibble$Closer
xbar <- mean(x)
s <- sd(x)
n <- length(x)
t.stat <- (xbar - mu0)/(s/sqrt(n))
p.val <- pt(q=-abs(t.stat), df = n-1)

#Calculating hedges value
closer_hedges_vals = hedges_g(x = x, mu = mu0, alternative = "greater")

#having t.test calculate values automatically and ensuring they match
closer_t_test = t.test(x=x, mu = mu0, alternative = "greater")
closer_CI = t.test(x=x)$conf.int #Caluclating the CI using a two sided test
```

   (b) "The far responses differed significantly from 0 ($p = 5.17 \times 10^{-8}$)."

```
#Calculating for further (left sided)
#Manually calculating statistics
x <- fig2_tibble$Further
xbar <- mean(x)
s <- sd(x)
n <- length(x)
t.stat <- (xbar - mu0)/(s/sqrt(n))
p.val <- pt(q=-abs(t.stat), df = n-1)

#Calculating hedges value
further_hedges_vals = hedges_g(x = x, mu = mu0, alternative = "less")

#having t.test calculate values automatically and ensuring they match
further_t_test = t.test(x=x, mu = mu0, alternative = "less")
further_CI = t.test(x=x)$conf.int #Caluclating the CI using a two sided test
```

   (c) "The difference between populations was significant ($p = 1.04 \times 10^{-8}$)."

```
#Calculating for difference (two-sided)

#Manually calculating statistics
x <- fig2_tibble$Difference
xbar <- mean(x)
s <- sd(x)
n <- length(x)
t.stat <- (xbar - mu0)/(s/sqrt(n))
p.val <- 2*pt(q=-abs(t.stat), df = n-1)
```

```
#Calculating hedges value
difference_hedges_vals = hedges_g(x = x, mu = mu0, alternative = "two.sided")

#having t.test calculate values automatically and ensuring they match
difference_t_test = t.test(x=x, mu = mu0, alternative = "two.sided")
```

5. Reverse engineer the hypothesis test plot from Lecture 20 to create accurate hypothesis testing plots for each part of the previous question.

   (a) Question 4, part(a).

```
ggdat.t <- tibble(t=seq(-5,5,length.out=1000))|>
mutate(pdf.null = dt(t, df=n-1))


#Making plot for closer
#Pulling t value
obs_t=closer_t_test$statistic[[1]]

ggdat.obs <- tibble(t    = obs_t,
                  y    = 0) # to plot on x-axis

# Resampling to approximate the sampling distribution
# on the data
R <- 1000
resamples <- tibble(t=numeric(R))
for(i in 1:R){
curr.sample <- sample(x=fig2_tibble$Closer,
                      size=n,
                      replace=T)
resamples$t[i] = (mean(curr.sample)-mu0)/(sd(curr.sample)/sqrt(n))
}


s <- sd(fig2_tibble$Closer)
t.breaks <- c(-5, 0,
            qt(0.95, df = n-1), 5,  # rejection region (right)
            obs_t)                  # t-statistic observed
xbar.breaks <- t.breaks * s/(sqrt(n)) + mu0



# Create Plot
closer_plot = ggplot() +
# null distribution
geom_line(data=ggdat.t,
          aes(x=t, y=pdf.null))+
geom_hline(yintercept=0)+
# rejection regions
geom_ribbon(data=subset(ggdat.t, t>=qt(0.95, df=n-1)),
            aes(x=t, ymin=0, ymax=pdf.null),
            fill="grey", alpha=0.5)+
# plot p-value (not visible)
geom_ribbon(data=subset(ggdat.t, t>=t.stat),
            aes(x=t, ymin=0, ymax=pdf.null),
            fill="reg", alpha=0.25)+
# plot observation point
geom_point(data=ggdat.obs, aes(x=t, y=y), color="red")+
# Resampling Distribution
stat_density(data=resamples,
             aes(x=t),
             geom="line", color="grey")+
# clean up aesthetics
theme_bw()+
ylab("Density")+
scale_x_continuous("t",
                   breaks = round(t.breaks,2),
                   sec.axis = sec_axis(~.,
                                       name = bquote(bar(x)),
                                       breaks = t.breaks,
                                       labels = round(xbar.breaks,2)))
```

   (b) Question 4, part(b).

```
#Making plot for further
#Pulling t value
obs_t=further_t_test$statistic[[1]]
```

```r
ggdat.obs <- tibble(t    = obs_t,
                  y    = 0) # to plot on x-axis

# Resampling to approximate the sampling distribution
# on the data
R <- 1000
resamples <- tibble(t=numeric(R))
for(i in 1:R){
curr.sample <- sample(x=fig2_tibble$Further,
                      size=n,
                      replace=T)
resamples$t[i] = (mean(curr.sample)-mu0)/(sd(curr.sample)/sqrt(n))
}

s <- sd(fig2_tibble$Further)
t.breaks <- c(-5, qt(0.05, df = n-1), # rejection region (left)
            0, 5,
            obs_t)                  # t-statistic observed
xbar.breaks <- t.breaks * s/(sqrt(n)) + mu0


# Create Plot
further_plot = ggplot() +
# null distribution
geom_line(data=ggdat.t,
          aes(x=t, y=pdf.null))+
geom_hline(yintercept=0)+
# rejection regions
geom_ribbon(data=subset(ggdat.t, t<=qt(0.05, df=n-1)),
            aes(x=t, ymin=0, ymax=pdf.null),
            fill="grey", alpha=0.5)+
# plot p-value (not visible)
geom_ribbon(data=subset(ggdat.t, t>=t.stat),
            aes(x=t, ymin=0, ymax=pdf.null),
            fill="reg", alpha=0.25)+
# plot observation point
geom_point(data=ggdat.obs, aes(x=t, y=y), color="red")+
# Resampling Distribution
stat_density(data=resamples,
             aes(x=t),
             geom="line", color="grey")+
# clean up aesthetics
theme_bw()+
ylab("Density")+
scale_x_continuous("t",
                   breaks = round(t.breaks,2),
                   sec.axis = sec_axis(~.,
                                       name = bquote(bar(x)),
                                       breaks = t.breaks,
                                       labels = round(xbar.breaks,2)))
```

(c) Question 4, part(c).

```r
#Making plot for difference
#Pulling t value
obs_t=difference_t_test$statistic[[1]]


ggdat.obs <- tibble(t    = obs_t,
                  y    = 0) # to plot on x-axis

# Resampling to approximate the sampling distribution
# on the data
R <- 1000
resamples <- tibble(t=numeric(R))
for(i in 1:R){
curr.sample <- sample(x=fig2_tibble$Difference,
                      size=n,
                      replace=T)
resamples$t[i] = (mean(curr.sample)-mu0)/(sd(curr.sample)/sqrt(n))
}

s <- sd(fig2_tibble$Difference)
t.breaks <- c(-5, qt(0.025, df = n-1), # rejection region (left)
            0,
            qt(0.975, df = n-1), 5,  # rejection region (right)
            obs_t)                  # t-statistic observed
xbar.breaks <- t.breaks * s/(sqrt(n)) + mu0
```

```
# Create Plot
difference_plot = ggplot() +
# null distribution
geom_line(data=ggdat.t,
          aes(x=t, y=pdf.null))+
geom_hline(yintercept=0)+
# rejection regions
geom_ribbon(data=subset(ggdat.t, t>=qt(0.975, df=n-1)),
            aes(x=t, ymin=0, ymax=pdf.null),
            fill="grey", alpha=0.5)+
geom_ribbon(data=subset(ggdat.t, t<=qt(0.025, df=n-1)),
            aes(x=t, ymin=0, ymax=pdf.null),
            fill="grey", alpha=0.5)+
# plot p-value (not visible)
geom_ribbon(data=subset(ggdat.t, t>=t.stat),
            aes(x=t, ymin=0, ymax=pdf.null),
            fill="reg", alpha=0.25)+
# plot observation point
geom_point(data=ggdat.obs, aes(x=t, y=y), color="red")+
# Resampling Distribution
stat_density(data=resamples,
             aes(x=t),
             geom="line", color="grey")+
# clean up aesthetics
theme_bw()+
ylab("Density")+
scale_x_continuous("t",
                   breaks = round(t.breaks,2),
                   sec.axis = sec_axis(~.,
                                       name = bquote(bar(x)),
                                       breaks = t.breaks,
                                       labels = round(xbar.breaks,2)))
```

References

Champely, S. (2020). *pwr: Basic Functions for Power Analysis*. R package version 1.3-0.

Kasdin, J., Duffy, A., Nadler, N., Raha, A., Fairhall, A. L., Stachenfeld, K. L., and Gadagkar, V. (2025). Natural behaviour is learned through dopamine-mediated reinforcement. *Nature*, pages 1–8.