Expressivity and Learnability: Statistics

Contents

Introduction

Reminder of the phases in the experiment:

- Phase 1. Identify a suitable starting language, in the middle of the ExL space
- Phase 2. Subject such a language to 3 different types of engagement: L-only, E-only, L+E
- Phase 3. Measure and plot the learnability of the languages that come out of these three different types of engagement by running another generation of learners on each.

This is a visual summary of the results:

This analysis test two questions:

- Q1: When both pressures apply, do the languages in phase 2 differ from the langauge in phase 1?

We hypothesise that when both expressivity and learnability pressures apply, the resulting languages will increase in both learnability and expressivity.

- Q2: Do the languages in phase 2 differ from each other by condition?

We hypothesise that the learnability should be lower when there is no pressure for learnability, and that the expressivity should be lower when there is no pressure for expressivity.

Q1 can be addressed using simple t-tests. Q2 is addressed in several ways. The first is to run two seperate mixed effects models as implemented in the R package `lme4`. However, languages are placed in a 2-dimensional space, so we also run a multivariate regression (the learnability and expressivity are both treated as the dependent variables). The datapoints are not independent: multiple participants in phase 3 are exposed to the same phase 2 language. To avoid collpsing the data over phase 2 languages we can use a hierarchical

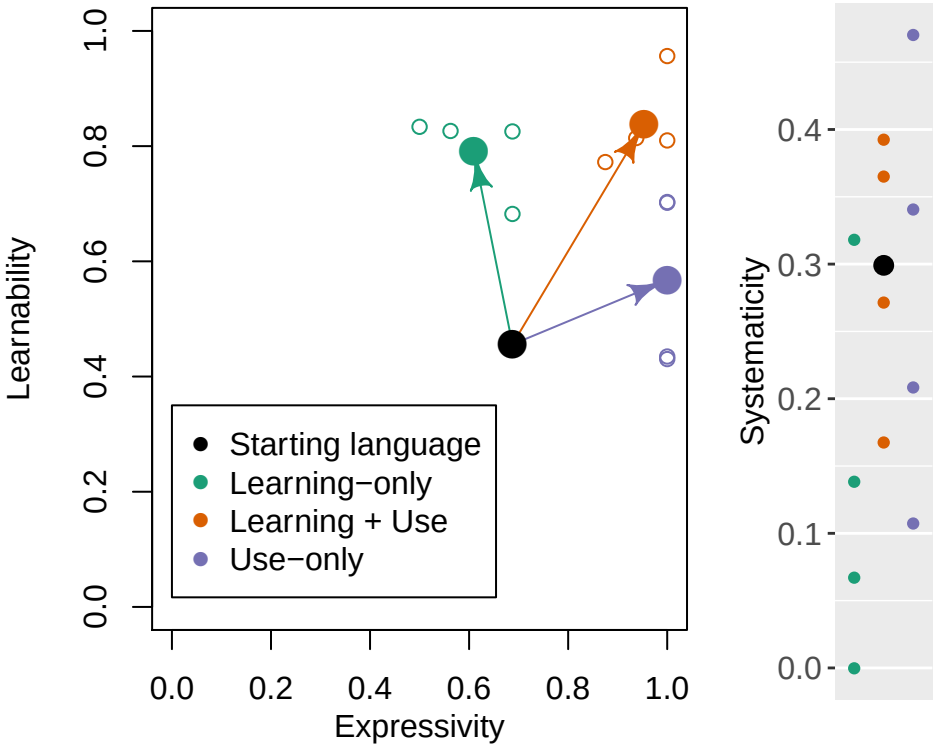


Figure 1: The black circle represents the seed language (expressivity and mean learnability). The open circles represent the phase 2 languages (expressivity and mean learnability, established in phase 3), and are coloured by the experimental condition. The filled circles are the mean points of the phase 2 languages for each condition. The arrows indicate the estimated movement of the language through one generation of transmission (file available in ../results/graphs/MeanPlots.pdf).

model, leading to a test using a multivariate Monte Carlo Markov Chain generalised linear mixed model, as implemented in the R package `MCMCglmm`.

We also demonstrate that the same general results are obtained when collapsing the data and using a simple linear model, and also when using a simple permutation test, which does not make assumptions about the sample size or underlying distribution of the data.

Structure of the data

Each language only has a single expressivity value that can be calculated exactly. However, the learnability of a language is estimated as the mean learnability from 4 participants learning the language. In the data for phase 2 below, each row represents a phase 3 measurement of learnability for a phase 2 language. So, for language A produced in phase 2, there are four rows in the data which represent the responses of four participants to learning language A.

Note that the data also include the learnability tests for all 4 candidate seed languages. These can be recognised if the variable `condition` is "Seed".

The variables in the data are as follows:

- `loadFile`: The identity of the phase 2 language being learned by phase 3 subjects.
- `condition`: The experimental condition
- `cond`: convenient short labels for condition
- `phase`: Phase of the experiment
- `seed`: Original starting language for the chain
- `parent`: Direct parent language
- `Expressivity.orig`: The expressivity measure (proportion of unique labels) in the original range between 0 and 1.
- `Expressivity`: Scaled and centered expressivity measure.
- `Expressivity.num`: The number of unique labels in the range 1 to 16.
- `Expressivity.num.flip`: Reversed number of labels so that 1 = all unique (17 - `Expressivity.num`).
- `Learnability.orig`: The learnability measure (proportion of labels correctly learned) in the original range between 0 and 1.
- `Learnability`: Scaled and centered learnability measure.

Load libraries

```
library(car)
library(caret)
library(MCMCglmm)
library(lme4)
library(lmerTest)
library(BSDA)
```

Load data

```
d = read.csv("../data/ExperimentData_with_all_Learnability.csv", stringsAsFactors = F)
```

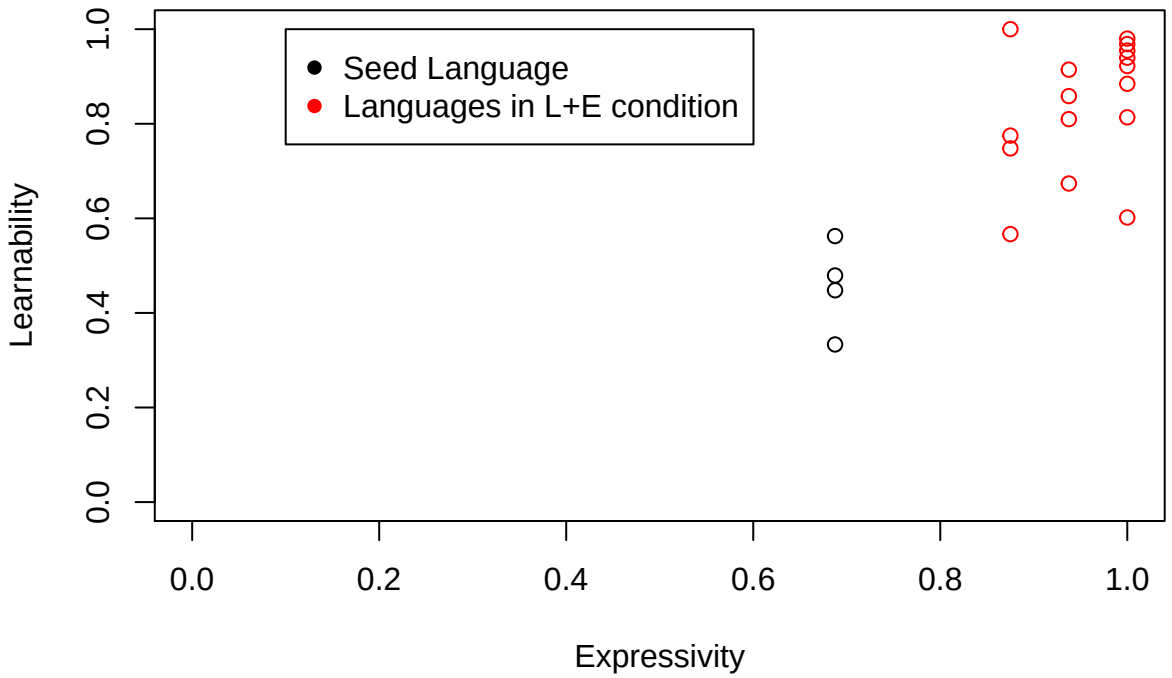
Mark conditions:

```
# Short form of conditions
d$cond = factor(d$condition, levels =
      c("Learnability + Expressivity",
        "Expressivity","Learnability"),
      labels = c("L+E","E","L"))
```

Q1: When both pressures apply, do the languages in phase 2 differ from the langauge in phase 1?

```
d.starting = d[d$condition=="Seed" & d$seed=="Start_4",]
d.LnE = d[d$cond=="L+E" & d$phase=="PH2",]

plot(d.starting$Learnability~d.starting$Expressivity,
     ylim=c(0,1),xlim=c(0,1),
     xlab="Expressivity",
     ylab="Learnability")
points(d.LnE$Learnability~d.LnE$Expressivity, col=2)
legend(0.1,1,legend = c("Seed Language","Languages in L+E condition"), col=1:2,pch=16)
```



Test whether the mean expressivity of the PH2 languages is greater than the expressivity of the seed language (i.e. does expressivity increase?):

```
orig.expressivity = d.starting$Expressivity[1]
p2.expressivity = tapply(d.LnE$Expressivity,
                         d.LnE$loadFile,
                         head, n=1)

t.ex = t.test(p2.expressivity,
              mu=orig.expressivity)
t.ex
```

```
##
##  One Sample t-test
##
## data:  p2.expressivity
## t = 8.878, df = 3, p-value = 0.003013
## alternative hypothesis: true mean is not equal to 0.6875
## 95 percent confidence interval:
```

```
##  0.8579075 1.0483425
## sample estimates:
## mean of x
##  0.953125
```

Does learnability increase?

```
t.test(d.starting$Learnability, d.LnE$Learnability)
```

```
##
##  Welch Two Sample t-test
##
## data:  d.starting$Learnability and d.LnE$Learnability
## t = -6.573, df = 6.4688, p-value = 0.0004358
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.5224148 -0.2425667
## sample estimates:
## mean of x mean of y
## 0.4557292 0.8382200
```

In general, the measures of learnability for the PH2 languages are higher than the measures of learnability for the seed language.

However, the observations for learnability above are not independent. Instead, we can test whether the mean learnability of each PH2 language is greater than the mean learnability of the seed language.

```
seed.mean.Learn = mean(d.starting$Learnability)
LnE.mean.Learn = tapply(d.LnE$Learnability,
           d.LnE$loadFile, mean)

t.ln = t.test(LnE.mean.Learn, mu=seed.mean.Learn)
t.ln
```

```
##
##  One Sample t-test
##
## data:  LnE.mean.Learn
## t = 9.4433, df = 3, p-value = 0.002517
## alternative hypothesis: true mean is not equal to 0.4557292
## 95 percent confidence interval:
##  0.7093185 0.9671214
## sample estimates:
## mean of x
##   0.83822
```

In summary: in the condition with both a learnability and expressivity pressure, the learnability and expressivity increase.

Additional test if the systematicity increases:

```
seed.Sys = d.starting$Systematicity[1]
LnE.Sys = tapply(d.LnE$Systematicity,
                         d.LnE$loadFile,
                         head, n=1)
t.sys = t.test(LnE.Sys,
               mu=seed.Sys)
t.sys
```

```
##
##  One Sample t-test
##
## data:  LnE.Sys
## t = -0.00010142, df = 3, p-value = 0.9999
## alternative hypothesis: true mean is not equal to 0.2991104
## 95 percent confidence interval:
##  0.1369662 0.4612442
## sample estimates:
## mean of x
## 0.2991052
```

Do L-only and E-only differ from the starting condition?

The findings above logically imply that L-only and E-only conditions differ from the starting condition. But let's just double-check that:

```
d.Lonly = d[d$cond=="L" & d$phase=="PH2" & !is.na(d$Learnability),]
Lonly.mean.Learn = tapply(d.Lonly$Learnability,
           d.Lonly$loadFile, mean)

t.Lonly.ln = t.test(Lonly.mean.Learn, mu=seed.mean.Learn)
t.Lonly.ln
```

```
##
##  One Sample t-test
##
## data:  Lonly.mean.Learn
## t = 9.1871, df = 3, p-value = 0.002727
## alternative hypothesis: true mean is not equal to 0.4557292
## 95 percent confidence interval:
##  0.6754804 0.9084172
## sample estimates:
## mean of x
## 0.7919488
```

```
# Take just one row of data per phase 2 output language
Lonly.expressivity = tapply(d.Lonly$Expressivity,
                        d.Lonly$loadFile,
                        head, n=1)

t.Lonly.ex = t.test(Lonly.expressivity,
              mu=orig.expressivity)
t.Lonly.ex
```

```
##
##  One Sample t-test
##
## data:  Lonly.expressivity
## t = -1.6667, df = 3, p-value = 0.1942
## alternative hypothesis: true mean is not equal to 0.6875
## 95 percent confidence interval:
##  0.4601978 0.7585522
## sample estimates:
## mean of x
##  0.609375
```

```
d.Eonly = d[d$cond=="E" & d$phase=="PH2" & !is.na(d$Learnability),]
Eonly.mean.Learn = tapply(d.Eonly$Learnability,
           d.Eonly$loadFile, mean)

t.Eonly.ln = t.test(Eonly.mean.Learn, mu=seed.mean.Learn)
t.Eonly.ln
```

```
##
##  One Sample t-test
##
## data:  Eonly.mean.Learn
```

```
## t = 1.4349, df = 3, p-value = 0.2468
## alternative hypothesis: true mean is not equal to 0.4557292
## 95 percent confidence interval:
##  0.3196077 0.8153831
## sample estimates:
## mean of x
## 0.5674954
```

All phase 2 languages in the E-only condition have an expressivity of 1.0, meaning that the standard deviation is 0 and the t-test does not apply. So we'll assume that the standard deviation is the same as for the L-only condition above:

```
tsum.test(mean.x=1.0, s.x=sd(Lonly.expressivity),
          n.x=4, mu=orig.expressivity)
```

```
## Warning in tsum.test(mean.x = 1, s.x = sd(Lonly.expressivity), n.x = 4, :
## argument 'var.equal' ignored for one-sample test.
```

```
##
##  One-sample t-Test
##
## data:  Summarized x
## t = 6.6667, df = 3, p-value = 0.006881
## alternative hypothesis: true mean is not equal to 0.6875
## 95 percent confidence interval:
##  0.8508228 1.1491772
## sample estimates:
## mean of x
##         1
```

Another way to look at the problem is to calculate the probability of observing 4 perfectly expressive languages, modelling the distribution of expressivity as being drawn from all other observations of expressivity. This gives the same order of probability:

```
nonE_ExpressivityValues = tapply(d[d$cond!="E" & !is.na(d$Expressivity),]$Expressivity,
                                 d[d$cond!="E" & !is.na(d$Expressivity),]$loadFile, mean)
probOfMaxExpressivity = sum(nonE_ExpressivityValues==1)/length(nonE_ExpressivityValues)
# Prob of observing 4
probOfMaxExpressivity^4
```

```
## [1] 0.001189546
```

Q2: Do the languages in phase 2 differ from each other by condition?

In the L+E, we know that learnability and expressivity increase. However, when one of the pressures is missing, does this affect the learnability?

Prepare the data

For convenience later, save phase 2 plus seed language.

```
d.PlusSeed = d[(d$phase == "PH2" & !is.na(d$Learnability)) |
                 (d$phase=="PH1" & d$condition=="Seed" &
                    d$seed=="Start_4"),]
```

Select just Phase 2 languages:

```
d = d[d$condition!="Seed" & d$phase == "PH2" & !is.na(d$Learnability),]
```

Scale measures

```
d$Learnability.orig = d$Learnability
d$Expressivity.orig = d$Expressivity
d$Expressivity.num = d$Expressivity.orig*16
d$Expressivity.num.flip = 17-d$Expressivity.num

pp = preProcess(d[,c('Learnability',"Expressivity")], method="BoxCox")
lambda.Learn = pp$bc$Learnability$lambda
lambda.Exp = pp$bc$Expressivity$lambda
d$Learnability = bcPower(d$Learnability, lambda = lambda.Learn)
d$Expressivity = bcPower(d$Expressivity, lambda = lambda.Exp)


d$Learnability = scale(d$Learnability)
d$Expressivity = scale(d$Expressivity)
scaled_scale_Learn = attr(d$Learnability,"scaled:scale")
scaled_center_Learn = attr(d$Learnability,"scaled:center")
scaled_scale_Exp = attr(d$Expressivity,"scaled:scale")
scaled_center_Exp = attr(d$Expressivity,"scaled:center")
```

Function to covnert scaled variable back into proportion scale:

```
rescale = function(X, lambda,
                   scaled_scale=attr(X,"scaled:scale"),
                   scaled_center = attr(X,"scaled:center")){
  # undo scale
  X = X *scaled_scale + scaled_center
  # BC power is:
  # X = (U^(lambda)-1)/lambda
  # invert:
  return(((X*lambda)+1)^(1/lambda))
}
```

Analysis

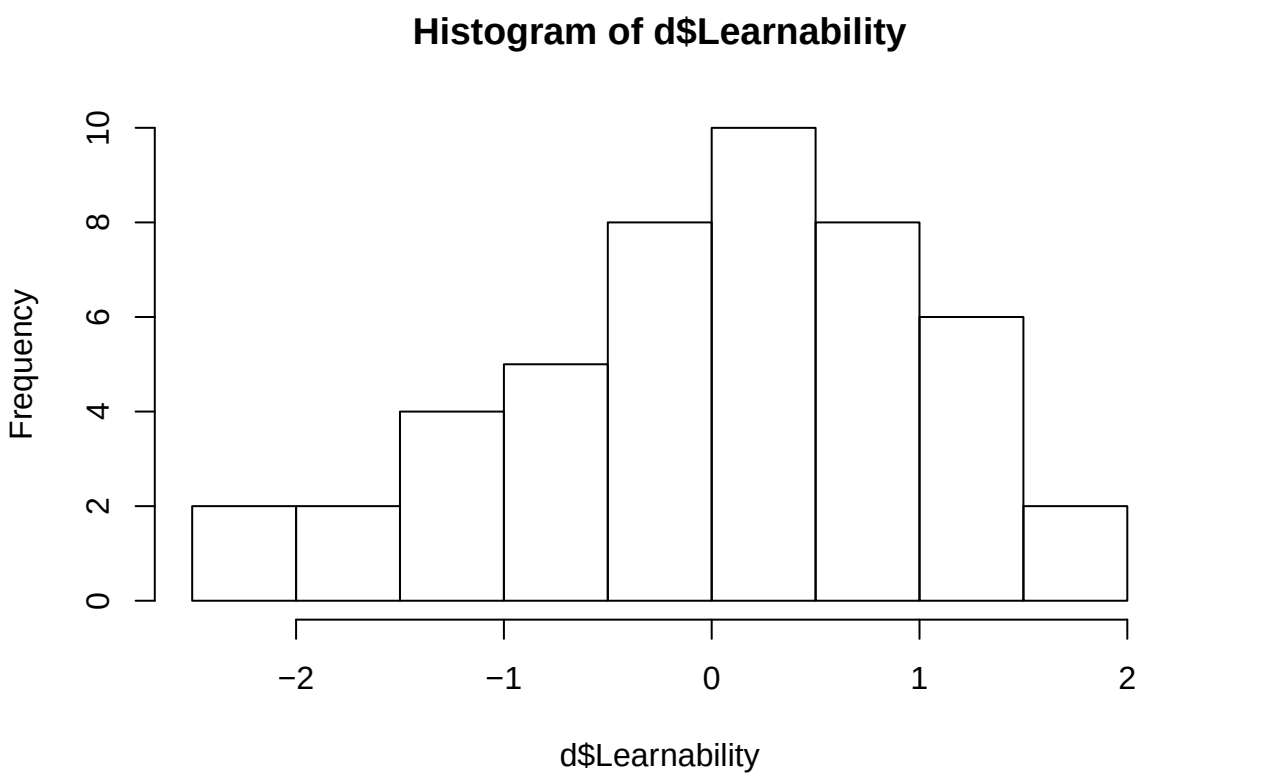
Fit linear mixed models.

Learnability

The datapoints are not independent - the measures of learnability for a language in phase 2 come from different participants in phase 3 learning the same langauge. So there needs to be a random effect for phase 2 source langauge. Random intercepts are added, but condition does not vary by phase 2 source language, so random slopes are not permissable.

Check the normality of the Learnability variable:

```
hist(d$Learnability)
```




```
shapiro.test(d$Learnability)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  d$Learnability
## W = 0.9667, p-value = 0.1977
```

Perform a mixed effects model:

```
mL = lmer(Learnability ~ 1 + cond +
              (1 | loadFile),
          data =d)

summary(mL)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: Learnability ~ 1 + cond + (1 | loadFile)
##    Data: d
##
## REML criterion at convergence: 111.8
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.9048 -0.5217  0.1657  0.6583  1.8325
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  loadFile (Intercept) 0.1631   0.4039
##  Residual             0.5235   0.7235
## Number of obs: 47, groups:  loadFile, 12
##
## Fixed effects:
##             Estimate Std. Error      df t value Pr(>|t|)
## (Intercept)   0.5794     0.2711  8.9639   2.137  0.06142 .
## condE        -1.4009     0.3834  8.9639  -3.654  0.00532 **
## condL        -0.3148     0.3866  9.2336  -0.814  0.43590
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##       (Intr) condE
## condE -0.707
## condL -0.701  0.496
```

Expressivity

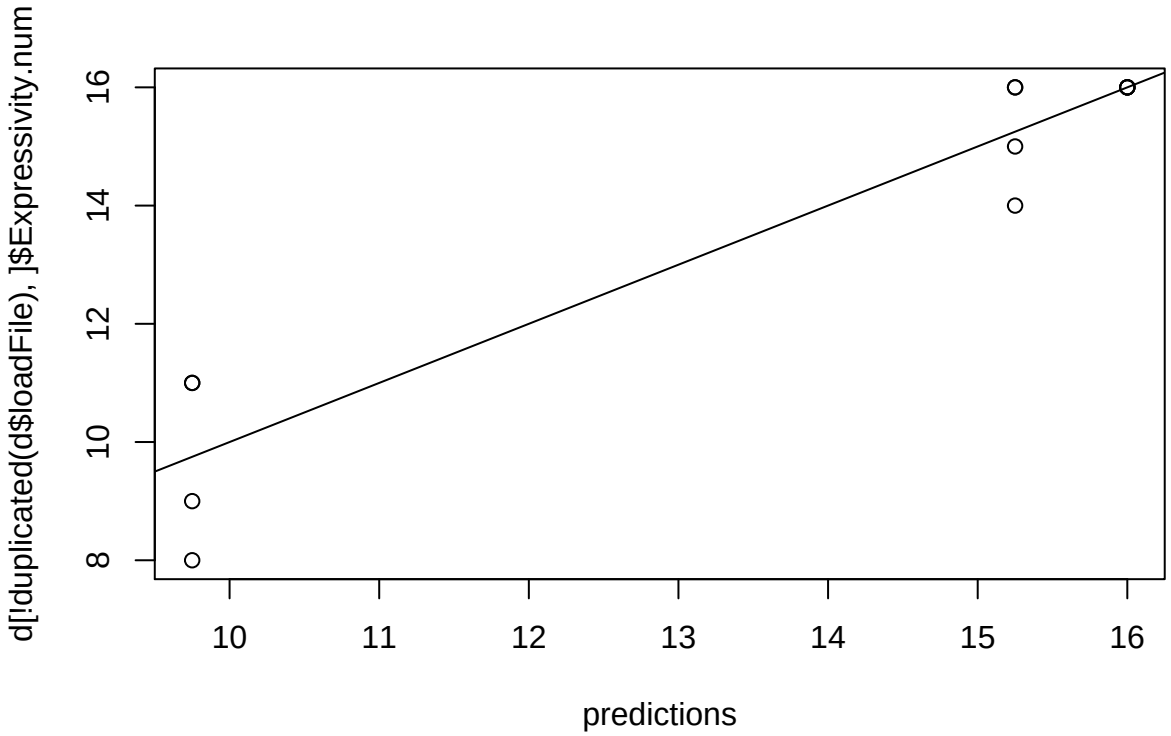
The expressivity values are calculable directly from the phase 2 output languages, and do not need the extra phase 3 steps. Therefore, only one row from the data for each phase 2 language should go into the analysis. This also means that the datapoints are independent, so no random effects are needed.

The expressivity measure is highly skewed, but is underlyingly only whole numbers (the number of unique labels), so we'll use a poisson distribution. The expressivity measure limits are from 1 to 16. The actual values range from 8 to 16. To capture the hard upper limit and prevent the model from predicting expressivity values above 1, we flip the scale (`17 - number of unique labels`).

```
mE = glm(Expressivity.num.flip ~ 1 + cond,
          data =d[!duplicated(d$loadFile),],
         family=poisson(link="log"))
```

Check predictions:

```
predictions = -(exp(predict(mE))-17)
plot(predictions,d[!duplicated(d$loadFile),]$Expressivity.num)
abline(0,1)
```



Results:

```
summary(mE)
```

```
##
## Call:
## glm(formula = Expressivity.num.flip ~ 1 + cond, family = poisson(link = "log"),
##     data = d[!duplicated(d$loadFile), ])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.6171  -0.4786   0.0000   0.2070   0.8567
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.5596     0.3780   1.481 0.138710
## condE        -0.5596     0.6268  -0.893 0.371943
## condL         1.4214     0.4211   3.375 0.000737 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 28.8698  on 11  degrees of freedom
## Residual deviance:  2.4549  on  9  degrees of freedom
## AIC: 41.367
##
## Number of Fisher Scoring iterations: 4
```

Rescale estimates

```
coefL = summary(mL)$coefficients[,1]
intrL = coefL[1]

learn_bothPressures = rescale(intrL,
                              lambda.Learn,
                              scaled_scale_Learn,
                              scaled_center_Learn)
learn_noLearnPressure = rescale(intrL+coefL["condE"],
                                lambda.Learn,
                                scaled_scale_Learn,
learn_noExpPressure = rescale(intrL+coefL["condL"],
                                lambda.Learn,
                                scaled_scale_Learn,
                                scaled_center_Learn)
coefE = summary(mE)$coefficients[,1]
intrE = coefE[1]
exp_bothPressures = (-(exp((intrE))-17))/16
exp_noLearnPressure = (-(exp((intrE+coefE["condE"]))-17))/16
exp_noExpPressure = (-(exp((intrE+coefE["condL"]))-17))/16
```

Summary

Q1:

We hypothesised that when both expresivity and learnability pressures apply, the resulting languages would increase in both learnability and expressivity. We used a simple t-test to compare the mean measures of the languages in phase 2 to the mean measures of the starting language.

When both pressures apply, the expressivity increases (t = 8.8779605, df = 3, p = 0.0030133) and the learnability increases (t = 9.4433113, df = 3, p = 0.0025167) compared to the seed language.

Q2:

Taking the condition with both learnability and expressivity pressures as a reference, we hypothesised that the learnability should be lower when there is no pressure for learnability, and that the expressivity should be lower when there is no pressure for expressivity.

Without a learnability pressure, the learnability drops from 0.8483317 to 0.6043175 (beta = -1.4 , std.err = 0.38 , df = 9 , t = -3.7 , p = 0.0053), but expressivity does not significantly differ (beta = -0.56 std.err 0.63 , z = -0.89 , p = 0.37).

Without an expressivity pressure, the expressivity drops from 0.953125 to 0.609375 (beta = 1.4 std.err 0.42 , z = 3.4 , p = 0.00074), but the learnability does not significantly differ (beta = -0.31 , std.err = 0.39 , df = 9.2 , t = -0.81 , p = 0.44).

Qualitatively similar results were obtained by simpler linear regression, mixed multivariate regression and permutation (below).

Permutation tests

The tests above make assumptions about the distribution of the variables. We can relax this assumption by using a permutation test on the mean learnability and expressivity measures for each language in phase 2. The tests below show that the results are qualitatively the same, and the effect sizes are also similar.

This is a function that tests whether the true difference in means between two samples is greater than the difference when the assignment of observations is permuted between the samples:

```
perm.diff = function(a,b,n=1000){
  # real difference between samples a and b
  true.diff = diff(c(mean(a),mean(b)))
  # permute the values between a and b, and return the difference in means
  perm.diff = replicate(n,
        diff(tapply(c(a,b),
         sample(c(rep("a",length(a)),
           rep("b",length(b)))),
         mean)))
  # proportion of permuted differences less than the true difference
  p = sum(perm.diff<true.diff)/n
  # If there are none, set the probability to 1 / the number of permutations
  if(p==0){
    p = 1/n
  }
  # Z-score
  z = (true.diff-mean(perm.diff))/sd(perm.diff)
  return(c(diff.in.means=true.diff,z=z,p=p))
}
```

Collapse means over each phase 2 language:

```
d.mean = data.frame(
  Learnability=tapply(d$Learnability,d$loadFile,mean),
  Expressivity=tapply(d$Expressivity,d$loadFile,mean),
  Learnability.orig = tapply(d$Learnability.orig,d$loadFile,mean),
  Expressivity.orig = tapply(d$Expressivity.orig,d$loadFile,mean),
  cond = as.factor(tapply(as.character(d$cond),d$loadFile,head,n=1))
)
d.mean$cond = relevel(d.mean$cond,"L+E")
```

Without a learnability pressure, the learnability is significantly lower:

```
set.seed(2378)
perm.diff(
  d.mean$Learnability.orig[d.mean$cond=="L+E"],
  d.mean$Learnability.orig[d.mean$cond=="E"])
```

```
## diff.in.means             z             p
##    -0.2707245    -2.0610973     0.0010000
```

... but expressivity does not significantly differ:

```
set.seed(212)
perm.diff(
  d.mean$Expressivity.orig[d.mean$cond=="L+E"],
  d.mean$Expressivity.orig[d.mean$cond=="E"])
```

```
## diff.in.means             z             p
```

```
##      0.046875      1.443244      0.789000
```

Without an expressivity pressure, the expressivity is significantly lower:

```
set.seed(887)
perm.diff(
  d.mean$Expressivity.orig[d.mean$cond=="L+E"],
  d.mean$Expressivity.orig[d.mean$cond=="L"])
```

```
## diff.in.means             z             p
##     -0.343750     -2.492493      0.001000
```

... but the learnability does not significantly differ:

```
set.seed(768)
perm.diff(
  d.mean$Learnability.orig[d.mean$cond=="L+E"],
  d.mean$Learnability.orig[d.mean$cond=="L"])
```

```
## diff.in.means             z             p
##   -0.04627117   -0.85062246    0.26500000
```

Multivariate model

The data can be thought of as two-dimnesional: languages have an expressivity score and a learnability score, so we can use a multivariate model. We used a multivariate hierarchical model, predicting both learnability and expressivity by condition, with random effects for the phase 2 language whose learnability was being measured.

```
# This sets L+E as the reference condition
d$cond2 = factor(d$cond,levels=c("L+E","E","L"))
set.seed(1234)
prior = list(R = list(V = diag(2)/3, n = 2),
             G = list(G1 = list(V = diag(2)/3, n = 2)))

m <- MCMCglmm(
  cbind(Learnability,Expressivity.num.flip) ~
                -1 + cond2:trait,
              random = ~ us(trait):loadFile,
              rcov = ~ us(trait):units,
              prior = prior,
              family = c("gaussian", "poisson"),
              nitt = 100000,
              burnin = 1000,
              thin = 10,
              data = d,
              verbose = F)

summary(m)
```

```
##
##  Iterations = 1001:99991
##  Thinning interval  = 10
##  Sample size  = 9900
##
##  DIC: 263.0713
##
##  G-structure:  ~us(trait):loadFile
##
##                                                                post.mean
## traitLearnability:traitLearnability.loadFile                     0.32062
## traitExpressivity.num.flip:traitLearnability.loadFile           -0.02331
## traitLearnability:traitExpressivity.num.flip.loadFile           -0.02331
## traitExpressivity.num.flip:traitExpressivity.num.flip.loadFile   0.21513
##                                                                l-95% CI
## traitLearnability:traitLearnability.loadFile                     0.05947
## traitExpressivity.num.flip:traitLearnability.loadFile           -0.27935
## traitLearnability:traitExpressivity.num.flip.loadFile           -0.27935
## traitExpressivity.num.flip:traitExpressivity.num.flip.loadFile   0.04104
##                                                                u-95% CI
## traitLearnability:traitLearnability.loadFile                       0.7388
## traitExpressivity.num.flip:traitLearnability.loadFile              0.2331
## traitLearnability:traitExpressivity.num.flip.loadFile              0.2331
## traitExpressivity.num.flip:traitExpressivity.num.flip.loadFile     0.4915
##                                                                eff.samp
## traitLearnability:traitLearnability.loadFile                       10210
```

```
## traitExpressivity.num.flip:traitLearnability.loadFile              5393
## traitLearnability:traitExpressivity.num.flip.loadFile              5393
## traitExpressivity.num.flip:traitExpressivity.num.flip.loadFile     5702
##
##  R-structure:  ~us(trait):units
##
##                                                                 post.mean
## traitLearnability:traitLearnability.units                        0.549075
## traitExpressivity.num.flip:traitLearnability.units              -0.002147
## traitLearnability:traitExpressivity.num.flip.units              -0.002147
## traitExpressivity.num.flip:traitExpressivity.num.flip.units      0.093688
##                                                                 l-95% CI
## traitLearnability:traitLearnability.units                        0.33172
## traitExpressivity.num.flip:traitLearnability.units              -0.15658
## traitLearnability:traitExpressivity.num.flip.units              -0.15658
## traitExpressivity.num.flip:traitExpressivity.num.flip.units      0.03572
##                                                                 u-95% CI
## traitLearnability:traitLearnability.units                          0.8282
## traitExpressivity.num.flip:traitLearnability.units                 0.1504
## traitLearnability:traitExpressivity.num.flip.units                 0.1504
## traitExpressivity.num.flip:traitExpressivity.num.flip.units        0.1729
##                                                                 eff.samp
## traitLearnability:traitLearnability.units                           9900
## traitExpressivity.num.flip:traitLearnability.units                  3530
## traitLearnability:traitExpressivity.num.flip.units                  3530
## traitExpressivity.num.flip:traitExpressivity.num.flip.units         5725
##
##  Location effects: cbind(Learnability, Expressivity.num.flip) ~ -1 + cond2:trait
##
##                                    post.mean l-95% CI u-95% CI eff.samp
## cond2L+E:traitLearnability            0.5816  -0.0763   1.2565     9900
## cond2E:traitLearnability             -0.8241  -1.5249  -0.1811     9900
## cond2L:traitLearnability              0.2632  -0.4164   0.9226     9900
## cond2L+E:traitExpressivity.num.flip   0.4439  -0.1666   1.0536     3489
## cond2E:traitExpressivity.num.flip    -0.1112  -0.8265   0.5688     2119
## cond2L:traitExpressivity.num.flip     1.9318   1.4029   2.4503     8139
##                                     pMCMC
## cond2L+E:traitLearnability         0.0826 .
## cond2E:traitLearnability           0.0212 *
## cond2L:traitLearnability           0.4347
## cond2L+E:traitExpressivity.num.flip 0.1576
## cond2E:traitExpressivity.num.flip  0.7608
## cond2L:traitExpressivity.num.flip  <1e-04 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Check convergence

```
for(i in list(1:3,4:6)){
  png(paste0("../results/MCMCglmm/MCMCConvergence_",i[1],".png"))
  plot(m$Sol[,i])
  dev.off()
}
for(i in list(1:4,5:8)){
  png(paste0("../results/MCMCglmm/MCMCConvergence_VCV_",i[1],".png"))
  plot(m$VCV[,i])
  dev.off()
}
```

Scale model estimates back into original units:

```
coef = summary(m)$solutions[,1]

learn_bothPressures = rescale(coef["cond2L+E:traitLearnability"],
                              lambda.Learn,
                              scaled_scale_Learn,
                              scaled_center_Learn)

learn_noLearnPressure = rescale(coef["cond2E:traitLearnability"],
                                lambda.Learn,
                                scaled_scale_Learn,
                                scaled_center_Learn)
learn_noExpPressure = rescale(coef["cond2L:traitLearnability"],
                                lambda.Learn,
                                scaled_scale_Learn,
                                scaled_center_Learn)
exp_bothPressures = (-(exp((coef["cond2L+E:traitExpressivity.num.flip"]))-17))/16
exp_noLearnPressure = (-(exp((coef["cond2E:traitExpressivity.num.flip"]))-17))/16
exp_noExpPressure = (-(exp((coef["cond2L:traitExpressivity.num.flip"]))-17))/16
```

Calculate confidence intervals:

```
cis = summary(m)$solutions[,2:3]
learn_bothPressures.ci = rescale(cis["cond2L+E:traitLearnability",],
                              lambda.Learn,
                              scaled_scale_Learn,
                              scaled_center_Learn)

learn_noLearnPressure.ci = rescale(cis["cond2E:traitLearnability",],
                                lambda.Learn,
                                scaled_scale_Learn,
                                scaled_center_Learn)
learn_noExpPressure.ci = rescale(cis["cond2L:traitLearnability",],
                                lambda.Learn,
                                scaled_scale_Learn,
                                scaled_center_Learn)
exp_bothPressures.ci = (-(exp((cis["cond2L+E:traitExpressivity.num.flip",]))-17))/16
exp_noLearnPressure.ci = (-(exp((cis["cond2E:traitExpressivity.num.flip",]))-17))/16
exp_noExpPressure.ci = (-(exp((cis["cond2L:traitExpressivity.num.flip",]))-17))/16
```

Check that the model estimates are accurate (circles are real data, diamonds are morel estimates):

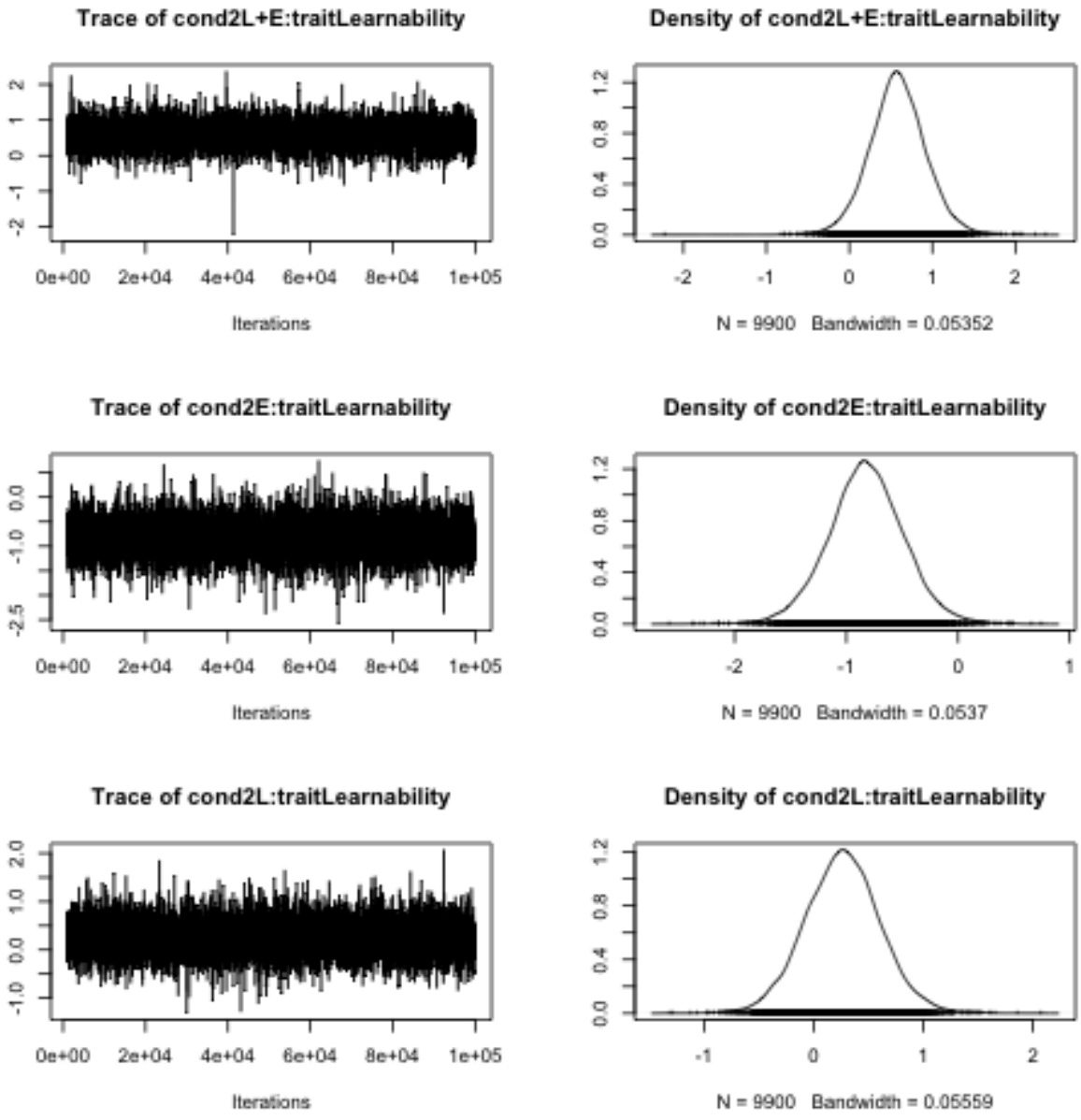

Trace of cond2L+E:traitLearnability
Density of cond2L+E:traitLearnability
Iterations
N = 9900 Bandwidth = 0.05352
Trace of cond2E:traitLearnability
Density of cond2E:traitLearnability
Iterations
N = 9900 Bandwidth = 0.0537
Trace of cond2L:traitLearnability
Density of cond2L:traitLearnability
Iterations
N = 9900 Bandwidth = 0.05559

Figure 2:

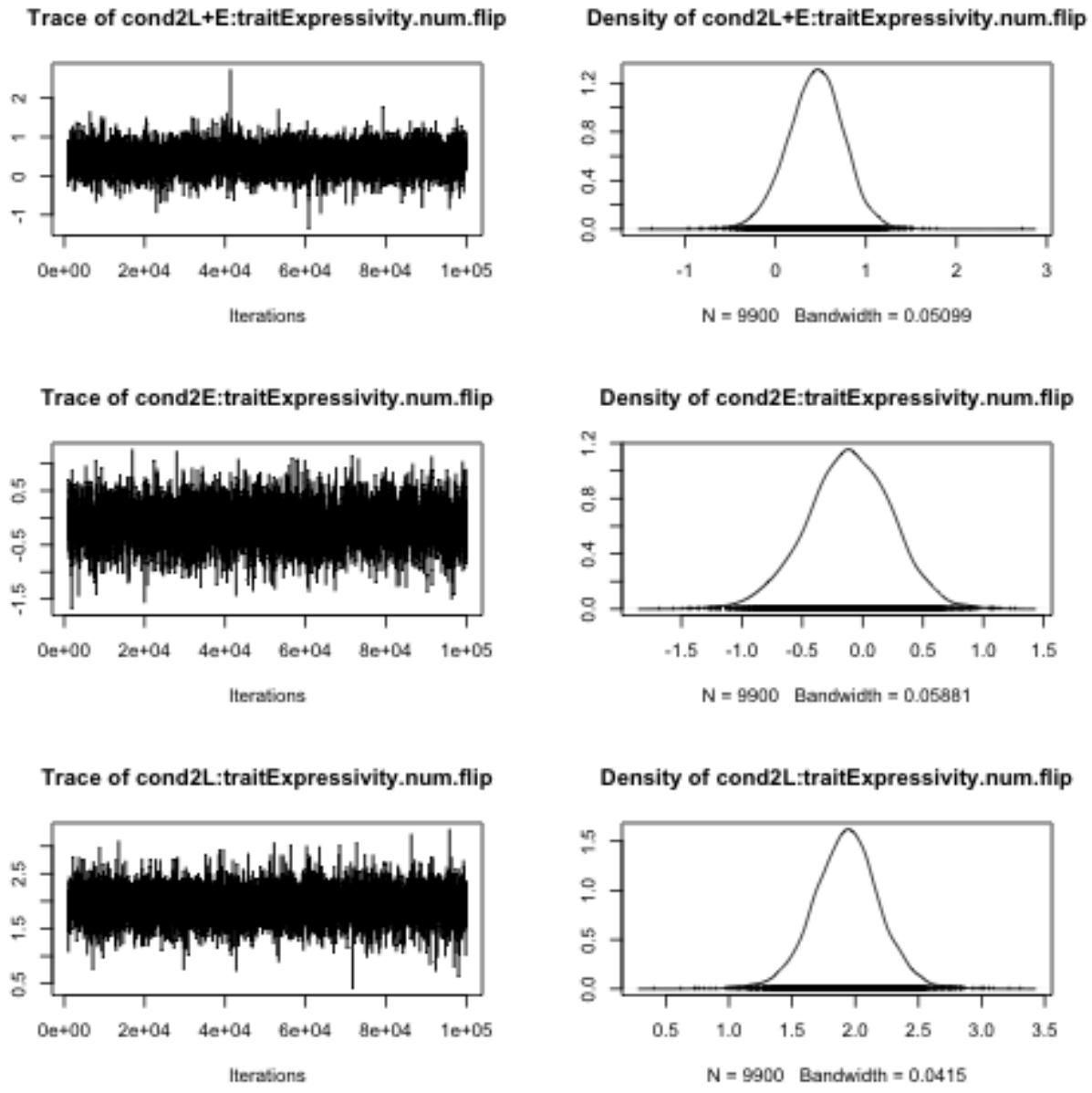

Trace of cond2L+E:traitExpressivity.num.flip
Density of cond2L+E:traitExpressivity.num.flip
Iterations
N = 9900 Bandwidth = 0.05099
Trace of cond2E:traitExpressivity.num.flip
Density of cond2E:traitExpressivity.num.flip
Iterations
N = 9900 Bandwidth = 0.05881
Trace of cond2L:traitExpressivity.num.flip
Density of cond2L:traitExpressivity.num.flip
Iterations
N = 9900 Bandwidth = 0.0415

Figure 3:

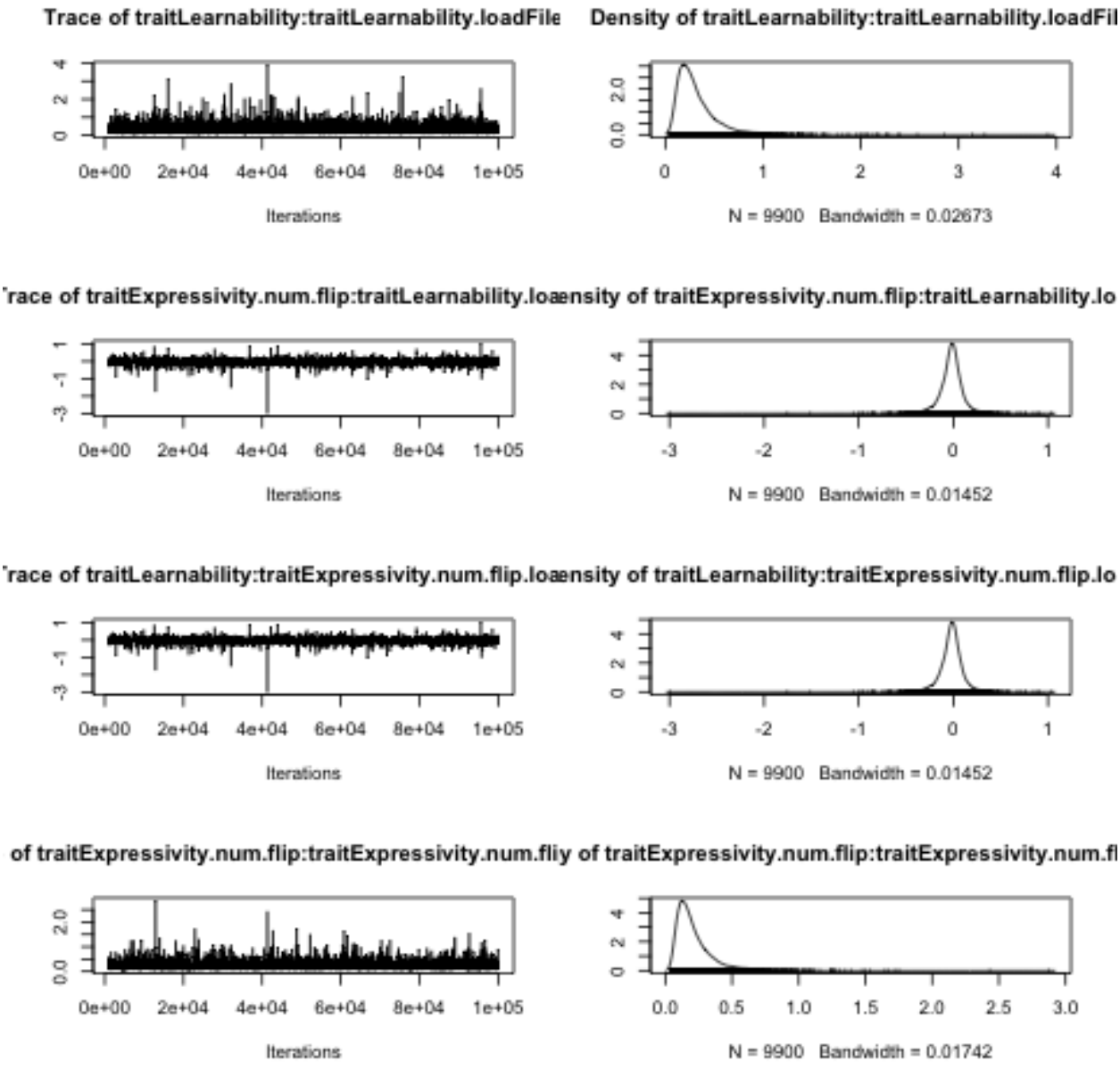

Trace of traitLearnability:traitLearnability.loadFile
Density of traitLearnability:traitLearnability.loadFil
Iterations
N = 9900 Bandwidth = 0.02673
race of traitExpressivity.num.flip:traitLearnability.loa
ensity of traitExpressivity.num.flip:traitLearnability.lo
Iterations
N = 9900 Bandwidth = 0.01452
race of traitLearnability:traitExpressivity.num.flip.loa
ensity of traitLearnability:traitExpressivity.num.flip.lo
Iterations
N = 9900 Bandwidth = 0.01452
of traitExpressivity.num.flip:traitExpressivity.num.fliy of traitExpressivity.num.flip:traitExpressivity.num.fl
Iterations
N = 9900 Bandwidth = 0.01742

Figure 4:

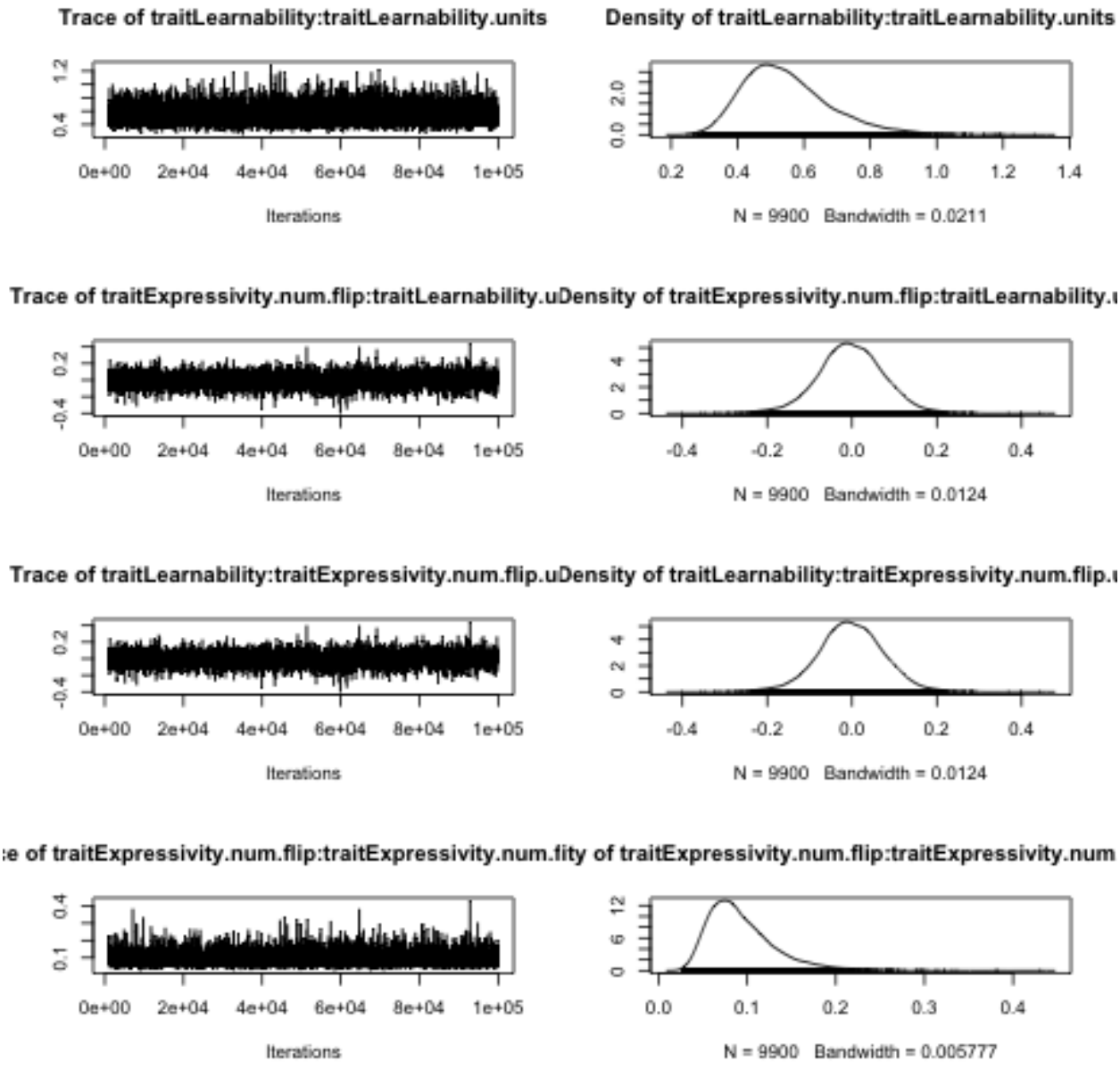
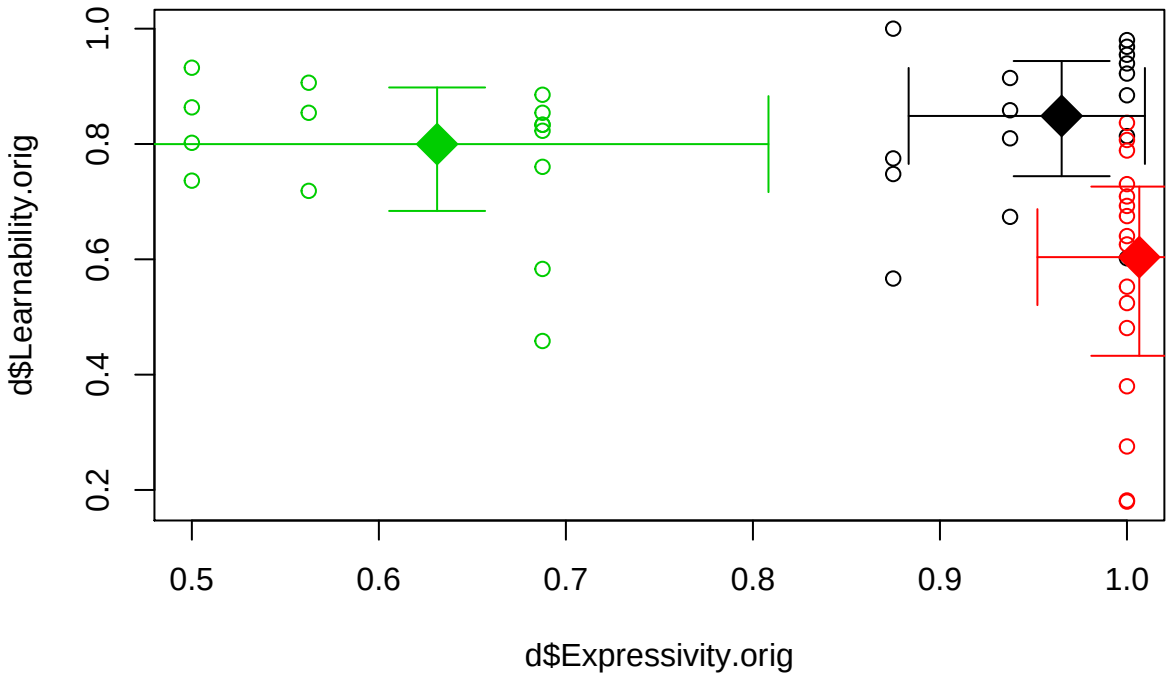

Trace of traitLearnability:traitLearnability.units
Density of traitLearnability:traitLearnability.units
Iterations
N = 9900 Bandwidth = 0.0211
Trace of traitExpressivity.num.flip:traitLearnability.u
Density of traitExpressivity.num.flip:traitLearnability.
Iterations
N = 9900 Bandwidth = 0.0124
Trace of traitLearnability:traitExpressivity.num.flip.u
Density of traitLearnability:traitExpressivity.num.flip.
Iterations
N = 9900 Bandwidth = 0.0124
e of traitExpressivity.num.flip:traitExpressivity.num.fi
ty of traitExpressivity.num.flip:traitExpressivity.num
Iterations
N = 9900 Bandwidth = 0.005777

Figure 5:

```
plot(d$Learnability.orig~
       d$Expressivity.orig,
     col=(as.numeric(d$cond2)))
# Mean points
points(exp_noLearnPressure, learn_noLearnPressure, pch=18,col=2, cex=3)
points(exp_noExpPressure, learn_noExpPressure, pch=18,col=3,cex=3)
points(exp_bothPressures, learn_bothPressures, pch=18,col=1,cex=3)

# CIs
arrows(exp_noLearnPressure,
       learn_noLearnPressure.ci[1],
       exp_noLearnPressure,
       learn_noLearnPressure.ci[2],angle = 90,code = 3,col=2)
arrows(exp_noLearnPressure.ci[1],
       learn_noLearnPressure,
       exp_noLearnPressure.ci[2],
       learn_noLearnPressure,angle = 90,code = 3,col=2)
arrows(exp_noExpPressure,
       learn_noExpPressure.ci[1],
       exp_noExpPressure,
       learn_noExpPressure.ci[2],angle = 90,code = 3,col=3)
arrows(exp_noExpPressure.ci[1],
       learn_noExpPressure,
       exp_noExpPressure.ci[2],
       learn_noExpPressure,angle = 90,code = 3,col=3)
arrows(exp_bothPressures,
       learn_bothPressures.ci[1],
       exp_bothPressures,
       learn_bothPressures.ci[2],angle = 90,code = 3,col=1)
arrows(exp_bothPressures.ci[1],
       learn_bothPressures,
       exp_bothPressures.ci[2],
       learn_bothPressures,angle = 90,code = 3,col=1)
```



Summary

Q2:

Without a learnability pressure, the learnability drops from 0.8486653 to 0.6037854.

Without an expressivity pressure, the expressivity drops from 0.9650794 to 0.6311473 (posterior mean = 1.9 [1.4,2.5], ESS = 8100, p = 1e-04).

Note that the probabilities above reflect the probability of the given parameter estimate being greater than zero, they are not indicative of a comparison between he L+E condition and the given parameter. However, the confidence intervals for the parameter estimates do not overlap for the predicted comparisons, indicating significance.

Linear model

Run a multivariate linear model:

```
m.lm = lm(cbind(Learnability,
                Expressivity) ~ cond, data=d.mean)
summary(m.lm)
```

```
## Response Learnability :
##
## Call:
## lm(formula = Learnability ~ cond, data = d.mean)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.60919 -0.43067 -0.01062  0.29971  0.77289
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.5794     0.2700   2.145  0.06048 .
## condE        -1.4009     0.3819  -3.668  0.00517 **
## condL        -0.3086     0.3819  -0.808  0.43994
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5401 on 9 degrees of freedom
## Multiple R-squared:  0.6228, Adjusted R-squared:  0.539
## F-statistic:  7.43 on 2 and 9 DF,  p-value: 0.01243
##
##
## Response Expressivity :
##
## Call:
## lm(formula = Expressivity ~ cond, data = d.mean)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.5115 -0.1390  0.0000  0.3124  0.3330
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.4872     0.1616   3.015   0.0146 *
## condE         0.3124     0.2286   1.367   0.2049
## condL        -1.8742     0.2286  -8.200 1.82e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3232 on 9 degrees of freedom
## Multiple R-squared:  0.9225, Adjusted R-squared:  0.9052
## F-statistic: 53.55 on 2 and 9 DF,  p-value: 1.006e-05
```

Rescale the estimates back into the original scales:

```
learnEandL = rescale(coef(m.lm)[1,1], lambda.Learn,
                     scaled_scale_Learn,
```

```
                      scaled_center_Learn)
expEandL = rescale(coef(m.lm)[1,2], lambda.Learn,
                   scaled_scale_Exp,
                   scaled_center_Exp)

learnDiff_noLearnPressure = rescale(coef(m.lm)[1,1]+coef(m.lm)[2,1],
                                    lambda.Learn,
                                    scaled_scale_Learn,
                                    scaled_center_Learn)
learnDiff_noExpPressure = rescale(coef(m.lm)[1,1]+coef(m.lm)[3,1],
                                  lambda.Learn,
                                  scaled_scale_Learn,
                                  scaled_center_Learn)
expDiff_noLearnPressure = rescale(coef(m.lm)[1,2]+coef(m.lm)[2,2],lambda.Exp,
                                  scaled_scale_Exp,
                                  scaled_center_Exp)
expDiff_noExpPressure = rescale(coef(m.lm)[1,2]+coef(m.lm)[3,2],
                                lambda.Exp,
                                scaled_scale_Exp,
                                scaled_center_Exp)

lx = as.character(learnEandL)
```

Summary

Without a learnability pressure, the learnability drops from 0.848331721933337 to 0.6037854 (p = 0.0051693), but expressivity does not significantly differ (p = 0.2048823).

Without an expressivity pressure, the expressivity drops from 0.9650794 to 0.6311473 (p = 0.000018), but the learnability does not significantly differ (p = 0.4399447).