PostHoc Power Analysis for Non Interactive Production Experiment

2024-03-13

Load Data

```
## Rows: 813
## Columns: 25
## $ participant_audio_id                <chr> "26386a99-facb-48a1-9ce7-eec8290b4d11~
## $ trial_index                         <dbl> 152, 156, 182, 188, 160, 156, 152, 16~
## $ verb                                <chr> "load", "load", "load", "load", "load~
## $ verb_type                           <chr> "critical", "critical", "critical", "~
## $ scene                               <chr> "fruitplane", "haywagon", "trashtrain~
## $ foregrounded                        <fct> loc, sub, loc, sub, sub, loc, loc, su~
## $ mirroring_condition                 <dbl> 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1~
## $ first_noun_automatic                <chr> "sub", "sub", "sub", "sub", "sub", "s~
## $ `coerced control verb?`             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ transcription                       <chr> "Sally loads fruit onto the plane.", ~
## $ `Correct Event Construal?`          <chr> "TRUE", "TRUE", "TRUE", "TRUE", "TRUE~
## $ perfectSprayLoad                    <chr> "TRUE", "TRUE", "TRUE", "TRUE", "TRUE~
## $ `Speech error`                      <chr> "FALSE", "FALSE", "FALSE", "FALSE", "~
## $ `Right Verb, used as spray-load`    <chr> "TRUE", "TRUE", "TRUE", "TRUE", "TRUE~
## $ bucket_file                         <chr> "cd6d0f15-0c1f-4942-84fe-59dcc6ebca6b~
## $ first_noun_factor                   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ first_noun                          <fct> sub, sub, sub, sub, sub, sub, sub, su~
## $ mirroringCondition                  <int> 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1~
## $ mean_surp                           <dbl> 5.777341, 5.602873, 5.343600, 5.33950~
## $ surprisal_ratio                     <dbl> 0.9809691, 0.9570475, 0.9021118, 0.94~
## $ affectedness                        <dbl> 56.54286, 88.25806, 60.67647, 74.5333~
## $ c_foregrounded                      <dbl> 0.503075, -0.496925, 0.503075, -0.496~
## $ c_affectedness                      <dbl> -12.1441316, 19.5710758, -8.0105181, ~
## $ c_mirroring                         <dbl> -0.5264453, 0.4735547, 0.4735547, 0.4~
## $ c_surprisalRatio                    <dbl> 0.018165057, -0.005756557, -0.0606921~
```

```
##     loc
## sub   0
## loc   1
```

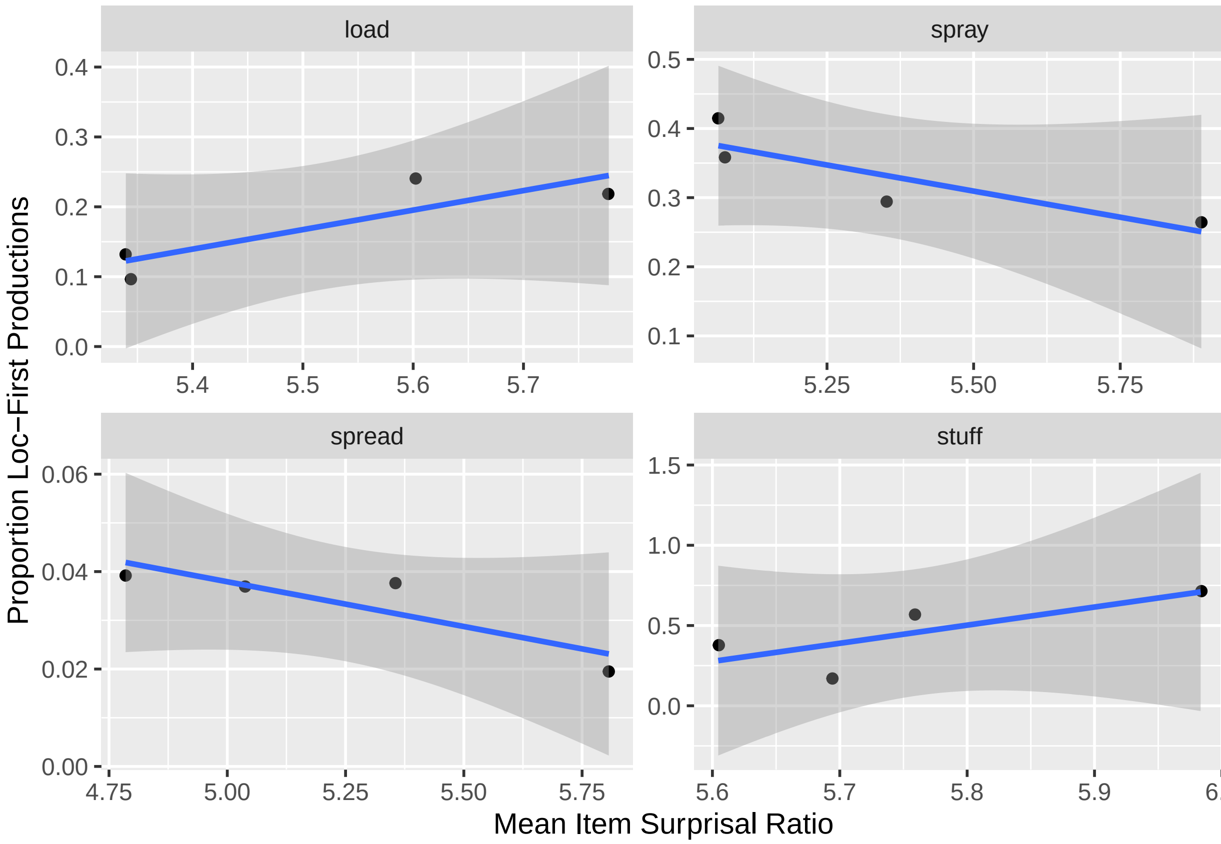
```
##     loc
## sub   0
## loc   1
```

Plot Fixed Effects

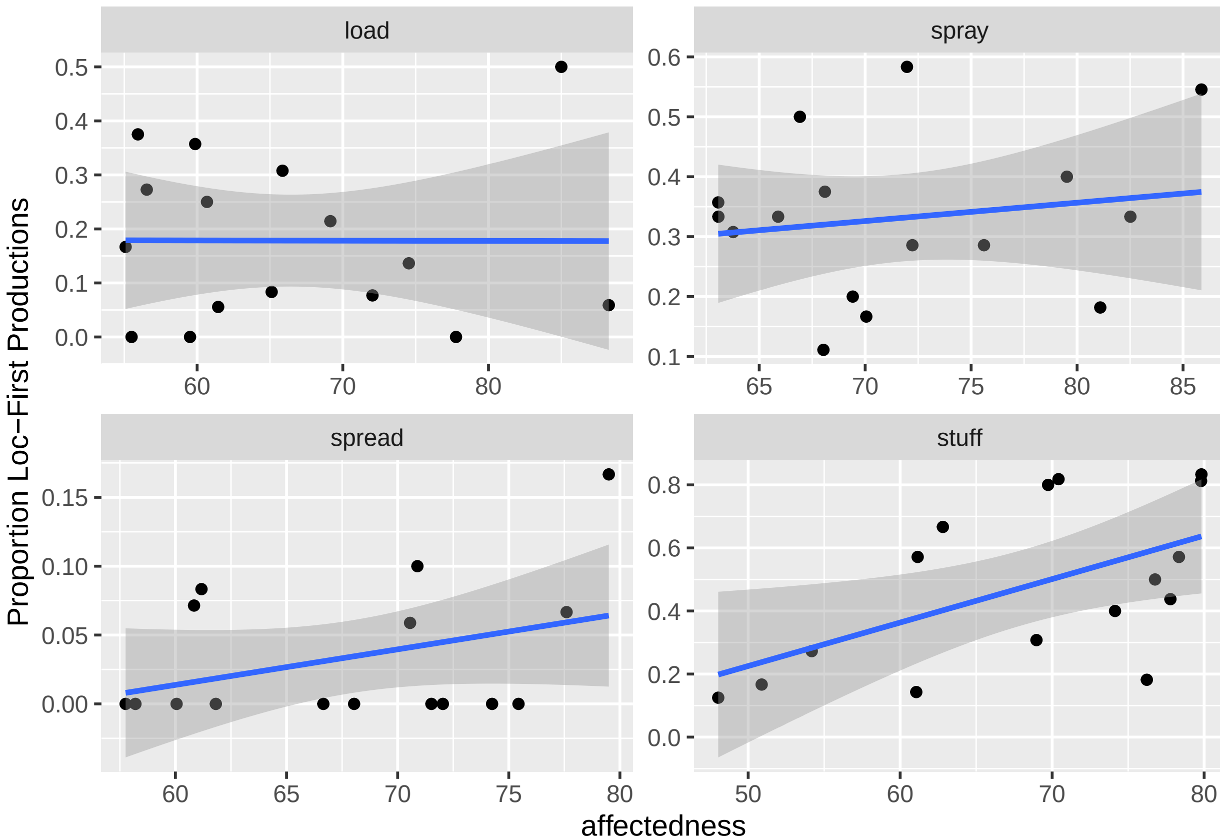
surprisal

```
## `summarise()` has grouped output by 'verb', 'scene'. You can override using the
```

```
## `.groups` argument.
## `geom_smooth()` using formula = 'y ~ x'
```




```
## `summarise()` has grouped output by 'verb', 'scene', 'foregrounded',
## 'mirroringCondition'. You can override using the `.groups` argument.
## `geom_smooth()` using formula = 'y ~ x'
```



Fit Model

A reasonable model that converges. I walked down from many maximal models that didn't converge but not as methodically as one could: you possibly could get a more complicated random effect structure to converge with more iterations.

No significant effect of surprisal. Locaion-first forms are significantly more likely with foregrounded locations and a very slight additional boost for location foregrounded + unit increase in affectedness norm.

Surprisal is insignificant but at least this time (as opposed to interactive) the effect is numerically in the right direction: Higher ratios are associated with more loc-first forms (higher ratio = sub-first has a higher surprisal).

```
fit.smallest = glmer(first_noun ~ c_foregrounded*c_affectedness + c_mirroring +
                       c_surprisalRatio +
                       (1 | participant_audio_id)+
                       (1 | scene),
                     data = df.model_data,
                     family = "binomial")

summary(fit.smallest)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
```

```
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: first_noun ~ c_foregrounded * c_affectedness + c_mirroring +
##     c_surprisalRatio + (1 | participant_audio_id) + (1 | scene)
##    Data: df.model_data
##
##      AIC      BIC   logLik deviance df.resid
##    695.6    733.2   -339.8    679.6      805
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.6518 -0.4186 -0.2054 -0.0470  4.1905
##
## Random effects:
##  Groups               Name        Variance Std.Dev.
##  participant_audio_id (Intercept) 2.339    1.529
##  scene                (Intercept) 2.346    1.532
## Number of obs: 813, groups:  participant_audio_id, 55; scene, 16
##
## Fixed effects:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   -1.961346   0.460469  -4.259 2.05e-05 ***
## c_foregrounded                 0.627694   0.226413   2.772  0.00557 **
## c_affectedness                 0.014490   0.021887   0.662  0.50794
## c_mirroring                   -0.366394   0.222747  -1.645  0.09999 .
## c_surprisalRatio              14.644793   9.510475   1.540  0.12359
## c_foregrounded:c_affectedness  0.001426   0.026702   0.053  0.95742
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) c_frgr c_ffct c_mrrr c_srpR
## c_foregrndd -0.044
## c_affctdnss  0.008  0.324
## c_mirroring  0.016 -0.050 -0.100
## c_surprslRt -0.032 -0.005 -0.032 -0.039
## c_frgrndd:_  0.040 -0.062  0.163 -0.262  0.019
```

Power Analysis

```r
model <- fit.smallest

# extract fixed effect names
f.effects <- row.names(summary(model)$coefficients)[-1] # remove Intercept

# First, check that we can caluclate a z test for each fixed effect
for (f in f.effects){
  p <- doTest(model,fixed(f,"z"))
  print(p)
}
```

```r
# Next Run powerSim on all fixed effects
# this loop gets power level and confidence interval for observed effect
# and stores results in data.frame powerSim.results
powerSim.results <- data.frame()
# 100 sims takes in on 2015 Macbook Pro (~2 min for each fixed effects)
n_sim <- 100

for (f in f.effects){
  res <- powerSim(model,fixed(f,"z"), nsim=n_sim)
  # calculate power as n. simulations where p-value < alpha
  # powersim default alpha is .05
  power <- sum(res$pval < res$alpha) / res$n
  # calculate confidence intervals same way as print out of powerSim does
  # formula here: https://github.com/pitakakariki/simr/blob/master/R/print.R
  cis <- as.matrix(binom.confint(sum(res$pval < res$alpha),
                           res$n,conf.level=0.95,
                           methods=getSimrOption("binom"),
                           alpha=res$alpha)[c("lower","upper")])
  # get effect size
  eff <- model@beta[which(f.effects == f)+1]
  # add results to powerSim.results
  powerSim.results <- rbind(
    powerSim.results,
    c(f,eff,power,cis[[1]],cis[[2]]))
}
colnames(powerSim.results) <- c("EFFECT","SIZE","POWER","LOWER95","UPPER95")
powerSim.results
# save(powerSim.results, file = "02_availabilityInProduction_PowerSim-results.Rda")
```

We have 80% power for the foregrounding

```r
load("02_availabilityInProduction_PowerSim-results.Rda")
colnames(powerSim.results) <- c("EFFECT","SIZE","POWER","LOWER95","UPPER95")
powerSim.results
```

```
##                          EFFECT                SIZE POWER           LOWER95
## 1                c_foregrounded   0.627694011458634   0.8 0.708157310911372
## 2                c_affectedness  0.0144903177482357  0.34 0.248223501544844
## 3                   c_mirroring   -0.36639403398053  0.43 0.331391016631383
## 4              c_surprisalRatio    14.6447927061885   0.5 0.398321129503301
## 5 c_foregrounded:c_affectedness 0.00142562994551713   0.3 0.212406420489537
##             UPPER95
## 1 0.873344447898044
## 2 0.441533267750959
## 3 0.532866261675154
## 4 0.601678870496699
## 5 0.399814676179804
```