

Fault Detection Modeling

Alexander Nielsen

January 23, 2026

Packages

```
pkgs <- c(
  "tidyverse",
  "lme4",
  "readr",
  "broom.mixed",
  "sjPlot",
  "emmeans",
  "ggplot2",
  "ggeffects",
  "marginaleffects",
  "devtools",
  "scales",
  "kableExtra"
)

to_install <- pkgs[!pkgs %in% rownames(installed.packages())]
if (length(to_install) > 0) install.packages(to_install)

invisible(lapply(pkgs, library, character.only = TRUE))
```

Load data

```
csv_path <- "../data/results-2026-01-23-115413/results.csv"

d_raw <- readr::read_csv(csv_path, show_col_types = FALSE)
d_raw
```

```
## # A tibble: 47,500 x 22
##   total_faults faults_detected preperation_time_ns reduction_time_ns   fft
##   <dbl>         <dbl>         <dbl>         <dbl> <dbl>
## 1         1         0         1894         158741 -1
## 2         1         0         1249         81927 -1
## 3         1         0          853         70245 -1
## 4         1         1          673         67341 75
## 5         1         1          720         70013 75
## 6         1         1          692         70502 75
## 7         1         1          657         54874 75
## 8         1         1          589         52269 75
## 9         1         1          690         85357 75
## 10        1         1          584         49625 75
```

```
## # i 47,490 more rows
## # i 17 more variables: tsr <dbl>, fdl <dbl>, apfd <dbl>, timestamp_utc <dtm>,
## #   test_suite <chr>, program <chr>, version <chr>, language <chr>,
## #   n_total_tests <dbl>, budget_prop_requested <dbl>, n_selected <dbl>,
## #   budget_prop_achieved <dbl>, run_id <dbl>, random_seed <dbl>, method <chr>,
## #   algorithm_family <chr>, representation <chr>
```

Prepare data

```
eps <- 1e-6

df <- d_raw %>%
  mutate(
    # Ensure types
    total_faults = as.integer(total_faults),
    faults_detected = as.integer(faults_detected),
    n_total_tests = as.integer(n_total_tests),

    # Derived columns
    failures = as.integer(total_faults - faults_detected),
    prop = as.numeric(faults_detected / total_faults),
    budget = as.numeric(budget_prop_achieved),
    budget_n = (budget - 0.1) * 100,
    budget_f = factor(budget_prop_requested),

    # Logit transform of budget (bounded away from 0/1)
    p = pmin(pmax(budget, eps), 1 - eps),
    budget_logit = as.numeric(qlogis(p)),

    # Factors
    program = factor(program),
    language = relevel(factor(language), ref = "C"),
    algorithm_family = relevel(factor(algorithm_family), ref = "cs"),
    representation = relevel(factor(representation), ref = "tf_srp"),

    # Paired-trial identifier: all methods share same (language, random_seed)
    trial_id = interaction(language, random_seed, drop = TRUE)
  ) %>%
  filter(
    !is.na(total_faults),
    !is.na(faults_detected),
    total_faults >= 0,
    faults_detected >= 0,
    faults_detected <= total_faults,
    representation != "SuiteLength"
  )

df

## # A tibble: 38,000 x 30
##   total_faults faults_detected preperation_time_ns reduction_time_ns   fft
##           <int>           <int>              <dbl>             <dbl> <dbl>
## 1             1             1          836199564          3880915     10
## 2             1             1          781444526           622127     10
```

```
## 3      1      1      676557106      655405      10
## 4      1      1      675780958      588977      10
## 5      1      1      678845008      567905      10
## 6      1      1      679787420      680354      10
## 7      1      1      684625560      630930      10
## 8      1      1      684702477      635234      10
## 9      1      1      684345841      603895      10
## 10     1      1      681123855      611206      10
## # i 37,990 more rows
## # i 25 more variables: tsr <dbl>, fdl <dbl>, apfd <dbl>, timestamp_utc <dtm>,
## #   test_suite <chr>, program <fct>, version <chr>, language <fct>,
## #   n_total_tests <int>, budget_prop_requested <dbl>, n_selected <dbl>,
## #   budget_prop_achieved <dbl>, run_id <dbl>, random_seed <dbl>, method <chr>,
## #   algorithm_family <fct>, representation <fct>, failures <int>, prop <dbl>,
## #   budget <dbl>, budget_n <dbl>, budget_f <fct>, p <dbl>, ...
```

Models

No interaction model

Hypothesis: Budget effect is non-linear

```
budget_linear <- glmer(
  cbind(faults_detected, failures) ~
    algorithm_family + representation + budget + language +
    (1 | test_suite / run_id),
  family = binomial(link = "logit"),
  data = df
)

budget_quad <- glmer(
  cbind(faults_detected, failures) ~
    algorithm_family + representation + (budget + I(budget^2)) + language +
    (1 | test_suite / run_id),
  family = binomial(link = "logit"),
  data = df
)

budget_logit <- glmer(
  cbind(faults_detected, failures) ~
    algorithm_family + representation + budget_logit + language +
    (1 | test_suite / run_id),
  family = binomial(link = "logit"),
  data = df
)

anova(budget_linear, budget_quad, test = "Chisq")
```

Test of budget linear vs non-linear

```
## Data: df
## Models:
## budget_linear: cbind(faults_detected, failures) ~ algorithm_family + representation + budget + language
## budget_quad: cbind(faults_detected, failures) ~ algorithm_family + representation + (budget + I(budget^2)) + language
##           npar   AIC   BIC logLik -2*log(L)  Chisq Df Pr(>Chisq)
```

```
## budget_linear    7 65386 65446 -32686    65372
## budget_quad      8 65387 65456 -32686    65371 0.7098  1    0.3995
```

```
AIC(budget_linear, budget_quad, budget_logit)
```

```
##           df      AIC
## budget_linear 7 65385.89
## budget_quad   8 65387.18
## budget_logit  7 65266.37
```

```
BIC(budget_linear, budget_quad, budget_logit)
```

```
##           df      BIC
## budget_linear 7 65445.71
## budget_quad   8 65455.54
## budget_logit  7 65326.19
```

Quadratic budget model is not significantly better than linear budget model. However, the logit budget model is strongly preferred by both AIC and BIC. Therefore, we will use the logit budget model for further modeling.

Final model

```
m <- glmer(
  cbind(faults_detected, failures) ~
    representation * algorithm_family +
    budget_logit +
    language +
    (1 | test_suite / run_id),
  family = binomial,
  data = df
)
```

```
anova(budget_logit, m, test = "Chisq")
```

```
## Data: df
## Models:
## budget_logit: cbind(faults_detected, failures) ~ algorithm_family + representation + budget_logit +
## m: cbind(faults_detected, failures) ~ representation * algorithm_family + budget_logit + language +
##           npar    AIC    BIC logLik -2*log(L)  Chisq Df Pr(>Chisq)
## budget_logit    7 65266 65326 -32626    65252
## m                8 65212 65280 -32598    65196 56.744  1  4.964e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Overdispersion check

```
overdisp_fun <- function(model) {
  rdf <- df.residual(model)
  rp <- residuals(model, type = "pearson")
  sqrt(sum(rp^2) / rdf)
}
```

```
overdisp_fun(m)
```

```
## [1] 1.105687
```

Final model estimated marginal means

```
emm <- emmeans(m, ~ representation * algorithm_family | language, type = "response")
pairs(emm, adjust = "holm")
```

```
## language = C:
## contrast odds.ratio SE df null z.ratio p.value
## tf_srp cs / emb cs 1.010 0.0201 Inf 1 0.478 0.6326
## tf_srp cs / tf_srp pp 1.076 0.0213 Inf 1 3.706 0.0006
## tf_srp cs / emb pp 0.878 0.0177 Inf 1 -6.453 <0.0001
## emb cs / tf_srp pp 1.066 0.0211 Inf 1 3.228 0.0025
## emb cs / emb pp 0.870 0.0175 Inf 1 -6.930 <0.0001
## tf_srp pp / emb pp 0.816 0.0163 Inf 1 -10.151 <0.0001
##
## language = Java:
## contrast odds.ratio SE df null z.ratio p.value
## tf_srp cs / emb cs 1.010 0.0201 Inf 1 0.478 0.6326
## tf_srp cs / tf_srp pp 1.076 0.0213 Inf 1 3.706 0.0006
## tf_srp cs / emb pp 0.878 0.0177 Inf 1 -6.453 <0.0001
## emb cs / tf_srp pp 1.066 0.0211 Inf 1 3.228 0.0025
## emb cs / emb pp 0.870 0.0175 Inf 1 -6.930 <0.0001
## tf_srp pp / emb pp 0.816 0.0163 Inf 1 -10.151 <0.0001
##
## P value adjustment: holm method for 6 tests
## Tests are performed on the log odds ratio scale
```

Final model summary

```
summary(m)

## Generalized linear mixed model fit by maximum likelihood (Laplace
## Approximation) [glmerMod]
## Family: binomial (logit)
## Formula:
## cbind(faults_detected, failures) ~ representation * algorithm_family +
## budget_logit + language + (1 | test_suite/run_id)
## Data: df
##
## AIC BIC logLik -2*log(L) df.resid
## 65211.6 65280.0 -32597.8 65195.6 37992
##
## Scaled residuals:
## Min 1Q Median 3Q Max
## -17.9556 -0.4722 0.2510 0.5445 12.3672
##
## Random effects:
## Groups Name Variance Std.Dev.
## run_id:test_suite (Intercept) 0.3506 0.5921
## test_suite (Intercept) 0.9254 0.9620
## Number of obs: 38000, groups: run_id:test_suite, 500; test_suite, 10
##
## Fixed effects:
## Estimate Std. Error z value Pr(>|z|)
## (Intercept) 3.086434 0.432260 7.140 9.32e-13 ***
```

```
## representationemb          -0.009503   0.019879  -0.478 0.632609
## algorithm_familypp         -0.073306   0.019782  -3.706 0.000211 ***
## budget_logit               0.978601   0.006160 158.861 < 2e-16 ***
## languageJava              -2.575163   0.611155  -4.214 2.51e-05 ***
## representationemb:algorithm_familypp 0.212601   0.028204   7.538 4.78e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##      (Intr) rprsnt algrt_ bdgt_l lnggJv
## reprsnttnmb -0.023
## algrthm_fml -0.023  0.503
## budget_logt  0.018 -0.001 -0.010
## languageJav -0.707  0.000  0.000 -0.010
## rprsnttnm:_  0.017 -0.705 -0.702  0.019 -0.001
```

```
fixed_effects_table <- tidy(
  m,
  effects = "fixed",
  conf.int = TRUE,
  conf.method = "Wald"
) %>%
  mutate(
    odds_ratio = exp(estimate),
    conf.low.or = exp(conf.low),
    conf.high.or = exp(conf.high)
  ) %>%
  select(
    term,
    estimate,
    std.error,
    odds_ratio,
    conf.low.or,
    conf.high.or,
    p.value
  )
```

```
fixed_effects_table
```

```
## # A tibble: 6 x 7
##   term          estimate std.error odds_ratio conf.low.or conf.high.or p.value
##   <chr>          <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
## 1 (Intercept)    3.09      0.432    21.9      9.39     51.1  9.32e-13
## 2 representatio~ -0.00950  0.0199    0.991     0.953     1.03  6.33e- 1
## 3 algorithm_fam~ -0.0733   0.0198    0.929     0.894     0.966 2.11e- 4
## 4 budget_logit   0.979     0.00616   2.66      2.63      2.69  0
## 5 languageJava  -2.58      0.611     0.0761    0.0230     0.252 2.51e- 5
## 6 representatio~  0.213     0.0282    1.24      1.17      1.31  4.78e-14
```

```
tab <- fixed_effects_table %>%
  mutate(
    estimate = round(estimate, 3),
    std.error = round(std.error, 3),
    odds_ratio = round(odds_ratio, 3),
    conf.low.or = round(conf.low.or, 3),
```

```

conf.high.or= round(conf.high.or, 3),
p.value      = format.pval(p.value, digits = 3, eps = 0.001)
) %>%
rename(
  Term = term,
  `Log-odds` = estimate,
  `SE` = std.error,
  `OR` = odds_ratio,
  `CI low` = conf.low.or,
  `CI high` = conf.high.or,
  `p` = p.value
)

kable(tab, format = "latex", booktabs = TRUE,
      caption = "GLMM fixed effects (odds ratios with 95\\% CI).") %>%
kable_styling(latex_options = c("hold_position"))

```

Table 1: GLMM fixed effects (odds ratios with 95% CI).

Term	Log-odds	SE	OR	CI low	CI high	p
(Intercept)	3.086	0.432	21.899	9.386	51.093	<0.001
representationemb	-0.010	0.020	0.991	0.953	1.030	0.633
algorithm_familypp	-0.073	0.020	0.929	0.894	0.966	<0.001
budget_logit	0.979	0.006	2.661	2.629	2.693	<0.001
languageJava	-2.575	0.611	0.076	0.023	0.252	<0.001
representationemb:algorithm_familypp	0.213	0.028	1.237	1.170	1.307	<0.001