reQTL Model Selection Using QQplot

Genevieve Roberts

2025-07-24

Load Packages

```
library(lme4)
library(lmerTest)
library(emmeans)
library(data.table)
library(here)
library(dplyr)
library(stringr)
library(tidyr)
library(tibble)
library(purrr)
library(ggplot2)
library(broom)
library(broom.mixed)
library(ggrepel)
```

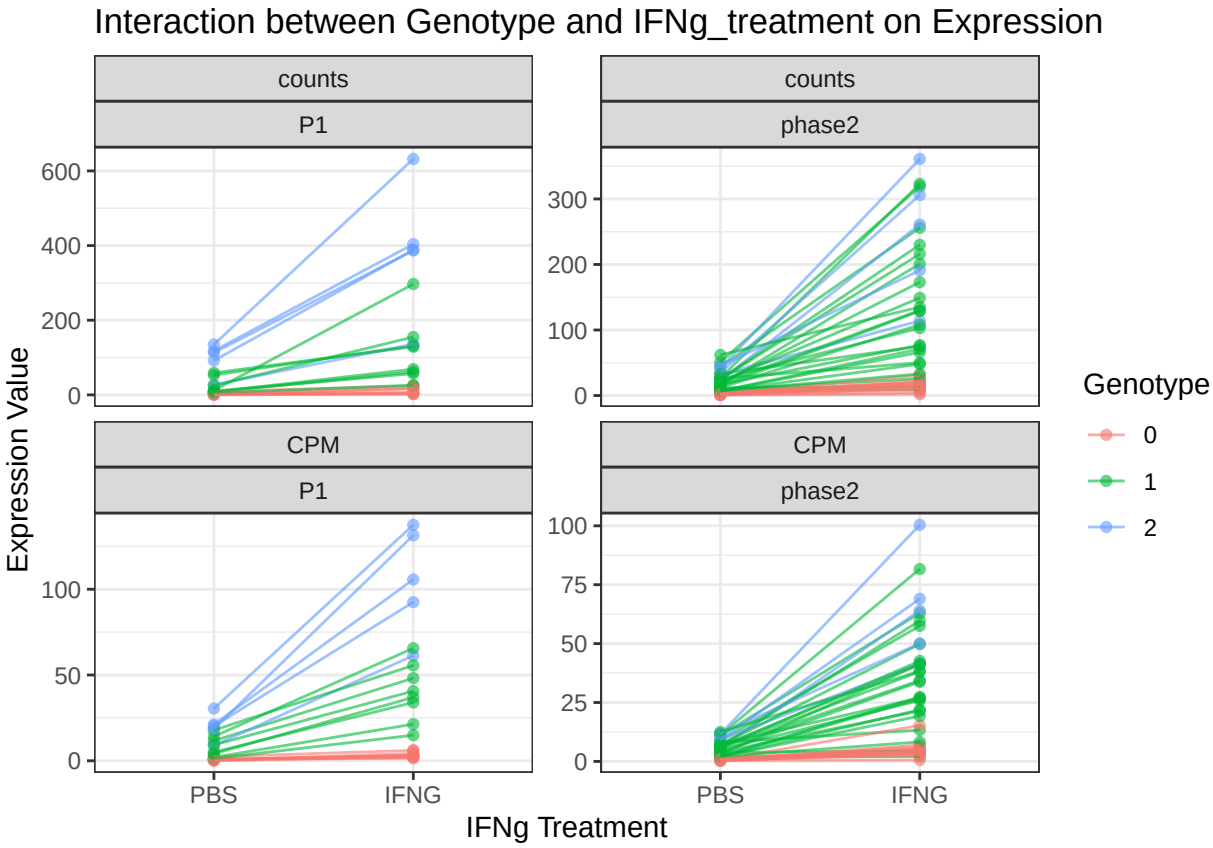
Explore the real data

This is a presumably "real" reQTL SNP in melanocytes. The genotype is rs2910686 and the gene expression is ERAP2. It's presumably real because it is also a known psoriasis GWAS SNP and in high LD to a causal SNP validated in vitro.

```
#load the real data
real_dat <- read.csv(here::here("notebooks/long_form_reQTL_data.csv")) %>%
  arrange(donor)

#make sure the reference group is set to "PBS"
real_dat$condition <- relevel(factor(real_dat$condition), ref = "PBS")
real_dat$genotype.nt <- relevel(factor(real_dat$genotype.nt), ref = "TT")
```

Plot the real data

```
# Create the plot

# Reshape to long format for faceting
real_dat_plot <- real_dat %>%
  pivot_longer(cols = c(counts, CPM), names_to = "scale_type", values_to = "expression_value")

# Create the plot
ggplot(real_dat_plot, aes(x = factor(condition),
                          y = expression_value,
                          color = factor(genotype.num))) +
  geom_point(alpha = 0.6) +
  geom_line(aes(group = donor), alpha = 0.6) +
  labs(
    title = "Interaction between Genotype and IFNg_treatment on Expression",
    x = "IFNg Treatment",
    y = "Expression Value",
    color = "Genotype"
  ) +
  facet_wrap(scale_type ~ phase, scales = "free_y") +
  theme_bw()
```


Interaction between Genotype and IFNg_treatment on Expression
counts
P1
counts
phase2
CPM
P1
CPM
phase2
Expression Value
600
400
200
0
300
200
100
0
100
50
0
100
75
50
25
0
PBS
IFNG
PBS
IFNG
IFNg Treatment
Genotype
0
1
2

Compare various model fits

Linear mixed effects (CPM) with random intercept for donor

```
# Fit the Poisson regression model
real_linear_model <- lmer(CPM ~ condition * genotype.num  +
                            (1 | donor), #random intercept for donor
                          data = real_dat)

# Clean up
real_tidy_linear <- tidy(real_linear_model)
pander(real_tidy_linear)
```

Table 1: Table continues below

effect	group	term	estimate	std.error
fixed	NA	(Intercept)	0.1691	2.876
fixed	NA	conditionIFNG	0.8466	3.287
fixed	NA	genotype.num	6.747	2.656
fixed	NA	conditionIFNG:genotype.num	32	3.047
ran_pars	donor	sd___(Intercept)	8.137	NA
ran_pars	Residual	sd___Observation	11.47	NA

statistic	df	p.value
0.05879	102.7	0.9532
0.2576	59.39	0.7976
2.54	103.7	0.01255
10.5	59.39	3.649e-15
NA	NA	NA
NA	NA	NA

Linear mixed effects (CPM) with random intercept for donor nested within phase

```
# Fit the Poisson regression model
real_linear_model <- lmer(CPM ~ condition * genotype.num  +
                            (1 | phase/donor), #random intercept for donor, nested within phase
                          data = real_dat)

# Clean up and pull the interaction term
real_tidy_linear <- tidy(real_linear_model)
pander(real_tidy_linear)
```

Table 3: Table continues below

effect	group	term	estimate	std.error
fixed	NA	(Intercept)	1.32	4.402
fixed	NA	conditionIFNG	0.8466	3.289
fixed	NA	genotype.num	6.569	2.584
fixed	NA	conditionIFNG:genotype.num	32	3.048
ran_pars	donor:phase	sd___(Intercept)	7.454	NA
ran_pars	phase	sd___(Intercept)	4.754	NA

effect	group	term	estimate	std.error
ran_pars	Residual	sd___Observation	11.48	NA

statistic	df	p.value
0.2998	2.026	0.7923
0.2574	59.22	0.7977
2.542	104.6	0.01249
10.5	59.22	3.86e-15
NA	NA	NA
NA	NA	NA
NA	NA	NA

Negative binomial mixed effects (counts) with random intercept for donor

```
# Fit the Poisson regression model
real_nb_model <- glmer.nb(counts ~ condition * genotype.num +
                                  (1 | donor),
                                data = real_dat)

# Clean up and pull the interaction term
real_tidy_nb <- tidy(real_nb_model)
pander(real_tidy_nb)
```

Table 5: Table continues below

effect	group	term	estimate	std.error
fixed	NA	(Intercept)	0.9906	0.1913
fixed	NA	conditionIFNG	1.663	0.1615
fixed	NA	genotype.num	1.676	0.1635
fixed	NA	conditionIFNG:genotype.num	0.01622	0.1341
ran_pars	donor	sd___(Intercept)	0.6431	NA

statistic	p.value
5.178	2.246e-07
10.3	6.949e-25
10.25	1.23e-24
0.1209	0.9038
NA	NA

LRT comparing nested models

```
# Fit a reduced model
real_poisson_model_red <- glmer.nb(counts ~ condition + genotype.nt + (1 | donor), #no interaction term
                           data = real_dat)

# Fit the Poisson regression model
real_poisson_model <- glmer.nb(counts ~ condition * genotype.nt + (1 | donor), #interaction term
```

```
                          data = real_dat)

# Do a likelihood ratio test
tidy_lrt <- tidy(anova(real_poisson_model_red, real_poisson_model, test = "LRT"))
tidy_lrt
```

```
## # A tibble: 2 x 9
##   term                npar   AIC   BIC logLik minus2logL statistic    df p.value
##   <chr>              <dbl> <dbl> <dbl>  <dbl>      <dbl>     <dbl> <dbl>   <dbl>
## 1 real_poisson_mode~     6 1043. 1060.  -516.      1031.    NA        NA  NA
## 2 real_poisson_model     8 1047. 1070.  -516.      1031.     0.246     2   0.884
```

Generate QQplots for Model Candidates

```
### functions
#function to permute outcome and record results for various model formulas
generate_permuted_null <- function(model_formula,
                                   data,
                                   interaction_term_regex = ":",
                                   permute_var = NULL,
                                   n_perm = 100,
                                   model_type = c("lmer", "glmer", "glmer.nb", "glm"),
                                   family = poisson) {


  model_type <- match.arg(model_type)
  model_formula_str <- paste(deparse(model_formula), collapse = " ")

  if (is.null(permute_var)) {
    outcome_var <- all.vars(model_formula)[1]
  } else {
    outcome_var <- permute_var
  }

  # Helper to fit a model and capture warnings
  fit_model_safely <- function(formula, data, model_type, family) {
    warnings <- character()

    result <- withCallingHandlers(
      tryCatch({
        fit <- switch(
          model_type,
          lmer     = lmerTest::lmer(formula, data = data),
          glmer    = glmer(formula, data = data, family = family),
          glmer.nb = glmer.nb(formula, data = data),
          glm      = glm(formula, data = data, family = family)
        )
        list(fit = fit, warnings = warnings)
      }, error = function(e) {
        list(fit = NULL, warnings = paste("Error:", conditionMessage(e)))
      }),
      warning = function(w) {
        warnings <<- c(warnings, conditionMessage(w))
        invokeRestart("muffleWarning")
      }
    )

    result$warnings <- paste(result$warnings, collapse = " | ")
    result
  }

  # Fit observed model
  obs_model_result <- fit_model_safely(model_formula, data, model_type, family)
  if (is.null(obs_model_result$fit)) return(tibble())  # skip if fail

  observed_results <- tidy(obs_model_result$fit) %>%
```

```
    filter(str_detect(term, interaction_term_regex)) %>%
    mutate(
      type = "observed",
      model_formula = model_formula_str,
      warning = obs_model_result$warnings
    )

  # Permuted results
  permuted_results <- map_dfr(1:n_perm, function(i) {
    permuted_data <- data %>%
      mutate(!!sym(outcome_var) := sample(!!sym(outcome_var)))

    perm_model_result <- fit_model_safely(model_formula, permuted_data, model_type, family)

    if (is.null(perm_model_result$fit)) return(tibble())

    tidy(perm_model_result$fit) %>%
      filter(str_detect(term, interaction_term_regex)) %>%
      mutate(
        type = "null",
        model_formula = model_formula_str,
        warning = perm_model_result$warnings
      )
  })

  bind_rows(observed_results, permuted_results)
}


plot_qq_pval_multi <- function(sim_df) {
  # Load ggrepel for smart label positioning
  library(ggrepel)

  # Ensure p-values are available and create model identifier
  sim_df <- sim_df %>%
    mutate(
      model_formula_family = str_c(model_formula, " ", paste0(family), " ", paste0(model_type))
    )

  # Calculate empirical p-values per model_formula
  empirical_pvals <- sim_df %>%
    group_by(model_formula_family) %>%
    summarise(
      observed_pval = p.value[type == "observed"],
      null_pvals = list(p.value[type == "null"]),
      empirical_pval = mean(null_pvals[[1]] <= observed_pval, na.rm = TRUE),
      .groups = "drop"
    )

  # Prepare data for QQ plot (null points only)
  # Convert p-values to -log10 scale for better visualization
  qq_df <- sim_df %>%
    filter(type == "null") %>%
```

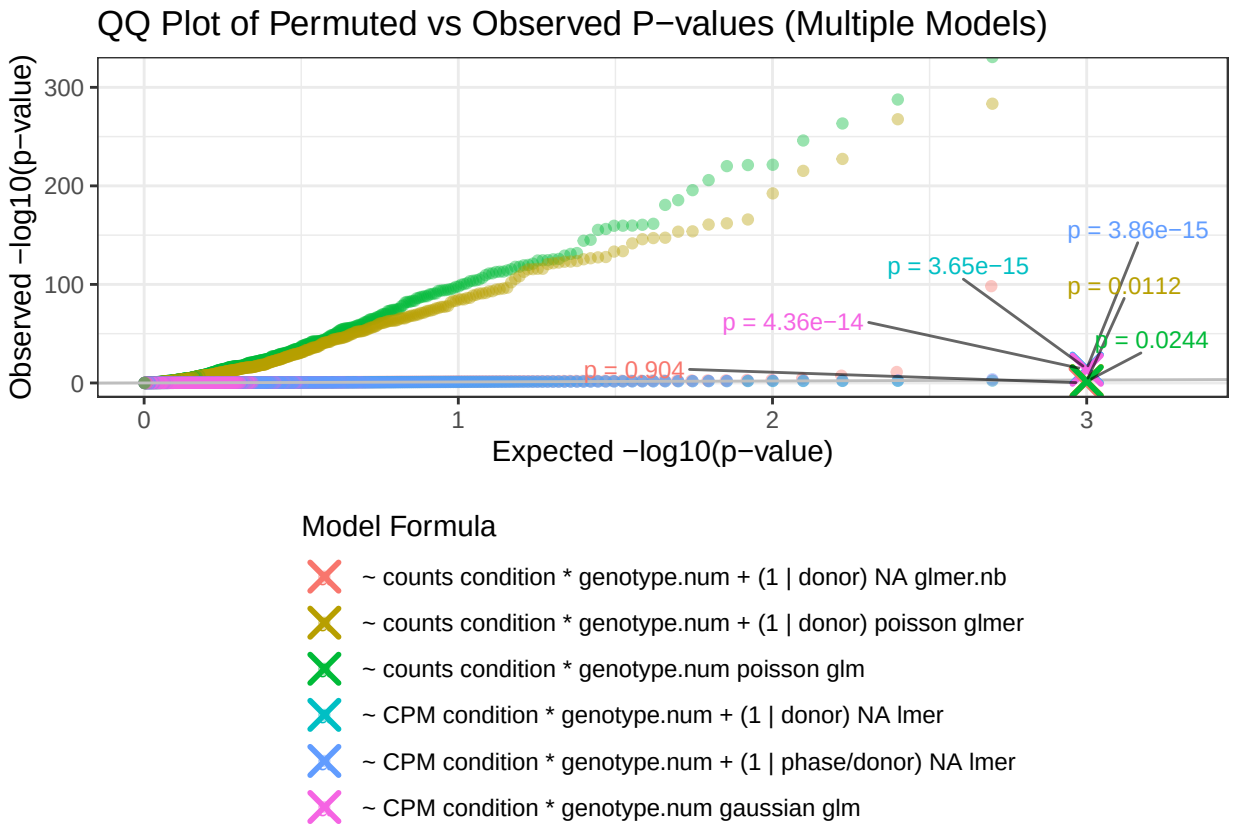
```
    group_by(model_formula_family) %>%
    arrange(p.value) %>%
    mutate(
      neg_log10_pval = -log10(p.value),
      expected_neg_log10 = -log10(ppoints(n(), a = 0))
    ) %>%
    ungroup()

  # Prepare observed points (one per model)
  observed_points <- empirical_pvals %>%
    mutate(
      observed_neg_log10 = -log10(observed_pval),
      # Expected value for the most extreme point
      expected_neg_log10 = -log10(0.5 / lengths(null_pvals)),
      label = paste0("p = ", signif(observed_pval, 3))
    )

  # Plot
  ggplot(qq_df, aes(x = expected_neg_log10, y = neg_log10_pval, color = model_formula_family)) +
    geom_point(alpha = 0.4) +
    geom_abline(slope = 1, intercept = 0, color = "gray") +
    geom_point(data = observed_points, aes(x = expected_neg_log10, y = observed_neg_log10),
               shape = 4, size = 4, stroke = 1.5) + # "X"
    geom_text_repel(data = observed_points,
                    aes(x = expected_neg_log10,
                        y = observed_neg_log10,
                        label = label,
                        color = model_formula_family),
                    size = 3,
                    min.segment.length = 0,  # Always draw connecting lines
                    segment.color = "black",
                    segment.alpha = 0.6,
                    box.padding = 0.5,       # Space around labels
                    point.padding = 0.3,     # Space around points
                    force = 2,               # Repulsion strength
                    max.overlaps = Inf) +    # Allow all labels to be shown
    scale_x_continuous(expand = expansion(mult = c(0.05, 0.15))) +
    scale_y_continuous(expand = expansion(mult = c(0.05, 0.15))) +
    labs(
      x = "Expected -log10(p-value)",
      y = "Observed -log10(p-value)",
      color = "Model Formula",
      title = "QQ Plot of Permuted vs Observed P-values (Multiple Models)"
    ) +
    theme_bw() +
    theme(
      legend.position = "bottom",
      legend.direction = "vertical"
    )
}
```

Create a grid of models and plot them together

```
# Define models we'd like to try
model_grid <- tibble(
  model_formula = list(
    as.formula("CPM ~ condition * genotype.num + (1 | donor)"),
    as.formula("CPM ~ condition * genotype.num + (1 | phase/donor)"),
    as.formula("CPM ~ condition * genotype.num"),
    as.formula("counts ~ condition * genotype.num"),
    as.formula("counts ~ condition * genotype.num + (1 | donor)"),
    as.formula("counts ~ condition * genotype.num + (1 | donor)")
  ),
  model_type = c("lmer", "lmer", "glm", "glm", "glmer", "glmer.nb"),
  family = list(NA, NA, gaussian(), poisson(), poisson(), NA)
)
```

First plot is all tested models

```
#plot the results
plot1 <- plot_qq_pval_multi(flat_qq_results)
plot1
```



Second plot is only the non-poisson models to “zoom in”

```
plot2 <- plot_qq_pval_multi(
  filter(flat_qq_results, is.na(family) | family != "poisson")
)
plot2
```