Platform SNP counts: a hierarchical model

Load and preprocess

```
meta_df <- read_tsv('../data/PGC_chip_QC_fields-good_qc_runs.tsv') %>%
  dplyr::select(c(study, ncases_postqc, ncontrols_postqc, nsnps_postqc, lambda_postqc, nsnpex_miss, nsnp
  transform(study_size = ncases_postqc + ncontrols_postqc) %>%
  replace_na(list( nsnpex_prekno = 0, nsnpex_prekno_autosomes = 0)) %>%
  mutate(nsnpex_prekno = nsnpex_prekno + nsnpex_prekno_autosomes)
```

```
## Parsed with column specification:
## cols(
##   study = col_character(),
##   `Stephan Comments` = col_logical(),
##   lambda_postqc = col_double(),
##   location = col_character(),
##   ncases_postqc = col_double(),
##   ncontrols_postqc = col_double(),
##   nsnpex_miss = col_double(),
##   nsnpex_prekno = col_double(),
##   nsnpex_prekno_autosomes = col_double(),
##   nsnps_postqc = col_double(),
##   rundate = col_character()
## )
```

```
# merge in platform annotations
plat_annot <-  read_tsv('../data/PGC_chip_QC-qc_runs_round2.tsv', col_types = cols()) %>%
  select(c(study, platform))
meta_df <- inner_join(meta_df, plat_annot, 'study')

# all Il5M are incorrect, should be GSA
meta_df <- meta_df %>%
    transform(platform = as.factor(platform))
levels(meta_df$platform) <- c(levels(meta_df$platform), 'GSA')
meta_df[which(meta_df$platform == 'Il5M'), 'platform'] <- 'GSA'
meta_df[which(meta_df$platform == 'GSAA'), 'platform'] <- 'GSA'
meta_df$platform <- droplevels(meta_df$platform)

platforms <- c("A5.0", "A6.0", "AXIO", "COEX", "I317", "I550", "I650", "Il1M", "OMEX", "P600", "PSYC",
levels(meta_df$platform) <- platforms

# all missing values are GSA as well
meta_df <- meta_df %>%
  mutate(platform = fct_explicit_na(platform, "GSA"))

variable_summaries <- meta_df %>% mutate(mean_snps = mean(nsnps_postqc), var_snps = var(nsnps_postqc)) %
  mutate(mean_snpex_miss = mean(nsnpex_miss), var_snpex_miss = var(nsnpex_miss)) %>%
  mutate(mean_study_size = mean(study_size)) %>%
  mutate(mean_lambda_postqc = mean(lambda_postqc, na.rm = T), var_lambda_postqc = var(lambda_postqc, na
```

```
  select(matches('mean_|var_')) %>%
  head(1)

# center any continuous measures of interest
data_df <- meta_df %>%
  transform(centered_study_size = study_size - mean(study_size, na.rm = T)) %>%
  transform(scaled_nsnpex_miss = scale(nsnpex_miss)) %>%
  transform(scaled_lambda_postqc = scale(lambda_postqc)) %>%
  transform(scaled_nsnps_postqc = scale(nsnps_postqc))

# remove anything with missing samples
data_df <- data_df[complete.cases(data_df),]

# remove studies on platforms with too few observations
data_df <- data_df  %>%
    group_by(platform) %>%
    filter(n()>5)

# remove runs where there are more than 100 GWS hits
data_df <- data_df %>% filter(nsnpex_prekno < 100)
```

Model fit

```
n_iter <- 10000
n_warmup <- 1000
stan_data <- list(y = data_df$nsnps_postqc,
                  X = dummy(data_df$platform, drop = F),
                  N = nrow(data_df),
                  P = length(levels(data_df$platform)),
                  N_tilde = 10)

fit <- stan(file = '../models/fit_nc_snp_counts.stan',
            data = stan_data,
            chains = 4,
            iter = n_iter,
            warmup = n_warmup)
```

```
## Trying to compile a simple C file
## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## clang -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Library/Frameworks/R.frame
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/3.5/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.5/Resources/library/RcppEigen/inclu
## In file included from /Library/Frameworks/R.framework/Versions/3.5/Resources/library/RcppEigen/inclu
## /Library/Frameworks/R.framework/Versions/3.5/Resources/library/RcppEigen/include/Eigen/src/Core/util/
## namespace Eigen {
## ^
## /Library/Frameworks/R.framework/Versions/3.5/Resources/library/RcppEigen/include/Eigen/src/Core/util/
## namespace Eigen {
##                 ^
##                 ;
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/3.5/Resources/library/StanHeaders/incl
```

```
## In file included from /Library/Frameworks/R.framework/Versions/3.5/Resources/library/RcppEigen/inclu
## /Library/Frameworks/R.framework/Versions/3.5/Resources/library/RcppEigen/include/Eigen/Core:96:10: fa
## #include <complex>
##          ^~~~~~~~
## 3 errors generated.
## make: *** [foo.o] Error 1
##
## SAMPLING FOR MODEL 'fit_nc_snp_counts' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000185 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.85 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 10000 [  0%]  (Warmup)
## Chain 1: Iteration: 1000 / 10000 [ 10%]  (Warmup)
## Chain 1: Iteration: 1001 / 10000 [ 10%]  (Sampling)
## Chain 1: Iteration: 2000 / 10000 [ 20%]  (Sampling)
## Chain 1: Iteration: 3000 / 10000 [ 30%]  (Sampling)
## Chain 1: Iteration: 4000 / 10000 [ 40%]  (Sampling)
## Chain 1: Iteration: 5000 / 10000 [ 50%]  (Sampling)
## Chain 1: Iteration: 6000 / 10000 [ 60%]  (Sampling)
## Chain 1: Iteration: 7000 / 10000 [ 70%]  (Sampling)
## Chain 1: Iteration: 8000 / 10000 [ 80%]  (Sampling)
## Chain 1: Iteration: 9000 / 10000 [ 90%]  (Sampling)
## Chain 1: Iteration: 10000 / 10000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 2.90334 seconds (Warm-up)
## Chain 1:                21.5075 seconds (Sampling)
## Chain 1:                24.4108 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'fit_nc_snp_counts' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 6.9e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.69 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 10000 [  0%]  (Warmup)
## Chain 2: Iteration: 1000 / 10000 [ 10%]  (Warmup)
## Chain 2: Iteration: 1001 / 10000 [ 10%]  (Sampling)
## Chain 2: Iteration: 2000 / 10000 [ 20%]  (Sampling)
## Chain 2: Iteration: 3000 / 10000 [ 30%]  (Sampling)
## Chain 2: Iteration: 4000 / 10000 [ 40%]  (Sampling)
## Chain 2: Iteration: 5000 / 10000 [ 50%]  (Sampling)
## Chain 2: Iteration: 6000 / 10000 [ 60%]  (Sampling)
## Chain 2: Iteration: 7000 / 10000 [ 70%]  (Sampling)
## Chain 2: Iteration: 8000 / 10000 [ 80%]  (Sampling)
## Chain 2: Iteration: 9000 / 10000 [ 90%]  (Sampling)
## Chain 2: Iteration: 10000 / 10000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 2.35513 seconds (Warm-up)
## Chain 2:                20.6002 seconds (Sampling)
```

```
## Chain 2:                22.9553 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'fit_nc_snp_counts' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 5.2e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.52 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 10000 [  0%]  (Warmup)
## Chain 3: Iteration: 1000 / 10000 [ 10%]  (Warmup)
## Chain 3: Iteration: 1001 / 10000 [ 10%]  (Sampling)
## Chain 3: Iteration: 2000 / 10000 [ 20%]  (Sampling)
## Chain 3: Iteration: 3000 / 10000 [ 30%]  (Sampling)
## Chain 3: Iteration: 4000 / 10000 [ 40%]  (Sampling)
## Chain 3: Iteration: 5000 / 10000 [ 50%]  (Sampling)
## Chain 3: Iteration: 6000 / 10000 [ 60%]  (Sampling)
## Chain 3: Iteration: 7000 / 10000 [ 70%]  (Sampling)
## Chain 3: Iteration: 8000 / 10000 [ 80%]  (Sampling)
## Chain 3: Iteration: 9000 / 10000 [ 90%]  (Sampling)
## Chain 3: Iteration: 10000 / 10000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 2.91581 seconds (Warm-up)
## Chain 3:                20.9978 seconds (Sampling)
## Chain 3:                23.9136 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'fit_nc_snp_counts' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 6.9e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.69 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 10000 [  0%]  (Warmup)
## Chain 4: Iteration: 1000 / 10000 [ 10%]  (Warmup)
## Chain 4: Iteration: 1001 / 10000 [ 10%]  (Sampling)
## Chain 4: Iteration: 2000 / 10000 [ 20%]  (Sampling)
## Chain 4: Iteration: 3000 / 10000 [ 30%]  (Sampling)
## Chain 4: Iteration: 4000 / 10000 [ 40%]  (Sampling)
## Chain 4: Iteration: 5000 / 10000 [ 50%]  (Sampling)
## Chain 4: Iteration: 6000 / 10000 [ 60%]  (Sampling)
## Chain 4: Iteration: 7000 / 10000 [ 70%]  (Sampling)
## Chain 4: Iteration: 8000 / 10000 [ 80%]  (Sampling)
## Chain 4: Iteration: 9000 / 10000 [ 90%]  (Sampling)
## Chain 4: Iteration: 10000 / 10000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 2.8251 seconds (Warm-up)
## Chain 4:                19.642 seconds (Sampling)
## Chain 4:                22.4671 seconds (Total)
## Chain 4:

## Warning: There were 22 divergent transitions after warmup. See
```

```
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
## Warning: The largest R-hat is 1.73, indicating chains have not mixed.
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#r-hat
```

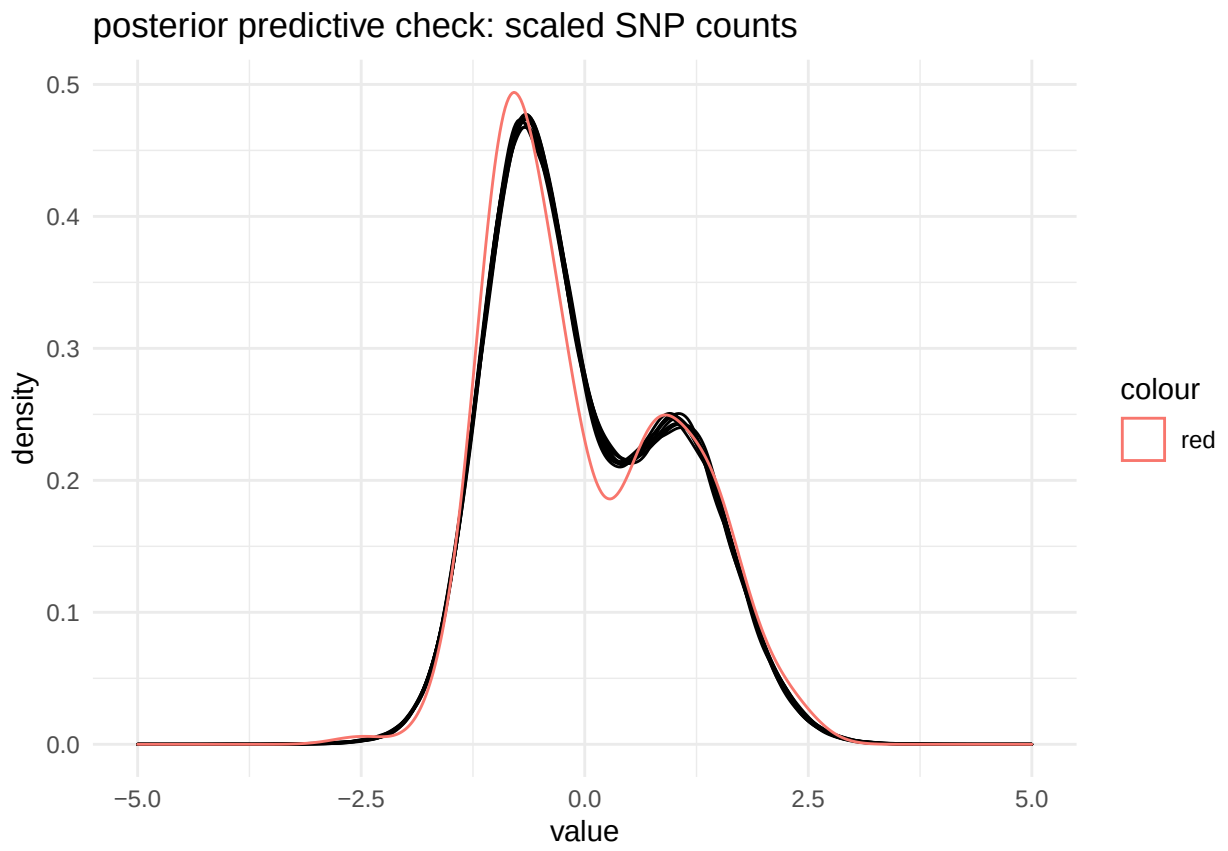
```
## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#bulk-ess
```

```
## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quanti
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#tail-ess
```

Model Checks

```
y_tilde <- rstan::extract(fit) %>%
  pluck('y_tilde') %>% c() %>%
  as_tibble() %>%
  mutate(iter = rep(1:nrow(data_df), 4* (n_iter - n_warmup)))

ppc_plt <- y_tilde %>% filter(iter <= 10) %>%
  ggplot(aes(x=value)) +
    geom_density(aes(group = iter)) +
    geom_density(data = data_df, aes(x = scale(nsnps_postqc), color = 'red')) +
    xlim(c(-5, 5)) +
    ggtitle('posterior predictive check: scaled SNP counts') +
    theme_minimal()
ppc_plt
```

```
## Warning: Removed 1 rows containing non-finite values (stat_density).
```



Evaluate

Hypothetical observation of a dataset typed on Psych-chip, with 500k SNPs observed.

```
platform <- 'PSYC'
platform_onehot <- dummy(factor(c(platform), levels = platforms), drop = FALSE)
snp_count <- 500000

new_data <- list(X = platform_onehot %>% c(),
                 y = (snp_count - mean(data_df$nsnps_postqc)) / sd(data_df$nsnps_postqc),
                 P = length(platforms),
                 N_draws = 4 * (n_iter - n_warmup),
                 mu = rstan::extract(fit) %>% pluck('mu') %>% unlist(),
                 sigma = rstan::extract(fit) %>% pluck('sigma') %>% unlist())
predictions <- stan(file = '../models/predict_snp_counts.stan',
            data = new_data,
            algorithm = 'Fixed_param',
            iter = 1,
            chains = 1)
```

```
## Trying to compile a simple C file
```

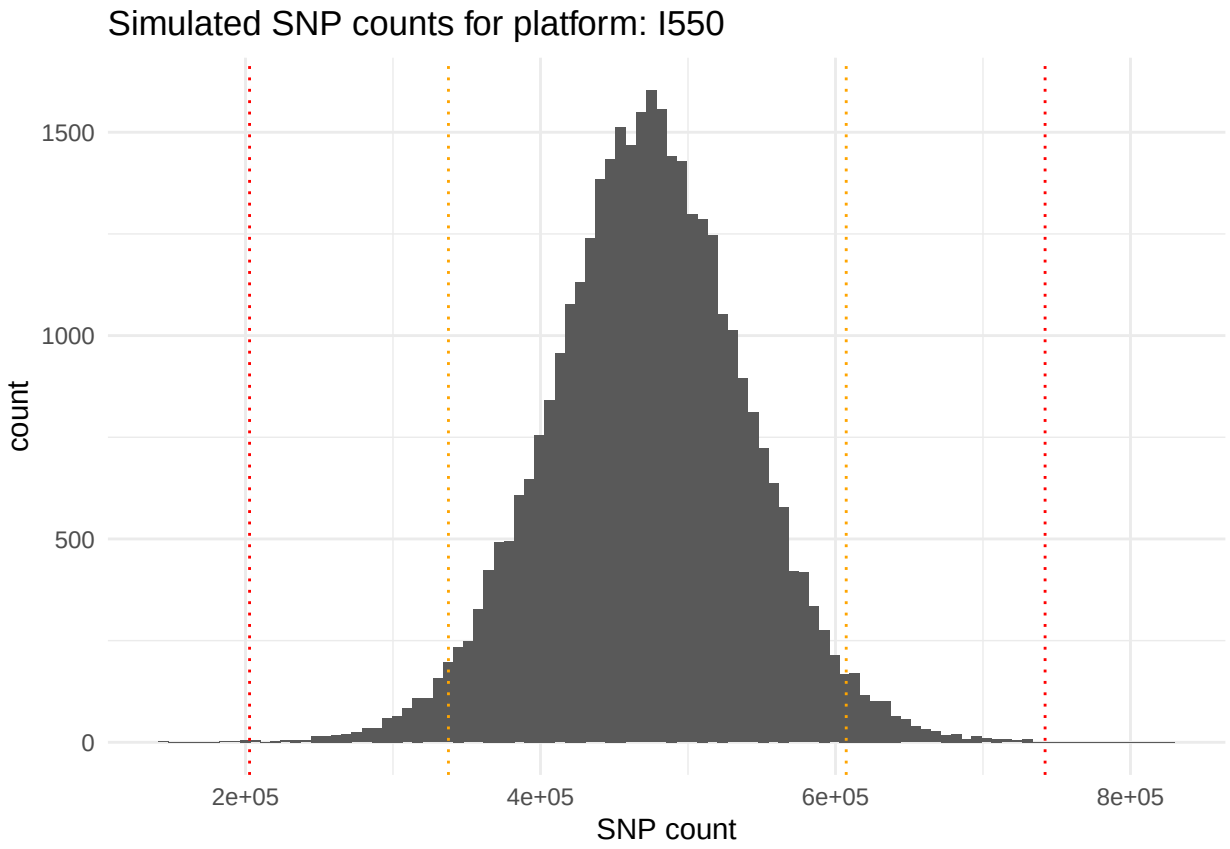
```
## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## clang -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Library/Frameworks/R.frame
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/3.5/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.5/Resources/library/RcppEigen/inclu
```

```
## In file included from /Library/Frameworks/R.framework/Versions/3.5/Resources/library/RcppEigen/inclu
## /Library/Frameworks/R.framework/Versions/3.5/Resources/library/RcppEigen/include/Eigen/src/Core/util/
## namespace Eigen {
## ^
## /Library/Frameworks/R.framework/Versions/3.5/Resources/library/RcppEigen/include/Eigen/src/Core/util/
## namespace Eigen {
##                 ^
##                 ;
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/3.5/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.5/Resources/library/RcppEigen/inclu
## /Library/Frameworks/R.framework/Versions/3.5/Resources/library/RcppEigen/include/Eigen/Core:96:10: fa
## #include <complex>
##          ^~~~~~~~
## 3 errors generated.
## make: *** [foo.o] Error 1
##
## SAMPLING FOR MODEL 'predict_snp_counts' NOW (CHAIN 1).
## Chain 1: Iteration: 1 / 1 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                0.029004 seconds (Sampling)
## Chain 1:                0.029004 seconds (Total)
## Chain 1:
```

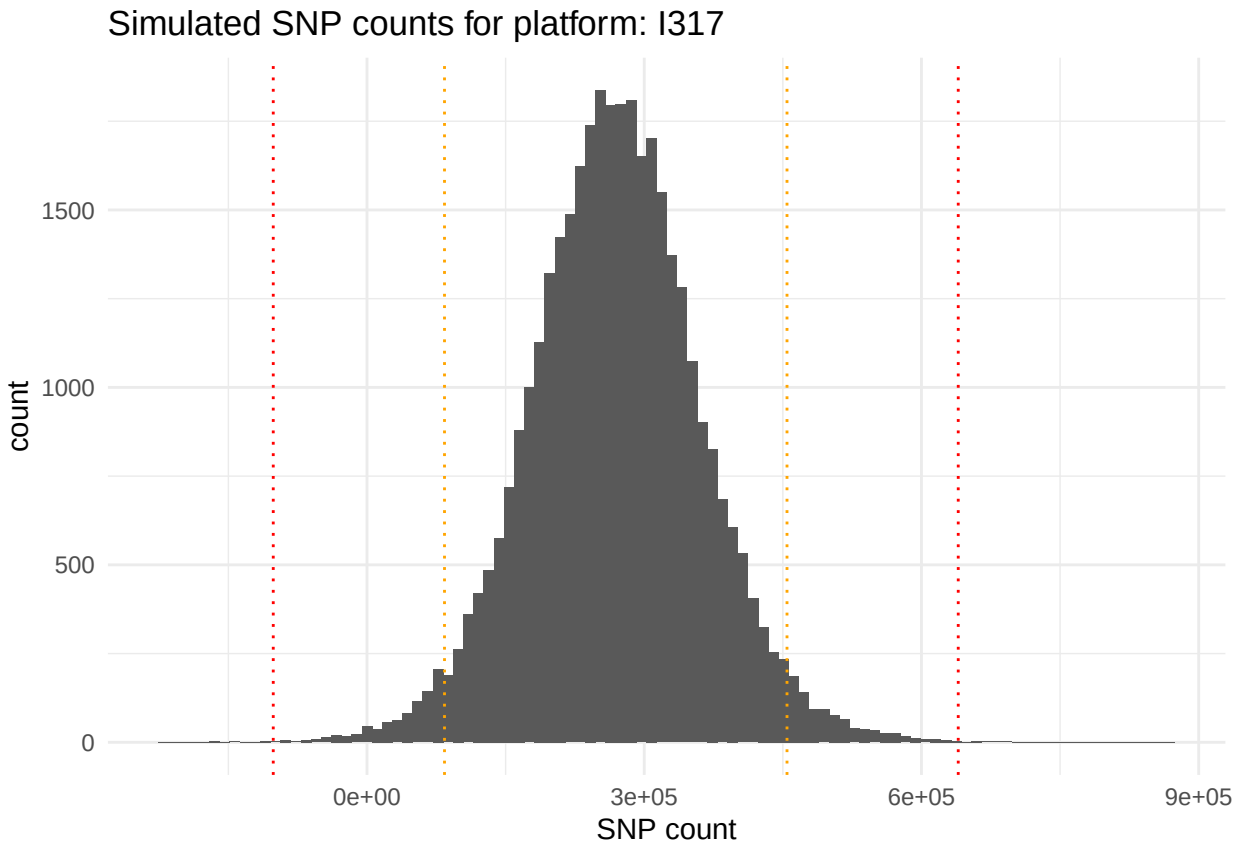
```
emp_pval <- function(generated_y, observed_y, one_sided = TRUE) {

  prop <- ifelse(observed_y <= mean(generated_y),
                 length(which(generated_y >= observed_y)) / length(generated_y),
                 length(which(generated_y <= observed_y)) / length(generated_y))

  pval <- ifelse(one_sided,
                 prop,
                 prop / 2.0)

  return(1.0 - pval)
}

sd_from_mean <- function(generated_y, observed_y) {
  (abs(mean(generated_y) - observed_y)) / sd(generated_y)
}


plts <- list()
i <- 1
for (plat in unique(data_df$platform)) {

  platform_onehot <- dummy(factor(c(plat), levels = platforms), drop = FALSE)
  snp_count <- 500000

  new_data <- list(X = platform_onehot %>% c(),
                   y = (snp_count - mean(data_df$nsnps_postqc)) / sd(data_df$nsnps_postqc),
                   P = length(platforms),
                   N_draws = 4 * ( n_iter - n_warmup),
```

```
                    mu = rstan::extract(fit) %>% pluck('mu') %>% unlist(),
                    sigma = rstan::extract(fit) %>% pluck('sigma') %>% unlist())
  predictions <- stan(file = '../models/predict_snp_counts.stan',
              data = new_data,
              algorithm = 'Fixed_param',
              iter = 1,
              chains = 1)

  eval_metrics <- tibble(y_tilde = rstan::extract(predictions) %>% pluck('y_tilde') %>% c(),
                         y_loglik = rstan::extract(predictions) %>% pluck('y_loglik') %>% c()) %>%
                  mutate(y_tilde = y_tilde * sd(data_df$nsnps_postqc)) + mean(data_df$nsnps_postqc)


  generative_sds <- c(-4, -2, 2, 4) %>%
    map(function(x) mean(eval_metrics$y_tilde) - x * sd(eval_metrics$y_tilde)) %>% unlist() %>%
    as_tibble()

  plts[[i]] <- ggplot(eval_metrics, aes(x = y_tilde)) +
    geom_histogram(bins = 100) +
    geom_vline(data = generative_sds, aes(xintercept = value), color = c('red', 'orange', 'orange', 'red
    ggtitle(paste0('Simulated SNP counts for platform: ', plat)) +
    xlab('SNP count') +
    theme_minimal()

  print(plts[[i]])
  i <- i + 1

}
```

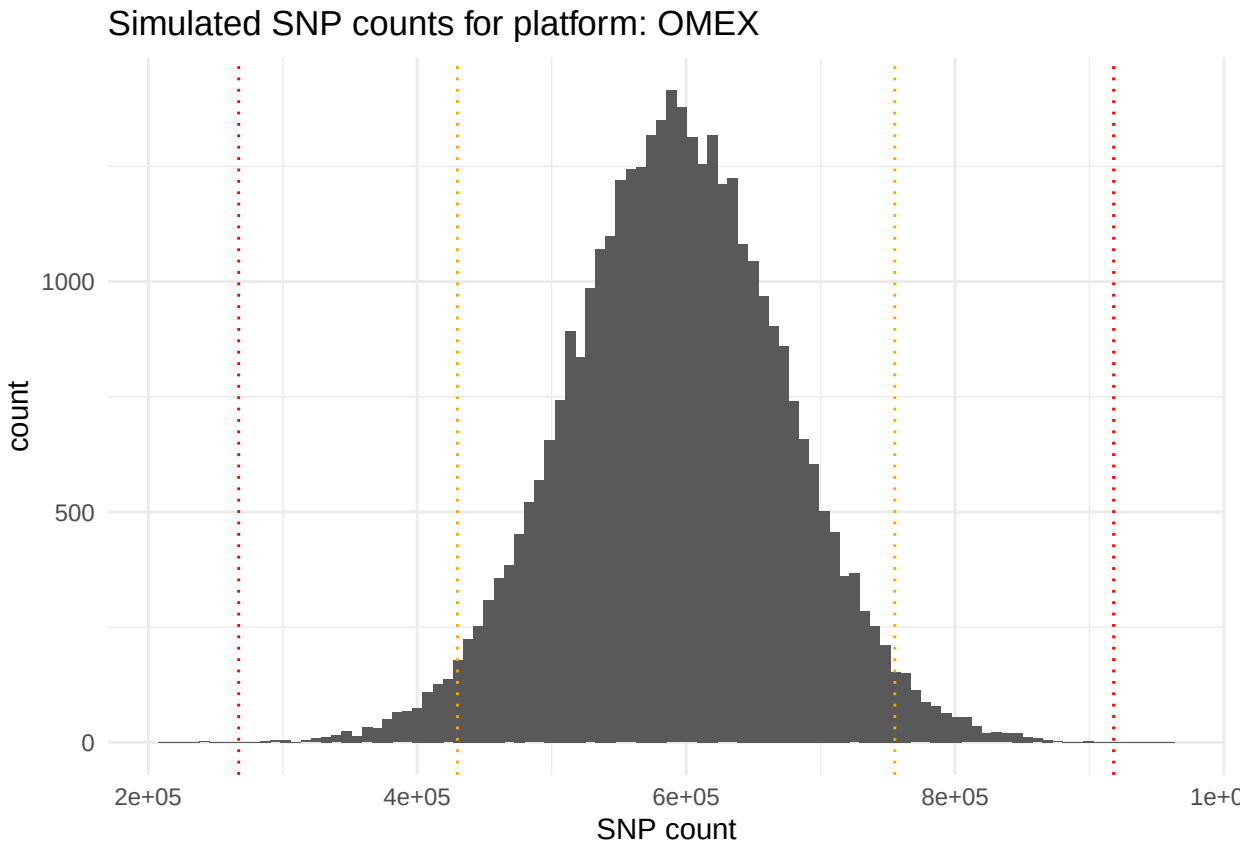
```
##
## SAMPLING FOR MODEL 'predict_snp_counts' NOW (CHAIN 1).
## Chain 1: Iteration: 1 / 1 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                0.036299 seconds (Sampling)
## Chain 1:                0.036299 seconds (Total)
## Chain 1:
```



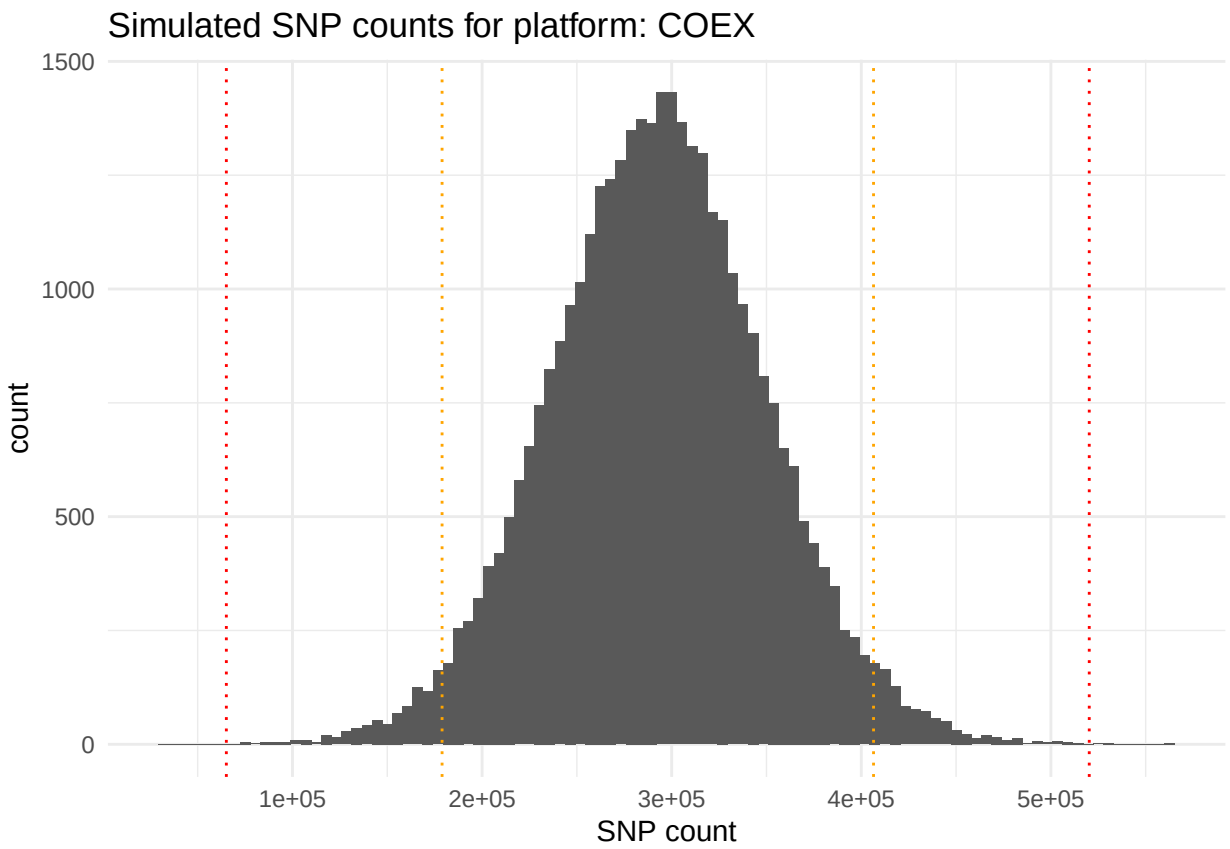

```
##
## SAMPLING FOR MODEL 'predict_snp_counts' NOW (CHAIN 1).
## Chain 1: Iteration: 1 / 1 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                0.026905 seconds (Sampling)
## Chain 1:                0.026905 seconds (Total)
## Chain 1:
```



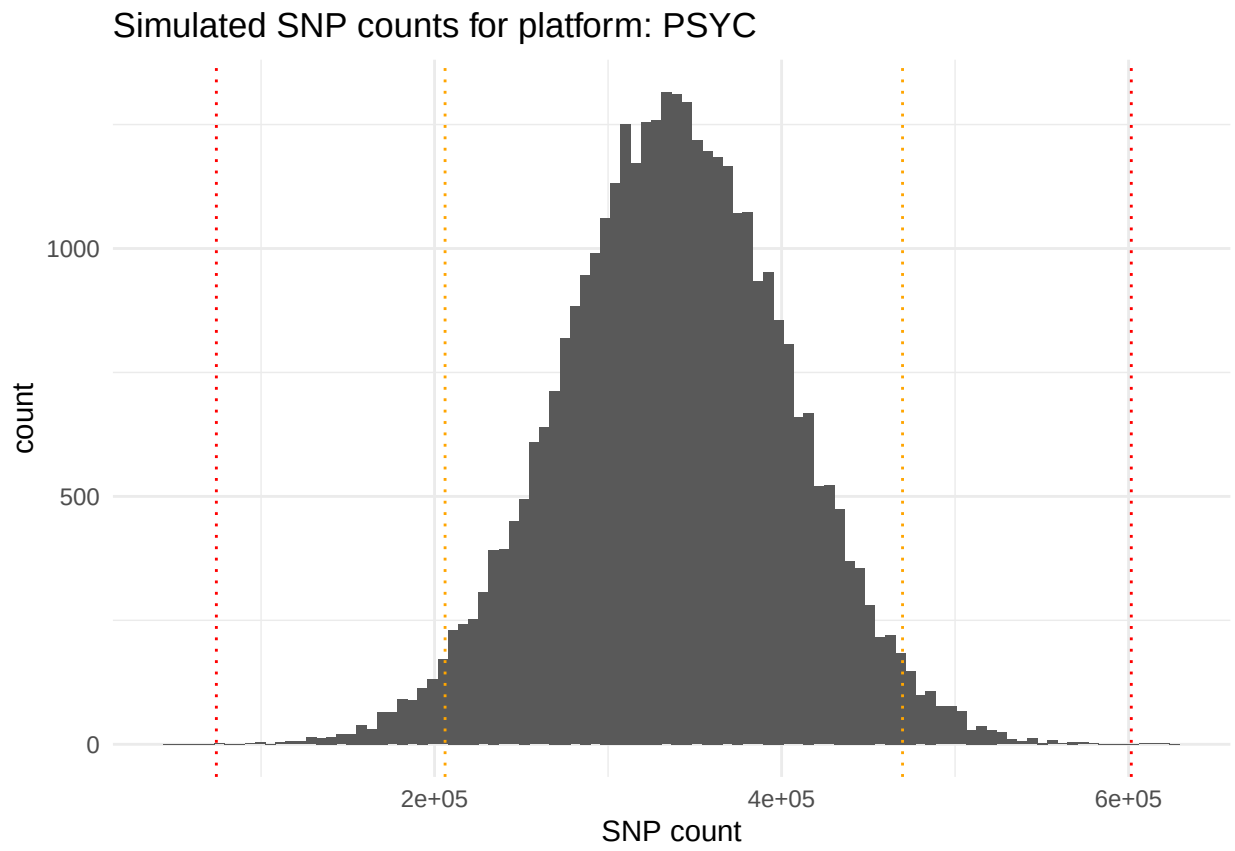

```
##
## SAMPLING FOR MODEL 'predict_snp_counts' NOW (CHAIN 1).
## Chain 1: Iteration: 1 / 1 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                0.02735 seconds (Sampling)
## Chain 1:                0.02735 seconds (Total)
## Chain 1:
```



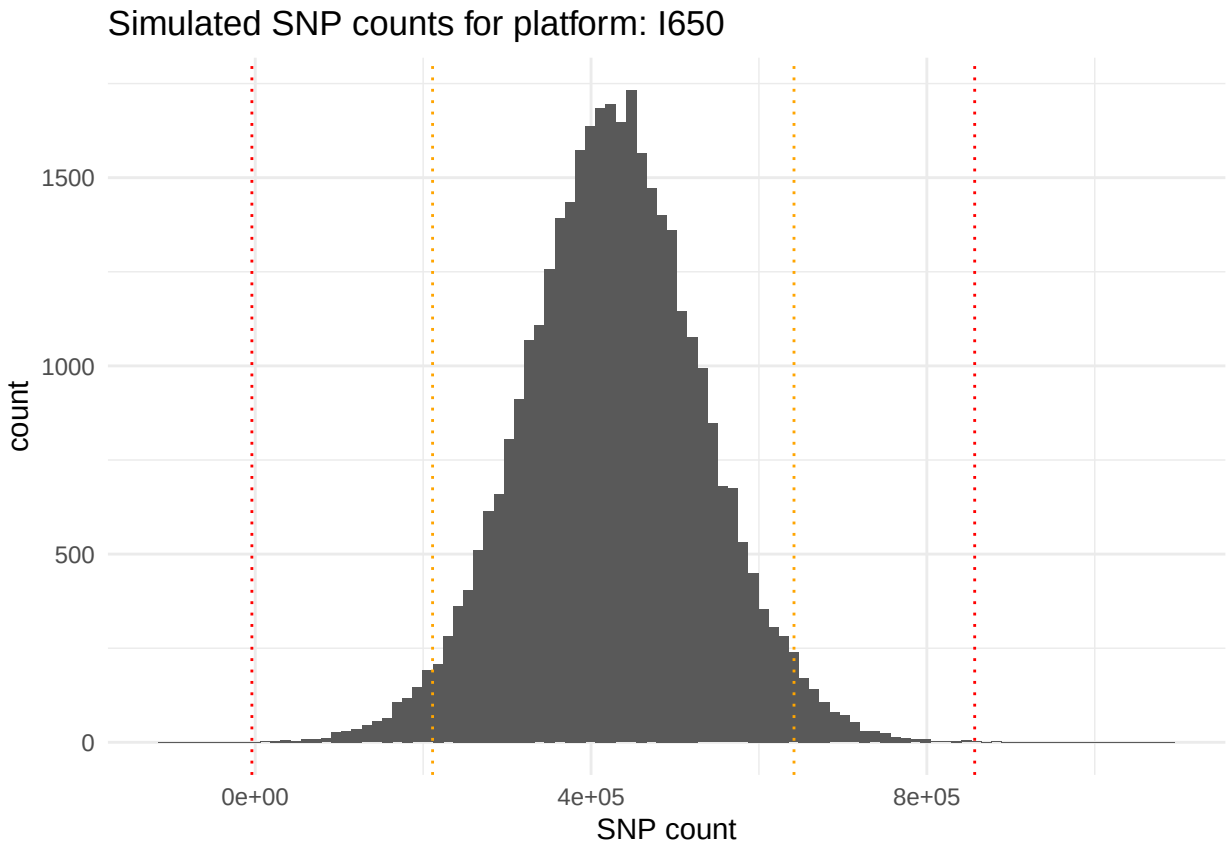

```
##
## SAMPLING FOR MODEL 'predict_snp_counts' NOW (CHAIN 1).
## Chain 1: Iteration: 1 / 1 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                0.029134 seconds (Sampling)
## Chain 1:                0.029134 seconds (Total)
## Chain 1:
```



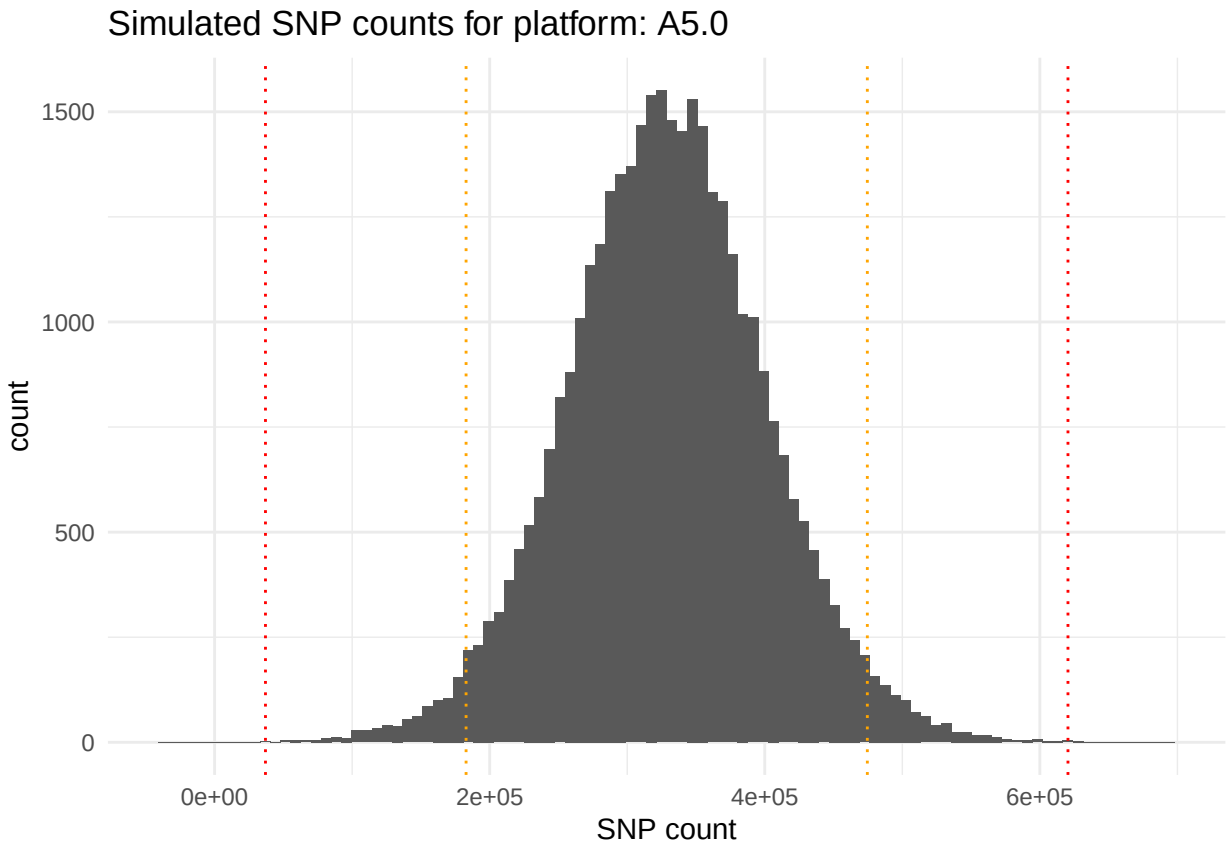

```
##
## SAMPLING FOR MODEL 'predict_snp_counts' NOW (CHAIN 1).
## Chain 1: Iteration: 1 / 1 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                0.030426 seconds (Sampling)
## Chain 1:                0.030426 seconds (Total)
## Chain 1:
```



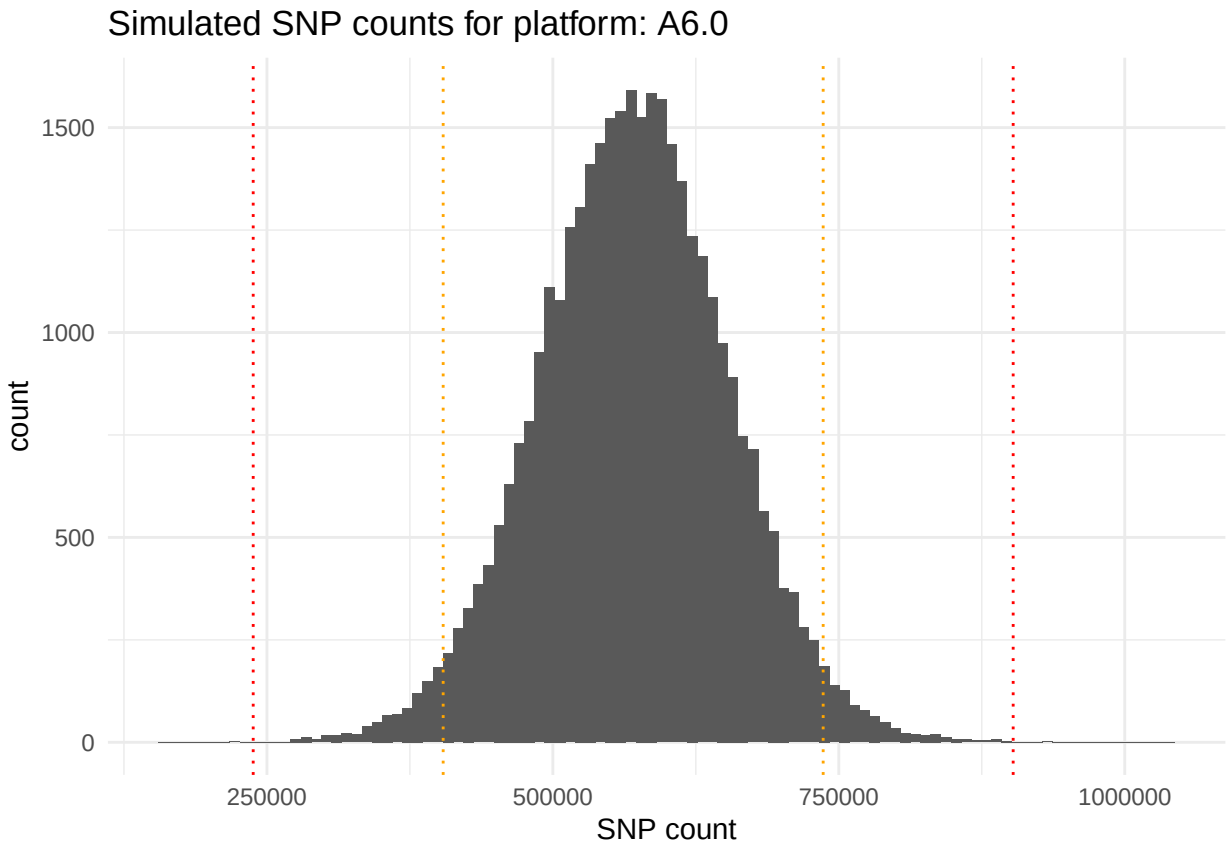

```
##
## SAMPLING FOR MODEL 'predict_snp_counts' NOW (CHAIN 1).
## Chain 1: Iteration: 1 / 1 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                0.030663 seconds (Sampling)
## Chain 1:                0.030663 seconds (Total)
## Chain 1:
```



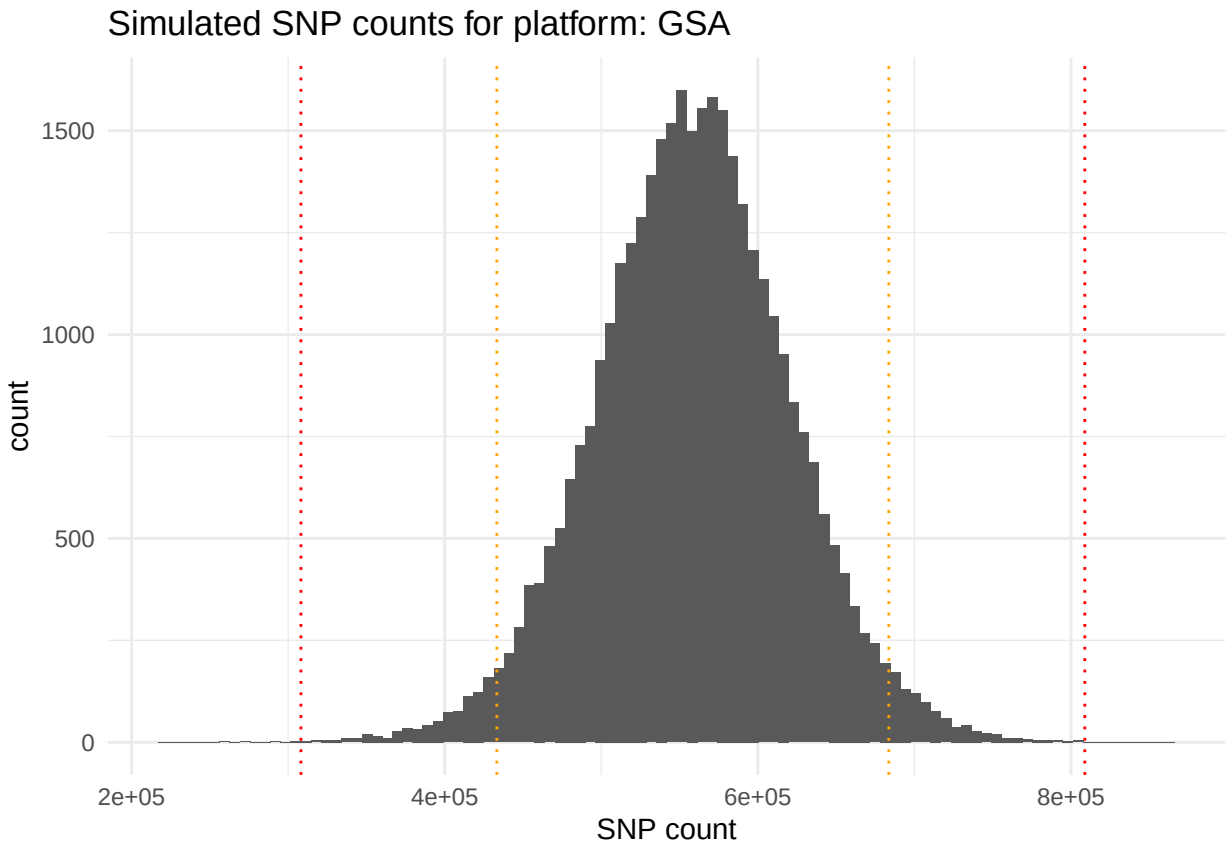

```
##
## SAMPLING FOR MODEL 'predict_snp_counts' NOW (CHAIN 1).
## Chain 1: Iteration: 1 / 1 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                0.029312 seconds (Sampling)
## Chain 1:                0.029312 seconds (Total)
## Chain 1:
```




```
##
## SAMPLING FOR MODEL 'predict_snp_counts' NOW (CHAIN 1).
## Chain 1: Iteration: 1 / 1 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                0.029924 seconds (Sampling)
## Chain 1:                0.029924 seconds (Total)
## Chain 1:
```




```
##
## SAMPLING FOR MODEL 'predict_snp_counts' NOW (CHAIN 1).
## Chain 1: Iteration: 1 / 1 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                0.027824 seconds (Sampling)
## Chain 1:                0.027824 seconds (Total)
## Chain 1:
```


Simulated SNP counts for platform: GSA
1500
1000
500
0
count
2e+05
4e+05
6e+05
8e+05
SNP count