Pooled Analysis

Aimee Taylor and James Watson

Preamble

Load R packages, functions and data.

```
#============================================================
# First let's extract some summaries and format the data frames
# ahead of summaries and analyses
#============================================================
total_patient_names = unique(Combined_Time_Data$patientid)
total_patient_count = length(total_patient_names)

# Add a variable with trial, drug arm (if VHX) and partner (if BPD - since drug arm was labelled CQ for
ind_VHX_ctd = grepl('VHX', Combined_Time_Data$patientid)
Combined_Time_Data$trial_arm_partner = NA
Combined_Time_Data$trial_arm_partner[ind_VHX_ctd] = paste('VHX', Combined_Time_Data$arm_num[ind_VHX_ctd]
Combined_Time_Data$trial_arm_partner[!ind_VHX_ctd] = paste('BPD', Combined_Time_Data$PMQ_partner[!ind_VH

# Note that Combined_Time_Data has censored rows for patients
Combined_Time_Data_no_cnsd_rows = Combined_Time_Data[Combined_Time_Data$Censored != 1,] # remove censor
uncensored_patientid_vector = Combined_Time_Data_no_cnsd_rows$patientid # vector of patientids excludin

# names of patients who recurred
recur_patient_names = names(which(table(uncensored_patientid_vector) > 1))

# Treatment of all those who recurred
recur_patient_treatment = sapply(recur_patient_names, function(id){
  inds = Combined_Time_Data$patientid == id
  treatment = unique(Combined_Time_Data$trial_arm_partner[inds])
  if(length(treatment) >1){stop('More than one treatment')}else{treatment}
  })

# Number of episodes per persons
All_VHX_epi_count = table(uncensored_patientid_vector[grepl('VHX_',uncensored_patientid_vector)])
All_BPD_epi_count = table(uncensored_patientid_vector[grepl('BPD_',uncensored_patientid_vector)])

# Add trial, drug arm (if VHX) and partner (if BPD - see above) to genetic data
MS_pooled$trial_arm_partner = sapply(1:nrow(MS_pooled), function(i){unique(Combined_Time_Data$trial_arm_

# Collapse rows due to complex infections (COI > 2)
MS_pooled_summary = MS_pooled[!duplicated(MS_pooled$Episode_Identifier),]
```

Summary of the clinical trial data (features in Table 1 of the main text)

```
##
## Number of patients: 1299 (644 in VHX; 655 in BPD)

##
```

```
## Number of patients by treatment:
##
##      VHX AS     VHX CHQ VHX CHQ/PMQ     BPD CHQ      BPD DP
##         224         222         198         329         326
##
## Number of patients who recurred by treatment, Nr:
## recur_patient_treatment
##      VHX AS     VHX CHQ VHX CHQ/PMQ     BPD CHQ      BPD DP
##         177         165          35          47          34
##
## Percent of N patients who recurred by treatment:
## recur_patient_treatment
##      VHX AS     VHX CHQ VHX CHQ/PMQ     BPD CHQ      BPD DP
##          39          36           8          10           7
##
## Number of individuals with at least one episode typed by treatment:
##
##      VHX AS     VHX CHQ VHX CHQ/PMQ     BPD CHQ      BPD DP
##          13          90          34          46          34
##
## Percent of Nr individuals with at least one episode typed by treatment:
##
##      VHX AS     VHX CHQ VHX CHQ/PMQ     BPD CHQ      BPD DP
##           7          55          97          98         100
##
## Number of individuals with at least one recurrence typed by treatment:
##
##      VHX AS     VHX CHQ VHX CHQ/PMQ     BPD CHQ      BPD DP
##          13          90          32          46          32
##
## Percent of Nr individuals with at least one recurrence typed by treatment:
##
##      VHX AS     VHX CHQ VHX CHQ/PMQ     BPD CHQ      BPD DP
##           7          55          91          98          94
##
## Number of recurrences by treatment:
##
##      VHX AS     VHX CHQ VHX CHQ/PMQ     BPD CHQ      BPD DP
##         722         587          40          53          39
##
## Number of recurrences typed by treatment:
##
##      VHX AS     VHX CHQ VHX CHQ/PMQ     BPD CHQ      BPD DP
##          13         358          35          52          36
```

```
##
## Percent of R recurrences typed by treatment:
```

```
##
##      VHX AS     VHX CHQ VHX CHQ/PMQ     BPD CHQ      BPD DP
##           2          61          88          98          92
```

```
##
##
## From BPD trial there are 80 individuals with total of 167 episodes typed (enrollment: 79; recurrent 8
```

```
## From VHX trial there are 137 individuals with total of 543 episodes typed (enrollment: 137; recurren
```

Next we summarise the number of episodes typed in people with one or more episodes typed

```
#=================================================================
# Number of episodes typed conditional on a person being selected for genotyping
#=================================================================
# No. of typed episodes per person with one or more typed episodes in VHX and BPD
no_of_typed_epi_per_person_typed_VHX = table(MS_pooled_summary$ID[grepl('VHX',MS_pooled_summary$ID)])
no_of_typed_epi_per_person_typed_BPD = table(MS_pooled_summary$ID[grepl('BPD',MS_pooled_summary$ID)])

# No. of total episodes per person with one or more typed episodes in VHX and BPD
no_of_epi_per_person_typed_VHX = All_VHX_epi_count[names(All_VHX_epi_count) %in% names(no_of_typed_epi_p
no_of_epi_per_person_typed_BPD = All_BPD_epi_count[names(All_BPD_epi_count) %in% names(no_of_typed_epi_p

#-------------------------
# VHX data set summary: brief because genotyping VHX was not exhaustive
#-------------------------
X0 = length(no_of_typed_epi_per_person_typed_VHX) # Number of people typed
ind_untyped = no_of_epi_per_person_typed_VHX != no_of_typed_epi_per_person_typed_VHX
X1 = sum(ind_untyped) # Number of people selected for genotyping but some episodes untyped
# How many untyped per person with incomplete set of episodes typed:
X2 = range(no_of_epi_per_person_typed_VHX[ind_untyped] - no_of_typed_epi_per_person_typed_VHX[ind_untype
X3 = sum(no_of_epi_per_person_typed_VHX - no_of_typed_epi_per_person_typed_VHX) # Total number untyped
writeLines(sprintf('\nVHX: for %s of %s VHX individual/s selected for genotyping: %s to %s of their epis
```

```
##
## VHX: for 27 of 137 VHX individual/s selected for genotyping: 1 to 7 of their episodes were not typed
```

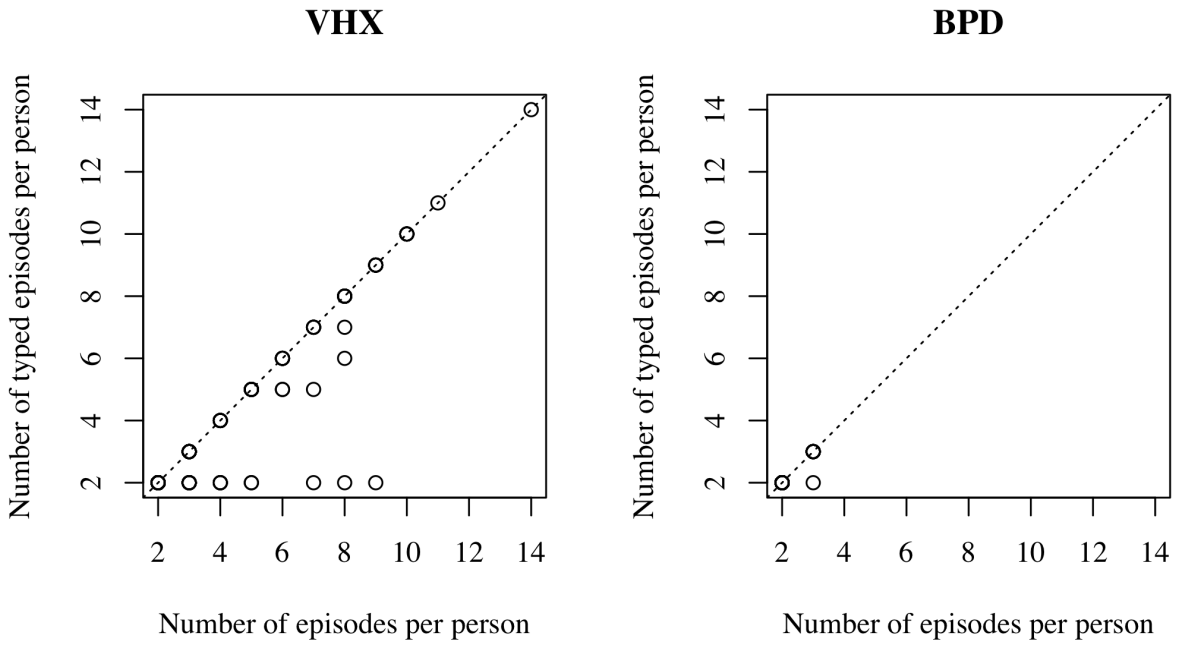
```
# How about for those who received PMQ in VHX?
no_of_typed_epi_per_person_typed_VHX_PMQ = table(MS_pooled_summary$ID[MS_pooled_summary$trial_arm_partne
no_of_epi_per_person_typed_VHX_PMQ = All_VHX_epi_count[names(All_VHX_epi_count) %in% names(no_of_typed_e
X0 = length(no_of_typed_epi_per_person_typed_VHX_PMQ) # Number of people typed
ind_untyped = no_of_epi_per_person_typed_VHX_PMQ != no_of_typed_epi_per_person_typed_VHX_PMQ
X1 = sum(ind_untyped) # Number of people selected for genotyping but some episodes untyped
# How many untyped per person with incomplete set of episodes typed:
X2 = range(no_of_epi_per_person_typed_VHX_PMQ[ind_untyped] - no_of_typed_epi_per_person_typed_VHX_PMQ[in
X3 = sum(no_of_epi_per_person_typed_VHX_PMQ - no_of_typed_epi_per_person_typed_VHX_PMQ) # Total number
writeLines(sprintf('\nVHX: for %s of %s PMQ+ treated VHX individual/s selected for genotyping: %s to %s
```

```
##
## VHX: for 4 of 34 PMQ+ treated VHX individual/s selected for genotyping: 1 to 1 of their episodes were
```

```
PMQ_treated_VHX_ids = names(no_of_typed_epi_per_person_typed_VHX_PMQ)

#-------------------------
# BPD data set: comprehensive because genotyping BPD was exhaustive
#-------------------------
# How many people who experience one or more recurrences had one or more episodes genotyped?
recurrences = All_BPD_epi_count[All_BPD_epi_count > 1]-1
indivs_who_recurred = names(recurrences)
indivs_who_were_typed = names(no_of_typed_epi_per_person_typed_BPD)
indivs_who_were_not_typed = indivs_who_recurred[!indivs_who_recurred %in% indivs_who_were_typed]

# Summary over individuals typed
X0 = length(indivs_who_were_typed) # Number of people typed
ind_untyped = no_of_epi_per_person_typed_BPD != no_of_typed_epi_per_person_typed_BPD
X1 = sum(ind_untyped) # Number of episodes untyped
# How many untyped per person with incomplete set of episodes typed:
X2 = range(no_of_epi_per_person_typed_BPD[ind_untyped] - no_of_typed_epi_per_person_typed_BPD[ind_untype
X3 = sum(no_of_epi_per_person_typed_BPD - no_of_typed_epi_per_person_typed_BPD)  # Total number untyped

# Individuals with not all episodes typed
ind_missing_typed_epi <- names(which(no_of_epi_per_person_typed_BPD != no_of_typed_epi_per_person_typed_

# All episodes of the BPD individuals missing one or episodes
X4 = lapply(ind_missing_typed_epi, function(x){
  ind = grepl(x, uncensored_patientid_vector)
  Combined_Time_Data_no_cnsd_rows$episode[ind]
})

# Typed episodes of the BPD individuals missing one or episodes
X5 = lapply(ind_missing_typed_epi, function(x){
  ind = grepl(x, MS_pooled_summary$ID)
  MS_pooled_summary$Episode[ind]
})

X6 = lapply(1:length(X5), function(i){setdiff(X4[[i]], X5[[i]])}) # Not typed episodes
X7 = sum(sapply(X6, function(x)sum(x>1))) # Not typed recurrence

writeLines(paste(sprintf('\nBPD: of %s of the people who recurred: %s person/people with %s recurrence/
                         length(unique(indivs_who_recurred)),
                         length(unique(indivs_who_were_not_typed)),
                         recurrences[indivs_who_were_not_typed]),
                 sprintf('Of %s of %s BPD individual/s selected for genotyping: %s to %s of their episo
                 sprintf('Of the %s episodes not typed %s were recurrences.',X3, X7),
                 sprintf('In total there were %s recurrences: %s untyped.', sum(recurrences), recurrenc
```

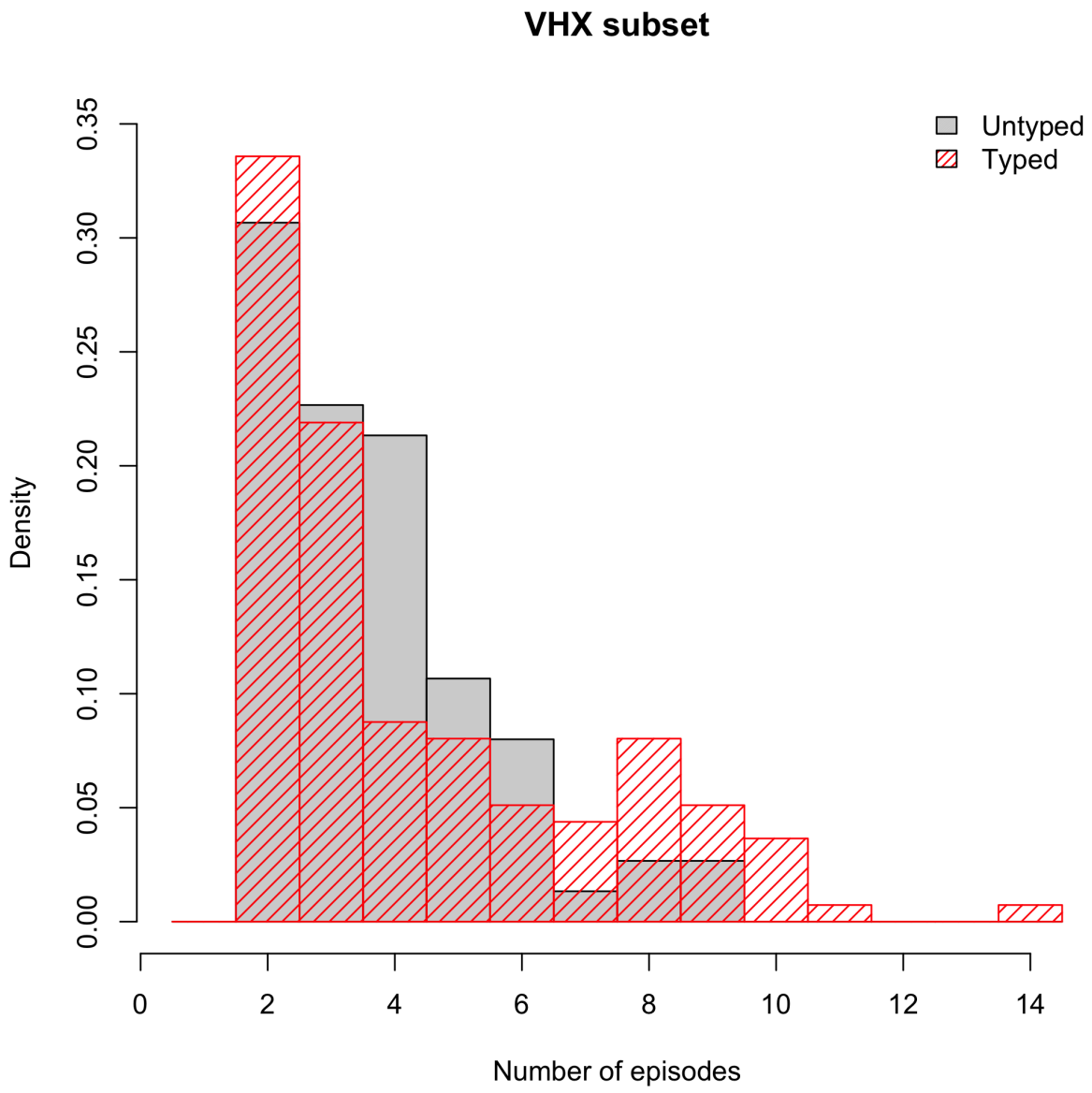
```
##
## BPD: of 81 of the people who recurred: 1 person/people with 1 recurrence/s was not selected for geno
```

These findings are summarised in the following plot.




```
#=================================================================
# Visual check of difference in episode counts between VHX
# that where genetically typed and not
#=================================================================
# Condition on CQ since these were the ones selected for genotyping
CQ_ind_epi <- Combined_Time_Data_no_cnsd_rows$arm_num == "CHQ"
# vector of patientids excluding censored rows and those without CQ
uncensored_patientid_vector = Combined_Time_Data_no_cnsd_rows$patientid[CQ_ind_epi]
# Number of episodes per person VHX
All_VHX_epi_count = table(uncensored_patientid_vector[grepl('VHX_',uncensored_patientid_vector)])

# Condition on those that have one or more recurrence
All_VHX_rec_count = All_VHX_epi_count[All_VHX_epi_count > 1]
x1 = no_of_epi_per_person_typed_VHX
```

```
x2 = All_VHX_rec_count[!names(All_VHX_rec_count) %in% names(x1)] # No of epi per person untyped
# setequal(names(x1), names(x2)) # Check mutually exclusive
# setequal(names(x2), unique(names(MS_pooled_summary$ID))) # Further check
max_rec = max(All_VHX_rec_count)
```



Summary of complexity of infection (COI) based on numbers of alleles observed.

This section is broken down by enrollment episodes (this is independent of drug given) and subsequent recurrences which could be drug dependent.

```
COIs = data.frame(t(sapply(unique(MS_pooled$Episode_Identifier), function(x){
  ind = which(MS_pooled$Episode_Identifier == x)
```

```
  c(MOI=max(MS_pooled$MOI_id[ind]),
    Enrollment = MS_pooled$Episode[ind[1]] == 1,
    Drug = MS_pooled$Treatment[ind[1]])
})))
COIs$MOI = as.numeric(COIs$MOI)
COIs$Enrollment = as.logical(COIs$Enrollment) # Converts factor to logical
COIs$PMQ = 0
COIs$PMQ[!COIs$Enrollment & COIs$Drug=='PMQ']=1
```

```
# XXX TAlk to James
# if we partition by receipt of PMQ and not under the hypothesis that if relapses are less diverse
# enrolment effect larger in partion who received PMQ:
COIs$PMQ = ifelse(COIs$Drug=='PMQ',1,0)
mod1 = glm(MOI ~ enrollment, family = 'poisson',
          data = data.frame(MOI= COIs$MOI - 1, # need to do minus 1 so it's Poisson appropriate
                            enrollment = as.numeric(COIs$Enrollment))[COIs$PMQ == 1,])
mod2 = glm(MOI ~ enrollment, family = 'poisson',
           data = data.frame(MOI= COIs$MOI - 1, # need to do minus 1 so it's Poisson appropriate
                             enrollment = as.numeric(COIs$Enrollment))[COIs$PMQ != 1,])
summary(mod1)
```

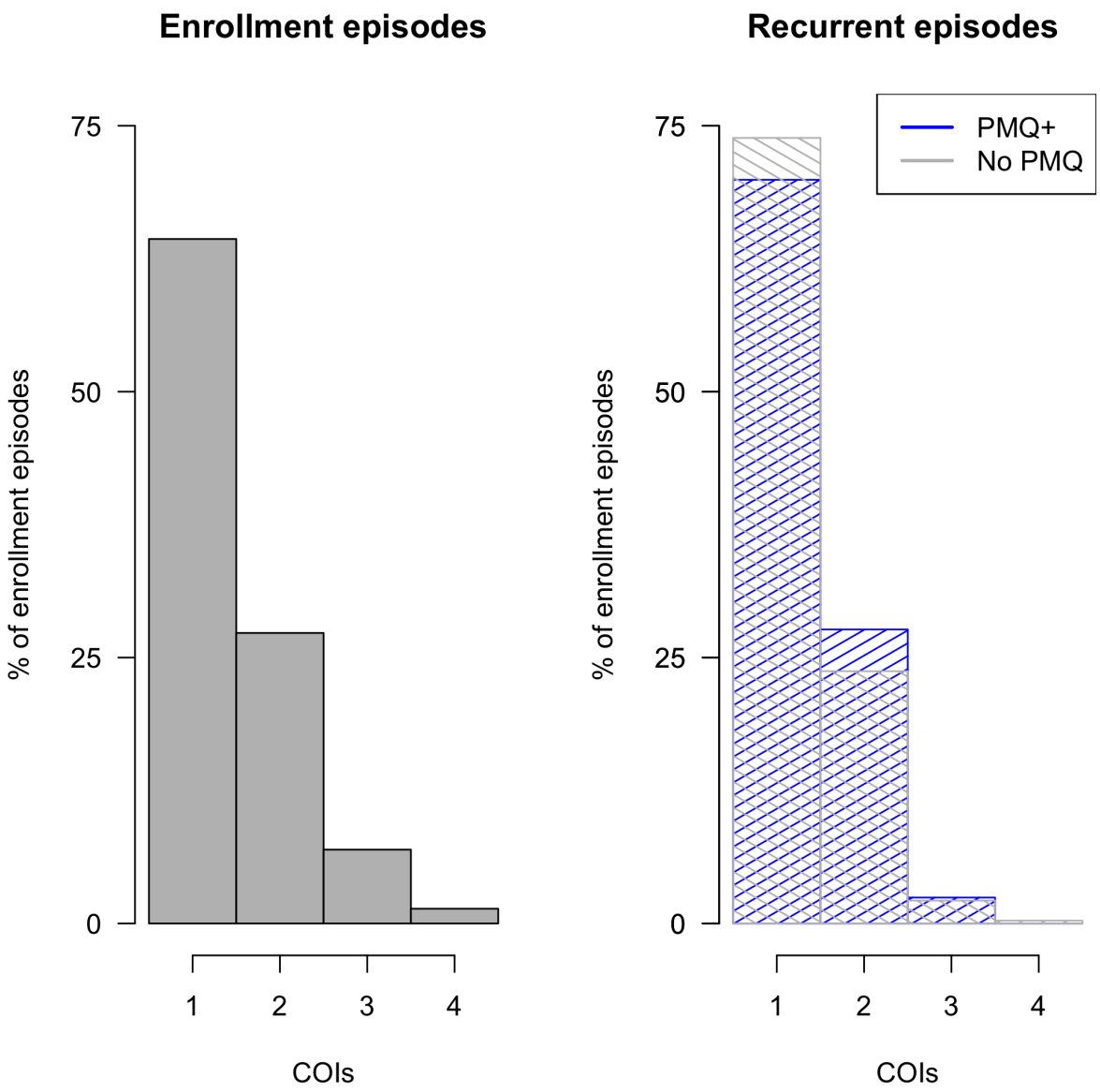
```
##
## Call:
## glm(formula = MOI ~ enrollment, family = "poisson", data = data.frame(MOI = COIs$MOI -
##     1, enrollment = as.numeric(COIs$Enrollment))[COIs$PMQ ==
##     1, ])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.8308  -0.8308  -0.8065   0.9044   2.7685
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.12330    0.15811  -7.104 1.21e-12 ***
## enrollment   0.05948    0.22504   0.264    0.792
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 202.73  on 235  degrees of freedom
## Residual deviance: 202.66  on 234  degrees of freedom
## AIC: 350.33
##
## Number of Fisher Scoring iterations: 6
```

```
summary(mod2)
```

```
##
## Call:
## glm(formula = MOI ~ enrollment, family = "poisson", data = data.frame(MOI = COIs$MOI -
##     1, enrollment = as.numeric(COIs$Enrollment))[COIs$PMQ !=
##     1, ])
##
## Deviance Residuals:
```

```
##     Min       1Q   Median       3Q      Max
## -1.0703  -0.7595  -0.7595   0.5099   2.9375
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.24337    0.09667 -12.862  < 2e-16 ***
## enrollment   0.68618    0.16216   4.232 2.32e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 416.98  on 473  degrees of freedom
## Residual deviance: 400.47  on 472  degrees of freedom
## AIC: 702.72
##
## Number of Fisher Scoring iterations: 6
```

```
# As it happens, opposite is true. Suggests to me the pattern in MOI might be to declining transmission
```




```
##
## Call:
## glm(formula = MOI ~ enrollment + drug, family = "poisson", data = data.frame(MOI = COIs$MOI -
##     1, enrollment = as.numeric(COIs$Enrollment), drug = COIs$PMQ))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9952  -0.7873  -0.7873   0.7659   2.8705
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.17136    0.08809 -13.297  < 2e-16 ***
## enrollment   0.46866    0.13561   3.456 0.000548 ***
## drug        -0.17476    0.14214  -1.230 0.218867
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 619.82  on 709  degrees of freedom
## Residual deviance: 608.18  on 707  degrees of freedom
## AIC: 1056.1
##
## Number of Fisher Scoring iterations: 6
```

```
## Mean complexity of recurrent episodes is 1.31, and mean complexity of enrollment episodes is 1.5
```

```
## Median COI in VHX and BPD: 1 and 1, respectively
```

```
## 30 of 710 episodes (4 percent) with COI greater than or equal to 3
```

From this Poisson regression, there appears to be evidence that enrollment episodes have higher complexities of infection than recurrences. This implies that relapses are more likely to be single hypnozoite activated infections? - XXXX

Allele frequencies

First we define the set of microsatellite markers used in this analysis:

```
MSs_all = c("PV.3.502","PV.3.27","PV.ms8",
            "PV.1.501","PV.ms1","PV.ms5",
            "PV.ms6","PV.ms7","PV.ms16")
```

We use a multinomial-dirichlet model with subjective weight ω. $\omega = 0$ recovers the empirical allele frequencies.

```
## Number of episodes used to compute frequencies: 216
```

```
#=====================================================================
# Save a data set of monoclonal data and allele frequencies for
# relatedness estimation (this was analysed elsewhere)
#=====================================================================
monoclonal_names = rownames(COIs)[COIs$MOI == 1]
#monoclonal_data = MS_pooled[MS_pooled$Episode_Identifier%in%monoclonal_names, ]
# save(monoclonal_data, Fs_Combined, file = '../RData/Data_for_relatedness.RData')
```

Calculate the effective marker cardinality for each microsatellite marker using a simulation approach.

```
N = 10^6
Effective_Allele_size = list()
for(ms in MSs_all){
  n = 1/sum(Fs_Combined[[ms]]^2)
  writeLines(sprintf('The effective cardinality for %s with %s observed alleles is %s', ms, length(Fs_C

  Effective_Allele_size[[ms]] = round(n,1)
}
```

```
## The effective cardinality for PV.3.502 with 16 observed alleles is 7.01
## The effective cardinality for PV.3.27 with 51 observed alleles is 13.74
## The effective cardinality for PV.ms8 with 50 observed alleles is 28.32
## The effective cardinality for PV.1.501 with 18 observed alleles is 12.97
## The effective cardinality for PV.ms1 with 7 observed alleles is 4.31
## The effective cardinality for PV.ms5 with 26 observed alleles is 11.96
```

```
## The effective cardinality for PV.ms6 with 27 observed alleles is 11.91
## The effective cardinality for PV.ms7 with 15 observed alleles is 6.91
## The effective cardinality for PV.ms16 with 43 observed alleles is 20.11
```

```
# The mean and range in our data set
writeLines(sprintf('The mean effective marker cardinality is %s, range: %s to %s',
                   round(mean(unlist(Effective_Allele_size)),2),
                   round(min(unlist(Effective_Allele_size)),2),
                   round(max(unlist(Effective_Allele_size)),2)))
```

```
## The mean effective marker cardinality is 13.02, range: 4.3 to 28.3
```

```
# Equivalent number of biallelic SNPs
sum(logb(unlist(Effective_Allele_size), base = 1.5)) # realistic SNPs
```

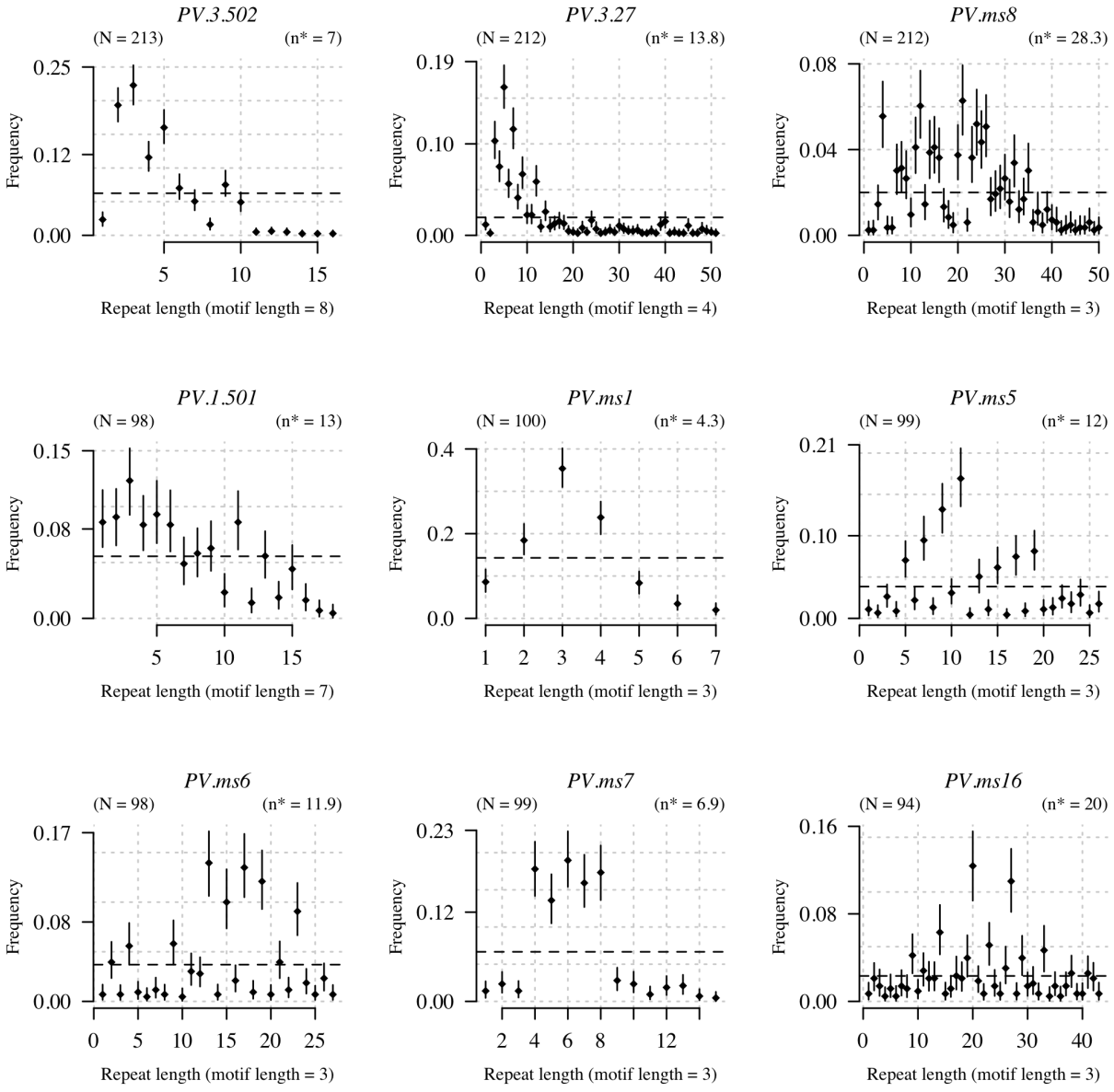
```
## [1] 53.82317
```

```
sum(logb(unlist(Effective_Allele_size), base = 2)) # ideal SNPs
```

```
## [1] 31.48454
```

Plotting allele frequencies

These are the mean posterior allele frequencies (dots) and 95% credible intervals (bars) given pooled enrollment data and $\omega =$ `D_weight_Prior`.



Summaries of samples selected to be genotyped at additional 6 markers

```
# Extract MS names
MS_core = MSs_all[1:3] # 1) core MSs
MS_addn = MSs_all[4:length(MSs_all)] # additionally typed MSs

# Format dataframe
MS_typd = MS_pooled_summary # Rename before relaplacing allele information with binary
MS_typd[,MSs_all][!is.na(MS_typd[,MSs_all])] = 1 # Replace allele information with binary

# Extract numbers of markers typed
```

```
MS_core_count = length(MS_core) - rowSums(is.na(MS_typd[,MS_core])) # Extract number of markers typed p
MS_typd_count = length(MSs_all) - rowSums(is.na(MS_typd[,MSs_all])) # Extract number of markers typed p
MS_addn_count = length(MS_addn) - rowSums(is.na(MS_typd[,MS_addn])) # Extract the number of additional
MS_addn_count_recurrent = length(MS_addn) - rowSums(is.na(MS_typd[MS_typd$Episode > 1,MS_addn])) # Extr

names(MS_typd_count) = names(MS_addn_count) = names(MS_core_count) = MS_typd$Episode_Identifier # Name

# Logical indeces
BPD_ind = grepl('BPD_', MS_typd$ID) # BPD individuals
VHX_ind = grepl('VHX_', MS_typd$ID) # VHX individuals
```

Summaries across all samples selected

```
#=============================================================================
# Summaries of core and additional (non-core) typed
# Zero samples failed at 6 of 6 non-core markers (email from Mallika June 12 2019)
# As such, zero non-core marker data means that zero additional were attempted
#=============================================================================
writeLines(sprintf('\nNumbers of %s samples partioned by number of markers successfully typed:', sum(tab
```

```
##
## Numbers of 710 samples partioned by number of markers successfully typed:
```

```
table(MS_typd_count)
```

```
## MS_typd_count
##   1   2   3   5   6   7   8   9
##   1   3 343   3   8   1  13 338
```

```
writeLines(sprintf('\nNumbers of %s samples partioned by number of core markers successfully typed:', su
```

```
##
## Numbers of 710 samples partioned by number of core markers successfully typed:
```

```
table(MS_core_count)
```

```
## MS_core_count
##   0   1   2   3
##   7   4   6 693
```

```
writeLines(sprintf('\nNumbers of %s samples partioned by number of non-core markers successfully typed:
```

```
##
## Numbers of 710 samples partioned by number of non-core markers successfully typed:
```

```
table(MS_addn_count)
```

```
## MS_addn_count
##   0   1   3   4   5   6
## 346   1   1   4  11 347
```

```
writeLines(sprintf('\nNumbers of %s recurrent samples partioned by number of non-core markers successful
```

```
##
## Numbers of 494 recurrent samples partioned by number of non-core markers successfully typed:
```

```
table(MS_addn_count_recurrent)
```

```
## MS_addn_count_recurrent
##   0   3   4   5   6
```

```
## 230   1   2   8 253
```

```
#================================================
# Episodes with missing core: share with Mallika
Episodes_missing_core = names(MS_core_count)[(which(MS_core_count < 3))] # Names
Data_missing_core = MS_typd[MS_typd$Episode_Identifier %in% Episodes_missing_core, ] # All data
MSData_missing_core = Data_missing_core[, c(MS_core, MS_addn)] # MS Data
rownames(MSData_missing_core) = Data_missing_core$Episode_Identifier # name rows
#write.csv(x = MSData_missing_core, file = '~/Dropbox/Genotyping/MSData_missing_core.csv', row.names =
#================================================

writeLines(sprintf('\nNumber of enrolment samples successfully genotyped at additional (non-core) marke
                   sum(MS_typd$Episode[MS_addn_count > 0] == 1),
                   sum(MS_typd$Episode == 1),
                   round(100*sum(MS_typd$Episode[MS_addn_count > 0] == 1)/sum(MS_typd$Episode == 1)),
                   sum(MS_typd$Episode[VHX_ind][MS_addn_count[VHX_ind] > 0] == 1), # VHX
                   sum(MS_typd$Episode[BPD_ind][MS_addn_count[BPD_ind] > 0] == 1))) # BPD
```

```
##
## Number of enrolment samples successfully genotyped at additional (non-core) markers: 100 of 216 (46 p
```

```
writeLines(sprintf('Number of recurrent samples successfully genotyped at additional (non-core) markers
                   sum(MS_typd$Episode[MS_addn_count > 0] != 1),
                   sum(MS_typd$Episode != 1),
                   round(100*sum(MS_typd$Episode[MS_addn_count > 0] != 1)/sum(MS_typd$Episode != 1)),
                   sum(MS_typd$Episode[VHX_ind][MS_addn_count[VHX_ind] > 0] != 1), # VHX
                   sum(MS_typd$Episode[BPD_ind][MS_addn_count[BPD_ind] > 0] != 1))) # BPD
```

```
## Number of recurrent samples successfully genotyped at additional (non-core) markers: 264 of 494 (53 p
```

```
writeLines(sprintf('Of those successfully genotyped at additional (non-core) markers, number genotyped a
                   sum(MS_addn_count[MS_addn_count > 0] == 6),
                   sum(MS_addn_count > 0),
                   round(100*sum(MS_addn_count[MS_addn_count > 0] == 6)/sum(MS_addn_count > 0)),
                   sum(MS_addn_count[VHX_ind][MS_addn_count[VHX_ind] > 0] == 6), # VHX
                   sum(MS_addn_count[BPD_ind][MS_addn_count[BPD_ind] > 0] == 6))) # BPD
```

```
## Of those successfully genotyped at additional (non-core) markers, number genotyped at all six: 347 o
```

Now we remove MS data for indiviuduals who miss paired enrolment and recurrent data in preparation for recurrent state inference

```
# First we remove MS data for which there are either no recurrent data or no enrolment data
IDs_Enrolment_n_Recurrent_typed = sapply(unique(MS_pooled_summary$ID), function(id){
  inds = MS_pooled_summary$ID == id
  episodes = MS_pooled_summary$Episode[inds]
  if(!1 %in% episodes){print(sprintf('enrolment missing for %s',id))}
  if(!length(episodes) > 1){print(sprintf('recurrent missing for %s',id))}
  if(length(episodes) > 1 & 1 %in% episodes){id}else{NA}  # one or more recurrent and contains and enro
  })
```

```
## [1] "recurrent missing for BPD_150"
## [1] "recurrent missing for BPD_453"
## [1] "enrolment missing for BPD_564"
## [1] "recurrent missing for BPD_564"
```

```
## [1] "recurrent missing for VHX_111"
## [1] "recurrent missing for VHX_557"
```

```
# Check that 'BPD_564' had only a single recurrence thus unanalzable: yes
MS_pooled$Episode[MS_pooled$ID == 'BPD_564']
```

```
## [1] 2
```

```
writeLines(sprintf('\nIndividuals removed due to missing enrolment or no recurrent data: \n %s',
paste(IDs_Enrolment_n_Recurrent_typed[is.na(IDs_Enrolment_n_Recurrent_typed)], collapse = ' ')))
```

```
##
## Individuals removed due to missing enrolment or no recurrent data:
##  NA NA NA NA NA
```

```
#-----------------------------------------------------
# Redefine dataframes XXX Consider re-nameing so code is more robust
#-----------------------------------------------------
MS_pooled = filter(MS_pooled, ID %in% IDs_Enrolment_n_Recurrent_typed[!is.na(IDs_Enrolment_n_Recurrent_t
MS_pooled_summary = MS_pooled[!duplicated(MS_pooled$Episode_Identifier),]  # recreate pooled summary da
MS_typd = MS_pooled_summary # Rename before relaplacing allele information with binary
MS_typd[,MSs_all][!is.na(MS_typd[,MSs_all])] = 1 # Replace allele information with binary

recur_ind = MS_pooled$Episode > 1
writeLines('\nNumber of individuals with at least one paired recurrence typed by treatment:')
```

```
##
## Number of individuals with at least one paired recurrence typed by treatment:
```

```
table(MS_pooled$trial_arm_partner[recur_ind][!duplicated(MS_pooled$ID[recur_ind])])[treatment_order]
```

```
##
##      VHX AS     VHX CHQ VHX CHQ/PMQ     BPD CHQ      BPD DP
##          13          90          32          46          31
```

```
writeLines('\nPercent of Nr individuals with at least one paired recurrence typed by treatment:')
```

```
##
## Percent of Nr individuals with at least one paired recurrence typed by treatment:
```

```
round(100*table(MS_pooled$trial_arm_partner[recur_ind][!duplicated(MS_pooled$ID[recur_ind])])[names(Nr)]
```

```
##
##      VHX AS     VHX CHQ VHX CHQ/PMQ     BPD CHQ      BPD DP
##           7          55          91          98          91
```

```
recur_ind = MS_pooled_summary$Episode > 1
writeLines('\nNumber of paired recurrences typed by treatment:')
```

```
##
## Number of paired recurrences typed by treatment:
```

```
table(MS_pooled_summary$trial_arm_partner[recur_ind])[treatment_order]
```

```
##
##      VHX AS     VHX CHQ VHX CHQ/PMQ     BPD CHQ      BPD DP
##          13         358          35          52          35
```

```
writeLines('\nPercent of paired recurrences typed by treatment:')
```

```
##
```

```
## Percent of paired recurrences typed by treatment:
```

```
round(100*table(MS_pooled_summary$trial_arm_partner[recur_ind])[names(R)]/R)[treatment_order]
```

```
##
##      VHX AS     VHX CHQ VHX CHQ/PMQ     BPD CHQ      BPD DP
##           2          61          88          98          90
```

```
writeLines(sprintf('\nNumber of individuals with one or more paired recurrence: %s',
                   length(unique(MS_pooled$ID))))
```

```
##
## Number of individuals with one or more paired recurrence: 212
```

```
writeLines(sprintf('Number of episodes in individuals with one or more paired recurrence: %s',
                   length(unique(MS_pooled$Episode_Identifier))))
```

```
## Number of episodes in individuals with one or more paired recurrence: 705
```

```
writeLines(sprintf('Number of paired recurrences: %s',
                   length(unique(MS_pooled$Episode_Identifier[MS_pooled$Episode>1]))))
```

```
## Number of paired recurrences: 493
```

```
# To add to Table 3
# Re-extract number of add. markers typed per episode
MS_addn_count_recurrent = length(MS_addn) - rowSums(is.na(MS_typd[MS_typd$Episode > 1,MS_addn]))
names(MS_addn_count_recurrent) = MS_typd[MS_typd$Episode > 1,'Episode_Identifier']
writeLines(sprintf('\nNumbers of %s paired recurrent samples partioned by number of non-core markers su
```

```
##
## Numbers of 493 paired recurrent samples partioned by number of non-core markers successfully typed:
```

```
table(MS_addn_count_recurrent)
```

```
## MS_addn_count_recurrent
##   0   3   4   5   6
## 229   1   2   8 253
```

Computing the probability of relatedness across infections

The approach is Bayesian and consists of the following:

- A prior probability vector for the recurrence state from the time-to-event model
- An allele frequency estimate from the posterior distribution of allele frequencies
- A likelihood based on the genetic data of being a *relapse*, a *recrudescence*, or a *reinfection* given the observed microsatellite data.

The following iterates through each individual and computes the probability of relatedness states.

Load the time-to-event priors

```
inds = grepl('mean_theta', colnames(Mod2_ThetaEstimates)) # Extract mean
p = data.frame(Episode_Identifier = Mod2_ThetaEstimates$Episode_Identifier,
               Mod2_ThetaEstimates[,inds], stringsAsFactors = F) # Reformat
colnames(p) = gsub(pattern = 'Recrudescence_mean_theta',replacement = 'C',x = colnames(p))
colnames(p) = gsub(pattern = 'Relapse_mean_theta',replacement = 'L',x = colnames(p))
colnames(p) = gsub(pattern = 'ReInfection_mean_theta',replacement = 'I',x = colnames(p))
```

```
# Get prior estimates for only those with genetic data
genetic_AND_time_data_eps = intersect(p$Episode_Identifier, MS_pooled$Episode_Identifier)
p = p[p$Episode_Identifier %in% genetic_AND_time_data_eps,]

# Extract posterior estimates only if running full posterior simple or inflated
if(RUN_MODELS_FULL_POSTERIOR | RUN_MODELS_FULL_POSTERIOR_INFLATED){
  Post_samples_matrix = Post_samples_matrix[Post_samples_matrix$Episode_Identifier %in% genetic_AND_time
}
```

Computation

```
# Remove upper complexity ids if EXCLUDE_COMPLEX = T
# This substantially speeds up the analysis
if(EXCLUDE_COMPLEX){
  load('../RData/UpperComplexityIDs.RData') # IDs to avoid
  ep_ids_excluded = unique(MS_pooled$Episode_Identifier[MS_pooled$ID %in% UpperComplexityIDs])
  MS_pooled = MS_pooled[!MS_pooled$ID %in% UpperComplexityIDs,]
  writeLines(sprintf('This is excluding data from %s individuals, totalling %s vivax episodes',
                     length(UpperComplexityIDs), length(ep_ids_excluded)))
}
```

Full posterior computation: non-complex cases

Using the time-to-event prior

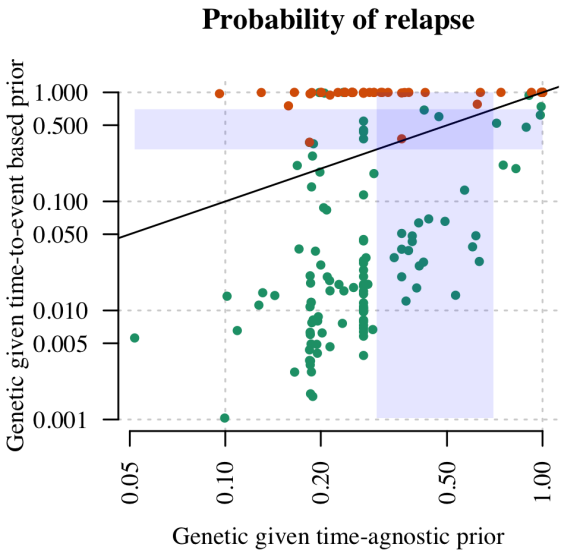
For genetic only efficacy estimate, run time agnostic (i.e. genetic only)

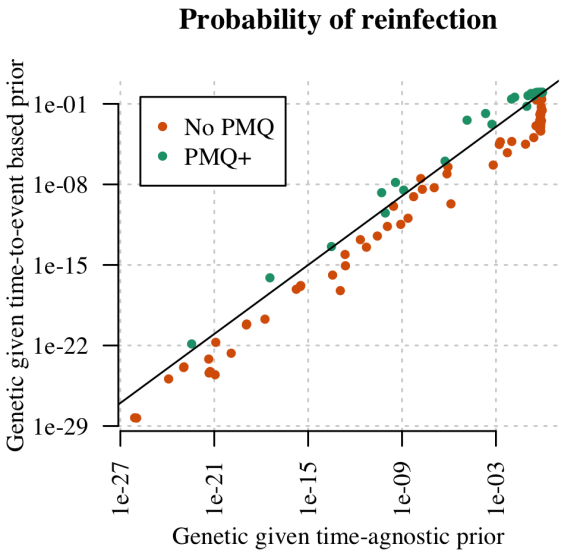
Plot results (thus far we have results for individuals with one or two recurrences only)

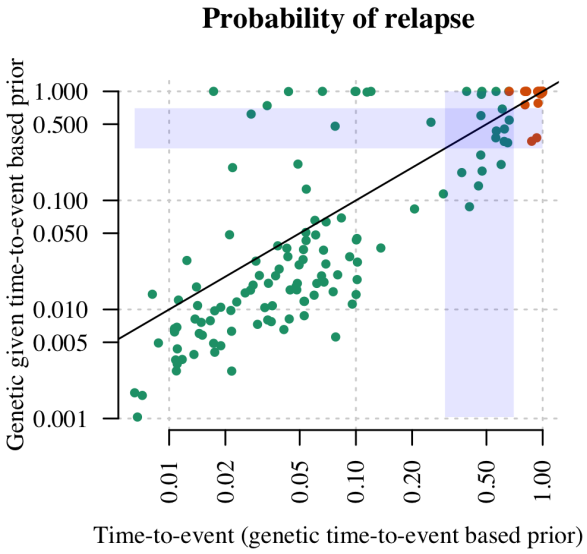
```
# Output of time-to-event model (sorted by episode number s.t. columns correspond)
Time_Estimates_1 = filter(Mod2_ThetaEstimates, Episode_Identifier %in% thetas_9MS$Episode_Identifier)
Time_Estimates_1 = arrange(Time_Estimates_1, Episode_Identifier)

# Outputs of genetic model with time prior
# sorted by episode number s.t. columns correspond and drug added
thetas_9MS = arrange(thetas_9MS, Episode_Identifier)
thetas_9MS$drug = Time_Estimates_1$arm_num # Add drug

# Outputs of genetic model without time prior
# sorted by episode number s.t. columns correspond and drug added
thetas_9MS_Tagnostic = arrange(thetas_9MS_Tagnostic, Episode_Identifier)
thetas_9MS_Tagnostic$drug = Time_Estimates_1$arm_num # Add drug
```

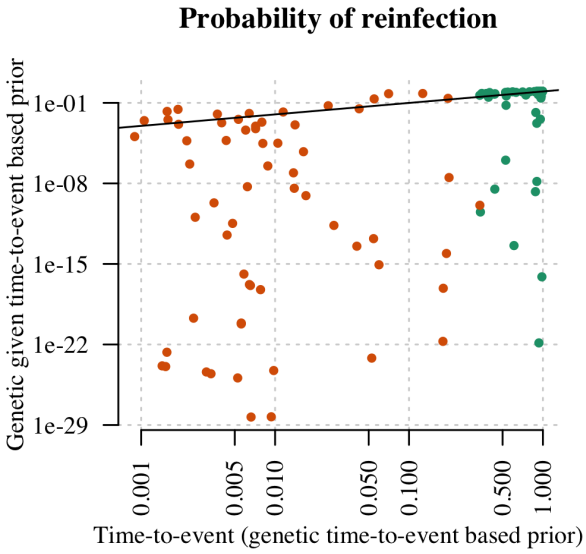
Going from time-to-event prior to posterior

Plotted by radical cure versus no radical cure, as that is the most informative distinction here.





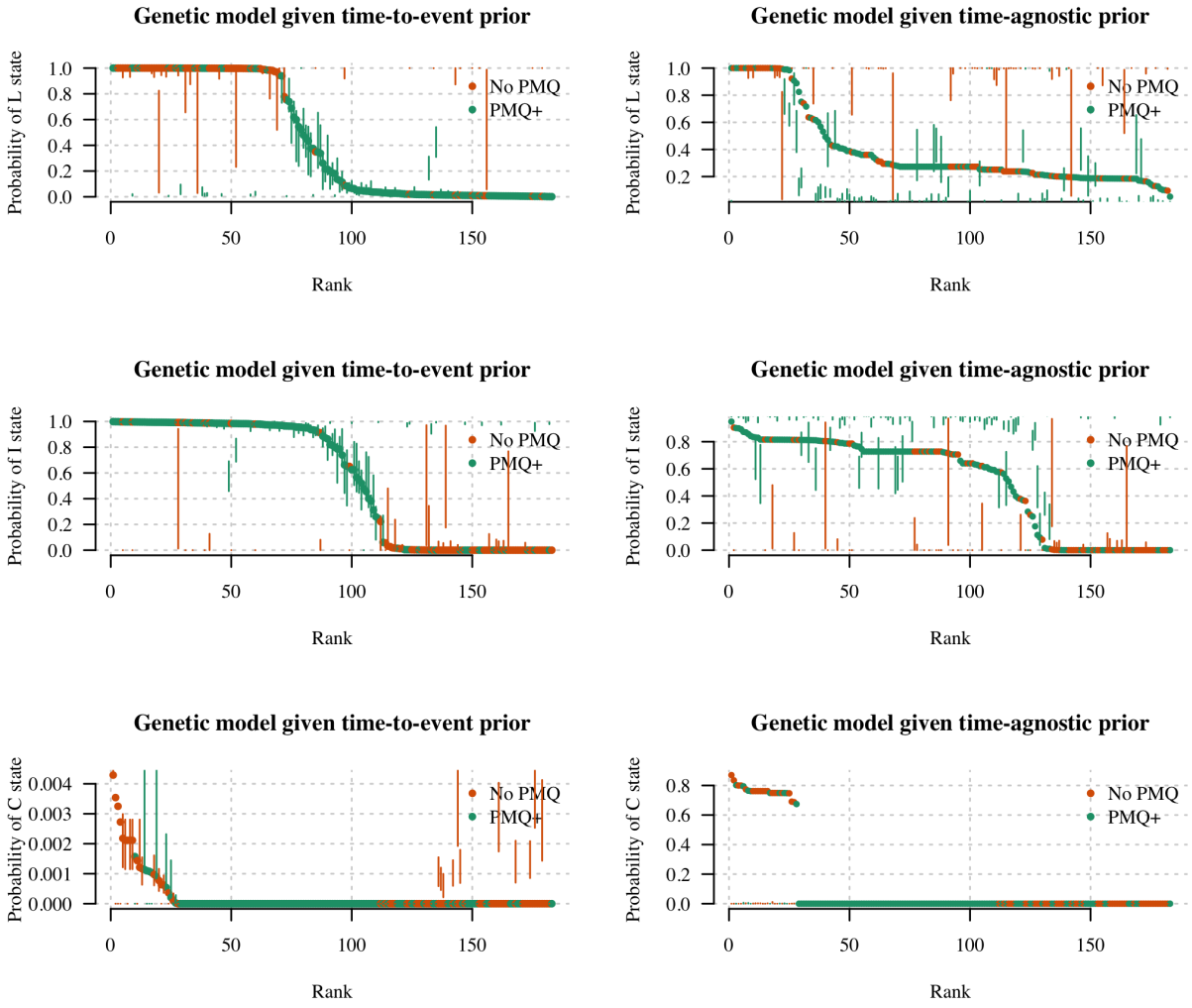





```
## Based on time-to-event alone, 65 of 66 No PMQ classified as relapse
## Based on genetic alone, 19 of 66 No PMQ classified as relapse
## Based on all available data, 61 of 66 No PMQ classified as relapse
## Based on time-to-event alone, 0 of 120 PMQ+ classified as relapse
## Based on genetic alone, 13 of 120 PMQ+ classified as relapse
## Based on all available data, 13 of 120 PMQ+ classified as relapse
```

Probability of states, ordered from most to least likely:

XXX Need to sort plotting bug - apprently appears only after first run



Extra computations for VHX: complex episodes

We remove the IDs that can be straightforwardly calculated:

```
ind_calculated = which(MS_pooled_summary$Episode_Identifier %in% thetas_9MS$Episode_Identifier)
IDs_calculated = unique(MS_pooled_summary$ID[ind_calculated])
IDs_remaining = unique(MS_pooled_summary$ID[! MS_pooled_summary$ID %in% IDs_calculated])
writeLines(sprintf('individuals with more than two recurrences: %s',length(IDs_remaining)))
```

```
## individuals with more than two recurrences: 54
```

We blow up the pooled analysis into all pairs within individuals:

Construct adjacency graphs and compute probabilities of relapse and reinfection.

```
MS_pooled_summary$L_or_C_state = MS_pooled_summary$TotalEpisodes = NA
MS_pooled_summary$L_lower = MS_pooled_summary$L_upper = MS_pooled_summary$L_median = NA
MS_pooled_summary$C_lower = MS_pooled_summary$C_upper = MS_pooled_summary$C_median = NA
MS_pooled_summary$I_lower = MS_pooled_summary$I_upper = MS_pooled_summary$I_median = NA
# Arrange by complexity
# Get single rows per episode (throw away the extra MOI information)
MS_inflated_summary = MS_inflated[!duplicated(MS_inflated$Episode_Identifier) &
```

```
                                    MS_inflated$Episode==2,]
Results_Inflated$Episode_Identifier = as.character(rownames(Results_Inflated))

for(i in 1:nrow(MS_inflated_summary)){
  if(!length(which(MS_inflated_summary$Episode_Identifier[i] ==
                   Results_Inflated$Episode_Identifier))>0){
    MS_inflated_summary = MS_inflated_summary[-i,]
    print('removing')
  }
}
```

```
## [1] "removing"
## [1] "removing"
```

```
Results_Inflated$ID_True = NA
Results_Inflated$First_EpNumber = NA
Results_Inflated$Second_EpNumber = NA
# The ordering has changed so need to be careful about naming
for(i in 1:nrow(Results_Inflated)){
  ind_MS_inflated = which(MS_inflated_summary$Episode_Identifier==Results_Inflated$Episode_Identifier[i]
  Results_Inflated$ID_True[i] =
    MS_inflated_summary$ID_True[ind_MS_inflated]
  Results_Inflated$First_EpNumber[i] =
    MS_inflated_summary$First_EpNumber[ind_MS_inflated]
  Results_Inflated$Second_EpNumber[i] =
    MS_inflated_summary$Second_EpNumber[ind_MS_inflated]
}



# Iterate through the ones we can calculate in one go
episodes_full_model = unique(thetas_9MS$Episode_Identifier)
for(ep in episodes_full_model){
  ind1 = MS_pooled_summary$Episode_Identifier==ep
  ind2 = thetas_9MS$Episode_Identifier==ep

  ## Summaries for relapse
  L_cols = grep('L',colnames(thetas_9MS))
  MS_pooled_summary$L_upper[ind1] = thetas_9MS[ind2, paste('L','97.5%',sep='')]
  MS_pooled_summary$L_lower[ind1] = thetas_9MS[ind2, paste('L','2.5%',sep='')]
  MS_pooled_summary$L_median[ind1] = thetas_9MS[ind2, paste('L','50%',sep='')]

  ## Summaries for recrudescence
  C_cols = grep('C',colnames(thetas_9MS))
  MS_pooled_summary$C_upper[ind1] = thetas_9MS[ind2, paste('C','97.5%',sep='')]
  MS_pooled_summary$C_lower[ind1] = thetas_9MS[ind2, paste('C','2.5%',sep='')]
  MS_pooled_summary$C_median[ind1] = thetas_9MS[ind2, paste('C','50%',sep='')]

  ## Summaries for reinfection
  I_cols = grep('I',colnames(thetas_9MS))
  MS_pooled_summary$I_upper[ind1] = thetas_9MS[ind2, paste('I','97.5%',sep='')]
  MS_pooled_summary$I_lower[ind1] = thetas_9MS[ind2, paste('I','2.5%',sep='')]
  MS_pooled_summary$I_median[ind1] = thetas_9MS[ind2, paste('I','50%',sep='')]
```

```
  # Just going to classify on relapse versus reinfection
  if(!is.na(MS_pooled_summary$L_upper[ind1])){
    if(MS_pooled_summary$L_upper[ind1]+MS_pooled_summary$C_upper[ind1] < Epsilon_lower){
      MS_pooled_summary$L_or_C_state[ind1] = 'I'
    } else if(MS_pooled_summary$L_lower[ind1]+MS_pooled_summary$C_lower[ind1] > Epsilon_upper){
      MS_pooled_summary$L_or_C_state[ind1] = 'L'
    } else {
      MS_pooled_summary$L_or_C_state[ind1] = 'Uncertain'
    }
  } else {
    MS_pooled_summary$L_or_C_state[ind1] = NA
  }
}
```

```
########## Complex cases **************************
# Now iterate through the complex ones
for(i in 1:length(IDs_remaining)){
  id = IDs_remaining[i]
  Doubles_Thetas = filter(Results_Inflated, ID_True==id)

  for(ep in unique(Doubles_Thetas$Second_EpNumber)){
    # indices on the MS_pooled_summary
    ind1 = which(MS_pooled_summary$ID==id & MS_pooled_summary$Episode==ep)
    # indices on Doubles thetas: looking for relapse evidence
    ind2 = which(Doubles_Thetas$Second_EpNumber == ep)
    # index for recrudescence evidence
    ind3 = which(Doubles_Thetas$Second_EpNumber == ep &
                   Doubles_Thetas$First_EpNumber == (ep-1))

    best_match_relapse = which.max(Doubles_Thetas[ind2, paste('L','50%',sep='')])
    if(length(best_match_relapse)>0){
      # Relapse probability
      MS_pooled_summary$L_lower[ind1] = Doubles_Thetas[ind2[best_match_relapse], paste('L','2.5%',sep='
      MS_pooled_summary$L_upper[ind1] = Doubles_Thetas[ind2[best_match_relapse], paste('L','97.5%',sep=
      MS_pooled_summary$L_median[ind1] = Doubles_Thetas[ind2[best_match_relapse], paste('L','50%',sep='

      # Reinfection probability
      MS_pooled_summary$I_lower[ind1] = Doubles_Thetas[ind2[best_match_relapse], paste('I','2.5%',sep='
      MS_pooled_summary$I_upper[ind1] = Doubles_Thetas[ind2[best_match_relapse], paste('I','97.5%',sep=
      MS_pooled_summary$I_median[ind1] = Doubles_Thetas[ind2[best_match_relapse], paste('I','50%',sep='

      # Recrudescence probability
      if(length(ind3)>0){
        # we can compute using previous episode
        MS_pooled_summary$C_lower[ind1] = Doubles_Thetas[ind3, paste('C','2.5%',sep='')]
        MS_pooled_summary$C_upper[ind1] = Doubles_Thetas[ind3, paste('C','97.5%',sep='')]
        MS_pooled_summary$C_median[ind1] = Doubles_Thetas[ind3, paste('C','50%',sep='')]
      }
      if(is.na(MS_pooled_summary$C_median[ind1])){
        MS_pooled_summary$C_lower[ind1] =
          1-MS_pooled_summary$L_lower[ind1]+MS_pooled_summary$I_lower[ind1]
        MS_pooled_summary$C_upper[ind1] =
          1-MS_pooled_summary$L_upper[ind1]+MS_pooled_summary$I_upper[ind1]
        MS_pooled_summary$C_median[ind1] =
```

```
          1-MS_pooled_summary$L_median[ind1]+MS_pooled_summary$I_median[ind1]
      }

    }
    if(!is.na(MS_pooled_summary$C_median[ind1])){
      if(MS_pooled_summary$L_upper[ind1] < MS_pooled_summary$L_lower[ind1]){
        writeLines(sprintf('Problem with ID %s',id))
        stop()
      }
      if(MS_pooled_summary$L_upper[ind1]+MS_pooled_summary$C_upper[ind1] < Epsilon_lower){
        MS_pooled_summary$L_or_C_state[ind1] = 'I'
      } else if(MS_pooled_summary$L_lower[ind1]+MS_pooled_summary$C_lower[ind1] > Epsilon_upper){
        MS_pooled_summary$L_or_C_state[ind1] = 'L'
      } else {
        MS_pooled_summary$L_or_C_state[ind1] = 'Uncertain'
      }
    }
  }
}

MS_pooled_summary$Drug = MS_pooled_summary$FU = NA
for(id in MS_pooled_summary$ID){
  ind = MS_pooled_summary$ID==id
  MS_pooled_summary$TotalEpisodes[ind] = max(MS_pooled_summary$Episode[ind])
  MS_pooled_summary$Drug[ind] = as.numeric(
    Combined_Time_Data$arm_num[Combined_Time_Data$patientid==id][1] == 'CHQ/PMQ') + 2
  MS_pooled_summary$FU[ind] = Combined_Time_Data$FU_time[Combined_Time_Data$patientid==id][1]
}
MS_pooled_summary$Plotting_pch_Values =
  as.numeric(mapvalues(MS_pooled_summary$L_or_C_state, from = c('L','Uncertain','I'), to = c(17,15,25)))
MS_pooled_summary$Plotting_col_Values =
  as.numeric(mapvalues(MS_pooled_summary$L_or_C_state, from = c('L','Uncertain','I'), to = 1:3))
```

```
# How many too complex to generate estimate for?
ind_recur = MS_pooled_summary$Episode > 1 # Filter out enrollment
ind_complex_recur = is.na(MS_pooled_summary$L_median[ind_recur])
no_complex_recur = sum(ind_complex_recur) # Recurrences with NAs
recur_removed = MS_pooled_summary$Episode_Identifier[ind_recur][ind_complex_recur]

# How many of the complex infections result in the loss of an individual
N_episodes_typed = table(MS_pooled_summary$ID)
indiv_removed = names(which(N_episodes_typed[MS_pooled_summary$ID[ind_recur][ind_complex_recur]] <= 2))
no_indiv_removed = length(indiv_removed)

# Final number of people with recurrences analysed total and by trial
indiv_recur_analysed = length(unique(MS_pooled_summary$ID[ind_recur][!ind_complex_recur]))
BPD_indiv_recur_analysed = sum(grepl('BPD', unique(MS_pooled_summary$ID[ind_recur][!ind_complex_recur])))
VHX_indiv_recur_analysed = sum(grepl('VHX', unique(MS_pooled_summary$ID[ind_recur][!ind_complex_recur])))

writeLines(sprintf('\nOf %s recurrences analysed, %s were too complex to estimate recurrence state proba
                   sum(ind_recur), no_complex_recur,
                   paste(recur_removed, collapse = ' '),
                   sum(!ind_complex_recur),
                   indiv_recur_analysed, BPD_indiv_recur_analysed, VHX_indiv_recur_analysed))
```

```
##
## Of 493 recurrences analysed, 7 were too complex to estimate recurrence state probabilities:
## VHX_239_2 VHX_33_2 VHX_39_2 VHX_461_2 VHX_52_2 VHX_551_2 VHX_583_2,
## resulting in probability estimates for a total of 486 recurrences from 208 individuals (77 BPD and 13
```

```
# Which drug arms do the unanalysed recurrences/individuals come from?
X0 = filter(MS_pooled_summary[MS_pooled_summary$Episode == 1,], ID %in% indiv_removed)
X1 = X0$trial_arm_partner; names(X1) = X0$ID
writeLines('\nIndividuals ommited due to computation complexity: ')
```

```
##
## Individuals ommited due to computation complexity:
```

```
X1
```

```
##   VHX_239    VHX_39   VHX_461    VHX_52
##  "VHX AS" "VHX CHQ"  "VHX AS" "VHX CHQ"
```

```
X0 = filter(MS_pooled_summary, Episode_Identifier %in% recur_removed)
X1 = X0$trial_arm_partner; names(X1) = X0$Episode_Identifier
writeLines('\nRecurrences ommited due to computation complexity:')
```

```
##
## Recurrences ommited due to computation complexity:
```

```
X1
```

```
##     VHX_239_2      VHX_33_2      VHX_39_2     VHX_461_2      VHX_52_2
##      "VHX AS"     "VHX CHQ"     "VHX CHQ"      "VHX AS"     "VHX CHQ"
##     VHX_551_2     VHX_583_2
##     "VHX CHQ" "VHX CHQ/PMQ"
```

```
recur_ind = MS_pooled_summary$Episode > 1
X0 = filter(MS_pooled_summary[recur_ind,], !ID %in% indiv_removed)
writeLines('\nNumber of individuals with at least one paired recurrence typed and analysed by treatment
```

```
##
## Number of individuals with at least one paired recurrence typed and analysed by treatment:
```

```
table(X0[!duplicated(X0$ID),'trial_arm_partner'])[treatment_order]
```

```
##
##      VHX AS     VHX CHQ VHX CHQ/PMQ     BPD CHQ      BPD DP
##          11          88          32          46          31
```

```
writeLines('\nPercent of Nr individuals with at least one paired recurrence typed  and analysed by trea
```

```
##
## Percent of Nr individuals with at least one paired recurrence typed  and analysed by treatment:
```

```
round(100*table(X0[!duplicated(X0$ID),'trial_arm_partner'])[names(Nr)]/Nr)[treatment_order]
```

```
##
##      VHX AS     VHX CHQ VHX CHQ/PMQ     BPD CHQ      BPD DP
##           6          53          91          98          91
```

```
recur_ind = MS_pooled_summary$Episode > 1
X0 = filter(MS_pooled_summary[recur_ind,], !Episode_Identifier %in% recur_removed)
writeLines('\nNumber of paired recurrences typed by treatment:')
```

```
##
## Number of paired recurrences typed by treatment:
```

```
table(XO$trial_arm_partner)[treatment_order]
```

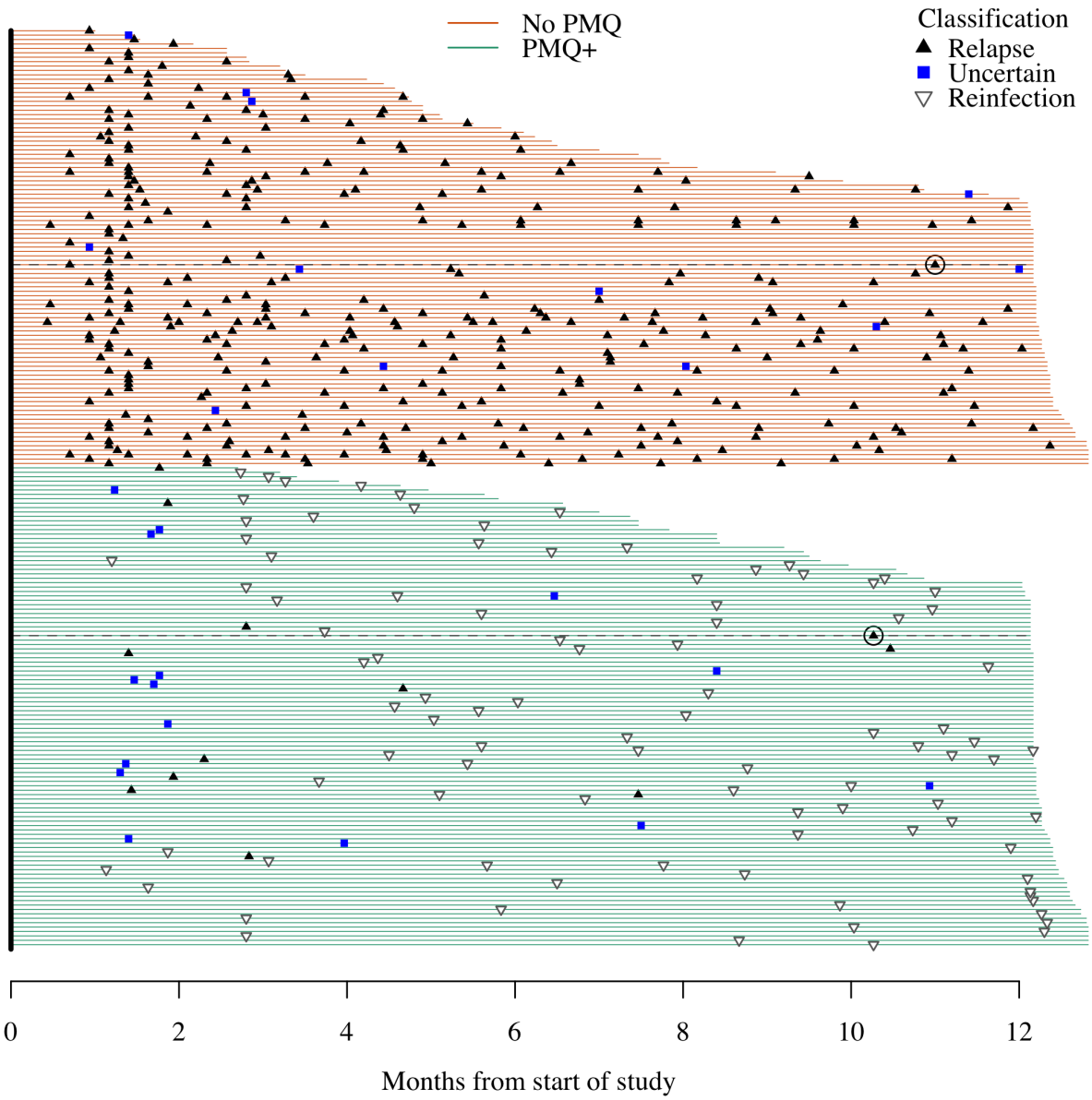
```
##
##      VHX AS     VHX CHQ VHX CHQ/PMQ     BPD CHQ      BPD DP
##          11         354          34          52          35
```

```
writeLines('\nPercent of paired recurrences typed by treatment:')
```

```
##
## Percent of paired recurrences typed by treatment:
```

```
round(100*table(XO$trial_arm_partner)[names(R)]/R)[treatment_order]
```

```
##
##      VHX AS     VHX CHQ VHX CHQ/PMQ     BPD CHQ      BPD DP
##           2          60          85          98          90
```

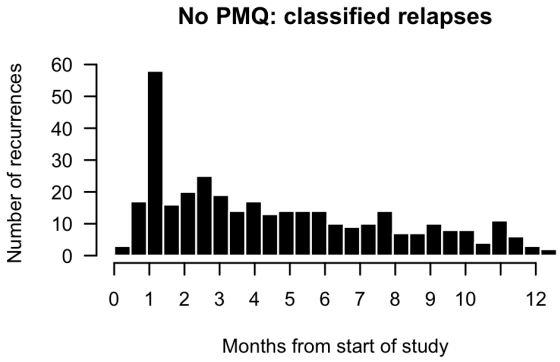


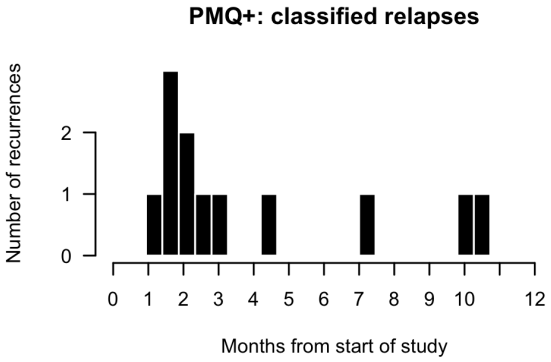

```
## The Coatney style plot is showing 486 recurrences in 208 individuals
```

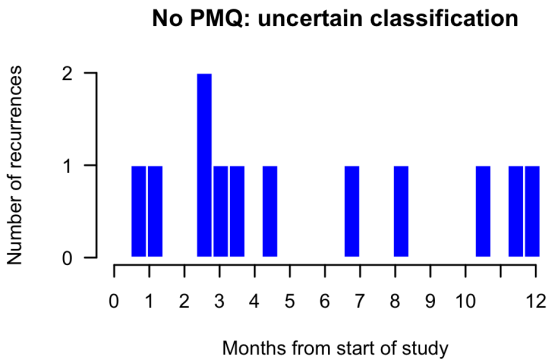
We show a histogram representation of these classification outputs as suggested by reviewer:

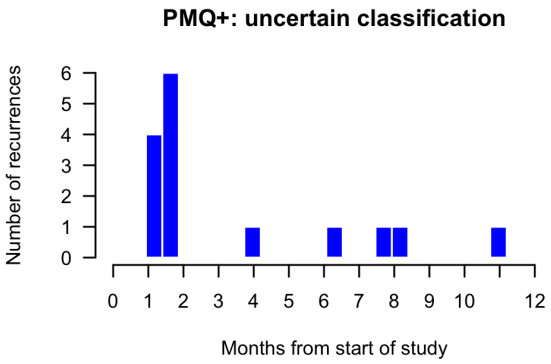
```
par(las=1, mfcol=c(3,2))
hist(MS_final$timeSinceEnrolment[MS_final$Plotting_col_Values==1 &
                                 MS_final$Treatment != 'PMQ'],
     main = 'No PMQ: classified relapses', xlab = 'Months from start of study',
     xaxt='n', ylab = 'Number of recurrences',  col = 'black', border = 'white',
     breaks = seq(0, 390, by = 14))
axis(1, at = 30*(0:12), labels = 0:12)
hist(MS_final$timeSinceEnrolment[MS_final$Plotting_col_Values==2 &
                                 MS_final$Treatment != 'PMQ'],
     main = 'No PMQ: uncertain classification',  ylab = 'Number of recurrences',
     xlab = 'Months from start of study', xaxt='n', yaxt='n',
```

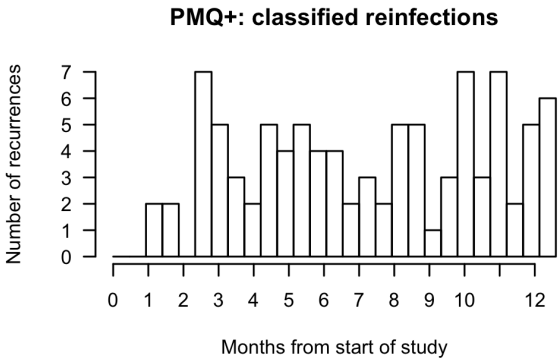
```
     breaks = seq(0, 390, by = 14), col = mycols_states_bg[2], border = 'white')
axis(1, at = 30*(0:12), labels = 0:12)
axis(2, at = 0:2)
plot(NA,NA,xlab='',ylab='',xlim=c(0,1),ylim=c(0,1),xaxt='n',yaxt='n',bty='n')

hist(MS_final$timeSinceEnrolment[MS_final$Plotting_col_Values==1 &
                                 MS_final$Treatment == 'PMQ'],
     main = 'PMQ+: classified relapses', ylab = 'Number of recurrences',
     xlab = 'Months from start of study', xaxt='n',yaxt='n',
     breaks = seq(0, 390, by = 14), col = 'black', border = 'white')
axis(1, at = 30*(0:12), labels = 0:12)
axis(2, at = 0:2)
hist(MS_final$timeSinceEnrolment[MS_final$Plotting_col_Values==2 &
                                 MS_final$Treatment == 'PMQ'],
     main = 'PMQ+: uncertain classification',ylab = 'Number of recurrences',
     xlab = 'Months from start of study', xaxt='n',
     col = mycols_states_bg[2],breaks = seq(0, 390, by = 14), border = 'white')
axis(1, at = 30*(0:12), labels = 0:12)

hist(MS_final$timeSinceEnrolment[MS_final$Plotting_col_Values==3 &
                                 MS_final$Treatment == 'PMQ'],
     main = 'PMQ+: classified reinfections',ylab = 'Number of recurrences',
     xlab = 'Months from start of study', xaxt='n',
     breaks = seq(0, 390, by = 14))
axis(1, at = 30*(0:12), labels = 0:12)
```

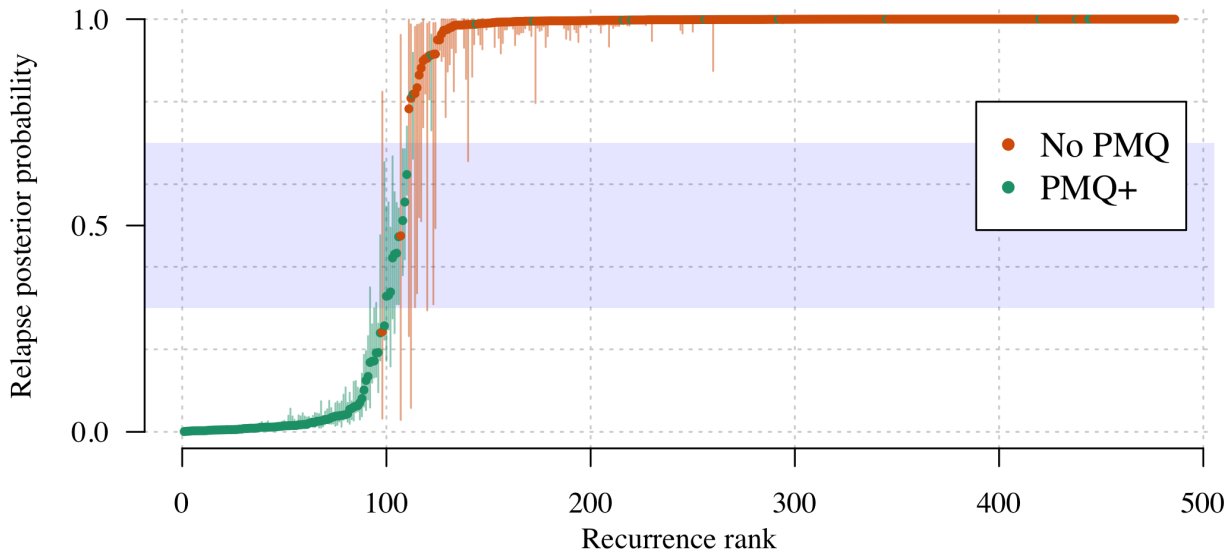

No PMQ: classified relapses
Number of recurrences
0
10
20
30
40
50
60
0 1 2 3 4 5 6 7 8 9 10 12
Months from start of study

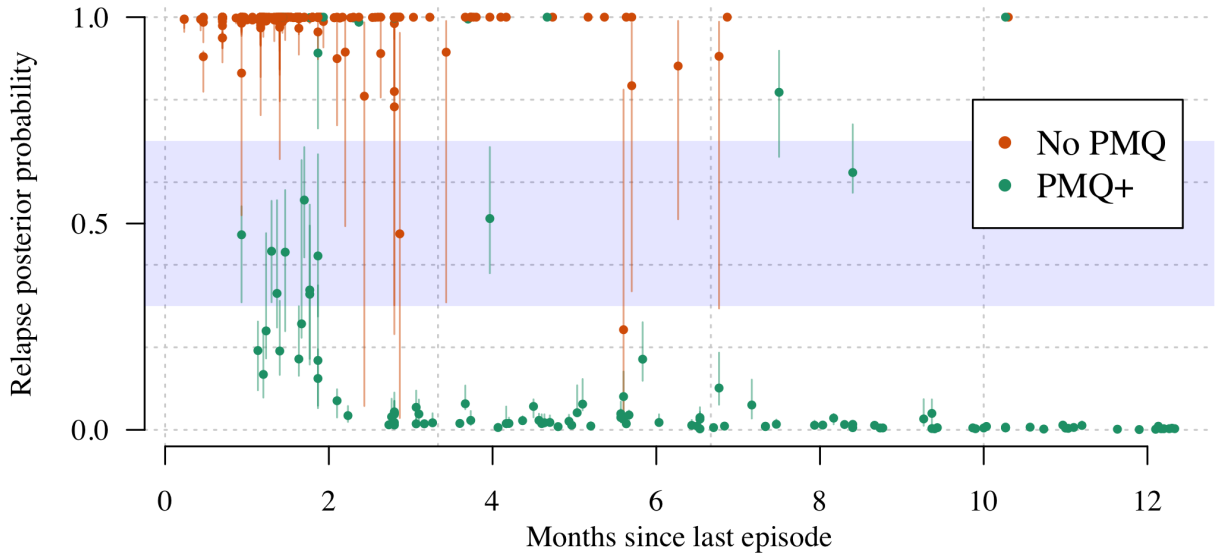

PMQ+: classified relapses
Number of recurrences
0
1
2
0 1 2 3 4 5 6 7 8 9 10 12
Months from start of study


No PMQ: uncertain classification
Number of recurrences
0
1
2
0 1 2 3 4 5 6 7 8 9 10 12
Months from start of study


PMQ+: uncertain classification
Number of recurrences
0
1
2
3
4
5
6
0 1 2 3 4 5 6 7 8 9 10 12
Months from start of study


PMQ+: classified reinfections
Number of recurrences
0
1
2
3
4
5
6
7
0 1 2 3 4 5 6 7 8 9 10 12
Months from start of study





Individuals who appear to relapse very late (more than 300 days after last episode):

```
MS_pooled = reformat_MSdata(MS_pooled)
IDs_late_relapse = MS_final[which(MS_final$timeSinceLastEpisode>300 & MS_final$L_lower>.9),'ID']

writeLines(sprintf('The episode ids of interest are: %s',
                   MS_final[which(MS_final$timeSinceLastEpisode>300 & MS_final$L_lower>.9),
                            'Episode_Identifier']))
```

```
## The episode ids of interest are: VHX_235_3
## The episode ids of interest are: BPD_27_2
```

```
print(MS_pooled[MS_pooled$ID%in%IDs_late_relapse,])
```

```
##          ID Episode Episode_Identifier Treatment MOI_id
## 60   BPD_27       1           BPD_27_1       PMQ      1
```

```
## 61   BPD_27       2           BPD_27_2       PMQ      1
## 62   BPD_27       2           BPD_27_2       PMQ      2
## 355 VHX_235       1          VHX_235_1       CHQ      1
## 356 VHX_235       1          VHX_235_1       CHQ      2
## 357 VHX_235       2          VHX_235_2       CHQ      1
## 358 VHX_235       3          VHX_235_3       CHQ      1
##     timeSinceLastEpisode timeSinceEnrolment PV.1.501 PV.3.27 PV.3.502
## 60                     0                 0        3      33        7
## 61                   308               308        3      33        7
## 62                   308               308        3      35        7
## 355                    0                 0        1       5        2
## 356                    0                 0        1       5        2
## 357                   21                21        1       5        3
## 358                  309               330        1       5        3
##     PV.ms1 PV.ms16 PV.ms5 PV.ms6 PV.ms7 PV.ms8 trial_arm_partner
## 60       4      27     24     15      5     17           BPD CHQ
## 61       4      27     24     15      5     17           BPD CHQ
## 62       4      27     24     15      5     17           BPD CHQ
## 355      3      23     13      9     10     12           VHX CHQ
## 356      3      23     13     15     10     33           VHX CHQ
## 357      4      20     13      9     10     12           VHX CHQ
## 358      4      23     11     15     10     12           VHX CHQ
```

The summaries of the final dataset. Results for all those genotyped who did not receive primaquine (artesunate or chloroquine monotherapy):

```
##
##  AS CHQ PMQ
##  11  88 109
```

```
## In no-primaquine individuals, the weighted average of relapse is 99 (96.5-99.8), for 365 recurrences
```

```
## In no-primaquine individuals, the weighted average of recrudescences is 0.3 (0.2-0.5), for 365 recur
```

```
## In no-primaquine individuals, the weighted average of reinfections is 0.9 (0.1-3.4), for 365 recurre
```

Results for all those genotyped who did receive primaquine (VHX and BPD studies combined):

```
## In primaquine treated individuals, the weighted average of relapses is 17 (14.4-20.7), for 121 recur
```

```
## In primaquine treated individuals, the weighted average of recrudescences is 0 (0-0), for 121 recurr
```

```
## In primaquine treated individuals, the weighted average of reinfections is 83 (79.3-85.6), for 121 re
```

Results for all those genotyped who did receive primaquine in the VHX study (unknown denominator)

```
## In primaquine treated individuals (VHX), the weighted average of relapses is 10.8 (8.8-13.3), for 34
```

```
## In primaquine treated individuals (VHX), the weighted average of recrudescences is 0 (0-0), for 34 re
```

```
## In primaquine treated individuals (VHX), the weighted average of reinfections is 89.2 (86.7-91.2), fo
```

Results for all those genotyped who did receive primaquine in the BPD study (known denominator)

```
## In primaquine treated individuals (BPD), the weighted average of relapses is 19.4 (16.6-23.7), for 87
```

```
## In primaquine treated individuals (BPD), the weighted average of recrudescences is 0.007 (0.003-0.01
```

```
## In primaquine treated individuals (BPD), the weighted average of reinfections is 80.6 (76.3-83.4), fo
```

False positive rate of relapse

We want to know how often our model estimates evidence of relapse across pairs of episodes when the episodes are in different people (i.e. have no possibility of being a relapse)

```
if(RUN_MODELS_FALSE_POSITIVE){
  # check if the massive pairwise dataset has been made, if not make it
  # (takes a long time ~20hours)
  if(!"APC_MSdata.bigRData"%in%list.files(path = '../RData/LargeFiles/')){
    # The pooled MS data from BPD and VHX
    load('../RData/GeneticModel/MS_data_PooledAnalysis.RData')
    tic()
    APC_MSdata = Make_All_Pairwise_Comparisons(MS_data = MS_pooled, ncores=42)
    save(APC_MSdata, file = '../RData/LargeFiles/APC_MSdata.bigRData')
    toc()
  }
  load('../RData/LargeFiles/APC_MSdata.bigRData')
  print('The inflated pairwise dataset is available, now running the analysis...')
  # Run the genetic model on the pairwise data
  tic()
  Inflated_Results = post_prob_CLI(MSdata = APC_MSdata,
                                   Fs = Fs_Combined,
                                   UpperComplexity = 10^6,
                                   verbose = T,
                                   cores = Ncores)
  toc()
  save(Inflated_Results, file = '../RData/LargeFiles/Inflated_Results.bigRData')
} else {
  load('../RData/LargeFiles/Inflated_Results.bigRData')
  Inflated_Results = Inflated_Results[!is.na(Inflated_Results$L),]
  load('../RData/LargeFiles/APC_MSdata.bigRData')
}
```

```
## The false-positive discovery rate of the genetic model is estimated as 2.17 percent.
##
## This is based on 247262 pairwise comparisons
```

Summaries of numbers of markers typed by classified recurrent samples

```
# Check that the episodes missing from MS_final are only those that were too complex: yes
inds = MS_typd$Episode[which(!MS_typd$Episode_Identifier %in% MS_final$Episode_Identifier)] > 1
missing_from_MS_final = MS_typd$Episode_Identifier[which(!MS_typd$Episode_Identifier %in% MS_final$Epis
all(recur_removed == missing_from_MS_final)
```

```
## [1] TRUE
```

```
#=============================================================================
# Classification of recurrent episodes genotyped at any markers
#=============================================================================
ind_classified = MS_final$Episode_Identifier %in% names(MS_typd_count)
recur_count = MS_typd_count[MS_final$Episode_Identifier[ind_classified]]
recur_typed_class = paste(recur_count, MS_final$L_or_C_state[ind_classified])
#table(recur_typed_class) # Table of number marks successful and classification
```

```
#=============================================================================
# Classification of recurrent episodes genotyped at additional markers
#=============================================================================
names_additional = names(MS_addn_count_recurrent) # Names of all
ind_classified = MS_final$Episode_Identifier %in% names_additional
recur_add_count = MS_addn_count_recurrent[MS_final$Episode_Identifier[ind_classified]]
recur_add_typed_class = paste(recur_add_count, MS_final$L_or_C_state[ind_classified])
ind_add_classified = MS_final$Episode_Identifier %in% names(which(MS_addn_count_recurrent > 1))

BPD_ind = grepl('BPD_', MS_final$ID) # Index for BPD individuals
VHX_ind = grepl('VHX_', MS_final$ID) # Index for VHX individuals

writeLines(sprintf('\nParition of %s recurrences analysed partioned by number of additional markers suc
                   sum(table(recur_add_count))))
```

```
##
## Parition of 486 recurrences analysed partioned by number of additional markers successfully typed:
```

```
table(recur_add_count)
```

```
## recur_add_count
##   0   3   4   5   6
## 229   1   2   8 246
```

```
writeLines(sprintf('\nClassification of %s recurrences analysed partioned by number of additional marke
                   sum(table(recur_add_typed_class))))
```

```
##
## Classification of 486 recurrences analysed partioned by number of additional markers successfully ty
```

```
table(recur_add_typed_class)
```

```
## recur_add_typed_class
##         0 I         0 L 0 Uncertain         3 L         4 L         5 L
##          89         117          23           1           2           8
##         6 I         6 L 6 Uncertain
##           5         237           4
```

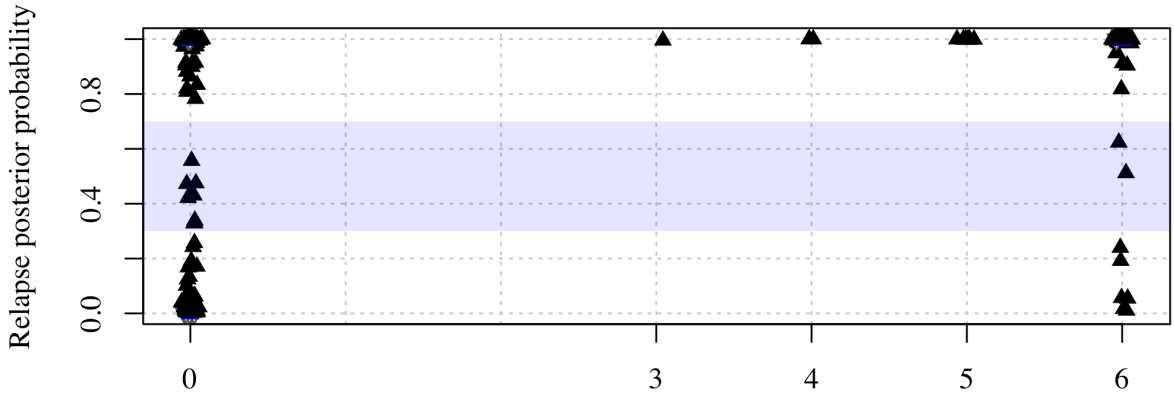
```
writeLines(sprintf('\nOf those recurrent genotyped at additional markers: %s of %s (%s percent, %s VHX,
                   sum(MS_final$L_or_C_state[ind_add_classified] == 'L'),
                   sum(ind_add_classified),
                   round(100*sum(MS_final$L_or_C_state[ind_add_classified] == 'L')/sum(ind_add_classifi
                   sum(MS_final$L_or_C_state[VHX_ind][ind_add_classified[VHX_ind]] == 'L'),
                   sum(MS_final$L_or_C_state[BPD_ind][ind_add_classified[BPD_ind]] == 'L')))
```

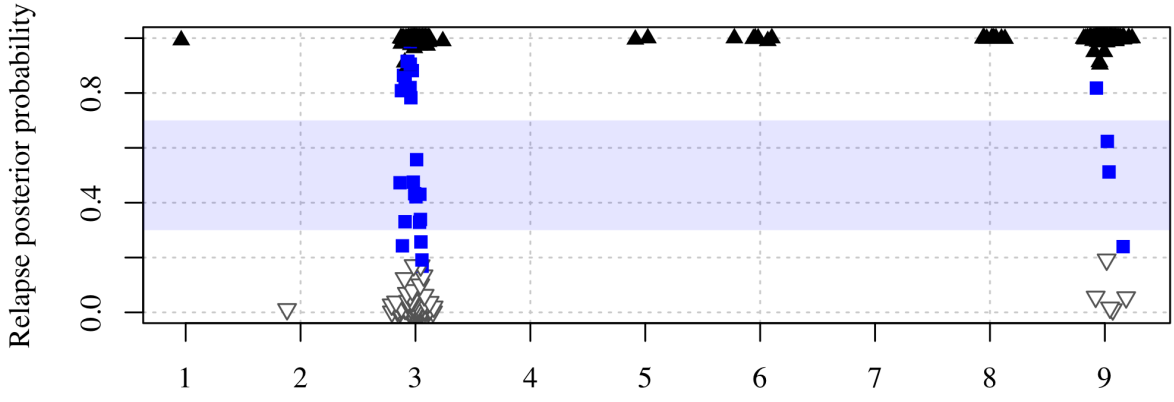
```
##
## Of those recurrent genotyped at additional markers: 248 of 257 (96 percent, 239 VHX, 9 BPD) classifi
```

Plots over classified recurrent samples

```
#=============================================================================
# Classification plots: why are some points beyond the uncertainity zone classified
# as uncertain
#=============================================================================
par(mfrow = c(2,1), family = 'serif')

plot(x = recur_add_count + rnorm(length(recur_add_count), 0, 0.025), # Add jitter
```

```
     y = MS_final$L_median[ind_classified],
     panel.first = grid(),
     pch = MS_final$Plotting_pch_Values[ind_add_classified],
     bg=mycols_states_bg[MS_final$Plotting_col_Values[ind_add_classified]],
     col=mycols_states_fg[MS_final$Plotting_col_Values[ind_add_classified]],
     ylab = 'Relapse posterior probability',
     xlab = 'Number of additional markers successfully typed',
     xaxt = 'n') # Number additional successful vs median relapse probablity
axis(side = 1, at = c(0,3:6)) # Add axis

# Add zone of uncertainity
polygon(x = c(-10,max(MS_final$timeSinceLastEpisode)+100,
              max(MS_final$timeSinceLastEpisode)+100,-10),
        y = c(Epsilon_lower,Epsilon_lower,Epsilon_upper,Epsilon_upper),
        col = transparent_blue_band, border = NA)

plot(x = recur_count + rnorm(length(recur_count), 0, 0.075), # Add jitter
     MS_final$L_median[ind_classified],
     panel.first = grid(),
     pch = MS_final$Plotting_pch_Values[ind_classified],
     bg=mycols_states_bg[MS_final$Plotting_col_Values[ind_classified]],
     col=mycols_states_fg[MS_final$Plotting_col_Values[ind_classified]],
     ylab = 'Relapse posterior probability',
     xlab = 'Number of markers successfully typed',
     xaxt = 'n') # Number additional successful vs median relapse probablity
axis(side = 1, at = 1:9) # Add axis

# Add zone of uncertainity
polygon(x = c(-10,max(MS_final$timeSinceLastEpisode)+100,
              max(MS_final$timeSinceLastEpisode)+100,-10),
        y = c(Epsilon_lower,Epsilon_lower,Epsilon_upper,Epsilon_upper),
        col = transparent_blue_band, border = NA)
```





Analysis of radical cure efficacy in BPD and VHX

Almost all recurrences in BPD were typed. The majority (34 out of 40) of the recurrences in VHX were typed also. Therefore we can estimate the true efficacy of high-dose primaquine by adjusting for the background reinfection rates in both studies.

```
# Add an episode identifier
Combined_Time_Data$Episode_Identifier = apply(Combined_Time_Data,1,
                                              function(x){paste(x['patientid'],as.integer(x['episode'])

# Create columns into which probabilities can be stored
Combined_Time_Data$Reinfection_Probability=
  Combined_Time_Data$Reinfection_Probability_LL=
  Combined_Time_Data$Reinfection_Probability_UL = NA
```

```
# Add an episode identifier
Mod2_ThetaEstimates$Failure_Identifier = apply(Mod2_ThetaEstimates, 1,
                                               function(x) paste(x['patientid'],as.integer(x['episode']]

# Iterate over every episode and use either the joint posterior
# or, if missing, the time probability, which could be time censored
sss=0
for(i in 1:nrow(Combined_Time_Data)){

  ep_id = Combined_Time_Data$Episode_Identifier[i] # Extract episode id

  # we look one episode ahead
  MS_id = paste(Combined_Time_Data$patientid[i],as.integer(Combined_Time_Data$episode[i])+1, sep='_')

  # If in MS_final, we extract the full reinfection probability
  if(MS_id %in% MS_final$Episode_Identifier){

    Combined_Time_Data$Reinfection_Probability[i] =
      MS_final$I_median[MS_final$Episode_Identifier==MS_id]
    Combined_Time_Data$Reinfection_Probability_UL[i] =
      MS_final$I_upper[MS_final$Episode_Identifier==MS_id]
    Combined_Time_Data$Reinfection_Probability_LL[i] =
      MS_final$I_lower[MS_final$Episode_Identifier==MS_id]

  } else { # use the time to event model

    ind = which(Mod2_ThetaEstimates$Failure_Identifier==MS_id)

    if(length(ind)>0){

      Combined_Time_Data$Reinfection_Probability[i] =
        Mod2_ThetaEstimates$ReInfection_mean_theta[ind]
      Combined_Time_Data$Reinfection_Probability_UL[i] =
        Mod2_ThetaEstimates$ReInfection_975_theta[ind]
      Combined_Time_Data$Reinfection_Probability_LL[i] =
        Mod2_ThetaEstimates$ReInfection_025_theta[ind]
      sss=sss+1
    }
  }
}
```

PK analysis with carboxy primaquine

Now we look at whether the PK (carboxy-primaquine) can predict failure: First we add the carboxy to the dataset:

```
Combined_Time_Data = arrange(Combined_Time_Data, patientid, episode)
load('../RData/PK_data/BPD_pk.RData')
BPD_pk = filter(BPD_pk, !is.na(Episode))
Combined_Time_Data$log10_carboxyPMQ = NA
Combined_Time_Data$log10_PMQ = NA
Combined_Time_Data$NumberDaysPMQ = NA
# The default is 14 days
```

```
Combined_Time_Data$NumberDaysPMQ[Combined_Time_Data$Censored != 1 &
                                 Combined_Time_Data$arm_num=='CHQ/PMQ'] = 14
for(i in 1:nrow(Combined_Time_Data)){
  id = Combined_Time_Data$patientid[i]
  ep_i = Combined_Time_Data$episode[i]
  all_id_eps = Combined_Time_Data$episode[Combined_Time_Data$patientid==id]
  pk_ind = which(BPD_pk$ID == id & BPD_pk$Episode==ep_i)
  if(length(pk_ind)>0){
    if(length(pk_ind)>1) print(id)
    Combined_Time_Data$log10_carboxyPMQ[i] = mean(BPD_pk$log10_carboxyPQ_PK[pk_ind])
    Combined_Time_Data$log10_PMQ[i] = mean(BPD_pk$log10_PQ_PK[pk_ind])
    Combined_Time_Data$NumberDaysPMQ[i] = BPD_pk$NumberofPKDays[pk_ind[1]]
  }
}
```

```
## [1] "BPD_34"
```

We exclude the two recurrences seen in patient BPD_444 who was G6PD deficient and received the 8 weekly regimen (not daily dosing).

```
# These are two outliers - have discussed with Cindy
BPD444_recurrences = Combined_Time_Data$patientid=='BPD_44' & Combined_Time_Data$episode>1
BPD_598 = which(Combined_Time_Data$patientid=='BPD_598')
ind_keep = !BPD444_recurrences #& !BPD_598
Combined_Time_Data$Failure_YN = Combined_Time_Data$Reinfection_Probability < 0.5
mod = glmer(Failure_YN ~ log10_carboxyPMQ + NumberDaysPMQ +
              (1 | patientid),
            family = 'binomial', data=Combined_Time_Data[ind_keep,])
```

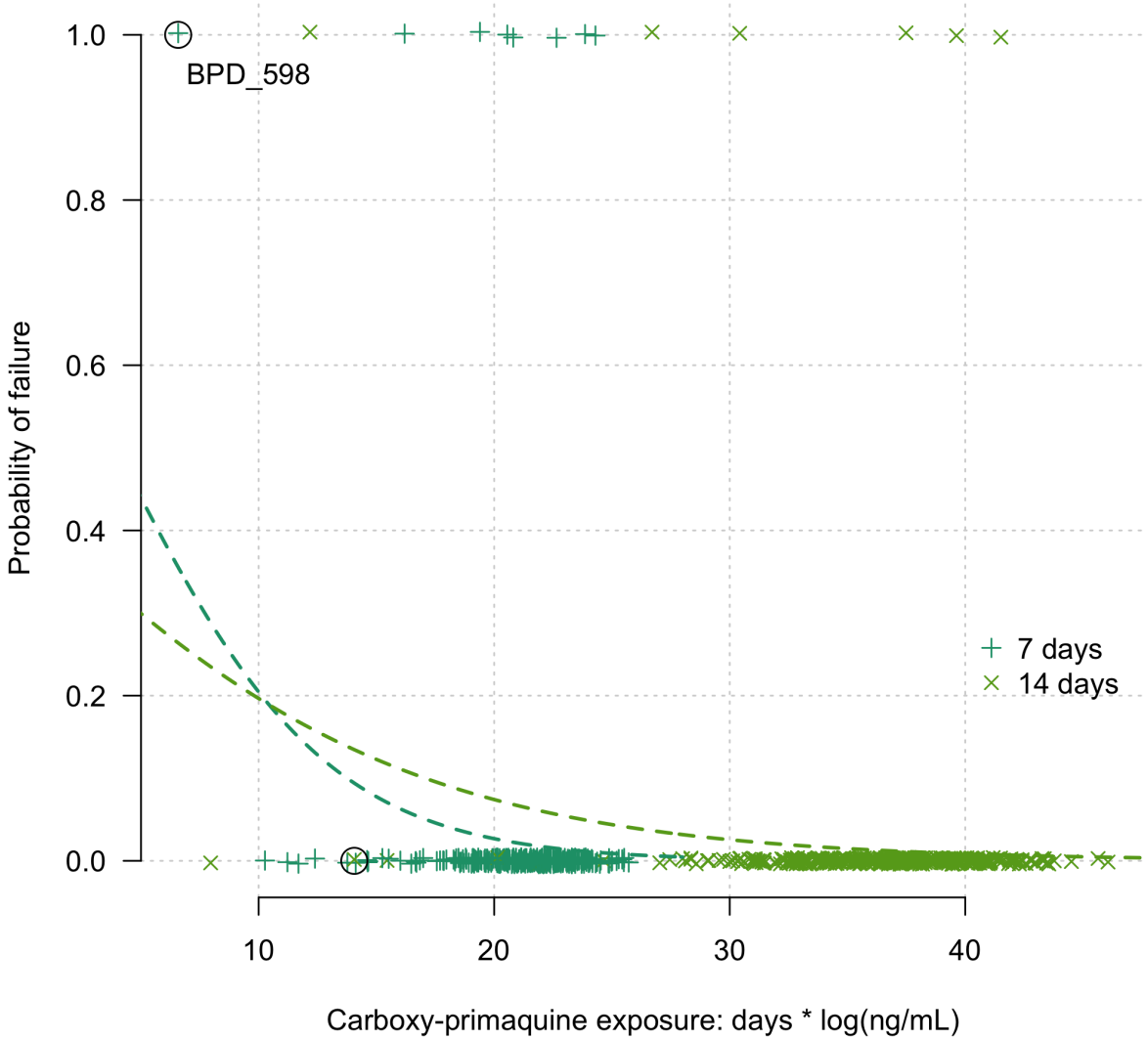
```
## boundary (singular) fit: see ?isSingular
```

```
summary(mod)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: Failure_YN ~ log10_carboxyPMQ + NumberDaysPMQ + (1 | patientid)
##    Data: Combined_Time_Data[ind_keep, ]
##
##      AIC      BIC   logLik deviance df.resid
##    137.1    155.4    -64.6    129.1      717
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -0.5542 -0.1433 -0.1226 -0.1042 11.7415
##
## Random effects:
##  Groups    Name        Variance  Std.Dev.
##  patientid (Intercept) 2.557e-15 5.056e-08
## Number of obs: 721, groups:  patientid, 639
##
## Fixed effects:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)       2.04111    1.72815   1.181  0.23757
## log10_carboxyPMQ -1.56314    0.48069  -3.252  0.00115 **
## NumberDaysPMQ    -0.16660    0.08352  -1.995  0.04609 *
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) l10_PM
## lg10_crbPMQ -0.875
## NumbrDysPMQ -0.720  0.329
## convergence code: 0
## boundary (singular) fit: see ?isSingular
```

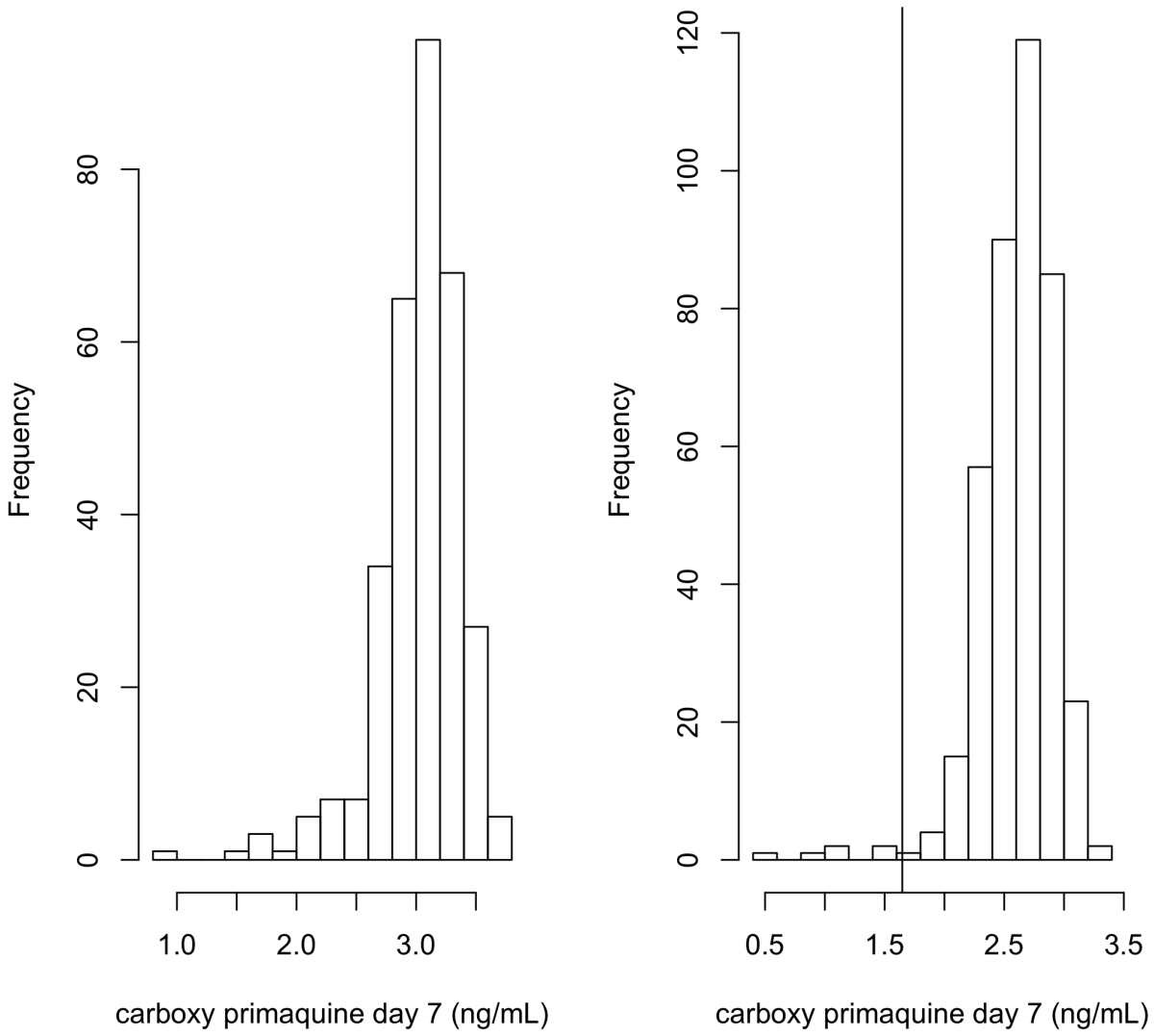
```
# Plot the data and model
xs=seq(0,4,by=.01)
par(las = 1, bty='n')
regimen_colors = brewer.pal(8, 'Dark2')[c(1,5)]
plot(Combined_Time_Data$log10_carboxyPMQ[ind_keep]*Combined_Time_Data$NumberDaysPMQ[ind_keep],
     jitter(as.numeric(Combined_Time_Data$Failure_YN[ind_keep]), factor = 0.02),
     col = regimen_colors[mapvalues(Combined_Time_Data$NumberDaysPMQ[ind_keep],c(7,14),c(1,2))],
     pch = mapvalues(Combined_Time_Data$NumberDaysPMQ[ind_keep],c(7,14),c(3,4)),
     xlab = 'Carboxy-primaquine exposure: days * log(ng/mL)',
     panel.first = grid(), ylab = 'Probability of failure')
legend(x = 40, y = 0.3, bty='n', col =regimen_colors,
       pch=c(3,4), legend = c('7 days','14 days'))
lines(xs*7, predict(mod, newdata=data.frame(log10_carboxyPMQ=xs, NumberDaysPMQ=7,
                                            patientid='new'),allow.new.levels=T,
                    type='response'), lwd=2, col= regimen_colors[1], lty=2)
lines(xs*14, predict(mod, newdata=data.frame(log10_carboxyPMQ=xs, NumberDaysPMQ=14,
                                             patientid='new'),allow.new.levels=T,
                     type='response'), lwd=2, col= regimen_colors[2], lty = 2)
points(Combined_Time_Data$log10_carboxyPMQ[BPD_598]*Combined_Time_Data$NumberDaysPMQ[BPD_598],
       Combined_Time_Data$Failure_YN[BPD_598], cex=2)
text(Combined_Time_Data$log10_carboxyPMQ[BPD_598[1]]*
       Combined_Time_Data$NumberDaysPMQ[BPD_598[1]]+3,
     Combined_Time_Data$Failure_YN[BPD_598[1]]-0.05, labels = 'BPD_598')
```



Now we remove outliers and fit the same model (CPMQ outliers)

```
mu_hat_7 = mean(Combined_Time_Data$log10_carboxyPMQ[Combined_Time_Data$NumberDaysPMQ==7])
sd_hat_7 = sd(Combined_Time_Data$log10_carboxyPMQ[Combined_Time_Data$NumberDaysPMQ==7])
mu_hat_14 = mean(Combined_Time_Data$log10_carboxyPMQ[Combined_Time_Data$NumberDaysPMQ==14],na.rm=T)
sd_hat_14 = sd(Combined_Time_Data$log10_carboxyPMQ[Combined_Time_Data$NumberDaysPMQ==14],na.rm=T)

par(mfrow=c(1,2))
hist(Combined_Time_Data$log10_carboxyPMQ[Combined_Time_Data$NumberDaysPMQ==7],
     main='', xlab='carboxy primaquine day 7 (ng/mL)')
abline(v = mu_hat_7 - sd_hat_7*3)

hist(Combined_Time_Data$log10_carboxyPMQ[Combined_Time_Data$NumberDaysPMQ==14],
     main = '', xlab='carboxy primaquine day 7 (ng/mL)')
```

```
abline(v = mu_hat_14 - sd_hat_14*3)
```




```
outliers7 = Combined_Time_Data$NumberDaysPMQ==7 & Combined_Time_Data$log10_carboxyPMQ<mu_hat_7 - sd_hat
outliers14 = Combined_Time_Data$NumberDaysPMQ==14 & Combined_Time_Data$log10_carboxyPMQ<mu_hat_14 - sd_

mod_No_Outliers = glmer(Failure_YN ~ log10_carboxyPMQ + NumberDaysPMQ +
                          (1 | patientid),
                        family = 'binomial', data=Combined_Time_Data[ind_keep & !outliers14 & !outliers
```

```
## fixed-effect model matrix is rank deficient so dropping 1 column / coefficient
```

```
## boundary (singular) fit: see ?isSingular
```

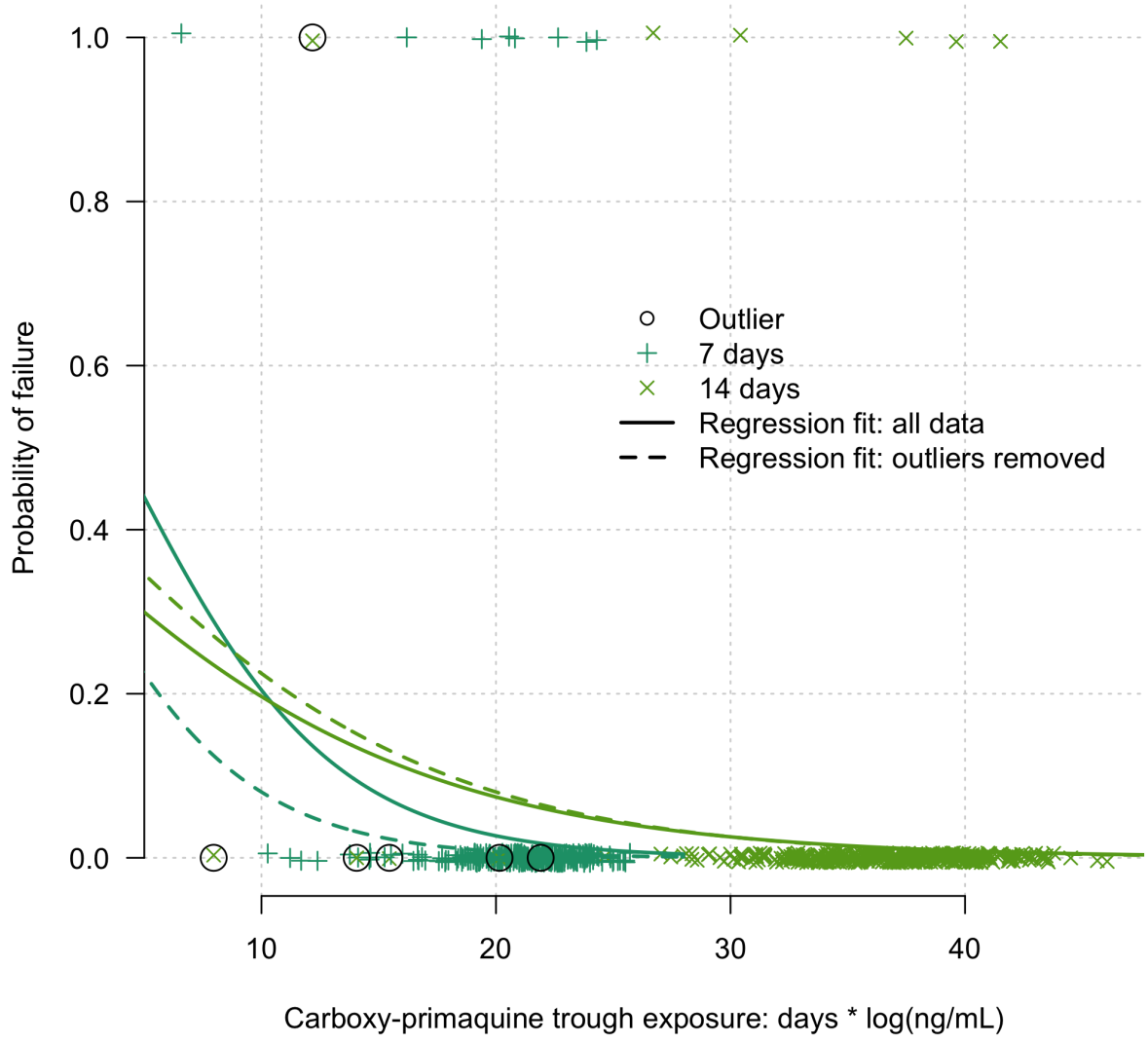
```
summary(mod_No_Outliers)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: Failure_YN ~ log10_carboxyPMQ + NumberDaysPMQ + (1 | patientid)
##    Data: Combined_Time_Data[ind_keep & !outliers14 & !outliers7, ]
##
##      AIC      BIC   logLik deviance df.resid
##     58.7     70.6    -26.3     52.7      393
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -0.2235 -0.1255 -0.1055 -0.0913 12.3212
##
## Random effects:
##  Groups    Name        Variance  Std.Dev.
##  patientid (Intercept) 3.074e-14 1.753e-07
## Number of obs: 396, groups:  patientid, 352
##
## Fixed effects:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -0.03691    4.20642  -0.009    0.993
## log10_carboxyPMQ -1.68141    1.66438  -1.010    0.312
##
## Correlation of Fixed Effects:
##             (Intr)
## lg10_crbPMQ -0.994
## fit warnings:
## fixed-effect model matrix is rank deficient so dropping 1 column / coefficient
## convergence code: 0
## boundary (singular) fit: see ?isSingular
```

Compare results with and without outliers:

```
par(las = 1, bty='n')
plot(Combined_Time_Data$log10_carboxyPMQ[ind_keep]*Combined_Time_Data$NumberDaysPMQ[ind_keep],
     jitter(as.numeric(Combined_Time_Data$Failure_YN[ind_keep]),factor = 0.03),
     col = regimen_colors[mapvalues(Combined_Time_Data$NumberDaysPMQ[ind_keep],c(7,14),1:2)],
     pch = mapvalues(Combined_Time_Data$NumberDaysPMQ[ind_keep],c(7,14),c(3,4)),
     xlab = 'Carboxy-primaquine trough exposure: days * log(ng/mL)',
     ylab = 'Probability of failure', panel.first = grid())
legend(x = 25, y = 0.7, bty='n', col = c(1, regimen_colors, 1,1),
       pch=c(1,3,4,NA,NA), lty = c(NA,NA,NA,1,2),lwd=c(NA,NA,NA,2,2),
       legend = c('Outlier','7 days','14 days',
                  'Regression fit: all data', 'Regression fit: outliers removed'))
lines(xs*7, predict(mod, newdata=data.frame(log10_carboxyPMQ=xs, NumberDaysPMQ=7,
                                            patientid='new'),allow.new.levels=T,
                    type='response'), lwd=2, col= regimen_colors[1], lty=1)
lines(xs*14, predict(mod, newdata=data.frame(log10_carboxyPMQ=xs, NumberDaysPMQ=14,
                                             patientid='new'),allow.new.levels=T,
                     type='response'), lwd=2, col= regimen_colors[2], lty = 1)

lines(xs*7, predict(mod_No_Outliers, newdata=data.frame(log10_carboxyPMQ=xs, NumberDaysPMQ=7,
                                                        patientid='new'),allow.new.levels=T,
                    type='response'), lwd=2, col= regimen_colors[1], lty=2)
lines(xs*14, predict(mod_No_Outliers, newdata=data.frame(log10_carboxyPMQ=xs, NumberDaysPMQ=14,
```

```
                                                    patientid='new'),allow.new.levels=T,
                    type='response'), lwd=2, col= regimen_colors[2], lty = 2)

# outline the outliers
points(Combined_Time_Data$log10_carboxyPMQ[outliers14|outliers7]*Combined_Time_Data$NumberDaysPMQ[outli
       Combined_Time_Data$Failure_YN[outliers14|outliers7], cex=2)
```



Failures after PMQ+

Now we calculate a compressed dataset and failure for each individual

```
# now we calculate the primaquine failure rate
# For individuals with two episodes: P(failure) = 1 - P(Rec 1 = I)*P(Rec 2 = I)
Summary_data = Combined_Time_Data[!duplicated(Combined_Time_Data$patientid),]
```

```
Summary_data$N_typed = NA
for(id in unique(Summary_data$patientid)){
  if(!(id %in% MS_pooled$ID)){
    Summary_data$N_typed[Summary_data$patientid==id]=0
  } else {
    Summary_data$N_typed[Summary_data$patientid==id] =
      sum(apply(MS_pooled[MS_pooled$ID==id, MSs_all],2,
                function(x) sum(is.na(x))) == 0)
  }
}

Summary_data$Failure_UL = Summary_data$Failure_LL =
  Summary_data$Failure = Summary_data$CPMQ =
  Summary_data$CPMQ = NA
for(i in 1:nrow(Summary_data)){
  ind = which(Combined_Time_Data$patientid==Summary_data$patientid[i])
  Summary_data$Failure[i] = 1-prod(Combined_Time_Data$Reinfection_Probability[ind],na.rm=T)
  Summary_data$Failure_UL[i] = 1-prod(Combined_Time_Data$Reinfection_Probability_UL[ind],na.rm=T)
  Summary_data$Failure_LL[i] = 1-prod(Combined_Time_Data$Reinfection_Probability_LL[ind],na.rm=T)
  Summary_data$CPMQ[i] = median(Combined_Time_Data$log10_carboxyPMQ[ind],na.rm=T)
}
VHX_PMQ_data = filter(Summary_data, Study_Period==1, arm_num=='CHQ/PMQ')
BPD_data = filter(Summary_data, Study_Period==2)

P_Failure=100*sum(BPD_data$Failure)/nrow(BPD_data)
# invert the intervals here - optimistic for not failure = pessimistic for failure
P_Failure_UL = 100*sum(BPD_data$Failure_LL)/nrow(BPD_data)
P_Failure_LL = 100*sum(BPD_data$Failure_UL)/nrow(BPD_data)

writeLines(sprintf('In BPD, the primaquine failure rate in the %s individuals is %s%% (%s-%s) over the
                   nrow(BPD_data), round(P_Failure,1),
                   round(P_Failure_LL,1),
                   round(P_Failure_UL,1), round(sum(BPD_data$FU_time)/365)))
```

```
## In BPD, the primaquine failure rate in the 655 individuals is 3% (2.4-4) over the course of 522 years
```

```
P_Failure_VHX=100*sum(VHX_PMQ_data$Failure)/nrow(VHX_PMQ_data)
# invert the intervals here - optimistic for not failure = pessimistic for failure
P_Failure_UL_VHX = 100*sum(VHX_PMQ_data$Failure_LL)/nrow(VHX_PMQ_data)
P_Failure_LL_VHX = 100*sum(VHX_PMQ_data$Failure_UL)/nrow(VHX_PMQ_data)

writeLines(sprintf('In VHX, the primaquine failure rate in the %s individuals is %s%% (%s-%s) over the
                   nrow(VHX_PMQ_data), round(P_Failure_VHX,1),
                   round(P_Failure_LL_VHX,1),
                   round(P_Failure_UL_VHX,1), round(sum(VHX_PMQ_data$FU_time)/365)))
```

```
## In VHX, the primaquine failure rate in the 198 individuals is 2.4% (1.7-3.3) over the course of 155 y
```

```
mean(Combined_Time_Data$Reinfection_Probability[Combined_Time_Data$Study_Period==1 &
                                                  Combined_Time_Data$arm_num=='CHQ/PMQ' &
                                                  Combined_Time_Data$Censored==0],na.rm=T)
```

```
## [1] 0.9526366
```

```
mean(Combined_Time_Data$Reinfection_Probability[Combined_Time_Data$Study_Period==2 &
                                                  Combined_Time_Data$arm_num=='CHQ/PMQ' &
```

```
                                                    Combined_Time_Data$Censored==0],na.rm=T)
```

```
## [1] 0.973358
```

```
PMQ_data = filter(Summary_data, arm_num == 'CHQ/PMQ')

P_Failure_all=100*sum(PMQ_data$Failure)/nrow(PMQ_data)
# invert the intervals here - optimistic for not failure = pessimistic for failure
P_Failure_UL_all = 100*sum(PMQ_data$Failure_LL)/nrow(PMQ_data)
P_Failure_LL_all = 100*sum(PMQ_data$Failure_UL)/nrow(PMQ_data)
writeLines(sprintf('In all primaquine treated individuals, the failure rate in the %s individuals is %s
                   nrow(PMQ_data), round(P_Failure_all,1),
                   round(P_Failure_LL_all,1),
                   round(P_Failure_UL_all,1), round(sum(PMQ_data$FU_time)/365)))
```

```
## In all primaquine treated individuals, the failure rate in the 853 individuals is 2.9% (2.3-3.8) ove
```

The above failure rate is based on all available data. Next we consider rates based on adjustments using time and genetic only.

```
# First create columns into which probabilities can be stored
Combined_Time_Data$Reinfection_Probability_time_only =
  Combined_Time_Data$Reinfection_Probability_UL_time_only =
  Combined_Time_Data$Reinfection_Probability_LL_time_only = NA

# Iterate over every episode using the time probability
sss=0
for(i in 1:nrow(Combined_Time_Data)){

  # Extract episode id
  ep_id = Combined_Time_Data$Episode_Identifier[i]

  # we look one episode ahead (MS_id) and locate MS_id in time results
  MS_id = paste(Combined_Time_Data$patientid[i],as.integer(Combined_Time_Data$episode[i])+1, sep='_')
  ind = which(Mod2_ThetaEstimates$Failure_Identifier==MS_id)

  if(length(ind)>0){
    Combined_Time_Data$Reinfection_Probability_time_only[i] = Mod2_ThetaEstimates$ReInfection_mean_theta
    Combined_Time_Data$Reinfection_Probability_UL_time_only[i] = Mod2_ThetaEstimates$ReInfection_975_th
    Combined_Time_Data$Reinfection_Probability_LL_time_only[i] = Mod2_ThetaEstimates$ReInfection_025_th
    sss = sss + 1
  }
}


# now we calculate the primaquine failure rate for individuals with two episodes: P(failure) = 1 - P(Re
Summary_data$Failure_UL_time_only = Summary_data$Failure_LL_time_only = Summary_data$Failure_time_only=
for(i in 1:nrow(Summary_data)){
  ind = which(Combined_Time_Data$patientid==Summary_data$patientid[i])
  Summary_data$Failure_time_only[i] = 1-prod(Combined_Time_Data$Reinfection_Probability_time_only[ind],
  Summary_data$Failure_UL_time_only[i] = 1-prod(Combined_Time_Data$Reinfection_Probability_UL_time_only
  Summary_data$Failure_LL_time_only[i] = 1-prod(Combined_Time_Data$Reinfection_Probability_LL_time_only
}

# Filter s.t. only BPD data
```

```
BPD_data = Summary_data[grep('BPD', Summary_data$patientid),]

# now we add the primaquine failure rate for BPD individuals, using genetic data only
BPD_data$Failure_UL_gene_only = BPD_data$Failure_LL_gene_only = BPD_data$Failure_gene_only = 0
for(i in 1:nrow(BPD_data)){
  pid = BPD_data$patientid[i]
  ind = grep(paste0(pid,'_'), thetas_9MS_Tagnostic$Episode_Identifier)
  if(any(ind)){
    BPD_data$Failure_gene_only[i] = 1-prod(thetas_9MS_Tagnostic$I[ind])
    BPD_data$Failure_UL_gene_only[i] = 1-prod(thetas_9MS_Tagnostic$`I97.5%`[ind])
    BPD_data$Failure_LL_gene_only[i] = 1-prod(thetas_9MS_Tagnostic$`I2.5%`[ind])
  }
}

# Point estimate
P_Failure_time_only=100*sum(BPD_data$Failure_time_only)/nrow(BPD_data)
P_Failure_gene_only=100*sum(BPD_data$Failure_gene_only)/nrow(BPD_data)

# invert the intervals here - optimistic for not failure = pessimistic for failure
P_Failure_UL_time_only = 100*sum(BPD_data$Failure_LL_time_only)/nrow(BPD_data)
P_Failure_LL_time_only = 100*sum(BPD_data$Failure_UL_time_only)/nrow(BPD_data)
P_Failure_UL_gene_only = 100*sum(BPD_data$Failure_LL_gene_only)/nrow(BPD_data)
P_Failure_LL_gene_only = 100*sum(BPD_data$Failure_UL_gene_only)/nrow(BPD_data)

writeLines(sprintf('The primaquine failure rate, based on the joint model, in the %s individuals is %s%
                   nrow(BPD_data), round(P_Failure,2),
                   round(P_Failure_LL,2),
                   round(P_Failure_UL,2), round(sum(BPD_data$FU_time)/365)))
```

```
## The primaquine failure rate, based on the joint model, in the 655 individuals is 3.05% (2.43-4.02) o
```

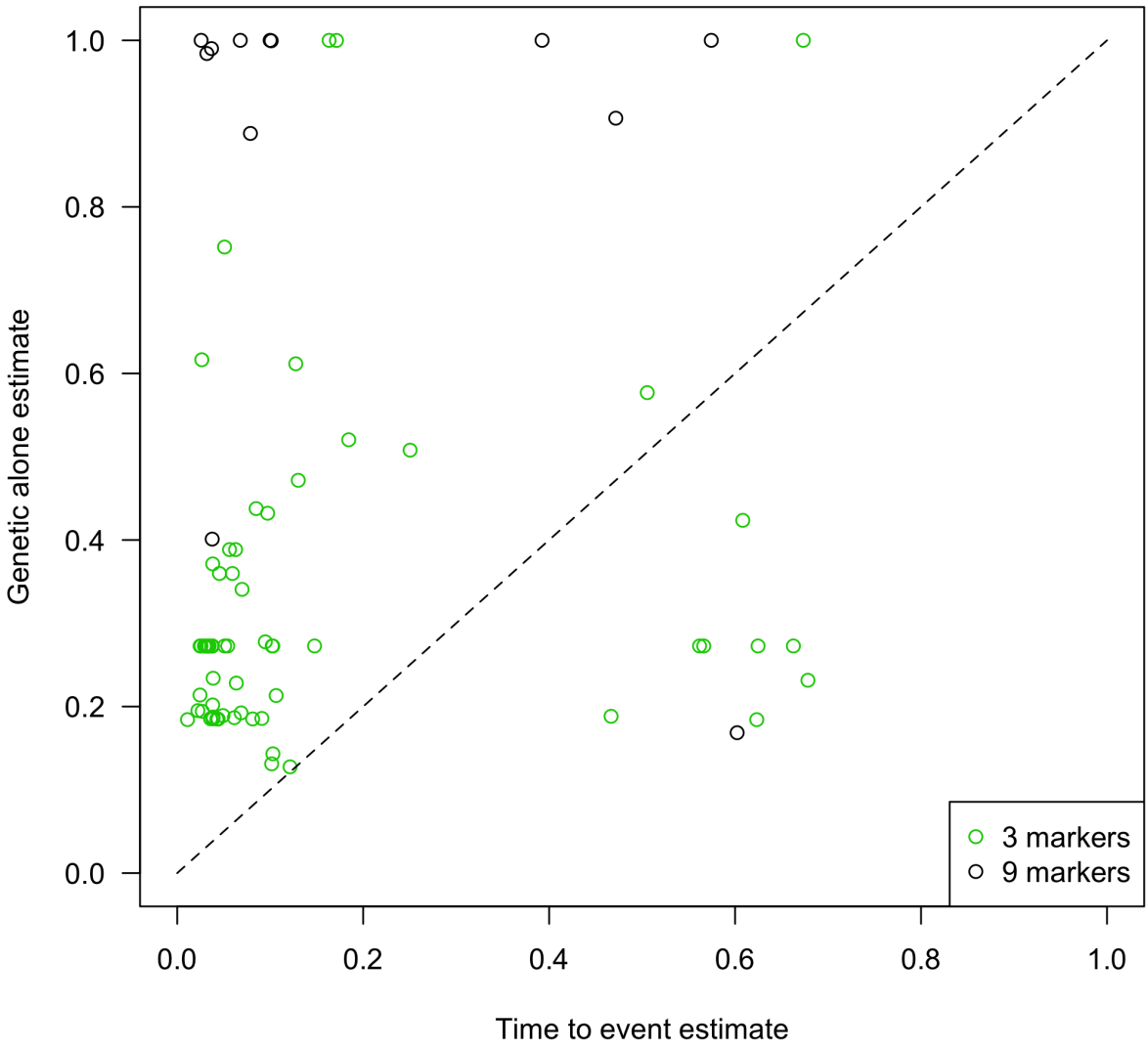
```
writeLines(sprintf('The primaquine failure rate, based on time-to-event model only, in the %s individua
                   nrow(BPD_data), round(P_Failure_time_only,2),
                   round(P_Failure_LL_time_only,2),
                   round(P_Failure_UL_time_only,2), round(sum(BPD_data$FU_time)/365)))
```

```
## The primaquine failure rate, based on time-to-event model only, in the 655 individuals is 2.27% (1.3
```

```
writeLines(sprintf('The primaquine failure rate, based on genetic model only, in the %s individuals is
                   nrow(BPD_data), round(P_Failure_gene_only,2),
                   round(P_Failure_LL_gene_only,2),
                   round(P_Failure_UL_gene_only,2), round(sum(BPD_data$FU_time)/365)))
```

```
## The primaquine failure rate, based on genetic model only, in the 655 individuals is 4.79% (4.68-4.88
```

```
par(las=1)
ind = BPD_data$Failure_gene_only>0
plot(BPD_data$Failure_time_only[ind],
     BPD_data$Failure_gene_only[ind],
     xlab='Time to event estimate', ylab = 'Genetic alone estimate',
     xlim=c(0,1),ylim=c(0,1), col  = BPD_data$N_typed[ind])
lines(0:1,0:1,lty=2)
legend('bottomright', pch = 1, col = c(3,9), legend = c('3 markers', '9 markers'))
```



save augmented summary data for other analyses

```
save(Summary_data, file = '../RData/Summary_data_ModelResults.RData')
```

Extra Analyses

Looking at the effect of inbreeding coefficient

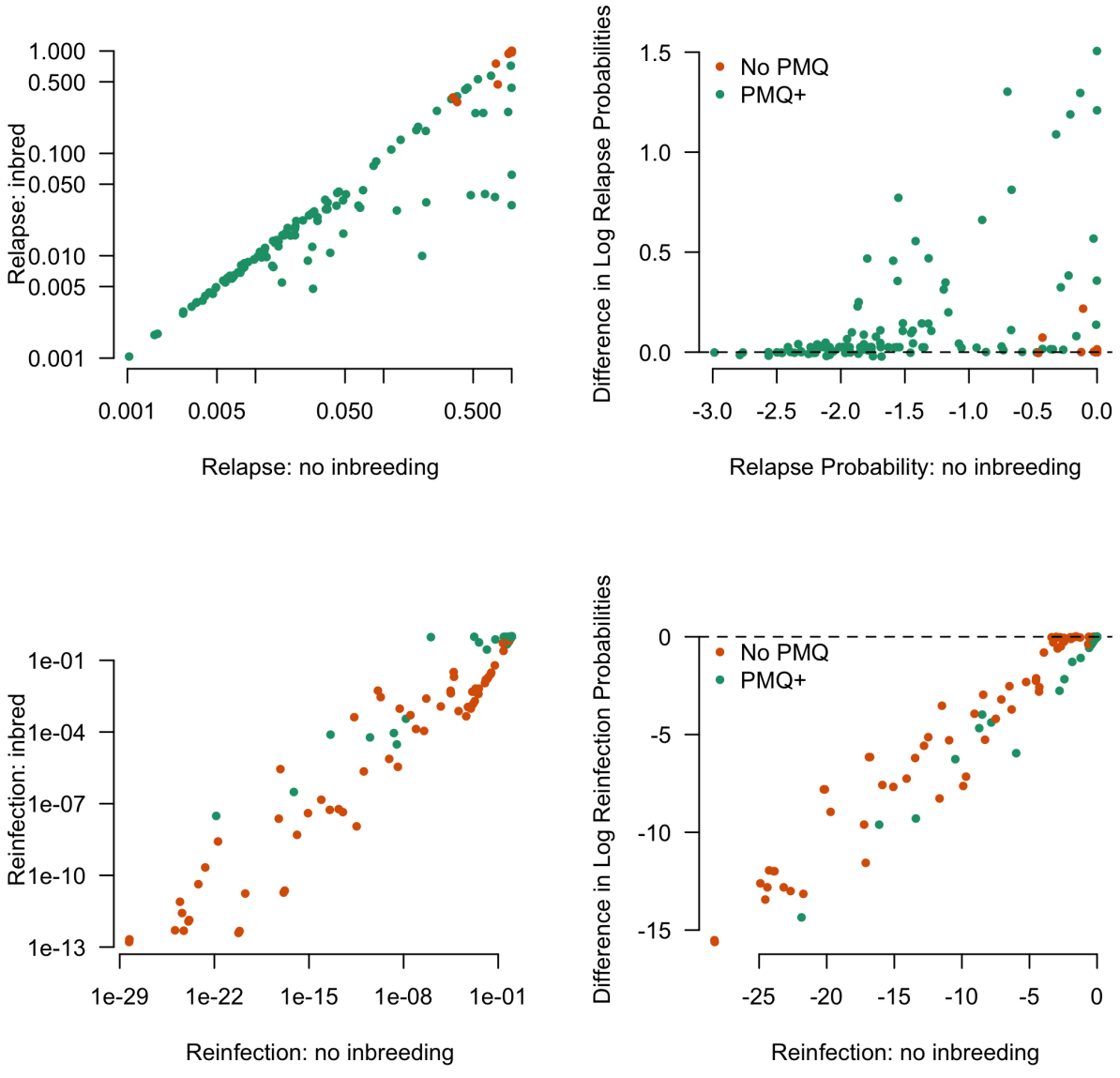
Our model has a parameter α which defines the level of inbreeding within the population. Taylor is developing methods for the estimation of α from genetic data (in preparation).

We look at the sensitivity of the results (all the above is with $\alpha = 0$) for a reasonable upper bound of $\alpha = 0.175$.

We rerun the analysis on the single run isolates (low computational complexity):

```
alphaUpper = 0.175
f_name = paste(f_name_prefix, 'thetas_9MS_alphaUpper.RData', sep='_')
if(RUN_MODELS_FULL_POSTERIOR){
  #============================================
  # Run (with time-to-event)
  #============================================
  # Approx 100 secs per full model run
  tic()
  thetas_9MS_alphaUpper = post_prob_CLI(MSdata = MS_pooled, Fs = Fs_Combined,
                                        p = p, cores = 6, verbose = F, alpha = alphaUpper)
  thetas_9MS_alphaUpper$Episode_Identifier = rownames(thetas_9MS_alphaUpper)
  save(thetas_9MS_alphaUpper, file = f_name)
  toc()
} else {
  load(f_name)
}
```

```
par(las=1, bty='n', mfrow=c(2,2))
plot(thetas_9MS$L,
     thetas_9MS_alphaUpper$L,
     log = 'xy',
     ylab = 'Relapse: inbred',
     xlab = 'Relapse: no inbreeding',
     col= drug_cols2[thetas_9MS$drug],pch=20)
lines(c(-10,10),c(-10,10),lty=2)

plot(log10(thetas_9MS$L), log10(thetas_9MS$L)-log10(thetas_9MS_alphaUpper$L),
     ylab = 'Difference in Log Relapse Probabilities',
     xlab = 'Relapse Probability: no inbreeding',
     col= drug_cols2[thetas_9MS$drug],pch=20)
abline(h=0,lty=2)
legend('topleft', legend = c('No PMQ', 'PMQ+'), col = drug_cols2[2:3], pch = 20, bty = 'n')

######### Reinfection : comparison ******#####
par(las=1, bty='n')
plot(thetas_9MS$I, thetas_9MS_alphaUpper$I,
     log = 'xy',
     ylab = 'Reinfection: inbred',
     xlab = 'Reinfection: no inbreeding',
     col= drug_cols2[thetas_9MS$drug],pch=20)
lines(c(-10,10),c(-10,10),lty=2)

plot(log10(thetas_9MS$I), log10(thetas_9MS$I)-log10(thetas_9MS_alphaUpper$I),
     ylab = 'Difference in Log Reinfection Probabilities',
     xlab = 'Reinfection: no inbreeding',
     col= drug_cols2[thetas_9MS$drug],pch=20)
legend('topleft', legend = c('No PMQ', 'PMQ+'), col = drug_cols2[2:3], pch = 20, bty = 'n')

abline(h=0,lty=2)
```




```
toc()
```

```
## 33.214 sec elapsed
```

Interpretation: Adding the inbreeding coefficent slightly changes some of the probabilities of relapse for some primaquine treated individuals (only green dots are being shifted).

This means that inbreeding would imply that fewer of the primaquine treated episodes are relapses, implying higher efficacy of the drug.

For the non-primaquine group, it is just tempering the very low probabilities of reinfection seen for some episodes.

In conclusion, this isn't changing the results significantly and would imply a greater primaquine efficacy than reported in the paper.