Untitled

Load data

Load datasets and mark DLI patients. Check number of clonotypes in donors and receptients - we have engough clones for statistics everywhere. We group clones by their abundance in donors: 1 (singletons), 2 (doubletons), 3 (tripletons, as good as doubletons, no need to hate them Vanya) and 4+ reads (Large). The choice is dictated by observing the fact that for rare events Poisson distribution shows huge difference in capture probability for $\lambda \in [1, 3]$ while smaller λ values are unlikely to be encountered and quantified. Moreover, for large clones, each hyperexpanded variant has its own history and, likely, its own dynamic, so binning them to different bins based on minor differences in frequency (e.g. 0.1% vs 0.01%) makes little sense.

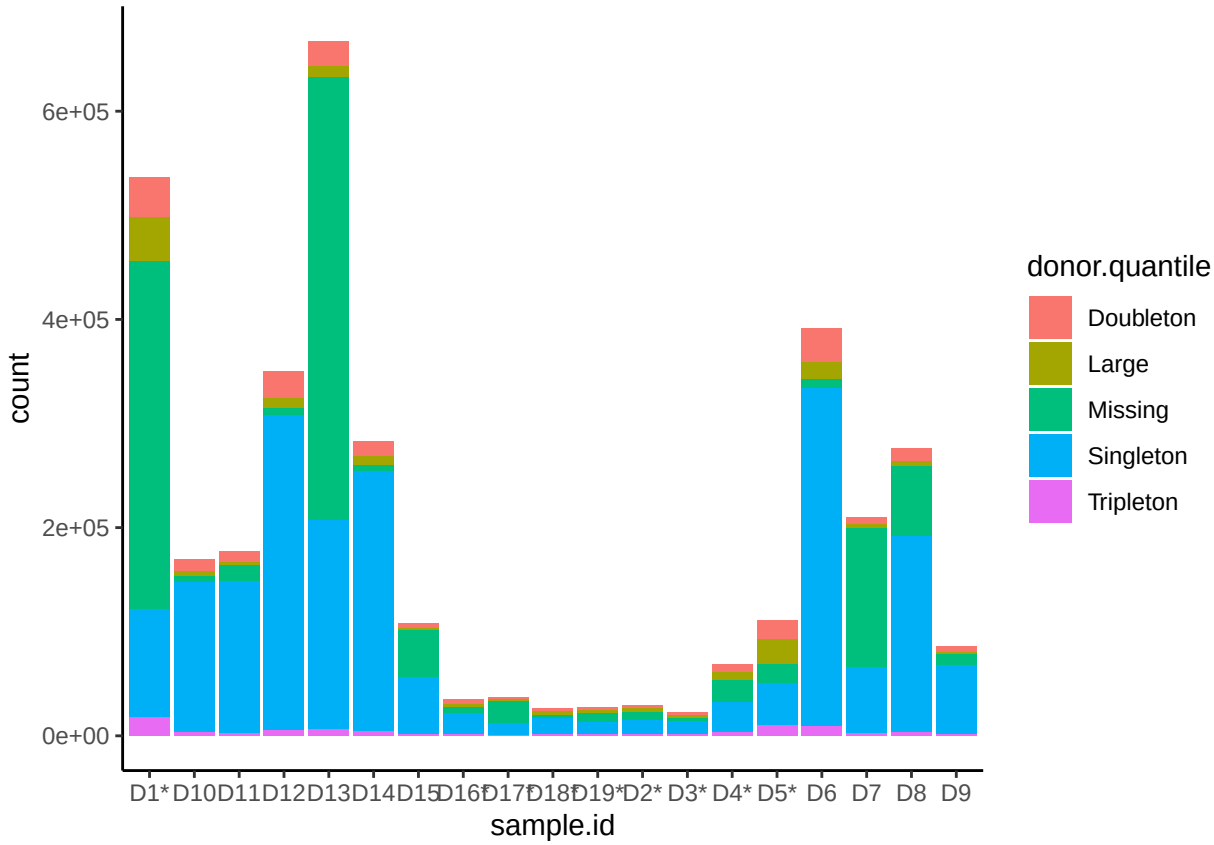
```
data <- list.files("data", full.names = T) %>%
  as.list %>%
  lapply(function(x) read_gz(x) %>% mutate(sample.id = x)) %>%
  rbindlist %>%
  mutate(sample.id.old = sample.id,
         dli = !str_starts(sample.id.old, fixed("data/sh.p")),
         sample.id = paste0("D", sample.id %>% as.factor %>% as.integer, ifelse(dli, "*", ""))) #%>%
```

```
## Taking input= as a system command ('zcat data/sh.Art.txt.gz') and a variable has been used in the exp
## Taking input= as a system command ('zcat data/sh.Bat.txt.gz') and a variable has been used in the exp
## Taking input= as a system command ('zcat data/sh.Hus.txt.gz') and a variable has been used in the exp
## Taking input= as a system command ('zcat data/sh.Kim.txt.gz') and a variable has been used in the exp
## Taking input= as a system command ('zcat data/sh.Make.txt.gz') and a variable has been used in the ex
## Taking input= as a system command ('zcat data/sh.p1005.txt.gz') and a variable has been used in the e
## Taking input= as a system command ('zcat data/sh.p1321.txt.gz') and a variable has been used in the e
## Taking input= as a system command ('zcat data/sh.p1694.txt.gz') and a variable has been used in the e
## Taking input= as a system command ('zcat data/sh.p1772.txt.gz') and a variable has been used in the e
## Taking input= as a system command ('zcat data/sh.p2768.txt.gz') and a variable has been used in the e
## Taking input= as a system command ('zcat data/sh.p2846.txt.gz') and a variable has been used in the e
## Taking input= as a system command ('zcat data/sh.p3514.txt.gz') and a variable has been used in the e
## Taking input= as a system command ('zcat data/sh.p3570.txt.gz') and a variable has been used in the e
## Taking input= as a system command ('zcat data/sh.p3602.txt.gz') and a variable has been used in the e
## Taking input= as a system command ('zcat data/sh.p754.txt.gz') and a variable has been used in the ex
## Taking input= as a system command ('zcat data/sh.Ser.txt.gz') and a variable has been used in the exp
## Taking input= as a system command ('zcat data/sh.Str.txt.gz') and a variable has been used in the exp
## Taking input= as a system command ('zcat data/sh.Zav.txt.gz') and a variable has been used in the exp
## Taking input= as a system command ('zcat data/sh.Zuk.txt.gz') and a variable has been used in the exp
```

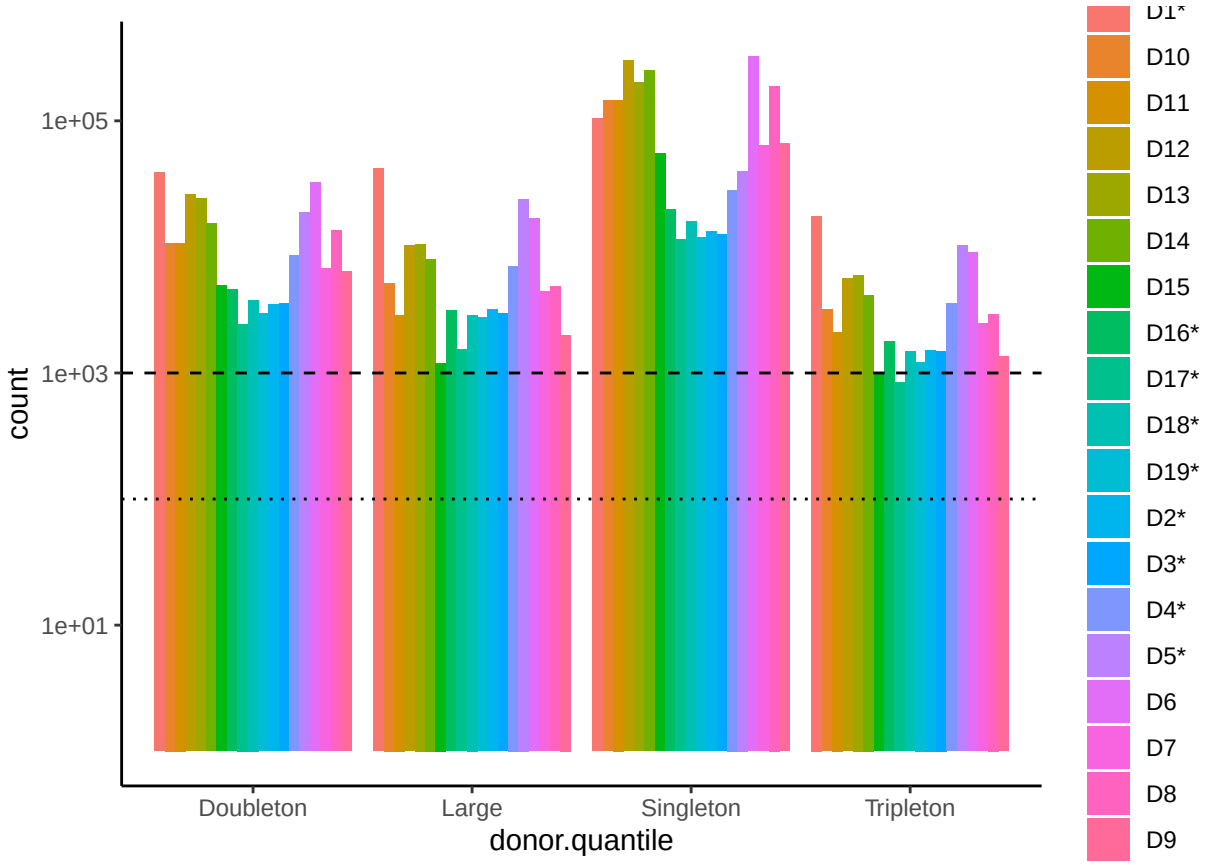
```
  #filter(dli)

data %>%
  select(sample.id.old, dli, sample.id) %>%
  unique
```

```
##            sample.id.old   dli sample.id
##  1:   data/sh.Art.txt.gz  TRUE       D1*
##  2:   data/sh.Bat.txt.gz  TRUE       D2*
##  3:   data/sh.Hus.txt.gz  TRUE       D3*
##  4:   data/sh.Kim.txt.gz  TRUE       D4*
##  5:  data/sh.Make.txt.gz  TRUE       D5*
##  6: data/sh.p1005.txt.gz FALSE        D6
##  7: data/sh.p1321.txt.gz FALSE        D7
##  8: data/sh.p1694.txt.gz FALSE        D8
##  9: data/sh.p1772.txt.gz FALSE        D9
## 10: data/sh.p2768.txt.gz FALSE       D10
## 11: data/sh.p2846.txt.gz FALSE       D11
## 12: data/sh.p3514.txt.gz FALSE       D12
## 13: data/sh.p3570.txt.gz FALSE       D13
## 14: data/sh.p3602.txt.gz FALSE       D14
## 15:  data/sh.p754.txt.gz FALSE       D15
## 16:   data/sh.Ser.txt.gz  TRUE      D16*
## 17:   data/sh.Str.txt.gz  TRUE      D17*
## 18:   data/sh.Zav.txt.gz  TRUE      D18*
## 19:   data/sh.Zuk.txt.gz  TRUE      D19*
```

```
data <- data %>%
  mutate(donor.quantile = case_when(
    is.na(cloneCount.don) ~ "Missing",
    cloneCount.don == 1 ~ "Singleton",
    cloneCount.don == 2 ~ "Doubleton",
    cloneCount.don == 3 ~ "Tripleton",
    T ~ "Large"
    ))

data %>%
  ggplot(aes(x = sample.id, fill = donor.quantile)) +
  geom_bar() +
  theme_classic()
```




```
data %>%
  filter(donor.quantile != "Missing") %>%
  ggplot(aes(x = donor.quantile,
             fill = sample.id)) +
  geom_bar(position = "dodge") +
  scale_y_log10() +
  geom_hline(yintercept = 100, linetype = "dotted") +
  geom_hline(yintercept = 1000, linetype = "dashed") +
  theme_classic()
```



We split our donor dataset into singletons, doubletons, tripletons and higher-order clonotypes. Each of these subsets contains enough clones to reliably estimate the probability of recapturing a clonotype from a given subset of donor clonotypes. Interestingly, the ration between recapturing probabilities of singletons, doubletons and tripletons is in line with exponential difference stemming from Poisson distribution.

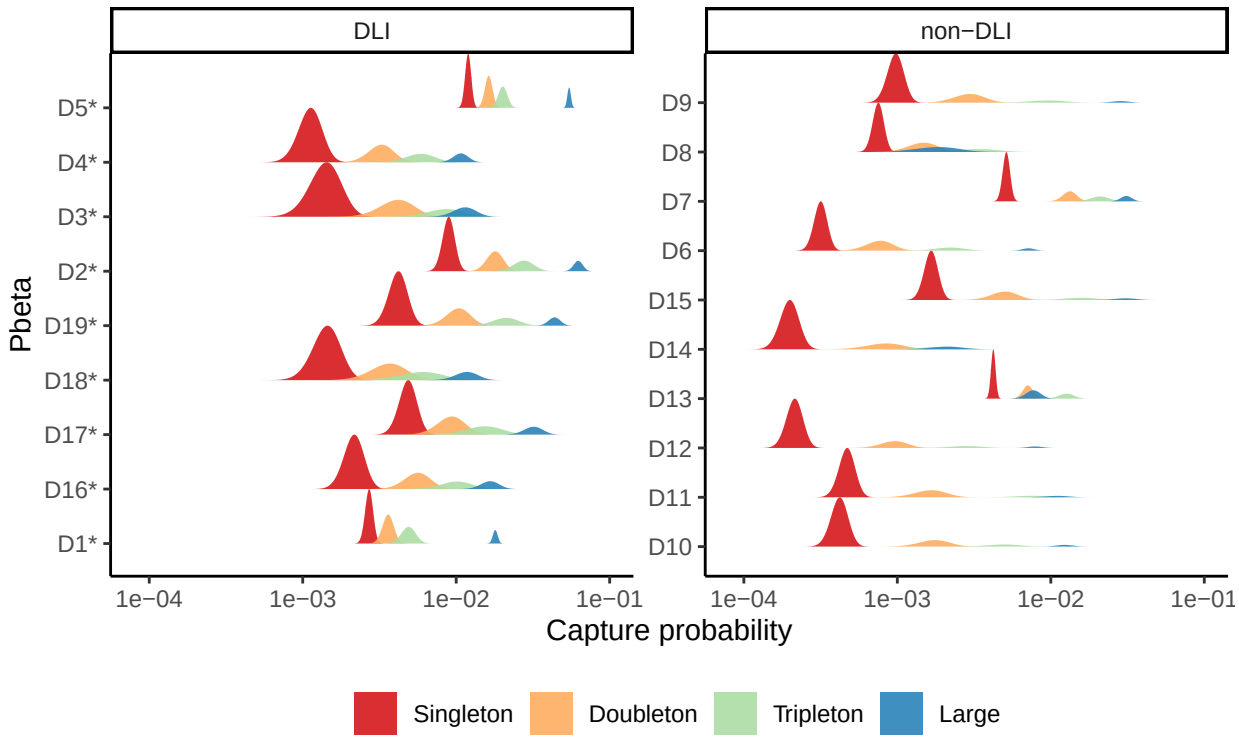
```
# summarize & estimate parameters of beta distribution
alpha.prior <- 1
beta.prior <- 1
data.s <- data %>%
  filter(donor.quantile != "Missing") %>%
  group_by(sample.id) %>%
  mutate(total.don = sum(cloneCount.don, na.rm = T),
         clones.don = length(unique(aaSeqCDR3.don)), # note here we count +1 for NA, maybe should modif
         total.rec = sum(cloneCount.rec, na.rm = T),
         clones.rec = length(unique(aaSeqCDR3.rec))) %>%
  group_by(sample.id, donor.quantile) %>%
  mutate(clones.don.quant = length(unique(aaSeqCDR3.don))) %>%
  group_by(dli, sample.id, donor.quantile, total.don, clones.don, total.rec, clones.rec, clones.don.quan
  summarize(alpha = sum(!is.na(cloneCount.rec)) + alpha.prior,
            beta = sum(is.na(cloneCount.rec)) + beta.prior) %>%
  ungroup
```

```
## `summarise()` regrouping output by 'dli', 'sample.id', 'donor.quantile', 'total.don', 'clones.don',
```

```
data.sp <- data.s %>%
  merge(tibble(p = c(0:1000/1000, 10^(-4000:-1000/1000)))) %>%
  group_by(sample.id, donor.quantile) %>%
  mutate(Pbeta = dbeta(p, alpha, beta)) %>%
```

```
  ungroup

data.sp %>%
  mutate(dli = ifelse(dli, "DLI", "non-DLI")) %>%
  group_by(sample.id) %>%
  mutate(height = Pbeta / max(Pbeta)) %>%
  ggplot(aes(x = p, y = sample.id, height = height,
             fill = factor(donor.quantile, levels = c("Singleton",
                                                      "Doubleton",
                                                      "Tripleton",
                                                      "Large"))
             )) +
  geom_ridgeline(color = NA, alpha = 0.9) +
  scale_x_log10("Capture probability", limits = c(0.8e-4, 1e-1)) + ylab("Pbeta") +
  scale_fill_brewer("", palette = "Spectral") +
  facet_wrap(~dli, scales = "free_y") +
  theme_classic() +
  theme(aspect = 1, legend.position = "bottom")
```

```
## Warning: Transformation introduced infinite values in continuous x-axis
```

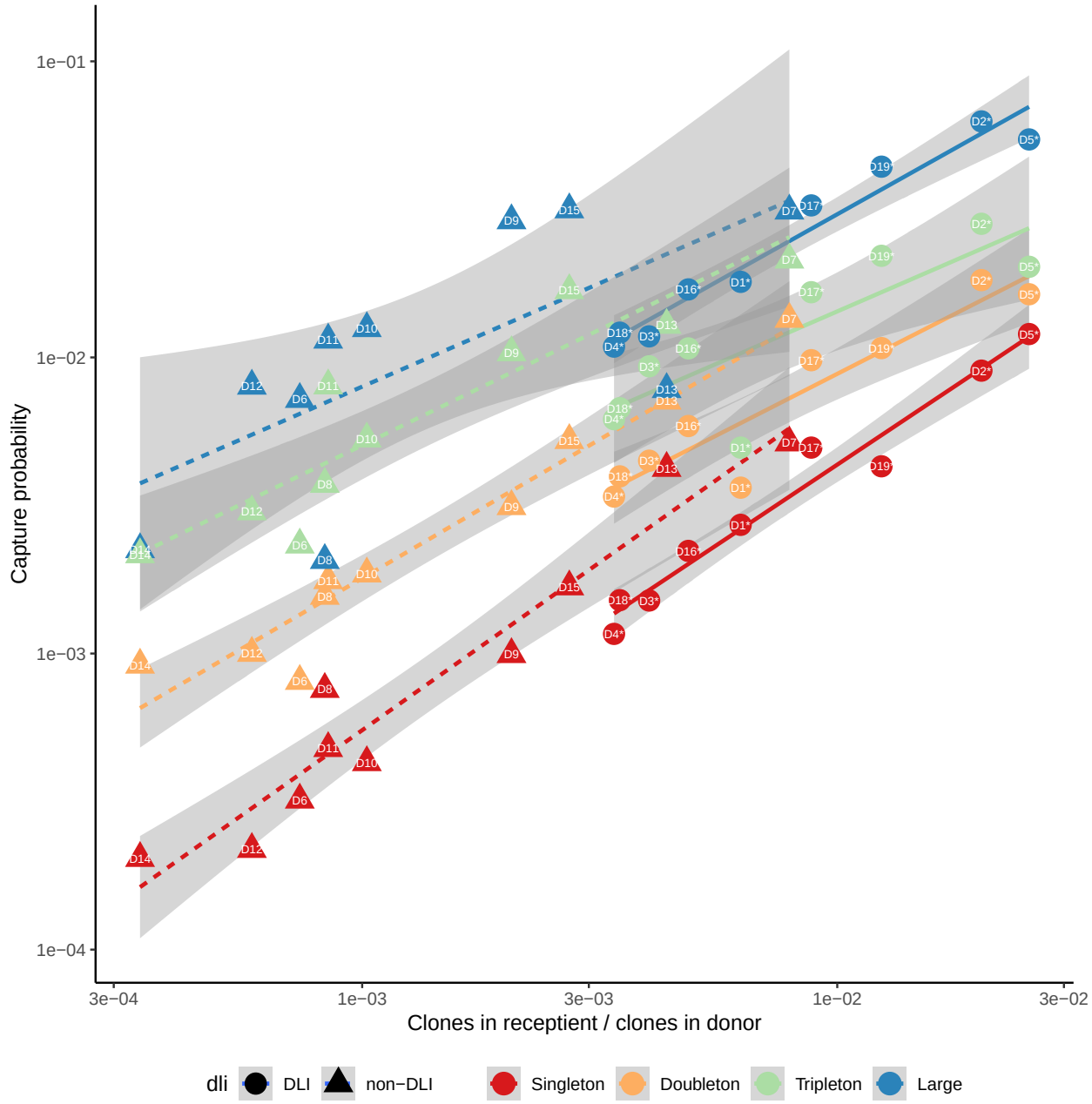


Interestingly, the TCR recovery rate is related both to the total number of clones in donor and recipient (* - VANYA write an explainaton based on what I've told you yesterdat)

```
data.s %>%
  mutate(dli = ifelse(dli, "DLI", "non-DLI")) %>%
  ggplot(aes(x = clones.rec / clones.don, y = alpha / (alpha + beta),
             group = paste(dli, donor.quantile),
             linetype = dli, shape = dli,
             color = factor(donor.quantile, levels = c("Singleton",
                                                       "Doubleton",
```

```
                                                "Tripleton",
                                                "Large")))) +
  geom_smooth(method = "lm", aes(), size = 1) +
  geom_point(size = 5) +
  geom_text(aes(label = sample.id), size = 2, color = "white") +
  scale_x_log10("Clones in receptient / clones in donor") +
  scale_y_log10("Capture probability") +
  scale_color_brewer("", palette = "Spectral") +
  theme_classic() +
  theme(aspect = 1, legend.position = "bottom")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Quantifying the effect of various factors – number of clones detected in donor, number of clones detected

in receptient and the frequency quantile of a given clonotype in donor – on the recapture probability. Log-transformed variables show extremely high correlation.

```
data.coord <- data.s %>%
  group_by(donor.quantile) %>%
  mutate(logRecaptureProb = log(alpha / (alpha + beta)),
         logClonesRecepient = log(clones.rec),
         logClonesDonor = log(clones.don)) %>%
  ungroup %>%
  mutate(donor.quantile = factor(donor.quantile, levels = c("Singleton",
                                                            "Doubleton",
                                                            "Tripleton",
                                                            "Large")))
```

Show coefficients of linear model

```
data.coord %>%
  ungroup %>%
  mutate(donor.quantile = as.factor(donor.quantile)) %>%
  do(lm(.$logRecaptureProb ~ .$donor.quantile + .$dli + .$logClonesRecepient + .$logClonesDonor) %>% ti
```

```
## # A tibble: 7 x 5
##   term                       estimate std.error statistic  p.value
##   <chr>                         <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)                   1.08     0.753       1.44 1.55e- 1
## 2 .$donor.quantileDoubleton     0.959    0.122       7.88 3.33e-11
## 3 .$donor.quantileTripleton     1.72     0.122      14.2  4.08e-22
## 4 .$donor.quantileLarge         2.28     0.122      18.8  8.52e-29
## 5 .$dliTRUE                    -0.758    0.143      -5.31 1.24e- 6
## 6 .$logClonesRecepient          0.827    0.0541     15.3  7.33e-24
## 7 .$logClonesDonor             -1.04     0.0705    -14.8  4.44e-23
```

Show variance explained (ANOVA)

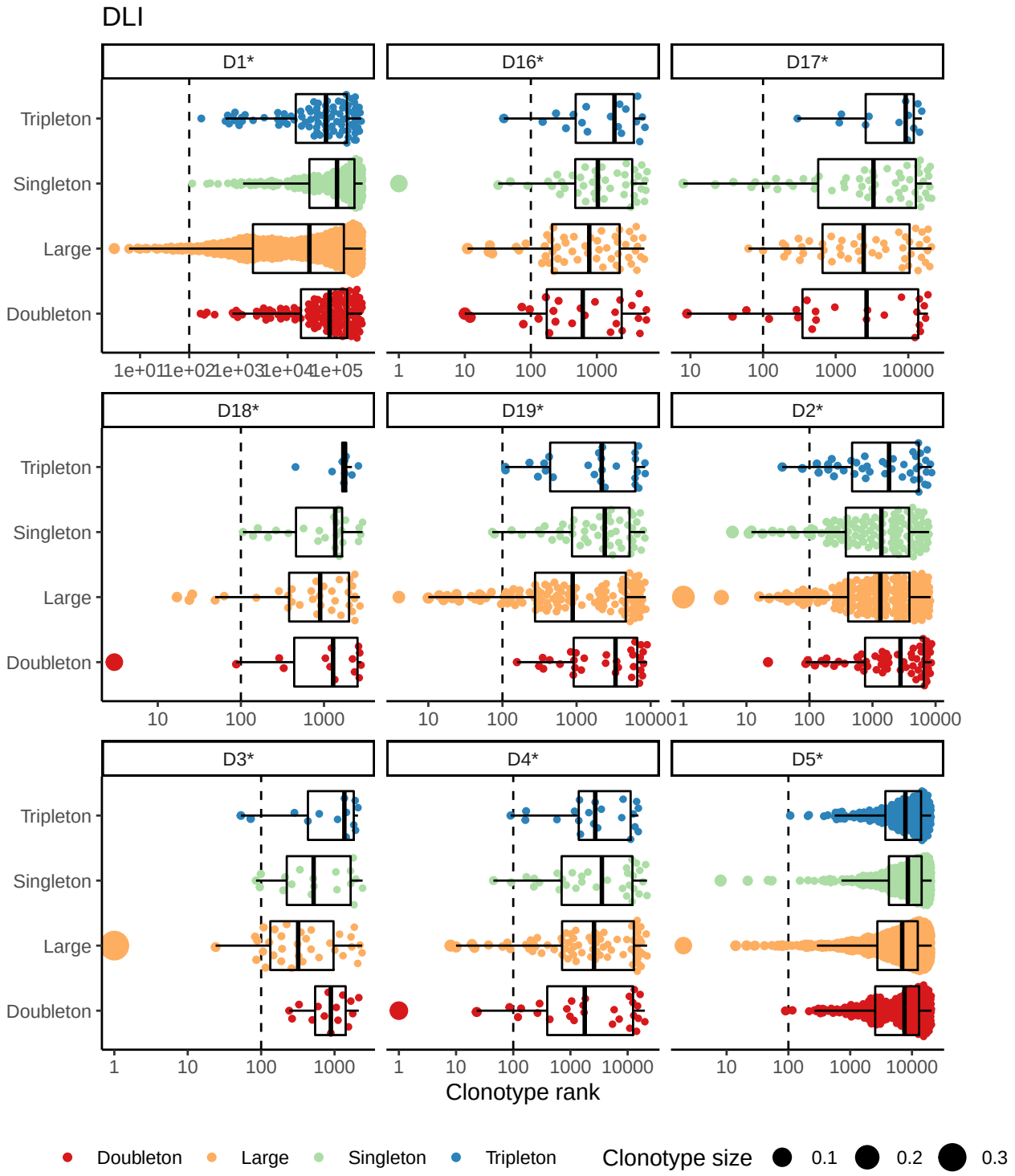
```
data.coord %>%
  ungroup %>%
  mutate(donor.quantile = as.factor(donor.quantile)) %>%
  do(lm(.$logRecaptureProb ~ .$donor.quantile + .$dli + .$logClonesRecepient + .$logClonesDonor) %>% ao
  mutate(var.explained.pct = sumsq / sum(sumsq) * 100)
```

```
## # A tibble: 5 x 7
##   term                    df sumsq meansq statistic   p.value var.explained.pct
##   <chr>                <dbl> <dbl>  <dbl>     <dbl>     <dbl>             <dbl>
## 1 .$donor.quantile         3 55.8   18.6       132.   1.52e-28             43.9
## 2 .$dli                    1 18.8   18.8       134.   8.55e-18             14.8
## 3 .$logClonesRecepient     1 12.0   12.0        85.3  1.12e-13              9.44
## 4 .$logClonesDonor         1 30.7   30.7       219.   4.44e-23             24.2
## 5 Residuals               69  9.70  0.141      NA    NA                    7.64
```

Origin of clones found in recepient: number of highly expanded clones that originated from expanded donor clones and rare donor clones varies and depends on donor.

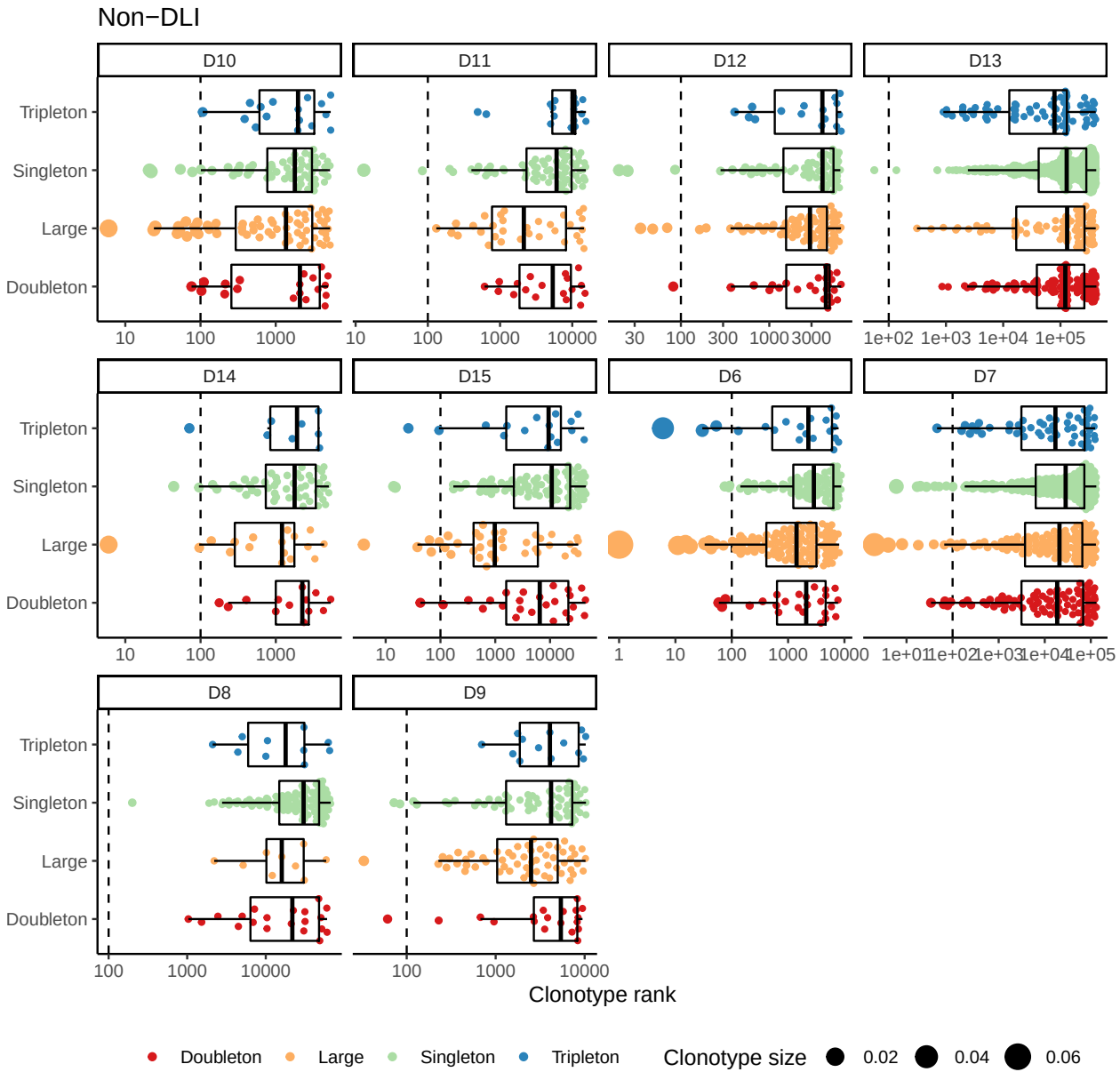
```
data %>%
  filter(dli & !is.na(cloneCount.rec)) %>%
  group_by(sample.id) %>%
  mutate(rank = rank(-cloneCount.rec, ties.method   = "first"),
         freq.rec = cloneCount.rec / sum(cloneCount.rec)) %>%
  filter(donor.quantile != "Missing") %>%
```

```
ggplot(aes(x = donor.quantile, y = rank)) +
geom_hline(yintercept = 100, linetype = "dashed") +
geom_quasirandom(aes(size = freq.rec, color = donor.quantile)) +
geom_boxplot(fill = NA, color = "black", outlier.colour = NA) +
coord_flip() +
scale_y_log10("Clonotype rank") +
xlab("") +
scale_size_continuous("Clonotype size") +
scale_color_brewer("", palette = "Spectral") +
facet_wrap(.~sample.id, scales = "free_x") +
theme_classic() +
theme(aspect = 1, legend.position = "bottom") +
ggtitle("DLI")
```




```
data %>%
  filter(!dli & !is.na(cloneCount.rec)) %>%
  group_by(sample.id) %>%
  mutate(rank = rank(-cloneCount.rec, ties.method   = "first"),
         freq.rec = cloneCount.rec / sum(cloneCount.rec)) %>%
  filter(donor.quantile != "Missing") %>%
  ggplot(aes(x = donor.quantile, y = rank)) +
  geom_hline(yintercept = 100, linetype = "dashed") +
```

```
  geom_quasirandom(aes(size = freq.rec, color = donor.quantile)) +
  geom_boxplot(fill = NA, color = "black", outlier.colour = NA) +
  coord_flip() +
  scale_y_log10("Clonotype rank") +
  xlab("") +
  scale_size_continuous("Clonotype size") +
  scale_color_brewer("", palette = "Spectral") +
  facet_wrap(.~sample.id, scales = "free_x") +
  theme_classic() +
  theme(aspect = 1, legend.position = "bottom") +
  ggtitle("Non-DLI")
```



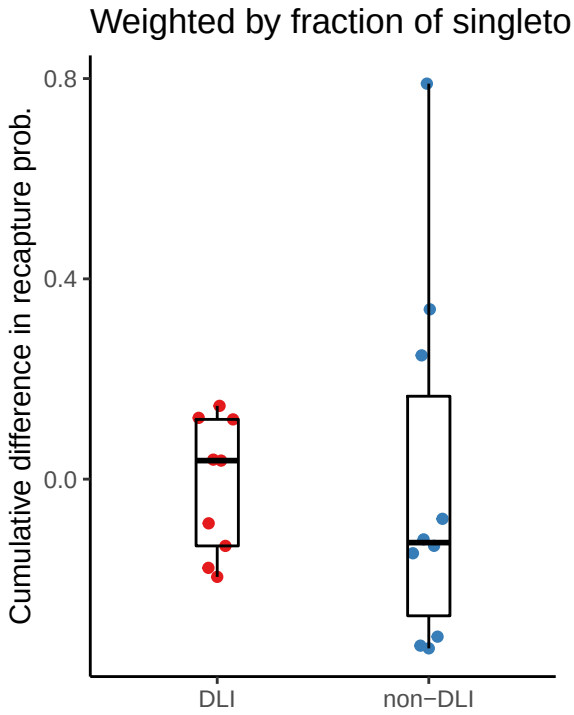
Examples of correcting sampling probability to various factors

Here is an example on how we can correct for sampling probability based on sample diversities and clonotype size and compare between DLI and non-DLI donors. Here we recompute a single $\Delta p = p_{observed} - p_{predicted}$ for every sample, i.e. if donor sample contains (by percent) ϕ_1 singletons, ϕ_2 doubletons, etc and difference for singletons is Δp_i - we compute a weighted sum $\Delta p = \sum_i \phi_i \Delta p_i$. Actually works not that great, however, if we treat Δp_i separately for each sample we are artificially boosting the number of "samples" for statistical testing. An alternative would be to look separately at singletons, doubletons, etc (I think multiple testing can be omitted here as we do like 2-3 tests at most).

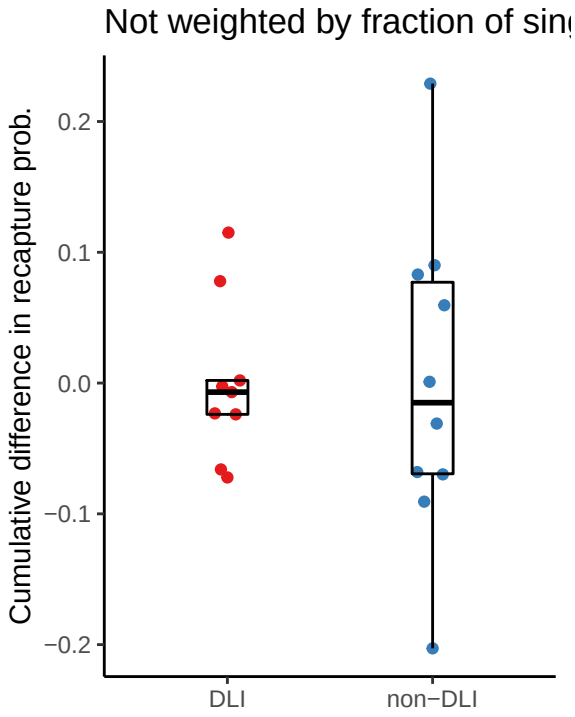
```
data.dli.pred <- data.coord %>% filter(dli)
lm(logRecaptureProb ~ donor.quantile + logClonesRecepient + logClonesDonor,
   data = data.dli.pred) -> lm.dli
data.dli.pred$logRecaptureProbPred <- predict(lm.dli, data.dli.pred)

data.ndli.pred <- data.coord %>% filter(!dli)
lm(logRecaptureProb ~ donor.quantile + logClonesRecepient + logClonesDonor,
   data = data.ndli.pred) -> lm.ndli
data.ndli.pred$logRecaptureProbPred <- predict(lm.ndli, data.ndli.pred)

rbind(data.dli.pred,
      data.ndli.pred) %>%
  group_by(sample.id) %>%
  mutate(clones.don.quant.frac = clones.don.quant / sum(clones.don.quant),
         delta = logRecaptureProb - logRecaptureProbPred) %>%
  group_by(sample.id, dli) %>%
  summarise(diff = sum(delta * clones.don.quant.frac),
            diff.unweighted = mean(delta),
            diff.singl = sum(delta * (donor.quantile == "Singleton"))) -> data.delta.summ
```

```
## `summarise()` regrouping output by 'sample.id' (override with `.groups` argument)
```

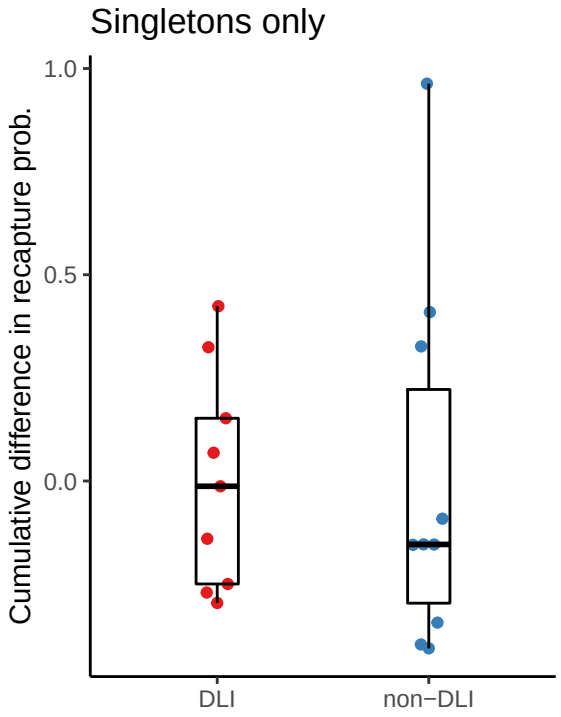
```
data.delta.summ %>%
  mutate(dli = ifelse(dli, "DLI", "non-DLI")) %>%
  ggplot(aes(x = dli, y = diff, color = dli)) +
  geom_quasirandom(width = 0.1) +
  geom_boxplot(width = 0.2, color = "black", fill = NA, outlier.colour = NA) +
  scale_color_brewer(guide = F, palette = "Set1") +
  xlab("") + ylab("Cumulative difference in recapture prob.") +
  ggtitle("Weighted by fraction of singletons, doubletons, etc") +
  theme_classic()
```




```
data.delta.summ %>%
  mutate(dli = ifelse(dli, "DLI", "non-DLI")) %>%
  ggplot(aes(x = dli, y = diff.unweighted, color = dli)) +
  geom_quasirandom(width = 0.1) +
  geom_boxplot(width = 0.2, color = "black", fill = NA, outlier.colour = NA) +
  scale_color_brewer(guide = F, palette = "Set1") +
  xlab("") + ylab("Cumulative difference in recapture prob.") +
  ggtitle("Not weighted by fraction of singletons, doubletons, etc") +
  theme_classic()
```




```
data.delta.summ %>%
  mutate(dli = ifelse(dli, "DLI", "non-DLI")) %>%
  ggplot(aes(x = dli, y = diff.singl, color = dli)) +
  geom_quasirandom(width = 0.1) +
  geom_boxplot(width = 0.2, color = "black", fill = NA, outlier.colour = NA) +
  scale_color_brewer(guide = F, palette = "Set1") +
  xlab("") + ylab("Cumulative difference in recapture prob.") +
  ggtitle("Singletons only") +
  theme_classic()
```



Some model comparison examples using ANOVA

```
mdl.1 <- lm(logRecaptureProb ~ donor.quantile + dli + logClonesRecepient + logClonesDonor,
            data = data.coord %>% ungroup %>% mutate(donor.quantile = as.factor(donor.quantile)))
mdl.1
```

```
##
## Call:
## lm(formula = logRecaptureProb ~ donor.quantile + dli + logClonesRecepient +
##     logClonesDonor, data = data.coord %>% ungroup %>% mutate(donor.quantile = as.factor(donor.quanti
##
## Coefficients:
##             (Intercept)  donor.quantileDoubleton  donor.quantileTripleton
##                  1.0820                   0.9585                   1.7241
##     donor.quantileLarge                  dliTRUE       logClonesRecepient
##                  2.2817                  -0.7584                   0.8273
##          logClonesDonor
##                 -1.0423
```

```
mdl.2 <- lm(logRecaptureProb ~ donor.quantile + logClonesRecepient + logClonesDonor,
            data = data.coord %>% ungroup %>% mutate(donor.quantile = as.factor(donor.quantile)))
mdl.2
```

```
##
## Call:
## lm(formula = logRecaptureProb ~ donor.quantile + logClonesRecepient +
##     logClonesDonor, data = data.coord %>% ungroup %>% mutate(donor.quantile = as.factor(donor.quanti
##
## Coefficients:
##             (Intercept)  donor.quantileDoubleton  donor.quantileTripleton
##                 -1.9379                   0.9585                   1.7241
##     donor.quantileLarge       logClonesRecepient           logClonesDonor
##                  2.2817                   0.7008                  -0.7445
```

```
anova(mdl.1, mdl.2)
```

```
## Analysis of Variance Table
##
## Model 1: logRecaptureProb ~ donor.quantile + dli + logClonesRecepient +
##     logClonesDonor
## Model 2: logRecaptureProb ~ donor.quantile + logClonesRecepient + logClonesDonor
##   Res.Df     RSS Df Sum of Sq      F    Pr(>F)
## 1     69  9.7022
## 2     70 13.6739 -1   -3.9717 28.246 1.242e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Bayesian modelling (can also use smth like http://mjskay.github.io/tidybayes/articles/tidy-brms.html)

```
brm(logRecaptureProb ~ donor.quantile+ logClonesRecepient + logClonesDonor + dli,
            data = data.coord %>% ungroup %>% mutate(donor.quantile = as.factor(donor.quantile))) -> mdl
```

```
## Compiling Stan program...
```

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

```
## Running /usr/lib/R/bin/R CMD SHLIB foo.c
## gcc -std=gnu99 -I"/usr/share/R/include" -DNDEBUG   -I"/home/mikesh/.lib/R/library/Rcpp/include/"  -I
## In file included from /home/mikesh/.lib/R/library/RcppEigen/include/Eigen/Core:88:0,
##                  from /home/mikesh/.lib/R/library/RcppEigen/include/Eigen/Dense:1,
##                  from /home/mikesh/.lib/R/library/StanHeaders/include/stan/math/prim/mat/fun/Eigen.hp
##                  from <command-line>:0:
## /home/mikesh/.lib/R/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:1: error: unknown type
##  namespace Eigen {
##  ^~~~~~~~~
## /home/mikesh/.lib/R/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:17: error: expected ‘=
##  namespace Eigen {
##                  ^
## In file included from /home/mikesh/.lib/R/library/RcppEigen/include/Eigen/Dense:1:0,
##                  from /home/mikesh/.lib/R/library/StanHeaders/include/stan/math/prim/mat/fun/Eigen.hp
##                  from <command-line>:0:
## /home/mikesh/.lib/R/library/RcppEigen/include/Eigen/Core:96:10: fatal error: complex: No such file or
##  #include <complex>
##           ^~~~~~~~~
## compilation terminated.
## /usr/lib/R/etc/Makeconf:172: recipe for target 'foo.o' failed
## make: *** [foo.o] Error 1
```

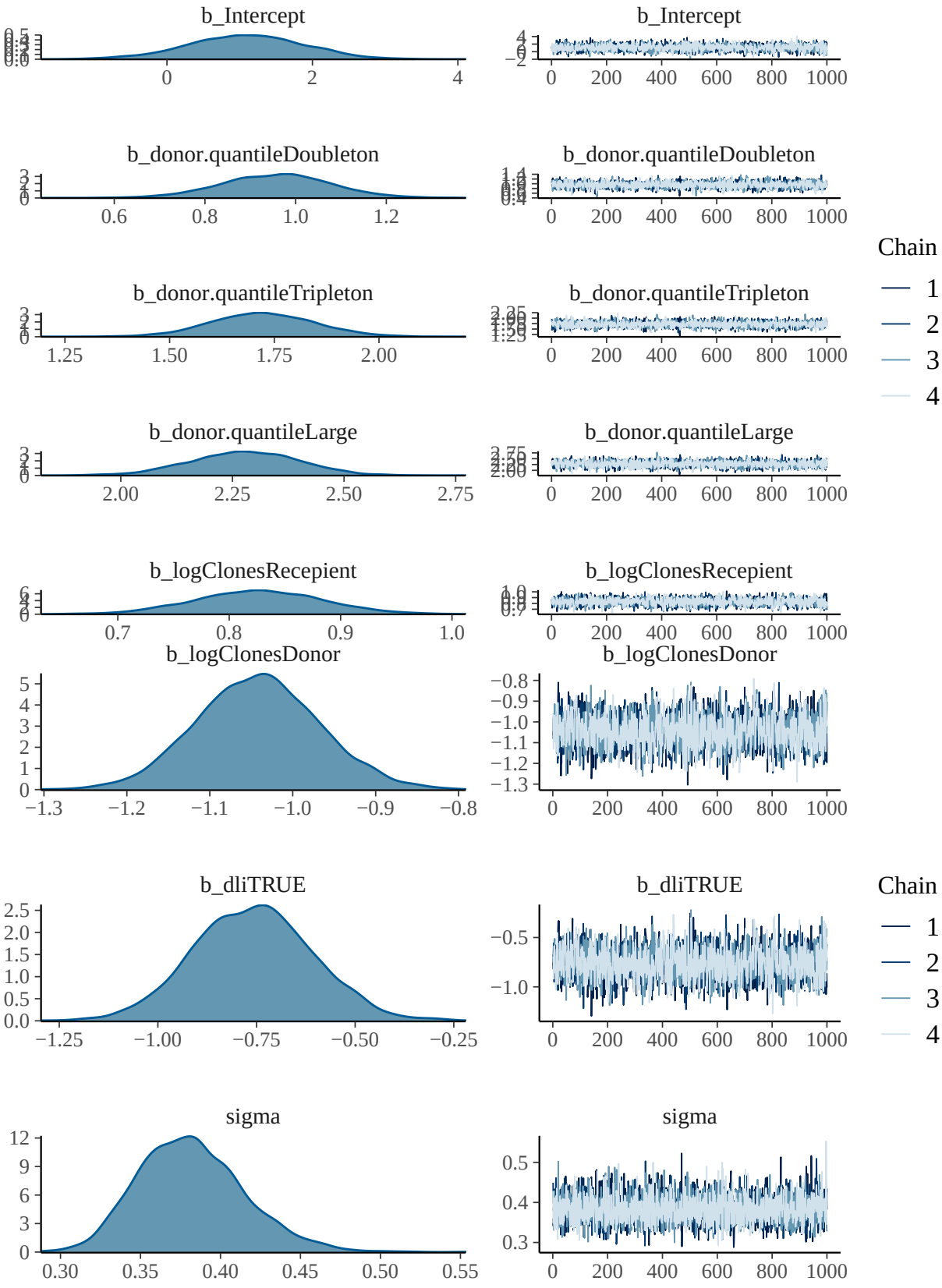
```
## Start sampling
```

```
##
## SAMPLING FOR MODEL '1ca9dda60febd82977fecbb6558b1905' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 5.6e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.56 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
```

```
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.055526 seconds (Warm-up)
## Chain 1:                0.038977 seconds (Sampling)
## Chain 1:                0.094503 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '1ca9dda60febd82977fecbb6558b1905' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.1 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.044973 seconds (Warm-up)
## Chain 2:                0.03393 seconds (Sampling)
## Chain 2:                0.078903 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '1ca9dda60febd82977fecbb6558b1905' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 8e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
```

```
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.034121 seconds (Warm-up)
## Chain 3:                0.029541 seconds (Sampling)
## Chain 3:                0.063662 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '1ca9dda60febd82977fecbb6558b1905' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 8e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.033761 seconds (Warm-up)
## Chain 4:                0.030014 seconds (Sampling)
## Chain 4:                0.063775 seconds (Total)
## Chain 4:
```

```
plot(mdl, ask = F)
```


b_Intercept
b_Intercept
b_donor.quantileDoubleton
b_donor.quantileDoubleton
b_donor.quantileTripleton
b_donor.quantileTripleton
b_donor.quantileLarge
b_donor.quantileLarge
b_logClonesRecepient
b_logClonesRecepient
b_logClonesDonor
b_logClonesDonor
b_dliTRUE
b_dliTRUE
sigma
sigma
Chain
1
2
3
4
Chain
1
2
3
4