RNA-seq data cleaning

voom to DEG

Kim Dill-McFarland, kadm@uw.edu

version April 28, 2022

Contents

Overview

This document covers the Hawn lab recommended analysis pipeline to determine differentially expressed genes (DEG). This pipeline includes simple linear modeling and linear mixed effects modeling with main effects, interaction terms, co-variates, and random effects. There is discussion of how to choose a 'best fit' model using fit, significant genes, and biological knowledge. The example data are human, bulk, paired-end RNA-seq, but this pipeline can be applied to other organisms or single-read libraries.

0. Setup

Software

This pipeline should be completed in R and RStudio. You should also install the following packages.

```
#CRAN packages
install.packages(c("tidyverse", "DiagrammeR"))

#Bioconductor packages
install.packages("BiocManager")
BiocManager::install(c("edgeR", "limma", "patchwork"))

#GitHub packages
install.packages("devtools")
devtools::install_github("BIGslu/BIGpicture", force=TRUE)
devtools::install_github("BIGslu/kimma", force=TRUE)
```

And load them into your current R session.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(kimma)
library(BIGpicture)
library(limma)
library(edgeR)
library(patchwork)

set.seed(651)
```

Directory structure

To directly use the code in this pipeline, you must organize your files as follows.

All of the data needed are contained in a single `Rdata` object in `data_clean/`.

Example data

Example data were obtained from virus-stimulated human plasmacytoid dendritic cells. Actual patient identifiers and metadata have been altered for this tutorial.

> Dill-McFarland KA, Schwartz JT, Zhao H, Shao B, Fulkerson PC, Altman MC, Gill MA. 2022. Eosinophil-mediated suppression and Anti-IL-5 enhancement of plasmacytoid dendritic cell interferon responses in asthma. J Allergy Clin Immunol. Epub ahead of print. doi: 10.1016/j.jaci.2022.03.025. — GitHub

Specifically, this tutorial uses data processed using our SEAsnake and counts to voom pipelines, resulting in voom-normalized, log2 counts per million (CPM) expression and associated sample metadata in a `limma EList` object in the `data_clean/` directory.

1. Load data

All counts, gene, and sample metadata are contained in a single object.

```
#Attach data
attach("data_clean/P259_voom.RData")
#Extract and rename data object
dat <- dat.abund.norm.voom
```

We access each data frame within this `Elist` using `$`. The normalized log2 CPM expression data are contained in `E`.

```
dat$E[1:3,1:7]
```

```
##                     lib1    lib10     lib2     lib3     lib4     lib5     lib6
## ENSG00000186827 7.558128 5.928606 6.514101 7.662422 6.379447 6.707211 6.126591
## ENSG00000186891 7.426584 4.491743 5.508152 6.793137 3.831638 5.374548 4.042058
## ENSG00000160072 5.373989 5.620690 4.611656 4.500602 4.982418 5.675978 5.487858
```

Library and donor metadata are in `targets`.

```
dat$targets[1:3,1:10]
```

```
##       group lib.size norm.factors libID   ptID virus batch  asthma age sex
## lib1      1  1217753    1.1158366  lib1 donor1  none     A healthy  35   F
## lib10     1  2733542    1.0596930 lib10 donor5   HRV     B  asthma  26   M
## lib2      1  1329340    0.6589573  lib2 donor1   HRV     A healthy  35   F
```

Gene metadata are in `genes`.

```
dat$genes[1:3,1:4]
```

```
##                 ensembl_gene_id entrezgene_id   gene_biotype chromosome_name
## ENSG00000284662 ENSG00000000003          7105 protein_coding               X
## ENSG00000186891 ENSG00000000419          8813 protein_coding              20
## ENSG00000160072 ENSG00000000457         57147 protein_coding               1
```

There are a couple other data frames in our `dat` object, but they are not relevant to this tutorial.

Additionally, models take a couple minutes to run. You can load all model results for this tutorial.

```
load("results/model_results.RData")
```

2. Simple linear model

2.1: A single main effect (virus)

First, we consider our research question and what variable(s) we need to answer that question. In these data, the first variable of interest is `virus` to determine how viral infection impacts gene expression. In R, this model is written as:

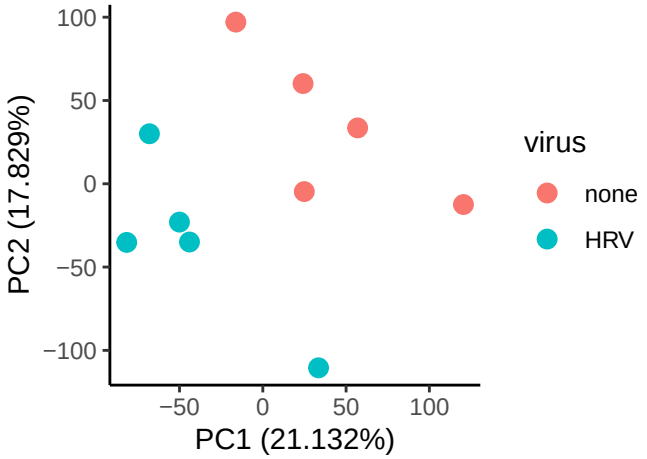
```
~ virus
```

On a coarse scale such as PCA, we can see that `virus` impacts gene expression with uninfected controls (none) grouping together away from HRV-infected samples.

```
plot_pca(dat, vars = "virus")
```

```
## Joining, by = "libID"
```

```
## $virus
```



We run this linear model in `kimma` using `kmFit`.

```
virus <- kmFit(dat = dat, model = "~ virus", run.lm = TRUE)
```

```
lm model: expression~virus
Input: 10 libraries from 5 unique patients
Model: 10 libraries
Complete: 13301 genes
Failed: 0 genes
```

We see that numerous genes are significant for `virus`. However, this is not the best model for these data since we have other factors that may impact expression.

```
summarise_kmFit(fdr = virus$lm)
```

```
## # A tibble: 2 x 7
##   variable             fdr_0.05 fdr_0.1 fdr_0.2 fdr_0.3 fdr_0.4 fdr_0.5
##   <fct>                   <int>   <int>   <int>   <int>   <int>   <int>
## 1 virusHRV                  212     492    1230    1856    2778    3692
## 2 total (nonredundant)      212     492    1230    1856    2778    3692
```

2.2: Multiple main effects (virus & asthma)

We are actually interested in how individuals with and without asthma respond to virus. We can add variables to our model with `+` such as

```
~ virus + asthma
```

However, this model only captures the main effects of each variable in isolation. Specifically, this model tells you how virus impacts genes expression and how asthma impacts gene expression. It does not address how viral impacts *differ* between those with and without asthma.

Interaction terms

One way to assess this is with an interaction term written as:

```
~ virus + asthma + virus:asthma
```

or short-handed with `*`. Note, these two models are equivalent in R.

```
~ virus * asthma
```

This model now tests both the main effects and their interaction.

```
virus_interact <- kmFit(dat = dat, model = "~ virus*asthma", run.lm = TRUE)
```

```
lm model: expression~virus*asthma
Input: 10 libraries from 5 unique patients
Model: 10 libraries
Complete: 13301 genes
Failed: 0 genes
```

We now see 3 variables in our results equivalent to the variables in the long form of our model equation. Notice that we've lost significance for a lot of genes in `virus` and not found any genes at FDR < 0.05 for asthma or the interaction term. Because this data set is small, an interaction model is likely too complex, and we do not appear to have the power to detect interaction effects.

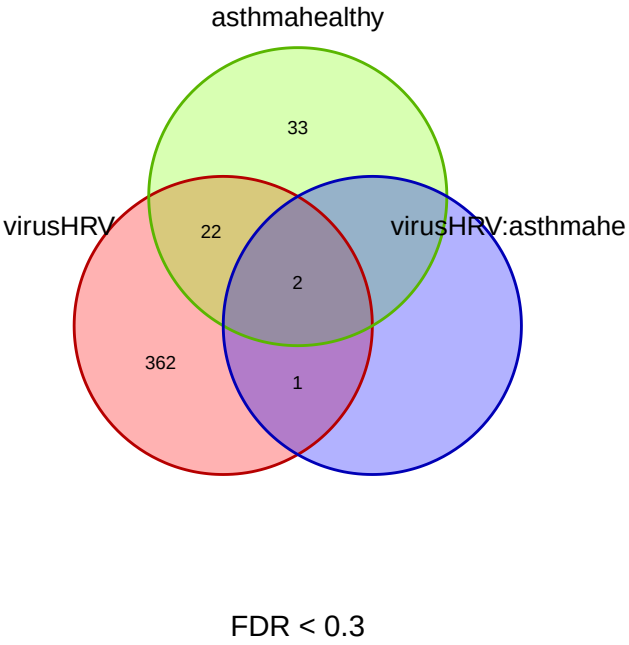
```
summarise_kmFit(fdr = virus_interact$lm)
```

```
## # A tibble: 4 x 7
##   variable               fdr_0.05 fdr_0.1 fdr_0.2 fdr_0.3 fdr_0.4 fdr_0.5
##   <fct>                     <int>   <int>   <int>   <int>   <int>   <int>
```

```
## 1 asthmahealthy                NA      NA       1      57     169     459
## 2 virusHRV                      3       3     162     387     698    1076
## 3 virusHRV:asthmahealthy       NA      NA      NA       3       6       6
## 4 total (nonredundant)          3       3     162     420     786    1342
```

Importantly, a gene with a significant interaction term cannot be assessed for the main effects. For example at a high FDR of 0.3, there are 3 genes here that is significant for the interaction (blue) and the virus (red) and/or asthma (green) main terms. However, we *cannot* use the asthma results for these genes, because they are comparing all healthy to all asthma samples without taking virus into account. Since we know there is an interaction effect for these genes, the asthma comparison alone is incorrectly averaging across samples we know to be different (none vs HRV). Similarly, we cannot use the virus results.

```
plot_venn_genes(model_result = virus_interact, model = "lm", fdr.cutoff = 0.3)
```



If this were our final model, our DEG list would be all interaction genes (blue) as well as the intersect of virus and asthma main terms (red-green). This second group encompasses genes that change with virus similarly in healthy and asthma donors but are always higher in one asthma group.

Pairwise contrasts

Another way to model interactions is with pairwise contrasts. Contrasts compare 2 or more groups to all other groups in that variable. For example, we're interested in the 4 groups within the interaction term: `none_healthy`, `none_asthma`, `HRV_healthy`, `HRV_asthma`. We run these comparisons with the same interaction model as above, only now we also set `run.contrast = TRUE`. We will get pairwise comparisons that we're not interested in since all combinations are run but will filter those of interest later on.

```
virus_contrast <- kmFit(dat = dat, model = "~ virus*asthma",
                        run.lm = TRUE,
                        run.contrast = TRUE, contrast.var = "virus:asthma")
```

```
lm model: expression~virus*asthma
Input: 10 libraries from 5 unique patients
Model: 10 libraries
Complete: 13301 genes
Failed: 0 genes
```

We see the same main model outcome as before.

```
summarise_kmFit(fdr = virus_contrast$lm)
```

```
## # A tibble: 4 x 7
##   variable               fdr_0.05 fdr_0.1 fdr_0.2 fdr_0.3 fdr_0.4 fdr_0.5
##   <fct>                     <int>   <int>   <int>   <int>   <int>   <int>
## 1 asthmahealthy                NA      NA       1      57     169     459
## 2 virusHRV                      3       3     162     387     698    1076
## 3 virusHRV:asthmahealthy       NA      NA      NA       3       6       6
## 4 total (nonredundant)          3       3     162     420     786    1342
```

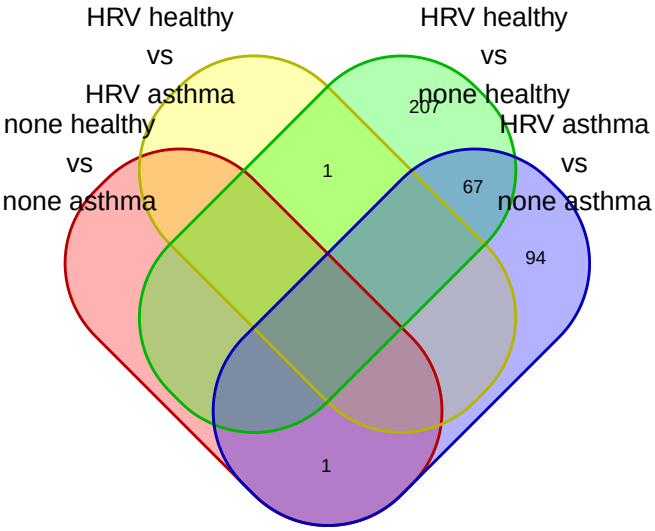
We also get all pairwise contrasts between the 4 groups.

```
summarise_kmFit(fdr = virus_contrast$lm.contrast)
```

```
## # A tibble: 7 x 9
##   variable    contrast_ref contrast_lvl fdr_0.05 fdr_0.1 fdr_0.2 fdr_0.3 fdr_0.4
##   <fct>       <chr>        <chr>           <int>   <int>   <int>   <int>   <int>
## 1 virus*asth~ HRV asthma   HRV healthy         1       1       1       4      22
## 2 virus*asth~ HRV asthma   none healthy        3       4     126     497    1106
## 3 virus*asth~ none asthma  HRV asthma          3       3     162     387     698
## 4 virus*asth~ none asthma  HRV healthy        84     391    1224    2180    3374
## 5 virus*asth~ none healthy HRV healthy        NA      54     275     787    1246
## 6 virus*asth~ none asthma  none healthy       NA      NA       1      57     169
## 7 total (non~ <NA>         <NA>              85     402    1335    2554    4028
## # ... with 1 more variable: fdr_0.5 <int>
```

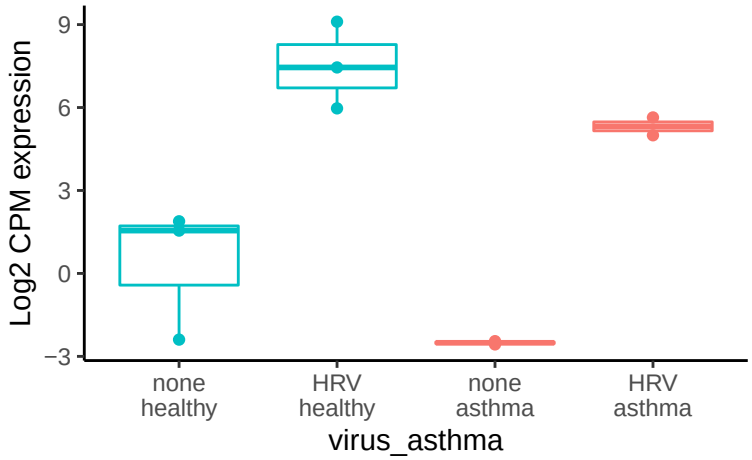
Not all of these contrasts are of interest, so we select just the effects of virus (red, yellow) and asthma (green, blue). We see that many genes change with virus in the asthma (blue) and/or healthy (green) groups. Only 2 genes differ between healthy and asthma either with (yellow) or without virus (red). The lack of significant genes in the left ovals is an artifact of this small data set with only 2 - 3 donors per group. In a real analysis, you may be interested in genes that only change with virus in one group (green and blue not overlapping) or those that change with virus (green and/or blue) AND are different with asthma (red and/or yellow).

```
plot_venn_genes(model_result = virus_contrast, model = "lm.contrast",
                fdr.cutoff = 0.2,
                contrasts = data.frame(
                  contrast_ref = c("none asthma","HRV asthma",
                                   "none healthy","none asthma"),
                  contrast_lvl = c("none healthy","HRV healthy",
                                   "HRV healthy","HRV asthma")))
```

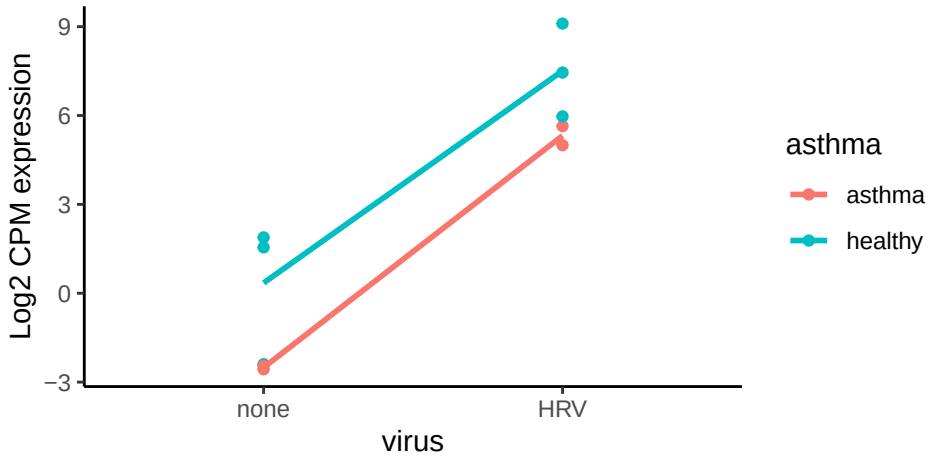


When to use interaction vs contrasts

At their heart, interaction and contrast models are trying to answer the same question. However, statistically, they are very different. Contrasts compare means between groups (see below) and you must select which pairwise comparisons are meaningful.



An interaction term tests if two slopes differ. In these data, this is comparing the change (slope) in response to virus in healthy vs asthmatic donors like below In most cases, it is more difficult to achieve significance when comparing slopes as opposed to means.



In general, we recommend using the interaction term to define differentially expressed genes (DEG). Then, as a post-hoc test, run contrasts only on significant DEGs to further probe the results. This is demonstrated later in this tutorial. It is like running an ANOVA to compare groups A,B,C and then TukeyHSD to determine which groups differ from which (A vs B, B vs C, A vs C).

Contrasts may be employed as your main model instead in cases such as:

- Small data sets where you are under-powered and would benefit from the reduced complexity of contrasts (1 variable) vs an interaction (3 variables)
- When there is no significance for the interaction term
- When you are only interested in a subset of the pairwise comparisons encompassed by the interaction term. For example, if you wanted to test a longitudinal study as time point 1 vs 2, 2 vs 3, 3 vs 4, etc

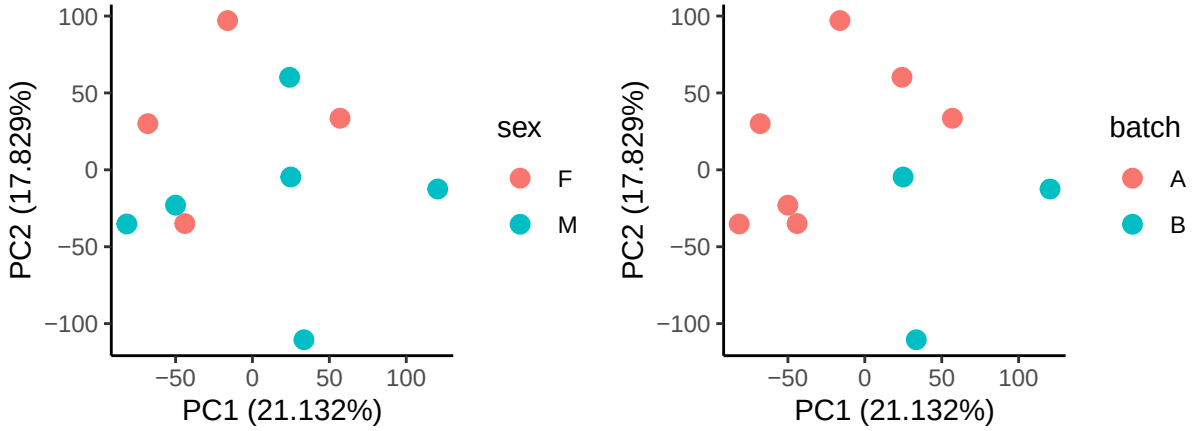
2.3: Co-variates (batch & sex)

We also want to consider the effects of variables that may impact or prevent us from seeing our main effects. In human studies, this often includes age and sex, though we only have sex for these data. We also want to consider batch since these data were sequenced in two batches.

To determine which co-variates to include, let's see if they have a large impact on the data in PCA.

```
plot_pca(dat, vars = c("sex","batch")) %>%
  wrap_plots(ncol=2)
```

```
## Joining, by = "libID"
```



We see grouping by batch; thus, we have evidence to include batch in the model despite the fact that these data were batch-corrected using `ComBat-Seq`. This is not uncommon with batch correction as we err on the

side of under-correcting by batch to retain true biological variation. Plus, we did a very poor-correction based on only 2 genes in the previous tutorial in order to keep computation time down.

In contrast, sex does not show any groupings and may not be needed. PCA alone, however, is not enough to determine what co-variates need to be in our models.

Next, we run the models with and without co-variates for comparison. To include co-variates, we add them to the models with `+`. We also set `metrics = TRUE` so that `kimma` gives us model fit metrics for comparison.

```
virus_interact <- kmFit(dat = dat, model = "~ virus*asthma",
                        run.lm = TRUE, metrics = TRUE)
```

```
lm model: expression~virus*asthma
Input: 10 libraries from 5 unique patients
Model: 10 libraries
Complete: 13301 genes
Failed: 0 genes
```

```
virus_batch <- kmFit(dat = dat, model = "~ virus*asthma + batch",
                     run.lm = TRUE, metrics = TRUE)
```

```
lm model: expression~virus*asthma+batch
Input: 10 libraries from 5 unique patients
Model: 10 libraries
Complete: 13301 genes
Failed: 0 genes
```

```
virus_batch_sex <- kmFit(dat = dat, model = "~ virus*asthma + batch + sex",
                         run.lm = TRUE, metrics = TRUE)
```

```
lm model: expression~virus*asthma+batch+sex
Input: 10 libraries from 5 unique patients
Model: 10 libraries
Complete: 13301 genes
Failed: 0 genes
```

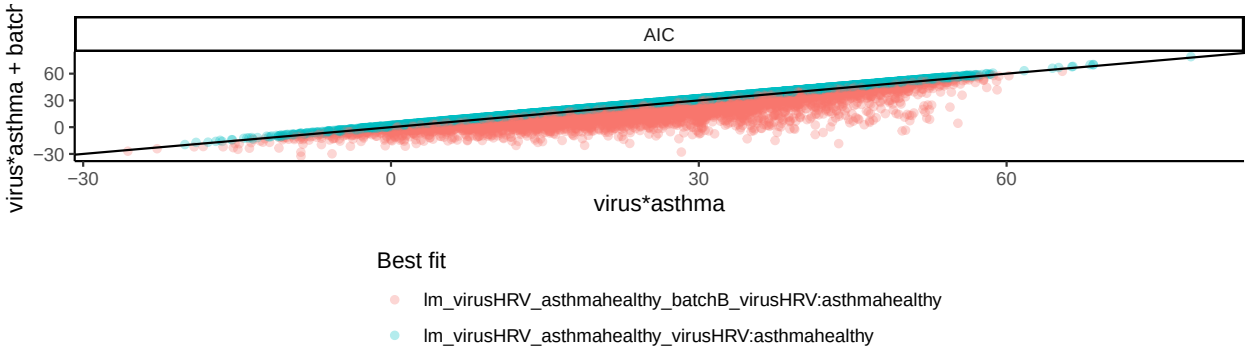
We compare model fits with AIC, which summarize model fit. For more information, see https://en.wikipedia.org/wiki/Akaike_information_criterion. In general, a difference in AIC < 2 is considered no evidence of better fit, 2 - 8 is moderate evidence, and > 8 is strong evidence. There are additional metrics also available in `kimma` including BIC, sigma, and R-squared.

Smaller AIC indicate a better fitting model so genes below the 1:1 line are better fit by the y-axis model (red). Those above the line are better fit by the x-axis model (teal). Throughout this tutorial, we put the more complex model as y. Our plotting function also outputs a message with how many genes are best fit by each model and mean and standard deviation of the difference in AIC between the two models.

```
plot_fit(model_result = virus_interact, model_result_y = virus_batch,
         x="lm", y="lm", metrics = "AIC") +
  labs(x = "virus*asthma", y = "virus*asthma + batch")
```

```
## Summary
```

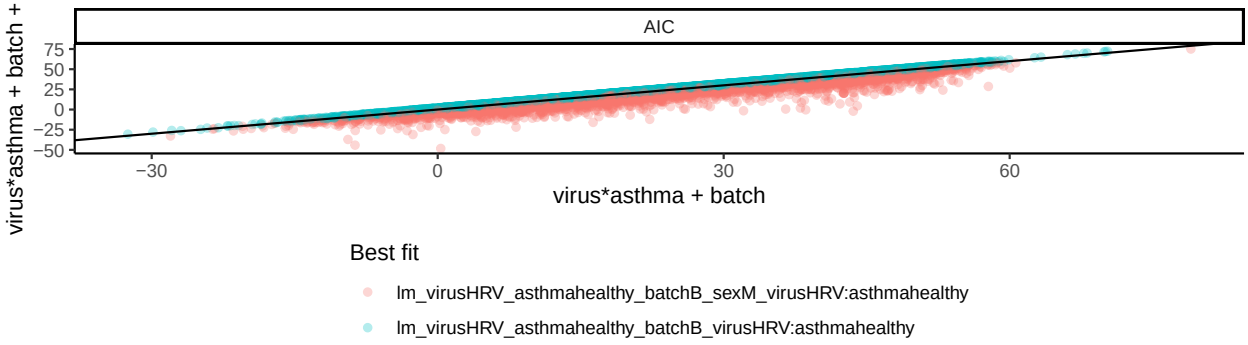
```
##                                                 Best fit Metric Total genes
## 1 lm_virusHRV_asthmahealthy_batchB_virusHRV:asthmahealthy    AIC        6737
## 2        lm_virusHRV_asthmahealthy_virusHRV:asthmahealthy    AIC        6564
##   Mean delta Stdev delta
## 1   6.471052    7.108772
## 2   1.371054    0.584548
```




```
plot_fit(model_result = virus_batch, model_result_y = virus_batch_sex,
         x="lm", y="lm", metrics = "AIC") +
  labs(x = "virus*asthma + batch", y = "virus*asthma + batch + sex")
```

```
## Summary
```

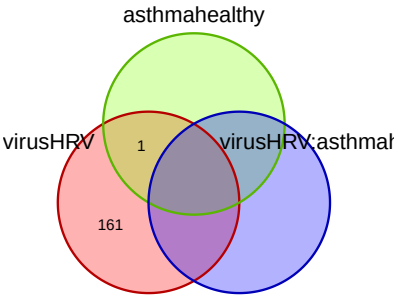
```
##                                                        Best fit Metric
## 1 lm_virusHRV_asthmahealthy_batchB_sexM_virusHRV:asthmahealthy    AIC
## 2      lm_virusHRV_asthmahealthy_batchB_virusHRV:asthmahealthy    AIC
##   Total genes Mean delta Stdev delta
## 1        5448   4.149363   4.7061414
## 2        7853   1.387260   0.5821378
```



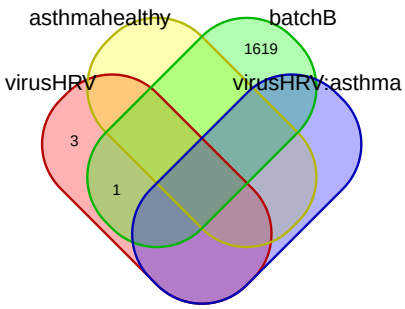
We see that each model is the best fit for a subset of genes. In the first plot, batch improves the model fit for just over half of genes, and this improvement is substantial for some genes (*e.g.* red dots far from the 1:1 line and mean Δ AIC = 6.5). In the second plot, sex further improves the fit for roughly 40% of genes but to a lesser extent (mean Δ AIC = 4.7).

Next, we compare how many genes are significant with and without co-variates. We arbitrarily select a high FDR cutoff of 0.2 though you may wish to look at a couple.

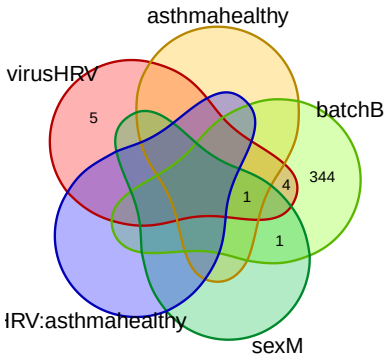
```
plot_venn_genes(model_result = virus_interact, model = "lm",
                fdr.cutoff = 0.2) +
  plot_venn_genes(model_result = virus_batch, model = "lm",
                  fdr.cutoff = 0.2) +
  plot_venn_genes(model_result = virus_batch_sex, model = "lm",
                  fdr.cutoff = 0.2)
```



FDR < 0.2



FDR < 0.2



FDR < 0.2

First, consider how many genes are significant for the co-variates themselves. Batch is significant for many genes while sex is significant for only 2. Next, compare the number of virus-significant genes. Adding batch and/or sex to the model decreases the number of virus (red) by a lot (162 vs 4 or 10). The dramatic decrease in virus genes is likely evidence that we do not have enough power to support the co-variate model complexity (*e.g.* too few samples relative to the number of variables). Given the size of this data set (N = 10), this is not surprising.

To summarize:

Batch

- Strongly impacts overall gene expression in PCA
- Improves model fit for a small majority of genes
- Significant for many genes
- Decreases the number of virus-significant genes

Sex

- Does not impact overall gene expression in PCA
- Improves model fit for a minority of genes
- Significant for a few genes
- Decreases the number of virus-significant genes

If it weren't for the impacts on virus genes, we would have clear evidence to include batch and exclude sex from the model. However, given the size of this data set and evidence that even 1 co-variate cannot be supported, we will not include any at this time.

```
~ virus*asthma
```

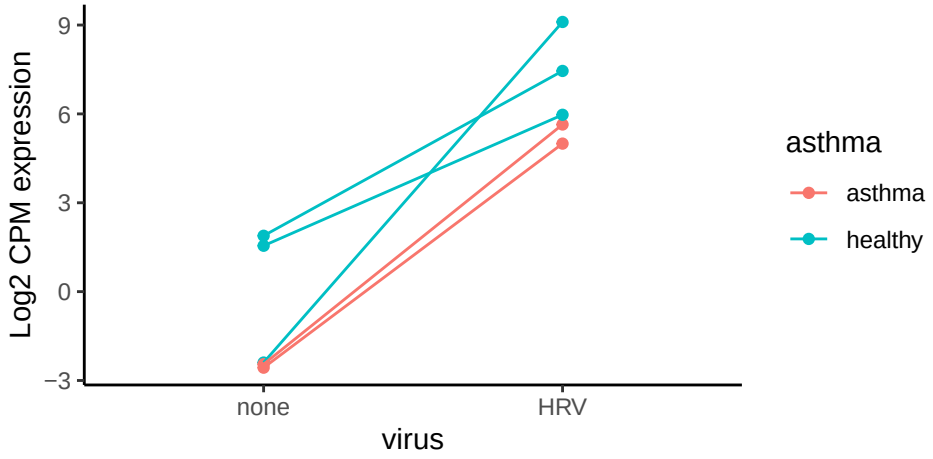
In your data, you may have co-variates that are not as clear as these. In that case, it's important to prioritize model fit and sample size over significant gene counts. You should not include or exclude a co-variate just because it gets you more significant genes for your main terms. This is especially true if the co-variate negatively impacts model fit.

You may also have non-statistical evidence for including a co-variate. If you have established biological evidence that a co-variate is important in your system or it's standard in your field, you may wish to include it even if it does not improve fit and is not significant. If the co-variate, however, greatly worsens fit, you should not include it even if it's standard. Instead, present fit plots like above to support its exclusion.

3. Linear mixed effects model

3.1: Random effects (ptID)

These data come from a paired study design with uninfected (none) and infected (HRV) samples from the same donor's cells. We take this into account in our model by using donor as a random effect. This allows the model to match samples from the same donor. It greatly improves our statistical power as we now have 1 slope per donor instead of 1 mean slope across all donors. So, we can see if all individual donors change similarly or not.



Random effects are added to the model with `+ (1|block)` where the block is the variable you want to pair samples by. Thus, we now run a mixed effects model `lme` with

`~ virus*asthma + (1|ptID)`

In `kimma`, this is run similarly to a simple model except we add our random term and ask it to `run.lme`.

```
virus_lme <- kmFit(dat = dat, model = "~ virus*asthma + (1|ptID)",
                   run.lme = TRUE, metrics = TRUE)
```

```
lme/lmerel model: expression~virus*asthma+(1|ptID)
Input: 10 libraries from 5 unique patients
Model: 10 libraries
Complete: 13301 genes
Failed: 0 genes
```

Note, we could run both models at once in `kmFit` if we set `run.lm=TRUE` and `run.lme=TRUE`. This is helpful when you know you want to compare multiple models from the start. Since we already ran the simple linear model, we'll skip that here.

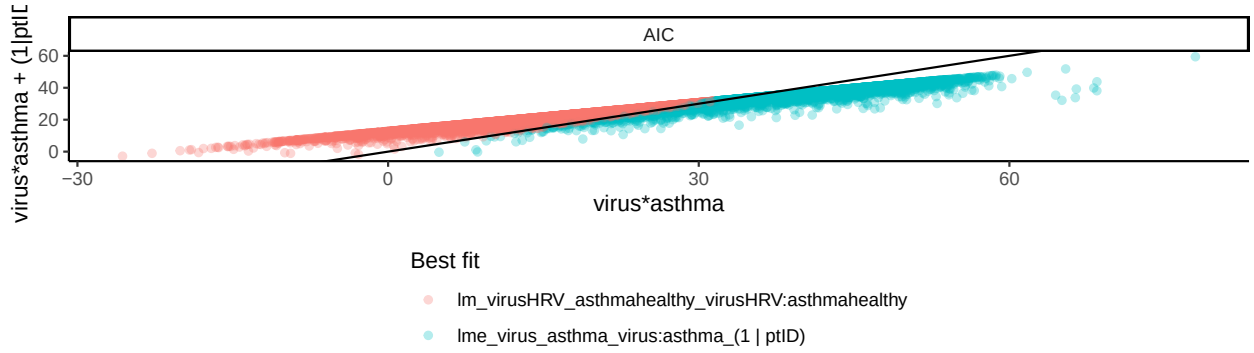
Comparing models as we did with co-variates, we see that genes are split by best fit to a similar degree (mean Δ AIC ~5) for both models. However, we see many more significant genes in the model with the random effect. This illustrates the much improved power of a mixed effects model when you have paired samples.

```
plot_fit(model_result = virus_interact, model_result_y = virus_lme,
         x="lm", y="lme", metrics = "AIC") +
  labs(x = "virus*asthma", y = "virus*asthma + (1|ptID)")
```

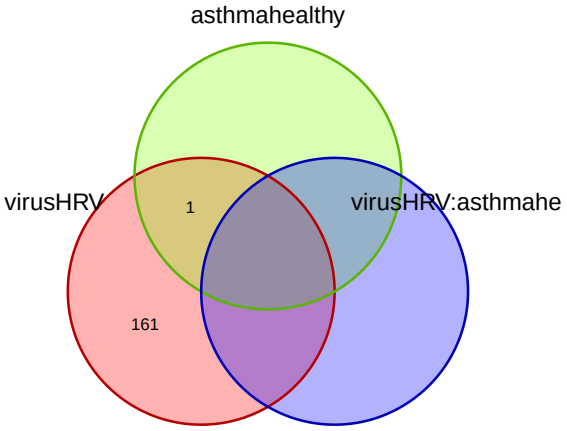
```
## Summary
```

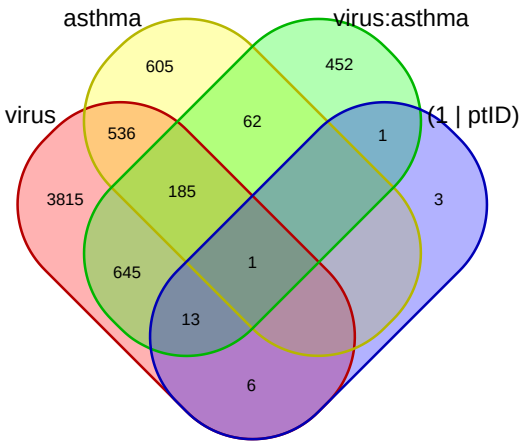
```
##                                          Best fit Metric Total genes
## 1 lm_virusHRV_asthmahealthy_virusHRV:asthmahealthy    AIC        7671
## 2         lme_virus_asthma_virus:asthma_(1 | ptID)    AIC        5630
##   Mean delta Stdev delta
```

```
## 1    5.678204     3.717770
## 2    4.703658     3.251966
```




```
plot_venn_genes(model_result = virus_interact, model = "lm",
                fdr.cutoff = 0.2) +
  plot_venn_genes(model_result = virus_lme, model = "lme",
                  fdr.cutoff = 0.2)
```





FDR < 0.2

FDR < 0.2

Comparison to `limma`

`limma` offers a sudo random effect in linear modeling with `duplicateCorrelation`. This estimates the random effect of donor across all genes and uses one value for all models. The full mixed effect model in `kimma` estimates the random effect for each gene, thus fitting a different value for each gene's model.

In our analyses, we've found `limma` and `kimma` results to be similar when the sample size or variable effects are large. In the case of small data sets or small effects, however, there can be a dramatic difference.

For example, we run a `limma` model with pseduo paired design for these data.

```
mm_limma <- model.matrix(~ virus*asthma, data=dat$targets)

#Block by donor
consensus.corr <- duplicateCorrelation(dat$E, mm_limma,
                                       block=dat$targets$ptID)$consensus.correlation
consensus.corr
```

```
## [1] 0.3074545
```

```
# Fit model to transformed count data. Calculate eBayes
fit_limma <- eBayes(lmFit(dat$E, mm_limma,
                          block=dat$targets$ptID,
                          correlation=consensus.corr))

#Extract pval
virus_limma <- extract_lmFit(design = mm_limma, fit = fit_limma)
```

We see that not taking the paired sample design into account results in few DEG (first table). Using duplicate correlation in `limma` gains some (second table) but `kimma`'s full mixed effects model best detects signal across all variables (third table).

```
# ~ virus*asthma in kimma
summarise_kmFit(fdr = virus_interact$lm)
```

```
## # A tibble: 4 x 7
##   variable               fdr_0.05 fdr_0.1 fdr_0.2 fdr_0.3 fdr_0.4 fdr_0.5
##   <fct>                     <int>   <int>   <int>   <int>   <int>   <int>
## 1 asthmahealthy                NA      NA       1      57     169     459
## 2 virusHRV                      3       3     162     387     698    1076
## 3 virusHRV:asthmahealthy       NA      NA      NA       3       6       6
## 4 total (nonredundant)          3       3     162     420     786    1342
```

```
# ~ virus*asthma + duplicate correlation in limma
summarise_lmFit(fdr = virus_limma)
```

```
## # A tibble: 4 x 7
##   variable               fdr_0.05 fdr_0.1 fdr_0.2 fdr_0.3 fdr_0.4 fdr_0.5
##   <fct>                     <int>   <int>   <int>   <int>   <int>   <int>
## 1 asthmahealthy                 0       8      25      60     127     256
## 2 virusHRV                    135     361     797    1344    1988    2752
## 3 virusHRV:asthmahealthy        5       7      22      58     105     194
## 4 total (nonredundant)        135     364     808    1369    2043    2869
```

```
# ~ virus*asthma + (1|ptID) in kimma
summarise_kmFit(fdr = virus_lme$lme)
```

```
## # A tibble: 5 x 7
##   variable             fdr_0.05 fdr_0.1 fdr_0.2 fdr_0.3 fdr_0.4 fdr_0.5
##   <fct>                   <int>   <int>   <int>   <int>   <int>   <int>
## 1 (1 | ptID)                 NA       3      24      39      70     114
## 2 asthma                    676     923    1389    1843    2295    2943
## 3 virus                    3512    4173    5201    6068    6934    7857
## 4 virus:asthma              839    1043    1359    1753    2118    2635
## 5 total (nonredundant)     4210    5027    6324    7405    8425    9521
```

3.2 Co-variates (batch & sex)

Given the large changes with a mixed effect model, let's re-consider batch and sex co-variates.

```
virus_batch_lme <- kmFit(dat = dat, model = "~ virus*asthma + batch + (1|ptID)",
                         run.lme = TRUE, metrics = TRUE)
```

```
lme/lmerel model: expression~virus*asthma+batch+(1|ptID)
Input: 10 libraries from 5 unique patients
Model: 10 libraries
Complete: 13301 genes
```

```
Failed: 0 genes
```

```
virus_batch_sex_lme <- kmFit(dat = dat,
                             model = "~ virus*asthma + batch + sex + (1|ptID)",
                             run.lme = TRUE, metrics = TRUE)
```

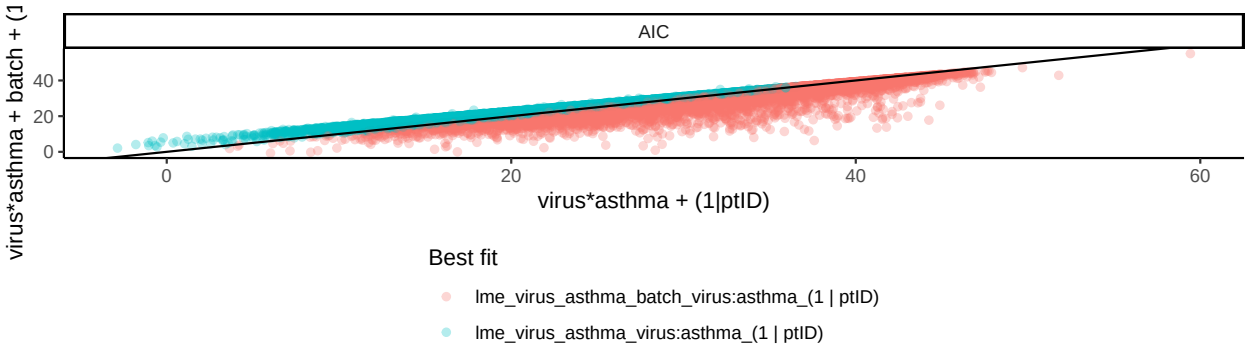
```
lme/lmerel model: expression~virus*asthma+batch+sex+(1|ptID)
Input: 10 libraries from 5 unique patients
Model: 10 libraries
Complete: 13301 genes
Failed: 0 genes
```

We see no clear evidence to keep co-variates based on model fit since genes are split for best fit model and mean Δ AIC is low from 1 to 3.

```
plot_fit(model_result = virus_lme, model_result_y = virus_batch_lme,
         x="lme", y="lme", metrics="AIC") +
  labs(x = "virus*asthma + (1|ptID)", y = "virus*asthma + batch + (1|ptID)")
```

```
## Summary
```

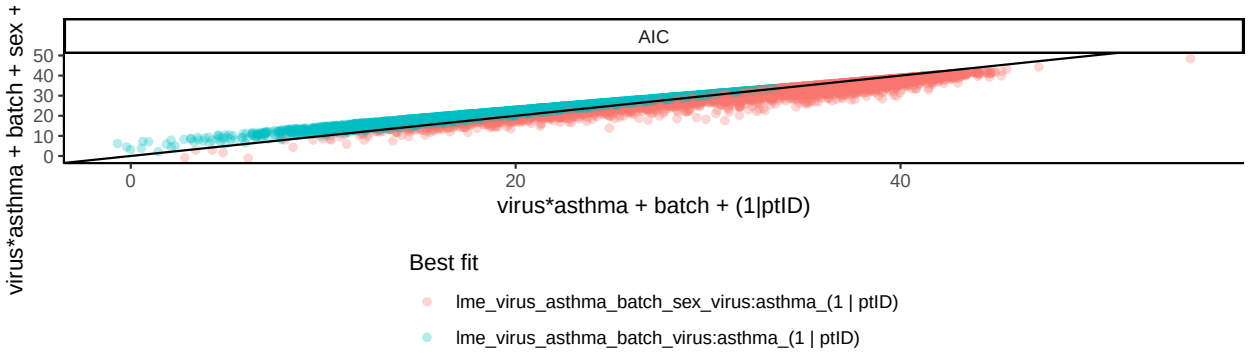
```
##                                       Best fit Metric Total genes Mean delta
## 1 lme_virus_asthma_batch_virus:asthma_(1 | ptID)    AIC        7241   2.903870
## 2       lme_virus_asthma_virus:asthma_(1 | ptID)    AIC        6060   1.739159
##   Stdev delta
## 1    3.456428
## 2    1.148527
```




```
plot_fit(model_result = virus_batch_lme, model_result_y = virus_batch_sex_lme,
         x="lme", y="lme", metrics="AIC") +
  labs(x = "virus*asthma + batch + (1|ptID)",
       y = "virus*asthma + batch + sex + (1|ptID)")
```

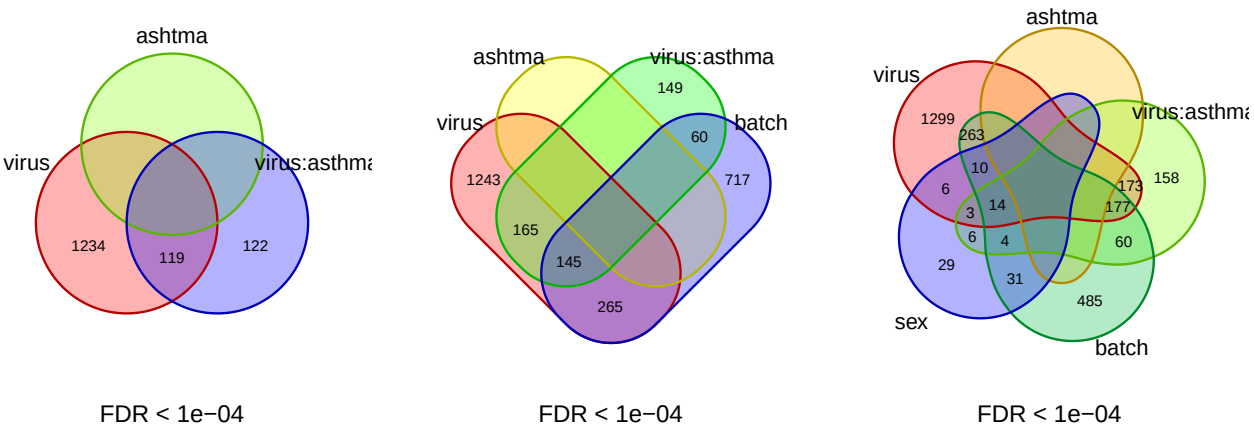
```
## Summary
```

```
##                                           Best fit Metric Total genes
## 1 lme_virus_asthma_batch_sex_virus:asthma_(1 | ptID)    AIC        5494
## 2     lme_virus_asthma_batch_virus:asthma_(1 | ptID)    AIC        7807
##   Mean delta Stdev delta
## 1   1.720881    1.491965
## 2   1.663087    1.040741
```



Both co-variates increase the number of virus-significant genes and are themselves significant for a number of genes. Here, we use a much lower FDR cutoff of 0.0001 so the values are easier to see.

```
plot_venn_genes(model_result = virus_lme, model = "lme",
                fdr.cutoff = 0.0001,
                variables = c("virus","ashtma","virus:asthma")) +
  plot_venn_genes(model_result = virus_batch_lme, model = "lme",
                  fdr.cutoff = 0.0001,
                  variables = c("virus","ashtma","virus:asthma", "batch")) +
  plot_venn_genes(model_result = virus_batch_sex_lme, model = "lme",
                  fdr.cutoff = 0.0001,
                  variables = c("virus","ashtma","virus:asthma","batch","sex"))
```



FDR < 1e−04 FDR < 1e−04 FDR < 1e−04

Importantly, some of the interaction genes (our main variable of interest) are significant for co-variates. This is further evidence to keep co-variates since they appear to play a role in the genes we'll focus on in our results. The being said, the co-variates do not improve model fit and you could argue to remove them. So, this is a case where there is no clear right answer. As long as you understand the potential impacts on your results, you could continue your analyses with any of these 3 mixed effects models.

For simplicity, we'll move forward without any co-variates.

```
~ virus*asthma + (1|ptID)
```

3.3: Kinship co-variate

Kinship is a summative measure of genetic relatedness. It can be from 0 to 1 with 1 being 100% identical (monozygotic twins). Some other common values are 0.5 for parent-child, 0.25 grandparent-grandchild, 0.125 first cousins, etc. This measure is a pairwise measure with 1 value per pair of individuals.

```
kin <- kimma::example.kin
```

```
kin
```

```
##        donor1 donor2 donor3 donor4 donor5 donor6
## donor1   1.00   0.50   0.10   0.20   0.15   0.10
## donor2   0.50   1.00   0.10   0.11   0.23   0.09
## donor3   0.10   0.10   1.00   0.10   0.20   0.15
## donor4   0.20   0.11   0.10   1.00   0.11   0.13
## donor5   0.15   0.23   0.20   0.11   1.00   0.17
## donor6   0.10   0.09   0.15   0.13   0.17   1.00
```

Because it is not a single value per sample or individual, kinship cannot be added to a model with `+ kin`. Instead, it is used as a random effect where you block by individual so the model can identify an individual's kinship values relative to all other individuals in the data set.

`kimma` incorporates the `lme4qtl` package's function `relmatLmer` to use kinship in linear mixed effects models. This feature is why `kimma` exists since pairwise kinship cannot be used in `limma`. We fit this type of model with `run.lmerel = TRUE` providing the kinship matrix.

```
virus_kin <- kmFit(dat = dat, kin = kin,
                   model = "~ virus*asthma + (1|ptID)",
                   run.lmerel = TRUE, metrics = TRUE)
```

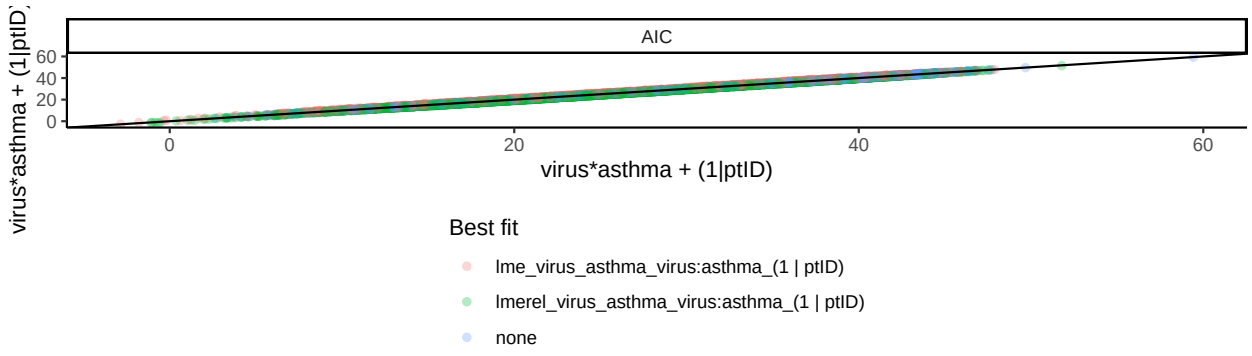
```
lme/lmerel model: expression~virus*asthma+(1|ptID)
Input: 10 libraries from 5 unique patients
Model: 10 libraries
Complete: 13301 genes
Failed: 0 genes
```

We see that kinship does not improve model fit with very small $\Delta < 1$ and even some genes with identical AIC for both models (Best fit = none). We also see kinship has very little impact on DEG detection. Of note, this is not surprising as the example kinship data is made up and does not reflex the actual relatedness of these patients.

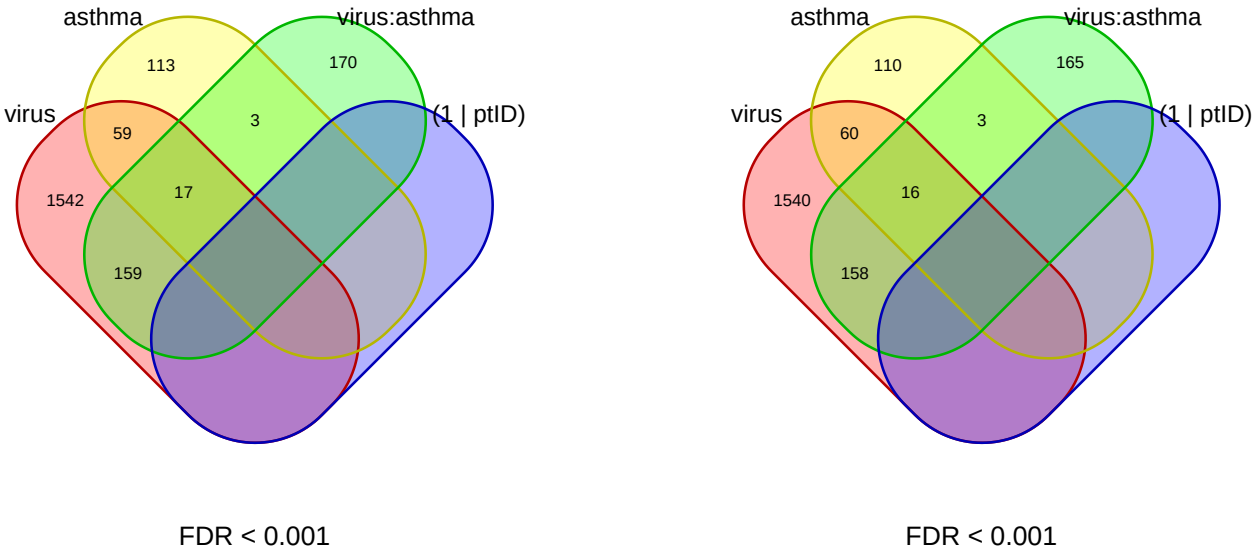
```
plot_fit(model_result = virus_lme, model_result_y = virus_kin,
         x="lme", y="lmerel", metrics = "AIC") +
  labs(x = "virus*asthma + (1|ptID)", y = "virus*asthma + (1|ptID) + kin")
```

```
## Summary
```

```
##                                       Best fit Metric Total genes Mean delta
## 1    lme_virus_asthma_virus:asthma_(1 | ptID)    AIC        3378  0.3209427
## 2 lmerel_virus_asthma_virus:asthma_(1 | ptID)    AIC        6724  0.2904533
## 3                                        none    AIC        3199  0.0000000
##   Stdev delta
## 1   0.3269153
## 2   0.2178309
## 3   0.0000000
```




```
plot_venn_genes(model_result = virus_lme, model = "lme",
                fdr.cutoff = 0.001) +
  plot_venn_genes(model_result = virus_kin, model = "lmerel",
                  fdr.cutoff = 0.001)
```



FDR < 0.001 FDR < 0.001

Thus, we consider our final best fit model to be the previously linear mixed effects model without kinship or co-variates.

`~ virus*asthma + (1|ptID)`

4. Post-hoc pairwise contrasts

If you have an interaction term in your model, you may wish to further probe the results to determine how the interaction is significant. Do healthy donors increase expression in response to virus while those with asthma show no change? Or does everyone decrease in expression but those with asthma decrease more? And other potential outcomes.

Similar to the unpaired model in section 2.2, we can do with for the mixed effects model with `run.contrasts = TRUE`. Here, though, we will only run contrasts on genes that were significant for the interaction term. First, we select interaction DEG at FDR < 0.05.

```
subset.genes <- virus_lme$lme %>%
  filter(variable == "virus:asthma" & FDR < 0.05) %>%
  pull(gene)
```

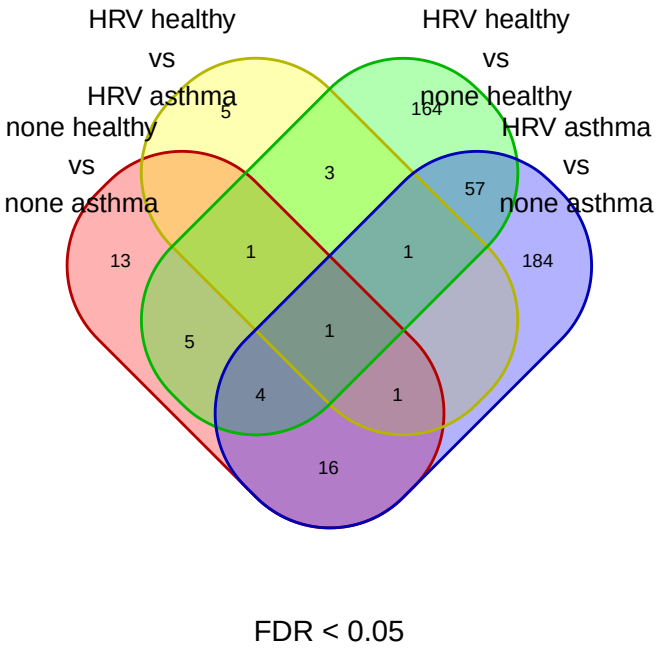
And then run the contrast model on these genes.

```
virus_contrast_signif <- kmFit(dat = dat,
                               model = "~ virus*asthma + (1|ptID)",
                               run.lme = TRUE, run.contrast = TRUE,
                               contrast.var = "virus:asthma",
                               subset.genes = subset.genes)
```

```
lme/lmerel model: expression~virus*asthma+(1|ptID)
Input: 10 libraries from 5 unique patients
Model: 10 libraries
Complete: 839 genes
Failed: 0 genes
```

We see that a number of genes only change significantly with virus in healthy or asthma individuals (right) and that some genes differ with asthma in the presence and/or absence of virus (left). The other two contrasts are not of interest in this analysis since they represent comparisons across both variables.

```
plot_venn_genes(model_result = virus_contrast_signif,
                model = "lme.contrast",
                fdr.cutoff = 0.05,
                contrasts=data.frame(
                  contrast_ref = c("none asthma","HRV asthma",
                                   "none healthy","none asthma"),
                  contrast_lvl = c("none healthy","HRV healthy",
                                   "HRV healthy","HRV asthma")))
```



As noted in section 2.2, you may have instead chosen to use a contrast model for all genes and thus, would not need this redundant post-hoc test.

5. Visualize DEG

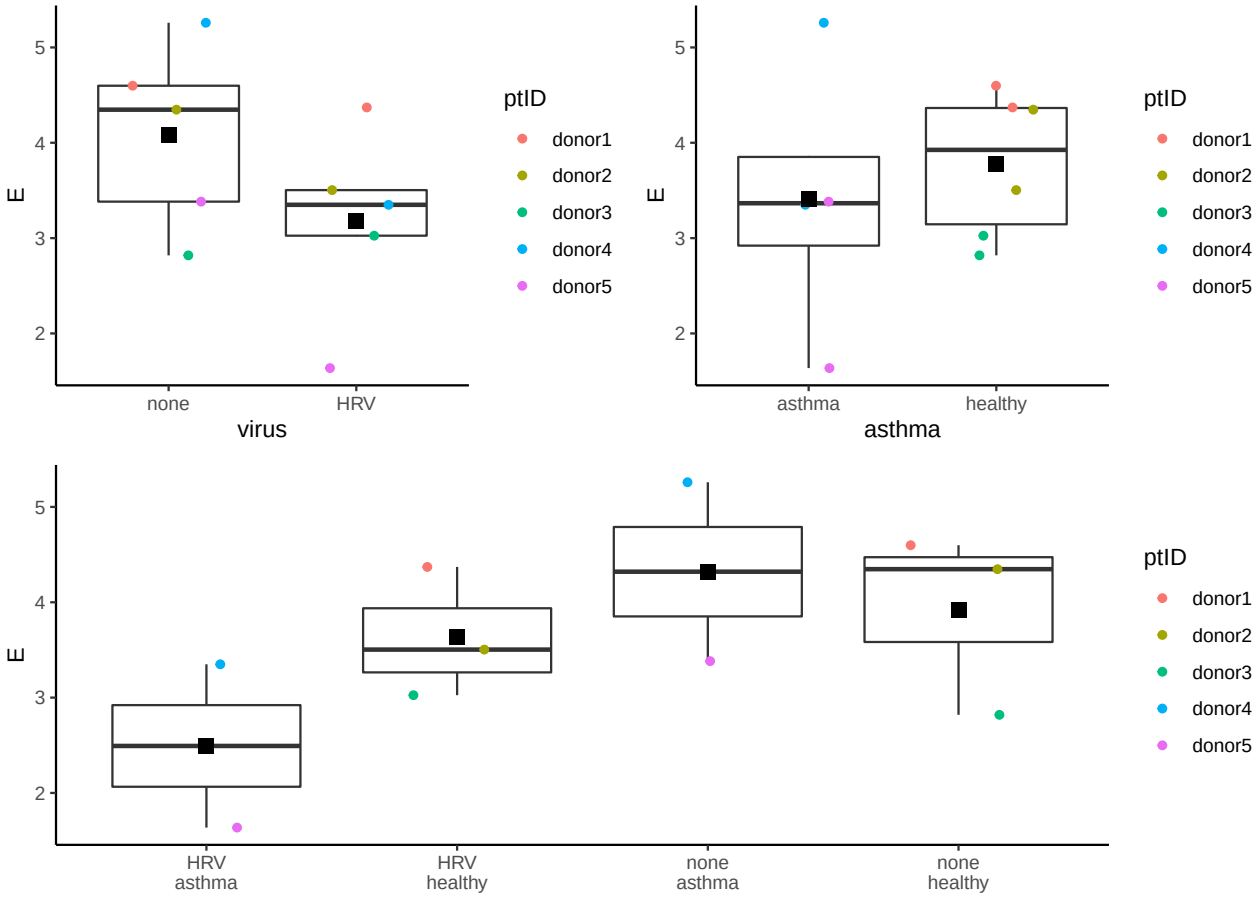
`BIGpicture` provides a function to plot expression for genes of interest across many variables. For example, we plot an interaction significant gene. Each circle is a RNA-seq library colored by donor. The black squares are overall group means and boxplots are standard interquartile ranges.

```
plot_genes(dat = dat, fdr = virus_lme$lme, subset.genes=subset.genes[1],
           variables = "virus*asthma", geneID = "ensembl_gene_id",
           colorID="ptID")
```

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

ENSG00000162496 DHRS3

model	gene	variable	estimate	pval	FDR
lme	ENSG00000162496	virus	−1.8277210	3.515041e−06	3.822858e−05
lme	ENSG00000162496	asthma	−0.3990917	6.777699e−01	9.358420e−01
lme	ENSG00000162496	virus:asthma	1.5392970	1.095259e−04	3.333647e−03
lme	ENSG00000162496	(1 | ptID)	5.1336277	2.346663e−02	6.047493e−01

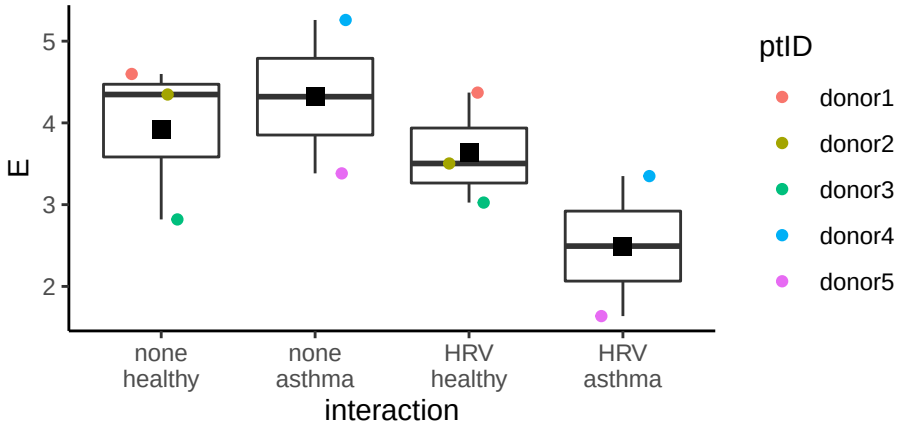


If you want custom ordering within variables, you need to create factors of the correct order.

```
dat$targets <- dat$targets %>%
  mutate(interaction = paste(virus, asthma, sep="\n"),
         interaction = factor(interaction,
                              levels=c("none\nhealthy","none\nasthma",
                                       "HRV\nhealthy","HRV\nasthma")))
```

```
plot_genes(dat = dat, subset.genes=subset.genes[1],
           variables = "interaction",
           geneID = "ensembl_gene_id",
           colorID="ptID")
```

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



6. An actual analysis

This tutorial is meant to walk you through the majority of experimental designs you will encounter in RNAseq analysis. But you do not need to run all these models on an actual data set. For these example data, here is the actual pipeline I would take to select a best fit model for these data.

1. What are the main variables of interest?
   - `virus` and `asthma`
2. Is an interaction necessary?
   - Yes, so my starting model is `~ virus*asthma`
3. Are the data from a paired sample design?
   - Yes, so I need a mixed effects model `~ virus*asthma + (1|ptID)`. I run this model for comparison to those in #4 and #5.
4. Are there potential co-variates?
   - Yes, so my most complicated model would be `~ virus*asthma + batch + sex + (1|ptID)`
5. Is kinship available and of interest?
   - No, so we don't add it to the #4 model.
6. Compare more complex models to the version in #3 as that is the most simple with all main variables and the correct paired sample design.
   - a. `~ virus*asthma + batch + (1|ptID)`. I determine that batch does not improve the model fit enough to be included.
   - b. Remove sex `~ virus*asthma + sex + (1|ptID)`. I determine that sex should not be included as it does not improve the fit.
   - Note that the exclusion of co-variates was also influenced by the small data set size and a desire to use the simplest model possible
   - Also if kinship had been important, I would check that fit before adding co-variates.
7. Run contrasts on interaction significant genes from best model (which ends up being #3)

```
#3
## Run base model
virus_interact <- kmFit(dat = dat,
                        model = "~ virus*asthma + (1|ptID)",
                        run.lme = TRUE, metrics = TRUE)
```

```
#6a
## Explore co-variate batch
plot_pca(dat, vars = "batch")

virus_batch <- kmFit(dat = dat,
                     model = "~ virus*asthma + batch + (1|ptID)",
                     run.lme = TRUE, metrics = TRUE)
## Compare fit
plot_fit(model_result = virus_interact, model_result_y = virus_batch,
           x="lme", y="lme", metrics = "AIC")
## Compare DEG
plot_venn_genes(model_result = virus_interact, model = "lme",
                fdr.cutoff = c(0.05)) +
plot_venn_genes(model_result = virus_batch, model = "lme",
                fdr.cutoff = c(0.05))

#6b
## Explore co-variate sex
plot_pca(dat, vars = "sex")

virus_sex <- kmFit(dat = dat,
                   model = "~ virus*asthma + sex + (1|ptID)",
                   run.lme = TRUE, metrics = TRUE)
## Compare fit
plot_fit(model_result = virus_interact, model_result_y = virus_sex,
           x="lme", y="lme", metrics = "AIC")
## Compare DEG
plot_venn_genes(model_result = virus_interact, model = "lme",
                fdr.cutoff = c(0.05)) +
plot_venn_genes(model_result = virus_sex, model = "lme",
                fdr.cutoff = c(0.05))

#7
## Interaction significant genes
subset.genes <- virus_interact$lme %>%
  filter(variable == "virus:asthma" & FDR < 0.05) %>%
  pull(gene)

contrast_model <- kmFit(dat = dat,
                        model = "~ virus*asthma + (1|ptID)",
                        run.lme = TRUE, run.contrast = TRUE,
                        contrast.var = "virus:asthma",
                        subset.genes = subset.genes)

plot_venn_genes(model_result = contrast_model, model = "lme.contrast",
                fdr.cutoff = 0.05,
                contrasts = data.frame(
                  contrast_ref = c("none asthma","HRV asthma",
                                   "none healthy","none asthma"),
                  contrast_lvl = c("none healthy","HRV healthy",
                                   "HRV healthy","HRV asthma")))
```

R session

```
sessionInfo()
```

```
## R version 4.1.2 (2021-11-01)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur 10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] DiagrammeR_1.0.8 patchwork_1.1.1  edgeR_3.36.0     limma_3.50.3
##  [5] BIGpicture_0.0.2 kimma_1.2.0      forcats_0.5.1    stringr_1.4.0
##  [9] dplyr_1.0.8      purrr_0.3.4      readr_2.1.2      tidyr_1.2.0
## [13] tibble_3.1.6     ggplot2_3.3.5    tidyverse_1.3.1
##
## loaded via a namespace (and not attached):
##  [1] nlme_3.1-155       fs_1.5.2           lubridate_1.8.0    doParallel_1.0.17
##  [5] webshot_0.5.2      RColorBrewer_1.1-3 httr_1.4.2         tools_4.1.2
##  [9] backports_1.4.1    utf8_1.2.2         R6_2.5.1           DBI_1.1.2
## [13] mgcv_1.8-38        colorspace_2.0-3   withr_2.5.0        tidyselect_1.1.2
## [17] processx_3.5.3     compiler_4.1.2     cli_3.3.0          rvest_1.0.2
## [21] xml2_1.3.3         labeling_0.4.2     scales_1.2.0       callr_3.7.0
## [25] digest_0.6.29      rmarkdown_2.11     pkgconfig_2.0.3    htmltools_0.5.2
## [29] dbplyr_2.1.1       fastmap_1.1.0      highr_0.9          htmlwidgets_1.5.4
## [33] rlang_1.0.2        readxl_1.3.1       rstudioapi_0.13    visNetwork_2.1.0
## [37] farver_2.1.0       generics_0.1.2     jsonlite_1.8.0     magrittr_2.0.3
## [41] Matrix_1.4-0       Rcpp_1.0.8.3       munsell_0.5.0      fansi_1.0.3
## [45] ggpolypath_0.1.0   lifecycle_1.0.1    stringi_1.7.6      yaml_2.3.5
## [49] grid_4.1.2         parallel_4.1.2     crayon_1.5.1       venn_1.10
## [53] lattice_0.20-45    haven_2.4.3        cowplot_1.1.1      splines_4.1.2
## [57] hms_1.1.1          locfit_1.5-9.5     knitr_1.39         ps_1.7.0
## [61] pillar_1.7.0       codetools_0.2-18   admisc_0.26        reprex_2.0.1
## [65] glue_1.6.2         evaluate_0.15      modelr_0.1.8       vctrs_0.4.1
## [69] tzdb_0.3.0         foreach_1.5.2      cellranger_1.1.0   gtable_0.3.0
## [73] assertthat_0.2.1   xfun_0.30          broom_0.8.0        iterators_1.0.14
## [77] statmod_1.4.36     ellipsis_0.3.2
```