```
##############################################
# Exon-level intensity data from EXTENDED set #
##############################################

# As done for integration with IGV
# for convenience, we'll refer to HEK293 as "A", 293S as "S", 293T as "T", 293SG
as "RicR" (Ricin Resistant), 293SGGD as "EndoT" and 293FTM as "FlpIn".

#########################################################################
# Background correction, normalisation and probeset summarisation (with APT) #
#########################################################################

# Using Affymetrix Power Tools, or APT under Linux
# Normalisation with the RMA method

# Download the CEL files and put them into your working directory.
# Download the library files for Affymetrix Human Exon arrays, and equally put
them into your working directory.
# Download APT for Linux, and make sure the APT bin folder is part of your PATH.

# Generate exon-level intensity estimates for extended probesets using RMA, and
save it in a folder called OUT_eEXON

apt-probeset-summarize -a rma-sketch -p HuEx-1_0-st-v2.r2.pgf -c
HuEx-1_0-st-v2.r2.clf -s HuEx-1_0-st-v2.r2.dt1.hg18.extended.ps --qc-probesets
HuEx-1_0-st-v2.r2.qcc -o OUT_eEXON *.CEL

#####################################################################
# Comparison with negative controls: antigenomic probes (with APT)#
#####################################################################

# For each probeset (so per exon), the estimated signal intensities are compared
with those from the antigenomic controls (Detection Above Background, or DABG)
with the same GC value. The degree of overlap is used to compute a p-value. This
DABG-file is a matrix of all the computed p-values per probeset and per array.

apt-probeset-summarize -a dabg -p HuEx-1_0-st-v2.r2.pgf -c HuEx-1_0-st-v2.r2.clf
-b HuEx-1_0-st-v2.r2.antigenomic.bgp -o ./OUT_DABG *.CEL

#############################################################
# Filtering of exon data for the full and extended set in R #
#############################################################

#### 1) remove undetected probes #####

# A probeset could be considered detected when the DABG p < 0.05 in 50% of the
samples of at least one group. The rationale for 'at least one group' is that a
skipped exon could be entirely unexpressed in one group but present in another.

pd <- read.table("hekphenodata.txt", header=TRUE)
d.exone <- read.table("OUT_eEXON/rma-sketch.summary.txt", sep="\t", header=T,
row.names=1)
dim(d.exone)
# 807038 18 - there are around 800 000 probesets in the extended set.
colnames(d.exone)

dabg <- read.table("./OUT_DABG/dabg.summary.txt", sep="\t", header=T,
row.names=1)
dim(dabg)
# 1411399 18, so there are about a million probesets that have been compared
with the antigenomic probes to generate a p-value for detection above
background. To check what the dabg file looks like:
head(dabg)

dabg.extended <- dabg[match(row.names(d.exone), row.names(dabg)),]
dim(dabg.extended)
# 807038 18, we now only kept the p-values for the extended probesets
head(dabg.extended)
```

```
# define a function to count how many samples have a detection P < 0.05 and
apply to each group separately

count.det <- function(x){length (which(x<0.05))}

dabg.extendedA <- cbind(dabg.extended[,1], dabg.extended[,7],
dabg.extended[,12])
head(dabg.extendedA)
rownames(dabg.extendedA)<- rownames(dabg.extended)
head(dabg.extendedA)
groupAe.det <-apply(dabg.extendedA[,1:3], 1, count.det)
head(groupAe.det)

dabg.extendedS <- cbind(dabg.extended[,2], dabg.extended[,8],
dabg.extended[,13])
head(dabg.extendedS)
rownames(dabg.extendedS)<- rownames(dabg.extended)
head(dabg.extendedS)
groupSe.det <-apply(dabg.extendedS[,1:3], 1, count.det)
head(groupSe.det)

dabg.extendedT <- cbind(dabg.extended[,3], dabg.extended[,9],
dabg.extended[,14])
head(dabg.extendedT)
rownames(dabg.extendedT)<- rownames(dabg.extended)
head(dabg.extendedT)
groupTe.det <-apply(dabg.extendedT[,1:3], 1, count.det)
head(groupTe.det)

dabg.extendedEndoT <- cbind(dabg.extended[,4], dabg.extended[,15],
dabg.extended[,18])
head(dabg.extendedEndoT)
rownames(dabg.extendedEndoT)<- rownames(dabg.extended)
head(dabg.extendedEndoT)
groupEndoTe.det <-apply(dabg.extendedEndoT[,1:3], 1, count.det)
head(groupEndoTe.det)

dabg.extendedFlpIn <- cbind(dabg.extended[,5], dabg.extended[,10],
dabg.extended[,16])
head(dabg.extendedFlpIn)
rownames(dabg.extendedFlpIn)<- rownames(dabg.extended)
head(dabg.extendedFlpIn)
groupFlpIne.det <-apply(dabg.extendedFlpIn[,1:3], 1, count.det)
head(groupFlpIne.det)

dabg.extendedRicR <- cbind(dabg.extended[,6], dabg.extended[,11],
dabg.extended[,17])
head(dabg.extendedRicR)
rownames(dabg.extendedRicR)<- rownames(dabg.extended)
head(dabg.extendedRicR)
groupRicRe.det <-apply(dabg.extendedRicR[,1:3], 1, count.det)
head(groupRicRe.det)


# retain probesets with P < 0.05 in two or more samples in at least one group

m <- union(which(groupAe.det>=2), which(groupSe.det>=2))
n <- union(which(groupTe.det>=2), which(groupEndoTe.det>=2))
o <- union(which(groupFlpIne.det>=2), which(groupRicRe.det>=2))
p <- union(m,n)
q <- union(p,o)

x <- sort(q)
d.exone.fil <- d.exone[x,]
dim(d.exone.fil) # 652549 18, which is about 80% of the initial 807038

# Save this in a file as a back-up.
```

```
write.table(d.exone.fil, "OUT_eEXON/rma-sketch.summary_extended_exon_filt1.txt",
sep="\t", quote=F, row.names=T)

###### 2) remove cross-hybridizing probesets #####

# For this you need an additional info file from the affy website, your exon
array, the NetAffx annotation files, and the probeset annotations file in cvs
format, latest release.
# Those probes that cross-hybridize typically have a higher signal than expected
as they hybridize to more RNA product. To keep the gene-level estimates as
reliable as possible, they are already removed in the mps-files, but not in the
exon ps-files.

annot <- read.table("HuEx-1_0-st-v2.na29.hg18.probeset.csv", sep=",", header=T)
# should give sth like 1425756 39, check
dim(annot)

# reduce annotation table to core probesets passing the detection filter:

annot.extended <- annot[match(row.names(d.exone.fil), annot[,1]),]
dim(annot.extended) # gives 652549 39
colnames(annot.extended)

# keep only probesets with a value of 1 in the crosshyb_type column (map
uniquely)
# rows containing non-cross-hybridizing probesets
keepe <- which(annot.extended$crosshyb_type==1)
# extract corresponding probeset IDs
idse <- annot.extended[keepe,1]
d.exone.fil3 <- d.exone.fil[match(idse, row.names(d.exone.fil)),]
dim(d.exone.fil3) # gives 526644, so 65% fo the initial 807038
write.table(d.exone.fil3,
"OUT_eEXON/rma-sketch.summary_extended_exon_filt3.txt", sep="\t", quote=F,
row.names=T)


#################################
# Some QC in R for extended set #
#################################

# The 'hekphenodata' text file (tab-delimited) describes which CEL files
correlate with which samples name and group.

pd <- read.table("hekphenodata.txt", header=TRUE)
pd
qcexone <- read.table("OUT_eEXON/rma-sketch.report.txt", sep="\t", header=T)

# Set the column names in the d.exone.fil3 object to match our sample
definitions
colnames(d.exone.fil3)<-pd$Name

# To plot the average raw signal intensity from the non-filtered sets
plot(qcexone$pm_mean, ylim=c(0,1000), xlab="Array", ylab="Signal Intensity",
main="Average Raw Intensity Signal")

# Plot of the deviation of residuals from median from the intensities gotten
from APT, non-filtered as well
plot(qcexone$all_probeset_mad_residual_mean, ylim=c(0,0.5), xlab="Array",
ylab="Mean absolute deviation", main="Deviation of Residuals from Median")

# Density plot before and after filtering
plot(density(d.exone[,1]), main="Raw RMA-normalized intensities", xlab="RMA
normalized intensity")
for(i in 2:ncol(d.exone)) {lines(density(d.exone[,i]))}

plot(density(d.exone.fil[,1]), main="Filter 1 (no undetected probes)", xlab="RMA
normalized intensity")
for(i in 2:ncol(d.exone.fil)) {lines(density(d.exone.fil[,i]))}
```

```
plot(density(d.exone.fil3[,1]), main="Filter 1+2 (no cross-hybdrizing)",
xlab="RMA normalized intensity")
for(i in 2:ncol(d.exone.fil3)) {lines(density(d.exone.fil3[,i]))}

# PCA
colors<-rainbow(nlevels(pd$Line))
pca <- princomp(d.exone.fil3)
plot(pca$loadings, main="Principal Component Analysis", col=colors[pd$Line],
pch=19, cex=2)
text(pca$loadings, as.character(pd$Name), pos=3, cex=0.8)

# Heat map
heatmap(cor(d.exone.fil3), symm=T)

# Boxplot
boxplot(d.exone.fil3,col=rainbow(18),names=pd$Name, main="RMA normalized",
outline=FALSE,las=3,cex.axis=0.4)

# Dendrogram (hierarchical clustering based on p-values)
corClustraw <- pvclust(d.exone, nboot=5, method.dist="correlation")
plot(corClustraw,cex=0.4, main="p-value clustering raw")
corClustfil <- pvclust(d.exone.fil3, nboot=5, method.dist="correlation")
plot(corClustfil,cex=0.4, main="p-value clustering filtered")

# Each plot can also be saved as an image file (.png for example)

png("Distribution_raw_intensities.png")

plot(density(d.exone[,1]), main="Raw RMA-normalized intensities", xlab="RMA
normalized intensity")
for(i in 2:ncol(d.exone)) {lines(density(d.exone[,i]))}

dev.off()

# Similarly, you can save the QC plots in one or separate pdf files.

pdf("extended_exon_QC.pdf")
par(mfrow=c(1,2))

plot(qcexone$pm_mean, ylim=c(0,1000), xlab="Array", ylab="Signal Intensity",
main="Average Raw Intensity Signal")
plot(qcexone$all_probeset_mad_residual_mean, ylim=c(0,0.5), xlab="Array",
ylab="Mean absolute deviation", main="Deviation of Residuals from Median")

par(mfrow=c(1,1))

colors<-rainbow(nlevels(pd$Line))
pca <- princomp(d.exone.fil3)
plot(pca$loadings, main="Principal Component Analysis", col=colors[pd$Line],
pch=19, cex=2)
text(pca$loadings, as.character(pd$Name), pos=3, cex=0.8)
heatmap(cor(d.exone.fil3), symm=T)
boxplot(d.exone.fil3,col=rainbow(18),names=pd$Name, main="RMA normalized",
outline=FALSE,las=3,cex.axis=0.4)

corClustraw <- pvclust(d.exone, nboot=5, method.dist="correlation")
plot(corClustraw,cex=0.4, main="p-value clustering raw")
corClustfil <- pvclust(d.exone.fil3, nboot=5, method.dist="correlation")
plot(corClustfil,cex=0.4, main="p-value clustering filtered")

par(mfrow=c(2,2)
plot(density(d.exone[,1]), main="Raw RMA-normalized intensities", xlab="RMA
normalized intensity")
for(i in 2:ncol(d.exone)) {lines(density(d.exone[,i]))}

plot(density(d.exone.fil[,1]), main="Filter 1 (no undetected probes)", xlab="RMA
normalized intensity")
```

```
for(i in 2:ncol(d.exone.fil)) {lines(density(d.exone.fil[,i]))}

plot(density(d.exone.fil3[,1]), main="Filter 1+2 (no cross-hybdrizing)",
xlab="RMA normalized intensity")
for(i in 2:ncol(d.exone.fil3)) {lines(density(d.exone.fil3[,i]))}

dev.off()


########################################
# Averaging and removal of noisy probes #
########################################

# To present the mean signal intensity per probeset, use the function rowMeans,
after creation of group-specific data frames
groupA <- cbind(d.exone.fil3[,1], d.exone.fil3[,7], d.exone.fil3[,12])
head(groupA)
rownames(groupA) <- rownames(d.exone.fil3)
meanA <- rowMeans(groupA)
lineA <- data.frame(meanA)
head(lineA)

groupS <- cbind(d.exone.fil3[,2], d.exone.fil3[,8], d.exone.fil3[,13])
head(groupS)
rownames(groupS) <- rownames(d.exone.fil3)
meanS <- rowMeans(groupS)
lineS <- data.frame(meanS)
head(lineS)

groupT <- cbind(d.exone.fil3[,3], d.exone.fil3[,9], d.exone.fil3[,14])
head(groupT)
rownames(groupT) <- rownames(d.exone.fil3)
meanT <- rowMeans(groupT)
lineT <- data.frame(meanT)
head(lineT)

groupEndoT <- cbind(d.exone.fil3[,4], d.exone.fil3[,15], d.exone.fil3[,18])
head(groupEndoT)
rownames(groupEndoT) <- rownames(d.exone.fil3)
meanEndoT <- rowMeans(groupEndoT)
lineEndoT <- data.frame(meanEndoT)
head(lineEndoT)

groupFlpIn <- cbind(d.exone.fil3[,5], d.exone.fil3[,10], d.exone.fil3[,16])
head(groupFlpIn)
rownames(groupFlpIn) <- rownames(d.exone.fil3)
meanFlpIn <- rowMeans(groupFlpIn)
lineFlpIn <- data.frame(meanFlpIn)
head(lineFlpIn)

groupRicR <- cbind(d.exone.fil3[,6], d.exone.fil3[,11], d.exone.fil3[,17])
head(groupRicR)
rownames(groupRicR) <- rownames(d.exone.fil3)
meanRicR <- rowMeans(groupRicR)
lineRicR <- data.frame(meanRicR)
head(lineRicR)

meantable <- cbind(lineA, lineS, lineT, lineEndoT, lineFlpIn, lineRicR)
head(meantable)

## Without removing noisy probesets ###

# Load in the probeset annotation file
annotfile <- "HuEx-1_0-st-v2.na29.hg18.probeset.csv"
anTable <- read.table(annotfile, sep=",", header=TRUE, quote="\"",row.names=1)
dim(anTable) # 1425756 38
colnames(anTable)
```

```
# We're only interested in the filtered probesets
extended_anTable <- anTable[rownames(d.exone.fil3),]
dim(extended_anTable) # 526644 38

final_eexon_list <- cbind(meantable, extended_anTable)
dim(final_eexon_list) # 526644 44
write.table(final_eexon_list, "OUT_eEXON/extended_exon_means_annotated.txt",
sep="\t", quote=F, row.names=T)

## Removal of noise ##

# Despite the extensive filtering done above, there is still some noise left.
This can both be seen on the density plots of the filtered data (widening at the
base if the graph), and on exons where we do not expect to get a signal, but get
one anyway (eg genes on the Y chromosome - the HEK lines are female).
# As the average intensity signal is around 6 on these 'forbidden' loci, we'll
consider only the signals 1 log2 above that noise to be true signals (value of
7).
# Following that rationale, we will only keep the probesets for which the
average value is 7 or more in at least 1 group (or cell line in this case). Note
that this is a very stringent requirement.

# Define a new function, to count how many lines have a log2 signal intensity
value >=7. Then apply on rows.
count.noise <- function(x){length (which(x>=7))}
count.table <- apply(meantable[,1:6], 1, count.noise)

# The probesets for which at least one cell line has a mean value of 7 or more
noisefree <- which(count.table>=1)
head(noisefree)
class(noisefree)
noisefree.sorted <- sort(noisefree)
length(noisefree.sorted) # is now 281117, so roughly half is kept

final_eexon_n <- final_eexon_list[noisefree.sorted,]
dim(final_eexon_n) # 281117 44
head(final_eexon_n[1:5,1:8])
write.table(final_eexon_n,
"OUT_eEXON/extended_exon_means_annotated_noiseout.txt", sep="\t", quote=F,
row.names=T)
```