# Code for GENOME/2011/134767

Please find the original *R* code included as an attachment to this document.

---

```
########## Initialization for RuleFit, see http://www-stat.stanford.edu/~jhf/R-RuleFit.html for instructions
platform = ''                                   # 'linux' or 'windows'
rfhome = ''                                     # directory with the RuleFit files
source(paste(rfhome, '/rulefit.r', sep=''))     # initialize RuleFit
library(akima, lib.loc=rfhome)
########## Other packages
library(rms)

########## Load input files
if (!exists('x_long')) {
    xpr_dir <- ''      # expression and peptide hit data directory
    # read expression data (x)
    # header: names of experiments,
    #         format: "RNASEQ_LongOrShortAndPolyaOrNonpolya_CellLine_Compartment_Replicate"
    # columns: experiments
    # rows: genes
    # cells: RPKM
    # sorted by GENCODE genes (1-18063: coding, 18064-27340: noncoding)
    x_long <- read.table( paste(xpr_dir, 'gencode_v7_hg19_gn_with115_cshl_long_quantif_All.txt', sep=''),
                          header = T, row.names=1)
    x_long <- x_long[sort(names(x_long))]
    # XXX remark: currently only the long RNAs are used
    # store the legible names in 'xnames'
    xnames <- names(x_long)
    names(xnames) <- names(x_long)
    for (i in names(x_long)) {
        num <- which(names(x_long) == i)
        i <- strsplit(i, '_')[[1]]
        j <- i[3]                       # cell line
        if (i[2] == 'LONGPOLYA') j <- c(j, 'PolyA+')
        #if (i[2] == 'LONGNONPOLYA') j <- c(j, 'PolyA−')
        if (i[2] == 'LONGNONPOLYA') j <- c(j, 'PolyA-')
        if (i[2] == 'TOTAL') j <- c(j, 'Total')
        if (i[4] == 'CELL') j <- c(j, "Whole Cell")
        else j <- c( j, paste( toupper(substring(i[4], 1, 1)), tolower(substring(i[4], 2)),
                               sep="", collapse=" ") )
```

```
    if (as.integer(i[5]) %% 2 != 0) j <- c(j, 'Rep1')
    else j <- c(j, 'Rep2')
    xnames[num] <- paste(j, collapse=' ')
    }
# GENCODE labels (1 if coding, 0 is non-coding)
gencode_long <- 1:dim(x_long)[1]
gencode_long[1:18063] <- 1                              # GENCODE coding genes
gencode_long[18064:length(gencode_long)] <- 0          # GENCODE non-coding genes
# header: sum(E_score) and num_prots
# columns: E-score and number of peptides
# rows: genes
# pooled peptides
# GM12878
y_long_gm12878 <- read.table( paste(xpr_dir,
                'gencode_v7_hg19_gn_with115_cshl_long_quantif_All_prots_GM12878_third.txt', sep=''),
                header = F, row.names=1)
# K562
y_long_k562 <- read.table( paste(xpr_dir,
                'gencode_v7_hg19_gn_with115_cshl_long_quantif_All_prots_K562_third.txt', sep=''),
                header = F, row.names=1)
# peptides by compartment
# GM12878
y_long_gm12878_nuclear <- read.table( paste(xpr_dir,
                'gencode_v7_hg19_gn_with115_cshl_long_quantif_All_prots_GM12878_nuclear.txt', sep=''),
                header = F, row.names=1)
y_long_gm12878_membrane <- read.table( paste(xpr_dir,
                'gencode_v7_hg19_gn_with115_cshl_long_quantif_All_prots_GM12878_membrane.txt', sep=''),
                header = F, row.names=1)
y_long_gm12878_mito <- read.table( paste(xpr_dir,
                'gencode_v7_hg19_gn_with115_cshl_long_quantif_All_prots_GM12878_mito.txt', sep=''),
                header = F, row.names=1)
y_long_gm12878_cell <- read.table( paste(xpr_dir,
                'gencode_v7_hg19_gn_with115_cshl_long_quantif_All_prots_GM12878_cell.txt', sep=''),
                header = F, row.names=1)
# K562
y_long_k562_nuclear <- read.table( paste(xpr_dir,
                'gencode_v7_hg19_gn_with115_cshl_long_quantif_All_prots_K562_nuclear.txt', sep=''),
                header = F, row.names=1)
y_long_k562_cytosol <- read.table( paste(xpr_dir,
                'gencode_v7_hg19_gn_with115_cshl_long_quantif_All_prots_K562_cytosol.txt', sep=''),
                header = F, row.names=1)
y_long_k562_membrane <- read.table( paste(xpr_dir,
                'gencode_v7_hg19_gn_with115_cshl_long_quantif_All_prots_K562_membrane.txt', sep=''),
                header = F, row.names=1)
y_long_k562_mito <- read.table( paste(xpr_dir,
                'gencode_v7_hg19_gn_with115_cshl_long_quantif_All_prots_K562_mito.txt', sep=''),
                header = F, row.names=1)
y_long_k562_cell <- read.table( paste(xpr_dir,
                'gencode_v7_hg19_gn_with115_cshl_long_quantif_All_prots_K562_cell.txt', sep=''),
                header = F, row.names=1)
```

```
    }


########## Variables for experiment selection:
#rnas = c('long', 'short') # short is currently not used
rnas = c('long')
cell_lines = c('gm12878', 'k562')
compartments <- c('cytosol', 'cell', 'nucleoplasm', 'chromatin', 'nucleus', 'nucleol')
polya <- '[^non]polya'
nonpolya <- 'nonpolya'
rep1 <- '_[13]' # replicate 1 or 3
rep2 <- '_[24]' # replicate 2 or 4

########## Functions
#### Experiment selection                          # Remark: ability to treat short RNA data is included
gx <- function(..., dataset='long'){                # ...: list of properties, dataset: "long" or "short"
    pattern <- list(...)
    xt <- get(paste('x', dataset, sep='_'))         # x_long or x_short
    xfilt <- 1:length(xt)
    for (p in pattern){
        if (p == 'all') { p <- '' }
#        if ((dataset=='short') & (p=='polya')) { p <- '' }     # not used
        if (p == 'polya') { p <- '[^non]polya' }                # fallback if "polya" variable is not used
        xfilt <- intersect(xfilt, grep(tolower(p), tolower(names(xt))))
        }
    if (length(xfilt)!=0)
        return(xt[xfilt])
    else
        return(F)
    }

#### Make sample
# 'all', 'same', 'remain': 50-50% peptide-no_peptide sample
# 'xv': sample for cross-validation (partition to k parts)
make_sample <- function(y, sample_length=1, type='remain', k=10) {
    yc_nz <- which(y>0)         # indices of nonzero peptide
    yc_z <- which(y<=0)         # indices of zero peptide
    # make a training set with 50% zero and nonzero peptide hits
    smpl_len <- length(yc_nz)*sample_length/2
    smpl_nz <- sample(yc_nz, smpl_len)
    smpl_z <- sample(yc_z, smpl_len)
    # combine training set
    smpl_trn <- c(smpl_nz, smpl_z)
    smpl_nontrn <- (1:length(y))[-smpl_trn]
    if (type=='same') smpl_test <- sample(smpl_nontrn, smpl_len)        # test on same length
    else smpl_test <- smpl_nontrn                                       # test on remaining data
    if (type=='all') smpl_test <- 1:length(y)                           # test on all
    if (type == 'xv') {
        smpl_xv <- c()
        smpl <- sample(1:length(y))
```

```
        len <- length(y)/k
        for (i in 1:k) smpl_xv <- cbind(smpl_xv, list(smpl[seq((i-1)*len+1,i*len)]))
        return(smpl_xv)
        }
    else return(list(train=smpl_trn, test=smpl_test))
    }

#### Fit the model
# valid models: 'rulefit', 'lrm' (logit from 'rms' package), 'glm' (logit) [not implemented]
# y: peptide hits (positive integer); signif: significant peptide hit is larger than this
# test: 'remain' -- test on remaining;
#       'same' -- test on same size sample;
#       'all' -- use all data;
#       data.frame(x,y) -- test on it;
#       'xv' cross validation
model_fit  <- function( x, y, model='rulefit', signif=0, sample_length=1, test='remain', xv=NULL,
                        # parameters for logistic regression
                        second.order=TRUE, maxit=1000,
                        # parameters for rulefit
                        sparse=1, tree.size=4, mod.sel=2, costs=c(1,1),
                        model.type="both",
                        ... )
    {
    yc <- y                             # classification
    yc[y>signif] <- 1                   # 1 if 1 or more peptides found
    if (model=='rulefit') {
        yc[y<=signif] <- -1             # -1 if zero peptide found (rulefit)
        }
    else {
        yc[y<=signif] <- 0              # 0 if zero peptide found (others)
        }
    if (!is.null(xv)) {
        if (test!='xv') {
            print('Warning: test method other than \"xv\" provided, using \"xv\" instead')
            test <- 'xv'
            }
        }
    if (test=='xv') {
        if (is.null(xv)) print('Error: no cross-validation sample (xv)')
        smpl_trn <- xv$train
        smpl_test <- xv$test
        }
    else {
        tmp <- make_sample(yc, type=test)
        smpl_trn <- tmp$train
        smpl_test <- tmp$test
    }
    if (is.data.frame(test)) {
        if (!exists(test$x) | !exists(test$y)) print('Error: wrong test set, use data.frame(y=..., x=...)')
        xt <- as.matrix(test$x)
```

```
        yt <- test$y
        }
    else {
        if (test != 'same' & test != 'remain' & test != 'all' & test != 'xv') {
            print('Error: test set is not a data.frame')
            }
        xt <- as.matrix(x[smpl_test,])
        yt <- yc[smpl_test]
        }
    # fit the model
    if (model == 'rulefit') {
        fit <- rulefit( x=as.matrix(x[smpl_trn,]), y=yc[smpl_trn], rfmode='class', quiet=T, ...)
        fitted <- rfpred(xt)
        fitted[fitted==-1] <- 0     # synchronize with logit
        yt[yt==-1] <- 0             # -''-
        vimp_tmp <- varimp(plot=F)
        vimp <- data.frame(varimp=vimp_tmp$imp, row.names=names(x)[vimp_tmp$ord])
        gof <- NULL
        }
    if (model == 'lrm') {
        if (second.order) formula <- y ~ .^2
        else formula <- y ~ .
        fit <- lrm( formula, data=data.frame(y=yc,x), subset=smpl_trn,
                    x=TRUE, y=TRUE, se.fit=TRUE, maxit=maxit)
        fitted <- predict.lrm(fit, newdata=xt) #?
        vimp_tmp <- fastbw(fit, rule='aic', aics=10000)
        vimp <- data.frame(varimp=vimp_tmp$result[,3]*100, row.names=names(vimp_tmp$result[,3]))
        gof <- residuals.lrm(fit, type='gof')
        }
#    TODO glm() fitting is not implemented
#    if (model == 'glm') {
#        fit <- glm( y~.^2, data=data.frame(y=yc,x), family=binomial(link='logit'),
#                    subset=smpl_trn, x=TRUE, y=TRUE, maxit=maxit)
#        fitted <- predict(fit, newdata=as.data.frame(xt))
#        vimp <- NULL
#        gof <- NULL
#        }
    return(list(    x = x,
                    xt = xt,
                    y = data.frame(fitted = fitted, true = yt),
                    smpl_trn = smpl_trn,
                    smpl_test = smpl_test,
                    model = fit,
                    gof = gof,
                    vimp = vimp
         ))
    }

# calculate metrics of fit
calc_metrics <- function(dataframe, cutoff=0.5) {
```

```
    Y <- dataframe$true                 # true peptide hits
    yhat <- dataframe$fitted            # fitted peptide hits
    tp <- sum(Y == 1 & yhat >= cutoff)
    tn <- sum(Y != 1 & yhat < cutoff)
    fp <- sum(Y != 1 & yhat >= cutoff)
    fn <- sum(Y == 1 & yhat < cutoff)
    prec <- tp/(tp+fp)
    recall <- tp/(tp+fn)
    return(list(    len=length(Y),
                    tp=tp,
                    tn=tn,
                    fp=fp,
                    fn=fn,
                    fpr=fp/tp,
                    fnr=fn/tn,
                    accuracy=(tp+tn)/length(Y),
                    prec=prec,
                    recall=recall,
                    F1=2*prec*recall/(prec+recall),
                    F05=(1+0.5^2)*prec*recall/(0.5^2*prec+recall),      # more weight on recall
                    F2=(1+2^2)*prec*recall/(2^2*prec+recall) ))         # -"- on precision
    }

# calculate the binary cutoff
calc_cutoff <- function(range, dataframe, metric = 'F1', quiet=T) {      # any metric from calc_metric
    r=c()
    for (i in range) {
        r <- c(r, calc_metrics(dataframe, i)[metric][[1]])
        }
    if (!quiet) print(calc_metrics(dataframe, range[which.max(r)]))
    return(range[which.max(r)])
    }

# make [0,1] output from the fitted values, based on a cutoff
normalize <- function(dataframe, cutoff=0.5) {
    fitted_norm <- dataframe$fitted
    fitted_norm[fitted_norm >= cutoff] = 1
    fitted_norm[fitted_norm < cutoff] = 0
    return(data.frame( fitted=fitted_norm, true=dataframe$true ))
    }

# cross-validation
xval <- function(x, y, k=10, cutoff=0, metric=NULL, ...) {
    smpl <- make_sample(y=y, type='xv', k=k)
    r <- c()
    vimp <- ''
    parameters <- list(...)
    if (length(parameters)==0) parameters <- list(NULL=NULL)
    longnames <- TRUE
    if (exists('model', parameters)) if (parameters$model=='lrm') longnames <- FALSE
```

```
    for (i in 1:k) {
        smpl_i <- data.frame(train=unlist(smpl[,-i]), test=unlist(smpl[,i]))
        fitted <- model_fit(x, y, test='xv', xv=smpl_i, ...)
        if (!is.null(metric)) {
            co <- calc_cutoff(seq(-5,5,0.1), fitted$y, metric = metric)
            ms <- calc_metrics(fitted$y, cutoff=co)
            }
        else ms <- calc_metrics(fitted$y, cutoff=cutoff)
        r <- rbind(  r, matrix( c(ms$accuracy, ms$prec, ms$recall, ms$F1, ms$F05, ms$F2),
                                nrow=1, ncol=6,
                                dimnames=list(i,c('accuracy','precision','recall','F1','F(0.5)','F(2)'))
                                )
                    )
        if (longnames) vimp <- paste( vimp, i, paste('\t', xnames[row.names(fitted$vimp)[1:5]],
                                                     '\t', fitted$vimp[1:5,], collapse='\n'),
                                      '\n', sep='')
        else vimp <- paste( vimp, i, paste(  '\t', row.names(fitted$vimp)[1:5],
                                             '\t', fitted$vimp[1:5,], collapse='\n'),
                            '\n', sep='')
        }
    return(list(measures=r, vimp=vimp))
    }

# expectations
expect <- function(y) {             # y = normalize(model_fit(..., test='all')$y)
    mrna <- 100*sum(1/(1+exp(-y$fitted[gencode_long==1])))/sum(gencode_long==1)
    lncrna <- 100*sum(1/(1+exp(-y$fitted[gencode_long==0])))/sum(gencode_long==0)
    print(paste('mRNA',mrna))
    print(paste('lncRNA',lncrna))
    return(list(mrna=mrna, lncrna=lncrna))
    }

##### Plots
# variable importance plots
viplot <- function(vimp, num=FALSE) {
    if (!num) num <- dim(vimp)[1]
    par(mar=c(20,4,4,2))
    barplot(as.matrix(vimp)[1:num], beside=T, space=0.3, names.arg=row.names(vimp)[1:num],
            ylim=c(0,100), las=2)
    }

# bivariate importance plots
pairplots <- function() {
    rf <- model_fit(gx('gm12878', rep1), y_long_gm12878)
    pdf('gm12878-pairplots.pdf')
    l <- dim(rf$x)[2]
    for (i in 1:(l-1)) {
        for (j in (i+1):l) {
            pairplot(i,j)
            }
```

```
        }
    dev.off()

    rf <- model_fit(gx('k562', rep1), y_long_k562)
    pdf('k562-pairplots.pdf')
    l <- dim(rf$x)[2]
    for (i in 1:(l-1)) {
        for (j in (i+1):l) {
            pairplot(i,j)
            }
        }
    dev.off()
    }

if ( !exists('minx') ) minx <- min(log10(unlist(x_long))[log10(unlist(x_long))!=-Inf])
if ( !exists('maxx') ) maxx <- max(log10(unlist(x_long))[log10(unlist(x_long))!=-Inf])

# scatterplots
scatterplots <- function() {
    labels <- c('Whole Cell', 'Cytosol', 'Nucleus', 'PolyA+', 'PolyA-')
    names(labels) <- c('cell', 'cytosol', 'nucleus', 'polya', 'nonpolya')
    pdf('scatterplots.pdf')
    for (compartment in c('cell', 'cytosol', 'nucleus')) {
        for (pa in c('polya', 'nonpolya')) {
            x = rowMeans(gx('gm12878', compartment, pa, dataset='long'))
            y = y_long_gm12878[[1]]
            fm = lm(log10(y)~log10(x), subset=(y!=0&x!=0))
            rsq = summary(fm)["r.squared"]
            corr = cor.test(x, y, method='spearman', exact=FALSE)
            smoothScatter(
                    x=log10(rowMeans(gx('gm12878', compartment, pa, dataset='long')))[gencode_long==1],
                            y=log10(y_long_gm12878[[1]])[gencode_long==1],
                            xlab=expression(paste(log[10], RPKM)),
                            ylab=expression(paste(log[10], sum(E-scores))),
                            main='',
                            nrpoints=0, xlim=c(minx,maxx), ylim=c(2,5.5), xaxs='i', yaxs='i'
                            )
            title(main = paste('GM12878', labels[compartment], labels[pa]))
            mtext(substitute(paste(r^2 == a, ', Spearman ', rho == b), list(a=format(rsq, digits=4),
                    b=format(corr$estimate, digits=4))), line=0.35, cex=0.85)
            points( x=log10(rowMeans(gx('gm12878', compartment, pa, dataset='long')))[gencode_long!=1],
                    y=log10(y_long_gm12878[[1]])[gencode_long!=1],
                    pch=16, cex=1, col="#FF9000C0", xlim=c(minx,maxx), ylim=c(2,5.5), xaxs='i', yaxs='i'
                    )
            }
        }
    for (compartment in c('cell', 'cytosol', 'nucleus')) {
        for (pa in c('polya', 'nonpolya')) {
            x = rowMeans(gx('k562', compartment, pa, dataset='long'))
            y = y_long_k562[[1]]
```

```
            fm = lm(log10(y)~log10(x), subset=(y!=0&x!=0))
            rsq = summary(fm)["r.squared"]
            corr = cor.test(x, y, method='spearman', exact=FALSE)
            smoothScatter(  x=log10(rowMeans(gx('k562', compartment, pa, dataset='long')))[gencode_long==1],
                            y=log10(y_long_k562[[1]])[gencode_long==1],
                            xlab=expression(paste(log[10],RPKM)),
                            ylab=expression(paste(log[10],sum(E-scores))),
                            main='',
                            nrpoints=0, xlim=c(minx,maxx), ylim=c(2,5.5), xaxs='i', yaxs='i'
                            )
            title(main = paste('K562', labels[compartment], labels[pa]))
            mtext(substitute(paste(r^2 == a, ', Spearman ', rho == b), list(a=format(rsq, digits=4),
                    b=format(corr$estimate, digits=4))), line=0.35, cex=0.85)

            points( x=log10(rowMeans(gx('k562', compartment, pa, dataset='long')))[gencode_long!=1],
                    y=log10(y_long_k562[[1]])[gencode_long!=1],
                    pch=16, cex=1, col="#FF9000C0",
                    xlim=c(minx,maxx), ylim=c(2,5.5), xaxs='i', yaxs='i')
            }
        }
    dev.off()
    }

#### Plots for the RuleFit3 model
twovarints <- function() {
    print('GM12878: ')
    pdf('twovarint-gm12878-rep1-1_vs_all.pdf')
    rf <- model_fit(gx('gm12878', rep1), y_long_gm12878)
    null.mods <- intnull(minimized=T)
    par(mar=c(18.1, 6.1, 4.1, 2.1))
    for (i in 1:length(names(rf$x))) { twovarint(names(rf$x)[i], names(rf$x)[-i], null.mods) }
    dev.off()

    print('K562: ')
    pdf('twovarint-k562-rep1-1_vs_all.pdf')
    rf <- model_fit(gx('k562', rep1), y_long_k562)
    null.mods <- intnull(minimized=T)
    par(mar=c(18.1, 6.1, 4.1, 2.1))
    for (i in 1:length(names(rf$x))) { twovarint(names(rf$x)[i], names(rf$x)[-i], null.mods) }
    dev.off()
    }

singleplots <- function() {
    pdf('singleplots-gm12878.pdf')
    rf <- model_fit(gx('gm12878', rep1), y_long_gm12878)
    for (i in 1:length(names(rf$x))) { singleplot(names(rf$x)[i]) }
    dev.off()

    pdf('singleplots-k562.pdf')
    rf <- model_fit(gx('k562', rep1), y_long_k562)
```

```
    for (i in 1:length(names(rf$x))) { singleplot(names(rf$x)[i]) }
    dev.off()
    }

m_gm <- matrix( c(
    c(0, 0, 0, 0, 0, 0),
    c(0, 0, 0, 1, 1, 0),
    c(0, 0, 0, 1, 0, 0),
    c(0, 0, 0, 0, 0, 0),
    c(0, 0, 0, 0, 0, 1),
    c(0, 0, 0, 0, 0, 0)
    ), nrow = 6, ncol=6, byrow=T)

m_k <- matrix( c(
    c(0, 1, 0, 1, 0, 0, 0, 0, 0),
    c(0, 0, 0, 1, 1, 1, 1, 0, 0),
    c(0, 0, 0, 0, 0, 0, 0, 0, 0),
    c(0, 0, 0, 0, 0, 0, 1, 0, 0),
    c(0, 0, 0, 0, 0, 0, 1, 0, 0),
    c(0, 0, 0, 0, 0, 0, 0, 0, 0),
    c(0, 0, 0, 0, 0, 0, 0, 1, 0),
    c(0, 0, 0, 0, 0, 0, 0, 0, 0),
    c(0, 0, 0, 0, 0, 0, 0, 0, 0)
    ), nrow = 9, ncol=9, byrow=T)

pairplots2 <- function() {
    result <- model_fit(gx('gm12878', rep1), y_long_gm12878)
    pdf('3D-pairplots-gm12878.pdf')
    l <- dim(result$x)[2]
    for (i in 1:(l-1)) {
        for (j in (i+1):l) {
            if (m_gm[i,j]!=0) pairplot(i,j, type='persp', theta=-40, phi=30)
            }
        }
    dev.off()
    result <- model_fit(gx('k562', rep1), y_long_k562)
    pdf('3D-pairplots-k562.pdf')
    l <- dim(result$x)[2]
    for (i in 1:(l-1)) {
        for (j in (i+1):l) {
            if (m_k[i,j]!=0) pairplot(i,j, type='persp', theta=-40, phi=30)
            }
        }
    dev.off()
    }

varimps <- function() {
    model_fit(gx('gm12878', rep1), y_long_gm12878)
    pdf('varimp-gm12878-rep1.pdf')
    par(mar=c(20.1, 6.1, 4.1, 2.1))
```

```
    varimp()
    dev.off()
    model_fit(gx('gm12878', rep2), y_long_gm12878)
    pdf('varimp-gm12878-rep2.pdf')
    par(mar=c(20.1, 6.1, 4.1, 2.1))
    varimp()
    dev.off()
    model_fit(gx('k562', rep1), y_long_k562)
    pdf('varimp-k562-rep1-all.pdf')
    par(mar=c(20.1, 6.1, 4.1, 2.1))
    varimp()
    dev.off()
    model_fit(gx('k562', rep2), y_long_k562)
    pdf('varimp-k562-rep2-all.pdf')
    par(mar=c(20.1, 6.1, 4.1, 2.1))
    varimp()
    dev.off()
    model_fit(gx('k562', 'nucleus|cytosol|cell', rep1), y_long_k562)
    pdf('varimp-k562-rep1-overlap.pdf')
    par(mar=c(20.1, 6.1, 4.1, 2.1))
    varimp()
    dev.off()
    model_fit(gx('k562', 'nucleus|cytosol|cell', rep2), y_long_k562)
    pdf('varimp-k562-rep2-overlap.pdf')
    par(mar=c(20.1, 6.1, 4.1, 2.1))
    varimp()
    dev.off()
    }

# correlations
correlation <- function(method='spearman') {        # 'pearson', 'spearman', or 'kendall'
    r <- c()
    for (column in names(gx('gm12878'))) {
        ct <- cor.test( x_long[column][[1]], y_long_gm12878[[2]],
                        method=method, exact=FALSE)
        r <- rbind( r, matrix(c(ct$estimate, ct$p.value),
                    nrow=1, ncol=2, dimnames=list(xnames[column],c('estimate','p_value'))))
        }
    for (column in names(gx('k562'))) {
        ct <- cor.test( x_long[column][[1]], y_long_k562[[2]],
                        method=method, exact=FALSE)
        r <- rbind( r, matrix(c(ct$estimate, ct$p.value),
                    nrow=1, ncol=2, dimnames=list(xnames[column],c('estimate','p_value'))))
        }
    return(r)
    }

# boxplots
boxplots <- function() {
    p=y_long_gm12878[[2]]
```

```
    p[y_long_gm12878[[2]]==1]=1
    p[y_long_gm12878[[2]]>=2&y_long_gm12878[[2]]<=5]=2
    p[y_long_gm12878[[2]]>=6&y_long_gm12878[[2]]<=10]=3
    p[y_long_gm12878[[2]]>=11&y_long_gm12878[[2]]<=20]=4
    p[y_long_gm12878[[2]]>=21&y_long_gm12878[[2]]<=50]=5
    p[y_long_gm12878[[2]]>=51&y_long_gm12878[[2]]<=100]=6
    p[y_long_gm12878[[2]]>=101&y_long_gm12878[[2]]<=200]=7
    p[y_long_gm12878[[2]]>=201&y_long_gm12878[[2]]<=300]=8
    p[y_long_gm12878[[2]]>=301]=9
    pg=factor(p)
    p=y_long_k562[[2]]
    p[y_long_k562[[2]]==1]=1
    p[y_long_k562[[2]]>=2&y_long_k562[[2]]<=5]=2
    p[y_long_k562[[2]]>=6&y_long_k562[[2]]<=10]=3
    p[y_long_k562[[2]]>=11&y_long_k562[[2]]<=20]=4
    p[y_long_k562[[2]]>=21&y_long_k562[[2]]<=50]=5
    p[y_long_k562[[2]]>=51&y_long_k562[[2]]<=100]=6
    p[y_long_k562[[2]]>=101&y_long_k562[[2]]<=200]=7
    p[y_long_k562[[2]]>=201&y_long_k562[[2]]<=300]=8
    p[y_long_k562[[2]]>=301]=9
    pk=factor(p)

    pdf('boxplots.pdf')
    for (i in names(gx('gm12878'))) {
        boxplot( x~grp, data=data.frame(x=log10(gx('gm12878'))[,i], grp=pg),
                        notch=F,
                        names=c('0', '1', '[2, 5]', '[6, 10]', '[11, 20]', '[21, 50]',
                                '[51, 100]', '[101, 200]', '[201, 300]',
                                expression(paste('[301, ',infinity,']'))),
                        ylim=c(-4,4),
                        las=2,
                        ylab=expression(paste(log[10],RPKM)),
                        main=paste(c('Peptide hits for ',xnames[i]),sep='',collapse='') )
        }
    for (i in names(gx('k562'))) {
        boxplot( x~grp, data=data.frame(x=log10(gx('k562'))[,1], grp=pk),
                        notch=F,
                        names=c('0', '1', '[2, 5]', '[6, 10]', '[11, 20]', '[21, 50]',
                                '[51, 100]', '[101, 200]', '[201, 300]',
                                expression(paste('[301, ',infinity,']'))),
                        ylim=c(-4,4),
                        las=2,
                        ylab=expression(paste(log[10],RPKM)),
                        main=paste(c('Peptide hits for ',xnames[i]),sep='',collapse='') )
        }
    dev.off()
}
```