Cardiomyocyte interactome, additional KEGG & Gene Ontology analyses

Sebastian Kurscheid

1 July 2015

Contents

R script for KEGG pathway analysis of cardiomyocyte RNA interactome proteins

Loading libraries and creating custom functions:

```r
library("biomaRt")
library("gdata")
library("GO.db")
library("KEGGREST")
library("ggplot2")
library("gplots")
library("grid")
library("scales")

# function for conversion of Entrez GeneIDs to KEGG gene IDs
keggConv.batch <- function(x, max = 100, org = "mmu", id.type = "ncbi-geneid") {
    if (max > 100) {
        on.exit(print("Maximum number of IDs at a given time is 100"))
    } else {
        x <- paste(id.type, x, sep = ":")
        if (length(x > 100)) {
            d1 <- split(x, ceiling(seq_along(x)/max))
            s1 <- lapply(d1, function(y) {
                keggConv(org, y)
            })
            return(unlist(s1))
        } else {
            d1 <- split(x, ceiling(seq_along(x)/10))
            s1 <- lapply(d1, function(y) {
                keggConv(org, y)
            })
            return(unlist(s1))
        }
    }
}

alternative = "greater"
p.adjust.method = "fdr"
```

Loading data tables.

```r
load("data/kegg.brite.rda")
load("data/interactome.rda")
load("data/wcl.rda")
```

Fetching KEGG identifiers:

```r
ids <- unlist(lapply(strsplit(kegg.brite$C, " "), function(x) x[1]))
rownames(kegg.brite) <- ids
total.keggIDs <- keggLink("mmu", "pathway")
save(total.keggIDs, file = "data/total.keggIDs.rda")
```

We have found a total of 25904 which are used for mapping the WCL and interactome data.

Mapping of WCL protein IDs to KEGG IDs and testing for enrichments against background of all KEGG proteins contained in KEGG pathways.

```r
# retrieved Entrez IDs from Biomart
mouse <- useMart("ensembl", dataset = "mmusculus_gene_ensembl")
human <- useMart("ensembl", dataset = "hsapiens_gene_ensembl")
attribs <- listAttributes(mouse)
pages <- attributePages(mouse)
hsap.attribs <- listAttributes(human)

entrez_ids <- getBM(attributes = c("ensembl_gene_id", "entrezgene"), values = wcl[,
    "ensembl_gene_id"], filters = "ensembl_gene_id", mart = mouse)
wcl.human_homologs <- getBM(attributes = c("ensembl_gene_id", "hsapiens_homolog_ensembl_gene"),
    values = wcl[, "ensembl_gene_id"], filters = "ensembl_gene_id", mart = mouse)

# remove ensembl_gene_ids which have duplicated entrez_ids
entrez_ids <- entrez_ids[-which(duplicated(entrez_ids$ensembl_gene_id)), ]
wcl <- merge(wcl, entrez_ids, by.x = "ensembl_gene_id", by.y = "ensembl_gene_id",
    all.x = T)
wcl.entrezIDs <- unique(wcl[!is.na(wcl$entrezgene), ]$entrezgene)
# this retrieval is fairly slow, therefore the results were written to
# './data'
wcl.keggIDs <- keggConv.batch(wcl.entrezIDs)
save(wcl.keggIDs, file = "data/wcl.keggIDs.rda")

wcl.keggQ <- lapply(wcl.keggIDs, function(x) keggGet(x))
save(wcl.keggQ, file = "data/wcl.keggQ.rda")

wcl.pathways <- unique(unlist(lapply(strsplit(names(unlist(lapply(wcl.keggQ,
    function(x) x[[1]]$PATHWAY))), "\\."), function(x) x[3])))
save(wcl.pathways, file = "data/wcl.pathways.rda")

wcl.pathways.genes <- lapply(wcl.pathways, function(x) keggLink("genes", x))
names(wcl.pathways.genes) <- wcl.pathways
save(wcl.pathways.genes, file = "data/wcl.pathways.genes.rda")

wcl.pathways.genes.entrez_ids <- unique(gsub("mmu:", "", as.character(unlist(wcl.pathways.genes))))
wcl.df <- kegg.brite[gsub("mmu", "", wcl.pathways), ]
wcl.df$ID <- rownames(wcl.df)
wcl.df$total <- rep(0, nrow(wcl.df))
wcl.df$total <- sapply(rownames(wcl.df), function(x) length(wcl.pathways.genes[[paste("mmu",
    x, sep = "")]]))
wcl.df$count <- rep(0, nrow(wcl.df))
wcl.df$frac <- rep(0, nrow(wcl.df))

for (i in rownames(wcl.df)) {
    # print(i)
    kL1 <- keggLink("mmu", paste("mmu", i, sep = ""))
    wcl.df[i, ]$count <- length(which(wcl.keggIDs %in% kL1))
    wcl.df[i, ]$frac <- round(length(which(wcl.keggIDs %in% kL1))/length(kL1) *
        100, 2)
}
```

```r
# extract list of IDs in pathway
wcl.in_path.IDs <- lapply(rownames(wcl.df), function(x) {
    kL1 <- keggLink("mmu", paste("mmu", x, sep = ""))
    in_path <- wcl.keggIDs[which(wcl.keggIDs %in% kL1)]
})

names(wcl.in_path.IDs) <- rownames(wcl.df)

# perform Fisher's Exact Test for each category
bkgd <- length(unique(total.keggIDs))
smpl <- length(wcl.keggIDs)
ftl <- apply(wcl.df[1, ], 1, function(x) {
    ct <- as.integer(x["count"])
    tt <- as.integer(x["total"])
    m1 <- matrix(c(ct, tt, smpl - ct, bkgd - tt), 2, 2)
    fisher.test(m1, alternative = alternative)
})

wcl.df$ft_pval <- unlist(lapply(ftl, function(x) {
    x$p.value
}))
wcl.df$ft_OR <- unlist(lapply(ftl, function(x) {
    x$estimate
}))
wcl.df$ft_fdr <- p.adjust(wcl.df$ft_pval, method = "fdr")
save(wcl.df, file = "data/wcl.df.rda")
```

Mapping of Interactome protein IDs to KEGG IDs and testing for enrichments against background of WCL proteins contained in KEGG pathways.

```r
interactome.entrez_ids <- getBM(attributes = c("ensembl_gene_id", "entrezgene"),
    values = interactome[, "ensembl_gene_id"], filters = "ensembl_gene_id",
    mart = mouse)
interactome.human_homologs <- getBM(attributes = c("ensembl_gene_id", "hsapiens_homolog_ensembl_gene"),
    values = interactome[, "ensembl_gene_id"], filters = "ensembl_gene_id",
    mart = mouse)

# remove ensembl_gene_ids which have duplicated entrez_ids
interactome.entrez_ids <- interactome.entrez_ids[-which(duplicated(interactome.entrez_ids$ensembl_gene_
    ]
interactome <- merge(interactome, interactome.entrez_ids, by.x = "ensembl_gene_id",
    by.y = "ensembl_gene_id", all.x = T)

interactome.entrezIDs <- unique(interactome[!is.na(interactome$entrezgene),
    ]$entrezgene)
save(interactome.entrezIDs, file = "data/interactome.entrezIDs")

interactome.keggIDs <- keggConv.batch(interactome.entrezIDs)
save(interactome.keggIDs, file = "data/interactome.keggIDs.rda")

interactome.keggQ <- lapply(interactome.keggIDs, function(x) keggGet(x))
```

```r
save(interactome.keggQ, file = "data/interactome.keggQ.rda")

interactome.pathways <- unique(unlist(lapply(strsplit(names(unlist(lapply(interactome.keggQ,
    function(x) x[[1]]$PATHWAY))), "\\."), function(x) x[3])))
save(interactome.pathways, file = "data/interactome.pathways.rda")

interactome.pathways.genes <- lapply(interactome.pathways, function(x) keggLink("genes",
    x))
names(interactome.pathways.genes) <- interactome.pathways
save(interactome.pathways.genes, file = "data/interactome.pathways.genes.rda")

interactome.pathways.genes.entrez_ids <- unique(gsub("mmu:", "", as.character(unlist(interactome.pathwa


# create dataframe for counting hits in pathways
interactome.df <- kegg.brite[gsub("mmu", "", interactome.pathways), ]
interactome.df$source <- rep("Interactome", nrow(interactome.df))
interactome.df$ID <- rownames(interactome.df)

# we are now using WCL as background to test for enrichment
i1 <- intersect(rownames(interactome.df), rownames(wcl.df))
interactome.df$total <- rep(0, nrow(interactome.df))
interactome.df[i1, ]$total <- wcl.df[i1, ]$count
interactome.df$count <- rep(0, nrow(interactome.df))
interactome.df$frac <- rep(0, nrow(interactome.df))

for (i in rownames(interactome.df)) {
    kL1 <- keggLink("mmu", paste("mmu", i, sep = ""))
    interactome.df[i, ]$count <- length(which(interactome.keggIDs %in% kL1))
    interactome.df[i, ]$frac <- round(length(which(interactome.keggIDs %in%
        kL1))/length(kL1) * 100, 2)
}

# extract list of IDs in pathway
interactome.in_path.IDs <- lapply(rownames(interactome.df), function(x) {
    kL1 <- keggLink("mmu", paste("mmu", x, sep = ""))
    in_path <- interactome.keggIDs[which(interactome.keggIDs %in% kL1)]
})

# perform Fisher's Exact Test for each category
bkgd <- length(unique(wcl.keggIDs))
smpl <- length(interactome.keggIDs)

ftl <- apply(interactome.df, 1, function(x) {
    ct <- as.integer(x["count"])
    tt <- as.integer(x["total"])
    m1 <- matrix(c(ct, tt, smpl - ct, bkgd - tt), 2, 2)
    fisher.test(m1, alternative = alternative)
})

interactome.df$ft_pval <- unlist(lapply(ftl, function(x) {
    x$p.value
}))
```

```r
interactome.df$ft_OR <- unlist(lapply(ftl, function(x) {
    x$estimate
}))
interactome.df$ft_fdr <- p.adjust(interactome.df$ft_pval, method = p.adjust.method,
    n = nrow(wcl.df))
save(interactome.df, file = "data/interactome.df.rda")
```

KEGG contains three levels/hierarchies (A>B>C), here we summarize the enrichment at B level:

```r
# ---------interactome-summarizing data at 'B' level before doing Fisher's
# Exact test------
interactome.B.df <- data.frame(matrix(ncol = 5, nrow = length(unique(interactome.df$B))))
colnames(interactome.B.df) <- c("B", "A", "total", "count", "source")
interactome.B.df$B <- unique(interactome.df$B)
interactome.B.df$A <- sapply(unique(interactome.df$B), function(x) {
    A <- unique(interactome.df[which(interactome.df$B %in% x), "A"])
})
interactome.B.df$source <- rep("Interactome", nrow(interactome.B.df))
interactome.B.df$total <- sapply(unique(interactome.df$B), function(x) {
    tot <- sum(interactome.df[which(interactome.df$B %in% x), "total"])
})
interactome.B.df$count <- sapply(unique(interactome.df$B), function(x) {
    count <- sum(interactome.df[which(interactome.df$B %in% x), "count"])
})

bkgd <- length(unique(wcl.keggIDs))
smpl <- length(interactome.keggIDs)

ftl <- apply(interactome.B.df, 1, function(x) {
    ct <- as.integer(x["count"])
    tt <- as.integer(x["total"])
    m1 <- matrix(c(ct, tt, smpl - ct, bkgd - tt), 2, 2)
    fisher.test(m1, alternative = alternative)
})

interactome.B.df$ft_pval <- unlist(lapply(ftl, function(x) {
    x$p.value
}))
interactome.B.df$ft_OR <- unlist(lapply(ftl, function(x) {
    x$estimate
}))
interactome.B.df$ft_fdr <- p.adjust(interactome.B.df$ft_pval, method = p.adjust.method,
    n = nrow(wcl.df))
save(interactome.B.df, file = "data/interactome.B.df.rda")
```

Interactome proteins are annotated as RNA-related and un-related proteins, based on annotation analysis (upstream of these steps):

```r
# -----------GO RNA unrelated-------------------------------------------
# subset the interactome table
interactome.go_rna_unrelated <- interactome[which(interactome$GO == "unrelated"),
    ]
interactome.go_rna_unrelated.entrezIDs <- unique(interactome.go_rna_unrelated[!is.na(interactome.go_rna_
    ]$entrezgene)
interactome.go_rna_unrelated.keggIDs <- keggConv.batch(interactome.go_rna_unrelated.entrezIDs)

# dataframe for count data
interactome.go_rna_unrelated.df <- interactome.df
interactome.go_rna_unrelated.df$source <- rep("GO_RNA_unrelated", nrow(interactome.go_rna_unrelated.df))
interactome.go_rna_unrelated.df$ID <- rownames(interactome.go_rna_unrelated.df)
interactome.go_rna_unrelated.df$total <- rep(0, nrow(interactome.go_rna_unrelated.df))

# we are now using WCL as background to test for enrichment
i1 <- intersect(rownames(interactome.go_rna_unrelated.df), rownames(wcl.df))
interactome.go_rna_unrelated.df[i1, ]$total <- wcl.df[i1, ]$count
interactome.go_rna_unrelated.df$count <- rep(0, nrow(interactome.go_rna_unrelated.df))

for (i in rownames(interactome.go_rna_unrelated.df)) {
    kL1 <- keggLink("mmu", paste("mmu", i, sep = ""))
    interactome.go_rna_unrelated.df[i, ]$count <- length(which(interactome.go_rna_unrelated.keggIDs %in%
        kL1))
}

# extract list of IDs in pathway
interactome.go_rna_unrelated.in_path.IDs <- lapply(rownames(interactome.go_rna_unrelated.df),
    function(x) {
        kL1 <- keggLink("mmu", paste("mmu", x, sep = ""))
        in_path <- interactome.go_rna_unrelated.keggIDs[which(interactome.go_rna_unrelated.keggIDs %in%
            kL1)]
    })
names(interactome.go_rna_unrelated.in_path.IDs) <- rownames(interactome.go_rna_unrelated.df)

# perform Fisher's Exact Test for each category Using WCL as background
bkgd <- length(unique(wcl.keggIDs))
smpl <- length(interactome.go_rna_unrelated.keggIDs)

ftl <- apply(interactome.go_rna_unrelated.df, 1, function(x) {
    ct <- as.integer(x["count"])
    tt <- as.integer(x["total"])
    m1 <- matrix(c(ct, tt, smpl - ct, bkgd - tt), 2, 2)
    fisher.test(m1, alternative = alternative)
})

interactome.go_rna_unrelated.df$ft_pval <- unlist(lapply(ftl, function(x) {
    x$p.value
}))
interactome.go_rna_unrelated.df$ft_OR <- unlist(lapply(ftl, function(x) {
    x$estimate
}))
interactome.go_rna_unrelated.df$ft_fdr <- p.adjust(interactome.go_rna_unrelated.df$ft_pval,
    method = p.adjust.method, n = nrow(wcl.df))
save(interactome.go_rna_unrelated.df, file = "data/interactome.go_rna_unrelated.df.rda")

# summarizing data at 'B' level before doing Fisher's Exact test
interactome.go_rna_unrelated.B.df <- data.frame(matrix(ncol = 5, nrow = length(unique(interactome.go_rna
colnames(interactome.go_rna_unrelated.B.df) <- c("B", "A", "total", "count",
    "source")
interactome.go_rna_unrelated.B.df$B <- unique(interactome.go_rna_unrelated.df$B)
interactome.go_rna_unrelated.B.df$A <- sapply(unique(interactome.go_rna_unrelated.df$B),
    function(x) {
        A <- unique(interactome.go_rna_unrelated.df[which(interactome.go_rna_unrelated.df$B %in%
            x), "A"])
    })
interactome.go_rna_unrelated.B.df$source <- rep("GO_RNA_unrelated", nrow(interactome.go_rna_unrelated.B
interactome.go_rna_unrelated.B.df$total <- sapply(unique(interactome.go_rna_unrelated.df$B),
    function(x) {
        tot <- sum(interactome.go_rna_unrelated.df[which(interactome.go_rna_unrelated.df$B %in%
            x), "total"])
    })
interactome.go_rna_unrelated.B.df$count <- sapply(unique(interactome.go_rna_unrelated.df$B),
    function(x) {
        count <- sum(interactome.go_rna_unrelated.df[which(interactome.go_rna_unrelated.df$B %in%
            x), "count"])
    })

# using WCL as background
ftl <- apply(interactome.go_rna_unrelated.B.df, 1, function(x) {
    ct <- as.integer(x["count"])
    tt <- as.integer(x["total"])
    m1 <- matrix(c(ct, tt, smpl - ct, bkgd - tt), 2, 2)
    fisher.test(m1, alternative = alternative)
})

interactome.go_rna_unrelated.B.df$ft_pval <- unlist(lapply(ftl, function(x) {
    x$p.value
}))
interactome.go_rna_unrelated.B.df$ft_OR <- unlist(lapply(ftl, function(x) {
    x$estimate
}))
interactome.go_rna_unrelated.B.df$ft_fdr <- p.adjust(interactome.go_rna_unrelated.B.df$ft_pval,
    method = p.adjust.method, n = nrow(wcl.df))
save(interactome.go_rna_unrelated.B.df, file = "data/interactome.go_rna_unrelated.B.df")
```

RNA-unrelated

```


# -----------GO RNA related-------------------------------------------
interactome.go_rna_related <- interactome[-which(interactome$GO == "unrelated"),
    ]

interactome.go_rna_related.entrezIDs <- unique(interactome.go_rna_related[!is.na(interactome.go_rna_rel
    ]$entrezgene)
interactome.go_rna_related.keggIDs <- keggConv.batch(interactome.go_rna_related.entrezIDs)

# we are testing this subset of 'interactome', therefore we include all the
# pathways from 'interactome'
interactome.go_rna_related.df <- interactome.df
i1 <- intersect(rownames(interactome.go_rna_related.df), rownames(wcl.df))
interactome.go_rna_related.df$total <- rep(0, nrow(interactome.go_rna_related.df))
interactome.go_rna_related.df[i1, ]$total <- wcl.df[i1, ]$count
interactome.go_rna_related.df$source <- rep("GO_RNA_related", nrow(interactome.go_rna_related.df))
interactome.go_rna_related.df$ID <- rownames(interactome.go_rna_related.df)
interactome.go_rna_related.df$count <- rep(0, nrow(interactome.go_rna_related.df))
interactome.go_rna_related.df$frac <- rep(0, nrow(interactome.go_rna_related.df))

for (i in rownames(interactome.go_rna_related.df)) {
    kL1 <- keggLink("mmu", paste("mmu", i, sep = ""))
    interactome.go_rna_related.df[i, ]$count <- length(which(interactome.go_rna_related.keggIDs %in%
        kL1))
}

# extract list of IDs in pathway
interactome.go_rna_related.in_path.IDs <- lapply(rownames(interactome.go_rna_related.df),
    function(x) {
        kL1 <- keggLink("mmu", paste("mmu", x, sep = ""))
        in_path <- interactome.go_rna_related.keggIDs[which(interactome.go_rna_related.keggIDs %in%
            kL1)]
    })
names(interactome.go_rna_related.in_path.IDs) <- rownames(interactome.go_rna_related.df)

# perform Fisher's Exact Test for each category Using WCL as background
bkgd <- length(unique(wcl.keggIDs))
smpl <- length(interactome.go_rna_related.keggIDs)

ftl <- apply(interactome.go_rna_related.df, 1, function(x) {
    ct <- as.integer(x["count"])
    tt <- as.integer(x["total"])
    m1 <- matrix(c(ct, tt, smpl - ct, bkgd - tt), 2, 2)
    fisher.test(m1, alternative = alternative)
})

interactome.go_rna_related.df$ft_pval <- unlist(lapply(ftl, function(x) {
    x$p.value
}))
interactome.go_rna_related.df$ft_OR <- unlist(lapply(ftl, function(x) {
    x$estimate
}))
interactome.go_rna_related.df$ft_fdr <- p.adjust(interactome.go_rna_related.df$ft_pval,
    method = p.adjust.method, n = nrow(wcl.df))
save(interactome.go_rna_related.df, file = "data/interactome.go_rna_related.df")

# summarizing data at 'B' level before doing Fisher's Exact test
interactome.go_rna_related.B.df <- data.frame(matrix(ncol = 5, nrow = length(unique(interactome.go_rna_
colnames(interactome.go_rna_related.B.df) <- c("B", "A", "total", "count", "source")
```

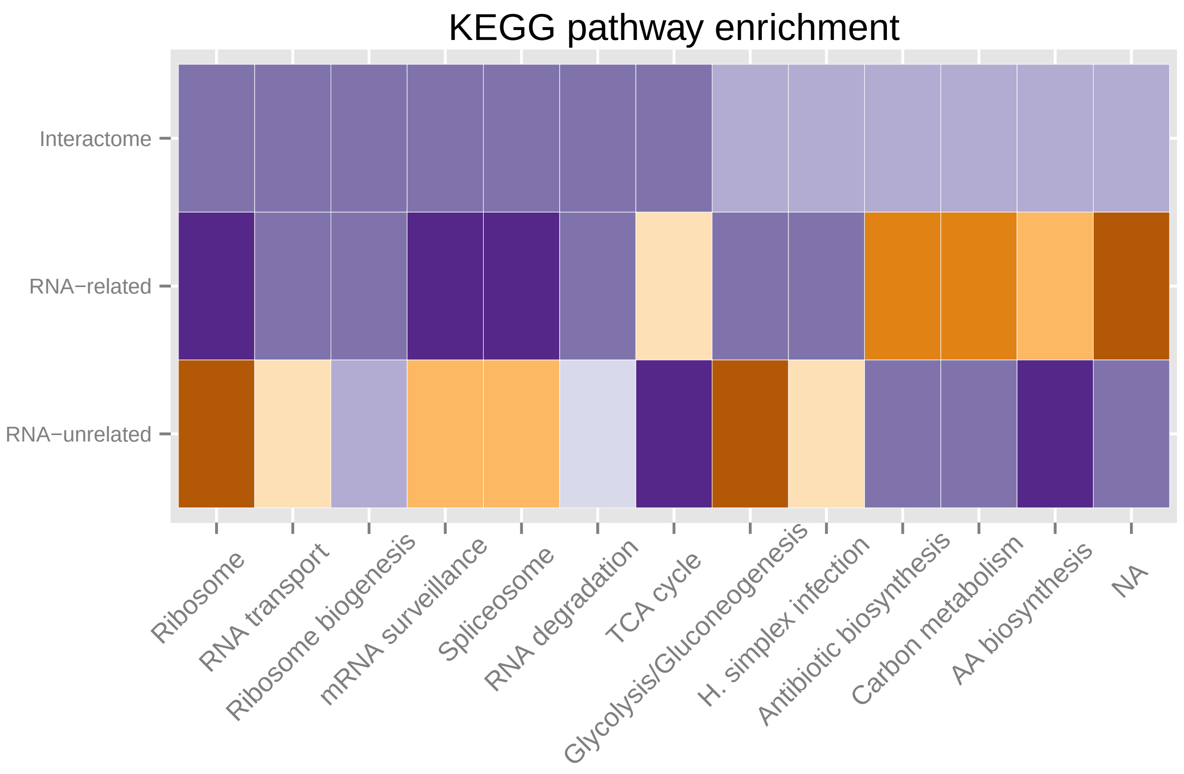
```r
interactome.go_rna_related.B.df$B <- unique(interactome.go_rna_related.df$B)
interactome.go_rna_related.B.df$A <- sapply(unique(interactome.go_rna_related.df$B),
    function(x) {
        A <- unique(interactome.go_rna_related.df[which(interactome.go_rna_related.df$B %in%
            x), "A"])
    })
interactome.go_rna_related.B.df$source <- rep("GO_RNA_related", nrow(interactome.go_rna_related.B.df))
interactome.go_rna_related.B.df$total <- sapply(unique(interactome.go_rna_related.df$B),
    function(x) {
        tot <- sum(interactome.go_rna_related.df[which(interactome.go_rna_related.df$B %in%
            x), "total"])
    })
interactome.go_rna_related.B.df$count <- sapply(unique(interactome.go_rna_related.df$B),
    function(x) {
        count <- sum(interactome.go_rna_related.df[which(interactome.go_rna_related.df$B %in%
            x), "count"])
    })

ftl <- apply(interactome.go_rna_related.B.df, 1, function(x) {
    ct <- as.integer(x["count"])
    tt <- as.integer(x["total"])
    m1 <- matrix(c(ct, tt, smpl - ct, bkgd - tt), 2, 2)
    fisher.test(m1, alternative = alternative)
})

interactome.go_rna_related.B.df$ft_pval <- unlist(lapply(ftl, function(x) {
    x$p.value
}))
interactome.go_rna_related.B.df$ft_OR <- unlist(lapply(ftl, function(x) {
    x$estimate
}))
interactome.go_rna_related.B.df$ft_fdr <- p.adjust(interactome.go_rna_related.B.df$ft_pval,
    method = p.adjust.method, n = nrow(wcl.df))
save(interactome.go_rna_related.B.df, file = "data/interactome.go_rna_related.B.df.rda")
```

RNA-related:

Plotting the results of the enrichment analysis:

```r
library(ggplot2)
library(grid)
library(scales)
load("data/interactome.df.rda")
load("data/interactome.go_rna_unrelated.df.rda")
load("data/interactome.go_rna_related.df")


dfC <- rbind(interactome.df[, c("A", "B", "C", "ft_OR", "ft_fdr", "source", "count")],
             interactome.go_rna_related.df[, c("A", "B", "C", "ft_OR", "ft_fdr", "source", "count")],
             interactome.go_rna_unrelated.df[, c("A", "B", "C", "ft_OR", "ft_fdr", "source", "count")]
)
```

```r
dfC$source <- as.factor(dfC$source)
dfC$source <- factor(dfC$source, levels = levels(dfC$source)[c(3,1,2)])

select1 <- unique(as.character(dfC[which(dfC$ft_fdr <= 0.1 & dfC$ft_OR > 1),]$C))
select1.pathIDs <- paste("mmu", unlist(lapply(strsplit(select1, "\\ "), function(x) x[1])), sep = "")
dfC <- dfC[which(dfC$C %in% select1),]

dfC$C <- as.factor(as.character(dfC$C))
dfC$ft_OR.cut <- cut(log2(dfC$ft_OR), breaks = c(-Inf,-4:4), right = F)
dfC$C <- factor(dfC$C, levels = levels(dfC$C)[dfC[dfC$source == "Interactome", "C"][order(dfC[which(dfC

# formatting labels etc for plotting
l1 <- levels(dfC$ft_OR.cut)
l1 <- gsub("\\[", "", l1)
l1 <- gsub("\\)", "", l1)
levels(dfC$ft_OR.cut) <- l1

l1 <- as.character(levels(dfC$C))
l1 <- unlist(lapply(strsplit(l1, " "), function(x) {
  for (i in 2:length(x)){
    if (i == 2){
      v <- x[i]
    } else {
      v <- paste(v, x[i])
    }
  }
  return(v)
}))
levels(dfC$C) <- l1

levels(dfC$source)[2:3] <- c("RNA-related", "RNA-unrelated")


levels(dfC$C)[3] <- "Ribosome biogenesis"
levels(dfC$C)[6] <- "TCA cycle"
levels(dfC$C)[7] <- "mRNA surveillance"
levels(dfC$C)[11] <- "AA biosynthesis"
levels(dfC$C)[8] <- "H. simplex infection"
levels(dfC$C)[9] <- "Antibiotic biosynthesis"
levels(dfC$C)[12] <- "Glycolysis/Gluconeogenesis"

flevels <- levels(dfC$source)

l1 <- factor(dfC$C, levels = levels(dfC$C)[c(1,2,3,7,4,5,6,12,8,9,10,11)])
dfC$C <- l1

p1 <- ggplot(data = dfC, aes(y = source, x = C)) +
      geom_tile(aes(fill = ft_OR.cut), colour = "white") +
      scale_fill_manual(values = brewer_pal(pal = "PuOr")(8), labels = levels(dfC$ft_OR.cut)) + #
      theme(axis.text.y = element_text(angle = 0, size = 8), axis.title = element_blank()) +
      guides(fill = guide_legend(label.position = "bottom", direction = "horizontal")) +
      theme(axis.text.x = element_text(angle = 45, vjust = 0.9, hjust = 0.8, size = 10)) +
      labs(fill = "Log2 OR") +
      scale_y_discrete(limits = rev(flevels)) +
      theme(legend.position = c(0.4,-1.92),
            legend.text = element_text(size = 4),
            legend.text.align = 0.5,
            legend.title = element_text(size = 4, vjust = 5),
            legend.key.size = unit(3.5, "mm"),
            legend.key.width = unit(3.5, "mm"),
            legend.margin = unit(0, "mm"),
            panel.margin = unit(1, "mm")) +
      ggtitle("KEGG pathway enrichment")

plot(p1)
```



R script for cardiovascular-associated GO term analysis of cardiomyocyte RNA interactome proteins

Load libraries

```
library(gdata)
library(biomaRt)
library(GO.db)
library(ggplot2)

# make lists from GO.db
go.term <- as.list(GOTERM)
```

```r
go.bp.offspring <- as.list(GOBPOFFSPRING)
go.cc.offspring <- as.list(GOCCOFFSPRING)
```

Connect to Biomart

```r
human <- useMart("ensembl", dataset = "hsapiens_gene_ensembl")
mouse <- useMart("ensembl", dataset = "mmusculus_gene_ensembl")
attribs <- listAttributes(mouse)
filters <- listFilters(mouse)
attribs.hsap <- listAttributes(human)
```

Load data files

```r
load("data/wcl.rda")
load("data/interactome.rda")
load("data/cv.assoc.proteins.rda")
```

ID mapping

```r
# biomaRt attribute uniprot_swissprot
mmus.cv.assoc <- getBM(attributes = c("ensembl_gene_id", "uniprot_swissprot"),
    filters = "uniprot_swissprot", values = cv.assoc.proteins[which(cv.assoc.proteins$Taxon ==
        "10090"), "ID"], mart = mouse)
hsap.cv.assoc <- getBM(attributes = c("ensembl_gene_id", "uniprot_swissprot"),
    filters = "uniprot_swissprot", values = cv.assoc.proteins[which(cv.assoc.proteins$Taxon ==
        "9606"), "ID"], mart = human)
hsap.cv.assoc.mmus.homologs <- getBM(attributes = c("ensembl_gene_id", "mmusculus_homolog_ensembl_gene"
    filters = "uniprot_swissprot", values = cv.assoc.proteins[which(cv.assoc.proteins$Taxon ==
        "9606"), "ID"], mart = human)

i1 <- intersect(unique(mmus.cv.assoc$ensembl_gene_id), interactome$ensembl_gene_id)
i2 <- intersect(unique(hsap.cv.assoc.mmus.homologs$mmusculus_homolog_ensembl_gene),
    interactome$ensembl_gene_id)
i3 <- c(i1[which(!i1 %in% intersect(i1, i2))], i2[which(!i2 %in% intersect(i1,
    i2))])

interactome.go_ids <- getBM(attributes = c("ensembl_gene_id", "go_id"), filters = "ensembl_gene_id",
    values = interactome$ensembl_gene_id, mart = mouse)
# subtract genes which are in common between interactome and WCL
wcl <- wcl[-which(wcl$ensembl_gene_id %in% interactome$ensembl_gene_id), ]
wcl.go_ids <- getBM(attributes = c("ensembl_gene_id", "go_id"), filters = "ensembl_gene_id",
    values = wcl$ensembl_gene_id, mart = mouse)

cv.go_terms.bp <- c("GO:0007507", "GO:0048738", "GO:0008015", "GO:0050878",
    "GO:0001944", "GO:0042060", "GO:0006979", "GO:0016055", "GO:0006520", "GO:0050817",
    "GO:0006629", "GO:0006936", "GO:0048771", "GO:0051145", "GO:0007517", "GO:0042692",
    "GO:0048659")
cv.go_terms.cc <- c("GO:0005739", "GO:0005578")
```

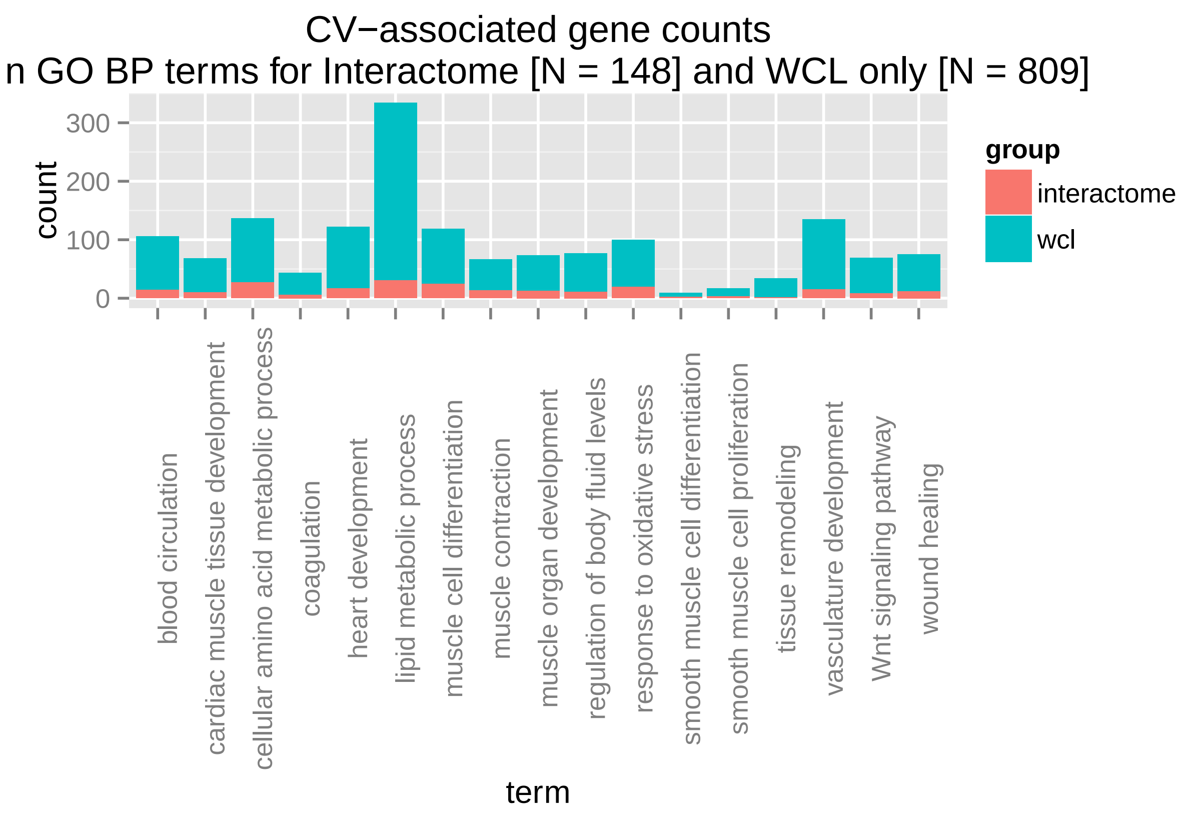
Count term frequencies in Interactome and WCL data and produce bar plots:

```
interactome.cv.go_bp.offsp <- sapply(cv.go_terms.bp, function(x) length(unique(interactome.go_ids[which
    unlist(go.bp.offspring[x])), "ensembl_gene_id"])))
interactome.cv.go_cc.offsp <- sapply(cv.go_terms.cc, function(x) length(unique(interactome.go_ids[which
    unlist(go.cc.offspring[x])), "ensembl_gene_id"])))
interactome.cv.go_bp.offsp.IDs <- sapply(cv.go_terms.bp, function(x) unique(interactome.go_ids[which(in
    unlist(go.bp.offspring[x])), "ensembl_gene_id"]))
interactome.cv.go_cc.offsp.IDs <- sapply(cv.go_terms.cc, function(x) unique(interactome.go_ids[which(in
    unlist(go.cc.offspring[x])), "ensembl_gene_id"]))

wcl.cv.go_bp.offsp <- sapply(cv.go_terms.bp, function(x) length(unique(wcl.go_ids[which(wcl.go_ids$go_i
    unlist(go.bp.offspring[x])), "ensembl_gene_id"])))
wcl.cv.go_cc.offsp <- sapply(cv.go_terms.cc, function(x) length(unique(wcl.go_ids[which(wcl.go_ids$go_i
    unlist(go.cc.offspring[x])), "ensembl_gene_id"])))
wcl.cv.go_bp.offsp.IDs <- sapply(cv.go_terms.bp, function(x) unique(wcl.go_ids[which(wcl.go_ids$go_id %
    unlist(go.bp.offspring[x])), "ensembl_gene_id"]))
wcl.cv.go_cc.offsp.IDs <- sapply(cv.go_terms.cc, function(x) unique(wcl.go_ids[which(wcl.go_ids$go_id %
    unlist(go.cc.offspring[x])), "ensembl_gene_id"]))
```

```
df.go_bp.interactome <- as.data.frame(interactome.cv.go_bp.offsp)
colnames(df.go_bp.interactome) <- "count"
df.go_bp.interactome$id <- rownames(df.go_bp.interactome)
df.go_bp.interactome$group <- "interactome"

df.go_bp.wcl <- as.data.frame(wcl.cv.go_bp.offsp)
colnames(df.go_bp.wcl) <- "count"
df.go_bp.wcl$group <- "wcl"
df.go_bp.wcl$id <- rownames(df.go_bp.wcl)

df.go_bp <- rbind(df.go_bp.interactome, df.go_bp.wcl)

df.go_bp$term <- sapply(df.go_bp$id, function(x) go.term[x][[1]]@Term)
n1 <- length(unique(interactome.go_ids[which(interactome.go_ids$go_id %in% unlist(go.bp.offspring[cv.go
    "ensembl_gene_id"]))
n2 <- length(unique(wcl.go_ids[which(wcl.go_ids$go_id %in% unlist(go.bp.offspring[cv.go_terms.bp])),
    "ensembl_gene_id"]))
```

Plotting the frequency of GO BP term descendants of major cardiovascular-associated GO terms

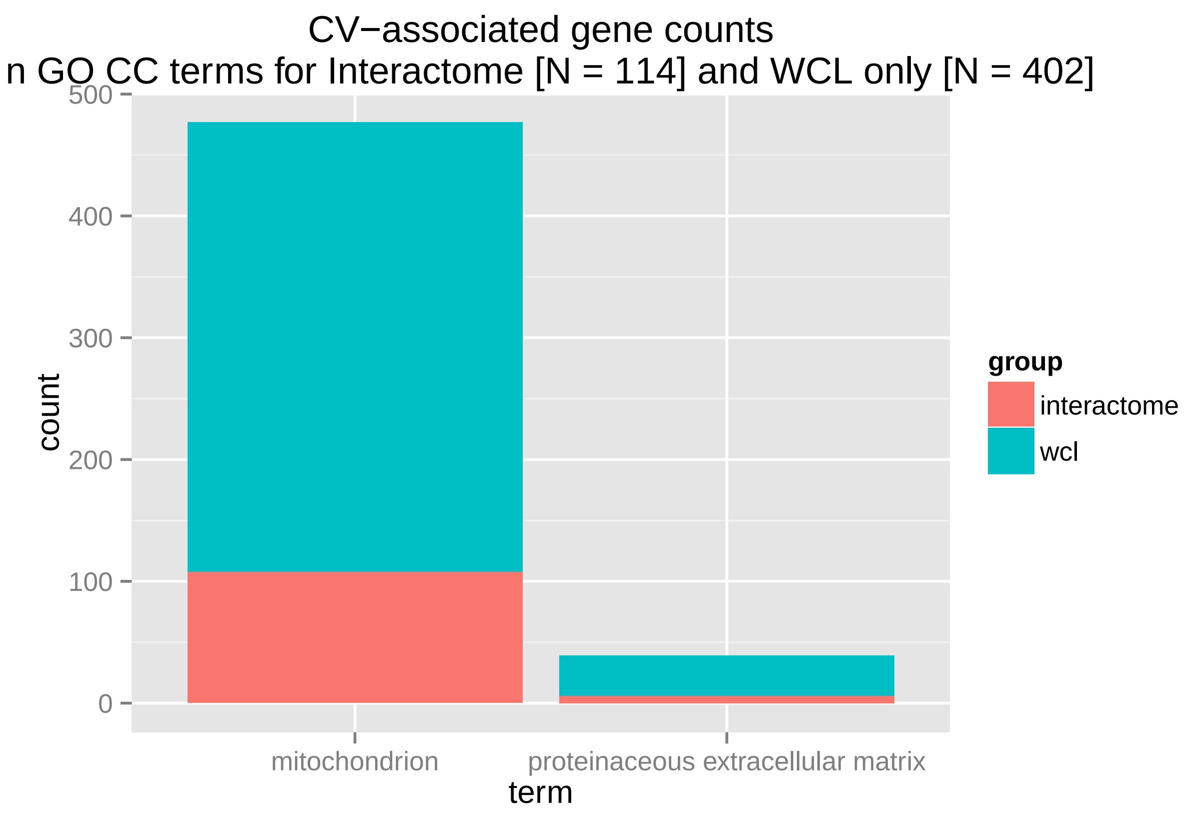
```
hist.go_bp <- ggplot(df.go_bp, aes(term, count, group = group, fill = group)) +
    geom_bar(postion = "dodge", stat = "identity")
hist.go_bp <- hist.go_bp + theme(axis.text.x = element_text(angle = 90))
hist.go_bp <- hist.go_bp + labs(title = paste("CV-associated gene counts\n in GO BP terms for Interacto
    n1, "] and WCL only [N = ", n2, "]", sep = ""))
print(hist.go_bp)
```



```
df.go_cc.interactome <- as.data.frame(interactome.cv.go_cc.offsp)
colnames(df.go_cc.interactome)[1] <- "count"
df.go_cc.interactome$group <- "interactome"
df.go_cc.interactome$id <- rownames(df.go_cc.interactome)
df.go_cc.interactome$term <- sapply(rownames(df.go_cc.interactome), function(x) go.term[x][[1]]@Term)

df.go_cc.wcl <- as.data.frame(wcl.cv.go_cc.offsp)
colnames(df.go_cc.wcl) <- "count"
df.go_cc.wcl$group <- "wcl"
df.go_cc.wcl$id <- rownames(df.go_cc.wcl)
df.go_cc.wcl$term <- sapply(rownames(df.go_cc.wcl), function(x) go.term[x][[1]]@Term)

df.go_cc <- rbind(df.go_cc.interactome, df.go_cc.wcl)
df.go_cc$group <- as.factor(df.go_cc$group)

n1 <- length(unique(interactome.go_ids[which(interactome.go_ids$go_id %in% unlist(go.cc.offspring[cv.go
    "ensembl_gene_id"]))
n2 <- length(unique(wcl.go_ids[which(wcl.go_ids$go_id %in% unlist(go.cc.offspring[cv.go_terms.cc])),
    "ensembl_gene_id"]))
```

Plotting the frequency of GO CC term descendants of major cardiovascular-associated GO terms

```
hist.go_cc <- ggplot(df.go_cc, aes(term, count, group = group, fill = group)) +
    geom_bar(postion = "dodge", stat = "identity")
hist.go_cc <- hist.go_cc + theme(axis.text.x = element_text(angle = 0))
hist.go_cc <- hist.go_cc + labs(title = paste("CV-associated gene counts\n in GO CC terms for Interacto
    n1, "] and WCL only [N = ", n2, "]", sep = ""))
print(hist.go_cc)
```



R script for comparing RBDpeps between human HeLa and mouse HL-1 interactomes

Load libraries

```
library("biomaRt")
library("gdata")
library("ggplot2")
library("Biostrings")
```

Load tables containing the RBDpep data for HL-1 and HeLa

```
load("data/RBDpep.HeLa.rda")
load("data/RBDpep.hl1.rda")

# sorting tables by Ensembl Gene ID and start position of fragment to
# 'linearize' data
RBDpep.hl1 <- RBDpep.hl1[order(RBDpep.hl1$ENSMBL.gene.ID, RBDpep.hl1$Start),
    ]
RBDpep.HeLa <- RBDpep.HeLa[order(RBDpep.HeLa$ENSG, RBDpep.HeLa$Start), ]
```

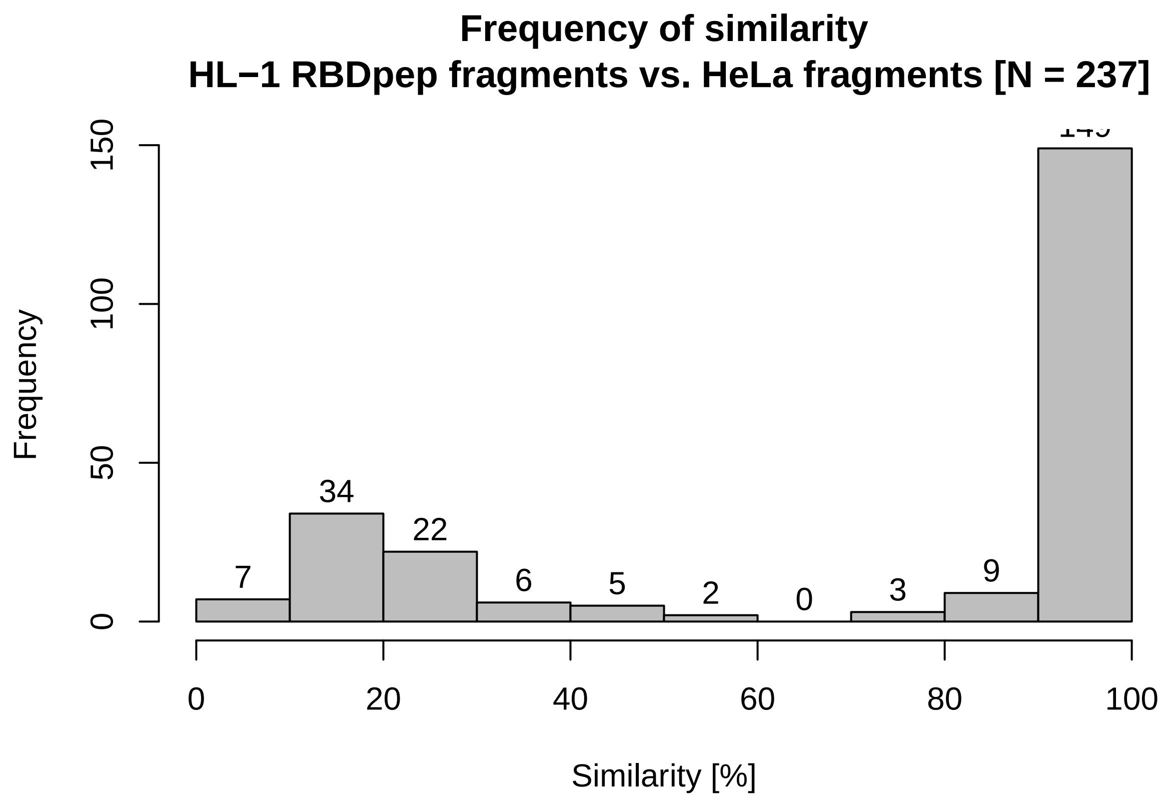
Retrieve human homologs of mouse [HL-1] proteins

```
mmus.RBDpep.hsap.homologs <- getBM(attributes = c("ensembl_gene_id", "description",
    "hsapiens_homolog_ensembl_gene"), filter = "ensembl_gene_id", values = RBDpep.hl1$ENSMBL.gene.ID,
    mart = mouse)

# have a look at SwissProt/TrEMBL UniProt IDs
mmus.uniprot <- getBM(attributes = c("ensembl_gene_id", "uniprot_sptrembl",
    "uniprot_swissprot"), filter = "ensembl_gene_id", values = RBDpep.hl1$ENSMBL.gene.ID,
    mart = mouse)

hsap.i1 <- intersect(RBDpep.HeLa$ENSG, mmus.RBDpep.hsap.homologs$hsapiens_homolog_ensembl_gene)
mmus.i1 <- mmus.RBDpep.hsap.homologs[mmus.RBDpep.hsap.homologs$hsapiens_homolog_ensembl_gene %in%
    hsap.i1, ]$ensembl_gene_id
```

Do pairwise-alignment of the RBDpeps

```
aln.blosum62 <- sapply(mmus.i1, function(x) {
    mmus.frag <- RBDpep.hl1[RBDpep.hl1$ENSMBL.gene.ID == x, ]$Fragment.sequence
    hsap.homolog <- mmus.RBDpep.hsap.homologs[mmus.RBDpep.hsap.homologs$ensembl_gene_id ==
        x, ]$hsapiens_homolog_ensembl_gene
    hsap.frag <- RBDpep.HeLa[RBDpep.HeLa$ENSG %in% hsap.homolog, ]$fragmentSequence
    p <- as.character(hsap.frag)
    s <- as.character(mmus.frag)
    pA <- sapply(s, function(sx) {
        p1 <- pairwiseAlignment(pattern = AAStringSet(p), subject = AAString(sx),
            substitutionMatrix = "BLOSUM62", gapOpening = -12, gapExtension = -5,
            type = "global-local")
        attr(p1, "pid") <- pid(p1)
        attr(p1, "cStr") <- compareStrings(p1)
        return(p1)
    })
})

# subject is fragment from human pattern is fragment from mouse
aln.best <- lapply(aln.blosum62, function(x) lapply(x, function(y) {
    r1 <- y[which.max(pid(y))]
    attr(r1, "pid") <- pid(r1)
    return(r1)
}))

# Mouse as pattern
RBDpep.merge <- RBDpep.hl1[RBDpep.hl1$ENSMBL.gene.ID %in% mmus.i1, ]
RBDpep.merge$hsapHomolog <- as.character(sapply(RBDpep.merge$ENSMBL.gene.ID,
    function(x) mmus.RBDpep.hsap.homologs[mmus.RBDpep.hsap.homologs$ensembl_gene_id ==
        x, ]$hsapiens_homolog_ensembl_gene))
```

```
RBDpep.merge$hsapAlignment <- unlist(lapply(unlist(aln.best[unique(RBDpep.merge$ENSMBL.gene.ID)]),
    function(x) compareStrings(x)))
RBDpep.merge$hsapSimilarity <- unlist(lapply(unlist(aln.best[unique(RBDpep.merge$ENSMBL.gene.ID)]),
    function(x) attr(x, "pid")))
RBDpep.merge$hsapScore <- unlist(lapply(unlist(aln.best[unique(RBDpep.merge$ENSMBL.gene.ID)]),
    function(x) attr(x, "score")))
RBDpep.merge$hsapFragment <- unlist(lapply(unlist(aln.best[unique(RBDpep.merge$ENSMBL.gene.ID)]),
    function(x) toString(unaligned(pattern(x)))))
RBDpep.merge$hsapFragmentStart <- unlist(lapply(apply(RBDpep.merge, 1, function(x) RBDpep.HeLa[RBDpep.He
    x["hsapFragment"], ]), function(y) unique(y["fragmentStart"])))
RBDpep.merge$hsapFragmentStop <- unlist(lapply(apply(RBDpep.merge, 1, function(x) RBDpep.HeLa[RBDpep.Hel
    x["hsapFragment"], ]), function(y) unique(y["fragmentStop"])))

h1 <- hist(RBDpep.merge[!duplicated(RBDpep.merge$hsapFragment), ]$hsapSimilarity,
    plot = F)
h1 <- hist(RBDpep.merge[!duplicated(RBDpep.merge$hsapFragment), ]$hsapSimilarity,
    labels = T, col = "gray", main = paste("Frequency of similarity\n HL-1 RBDpep fragments vs. HeLa fra
        sum(h1$counts), "]", sep = ""), xlab = "Similarity [%]")
```



```
sessionInfo()
```

```
## R version 3.2.1 (2015-06-18)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: OS X 10.10.4 (Yosemite)
##
## locale:
## [1] en_AU.UTF-8/en_AU.UTF-8/en_AU.UTF-8/C/en_AU.UTF-8/en_AU.UTF-8
```

```
##
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
## [1] Biostrings_2.36.1   XVector_0.8.0       IRanges_2.2.5
## [4] S4Vectors_0.6.1     BiocGenerics_0.14.0 scales_0.2.5
## [7] ggplot2_1.0.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.11.6      knitr_1.10.5     magrittr_1.5     zlibbioc_1.14.0
##  [5] MASS_7.3-42      munsell_0.4.2    colorspace_1.2-6 stringr_1.0.0
##  [9] plyr_1.8.3       tools_3.2.1      grid_3.2.1       gtable_0.1.2
## [13] htmltools_0.2.6  yaml_2.1.13      digest_0.6.8     reshape2_1.4.1
## [17] formatR_1.2      codetools_0.2-11 evaluate_0.7     rmarkdown_0.7
## [21] stringi_0.5-5    proto_0.3-10
```