iCLIP project - Day 3: U2AF65

Melina Klostermann & Kathi Zarnack

27 August, 2020

Contents

```
library(GenomicRanges)
library(rtracklayer)
library(knitr)
library(GenomicFeatures)
library(dplyr)
library(ggpubr)
library(BSgenome.Hsapiens.UCSC.hg19)
#library(BSgenome.Hsapiens.NCBI.GRCh38)
library(biomaRt)
library(ggpointdensity)
library(hypeR)
```

```
load(BS_anno_out_file)
```

1 Further exploration of RBP binding behaviour

1.1 Prediction of binding motifs

Many RBPs recognise a certain binding motif. If the binding motif of our RBP is not yet known, we can try to predict it from the binding that we observe in our iCLIP data.

For this we can use the motif prediction tools MEME and DREAM. Both can be run via command line, but there is also a web-suite that can be used:

http://meme-suite.org/tools/meme

http://meme-suite.org/tools/dreme

1.1.1 Export fasta file

Both tools use as input fasta files with the sequences that potentially contain the motif. In case of RBP binding to RNAs in iCLIP data, we would expect a binding motif of the RBP to lie inside or in proximity to our defined binding sites. Usually, a region of 20 nt on either side of the binding sites is a good window to start searching. As calculation of motif enrichment on all binding sites will take quite long, we can select the top 500 binding sites with the highest PureCLIP score.

```
#########################
# prepare binding sites for motif prediction
#########################

Binding_site_window_for_motif <-
  # make data.frame from GRanges to use arrange function
  as.data.frame(Binding_sites_inside) %>%
  # sort by PureCLIP score
  arrange(desc(score)) %>%
  # subset for top 500 (first 500 rows)
  .[1:500,] %>%
  # turn back into GRanges object
  makeGRangesFromDataFrame(keep.extra.columns = T) %>%
  # enlarge GRanges by 20 nt on both sites
  + 20
```

Next, we need to get the sequence underlying our extended binding site windows. Sequence information of many genomes is stored in BSgenome packages from Bioconductor. We can use the getSeq() function to get the sequences corresponding to a GRanges object

```
# load sequences of genome
genome <- BSgenome.Hsapiens.UCSC.hg19 # use GRCH38 for other RBPs

# test getSeq
Binding_sites_inside[1:3]
getSeq(genome, Binding_sites_inside[1:3])

# get sequences for windows around binding sites
Seq_Binding_site_window_for_motif <- getSeq(genome, Binding_site_window_for_motif)
Seq_Binding_site_window_for_motif

# set name for each sequence in the fasta file
names(Seq_Binding_site_window_for_motif) <- c(seq(1:length(Seq_Binding_site_window_for_motif)))
```

```
# path to save fasta file
fasta_output_file <- "/path/to/folder/BS_window_40nt_<RBP>.fasta"
```

```
# export as fasta file
writeXStringSet(Seq_Binding_site_window_for_motif, fasta_output_file)
```

1.1.2 MEME suite

Go to the MEME suite at http://meme-suite.org/tools/meme. For a simple analysis choose the following settings:

- Motif discovery mode: classical mode
- Sequence alphabet: DNA, RNA or Protein
- Input primary sequence: upload the fasta file from above here
- Site distribution: Zero or one Occurence (We expect that a binding site contains one binding motif)
- How many motifs should MEME find?: choose something between 5-15; more motifs will need a longer calculation time, but it can be interesting to see how many significant motifs MEME finds
- Click advanced options
- How wide can motifs be?: you can set something like 5 to 20; you can play around with this parameter
- Can motif sites be on both strands? click search given strand only! (These are RNA sequences that do not have a second strand)
- Start search and wait (This will take a while; you can already open a second tab to start DREME)

1.1.3 DREME suite

Go to the MEME suite at http://meme-suite.org/tools/dreme. The settings are similar as for MEME:

- For control sequences use: Shuffled input sequences (for User-provided sequences we could make a second fasta with random sequences)
- Sequence alphabet: DNA, RNA or Protein
- Input primary sequence: upload the fasta file from above here
- Again go to advanced and choose “Search given strand only”
- Start search (again this may take a while; we can go on with the next analysis and look at the results later; !do not close the browser!)

1.2 Functional annotation of bound genes

In order to learn more about the RBP of interest, we can look at the functional annotations of the bound transcripts. Several databases are available that provide a comprehensive description of the known or inferred functions of the protein products of genes. Based on functional annotations for all genes, a list of genes can be tested for enrichment of specific functions. To this end, for each annotation term, the fraction of genes from the list that are associated with this term is compared to its overall occurrence to identify terms that are significantly over-represented. Significance is often calculated using a p-value from a hypergeometric test. It is important to note that the "enrichment" can be strongly influenced by the choice of the baseline, i.e. whether enrichment is tested against all genes in the genome or a specific set of control genes.

The Bioconductor package HypeR stores the information of different databases with functional annotations for several organisms:

```
# hypeR provides different annotations for several organisms
msigdb_info()
```

From this, we can load for example the REACTOME pathway information:

```
# get pathway info from REACTOME
reactome_geneset <- msigdb_gsets("Homo sapiens", "C2", "CP:REACTOME")
```

Now we want to know how many of the genes that were bound by our RBP belong to which of the REACTOME pathways. For this, we make a list of the names of all bound genes and test for an enrichment of pathways:

```
# load bound genes from Day 2
load(hitgenes_out_file)

# get names of bound genes
bound_gene_names <- hit_genes$gene_name

# test enrichment of genes in Reactome pathways
reactome_hyp <- hypeR(bound_gene_names, reactome_geneset, test="hypergeometric", fdr=0.05)

reactome_hyp

# generate dot plot
reactome_dot <- hyp_dots(reactome_hyp)+
  ggtitle("REACTOME pathways")

reactome_dot
```

Exercise: GO terms Another commonly used resource is Gene Ontology (GO) annotation (http://geneontology.org/). GO stores three types of information, which describe a protein's physiological role ("Biological Process"), its molecular activity ("Molecular Function") and its position within the cell ("Cellular Component"). All three types of GO annotation are availabe from hypeR (See msigdb_info()). Make the corresponding dotplots for these annotation types.

1.3 Overlap binding sites of two RBPs

One way to compare the binding of two RBPs is to test for overlapping binding sites. To this end, we load the .RData objects with the binding sites for two different RBPs. As we used the same code for defining the binding sites for both RBPs, the name of the binding site objects is identical. You should therefore rename the binding site object for the first RBP prior to loading the binding sites from the second RBP of your group mates.

```
# store binding sites of your RBP in a new variable to avoid mixups
RBP1_binding_sites <- Binding_sites_inside
rm(Binding_sites_inside)
```

Get the final annotated binding sites for the second RBP from your group mates, and also load them into your R session.

```
# load second RBP from your group
path_RBP2_binding_sites <- "/path/to/binding/sites/to/compare"
```

```
# load and rename binding sites of RBP to compare
# here: SRSF6
load(path_RBP2_binding_sites)
RBP2_binding_sites <- Binding_sites_inside
```

In order to search for overlaps between binding sites, we can use the subsetByOverlaps() function from the GenomicRanges packages. From the resulting overlaps object, you can then extract the number of overlapping binding sites and put it into relation to the total number of binding sites from both RBPs.

```
# find overlapping binding sites of both RBPs
overlaps <- subsetByOverlaps(RBP1_binding_sites, RBP2_binding_sites)

# get number of overlapping binding sites
n_overlaps <- NROW(overlaps)

# get total number of binding sites for both RBPs
n_RBP1 <- NROW(RBP1_binding_sites)
n_RBP2 <- NROW(RBP2_binding_sites)
```

To visualise the comparison, you can make a Venn diagram with the eulerr package. You will have to first install and load the package.

```
#install.packages("eulerr")
library(eulerr)

# combine numbers
combined_numbers <-  c(RBP1 = n_RBP1-n_overlaps, RBP2 = n_RBP2-n_overlaps,
                     "RBP1&RBP2" = n_overlaps)
fit <- euler(combined_numbers,  shape = "ellipse")
plot(fit, quantities = TRUE)
```

Exercise: Look for binding sites within a certain distance from each other

In the analysis above, we only found a very small overlap. But would we really expect the RBPs to bind in the same place? More likely they might bind near each other. The subsetByOverlaps() function has a parameter called maxgap which allows the set the maximum gaps size between regions to still be called as overlapping. With this, you can find binding sites near each other. For example subsetByOverlaps(RBP1_binding_sites, RBP2_binding_sites, maxgap=10) selects all binding sites that have less than 10 nt between them. Try out different gap sizes and try to work out what works and what makes sense.

Exercise: Change order in subsetByOverlaps

We can change the order of the two RBP binding site sets in subsetByOverlaps(RBP2_binding_sites, RBP1_binding_sites). Try it out. Why do you get different values than with subsetByOverlaps(RBP1_binding_sites, RBP2_binding_sites)?

Exercise if you still have time: Overlap on gene level

Instead of directly overlapping binding sites, you can also compare if the same genes were bound. For this, you have to load the file with bound genes from Day 2. You can subset like this:

Make a Venn diagramm from this comparison.