iCLIP data analysis - Day 1 (U2AF65)

Melina Klostermann & Kathi Zarnack

24 August, 2020

Contents

1 Overview 2

2 UV crosslinking and immunoprecipitation (CLIP) 2

3 Basic iCLIP data processing 3

4 Exploring the crosslink profile of your RBP 3

4.1 Look at your data in IGV 3

5 Group projects: Identify binding sites of two RBPs and compare their binding behaviour 5

5.1 Data sets 5

6 Explore your data sets 5

6.1 Look at the data in IGV 5

6.2 Load the bigWig files in R 5

6.3 Chromosome names 6

6.4 How many crosslink events occurred at how many sites in the genome? 7

6.5 Histogram of crosslink events per crosslink site 8

6.6 Merge bigWig files 9

7 Peak calling with PureCLIP 10

7.1 Executing PureCLIP on your iCLIP data 10

7.2 Load PureCLIP sites into R as GRanges object 10

8 Define binding sites from PureCLIP sites 11

9 Save binding sites for tomorrow 14

References 14

1 Overview

Day 1:

- Explore crosslinks of iCLIP experiments
- Peak calling and binding site definition

Day2:

- Reproducibility between replicates
- Bound gene types and transcript regions

Day3:

- Prediction of binding motifs
- RNAmaps
- Gene Ontology (GO) analysis

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

2 UV crosslinking and immunoprecipitation (CLIP)

UV crosslinking and immunoprecipitation (CLIP) experiments offer a snapshot of protein-RNA interactions throughout the transcriptome. At the basis of all CLIP protocols, UV irradiation is used to covalently crosslink protein-RNA complexes in living cells. After purification of the crosslinked protein-RNA complexes, the co-purified RNA fragments are converted into specialized cDNA libraries and subjected to next-generation sequencing (NGS). CLIP protocols exist in different flavours, with the most commonly used variants being PAR-CLIP (photoactivatable ribonucleoside-enhanced CLIP), iCLIP (individual-nucleotide resolution CLIP) and eCLIP (enhanced CLIP).

An overview of the iCLIP protocol can be found in Figure 1.

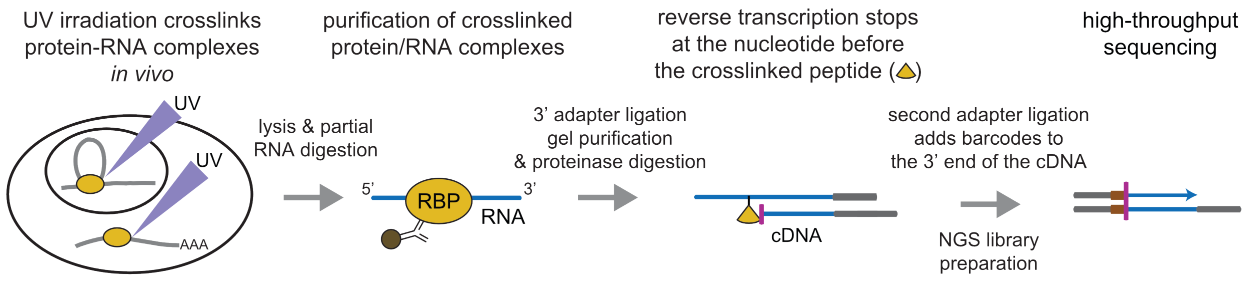
UV irradiation crosslinks protein-RNA complexes *in vivo*
purification of crosslinked protein/RNA complexes
reverse transcription stops at the nucleotide before the crosslinked peptide (▲)
high-throughput sequencing
UV
UV
AAA
lysis & partial RNA digestion
5'
RBP
3'
RNA
3' adapter ligation gel purification & proteinase digestion
cDNA
second adapter ligation adds barcodes to the 3' end of the cDNA
NGS library preparation

Figure 1: Overview of major steps of the iCLIP protocol. Image taken from (Busch et al. 2019): A complete pipeline from sequencing reads to RBP binding sites For more details on the experimental procedures, please take a look at (Huppertz et al. 2014) & (Buchbender et al. 2019))

3 Basic iCLIP data processing

The sequencing reads from an iCLIP experiment undergo a series of processing steps which are common to many NGS-based data. These include an initial quality control, specific adapter trimming, genomic mapping and postprocessing of the mapped reads.

An outline of the workflow can be found in Figure 2:

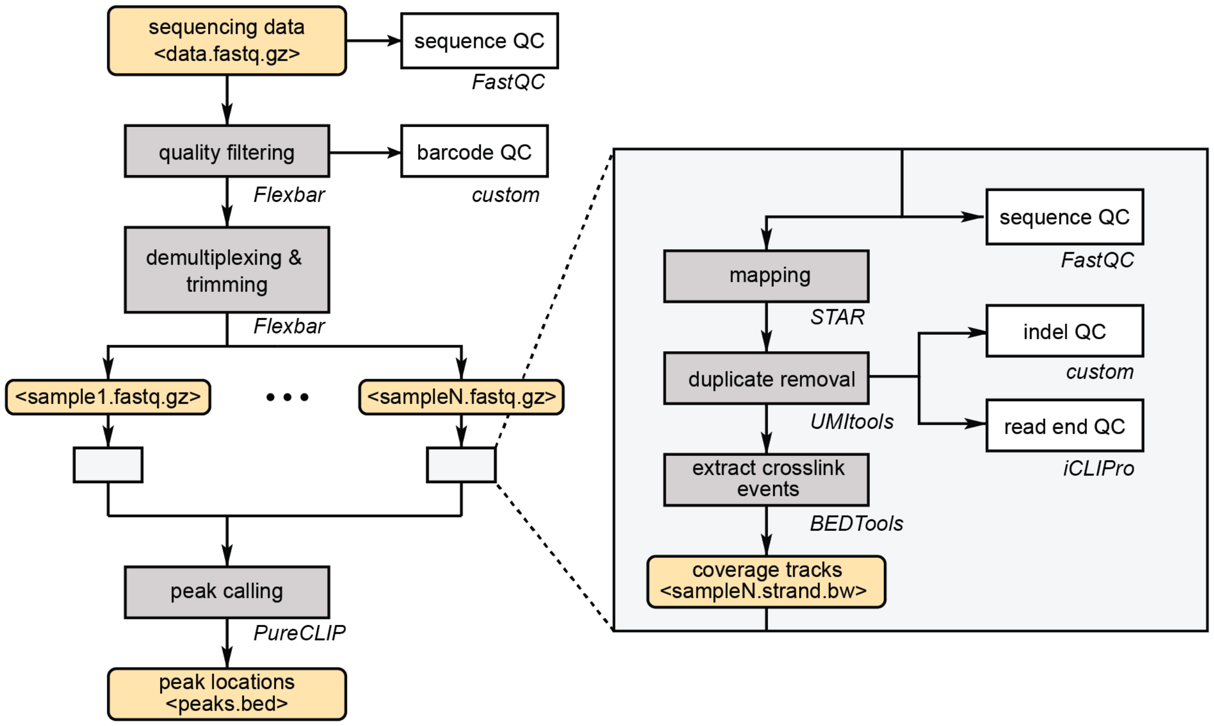

Figure 2: Major steps of the iCLIP data processing pipeline, including intermediate files and tools used. Image taken from (Busch et al. 2019)

4 Exploring the crosslink profile of your RBP

4.1 Look at your data in IGV

At the end of the basic iCLIP data processing, you receive bigWig files (.bw) which contain the genomic coordinates of RBP crosslink sites and the number of crosslink events at each crosslink site. We can look at the crosslink events in the Integrative Genomics Viewer (IGV): http://software.broadinstitute.org/software/igv/

Select the genome version

During the alignment step with STAR, the iCLIP reads were mapped to the genome of the organism that was used for the iCLIP experiment. Note that for most organisms, different versions of the versions of the genome assembly are available, such as hg19 (GRCh27, February 2009) or hg38 (GRCh38, December 2013) for the human genome. In IGV, you can select the genome assembly that corresponds to the mapping of your data in the top left corner (for more options, select "More. . . ". If your organism or version are not available, you can upload your own genome sequence and annotation.

Load a file into IGV

You can load your bigWig files into IGV via "File/Load from File. . . ".

Important: Turn off windowing function

By default, IGV uses a windowing function to smooth the displayed data. However, this can skew the signal. To turn off this function, right-click on the name of the bigWig track in the left column of IGV and select "None" under "Windowing Function". You can do this for multiple tracks at the same time.

Customize track appearance

- By right-click on the track name, you can also change the name/height/color/etc. of the track.
- Very useful is the option "Autoscale". When Autoscale is selected, IGV will use the highest signal in the viewed region (= the highest number of crosslink events at a given position) as the maximum for the y-axis of the track. In each track, the displayed data range is shown in the top left corner eg., [0-130]. The related option "Group Autoscale" chooses the best data range for group of samples that are selected together.

Gene annotation

For orientation, you can see a gene annotation track below your data tracks. By default, IGV loads RefSeq annotation for most genomes (https://www.ncbi.nlm.nih.gov/refseq/), but you can also upload other annotation file taken eg., from ENSEMBL or GENCODE.

If multiple isoforms exist for a gene, these can be merged into one overlaid track ("Collapsed") or displayed separately ("Expanded"; to be selected by right-click). Small arrows in the introns and exons indicate whether a gene lies on the plus or minus strand.

Move through the data

You can zoom into a region by - using the + and - button on the top right, - click and drag in the upmost row (above your first data track) to mark the region you want to see.

You can move to the left or right by clicking on any data track and dragging the mouse right or left. Note that if you use Autoscale, IGV will automatically change the scales when you move around, so it is worth to check the scale in the left top corner once in a while.

Select a location or gene of interest

If you want to look at a certain location, type the genomic coordinates into the box at the top (eg chr8:51,783,987-51,869,598). The same box can be used to directly search for a gene name (eg., MALAT1, PAPBC1, . . .) or sometimes other information such as ENCODE gene IDs (eg., ENSG00000251562).

Genomic sequence

If you zoom in closely, IGV will display the genomic sequence in a separate Sequence track. The arrow on the left indicates the strand which is displayed. By default, the sequence is shown for the plus strand, to be read from left to right. If the gene of interest lies on the minus strand, you can click on the arrow to display the complementary minus strand, to be read from right to left.

Screenshots

If you find something interesting or strange while scrolling through your data, it is helpfull to directly make a screenshot to easlily find it again later. You can do this via "File/Save Image".

Save IGV session

You can save your IGV session to look at the same data again later by "File/Save Session". This also saves all setting regarding appearance, regions of interest, etc. Next time use "File/Open Session"" instead of importing the bigWig files again.

Exercise: Binding patterns of different RBPs

Download the example bigWig files for the different RBPs. Load the files into IGV and explore the binding patterns of the different RBPs. Try to guess what the function of the different RBPs might be (Hint: Where in the genes do they bind?). Please collect screenshots of what you find. We will go through the pictures for each RBP together in the end and resolve their function and identity.

5 Group projects: Identify binding sites of two RBPs and compare their binding behaviour

5.1 Data sets

Group 1: U2AF65 + SRSF6

Group 2: MKRN1 + PABPC4

6 Explore your data sets

6.1 Look at the data in IGV

The iCLIP experiments were performed in several biological replicates. For your RBP, you should have bigWig files for the separate replicates as well as for the merged data from all replicates. Load all files in IGV and compare the patterns.

How does the merging of replicates influence the signal? How does it influence background noise?

6.2 Load the bigWig files in R

To analyze the iCLIP data in more detail, load the bigWig files of your project into R using Rstudio.

First enter the path to your files:

Now you can import the bigWig files. As the bigWig format does not encode whether the crosslink events occurred on the plus or minus strand, we import plus and minus strand seperately and manually add the strand information.

```
########################
# import bigWig files (crosslink events)
########################

# import replicate 1
# import
bw_1_plus <- import.bw(bw_1_plus_path)
bw_1_minus <- import.bw(bw_1_minus_path)

# show imported files
bw_1_plus
bw_1_minus

# replicate 2
bw_2_plus <- import.bw(bw_2_plus_path)
bw_2_minus <- import.bw(bw_2_minus_path)

bw_2_plus
bw_2_minus

# replicate 3
bw_3_plus <- import.bw(bw_3_plus_path)
bw_3_minus <- import.bw(bw_3_minus_path)
```

```
bw_3_plus
bw_3_minus

# replicate 4

bw_4_plus <- import.bw(bw_4_plus_path)
bw_4_minus <- import.bw(bw_4_minus_path)

bw_4_plus
bw_4_minus
```

6.2.1 Include strand information

After the import, the strand column displays "*" for both strands. This mean that the GRanges object lacks information about the strand on which the crosslinks lie. We can change the strand to "+" or "-" with the strand(GRanges) function.

```
# strand column says "*", meaning that the strand is unkown

# add strand information to the GRanges object
strand(bw_1_plus) <- "+"
strand(bw_1_minus) <- "-"
# and have a look again:
bw_1_plus
bw_1_minus
# now the strands are annotated correctly with "+" or "-"
```

Exercise: Strand Info

Add the right strand to the crosslinks (bw) of the other replicates.

6.3 Chromosome names

The first column of a GRanges object specifies its seqnames, ie. the names of the chromosomes, which can be extracted with seqnames().

Exercise: Seqnames

- How many chromosomes are in the human/mouse genome?
- How many seqnames are present in the GRanges object of bw_1_plus?
- Can you explain the difference?

```
# look at seqnames
seqnames(bw_1_plus)
```

To simplify our analysis, we can choose to only work with crosslink events that were mapped to the standard chromosomes. For human, there are 25 standard chromosomes: chr1-chr22, chrX, chrY and chrM (= the mitochondrial chromosome). We can use the function keepStandardChromosome(GRanges, pruning.mode = "coarse"):

```
#######################
# remove additional scaffolds
#######################

#sample 1
bw_1_plus <- keepStandardChromosomes(bw_1_plus, pruning.mode = "coarse")
bw_1_minus <- keepStandardChromosomes(bw_1_minus, pruning.mode = "coarse")
```

```r
bw_1_plus
bw_1_minus

#sample 2
bw_2_plus <- keepStandardChromosomes(bw_2_plus, pruning.mode = "coarse")
bw_2_minus <- keepStandardChromosomes(bw_2_minus, pruning.mode = "coarse")

#sample 3
bw_3_plus <- keepStandardChromosomes(bw_3_plus, pruning.mode = "coarse")
bw_3_minus <- keepStandardChromosomes(bw_3_minus, pruning.mode = "coarse")

#sample 4
bw_4_plus <- keepStandardChromosomes(bw_4_plus, pruning.mode = "coarse")
bw_4_minus <- keepStandardChromosomes(bw_4_minus, pruning.mode = "coarse")
```

6.4 How many crosslink events occurred at how many sites in the genome?

We can collect a few simple metrics to get a first overview of the data: - We can count the number of crosslink sites. This is equal to the number of rows in the GRanges objected, obtained with length() or NROW(). - We can calculate the total number of crosslink events at all crosslink sites, by summing up all scores. - We can calculate the ratio of crosslink events / crosslink sites, ie. how many crosslink events on average happened at the same crosslink site. - With data.frame() we can make a table for all samples.

```r
# number of crosslink sites
NROW(bw_1_plus)

# number of crosslink events
sum(bw_1_plus$score)

# average crosslink events per site
sum(bw_1_plus$score)/NROW(bw_1_plus)

# make a table of crosslink sites and events of all replicates
crosslinks_overview_table <- data.frame(sample = c("sample 1 +", "sample 1 -",
                      "sample 2 +", "sample 2 -",
                      "sample 3 +", "sample 3 -",
                      "sample 4 +", "sample 4 -"
                      ),
           cl_sites = c(NROW(bw_1_plus), NROW(bw_1_minus),
                           NROW(bw_2_plus), NROW(bw_2_minus),
                           NROW(bw_3_plus), NROW(bw_3_minus),
                           NROW(bw_4_plus), NROW(bw_4_minus)
                           ),
           cl_events = c(sum(bw_1_plus$score), sum(bw_1_minus$score),
                         sum(bw_2_plus$score), sum(bw_2_minus$score),
                         sum(bw_3_plus$score), sum(bw_3_minus$score),
                         sum(bw_4_plus$score), sum(bw_4_minus$score)
           ))

crosslinks_overview_table

# calculate average events per location
crosslinks_overview_table$ratio <- with(crosslinks_overview_table, cl_events / cl_sites)
```

```
crosslinks_overview_table
```

6.5 Histogram of crosslink events per crosslink site

We can also display the information graphically as a histogram with ggplot and geom_hist.

```
#################
# make histograms of crosslink events
#################

# combine + and - strand of each sample to make one histogram per sample
bw_1 <- c(bw_1_plus, bw_1_minus)
bw_1

bw_2 <- c(bw_2_plus, bw_2_minus)
bw_3 <- c(bw_3_plus, bw_3_minus)
bw_4 <- c(bw_4_plus, bw_4_minus)

bw_1_df <- as.data.frame(bw_1)
bw_2_df <- as.data.frame(bw_2)
bw_3_df <- as.data.frame(bw_3)
bw_4_df <- as.data.frame(bw_4)

# make a histogram
ggplot(bw_1_df, aes(x=score))+
  geom_histogram(binwidth = 1) # with binwidth you can set the size of the bins
```

What do you see in this histogram?

The histogram displays the number of crosslink events (ie. score) per site on the x-axis and the number of sites with this number of crosslink events on the y-axis. When displaying the complete data range, we cannot see much because there are very few crosslink sites with a very high number of crosslink events. On the other hand, many sites harbor only 1 crosslink event.

```
# Maximum number of cl events in this sample:
max(bw_1_df$score)

# How many crosslink site with 1 crosslink event
sum(bw_1_df$score == 1)
```

Exercise: Zoom in

- Make a zoom-in into the histogram plot which only shows crosslink sites with up to 20 crosslink events, ie. only crosslink sites with a score between 1 and 20, using coord_catesian(). Use the R Cheatsheet to find out more.

```
# histogram with zoom into distribution between 1 and 20
hist_bw_1 <- ggplot(bw_1_df, aes(x=score))+
  geom_histogram(binwidth = 1)+
  coord_cartesian(xlim = c(0,20))

hist_bw_1
```

```
hist_bw_2 <- ggplot(bw_2_df, aes(x=score))+
  geom_histogram(binwidth = 1)+
  coord_cartesian(xlim = c(0,20))
```

```r
hist_bw_3 <- ggplot(bw_3_df, aes(x=score))+
  geom_histogram(binwidth = 1)+
  coord_cartesian(xlim = c(0,20))

hist_bw_4 <- ggplot(bw_4_df, aes(x=score))+
  geom_histogram(binwidth = 1)+
  coord_cartesian(xlim = c(0,20))

hist_bw_1
```

```r
hist_bw_2
```

```r
hist_bw_3
```

```r
hist_bw_4
```

- Compare the histograms of different samples. Which sample is the weakest? How big is the difference between samples?

6.6 Merge bigWig files

We will later need a merge of the bigWig files from all replicates. There are several ways to do this merge in bash. An R possibility uses the the unique_granges() function with the option sum.cols = "score", meaning that it sums up the scores at the same locations, from the package gintools. As this again takes some more computing power, we already merged the crosslinks for you. Here we are showing an exemplary code. (Do not run.)

```r
## make lists of what should be combined
# bw_all_plus <- c(bw_1_plus,bw_2_plus, bw_3_plus, bw_4_plus)
# bw_all_minus <- c(bw_1_minus,bw_2_minus, bw_3_minus, bw_4_minus)
#
## combine samples and add scores: unique_granges() from gintools package
# bw_all_plus <- unique_granges(bw_all_plus, sum.cols = "score")
# bw_all_minus <- unique_granges(bw_all_minus, sum.cols = "score")
#
## the merged GRanges can be exported again as a bigWig file
# path_to_save_plus <- "/path/to/your/folder/crosslinks_<RBP>_merged.plus.bw"
# path_to_save_minus <- "/path/to/your/folder/crosslinks_<RBP>_merged.minus.bw"
#
# export.bw(bw_all_plus, path_to_save_plus, fixedSummaries= TRUE)
# export.bw(bw_all_minus, path_to_save_minus, fixedSummaries= TRUE)
```

Exercise: Load merged bigWig files

Now load the merged bigWig files with the import.bw() function (same as you did before with the replicate data). Do not simply trust a program, but always check whether it did what you expected. IGV is a good option to check by eye what happend. Load the merged bigWig files in IGV together with the bigWig files of the individual samples that you looked at earlier. Did the merge work correctly?

7 Peak calling with PureCLIP

In IGV, we already saw that the crosslink events often assemble into peaks with a bell-like shape over a window of a few nucleotides. These peaks are likely to reflect "true" RBP binding events. However, we can also observe a considerable amount of background crosslink signal that often spreads across the complete transcripts. In order to reliably detect positions with significant crosslink signal that likely reflects RBP binding sites, we use the peak calling algorithm PureCLIP. PureCLIP is a command-line tool that trains a hidden Markov model based on the diagnostic truncation sites, ie. the crosslink sites, and the complete RBP-bound fragments.

7.1 Executing PureCLIP on your iCLIP data

PureCLIP requires three files as input: - a .bam file with the mapped reads from your iCLIP experiment your iCLIP experiment, - a .bam.bai file which is the index corresponding to the .bam file, and - a .fasta file with genome sequence which should be the same as used when aligning the reads with STAR.

In order to boost the sensitivity of peak detection, we commonly merge the replicates of an experiment for the peak calling step (and subsequently separate them again to assess reproducibility).

An example command to execute PureCLIP is shown below. A more detailed description can be found here: https://pureclip.readthedocs.io/en/latest/

Since peak calling with PureCLIP on a complete iCLIP dataset can take several days for calculation, we ran PureCLIP for you in advance. You therefor do not need to execute the PureCLIP command given below!

```
##########################
# peak calling with PureCLIP
#######################

#### ! do not run !!!
pureclip -i iCLIP_reads_<RBP>_merged.bam # bam file with the merged replicates of your experiments
-bai iCLIP_reads_<RBP>_merged.bam.bai # corresponding bai index
-o peakcalling_<RBP>_sites.bed # name of the output sites
-or peakcalling_<RBP>_regions.bed # a second type of output that we do not use here
-nt 10 # nodes for calculation
-iv 'chr1;chr2;chr3;' # chromosomes used to train the hidden markov model
                      # (more take longer to calulate, but might be more accurate;
                      # usually 3 chromosomes are sufficient)
```

The output of PureCLIP includes a .bed file with all nucleotides harbor a signficant number of crosslink events (peakcalling_*sites.bed). You can download the PureCLIP output file peakcalling_*sites.bed for your protein in this project from the course repository.

Exercise: PureCLIP sites in IGV

Load the PureCLIP output file peakcalling___sites.bed of your protein into IGV together with the bigWig files that you looked at earlier. Compare the crosslink signal in the bigWig files with the peak calling results. How did PureCLIP perform on this data set?

7.2 Load PureCLIP sites into R as GRanges object

Similar to the bigWig files before, you can load the PureCLIP output file peakcalling___sites.bed as a Granges object into R:

```
# set path to your file
pureclip_path <- "/path/to/your/folder/peakcalling_<RBP>_sites.bed.gz"
```

```
##########################
# load PureCLIP sites as GRanges
##########################

pureclip_sites <- import(pureclip_path, format = "bedgraph")
pureclip_sites

# note that that strand and the names of some metadata columns got lost during import
# restore
strand(pureclip_sites) <- pureclip_sites$NA.1
mcols(pureclip_sites) <- DataFrame(score = pureclip_sites$NA.)
pureclip_sites$round_score <- round(pureclip_sites$score, digits = 1)

pureclip_sites <- keepStandardChromosomes(pureclip_sites, pruning.mode = "coarse")
pureclip_sites
```

8 Define binding sites from PureCLIP sites

The significant sites called by PureCLIP are only 1-nt wide. However, we saw before in IGV that "true" binding sites show in a bell-shaped peak across several nucleotides. We would also expect that the binding interface of the RBP with the RNA is usually bigger than one nucleotide. For many applications, it is therefore advisable to postprocess the significant PureCLIP sites to obtain binding sites of a uniform width. The chosen width depends on the expected size of the RBP's footprint, which often relies on prior knowledge, such as the type of RNA binding domains. Estimates can also be deduced from the data, e.g. based on visual inspection of the RBP crosslink events in the genome browser.

Exercise: Try to estimate a good binding site width for your RBP from visual inspection in IGV

Look again at the PureCLIP sites and bigWig files in IGV. How wide would you set the binding sites?

In order to define these binding sites, we will perform the following steps:

- Starting from the PureCLIP output, the significant crosslink sites are first clustered into regions. The maximum distance between crosslink sites to be clustered together is chosen such that no binding sites will overlap after resizing. Practically, in our example, this means that when crosslink sites will later be extended by 4 nt to either side to obtain 9-nt binding sites, clustering together all crosslink sites with a distance up to 8 nt ensures that binding sites after resizing can touch but not overlap.

```
# merge significant crosslink sites over distances < 8 nt
pureclip = reduce(pureclip_sites, min.gapwidth = 8)
```

- Next, all resulting regions shorter than 3 nt are removed with the intention to focus on binding sites with substantial signal over a minimum width.

```
# remove sites with 1 or 2 nt length
pureclip = pureclip[width(pureclip) > 2]
```

- Now we use a matrix of all PureCLIP sites. We select the highest site in the matrix and span a window of chosen size (eg., 7 nt) around this highest site (the highest site will be at the center of this newly made binding site). We will then cut out this newly made binding site from the matrix, and restart the same process for the second highest peak in the matrix (which is now the highest).

For this approach, we make use of R's Run-length encoding (Rle):

```
# here is an example coverage vector crosslink events on subsequent positions
example_crosslinks = c(rep(0,5), rep(1,3), 0, 1, 2, rep(1,3), rep(0,4), 1, 0)
```

```r
example_crosslinks
# this is the same vector in Rle encoding
Rle(example_crosslinks)

# you can obtain Rle objects for our crosslink events by applying coverage()
coverage(bw_1_plus)
```

```r
# set path to bigWig files of merged replicates
bw_plus_path <- "/path/to/your/folder/crosslinks_<RBP>_merged.plus.bw"
bw_minus_path <- "/path/to/your/folder/crosslinks_<RBP>_merged.minus.bw"
```

```r
# first we have to decide what size our binding sites should have
# as GRanges can easily be resized symetrically on both sites with +n,
# we specify the half window size
# eg. window.radius <- 3 will result in
# 3 nt downstream + 1 nt center + 3 nt upstream = a 7-nt binding site
window.size <- 3


define_binding_sites_from_matrix <- function (bw_path, pureclip, window.size){

  # get crosslinks as Rle
  bw <- import.bw(bw_path, as="Rle")

  # we initalise two GRanges objects
  # 1. a GRanges objects which will contain our binding sites in the end (final BS)
  final.BS.gr <- GRanges()

  # 2. a GRAnges with the regions that are not yet resized (remaining regions)
  # as in the beginning nothing is resized, we put all PureCLIP sites into remaining regions
  remaining.regions.gr <- pureclip

  ########################
  # loop over a matix of all PureCLIP calls to define binding sites
  ########################

  while(length(remaining.regions.gr) != 0){
    # get the raw CL counts in the remaining PureCLIP CL regions
    # return Rle list of all regions and turn it into matrix
    remaining.PureCLIP.CL.regions.m <- as.matrix(bw[remaining.regions.gr])

    # identify the center of the PureCLIP CL regions (position with max counts)
    # and store its indice
    # set NA to -Infinite
    remaining.PureCLIP.CL.regions.m[is.na(remaining.PureCLIP.CL.regions.m)] <- -Inf
    max.pos.indice <- max.col(remaining.PureCLIP.CL.regions.m, ties.method = "first")

    # create a peak region that is centered to the max position
    binding_site <- remaining.regions.gr
    start(binding_site) <- start(binding_site) + max.pos.indice - 1
    width(binding_site) <- 1
    binding_site <- binding_site + window.size
    # add the new binding site in the output GRanges
    final.BS.gr <- c(final.BS.gr, binding_site)

    # remove the peaks from the CL regions to search for additional peaks
    # excise additionally x nucleotides up and downstream
    binding_site_surround <- as(binding_site + window.size, "GRangesList")

    remaining.regions.gr <- unlist(psetdiff(remaining.regions.gr, binding_site_surround))
  }
  return(final.BS.gr)
}

# define binding sites with the function above
# binding sites on plus strand
binding_sites_plus <- define_binding_sites_from_matrix(bw_path = bw_plus_path,
    pureclip = pureclip[strand(pureclip)=="+"], window.size = window.size)
# binding sites on minus strand
binding_sites_minus <- define_binding_sites_from_matrix(bw_path = bw_minus_path,
    pureclip = pureclip[strand(pureclip)=="-"], window.size = window.size)

binding_sites_plus
binding_sites_minus

# you can see that the binding sites are not sorted by chromosome anymore
# sort order of chromosomes
binding_sites_plus <- sort(binding_sites_plus)
binding_sites_minus <- sort(binding_sites_minus)
binding_sites_plus

# merge binding sites from plus and minus strand
Binding_sites <- c(binding_sites_plus, binding_sites_minus)
```

```
# set path to output file
bs_out_file <- "/path/to/your/folder/binding_sites_<RBP>.bed"
```

```
# export binding sites for visualization in IGV
export(Binding_sites,
       con = bs_out_file,
       format = "bed")
```

- Finally, depending on the depth of the signal we can require a certain number of positions with crosslink signal within one binding site to assure sufficient support of the binding site.

```
###########################
# keep only binding sites with at least 2  crosslink sites
############################

# make a matrix of crosslinks within binding sites
# we use an Rle of the crosslink sites
bw_plus_rle <- import.bw(bw_plus_path, as="Rle")
bw_minus_rle <- import.bw(bw_minus_path, as="Rle")
# each row will be one binding site
Binding_sites_crosslink_matrix_plus <- as.matrix(bw_plus_rle[binding_sites_plus])
Binding_sites_crosslink_matrix_minus <- as.matrix(bw_minus_rle[binding_sites_minus])
```

```
head(Binding_sites_crosslink_matrix_plus)

# sum up positions with crosslinks per binding site
n_crosslink_per_site_plus <- apply(Binding_sites_crosslink_matrix_plus,
                                   1, function(x) sum(x > 0))
n_crosslink_per_site_minus <- apply(Binding_sites_crosslink_matrix_minus,
                                    1, function(x) sum(x > 0))

head(n_crosslink_per_site_plus)

# set a cut-off
Binding_sites_n_cl_plus <- binding_sites_plus[n_crosslink_per_site_plus >= 3]
Binding_sites_n_cl_minus <- binding_sites_minus[n_crosslink_per_site_minus >= 3]

Binding_sites_n_cl <- c(Binding_sites_n_cl_plus, Binding_sites_n_cl_minus)
```

Exercise: Crosslink positions in binding sites

Take a look at the binding pattern of your RBP in IGV to decide what would be a good number of crosslink positions per bindig site. Rerun the filtering step above with this setting. How many binding sites do you obtain? How many binding sites do you lose if you require one more crosslink position? How many do you gain when you require one less? Export the filtered binding sites and look at example in IGV where binding sites were filtered out.

9 Save binding sites for tomorrow

```
BS_out_path <- "/path/to/save/BS_<RBP>.RData"
```

```
save(Binding_sites_n_cl, file = BS_out_path)
```

References

Buchbender, Andreas, Holger Mutter, F X Reymond Sutandy, Nadine Körtel, Heike Hänel, Anke Busch, Stefanie Ebersberger, and Julian König. 2019. "Improved library preparation with the new iCLIP2 protocol." *Methods*. https://doi.org/10.1016/j.ymeth.2019.10.003.

Busch, Anke, Mirko Brüggemann, Stefanie Ebersberger, and Kathi Zarnack. 2019. "iCLIP data analysis: A complete pipeline from sequencing reads to RBP binding sites." *Methods*. https://doi.org/10.1016/j.ymeth.2019.11.008.

Huppertz, Ina, Jan Attig, Andrea D'Ambrogio, Laura E Easton, Christopher R Sibley, Yoichiro Sugimoto, Mojca Tajnik, Julian König, and Jernej Ule. 2014. "iCLIP: Protein–RNA interactions at nucleotide resolution." *Methods* 65 (3): 274–87. https://doi.org/10.1016/j.ymeth.2013.10.011.