smartPARE degradome-seq analysis refinements

smartPARE is an R package designed for sRNA cleavage confirmation based on degradome data. smartPARE utilizes deep sequencing convolutional neural networks (CNN) to segregate true and false predictions of sRNA cleavage data produced by any sRNA cleavage prediction tool.

Overview - smartPARE consists of the 3 following stages:

1. Generation of cleavage windows
2. Cleavage window training
3. Cleavage confirmations

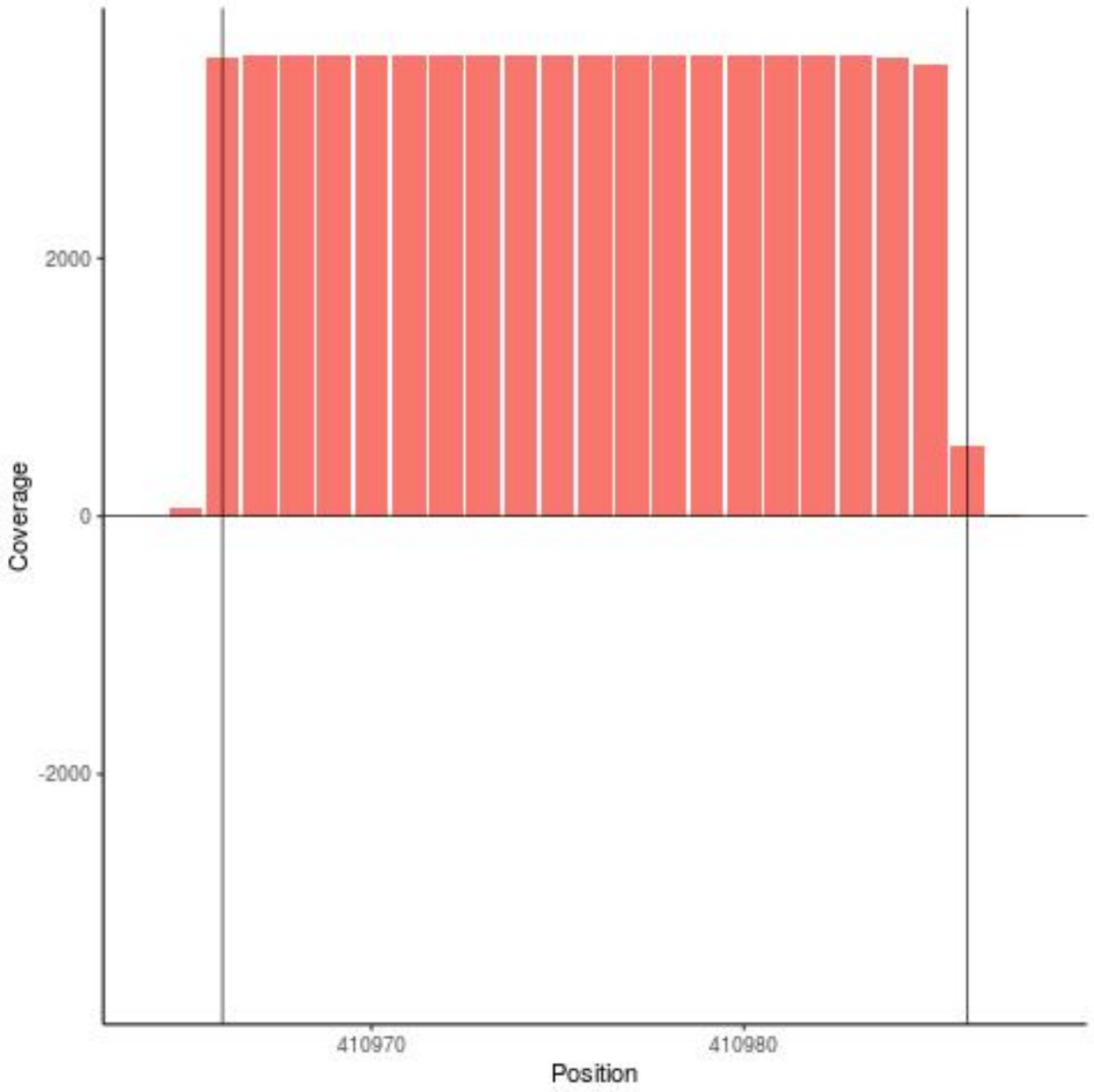
Installation

```
#Installation requires devtools
#install.packages("devtools")

#devtools::install_github('kristianHoden/smartPARE')
#Load the smartPARE R package
library(smartPARE)
```

Generation of cleavage windows

smartPARE identifies cleavages based on images of the sRNA cleavages sites (windows). The windows are created from degradome sequencing data transcript-aligned bam-files. The standard length of dagradome-seq reads are 20 nt. Hence, the default windows are generated from an area 1 nucleotides (nt) upstream the cleavage site to 21 nt downstream. The narrow mariginal to the proposed cleavage site reads to exclude as much noise as possible but still include the characteristic cleavage block (seen in the following image).




```
smartPARE_cleavageWindows(dirO = "pathOut/dir_smart"),
cleavageData = cleavageDataDataset,
aliFilesPath = "path/bamTranscriptome/",
aliFilesPattern1 = "pattern1.sorted.bam$",
aliFilesPattern2 = "pattern2.sorted.bam$",
ylim1 = 5,
edgesExtend1 = c(1,21),
gffTrans = gffTrans)
```

Parameters:

`gffTrans` - a dataframe of your imported gff with V10 = transcript ID. See for example extendGffTrans for example

`aliFilesPattern1` - regular expression defining the pattern of the first degradome bam file

`aliFilesPattern2` - regular expression defining the pattern of the second bam file

`dir0` - output path. Please end the path with _extension to make the dirs identifiable by smartPARE.

`cleavageData` - a dataframe containing at least columns genesT (the target genes) and posT (target position in the transcript)

`edgesExtend1` - Nt positions in relation to the cleavage site that are included in the images. Witten in the format: `c(upstream,downstream)`. Default is c(1,21).

`ylim1` - The minimum plotted height on the y-axis. In practise this means that a lower "cleavage block" might not be recognized by the CNN. This to exclude potential false positives caused by noice.

`aliFilesPath` - path to the bam files of the user defined degradome.

Output:

Images displaying in the output dir `dir0` the read coverage surrounding the cleavage site labeled `transcriptID_cleavagePosition`

Cleavage window training

This stage is only necessary if deciding to create a new CNN model. Our pre-trained model is found in ../data/model/

Preparation:

- Create a directory called train with three subdirs called goodUp, goodDown and bad.
- Collect manually identified images of true cleavages on the 5' strand in the goodUp subdir.
- Collect manually identified images of true cleavages on the 3' strand in the goodDown subdir.
- Collect manually identified images of false cleavages in the the bad subdir.
- **N.B.** Ensure to get as great variation as possible among the images of each subdir.

```
modelData <- smartPARE_train(homePath1 = "example/",
                             pixels1 = 28,
                             search_bound = list(denseLoop = c(0,4),
                                                 epochs2 = c(100, 300),
                                                 batch_size2 = c(32,128),
                                                 dropout2 = c(0, 0.3),
                                                 validation_split2 = c(0.1,0.4),
                                                 convolutionalLoop = c(1,4),
                                                 NO_pooling2 = c(1,2)),
                             n_iter = 100)
```

Parameters:

`homePath1` Path to the root directory of the train directory metioned above.

`pixels1` Number of pixels each each image will be converted to `search_bound` List of min and max values for the following variables: denseLoop2, epochs2, batch_size2, validation_split2, convolutionalLoop2 and NO_pooling2

`n_iter` Number of iterations the Bayesian optimization will run, each run creating a model

Output:

`modelData` contains a list with the following 4 entries:

1. `$Best_Par` shows the hyperparameters for the best performing model.
2. `$Best_Value` displays the score of the best model. The score is based on the inverted loss of the model.
3. `$History` the history of hyperparameters for all the created models.
4. `$Pred` the inverted loss of the cross-validated data.

Each created model is witten into "homePath1/bayesmodels/", where a subdir ("pdf") with pdfs displaying the performance of each model also is found. The model names in the directory is based on the iteration order followed by then accuracy (0-1), then loss, then the value of each hyperparameter. This makes it possible to identify the performance and setting of the model even if the `modelData` info is lost.

Cleavage confirmations

Preparation Gather images that you want to evaluate in subdirs to your wd (homePath1) at a certain level of recursion. Please end the dirs with _extension e.g. example_smart. The underline will be identified by smartPARE indicate that these dirs contains cleavage images. In an interaction study you might for example have cleavages from 2 different species gathered in different subdirs (specie1 & specie2). Maybe you also have different timepoints for each specie (tp1 & tp2). You might then gather your images in homePath1/specie1/tp1/ etc. The level of recursion will then be 3.

Load the preferred model according to one of the following:

If you designed your own model

```
model <- keras::load_model_hdf5("example/bayesmodels/modelNumberAndContinousName.h5")
```

If you are using our model

```
model <- keras::load_model_hdf5("data/model/CNNmodel.h5")
```

View the loaded model

```
‘model %>% summary()‘
```

Define the directories you want to examine

```
extDirs <- unique(dirname(list.files(homePath1,rec=T)))
```

Exclude the training dirs (if they are in the same path)

```
extDirs[-which(startsWith(prefix = "train",extDirs))]
extDirs <- extDirs[-which(startsWith(prefix = "train",extDirs))]
rootExt <- paste0(homePath1,extDirs)
```

Examine the cleavages images in rootExt dirs. `pixels` must be the same as when creating the model. Our model was based on 28 pixels.

```
smartPARE_examineCleavages(examinePath = rootExt, model = model,pixels = 28)
```

Construct a list of the true cleavages, `recursionLevel` counts the homePath1 level as level 0. the first subdir level is level 1 etc. Example images are found in ‘example/evaluate_smart/".

```
smartPARE_ListTrue(pathToTrue = homePath1,recursionLevel = 0)
```

Output

Output will be written to your homePath into a dir called "subdir"_results, e.g. example_results

Extract the information about the true cleavage windows from your results file

```
trueCleavagesDf <- smartPARE_parse(smartPAREresultFile = paste0(homePath1,"example_results/true.txt"))
```

Output

`trueCleavagesDf` contains the following four columns that can be used to filter out false cleavage sites in the dataset used for generating the cleavage images:

- ds, (dataset) based on the name of the dir the cleavage images were stored in
- posT, cleavage position in the transcript
- ext, the extension of the cleavage images

- transcript, transcript ID

ds	posT	ext	transcript
evaluate	987	R1.jpg	PGSC0003DMG400002426
evaluate	987	R2.jpg	PGSC0003DMG400002426
evaluate	988	R1.jpg	PGSC0003DMG400002426
evaluate	1072	R2.jpg	PGSC0003DMT400006358
evaluate	1363	R2.jpg	PGSC0003DMT400006361