Standard Analysis

Robert A Policastro

A transcription Start Site (TSS) represents the position of the first base of a transcript to be transcribed by an RNA polymerase. Most genes do not have a single TSS, but rather a collection of TSS positions collectively referred to as a Transcription Start Region (TSR). Variation in TSS selection influences transcript stability and translation as the function of the encoded protein. Furthermore, variations in TSS usage have been observed during organismal development as well as in diseases such as cancer.

TSRexploreR offers a series of analysis and plotting functions that allow deep exploration of TSSs and TSRs. Here, we describe the standard TSRexploreR analysis pipeline, qq

Preparing Data

This example uses a set of *S. cerevisiae* TSSs identified with the STRIPE-seq method. There are many ways to import TSSs into TSRexploreR; this vignette uses a named list of GRanges as input into the function that creates the TSRexploreR object.

```
library("TSRexploreR")

# Example TSSs.
TSSs <- system.file("extdata", "S288C_TSSs.RDS", package="TSRexploreR")
TSSs <- readRDS(TSSs)

# Keep only the 3 WT samples for now.
TSSs <- names(TSSs) %>%
  stringr::str_detect("WT") %>%
  purrr::keep(TSSs, .)

# Genome assembly and annotation.
assembly <- system.file("extdata", "S288C_Assembly.fasta", package="TSRexploreR")
annotation <- system.file("extdata", "S288C_Annotation.gtf", package="TSRexploreR")

# Sample sheet.
sample_sheet <- data.frame(
  sample_name=sprintf("S288C_WT_%s", seq_len(3)),
  file_1=NA, file_2=NA,
  condition=rep("Untreated", 3)
)

# Create the TSRexploreR object.
exp <- tsr_explorer(
  TSSs,
  sample_sheet=sample_sheet,
  genome_annotation=annotation,
  genome_assembly=assembly
)
```

Initial TSS Processing

After the TSSs are loaded into the TSRexploreR object, a few processing steps are necessary in order to prepare the data for analysis.

Format TSSs

The first step is to convert the TSSs into a table that will facilitate downstream analysis.

```
exp <- format_counts(exp, data_type="tss")
```

Normalize TSSs

The next step is to normalize TSS counts using the Counts Per Million (CPM) approach. This step is optional - if the counts you input were already normalized, this step can be safely skipped.

```
exp <- normalize_counts(exp, data_type="tss")
```

TSS Annotation

After formatting counts and optionally CPM normalizing them, TSSs will be annotated relative to known genomic features. This function takes either the path and file name of a 'GTF' or 'GFF' file or a 'TxDb' package from bioconductor. The example below uses a 'GTF' file from Ensembl (R64-1-1 Release 99) and annotates each TSS to the beginning of the nearest transcript.

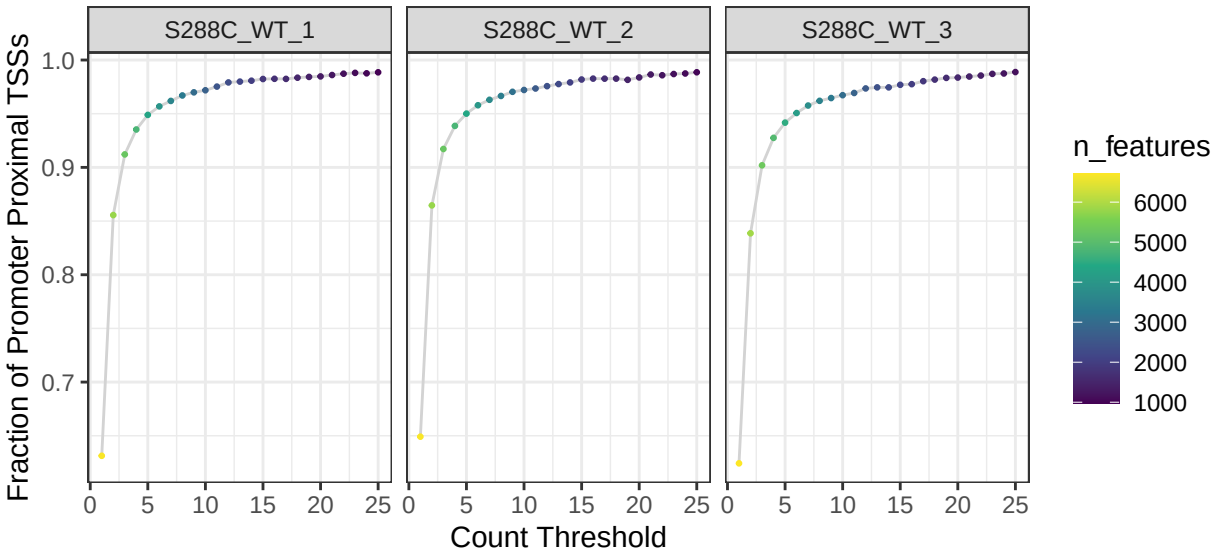
```
exp <- annotate_features(exp, data_type="tss", feature_type="transcript")
```

Naive Threshold Exploration

All TSS mapping methods contain some degree of background in the form of low-count TSSs, particularly within gene bodies. This background can complicate downstream analysis, but it can largely be removed via a simple thresholding approach wherein TSSs below a certain number of counts are removed from further consideration. If a genome annotation is available, a thresholding plot may help in picking this count threshold. In this approach, the fraction of TSSs that are promoter-proximal is plotted against a range of read thresholds. The number of features (genes or transcripts) with at least one unique promoter-proximal TSS position at each threshold is also indicated graphically.

```
threshold_data <- explore_thresholds(exp, steps=1)

plot_threshold_exploration(threshold_data, ncol=3, point_size=0.5) +
  scale_color_viridis_c()
```



Looking at the plot, a threshold of three or more reads is a sensible choice: there is a precipitous drop in detected features for a comparatively low gain in promoter-proximal TSS fraction at higher thresholds. TSSs will be discarded if no samples have at least 1 TSS at that position with three or more reads.

```
exp <- apply_threshold(exp, threshold=3)
```

TSS Correlation

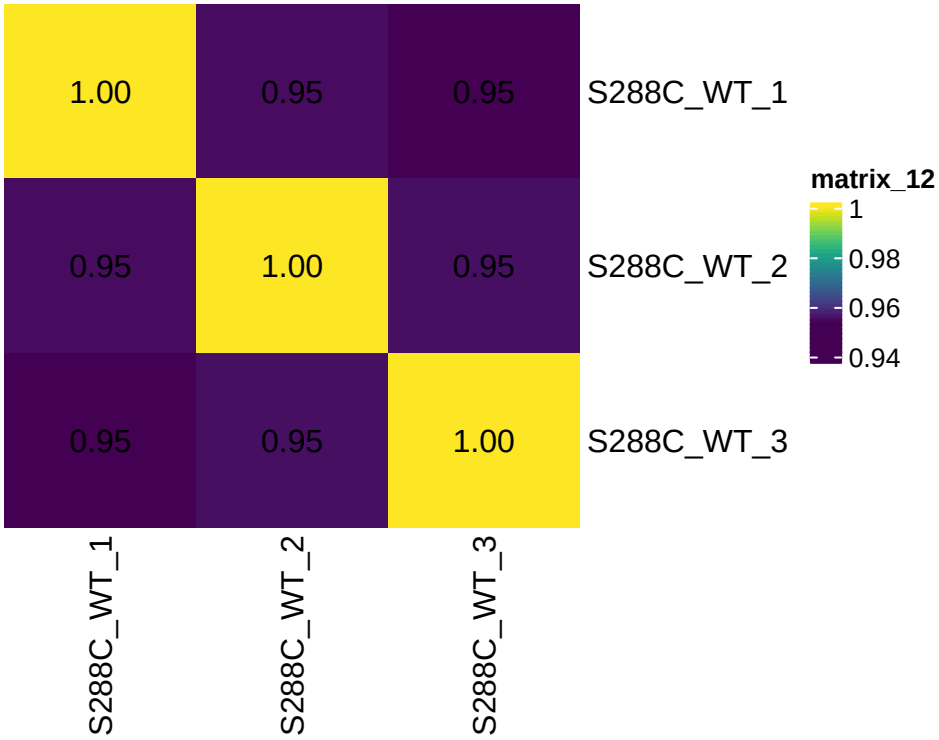
Assessing correlation between samples provides information on replicate similarity and can also give a cursory indication of the degree of difference between biologically distinct samples (e.g. different genotypes or treatments).

Correlation Matrix Plots

For correlation analysis, TSRexploreR uses TMM normalization via the edgeR package. This method was designed to make comparison **between** samples more efficacious. In the example below, TSSs are retained only if at least one sample has a count of 3 or more reads.

After TMM normalizing the samples, various correlation plots can be generated. In this example, a combined scatterplot/heatmap is generated.

```
plot_correlation(
  exp, data_type="tss", font_size=12,
  heatmap_colors=viridis::viridis(100)
)
```



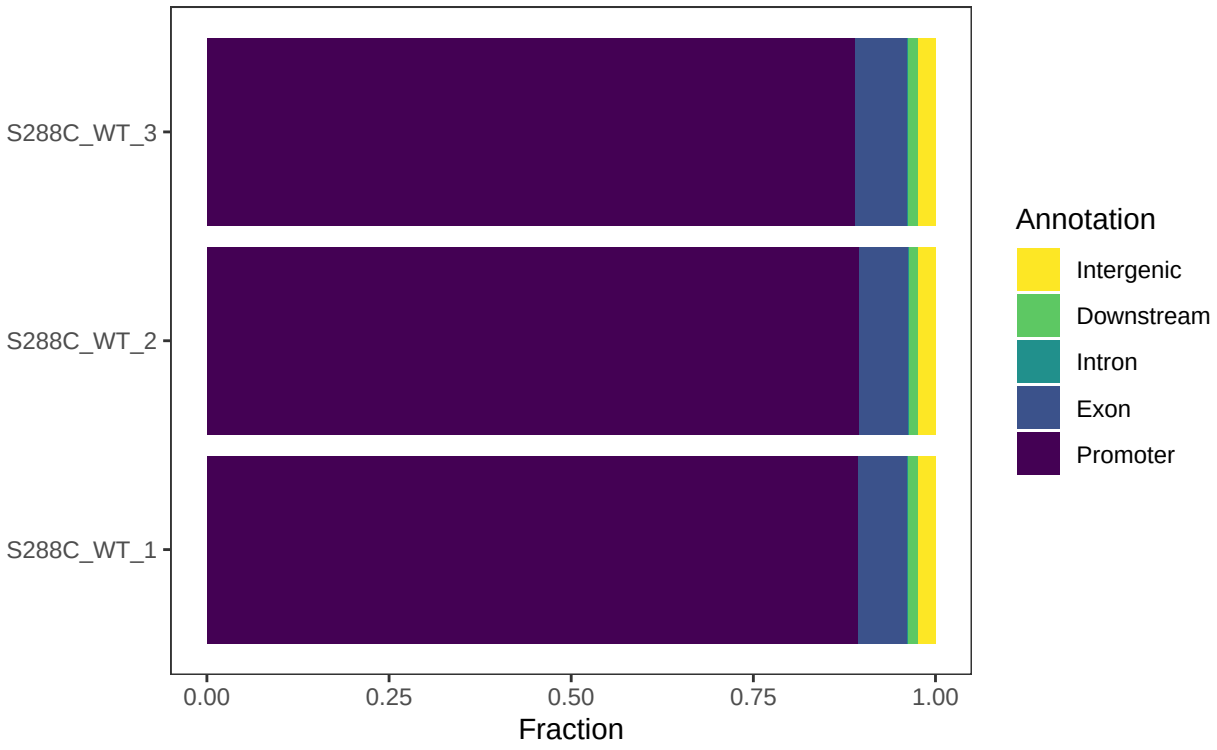
TSS Genomic Distribution

As part of the initial TSS processing, TSSs are annotated relative to known features. This information can be used to explore the genomic distribution of TSSs. Additionally, it can be used to plot the total number of detected features and the number of detected features with a promoter-proximal TSS.

Genomic Distribution Plot

A stacked bar plot can be generated to showcase the fractional distribution of TSSs relative to known features.

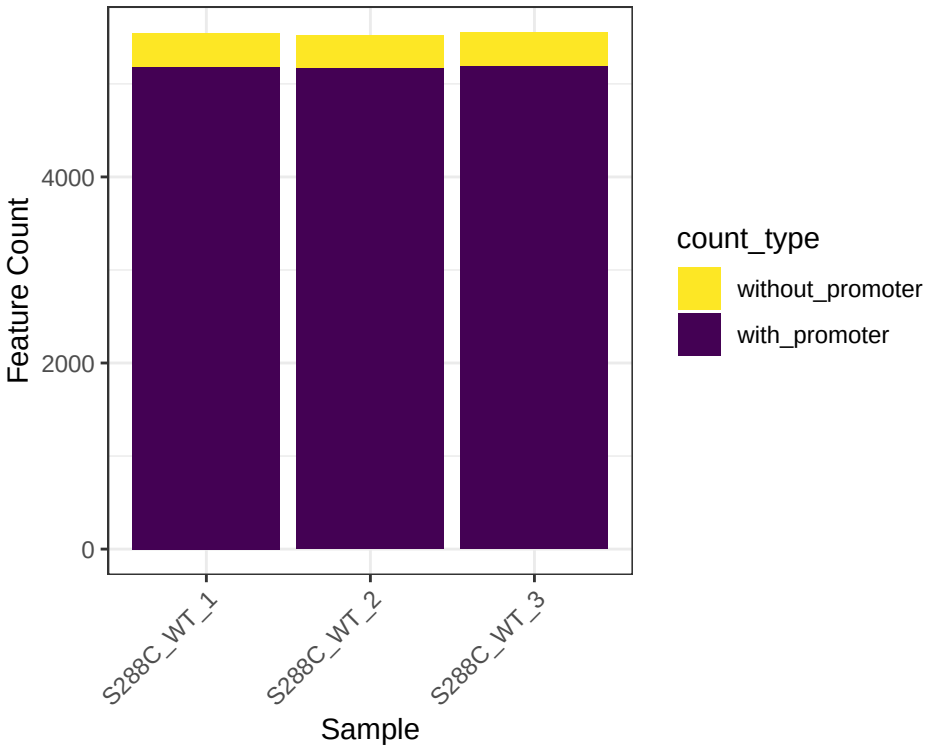
```
plot_genomic_distribution(exp, data_type="tss") +
  scale_fill_viridis_d(direction=-1, name="Annotation")
```



Feature Detection Plot

The number of features detected and fraction of features with a promoter-proximal TSSs can be displayed as a stacked bar plot or jitter plot.

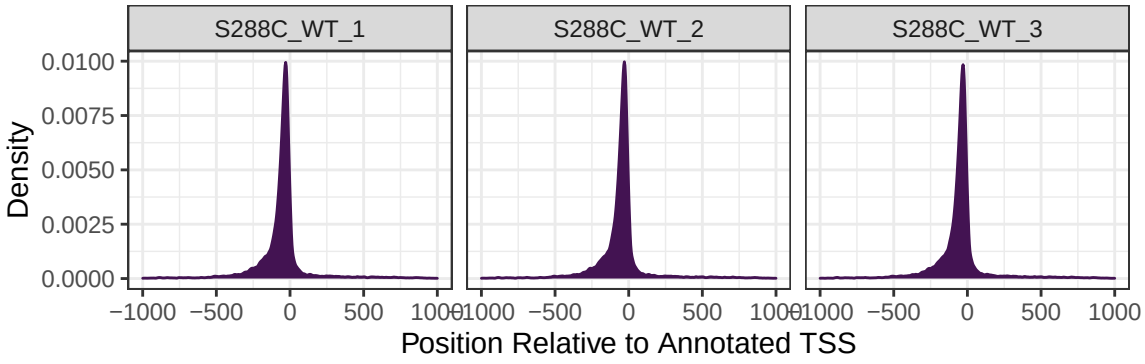
```
plot_detected_features(exp, data_type="tss") +
  scale_fill_viridis_d(direction=-1)
```



Density Plots

Density plots are useful for visualizing where TSSs tend to be located relative to annotated TSSs. The current yeast genome annotation does not contain information on 5' UTRs, so density plots are centered on annotated start codons and it is expected that the maximum density would be slightly upstream of the plot center. Most other organisms have 5' UTR information in their genome annotations, with the UTR length corresponding to the distance between the start codon and the most distal TSS detected. This would result in a centered density plot.

```
plot_density(exp, data_type="tss", ncol=3)
```

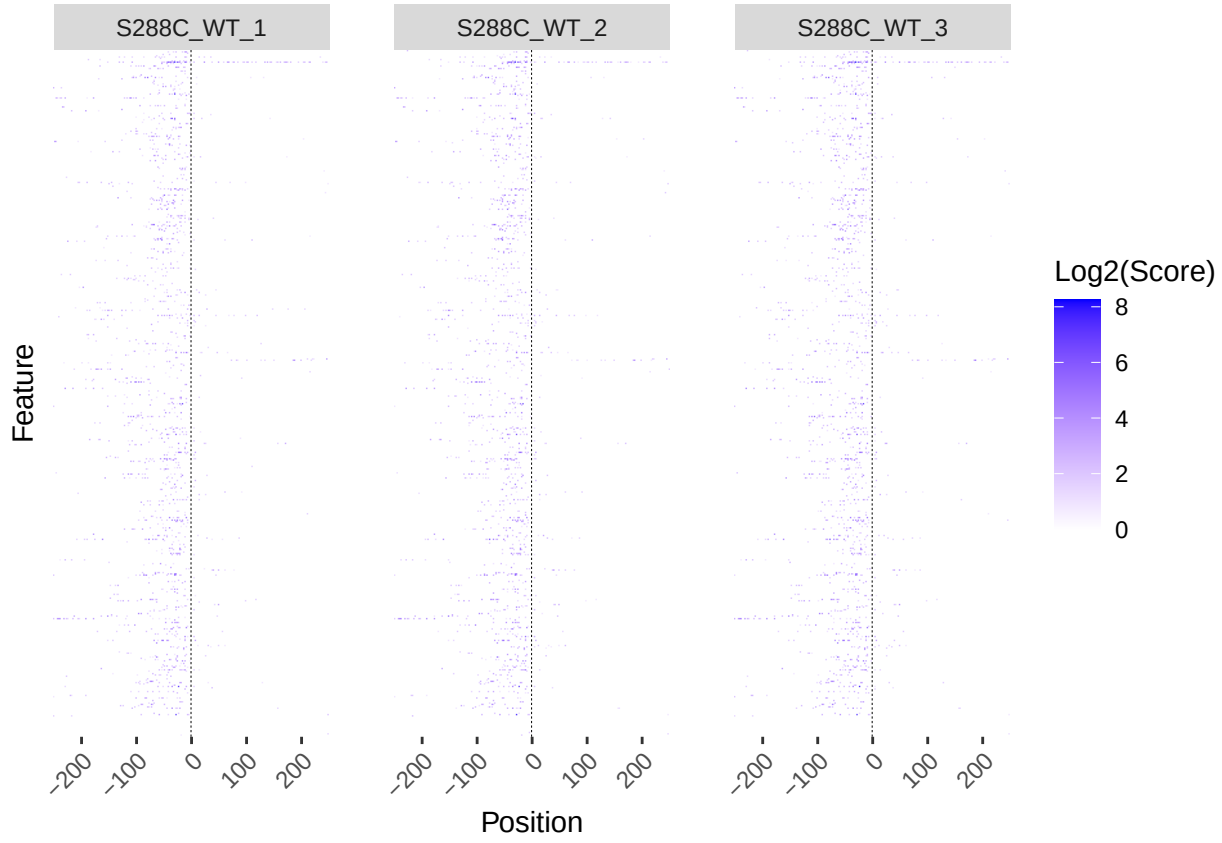


Heatmaps

While a density plot gives a general overview of TSS distribution relative to annotated start codons or TSSs, information may be lost in the process due to the aggregate nature of this type of plot. To get a less

condensed view of the data, it can be useful to generate a heatmap of TSS positions relative to all features. However, as TSS are single points and may be sparsely distributed, it can be somewhat difficult to visualize them as a heatmap.

```
plot_heatmap(
  exp, data_type="tss",
  upstream=250, downstream=250,
  rasterize=TRUE, raster_dpi=150
)
```



Sequence Analysis

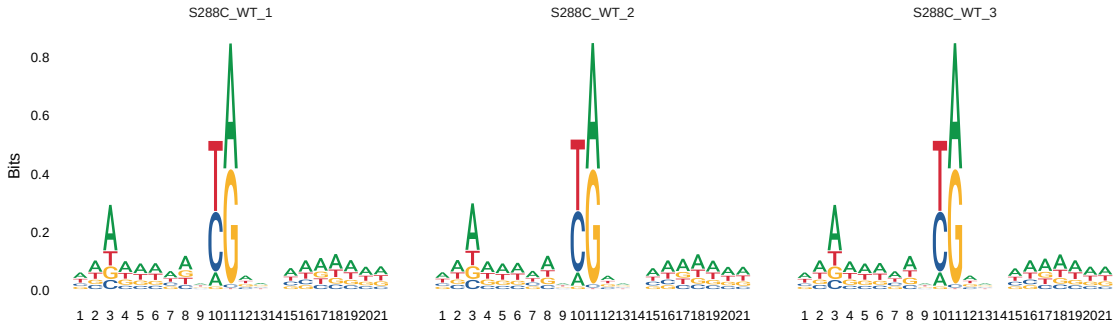
TSSs tend to occur within certain sequence contexts, and this context can vary between species. Knowing this TSS sequence bias can give mechanistic and biologically relevant information on promoter structure.

TSS Sequence Logo

Generating sequence logos around TSSs is a good first step in ascertaining the sequence context of TSSs. For example, in *S. cerevisiae* it has been reported that there is a pyrimidine-purine bias at the -1 and +1 positions, respectively. Furthermore, stronger TSSs tend to have an adenine at the -8 position, the loss of which diminishes promoter strength.

To generate analyze TSS sequence context, TSRexploreR retrieves sequences centered on TSSs from a 'FASTA' genome assembly or 'BSgenome' object. This example uses the Ensembl R64-1-1 Release 99 FASTA.

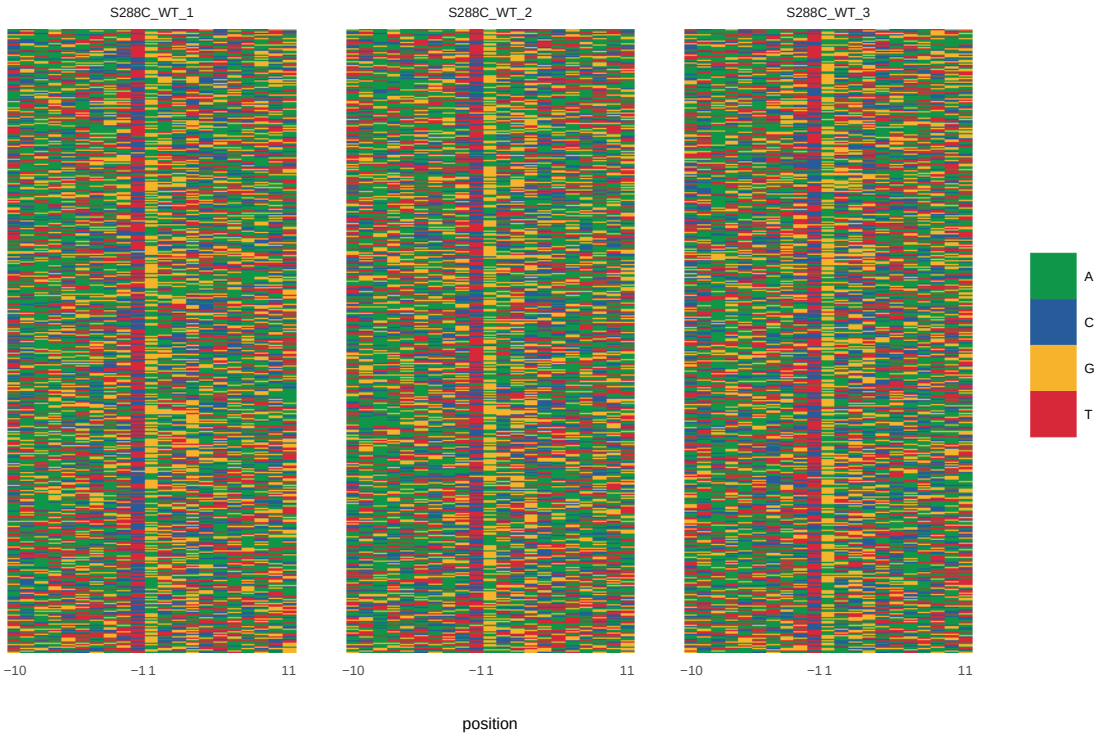
```
plot_sequence_logo(exp, ncol=3)
```



Sequence Color Map

A sequence logo "averages" the bases when displaying data, but as with TSSs, it can be useful to view the information without aggregation. In a sequence color map, each row represents the window around a TSS and each column represents a specific position relative to this TSS. These display conventions are analogous to those used in a heatmap. Each base is assigned a color and displayed in this two-dimensional space. The prevalence of colors in certain positions can give further evidence towards putative sequence contexts. The same genome assembly and retrieved sequences that were used to make the sequence logos above are used here.

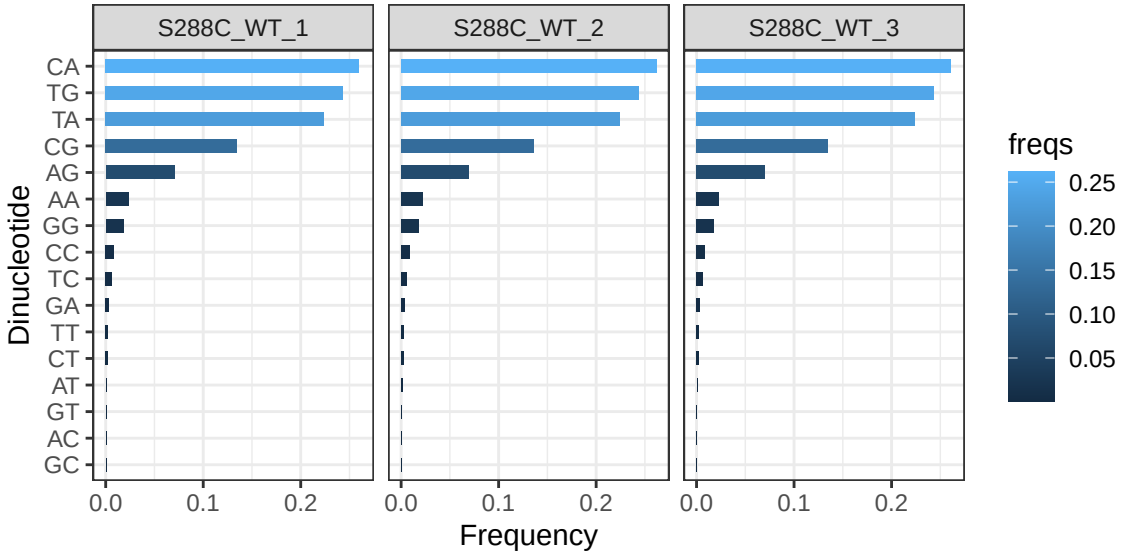
```
plot_sequence_colormap(exp, ncol=3, rasterize=TRUE)
```



Dinucleotide Frequency

As previously discused, organisms often have a specific TSS sequence context. This function explores the fraction of each potential dinucleotide at the -1/+1 positions of all TSSs, where the +1 position is the TSS.

```
plot_dinucleotide_frequencies(exp)
```



Clustering TSSs

Initiation is rarely homogenous. It is generally the case that a gene will have multiple TSSs clustered into a TSR, and so analyzing TSRs versus individual TSSs will provide more realistic information on initiation. Moreover, TSR shape features have been shown to correlate with distinct gene classes.

Distance Clustering

To identify TSRs, TSRexploreR uses a distance clustering method. This approach groups TSSs that pass a certain read threshold and are within a certain distance from one another into TSRs. However, if you have called TSRs using other methods (e.g. Paraclu or RECLU), they can be imported directly.

```
exp <- tss_clustering(exp, threshold=3, max_distance=25)
```

```
## Warning in .local(x, row.names, optional, ...): Arguments in '...' ignored

## Warning in .local(x, row.names, optional, ...): Arguments in '...' ignored

## Warning in .local(x, row.names, optional, ...): Arguments in '...' ignored

## Warning in .local(x, row.names, optional, ...): Arguments in '...' ignored

## Warning in .local(x, row.names, optional, ...): Arguments in '...' ignored

## Warning in .local(x, row.names, optional, ...): Arguments in '...' ignored
```

Associating TSSs with TSRs

TSRexploreR provides great flexibility for working with TSSs and TSRs. After TSRs are called or imported, TSSs can be assigned to TSRs they overlap with. In this example, TSSs are associated with the TSRs detected above using distance clustering. However, this function could also be used to associate TSSs with TSRs called elsewhere or a merged/consensus TSR set.

```
exp <- associate_with_tsr(exp)
```

Initial TSR Processing

Now that the TSSs have been clustered and associated with TSRs, the TSRs can undergo initial processing. This involves annotation, calculating various metrics of TSR shape and strength, and optional normalization of TSR scores.

TSR Metrics

After TSSs have been associated with TSRs, various metrics describing TSR shape and strength can be computed. First, the dominant TSS (that is, the highest-scoring TSS) within a TSR will be annotated. Then, various shape features such as inter-quantile range, shape index, and balance will be calculated.

```
exp <- mark_dominant(exp, data_type="tss")
exp <- tsr_metrics(exp)
```

Annotate TSRs

It is recommended to annotate TSRs if a genome annotation is available.

```
exp <- annotate_features(exp, data_type="tsr", feature_type="transcript")
```

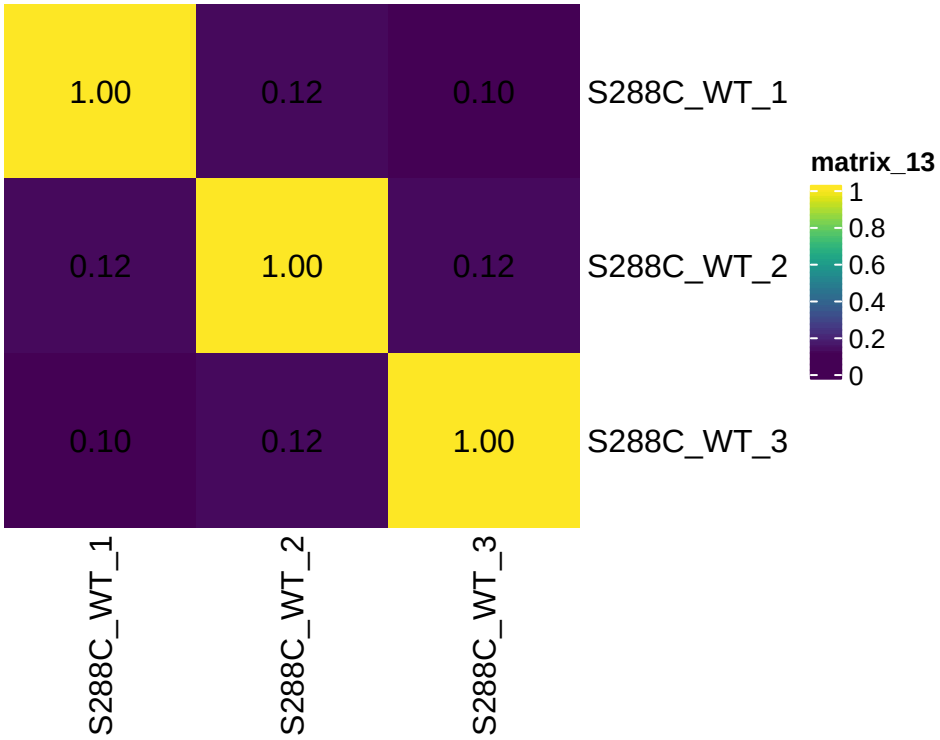
TSR Correlation

Similar to looking at TSS correlation, assessing TSR correlation before further analysis is good practice.

Correlation Matrix Plots

After TMM normalization, correlation plots can be generated.

```
plot_correlation(
  exp, data_type="tsr", font_size=12,
  heatmap_colors=viridis::viridis(100)
)
```



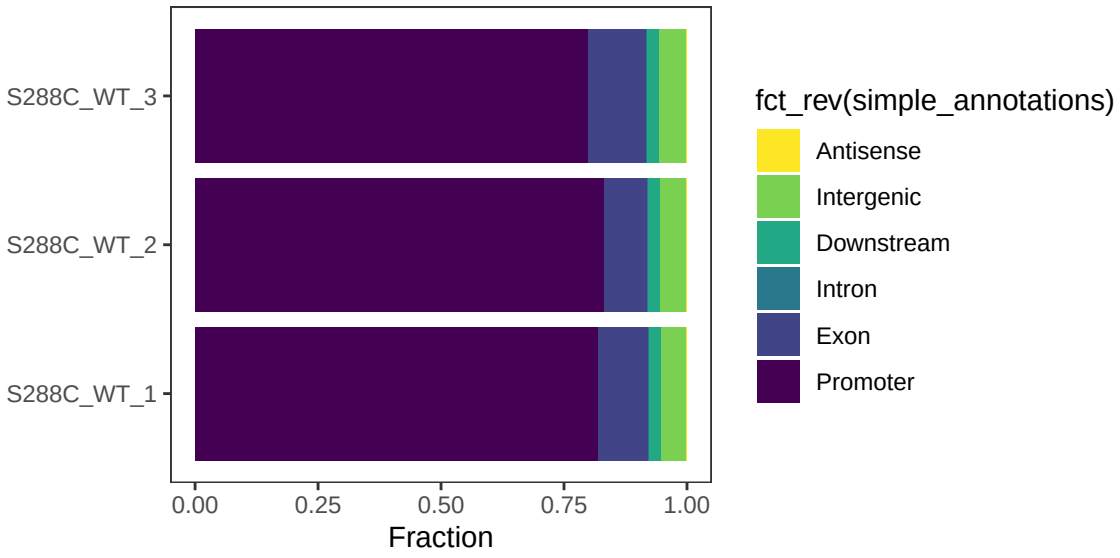
TSR Genomic Distribution

As for TSSs, it is expected that TSRs will be be enriched upstream of annotated start codons.

Genomic Distribution Plot

A stacked bar plot can be generated to visualize the fractional distribution of TSRs relative to known features.

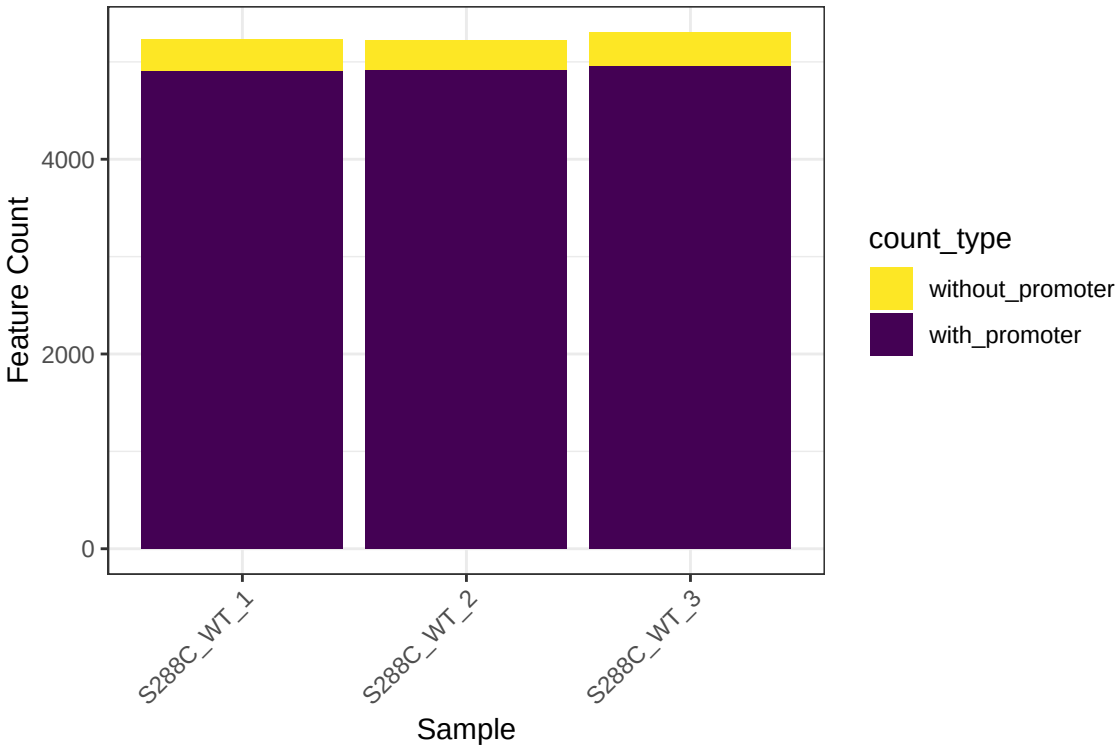
```
plot_genomic_distribution(exp, data_type="tsr") +
  scale_fill_viridis_d(direction=-1)
```



Feature Detection Plot

The number of features detected and fraction of features with a promoter-proximal TSSs can be displayed as a stacked bar plot or jitter plot.

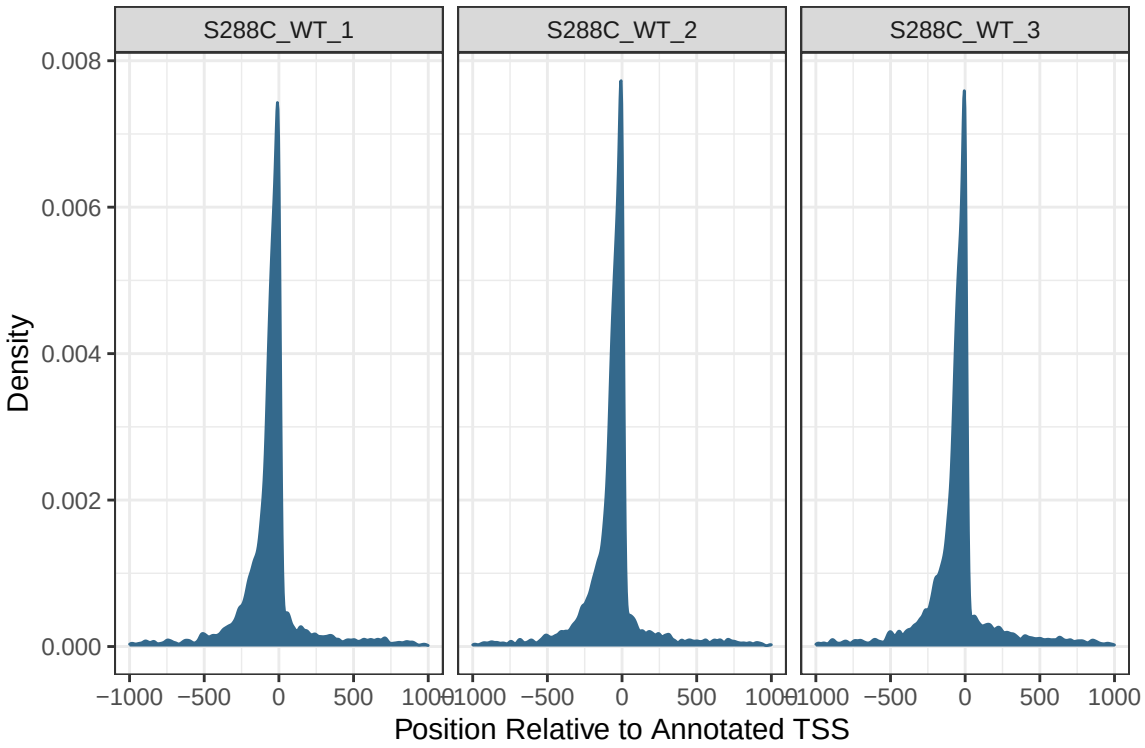
```
plot_detected_features(exp, data_type="tsr") +
  scale_fill_viridis_d(direction=-1)
```



Density Plots

As for TSSs, density plots can be generated for TSRs.

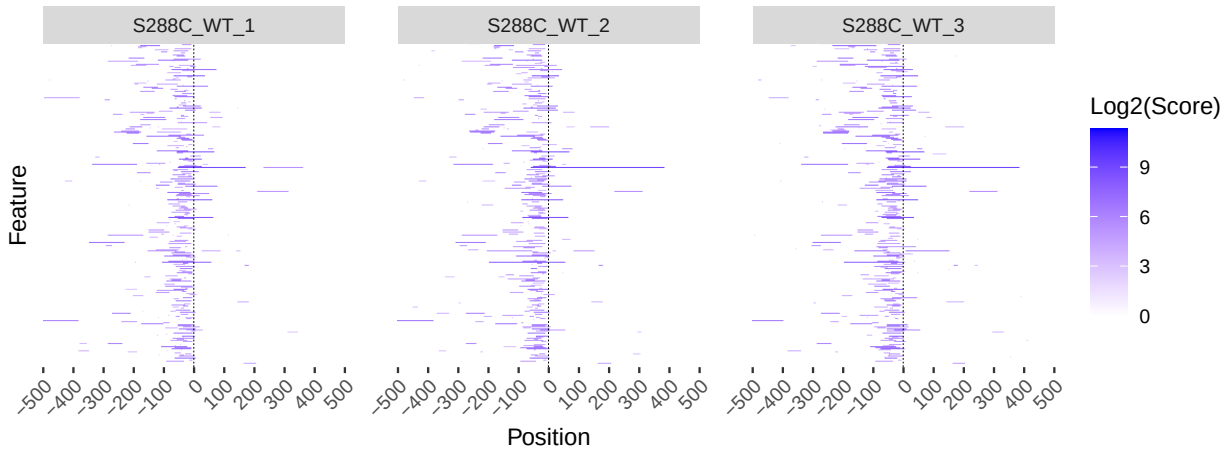
```
plot_density(exp, data_type="tsr", ncol=3)
```



Heatmaps

TSR heatmaps provide a global view of TSR distribution relative to annotated start codons or TSSs.

```
plot_heatmap(
  exp, data_type="tsr",
  upstream=500, downstream=500,
  rasterize=TRUE, raster_dpi=150
)
```



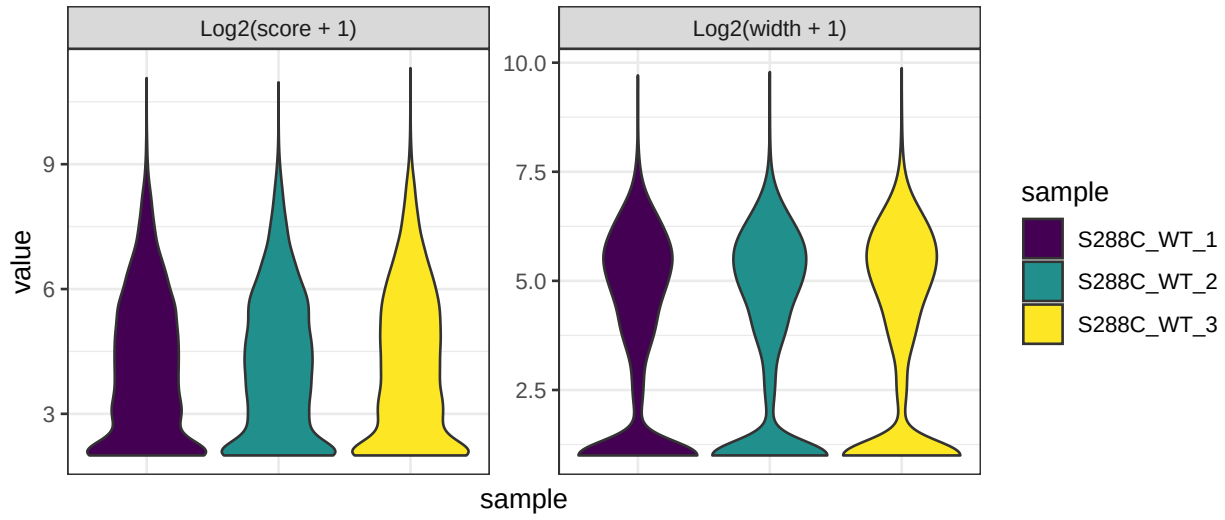
TSR Metrics and Shape

After clustering TSSs into TSRs, various descriptive and quantative measures are calculated: TSR width, log2 + 1 TSR score, shape index, inter-quantile width, peak balance, and peak concentration.

Summary Plots

Summary plots can be generated for most TSR metrics. In this example, we generate violin plots of the log2 + 1 transformed TSR score and TSR width.

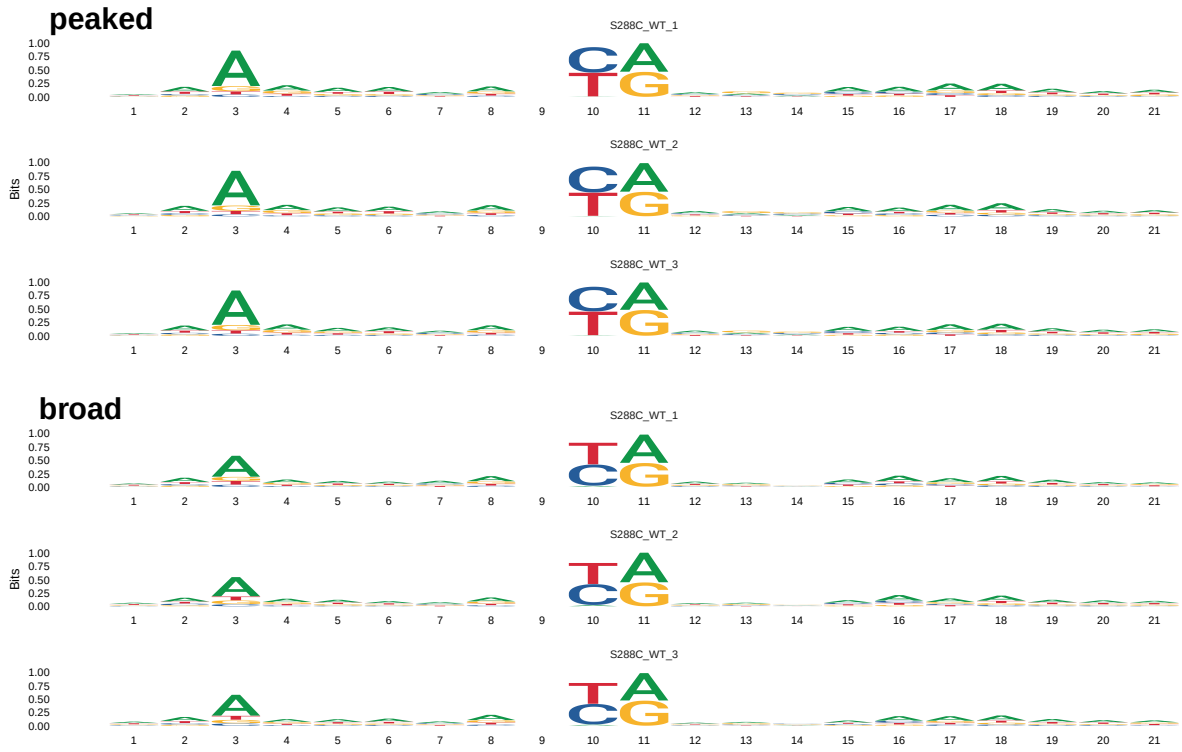
```
plot_tsr_metric(exp, tsr_metrics=c("score", "width"), log2_transform=TRUE, ncol=2)
```



Descriptive Plots

Most plots in this vignette can be filtered, ordered, grouped, and quantiled by any of the desired metrics. Here is an example of what can be done with the TSS sequence motif plot using TSR metrics. In this plot, only the dominant TSSs of each TSR is considered, and TSSs with a score below 10 are discarded. The plot is then split by the shape class of the TSR. A more detailed guide to advanced plotting can be found here.

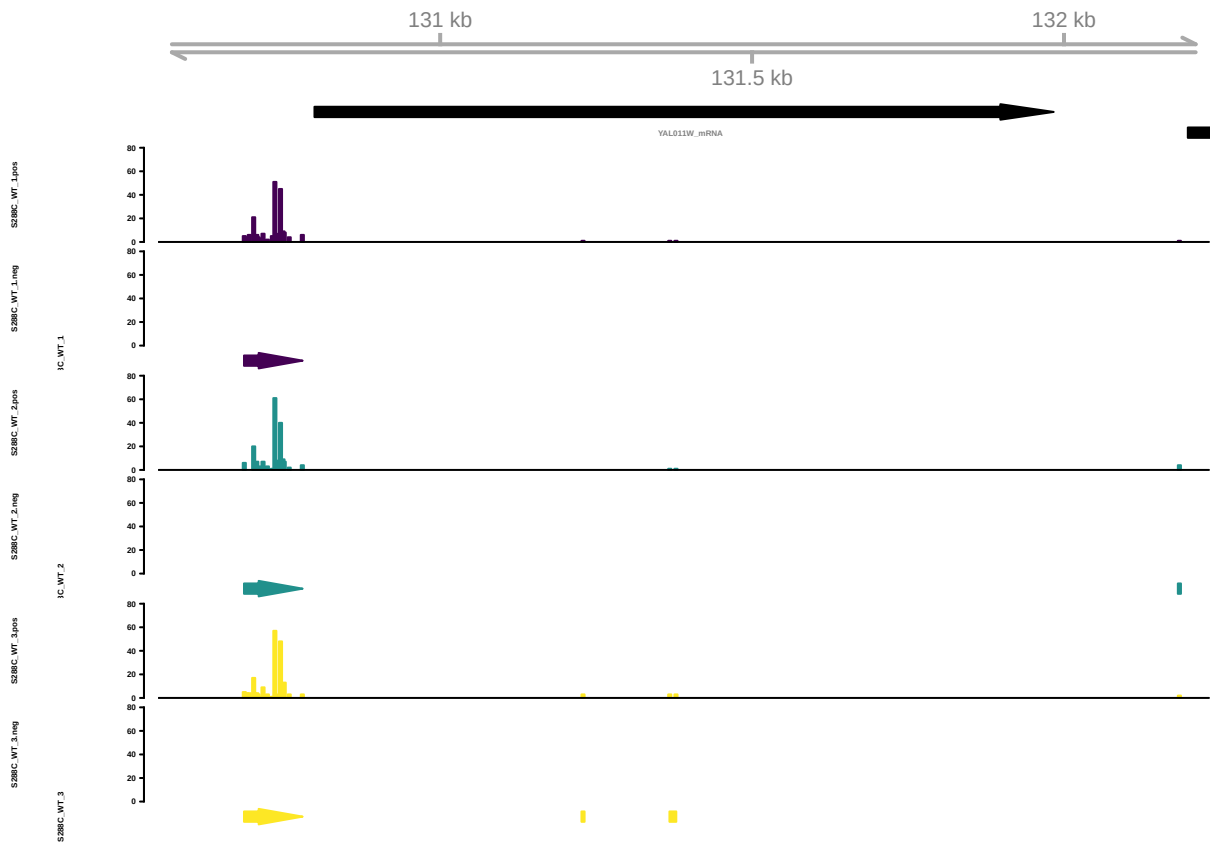
```
plot_sequence_logo(
  exp, threshold=10, dominant=TRUE,
  data_conditions=conditions(data_grouping=shape_class)
)
```



Gene Tracks

Viewing TSSs and/or TSRs in gene tracks can be useful for a variety of reasons. It can make clear which 5' isoforms of a transcript are expressed, hint at potential misannotation of genes, uncover 5' UTR structure, and various other goodies. In TSRexploreR, tracks can be created based on a gene/transcript name or genomic coordinates. Additionally, if tracks are generated based on genes/transcripts, the promoter region alone can optionally be displayed.

```
gene_tracks(
  exp, feature_name="YAL012W",
  samples=c(
    TSS="S288C_WT_1", TSR="S288C_WT_1",
    TSS="S288C_WT_2", TSR="S288C_WT_2",
    TSS="S288C_WT_3", TSR="S288C_WT_3"
  ),
  ymax=80, tss_colors=viridis::viridis(3), tsr_colors=viridis::viridis(3)
)
```


131 kb
132 kb
131.5 kb
YAL011W_mRNA
S288C_WT_1.pos
S288C_WT_1.neg
IC_WT_1
S288C_WT_2.pos
S288C_WT_2.neg
IC_WT_2
S288C_WT_3.pos
S288C_WT_3.neg
IC_WT_3