SUPPLEMENTAL MATERIAL

Kathrin M. Seibt, Thomas Schmidt and Tony Heitkam

FlexiDot: Highly customizable, ambiguity-aware dotplots for visual sequence investigation

Index

1 FlexiDot background

Dotplots are effective, illustrative and simple tools for sequence investigation, leading to a range of dotplot applications. Yet, essential features are absent or scattered across available implementations. Our dotplot suite FlexiDot combines established (reverse complements, all-against-all modes) and new options in one package. We introduce ambiguity handling, similarity shading modes, and integrate gff3-type sequence annotation, while retaining high flexibility for customization and automation.

The tool FlexiDot creates informative, high quality dotplots for a wide range of applications in sequence analysis and comparison. These dotplots allow identification of shared domains, diverged homologous regions, repetitive DNA classes, or structural rearrangements on a variety of targets, such as short sequences, genome assembly regions of more than 100 kb or error-prone single molecule real-time (SMRT) reads.

1.1 FlexiDot requirements and installation

FlexiDot is implemented in Python 2.7, using numpy, matplotlib, biopython, regex, colormap, and colour bindings (Supplemental Table 1). It is started over the command line interface with

```
python flexidot.py -i input.fas [optional arguments]
```

Upon its first start, FlexiDot calls all required modules. If absent, it installs them automatically using Python’s install manager `pip`. To facilitate the installation process, it is recommended to start FlexiDot with administrator privileges on its first run.

Supplemental Table 1: FlexiDot requirements

Python module	**Website**
numpy	https://pypi.python.org/pypi/numpy
matplotlib	https://pypi.python.org/pypi/matplotlib
biopython	https://pypi.python.org/pypi/biopython
regex	https://pypi.python.org/pypi/regex
colormap	https://pypi.python.org/pypi/colormap
easydev (required for color map)	https://pypi.python.org/pypi/easydev
colour	https://pypi.python.org/pypi/colour

1.2 Input sequences

Both nucleotide and amino acid sequences in fasta format can be used as input for Flexidot. If the sequences contain gaps, automatic degapping is initiated before proceeding. Please note, that sequence names must be unique.

There are three possibilities to provide the input fasta sequences:

- as single fasta file (option `-i/--in_file`)
- as multiple input files (comma-separated list with `-i/-in_file` or reusing `-i/--in_file`)

- using the automatic detection of all fasta sequence files in the directory in which the Python script is located (option `-a/--auto_fas`) – all files with the filename extensions “.fasta”, “.fas”, “.fa”, and “.fna” are automatically included in the analysis

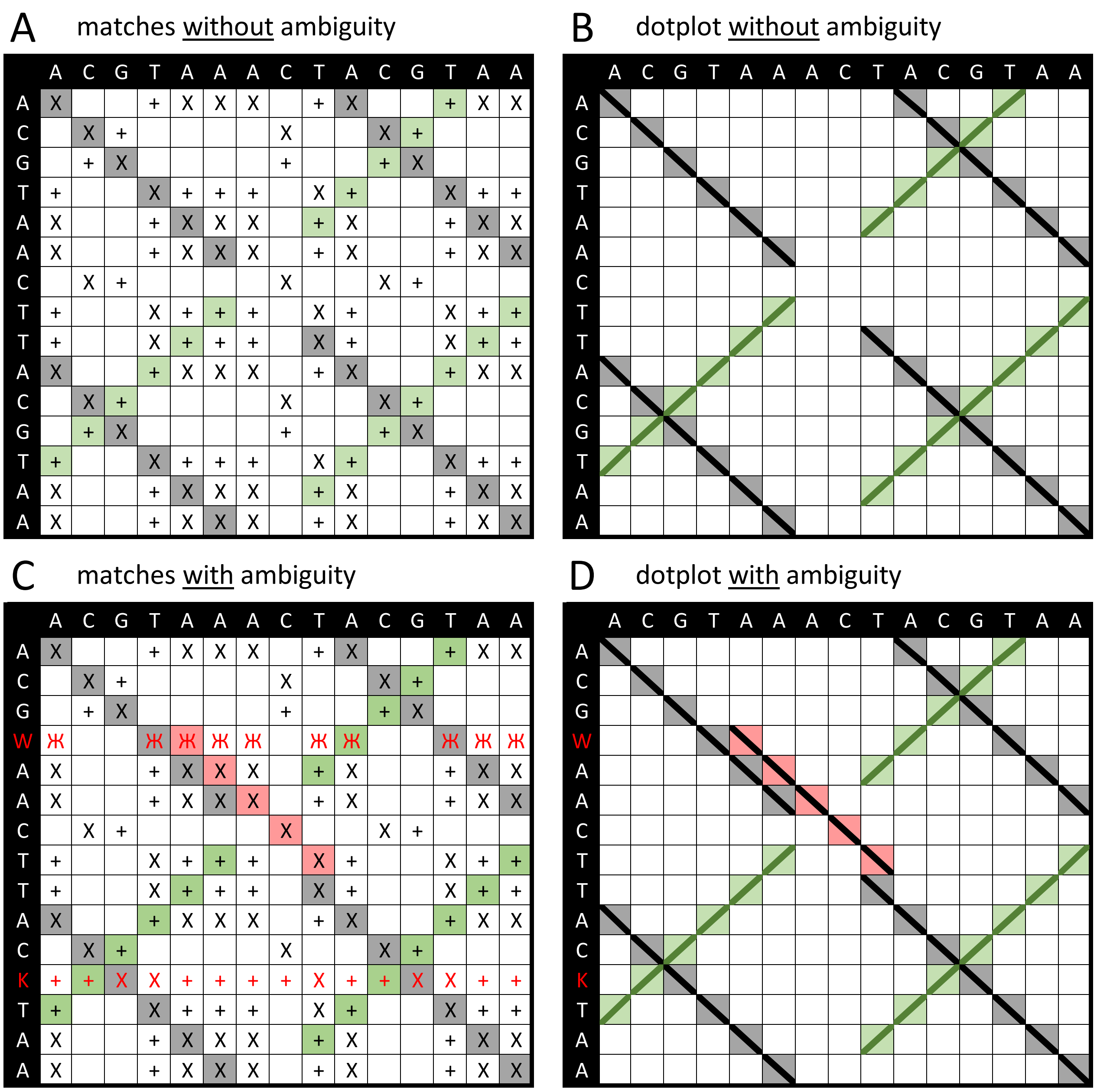
In order to illustrate FlexiDot features, we use six artificial nucleotide sequences (Supplemental Table 2). These test sequences harbor forward and reverse matches, sequence ambiguities, repeats and inverted repeats – all derived from a shared monomeric subunit. The characteristics included in the individual sequences are as follows:

Supplemental Table 2: Artificial test sequences for FlexiDot performance demonstration and evaluation of commonly used dotplot software.

Title	**Sequence**	**Description**
Seq1	`CGAACCGATATAATCAGTGACAGGATATCGAGTAAAGTGACAGGATATCGAGTAAAGTGACAGGATATCGAGTAAAATGAGCTATAGGACAGTGACGTGACTTAGAGG`	tandem repeat
Seq2	`CGAACCGATATAATCAGTGACAGGATATCGAGTAAAGTGACAGGATATCGAGTAAAGTGACAGGATATCGAGTAAAATGAGCTATAGGACAGTGATTACTCGATATCCTGTCACTTTACTCGATATCCTGTCACTTTACTCGATATCCTGTCACTTTACTCGATATCCTGTCACTTTACTCGATATCCTGTCACTTTCGTGACTTAGAGG`	tandem repeat structures in forward and reverse orientation (reverse complement of Seq3)
Seq3	`CCTCTAAGTCACGAAAGTGACAGGATATCGAGTAAAGTGACAGGATATCGAGTAAAGTGACAGGATATCGAGTAAAGTGACAGGATATCGAGTAAAGTGACAGGATATCGAGTAATCACTGTCCTATAGCTCATTTTACTCGATATCCTGTCACTTTACTCGATATCCTGTCACTTTACTCGATATCCTGTCACTGATTATATCGGTTCG`	tandem repeat structures in reverse and forward orientation (reverse complement of Seq2)
Seq4	`CCTCTAAGTCACGDAAGTGACSGGATATCGAGTAAAGTGNCAGGATATNGAGYAAAGVGACAGGAKATCGAGTHAAGTGACAGGATATCGAGTAAAGTGACAGGATBSCGAHTAATCAVTGVCCTATAGRRCATTTTACTCGATATCCVGTYACTTTBSTCGATATCCTGKCACTTMACTCGATATCCTGTCACTGATTATATCGGTTCG`	tandem repeat structures in reverse and forward orientation with mismatches and sequence ambiguities (diverged, ambiguous Seq3)
Seq5	`CCAAAGTGACAGGATATCGAGTCCGG`	monomeric sequence motif from repeats in Seq1 to Seq4
Seq6	`AATCAGTGACAGGATATCCTGTCACTGATT`	inverted repeat derived from repeat monomer (Seq4)

1.3 FlexiDot match calculation

FlexiDot conducts presence/absence matching for sequence inspection and comparison (Supplemental Figure 1 A). Optionally, ambiguity and mismatch handling can be used to allow the comparison of sequences containing wobble residues and error-prone or diverged sequences, respectively (Supplemental Figure 1 C-D). Ambiguous residues are treated as matches, if the residues encoded by the ambiguities are present in the other sequence (or their reverse complement in case of reverse matching).


A matches without ambiguity
B dotplot without ambiguity
C matches with ambiguity
D dotplot with ambiguity
A C G T A A A C T A C G T A A
A C G T A A C T T A C G T A A
A C G W A A C T T A C K T A A

A

	A	C	G	T	A	A	A	C	T	A	C	G	T	A	A
A	X			+	X	X	X		+	X			+	X	X
C		X	+					X			X	+			
G		+	X					+			+	X			
T	+			X	+	+	+		X	+			X	+	+
A	X			+	X	X	X		+	X			+	X	X
A	X			+	X	X	X		+	X			+	X	X
C		X	+					X			X	+			
T	+			X	+	+	+		X	+			X	+	+
T	+			X	+	+	+		X	+			X	+	+
A	X			+	X	X	X		+	X			+	X	X
C		X	+					X			X	+			
G		+	X					+			+	X			
T	+			X	+	+	+		X	+			X	+	+
A	X			+	X	X	X		+	X			+	X	X
A	X			+	X	X	X		+	X			+	X	X

C

	A	C	G	T	A	A	A	C	T	A	C	G	T	A	A
A	X			+	X	X	X		+	X			+	X	X
C		X	+					X			X	+			
G		+	X					+			+	X			
W	Ж			Ж	Ж	Ж	Ж		Ж	Ж			Ж	Ж	Ж
A	X			+	X	X	X		+	X			+	X	X
A	X			+	X	X	X		+	X			+	X	X
C		X	+					X			X	+			
T	+			X	+	+	+		X	+			X	+	+
T	+			X	+	+	+		X	+			X	+	+
A	X			+	X	X	X		+	X			+	X	X
C		X	+					X			X	+			
K	+	+	X	X	+	+	+	+	X	+	+	X	X	+	+
T	+			X	+	+	+		X	+			X	+	+
A	X			+	X	X	X		+	X			+	X	X
A	X			+	X	X	X		+	X			+	X	X

Supplemental Figure 1: FlexiDot dotplot calculation scheme for wordsize 5. Matches are indicated by “x” (forward; panel A, C), “+” (complement; panel A, C), and Ж (forward and complement, panel C). Shared subsequences with a wordsize ≥ 5 are shaded and represented with lines in the dotplot image (B, D). Examples are shown for forward (grey) and reverse matches (green) without (A, B). Ambiguity handling enables the detection of an additional match (red; C, D).

2 FlexiDot features

2.1 FlexiDot base ambiguity and mismatch handling

The matching stringency of FlexiDot can be reduced in two ways: (1) Ambiguity handling (option `-w/--wobble_conversion`) treats ambiguous residues as matches, if the ambiguous residues are present in the other sequence (Supplemental Figure 1 C-D). This allows the comparison of species-specific representations of multigene or repeat families as well as common variants or sequence subfamilies. (2) Mismatch handling `-S, --substitution_count`) allows a predefined number of substitution mutations, useful for analyzing error-prone or diverged sequences.

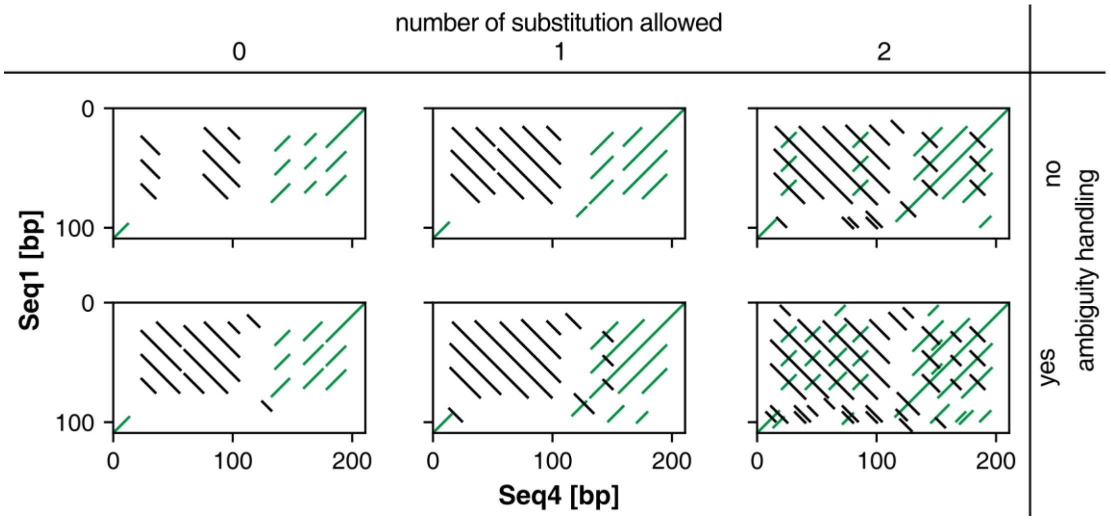
Ambiguity handling is available in all three plotting modes and is switched on or off by option `-w/--wobble_conversion`. The considered ambiguities follow the IUPAC notation for nucleic and amino acid ambiguities as shown in Supplemental Table 3. If poly(N)-stretches contribute half of the wordsize or more, they are masked to prevent large, uninformative dotplot regions. The same applies to the X-residue in proteins.

Supplemental Table 3: FlexiDot's implemented ambiguity code

Ambiguity	Ambiguity content	Comment
nucleic acid sequence		
N	A, C, G, T	any
B	C, G, T	not A
D	A, G, T	not C
H	A, C, T	not G
V	A, C, G	not T
Y	C, T	pyrimidine
R	A, G	purine
W	A, T	weak
S	C, G	strong
K	G, T	keto
M	A, C	amino
amino acid sequence		
X	A, R, N, D, C, E, Q, G, H, I, L, K, M, F, P, S, T, W, Y, V, U, O, *	any
J	I, L	---
Z	Q, E	---
B	N, D	---

To illustrate the performance of the mismatch and ambiguity handling features, we show FlexiDot pairwise dotplots of test sequences Seq1 and ambiguity-containing Seq4 (Supplemental Table 2). The resulting dotplots show reduced matches without wobble recognition (Supplemental Figure 2, upper row). Switching on the ambiguity handling feature can result in additional and longer matches (Supplemental Figure 2, lower row). Thereby, similarities in consensus sequences of diverged repeats can be identified, e.g. in case of frequently observed transition mutation events in nucleotide sequences, such as the common conversion of methylated cytosine to thymine.

Mismatch toleration is another possibility to obtain longer matches. Increasing the tolerated mismatches leads to a fast decrease of matching stringency (Supplemental Figure 2, second and third column). The number of accepted mismatches within the search window is specified by option `-S, --substitution_count` and allows single nucleotide polymorphisms by substitution. Deletions and insertions are not allowed. This is particularly helpful when dealing with error-prone SMRT reads, where nucleotide substitution rates can exceed 10%.



Supplemental Figure 2: FlexiDot pairwise sequence comparison of an ambiguity-containing nucleotide sequence without and with base ambiguity handling. In addition, 0, 1 and 2 mismatches are tolerated, respectively. Recognition of base ambiguities and mismatches leads to longer and additional diagonals and facilitates identification of diverged repeat motifs.

```
Used commands Supplemental Figure 2:
Panel tl>> python flexidot.py -i Seq4.fas,Seq1.fas -p 1 -D n -f 0 -c n -k 10 -w n -r y -x n
Panel tm>> python flexidot.py -i Seq4.fas,Seq1.fas -p 1 -D n -f 0 -c n -k 10 -w n -r y -x n -S 1
Panel tr>> python flexidot.py -i Seq4.fas,Seq1.fas -p 1 -D n -f 0 -c n -k 10 -w n -r y -x n -S 2
Panel bl>> python flexidot.py -i Seq4.fas,Seq1.fas -p 1 -D n -f 0 -c n -k 10 -w y -r y -x n
Panel bm>> python flexidot.py -i Seq4.fas,Seq1.fas -p 1 -D n -f 0 -c n -k 10 -w y -r y -x n -S 1
Panel br>> python flexidot.py -i Seq4.fas,Seq1.fas -p 1 -D n -f 0 -c n -k 10 -w y -r y -x n -S 2
```

2.2 FlexiDot plotting modes and output options for multifasta sequences

FlexiDot allows sequence investigation in three run modes using the option `-p/--plotting_modes` (see Figure 1, main manuscript):

- `-p 0` self sequence comparison (chapter 2.2.1)
- `-p 1` pairwise sequence comparison (chapter 2.2.2)
- `-p 2` all-to-all sequence comparison (chapter 2.2.3)

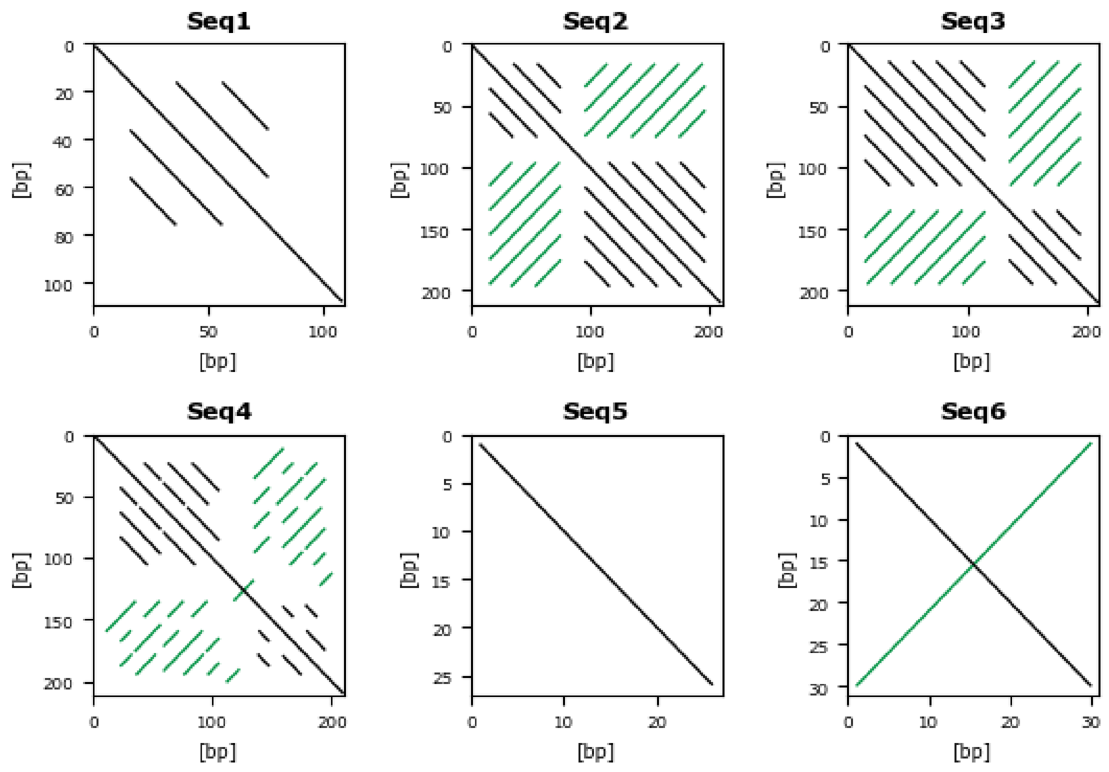
Multiple running modes can be analyzed consecutively by reusing `-p/--plotting_modes` or by providing comma-separated integers as argument.

If more than one sequence is used for plotting modes `-p 0` and `-p 1`, FlexiDot can either combine the output to a dotplot collage (option `-c/--collage_output 1` [default]) or save the output in individual files (option `-c/--collage_output 0`). For collages, the user can specify the number of columns and rows per page (options `-m/--`

`m_col` and `-n/--n_row`). Please note, that in collages all dotplot images are scaled to the same size despite differing input sequence lengths. The according axis labels are included in the images.

2.2.1 Self dotplots (plotting mode `-p 0`)

In the self dotplot mode, each sequence is compared with itself. This enables the identification of structural motifs such as inverted or direct repeats as well as sequence duplications. As mentioned, the resulting dotplots can be arranged as a collage (default; Supplemental Figure 3) or written to separate files.



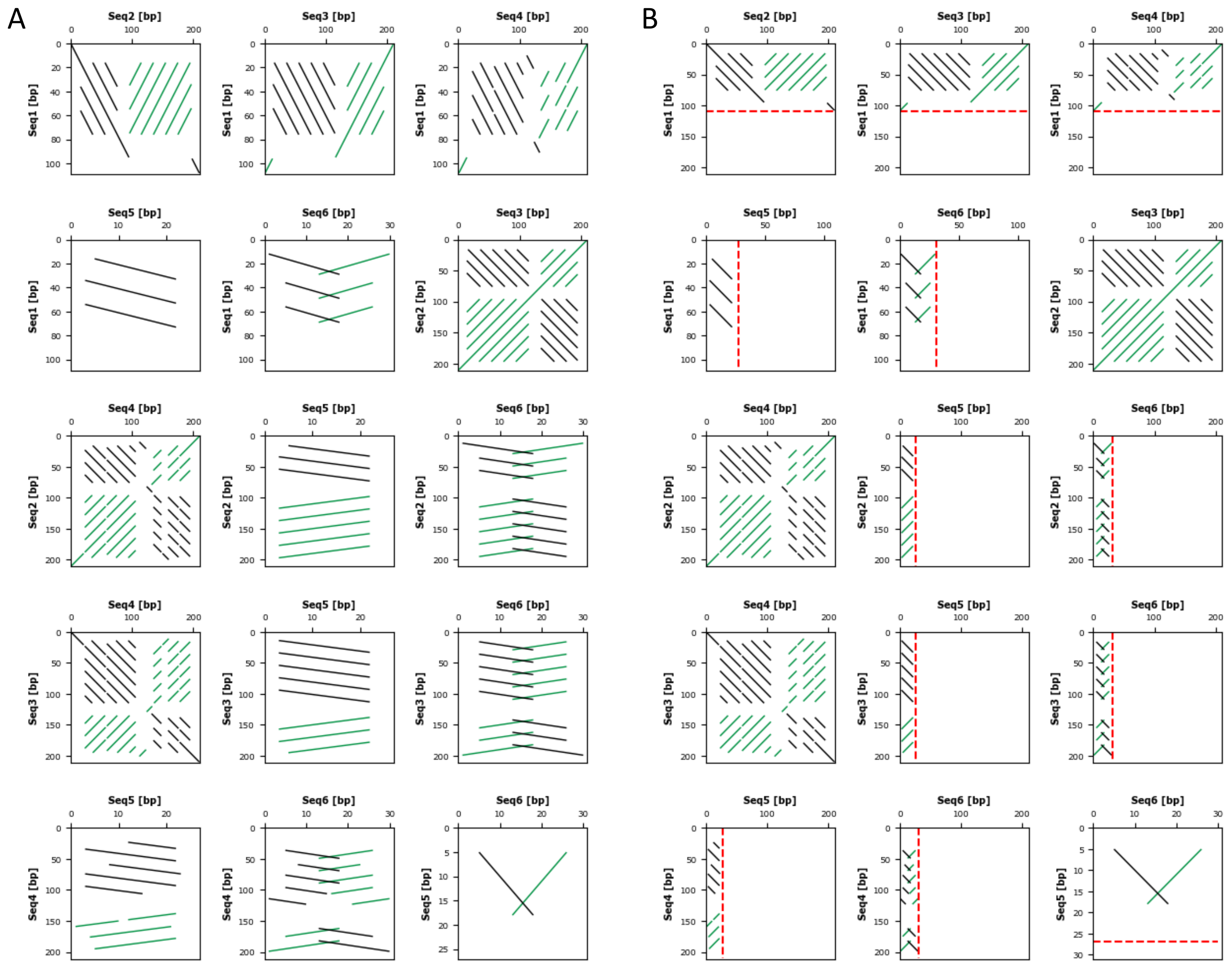
Supplemental Figure 3: FlexiDot self comparison with sequence ambiguity handling for all six test sequences as 3x2 collage. The sequence features, as presented in Supplemental Table 2, are visible. Forward and reverse complementary matches are indicated by black and green lines, respectively. Parallel lines show tandem repeats. The crossing lines in Seq4 and Seq6 indicate an inverted repeat.

```
Used command Supplemental Figure 3:
>> python flexidot.py -i test-seqs.fas -p 0 -D y -f 0 -k 10 -w y -r y -x n -m 3
```

2.2.2 Pairwise dotplots (plotting mode `-p 1`)

The pairwise plotting mode can be used to compare each sequence pair individually. This is useful for identification of similarities like shared structural domains or sequence motifs. To reduce the number of generated files, we recommend output as a collage. The collage of the six test sequences comprises 15 pairwise dotplots (Supplemental Figure 4). By default, dotplot images are in square format (Supplemental Figure 4 A). This maximizes the visibility of matches, if the compared sequences differ drastically in length. To enable scaling according to the respective sequence lengths, the FlexiDot scaling feature can be executed by option `-`

`L/--length_scaling` (Supplemental Figure 4 B). If scaling is enabled, a red line indicates the end of the shorter sequence in the collage output.

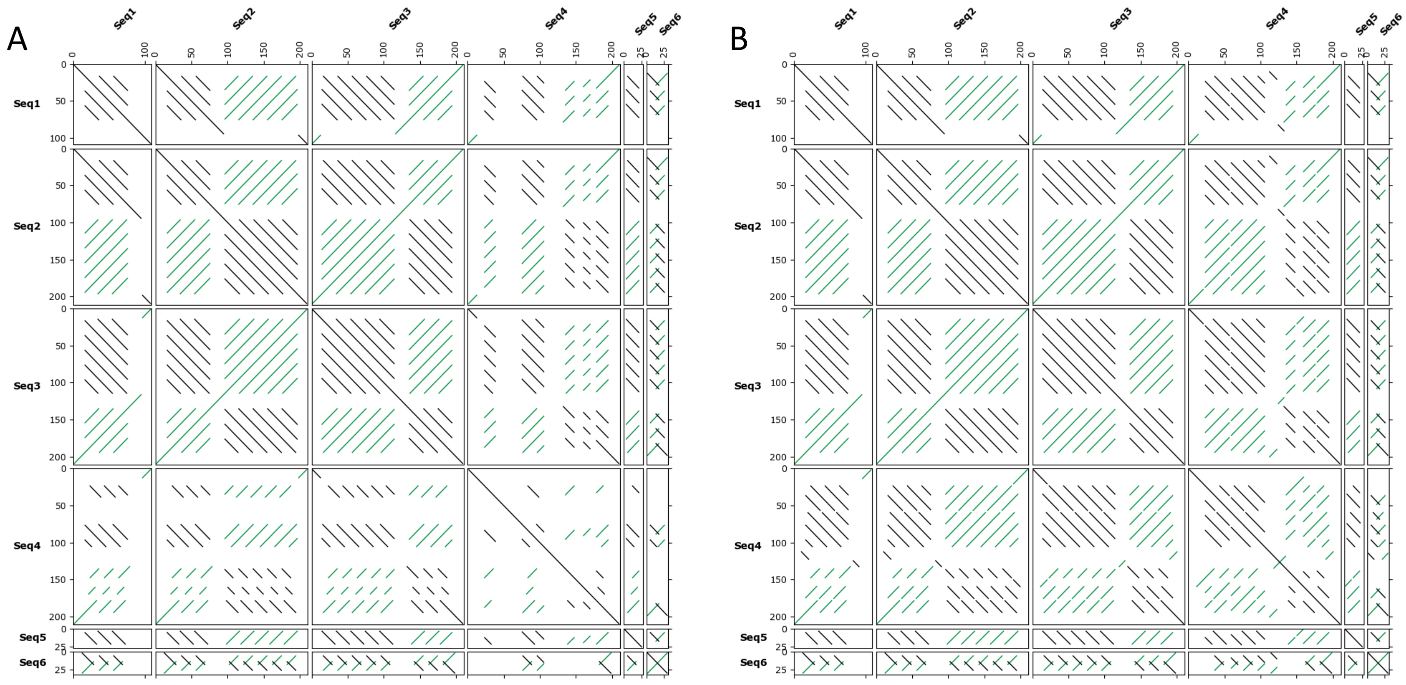


Supplemental Figure 4: Pairwise dotplot collage of the test sequences with sequence ambiguity handling. All sequences are compared against all other sequences. Output options comprise unscaled (A) and scaled (B) dotplot representations. The end of the shorter sequence is represented as a dotted red line. The unscaled images maximize the visibility of the matches at the cost of distorting the lines (e.g. Seq5 vs. Seq1).

```
Used commands Supplemental Figure 4:
Panel A>> python flexidot.py -i test-seqs.fas -p 1 -D y -f 0 -k 10 -w y -r y -m 5 -c y -L n
Panel B>> python flexidot.py -i test-seqs.fas -p 1 -D y -f 0 -k 10 -w y -r y -m 5 -c y -L y
```

2.2.3 All-against-all dotplots (plotting mode `-p 2`)

All-against-all dotplots allow comprehensive sequence comparisons. They effectively combine self (middle diagonal) and pairwise comparisons (Supplemental Figure 5). To enable immediate identification of sequences sharing long matches, shading can be applied (see chapter 2.3.3).



Supplemental Figure 5: FlexiDot all-against-all sequence comparison without (A) and with (B) base ambiguity handling. Ambiguity matching leads to longer matches to Seq4.

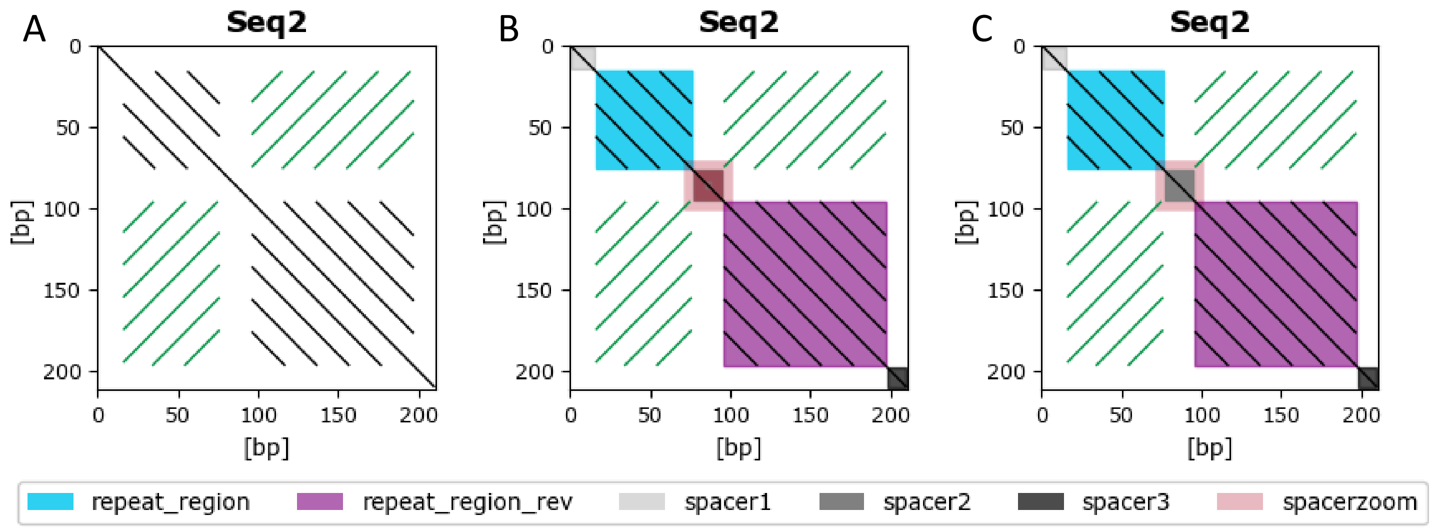
```
Used commands Supplemental Figure 5:
Panel A>> python flexidot.py -i test-seqs.fas -p 2 -D y -f 0 -t y -k 10 -w n -r y -x n
Panel B>> python flexidot.py -i test-seqs.fas -p 2 -D y -f 0 -t y -k 10 -w y -r y -x n
```

2.3 Plotting mode-specific features

2.3.1 GFF coloring for self dotplots

In FlexiDot self dotplots, annotated sequence regions can be highlighted by shading to allow clear assignment of dotplot regions to specific sequence contexts (Supplemental Figure 6). The underlying annotation information must be provided in general feature format (gff3), either as individual file or comma-separated file list using the `-g/--input_gff_files` option. Shadings according to the annotation feature type (column 3 in gff3 files) follow a color scheme, which can be customized. The default coloring scheme defines aesthetics for predefined feature types, such as “gene”, “exon”, “intron”, “UTR”. Unknown feature types will be treated as “other” and shaded grey. A color legend is generated as a separate figure file.

To customize GFF-based shading, a user-defined configuration file can be provided with the `-G/--gff_color_config` option (exemplary files for Supplemental Figure 6 are shown). The configuration file must be tab-delimited with the following four columns: feature type, color (e.g. “red” or “#FF00FF”), alpha (floating point number; 0 = transparent, 1 = opaque), and zoom (floating point number). The alpha value can be used to illustrate overlapping annotations, whereas a zoom can improve visibility of short annotations. The zoom value is additive to the specified annotation span and is automatically corrected to not exceed the length of the analyzed sequence. Distinct configurations can be specified depending on the feature orientation, if an additional feature with the suffix "_rev" is listed. In the configuration file, user-defined comments can be included as additional columns as well as comment lines starting with "#". Please note, that the order of the annotations in the gff3 file determines the order of the drawn highlights (Supplemental Figure 6).



Supplemental Figure 6: Custom annotation shading of FlexiDot self dotplots. The Seq2 self dotplot is presented without (A), and with annotation-based shadings (B, C). The modified order of the annotations in the gff3 file changes the plotting order of the shaded regions around 85 bp – compare panel B and C, where the annotation field of "*Spacer3*" (grey) is drawn on top or underneath the "*SpacerZoom*" shading (red).

```
Used commands Supplemental Figure 6:
Panel A>> python flexidot.py -i Seq2.fas -p 0 -D y -f 0 -k 10 -w y -r y -x n -m 3 -P 15
Panel B>> python flexidot.py -i Seq2.fas -p 0 -D y -f 0 -k 10 -w y -r y -x n -m 3 -P 15 -g example1.gff3 -G example.config
Panel C>> python flexidot.py -i Seq2.fas -p 0 -D y -f 0 -k 10 -w y -r y -x n -m 3 -P 15 -g example2.gff3 -G example.config
```

```
example1.gff3 (tab-delimited)

Seq2   manual_annotations  Spacer1            1     15    .    -    .    ID=0001
Seq2   manual_annotations  repeat_region      16    76    .    +    .    ID=0002
Seq2   manual_annotations  Spacer2            77    95    .    +    .    ID=0003
Seq2   manual_annotations  SpacerZoom         77    95    .    +    .    ID=0004
Seq2   manual_annotations  repeat_region_rev  96    197   .    -    .    ID=0005
Seq2   manual_annotations  Spacer3            198   210   .    +    .    ID=0006
```

```
example2.gff3 (tab-delimited)

Seq2   manual_annotations  Spacer1            1     15    .    -    .    ID=0001
Seq2   manual_annotations  repeat_region      16    76    .    +    .    ID=0002
Seq2   manual_annotations  SpacerZoom         77    95    .    +    .    ID=0004
Seq2   manual_annotations  Spacer2            77    95    .    +    .    ID=0003
Seq2   manual_annotations  repeat_region_rev  96    197   .    -    .    ID=0005
Seq2   manual_annotations  Spacer3            198   210   .    +    .    ID=0006
```

```
example.config (tab-delimited)

#annotation_type    color       alpha       zoom (linewidth_adjustment)
repeat_region       #2dd0f0     1           0
repeat_region_rev   purple      0.6         0
Spacer1             black       0.15        0
Spacer2             grey        1           0
Spacer3             black       0.7         0
SpacerZoom          #b41a31     0.3         8
```

2.3.2 Longest Common Subsequence (LCS) table for pairwise and all-against-all comparisons

During pairwise sequence comparison, quantitative information is acquired on the LCS length. Accordingly, an output table is provided for pairwise and all-against-all comparisons containing the lengths of forward and reverse LCS for each sequence pair. For the test sequences (Supplemental Table 2), the overall longest match is 210 bp (all-against-all LCS table, Supplemental Table 4). It is detected for self-comparisons between the sequences Seq2, Seq3, and Seq4. As expected, a match of the same length in reverse orientation is observed between Seq2 and Seq3. If self comparisons are excluded, the longest forward match is shorter, with 95 bp only.

Supplemental Table 4: LCS all-against-all table for the example sequences. The longest forward and reverse matches are highlighted (underlined font; grey (forward) and green shading (reverse complementary), respectively. Rows corresponding to self-comparisons are presented in grey font.

#seq1	seq2	len_seq1	len_seq2	len_lcs_for	%_min_seq_len	len_lcs_rev	%_min_seq_len
Seq1	Seq1	108	108	108	100.000	6	5.556
Seq2	Seq1	210	108	95	87.963	61	56.481
Seq3	Seq1	210	108	61	56.481	95	87.963
Seq4	Seq1	210	108	48	44.444	33	30.556
Seq5	Seq1	26	108	20	76.923	6	23.077
Seq6	Seq1	30	108	18	60.000	18	60.000
Seq2	Seq2	210	210	210	100.000	61	29.048
Seq3	Seq2	210	210	61	29.048	210	100.000
Seq4	Seq2	210	210	48	22.857	48	22.857
Seq5	Seq2	26	210	20	76.923	20	76.923
Seq6	Seq2	30	210	18	60.000	18	60.000
Seq3	Seq3	210	210	210	100.000	61	29.048
Seq4	Seq3	210	210	48	22.857	48	22.857
Seq5	Seq3	26	210	20	76.923	20	76.923
Seq6	Seq3	30	210	18	60.000	18	60.000
Seq4	Seq4	210	210	210	100.000	34	16.19
Seq5	Seq4	26	210	20	76.923	18	69.231
Seq6	Seq4	30	210	18	60.000	18	60.000
Seq5	Seq5	26	26	26	100.000	6	23.077
Seq6	Seq5	30	26	14	53.846	14	53.846
Seq6	Seq6	30	30	30	100.000	30	100.000

```
Used commands Supplemental Table 4*:
python flexidot.py -i test-seqs.fas -p 2 -D y -f 0 -t y -k 3 -w y -r y -x n
*every FlexiDot command in plotting mode 1 or 2 generates an LCS table.
```

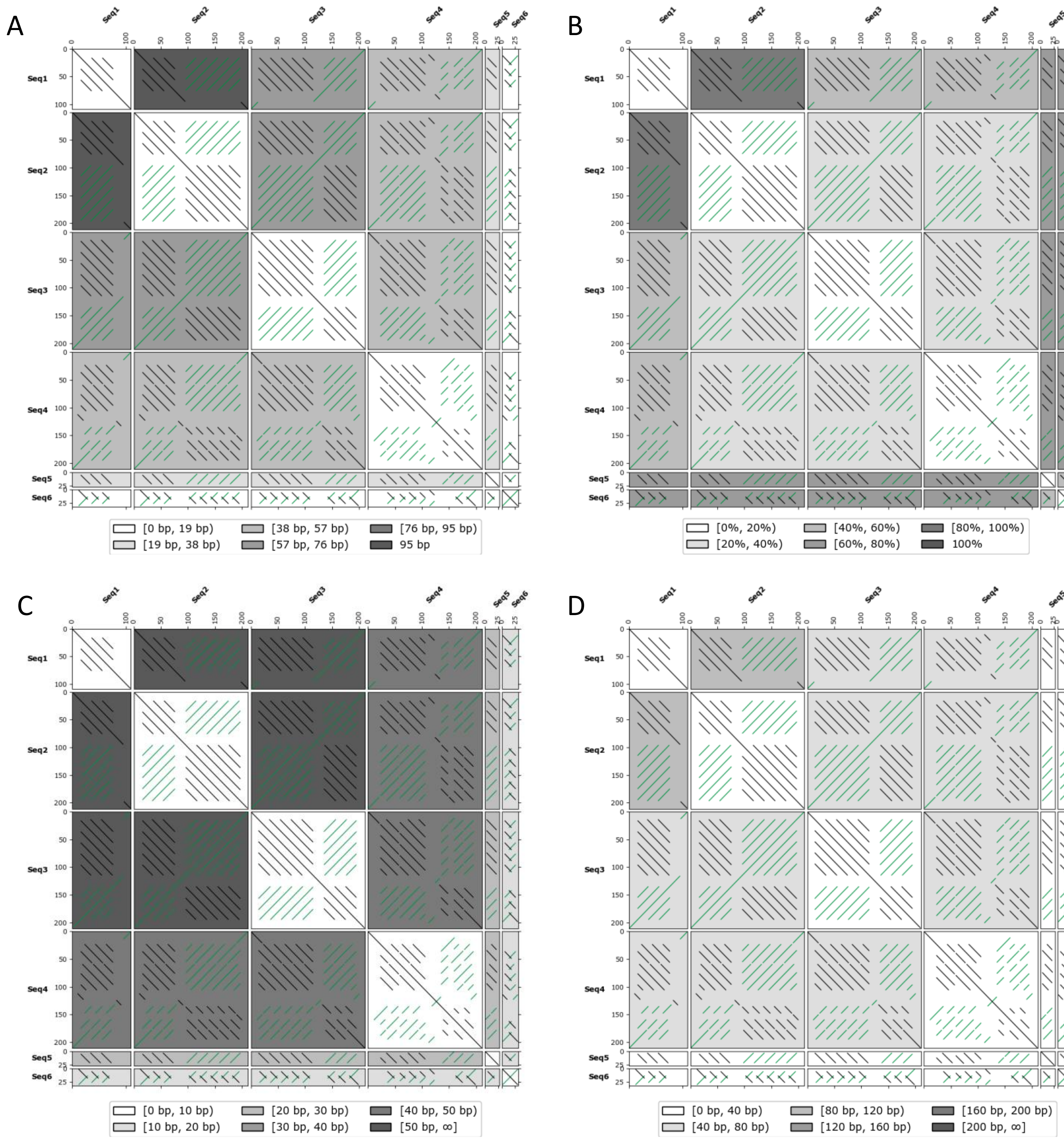
2.3.3 FlexiDot all-against-all similarity shading based on the Longest Common Subsequence

In all-against-all mode, FlexiDot analyzes each pair from the input sequence set. As an indicator of similarity, FlexiDot offers dotplot shading based on the LCS length in all-against-all comparisons (switched on/off with `option -x/--lcs_shading`). Longer matches are represented by darker background color. A separate shading legend file is created, written according to mathematical interval notation, where interval boundaries are represented by a pair of numbers. Consequently, the symbols "(" or ")" represent exclusion, whereas "[" or "]" represent inclusion of the respective number.

FlexiDot similarity shading is highly customizable with the following parameters:

- Reference for shading (option `-y/--lcs_shading_ref`, Supplemental Figure 7): The similarity shading can be based on the LCS length compared to
    - the overall longest LCS length for the given input sequences (option `-y 0`, Supplemental Figure 7 panel A)
    - the longest possible LCS length for each sequence pair, i.e. the length of the shortest input sequence in pairwise comparisons (option `-y 1`, Supplemental Figure 7 panel B)
    - manually defined length intervals (in bp or aa) (option `-y 2`, which is specified by `-Y/--lcs_shading_interval_len`, Supplemental Figure 7 panel C and D).
- Number of shading intervals (option `-X/--lcs_shading_num`, Supplemental Figure 8)
- Considered sequence orientation (for DNA only, option `-z/--lcs_shading_ori`, Supplemental Figure 9);
    - only forward sequence similarity (symmetric shading, option `-z 0`, Supplemental Figure 9 panel A)
    - only reverse complement sequence similarity (symmetric shading, option `-z 1`, Supplemental Figure 9 panel B)
    - both forward and reverse complement sequence similarity (asymmetric shading, option `-z 2`, Supplemental Figure 9 panel C)

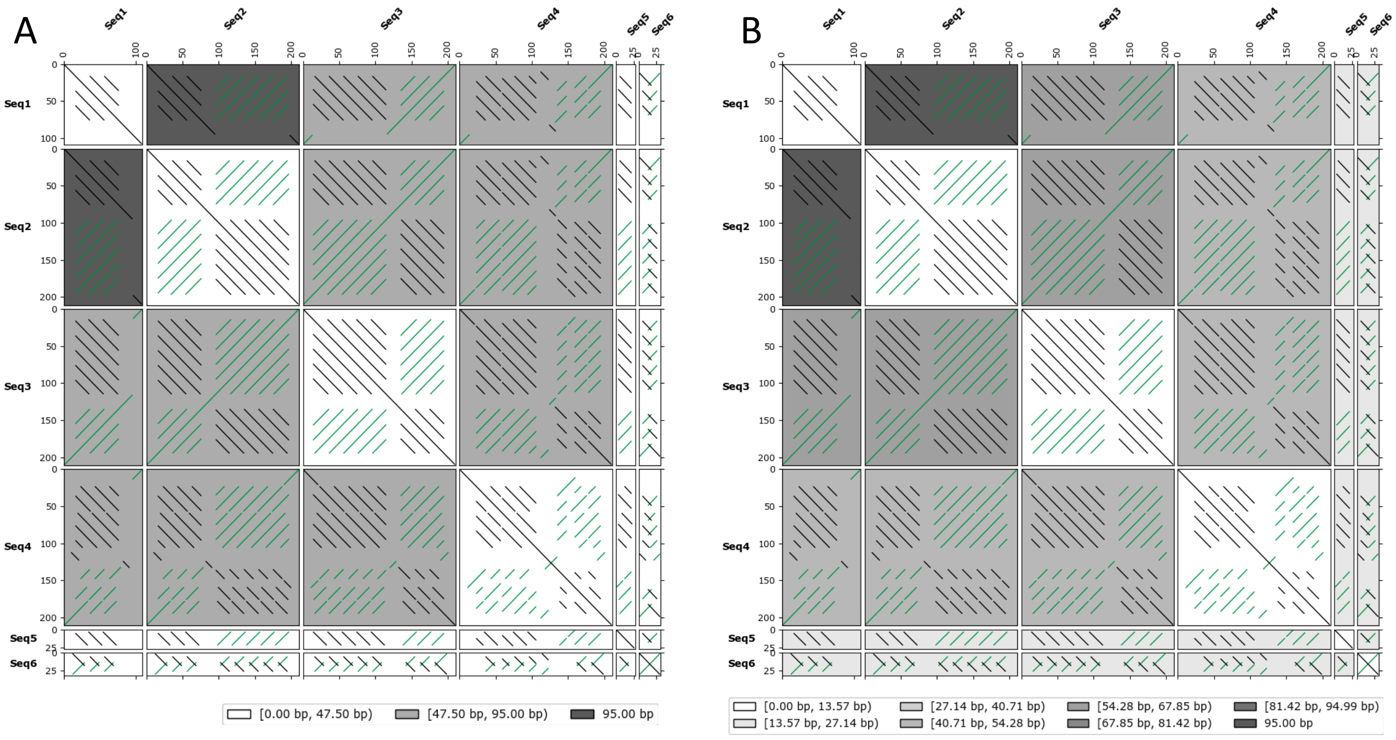
Recommendation for reference choice: The overall longest LCS length (`-y 0`) facilitates the identification of the sequence pairs with the longest matches. If sequences differ in length, the shading relative to the sequence lengths (`-y 1`) is helpful. If multiple all-against-all dotplots are generated, standardized shading using length intervals (`-y 2`) enables comparability.



Supplemental Figure 7: Impact of the reference on shading patterns. If the overall longest LCS is used as reference, the tile with the longest LCS has the darkest shade, here the comparison of Seq1 and Seq2 (LCS=95, panel A). If we compare LCS length relative to the input sequence length, the shading intensity is increased for the shorter sequences Seq5 and Seq6 (panel B). User-defined intervals of 10 and 40 bp affect the coloring pattern and intensities (panels C, D).

Used commands Supplemental Figure 7:

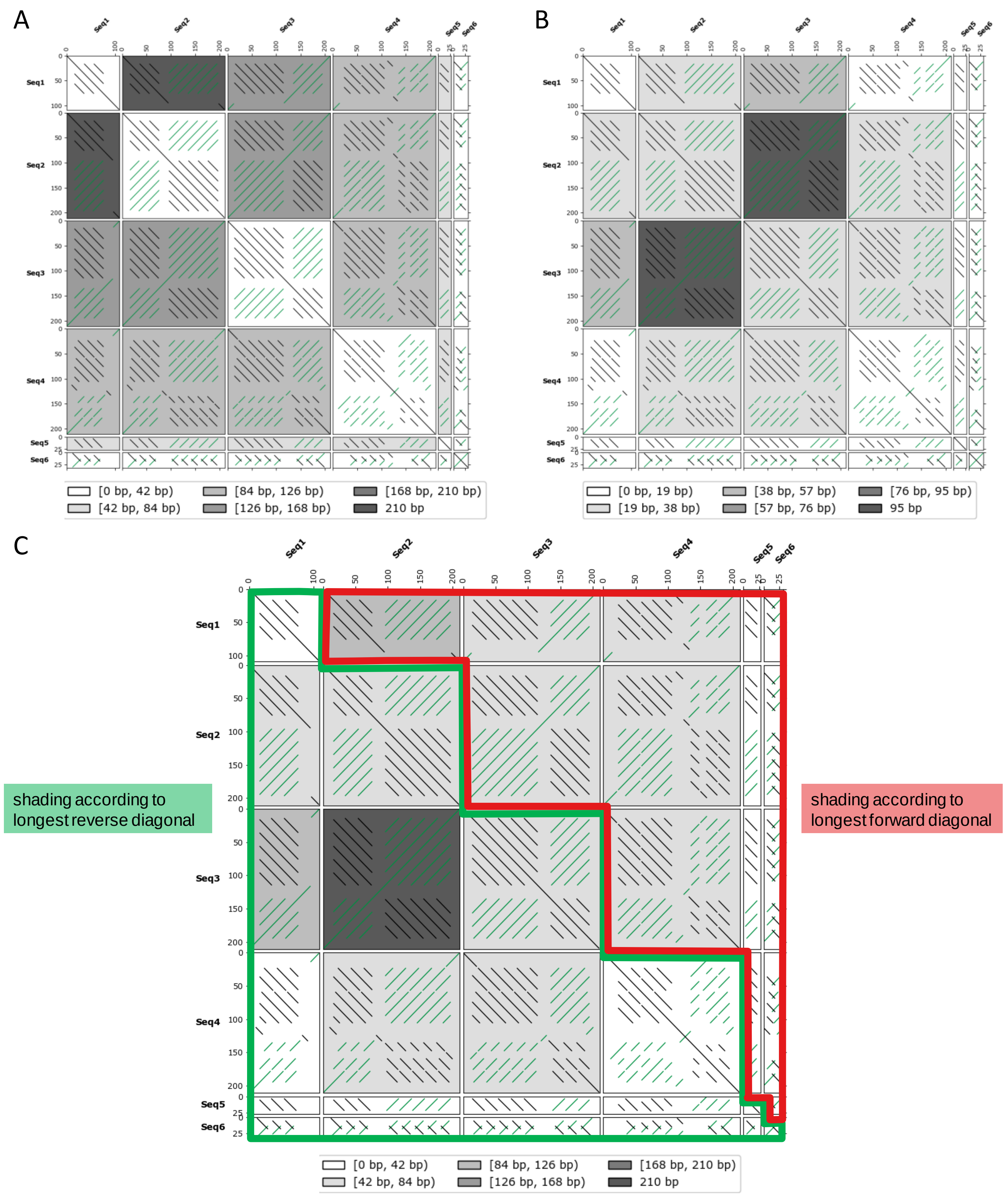
```
Panel A>> python flexidot.py -i test-seqs.fas -p 2 -D y -f 0 -t y -k 10 -w y –X 6 -x y -y 0

Panel B>> python flexidot.py -i test-seqs.fas -p 2 -D y -f 0 -t y -k 10 -w y -X 6 -x y -y 1

Panel C>> python flexidot.py -i test-seqs.fas -p 2 -D y -f 0 -t y -k 10 -w y -X 6 -x y -y 2 -Y 10

Panel D>> python flexidot.py -i test-seqs.fas -p 2 -D y -f 0 -t y -k 10 -w y -X 6 -x y -y 2 -Y 40
```



Supplemental Figure 8: Impact of the shading interval number on the final shading: For shading based on longest forward LCS length, the shade number reflects the number of color intervals defined (A: three intervals; (B): eight intervals). More intervals allow better differentiation between multiple sequences.

```
Used commands Supplemental Figure 8:
Panel A>> python flexidot.py -i test-seqs.fas -p 2 -D y -f 0 -t y -k 10 -r y -x y -y 0 -X 3
Panel B>> python flexidot.py -i test-seqs.fas -p 2 -D y -f 0 -t y -k 10 -r y -x y -y 0 -X 8
```

```
Used commands Supplemental Figure 9:
Panel A>> python flexidot.py -i test-seqs.fas -p 2 -D y -f 0 -t y -k 10 -w n -X 6 -x y -y 0 -z 0
Panel B>> python flexidot.py -i test-seqs.fas -p 2 -D y -f 0 -t y -k 10 -w n -X 6 -x y -y 0 -z 1
Panel C>> python flexidot.py -i test-seqs.fas -p 2 -D y -f 0 -t y -k 10 -w n -X 6 -x y -y 0 -z 2
```

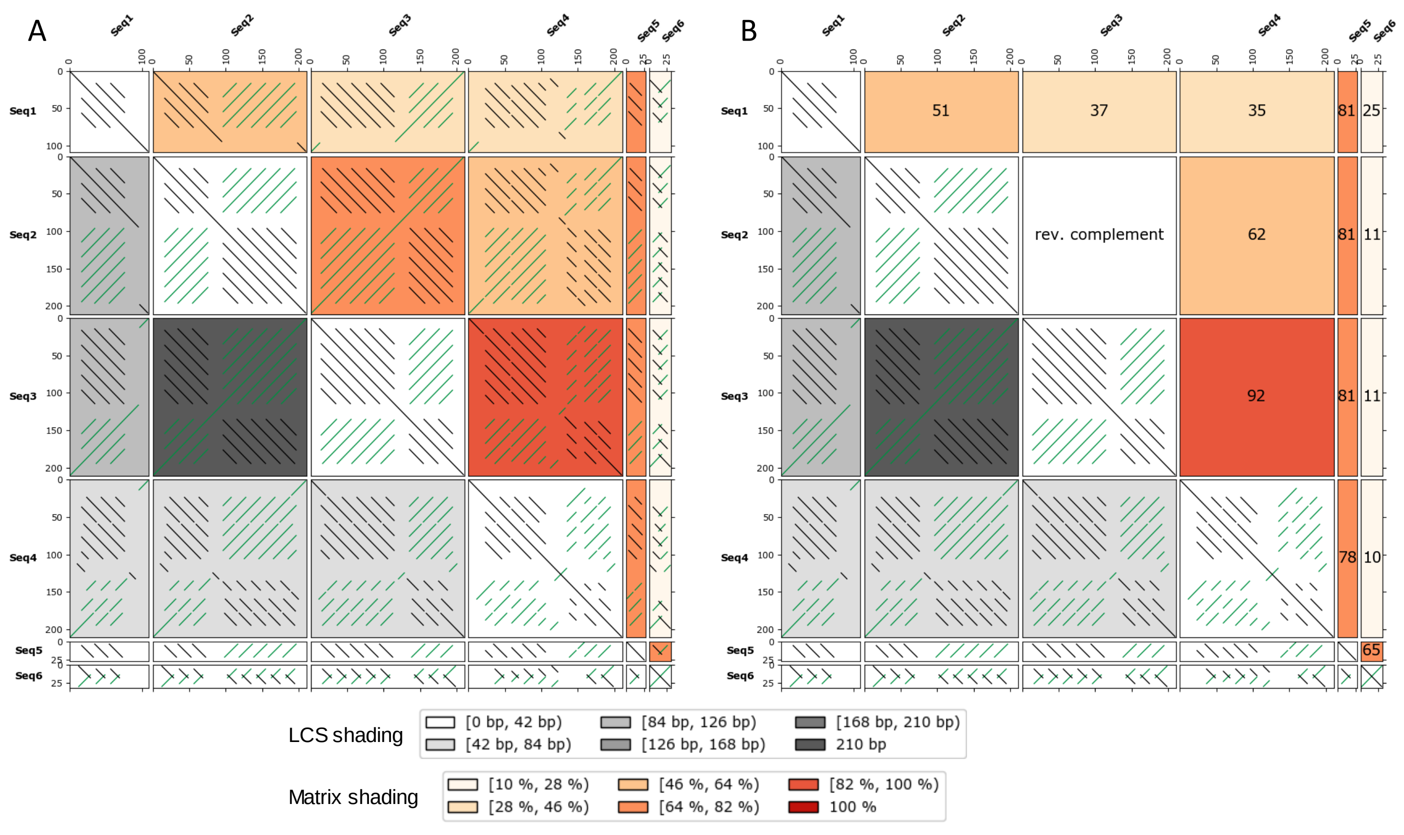


Supplemental Figure 9: FlexiDot shading is possible in symmetric forward (A), or symmetric reverse mode (B). Alternatively, forward and reverse LCS shading can be combined (C). In asymmetric shading, the diagonal is always shaded according to the reverse LCS. Please note that the overall longest match occurs in reverse comparison. All shades are scaled in reference to this longest match.

2.3.4 FlexiDot all-against-all matrix shading

In order to expand customization possibilities, we implemented matrix-based shading. The user can provide a tab-delimited or comma-separated matrix file via the command `-u/--input_user_matrix_file`. If the matrix contains numbers, the number range is translated into shading intervals and a separate legend figure file is created (Supplemental Figure 10 A). It is possible to complement matrix shading with LCS shading. In the example illustrated in Supplemental Figure 10, a matrix of pairwise sequence identities from a multiple sequence alignment is used together with the LCS shading. Please note, that the behavior of LCS shading considering both orientations (`-z/--lcs_shading_ori 2`) is changed. As matrix shading is applied in the top right panels, a combined LCS representation for both orientations is used. In detail, forward and reverse LCS are calculated, but only the best is used for shading (Supplemental Figure 10 A, B).

With the additional command `-U/--user_matrix_print y` the matrix content will be printed and replaces the upper right dotplots (Supplemental Figure 10 B). If the matrix contains strings, the corresponding field is not shaded, but the matrix content can be displayed (see comparison of Seq2 and Seq3 in Supplemental Figure 10 B).



Supplemental Figure 10: All-against-all dotplots with a combination of LCS- and matrix-based shading. The bottom left part of the dotplot is shaded according to the longest overall LCS (comparison of Seq2 and Seq3), whereas the upper part is shaded according to the overall highest sequence similarity provided as matrix (comparison of Seq3 and Seq4). The upper part of the visualization can contain the dotplots (A), or the matrix values (B). If strings are used, shading is not possible, but the string can be printed (see comparison of Seq2 and Seq3 in panel B).

Used commands Supplemental Figure 10:

```
Panel A>> python flexidot.py -i test-seqs.fas -p 2 -D y -f 2 -t y -k 10 -w y -r y -x y -y 0 -z 2 -u custom_matrix.txt -U n

Panel B>> python flexidot.py -i test-seqs.fas -p 2 -D y -f 2 -t y -k 10 -w y -r y -x y -y 0 -z 2 -u custom_matrix.txt -U y
```

Custom matrix* Supplemental Figure 10:

* FlexiDot uses a tabulator as delimiter.

Panel A>>

	Seq1	Seq2	Seq3	Seq4	Seq5	Seq6
Seq1	100					
Seq2	51	100				
Seq3	37	66	100			
Seq4	35	62	92	100		
Seq5	81	81	81	78	100	
Seq6	25	11	11	10	65	100

Panel B>>

	Seq1	Seq2	Seq3	Seq4	Seq5	Seq6
Seq1	100					
Seq2	51	100				
Seq3	37	reverse complement	100			
Seq4	35	62	92	100		
Seq5	81	81	81	78	100	
Seq6	25	11	11	10	65	100

3 Test of commonly used dotplot software

We compare FlexiDot performance with Dotmatcher, Dotter, Dottup, Gepard, and the YASS webserver (Sonnhammer and Durbin 1995, Rice et al. 2000, Noé et al. 2005, Krumsiek et al. 2007, listed in Table 1, main manuscript), using the six test sequences introduced in chapter 1.2. The output of the tools is assessed in their pairwise (section 3.1) and all-against-all functionality (section 3.2).

3.1 Single dotplot functionality

Dotplots for pairwise sequence comparisons can be created using FlexiDot, Dotmatcher, Dotter, Dottup, Gepard and YASS (Sonnhammer and Durbin 1995, Rice et al. 2000, Krumsiek et al. 2007) (Supplemental Figure 11). The input sequences contain forward and reverse matches as well as sequence ambiguities.

Forward and reverse complimentary matches: All tools recognize forward matches, whereas reverse diagonals are recognized by four tools only (FlexiDot, Dotter, Gepard and YASS). In contrast, reverse diagonals are absent for Dotmatcher and Dotter (Supplemental Figure 11 C and E). Surprisingly, Dotmatcher shows additional forward diagonals in the region with the reverse matches (Supplemental Figure 11 C).

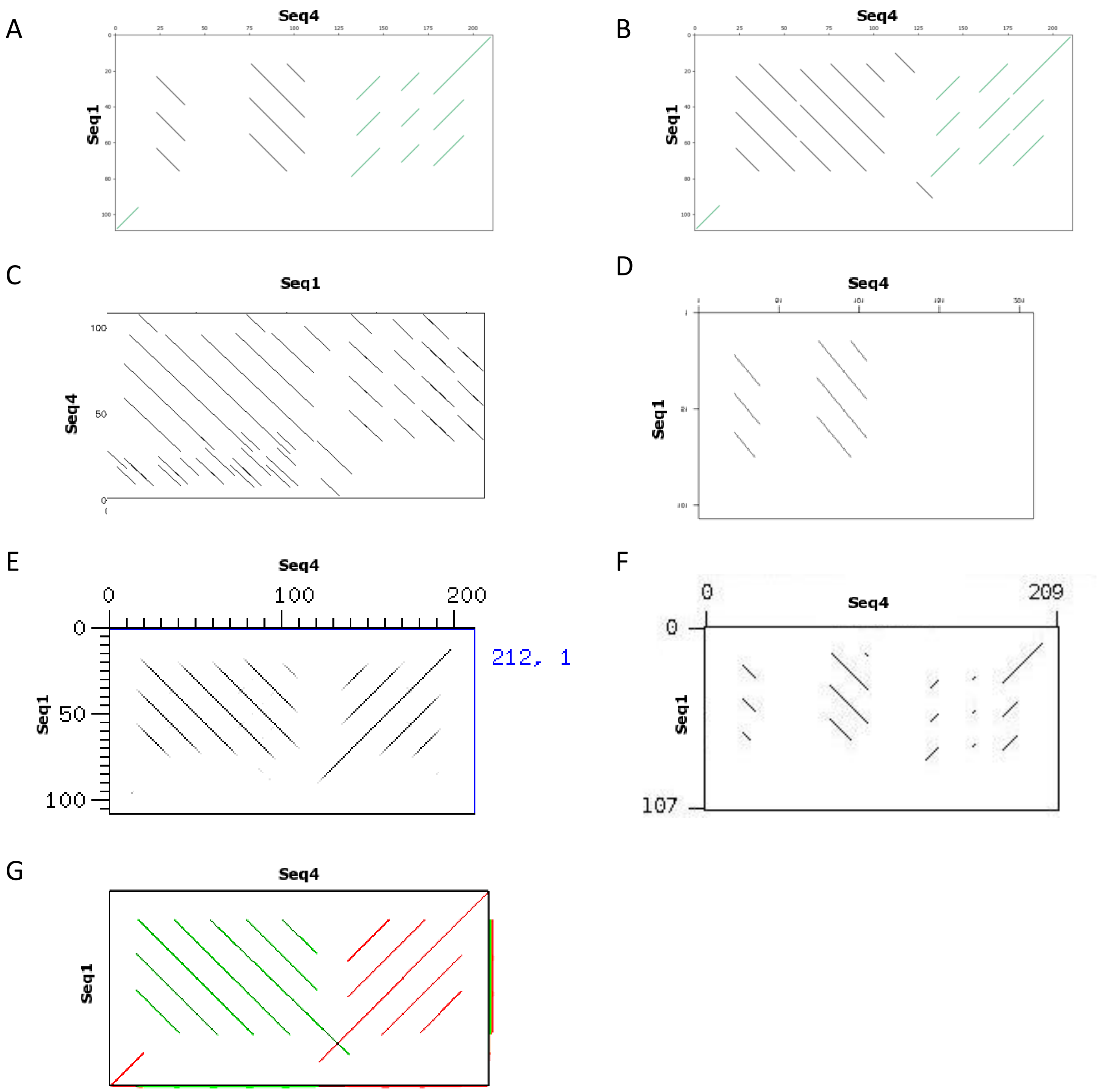
Match lengths: With FlexiDot's sequence ambiguity and mismatch handling (Supplemental Figure 11 B), longer matches are detected. Sequence ambiguities are only recognized by FlexiDot and YASS. Due to their fuzzy matching algorithm, Dotmatcher and Dotter (Supplemental Figure 11 C, E) find nearly as long matches as FlexiDot does in the ambiguity mode. Fuzzy matching as implemented in YASS leads to the longest diagonals (Supplemental Figure 11 G)

Labeling: Unexpectedly, Dotmatcher switches the axis labels.

Reproducibility: FlexiDot, Dotmatcher, Dottup, Gepard and YASS dotplots are reproducible with the given parameters. Dotter provides a user-friendly, but intransparent tuning tool to regulate match stringency/intensity. Hidden parameterization limits Dotter's reproducibility.

Interactive graphical user interface (GUI): Only Dotter, Gepard and YASS provide interactive GUIs. Clicking on the diagonals toggles display of the underlying sequence alignment.

Output formats: Dotmatcher, Dottup and FlexiDot support raster and vector images. In addition, FlexiDot allows collage outputs and annotation-based shading. Gepard also allows to integrate functional annotations. Gepard and YASS solely support raster graphics whereas for Dotter output image generation is only possible by screenshot or by installation of the JDotter java wrapper (Brodie et al. 2004).



Supplemental Figure 11: Pairwise sequence comparisons by different dotplot tools. (A) FlexiDot without ambiguity handling, (B) FlexiDot with ambiguity handling, (C) Dotmatcher, (D) Dottup, (E) Dotter, (F) Gepard, (G) YASS. Output of Dotmatcher and Dottup had to be mirrored for comparability (C, D). For better visibility, axes labels were manually adjusted (E).

3.2 All-against-all functionality

FlexiDot, Dotter, Gepard, and PolyDot can be used to generate all-against-all dotplots (Supplemental Figure 12).

Forward and reverse complimentary matches: Forward and reverse complementary matches are identified by FlexiDot, Dotter, and Gepard, whereas PolyDot lacks the reverse complementary information.

Match lengths: Since FlexiDot recognizes base ambiguities and can tolerate substitutions, it detects longer stretches of similarities than Gepard and PolyDot. Due to fuzzy matching, Dotter finds nearly as long matches (see chapter 3.1).

Shading options: FlexiDot evaluates sequence similarity and translates the length of the longest shared subsequence into differential shading (Supplemental Figure 12 panel A-C). Here, above the diagonal of the dotplot shading is based on the longest forward match, whereas for self comparisons and for pairwise comparisons below the diagonal, shading reflects the longest reverse complementary match. Matrix shading further expands the set of dotplot annotation tools.

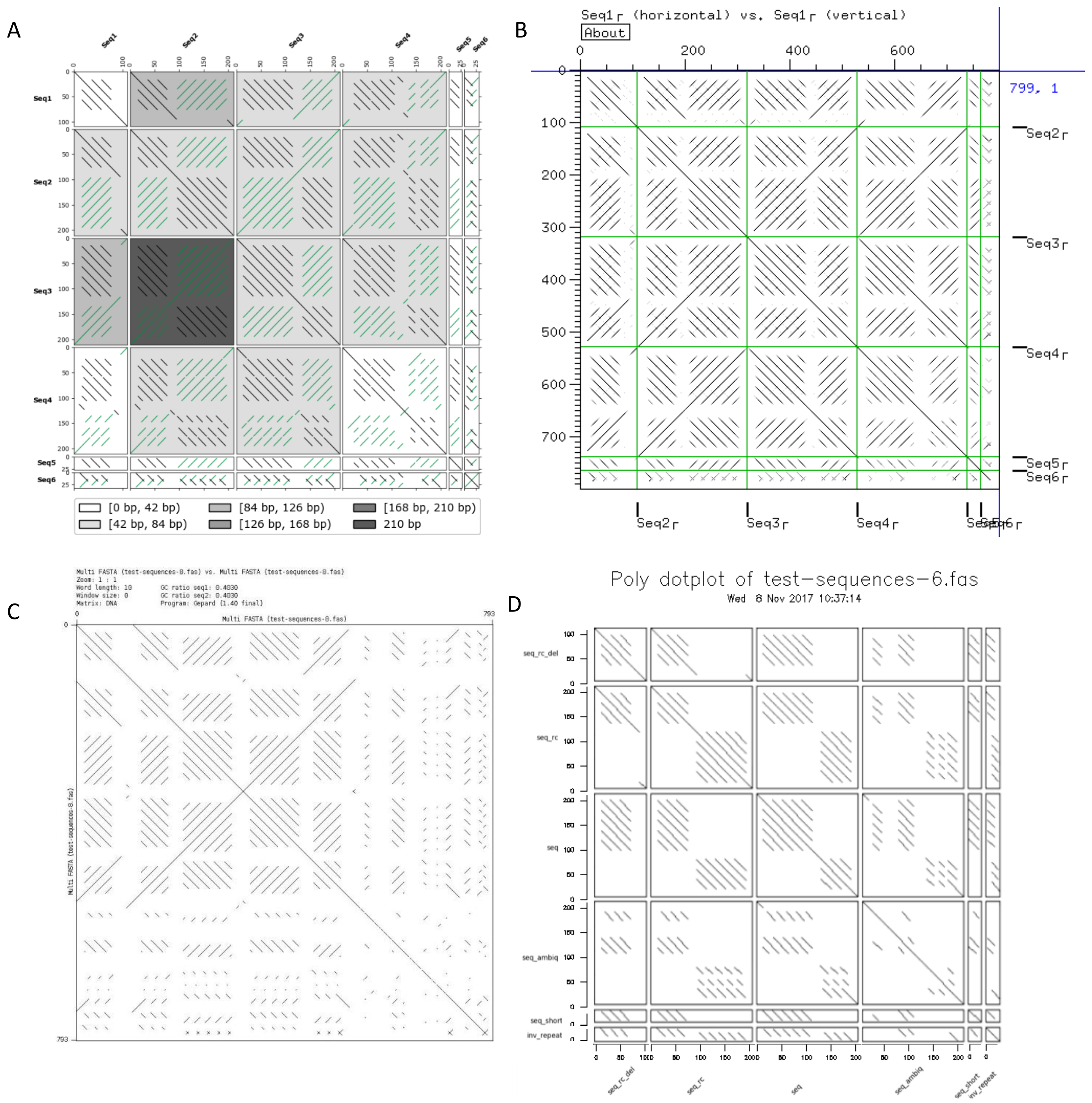
Reproducibility: FlexiDot, PolyDot, and Gepard dotplots are reproducible with the given parameters. As mentioned, hidden parameters limit Dotter's dotplot reproducibility.

Labeling: In contrast to the other tools, Gepard does not indicate sequence borders.

Output formats: FlexiDot and PolyDot allow raster and vector outputs. Gepard supports only raster graphics, and Dotter output can only be saved using screenshots.

3.3 Conclusion

In sum, only FlexiDot combines the visualization of informative forward and reverse diagonals with suitable output formats. Extra features (such as collages, annotation shading, and ambiguity consideration) allow further customization. This enables a clear, reproducible visualization of internal sequence structures and shared motifs or regions.



```
Multi FASTA (test-sequences-8.fas) vs. Multi FASTA (test-sequences-8.fas)
Zoom: 1 : 1
Word length: 10        GC ratio seq1: 0.4030
Window size: 0         GC ratio seq2: 0.4030
Matrix: DNA            Program: Gepard (1.40 final)
```

Supplemental Figure 12: All-against-all view of pairwise and self comparisons of different tools. (A) FlexiDot, (B) Dotter, (C) Gepard, (D) PolyDot. Output of PolyDot had to be rotated for comparability (D).

4 Application use cases

In order to show FlexiDot application areas, we emulate studies referenced in Supplemental Table 5 and show how FlexiDot performs with the chosen datasets. For use cases #1a and #1b, we create self dotplots of long, error-prone reads generated by single molecule real-time sequencing (SMRT) to investigate organization of coding (use case #1a) and non-coding (use case #1b) tandem repeats as reported in recent studies (Sevim et al. 2016, Symonova et al. 2017). Use case #2 focuses on repetitive transposable elements. Firstly, we illustrate their structural motifs by combining a dotplot with functional annotations (use case #2a). Secondly, FlexiDot's all-against-all mode is used to compare consensus representations of repetitive sequences across retroelement families (use case #2b) and different species (use case #2c). The LCS shading feature and a complementary shading according to a multiple sequence alignment similarity matrix is exemplified. Use case #3 illustrates the application of FlexiDot on amino acid sequences for filamin repeat containing proteins (use case #3a) and nucleotide binding leucine-rich repeat proteins (use case #3b).

Supplemental Table 5: Use case overview

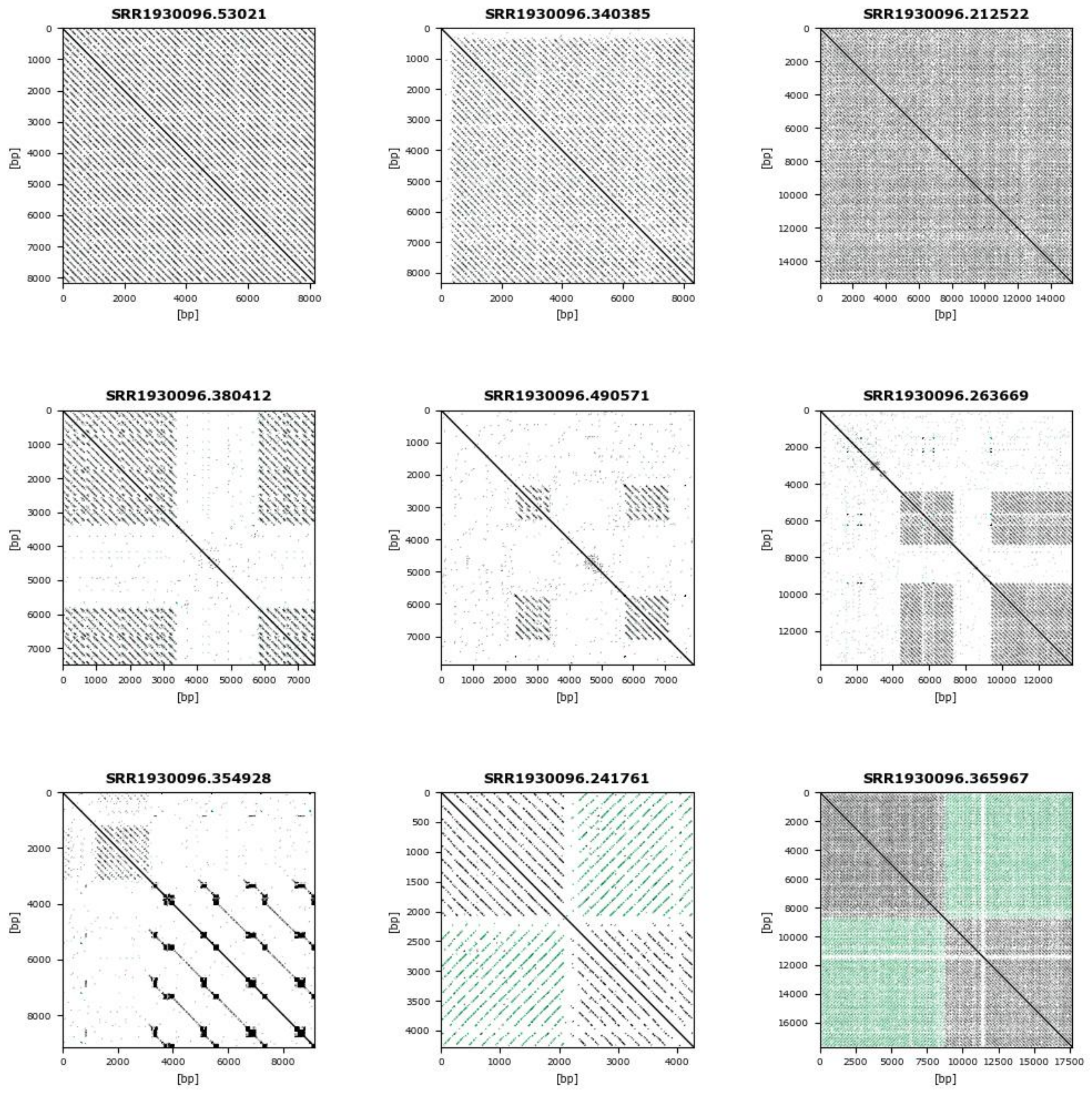
Use case	Title	Referred study	Accession numbers
#1a	SMRT reads for structural analysis of pike 5S rDNA arrays	Symonova et al. 2017	NCBI SRA: SRR1930096
#1b	SMRT reads for alpha satellite higher order arrangements	Sevim et al. 2016	NCBI SRA: SRR4016879
#2a	Combined depiction of retrotransposon structure and domain annotations	Weber et al. 2013	paper supplement
#2b	Similarity of long terminal repeat retrotransposons	Weber et al. 2013	paper supplement
#2c	Comparison of degenerated consensus sequences across genomes (example sugar beet SINE consensuses)	Schwichtenberg et al. 2016	paper supplement
#3a	Similarity of eight filamin-containing proteins	PFAM download	PFAM: I3JPF4, R7VQ48, U4UN69, A0A164QD41, W5K5V4, W5NZD8, U3KF39, A0A0Q3X9J3
#3b	Similarity of six nucleotide-binding leucine-rich repeat proteins in Solanaceae	PFAM download	PFAM: M1C459, M1BKT1, M1BKT0, M1CR22, K4BXZ6, M1B1V4

4.1 Use case #1: Structural organization of tandem repeat arrays in SMRT reads

FlexiDot is able to generate self dotplot collages of multiple input sequences (plotting mode 0). This feature is especially useful for investigation of potential tandem repeat higher order structures. According to two recent studies, we downloaded SMRT read data and used BLAST to preselect 250 SMRT read sequences with similarities to a representative query. Nine FlexiDot self dotplots are chosen for illustration of structural patterns.

4.1.1 Use case #1a: pike 5S rDNA arrays

Based on the study by Symonová et al. (2017), 5S rDNA-containing SMRT reads (NCBI SRR1930096) were selected with a pike 5S rDNA clone (NCBI KX950799). FlexiDot generates self dotplots of each read and arranges them as a collage for visual inspection. The dotplots of nine representative SMRT reads (Supplemental Figure 13) allow to categorize the 5S rDNA arrays according to their organization, such as "continuous arrays" (first row), “array interruptions” (second row), “association with other repeats” (third row dotplot 1) and “array inversions” (third row dotplot 2 and 3).



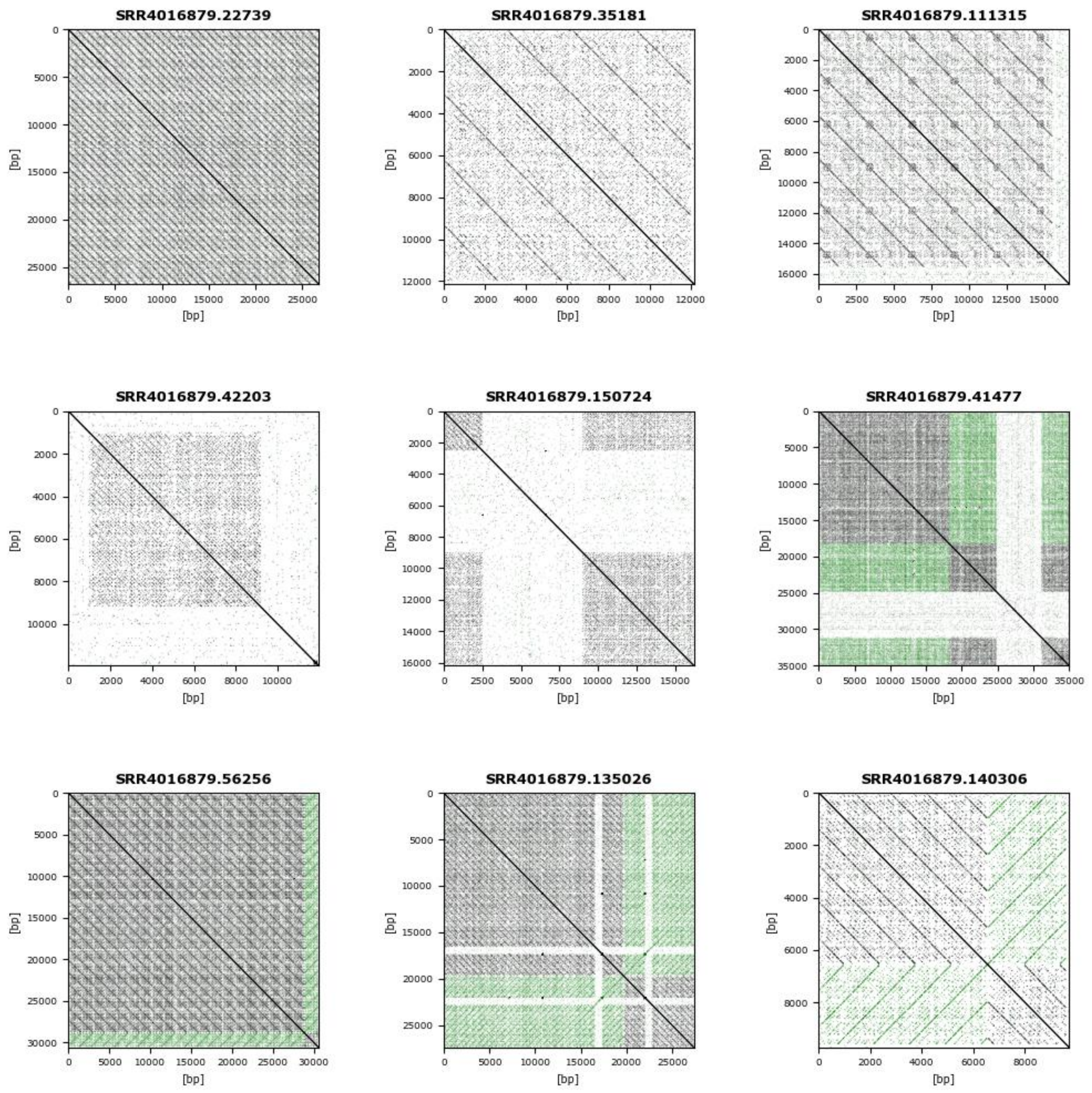
Supplemental Figure 13: FlexiDot self dotplots of nine 5S rDNA-containing pike SMRT reads. FlexiDot has been run in self mode, with a wordsize of 9 bp and a combined 3x3 collage output has been chosen. Dependent on the line pattern, the reads are characterized as “continuous arrays” (first row), “array interruptions” (second row), “association with other repeats” (third row dotplot 1) and “array inversions” (third row dotplot 2 and 3).

Used commands Supplemental Figure 13:

```
python flexidot.py -i Selected_SMRT_reads.fas -m 3 -n 3 -f 0 -k 9 -p 0
```

4.1.2 Use case #1b: Human alpha satellite

Similar to 5S rDNA in use case #1a, non-coding satellite arrays can also be organized in various structures. More often than rDNA arrays, satellite arrays form higher order arrangements in the kilobase scale, easily visualized by dotplots. Similar to Sevim et al. (2016), we preselected human SMRT reads from the SRR4016879 data set with the alpha satellite monomer (NCBI: X07685). We observe four different structural categories of alpha satellite arrays. Supplemental Figure 14 shows examples for each category: “continuous arrays of higher order arrangements” (first row), “short arrays (second row dotplot 1)”, “array interruptions” (second row dotplot 2 and 3), and “array inversions” (second row dotplot 3, third row).



Supplemental Figure 14: FlexiDot self dotplots of nine alpha satellite-containing human SMRT reads. FlexiDot has been run in self mode, with a wordsize of 9 bp and a combined 3x3 collage output has been chosen. Evenly-spaced diagonals indicate conventional arrays. Irregularities in line spacing and intensities indicate higher order structures. Absence of parallel diagonal lines results from interruptions of the arrays. Head-to-head structures are result of monomer orientation changes and are indicated by a color switch (black = forward, green = reverse complement matches).

Used commands Supplemental Figure 14:

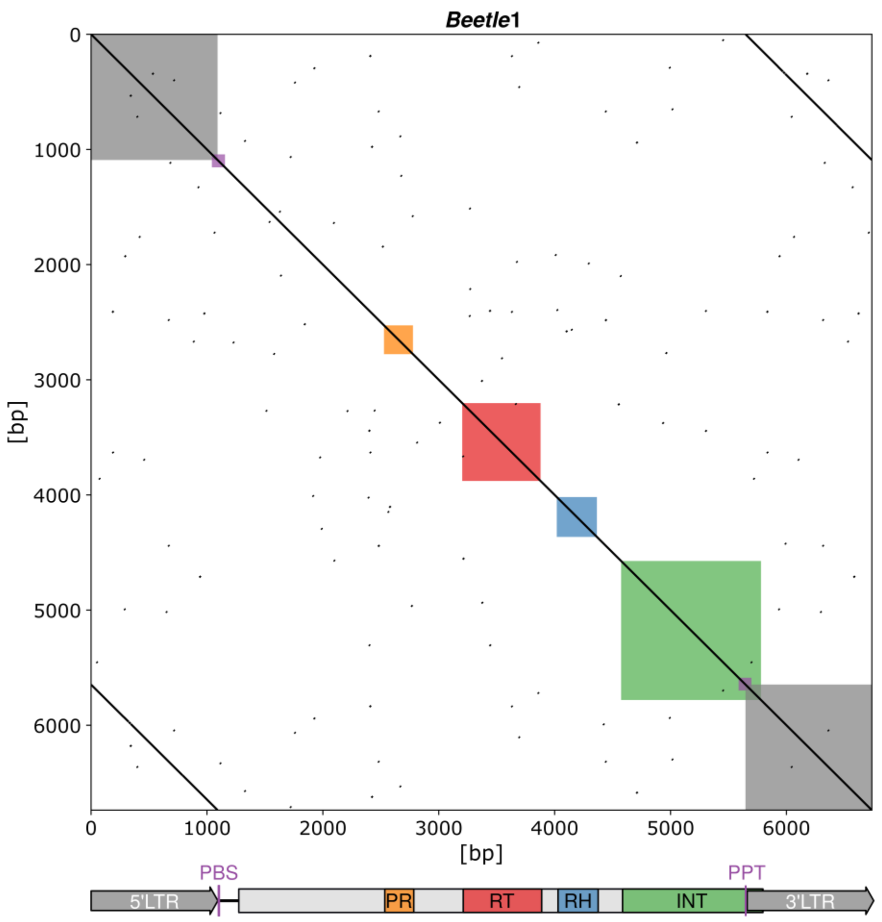
```
python flexidot.py -i Selected_SMRT_reads.fas -m 3 -n 3 -f 0 -k 9 -p 0
```

4.2 Use case #2: Structure and sequence similarity of retrotransposons

FlexiDot is able to combine visualizations of sequence structure and corresponding functional annotations. In addition, it can be used to compare members of a gene or repeat family. Here, we present an example on how to use FlexiDot features to characterize a repeat family. We first focus on different families of *Beetle*-type retrotransposons, illustrating the structure of a single family member (chapter 4.2.1) and then comparing several members to each other (chapter 4.2.2). These sugar and wild beet retrotransposons contain open reading frames and are flanked by long terminal repeats (LTRs, Weber et al. 2009, 2013). In a second approach, we compare non-coding retrotransposons from a single family between different species. Representative, ambiguity-containing consensus sequences are used to demonstrate FlexiDot's ambiguity handling (chapter 4.2.3).

4.2.1 Use case 2a: Combined depiction of *Beetle*1 structure and domain annotations

To get an overview of the structural organization of *Beetle* retrotransposons, a *Beetle*1 self dotplot was generated and FlexiDot's gff-mediated annotation shading was used to mark positions of retrotransposon domains (Supplemental Figure 15).



Supplemental Figure 15: Dotplot (wordsize 10) and schematic representation of the *Beetle*1 retrotransposon. Long terminal repeats are visible as short parallel diagonals in the upper right and lower left corners. Retrotransposon domains are highlighted. Long terminal repeat (LTR , grey), protease (PR, orange), reverse transcriptase (RT, red), ribonuclease H (RH, blue), integrase (INT, green), primer binding site, and polypurine tract (PBS/PPT, violet).

*Beetle*1 is flanked by LTRs, present as additional diagonals in the upper right and lower left corners. Strikingly, the integrase region (INT, green) extends into the LTR (grey) as visible by overlapping green and grey annotations. In order to visualize short sequence stretches, such as PBS and PPT (violet regions), we magnified the annotated region with an additive zoom as defined in the color configuration file `Beetle1_color.config`.

```
Used commands Supplemental Figure 15:

python flexidot.py -i Beetle1.fas -g Beetle1_anno.gff3 -G Beetle1_color.config -p 0 -c n -r n -k
10 -A 1.5 -E 15
```

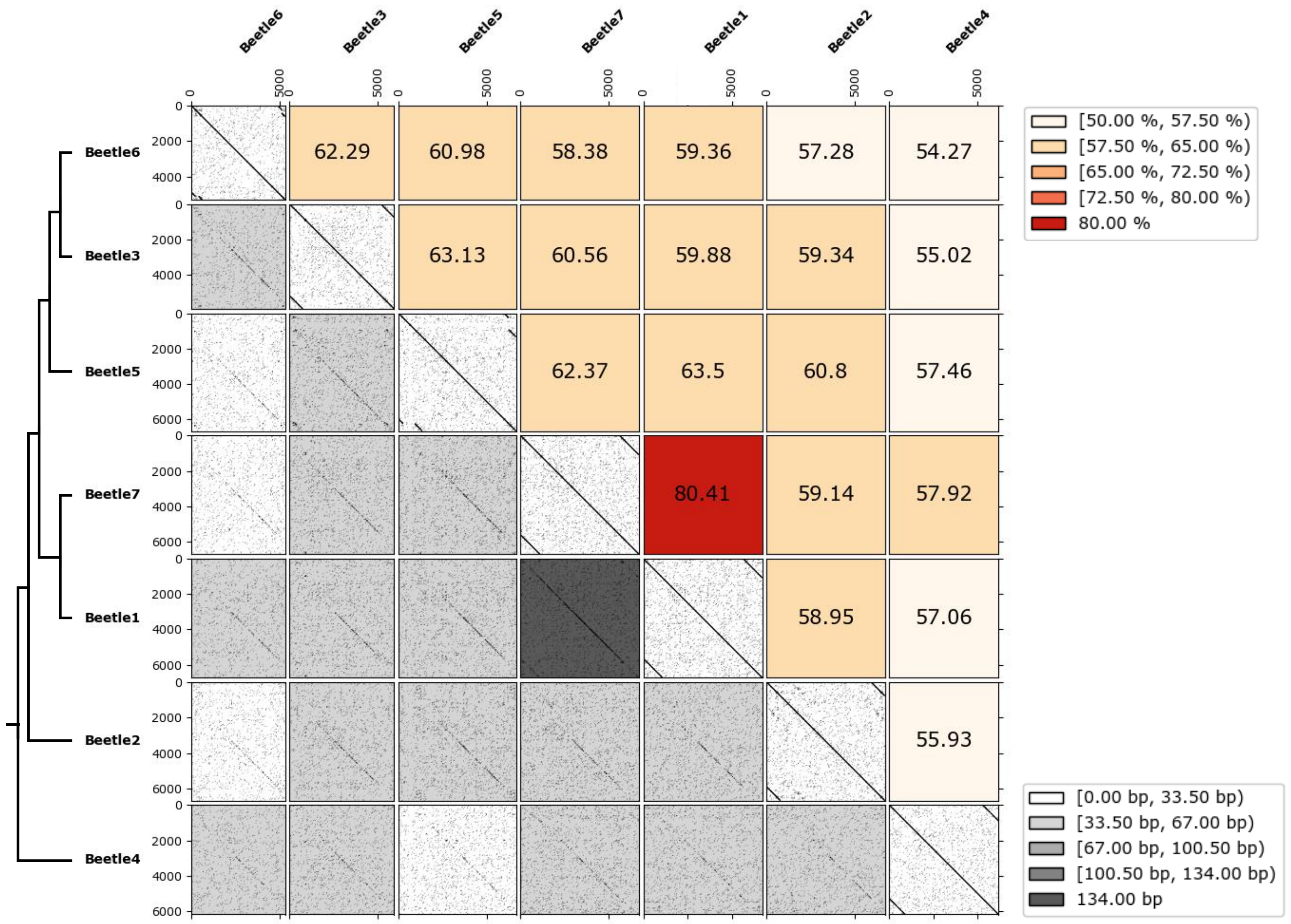
```
Beetle1_anno.gff3 (tab-delimited)
Beetle1      manual_annotations  LTR    0      1091   .      +      .      ID=1
Beetle1      manual_annotations  PBS    1093   1105   .      +      .      ID=2
Beetle1      manual_annotations  PROT   2528   2777   .      +      .      ID=3
Beetle1      manual_annotations  RT     3203   3878   .      +      .      ID=4
Beetle1      manual_annotations  RNH    4019   4364   .      +      .      ID=5
Beetle1      manual_annotations  INT    4574   5780   .      +      .      ID=6
Beetle1      manual_annotations  PPT    5638   5647   .      +      .      ID=7
Beetle1      manual_annotations  LTR    5647   6736   .      +      .      ID=8
```

```
Beetle1_color.config (tab-delimited)
#annotation_type    color        alpha        zoom (linewidth_adjustment)
LTR                 grey         0.7          0
PPT                 #984ea3      0.7          50
PBS                 #984ea3      0.7          50
PROT                #ff7f00      0.7          0
RT                  #e41a1c      0.7          0
RNH                 #377eb8      0.7          0
INT                 #4daf4a      0.7          0
```

4.2.2 Use case 2b: Similarity of *Beetle*-type long terminal repeat retrotransposons

In order to demonstrate usability of FlexiDot for a composite illustration of *Beetle* retrotransposon family relationships, we combined a dually shaded dotplot with a schematic Neighbor Joining tree (Supplemental Figure 16). Using the *pol* regions of representative *Beetle* elements from Weber et al. (2013), we performed a nucleic acid multiple sequence alignment with MAFFT and calculated sequence identity values and the Neighbor Joining tree with MEGA7 (Katoh and Standley, 2013, Tamura et al., 2016).

The flanking LTRs are visible as short diagonals in the top right and the bottom left corners of each self dotplot (arranged on the main diagonal). The continuous LTR diagonals reflect intact LTRs, whereas for *Beetle*5 and *Beetle*6 interruptions illustrate internal deletions. In pairwise comparisons between different *Beetle* copies, matches are mainly restricted to the internal, coding region implying a higher variability in the LTR than in the untranslated regions. Sequence similarity shading according to the LCS (grey) and sequence identity (orange) consistently show that *Beetle*1 and *Beetle*7 share the longest match (dark grey) and the highest sequence identity (dark orange). In particular, *Beetle*1 from *Patellifolia procumbens* and *Beetle*7 from *Beta vulgaris* share considerable sequence identity and are both suggested to play a significant role in the formation of functional centromeres (Weber et al., 2013).



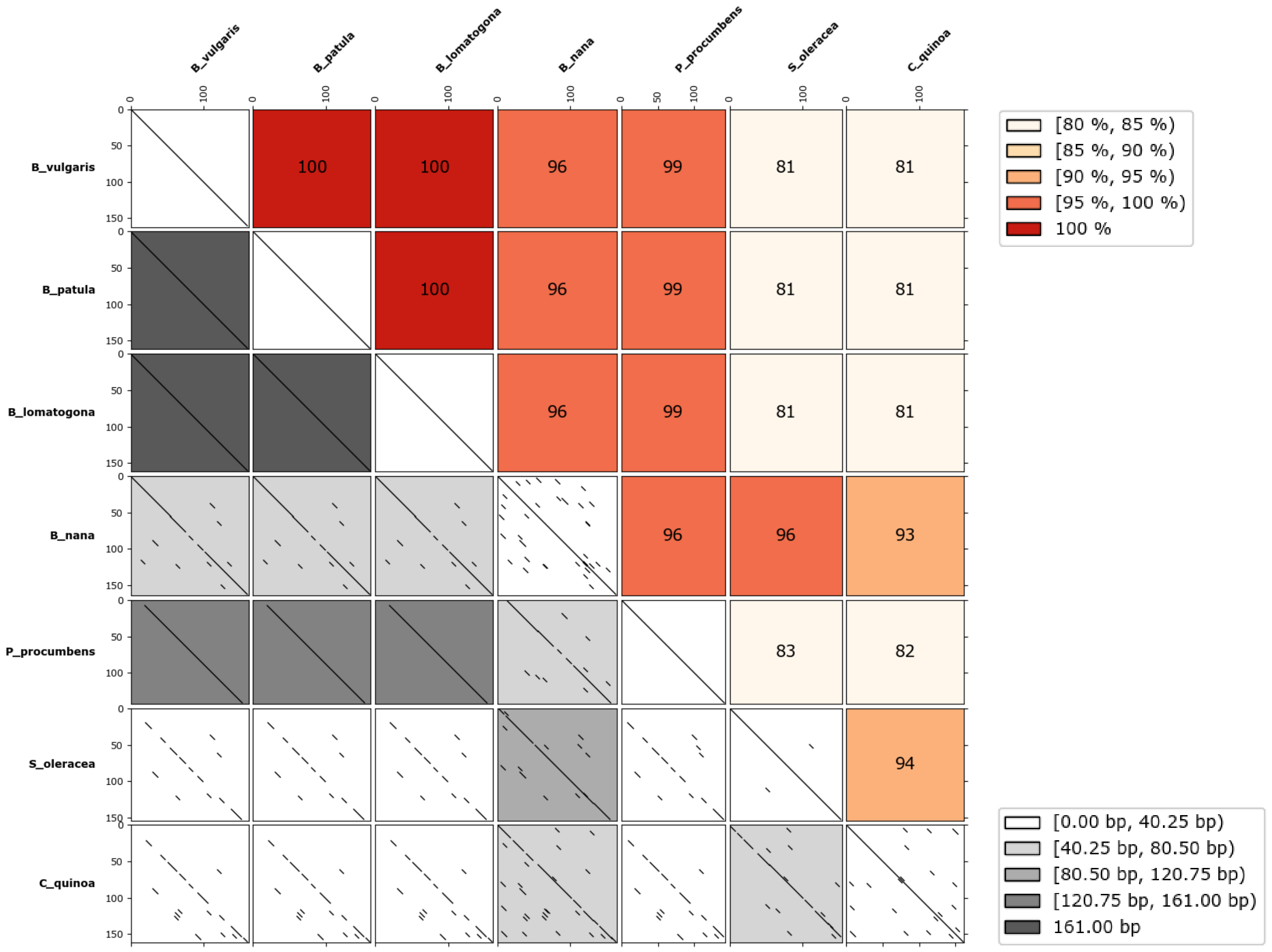
Supplemental Figure 16: Dotplot analysis shows relationship between seven *Beetle* retrotransposon families. A schematic Neighbor Joining tree is combined with shadings according to LCS length (grey) and sequence identity (orange). Pairwise comparison of *Beetle*1 and *Beetle*7 shows both the longest match and the highest sequence identity. All *Beetle* retrotransposons are flanked by long terminal repeats, visible by short diagonals parallel to the main diagonal of each dotplot. On a side note, the *Beetle*5 and *Beetle*6 self dotplot depicts a deletion in the LTR region.

Used commands Supplemental Figure 16:

```
python flexidot.py -i Beetle.fas -p 2 -x y -k 10 -S 1 -r n -u custom_matrix.txt -U y
```

4.2.3 Use case #2c: Comparison of nucleotide consensus sequences in related genomes

Since FlexiDot recognizes base ambiguities (see chapter 2.1), it is possible to use species-specific and ambiguity-containing consensus sequences for dotplot inspection. Exemplarily, we analyzed sequence divergence of evolutionarily old, non-coding repetitive elements. The short interspersed nuclear element (SINE) family AmaS-XIII was described in sugar beet and related Amaranthaceae species by Schwichtenberg et al. 2016 (Supplemental Figure 17). Due to FlexiDot's similarity shading, we easily deduce, that the consensus elements of *Beta vulgaris*, *Beta patula* and *Beta lomatogona* share long sequence stretches, whereas SINEs retrieved from distantly related species (*Chenopodium quinoa* and *Spinacia oleracea)* differ from the *Beta* elements. Also the SINE from *Patellifolia procumbens*, a former member of the genus *Beta*, shows a high similarity to the *Beta* consensus elements. LCS shading was chosen for the dotplots below the main diagonal (grey), whereas shading based on sequence identities has been applied to the upper right part (orange). Consideration of sequence ambiguities increases dotplot sensitivity, thus revealing family similarities between more diverged consensus sequences. LCS lengths are correlated with identity values derived from their multiple sequence alignment.



Supplemental Figure 17: All-against-all dotplot showing sequence similarity of the SINE family AmaS-XIII in seven related Amaranthaceae. Similarity shading highlights that the AmaS-XIII SINEs from *Beta vulgaris*, *Beta patula* and *Beta lomatogana* are highly similar with shading according to the longest LCS in grey and the overall pairwise sequence identity in orange. Interpretation of ambiguities makes similarity in more distantly related genomes visible and leads to high LCS lengths. Dotplots above the diagonal are replaced by pairwise sequence identities.

Used commands Supplemental Figure 17:

```
Panel A>> python flexidot.py -i SINEs.fas -p 2 -k 7 -r n -D y -P 12.5 -y 0 -x y -w n -U n -u
id_matrix.txt

Panel B>> python flexidot.py -i SINEs.fas -p 2 -k 7 -r n -D y -P 12.5 -y 0 -x y -w y -U y -u
id_matrix.txt
```

4.3 Use case #4: FlexiDot amino acid handling

In order to demonstrate FlexiDot's handling of protein sequences, we analyze two protein groups in all-against-all mode. We first compare highly repetitive filamin proteins (chapter 4.3.2) and then analyze different nucleotide-binding leucine-rich proteins (chapter 4.3.1).

4.3.1 Use case #4a: Similarity of eight filamin repeat proteins

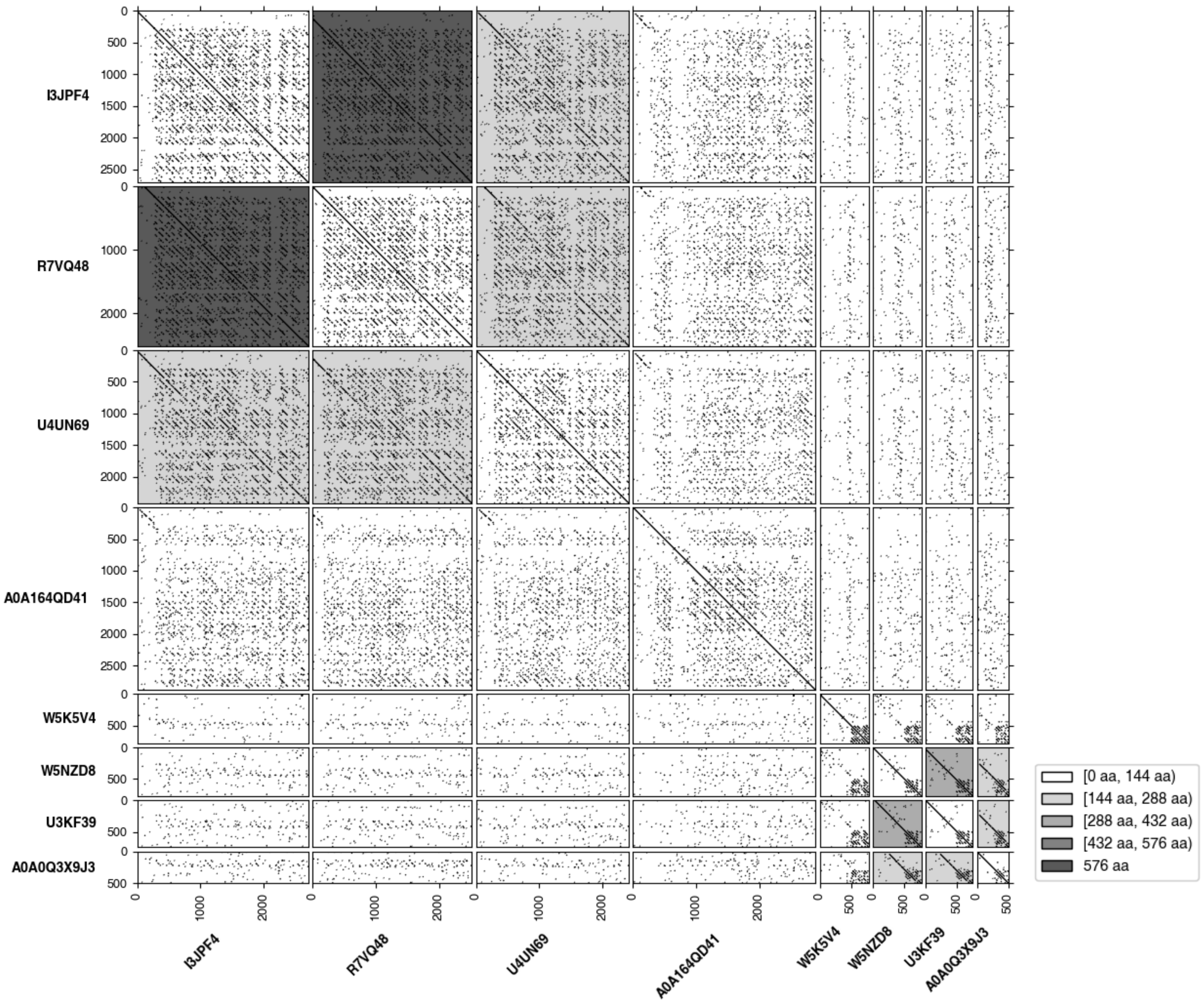
Filamins are eukaryotic proteins belonging to the actin-binding proteins (van der Flier, 2001). They contain a variable number of repeat modules (filamin domains) and are thus ideally suited to demonstrate FlexiDot's handling of amino acid sequences. The protein architecture of eight selected filamin-containing proteins from the PFAM database (Finn et al. 2016) is summarized in Supplemental Table 6. It is noteworthy that the varying structural architectures and differences in repeat numbers complicate filamin protein analysis by multiple sequence alignment. Alternatively, analysis using all-against-all dotplots enables the comparison of all monomeric units with each other and facilitates the observation of patterns otherwise hidden in a standard alignment.

Supplemental Table 6: Filamin-contain proteins used for our comparison

#	Organism	PFAM accession	Architecture
1	*Oreochromis niloticus*	`I3JPF4_ORENI (I3JPF4)`	CH x 2, **Filamin x 23**
2	*Columba livia*	`R7VQ48_COLLI (R7VQ48)`	CH x 1, **Filamin x 23**
3	*Dendroctonus ponderosae*	`U4UN69_DENPD (U4UN69)`	CH x 2, **Filamin x 19**
4	*Daphnia magna*	`A0A164QD41_9CRUS (A0A164QD41)`	CH x 2, **Filamin x 20**
5	*Astyanax mexicanus*	`W5K5V4_ASTMX (W5K5V4)`	zf-B_box x 1, **Filamin x 1**, NHL x 6
6	*Ovis aries*	`W5NZD8_SHEEP (W5NZD8)`	zf-RING_UBOX, zf-B_box, **Filamin x 1**, NHL x 6
7	*Ficedula albicollis*	`U3KF39_FICAL (U3KF39)`	zf-RING_UBOX, zf-B_box, **Filamin x 1**, NHL x 5
8	*Amazona aestiva*	`A0A0Q3X9J3_AMAAE (A0A0Q3X9J3)`	**Filamin x 1**, NHL x 5

Regarding domain architecture, sequences #1 to #4 (I3JPF4, R7VQ48, U4UN69 and A0A164QD41) and #5 to #8 (W5K5V4, W5NZD8, U3KF39 and A0A0Q3X9J3) are similar, respectively. The first four sequences contain one or two CH-domains and 19-23 filamin domains. This is visualized by the overall dotplot organization (Supplemental Figure 18), where multiple filamin domains are represented by parallel lines shared by sequences #1 through #4.

In addition to a single filamin domain, sequences #5 to #8 contain repetitive NHL domains, visible as filamin-unrelated repeated structures in the dotplot. The filamin monomer in the second sequence group is visible as a thin band of short parallel lines shared with the sequences of the first group. The highest sequence similarities occur between I3JPF4 and R7VQ48 as well as W5NZD8 and U3KF39, respectively, as indicated by LCS similarity shading.



Supplemental Figure 18: Eight filamin-containing proteins are compared by all-against-all dotplots. The repetitive filamin region is visible in the dotplots as parallel lines in pairwise sequence comparison of the first four sequences (I3JPF4, R7VQ48, U4UN69 and A0A164QD41). The remaining four sequences (W5K5V4, W5NZD8, U3KF39 and A0A0Q3X9J3) contain a single filamin domain, only, as well as multiple, filamin-unrelated NHL domains visible as additional repeat patterns in pairwise comparison.

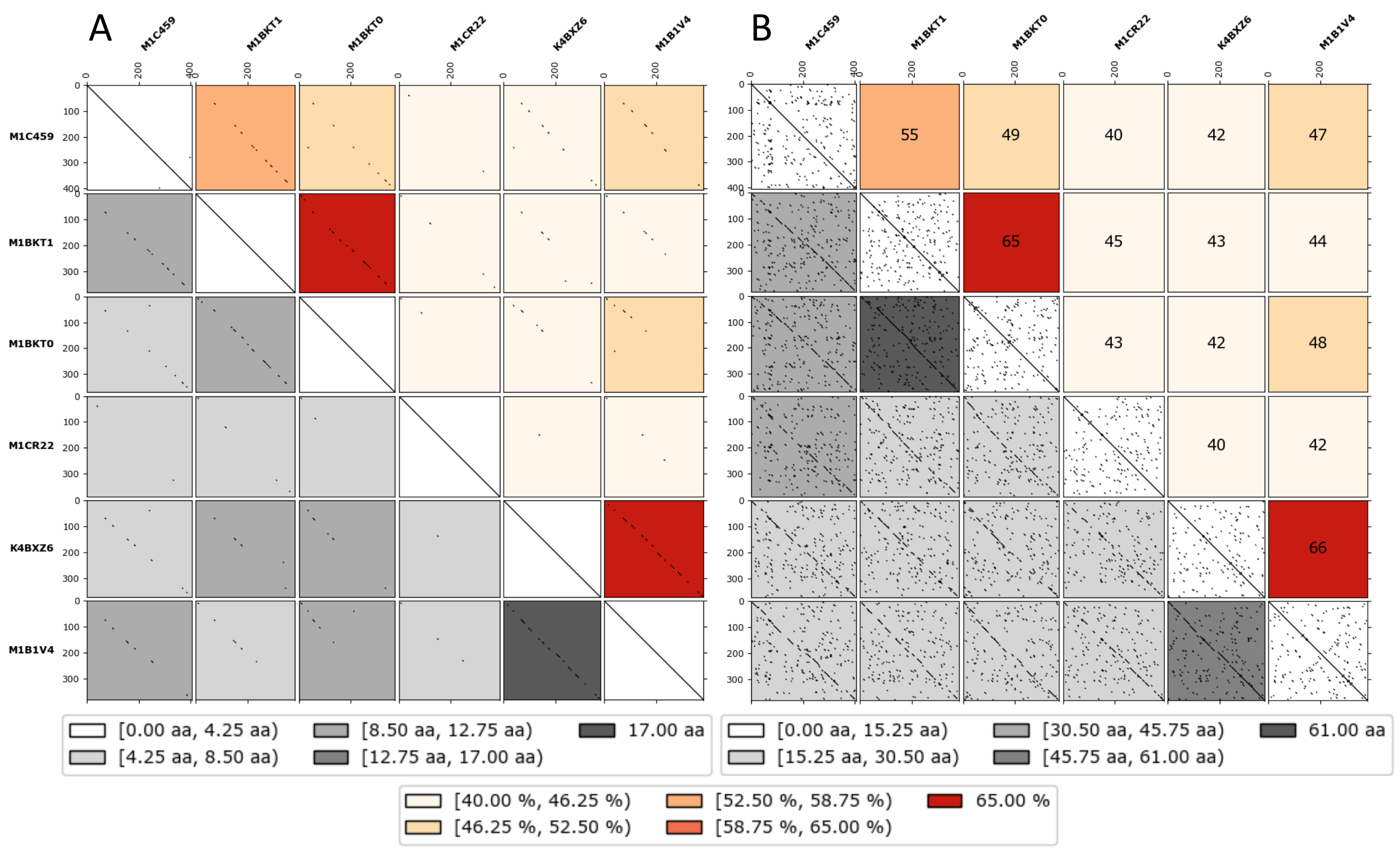
Used commands Supplemental Figure 18:

```
python flexidot.py -t n -k 10 -S 5 -p 0 -i Filamin_proteins.fas -p 2 -x y -P 10.5
```

4.3.2 Use case #4b: Similarity of potato nucleotide-binding leucine-rich repeat proteins

We compared the amino acid sequences of six Solanaceae (nightshades) nucleotide-binding leucine-rich repeat (NB-LRR) proteins in all-against-all dotplots (Supplemental Figure 19). A high global similarity between all NB-LRR genes is visible as long diagonal in each pairwise comparison. On the basis of a multiple sequence alignment (not shown), we extracted sequence identities for all NB-LRR pairs and use these for matrix shading (Supplemental Figure 19, orange shading). The identity values largely agree with the pattern observed in the dotplot analysis with LCS shading (Supplemental Figure 19, grey shading).

The two most similar proteins (NCBI accessions K4BXZ6 and M1B1V4) are immediately discernible with wordsize 5 (Supplemental Figure 19 panel A). The accompanying LCS information (Supplemental Table 7) shows that they share the longest LCS (17 amino acids). Toleration of two mismatches creates visible diagonals in all pairwise comparisons and reveals high sequence similarity and LCS in a second pair (M1BKT0 and M1BKT1, Supplemental Figure 19 panel B, Supplemental Table 7).



Supplemental Figure 19: All-against-all sequence comparison of six NB-LRR proteins. For window size 5 diagonals in dissimilar proteins are barely observed and general noise is repressed (A). If the window size is kept, but two mismatches are allowed all pairwise comparisons create visible diagonals (B). In (A) and (B) we combine LCS shading (grey) with custom-matrix shading (orange). The custom values represent pairwise sequence identities. Both shadings highlight the most similar protein pairs (M1B1V4 and K4BXZ6 as well as M1BKT0 and M1BKT1) sharing the overall longest LCS lengths and highest sequence identities.

Used commands Supplemental Figure 19:

```
Panel A>> python flexidot.py -i PF12061_seed.fas -f 0 -k 5 -p 2 -x Y -t n -y 0 -z 0 -D y -S 0 -U n
-u custom_matrix.txt

Panel B>> python flexidot.py -i PF12061_seed.fas -f 0 -k 5 -p 2 -x Y -t n -y 0 -z 0 -D y -S 2 -U y
-u custom_matrix.txt
```

Used commands (Supplemental Table 7)*:

```
python flexidot.py -i PF12061_seed.fas -f 0 -k 3 -p 2 -x Y -t n -w n -y 0 -z 0 -D Y
```

*every FlexiDot command in plotting mode 1 or 2 generates an LCS table. Minimal LCS considered are dependent on the wordsize. Here, for the sake of the LCS table, a small wordsize of 3 was chosen.

Supplemental Table 7: LCS table for the NBL-RR protein analysis. The forward LCS column is shaded in grey. The sequence pair with the longest forward LCS (K4BXZ6 and M1B1V4) is highlighted (bold, underlined).

#seq1	seq2	len_seq1	len_seq2	len_lcs_for (exact match)	%_min_seq_len	len_lcs_for (2 mismatches allowed)	%_min_seq_len
M1C459	M1C459	404	404	404	100.000	404	100.000
M1BKT1	M1C459	381	404	11	2.887	35	9.186
M1BKT0	M1C459	369	404	8	2.168	32	8.672
M1CR22	M1C459	388	404	6	1.546	33	8.505
K4BXZ6	M1C459	372	404	7	1.882	26	6.989
M1B1V4	M1C459	382	404	11	2.880	29	7.592
M1BKT1	M1BKT1	381	381	381	100.000	381	100.000
M1BKT0	M1BKT1	369	381	12	3.252	61	16.531
M1CR22	M1BKT1	388	381	7	1.837	24	6.299
K4BXZ6	M1BKT1	372	381	9	2.419	26	6.989
M1B1V4	M1BKT1	382	381	6	1.575	27	7.087
M1BKT0	M1BKT0	369	369	369	100.000	369	100.000
M1CR22	M1BKT0	388	369	6	1.626	23	6.233
K4BXZ6	M1BKT0	372	369	10	2.710	24	6.504
M1B1V4	M1BKT0	382	369	10	2.710	28	7.588
M1CR22	M1CR22	388	388	388	100.000	388	100.000
K4BXZ6	M1CR22	372	388	6	1.613	29	7.796
M1B1V4	M1CR22	382	388	6	1.571	25	6.545
K4BXZ6	K4BXZ6	372	372	372	100.000	372	100.000
M1B1V4	**K4BXZ6**	**382**	**372**	**<u>17</u>**	**4.570**	59	15.860
M1B1V4	M1B1V4	382	382	382	100.000	382	100.000

5 Literature

Brodie,R., Roper,R., Upton,C. (2004) JDotter: a Java interface to multiple dotplots generated by dotter. *Bioinformatics*, **20**, 279-281

Finn,R.D., Coggill,P., Eberhardt,R.Y., Eddy,S.R., Mistry,J., Mitchell,A.L., Potter,S.C., Punta,M., Qureshi,M., Sangrador-Vegas,A., Salazar,G.A., Tate,J., Bateman,A. (2016) The Pfam protein families database: towards a more sustainable future. *Nucleic Acids Research* **44** D279-D285

Katoh,K., and Standley,D.M. (2013) MAFFT multiple sequence alignment software version 7: Improvements in performance and usability. *Molecular Biology and Evolution* **30**, 772-780

Krumsiek,J., Arnold,R., Rattei,T. (2007) Gepard: a rapid and sensitive tool for creating dotplots on genome scale. *Bioinformatics,* **23**, 1026-1028

Noé,L., Kucherov,G. (2005) YASS: enhancing the sensitivity of DNA similarity search, *Nuc. Acids. Res.*, **33**, W540-W543

Rice P., Longden I., Bleasby A. (2000) EMBOSS: The European Molecular Biology Open Software Suite. *Trends in Genetics*. **16** (6), 276–277

Schwichtenberg,K., Wenke,T., Zakrzewski,F., Seibt,K.M., Minoche,A.E., Dohm,J.C., Weisshaar,B., Himmelbauer,H., and Schmidt,T. (2016) Diversification, evolution and methylation of Short Interspersed Nuclear Element families in sugar beet and related Amaranthaceae species. *Plant Journal* **85**, 229-244

Sevim,V., Bashir,A., Chin,C.S. and Miga,K.H. (2016) Alpha-CENTAURI: Assessing novel centromeric repeat sequence variation with long read sequencing. *Bioinformatics*. **32**(13):1921-1924

Sonnhammer,E.L. and Durbin,R. (1995) A dot-matrix program with dynamic threshold control suited for genomic DNA and protein sequence analysis. *Gene*, **167**, GC1–GC10

Symonová,R., Ocalewicz,K., Kirtiklis,L., Delmastro,G.B., Pelikánová,S., GarciaS., and Kovařík,A. (2017) Higher-order organisation of extremely amplified, potentially functional and massively methylated 5S rDNA in European pikes (*Esox* sp.). *BMC Genomics*, **18**, 391

Tamura,K., Stecher,G. and Kumar,S. (2016) MEGA7: Molecular Evolutionary Genetics Analysis version 7.0 for bigger datasets. *Molecular Biology and Evolution* **33**, 1870-1874

Van der Flier,A., Sonnenberg,A. (2001) Structural and functional aspects of filamins. Biochimica et Biophysica Acta (BBA) - *Molecular Cell Research*. **1538**(2-3) 99-117

Weber,B., and Schmidt,T. (2009) Nested Ty3-*gypsy* retrotransposons of a single *Beta procumbens* centromere contain a putative chromodomain. *Chromosome Research* **17**, 379-396

Weber,B., Heitkam,T., Holtgräwe,D., Weisshaar,B., Minoche,A.E., Dohm,J.C., Himmelbauer,H., and Schmidt,T. (2013) Highly diverse chromoviruses of *Beta vulgaris* are classified by chromodomains and chromosomal integration. Mobile DNA **4**, 8