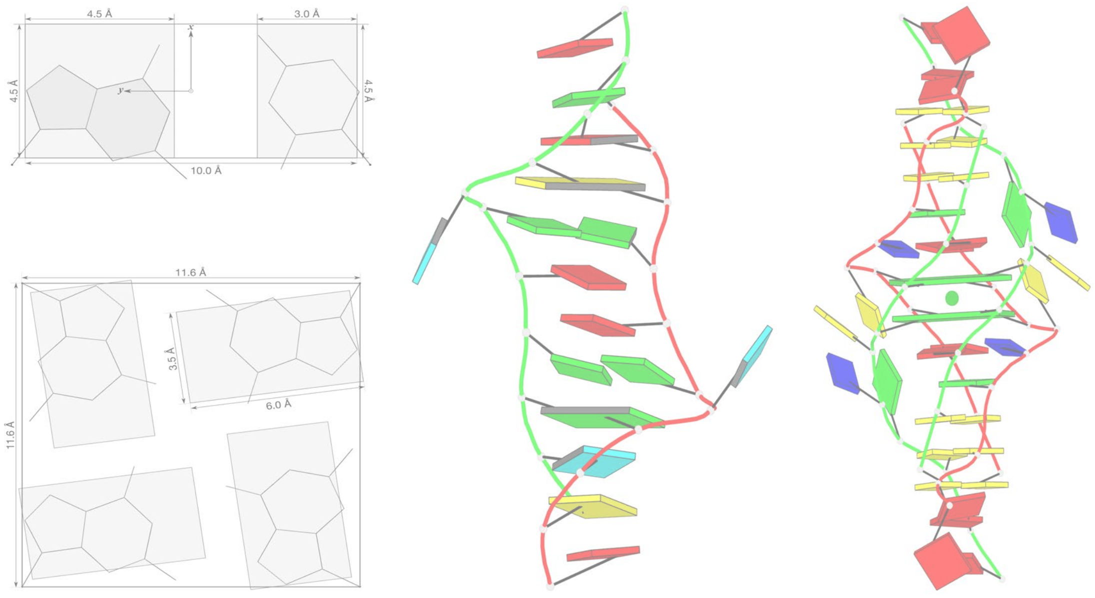
Supplementary Data

doi: 10.1093/nar/gkaa426

DSSR-enabled innovative schematics of 3D nucleic acid structures with PyMOL

A Practical Guide

(Last updated: May 16, 2020)



CLI (command-line interface) · PyMOL plugin · `http://skmatic.x3dna.org` · web API

Xiang-Jun Lu

xiangjun@x3dna.org

Department of Biological Sciences

Columbia University, New York

NY 10027, USA

Contents

1. Introduction

DSSR (Dissecting the Spatial Structure of RNA) is an integrated computational tool, designed from the bottom up to streamline the analysis and annotation of 3D nucleic acid structures (Lu *et al.*, 2015). Starting from an atomic coordinate file in PDB (`.pdb`) or PDBx/mmCIF (`.cif`) format, the program automatically identifies and characterizes numerous structural features, including modified nucleotides, arrays of stacked bases, (non-canonical) base pairs, higher-order base associations (multiplets), pseudoknots of arbitrary complexity, various types of 'closed' loops, canonical stems, coaxially stacked helices, and G-quadruplexes. It is efficient and robust due to extensive tests using all nucleic-acid-containing structures in the PDB (Burley *et al.*, 2018) and is continuously developed following user feedback. Because of its unmatched set of features, DSSR has been adopted widely in other structural bioinformatics resources (Baulin *et al.*, 2016; Zok *et al.*, 2018; Antczak *et al.*, 2019; Gallego *et al.*, 2019; Thiel *et al.*, 2019; Yesselman *et al.*, 2019; Sagendorf *et al.*, 2020; Zok *et al.*, 2020) and cited extensively in scientific journals (Desai *et al.*, 2017; Bayrak *et al.*, 2017; Meier *et al.*, 2018; Tan *et al.*, 2018; Berger *et al.*, 2019; Giambaşu *et al.*, 2019; Cuturello *et al.*, 2020; Kribelbauer *et al.*, 2020).

DSSR has been integrated into Jmol/JSmol (Hanson and Lu, 2017) via a JSON interface. Notably, the DSSR-Jmol integration introduces a novel and powerful SQL-like selection mechanism of DSSR-derived structural features[1] and simplified representations of nucleic acids via step diagrams and base blocks. The DSSR-Jmol integration fills a gap in RNA structural bioinformatics, and serves as an example for integrating DSSR-derived features into other (molecular graphics) programs.

PyMOL is an industry leader in 3D molecular visualization. It is especially sophisticated for protein structures. For nucleic acids, however, PyMOL is deficient in two main aspects. Firstly, it lacks a selection mechanism for RNA/DNA-specific features (including WC pairs or G-quadruplexes). Secondly, its representation styles of nucleic acids, most notably the nitrogenous bases, are limited. While molecular visualization tools with RNA/DNA-specific features are available (Couch, 2006; Lindow *et al.*, 2019), DSSR enhance the selection and visualization of nucleic acid structures in PyMOL in unprecedented ways. Specifically, DSSR creates schematic block representations in diverse styles that can be seamlessly integrated into and complemented with other popular visualization options of PyMOL. DSSR dramatically

[1]Here are some simple examples:

- `SELECT within(dssr, "nts WHERE is_modified")` for modified nucleotides
- `SELECT within(dssr, "pairs WHERE name = 'WC' OR name = 'Wobble'")` for canonical pairs
- `SELECT within(dssr, "pairs WHERE name !='WC'")` for non-Watson-Crick base pairs
- `Select WITHIN(dssr, "pairs WHERE name = 'Hoogsteen'")` for Hoogsteen pairs
- `select junctions` for all multi-branch junction loops
- `SELECT within(dssr, "junctions WHERE num_stems = 3")` for all three-way junction loops

See the DSSR-Jmol User Manual for details.

simplifies the depiction of G-quadruplexes by automatically detecting G-tetrads and depicting them as large square blocks. The DSSR-PyMOL schematics are simple, informative, and appealing. They are especially effective for RNA and DNA structures with up to dozens of nucleotides (Figure S1).

The DSSR-PyMOL schematic is built upon the base-block representation that can be created using `blocview`, a component of the 3DNA suite of programs (Lu and Olson, 2003, 2008; Li *et al.*, 2019). Specifically, the `blocview` script calls other 3DNA programs to generate base blocks and set the view, MolScript (Kraulis, 1991) to produce backbone ribbons, and Raster3D (Merritt and Bacon, 1997) to render the composite image. The idea of representing bases and pairs as rectangular blocks came from the pioneering work of Calladine *et al.* (Calladine and Drew, 1984; Calladine *et al.*, 2004) and was first implemented in the SCHNAaP/SCHNArP pair of programs (Lu *et al.*, 1997*a*,*b*). The DSSR-PyMOL integration presented here bypasses all 3DNA programs, MolScript, and Raster3D altogether. It is easier to use than the 3DNA `blocview` approach and produces better images. Figure S1 provides a brief summary of what the DSSR-PyMOL integration has to offer, with definitions of the various blocks and two representative schematic images.

There are four ways to bring DSSR-enhanced visualizations of nucleic acids structures into PyMOL: the command-line interface (CLI), the `dssr_block` PyMOL plugin, the web application, and the web API (application programming interface). Together, these four approaches cover virtually all conceivable cases of usage.

This document is intended to serve as a practical guide, with complete and reproducible examples. Even beginners or occasional users should be able to get started quickly, especially via the web application (`http://skmatic.x3dna.org`).

2. Installation

DSSR installation is only required for the CLI and PyMOL plugin interfaces. Assuming PyMOL is already available, one can download the `dssr_block.py` script (by Thomas Holder) from the PyMOL wiki page: the script can be executed (`run`), imported as a Python module, or installed with the ‘Plugin Manager’. The easiest way to benefit from the RNA/DNA block schematics in PyMOL is via the web interface at `http://skmatic.x3dna.org`: users just need to have a browser with internet connection, no need to install DSSR or PyMOL (or the plugin). Software developers can take advantage of the web API (`http://skmatic.x3dna.org/api`) programmatically, without the need of any installation.

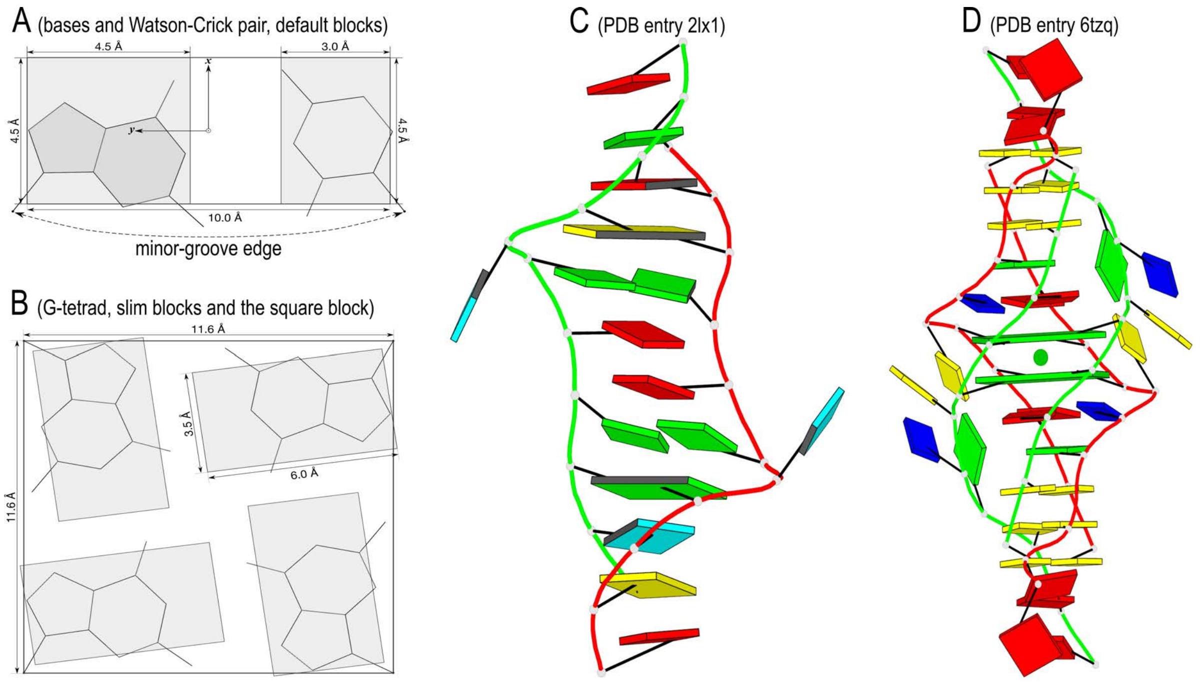


Figure S1: Definitions of DSSR blocks and two representative schematic images rendered in PyMOL. (A) Base blocks and the WC-pair block in default dimensions. The standard base reference frame (Olson *et al.*, 2001) is attached, and the minor-groove edge is marked. (B) The slim guanine blocks and the square G-tetrad block for simplified visualization of G-quadruplexes. (C) An RNA structure featuring WC-pair blocks and the minor-groove edges (colored black). The PDB entry used is `2lx1` (Kennedy *et al.*, 2012). (D) A DNA G-quadruplex highlighting the two G-tetrad blocks in the middle. The PDB entry used is `6tzq` (Chu *et al.*, 2019). In (A-B), the bases are in an idealized planar geometry, and the blocks have a default thickness of 0.5 Å. Color codes in (C-D): A, red; C, yellow; G, green; T, blue; U, cyan; G-tetrad, green; WC-pairs, per base in the leading strand. The DSSR-PyMOL schematics make the base identity, pairing geometry, stacking interactions, double-helical stems, and the two-layered G-quadruplex obvious.

DSSR is designed with simplicity and robustness in mind; it gets the job done, and then stays out of the way. The program has been implemented in ANSI C as a standalone, command-line program. DSSR is self-contained and the binary executable (on macOS, Linux and Windows) is small (< 2MB), with *zero* runtime dependencies on third-party libraries. Getting DSSR up and running is thus straightforward, as detailed below.

1. Register at the 3DNA Forum to download DSSR and ask questions.

2. Log in, at the top-left corner under "Welcome", click "Downloads" and "3DNA download". Proceed to download the DSSR binary executable (named '`x3dna-dssr`' for Linux/macOS[2] or '`x3dna-dssr.exe`' for Windows) for your operating system and architecture. Currently, compiled versions are available for the most common systems: Linux (32-bit and 64-bit), macOS, and Windows (also runs under Cygwin and MinGW/MSYS). As of this writing, DSSR v1.9.9-2020feb06 is the latest release.

3. On macOS or Linux, run the command '`chmod u+x x3dna-dssr`' to make DSSR executable. On Windows, this step is not necessary.

4. Preferably, move DSSR into a folder on your command search path (e.g., `~/bin`) so you can run the program conveniently from anywhere. Or, you can simply put DSSR somewhere (e.g., your current working directory) and specify the path explicitly to execute the program.

5. Type '`x3dna-dssr -h`' to verify your installation. You should see a help message with DSSR version info, its usage options, and some examples. In case you see otherwise (exceedingly rare), please report issues on the 3DNA Forum.

DSSR is stable in terms of basic functionality and main output (especially with the JSON format). Meanwhile, DSSR is actively maintained and developed via new features and bug fixes. For example, the interface to PyMOL rendering has been stable in terms of the `--block-file` and `--block-color` options since DSSR v1.5.2-2016apr02. However, the features available via these two options have been refined and expanded in later on releases (e.g., `--block-file=g4` to draw square blocks for G-tetrads in G-quadruplexes). To keep up to date, users can simply download DSSR again from the 3DNA Forum to replace (i.e., overwrite) the old copy.

[2]On macOS with the default **Safari** browser (but not Chrome or Firefox), the *extra* `.dms` extension may be added to the downloaded DSSR executable file. Thus, instead of `x3dna-dssr`, one gets `x3dna-dssr.dms`. To avoid confusions in following the guide, please remove the `.dms` extension by running: `mv x3dna-dssr.dms x3dna-dssr`.

3. The command-line interface

The command-line interface (CLI) is the most efficient and versatile way to explore the numerous features that DSSR has to offer. It also lays the foundation for the other three interfaces—the PyMOL plugin, the web application, and the web API—that make these DSSR functionalities easily accessible to a wider audience. This section explores in depth the five major options for DSSR-enhanced visualization of nucleic acid structures with PyMOL: `--block-file`, `--block-color`, `--block-depth` (i.e., thickness), `--cartoon-block`, and `--blocview` (shortened form for `--block-view`). DSSR has far more features than those documented here. Interested users are referred to the User Manual for more information. DSSR-related questions are welcome on the 3DNA Forum.

Listing 1 and Figure S2 present the DSSR commands and the corresponding PyMOL screenshot images using PDB entry `2lx1` as an example. Note that for better visualization, the atomic coordinate file `2lx1.pdb` downloaded from the RCSB PDB has been transformed via the DSSR `--view` option[3] to `2lx1-bestview.pdb` which was used for the PyMOL-rendered images. Calling `reinitialize; bg_color white` three times (line nos. 8, 14, and 20) in Listing 1 ensures that each image was created clutter free, with PyMOL default settings except for a white background. Following these concrete instructions, users should be able to quickly understand the process of creating DSSR-PyMOL schematics.

Listing 1: Commands used to create the sample images shown in Figure S2

```
 1  # Download entry '2lx1' from the RCSB PDB website, saved to file '2lx1.pdb'
 2  curl https://files.rcsb.org/download/2lx1.pdb -o 2lx1.pdb
 3  # Re-orient '2lx1.pdb' to an extended view (vertically)
 4  x3dna-dssr -i=2lx1.pdb --view -o=2lx1-bestview.pdb
 5
 6  # Generate base blocks with default settings in Raster3D (.r3d) format
 7  x3dna-dssr -i=2lx1-bestview.pdb --block-file -o=2lx1-base.r3d
 8  reinitialize; bg_color white # within PyMOL
 9  load 2lx1-base.r3d # blocks: screenshot as in Figure S2A
10  load 2lx1-bestview.pdb # blocks + cartoon: screenshot as in Figure S2B
11
12  # Generate WC-pair blocks, with minor-groove edges in black
13  x3dna-dssr -i=2lx1-bestview.pdb --block-file=wc-minor -o=2lx1-wc.r3d
14  reinitialize; bg_color white # within PyMOL
15  load 2lx1-wc.r3d
16  load 2lx1-bestview.pdb # screenshot as in Figure S2C
17
18  # Default base blocks, with R in red and Y in gray, each block 1.2-A thick
19  x3dna-dssr -i=2lx1-bestview.pdb block-color='R:red,Y:gray' block-depth=1.2 -o=2lx1-colx.r3d
20  reinitialize; bg_color white
21  load 2lx1-colx.r3d
22  load 2lx1-bestview.pdb # screenshot as in Figure S2D
```

[3] As shown below, the `--view` option is directly related to `--blocview` that is used for generating images oriented in the most extended view, automatically.

3.1. The `--block-file` option

The DSSR `--block-file` (or `block_file`) option[4] turns on the block representation in its default settings, as shown in Figure S2A. It creates an output file in Raster3D (Merritt and Bacon, 1997) `.r3d` format, which is automatically converted to compiled graphics objects (CGO) and rendered by PyMOL (Listing 1). This option can be further customized with numerous features specified as a list of keywords, in the form of `--block-file=key1-key2-...`. The keys for the most commonly-used features are documented below (see also examples in Section 3.6).

- `face` (i.e., specified as `--block-file=face`): draw the six sides of a (rectangular) block in solid color. This is the default feature, so that `--block-file` by itself has the same effect. The default sizes of the various blocks are shown in Figure S1, and the default colors are: A, red; C, yellow; G, green; T, blue; U, cyan.
- `edge`: draw only an outline of the blocks. The `face` and `edge` features can be combined, i.e., `--block-file=face-edge`, where the separator hyphen can be omitted or replaced by another character (e.g., `+`). This features-selection scheme is flexible and powerful since it allows for new functionalities to be added with the same interface (see below for the `fill-hbond` combination).
- `wc`: draw each Watson-Crick (WC) pair as a long block instead of two individual base blocks (Figure S1). By default, a WC pair is colored by the leading base (positioned ahead in the atomic coordinate file): e.g., G–C pair is colored as G, and C–G is colored as C. This feature makes WC pairs and double-helical regions stand out (Figure S2C).
- `g4`: draw each G-tetrad in G-quadruplexes (G4s) as a square block instead of four G-base blocks (Figure S1). By default, the G-tetrad is colored green, as for G. This feature makes it easy to visualize G4s in a structural context (Figure S1). See `http://g4.x3dna.org` for a compilation of G4s auto-annotated using DSSR from the PDB.

[4]The DSSR block-file option is matched by the regular expression `^-?-?block?[-_]?file`, case *insensitively*. Here each character (or set) before `?` is optional, so that: (1) Each of the first two hyphens can be omitted. (2) The separator between `block` and `file` can be a hyphen, an underscore, or omitted. (3) The letter `k` in `block` can also be left out (i.e., `bloc`). Thus, this option allows for many common variations, including: `--block-file`, `-block-file`, `block-file`, `blocfile`, `Block-File`, and `block_file` (as used in the DSSR-PyMOL plugin, see Section 4), etc. The same flexibility goes for other DSSR options: for example, `--block-view` can be shortened as `--blocview`, or `blocview` (which is the name of a standalone script in the 3DNA distribution).

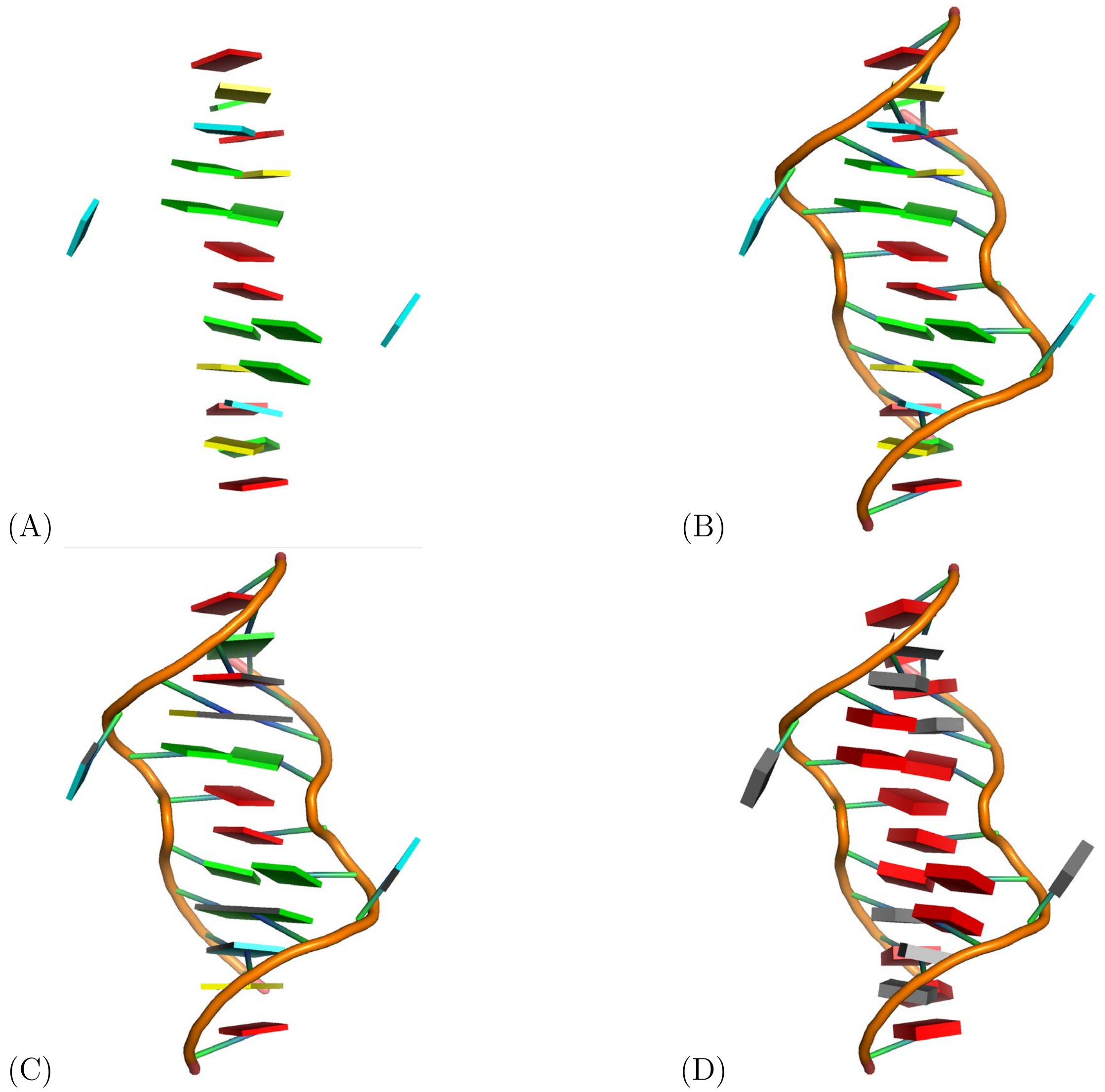


Figure S2: Sample images of PDB entry `2lx1` generated using commands in Listing 1. (A) Default DSSR base blocks, in Raster3D (Merritt and Bacon, 1997) `.r3d` format, which were converted to compiled graphics objects (CGO) and rendered by PyMOL. Color codes: A, red; C, yellow; G, green; U, cyan. With this block representation, the following structural features are immediately obvious: the AA stack in the middle, the two G·G noncanonical pairs (one above and the other below the AA stack), and the two bulged-out U's. (B) Base blocks as in (A), but combined with the default cartoon representation of nucleic acids in PyMOL. (C) As in (B), but with WC-pair blocks enabled and the minor-groove edges in black. It is clear that the structure has two right-handed WC helical fragments (termed stems in DSSR), each one stacking with a G·G pair mentioned above. WC blocks are colored by the leading base of a pair (e.g., A–U is colored red as for A, and U–A is colored cyan as for U). (D) As in (B), but with purines (R) in red and pyrimidines (Y) in gray, and each block 1.2-Å thick. Clearly, there are six R's the middle of the RNA molecule.

- `minor`: highlight the minor-groove edge of WC pairs or bases in black. With settings `--block-file=wc-minor`, the WC double-helical regions and their right-handedness become obvious (Figure S2C).

- `slim`: draw blocks with a more oblong shape than the default (Figure S1). For example, R (purine) is now of dimension 6Å-by-3.5Å instead of the default 4.5Å-by-4.5Å. This option was initially introduced to highlight the H-bonding directionality of G-tetrads (see Figures S1 and S6).

- `syn`: highlight nucleotides in *syn*-conformations by a special color (teal by default). This feature is particularly relevant for G4s where *syn*-guanosines are abundant. It is the prominent connection between *syn* and G4s that prompted me to pick the teal color (a dark *green*ish-blue, see Figure S6).

- `fill`: fill base rings with solid color. The color codes are as for the corresponding base blocks (see above). When this feature is enabled, `face` is turned off (since the filled base rings would be completely covered by the solid blocks).

- `hbond` (or `h-bond`): identify and draw H-bonds (as dashed lines, colored magenta by default). As show in examples below (Section 3.6), the `--block-file=fill-hbond` combination is an effective way to draw molecular images of nucleic acid fragments.

The order of the keywords in a list does not matter. For instance, `g4-wc-minor` would have the same effect as `wc-minor-g4`. If a structure does not possess a feature, the corresponding keyword will be ignored. For example, if `wc` is specified for a structure without WC pairs, nothing will occur.

There are actually quite a few experimental (*undocumented*) attributes associated with the option `--block-file`. Importantly, the design allows for additional features (as new keywords) to be implemented, without any change of the interface.

3.2. The `--block-color` option

This option has been designed to customize the color codes of bases, or features, specified via `--block-file` (see above). It takes a list of comma-separated `key:color` pairs, i.e., `--block-color='key1:color1,key2:color2,...'`. Here are the forms that `key` and `color` can take, and some examples:

- `key` can be: (1) a standard base (A, C, G, T, or U); (2) an IUPAC degenerate symbol (e.g., R for A/G, Y for C/T/U, or N for A/C/G/T/U); (3) a feature that can be specified via `--block-file` (e.g., `minor`, `wc`, or `hbond`).
- `color` can be: (1) a common color name (e.g., red, blue, black); (2) an RGB triplet optionally included in a pair of brackets (e.g., [0 1 1] for cyan); (3) a decimal number in range [0, 1] for a shade of gray (e.g., 0.8).
- Some examples: (1) `--block-color='R:red,Y:yellow'` to color purines (A and G) red, and pyrimidines (C, T, and U) yellow; (2) `--block-color='wc:pink,minor:0.1'` to color WC pairs pink, and the minor-groove edges 10% white (close to black); (3) `--block-color='hbond:black'` to color H-bonds black. Note the quotation marks around the `key:color` lists.

3.3. The `--block-depth` option

This option sets the thickness of blocks, as in `--block-depth=1.2`. It takes a value greater than 0 and less than 3.6 Å, with a default of 0.5 Å.

3.4. The `--cartoon-block` option

The three DSSR options documented above produce schematic blocks, in various styles, outputted as `.r3d` files (for examples, see line nos. 7, 13, and 19 of Listing 1). In practice, these blocks are seldomly used alone, but often combined with popular representations in PyMOL to produce final molecular images (Figure S2). PyMOL comes with *numerous* settings that can be customized for the rendered graphics. The `--cartoon-block` option combines PyMOL *cartoon* representations and DSSR *block* schematics, together with a selected set of settings. It automates the process of creating DSSR-enhanced visualization of nucleic acid structures in PyMOL.

Listing 2: Examples of applying `--cartoon-block` and `--blocview` on PDB entry `2lx1`

```
x3dna-dssr -i=2lx1.pdb --cartoon-block -o=2lx1.pml
# Run the following three commands within PyMOL
load 2lx1.pml
ray 1800
png 2lx1-pymol.png # see Figure S3A

x3dna-dssr -i=2lx1.pdb --blocview -o=2lx1-view.pml
# Run the following three commands within PyMOL
load 2lx1-view.pml
```

```
10 ray 1800
11 png 2lx1-view-pymol.png # see Figure S3B
```

The direct output of the `--cartoon-block` option is a PyMOL script file (with extension `.pml`) as shown in line no. 1 of Listing 2 for PDB entry `2lx1`. By itself, the option `--cartoon-block` enables *default* DSSR block settings, which can be further customized via the `--block-file`, `--block-color`, and `--block-depth` options documented above. PyMOL parameters can also be easily adapted by modifying the resultant script file (`2lx1.pml`). After loading `2lx1.pml` into PyMOL and ray-tracing the scenes, one can create a PNG image as shown in Figure S3A (line nos. 3–5 of Listing 2, with empty whitespace cropped, see below). The quality of the image can be changed via the `ray` command (e.g., `ray 2400` or `ray 1200` for a higher or lower resolution).

The full content of the PyMOL script file `2lx1.pml` is shown in Listing 3. In addition to loading both the PDB file (line no. 3) and DSSR block file (line no. 14), the script sets many PyMOL parameters. Among the settings, please note the followings:

- DSSR automatically identifies two RNA chains in PDB entry `2lx1`: they are rendered as cartoons in PyMOL, with chain A colored red, and B colored green (line nos. 6–12).
- The `cartoon_nucleic_acid_mode` is set to 1 (line no. 21), which uses C3′ atoms for backbone trace instead of the default P atoms.
- Background color is set to white, all solvents are removed, and all hydrogen atoms are hidden (line no. 23–25).
- Image perspective is turned off (line no. 35).

Listing 3: The PyMOL script generated with `--cartoon-block` on PDB entry `2lx1`

```
 1 reinitialize
 2
 3 load 2lx1.pdb, whole_str
 4 hide everything, whole_str
 5
 6 create na_A, chain A
 7 set cartoon_nucleic_acid_color, red, na_A
 8 show cartoon, na_A
 9
10 create na_B, chain B
11 set cartoon_nucleic_acid_color, green, na_B
12 show cartoon, na_B
13
14 load 2lx1.r3d, block
15
```

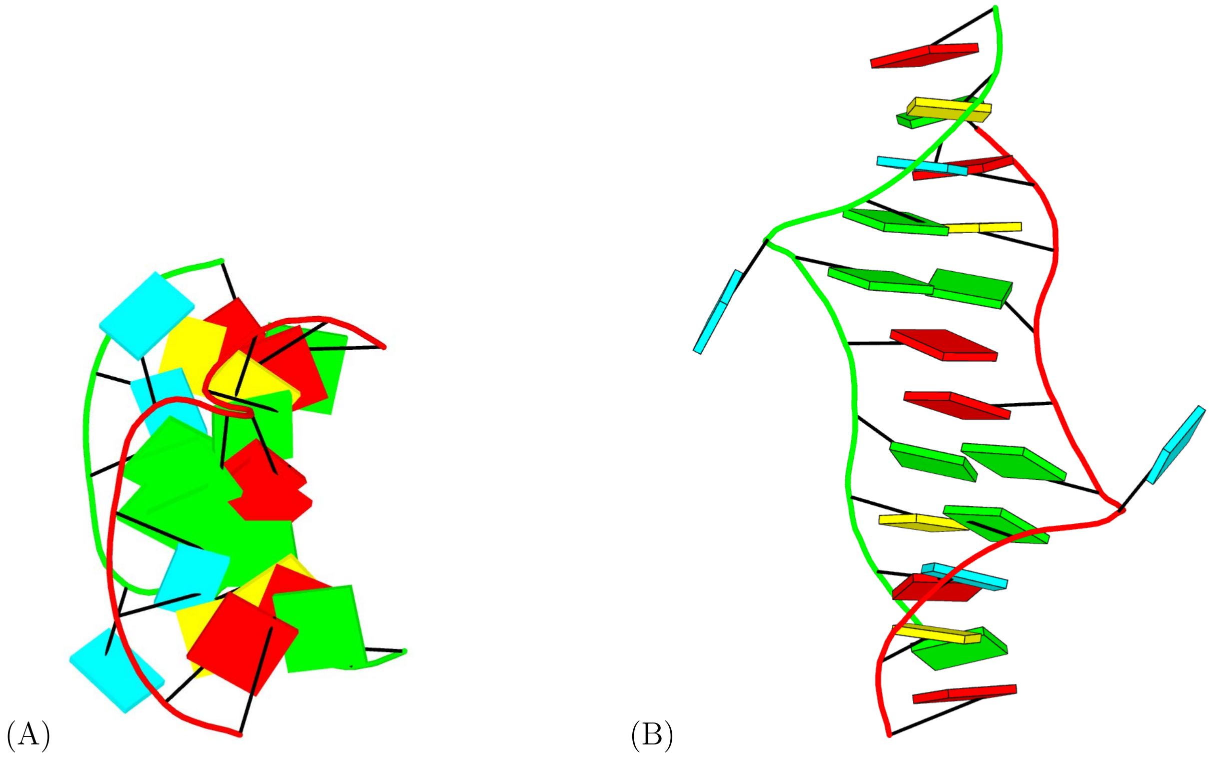


Figure S3: Schematic images of PDB entry `2lx1` combining PyMOL backbone cartoons with DSSR base blocks. (A) Image generated with the `--cartoon-block` option, in a view defined by the original PDB atomic coordinates. (B) Image created using the `--blocview` option that reorients the structure into the most extended view vertically, with black outlines around the solid-colored base blocks.

```
set cartoon_ladder_mode, 1
set cartoon_ladder_radius, 0.1
set cartoon_ladder_color, black

set cartoon_tube_radius, 0.16889
set cartoon_nucleic_acid_mode, 1

bg_color white
remove solvent
hide everything, hydro

util.cbaw
set sphere_quality, 4
set stick_quality, 16

set depth_cue, 0
set ray_trace_fog, 0

set ray_shadow, off
set orthoscopic, 1

set antialias, 1
set valence, 0

set ambient, 0.68
set reflect, 0
set direct, 0.6
set spec_direct, 0
set light_count, 1
```

Moreover, the `--cartoon-block` option can be supplemented with a list of keywords (as for `--block-file`). Here are the keywords for the most common use-cases and the corresponding PyMOL features they introduce (via the DSSR-derived `.pml` file):

- `png`: add PyMOL commands `ray 1800` and `png 2lx1-pymol.png` (using PDB entry `2lx1` as an example) to ray-trace the scenes and output a PNG image. Note that the PNG image file is named with the `-pymol` suffix (line nos. 4–5, Listing 2).
- `stick`: add a simple PyMOL representation where bonds are drawn as sticks, and atoms as small spheres.
- `label`: add labels in style A16 (for adenine, no. 16) based on C1′ atoms.
- `c3`: draw gray spheres on C3′ atoms which are used to draw RNA/DNA backbone cartoons (see above).
- `metal`: draw metals as spheres.
- `organic`: draw non-polymer organic compounds (ligands) in ball-and-stick representation.

With `--cartoon-block=png`, for example, the interactive steps for generating a PNG image (lines nos. 3–5, Listing 2) can be automated via PyMOL command-line options. Try the following three commands (which can be put into a script):

```
1 x3dna-dssr -i=2lx1.pdb --cartoon-block=png -o=2lx1-png.pml
2 pymol -Q -k -c 2lx1-png.pml # generate 2lx1-pymol.png, with extra whitespace
3 convert -trim +repage -border 10 -bordercolor white 2lx1-pymol.png 2lx1-trim.png
```

Here the `-Q` option (quiet) suppresses all text output, `-k` stops PyMOL from loading `pymolrc` or plugins, and `-c` launches PyMOL in command-line only mode for batch processing (line no. 2). The three PyMOL command-line options can be combined as `-Qkc`. The PNG image directly from PyMOL contains extra white background which is automatically cropped using `convert` from the ImageMagick suite. The cropped PNG file (`2lx1-trim.png`) has a maximum of 10px white border, and it is the version actually used in Figure S3A. Other images (e.g., Figure S3B) in the guide have been processed similarly.

3.5. The `--blocview` option

Th `--blocview` option (short form for `--block-view`) behaves just like `--cartoon-block` but with two additional features, as shown below:

- The atomic coordinate file is automatically transformed, as with the `--view` option (see line no. 4 in Listing 1).
- The option `--block-file=face-edge` is enabled, leading to blocks with solid faces (in default colors) and black outlines.

Figure S3B shows an example image for PDB entry `2lx1` using commands in line nos. 7–11 of Listing 2. Compared to commands in line nos. 1–5 of Listing 2 and Figure S3A, the differences between the two options are obvious. Overall, the `--cartoon-block` option allows for more user control over the rendered image, whilst `--blocview` serves as an excellent default (or starting point) for RNA/DNA visualization. The pre-calculated PDB entries (see Section 5) are based the `--blocview` option, together with settings specified via `--block-file=wc-g4-minor`. As a matter of fact, nucleic-acid-containing entries (e.g., `2lx1`) in the RCSB PDB (Burley *et al.*, 2018) and the NDB (Narayanan *et al.*, 2014) have been created using the standalone `blocview` script in 3DNA (Lu and Olson, 2003, 2008). That is why DSSR has introduced the `--blocview` variant (or just `blocview`, see Footnote 4) as a shorthand form of `--block-view`.

3.6. Sample applications

The above sections have documented the five major block-related options in DSSR, i.e., `--block-file`, `--block-color`, `--block-depth`, `--cartoon-block`, and `--blocview`. In particular, the DSSR commands and PyMOL scripts listed therein should allow users to reproduce schematic images in Figures S2 and S3. This section gives further sample applications to showcase salient features enabled by DSSR for enhanced visualizations of RNA/DNA structures in PyMOL.

To serve as a tutorial, detailed command listings are provided so users can copy-and-paste them to a terminal window to *reproduce* figures reported in this section. In practice, these commands are normally put into a shell script to run automatically. That is actually how all the figures in this guide have been created.

Essentially, the creation of an image in this section takes four steps: (1) get an atomic coordinate file in `.pdb` or `.cif` format (downloaded from the PDB using `curl`, or another resource); (2) run DSSR on the coordinate file with specific options to generate a PyMOL `.pml` file; (3) run PyMOL in command-line mode to convert the `.pml` file to a PNG image; (4) crop whitespace in PyMOL-generated PNG image using `convert`. Further notes are added in the listings when appropriate (following the `#` symbol, e.g., line no. 18 of Listing 4 on the `--symm` option). To avoid confusion and file overwritten, each set of samples has been put into a separate directory.

3.6.1. Cartoon-block schematics

Listing 4 gives the commands to produce the schematic images shown in Figure S4 for six PDB entries: (A) `4kz2`, a crystal structure of phi29 pRNA 3WJ core (Zhang *et al.*, 2013); (B) `2lx1`, the major conformation of the internal loop 5′GAGU/3′UGAG (Kennedy *et al.*, 2012); (C) `6tzq`, a DNA G-quadruplex/i-motif hybrid (Chu *et al.*, 2019); (D) `2hoj`, the crystal structure of an *E. coli* thi-box riboswitch bound to thiamine pyrophosphate (Edwards and Ferré-D'Amaré, 2006); (E) `6rjg`, a cryo-EM structure of the St1Cas9-sgRNA-AcrIIA6-tDNA59-ntPAM complex (Fuchsbauer *et al.*, 2019); and (F) `6pj6`, a high resolution cryo-EM structure of *E.coli* 50S (Stojković *et al.*, 2020).

Figure S4 showcases settings (`--blocview=png-c3` and `--block-file=wc-minor`) that work well in common cases. Additional features can be turned on: Figure S4C shows the Ba^{2+} `metal` ion between the two G-tetrads (`g4`), and Figure S4D highlights the thiamine diphosphate (a small-molecule metabolite, `organic`). As depicted in Figure S4A-E, the cartoon-block schematics are simple and aesthetic, highly effective for nucleic acid structures with dozens of nucleotides. For a gigantic ribosomal structure (available only in `.cif` format in the PDB) such as `6pj6` (Figure S4F), the image becomes a colorful blob, yet it remains easily recognizable.

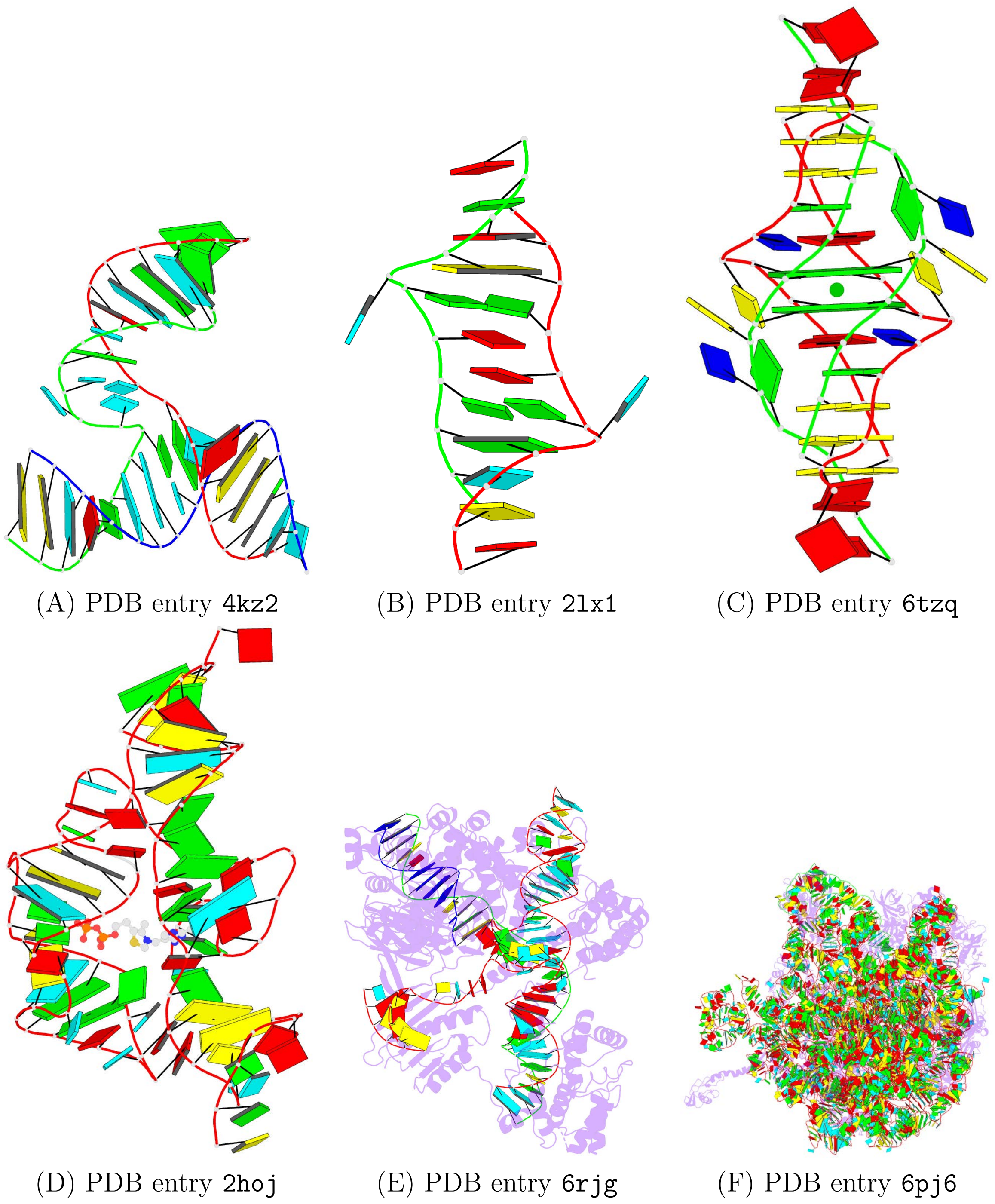
By default, DSSR reads in the *first* model and thus creates only a representative cartoon-block image of an NMR ensemble. With the DSSR-plugin to PyMOL (see below), it is easy to produce an aggregate image with all models to show conformational variations. Biological units of X-ray crystal structures may contain multiple models formatted as an NMR-like ensemble in the PDB. This is confusing since the different models in a biological unit are symmetry related instead of being fully independent as in an NMR ensemble. In such cases, the DSSR `--symm` option is required to process the entire structure of a biological unit with multiple models. As an example, please see line nos. 18–19 of Listing 4 and Figure S4C for PDB entry `6tzq`. Without `--symm`, the two G-tetrads will *not* be detected: the first model (as in the asymmetric unit) contains only half of the whole structure (e.g., two guanines of a G-tetrad). DSSR also auto-detects polypeptide chains in proteins, and outputs PyMOL commands to render them as purple cartoons with 70% transparency (Figure S4E-F).

Listing 4: Commands used to create schematic images for the six PDB entries in Figure S4

```
mkdir -p example1; cd example1 # to run the following commands in a new directory (folder)

# Figure S4A
curl https://files.rcsb.org/download/4kz2.pdb -o 4kz2.pdb
x3dna-dssr -i=4kz2.pdb --blocview=png-c3 --block-file=wc-minor -o=4kz2.pml
pymol -Qkc 4kz2.pml # create: 4kz2-pymol.png
convert -trim +repage -border 10 -bordercolor white 4kz2-pymol.png 4kz2.png

# Figure S4B (also Figure S1C)
curl https://files.rcsb.org/download/2lx1.pdb -o 2lx1.pdb
x3dna-dssr -i=2lx1.pdb --blocview=png-c3 --block-file=wc-minor -o=2lx1.pml
pymol -Qkc 2lx1.pml # create: 2lx1-pymol.png
convert -trim +repage -border 10 -bordercolor white 2lx1-pymol.png 2lx1.png

# Figure S4C (also Figure S1D)
curl https://files.rcsb.org/download/6tzq.pdb1.gz -o 6tzq.pdb1.gz
gunzip -f 6tzq.pdb1.gz # biological unit with two models in a MODEL/ENDMDL ensemble
# note the --symm option to read in both models, and --block-file=g4 for G-tetrad square blocks
x3dna-dssr -i=6tzq.pdb1 --symm --blocview=png-c3-metal --block-file=g4 -o=6tzq.pml
# edit '6tzq.pml' by adding 'turn y, -90' before PNG output to change the view
pymol -Qkc 6tzq.pml # create: 6tzq-pymol.png
convert -trim +repage -border 10 -bordercolor white 6tzq-pymol.png 6tzq.png

# Figure S4D -- with ligand in ball-and-stick presentation
curl https://files.rcsb.org/download/2hoj.pdb -o 2hoj.pdb
x3dna-dssr -i=2hoj.pdb --blocview=png-c3-organic --block-file=wc-minor -o=2hoj.pml
pymol -Qkc 2hoj.pml # create: 2hoj-pymol.png
convert -trim +repage -border 10 -bordercolor white 2hoj-pymol.png 2hoj.png

# Figure S4E -- with protein cartoons in purple
curl https://files.rcsb.org/download/6rjg.pdb -o 6rjg.pdb
x3dna-dssr -i=6rjg.pdb --blocview=png-c3 --block-file=wc-minor -o=6rjg.pml
pymol -Qkc 6rjg.pml # create: 6rjg-pymol.png
convert -trim +repage -border 10 -bordercolor white 6rjg-pymol.png 6rjg.png

# Figure S4F -- a ribosomal structure, in mmCIF format
curl https://files.rcsb.org/download/6pj6.cif -o 6pj6.cif
x3dna-dssr -i=6pj6.cif --blocview=png-c3 --block-file=wc-minor -o=6pj6.pml
pymol -Qkc 6pj6.pml # create: 6pj6-pymol.png
convert -trim +repage -border 10 -bordercolor white 6pj6-pymol.png 6pj6.png
```

3.6.2. Base stacks

DSSR can automatically derive many structural features, as documented in the User Manual. One of the features is ‘base stack’, defined as an *ordered* list of nucleotides assembled



(A) PDB entry `4kz2` (B) PDB entry `2lx1` (C) PDB entry `6tzq`

(D) PDB entry `2hoj` (E) PDB entry `6rjg` (F) PDB entry `6pj6`

Figure S4: DSSR-PyMOL cartoon-block images for six representative PDB entries. The commands used to create these six schematic images are shown in Listing 4. Note the WC-pairs as long blocks in (A-B), the G-tetrads as square blocks and the metal ion as a green sphere in (C), the ligand thiamine diphosphate as balls-and-sticks in (D), and proteins as purple cartoons in (E-F).

together via base-stacking interactions, regardless of backbone connectivity. By default, stacking interactions within a stem[5] are excluded (see Figure S5). As is often the case, the concept becomes clear with an example. Running DSSR on PDB entry `2lx1` (line no. 4 of Listing 5) identifies five stacks (ordered by length and residue number), as listed below:

```
List of 5 stacks
  Note: a stack is an ordered list of nucleotides assembled together via
        base-stacking interactions, regardless of backbone connectivity.
        Stacking interactions within a stem are *not* included.
   1 nts=2 GG A.G6,A.G8
   2 nts=2 CA A.C10,A.A11
   3 nts=2 GG B.G17,B.G19
   4 nts=2 CA B.C21,B.A22
   5 nts=6 CGAAGC A.C3,A.G4,B.A16,A.A5,B.G15,B.C14
```

The longest base stack, no. 5 in the above list, contains six nucleotides (nts) of base sequence `CGAAGC`. This 6-base stack is used here for illustration: the commands are shown in Listing 5 and the resultant images are provided in Figure S5. Line nos. 4–6 of Listing 5 worth a note: (line no. 4) in addition to the main output file (`2lx1.out`), the DSSR run also generates a file named `dssr-stacks.pdb`. This file contains the five stacks of bases in an NMR-like ensemble delineated by MODEL/ENDMDL tags; (line no. 5) the auto-generated file `dssr-stacks.pdb` is renamed `2lx1-stacks.pdb` to avoid being overwritten or deleted by another DSSR run; (line no. 6) the 6-base stack (the 5th model) is extracted to file `2lx1-6bases.pdb`, which is the atomic coordinate file used for Figure S5C.

The stacked ensemble of six bases is crystal clear from the DSSR-PyMOL schematic representation in Figure S5C. By following the color codes, even the identities of the stacked bases can be easily read out: C(yellow)-G(green)-A(red)-A(red)-G(green)-C(yellow). It is also obvious that the six stacked bases are from two fragments, each of 3 nts, interdigitated with adenines in the middle.

Figure S5A is copied directly from Figure S4B to show the concept of stem (see Footnote 5): there are two stems towards both ends in PDB entry `2lx1`. Figure S5B is in the same view as Figure S5A (because of `--blocview` on the same `2lx1.pdb` coordinate file): it shows the whole `2lx1` structure, with base blocks and *labels*, to put the six stacked bases in context. Note that the C3 and C14 bases are at the termini of the upper and lower stems, each consisting of three WC pairs (cf. Figure S5A). It is also worth noting that Figure S5C is oriented via a `--blocview` transformation on the 6 nts only (`2lx1-6bases.pdb`). As a result, the six bases are *not* in the same view as the corresponding ones in Figure S5B.

[5]Stem is defined in DSSR as a double-helical region consisting of canonical (WC or G–U wobble) pairs and uninterrupted backbones along both strands.

Listing 5: Commands to create the base-stack schematics in PDB entry `2lx1` (Figure S5)

```
mkdir -p example2; cd example2

curl https://files.rcsb.org/download/2lx1.pdb -o 2lx1.pdb
x3dna-dssr -i=2lx1.pdb -o=2lx1.out # main output file, plus many auxiliary files
cp -f dssr-stacks.pdb 2lx1-stacks.pdb # fixed name dssr-stacks.pdb
x3dna-dssr -i=2lx1-stacks.pdb --select-model=5 -o=2lx1-6bases.pdb # the 5th model with 6 stacked bases

# Figure S5A is copied from Figure S4B

# Figure S5B, with individual base blocks and labels
x3dna-dssr -i=2lx1.pdb --blocview=png-c3-label -o=2lx1.pml
pymol -Qkc 2lx1.pml # create: 2lx1-pymol.png
convert -trim +repage -border 10 -bordercolor white 2lx1-pymol.png 2lx1.png

# Figure S5C, a continuous stack of six bases
x3dna-dssr -i=2lx1-6bases.pdb --blocview=png-c3-label -o=2lx1-6bases.pml
pymol -Qkc 2lx1-6bases.pml # create: 2lx1-6bases-pymol.png
convert -trim +repage -border 10 -bordercolor white 2lx1-6bases-pymol.png 2lx1-6bases.png
```

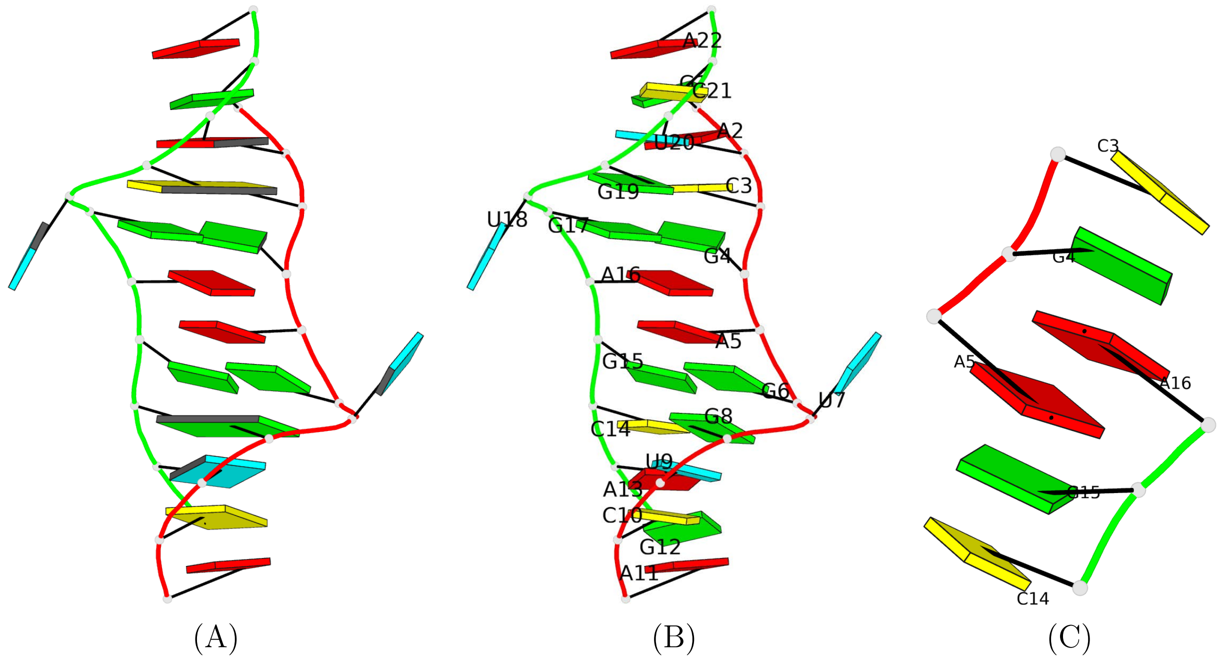


Figure S5: Cartoon-block schematics of the 6-base stack derived by DSSR from PDB entry `2lx1`. The commands used to create the images are shown in Listing 5, with the `--blocview` option for the most extended views. (A) The whole structure with WC-pair blocks and black minor grooves to highlight the two double-helical stems. (B) The whole structure with base blocks and nucleotide *labels* to show the 6-base stack in context. (C) The stacked ensemble of the six nucleotides only, in the most extended view.

3.6.3. Base multiplets (using G-tetrad as an example)

Another DSSR-derived feature is 'base multiplet', defined as "three or more bases associated in a coplanar geometry via a network of hydrogen-bonding interactions." (Lu *et al.*, 2015). The G-tetrad motif in G-quadruplexes, where four guanines are associated via four consecutive G+G[6] pairs in a square planar geometry, is a special type of multiplets. This section uses PDB entry `6r9k` (Karg *et al.*, 2019) to illustrate the procedures on how to identify and visualize a G-tetrad.

Listing 6 shows the commands used to create the four images in Figure S6, using the 2nd G-tetrad in `6r9k` as an example. For the above illustration of stacked bases (Listing 5 and Figure S5), we have used the `--blocview` option to get the most extended view and solid colored blocks with black outlines. Here we employ the `--cartoon-block` option, together with other settings and DSSR-derived features, to demonstrate additional visualization styles that are especially useful for depicting a G-tetrad.

Other than cartoon schematics as in the previous figures, Figure S6A combines the traditional ball-and-stick (`stick`) molecular image with filled base rings and auto-detected H-bonds (`fill-hbond`). The large square block for a G-tetrad (here in blue outline) can be overlaid on top of the molecular image (Figure S6B). The G-tetrad blocks, unique to DSSR, greatly simplify the visualization of complicated G-quadruplexes (see Figure S1, and the detailed steps for generating Figure S4C). By default, DSSR represents the R block as a square (see Figure S1), which is effective for visualizing base-pairing and stacking interactions (Figure S6C, see also Figures S4 and S5). With square-R blocks, however, the circular H-bonding directionality (from the WC-edge to the major-groove edge, i.e., the Hoogsteen-edge) of the G-tetrad is *not* clearly noticeable. That is where the `slim` block setting comes in, as illustrated in Figure S6D. Moreover, DSSR makes the `syn`-conformation of the modified nucleotide BGM (8-bromo-2′-deoxyguanosine) stand out by coloring it differently (in teal by default).

Note that in Figure S6, the images are all in the same view, defined by the original atomic coordinates in file `6r9k-Gtetrad2.pdb`. The ray-traced PNG images directly from PyMOL are all named `6r9k-Gtetrad2-pymol.png`, derived from the same PDB coordinate file. The `convert` steps of Listing 6 generate the different images used in the figure.

[6]The plus symbol denotes the parallel directionality of z-axes of the standard reference frames of the two guanines in the pair. In contrast, the two bases in a WC pair (e.g., G–C) are anti-parallel. The distinction of $\uparrow\uparrow$ (+) vs. $\uparrow\downarrow$ (−) pairs is unique to 3DNA/DSSR. Please refer to the publications on 3DNA (Lu and Olson, 2003) and DSSR (Lu *et al.*, 2015), and the DSSR User Manual for further details.

Listing 6: Commands used to create the images of a G-tetrad in Figure S6

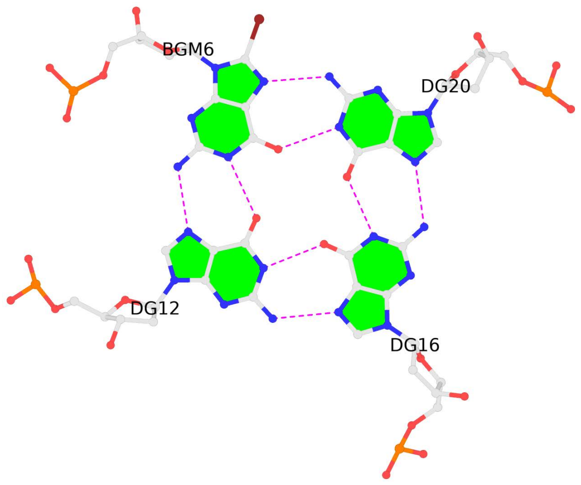
```
mkdir -p example3; cd example3

curl https://files.rcsb.org/download/6r9k.pdb -o 6r9k.pdb
x3dna-dssr -i=6r9k.pdb -o=6r9k.out # main output file, plus many auxiliary files
cp -f dssr-multiplets.pdb 6r9k-multiplets.pdb # fixed name dssr-multiplets.pdb
x3dna-dssr -i=6r9k-multiplets.pdb --select-model=2 -o=6r9k-Gtetrad2.pdb # the 2nd G-tetrad

# Figure S6A, with filled base rings and H-bonds, no blocks
x3dna-dssr -i=6r9k-Gtetrad2.pdb --cartoon-block=png-sticks-label --block-file=fill-hbond -o=6r9k-img1.pml
pymol -Qkc 6r9k-img1.pml # create: 6r9k-Gtetrad2-pymol.png
convert -trim +repage -border 10 -bordercolor white 6r9k-Gtetrad2-pymol.png 6r9k-img1.png

# Figure S6B, as Figure S6A, but with an outline of the G-tetrad square block in blue
x3dna-dssr -i=6r9k-Gtetrad2.pdb --cartoon-block=png-sticks-label --block-file=fill-hbond-g4-edge \
          --block-color='G4:blue' -o=6r9k-img2.pml
pymol -Qkc 6r9k-img2.pml # create: 6r9k-Gtetrad2-pymol.png
convert -trim +repage -border 10 -bordercolor white 6r9k-Gtetrad2-pymol.png 6r9k-img2.png

# Figure S6C, with H-bonds (but no 'fill') and default blocks for guanines
x3dna-dssr -i=6r9k-Gtetrad2.pdb --cartoon-block=png-sticks-label --block-file=hbond-syn-face-edge \
          -o=6r9k-img3.pml
pymol -Qkc 6r9k-img3.pml # create: 6r9k-Gtetrad2-pymol.png
convert -trim +repage -border 10 -bordercolor white 6r9k-Gtetrad2-pymol.png 6r9k-img3.png

# Figure S6D, as Figure S6C, but using slim blocks for guanines
x3dna-dssr -i=6r9k-Gtetrad2.pdb --cartoon-block=png-sticks-label --block-file=hbond-slim-syn-face-edge \
          -o=6r9k-img4.pml
pymol -Qkc 6r9k-img4.pml # create: 6r9k-Gtetrad2-pymol.png
convert -trim +repage -border 10 -bordercolor white 6r9k-Gtetrad2-pymol.png 6r9k-img4.png
```

4. The `dssr_block` PyMOL plugin

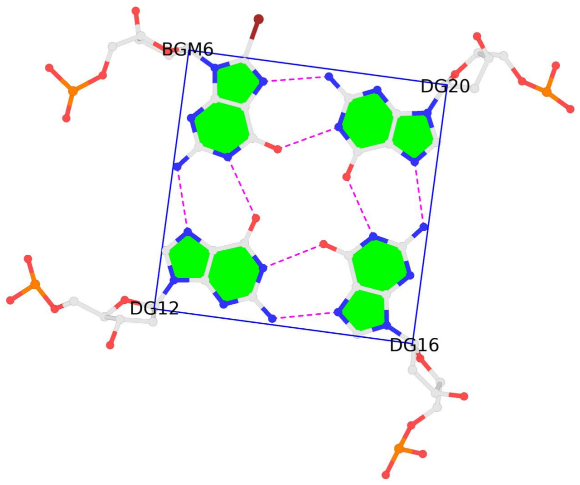
In early 2015, Thomas Holder (PyMOL Principal Developer, Schrödinger, Inc.) and I agreed to work together on connecting DSSR to PyMOL. From the very beginning, we had envisioned that the DSSR-PyMOL integration could include two components: to bring DSSR-derived RNA/DNA structural features into the selection mechanism of PyMOL, and to render the unique and informative DSSR base-rectangular block representations directly in PyMOL. The latter feature was implemented by Thomas as a PyMOL plugin, via the `dssr_block.py` Python script. Specifically, the plugin adds the `dssr_block` command to PyMOL for *interactive* creation of base blocks in various styles.

```
dssr_block [ selection [, state [, block_file [, block_depth [, block_color [, name [, exe ]]]]]]]
```

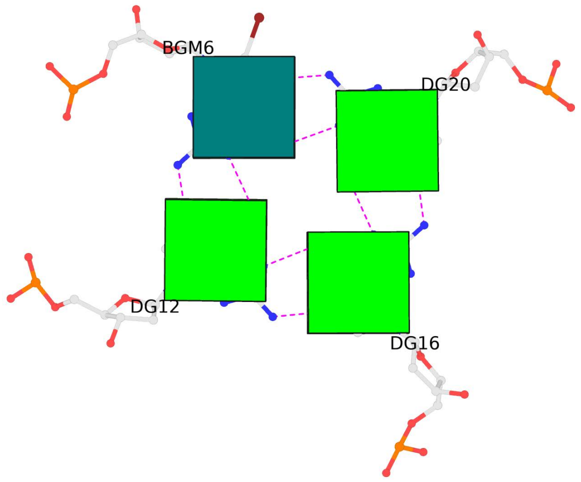
Other than the `dssr_block` command itself, all other arguments are optional (as hinted by the square brackets). The optional arguments make the plugin versatile: in addition to the three DSSR options (`block_file`, `block_depth`, and `block_color`), the plugin command



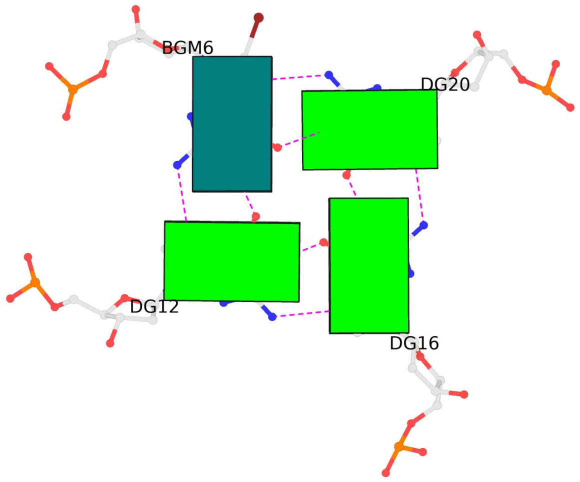
(A) Filled base rings, plus H-bonds



(B) As in (A), plus a G4 block



(C) Guanines as default R blocks



(D) Guanines as `slim` R blocks

Figure S6: Molecular images of a G-tetrad in PDB entry `6r9k`, with overlaid blocks of different styles. The commands used to create the images are shown in Listing 6, with the `--cartoon-block` option to retain the original orientation. The images in (A) and (B) will not be possible with the `--blocview` option since it will always enable solid colored blocks. The directionality of the circular H-bonding orientation of the G-tetrad becomes obvious when the default square-R blocks (C) are replaced by the oblong variants (D), enabled by keyword `slim` in the `--block-file` option. The `syn` conformation of the modified nucleotide 8-bromo-2′-deoxyguanosine (`BGM`, in teal color) is also unmistakably clear.

takes atom `selection`, object `state`, a customized `name` of the created CGO, and the specific path to the executable (`x3dna-dssr`). As one may guess, the `block_file` argument here corresponds to the DSSR `--block-file` option, and so are `block_depth` vs. `--block-depth` and `block_color` vs. `--block-color` (see Footnote 4).

The `dssr_block.py` script is short, consisting of 123 lines including documentation, with only ∼50 line-of-code (see Appendix A). The source code is clearly written and is easy to understand, even for non-Python programmers. Reading the code is necessary for understanding how the plugin works behind the hood. Essentially, it sends the atomic coordinates of the current PyMOL *selection* (default to *all*), together with `block_file`, `block_depth`, or `block_color` options (if specified) to DSSR, and collects the resultant blocks in `.r3d` format for rendering. Thus, it should be clear that all DSSR features available via the corresponding command-line options are also accessible via the PyMOL plugin: `block_file=fill-hbond`, for example, can be used to fill base rings and draw H-bonds (see Figure S6A), even though these (relatively) new features are *not* yet documented for the plugin.

Listing 7: Usages of the DSSR-plugin for PyMOL on PDB entry `2lx1` (Figure S7)

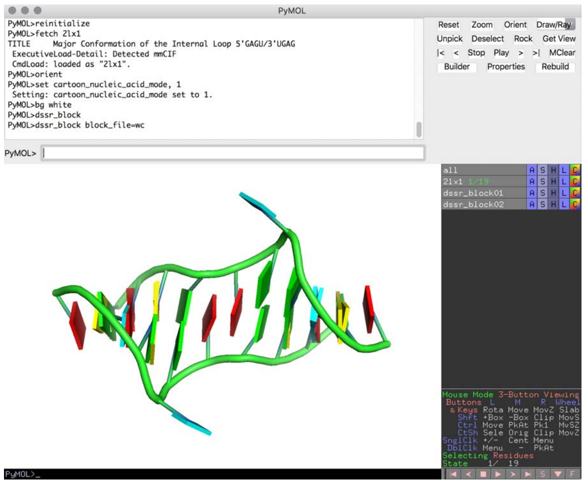
```
 1  reinitialize
 2  run dssr_block.py
 3
 4  # Figure S7A, two CGO objects overlaid
 5  fetch 2lx1
 6  orient
 7  set cartoon_nucleic_acid_mode, 1
 8  bg_color white
 9  dssr_block
10  dssr_block block_file=wc
11
12  # Figure S7B, an aggregate image of the NMR ensemble
13  delete dssr_block0*
14  set all_states, 1
15  dssr_block state=0
16  ray 1800
17  png 2lx1-ensemble-pymol.png
```

Figure S7A is a screenshot generated using the PyMOL commands in line nos. 5–10 of Listing 7, based on PDB entry `2lx1`. Two block representations were created: one in the default settings using simply the `dssr_block` command (line no. 9), and the other with the additional `block_file=wc` argument to draw WC-pair blocks (line no. 10). The two CGO objects (named `dssr_block01` and `dssr_block02`, respectively, by default) are overlaid. Either can be conveniently toggled on and off, as for any other PyMOL objects, interactively.

Figure S7B shows an aggregate image of 19 models in the NMR ensemble. It was generated with PyMOL commands in line nos. 13–17 of Listing 7: (1) The two CGO objects from previous steps are deleted (line no. 13). (2) All states (models) are displayed (line no. 14). (3) The base blocks for all states are created (line no. 15). (4) The scenes are ray-traced and a PNG image is created (line nos. 16–17), as shown in the examples in Section 3.

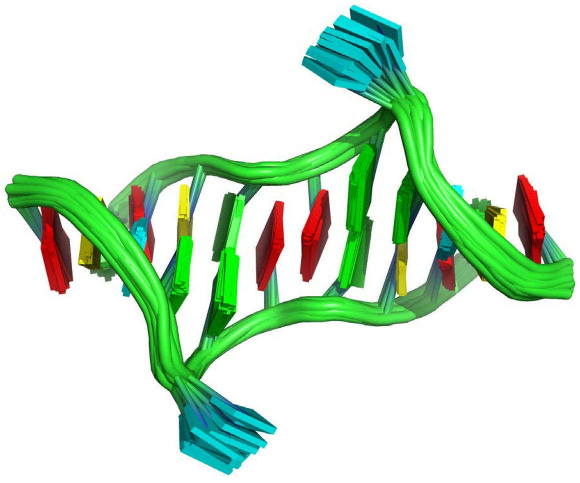
The commands in Listing 7 can be typed one-by-one into a PyMOL session to create the two images in Figure S7. The PyMOL interactive process, combining a graphical user interface (GUI) and a command input area, is convenient for learning and exploration. The `dssr_block` PyMOL plugin makes DSSR transparent to users who are not familiar with the CLI. On the other hand, PyMOL commands (e.g., Listing 7) can be put into a `.pml` file to automate the image generation process (as demonstrated in Section 3). To ensure reproducibility, Listing 7 first calls `reinitialize` (line no. 1), and it then runs the `dssr_block.py` script (line no. 2) to bring the `dssr_block` command available for later use.

Overall, the `dssr_block` plugin brings RNA/DNA-specific block schematics from DSSR into the popular PyMOL molecular visualization system. By combining novel DSSR blocks with classic PyMOL representations, users can easily create publication quality images that highlight base-pairing and stacking interactions in nucleic acid structures. By design, the DSSR plugin allows for addition of new features without breaking the interface.


PyMOL
Reset Zoom Orient Draw/Ray
Unpick Deselect Rock Get View
|< < Stop Play > >| MClear
Builder Properties Rebuild
all
21x1
dssr_block01
dssr_block02

```
PyMOL>reinitialize
PyMOL>fetch 2lx1
TITLE     Major Conformation of the Internal Loop 5'GAGU/3'UGAG
 ExecutiveLoad-Detail: Detected mmCIF
 CmdLoad: loaded as "2lx1".
PyMOL>orient
PyMOL>set cartoon_nucleic_acid_mode, 1
 Setting: cartoon_nucleic_acid_mode set to 1.
PyMOL>bg white
PyMOL>dssr_block
PyMOL>dssr_block block_file=wc
```

(A) Overlaid base and WC-pair blocks

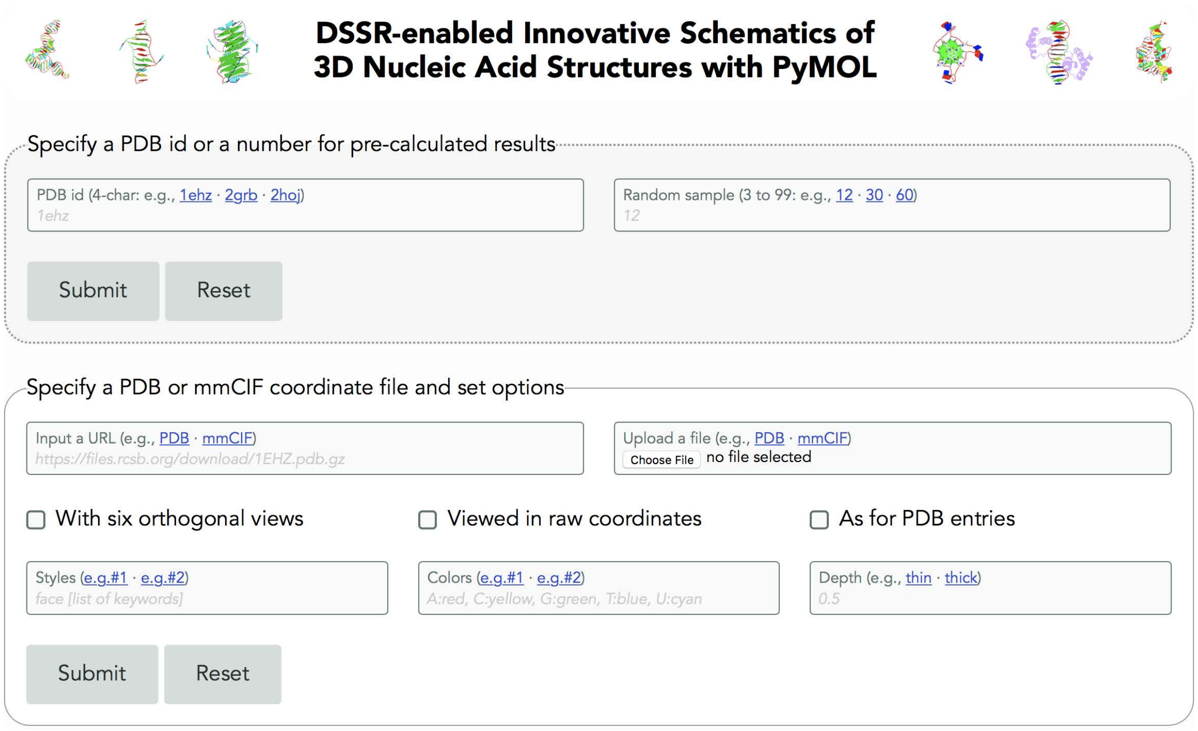


(B) An aggregate of the NMR ensemble

Figure S7: Screenshots of running the DSSR-PyMOL plugin on PDB entry `2lx1`. The images were created using PyMOL commands in Listing 7. (A) Screenshot showing the commands typed in, the graphics with base and WC-pair blocks overlaid, and the objects created. (B) Ray-traced PNG image of the whole NMR ensemble (with extra whitespace cropped).

5. The web application

To make DSSR-PyMOL schematics even more easily and widely accessible, I have created http://skmatic.x3dna.org, a cross-platform and web-based application. The site runs on Ubuntu Linux, employing the Apache HTTP server, the Phusion Passenger application server, and Roda, a routing-tree toolkit for building web applications in Ruby. Other significant software tools include: the Cutestrap CSS framework, the jQuery (http://jquery.com) JavaScript library and its three plugins (jBox for tooltips/images gallery, Validation for client-side form validations, and toc for generating a table of contents), and 3Dmol.js (Rego and Koes, 2015) for interactive display of 3D models. Implemented with standard web technologies, the client-side user interface works in all modern web browsers (e.g., Chrome, Safari, or Firefox) across a wide spectrum of platforms: Windows, macOS, or Linux on desktops and laptops; iOS or Android on tablets and smartphones.


DSSR-enabled Innovative Schematics of 3D Nucleic Acid Structures with PyMOL

Specify a PDB id or a number for pre-calculated results

PDB id (4-char: e.g., 1ehz · 2grb · 2hoj)
1ehz

Random sample (3 to 99: e.g., 12 · 30 · 60)
12

Submit Reset

Specify a PDB or mmCIF coordinate file and set options

Input a URL (e.g., PDB · mmCIF)
https://files.rcsb.org/download/1EHZ.pdb.gz

Upload a file (e.g., PDB · mmCIF)
Choose File no file selected

☐ With six orthogonal views ☐ Viewed in raw coordinates ☐ As for PDB entries

Styles (e.g.#1 · e.g.#2)
face [list of keywords]

Colors (e.g.#1 · e.g.#2)
A:red, C:yellow, G:green, T:blue, U:cyan

Depth (e.g., thin · thick)
0.5

Submit Reset

Figure S8: Homepage of the web application for DSSR-PyMOL schematics. On the title line, clicking an image would show a gallery of 12 representative PDB schematics. The form for pre-calculated PDB entries contains two input fields, with straightforward meanings. The bottom form allows for the specification of an atomic coordinate file, along with customizations. Hovering over an item to see its tooltip. The web application is simple and intuitive: it takes no time to get started.

The web application for DSSR-PyMOL schematics consists of two components (Figure S8), as detailed below. The first part covers pre-calculated PDB entries, a very common use case. The second one represents the general application, necessitating the input of an atomic coordinate file, together with optional customizations via selections of checkboxes or block settings using text input boxes. Hovering over a form field displays the corresponding tooltip, and clicking an example link would fill out the matching field with a typical use case. Overall, the web application has been designed with simplicity in mind. Even novice or occasional users can quickly get started and benefit from it.

5.1. Pre-calculated PDB entries

For user convenience, the web application includes pre-calculated images for PDB entries. Figure S9 shows a screenshot for PDB entry `2lx1` (Kennedy *et al.*, 2012). The results include two parts:

- The top portion provides summary information and the primary citation (including the abstract, if available) compiled from PDB and PubMed. It also contains DSSR-derived structural features in human-readable text and machine-friendly JSON formats.
- The second part presents cartoon-block schematics in six orthogonal views (downloadable as a tarball): the `front` view (top-left image) is created with the DSSR options `--blocview=png-organic-c3` and `--block-file=wc-minor-g4`. The PyMOL session file for this image is available. The `blocview`-transformed PDB coordinate file can also be downloaded, or visualized directly using 3Dmol.js (Rego and Koes, 2015). The other five images are derived from this orientation via camera rotations in PyMOL, as follows, respectively:
  - `right` view (top-middle): `turn y, -90`
  - `top` view (top-right): `turn x, 90`
  - `back` view (bottom-left): `turn y, 180`
  - `left` view (bottom-middle): `turn y, 90`
  - `bottom` view (bottom-right): `turn x, -90`

As made clear in Figure S9, the meta data and the six perspectives (`front`, `right`, `top`; `back`, `left`, `bottom`) give users a quick and simplified overview of a PDB entry. With the session file, users can also fine-tune PyMOL settings to customize the rendered image.

For simplicity and other practical considerations, the pre-calculated schematics do *not* include every nucleic-acid-containing PDB entry. Specifically, the following three restrictions are applied: (1) Only structures with PDB-formatted coordinate files (`.pdb`) are incorporated. Thus, huge structures (available exclusively in `.cif` format in the PDB) such as `6pj6` (Figure S4F) are excluded. (2) For a crystal structure, only the asymmetric unit is processed, *not* the biological unit. (3) For an NMR ensemble, only the first model is used (Figure S7).

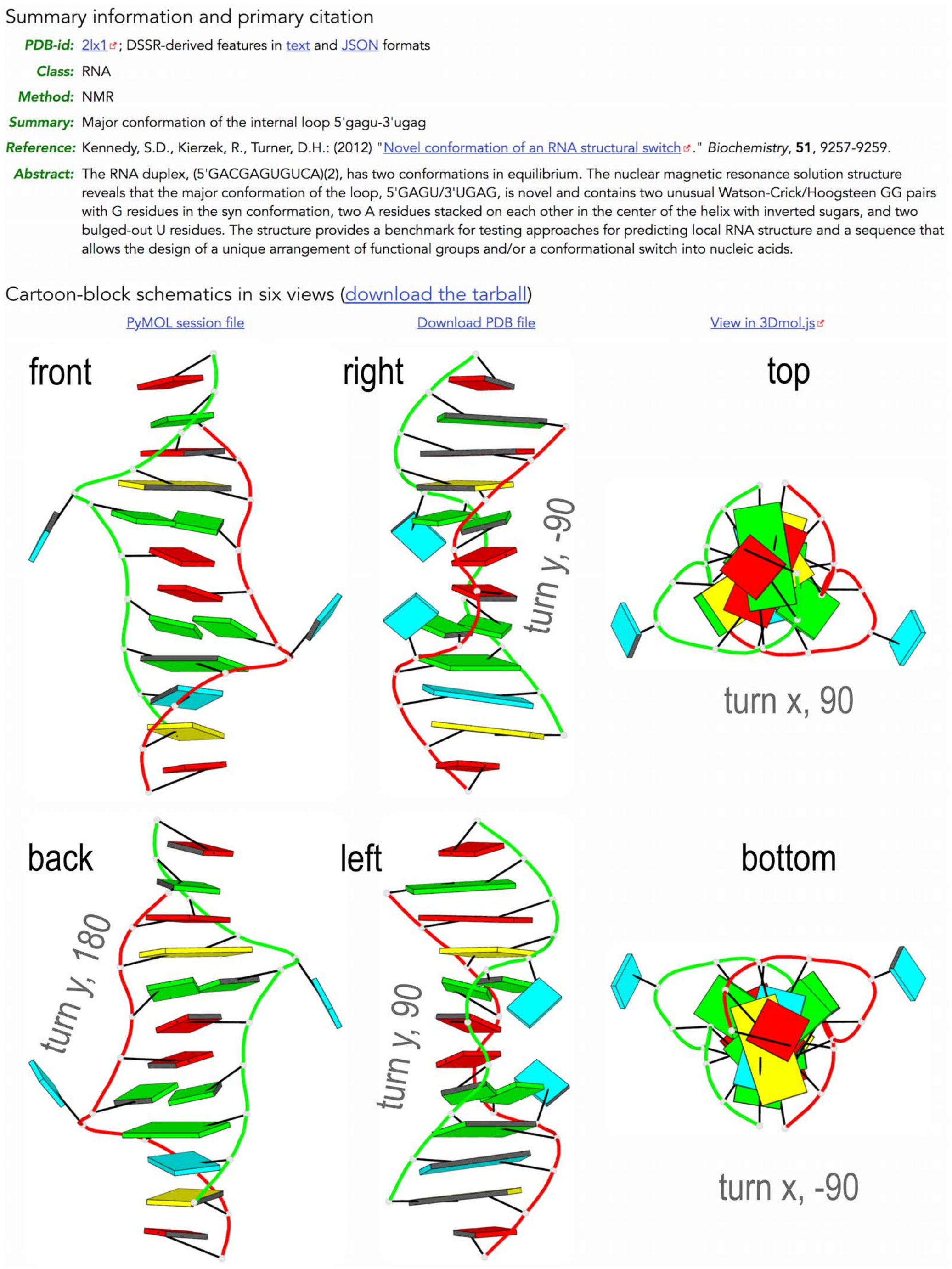
The form also provides an input field to display a random sample of schematics from pre-calculated PDB entries. The sample can range from 3 to 99 PDB entries, with a default size of 12. All images are oriented in the `front` view, as defined by `--blocview` (see above). Clicking an image in the sample page leads to the corresponding PDB entry, with details as shown in Figure S9 for `2lx1`.

5.2. User-supplied coordinate files

This part allows users to specify an atomic coordinate file (in either `.pdb` or `.cif` format) via a URL or direct upload. The coordinate file can be optionally gzipped (with the `.gz` suffix), with a size limit of 6 MB. The schematic styles are controlled by the three input boxes at the bottom, corresponding to the DSSR options `--block-file`, `--block-color`. and `--block-depth`, respectively (Section 3). By default, only one image is created (Figure S3): if the checkbox '`Viewed in raw coordinates`' is turned on, the image is in the original view (via `--cartoon-block`); otherwise it is in the extended view (via `--blocview`). When the checkbox '`With six orthogonal views`' is selected, six images will be generated. Finally, if the checkbox '`As for PDB entries`' is chosen, the settings for pre-calculated PDB entries (Section 5.1) are employed: the result would be six cartoon-block images as those shown in Figure S9 (void of the section of summary and citation).

6. The web API

The web API offers a programmatic means to generate DSSR-enhanced schematics with PyMOL, without any installation. It has been implemented in Ruby, using the Roda routing-tree web toolkit (Section 5). The main access endpoint is `http://skmatic.x3dna.org/api`. For a pre-calculated PDB entry (e.g., `2lx1`), the PNG image in `front` view (`--blocview`) is accessible via `http://skmatic.x3dna.org/api/pdb/2lx1` (Figure S9). Detailed options and usage examples are available at `http://skmatic.x3dna.org/api/help` (see Listing 8 for an abbreviated version, using `curl` for the examples).



Summary information and primary citation

PDB-id: 2lx1; DSSR-derived features in text and JSON formats

Class: RNA

Method: NMR

Summary: Major conformation of the internal loop 5'gagu-3'ugag

Reference: Kennedy, S.D., Kierzek, R., Turner, D.H.: (2012) "Novel conformation of an RNA structural switch." *Biochemistry*, **51**, 9257-9259.

Abstract: The RNA duplex, (5'GACGAGUGUCA)(2), has two conformations in equilibrium. The nuclear magnetic resonance solution structure reveals that the major conformation of the loop, 5'GAGU/3'UGAG, is novel and contains two unusual Watson-Crick/Hoogsteen GG pairs with G residues in the syn conformation, two A residues stacked on each other in the center of the helix with inverted sugars, and two bulged-out U residues. The structure provides a benchmark for testing approaches for predicting local RNA structure and a sequence that allows the design of a unique arrangement of functional groups and/or a conformational switch into nucleic acids.

Cartoon-block schematics in six views (download the tarball)

Figure S9: Pre-calculated DSSR-PyMOL schematics for PDB entry `2lx1`. In the screenshot, the `front` view (top-left image) is based on the `--blocview` transformation, with corresponding PyMOL session file and PDB coordinate file available for download. The other five views are annotated with PyMOL transformations related to the `front` (extended) view.

```
curl http://skmatic.x3dna.org/api -F 'model=@2lx1.pdb' -F 'block_file=wc-minor' -o 2lx1.png
```

To create a customized image, users need to specify an atomic coordinate files (see Section 5.2 for acceptable formats and size limit), along with optional settings. Specifically, the *local* coordinate file needs to be HTTP `post`-ed to the server and named `model`. In the listing above, the `curl -F` (or `--form`) option is used to `post` form content to the server, and the `@` sign specifies a file (as in `model=@2lx1.pdb`). Please see the tutorial "Using `curl` to automate HTTP jobs" for more details.

The three options `block_file`, `block_color`, and `block_depth` have obvious meanings. With the `r3d_file` option set to `true` (`yes`, `on`, or `1`), the web API outputs a `.r3d` file instead of a PNG image. Finally, when the `raw_xyz` option is switched on, the API returns the output image in its original view (`--cartoon-block`) instead of the extended view (`--blocview`) (see Figure S3).

Listing 8: Usage of the DSSR-PyMOL web API

```
Usage
   http://skmatic.x3dna.org/api [options] required-model-file
   http://skmatic.x3dna.org/api/pdb/pdb_id # for a pre-calculated PDB entry
   http://skmatic.x3dna.org/api/help # display a detailed help message
Options
   block_file=styles-as-keywords-list # block_file=wc-minor
   block_color=selection-and-color # block_color='A:pink,G:gray'
   block_depth=thickness-of-block # block_depth=1.2
   r3d_file=true-or-false(default) # r3d_file=true (or 1 or yes)
   raw_xyz=true-or-false(default) # raw_xyz=true
Required parameter
   url=URL-to-coordinate-file # url=https://files.rcsb.org/download/1ehz.pdb.gz
   model=@coordinate-file # model=@1ehz.cif
   # Only one must be specified. 'url' takes precedence over 'model' when both are specified.
   # The coordinate file must be in PDB or PDBx/mmCIF format, optionally gzipped.
Examples
   curl http://skmatic.x3dna.org/api -F 'model=@1msy.pdb' -F 'block_file=wc-minor' -F 'r3d_file=1'
   curl 'http://skmatic.x3dna.org/api?r3d_file=1&block_file=wc-minor' -F 'model=@1msy.pdb' # as above
   curl http://skmatic.x3dna.org/api -F 'url=https://files.rcsb.org/download/1ehz.pdb.gz' -o 1ehz.png
   curl http://skmatic.x3dna.org/api/pdb/1ehz -o 1ehz.png
```

References

Antczak,M., Zablocki,M., Zok,T., Rybarczyk,A., Blazewicz,J. and Szachniuk,M. (2019) RNAvista: a webserver to assess RNA secondary structures with non-canonical base pairs. *Bioinformatics,* **35** (1), 152–155.

Baulin,E., Yacovlev,V., Khachko,D., Spirin,S. and Roytberg,M. (2016) URS DataBase: universe of RNA structures and their motifs. *Database,* **2016**, baw085.

Bayrak,C.S., Kim,N. and Schlick,T. (2017) Using sequence signatures and kink-turn motifs in knowledge-based statistical potentials for RNA structure prediction. *Nucleic Acids Res.,* **45** (9), 5414–5422.

Berger,K.D., Kennedy,S.D. and Turner,D.H. (2019) Nuclear magnetic resonance reveals that GU base pairs flanking internal loops can adopt diverse structures. *Biochemistry,* **58** (8), 1094–1108.

Burley,S.K., Berman,H.M., Christie,C., Duarte,J.M., Feng,Z., Westbrook,J., Young,J. and Zardecki,C. (2018) RCSB Protein Data Bank: sustaining a living digital data resource that enables breakthroughs in scientific research and biomedical education. *Protein Sci.,* **27** (1), 316–330.

Calladine,C.R. and Drew,H.R. (1984) A base-centred explanation of the B-to-A transition in DNA. *J. Mol. Biol.,* **178** (3), 773–782.

Calladine,C.R., Drew,H.R., Luisi,B.F. and Travers,A.A. (2004) *Understanding DNA: the molecule & how it works.* Third edition, Elsevier, Amsterdam.

Chu,B., Zhang,D. and Paukstelis,P.J. (2019) A DNA G-quadruplex/i-motif hybrid. *Nucleic Acids Res.,* **47** (22), 11921–11930.

Couch,G.S. (2006) Nucleic acid visualization with UCSF Chimera. *Nucleic Acids Res.,* **34** (4), e29.

Cuturello,F., Tiana,G. and Bussi,G. (2020) Assessing the accuracy of direct-coupling analysis for RNA contact prediction. *RNA,* **26** (5), 637–647.

Desai,N., Brown,A., Amunts,A. and Ramakrishnan,V. (2017) The structure of the yeast mitochondrial ribosome. *Science,* **355** (6324), 528–531.

Edwards,T.E. and Ferré-D'Amaré,A.R. (2006) Crystal structures of the thi-box riboswitch bound to thiamine pyrophosphate analogs reveal adaptive RNA-small molecule recognition. *Structure,* **14** (9), 1459–1468.

Fuchsbauer,O., Swuec,P., Zimberger,C., Amigues,B., Levesque,S., Agudelo,D., Duringer,A., Chaves-Sanjuan,A., Spinelli,S., Rousseau,G.M., Velimirovic,M., Bolognesi,M., Roussel,A., Cambillau,C., Moineau,S., Doyon,Y. and Goulet,A. (2019) Cas9 allosteric inhibition by the anti-CRISPR protein AcrIIA6. *Mol. Cell,* **76** (6), 922–937.e7.

Gallego,D., Darré,L., Dans,P.D. and Orozco,M. (2019) VeriNA3d: an R package for nucleic acids data mining. *Bioinformatics,* **35** (24), 5334–5336.

Giambaşu,G.M., Case,D.A. and York,D.M. (2019) Predicting site-binding modes of ions and water to nucleic acids using molecular solvation theory. *J. Am. Chem. Soc.,* **141** (6), 2435–2445.

Hanson,R.M. and Lu,X.J. (2017) DSSR-enhanced visualization of nucleic acid structures in Jmol. *Nucleic Acids Res.,* **45** (W1), W528–W533.

Karg,B., Mohr,S. and Weisz,K. (2019) Duplex-guided refolding into novel G-quadruplex (3+1) hybrid conformations. *Angew. Chem. Int. Ed.,* **58** (32), 11068–11071.

Kennedy,S.D., Kierzek,R. and Turner,D.H. (2012) Novel conformation of an RNA structural switch. *Biochemistry,* **51** (46), 9257–9259.

Kraulis,P.J. (1991) MolScript: a program to produce both detailed and schematic plots of protein structures. *J. Appl. Crystallogr.,* **24** (5), 946–950.

Kribelbauer,J.F., Lu,X.J., Rohs,R., Mann,R.S. and Bussemaker,H.J. (2020) Toward a mechanistic understanding of DNA methylation readout by transcription factors. *J. Mol. Biol.,* **432** (6), 1801–1815.

Li,S., Olson,W.K. and Lu,X.J. (2019) Web 3DNA 2.0 for the analysis, visualization, and modeling of 3D nucleic acid structures. *Nucleic Acids Res.,* **47** (W1), W26–W34.

Lindow,N., Baum,D., Leborgne,M. and Hege,H.C. (2019) Interactive visualization of RNA and DNA structures. *IEEE Trans. Vis. Comput. Graph.,* **25** (1), 967–976.

Lu,X.J., Bussemaker,H.J. and Olson,W.K. (2015) DSSR: an integrated software tool for dissecting the spatial structure of RNA. *Nucleic Acids Res.,* **43** (21), e142.

Lu,X.J., El Hassan,M. and Hunter,C. (1997*a*) Structure and conformation of helical nucleic acids: analysis program (SCHNAaP). *J. Mol. Biol.,* **273** (3), 668–680.

Lu,X.J., El Hassan,M. and Hunter,C. (1997*b*) Structure and conformation of helical nucleic acids: rebuilding program (SCHNArP). *J. Mol. Biol.,* **273** (3), 681–691.

Lu,X.J. and Olson,W.K. (2003) 3DNA: a software package for the analysis, rebuilding and visualization of three-dimensional nucleic acid structures. *Nucleic Acids Res.,* **31** (17), 5108–5121.

Lu,X.J. and Olson,W.K. (2008) 3DNA: a versatile, integrated software system for the analysis, rebuilding and visualization of three-dimensional nucleic-acid structures. *Nat. Protoc.,* **3** (7), 1213–1227.

Meier,M., Moya-Torres,A., Krahn,N.J., McDougall,M.D., Orriss,G.L., McRae,E.K.S., Booy,E.P., McEleney,K., Patel,T.R., McKenna,S.A. and Stetefeld,J. (2018) Structure and hydrodynamics of a DNA G-quadruplex with a cytosine bulge. *Nucleic Acids Res.,* **46** (10), 5319–5331.

Merritt,E.A. and Bacon,D.J. (1997) Raster3D: photorealistic molecular graphics. *Methods Enzymol.,* **277**, 505–524.

Narayanan,B.C., Westbrook,J., Ghosh,S., Petrov,A.I., Sweeney,B., Zirbel,C.L., Leontis,N.B. and Berman,H.M. (2014) The Nucleic Acid Database: new features and capabilities. *Nucleic Acids Res.,* **42** (D1), D114–D122.

Olson,W.K., Bansal,M., Burley,S.K., Dickerson,R.E., Gerstein,M., Harvey,S.C., Heinemann,U., Lu,X.J., Neidle,S., Shakked,Z., Sklenar,H., Suzuki,M., Tung,C.S., Westhof,E., Wolberger,C. and Berman,H.M. (2001) A standard reference frame for the description of nucleic acid base-pair geometry. *J. Mol. Biol.,* **313** (1), 229–237.

Rego,N. and Koes,D. (2015) 3Dmol.js: molecular visualization with WebGL. *Bioinformatics,* **31** (8), 1322–1324.

Sagendorf,J.M., Markarian,N., Berman,H.M. and Rohs,R. (2020) DNAproDB: an expanded database and web-based tool for structural analysis of DNA–protein complexes. *Nucleic Acids Res.,* **48** (D1), D277–D287.

Stojković,V., Myasnikov,A.G., Young,I.D., Frost,A., Fraser,J.S. and Fujimori,D.G. (2020) Assessment of the nucleotide modifications in the high-resolution cryo-electron microscopy structure of the Escherichia coli 50S subunit. *Nucleic Acids Res.,* **48** (5), 2723–2732.

Tan,D., Piana,S., Dirks,R.M. and Shaw,D.E. (2018) RNA force field with accuracy comparable to state-of-the-art protein force fields. *Proc. Natl. Acad. Sci.,* **115** (7), E1346–E1355.

Thiel,B.C., Beckmann,I.K., Kerpedjiev,P. and Hofacker,I.L. (2019) 3D based on 2D: calculating helix angles and stacking patterns using forgi 2.0, an RNA Python library centered on secondary structure elements. *F1000Research,* **8**, 287.

Yesselman,J.D., Eiler,D., Carlson,E.D., Gotrik,M.R., d'Aquino,A.E., Ooms,A.N., Kladwang,W., Carlson,P.D., Shi,X., Costantino,D.A., Herschlag,D., Lucks,J.B., Jewett,M.C., Kieft,J.S. and Das,R. (2019) Computational design of three-dimensional RNA structure and function. *Nat. Nanotechnol.,* **14** (9), 866–873.

Zhang,H., Endrizzi,J.A., Shu,Y., Haque,F., Sauter,C., Shlyakhtenko,L.S., Lyubchenko,Y., Guo,P. and Chi,Y.I. (2013) Crystal structure of 3WJ core revealing divalent ion-promoted thermostability and assembly of the phi29 hexameric motor pRNA. *RNA,* **19** (9), 1226–1237.

Zok,T., Antczak,M., Zurkowski,M., Popenda,M., Blazewicz,J., Adamiak,R.W. and Szachniuk,M. (2018) RNApdbee 2.0: multifunctional tool for RNA structure annotation. *Nucleic Acids Res.,* **46** (W1), W30–W35.

Zok,T., Popenda,M. and Szachniuk,M. (2020) ElTetrado: a tool for identification and classification of tetrads and quadruplexes. *BMC Bioinformatics,* **21** (1), 40.

A. Source code listing of the dssr_block PyMOL plugin

```python
'''
http://pymolwiki.org/index.php/dssr_block

(c) Thomas Holder, Schrodinger LLC

License: BSD-2
'''

from pymol import cmd, CmdException

def unquote(s):
    s = str(s)
    if s.rstrip()[-1:] not in ('"', "'"):
        return s
    return cmd.safe_eval(s)

def dssr_block(selection='all', state=-1,
        block_file='face',
        block_depth=0.5,
        block_color='',
        name='',
        exe='x3dna-dssr',
        quiet=1):
    '''
DESCRIPTION

    Create a nucleic acid base "block" cartoon with DSSR.

    Requires the "x3dna-dssr" program, available from http://x3dna.org/

USAGE

    dssr_block [ selection [, state [, block_file [, block_depth
        [, block_color [, name [, exe ]]]]]]]

ARGUMENTS

    selection = str: atom selection {default: all}

    state = int: object state (0 for all states) {default: -1, current state}

    block_file = face|edge|wc|equal|minor|gray: Corresponds to the --block-file
    option (see DSSR manual). Values can be combined, e.g. "wc-minor".
    {default: face}

    block_depth = float: thickness of rectangular blocks {default: 0.5}

    block_color = str: Corresponds to the --block-color option (new in DSSR
    v1.5.2) {default: }

    name = str: name of new CGO object {default: dssr_block##}

    exe = str: path to "x3dna-dssr" executable {default: x3dna-dssr}

EXAMPLE

    fetch 1ehz, async=0
    as cartoon
    dssr_block
    set cartoon_ladder_radius, 0.1
    set cartoon_ladder_color, gray
    set cartoon_nucleic_acid_mode, 1

    # multi-state
    fetch 2n2d, async=0
    dssr_block 2n2d, 0
    set all_states

    # custom coloring
    fetch 1msy, async=0
    dssr_block block_color=N red | minor 0.9 | major yellow
    '''
    import subprocess
    import tempfile, os

    state, quiet = int(state), int(quiet)

    tmpfilepdb = tempfile.mktemp('.pdb')
    tmpfiler3d = tempfile.mktemp('.r3d')

    args = [exe,
        '--block-file=' + unquote(block_file),
        '--block-depth=' + unquote(block_depth),
        '-i=' + tmpfilepdb,
        '-o=' + tmpfiler3d,
    ]

    if block_color:
        args.append('--block-color=' + unquote(block_color))

    if not name:
        name = cmd.get_unused_name('dssr_block')

    states = [state] if state != 0 else \
            range(1, cmd.count_states(selection) + 1)

    try:
        for state in states:
            cmd.save(tmpfilepdb, selection, state)
            subprocess.check_call(args)
            cmd.load(tmpfiler3d, name, max(1, state), zoom=0)
    except subprocess.CalledProcessError:
        raise CmdException('"' + exe + '" failed')
    except OSError:
        raise CmdException('Cannot execute exe="' + exe + '"')
    finally:
        try:
            os.remove(tmpfilepdb)
            os.remove(tmpfiler3d)
        except OSError:
            pass

cmd.extend('dssr_block', dssr_block)

# tab-completion of arguments
cmd.auto_arg[0].update({
    'dssr_block' : cmd.auto_arg[0]['zoom'],
})
cmd.auto_arg[2].update({
    'dssr_block' : [cmd.Shortcut(['face', 'edge', 'wc', 'equal', 'minor', 'gray']), 'block_file', ''],
})

# vi: expandtab:smarttab
```