eda_1

October 11, 2022

1 Explorative Datenanalyse Part 1

1.1 Imports

Der Einfachheit halber regelt rnaloops.data_explorer.import_helper alle Imports. Alle Module liegen in einem Package rnaloops
und sollten nur außerhalb von diesem importiert werden (um relative import Probleme zu vermeiden).
Wir brauchen nur alle Module aus import_helper in einem neuen Notebook zu importieren und haben alles benötigte zur Verfügung.
User-definierte Funktionen zur EDA liegen z.B. in rnaloops.data_explorer.explore_fcts und werden automatisch importiert.

[1]:
```
from rnaloops.data_explorer.import_helper import *
```

```
imported pandas, numpy, seaborn, matplotlib and all functions from
data_explorer/explore_fcts.py
```

1.2 Daten laden und prüfen

Daten liegen in rnaloops/data in verschiedenen Formaten. Am besten ist .pkl um gegebenenfalls Metadaten der DatFrames mit zu speichern.
Es gibt eine Datei mit Daten nach parsing aus den pdfs (rnaloops_data) und eine aufbereitete (rnaloops_data_prepared). Hier ein Vergleich der Spalten:

[98]:
```
df_raw = get_df()  # Fct from getter_fcts.py to load parsed data
df = get_prepared_df()  # Fct from getter_fcts.py to load prepared data
compare_raw_and_prepared(df_raw, df)  # Fcts to compare them, i.e. RAM use

# Note on string memory reduction:
# Reduction in memory for string type columns due to switch from string to
# string arrow, which is shown as string by pandas. The difference emerges
# sice string arrow avoids python overhead on strings (the shorter the strings
# the better the improvement).

# Note on data getting:
# Data is aquired by screening online db in relevant index range and download
# from all pages showing structures. Infos are downloaded as pdf and first
# parsed in plain files, then in a single table. By this we retrieve all
```

```
# information on the structure provided by the online db, EXCEPT the first and
# second strand of the conection helices, which (for unknown reasons) is given
# in pdf, but has always value '-'. At his point we dont need that info.
# For more details on the process to get rnaloops_data.pkl check data_getter.py.
```

[98]:

	raw_name	new_name	raw_dtype	new_dtype
0	Index	Index	int64	int64
0	Loop type	loop_type	int64	category
1	Home structure (PDB id)	home_structure	string	category
2	DB notation	db_notation	string	string
3	Whole Sequence	whole_sequence	string	category
4	Length (bps) 1	helix_1_bps	float64	int8
18	Length (nts) 1	strand_1_nts	float64	int16
32	End position 1	end_1	string	string
46	Euler Angle X 1	euler_x_1	float64	float32
60	Start position 1	start_1	string	string
74	Euler Angle Y 1	euler_y_1	float64	float32
88	Euler Angle Z 1	euler_z_1	float64	float32
102	Planar Angle 1	planar_1	float64	float32
116	Sequence 1	sequence_1	string	category
Summe				

	raw_MB	prep_MB	diff_MB
0	0.644728	0.601776	0.042952
0	0.644728	0.076095	0.568633
1	4.926122	0.261239	4.664883
2	13.277000	3.920161	9.356839
3	13.277000	0.552349	12.724651
4	0.644728	0.075222	0.569506
18	0.644728	0.150444	0.494284
32	4.931360	0.666066	4.265294
46	0.644728	0.300888	0.343840
60	4.929873	0.664675	4.265198
74	0.644728	0.300888	0.343840
88	0.644728	0.300888	0.343840
102	0.644728	0.300888	0.343840
116	5.093628	0.236890	4.856738
Summe	297.779877	41.359004	256.420873

Wir sehen nicht alle der 144 Spalten, wo eine 1 im Namen steht gibt es immer 14, eine für jeden (möglichen) Stem.
Der DataFrame trägt auch eine Funktion um Spalten infos auszugeben, wir können hier später alles relevante ergänzen:

[18]:
```
df.columns_info()
```

```
Information on RNALoops DataFrame columns:
```

- loop_type (int8, NOT NULL): Any of 3-14 = number of structure stems

- home_structure (category, NOT NULL): Where the structure is found in PDB

- db_notation (str_arrow, NOT NULL): bracket notation, explanation see e.g. https://www.tbi.univie.ac.at/RNA/ViennaRNA/doc/html/rna_structure_notations.html#dot-bracketnotation

- whole_sequence (str_arrow, NOT NULL): The base sequence of the structure

- helix_{1…14}_bps (int8): Length of helices 1…14 in base pairs, -1 for non existent, at least 1, 2 and 3 must not be -1, to have a multiloop

- strand_{1…14}_nts (int16): Length of strands 1…14 in units, again -1 for non existent, can be 0 if two helices are next to each other

- start_{1…14} (str_arrow): The start position of strand 1…14

- end_{1…14} (str_arrow): The end position of strand 1…14

- euler_{x|y|z}_{1…14} (float32, 0<=…<=180): Euler angles in ° of strand 1…14 in x, y and z

- planar_{x|y|z}_{1…14} (float32, 0<=…<=180): Planar angle in °, for Details on angles read https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9438955/

Beim Präparieren wurden Spaltennamen vereinfacht und Dateitypen angepasst um Memory zu sparen (vgl. data_getter.getter_fcts.prepare_df).
Im Folgenden konzentrieren wir und vor allem auf die numerischen Spalten und ggbf. die kategorischen zur Gruppierung der Daten.
Wir können diese nochmal genauer angucken:

[99]:
```
df.describe()
```

[99]:

	helix_1_bps	helix_2_bps	helix_3_bps	helix_4_bps	helix_5_bps	\
count	75222.000000	75222.000000	75222.000000	75222.000000	75222.000000	
mean	2.986134	3.312768	2.595464	1.169844	0.067706	
std	2.388288	2.670049	2.020922	2.498688	1.952590	
min	1.000000	1.000000	-1.000000	-1.000000	-1.000000	
25%	1.000000	1.000000	1.000000	-1.000000	-1.000000	
50%	2.000000	3.000000	2.000000	1.000000	-1.000000	
75%	4.000000	5.000000	4.000000	2.000000	1.000000	
max	24.000000	16.000000	16.000000	13.000000	12.000000	

	helix_6_bps	helix_7_bps	helix_8_bps	helix_9_bps	helix_10_bps	\
count	75222.000000	75222.000000	75222.000000	75222.000000	75222.000000	

mean	-0.508335	-0.646221	-0.801255	-0.887294	-0.931456
std	1.282699	1.137198	0.731796	0.551546	0.427927
min	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000
25%	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000
50%	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000
75%	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000
max	9.000000	12.000000	8.000000	10.000000	7.000000

	helix_11_bps	helix_12_bps	helix_13_bps	helix_14_bps	strand_1_nts	\
count	75222.000000	75222.000000	75222.000000	75222.000000	75222.000000	
mean	-0.929967	-0.988886	-0.992555	-0.993592	2.633551	
std	0.532671	0.151597	0.121795	0.113024	2.728873	
min	-1.000000	-1.000000	-1.000000	-1.000000	0.000000	
25%	-1.000000	-1.000000	-1.000000	-1.000000	1.000000	
50%	-1.000000	-1.000000	-1.000000	-1.000000	2.000000	
75%	-1.000000	-1.000000	-1.000000	-1.000000	4.000000	
max	9.000000	8.000000	1.000000	1.000000	79.000000	

	strand_2_nts	strand_3_nts	strand_4_nts	strand_5_nts	strand_6_nts	\
count	75222.000000	75222.000000	75222.000000	75222.000000	75222.000000	
mean	3.254832	2.823868	0.847904	0.201351	-0.492476	
std	3.116770	2.837187	2.630209	2.243865	1.453155	
min	0.000000	-1.000000	-1.000000	-1.000000	-1.000000	
25%	1.000000	1.000000	-1.000000	-1.000000	-1.000000	
50%	3.000000	2.000000	0.000000	-1.000000	-1.000000	
75%	5.000000	4.000000	1.000000	1.000000	-1.000000	
max	85.000000	69.000000	54.000000	53.000000	32.000000	

	strand_7_nts	strand_8_nts	strand_9_nts	strand_10_nts	strand_11_nts	\
count	75222.000000	75222.000000	75222.000000	75222.000000	75222.000000	
mean	-0.556712	-0.679854	-0.852782	-0.924251	-0.888437	
std	1.370773	1.216909	0.765178	0.617566	0.757215	
min	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	
25%	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	
50%	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	
75%	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	
max	39.000000	32.000000	27.000000	27.000000	38.000000	

	strand_12_nts	strand_13_nts	strand_14_nts	euler_x_1	\
count	75222.000000	75222.000000	75222.000000	75222.000000	
mean	-0.975539	-0.988753	-0.987743	105.244179	
std	0.388358	0.229980	0.254150	50.970524	
min	-1.000000	-1.000000	-1.000000	0.000000	
25%	-1.000000	-1.000000	-1.000000	66.062248	
50%	-1.000000	-1.000000	-1.000000	112.903000	
75%	-1.000000	-1.000000	-1.000000	149.100006	
max	30.000000	19.000000	13.000000	180.000000	

	euler_x_2	euler_x_3	euler_x_4	euler_x_5	euler_x_6
count	75222.000000	75221.000000	44410.000000	24637.000000	13340.000000
mean	92.838730	101.747887	96.028313	86.446350	71.941940
std	56.794224	53.420174	54.803646	50.587391	60.333614
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	43.347000	59.657001	47.889252	45.252998	17.995501
50%	99.416000	111.129997	103.690002	83.045998	54.090500
75%	142.626255	149.011993	146.082748	128.472000	131.406250
max	180.000000	180.000000	180.000000	179.992004	179.998993

	euler_x_7	euler_x_8	euler_x_9	euler_x_10	euler_x_11
count	10001.000000	6599.000000	3756.000000	2251.000000	2041.000000
mean	80.086456	93.230118	98.241081	81.227737	90.753967
std	53.069675	51.064781	60.392536	48.091969	54.526382
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	30.070999	57.184998	43.006001	43.715500	41.652000
50%	82.858002	92.397003	96.679504	82.792000	103.805000
75%	129.684006	137.858505	156.884003	110.589500	134.628998
max	179.983994	179.977005	179.996994	179.932007	179.768997

	euler_x_12	euler_x_13	euler_x_14	euler_y_1	euler_y_2
count	414.000000	280.000000	241.000000	75222.000000	75222.000000
mean	33.941204	110.513512	105.032585	100.603241	92.283539
std	31.647486	53.463257	70.267899	51.772068	56.380585
min	0.000000	0.000000	1.460000	0.000000	0.000000
25%	13.200250	80.498749	20.365000	57.598499	42.722501
50%	29.409500	119.026505	155.587997	107.004498	100.574997
75%	37.943253	160.524750	171.906998	145.432751	141.941750
max	174.026001	179.934006	179.645996	180.000000	180.000000

	euler_y_3	euler_y_4	euler_y_5	euler_y_6	euler_y_7
count	75221.000000	44410.000000	24637.000000	13340.000000	10001.000000
mean	102.793251	98.956909	84.322853	64.243675	81.656464
std	52.734119	54.479156	53.785870	56.132008	52.366680
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	62.945000	55.144753	38.646999	13.950750	39.508999
50%	110.859001	104.606003	77.222000	44.862000	81.832001
75%	149.100006	149.095001	130.863007	111.930252	120.490997
max	180.000000	180.000000	179.990005	179.987000	179.938995

	euler_y_8	euler_y_9	euler_y_10	euler_y_11	euler_y_12
count	6599.000000	3756.000000	2251.000000	2041.000000	414.000000
mean	85.596199	99.912918	80.360725	98.812202	68.824989
std	53.493900	61.777218	60.984196	61.791641	54.731640
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	41.218002	35.500751	26.191500	41.847000	18.234749

```
50%      73.025002   101.778999    59.646999   104.209999   48.206501
75%     140.802994   164.387253   140.612000   157.686996  124.192245
max     179.985992   179.908997   179.979996   179.955994  175.595001

       euler_y_13  euler_y_14     euler_z_1     euler_z_2     euler_z_3  \
count  280.000000  241.000000  75222.000000  75222.000000  75221.000000
mean   110.439041  109.347687    101.395035     93.228867    101.522522
std     54.244652   56.799435     51.837280     55.901161     53.503819
min      0.000000    1.595000      0.000000      0.000000      0.000000
25%     53.192749   53.060001     60.522999     46.219002     59.124001
50%    128.246506  139.585999    108.668499    102.775501    110.391998
75%    154.583496  156.845001    146.009747    141.316002    148.813004
max    179.005997  179.147995    180.000000    180.000000    180.000000

          euler_z_4     euler_z_5     euler_z_6     euler_z_7    euler_z_8  \
count  44410.000000  24637.000000  13340.000000  10001.000000  6599.000000
mean      97.837502     85.630089     64.948853     77.563683    89.301277
std       55.242184     52.745186     54.542343     53.711147    53.725685
min        0.000000      0.000000      0.000000      0.000000     0.000000
25%       51.673500     39.799000     19.824001     31.322001    38.774002
50%      104.658997     77.668999     47.807999     71.017998    85.222000
75%      147.782501    133.013000    110.458748    118.332001   142.070007
max      180.000000    179.990005    179.998993    179.903000   179.931000

         euler_z_9   euler_z_10   euler_z_11  euler_z_12  euler_z_13  \
count  3756.000000  2251.000000  2041.000000  414.000000  280.000000
mean     96.606270    75.432381    99.772774   60.882175  125.201912
std      60.490429    55.858627    58.359795   53.534912   59.071404
min       0.000000     0.000000     0.000000    0.000000    0.000000
25%      36.576500    32.208500    43.415001   12.146999   59.672249
50%      89.949005    59.101002   114.225998   40.782501  162.074493
75%     157.863007   125.050995   155.509995   99.713005  175.391998
max     179.973999   179.970001   179.714996  179.979004  179.716995

       euler_z_14      planar_1      planar_2      planar_3      planar_4  \
count  241.000000  75222.000000  75222.000000  75221.000000  44410.000000
mean   103.518883    101.655197     92.517677    102.317772     97.761078
std     60.806770     38.678852     45.386295     41.058601     43.735291
min      1.842000      0.000000      0.000000      0.000000      0.000000
25%     38.924000     82.471001     66.144999     80.962997     60.970001
50%    154.539993    104.004997     97.430000    103.485001     96.047501
75%    158.306000    127.703753    130.203995    137.910004    139.544006
max    179.940994    180.000000    180.000000    180.000000    180.000000

          planar_5      planar_6      planar_7     planar_8     planar_9  \
count  24637.000000  13340.000000  10001.000000  6599.000000  3756.000000
mean      86.033241     67.771423     81.429741    88.548126    98.916649
```

std	41.203915	48.573021	39.314079	42.984428	52.885262
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	56.582001	32.653000	55.967999	61.057999	59.402500
50%	76.864998	54.722500	92.714996	66.806000	83.168503
75%	112.614998	101.876251	107.920998	134.742493	159.684006
max	177.608002	171.598007	176.057007	177.399994	177.134003

	planar_10	planar_11	planar_12	planar_13	planar_14
count	2251.000000	2041.000000	414.000000	280.000000	241.000000
mean	78.473328	96.354645	52.091805	114.483894	103.962410
std	43.649445	52.159992	37.641174	47.978886	58.348415
min	0.000000	0.000000	0.000000	0.000000	9.808000
25%	49.906998	47.804001	15.734750	93.161751	36.629002
50%	58.452000	83.931999	40.811501	94.669495	154.979004
75%	115.386505	153.384995	92.227501	161.612503	158.972000
max	174.225998	170.899994	167.341995	166.117004	159.787003

Wir haben nur noch 75222 Einträge. Laut Autoren sollen 84261 Strukturen vorliegen, es konnen aber nur knapp 80k heruntergeladen werden.
Der Rest existiert entweder nicht oder ist sehr gut versteckt ;) … Außerdem gibt es ca. 2300 Strukturen ohne Infos, die auch entfernt wurden.
Dazu sind ~3000 Strukturen doppelt vorhanden, die Duplikate wurden auch entfernt.

Spalten die keine Daten enthalten sollten, weil die Struktur weniger als 14 stems hat werden wie folgt behandelt: - Bei Winkeln wird NaN gesetzt (okay da Winkel ohnehin floats) - Bei Längen wird -1 gesetzt (NaN nicht sinnvoll bei int, 0 verwirrend weil Strands mit Länge 0 tatsächlich existieren) - Bei Positionen und Sequenzen wird NA gesetzt (es gibt auch '-', was so in der DB steht und heißt der Wert ist unbekannt?) - Beim Gruppieren nach loop_type können die Spalten einfach weggelassen werden

1.3 Ein paar Beispiele

Vor den ersten zusammenfassenden Statistiken auf den Daten erstmal ein paar Beispiele für Strukturen.
Wir können eine einzelne Struktur mittels show_structure() aus explore_fcts als pd.Series erhalten. Dabei haben wir die Optionen: - indices (int | Iterable): Liste/int = Gewünschte(r) Struktur idx (default: n zufällige Indices, wähle n beliebig) - web (bool): Zeigt Website der Struktur(en) aus DB in Browser (benötigt driver, siehe unten). Default False. - pdf (bool): Zeigt pdf der Struktur(en) in Browser pdf Viewer (lädt herunter falls nicht vorhanden). Default False. - svg (bool): Zeigt svg image der Struktur(en) in Notebook (lädt vorher herunter). Default False. - keep_files (bool): Ob die heruntergeladenen Dateien behalten werden sollen. Default: False.

Webanzeige der Struktur benötigt einen webdriver. Bsp. chromedriver von https://chromedriver.chromium.org/downloads (i.d.R. V106).
Der driver muss in $PATH zu finden sein, Funktion ohne Gewähr.

Wir können z.B. Strukturen vergleichen (hier zufällige):

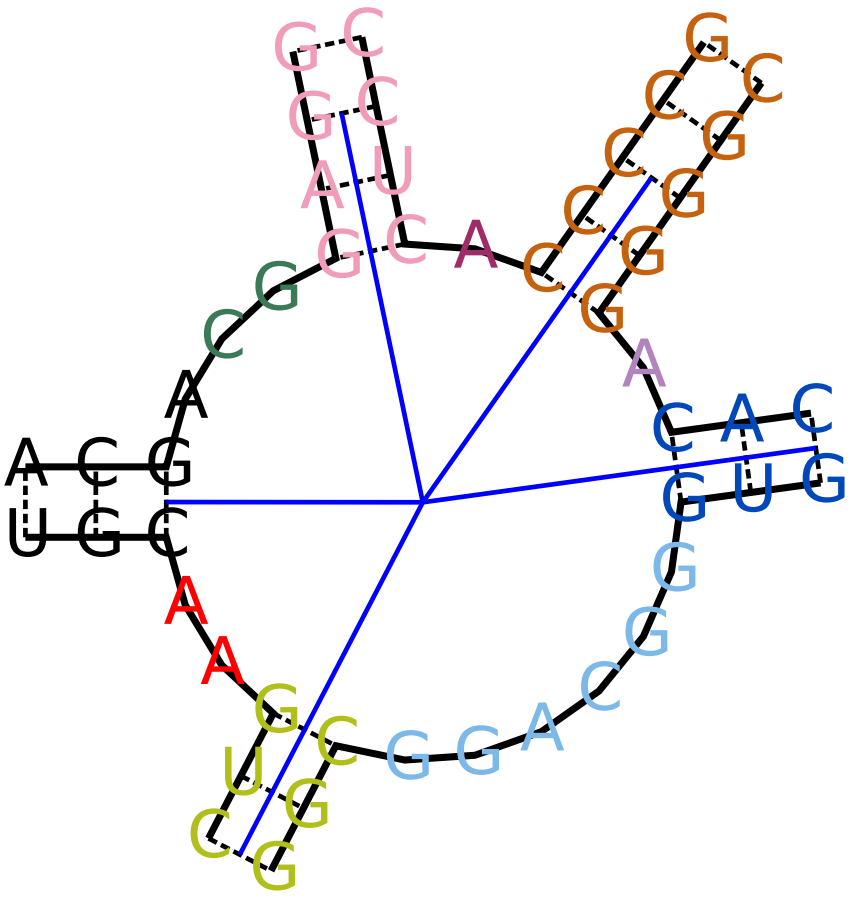
[100]:
```
s1 = show_structure(df, web=False, pdf=False, svg=False, keep_files=False, n=3)
s1.T.head(7)
```

[100]:

	175847	121536	97597
loop_type	4-way	9-way	4-way
home_structure	5on6	4v6d	4wzo
db_notation	(((((((…(-)..(-).(((-)))…)))))))	((-)…[- [(- (..(-)…(-)] -](-)…	(((…((((((((-)))))))))(((-))).((((- …
whole_sequence	UCCUUGUCUAUG - CGUC - GAGGG - CCCGUGUGGCGAGGA	UC - GGUGAG - UC - ACCC - GGAAA - UA - CU - AA…	GUGAGUAACGCGUGGGU - GCCCGCGUCCC - GGGUAGCCG - …
helix_1_bps	8	1	3
helix_2_bps	1	1	8
helix_3_bps	1	1	3

Oder Strukturen visualisieren:

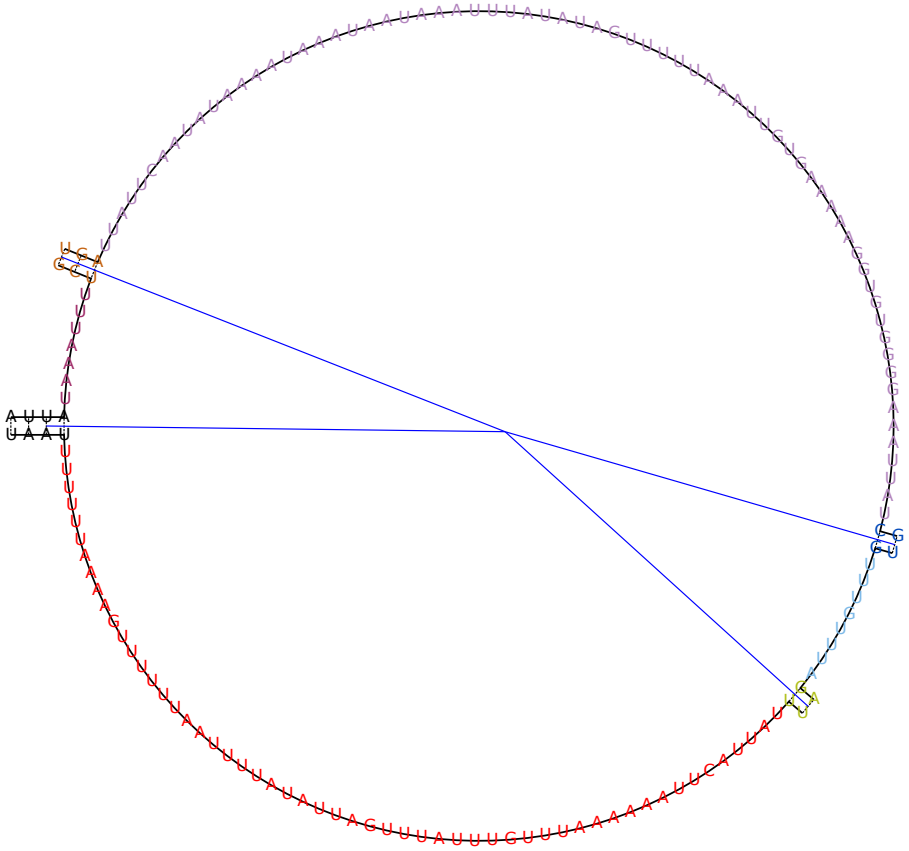
[57]:
```
_ = show_structure(df, svg=True)  # Braucht Internet, kann etwas dauern
```



```
File removed to keep cwd clean, keep_svg=True to keep
Structure svg file:
6nd5_5WJ_2a-56_2a-63_2a-104_2a-115_2a-312_2a-320_2a-333_2a-342_2a-347_2a-356.svg
```

Ein paar exotische Beispiele:

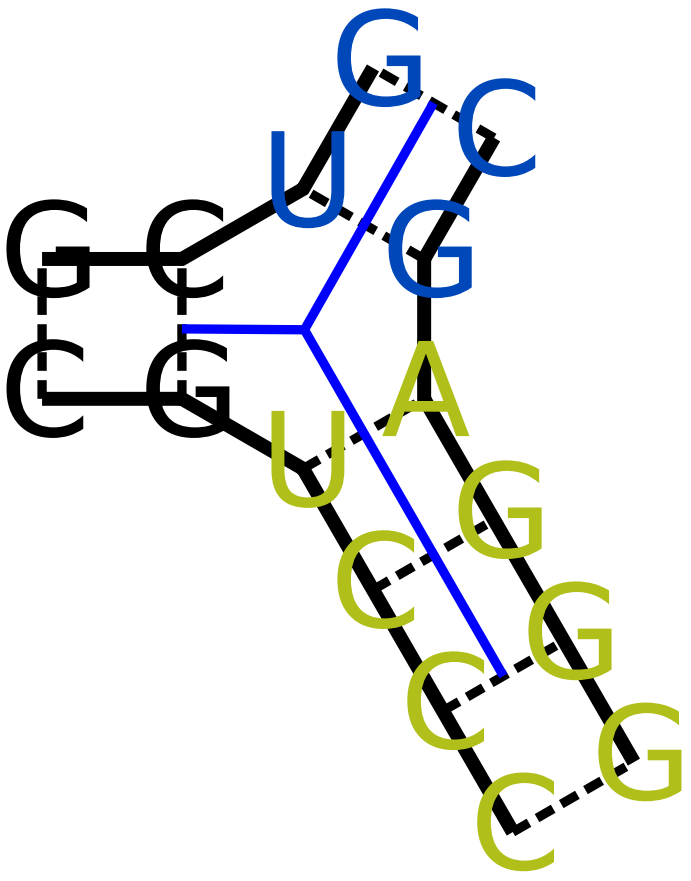
[101]:
```
large = df[df.strand_1_nts > 50]
_ = show_structure(large, svg=True)
```



```
File removed to keep cwd clean, keep_svg=True to keep
Structure svg file: 7ane_4WJ_2-22_2-93_2-127_2-138_2-143_2-216_2-252_2-265.svg
```

[119]:
```
small = df[(df.loop_type == "3-way") # Structure without any stems
         & (df.strand_1_nts + df.strand_2_nts + df.strand_3_nts == 0)]
```

[120]:
```
_ = show_structure(small, svg=True)
```




```
File removed to keep cwd clean, keep_svg=True to keep
Structure svg file: 6g90_3WJ_1-172_1-184_1-305_1-310_2-35_2-38.svg
```

1.4 Übersicht der Daten

Nächster Schritt wäre jetzt Gruppen von Strukturen zu betrachten und Verteilungen der Feature explorativ zu untersuchen.
Wechsel dazu der Übersichtlichkeit halber in eda_2.ipynb...