

REINVENT 4: Modern Generative AI for Molecular Design (Supplement)

Hannes H. Loeffler^{1*}, Jiazhen He¹, Alessandro Tibo¹,
Jon Paul Janet¹, Alexey Voronov¹, Lewis Mervin², Ola Engkvist¹

^{1*}Molecular AI, Discovery Sciences, R&D, AstraZeneca, Gothenburg,
Sweden.

²Molecular AI, Discovery Sciences, R&D, AstraZeneca, Cambridge, UK.

*Corresponding author(s). E-mail(s): hannes.loffler@astrazeneca.com;

1 Effect of Inception and Scaffold Diversity Filter

Figure S1 summarizes how inception and scaffold diversity filter (DF) affect Reinforcement Learning (RL). All results are taken as the average over 5 independent runs. The highlighted areas denote the standard deviation for each data point.

The data (orange and blue lines) in S1a show how inception leads to finding significantly more compounds with a smaller binding free energy. When inception is switched on, DF displays higher ΔG values. With DF on RL is forced to create more diverse structures and does not allow the run to exploit a smaller number of scaffolds.

Subfigure S1b displays the fraction of valid SMILES found in each RL step. When DF is on the number of valid SMILES drops relative to when DF is off.

In S1c we see the effect of inception on the number of unique scaffolds found (DF is on). Inception leads to a larger diversity in sampled structures.

Subfigure S1d shows how inception affects how many unique scaffolds are found more than N times. N is the size of the DF memory and was taken as 10 in this case. Clearly, with inception RL samples larger numbers of scaffolds in line with S1c.

The scoring function used for this examples is composed of QED, custom alerts, number of stereocentres (set to zero) and binding free energy. The latter is built with a QSAR-like response model derived from about 10 000 MM-PBSA calculations of ligands bound to the main protease of SARS-2. The data here only serves as a demonstration. Detailed results will be published elsewhere. The batch size was set to 100, the RL learning strategy was DAP with $\sigma = 128$ and a learning rate of 0.0001, DF was of type Murcko scaffold with a memory size of 10 and minimum score threshold

for storage of 0.4 and the inception memory size was set to 50 with a sample size of 10. RL was executed for 300 steps.

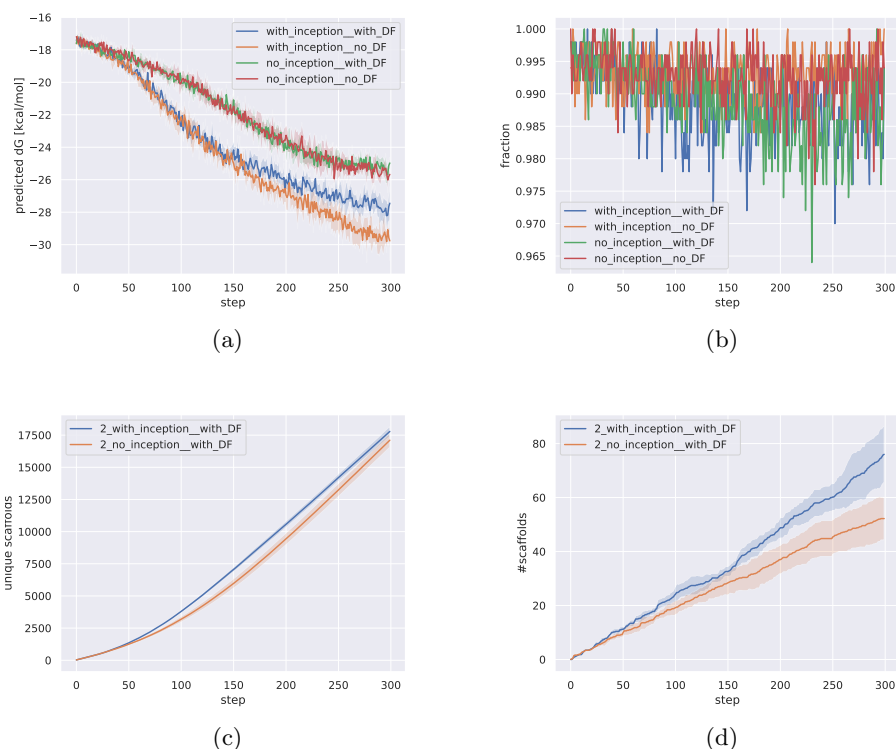


Fig. S1: Effect of inception and scaffold diversity filter. a) Average binding free energy, b) Fraction of valid SMILES found per batch (typically above 95%), c) Number of unique scaffolds, d) Number of scaffolds found more than 10 times.

2 Input configuration examples

Listing S1: Staged learning input example.

```

1 # REINVENT4 TOML input example for reinforcement/curriculum learning
2 #
3 #
4 # Curriculum learning in REINVENT4 is a multi-stage reinforcement
   learning
5 # run. One or more stages (auto CL) can be defined. But it is also

```

```

6 # possible to continue a run from any checkpoint file that is
   generated
7 # during the run (manual CL). Currently checkpoints are written at
   the end
8 # of a run also when the run is forcefully terminated with Ctrl-C.
9
10
11 run_type = "staged_learning"
12 use_cuda = true # run on the GPU if true, on the CPU if false
13 tb_logdir = "tb_logs" # name of the TensorBoard logging directory
14 json_out_config = "_staged_learning.json" # write this TOML to JSON
15
16 [parameters]
17
18 # Uncomment one of the comment blocks below. Each generator needs a
   model
19 # file and possibly a SMILES file with seed structures. If the run is
   to
20 # be continued after termination, the agent_file would have to be
   replaced
21 # with the checkpoint file.
22
23 use_checkpoint = false # if true read diversity filter from agent_file
24 summary_csv_prefix = "staged_learning" # prefix for the CSV file
25
26 ## Reinvent
27 prior_file = "models/reinvent.prior"
28 agent_file = "models/reinvent.prior"
29
30 ## LibInvent
31 #prior_file = "models/libinvent.prior"
32 #agent_file = "models/libinvent.prior"
33 #smiles_file = "scaffolds.smi" # 1 scaffold per line with attachment
   points
34
35 ## LinkInvent
36 #prior_file = "models/linkinvent.prior"
37 #agent_file = "models/linkinvent.prior"
38 #smiles_file = "warheads.smi" # 2 warheads per line separated with '|'
39
40 ## Mol2Mol
41 #prior_file = "models/mol2mol_scaffold_generic.prior"
42 #agent_file = "models/mol2mol_scaffold_generic.prior"
43 #smiles_file = "mol2mol.smi" # 1 compound per line

```

```

44 #sample_strategy = "multinomial" # multinomial or beamsearch
    (deterministic)
45 #distance_threshold = 100
46
47 batch_size = 64          # network
48
49 unique_sequences = true # if true remove all duplicates raw sequences
    in each step
50                                # only here for backward compatibility
51 randomize_smiles = true # if true shuffle atoms in SMILES randomly
52
53
54 [learning_strategy]
55
56 type = "dap"             # dap: only one supported
57 sigma = 128             # sigma of the RL reward function
58 rate = 0.0001           # for torch.optim
59
60
61 [diversity_filter] # optional, comment section out or remove if
    unneeded
62                                # NOTE: also memorizes all seen SMILES
63
64 type = "IdenticalMurckoScaffold" # IdenticalTopologicalScaffold,
65                                # ScaffoldSimilarity, PenalizeSameSmiles
66 bucket_size = 25         # memory size in number of compounds
67 minscore = 0.4           # only memorize if this threshold is
    exceeded
68 minsimilarity = 0.4      # minimum similarity for
    ScaffoldSimilarity
69 penalty_multiplier = 0.5 # penalty factor for PenalizeSameSmiles
70
71
72 # Reinvent only: guide RL in the initial phase
73 #[inception] # optional, comment sectionout or remove if unneeded
74
75 #smiles_file = "sampled.smi" # "good" SMILES for guidance
76 #memory_size = 100 # number of total SMILES held in memory
77 #sample_size = 10 # number of SMILES randomly chosen each epoch
78
79
80 ### Stage 1
81 ### Note that stages must always be a list i.e. double brackets
82 [[stage]]
83

```

```

84 | chkpt_file = 'test1.chkpt' # name of the checkpoint file, can be
    | reused as agent
85 |
86 | termination = "simple" # termination criterion for this stage
87 | max_score = 0.6 # terminate if this total score is exceeded
88 | min_steps = 25 # run for at least this number of steps
89 | max_steps = 100 # terminate entire run when exceeded
90 |
91 | [stage.scoring]
92 | type = "geometric_mean" # aggregation function
93 |
94 | [[stage.scoring.component]]
95 | # Custom alerts if used in a custom_product filter out unwanted groups
96 | [stage.scoring.component.custom_alerts]
97 |
98 | [[stage.scoring.component.custom_alerts.endpoint]]
99 | name = "Unwanted SMARTS" # user chosen name for output
100 | weight = 0.79 # weight to fine-tune the relevance of this component
101 |
102 | # parameters for the component:
103 | # a list of unwanted SMARTS(!) to be scored as zero
104 | params.smarts = [
105 |     " [*;r8] ",
106 |     " [*;r9] ",
107 |     " [*;r10] ",
108 |     " [*;r11] ",
109 |     " [*;r12] ",
110 |     " [*;r13] ",
111 |     " [*;r14] ",
112 |     " [*;r15] ",
113 |     " [*;r16] ",
114 |     " [*;r17] ",
115 |     " [#8] [#8] ",
116 |     " [#6;+] ",
117 |     " [#16] [#16] ",
118 |     " [#7;!n] [S;!$(S(=O)=O)] ",
119 |     " [#7;!n] [#7;!n] ",
120 |     " C#C ",
121 |     " C(=[O,S])[O,S] ",
122 |     " [#7;!n] [C;!$(C(=[O,N])[N,O])] [#16;!s] ",
123 |     " [#7;!n] [C;!$(C(=[O,N])[N,O])] [#7;!n] ",
124 |     " [#7;!n] [C;!$(C(=[O,N])[N,O])] [#8;!o] ",
125 |     " [#8;!o] [C;!$(C(=[O,N])[N,O])] [#16;!s] ",
126 |     " [#8;!o] [C;!$(C(=[O,N])[N,O])] [#8;!o] ",
127 |     " [#16;!s] [C;!$(C(=[O,N])[N,O])] [#16;!s] "

```

```

128 ]
129
130 [[stage.scoring.component]]
131 [stage.scoring.component.MolecularWeight]
132
133 [[stage.scoring.component.MolecularWeight.endpoint]]
134 name = "Molecular weight" # user chosen name for output
135 weight = 0.342 # weight to fine-tune the relevance of this component
136
137 # A transform ensures that the output from the scoring component
138 # ranges
139 # from 0 to 1 to serve as a proper score. Here we use a double sigmoid
140 # to transform weights into the range 200-500 a.u.
141 transform.type = "double_sigmoid"
142 transform.high = 500.0
143 transform.low = 200.0
144 transform.coef_div = 500.0
145 transform.coef_si = 20.0
146 transform.coef_se = 20.0
147
148 ### Stage 2
149 # Alternatively only the first stage above can be run and the new
150 # input file
151 # sets agent_file = 'test1.chkpt'.
152
153 [[stage]]
154
155 chkpt_file = 'test2.chkpt'
156
157 termination = "simple"
158 max_score = 0.7
159 min_steps = 10
160 max_steps = 100
161
162 [stage.scoring] # the scoring components can be read from a score file
163 type = "geometric_mean" # aggregation function
164 filename = "stage2_scoring.toml" # file with scoring setup for this
165 # stage
166 filetype = "toml" # file format: TOML or JSON, no default, must be
167 # present
168
169 ### Stage 3
170 # just as above

```

Listing S2: Scoring components read from a separate file.

```

1  ### Stage 2 scoring components
2  [[component]]
3  [component.QED]
4
5  [[component.QED.endpoint]]
6  weight = 0.5 # user chosen name for output
7  name = "QED Score" # weight to fine-tune the relevance of this
    component

```

Listing S3: Example scaffold file for REINVENT 4 LibInvent.

```

1  # One scaffold per line
2  # Each scaffold must be annotated by 1 to 4 '*'s to locate the
3  # attachment points followed with the label number after ':'
4
5  [*:1]Cc2ccc1cncc(C[*:2])c1c2
6  [*:1]Cc2cnc1cncc(C[*:2])c1c2

```

Listing S4: Example warhead file for REINVENT 4 LinkInvent.

```

1  # One warhead pair per line
2  # Each warhead must be annotated with '*' to locate the attachment
   points
3  # The two warheads must be separated by the pipe symbol '|'
4
5  Oc1cncc(*)c1|*c1ccoc1
6  Nc1c([F])ccc(*)c1|*c1c([Cl])cnc1

```

Listing S5: Sampling input example.

```

1  # REINVENT4 TOML input example for sampling
2  #
3
4
5  run_type = "sampling"
6  use_cuda = true # run on the GPU if true, on the CPU if false
7  json_out_config = "_sampling.json" # write this TOML to JSON
8
9
10 [parameters]
11
12 # Uncomment one of the comment blocks below. Each generator needs a
    model

```

```

13 # file and possibly a SMILES file with seed structures.
14
15 ## Reinvent: de novo sampling
16 model_file = "models/reinvent.prior"
17
18 ## LibInvent: find R-groups for the given scaffolds
19 #model_file = "models/libinvent.prior"
20 #smiles_file = "scaffolds.smi" # 1 scaffold per line with attachment
    points
21
22 ## LinkInvent: find a linker/scaffold to link two fragments
23 #model_file = "models/linkinvent.prior"
24 #smiles_file = "warheads.smi" # 2 warheads per line separated with '|'
25
26 ## Mol2Mol: find molecules similar to the provided molecules
27 #model_file = "models/mol2mol_medium_similarity.prior"
28 #smiles_file = "mol2mol.smi" # 1 compound per line
29 #sample_strategy = "beamsearch" # multinomial or beamsearch
    (deterministic)
30 #temperature = 1.0 # temperature in multinomial sampling
31 #tb_logdir = "tb_logs" # name of the TensorBoard logging directory
32
33 output_file = 'sampling.csv' # sampled SMILES and NLL in CSV format
34
35 num_smiles = 157 # number of SMILES to be sampled, 1 per input SMILES
36 unique_molecules = true # if true remove all duplicates and canonicalize
    smiles
37 randomize_smiles = true # if true shuffle atoms in SMILES randomly

```

Listing S6: Scoring input example.

```

1 # REINVENT4 TOML input example for scoring
2 #
3
4 run_type = "scoring"
5 output_csv = "scoring.csv" # run on the GPU if true, on the CPU if
    false
6 json_out_config = "_scoring.json" # write this TOML to JSON
7
8 [parameters]
9
10 smiles_file = "compounds.smi" # SMILES file with 1 molecule per line
11
12
13 [scoring]

```



```

14
15 type = "geometric_mean" # or arithmetic_mean
16 parallel = false # do not run scoring components in parallel
17
18 [[scoring.component]]
19 # Custom alerts if a filter of unwanted groups
20 [scoring.component.custom_alerts]
21
22 [[scoring.component.custom_alerts.endpoint]]
23 name = "Alerts" # user chosen name for output
24 # no weight because as a filter it will be applied globally
25
26 params.smarts = [
27     " [*;r8] ",
28     " [*;r9] ",
29     " [*;r10] ",
30     " [*;r11] ",
31     " [*;r12] ",
32     " [*;r13] ",
33     " [*;r14] ",
34     " [*;r15] ",
35     " [*;r16] ",
36     " [*;r17] ",
37     " [#8] [#8] ",
38     " [#6;+] ",
39     " [#16] [#16] ",
40     " [#7;!n] [S;!$(S(=0)=0)] ",
41     " [#7;!n] [#7;!n] ",
42     " C#C ",
43     " C(=[0,S])[0,S] ",
44     " [#7;!n] [C;!$(C(=[0,N])[N,0])] [#16;!s] ",
45     " [#7;!n] [C;!$(C(=[0,N])[N,0])] [#7;!n] ",
46     " [#7;!n] [C;!$(C(=[0,N])[N,0])] [#8;!o] ",
47     " [#8;!o] [C;!$(C(=[0,N])[N,0])] [#16;!s] ",
48     " [#8;!o] [C;!$(C(=[0,N])[N,0])] [#8;!o] ",
49     " [#16;!s] [C;!$(C(=[0,N])[N,0])] [#16;!s] "
50 ]
51
52 [[scoring.component]]
53 [scoring.component.QED]
54
55 [[scoring.component.QED.endpoint]]
56 name = "QED"
57 weight = 0.25 # weight to fine-tune the importance of this component
58

```

```

59 [[scoring.component]]
60 [scoring.component.MolecularWeight]
61
62 [[scoring.component.MolecularWeight.endpoint]]
63 name = "MW"
64 weight = 0.25
65
66 # transforms can be used for any component if desired
67 transform.type = "double_sigmoid"
68 transform.high = 500.0
69 transform.low = 200.0
70 transform.coef_div = 500.0
71 transform.coef_si = 20.0
72 transform.coef_se = 20.0
73
74 [[scoring.component]]
75 [scoring.component.TanimotoDistance]
76
77 [[scoring.component.TanimotoDistance.endpoint]]
78 name = "Tanimoto similarity ECF6"
79 weight = 0.1
80
81 # component specific parameters
82 params.smiles = [
83     "n1(nc(c(c1C)-c2n[nH]c(c2)[C@@]3([C@@H](CN(CC3)Cc4nc5c(c(n4)C)cccc5)O)OC)C)C"
84 ]
85 params.radius = 3
86 params.use_counts = true
87 params.use_features = true
88
89
90 # Multiple end-points for many scoring components: each is a separate
    component
91 # with its own weight
92 [[scoring.component]]
93 [scoring.component.pmi]
94 name = "PMI 3D-likeness"
95
96 [[scoring.component.pmi.endpoint]]
97 weight = 0.79
98 params.property = 'npr1' # component specific parameters
99
100 [[scoring.component.pmi.endpoint]]
101 weight = 0.21
102 params.property = 'npr2' # component specific parameters

```

Listing S7: Transfer Learning input example.

```

1 # REINVENT4 TOML input example for transfer learning
2 #
3 # Focus a given model towards a set of input SMILES. This can also be
  used to
4 # start from scratch from and an untrained prior.
5
6
7 run_type = "transfer_learning"
8 use_cuda = true # run on the GPU if true, on the CPU if false
9 tb_logdir = "tb_TL" # name of the TensorBoard logging directory
10 json_out_config = "json_transfer_learning.json" # write this TOML to
   JSON
11
12
13 [parameters]
14
15 num_epochs = 3 # number of steps to run
16 save_every_n_epochs = 3 # save checkpoint model file every N steps
17 batch_size = 50
18 num_refs = 100 # number of reference molecules randomly chosen for
   similarity
19 sample_batch_size = 100 # number of sampled molecules chosen for
   similarity
20 # Uncomment one of the comment blocks below. Each generator needs a
   model
21 # file and possibly a SMILES file with seed structures.
22
23 ## Reinvent
24 #input_model_file = "models/reinvent.prior"
25 #smiles_file = "TL_reinvent_100.smi" # read 1st column
26 #output_model_file = "TL_reinvent.model"
27 #validation_smiles_file = "TL_reinvent_100.smi"
28
29 ## Mol2Mol
30 input_model_file = "models/mol2mol_scaffold_generic.prior"
31 smiles_file = "compounds.smi" # read 1st column
32 output_model_file = "TL_Mol2Mol.model"
33 validation_smiles_file = "validation_compounds.smi" # read 1st column
34
35 # Define the type of similarity and its parameters
36 pairs.type = "tanimoto"
37 pairs.upper_threshold = 1.0
38 pairs.lower_threshold = 0.7
39 pairs.min_cardinality = 1

```

```

40 pairs.max_cardinality = 199
41
42 ### It may not make much sense to use TL for Lib/Linkinvent because
43 ### both methods use constraints which may leave only a small portion
   of
44 ### the molecule to be optimizable
45 ## LibInvent
46 #input_model_file = "models/libinvent.prior"
47 #smiles_file = "TL_libinvent_100.smi" # read first 2 columns:
   warheads, linker
48 #validation_smiles_file = "TL_libinvent_100.smi" # read first 2
   columns: warheads, linker
49 #output_model_file = "TL_libinvent.model"
50 #
51 ## LinkInvent
52 #input_model_file = "models/linkinvent.prior"
53 #smiles_file = "TL_linkinvent_100.smi" # read first 2 columns:
   scaffold, R-groups
54 #output_model_file = "TL_linkinvent.model"
55 #validation_smiles_file = "TL_linkinvent_100.smi" # read first 2
   columns: scaffold, R-groups

```

Listing S8: Structure based drug design example, RL

```

1 version = 4
2 run_type = "staged_learning"
3 use_cuda = true
4 output_csv = "rl_direct.csv"
5 tb_logdir = "tb_rl_direct"
6
7 [parameters]
8 use_checkpoint = false
9 summary_csv_prefix = "rl_direct"
10 agent_file = "/models/reinvent.prior"
11 prior_file = "/models/reinvent.prior"
12
13 batch_size = 128
14 uniquify_smiles = true
15 randomize_smiles = true
16
17 [learning_strategy]
18
19 type = "dap"
20 sigma = 128
21 rate = 0.0005

```

```

22
23 [diversity_filter]
24
25 type = "IdenticalMurckoScaffold"
26 bucket_size = 50
27 minscore = 0.4
28
29 [[stage]]
30 chkpt_file = 'rl_direct.chkpt'
31 termination = "simple"
32 max_steps = 50
33
34 [stage.scoring]
35 type = "geometric_mean"
36
37 [[stage.scoring.component]]
38 [[stage.scoring.component.DockStream.endpoint]]
39 name = "Docking into 2XCH grid"
40 weight = 1
41 params.configuration_path = "dockstream_rl_direct_uncs.json"
42 params.docker_script_path = "[PATH_TO_DOCKSTREAM]/docker.py"
43 params.docker_python_path = "[PATH_TO_DOCKSTREAM_ENV]/python"
44 transform.type = "reverse_sigmoid"
45 transform.high = -7
46 transform.low = -13.5
47 transform.k = 0.2
48
49 [[stage.scoring.component]]
50 [stage.scoring.component.QED]
51 [[stage.scoring.component.QED.endpoint]]
52 name = "QED"
53 weight = 1

```

Listing S9: Structure based drug design example, TL

```

1 # Transfer learning on a set of related active compounds
2
3 run_type = "transfer_learning"
4 use_cuda = false
5 tb_logdir = "tb_TL"
6
7 [parameters]
8
9 num_epochs = 10
10 save_every_n_epochs = 5

```

```

11 batch_size = 50
12 num_refs = 50
13 sample_batch_size = 50
14
15 input_model_file = "models/reinvent.prior"
16 smiles_file = "AID_1798002_actives.smi"
17 output_model_file = "AID_1798002_actives.model"
18 validation_smiles_file = "AID_1798002_actives.smi"

```

3 Supported Chemistry

The listing [S10](#) summarizes the supported chemistry expressed as tokens for each generator. Basically, all priors support the same elements. The main differences are the ring sizes and that Mol2Mol accepts and generates chiral centers at C and (quaternary) N. Mol2Mol also supports double bond isomers.

Listing S10: Supported tokens in the prior models. Slight variations between models are owing to the specific data set preparation. Special tokens for sequence start, end and padding have been left out. * and | denote attachment points and fragment separator for Libinvent and Linkinvent and are also used in input (see Listings [S3](#) and [S4](#)).

```

1 Reinvent
2
3 #, =, -, (, ), 1, 2, 3, 4, 5, 6, 7, 8, 9, %10, Br, C, Cl, F, N, O, S,
4   [N+], [N-], [O-], [S+], [n+], [nH], c, n, o, s
5
6 Libinvent
7 decorator:
8 #, =, -, (, ), 1, 2, 3, 4, 5, 6, Br, C, Cl, F, N, O, S, [N+], [N-],
9   [N], [O-], [O], [S+], [n+], [nH], [s+], c, n, o, s, *, |
10 scaffold:
11 #, =, -, (, ), 1, 2, 3, 4, 5, 6, 7, 8, 9, Br, C, Cl, F, N, O, S, [*],
12   [N+], [N-], [N], [O-], [O], [S+], [n+], [nH], [s+], c, n, o, s
13
14 Linkinvent
15 warheads:
16 #, =, -, (, ), 1, 2, 3, 4, 5, 6, Br, C, Cl, F, N, O, S, [N+], [O-],
17   [O], [S+], [n+], [nH], [s+], c, n, o, s, *, |
18 linker:
19 #, =, -, (, ), 1, 2, 3, 4, 5, 6, 7, Br, C, Cl, F, N, O, S, [*], [N+],
20   [N-], [O-], [S+], [n+], [nH], [s+], c, n, o, s
21
22 Mol2mol (high, medium, low similarities)

```

```

18 #, =, -, /, \, (, ), 1, 2, 3, 4, 5, 6, 7, 8, Br, C, Cl, F, I, N, O,
    S, [C@@H], [C@@], [C@H], [C@], [N+], [N@+], [N@@+], [N@], [O-],
    [O], [S@@], [S@], [n+], [nH], c, n, o, s
19
20 Mol2mol mmp:
21 #, =, -, /, \, (, ), 1, 2, 3, 4, 5, 6, 7, 8, Br, C, Cl, F, I, N, O,
    S, [C@@H], [C@@], [C@H], [C@], [N+], [N@+], [N@@+], [O-], [O],
    [S@@], [S@], [n+], [nH], c, n, o, s
22
23 Mol2mol scaffold:
24 #, =, -, /, \, (, ), 1, 2, 3, 4, 5, 6, 7, 8, Br, C, Cl, F, I, N, O,
    S, [C@@H], [C@@], [C@H], [C@], [N+], [N@+], [N@@+], [O-], [O],
    [S@@], [S@], [n+], [nH], c, n, o, s
25
26 Mol2mol scaffold_generic:
27 #, =, -, /, \, (, ), 1, 2, 3, 4, 5, 6, 7, 8, Br, C, Cl, F, I, N, O,
    S, [C@@H], [C@@], [C@H], [C@], [N+], [N@+], [N@@+], [O-], [O],
    [S@@], [S@], [n+], [n-], [nH], c, n, o, s

```