Boltz-2 Screning Tutorial

High-throughput structure prediction and evaluation with `boltz-screen.sh`

Quick Start

1. **Prepare input files**
   - **`bait.txt`** – every entity present in every complex
   - **`screen.txt`** – one target (UniProt ID / FASTA / CCD) per line
2. **Run the screen**

```
boltz-screen.sh --bait bait.txt --screen screen.txt -n my_screen
```

`boltz-screen.sh` automates the entire workflow of predicting many *targets* against one or more *bait* entities. The script:

1. **Generates YAML input files** for every bait–target pair, supporting a number of user-supplied infput formats.
2. **Creates a Slurm job array** (or runs locally) and chooses GPU/host-RAM requirements automatically.
3. **Copies every prediction's output** into a clean `results/<JOB_NAME>/…` folder and keeps all Slurm logs in `slurm/`.
4. **Launches `boltz-analysis.sh`** afterwards to collate metrics, make plots and dashboards, and write ready-made ChimeraX sessions.

Script path

The boltz-screen.sh should be in your path variable (meaning, you can launch it from anywhere by typing `boltz-screen.sh`). If this does not work, execute this **once**:

```
/groups/plaschka/shared/software/scripts/update_bashrc.sh
```

Restart your terminal and try again.

Overview

The screen allows you to predict a single bait complex against many targets. The bait complex may contain multiple components such as proteins, nucleic acids, ligands, and ions. You need to provide minimally two files:

- **Bait file** – lists all entities that will appear in every complex.
- **Screen file** – lists all targets to be predicted against the bait.
- **Optional**: **Chain mapping file** – assigns human-readable names to the chains in the final prediction.

Before you start: Memory considerations on our hardware

The script tries to allocate appropriate GPU and host RAM for each job based on the input composition. The following table summarizes the typical VRAM requirements for different input compositions. If your job exceeds the VRAM limit, boltz will abort the run, but the SLURM job will still be shown as COMPLETED. The downstream analysis script will later identify failed runs.

Input composition (single prediction)	Typical VRAM on GPU card A100-40 GB¹	When it all fits
Protein-only		
≤ 1,000 residues, default `sampling_steps=100`, `recycles=3`	28-35 GB	A100-40 GB OK
Protein + RNA (two polymer chains, total ~1,200 nt/aa)	34-40 GB	A100-40 GB borderline → often OOM; A100-80 GB or H100 advised
Protein/RNA + ≤ 4 ligands/ions	+0.4-0.6 GB per CCD	still fits if base system < 34 GB
> >1.5 k total residues	40-70 GB	Needs A100-80 GB (g4) or H100-80 GB, which aren't available on our cluster

##Input files and formatting rules

The wrapper accepts a variety of input formats, which are converted to Boltz-2 YAML files internally, such as:

- Uniprot IDs
- Path to a FASTA file (protein or nucleic acid)
- CCD codes (ligands or ions, check here https://www.ebi.ac.uk/pdbe-srv/pdbechem/)
- PTMs (see next section)

Inline PTM syntax

Add one or more *position:CCD* tokens **after** the sequence token:

- `Q13838 38:SEP` → Ser-38 is phosphorylated (SEP) in chain *Q13838*
- `P05067 15:CSO 42:MSE` → two modifications in the same chain

The most common PTM codes are:

PTM (type of modified residue)	CCD code
Phospho-Serine	**SEP**
Phospho-Threonine	**TPO**
Phospho-Tyrosine	**PTR**
Seleno-Methionine	**MSE**

4.1 Bait file (`--bait <FILE>`)

- **Exactly one file** listing every entity that must appear in **every** complex.
- Accepted line types (one per line, blanks & lines starting with `#` are ignored):

Example	Meaning
`Q13838`	UniProt protein
`Q13838 38:SEP`	Protein with an inline PTM (see below)
`data/myRNA.fa`	FASTA file (protein or nucleic acid)
`ATP`	Ligand/ion (3-letter CCD or ion code, case-insensitive)

Example	Meaning
`SMILES NAME`	Custom ligand from *.smi* file

Ordering matters – Boltz numbers chains *strictly* according to polymer type. You should therefore also follow the convention and order the entities in your bait file like this:

> 1. Proteins
> 2. RNAs
> 3. DNAs
> 4. Ligands
> 5. Small ions

4.2 Screen file (`--screen <FILE>`)

Plain text: **one target per line**. You can use

- UniProt ID
- FASTA path
- CCD code (ligand or ion)

PTMS maybe included in the same way as in the bait file, e.g. `Q13838 38:SEP` .

4.3 Chain-mapping file (optional, but highly recommended)

Chain-map file

To give **human-readable labels** in the analysis stage, create a **plain-text chain-map file** that assigns names to your prediction targets. The chain mapping file lists which chain corresponds to which entity in the final prediction

> **Heads-up**
> The final chain ID of the *screen* entity depends on what's already in your **bait**:
> if your bait contains both protein/NA chains **and** ligands, the screened chain is inserted **between** the bait protein/NA chains and any ligand chains.

Example chain-map file

Example 1 – Two bait proteins (second is pSer-38) vs two RNAs

Input files

`bait_proteins.txt`

```
Q13838          # UniProt of bait protein
./RNA.fa        # Bait RNA sequence
ATP             # Ligand (CCD code)
MG              # Mg²⁺ ion
```

`screen_rnas.txt`

```
Q09161          # Screen target 1
Q13838 38:SEP   # Screen target 2 (Ser-38 phosphorylated)
./RNA2.fa       # Screen RNA sequence
```

`chain_mapping.txt`

```
0=UAP56         # bait protein
1=baitRNA       # bait RNA
2=target        # screened protein/RNA (inserted before ligands)
3=ATP           # ligand
4=Mg2+          # ion
```

Submission command

```
boltz-screen.sh --bait bait.txt \
                     --screen screen.txt \
                     --chain_mapping chain_mapping.txt \
                     -n MyScreenName
```

Example 2 – Screening a protein + RNA + ATP + Mg^{2+} bait complex against proteins/RNAs

Input files

`bait_proteins.txt`

```
Q09161
Q13838 38:SEP
```

screen_rnas.txt

```
/data/RNA1.fa
/data/RNA2.fa
```

chain_mapping.txt

```
0=UAP56         # first bait protein
1=baitRNA       # second bait protein (pSer-38)
2=target        # screened RNA (still before ligands, if any)
```

Submission command

```
boltz-screen.sh --bait bait_proteins.txt \
                    --screen screen_rnas.txt \
                    -n prot_vs_rna
```

Example 3 – Protein + RNA bait vs different ligands

Input files

bait_combo.list

```
P69905
bait_rna.fa
MG
```

screen_proteins.txt

```
ATP
ADP
AMP
```

chain_mapping.txt

```
0=UAP56         # bait protein
1=baitRNA       # bait RNA
2=targetLigand  # screened ligand (inserted after proteins/RNAs)
3=MG            # ion from bait
```

Submission command

```
boltz-screen.sh --bait bait_combo.list \
                     --screen screen_proteins.txt \
                     --chain_mapping chain_mapping.txt \
                     -n lig_screen
```

6. Interpreting the output

Folder layout produced by the wrapper

```
boltz_screen_(<DATE>or<NAME>)/
├─ inputs/                  # original bait, screen & mapping files (copied)
├─ results/
│   └─ 001_<JOB_NAME>/      # one folder per target (YAML + prediction output)
│       ├─ *.yaml
│       ├─ slurm_<ID>.out
│       ├─ lightning_logs/ …       (standard Boltz output)
│       ├─ predictions/ …          (CIF, PAE, pLDDT …)
│       └─ CHIMERAX_<JOB>_analysis.cxc
├─ slurm/                   # job-array *.tmp.out + analysis logs
├─ plots/                   # dashboards & dot-plots
├─ analytics/               # CSV tables (slurm_metrics, failed jobs …)
└─ summary_metrics.csv
```

1. **After the array finishes** the analysis job runs automatically.
2. Open
   plots/scatter_dashboard.html – interactive metric explorer (defaults: confidence_score × complex_pLDDT).
3. **summary_metrics.csv** – one row per job with all numeric metrics (OK and failed).
4. **CHIMERAX_…_analysis.cxc** inside every prediction folder – double-click to load the model with rainbow chains, interfaces, ligands highlighted & PAE overlay.

Using the scatter_dashboard.html:

When predicting a confident bait vs a list of candidates, the overall quality metrics such as plDDT scorte will be dominated by the confident bait protein.

It is therefore best to look at specific chain-chain contact metrics, which are given in the in `ipmtm_chain_i_vs_chain_j` metrics. Let us look at a specific example:

We ran a prediction with:

```
#bait:
P38919
P61326
Q9Y5S9
./RNA.fa
ATP
MG


#screen
a bunch of uniprots


#EJC_chain_mapping.txt
0=EIF4A3
1=MAGOH
2=Y14
3=RNA
4=screen
5=ATP
6=MG
```

In this case, The dropdowns from the dashboard will show amongst many more) the following metrics:

- *iptm_EIF4A3_vs_screen*
  and
- *iptm_screen_vs_EIF4A3*

Those scores will not be identical, since the metric is directional; the two numbers answer subtly different biological questions.

- The *iptm_EIF4A3_vs_screen* score asnswers
  “How well is the screened protein positioned relative to the (assumed correct) scaffold of EIF4A3?”.
- The *iptm_screen_vs_EIF4A3* score answers “How well is EIF4A3 positioned when I trust the screened protein as the anchor?”

More information on the metrics is available in the next section.

The Boltz-2 scatter dashboard displays several quantitative metrics for each predicted complex, plotted along the X-axis, Y-axis, marker size, and marker colour. The default configuration is:

- **X-axis**: **confidence_score**
- **Y-axis**: **complex_plddt**
- **Marker size**: **iptm**
- **Marker colour**: **complex_iplddt**

Each metric captures a different aspect of model confidence or structural accuracy:

- **confidence_score** is a composite ranking score combining overall per-residue confidence and interface accuracy. wemol.wecomput.com
- **complex_plddt** measures average per-residue confidence across the whole complex; high values indicate reliable fold predictions. wemol.wecomput.com
- **iptm** (interface predicted TM-score) quantifies the predicted accuracy of subunit interfaces; values > 0.8 imply high-quality interface modelling. ebi.ac.uk
- **complex_iplddt** weights per-residue confidence toward interface residues, highlighting how well the model resolved the binding region. wemol.wecomput.com

Below we unpack each metric, explain its range and interpretation, and suggest how to use it in practice.

1 confidence_score

- **Definition:**
  A linear combination of interface-focused confidence (**iptm**) and global per-residue confidence (**complex_plddt**):
  > `confidence_score = 0.8 × complex_plddt + 0.2 × iptm` wemol.wecomput.com
- **Range:** 0 to 1.0
- **Interpretation:**
  Combines overall fold reliability with interface accuracy into a single ranking score. Higher values indicate both a well-predicted global fold and a confidently modelled interface.
- **Use case:**
  • **Rapid prioritization** – sort by `confidence_score` to pick top candidates for detailed inspection or experimental validation.

• **Balancing fold vs. interface** – because it weights fold confidence more heavily (80%), high-scoring complexes generally have little risk of gross misfolding. wemol.wecomput.com

2 complex_plddt

- **Definition:**
  The **predicted Local Distance Difference Test** (pLDDT) averaged across all residues in the complex. pLDDT is AlphaFold-derived and repurposed by Boltz-2 for complexes. ebi.ac.uk
- **Range:** 0 to 1.0
- **Thresholds:**
  - > 0.7 – high overall confidence
  - 0.5–0.7 – moderate confidence ("gray zone")
  - < 0.5 – low confidence; likely disordered or misfolded regions
- **Interpretation:**
  Reflects local structural reliability at the residue level; values near 1.0 indicate the model is very confident in atomic positions. wemol.wecomput.com
- **Use case:**
  • **Assess global fold** – ensure your complex is not fundamentally misfolded before interpreting interface metrics.
  • **Color-coding** – use complex_plddt to colour survivors vs. failures in multi-parameter sweeps.

3 iptm (interface pTM)

- **Definition:**
  The **predicted TM-score** focused on residue-pair contacts at the interface between chains in a complex. It measures the expected structural similarity of interfaces to the true bound state. ebi.ac.uk
- **Range:** 0 to 1.0
- **Thresholds:**
  - > 0.8 – high-quality interface prediction
  - 0.6–0.8 – ambiguous ("gray zone")
  - < 0.6 – likely failed interface modelling
- **Interpretation:**
  Captures how well the model predicts relative positioning of subunits; high ipTM but low global pLDDT suggests the fold is good but local regions may be uncertain. wemol.wecomput.com

- **Use case:**
  • **Interface screening** – for affinity or docking applications, you often want ipTM > 0.8.
  • **Outlier detection** – points with high complex_plddt but low ipTM may have correct folds but misaligned interfaces.

4 complex_iplddt

- **Definition:**
  A variant of pLDDT that **weights per-residue confidence toward interface residues**, highlighting how confidently the model localizes binding-site regions. wemol.wecomput.com
- **Range:** 0 to 1.0
- **Interpretation:**
  Values close to global complex_plddt unless interface regions are especially poorly or well-resolved. A gap between complex_plddt and complex_iplddt signals heterogeneous confidence distributions. wemol.wecomput.com
- **Use case:**
  • **Refine ligand-binding candidates** – prefer complexes where both complex_plddt and complex_iplddt are high, indicating uniform confidence.
  • **Detect interface disorder** – a drop in complex_iplddt vs. complex_plddt warns of flexible or unresolved binding loops.

5 Additional metrics you might encounter

Although the scatter dashboard focuses on the four metrics above, you may also see:

- **pTM (global pTM)** – the overall TM-score for the entire complex, less interface-specific than ipTM ebi.ac.uk
- **Predicted Aligned Error (PAE)** – a 2D matrix showing residue-pair error estimates; use the PAE heat-maps for domain-interaction confidence en.wikipedia.org

These supplementary metrics can be plotted or referenced if you enable **all-plots** mode during analysis.

Frequently Asked Questions

Question	Answer
Can I screen ligand libraries?	Yes – put each CCD code or SMILES line in `--screen` .
Do I need MSAs for RNA/DNA?	No – MSA slots are ignored for nucleic acids.
Can I resume a failed screen?	Re-run the same command; existing `results/*/confidence_*.json` folders are skipped automatically.
Can start the analysis manually	Yes, you can run `boltz-analysis.sh <screen_dir> -c chain_map.txt` ’
Why are some predictions filtered out before they are even submitted	boltz-2 needs a lot of memory for large predictions, and in order not to waste resourced the wrapper script won’t submit predictions that will certainly fail on our hardware. (see memory consideration section)` ’