

Guide for trident v1.2.0.0 to v1.2.1.0

Contents

1	The trident CLI	1
1.1	General notes	4
1.1.1	Logging and command line output	4
1.1.2	Duplicates	4
1.1.3	Group names in .fam files	4
1.1.4	Whitespaces in the .janno file	4
2	Package creation and manipulation commands	4
2.1	Init command	4
2.2	Fetch command	6
2.3	Forge command	7
2.3.1	The forge selection language	10
2.3.2	Treatment of the .janno file while merging	11
2.3.3	Treatment of the .ssf file while merging	12
2.3.4	Other options	12
2.4	Genoconvert command	13
2.5	Update command	14
3	Inspection commands	16
3.1	List command	16
3.2	Summarise command	17
3.3	Survey command	17
3.4	Validate command	18

1 The trident CLI

Trident is a command line software tool structured in multiple subcommands. If you installed it properly you can call it on the command line by typing `trident`. This will show an overview of the general options and all subcommands, which are explained in detail below.

```
Usage: trident [--version] [--logMode ARG] [--errLength ARG]
           [--inPlinkPopName ARG] (COMMAND | COMMAND)
```

`trident` is a management and analysis tool for Poseidon packages. Report issues here: <https://github.com/poseidon-framework/poseidon-hs/issues>

34 Available options:

```
35  -h,--help          Show this help text
36  --version          Show version number
37  --logMode ARG      How information should be reported: NoLog, SimpleLog,
38                    DefaultLog, ServerLog or VerboseLog
39                    (default: DefaultLog)
40  --errLength ARG    After how many characters should a potential error
41                    message be truncated. "Inf" for no truncation.
42                    (default: CharCount 1500)
43  --inPlinkPopName ARG Where to read the population/group name from the FAM
44                    file in Plink-format. Three options are possible:
45                    asFamily (default) | asPhenotype | asBoth.
```

47 Package creation and manipulation commands:

```
48  init              Create a new Poseidon package from genotype data
49  fetch             Download data from a remote Poseidon repository
50  forge             Select packages, groups or individuals and create a
51                    new Poseidon package from them
52  genoconvert       Convert the genotype data in a Poseidon package to a
53                    different file format
54  update            Update POSEIDON.yml files automatically
```

56 Inspection commands:

```
57  list             List packages, groups or individuals from local or
58                    remote Poseidon repositories
59  summarise        Get an overview over the content of one or multiple
60                    Poseidon packages
61  summarize        Synonym for summarise
62  survey           Survey the degree of context information completeness
63                    for Poseidon packages
64  validate         Check one or multiple Poseidon packages for
65                    structural correctness
```

66 Trident allows to work directly with genotype data (see `-p` below), but its optimized for the interaction
67 with [Poseidon packages](#), which wrap and contextualize the data. Most trident subcommands therefore have a
68 central parameter, called `--baseDir` or simply `-d` to specify one or more base directories to look for packages.
69 For example, if all Poseidon packages live inside a repository at `/path/to/poseidon/packages` you would
70 simply say `trident <subcommand> -d /path/to/poseidon/dirs/` and `trident` would automatically search
71 all subdirectories inside of the repository for valid Poseidon packages (as identified by valid `POSEIDON.yml`
72 files).

73 You can arrange a poseidon repository in a hierarchical way. For example:

```
74 /path/to/poseidon/packages
75   /modern
76     /2019_poseidon_package1
77     /2019_poseidon_package2
```

```

78     /ancient
79     /...
80     /...
81     /Reference_Genomes
82     /...
83     /...

```

84 You can use this structure to select only the level of packages you're interested in, even individual ones, and you
85 can make use of the fact that `-d` can be given multiple times.

86 Being able to specify one or multiple repositories is often not enough, as you may have your own data to
87 co-analyse with the main repository. This is easy to do, as you simply need to provide your own genotype data
88 as yet another Poseidon package to be added to your `trident` command. For example, let's say you have
89 genotype data in `EIGENSTRAT` format (`trident` supports `EIGENSTRAT` and `PLINK` as formats.):

```

90 ~/my_project/my_project.geno
91 ~/my_project/my_project.snp
92 ~/my_project/my_project.ind

```

93 then you can make that to a skeleton Poseidon package with the `init` command. You can also do it manually
94 by simply adding a `POSEIDON.yml` file, with for example the following content:

```

95 poseidonVersion: 2.5.0
96 title: My_awesome_project
97 description: Unpublished genetic data from my awesome project
98 contributor:
99   - name: Stephan Schiffels
100     email: schiffels@institute.org
101 packageVersion: 0.1.0
102 lastModified: 2020-10-07
103 genotypeData:
104   format: EIGENSTRAT
105   genoFile: my_project.geno
106   snpFile: my_project.snp
107   indFile: my_project.ind
108   jannoFile: my_project.janno
109   bibFile: sources.bib

```

110 Two remarks: 1) all file paths are considered *relative* to the directory in which `POSEIDON.yml` resides. Here we
111 assume that you put this file into the same directory as the three genotype files. 2) Besides the genotype data
112 files there are two (technically optional) files referenced by this example `POSEIDON.yml` file: `sources.bib`
113 and `my_project.janno`. Of course you can add them manually - `init` automatically creates empty dummy
114 versions.

115 Once you have set up your own “Poseidon” package (which is really only a skeleton so far), you can add it to
116 your `trident` analysis, by simply adding your project directory to the command using `-d`, for example:

```

117 trident list -d /path/to/poseidon/packages/modern \
118   -d /path/to/poseidon/packages/ReferenceGenomes
119   -d ~/my_project --packages

```

1.1 General notes

1.1.1 Logging and command line output

For all subcommands the general argument `--logMode` defines how trident reports messages (to stderr) on the command line:

- *NoLog*: Hides all messages.
- *SimpleLog*: Plain and simple output to stderr.
- *DefaultLog*: Adds severity indicators before each message. (default setting)
- *ServerLog*: Additionally adds timestamps before each message.
- *VerboseLog*: Shows not just messages on the log levels `Info`, `Warning` and `Error` like the other modes, but also on the more verbose level `Debug`. Use this for debugging.

1.1.2 Duplicates

- If multiple packages in a package repository share the same `title`, then trident will try to select the one with the highest version number. If this is not sufficient to resolve the conflict, trident will stop. An exception for that is the `list` subcommand, which will read and report all packages/groups/individuals in all versions.
- Individual/sample names (`Poseidon_ID` s) within one package have to be unique, or trident will stop.
- We generally also discourage ID duplicates across packages in package repositories, but trident will generally continue with them after printing a warning. This does not apply for `validate`, by default (you can change this behaviour with `--ignoreDuplicates`), and `forge`. `forge` offers a special mechanism to resolve duplicates within its selection language (see below).

1.1.3 Group names in .fam files

The `.fam` file of Plink-formatted genotype data is used inconsistently across different popular aDNA software tools to store group/population name information. The (global) option `--inPlinkPopName` with the arguments `asFamily` (default), `asPhenotype` and `asBoth` allows to control the reading of the population name from Plink `.fam` files. The subcommands that write genotype data (`forge`, `genoconvert`) have a corresponding option `--outPlinkPopName` to specify this for the output.

1.1.4 Whitespaces in the .janno file

While reading the `.janno` file `trident` trims all leading and trailing whitespaces around individual cells. Also all instances of the `No-Break Space` unicode character will be removed. This means these whitespaces will not be preserved when a package is `forge` d.

2 Package creation and manipulation commands

2.1 Init command

`init` creates a new, valid Poseidon package from genotype data files. It adds a valid `POSEIDON.yml` file, a dummy `.janno` file for context information and an empty `.bib` file for literature references.

[Click here for command line details](#)

Usage: `trident init ((-p|--genoOne ARG) | --inFormat ARG --genoFile ARG`

```

156         --snpFile ARG --indFile ARG) [--snpSet ARG]
157         (-o|--outPackagePath ARG) [-n|--outPackageName ARG]
158         [--minimal]
159     Create a new Poseidon package from genotype data
160
161     Available options:
162     -h,--help                Show this help text
163     -p,--genoOne ARG         one of the input genotype data files. Expects .bed or
164                             .bim or .fam for PLINK and .geno or .snp or .ind for
165                             EIGENSTRAT. The other files must be in the same
166                             directory and must have the same base name
167     --inFormat ARG           the format of the input genotype data: EIGENSTRAT or
168                             PLINK (only necessary for data input with --genoFile
169                             + --snpFile + --indFile)
170     --genoFile ARG           the input geno file path
171     --snpFile ARG            the input snp file path
172     --indFile ARG            the input ind file path
173     --snpSet ARG             the snpSet of the package: 1240K, HumanOrigins or
174                             Other. (only relevant for data input with
175                             -p|--genoOne or --genoFile + --snpFile + --indFile,
176                             because the packages in a -d|--baseDir already have
177                             this information in their respective POSEIDON.yml
178                             files) Default: Other
179     -o,--outPackagePath ARG  the output package directory path
180     -n,--outPackageName ARG  the output package name - this is optional: If no
181                             name is provided, then the package name defaults to
182                             the basename of the (mandatory) --outPackagePath
183                             argument
184     --minimal                should only a minimal output package be created?
185
186     The command
187
188     trident init \
189     --inFormat EIGENSTRAT/PLINK \
190     --genoFile path/to/geno_file \
191     --snpFile path/to/snp_file \
192     --indFile path/to/ind_file \
193     --snpSet 1240K|HumanOrigins|Other \
194     -o path/to/new_package_name
195
196     requires the format ( --inFormat ) of your input data (either EIGENSTRAT or PLINK), the paths to the
197     respective files ( --genoFile , --snpFile , --indFile ), and optionally the “shape” of these files ( --snpSet ),
198     so if they cover the 1240K, the HumanOrigins or an Other SNP set. A simpler interface added in trident
199     0.29.0 is available with -p (+ --snpSet) .

```

	EIGENSTRAT	PLINK
genoFile	.geno	.bed

	EIGENSTRAT	PLINK
snpFile	.snp	.bim
indFile	.ind	.fam

The output package of `init` is created as a new directory `-o`, which should not already exist, and gets the package `title` corresponding to the basename of `-o`. You can also set the title explicitly with `-n`. The `--minimal` flag causes `init` to create a minimal package with a very basic `POSEIDON.yml` and no `.bib` and `.janno` files.

2.2 Fetch command

`fetch` allows to download Poseidon packages from a remote Poseidon server. Read more about this repository [here](#).

Click here for command line details

```
Usage: trident fetch (-d|--baseDir DIR)
                (--downloadAll |
                (--fetchFile ARG | (-f|--fetchString ARG)))
                [--remoteURL ARG]
```

Download data from a remote Poseidon repository

Available options:

<code>-h,--help</code>	Show this help text
<code>-d,--baseDir DIR</code>	a base directory to search for Poseidon Packages (could be a Poseidon repository)
<code>--downloadAll</code>	download all packages the server is offering
<code>--fetchFile ARG</code>	A file with a list of packages. Works just as <code>-f</code> , but multiple values can also be separated by newline, not just by comma. <code>-f</code> and <code>--fetchFile</code> can be combined.
<code>-f,--fetchString ARG</code>	List of packages to be downloaded from the remote server. Package names should be wrapped in asterisks: <code>*package_title*</code> . You can combine multiple values with comma, so for example: <code>"*package_1*, *package_2*, *package_3*"</code> . <code>fetchString</code> uses the same parser as <code>forgeString</code> , but does not allow excludes. If groups or individuals are specified, then packages which include these groups or individuals are included in the download.
<code>--remoteURL ARG</code>	URL of the remote Poseidon server (default: "https://server.poseidon-adna.org")

It works with

```
trident fetch -d ... -d ... \
-f "*package_title_1*,*package_title_2*,*package_title_3*,group_name,<individual1>"
```

and the entities you want to download must be listed either in a simple string of comma-separated values, which can be passed via `-f / --fetchString`, or in a text file (`--fetchFile`). Entities are then combined from these sources.

Entities are specified using a special syntax (see also the documentation of `forge` below): Package titles are wrapped in asterisks: `*package_title*`, group names are spelled as is, and individual names are wrapped in angular brackets, so `<individual1>`. Fetch will figure out which packages need to be downloaded to include all specified entities. `--downloadAll`, which can be given instead of `-f` and `--fetchFile`, causes fetch to download all packages from the server. The downloaded packages are added in the first (!) `-d` directory (which gets created if it doesn't exist), but downloads are only performed if the respective packages are not already present in the latest version in any of the `-d` dirs.

Note that `trident fetch` makes most sense in combination with `trident list --remote`: First one can inspect what is available on the server, then one can create a custom fetch command.

`fetch` also has the optional arguments `--remote https://..."` to name an alternative poseidon server. The default points to the [DAG server](#).

2.3 Forge command

`forge` creates new Poseidon packages by extracting and merging packages, populations and individuals from your Poseidon repositories.

[Click here for command line details](#)

```
Usage: trident forge ((-d|--baseDir DIR) |
    ((-p|--genoOne ARG) | --inFormat ARG --genoFile ARG
    --snpFile ARG --indFile ARG) [--snpSet ARG])
    [--forgeFile ARG | (-f|--forgeString ARG)]
    [--selectSnps ARG] [--intersect] [--outFormat ARG]
    [--minimal] [--onlyGeno] (-o|--outPackagePath ARG)
    [-n|--outPackageName ARG] [--packagewise]
    [--outPlinkPopName ARG]
Select packages, groups or individuals and create a new Poseidon package from
them
```

Available options:

<code>-h,--help</code>	Show this help text
<code>-d,--baseDir DIR</code>	a base directory to search for Poseidon Packages (could be a Poseidon repository)
<code>-p,--genoOne ARG</code>	one of the input genotype data files. Expects .bed or .bim or .fam for PLINK and .geno or .snp or .ind for EIGENSTRAT. The other files must be in the same directory and must have the same base name
<code>--inFormat ARG</code>	the format of the input genotype data: EIGENSTRAT or PLINK (only necessary for data input with --genoFile + --snpFile + --indFile)
<code>--genoFile ARG</code>	the input geno file path
<code>--snpFile ARG</code>	the input snp file path

276 --indFile ARG the input ind file path

277 --snpSet ARG the snpSet of the package: 1240K, HumanOrigins or

278 Other. (only relevant for data input with

279 -p|--genoOne or --genoFile + --snpFile + --indFile,

280 because the packages in a -d|--baseDir already have

281 this information in their respective POSEIDON.yml

282 files) Default: Other

283 --forgeFile ARG A file with a list of packages, groups or individual

284 samples. Works just as -f, but multiple values can

285 also be separated by newline, not just by comma.

286 Empty lines are ignored and comments start with "#",

287 so everything after "#" is ignored in one line.

288 Multiple instances of -f and --forgeFile can be

289 given. They will be evaluated according to their

290 input order on the command line.

291 -f,--forgeString ARG List of packages, groups or individual samples to be

292 combined in the output package. Packages follow the

293 syntax *package_title*, populations/groups are simply

294 group_id and individuals <individual_id>. You can

295 combine multiple values with comma, so for example:

296 "*package_1*, <individual_1>, <individual_2>,"

297 group_1". Duplicates are treated as one entry.

298 Negative selection is possible by prepending "-" to

299 the entity you want to exclude (e.g. "*package_1*,"

300 -<individual_1>, -group_1"). forge will apply

301 excludes and includes in order. If the first entity

302 is negative, then forge will assume you want to merge

303 all individuals in the packages found in the baseDirs

304 (except the ones explicitly excluded) before the

305 exclude entities are applied. An empty forgeString

306 (and no --forgeFile) will therefore merge all

307 available individuals. If there are individuals in

308 your input packages with equal individual id, but

309 different main group or source package, they can be

310 specified with the special syntax

311 "<package:group:individual>".

312 --selectSnps ARG To extract specific SNPs during this forge operation,

313 provide a Snp file. Can be either Eigenstrat (file

314 ending must be '.snp') or Plink (file ending must be

315 '.bim'). When this option is set, the output package

316 will have exactly the SNPs listed in this file. Any

317 SNP not listed in the file will be excluded. If

318 option '--intersect' is also set, only the SNPs

319 overlapping between the SNP file and the forged

320 packages are output.

321 `--intersect` Whether to output the intersection of the genotype
322 files to be forged. The default (if this option is
323 not set) is to output the union of all SNPs, with
324 genotypes defined as missing in those packages which
325 do not have a SNP that is present in another package.
326 With this option set, the forged dataset will
327 typically have fewer SNPs, but less missingness.

328 `--outFormat ARG` the format of the output genotype data: EIGENSTRAT or
329 PLINK. Default: PLINK

330 `--minimal` should only a minimal output package be created?

331 `--onlyGeno` should only the resulting genotype data be returned?
332 This means the output will not be a Poseidon package

333 `-o,--outPackagePath ARG` the output package directory path

334 `-n,--outPackageName ARG` the output package name - this is optional: If no
335 name is provided, then the package name defaults to
336 the basename of the (mandatory) `--outPackagePath`
337 argument

338 `--packagewise` Skip the within-package selection step in forge. This
339 will result in outputting all individuals in the
340 relevant packages, and hence a superset of the
341 requested individuals/groups. It may result in better
342 performance in cases where one wants to forge entire
343 packages or almost entire packages. Details: Forge
344 conceptually performs two types of selection: First,
345 it identifies which packages in the supplied base
346 directories are relevant to the requested forge, i.e.
347 whether they are either explicitly listed using
348 `*PackageName*`, or because they contain selected
349 individuals or groups. Second, within each relevant
350 package, individuals which are not requested are
351 removed. This option skips only the second step, but
352 still performs the first.

353 `--outPlinkPopName ARG` Where to write the population/group name into the FAM
354 file in Plink-format. Three options are possible:
355 `asFamily` (default) | `asPhenotype` | `asBoth`. See also
356 `--inPlinkPopName`.

357 `forge` can be used with

```
358 trident forge -d ... -d ... \
359   -f "*package_name*, group_id, <individual_id>" \
360   -o path/to/new_package_name
```

361 where the entities (packages, groups/populations, individuals/samples) you want in the output package can be
362 denoted either as a string on the command line (`-f / --forgeString`), or in an input text file (`--forgeFile`).
363 See the section below for the syntax of this selection language. Do not forget to wrap the `--forgeString` query
364 in quotes.

365 Including one or multiple Poseidon packages with `-d` is not the only way to include data for a forge
 366 operation. It is also possible to consider unpackaged genotype data directly with `-p (+ --snpSet)` or
 367 `--inFormat + --genoFile + --snpFile + --indFile (+ --snpSet)`. This makes the following example
 368 possible, where we merge data from one Poseidon package and two genotype datasets to get a new EIGENSTRAT
 369 dataset.

```
370 trident forge \
371   -d 2017_GonzalesFortesCurrentBiology \
372   -p 2018_VeeramahPNAS/2018_VeeramahPNAS.fam \
373   --inFormat PLINK \
374   --genoFile 2017_HaberAJHG/2017_HaberAJHG.bed \
375   --snpFile 2017_HaberAJHG/2017_HaberAJHG.bim \
376   --indFile 2017_HaberAJHG/2017_HaberAJHG.fam \
377   -f "<STR241.SG>,<ERS1790729.SG>,Iberia_HG.SG" \
378   -o testpackage \
379   --outFormat EIGENSTRAT \
380   --onlyGeno
```

381 2.3.1 The forge selection language

382 The text in `--forgeString` and `--forgeFile` are parsed as a domain specific query language that describes
 383 precisely which entities should be compiled in the output package of a given `forge` operation. The language
 384 has multiple syntactic elements and a specific evaluation logic.

385 In general a `--forgeString` query consists of multiple entities, separated by `,`. The main entities are Poseidon
 386 packages, groups/populations and individuals/samples:

- 387 • Each package title is surrounded by `* : *package*`. That means if you want all individuals of the Poseidon
 388 package `2019_Jeong_InnerEurasia` in the output package you would add `*2019_Jeong_InnerEurasia*`
 389 to the query.
- 390 • Groups/populations are not specially marked: `group`. So to get all individuals of the group
 391 `Swiss_Roman_period`, you would simply add `Swiss_Roman_period`.
- 392 • Individuals/samples are surrounded by `<` and `>`: `<individual>`. `ALA026` therefore becomes
 393 `<ALA026>`. A second way to denote individuals is with the more verbose and specific syntax
 394 `<package:group:individual>`. Such defined individuals take precedence over differently defined ones
 395 (so: directly with `<individual>` or as a subset of `*package*` or `group`). This allows to resolve
 396 duplication issues precisely – at least in cases where the duplicated individuals differ in source package or
 397 primary group.

398 In the `--forgeFile` each line is treated as a separate `forgeString`, empty lines are ignored and `#`s start
 399 comments. So this is a valid `forgeFile`:

```
400 # Packages
401 *package1*, *package2*
402
403 # Groups and individuals from other packages beyond package1 and package2
404 group1, <individual1>, group2, <individual2>, <individual3>
405
406 # group2 has two outlier individuals that should be ignored
```

```

407 -<bad_individual1> # This one has very low coverage
408 -<bad_individual2> # This one is from a different time period

```

By prepending `-` to the bad individuals, we can exclude them from the forged package. `forge` figures out the final list of samples to include by executing all forge-entities in order. So an entity list `*PackageA*,-<Individual1>,GroupA` may result in a different outcome than `*PackageA*,GroupA,-<Individual1>`, depending on whether `<Individual1>` belongs to `GroupA` or not. If the forge entity list starts with a negative entity, or if the entity list is empty, `forge` will implicitly assume you want to include all individuals in all packages found in the baseDirs (except the ones explicitly excluded, of course).

An empty forgeString will therefore merge all available individuals.

2.3.2 Treatment of the .janno file while merging

`forge` merges and subsets .janno files along with the genotype data. If a package lacks a .janno file, then a basic one will be created internally based on the information in the genotype data, and used for the output. Missing columns across packages will be filled with `n/a`.

For merging two .janno files **A** and **B** the following rules apply regarding undefined, arbitrary additional columns:

- If **A** has an additional column which is not in **B** then empty cells in the rows imported from **B** are filled with `n/a`.
- If **A** and **B** share additional columns with identical column name, then they are treated as semantically identical units and merged accordingly.
- In the resulting .janno file, all additional columns from both **A** and **B** are sorted alphabetically and appended after the normal, specified variables.

The following example illustrates the described behaviour:

A.janno

Poseidon_ID	Group_Name	Genetic_Sex	AdditionalColumn1	AdditionalColumn2
XXX011	POP1	M	A	D
XXX012	POP2	F	B	E
XXX013	POP1	M	C	F

B.janno

Poseidon_ID	Group_Name	Genetic_Sex	AdditionalColumn3	AdditionalColumn2
YYY022	POP5	F	G	J
YYY023	POP5	F	H	K
YYY024	POP5	M	I	L

A.janno + B.janno

Poseidon_ID	Group_Name	Genetic_Sex	AdditionalColumn1	AdditionalColumn2	AdditionalColumn3
XXX011	POP1	M	A	D	n/a
XXX012	POP2	F	B	E	n/a

Poseidon_ID	Group_Name	Genetic_Sex	AdditionalColumn1	AdditionalColumn2	AdditionalColumn3
XXX013	POP1	M	C	F	n/a
YYY022	POP5	F	n/a	J	G
YYY023	POP5	F	n/a	K	H
YYY024	POP5	M	n/a	L	I

2.3.3 Treatment of the .ssf file while merging

The Sequencing Source File (short .ssf file) is forged in exactly the same way as the janno file. SSF files that are present are included in the forge product in the way that the user expects, following selection of those entities which are listed in the `poseidon_IDs` columns of the SSF files. Columns that are only present in some packages, including those not defined by our [Schema] are also included in the forged product in the same way as described for Janno above.

2.3.4 Other options

Just as for `init` the output package of `forge` is created as a new directory `-o`. The title can also be explicitly defined with `-n`.

`--minimal` allows for the creation of a minimal output package without `.bib` and `.janno`. This is especially useful for data analysis pipelines, where only the genotype data is required. Even more basic output comes with `--onlyGeno`, which means that only the genotype data is returned without any Poseidon package.

`forge` has a an optional flag `--intersect`, that defines, if the genotype data from different packages should be merged with an `union` or an `intersect` operation. The default (if this option is not set) is to output the union of all SNPs, with genotypes defined as missing in samples from packages which do not have a SNP that is present in another package. With this option set, on the other hand, the forged dataset will typically have fewer SNPs, but less missingness.

`--intersect` also influences the automatic determination of the `snpSet` field in the POSEIDON.yml file for the resulting package. If the `snpSet` s of all input packages are identical, then the resulting package will just inherit this configuration. Otherwise `forge` applies the following pairwise merging logic:

Input snpSet A	Input snpSet B	<code>--intersect</code>	Ouput snpSet
Other	*	*	Other
1240K	HumanOrigins	True	HumanOrigins
1240K	HumanOrigins	False	1240K

`--selectSnps` allows to provide `forge` with a SNP file in EIGENSTRAT (`.snp`) or PLINK (`.bim`) format to create a package with a specific selection. When this option is set, the output package will have exactly the SNPs listed in this file. Any SNP not listed in the file will be excluded. If `--intersect` is also set, only the SNPs overlapping between the SNP file and the forged packages are output.

Merging genotype data across different data sources and file formats is tricky. `forge` is more verbose about potential issues, if the `--logMode` flag is set to `VerboseLog`.

The `--onlyGeno` command specifies that only genotype data should be output, not an entire Poseidon package.

458 With `--packagewise` the within-package selection step in forge can be skipped. This will result in outputting
459 all individuals in the relevant packages, and hence a superset of the requested individuals/groups. It may result
460 in better performance in cases where one wants to forge entire packages.

461 2.4 Genoconvert command

462 `genoconvert` converts the genotype data in a Poseidon package to a different file format. The respective entries
463 in the POSEIDON.yml file are changed accordingly.

464 [Click here for command line details](#)

```
465 Usage: trident genoconvert ((-d|--baseDir DIR) |  
466                             ((-p|--genoOne ARG) | --inFormat ARG --genoFile ARG  
467                             --snpFile ARG --indFile ARG) [--snpSet ARG])  
468                             --outFormat ARG [--onlyGeno]  
469                             [-o|--outPackagePath ARG] [--removeOld]  
470                             [--outPlinkPopName ARG]
```

471 Convert the genotype data in a Poseidon package to a different file format

472
473 Available options:

474	<code>-h,--help</code>	Show this help text
475	<code>-d,--baseDir DIR</code>	a base directory to search for Poseidon Packages (could be a Poseidon repository)
476		
477	<code>-p,--genoOne ARG</code>	one of the input genotype data files. Expects .bed or 478 .bim or .fam for PLINK and .geno or .snp or .ind for 479 EIGENSTRAT. The other files must be in the same 480 directory and must have the same base name
481	<code>--inFormat ARG</code>	the format of the input genotype data: EIGENSTRAT or 482 PLINK (only necessary for data input with --genoFile 483 + --snpFile + --indFile)
484	<code>--genoFile ARG</code>	the input geno file path
485	<code>--snpFile ARG</code>	the input snp file path
486	<code>--indFile ARG</code>	the input ind file path
487	<code>--snpSet ARG</code>	the snpSet of the package: 1240K, HumanOrigins or 488 Other. (only relevant for data input with 489 -p --genoOne or --genoFile + --snpFile + --indFile, 490 because the packages in a -d --baseDir already have 491 this information in their respective POSEIDON.yml 492 files) Default: Other
493	<code>--outFormat ARG</code>	the format of the output genotype data: EIGENSTRAT or 494 PLINK.
495	<code>--onlyGeno</code>	should only the resulting genotype data be returned? 496 This means the output will not be a Poseidon package
497	<code>-o,--outPackagePath ARG</code>	the output package directory path - this is optional: 498 If no path is provided, then the output is written to 499 the directories where the input genotype data file 500 (.bed/.geno) is stored

```

501  --removeOld          Remove the old genotype files when creating the new
502                      ones
503  --outPlinkPopName ARG Where to write the population/group name into the FAM
504                      file in Plink-format. Three options are possible:
505                      asFamily (default) | asPhenotype | asBoth. See also
506                      --inPlinkPopName.

```

507 With the default setting

```

508 trident genoconvert -d ... -d ... --outFormat EIGENSTRAT|PLINK

```

509 all packages in `-d` will be converted to the desired `--outFormat` (either `EIGENSTRAT` or `PLINK`), if the data
510 is not already in this format. This includes updating the respective POSEIDON.yml files.

511 The “old” data is not deleted, but kept around. That means conversion can result in a package with both PLINK
512 and EIGENSTRAT data, but only one is linked in the POSEIDON.yml file, and that is what will be used by
513 trident. To delete the old data in the conversion you can add the `--removeOld` flag.

514 Instead of `-d` to change Poseidon packages, the `-p (+ --snpSet)` or `--inFormat + --genoFile + --snpFile + --indFi`
515 allow to directly convert genotype data that is not wrapped in a Poseidon package and store it to a directory
516 given in `-o`. See this example:

```

517 trident genoconvert \
518   -p 2018_Mittnik_Baltic/Mittnik_Baltic.bed \
519   --outFormat EIGENSTRAT
520   -o my_directory

```

521 2.5 Update command

522 `update` automatically harmonizes POSEIDON.yml files of one or multiple packages if the packages were
523 changed. This is not an automatic update from one Poseidon version to the next!

524 [Click here for command line details](#)

```

525 Usage: trident update (-d|--baseDir DIR) [--poseidonVersion ARG]
526                      [--ignorePoseidonVersion] [--versionComponent ARG]
527                      [--noChecksumUpdate] [--newContributors ARG]
528                      [--logText ARG] [--force]

```

529 Update POSEIDON.yml files automatically

531 Available options:

```

532  -h,--help          Show this help text
533  -d,--baseDir DIR   a base directory to search for Poseidon Packages
534                      (could be a Poseidon repository)
535  --poseidonVersion ARG Poseidon version the packages should be updated to:
536                      e.g. "2.5.3" (default: Nothing)
537  --ignorePoseidonVersion Read packages even if their poseidonVersion is not
538                      compatible with the trident version. The assumption
539                      is, that the package is already structurally adjusted
540                      to the trident version and only the version number is
541                      lagging behind.

```

```

542 --versionComponent ARG    Part of the package version number in the
543                             POSEIDON.yml file that should be updated: Major,
544                             Minor or Patch (see https://semver.org)
545                             (default: Patch)
546 --noChecksumUpdate        Should update of checksums in the POSEIDON.yml file
547                             be skipped
548 --ignoreGeno              ignore SNP and GenoFile
549 --newContributors ARG     Contributors to add to the POSEIDON.yml file in the
550                             form "[Firstname Lastname](Email address);..."
551 --logText ARG             Log text for this version jump in the CHANGELOG file
552                             (default: "not specified")
553 --force                   Normally the POSEIDON.yml files are only changed if
554                             the poseidonVersion is adjusted or any of the
555                             checksums change. With --force a package version
556                             update can be triggered even if this is not the case.

```

557 It can be called with a lot of optional arguments

```

558 trident update -d ... -d ... \
559   --poseidonVersion "X.X.X" \
560   --versionComponent Major/Minor/Patch \
561   --noChecksumUpdate
562   --ignoreGeno
563   --newContributors "[Firstname Lastname](Email address);..."
564   --logText "short description of the update"
565   --force

```

566 By default `update` will not edit a package's POSEIDON.yml file, even when arguments like `--versionComponent`,
567 `--newContributors` or `--logText` are explicitly set. This default exists to run the function on a large set of
568 packages where only few of them were edited and need an active update. A package will only be modified by
569 `update` if either

- 570 • any of the files with checksums (e.g. the genotype data) in it were modified,
- 571 • the `--poseidonVersion` argument differs from the `poseidonVersion` in the package's POSEIDON.yml
572 file
- 573 • or the `--force` flag was set in `update`.

574 If any of these applies to a package in the search directory (`--baseDir / -d`), it will be updated. This includes
575 the following steps:

- 576 • If `--poseidonVersion` is different from the `poseidonVersion` field in the package, then that will be
577 updated.
- 578 • The `packageVersion` will be incremented. If `--versionComponent` is not set, then it falls back to
579 `Patch`, so a change in the last position of the three digit version number. `Minor` increments the middle,
580 and `Major` the first position (see [semantic versioning](#)).
- 581 • The `lastModified` field will be updated to the current day (based on your computer's system time).
- 582 • The contributors in `--newContributors` will be added to the `contributor` field if they're not there
583 already.
- 584 • If any checksums changed, then they will be updated. If certain checksums are not set yet, then they will

585 be added. The checksum update can be skipped with `--noChecksumUpdate` or partially skipped for the
 586 genotype data with `--ignoreGeno`.

- 587 • The CHANGELOG.md file will be updated with a new row for the new version and the text in `--logText`
 588 (default: “not specified”), which will be appended as the first line of the file. If no CHANGELOG.md file
 589 exists, then it will be created and referenced in the POSEIDON.yml file.

590 :heavy_exclamation_mark: As `update` reads and rewrites POSEIDON.yml files, it may change their inner
 591 order, layout or even content (e.g. if they have fields which are not in the [Poseidon package definition](#)). Create a
 592 backup of the POSEIDON.yml file before running `update` if you are uncertain.

593 3 Inspection commands

594 3.1 List command

595 `list` lists packages, groups and individuals of the datasets you use, or of the packages available on the server.

596 [Click here for command line details](#)

```
597 Usage: trident list ((-d|--baseDir DIR) | --remote [--remoteURL ARG])
598                 (--packages | --groups | --individuals
599                 [-j|--jannoColumn JANNO_HEADER]) [--raw]
```

601 List packages, groups or individuals from local or remote Poseidon
 602 repositories

603 Available options:

605 <code>-h,--help</code>	Show this help text
606 <code>-d,--baseDir DIR</code>	a base directory to search for Poseidon Packages (could be a Poseidon repository)
607 <code>--remote</code>	list packages from a remote server instead the local file system
608 <code>--remoteURL ARG</code>	URL of the remote Poseidon server (default: "https://server.poseidon-adna.org")
609 <code>--packages</code>	list all packages
610 <code>--groups</code>	list all groups, ignoring any group names after the first as specified in the Janno-file
611 <code>--individuals</code>	list individuals
612 <code>-j,--jannoColumn JANNO_HEADER</code>	list additional fields from the janno files, using the Janno column heading name, such as Country, Site, Date_C14_Uncal_BP, Endogenous, ...
613 <code>--raw</code>	output table as tsv without header. Useful for piping into grep or awk

622 To list packages from your local repositories, as seen above you can run

```
623 trident list -d ... -d ... --packages
```

624 This will yield a nicely formatted table of all packages, their last update and the number of individuals in it.

625 To view packages on the remote server, instead of using directories to specify the locations of repositories on
626 your system, you can use `--remote` to show packages on the remote server. For example

```
627 trident list --packages --remote
```

628 will result in a view of all published packages in our [public online repository](#).

629 You can also list groups, as defined in the third column of EIGENSTRAT `.ind` files (or the first/last column of
630 a PLINK `.fam` file), and individuals with `--groups` and `--individuals` instead of `--packages`.

631 The `--individuals` flag provides a way to immediately access information from the `.janno`
632 files on the command line. This works with the `-j / --jannoColumn` option. For example adding
633 `-j Country -j Date_C14_Uncal_BP` to the commands above will add the `Country` and the `Date_C14_Uncal_BP`
634 columns to the respective output tables.

635 Note that if you want a less fancy table, for example because you want to load this into Excel, or pipe into
636 another command that cannot deal with the neat table layout, you can use the `--raw` option to output that
637 table as a simple tab-delimited stream.

638 3.2 Summarise command

639 `summarise` prints some general summary statistics for a given poseidon dataset taken from the `.janno` files.

640 [Click here for command line details](#)

```
641 Usage: trident summarise (-d|--baseDir DIR) [--raw]
```

```
642   Get an overview over the content of one or multiple Poseidon packages
```

```
643
644 Available options:
645   -h,--help           Show this help text
646   -d,--baseDir DIR    a base directory to search for Poseidon Packages
647                       (could be a Poseidon repository)
648   --raw               output table as tsv without header. Useful for piping
649                       into grep or awk
```

650 You can run it with

```
651 trident summarise -d ... -d ...
```

652 which will show you context information like – among others – the number of individuals in the dataset, their
653 sex distribution, the mean age of the samples (for ancient data) or the mean coverage on the 1240K SNP array
654 in a table. `summarise` depends on complete `.janno` files and will silently ignore missing information for some
655 statistics.

656 You can use the `--raw` option to output the summary table in a simple, tab-delimited layout.

657 3.3 Survey command

658 `survey` tries to indicate package completeness (mostly focused on `.janno` files) for poseidon datasets.

659 [Click here for command line details](#)

```
660 Usage: trident survey (-d|--baseDir DIR) [--raw]
```

```
661   Survey the degree of context information completeness for Poseidon packages
```

662

663 Available options:

664	<code>-h,--help</code>	Show this help text
665	<code>-d,--baseDir DIR</code>	a base directory to search for Poseidon Packages (could be a Poseidon repository)
666		
667	<code>--raw</code>	output table as tsv without header. Useful for piping into grep or awk
668		

669 Running

670 `trident survey -d ... -d ...`

671 will yield a table with one row for each package. See `trident survey -h` for a legend which cell of this table
672 means what.

673 Again you can use the `--raw` option to output the survey table in a tab-delimited format.

674 3.4 Validate command

675 `validate` checks poseidon datasets for structural correctness.

676 Click here for command line details

677 Usage: `trident validate (-d|--baseDir DIR)`

678 Check one or multiple Poseidon packages for structural correctness

679

680 Available options:

681	<code>-h,--help</code>	Show this help text
682	<code>-d,--baseDir DIR</code>	a base directory to search for Poseidon Packages (could be a Poseidon repository)
683		
684	<code>--ignoreGeno</code>	ignore SNP and GenoFile
685	<code>--fullGeno</code>	test parsing of all SNPs (by default only the first 100 SNPs are probed)
686		
687	<code>--noExitCode</code>	do not produce an explicit exit code
688	<code>--ignoreDuplicates</code>	do not stop on duplicated individual names in the package collection
689		

690 You can run it with

691 `trident validate -d ... -d ...`

692 and it will either report a success (`Validation passed`) or failure with specific error messages to simplify
693 fixing the issues.

694 `validate` tries to ensure that each package in the dataset adheres to the [schema definition](#). Here is a list of
695 what is checked:

- 696 • Presence of the necessary files
- 697 • Full structural correctness of .bib and .janno file
- 698 • Superficial correctness of genotype data files by parsing the first 100 SNPs. A full check that parses all
699 SNPs can be run with the `--fullGeno` option
- 700 • Correspondence of BibTeX keys in .bib and .janno

701 • Correspondence of individual and group IDs in .janno and genotype data files

702 In fact much of this validation already runs as part of the general package reading pipeline invoked for many
703 trident subcommands (e.g. `forge`). `validate` is meant to be more thorough, though, and will explicitly fail if
704 even a single package is broken.

705 Remember to run it with `--logMode VerboseLog` to get more information if the output is not sufficient to
706 debug an issue.