

Contents

0.1	Guide for trident v1.1.11.0 to v1.1.12.0	1
0.1.1	The trident CLI	1
0.1.2	Package creation and manipulation commands	4
0.1.3	Inspection commands	15

0.1 Guide for trident v1.1.11.0 to v1.1.12.0

0.1.1 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>

Available options:

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

Package creation and manipulation commands:

<code>init</code>	Create a new Poseidon package from genotype data
<code>fetch</code>	Download data from a remote Poseidon repository
<code>forge</code>	Select packages, groups or individuals and create a new Poseidon package from them
<code>genoconvert</code>	Convert the genotype data in a Poseidon package to a different file format
<code>update</code>	Update POSEIDON.yml files automatically

Inspection commands:

<code>list</code>	List packages, groups or individuals from local or remote Poseidon repositories
<code>summarise</code>	Get an overview over the content of one or multiple Poseidon packages

```

43  summarize          Synonym for summarise
44  survey             Survey the degree of context information completeness
45                    for Poseidon packages
46  validate           Check one or multiple Poseidon packages for
47                    structural correctness

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

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

55  /path/to/poseidon/packages
56      /modern
57          /2019_poseidon_package1
58          /2019_poseidon_package2
59      /ancient
60          /...
61          /...
62      /Reference_Genomes
63          /...
64          /...

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

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

71  ~/my_project/my_project.geno
72  ~/my_project/my_project.snp
73  ~/my_project/my_project.ind

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

76  poseidonVersion: 2.5.0
77  title: My_awesome_project
78  description: Unpublished genetic data from my awesome project
79  contributor:
80      - name: Stephan Schiffels
81        email: schiffels@institute.org
82  packageVersion: 0.1.0
83  lastModified: 2020-10-07
84  genotypeData:

```

```

85  format: EIGENSTRAT
86  genoFile: my_project.geno
87  snpFile: my_project.snp
88  indFile: my_project.ind
89  jannoFile: my_project.janno
90  bibFile: sources.bib

```

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

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

```

97 trident list -d /path/to/poseidon/packages/modern \
98   -d /path/to/poseidon/packages/ReferenceGenomes
99   -d ~/my_project --packages

```

100 0.1.1.1 General notes

101 **0.1.1.1.1 Logging and command line output** For all subcommands the general argument `--logMode`
 102 defines how trident reports messages (to stderr) on the command line:

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

109 0.1.1.1.2 Duplicates

- 110 • If multiple packages in a package repository share the same **title**, then trident will try to select the one
 111 with the highest version number. If this is not sufficient to resolve the conflict, trident will stop.
- 112 • Individual/sample names (**Poseidon_IDs**) within one package have to be unique, or trident will stop.
- 113 • We generally also discourage ID duplicates across packages in package repositories, but trident will generally
 114 continue with them after printing a warning. This does not apply for **validate**, by default (you can
 115 change this behaviour with `--ignoreDuplicates`), and **forge**. **forge** offers a special mechanism to resolve
 116 duplicates within its selection language (see below).

117 **0.1.1.1.3 Group names in .fam files** The `.fam` file of Plink-formatted genotype data is used inconsistently
 118 across different popular aDNA software tools to store group/population name information. The (global) option
 119 `--inPlinkPopName` with the arguments `asFamily` (default), `asPhenotype` and `asBoth` allows to control the
 120 reading of the population name from Plink `.fam` files. The subcommands that write genotype data (**forge**,
 121 **genoconvert**) have a corresponding option `--outPlinkPopName` to specify this for the output.

122 **0.1.1.1.4 Whitespaces in the .janno file** While reading the `.janno` file `trident` trims all leading and
 123 trailing whitespaces around individual cells. Also all instances of the No-Break Space unicode character will be

124 removed. This means these whitespaces will not be preserved when a package is **forged**.

125 0.1.2 Package creation and manipulation commands

126 **0.1.2.1 Init command** `init` creates a new, valid Poseidon package from genotype data files. It adds a valid
127 `POSEIDON.yml` file, a dummy `.janno` file for context information and an empty `.bib` file for literature references.

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

```
129 Usage: trident init ((-p|--genoOne ARG) | --inFormat ARG --genoFile ARG
130                  --snpFile ARG --indFile ARG) [--snpSet ARG]
131                  (-o|--outPackagePath ARG) [-n|--outPackageName ARG]
132                  [--minimal]
```

133 Create a new Poseidon package from genotype data

135 Available options:

136	<code>-h,--help</code>	Show this help text
137	<code>-p,--genoOne ARG</code>	one of the input genotype data files. Expects <code>.bed</code> or 138 <code>.bim</code> or <code>.fam</code> for PLINK and <code>.geno</code> or <code>.snp</code> or <code>.ind</code> for 139 EIGENSTRAT. The other files must be in the same 140 directory and must have the same base name
141	<code>--inFormat ARG</code>	the format of the input genotype data: EIGENSTRAT or 142 PLINK (only necessary for data input with <code>--genoFile</code> 143 + <code>--snpFile</code> + <code>--indFile</code>)
144	<code>--genoFile ARG</code>	the input geno file path
145	<code>--snpFile ARG</code>	the input snp file path
146	<code>--indFile ARG</code>	the input ind file path
147	<code>--snpSet ARG</code>	the snpSet of the package: 1240K, HumanOrigins or 148 Other. (only relevant for data input with 149 <code>-p --genoOne</code> or <code>--genoFile</code> + <code>--snpFile</code> + <code>--indFile</code> , 150 because the packages in a <code>-d --baseDir</code> already have 151 this information in their respective <code>POSEIDON.yml</code> 152 files) Default: Other
153	<code>-o,--outPackagePath ARG</code>	the output package directory path
154	<code>-n,--outPackageName ARG</code>	the output package name - this is optional: If no 155 name is provided, then the package name defaults to 156 the basename of the (mandatory) <code>--outPackagePath</code> 157 argument
158	<code>--minimal</code>	should only a minimal output package be created?

159 The command

```
160 trident init \  
161   --inFormat EIGENSTRAT/PLINK \  
162   --genoFile path/to/geno_file \  
163   --snpFile path/to/snp_file \  
164   --indFile path/to/ind_file \  
165   --snpSet 1240K|HumanOrigins|Other \  

```

166 -o path/to/new_package_name

167 requires the format (`--inFormat`) of your input data (either EIGENSTRAT or PLINK), the paths to the respective
168 files (`--genoFile`, `--snpFile`, `--indFile`), and optionally the “shape” of these files (`--snpSet`), so if they cover
169 the 1240K, the HumanOrigins or an Other SNP set. A simpler interface added in trident 0.29.0 is available with
170 `-p (+ --snpSet)`.

	EIGENSTRAT	PLINK
genoFile	.geno	.bed
snpFile	.snp	.bim
indFile	.ind	.fam

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

174 **0.1.2.2 Fetch command** `fetch` allows to download Poseidon packages from a remote Poseidon server. Read
175 more about this repository [here](#).

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

177 Usage: trident fetch (-d|--baseDir DIR)
178 (--downloadAll |
179 (--fetchFile ARG | (-f|--fetchString ARG)))
180 [--remoteURL ARG] [-u|--upgrade]

181 Download data from a remote Poseidon repository

182
183 Available options:

184 -h,--help	Show this help text
185 -d,--baseDir DIR	a base directory to search for Poseidon Packages 186 (could be a Poseidon repository)
187 --downloadAll	download all packages the server is offering
188 --fetchFile ARG	A file with a list of packages. Works just as -f, but 189 multiple values can also be separated by newline, not 190 just by comma. -f and --fetchFile can be combined.
191 -f,--fetchString ARG	List of packages to be downloaded from the remote 192 server. Package names should be wrapped in asterisks: 193 *package_title*. You can combine multiple values with 194 comma, so for example: "*package_1*, *package_2*, 195 *package_3*". <code>fetchString</code> uses the same parser as 196 <code>forgeString</code> , but does not allow excludes. If groups 197 or individuals are specified, then packages which 198 include these groups or individuals are included in 199 the download.
200 --remoteURL ARG	URL of the remote Poseidon server 201 (default: "https://c107-224.cloud.gwdg.de")
202 -u,--upgrade	overwrite outdated local package versions

203 It works with

```
204 trident fetch -d ... -d ... \  
205 -f "*package_title_1*,*package_title_2*,*package_title_3*,group_name,<Individual1>"
```

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

209 Entities are specified using a special syntax (see also the documentation of `forge` below): Package titles are
210 wrapped in asterisks: *package_title*, group names are spelled as is, and individual names are wrapped in angular
211 brackets, like `<Individual1>`. Fetch will figure out which packages need to be downloaded to include all specified
212 entities. `--downloadAll`, which can be given instead of `-f` and `--fetchFile`, causes fetch to download all
213 packages from the server. The downloaded packages are added in the first (!) `-d` directory (which gets created
214 if it doesn't exist), but downloads are only performed if the respective packages are not already present in an
215 up-to-date version in any of the `-d` dirs.

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

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

220 To overwrite outdated package versions with `fetch`, the `-u/--upgrade` flag has to be set. Note that many file
221 systems do not offer a way to recover overwritten files. So be careful with this switch.

222 **0.1.2.3 Forge command** `forge` creates new Poseidon packages by extracting and merging packages,
223 populations and individuals from your Poseidon repositories.

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

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

235
236 Available options:

237 -h,--help	Show this help text
238 -d,--baseDir DIR	a base directory to search for Poseidon Packages 239 (could be a Poseidon repository)
240 -p,--genoOne ARG	one of the input genotype data files. Expects .bed or 241 .bim or .fam for PLINK and .geno or .snp or .ind for 242 EIGENSTRAT. The other files must be in the same 243 directory and must have the same base name

244 --inFormat ARG the format of the input genotype data: EIGENSTRAT or
245 PLINK (only necessary for data input with --genoFile
246 + --snpFile + --indFile)
247 --genoFile ARG the input geno file path
248 --snpFile ARG the input snp file path
249 --indFile ARG the input ind file path
250 --snpSet ARG the snpSet of the package: 1240K, HumanOrigins or
251 Other. (only relevant for data input with
252 -p|--genoOne or --genoFile + --snpFile + --indFile,
253 because the packages in a -d|--baseDir already have
254 this information in their respective POSEIDON.yml
255 files) Default: Other
256 --forgeFile ARG A file with a list of packages, groups or individual
257 samples. Works just as -f, but multiple values can
258 also be separated by newline, not just by comma.
259 Empty lines are ignored and comments start with "#",
260 so everything after "#" is ignored in one line.
261 Multiple instances of -f and --forgeFile can be
262 given. They will be evaluated according to their
263 input order on the command line.
264 -f,--forgeString ARG List of packages, groups or individual samples to be
265 combined in the output package. Packages follow the
266 syntax *package_title*, populations/groups are simply
267 group_id and individuals <individual_id>. You can
268 combine multiple values with comma, so for example:
269 "*package_1*, <individual_1>, <individual_2>,"
270 group_1". Duplicates are treated as one entry.
271 Negative selection is possible by prepending "-" to
272 the entity you want to exclude (e.g. "*package_1*,
273 -<individual_1>, -group_1"). forge will apply
274 excludes and includes in order. If the first entity
275 is negative, then forge will assume you want to merge
276 all individuals in the packages found in the baseDirs
277 (except the ones explicitly excluded) before the
278 exclude entities are applied. An empty forgeString
279 (and no --forgeFile) will therefore merge all
280 available individuals. If there are individuals in
281 your input packages with equal individual id, but
282 different main group or source package, they can be
283 specified with the special syntax
284 "<package:group:individual>".
285 --selectSnps ARG To extract specific SNPs during this forge operation,
286 provide a Snp file. Can be either Eigenstrat (file
287 ending must be '.snp') or Plink (file ending must be
288 '.bim'). 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 option '--intersect' is also set, only the SNPs overlapping between the SNP file and the forged packages are output.

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

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

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

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

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

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

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

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

forge can be used with

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


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

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

```
trident forge \
  -d 2017_GonzalesFortesCurrentBiology \
  -p 2018_VeeramahPNAS/2018_VeeramahPNAS.fam \
  --inFormat PLINK \
  --genoFile 2017_HaberAJHG/2017_HaberAJHG.bed \
  --snpFile 2017_HaberAJHG/2017_HaberAJHG.bim \
  --indFile 2017_HaberAJHG/2017_HaberAJHG.fam \
  -f "<STR241.SG>,<ERS1790729.SG>,Iberia_HG.SG" \
  -o testpackage \
  --outFormat EIGENSTRAT \
  --onlyGeno
```

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

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

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

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

```
# Packages
*package1*, *package2*

# Groups and individuals from other packages beyond package1 and package2
group1, <individual1>, group2, <individual2>, <individual3>
```

```

376 # group2 has two outlier individuals that should be ignored
377 -<bad_individual1> # This one has very low coverage
378 -<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.

0.1.2.3.2 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

Poseidon_ID	Group_Name	Genetic_Sex	AdditionalColumn1	AdditionalColumn2	AdditionalColumn3
XXX012	POP2	F	B	E	n/a
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

0.1.2.3.3 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.

0.1.2.3.4 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 `snpSets` 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>	Output 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.

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

429 **0.1.2.4 Genoconvert command** genoconvert converts the genotype data in a Poseidon package to a
430 different file format. The respective entries in the POSEIDON.yml file are changed accordingly.

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

432 Usage: trident genoconvert ((-d|--baseDir DIR) |
433 (-p|--genoOne ARG) | --inFormat ARG --genoFile ARG
434 --snpFile ARG --indFile ARG) [--snpSet ARG])
435 --outFormat ARG [--onlyGeno]
436 [-o|--outPackagePath ARG] [--removeOld]
437 [--outPlinkPopName ARG]

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

439
440 Available options:

441 -h,--help	Show this help text
442 -d,--baseDir DIR	a base directory to search for Poseidon Packages 443 (could be a Poseidon repository)
444 -p,--genoOne ARG	one of the input genotype data files. Expects .bed or 445 .bim or .fam for PLINK and .geno or .snp or .ind for 446 EIGENSTRAT. The other files must be in the same 447 directory and must have the same base name
448 --inFormat ARG	the format of the input genotype data: EIGENSTRAT or 449 PLINK (only necessary for data input with --genoFile 450 + --snpFile + --indFile)
451 --genoFile ARG	the input geno file path
452 --snpFile ARG	the input snp file path
453 --indFile ARG	the input ind file path
454 --snpSet ARG	the snpSet of the package: 1240K, HumanOrigins or 455 Other. (only relevant for data input with 456 -p --genoOne or --genoFile + --snpFile + --indFile, 457 because the packages in a -d --baseDir already have 458 this information in their respective POSEIDON.yml 459 files) Default: Other
460 --outFormat ARG	the format of the output genotype data: EIGENSTRAT or 461 PLINK.
462 --onlyGeno	should only the resulting genotype data be returned? 463 This means the output will not be a Poseidon package
464 -o,--outPackagePath ARG	the output package directory path - this is optional: 465 If no path is provided, then the output is written to 466 the directories where the input genotype data file 467 (.bed/.geno) is stored
468 --removeOld	Remove the old genotype files when creating the new 469 ones
470 --outPlinkPopName ARG	Where to write the population/group name into the FAM 471 file in Plink-format. Three options are possible: 472 asFamily (default) asPhenotype asBoth. See also 473 --inPlinkPopName.

474 With the default setting

```
475 trident genoconvert -d ... -d ... --outFormat EIGENSTRAT|PLINK
```

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

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

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

```
484 trident genoconvert \  
485   -p 2018_Mittnik_Baltic/Mittnik_Baltic.bed \  
486   --outFormat EIGENSTRAT  
487   -o my_directory
```

488 **0.1.2.5 Update command** `update` automatically harmonizes `POSEIDON.yml` files of one or multiple
489 packages if the packages were changed. This is not an automatic update from one Poseidon version to the next!

490 Click here for command line details

```
491 Usage: trident update (-d|--baseDir DIR) [--poseidonVersion ARG]  
492               [--ignorePoseidonVersion] [--versionComponent ARG]  
493               [--noChecksumUpdate] [--newContributors ARG]  
494               [--logText ARG] [--force]
```

495 Update `POSEIDON.yml` files automatically

496
497 Available options:

498	<code>-h,--help</code>	Show this help text
499	<code>-d,--baseDir DIR</code>	a base directory to search for Poseidon Packages 500 (could be a Poseidon repository)
501	<code>--poseidonVersion ARG</code>	Poseidon version the packages should be updated to: 502 e.g. "2.5.3" (default: Nothing)
503	<code>--ignorePoseidonVersion</code>	Read packages even if their <code>poseidonVersion</code> is not 504 compatible with the <code>trident</code> version. The assumption 505 is, that the package is already structurally adjusted 506 to the <code>trident</code> version and only the version number is 507 lagging behind.
508	<code>--versionComponent ARG</code>	Part of the package version number in the 509 <code>POSEIDON.yml</code> file that should be updated: Major, 510 Minor or Patch (see https://semver.org) 511 (default: Patch)
512	<code>--noChecksumUpdate</code>	Should update of checksums in the <code>POSEIDON.yml</code> file 513 be skipped
514	<code>--ignoreGeno</code>	ignore SNP and <code>GenoFile</code>
515	<code>--newContributors ARG</code>	Contributors to add to the <code>POSEIDON.yml</code> file in the

```

516         form "[Firstname Lastname](Email address);..."
517 --logText ARG      Log text for this version jump in the CHANGELOG file
518                     (default: "not specified")
519 --force            Normally the POSEIDON.yml files are only changed if
520                     the poseidonVersion is adjusted or any of the
521                     checksums change. With --force a package version
522                     update can be triggered even if this is not the case.

```

523 It can be called with a lot of optional arguments

```

524 trident update -d ... -d ... \
525   --poseidonVersion "X.X.X" \
526   --versionComponent Major/Minor/Patch \
527   --noChecksumUpdate
528   --ignoreGeno
529   --newContributors "[Firstname Lastname](Email address);..."
530   --logText "short description of the update"
531   --force

```

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

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

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

- 542 • If `--poseidonVersion` is different from the `poseidonVersion` field in the package, then that will be
543 updated.
- 544 • The `packageVersion` will be incremented. If `--versionComponent` is not set, then it falls back to `Patch`,
545 so a change in the last position of the three digit version number. `Minor` increments the middle, and `Major`
546 the first position (see [semantic versioning](#)).
- 547 • The `lastModified` field will be updated to the current day (based on your computer's system time).
- 548 • The contributors in `--newContributors` will be added to the `contributor` field if they're not there already.
- 549 • If any checksums changed, then they will be updated. If certain checksums are not set yet, then they will
550 be added. The checksum update can be skipped with `--noChecksumUpdate` or partially skipped for the
551 genotype data with `--ignoreGeno`.
- 552 • The CHANGELOG.md file will be updated with a new row for the new version and the text in `--logText`
553 (default: "not specified"), which will be appended as the first line of the file. If no CHANGELOG.md file
554 exists, then it will be created and referenced in the POSEIDON.yml file.

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

558 0.1.3 Inspection commands

559 **0.1.3.1 List command** `list` lists packages, groups and individuals of the datasets you use, or of the
560 packages available on the server.

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

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

565 List packages, groups or individuals from local or remote Poseidon
566 repositories

567 Available options:

```
568 -h,--help           Show this help text
569 -d,--baseDir DIR    a base directory to search for Poseidon Packages
570                     (could be a Poseidon repository)
571 --remote            list packages from a remote server instead the local
572                     file system
573 --remoteURL ARG     URL of the remote Poseidon server
574                     (default: "https://c107-224.cloud.gwdg.de")
575 --packages          list all packages
576 --groups            list all groups, ignoring any group names after the
577                     first as specified in the Janno-file
578 --individuals       list individuals
579 -j,--jannoColumn JANNO_HEADER
580                     list additional fields from the janno files, using
581                     the Janno column heading name, such as Country, Site,
582                     Date_C14_Uncal_BP, Endogenous, ...
583 --raw              output table as tsv without header. Useful for piping
584                     into grep or awk
585 --ignoreGeno        ignore SNP and GenoFile
```

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

```
588 trident list -d ... -d ... --packages
```

589 This will yield a table like this

```
590 .----- .----- .-----
591 |           Title           |    Date    | Nr Individuals |
592 :===== :===== :=====
593 | 2015_1000Genomes_1240K_haploid_pulldown | 2020-08-10 | 2535          |
594 | 2016_Mallick_SGDP1240K_diploid_pulldown | 2020-08-10 | 280           |
595 | 2018_BostonDatashare_modern_published   | 2020-08-10 | 2772          |
596 | ...                                     | ...         |               |
597 '-----'-----'-----'
```

598 so a nicely formatted table of all packages, their last update and the number of individuals in it.

599 To view packages on the remote server, instead of using directories to specify the locations of repositories on

your system, you can use `--remote` to show packages on the remote server. For example

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

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

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

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

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

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

[Click here for command line details](#)

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

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

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>--raw</code>	output table as tsv without header. Useful for piping into <code>grep</code> or <code>awk</code>

You can run it with

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

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

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

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

[Click here for command line details](#)

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

Survey the degree of context information completeness for Poseidon packages

Available options:


```

638  -h,--help          Show this help text
639  -d,--baseDir DIR   a base directory to search for Poseidon Packages
640                     (could be a Poseidon repository)
641  --raw              output table as tsv without header. Useful for piping
642                     into grep or awk

```

Running

```

644  trident survey -d ... -d ...

```

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

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

0.1.3.4 Validate command `validate` checks poseidon datasets for structural correctness.

Click here for command line details

```

650  Usage: trident validate (-d|--baseDir DIR)
651         Check one or multiple Poseidon packages for structural correctness
652
653  Available options:
654  -h,--help          Show this help text
655  -d,--baseDir DIR   a base directory to search for Poseidon Packages
656                     (could be a Poseidon repository)
657  --ignoreGeno        ignore SNP and GenoFile
658  --fullGeno          test parsing of all SNPs (by default only the first
659                     100 SNPs are probed)
660  --noExitCode        do not produce an explicit exit code
661  --ignoreDuplicates  do not stop on duplicated individual names in the
662                     package collection

```

You can run it with

```

664  trident validate -d ... -d ...

```

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

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

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

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

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