cube2segy(1)

Name

cube2segy - convert Cube data to SEG-Y format

Synopsis

> cube2segy --project=*FILE* --shot-gather=*FFIDs* | --receiver-gather=*CHANNELs*
>
> [-v|--verbose] [--include-pattern=*PATTERN*]... [--index-cache=*FILE*]
> [--timing-control=*ALGORITHM*] [--fringe-samples=*MODE*]
> [--output-dir=*DIRECTORY* [--force-overwrite] [--force-concat]]
> [--segy-format=*FORMAT*] [--trace-length=*DURATION*]
> [--trace-offset=*SHIFT*] [--reduction-velocity=*VELOCITY*]
> *file* | *directory*...

> cube2segy [-h|--help] [--version] [--sysinfo]

Description

The GIPPtool utility **cube2segy** gathers seismic data from one or more recorders and integrate the traces into seismic sections. It supports the creation of shot as well as receiver gathers. You can switch between the two gather modes by specifying **--shot-gather** or **--receiver-gather** at the command line. The option **--segy-format** is used to further specify the output file format (SEG-Y, Seismic Unix, ...).

To convert the recorded data into seismic sections, the program must know about the setup of the experiment. This is done by providing a "project file" (command line option **--project**) containing information about the location of every shot and receiver point in the experiment as well as the times when the receivers were actually recording, respectively when the sources were triggered.

After parsing the project file, the program will next index the recorded seismic data contained in the files given at the command line. This step is necessary so that the program later knows, which of the input files contain the traces required for writing the seismic section. If a *directory* is given at the command line, the program searches recursively for input files inside that directory. The search can be shortened to contain only files matching a pattern given by the **--include-pattern** option. If you plan to use this program repeatedly on the same dataset, you might also consider saving the index to a file in between program runs (option **--index-cache**).

After all the required information is available, the utility will begin to prepare (internal) lists of time windows describing what data belongs into the respective shot/receiver gather. This is usually done by taking the trigger time of the seismic source and then looking-up, which receivers were

actually recording at that time. However, you can further influence the calculation of relevant time windows by applying a reduction velocity, changing the trace length and/or setting a trace offset. (See **--reduction-velocity**, **--trace-length** and **--trace-offset** options.)

Finally, the information from the internal shot/receiver gather list are used to look-up the respective input files in the previously created index of the available seismic data. The program then reads the necessary data snippets from the input files, converts them to SEG-Y or Seismic Unix format and writes the resulting seismic sections to the current working directory or in a separate directory (use option **--output-dir**).

Options

The program pretty much follows expected Unix command line syntax. Some of the command line options have two variants, one long and an additional short one (for convenience). These are shown below, separated by commas. However, most options only have a long variant. The '=' for options that take a parameter is required and can not be replaced by a whitespace.

-h, --help

Print a brief summary of all available command line options and exit.

--version

Print the **cube2segy** release information and exit.

--sysinfo

Provide some basic system information and exit.

-v, --verbose

This option increases the amount of information given to the user during the program execution. By default (i.e. without this option) the GIPPtool utilities only report warnings and errors. (See also the diagnostics section below.)

--include-pattern=*PATTERN*

Only read data from files whose filename matches the given *PATTERN*. Files with a name not matching the search *PATTERN* will be ignored. This option is quite useful to speed up recursive searches through large subdirectory trees and can be used more than once in the same command line.

You can use the two wild card characters (*, ?) when specifying a PATTERN (e.g. *.pri?). Or alternatively, you can also use a predefined filter called GIPP that can be used exclude all files not following the usual GIPP naming convention for Cube files.

The given search *PATTERN* is only applied to the filename part and not to the full pathname of a file.

--index-cache=*FILE*

Enable caching of the (internal) input file index. Before this program can write seismic sections it first must know which input file contains the necessary traces. For that purpose an index of all

available input files is build first. However, since the content of all input files must be scanned for the index, this can be a quiet time consuming process. To avoid lengthy, repeated scans every time the program is started, the **--index-cache** option can be used to cache the index of all input files in a separate *FILE*.

It is the responsibility of the user to ensure that the cached index is up-to-date and (still) corresponds to the files and directories given as input at the command line. If in doubt, simply delete the cache manually and let the utility generate and save a new cache file during the next program run.

If the cache *FILE* already exist the index will be read from it. If the file does not exist (yet), the index will be written to it after scanning the input files. Already existing cache files will never be overwritten! By default (i.e. without this option) caching of the index is disabled.

--timing-control=***ALGORITHM***

Cube data loggers keep track of the time by tagging selected sample values with precise time information. These (time) tagged samples are the foundation of the overall timing accuracy of the recording. To ensure a high precision it is essential to verify the integrity and premium of the recorded time tags. Use this option to select one of the following quality control algorithms:

LLS

Compute a "local least squares" (LLS) approximation to detect outliers and other dubious time information.

The algorithm will determine the timing quality from the squared residual error ("misfit") of an individual time tag compared to a fitted line through the respective surrounding time tags. Any unexpected large misfit is a good indicator for the presence of a "bad" time tag (e.g. an outlier). All suspicious time tags are excluded from further processing.

This is the default timing quality control algorithm.

RULE

Do a rule-based evaluation of the time tags. The rules are predefined and hard-coded into the program. They were determined by trial and error.

NONE

Skip quality control altogether! This will use any available timing information without further qualification.

FAKE

This "quality control" algorithm will completely overwrite any time information recorded in a Cube trace with a made-up fake time. (All trace start times are set to 1970-01-01 00:00.) Obviously, this will completely screw up the timing information! Use it at your own risk.

Using the *FAKE* time algorithm will only succeed if the **--fringe-samples**=**NOMINAL** command line option is used as well.

The main advantage of the *LLS* algorithm is its flexibility. It was designed to adapt to different

situations and to handle different time keeping hardware as well. The *RULE* based algorithm is faster and much simpler. However, the fixed rule set only works effectively for anticipated situations and is limited to the current build-in and well-known GPS hardware. Future Cube generations e.g. will probably require an updated set of rules to reliably detect bad time tags due to different time keeping hardware. The *NONE* "algorithm" basically disables any timing quality control. It should only be used if you can trust all time tags unconditionally (or do not care). Finally, the *FAKE* time algorithm is intended for worst case scenarios only, where a user absolutely must recover a Cube data stream that cannot be processed normally due to total lack of (recorded) timing information. By adding a fake time the Cube file(s) becomes "processable" again, although at the price of a completely made-up time information.

In addition to the above listed algorithms, recorded time tags are also screened for overall data integrity (range check, checksum) and completeness. Also, there a certain hardware limitation common to all recorders of the Cube family that occasionally cause individual time tags to be discarded. This is done transparently in the background and before any of the above algorithms are applied. This cannot be influenced by the user!

--fringe-samples=*MODE*

Cube recordings basically consist of a continuous stream of sample (amplitude) values, where occasional a single sample is additionally time stamped with the precise time of its recording (taken from GPS). This command line options determines how to treat samples that were recorded before the first GPS time fix or after the last GPS time fix taken by the Cube unit. Determining the precise recording time of these "fringe samples" is problematic because without a second time tag on the other side of the sample, the precise sampling rate inside that segment cannot be computed. Valid options are:

SKIP

Simply exclude all samples without good time control from the conversion. (Default)

NOMINAL

Include fringe samples assuming a perfect nominal sample rate (e.g. 50 Hz, 100 Hz, 200 Hz, etc.; as configured in the Cube recorder setup).

CONSTANT

Include fringe samples assuming a constant (linear) clock drift over the whole recording. The clock drift is calculated from the very first and last available GPS fix in the recording.

Usually, a Cube recording contains only about one second of data before the very first GPS time fix occurs. At the end of recording, the time without GPS fix depends on the recorder configuration. (GPS running continuous or in cycled mode? How long is the cycle?) So, unless you power down and pick up the Cube unit immediately after the recording there should be no problem to just skip and ignore all fringe samples, which is the default behavior.

The situation is different however, when the Cube is deployed in locations without (reliable) GPS reception, e.g. in water or underground in a tunnel. Especially, if the Cube runs out of power before it can obtain a last GPS fix. Here it might become important to include any recorded sample despite the lack of good (GPS) time control. For these cases the *NOMINAL* and *CONSTANT* mode are intended.

--output-dir=*DIRECTORY*

Save the resulting seismic sections to this *DIRECTORY*. The directory must already exist and be writable! Already existing files in that directory will not be overwritten unless the option **--force -overwrite** is used as well.

--force-overwrite

If this option is used, already existing files in the output directory will be overwritten without mercy!

The default behavior however is **not** to overwrite already existing files. Instead a new file is created with an additional number in between filename and extension.

--force-concat

Use this option to concatenate all resulting shot/receiver gathers into a single file. This can be useful if you plan to import the gathers into other software packages for further processing. (You only need to import one single file instead of many different files.)

By default however, a new output file is created for every single gather created.

--segy-format=*FORMAT*

+

Select one of the following predefined output formats:

SEGY

Standard SEG-Y revision 1 (default).

SUOLD

Seismic Unix (old) native binary format.

SUXDR

Seismic Unix platform independent XDR format.

--project=***FILE***

Use this mandatory option to indicate the file describing the experiment setup. For a detailed description of the file format see the project file section below.

Sometimes, this file is also called the "master" or the "geometry" file.

--shot-gather=***FFIDs***

The resulting seismic sections are organized as shot gather. If no *FFID* range is specified, the program will try to write a seismic section for every FFID found in the projects (geometry) configuration file (see project file section below).

Alternatively, you can also specify a comma separated list of single FFIDs or ranges of FFIDs. Use two dots between the first and last FFID to specify a range. Please note that FFID lists and ranges must not contain any space characters!

Example: To produce seismic sections of the shots with FFID 1, 4, 5 and 6 you could use --shot-gather=1,4..6.

--receiver-gather=*CHANNELS*

The resulting seismic sections are organized as receiver gather. If no *CHANNEL* range is given, the program will try to write a seismic section for every channel found in the projects (geometry) configuration file (see project file section below).

Alternatively, you can also specify a comma separated list of single channels or ranges of channels. Use two dots between the first and last channel to specify a range. Please note that channel lists and ranges must not contain any space characters!

Example: To produce seismic sections for receivers with the channel ids 10, 11, 24, 25 and 26 you would use --receiver-gather=10,11,24..26.

--trace-length=*DURATION*

Length of the traces in the resulting seismic section. The *DURATION* is given in seconds. Fractions of seconds will be rounded to microsecond accuracy. If there are not enough samples in the input, the trace in the seismic section will be padded. The default trace length is one minute.

Example: Use --trace-length=120 to obtain two minute long seismic sections consisting of 12000 samples per trace (assuming input data recorded at 100 Hz). To produce 12001 samples per trace you would use --trace-length=120.01 as command line option.

--trace-offset=*SHIFT*

Use this option to shift the start time of the traces in the seismic section relative to the trigger time of the shot (as read from the project file). The *SHIFT* is given in seconds. If no time offset is given, the program will default to begin the trace as close as possible to shot time.

Example: To start the seismic section 2 seconds before the shot time use --trace-offset=-2. (Note the minus sign! No spaces!)

--reduction-velocity=*VELOCITY*

Add a time delay proportional to the distance between source and receiver to every trace in the seismic section. This factor, more commonly known as reduction velocity, is given in meters per second. By default no reduction velocity is applied.

The distance between source and receiver point is calculated using the coordinates in the project file (see project file section below). Obviously, applying a reduction velocity must fail If the project file only contains dummy / place holder coordinates!

Project file

A project file plays an important role when building shot or receiver gathers as it contains a description of experiment setup. Chiefly, this are geographic locations as well as time information.

Project files are simple text files where every (non-comment) line represents one source or receiver point of the experiment. The general syntax rules are:

1. Everything from a # character up to the end of line is considered to be a comment (and will be ignored by the program).
2. All empty lines are ignored as well.
3. Any sequence of space characters or tab-stops in a line containing (any) text will be interpreted as column separator! The use of spaces inside (column) strings is not supported.

The number and content of the different columns varies. Lines describing seismic sources ("shots") will e.g. contain the location and trigger time of the source. Receiver lines, however, describe where and when the recorders were operating. The following listing is an example describing three blasts (the "seismic sources") carried out during an experiment in South Africa.

```
# ------------------------------------------------------------
# name     lat/lon/elev     ffid      shot time          optional
# ------------------------------------------------------------

S s21  -33.1968 22.0695 579  1  2005-11-17T06:05:01.170  7.5
S s32  -33.1882 22.0644 566  2  2005-11-17T06:36:29.593  5.0 10
S s41  -33.1767 22.0592 540  3  2005-11-17T07:12:36.225  7.5
```

In detail the columns of source point lines have the following meaning:

Column #1

Every source point line **must** start with the character S in the first text column. (The software uses this as indicator to distinguish Source point lines from lines describing receiver points.) Capitalization does not matter.

Column #2

The second "name" column contains an arbitrary text string that makes it easier for humans to work with this file. You can place a description of the source point here ("at_yellow_house"), mileage along the profile, a stake number or anything else you think might be helpful.

This column is only used for user feedback by the GIPPtools software and its content will not appear in the resulting seismic section. You can also just use the same dummy string for each source point line. However, the column must exist! Otherwise the software will get out of sync and try to interpret the following longitude column as latitude, elevation values as longitudes, etc.)

Column #3, #4, and #5

The next three columns define the location of the source point (latitude, longitude and elevation in that order). Latitude and longitude should be given in decimal degrees. Latitudes south of the equator are negative as well as longitudes west of Greenwich. Elevation should be given in meters. If you dont have the coordinates of your source points (yet) use some dummy values (like 0.0). The coordinates given here are entered into the SEG-Y trace header.

The coordinates are also used to calculate the absolute (i.e. non-negative) distance between source and receiver, which is required when applying a seismic reduction velocity to the seismic section (option **--reduction-velocity**).

Column #6

The sixth column is the Field File IDentification (FFID). Every source point must have a unique (positive integer) FFID assigned to it. The FFID will be entered into the resulting SEG-Y trace header. Usually, seismic processing software will use this number to identify the recorded traces.

Column #7

The trigger time of the seismic source goes into the seventh column. It consists of date and time information given in ISO-8601 format (example: 2005-11-17T16:05:01.170). All programs of the GIPPtools package resolve time down to microseconds.

Use '*T*' or '_' (underscore) to concatenate date and time. If you use a space character instead, the time information will be interpreted as the next (optional) column.

Column #8, #9, ...

The date/time information is followed by a variable number of optional columns. Unlike the previous columns no place holder / dummy entry is needed if no value is available!

The intended use for these columns is to transport arbitrary, additional information that may be required later by further processing steps into the resulting seismic section (e.g. "amount of explosives used" or "water depth"). If optional columns are used, the value given is always stored as a 4 byte IEEE-754 floating point number. The value of the first optional column is entered into the 240 byte long SEG-Y trace header at its end (bytes #237 to #240). A following second optional value is placed right before the value of the first column (bytes #232 to #236). The third optional again is placed before the second (at #228 to #231) and so on.

If you use to many optional values (there is no hard limit build into the software) you will begin to overwrite important fields in the SEG-Y trace header.

Unlike the variable length source point lines, receiver point lines always contain ten values (columns) describing the equipment used during the measurement, when they were recording data and where they were located while doing so. The following listing again is an example.

```
# ------------------------------------------------------------
# name      lat/lon/elev   chan recorder     start      stop
# ------------------------------------------------------------

R rp1  -33.2133 22.0783 598  1  e3168 p0  2005-11-13 2005-11-19
R rp2  -33.2125 22.0780 598  2  e3168 p1  2005-11-13 2005-11-19
R rp3  -33.2116 22.0777 597  3  e3168 p2  2005-11-14 2005-11-19
R rp4  -33.2110 22.0775 597  4  e3185 p0  2005-11-14 2005-11-20
R rp5  -33.2102 22.0773 595  5  e3185 p1  2005-11-14 2005-11-20
R rp6  -33.2093 22.0769 596  6  e3185 p2  2005-11-14 2005-11-20
R rp7  -33.2083 22.0765 594  7  e3130 p0  2005-11-15 2005-11-20
R rp8  -33.2074 22.0763 594  8  e3130 p1  2005-11-15 2005-11-20
R rp9  -33.2065 22.0760 593  9  e3130 p2  2005-11-15 2005-11-21
```

In detail the columns of receiver point lines have the following meaning:

Column #1

Every receiver point line **must** start with the character R in the first column. (The software uses this as indicator to distinguish Receiver point lines from lines describing source points.) Capitalization does not matter.

Column #2

The second name column is an arbitrary text string that makes it easier for humans to work with this file. You can place a description of the receiver point here ("close_to_big_tree"), mileage along the profile, a stake number or anything else you think might be helpful.

This column is only used for user feedback by the GIPPtools software and its content will not appear in the resulting seismic section. You can also just use the same dummy string for each receiver point line. However, the column must exist! Otherwise the software will get out of sync and try to interpret the following longitude column as latitude, elevation values as longitudes, etc.)

Column #3, #4, and #5

The next three columns define the location of the receiver point (latitude, longitude and elevation in that order). Latitude and longitude should be given in decimal degrees. Latitudes south of the equator are negative as well as longitudes west of Greenwich. Elevation should be given in meters. If you dont have the coordinates of your receiver points (yet) use some dummy values (like 0.0).

The coordinates are also used to calculate the absolute (i.e. non-negative) distance between source and receiver, which is required when applying a seismic reduction velocity to the seismic section (option **--reduction-velocity**).

Column #6

The sixth column is the channel number. Each receiver point in the experiment must have an unique positive integer channel number assigned to it. The channel number will be entered into the resulting SEG-Y trace header. Usually seismic processing software will use this number to identify the recorded traces.

Do not confuse this (experiment/profile unique) channel number with the instrument recording channel (see column #8).

Column #7 and #8

The next two columns are needed to locate the data in recorded files. Column seven contains the recorder unit name used to record the data at. At the GIPP this is usually a five character long string like "c0043" for Cube data logger.

Column eight is used to indicate the recording channel of the respective recording unit. Possible values are p0 to p2 for the (primary) recording channels of a three channel Cube data logger.

If you are unsure about the correct values to enter use the **cubeinfo** utility to inspect your input data. The program can list the "correct" recorder unit id and the recorder channel name.

Column #9 and #10

The last two columns describe the begin and end of the recording. They consist of date and time information given in ISO-8601 format (example: 2005-11-17T16:05:01.170). Depending on your experiment setup it may be enough to give just date information. But if you enter also time of day information here, you should at least specify hour and minutes. (Without time of day information "midnight" of the respective day is assumed.)

Use '*T*' or '_' (underscore) to concatenate date and time. If you use a space character instead, the time information will be interpreted correctly.

Environment

The following environment variables can optionally be used to influence the behavior of the various GIPPtool utilities during startup.

GIPPTOOLS_HOME

This environment variable is used to find the location of the GIPPtools installation directory. In particular, the Java class files that make up the GIPPtools are expected to be in the java subdirectory of **GIPPTOOLS_HOME**.

GIPPTOOLS_JAVA

The utilities of the GIPPtools are written in the programming language Java and consequently need a Java Runtime Environment (JRE) to execute. Use this variable to specify the location of the JRE which should be used.

GIPPTOOLS_OPTS

You can use this environment variable for additional fine-tuning of the Java runtime environment. This is typically used to set the Java heap size available to GIPPtool programs.

GIPPTOOLS_LEAP

The GIPPtools require up-to-date leap second information to correctly interpret Cube files. Usually, this information is obtained from the leap-seconds.list file located in the config subdirectory of the GIPPtools installation directory. This environment variable can be used to provide a more up-to-date leap second list to GIPPtool programs.

It is usually not necessary to define any of those variables as suitable values should be selected automatically. However, if the automatic detection build into the start script fails or you need to choose between different GIPPtool or Java runtime releases installed on your computer, these environment variables might become quite helpful to troubleshoot the situation.

Diagnostics

Occasional, the program will produce user feedback. In general, user messages are classified as *INFO*, *WARNING* or *ERROR*. The *INFO* messages are only displayed when the **--verbose** command line option is used. They usually report about the progress of the program run.

More important are *WARNING* messages. In general, they warn about (possible) problems that may influence the output. Although the program will continue with execution, you certainly should check the results carefully. You might not have gotten what you (thought you) asked for. Finally, *ERROR* messages inform about problems that can not be resolved automatically. Program execution usually stops and the user must fix the problem first.

Exit codes

Use the following exit codes when calling the GIPPtool utility from scripts or other programs to see if finished successfully. Any non-zero code indicates an ERROR.

0

Success.

64

Command line syntax or usage error.

65

Input data error.

66

Input file did not exist or could not be opened.

70

Error in internal program logic.

74

I/O error.

99

Other, unspecified errors.

Examples

1. To prepare seismic sections use:

   ```
   *cube2segy --shot-gather --project=example.project ./data/
   ```

   This will produce a shot gather for every source point defined in the example.project file. The

Cube data will be read from the data directory and the seismic section will be written to the current working directory.

2. Prepare seismic sections for the shots with FFID 2001 and 2002 only.

```
cube2segy --shot-gather=2001,2002 --project=example.project ./data/
```

3. Apply a reduction velocity of 6.5 km/s and shift all traces by half a second towards earlier times.

```
cube2segy --shot-gather --project=example.project --reduction-velocity=6500 --trace
-offset=-0.5 ./data/
```

4. Create seismic section in the new (XDR) Seismic Unix format

```
cube2segy --shot-gather --project=example.project --segy-format=suxdr ./data/
```

Files

$GIPPTOOLS_HOME/bin/cube2segy

The **cube2segy** "program". Usually just a symbolic link pointing to the standard GIPPtools start script.

$GIPPTOOLS_HOME/bin/gipptools

The GIPPtools start script. Almost all utilities of the GIPPtools package are started from this shell script.

See also

gipptools(1), **cube2ascii**(1), **cube2mseed**(1), **cubeevent**(1), **cubeinfo**(1), **mseed2ascii**(1), **mseed2mseed**(1), **mseed2pdas**(1), **mseed2segy**(1), **mseedcut**(1), **mseedinfo**(1), **mseedrecover**(1), **mseedrename**(1)

Bugs and caveats

- The program does not (yet) support Cartesian coordinate systems. Shot and receiver point positions must be given as degrees latitude and degrees longitude.