

Listening Test Conduction Handbook

Roberto Amorim, PITA

May 2005

Version 0.3 – December 2006

Summary

Acknowledgements	3
Introduction	4
The ABC/hr test methodology	5
Steps to conduct a successful listening test.....	7
Test discussion.....	8
Preparing the sample packages	9
Creating the config files and finishing packaging	11
Starting the test	13
Preparing the results	14
Making Plots	16
Finishing at Last.....	17
Opening the Cans of Worms.....	18
High bitrate testing.....	19
VBR versus CBR	20
Dealing with ranked references	22
Hints	23
Annex 1: Model test readme	24

Acknowledgements

I have lots of people to thank for all the experience and knowledge I gathered while conducting listening tests from mid-2003 to mid-2004.

First and foremost, Mr. Darryl Miyaguchi (ff123). If it wasn't for him, I wouldn't have even started conducting tests. He was always there to provide support and guidance whenever a particularly tricky issue surfaced. He also helped me a lot fending off clueless critics and debating with clued-in ones. He's also a very smart guy and a great friend.

Then, the people that helped me with valuable criticism and comments: Juha Laaksonheimo (JohnV), Gian-Carlo Pascutto (Garf), Francis Niechcial (Guruboolez), Darin Morrison (Dibrom), [proxima], tigre and so many others. They helped me make my tests much better and more trustworthy.

Also, the people that supported me with software they developed and hosting of samples: Darryl Miyaguchi, Thomas Schumm (Phong), Benno K. (schnofler), Paul Harris (verloren), Menno Bakker, Darin Morrison and spoon.

Darryl Miyaguchi, Thomas Schumm and Jason Anthony were extremely helpful proof-reading this document.

Last but not least, the listening test participants. They contributed their time and hearing to make my tests meaningful and informative.

To all of you, my heartfelt thanks.

Curitiba, May 2005

Introduction

Blind tests were created as a method to test a product where the tester's psychology could influence the test results. Such a phenomenon is often called "placebo effect". The name comes from medication effectiveness tests (probably the field where blind tests are most widely used), and refers to test participants that display changes in their condition even though they were taking the placebo – usually tablets containing only sugar or flour. In that case, psychology alone is affecting their reaction to the placebo.

A similar phenomenon can happen when the test depends on the auditory system. Subjectivists (people that prefer to rely on subjective opinions) want to believe some piece of audio hardware sounds better than another just because it costs 20 times more (a claim quickly dismissed by objectivists as placebo or "snake oil" until objective proof is produced).

The battle between subjectivists and objectivists is an endless one, and has seen some of its highest points at the rec.audio.opinion Usenet newsgroup. At times, the threads there become ablaze in vitriolic rage as holy wars between the two rival groups take place.

To counter the claims of the subjectivists, the objectivists created a method called ABX to reliably compare two audio signals. In this test, the person taking the test sits in front of a testing device connected to the signal sources A and B. The test consists of several trials, and in each one the signal from one of the sources is routed to the loudspeaker and the other signal is muted. Sources are picked randomly at each trial. If the participant can't consistently differentiate source A from source B, the signals are considered to sound the same to that person and any claim about superior quality in one device is considered to be a result of the placebo effect.

Besides countering placebo, this methodology also is effective against bias. For instance, it's likely that a die-hard Free Software enthusiast will claim Vorbis is better than WMA, no matter in what situation. In a blind environment, he won't be able to let that prejudice influence his test results.

In the late nineties, the International Telecommunications Union (ITU) created a method to compare several audio sources at once with attention to relative quality among sources, and not only the ability to detect differences among them. That method is called ABC/hr, and is the core of audio codec listening tests.

The ABC/hr test methodology

The International Telecommunications Union created the ABC/hr (ABC/hidden reference) methodology as a solution to test their own telecommunication codecs easily and reliably while avoiding interferences related to placebo. It is officially called “*Methods For The Subjective Assessment Of Small Impairments In Audio Systems Including Multichannel Sound Systems*” and is referenced as document ITU-R BS.1116-1¹.

Here is how a test is assembled:

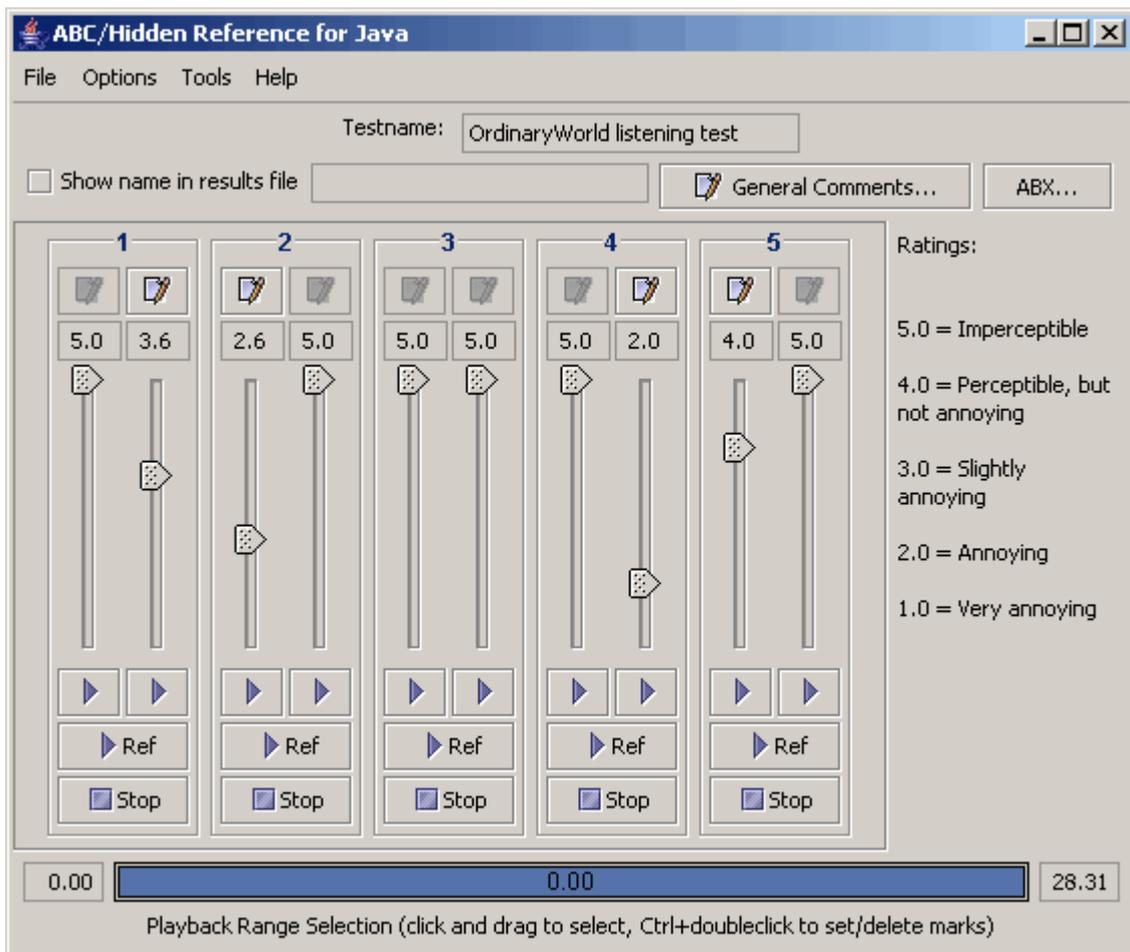


Fig. 1: Example of a test going on in ABC/hr for Java

1. The test conductor prepares the samples beforehand by encoding them with the encoders being tested and decoding said samples back to .wav
2. The samples and the uncompressed reference are loaded into the comparer program. Advanced ABC/hr comparers like ABC/hr for Java can automatically detect and correct offset and gain differences among samples
3. The program randomizes the order of the samples to be compared so that it's pretty much impossible to know what sample is being tested at a given time. It then creates sliders for each sample. The sliders come in left-right pairs. One of them is the uncompressed reference and the other is the

¹ <http://www.itu.int/rec/R-REC-BS.1116-1-199710-I/e>

sample to be tested. The position of the reference and the sample is also randomized

4. The test starts. By clicking the play button under a slider, the test participant listens to the stream represented by that slider. Clicking the “Ref” play button, he/she can listen to the reference in that test without being told which slider is actually the reference
5. The bar at the bottom is used to select a range of audio data that the participant deems particularly interesting for that test. When a range is selected, the other parts of the sample aren't played.
6. Once he/she has listened to the samples, the listener moves the sliders he/she believes to be the samples attributing scores to the feature being tested (usually absolute quality). He/she can click the notepad on top of the moved slider to comment on that sample
7. Having finished the test, the listener saves his/her results and reads them, or sends them to the test administrator in case they were encrypted to avoid tampering.

One of the biggest strengths of this test is the fact that the reference is hidden next to the sample. That way, clowns that took the test and just moved sliders randomly will have a big chance of being caught for ranking the samples several times. Of course, not all people ranking references are necessarily trying to mess your test, or deaf, as will be discussed later on.

The ABC/hr method can be used to test not only audio codecs, but also video codecs, audio hardware, and several other products. This document will focus on audio codec tests alone, since audio hardware tests are very difficult to set up properly and the software tools to set up an audio codec test are easily available, which doesn't apply to video codec testing.

So you got sold and want to conduct your very own public listening test, but you have doubts how to proceed?

I'm here to help.

Steps to conduct a successful listening test

In this section, I'll try to guide you in the steps to conduct a listening test, from the early discussion to the publication of the results.

First and foremost, find a place to discuss your test. You should be open to accept criticism and suggestions, otherwise people won't be very interested in your test.

Probably the most friendly place to conduct such discussion is the Hydrogenaudio forum (<http://www.hydrogenaudio.org>), with what is probably the largest population of objectivists on the Internet, and an eventual subjectivist that serves as laughing stock.

Test discussion

Once you settle with a forum, start the discussion explaining your plans: what codecs you plan to test, at what bitrate (bits per second, AKA kbps), using which settings.

You can start the discussion with a collection of samples in mind (in that case list them when you start talking about your test since they should be open to discussion as well) or you can ask other forum members for suggestions on good samples.

A good amount of samples varies between 12 (less than that, and your final results won't be statistically significant) and 18 (more than that, and your participants will be too spread among the samples). If you are very sure you'll have lots of participants (will hardly happen), you can go over 20 samples. The more samples you use, the most statistically significant the final score will be.

Still on the samples subject: avoid the obvious choice of problem samples (samples that trip codecs producing very nasty artifacts) with sharp attacks or similar audio structures because their artifacts are easily detectable, and therefore less fatiguing for your listeners. The reason is that such samples are not representative of the whole scope of music. You'll inevitably have to feature some pop, rock, jazz, classical, etc. otherwise your test results will be meaningless to everyone but the people that only listen to such hard-to-encode music.

Each sample should be as homogenous as reasonably possible, otherwise the listener may have difficulty rating a codec (e.g., in the first part codec A was better, but in the last part, codec B was better).

This is a list of things that should be decided during the test discussion:

- Before you even start the test discussion, decide what it WILL be testing: portable formats, high quality formats, popular formats, formats on sale at online stores, formats for dial-up streaming... That will ease the choice of codecs immensely.
- Codecs to be tested, bitrate and settings
- Amount of samples, and samples to be tested.
- Test duration (at least 10 days, and no more than a month)
- CBR or VBR (check the Can of Worms section for a lengthy discussion on this)
- How results with ranked reference will be treated (again, look at the Can of Worms)
- If the test warrants encryption (requires ABC/hr Java) or not

Preparing the sample packages

Once all the test details have been ironed out, it's time to start the hard work. Announce what you have decided for your test. That will be a sort of "soft-freeze" – you'll only change those decisions for very good reasons. Then pick up the samples you chose, and encode them using the selected encoders with the selected settings at the selected bitrates. Formats that provide an open source command line decoder (MP3, Vorbis, AAC/HE AAC and Musepack) can be distributed as-is. This saves a lot of bandwidth when distributing the sample packages. Formats with no open source command line decoder (WMA, ATRAC3, MP3pro and Real Audio) must be pre-decoded by you and losslessly compressed with FLAC or WavPack.

Once all samples have been processed, create a folder structure for them in order to avoid a mess. This is the structure I use:

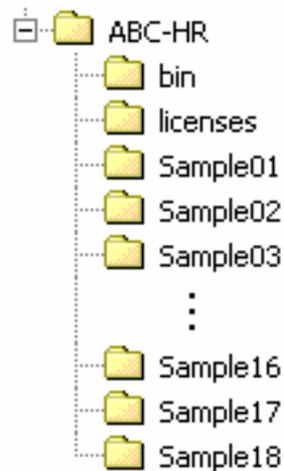


fig. 2: Folder structure

abchr.jar, the readmes and test configuration files go in the root (ABC-HR) folder. `.\bin` contains the apps to decode and decompress the samples, as well as batch files to make the operation as painless as possible at the tester's side. `.\licenses` is the place for legalese documents that should be distributed with ABC/hr, and `.\SampleXX` is where the encoded files and reference related to each sample are located.

I recommend you name the reference file "samplefile.wav", and the encoded files "samplefile1.wav", "samplefile2.wav", etc. It will make things easier later on.

Once the files have been distributed among the SampleXX folders, zip each one of these folders from the parent directory, to make sure the relative path is included inside the .zip. This makes things easier to test participants. You should end up with several SampleXX.zip files, and when one of these files is decompressed at a folder, it creates a subfolder `./SampleXX` containing the encoded files and the reference.

Then, work on the batch files. Create a starting .bat in the .\bin folder where each encoded file in the folder ..\Sample01 is processed with its respective decoder/decompressor (which should also be located at the bin folder) so that in the end of the processing you have only the .wavs that will be loaded into ABC/hr.

Once you have one batch file tested and ready, create the other files from that one, only caring about replacing the file name and folder location.

With all the batch files done, run them all. This is to test them, and also to create the WAVs that will be used in the next step – creating the config files for ABC/hr Java.

Creating the config files and finishing packaging

ABC/hr-Java² makes everything easy for you. Creating config files is as simple as 1-2-3.

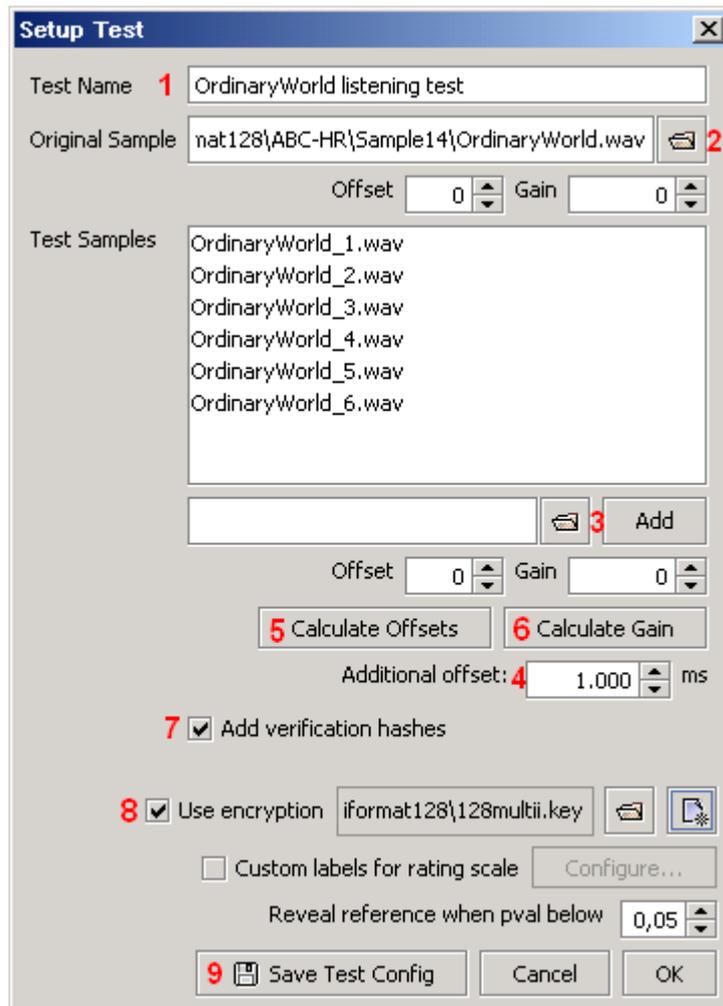


fig. 3: Test setup screen

1. Define a name for your listening test
2. Select the reference in the “Original Sample” box
3. Select the samples to be tested in the “Test samples” box
4. Add 1000 ms to the “Additional offset” box (explanation later)
5. Hit the “Calculate Offsets” button
6. Hit the “Calculate Gain” button
7. Select “Add verification hashes” for further security
8. If you want to use encryption, mark the checkbox and either create a new encryption key, or select one you already created
9. Save the test config

² <http://www.rarewares.org/others.html>

If you want, you can also use “Custom labels for rating scale” to use labels other than “5.0 = Imperceptible” to “1.0 = Very annoying”. With most test cases, it’s unnecessary.

The “Additional offset” setting increases the playback start point in each sample. As explained by Gabriel Bouvigne, LAME developer:

“This is not because of problems in codecs, but to allow a realistic behavior. With real encoding, encoders might be adapting themselves to content. In real tracks, this adaptation can be progressively done as the track is starting, thus the beginning of a extract from a track is not representative of the encoding of this part inside the full length sample.”

Create a config file for each sample set following the above sequence and save it to the root folder next to abchr.jar.

If your test doesn’t require encryption, you can save your test configs in .txt format, so that they can be used in ff123’s ABC/hr comparer. Keep in mind though that it won’t apply gain correction.

Once you finish creating the config files (and test them one by one!), create a readme file for your test. It should contain simple instructions on how to take the test, links to the sample packages, the expected deadline, information about anonymity and information on how to contact you.

After you finish the readme, zip together: abchr.jar and related .jars, the config files, the readme, the .bin folder including all .bat files and decoding/decompressing binaries, and the licenses folder to make lawyers happy. Upload this package and the samples packages, and get ready to start your test...

Starting the test

First, create a web page inviting people to participate in your test. Explain the motivations that led you to create this test, the codecs and settings being tested, and when the test is expected to end. Link to your starter packages (the .zips containing abchr.jar, the readme, the configs, etc.).

After you double-check everything to make sure it's working (it might be a good idea to upload everything one day before the test officially starts and ask a few friends to take the test, following the steps you laid out. This helps spot flaws, if any), prepare an announcement text to be copy-pasted at forums, news sites, and so on. Here is a list of places you should consider announcing your test at:

- Hydrogenaudio³, of course
- rec.audio.opinion⁴ Usenet group (only if you have thick skin)
- Doom9⁵ forum
- audiocoding.com⁶
- rarewares.org⁷
- Listening test mailing list⁸ (discuss this with the author beforehand)
- Slashdot⁹ (but disregard the clueless replies. People there once wrote I should test FLAC's quality because it is free and open)

With the test announced, sit back and wait for people's results to start flowing in (and pray that you didn't commit any horrible mistake that will force you to call the test off while it is going on. It happened once to me).

³ <http://www.hydrogenaudio.org>

⁴ <http://groups.google.com/group/rec.audio.opinion>

⁵ <http://forum.doom9.org>

⁶ <http://www.audiocoding.com/modules/newbb/>

⁷ <http://www.rarewares.org>

⁸ <http://www.rjamorim.com/test/newsletter.html>

⁹ <http://slashdot.org>

Preparing the results

As you start approaching the deadline, gather the results you already received and see if they are enough to come up with statistically valid rankings (no less than 10 results per sample). If they aren't enough, you can consider either postponing the deadline or calling it a day. Hopefully, if you did a good job convincing people that participating in your test will help improve World peace and find a cure for AIDS, that won't be a problem, so let's see what to do with all those results.

Start with sorting them by sample. Create a folder for each sample set, and place the respective results in there.

Several people name their result files simply "resultXX.erf". To avoid file name conflicts, use ABC-HR Java's result processor which automatically detects if the test participant used the "Show name in results file" setting and uses it for file naming. If he didn't use it, you should name the file something like "anonXX-resultXX.xml" to clarify that this tester preferred to remain anonymous.

This is indeed a very boring process, and you'll sooner or later find ways to make it easier and slightly more automatic. Hopefully sooner than later...

With the test finished and all result files in their respective places, go for the next step: putting the results together, while screening for ranked references.

In my first tests, this was a very painful process that involved using a hackish program in each single folder to create tables with the results for each participant arranged in lines, and codecs sorted in columns. Some results had to be screened individually to make sure the listener did or didn't rank references. All in all, it took a few hours.

Now, it takes about 10 seconds, thanks to Mr. Thomas Schumm (Phong).

He created a breathtaking little Python script called Chunky¹⁰, that goes through all your results subfolders gathering data, screening them for ranked references and successful or unsuccessful ABX tests, and outputs nice tables that are ready to be fed into the statistical analyzer. After you finish running Chunky, you can already know with some confidence who won and who lost the test. From now on, it's only eye-candy and precision.

To run Chunky, create a text file (codecs.txt) linking each codec to the number used in the filenames, so that it looks like:

¹⁰ <http://www.phong.org/chunky/>

```
1, Vorbis
2, AAC
3, Lame
...
```

Don't use any spaces in the codec names, or it will break the statistical analyzer later on.

Then, run Chunky from the command prompt with a line similar to:

```
chunky.exe --codec-file="path\to\codecs.txt" -n --
ratings=results --warn -p 0.05
```

in the parent folder where the results are sorted in subfolders.

If everything goes well, you'll see on the command prompt a list of results with ranked references that didn't reach a minimum ABX pval, and in the directory where you ran chunky from there'll be several text files "resultsXX.txt" (that directory MUST be empty before you run chunky, except for the subfolders that should ONLY contain the decrypted results). If you look into results.txt, the last two lines should look like:

```
% Codec averages:
% 4.59      4.46      4.18      4.26      3.73      3.98
```

Those are your final results. The only thing missing from them is the error margin, that will be calculated now.

To calculate error margins, you must use ff123's statistical analysis tool¹¹ from the command prompt. Run it as:

```
friedman -tp resultsXX.txt
```

and it'll print to screen the analysis done on that results table. If you want Friedman to save the analysis to a file, use output redirection:

```
friedman -tp resultsXX.txt > analysisXX.txt
```

Run the analysis tool for each result file. When you have all done – including the final analysis on results.txt – move on to the next step: creating the plots.

¹¹ <http://ff123.net/friedman/stats.html>

Making Plots

I don't think there is much point explaining step-by-step how to create charts in Excel/OpenOffice Calc/Grapher/etc., specially since I'm not even a good teacher at that. You're better off researching the documentation that comes with each of these tools. Just some generic hints:

- In Microsoft Excel, the plot type used is “Stock – High-Low-Close”, which is called “Stock Chart 1” on OpenOffice 1.x.
- The error margins are the HSD value output by Friendman. Use half HSD for the upper error margin, and half for the lower.
- Don't forget to include codec names, ranking values and amount of participants in the plots.
- It can be a good idea to create a zoomed version of the final plot, focusing only on the score range of the winning codecs.

Finishing at Last

After you get all your plots done, create a web page featuring them alongside comments on who won, who lost and why (if you manage to detect that).

Include information such as what happened during the test, what justified your choices for this or that encoder, how much codecs improved since the last test... the surprises and the let-downs.

Upload everything, alongside with the result files (so that people can reproduce your findings or assemble them in alternative ways) and announce the results at the same places you announced the test itself some days before.

Congratulations! You completed your first listening test. :-)
Now start getting prepared for the next one...

Opening the Cans of Worms

In this section, we discuss at length the most controversial and polemic subjects that *always* surface while a test is being discussed. It will hopefully give you ammunition to fight clueless critics and the like.

High bitrate testing

rj amorim: There's some inverse proportionality there
rj amorim: At low bitrates nobody is interested,
but the results are easy to obtain
rj amorim: At high bitrates everyone is interested,
but you practically can't obtain usable results
ff123: s/bitrates/beauty and s/results/fucks

It's unavoidable: You will have someone asking you to conduct a high bitrate test – something above 160kbps. Usually more than once. The usual argument is that most people (the ones that care about listening tests, of course) use bitrates in the 190-200kbps range (LAME preset standard, Musepack --standard, Vorbis q6...) therefore mid-low bitrates listening tests are meaningless to them.

To these people, I offer this canned response:

Why a listening test at high bitrates (~192kbps) wouldn't work:

- 1) Most samples would have already reached transparency at that bitrate. And choosing only problem samples would make the test less significant since you wouldn't be testing a broad range of musical styles.
- 2) Only a handful of golden ears would be able to reliably ABX the samples, and even after ABXing they would hardly give scores lower than 4.5
- 3) Since most scores would be around 4.5 and higher, the error margins would be big enough to make all codecs tied. To avoid that, you would need to have hundreds of participants, in an attempt to bring the error margins down.
- 4) You would have a hard time finding hundreds of golden ears, and even more so, hundreds of golden ears willing to participate, because the test would be very fatiguing and frustrating due to the difficulty of ABXing.
- 5) At the end, even if you managed to bring the error margins down, the codecs would be ranked so close that you wouldn't be able to produce a decent conclusion. All codecs would seem (or be) tied to each other.

VBR versus CBR

This is another undying issue that will haunt most of your tests. Some people will be polarized asking for VBR, others will beg you for CBR.

The argument used by people asking for VBR is that it's what everybody is (or should be) using, given that it's the most efficient way of distributing data across the audio stream. Easy-to-encode data gets fewer bits, hard-to-encode data gets more bits, and quality ends up being more or less constant.

The CBR pushers, on the other hand, argue that CBR is the most fair method to compare codecs. When you encode tracks in VBR, unavoidably some samples will get more bits than the test target bitrate, others will get less. Since codecs' methods of allocating bits vary amongst themselves, in the final bitrate sum, one codec will end up using higher bitrates than the competition. That's not very fair.

But then again, codecs like Musepack don't offer a CBR mode at all. Vorbis is also known for performing much better in VBR mode than CBR mode. Other codecs, like Real Audio and Atrac3 are CBR-only. And WMA has an unreliable VBR mode that makes bitrates fluctuate insanely among samples. Therefore, why punish codecs that implement a good VBR mode (LAME, Vorbis), forcing them to the lowest-common-denominator of CBR?

So where to settle?

I believe the best compromise is to use VBR whenever possible, and CBR only when needed. To shield yourself from bitrate variation criticism, test the VBR encoder with several quality settings over several CDs of different musical styles. Settle with the setting that comes closer to the target bitrate over the collection of encoded files. Therefore, that would be the setting that comes closer to the bitrate desired in *everyday usage*, even though it might vary in small 30-second samples.

Avoid the temptation of encoding each sample at several VBR qualities until it comes closest to the target bitrate. Even though that would create some nice and fair bitrate distribution, that's completely unrelated to what people do at home – no one encodes his own CDs several times until the bitrates come close to a target. And what's the use of a listening test that bears no semblance to reality?

The samples should be about the same duration. The idea is that the average bitrate of the sample set depends on the individual sample durations as well as their difficulty -- the longer the sample, the more it affects the overall bitrate.

For a VBR codec, the distribution of bitrates in the small sample set (eg.,

20 samples), supposing you draw a histogram of it, should resemble the distribution of a large sample set chosen from a wide variety of music. That should give you more arguments to counter fairness issues.

Dealing with ranked references

The higher the bitrates you test, the more often you'll receive result files with ranked references. By ranked references, I mean that the listener gave a score to the reference file instead of the encoded sample. Obviously, that shouldn't happen (although in some cases it is even justifiable, as we'll see later). What to do with those results?

You have two choices. If you already have plenty of results, you can just delete them and move on, using only the unranked results. But if you are indeed conducting a mid-high bitrate listening test, odds are that you don't have plenty of results. In that case, my suggestion is that you screen these results before discarding them.

You should check them for ABX results. ABC/hr Java saves inside the results files if the listener did an ABX test, how many trials he took, and what was the outcome. If he got a pval smaller than 0.05, that means he could differentiate the reference from a sample with great confidence, and the error in sliding can probably be credited to fatigue or lack of attention. In that case, the score can be safely computed.

Another explanation to the "error" is that the sample really sounded better than the reference. Although that sounds impossible at first analysis, in some cases it might happen. Consider this: you are testing an old recording that has lots of high frequency hissing. The reference file will keep all the annoying hissing there, while a codec will probably lowpass most of it during encoding. In that case, the encoded file will probably sound better than the reference.

On the other hand, if there are no ABX test results, you should just discard the ranked results, as, for all that we know, it might be that the listener just moved sliders randomly.

Hints

*A substantial part of this section
was contributed by ff123*

- **Sample01**

A note on listener psychology: they will tend to download and listen to sample01 first, and then decide whether they want to continue based on their experience on that first sample. Ideally, there would be some sort of randomizer which assigns different music to each of the samples dynamically, but that would require some way to sort things out in the end. Barring that, you should try to make Sample01 as friendly as possible.

- **Ask the devs**

Whenever possible, ask the developers of each codec being tested what settings you should be using for that test, and what codec version (if there is more than one). It'll be an awesome shield against criticism: people won't be able to complain that you used a wrong setting or version, as the developer surely knows better than pretty much anyone out there.

Of course you won't likely be able to get a hold of WMA or ATRAC developers, but there's a very good chance you'll be able to swap some ideas with Vorbis, Nero AAC and LAME developers... if you are interested in conducting a listening test, contact the author of this document so that he can point you to the right people (check the annexes)

- **Keep the ball rolling**

Try to keep the discussion thread going during the test to keep interest up, and to avoid people forgetting about it.

- **Read the documentation**

Study carefully the ABC/hr specification document from ITU. Try to abide to it as much as possible. Some parts are simply undoable on an informal listening test with \$0 budget, like considerations on testing equipment and environment. So focus on the "cheaper" parts. Once you understand enough about the methodology, you'll be able to easily dismiss criticisms to it: few people can claim they know better than the ITU about blind audio tests.

- **Listener training**

It's a good idea to provide some instructions to your prospective participants on how to properly participate in a listening test. One solution is linking to ff123's very complete training page:

<http://ff123.net/64test/practice.html>

Annex 1: Model test readme

<test name> public listening test =readme=

Be welcome, and thank-you very much for taking part in this listening test. Your results will be very valuable.

Here are the instructions:

1. Decompress the ABC-HR_bin.zip package to it's own folder
2. Download one (or more) of the sample packages. Location of the packages is further down.
3. Place the sample package (SampleXX.zip) in the same folder as ABC/hr and uncompress it. Make sure to keep directory formatting intact, or the config files won't work.
For an idea on how the folders should look like (in case you downloaded all the packages), check folder-setup.png
4. If you use windows:
Go to the bin/ directory and run SampleXX.bat (XX being the number of the sample package you want to prepare for testing)
Wait until the prompt screen goes off.
If you use Linux
Same as above, but to run the .bat file, use Wine with the command
\$ wine cmd SampleXX.bat
5. ****You need the Sun Java Runtime v1.4 or superior to run abchr!**
If you don't have Java, download it from here:
<http://java.com/java/download/index.jsp>

****You need to use the ABC/hr java version provided in the abc-hr_bin.zip package together with this readme. Older versions aren't compatible with the config files format.**

If you use Windows:

Double-click abchr.jar

If you use *nix:

From the shell run "java -jar abchr.jar" without quotes

Once ABC/hr is open, do File -> Open test file. Then, load the file configXX.ecf. Again, XX is the number of the test you want to take. Make sure you have run the .bat respective to that package before.

6. Take the test. If you want info on how to properly do that, check ff123's page:

<http://ff123.net/64test/practice.html>

7. After you finish the test, save the results, zip them together and mail the .zip to <test conductor e-mail> . The test ends on <date>. No results will be accepted after that date, unless the test is extended. Eventual extensions will be announced at the test page (<URL to test page>)

You don't need to test all samples to participate. Even one single result is already very helpful. Of course, the more you test, the better for the final results' significance.

All results and comments I receive will be published. If you want to be associated with your results, please enter your (nick)name in the "Show name in results file" field in ABC/hr (check the checkbox next to the text to activate the field). Otherwise, your results will be anonymous.

These are the sample packages:

If you use BitTorrent, downloading through it and helping seed would be of much help to me. A recommended BitTorrent client is Azureus: <http://azureus.sourceforge.net/>
<links to the torrents>

If you prefer standard http downloads:

<links to the sample packages>

The average package size is <size>

Again, thank-you very much;

<conductor's name>