

```
export('data_segmentation_01.mlx', 'data_segmentation_01.pdf');
```

```
%% ==== CONFIG ====
clear; clc;

% Point to your "Data" directory (the one with 5 subfolders)
dataRoot = "/Users/dghifari/02-University/SEM-2-2025/ELEC5305/elec5305-
project-520140154/Data";

% Segmentation
targetFs = 16000;    % standardize SR
clipSec   = 3;       % window length (s)
hopSec    = 3;       % step (s) -> non-overlap; try 1.5 for 50% overlap
padLast   = true;    % zero-pad tail to full window
rmsDb     = -40;     % drop clips below this RMS (dBFS)

% Dynamic per-file caps (limits AFTER silence filtering)
baseCap   = 40;      % neutral cap
minCap    = 20;      % lower bound
maxCap    = 80;      % upper bound
wFam      = 0.6;     % weight: family scarcity
wInstr    = 0.4;     % weight: instrument scarcity
```

```
%% ==== DATASTORE & DURATIONS ====
ads = audioDatastore(dataRoot, "IncludeSubfolders", true, "LabelSource",
"foldernames");
disp(countEachLabel(ads))
```

Label	Count
bass	1
cello	2
clarinet	2
conga	2
cymbal	2
drum	6
flute	3
guitar	3
harpsichord	3
hum	3
organ	2
piano	3
saxophone	4
sing	4
spoken	4
synth	4
tambourine	3

timpani	2
trombone	5
trumpet	5
viola	3
violin	2

```

files = ads.Files;
labs = ads.Labels;
numFiles = numel(files);

dur = nan(numFiles,1);
for i = 1:numFiles
    try
        info = audioinfo(files{i});
        dur(i) = info.Duration;
    catch
        dur(i) = NaN;
    end
end
T = table(files, labs, dur, 'VariableNames', {'File','Label','Duration_s'});
T = sortrows(T, "Duration_s", "descend");
head(T, 8)

```

#### File

```

{'/Users/dghifari/02-University/SEM-2-2025/ELEC5305/elec5305-project-520140154/Data/keyboards/harpsichord/harpsichord.wav', 'harpsichord', 1.5}
{'/Users/dghifari/02-University/SEM-2-2025/ELEC5305/elec5305-project-520140154/Data/winds/flute/winds/flute.wav', 'flute', 1.5}
{'/Users/dghifari/02-University/SEM-2-2025/ELEC5305/elec5305-project-520140154/Data/keyboards/harpsichord/harpsichord.wav', 'harpsichord', 1.5}
{'/Users/dghifari/02-University/SEM-2-2025/ELEC5305/elec5305-project-520140154/Data/voice/hum/voice_hum.wav', 'voice_hum', 1.5}
{'/Users/dghifari/02-University/SEM-2-2025/ELEC5305/elec5305-project-520140154/Data/percussion/drum/percussion_drum.wav', 'percussion_drum', 1.5}
{'/Users/dghifari/02-University/SEM-2-2025/ELEC5305/elec5305-project-520140154/Data/strings/bass/strings_bass.wav', 'strings_bass', 1.5}
{'/Users/dghifari/02-University/SEM-2-2025/ELEC5305/elec5305-project-520140154/Data/winds/clarinet/winds/clarinet.wav', 'winds/clarinet', 1.5}
{'/Users/dghifari/02-University/SEM-2-2025/ELEC5305/elec5305-project-520140154/Data/keyboards/piano/keyboards_piano.wav', 'keyboards_piano', 1.5}

```

```

%% ==== BUILD SCARCITY MAPS FOR DYNAMIC CAPS ====
% --- FAMILY MAP: count *files* per top-level family folder ---
famMap = containers.Map; % key = family name (strings, percussion, winds,
keyboards, voice)
for i = 1:numFiles
    [fam_i, ~] = localPathLabels(files{i}, dataRoot, labs(i));
    famKey = lower(char(fam_i));
    if isKey(famMap, famKey)
        famMap(famKey) = famMap(famKey) + 1;
    else
        famMap(famKey) = 1;
    end
end
medianFam = median(cell2mat(values(famMap)));

% --- INSTRUMENT MAP: count *files* per <family>_<instrument> pair ---
instrMap = containers.Map; % key = 'family_instrument'

```

```

for i = 1:numFiles
    [fam_i, instr_i] = localPathLabels(files{i}, dataRoot, labs(i));
    instrKey = char(strcat(string(fam_i), "_", string(instr_i)));
    if isKey(instrMap, instrKey)
        instrMap(instrKey) = instrMap(instrKey) + 1;
    else
        instrMap(instrKey) = 1;
    end
end
medianInstr = median(cell2mat(values(instrMap)));

disp("Family file counts:");

```

Family file counts:

```

for k = keys(famMap)
    fprintf("  %-10s : %d files\n", k{1}, famMap(k{1}));
end

```

```

keyboards   : 12 files
percussion  : 15 files
strings     : 11 files
voice       : 11 files
winds       : 19 files

```

```

%% Helper to compute per-file cap safely
getCap = @(fam,instr) localCap( ...
    famMap, instrMap, ...
    char(lower(string(fam))), ...
    char(lower(string(fam) + "_" + string(instr))), ...
    baseCap, minCap, maxCap, wFam, wInstr, medianFam, medianInstr);

```

```

%% ==== FULL PASS: ESTIMATE CLIPS WITH DYNAMIC CAPS ====
disp("Analyzing total estimated clips per file (dynamic caps)...");

```

Analyzing total estimated clips per file (dynamic caps)...

```

familyCount      = containers.Map; % fam -> total clips
instrumentCount = containers.Map; % 'fam_instr' -> total clips

for i = 1:numFiles
    fpath = files{i};

    % Read & SR
    try
        [x, fs] = audioread(fpath);
    catch
        warning("Could not read: %s", fpath);
    end
end

```

```

        continue;
    end
    if size(x,2) > 1, x = mean(x,2); end
    if fs ~= targetFs, x = resample(x, targetFs, fs); end

    % Segment + silence filter
    try
        clips = segmentAudio(x, targetFs, clipSec, hopSec, padLast);
        if ~isempty(clips)
            keep = cellfun(@(c) rmsMask(c, rmsDb), clips);
            clips = clips(keep);
        end
    catch
        warning("Segmentation failed: %s", fpath);
        clips = {};
    end

    % Derive fam/instr from path (robust, relative to dataRoot)
    [fam, instr] = localPathLabels(fpath, dataRoot, labs(i));

    % Apply dynamic cap
    capThis = getCap(fam, instr);
    if numel(clips) > capThis, clips = clips(1:capThis); end
    nClips = numel(clips);

    % Accumulate family totals
    famKey = char(fam);
    if isKey(familyCount, famKey), familyCount(famKey) =
familyCount(famKey) + nClips;
    else, familyCount(famKey) = nClips; end

    % Accumulate instrument totals
    instrKey = char(strcat(string(fam), "_", string(instr)));
    if isKey(instrumentCount, instrKey), instrumentCount(instrKey) =
instrumentCount(instrKey) + nClips;
    else, instrumentCount(instrKey) = nClips; end
end

```

```

%% ==== REPORT ====
disp("----- total clips per family (dynamic) -----");

```

```

----- total clips per family (dynamic) -----

```

```

kF = sort(string(familyCount.keys));
for ii = 1:numel(kF)
    fprintf("%s : %d clips\n", kF(ii), familyCount(char(kF(ii))));
end

```

```
keyboards : 214 clips
percussion : 138 clips
strings : 142 clips
voice : 170 clips
winds : 187 clips
```

```
disp("----- total clips per instrument (dynamic) -----");
```

```
----- total clips per instrument (dynamic) -----
```

```
kI = sort(string(instrumentCount.keys));
for ii = 1:numel(kI)
    fprintf("%s : %d clips\n", kI(ii), instrumentCount(char(kI(ii))));
end
```

```
keyboards_harpsichord : 99 clips
keyboards_organ : 38 clips
keyboards_piano : 61 clips
keyboards_synth : 16 clips
percussion_conga : 9 clips
percussion_cymbal : 6 clips
percussion_drum : 105 clips
percussion_tambourine : 4 clips
percussion_timpani : 14 clips
strings_bass : 50 clips
strings_cello : 22 clips
strings_guitar : 30 clips
strings_viola : 17 clips
strings_violin : 23 clips
voice_hum : 52 clips
voice_sing : 78 clips
voice_spoken : 40 clips
winds_clarinet : 41 clips
winds_flute : 54 clips
winds_saxophone : 22 clips
winds_trombone : 26 clips
winds_trumpet : 44 clips
```

```
%% ==== LOCAL FUNCTIONS ====
function clips = segmentAudio(x, fs, clipSec, hopSec, padLast)
    if size(x,2) > 1, x = mean(x,2); end
    clipSamp = max(1, round(clipSec*fs));
    hopSamp = max(1, round(hopSec*fs));
    if padLast
        % pad so the last hop lands cleanly
        remSamp = mod(max(0, numel(x) - clipSamp), hopSamp);
        if remSamp ~= 0
            x = [x; zeros(hopSamp - remSamp, 1)];
        end
    end
    starts = 1:hopSamp:max(1, numel(x)-clipSamp+1);
    clips = cell(numel(starts),1);
```

```

    for j = 1:numel(starts)
        s = starts(j); e = s + clipSamp - 1;
        if e > numel(x), e = numel(x); s = max(1, e-clipSamp+1); end
        seg = x(s:e);
        if numel(seg) < clipSamp, seg(end+1:clipSamp) = 0; end
        clips{j} = seg;
    end
end

function keep = rmsMask(clip, threshDb)
    r = sqrt(mean(clip.^2) + 1e-12);
    keep = (20*log10(r + 1e-12) >= threshDb);
end

function [fam, instr] = localPathLabels(fpath, dataRoot, labelFromFolder)
    % Prefer path relative to dataRoot: Data/<family>/<instrument>/...
    fam = lower(char(labelFromFolder));
    instr = "unknown";
    try
        root = char(dataRoot);
        p = char(fpath);
        if strncmpi(p, root, length(root))
            rel = strrep(p(length(root)+2:end), '\\', '/'); % +2 to skip
filesep
            parts = split(string(rel), "/");
            if numel(parts) >= 1 && strlength(parts(1))>0
                fam = lower(char(parts(1)));
            end
            if numel(parts) >= 2 && strlength(parts(2))>0
                instr = lower(char(parts(2)));
            end
        end
    catch
        % fall back to datastore label only
    end
end

function v = safeGetMap(m, keyChar, defVal)
    if isKey(m, keyChar), v = m(keyChar); else, v = defVal; end
end

function cap = localCap(famMap, instrMap, famKey, instrKey, ...
    baseCap, minCap, maxCap, wFam, wInstr, medianFam,
    medianInstr)
    famCount = safeGetMap(famMap, famKey, medianFam);
    instrCount = safeGetMap(instrMap, instrKey, medianInstr);
    scarcity = wFam*(medianFam/max(1,famCount)) + wInstr*(medianInstr/
    max(1,instrCount));
    cap = max(minCap, min(maxCap, round(baseCap * scarcity)));
end

```

```
%% ==== CONFIG: OUTPUT & RUN INFO ====
```

```
outRoot = fullfile(fileparts(dataRoot), "Segmented");    % sibling to Data/  
runID    = datestr(now, "yyyymmdd_HHMMSS");  
rng(5305); % reproducible shuffle
```

```
if ~exist(outRoot, 'dir'); mkdir(outRoot); end  
fprintf("Output root: %s\nRun: %s\n\n", outRoot, runID);
```

```
Output root: /Users/dghifari/02-University/SEM-2-2025/ELEC5305/elec5305-project-520140154/Segmented  
Run: 20251012_173319
```

```
%% ==== FIXED FAMILY LIST & INSTRUMENT MAPPING ====
```

```
families = ["strings", "percussion", "winds", "keyboards", "voice"];  
family2id = containers.Map(families, num2cell(1:numel(families)));
```

```
instrs = unique(lower(string(ads.Labels))); % datastore labels =  
instruments
```

```
instrs = sort(instrs);  
instr2id = containers.Map(instrs, num2cell(1:numel(instrs)));
```

```
disp("Fixed family list:")
```

```
Fixed family list:
```

```
disp(families)
```

```
"strings"    "percussion"    "winds"    "keyboards"    "voice"
```

```
disp("Sample instrument labels:")
```

```
Sample instrument labels:
```

```
disp(instrs(1:min(end,10)))
```

```
"bass"  
"cello"  
"clarinet"  
"conga"  
"cymbal"  
"drum"  
"flute"  
"guitar"  
"harpsichord"  
"hum"
```

```
%% ==== EXPORT: SEGMENT, FILTER, CAP, WRITE ====  
fprintf("Exporting segmented clips...\n");
```

Exporting segmented clips...

```
mf_filepath    = strings(0,1);  
mf_family      = strings(0,1);  
mf_instrument = strings(0,1);  
mf_sourcefile  = strings(0,1);  
mf_clipIndex   = zeros(0,1);  
mf_duration    = zeros(0,1);  
mf_rmsDb       = zeros(0,1);  
mf_fs          = zeros(0,1);  
mf_capUsed     = zeros(0,1);  
  
perSourceCounter = containers.Map;  
  
for i = 1:numFiles  
    fpath = files{i};  
  
    % Read & resample  
    try  
        [x, fs] = audioread(fpath);  
    catch  
        warning("Could not read: %s", fpath);  
        continue;  
    end  
    if size(x,2) > 1, x = mean(x,2); end  
    if fs ~= targetFs, x = resample(x, targetFs, fs); end  
  
    % Segment + silence filter  
    try  
        clips = segmentAudio(x, targetFs, clipSec, hopSec, padLast);  
        if ~isempty(clips)  
            keep = cellfun(@(c) rmsMask(c, rmsDb), clips);  
            clips = clips(keep);  
        end  
    catch  
        warning("Segmentation failed: %s", fpath);  
        clips = {};  
    end  
  
    % Derive labels  
    [fam, instr] = localPathLabels(fpath, dataRoot, labs(i));  
    fam = string(lower(fam));  
    instr = string(lower(instr));  
    if ~ismember(fam, families)
```

```

        warning("Unknown family '%s' in %s. Skipping.", fam, fpath);
        continue;
    end
    if ~isKey(instr2id, instr)
        warning("Unknown instrument '%s' in %s. Skipping.", instr, fpath);
        continue;
    end

    % Apply dynamic cap
    capThis = getCap(fam, instr);
    if numel(clips) > capThis, clips = clips(1:capThis); end

    % Prepare destination folder
    dstDir = fullfile(outRoot, char(fam), char(instr));
    if ~exist(dstDir, 'dir'); mkdir(dstDir); end

    % Track clip numbering per source
    [~,srcName,~] = fileparts(fpath);
    if ~isKey(perSourceCounter, srcName), perSourceCounter(srcName) = 0; end

    % Write each clip
    for k = 1:numel(clips)
        c = clips{k};
        perSourceCounter(srcName) = perSourceCounter(srcName) + 1;
        idxStr = sprintf("%03d", perSourceCounter(srcName));
        outName = sprintf("%s_%s_%s_clip_%s.wav", char(fam), char(instr),
srcName, idxStr);
        outPath = fullfile(dstDir, outName);
        audiowrite(outPath, c, targetFs);

        dur_s = numel(c)/targetFs;
        r      = sqrt(mean(c.^2) + 1e-12);
        db     = 20*log10(r + 1e-12);

        mf_filepath(end+1,1) = string(outPath);
        mf_family(end+1,1)   = fam;
        mf_instrument(end+1,1) = instr;
        mf_sourcefile(end+1,1) = string(srcName);
        mf_clipIndex(end+1,1) = perSourceCounter(srcName);
        mf_duration(end+1,1) = dur_s;
        mf_rmsDb(end+1,1)    = db;
        mf_fs(end+1,1)       = targetFs;
        mf_capUsed(end+1,1)  = capThis;
    end
end

fprintf("Export complete. Total clips written: %d\n\n",
numel(mf_clipIndex));

```

Export complete. Total clips written: 851

```

%% ==== MANIFEST: TABLE, IDs, SAVE ====
projRoot = fileparts(dataRoot);
filepath_rel = erase(mf_filepath, projRoot + filesep);

family_id = arrayfun(@(f) family2id(f), mf_family);
instrument_id = arrayfun(@(i) instr2id(i), mf_instrument);

Manifest = table( ...
    filepath_rel, mf_family, mf_instrument, family_id, instrument_id, ...
    mf_sourcefile, mf_clipIndex, mf_duration, mf_rmsDb, mf_fs, mf_capUsed,
    ...
    'VariableNames', ...
    {'filepath_rel','family','instrument','family_id','instrument_id', ...
    'source_file','clip_index','duration_s','rms_db','fs','cap_used'});

manDir = fullfile(projRoot, "Manifests");
if ~exist(manDir,'dir'); mkdir(manDir); end
csvPath = fullfile(manDir, "manifest_v1_" + runID + ".csv");
matPath = fullfile(manDir, "manifest_v1_" + runID + ".mat");
writetable(Manifest, csvPath);
save(matPath, "Manifest", "family2id", "instr2id", "runID", "-v7.3");

fprintf("Manifest saved:\n  %s\n  %s\n\n", csvPath, matPath);

```

```

Manifest saved:
/Users/dghifari/02-University/SEM-2-2025/ELEC5305/elec5305-project-520140154/Manifests/manifest_v1_2025
/Users/dghifari/02-University/SEM-2-2025/ELEC5305/elec5305-project-520140154/Manifests/manifest_v1_2025

```

```

%% ==== QC SUMMARY ====
[grpFam,~,idxFam] = unique(Manifest.family);
countsFam = accumarray(idxFam,1);
meanRMS    = splitapply(@mean, Manifest.rms_db, idxFam);
stdRMS     = splitapply(@std,  Manifest.rms_db, idxFam);

disp("----- Clips per family -----");

```

```

----- Clips per family -----

```

```

for i = 1:numel(grpFam)
    fprintf("%-12s : %5d clips | mean RMS %6.2f dB (± %.2f)\n", grpFam{i},
    countsFam(i), meanRMS(i), stdRMS(i));
end

```

```

keyboards      :    214 clips | mean RMS -27.37 dB (± 5.36)
percussion     :    138 clips | mean RMS -23.05 dB (± 4.55)
strings        :    142 clips | mean RMS -23.57 dB (± 6.40)
voice          :    170 clips | mean RMS -20.80 dB (± 5.43)
winds          :    187 clips | mean RMS -25.73 dB (± 6.07)

```

```
durMean = mean(Manifest.duration_s);  
durStd  = std(Manifest.duration_s);  
fprintf("\nClip duration ~ %.3f s ( $\pm$  %.3f)\n", durMean, durStd);
```

Clip duration ~ 3.000 s ( $\pm$  0.000)

```
quietFrac = mean(Manifest.rms_db < (rmsDb + 3));  
fprintf("Near-threshold quiet clips: %.1f%%\n", 100*quietFrac);
```

Near-threshold quiet clips: 2.0%