Humdrum Rhythm Tools

craig @ ccrma.stanford.edu

19 April 2016

Humdrum program documentation

timebase –h gives one-page summary of *mint* command (similar for all Humdrum Toolkit programs)
beat --options list options for serialize command (same for all Humdrum Extras programs)

Humdrum Toolkit man pages:
http://www.humdrum.org/Humdrum/commands/timebase.html

Humdrum Extras man pages:
http://extras.humdrum.org/man/beat

Chapter 23 in the Humdrum Users' Guide (Rhythm):
http://www.humdrum.org/Humdrum/guide23.html

List of various Humdrum resources:
http://humdrum.ccarh.org

Humdrum Users' Group (**HUG):
https://groups.google.com/forum/?fromgroups#!forum/starstarhug

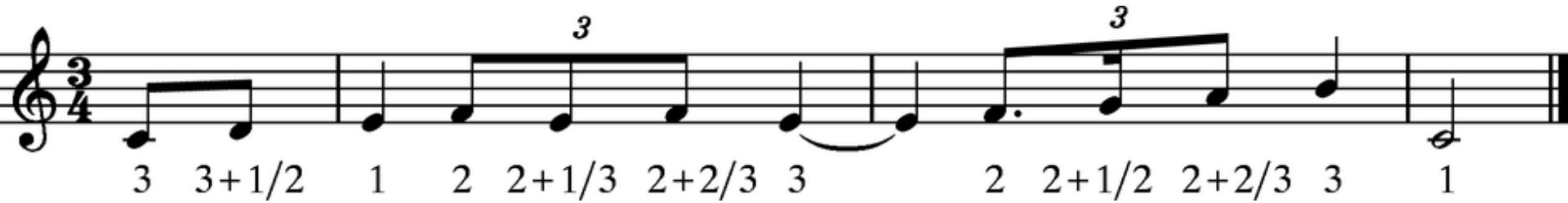
beat

http://extras.humdrum.org/man/beat

	beat	beat -f	beat -c	beat -d
**kern	**beat	**beat	**absb	**dur
*M3/4	*M3/4	*M3/4	*M3/4	*M3/4
*clefG2	*	*	*	*
8cL	3	3	0	0.5
8dJ	3.5	3+1/2	0.5	0.5
=1	=1	=1	=1	=1
4e	1	1	1	1
12fL	2	2	2	0.333333
12e	2.33333	2+1/3	2.33333	0.333333
12fJ	2.66667	2+2/3	2.66667	0.333333
[4e	3	3	3	2
=2	=2	=2	=2	=2
4e]	.	.	4	.
12.fL	2	2	5	0.5
24gk	2.5	2+1/2	5.5	0.166667
12aJ	2.66667	2+2/3	5.66667	0.333333
4b	3	3	6	1
=3	=3	=3	=3	=3
2c	1	1	7	2
==	==	==	==	==
*-	*-	*-	*-	*-

beat

beat –fa input.krn | hum2muse --text beat | muse2ps =z21j \
| convert –density 300 - -trim -resize 50% output.png



Isorhythmic analysis of WTC

beat –d h://wtc/wtc1p01.krn

Michael Schutz,
tempo variability
in isorhythmc vs
non isorhythmic
works

```
**dur
*
*
*M4/4
*MM108
=1-
*
0.25
0.25
0.25
0.25
0.25
0.25
0.25
0.25
0.25
0.25
0.25
0.25
0.25
0.25
0.25
0.25
0.25
0.25
0.25
=2
```

Isorhythm

humsplit h://wtc

Download all WTC preludes and fugues

How many preludes/fugues are isorhythmic?
Are preludes or fugues more commonly isorhythmic?

```
beat –d wtc1p01.krn | ridx –H | uniq –c
    544     0.25
      1      4
```

```
beat –d wtc1f01.krn | ridx –H | uniq –c
      3 0.5
    1 0.75
    2 0.125
    3 0.5
    1 0.75
   11 0.25
    1 0.5
    1 0.25
    2 0.125
    3 0.5
etc.
```

Isorhythm script

```
use strict;
my @files = @ARGV;
print "**file\t**rseqnum\t**unum\t**unum2”;
Print “\t**most\t**score1\t**score2\n";
foreach my $file (@files) {
  analyzeFile($file);
}
print "*-\t*-\t*-\t*-\t*-\n";


exit(0);
```

```
sub analyzeFile {
  my ($file) = @_;

  my $rseq    = `beat -d $file | ridx -H`;
  my @rseqA   = split(/\n/, $rseq);
  my $rseqnum = @rseqA;
  my $rnum    = `beat -d $file | ridx -H | sort | uniq -c | wc -l `;
  chomp $rnum;
  my $useq    = `beat -d $file | ridx -H | uniq -c`;
  my $useqnum = `beat -d $file | ridx -H | uniq -c | wc -l`;
  chomp $useqnum;
  my $useqmax = `beat -d $file | ridx -H | uniq -c | sort -nr | head -n 1`;
  chomp $useqmax;
  $useqmax =~ /(\d+)/;
  $useqmax = $1;
  my $mostcommonrhythm = `beat -d $file | ridx -H | sort -n | uniq -c | sort -nr | head -n 1`;
  chomp $mostcommonrhythm;
  $mostcommonrhythm =~ /(\d+)\s+([.\d]+)/;
  my $mostcount = $1;
  $mostcommonrhythm = $2;

  # score 1: divide maximum length sequence by total rhythms:
  my $score1 = $useqmax / $rseqnum;
  $score1 = int($score1 * 1000.0 + 0.5) / 1000.0;

  # score 2: divide most common rhythm by total rhythms:
  my $score2 = $mostcount / $rseqnum;
  $score2 = int($score2 * 1000.0 + 0.5) / 1000.0;

  print "$file\t$rseqnum\t$rnum\t$useqnum\t$mostcommonrhythm\t$score1\t$score2\n";
}
```

gettime

http://extras.humdrum.org/man/gettime

- Calculate the performance time in seconds/milliseconds for a score.

gettime –T :: calculate the total performance duration of the score(s)

gettime –T h://chopin/preludes/prelude28-01.krn
Total time: 25.5 seconds

gettime –T h://chopin/preludes
Total time: 31:05.83537 minutes

gettime –T h://chopin/mazurkas
Total time: 1:51:39.4903 hours

thrux h://beethoven/sonatas | gettime –T
Total time: 9:23:55.3539 hours

thrux –v norep h://beethoven/sonatas | gettime –T
Total time: 7:44:52.8678 hours

Longest Beethoven sonata

```
humsplit h://beethoven/sonatas
for i in `seq -w 32`
do
    echo Sonata $i `cat sonata$i* | thrux | gettime -T | tail -n 1`;
done | sort -n -k 2 -t :
```

Sonata 09 Total time: 15:33.7273 minutes
Sonata 12 Total time: 17:08.61713 minutes
Sonata 01 Total time: 19:37.2288 minutes
Sonata 15 Total time: 20:17.9069 minutes
Sonata 17 Total time: 20:48.8071 minutes
Sonata 32 Total time: 20:44.6422 minutes
Sonata 11 Total time: 21:55.8766 minutes
Sonata 16 Total time: 21:59.6136 minutes
Sonata 18 Total time: 21:14.15 minutes
Sonata 21 Total time: 21:45.6618 minutes
Sonata 03 Total time: 22:15.5389 minutes
Sonata 07 Total time: 22:30.0455 minutes
Sonata 23 Total time: 23:29.6134 minutes
Sonata 02 Total time: 26:03.92308 minutes
Sonata 04 Total time: 30:50.8052 minutes
Sonata 29 Total time: 42:11.2835 minutes

Sonata 09 Total time: 15:33.7273 minutes
Sonata 12 Total time: 17:08.61713 minutes
Sonata 01 Total time: 19:37.2288 minutes
Sonata 15 Total time: 20:17.9069 minutes
Sonata 17 Total time: 20:48.8071 minutes
Sonata 32 Total time: 20:44.6422 minutes
Sonata 11 Total time: 21:55.8766 minutes
Sonata 16 Total time: 21:59.6136 minutes
Sonata 18 Total time: 21:14.15 minutes
Sonata 21 Total time: 21:45.6618 minutes
Sonata 03 Total time: 22:15.5389 minutes
Sonata 07 Total time: 22:30.0455 minutes
Sonata 23 Total time: 23:29.6134 minutes
Sonata 02 Total time: 26:03.92308 minutes
Sonata 04 Total time: 30:50.8052 minutes
Sonata 29 Total time: 42:11.2835 minutes

minrhy

http://extras.humdrum.org/man/minrhy

- Calculates the minimum rhythmic unit for which all rhythms in the score are integer multiples.

```
**kern
*M3/4
*clefG2
8cL
8dJ
=1
4e
12fL
12e
12fJ
[4e
=2
4e]
12.fL
24gk
12aJ
4b
=3
2c
==
*-
```

minrhy
24

- Smallest rhythm value not necessarily present:

```
**kern
4c
8d
8d
12e
12e
12e
*-
```

24

scordur

Scordur h://chopin/preludes/prelude28-01.krn
34 (quarter notes)

Useful for checking the duration of parts before assembling into a score

timebase

http://www.humdrum.org/Humdrum/commands/timebase.html

- Insert null-tokens lines to make all data lines represent the same duration.
- Use minrhy to calculate the time unit to use with timebase
- Data loss if timebase rhythm is too large
- Grace notes are lots (should be fixed by converting to reversible comments)

```
**kern
2c
8d
4e
8f
4g
*-
```

minrhy

```
8
```

timebase –t 8

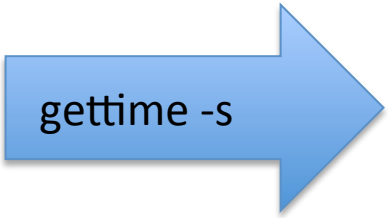
```
**kern
*tb8
2c
.
.
.
8d
4e
.
8f
4g
.
*-
```

timebase –t 16

```
**kern
*tb16
2c
.
.
.
.
.
.
8d
.
4e
.
.
.
8f
.
4g
.
.
.
*-
```

gettime (2)

```
**kern
*M4/4
*MM60
=1-
4c
4d
4e
4f
=2
*MM120
4c
4d
4e
4f
=3
*MM144
4c
4d
4e
4f
==
*
```




```
**time
*u=sec
*M4/4
*MM60
=1-
0
1
2
3
=2
*MM120
4
4.5
5
5.5
=3
*MM144
6
6.41667
6.83333
7.25
==
*
```

gettime –as | gettime -at

```
**kern  **time  **tempo
*       *u=sec  *
*M4/4   *M4/4   *M4/4
*MM60   *MM60   *MM60
=1-     =1-     =1-
4c      0       60
4d      1       60
4e      2       60
4f      3       60
=2      =2      =2
*MM120  *MM120  *MM120
4c      4       120
4d      4.5     120
4e      5       120
4f      5.5     120
=3      =3      =3
*MM144  *MM144  *MM144
4c      6       144
4d      6.41667 144
4e      6.83333 144
4f      7.25    144
==      ==      ==
*       *       *
```

assemble

- Reverses the process of extract command, joining parts into full score

1. Calculate the minimum rhythmic unit (least common multiple of all rhythms in the input to assemble).
2. Apply timebase at that minimum rhythm (warning about grace notes)
3. Run assemble command
4. Remove null-token lines

file1.krn

```
**kern
*M3/4
=1-
4c/
8aL/
8aL/
4c/
==
*-
```

file2.krn

```
**kern
*M3/4
=1-
4cc\
12ccL\
12cc\
12ccJ\
4cc\
==
*-
```

```
minrhy file1.krn file2.krn
file1.krn:      8
file2.krn:     12
all:           24

timebase –t 24 file1.krn > t1.krn
timebase –t 24 file2.krn > t2.krn
assemble t2.krn t1.krn | rid –d > score.krn
```

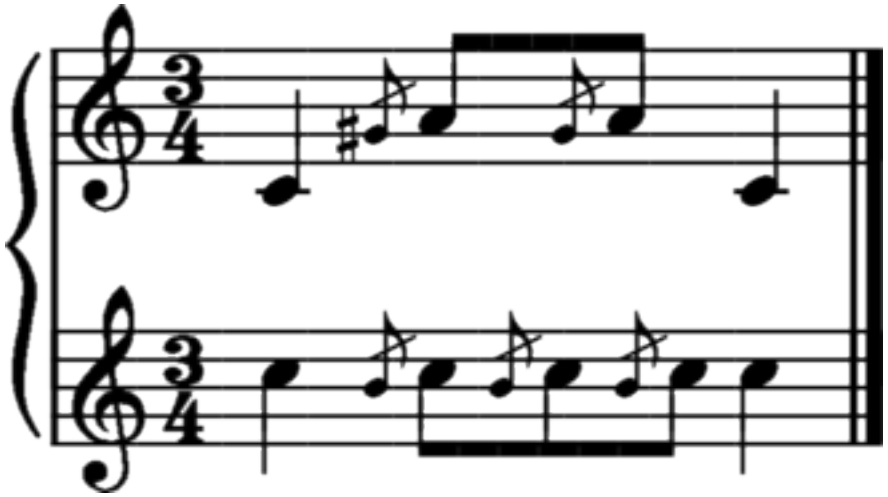
partjoin

- timebase command eats grace notes (**kern notes which contain “q” character)
- partjoin script automates minrhy/timebase/assemble & handles grace notes

```
#!/usr/bin/perl
use strict;
my @files = @ARGV;
die "Usage: $0 file1 file2 ...\n" if @files < 2;
my $filelist = join(" ", @files);
my $minrhy = `minrhy $filelist`;
$minrhy =~ /all:\s*(\d+)/;
$minrhy = $1;
my @assemblefiles;
foreach my $file (@files) {
  my $contents = getContents($file);
  open (PROCESS, "|timebase -t $minrhy > $file-timebase$minrhy") or die;
  print PROCESS $contents;
  close PROCESS;
  $assemblefiles[@assemblefiles] = "$file-timebase$minrhy";
}
my $assemblelist = join(" ", reverse @assemblefiles);
my $result = `assemble $assemblelist | rid -d | grep -v "^\*tb"`;
printResult($result);
`rm -f $assemblelist`;
exit(0);
```

```
sub getContents {
  my ($file) = @_;
  open (FILE, $file) or die;
  my @contents = <FILE>;
  for (my $i=0; $i<@contents; $i++) {
    my $line = $contents[$i];
    next if $line =~ /^!/;
    next if $line =~ /^\*/;
    if ($line =~ /q/) {
      my @fields = split(/\t/, $line);
      for (my $j=0; $j<@fields; $j++) {
        $fields[$j] = "!QqQ$fields[$j]";
      }
      $contents[$i] = join("\t", @fields);
    }}
  return join("", @contents);
}
sub printResult {
  my ($result) = @_;
  my @output = split(/\n/, $result);
  my $line;
  foreach $line (@output) {
   if ($line !~ /!QqQ/) {
     print "$line\n";
     next;
   }
   my @fields = split(/\t/, $line);
   for (my $j=0; $j<@fields; $j++) {
     my $token = $fields[$j];
     if ($token eq "!") {
       print ".";
     } else {
       $token =~ s/^!QqQ//;
       print $token;
     }
     print "\t" if $j < $#fields;
   }
   print "\n";
  }}
```

partjoin (2)



input1.krn

```
**kern
*M3/4
=1-
4c
g#q
8a
g#q
8a
4c
==
*-
```

input2.krn

```
**kern
*M3/4
=1-
4cc\
bq
12ccL\
bq
12cc\
bq
12ccJ\
4cc\
==
*-
```

partjoin input1.krn input2.krn > output.krn

output.krn

```
**kern    **kern
*M3/4     *M3/4
=1-       =1-
4cc\      4c
bq        g#q
12ccL\    8a
bq        .
12cc\     .
.         g#q
.         8a
bq        .
12ccJ\    .
4cc\      4c
==        ==
*-        *-
```

sample

Extra sonorities at selected rhythms

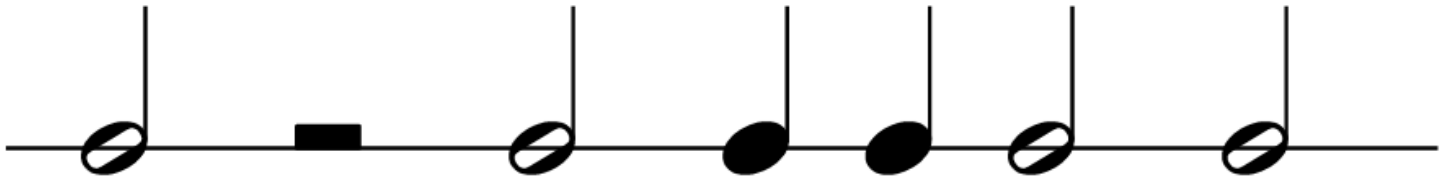
sample h://chorales/chor001.krn

**kern	**kern	**kern	**kern
*k[f#]	*k[f#]	*k[f#]	*k[f#]
*M3/4	*M3/4	*M3/4	*M3/4
4GG	4B	4d	4g
=1	=1	=1	=1
4G	4B	4d	4g
4E	4c	4e	4g
4F#	4A	4d	4dd
=2	=2	=2	=2
4G	4G	4d	4b
4D	4F#	4d	4b
4E	4G	4B	4g
=3	=3	=3	=3
4C	4c	4e	4g
4BB	4c	4e	4g
4GG	4d	4g	4b
=4	=4	=4	=4
4D	4d	4f#	4a
4D	4d	4f#	4a
4GG	4d	4g	4b
=5	=5	=5	=5

Mensuration patterns

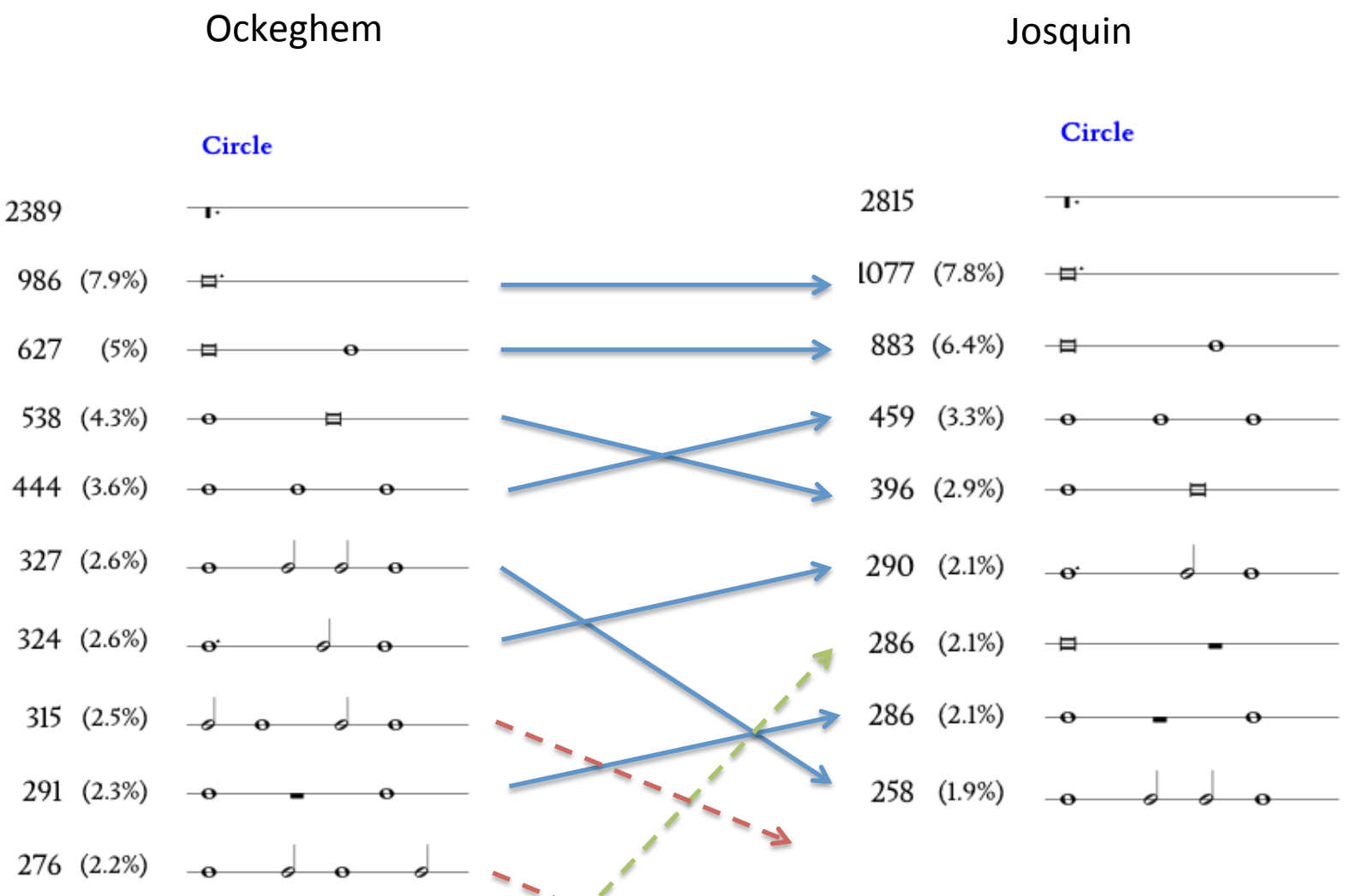
http://josquin.stanford.edu/images/menpat

http://josquin.stanford.edu/images/menpat/m_mr_m_q_q_m_m.svg



JRP Patterns by composer

http://josquin.stanford.edu/analysis/rhythm

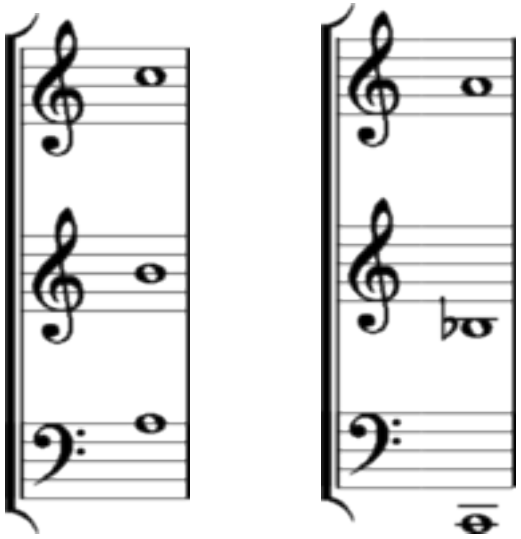


Pattern sequences


```
[
{
        "jrpid"          :       "Jos2721",
        "title"          :       "La Bernardina",
        "genre"          :       "Chanson",
        "vcount"         :       3,
        "voices"         :       ["Superius", "Tenor", "Contra"],
        "features": {
                "rhythm": [
                        ["q_q_q_q_m_m", "tq_q_w_m", "w_mr_m", "tm_m_w", "md_q_m_m", "tq_q_m_w", "w_wr",
"w_m_m", "tq_q_q_q_m_m", "tq_q_q_q_m_m", "tq_q_q_q_w", "md_q_m_m", "md_q_m_m", "md_q_w", "b", "b", "b",
"b", "b", "b", "b", "b", "b", "wr_w", "w_w", "wd_q_q", "w_w", "b", "md_q_q_q_q_q", "m_m_m_q_q",
"q_q_q_q_m_m", "m_q_q_q_q_q_q", "m_m_md_q", "q_q_q_q_m_m", "q_q_q_q_q_q_q_q", "m_m_m_m", "tm_m_m_m",
"md_q_q_q_m", "mr_md_q_m", "m_md_q_q_q", "q_q_m_m_m", "m_m_mr_m", "tb", "tm_m_m_m", "m_m_m_m", "w_w"],
                        ["br", "br", "q_q_q_q_m_m", "tq_q_w_m", "b", "tw_w", "wd_m", "m_m_m_m", "wd_m",
"wd_m", "m_w_m", "b", "wr_w", "md_q_m_m", "b", "br", "wr_w", "md_q_m_m", "b", "br", "wr_w", "md_q_m_m",
"b", "wd_q_q", "w_w", "wd_m", "m_m_w", "wd_m", "b", "tw_w", "tb", "b", "tw_w", "tb", "b", "tw_w",
"w_md_q", "m_md_q_q_q", "w_m_m", "w_md_q", "m_m_w", "m_md_q_q_q", "q_q_q_q_w", "w_m_m", "m_m_m_m",
"w_w"],
                        ["br", "br", "br", "br", "q_q_q_q_m_m", "tq_q_w_m", "wd_q_q", "m_m_m_m",
"tq_q_q_q_m_m", "tq_q_q_q_m_m", "tq_q_q_q_w", "b", "tb", "br", "wr_w", "md_q_m_m", "b", "br", "wr_w",
"md_q_m_m", "b", "br", "wr_w", "md_q_m_m", "w_w", "tm_m_m_m", "w_w", "tm_q_q_w", "mr_md_q_q_q",
"m_m_mr_m", "tq_q_q_q_m_m", "mr_md_q_q_q", "m_m_mr_m", "tq_q_q_q_m_m", "mr_md_q_q_q", "m_m_w", "w_m_m",
"tq_q_q_q_m_m", "tq_q_q_q_m_m", "mr_md_q_q_q", "m_m_m_m", "m_md_q_q_q", "md_q_q_q_m",
"tq_q_q_q_q_q_q_q", "m_w_m", "tm_m_w"]
                        ]
                }
}
]
```

Search for particular sonority

- How many times and where does the sonority “A B C” occur in a repertory?



```
**kern
a b c
a b- c
*-
```

tntype

```
**tnt
3-2B
3-2A
*-
```

ABC sonority

```
tntype -at h://jrp/Jos | ditto -p | hgrep -mbH 3-2  > abc.txt
```

```
os0301a-Missa_Ave_maris_stella-Kyrie.krn:measure 22:beat 7:4E\2d\     (2.F/)  (0r)    3-2BT02
Jos0301a-Missa_Ave_maris_stella-Kyrie.krn:measure 30:beat 7:2E\       2G/     ([1f)   4ee\   3-2AT04
Jos0301b-Missa_Ave_maris_stella-Gloria.krn:measure 47:beat 7:(1A)     (1c)    4B-\    (0r)   3-2AT09
Jos0301b-Missa_Ave_maris_stella-Gloria.krn:measure 63:beat 4:4E\      (0f)    (1d)    (0r)   3-2BT02
Jos0301b-Missa_Ave_maris_stella-Gloria.krn:measure 71:beat 6:(0D)     ([0d)   4e\     (2.ff\) 3-2BT02
Jos0301c-Missa_Ave_maris_stella-Credo.krn:measure 36:beat 12:(2E\)    (2G/)   4f\     (2g/)  3-2AT04
Jos0301c-Missa_Ave_maris_stella-Credo.krn:measure 41:beat 10:(2BB-/)  4A/     (2B-\)  4c/    3-2AT09
Jos0301c-Missa_Ave_maris_stella-Credo.krn:measure 144:beat 7:(0r)     (0G)    4A/     (1b-)  3-2BT07
Jos0301c-Missa_Ave_maris_stella-Credo.krn:measure 156:beat 1:2G\      2B-\    2r      4a/]   3-2BT07
Jos0301d-Missa_Ave_maris_stella-Sanctus.krn:measure 5:beat 4:(1d)     (0.d)   8e\L    (1.ff) 3-2BT02
…
```

```
wc –l abc.txt
1574
```

Count 3-attack: `grep -v '[](]' abc.txt | wc -l`

Classify by transposition: `sed 's/.*3-2/3-2/' abc.txt | sort | uniq -c`

Classify by metric position: `sed 's/.*beat //; s/:.*//' abc.txt | sort -n | uniq –c`