Harmony Tools in Humdrum I

craigsapp@stanford.edu

29 March 2016

Installing Humdrum Tools

Best way is by using git to download from github:

https://github.com/humdrum-tools/humdrum-tools

Windoze: first install Cygwin: http://www.cygwin.com

Above contains two software packages for processing Humdrum data files:

https://github.com/humdrum-tools/humdrum "Humdrum Toolkit"

https://github.com/humdrum-tools/humextra "Humdrum Extras"

Websites:

"Humdrum Toolkit" https://www.humdrum.org

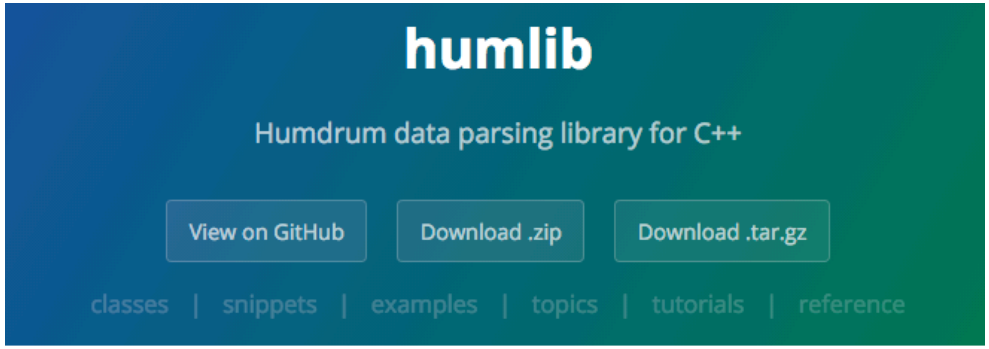
"Humdrum Extras" https://extras.humdrum.org

Humdrum Documentation

Online:	Humdrum Toolkit User Guide	http://www.humdrum.org/guide
	Humdrum Toolkit Manual	http://www.humdrum.org/man
	Humdrum Extras Manual	https://extra.humdrum.org/man
Local:	Humdrum Toolkit help	*program* -h
	Humdrum Toolkit help	man *program*
	Humdrum Extras help	*program* --options
Other:	CCARH Humdrum Portal	http://humdrum.ccarh.org

Humdrum Programming

http://humlib.humdrum.org



The humlib library is of a set of C++ classes for parsing Humdrum data files. It is easy to incorporate into your project by adding these two files:

1. An include file humlib.h
2. and a source file humlib.cpp

The source code uses some C++11-specific features, so add the `-stc=c++11` option when compiling with GNU g++ or the clang++ compiler. Also include the `-stdlib=libc++` option when compiling with clang. See the Makefile for compiling the library and Makefile.examples for compiling and linking executables.

Harmonic Tools

1. Intervals

mint: melodic intervals HT

hint: harmonic intervals HT

cint: composite intervals (counterpoint modules) HE

2. Chords

tntype: sonority types HE

sonority: triadic sonority types HE

3. Key

key/**keycor**: musical key by correlation HE/HT

mkeyscape: keyscape HE

finalis: Identify finalis tone of music HE

Humdrum Data

Download a large chunk: https://github.com/humdrum-tools/humdrum-data

Mostly from: http://kern.humdrum.org

Humdrum Extras programs can download internally:

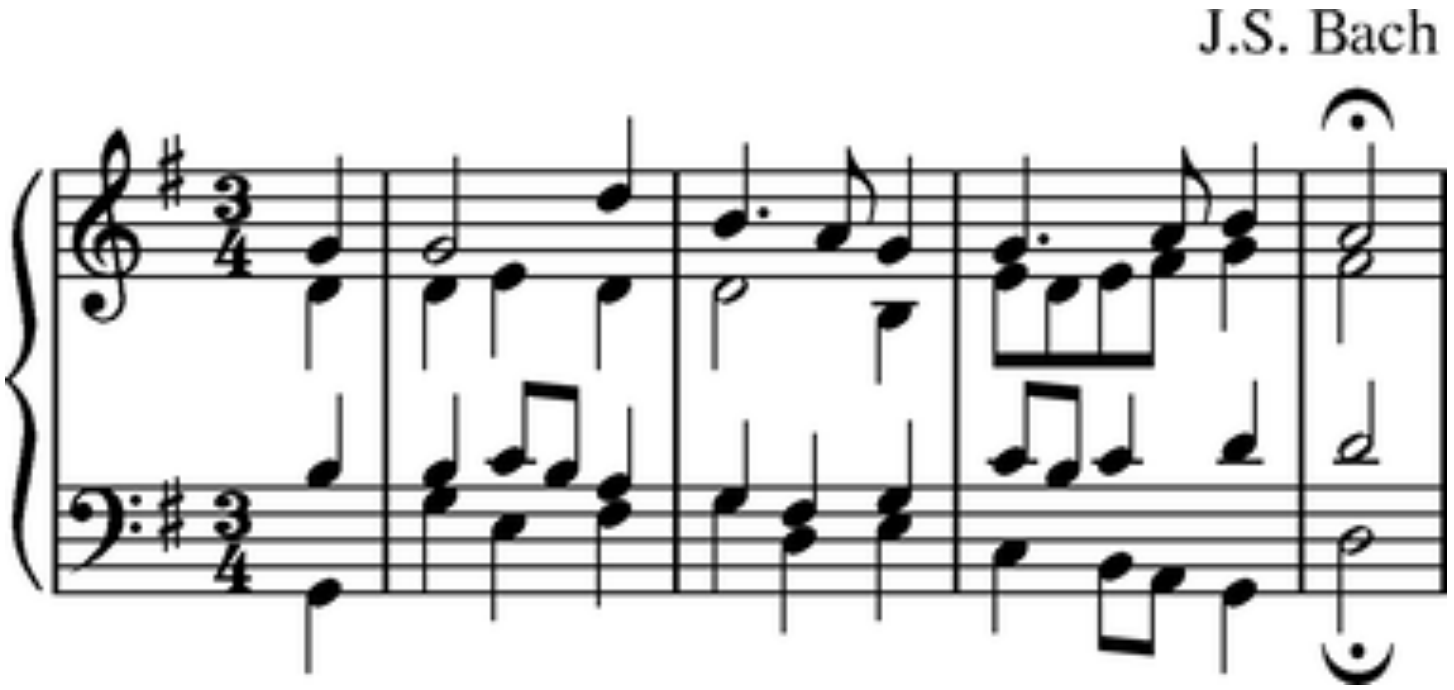
One file: keycor h://chorales/chor001.krn

All files in a set:
```
mkdir chorales; cd chorales
humsplit h://chorales
```

Download to HT programs: humcat h://chorales/chor001.krn | key

multiple files: humcat –s h://chorales | census -k

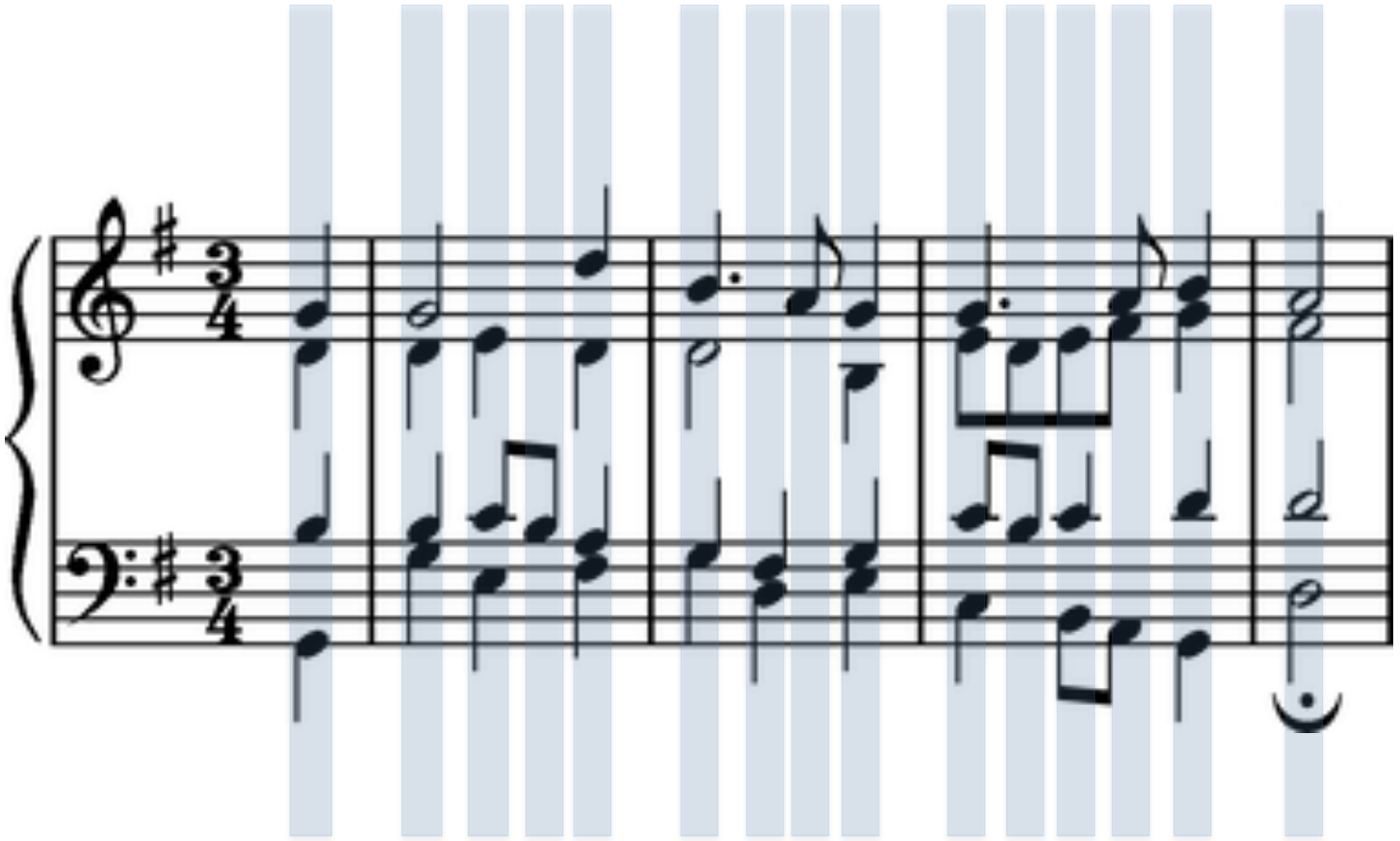
Humdrum score encoding



Voice→

Time →

Humdrum score encoding

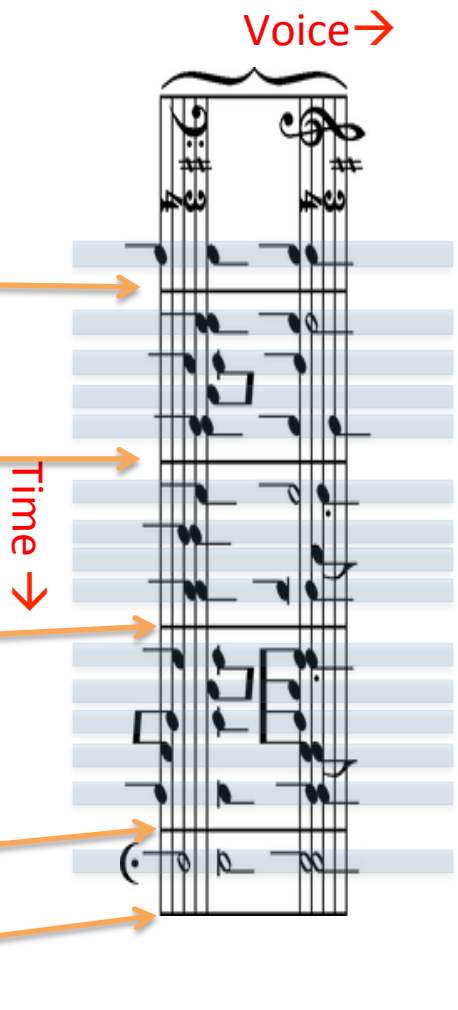


Voice→

Time →

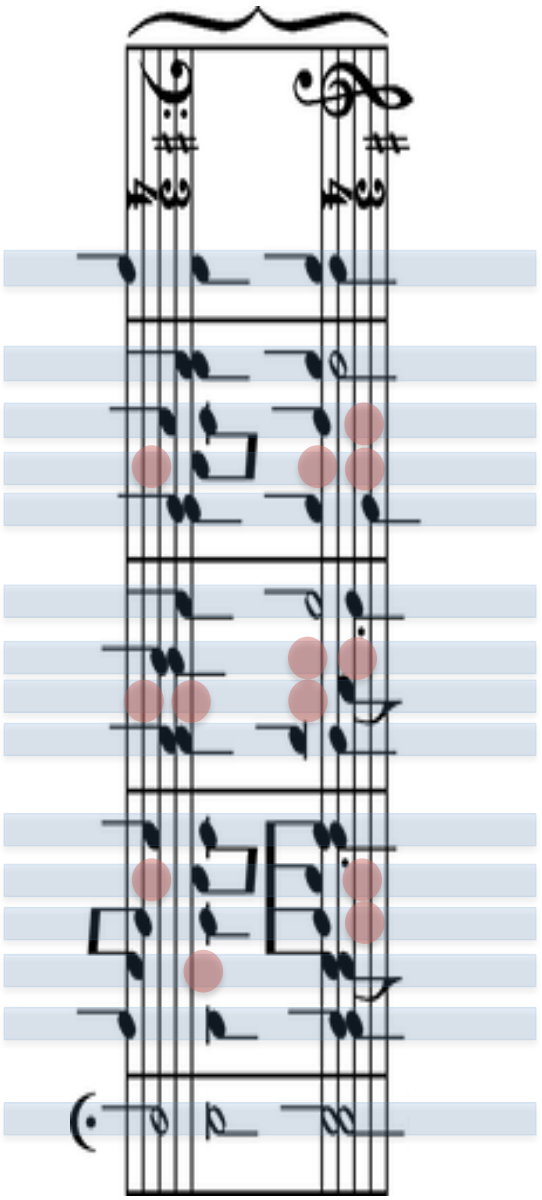
Humdrum score encoding

```
!!!COM:      Bach, Johann Sebastian
!!!CDT:      1685/02/21/-1750/07/28/
!!!OTL@@DE:  Aus meines Herzens Grunde
!!!OTL@EN:   From the Depths of My Heart
!!!SCT:      BWV 269
**kern       **kern       **kern       **kern
*Ibass       *Itenor      *Ialto       *Isoprn
*clefF4      *clefGv2     *clefG2      *clefG2
*k[f#]       *k[f#]       *k[f#]       *k[f#]
*G:          *G:          *G:          *G:
*M3/4        *M3/4        *M3/4        *M3/4
4GG          4B           4d           4g
=1           =1           =1           =1
4G           4B           4d           2g
4E           8cL          4e           .
.            8BJ          .            .
4F#          4A           4d           4dd
=2           =2           =2           =2
4G           4G           2d           4.b
4D           4F#          .            .
.            .            .            8a
4E           4G           4B           4g
=3           =3           =3           =3
4C           8cL          8eL          4.g
.            8BJ          8d           .
8BBL         4c           8e           .
8AAJ         .            8f#J         8a
4GG          4d           4g           4b
=4           =4           =4           =4
2D;          2d;          2f#;         2a;
4GG          4d           4g           4b
=            =            =            =
```



Humdrum score encoding

```
!!!COM:      Bach, Johann Sebastian
!!!CDT:      1685/02/21/-1750/07/28/
!!!OTL@@DE:  Aus meines Herzens Grunde
!!!OTL@EN:   From the Depths of My Heart
!!!SCT:      BWV 269
**kern       **kern       **kern       **kern
*Ibass       *Itenor      *Ialto       *Isoprn
*clefF4      *clefGv2     *clefG2      *clefG2
*k[f#]       *k[f#]       *k[f#]       *k[f#]
*G:          *G:          *G:          *G:
*M3/4        *M3/4        *M3/4        *M3/4
4GG          4B           4d           4g
=1           =1           =1           =1
4G           4B           4d           2g
4E           8cL          4e           .
.            8BJ          .            .
4F#          4A           4d           4dd
=2           =2           =2           =2
4G           4G           2d           4.b
4D           4F#          .            .
.            .            .            8a
4E           4G           4B           4g
=3           =3           =3           =3
4C           8cL          8eL          4.g
.            8BJ          8d           .
8BBL         4c           8e           .
8AAJ         .            8f#J         8a
4GG          4d           4g           4b
=4           =4           =4           =4
2D;          2d;          2f#;         2a;
4GG          4d           4g           4b
=            =            =            =
```



mint HT

Melodic Intervals

http://www.humdrum.org/man/mint

http://www.humdrum.org/guide/ch11

See a list of options:

mint -h

```
oznin:~ craig$ mint -h

MINT      : Determine melodic intervals between successive pitches.

     This command outputs the distance between successive pitches ex-
     pressed as diatonic interval size plus interval quality (e.g. m7).

Inputs processed:

     **kern,    **pitch,   **solfg,   **Tonh

Syntax:

     mint [-acde] [-b regexp] [-s regexp] [inputfile ...]

Options:

  -a        : output absolute pitch interval (no leading + or -)
  -c        : output compound intervals as non-compound intervals
  -d        : output diatonic interval size only, without interval quality
  -e        : with -s option, output skipped input data rather than null tokens
  -b regexp : break; do not calculate for records matching regexp
  -s regexp : skip; ignore records matching regexp; output null tokens

  Refer to reference manual for further details.
```

Local manpage:

man mint

```
mint(1)                                                          mint(1)

NAME
        mint   --  determine melodic intervals between successive pitches for
                   Humdrum inputs

SYNOPSIS
        mint  [-acd]  [-b regexp]  [-s  regexp]  [inputfile]  [  >  output-
              file.mnt]

DESCRIPTION
        The  mint  command determines the distance (interval) between succes-
        sive pitches.  Output pitch intervals are  expressed  as  a  diatonic
        interval  size  plus  interval  quality; a leading plus or minus sign
        indicates whether the interval is ascending or descending.  By way of
        illustration,  mint  will change a sequence of **pitch data tokens --
        such as C4, A4, E4 -- to the interval sequence +M6, -P4.  Each pitch-
        related  input  spine is transformed to a corresponding **mint output
        spine.

        The mint command determines melodic intervals only for  pitch  tokens
        within  individual  spines.  Pitch  intervals  across spines are not
        determined by mint.
```

mint

What are the melodic intervals in a chorale?:

mint chor001.krn | less

humcat h://chorales/chor001.krn | mint | less

What is the most common intervals in the chorale?:

mint chor001.krn | serialize | ridx –H | sort | uniq -c

HE HE U U

mint chor001.krn | serialize | ridx –H | sortcount

HE

rid –H

rid –GLId | grep –v =

HT U

As a percentage:

mint chor001.krn | serialize | ridx –H | sortcount -p

Super mint

What is the most common melodic interval in all of the chorales:?

```
cat chor*.krn | mint | serialize | ridx –H | sortcount  -p | head –n 20
```

U U

```
cat chor*.krn | mint  | serialize | ridx –H | grep –v '[]r]' | sortcount -p
```

U

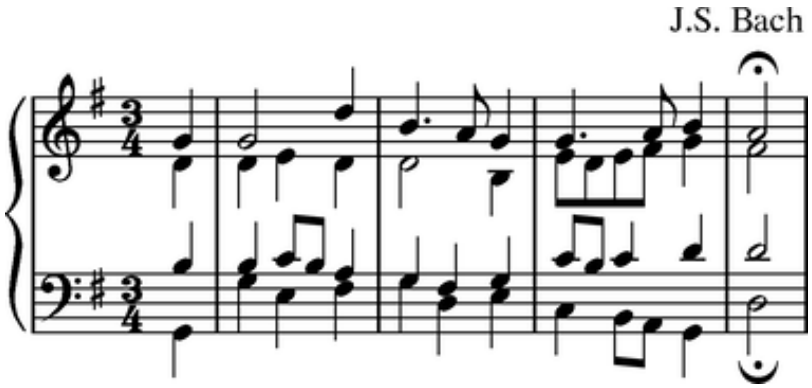
Least common intervals:

```
cat chor*.krn | mint  | serialize | ridx –H | grep –v '[]r]' | sortcount | tail –n 20
```

U

hint HT

hint chor001.krn




```
!!!COM:      Bach, Johann Sebastian
!!!CDT:      1685/02/21/-1750/07/28/
!!!OTL@@DE: Aus meines Herzens Grunde
!!!OTL@EN:  From the Depths of My Heart
!!!SCT:      BWV 269
**kern       **kern       **kern       **kern       **hint
*Ibass       *Itenor      *Ialto       *Isoprn      *
*clefF4      *clefGv2     *clefG2      *clefG2      *
*k[f#]       *k[f#]       *k[f#]       *k[f#]       *k[f#]
*G:          *G:          *G:          *G:          *G:
*M3/4        *M3/4        *M3/4        *M3/4        *M3/4
4GG          4B           4d           4g           M10 m3 P4
=1           =1           =1           =1           =1
4G           4B           4d           2g           M3 m3 P4
4E           8cL          4e           .            m6 M3
.            8BJ          .            .            -
4F#          4A           4d           4dd          m3 P4 P8
=2           =2           =2           =2           =2
4G           4G           2d           4.b          P1 P5 M6
4D           4F#          .            .            M3
.            .            .            8a           -
4E           4G           4B           4g           m3 M3 m6
```

```
hint chor001.krn > temp
assemble chor001.krn temp | less
```

```
satb2gs file.krn | autostem | hum2muse \
  | muse2ps =z21v120,120c120T^^ \
  | pstopnm -dpi=300 | convert - -trim \
    -resize '33%' file.png
```

hint documentation

hint –h gives one-page summary of *hint* command (same for all Humdrum Toolkit programs)
man hint gives command-line manual page

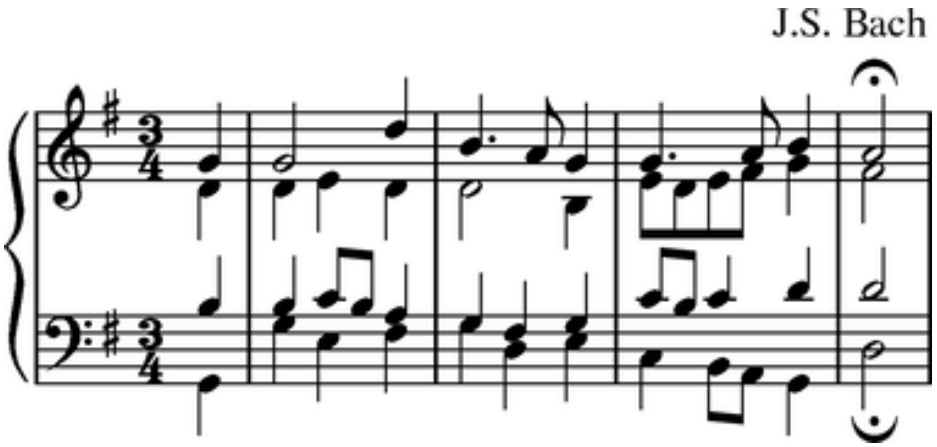
Humdrum Toolkit man pages:
http://www.humdrum.org/man
Older version
http://www.humdrum.org/Humdrum/commands/hint.html

Chapter 15 in the Humdrum Users' Guide (Harmonic Intervals):
http://www.humdrum.org/guide/ch15
Older version
http://www.humdrum.org/Humdrum/guide15.html

hint -a

- -a option shows intervals for all note permutations, not just “stacked intervals”

hint –a chor001.krn



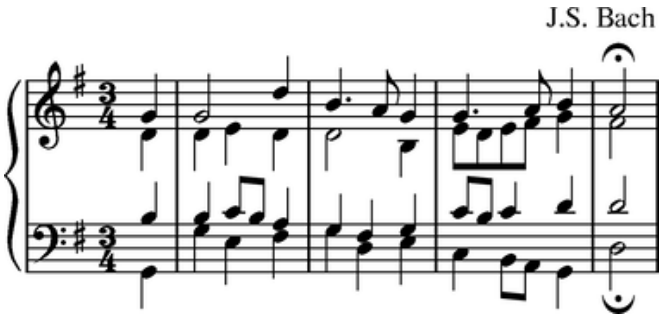

```
!!!COM: Bach, Johann Sebastian
!!!CDT: 1685/02/21/-1750/07/28/
!!!OTL@@DE:     Aus meines Herzens Grunde
!!!OTL@EN:      From the Depths of My Heart
!!!SCT: BWV 269
**kern   **kern   **kern   **kern   **hint
*Ibass   *Itenor  *Ialto   *Isoprn  *
*clefF4  *clefG2  *clefG2  *clefG2  *
*k[f#]   *k[f#]   *k[f#]   *k[f#]   *k[f#]
*G:      *G:      *G:      *G:      *G:
*M3/4    *M3/4    *M3/4    *M3/4    *M3/4
4GG      4B       4d       4g       M10 P12 P15 m3 m6 P4
=1       =1       =1       =1       =1
4G       4B       4d       2g       M3 P5 P8 m3 m6 P4
4E       8cL      4e       .        m6 P8 M3
.        8BJ      .        .        -
4F#      4A       4d       4dd      m3 m6 m13 P4 P11 P8
=2       =2       =2       =2       =2
4G       4G       2d       4.b      P1 P5 M10 P5 M10 M6
4D       4F#      .        .        M3
.        .        .        8a       -
4E       4G       4B       4g       m3 P5 m10 M3 P8 m6
```

(GG,B), (GG,d), (GG, g), (B,d), (B, g), (d, g)

hint -c

- Collapse the interval to a single octave. Such as: P12 → P8+P4 → P4

hint –ac chor001.krn
hint –a -c chor001.krn



**kern	**kern	**kern	**kern	**hint
*Ibass	*Itenor	*Ialto	*Isoprn	*
*clefF4	*clefGv2	*clefG2	*clefG2	*
*k[f#]	*k[f#]	*k[f#]	*k[f#]	*k[f#]
*G:	*G:	*G:	*G:	*G:
*M3/4	*M3/4	*M3/4	*M3/4	*M3/4
4GG	4B	4d	4g	M3 P5 P1 m3 m6 P4
=1	=1	=1	=1	=1
4G	4B	4d	2g	M3 P5 P1 m3 m6 P4
4E	8cL	4e	.	m6 P1 M3
.	8BJ	.	.	-
4F#	4A	4d	4dd	m3 m6 m6 P4 P4 P1
=2	=2	=2	=2	=2
4G	4G	2d	4.b	P1 P5 M3 P5 M3 M6
4D	4F#	.	.	M3
.	.	.	8a	-
4E	4G	4B	4g	m3 P5 m3 M3 P1 m6
=3	=3	=3	=3	=3
4C	8cL	8eL	4.g	P1 M3 P5 M3 P5 m3
.	8BJ	8d	.	m3
8BBL	4c	8e	.	m2 P4 M3
8AAJ	.	8f#J	8a	M6 P1 m3
4GG	4d	4g	4b	P5 P1 M3 P4 M6 M3

M10 → M3
P12 → P5
P15 → P1
m3 → m3
m6 → m6
P4 → P4

Most common harmonic intervals

hint –ac chor001.krn | serialize –c | ridx –H | sort | uniq –c | sort -nr

1. hint –ac chor001.krn: do harmonic interval analysis (Humdrum Toolkit)
2. serialize –c: force intervals one to a line (Humdrum Extras)
3. ridx –H: remove Humdrum file structure from data (Humdrum Extras)
4. sort: sort lines alphabetically (Unix)
5. uniq –c: output lines without repetitions, counting occurrences (Unix)
6. sort –nr: sort numerically, in reverse order (largest count first) (Unix)

```
**hint
*k[f#]
*G:
*M3/4
M3 P5 P1 m3 m6 P4
=1
M3 P5 P1 m3 m6 P4
m6 P1 M3
-
m3 m6 m6 P4 P4 P1
=2
P1 P5 M3 P5 M3 M6
M3
-
m3 P5 m3 M3 P1 m6
```

```
**hint
*k[f#]
*G:
*M3/4
M3
P5
P1
m3
m6
P4
=1
M3
P5
P1
m3
m6
P4
```

```
M3
P5
P1
m3
m6
P4
M3
P5
P1
m3
m6
P4
m6
P1
M3
m3
m6
m6
```

```
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
```

```
21 –
2  A4
4  M2
55 M3
23 M6
43 P1
30 P4
44 P5
 3 d5
 2 m2
42 m3
24 m6
 6 m7
```

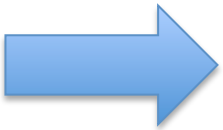
```
55 M3
44 P5
43 P1
42 m3
30 P4
24 m6
23 M6
21 –
 6 m7
 4 M2
 3 d5
 2 m2
 2 A4
```

ditto HT

- *hint* ignores null tokens, resulting in harmonic intervals between note attacks only
- To also include intervals to sustained intervals from previous attacks, use *ditto*
- This fills in the null token with the continuing data token it represents

ditto chor001.krn

**kern	**kern	**kern	**kern
*Ibass	*Itenor	*Ialto	*Isoprn
*clefF4	*clefG2	*clefG2	*clefG2
*k[f#]	*k[f#]	*k[f#]	*k[f#]
*G:	*G:	*G:	*G:
*M3/4	*M3/4	*M3/4	*M3/4
4GG	4B	4d	4g
=1	=1	=1	=1
4G	4B	4d	2g
4E	8cL	4e	.
.	8BJ	.	.
4F#	4A	4d	4dd
=2	=2	=2	=2
4G	4G	2d	4.b
4D	4F#	.	.
.	.	.	8a
4E	4G	4B	4g



**kern	**kern	**kern	**kern
*Ibass	*Itenor	*Ialto	*Isoprn
*clefF4	*clefG2	*clefG2	*clefG2
*k[f#]	*k[f#]	*k[f#]	*k[f#]
*G:	*G:	*G:	*G:
*M3/4	*M3/4	*M3/4	*M3/4
4GG	4B	4d	4g
=1	=1	=1	=1
4G	4B	4d	2g
4E	8cL	4e	**2g**
4E	8BJ	**4e**	**2g**
4F#	4A	4d	4dd
=2	=2	=2	=2
4G	4G	2d	4.b
4D	4F#	**2d**	**4.b**
4D	**4F#**	**2d**	8a
4E	4G	4B	4g

Most common harmonic interval including sustained notes

hint –ac chor001.krn | serialize –c | ridx –H | sort | uniq –c | sort -nr

ditto chor001.krn | hint –ac | serialize –c | ridx –H | sort | uniq –c | sort -nr

attacks

```
55 M3
44 P5
43 P1
42 m3
30 P4
24 m6
23 M6
21 -
 6 m7
 4 M2
 3 d5
 2 m2
 2 A4
```

+sustains

```
76 P5
75 M3
74 m3
59 P1
56 P4
43 M6
41 m6
18 m7
16 M2
 9 A4
 7 d5
 4 m2
 2 M7
```

All Bach chorales

```
cat chor*.krn | hint –ac | serialize –c \
| ridx –H | sort | uniq –c | sort -nr
```

Attacked minor thirds more common than attacked 5ths.

```
18053 m3
17545 P5
16147 M3
15263 P1
10812 P4
10035 M6
 9935 m6
 2537 M2
 1542 m7
 1534 A4
 1103 d5
  397 m2
  248 M7
  175 d7
   64 A2
   62 d4
   43 A5
    2 A6
    1 d1
```

```
cat chor*.krn | ditto | hint –ac | serialize –c \
| ridx –H | sort | uniq –c | sort -nr
```

```
29352 P5
28149 m3
23975 M3
22049 P1
20642 P4
17349 M6
15016 m6
 7015 M2
 6498 m7
 3721 A4
 3131 d5
 1555 m2
 1183 M7
  299 d7
  182 r
  176 d4
  146 A2
  111 A5
    3 d1
    3 A6
    1 d3
```

Sustained 5ths more common sustained minor thirds.

Beethoven string quartets

download and save all quartets locally:

humcat –s h://beethoven/quartets > beethoven-quartets.krns

or humsplit h://beethoven/quartests

hint –ac beethoven-quartets.krns | serialize –c | ridx –H | sort | uniq –c | sort -nr

ditto beethoven-quartets.krns | hint –ac | serialize –c | ridx –H | sort | uniq –c | sort -nr

```
60156 P1
 40163 m3          269 A6
 31479 M3          104 A1
 26374 P5           80 d6
 24285 M6           67 d3
 21775 P4           55 d1
 18348 m6           39 A3
 10697 m7           27 AA4
  9520 M2           23 d2
  7491 A4           18 A7
  6559 d5           10 dd5
  2237 M7            7 dd1
  1659 A2            7 AA2
  1419 m2            5 dd7
  1395 d7            2 dd4
   529 A5            2 AA5
   431 d4
```

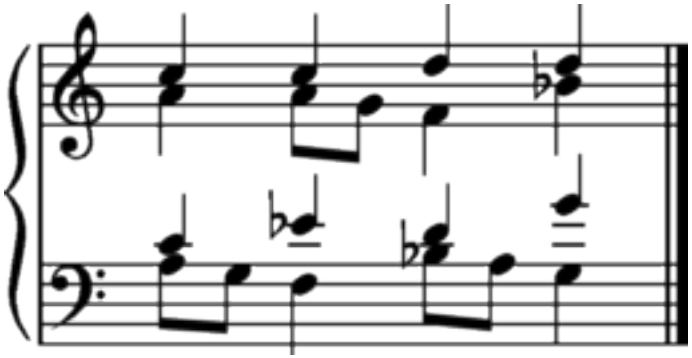
```
102597 P1
 69409 m3          403 A1
 57966 M3          223 d1
 55746 P5          147 d6
 46270 M6          121 A3
 45913 P4          113 d3
 35295 m6           72 d2
 23213 m7           40 A7
 22159 M2           37 AA4
 14037 A4           34 dd5
 12924 d5           15 dd1
  6139 M7           13 dd7
  4529 m2           13 AA2
  3020 A2            3 dd4
  2764 d7            3 AA5
  1453 A5            1 AA6
  1155 d4            1 AA3
   533 A6            1 AA1
```

tntype

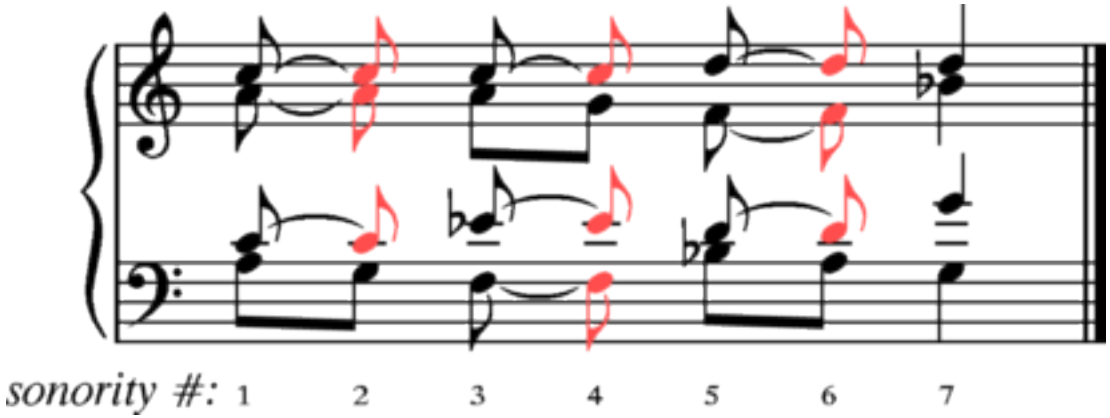
Tool for generalized description of sonority types (pitch-class sets sounding together)

documentation: http://extras.humdrum.org/man/tntype

similar to Humdrum Toolkit command pcset http://www.humdrum.org/Humdrum/commands/pcset.html

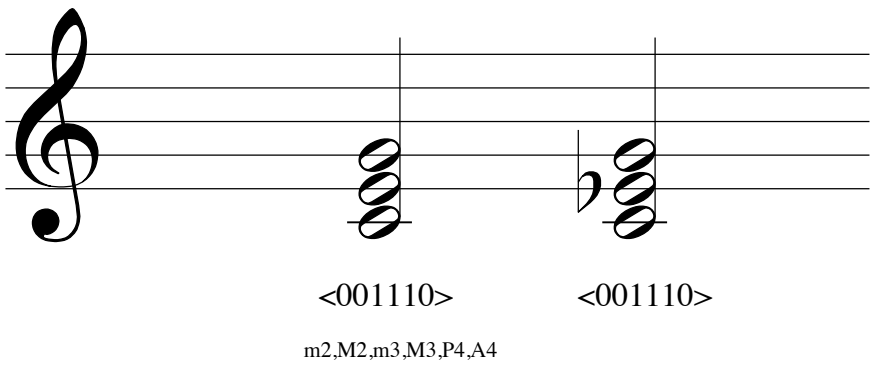


seven successive “sonorities” present in the above music:



Interval vectors

- Similar data generated by *hint*, but more compact

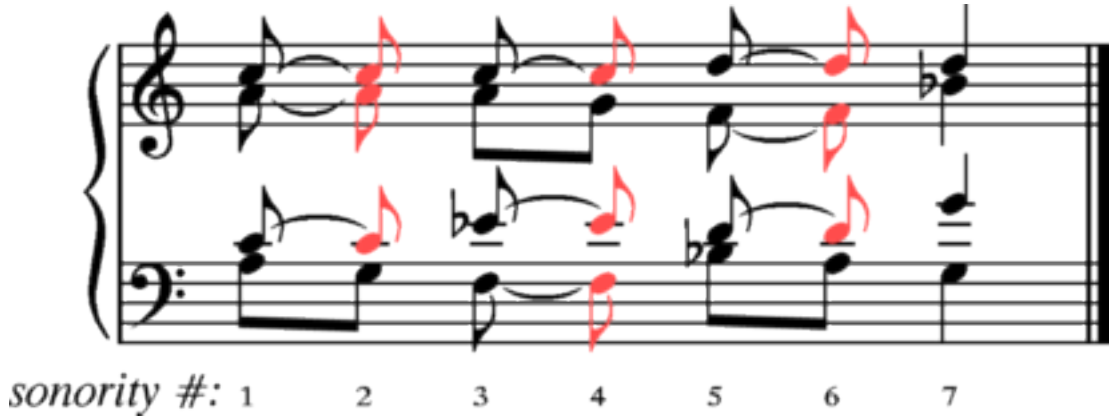


- Major and minor triads have the same interval content: m3, M3, P5
- <001110> is can be represented by the Forte number enumeration 3-11.
  - 3 = three pitch classes in set
  - 11 = 11th most compact organization of 3 pitch classes

    3-1 = c,c#,d 3-2 = c,c#,d# 3-3 = c,c#,e 3-4 = c,c#,f 3-5 = c,c#,f# etc.

tntype -d

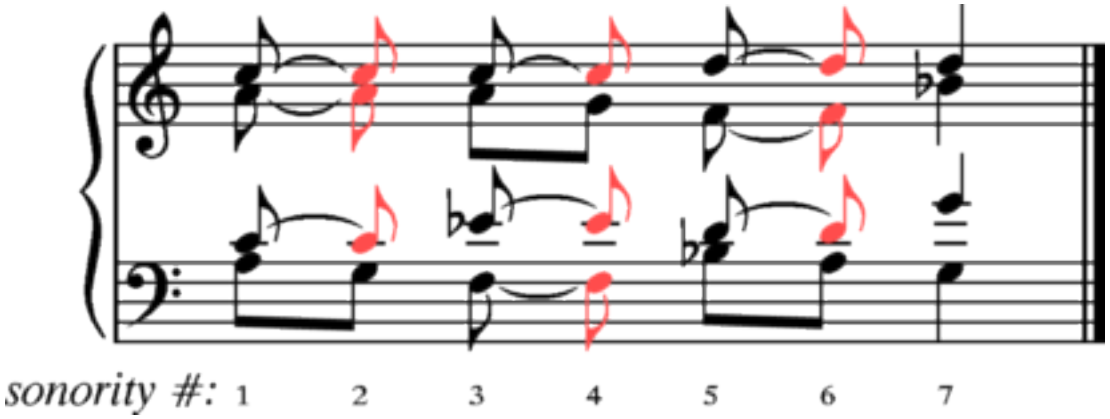
Generate a **dpc (diatonic pitch-class) spine listing unique pitch classes in sonorities



tntype –ad file.krn

```
**kern   **kern   **kern   **kern   **num   **dpc
=1-      =1-      =1-      =1-      =1-     =1-
8AL      4c       4a       4cc      1       A c
8GJ      .        .        .        2       (a) (c) G
4F       4e-      8aL      4cc      3       F a c e-
.        .        8gnXJ    .        4       (c) (e-) (F) g
8B-L     4d       4f       4dd      5       B- d f
8AJ      .        .        .        6       (d) (f) A
4G       4g       4b-      4dd      7       G b- d
==       ==       ==       ==       ==      ==
*-       *-       *-       *-       *-      *-
```

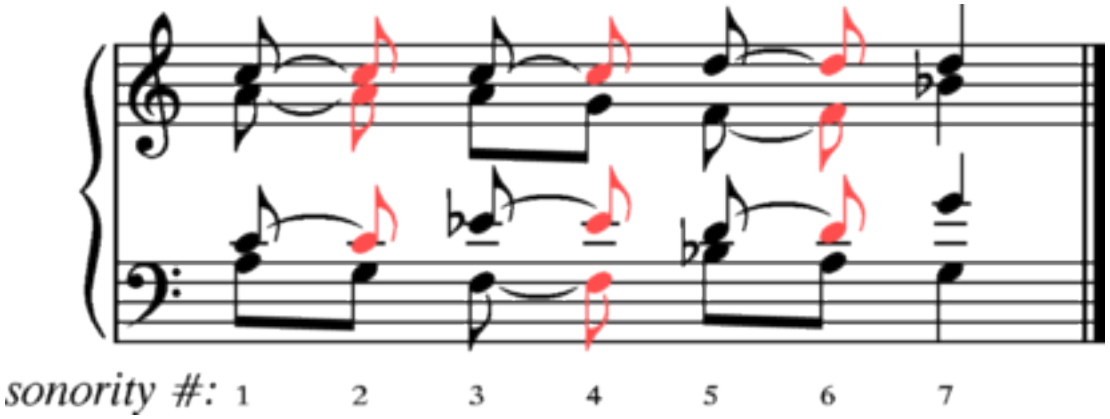
Normal form



tntype –na file.krn

**kern	**kern	**kern	**kern	**num	**nf
=1-	=1-	=1-	=1-	=1-	=1-
8AL	4c	4a	4cc	1	[90]
8GJ	.	.	.	2	[790]
4F	4e-	8aL	4cc	3	[9035]
.	.	8gnXJ	.	4	[0357]
8B-L	4d	4f	4dd	5	[A25]
8AJ	.	.	.	6	[259]
4G	4g	4b-	4dd	7	[7A2]
==	==	==	==	==	==
*-	*-	*-	*-	*-	*-

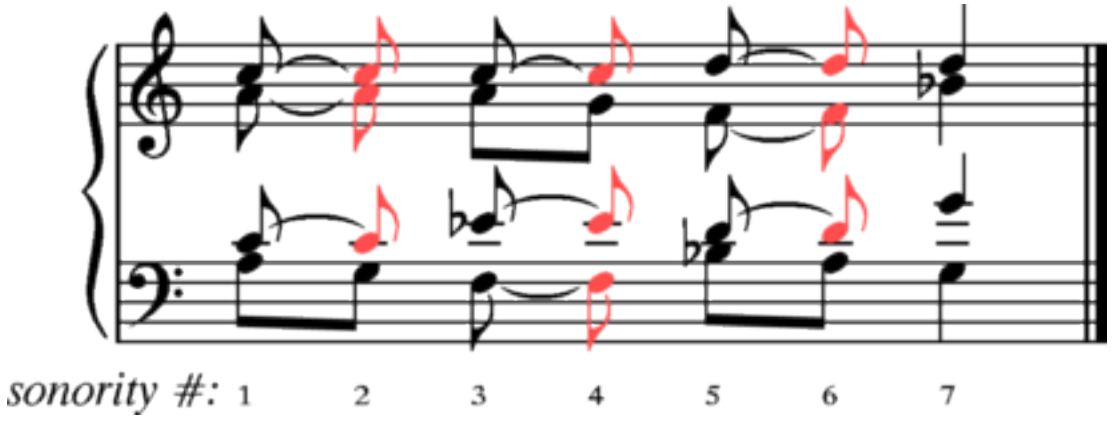
Transposed normal form



tntype –af file.krn

**kern	**kern	**kern	**kern	**num	**tnf
=1-	=1-	=1-	=1-	=1-	=1-
8AL	4c	4a	4cc	1	{03}
8GJ	.	.	.	2	{025}
4F	4e-	8aL	4cc	3	{0368}
.	.	8gnXJ	.	4	{0357}
8B-L	4d	4f	4dd	5	{047}
8AJ	.	.	.	6	{037}
4G	4g	4b-	4dd	7	{037}
==	==	==	==	==	==
*-	*-	*-	*-	*-	*-

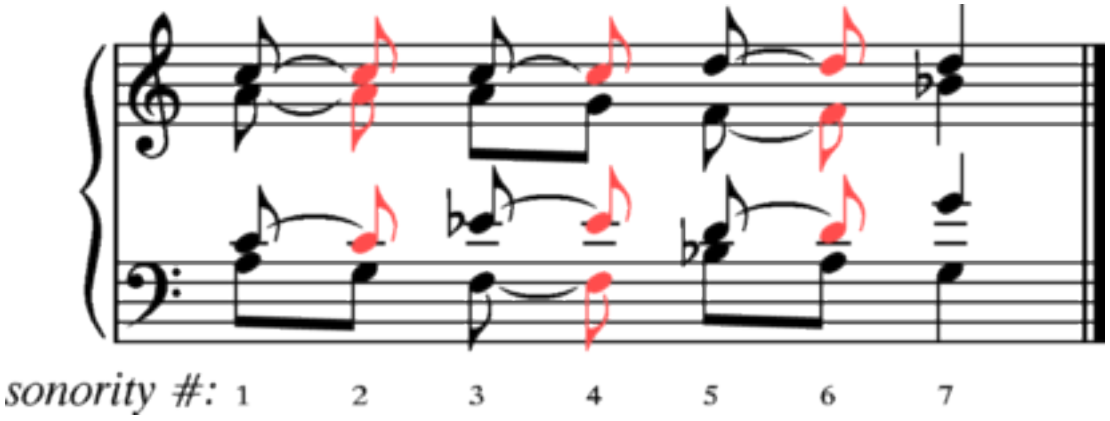
Preserving transposition



tntype –an input | tntype –aft

**kern	**kern	**kern	**kern	**num	**nf	**tnf
=1-	=1-	=1-	=1-	=1-	=1-	=1-
8AL	4c	4a	4cc	1	[90]	{03}T9
8GJ	.	.	.	2	[790]	{025}T7
4F	4e-	8aL	4cc	3	[9035]	{0368}T9
.	.	8gnXJ	.	4	[0357]	{0357}T0
8B-L	4d	4f	4dd	5	[A25]	{047}T10
8AJ	.	.	.	6	[259]	{037}T2
4G	4g	4b-	4dd	7	[7A2]	{037}T7
==	==	==	==	==	==	==
*-	*-	*-	*-	*-	*-	*-

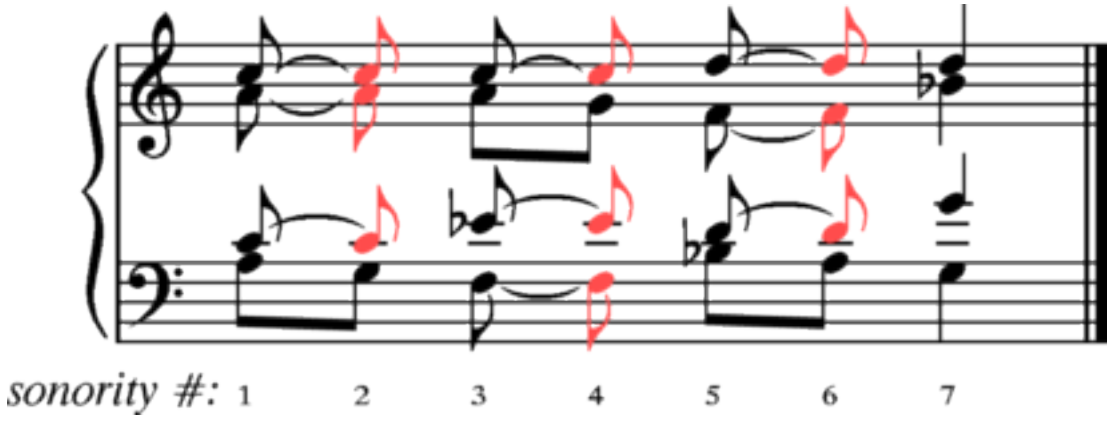
Forte pc-set enumerations



Tntype –an | tntype –aft | tntype -aF

**kern	**kern	**kern	**kern	**num	**nf	**tnf	**forte
=1-	=1-	=1-	=1-	=1-	=1-	=1-	=1-
8AL	4c	4a	4cc	1	[90]	{03}T9	2-3
8GJ	.	.	.	2	[790]	{025}T7	3-7
4F	4e-	8aL	4cc	3	[9035]	{0368}T9	4-27
.	.	8gnXJ	.	4	[0357]	{0357}T0	4-22
8B-L	4d	4f	4dd	5	[A25]	{047}T10	3-11
8AJ	.	.	.	6	[259]	{037}T2	3-11
4G	4g	4b-	4dd	7	[7A2]	{037}T7	3-11
==	==	==	==	==	==	==	==
*-	*-	*-	*-	*-	*-	*-	*-

Forte numbers, without inversions



tntype –an | tntype –aft | tntype –aF --Tn

**kern	**kern	**kern	**kern	**num	**nf	**tnf	**Tn
=1-	=1-	=1-	=1-	=1-	=1-	=1-	=1-
8AL	4c	4a	4cc	1	[90]	{03}T9	2-3
8GJ	.	.	.	2	[790]	{025}T7	3-7
4F	4e-	8aL	4cc	3	[9035]	{0368}T9	4-27
.	.	8gnXJ	.	4	[0357]	{0357}T0	4-22
8B-L	4d	4f	4dd	5	[A25]	{047}T10	3-11B
8AJ	.	.	.	6	[259]	{037}T2	3-11A
4G	4g	4b-	4dd	7	[7A2]	{037}T7	3-11A
==	==	==	==	==	==	==	==
*-	*-	*-	*-	*-	*-	*-	*-

Chord interpretations

humcat h://371chorales/chor001.krn | tntype -a | tntype -tfa | tntype -Da

**kern	**kern	**kern	**kern	**tnt	**tnf	**description
*Ibass	*Itenor	*Ialto	*Isoprn	*	*	*
*k[f#]	*k[f#]	*k[f#]	*k[f#]	*k[f#]	*k[f#]	*k[f#]
*M3/4	*M3/4	*M3/4	*M3/4	*M3/4	*M3/4	*M3/4
4GG	4B	4d	4g	3-11B	{047}T07	Major Chord
=1	=1	=1	=1	=1	=1	=1
4G	4B	4d	2g	3-11B	{047}T07	Major Chord
4E	8cL	4e	.	3-11B	{047}T00	Major Chord
.	8BJ	.	.	3-11A	{037}T04	Minor Chord
4F#	4A	4d	4dd	3-11B	{047}T02	Major Chord
=2	=2	=2	=2	=2	=2	=2
4G	4G	2d	4.b	3-11B	{047}T07	Major Chord
4D	4F#	.	.	3-11A	{037}T11	Minor Chord
.	.	.	8a	3-11B	{047}T02	Major Chord
4E	4G	4B	4g	3-11A	{037}T04	Minor Chord
=3	=3	=3	=3	=3	=3	=3
4C	8cL	8eL	4.g	3-11B	{047}T00	Major Chord
.	8BJ	8d	.	4-14B	{0457}T07	Perfect-fourth Major Tetrachord
8BBL	4c	8e	.	4-20	{0158}T11	Major-seventh Chord
8AAJ	.	8f#J	8a	3-10	{036}T06	Diminished Chord
4GG	4d	4g	4b	3-11B	{047}T07	Major Chord
=4	=4	=4	=4	=4	=4	=4

Bach chorale sonority types

- Are there more major or minor sonorities in Bach chorales?

By musical description:

```
humcat -s h://371chorales | tntype -D | ridx –H | sort | uniq -c | sort -nr
```

By Forte number:

```
humcat -s h://371chorales | tntype | ridx –H | sort | uniq -c | sort -nr
```

- Do Bach chorales in minor keys have more major or minor sonorities?

```
humcat –s h://371chorales | humsplit
humcat -s `egrep –l "^\*[a-g][#-]?:" chor*.krn` | tntype -D | ridx –H | sort | uniq -c | sort -nr
```

- What is the most common 7th chord sonority?

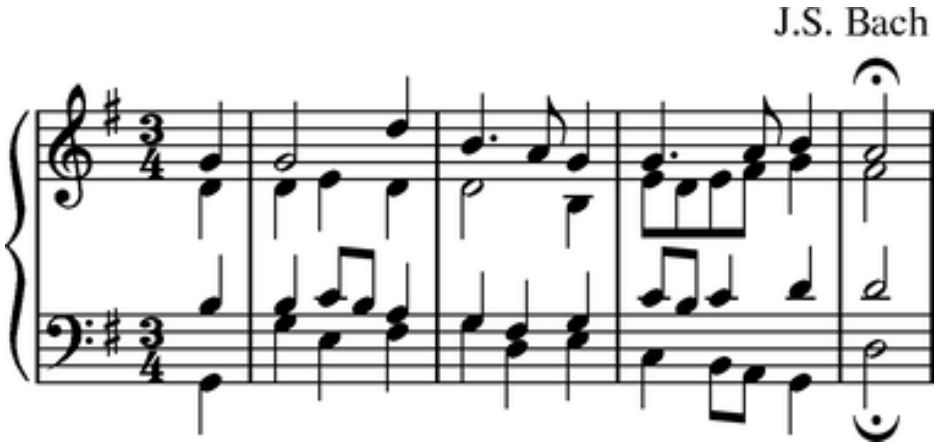
sonority

- similar to tntype program but has more triad-centered descriptions of sonorities

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

sonority -a h://371chorales/chor001.krn

**kern	**kern	**kern	**kern	**qual
*ICvox	*ICvox	*ICvox	*ICvox	*ICvox
*Ibass	*Itenor	*Ialto	*Isoprn	*
*k[f#]	*k[f#]	*k[f#]	*k[f#]	*k[f#]
*G:	*G:	*G:	*G:	*G:
*M3/4	*M3/4	*M3/4	*M3/4	*M3/4
*MM100	*MM100	*MM100	*MM100	*MM100
4GG	4B	4d	4g	maj:0:G
=1	=1	=1	=1	=1
4G	4B	4d	2g	maj:0:G
4E	8cL	4e	.	maj:1:C
.	8BJ	.	.	min:0:E
4F#	4A	4d	4dd	maj:1:D
=2	=2	=2	=2	=2
4G	4G	2d	4.b	maj:0:G
4D	4F#	.	.	min:1:B
.	.	.	8a	maj:0:D
4E	4G	4B	4g	min:0:E
=3	=3	=3	=3	=3
4C	8cL	8eL	4.g	maj:0:C
.	8BJ	8d	.	X
8BBL	4c	8e	.	majmaj:3:C
8AAJ	.	8f#J	8a	dim:1:F#
4GG	4d	4g	4b	maj:0:G
=4	=4	=4	=4	=4
2D;	2d;	2f#;	2a;	maj:0:D



Starting/Ending sonority

- Do Bach chorales start and end on the same sonority?
- How does the starting/ending chord root relate to the key of the chorale?

humcat -s h://371chorales | humsplit

```
#!/usr/bin/perl

@filelist = @ARGV;

foreach $file (@filelist) {
   processFile($file)
}

sub processFile {
  my ($file) = @_;
  $first_sonority = `sonority $file | ridx -GLIMd | grep -v "::" | head -n 1`;
  $last_sonority  = `sonority $file | ridx -GLIMd | grep -v "::" | tail -n 1`;
  $key = `egrep -i '^\\*[A-G][#-]?:' $file | head -n 1 | sed 's/\t.*//'`;
  chomp $first_sonority;
  chomp $last_sonority;
  chomp $key;
  print "$file\t$key\t$first_sonority\t$last_sonority\n";
}
```

Starting/Ending sonority (2)

chor001.krn	*G:	maj:0:G	maj:0:G
chor002.krn	*A:	maj:0:A	maj:0:A
chor003.krn	*a:dor	maj:0:E	maj:0:E
chor004.krn	*E:	maj:0:E	maj:0:E
chor005.krn	*G:	maj:0:G	maj:0:G
chor006.krn	*F:	maj:0:F	maj:0:F
chor007.krn	*A:	maj:0:A	maj:0:A
chor008.krn	*f:dor	min:0:F	maj:0:F
chor009.krn	*G:	maj:0:G	maj:0:G
chor010.krn	*a:	domsev:3:E	maj:0:E
chor011.krn	*C:	maj:0:C	maj:0:C
chor012.krn	*a:	min:0:A	maj:0:A
chor013.krn	*a:	min:0:A	maj:0:A
chor014.krn	*G:	maj:0:G	maj:0:G
chor015.krn	*d:dor	min:0:D	maj:0:D
chor016.krn	*b:	maj:0:F#	maj:0:F#
chor017.krn	*e:	min:0:E	maj:0:E
chor018.krn	*G:	maj:0:G	maj:0:G
chor019.krn	*g:dor	min:0:G	maj:0:G
chor020.krn	*D:	maj:0:D	maj:0:D
chor021.krn	*a:	domsev:3:E	maj:0:E
chor022.krn	*E-:	maj:0:E-	maj:0:E-
chor023.krn	*a:	min:0:A	maj:0:A
chor024.krn	*D:	maj:0:D	maj:0:D
chor025.krn	*f:dor	min:0:F	maj:0:F

Check for unusual cases

```
#!/usr/bin/perl

@filelist = @ARGV;

foreach $file (@filelist) {
    processFile($file)
}

sub processFile {
   my ($file) = @_;
   $first_sonority = `sonority $file | ridx -GLIMd | grep -v "::" | head -n 1`;
   $last_sonority  = `sonority $file | ridx -GLIMd | grep -v "::" | tail -n 1`;
   $key = `egrep -i '^\\*[A-G][#-]?:' $file | head -n 1 | sed 's/\t.*//'`;
   chomp $first_sonority;
   chomp $last_sonority;
   chomp $key;
   $first_sonority =~ /:([^:]*)$/;
   $first_root = $1;
   $last_sonority =~ /:([^:]*)$/;
   $last_root = $1;
   $key =~ /^\*([A-G][#-]?):/;
   $key_root = uc($1);
   if (($first_root ne $last_root) or ($first_root ne $key_root)) {
      print "$file\t$key\t$first_sonority\t$last_sonority\n";
   }
}
```

Inconsistent start/end/key

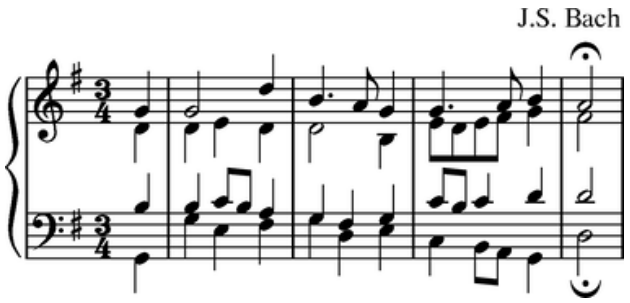
	chor056.krn	*b:	min:0:E	maj:0:F#
	chor057.krn	*a:	maj:0:E	min:0:A
	chor066.krn	*a:	min:0:D	maj:0:A
	chor071.krn	*e:	maj:1:B	maj:0:E
	chor074.krn	*F:	min:0:D	maj:0:F
	chor077.krn	*A:	min:0:F#	maj:0:A
	chor079.krn	*a:	min:0:A	maj:0:E
	chor083.krn	*A:	min:0:F#	maj:0:A
	chor089.krn	*b:	min:0:B	maj:0:F#
	chor119.krn	*c:dor	maj:0:B-	maj:0:C
	chor121.krn	*A:	min:0:F#	maj:0:A
	chor154.krn	*G:mix	min:0:D	maj:0:G
	chor162.krn	*d:dor	maj:0:A	maj:0:E
	chor181.krn	*e:	min:0:E	maj:0:B
	chor205.krn	***C:**	**min:0:E**	**maj:0:E → *e:phr**
	chor208.krn	*e:	min:0:E	maj:0:B
	chor227.krn	*d:	min:0:G	maj:0:D
	chor248.krn	*G:	maj:1:D	maj:0:G
bwv 77/6	**chor253.krn**	***g:**	**maj:0:A**	**maj:0:D**
	chor255.krn	*D:	maj:1:A	maj:0:D
	chor275.krn	*A:	maj:1:C#	maj:0:A
	chor284.krn	*C:mix	maj:0:F	maj:0:C
	chor286.krn	*b:	min:0:B	maj:0:F#
	chor288.krn	*A:mix	min:0:F#	maj:0:A
	chor291.krn	*D:	maj:1:A	maj:0:D
	chor311.krn	*F:	maj:1:C	maj:0:F
	chor314.krn	*e:	maj:0:B	maj:0:F#
	chor315.krn	*G:	min:0:E	maj:0:G
	chor333.krn	*D:	maj:0:F#	maj:0:D
	chor337.krn	*F:	maj:0:A	maj:0:F
	chor341.krn	*A:	min:0:F#	maj:0:A
	chor357.krn	*G:mix	maj:0:C	maj:0:G
	chor359.krn	*b:	maj:0:D	incmaj:0:B
	chor364.krn	*b:	maj:0:F#	maj:0:B
	chor367.krn	*b:	maj:0:D	maj:0:F#

key

humcat h://371chorales/chor001.krn | key

Estimated key: G major (r=0.9501) confidence: 52.3%

humcat h://371chorales/chor001.krn | key -a



Tonic[0]	major 0.441131	minor 0.0554652
Tonic[1]	major -0.711388	minor -0.415044
Tonic[2]	major 0.775722	minor 0.354884
Tonic[3]	major -0.301544	minor -0.42342
Tonic[4]	major -0.085096	minor 0.540753
Tonic[5]	major 0.00550523	minor -0.515126
Tonic[6]	major -0.407599	minor 0.0398076
Tonic[7]	major 0.9501	minor 0.434019
Tonic[8]	major -0.602254	minor -0.310748
Tonic[9]	major 0.158757	minor 0.224714
Tonic[10]	major -0.11878	minor -0.679797
Tonic[11]	major -0.104554	minor 0.694493

Estimated key: G major (r=0.9501) confidence: 52.3%

keycor

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

- Generalized version of the Humdrum Toolkit key program.

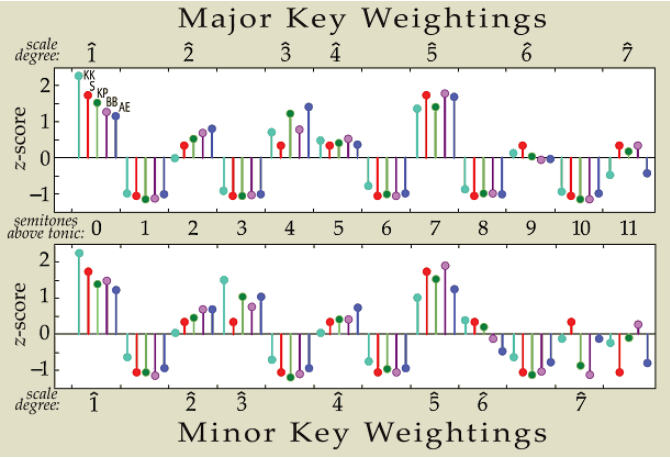
keycor h://371chorales/chor001.krn

The best key is: G Major

keycor -c h://371chorales/chor001.krn

$$R(x,y) = \frac{\sum(x_n-\bar{x})(y_n-\bar{y})}{\sqrt{\sum(x_n-\bar{x})^2\sum(y_n-\bar{y})^2}}$$

$$\text{key}_k = \arg\max_k R(x, y_k)$$

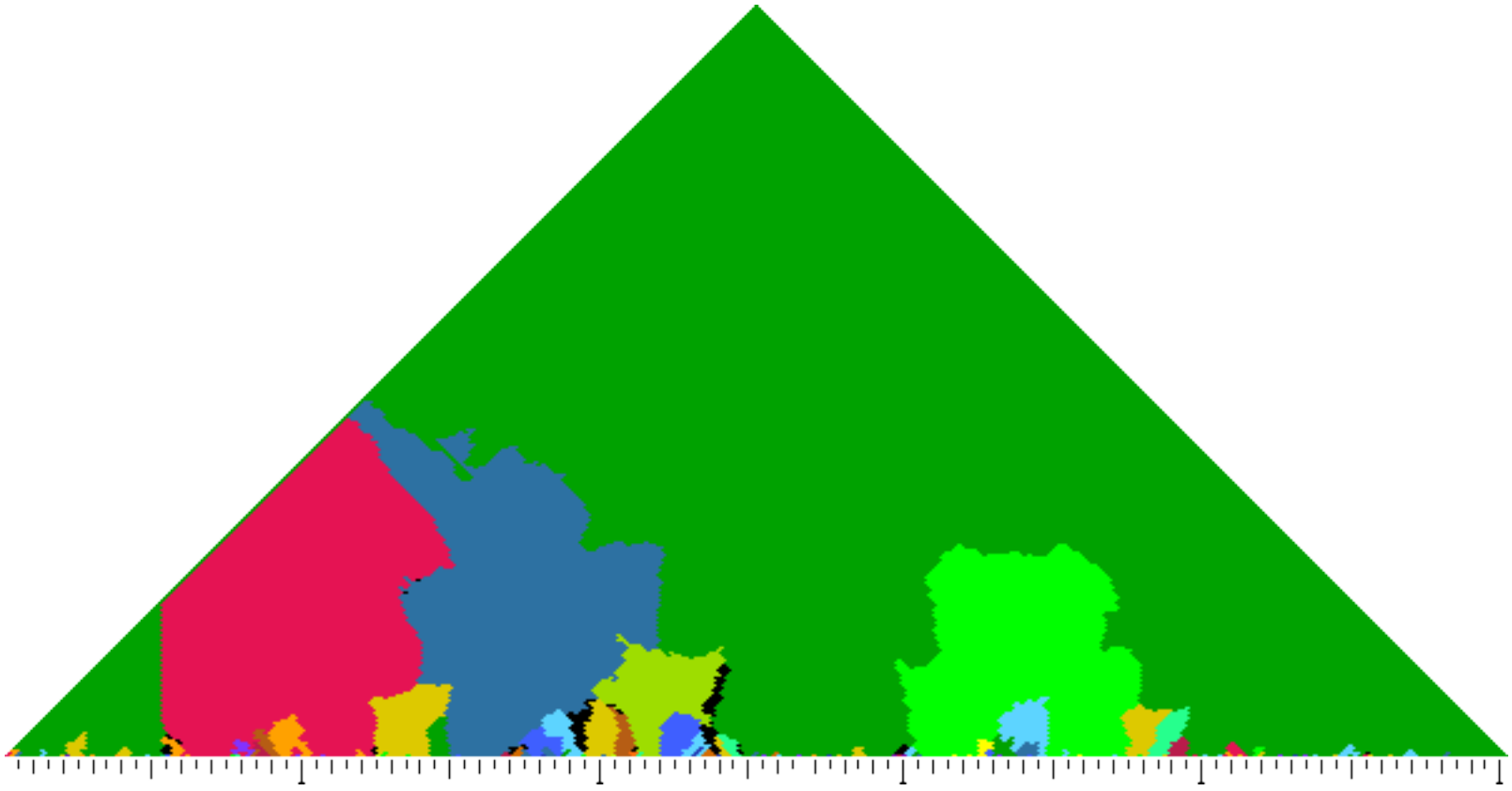


**key	**rval	**conf	**start	**mid	**end
G	0.952	82	=0	=5	=11
G	0.956	87	=1	=6	=11
G	0.973	87	=1	=6	=12
G	0.974	97	=1	=6	=12
G	0.969	90	=2	=7	=12
G	0.975	94	=2	=7	=13
G	0.971	86	=2	=7	=13
G	0.969	92	=3	=8	=13
G	0.959	92	=3	=8	=14
G	0.959	86	=3	=8	=14
G	0.969	81	=4	=9	=14
G	0.958	71	=4	=9	=15
G	0.959	70	=4	=9	=15
G	0.962	71	=5	=10	=15
G	0.963	64	=5	=10	=16
G	0.960	67	=5	=10	=16
G	0.943	64	=6	=11	=16
G	0.960	69	=6	=11	=17
G	0.954	72	=6	=11	=17
G	0.948	64	=7	=12	=17
G	0.952	66	=7	=12	=18
G	0.966	76	=7	=12	=18
G	0.976	86	=8	=13	=18
G	0.975	83	=8	=13	=19
G	0.965	87	=8	=13	=19
G	0.970	93	=9	=14	=19
G	0.975	88	=9	=14	=20
G	0.972	82	=9	=14	=20
G	0.978	80	=10	=15	=20
G	0.980	78	=10	=15	=21
G	0.972	68	=10	=15	=21
*-	*-	*-	*-	*-	*-

mkeyscape

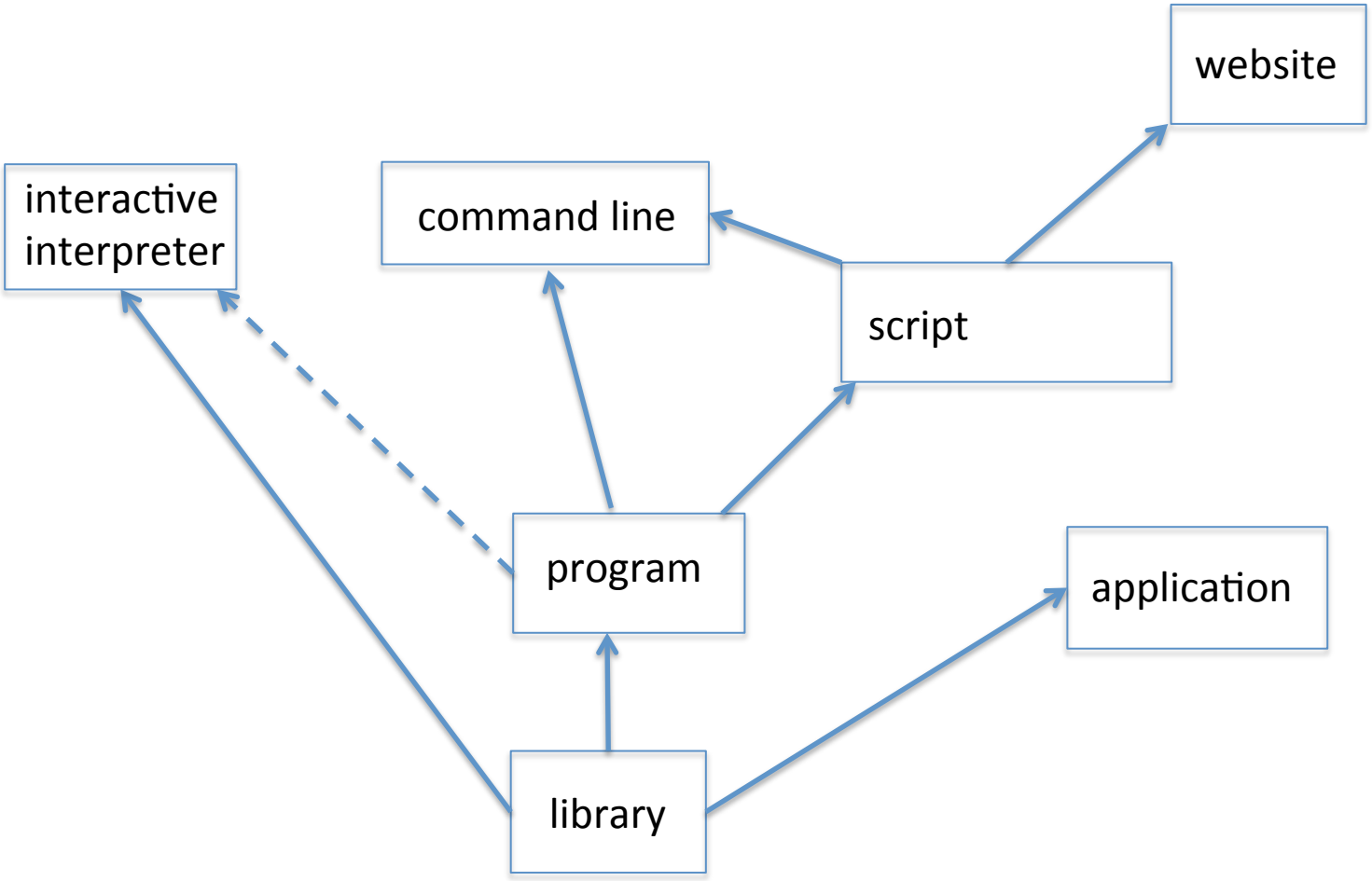
http://extras.humdrum.org/man/mkeyscape

- Structural analysis of key in a piece of music



* Beethoven's 5th symphony in C minor, first movement

Humdrum Interfacing



C++ harmony analysis skeleton

git clone https://github.com/craigsapp/humextras
cd humextras
make library

Then place the following code into humextras/src-programs/midinotes.cpp
and then type "make midinotes", then type "bin/midinotes h://371chorales/chor001.krn"

```
**kern  **kern  **kern  **kern
*Ibass  *Itenor *Ialto  *Isoprn
*k[f#]  *k[f#]  *k[f#]  *k[f#]
*G:     *G:     *G:     *G:
*M3/4   *M3/4   *M3/4   *M3/4
4GG     4B      4d      4g
=1      =1      =1      =1
4G      4B      4d      2g
4E      8cL     4e      .
.       8BJ     .       .
4F#     4A      4d      4dd
=2      =2      =2      =2
4G      4G      2d      4.b
4D      4F#     .       .
.       .       .       8a
4E      4G      4B      4g
=3      =3      =3      =3
4C      8cL     8eL     4.g
.       8BJ     8d      .
8BBL    4c      8e      .
```

```
43 59 62 67
55 59 62 67
52 60 64 67
52 59 64 67
54 57 62 74
55 55 62 71
50 54 62 71
50 54 62 69
52 55 59 67
48 60 64 67
48 59 62 67
47 60 64 67
```

midinotes.cpp (1)

```
// This program takes multiple input files or standard input and outputs a
// list of MIDI pitches sounding at a every time (line) in the input score(s).

#include "humdrum.h"

void processSegment      (HumdrumFile& infile);
void processLine         (HumdrumFile& infile, int line);
void addFieldMidiNotes   (Array<int>& notelist, HumdrumFile& infile, int line,
                          int field);

int main(int argc, char** argv) {
   Options options;
   options.process(argc, argv);
   HumdrumFileSet infiles;
   int i;
   int incount = options.getArgCount();
   if (incount < 1) {
      infiles.read(cin);
   } else {
      for (i=0; i<incount; i++) {
         infiles.readAppend(options.getArg(i+1));
      }
   }

   for (i=0; i<infiles.getCount(); i++) {
      processSegment(infiles[i]);
   }

   return 0;
}
```

midinotes.cpp (2)

```
// processSegment -- handle data extraction from one Humdrum file segment
//    (such as a movement, or individual work from a collection).
void processSegment(HumdrumFile& infile) {
   for (int i=0; i<infile.getNumLines(); i++) {
      if (!infile[i].isData()) {
         continue;
      }
      processLine(infile, i);
   }
}

// processLine --  Print notes for one line of data.
void processLine(HumdrumFile& infile, int line) {
   Array<int> notelist;
   notelist.setSize(1000);
   notelist.setSize(0);
   for (int j=0; j<infile[line].getFieldCount(); j++) {
      if (infile[line].isExInterp(j, "**kern")) {
         addFieldMidiNotes(notelist, infile, line, j);
      }
   }
   for (int i=0; i<notelist.getSize(); i++) {
      cout << notelist[i];
      if (i < notelist.getSize()-1) {
         cout << ' ';
      }
   }
   if (notelist.getSize() > 0) {
      cout << '\n';
   }
}
```

midinotes.cpp (3)

```
// addFieldMidiNotes -- Print one or more notes in a Humdrum **kern token.
//     Don't do anything if there is a rest.

void addFieldMidiNotes(Array<int>& notelist, HumdrumFile& infile, int line,
      int field) {
   int k;
   int midinote;
   int tline  = line;
   int tfield = field;
   char buffer[1024] = {0};

   if (strcmp(infile[line][field], ".") == 0) {
      // resolve data represented by null token
      tline  = infile[line].getDotLine(field);
      tfield = infile[line].getDotSpine(field);
   }

   int tcount = infile[tline].getTokenCount(tfield);
   for (k=0; k<tcount; k++) {
      infile[tline].getToken(buffer, tfield, k);
      if (strchr(buffer, 'r') != NULL) {
         // ignore rests
         continue;
      }
      midinote = Convert::kernToMidiNoteNumber(buffer);
      notelist.append(midinote);
   }
}
```