Cross-Drive Analysis with bulk_extractor and CDA tool

Simson L. Garfinkel

- Wednesday, October 3r, 2012. 13:00–13:35 *Track 1*
  - *Cross-drive analysis (CDA) is a forensic technique that correlates information found on multiple digital devices (hard drives, camera cards, cell phones, etc.). Unlike existing approaches for analyzing multiple devices, CDA takes into account the number of devices on which identifiers are found, so that an email address that appears on three or four drives is more important than an email address that appears on just one or two. CDA can also be applied to a large corpus of drives, so that email addresses can automatically be pruned out if they appear in the "background."*

NPS is the Navy's Research University.

Location: Monterey, CA

Students: 1500

- US Military (All 5 services)
- US Civilian (Scholarship for Service & SMART)
- Foreign Military (30 countries)

Schools:

- Business & Public Policy
- Engineering & Applied Sciences
- Operational & Information Sciences
- International Graduate Studies

NCR Initiative:

- 8 offices on 5th floor, 900N Glebe Road, Arlington
- Current staffing: 4 professors, 2 lab managers, 2 programmers, 4 contractors
- **WE ARE HIRING!**
- **OPEN SLOTS FOR .GOV PHDs!**

Monterey, CA

900N Glebe, Arlington, VA

Outline of the next 35 minutes...

Introducing bulk_extractor

- Overview and history
- Announcing bulk_extractor 1.3!

Cross drive analysis

- What is it?
- Why do it?

Context-sensitive stop lists

- Simple CDA

Using cda_tool

- making stop-lists

Introducing bulk_extractor

Between 2005 and 2008, I interviewed law enforcement regarding their use of forensic tools.

Law enforcement officers wanted a *highly automated* tool for finding:

- Email addresses & Credit card numbers (including track 2 information)
- Phone numbers, GPS coordinates & EXIF information from JPEGs
- Search terms (extracted from URLs)
- All words that were present on the disk (for password cracking)

The tool had to:

- Run on Windows, Linux, and Mac-based systems
- Run with *no* user interaction
- Operate on raw disk images, split-raw volumes, E01 files, and AFF files
- Run at maximum I/O speed of physical drive
- Never crash

Moving technology from the lab to the field has been challenging:

- Must work with evidence files of *any size* and on *limited hardware.*
- Users can't provide their data when the program crashes.
- Users are *analysts* and *examiners*, not engineers.

The tool implements Stream-Based Disk Forensics: Scan the disk from beginning to end; do your best.

0 1TB

3 hours, 20 min to *read* the data

1. Read all of the blocks in order.
2. Look for information that might be useful.
3. Identify & extract what's possible in a single pass.

Primary Advantage: Speed

No disk seeking! (Good for HDs, SSDs, & E01 files)

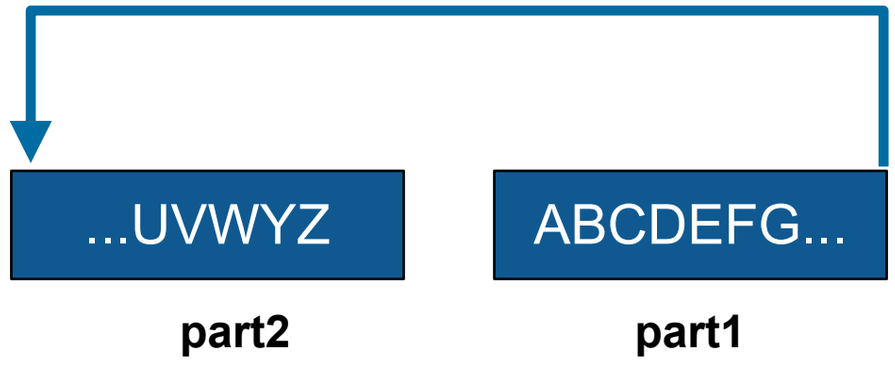
Easy to parallelize ("embarrassingly parallel")

Reads all the data — allocated files, deleted files, file fragments

Caveats:

- Compressed data must be decompressed
  - —*Fragmented, compressed files may not be recovered*
- Can only read at maximum I/O transfer rates if data can be *processed*
  - —*Even 24+ cores may not be enough*
- Does not provide file names
  - —*File names can be determined with a separate metadata extraction step.*

Fragmented files may not be recovered



ZIP, GZIP & LZMA use *adaptive* compression algorithms.

- Part 1 required to decompress part 2.
- Also an issue for JPEG.

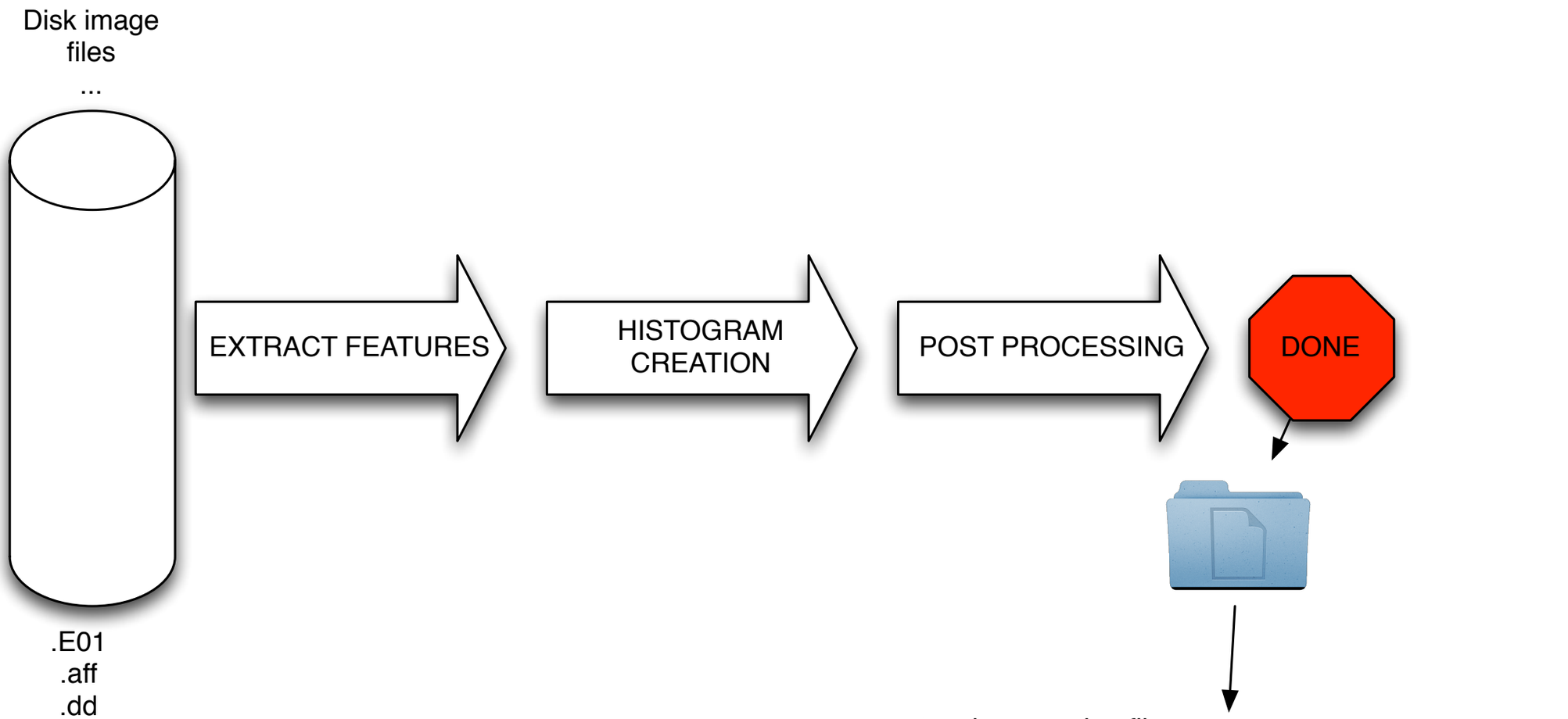
Fortunately, most files are *not* fragmented.

- Individual components of a ZIP can be recovered (e.g. `word/document.xml`)

Most files that *are* fragmented have carvable internal structure:

- Log files, Outlook PST files, etc.

bulk_extractor has three phases of operation: Feature Extraction; Histogram Creation; Post Processing



Output is a *directory* containing:

- feature files; histograms; carved objects
- Mostly in UTF-8; some XML
- Can be bundled into a ZIP file and process with bulk_extractor_reader.py

Feature files are UTF-8 files that contain extracted data.

```
# UTF-8 Byte Order Marker; see http://unicode.org/faq/utf_bom.html
# bulk_extractor-Version: 1.3b1-dev2
# Filename: /corp/nps/drives/nps-2009-m57-patents/charlie-2009-12-11.E01
# Feature-Recorder: telephone
# Feature-File-Version: 1.1
...
6489225486    (316) 788-7300   Corrine Porter (316) 788-7300,,,,,,Phase I En
6489230027    620-723-2638    ,,,,Dan Hayse - 620-723-2638,,,,,,Phase I En
6489230346    620-376-4499    Bertha Mangold -620-376-4499,,,,,,Phase I En
...
3772517888-GZIP-28322  (831) 373-5555  onterey-<nobr>(831) 373-5555</nobr>
3772517888-GZIP-29518  (831) 899-8300  Seaside - <nobr>(831) 899-8300</nobr>
5054604751    716-871-2929    a%,888-571-2048,716-871-2929\x0D\x0ACPV,,,%Cape
```

Offset **Feature** **Context**

Designed for easy processing by python, perl or C++ program

- “Loosely ordered.”
- -GZIP- indicates that data was decompressed
- Non-UTF-8 characters are escaped

Histogram system automatically summarizes features.

```
# UTF-8 Byte Order Marker; see http://unicode.org/faq/utf_bom.html
# bulk_extractor-Version: 1.3b1-dev2
# Filename: /corp/nps/drives/nps-2009-m57-patents/charlie-2009-12-11.E01
# Feature-Recorder: email
# Histogram-File-Version: 1.1
...
n=875   mozilla@kewis.ch          (utf16=3)
n=651   charlie@m57.biz (utf16=120)
n=605   ajbanck@planet.nl
...
n=288   mattwillis@gmail.com
n=281   garths@oeone.com
n=226   michael.buettner@sun.com          (utf16=2)
n=225   bugzilla@babylonsounds.com
n=218   berend.cornelius@sun.com
n=210   ips@mail.ips.es
n=201   mschroeder@mozilla.x-home.org
n=186   pat@m57.biz      (utf16=1)
```

-

New in bulk_extractor 1.3

New supported data types:

- Windows PE Scanner
- Linux ELF Scanner
- VCARD Scanner
- BASE16 scanner
- Windows directory carver

Better Unicode Support:

- Histograms now UTF-8 / UTF-16 aware
- Feature files are UTF-8 clean

Limited support for file carving:

- packets carved into pcap files
  - —*(IPv4 and IPv6)*
- VCARD carver

New Histogram options:

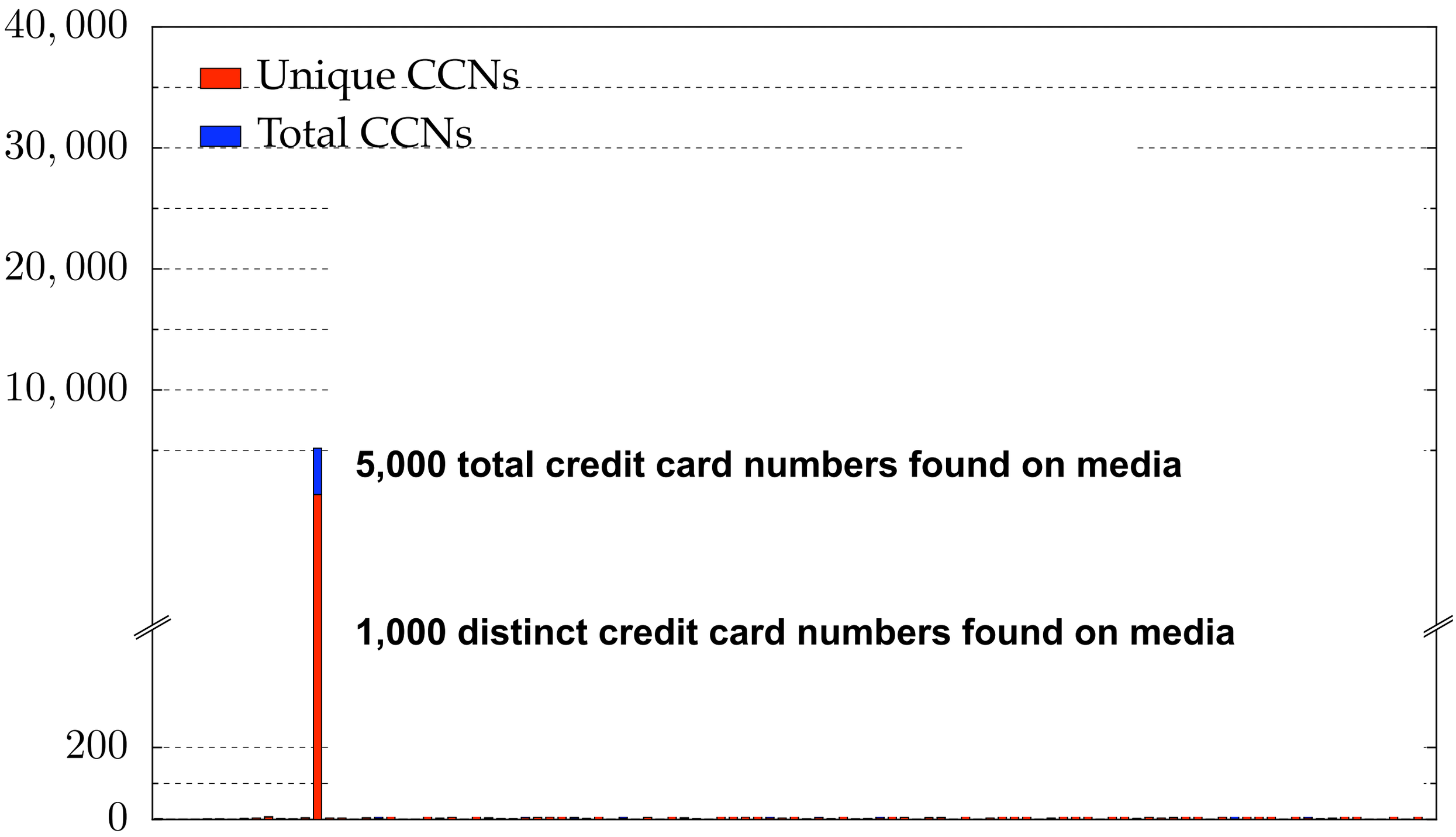
- Numeric only option for phone numbers
- Supports new Facebook ID

More Reliability:

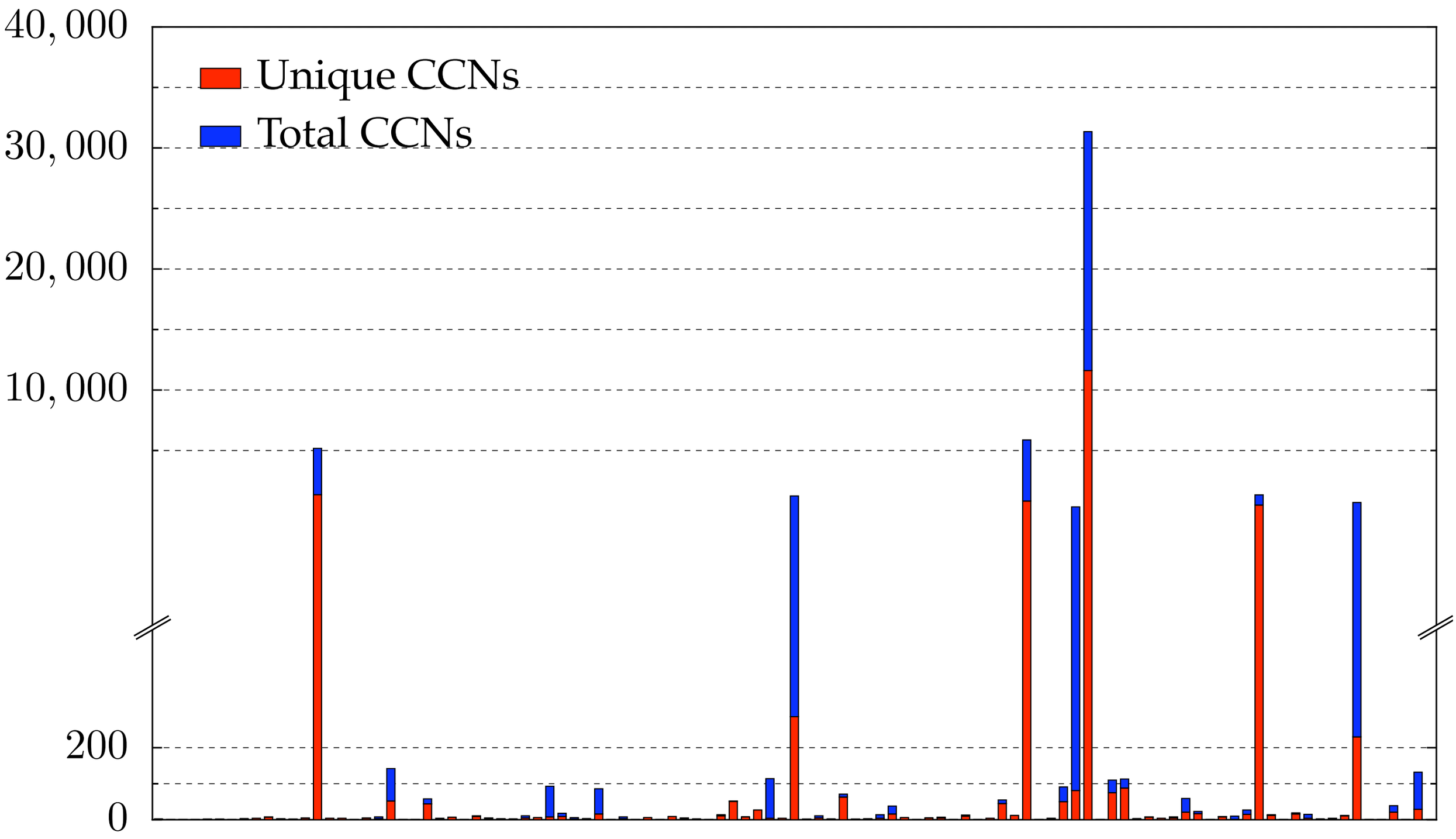
- Most scanners now use sbuf_t for brokered access to subject data.
- bulk_extractor now tolerant of libewf read errors.
- plug-in system now catches C++ exceptions.

Introducing Cross-Drive Analysis

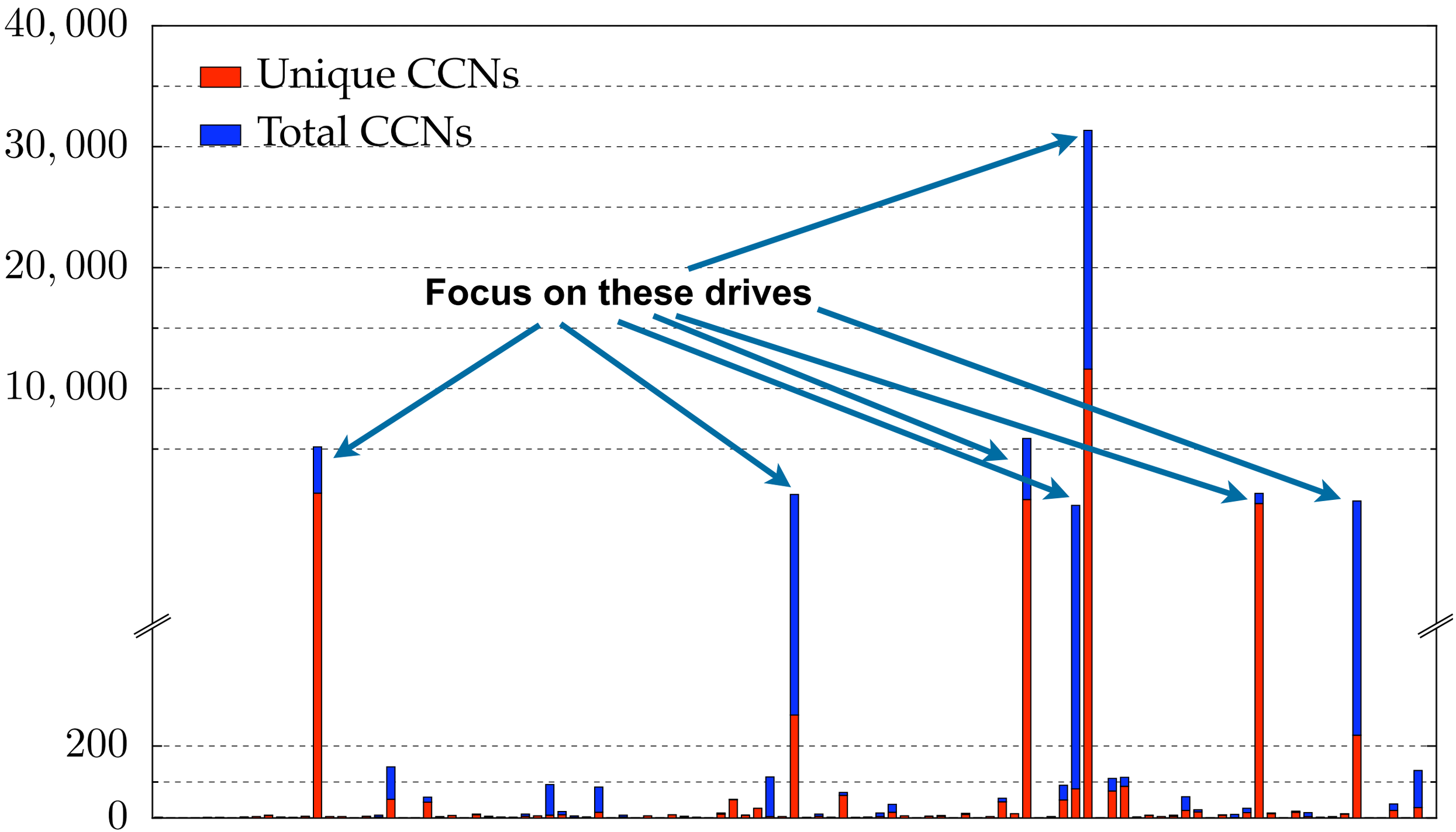
Frequently we get data from evidence and don't know how to interpret it.



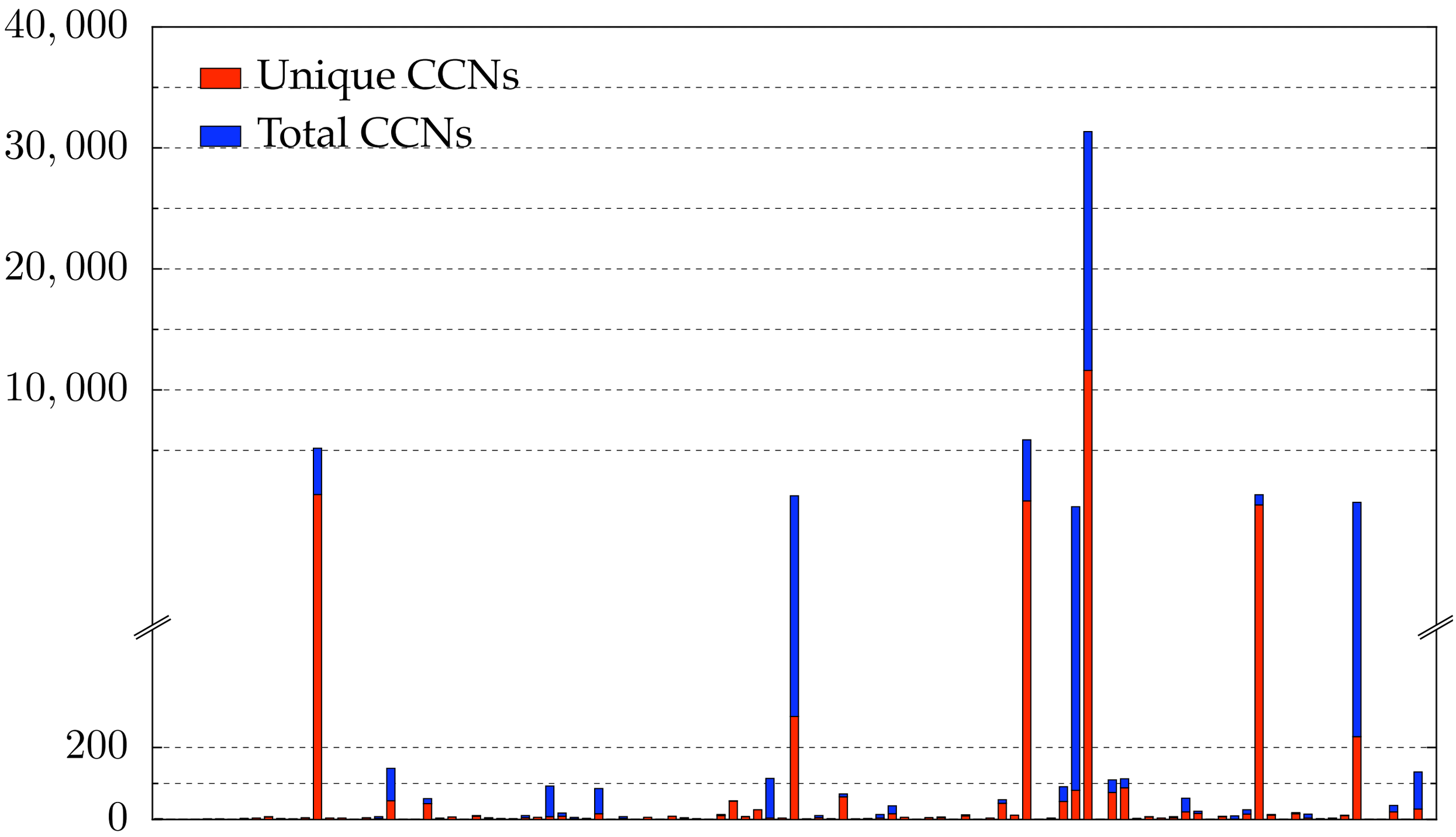
First order cross drive analysis:
Use a collection of data to find outliers.



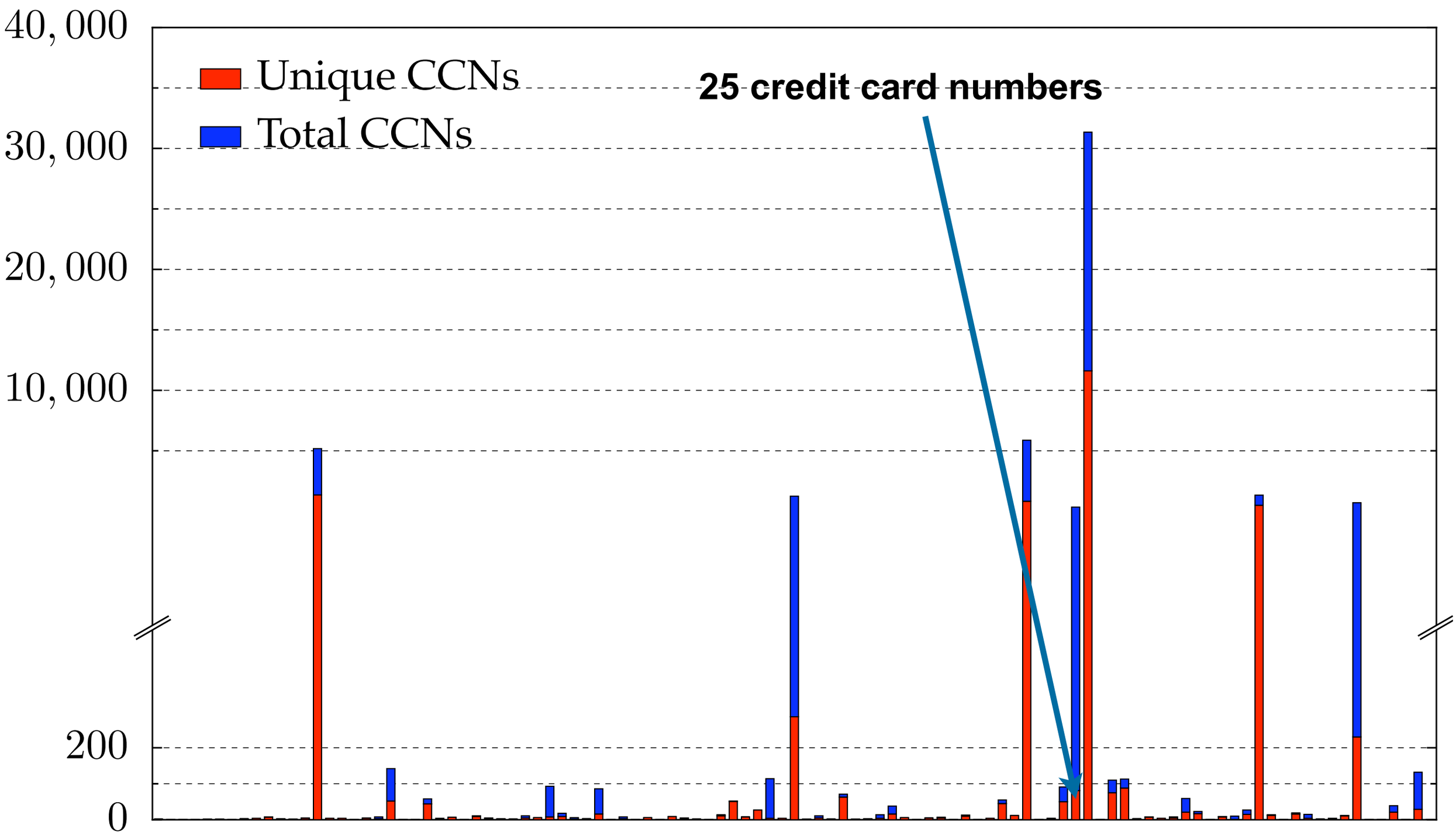
First order cross drive analysis:
Use a collection of data to find outliers.



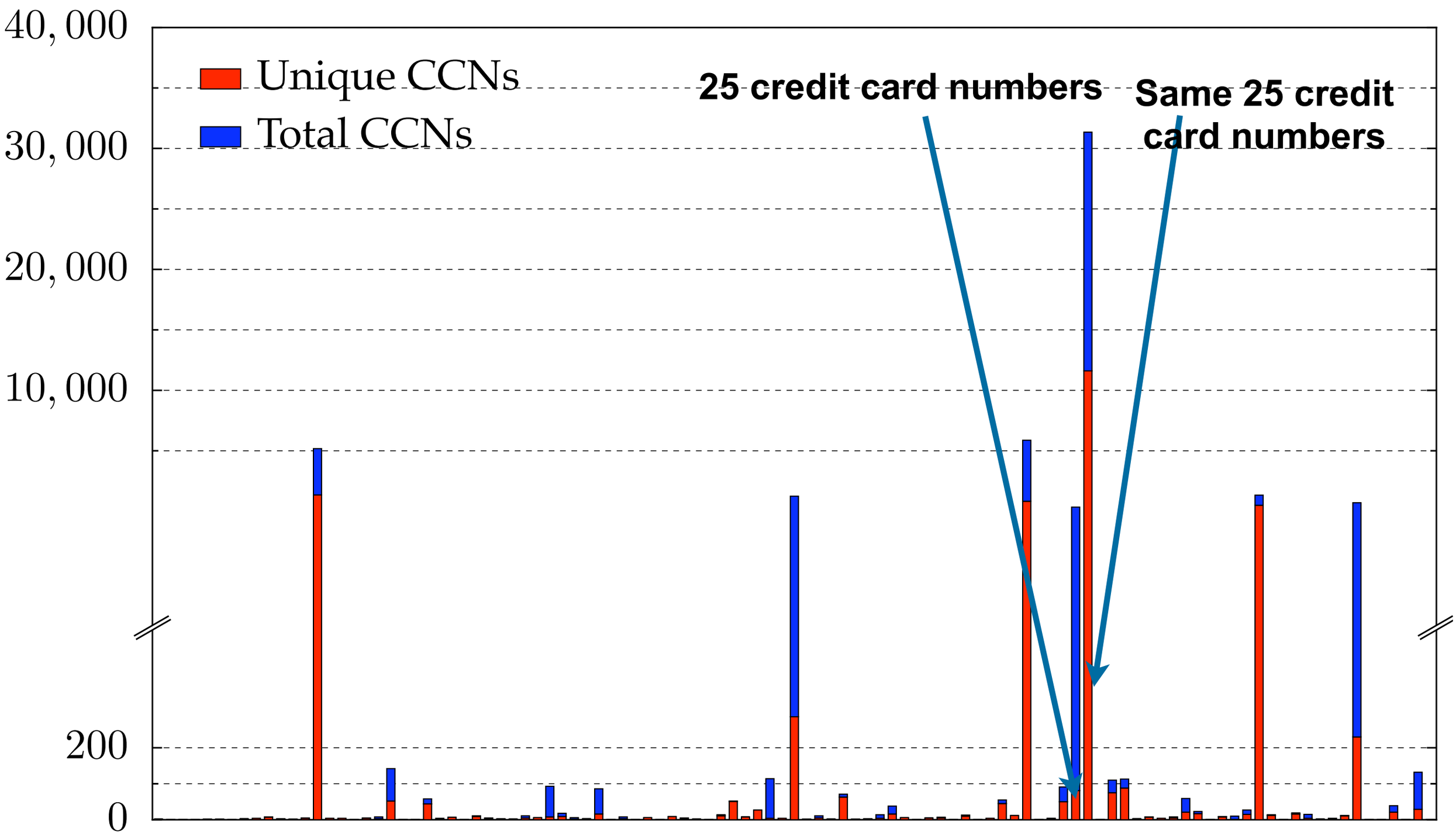
Second Order Cross Drive Analysis: Look for correlations between subject drives.



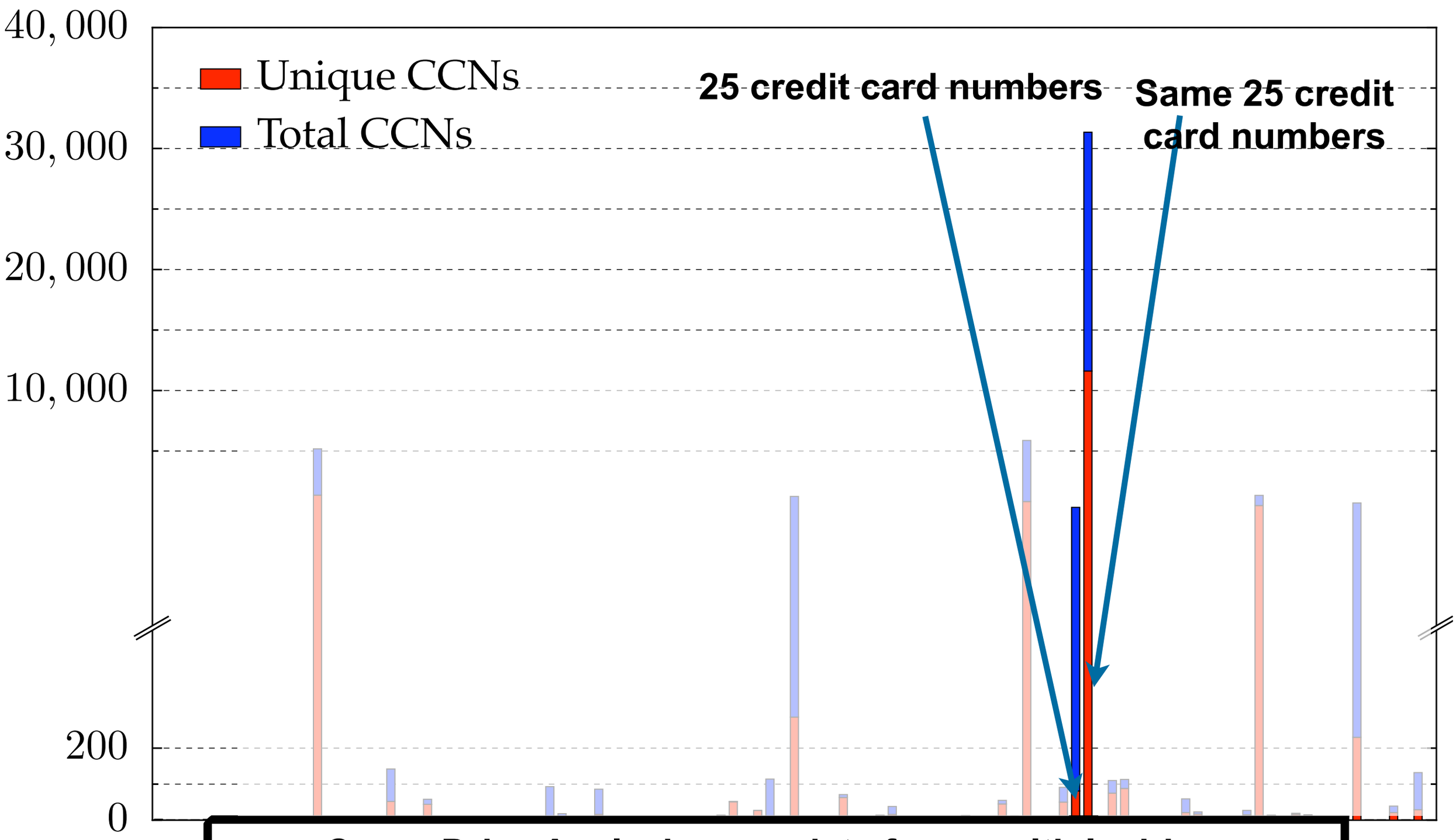
Second Order Cross Drive Analysis: Look for correlations between subject drives.



Second Order Cross Drive Analysis:
Look for correlations between subject drives.



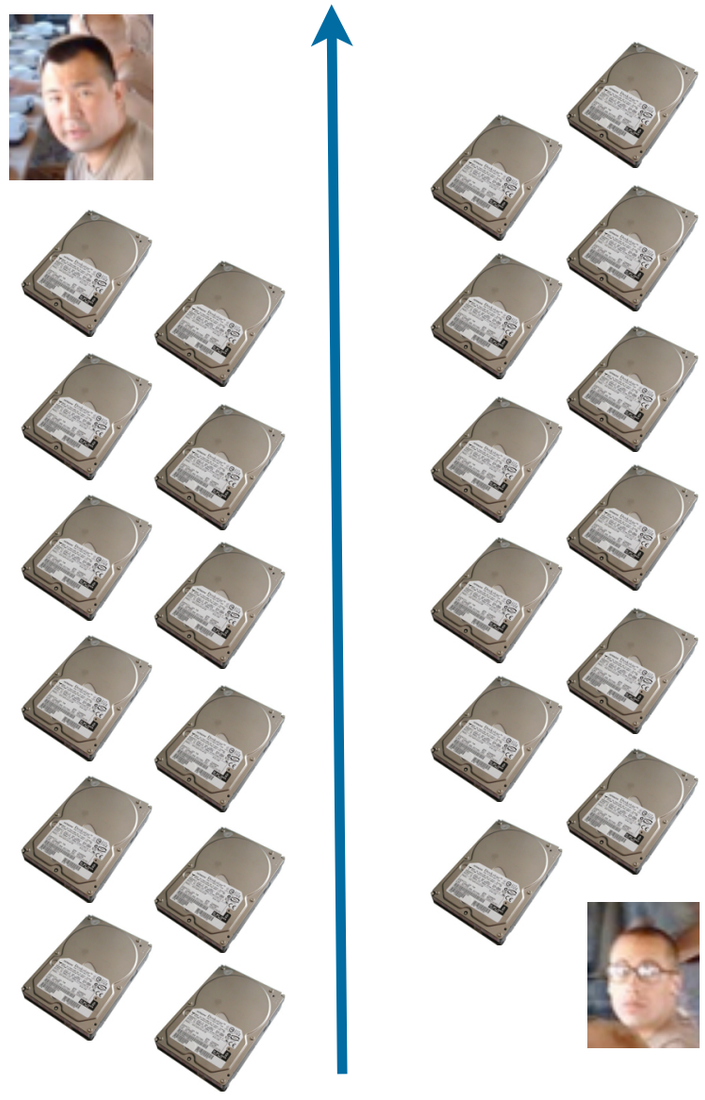
Second Order Cross Drive Analysis:
Look for correlations between subject drives.



Cross-Drive Analysis uses data from multiple drives to provide analysts with a better understanding of their targets.

Manual analysis misses opportunities for correlation.

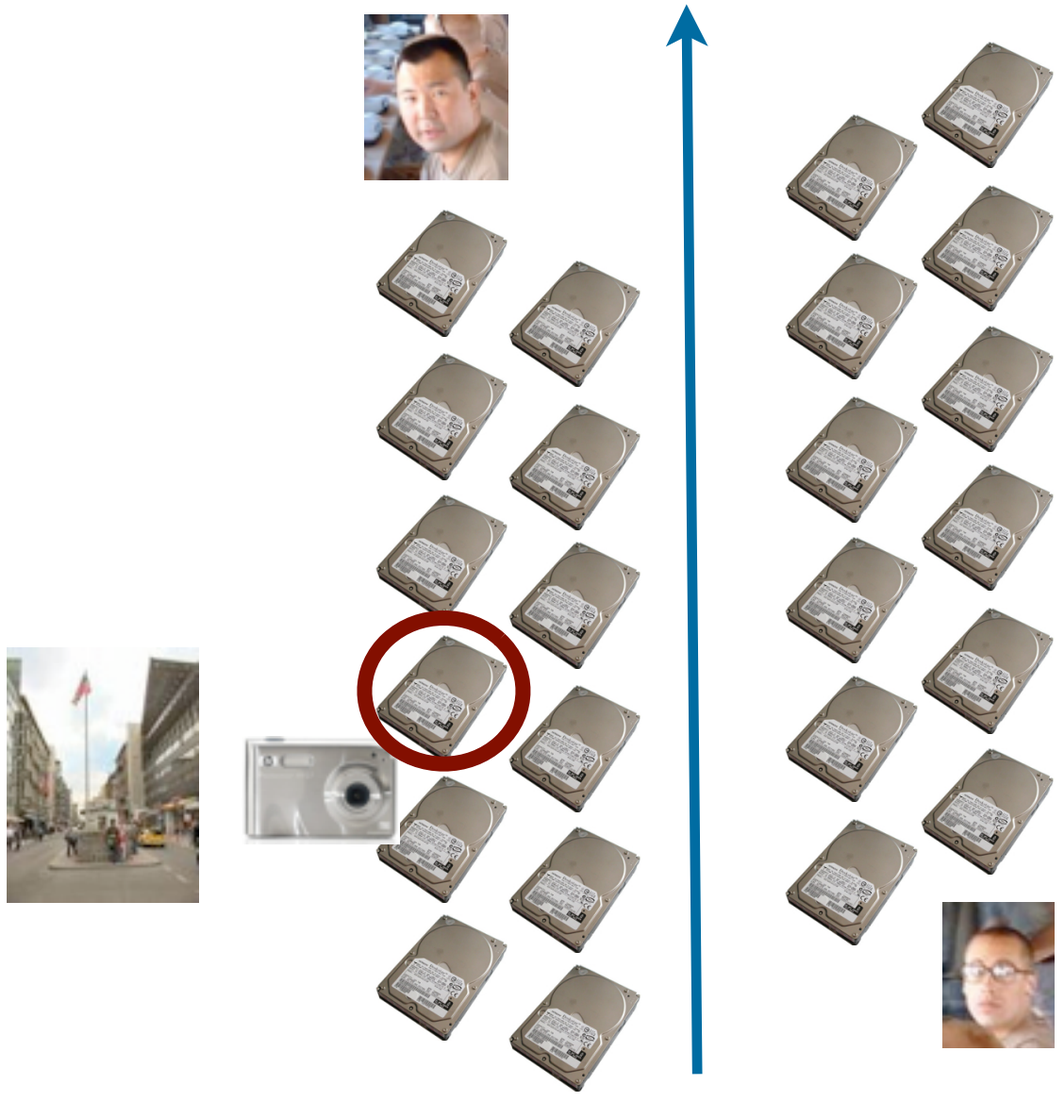
Different analysts see different hard drives.



Keyword searches don’t connect the dots.

Manual analysis misses opportunities for correlation.

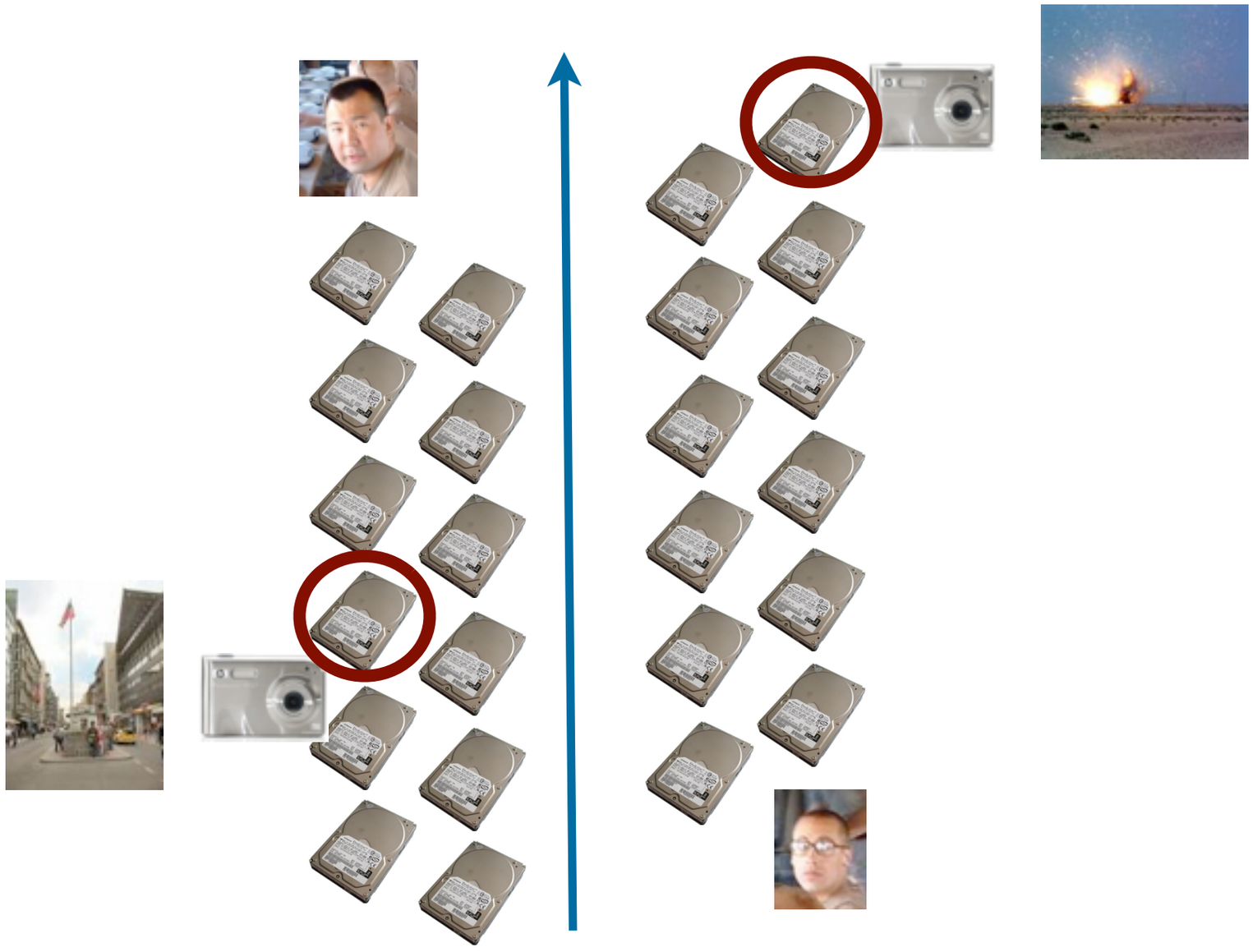
Different analysts see different hard drives.



Keyword searches don’t connect the dots.

Manual analysis misses opportunities for correlation.

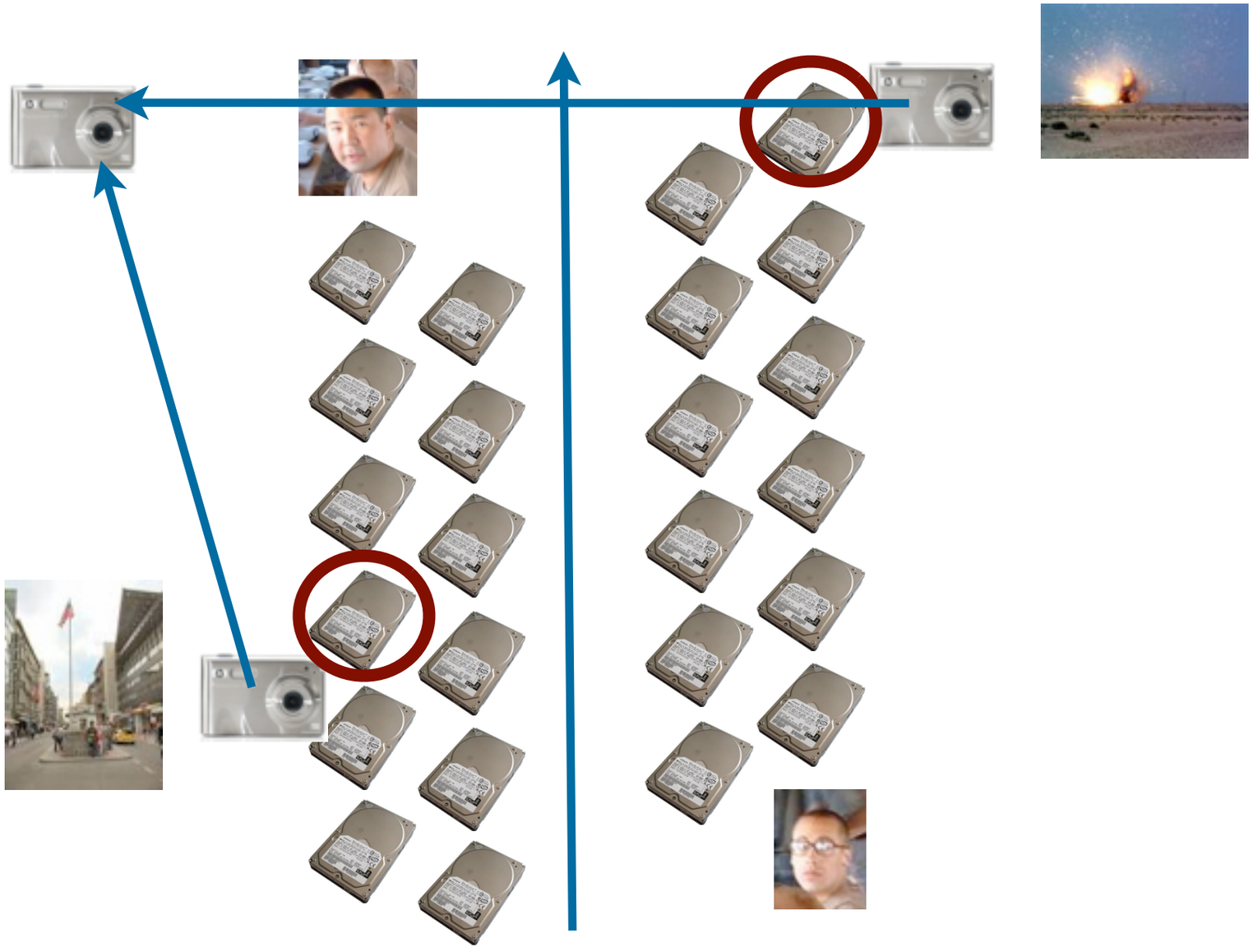
Different analysts see different hard drives.



Keyword searches don’t connect the dots.

Manual analysis misses opportunities for correlation.

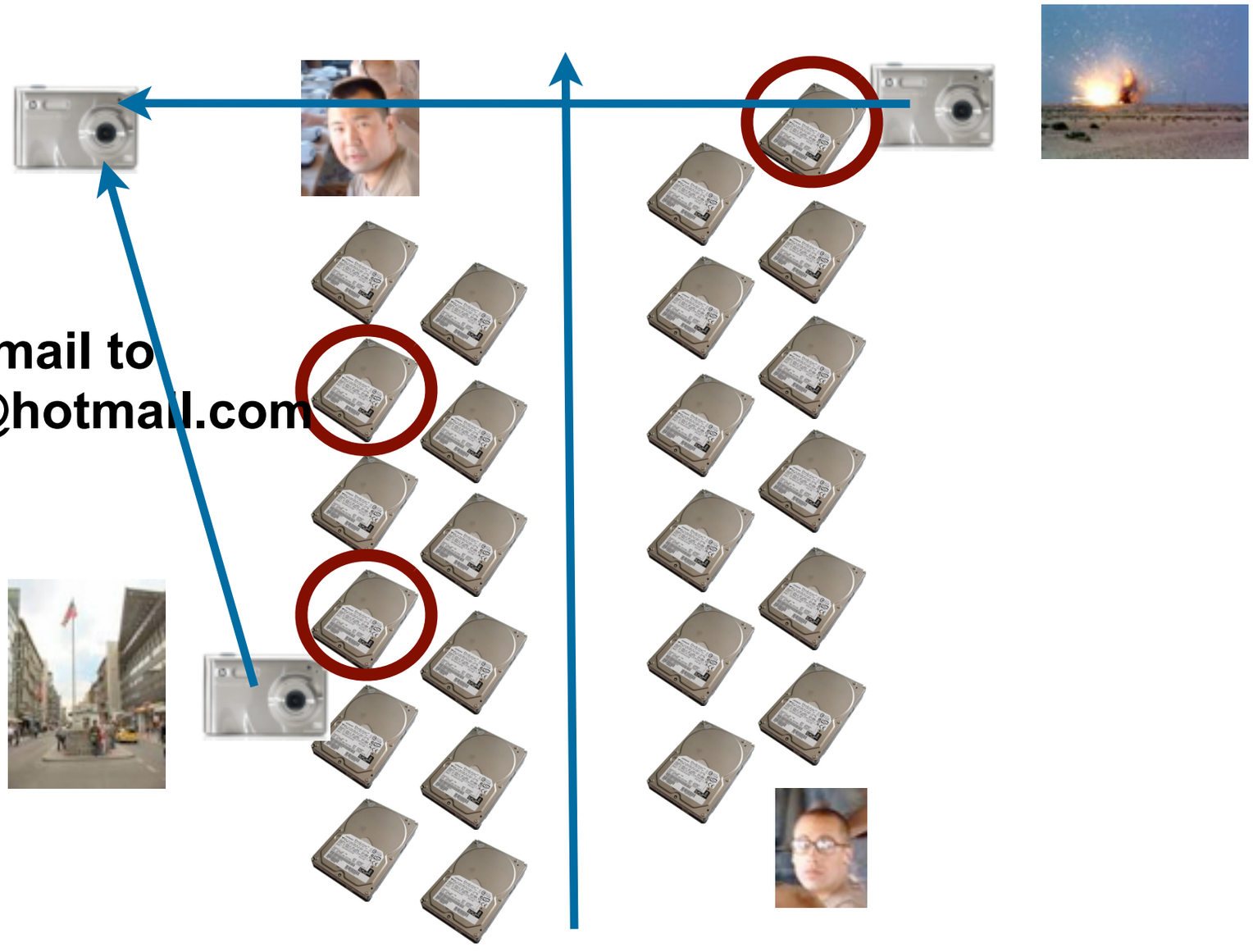
Different analysts see different hard drives.



Keyword searches don’t connect the dots.

Manual analysis misses opportunities for correlation.

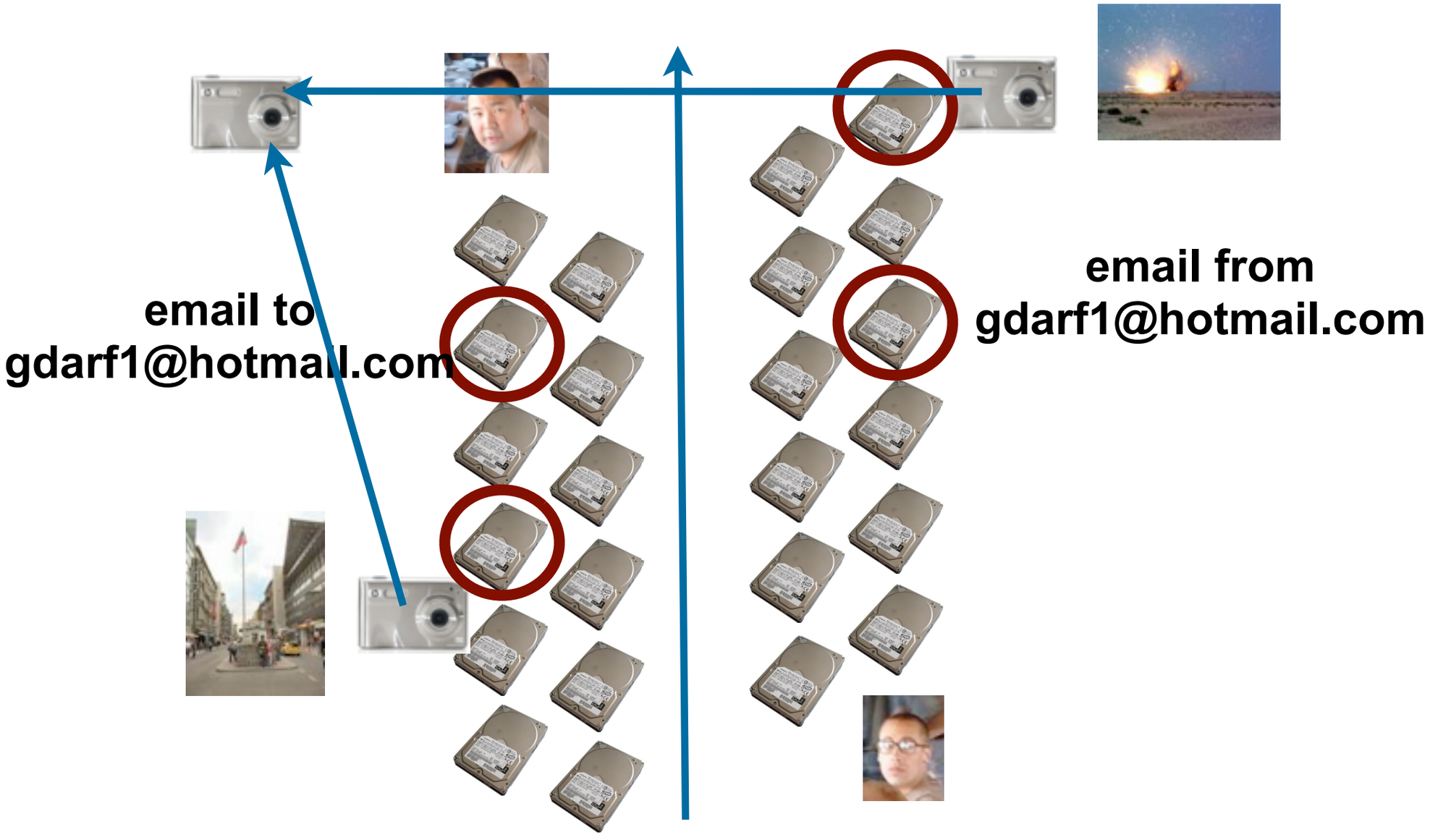
Different analysts see different hard drives.



Keyword searches don't connect the dots.

Manual analysis misses opportunities for correlation.

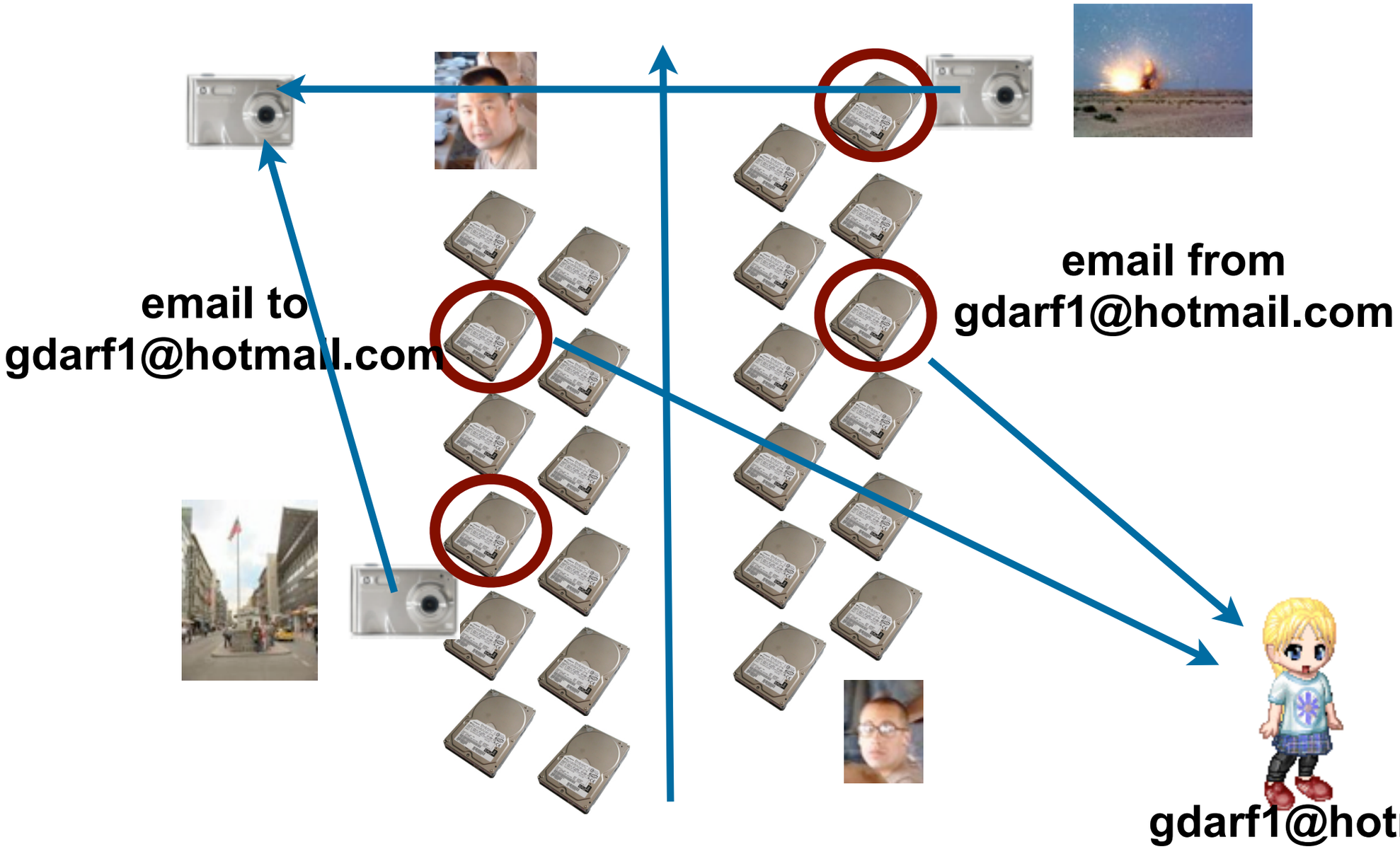
Different analysts see different hard drives.



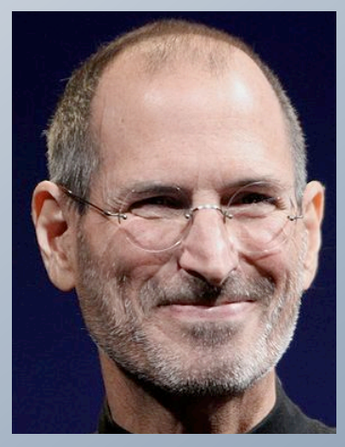
Keyword searches don’t connect the dots.

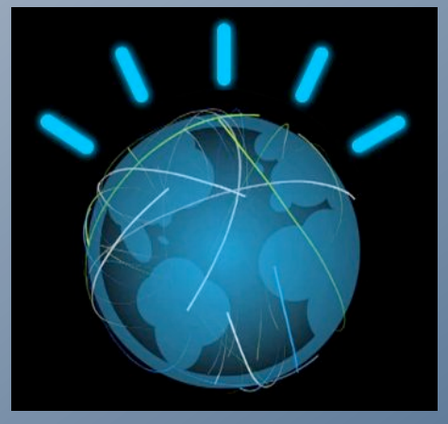
Manual analysis misses opportunities for correlation.

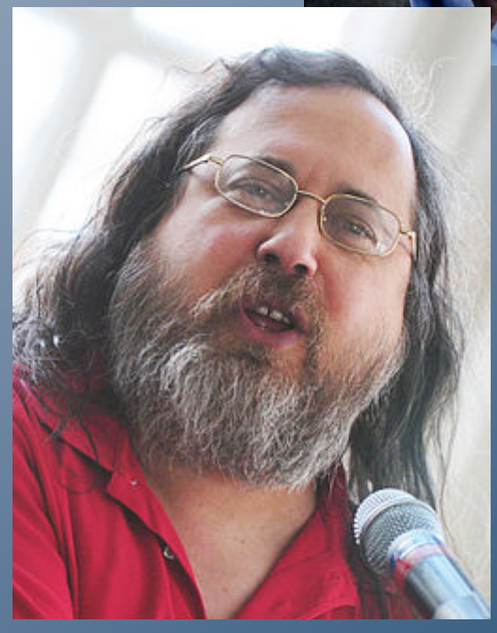
Different analysts see different hard drives.



Keyword searches don’t connect the dots.







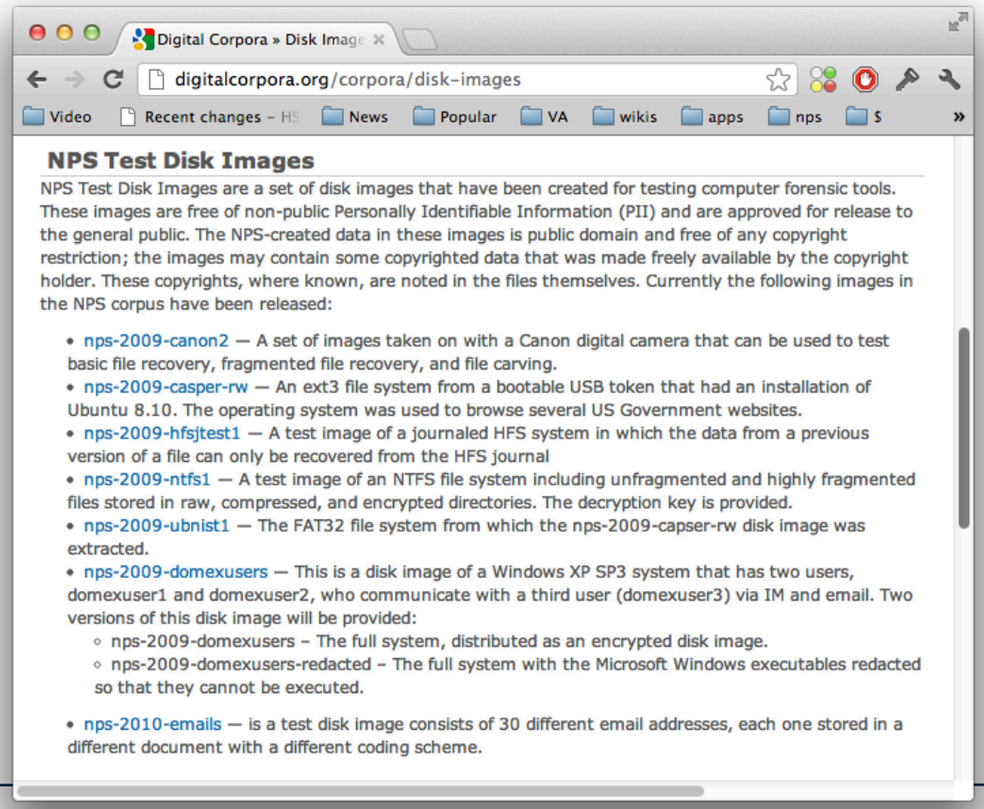
Suppressing bulk_extractor False Positives

For this section we will work with ubnist1

nps-2009-ubnist1 — Bootable Linux USB used to browse USG sites

```
-rw-rw-r--  1 simsong  staff         685 Jan 29  2009 narrative.txt
-rw-rw-r--  1 simsong  staff  2106589184 Jan  6  2009 ubnist1.gen0.raw
-rw-rw-r--  1 simsong  staff  2106589184 Jan  6  2009 ubnist1.gen1.raw
-rw-rw-r--  1 simsong  staff  2106589184 Jan  9  2009 ubnist1.gen2.raw
-rw-rw-r--  1 simsong  staff  2106589184 Jan  7  2009 ubnist1.gen3.raw
```

Four snapshots: gen0, gen1, gen2 & gen3



Hard drives are *filled* with email addresses.

Bulk_extractor finds email addresses is many places:

- Windows binaries; SSL certificates
- Documents
- Cached web pages; Memory; Hibernation Files

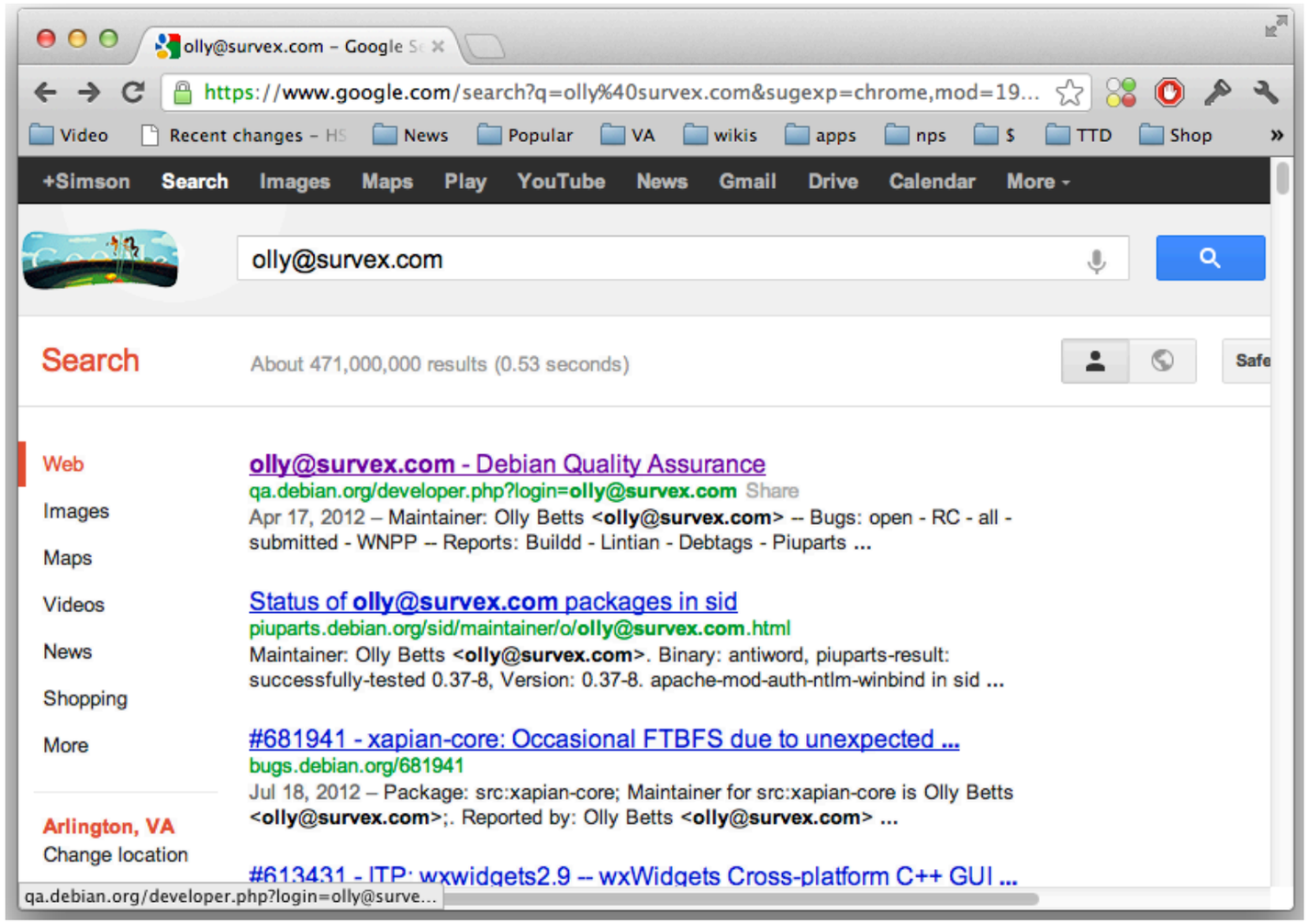
UBNIST1 have a LOT of email addresses; each snapshot sees more...

gen0	gen1	gen2	gen3
n=3447 olly@survex.com n=2237 hadess@hadess.net n=2040 daniel@veillard.com n=990 airlied@linux.ie n=910 cjwatson@debian.org n=909 dsandras@seconix.com n=893 rene@debian.org n=884 anholt@freebsd.org n=767 jirka@5z.com n=760 guillem@debian.org n=744 wakkerma@debian.org n=708 dshaw@jabberwocky.com n=660 doogie@debian.org n=659 brian.cameron@sun.com n=656 dsandras@gnome.org n=654 kyoshida@novell.com n=632 priikone@silcnet.org n=626 daniel@fooishbar.org ...	n=3455 olly@survex.com n=3254 ubuntu-devel-discuss@lists.ubuntu.com n=2241 hadess@hadess.net n=2040 daniel@veillard.com n=990 airlied@linux.ie n=930 cjwatson@debian.org n=910 rene@debian.org n=909 dsandras@seconix.com n=884 anholt@freebsd.org n=767 jirka@5z.com n=760 guillem@debian.org n=744 wakkerma@debian.org n=708 dshaw@jabberwocky.com n=660 doogie@debian.org n=659 brian.cameron@sun.com n=656 dsandras@gnome.org n=654 kyoshida@novell.com ...	n=27364 ubuntu-users@lists.ubuntu.com n=17213 ubuntu-motu@lists.ubuntu.com n=14291 ubuntu-devel-discuss@lists.ubuntu.com n=4086 language-packs@ubuntu.com n=3481 olly@survex.com n=2280 ubuntu-desktop@lists.ubuntu.com n=2239 hadess@hadess.net n=2040 daniel@veillard.com n=1696 debian-x@lists.debian.org n=1202 strk@keybit.net n=1122 bwy@benjaminwolsey.de n=1044 pkg-perl-maintainers@lists.alioth.debian.org n=1036 cjwatson@debian.org n=1017 rene@debian.org n=990 airlied@linux.ie ...	n=27672 ubuntu-users@lists.ubuntu.com n=17133 ubuntu-motu@lists.ubuntu.com n=12936 ubuntu-devel-discuss@lists.ubuntu.com n=4032 language-packs@ubuntu.com n=3477 olly@survex.com n=2239 hadess@hadess.net n=2040 daniel@veillard.com n=1966 ubuntu-desktop@lists.ubuntu.com n=1484 debian-x@lists.debian.org n=1202 strk@keybit.net n=1122 bwy@benjaminwolsey.de n=1023 cjwatson@debian.org n=1010 rene@debian.org n=1006 pkg-perl-maintainers@lists.alioth.debian.org n=990 airlied@linux.ie ...
6596 total	**6929 total**	**8734 total**	**8734 total**

It's important to distinguish email addresses that are relevant to a case from those that are not.

The top address is olly@survex.com

- We should probably ignore Mr. Betts:



gen0	
n=3447	olly@survex.com
n=2237	hadess@hadess.net
n=2040	daniel@veillard.com
n=990	airlied@linux.ie
n=910	cjwatson@debian.org
n=909	dsandras@seconix.com
n=893	rene@debian.org
n=884	anholt@freebsd.org
n=767	jirka@5z.com
n=760	guillem@debian.org
n=744	wakkerma@debian.org
n=708	dshaw@jabberwocky.com
n=660	doogie@debian.org
n=659	brian.cameron@sun.com
n=656	dsandras@gnome.org
n=654	kyoshida@novell.com
n=632	priikone@silcnet.org
n=626	daniel@fooishbar.org
6596 total	

Other sources that we might wish to ignore

- Windows binaries; SSL certificates; Sample documents; News Stories

stop lists specify features to be "stopped"

Stopped features *are not ignored!*

- Stopped features are moved from *email.txt* to *email_stopped.txt*.
- This is important for validation and error-diagnosis.

Stop lists are implemented with the `word_and_context_list` class.

- "words" — Anything that might be in the "feature" column
  - —*Email address*
  - —*MD5 hash of the first 4KiB of a file (JPEG)*
  - —*AES key*
- "context" — Anything that might be in the "context" column
  - —*Includes the feature*
  - —*Will suppress a specific instance of a feature*
- regular expressions
  - —*Dramatically slows down the process*

bulk_extractor also supports *alert lists*

- "words" or "context" that should be flagged

stop lists and alert lists are specified with the “-w” and “-r” options.

```
Usage: bulk_extractor [options] imagefile

...
   -r alert_list.txt  - a file containing the alert list of features to alert
                        (can be a feature file or a list of globs)
                        (can be repeated.)
   -w stop_list.txt   - a file containing the stop list of features (white list
                        (can be a feature file or a list of globs)s
                        (can be repeated.)
```

The list can be a list of words or a feature file

- For example
  - *stop_list.txt:*

```
olly@survex.com
hadess@hadess.net
daniel@veillard.com
```

  - *alert_list.txt:*

```
daniel@fooishbar.org
```

- command to use is:

```
bulk_extractor -r alert_list.txt -w stop_list.txt -o ubnist1.gen0-v1 ubnist1.gen0.raw
```

Stop lists processing is reflected in the feature files.

No stop list:

```
7324005 Aug  4 10:25 ubnist1.gen0/email.txt
 169609 Aug  4 10:25 ubnist1.gen0/email_histogram.txt

$ wc -l ubnist1.gen0/email*
   72965 ubnist1.gen0/email.txt
    6596 ubnist1.gen0/email_histogram.txt
```

With stop list:

```
6566160 Aug  4 11:20 email.txt
 169534 Aug  4 11:20 email_histogram.txt
 827940 Aug  4 11:20 email_stopped.txt

$ wc -l ubnist1.gen0-v1/email*
   65241 ubnist1.gen0-v1/email.txt
    6593 ubnist1.gen0-v1/email_histogram.txt
    7729 ubnist1.gen0-v1/email_stopped.txt

$ head ubnist1.gen0-v1/email_stopped.txt
# UTF-8 Byte Order Marker; see http://unicode.org/faq/utf_bom.html
# Filename: /corp/nps/drives/nps-2009-ubnist1/ubnist1.gen0.raw
# Feature-Recorder: email_stopped
# Feature-File-Version: 1.1
317986809-GZIP-47       daniel@veillard.com  aniel Veillard <daniel@veillard.com>\x5Cx0A
\x5Cx0A\x5Cx09* configure.
317986809-GZIP-209      daniel@veillard.com  aniel Veillard <daniel@veillard.com>\x5Cx0A
\x5Cx0A\x5Cx09* libxslt/xs
317986809-GZIP-635      daniel@veillard.com aniel Veillard <daniel@veillard.com>\x5Cx0A
```

—*stop_list.txt:*

```
olly@survex.com
hadess@hadess.net
daniel@veillard.com
```

—*alert_list.txt:*

```
daniel@fooishbar.org
```

Here is more of the email stopped list:

```
317986809-GZIP-47     daniel@veillard.com    aniel Veillard <daniel@veillard.com>\x5Cx0A\x5Cx0A\x5Cx09* configure.
317986809-GZIP-209    daniel@veillard.com    aniel Veillard <daniel@veillard.com>\x5Cx0A\x5Cx0A\x5Cx09* libxslt/xs
317986809-GZIP-635    daniel@veillard.com    aniel Veillard <daniel@veillard.com>\x5Cx0A\x5Cx0A\x5Cx09* libxslt/do
317986809-GZIP-1211   daniel@veillard.com    aniel Veillard <daniel@veillard.com>\x5Cx0A\x5Cx0A\x5Cx09* doc/xsltpr
317986809-GZIP-1377   daniel@veillard.com    aniel Veillard <daniel@veillard.com>\x5Cx0A\x5Cx0A\x5Cx09* libxslt/pa
317986809-GZIP-1581   daniel@veillard.com    aniel Veillard <daniel@veillard.com>\x5Cx0A\x5Cx0A\x5Cx09* configure.
317986809-GZIP-1692   daniel@veillard.com    aniel Veillard <daniel@veillard.com>\x5Cx0A\x5Cx0A\x5Cx09* libexslt/d
317986809-GZIP-1824   daniel@veillard.com    aniel Veillard <daniel@veillard.com>\x5Cx0A\x5Cx0A\x5Cx09* python/gen
317986809-GZIP-1943   daniel@veillard.com    aniel Veillard <daniel@veillard.com>\x5Cx0A\x5Cx0A\x5Cx09* libxslt/xs
317986809-GZIP-2085   daniel@veillard.com    aniel Veillard <daniel@veillard.com>\x5Cx0A\x5Cx0A\x5Cx09* libexslt/d
317986809-GZIP-2315   daniel@veillard.com    aniel Veillard <daniel@veillard.com>\x5Cx0A\x5Cx0A\x5Cx09* doc/xsltpr
317986809-GZIP-2724   daniel@veillard.com    aniel Veillard <daniel@veillard.com>\x5Cx0A\x5Cx0A\x5Cx09* libxslt/wi
317986809-GZIP-2940   daniel@veillard.com    aniel Veillard <daniel@veillard.com>\x5Cx0A\x5Cx0A\x5Cx09* doc/xsltpr
317986809-GZIP-3331   daniel@veillard.com    aniel Veillard <daniel@veillard.com>\x5Cx0A\x5Cx0A\x5Cx09* libxslt/xs
317986809-GZIP-3494   daniel@veillard.com    aniel Veillard <daniel@veillard.com>\x5Cx0A\x5Cx0A\x5Cx09* python/tes
317986809-GZIP-3647   daniel@veillard.com    aniel Veillard <daniel@veillard.com>\x5Cx0A\x5Cx0A\x5Cx09* doc/xslt.h
...
```

```
330031689-GZIP-275885   olly@survex.com 07  Olly Betts <olly@survex.com>\x5Cx0A\x5Cx0A\x5Cx09* queryparse
330031689-GZIP-276902   olly@survex.com 07  Olly Betts <olly@survex.com>\x5Cx0A\x5Cx0A\x5Cx09* tests/api_
330031689-GZIP-277660   olly@survex.com 07  Olly Betts <olly@survex.com>\x5Cx0A\x5Cx0A\x5Cx09* configure.
330031689-GZIP-277778   olly@survex.com 07  Olly Betts <olly@survex.com>\x5Cx0A\x5Cx0A\x5Cx09* tests/harn
330031689-GZIP-277951   olly@survex.com 07  Olly Betts <olly@survex.com>\x5Cx0A\x5Cx0A\x5Cx09* net/remote
330031689-GZIP-278061   olly@survex.com 07  Olly Betts <olly@survex.com>\x5Cx0A\x5Cx0A\x5Cx09* tests/harn
330031689-GZIP-278198   olly@survex.com 07  Olly Betts <olly@survex.com>\x5Cx0A\x5Cx0A\x5Cx09* tests/harn
330031689-GZIP-278379   olly@survex.com 07  Olly Betts <olly@survex.com>\x5Cx0A\x5Cx0A\x5Cx09* tests/harn
330031689-GZIP-278529   olly@survex.com 07  Olly Betts <olly@survex.com>\x5Cx0A\x5Cx0A\x5Cx09* configure.
330031689-GZIP-278655   olly@survex.com 07  Olly Betts <olly@survex.com>\x5Cx0A\x5Cx0A\x5Cx09* net/progcl
330031689-GZIP-278783   olly@survex.com 07  Olly Betts <olly@survex.com>\x5Cx0A\x5Cx0A\x5Cx09* net/remote
```

A feature file can be used as a context stop list. We can use the gen0 email as a stop for gen1

The gen0 feature file becomes a *filter:*

This isn't hard to do in practice:

```
$ src/bulk_extractor  -w ubnist1.gen0/email.txt  -o ubnist1.gen1-filtered_by_gen0 ubnist1.gen1.raw
Reading context stop list ubnist1.gen0/email.txt
Stop list read.
  Total features read: 72961
  List Size: 32084
  Context Strings: 32083
  Regular Expressions: 0
```

The result of filtering gen1 with gen0: We only see the new *uses* of email addresses

```
8424165 Aug  4 10:24 ubnist1.gen1/email.txt
 179549 Aug  4 10:24 ubnist1.gen1/email_histogram.txt
```

Features in gen1

−

```
7324005 Aug  4 10:25 ubnist1.gen0/email.txt
 169609 Aug  4 10:25 ubnist1.gen0/email_histogram.txt
```

Features in gen0

=

```
1083873 Aug  4 11:49 ubnist1.gen1-filtered_by_gen0/email.txt
  13867 Aug  4 11:50 ubnist1.gen1-filtered_by_gen0/email_histogram.txt
7805523 Aug  4 11:49 ubnist1.gen1-filtered_by_gen0/email_stopped.txt
```

Features only in gen1

The resulting histogram is a histogram of the filtered email.txt:

```
n=3178   ubuntu-devel-discuss@lists.ubuntu.com
n=638    ubuntu-desktop@lists.ubuntu.com
n=444    debian-x@lists.debian.org
n=219    debian-boot@lists.debian.org
n=204    ubuntu-motu@lists.ubuntu.com
n=144    seb128@debian.org
n=132    pkg-gnome-maintainers@lists.alioth.debian.org
```

Stop lists and alert lists have minor impact on performance.

Longer stop lists are slower to process. t ∝ (features x stops)

lines in stop list	ubnist1.gen0 Execution time
0	**54.51 s**
3	**55.47 s**
235,886	**55.41 s**

Regular expressions slow down stop lists dramatically:

```
olly@survex.com
hadess@hadess.net
daniel@veillard.com
```

```
.*@survex.com
.*@hadess.net
.*@veillard.com
```

re lines in stop list	ubnist1.gen0 Execution time
0	**54.51 s**
3	**66.41 s**

Stop and alert lists must be applied when bulk_extractor is run.

- A future version may allow filtering after-the-fact.

Context-sensitive stop lists are important when looking for unknown individuals.

Recall all of those email addresses on ubnist1

Although these emails are widely seen, they should not be whitelisted:

- Email addresses can be shared
- Email addresses can be sold
- A Linux developer might be engaged in a criminal enterprise

gen0	
n=3447	olly@survex.com
n=2237	hadess@hadess.net
n=2040	daniel@veillard.com
n=990	airlied@linux.ie
n=910	cjwatson@debian.org
n=909	dsandras@seconix.com
n=893	rene@debian.org
n=884	anholt@freebsd.org
n=767	jirka@5z.com
n=760	guillem@debian.org
n=744	wakkerma@debian.org
n=708	dshaw@jabberwocky.com
n=660	doogie@debian.org
n=659	brian.cameron@sun.com
n=656	dsandras@gnome.org
n=654	kyoshida@novell.com
n=632	priikone@silcnet.org
n=626	daniel@fooishbar.org
6596 total	

By using context-sensitive stop lists, we:

- Ignore the email address where it is widely seen
- Will still notice the email address in a new context

Context-sensitive lists need to be maintained!

- Build from default installs of operating systems & applications.
- NIST is running bulk_extractor over the entire NSRL and will make the results available.
- Organizations are free to trade the feature files amongst themselves.

```
$ python cda_tool.py --help
usage: cda_tool.py [-h] [--netmap] [--idcor] [--makestop MAKESTOP]
                   [--threshold THRESHOLD] [--idfeatures IDFEATURES]
                   reports [reports ...]

Cross Drive Analysis with bulk_extractor output

positional arguments:
  reports               bulk_extractor report directories or ZIP files

optional arguments:
  -h, --help            show this help message and exit
  --netmap              General GraphViz data for network correlation map
  --idcor               Perform identity-based correlation
  --makestop MAKESTOP   Make a stop list of identity features on more than
                        THRESHOLD drives
  --threshold THRESHOLD
                        Specify the faction of drives for the threshold
  --idfeatures IDFEATURES
                        Specifies feature files used for identity operations
$
```

cda_tool.py

cda_tool.py: a platform for cross-drive analysis

cda_tool.py is a python program that:

- Reads 2 or more bulk_extractor reports
- Performs multi-drive correlation
- Outputs the results in a format that can be used for additional processing

Cross-Drive correlation depends on two Python classes:

- `bulk_extractor_reader.BulkReport`
  - *Reads the directory or a ZIP file of the directory.*
- `cda_tool.Correlator`
  - *Performs the actual correlation*

Correlation is performed with feature *histograms.* *(This means that local context is ignored.)*

Start with drives, features and counts:

gen0	gen1
n=3447 olly@survex.com	n=3455 olly@survex.com
n=2237 hadess@hadess.net	n=3254 ubuntu-devel-discuss@lists.ubuntu.com
n=2040 daniel@veillard.com	n=2241 hadess@hadess.net
n=990 airlied@linux.ie	n=2040 daniel@veillard.com
n=910 cjwatson@debian.org	n=990 airlied@linux.ie
n=909 dsandras@seconix.com	n=930 cjwatson@debian.org
n=893 rene@debian.org	n=910 rene@debian.org
n=884 anholt@freebsd.org	n=909 dsandras@seconix.com
n=767 jirka@5z.com	n=884 anholt@freebsd.org
n=760 guillem@debian.org	n=767 jirka@5z.com
n=744 wakkerma@debian.org	n=760 guillem@debian.org
n=708 dshaw@jabberwocky.com	n=744 wakkerma@debian.org
n=660 doogie@debian.org	n=708 dshaw@jabberwocky.com
n=659 brian.cameron@sun.com	n=660 doogie@debian.org
n=656 dsandras@gnome.org	n=659 brian.cameron@sun.com
n=654 kyoshida@novell.com	n=656 dsandras@gnome.org
n=632 priikone@silcnet.org	n=654 kyoshida@novell.com
n=626 daniel@fooishbar.org	
...	...

Correlator produces per-drive counts

```
{
"olly@survex.com":
  {"gen0":3447,
   "gen1":3455},

"hadess@hadess.net":
  {"gen0":2237,
   "gen1":2241},

"airlied@linux.ie":
  {"gen0":990,
   "gen1":990},

 "ubuntu-devel-
discuss@lists.ubuntu.com",
  {"gen1":3254}
...
```

Use cda_tool is to create a stoplist of features present on more than f drives:

```
$ python cda_tool.py --help
usage: cda_tool.py [-h] [--netmap] [--idcor] [--makestop MAKESTOP]
                   [--threshold THRESHOLD] [--idfeatures IDFEATURES]
                   reports [reports ...]
...
positional arguments:
  reports               bulk_extractor report directories or ZIP files

optional arguments:
  -h, --help            show this help message and exit
  --netmap              General GraphViz data for network correlation map
  --idcor               Perform identity-based correlation
  --makestop MAKESTOP   Make a stop list of identity features on more than
                        THRESHOLD drives
  --threshold THRESHOLD
                        Specify the faction of drives for the threshold
  --idfeatures IDFEATURES
                        Specifies feature files used for identity operations
$
```

Example:

```
$ ./cda_tool.py --makestop stoplist.txt --threshold 3  --idfeatures email *.zip
```

M57-patents is a worked scenario on digitalcorpora.org

On my laptop I had a sample of reports from 2009-12-11:

```
$ ls -l current/tutorial/
total 289188
-rw-r--r--  1  22484045 Jul 10 11:16 charlie-2009-12-11.zip
-rw-r--r--  1     52194 Jul 10 11:16 charlie-work-usb-2009-12-11.zip
-rw-r--r--  1  18719633 Jul 10 11:19 jo-2009-12-11-002.zip
-rw-r--r--  1   2702376 Jul 10 11:19 jo-work-usb-2009-12-11.zip
-rw-r--r--  1  81798305 Jul 10 11:40 terry-2009-12-10.zip
-rw-r--r--  1  81734461 Jul 10 11:41 terry-2009-12-11-001.zip
-rw-r--r--  1  82532973 Jul 10 11:42 terry-2009-12-11-002.zip
-rw-r--r--  1   6089981 Jul 10 11:42 terry-work-usb-2009-12-11.zip
```

We can use cda_tool.py to create a stoplist of identity features on more than 50% of the drives:

```
$ python cda_tool.py --makestop stoplist.txt --threshold 0.5 *.zip
Scanning  /corp/nps/drives/nps-2009-m57-patents-redacted/terry-work-
usb-2009-12-11.E01 for email
Scanning  /corp/nps/drives/nps-2009-m57-patents-redacted/terry-2009-12-11-002.E01
for email
Scanning  /corp/nps/drives/nps-2009-m57-patents-redacted/terry-2009-12-11-002.E01
for telephone
Scanning  /corp/nps/drives/nps-2009-m57-patents-redacted/jo-2009-12-11-002.E01
for ccn
...
Stoplist stoplist.txt created with 763 features
   DPF   Feature Count
     1          733
     2          345
     3          752
     4           52
     5          702
     6            9
---------------------
DPF = Drives per Feature
Only features on 4 or more drives were written.
$
```

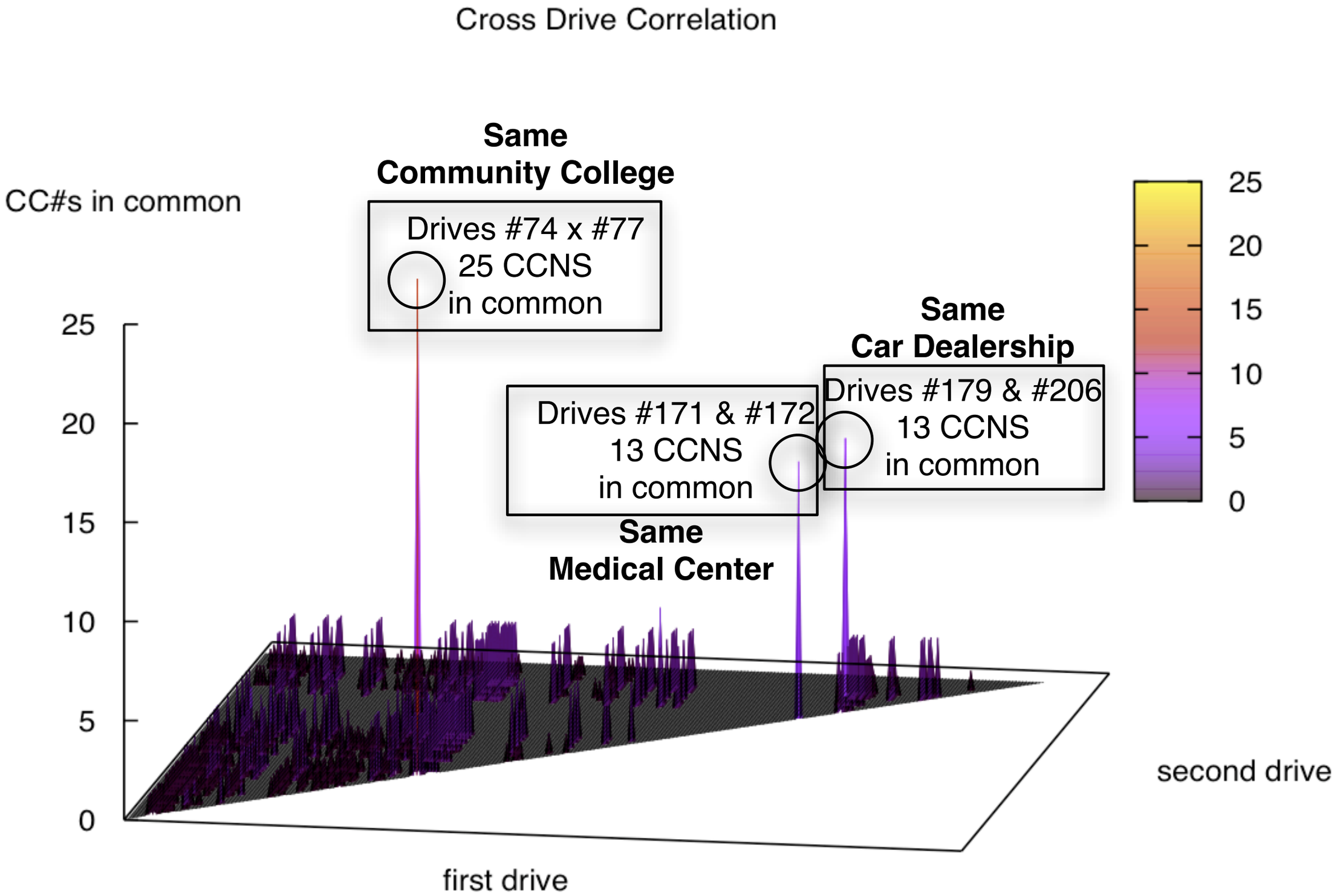
The result is a list of features written to stoplist.txt:

```
$ wc stoplist.txt
     763     944   18994 stoplist.txt
$ head -20 stoplist.txt
(000)-000-0000
(111) 555-4444
(222) 555-1111
(222) 555-2222
(506) 298-2000
(777) 777-7777
+254-20-3742774
+254-20-3744660
+254-20-3750230
+31 20 346 9363
+34 91 377 8170
+351 707 5000
+39 02 4586
+420 555 123321
+420 555 654321
+49 69 5007
+81-422-42-7633
0.0014F8A2@mail.groupcare.dk
001-800-2468
098 345.6044
```

This stoplist can be read with bulk_extractor's **-w** option.

Use cda_tool.py to find drive affinity...

Use cda_tool.py to find drive affinity...



We use the formula from Garfinkel 2006

Let:

$$
\begin{aligned}
D &= \#\text{ of drives} \\
F &= \#\text{ of extracted features} \\
d_0 \ldots d_D &= \text{Drives in corpus} \\
f_0 \ldots f_F &= \text{Extracted features} \\
FP(f_n, d_n) &= \begin{cases} 0 & f_n \text{ not present on } d_n \\ 1 & f_n \text{ present on } d_n \end{cases}
\end{aligned}
$$

$$DC(f) = \sum_{n=0}^{D} FP(f, d_n) = \#\text{ of drives with feature } f$$

$$S_2(d_1, d_2) = \sum_{n=0}^{F} \frac{FP(f_n, d_1) \times FP(f_n, d_2)}{DC(f_n)}$$

Use the cda_tool.py --idcor

```
python cda_tool.py  --idc  ~/current/tutorial/*.zip
Scanning  /corp/nps/drives/nps-2009-m57-patents-redacted/terry-work-usb-2009-12-11.E01 for email
Scanning  /corp/nps/drives/nps-2009-m57-patents-redacted/terry-work-usb-2009-12-11.E01 for telephone
Scanning  /corp/nps/drives/nps-2009-m57-patents-redacted/charlie-2009-12-11.E01 for ccn
Scanning  /corp/nps/drives/nps-2009-m57-patents-redacted/charlie-2009-12-11.E01 for email
Scanning  /corp/nps/drives/nps-2009-m57-patents-redacted/charlie-2009-12-11.E01 for telephone
...
```

cda_tool.py --idcor shows the drive affinity between each drive, and the features that contributed

```
Identity-based correlation: computes drive affinity using TF-IDF
Drive A:  /corp/nps/drives/nps-2009-m57-patents-redacted/terry-2009-12-11-001.E01
Drive B:  /corp/nps/drives/nps-2009-m57-patents-redacted/terry-2009-12-11-002.E01
Score  : 405.73333333332454
Top factors:
        0.3333  4890008118593969
        0.3333  5467663276676632
        0.3333  5540655860834388
        0.2  6543210123456788
        0.3333  5396180804577535

Drive A:  /corp/nps/drives/nps-2009-m57-patents-redacted/terry-2009-12-11-001.E01
Drive B:  /corp/nps/scenarios/2009-m57-patents/drives-redacted/terry-2009-12-10.E01
Score  : 400.73333333332465
Top factors:
        0.3333  4890008118593969
        0.3333  5467663276676632
        0.3333  5540655860834388
        0.2  6543210123456788
        0.3333  5396180804577535

...
```

Drive names come from BE report

No 3D graph yet (sorry!)

In conclusion: bulk_extractor 1.3: new features & working CDA!

New scanners designed for Windows and malware analysis:

- Windows PE Scanner
- Linux ELF Scanner
- VCARD Scanner
- BASE16 scanner
- Windows directory carver

Cross-Drive analysis in cda_tool.py

Download from https://github.com/simsong/bulk_extractor

- bulk_extractor1.3 in Downloads
- python and plugins moving to git repository!

Contact info:

- simsong@acm.org

Questions?