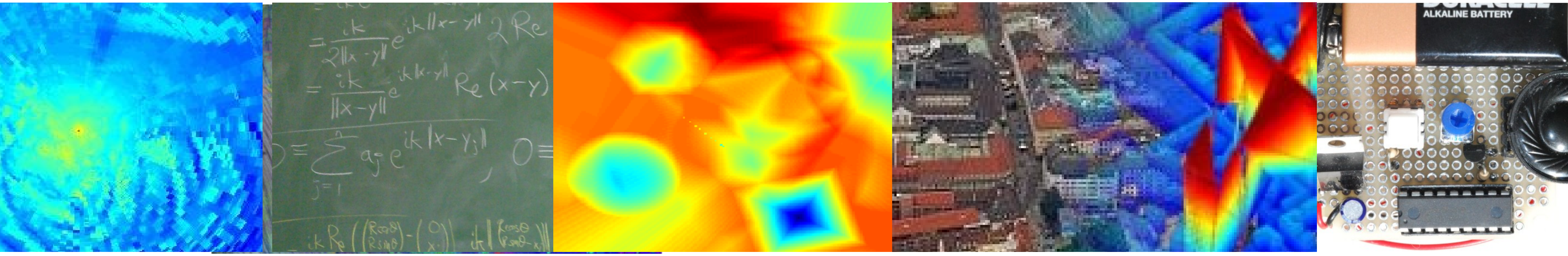
Statistical detection of format dialects *using the* weighted Dowker complex



Michael Robinson, Letitia Li,
Cory Anderson, Steve Hunstman

DISTRIBUTION STATEMENT A. Approved for Public Release

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001119C0072. Any opinions, findings and conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA.

Acknowledgments

- Additional collaborators:
  - Denley Lam (BAE Systems)
- Students:
  - Tate Altman, Ken Ewing, Natalie Tsuei (American Univ.)
- Data from:
  - Tim Allen (NASA/JPL)
  - Peter Wyatt (PDF Association)
- Funding:
  - Sergey Bratus (DARPA/I2O) SafeDocs

Problem definition

Suppose you have:

- A set of *files* supposedly complying with one of several dialects of a format
- A set of *parsers* for the format, that generate a set of error/status *messages* for each file

Questions:

- Is there consensus as to which files “really” comply with the format?
- Are the parsers detecting the same thing?
- Are there bugs in any given parser?
- Is the format well-specified? Even partially?

Key points

Statistics is for ensembles, even deterministic ones!

- Format consensus is an *ensemble property* of files and parsers **jointly**

Easy-to-use statistical tools:

1. The weighted Dowker complex for clustering files based on the pattern of messages they produce

2. Posterior probability computed for each pattern of messages

3. Classification via thresholding the posterior probability

Example consensus format: PDF

- The PDF format (ISO 32000-2) is written in “plain English”
  - Not machine parseable, no reference implementation
- Yet many *parsers* will consume PDF…
  - They make different choices where the format is ambiguous
  - When files are not compliant with the format, parsers often try to “fix” the file and continue
  - And all of them have bugs…
- **Takeaway:** Cannot trust any individual parser
  - An **ensemble** of many parsers should be generally more reliable than any single parser

Safely testing dangerous files

- Tactic: Confine the file and parser to an environment with very limited input/output access
- Representation: Use a *binary relation,* recording which files trigger which regexes from a fixed set
- Hypothesis: Anomalous files or parsers will manifest within the context of this relation

$$\begin{array}{ccc} & & \text{files} \\ \text{messages} & \begin{matrix} A \\ B \\ C \\ D \end{matrix} & \begin{pmatrix} 10000011111100001111 \\ 01100011100010000000 \\ 00011000010011111111 \\ 00000100001101111111 \end{pmatrix} \end{array}$$

1 = regex match
0 = no match

It's probably the case that there are many more files than messages, but this isn't terribly crucial

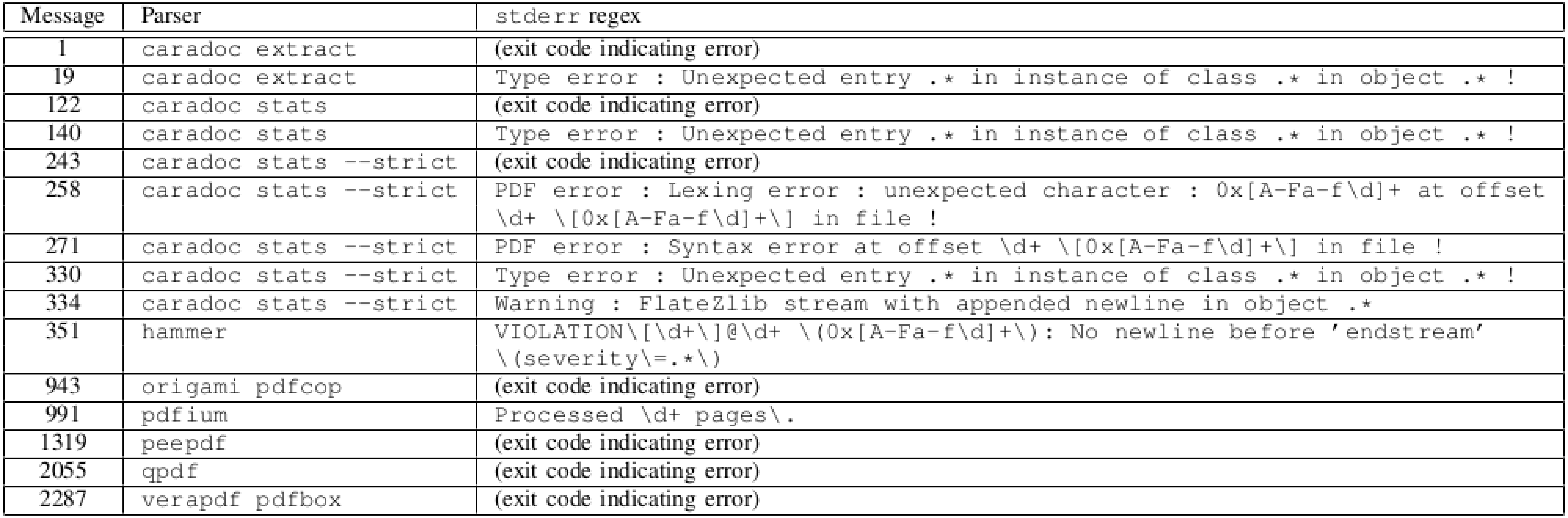
Parsers and regexes used

- Uninstrumented open-source parsers for PDF
- Capture `stderr` and return code
  - Apply all regexes to `stderr`, collect the matches
  - Various runtime options used, which leads to some duplication of regexes
- Many different regexes ensures good coverage of all file behaviors… this is hard to quantify!

Parser	Possible options	Messages
`caradoc`	`extract`	121
	`stats`	121
	`stats --strict`	94
`hammer`	(none)	69
`mutool`	`clean`	214
	`draw`	248
	`show`	75
`origami`	`pdfcop`	40
`pdfium`	(none)	26
`pdfminer`	`dumppdf`	88
	`pdf2txt`	155
`pdftk`	`server`	33
`pdftools`	`pdfid`	4
	`pdfparser`	30
`peepdf`	(none)	4
`poppler`	`pdffonts`	100
	`pdfinfo`	90
	`pdftocairo`	214
	`pdftoppm`	155
	`pdftops`	189
	`pdftotext`	139
`qpdf`	(none)	192
`verapdf`	`greenfield`	40
	`pdfbox`	50
`xpdf`	`pdffonts`	82
	`pdfinfo`	70
	`pdftoppm`	122
	`pdftops`	157
	`pdftotext`	100
Total		3022

Example regexes

- Regexes aggregated using basic field detection and matching

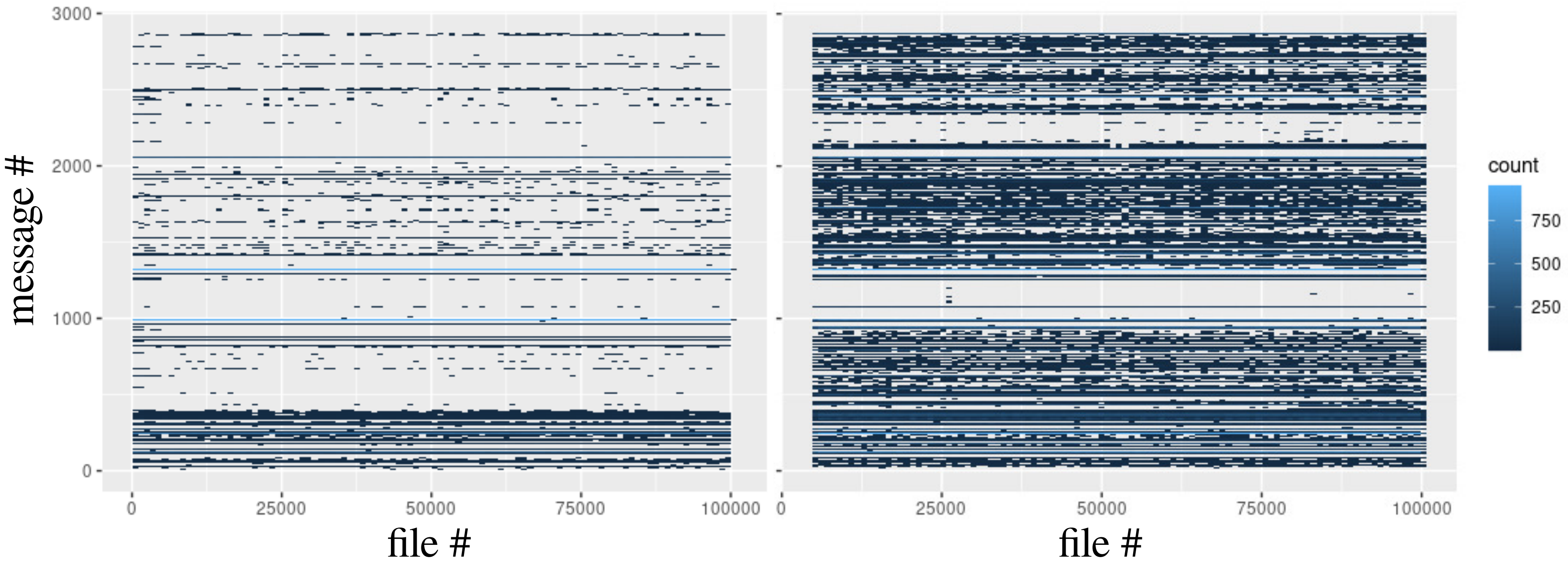


Message	Parser	stderr regex
1	caradoc extract	(exit code indicating error)
19	caradoc extract	Type error : Unexpected entry .* in instance of class .* in object .* !
122	caradoc stats	(exit code indicating error)
140	caradoc stats	Type error : Unexpected entry .* in instance of class .* in object .* !
243	caradoc stats --strict	(exit code indicating error)
258	caradoc stats --strict	PDF error : Lexing error : unexpected character : 0x[A-Fa-f\d]+ at offset \d+ \[0x[A-Fa-f\d]+\] in file !
271	caradoc stats --strict	PDF error : Syntax error at offset \d+ \[0x[A-Fa-f\d]+\] in file !
330	caradoc stats --strict	Type error : Unexpected entry .* in instance of class .* in object .* !
334	caradoc stats --strict	Warning : FlateZlib stream with appended newline in object .*
351	hammer	VIOLATION\[\d+\]@\d+ \(0x[A-Fa-f\d]+\): No newline before ’endstream’ \(severity\=.*\)
943	origami pdfcop	(exit code indicating error)
991	pdfium	Processed \d+ pages\.
1319	peepdf	(exit code indicating error)
2055	qpdf	(exit code indicating error)
2287	verapdf pdfbox	(exit code indicating error)

Our dataset: DARPA SafeDocs Eval 3

Our team ran a battery of 3022 regexes against each file we were given

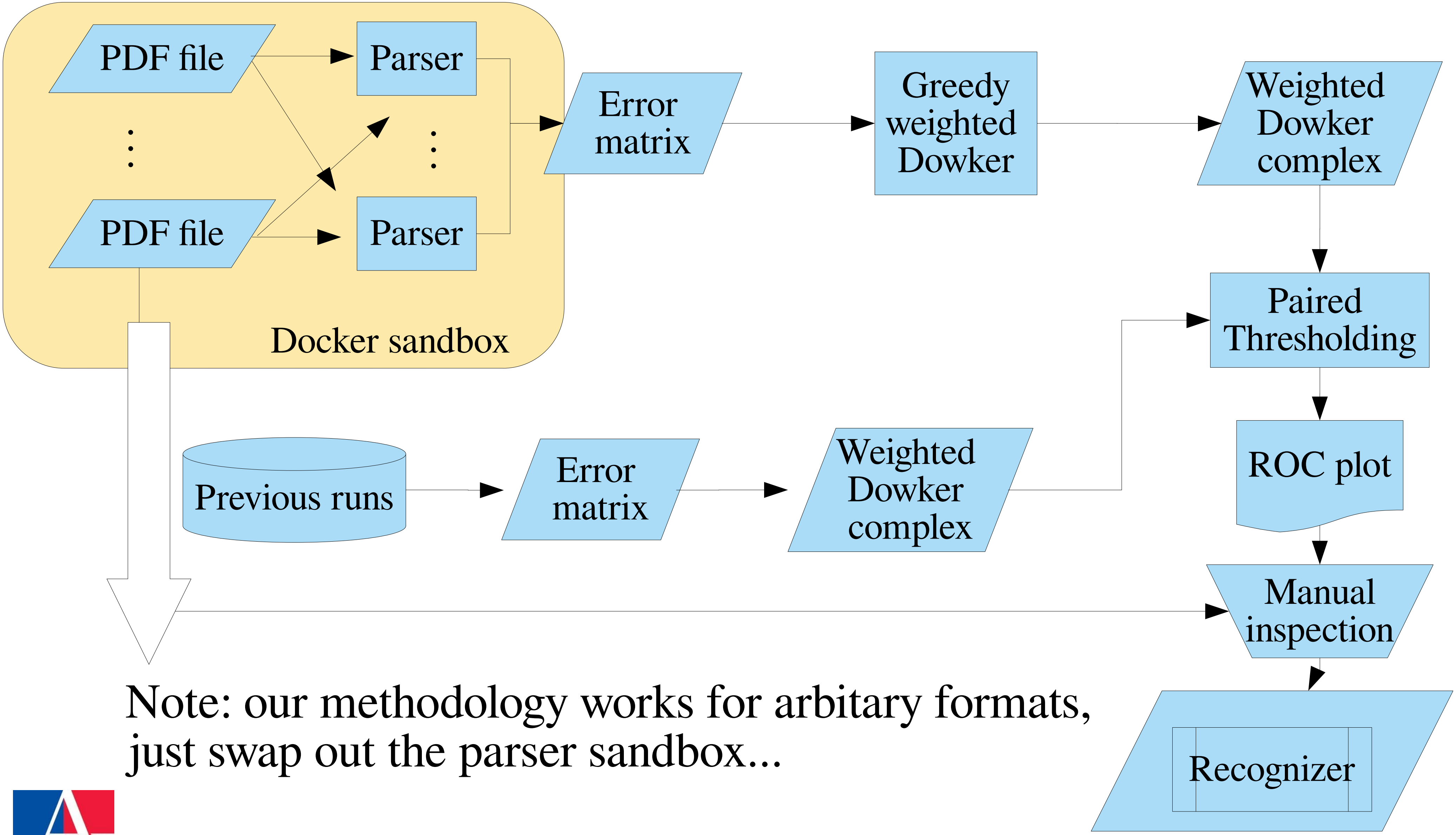
- Each regex → a "message"
- Each file is a distinct PDF



Good files

Bad files

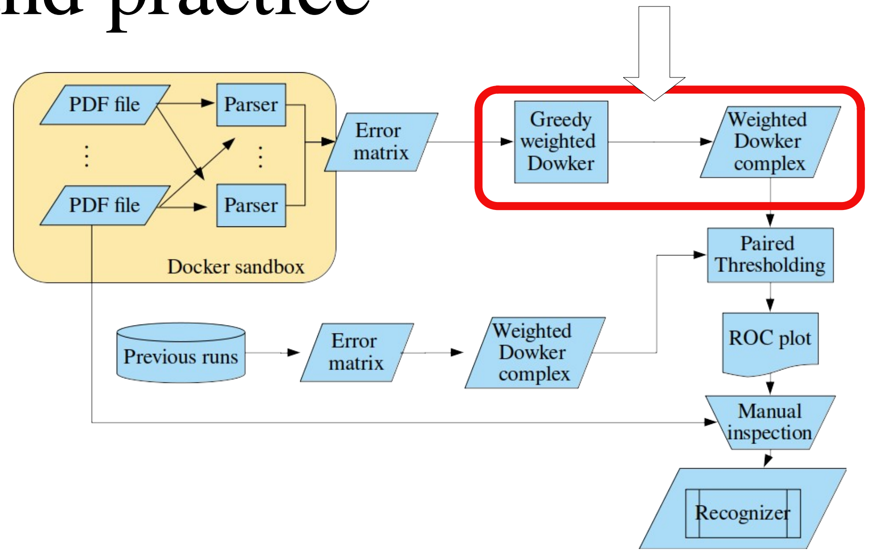
Workflow for [whatever] format files



Note: our methodology works for arbitary formats,
just swap out the parser sandbox...

Weighted Dowker complexes

Topological theory and practice

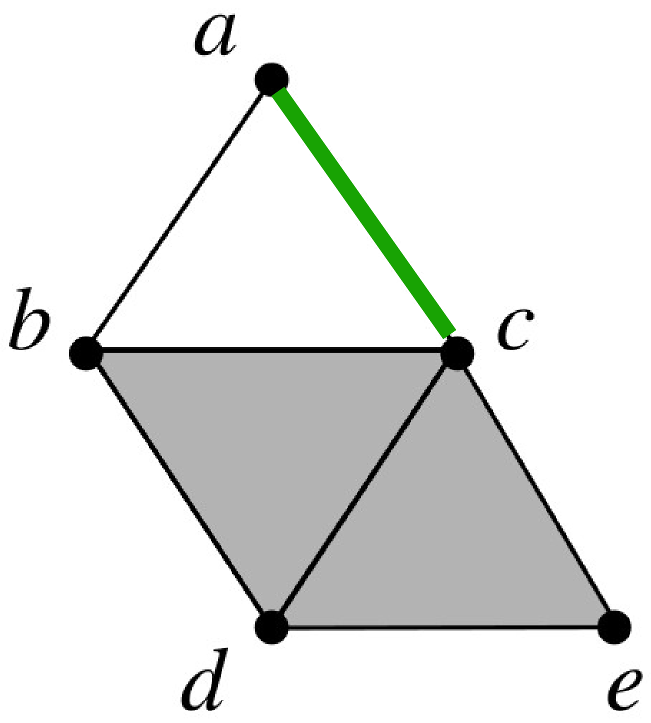


Michael Robinson

Topological features: Dowker complex

- Each row specifies a vertex
- Each column specifies (at least one) simplex by selecting subsets of vertices

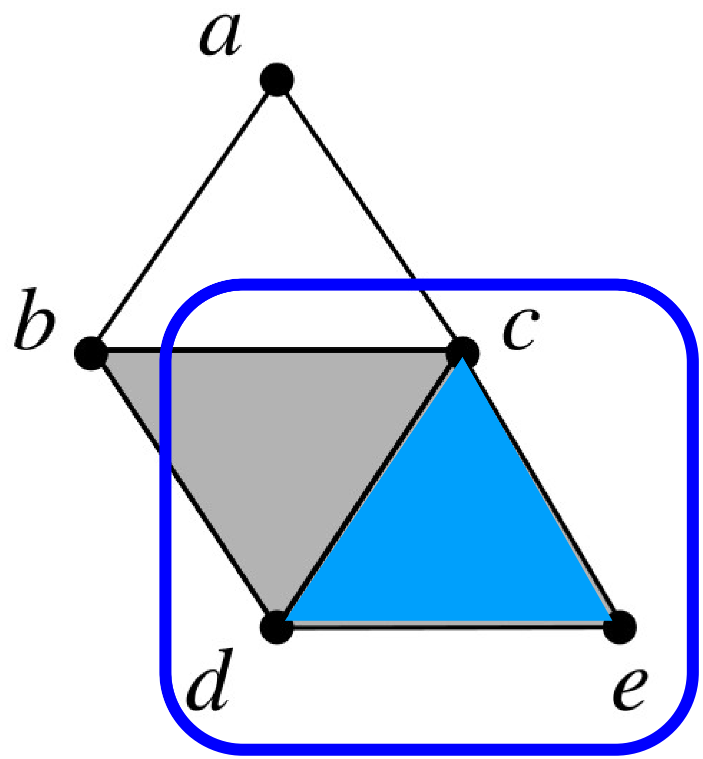
$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline a & 1 & 1 & 0 & 0 & 0 \\ b & 1 & 0 & 1 & 0 & 0 \\ c & 0 & 1 & 1 & 1 & 1 \\ d & 0 & 0 & 1 & 1 & 0 \\ e & 0 & 0 & 0 & 1 & 1 \end{array}$$



Topological features: Dowker complex

- Each row specifies a vertex
- Each column specifies (at least one) simplex by selecting subsets of vertices

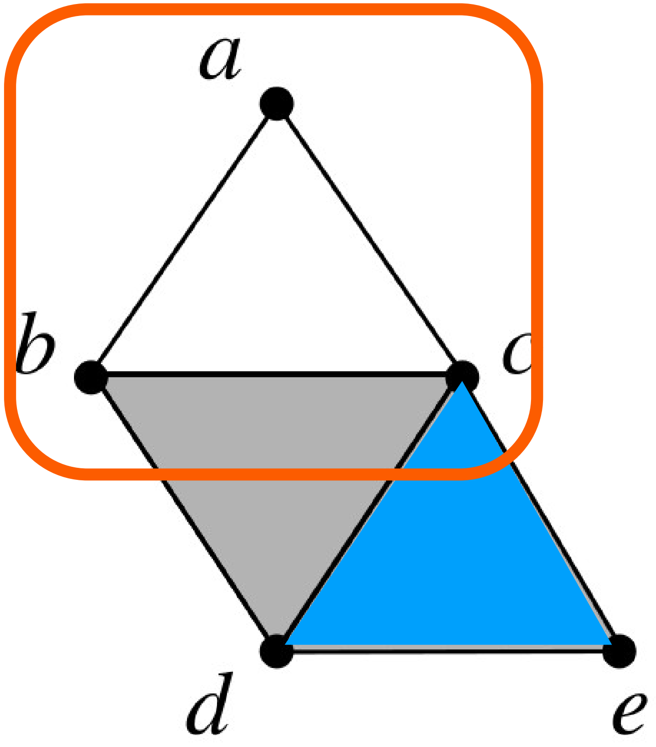
$$\begin{array}{cc} & \begin{array}{ccccc}1 & 2 & 3 & 4 & 5\end{array} \\ \begin{array}{c}a\\ b\\ c\\ d\\ e\end{array} & \begin{pmatrix}1 & 1 & 0 & 0 & 0\\ 1 & 0 & 1 & 0 & 0\\ 0 & 1 & 1 & 1 & 1\\ 0 & 0 & 1 & 1 & 0\\ 0 & 0 & 0 & 1 & 1\end{pmatrix}\end{array}$$



Topological features: Dowker complex

- Parsers *a, b, c* have less agreement about the files to accept than *b, c, d* – the loop **witnesses the presence of a format disagreement**

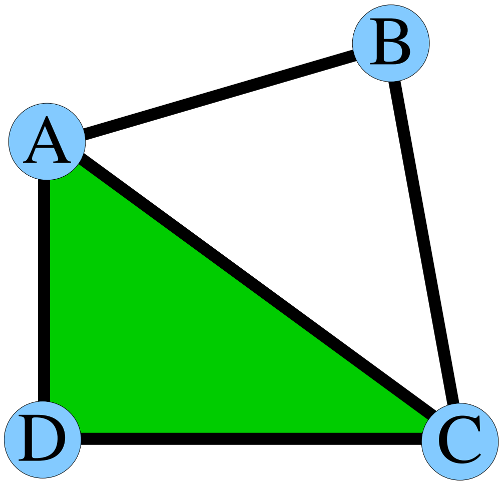
$$\begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
a & 1 & 1 & 0 & 0 & 0 \\
b & 1 & 0 & 1 & 0 & 0 \\
c & 0 & 1 & 1 & 1 & 1 \\
d & 0 & 0 & 1 & 1 & 0 \\
e & 0 & 0 & 0 & 1 & 1
\end{array}$$



Dowker is lossy

- Dowker ignores duplicate columns
- Here are several non-isomorphic relations inducing the same complex

$$\text{messages}\ \begin{array}{c} \\ \begin{matrix}A\\B\\C\\D\end{matrix}\end{array}\ \begin{array}{c}\text{files}\\ \begin{pmatrix}10000011111100001111\\01100011100010000000\\00011000010011111111\\00000100001101111111\end{pmatrix}\end{array}$$



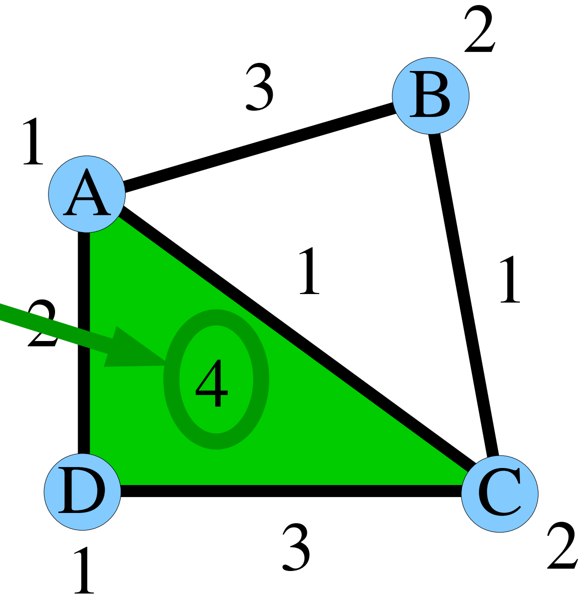
$$\text{messages}\ \begin{array}{c} \\ \begin{matrix}A\\B\\C\\D\end{matrix}\end{array}\ \begin{array}{c}\text{files}\\ \begin{pmatrix}101001\\110000\\011101\\001010\end{pmatrix}\end{array}$$

$$\text{messages}\ \begin{array}{c} \\ \begin{matrix}A\\B\\C\\D\end{matrix}\end{array}\ \begin{array}{c}\text{files}\\ \begin{pmatrix}00110110\\01100000\\01011000\\00010101\end{pmatrix}\end{array}$$

Weighted Dowker complex

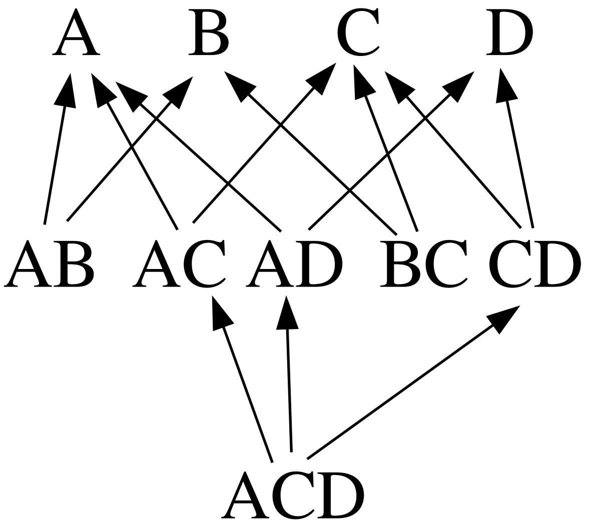
- Weighting: Count how many times each simplex appears
- Theorem: The matrix is determined (up to isomorphism) by the Dowker complex with this weight function
- Deeper theorem: This can be enriched into a *cosheaf representation*

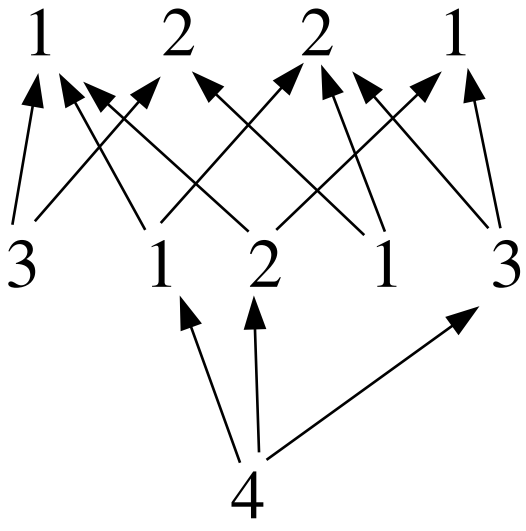
$$\text{messages}\quad\begin{array}{cc} & \text{files} \\ \begin{array}{c} A \\ B \\ C \\ D \end{array} & \left(\begin{array}{l} 1000001111110000\boxed{1111} \\ 0110001110001000\boxed{0000} \\ 0001100001001111\boxed{1111} \\ 0000010000110111\boxed{1111} \end{array}\right. \end{array}$$

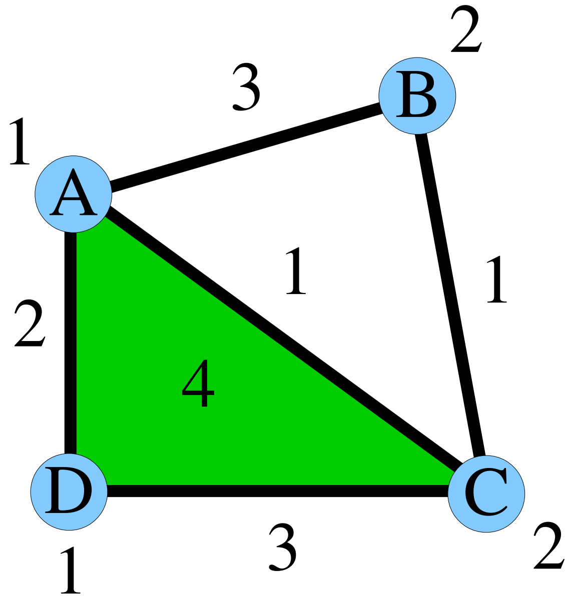


Weighted Dowker complex

- Practically speaking: Better to display as a poset, which evokes the *formal concept lattice*
- This helps if there are message patterns with many messages





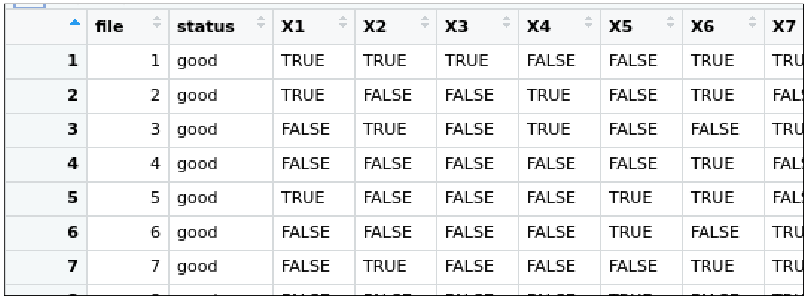


Dowker is easy to compute...

- Since I'll be doing lots of file stats, let's use R*

Messages X?

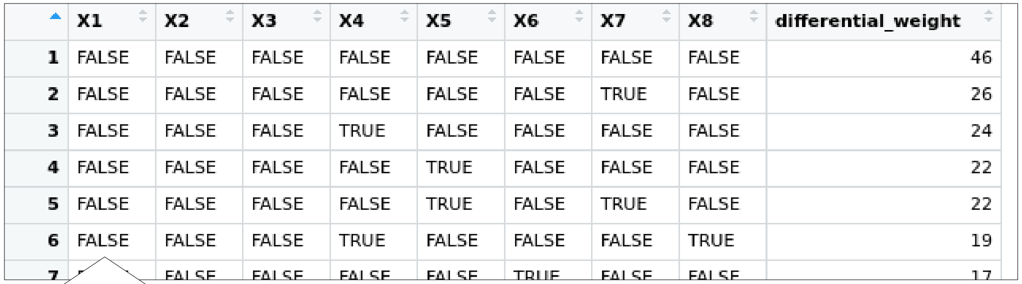
Files



	file	status	X1	X2	X3	X4	X5	X6	X7
1	1	good	TRUE	TRUE	TRUE	FALSE	FALSE	TRUE	TRU
2	2	good	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	FAL
3	3	good	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRU
4	4	good	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FAL
5	5	good	TRUE	FALSE	FALSE	FALSE	TRUE	TRUE	FAL
6	6	good	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRU
7	7	good	FALSE	TRUE	FALSE	FALSE	FALSE	TRUE	TRU

Message patterns

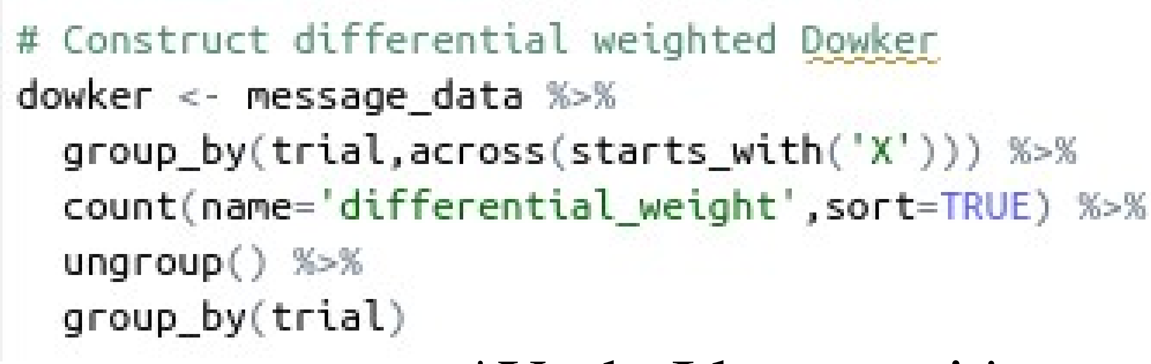
Messages + Dowker weight



	X1	X2	X3	X4	X5	X6	X7	X8	differential_weight
1	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	46
2	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	26
3	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	24
4	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	22
5	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	22
6	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	TRUE	19
7	[illegible]	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	17

Input data from parsers

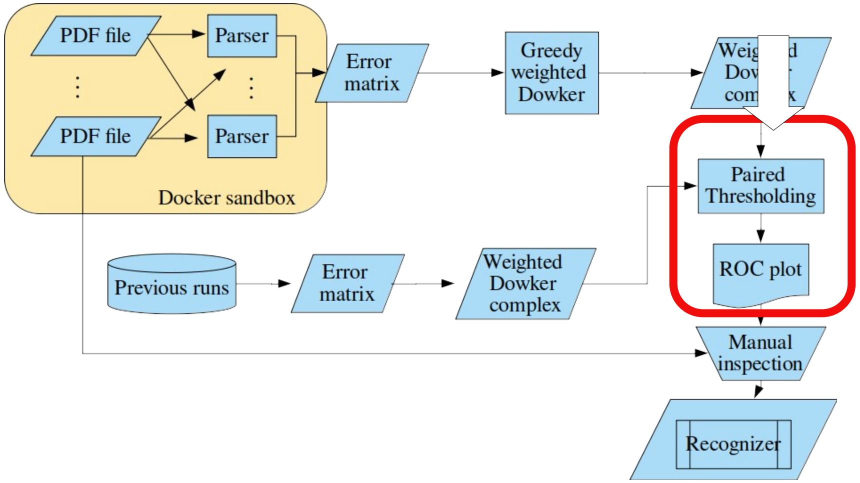
Weighted Dowker complex data, ready for further processing



```
# Construct differential weighted Dowker
dowker <- message_data %>%
  group_by(trial,across(starts_with('X'))) %>%
  count(name='differential_weight',sort=TRUE) %>%
  ungroup() %>%
  group_by(trial)
```

*Yeah, I know… it's not very memory efficient

Statistical weighted Dowker



Key theory insight

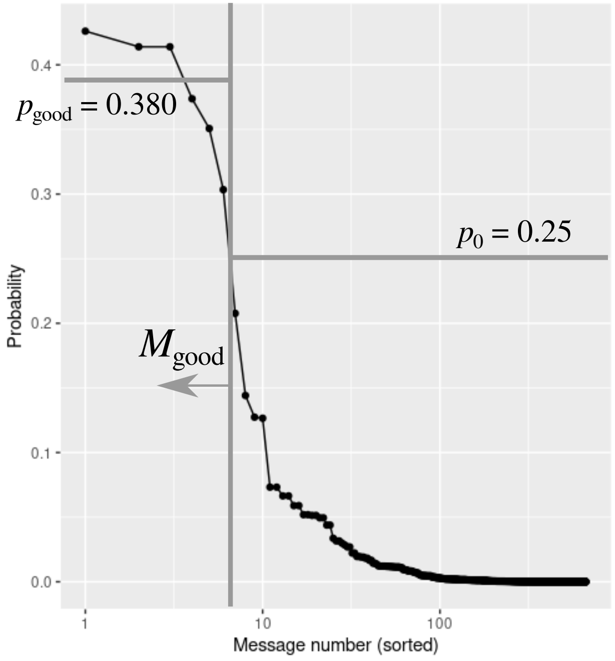
In the corpora we've examined, there are really only two kinds of messages:

- Those that happen frequently
- Those that are more sporadic

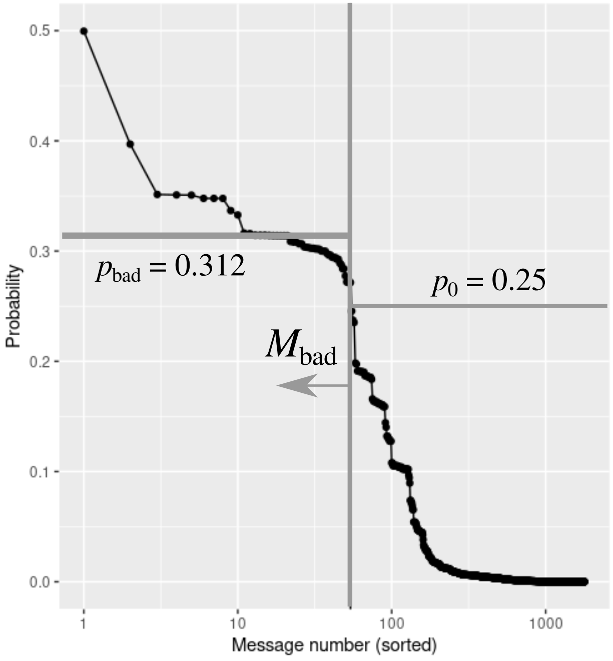
<u>Tactic:</u>

Threshold messages into two classes based on frequency

These classes differ for different dialects!

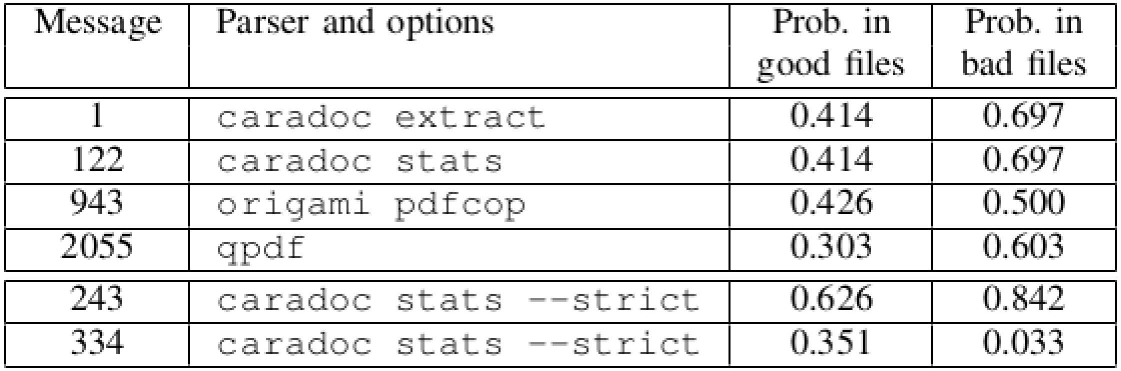


Good files



Bad files

Good/bad message overlap?



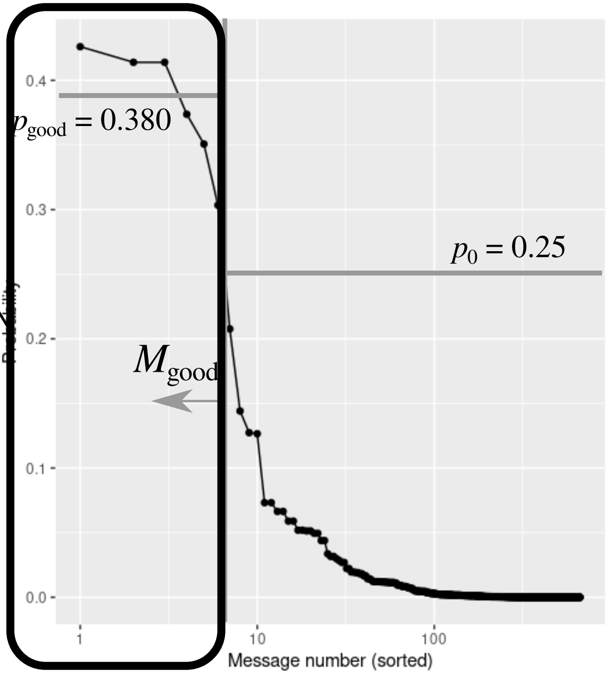
Message	Parser and options	Prob. in good files	Prob. in bad files
1	`caradoc extract`	0.414	0.697
122	`caradoc stats`	0.414	0.697
943	`origami pdfcop`	0.426	0.500
2055	`qpdf`	0.303	0.603
243	`caradoc stats --strict`	0.626	0.842
334	`caradoc stats --strict`	0.351	0.033

Also among the most frequent in Bad files

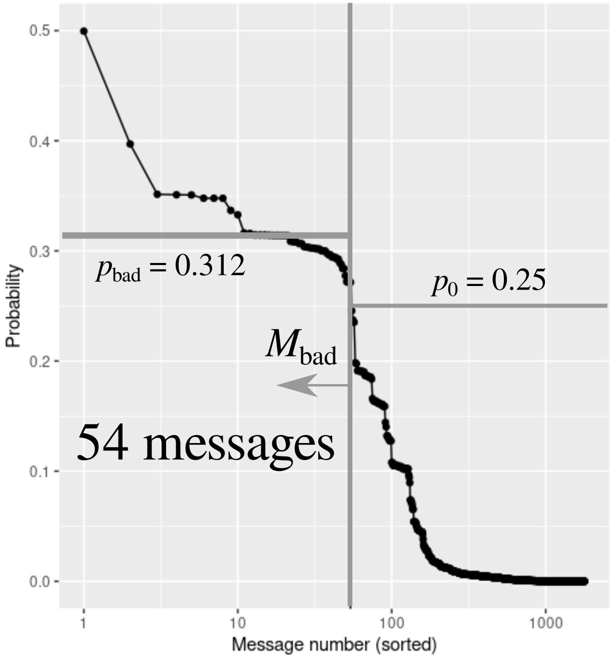
Effectively unique* to Good files

The most frequent messages in Good files

*Messages that occur more on more than 50% files are best tested for absence not presence!



Good files



Bad files

Following the key insight

What's the probability that a file from dialect *A* exhibits a set of messages *K*?

Given our thresholded message probabilities, this is easy if we assume* messages are independent when conditioned on dialect:

$$
\begin{aligned}
P(K|A) =& p_0^{\#(K\cap M_A^c)}(1-p_0)^{\#(K^c\cap M_A^c)}\times \\
& p_A^{\#(K\cap M_A)}(1-p_A)^{\#(K^c\cap M_A)}.
\end{aligned}
$$

background less frequent messages

dialect *A* more frequent messages

Message didn't happen

Message did happen

*Hold that thought...

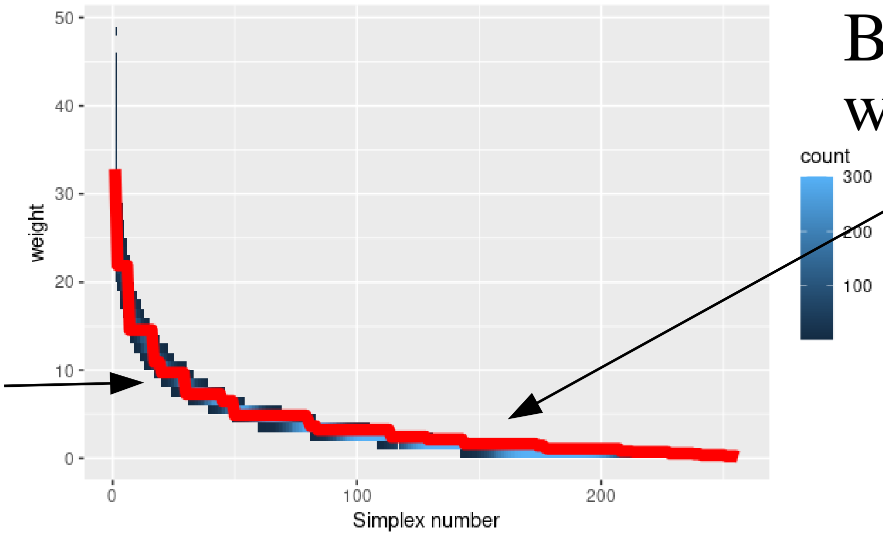
Payoff: a formula for file counts

Consider a pattern with *n* messages, *k* of which are of the more frequent kind for dialect *A*.

There are $\binom{\#M_A}{k}\binom{\#M-\#M_A}{n-k}$

ways that this can happen… and each has probability

$$P(\#K = n, \#(K \cap M_A) = k|A) =$$
$$p_0^{n-k}(1-p_0)^{(\#M-\#M_A-(n-k))}p_A^k(1-p_A)^{\#M_A-k}.$$



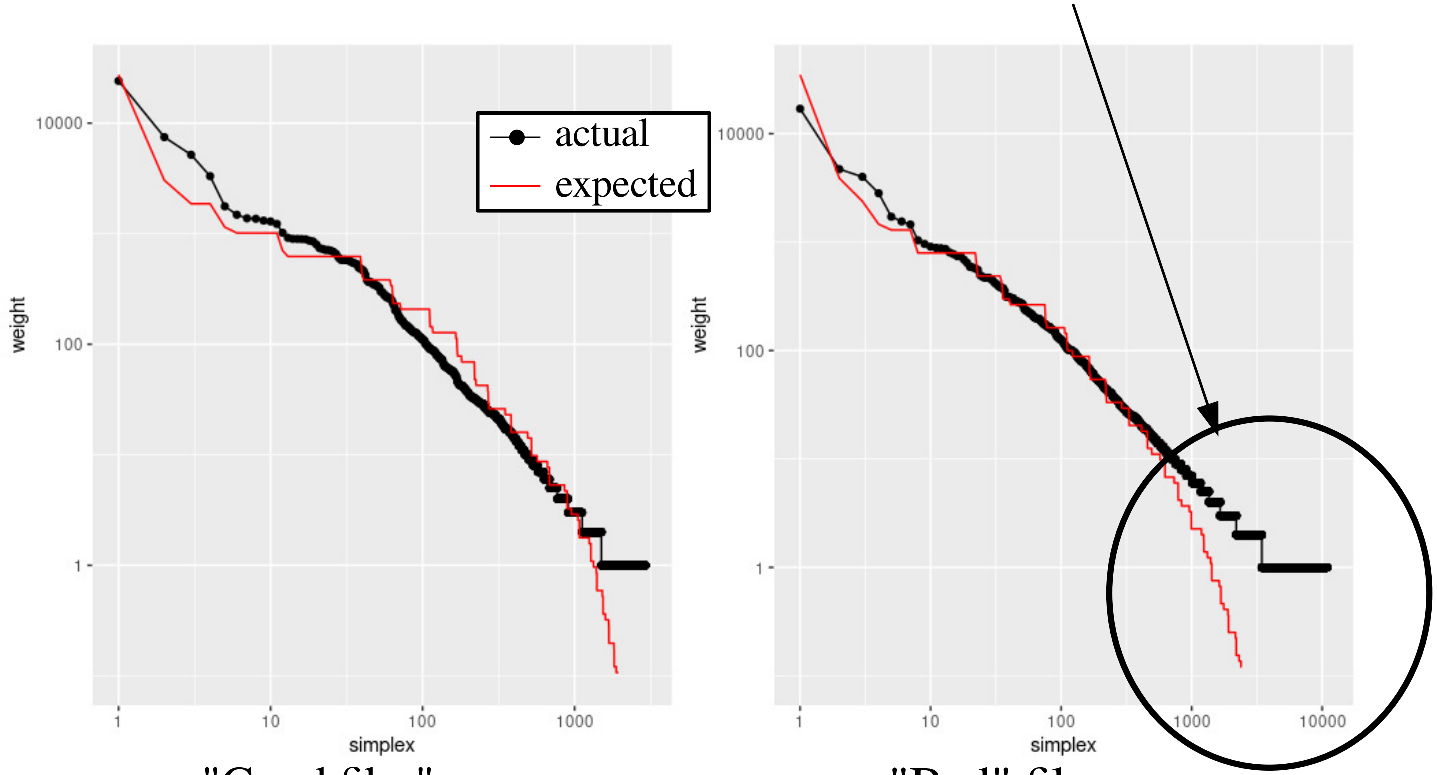
Blue: simulated data with 1000 files, 8 messages

Red: the formula

tl;dr formula works **well**

Payoff (cont'd): formula works in practice, too

- A few files exhibit strange message patterns for definite reasons
  - These happen more frequently than expected in the bad files… these are the **really** interesting files

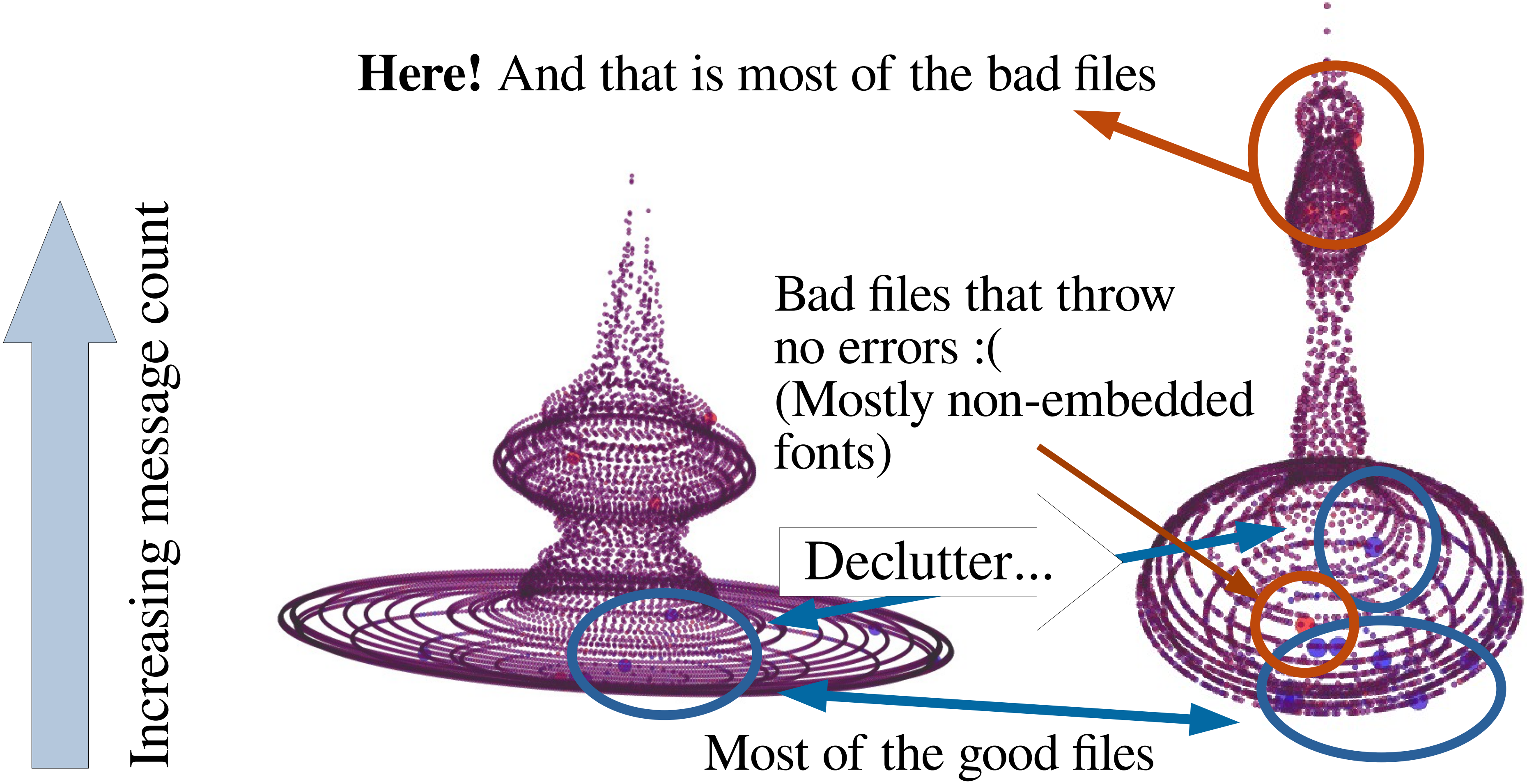


"Good files"

"Bad" files

Where are those interesting files?

Poset representation: each point is a message pattern, colored by file count



Here! And that is most of the bad files

Bad files that throw no errors :((Mostly non-embedded fonts)

Declutter...

Most of the good files

Figure credit: Letitia Li (BAE Systems)

Find the files: weight thresholding

Once files are grouped by message pattern, assign the posterior probability of being in a given dialect…

Risk factor: how many bad files expected?

$$P(A|K) = P(K|A)\frac{P(A)}{P(K)}.$$

Prevalence of message pattern *K* in test data

File is in dialect *A* (ie. Bad) …

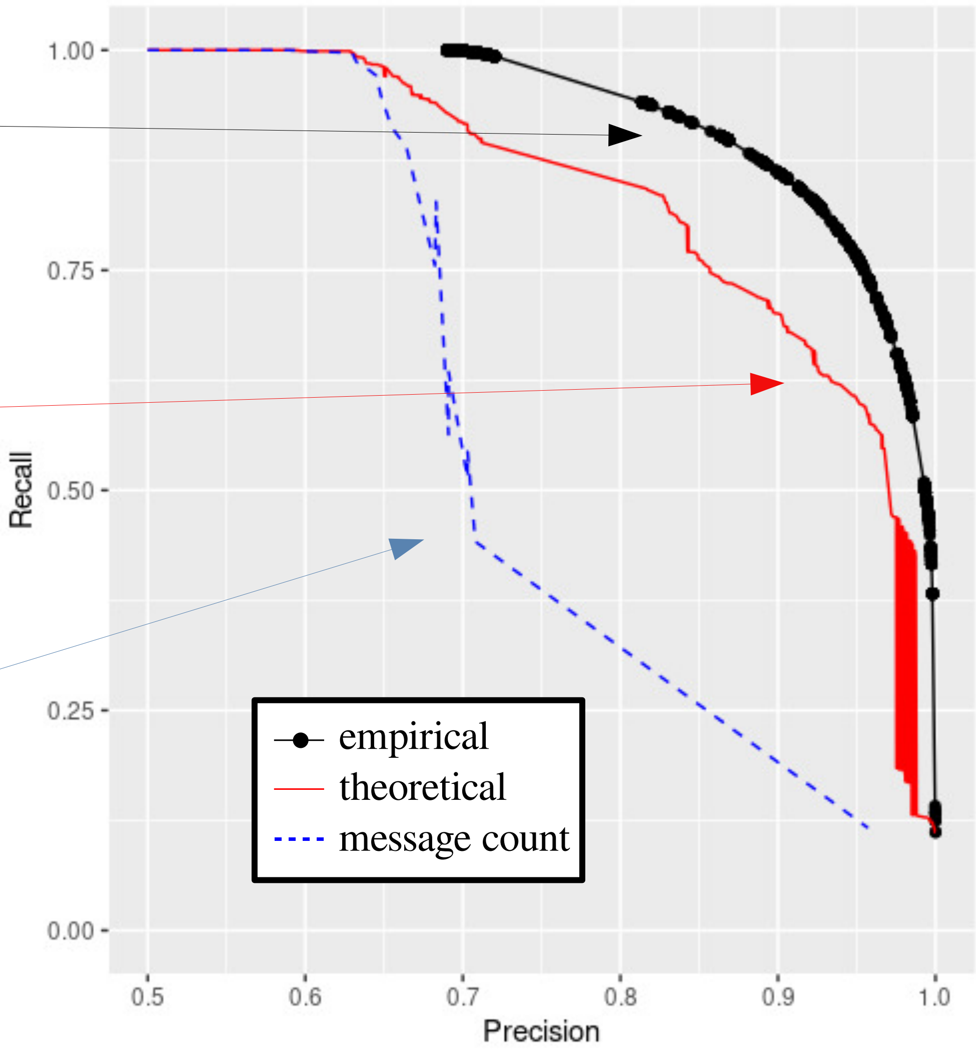
given that … we see pattern *K*

Normalized Dowker weight for Bad training files

… and then you can simply threshold this probability to classify files!

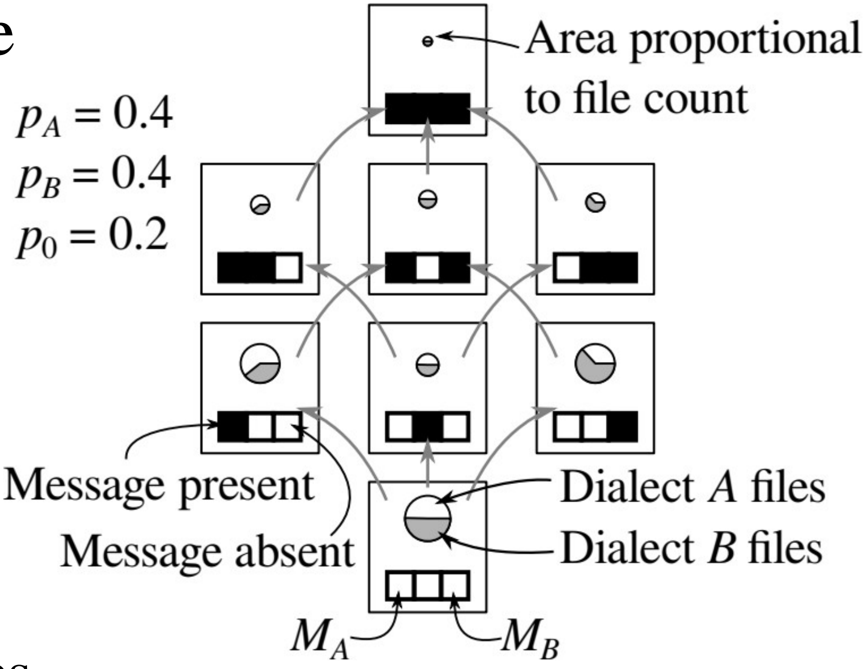
Weight thresholding performance

- **Best**: If you have training data, then estimate *P*(*K* | *A*) empirically
- **Good**: If you don't have training data, we can now **bootstrap** using our model's theoretical *P*(*K* | *A*)
- **Subpar**: For comparison, what if we just reject files that throw too many errors? :-(



Why does Dowker thresholding work?

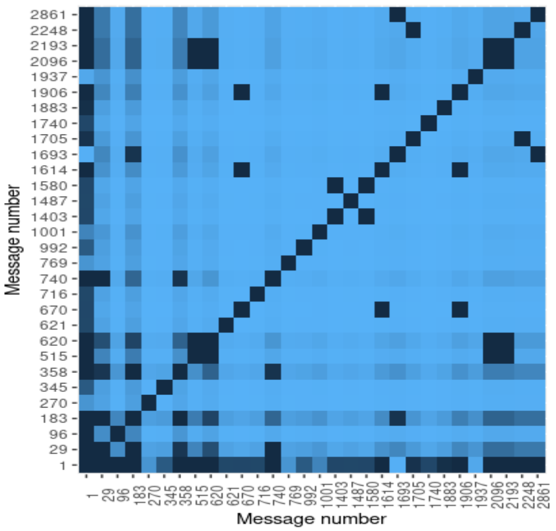
- For 3 messages, the Dowker complex only has 8 message patterns
- We can estimate **everything** about the file breakdown per message pattern
- For two dialects:
  - Can split dialects for many files using message pattern alone
  - There are some ambiguous patterns
- Explainability:
  - Lemma: Each message pattern isolates a semantic *formal concept*
  - (Empirical) Conjecture: these formal concepts align with semi-manually created message ontologies



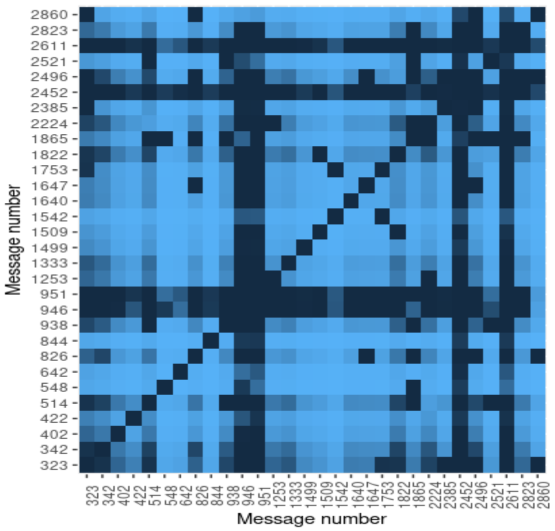
That pesky independence assumption

- The independence assumption need not be taken without justification!
  - There's a statistical test for independence: χ^2
  - My freshmen STAT students would not want to compute ~1000 χ^2 tests, but my computer is OK with this task!

Conclusion: **Independence holds**, except for some notable cases (duplicate messages from parsers deployed with different options, actually)

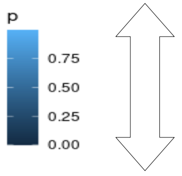


Good files



Bad files

Independent



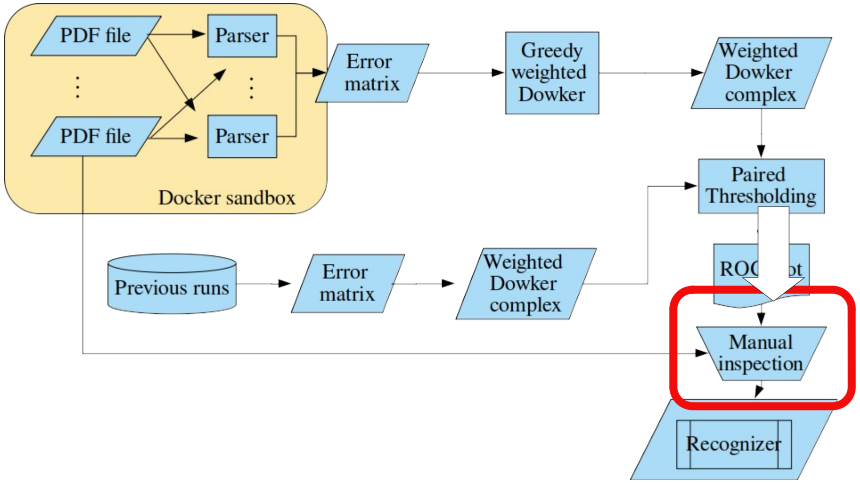
Dependent

Michael Robinson

Limitation: Need sufficient message diversity

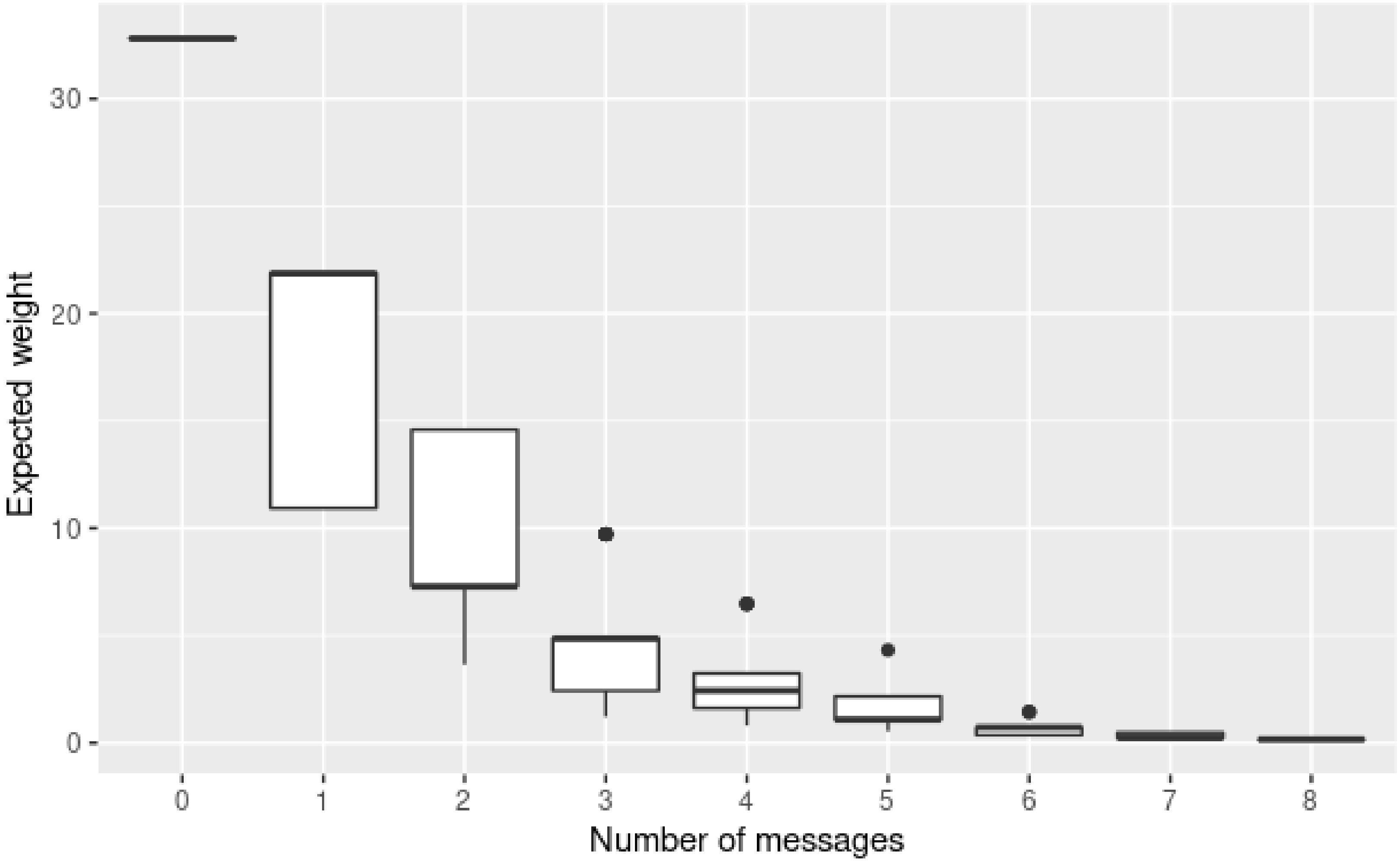
- Thanks Cory!!! (your message sets rock!)
- Conditional independence is a starting point
- Next step: What is the correct measure of diversity?
  - Can we tell when we're missing something? (I suspect so, but that's only a hunch)
  - Especially important if we don't have ground truth
  - This **may be visible** in the formal concept lattice or the topology of the Dowker complex
- Conjecture: A small number of the messages are really useful, while the rest are moderately useful

Finer subsetting of files



Inconsistent edges

- Patterns with **more** messages are usually **less** frequent



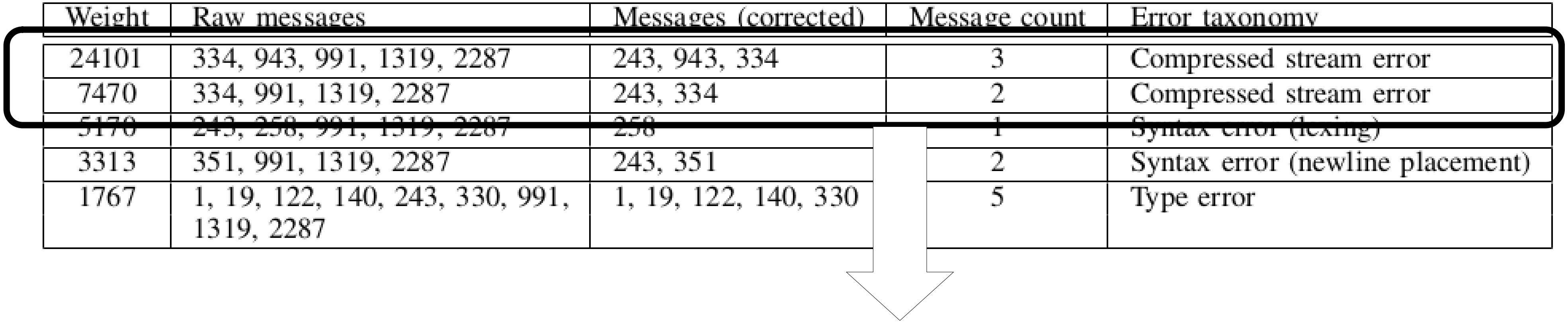
Inconsistent edges

- Patterns with **more** messages are usually **less** frequent
- Definition: An *inconsistent edge* is when the opposite happens:
  - A pair of message patterns differing by one message where the pattern with **more** messages occurs on **more** files
- Theorem: Under our theoretical model, inconsistent edges **never happen**
- When inconsistent edges happen, it means that the assumptions underlying our model are wrong
  - Clear signal that **something interesting is happening** for the subset of files that are implicated

Inconsistency in action!

Our data shows several inconsistent edges, corresponding to definite semantic features

Here's the top weight message patterns in good files



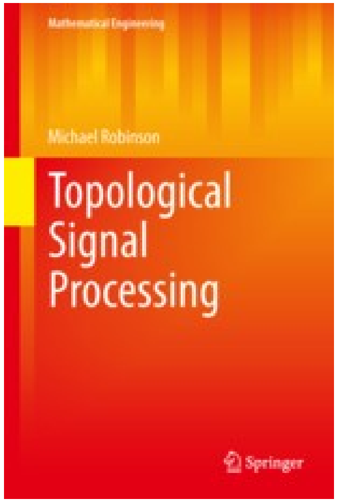
Weight	Raw messages	Messages (corrected)	Message count	Error taxonomy
24101	334, 943, 991, 1319, 2287	243, 943, 334	3	Compressed stream error
7470	334, 991, 1319, 2287	243, 334	2	Compressed stream error
5170	243, 258, 991, 1319, 2287	258	1	Syntax error (lexing)
3313	351, 991, 1319, 2287	243, 351	2	Syntax error (newline placement)
1767	1, 19, 122, 140, 243, 330, 991, 1319, 2287	1, 19, 122, 140, 330	5	Type error

Addition of message 943 nearly triples the number of files!
Message 943 is an otherwise unspecified exit code of `orgami pdfcop`
Semantic inference: this message is often relevant to the validity of compressed streams, even though it emits no text to `stderr` at all!

Conclusions

- Our weighted Dowker pipeline is largely format agnostic
  - It should “work” for other formats…
  - … provided we have enough features to exploit
- What constitutes a *feature* is a bit amorphous:
  - Parsers
  - Traces
  - Messages
  - System calls
  - Machine learning results
  - Grammar productions… and more?
- But this doesn’t matter if you work statistically!
- Thresholding the posterior probability works very well at splitting dialects in practice!

To learn more...


Mathematical Engineering

Michael Robinson

Topological Signal Processing

Springer

Michael Robinson

michaelr@american.edu

http://drmichaelrobinson.net

Relevant preprints:

https://arxiv.org/abs/2201.08267

https://arxiv.org/abs/2003.00976

https://arxiv.org/abs/2005.12348

Software:

https://github.com/kb1dds