Comparing ODF and OOXML

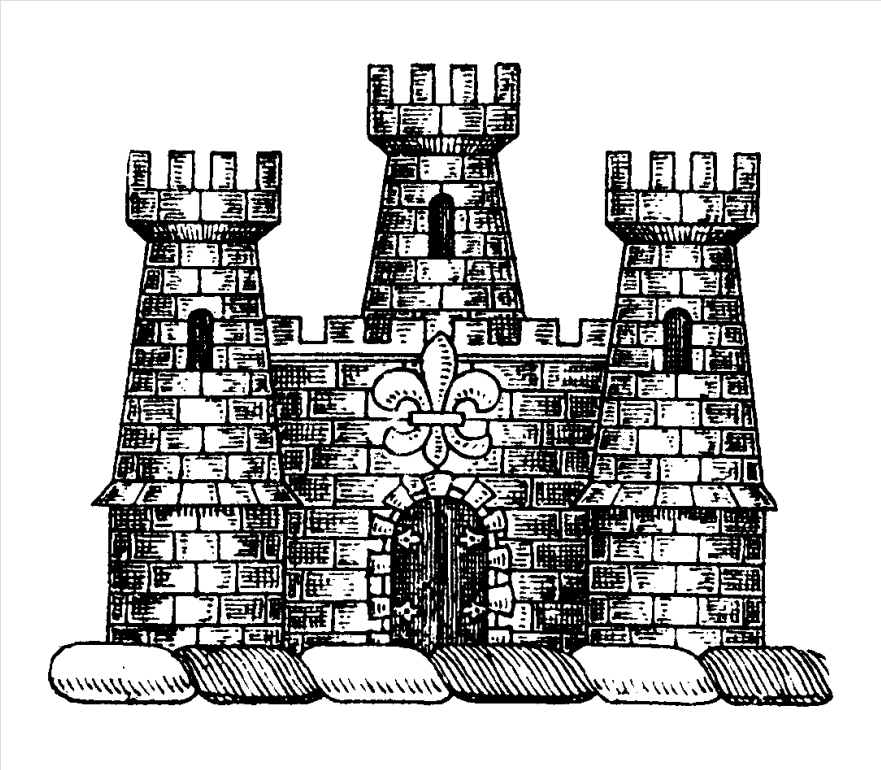
Treating the extensibility, modularization, expressivity, packaging, performance, reuse of standards, programmability, ease of use, application/OS neutrality of the formats, along with other whims of the author

Rob Weir
IBM
robert_weir@us.ibm.com
http://www.robweir.com/blog

OpenOffice.org Conference
Lugdunum, Gaul
Ides of September, 2006

The age of proprietary formats

- Created by a single vendor
- Controlled a single vendor
- Evolved by a single vendor



Rich Text Format

"The RTF standard provides a format for text and graphics interchange that can be used with different output devices, operating environments, and operating systems. RTF uses the ANSI, PC-8, Macintosh, or IBM PC character set to control the representation and formatting of a document, both on the screen and in print. With the RTF standard, you can transfer documents created under different operating systems and with different software applications among those operating systems and applications"

From RTF 1.0 specification (1987)

We once had documentation

- *Microsoft Excel Developers Handbook*, Microsoft Press, 1997

- MSDN CD's had the Office binary file format documentation
- MSDN web site had the Office binary file format documentation
- Visual C++ came with the Office binary file format documentation

But at some point, the updates stopped coming,
and the documentation was pulled.
What happened ???

The door is shut...

“...you may use documentation identified in the MSDN Library portion of the SOFTWARE PRODUCT as the file format specification for Microsoft Word, Microsoft Excel, Microsoft Access, and/or Microsoft PowerPoint ("File Format Documentation") solely in connection with your development of software product(s) that operate in conjunction with Windows or Windows NT that are not general purpose word processing, spreadsheet, or database management software products or an integrated work or product suite whose components include one or more general purpose word processing, spreadsheet, or database management software products.”

MSDN Licence, 1998

...and locked...

By 1999 the format documentation is no longer available for download.

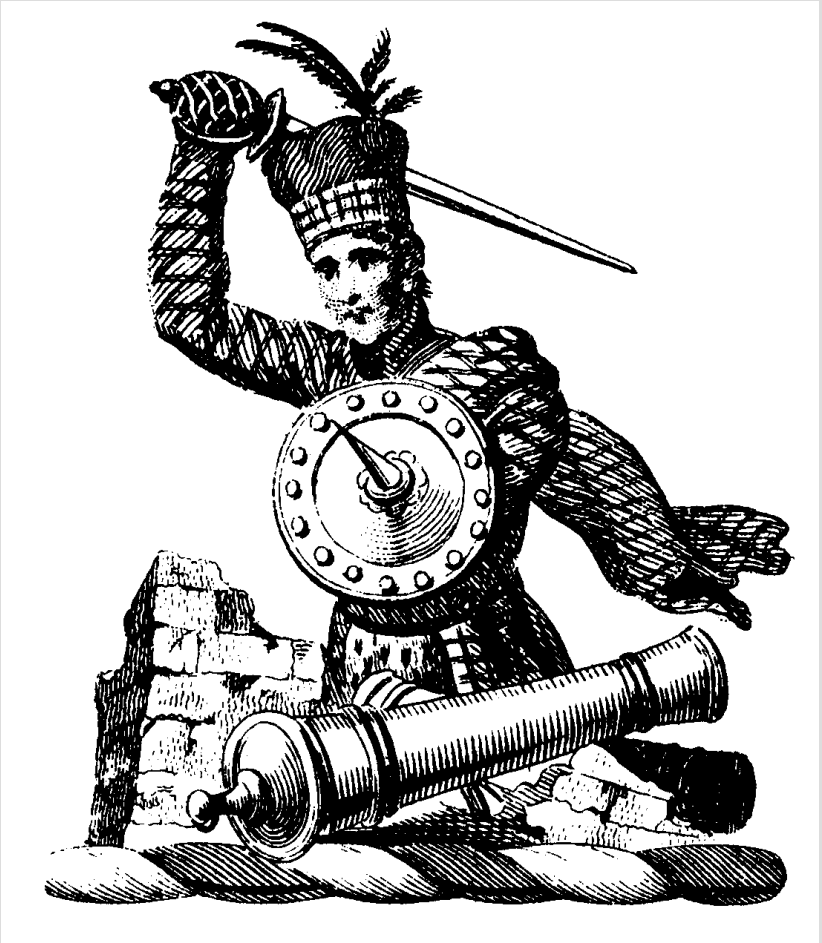
Alternative is to license from Microsoft under these terms;

“ISV License Program

This program entitles qualified software developers to license the Microsoft .doc, .xls, or .ppt file format documentation for use in the development of commercial software products and solutions that support the .doc, .xls, or .ppt file formats from Microsoft and to complement Microsoft Office.”

..and a guard posted at the door

- Since Office 2003, Digital Rights Management is being pushed into Office.
- The Digital Millennium Copyright Act and the EU Copyright Directive have provisions which make it illegal to circumvent DRM
- So although progress at interop to date has been heroic, proponents of closed formats have the technological and legal means to prevent document exchange if they wish. This is true in XML word as well as in the binary world.



Standardization Process

- ODF
  - Based on OO.org XML formats
  - 12 Dec 2002 -- submitted to OASIS
  - 1 May 2005 – OASIS ODF standard released
  - 16 Nov 2006 – Submitted to ISO/IEC JTC1 under Publicly Available Specification (PAS) rules.
  - 3 May 2006 – ISO/IEC IS 26300 approved
  - 706 page specification in 867 days
- OOXML
  - Based on Office 2003 XML formats
  - 15 Dec 2005 -- submitted to Ecma
  - est. by 31 Dec 2006 – Ecma standard approved at Ecma General Assembly
  - est. by 31 January 2007 – Submitted to ISO/IEC JTC1 under FastTrack rules.
  - est. Q1/2008 – ISO/IEC JTC1 approval of OOXML???
    - Gartner forecasts low likelihood of approval *
  - 5,419 page specification (draft 1.4) in 254 days

* http://www.gartner.com/resources/140100/140101/iso_approval_of_oasis_opendo_140101.pdf

Know the SDO's

- OASIS
  - DocBook
  - DITA
  - RELAX NG
  - ebXML
  - LegalXML
  - Emphasis on e-business standards
- Ecma
  - C#
  - CLI
  - EcmaScript
  - Eiffel Programing Language
  - As well as various hardware and media standards

How open is open?

	OASIS	Ecma
Allows individual members	Yes	No
Mailing lists viewable by the public	Yes	No
Meeting agendas and minutes public	Yes	Only report of face-to-face meetings
Received public comments are viewable	Yes	No

Reuse of standards

"If I have seen a little further it is by standing on the shoulders of Giants."

Isaac Newton, letter to Robert Hooke, 1676

Choose reuse because:

- Reduced time to write specification
- Higher quality specifications
- Can leverage existing community expertise
- Can leverage existing education materials
- Better interop, especially in a word of promiscuous mashups, not monolithic silos
- Network effects – synergy is good

Reuse: Head to Head

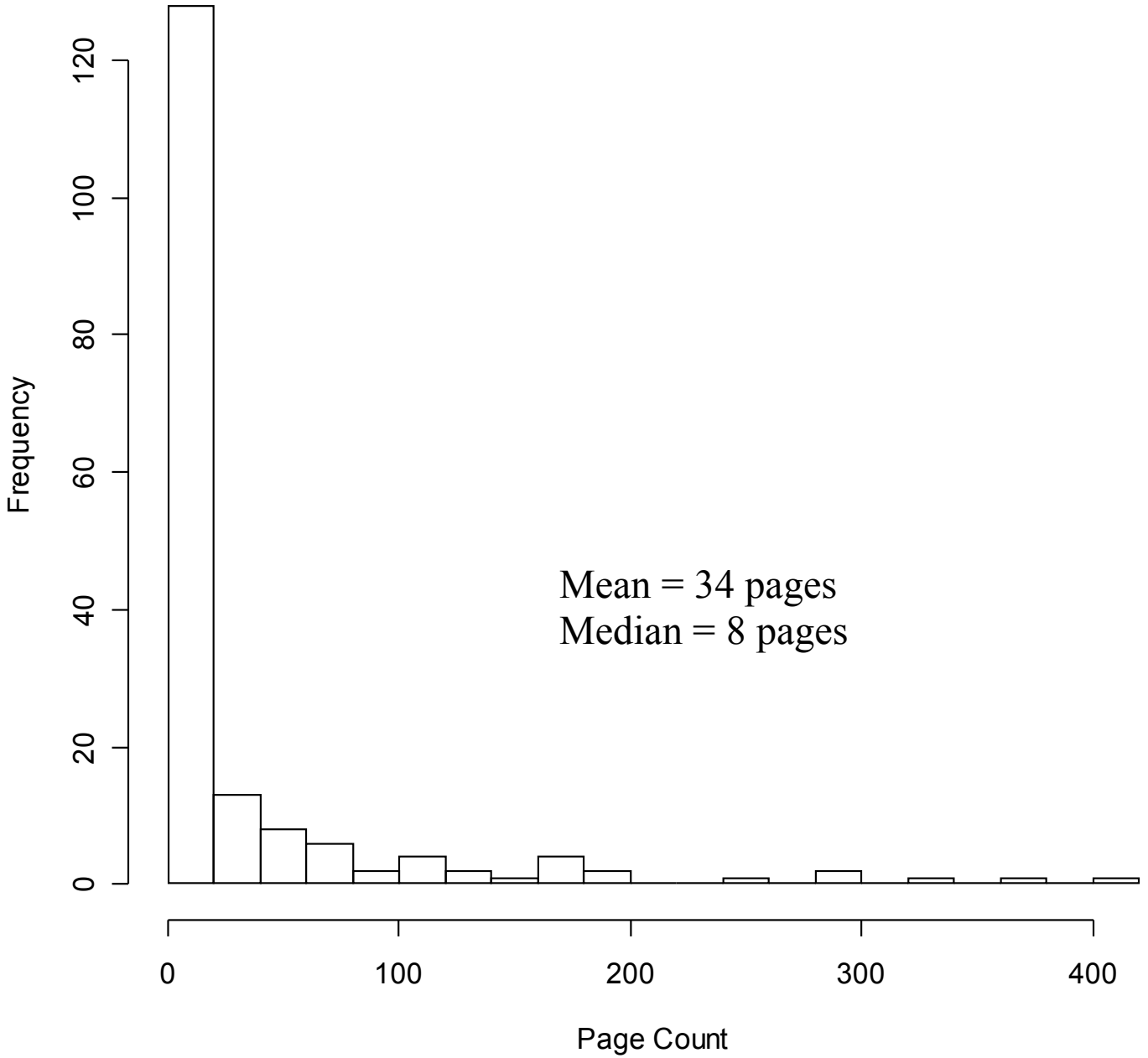
- ODF reuses:
  - Dublin Core
  - XLS:FO
  - SVG
  - MathML
  - XLink
  - SMIL
  - XForms
- OOXML reuses:
  - Dublin Core

Packaging

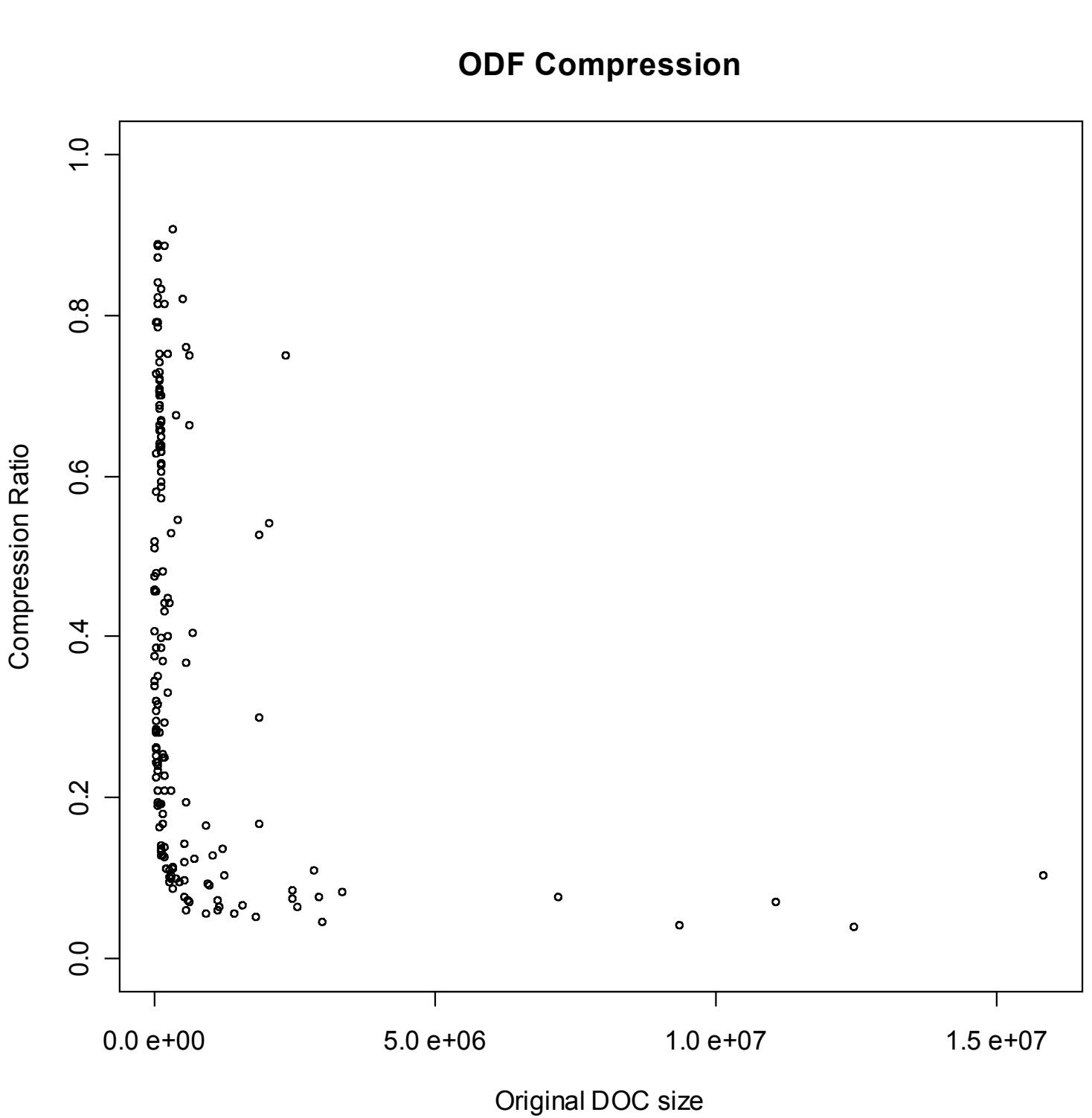
- Both formats use a ZIP-format container file
  - Good balance of compression and runtime efficiency, allows easy access to subdocuments and works with existing tools
  - Notable that Microsoft did not go with their proprietary CAB format.
  - A loose end to clean up? The ZIP format is 18 years old, but was never formally submitted for standardization.

Some comparative metrics

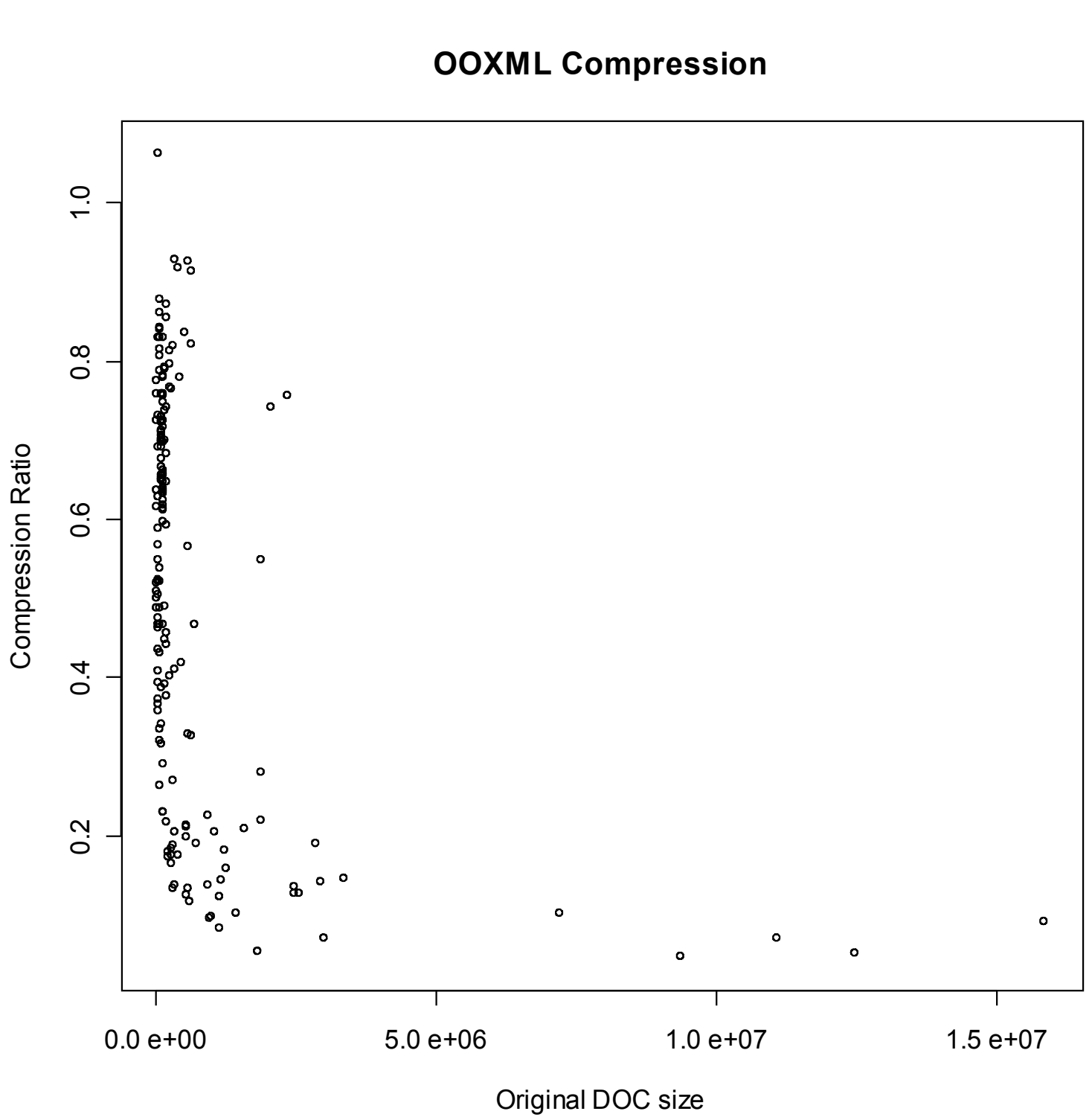
- 176 Word documents from Ecma TC45's document library
- Convert all to OOXML and ODF format
- Record:
  - Number of pages
  - ZIP size
  - Numbered of contained files
  - Numbered of contained XML files
  - Total uncompressed size of contained files
  - Total uncompressed size of contained XML files.



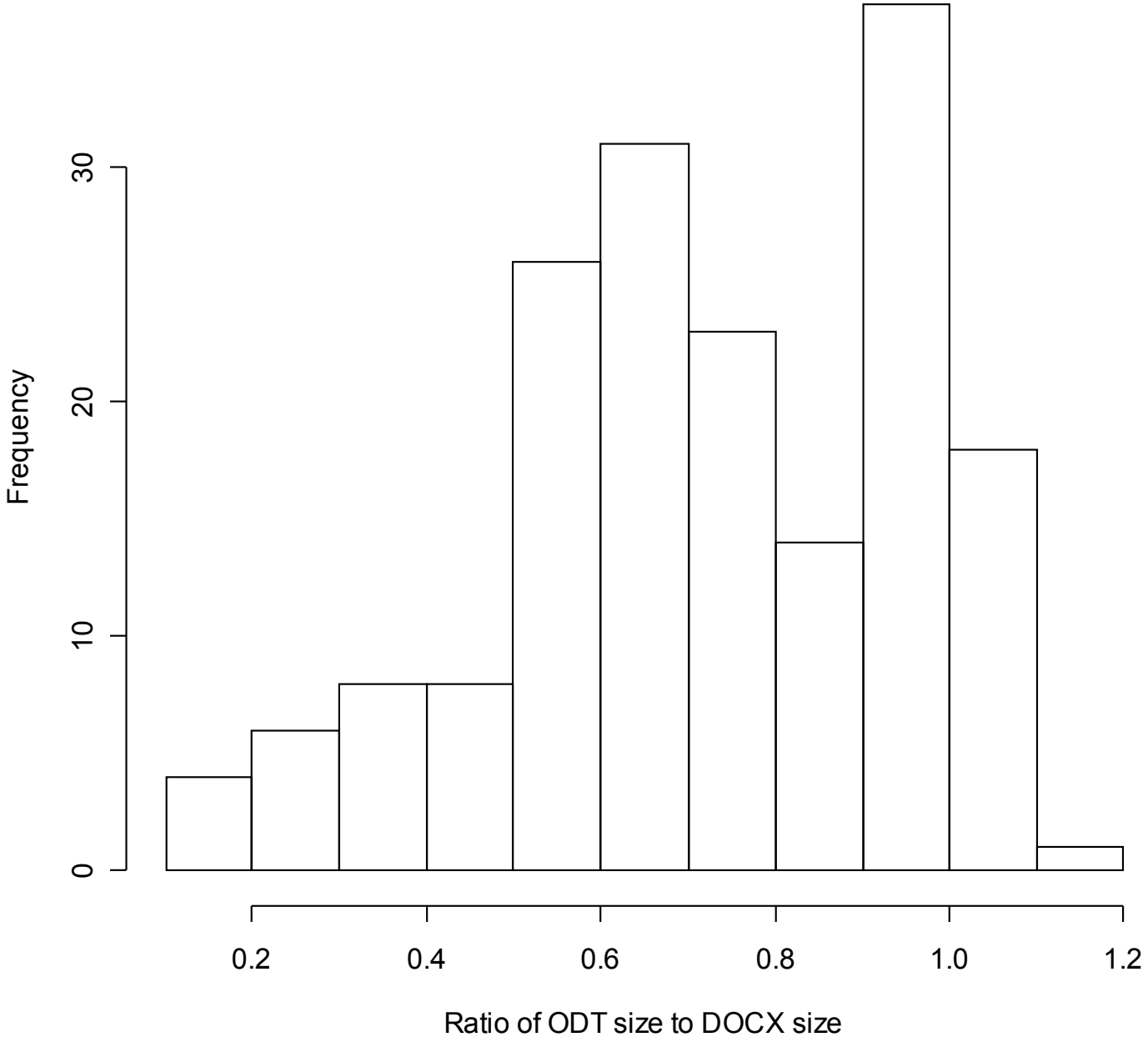
* All charts and calculations done with the excellent open source "R" environment; http://www.r-project.org/


ODF Compression
Compression Ratio
0.0
0.2
0.4
0.6
0.8
1.0
0.0 e+00
5.0 e+06
1.0 e+07
1.5 e+07
Original DOC size

Mean = 0.38


OOXML Compression
Compression Ratio
0.2
0.4
0.6
0.8
1.0
0.0 e+00
5.0 e+06
1.0 e+07
1.5 e+07
Original DOC size

Mean = 0.50


Frequency
0
10
20
30
0.2
0.4
0.6
0.8
1.0
1.2
Ratio of ODT size to DOCX size

Observed compression ratios

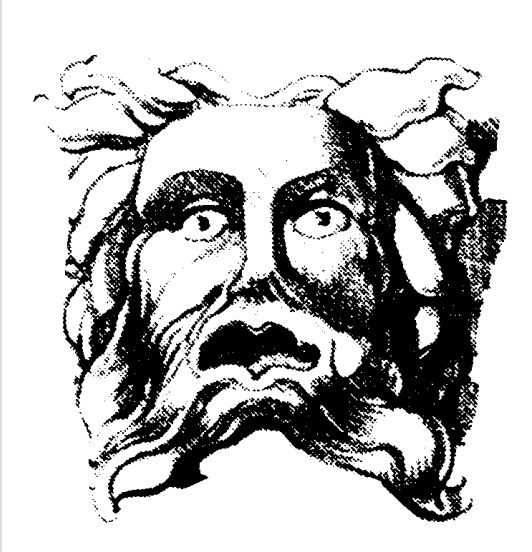
- ODF size / DOC size = 0.38
- OOXML size / DOC size = 0.50

- Net is ODF documents were smaller, on average 72% of the size of the OOXML document

- Double check with empty files
  - ODF size = 6,888 bytes
  - OOXML size = 10,001 bytes
  - ODF/OOXML = 0.69

Platform/Application Neutrality

- ODF's status is clear from the multiple implementations in the market today, from multiple vendors on multiple platforms
- The situation with OOXML is not so clear



Things to look for in the spec

- Windows
- blob
- Base64Binary
- bit
- Implementation-defined
- Undefined
- Legacy
- Backwards compatibility
- Reserved

3.2.1.68 – DEVMODE

3.2.1.68 printOptions (Print Options)

Print options for the sheet. Printer-specific settings are stored separately in the DevMode part. External documentation found here (§xx) [See MSDN reference material].

This binary blob stores printer settings in a format that can only be understood on Windows.

7.4.1.5 – Clipboard formats

7.4.1.5 cf (Clipboard Data)

The cf element specifies a base64 binary clipboard data variant type. Attributes: The format attribute specifies the clipboard data format and must be -3, -2, -1, 0, or any positive integer.

The OOXML specification does not say what any of these values mean, but merely restricts them to seemingly arbitrary numbers.

So, not only is the data stored in binary format, it is in a an unspecified format identified merely by number.

3.1.29 – Sheet-level passwords

revisionsPassword (Revisions Password)	Specifies the hash of the password required for unlocking revisions in this workbook. The hash used depends on the application. Microsoft Office Excel uses Cyclic Redundancy Check (CRC) algorithm to compute the hash from a string. The CRC is generated from an 8-bit wide character. 16-bit Unicode characters must be converted down to 8 bits before the hash is computed. In SpreadsheetML, passwords can be up to 255 letters, numbers, spaces, and symbols. Note that passwords are case sensitive. The possible values for this attribute are defined by the ST_UnsignedShortHex simple type (§3.17.87).
workbookPassword (Workbook Password)	Specifies the hash of the password required for opening this workbook. The hash used depends on the application. Microsoft Office Excel uses Cyclic Redundancy Check (CRC) algorithm to compute the hash from a string. The CRC is generated from an 8-bit wide character. 16-bit Unicode characters must be converted down to 8 bits before the hash is computed.

Problem is a CRC is not defined unless you give the polynomial as well as the bit length. We would also need to know exactly how Unicode characters are to be turned into 8 bit ones. Hex encode? Throw out the high bits? Two bytes for each character?

Insufficient information is disclosed to allow interop.

2.7.2.17 – Locale Signature

C:

```
typedef struct tagLOCALESIGNATURE {
  DWORD  lsUsb[4];
  DWORD  lsCsbDefault[2];
  DWORD  lsCsbSupported[2];
} LOCALESIGNATURE, *PLOCALESIGNATURE;
```

XML:

```
<w:font w:name="Times New Roman">
  <w:sig w:usb0="20002A87" w:usb1="80000000" w:usb2="00000008"
w:usb3="00000000" w:csb0="000001FF" w:csb1="00000000" />
  …
</w:font>
```

Can you tell the difference?

2.7.2.17 – Locale Signature

Attributes	Description
csb0 (Lower 32 Bits of Code Page Bit Field)	Specifies a four digit hexadecimal encoding of the first 32 bits of the 64-bit code-page bit field that identifies which specific character sets or code pages are supported by the parent font. Each bit in this 32 bits represents the following code page:

Bit	Description
0	Latin 1
1	Latin 2: Eastern Europe
2	Cyrillic
3	Greek
4	Turkish
5	Hebrew
6	Arabic

Bitmasks in XML ?!

Conformance according to ODF

- Documents may contain elements in foreign namespaces
- Document Consumers must be able to read any document which would have been valid if all foreign markup were removed.
- Document Consumers may preserve such foreign markup

Implied validation pipeline

1) Load XML
2) Remove elements and attributes that are not in an ODF namespace
3) Validate the resulting document according to ODF schema

OOXML's approach

- Part 5 of recently posted 1.4 draft OOXML
- Much more complex, 37 pages describing a sophisticated validation pipeline
- A new XML ML for Markup Compatibility

Markup Compatibility

- Ignorable
- MustUnderstand
  - List of namespaces which either can be safely ignored, or must not be ignored
- PreserveElements
- PreserveAttributes
  - List of elements or namespaces which should be preserved in editing, even if the namespace is ignorable
- ProcessContent
  - Content is processed even if element is ignored

AlternateContent/Choice/Fallback

- Similar to a “switch/case/default” construct

See example compliance.xml

Implied validation pipeline

1) Verify validity of Markup Compatibility markup
2) Process MustUnderstand's and generate errors if needed
3) Remove Choice/Fallback markup for cases that were not used
4) Namespace subsumption – remove markup in obsolete namespaces and replace with new namespaces
5) Process the Ignorable's removing the ones you don't understand
6) Process the ProcessContent content

My take on it

- Intriguing idea, but not fully baked (still draft).
- Gives a lot of (too much?) flexibility in negotiating fidelity of representation based on capabilities of the consumer.
- Danger – it essentially lets a producer rewrite the standard and the schema outside of a standards setting. Could enable Office to maintain a two-tier file format, with the high-fidelity version not documented, and the low-fidelity version available only in <Fallback>. Remember RTF?

Performance

- How to measure the performance of a format versus the performance of an application?

- Some factors:
  - Number of XML files in the Zip which must be parsed
  - Size of the XML files
  - Preprocessing required to resolve Ignorable, MustUnderstand, etc.

  - Can't give an absolute answer, since not all consumers of a the document are attempting the same thing.

Licensing Problem

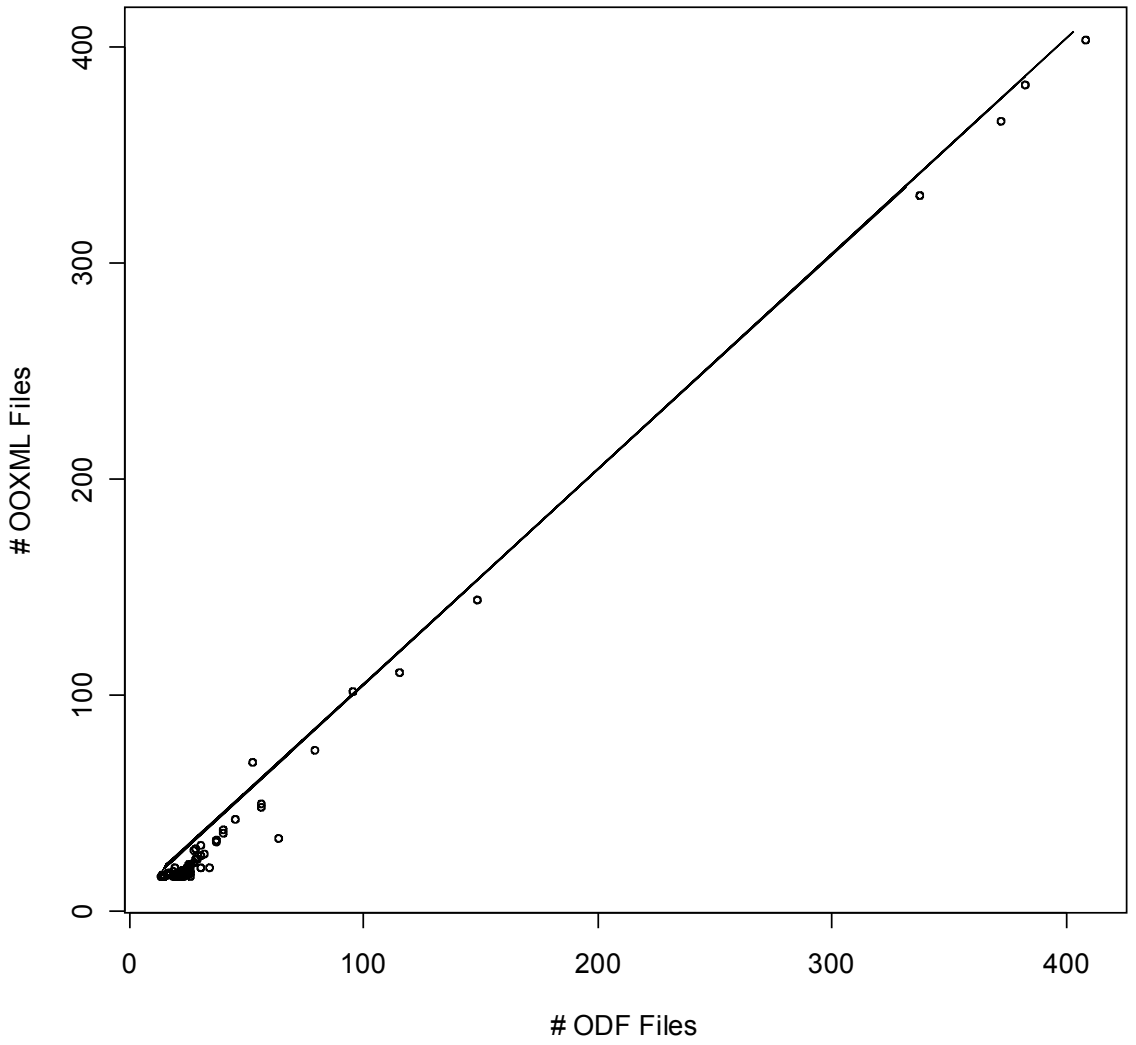
- The only implementation of OOXML is the Office 2007 beta, and the End User License Agreement (EULA) has this restriction:

"7. SCOPE OF LICENSE. ...You may not disclose the results of any benchmark tests of the software to any third party without Microsoft's prior written approval"

Solution: test the XML

- Take 176 working documents from Ecma TC45's document library, in Word format
- Convert to ODF and OOXML formats
- Collect static and runtime metrics for these document pairs, including:
  - Size, compressed and uncompressed
  - Size of just the XML
  - Number of pages
  - Number of files in the Zip
  - Time to parse the XML – this is the core of any tool which will consume these documents so large differences here will directly map into large differences in applications

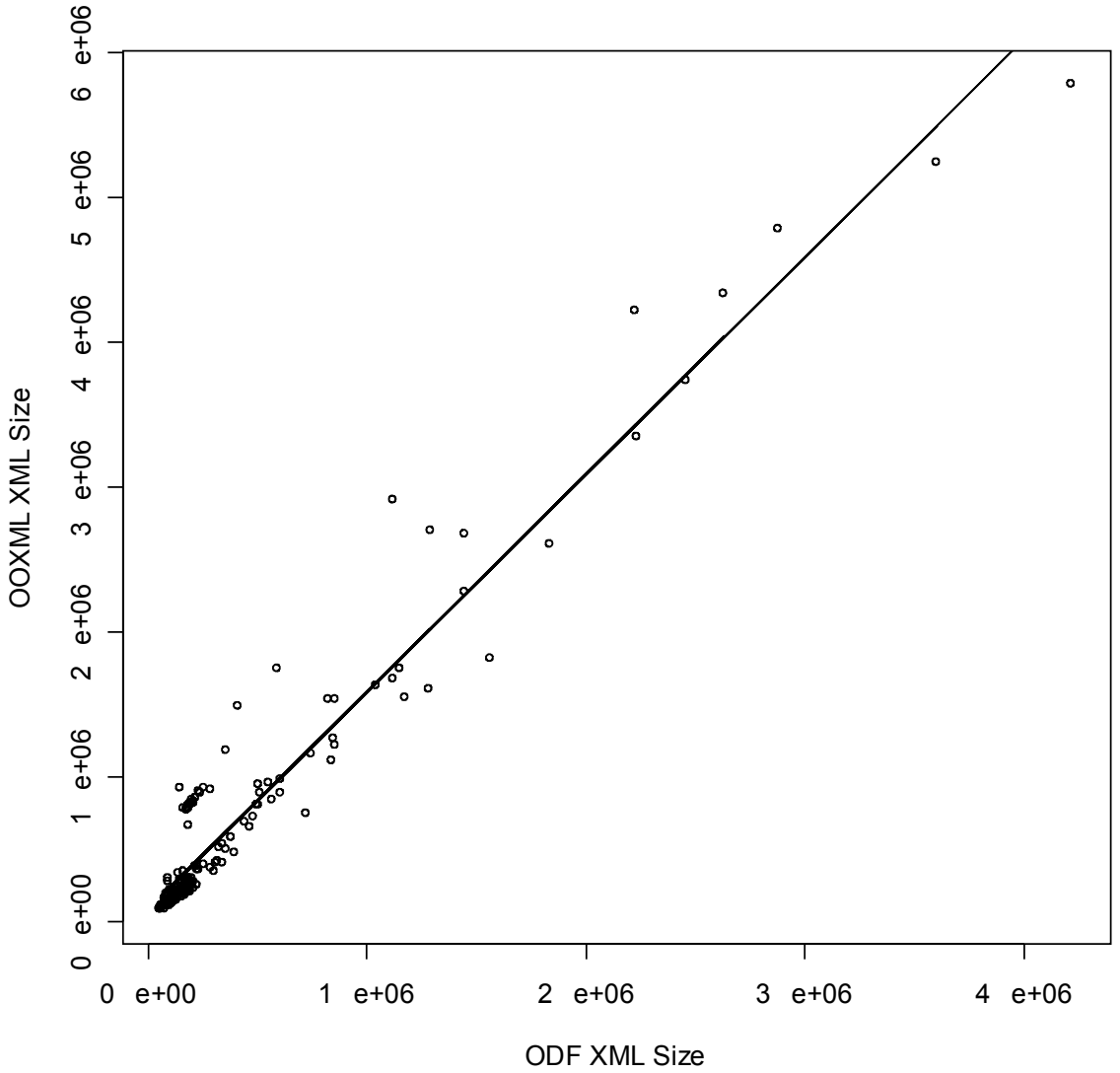
Number of files in the Zip



OOXML files = 5.7 + ODF files

R^2 = 0.9958

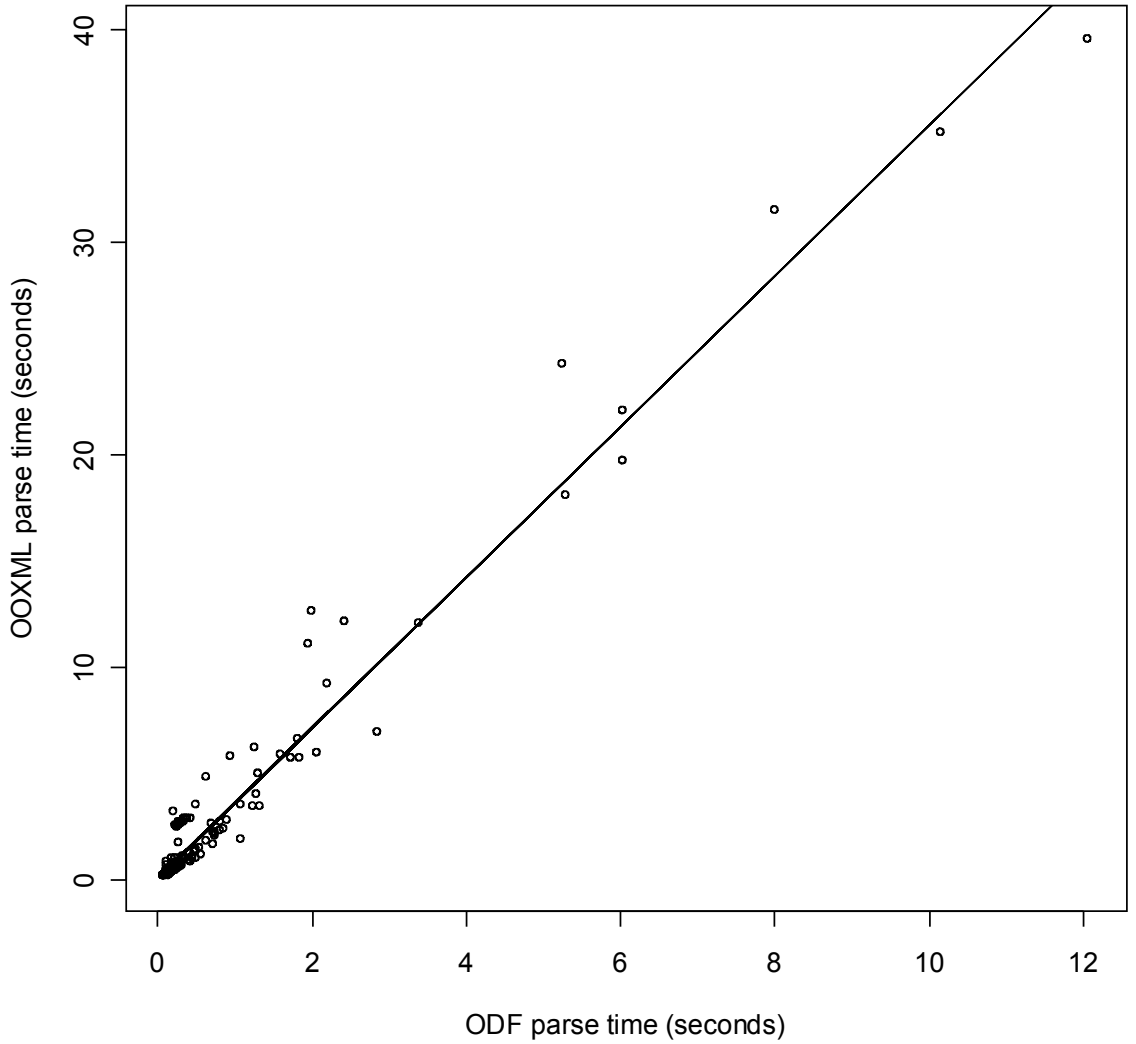
Total Size of the XML's



OOXML size = 82,000 bytes + 1.5 * ODF size

R^2 = 0.92

Net effect on parse time

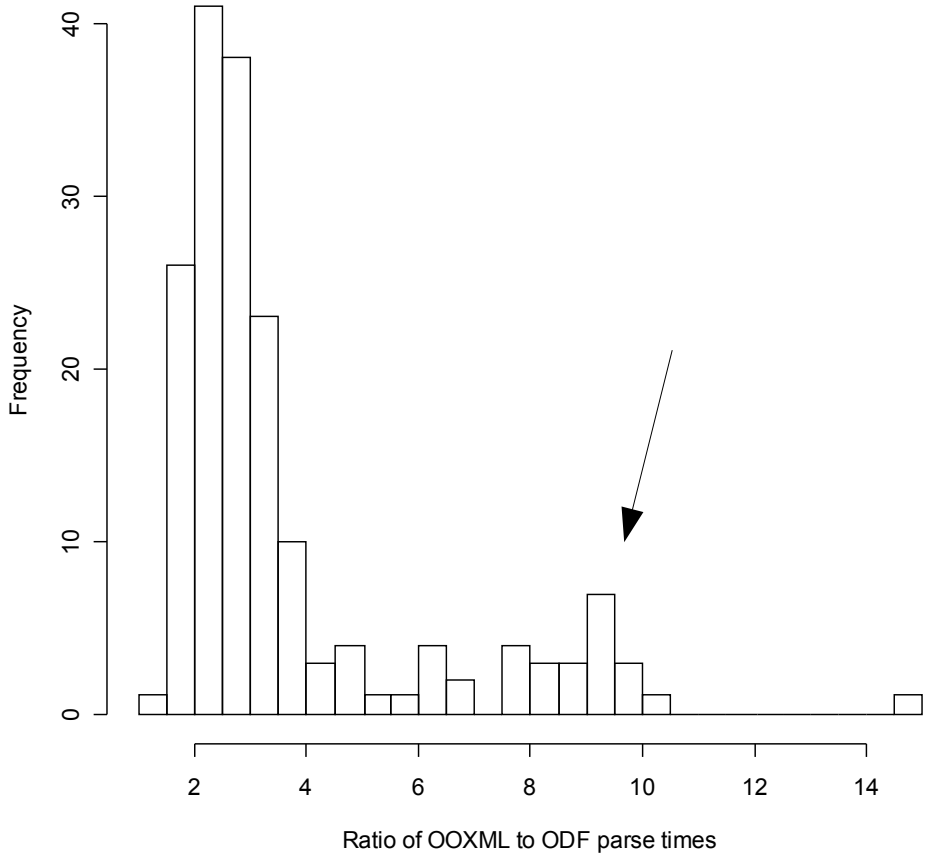


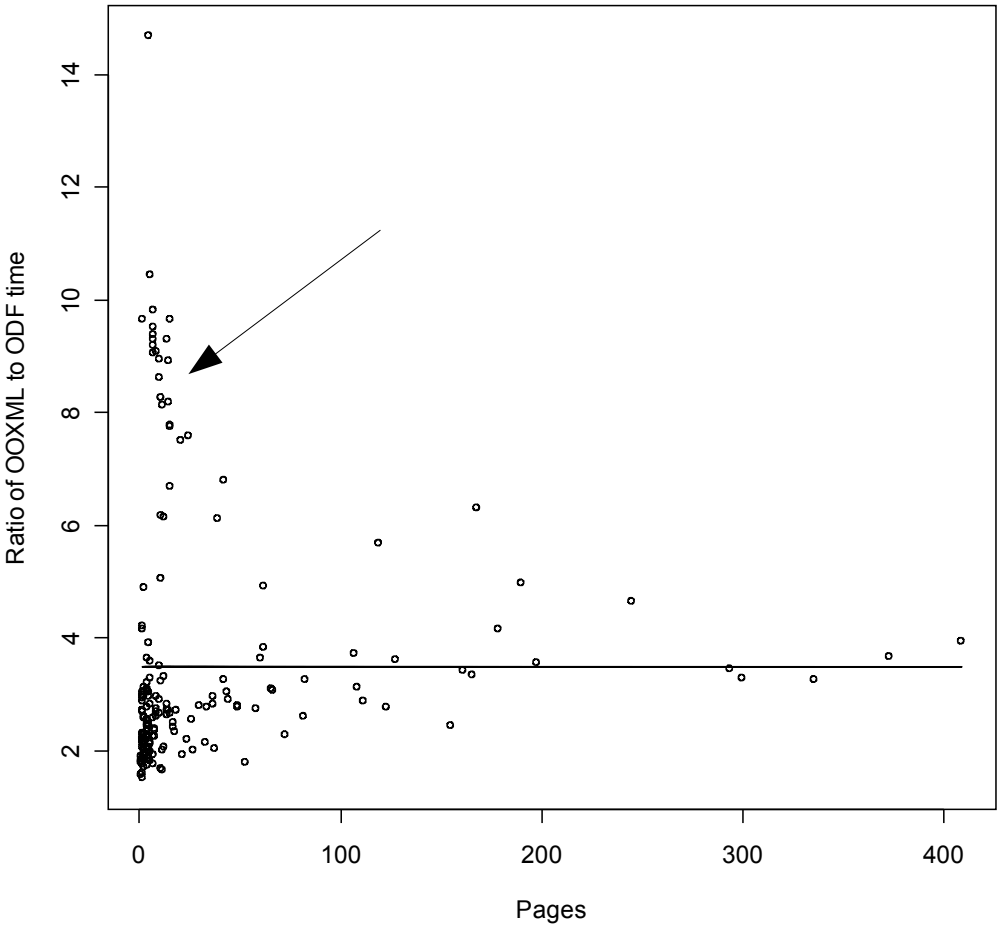
OOXML time = 3.5 * ODF time

R^2 = 0.9596

Time is time to parse all XML files in the Zip archive with Python's minidom

Bimodal behavior?





Performance Conclusions

- Choice of an XML parser is key
  - ODF files have larger, but fewer XML files
  - OOXML have many small XML files
  - Many (most?) parsers are not well optimized for the 2nd case.
- Be a wise consumer of benchmark data
  - Beware of tests which confuse application performance for file format performance
  - Look to see if the documents used in the test are typical.
  - Consider the performance of all types of applications, not just heavy-weight desktop editor

The End

Thank you