A LOOK INTO 30 YEARS OF MALWARE DEVELOPMENT FROM A SOFTWARE METRICS PERSPECTIVE

Alejandro Calleja[1], Juan Tapiador[1], Juan Caballero[2]

[1] Universidad Carlos III de Madrid, Spain

[2] IMDEA Software Institute, Spain

RAID 2016 – 19TH INTERNATIONAL SYMPOSIUM ON RESEARCH IN ATTACKS, INTRUSIONS AND DEFENSES

OVERVIEW

OVERVIEW

- The malware problem
  - By the numbers: 1.7B known malware samples, 317M new samples discovered in 2014 (Symantec, 2015)
  - A commodity

OVERVIEW

- The malware problem
  - By the numbers: 1.7B known malware samples, 317M new samples discovered in 2014 (Symantec, 2015)
  - A commodity
- Increasing access to malware source code

OVERVIEW

- The malware problem
  - By the numbers: 1.7B known malware samples, 317M new samples discovered in 2014 (Symantec, 2015)
  - A commodity
- Increasing access to malware source code
- What can we learn from it?
  - This work: **measurements** of malware as a software product and its evolution over the last 30 years

OUTLINE OF THE TALK

1. Software metrics
2. Our dataset
3. Analysis
4. Conclusions

SOFTWARE METRICS

SOFTWARE METRICS

- Measuring **software size**

SOFTWARE METRICS

- Measuring **software size**
  - **SLOC** (Source Lines Of Code)

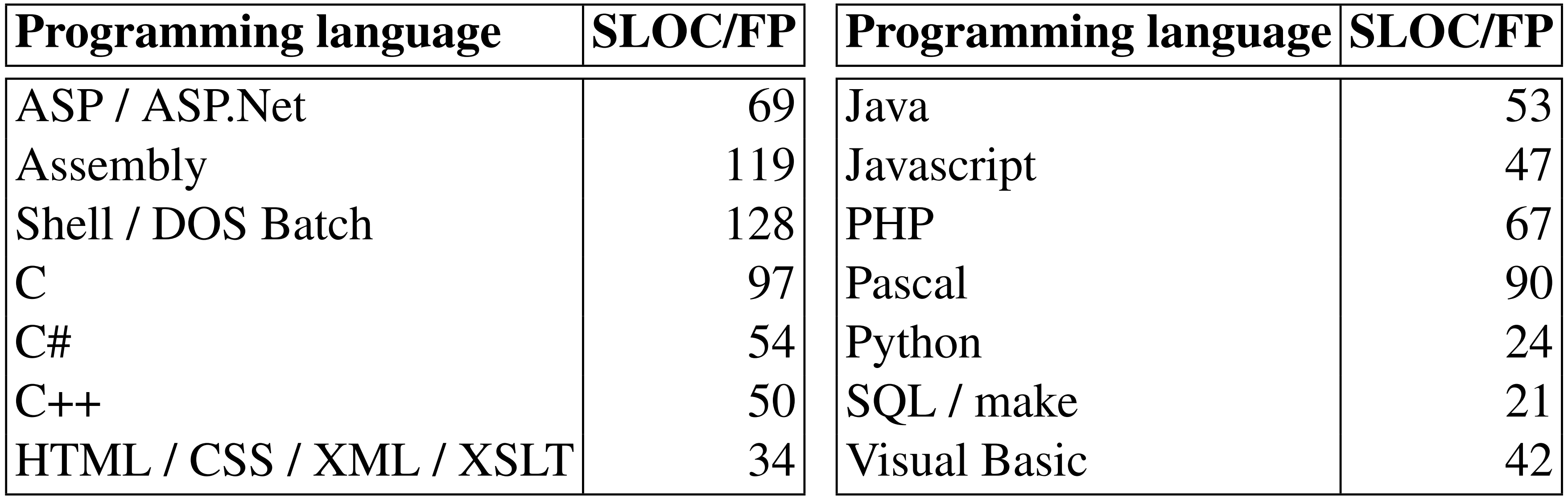
SOFTWARE METRICS

- Measuring **software size**
  - **SLOC** (Source Lines Of Code)
  - **FP** (Function Points)

SOFTWARE METRICS

- Measuring **software size**
  - **SLOC** (Source Lines Of Code)
  - **FP** (Function Points)

↔ "Backfiring"



Programming language	SLOC/FP
ASP / ASP.Net	69
Assembly	119
Shell / DOS Batch	128
C	97
C#	54
C++	50
HTML / CSS / XML / XSLT	34

Programming language	SLOC/FP
Java	53
Javascript	47
PHP	67
Pascal	90
Python	24
SQL / make	21
Visual Basic	42

SOFTWARE METRICS

SOFTWARE METRICS

- Estimating effort

SOFTWARE METRICS

- Estimating effort
  - COCOMO (Constructive Cost Model), 1980s
    Basic, Intermediate, Advanced

SOFTWARE METRICS

- Estimating **effort**
  - COCOMO (Constructive Cost Model), 1980s
    Basic, Intermediate, Advanced

$$E = a_b(\mathrm{KLOC})^{b_b}$$

$$D = c_b E^{d_b}$$

$$P = \frac{E}{D}$$

Software Project	a_b	b_b	c_b	d_b
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

SOFTWARE METRICS

SOFTWARE METRICS

- Estimating **complexity and maintainability**

SOFTWARE METRICS

- Estimating **complexity and maintainability**
  - McCabe's Cyclomatic Complexity
    - No. linearly independent paths in the CFG
    - Should be <10 for each module

SOFTWARE METRICS

- Estimating **complexity and maintainability**
  - McCabe's Cyclomatic Complexity
    - No. linearly independent paths in the CFG
    - Should be <10 for each module
  - Maintainability Index (MI)
    - Value in [0, 100]
    - Used by Visual Studio, JSComplexity, Radon
    - Not agreed upon thresholds (e.g., VS flags MI < 20)

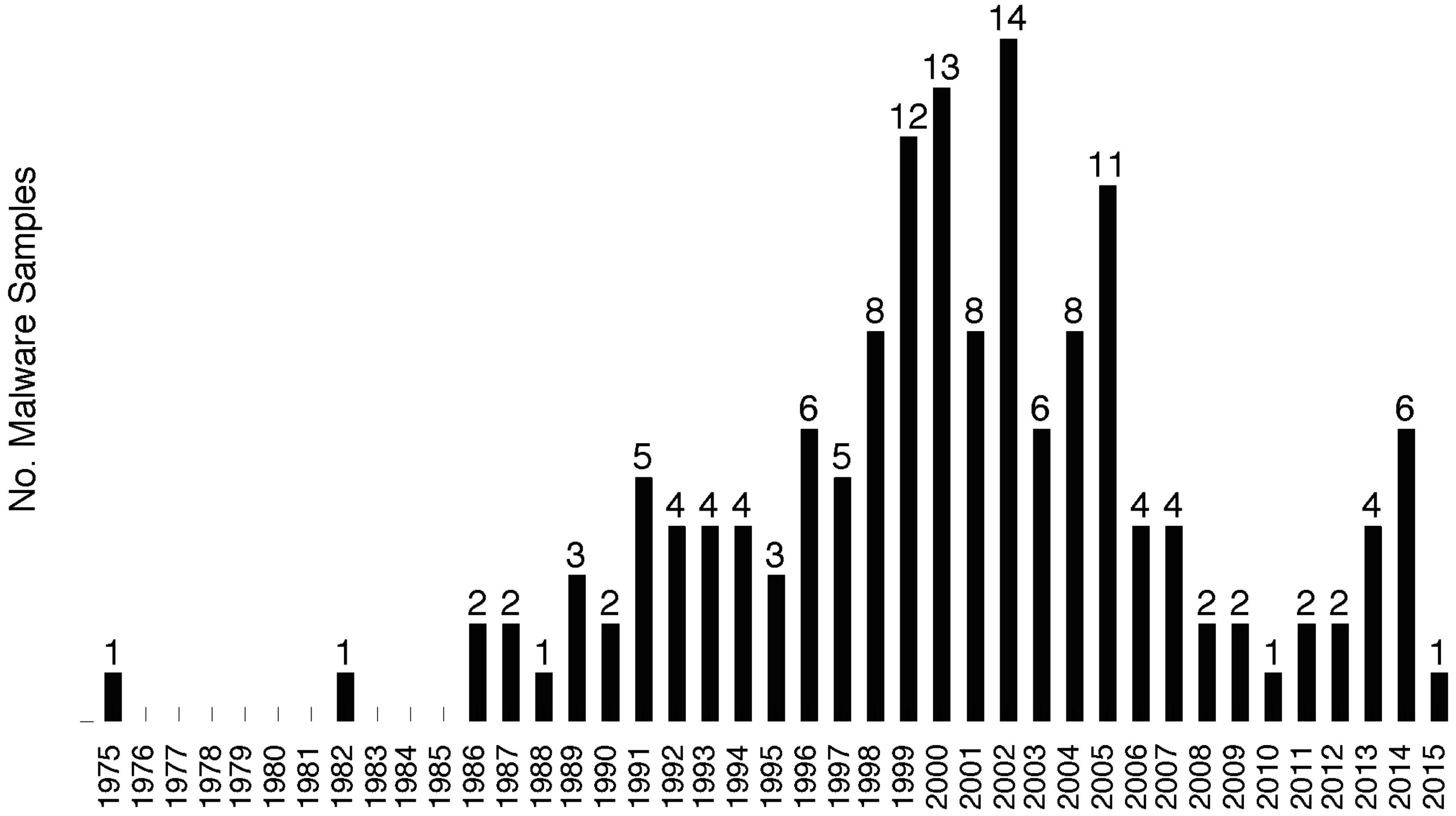
OUR DATASET

- 151 samples of malware source code collected over several months in 2015
- Sources:
  - Malware collection sites (e.g., VX Heaven)
  - Github
  - Classical e-zines (e.g., 29A)
  - Other malware exchange forums available on the web
- Original collection contained 210 samples but around 30% didn’t survive:
  - Turned out to be fake
  - Couldn’t be compiled & tested

OUR DATASET

Year	No. samples
1975	1
1982	1
1985-1994	28 (~18.5%)
1995-2005	94 (~62.3%)
2006-2015	27 (~17.8%)

Category	No. samples
Viruses	92
Worms	33
Trojans	11
RATs	9
MacroViruses	3
Botnets	3



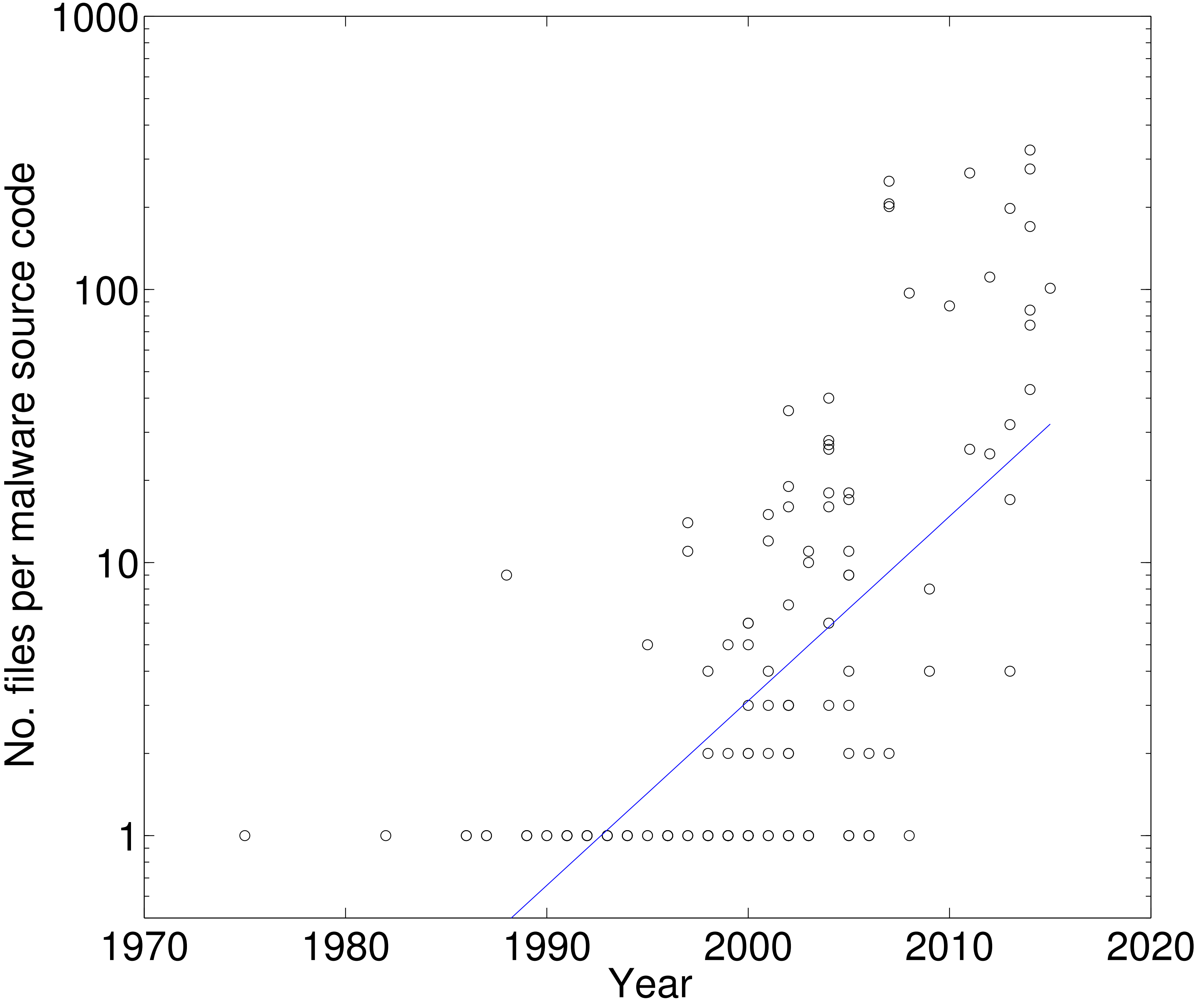
ANALYSIS

Source code analytics	Number of files
	SLOC count
	Density of comments
	FP count
	Programming languages
Cost estimates	Effort
	Development time
	Team size
Code quality	Complexity
	Maintainability
Comparison with regular software	

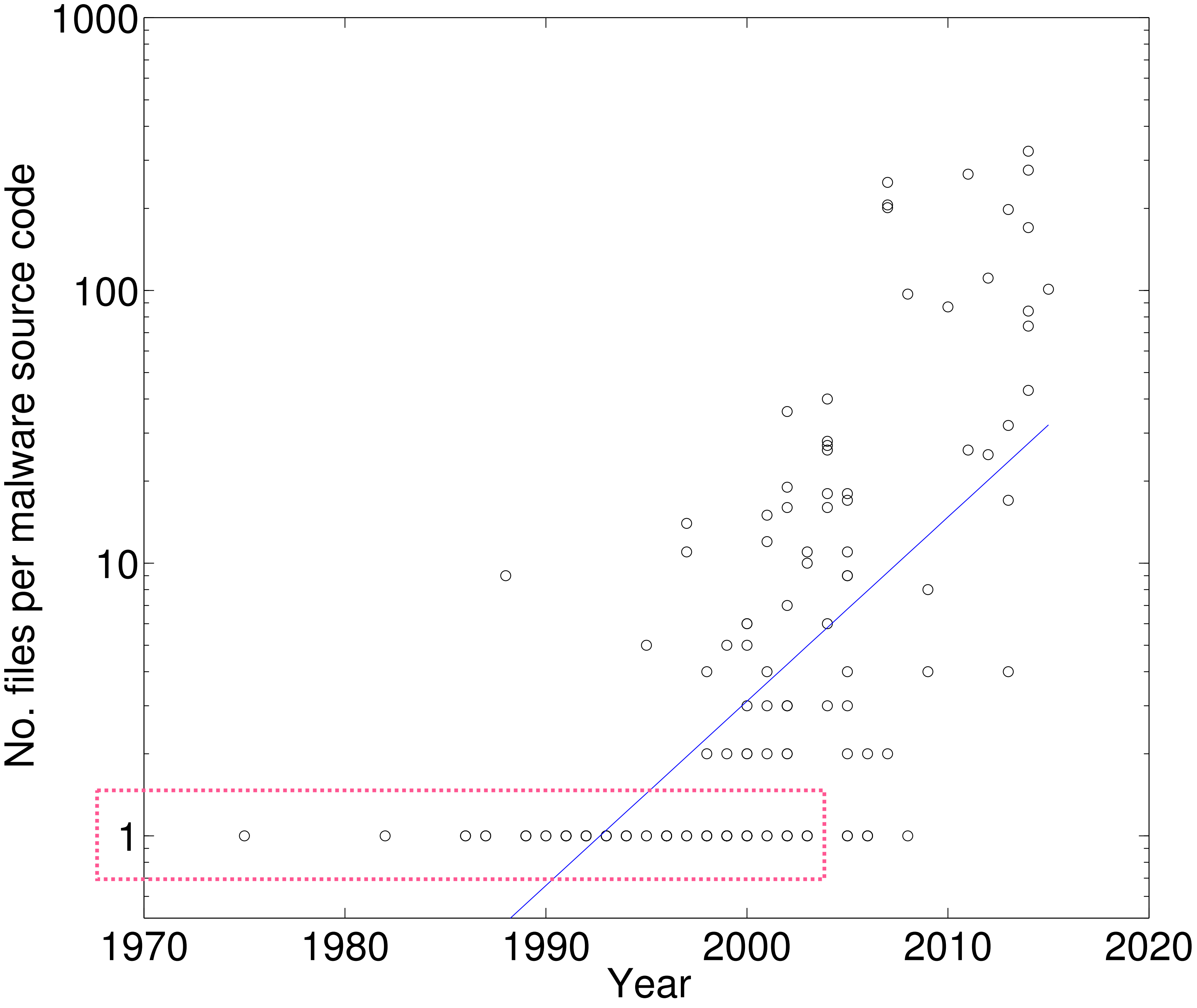
ANALYSIS

Source code analytics	Number of files
	SLOC count
	Density of comments
	FP count
	Programming languages
Cost estimates	Effort
	Development time
	Team size
Code quality	Complexity
	Maintainability
Comparison with regular software	

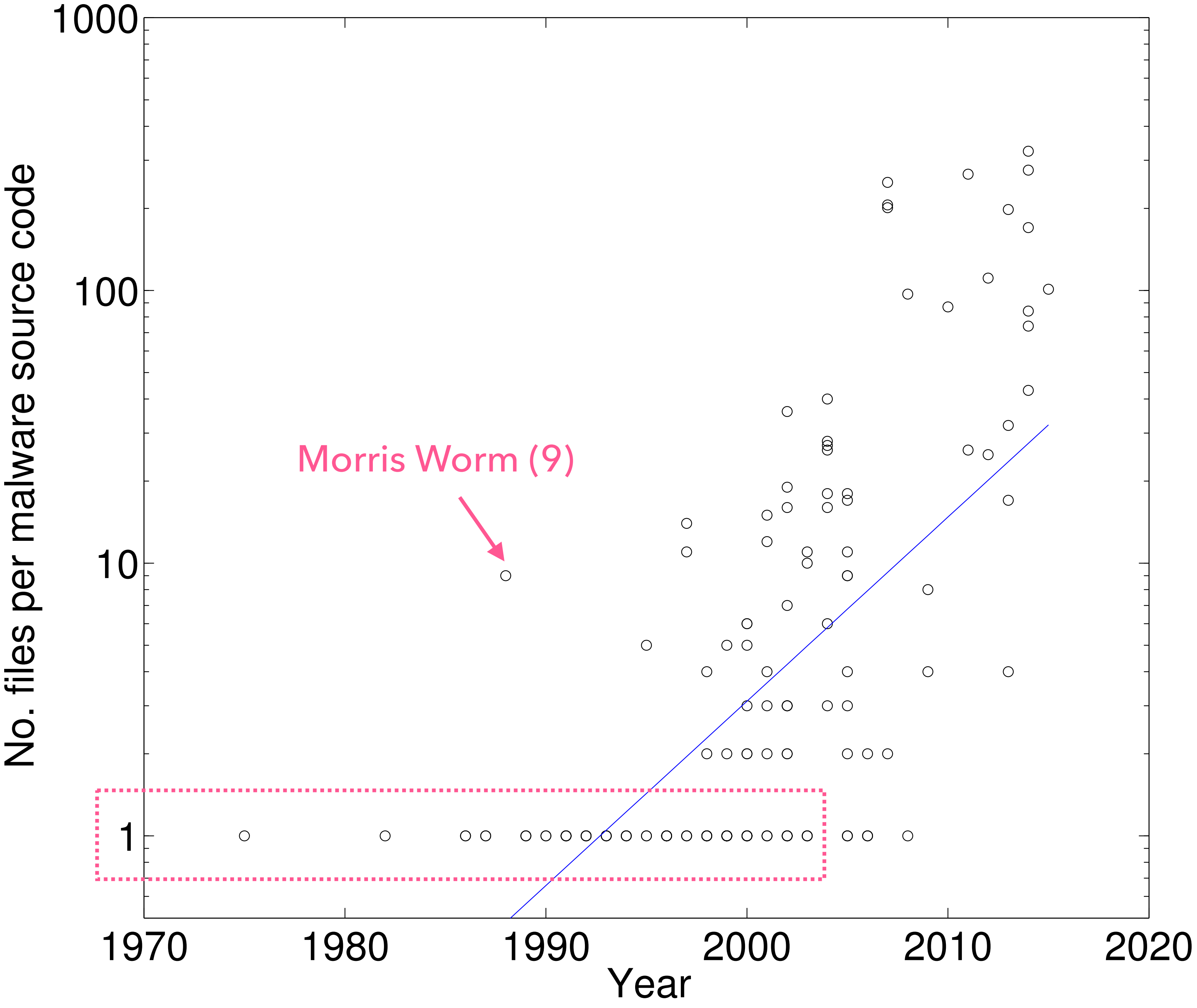
ANALYSIS


1000
100
10
1
No. files per malware source code
1970
1980
1990
2000
2010
2020
Year

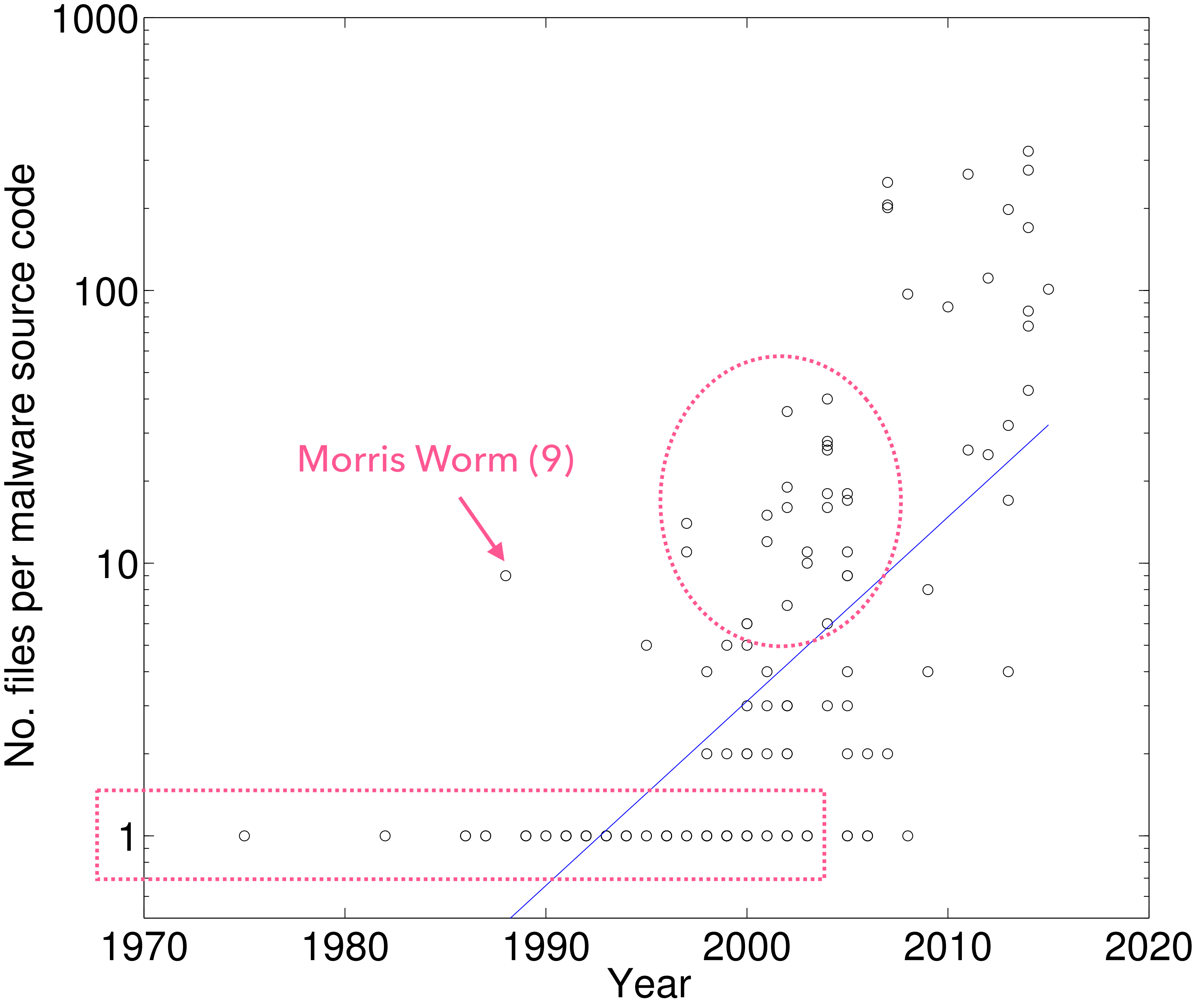
ANALYSIS


No. files per malware source code
1000
100
10
1
1970
1980
1990
2000
2010
2020
Year

ANALYSIS

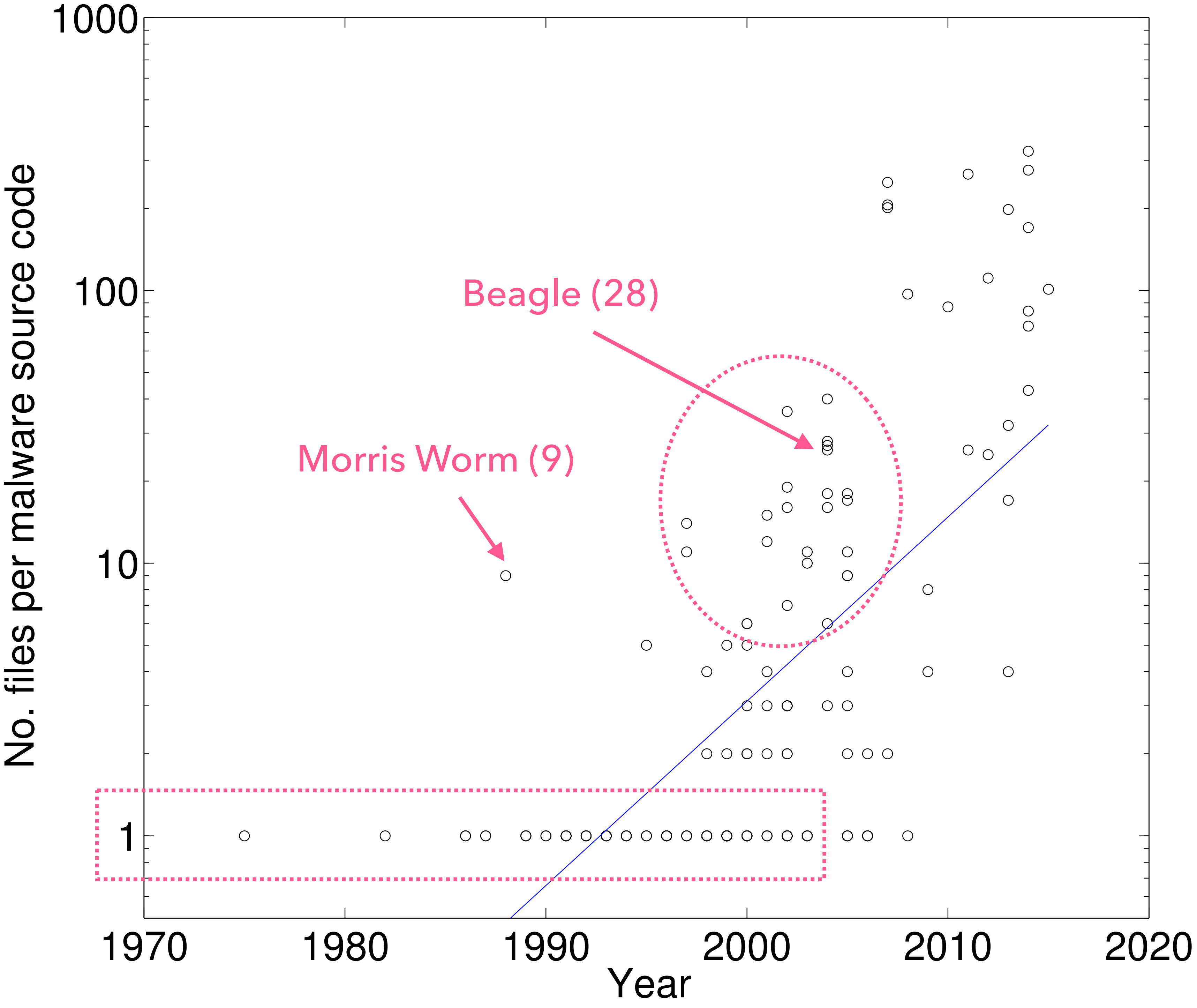

1000
100
10
1
No. files per malware source code
1970
1980
1990
2000
2010
2020
Year
Morris Worm (9)

ANALYSIS

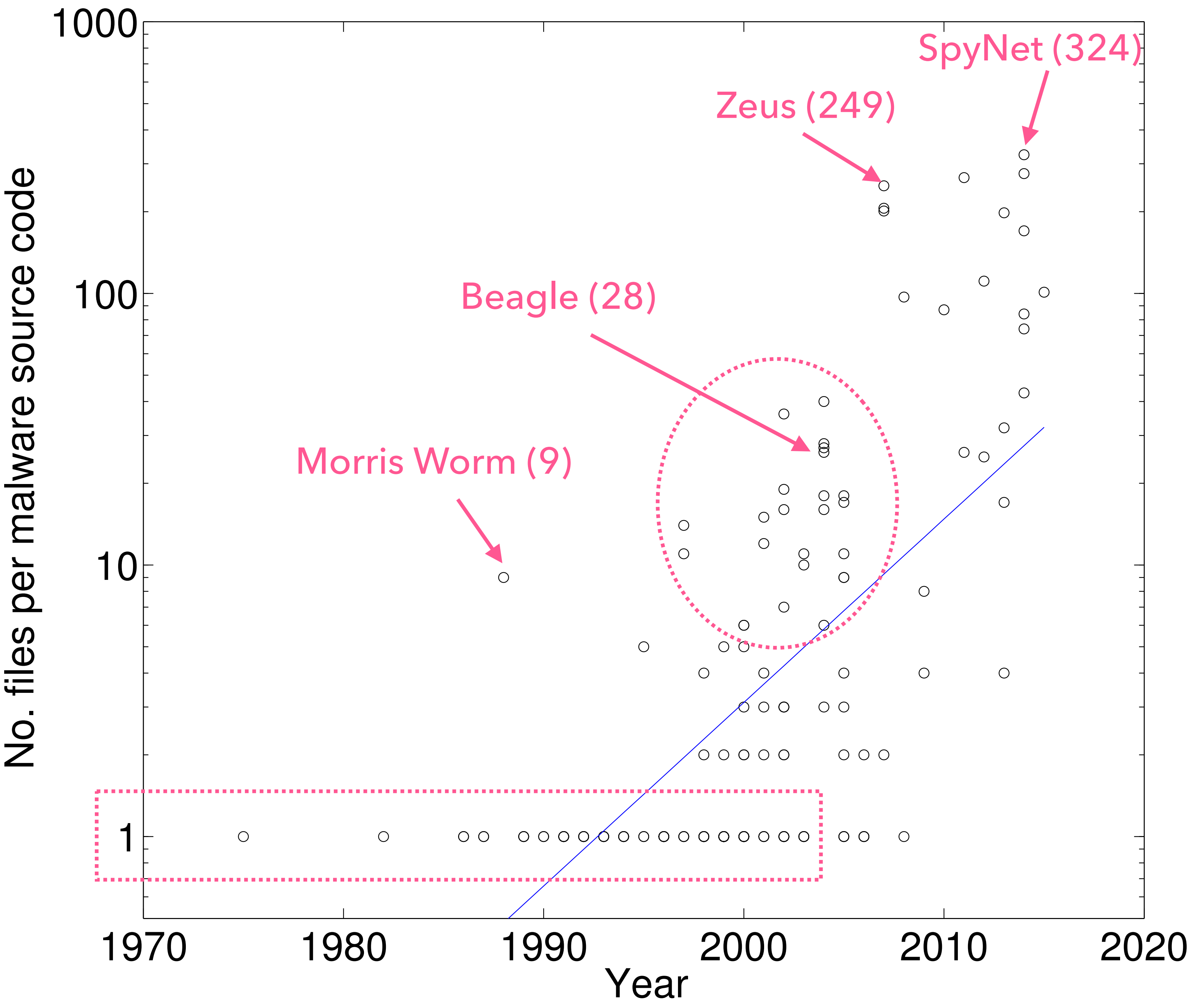


1000
100
10
1
No. files per malware source code
1970
1980
1990
2000
2010
2020
Year
Morris Worm (9)

ANALYSIS

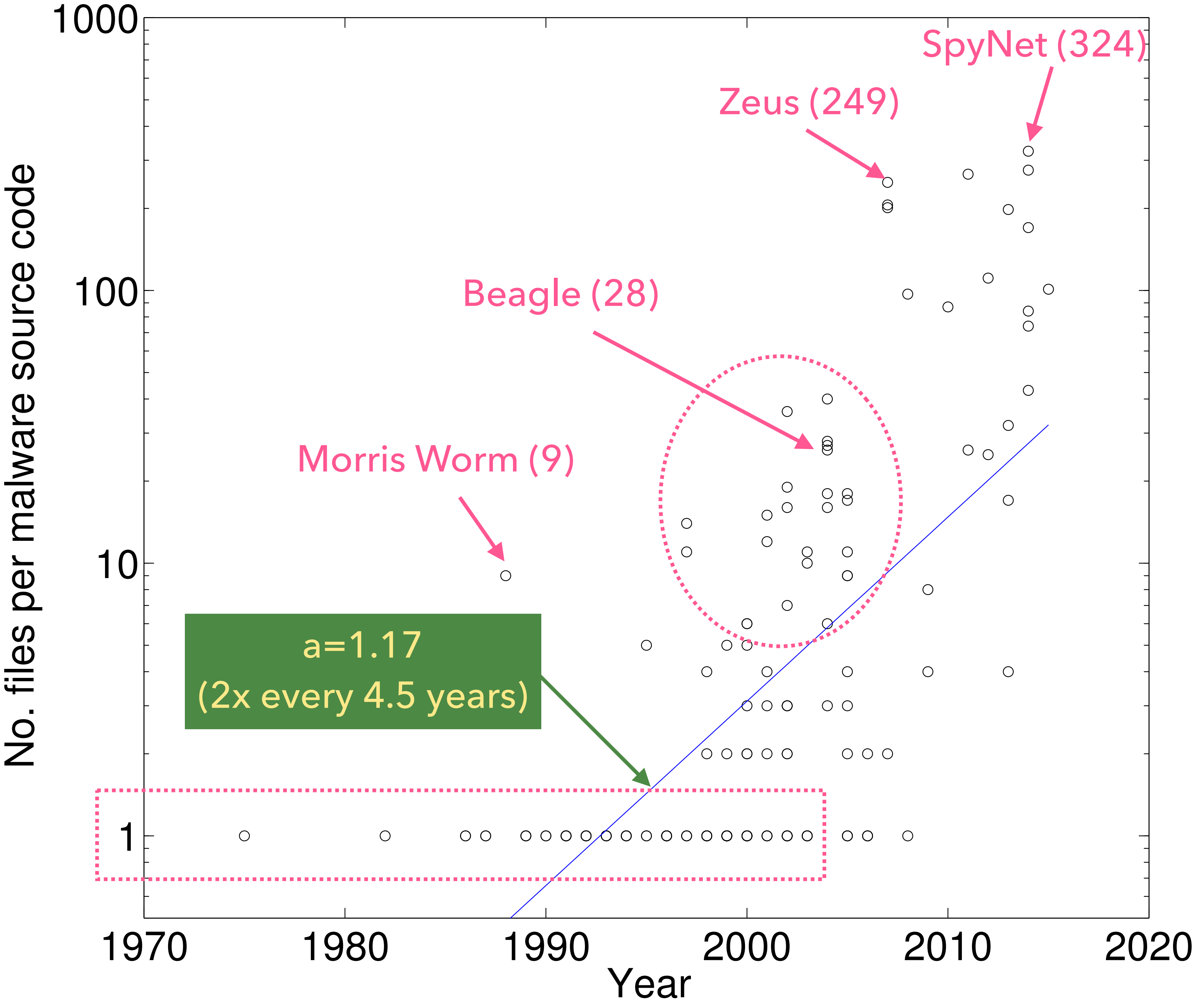

1000
100
10
1
No. files per malware source code
1970
1980
1990
2000
2010
2020
Year
Beagle (28)
Morris Worm (9)

ANALYSIS



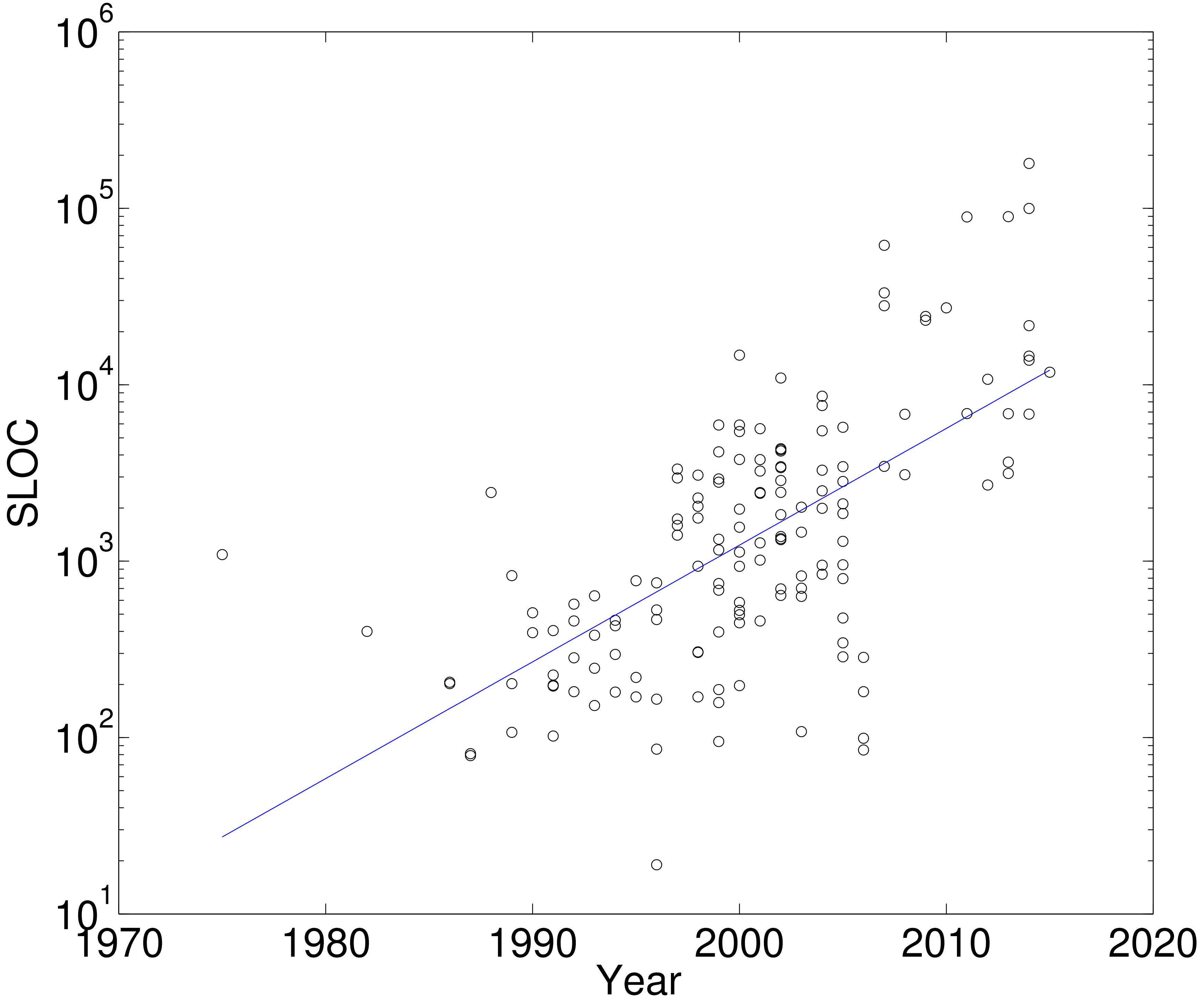
SpyNet (324)
Zeus (249)
Beagle (28)
Morris Worm (9)
1000
100
10
1
No. files per malware source code
1970
1980
1990
2000
2010
2020
Year

ANALYSIS



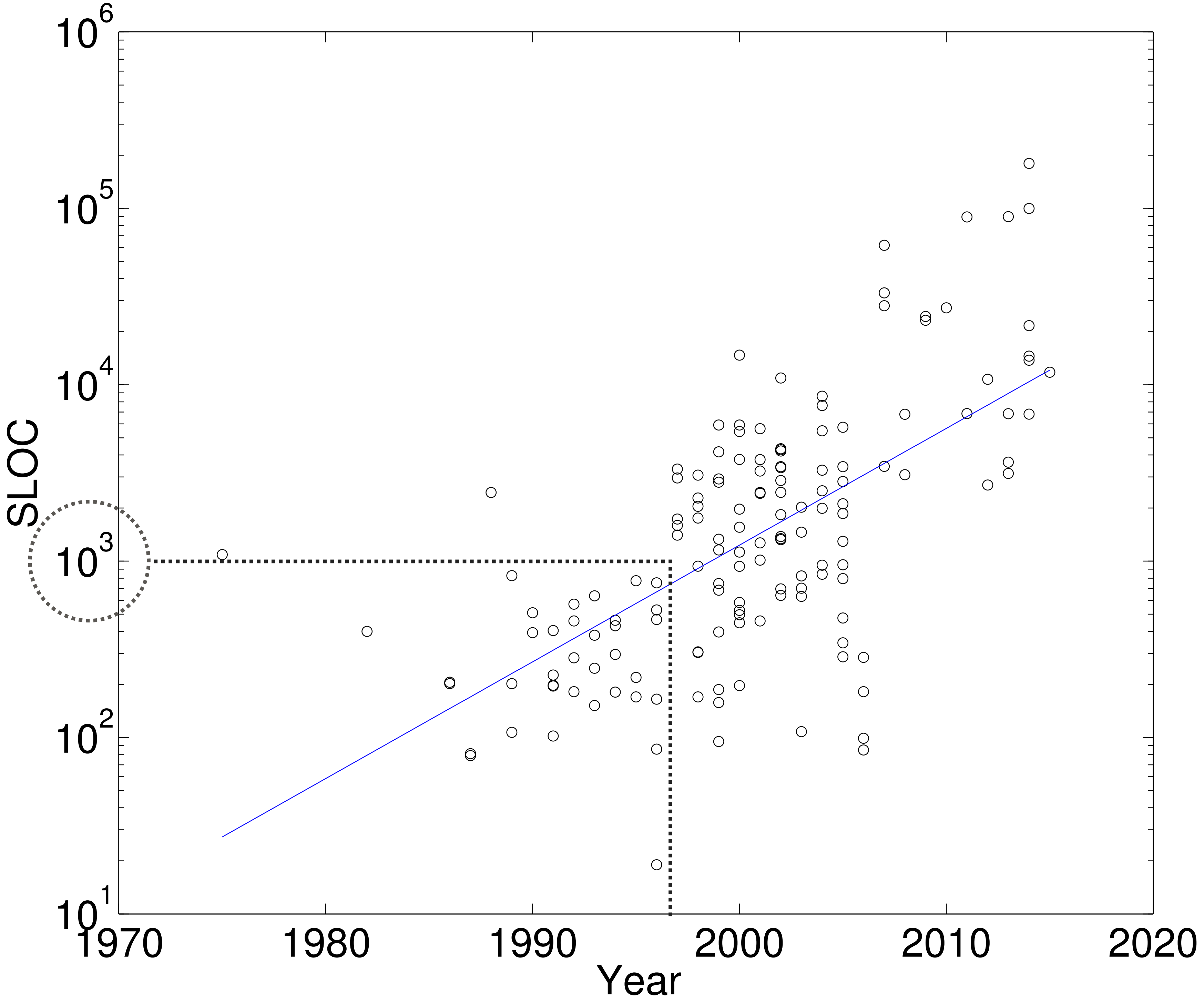
SpyNet (324)
Zeus (249)
Beagle (28)
Morris Worm (9)
a=1.17
(2x every 4.5 years)
1000
100
10
1
No. files per malware source code
1970
1980
1990
2000
2010
2020
Year

ANALYSIS



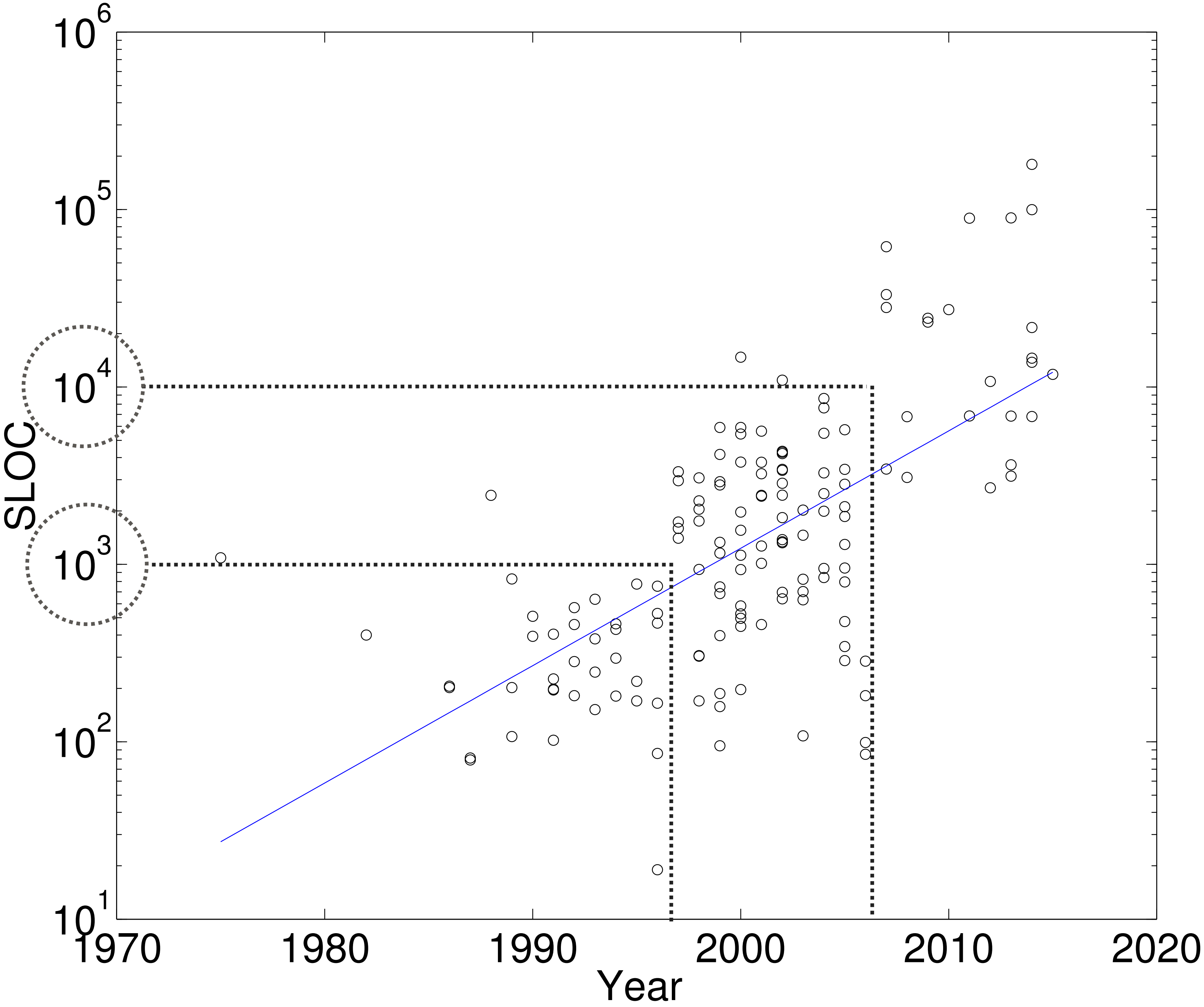
10^6
10^5
10^4
10^3
10^2
10^1
SLOC
1970
1980
1990
2000
2010
2020
Year

ANALYSIS

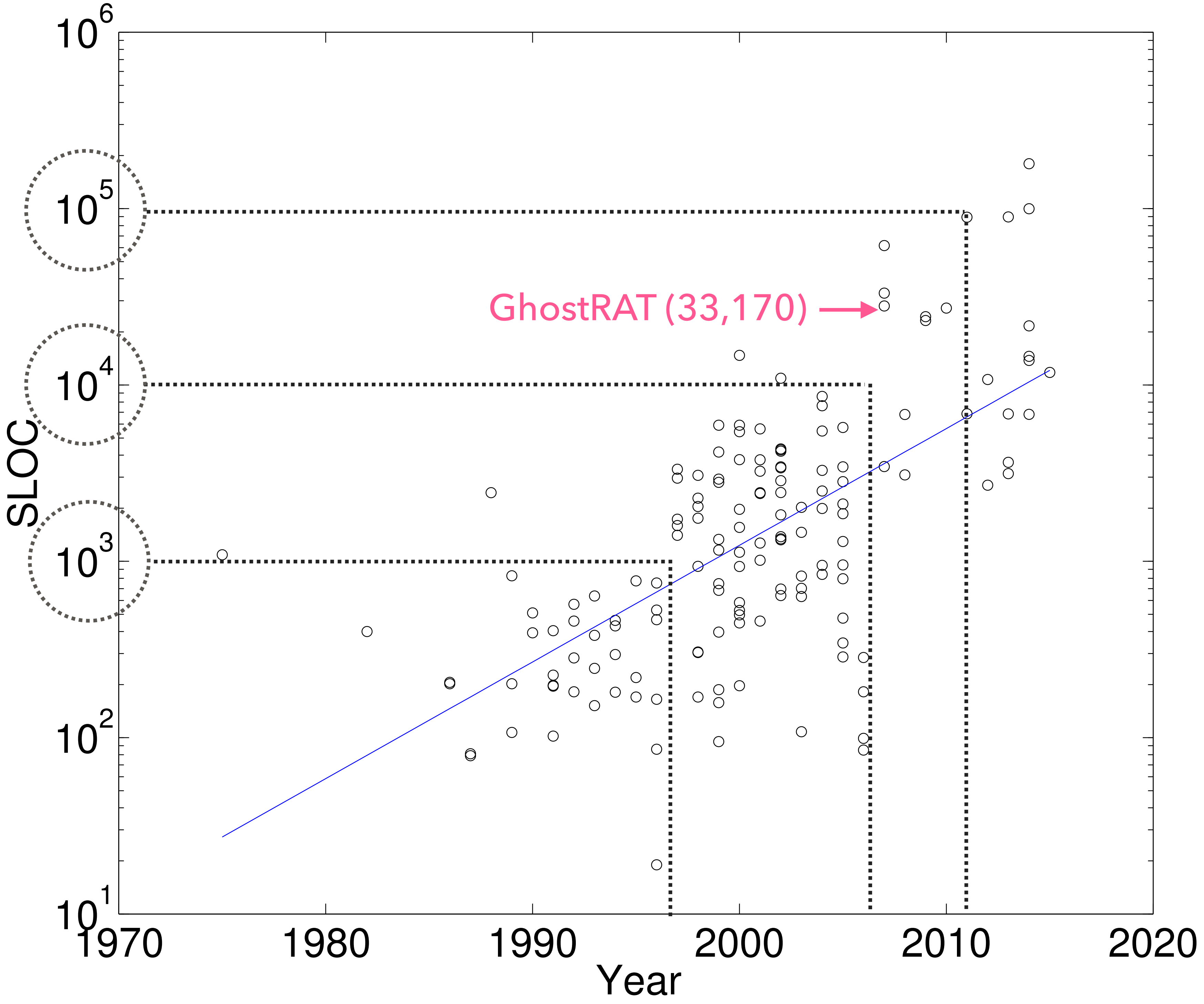


10^6
10^5
10^4
10^3
10^2
10^1
SLOC
1970
1980
1990
2000
2010
2020
Year

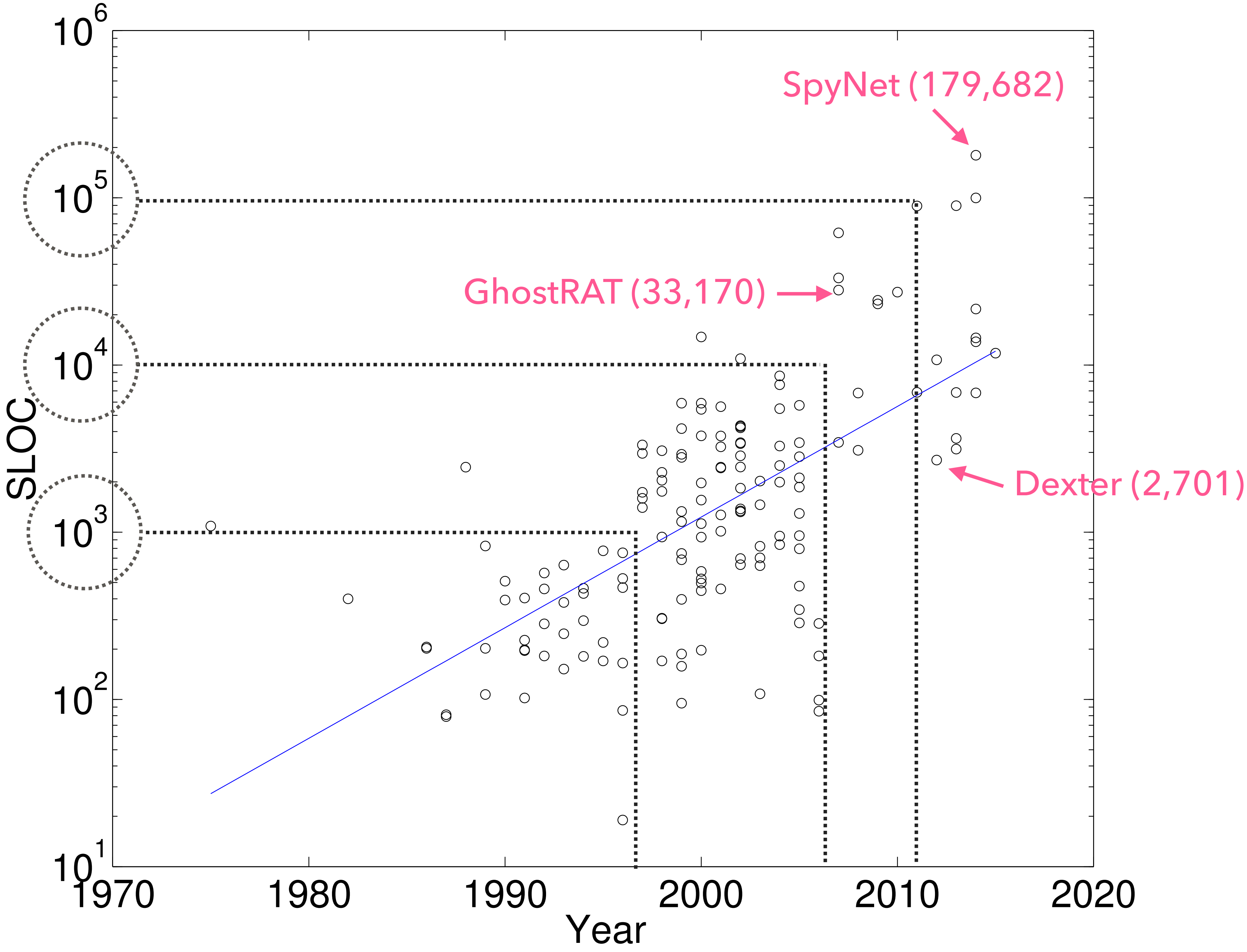
ANALYSIS


10^6
10^5
10^4
10^3
10^2
10^1
SLOC
1970
1980
1990
2000
2010
2020
Year

ANALYSIS


10^6
10^5
10^4
10^3
10^2
10^1
SLOC
1970
1980
1990
2000
2010
2020
Year
GhostRAT (33,170)

ANALYSIS

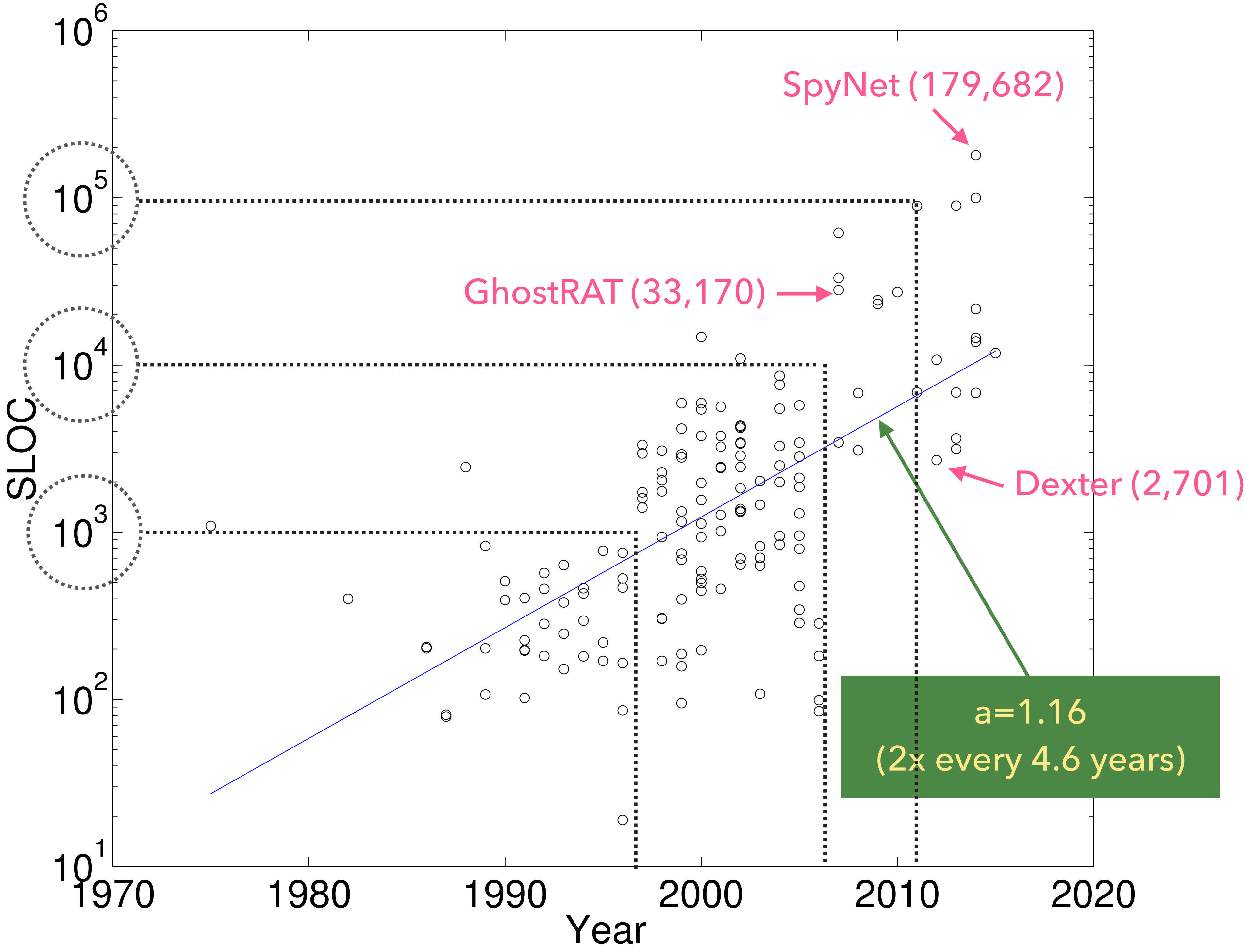


SpyNet (179,682)

GhostRAT (33,170)

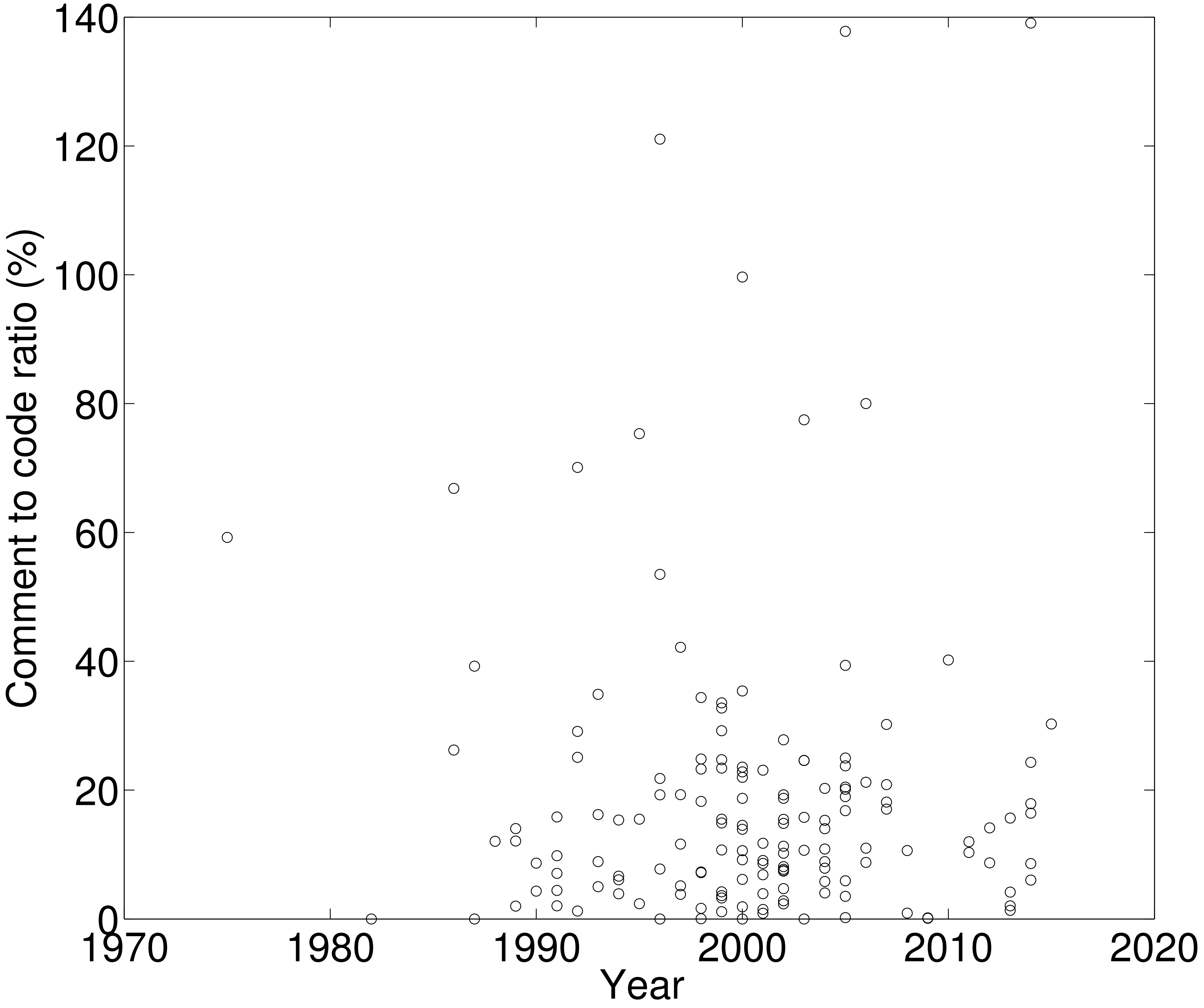
Dexter (2,701)

ANALYSIS

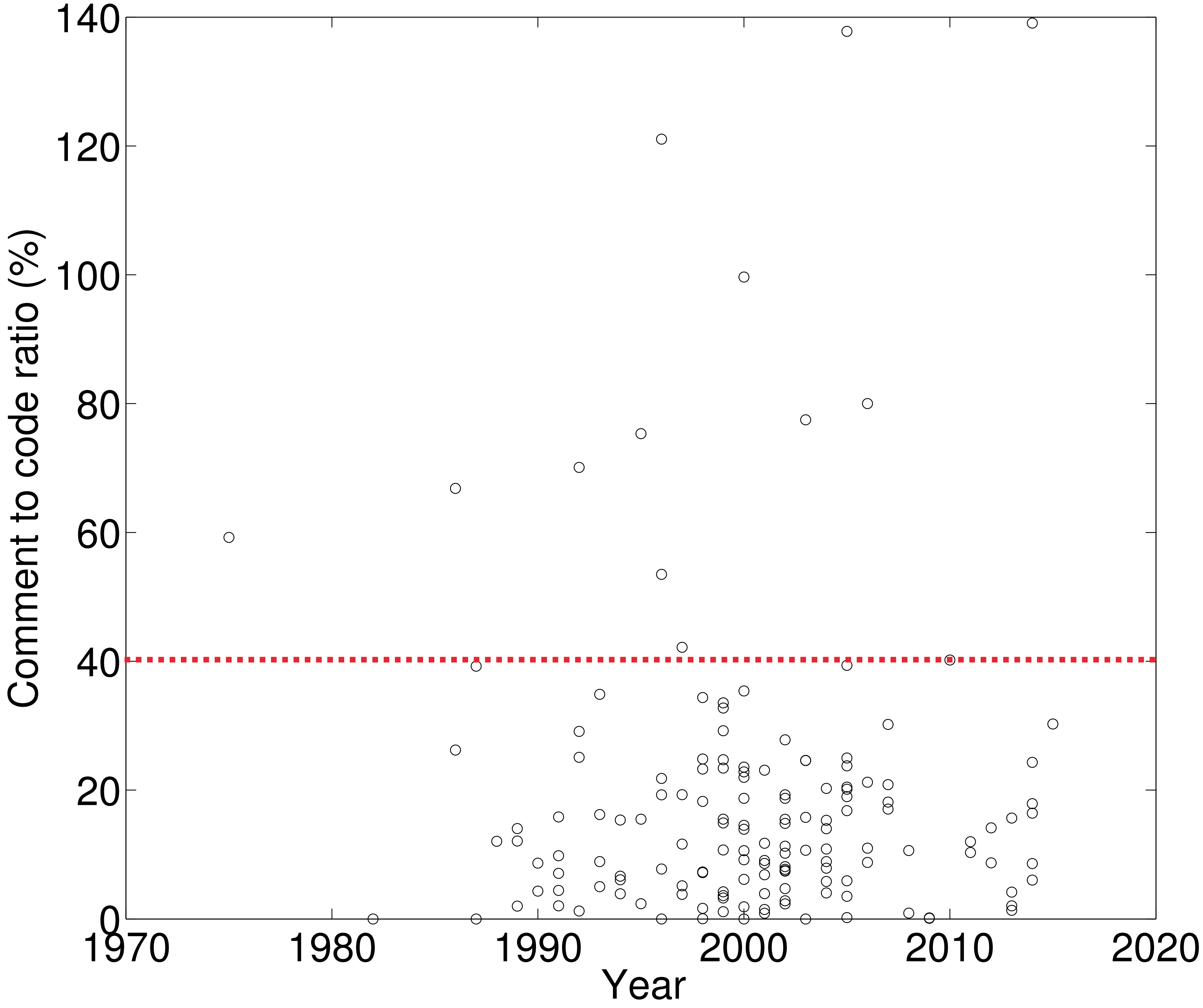


a=1.16
(2x every 4.6 years)

ANALYSIS

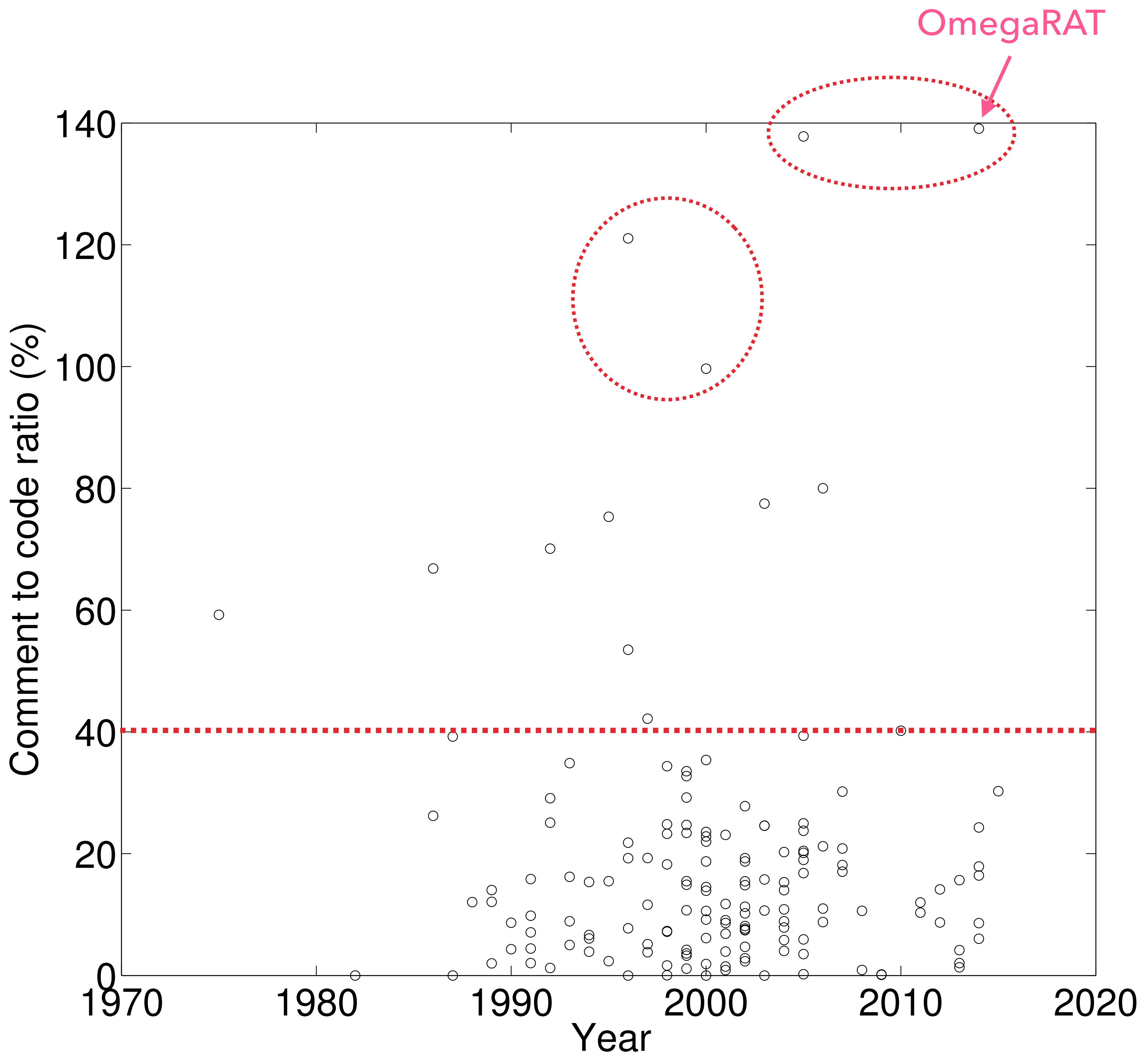

Comment to code ratio (%)
140
120
100
80
60
40
20
0
1970
1980
1990
2000
2010
2020
Year

ANALYSIS

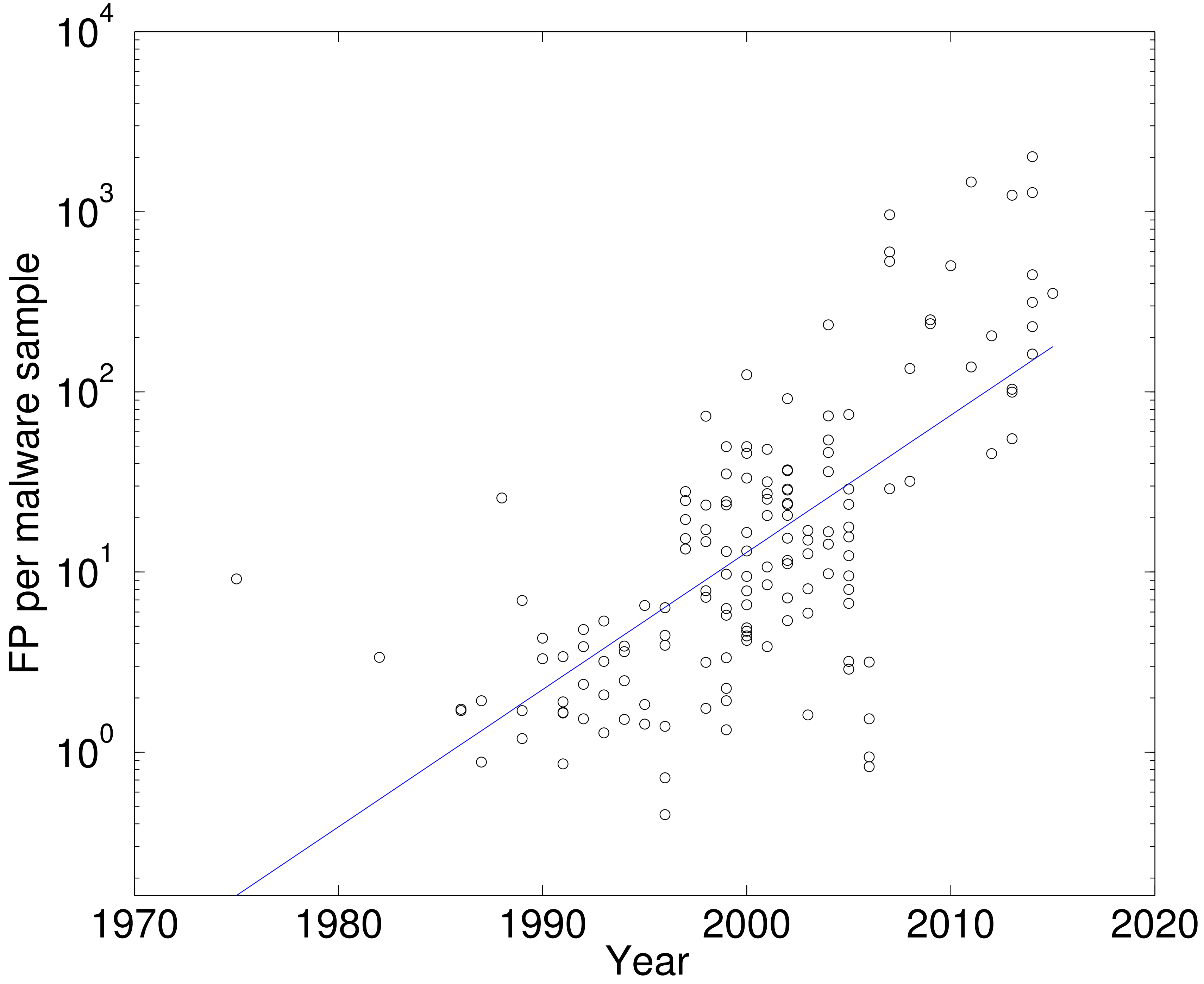


Comment to code ratio (%)
140
120
100
80
60
40
20
0
1970
1980
1990
2000
2010
2020
Year

ANALYSIS

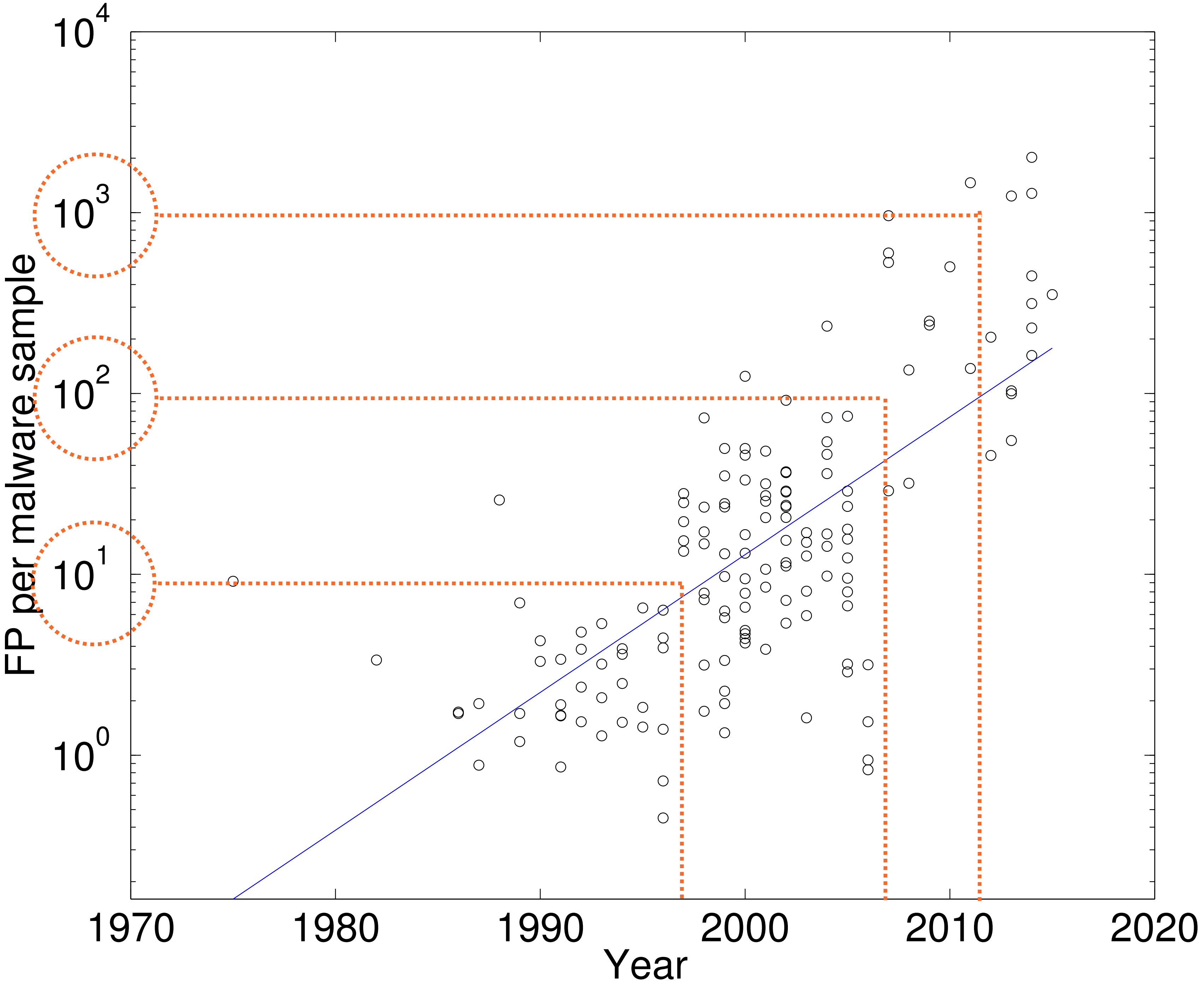

OmegaRAT
Comment to code ratio (%)
0
20
40
60
80
100
120
140
1970
1980
1990
2000
2010
2020
Year

ANALYSIS

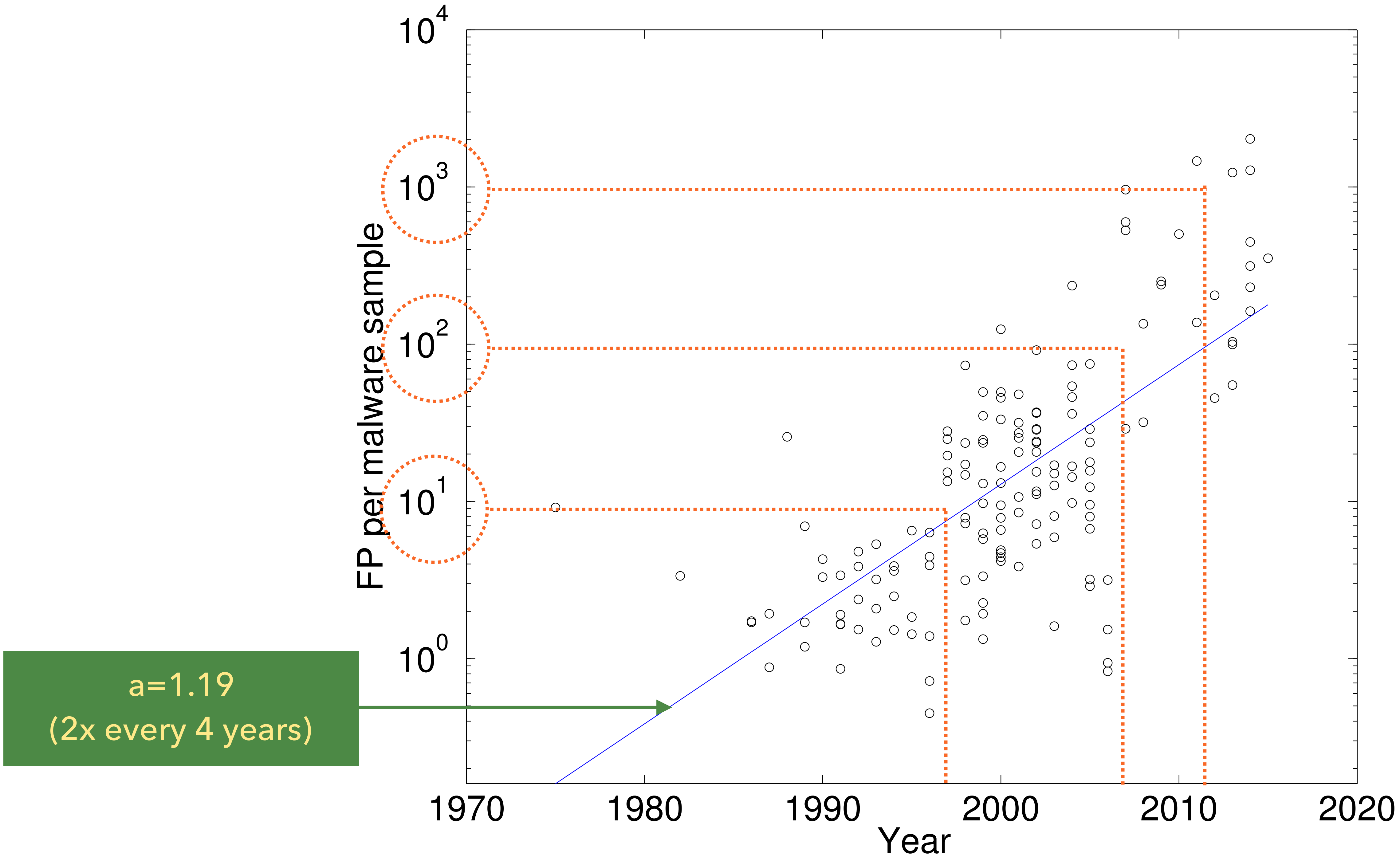


10^4
10^3
10^2
10^1
10^0
FP per malware sample
1970
1980
1990
2000
2010
2020
Year

ANALYSIS

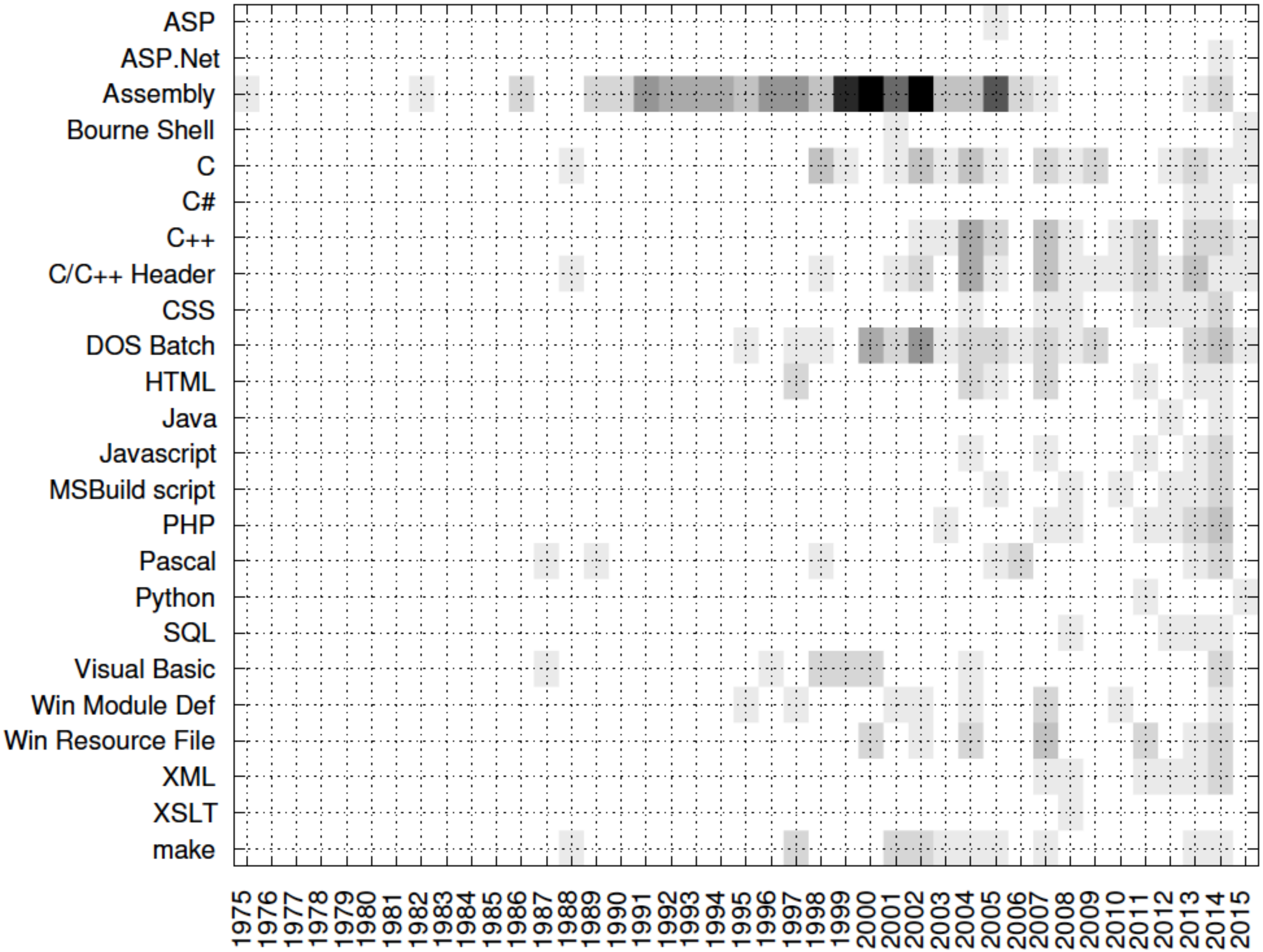

10^4
10^3
10^2
10^1
10^0
FP per malware sample
1970
1980
1990
2000
2010
2020
Year

ANALYSIS



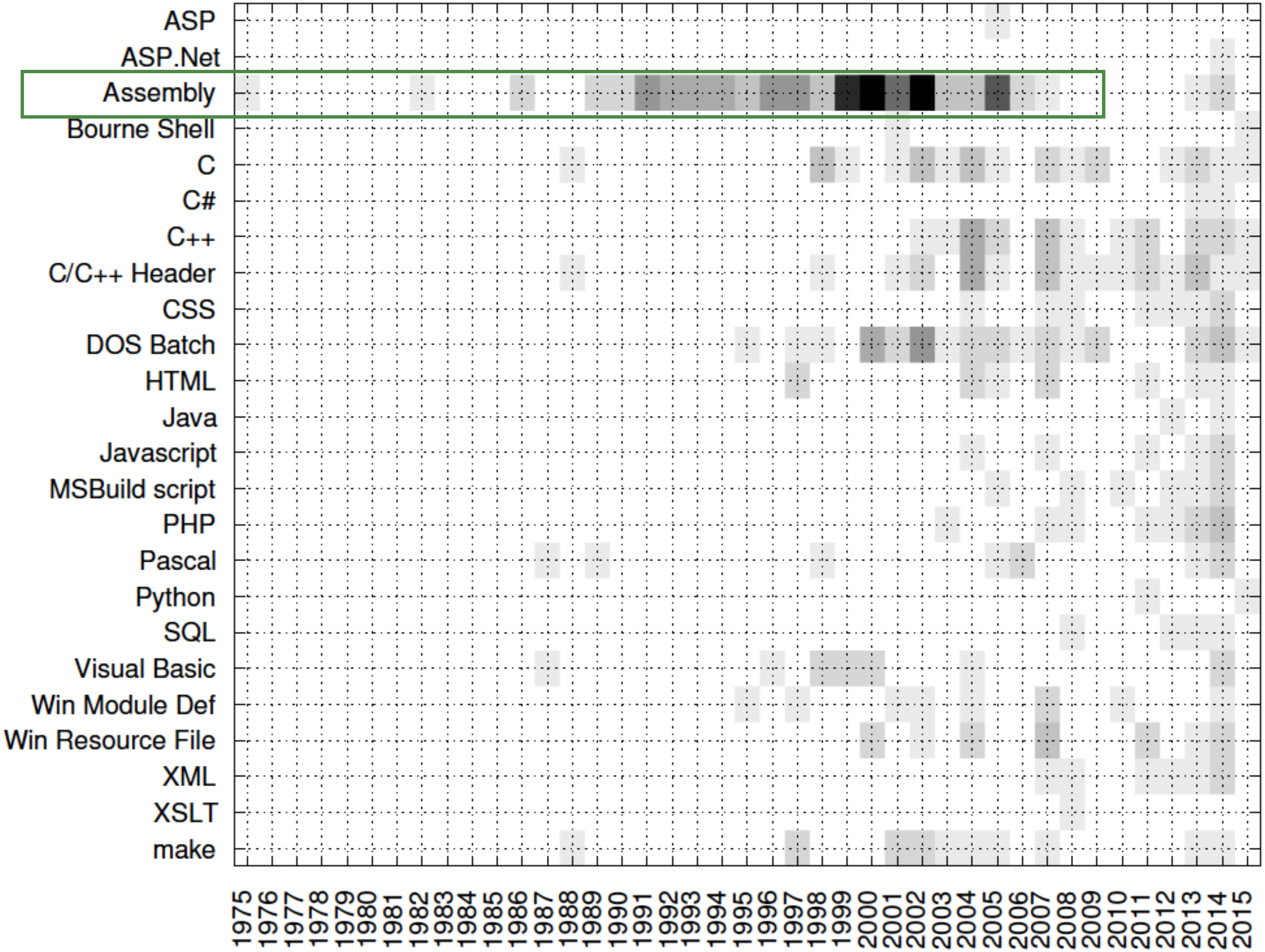
10^4
10^3
10^2
10^1
10^0
FP per malware sample
1970
1980
1990
2000
2010
2020
Year
a=1.19
(2x every 4 years)

ANALYSIS

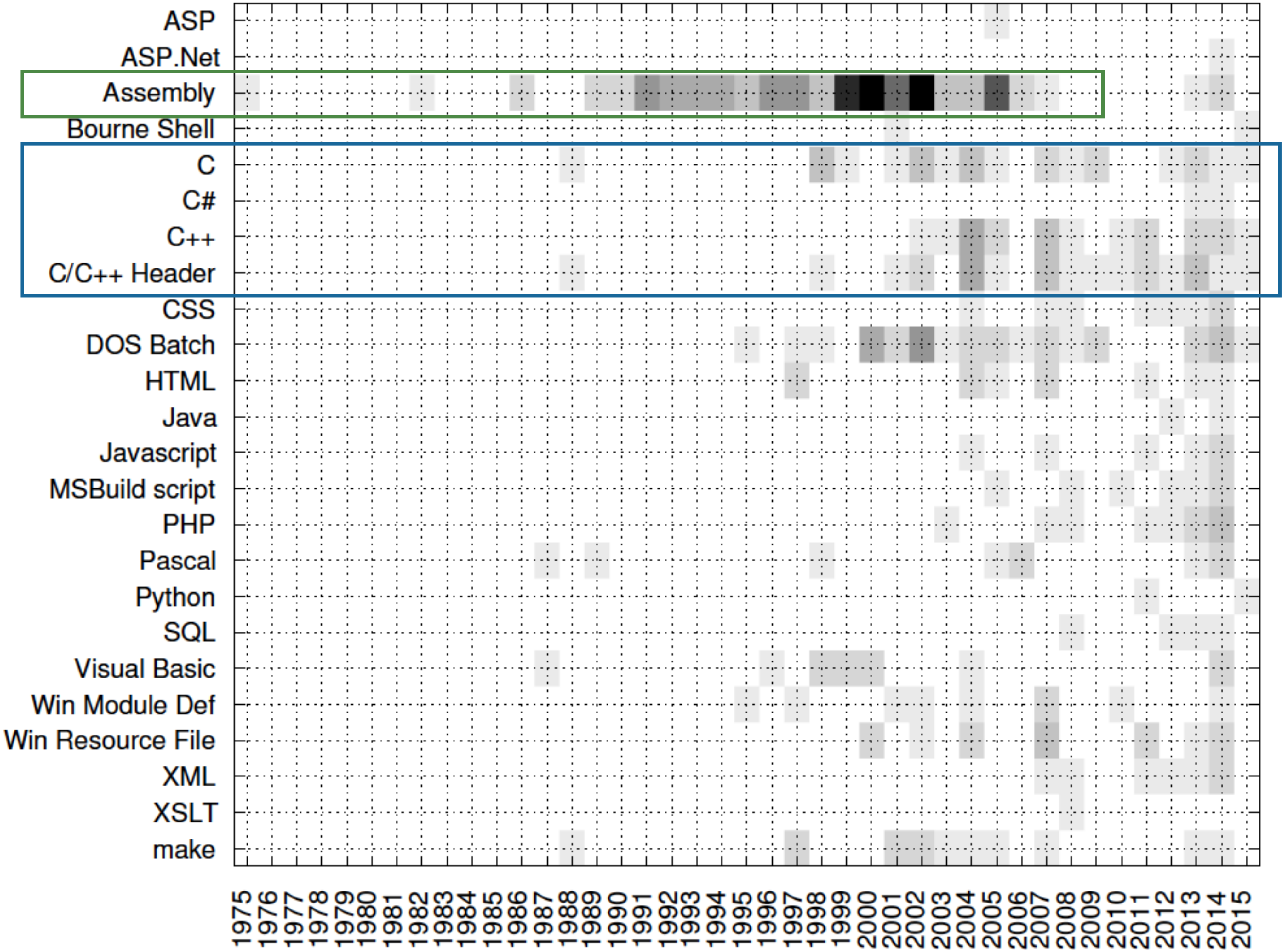


ASP
ASP.Net
Assembly
Bourne Shell
C
C#
C++
C/C++ Header
CSS
DOS Batch
HTML
Java
Javascript
MSBuild script
PHP
Pascal
Python
SQL
Visual Basic
Win Module Def
Win Resource File
XML
XSLT
make
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015

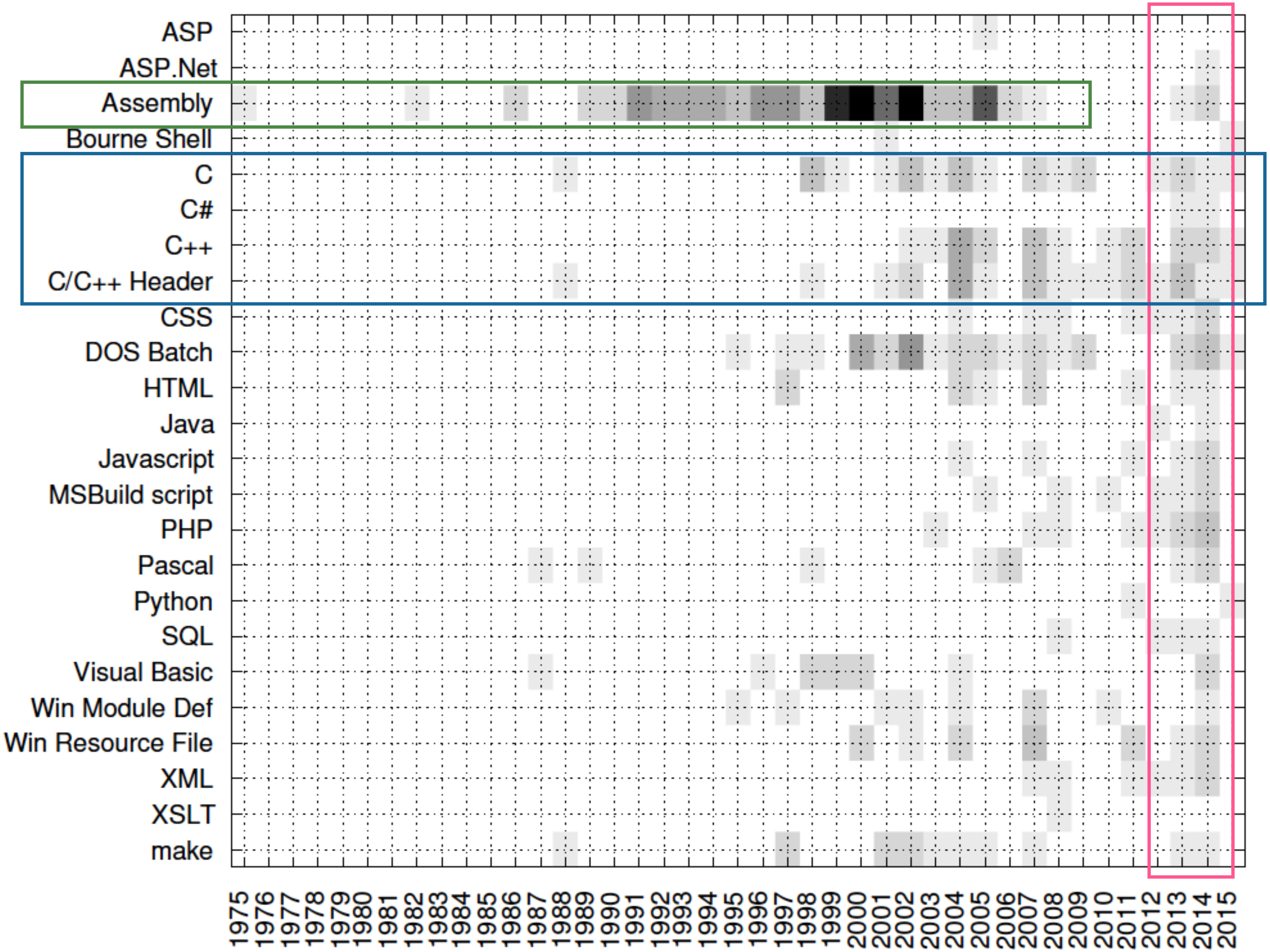
ANALYSIS


ASP
ASP.Net
Assembly
Bourne Shell
C
C#
C++
C/C++ Header
CSS
DOS Batch
HTML
Java
Javascript
MSBuild script
PHP
Pascal
Python
SQL
Visual Basic
Win Module Def
Win Resource File
XML
XSLT
make
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015

ANALYSIS


ASP
ASP.Net
Assembly
Bourne Shell
C
C#
C++
C/C++ Header
CSS
DOS Batch
HTML
Java
Javascript
MSBuild script
PHP
Pascal
Python
SQL
Visual Basic
Win Module Def
Win Resource File
XML
XSLT
make
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015

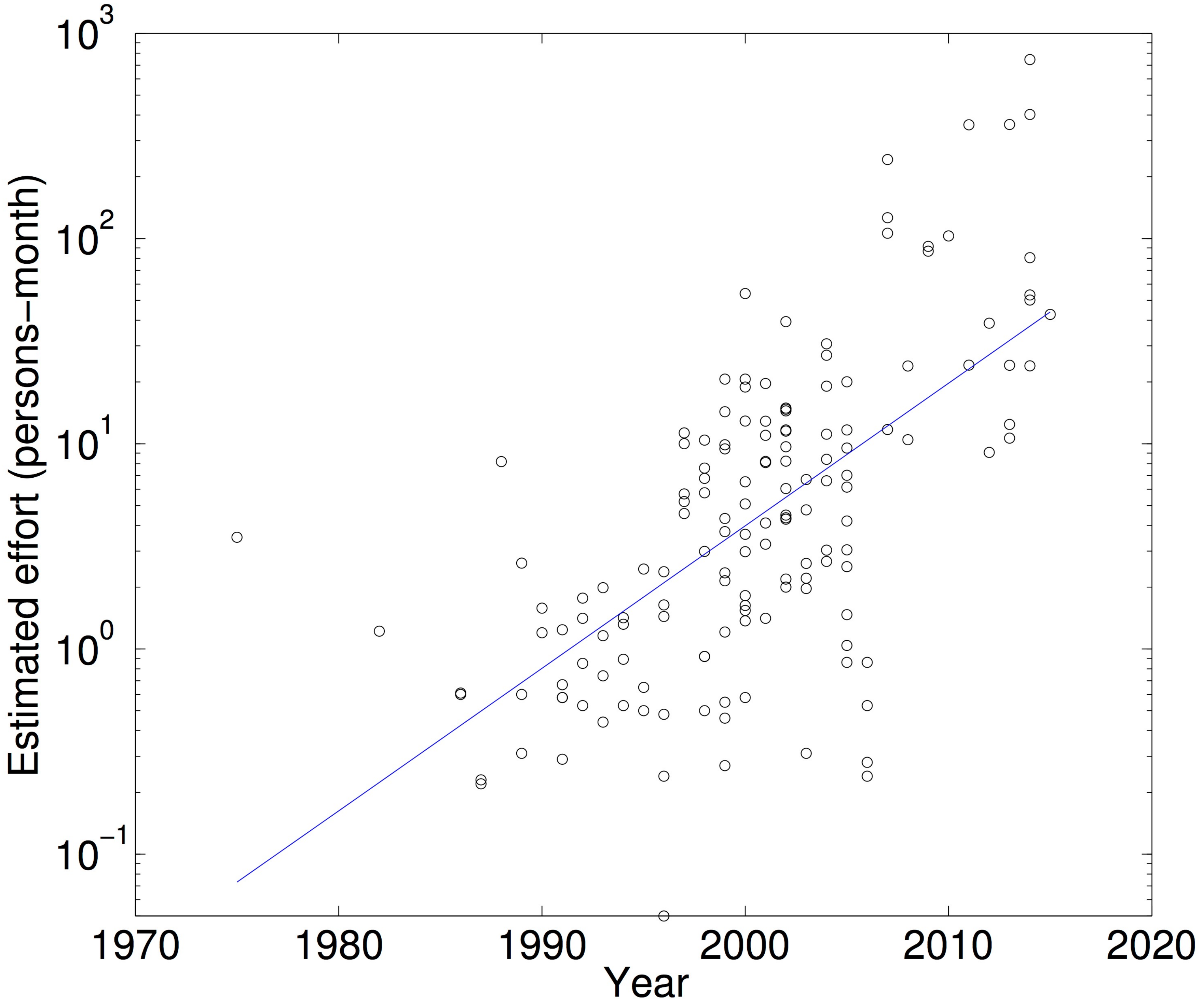
ANALYSIS


ASP
ASP.Net
Assembly
Bourne Shell
C
C#
C++
C/C++ Header
CSS
DOS Batch
HTML
Java
Javascript
MSBuild script
PHP
Pascal
Python
SQL
Visual Basic
Win Module Def
Win Resource File
XML
XSLT
make
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015

ANALYSIS

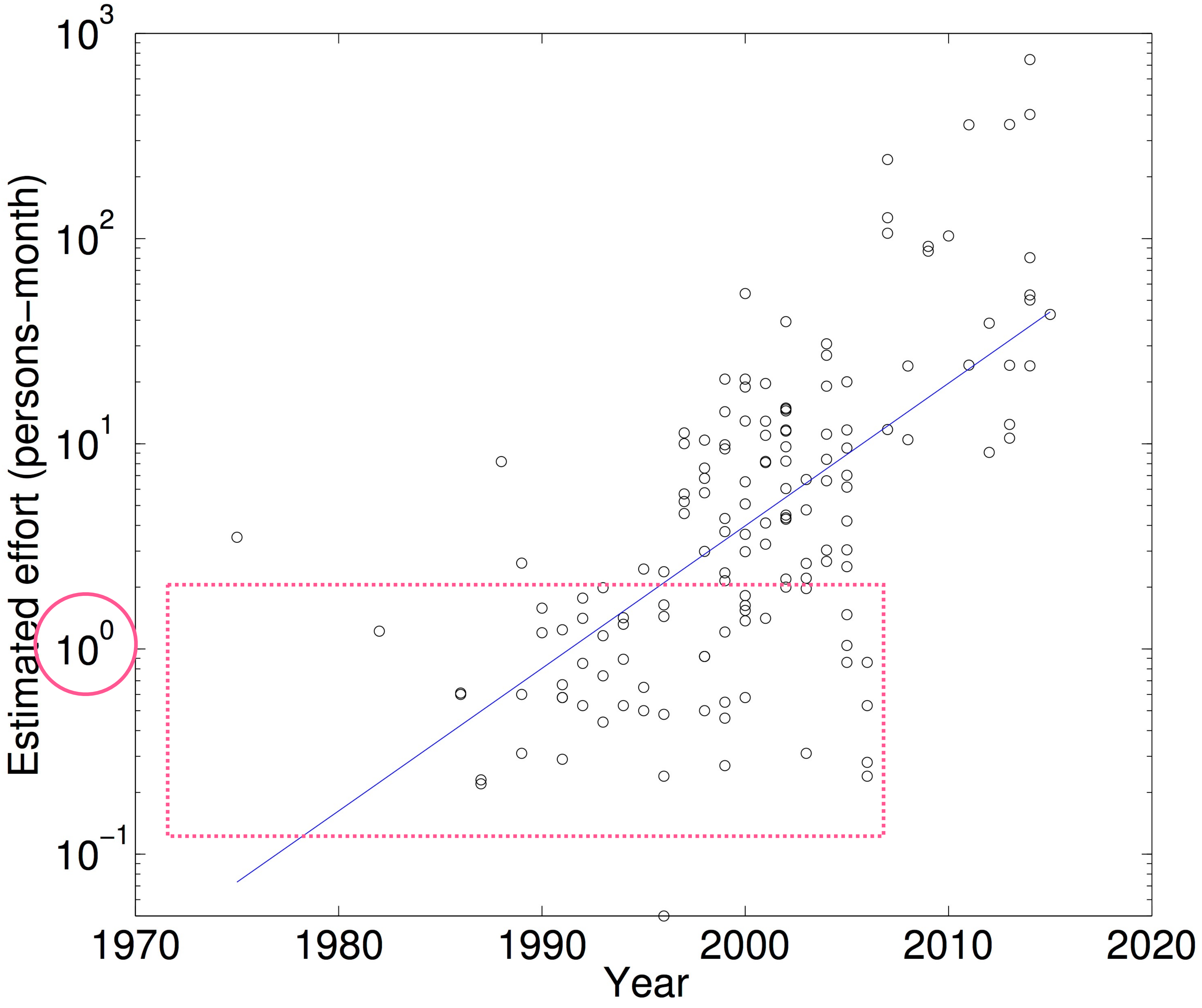
Source code analytics	Number of files
	SLOC count
	Density of comments
	FP count
	Programming languages
Cost estimates	Effort
	Development time
	Team size
Code quality	Complexity
	Maintainability
Comparison with regular software	

ANALYSIS



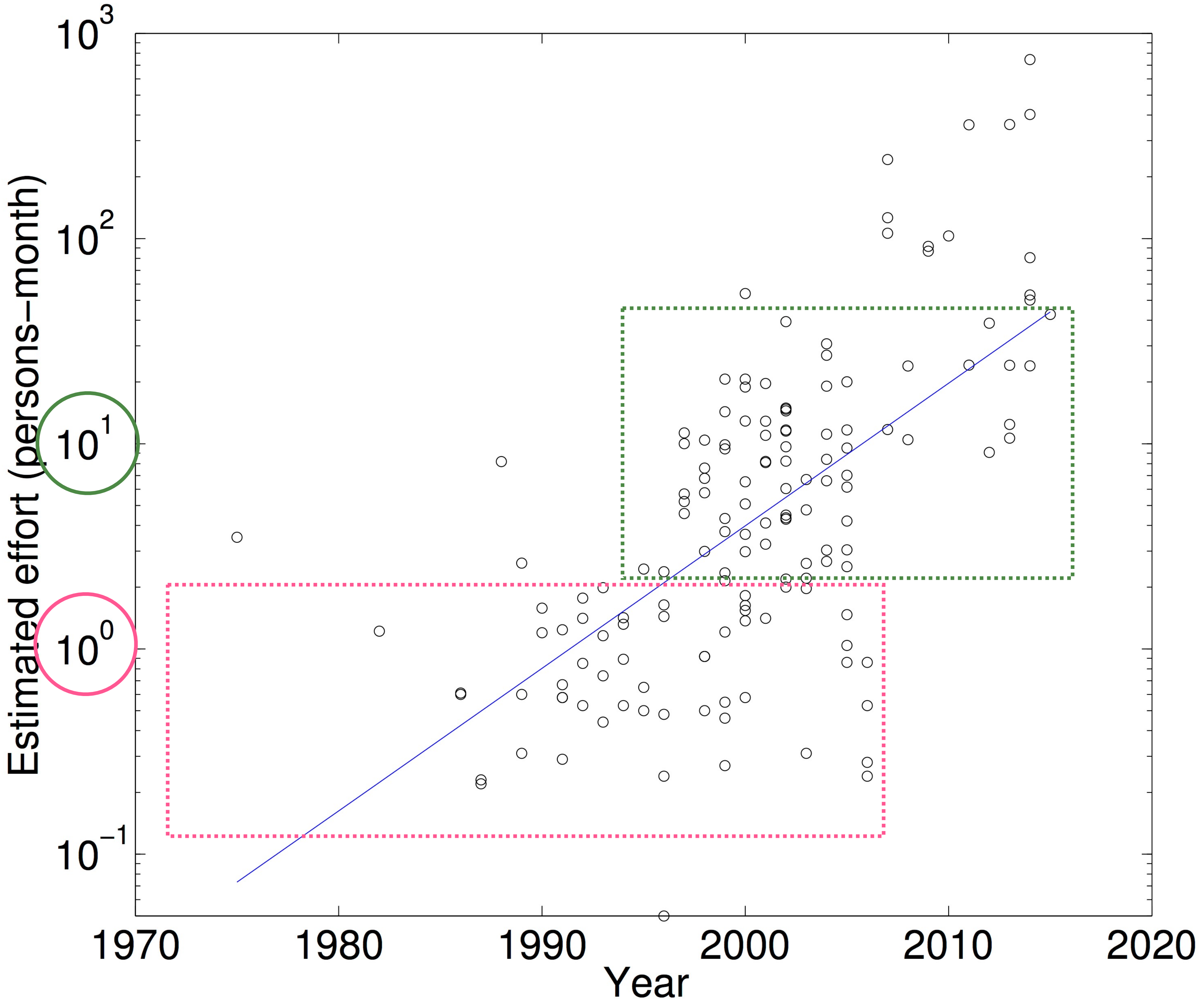
10^3
10^2
10^1
10^0
10^{-1}
Estimated effort (persons−month)
1970
1980
1990
2000
2010
2020
Year

ANALYSIS



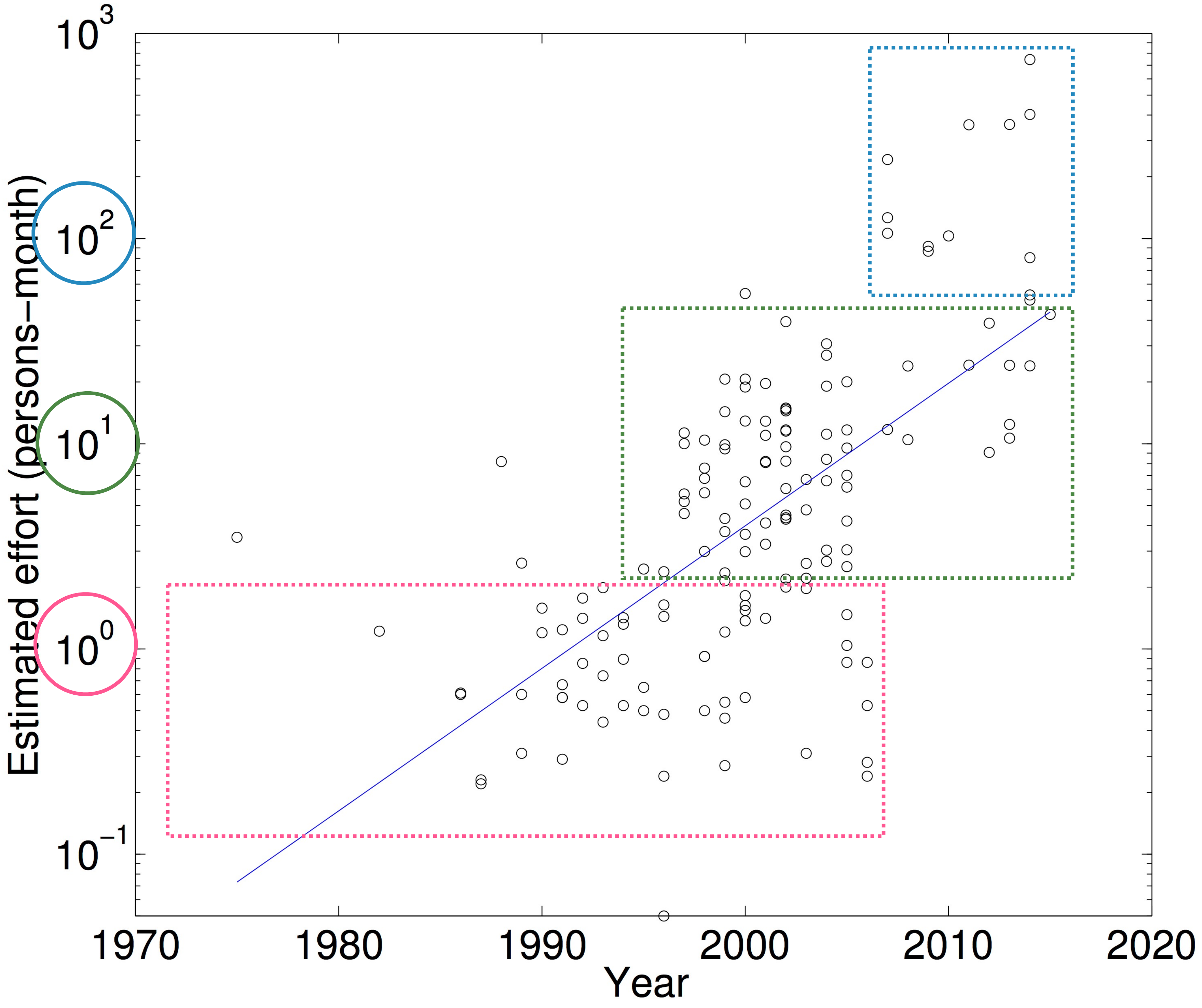
Estimated effort (persons–month)
10^3
10^2
10^1
10^0
10^{-1}
1970
1980
1990
2000
2010
2020
Year

ANALYSIS

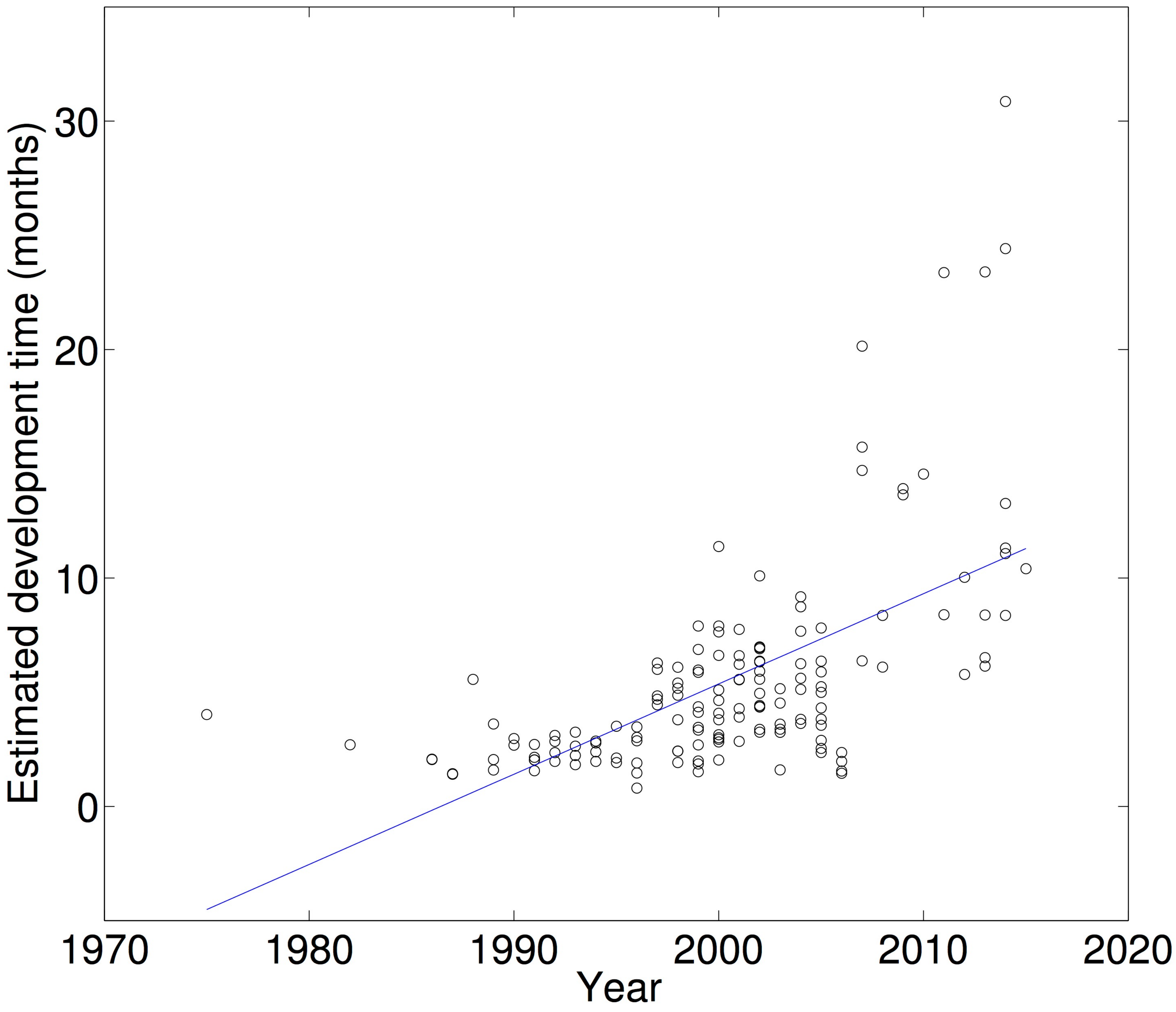


Estimated effort (persons–month)
10^3
10^2
10^1
10^0
10^{-1}
1970
1980
1990
2000
2010
2020
Year

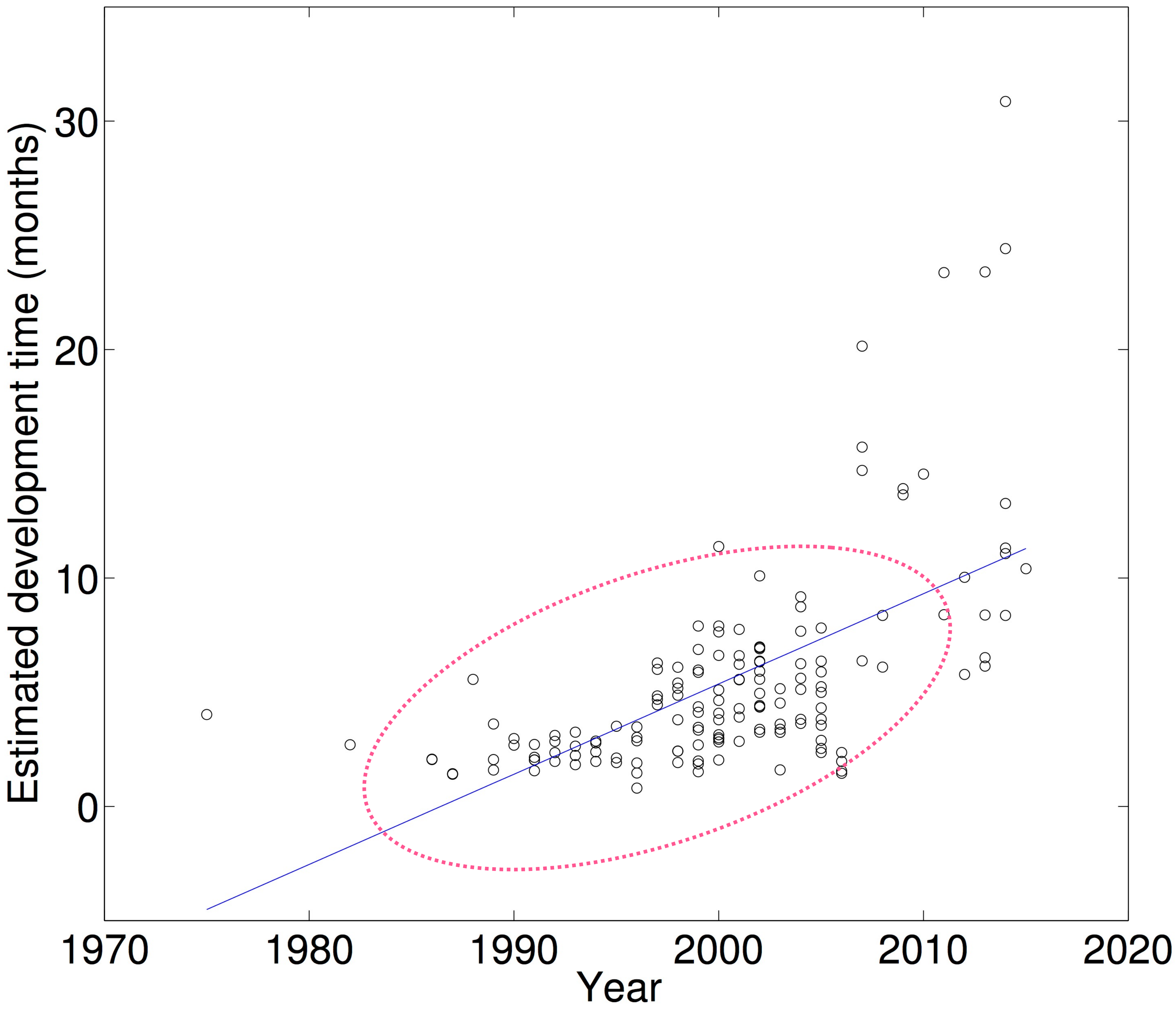
ANALYSIS


10^3
10^2
10^1
10^0
10^{-1}
Estimated effort (persons−month)
1970
1980
1990
2000
2010
2020
Year

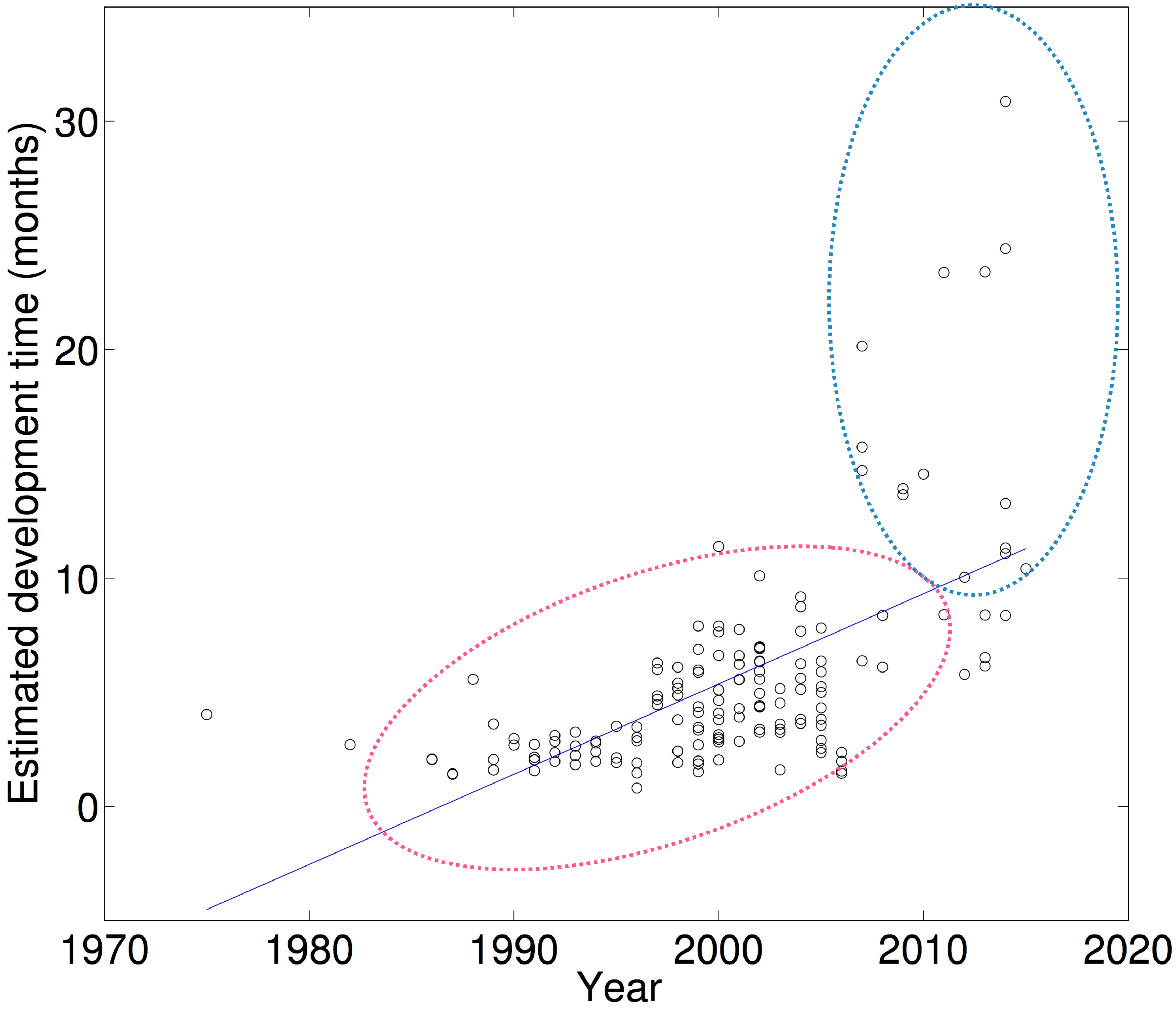
ANALYSIS


Estimated development time (months)
0
10
20
30
1970
1980
1990
2000
2010
2020
Year

ANALYSIS

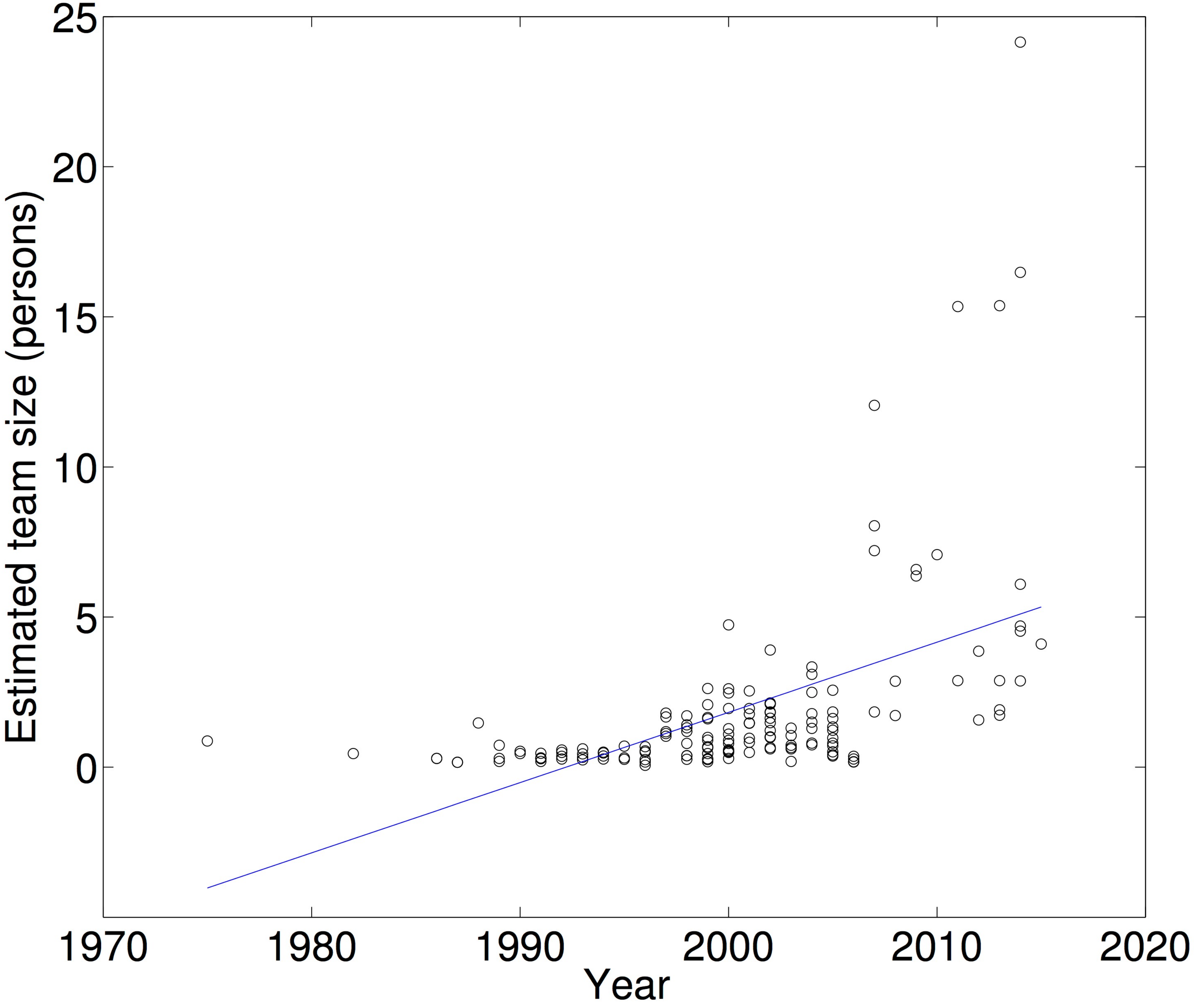

Estimated development time (months)
30
20
10
0
1970
1980
1990
2000
2010
2020
Year

ANALYSIS

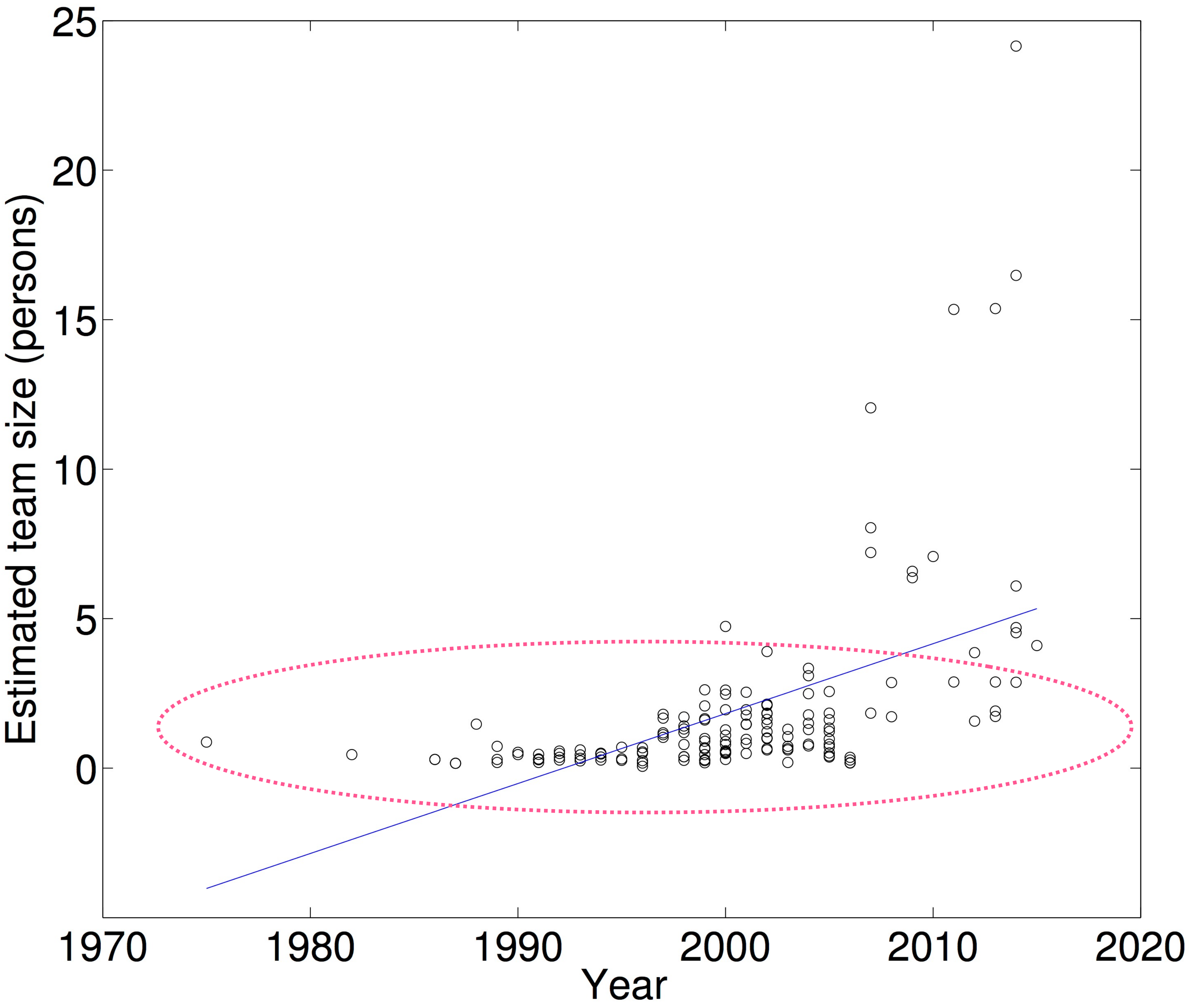


Estimated development time (months)
30
20
10
0
1970
1980
1990
2000
2010
2020
Year

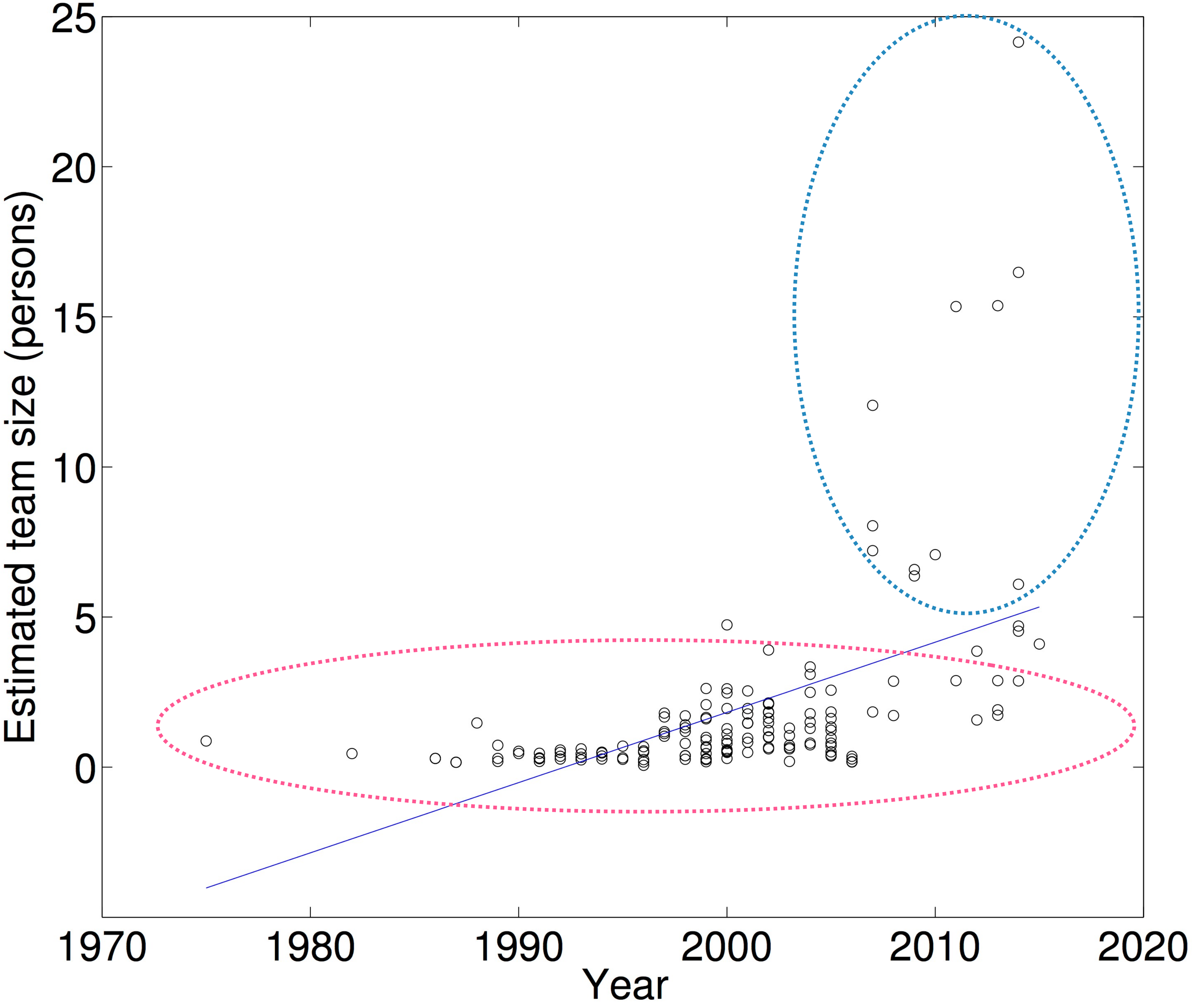
ANALYSIS


25
20
15
10
5
0
Estimated team size (persons)
1970
1980
1990
2000
2010
2020
Year

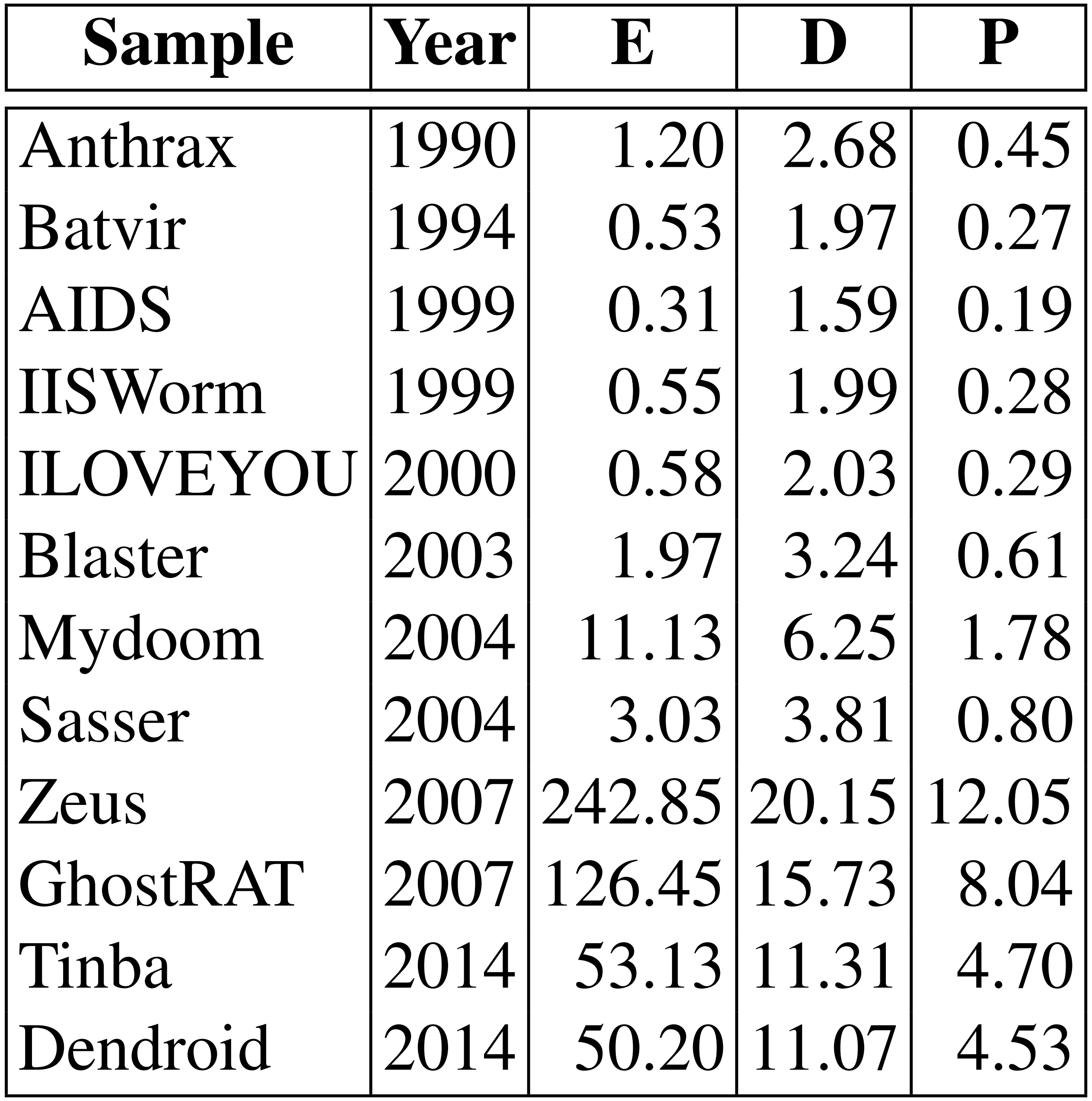
ANALYSIS


25
20
15
10
5
0
Estimated team size (persons)
1970
1980
1990
2000
2010
2020
Year

ANALYSIS


Estimated team size (persons)
25
20
15
10
5
0
1970
1980
1990
2000
2010
2020
Year

ANALYSIS

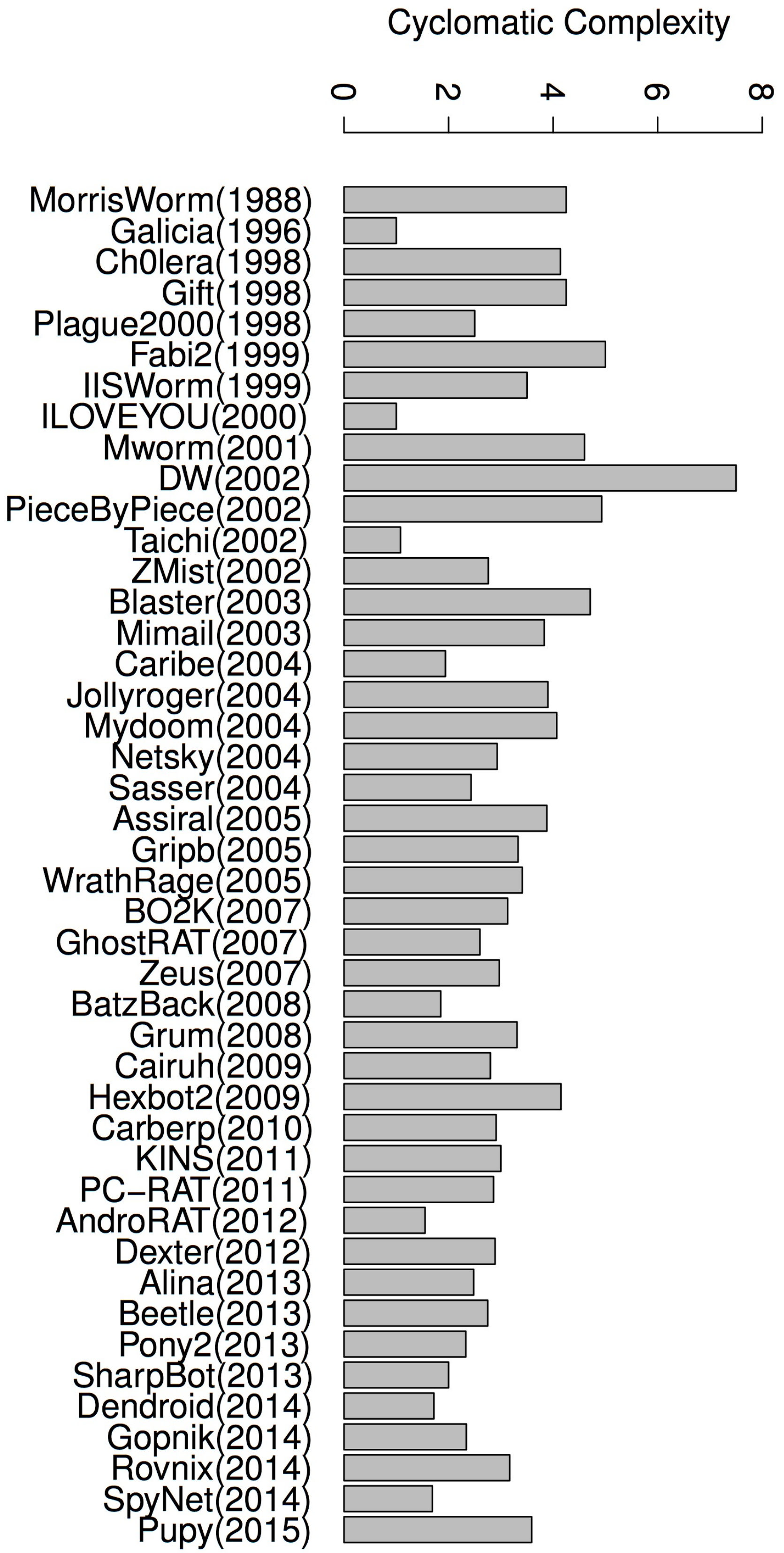


Sample	Year	E	D	P
Anthrax	1990	1.20	2.68	0.45
Batvir	1994	0.53	1.97	0.27
AIDS	1999	0.31	1.59	0.19
IISWorm	1999	0.55	1.99	0.28
ILOVEYOU	2000	0.58	2.03	0.29
Blaster	2003	1.97	3.24	0.61
Mydoom	2004	11.13	6.25	1.78
Sasser	2004	3.03	3.81	0.80
Zeus	2007	242.85	20.15	12.05
GhostRAT	2007	126.45	15.73	8.04
Tinba	2014	53.13	11.31	4.70
Dendroid	2014	50.20	11.07	4.53

ANALYSIS

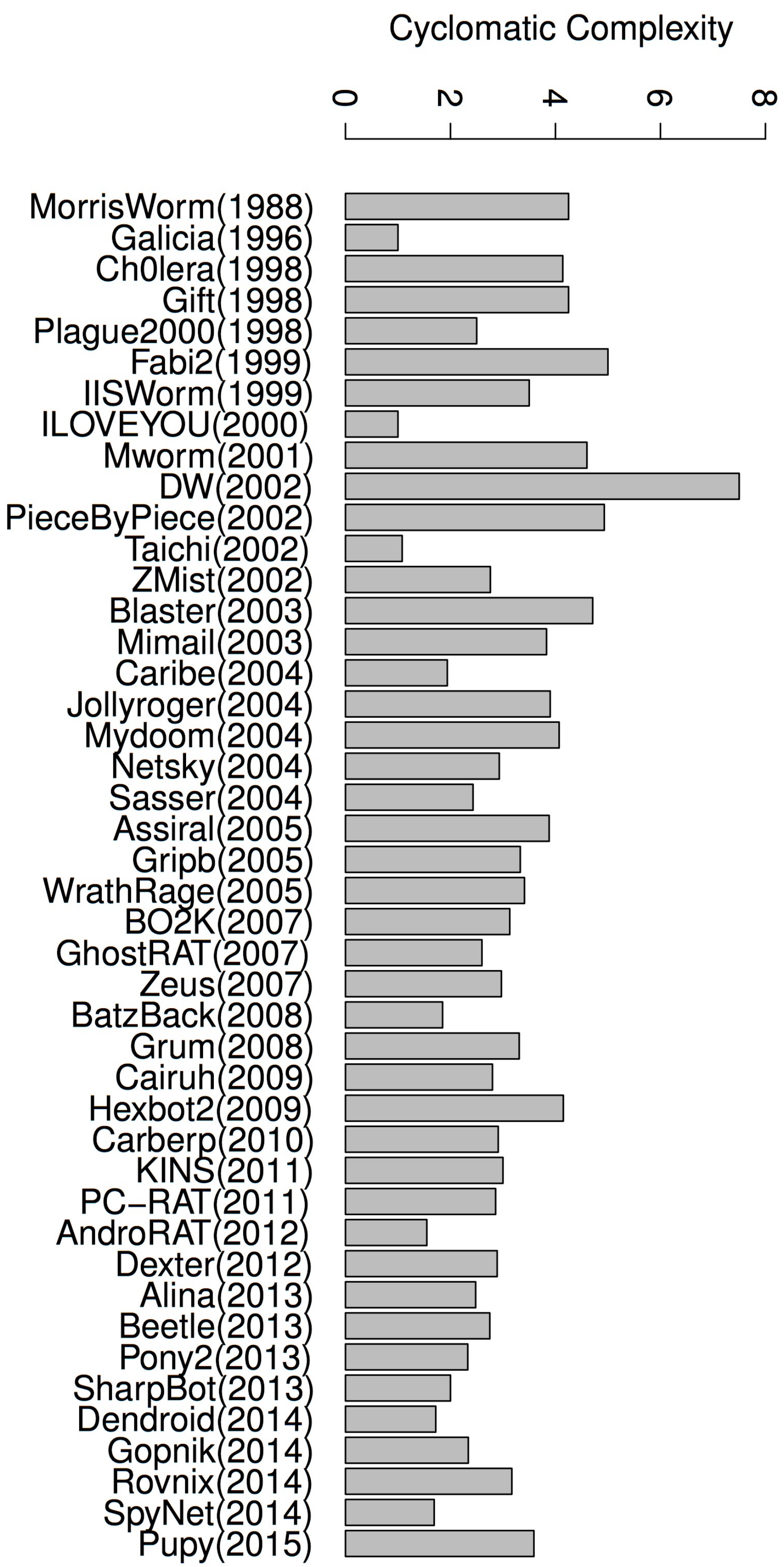
Source code analytics	Number of files
	SLOC count
	Density of comments
	FP count
	Programming languages
Cost estimates	Effort
	Development time
	Team size
Code quality	Complexity
	Maintainability
Comparison with regular software	

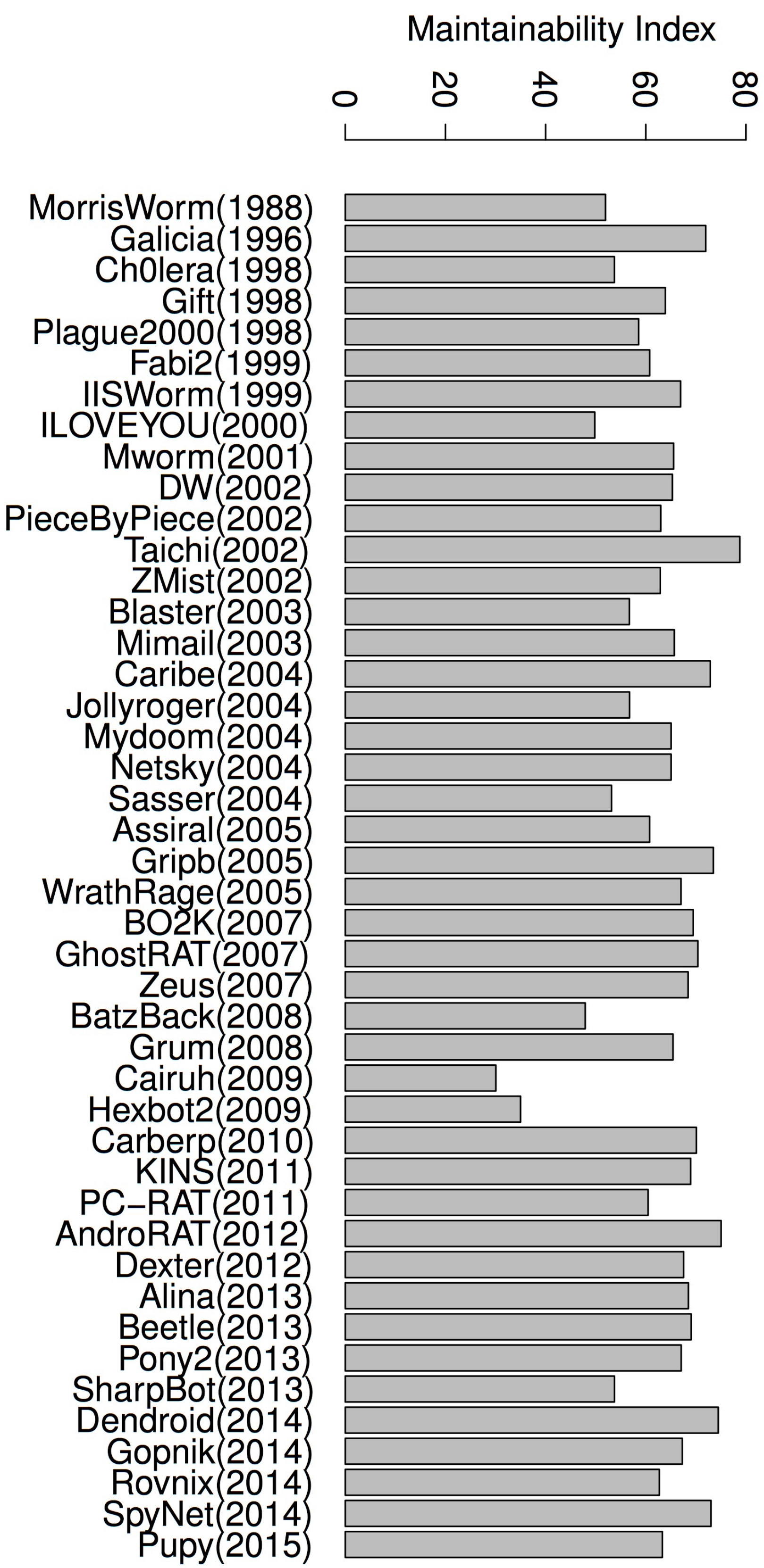
ANALYSIS



Cyclomatic Complexity
0
2
4
6
8
MorrisWorm(1988)
Galicia(1996)
Ch0lera(1998)
Gift(1998)
Plague2000(1998)
Fabi2(1999)
IISWorm(1999)
ILOVEYOU(2000)
Mworm(2001)
DW(2002)
PieceByPiece(2002)
Taichi(2002)
ZMist(2002)
Blaster(2003)
Mimail(2003)
Caribe(2004)
Jollyroger(2004)
Mydoom(2004)
Netsky(2004)
Sasser(2004)
Assiral(2005)
Gripb(2005)
WrathRage(2005)
BO2K(2007)
GhostRAT(2007)
Zeus(2007)
BatzBack(2008)
Grum(2008)
Cairuh(2009)
Hexbot2(2009)
Carberp(2010)
KINS(2011)
PC−RAT(2011)
AndroRAT(2012)
Dexter(2012)
Alina(2013)
Beetle(2013)
Pony2(2013)
SharpBot(2013)
Dendroid(2014)
Gopnik(2014)
Rovnix(2014)
SpyNet(2014)
Pupy(2015)

ANALYSIS

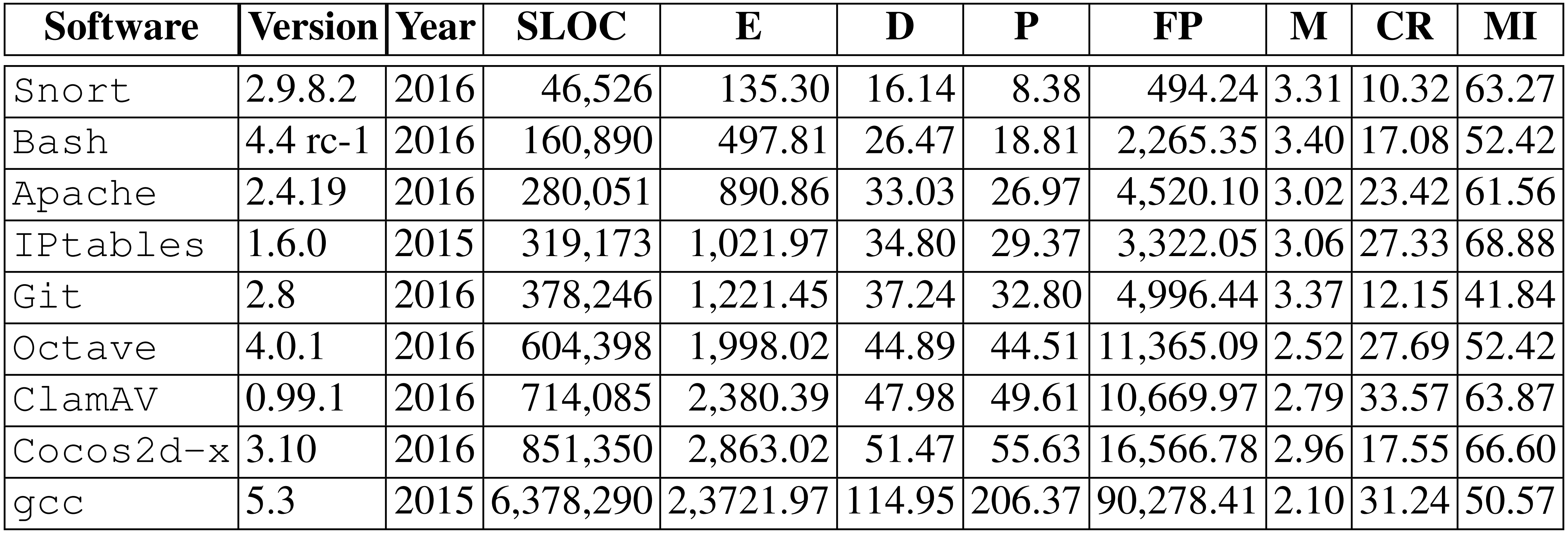

Cyclomatic Complexity


Maintainability Index

ANALYSIS

Source code analytics	Number of files
	SLOC count
	Density of comments
	FP count
	Programming languages
Cost estimates	Effort
	Development time
	Team size
Code quality	Complexity
	Maintainability
Comparison with regular software	

ANALYSIS



Software	Version	Year	SLOC	E	D	P	FP	M	CR	MI
Snort	2.9.8.2	2016	46,526	135.30	16.14	8.38	494.24	3.31	10.32	63.27
Bash	4.4 rc-1	2016	160,890	497.81	26.47	18.81	2,265.35	3.40	17.08	52.42
Apache	2.4.19	2016	280,051	890.86	33.03	26.97	4,520.10	3.02	23.42	61.56
IPtables	1.6.0	2015	319,173	1,021.97	34.80	29.37	3,322.05	3.06	27.33	68.88
Git	2.8	2016	378,246	1,221.45	37.24	32.80	4,996.44	3.37	12.15	41.84
Octave	4.0.1	2016	604,398	1,998.02	44.89	44.51	11,365.09	2.52	27.69	52.42
ClamAV	0.99.1	2016	714,085	2,380.39	47.98	49.61	10,669.97	2.79	33.57	63.87
Cocos2d-x	3.10	2016	851,350	2,863.02	51.47	55.63	16,566.78	2.96	17.55	66.60
gcc	5.3	2015	6,378,290	2,3721.97	114.95	206.37	90,278.41	2.10	31.24	50.57

CONCLUSIONS

- Numerical evidence of how malware has become increasingly complex over the last 30 years:
  - Increments of approx. a factor of 10 per decade in number of files, SLOC and FP counts.
  - Development costs: from small projects of 1 person working 1-2 months to larger programming teams working 6-8 months (and more).
  - Largest malware samples similar in complexity to small benign products.
  - No significant difference in terms of code quality.

CONCLUSIONS

- Potential limitations:
  - Software metrics, really?
  - Dataset quality
- Ongoing work:
  - Extended dataset
  - Code sharing
  - Authorship attribution

THANKS!