Seminar Software Composition: Project P6

How are Software Visualizations Evaluated?

Presented by: Lukas Imstepf

lukas.imstepf@unifr.ch

Department of Informatics
University of Fribourg

April 11, 2017

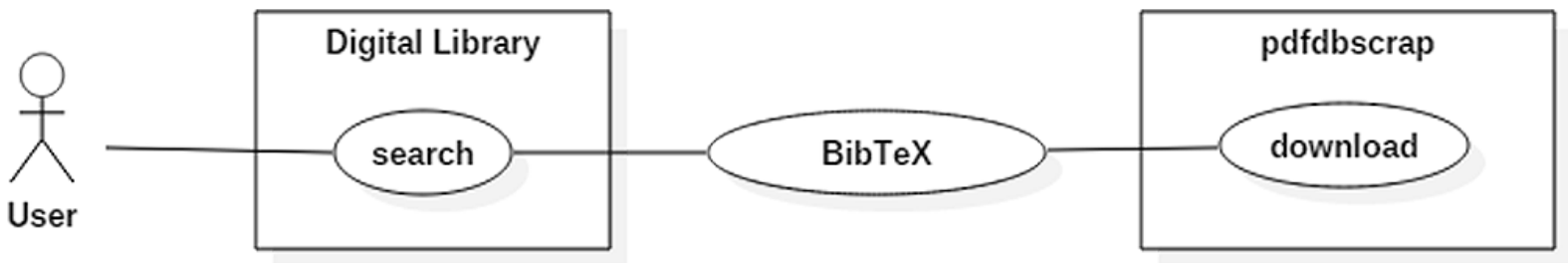
MASTER IN COMPUTER SCIENCE

How to download a bunch of PDF files . . .

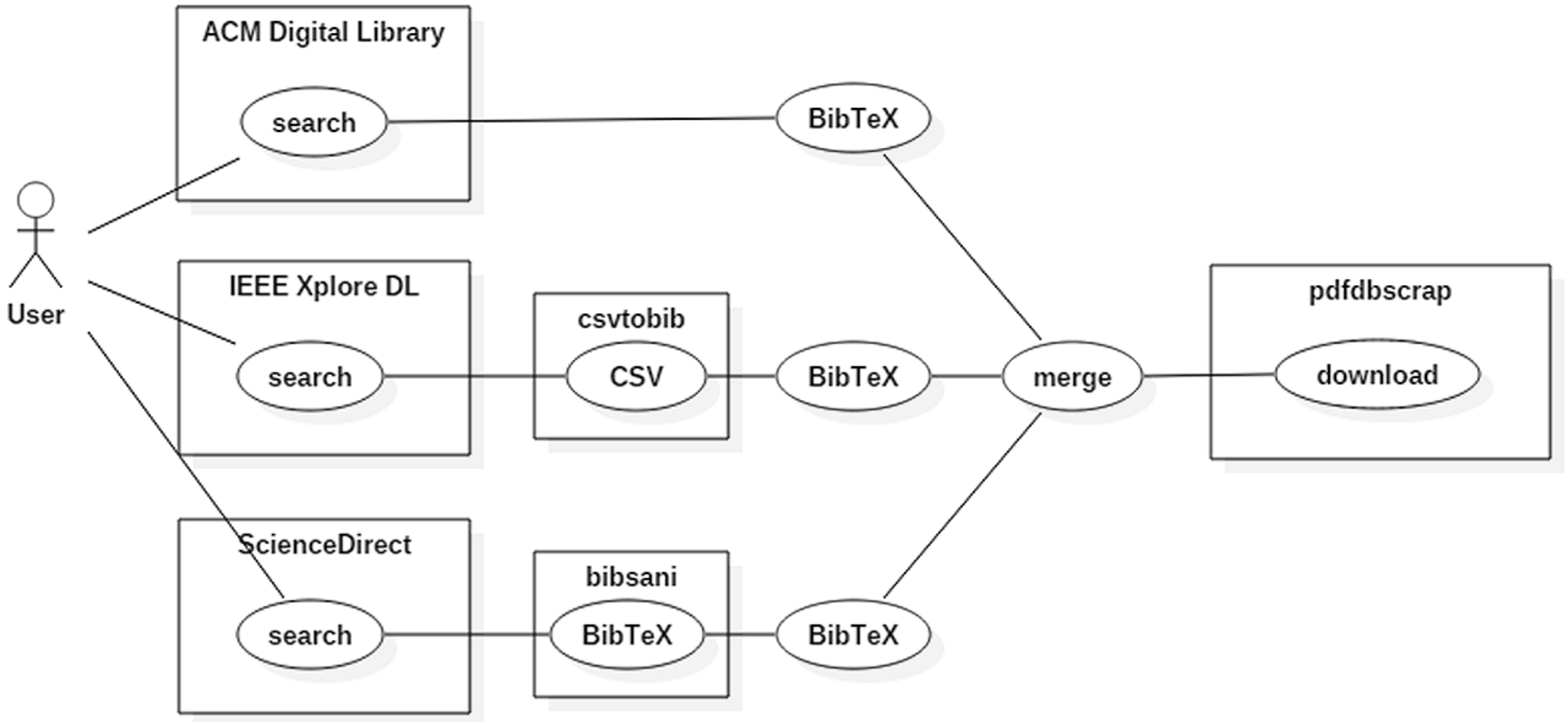
How to download a bunch of PDF files . . .
. . . and what to do with them.

PDF Database Scrap(er): `pdfdbscrap`

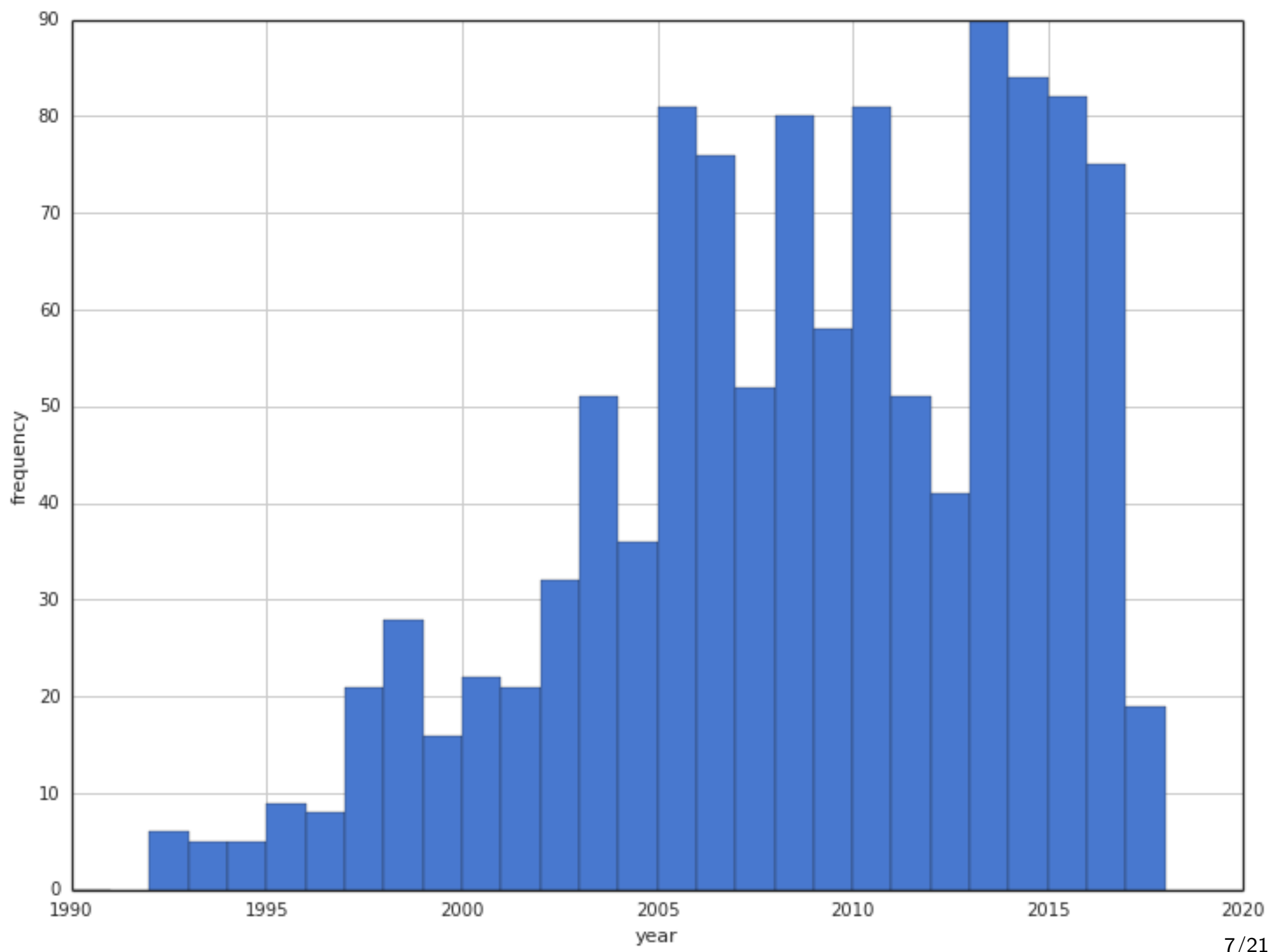
pdfdbscrap: General Use Case

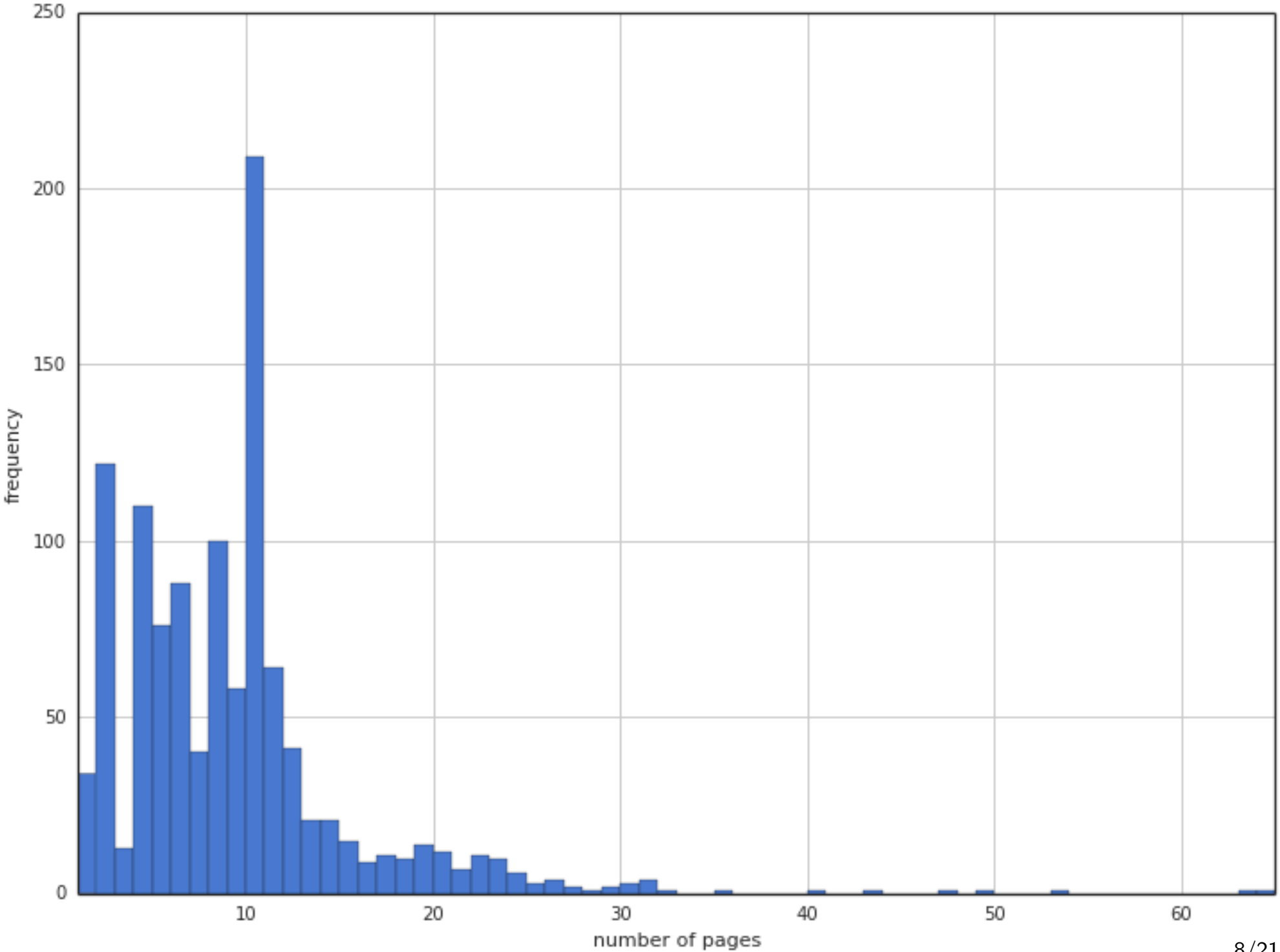


pdfdbscrap: Multi-Source Use Case



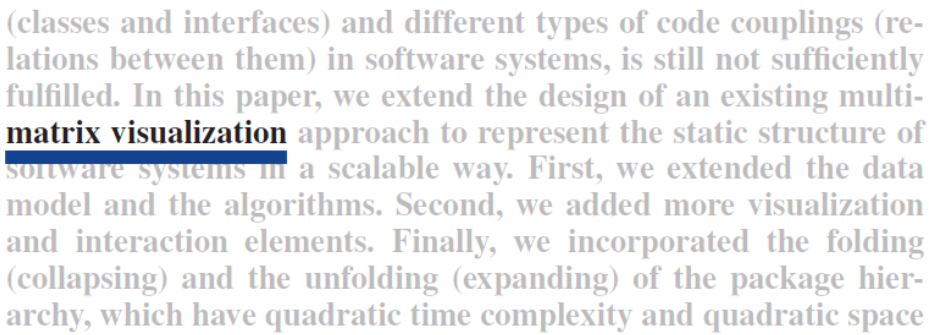
. . . and what to do with them.

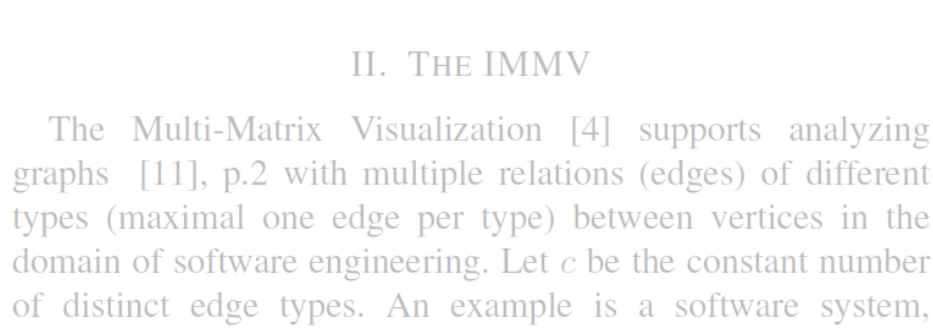

90
80
70
60
50
40
30
20
10
0
frequency
1990
1995
2000
2005
2010
2015
2020
year


250
200
150
100
50
0
frequency
10
20
30
40
50
60
number of pages

PDF Hi(stogram) Score: `pdfhiscore`

pdfhiscore

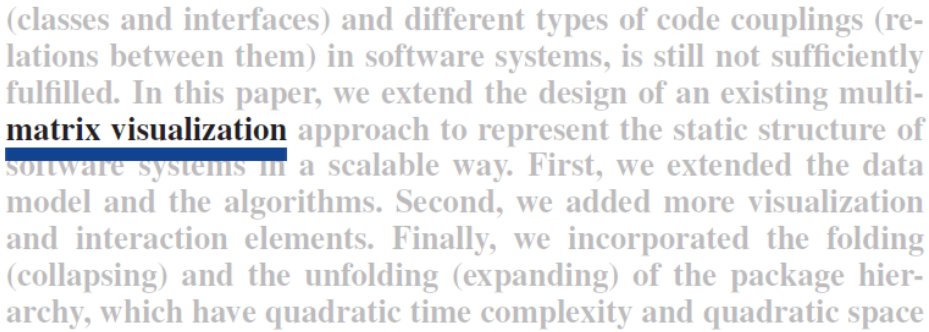

(classes and interfaces) and different types of code couplings (relations between them) in software systems, is still not sufficiently fulfilled. In this paper, we extend the design of an existing multi-matrix visualization approach to represent the static structure of software systems in a scalable way. First, we extended the data model and the algorithms. Second, we added more visualization and interaction elements. Finally, we incorporated the folding (collapsing) and the unfolding (expanding) of the package hierarchy, which have quadratic time complexity and quadratic space

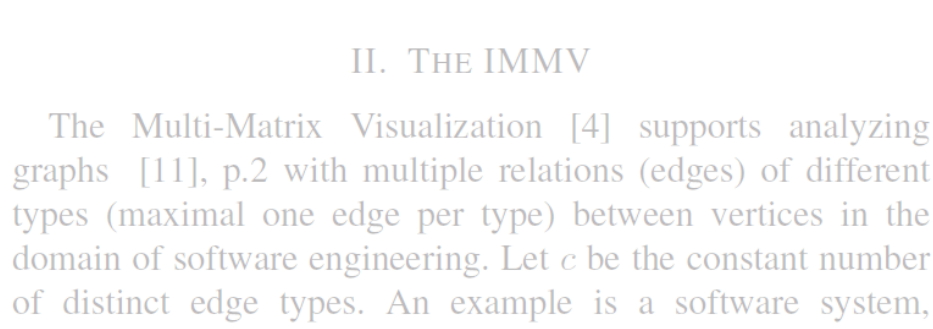

II. The IMMV

The Multi-Matrix Visualization [4] supports analyzing graphs [11], p.2 with multiple relations (edges) of different types (maximal one edge per type) between vertices in the domain of software engineering. Let c be the constant number of distinct edge types. An example is a software system,

- Extract two word histograms from the text extracted from the PDF file:
  1. a single word histogram, and
  2. a compound word histogram (two consecutive words)

pdfhiscore

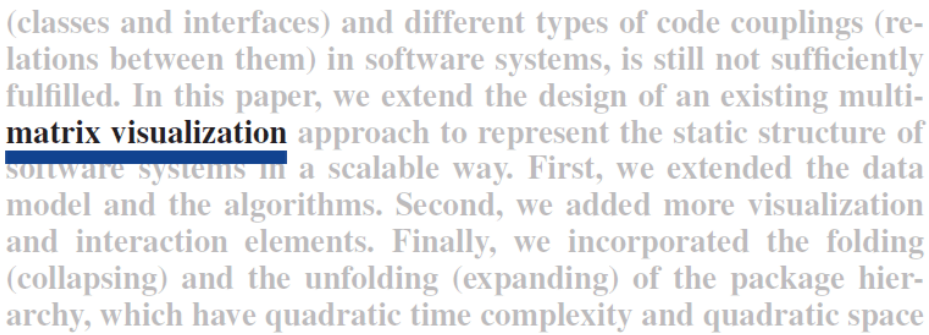

(classes and interfaces) and different types of code couplings (relations between them) in software systems, is still not sufficiently fulfilled. In this paper, we extend the design of an existing multi-matrix visualization approach to represent the static structure of software systems in a scalable way. First, we extended the data model and the algorithms. Second, we added more visualization and interaction elements. Finally, we incorporated the folding (collapsing) and the unfolding (expanding) of the package hierarchy, which have quadratic time complexity and quadratic space

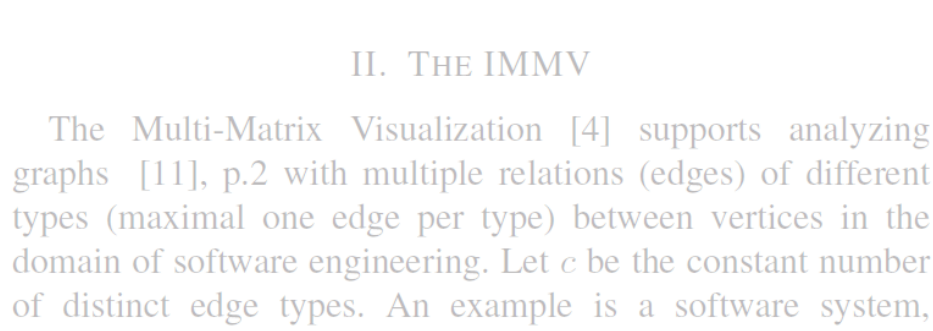

II. The IMMV

The Multi-Matrix Visualization [4] supports analyzing graphs [11], p.2 with multiple relations (edges) of different types (maximal one edge per type) between vertices in the domain of software engineering. Let c be the constant number of distinct edge types. An example is a software system,

- Extract two word histograms from the text extracted from the PDF file:
  1. a single word histogram, and
  2. a compound word histogram (two consecutive words)
- Evaluate Histogram Query Language (HQL) expressions to query the histograms

pdfhiscore


(classes and interfaces) and different types of code couplings (relations between them) in software systems, is still not sufficiently fulfilled. In this paper, we extend the design of an existing multi-matrix visualization approach to represent the static structure of software systems in a scalable way. First, we extended the data model and the algorithms. Second, we added more visualization and interaction elements. Finally, we incorporated the folding (collapsing) and the unfolding (expanding) of the package hierarchy, which have quadratic time complexity and quadratic space


II. The IMMV

The Multi-Matrix Visualization [4] supports analyzing graphs [11], p.2 with multiple relations (edges) of different types (maximal one edge per type) between vertices in the domain of software engineering. Let c be the constant number of distinct edge types. An example is a software system,

- Extract two word histograms from the text extracted from the PDF file:
  1. a single word histogram, and
  2. a compound word histogram (two consecutive words)
- Evaluate Histogram Query Language (HQL) expressions to query the histograms
- Output human-readable and easily parsable results (YAML)

pdfhiscore: Histogram Query Language (HQL)

- A simple boolean language (&&, ||, !) in prefix notation

```
grammar HistogramQueryLanguage;

prog: AND expr+
    | OR expr+
    | GT value+ INT
    | expr+
    ;

expr: '(' AND expr+ ')'
    | '(' OR expr+ ')'
    | '(' GT value+ INT ')'
    | NOT expr
    | value
    ;

value: WORD | SQWORD | DQWORD;

AND:    '&&';
OR:     '||';
NOT:    '!';
GT:     '>';

SQWORD: '\'' .*? '\'';
DQWORD: '"' .*? '"';
WORD:   ALPHA (ALPHA|DIGIT)*;
INT:    DIGIT+;

fragment ALPHA: [a-zA-Z_\-];
fragment DIGIT: [0-9];

WS: [ \t\r\n]+ -> skip;
```

pdfhiscore: Histogram Query Language (HQL)

- A simple boolean language (&&, ||, !) in prefix notation
- Words evaluate to true if they're in the histogram

```
grammar HistogramQueryLanguage;

prog: AND expr+
    | OR expr+
    | GT value+ INT
    | expr+
    ;

expr: '(' AND expr+ ')'
    | '(' OR expr+ ')'
    | '(' GT value+ INT ')'
    | NOT expr
    | value
    ;

value: WORD | SQWORD | DQWORD;

AND:    '&&';
OR:     '||';
NOT:    '!';
GT:     '>';

SQWORD: '\'' .*? '\'';
DQWORD: '"' .*? '"';
WORD:   ALPHA (ALPHA|DIGIT)*;
INT:    DIGIT+;

fragment ALPHA: [a-zA-Z_\-];
fragment DIGIT: [0-9];

WS: [ \t\r\n]+ -> skip;
```

`pdfhiscore`: Histogram Query Language (HQL)

- A simple boolean language (&&, ||, !) in prefix notation
- Words evaluate to true if they're in the histogram
- Similarly we may query a minimum count with the $>$ operator

```
1  grammar HistogramQueryLanguage;
2
3  prog: AND expr+
4      | OR expr+
5      | GT value+ INT
6      | expr+
7      ;
8
9  expr: '(' AND expr+ ')'
10     | '(' OR expr+ ')'
11     | '(' GT value+ INT ')'
12     | NOT expr
13     | value
14     ;
15
16 value: WORD | SQWORD | DQWORD;
17
18 AND:    '&&';
19 OR:     '||';
20 NOT:    '!';
21 GT:     '>';
22
23 SQWORD: '\'' .*? '\'';
24 DQWORD: '"' .*? '"';
25 WORD:   ALPHA (ALPHA|DIGIT)*;
26 INT:    DIGIT+;
27
28 fragment ALPHA: [a-zA-Z_\-];
29 fragment DIGIT: [0-9];
30
31 WS: [ \t\r\n]+ -> skip;
```

pdfhiscore: Histogram Query Language (HQL)

- A simple boolean language (&&, ||, !) in prefix notation
- Words evaluate to true if they're in the histogram
- Similarly we may query a minimum count with the $>$ operator

Example Expression:

&& *w*1 (|| *w*2 *w*3) !*w*4 ($>$ *w*5 *w*6 10)

```
1  grammar HistogramQueryLanguage;
2
3  prog: AND expr+
4      | OR expr+
5      | GT value+ INT
6      | expr+
7      ;
8
9  expr: '(' AND expr+ ')'
10     | '(' OR expr+ ')'
11     | '(' GT value+ INT ')'
12     | NOT expr
13     | value
14     ;
15
16 value: WORD | SQWORD | DQWORD;
17
18 AND:    '&&';
19 OR:     '||';
20 NOT:    '!';
21 GT:     '>';
22
23 SQWORD: '\'' .*? '\'';
24 DQWORD: '"' .*? '"';
25 WORD:   ALPHA (ALPHA|DIGIT)*;
26 INT:    DIGIT+;
27
28 fragment ALPHA: [a-zA-Z_\-];
29 fragment DIGIT: [0-9];
30
31 WS: [ \t\r\n]+ -> skip;
```

pdfhiscore: Histogram Query Language (HQL)

- A simple boolean language (&&, ||, !) in prefix notation
- Words evaluate to true if they're in the histogram
- Similarly we may query a minimum count with the $>$ operator

Example Expression:

&& *w*1 (|| *w*2 *w*3) !*w*4 ($>$ *w*5 *w*6 10)

- A full query is composed of multiple, weighted HQL expressions

```
1  grammar HistogramQueryLanguage;
2
3  prog: AND expr+
4      | OR expr+
5      | GT value+ INT
6      | expr+
7      ;
8
9  expr: '(' AND expr+ ')'
10     | '(' OR expr+ ')'
11     | '(' GT value+ INT ')'
12     | NOT expr
13     | value
14     ;
15
16 value: WORD | SQWORD | DQWORD;
17
18 AND:    '&&';
19 OR:     '||';
20 NOT:    '!';
21 GT:     '>';
22
23 SQWORD: '\'' .*? '\'';
24 DQWORD: '"' .*? '"';
25 WORD:   ALPHA (ALPHA|DIGIT)*;
26 INT:    DIGIT+;
27
28 fragment ALPHA: [a-zA-Z_\-];
29 fragment DIGIT: [0-9];
30
31 WS: [ \t\r\n]+ -> skip;
```

`pdfhiscore`: Histogram Query Language (HQL)

- A simple boolean language (&&, ||, !) in prefix notation
- Words evaluate to true if they're in the histogram
- Similarly we may query a minimum count with the $>$ operator

Example Expression:

&& *w*1 (|| *w*2 *w*3) !*w*4 ($>$ *w*5 *w*6 10)

- A full query is composed of multiple, weighted HQL expressions
- *hiscore*: the weighted sum of all expressions that evaluate to true

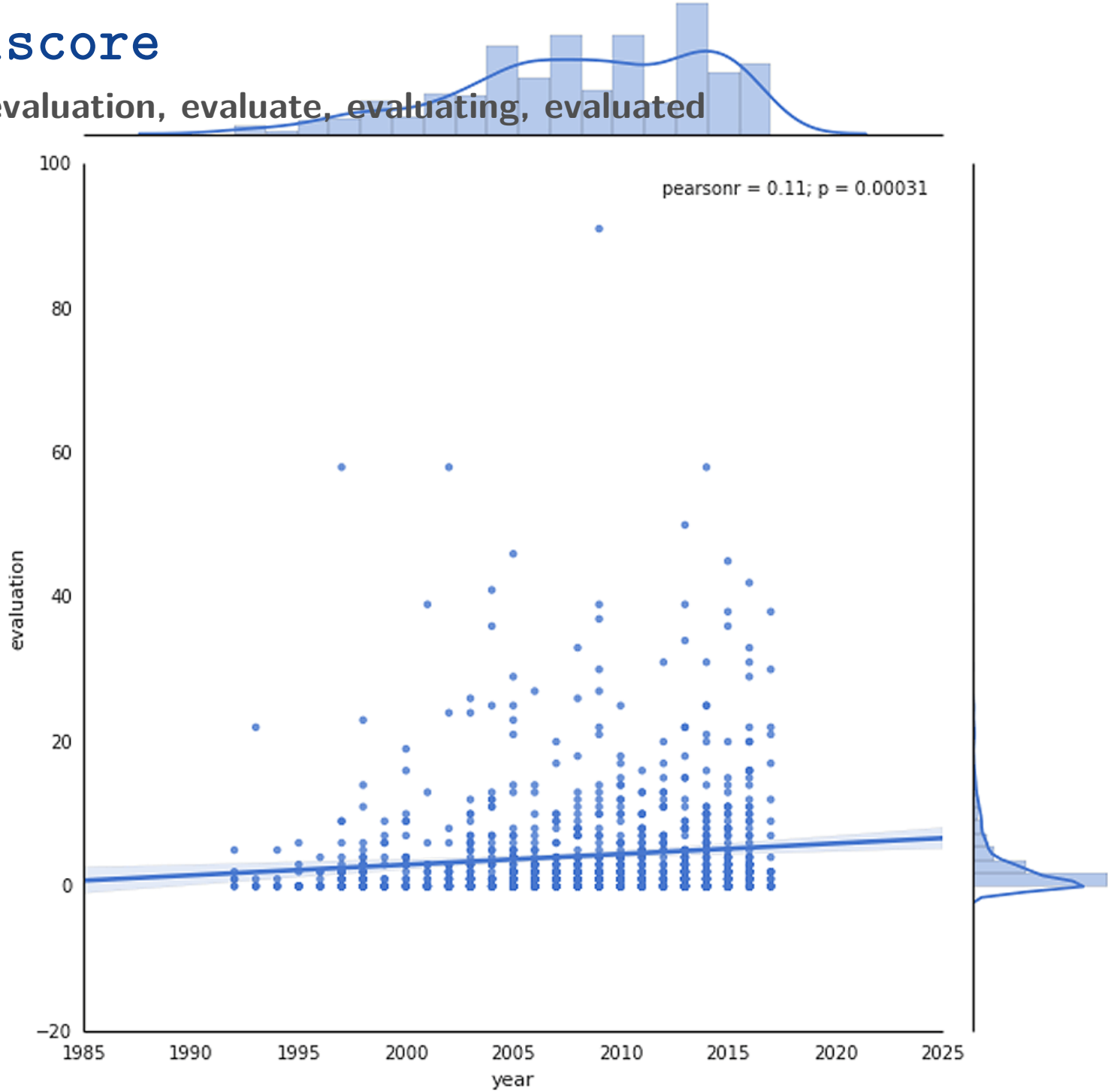
```
1  grammar HistogramQueryLanguage;
2
3  prog: AND expr+
4      | OR expr+
5      | GT value+ INT
6      | expr+
7      ;
8
9  expr: '(' AND expr+ ')'
10     | '(' OR expr+ ')'
11     | '(' GT value+ INT ')'
12     | NOT expr
13     | value
14     ;
15
16 value: WORD | SQWORD | DQWORD;
17
18 AND:    '&&';
19 OR:     '||';
20 NOT:    '!';
21 GT:     '>';
22
23 SQWORD: '\'' .*? '\'';
24 DQWORD: '"' .*? '"';
25 WORD:   ALPHA (ALPHA|DIGIT)*;
26 INT:    DIGIT+;
27
28 fragment ALPHA: [a-zA-Z_\-];
29 fragment DIGIT: [0-9];
30
31 WS: [ \t\r\n]+ -> skip;
```

pdfhiscore


pearsonr = 0.16; p = 4.8e-08
score
year
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
1985
1990
1995
2000
2005
2010
2015
2020
2025

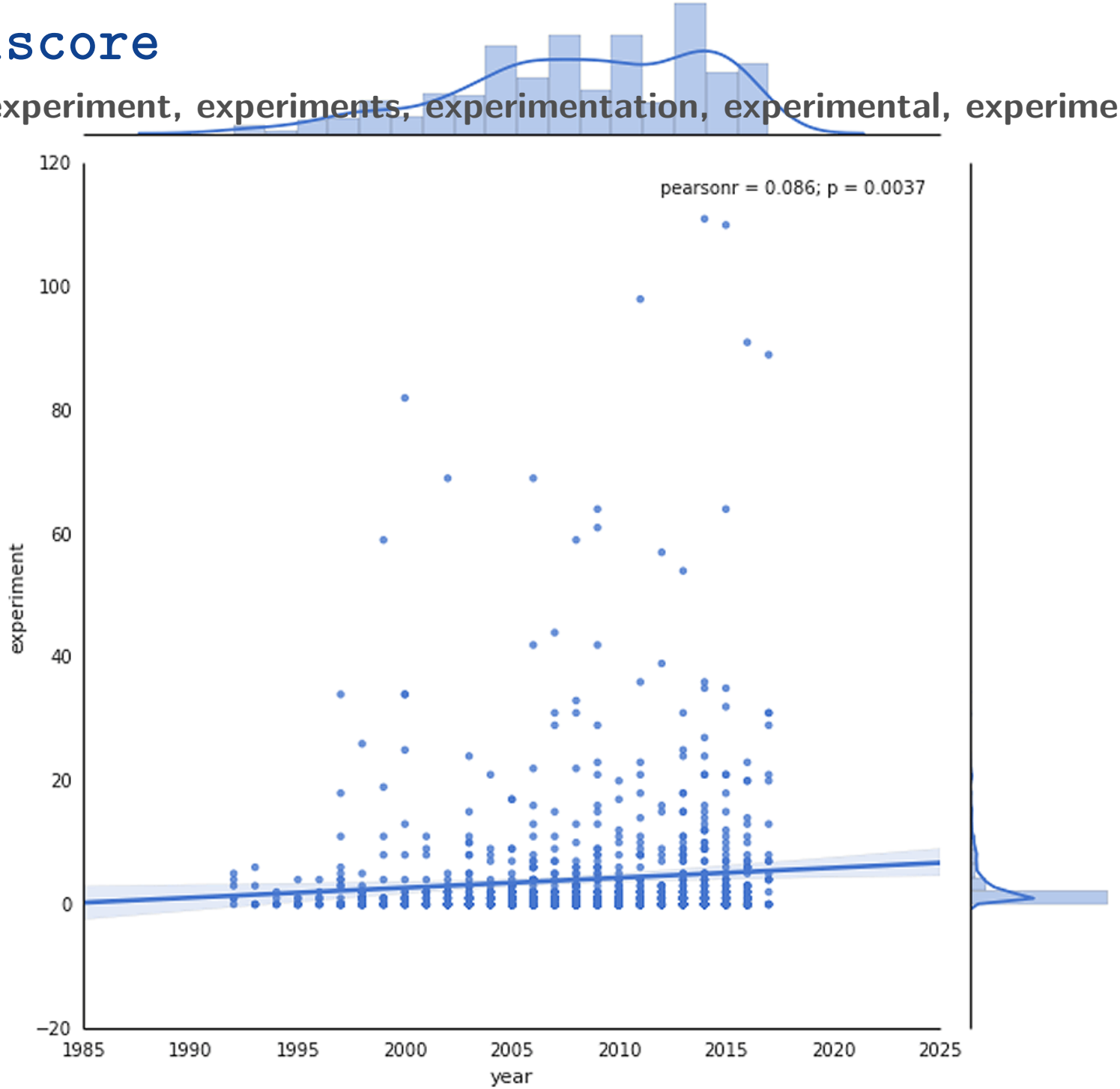
pdfhiscore

Words: evaluation, evaluate, evaluating, evaluated



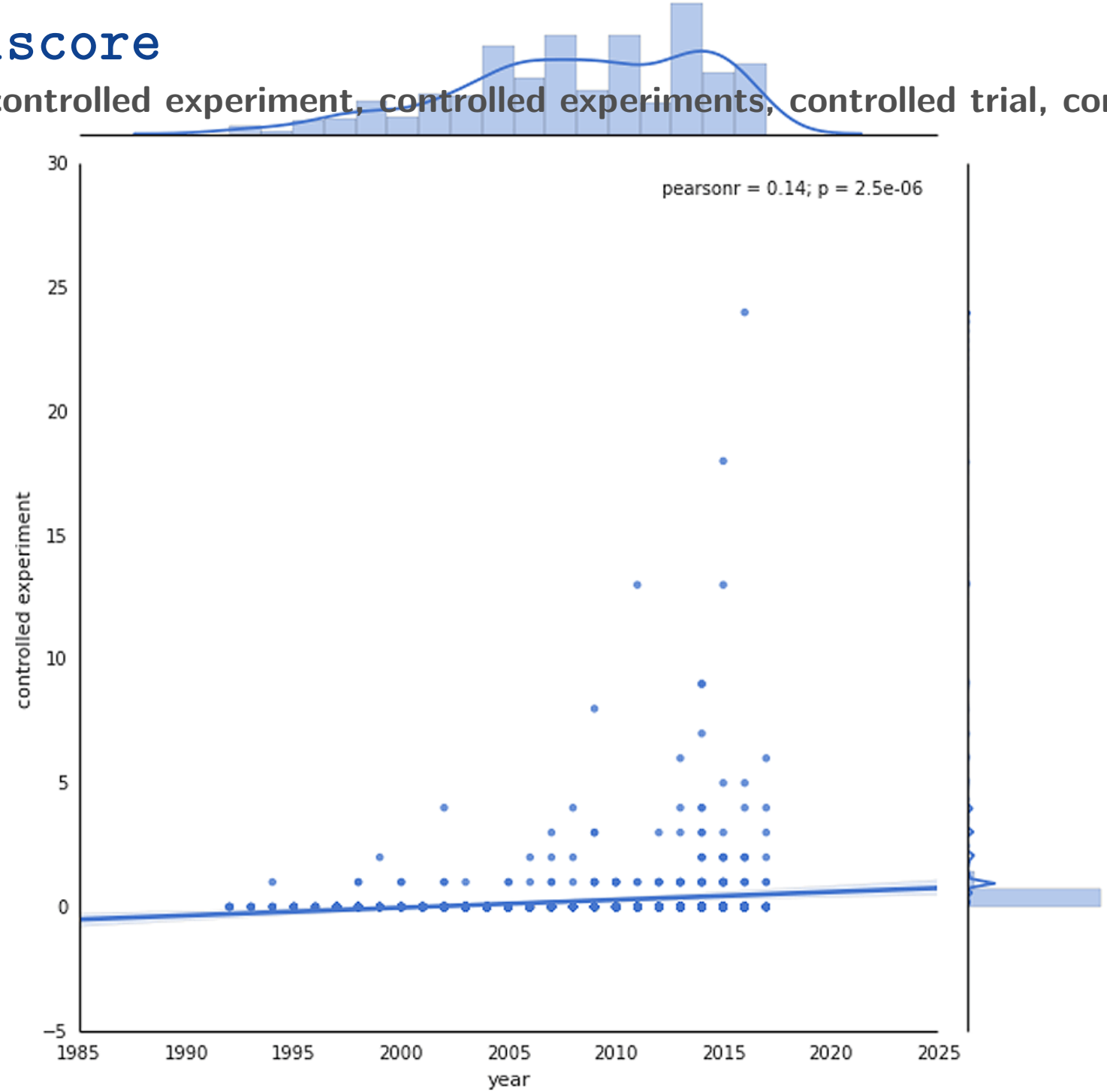
pdfhiscore

Words: experiment, experiments, experimentation, experimental, experimentally



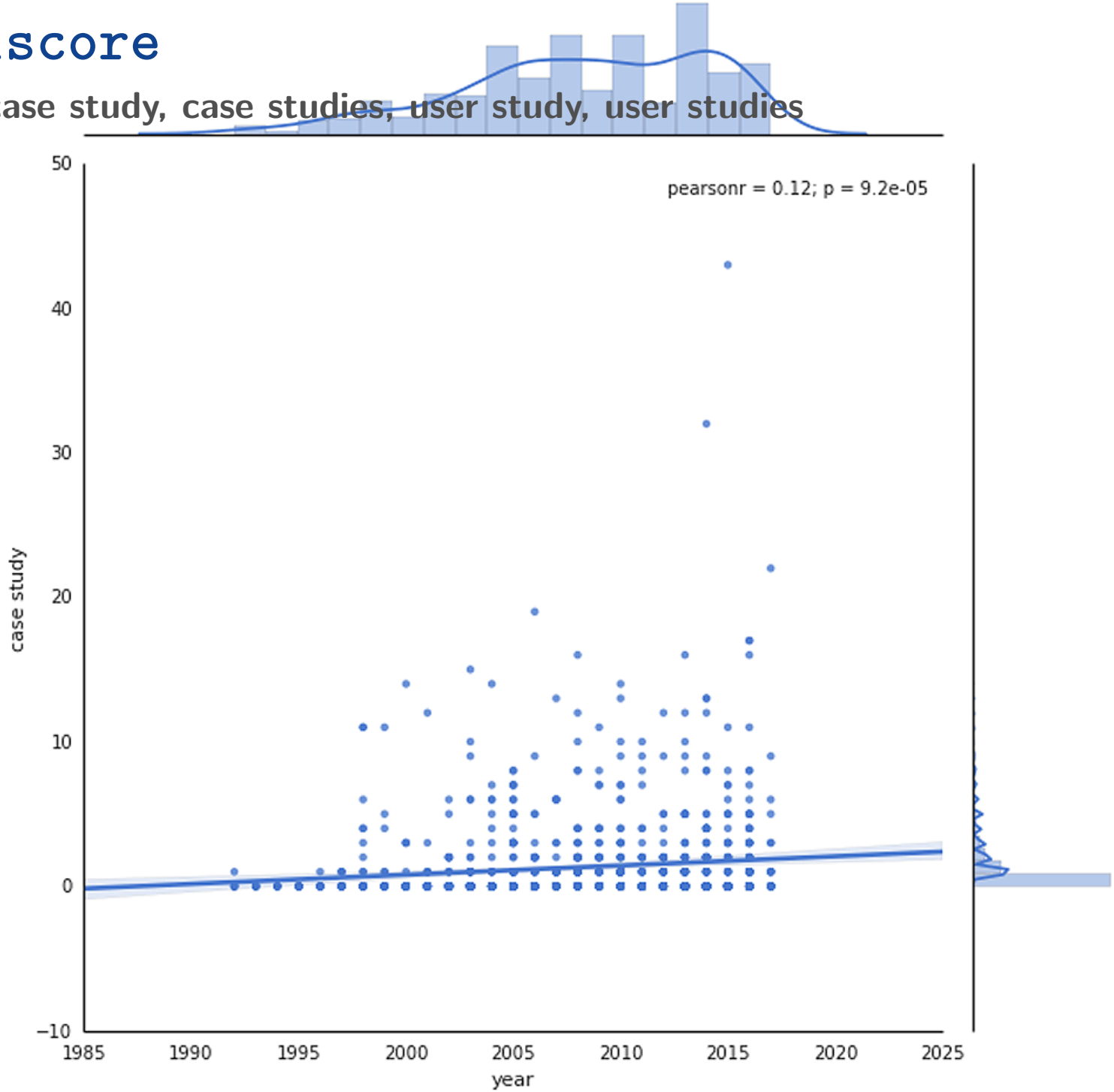
pdfhiscore

Words: controlled experiment, controlled experiments, controlled trial, controlled trials



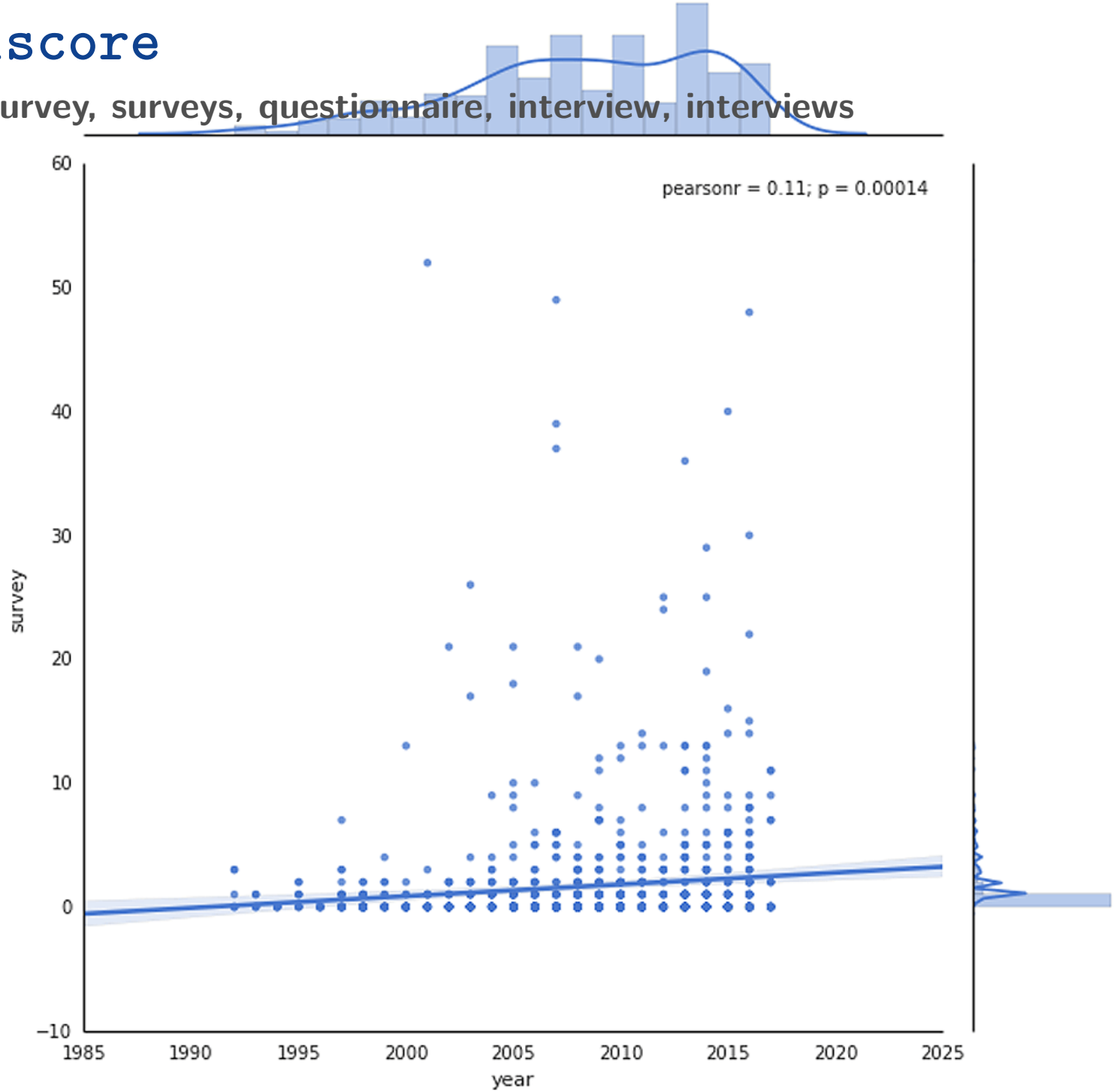
pdfhiscore

Words: case study, case studies, user study, user studies



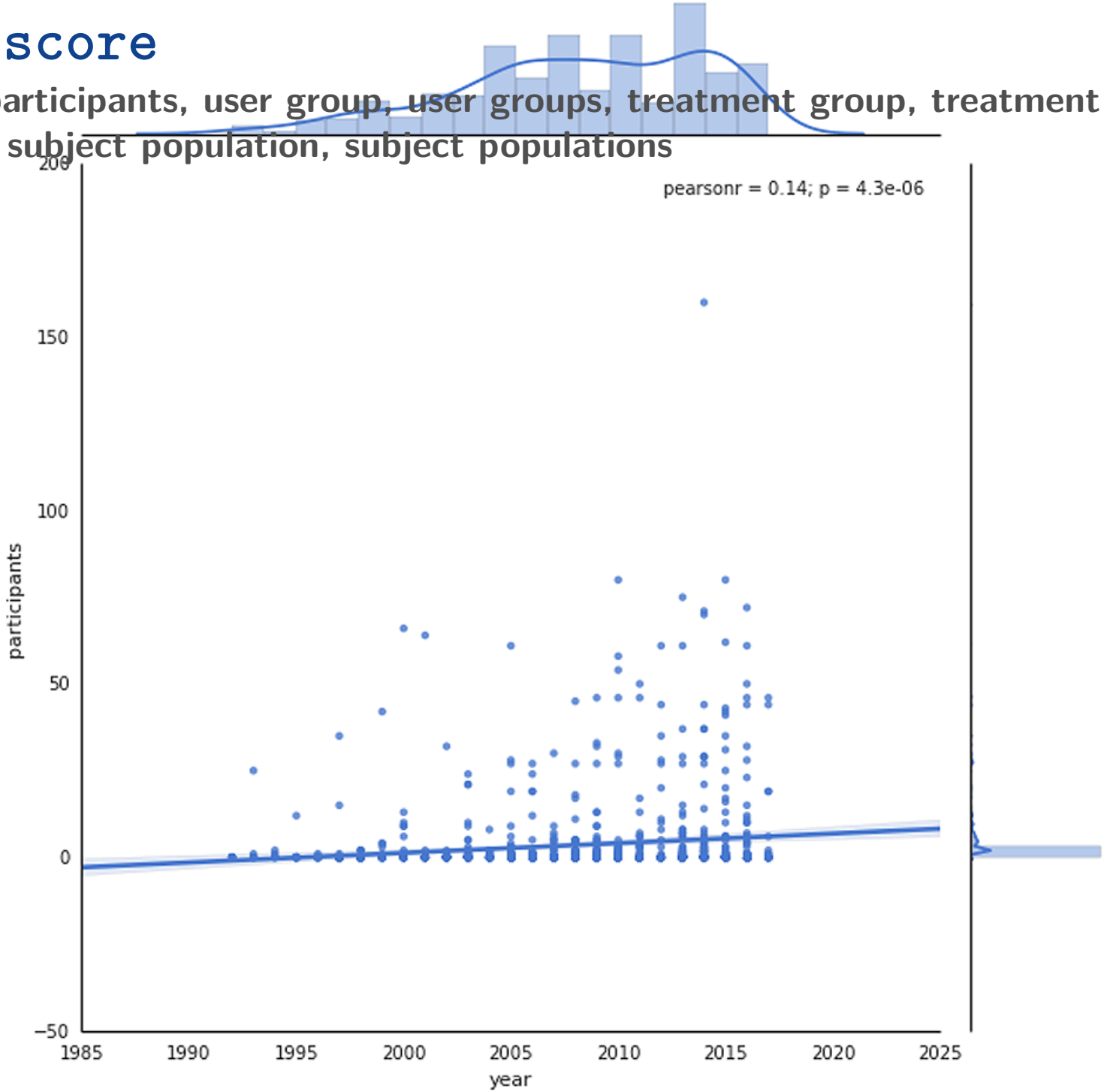
pdfhiscore

Words: survey, surveys, questionnaire, interview, interviews



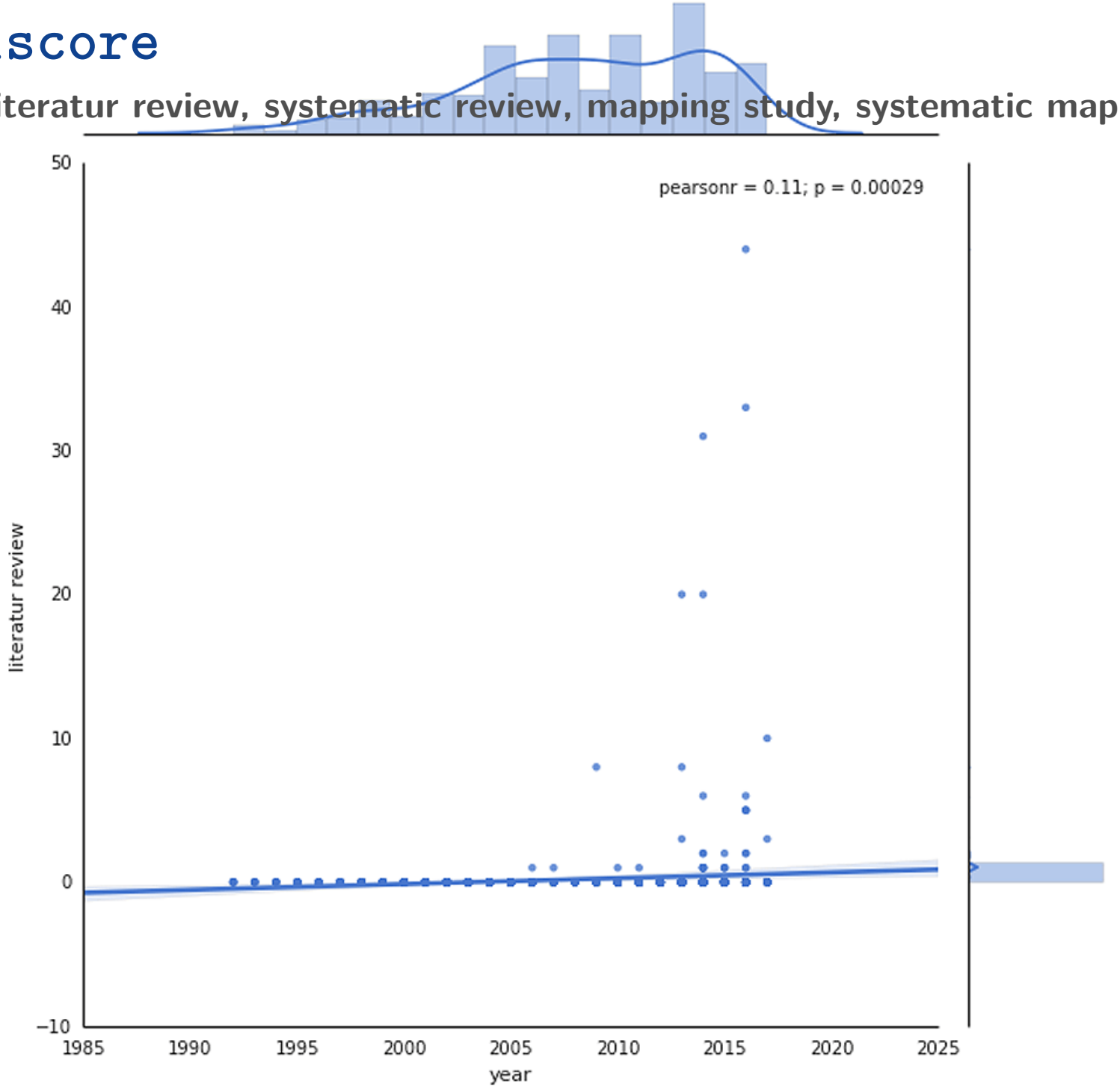
pdfhiscore

Words: participants, user group, user groups, treatment group, treatment groups, subjects, subject population, subject populations



pdfhiscore

Words: literatur review, systematic review, mapping study, systematic mapping



Thanks for listening

Code

- https://github.com/limstepf/pdfdbscrap
- https://github.com/limstepf/csvtobib
- https://github.com/limstepf/bibsani
- https://github.com/limstepf/pdfhiscore

Questions/Discussion