

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

?

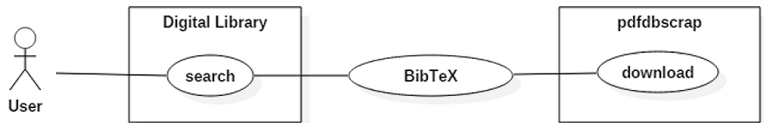


How to download a bunch of PDF files ...

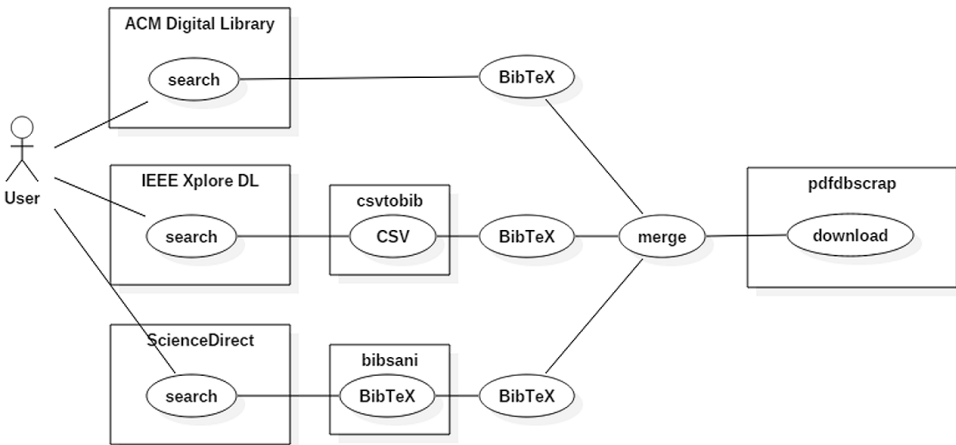
How to download a bunch of PDF files ...
...and what to do with them.

PDF Database Scrap(er): pdfdbscrap

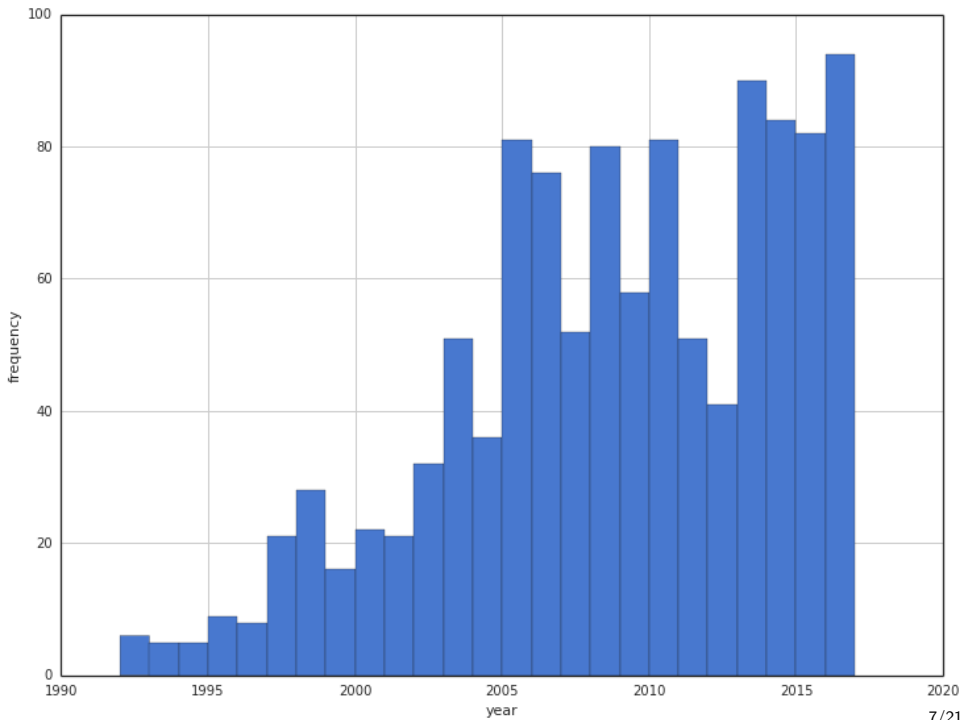
pdfdbscrap: General Use Case

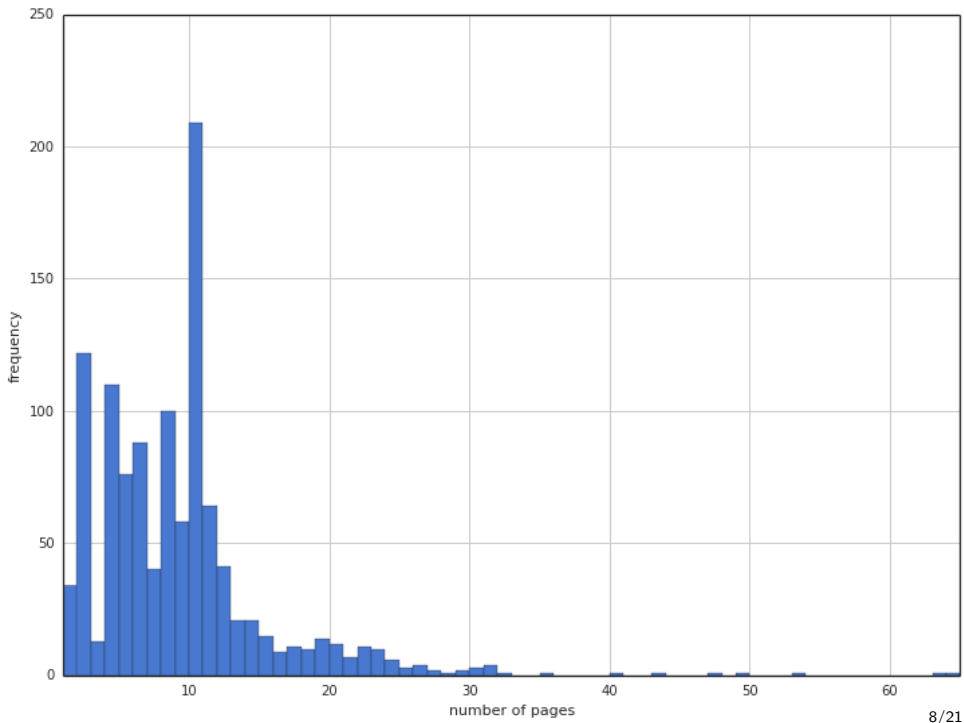


pdfdbscrap: Multi-Source Use Case



...and what to do with them.





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)

(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

(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

```
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

```
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

```
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:

&& w1 (|| w2 w3) !w4 (> w5 w6 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:

&& w1 (|| w2 w3) !w4 (> w5 w6 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

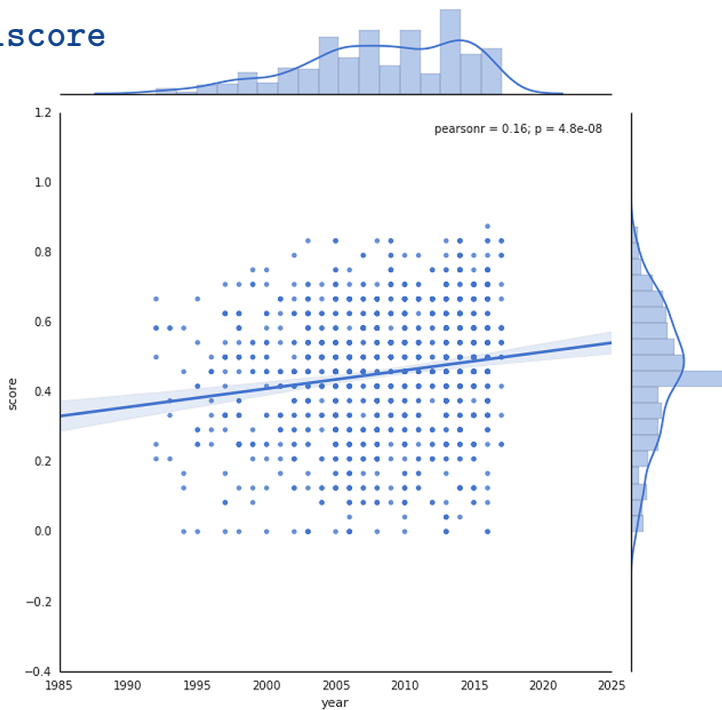
Example Expression:

&& w1 (|| w2 w3) !w4 (> w5 w6 10)

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

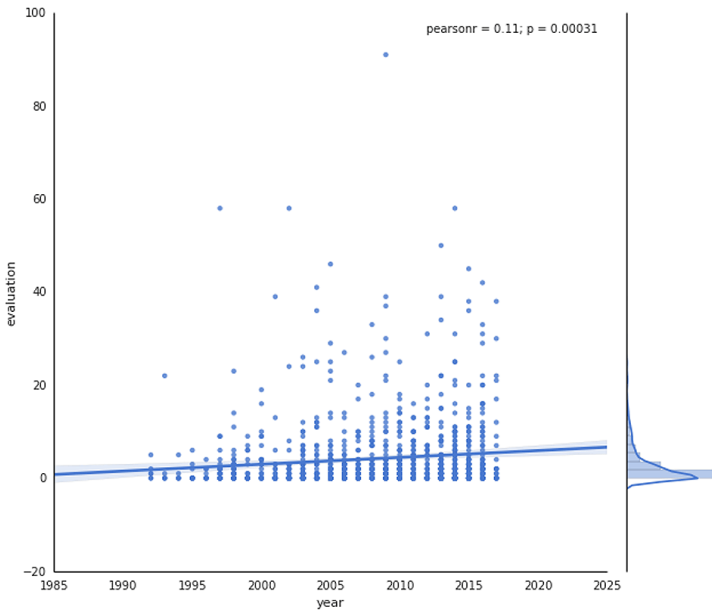
```
1 grammar HistogramQueryLanguage;
2
3 prog: AND expr+
4     | OR expr+
5     | GT value+ INT
6     | expr+
7     ;
8
9 expr: '(' AND expr+ ')'
10     | '(' OR expr+ ')'
11     | '(' GT value+ INT ')'
12     | NOT expr
13     | value
14     ;
15
16 value: WORD | SQWORD | DQWORD;
17
18 AND: '&&';
19 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



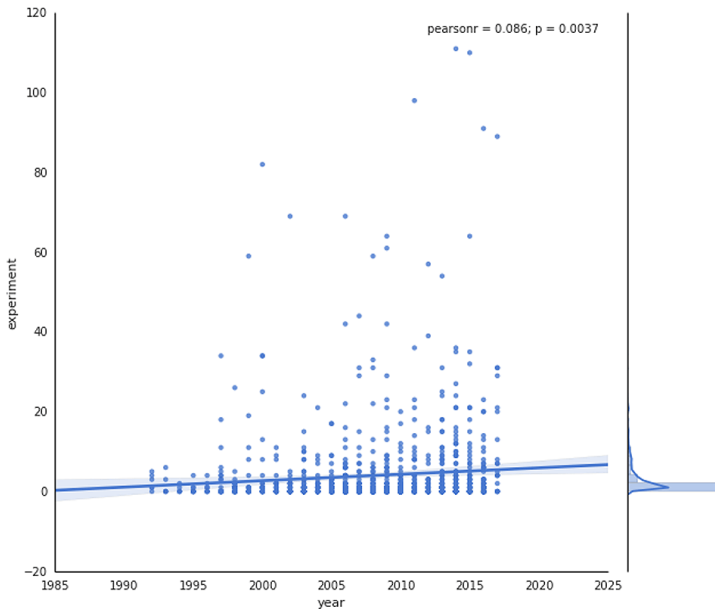
pdfhiscore

Words: evaluation, evaluate, evaluating, evaluated



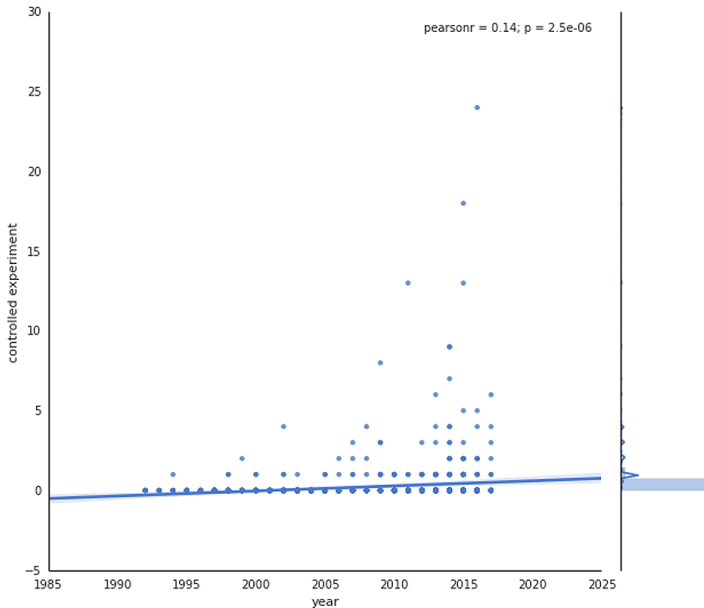
pdfhiscore

Words: experiment, experiments, experimentation, experimental, experimentally



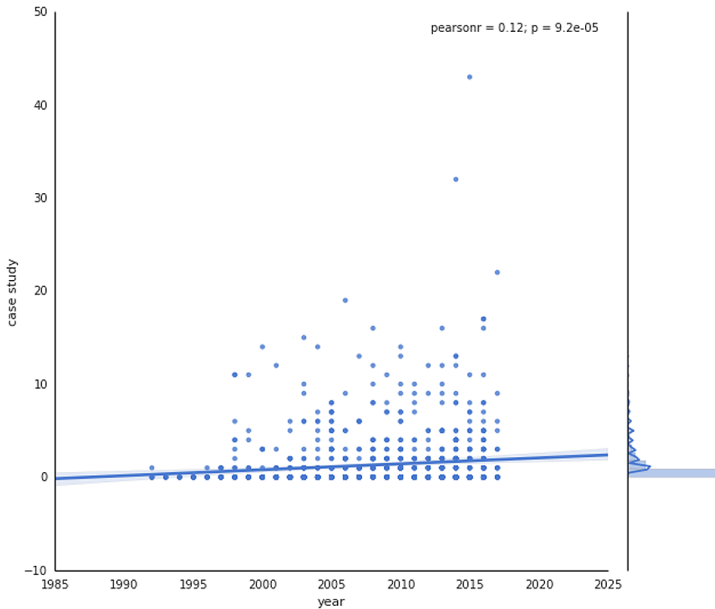
pdfhiscore

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



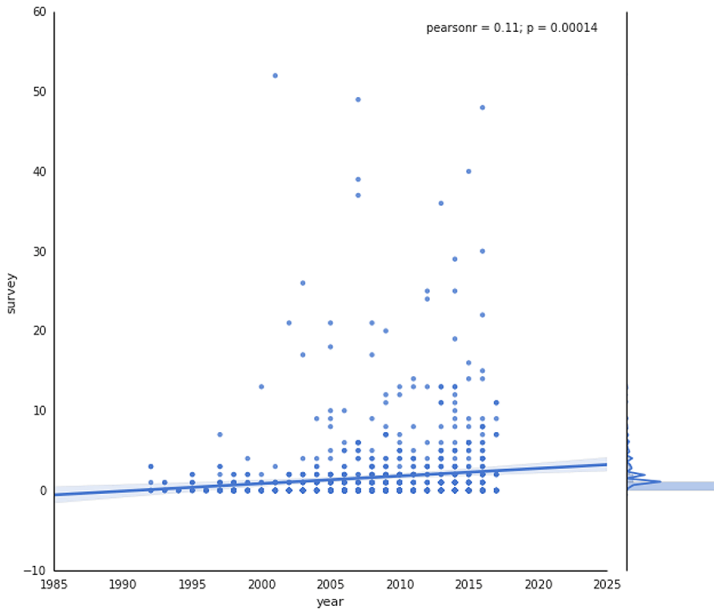
pdfhiscore

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



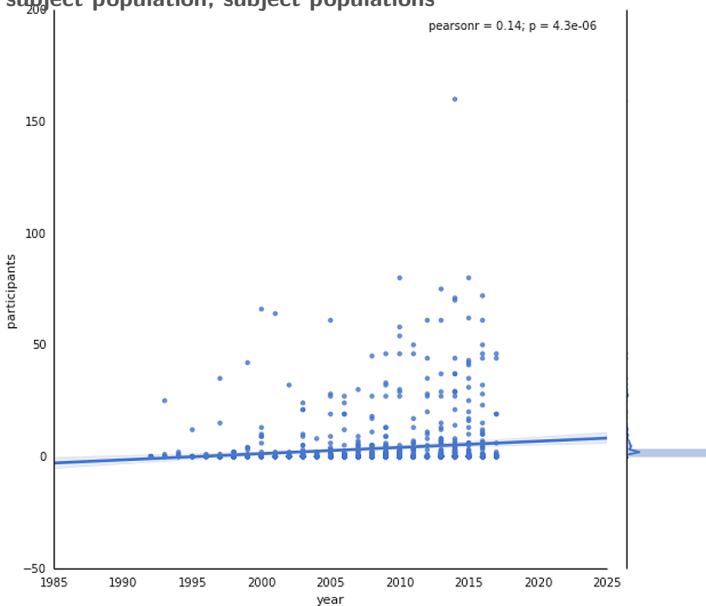
pdfhiscore

Words: survey, surveys, questionnaire, interview, interviews



pdfhiscore

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



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