

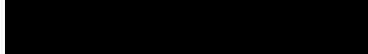
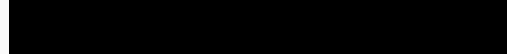


# ConfigCrusher: Towards White-Box Performance Analysis for Configurable Systems (Supplementary material)

## Main repo


 This repository Search Pull requests Issues Marketplace Explore

  Private 

[Code](#) [Issues 20](#) [Pull requests 0](#) [Boards](#) [Reports](#) [Projects 0](#) [Wiki](#) [Insights](#) [Settings](#)

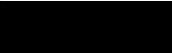
No description, website, or topics provided. [Edit](#)






[Add topics](#)


[982 commits](#) [13 branches](#) [0 releases](#) 

Branch: [supplementary](#) [New pull request](#) [Create new file](#) [Upload files](#) [Find file](#) [Clone or download](#)

This branch is 224 commits ahead of develop. [Pull request](#) [Compare](#)

 updated readme and license Latest commit aac970b an hour ago

 <a href="#">src</a>	got new data email	an hour ago
 <a href="#">.gitattributes.backup</a>	Used all bandwidth for git lfs	8 months ago
 <a href="#">.gitignore</a>	Got data for elevator	9 days ago
 <a href="#">LICENSE.md</a>	updated readme and license	an hour ago
 <a href="#">README.md</a>	updated readme and license	an hour ago

[README.md](#) 

## ConfigCrusher: Towards White-Box Performance Analysis for Configurable Systems

This repo contains all material (implementations of ConfigCrusher and state-of-the-art black-box and white-box approaches, data, scripts, results, etc.) of our novel white-box performance analysis and empirical evaluation to state-of-the-art approaches. This research shows the benefits and potential of our white-box analysis to efficiently generate performance models.

### Abstract

In configurable software systems, stakeholders are often interested in knowing how configuration options influence the performance of a system to facilitate, for example, the debugging and optimization processes of these systems. There are several black-box approaches to obtain this information, but they either sample the system end-to-end with a large number of configurations to make accurate predictions or miss important performance-influencing interactions when sampling few configurations. In addition, these approaches cannot pinpoint the parts of a program that are responsible for performance differences among configurations. This paper proposes ConfigCrusher, a white-box performance analysis that inspects the implementation of a system to guide the performance analysis and exploits several insights about configurable systems in the process. ConfigCrusher employs a static data-flow analysis to identify how configuration options may influence control-flow decisions and instruments code regions corresponding to these decisions to dynamically analyze the influence of configuration options on the regions' performance. Our evaluation shows the feasibility of our white-box approach to more efficiently build performance models that are similar to or more accurate than current state-of-the-art approaches on 10 configurable systems. Overall, we showcase the benefits of white-box performance analyses and their potential to outperform black-box approaches and provide additional information for analyzing configurable systems.

### Supplementary material

[Link](#)

## License

MIT License

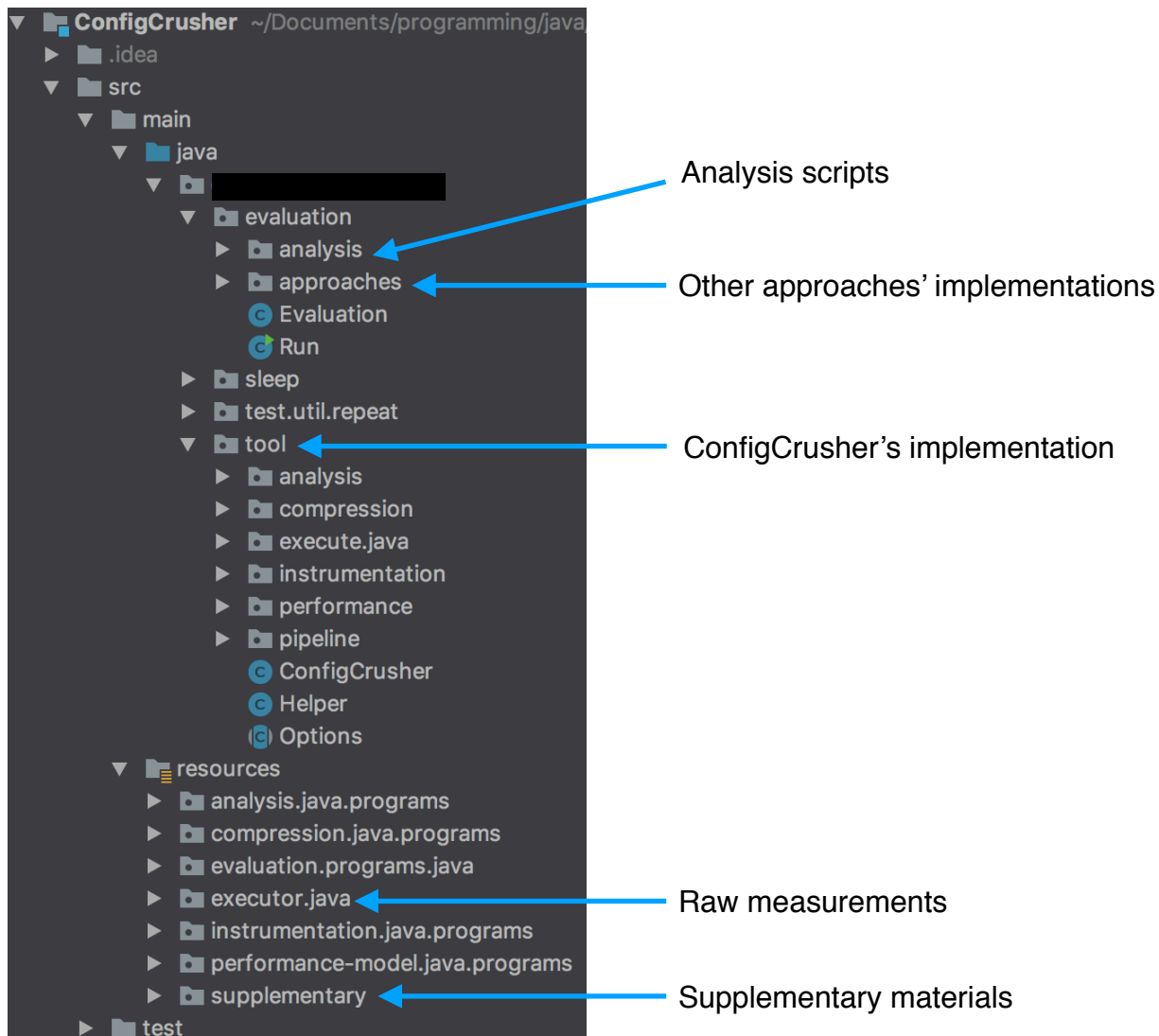
Copyright (c) 2018 [REDACTED]

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## Structure



Static taint analysis repo

<> CodePull requests 0BoardsReportsProjects 0WikiInsightsSettings

No description, website, or topics provided.

Edit

Add topics

Branch: developNew pull request

Create new fileUpload filesFind fileClone or download

Latest commit f1528a6 12 days ago

.idea	Executed after not being able to compile	6 months ago
dotStringOutput	Analyzed elevator	12 days ago
heros	Subtree heros	6 months ago
jasmin	Subtree jasmin	6 months ago
soot-infowflow	Fixed, yet again, how static methods are handled	5 months ago
soot	Added nops before each statement since the analysis was incorrect for	5 months ago
sootOutput	Analyzed elevator	12 days ago
src	Analyzed elevator	12 days ago
.gitignore	Updated gitignore	7 months ago
pom.xml	Updated name	6 months ago

Subject systems repo

Private

<> CodeIssues 0Pull requests 0BoardsReportsProjects 0WikiInsightsSettings

No description, website, or topics provided.

Edit

Add topics

150 commits1 branch0 releases

Branch: masterNew pull request

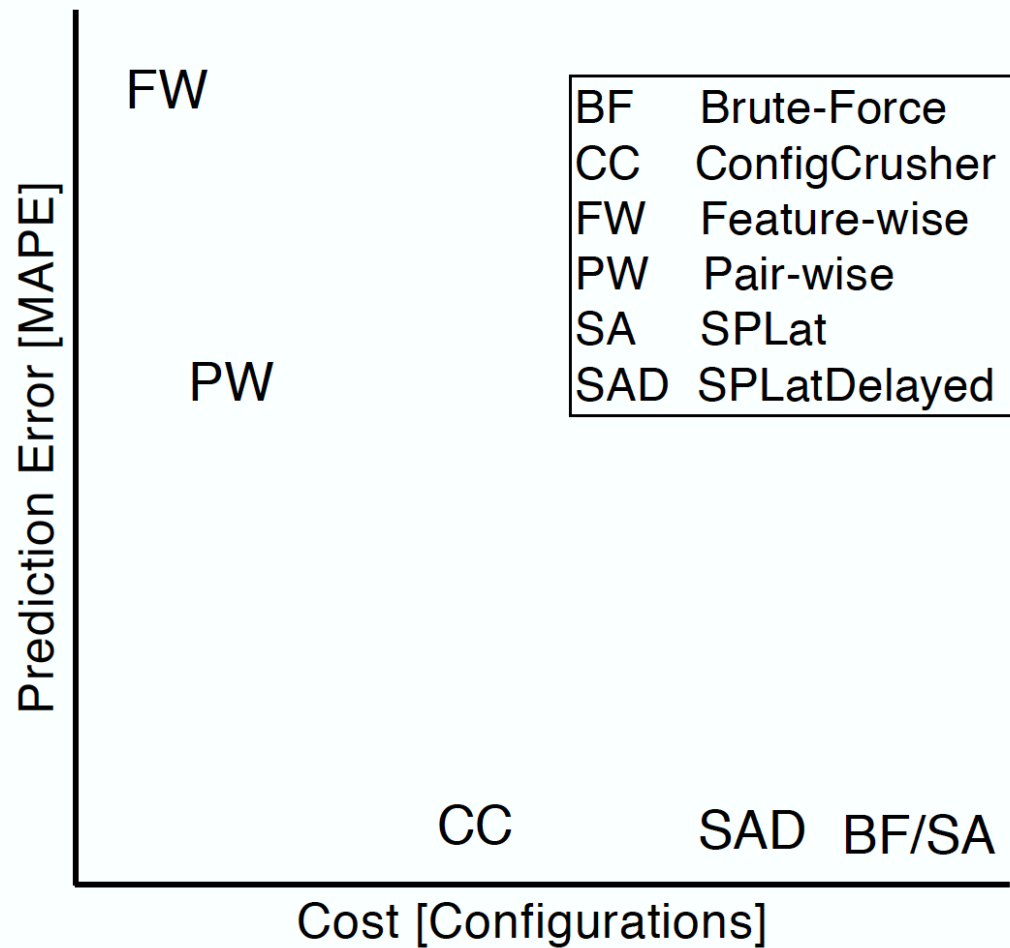
Create new fileUpload filesFind fileClone or download

Latest commit c882c4b a day ago

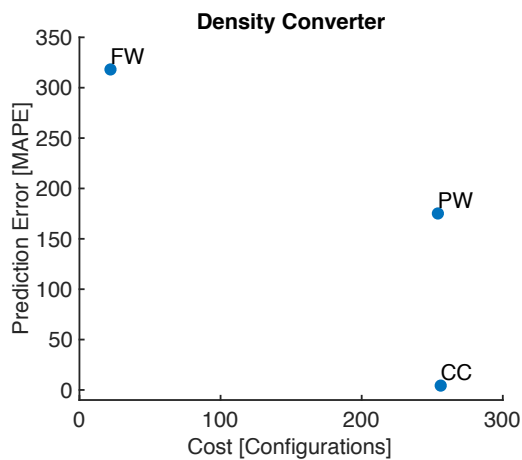
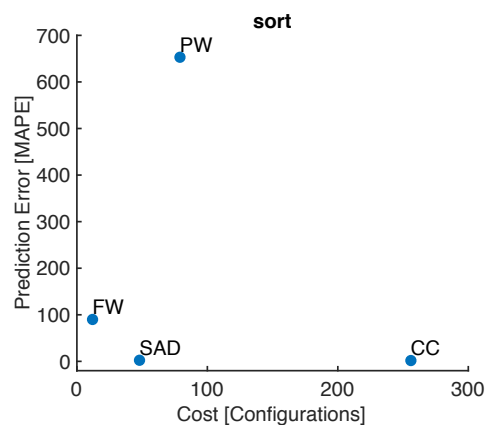
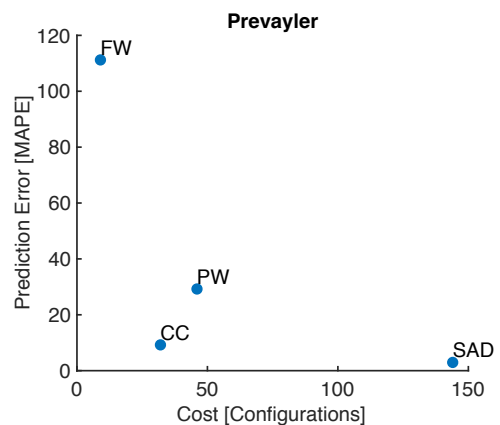
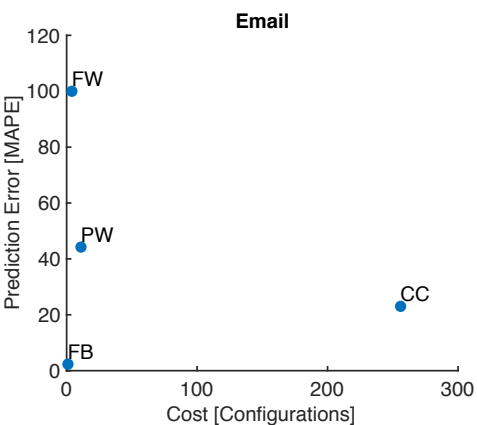
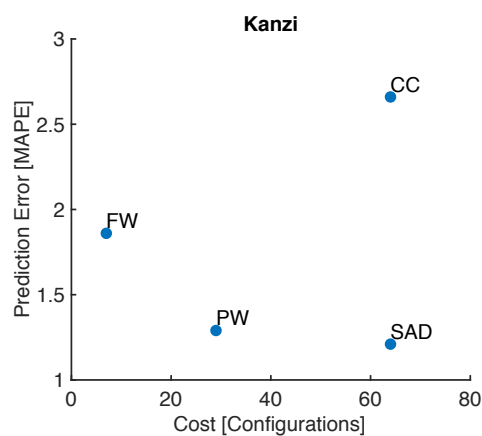
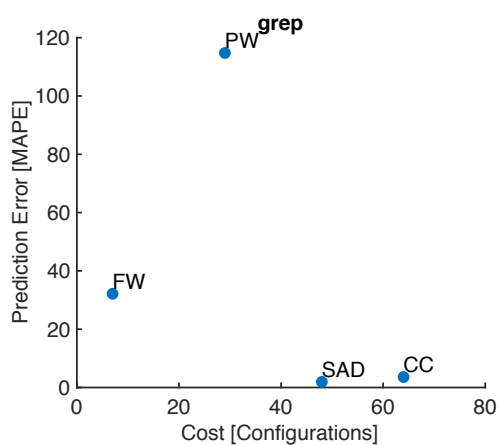
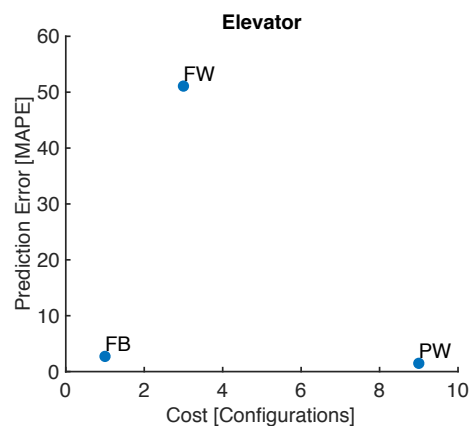
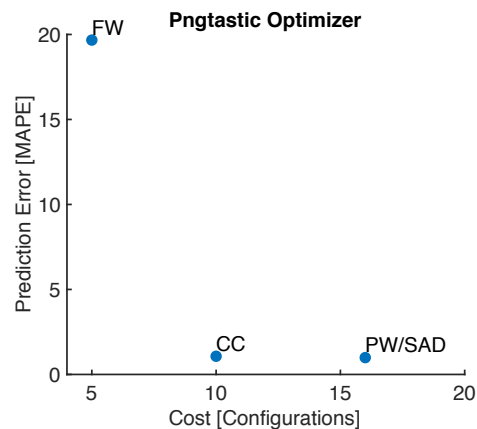
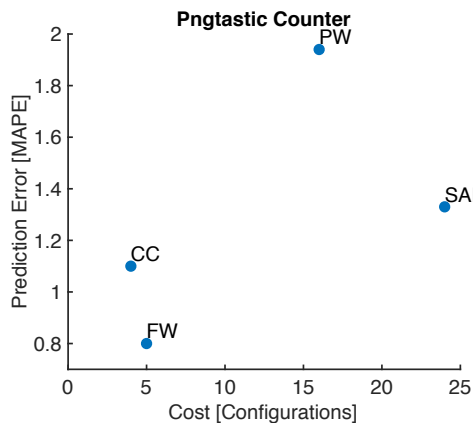
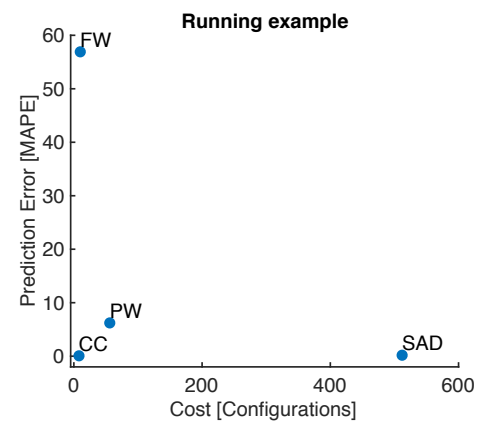
family	Added systems	5 days ago
instrumented	got email ready	a day ago
original	got email ready	a day ago
splat	Got data for email	4 days ago
.gitignore	Updated gitignore	11 days ago
performance-mapper-evaluation.iml	Added source code for java-lame	9 months ago

Conjecture on cost and prediction error comparison between state-of-the-art approaches

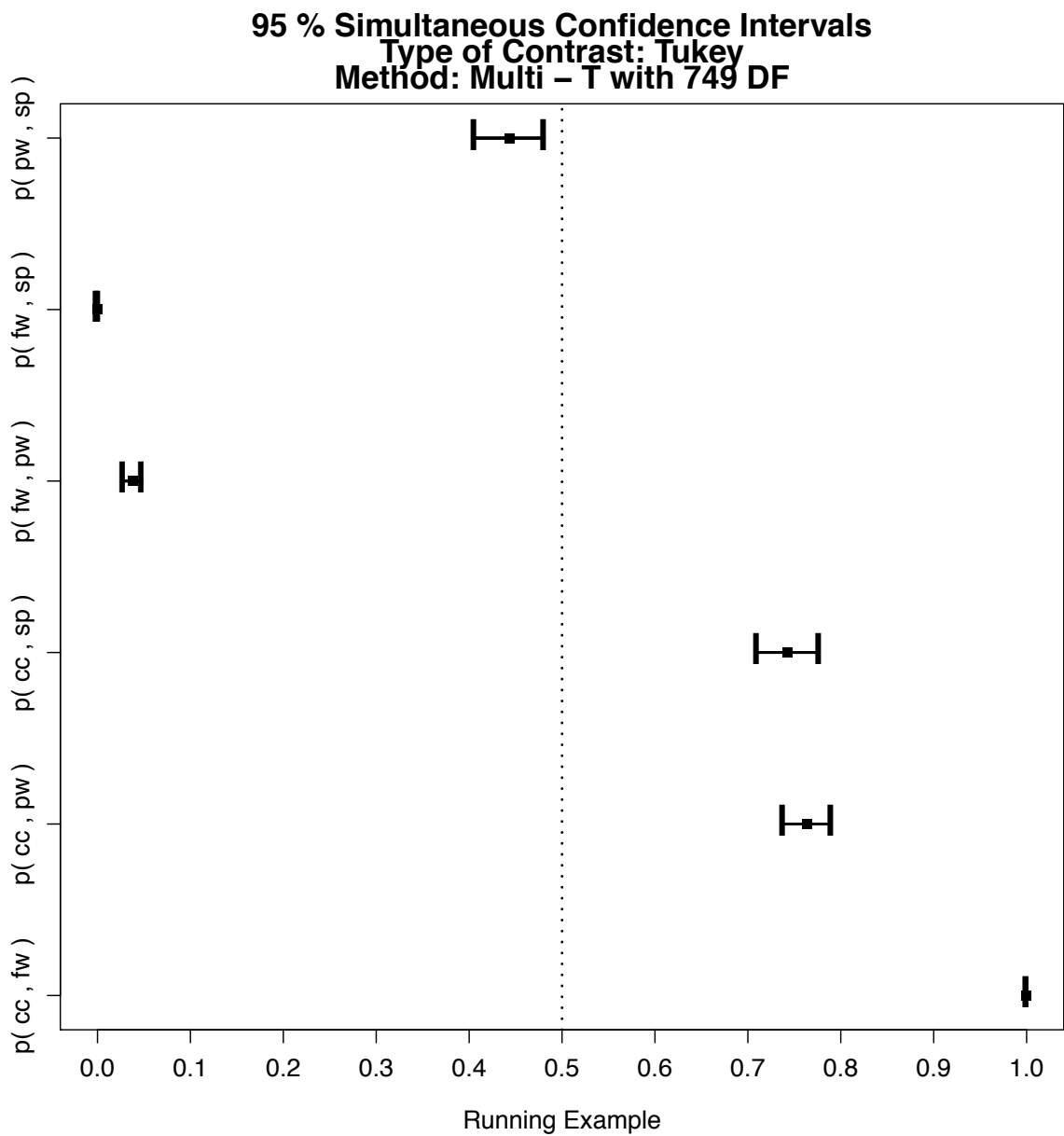
## Performance modeling approaches



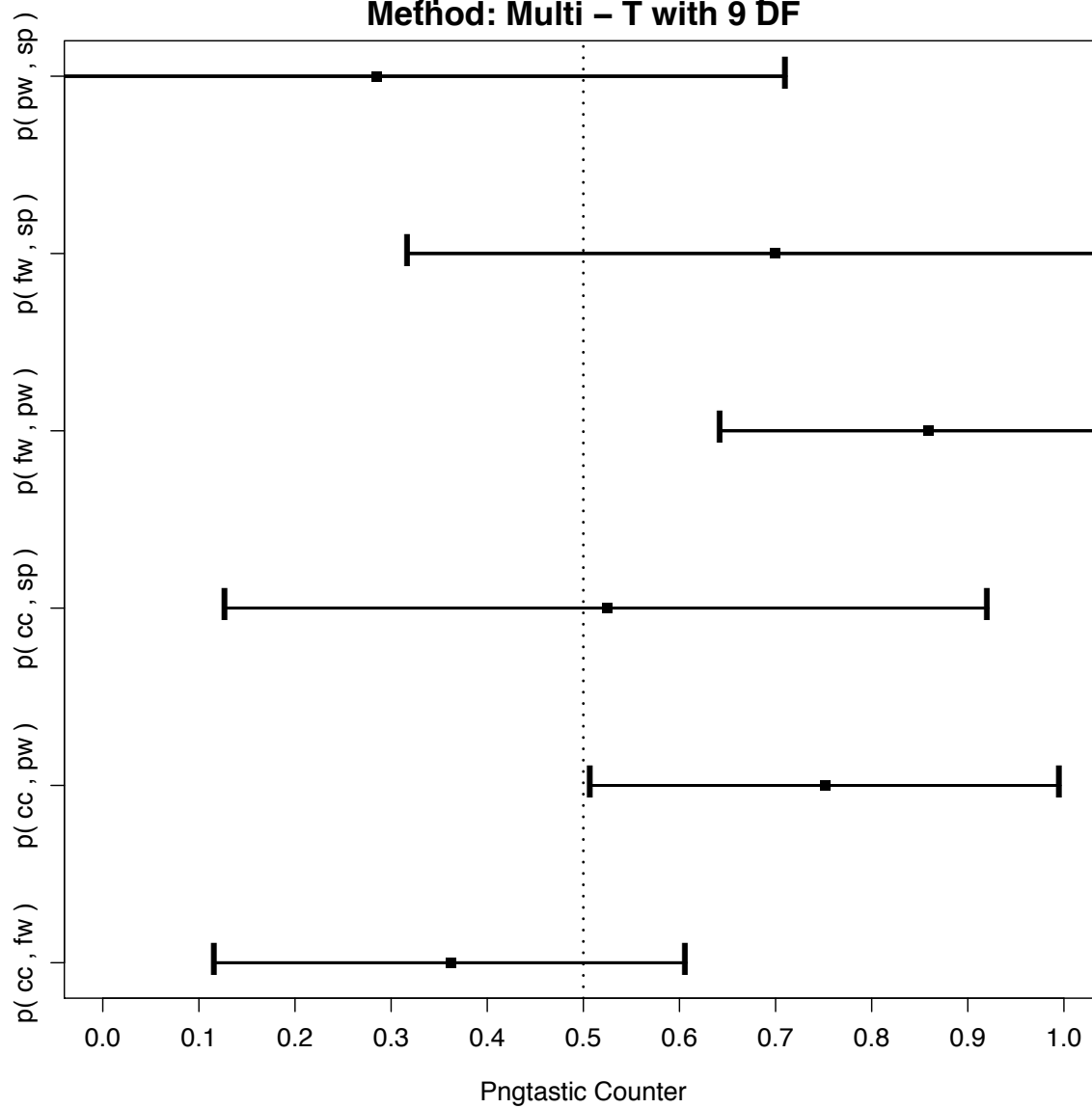
## Cost vs prediction error



T-procedure error comparison (Interpretation  $p(a,b) > 1/2$  :  $b$  tends to be larger than  $a$ )

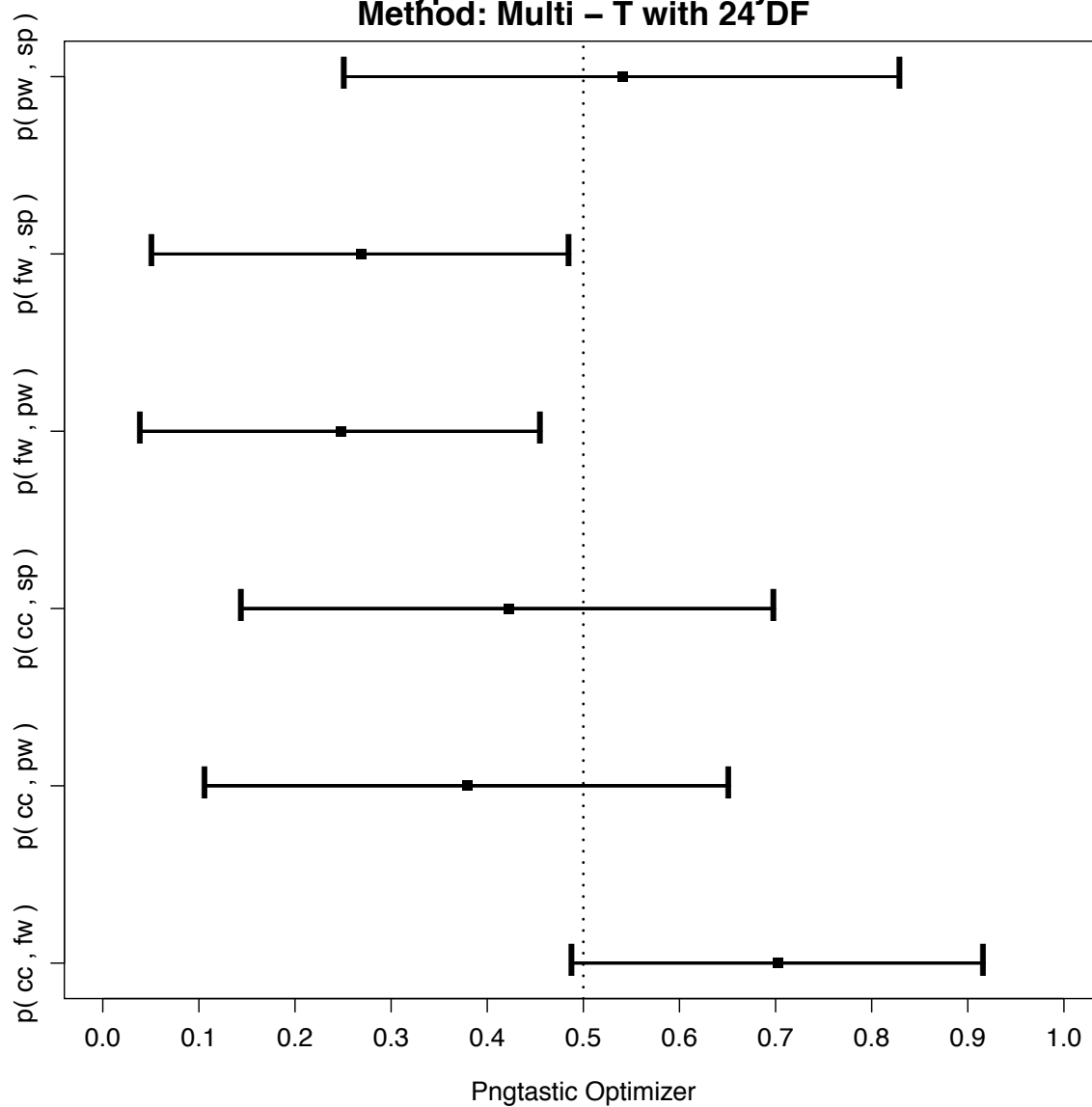


95 % Simultaneous Confidence Intervals  
Type of Contrast: Tukey  
Method: Multi - T with 9 DF



# 95 % Simultaneous Confidence Intervals

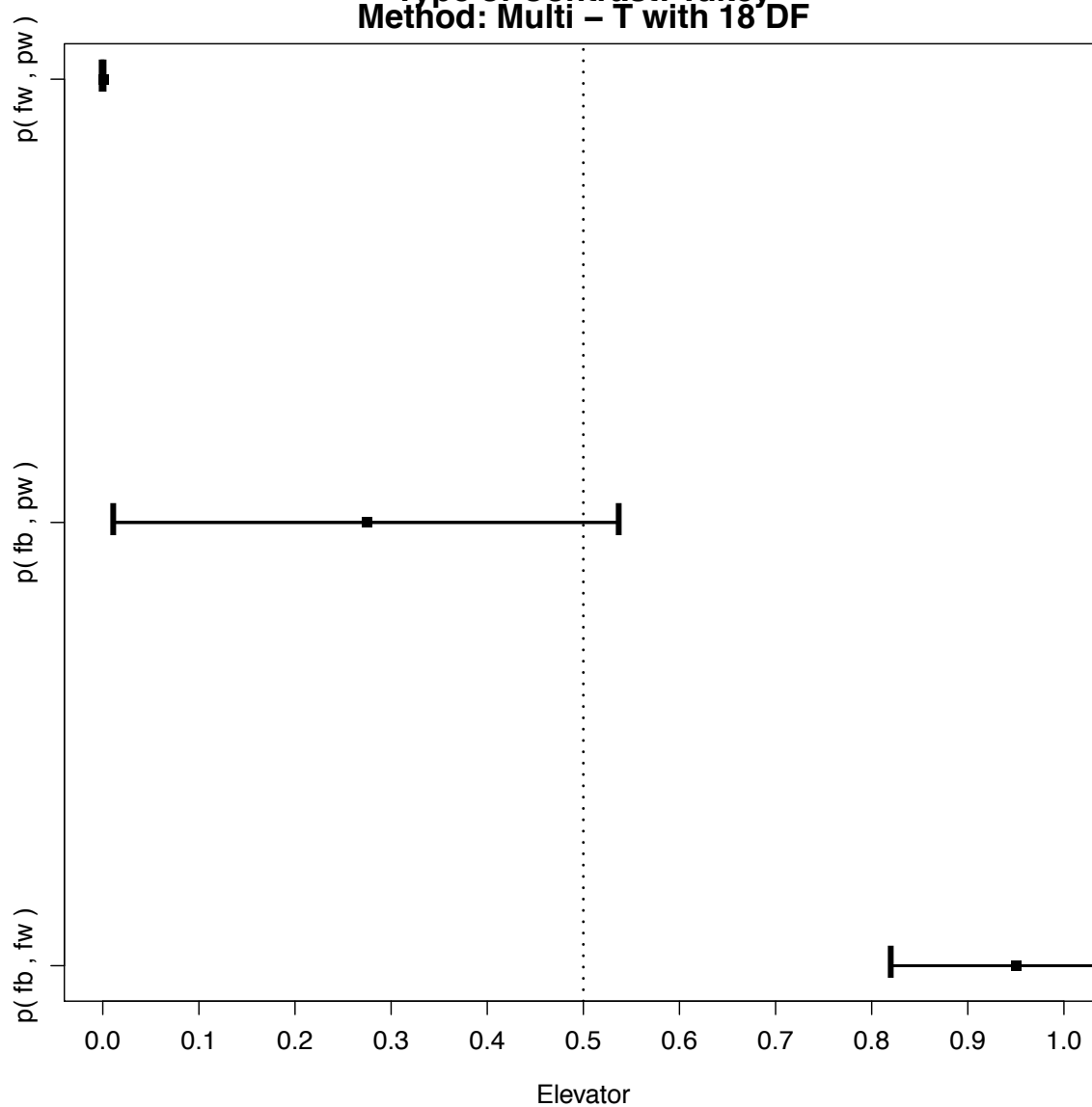
Type of Contrast: Tukey  
Method: Multi - T with 24 DF





# 95 % Simultaneous Confidence Intervals

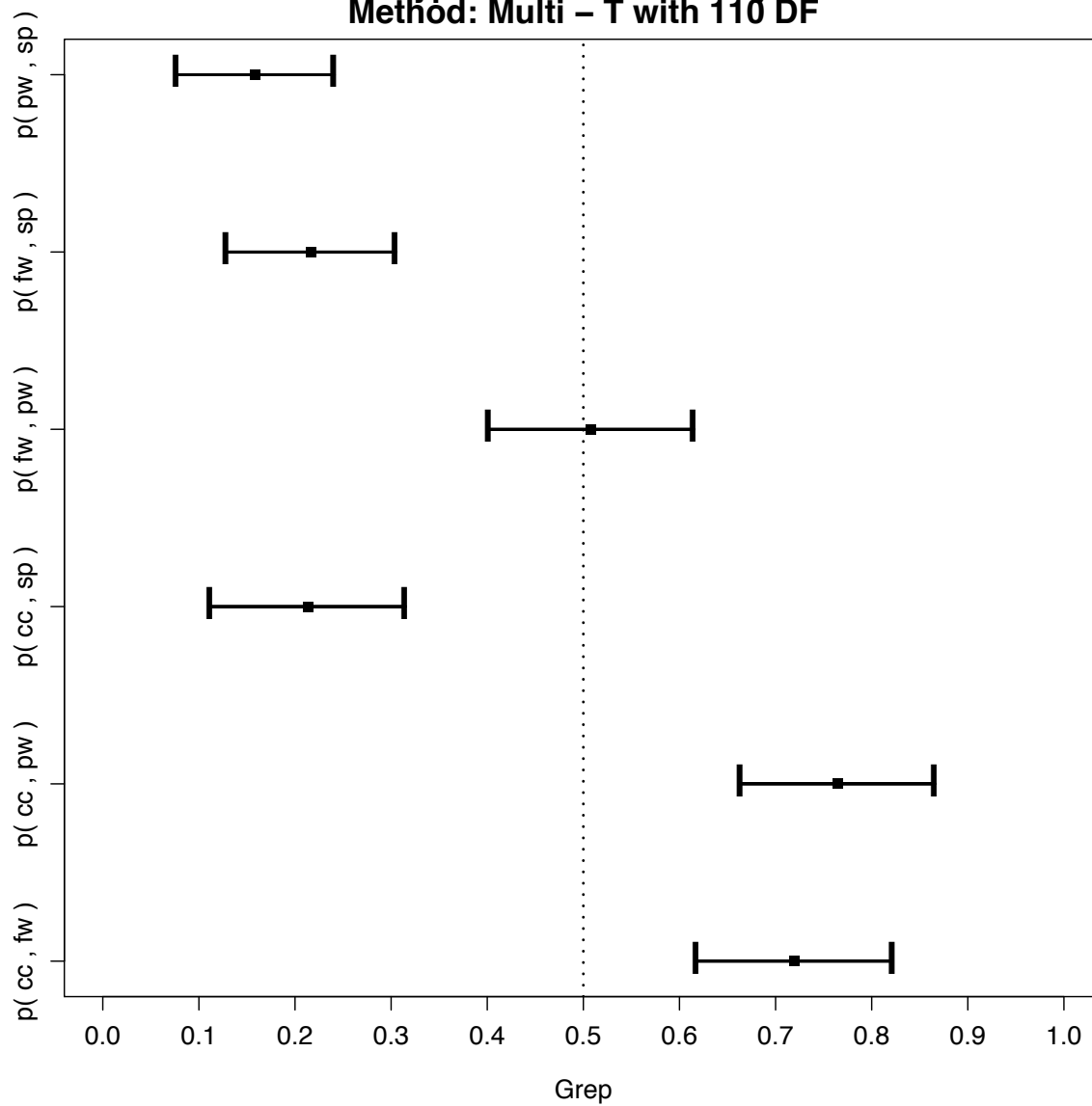
Type of Contrast: Tukey  
Method: Multi - T with 18 DF



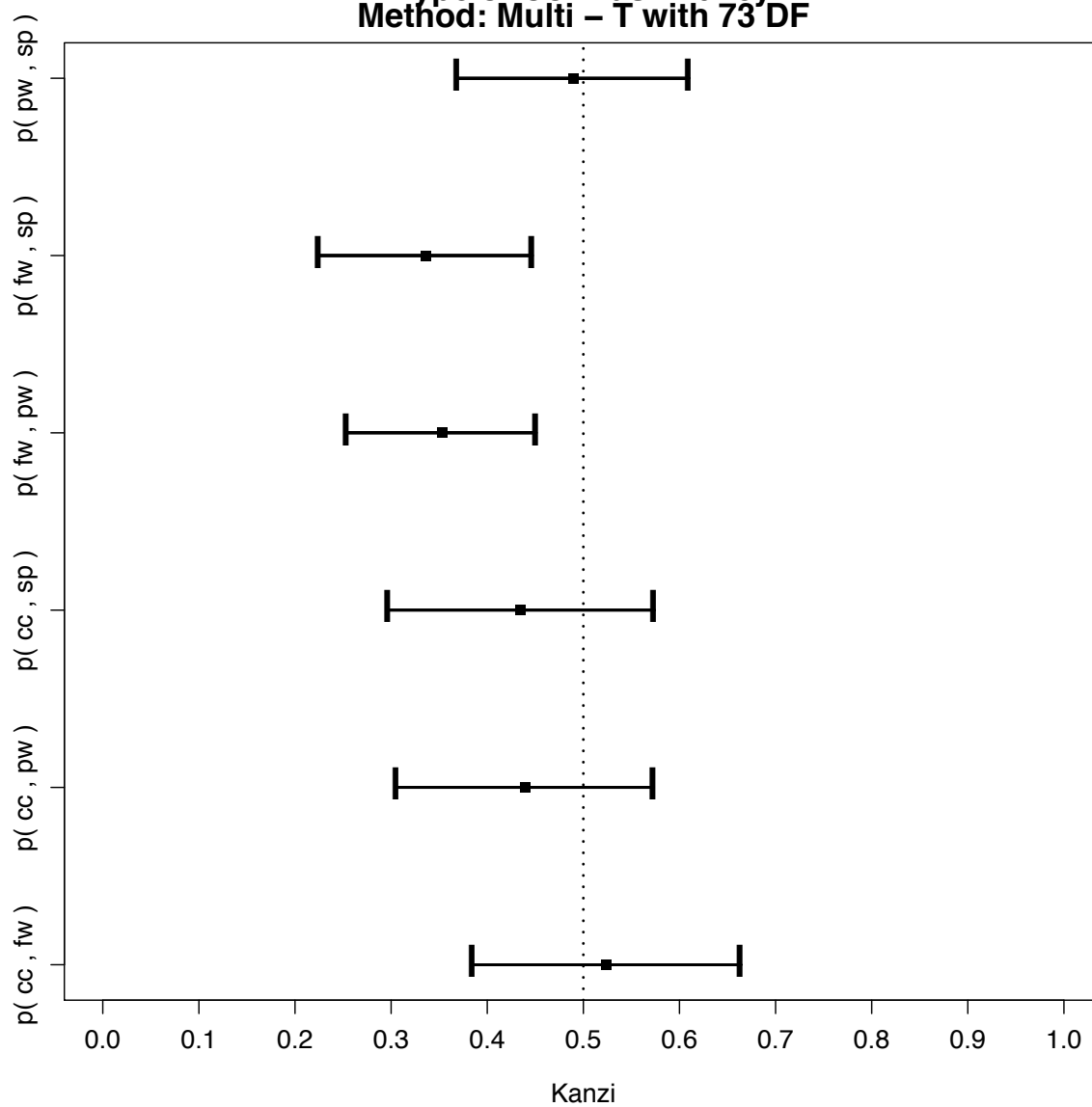
95 % Simultaneous Confidence Intervals

Type of Contrast: Tukey

Method: Multi - T with 110 DF

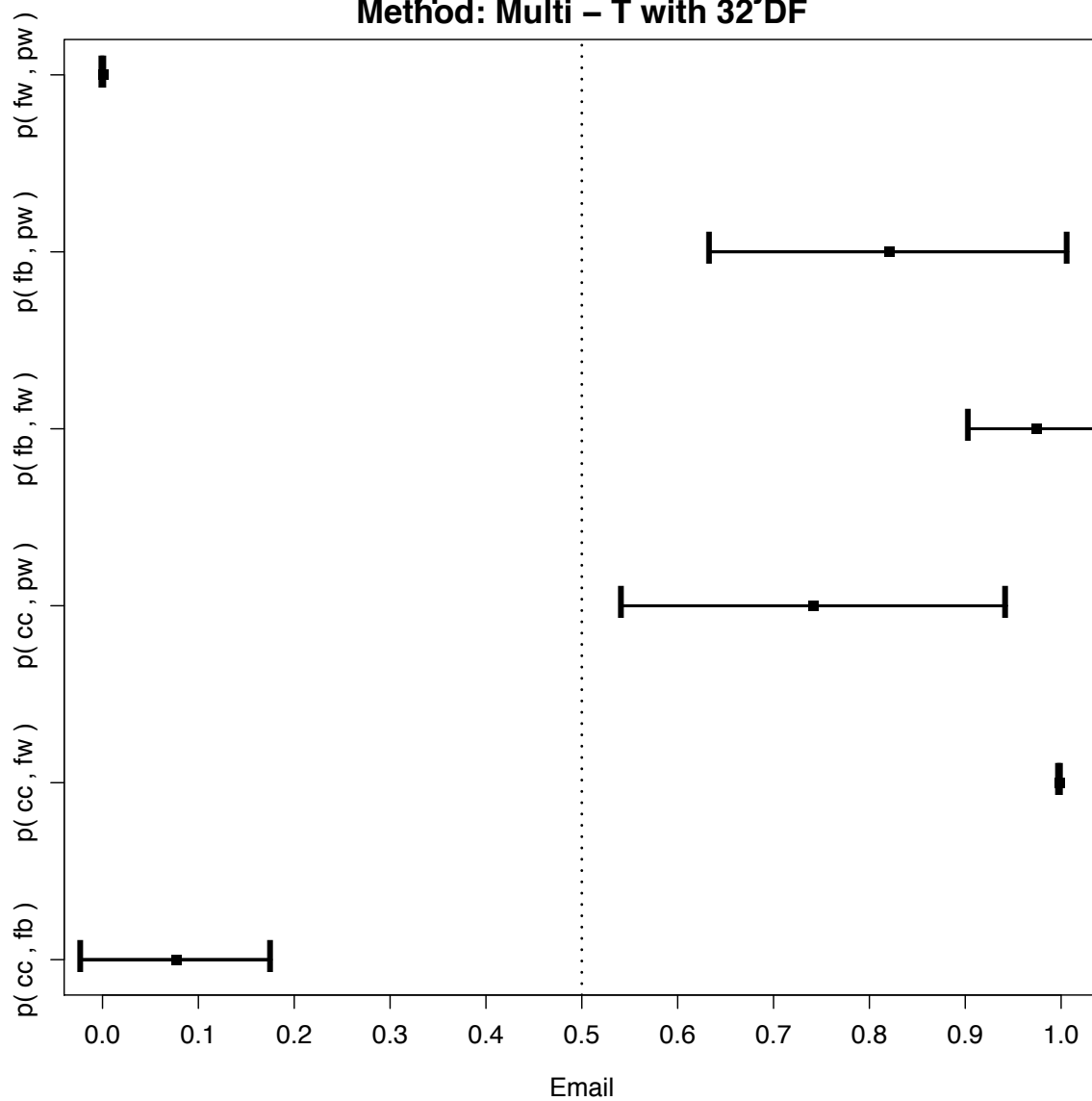


95 % Simultaneous Confidence Intervals  
Type of Contrast: Tukey  
Method: Multi - T with 73 DF

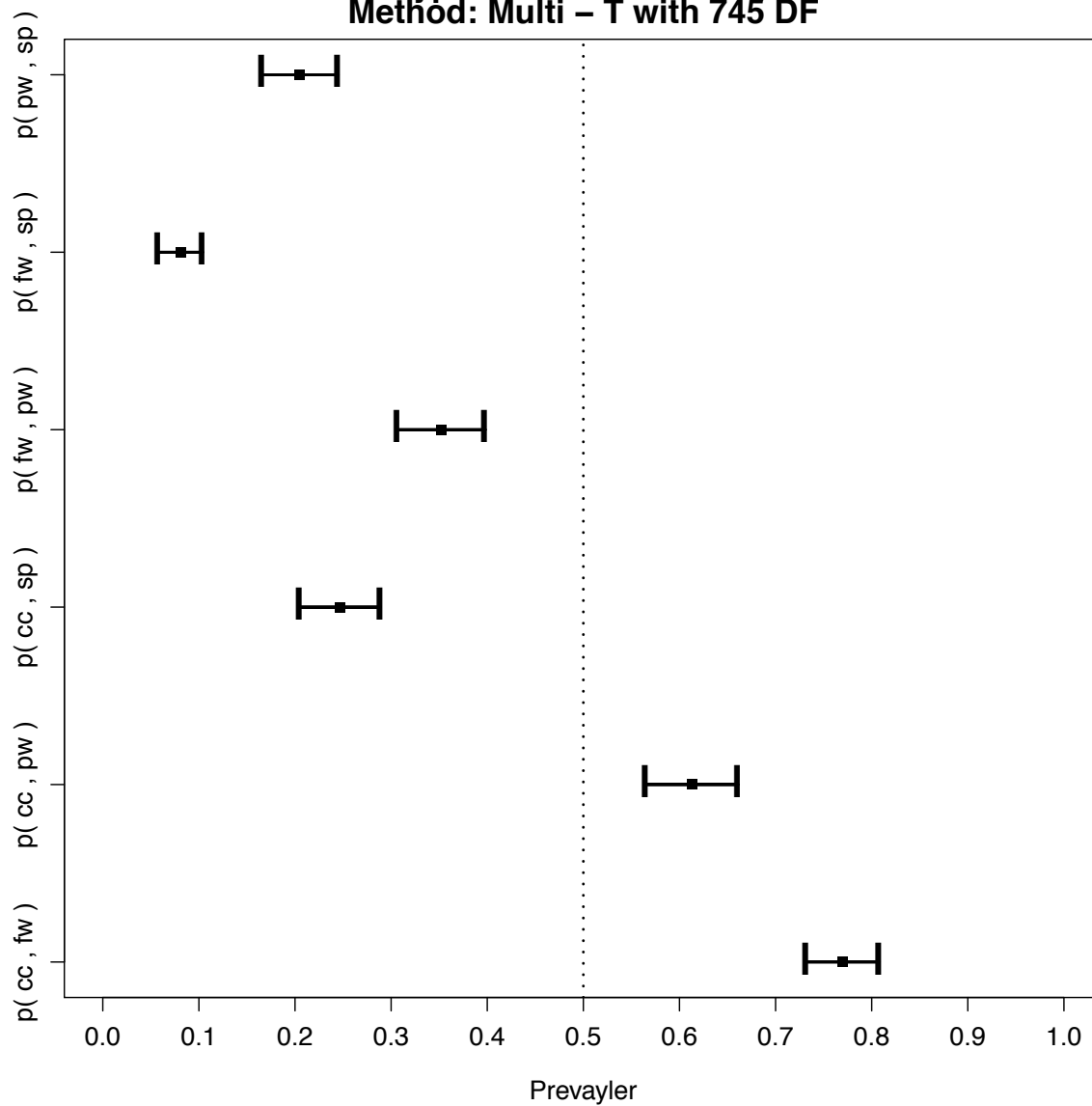


# 95 % Simultaneous Confidence Intervals

Type of Contrast: Tukey  
Method: Multi - T with 32 DF



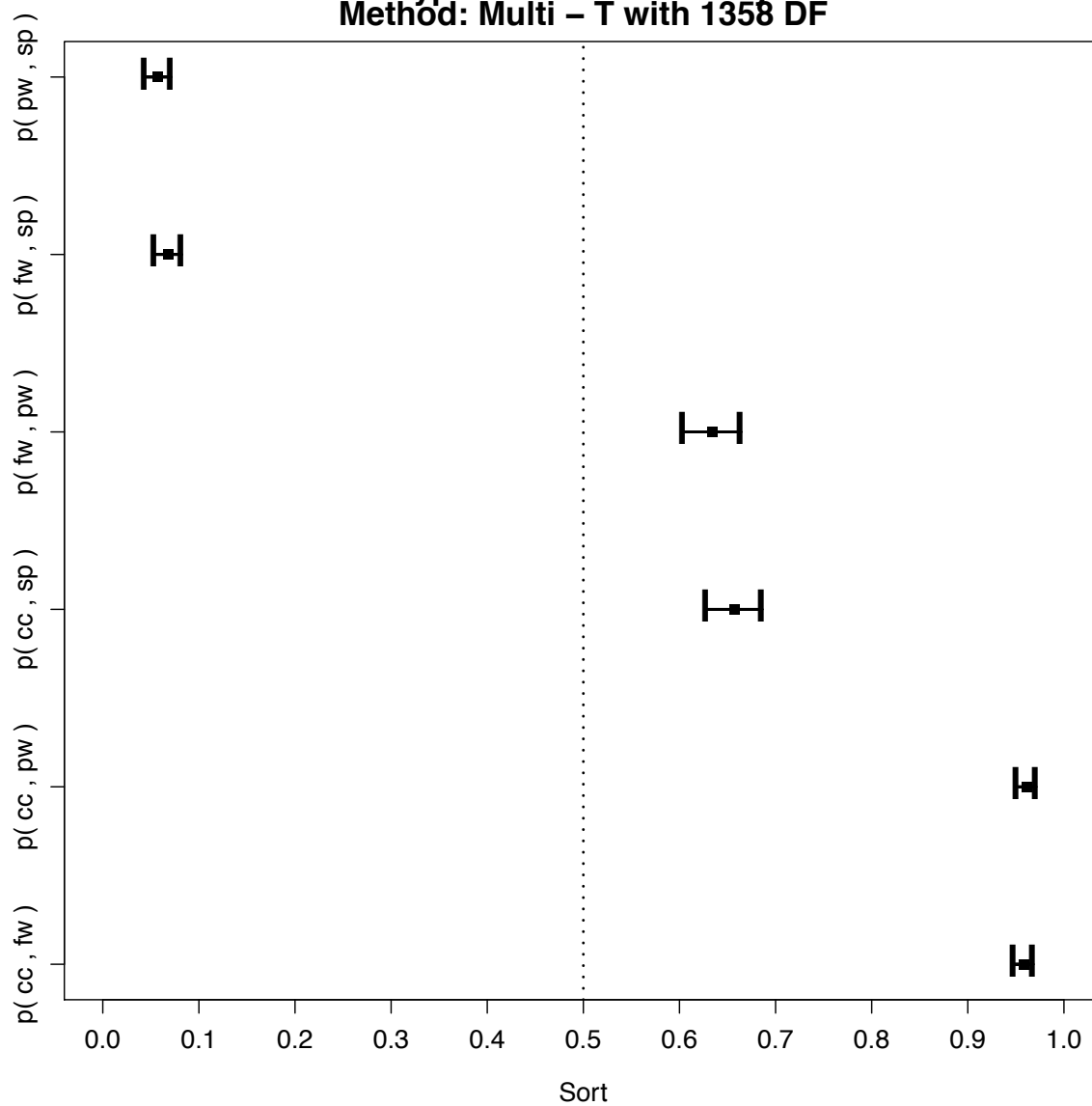
95 % Simultaneous Confidence Intervals  
Type of Contrast: Tukey  
Method: Multi - T with 745 DF



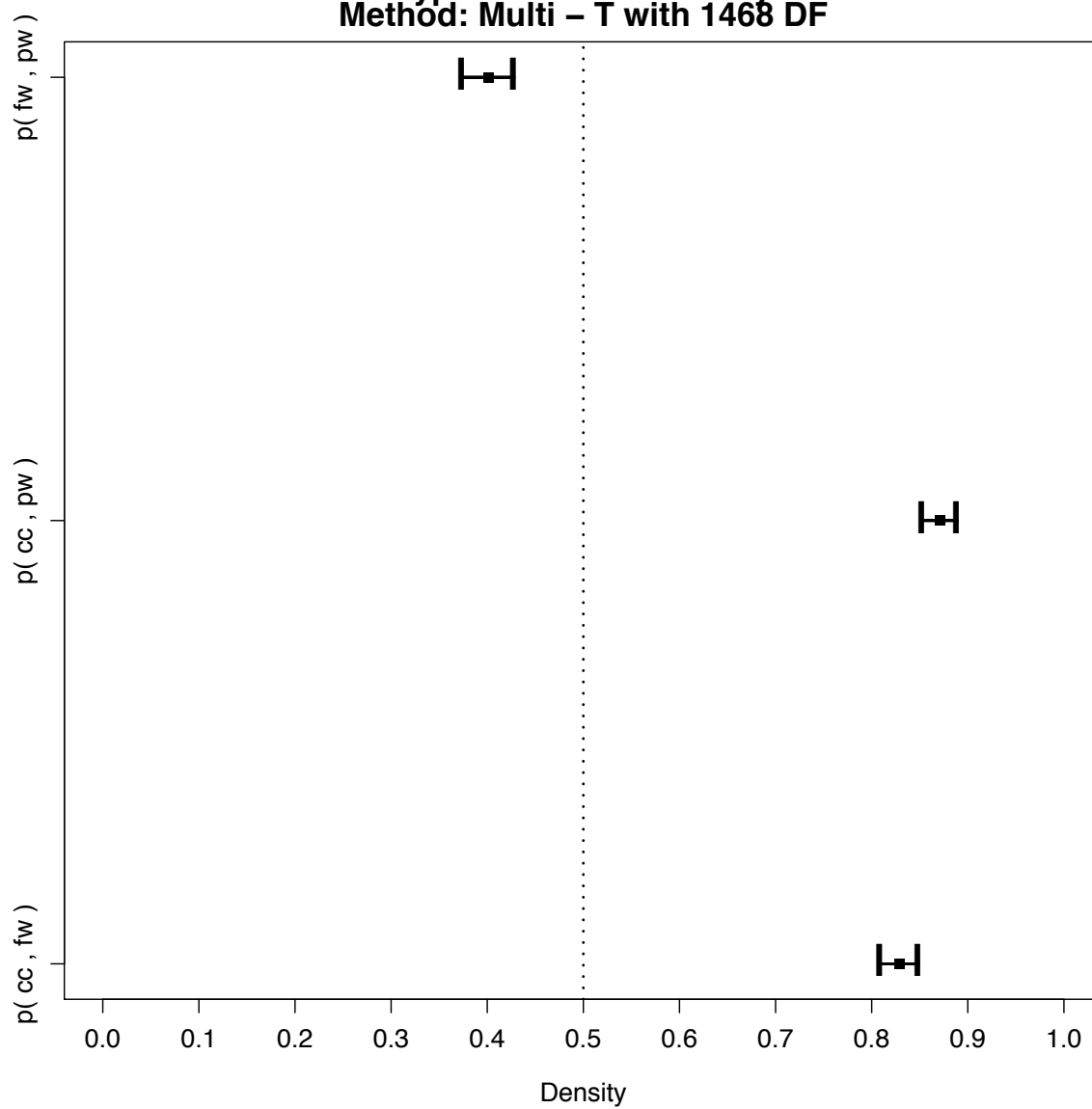
**95 % Simultaneous Confidence Intervals**

**Type of Contrast: Tukey**

**Method: Multi - T with 1358 DF**



95 % Simultaneous Confidence Intervals  
Type of Contrast: Tukey  
Method: Multi - T with 1468 DF



## Prediction error of entire configuration space

S	BF/SA	FW	PW	SAD	FB	CC
1	0.18	56.51	5.89	0.18	N/A	0.07
2	1.54	0.79	1.52	1.59	N/A	1.07
3	0.88	16.74	0.81	0.88	N/A	0.99
4	1.23	46.98	1.44	1.23	2.81	1.23
5	1.95	30.45	89.77	1.99	N/A	3.53
6	1.23	1.82	1.21	1.22	N/A	3.14
7	0.35	100	32.97	1.68	N/A	19.36
8	2.84	109.30	26.87	2.70	N/A	9.08
11	2.31	89.23	614.95	2.47	N/A	1.52
12	0.72	625.37	179.78	N/A	N/A	6.52



## Propagation algorithms

---

### Algorithm 2: Propagate influence down

---

**Input:**  $stmt, CFG, S \rightarrow \mathcal{P}(O)$   
**Output:**  $S \rightarrow \mathcal{P}(O)'$

```
1 Function propagate_down
2    $ipdom := ipdom(stmt, CFG)$   $\triangleright$  Get immediate post-dominator
    $\triangleright$  Get set of statements in all paths
3    $pstmts := paths\_stmts(stmt, ipdom) - ipdom$ 
4   for each  $ps \in pstmts$  do
5      $\triangleright influence(ps): S \rightarrow \mathcal{P}(O)$ 
6     if  $influence(ps) \supset influence(stmt)$  then
7        $influence(ps) := influence(stmt)$ 
8     end
9 end
```

---

---

### Algorithm 3: Propagate regions up

---

**Input:**  $stmt, CFG, S \rightarrow \mathcal{P}(O), GI : \mathcal{P}(O)$   
**Output:**  $S \rightarrow \mathcal{P}(O)'$

```
1 Function propagate_up
2   for each  $pred \in preds(stmt, CFG)$  do
3      $\triangleright influence(s): S \rightarrow \mathcal{P}(O)$ 
4     if  $influence(pred) \cup influence(stmt) \in GI \wedge influence(pred) \neq influence(pred) \cup influence(stmt)$ 
5       then
6          $influence(pred) := influence(stmt)$ 
7     end
8 end
```

---