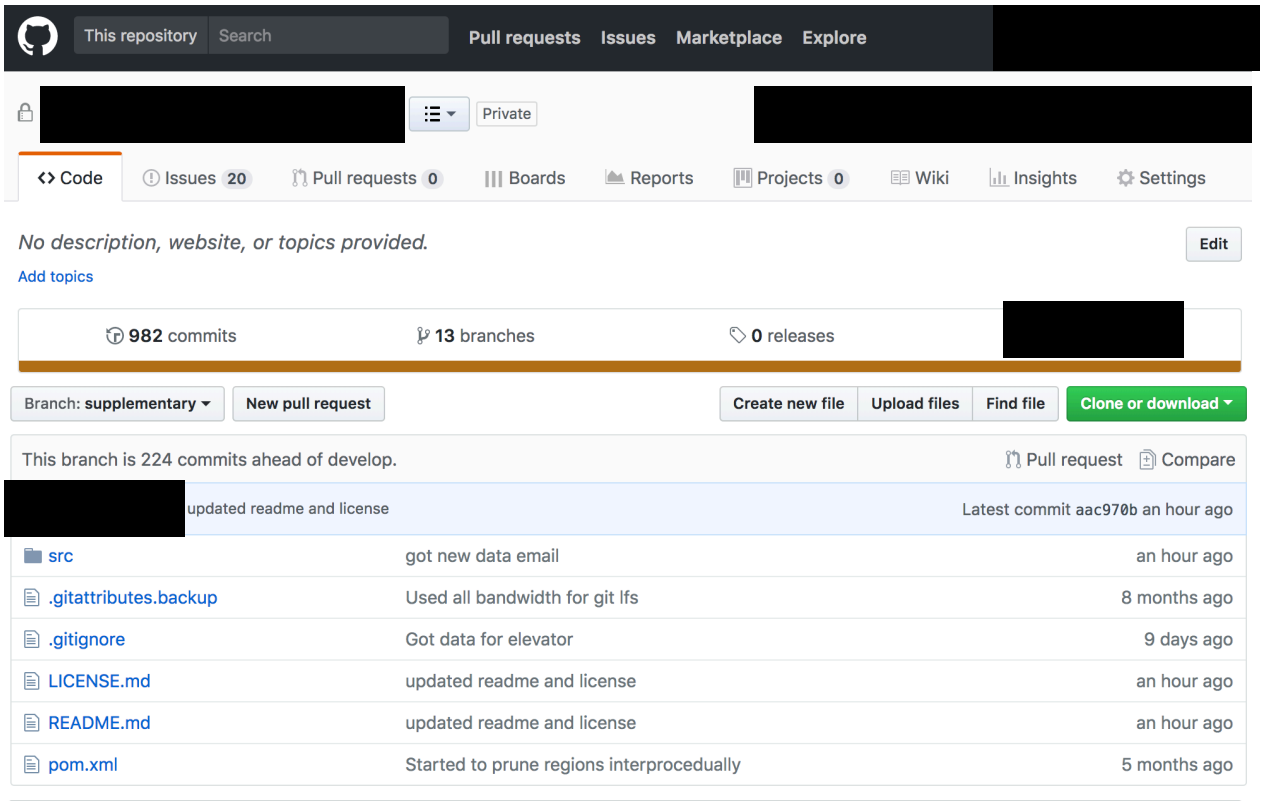
ConfigCrusher: White-Box Performance Analysis for Configurable Systems (Supplementary material)

Main repo



ConfigCrusher: 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 modern configurable systems, we are often interested in knowing how configuration options influence the performance of the system. Several approaches exist to obtain this information, but they usually require a large number of samples to make accurate predictions, and some impose limitations on the systems that they can analyze. This paper proposes ConfigCrusher, a new white-box performance analysis approach for configurable systems. ConfigCrusher employs a static taint analysis to identify how configuration options may influence control-flow decisions (considering control-flow and data-flow dependencies) and instruments code regions corresponding to these decisions to dynamically analyze the influence of options on the regions' performance. Our evaluation using 10 real-world configurable systems shows that ConfigCrusher is more efficient at building performance models that are similar to or more accurate than current state-of-the-art black-box and white-box approaches. Overall, this paper showcases the benefits and potential of our white-box performance analysis to outperform other approaches.

Supplementary material

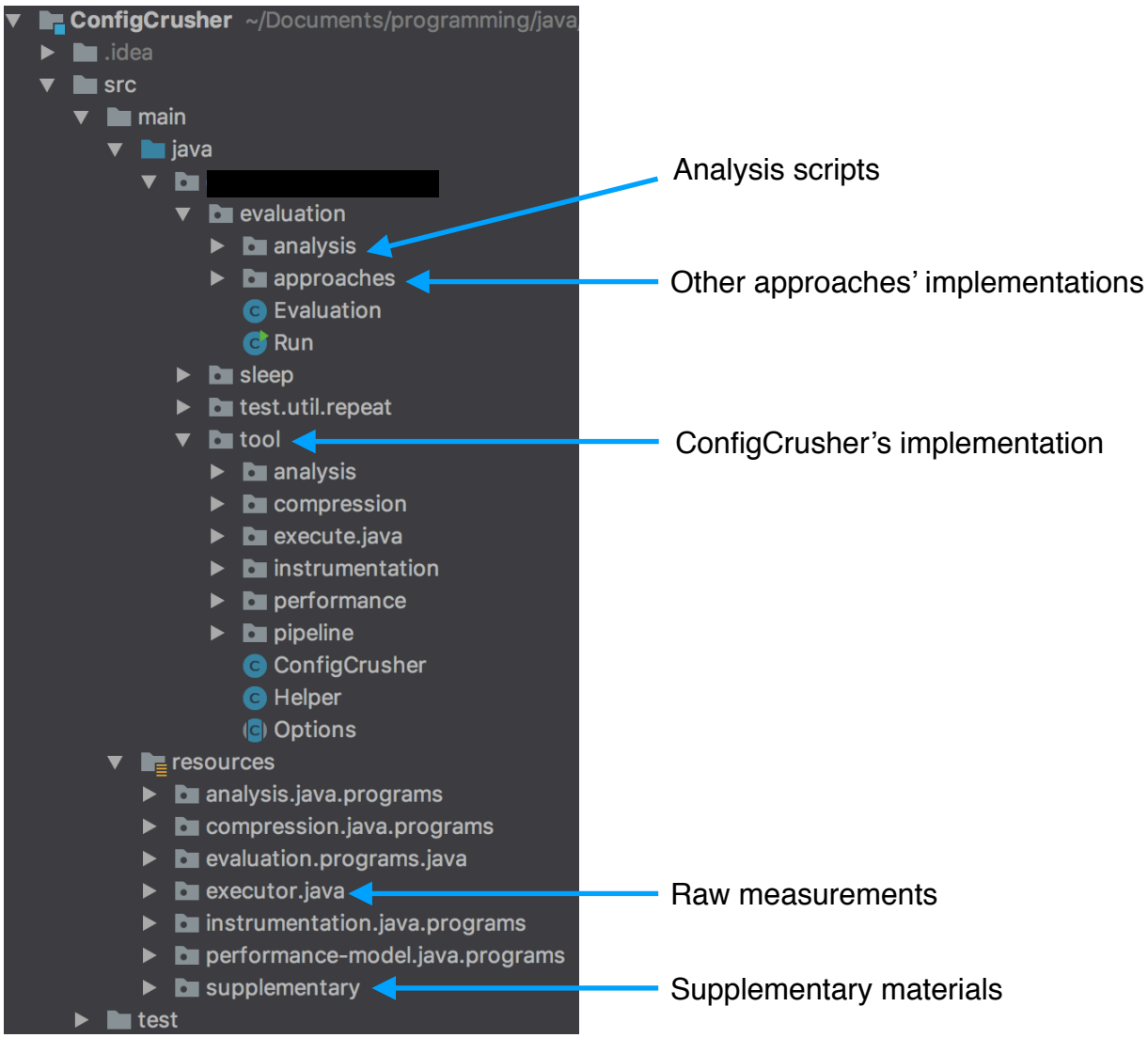
Link

License

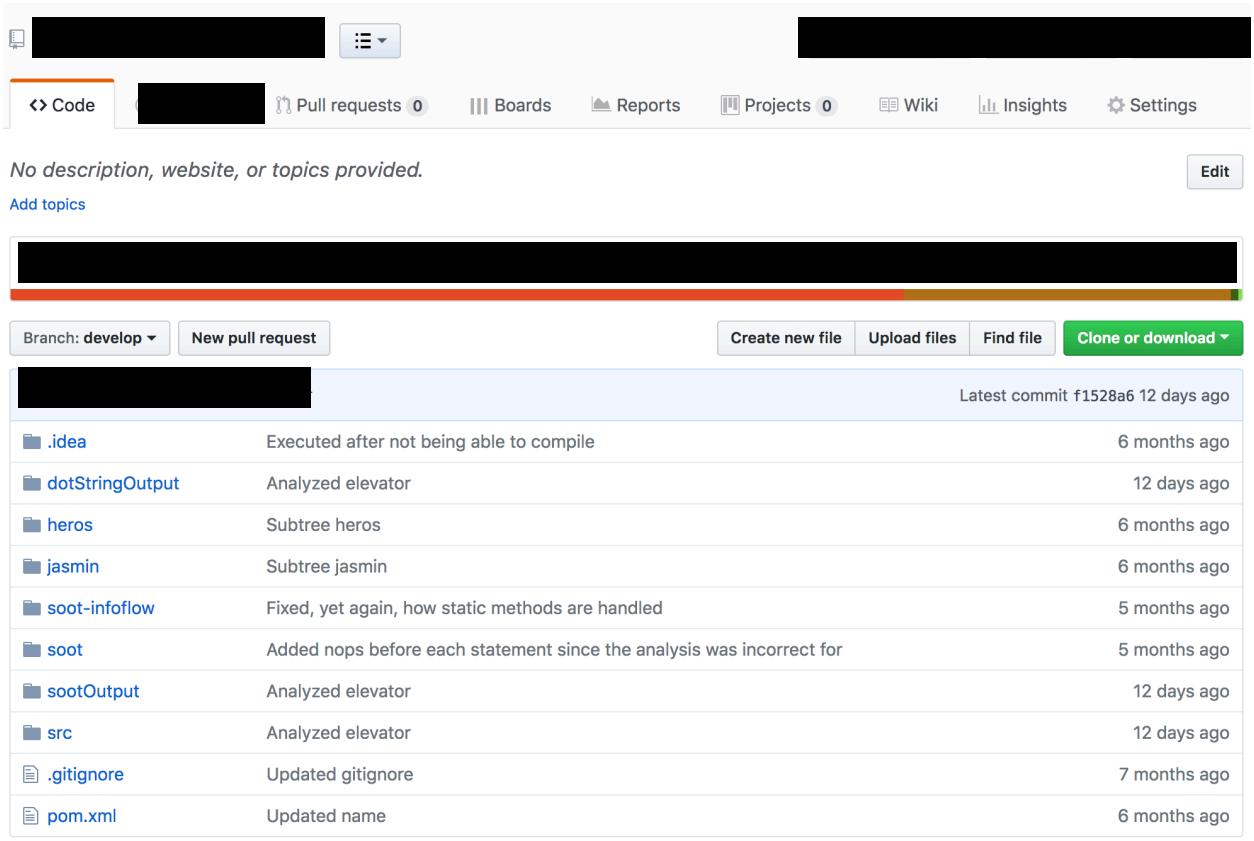
MIT License

Copyright (c) 2018 [redacted]

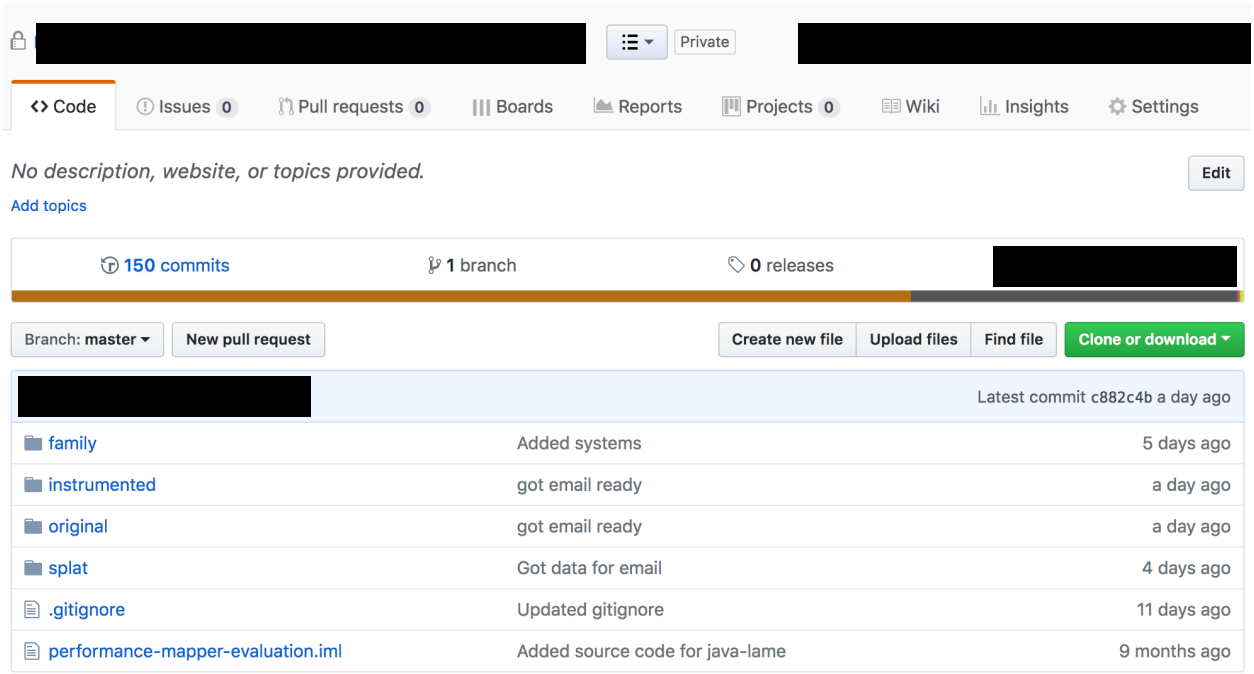
Structure



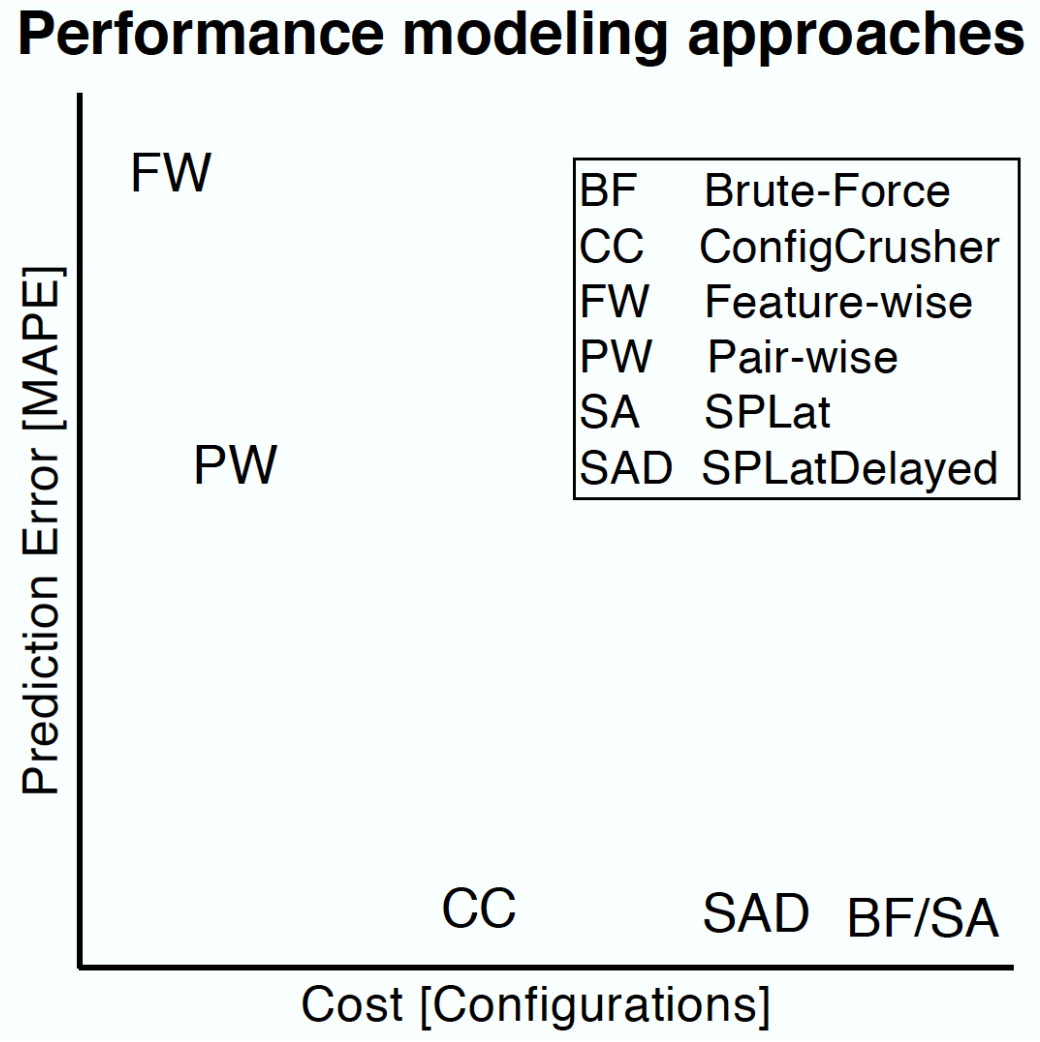
Static taint analysis repo



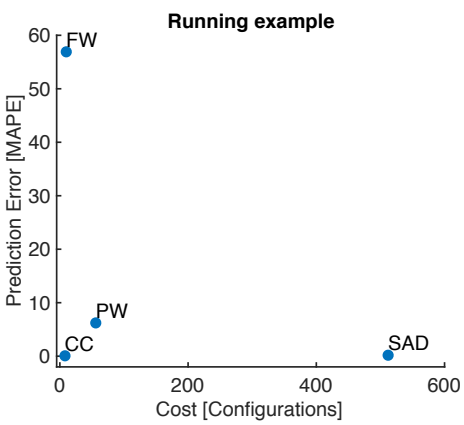
Subject systems repo

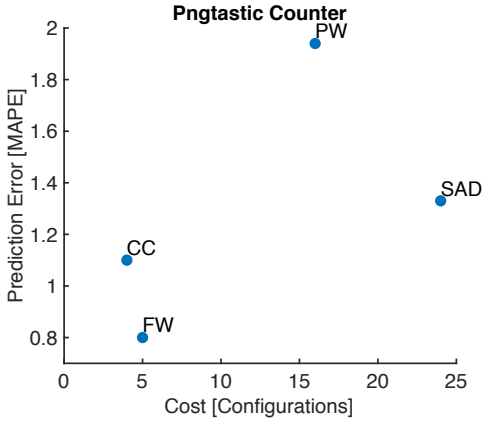


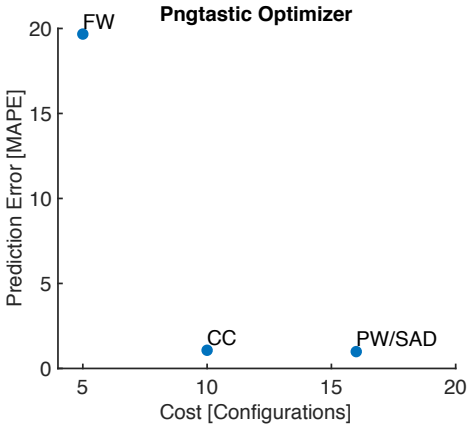
Conjecture on cost and prediction error comparison between state-of-the-art approaches

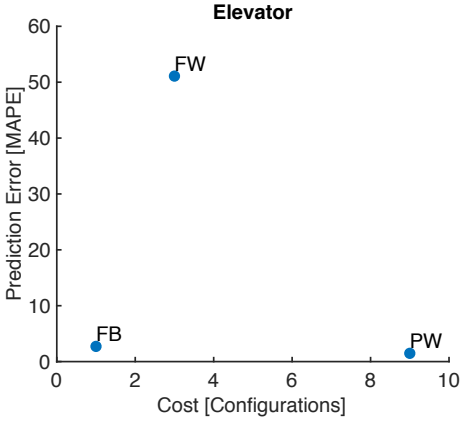


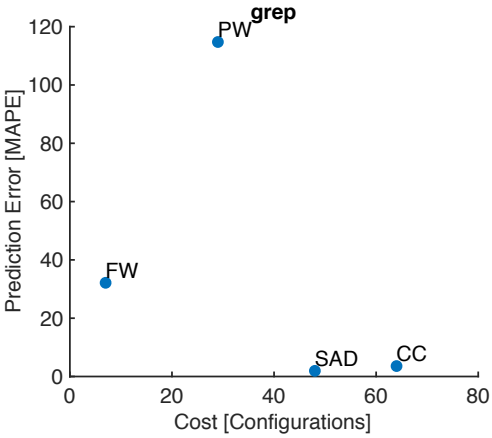
Cost vs prediction error

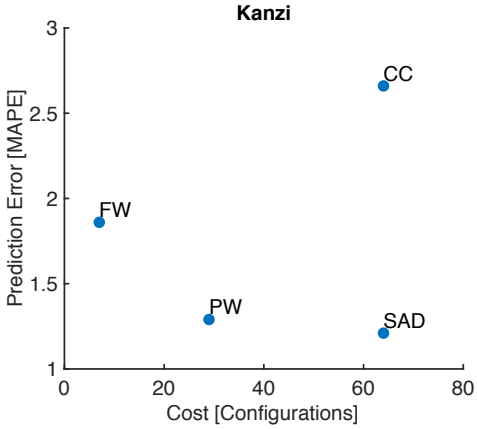


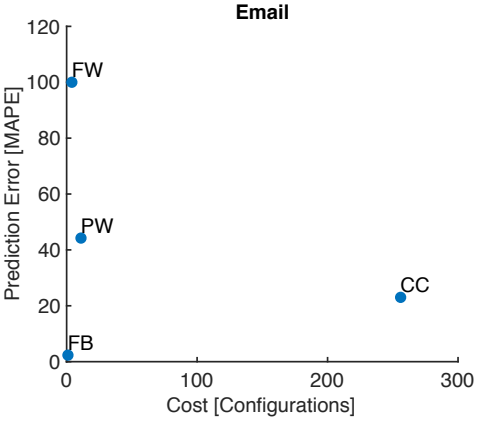


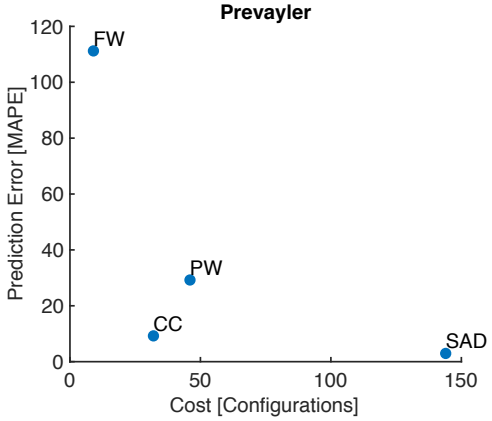


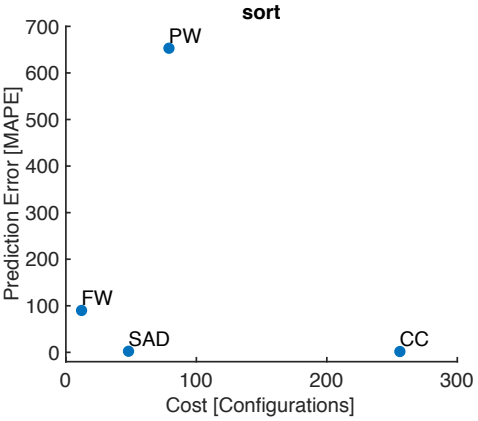


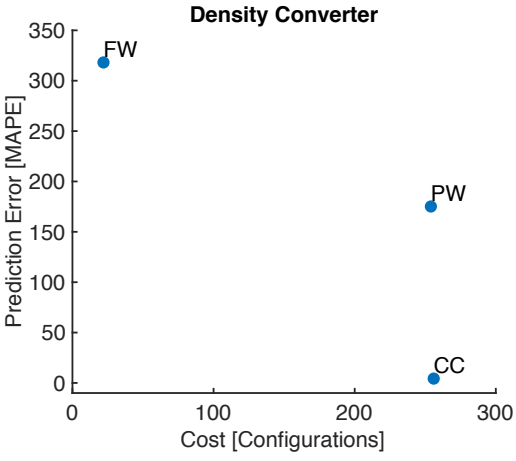




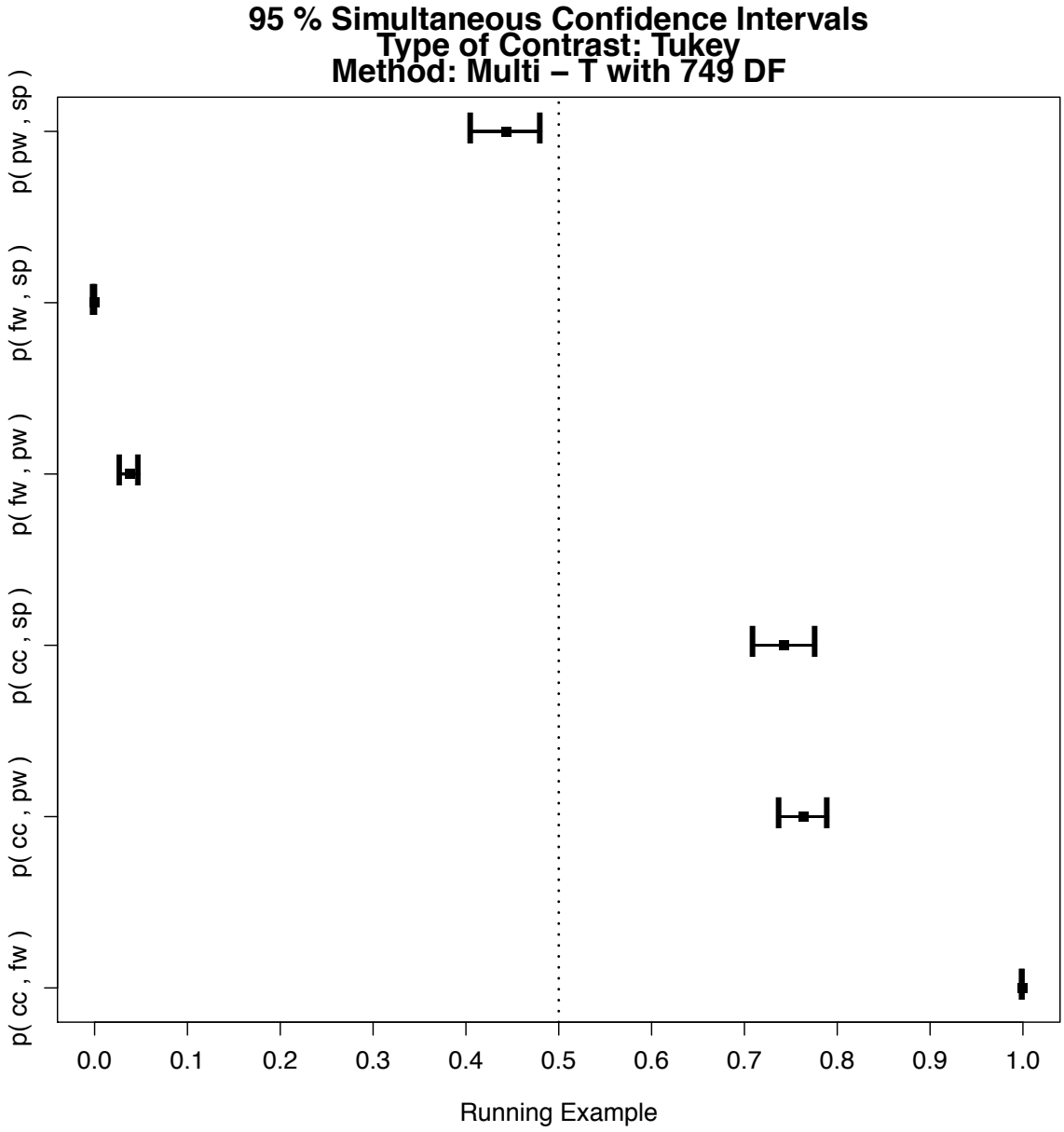


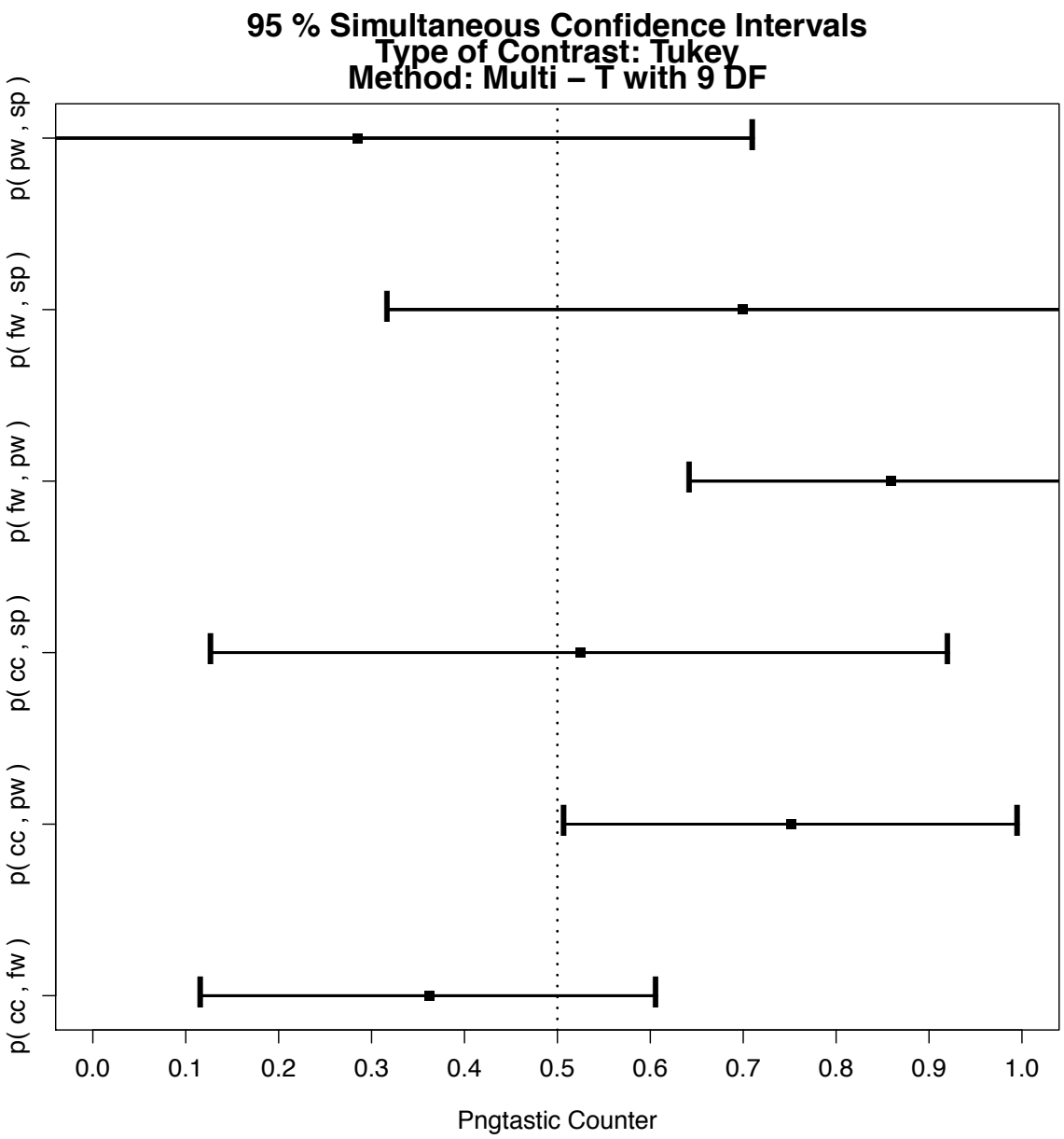


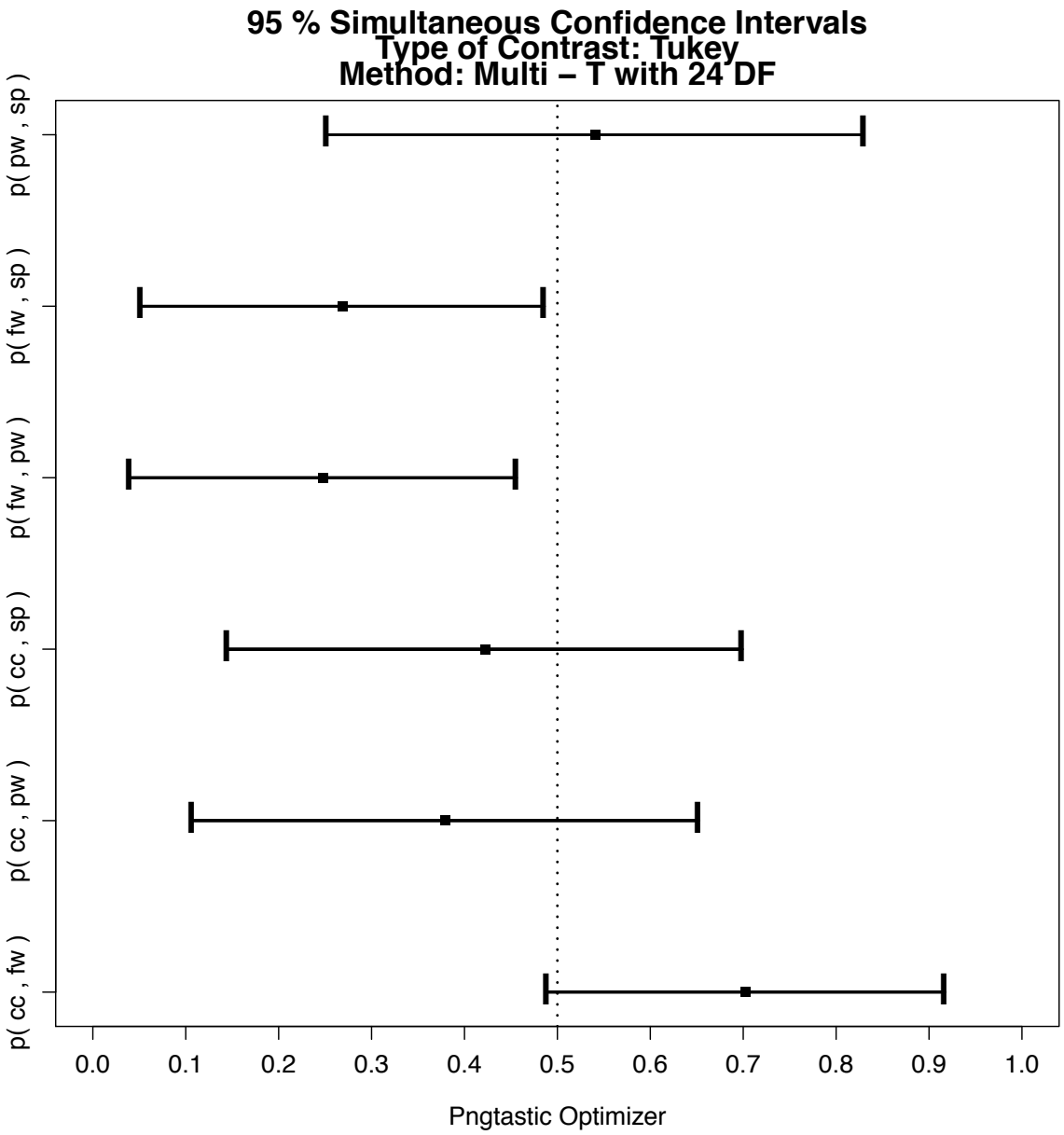


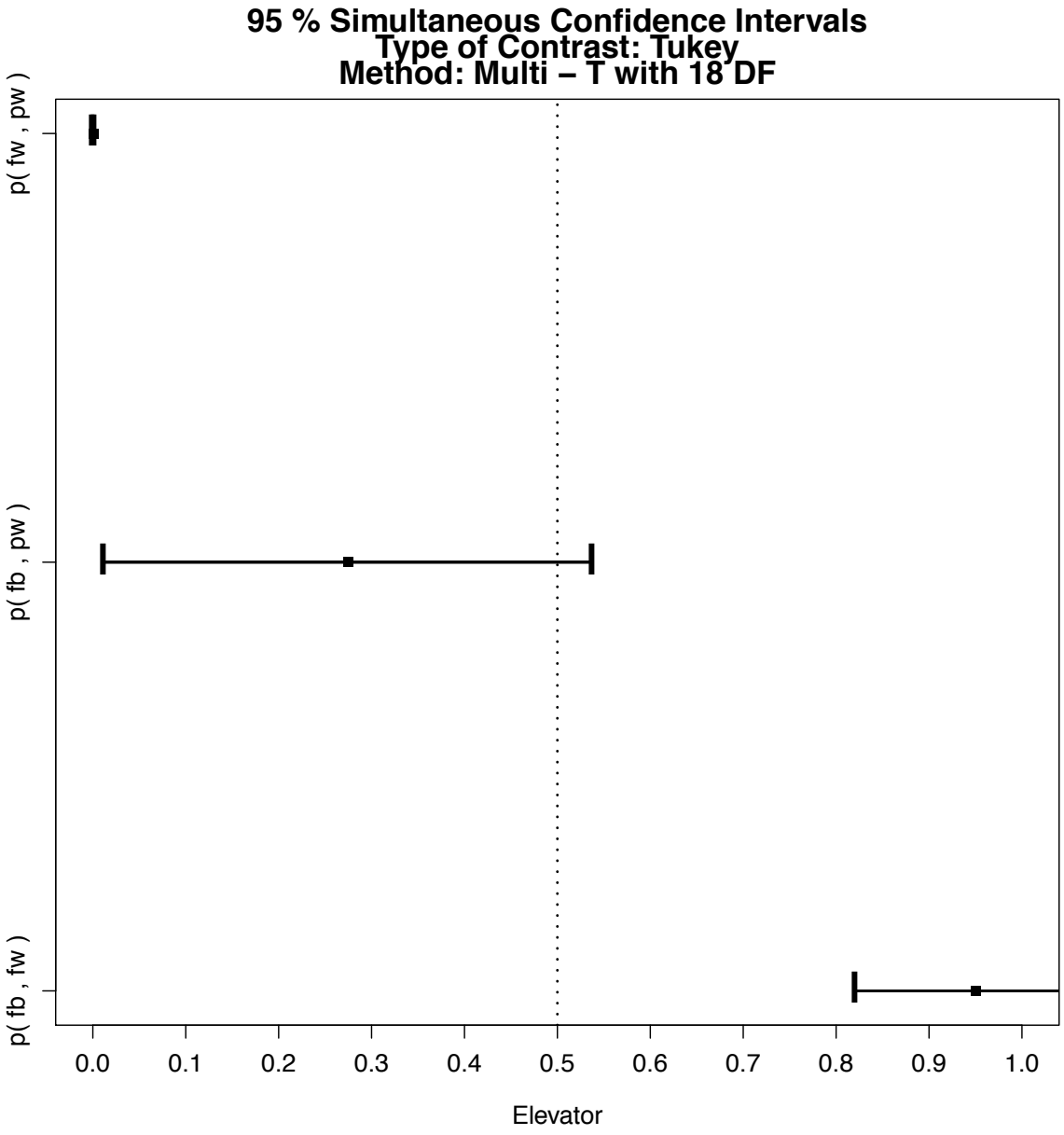


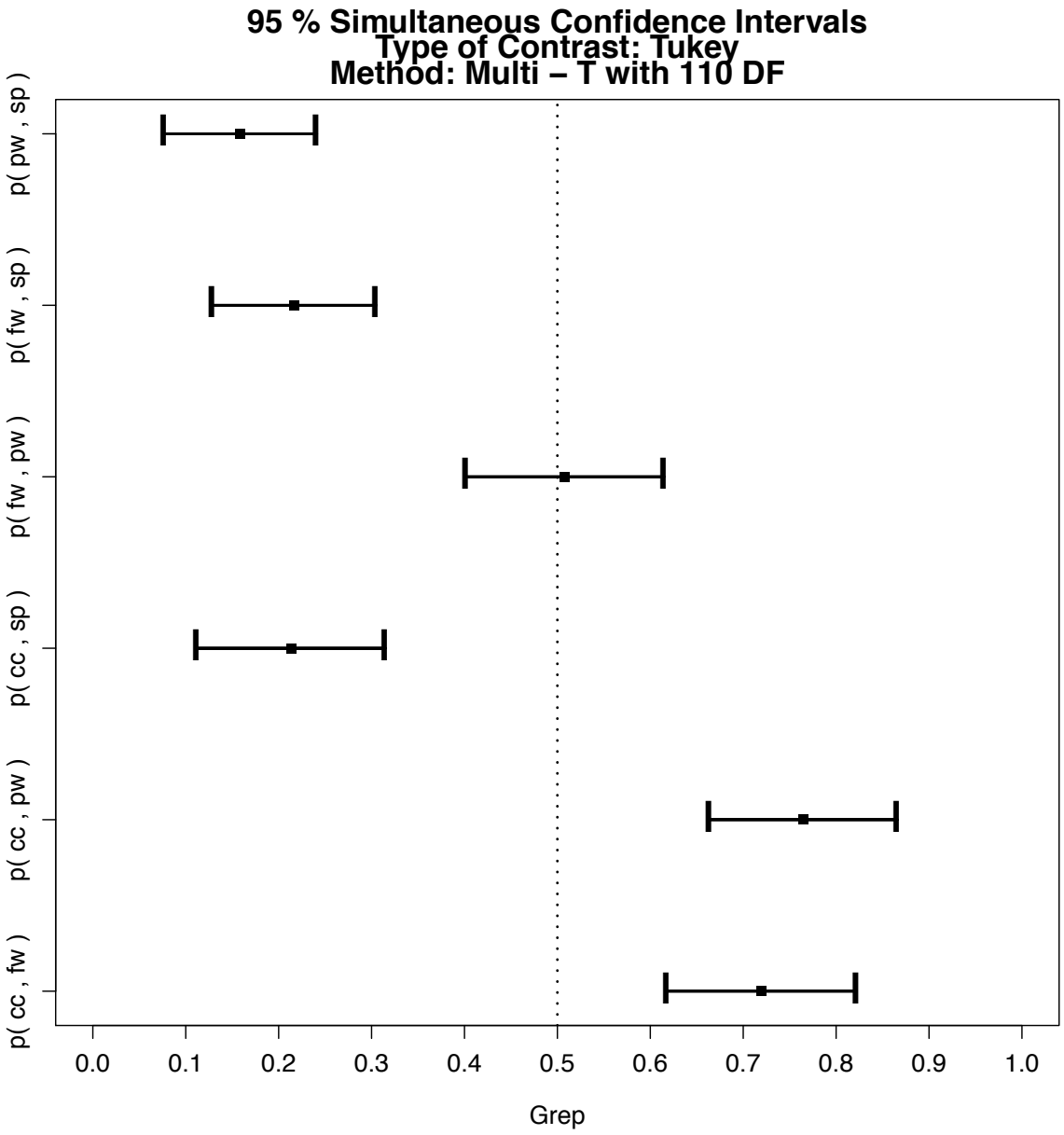
T-procedure error comparison *(Interpretation p(a,b) > 1/2 : b tends to be larger than a)*

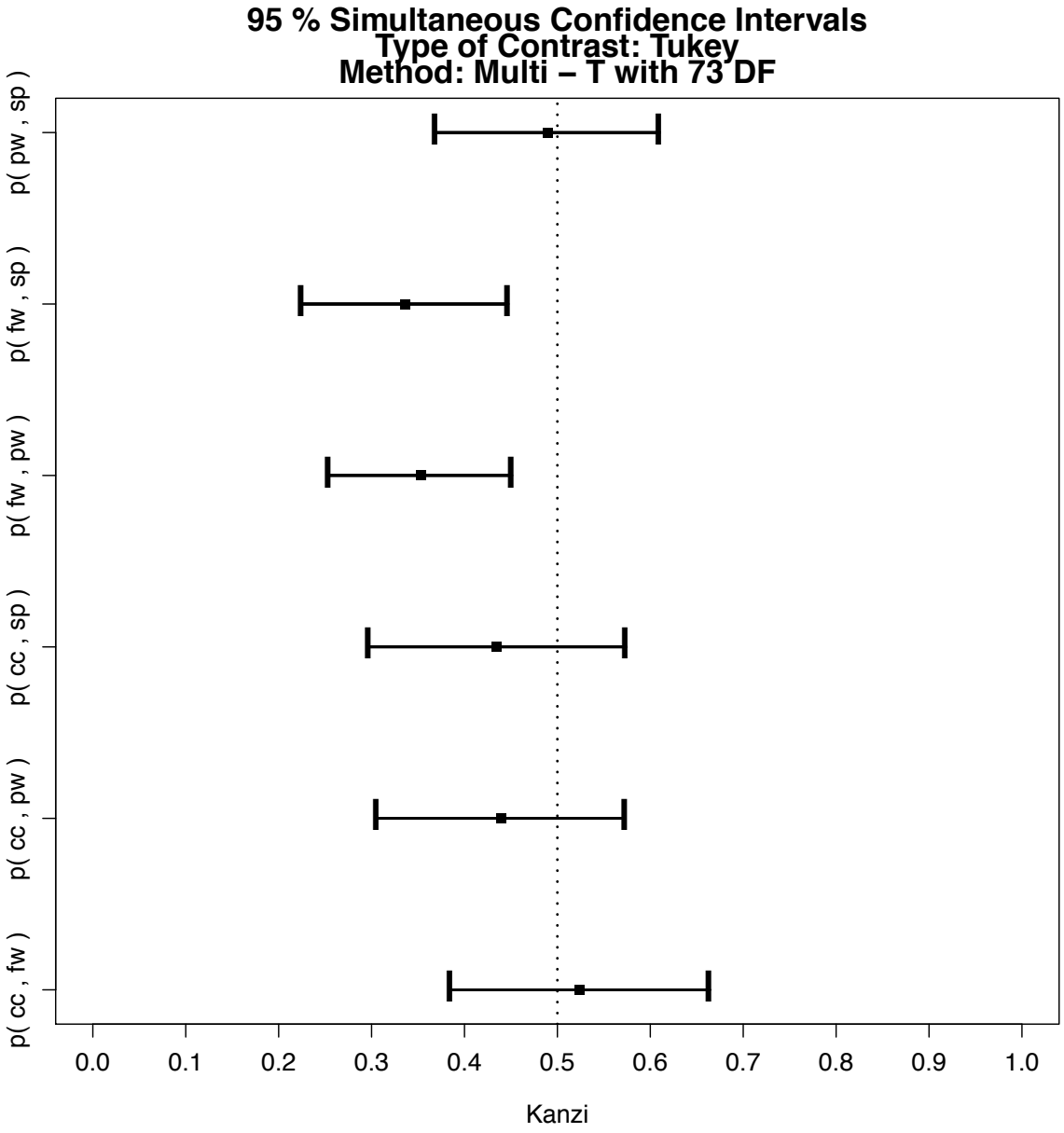


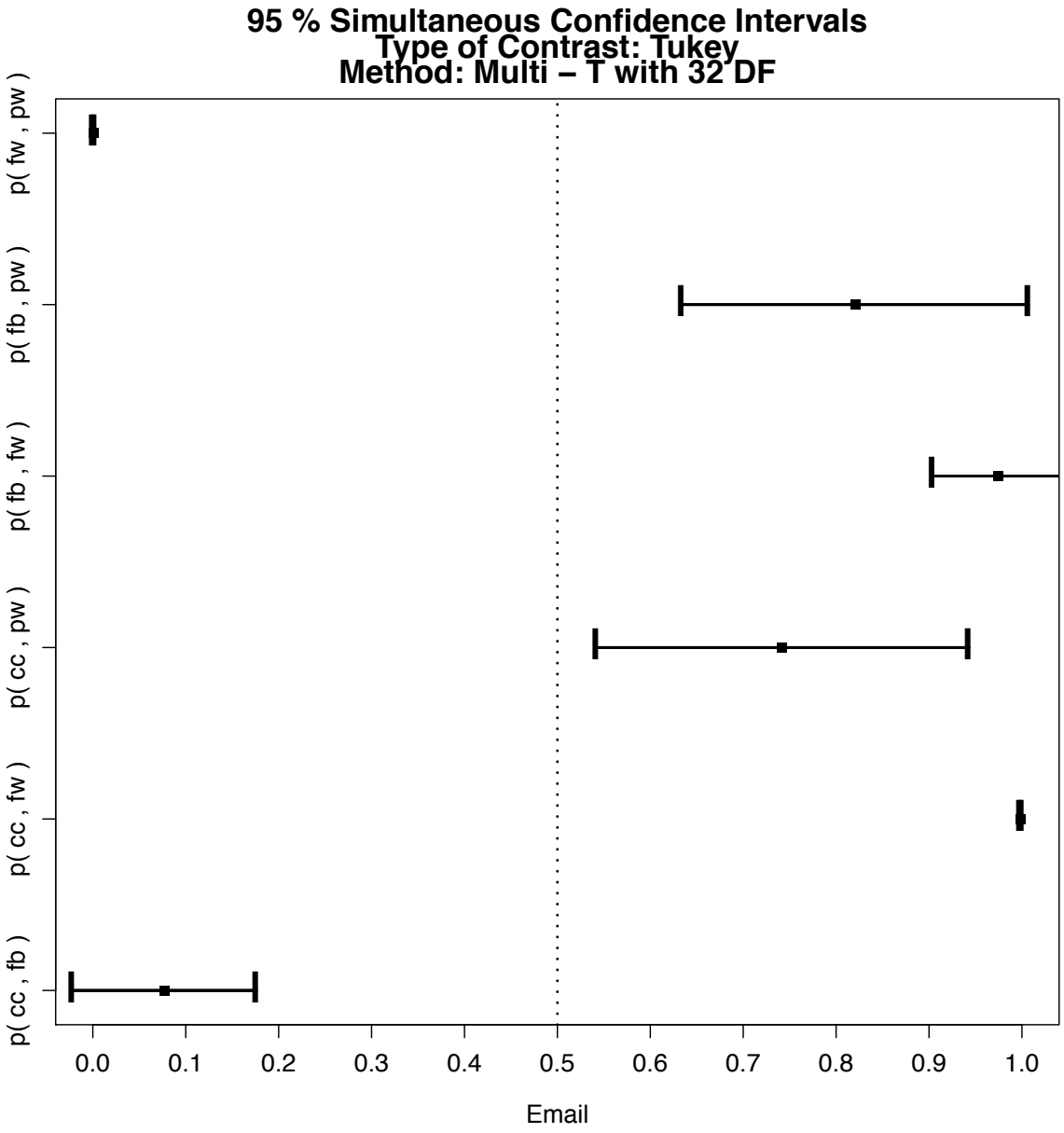

95 % Simultaneous Confidence Intervals
Type of Contrast: Tukey
Method: Multi – T with 9 DF
p(pw , sp)
p(fw , sp)
p(fw , pw)
p(cc , sp)
p(cc , pw)
p(cc , fw)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Pngtastic Counter

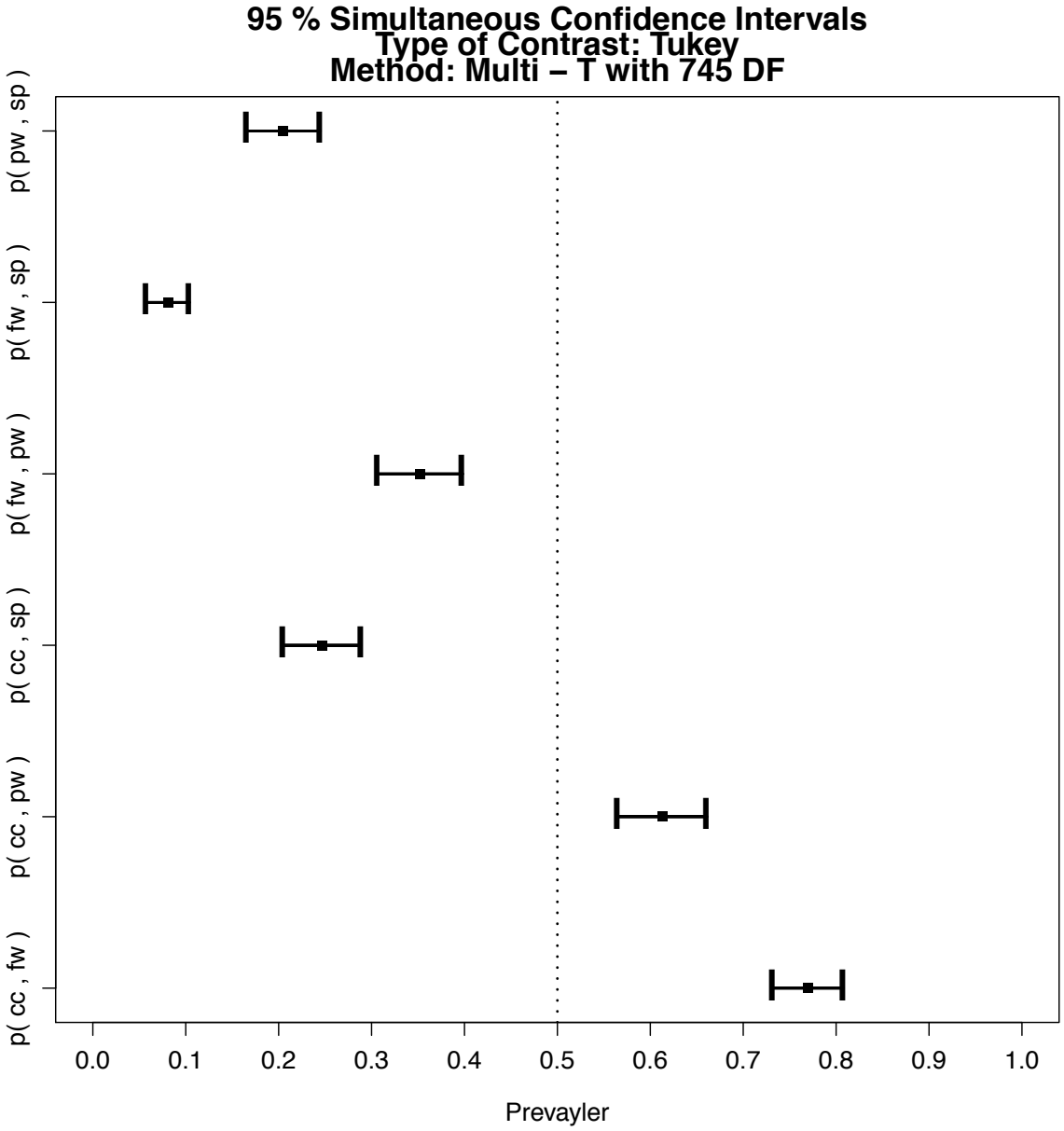

95 % Simultaneous Confidence Intervals
Type of Contrast: Tukey
Method: Multi – T with 24 DF
p(pw , sp)
p(fw , sp)
p(fw , pw)
p(cc , sp)
p(cc , pw)
p(cc , fw)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Pngtastic Optimizer

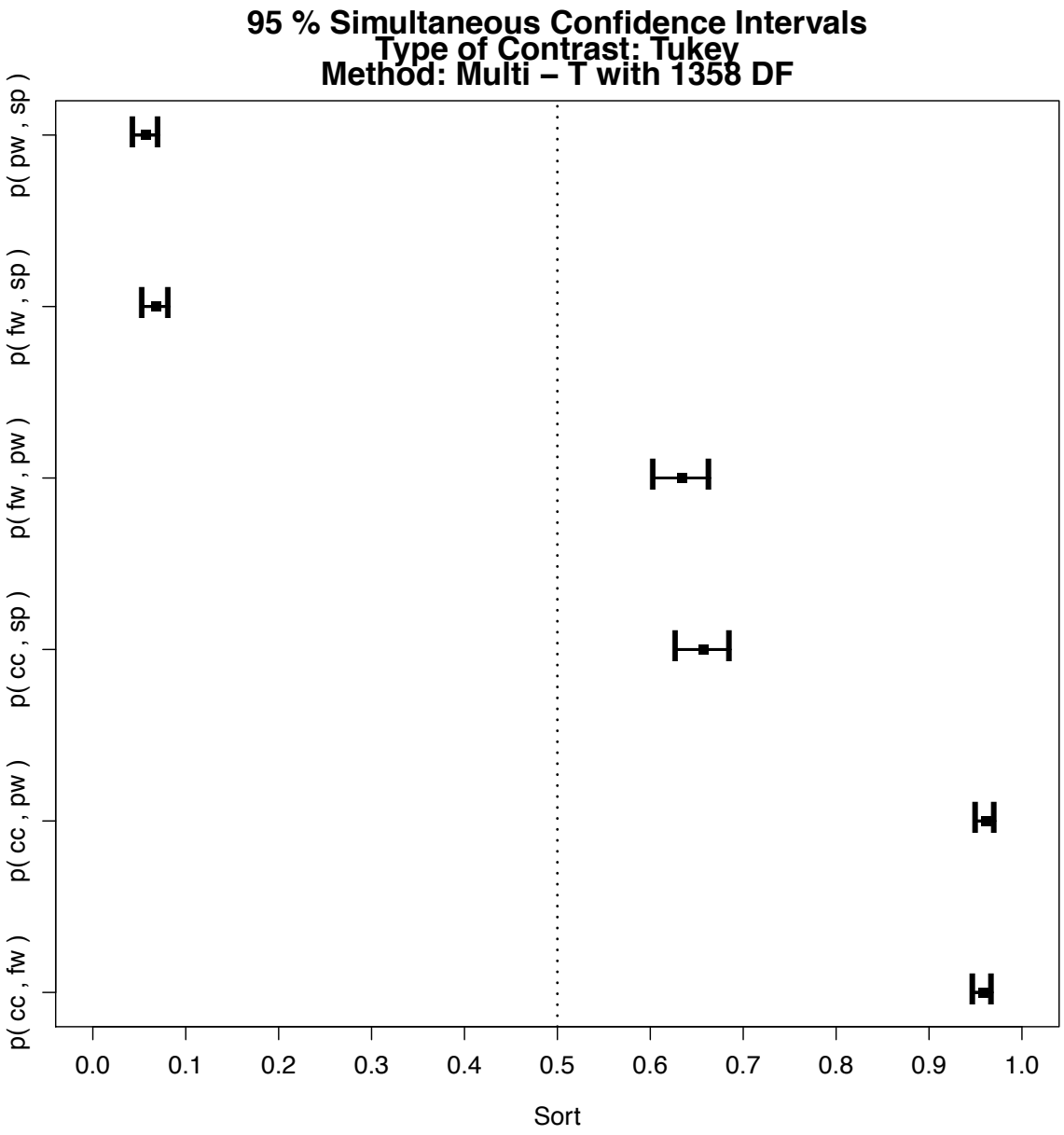

95 % Simultaneous Confidence Intervals
Type of Contrast: Tukey
Method: Multi – T with 18 DF
p(fw , pw)
p(fb , pw)
p(fb , fw)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Elevator

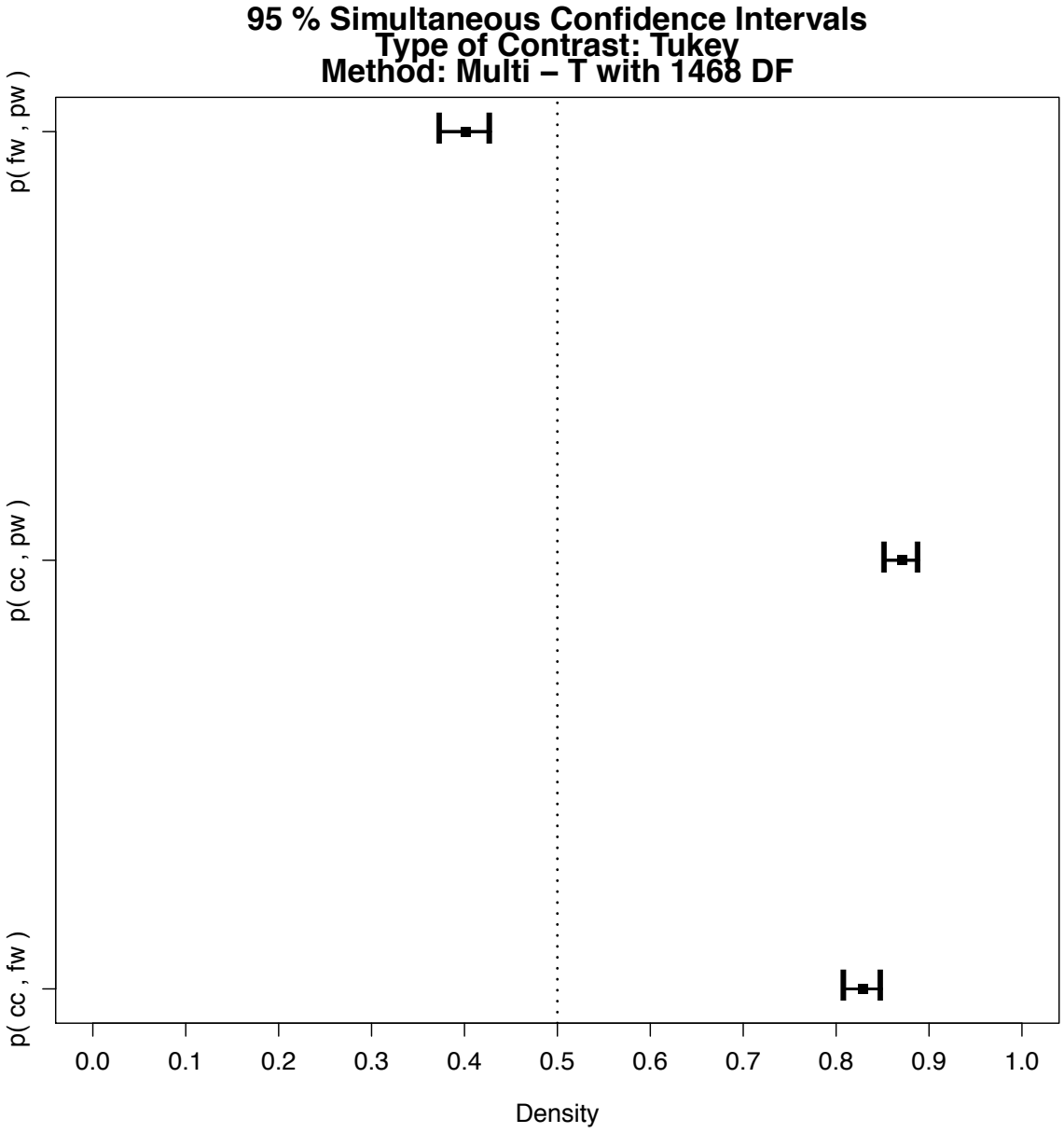

95 % Simultaneous Confidence Intervals
Type of Contrast: Tukey
Method: Multi – T with 110 DF
p(pw , sp)
p(fw , sp)
p(fw , pw)
p(cc , sp)
p(cc , pw)
p(cc , fw)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Grep


95 % Simultaneous Confidence Intervals
Type of Contrast: Tukey
Method: Multi – T with 73 DF
p(pw , sp)
p(fw , sp)
p(fw , pw)
p(cc , sp)
p(cc , pw)
p(cc , fw)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Kanzi


95 % Simultaneous Confidence Intervals
Type of Contrast: Tukey
Method: Multi – T with 32 DF
p(fw , pw)
p(fb , pw)
p(fb , fw)
p(cc , pw)
p(cc , fw)
p(cc , fb)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Email


95 % Simultaneous Confidence Intervals
Type of Contrast: Tukey
Method: Multi – T with 745 DF
p(pw , sp)
p(fw , sp)
p(fw , pw)
p(cc , sp)
p(cc , pw)
p(cc , fw)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Prevayler


95 % Simultaneous Confidence Intervals
Type of Contrast: Tukey
Method: Multi – T with 1358 DF
p(pw , sp)
p(fw , sp)
p(fw , pw)
p(cc , sp)
p(cc , pw)
p(cc , fw)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Sort


95 % Simultaneous Confidence Intervals
Type of Contrast: Tukey
Method: Multi – T with 1468 DF
p(fw , pw)
p(cc , pw)
p(cc , fw)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Density

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

```
1 Function propagate_down
2     ipdom := ipdom(stmt, CFG) ▷ Get immediate post-dominator
      ▷ Get set of statements in all paths
3     pstmts := paths_stmts(stmt, ipdom) − ipdom
4     for each ps ∈ pstmts do
          ▷ influence(ps): S → P(O)
5         if influence(ps) ⊃ influence(stmt) then
6             influence(ps) := influence(stmt)
7         end
8     end
9 end
```

Algorithm 3: Propagate regions up

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

```
1 Function propagate_up
2     for each pred ∈ preds(stmt, CFG) do
          ▷ influence(s): S → P(O)
3         if influence(pred) ∪ influence(stmt) ∈ GI ∧ influence(pred) ≠ influence(pred) ∪ influence(stmt)
           then
4             influence(pred) := influence(stmt)
5         end
6     end
7 end
```