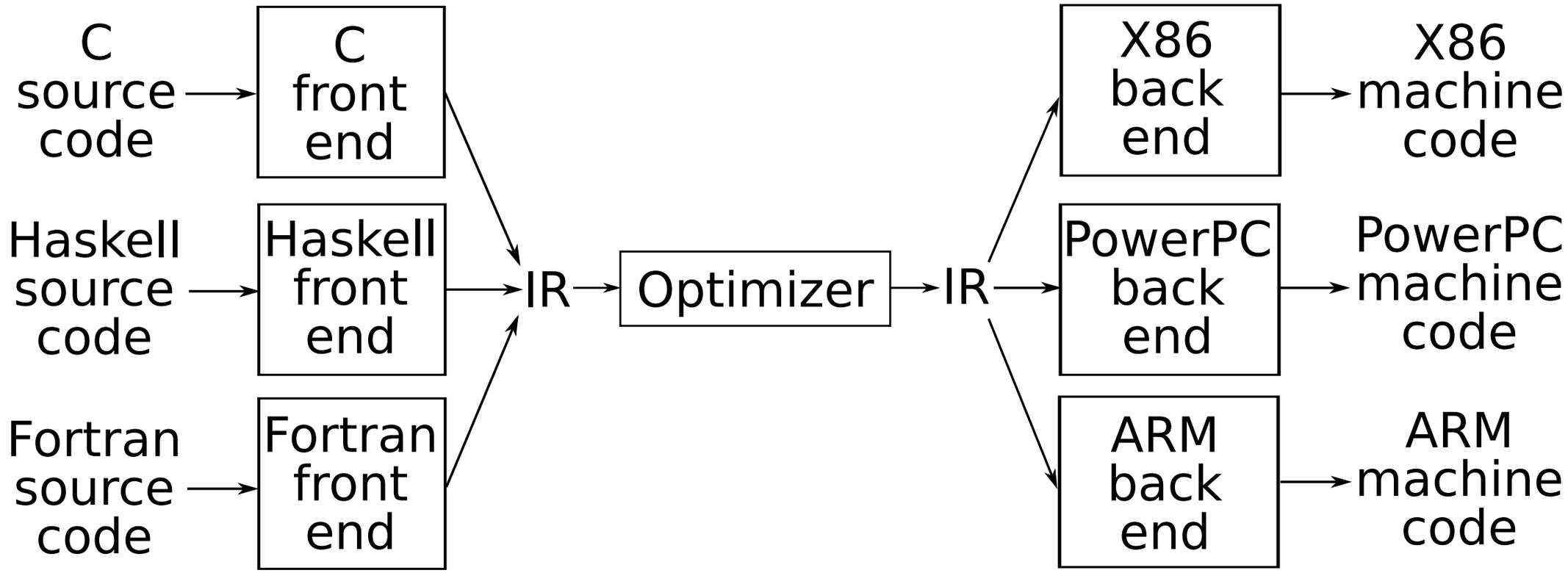
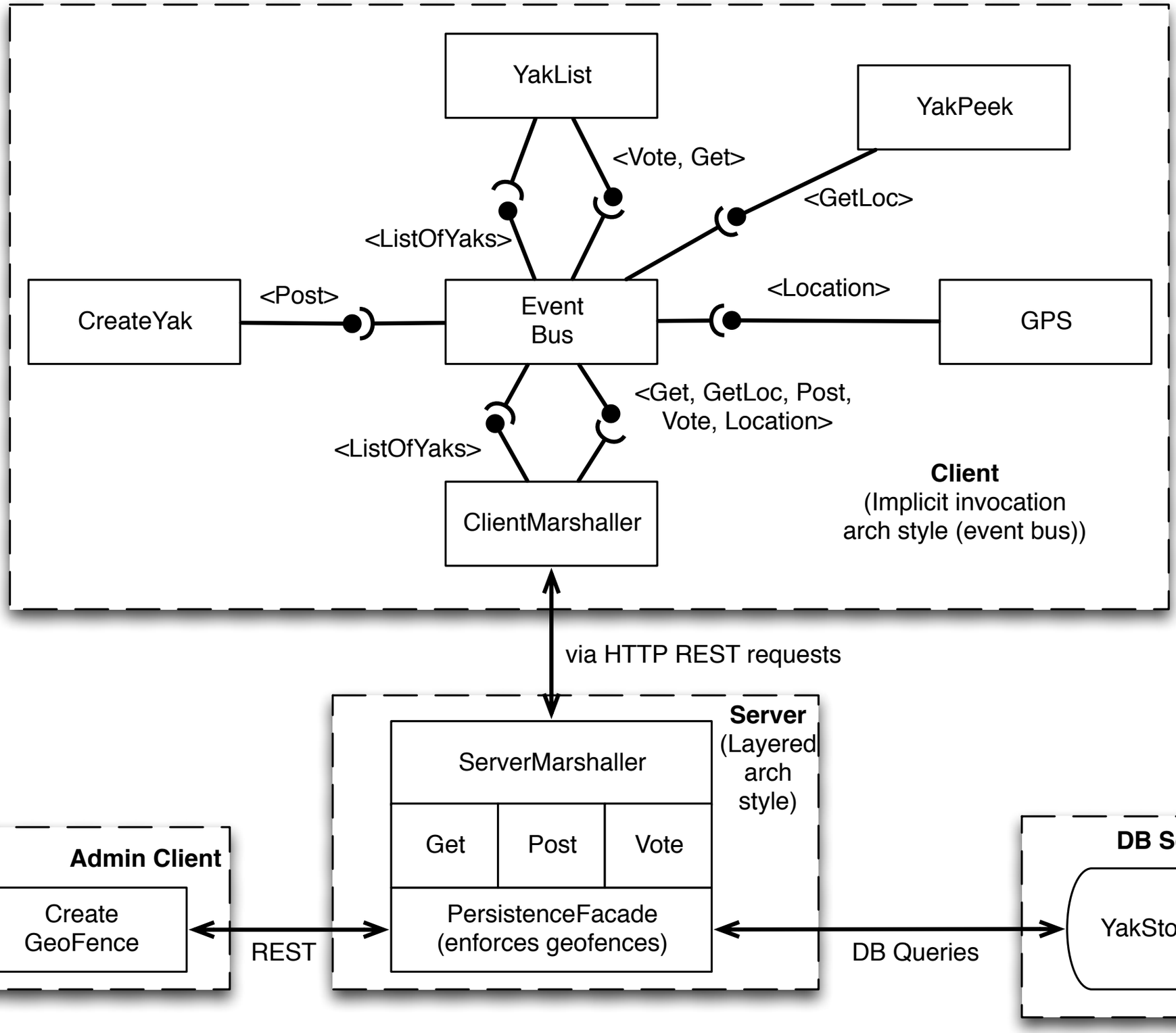
Coverage Metrics

Reid Holmes

Testability wrt LLVM



Testability wrt YikYak



Key questions

Is a test suite:

Sufficiently broad?

Sufficiently deep?

Test suite breadth

Code coverage

Code:

```
int eval(int x,
                boolean c1,
                boolean c2){
   if (c1)
      x++;
   if (c2)
      x--;
   return x;
}
```

Code:

Test:

```
eval(0,false,false);
```

```
int eval(int x,
         boolean c1,
         boolean c2){
   if (c1)
     x++;
   if (c2)
     x--;
   return x;
}
```

Code:

```
int eval(int x,
         boolean c1,
         boolean c2){
    if (c1)
      x++;
    if (c2)
      x--;
    return x;
}
```


Test:

```
eval(0,false,false);
```

Statement:

Code:

```
int eval(int x,
         boolean c1,
         boolean c2){
   if (c1)
      x++;
   if (c2)
      x--;
   return x;
}
```

9

Test:

```
eval(0,false,false);
```

Statement:
60%

Code:

```
int eval(int x,
         boolean c1,
         boolean c2){
    if (c1)
      x++;
    if (c2)
      x--;
    return x;
}
```

Test:

```
eval(0,true,true);
```

Statement:
100%

Code:

```
int eval(int x,
         boolean c1,
         boolean c2){
   if (c1)
      x++;
   if (c2)
      x--;
   return x;
}
```

Test:

```
eval(0,true,true);
```

Decision:

Code:

```
int eval(int x,
         boolean c1,
         boolean c2){
    if (c1)
      x++;
    if (c2)
      x--;
    return x;
}
```

Test:

```
eval(0,true,true);
```

Decision:
50%

Code:

```
int eval(int x,
         boolean c1,
         boolean c2){
    if (c1)
      x++;
    if (c2)
      x--;
    return x;
}
```

Test:

```
eval(0,true,true);
eval(0,false,false);
```

Decision:
100%

Code:

Test:

```
eval(0,true,true);
```

```
int eval(int x,
         boolean c1,
         boolean c2){
   if (c1)
      x++;
   if (c2)
      x--;
   return x;
}
```

FAADO-178B

MCC:

Code:

```
int eval(int x,
         boolean c1,
         boolean c2){
   if (c1)
      x++;
   if (c2)
      x--;
   return x;
}
```

Test:

```
eval(0,true,true);
eval(0,false,false);
```

MCC:
50%

Code:

```
int eval(int x,
         boolean c1,
         boolean c2){
   if (c1)
      x++;
   if (c2)
      x--;
   return x;
}
```

Test:

```
eval(0,true,true);
eval(0,false,false);
eval(0,true,false);
eval(0,false,true);
```

MCC:
100%

100% Statement

```
eval(0,true,true);
```

100% Decision

```
eval(0,true,true);
eval(0,false,false);
```

100% MCC

```
eval(0,true,true);
eval(0,false,false);
eval(0,true,false);
eval(0,false,true);
```

"Testing shows the presence, not the absence of bugs."

— Dijkstra

Code coverage

Actionable

Cheap

Intuitive

!Correctness

Past studies:

Small programs

Synthetic suites

!Control suite size

Year	Corr?	Large programs	Realistic suites	# of tests controlled
1993	~	✗	✗	✓
1994	✓	✗	✗	✓
1994	✓	✗	~	✓
1997	~	✗	✗	✓
1998	~	✗	✗	✓
1999	✗	✗	✓	✓
2005	~	✗	~	✗
2006	✓	✗	~	✓
2009	~	✗	~	✓
2013	✓	✓	~	✗
2014	✓	✓	✓	✗

Year				# of tests rolled
1993				✓
1994				✓
1994	✓	✗	✓	✓
1997				✓
1998				✓
1999				✓
2005		✗		✗
2006				✓
2009				✓
2013				✗
2014	✓	✓	✓	✗

Is effectiveness correlated with suite size?

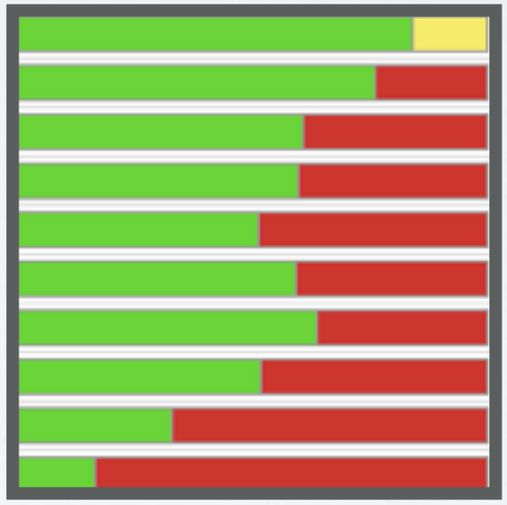
Is effectiveness correlated with suite coverage?

Are stronger coverage criteria more indicative of effectiveness?

Experimental Method

Select Programs

Make Test Suites



Measure Suite Coverage

Measure Suite Effectiveness

Experimental method

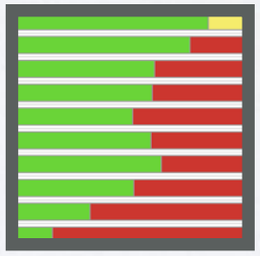
Non-trivial programs

Developer-written test suites

	Apache POI	Joda Time	HSQLDB	Closure Compiler	JFreeChart
KLOC	284	80	178	724	126
# Tests	1,415	3,857	628	7,947	1,764

Select Programs

Make Test Suites



Measure Suite Coverage

Measure Suite Effectiveness

Experimental method

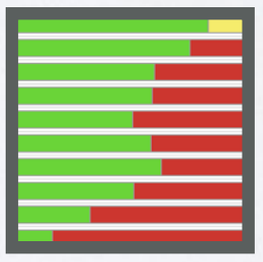
Selected randomly from developer-written tests

Suite sizes: 3, 10, 30,100,300,1000,3000

1000 suites at each size, 31,000 total suites

Select Programs

Make Test Suites



Measure Suite Coverage

Measure Suite Effectiveness

Experimental

method

Statement

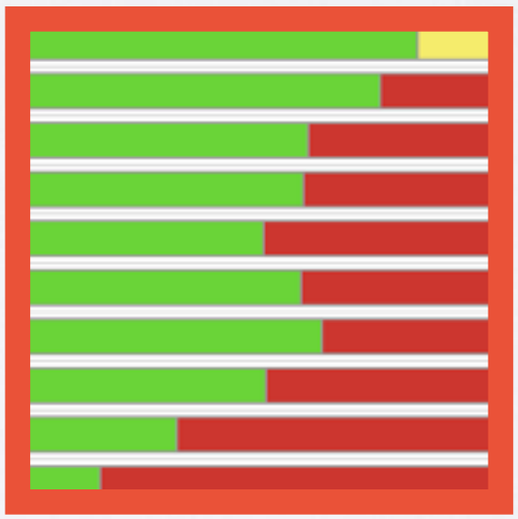
Decision

Code Cover

MCC

Select Programs

Make Test Suites



Measure Suite Coverage

Measure Suite Effectiveness

Experimental

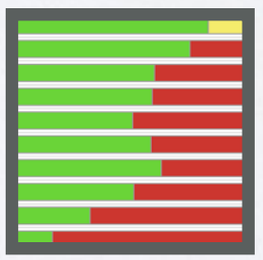
Mutant kill score

method

Pit

Select Programs

Make Test Suites

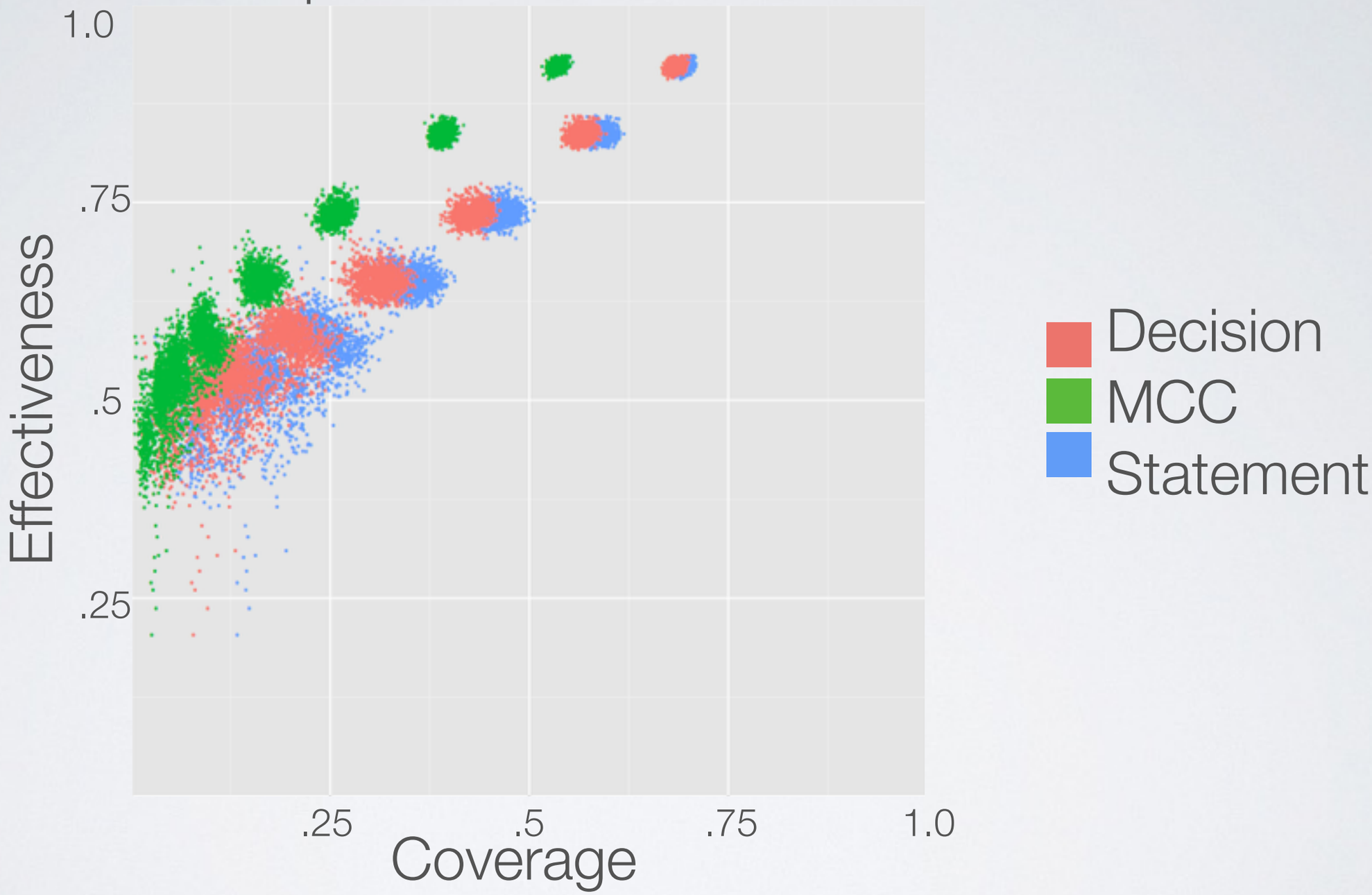


Measure Suite Coverage

Measure Suite Effectiveness

Experimental results

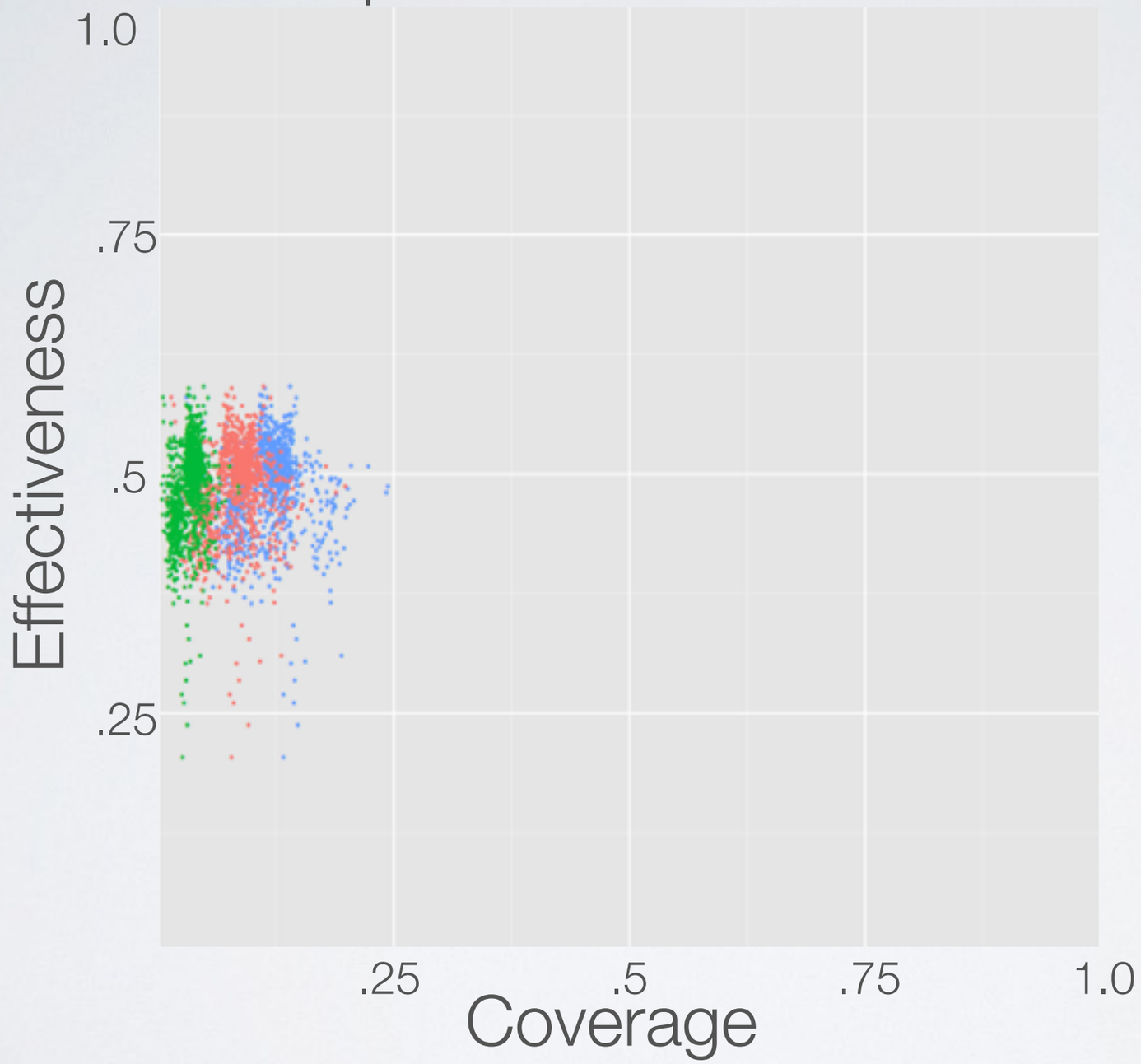
Closure Compiler: All Sizes



Effectiveness not correlated with coverage

Experimental results

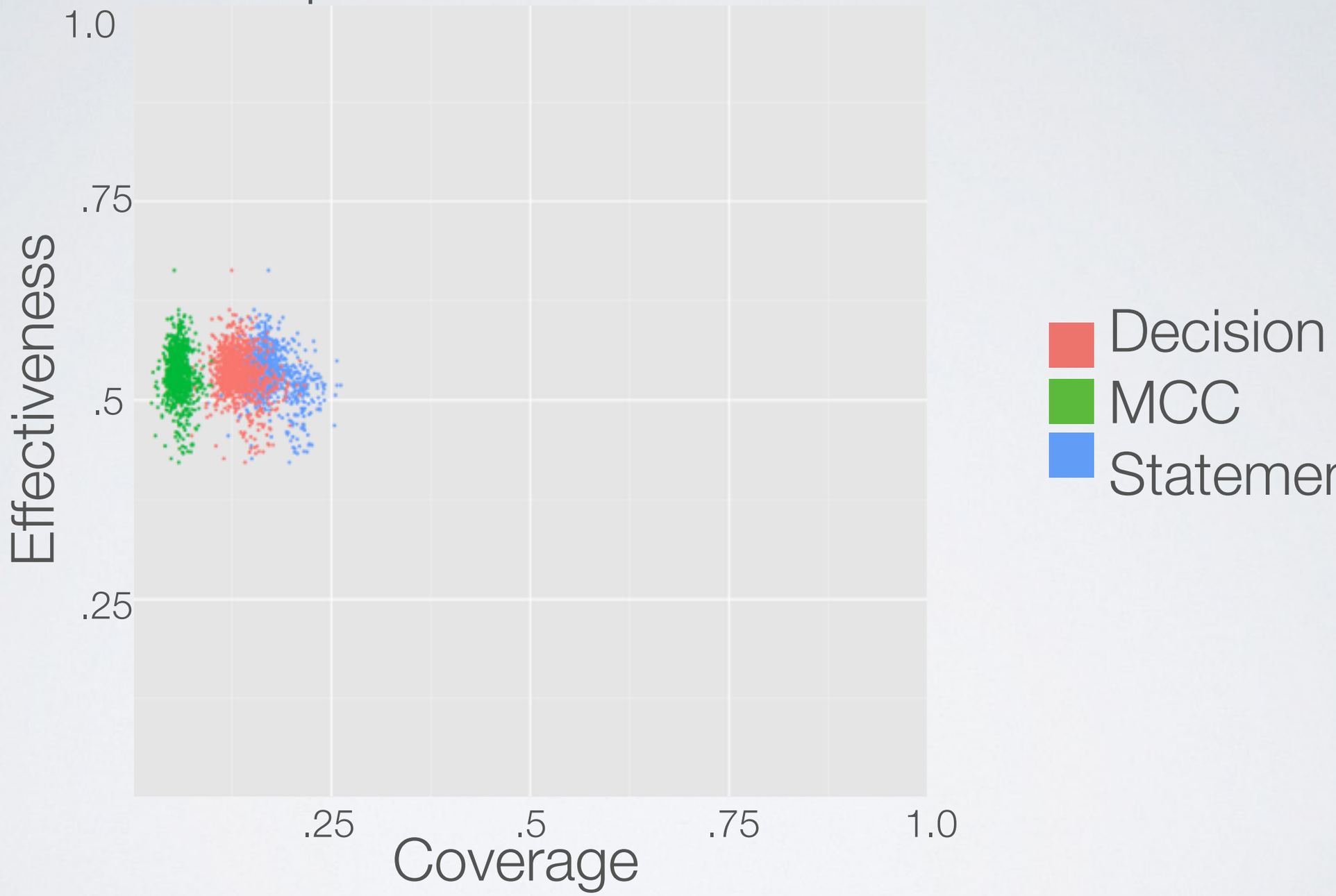
Closure Compiler: Size 3



Effectiveness not correlated with coverage

Experimental results

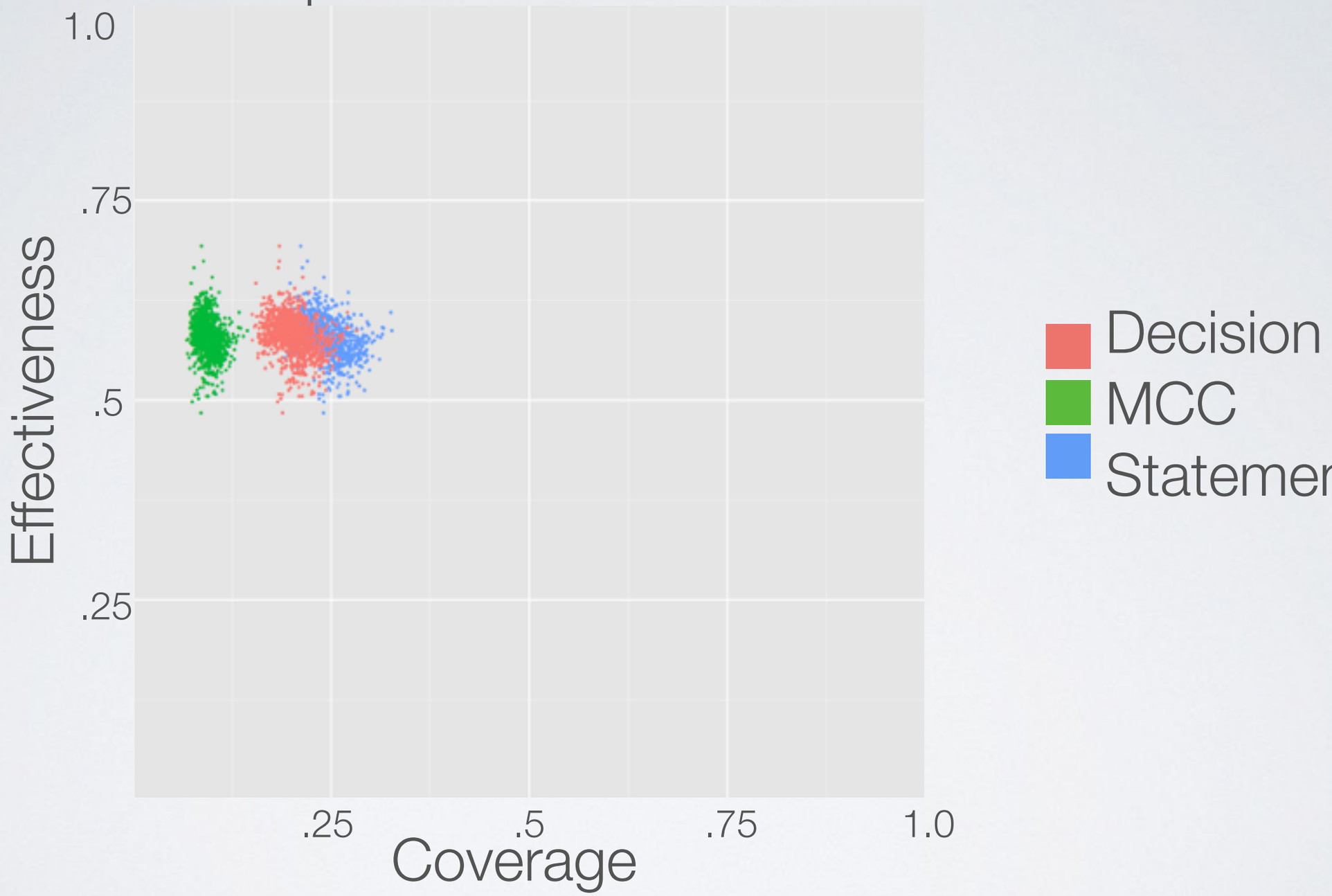
Closure Compiler: Size 10



Effectiveness not correlated with coverage

Experimental results

Closure Compiler: Size 30

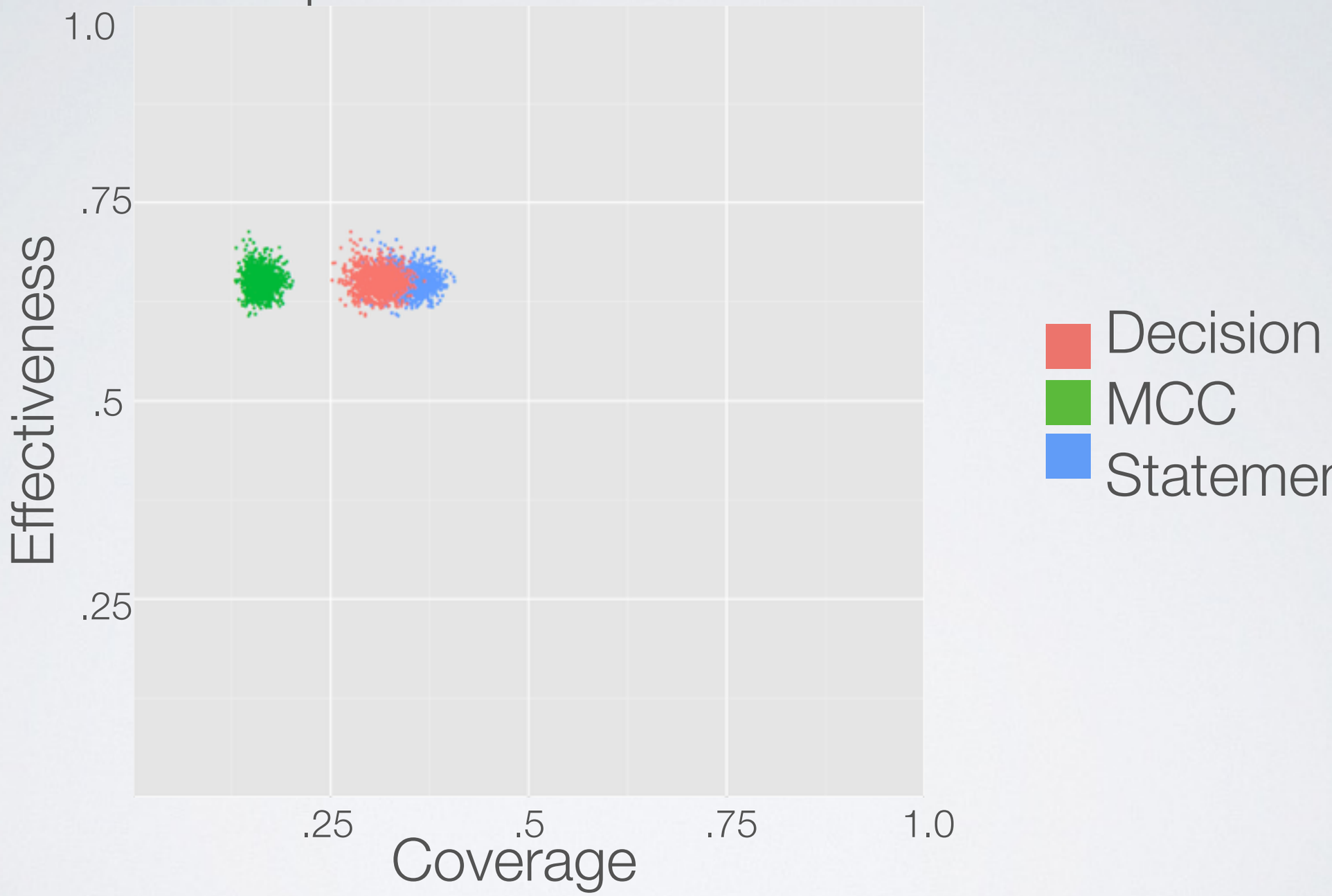


30

Effectiveness not correlated with coverage

Experimental results

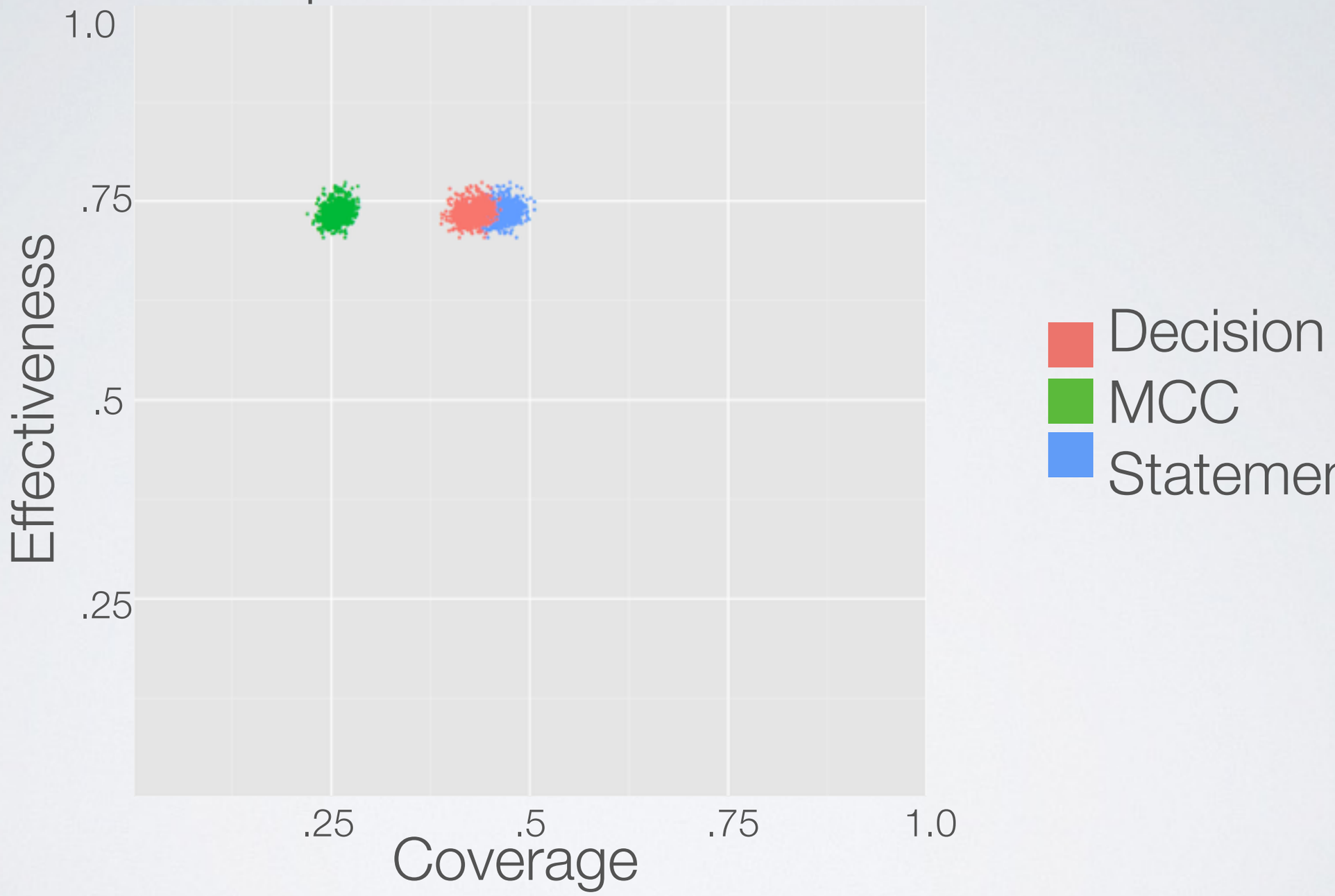
Closure Compiler: Size 100



Effectiveness not correlated with coverage

Experimental results

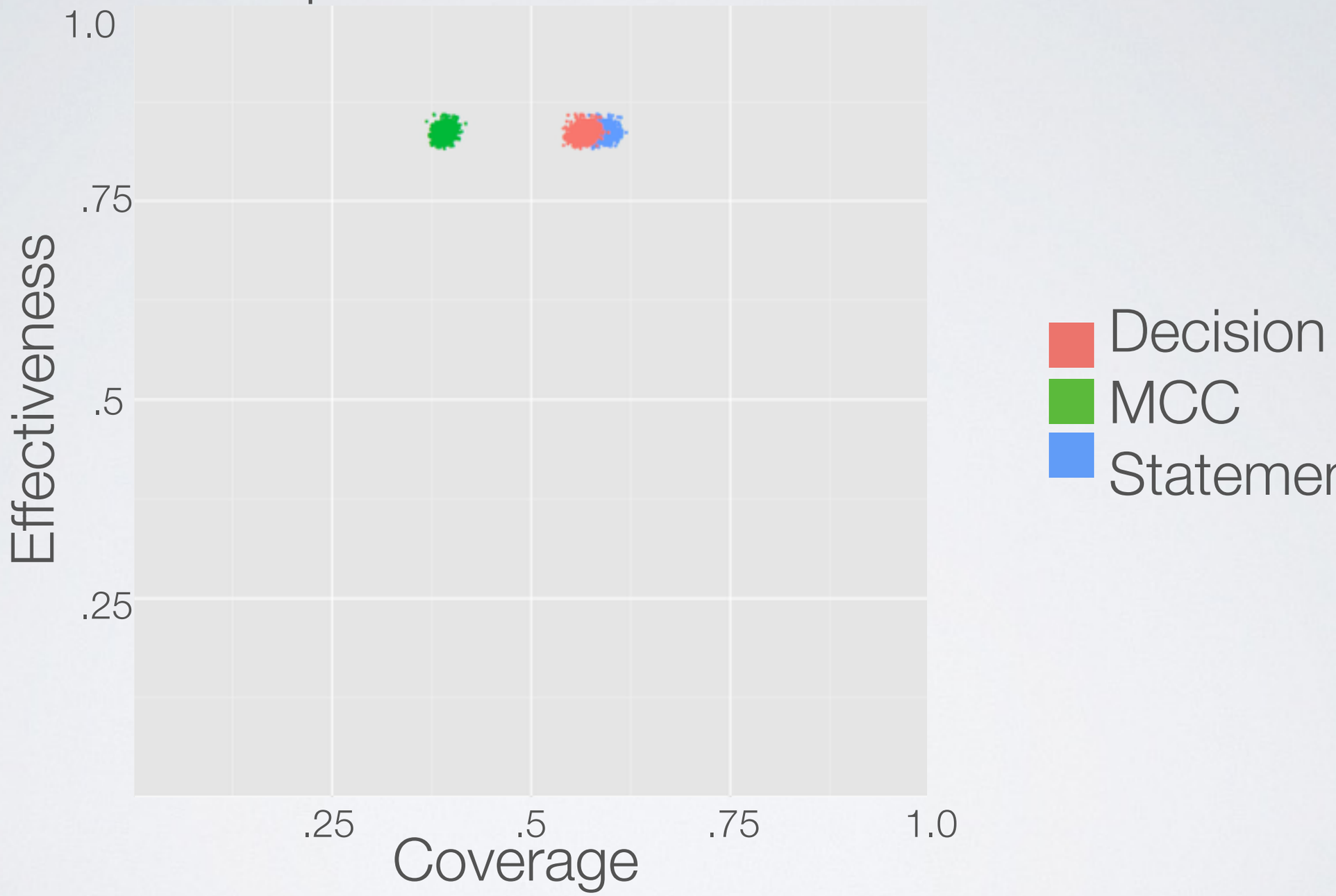
Closure Compiler: Size 300



Effectiveness not correlated with coverage

Experimental results

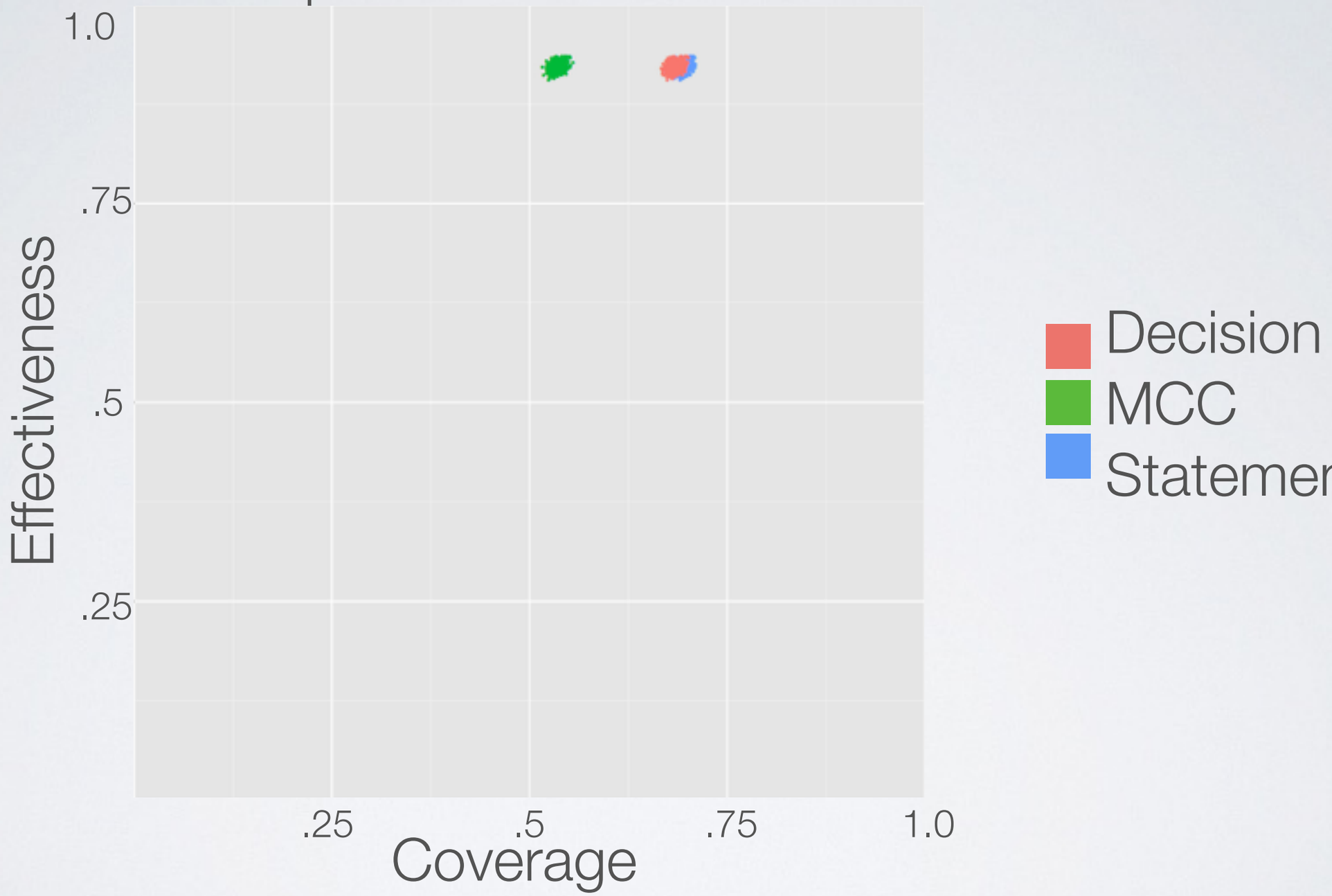
Closure Compiler: Size 1000



Effectiveness not correlated with coverage

Experimental results

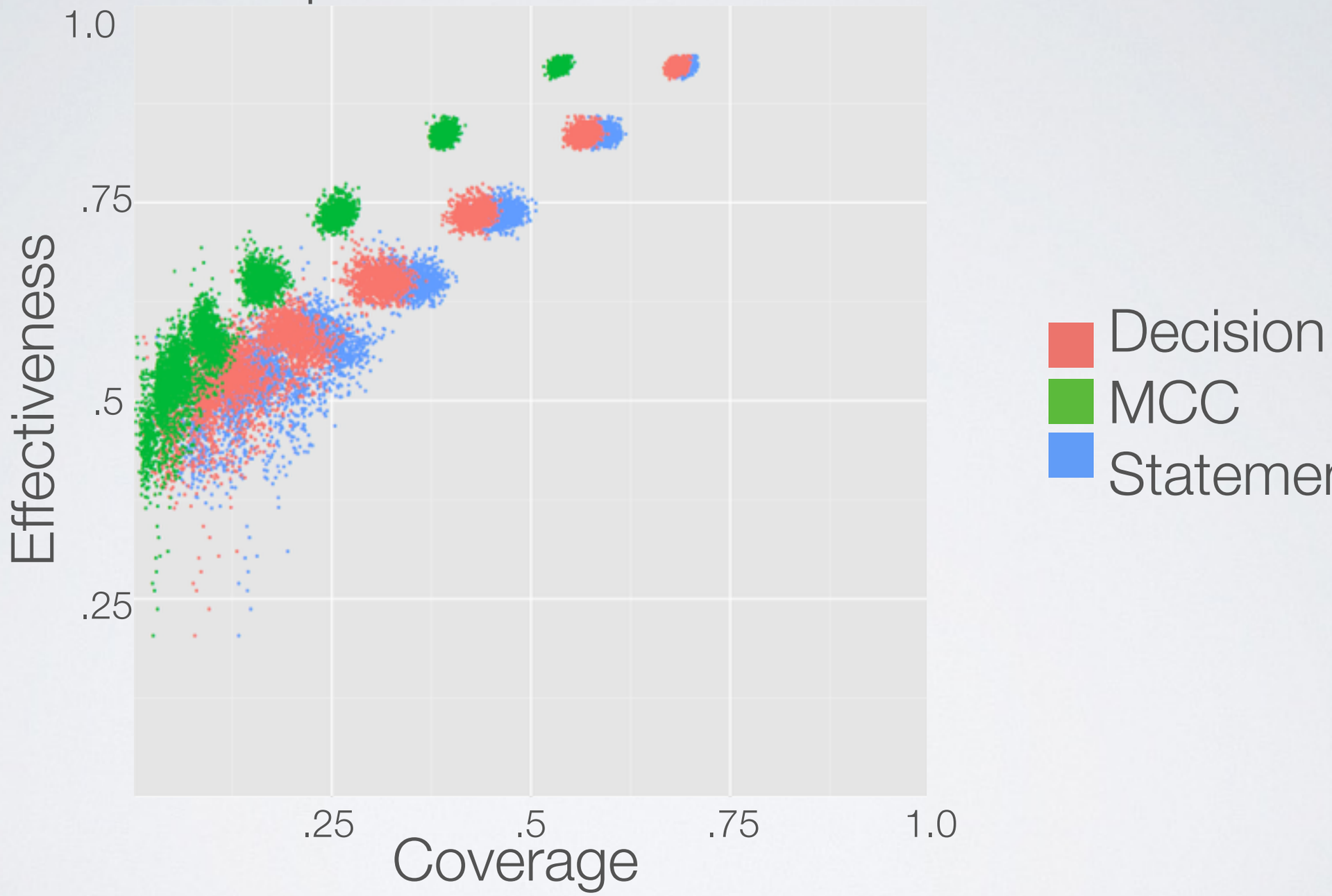
Closure Compiler: Size 3000



Effectiveness not correlated with coverage

Experimental results

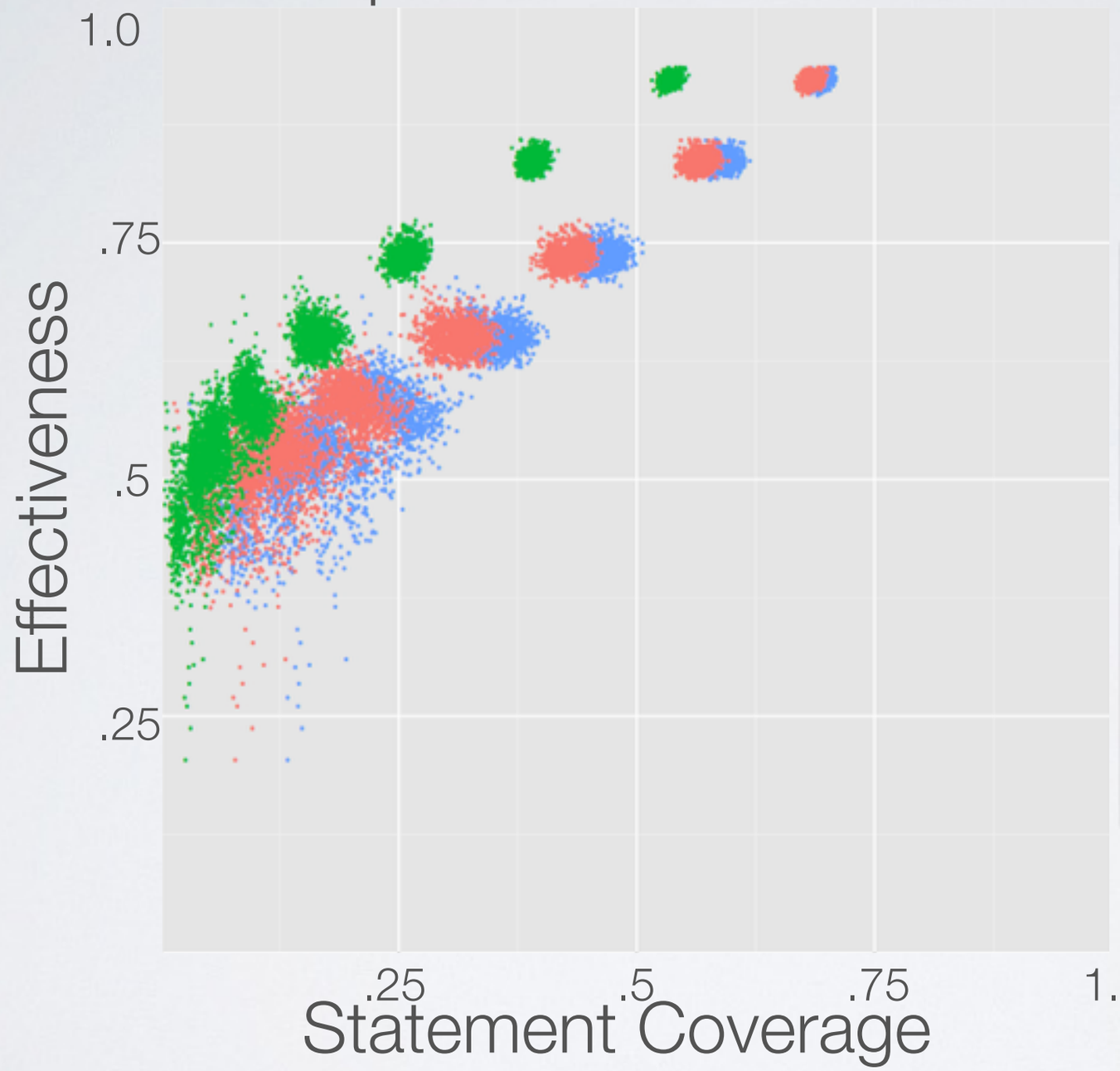
Closure Compiler: All Sizes



Effectiveness not correlated with coverage

Experimental results

Closure Compiler: All Sizes



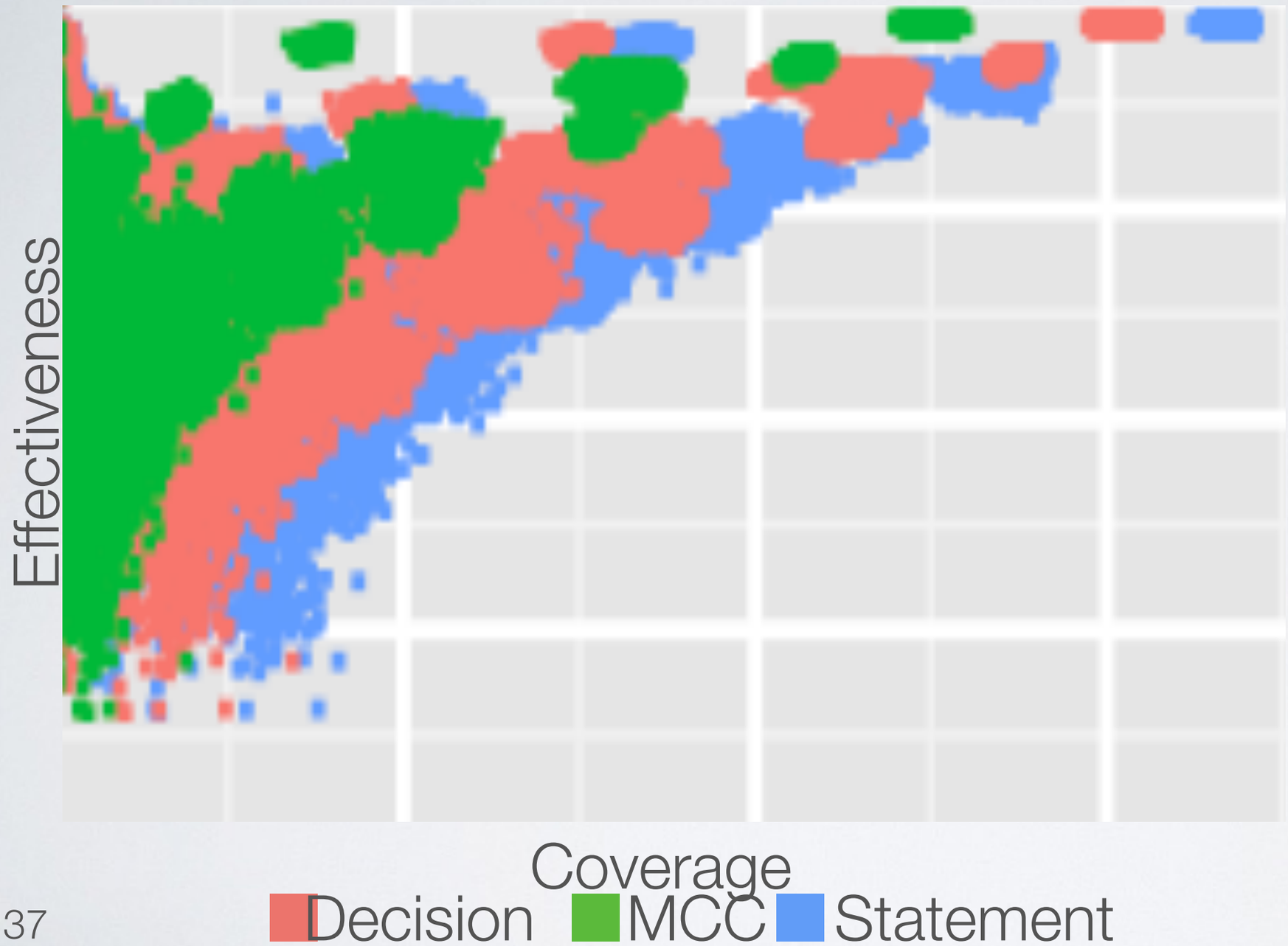
STRONG
Coverage & Size : Effectiveness

WEAK
Coverage (-size) : Effectiveness

Effectiveness not correlated with coverage

Experimental results

All projects, all sizes



EQUIVALENT
Criteria :
Effectiveness

Effectiveness not correlated with coverage

Coverage takeaway

More testing is better than artificially broad testing.

High-level comments

- Testability is a function of the structure of the code, not a function of what it does.
- Structural constraints inhibit testing.
  - Global state
  - Hard-coded object creation (aka `new`)
  - References to time
- Code should always ‘ask’ for what it needs, not search for it.
  - E.g., constructors should state dependencies.
- Don’t assume the system is fixed: sometimes tests require restructuring.

An example

http://bit.ly/92Ozrz

```
public class InboxSyncer {

  private static final InboxSyncer instance =
     new InboxSyncer(Config.get());
  public static getInstance() { return instance; }
  private final Certificate cert;
  private final String username;
  private final String password;
  private long lastSync = -1;

  private InboxSyncer(Config config) {
    this.cert = DESReader.read(config.get("cert.path"));
    User user = config.getUser();
    this.username = user.getUsername();
    this.password = user.getPassword();
  }
  public sync() {
    …
  }
}
```

An example

```
public sync() {
     long syncFrom = lastSync;
     if (syncFrom == -1) {
       syncFrom = new Date().getTime() - Def.SYNC_AMOUNT;
     }
     POPConnector pop =
        new POPConnector(cert, username, password);
     pop.connect();
     try {
          …
   } finally {
      pop.disconnect();
     }
}
```

Global state

```
public class InboxSyncer {

  private static final InboxSyncer instance =
     new InboxSyncer(Config.get());
  public static getInstance() { return instance; }
  private final Certificate cert;
  private final String username;
  private final String password;
  private long lastSync = -1;

  private InboxSyncer(Config config) {
    this.cert = DESReader.read(config.get("cert.path"));
    User user = config.getUser();
    this.username = user.getUsername();
    this.password = user.getPassword();
  }
  public sync() {
    …
  }
}
```

Other configuration

```
public class InboxSyncer {

   private static final InboxSyncer instance =
      new InboxSyncer(Config.get());
   public static getInstance() { return instance; }
   private final Certificate cert;
   private final String username;
   private final String password;
   private long lastSync = -1;

   private InboxSyncer(Config config) {
      this.cert = DESReader.read(config.get(“cert.path”));
      User user = config.getUser();
      this.username = user.getUsername();
      this.password = user.getPassword();
   }
   public sync() {
     …
   }
}
```

An example

```
public sync() {
    long syncFrom = lastSync;
    if (syncFrom == -1) {
      syncFrom = new Date().getTime() - Def.SYNC_AMOUNT;
    }
    Inbox inbox = Inbox.get(username);
    POPConnector pop =
       new POPConnector(cert, username, password);
    pop.connect();
    try {
         …
   } finally {
      pop.disconnect();
    }
}
```

Improved structure

```
public class InboxSyncer {

  private final POPConnector _pop;
  private final String _username;
  private final String _password;
  private long _lastSync = -1;

  private InboxSyncer(POPConnector pop,
      String user, String pass, long defaultSync) {
        _pop = pop;
        _username = user;
        _password = pass;
        _lastSync = defaultSync;
  }
  public sync() {
    …
  }
}
```

Improved structure

```
public sync(Date now) {
    Date syncFrom = new Date(now.getTime() - _lastSync);
    _pop.connect();
    try {
            …
    } finally {
        _pop.disconnect();
     }
}
```