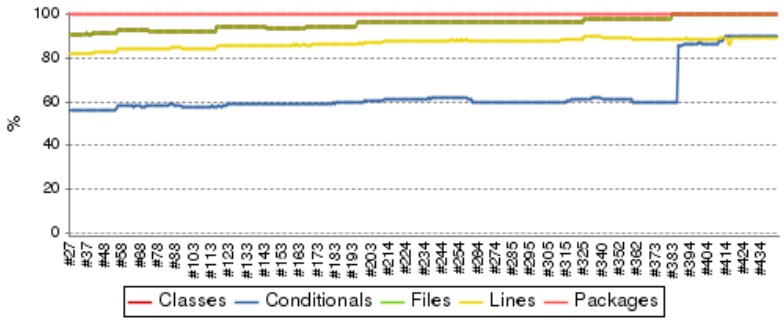
khmer project

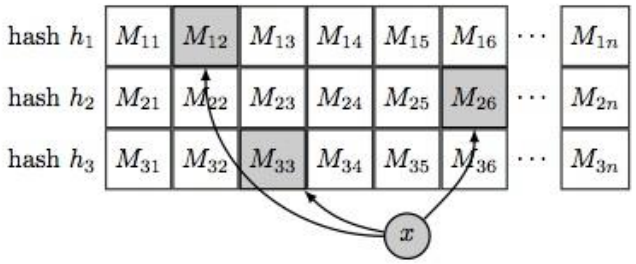
- Repository: https://github.com/dib-lab/khmer
- Unit testing (some integration testing too)
- ~90% coverage (lines and conditionals)
- CI server: http://ci.oxli.org
- Every PR is tested
- Sketches for biological data analysis
  - Diginorm
  - Partitioning
  - Read mapping/alignment



Algorithm 1 Count-Min Sketch

insert(x):
for $i = 1$ **to** d **do**
  $M[i, h_i(x)] \leftarrow M[i, h_i(x)] + 1$
end for

query(x):
$c = \min \{M[i, h_i(x)]$ for all $1 \leq i \leq d\}$
return c



Property-based testing

- Properties = testable specification
- Automatic input generation
  - Random
  - Customized generators
- Shrinking

```
[<Test>]
let ``When I add 1 + 2, I expect 3``()=
    let result = add 1 2
    Assert.AreEqual(3,result)
```

```
let add x y =
    if x=1 && y=2 then
        3
    else
        0
```

http://fsharpforfunandprofit.com/posts/property-based-testing/

Property-based testing

```
[<Test>]
let ``Adding 1 twice is the same as adding 2``()=
    for _ in [1..100] do
        let x = randInt()
        let y = randInt()
        let result1 = x |> add 1 |> add 1
        let result2 = x |> add 2
        Assert.AreEqual(result1,result2)
```

```
[<Test>]
let ``When I add two numbers, the result should not depend on parameter order``()=
    for _ in [1..100] do
        let x = randInt()
        let y = randInt()
        let result1 = add x y
        let result2 = add y x // reversed params
        Assert.AreEqual(result1,result2)
```

```
[<Test>]
let ``Adding zero is the same as doing nothing``()=
    for _ in [1..100] do
        let x = randInt()
        let result1 = x |> add 0
        let result2 = x
        Assert.AreEqual(result1,result2)
```

```
let add x y =
    if x=1 && y=2 then
        3
    else
        0
```

http://fsharpforfunandprofit.com/posts/property-based-testing/