

Method Chaining Redux: An Empirical Study of Method Chaining in Java, Kotlin, and Python



Ali M. Keshk and Robert Dyer

IN OUR GRIT, OUR GLORY™

What is Method Chaining?

without method chaining

```
1 JSONObject obj = new JSONObject();  
2 obj.put("Conference", "MSR");  
3 obj.put("Year", "2023");  
4 String json = obj.toString();
```



What is Method Chaining?

without method chaining

```
1 JSONObject obj = new JSONObject();  
2 obj.put("Conference", "MSR");  
3 obj.put("Year", "2023");  
4 String json = obj.toString();
```



What is Method Chaining?

without method chaining

```
1 JSONObject obj = new JSONObject();  
2 obj.put("Conference", "MSR");  
3 obj.put("Year", "2023");  
4 String json = obj.toString();
```



What is Method Chaining?

without method chaining

```
1 JSONObject obj = new JSONObject();  
2 obj.put("Conference", "MSR");  
3 obj.put("Year", "2023");  
4 String json = obj.toString();
```

with method chaining

```
1 String json = new JSONObject()  
2     .put("Conference", "MSR")  
3     .put("Year", "2023")  
4     .toString();
```



What is Method Chaining?

without method chaining

```
1 JSONObject obj = new JSONObject();  
2 obj.put("Conference", "MSR");  
3 obj.put("Year", "2023");  
4 String json = obj.toString();
```

with method chaining

```
1 String json = new JSONObject()  
2               .put("Conference", "MSR")  
3               .put("Year", "2023")  
4               .toString();
```

Definition: We define a method chain as a sequence of one or more method invocations joined by the '.' symbol.

- Language implementers could optimize their compilers to account for method chaining.
- High method chain use can lead API developers to write fluent designs.
- There is a debate on whether method chaining is good or bad practice.



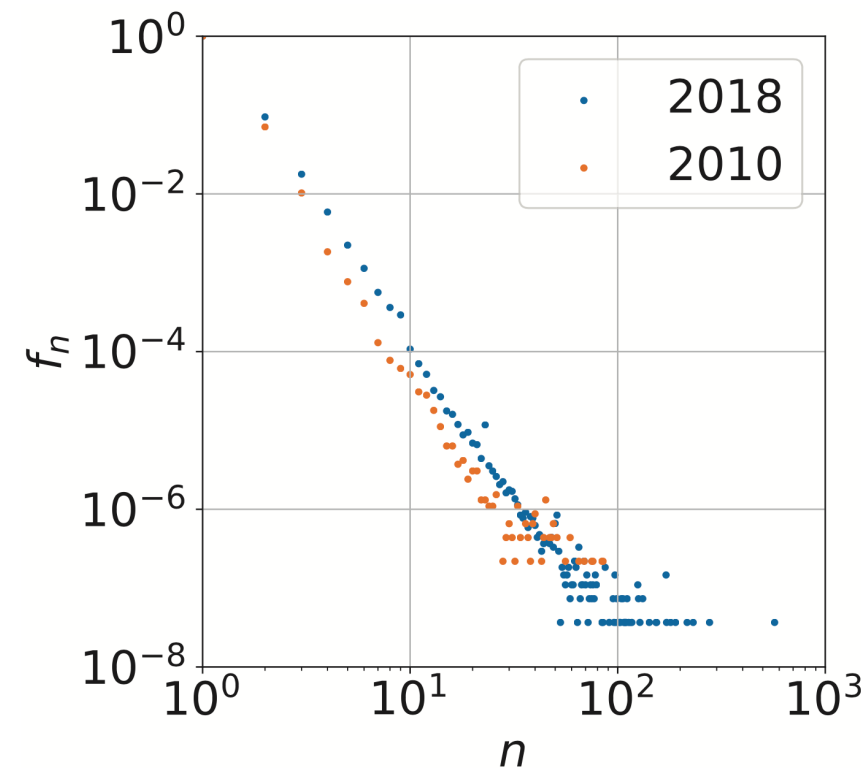
Nakamaru et al. MSR'20

- Do developers use method chaining in Java?

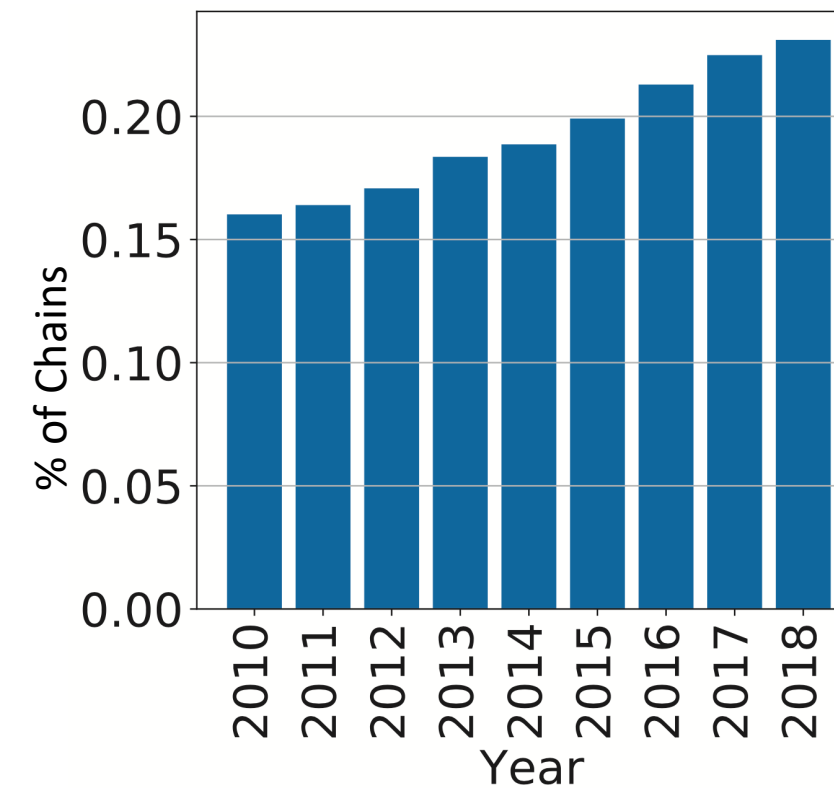


Nakamaru et al. MSR'20

- Do developers use method chaining in Java?
- Finding: use of method chains in Java is **popular** and **increasing**



(a) f_n in 2010 and 2018



(b) r

RQ1 Replication

RQ2 Extend to larger Java dataset?

RQ3 Generalize to Kotlin?

RQ4 Generalize to Python?

RQ5 Can we support the language extensions proposed by the prior study?



	Boa Dataset	Number of Projects
Java Original	"2021 Method Chains"	2,659
Java Full	"2022 Jan/Java"	89,088
Kotlin	"2021 Aug/Kotlin"	26,273
Python	"2022 Feb/Python"	98,202

- Filtered older years with less than 250 projects
- Filtered so all data ends 31 Dec 2020 to avoid partial years
- Duplicate files across projects are filtered out using the hash value of each file's AST






```
1  new C();          // not a method chain
2  super();          // not a method chain

3  // length: 1
4  o.m();            // explicit receiver
5  m();              // implicit 'this' receiver
6  new C().m();       // constructor not included
7  super.m();         // super is not a call here

8  // length: 2
9  o.m().n();

10 // two chains, each length: 1
11 m(n());
12 m().f.n();
```


$$f_n = \frac{m_n}{m_1}$$

$$\frac{\text{number of chains of length } n}{\text{number of not-chained method invocations}}$$

$$r = \frac{\sum_{n \geq 2} n \cdot m_n}{\sum_{n \geq 1} n \cdot m_n}$$

$$\frac{\text{all chained method invocations}}{\text{all method invocations (chained or not)}}$$

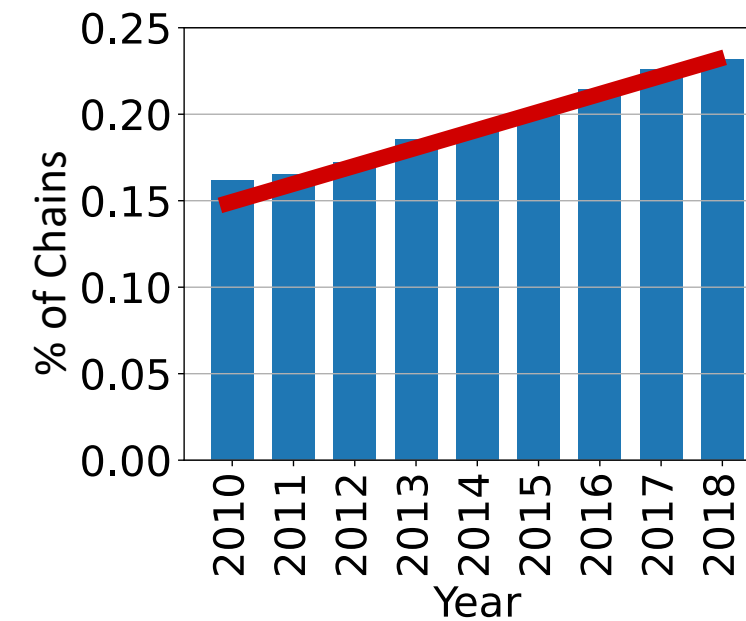
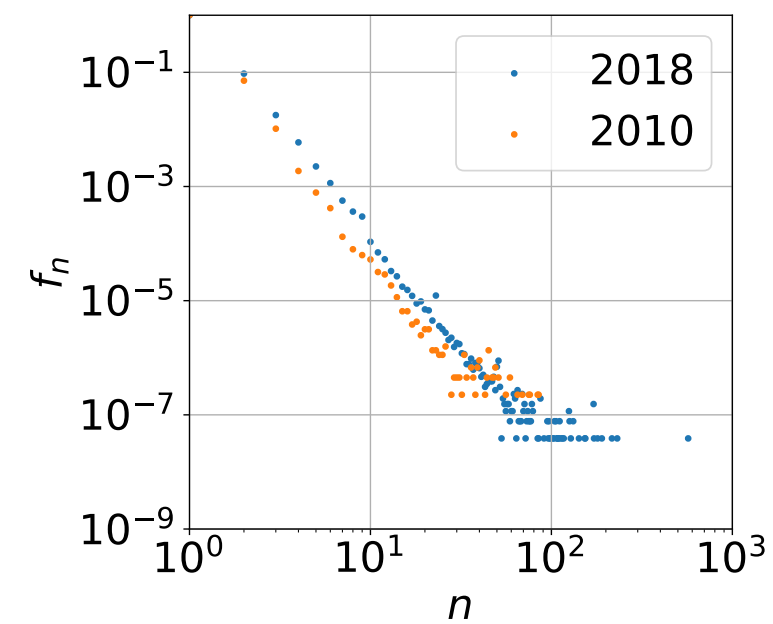
U_n : percent of all projects containing at least one chain whose length is $\geq n$

Testing code: file path including case-insensitive string "test" anywhere

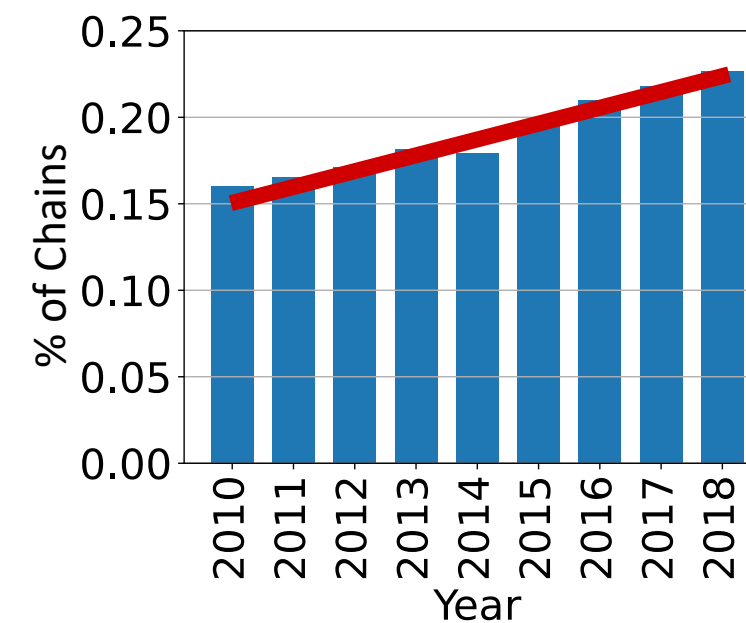
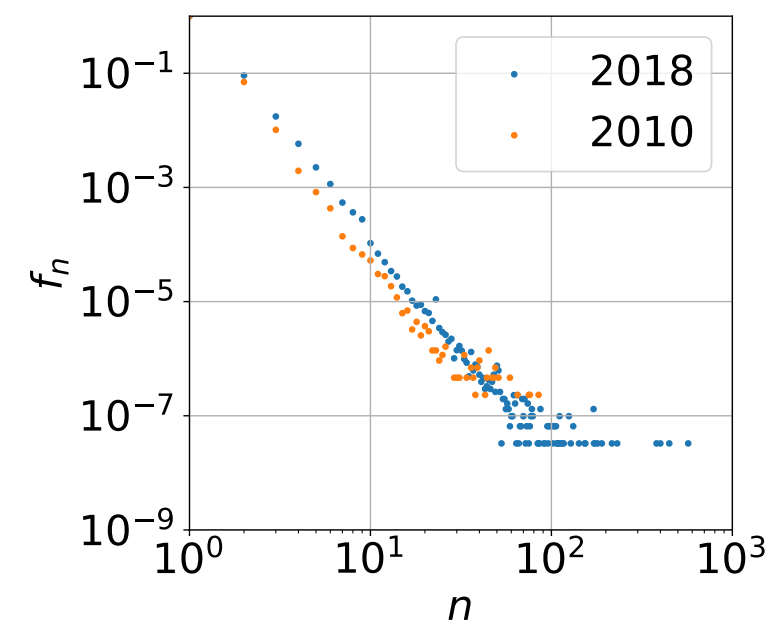


RQ1: Replication

Previous Study's Dataset

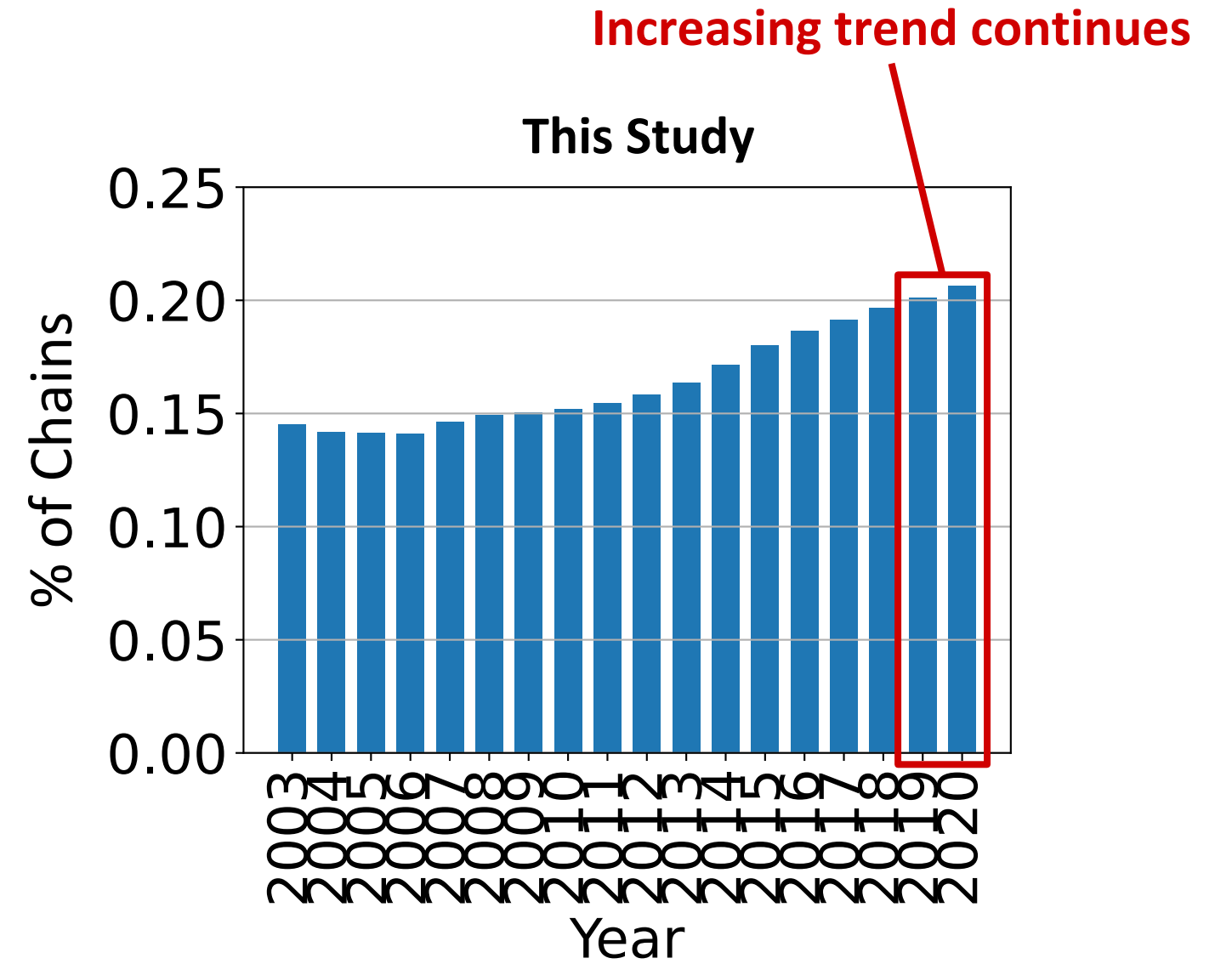
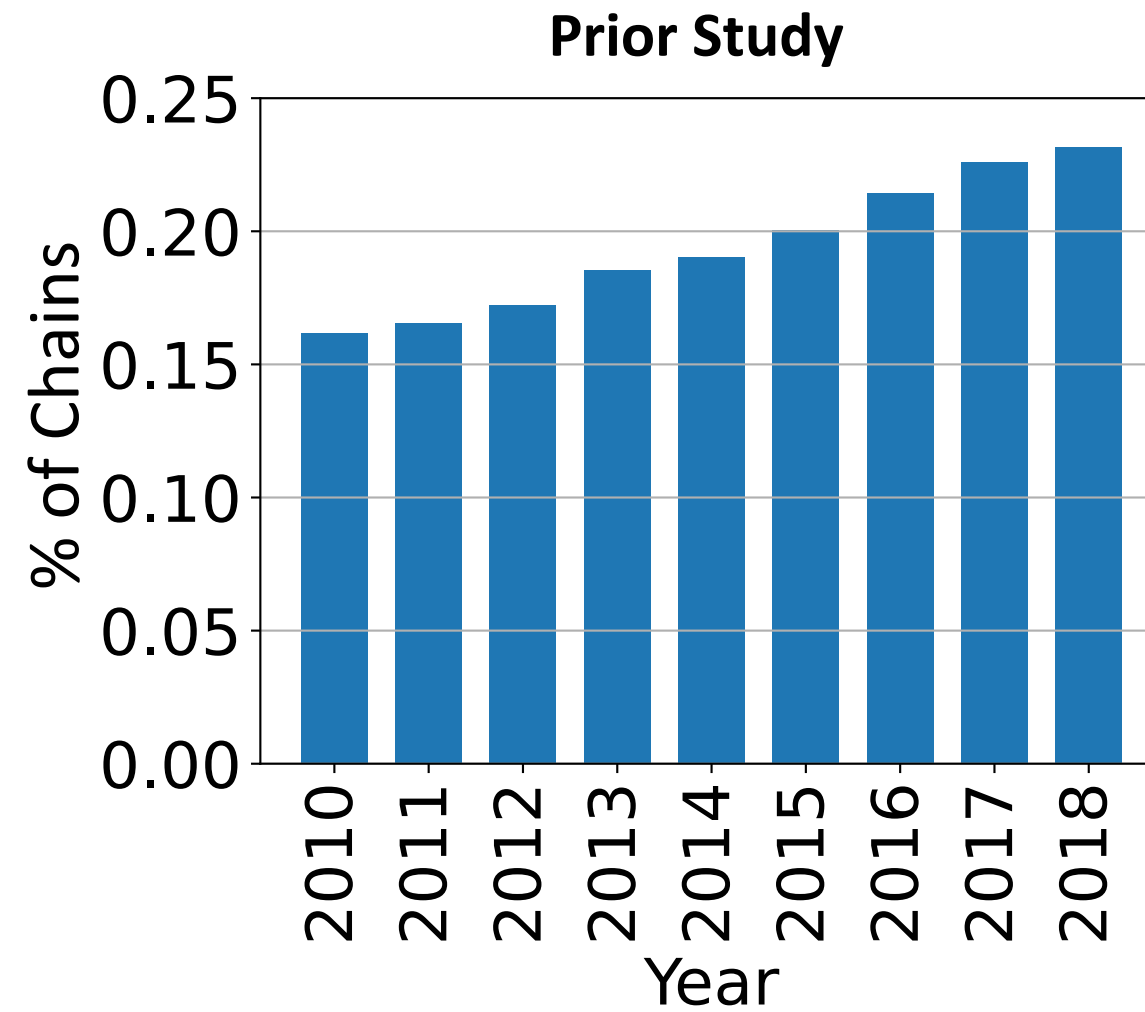


Our Dataset



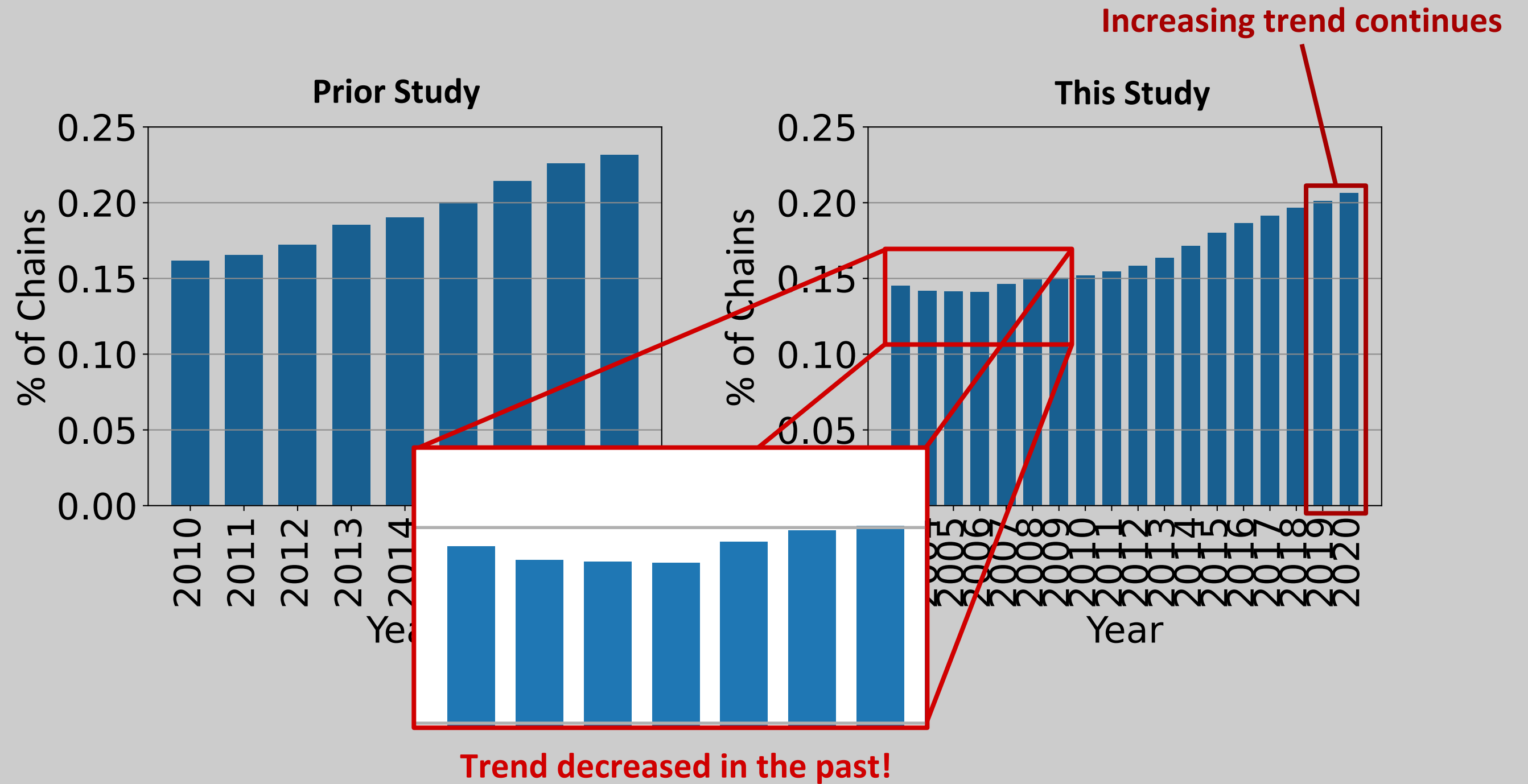


RQ2: Larger Java Dataset



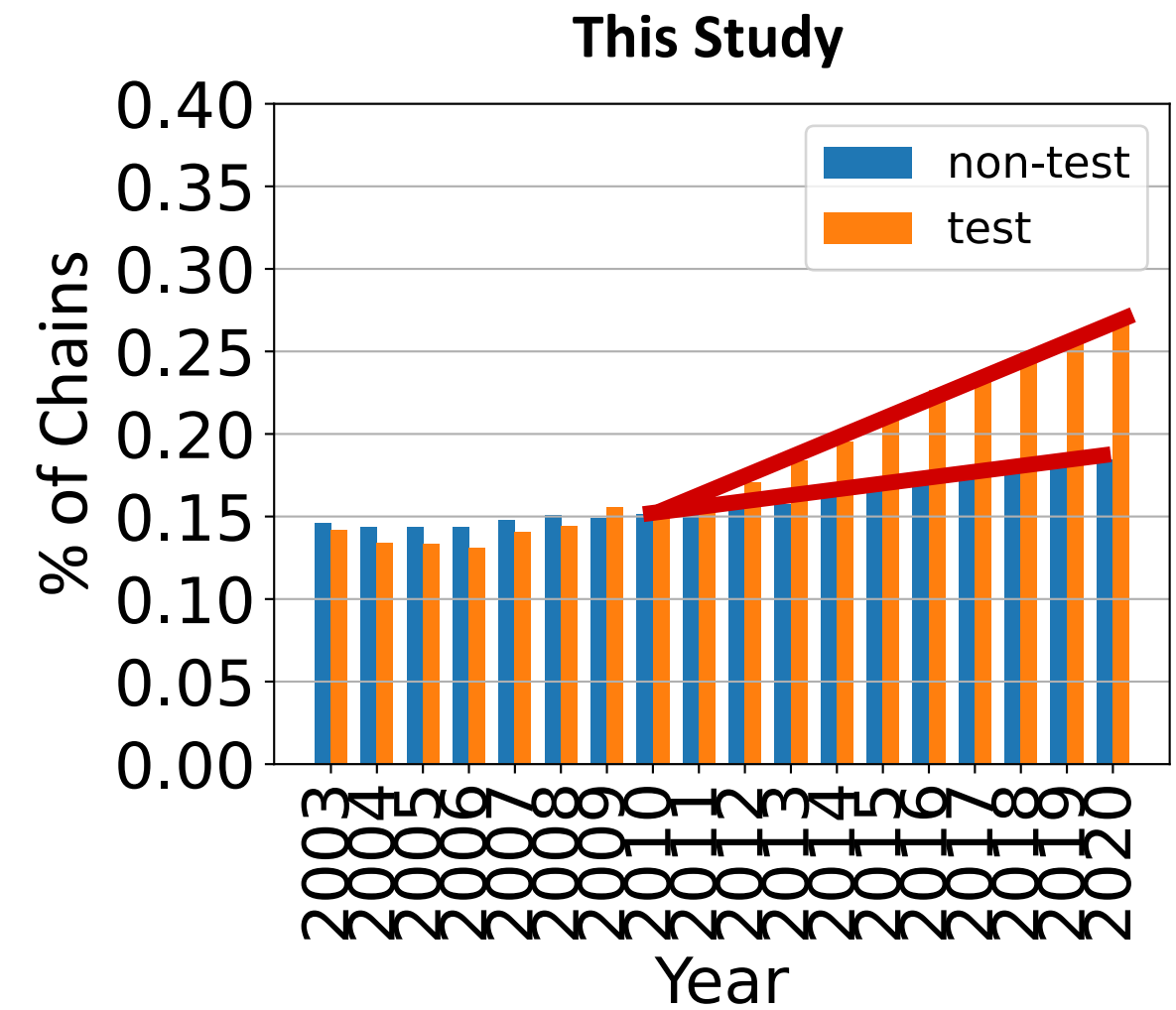
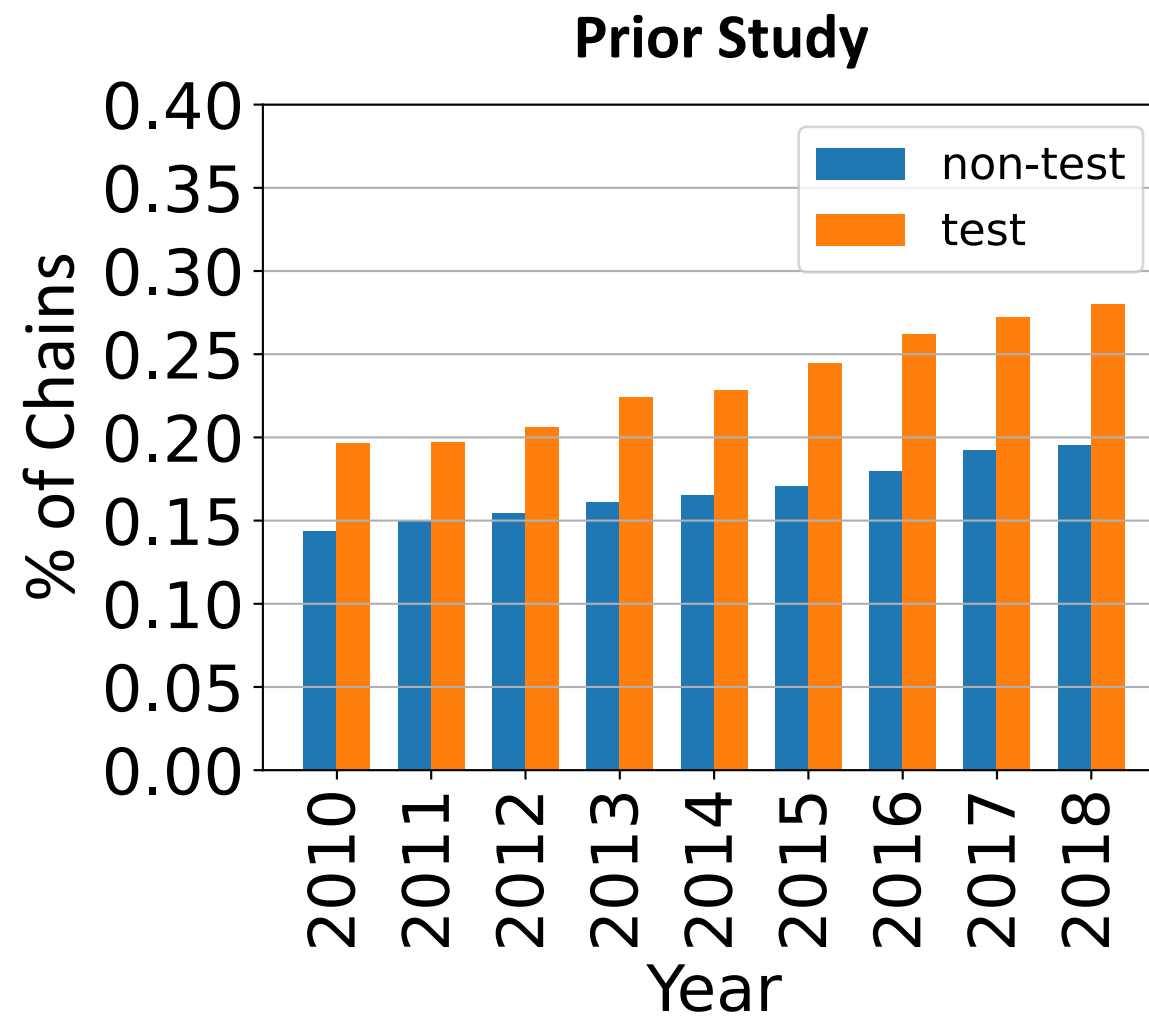


RQ2: Larger Java Dataset



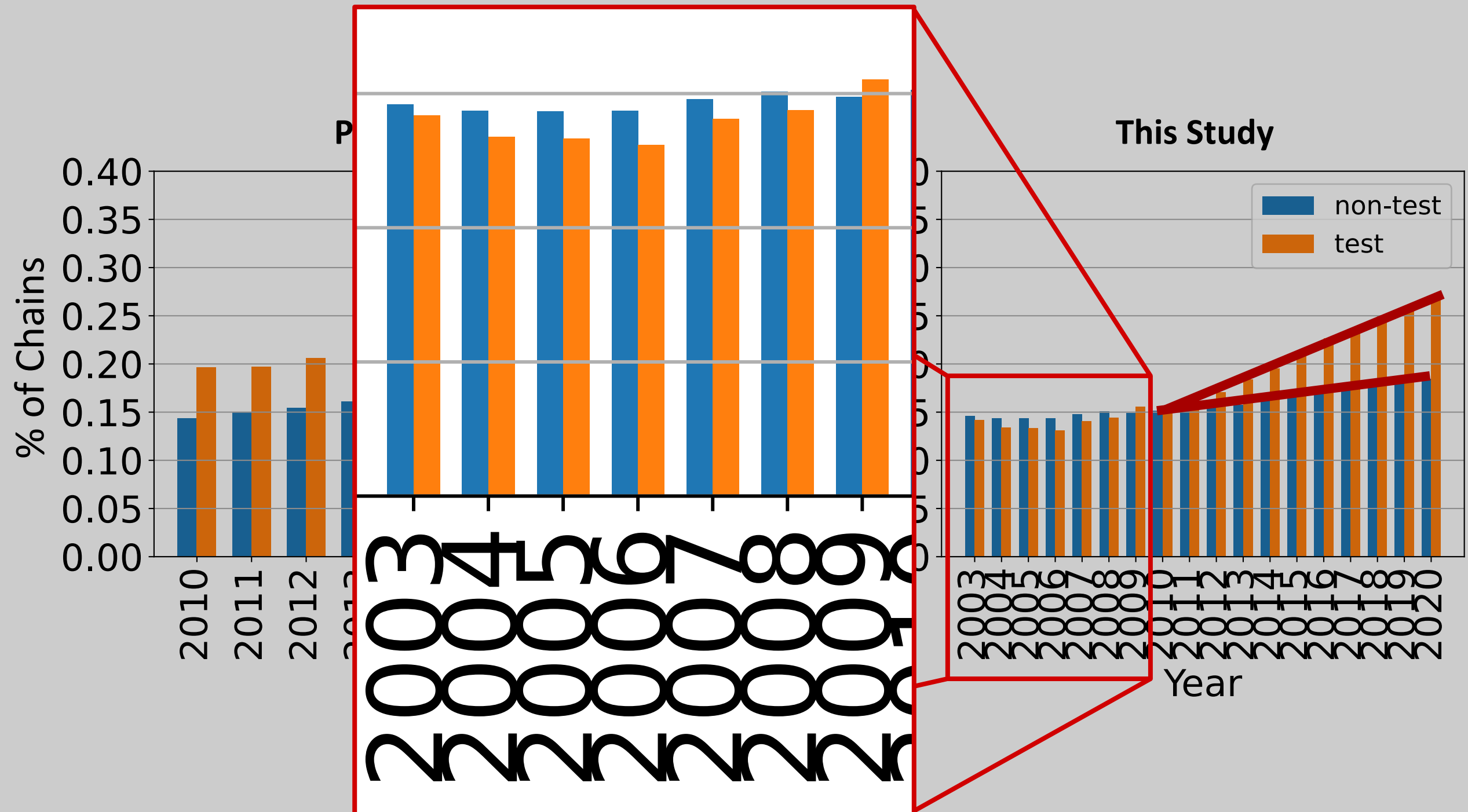


RQ2: Larger Java Dataset





RQ2: Larger Java Dataset

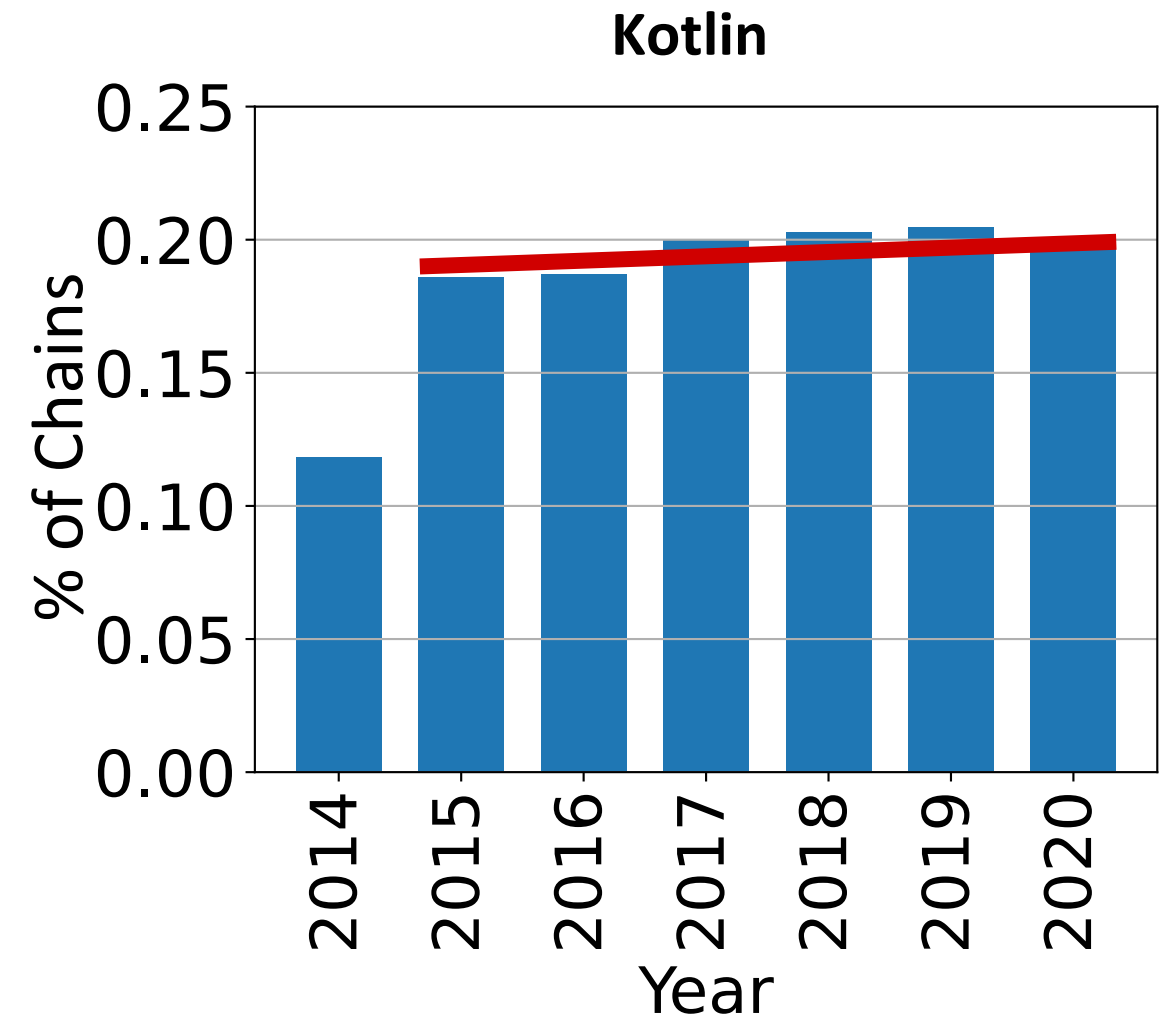
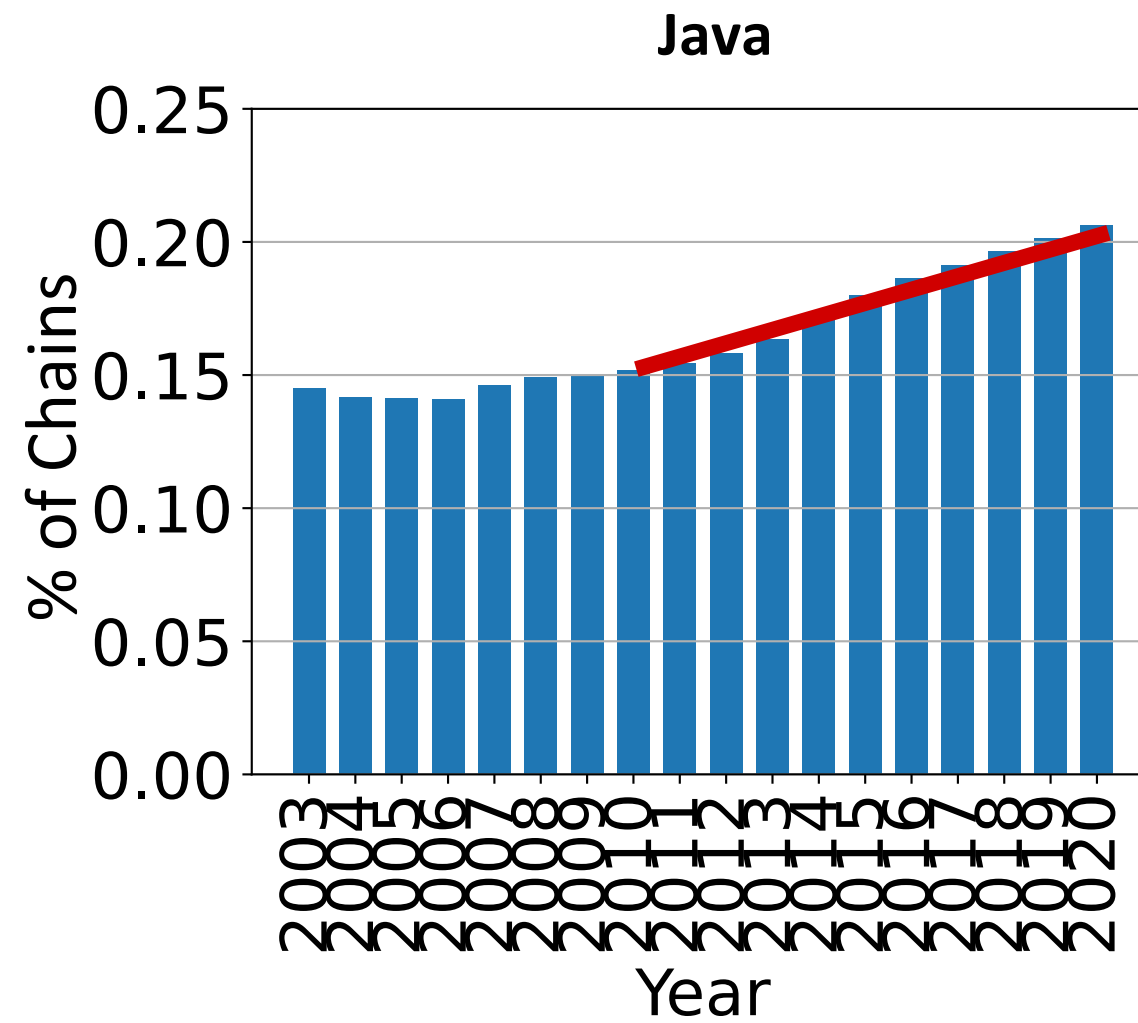


- 1/3 of extra long chains (n in top 5%) come from the following Java APIs:
 - `java.lang.StringBuilder`
 - `org.springframework.security...builders.HttpSecurity`
 - `com.google.common.collect.ImmutableMap`
 - `java.lang.StringBuffer`
 - `javax.swing.GroupLayout`



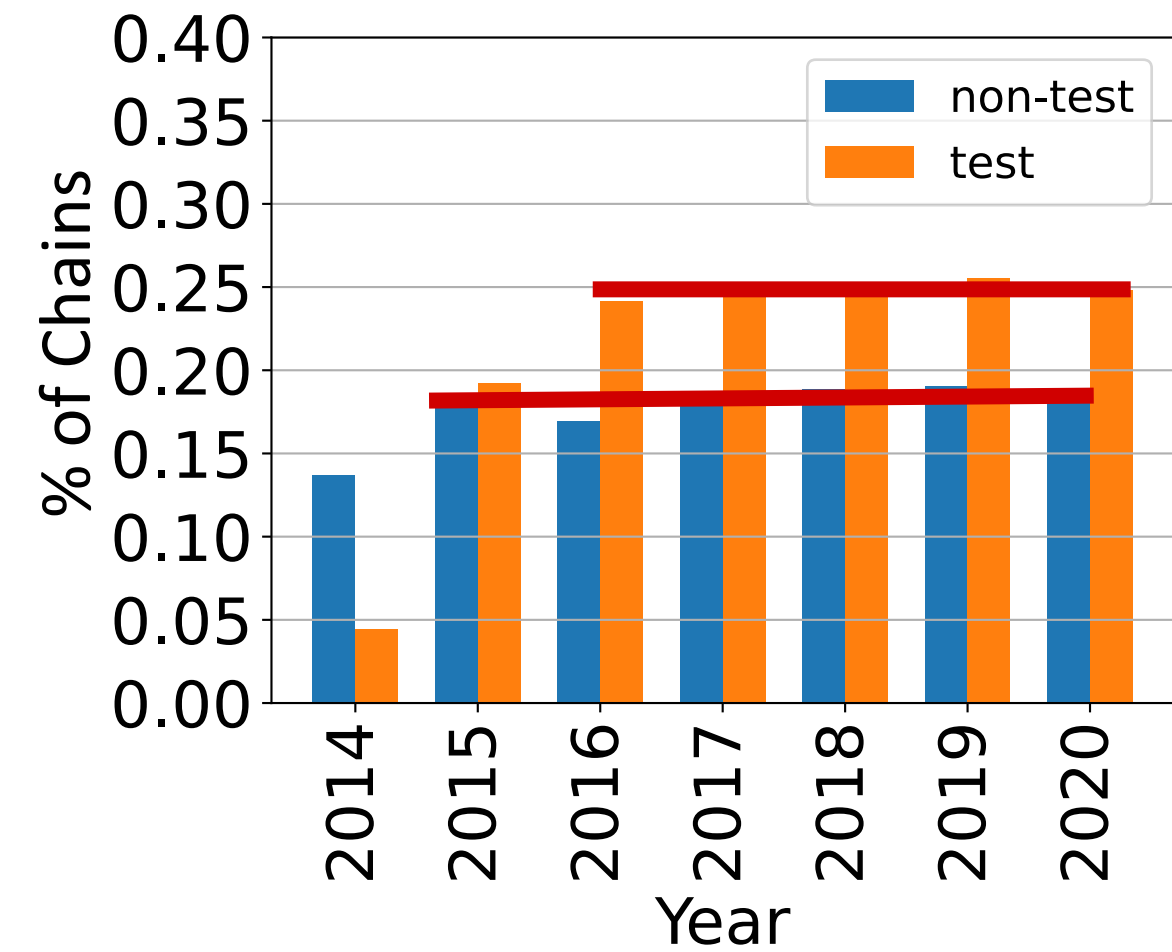
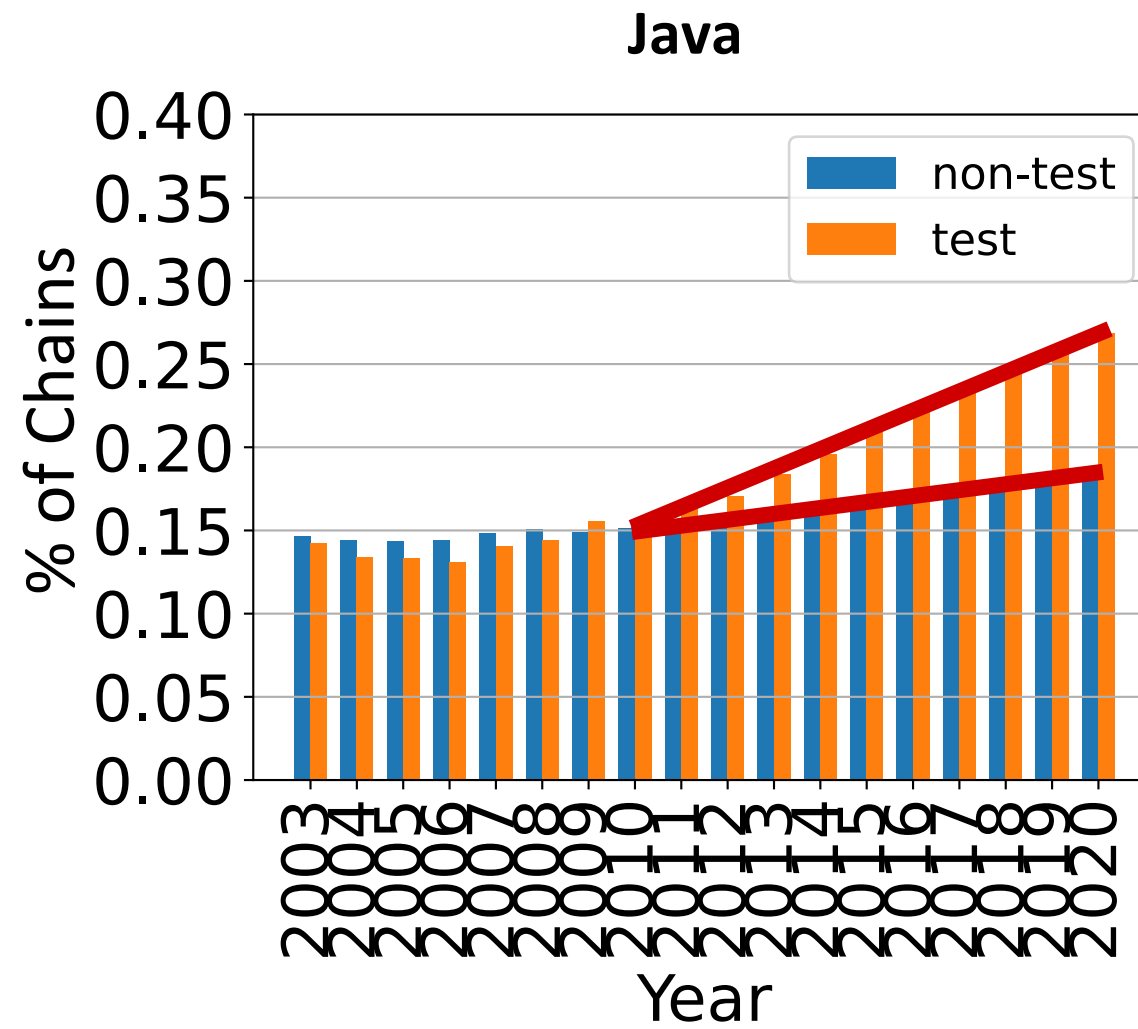


RQ3: Kotlin



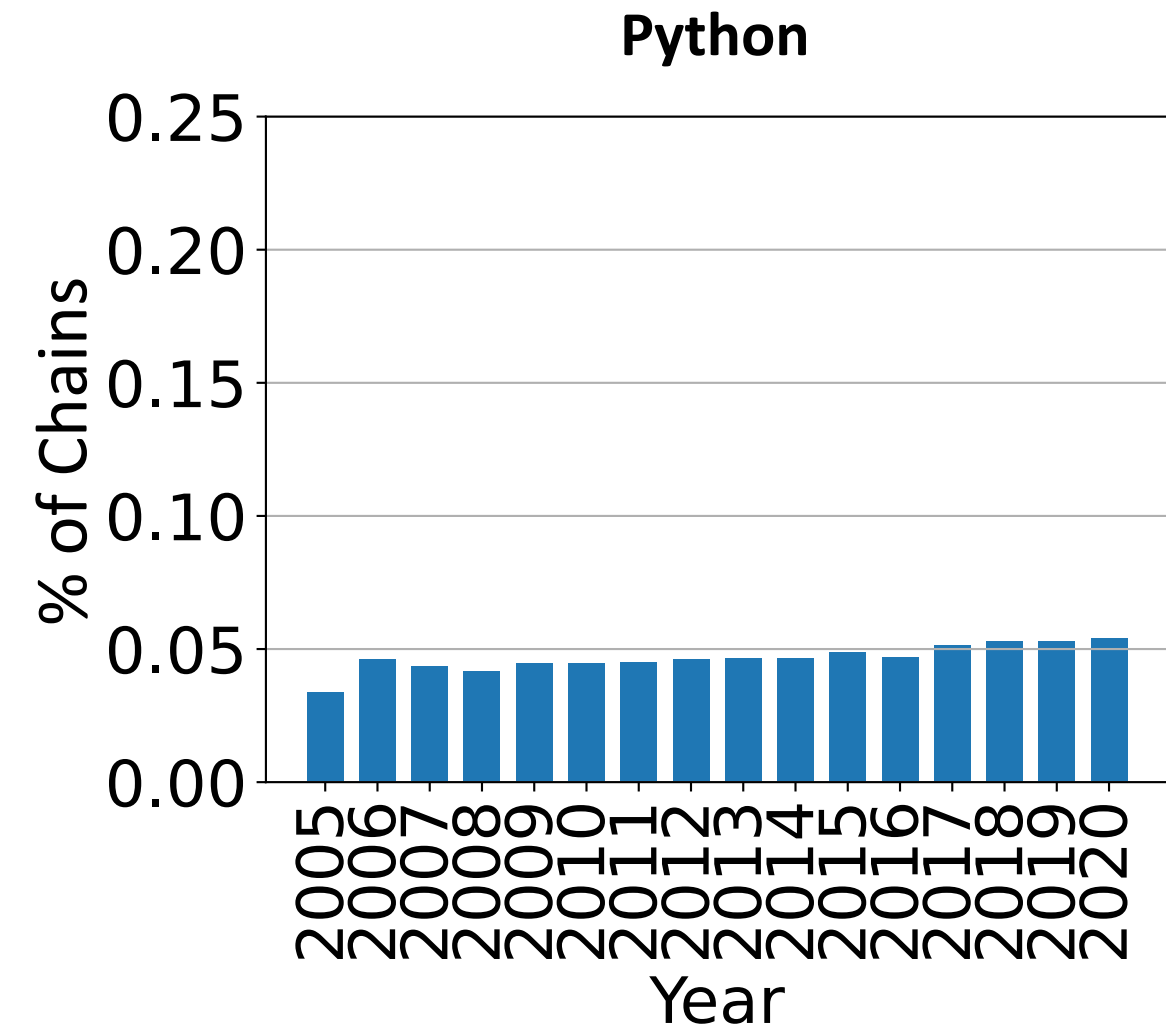
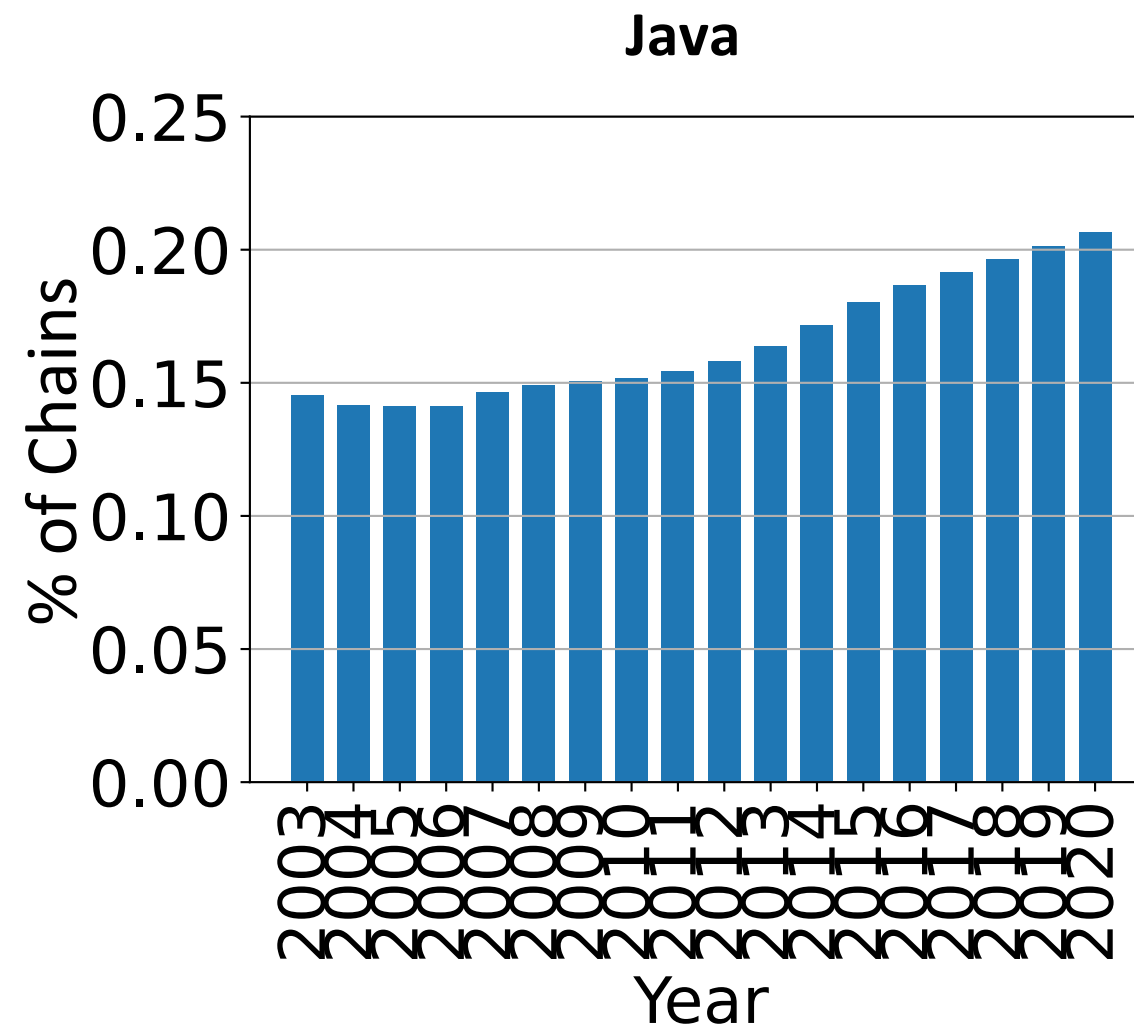


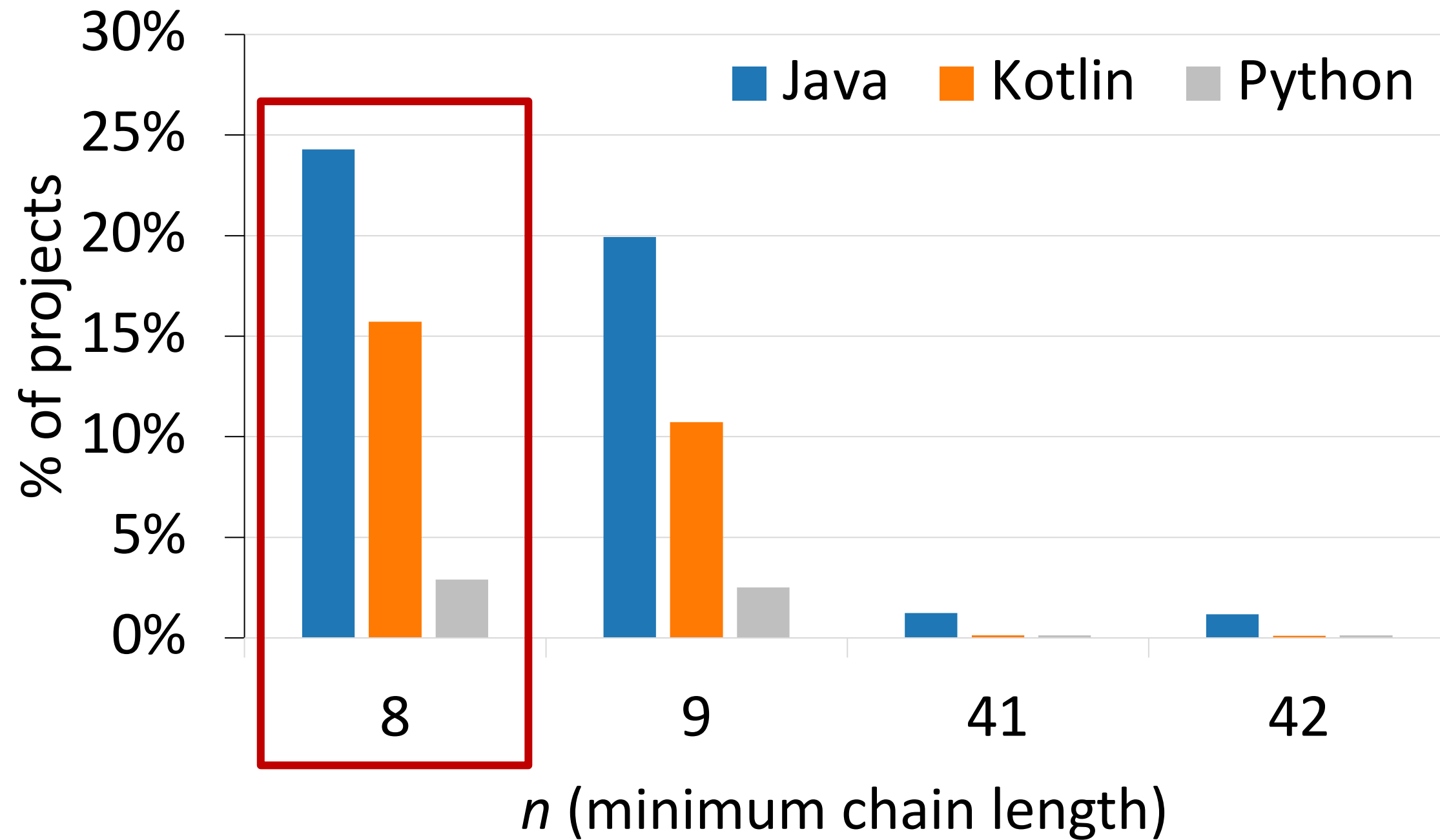
RQ3: Kotlin





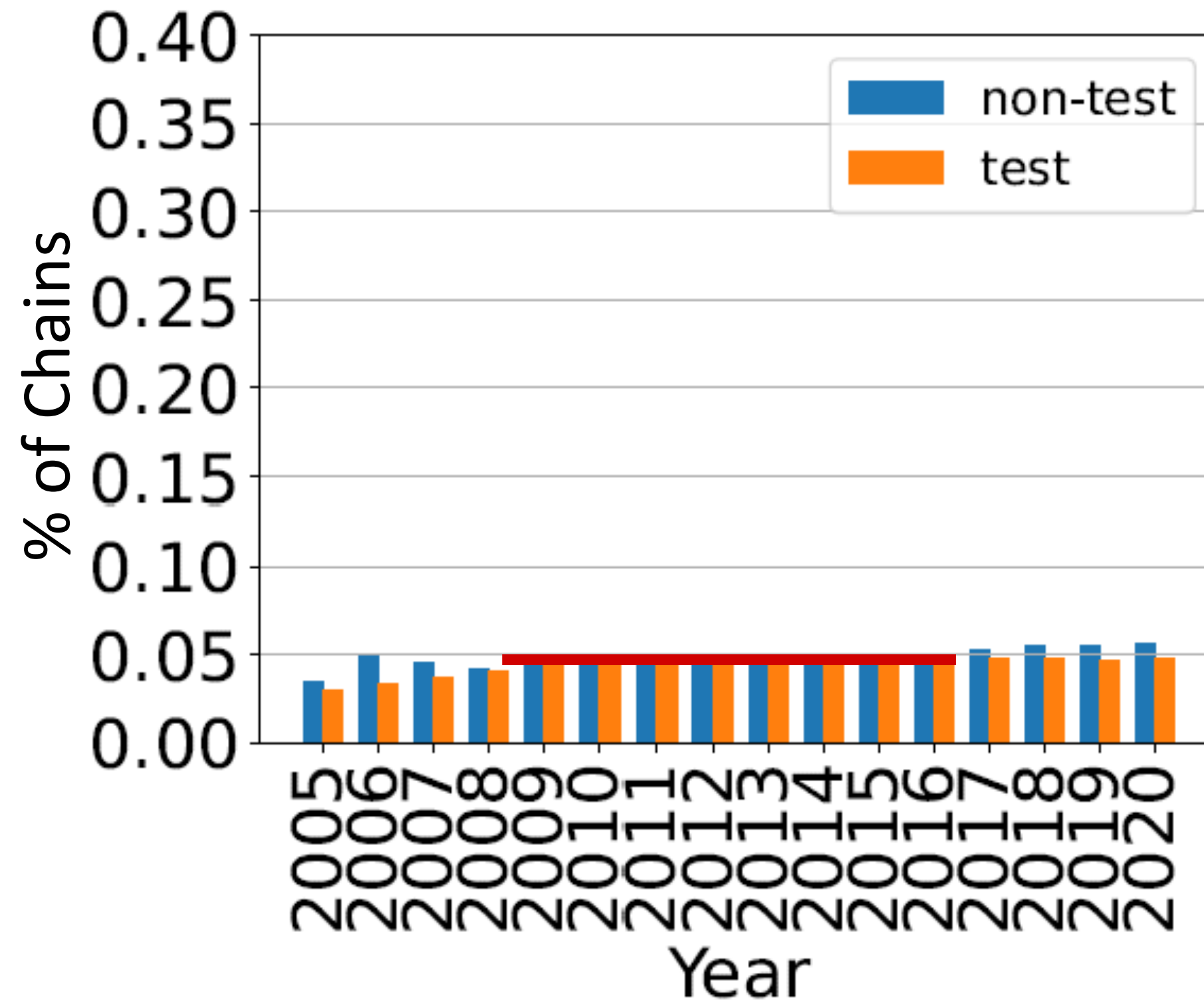
RQ4: Python



U_n values by language

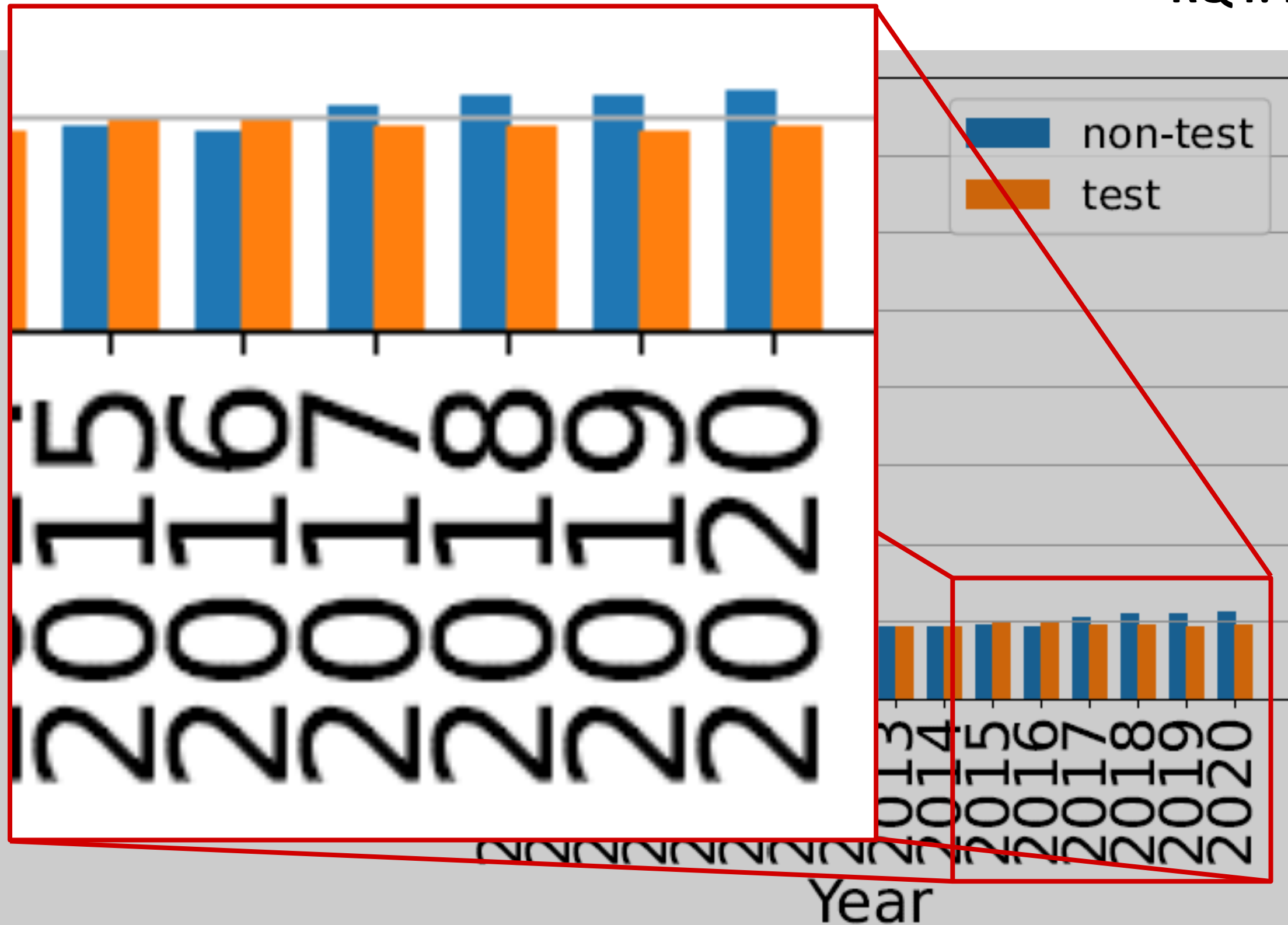


RQ4: Python





RQ4: Python



NullPointerExceptionAvoidance

```
m1 () ? .m2 () .m3 ()
```

**51.5% of 499k
Kotlin projects**

RepeatedReceiver

```
o .m1 () ; o .m2 () ;
```

> 19% of chains

DownCast

```
m1 () .asC () .m2 () .m3 ()
```

< 0.1% of chains

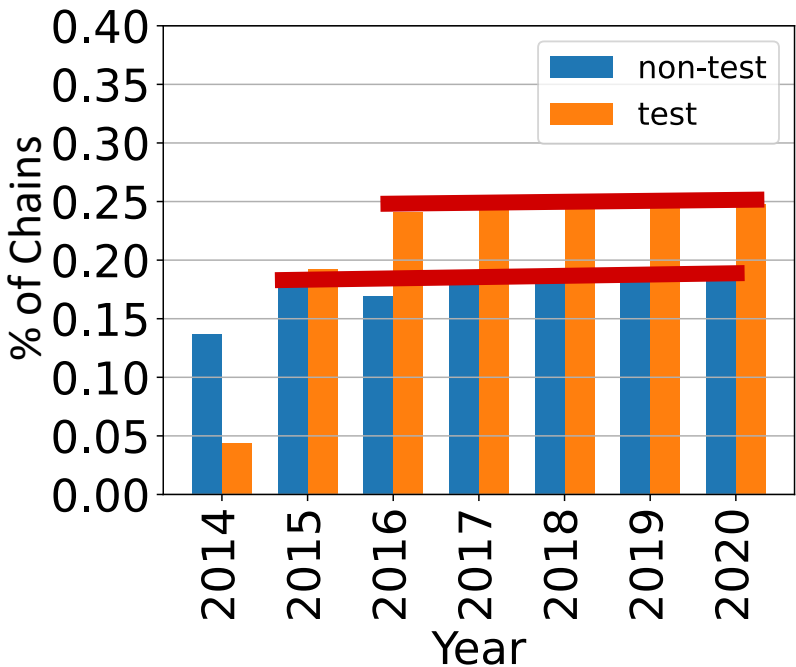
ConditionalExecution

```
if (o .m1 () ) o .m2 ()
```

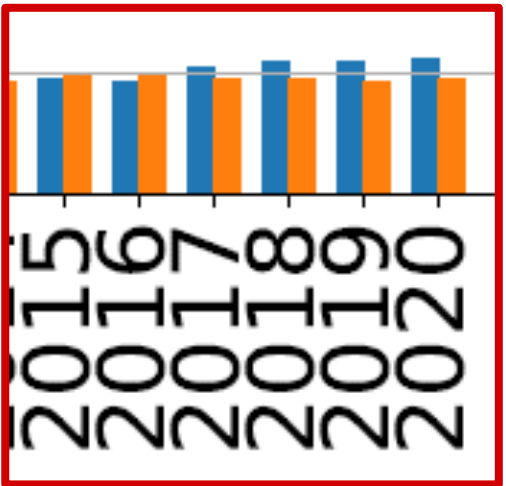
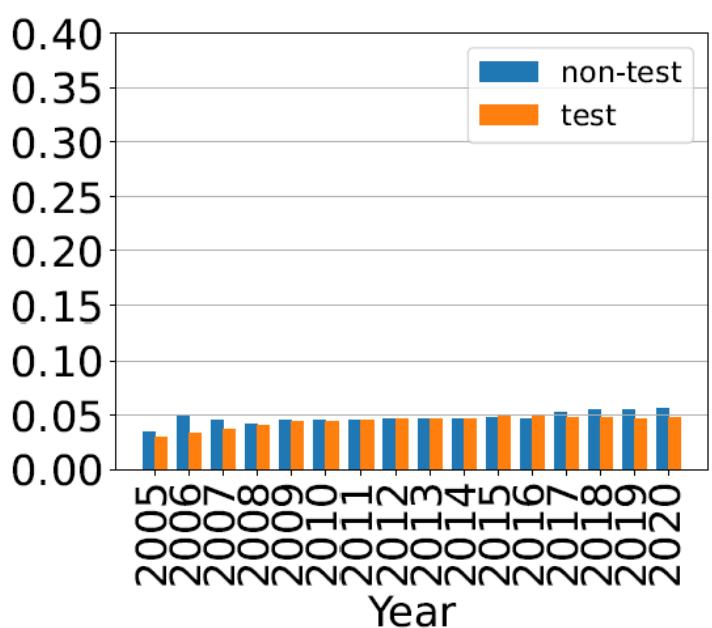
< 0.1% of chains



Kotlin similar to Java, but not growing

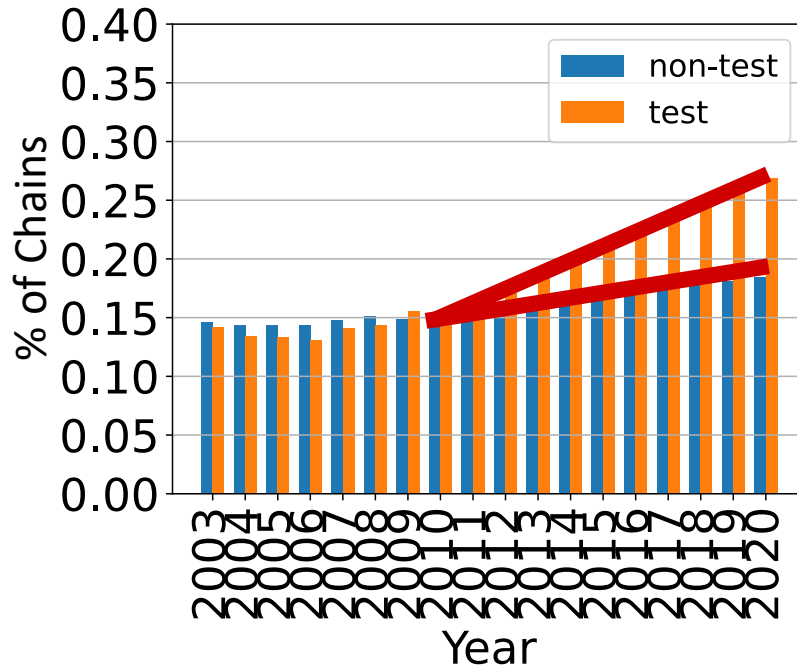


Python overall less use, but more in non-tests...



N

Java results similar to prior study



Language feature support

NullPointerExceptionAvoidance
51.5% of 499k
Kotlin projects

RepeatedReceiver
> 19% of chains

...smaller chains

