ECMAScript Observations

Stages

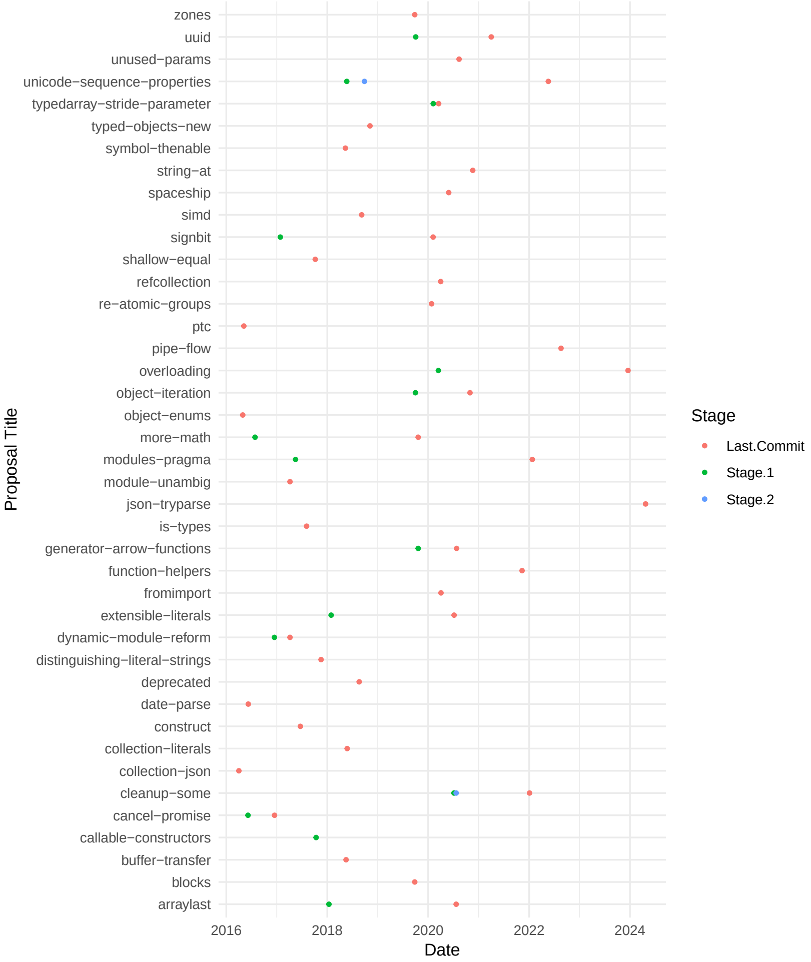
- Stages:
  - Inactive
    - Currently not being worked on.
  - Stage 0
    - This is a new proposal. It is not currently being considered by the committee.
  - Stage 1
    - This proposal is under consideration. The committee expects to devote time to examining the identified problem space, the full breadth of possible solutions, and cross-cutting concerns.
  - Stage 2
    - The committee has chosen a preferred solution or solution space, but the design is a draft and may still change significantly. The committee expects the feature to be developed and eventually included in the standard, but due to reasons that may not be apparent at this stage, the feature may never be included in the standard.
  - Stage 2.7
    - The proposal is approved in principle and undergoing validation. The solution is complete and no further work is possible without feedback from tests, implementations, or usage. No changes to the proposal will be requested by the committee aside from those elicited through testing, implementation, or usage experience.
  - Stage 3
    - The proposal has been recommended for implementation. No changes to the proposal are expected, but some necessary changes may still occur due to web incompatibilities or feedback from production-grade implementations.
  - Stage 4
    - The proposed feature is complete and ready to be included in the standard. No further changes will be made to the proposal.

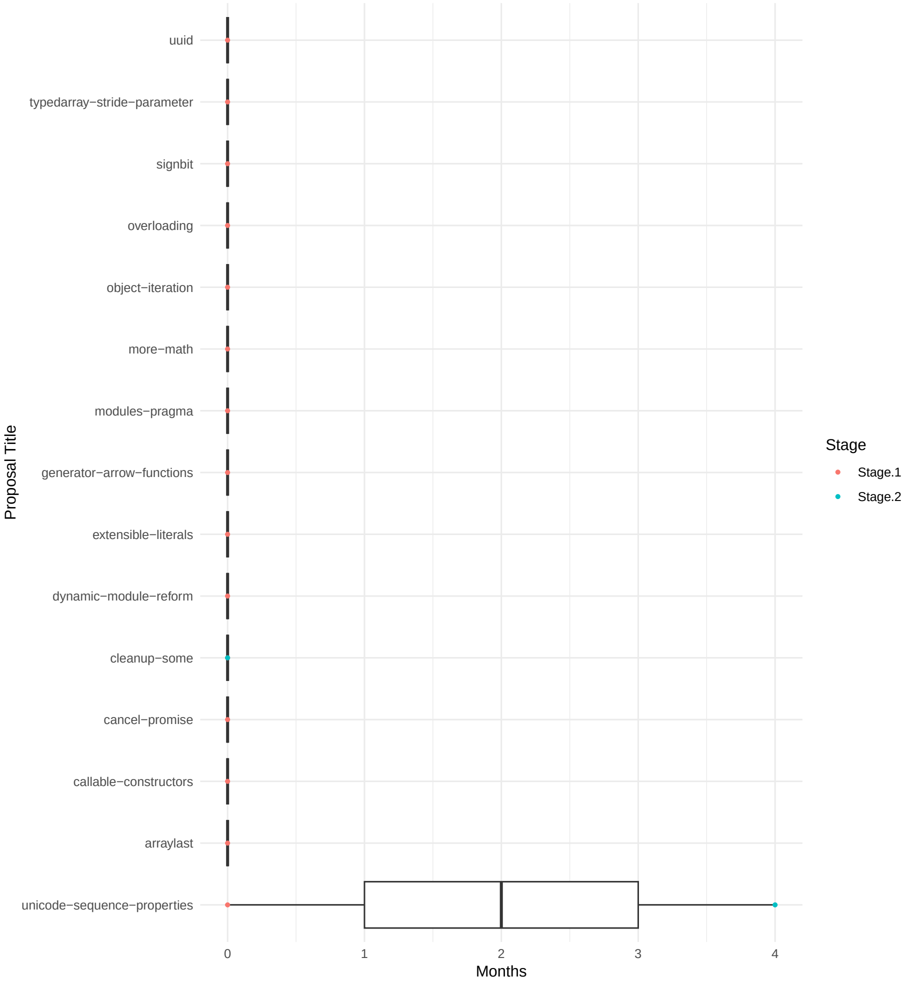
Citation: https://tc39.es/process-document/

Classifications

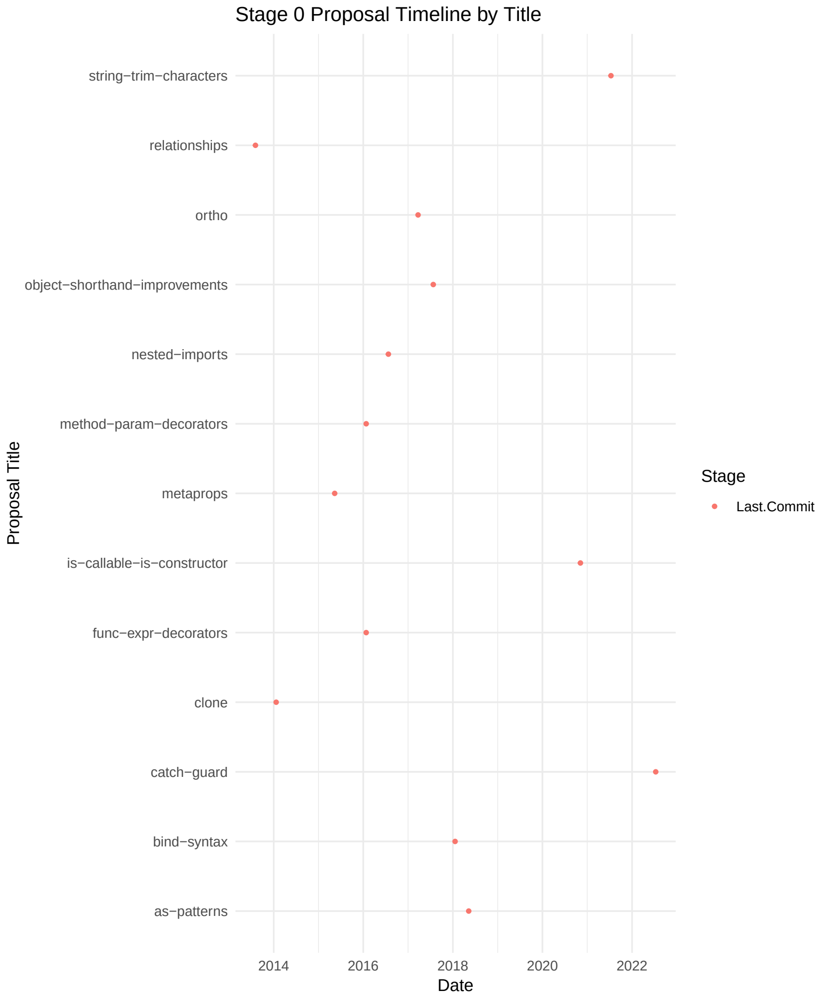
- API Change
  - Modifies or introduces new functions, objects, or methods in the standard library. These changes do not affect the syntax of the language but add new functionality to existing features.
- Semantic Change
  - Changes the meaning of the JavaScript code even if the syntax remains the same. These changes can alter the behavior of existing JavaScript programs in subtle or breaking ways. Usually involves modifying execution rules rather than introducing new syntax.
- Syntactic Change:
  - Introduces new syntax or modifies existing syntax rules. Usually involves new keywords, operators, or expressions. These changes often require updates to parsers and affect how JavaScript code is written.

Inactive

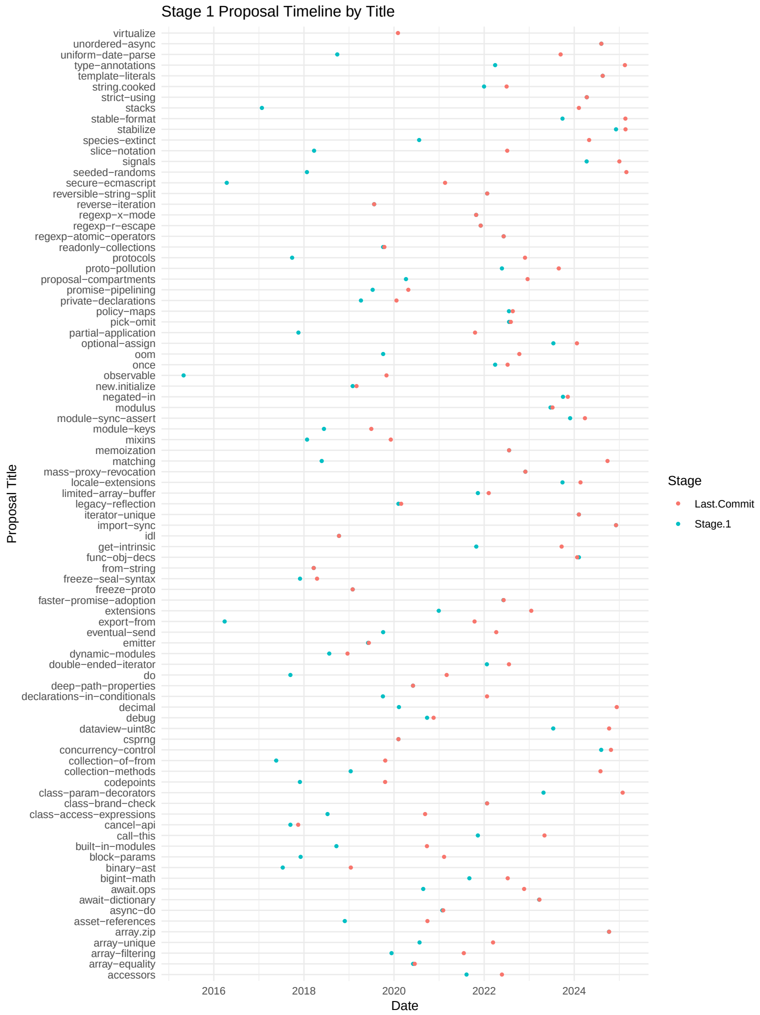

Inactive Proposal Timeline by Title


Inactive Proposals Date Spread per Proposal

Stage 0

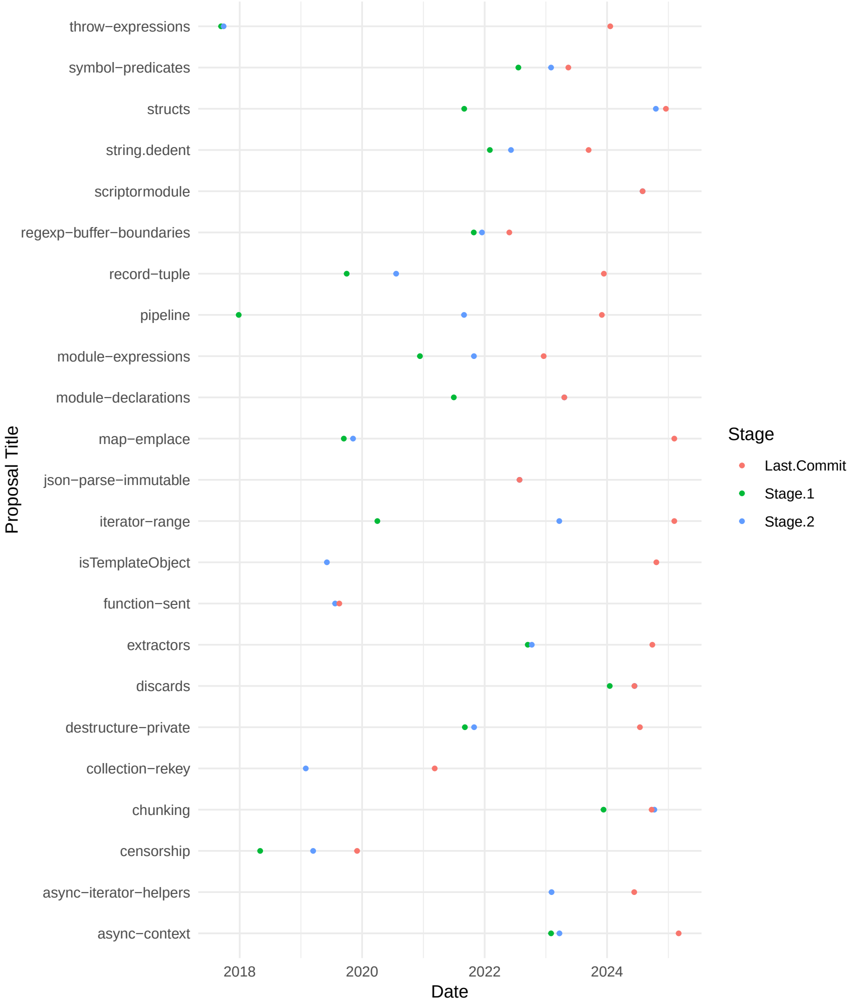


Stage 1

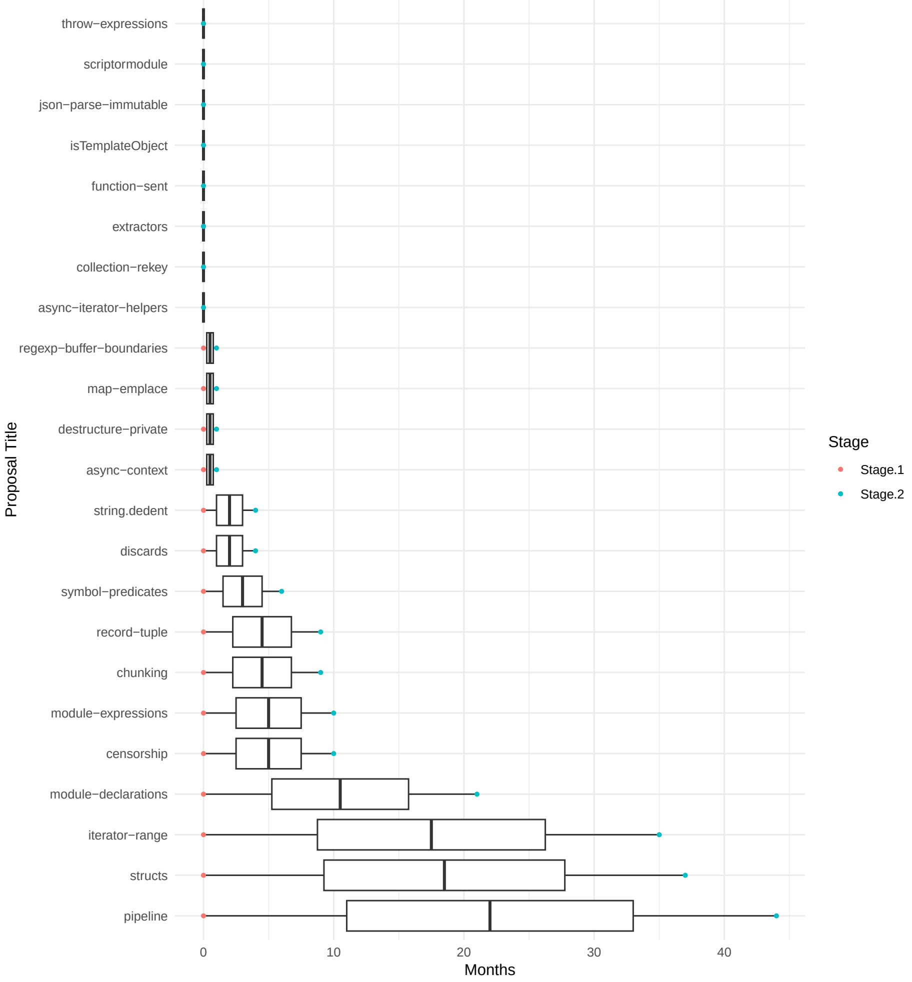


Stage 2

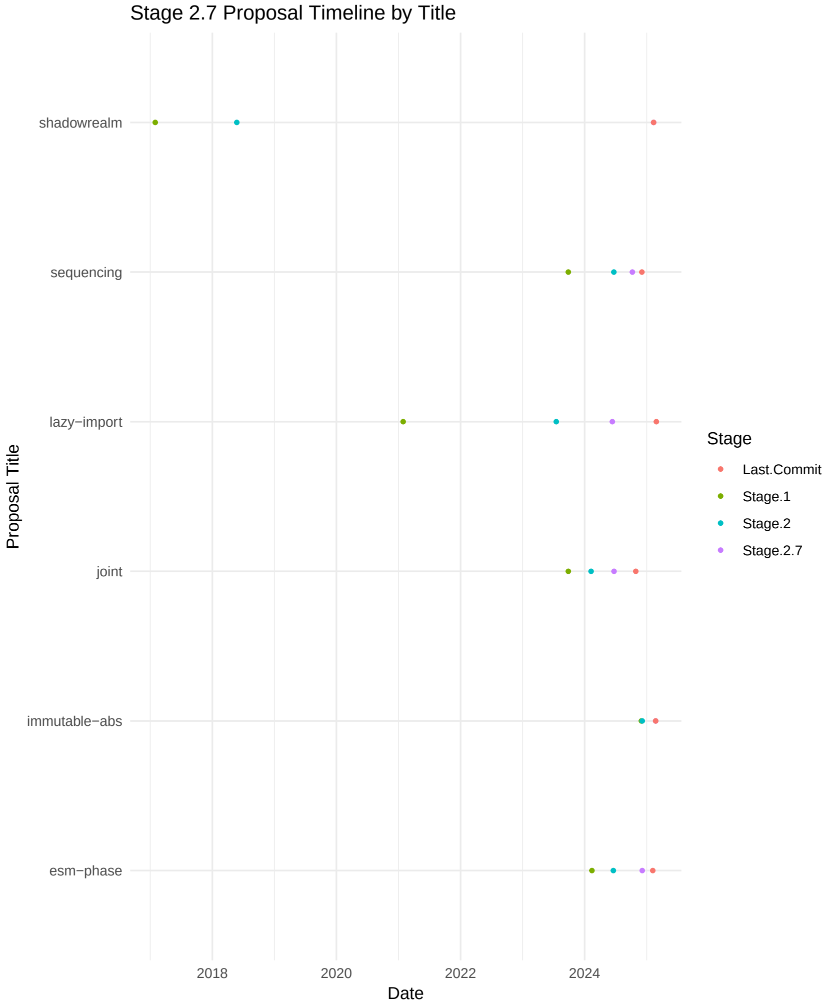
Stage 2 Proposal Timeline by Title

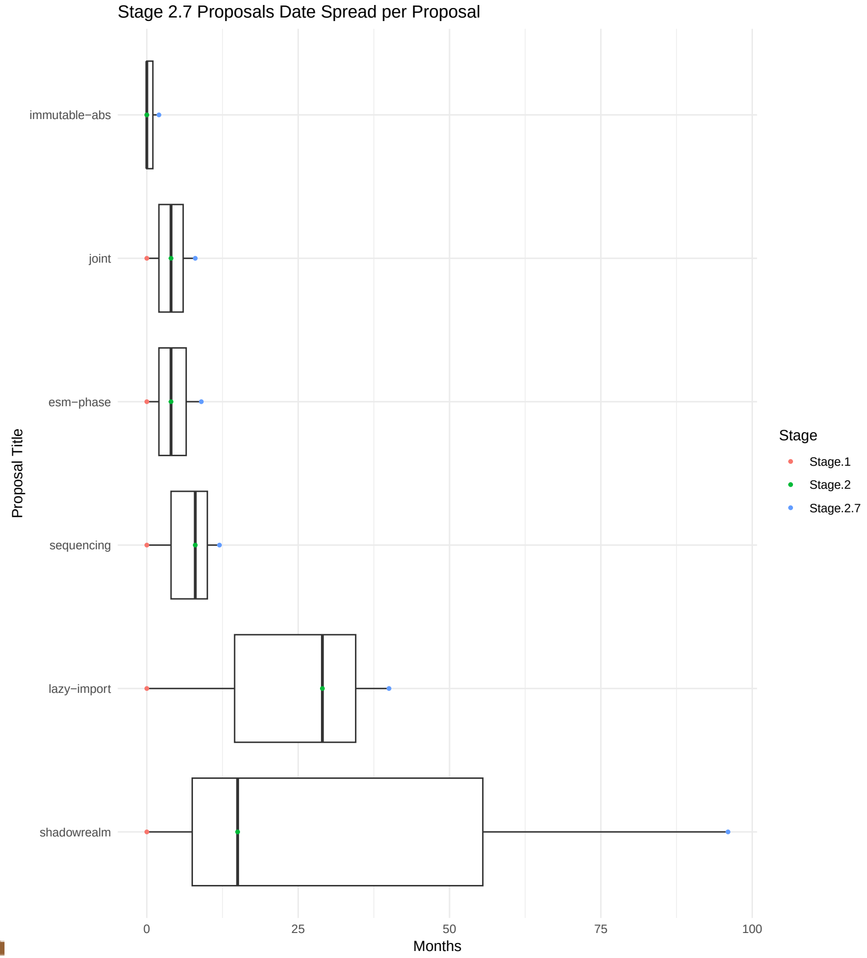


Stage 2 Proposals Date Spread per Proposal



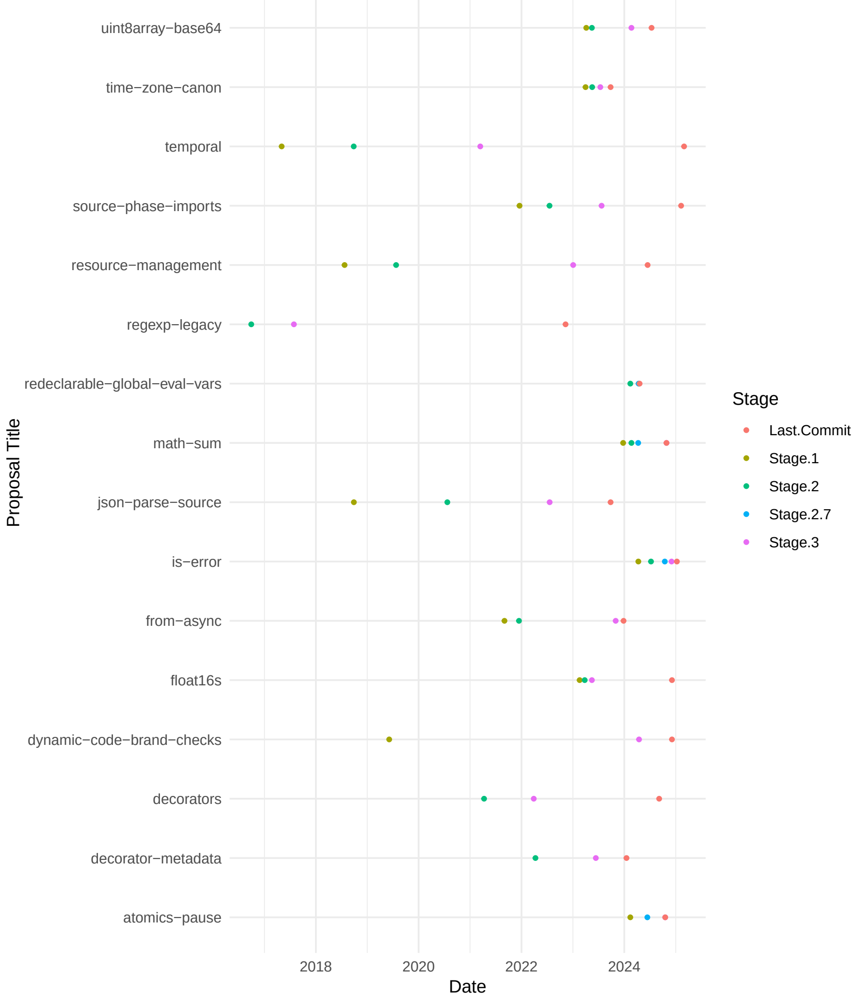
Stage 2.7


Stage 2.7 Proposal Timeline by Title
Proposal Title
shadowrealm
sequencing
lazy−import
joint
immutable−abs
esm−phase
2018
2020
2022
2024
Date
Stage
Last.Commit
Stage.1
Stage.2
Stage.2.7

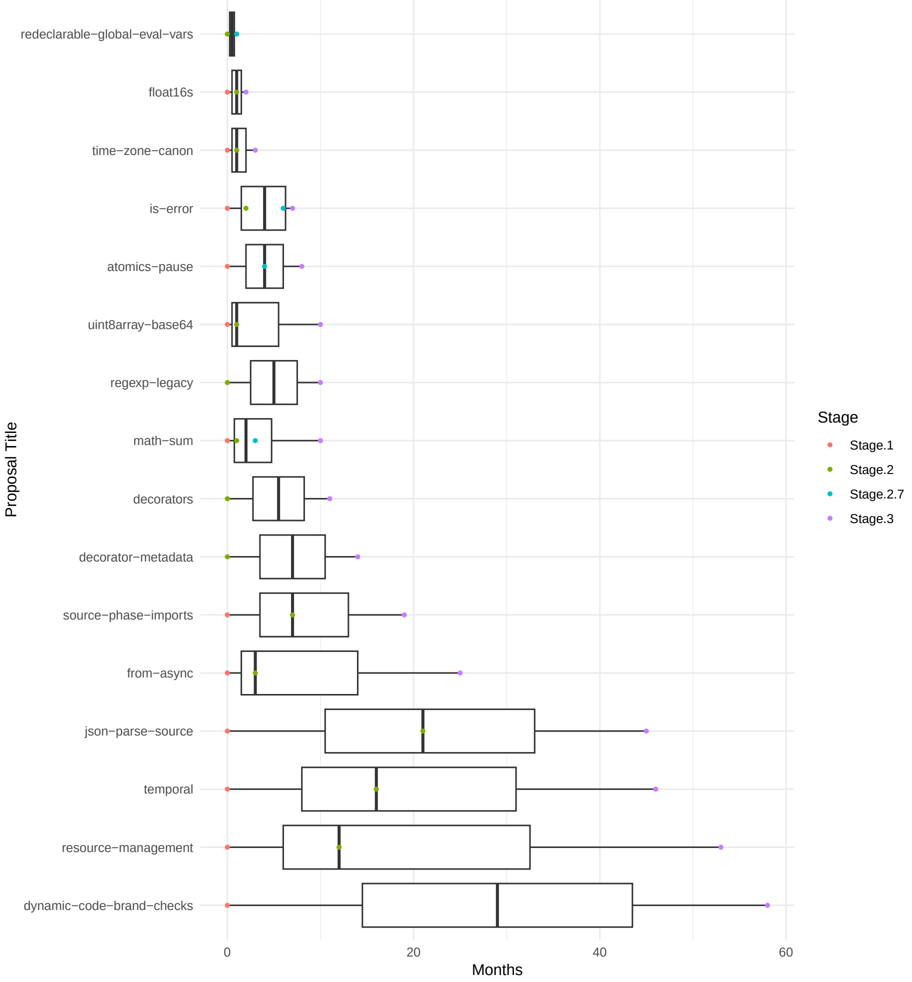

Stage 2.7 Proposals Date Spread per Proposal
Proposal Title
immutable−abs
joint
esm−phase
sequencing
lazy−import
shadowrealm
0
25
50
75
100
Months
Stage
Stage.1
Stage.2
Stage.2.7

Stage 3

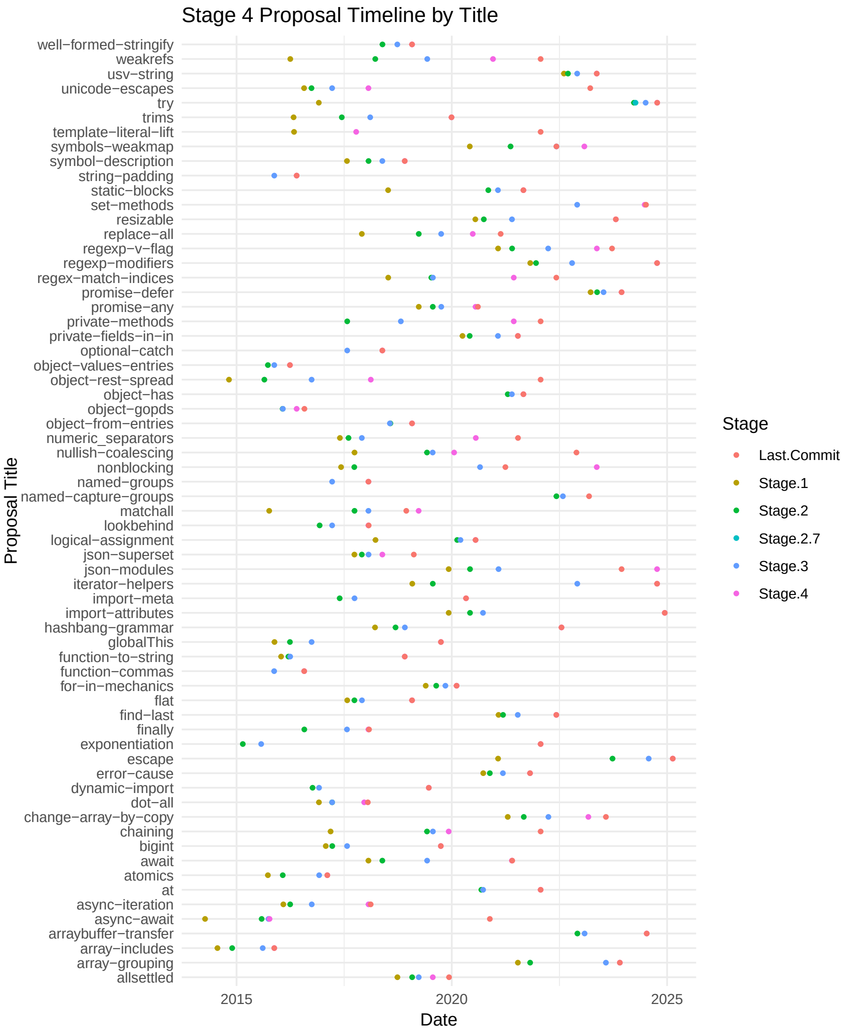
Stage 3 Proposal Timeline by Title

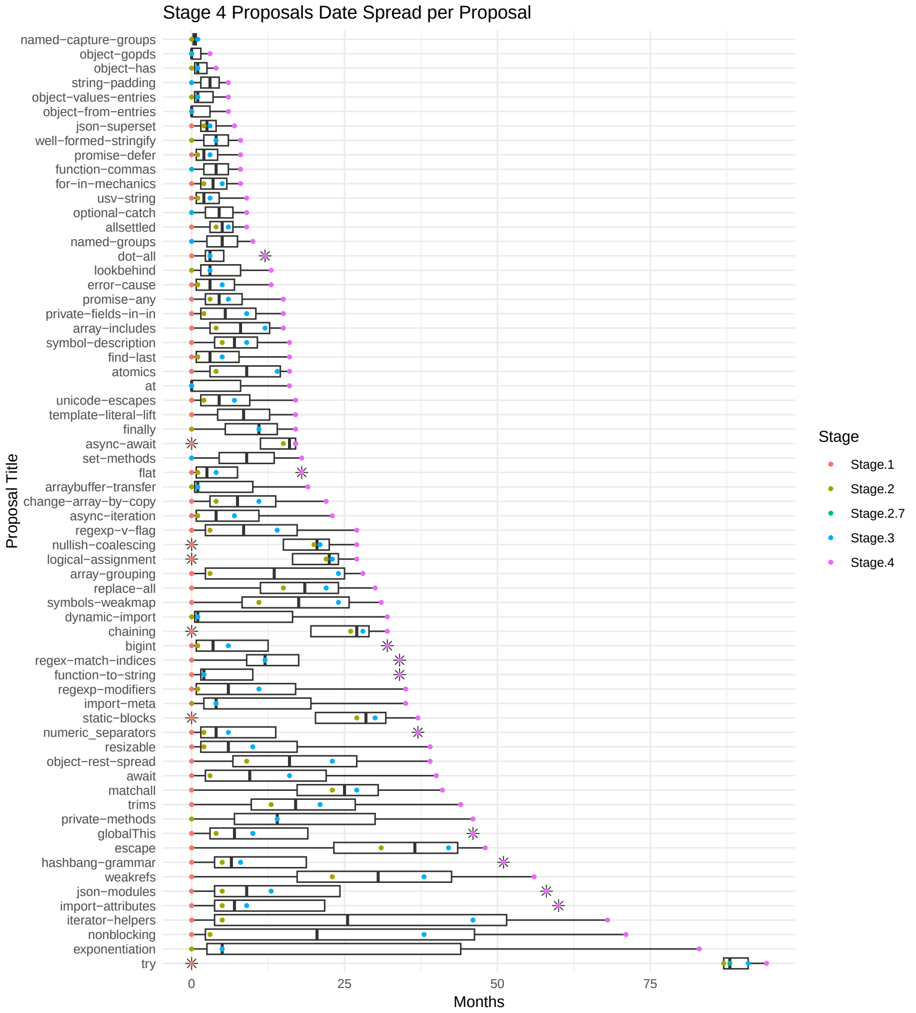


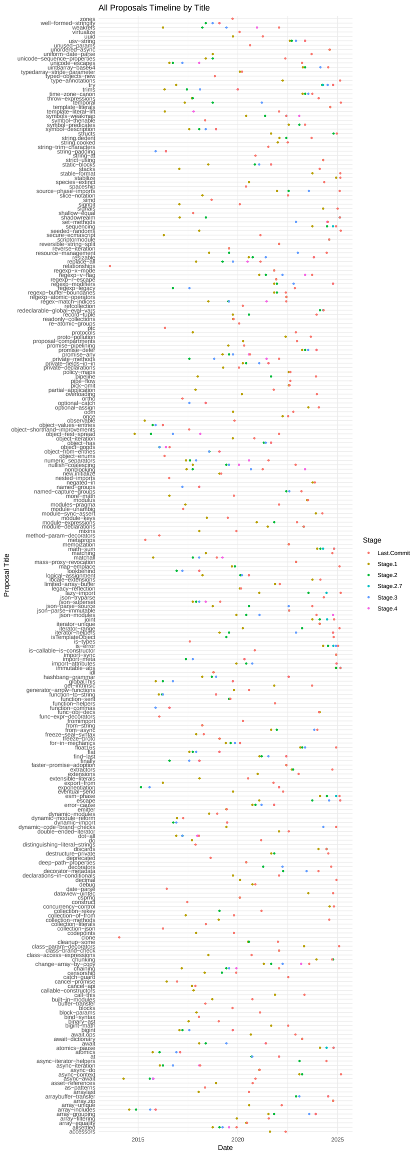
Stage 3 Proposals Date Spread per Proposal



Stage 4

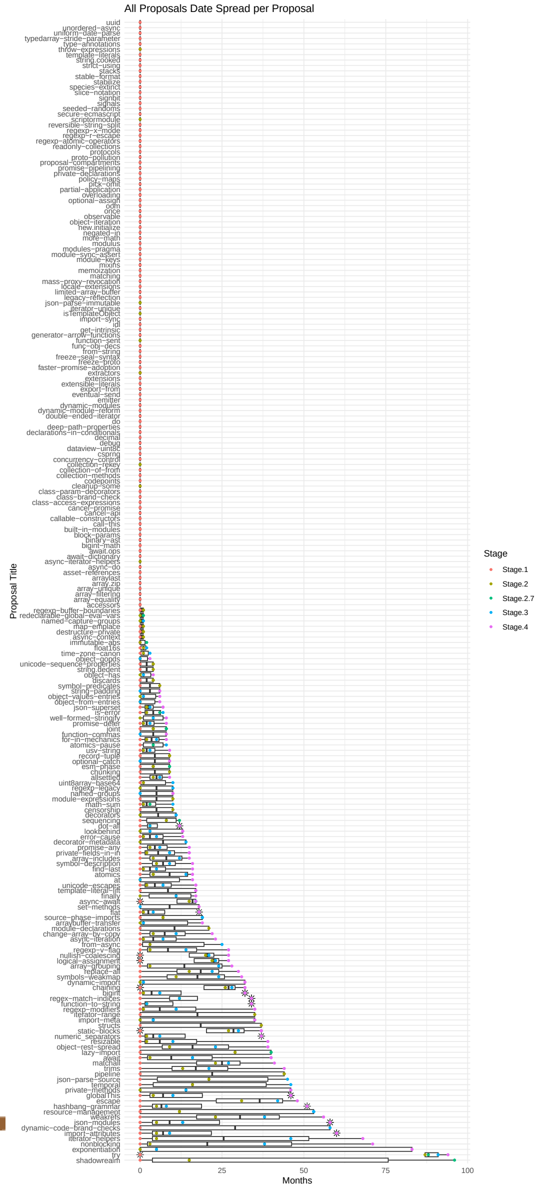

Stage 4 Proposal Timeline by Title


Stage 4 Proposals Date Spread per Proposal



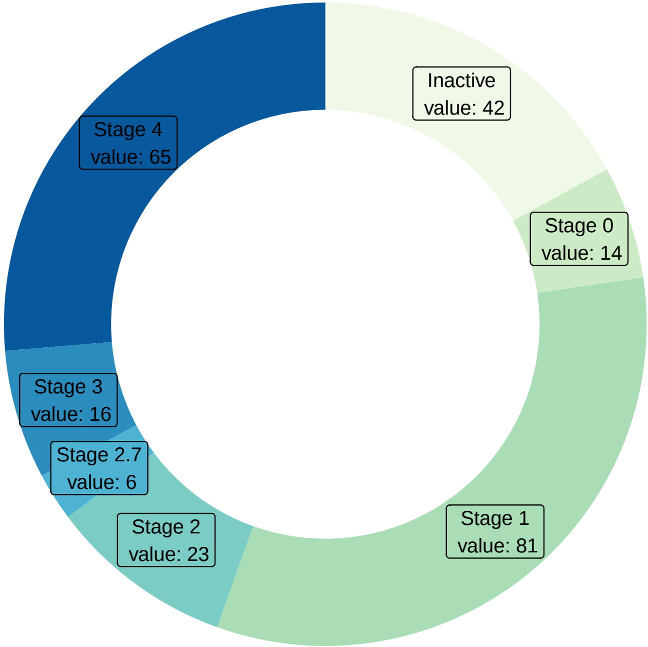
All together

What data can be extracted?
- Classifications
- Stage distribution
- Average duration per stage

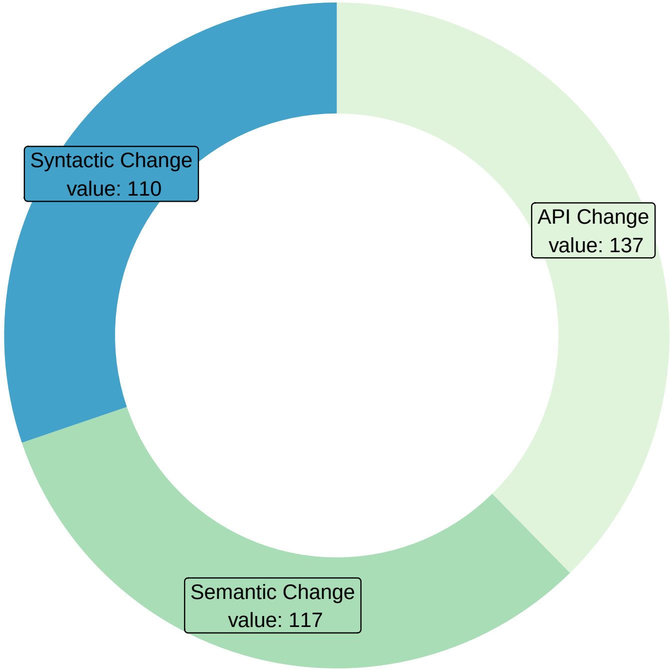


For starters

- Total number of proposals: 257



- Per Classification:



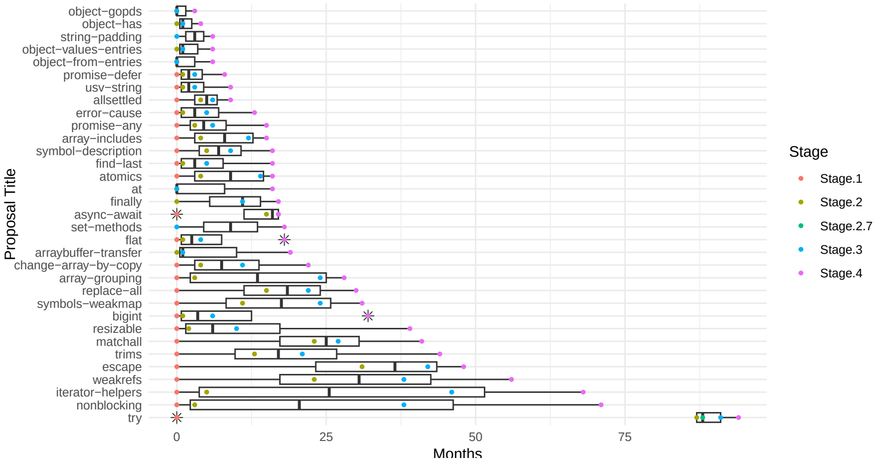
Note: Proposals can overlap classifications

Lets look at Stage 4

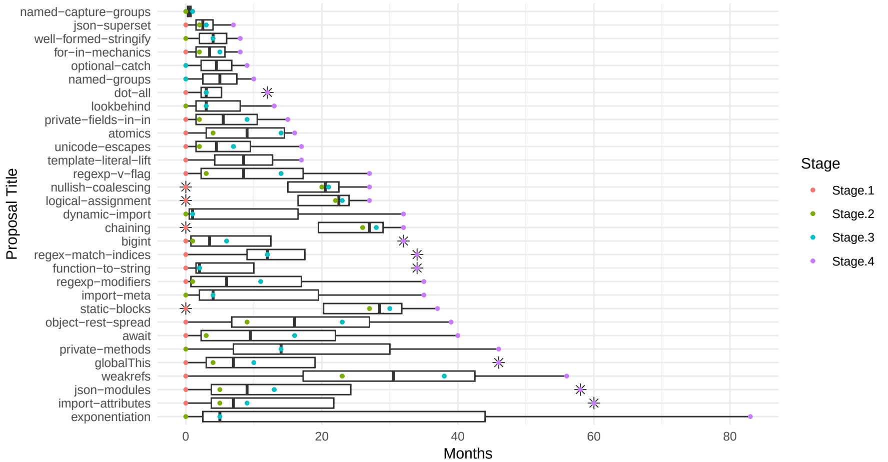
- Most complete data set
- Data gets skewed by the earlier stages

Stage 4: Average Duration per Change

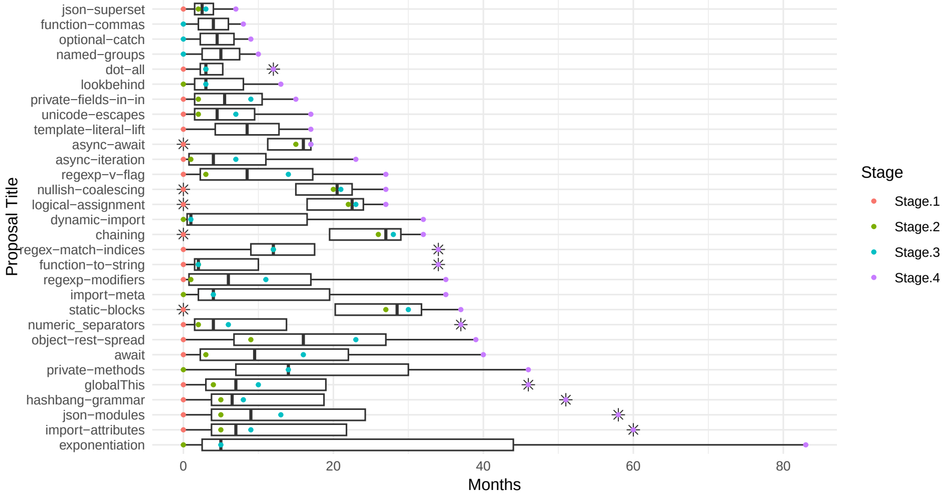
Stage 4 API Changes



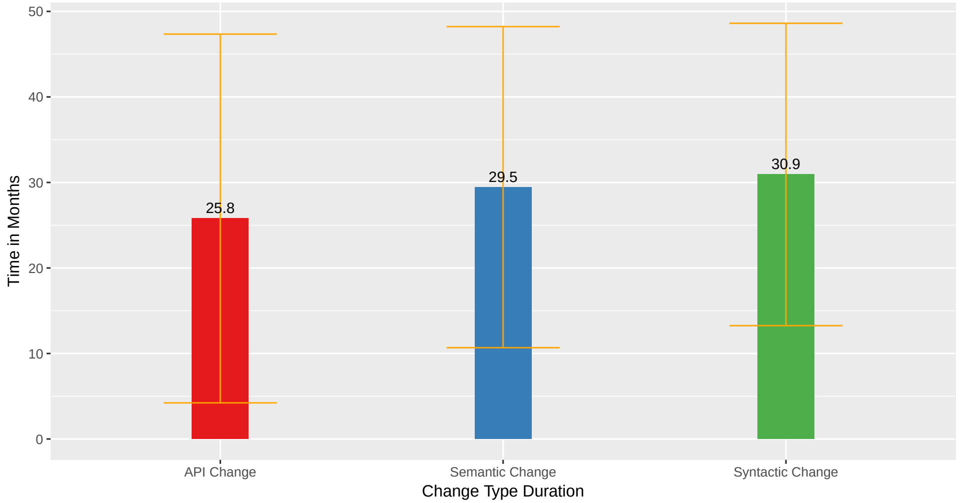
Stage 4 Semantic Changes



Stage 4 Syntactic Changes



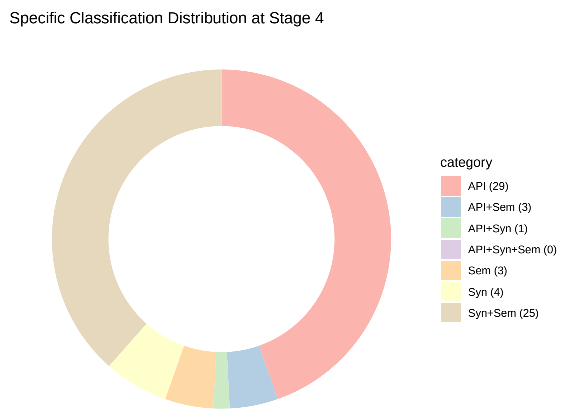
Proposal Duration per Change Type

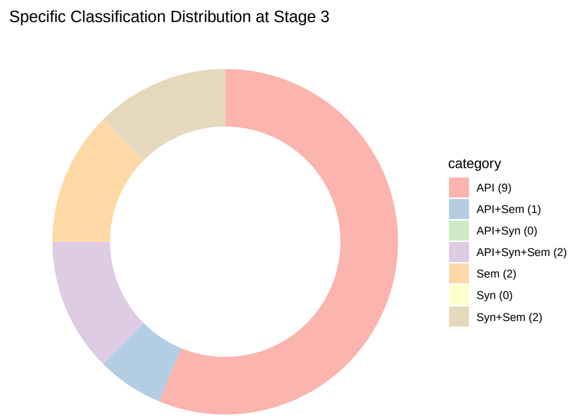


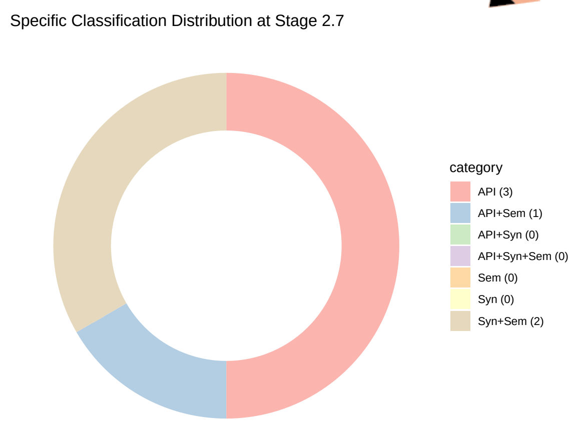
Lets look at more granular classifications

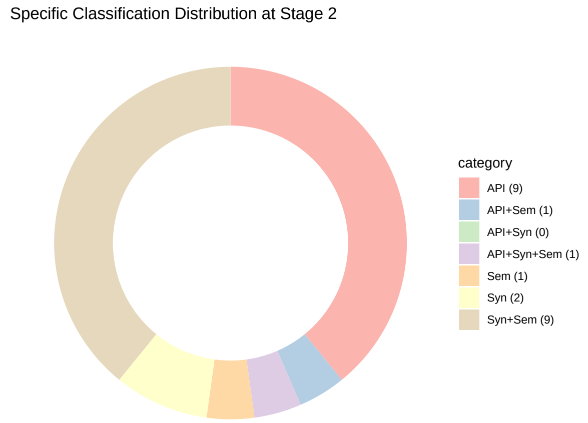
- API only
- Semantic only
- Syntactic only
- API and Semantic
- API and Syntactic
- Semantic and Syntactic
- API and Semantic and Syntactic

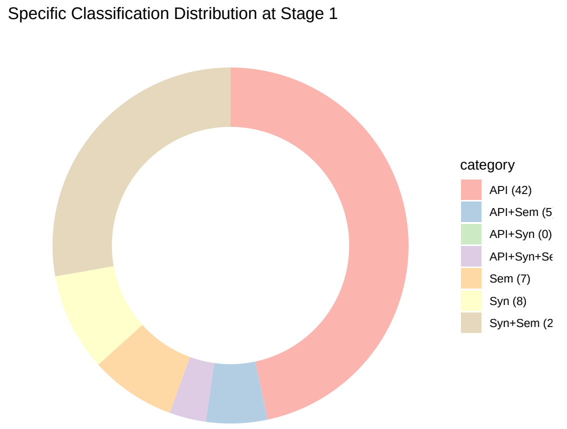
Specific Classifications

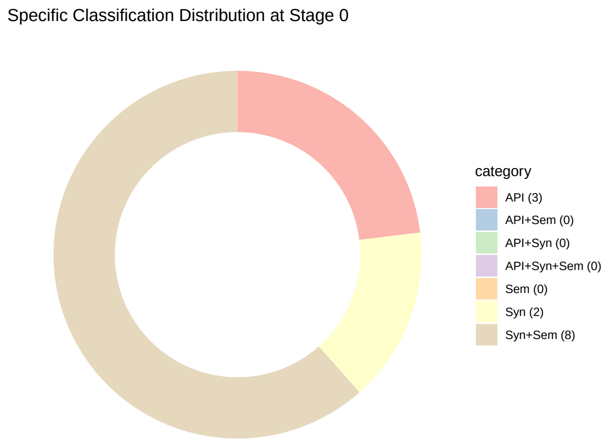

Specific Classification Distribution at Stage 4


Specific Classification Distribution at Stage 3

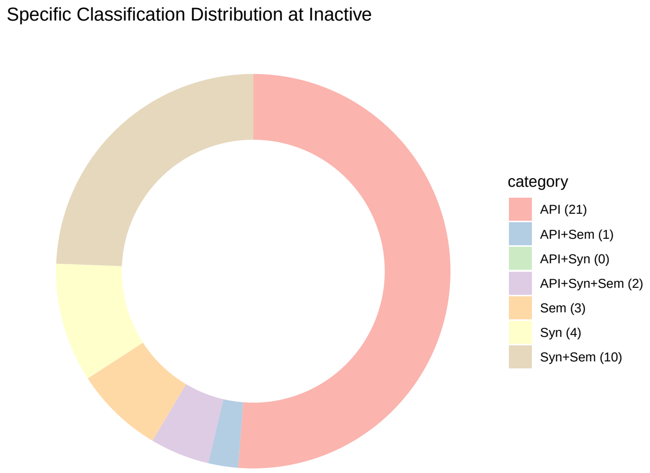

Specific Classification Distribution at Stage 2.7


Specific Classification Distribution at Stage 2


Specific Classification Distribution at Stage 1

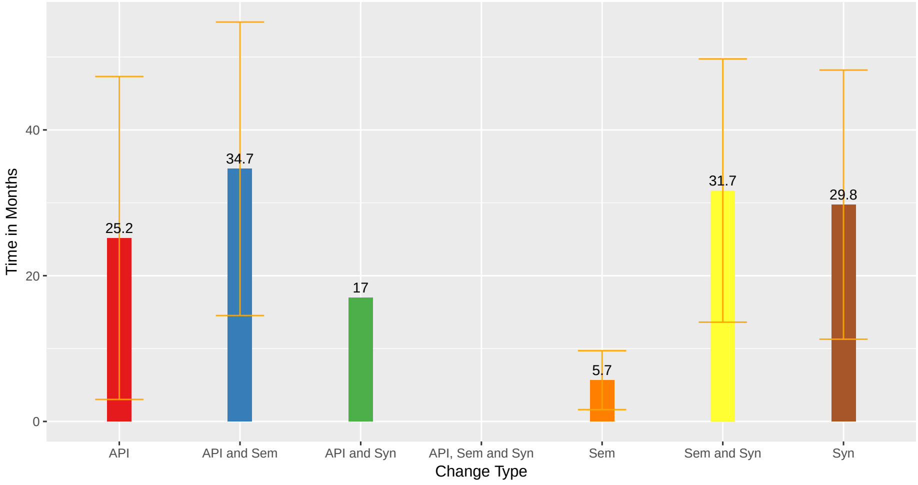

Specific Classification Distribution at Stage 0

Continuing with Inactive

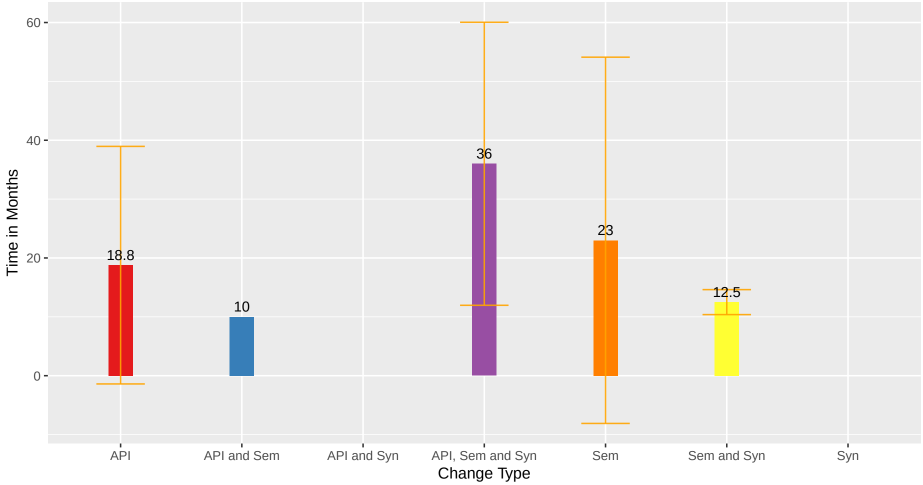


Time From Stage 1

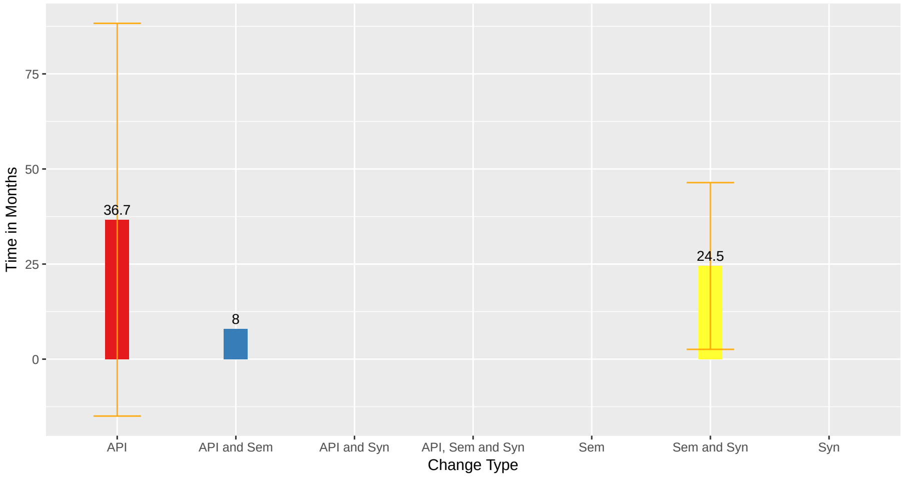
Stage 4: Time from Stage 1 to Stage 4



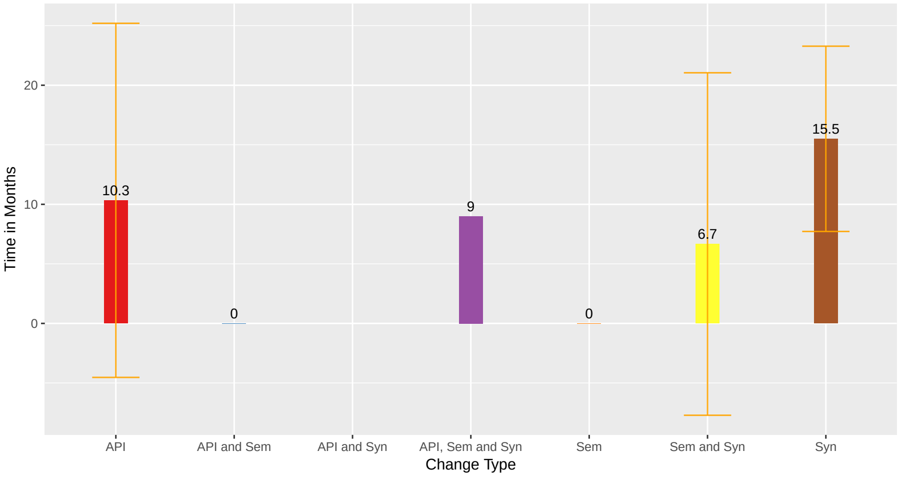
Stage 3: Time from Stage 1 to Stage 3



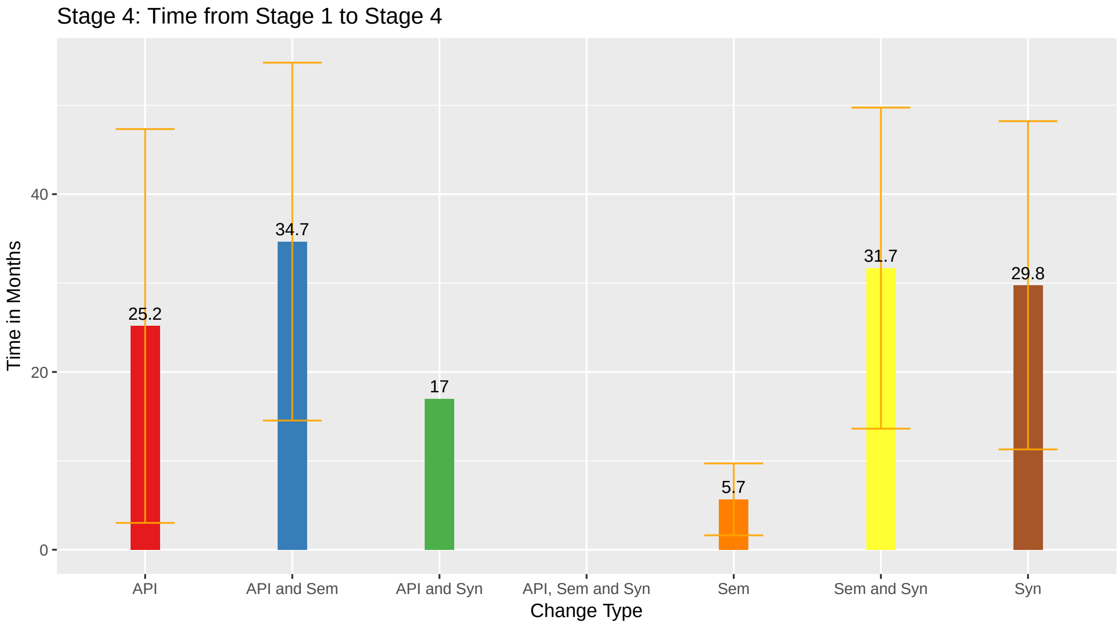
Stage 2.7: Time from Stage 1 to Stage 2.7

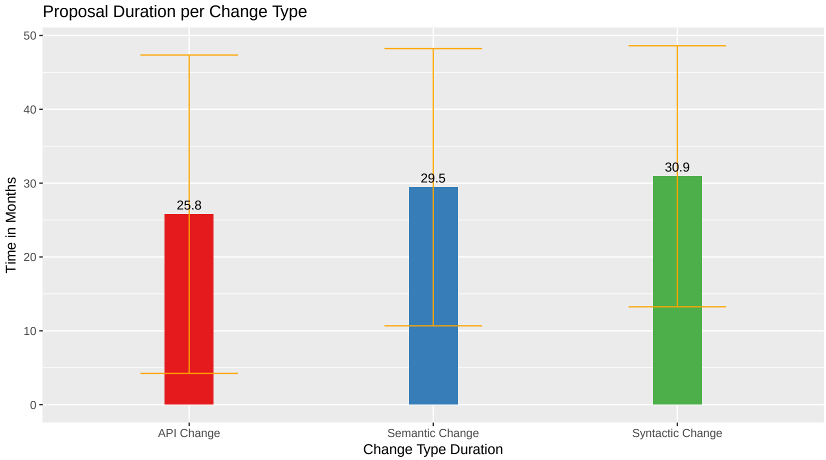


Stage 2: Time from Stage 1 to Stage 2

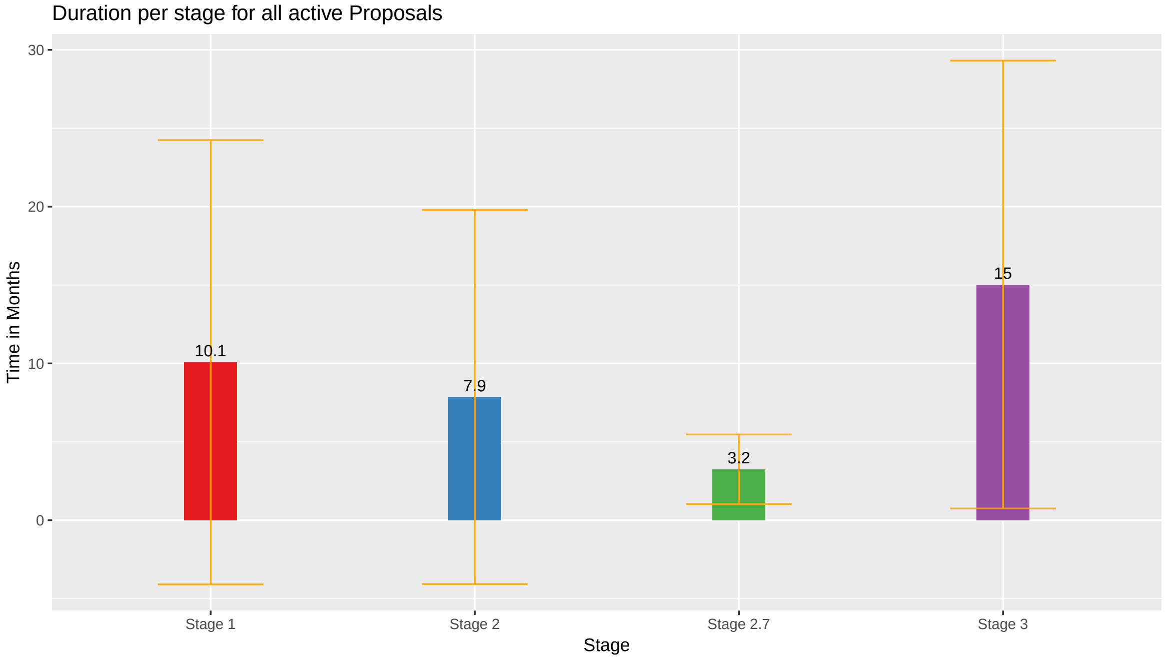


Comparison granular vs overlapping classification





Duration per Stage

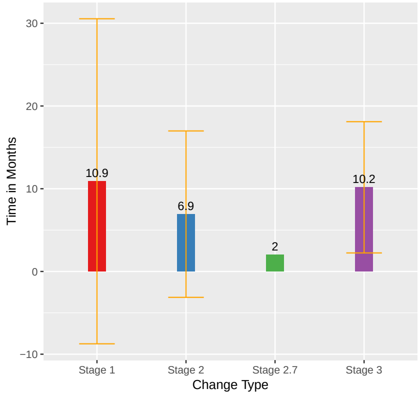


Observations:

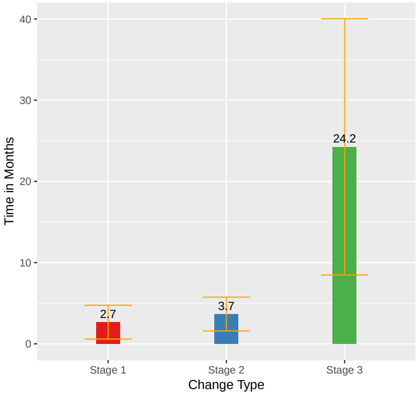
- Length Stage 3 → Stage 1 → Stage 2 → Stage 2.7
- Large SD
- Stage 2.7 is the smallest group

Durations per stage for Stage 4 per classification

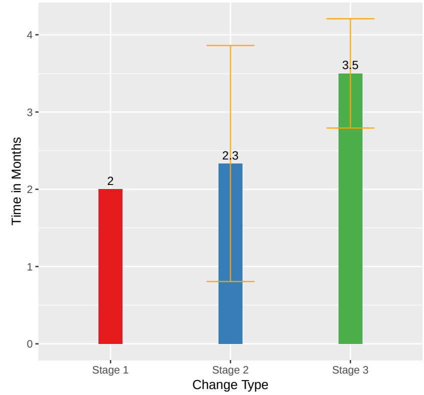
API Change Stage 4 durations with SD



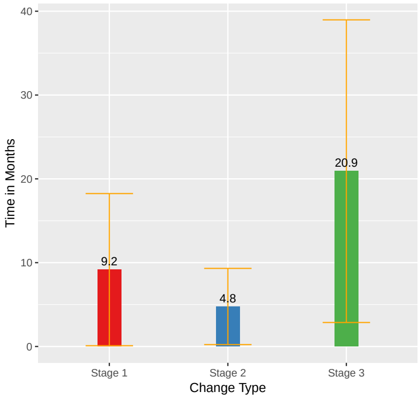
Syn Change Stage 4 durations with SD



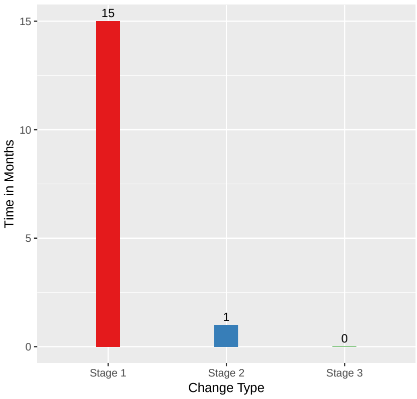
Sem Change Stage 4 durations with SD



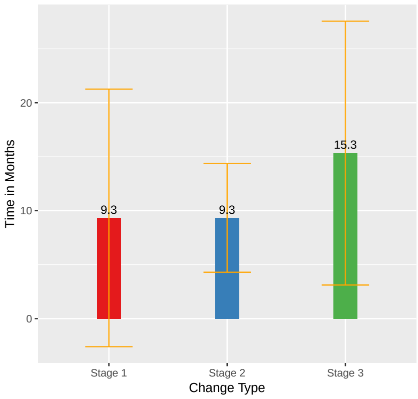
Syn+Sem Change Stage 4 durations with SD



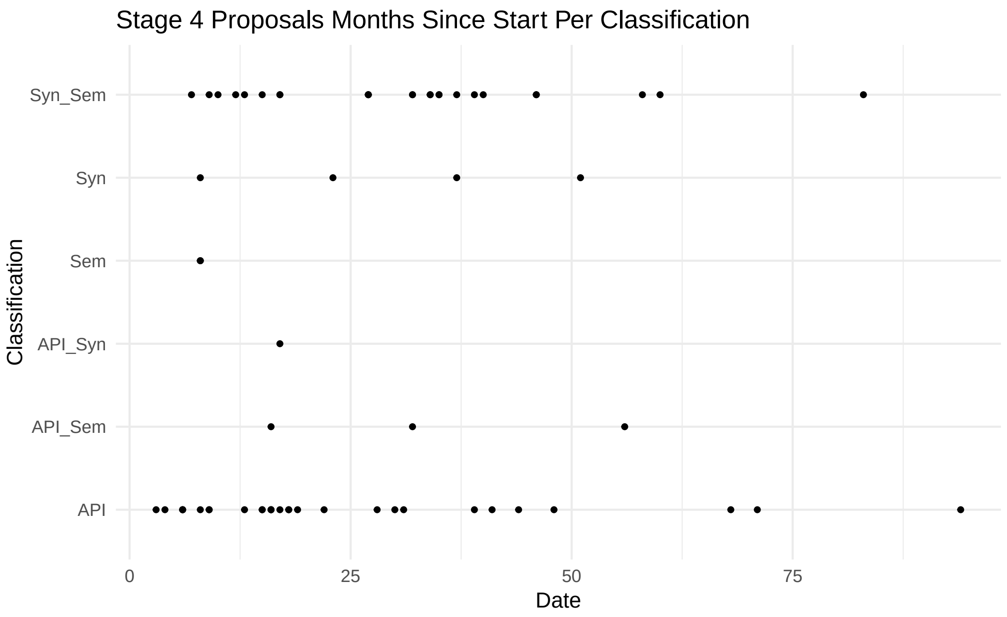
API+Syn Change Stage 4 durations with SD

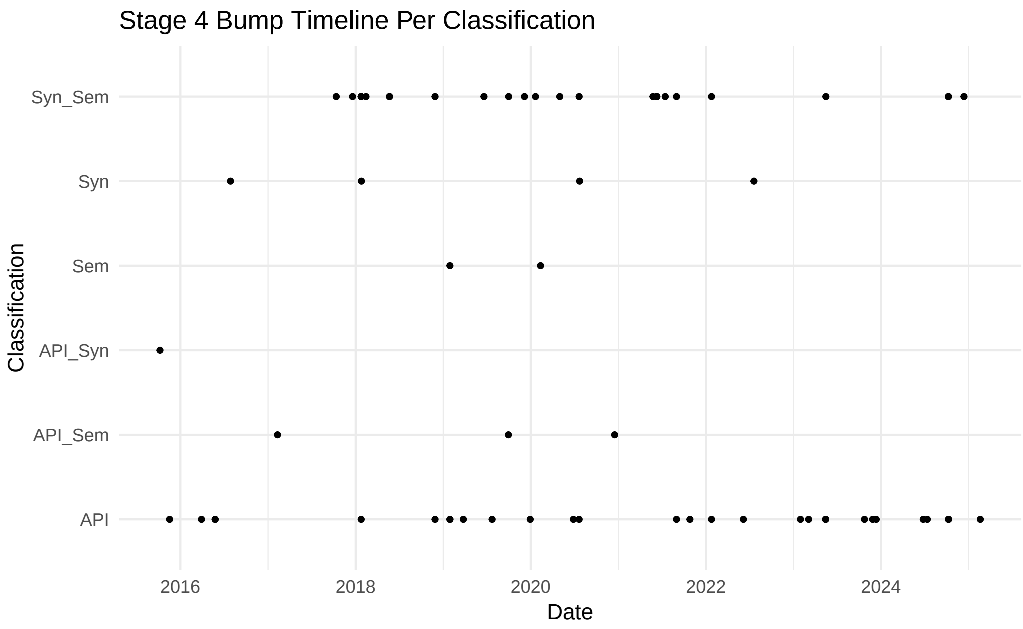


API+Sem Change Stage 4 durations with SD



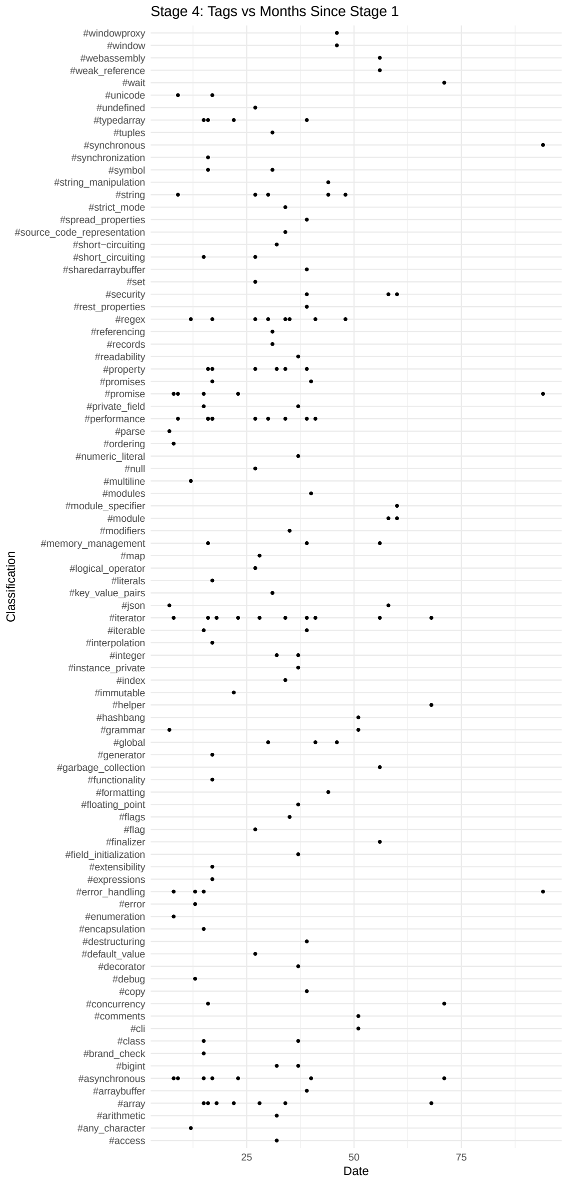
Stage 4 Proposals per Classifications



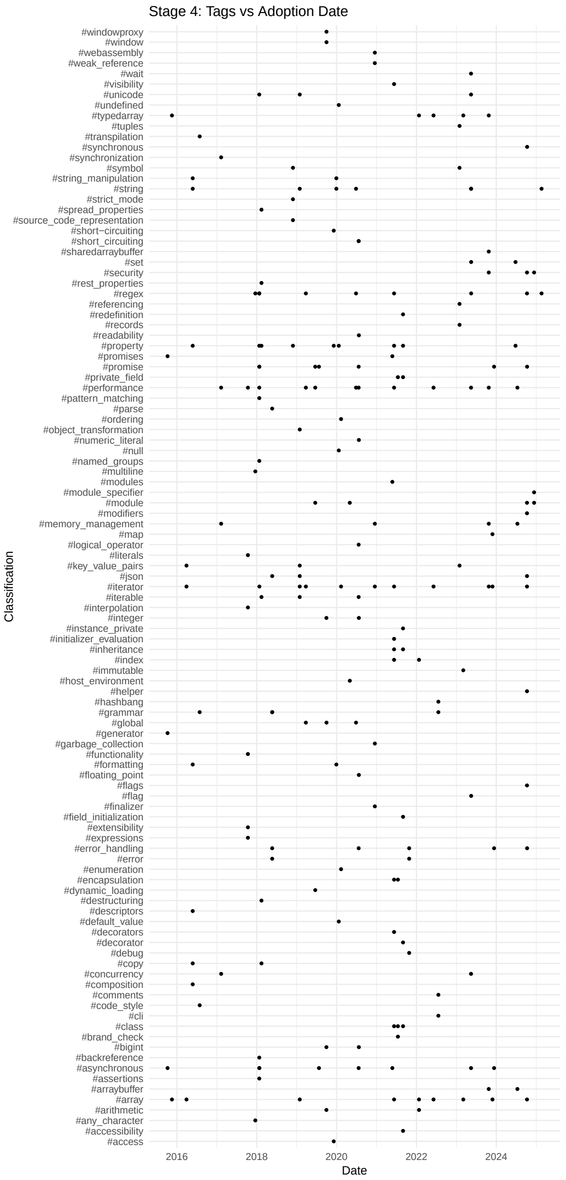


326 Tags

Rank	Tag	n
1	#performance	32
2	#iterator	26
3	#asynchronous	22
4	#promise	22
5	#module	21
6	#regex	20
7	#array	19
8	#property	19
9	#class	18
10	#security	18
11	#string	17
12	#error_handling	15
13	#memory_management	15
14	#typedarray	12
15	#concurrency	10
16	#arithmetic	9
17	#destructuring	8
18	#map	8
19	#numeric	8
20	#arraybuffer	7
21	#decorator	7
22	#json	7
23	#math	7
24	#unicode	7
25	#bigint	6
26	#generator	6
27	#global	6
28	#iterable	6
29	#key_value_pairs	6
30	#parse	6
31	#realm	6
32	#symbol	6
33	#date_time	5
34	#encapsulation	5
35	#grammar	5
36	#metadata	5
37	#operator	5
38	#pattern_matching	5
39	#readability	5
40	#resource_management	5
41	#set	5
42	#string_manipulation	5
43	#synchronous	5
44	#wait	5
45	#accessor	4



Tags Continued


Stage 4: Tags vs Adoption Date
Classification
#windowproxy
#window
#webassembly
#weak_reference
#wait
#visibility
#unicode
#undefined
#typedarray
#tuples
#transpilation
#synchronous
#synchronization
#symbol
#string_manipulation
#string
#strict_mode
#spread_properties
#source_code_representation
#short−circuiting
#short_circuiting
#sharedarraybuffer
#set
#security
#rest_properties
#regex
#referencing
#redefinition
#records
#readability
#property
#promises
#promise
#private_field
#performance
#pattern_matching
#parse
#ordering
#object_transformation
#numeric_literal
#null
#named_groups
#multiline
#modules
#module_specifier
#module
#modifiers
#memory_management
#map
#logical_operator
#literals
#key_value_pairs
#json
#iterator
#iterable
#interpolation
#integer
#instance_private
#initializer_evaluation
#inheritance
#index
#immutable
#host_environment
#helper
#hashbang
#grammar
#global
#generator
#garbage_collection
#functionality
#formatting
#floating_point
#flags
#flag
#finalizer
#field_initialization
#extensibility
#expressions
#error_handling
#error
#enumeration
#encapsulation
#dynamic_loading
#destructuring
#descriptors
#default_value
#decorators
#decorator
#debug
#copy
#concurrency
#composition
#comments
#code_style
#cli
#class
#brand_check
#bigint
#backreference
#asynchronous
#assertions
#arraybuffer
#array
#arithmetic
#any_character
#accessibility
#access
Date