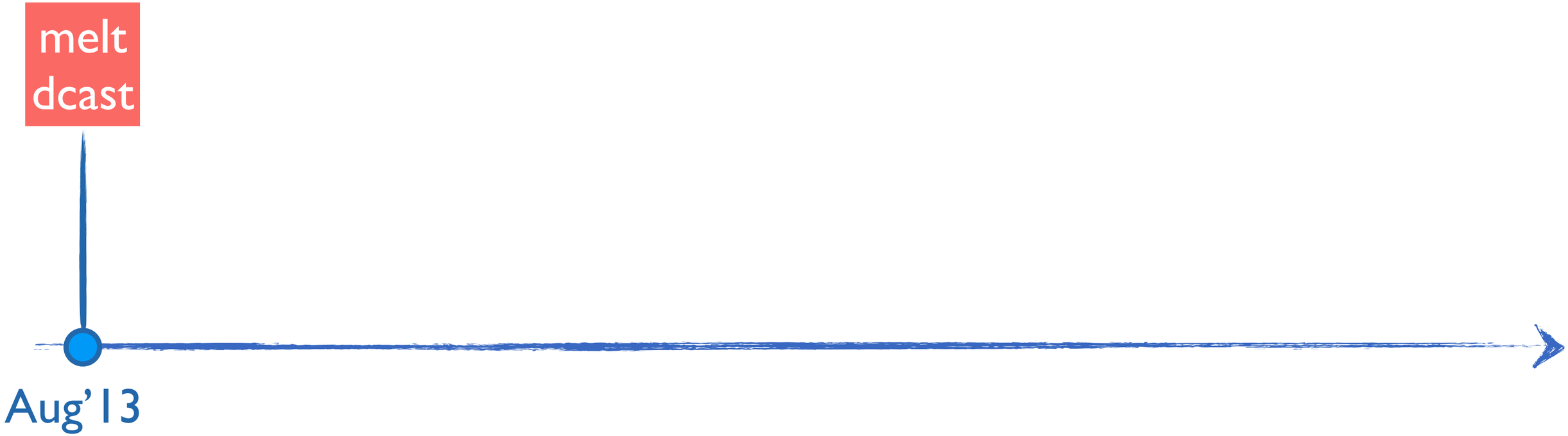
Arun Srinivasan


@arun_sriniv

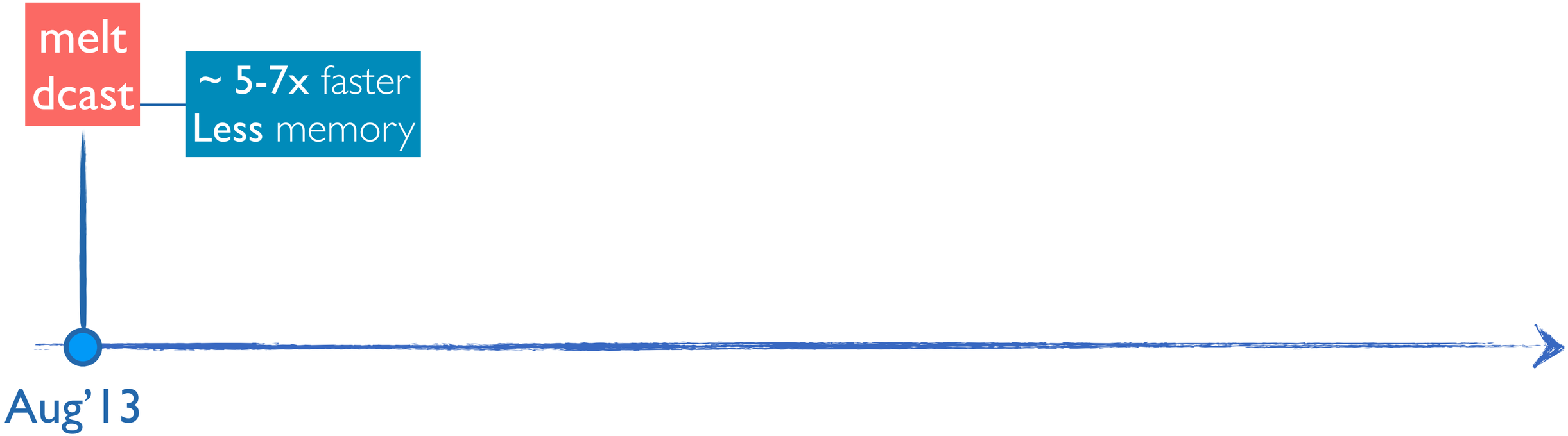
Phd student, MPIPZ, Germany

co-developer, data.table

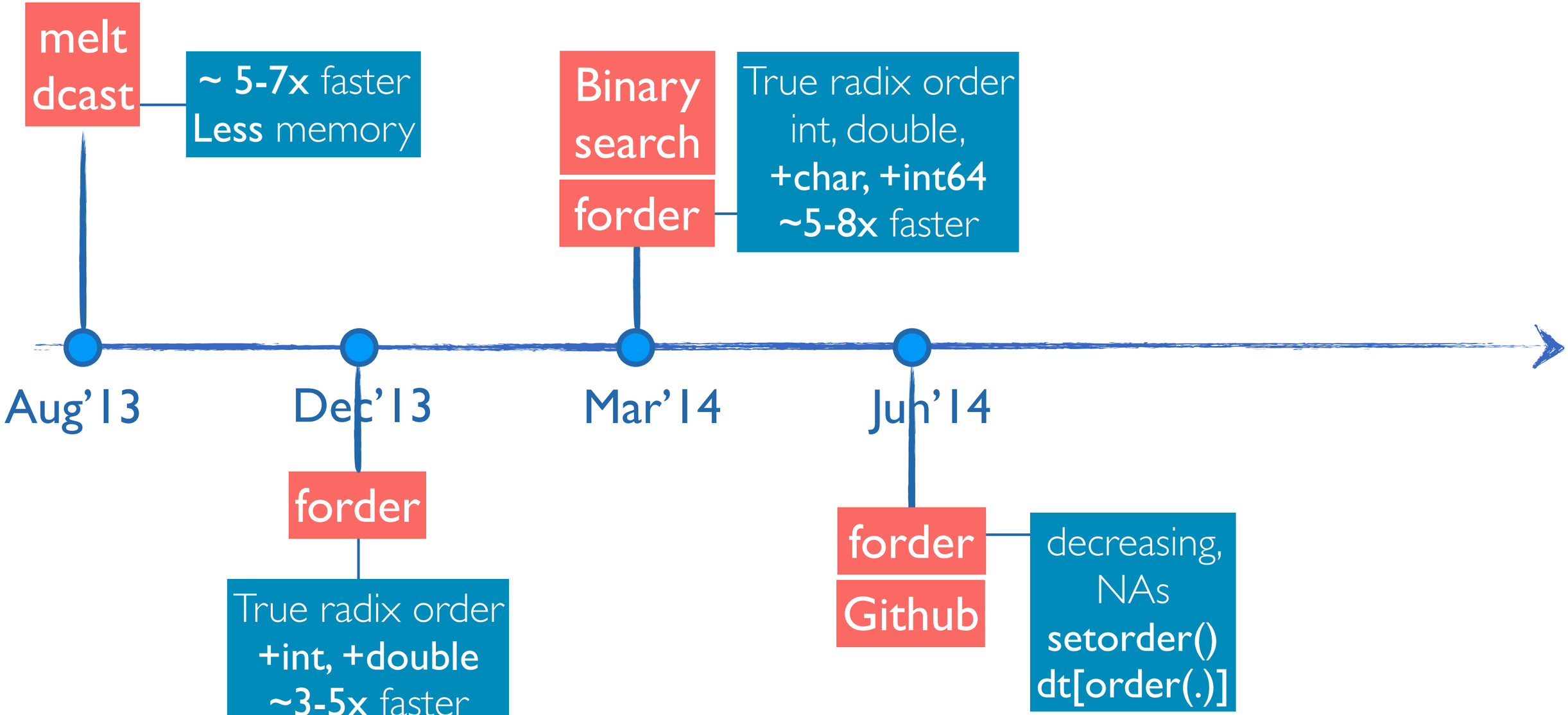
TIMELINE



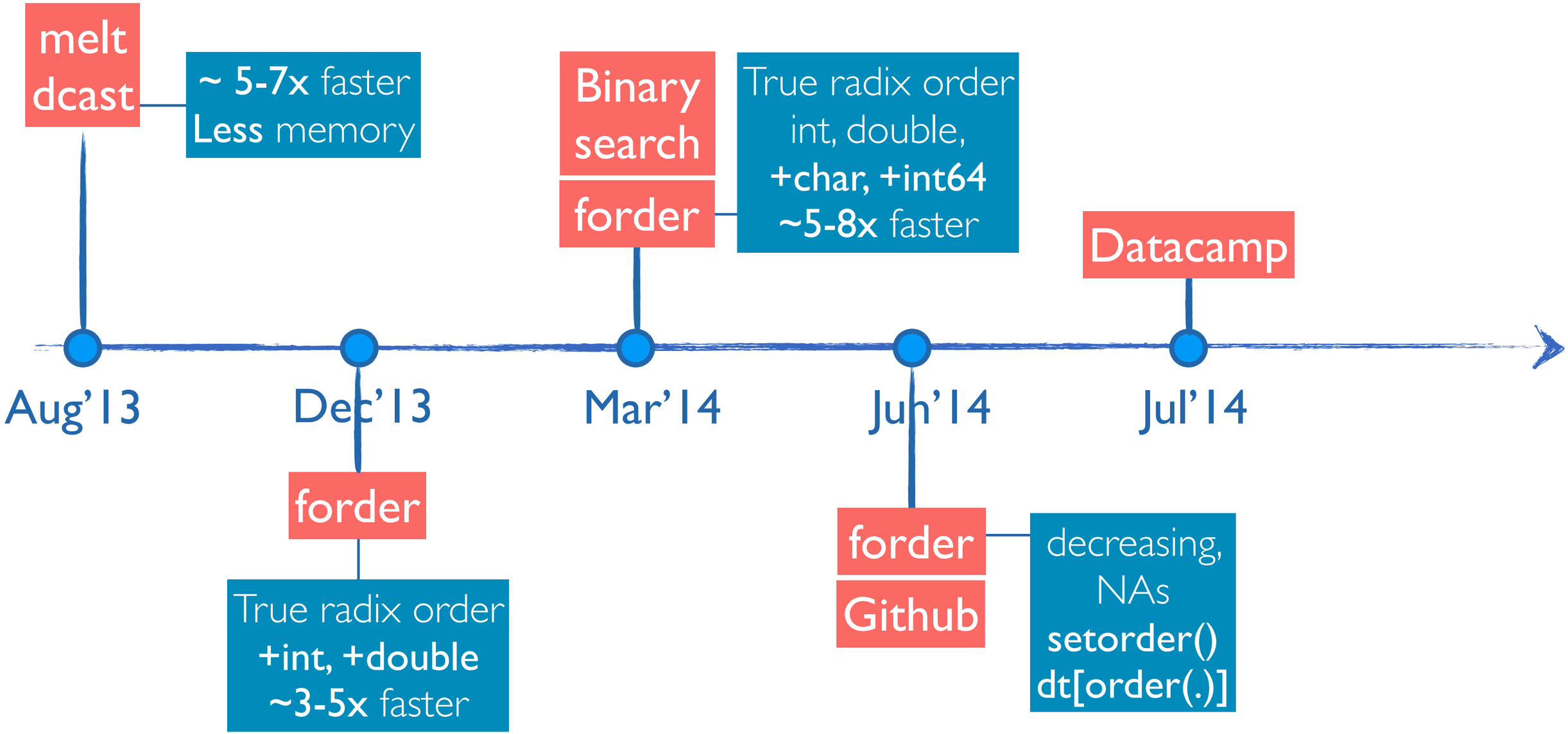
TIMELINE



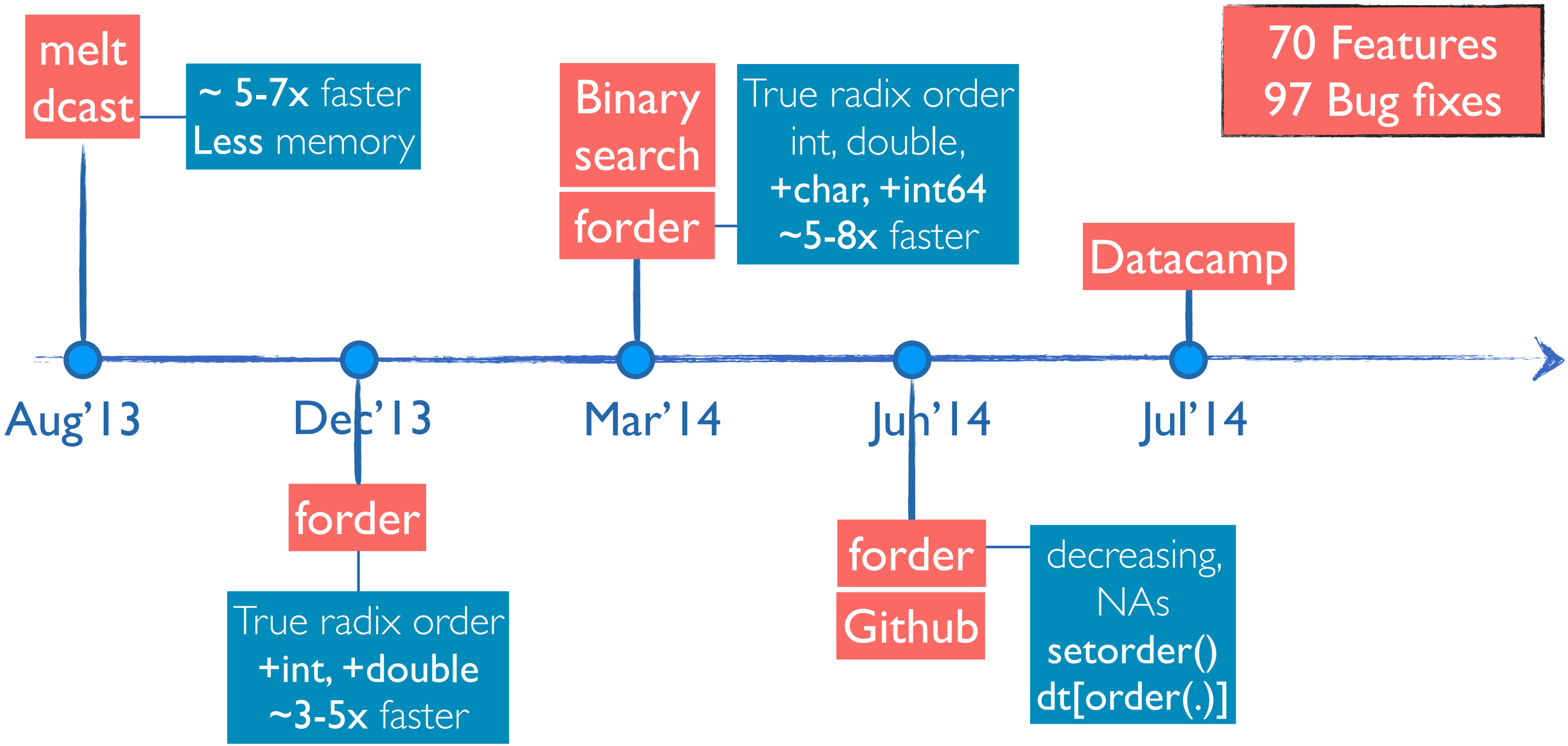
TIMELINE


melt
dcast
~ 5-7x faster
Less memory
Binary search
forder
True radix order
int, double,
+char, +int64
~5-8x faster
Aug'13
Dec'13
Mar'14
Jun'14
forder
True radix order
+int, +double
~3-5x faster
forder
Github
decreasing,
NAs
setorder()
dt[order(.)]

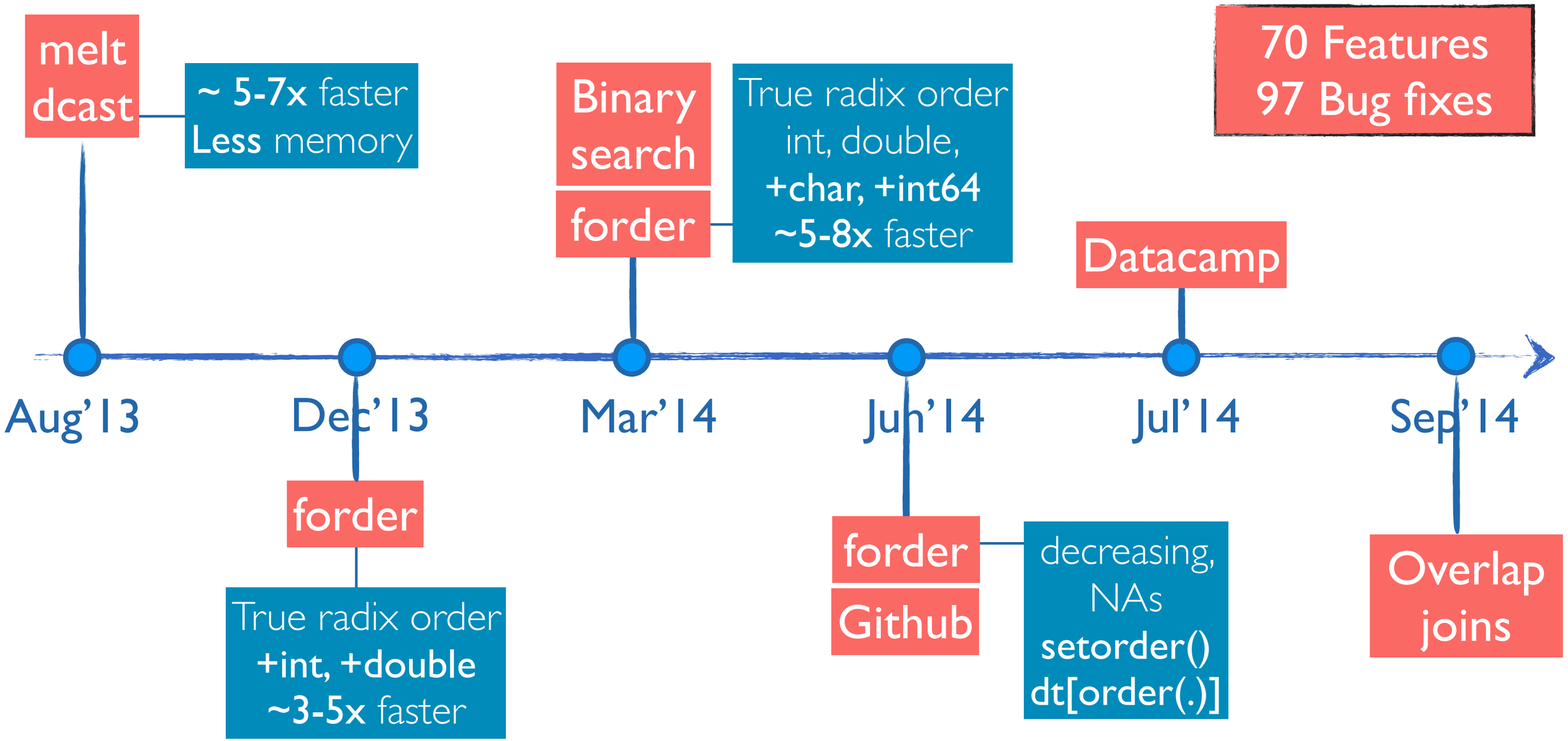
TIMELINE


melt
dcast
~ 5-7x faster
Less memory
Binary search
forder
True radix order
int, double,
+char, +int64
~5-8x faster
Datacamp
Aug'13
Dec'13
Mar'14
Jun'14
Jul'14
forder
True radix order
+int, +double
~3-5x faster
forder
Github
decreasing,
NAs
setorder()
dt[order(.)]

TIMELINE


melt
dcast
~ 5-7x faster
Less memory
Binary search
forder
True radix order
int, double,
+char, +int64
~5-8x faster
70 Features
97 Bug fixes
Datacamp
Aug'13
Dec'13
Mar'14
Jun'14
Jul'14
forder
True radix order
+int, +double
~3-5x faster
forder
Github
decreasing,
NAs
setorder()
dt[order(.)]

TIMELINE

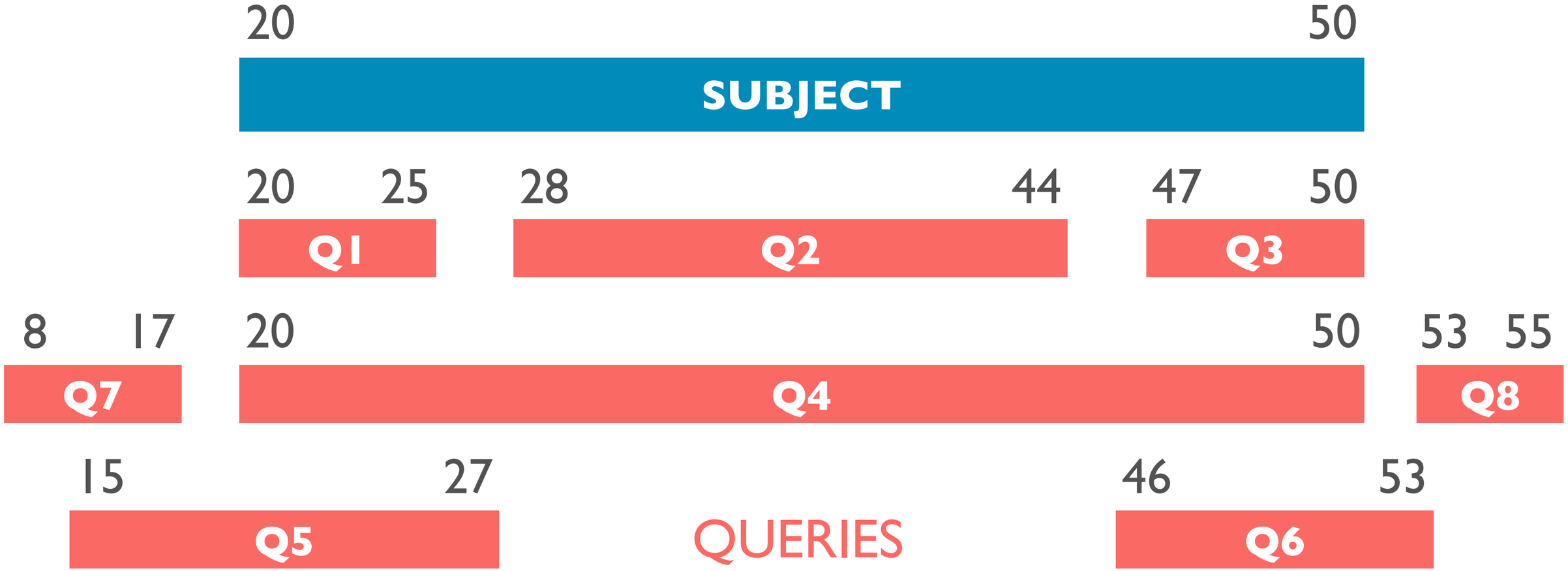

melt
dcast
~ 5-7x faster
Less memory
Binary search
forder
True radix order
int, double,
+char, +int64
~5-8x faster
70 Features
97 Bug fixes
Datacamp
Aug'13
Dec'13
Mar'14
Jun'14
Jul'14
Sep'14
forder
True radix order
+int, +double
~3-5x faster
forder
Github
decreasing,
NAs
setorder()
dt[order(.)]
Overlap
joins

OVERLAP JOINS

WHAT ARE OVERLAPS?

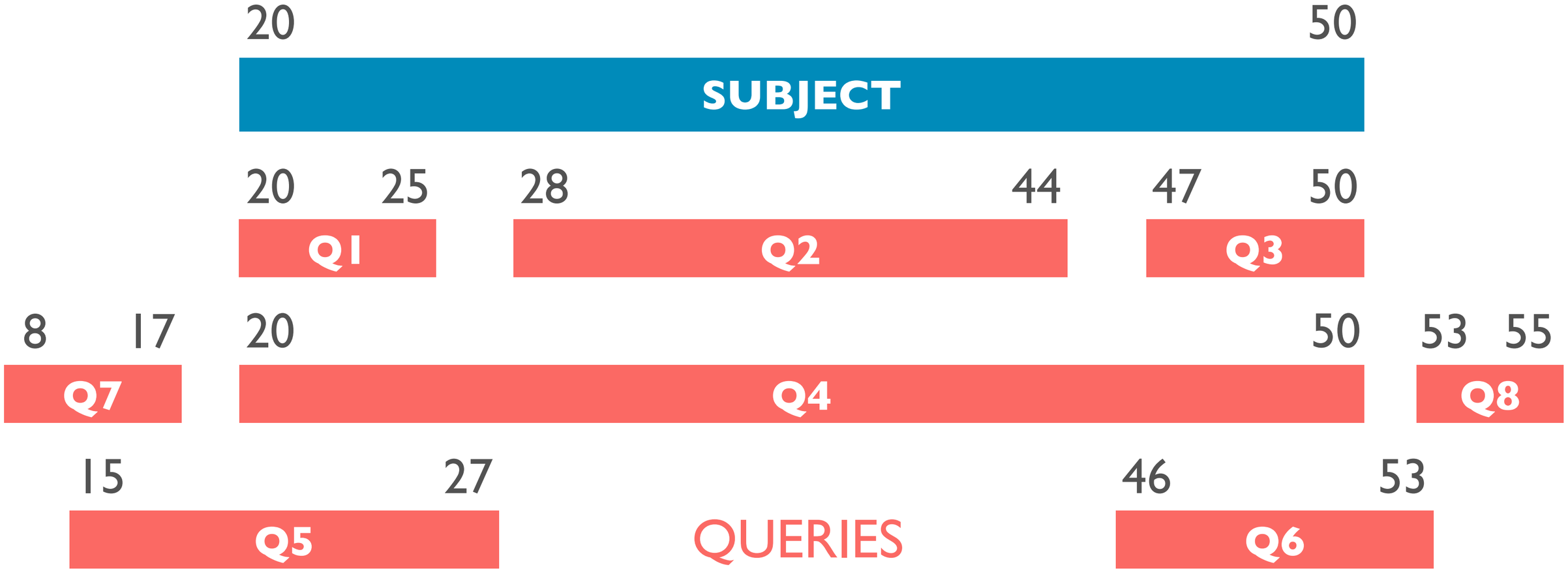
Lawrence et al., PLOS Computational Biology, 2013

WHAT ARE OVERLAPS?



Lawrence et al., PLOS Computational Biology, 2013

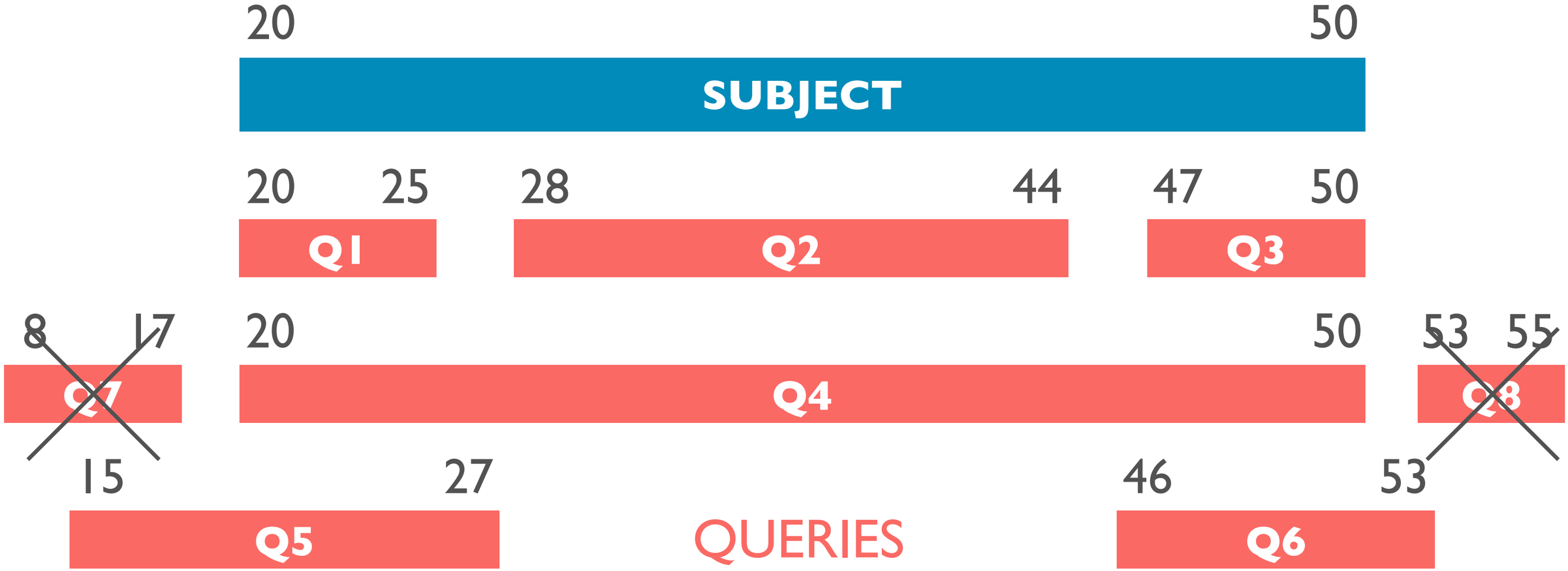
WHAT ARE OVERLAPS?



Which of the red ranges fall completely within the blue range?

Lawrence et al., PLOS Computational Biology, 2013

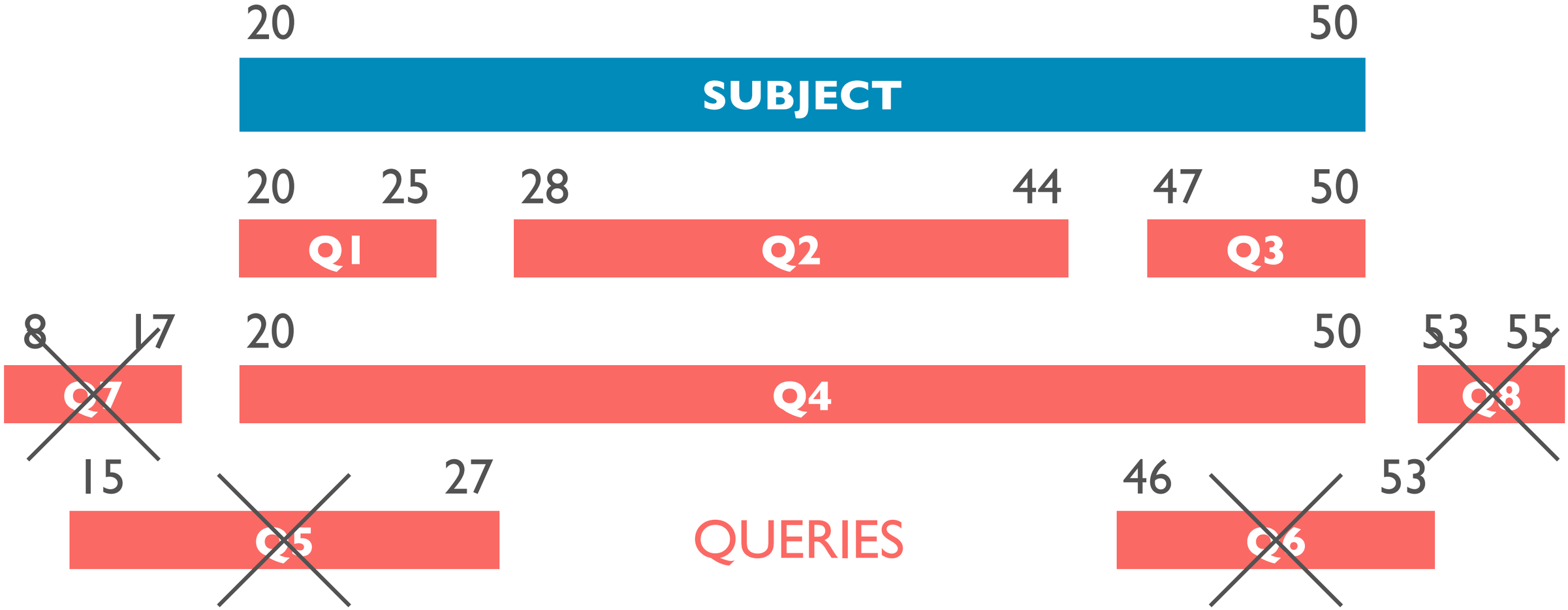
WHAT ARE OVERLAPS?



Which of the red ranges fall completely within the blue range?

Lawrence et al., PLOS Computational Biology, 2013

WHAT ARE OVERLAPS?



Which of the red ranges fall completely within the blue range?

Lawrence et al., PLOS Computational Biology, 2013

HOW IT WORKS: STEP 1

SUBJECT is *usually* much smaller than QUERY

	start	end
1:	10	16
2:	20	35
3:	30	45

SUBJECT

2D-form

HOW IT WORKS: STEP 1

SUBJECT is *usually* much smaller than QUERY

	start	end
1:	10	16
2:	20	35
3:	30	45

SUBJECT

2D-form

1. Sort SUBJECT by **start, end**
2. Add one to **end**
3. collapse and sort *again*

HOW IT WORKS: STEP 1

SUBJECT is *usually* much smaller than QUERY

	start	end
1:	10	16
2:	20	35
3:	30	45

SUBJECT

2D-form

1. Sort SUBJECT by **start, end**
2. Add one to **end**
3. collapse and sort *again*

pos
10
17
20
30
36
46

HOW IT WORKS: STEP 1

SUBJECT is *usually* much smaller than QUERY

	start	end
1:	10	16
2:	20	35
3:	30	45

SUBJECT

2D-form

1. Sort SUBJECT by **start, end**
2. Add one to **end**
3. collapse and sort *again*
4. Get **row numbers**

pos	row number
10	1
17	-
20	2
30	2,3
36	3
46	-

1D -form

HOW IT WORKS: STEP 1

SUBJECT is *usually* much smaller than QUERY

	start	end
1:	10	16
2:	20	35
3:	30	45

SUBJECT

2D-form

1. Sort SUBJECT by **start, end**
2. Add one to **end**
3. collapse and sort *again*
4. Get **row numbers**

pos	row number
10	1
17	-
20	2
30	2,3
36	3
46	-

1D -form

Thanks to **Matt** for describing the technique based on conversation with **@corone**

HOW IT WORKS: STEP 2

Use *rolling joins* on QUERY's **start** and **end** separately

start	end
12	15
41	50
7	9
33	34

QUERY

pos	row number
10	1
17	-
20	2
30	2,3
36	3
46	-

1D -form

roll=**TRUE**
LOCF

HOW IT WORKS: STEP 2

Use *rolling joins* on QUERY's **start** and **end** separately

start	end
12	15
41	50
7	9
33	34

QUERY

pos	row number
10	1
17	-
20	2
30	2,3
36	3
46	-

1D -form

roll=**TRUE**
LOCF

HOW IT WORKS: STEP 2

Use *rolling joins* on QUERY's **start** and **end** separately

start	end
12	15
41	50
7	9
33	34

QUERY

pos	row number
10	1
17	-
20	2
30	2,3
36	3
46	-

ID -form

roll=**TRUE**
LOCF

HOW IT WORKS: STEP 2

Use *rolling joins* on QUERY's **start** and **end** separately

start	end
12	15
41	50
7	9
33	34

QUERY

pos	row number
10	1
17	-
20	2
30	2,3
36	3
46	-

1D -form

roll=TRUE
LOCF

start	end
10	16
NA	NA
NA	NA
20	35
30	45

RESULT

CODE

```
foverlaps(QUERY,
          SUBJECT,
          type = ”within”)
```

WHAT ELSE?

WHAT ELSE?

- Intervals can be **integer**, **numeric**, **integer64**, **Date**, **POSIXct** etc

WHAT ELSE?

- Intervals can be **integer**, **numeric**, **integer64**, **Date**, **POSIXct** etc
- QUERY need not be sorted; its order is preserved in **RESULT**

WHAT ELSE?

- Intervals can be **integer**, **numeric**, **integer64**, **Date**, **POSIXct** etc
- QUERY need not be sorted; its order is preserved in **RESULT**
- Oh... and it's **FAST**!

BENCHMARKS

QUERY	SUBJECT	Data dimensions	data.table **foverlaps**	GenomicRanges (bioconductor) **findOverlaps**
80M rows	33K rows	5 chromosomes, start, end N = 6 samples	2min	16min

"GenomicRanges builds upon IRanges to add biological semantics to metadata"

BENCHMARKS

QUERY	SUBJECT	Data dimensions	data.table **foverlaps**	GenomicRanges (bioconductor) **findOverlaps**
80M rows	33K rows	5 chromosomes, start, end N = 6 samples	2min	16min
65M rows	35K rows	~7500 scaffolds, start, end N = 14 samples (time-course)	4min	>28hrs ~2hrs (parallel)

"GenomicRanges builds upon IRanges to add biological semantics to metadata"