Supplemental Information

HeatITup Algorithm:

Choice of preprocessing

HeatITup has two implementations: `heatitup` and `heatitup-complete`. While `heatitup` accepts `FASTA` files and `heatitup-complete` accepts `BAM` files, both programs output the same CSV file. Before duplication detection, there is a choice of preprocessing built into `heatitup-complete`. `heatitup-complete` can join together the mate pairs to lengthen reads in paired end sequencing. For complex reads, `heatitup-complete` can use Trinity[1] to create contigs from the reads and find duplications within those longer sequences. `heatitup-complete` then calls `heatitup` to detect the repeated sequences. For cases in which preprocessing to elongate or join reads is not needed, or a custom preprocessing is run, we recommend directly using `heatitup`. The following sections cover the algorithm that is implemented by `heatitup`. The complete list of arguments are documented with the `heatitup -h` or `heatitup-complete -h` command.

Identify the duplication sequence of ITD

The longest repeated substring problem can be solved using suffix trees[2]. The algorithm first identifies the locations of the duplications by using an efficient functional lazy suffix tree implementation (`suffixtree` package)[3]. `heatitup` follows this method to identify such a substring and checks if the resulting substring overlaps using the Boyer-Moore algorithm (`stringsearch` package)[4]. In the end, `heatitup` identifies the deepest path in the tree resulting in the current candidate for the duplication (the left candidate CD_l and the right candidate CD_r, both ordered sets of positions, for instance CD_l could be $\{20, 21, 22, 23, 24\}$).

In order to account for point mutations (real or due to sequencing errors) in the duplications, we provide the `--min-mutations` parameter which allows for a certain number m of nucleotides in-between mutations. The algorithm identifies mutations by mutating the nucleotide at $\min(CD_l) - 1$ and the nucleotide at $\min(CD_r) - 1$ ("end mutations") so that both nucleotides are equal. We then reconstruct the suffix tree and obtain a new substring with a new CD_l and CD_r. We do the same procedure for the nucleotide on the right side, $\max(CD_l) + 1$ and $\max(CD_r) + 1$. If the longest of the two substrings is longer than the previously found substring by m or more, then we call the changed nucleotide a mutation and continue to search for more mutations until we end up with candidate substrings that are not longer than the current best by m (Figures 1b and 1c). These positions are then set as D_l and D_r.

Identify the exogenous nucleotides of the spacer of ITD

We call the set of nucleotides in-between the duplication locations, $S = \{\max(D_l) + 1, \max(D_l) + 2, \ldots, \min(D_r) - 1\}$, the "spacer". If $\max(D_l) + 1 = \min(D_r)$, then the spacer is the empty set and thus the sequence is classified as a typical *FLT3*-ITD. Otherwise, we slide the spacer sequence with the left or right duplication, $D_l \cup S$ or $S \cup D_r$ across the reference sequence and find the location with the minimum Hamming distance. The mismatches between these two strings are candidates for exogenous nucleotides within the spacer. The reference sequence here is contained in a `FASTA` file and is at the location of the suspected ITD. For instance, here we use the *FLT3* exon 14 sequence as the reference. Using an incorrect reference (here any gene or sequence that is not *FLT3*) may result in wrong exogenous assignment and filters.

Based on empirical observations, exogenous nucleotides are differentiated from point mutations in the spacer by being present in a long continuous string, whereas a point mutation would be relatively isolated. We chose to use a heat equation as a model for point mutation identification. That is, over time a location with continuous spikes of heat takes longer to disperse under a certain threshold than an isolated spike (Figures 1b and 1c). In our application, the spikes of heat are analogous to mismatched nucleotides, so we implemented the heat equation using the discrete Gaussian kernel. The parameters of the kernel are customizable in the `--gaussian-window`, `--gaussian-time`, and `--gaussian-threshold` parameters. `--gaussian-window` refers to the length of the window for the Gaussian kernel, `--gaussian-time` refers to the number of steps to run the diffusion, and the `--gaussian-threshold` refers to the minimum heat cutoff to be considered an exogenous nucleotide rather than a point mutation.

If there are no exogenous nucleotides within the spacer, the sequence is classified as a typical *FLT3*-ITD, otherwise the sequence is classified as an atypical *FLT3*-ITD.

Filter out false positive duplications

Looking for all repeated sequences within the reads would naturally be prone to false positives – repeated sequences can occur naturally. Especially for smaller repeated sequences this can become an issue where the signal is lost in the noise. In order to remove repeated sequences that can be found in the reference, we have implemented several filters. We provide the `--blacklist-input` parameter to point to a fasta file containing known false positives. If set, a candidate duplication is compared to each sequence in the blacklist. If the two sequences share a Levenshtein distance[5] of less than the parameter `--levenshtein-distance`, the sequence is considered a false positive and the next longest duplication candidate is evaluated.

We can utilize known information to get a more specific filter using the reference sequence itself. If the reference sequence is provided via the `--reference-input` parameter, we can computationally derive two types of blacklists in addition to the provided one: a basic "match" and a recursive blacklist creation. For the basic match, `--reference-check-blacklist`, the candidate duplication sequence is located at each position in the reference sequences. If there are two or more non-overlapping occurrences within a sequence, the candidate is called a false positive. These checks are efficient and are recommended over the following process.

The alternative approach to generate a blacklist is through the `--reference-recursive-blacklist` parameter. This option creates a blacklist by running the duplication finder algorithm on the sequences and finding the longest repeated substring in each one. Those sequences not in the blacklist (checked with the Levenshtein distance) are added to the blacklist and the algorithm is

run again, getting the next longest duplication and so on until all duplications are in the blacklist. This process may be faster and more precise (due to the Levenshtein distance being applied to this blacklist whereas `--reference-check-blacklist` only searches for exact matches) for smaller reference sequences, but becomes much slower for longer sequences due to the worse scalability of the duplication finder versus the substring matching.

The reference can also provide a template for contigs generated from unmapped reads in `heatitup-complete`. Using the `--blast-command` argument in the assembly method removes contigs with no significant alignments to the provided reference. Furthermore, if the reads were aligned and provided as BAM input, we can use the CIGAR to efficiently remove reads matching all positions that are unlikely to contain any duplications. These steps remove much of the noise that results in repeated strings that are not actual duplications.

Output format

The output of both `heatitup` and `heatitup-complete` is a `CSV`.

Explanation of each column in the `heatitup` output `CSV`.

Column name	Definition
`label`	Inputted label from the `--label` field
`fHeader`	Header of the sequence
`fSequence`	Read sequence
`dSubstring`	Duplicated segment
`dLocations`	Positions of the duplicated segment and its origin
`dMutations`	Positions of the point mutations pairs
`sSubstring`	Spacer segment
`sLocation`	Position of the spacer
`sOtherLocations`	Positions of the exogenous nucleotides
`classification`	Classification of the read: Typical, Atypical, or Normal (no duplication)

Computational complexity

The algorithm uses repeated lazy functional suffix tree constructions and searches, as well as a sliding window for the spacer identification. Using a constant alphabet of nucleotides, we can expect to construct the lazy suffix tree in time complexity $O(kn \log n)$, with a worst case of $O(kn^2)$, where k is the size of the alphabet (a constant here) and n is the length of the string[3]. With larger alphabets, this implementation is faster than both Ukkonen and McCreight's algorithms[3]. With mutations, we construct the suffix tree O(n) times. Similarly, we find the candidate atypical mutations using a sliding window, with a worst case of the entire sequence being a spacer, $O(n^2)$. Finally, the discrete Gaussian kernel is constant, with just the input of the heat signal creating a complexity of O(n). Thus the algorithm in total has complexity of $O(n^2)$.

The algorithm is implemented in a streaming fashion, so constant space is taken for all input sequences as we can classify each sequence independently. For the data from the TCGA cohort consisting of 8153681 reads from 321 samples, the assembly based algorithm with Trinity (using the entire *FLT3* gene as a reference for the blacklist) ran in 41:32 hours on a single thread. Without

assembly, the algorithm ran in 21:45 minutes on a single thread, although this process could be easily parallelized.

The algorithm was implemented in Haskell and available at https://github.com/faryabib/HeatITup.

ITD Characterization

We determined the proportion of typical ITD by (# typical clones / (# atypical clones + # typical clones)). In comparing the lengths of the duplicated segment and spacer segment, we used the Mann-Whitney U test. We quantified the fraction of the spacer that was exogenous by counting the number of like nucleotides and dividing them by the length of the spacer. Each atypical clone contributes a value to each nucleotide, so if the exogenous segment was "AGG" and the spacer was "AGGTTC" the resulting values would be (A, 1), (C, 0), (G, 2), (T, 0) before percentage conversion. We used the Mann-Whitney U test for comparing purines versus pyrimidines and Tukey's HSD after ANOVA to find significance of each individual nucleotide.

Statistical Analysis

ITD Clonal assignment

All reads with a similar *FLT3*-ITD mutation were assigned to a clone with an allele frequency (AF). We provide the helper program `collapse-duplication` to assist in grouping together similar mutations into clones. Using the assumption that the anchor points of the duplication should remain relatively stable, the program groups together reads with similar duplication and spacer segment locations and lengths. The window length for the positions to be near each other between reads, as well as the difference in lengths for the spacer and duplication, can be lenient (the beginning position of the spacer can be within 5 nucleotides between reads, for instance, using the `--wiggle 5` option). We can then use the grouped together reads to calculate the AF. The amount of leniency between reads was set to 5 nucleotides in this analysis. We removed clones with an AF smaller than 1%, resulting in 189 typical and 50 atypical clones in all 97 individuals. For the TCGA cohort, we included clones if their *FLT3*-ITD duplication length was 15 or greater. Results from `heatitup` on TCGA were compared to the table in Rustagi *et al.*[6]. Each individual's representative *FLT3*-ITD clone was determined based on the dominant clone from the earliest sample (University of Pennsylvania) or, due to less coverage, longest duplication (TCGA).

Survival analysis

Overall survival (OS) curves for both the FLT3i-treated and non-FLT3i-treated studies were calculated from the date of diagnosis to the date of censor, either death or latest follow-up. A Cox proportional hazards model was applied to each cohort for each relevant variable using the `survival` R library. The proportional hazards assumption for each presented variable was tested and met using Schoenfeld residuals[7].

To identify possible confounding variables in this analyses, a univariate Cox regression was used for each candidate confounding variable: sex, age, allogenic transplant, disease type (*de novo* or relapsed), remission at initiation of TKI treatment, and FLT3i type. Allogenic transplant

and remission at initiation of TKI treatment were significant contributors in this analysis and were included in all subsequent survival analyses where applicable.

For the survival curves, the `survminer` was used to plot the raw Kaplan-Meier curves for the cohort. For the exogenous ratio curve, the ratio resulting in the maximum separation between the two groups was used for visualization only, the complete spectrum of values was used in the Cox model.

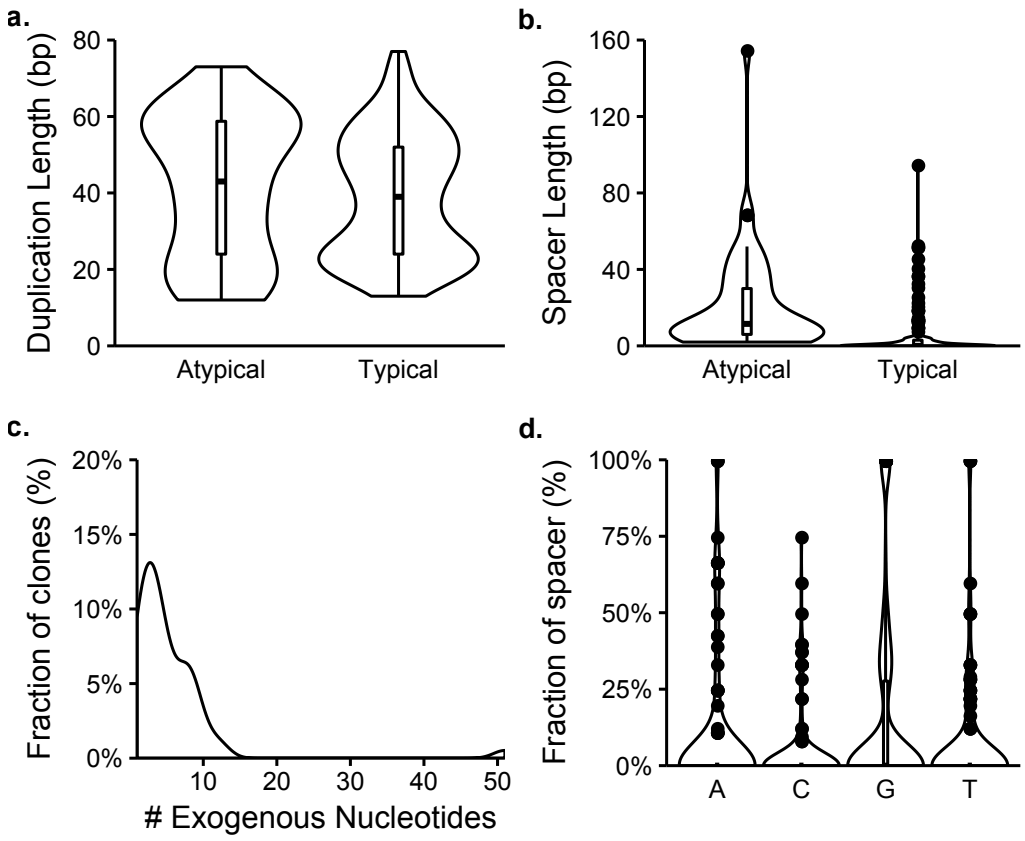

Figure S1: Characterization of *FLT3*-ITD clones in the FLT3i cohort. a) Violin plots of duplicated segment lengths for typical and atypical clones (Typical median: 39 bp, Atypical median: 43, Mann-Whitney U test: p = 0.294). b) Violin plots of spacer segments for typical and atypical clones (Typical median: 0, Atypical median: 11.5, Mann-Whitney U test: $p < 2.22e-16$). c) Distribution of the number of exogenous nucleotides per clone (median (mean): 4 (5.44) bp). d) Exogenous nucleotide usage. By Tukey's HSD after ANOVA, G > A (p = 0.0588), G > C ($p = 9.54e-5$), G > T (p = 0.00490), and ($p > 0.05$) for all other comparisons. 186 typical clones and 50 atypical clones.

Table S1: Confusion matrix of `HeatITup` and genetic reviewers manual annotation for sequences in Figure 3.

		Expert visual annotation	
		Typical	Atypical
`HeatITup`	Typical	22	0
	Atypical	0	17

Table S2: Univariate Cox regression analyses for the *FLT3*-ITD-positive UPENN cohort.

Variable	HR	CI	p-value
Age	1.00	1.00 – 1.00	0.737
Sex	1.03	0.604 – 1.76	0.916
Disease status	0.605	0.333 – 1.10	0.0991
Allogenic transplant	0.440	0.256 – 0.757	0.00301
Remission at TKI treatment (FLT3i+ only)	4.35	1.54 – 12.2	0.00539
Clone frequency	2.26	0.430 – 11.89	0.335
FLT3i type (FLT3i+ only)	0.783	0.528 – 1.16	0.221

Table S3: Cox regression analysis of *FLT3*-ITD duplication length for the non-FLT3i-treated cohort.

Variable	HR	CI	p-value
Duplication length	1.01	0.991 – 1.04	0.250
Cohort	1.21	0.578 – 2.54	0.612

Table S4: Cox regression analysis of *FLT3*-ITD classification for the non-FLT3i-treated cohort.

Variable	HR	CI	p-value
Typical or atypical	0.777	0.419 – 1.44	0.422
Cohort	0.917	0.483 – 1.74	0.793

Table S5: Cox regression analysis of *FLT3*-ITD duplication length with classification for the non-FLT3i-treated cohort.

Variable	HR	CI	p-value
Duplication length	1.01	0.991 – 1.04	0.244
Typical or atypical	0.772	0.418 – 1.43	0.409
Cohort	1.15	0.542 – 2.46	0.711

Table S6: Cox regression analysis of *FLT3*-ITD classification for the top third longest ITD duplications in the non-FLT3i-treated cohort (Benjamini-Hochberg adjusted p-values from long non-FLT3i-treated cohort analyses).

Variable	HR	CI	p-value	Adjusted p-value
Typical or atypical	0.226	0.0764 – 0.669	0.00722	0.0253
Cohort	2.069	0.757 – 5.66	0.156	-

Table S7: Cox regression analysis of *FLT3*-ITD spacer sequence exogenous fraction for the non-FLT3i-treated cohort.

Variable	HR	CI	p-value
Exogenous fraction	1.43	0.412 – 4.96	0.573
Cohort	0.723	0.324 – 1.615	0.429

Table S8: Cox regression analysis of *FLT3*-ITD spacer sequence exogenous fraction for the top third longest ITD duplications in the non-FLT3i-treated cohort.

Variable	HR	CI	p-value	Adjusted p-value
Exogenous fraction	159	4.92 – 5160	0.00427	0.0253
Cohort	14.8	1.77 – 123	0.0129	-

Table S9: Cox regression analysis of FLT3i treatment for the UPENN cohort.

Variable	HR	CI	p-value
FLT3i treated	0.0432	0.00583 – 0.320	0.00209
Allogenic transplant	0.487	0.279 – 0.849	0.0111
Remission at TKI treatment (FLT3i+ only)	4.64	1.69 – 12.8	0.00296

Table S10: Cox regression analysis of *FLT3*-ITD duplication length for the FLT3i-treated cohort (Benjamini-Hochberg adjusted p-values from FLT3i cohort analyses).

Variable	HR	CI	p-value	Adjusted p-value
Duplication length	0.993	0.973 – 1.01	0.531	0.531
Allogenic transplant	0.413	0.209 – 0.814	0.0106	-
Remission at TKI treatment	4.48	1.56 – 12.8	0.00531	-

Table S11: Cox regression analysis of *FLT3*-ITD classification for the FLT3i-treated cohort (Benjamini-Hochberg adjusted p-values from FLT3i cohort analyses).

Variable	HR	CI	p-value	Adjusted p-value
Typical or atypical	0.337	0.165 – 0.689	0.00286	0.00572
Allogenic transplant	0.306	0.154 – 0.607	0.000698	-
Remission at TKI treatment	4.82	1.69 – 13.8	0.00326	-

Table S12: Cox regression analysis of *FLT3*-ITD spacer sequence exogenous fraction for the FLT3i-treated cohort.

Variable	HR	CI	p-value
Exogenous fraction	4.96	1.33 – 18.5	0.0173
Allogenic transplant	0.784	0.313 – 1.96	0.603
Remission at TKI treatment	3.25	0.913 – 11.6	0.0689

References

1. Grabherr, M. G. *et al.* Full-length transcriptome assembly from RNA-Seq data without a reference genome. *Nature Biotechnology* **29,** 644–652 (2011).
2. Karp, R. M., Miller, R. E. & Rosenberg, A. L. *Rapid identification of repeated patterns in strings, trees and arrays* in (ACM Press, 1972), 125–136.
3. Giegerich, R. & Kurtz, S. A comparison of imperative and purely functional suffix tree constructions. *Science of Computer Programming* **25,** 187–218 (1995).
4. Boyer, R. S. & Moore, J. S. A fast string searching algorithm. *Communications of the ACM* **20,** 762–772 (1977).
5. Levenshtein, V. I. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady* **10,** 707 (1966).
6. Rustagi, N. *et al.* ITD assembler: an algorithm for internal tandem duplication discovery from short-read sequencing data. *BMC Bioinformatics* **17** (2016).
7. Schoenfeld, D. Partial Residuals for The Proportional Hazards Regression Model. *Biometrika* **69,** 239 (1982).