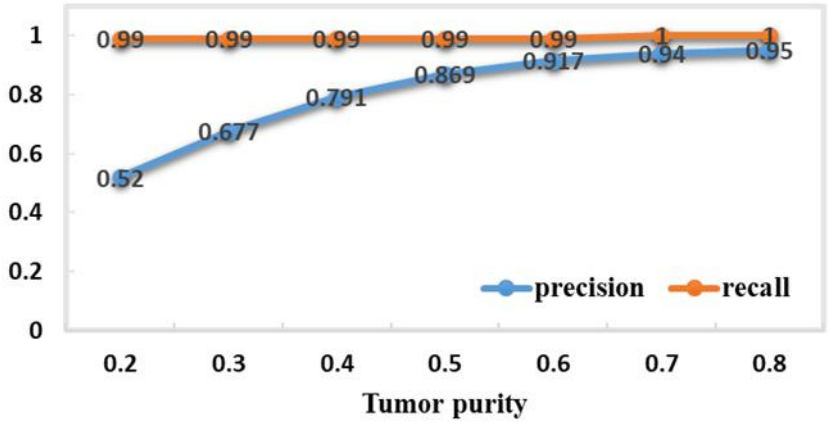
Supplementary text

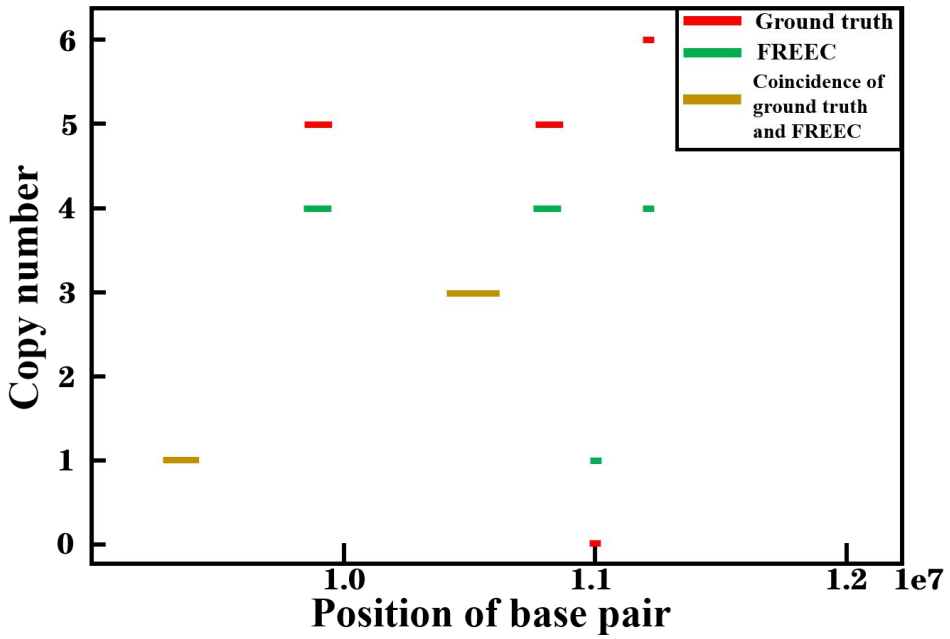
1. STIC and FREEC results

STIC is a recently developed method to detect single nucleotide variants(SNVs) SNVs and distinguish germline and somatic mutations. STIC detects SNVs by training a back propagation (BP) neural network algorithm on a set of extracted features and creating an iterative process to distinguish somatic and germline SNVs. STIC input is a single tumor sample. Here, we used STIC to detect SNVs for downstream analysis. To test the detection effect of STIC on SNVs, a simulation experiment was performed. Both precision and recall were used to estimate STIC performance, with results shown in Supplementary Figure 1.



Supplementary Figure 1. STIC performance based on sensitivity and recall using simulation datasets with tumor purity ranging from 0.2 to 0.8 and coverage of 30X.

The FREEC tool can automatically calculate copy number and allele content spectra in next-generation sequencing (NGS) data, thereby predicting the regions where the genome has changed. We used FREEC to detect copy number variations (CNVs). The copy number detection results are presented in Supplementary Figure 2.



Supplementary Figure 2. Performance of FREEC based on simulation datasets with tumor purity of 0.7 and coverage of 30X.

In Supplementary Figure 2, the simulation datasets had a tumor purity of 0.7, coverage of 30X, and intercepted positions of 1.0×10^7 to 1.2×10^7, representing the copy number detection results. The red line represents the simulation data results, the green line represents the FREEC detection results, and the yellow line represents their coincidence. FREEC was accurate at detecting CNV length; although copy number was slightly different from the simulation results, which may impact our downstream analysis, overall, the FREEC detection results for CNVs were good.

2. SVM classifier

Here, we used the support vector machine(SVM) as a binary classifier. The training dataset was denoted as $\{x_i,y\_i\}$, where $x_i \in R^n, i = 1, 2,...N$, x_i is the i-th input vector, and y_i is the corresponding class label (1 or 0), with 1 and 0 representing germline and somatic mutations, respectively. The training dataset contained only two data

types, labeled 1 and 0. To improve the classifier's ability, the boundary distance between the two types of data needs to be maximized (Brereton & Lloyd, 2010). The SVM classifier was obtained by optimizing the objective function, which can balance the terms of forcing separation classes and maximizing margin separation (Hastie & Tibshirani, 1998). SVM training was performed to determine the only relevant samples of the optimal separation boundary, called support vectors (SVs). The problem was transformed into dual space and solved by introducing the Lagrange Multipliers Vector in the following form (Gold & Sollich, 2003; Suykens, 2002):

$$max_{\alpha} W(\alpha) = \sum_{j=1}^{N} \alpha_j - \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j K(x_i, x_j)$$

$$\text{s.t.} \begin{cases} \alpha_i \geq 0 \ , i = 1,...,N \\ \sum_{i=1}^{N} \alpha_i y_j = 0 \end{cases}, \quad (1)$$

where $\alpha = (\alpha_1,...,\alpha_N)$ is the Lagrange multipliers vector and $K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$ is the kernel function (Brereton & Lloyd, 2010).

As real data classification is usually inseparable and non-linear, the input vector is mapped to high-dimensional feature space, so that data in the feature space will become linearly separable. The linear SVM method is then used to train the separation hyperplane parameters. The high-dimensional calculations increase sharply, introducing dimensionality and kernel issues. Here, we used a Radial Basis Function (RBF) kernel, which has excellent non-linear modeling performance (Hsu et al., 2008; Renukadevi & Thangaraj, 2013).

$$K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) \quad (2)$$

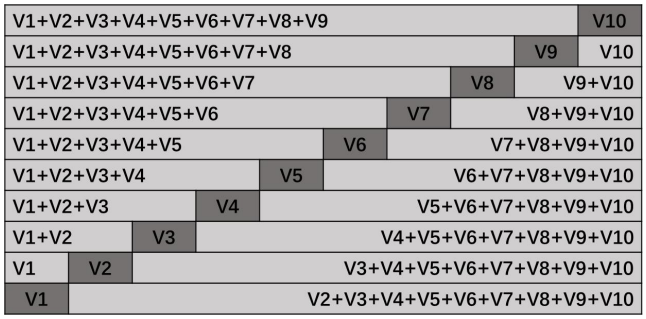
where σ^2 is the common width. The decision function of SVM, $y(x)$, and output prediction label of the input vector, x, can be written as:

$$y(x) = sign\left[\sum_{i=1}^{N} y_i \alpha_i K(x, x_i) + b\right] \quad (3)$$

3. 10-fold cross-validation

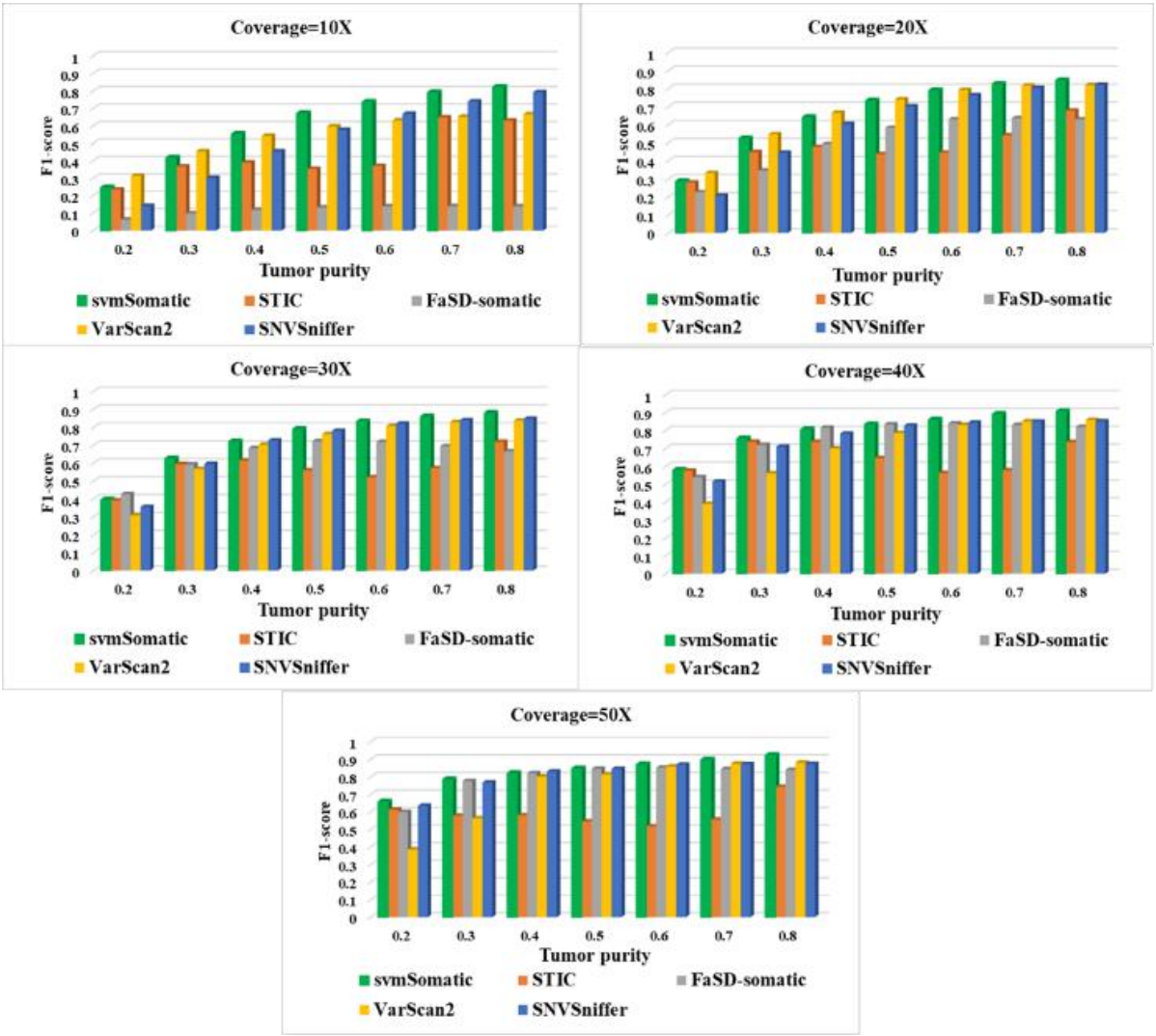
To obtain the best classification effect, the C and γ parameters need to be set when RBF is the kernel function (Schölkopf & Smola, 2002). Parameter C is the penalty coefficient and the tolerance of the model to errors decreases as C increases. Parameter γ, which is attached to RBF, determines the distribution of the original data mapped to the high-dimensional feature space, which is inversely proportional to the number of support vectors. The number of support vectors affects the classification speed of the classifier. Therefore, the selection of an appropriate C and γ is required to improve classification efficiency by ensuring performance of the classifier.

There are multiple ways to perform a hyperparameter search, including grid, random, and heuristic searches. Grid searches are straightforward and practical (Budiman et al., 2017) and entail looping through all candidate parameters and trying every possible combination, resulting in the best-performing set of parameters. Simultaneously, acting on the initial data division results and performing cross-validation reduces contingency (Tsamardinos et al., 2015). Cross-validation randomly divides the dataset into v partitions, called v-folds. In general, the dataset is divided into 10-fold cross-validations (Syarif et al., 2016), with one part used interchangeably for the test dataset, and the other parts used for training. As shown in Supplementary Figure 3, the darker shade represents the test fold and lighter shade represents the training folds.



Supplementary Figure 3. 10-fold cross-validation.

4. Simulation study results



Supplementary Figure 4. Performance comparisons of five methods based on F1-scores using simulation datasets, with tumor purity ranging from 0.2 to 0.8 and coverage ranging from 10X to 50X.

5. Values of recall and precision for five methods using simulation dataset

Coverage=10X

Method		Purity=0.2	Purity=0.3	Purity=0.4	Purity=0.5	Purity=0.6	Purity=0.7	Purity=0.8
svmSomatic	precision	0.957	0.931	0.85	0.777	0.868	0.917	0.899
	recall	0.141	0.269	0.41	0.592	0.638	0.695	0.759
STIC	precision	0.825	0.825	0.839	0.558	0.641	0.754	0.669
	recall	0.137	0.236	0.254	0.258	0.26	0.568	0.595
FaSD-somatic	precision	0.742	0.814	0.839	0.852	0.857	0.856	0.856
	recall	0.033	0.052	0.064	0.072	0.075	0.076	0.075
VarScan2	precision	0.997	0.998	0.998	0.999	0.999	0.999	0.999
	recall	0.185	0.294	0.373	0.426	0.461	0.484	0.5
SNVSniffer	precision	0.657	0.819	0.884	0.914	0.93	0.94	0.946
	recall	0.079	0.185	0.306	0.421	0.523	0.61	0.682

Coverage=20X

Method		Purity=0.2	Purity=0.3	Purity=0.4	Purity=0.5	Purity=0.6	Purity=0.7	Purity=0.8
svmSomatic	precision	0.854	0.831	0.782	0.757	0.78	0.808	0.824
	recall	0.171	0.372	0.544	0.711	0.8	0.842	0.866
STIC	precision	0.865	0.901	0.895	0.87	0.856	0.87	0.889
	recall	0.166	0.298	0.324	0.292	0.3	0.393	0.546
FaSD-somatic	precision	0.783	0.846	0.871	0.885	0.891	0.89	0.878
	recall	0.132	0.217	0.343	0.434	0.487	0.495	0.491
VarScan2	precision	0.998	0.998	0.999	0.999	0.999	0.999	0.999
	recall	0.199	0.377	0.499	0.587	0.656	0.689	0.694
SNVSniffer	precision	0.987	0.994	0.996	0.997	0.997	0.997	0.997
	recall	0.115	0.286	0.433	0.542	0.618	0.672	0.696

Coverage=30X

Method		Purity=0.2	Purity=0.3	Purity=0.4	Purity=0.5	Purity=0.6	Purity=0.7	Purity=0.8
svmSomatic	precision	0.917	0.887	0.818	0.776	0.794	0.85	0.853
	recall	0.251	0.482	0.643	0.802	0.875	0.868	0.887
STIC	precision	0.931	0.944	0.933	0.907	0.881	0.88	0.902
	recall	0.247	0.434	0.458	0.404	0.369	0.422	0.596
FaSD-somatic	precision	0.826	0.881	0.901	0.909	0.909	0.903	0.897
	recall	0.287	0.446	0.549	0.598	0.594	0.563	0.528
VarScan2	precision	1	1	1	1	1	1	1
	recall	0.182	0.396	0.531	0.616	0.675	0.706	0.72
SNVSniffer	precision	1	1	1	1	1	1	1
	recall	0.208	0.423	0.57	0.639	0.695	0.723	0.735

Coverage=40X

Method		Purity=0.2	Purity=0.3	Purity=0.4	Purity=0.5	Purity=0.6	Purity=0.7	Purity=0.8
svmSomatic	precision	0.971	0.937	0.865	0.792	0.849	0.895	0.897
	recall	0.414	0.633	0.754	0.88	0.871	0.888	0.917
STIC	precision	0.972	0.969	0.956	0.929	0.894	0.883	0.907
	recall	0.409	0.595	0.6	0.495	0.411	0.43	0.62
FaSD-somatic	precision	0.937	0.952	0.957	0.958	0.957	0.954	0.947
	recall	0.378	0.579	0.711	0.74	0.748	0.738	0.725
VarScan2	precision	1	1	1	1	1	1	1
	recall	0.242	0.39	0.538	0.647	0.712	0.742	0.753
SNVSniffer	precision	1	1	1	1	1	1	1
	recall	0.347	0.55	0.643	0.705	0.73	0.739	0.742

Coverage=50X

Method		Purity=0.2	Purity=0.3	Purity=0.4	Purity=0.5	Purity=0.6	Purity=0.7	Purity=0.8
svmSomatic	precision	0.897	0.85	0.786	0.793	0.824	0.862	0.899
	recall	0.519	0.729	0.857	0.893	0.921	0.935	0.948
STIC	precision	0.981	0.971	0.956	0.93	0.899	0.893	0.921
	recall	0.211	0.412	0.416	0.386	0.364	0.403	0.621
FaSD-somatic	precision	0.957	0.97	0.974	0.975	0.976	0.975	0.971
	recall	0.436	0.647	0.723	0.747	0.756	0.744	0.739
VarScan2	precision	1	1	1	1	1	1	1
	recall	0.239	0.392	0.668	0.684	0.751	0.776	0.786
SNVSniffer	precision	1	1	1	1	1	1	1
	recall	0.465	0.622	0.709	0.732	0.768	0.773	0.775

REFERENCES

Brereton RG, Lloyd GR. 2010. Support vector machines for classification and regression. *Analyst*, **135**(2): 230–267.

Budiman F, Suhendra A, Agushinta D, Tarigan A. 2017. Determination of SVM-RBF kernel space parameter to optimize accuracy value of indonesian batik images classification. *Journal of Computer Science*, **13**(11): 590–599.

Gold C, Sollich P. 2003. Model selection for support vector machine classification. *Neurocomputing*, **55**(1–2): 221–249.

Hastie T, Tibshirani R. 1998. Classification by pairwise coupling. *The Annals of Statistics*, **26**(2): 451–471.

Hsu C, Chang C, Lin C. 2008. A practical guide to support vector classification. *BJU International*, **101**(1): 1396–1400.

Renukadevi NT, Thangaraj P. 2013. Performance evaluation of SVM – RBF kernel for medical image classification. *Global Journal of Computer Science and Technology*, **13**(4).

Schölkopf B, Smola AJ. 2002. Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. Cambridge: The MIT Press, 1.

Suykens JAK. 2002. Least Squares Support Vector Machines. River Edge, NJ: World Scientific, 605–615.

Syarif I, Prugel-Bennett A, Wills G. 2016. SVM parameter optimization using grid search and genetic algorithm to improve classification performance. *Telkomnika*, **14**(4): 1502–1509.

Tsamardinos I, Rakhshani A, Lagani V. 2015. Performance-estimation properties of cross-validation-based protocols with simultaneous hyper-parameter optimization. *International Journal on Artificial Intelligence Tools*, **24**(5): 1540023.