Supplementary Information:

BayesPI – In 2009, BayesPI [1] was developed based on the biophysical theory of protein-DNA interactions, which can be used to investigate high throughput sequencing data (e.g., ChIP-Seq) for predicting transcript factor (TF) binding sites. It is based on the Bayesian statistical method that allows to predict TF binding motifs by incorporating prior knowledge of TF motif sequence. Though the primary goal of the work was to predict the TF binding sites at DNA sequences (or position weight matrix – PWM), it could compute the binding score for each known-TF with any targeting DNA sequence. Bayesian model regularization [2] was implemented in BayesPI for non-linear parameter fitting. Here, a common biophysical modelling of protein-DNA interactions is:

$$\text{protien} + \text{DNA} \Leftrightarrow \text{protien} - \text{DNA}$$

where binding of protein to DNA with reaction constants k_a (protein-DNA association) and k_d (protein-DNA dissociation) can be expressed as the quotient:

$$\frac{k_a(S)}{k_d(S)} = K \exp\left(\frac{-\Delta G(S)}{RT}\right)$$

Where ΔG is standard Gibbs free energy exchange of a protein binding to a short stretch of DNA with sequence S, R Is the gas constant, K is an inverse equilibrium concentration and T is the absolute temperature. Based on above quotient, a probability of the DNA sequence S to be bound by a protein in a solution is given by P(S) = 1/(1+ kd(S)/ka(S)), or as following:

$$P(S) = \frac{1}{1 + \exp\left(\frac{\Delta G(S) - \mu}{RT}\right)}$$

where μ is the chemical potential set by the protein concentration. The theoretically estimated probability P(S) of protein-DNA interactions can be used to fit a regression model based on experimental observations (e.g., ChIP-Seq experiment). This is a nonlinear minimalization problem, which can be solved using novel Bayesian method. For example, a model error function is given as:

$$E_D = \frac{1}{2}\sum_{i=1}^{g}(t_i - Y_i)^2$$

where t_i is the measured ChIP-seq affinity profile to a DNA sequence i, and Y_i is the predicted/estimated TF occupancy score for the same sequence i based on the TF binding probability P(S). Y_i is given by:

$$Y_i = w \sum_{l=1}^{n-M+1} P_{i,l} + b$$

where b is a value of bias and w is motif weight, $P_{i,l}$ is the protein binding probability and is given as:

$$P_{i,l}(S_{i,l,j}) = \frac{1}{1 + \exp\left(\sum_{j=1}^{4M} w'_j S_{i,l,j} - b'\right)}$$

In this equation, l = 1…n-M+1, while n and M are sequence length and motif length, respectively; i = 1…g while g is the total number of DNA sequences (fragments) in data; b' represent the chemical potential μ.

BayesPI-BAR – In 2015, a new type of application of biophysical modelling of protein-DNA interactions was developed, which was used to predict TFs binding affinities affected significantly by mutations in DNA sequences - Bayesian method for protein-DNA interaction with binding affinity ranking (or BayesPI-BAR)[3]. BayesPI-BAR was aimed for the prediction of functional regulatory mutations, which play an important role in gene transcription regulation. For example, a DNA sequence variation in the regulatory regions (e.g., promoter or enhancer) may disrupt transcript regulations and result in diseases. For BayesPI-BAR application, the DNA sequences around point mutations/SNPs (e.g., 20bp to two-sides of a SNP) and a list of PWMs for known TFs are required as input, in order to predict the TF binding affinity changes that affected significantly due to mutations. Finally, the top ranked positively and negatively affected TFs in DNA binding affinity are exported, where the positive and negative ones represent TF binding affinity are increased and decreased due to mutations, respectively. In this package, the effect of sequence variation on TF binding was further ranked by shifted differential binding affinity, where the differential binding affinity (dbA) is given by:

$$dbA(S_i) = w \sum_{l=1}^{N-m+1} P_{i,l}(S_i) - \frac{\sum_{r=1}^{R} w \sum_{l=1}^{N-m+1} P_{i,l,r}(S_{i,r})}{R}$$

S_i represents a DNA sequence, $S_{i,r}$ is the randomly mutated variant inside S_i, m is the length of the PWM of TF motif, N is the length of the sequence, R is the total number of random shuffling of DNA sequence S_i, w is a weight coefficient, $P_{i,l}(S_i)$ is representation of the estimated protein–DNA binding probability at sequence S_i and $P_{i,l,r}(S_{i,r})$ represents estimated protein–DNA binding probability at the randomly mutated sequence $S_{i,r}$. The $dbA(S_i)$ measures the strength of a TF-DNA binding when it is compared to that of the background model. R controls the precision of estimations, which increases with increasing R. Usually, $R = 10000$ gives sufficient accuracy for the prediction. The *dba* values have been used to distinguish between direct and indirect protein-DNA interactions. The shifted differential binding affinity is given by:

$$\delta dbA(S_{ref}, S_{alt}, w, \mu, R) = dbA(S_{ref}, w, \mu, R) - dbA(, w, \mu, R)$$

where S_{ref} is the reference and S_{alt} is alternate (mutated) DNA sequence. The binding energy matrices w can be obtained from common position specific weight matrices (PWM), which are available for many human TFs. The δdbA values are computed by using a set of predefined chemical potential μ (e.g., μ = 0, -10, -13, -15, -18, and -20). Thus, the δdbA values can be used to evaluate the significance of TF binding affinity changes due to DNA sequence mutations.

BayesPI-BAR2 – In 2019, an updated version BayesPI-BAR2 [4] was developed to predict functional regulatory mutations in cancer patient cohort based on both whole genome-wide sequencing (WGS) and RNA-Seq data sets. BayesPI-BAR2 considers information from both multiple mutations and multiple patients and was implemented in python scripts instead of the multiple languages (e.g., R, MATLAB and Perl et al) in the earlier version. The significance of the TF-DNA binding affinity changes is mainly achieved by three steps: 1) the predicted patient-specific mutation blocks are considered as foreground mutation blocks, which are evaluated by 1772 PWMS of human TFs; 2) the randomly extracted mutation blocks from promoters regions (e.g., the sequence length is kept same as the patient -specific mutation block; the position of the nucleotide is randomly selected to produced mutation) are used as background mutation blocks, which are tested by the same 1772 PWMs for ranking top affected TFs; 3) by comparing the results between the foreground and the background models, to predict TFs with significant binding affinity changes due to patients mutations. The BayesPI-BAR2 is computationally efficient and works on parallel processes, which reduces overall waiting time

significantly and can be used to predict functional regulatory mutations genome-widely in cancer.

BayesPI-BAR3 or bpb3 – In 2023, a new python package bpb3 was developed for predicting functional non-coding mutations, where the clustered PWMs based on DBD information from abc4pwm are integrated in it. Thus, the new bpb3 tool can not only predict the effect of sequence variations on the TF binding affinity changes, but also on that of clustered/representative TFs. In this way, it reduces the number of evaluations from thousands of TFs to a few hundreds of representative motifs of clustered TFs, which may significantly reduce the cost of total CPU hours. Furthermore, the new bpb3 can predict functional regulatory mutations based on any predefined genomic regions (e.g., differential methylated regions -DMRs). Thus, the new version extends its application from single predict TF binding motif to identify functional mutation and mutation blocks. The input data set extends from single ChIP-seq or SNPs sequences to integration of multiples omics data together such as genome, transcriptome and methylome. It is a user-friendly Python package with easy installation and set up. Especially, with a predefined configuration file, the whole analysis pipeline of bpb3 can be applied on a new data set automatically, by simply modifying the parameters in the configure file. In Supplementary Methods Figure S7, an example of the application of two representative motifs (e.g., cluster1 and cluster2 of bZIP binding family) in estimating the TF binding affinity changes is given, in which the two reprehensive motifs replace the original seven motifs that are required to be evaluated in the previous BayesPI-BAR tool. Overall, bpb3 incorporates multi-omics data from multiple samples (such as patients) to predict functional regulatory mutations. This is an easier and more efficient tool than the previous ones. Further details of the application and tutorial of bpb3 in data analysis are provided on online, please refer to (https://github.com/bpb3/bpb3) and (https://bpb3.github.io/bpb3/).

A brief user guide of BayesPI-BAR3 (bpb3) Python package:

BayesPI-BAR3 package contains a set of command line tools written in Python 3, which can be executed automatically based on a new configuration file.

I. Run whole pipeline

All demos in the package can be run either directly or through a module "bpb3 run_pipeline" that designed for the execution of the whole analysis pipeline. Below is a step-by-step description of the pipeline that runs in the following order:

1. *Bpb3 differential_expression*: This module of the package is for identifying a set of differentially expressed genes. Input: two sets of RNA-Seq gene expression data files, one for patients and the other for normal controls; the length of each gene in the reference genome must be provided. Output: a list of differentially expressed genes.
2. *Bpb3 gene_regions*: This module can be used for selecting DNA regions near TSS of differentially expressed genes. Input: a list of differentially expressed genes and a gene annotation file (GTF format). Output: a BED file with selected regions of interest.
3. *Bpb3 mussd*: This module can be used for running MuSSD algorithm to find patient mutation hot regions. Input: A Variant Call Format (VCF) file with somatic mutations for each patient, reference genome, regions of interest from the step 2. Optionally, VCF files with germline variants can be provided as well, in which case they will be used to generate personal reference sequences for every patient. Output: text files with the information of patient mutation blocks; for each patient mutation block, it creates a BED file which contains mutation location, a FASTA file with reference sequence, a FASTA file with mutated alternate sequences of each patient, and a TSV file listing all mutations in the block; finally, a summary information of all blocks, mutations, and patients.
4. *Bpb3 make_cluster4pwm:* This module makes input PWMs files for bpb3 by using clustered PWMs from abc4pwm. Input: A folder that contains all PWMs. Output: PWMs in their respective clusters and representative motifs inside each cluster.
5. *Bpb3 highly_mutated_blocks*: This module is used for identifying significantly mutated patient mutation blocks (blocks that are mutated more than average) based on random expectation by chance. This is an additional filter if there are too many patient mutation blocks recovered by MuSSD. Input: results from step 3. Output: a name list of highly mutated patient mutation blocks.
6. *Bpb3 bayespi_bar*: a new implementation of BayesPI-BAR algorithm. Input: reference/alternate sequence pairs (a pair of FASTA files); a set of position weight matrices (PWMs) for TFs of interest (a default set of 1772 PWMs from (Kheradpour and Kellis, 2014) is included in the package), which are in BayesPI MLP format. Output: a table of δdbA scores for each PWM in every patient-specific mutation block.
7. *Bpb3 choose_background_parameters*: Next, this module is used for computing background mutation model. Input: results from module *mussd*, the regions of interest, the reference genome, and a set of PWMs that was used in step 5. Optionally, a mutation signature file can be provided, which specifies the probability for mutations in each possible k-mer. If it is provided, the background mutations will be generated according to the given probability distribution; otherwise, all possible mutations will have equal probabilities. Another option for background generation is to directly provide a set of background mutations, which normally will be derived from tumor samples. If these mutations are given, they will be randomly sampled to generate background mutation blocks. Only mutations located inside regions of

interest are used. Output: creates a shell script that calls *bayespi_bar.py* to compute δdbA scores in random background model; after executing the shell script, a table of δdbA scores for each PWM in background model is produced.

8. *Bpb3 affinity_change_significance_test*: For the testing of significance of TF binding affinity changes between the patient blocks and the background mutation model we use this module. Input: a pair of *bayespi_bar* module results (one for a patient mutation block, and the other for a background model). Output: a TSV file containing a table of significantly affected TFs in the patient block.
9. *Bpb3 funtion2filter_results_by_gene_expression*: This module is for removing predicted TFs with very low gene expression. Input: a table of significantly affected TFs and their gene expression file. Output: a table of remaining TFs after the filtration. There is another module for clustered PWMs as well which is designed to handle clustered PWMs with name *filter_results_by_gene_expression_cluster4pwm.* This module filters those TF whose expression is too low in clustered PWMs. demos are provided.
10. *Bpb3 function2make_plots*: This module creates a summary plot of predicted mutation block. Inputs: the background and foreground results from *bayespi_bar* and a table of significantly affected TFs. Output: a heat map in PNG format.
11. *Bpb3 bpb3selectedPWM:* This module is the second level analysis of bpb3, to evaluate PWMs of the selected top N clustered PWMs from the first level analysis of bpb3 by using the clustered PWMs.
12. *Bpb3 clean_tmp:* This module is called in the end to remove temporary files produced both in the background and foreground calculations.
13. *Bpb3 compute_mutation_socre:* compute mutation score for SNP or mutation block based on the absolute mean scores of the top N ranked (both positive and negative changed) TFs.

BayesPI-BAR3 package also provides simple error checking, logging, and it consolidates the pipeline parameters in one place. A description of command line options can be obtained by typing the *--help* option. All input and output data files and their types are described in the package help menu, which are easy to inspect and modify. The default module, *run_pipeline*, runs the whole pipeline in the sequence. Evaluation of the genome-wide skin cancer, follicular lymphoma and pre-defined differential methylation regions were performed and provided as demo. More information of bpb3 please refer to (https://bpb3.github.io/bpb3/).

II. Input data in bpb3

To apply bpb3 package on real mutation dataset, following files are needed: VCF file per patient containing somatic mutations, the gene expression data for patients (e.g., RNA-Seq count files), the control gene expression data, a reference genome FASTA file, and a gene annotation GTF file. Before running bpb3, a user should provide the path of input data and adjust pipeline parameters in the configuration file that given with the package, as well as to provide the number of parallel cores in "*parallel_options.txt"* if parallel computation is needed. After the configure file is ready, run the program by typing "*bpb3 run_pipeline --import_bpb3_config bpb3_config_file_name.py"* in the terminal. The tool will write a text log to track the progress and errors (e.g., missing input data). The package is designed to be robust against interruptions: for example, if the package stops at some point, it can be restarted again from the place where it had been stopped. In BayesPI-BAR3 Python package, there is also a demo script to reproduce all results from analyzing skin cancer data, Follicular Lymphoma data and Predefined Differentially Methylated Regions data. The demo scripts can be run directly

without any modifications. More detailed user guide and demos, including usage information for each module in the package, is available at the website (https://bpb3.github.io/bpb3/).

III. Information of Cloud computing – In the current work, all of coding processes were performed at a high-performance computer SAGA that placed at NTNU in Trondheim Norway, but we are physically located in Oslo Norway. SAGA has a computational capacity of 140 million CPU hours a year, which has 200 standard computer nodes with 40 cores and 193 GiB memory each. It also has 8 big memory nodes with 40 cores and 384 GiB of memory each. More detailed information is available at (https://documentation.sigma2.no/hpc_machines/saga.html). For that reason, bpb3 can be installed and run by Python in a remote machine such as SAGA or cloud servers provided by Amazon Web Services, Google Cloud platform or Microsoft Azure etc. Though the configuration of cloud may vary between different companies, there are a few common things for running bpb3 on cloud servers:

1. *Virtual Environment:*
   Create a virtual environment for the package e.g., venv_bpb3. Check if all necessary dependencies of the package are available so that this package can be isolated from the rest of the system and can run smoothly. Here, we use "miniconda" as an example

   ```
   conda create -venv bpb3_env
   conda activate bpb3_env
   ```

2. *Install dependencies*:
   Install dependencies of the package inside the environment. It is recommended and helpful because this will avoid any compatibility issue between the package libraries and home system libraries. For example, install bedtools:

   ```
   conda install bedtools
   ```

   Repeat above command for the following dependencies:
   Setuptools, itertools, pandas, numpy, argparse, os, shutil, multiprocessing, matplotlib, seaborn, datetime, scipy, tempfile, time, numba

3. *Install the Python package*:
   Next step is to install the bpb3 package inside the virtual environment venv_bpb3. Change directory to the root directory of provided package and install by typing the command:

   ```
   python setup.py install
   ```

4. After installing bpb3 package, system might ask you to install any missing dependencies. Install all that are prompted.
5. The package would have been now successfully installed. You can try following command to start using your package on cloud:

   ```
   bpb3 -h
   ```

Some cloud providers may have specific requirements, the aforementioned steps are sufficient in most of cases to run bpb3 on a cloud machine. The cost for CPU hours and memory depended on the cloud service provider. Here, an example of CPU hours and memory that used by bpb3 for the current work is given in supplementary Table S4, which may help users estimate the cost of cloud computing according to the commercial providers.

References:

1. Wang, J. (2009). BayesPI-a new model to study protein-DNA interactions: a case study of condition-specific protein binding parameters for Yeast transcription factors. BMC bioinformatics *10*, 1-17.
2. Mackay, D.J.C. (1992). Bayesian methods for adaptive models (California Institute of Technology).
3. Wang, J., and Batmanov, K. (2015). BayesPI-BAR: a new biophysical model for characterization of regulatory sequence variations. Nucleic acids research *43*, e147-e147.
4. Batmanov, K., Delabie, J., and Wang, J. (2019). BayesPI-BAR2: a new python package for predicting functional non-coding mutations in cancer patient cohorts. Frontiers in genetics *10*, 282.