Published in final edited form as:

J Appl Stat. 2018 ; 45(14): 2658–2676. doi:10.1080/02664763.2018.1432577.

Genomic Feature Selection by Coverage Design Optimization

Stephen Reid^{a,*}, Aaron M. Newman^b, Maximilian Diehn^b, Ash A. Alizadeh^b, and Robert Tibshirani^a

^aDepartment of Statistics, Stanford University, 390 Serra Mall, Stanford, CA, USA

^bDivision of Oncology, Department of Medicine, Stanford University, Stanford, CA, USA

Abstract

We introduce a novel data reduction technique whereby we select a subset of tiles to “cover” maximally events of interest in large-scale biological datasets (e.g., genetic mutations), while minimizing the number of tiles. A tile is a genomic unit capturing one or more biological events, such as a sequence of base pairs that can be sequenced and observed simultaneously. The goal is to reduce significantly the number of tiles considered to those with areas of dense events in a cohort, thus saving on cost and enhancing interpretability. However, the reduction should not come at the cost of too much information, allowing for sensible statistical analysis after its application. We envisage application of our methods to a variety of high throughput data types, particularly those produced by next generation sequencing (NGS) experiments.

The procedure is cast as a convex optimization problem, which is presented, along with methods of its solution. The method is demonstrated on a large dataset of somatic mutations spanning 5000+ patients, each having one of 29 cancer types. Applied to these data, our method dramatically reduces the number of gene locations required for broad coverage of patients and their mutations, giving subject specialists a more easily interpretable snapshot of recurrent mutational profiles in these cancers. The locations identified coincide with previously identified cancer genes. Finally, despite considerable data reduction, we show that our covering designs preserve the cancer discrimination ability of multinomial logistic regression models trained on all of the locations ($> 1M$).

Keywords

genomics; non-convex optimisation; feature selection; multinomial logistic regression; mutation coverage

1. Introduction

In this paper, we consider a statistical design problem: we wish to choose a set of features or ‘tiles’, representing a window of contiguous items that we can observe simultaneously (say, as adjacent genomic regions spanning defined base pairs in a set of chromosomes). The observed items are investigated for their properties (for example, to see whether base pairs

*Corresponding author: sreid1652@gmail.com.

are mutated or not) and these properties interpreted in terms of some underlying property (for example, the patient has a certain type of disease, such as a cancer subtype, or a disease predisposition, such as a heritable syndrome). We are initially confronted with a vast number of potential tiles from which to choose, given the size of the genome and the feature space. However, due to cost, time and interpretability constraints, we wish to limit drastically the number of tiles placed. Equipped with a small set of more cheaply and accurately observable genomic segments, one could, for example, present an easily interpreted snapshot of cancer genomic profiles, or attempt the design of cheaper, non-invasive, high-sensitivity cancer detection technology. Here, we consider general methods for selecting effectively such a limited number of tiles, toward maximally covering biological features of interest without surrendering broad sample coverage (e.g., patients in a clinical study).

Given the generality of our methods, we anticipate diverse applications in basic and translational genome research. For example, our approach has utility for (i) the prioritization of recurrent somatic, germline, or epigenomic markers for focused NGS analyses: Rehm [14], Newman et al. [12], Welander et al. [19], Deng et al. [5], and (ii) the selection of small sets of differentially expressed genes for targeted capture experiments (Mercer et al. [11]). Relevant data types include high throughput genomic profiles, such as those generated by The Cancer Genome Atlas (TCGA), heritable variants defined by Genome-Wide Association Studies (GWAS), and genotype to phenotype data characterized by Quantitative Trait Locus (QTL) experiments. Here, we motivate and demonstrate our approach using publicly available cancer somatic mutation data generated by TCGA.

Suppose we have a group of n randomly sampled patients, all sharing some characteristic, say, having one from a set of cancer types. For each patient i , we have a list, of length N_i , of observed mutated base pair locations, making a total of $\sum_{i=1}^n N_i = N$ across all patients.

Here we count unique mutation-patient pairs, so that a mutation of a base pair in two different patients is counted twice. We have at our disposal a technology allowing us to sequence accurately a small set of contiguous base pairs at a time. Fixing ideas, suppose that the length of such a window (or “tile”) is $\beta = 200$ bps. There are p (very large) possible locations along the genome at which we can center our measuring instrument, sequencing the 100 base pairs on either side of this center location. Once the segment is sequenced, we can determine whether each of the constituent base pairs is mutated or not. Our goal is to extract a limited number (say $R \ll p$) of locations along the genome such that the resulting set of observed segments (hereafter called “tiles”) covers as many of the observed mutations for as many patients as possible.

A simple schematic for the procedure is presented in Figure 1. There are two patients (labeled $p1$ and $p2$), each with “genome” of length 12 base pairs, represented by a row in one of the top four panels. The patients share 7 mutations amongst them (labeled ‘a’ to ‘g’). Tiles – representing the window of contiguous base pairs that can be observed at one time – are of length 5 base pairs and shaded in light blue. Tile centers are in darker blue. In the top left panel, we place a tile with centre at position 3, covering 5 distinct mutations over the two patients. Similarly, the top right, centre left and centre right panels represent tiles placed at positions 5, 8 and 11 respectively. As we slide the tile centre from position 1 toward

position 12, we can record the mutations covered by each tile and represent the information in the mutation coverage matrix in the bottom panel. Each distinct mutation has a row of length 12 (the genome length) and has a cross at entry j if a tile placed with centre at position j covers that mutation. We can cover all mutations by placing two tiles: one at position 3 and one at position 11. Position 8 will be avoided.

We approach this as a *set cover* problem. First, we construct a matrix X of size $N \times p$, with $X_{lj} = 1, l = 1, 2, \dots, N; j = 1, 2, \dots, p$, if a tile centered at location j covers (i.e. contains in its sequenced segment) the l^{th} observed mutation location. $X_{lj} = 0$ otherwise. X is a binary matrix, called the *mutation coverage matrix* and is likely to be very large, albeit rather sparse, in applications. The mutation covering algorithms select a much smaller set of columns (called “tile centers”) representing dense mutation locations. It looks for columns in X with many 1s, subject to constraints ensuring that the same mutations and patients are not identified repeatedly. Mutation “density” then is measured as the count of mutated base pairs, across patients, observed within the window of a placed tile. In the lung adenocarcinoma application of Section 4, for example, tiles cover on average 2.3 mutated base pairs, with a median of 2. Approximately 41% of tiles cover only 1 mutation. These are candidates to be ignored by the procedure. 5.5% of tiles cover 5 or more mutations, some covering as many 112 – obvious candidates for the reduced set.

This is an *unsupervised* problem, absent of a response variable, and we do not have the luxury of supervised variable selection procedures like best subset selection, stepwise regression, least angle regression (LARS) of Efron et al. [7] the lasso of Tibshirani [17] and the elastic net of Zou and Hastie [21]. We are also not interested in “interesting directions” in the data as provided by principal components analysis (PCA) or partial least squares or the sparse versions of the former, as developed by Zou et al. [22].

Rather, we would like a representative set of columns extracted directly from the original matrix. Our methodology can easily be applied in any application where the immediate goal is to find segments of the dataset where some trait or occurrence is densely present.

The problem is cast a non-convex, albeit easily solved, optimisation problem. Rapid solution of the problem is possible because its form allows us to solve it with repeated convex approximations.

This paper is organized as follows: in Section 2 we briefly discuss the real dataset that inspired the development of our methods. Section 3 presents the optimisation problems associated with our extraction methods, Section 4 a particular application to a single (lung) cancer dataset, while Section 5 presents the results of a simulation study demonstrating the behaviour of our methods on toy datasets. In Section 6 we show how our single class methods of Section 3 can be extended to handle multiple classes. Finally, in Section 7 we subject a real world, multi-cancer dataset to our methods. We conclude with a short discussion.

2. A real world motivating dataset

Our covering design algorithms were inspired by a particular real world application. Data derives from a study using genetic information to classify cancer type and pertains to patients known to have certain types of cancers. For each patient, we are given a unique (but anonymous) identifier, their cancer type and a list of positions along their genome where they are seen to have base pair mutations. There is a vast number of mutation locations in this dataset – totaling more than a million. In such a dataset, the need for identifying a reduced set of interesting genome locations become obvious.

Possible tile center locations were limited to the unique mutation locations observed in the dataset. Locations along the genome completely bereft of observed mutations need not be considered – our algorithms will never consider those to be viable tile center locations. The number of unique mutation locations thus determines p .

There were $K = 29$ different cancer types (classes). Data was split roughly 70–30 into training and test sets (split by patients). Classes in the training set ranged from having 28 to 721 patients, with a total of $\sum_{k=1}^K n_k = 5190$. The test set had classes ranging from 12 to 310 patients, with a total of 2242. Each patient had multiple mutations, leading to a total of 1, 134, 056 mutations over all training set classes. The test set had a total of 510, 335 mutations. We formed the training mutation coverage matrix for each class: $X^{(k)}$ of both the training and test sets, as described in Section 1.

There were $p = 991, 902$ initial tile center locations. Individual class matrices $X^{(k)}$ were very sparse with non-zero entries totaling less than 0.001% of all entries. The goal was to reduce these $p = 991, 902$ potential locations into a smaller set of size $R = 2000$ for the individual cancer types and to a master set of size $R_0 = 3000$. The composition of the master set was to be informed by that of the individual class sets.

3. Coverage design extraction methods

Our goal is to place tiles on a small set, of predetermined size, of *mutation rich* columns. When a column j is selected as a tile center and contains a non-zero entry in row l , then j is said to *cover* l . Thus, we select predictors that between them cover as many rows as possible. This task evokes that of *set coverage* and we draw inspiration from Bien and Tibshirani [2].

Note that, while we use methodology similar to (but not the same as) that of Bien and Tibshirani [2], our ambitions differ. They have observations, divided in to groups or classes (as do we, potentially), but wish to extract a small set of prototypes to serve as representatives of the classes. Hence they attempt to extract representative *rows* from a matrix by drawing balls of predetermined radius around each row and selecting those with balls covering as many other observations in their class as possible, while covering the least possible observations in other classes. Our methods, on the other hand, select *columns*, for which we never have such class information. We use the class information of the observations (rows) to inform our choice of “good” columns to select.

3.1 The optimisation problems

Recall that we have $N \times p$ matrix X with zero-one entries, X_{lj} and rows X_l . Starting with the simplest instance of our methodology, consider the non-convex optimisation problem in the variables α (a $p \times 1$ vector with entries α_j) and ξ (an $N \times 1$ vector with entries ξ_l):

$$\text{minimize } \sum_{l=1}^N \xi_l \text{ subject to } \quad (1)$$

$$\alpha \in \{0, 1\}^p, \quad \sum_{j=1}^p \alpha_j \leq R, \quad X_l^\top \alpha \geq 1 - \xi_l, \quad \xi_l \geq 0, \quad l = 1, 2, \dots, N \quad (2)$$

where $a > b$, if a and b are vectors, denotes element-wise inequality.

Note that this is an integer program and almost a linear program in that a simple relaxation of the constraint $\alpha \in \{0, 1\}^p$ to $0 \leq \alpha_j \leq 1, j = 1, 2, \dots, p$ makes it a convex (linear) optimisation problem. We call this an *almost linear* program.

We call α the *reducing vector*, for it represents our reduced set of tile centers extracted from the full dataset. If $\alpha_j = 1$, then column j is included in our set of selected columns (i.e. we place a tile centered at the location implied by column j), if $\alpha_j = 0$, it is not. The last constraint $\sum_{j=1}^p \alpha_j \leq R$ restricts the number of columns selected to the pre specified value R (a positive integer supplied by the user).

Intuitively, the entries of ξ tell us whether the mutations are covered or not. The objective is defined to encourage the setting of many ξ_l to zero. We cannot set $\xi_l = 0$ unless the l^{th} row of constraint (3) is met. This only happens if at least one j covers l and is selected to our smaller set of predictors. The fact that we constrain ξ_l to be non-negative encourages the solution algorithm to ignore those mutations already covered. Since we cannot drive ξ_l down further by selecting more columns to cover it, we are encouraged to look elsewhere for previously uncovered mutations. Furthermore, since we have an upper limit on the number of columns we select, we are encouraged to look for columns with maximal uncovered mutations at every step.

Note however that any grouping amongst the row observations is ignored. This problem formulation is unaware that the mutations can be grouped according to which patient they belong. It focuses solely on the coverage of the maximal number of mutations.

Suppose however that we want to incorporate the patient grouping information on the mutations. The fear is that our simple formulation may favour some patients over others. In particular, if we have asymmetric sizes for the mutation sets and mutations more densely distributed in some patients, rather than others, we may find the simple formulation picking columns covering mutations involving only these large, densely mutated patients.

In addition, define the $n \times N$ matrix M with entries $M_{ij} = 1$ if mutation j belongs to patient i and $M_{ij} = 0$ otherwise. Then let $Q = MX$ and P be an $n \times p$ matrix such that $P_{ij} = 1$ if $Q_{ij} > 0$ and zero otherwise. Intuitively, where we see X as a *mutation coverage* matrix – one defining which tiles cover which mutations – we define P as the *patient coverage* matrix, showing which tiles cover which patients. A patient is said to be *covered* by a tile if at least one mutation belonging to that patient is covered by it.

Now consider an extension to the simple formulation above, where we introduce optimisation variable vector η of size $n \times 1$ with entries η_i :

$$\text{minimize} \quad \left[\gamma \cdot \sum_{l=1}^N \xi_l + (1 - \gamma) \cdot \sum_{i=1}^n \eta_i \right] \quad \text{subject to} \quad (3)$$

$$\alpha \in \{0, 1\}^p, \quad X\alpha \geq 1 - \xi, \quad P\alpha \geq 1 - \eta, \quad \xi \geq 0, \quad \eta \geq 0, \quad \sum_{j=1}^p \alpha_j \leq R \quad (4)$$

which is an almost linear program much as before, but with the inclusion of a term and constraint to encourage coverage of patients as well as individual mutations. The tuning parameter γ is imported to allow the user to tradeoff the intensity in looking for overall mutation, as opposed to overall patient, coverage.

If $\gamma = 1$, we are in the simple formulation above and it is encouraged to look for columns with many non-zero entries, even if those entries belong to mutations mostly encountered in one or a handful of patients. However, when we set $\gamma = 0$, we are encouraged to look for columns with the greatest variety of patient membership. This form of the problem is solved similarly to the simple formulation. Only here the columns are not ranked by non-zero entry count each time, but by a weighted sum (weighted by γ) of the number of non-zero entries and the number of distinct patients with non-zero entries in the column. It is not difficult to think of cases where this formulation might produce vastly different solutions than does the simple formulation. It tends to return columns with many non-zero entries *and* a larger variety of patient membership, shying away from columns dominated by only a few of the patients. This behavior is illustrated more clearly in Section 5.

Ultimately, we were interested here in covering as many mutations per patient as possible. The reason is discussed in Section 4. As such, we prefer to set $\gamma = 1$ and then checking afterward that every patient in the training set was covered by at least one tile. Should some patients not be covered, one would decrease γ until these are also covered. Decreasing γ reduces the mutations covered per patient and we want to keep this as large as possible. This might be a good rule-of-thumb for other practitioners when setting this tuning parameter. R is set by subject specialists and depends on practical considerations like cost of sequencing and sensitivity of equipment.

A third formulation uses the Q matrix as defined above. Note that the entry Q_{ij} is the count of mutations associated with patient i covered by a tile centered at location j .

$$\text{minimize} \quad \left[\sum_{l=1}^N \xi_l + \sum_{i=1}^n \eta_i \right] \quad \text{subject to} \quad (5)$$

$$\alpha \in \{0, 1\}^P, \quad X\alpha \geq 1_N - \xi, \quad Q\alpha \geq T1_n - \eta, \quad \xi \geq 0, \eta \geq 0, \quad \sum_{j=1}^p \alpha_j \leq R \quad (6)$$

where 1_n is a vector of size $n \times 1$ with all entries equal to 1.

In this formulation, we use matrix Q , and not P , introduce another tuning parameter, T , and drop γ . We drop γ to limit the number of tuning parameters.

Notice the set of constraints $Q\alpha \geq T - \eta$, where $T - 1$ is a non-negative integer. Increasing T makes these constraints more difficult to satisfy. The intent is to create an incentive for the optimisation algorithm to find columns with large mutation counts for *each* patient (not necessarily in the same column). By setting T larger than 1, we encourage further search beyond columns merely capturing 1 or two mutations for a patient. Furthermore, we use the matrix Q of counts and not P , because we want to encourage the algorithm to find columns with relatively large counts. With a limited budget of columns to select (R), our algorithm is encouraged to find the highest mutation count columns for each patient so that it can satisfy the third set of constraints and drive down the objective. Individual mutation coverage are still encouraged only to be covered once (see second set of constraints for (5)). This might lead to a better spread of selected columns than the simple formulation.

This last formulation is not as easily solved as the former two and requires a non-convex solver. It is non-convex, but is easily and rapidly solved, in our experience. Some of the techniques for solving such a non-convex problem are discussed in a subsequent section. We do not use this last formulation in any of our applications, but study its behaviour in a simulation experiment in Section 5

3.2 Analysis of the KKT conditions

For some intuition about how columns are selected, let us consider the Karush-Kuhn-Tucker (KKT) conditions of the convex relaxation of the third formulation of Equation (5). We obtain this relaxation by replacing the constraints $\alpha \in \{0, 1\}^P$ with the constraints $0 \leq \alpha_j \leq 1$ for $j = 1, 2, \dots, p$. The Lagrangian of the (primal) problem becomes, after incorporating dual variables $\lambda_1, \lambda_2 \in \mathbb{R}^P$, $\lambda_3, \lambda_5 \in \mathbb{R}^N$, $\lambda_4, \lambda_6 \in \mathbb{R}^n$ and $\lambda_7 \in \mathbb{R}$ (and dropping subscripts on 1 vectors for ease of exposition):

$$\mathcal{L}(\alpha, \xi, \eta, \lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6, \lambda_7) = 1'\xi + 1'\eta - \lambda_1'\alpha + \lambda_2'(\alpha - 1) - \lambda_3'(X\alpha - 1 + \xi) - \lambda_4'(Q\alpha - T1 + \eta) - \lambda_5'\xi - \lambda_6'\eta + \lambda_7(1'\alpha - R)$$

A dual formulation of this problem is

$$\text{minimize } 1'\lambda_2 + R\lambda_7 + 1'\lambda_5 + T1'\lambda_6 \text{ subject to } \quad (7)$$

$$\lambda_2, \lambda_7 \geq 0 \quad 0 \leq \lambda_5, \lambda_6 \leq 1 \quad (8)$$

$$\sum_{l=1}^N (1 - \lambda_{5,l})x_{lj} + \sum_{i=1}^n (1 - \lambda_{6,i})q_{ij} \leq \lambda_7 + \lambda_{2,j}, j = 1, 2, \dots, p \quad (9)$$

where x_{lj} and q_{ij} are the l^{th} and i^{th} entries of X and Q respectively.

Optimal points of these problems (denoted by * as variable name superscript) satisfy the KKT conditions:

$$\sum_{l=1}^N \lambda_{3,l}^* x^{(l)} + \sum_{i=1}^n \lambda_{4,i}^* q^{(i)} = \lambda_7^* 1 + \lambda_2^* - \lambda_1^* \quad (10)$$

$$\lambda_5^* = 1 - \lambda_3^*, \quad \lambda_6^* = 1 - \lambda_4^*, \quad \lambda_{1,j}^* \alpha_j^* = 0, \quad \lambda_{2,j}^* (1 - \alpha_j^*) = 0 \quad (11)$$

$$\lambda_{3,l}^* (x^{(l)'} \alpha^* - 1 + \xi_l^*) = 0, \quad \lambda_{4,i}^* (q^{(i)'} \alpha^* - 1 + \eta_i^*) = 0 \quad (12)$$

$$\lambda_{5,l}^* \xi_l^* = 0, \quad \lambda_{6,i}^* \eta_i^* = 0, \quad \lambda_7^* (1'\alpha^* - R) = 0 \quad (13)$$

where $x^{(l)}$ and $q^{(i)}$ are the l^{th} and i^{th} rows of matrices X and Q respectively.

The intuition is now clear from the dual objective (7): there is an incentive to set the entries of λ_5 and λ_6 to zero, the latter more so as T is increased. However, decreasing some of these quantities makes the inequalities (9) more difficult to meet. The algorithm must then decide which of the λ_5 and λ_6 entries must be set to zero. Notice that from Equation (13) that the

setting of these entries to zero allows the corresponding ξ and η entries to be non-zero, which, in the primal problem, is seen to imply that we leave the corresponding mutations in the matrices X and Q uncovered. The algorithm thus decides to set those entries of λ_5 and λ_6 to zero that would leave the inequalities (9) most easily met. It makes intuitive sense that it will choose those l and i with (in general) the smallest entries x_{lj} and q_{ij} .

Problem constraints cannot be met without increasing either λ_7 or some of the $\lambda_{2,j}$ (which again increases in the dual objective). In fact, it is most likely to increase the $\lambda_{2,j}$ corresponding (again in general) to the inequalities with larger x_{lj} and q_{ij} . Now, from the KKT conditions (11), we see that $\lambda_{2,j}^* > 0 \Rightarrow \alpha_j^* = 1$, which reveals the algorithm's tendency to select the columns with largest mutation and patient counts.

3.3 Algorithms for solving the column reduction problems

Although there seem to be reasonably straightforward methods of solution to at least the first two formulations above, all three still belong to the class of *Mixed Integer Linear Programs*, which are deemed to be NP-hard, with the concomitant pessimistic outlook on their efficient solution, especially at scale. However, as has been alluded to above, these problems are very close to standard convex linear programs. This close connection is leveraged and many rapid algorithms exist for their solution, even at scale. Note that these methods produce *exact* solutions. Examples include generalized Benders decomposition (Geoffrion [8]), branch-and-bound (Gupta and Ravindran [9]), outer approximation (Duran and Grossman [6]), LP/NLP-based branch-and-bound ([13]), the extended cutting plane method (Westerlund and Pettersson [20]), branch-and-cut (Stubbs and Mehrotra [15]) and some hybrid methods (Abhishek et al. [1], Bonami et al. [3]). A good review of convex and non-convex mixed integer programs is provided by Burer and Letchford [4].

To give the reader a flavor of how a solution is obtained, we will briefly discuss the branch-and-bound method. This method requires two components: one providing an upper bound for the optimal value of the objective over a given subset of the feasible set and one providing a lower bound. Upper bounds are easily obtained – we merely plug in a feasible point. Lower bounds are obtained, for example, by forming a convex relaxation to the problem and solving that at each step. Convex optimisation problems are solved in polynomial time. At each step, we partition the feasible set into convex sets and form these upper and lower bounds. If, in a given subset, these are close enough to each other (say within some pre specified tolerance), we return the objective value and either point producing one of the bounds. If the bounds are not close enough, we further partition this subset and repeat. The algorithm:

1. Set tolerance $\epsilon > 0$ and start with a set S_0 over which we find the upper (U_0) and lower (L_0) bounds of the objective function.
2. If $U_0 - L_0 < \epsilon$, return.
3. Split S_0 into $S_0 = S_1 \cup S_2$ (two rectangles) and compute upper (U_i) and lower (L_i) bounds on S_i , $i = 1, 2$. One such branching method is described in the paragraph below.

4. Update overall upper (U^*) and lower (L^*) bounds: $L^* = \min\{L_1, L_2\}$ and $U^* = \min\{U_1, U_2\}$. If $U^* - L^* < \epsilon$, return.
5. Repeat steps 3 and 4, using either S_1 or S_2 instead of S_0 .

Pruning strategies also exist to speed up computation. In particular, we can prune any subset with upper bound greater than the current overall upper bound U^* .

For our mixed integer (actually, binary) programs, the lower bound is found merely by forming the convex relaxation as described before. The upper bound is gleaned in one of a few ways. The simplest here is simply to round the non-binary elements of the solution of the convex relaxation to 0 or 1, whichever is closer. Branching is done by picking an index at random (call it j^*) and solving two (smaller) problems: one where we constrain $j^* = 1$ and one where $j^* = 0$.

The upshot is that we can solve the non-convex integer program globally via the solution of a succession of convex optimisation problems. These are all linear programs and can be very rapidly solved at scale. The hope is that the global solution requires only a handful of convex approximations before convergence. Our experience is that convergence for our formulations above is usually quite rapid.

4. Small example – lung adenocarcinoma

4.1 Base pair level

As a demonstration of the efficacy of our coverage procedure, we take a short detour and consider a small, real application: a dataset comprised of patients known to have lung adenocarcinoma. Patients are split into training and test sets, consisting of 381 and 150 patients, sharing 158822 and 52640 mutations among them, respectively. The goal was to place $R = 1000$ tiles, each of width $\beta = 175$ base pairs, in order to cover as many mutations per patient as possible.

Four coverage procedures were applied to the training set, each hunting for sections of the genome they deem “dense” with mutations. First, **our coverage procedure** of (3), using $\gamma = 1$. Other values of γ were attempted, with similar, but slightly worse results. Second, **a naive-greedy mutation counting procedure**. We count the number of patients having a mutation at a given base pair. Tiles were placed at the $R = 1000$ highest count locations (greedy) regardless of whether a location was already covered by a tile placed at a higher count location. Further, **a naive-greedy tile coverage procedure**, where we summed over the rows of the X matrix, finding the columns with highest non-zero count and using the top $R = 1000$ as tile centers. This method can differ markedly from the mutation counting procedure. If we have a single base pair location, mutated across a moderate number of patients, it may well become a tile centre in the counting approach. However, if this base pair is fairly remote in the genome from other mutations, the location may not make it into the top group for this approach. Rather, a segment of genome with a handful of tightly mutated locations (say, where a tile covers three or four mutations), even over only a small number of patients, could still make it into the top group at the expense of the former

location. Finally, a **greedy (non-naive) version of the tile coverage procedure**, where we ensure that tile centers have not been covered by other tile centers.

Methods are compared on the distribution of the number of observed mutations covered per patient by the $R = 1000$ tiles (both over the training and test sets). The higher the median number of mutations covered per patient; the lower the cancer detection threshold, allowing us more sensitivity to detect cancers early by considering only this comparatively tiny subset of genome locations. Details are discussed in Newman et al. [12].

Figure 2 plots a version of the sample complementary cumulative distribution function (CCDF) of the number of unique mutations covered by the different tile sets, over both the training and the test set: $\tilde{F}(x) = \frac{1}{n} \sum_{i=1}^n I\{M_i \geq x\}$ where n is the number of patients in the dataset and M_i the number of patient i 's mutations covered by our tiles. We plot this function for $x = 0, 1, 2, \dots, 20$. Table 1 summarizes the median mutations covered for each (where the CCDFs cross the 0.5 mark). Our method substantially outperforms the other methods using the same genomic space, $\sim 175\,000$ base pairs.

4.2 Exon and gene level

Most previous research has focused on gene level mutations – a much coarser granularity than what we are considering. Examples of such research include Vandin et al. [18] and the references therein. These authors consider mutations at a gene level and seek sets of genes seen to be frequently mutated across many patients, subject to limits on the overlap these sets allow. The output of their methods – they describe two: a greedy method and one based on Monte Carlo Markov Chain (MCMC) sampling – is a set of G genes deemed potentially important for cancer development. The parameter G is given to the algorithm beforehand.

It should be noted that our method is perhaps more ambitious in that it attempts to pinpoint specific collections of base pairs that are to be subjected to further scrutiny. However, once we have identified these collections (tiles), we can easily identify the gene to which they belong. Indeed, since our procedure operates at the base pair level, we can consider structures at finer granularity than genes, such as exons. Exons have lengths that correspond more closely to the size of our tiles. Our method can thus also be used to identify exons or genes with potential functional significance in cancer.

For each tile placed by our method in the lung adenocarcinoma dataset, we could identify the overlapping exons, as well as a list of patients with mutations covered by the tile along each exon. Ranking the count of unique patients with covered mutations within an exon (normalised by the overall length of overlap of placed tiles with that exon), we can identify a list (here of size $G = 20$) of potentially predictive exons. The exons themselves may be considered, or their surrounding genes.

Furthermore, we can apply the greedy and MCMC procedures of Vandin et al. [18] to this data, with $G = 20$ and quote the resulting gene sets. We defer the detailed analysis of the lung adenocarcinoma data to Section 3 of the Supplementary Material. See Section 7.2 of the main text for a detailed exon and gene level analysis for a pan cancer study.

Of course, our procedure does not take gene or exon labels into account a priori. With $\gamma = 1$, it seeks out tile locations covering the most mutations. Gene labels can also be taken into account by defining a *gene coverage* matrix H , of size $g \times p$, where g is the number of unique genes seen to have at least one mutation along it in the dataset. $H_{ij} = 1$ if gene i is covered by tile j . Adding a penalty term to the objective in (3) – perhaps $\sum_{i=1}^g \phi_i$, with its own tuning parameter – and additional constraints $H\alpha = 1 - \phi$ would incorporate gene label information into the optimization procedure. The procedure will be encouraged to cover a wide array of genes in its search for specific tile locations. Space considerations preclude us from pursuing this further here.

5. A simulation study

Here we conducted a simulation experiment showing how different settings of the tuning parameters in the formulations above can lead to qualitatively different tile center sets. We thought such a simulation useful in the demonstration of the versatility of our methods. Due to space considerations, we defer the discussion to Section 2 of the Supplementary Material.

6. Introducing labelled classes

Suppose now that, instead of a single binary matrix X , we have $K > 1$ of them: $X^{(k)}$. Each matrix is of size $N_k \times p$, with the rows in matrix $X^{(k)}$ representing mutations divided amongst n_k patients. A representative sampling scheme might be that of K classes or cancer types, with n_k , $k = 1, 2, \dots, K$ patients within each class, with each patient being associated multiple mutations.

Furthermore, suppose we would like to obtain a master tile center set from the entire set of coverage matrices. Not only do we want tile center sets for the individual $X^{(k)}$, but we also want to amalgamate these sets into one master tile set, the membership of which is informed by the class information as well as the structure of the matrices in each class. An example application could be to use the master set for overarching interpretability and perhaps to perform a first round classification into one of the K classes. The hope here would be that the master set, although of size much less than the overall number of predictors, retains enough of the class information to discriminate between them effectively. Discrimination between classes can be facilitated by augmenting the genetic features from the master set with other clinical features. These include, perhaps, age, gender, etc and might be included as additional predictors in a classification model as in Section 7.3. The construction of such a classifier is beyond the scope of the current paper, but is a promising vein of future research.

Once an initial classification is done (say to class k^*), the reduced set for that class can then be used for more fine-grained interpretation and perhaps further statistical analysis. This section presents the optimisation problem, while the next discusses some applications of the procedure.

6.1 Methods for reduction and amalgamation

In this section we discuss three reduction-amalgamation methods. The first ignores the class labels and merely distills a set of tile centers. The second, arrives at tile center sets for each class separately and then amalgamates in an ad hoc manner. Finally, we consider a slightly modified optimisation problem that seamlessly integrates the two processes. We discuss these multi-class extensions only for the formulation of Equation (3). The other formulation (Equation (5)) follows by the same logic. Furthermore, our real-world application (discussed in the next section) is based on the former.

The first reduction-amalgamation method, which we will call the *overall* method, merely stacks all the matrices $X^{(k)}$ into one large $\sum_{k=1}^K N_k \times p$ matrix X and runs the coverage design algorithms discussed above. Note that this method produces only a master tile set, represented by the binary vector $\alpha^{(0)} \in \mathbb{R}^p$, and not separate sets for each class.

The second method, dubbed the *separate* method, runs the above coverage design algorithms separately (and in parallel) for each of the classes, producing reduced tile sets, represented by binary vectors $\alpha^{(k)} \in \mathbb{R}^p$, for each class. For every column index j we can then find the sum $\sum_{k=1}^K \alpha_j^{(k)}$ and select the R indices corresponding to the largest of these sums (breaking ties by sorting and selecting randomly) to the master set, $\alpha^{(0)}$.

Perhaps a more principled approach is to define an objective that incorporates both reduction and amalgamation simultaneously. Individual class tile sets are then constructed with heed being taken of the master tile set. To this end, consider the following optimisation problem:

$$\begin{aligned} & \text{minimise } \sum_{k=1}^K \left[\gamma \cdot \sum_{l=1}^{N_k} \xi_l^{(k)} + (1-\gamma) \cdot \sum_{i=1}^{n_k} \eta_i^{(k)} \right] \\ & + \sum_{k=1}^K \left[\delta \cdot \sum_{j=1}^p (\alpha_j^{(k)} - \alpha_j^{(0)})_- + (1-\delta) \cdot \sum_{j=1}^p (\alpha_j^{(k)} - \alpha_j^{(0)})_+ \right] \quad \text{subject to} \end{aligned} \quad (14)$$

$$\alpha^{(k)} \in \{0, 1\}^p, \quad \sum_{j=1}^p \alpha_j^{(k)} \leq R_k \quad k = 0, 1, \dots, K \quad (15)$$

$$X^{(k)} \alpha^{(k)} \geq 1 - \xi^{(k)}, \quad P^{(k)} \alpha^{(k)} \geq 1 - \eta^{(k)}, \quad \xi^{(k)} \geq 0, \quad \eta^{(k)} \geq 0, \quad k = 1, 2, \dots, K \quad (16)$$

where, as expected, $\xi^{(k)} \in \mathbb{R}^{N_k}$ and $\eta^{(k)} \in \mathbb{R}^{n_k}$. Note that, in principle, we can have different reducing vector sizes R_k for each class and the master. However, for simplicity, we usually set $R_1 = R_2 = \dots = R_K = R = R_0$, leaving us with four tuning parameters: γ , δ , R_0 and R . We define $(x)_+ = \max\{x, 0\}$ and $(x)_- = \max\{-x, 0\}$.

The above problem is a relatively simple extension of single matrix case, as can be seen by fixing k and considering any term in the sum in the objective. We still have terms encouraging the coverage of individual mutations (as well as at least one row in each group), with the addition of terms ensuring coherence to the current master tile set. The particular form for these terms is chosen for the simple and intuitive implications it has for the updating of the master tile center set. This is discussed next. We call this the *combined* problem.

6.2 Solving the combined problem

Solution of the combined problem follows iteratively. First we fix a value for $\alpha^{(0)}$ (say $\alpha^{(0)} = 0$). Now rewrite the problem above in the equivalent form

$$\text{minimise } \sum_{k=1}^K \left[\gamma \cdot \sum_{l=1}^{N_k} \xi_l^{(k)} + (1-\gamma) \cdot \sum_{i=1}^{n_k} \eta_i^{(k)} + \delta \cdot \sum_{j=1}^p s_j^{(k)} + (1-\delta) \cdot \sum_{j=1}^p t_j^{(k)} \right] \quad \text{subject to} \quad (17)$$

$$\alpha^{(k)} \in \{0, 1\}^p, \quad \sum_{j=1}^p \alpha_j^{(k)} \leq R_k, \quad k = 0, 1, \dots, K \quad (18)$$

$$X^{(k)} \alpha^{(k)} \geq 1 - \xi^{(k)}, \quad P^{(k)} \alpha^{(k)} \geq 1 - \eta^{(k)}, \quad \xi^{(k)} \geq 0, \eta^{(k)} \geq 0, \quad k = 1, 2, \dots, K \quad (19)$$

$$-s_j^{(k)} \leq \alpha_j^{(k)} - \alpha_j^{(0)} \leq t_j^{(k)} \quad s_j^{(k)}, t_j^{(k)} \geq 0 \quad (20)$$

which is a mixed integer (binary) linear program and is easily solved as discussed above.

Now, fixing the $\alpha^{(k)}$, remembering that all $\alpha^{(k)} \in \{0, 1\}^p$ and rewriting the part of the objective involving $\alpha^{(0)}$, we obtain, for fixed j :

$$\begin{aligned} & \sum_{k=1}^K \delta \cdot \sum_{j=1}^p (\alpha_j^{(k)} - \alpha_j^{(0)})_- + (1-\delta) \cdot \sum_{j=1}^p (\alpha_j^{(k)} - \alpha_j^{(0)})_+ = \delta \cdot \sum_{k: \alpha_j^{(k)} = 0} \alpha_j^{(0)} + (1-\delta) \cdot \sum_{k: \alpha_j^{(k)} = 1} (1 - \alpha_j^{(0)}) \\ & = (1-\delta) \cdot \# \{k: \alpha_j^{(k)} = 1\} + \alpha_j^{(0)} \left(K \cdot \delta - \# \{k: \alpha_j^{(k)} = 1\} \right) \end{aligned}$$

from which we see that the objective is minimized by setting $\alpha_j^{(0)} = 1$ if $K \cdot \delta < \# \{k: \alpha_j^{(k)} = 1\}$ and to zero otherwise. We remember to obey the maximum reduced set size constraint. Notice that this implies that the constituent tile sets each get a “vote” in deciding whether the master set gets that index or not, with δ controlling the minimum proportion of votes required to merit inclusion. Once we have performed this step, we again take α^0 as fixed and resolve the mixed integer linear program above, alternating until convergence.

Since the objective is not convex, we cannot guarantee convergence to the global optimum. The user can use a number of random starts for $\alpha^{(0)}$ and use the solution with smallest objective. However, we have found that $\alpha^{(0)} = 0$ works well, leads to rapid convergence after only a few alternating steps and produces reduced tile sets with reasonable performance. Setting up the solution algorithm in the way described allows one to solve constituent mixed integer linear programs *in parallel*, with a brief call back for the voting step and then returning to parallel mode, etc. Upon convergence, we have reduced sets for each constituent matrix ($\alpha^{(k)}$) and a master set ($\alpha^{(0)}$).

7. More on the real world example

We have already introduced the reader to the real world inspiration behind our methods. This was done in Section 2. The reader is referred to that section to remind themselves of the nature and scope of the problem. Here we present results of a particular column reduction and post-reduction statistical analysis.

Each of the *overall*, *separate* and *combined* methods was applied to the training set, producing master sets and individual reduced sets (where applicable). These were then used for interpretation. Solutions were found using the proprietary *Gurobi* software on a shared computing cluster, using 16 parallel nodes. Timings are difficult to measure, but at each parameter setting, a single sweep to compute all the $K = 29$ individual reduced sets for a given master set took roughly between 90 and 180 seconds, while 12 to 20 alternating iterations were usually enough to ensure convergence of the objective function. We found that $\gamma = 1$ and $\delta = 0.01$ worked well for the application.

7.1 Mutation coverage

First we consider only the master selector $\alpha^{(0)}$ derived from the *overall* procedure and compare its mutation covering performance against tile coverage procedures, as described in Section 4.1. For each of the naive and greedy tile coverage procedures we constructed two tile index sets: one using training data on *all* $K = 29$ cancers and one using only data pertaining to the specific cancer in question. All tile index sets were of size $R_0 = 3000$. Table 2 shows the median number of mutations covered per patient in the *test* set for each of the 8 cancers with most observed patients and for the whole dataset comprising the $K = 29$ different cancers.

Conclusions are very similar to those in Section 4.1: our procedure tends to cover more mutations per patient in general. However, there are some cancers where performance is better than others. An index for associating a cancer code in the table to actual cancer name is given in Table 4 of the Supplementary Material.

7.2 Exon and gene level selection

One could also use the *overall* master tile set to perform an exon and gene level analysis as in Section 4.2. The master tile set has size $R_0 = 3000$.

We compare the ability of our method to identify cancer genes with those of Vandin et al. [18]. For our tile set, we rank exons based on the number of unique patients with covered mutations along that exon (normalized for the length of overlap of all tiles with that exon), picking the top 20. Table 3 summarises results.

We note that many of the genes highlighted by the master tile set are so-called “high-confidence driver” (HCD) genes described by Tamborero et al. [16]. There are 13 of them, which is somewhat more than the 7 found by the greedy method and the 5 found by the MCMC method. None of the methods highlight any of the spurious cancer genes indicated by Lawrence et al. [10], suggesting that our method might be used to identify previously unidentified cancer drivers (see Figure 6 of Supplementary Material).

There are 4 genes highlighted by all three methods: *BRAF*, *NRAS*, *PIK3CA* and *TP53*. The greedy and MCMC methods also highlight *VHL*. The good performance of the master tile set in highlighting HCDs, without highlighting the spurious genes of some other methods, suggests that the unbolded genes in the fourth column of Table 3 might be good candidates for further investigation of their cancer predicting properties.

The reader might note that analysis for the competing greedy and MCMC methods is done at the gene level, but at the exon level for our method. This apparent inconsistency is introduced so as to show the competing methods in the best light with respect to the detection of previously defined HCDs and ignoring spurious genes. For completeness, we have performed an exon-level analysis of the competing methods and quoted the results in Tables 2 and 3 of the Supplementary Material. We note that ability to select HCDs seems to be severely hampered for these methods at the exon level.

7.3 Predictive performance of distillation sets

Even though our method was not explicitly designed to aid a classification task, one could use distilled sets as (reduced) inputs to further statistical analysis, including cancer classifiers. The dimension reduction inherent in the distillation set allows perhaps for more efficient fitting of classifiers. Moreover, classifiers that could previously not be fit feasibly due to excessive predictors could perhaps now be fit. The question is whether the distilled sets retain discriminative power amongst cancers, despite the dramatic reduction in dimension. We address this concern in Section 3 of the Supplementary Materials. The reader is encouraged to glean details there.

Figure 3 presents the output of the analysis. The Rank proportion plot (RPP) measures the efficacy of a classifier at predicting the true label in a multi class setup (over a test set of previously unseen patients). We define it at rank r :

$$RPP(r) = \frac{1}{n} \sum_{i=1}^n I\{r_i \leq r\}$$

and plot this as a function of r . Note that this is the empirical cumulative distribution function of the r_i defined above. See the left panel of Figure 3. Obviously this curve is monotonically increasing in r with $RPP(1) = 1 - \text{misclassification rate}$, if we classify to the most probable class (as we usually do). A good classification procedure has curves lying as high as possible for small values of r . The intuition is similar for ROC (Receiver Operator Characteristic) curves. A perfect predictive model – one assigning the largest estimated probability to the correct cancer class for each observation – produces an inverted, right-angled L-shaped curve

A key observation from Figure 3 is that multinomial classifiers (be it logistic or gradient boosted tree sequences) fit to the reduced set of genetic features suffer comparatively little degradation of classification ability when compared to the same models fit to all potential mutation locations, while enjoying far better performance than a reduced set selected randomly from the set of potential tile centers.

8. Discussion

In summary, we have described a novel method for the extraction of coverage designs with applications for feature prioritization from genomic data. Here, we demonstrated the utility of our approach using large somatic mutation and patient coverage matrices. The method can be applied to a single matrix or to multiple matrices, each from a different class. It was demonstrated that these coverage design methods are reasonably flexible, identifying different structures, depending on the chosen formulation and tuning parameters. Furthermore, the multi class version seems to preserve class information quite well (at least in the application we considered), with multi class prediction models producing prediction results very similar to that of the full dataset.

Benefits of column reduction include: enhanced interpretability, even in the absence of supervised data where no variable selection methods can be used; increased modeling flexibility as more model types can now be fit to the smaller dataset; potentially lower cost and improved measuring accuracy for future prediction or analysis as we have a much smaller set of predictors to consider.

Furthermore, our method is versatile. Apart from feature prioritization, it also flags important structures at user-defined granularities: at exon and gene level, for example. Its ability to identify genes with functional relevance in cancer compares favorably to previous methods dedicated to this task. However, because the procedure is general, its utility is not limited to genes or even mutations. Given these advantages, we envision that our approach will have broad utility for applications that require robust and flexible feature selection from large genomic datasets.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgments

R. Tibshirani is also affiliated with the Department of Health Research and Policy, Redwood Building T152, Stanford University, 150 Governor's Lane, Stanford, CA, USA.

A. Newman, M. Diehn and A. Alizadeh are affiliated with the Institute for Stem Cell Biology and Regenerative Medicine, Stanford University, Stanford, CA, USA, while the latter two are also affiliated with the Stanford Cancer Institute, Stanford University, Stanford, CA, USA.

References

1. Abhishek K, Leyffer S, Linderoth J. Filmint: an outer-approximation-based solver for nonlinear mixed integer programs. *INFORMS Journal on Computing*. 2010; 22:555–567.
2. Bien J, Tibshirani R. Prototype selection for interpretable classification. *Annals of Applied Statistics*. 2011; 5(4):2403–2424.
3. Bonami P, Biegler L, Conn A, Cornujols G, Grossman I, Laird C, Lee J, Lodi F, Margot F, Sawaya N. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete optimization*. 2008; 5:186–204.
4. Burer S, Letchford A. Non-convex mixed-integer nonlinear programming: A survey. *Surveys in Operations Research and Management Science*. 2012; 17:97–106.
5. Deng J, Shoemaker R, Xie B, Gore A, LeProust E, Antosiewicz-Bourget J, Egli D, Maherali N, Park I, Yu J, Daley G, Eggan K, Hochedlinger K, Thomson J, Wang W, Gao Y, Zhang K. Targeted bisulfite sequencing reveals changes in dna methylation associated with nuclear reprogramming. *Nature Biotechnology*. 2009; 27(4):353–360.
6. Duran M, Grossman I. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*. 1986; 36:307–339.
7. Efron B, Hastie T, Johnstone I, Tibshirani R. Least angle regression. *Annals of Statistics*. 2004; 32(2):407–499. With discussion, and a rejoinder by the authors.
8. Geoffrion A. Generalized benders decomposition. *Journal of Optimisation Theory and Applications*. 1972; 10:237–260.
9. Gupta O, Ravindran V. Branch and bound experiments in convex linear programming. *Management Science*. 1985; 31:1533–1546.
10. Lawrence MS, Stojanov P, Polak P, Kryukov GV, Cibulskis K, Sivachenko A, Carter SL, Stewart C, Mermel CH, Roberts SA, Kiezun A, Hammerman PS, McKenna A, Drier Y, Zou L, Ramos AH, Pugh TJ, Stransky N, Helman E, Kim J, Sougnez C, Ambrogio L, Nickerson E, Shefler E, Cortes ML, Auclair D, Saksena G, Voet D, Noble M, DiCara D, Lin P, Lichtenstein L, Heiman DI, Fennell T, Imielinski M, Hernandez B, Hodis E, Baca S, Dulak AM, Lohr J, Landau D-A, Wu CJ, Melendez-Zajgla J, Hidalgo-Miranda A, Koren A, McCarroll SA, Mora J, Lee RS, Crompton B, Onofrio R, Parkin M, Winckler W, Ardlie K, Gabriel SB, Roberts CWM, Biegel JA, Stegmaier K, Bass AJ, Garraway LA, Meyerson M, Golub TR, Gordenin DA, Sunyaev S, Lander ES, Getz G. Mutational heterogeneity in cancer and the search for new cancer-associated genes. *Nature*. 2013; 499(7457):214–218. URL: DOI: 10.1038/nature12213 [PubMed: 23770567]
11. Mercer T, Gerhardt D, Dinger M, Crawford J, Trapnell C, Jeddalo J, Mattick J, Rinn J. Targeted rna sequencing reveals the deep complexity of the human transcriptome. *Nature Biotechnology*. 2011; 30(1):99–104.
12. Newman AM, Bratman SV, To J, Wynne JF, Eclow NCW, Modlin LA, Liu CL, Neal JW, Wakelee HA, Merritt RE, Shrager JB, Loo BW Jr, Alizadeh AA, Diehn M. An ultrasensitive method for quantitating circulating tumor dna with broad patient coverage. *Nature Medicine*. 2014; 20(5): 548–554.
13. Quesada I, Grossman I. An lp/nlp based branch and bound algorithm for convex minlp optimization problems. *Computers and Chemical Engineering*. 1992; 16:937–947.

14. Rehm H. Disease-targeted sequencing: a cornerstone in the clinic. *Nature Reviews Genetics*. 2012; 14(4):295–300.
15. Stubbs R, Mehrota S. A branch-and-cut method for 0–1 mixed convex programming. *Mathematical Programming*. 1999; 86:515–532.
16. Tamborero D, Gonzalez-Perez A, Perez-Llamas C, Deu-Pons J, Kandoth C, Reimand J, Lawrence MS, Getz G, Bader GD, Ding L, Lopez-Bigas N. Comprehensive identification of mutational cancer driver genes across 12 tumor types. *Scientific Reports*. 2013; 3:2650. EP –. URL: doi: 10.1038/srep02650 [PubMed: 24084849]
17. Tibshirani R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*. 1996; 58:267–288.
18. Vandin F, Upfal E, Raphael B. De novo discovery of mutated driver pathways in cancer. *Genome Research*. 2012; 22(2):375–385. [PubMed: 21653252]
19. Welander J, Andreasson A, Juhlin C, Wiseman R, Bäckdahl M, Höög A, Larsson C, Gimm O, Söderkvist P. *The journal of clinical endocrinology and metabolism*. 2014; 99(7):1352–1360.
20. Westerlund T, Pettersson F. A cutting plane method for solving convex minlp problems. *Computers and Chemical Engineering*. 19:131–136. [130].
21. Zou H, Hastie T. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B*. 2005; 67(2):301–320.
22. Zou H, Hastie T, Tibshirani R. Sparse principal component analysis. *Journal of computational and graphical statistics*. 2006; 15(2):265–286.

Figure 1. Schematic for the mutation coverage problem.

Figure 2.

Sample complementary cumulative distribution function (CCDF) of mutations covered per patient over training (**left panel**) and test (**right panel**) sets for each of the three coverage procedures. Procedure of (3) in red, most-frequent mutations procedure in blue and maximum tile coverage in green. Tiles extracted from training set only. Training set CCDFs with circles; test CCDFs with crosses.

Figure 3.

Rank proportion plots (RPP) for some multi class prediction models based on the distillation methods. Red curve represents test set RPP on full data; green, blue and purple the sets for different reducing methods; brown the most-frequent mutations; yellow the maximal tile coverage and orange a random reducing vector. Solid lines are for multinomial logistic regression models; dashed ones for the gradient boosted multinomial model.

Table 1

Median number of mutations covered per patient for three coverage procedures over training and test sets. Methods numbered in top row are i) Equation (3) with $\gamma = 1$; ii) Mutation count; iii) Naive tile coverage; iv) Greedy tile coverage

Method	i)	ii)	iii)	iv)
Training	13	7	5	8
Test	7	4	3	4

Table 2

Median mutations per patient covered in the **test** set for five selectors: our overall selector, using the **overall** method, naive mutation count procedures using both overall and per cancer data, each with $R_0 = 3000$ tiles and their greedy, non-naive versions. Table elements show the median (linearly interpolated between neighbouring integers) for the 8 cancers with the most patients and the overall dataset.

Method	BRCA	LUAD	HNSC	LGG	STAD	SKCM	overall
Master	2	12	7	4	9	14	5
Naive (over.)	1	4	3	3	3	4	2
Naive (p. c.)	2	5	4	4	3	8	3
Greedy (over.)	1	4	3	3	3	4	2
Greedy (p. c.)	2	5	4	4	3	8	3

Table 3

Results of exon and gene ranking exercise for the pan cancer master tile set and competing methods. **First four columns:** A ranked list of exons and their coordinates (chromosome, start and end base pairs along that chromosome) and overlapping genes. List ranked by count of unique patients with mutations covered along exon normalised by tile length overlapping with exon. **Fifth and sixth columns:** List, in lexicographical order of G = 20 genes selected by the greedy and Monte Carlo Markov Chain (MCMC) methods of Vandin et al. [18]. Bolded genes are HCD genes of Tamborero et al. [16].

Chromo.	Start BP	End BP	Gene	Greedy	MCMC
chr7	140453074	140453193	BRAF	<i>AC007563.5</i>	<i>AC007879.2</i>
chr2	209113092	209113384	IDH1	<i>AK300121</i>	<i>AGAP5</i>
chr17	7578370	7578554	TP53	<i>APOBEC3A</i>	<i>ANXA2P2</i>
chr12	25398207	25398318	KRAS	<i>ARHGEF39</i>	<i>BCL2</i>
chr17	7577018	7577155	TP53	<i>BC080605</i>	BRAF
chr20	29625872	29625984	FRG1B	<i>BCL2</i>	<i>CTD-2001C12.1</i>
chr3	178935997	178936122	PIK3CA	BRAF	<i>H2AFZ</i>
chr17	7577498	7577608	TP53	CEBPA	<i>HMCN2</i>
chr1	115256420	115256599	NRAS	<i>FAM182A</i>	<i>IGHV2-70</i>
chr3	178951881	178952152	PIK3CA	HRAS	NRAS
chr10	89692769	89693008	PTEN	<i>IGHV2-70</i>	PIK3CA
chr17	7578176	7578289	TP53	<i>LAMTOR4</i>	<i>PPP1R3G</i>
chr9	21970900	21971207	CDKN2A	<i>LOC650368</i>	<i>PRTN3</i>
chr17	7579311	7579569	TP53	<i>MYD88</i>	<i>RPI1-138A23.2</i>
chr16	90161541	90162011	<i>BC1397191</i>	NRAS	<i>RPI1-145E5.5</i>
chr3	41266016	41266264	<i>CTNNB1</i> ²	PIK3CA	<i>RPI1-4104.2</i>
chr1	6257711	6257816	RPL22	<i>PPDPF</i>	<i>RPI1-7F17.5</i>
chr17	7576852	7576926	TP53	TP53	<i>SRSF10</i>
chr17	56434828	56436184	RNF43	<i>UQCRCF51</i>	TP53
chr5	67590975	67591152	PIK3RI	VHL	VHL

¹ Also TUBB4Q.
² Also AK311005.