# Supplemental Material

## Varying levels of complexity in transcription factor binding motifs

Jens Keilwagen and Jan Grau

## Text S1 STEP-BY-STEP EXAMPLE OF DEPENDENCY LOGO GENERATION

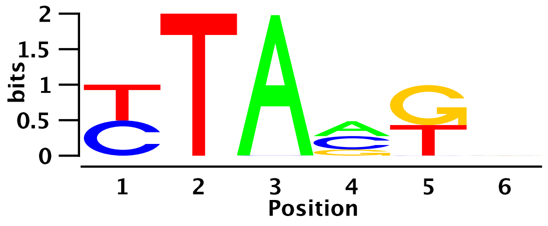
In the following we explain by means of a toy example how dependency logos are generated from a set of binding sites. We assume a set of 500 aligned binding sites corresponding to a motif of length 6. These binding sites are represented by the subset of sequences displayed below. We instantly see that the second and third position are completely conserved with a T and an A, respectively. In addition, we might observe that position 1 is either a C or a T, and that position 5 is either G or T.

| | |
|---|---|
| TTAATG | 8 |
| CTAGGC | 4 |
| TTAGGA | 4 |
| TTACGA | 8 |
| TTAGGT | 2 |
| CTACGC | 5 |
| CTAATG | 10 |
| TTACGC | 5 |
| CTAATA | 10 |
| CTAATC | 9 |
| TTAATA | 7 |
| CTAATT | 11 |
| CTACGT | 5 |
| TTACGG | 6 |
| CTAGGT | 3 |
| CTACGA | 7 |
| TTAGGC | 3 |
| CTAGGG | 2 |
| CTAGGA | 6 |
| TTAATC | 9 |
| TTAATT | 12 |
| TTAGGG | 4 |
| TTACGT | 6 |
| CTACGG | 6 |

...

Each of these sequence also has an associated value, e.g., probe intensities in a gcPBM experiment, peak statistics for ChIP-seq data, or scores according to a motif model, illustrated by the number on the right of each sequence.

First, we might want to plot a traditional sequence logo of this set of binding sites. The corresponding sequence logo looks like this:



We find our observations regarding position 1, 2, 3, and 5 confirmed, whereas positions 4 and 6 are substantially less conserved.

To get a better impression of the dependency structure of these binding sites, we generate a dependency logo of these sequences in the following.

We start with the computation of the dependencies between binding site positions using mutual information (termed $M_{i,j}$ in Methods) and use these to determine that position with the strongest dependencies to other positions (measured by $D(i)$, see Methods).

For this toy data set, the position with the strongest dependencies to other positions is position $5$, which shows the strongest dependency (i.e. $M_{5,j}$) to position $j = 4$ (and vice versa).

Hence, we group all binding sites in the toy data set according to the nucleotides at positions 5 and 4. For this specific data set, this yields only 3 non-empty groups: one for G at position 5 and C at position 4, one for T at position 5 and A at position 4, and one for G at position 5 and G at position 4:

| Group 1 | | Group 2 | | Group 3 | |
|---|---|---|---|---|---|
| G at position 5, C at position 4: | | T at position 5, A at position 4: | | G at position 5, G at position 4: | |
| TTACGC | 5 | CTAATC | 9 | CTAGGA | 6 |
| TTACGG | 6 | TTAATG | 8 | TTAGGG | 4 |
| CTACGG | 6 | TTAATT | 12 | TTAGGA | 4 |
| CTACGA | 7 | TTAATA | 7 | CTAGGG | 2 |
| TTACGT | 6 | TTAATC | 9 | TTAGGC | 3 |
| CTACGC | 5 | CTAATA | 10 | CTAGGT | 3 |
| TTACGA | 8 | CTAATG | 10 | CTAGGC | 4 |
| CTACGT | 5 | CTAATT | 11 | TTAGGT | 2 |
| ... | | ... | | ... | |

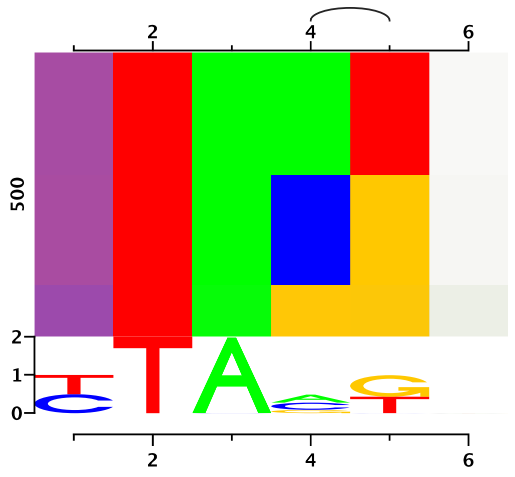
We do not find further strong dependencies between any two positions within these sub-groups. Hence, these groups form our final partitioning for this dependency logo. Otherwise, we could have repeated this procedure recursively within each (or a subset) of these groups.

In the plot, each of these groups will be displayed as one row that represents a set of binding sites with common nucleotides at a subset of binding site positions, which we have selected based on the strength of their dependencies to other positions.

Next, we need to determine the ordering of these groups in the dependency logo plot. We order the groups by the average of the associated values of the binding sites within the groups. In the toy example, the sequences of group 2 have the largest associated values, followed by group 1 and, finally, group 3. Hence, we obtain the final ordering as

| | |
|---|---|
| CTAATC | 9 |
| TTAATG | 8 |
| TTAATT | 12 |
| TTAATA | 7 |
| TTAATC | 9 |
| CTAATA | 10 |
| CTAATG | 10 |
| CTAATT | 11 |
| ... | |
| TTACGC | 5 |
| TTACGG | 6 |
| CTACGG | 6 |
| CTACGA | 7 |
| TTACGT | 6 |
| CTACGC | 5 |
| TTACGA | 8 |
| CTACGT | 5 |
| ... | |
| CTAGGA | 6 |
| TTAGGG | 4 |
| TTAGGA | 4 |
| CTAGGG | 2 |
| TTAGGC | 3 |
| CTAGGT | 3 |
| CTAGGC | 4 |
| TTAGGT | 2 |
| ... | |

In the following dependency logo, we find exactly these groups as rows of colored boxes.



The first row represents the previous group 2 (T at 5 and A at 4) as can be perceived from the red and green boxes at these positions. In complete analogy, we discover group 1 as orange and blue boxes and group 3 as two orange boxes at positions 5 and 4, respectively.

Since positions 2 and 3 are conserved across all binding sites, these are shown as red and green boxes, respectively, in the rows of all three groups. Notably, position 1, which showed similar conservation as position 5 in the sequence logo, appears to be largely independent of the groups, i.e., position 1 is T or C with similar probability in each of the groups. Accordingly, the boxes at position 1 are colored in violet (mixture of blue and red representing C and T) in all three groups. In contrast, we now can visually perceive the dependency between positions 4 and 5, where, for instance, position 4 is always A if position 5 is T. We additionally highlight the dependency between positions 4 and 5 by an arc between those two positions in the upper part of the dependency logo.

Finally, all four nucleotides appear with similar probability and independent of the groups at position 6. Hence, this position has only a low saturation in all three groups, in analogy to the nucleotide stack of height almost 0 in the sequence logo.

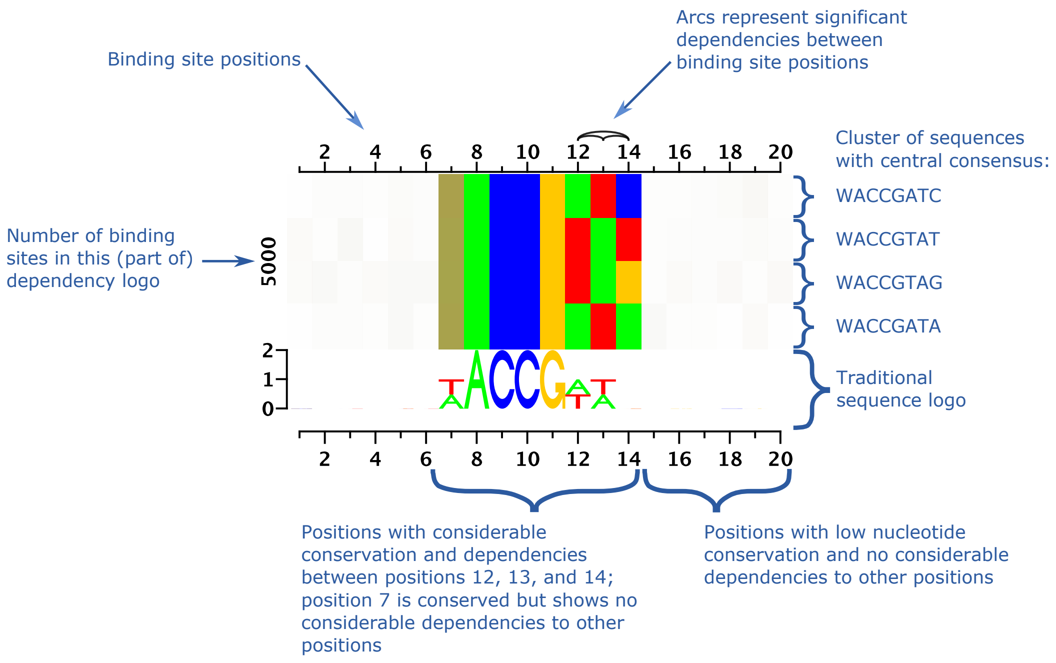


**Figure S1.** Annotated dependency logo explaining the different properties of dependency logos and supporting their interpretation.

## Text S2 DEPENDENCY LOGOS

In Figure S2, we illustrate five examples, where the sequence logo is identical but the actual motif models differ. The figure demonstrates that dependency logos are capable of visualizing no dependencies, neighboring and non-neighboring dependencies, and differing probabilities.

Nevertheless, it might be hard to distinguish between dependencies and heterogeneities. One illustrative example is given in Figure S3. If the number of positions that depend on each other is large and the number of highly conserved positions is low, one might think of heterogeneities. Figure S3D depicts 5 positions that depend on each other and 3 that are highly conserved. Alternatively, we can model this motif using a mixture of 3 highly conserved components TACCGATC, TACCGCGC, and TACGAGAT. However, the transition between perceived dependencies to perceived heterogeneities is smooth leading to cases where it is hard to decide whether the dependency logo shows heterogeneities or dependencies.

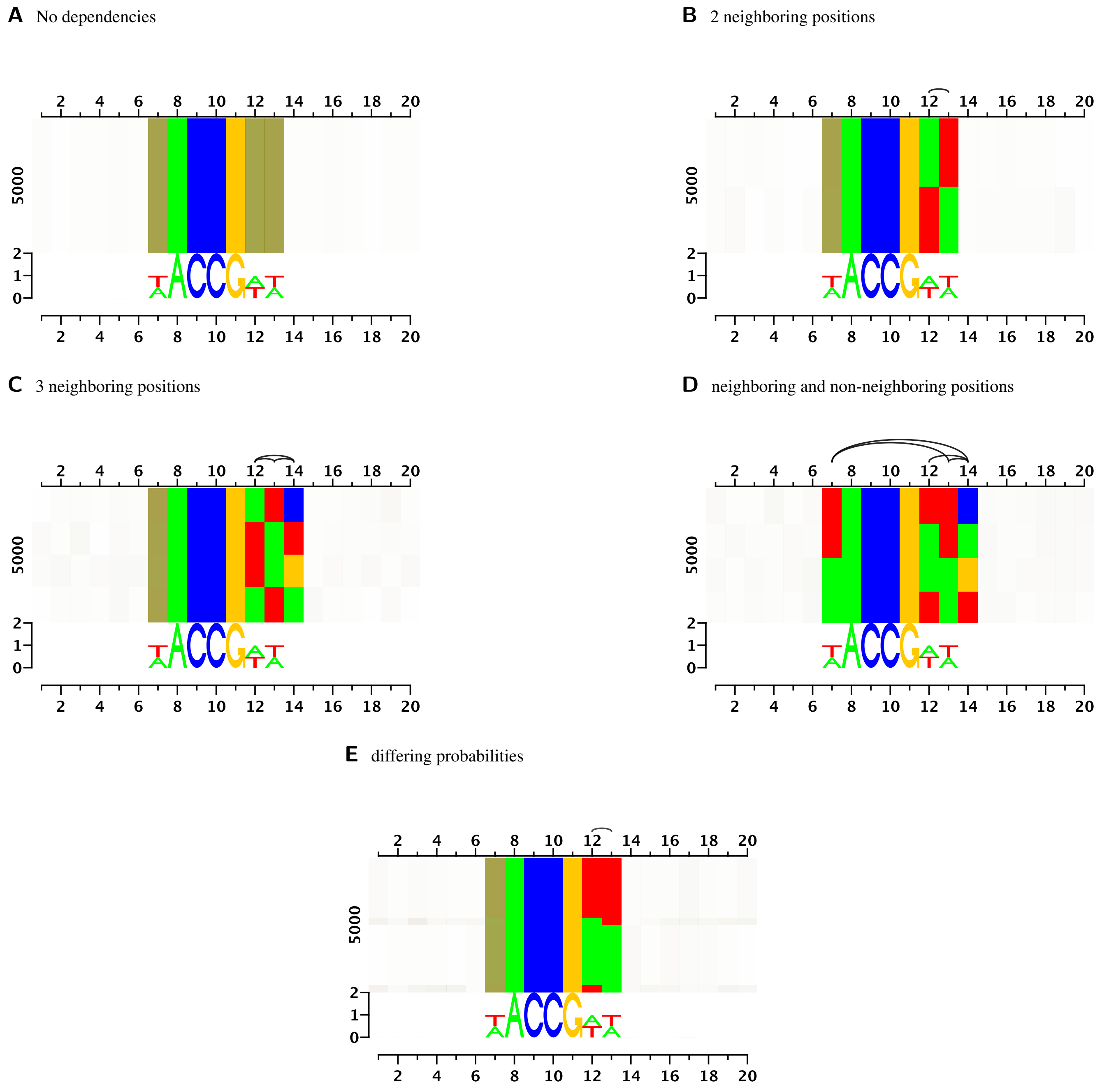


**Figure S2.** Dependency logos provide insights into the structure of dependencies.

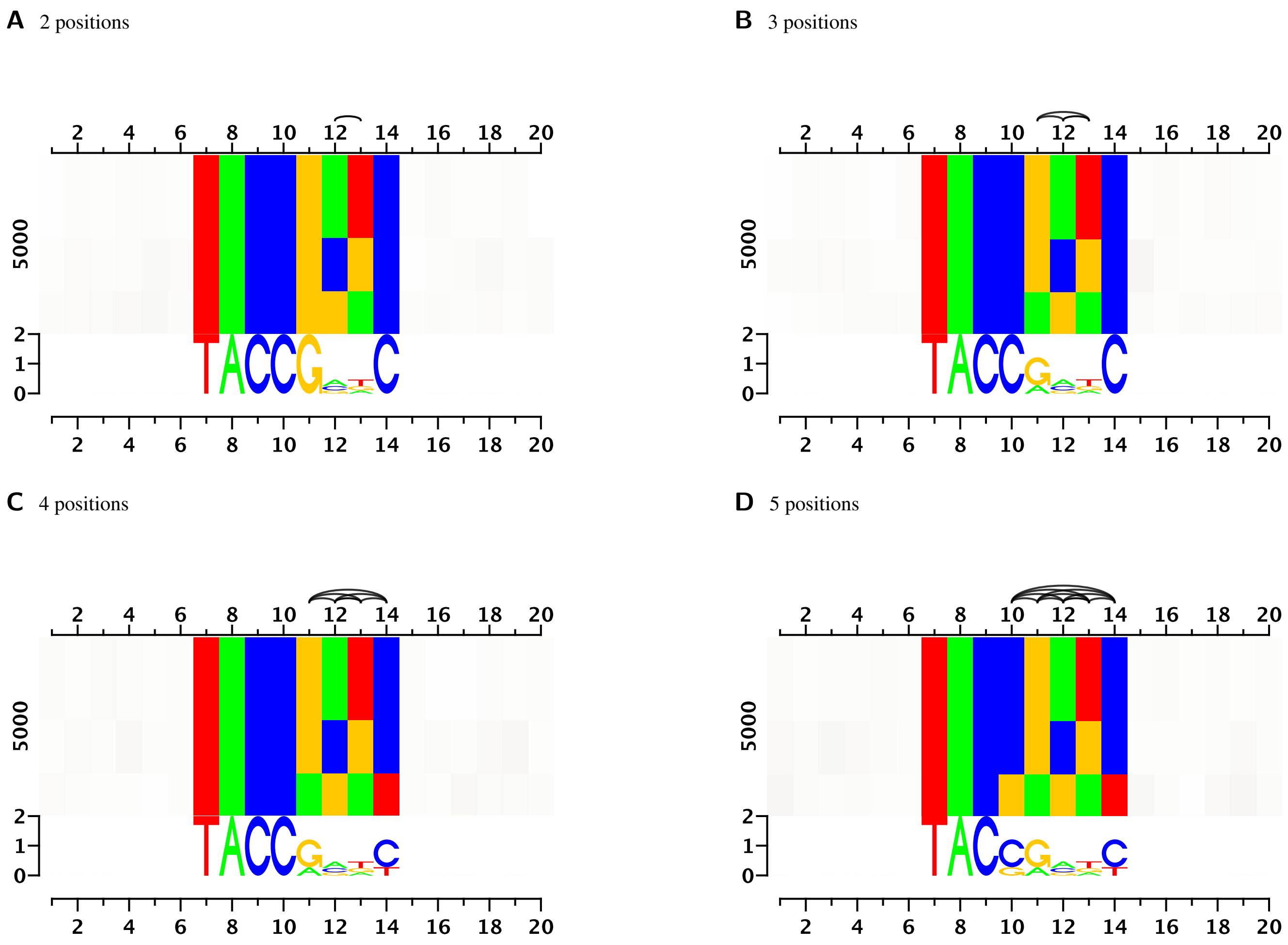


**Figure S3.** Smooth transition from perceived dependencies to perceived heterogeneities.

## Text S3 PERFORMANCE ASSESSMENT

### Text S3.1 gcPBM data

We assess the performance of classifiers using the different foreground models by means of the squared Pearson correlation ($R^2$) between prediction scores and PBM intensities as proposed by (12). We then compute arithmetic mean and standard error of the $R^2$ values over all 10 cross validation iterations for each transcription factor. Since we use the identical partitioning, mean values are directly comparable to (12).

### Text S3.2 ENCODE ChIP-seq data

We assess the prediction performance of different approaches using different motif models on the ENCODE data sets for different cell types. We train each of the approaches on the ChIP-seq data set measured for the cell type with the largest number of peaks and test its prediction accuracy on the ChIP-seq data sets for the remaining cell types. This scenario resembles likely practical applications, e.g., combining computational predictions with cell type-specific DNase I hypersensitivity experiments. We test each of the approaches on the ChIP-seq data sets for the remaining cell type(s) using the identical sequences of length 1000 bp around the peak center for all approaches considered to obtain comparable results, but obtain highly similar results using sequences of length 100 bp (data not shown). For the assessment, we need a single prediction score for each input sequence. We use the ZOOPS score (cf. (15)), but we also test the maximum score and obtain similar results (data not shown).

We additionally test Dimont using the different motif models in a 10-fold cross validation experiment using only the ChIP-seq data set with the largest number of peaks. This scenario allows for a more rigorous assessment of prediction performance and avoids an over-estimation of prediction performance due to overfitting effects with increasing model complexity, since ChIP-seq experiments for different cell types but the same transcription factors may produce substantially overlapping peak regions (Table S1).

The performance measures considered for the ENCODE data are

**AUC-ROC and AUC-PR** We evaluate all approaches for the classification problem of distinguishing the top 500 ChIP-seq regions of length 1000 bp from 5000 negative regions of the same length, which are sampled uniformly from the human genome (hg19). As performance measure, we use the area under the ROC curve (AUC-ROC) and the area under the precision-recall curve (AUC-PR) (72).

**AUC-ROC and AUC-PR (shuffled)** We evaluate all approaches for the classification problem of distinguishing the top 500 ChIP-seq regions of length 1000 bp from di-nucleotide shuffled versions of the same sequences. As performance measure, we use the area under the ROC curve (AUC-ROC) and the area under the precision-recall curve (AUC-PR) (72).

**wAUC-ROC and wAUC-PR** We evaluate all approaches for the weighted classification problem distinguishing highly occupied from less occupied peak regions using all ChIP-seq regions in the corresponding test set, where each region is assigned a weight (cf. section "Learning model parameters") based on the corresponding ChIP-seq peak statistics. As performance measures we use weighted AUC-ROC and weighted AUC-PR (73).

**Pearson and Spearman correlation** We evaluate all approaches for the regression problem of reconstructing the ChIP-seq peak statistics by predictions scores using Pearson correlation and Spearman correlation between prediction scores and peak statistics.

## Text S4 PROOF OF CONCEPT - SLIM ON ARTIFICIAL DATA

As a proof of concept, we evaluate the performance of a sparse local inhomogeneous mixture model using artificial data. Assessing the feature selection ability of Slim models, we compare the performance of binary classifiers comprising two inhomogeneous Markov models, Bayesian trees and sparse local inhomogeneous mixture models, respectively, in a 100-fold simulation. In each iteration, we generate training and test data from known statistical models. As generating models, we use two maximum entropy models (22) that share some dependencies. In more detail, we use a maximum entropy model with constraints m2sx for the foreground class. For the background class, we also use a maximum entropy model but sample constraints with probability of 50% from the constraints m2sx. Given two maximum entropy models, we sample training and test data sets of same size and class ratio of 1:1. Evaluating the influence of the size of the training data set on classifier performance, we subsample the training data set obtaining three training data sets with size 100, 1,000, and 10,000 sequences. Subsequently, we train each classifier on one training data set and assess each classifier in terms of AUC-ROC using the test data set.

We summarize the results of the 100-fold simulation by mean and standard error of AUC-ROC as visualized in Figure S4. We find that the performance increased with increasing size of the data sets for all classifiers indicating that the size of the training data set has a decisive influence on the estimation of model parameters and the identification of relevant features.

Comparing the classifiers among each other, we observe that the performance is increasing with increasing order for classifiers comprising two inhomogeneous Markov models. Since the data of the foreground and background class only differ in some dependencies, the classifier based on two inhomogeneous Markov models of order 0 is not able to separate the classes. In contrast the classifier based on two inhomogeneous Markov models of order 1 can at least capture dependencies between neighboring

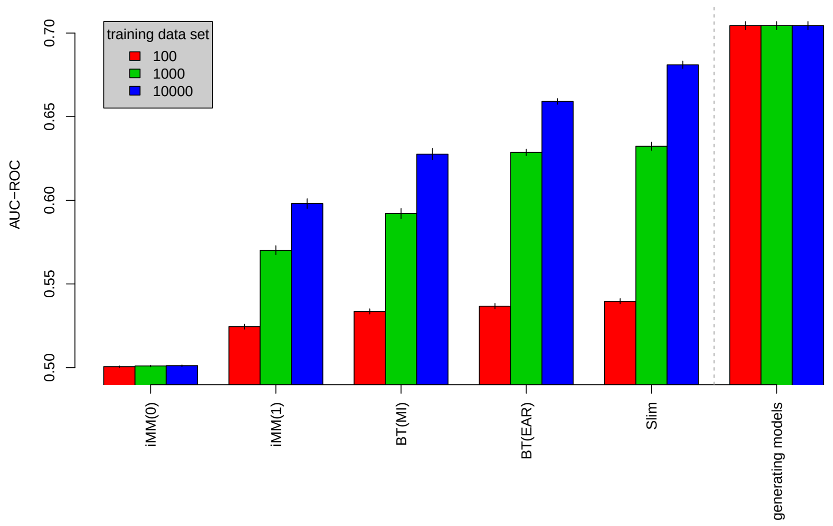


**Figure S4.** Mean AUC-ROC for discriminatively trained binary classifiers in a 100-fold simulation. Each classifier utilizes the same model type for foreground and background class varying from inhomogeneous Markov models (iMM) of order 0 and 1, Bayesian trees (BT) to sparse local inhomogeneous mixture (Slim) models. We also include the performance of a classifier comprising the generating models of the data sets as a reference.

positions and, hence, can moderately separate the classes. However, we find that this classifier can be outperformed by Bayesian trees.

Considering classifiers comprising two Bayesian trees, we find that the feature selection criteria influences the performance as expected. We find that using explaining away residue (EAR) yields slightly better performance compared to mutual information (MI), which might be explained by the discriminative character of this feature selection criterion (54, 55).

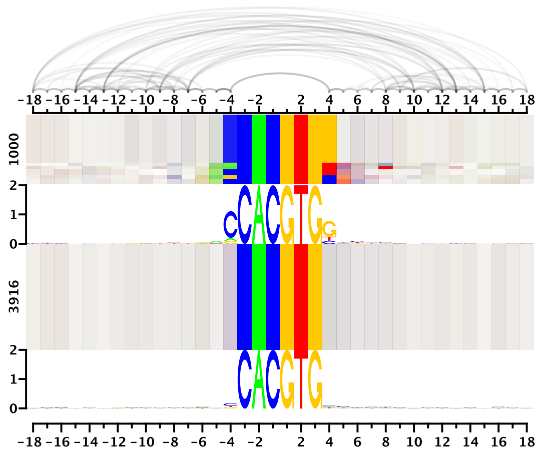
Finally, comparing the results of the classifier using Bayesian trees and EAR with the Slim model, we find that the Slim model performs equally for small and medium sized training data sets, while it outperforms the Bayesian tree on the largest data set.

Investigating the significance of the performance differences for different models, we utilize the standard error measured in the simulation study. For the largest data set, we find that the difference between iMM(0) and iMM(1), iMM(1) and BT(MI), BT(MI) and BT(EAR), and BT(EAR) and Slim are larger than twice the standard error. Hence, the observed difference between the classifiers based on these models are significant. For this reason, we conclude that feature selection significantly improves the performance and that Slim models are able to compete with Bayesian trees.

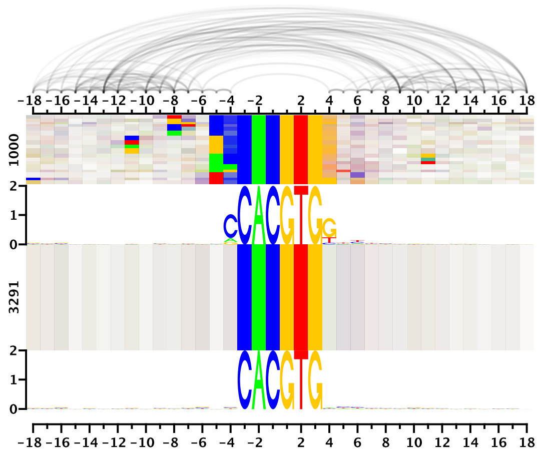
In addition, if different subclasses of the sequences exist within one data set or the sequences are not aligned, feature selection before numerical parameter estimation is likely to fail. Hence, the utilization of Bayesian trees in mixture models and for de-novo motif discovery using discriminative learning principles is limited. In contrast, the Slim model is able to adjust the feature weights during numerical parameter estimation, allowing for feature selection in mixture models and for de-novo motif discovery. We investigate the behavior of the Slim model for real data sets in the main text.

## Text S5 DEPENDENCY LOGOS FOR MYC, MAD AND MAX USING DATA OF MORDELET et al.

**A** Myc



**B** Mad



**C** Max

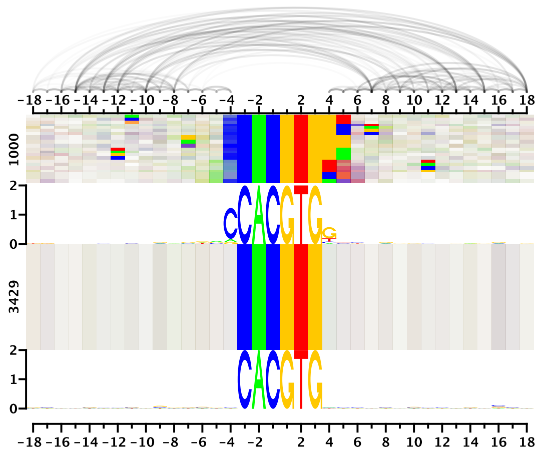


**Figure S5.** Dependency logo for the Myc, Mad and Max data sets. We plot dependency logos for the top 1000 sites according to the prediction scores of the LSlim model and for the remaining sites.

## Text S6 DE-NOVO MOTIF DISCOVERY APPROACH

We compare the prediction accuracy of the Dimont framework using baseline models, namely PWMs and WAMs, against several other state of the art approaches, namely MEME (37, 56), DiChIPMunk (14), and TFFMs (13). MEME is widely used for ChIP-seq data (74, 75, 76, 77, 78) and is the standard of factorbook (79). Hence, it can still be seen as *de facto* standard for motif discovery. DiChIPMunk extends ChIPMunk (80), which is one of the best approaches for motif discovery from ChIP data (81), to a dinucleotide model. Transcription factor flexible models (TFFMs) use a hidden Markov model approach to represent variable-length motifs including dinucleotide dependencies, and will be included in a future version of JASPAR (13, 82) as an alternative to PWMs.

### Text S6.1 List of ChIP-seq data sets

The following list contains the file names of all ChIP-seq data sets for human transcription factors from ENCODE that we used in the ChIP-seq analysis section. All files can be downloaded from `http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeAwgTfbsUniform/` and refer to the human genome (hg19).

wgEncodeAwgTfbsBroadGm12878CtcfUniPk.narrowPeak
wgEncodeAwgTfbsBroadGm12878Ezh239875UniPk.narrowPeak
wgEncodeAwgTfbsBroadH1hescCtcfUniPk.narrowPeak
wgEncodeAwgTfbsBroadH1hescEzh239875UniPk.narrowPeak
wgEncodeAwgTfbsBroadH1hescRbbp5a300109aUniPk.narrowPeak
wgEncodeAwgTfbsBroadK562Chd1a301218aUniPk.narrowPeak
wgEncodeAwgTfbsBroadK562CtcfUniPk.narrowPeak
wgEncodeAwgTfbsBroadK562Ezh239875UniPk.narrowPeak
wgEncodeAwgTfbsBroadK562Hdac2a300705aUniPk.narrowPeak
wgEncodeAwgTfbsBroadK562P300UniPk.narrowPeak
wgEncodeAwgTfbsBroadK562Rbbp5a300109aUniPk.narrowPeak
wgEncodeAwgTfbsHaibGm12878Atf2sc81188V0422111UniPk.narrowPeak
wgEncodeAwgTfbsHaibGm12878Atf3Pcr1xUniPk.narrowPeak
wgEncodeAwgTfbsHaibGm12878Bcl11aPcr1xUniPk.narrowPeak
wgEncodeAwgTfbsHaibGm12878Bcl3V0416101UniPk.narrowPeak
wgEncodeAwgTfbsHaibGm12878Bclaf101388V0416101UniPk.narrowPeak
wgEncodeAwgTfbsHaibGm12878Cebpbsc150V0422111UniPk.narrowPeak
wgEncodeAwgTfbsHaibGm12878Egr1Pcr2xUniPk.narrowPeak
wgEncodeAwgTfbsHaibGm12878Elf1sc631V0416101UniPk.narrowPeak
wgEncodeAwgTfbsHaibGm12878Ets1Pcr1xUniPk.narrowPeak
wgEncodeAwgTfbsHaibGm12878GabpPcr2xUniPk.narrowPeak
wgEncodeAwgTfbsHaibGm12878Mef2aPcr1xUniPk.narrowPeak
wgEncodeAwgTfbsHaibGm12878NrsfPcr1xUniPk.narrowPeak
wgEncodeAwgTfbsHaibGm12878P300Pcr1xUniPk.narrowPeak
wgEncodeAwgTfbsHaibGm12878Pmlsc71910V0422111UniPk.narrowPeak
wgEncodeAwgTfbsHaibGm12878Pu1Pcr1xUniPk.narrowPeak
wgEncodeAwgTfbsHaibGm12878Rad21V0416101UniPk.narrowPeak
wgEncodeAwgTfbsHaibGm12878RxraPcr1xUniPk.narrowPeak
wgEncodeAwgTfbsHaibGm12878Six5Pcr1xUniPk.narrowPeak
wgEncodeAwgTfbsHaibGm12878Sp1Pcr1xUniPk.narrowPeak
wgEncodeAwgTfbsHaibGm12878SrfPcr2xUniPk.narrowPeak
wgEncodeAwgTfbsHaibGm12878Stat5asc74442V0422111UniPk.narrowPeak
wgEncodeAwgTfbsHaibGm12878Taf1Pcr1xUniPk.narrowPeak
wgEncodeAwgTfbsHaibGm12878Tcf12Pcr1xUniPk.narrowPeak
wgEncodeAwgTfbsHaibGm12878Usf1Pcr2xUniPk.narrowPeak
wgEncodeAwgTfbsHaibGm12878Yy1sc281Pcr1xUniPk.narrowPeak
wgEncodeAwgTfbsHaibGm12878Zbtb33Pcr1xUniPk.narrowPeak
wgEncodeAwgTfbsHaibH1hescAtf2sc81188V0422111UniPk.narrowPeak
wgEncodeAwgTfbsHaibH1hescAtf3V0416102UniPk.narrowPeak
wgEncodeAwgTfbsHaibH1hescBcl11aPcr1xUniPk.narrowPeak
wgEncodeAwgTfbsHaibH1hescEgr1V0416102UniPk.narrowPeak
wgEncodeAwgTfbsHaibH1hescFosl1sc183V0416102UniPk.narrowPeak
wgEncodeAwgTfbsHaibH1hescGabpPcr1xUniPk.narrowPeak
wgEncodeAwgTfbsHaibH1hescHdac2sc6296V0416102UniPk.narrowPeak

wgEncodeAwgTfbsHaibH1hescJundV0416102UniPk.narrowPeak
wgEncodeAwgTfbsHaibH1hescNrsfV0416102UniPk.narrowPeak
wgEncodeAwgTfbsHaibH1hescP300V0416102UniPk.narrowPeak
wgEncodeAwgTfbsHaibH1hescRad21V0416102UniPk.narrowPeak
wgEncodeAwgTfbsHaibH1hescRxraV0416102UniPk.narrowPeak
wgEncodeAwgTfbsHaibH1hescSin3ak20Pcr1xUniPk.narrowPeak
wgEncodeAwgTfbsHaibH1hescSix5Pcr1xUniPk.narrowPeak
wgEncodeAwgTfbsHaibH1hescSp1Pcr1xUniPk.narrowPeak
wgEncodeAwgTfbsHaibH1hescSp2V0422111UniPk.narrowPeak
wgEncodeAwgTfbsHaibH1hescSrfPcr1xUniPk.narrowPeak
wgEncodeAwgTfbsHaibH1hescTaf1V0416102UniPk.narrowPeak
wgEncodeAwgTfbsHaibH1hescTaf7sc101167V0416102UniPk.narrowPeak
wgEncodeAwgTfbsHaibH1hescTcf12Pcr1xUniPk.narrowPeak
wgEncodeAwgTfbsHaibH1hescTead4sc101184V0422111UniPk.narrowPeak
wgEncodeAwgTfbsHaibH1hescUsf1Pcr1xUniPk.narrowPeak
wgEncodeAwgTfbsHaibH1hescYy1sc281V0416102UniPk.narrowPeak
wgEncodeAwgTfbsHaibK562Atf3V0416101UniPk.narrowPeak
wgEncodeAwgTfbsHaibK562Bcl3Pcr1xUniPk.narrowPeak
wgEncodeAwgTfbsHaibK562Bclaf101388Pcr1xUniPk.narrowPeak
wgEncodeAwgTfbsHaibK562Cebpbsc150V0422111UniPk.narrowPeak
wgEncodeAwgTfbsHaibK562Egr1V0416101UniPk.narrowPeak
wgEncodeAwgTfbsHaibK562Elf1sc631V0416102UniPk.narrowPeak
wgEncodeAwgTfbsHaibK562Ets1V0416101UniPk.narrowPeak
wgEncodeAwgTfbsHaibK562Fosl1sc183V0416101UniPk.narrowPeak
wgEncodeAwgTfbsHaibK562GabpV0416101UniPk.narrowPeak
wgEncodeAwgTfbsHaibK562MaxV0416102UniPk.narrowPeak
wgEncodeAwgTfbsHaibK562Mef2aV0416101UniPk.narrowPeak
wgEncodeAwgTfbsHaibK562NrsfV0416102UniPk.narrowPeak
wgEncodeAwgTfbsHaibK562Pmlsc71910V0422111UniPk.narrowPeak
wgEncodeAwgTfbsHaibK562Pu1Pcr1xUniPk.narrowPeak
wgEncodeAwgTfbsHaibK562Sin3ak20V0416101UniPk.narrowPeak
wgEncodeAwgTfbsHaibK562Six5Pcr1xUniPk.narrowPeak
wgEncodeAwgTfbsHaibK562Sp1Pcr1xUniPk.narrowPeak
wgEncodeAwgTfbsHaibK562Sp2sc643V0416102UniPk.narrowPeak
wgEncodeAwgTfbsHaibK562SrfV0416101UniPk.narrowPeak
wgEncodeAwgTfbsHaibK562Stat5asc74442V0422111UniPk.narrowPeak
wgEncodeAwgTfbsHaibK562Taf1V0416101UniPk.narrowPeak
wgEncodeAwgTfbsHaibK562Taf7sc101167V0416101UniPk.narrowPeak
wgEncodeAwgTfbsHaibK562Tead4sc101184V0422111UniPk.narrowPeak
wgEncodeAwgTfbsHaibK562Usf1V0416101UniPk.narrowPeak
wgEncodeAwgTfbsHaibK562Yy1V0416101UniPk.narrowPeak
wgEncodeAwgTfbsHaibK562Zbtb33Pcr1xUniPk.narrowPeak
wgEncodeAwgTfbsSydhGm12878Bhlhe40cIggmusUniPk.narrowPeak
wgEncodeAwgTfbsSydhGm12878Brca1a300IggmusUniPk.narrowPeak
wgEncodeAwgTfbsSydhGm12878CfosUniPk.narrowPeak
wgEncodeAwgTfbsSydhGm12878Chd1a301218aIggmusUniPk.narrowPeak
wgEncodeAwgTfbsSydhGm12878Chd2ab68301IggmusUniPk.narrowPeak
wgEncodeAwgTfbsSydhGm12878Corestsc30189IggmusUniPk.narrowPeak
wgEncodeAwgTfbsSydhGm12878E2f4IggmusUniPk.narrowPeak
wgEncodeAwgTfbsSydhGm12878Elk112771IggmusUniPk.narrowPeak
wgEncodeAwgTfbsSydhGm12878JundUniPk.narrowPeak
wgEncodeAwgTfbsSydhGm12878MaxIggmusUniPk.narrowPeak
wgEncodeAwgTfbsSydhGm12878Mazab85725IggmusUniPk.narrowPeak
wgEncodeAwgTfbsSydhGm12878Mxi1IggmusUniPk.narrowPeak
wgEncodeAwgTfbsSydhGm12878Nfe2sc22827UniPk.narrowPeak
wgEncodeAwgTfbsSydhGm12878NfyaIggmusUniPk.narrowPeak
wgEncodeAwgTfbsSydhGm12878NfybIggmusUniPk.narrowPeak
wgEncodeAwgTfbsSydhGm12878Nrf1IggmusUniPk.narrowPeak
wgEncodeAwgTfbsSydhGm12878Rfx5200401194IggmusUniPk.narrowPeak
wgEncodeAwgTfbsSydhGm12878Sin3anb6001263IggmusUniPk.narrowPeak
wgEncodeAwgTfbsSydhGm12878Smc3ab9263IggmusUniPk.narrowPeak

wgEncodeAwgTfbsSydhGm12878Stat1UniPk.narrowPeak
wgEncodeAwgTfbsSydhGm12878Tblr1ab24550IggmusUniPk.narrowPeak
wgEncodeAwgTfbsSydhGm12878TbpIggmusUniPk.narrowPeak
wgEncodeAwgTfbsSydhGm12878Tr4UniPk.narrowPeak
wgEncodeAwgTfbsSydhGm12878Usf2IggmusUniPk.narrowPeak
wgEncodeAwgTfbsSydhGm12878Znf143166181apUniPk.narrowPeak
wgEncodeAwgTfbsSydhH1hescBach1sc14700IggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhH1hescBrca1IggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhH1hescCebpbIggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhH1hescChd1a301218aIggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhH1hescChd2IggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhH1hescCjunIggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhH1hescCmycIggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhH1hescGtf2f1IggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhH1hescMaxUcdUniPk.narrowPeak
wgEncodeAwgTfbsSydhH1hescMxi1IggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhH1hescNrf1IggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhH1hescRfx5200401194IggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhH1hescTbpIggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhH1hescUsf2IggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhH1hescZnf143IggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhK562Bach1sc14700IggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhK562Bhlhe40nb100IggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhK562CfosUniPk.narrowPeak
wgEncodeAwgTfbsSydhK562Chd2ab68301IggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhK562CjunIfna30UniPk.narrowPeak
wgEncodeAwgTfbsSydhK562CmycIggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhK562Corestab24166IggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhK562E2f4UcdUniPk.narrowPeak
wgEncodeAwgTfbsSydhK562Elk112771IggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhK562Gtf2f1ab28179IggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhK562JundIggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhK562Mazab85725IggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhK562Mxi1af4185IggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhK562Nfe2UniPk.narrowPeak
wgEncodeAwgTfbsSydhK562NfyaUniPk.narrowPeak
wgEncodeAwgTfbsSydhK562NfybUniPk.narrowPeak
wgEncodeAwgTfbsSydhK562Nrf1IggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhK562Rfx5IggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhK562Smc3ab9263IggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhK562Stat1Ifng30UniPk.narrowPeak
wgEncodeAwgTfbsSydhK562Tblr1ab24550IggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhK562TbpIggmusUniPk.narrowPeak
wgEncodeAwgTfbsSydhK562Tr4UcdUniPk.narrowPeak
wgEncodeAwgTfbsSydhK562Usf2IggrabUniPk.narrowPeak
wgEncodeAwgTfbsSydhK562Znf143IggrabUniPk.narrowPeak
wgEncodeAwgTfbsUtaGm12878CmycUniPk.narrowPeak

## Text S6.2 Overlap between ChIP-seq data sets

For the evaluation of different de-novo motif discovery tools, we train these tools on ChIP-seq data from one cell type and assess their performance on ChIP-seq data from another cell type but the same transcription factor. Although the data sets are based on different cell types, there might be an overlap of the ChIP-seq regions. However, it is hard to quantify this overlap as ChIP-seq regions of different cell types or experiments might overlap only to a certain degree, i.e. certain number of base pairs.

In Table S1, we report overlap if two ChIP-seq regions (peak start to peak end as given in narrowPeak files) share at least 1 bp. This is very stringent, but avoids missing any overlapping ChIP-seq regions. The percentage of ChIP-seq regions in the test data set that are also in the training data set varies between $2.83\%$ and $95.1\%$ with a mean of approximately $50\%$.

**Table S1.** Statistics for each ENCODE data set used. For each transcription factor, the sizes of the cell type specific data sets, the intersection and the percentage of the test data set that is contained in the training data set is computed.

| TF | training | test | training size | test size | intersection | % test in train |
|---|---|---|---|---|---|---|
| Atf2 | Gm12878 | H1hesc | 23467 | 5998 | 1969 | 32.83 |
| Atf3 | K562 | Gm12878 | 16011 | 1677 | 1226 | 73.11 |
| Atf3 | K562 | H1hesc | 16011 | 4808 | 3241 | 67.41 |
| Bach1 | H1hesc | K562 | 11457 | 3806 | 1921 | 50.47 |
| Bcl11a | Gm12878 | H1hesc | 17876 | 2518 | 90 | 3.57 |
| Bcl3 | Gm12878 | K562 | 15455 | 1603 | 337 | 21.02 |
| Bclaf | Gm12878 | K562 | 6114 | 4444 | 1434 | 32.27 |
| Bhlhe40 | K562 | Gm12878 | 22497 | 13986 | 5553 | 39.70 |
| Brca1 | H1hesc | Gm12878 | 2025 | 551 | 524 | 95.10 |
| Cebpb | K562 | Gm12878 | 22240 | 5786 | 820 | 14.17 |
| Cebpb | K562 | H1hesc | 22240 | 15557 | 5615 | 36.09 |
| Cfos | K562 | Gm12878 | 7646 | 2239 | 1948 | 87.00 |
| Chd1a | K562 | Gm12878 | 9350 | 6668 | 1709 | 25.63 |
| Chd1a | K562 | H1hesc | 9350 | 2191 | 814 | 37.15 |
| Chd2 | Gm12878 | H1hesc | 15597 | 6849 | 3952 | 57.70 |
| Chd2 | Gm12878 | K562 | 15597 | 7797 | 4377 | 56.14 |
| Cjun | K562 | H1hesc | 8827 | 2148 | 653 | 30.40 |
| Cmyc | K562 | Gm12878 | 24153 | 3690 | 2928 | 79.35 |
| Cmyc | K562 | H1hesc | 24153 | 4551 | 3076 | 67.59 |
| Corest | K562 | Gm12878 | 6371 | 1397 | 158 | 11.31 |
| Ctcf | H1hesc | Gm12878 | 66551 | 44982 | 39204 | 87.15 |
| Ctcf | H1hesc | K562 | 66551 | 51992 | 42014 | 80.81 |
| E2f4 | K562 | Gm12878 | 8181 | 3440 | 2265 | 65.84 |
| Egr1 | K562 | Gm12878 | 36997 | 16331 | 11099 | 67.96 |
| Egr1 | K562 | H1hesc | 36997 | 8743 | 5403 | 61.80 |
| Elf1 | K562 | Gm12878 | 27780 | 23008 | 13412 | 58.29 |
| Elk | Gm12878 | K562 | 5584 | 2961 | 1914 | 64.64 |
| Ets1 | K562 | Gm12878 | 10726 | 4120 | 2627 | 63.76 |
| Ezh | H1hesc | Gm12878 | 6370 | 2472 | 611 | 24.72 |
| Ezh | H1hesc | K562 | 6370 | 1685 | 281 | 16.68 |
| Fosl1 | K562 | H1hesc | 11174 | 1113 | 349 | 31.36 |
| Gabp | K562 | Gm12878 | 14393 | 6566 | 5175 | 78.82 |
| Gabp | K562 | H1hesc | 14393 | 5653 | 2803 | 49.58 |
| Gtf2f1 | K562 | H1hesc | 3621 | 3548 | 1195 | 33.68 |
| Hdac2 | H1hesc | K562 | 5644 | 5247 | 426 | 8.12 |
| Jund | K562 | Gm12878 | 40052 | 2472 | 445 | 18.00 |
| Jund | K562 | H1hesc | 40052 | 8447 | 3843 | 45.50 |
| Max | K562 | Gm12878 | 46171 | 12542 | 8545 | 68.13 |
| Max | K562 | H1hesc | 46171 | 11129 | 5750 | 51.67 |
| Maz | K562 | Gm12878 | 33323 | 18952 | 11831 | 62.43 |
| Mef2a | Gm12878 | K562 | 17605 | 5631 | 1335 | 23.71 |
| Mxi1 | Gm12878 | H1hesc | 17735 | 6351 | 3532 | 55.61 |
| Mxi1 | Gm12878 | K562 | 17735 | 6711 | 4163 | 62.03 |
| Nfe2 | K562 | Gm12878 | 2637 | 772 | 240 | 31.09 |
| Nfya | K562 | Gm12878 | 4286 | 1841 | 1538 | 83.54 |
| Nfyb | Gm12878 | K562 | 13295 | 10096 | 7270 | 72.01 |
| Nrf1 | Gm12878 | H1hesc | 5683 | 4513 | 3447 | 76.38 |
| Nrf1 | Gm12878 | K562 | 5683 | 4211 | 3342 | 79.36 |
| Nrsf | K562 | Gm12878 | 15849 | 6906 | 4512 | 65.33 |
| Nrsf | K562 | H1hesc | 15849 | 13286 | 5975 | 44.97 |
| P300 | H1hesc | Gm12878 | 8934 | 5168 | 733 | 14.18 |
| P300 | H1hesc | K562 | 8934 | 2674 | 162 | 6.06 |
| Pml | Gm12878 | K562 | 16678 | 15895 | 6876 | 43.26 |
| Pu1 | Gm12878 | K562 | 42938 | 28677 | 13902 | 48.48 |
| Rad21 | H1hesc | Gm12878 | 75680 | 40019 | 33859 | 84.61 |
| Rbbp5 | H1hesc | K562 | 16151 | 14258 | 6688 | 46.91 |
| Rfx5 | Gm12878 | H1hesc | 4341 | 1695 | 731 | 43.13 |
| Rfx5 | Gm12878 | K562 | 4341 | 2201 | 907 | 41.21 |
| Rxra | Gm12878 | H1hesc | 1704 | 1306 | 95 | 7.27 |
| Sin3a | K562 | Gm12878 | 12700 | 10392 | 5245 | 50.47 |
| Sin3a | K562 | H1hesc | 12700 | 8977 | 3697 | 41.18 |
| Six5 | Gm12878 | H1hesc | 4839 | 3425 | 2560 | 74.74 |
| Six5 | Gm12878 | K562 | 4839 | 4194 | 3353 | 79.95 |
| Smc3 | Gm12878 | K562 | 30517 | 23598 | 18883 | 80.02 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Sp1 | Gm12878 | H1hesc | 18248 | 15110 | 5524 | 36.56 |
| Sp1 | Gm12878 | K562 | 18248 | 7206 | 4510 | 62.59 |
| Sp2 | K562 | H1hesc | 3124 | 2469 | 2005 | 81.21 |
| Srf | Gm12878 | H1hesc | 8544 | 5105 | 2112 | 41.37 |
| Srf | Gm12878 | K562 | 8544 | 4717 | 2464 | 52.24 |
| Stat1 | K562 | Gm12878 | 2203 | 1769 | 50 | 2.83 |
| Stat5 | K562 | Gm12878 | 9811 | 7423 | 656 | 8.84 |
| Taf1 | H1hesc | Gm12878 | 20547 | 14278 | 8850 | 61.98 |
| Taf1 | H1hesc | K562 | 20547 | 15246 | 10654 | 69.88 |
| Taf7 | H1hesc | K562 | 10475 | 3422 | 1963 | 57.36 |
| Tblr1 | Gm12878 | K562 | 13702 | 5086 | 1177 | 23.14 |
| Tbp | K562 | Gm12878 | 17558 | 14893 | 5952 | 39.97 |
| Tbp | K562 | H1hesc | 17558 | 17194 | 8694 | 50.56 |
| Tcf12 | Gm12878 | H1hesc | 20437 | 7833 | 1604 | 20.48 |
| Tead4 | K562 | H1hesc | 31030 | 19857 | 3108 | 15.65 |
| Tr4 | Gm12878 | K562 | 1263 | 587 | 416 | 70.87 |
| Usf1 | H1hesc | Gm12878 | 26042 | 9778 | 5928 | 60.63 |
| Usf1 | H1hesc | K562 | 26042 | 18521 | 9634 | 52.02 |
| Usf2 | Gm12878 | H1hesc | 9022 | 6952 | 3136 | 45.11 |
| Usf2 | Gm12878 | K562 | 9022 | 3083 | 1903 | 61.73 |
| Yy1 | Gm12878 | H1hesc | 30994 | 18328 | 9798 | 53.46 |
| Yy1 | Gm12878 | K562 | 30994 | 12677 | 9788 | 77.21 |
| Zbtb33 | K562 | Gm12878 | 3285 | 2144 | 1171 | 54.62 |
| Znf143 | H1hesc | Gm12878 | 30687 | 20024 | 14791 | 73.87 |
| Znf143 | H1hesc | K562 | 30687 | 29069 | 18178 | 62.53 |

## Text S6.3 Parameter settings

We assess the prediction performance of MEME, TFFMs, DiChIPMunk, and Dimont using different motif models on the ENCODE data sets for different cell types. We train each of the approaches on the ChIP-seq data set measured for the cell type with the largest number of peaks and test its prediction accuracy on the ChIP-seq data sets for the remaining cell types. This scenario resembles likely practical applications, e.g., combining computational predictions with cell type-specific DNase I hypersensitivity experiments. We train each of the approaches according to the suggestions of the corresponding publications. We test each of the approaches on the ChIP-seq data sets for the remaining cell type(s) using the identical sequences of length 1000 bp around the peak center for all approaches considered to obtain comparable results, but obtain highly similar results using sequences of length 100 bp (data not shown).

We extract for each approach the most suitable sub-sequences in the peak region as noted in the original publications. More specifically, we extract sequences for

**MEME and TFFM** using 50 bp on each flank of the peak summit as suggested by (13); for training MEME, we use only the top 500 peaks of each data set according to the peak statistic;

**DiChIPMunk** according to the peak boundaries given in the narrowPeak file and annotate these using a triangular “prior” with its maximum at the peak summit (cf. (14));

**Dimont** using 500 bp on each flank of the peak center as suggested by (15).

We train each of the approaches as follows:

**MEME** is trained using DNA alphabet and default parameters on the top 500 peaks as is the standard procedure of several publications;

**DiChIPMunk** is trained for a motif length between 10 and 25 bp, a ZOOPS factor of 1.0, using peak data and the remaining parameters set to their defaults (14);

**TFFMs** are initialized by the MEME result on the sequences of the top peaks and trained using “first order” and “detailed” models on the complete data sets (13);

**Dimont** is trained as described previously (15) with minor modifications to the initialization strategy and shift heuristic.

While the original version of Dimont used single 7-mers for initialization (15), we augment the initialization set to all 20-mers for that the central 7-mer has a Hamming distance of at most 3 to the original 7-mer. We modify the length adaption heuristic of Dimont to a shift heuristic that preserves the motif length and allows shifts of the motif model of at most 5 positions in either direction. We shift positions out of the model if these neither have a Kullback-Leibler divergence to the background distribution of nucleotides greater than 0.2 nor a mutual information above 0.2 to any of the other motif positions. If such positions exists at both flanks of the model, we shift the motif model such that the conserved part lies in the model center.

We start MEME using the command line

```
meme -p 3 -dna <sequences.fa>
```

We start DiChIPMunk using the command line

```
java -Xms512M -Xmx4G autosome.ru.di.ChIPMunk 10 25 yes 1.0
        p:<sequences.fa> 200 20 1 8
```

We learn first order TFFMs using custom python code checked against the web-application at `http://cisreg.cmmt.ubc.ca/TFFM/`:

```
import sys

sys.path.append("./TFFM-master")

import tffm_module

from constants import TFFM_KIND

tffm_first_order = tffm_module.tffm_from_meme("meme_out/meme.txt", TFFM_KIND.FIRST_ORDER)

tffm_first_order.train(sys.argv[1])
tffm_first_order.write("tffm_first_order.xml")
```

and in complete analogy (using `TFFM_KIND.DETAILED`) for detailed TFFMs.

## Text S6.4 Binding site prediction

For the assessment, we need a single prediction score for each input sequence. Following the suggestions of the original publications, we use the maximum score of a sliding window of the motif width in case of MEME, TFFMs, and DiChIPMunk (cf. (13, 14)) and the ZOOPS score in case of Dimont (cf. (15); We also test the maximum score in case of Dimont and obtain similar results (data not shown).

We only use the first motif model (according to internal ranking) returned by MEME and Dimont, because DiChIPMunk and TFFMs return only a single model.

For predictions, we test in case of MEME for each performance measure the maximum score of the "scoring matrix" and the "probability matrix" and chose that matrix yielding the better performance. In the same manner, we test both scoring matrices returned by DiChIPMunk and we test the "first order" and the "detailed" TFFM and decide for the better option for each performance measure.

We make predictions using the weight matrices of MEME and the di-nucleotide models of DiChIPMunk using custom Java code.

We make predictions for TFFMS using custom python code:

```
import sys

sys.path.append("./TFFM-master")

import tffm_module

from constants import TFFM_KIND

tffm_first_order = tffm_module.tffm_from_xml(sys.argv[2], TFFM_KIND.FIRST_ORDER)

for hit in tffm_first_order.scan_sequences(sys.argv[1],only_best=True):
    if hit:
        print hit
```

and in complete analogy (using `TFFM_KIND.DETAILED`) for detailed TFFMs.

## Text S6.5 Comparison to other tools

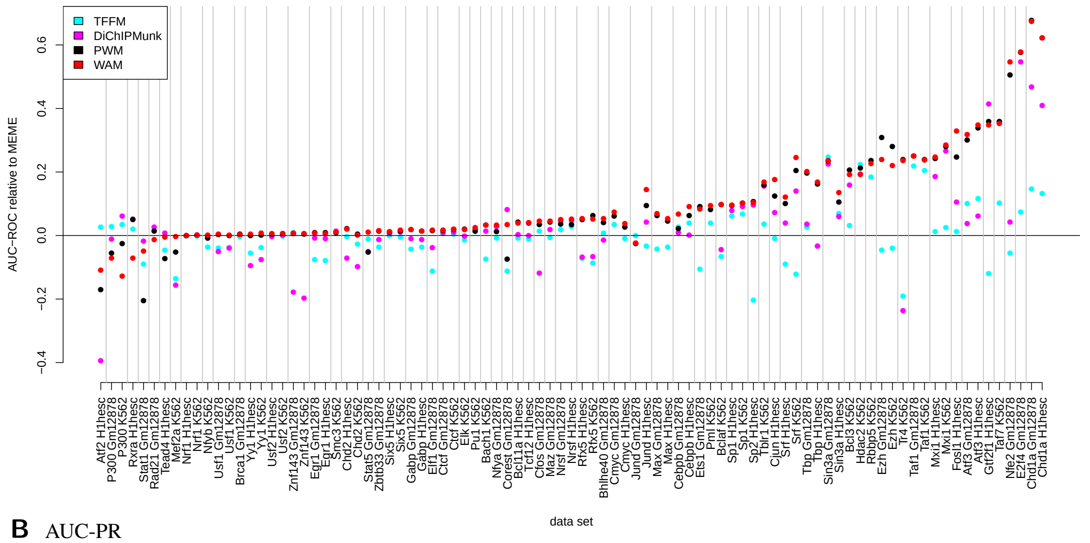
In Figure S6A, we plot the AUC-ROC values, which is a widely accepted measure for ChIP-seq prediction accuracy (6, 13, 15, 83, 84), achieved by DiChIPMunk, TFFMs and the Dimont framework relative to the performance achieved by MEME. Here, each of the approaches is trained on the data set for one cell type and predictions are made on the data set(s) for the remaining cell types of the Tier1 data sets of the ENCODE project. We find that for approximately half of the data sets, DiChIPMunk and TFFMs yield a better classification performance than MEME, while for the other half the performance observed in worse than that of MEME. In contrast, Dimont using PWMs or WAMs scores worse than MEME only for a small fraction of data sets, whereas it yields a considerably larger AUC-ROC than MEME for more than half of the data sets. Dimont also performs better than DiChIPMunk and TFFMs for approximately four thirds of the data set, although to a varying degree. This picture is widely consistent for AUC-PR, weighted AUC-ROC and AUC-PR, and Pearson and Spearman correlation (cf. Figures S6 and S7). It is important to note that this does *not* mean that MEME or the other tools fail to infer informative motifs from these ChIP-seq data sets (cf. Figures S10 and S11), since for most data sets, all of the tools considered yield for instance AUC-ROC values greater than 0.5, which would be the performance of a random guesser.

Notably, the performance of all approaches substantially drops for AUC-ROC and AUC-PR using di-nucleotide shuffled versions of the top 500 sequences (Figures S8 and S12). One explanation for this observation is that Dimont using WAMs, DiChIPMunk and TFFMs consider dependencies between adjacent di-nucleotides in their models, which complicates the classification task if the negative sequences preserve an identical di-nucleotide composition. In part, this may also apply to Dimont using PWMs, since discriminative learning principles work substantially better than generative ones if the model assumption is wrong, and may to some extent accomodate the lack of di-nucleotide features. To further investigate this issue, we re-train the Dimont models on training data that additionally contain di-nucleotide shuffled version of the *training* data as negative examples. We find (Figures S9 and S13) that in this case, Dimont using PWMs and WAMs yields a substantially better classification performance and largely restores the tendencies observed for the other performance measures in Figures S6, S7, S10, and S11. This underlines that discriminative learning principles like the weighted MSP principle employed by Dimont profit from a careful selection of negative examples in training and, hence, negative data should be selected application-specific. However, for the following studies comparing different models within the Dimont framework, we use the same training data for all performance measures to keep models and classifiers consistent between different performance measures and within individual iterations of cross-validation experiments.

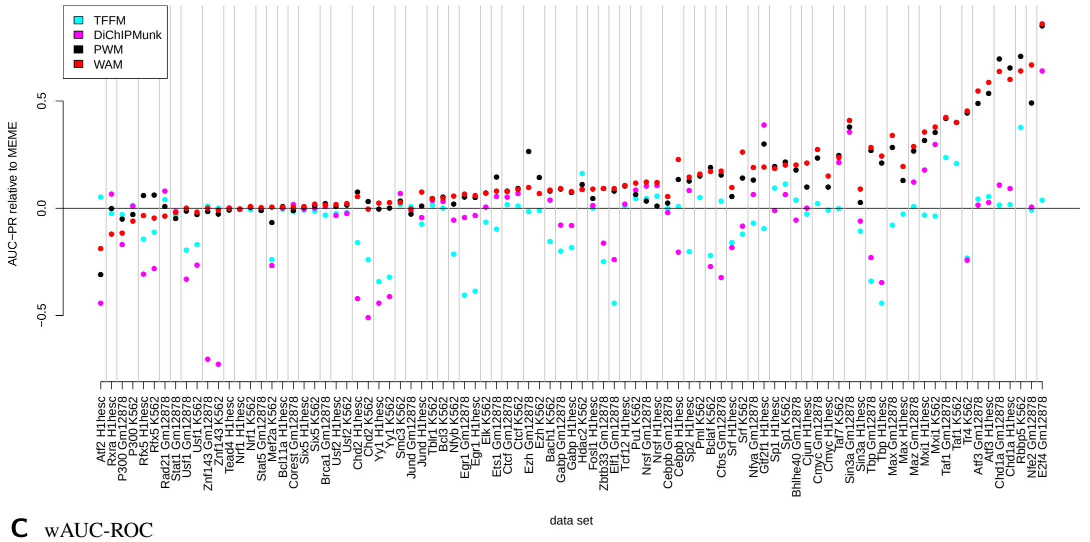
The comparison across cell types might be biased by overlaps of the ChIP regions of different cell types and, hence, might be skewed by overfitting effects. To investigate this issue, we compute the overlap between the cell-type specific ChIP-regions for each transcription factor (Table S1) and find considerable overlaps for several transcription factors.

Summarizing the results of this benchmark, Dimont yields at least the prediction performance achieved by alternative approaches using dependency models for the majority of ChIP-seq data sets and may, hence, serve as a solid framework for evaluating different dependency models including Slim and LSlim models in the following sections.

**A** AUC-ROC



**B** AUC-PR



**C** wAUC-ROC

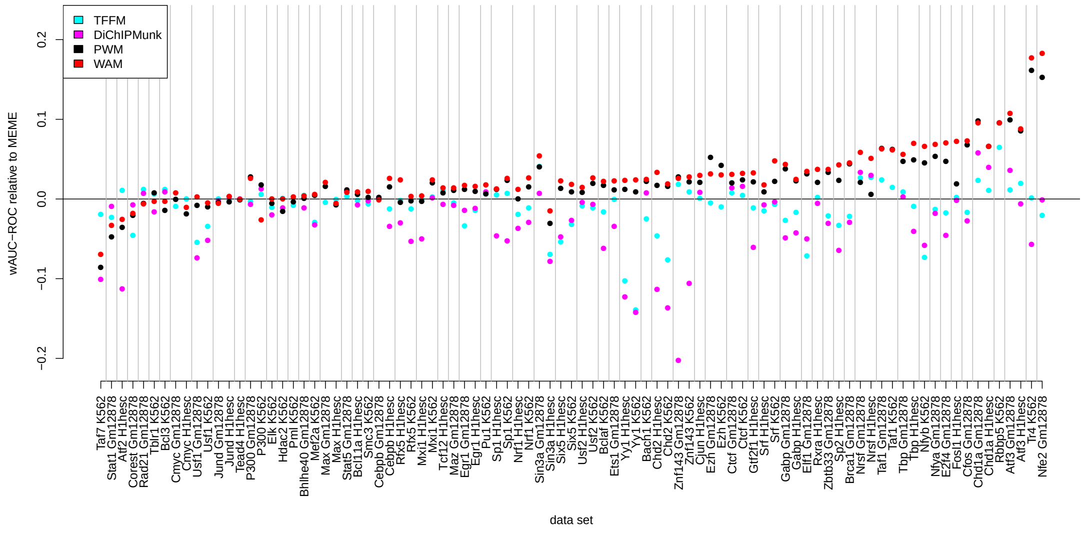


**Figure S6.** Comparison of the Dimont framework using PWM and WAM models to MEME (baseline), DiChIPMunk, and TFFMs. As performance measure, we use AUC-ROC and AUC-PR for the sequences under the top 500 peaks vs. randomly sampled genomic sequences, and wAUC-ROC using all sequences under peaks. We compute relative values by subtracting for each data set the corresponding value of MEME from those of the other approaches. Vertical lines separate the data sets of different transcription factors.

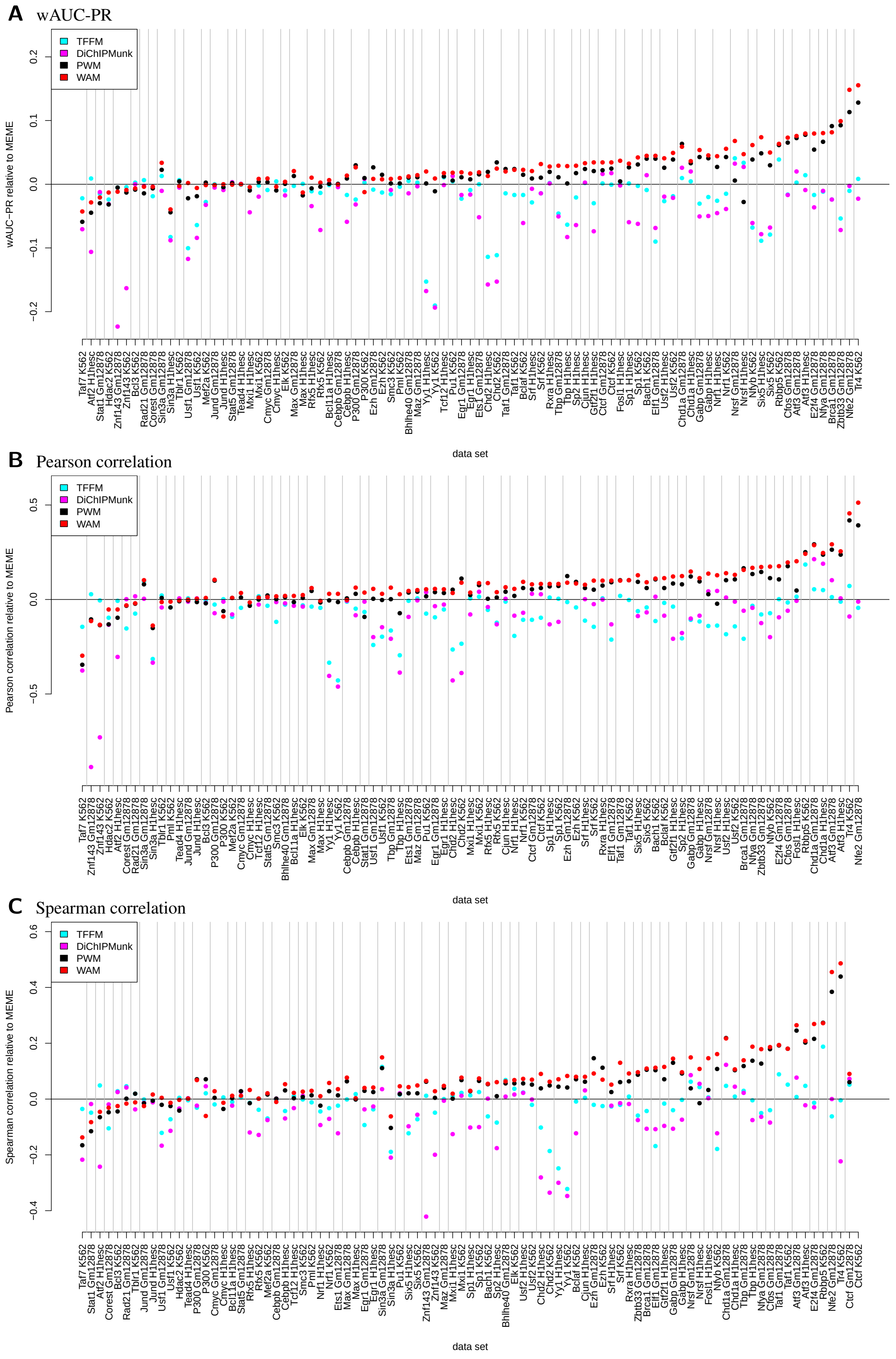
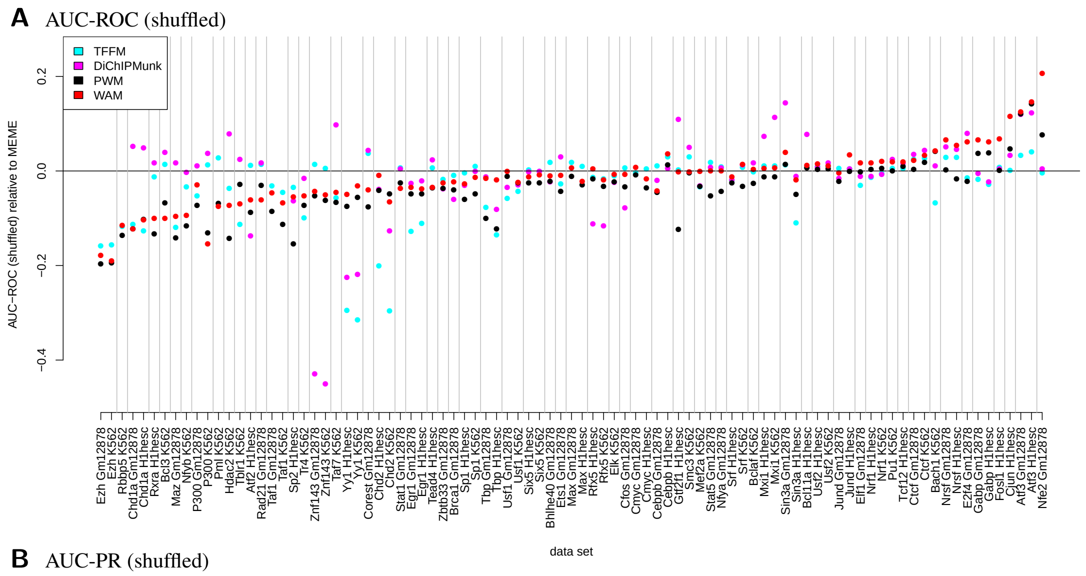


**Figure S7.** Comparison of the Dimont framework using PWM and WAM models to MEME (baseline), DiChIPMunk, and TFFMs. As performance measure, we use wAUC-PR, Pearson and Spearman correlation using all sequences under peaks. We compute relative values by subtracting for each data set the corresponding value of MEME from those of the other approaches.



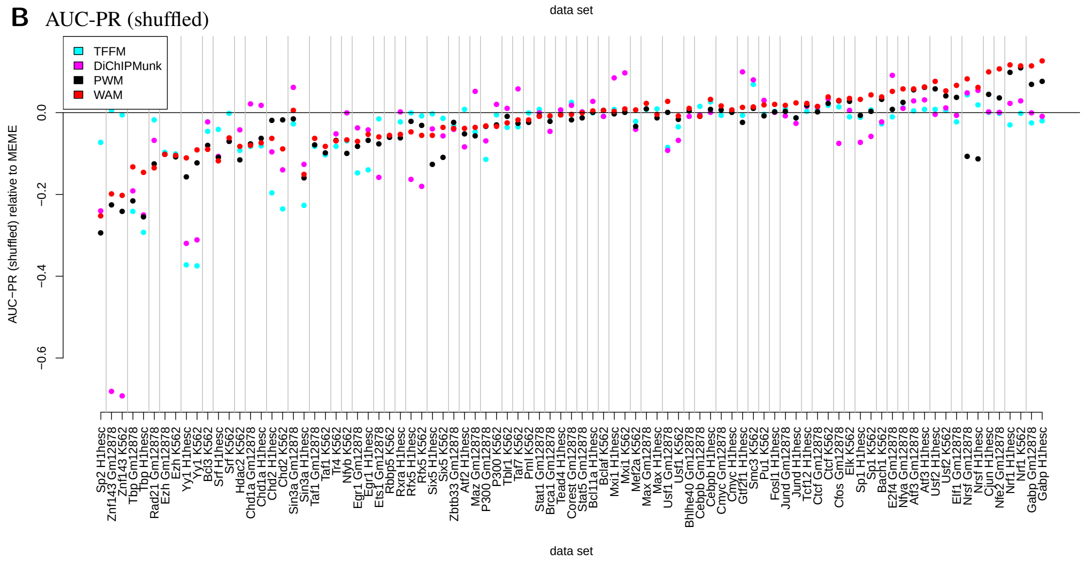
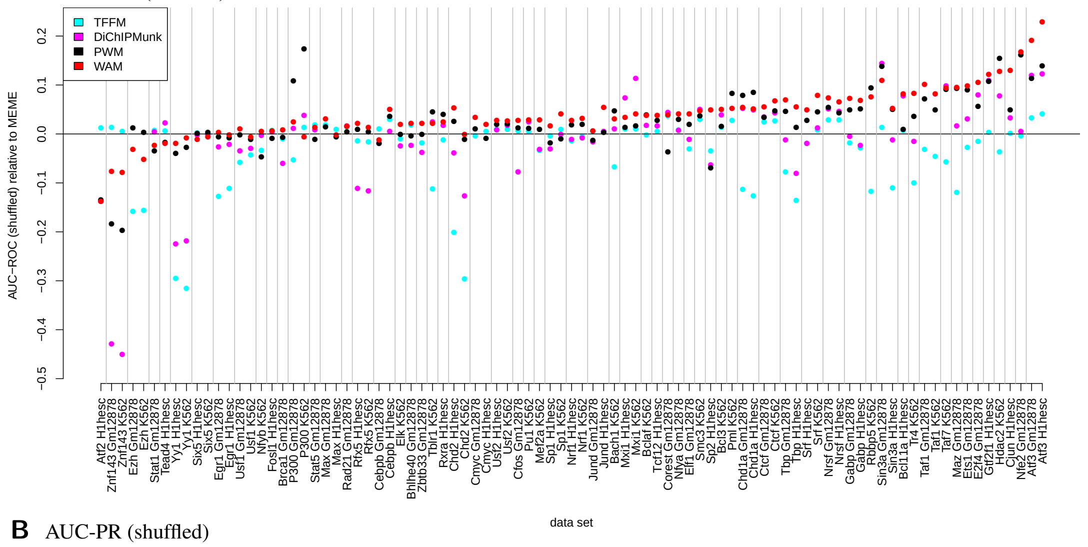


**Figure S8.** Comparison of the Dimont framework using PWM and WAM models to MEME (baseline), DiChIPMunk, and TFFMs. As performance measure, we use AUC-ROC and AUC-PR for the sequences under the top 500 peaks vs. di-nucleotide shuffled versions of the same sequences. We compute relative values by subtracting for each data set the corresponding value of MEME from those of the other approaches.

**A** AUC-ROC (shuffled)



**B** AUC-PR (shuffled)

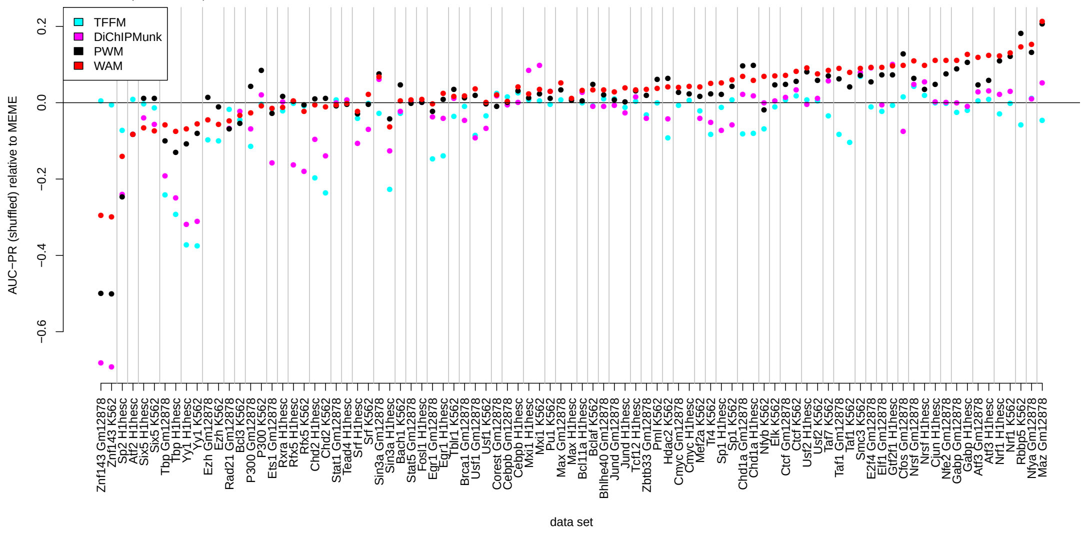
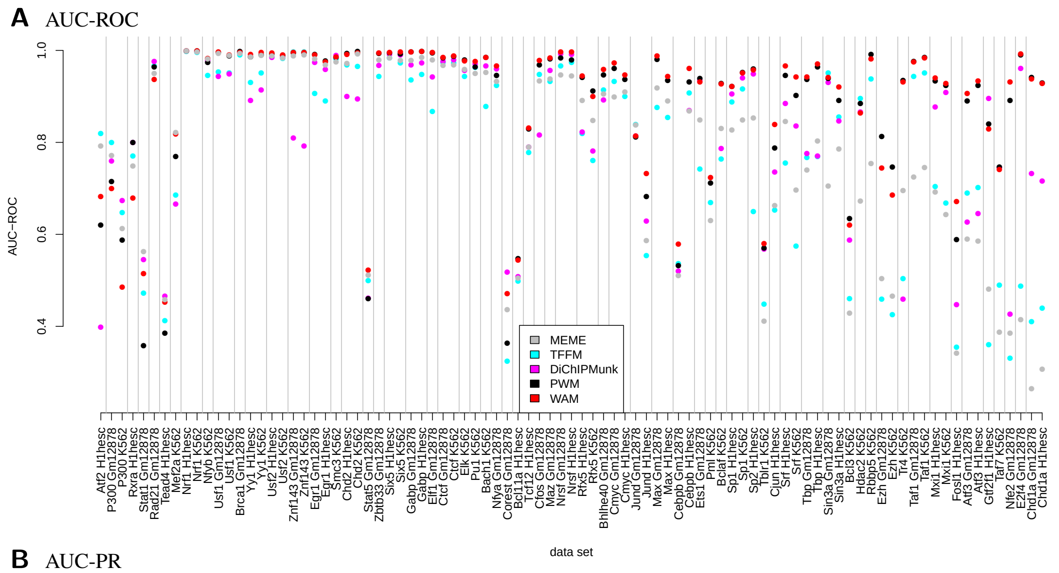
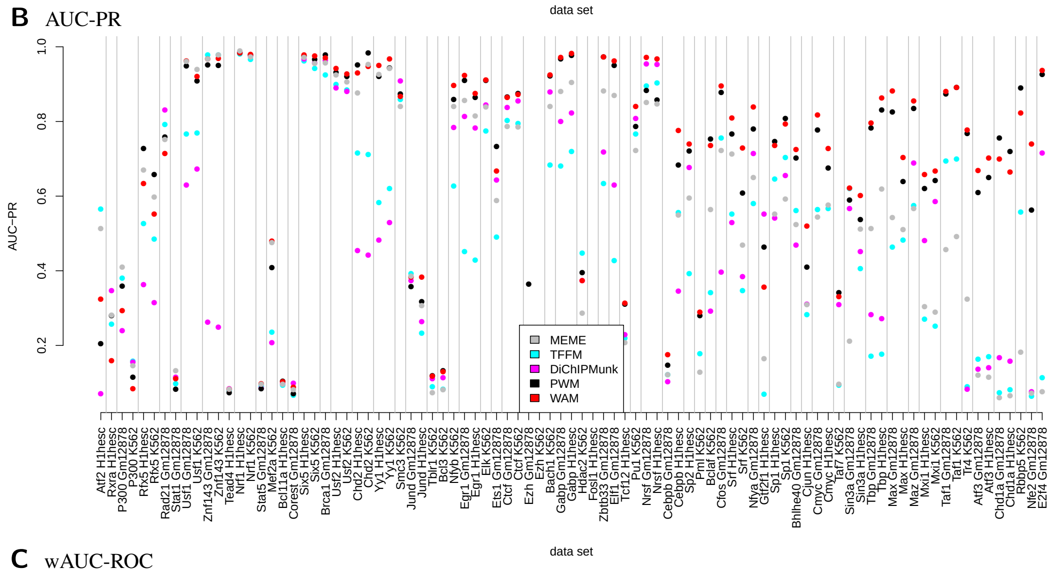


**Figure S9.** Comparison of the Dimont framework using PWM and WAM models to MEME (baseline), DiChIPMunk, and TFFMs. As performance measure, we use AUC-ROC and AUC-PR for the sequences under the top 500 peaks vs. di-nucleotide shuffled versions of the same sequences. In contrast to Figure S8, we trained Dimont using di-nucleotide shuffled versions of the *training* sequences as additional negative data, while MEME, DiChIPMunk and TFFMs do not consider negative data in their training methods and, hence, remain unchanged. We compute relative values by subtracting for each data set the corresponding value of MEME from those of the other approaches.





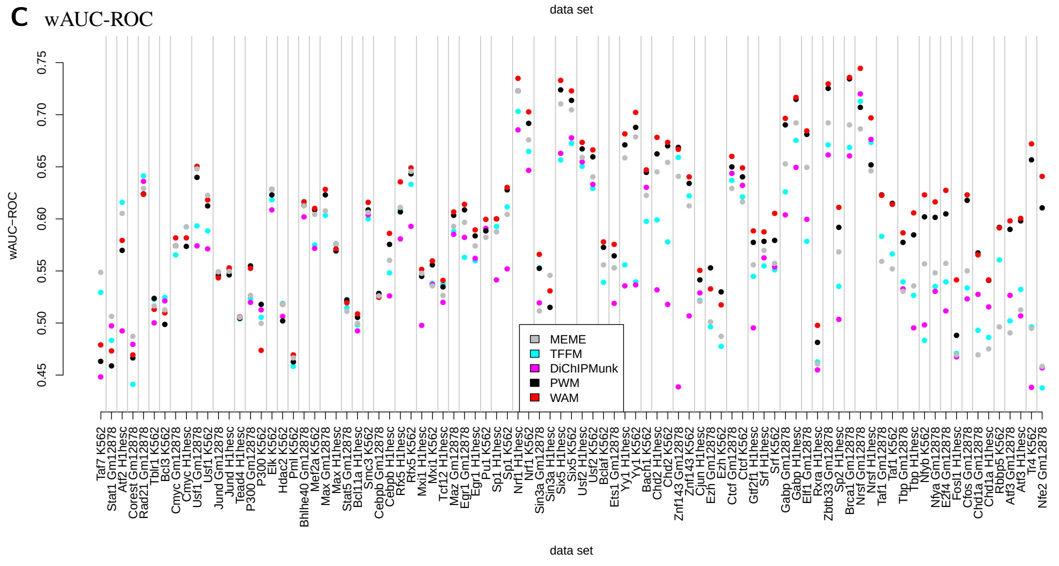


**Figure S10.** Comparison of the Dimont framework using PWM and WAM models to MEME, DiChIPMunk, and TFFMs. As performance measure, we use AUC-ROC and AUC-PR for the sequences under the top 500 peaks vs. randomly sampled genomic sequences, and wAUC-ROC using all sequences under peaks. In contrast to Figure S6, we present absolute values of the performance measures.

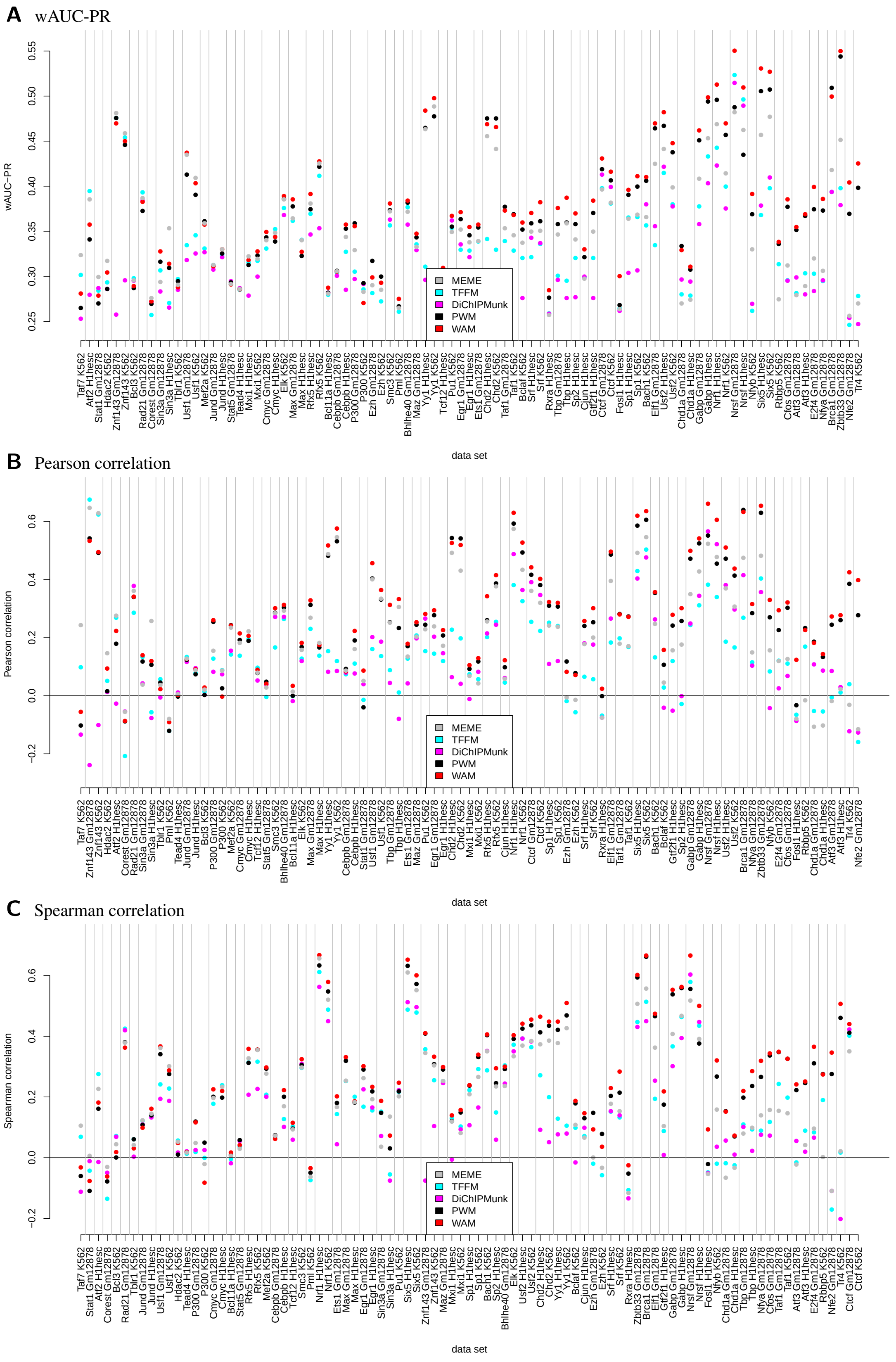
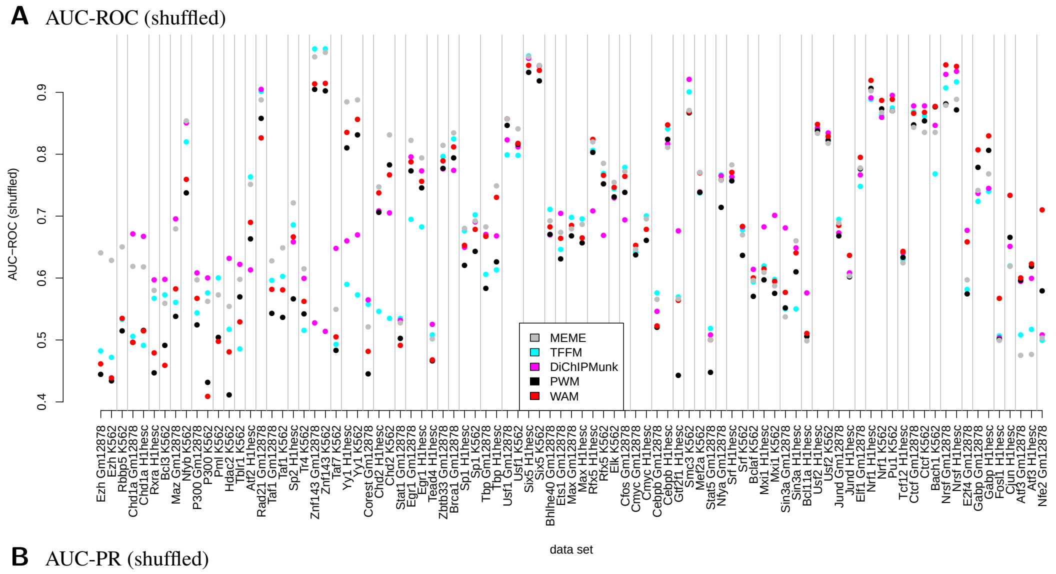


**Figure S11.** Comparison of the Dimont framework using PWM and WAM models to MEME, DiChIPMunk, and TFFMs. As performance measure, we use wAUC-PR, Pearson and Spearman correlation using all sequences under peaks. In contrast to Figure S7, we present absolute values of the performance measures.



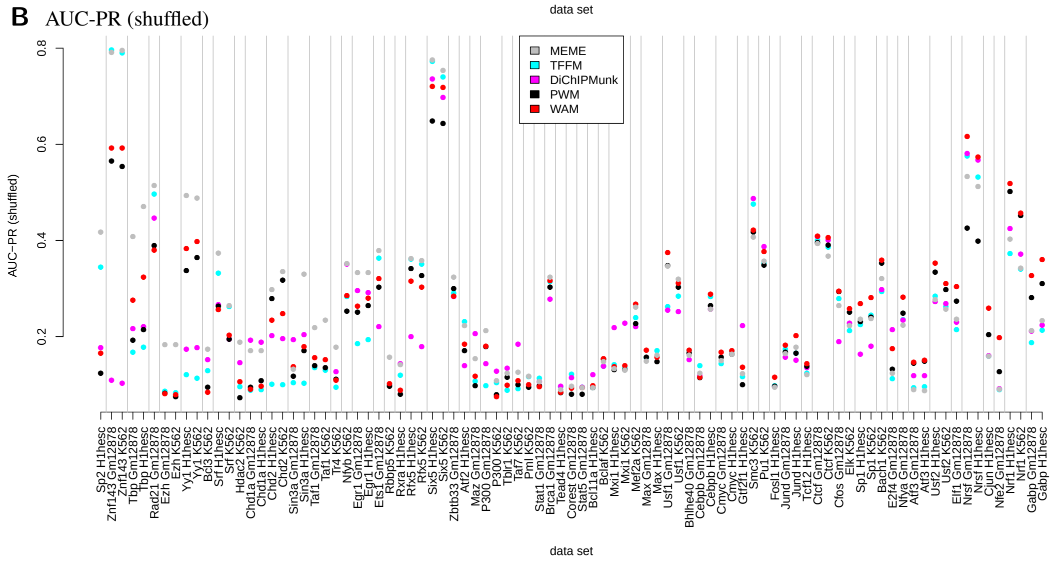
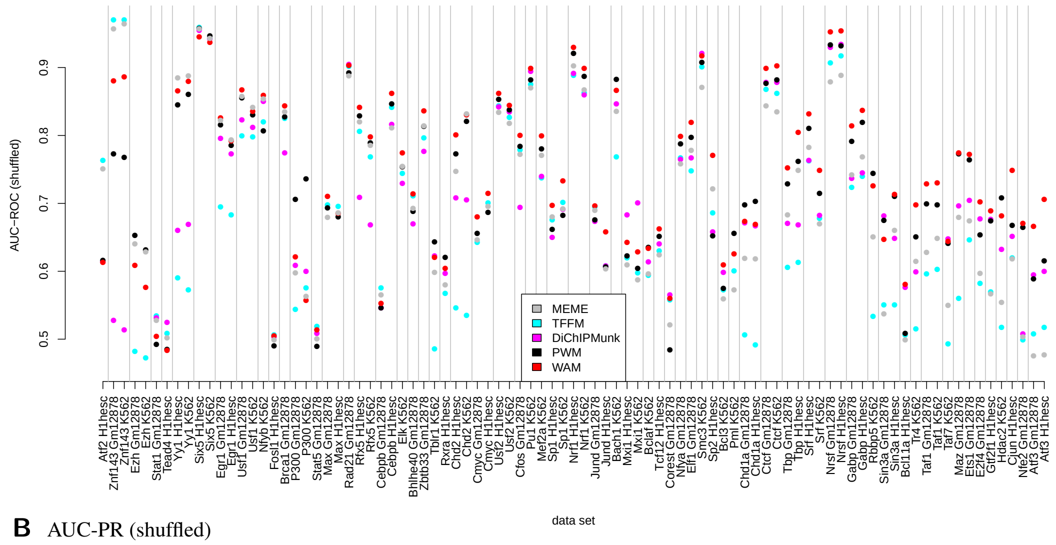


**Figure S12.** Comparison of the Dimont framework using PWM and WAM models to MEME, DiChIPMunk, and TFFMs. As performance measure, we use AUC-ROC and AUC-PR for the sequences under the top 500 peaks vs. di-nucleotide shuffled versions of the same sequences. In contrast to Figure S8, we present absolute values of the performance measures.

**A** AUC-ROC (shuffled)



**B** AUC-PR (shuffled)

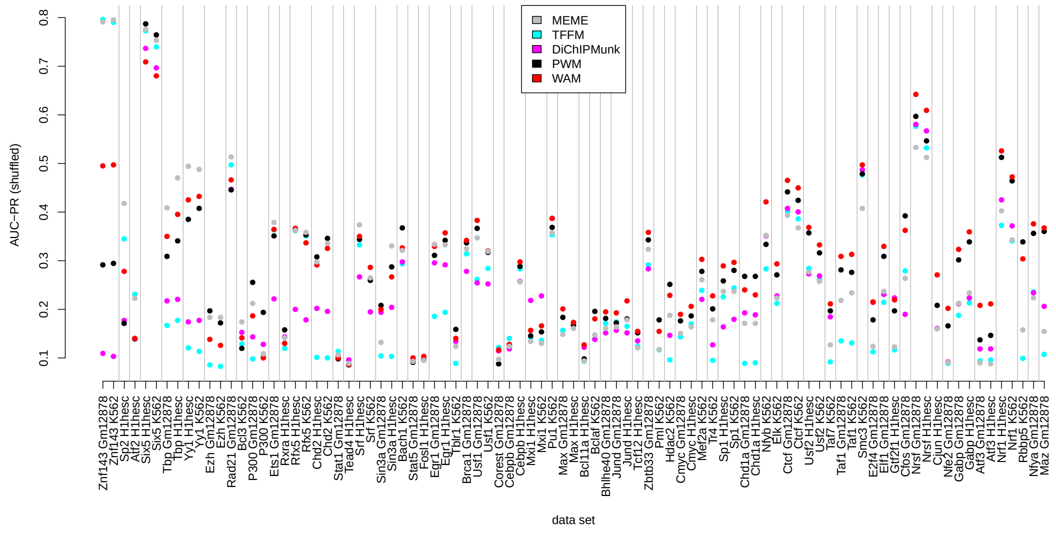


**Figure S13.** Comparison of the Dimont framework using PWM and WAM models to MEME, DiChIPMunk, and TFFMs. As performance measure, we use AUC-ROC and AUC-PR for the sequences under the top 500 peaks vs. di-nucleotide shuffled versions of the same sequences. In contrast to Figure S12, we trained Dimont using di-nucleotide shuffled versions of the *training* sequences as additional negative data, while MEME, DiChIPMunk and TFFMs do not consider negative data in their training methods and, hence, remain unchanged. In contrast to Figure S9, we present absolute values of the performance measures.

## Text S6.6 Comparison of models

We compare the performance of WAM, Slim, and LSlim(5) models to that of PWM models on the data sets of the Tier1 data sets of the ENCODE project, i.e., on the same data sets that we also use for the comparison to other tools in Text S6.5. We present the results of this comparison in Figures S14 and S15.

First, we focus on wAUC-PR as performance measure depicted in Figures S15A, since wAUC-PR measures the ability of classifying highly occupied peaks from less occupied ones but also the ability of predicting peak abundances from sequence data. We find that for the majority of data sets, all of the dependency models (WAM, Slim, or LSlim) yield an improved prediction performance compared to the PWM assuming position independence. WAM, Slim and LSlim models each yield the maximum performance for approximately one third of the data sets, where the exact proportions vary slightly for other performance measures.

The absolute improvements of wAUC-PR vary substantially between data sets. For some data sets we find no considerable improvement over the PWM model (e.g., Bach1, Chd1a, Elf1, Elk, Tcf12), whereas for other data sets we find a considerable improvement (e.g., Atf3, Brca1, E2F4, Fosl1, Gtf2f1, Nfya, Nfyb, Nfe2, Nrsf). Notably, for some of the data sets (e.g., Brca1, E2f4, Nfyb, Six5), we see an improvement for the performance measures taking peak statistics[1] into account (wAUC-ROC, wAUC-PR, correlation) but less for the classification-related performance measures (AUC-ROC, AUC-PR), which might be an indication that the dependencies discovered are less relevant for binding sites under the peaks with the largest peak statistic.

Second, we compare the models in a cross validation as depicted in Figures S17 and S18. Finally in Table S2, we directly compare the models by analyzing how often a model is significantly better or worse than any other model. Both aspects are discussed in the main manuscript.

**Table S2.** Overview of significant improvements. For each combination of models, we count for how many of the 63 data sets one model (rows) achieves a significantly better performance (difference greater 2-fold standard error in a 10-fold cross validation) than the other model (columns).

**A** wAUC-PR

| | better than | | | |
|---|---|---|---|---|
| | PWM | WAM | LSlim(5) | Slim |
| PWM | 0 | 4 | 1 | 0 |
| WAM | 25 | 0 | 2 | 3 |
| LSlim(5) | 36 | 21 | 0 | 4 |
| Slim | 30 | 12 | 0 | 0 |

**B** AUC-PR

| | better than | | | |
|---|---|---|---|---|
| | PWM | WAM | LSlim(5) | Slim |
| PWM | 0 | 13 | 4 | 4 |
| WAM | 19 | 0 | 2 | 5 |
| LSlim(5) | 26 | 22 | 0 | 3 |
| Slim | 22 | 21 | 4 | 0 |

**C** wAUC-ROC

| | better than | | | |
|---|---|---|---|---|
| | PWM | WAM | LSlim(5) | Slim |
| PWM | 0 | 2 | 1 | 0 |
| WAM | 32 | 0 | 1 | 4 |
| LSlim(5) | 41 | 15 | 0 | 1 |
| Slim | 31 | 15 | 0 | 0 |

**D** AUC-ROC

| | better than | | | |
|---|---|---|---|---|
| | PWM | WAM | LSlim(5) | Slim |
| PWM | 0 | 11 | 2 | 0 |
| WAM | 22 | 0 | 1 | 1 |
| LSlim(5) | 27 | 23 | 0 | 1 |
| Slim | 25 | 20 | 1 | 0 |

**E** Pearson correlation

| | better than | | | |
|---|---|---|---|---|
| | PWM | WAM | LSlim(5) | Slim |
| PWM | 0 | 4 | 1 | 1 |
| WAM | 29 | 0 | 3 | 6 |
| LSlim(5) | 39 | 15 | 0 | 3 |
| Slim | 35 | 15 | 0 | 0 |

**F** Spearman correlation

| | better than | | | |
|---|---|---|---|---|
| | PWM | WAM | LSlim(5) | Slim |
| PWM | 0 | 1 | 0 | 0 |
| WAM | 26 | 0 | 1 | 3 |
| LSlim(5) | 38 | 15 | 0 | 1 |
| Slim | 31 | 14 | 0 | 0 |

**G** AUC-ROC (shuffled)

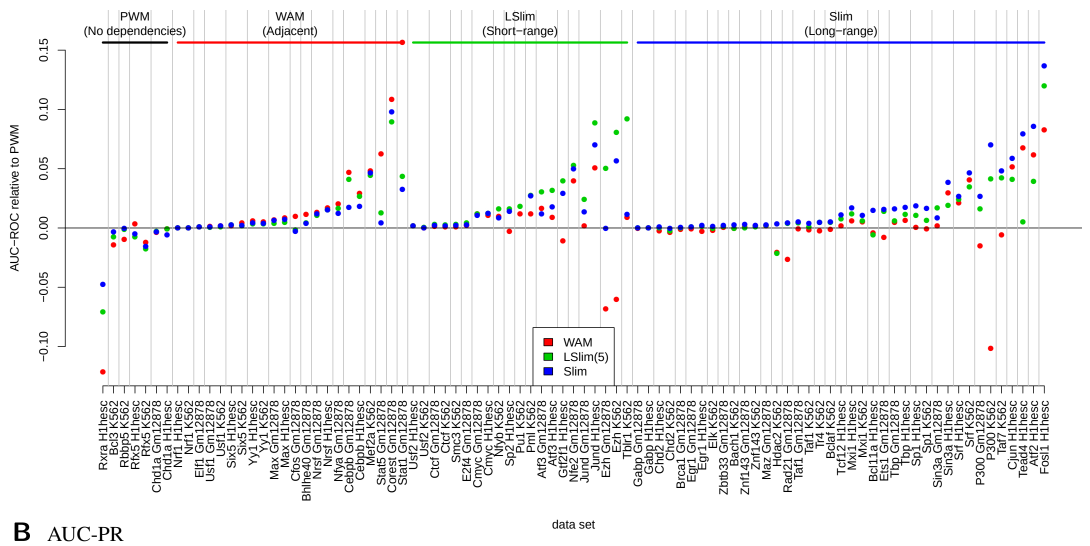
| | better than | | | |
|---|---|---|---|---|
| | PWM | WAM | LSlim(5) | Slim |
| PWM | 0 | 2 | 2 | 5 |
| WAM | 33 | 0 | 7 | 23 |
| LSlim(5) | 26 | 8 | 0 | 12 |
| Slim | 17 | 6 | 1 | 0 |

**H** AUC-PR (shuffled)

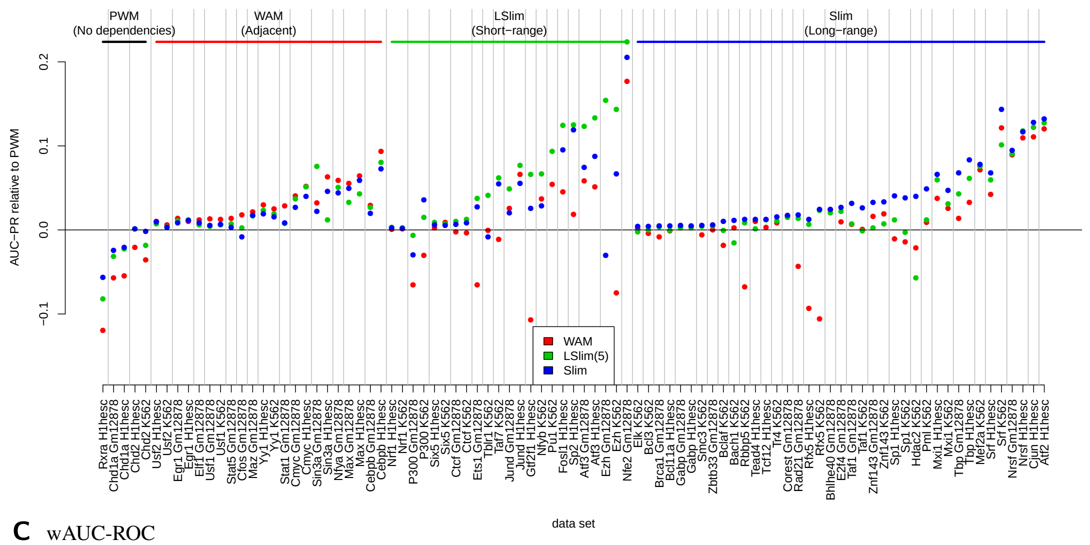
| | better than | | | |
|---|---|---|---|---|
| | PWM | WAM | LSlim(5) | Slim |
| PWM | 0 | 3 | 2 | 3 |
| WAM | 16 | 0 | 6 | 16 |
| LSlim(5) | 18 | 11 | 0 | 6 |
| Slim | 11 | 8 | 0 | 0 |

[1] "Peak statistics" refers to the peak-specific score reported by most peak callers, which is typically related to the abundance of reads under a ChIP-seq peak.

**A** AUC-ROC



**B** AUC-PR



**C** wAUC-ROC

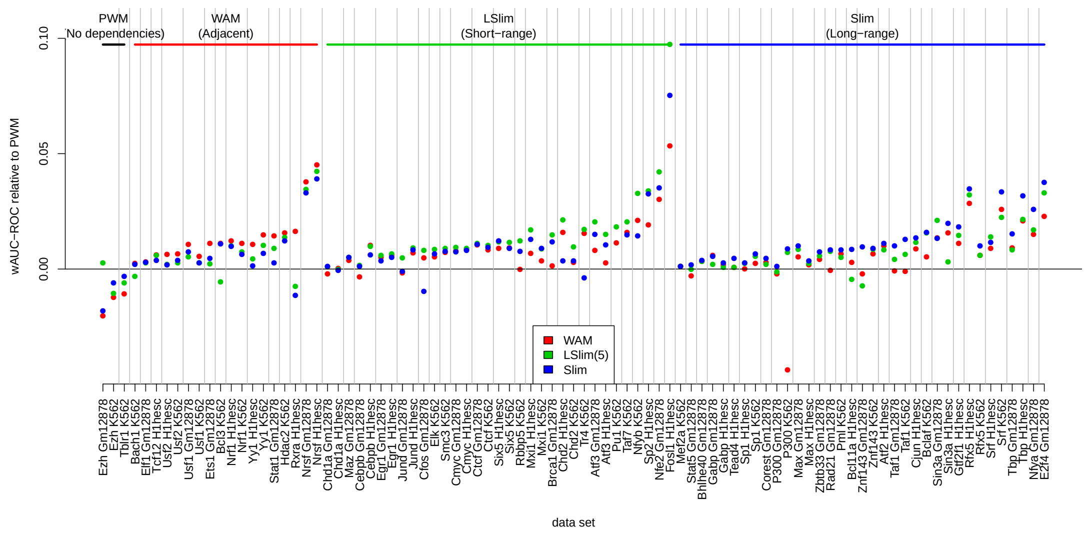
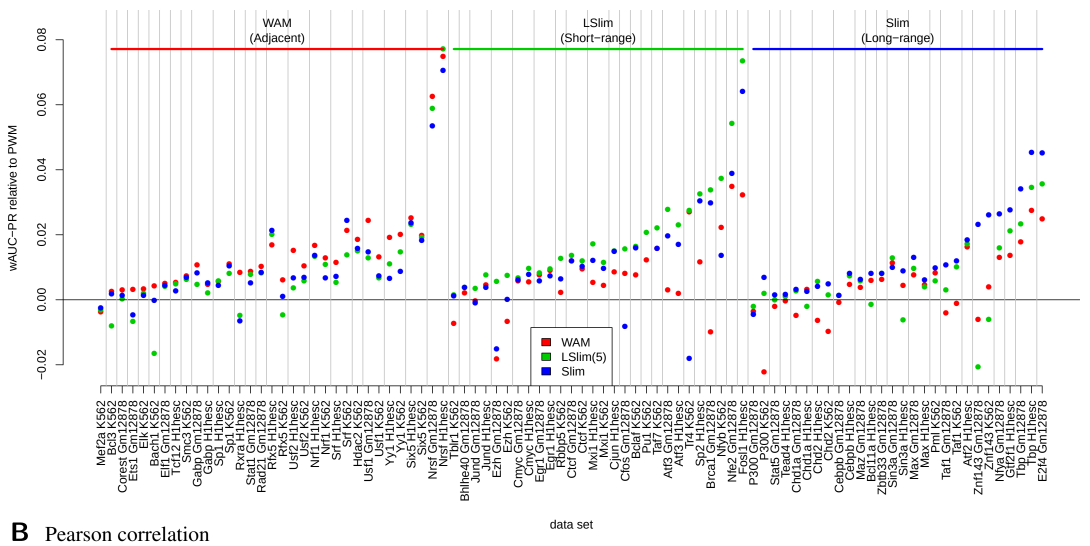
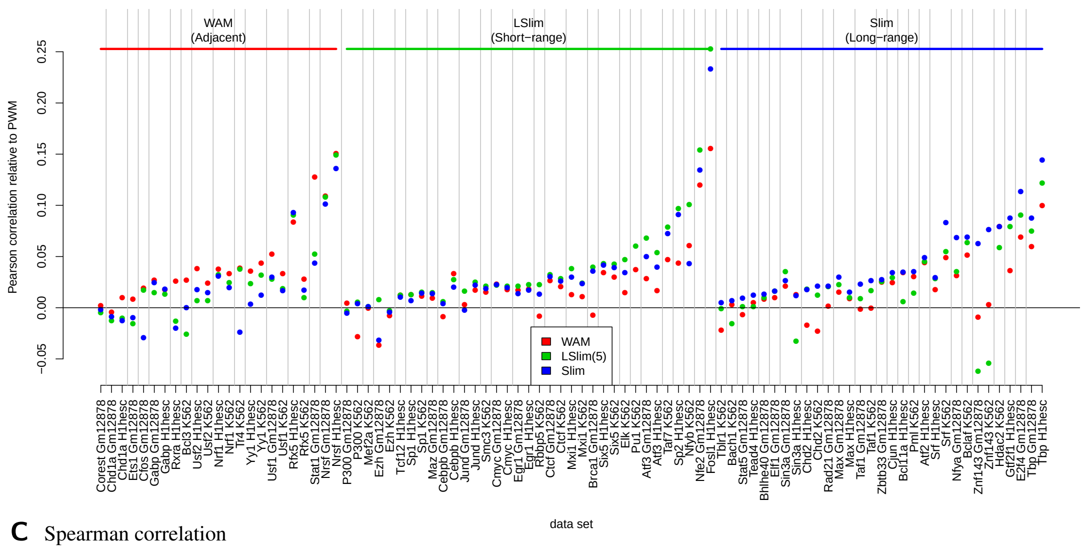


**Figure S14.** Comparison of models across cell types. We compare WAM, Slim, and LSlim models to the baseline PWM model, where each model has been trained on data for one cell type and predictions are made for another cell type.

**A** wAUC-PR



**B** Pearson correlation



**C** Spearman correlation

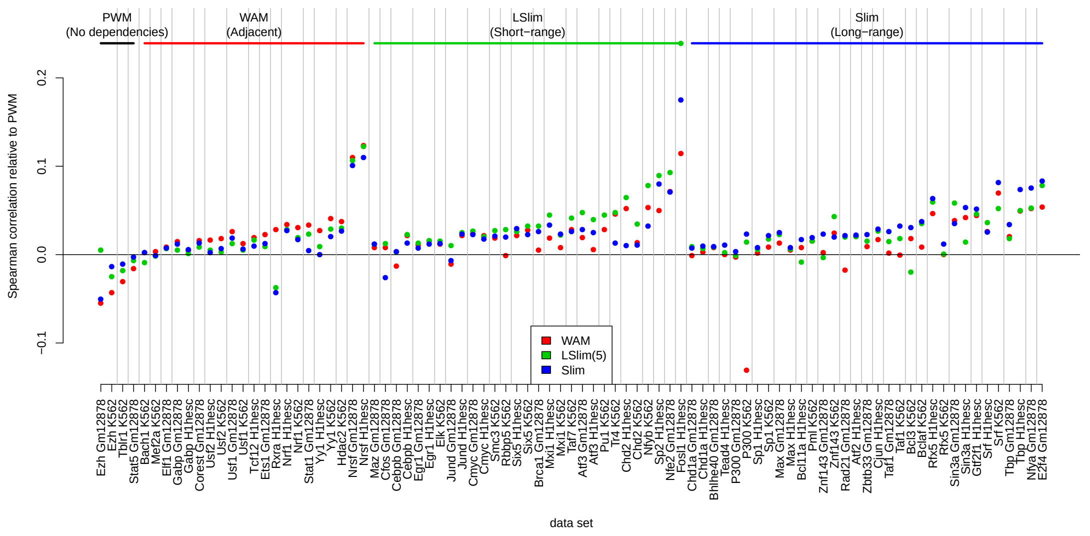
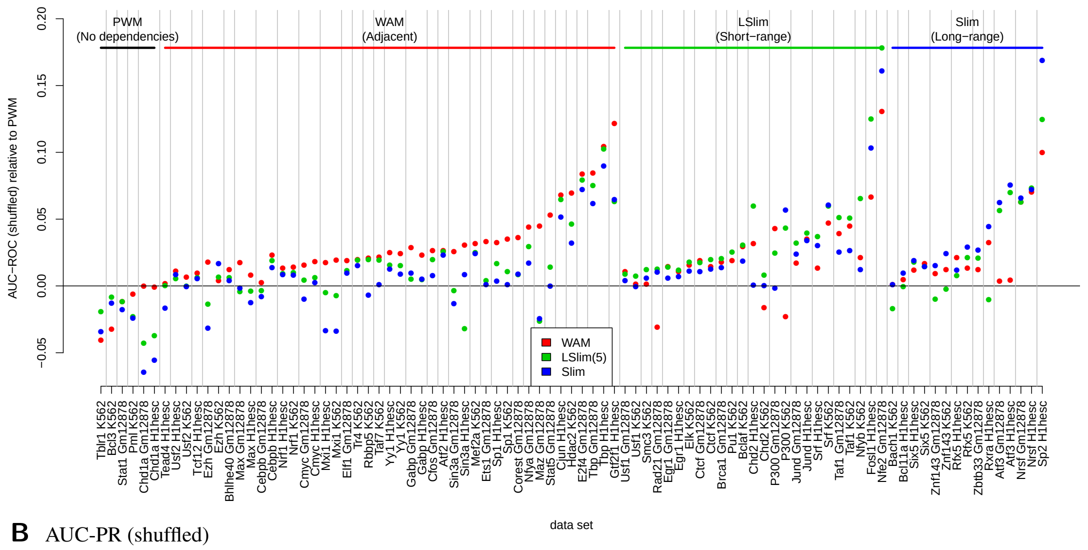


**Figure S15.** Comparison of models across cell types (2). We compare WAM, Slim, and LSlim models to the baseline PWM model, where each model has been trained on data for one cell type and predictions are made for another cell type.

**A** AUC-ROC (shuffled)



**B** AUC-PR (shuffled)

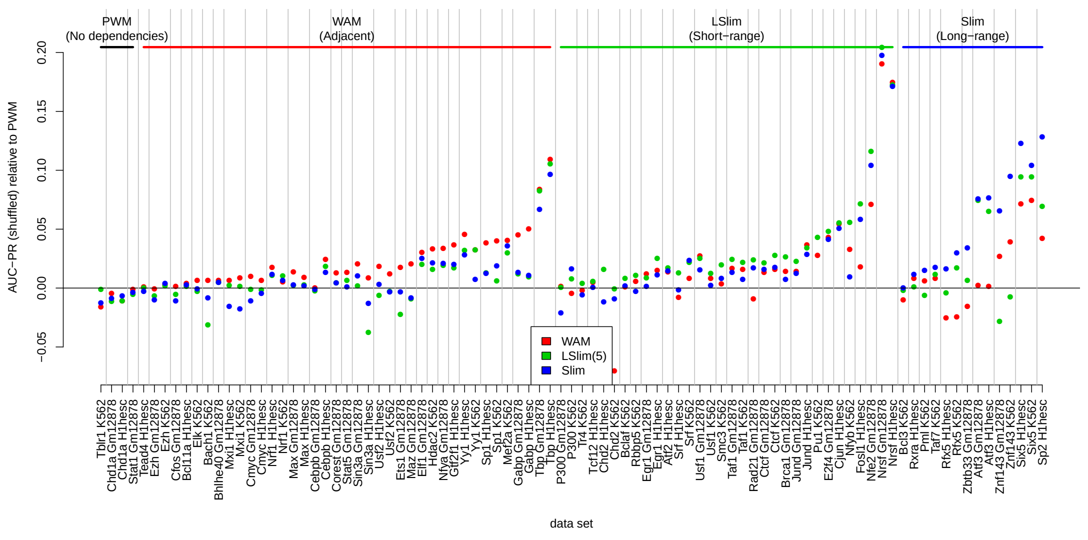
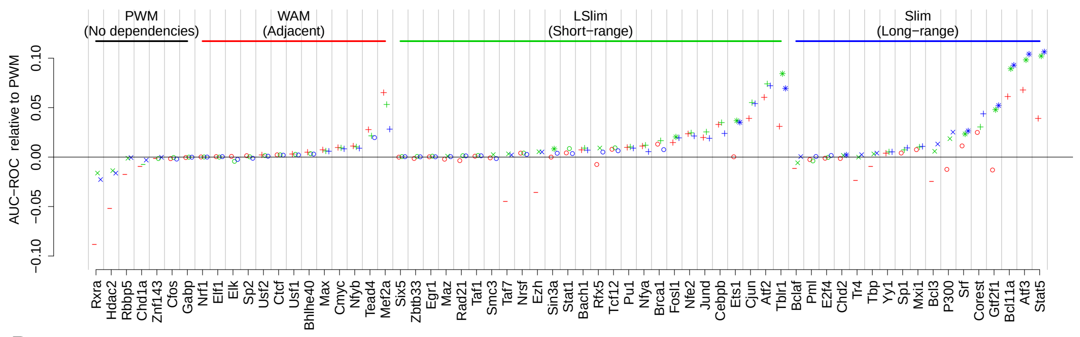
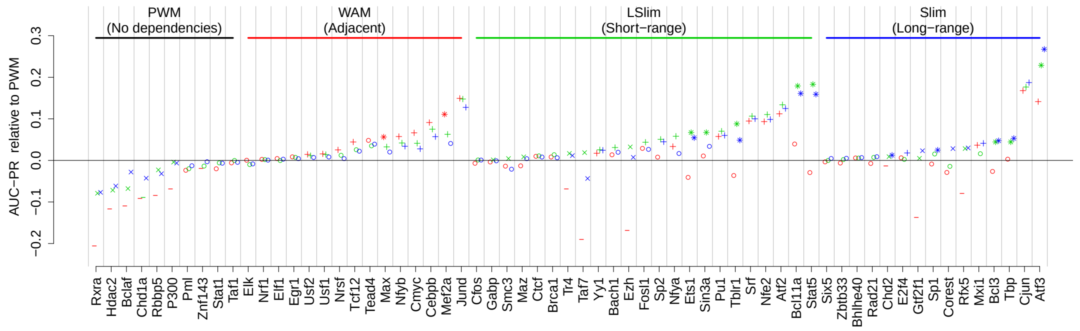


**Figure S16.** Comparison of models across cell types. We compare WAM, Slim, and LSlim models to the baseline PWM model, where each model has been trained on data for one cell type and predictions are made for another cell type.

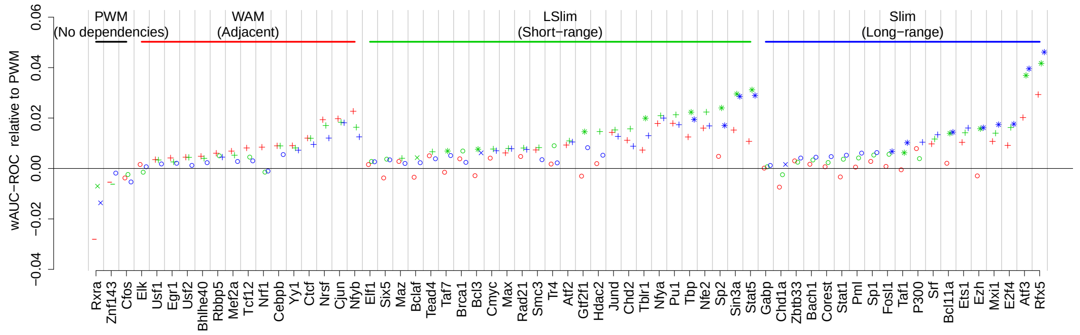
**A** AUC-ROC



**B** AUC-PR



**C** wAUC-ROC



**D** wAUC-PR

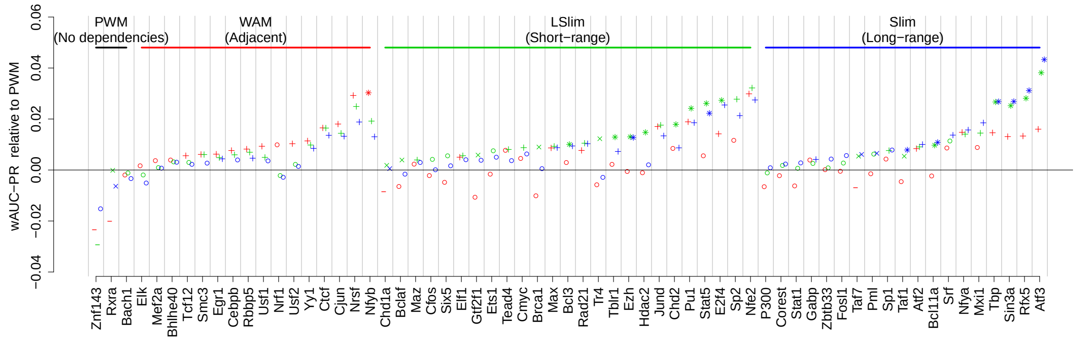
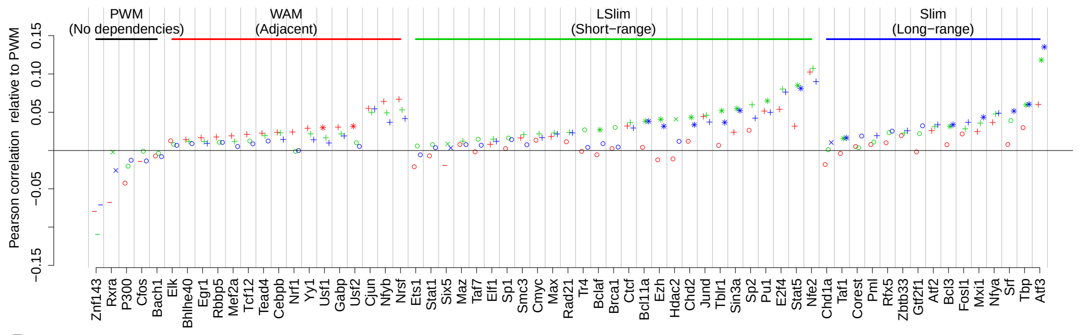
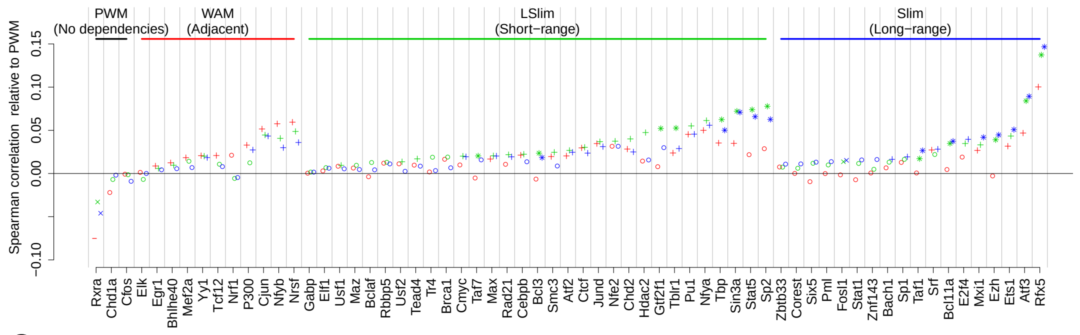


**Figure S17.** Comparison of models in a cross validation experiment. We compare WAM, Slim, and LSlim models to the baseline PWM model in a 10-fold cross validation experiment using ENCODE Chip-seq data sets for 63 transcription factors.

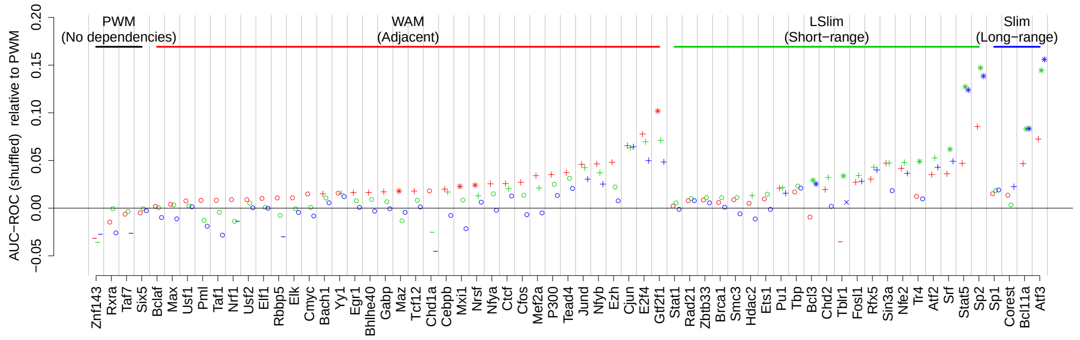
**A** Pearson correlation



**B** Spearman correlation



**C** AUC-ROC (shuffled)



**D** AUC-PR (shuffled)

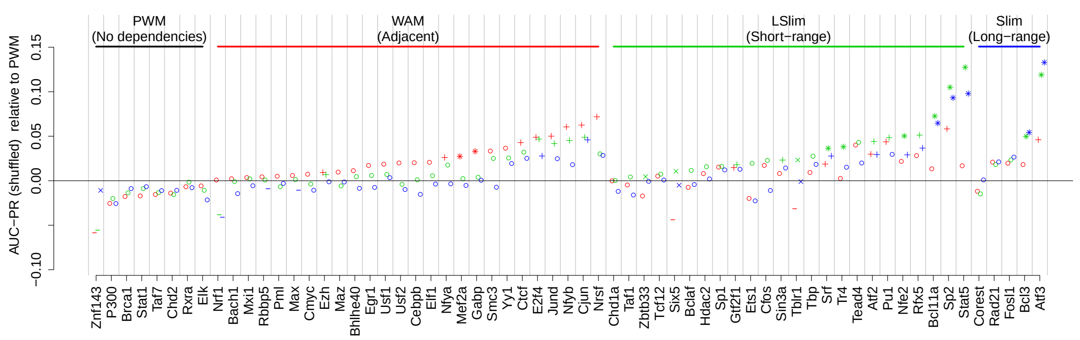
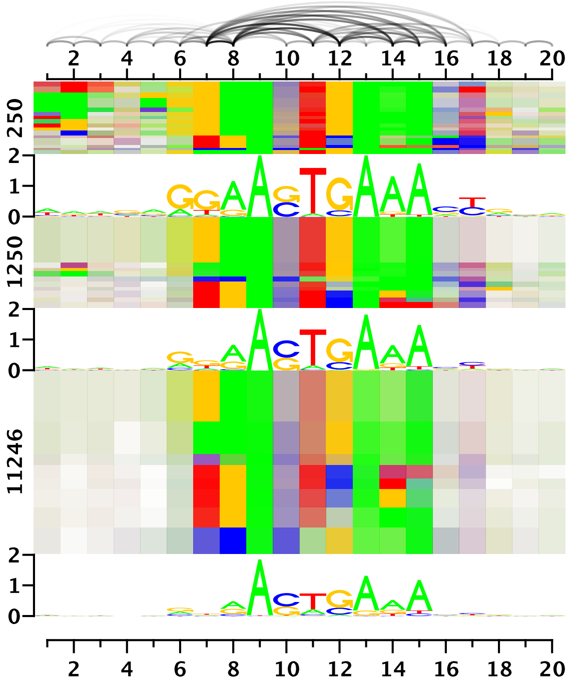


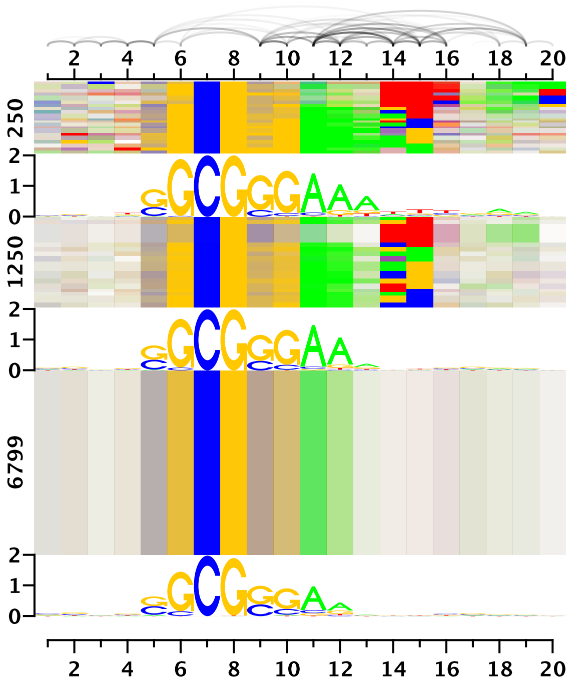
**Figure S18.** Comparison of models in a cross validation experiment (2). We compare WAM, Slim, and LSlim models to the baseline PWM model in a 10-fold cross validation experiment using ENCODE Chip-seq data sets for 63 transcription factors

## Text S6.7 Further examples of dependency logos

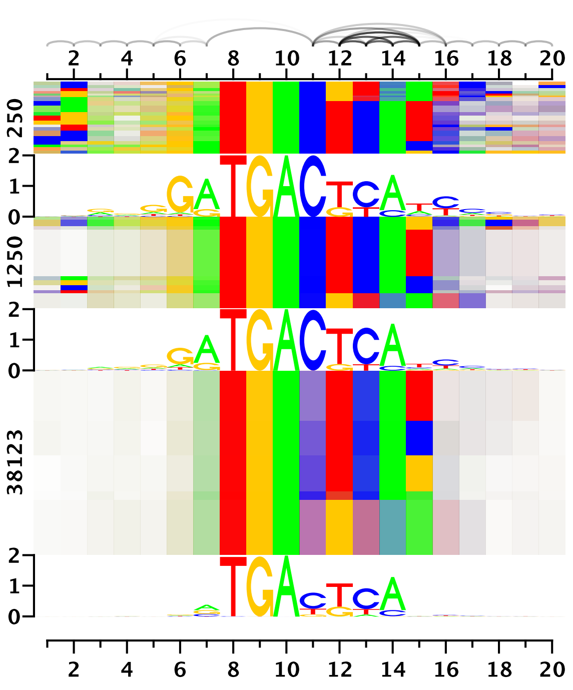
**A** Tblr1



**B** E2F4



**C** Jund



**D** Chd2

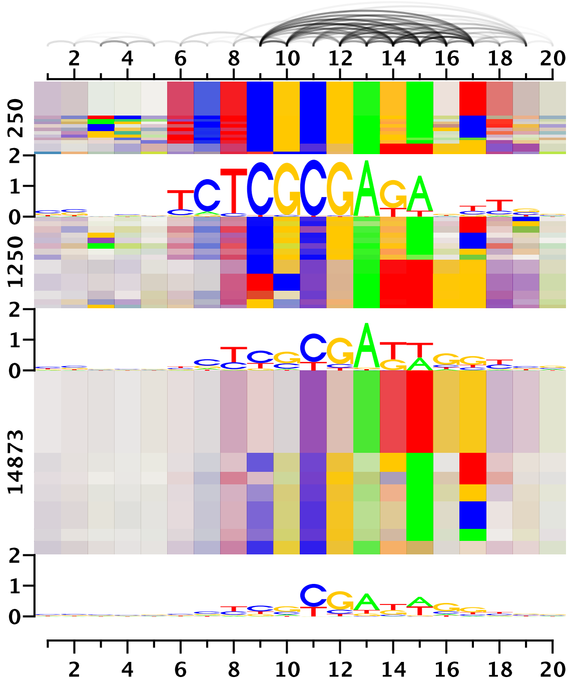
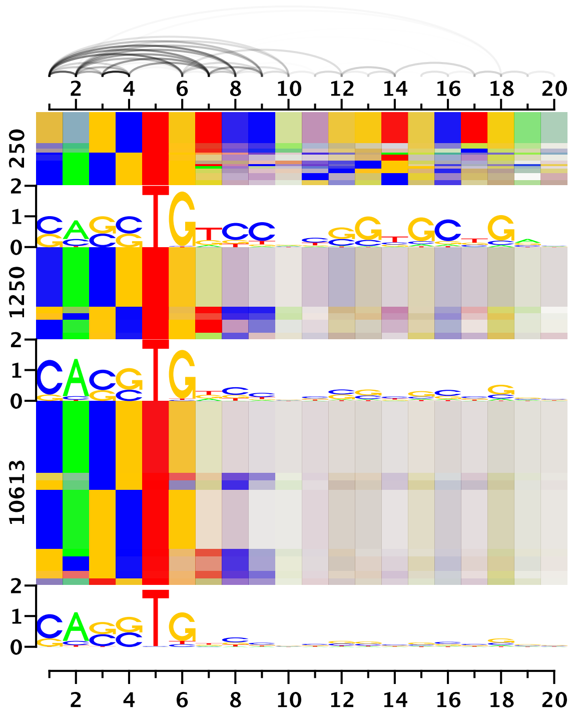
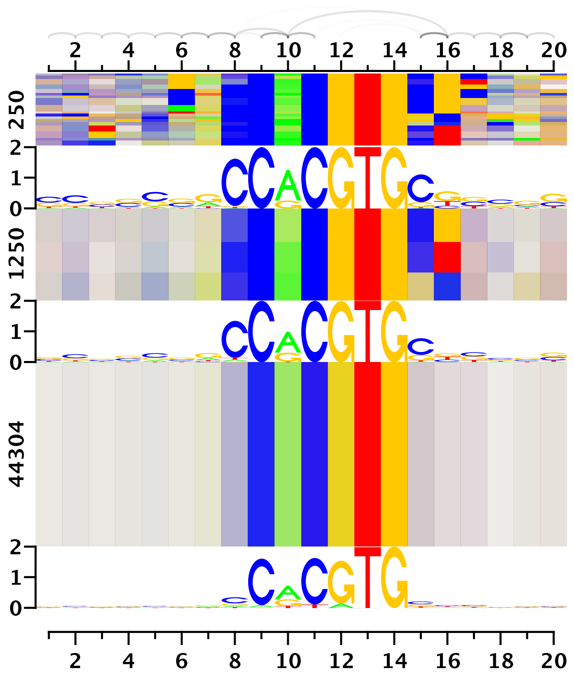


**Figure S19.** Dependency logos of binding sites predicted by the Slim model for different ChIP-seq data sets.

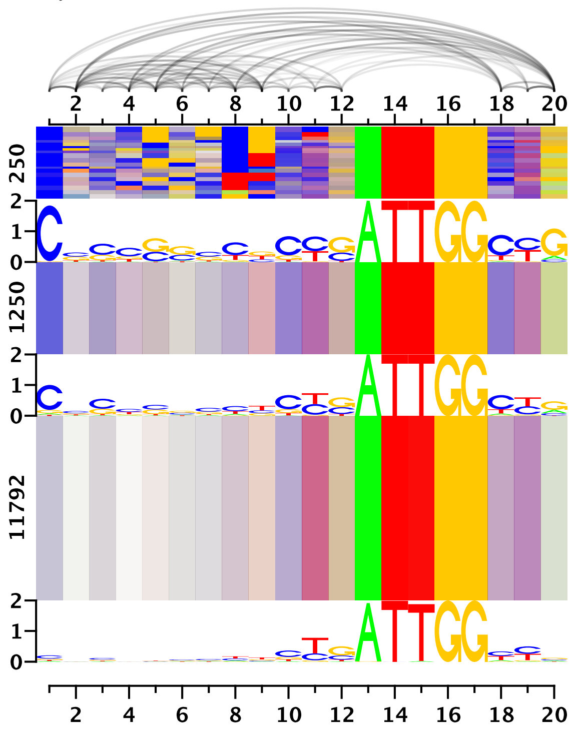
**A** Sin3a



**B** Max



**C** Nfyb



**D** Stat5

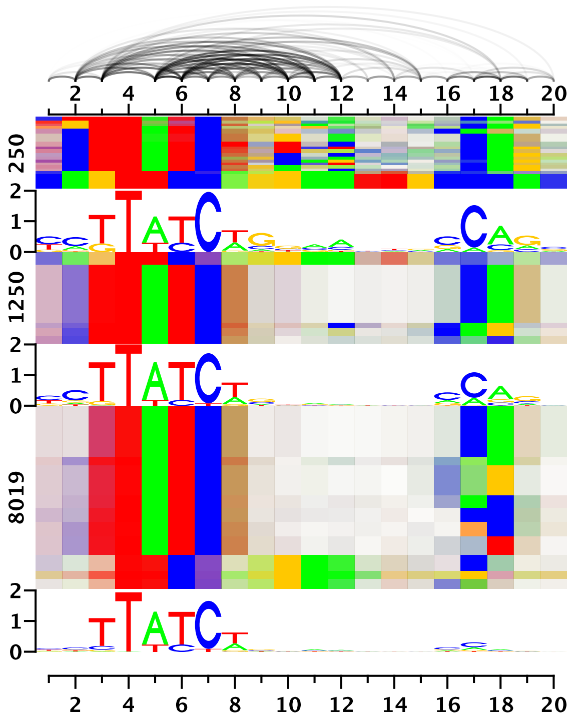


**Figure S20.** Dependency logos of binding sites predicted by the Slim model for different ChIP-seq data sets.

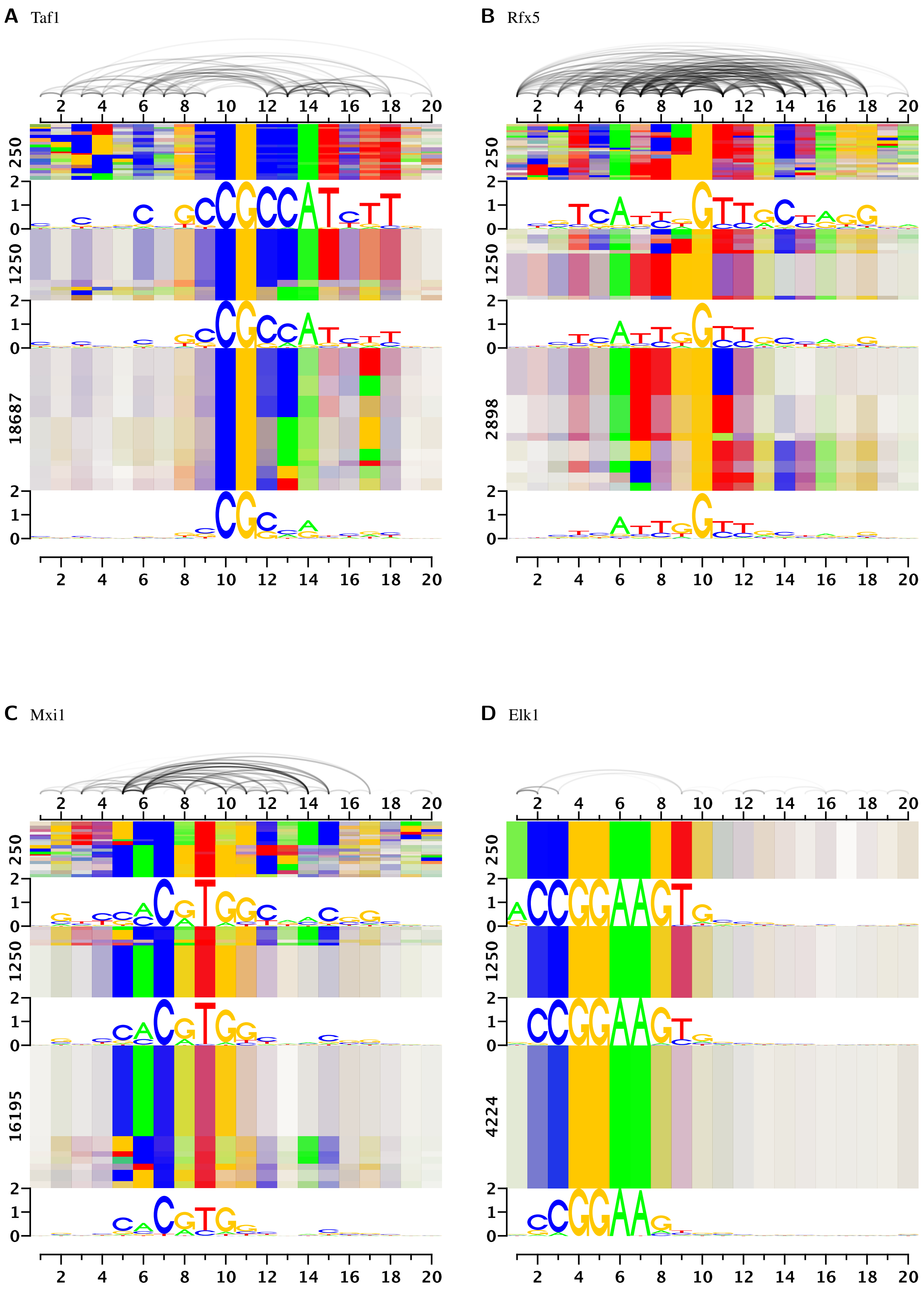
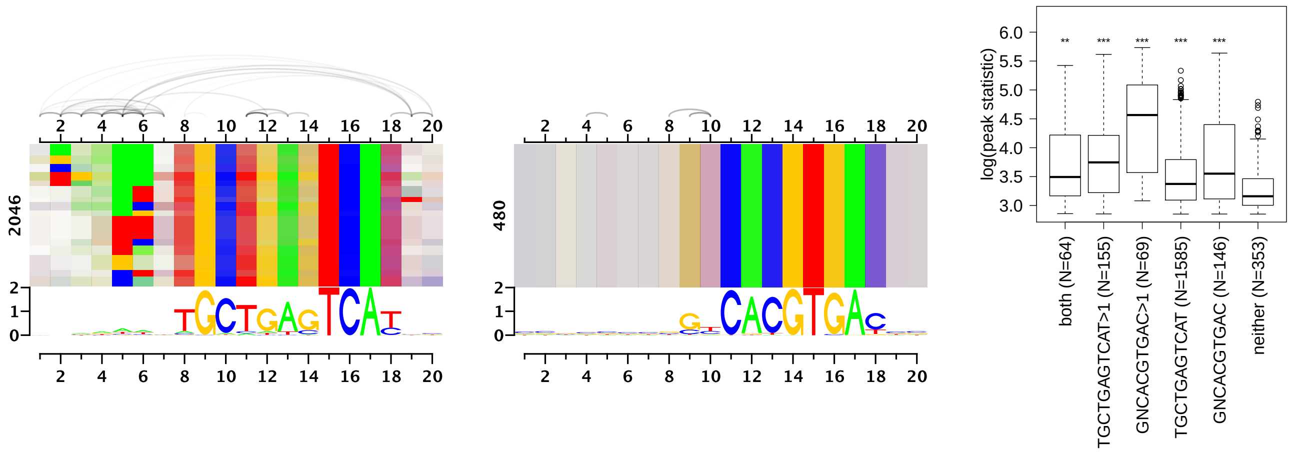
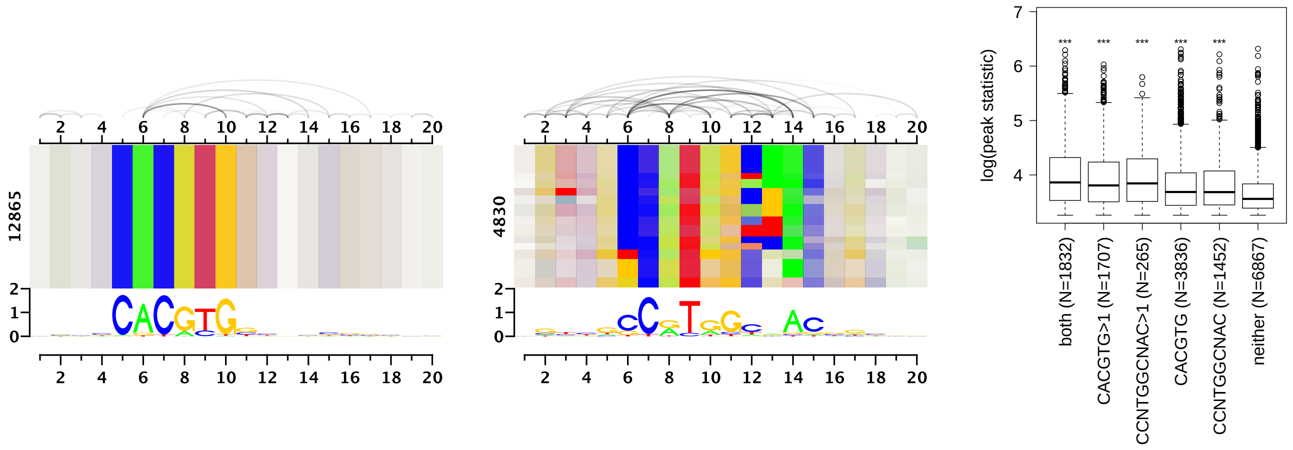


**Figure S21.** Dependency logos of binding sites predicted by the Slim model for different ChIP-seq data sets.

**A** Nfe2



**B** Mxi1



**C** Nrsf

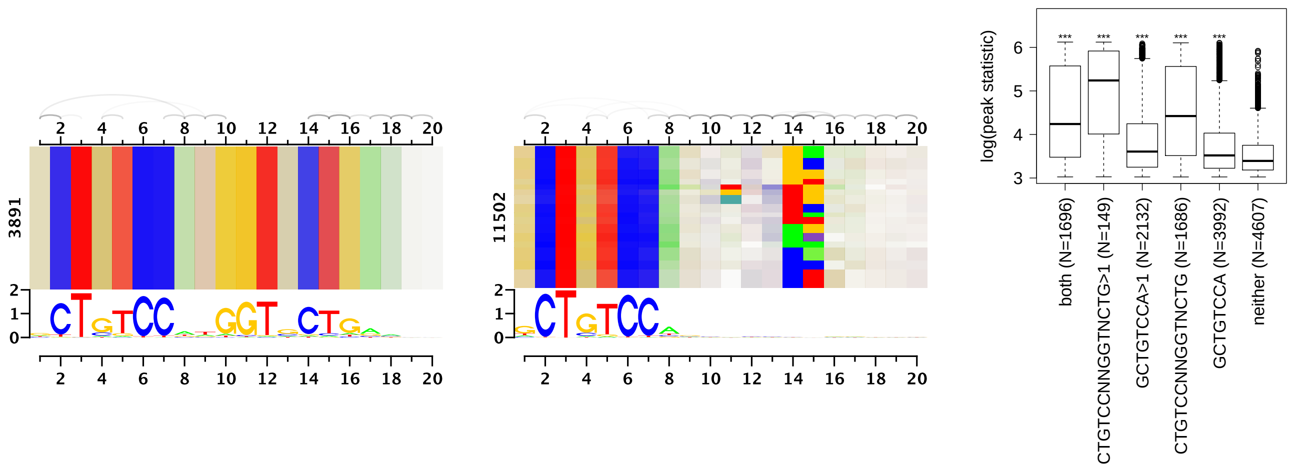


**Figure S22.** Dependency logos representing the two components of a mixture model and boxplots of the ChIP-seq peak statistics for the sequences containing the representative motifs of both components for A) Nfe2, B) Mxi1 and C) Nrsf. (Kolmogorov-Smirnov test, corrected p-values, ***: $p < 10^{-5}$; **: $p < 0.001$; *: $p < 0.01$).

## Text S6.8 Global picture of dependency structures

With the goal of obtaining a broader picture of the wealth of dependency structures present in the ENCODE ChIP-seq data sets, we aim at simple, numerical measures that describe neighboring and non-neighboring dependencies and heterogeneities in the predicted binding sites. For neighboring and non-neighboring dependencies, we consider for each position that position yielding the maximum mutual information and test this value for significance ($\alpha = 10^{-20}$, Chi-squared distribution, accounting for the size of data sets). If the mutual information is significant, we count this dependency as neighboring if the maximum is achieved by a position neighboring the current one and as non-neighboring otherwise. For assessing heterogeneity, we partition the predicted binding sites by the nucleotides at that position $j$ with the greatest $D(j)$ and compute the average, pairwise Kullback-Leibler divergence (85) between the PWMs learned on each of the partitions. We compute these measures for each of 10 cross-validation iterations on each of the ChIP-seq data sets to assess their technical variance due to different partitionings and different initializations of the algorithm (Table S3).

We find that the number of neighboring (Figure S23A) and non-neighboring dependencies(Figure S23B) and heterogeneity (Figures S23C) vary between different transcription factors to a greater extent than between the different replicates ($p \ll 10^{-10}$ for each measure, Kruskal-Wallis test). Among those factors yielding an exceptionally large number of neighboring dependencies are Nrsf, Ctcf, Mef2a, Cjun, Jund, and YY1, whereas we observe a large number of non-neighboring dependencies for other factors including Rfx5, Ets1, Ezh, and Bcl3. Finally, Sp2, Bcl11a, Rfx5, Cjun, Atf2 and Atf3, Bcl3, Tblr1, and Stat5 are among the factors with the greatest heterogeneities.

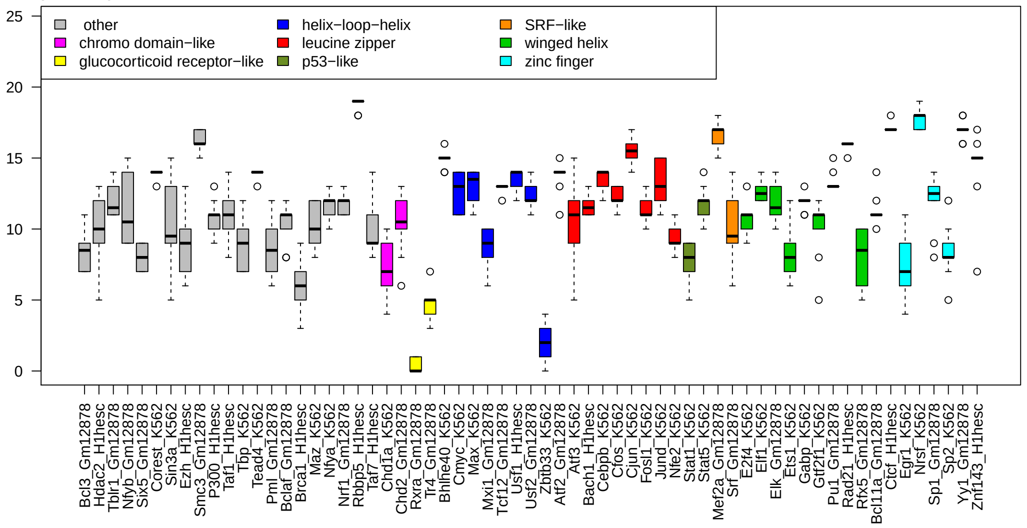
To investigate if we also find general tendencies for different families of transcription factors, we group transcription factors by their family, omitting groups with less than 3 members. Indeed, we find different tendencies for the three measures considered. Zinc finger transcription factors appear to exhibit a greater number of neighboring dependencies than the other families, which is consistent with the known dependence between the positions bound by a single finger but less between different fingers. Leucine zippers appear to show less non-neighboring dependencies than the other families, especially helix-loop-helix factors, which might be explained by independence of the two halves of the zipper and rather strict binding within each zipper. In contrast, leucine zippers show the greatest heterogeneity of all families, which might be due to the flexibility of the spacer between the two halves of the zipper as we observed for c-Jun. Helix-loop-helix factors show the least heterogeneity, which might be due to the clear pattern of the typical E-box motif.

Notably, none of the observed differences between transcription factor families is statistically significant (Kolmogorov-Smirnov test). Hence, we cannot select appropriate models or dependency structures captured by these models *a-priorily* considering a transcription factor's family alone. Fortunately, the Slim model proposed in this paper does not require such a pre-selection but adapts to the different dependency structures without user intervention. Dependency logos assist the user to dissect the detected dependency structure in an intuitive, visual way.

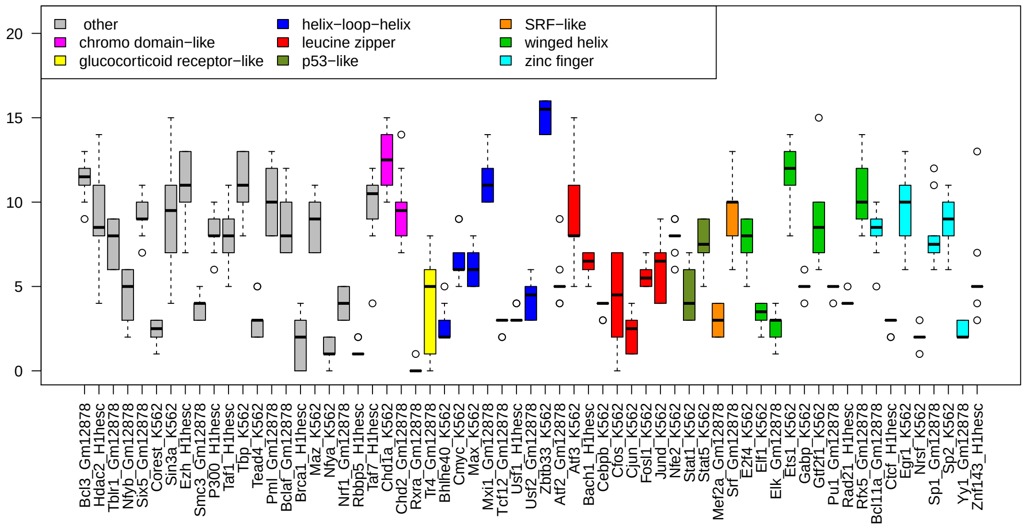
| data set | family | neighboring | non-neighboring | heterog. |
|---|---|---|---|---|
| Atf2 | leucine zipper | 14.0 | 4.0 | 0.533 |
| Atf3 | leucine zipper | 11.0 | 8.0 | 0.505 |
| Bach1 | leucine zipper | 11.5 | 2.0 | 0.286 |
| Bcl11a | zinc finger | 11.0 | 5.5 | 0.564 |
| Bcl3 | ankyrin repeat | 8.5 | 10.5 | 0.601 |
| Bclaf | unknown | 11.0 | 2.5 | 0.304 |
| Bhlhe40 | helix-loop-helix | 15.0 | 1.0 | 0.213 |
| Brca1 | unknown | 6.0 | 0.0 | 0.276 |
| Cebpb | leucine zipper | 14.0 | 3.0 | 0.274 |
| Cfos | leucine zipper | 12.0 | 1.5 | 0.251 |
| Chd1a | chromo domain-like | 7.0 | 10.0 | 0.309 |
| Chd2 | chromo domain-like | 10.5 | 8.5 | 0.424 |
| Cjun | leucine zipper | 15.5 | 1.0 | 0.547 |
| Cmyc | helix-loop-helix | 13.0 | 4.0 | 0.289 |
| Corest | indirect | 14.0 | 2.0 | 0.248 |
| Ctcf | zinc finger | 17.0 | 2.5 | 0.285 |
| E2f4 | winged helix | 11.0 | 4.5 | 0.334 |
| Egr1 | zinc finger | 7.0 | 9.5 | 0.292 |
| Elf1 | winged helix | 12.5 | 3.0 | 0.242 |
| Elk | winged helix | 11.5 | 0.0 | 0.289 |
| Ets1 | winged helix | 8.0 | 9.0 | 0.160 |
| Ezh | polycomb | 9.0 | 7.0 | 0.333 |
| Fosl1 | leucine zipper | 11.0 | 3.0 | 0.269 |
| Gabp | winged helix | 12.0 | 3.0 | 0.276 |
| Gtf2f1 | winged helix | 11.0 | 2.5 | 0.283 |
| Hdac2 | arginase/deacetylase | 10.0 | 7.0 | 0.341 |
| Jund | leucine zipper | 13.0 | 4.0 | 0.372 |
| Max | helix-loop-helix | 13.5 | 5.0 | 0.283 |
| Maz | unknown | 10.0 | 7.5 | 0.213 |
| Mef2a | SRF-like | 17.0 | 1.5 | 0.295 |
| Mxi1 | helix-loop-helix | 9.0 | 9.0 | 0.301 |
| Nfe2 | leucine zipper | 9.0 | 5.0 | 0.418 |
| Nfya | unknown | 12.0 | 0.0 | 0.270 |
| Nfyb | histone-fold | 10.5 | 5.0 | 0.264 |
| Nrf1 | unknown | 12.0 | 0.0 | 0.265 |
| Nrsf | zinc finger | 18.0 | 2.0 | 0.143 |
| P300 | TAZ | 11.0 | 6.0 | 0.289 |
| Pml | TRIM | 8.5 | 9.5 | 0.337 |
| Pu1 | winged helix | 13.0 | 4.0 | 0.155 |
| Rad21 | winged helix | 16.0 | 4.0 | 0.281 |
| Rbbp5 | unknown | 19.0 | 1.0 | 0.394 |
| Rfx5 | winged helix | 8.5 | 7.0 | 0.562 |
| Rxra | glucocorticoid receptor-like | 0.0 | 0.0 | 0.271 |
| Sin3a | PAH2 | 9.5 | 5.5 | 0.441 |
| Six5 | homeodomain-like | 8.0 | 2.0 | 0.265 |
| Smc3 | smc hinge | 16.0 | 2.0 | 0.246 |
| Sp1 | zinc finger | 12.5 | 6.0 | 0.261 |
| Sp2 | zinc finger | 8.0 | 5.0 | 0.436 |
| Srf | SRF-like | 9.5 | 7.5 | 0.253 |
| Stat1 | p53-like | 8.0 | 0.0 | 0.314 |
| Stat5 | p53-like | 12.0 | 6.0 | 0.423 |
| Taf1 | TBP-binding fragment | 11.0 | 6.5 | 0.174 |
| Taf7 | unknown | 9.0 | 5.0 | 0.303 |
| Tblr1 | F box-like | 11.5 | 7.0 | 0.474 |
| Tbp | TBP-like | 9.0 | 10.0 | 0.355 |
| Tcf12 | helix-loop-helix | 13.0 | 1.0 | 0.322 |
| Tead4 | TEA/ATTS | 14.0 | 0.0 | 0.293 |
| Tr4 | glucocorticoid receptor-like | 5.0 | 0.0 | 0.291 |
| Usf1 | helix-loop-helix | 14.0 | 3.0 | 0.281 |
| Usf2 | helix-loop-helix | 12.0 | 1.0 | 0.226 |
| Yy1 | zinc finger | 17.0 | 1.0 | 0.154 |
| Zbtb33 | helix-loop-helix | 2.0 | 3.0 | 0.297 |
| Znf143 | zinc finger | 15.0 | 3.0 | 0.256 |

**Table S3.** For each ChIP-seq data set of human transcription factors from ENCODE, we list transcription factor, its family, and median values of the dependency statistics for neighboring and non-neighboring dependencies and heterogeneities.

**A** Neighboring



**B** Non-neighboring



**C** Heterogeneity

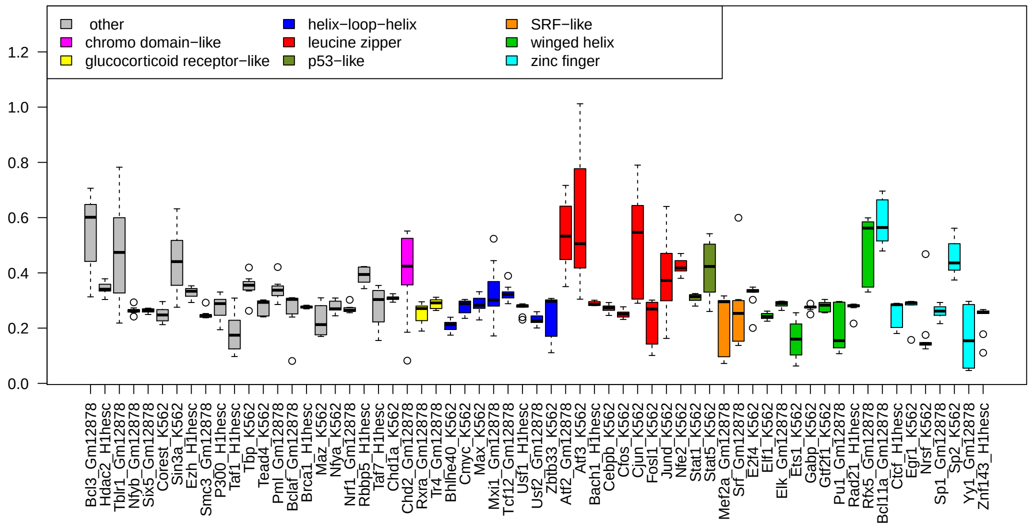


**Figure S23.** Dependency structure is highly factor-dependent. For each of the ChIP-seq data sets, we show a box plot of the number of significant non-neighboring dependencies (A) and heterogeneity measures (average Kullback-Leibler divergence, B) over the 10 cross validation runs. Boxes are colored according to transcription factor families.