Prospectus: Removing Nonlinear Batch Effects from Biological Data

Jonathan Dayton

April 29, 2018

Abstract

Many biological studies, particularly studies involving the genetics of human disease, could discover items of greater statistical and biological significance if performed with larger data sets. Many data sets are publicly available and could be combined in order to increase these studies' significance, but even when the same type of data are collected by multiple studies, minor differences in collection methodologies and environments can lead to confounding effects in the data which can in turn cause incorrect results. In order to do so, we would like to apply deep learning to remove these effects and combine molecular biological data sets. We will create a neural network with a dual loss function in order to remove confounding effects from the data while minimizing the amount of change to the input data. This model will output data the same shape and size as the input data but with confounding effects removed. We plan to test the model on artificially constructed data and commonly used real biological data sets and compare the output with the output from other common batch adjustment algorithms. We will also apply the model to combine microarray and RNA-Seq data sets and to remove the tissue signal from pan-cancer data. We will publish our software to enable other scientists to remove confounding effects from their own data using our methodologies.

1 Introduction

When working with biological data, problems can arise from combining data from multiple sources. For example, if two labs quantify the same 300 RNA samples for lung cancer patients but use different equipment to do so, each lab will come up with slightly different measurements. If a third party were to then study the patients using data from both labs, the differences in measurements could alter their analysis. These systemic effects are known as "batch effects" and are understood to have a nontrivial impact on high-throughput biological data [9]. In one study, researchers found that, contrary to previous knowledge, molecular data from mice and humans clustered more closely by species than by tissue type [21]. Referees showed in a rebuttal that when accounting for batch effects, these data actually clustered more closely by tissue type, as expected [6]. This type of bias can be even more pronounced when using data collected from different tissue types and can skew results, such as in pan-cancer analyses [4].

Several methods exist for removing batch effects from biological datasets. Two commonly used methods are ComBat [8] and SVA [10]. ComBat uses an empirical Bayes method to estimate batch effect parameters and then uses linear regression to remove the effects, and SVA uses singular value decomposition to identify and remove batch effects. Since both of these methods use linear adjustments to remove batch effects, they do not account for nonlinear effects. However, recent advances in neural networks have introduced new ways to account for higher-order, nonlinear relationships in data.

Artificial neural networks are a machine learning tool inspired by the way human brains function; input values pass through layers of linear and nonlinear functions, the final output values are measured against objectives, and the layers of functions are adjusted to bring the outputs closer to the objectives. This process is repeated until the outputs are sufficiently close to their targets [16]. Autoencoders are a type of

neural network that encode and then reconstruct their input, and their traditional objective function is to construct the output as similarly to the input as possible [7]. Neural networks have historically decreased in effectiveness when working with data from multiple domains [5], in part because they may learn based on confounding effects (e.g. which researcher collected the data) instead of learning based on biologically interesting causal effects (e.g. which gene is consistently upregulated in a disease) [11]. Recently, researchers have experimented with discouraging the networks from distinguishing between classes by giving them two objective functions: 1. to learn as much as possible about the input data and 2. to forget any patterns that help distinguish between classes [5, 18]. This type of dual objective function has been used in conjunction with autoencoders [12]; the autoencoder is "rewarded" for reconstructing the input faithfully, but "punished" for keeping class-specific patterns. Research has also shown that neural networks are effective in working with biological data; for example, neural networks have been applied to extract biologically relevant latent spaces in RNA-Seq data with as few as 10,000 samples [19], and some researchers have used that latent space to generate realistic synthetic biomedical data for other scientific studies [3].

Our approach will build on this previous research by creating an adversarial autoencoder that removes confounding effects from multi-origin data. This autoencoder will be packaged and made available to allow other researchers to remove these effects from their own data to enable further analysis.

2 Aims

2.1 Adversarial Network

We will create an adversarial autoencoder that removes batch effects from two input datasets. Our network will accept two sets of floating point vectors (such as RNA-Seq data) as inputs. Each set will represent a batch. We will create an autoencoder that tries to reproduce the input as faithfully as possible while also removing distinct differences between batches. This will be made possible by a neural network classifier that learns to discriminate between the two batches after they have been fed through the autoencoder. Over the course of training, we anticipate that the discriminator will get better at finding higher-order, nonlinear differences between the batches. As this happens, the autoencoder's loss will start to increase, and the autoencoder will make adjustments to account for this. Eventually, the autoencoder will learn to completely remove the batch effects and the discriminator's loss will match what would be expected if it were making random guesses. We have already created a preliminary network with this structure and loss function (see "Preliminary Results").

We may want to train the discriminator prior to training the autoencoder to ensure that the discriminator is learning real patterns and then being fooled by the autoencoder. We may also determine that we need a more complex loss function and may model our loss after the Wasserstein GAN [2] or after Harman, another SVD-based batch correction method [14].

2.2 Test on MNIST Data with Synthetic Batch Effects

We will generate 3-5 types of synthetic "batch effects" to apply to the MNIST dataset. Then we will apply our adversarial network along with the ComBat and SVA methods for batch correction and compare the three methods. We'll display the images and use PCA charts to see how well the batch effects are removed and then train a new classifier on the output data in order to determine whether a batch effect remains after adjustment with each of the algorithms.

The success of the model will be assessed based on two metrics: mean squared error (how well is the original data recovered?) and classifier accuracy (how well is the effect removed?). These two metrics will have somewhat of an inverse relationship: as confounding effects are removed, the network's output will begin to differ from the input. We anticipate that for some datasets with particularly strong confounding effects, classifier accuracy will need to be prioritized over mean squared error. Therefore, we will explore

the question of balancing the network between these two metrics and will add parameters in the network for tuning this balance. These metrics will be collected and compared for each of our 3-5 types of synthetic batch effect and each batch correction method.

MNIST will be used to test our methods for two reasons. First, each instance comes in a one-dimensional array of floats, just like RNA-Seq data. Second, the effectiveness of batch adjustment can be easily visualized since the data can be viewed as images. Although the values in an MNIST vector have a spatial relationship with each other, RNA-Seq vector values do not; thus we will avoid using image-specific methods like convolutional layers in our network.

We have already seen some success in removing artificial batch effects from MNIST data (see "Preliminary Results").

2.3 Test on RNA-Seq Data with Synthetic Batch Effects

We will repeat the tests that we performed on the MNIST dataset on an RNA-Seq data set, including splitting the data set in half, applying the same synthetic effects, and collecting the same summary statistics. We plan to use pan-cancer data from The Cancer Genome Atlas (TCGA), which includes roughly 10,000 instances across several cancer types [20].

2.4 Test on Data with Real Batch Effects

Once we have shown that our methods are effective in removing synthetic batch effects, we will apply the methods to data with natural batch effects. We will test our methods on the mouse-human data set referenced in [21] and analyze how the data cluster after adjustment using the same clustering methods they used.

We will also apply the model to combine microarray and RNA-Seq data sets. We'll compare our results here with TDM, a model that transforms RNA-Seq data into the same distribution as microarray data [17].

Additionally, we will explore the effectiveness of removing the cancer type effect from the TCGA data in order to make pan-cancer predictions [4].

2.5 Explore Limitations

We will dedicate research time and efforts to exploring the limitations of our software. We will ask questions such as the following: How many instances do we need in each class before MSE gets too high, and could we effectively use transfer learning to mitigate this issue [15]? What problems do we see when we have class imbalance [13]? How can we remove one type of effect without removing another type of effect?

We will explore the implications of these questions using the test datasets previously mentioned (e.g. with MNIST, if one batch has more 0s and the other has more 3s, do the outputs start to look like a combination of 0s and 3s?).

2.6 Make code available

We will document and publish our software on GitHub to enable other scientists to remove confounding effects from their own data using our methodologies.

3 Timeline

The completion of these aims will be an iterative process, and the knowledge gained in later steps may require revisiting earlier steps. However, the following represents generally when we expect to work on and complete our aims.

Apr–Jul 2018	Make a working adversarial network
Aug 2018	Test on MNIST data
Sep 2018	Repeat w/ RNA-Seq data
Oct–Nov 2018	Study the model's effectiveness on data with real confounders
Dec–Feb 2019	Validate and explore limitations
Feb–Apr 2019	Publish results and make code available

4 Preliminary Results

4.1 Network

We implemented the described neural network in TensorFlow [1]. The autoencoder's encoding subnetwork has three fully connected layers, each of which is followed by a ReLU activation. The first two layers are followed by batch normalization. The decoding subnetwork has the same structure. The discriminator has six fully connected layer, and all are followed by ReLU activations except the last, which is followed by a sigmoid function (see Figures 1 and 2).

The discriminator's loss function is the mean squared error between the output layer and one-hot encoded targets. The autoencoder's loss function is $MSE+(1-loss_d)$, where MSE is the mean squared error between inputs and reconstructed outputs and where $loss_d$ represents the discriminator's loss. Both networks were optimized using the Adam Optimizer with default parameters.

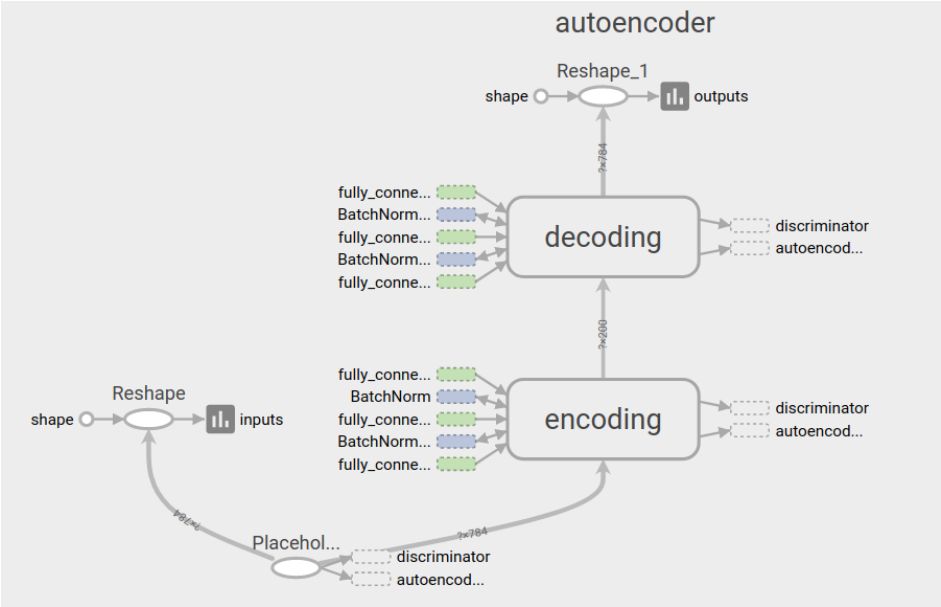

Figure 1: Computation graph diagram of the discriminator.

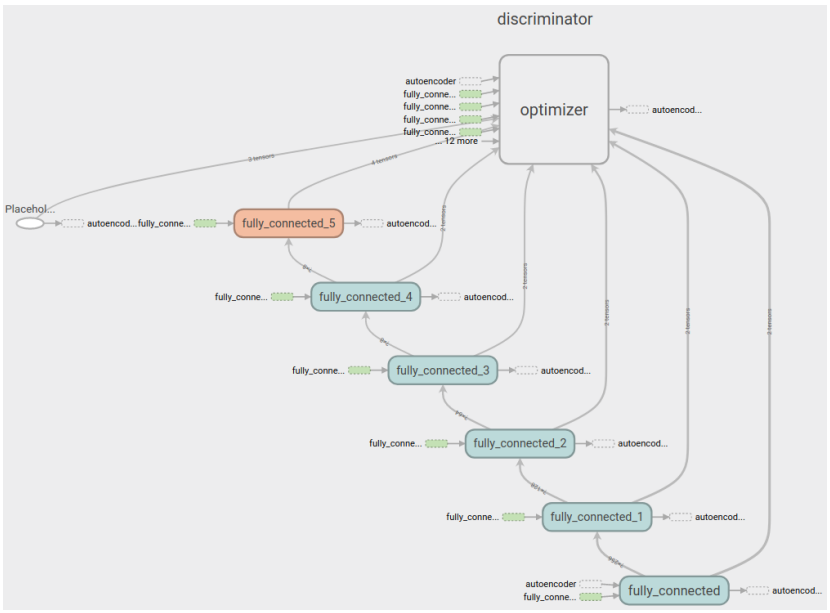

Figure 2: Computation graph diagram of the discriminator.

4.2 MNIST

We loaded the MNIST dataset and split it into two random, equal-sized batches. We randomly generated noise the same shape and size as the input data and normally distributed about zero. In order to simulate batch effects, this noise was applied to one half of the data in one of two ways. The first application was additive noise: we scaled the noise by a factor of 0.1 and then added it to the inputs. The second application was multiplicative noise: we took the entrywise product of the noise and the inputs. In both cases, we normalized values to $[0, 1]$ after applying noise. The noised and non-noised inputs were then input as separate classes into the network in minibatches of size 100.

Note: neither the additive nor the multiplicative adjustment causes a nonlinear or higher-order adjustment. In future iterations, care will be taken to ensure that nonlinear effects are thoroughly simulated and tested.

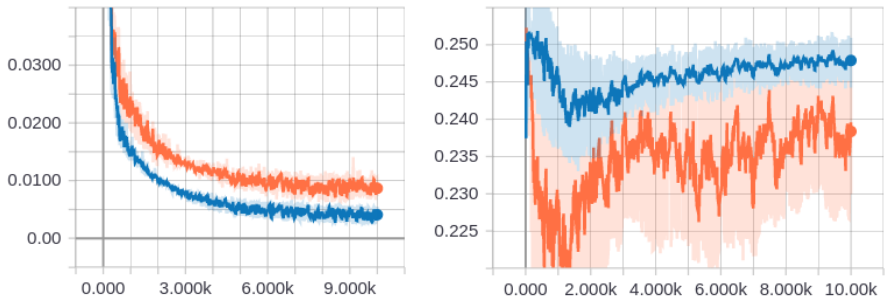

Figure 3: Loss over 10,000 training steps with mini-batch size 100 for (left) the autoencoder and (right) the discriminator when applied to the noised MNIST data (see Figure 4 and Section 2.2). In both cases, orange represents multiplicative noise and blue represents additive noise.

We saw varying degrees of success with both noise applications, but the autoencoder achieved lower average losses on the additive data than on the multiplicative data, whereas the discriminator had lower losses on the multiplicative data than with the additive (see Figure 3. In both cases, the autoencoder's loss

decreased overtime while the discriminator's loss initially decreased (presumably when learning differences between "batches") and then increased (when the autoencoder learned to "fool" it). We interpret this to mean that the autoencoder had an easier time removing the additive batch effect than the multiplicative effect, which seems to be verified by visual inspection of the outputs (see Figure 4). One explanation for this is that since multiplicative noise altered the input far more than did the additive noise, the autoencoder was more hesitant to remove the effects, which would therefore make the outputs even more different from the inputs and increase the MSE loss term. Future addition of parameters by which to scale loss terms may increase or decrease this hesitancy.

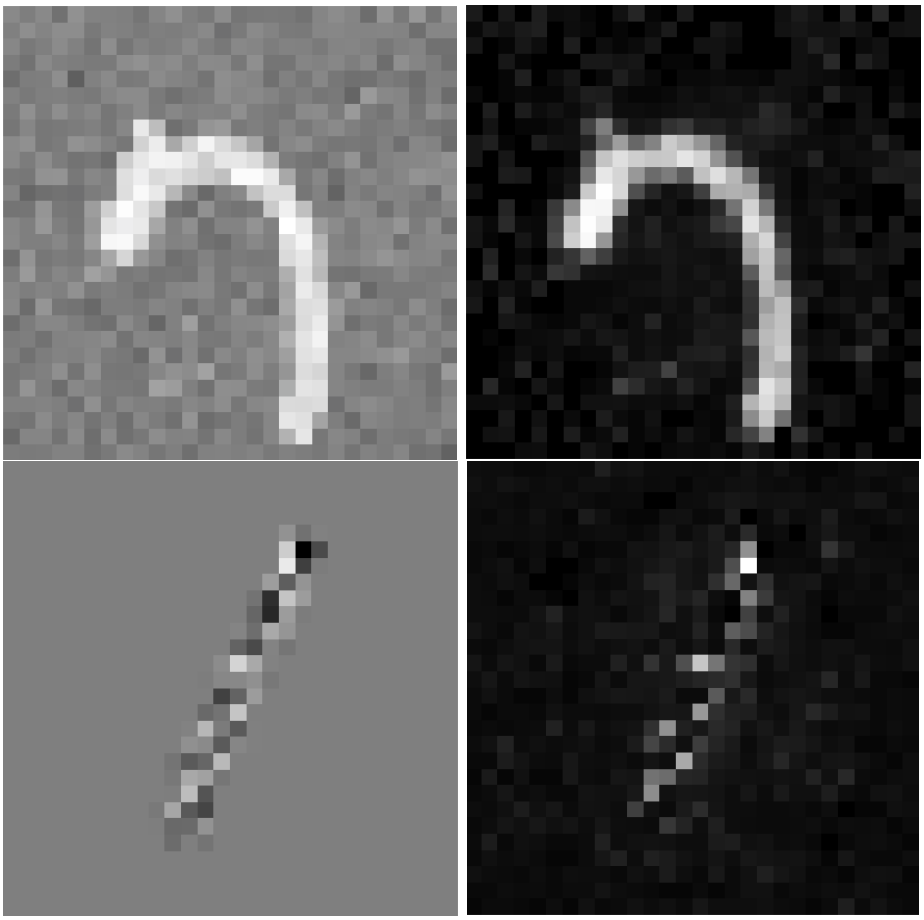

Figure 4: MNIST digits before and after going through the adversarial autoencoder for batch adjustment. Top: additive noise. Bottom: multiplicative noise.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv:1603.04467 [cs]*, March 2016. arXiv: 1603.04467.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *arXiv:1701.07875 [cs, stat]*, January 2017. arXiv: 1701.07875.

[3] Brett K. Beaulieu-Jones, Zhiwei Steven Wu, Chris Williams, James Brian Byrd, and Casey S. Greene. Privacy-preserving generative deep neural networks support clinical data sharing. *bioRxiv*, page 159756, November 2017.

[4] Jonathan B. Dayton and Stephen R. Piccolo. Classifying cancer genome aberrations by their mutually exclusive effects on transcription. *bioRxiv*, page 122549, September 2017.

[5] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-Adversarial Training of Neural Networks. *arXiv:1505.07818 [cs, stat]*, May 2015. arXiv: 1505.07818.

[6] Yoav Gilad and Orna Mizrahi-Man. A reanalysis of mouse ENCODE comparative gene expression data. *F1000Research*, 4, May 2015.

[7] G. E. Hinton and R. R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, July 2006.

[8] W. Evan Johnson, Cheng Li, and Ariel Rabinovic. Adjusting batch effects in microarray expression data using empirical Bayes methods. *Biostatistics (Oxford, England)*, 8(1):118–127, January 2007.

[9] Jeffrey T. Leek, Robert B. Scharpf, Héctor Corrada Bravo, David Simcha, Benjamin Langmead, W. Evan Johnson, Donald Geman, Keith Baggerly, and Rafael A. Irizarry. Tackling the widespread and critical impact of batch effects in high-throughput data. *Nature Reviews. Genetics*, 11(10):733–739, 2010.

[10] Jeffrey T. Leek and John D. Storey. Capturing Heterogeneity in Gene Expression Studies by Surrogate Variable Analysis. *PLOS Genetics*, 3(9):e161, September 2007.

[11] Christos Louizos, Uri Shalit, Joris Mooij, David Sontag, Richard Zemel, and Max Welling. Causal Effect Inference with Deep Latent-Variable Models. *arXiv:1705.08821 [cs, stat]*, May 2017. arXiv: 1705.08821.

[12] Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard Zemel. The Variational Fair Autoencoder. *arXiv:1511.00830 [cs, stat]*, November 2015. arXiv: 1511.00830.

[13] Vegard Nygaard, Einar Andreas Rødland, and Eivind Hovig. Methods that remove batch effects while retaining group differences may lead to exaggerated confidence in downstream analyses. *Biostatistics (Oxford, England)*, 17(1):29–39, January 2016.

[14] Yalchin Oytam, Fariborz Sobhanmanesh, Konsta Duesing, Joshua C. Bowden, Megan Osmond-McLeod, and Jason Ross. Risk-conscious correction of batch effects: maximising information extraction from high-throughput genomic datasets. *BMC Bioinformatics*, 17:332, September 2016.

[15] Hilary S. Parker, Héctor Corrada Bravo, and Jeffrey T. Leek. Removing batch effects for prediction problems with frozen surrogate variable analysis. *PeerJ*, 2:e561, September 2014.

[16] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, January 2015.

[17] Jeffrey A. Thompson, Jie Tan, and Casey S. Greene. Cross-platform normalization of microarray and RNA-seq data for machine learning applications. *PeerJ*, 4:e1621, January 2016.

[18] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep Domain Confusion: Maximizing for Domain Invariance. *arXiv:1412.3474 [cs]*, December 2014. arXiv: 1412.3474.

[19] Gregory P. Way and Casey S. Greene. Extracting a Biologically Relevant Latent Space from Cancer Transcriptomes with Variational Autoencoders. *bioRxiv*, page 174474, October 2017.

[20] John N. Weinstein, Eric A. Collisson, Gordon B. Mills, Kenna M. Shaw, Brad A. Ozenberger, Kyle Ellrott, Ilya Shmulevich, Chris Sander, and Joshua M. Stuart. The Cancer Genome Atlas Pan-Cancer Analysis Project. *Nature genetics*, 45(10):1113–1120, October 2013.

[21] Feng Yue, Yong Cheng, Alessandra Breschi, Jeff Vierstra, Weisheng Wu, Tyrone Ryba, Richard Sandstrom, Zhihai Ma, Carrie Davis, Benjamin D. Pope, Yin Shen, Dmitri D. Pervouchine, Sarah Djebali, Robert E. Thurman, Rajinder Kaul, Eric Rynes, Anthony Kirilusha, Georgi K. Marinov, Brian A. Williams, Diane Trout, Henry Amrhein, Katherine Fisher-Aylor, Igor Antoshechkin, Gilberto DeSalvo, Lei-Hoon See, Meagan Fastuca, Jorg Drenkow, Chris Zaleski, Alex Dobin, Pablo Prieto, Julien Lagarde, Giovanni Bussotti, Andrea Tanzer, Olgert Denas, Kanwei Li, M. A. Bender, Miaohua Zhang, Rachel Byron, Mark T. Groudine, David McCleary, Long Pham, Zhen Ye, Samantha Kuan, Lee Edsall, Yi-Chieh Wu, Matthew D. Rasmussen, Mukul S. Bansal, Manolis Kellis, Cheryl A. Keller, Christapher S. Morrissey, Tejaswini Mishra, Deepti Jain, Nergiz Dogan, Robert S. Harris, Philip Cayting, Trupti Kawli, Alan P. Boyle, Ghia Euskirchen, Anshul Kundaje, Shin Lin, Yiing Lin, Camden Jansen, Venkat S. Malladi, Melissa S. Cline, Drew T. Erickson, Vanessa M. Kirkup, Katrina Learned, Cricket A. Sloan, Kate R. Rosenbloom, Beatriz Lacerda de Sousa, Kathryn Beal, Miguel Pignatelli, Paul Flicek, Jin Lian, Tamer Kahveci, Dongwon Lee, W. James Kent, Miguel Ramalho Santos, Javier Herrero, Cedric Notredame, Audra Johnson, Shinny Vong, Kristen Lee, Daniel Bates, Fidencio Neri, Morgan Diegel, Theresa Canfield, Peter J. Sabo, Matthew S. Wilken, Thomas A. Reh, Erika Giste, Anthony Shafer, Tanya Kutyavin, Eric Haugen, Douglas Dunn, Alex P. Reynolds, Shane Neph, Richard Humbert, R. Scott Hansen, Marella De Bruijn, Licia Selleri, Alexander Rudensky, Steven Josefowicz, Robert Samstein, Evan E. Eichler, Stuart H. Orkin, Dana Levasseur, Thalia Papayannopoulou, Kai-Hsin Chang, Arthur Skoultchi, Srikanta Gosh, Christine Disteche, Piper Treuting, Yanli Wang, Mitchell J. Weiss, Gerd A. Blobel, Xiaoyi Cao, Sheng Zhong, Ting Wang, Peter J. Good, Rebecca F. Lowdon, Leslie B. Adams, Xiao-Qiao Zhou, Michael J. Pazin, Elise A. Feingold, Barbara Wold, James Taylor, Ali Mortazavi, Sherman M. Weissman, John A. Stamatoyannopoulos, Michael P. Snyder, Roderic Guigo, Thomas R. Gingeras, David M. Gilbert, Ross C. Hardison, Michael A. Beer, Bing Ren, and Mouse ENCODE Consortium. A comparative encyclopedia of DNA elements in the mouse genome. *Nature*, 515(7527):355–364, November 2014.