

PowerGraph: Using neural networks and principal components to multivariate statistical power trade-offs

Ajinkya K Mulay, Sean Lane, Erin Hennes

Purdue University
610 Purdue Mall
West Lafayette, Indiana, 47907
mulay@purdue.edu, lane84@purdue.edu, ehennes@purdue.edu

Abstract

It is increasingly acknowledged that a priori statistical power estimation for planned studies with multiple model parameters is inherently a multivariate problem. Power for individual parameters of interest cannot be reliably estimated univariately because sampling variability in, correlation with, and variance explained relative to one parameter will impact the power for another parameter, all usual univariate considerations being equal. Explicit solutions in such cases, especially for models with many parameters, are either impractical or impossible to solve, leaving researchers with the prevailing method of simulating power. However, point estimates for a vector of model parameters are uncertain, and the impact of inaccuracy is unknown. In such cases, sensitivity analysis is recommended such that multiple combinations of possible observable parameter vectors are simulated to understand power trade-offs. A limitation to this approach is that it is computationally expensive to generate sufficient sensitivity combinations to accurately map the power trade-off function in increasingly high dimensional spaces for the models that social scientists estimate. This paper explores the efficient estimation and graphing of statistical power for a study over varying model parameter combinations. Optimally powering a study is crucial to ensure a minimum probability of finding the hypothesized effect. We first demonstrate the impact of varying parameter values on power for specific hypotheses of interest and quantify the computational intensity of computing such a graph for a given level of precision. Finally, we propose a simple and generalizable machine learning inspired solution to cut the computational cost to less than 7% of what could be called a brute force approach. Empirically we show that such a model can achieve over 97% testing accuracy, which is within 1% error for statistical power. We achieve such a high model performance with careful tuning for a simple linear regression and repeated measures ANOVA, and by adopting the approach, well-beyond into much more complex models. Moreover, we can automate the tuning procedure based on user-specified levels of desired precision.

Statistical power is quantitatively equal to the probability and thus, plays a crucial role in ensuring the probability of finding an effect of hypothesized interest in any study. Historically, the simplest and most direct way to improve power is to increase the study's sample size, as mathematically, it

has the most significant impact with respect to what the researcher can control (Cohen 1992). However, there are practical considerations to make when increasing sample size. In the case of a rare disease study, the total population itself might be small, difficult to locate, and even more challenging to engage, as in early- and late-stage neuro-genetic syndromes (Button et al. 2013; Szucs and Ioannidis 2017). On the other hand, there could be access or funding issues with increasing the number of participants a researcher wishes to obtain.

In this article, we empirically show that even tuning the model parameters (*i.e.*, weights) can dramatically change the study's power. Thus, the change in model parameters can push the study from a mid-powered study to a well-powered study even for a constant sample size. For the rest of the article, we assume that a well-powered study is one where the power is higher than 0.8. Fig. 1 demonstrates that even for a fixed sample size (here, $N = 105$), the power can range from 0.3 to 1 varying on the values of the model weights. In fig. 1 we compute power *gradients* by considering the directional change between clusters derived by KMeans where power and the squared L2-norm of the model parameters represent the point coordinates.

Thus, generating such a manifold, as shown in Fig. 1 can exceptionally aid researchers in identifying areas of high power. Enabling access to these plots is thus a simple way of reducing the *ideal* sample size without sacrificing power. However, as described in Algorithm 1, computing power even once requires 200-1000 simulations. Further, the manifold parameter space, including the model weights, sample size, and additional hyperparameter choices, is enormous. For instance, for a seven predictor model, with 11 choices per predictor and 11 choices for the N , we have a parameter space of 11^{7+1} . Computation for this vast space might even take days.

Thus, in this work, we look at *cheaper* alternatives to simulating the entire power manifold. We present three different contributions-

- We first provide an unsupervised algorithm, *POWERCLUSTER*, to easily separate the parameter space into areas of distinct power values with *no* power calculation.
- Since *POWERCLUSTER* does not generalize well; We provide a supervised learning algorithm *POWERNET-WORK* which utilizes a simple neural network to predict

power in the manifold with accuracy greater than 97%. We demonstrate the efficacy of the model on two standard models-*REG* AND *RMANOVA* (both described later in the Experiments section).

- Finally, we provide insights into the limitations of both methods over the change in the number of predictors and the complexity of the underlying model.

We organize the rest of the article as follows. We first detail the relevant literature and highlight the background information required to build our algorithms. We provide pseudocodes for the core and auxiliary algorithms in this section. Finally, we present our empirical results, highlight the significant wins and report the limitations in the next section. Finally, we also present potential future work.

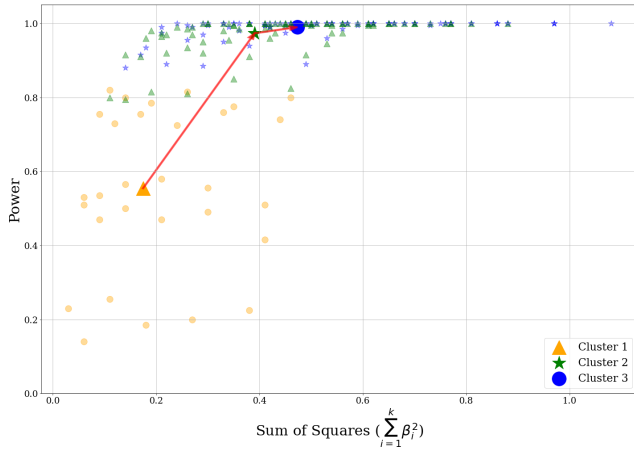


Figure 1: Change in power over varying model coefficients (β) and a fixed $N(= 105)$. We observe significant power gradient over clusters derived by KMeans for a three-predictor linear regression model for a partial f-test.

Related Work

Previous articles ((Bakker, Van Dijk, and Wicherts 2012), (Bakker et al. 2016), (Maxwell 2004), (Cohen 1992)) have pointed out that studies are frequently underpowered and thus lead to statistically insignificant results. Underpowered studies are primarily a result of a lack of a formal power analysis. (Bakker, Van Dijk, and Wicherts 2012) even provides power simulations for studies using Questionable Research Practices (QRPs) and lower sample sizes with more trials. The results show that such practices can significantly inflate statistical power and provide misleading evidence about the effect size while hampering the study’s reproducibility. QRPs could include running multiple trials with a much smaller sample size (underpowered), rerunning analyses after adding more subjects, or removing outliers. (Bakker, Van Dijk, and Wicherts 2012) suggests that to avoid such underpowered studies, we should use sample sizes derived after a formal power analysis.

(Baker et al. 2021) solves the power issue by providing an online tool with power contours to demonstrate the effect of

sample size (N) and trials per participant on statistical power (k). With the results provided in the article, it is clear that changes to N or even k can dramatically change the power region and convert an underpowered study to an appropriately powered one. (Rast and Hofer 2014) also demonstrates the powerful (inversely correlated) impact of sample size on both the effect size and the study design.

(Lane and Hennes 2018) and (Lane and Hennes 2019) provide a clear guide to conducting formal power analysis based on simulation methods (rather than a formula-based approach). Even though the simulation approach is universal, it often requires a large number of computational resources. The resource usage increases exponentially with a linear increase in the number of predictors or other factors impacting power. Thus, in this work, we highlight machine learning-based approaches that can seriously reduce resource usage with accuracies greater than 95%.

Proposed Work

We first introduce some standard terms used in statistical power and machine learning.

Principal Component Analysis Complex datasets include several features, and it often becomes necessary to reduce data dimensionality to conserve resources or speed up training. Further, we wish to remove redundancy in features. Removing highly correlated features can improve training speeds or data processing speeds, reduce bias, and improve the interpretability of our dataset.

A common way to achieve these goals is to use Principal Component Analysis (PCA) (Wold, Esbensen, and Geladi 1987). PCA works by finding the direction with maximal variance. Next, it finds the direction with the maximal variance such that this direction is uncorrelated with the previous direction. We continue in this fashion so that any pair of directions are uncorrelated with each other. We then re-orient our dataset onto these new components to compute our new dataset. Suppose our original dataset is $X \in \mathbb{R}^{N \times p}$ and $v \in \mathbb{R}^{p \times k}$ are the derived Principal Components (PCs) where N is the total samples in X , $p = \dim(X)$ and k is the number of PCs to be selected. Then the new dataset is given by,

$$X' = X \times v.$$

For brevity, we skip further details of PCA. We, however, provide a simple PCA algorithm in Algorithm 2.

Computing Power

We present a power computation algorithm in Algorithm 1 for t-tests in a linear regression model. We can extend the f-test by replacing the t-test in Line 9 with an f-test or a partial f-test. For extensions to other models, we need to modify the data generation process of Line 5.

Computing power for multiple model weights requires us to call the *COMPUTE-POWER* function each time. Thus, the cost of computing power boils down to the number of calls made to *COMPUTE-POWER*. We wish to reduce the number of calls while still predicting power for the entire parameter space in our work.

Improving correlation with PCA

We have already noted that increasing the sample size N can increase the power of a study. To visualize the impact of each parameter in the model, we take a look at a partial f-test. The data follows the distribution of the first three variables from Table 2. We compute the power of the partial f-test to test for the significance of the first and third predictors.

To visualize the importance of each parameter we draw a correlation graph in Figure 2. We can easily verify that as expected β_1 and β_3 have higher correlation with the power. Further, N has a high correlation as we would expect. Finally, we define a new feature *scaled standard deviation* computed as $N\sigma$, where $\sigma = \sqrt{\sum_{i=1}^k \beta_i^2}$ (k denotes the number of predictors). We can see that this feature improves the correlation further.

Finally, including the principal components bumps up the correlation to ≥ 0.90 for the first component. We believe that transforming the data with PCA can thus maximize the variance while ignoring the redundant features. We, therefore, use this new dataset to train both of our algorithms.

POWER-CLUSTER

Here we provide our first algorithm, *POWERCLUSTER* (also referred to as *P-CLUSTER*). We run a simple K-Means on PC_1 and $N\sigma$ for identifying two clusters with two vectors from our dataset. We provide its pseudocode in Algorithm 4. K-Means identifies clusters of points that minimize the intra-cluster distances. Since both of our feature vectors are independent of power, we can cluster our data points without computing the *true* power.

P-CLUSTER can provide us with a quick scope of the parameter space and help us quickly filter out low-power domains. Thus, we could avoid exploring these low power domains and reduce the power computation cost (*i.e.* by reducing calls to *COMPUTE-POWER*).

POWER-NETWORK

Even though *P-CLUSTER* gives us a good idea of the parameter space, it lacks knowledge of the underlying ground truth. Thus, it is unrealistic to expect it to perform well without the knowledge of the *true* power values. Thus, we employ the use of a simple neural network as described in Table 1 trained on the *new* dataset. We denote this approach by *POWER-NETWORK* or *POWER-Neural Network (PNN)*.

Experiments

We test the efficacy of the Algorithms 4 and 5 with two simple models- linear regression (*REG-p*) and Repeated Measures ANOVA (*RMANOVA*). *REG-p* uses distribution from Table 2 with only the first p -predictors. *RMANOVA* uses the same p -predictors but with two factors (*i.e.*, observations are twice the sample size). For simulation, we use Google Colaboratory with the standard run-time. We use a grid-like parameter space for sampling our parameters. As a baseline, we consider the brute-force approach, which involves simulating a fine grid of parameters, and it exceeds 8000 calls for

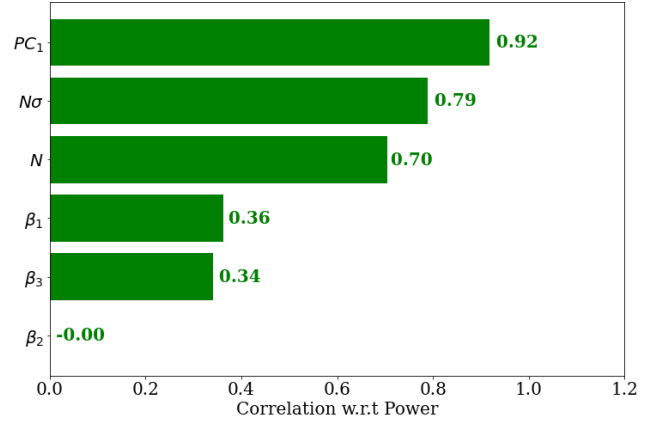


Figure 2: Correlation of parameters with *true* power. With feature engineering and PCA we are able to obtain correlation > 0.9 for a linear regression model with 3 parameters.

Algorithm 1: Computing Power of a t-test for a Linear Regression Model

```

1: Input: Distributions of columns in the dataset- $\mathcal{D} = \{D_{X_1}, D_{X_2}, \dots, D_{X_p}\}$ , Sample Size- $N$ , Model Weight- $\beta \in \mathbb{R}^p$ , Number of predictors- $p$ , sensitivity- $\alpha$  (default: 0.05), number of simulations-sims (default: 200), Error Distribution- $\mathcal{E}$ 
2: Output: Power of t-test
3: procedure COMPUTE-POWER( $X, k$ )
4:   significance  $\leftarrow 0$ 
5:   for 1:sims do
6:      $X \leftarrow$  generate  $N$  data samples from distribution( $\mathcal{D}$ )
7:      $e \leftarrow$  generate error from distribution( $\mathcal{E}$ )
8:      $y \leftarrow (X \times \beta) + e$ 
9:      $\mathcal{M} \leftarrow$  Fit  $(X, y)$  to a linear regression model
10:    if p-value of t-test for model  $\mathcal{M} \leq \alpha$  then
11:      significance  $+= 1$ 
12:  power  $\leftarrow \frac{\text{significance}}{\text{sims}}$ 
13:  return power

```

the *REG-3* model). Complex models further increase these calls.

P-CLUSTER can quickly identify appropriately varying power domains as seen from Fig. 4. If we compare the clustering to the standard definition of a high-powered study (*i.e.*, power is more significant than 0.8), we can segregate power with high performance. We demonstrate the results in Figure 3. We can see that *P-CLUSTER* performs well only for the simpler models. Thus, we can empirically conclude that not including label information will lead to poor performance for complex models.

To counter this issue, we instead use the more powerful neural network- *PNN*. *PNN* consistently outperforms *P-CLUSTER* and can achieve high performance for even the *RMANOVA* model.

We tune the hyperparameters of the *PNN* with Optuna

Algorithm 2: Transform dataset with first k Principal Components

```

1: Input:  $X \in \mathbb{R}^{N \times p}$  dataset, number of samples in  $X$ - $N$ ,
   data dimension- $p$ , number of components to be included
   with PCA- $k$  ( $k < p$ )
2: Output: Transformed PCA dataset
3: procedure PCA-FIT-TRANSFORM( $X, k$ )
4:   for  $X_i$  in columns( $X$ ) do
5:      $\mu_{X_i} \leftarrow \text{MEAN}(X_i)$ 
6:      $\sigma_{X_i} \leftarrow \text{STANDARD-DEVIATION}(X_i)$ 
7:      $X_i \leftarrow \frac{(X_i - \mu_{X_i})}{\sigma_{X_i}}$ 
8:    $Y \leftarrow X.T \times X$ 
9:    $V_k \leftarrow \text{FIRST-}k\text{-EIGENVECTORS}(Y)$ 
10:  return  $X \times V_k$ 

```

Algorithm 3: Training Data Collection

```

1: Input: Length of training set- $L$ , model parameter
   space- $\mathcal{S}$  of length  $L$  (each parameter consists of the
   weight vector  $\beta$  and sample size  $N$ ), power sensitivity- $\alpha$ 
   (default 0.05), number of simulations- $\text{sims}$  (default 200)
2: Output: Training Set  $\mathcal{S}$  with parameters and corre-
   sponding powers
3: procedure GENERATE-DATA( $\mathcal{S}, \alpha, \text{sims}$ )
4:   for parameter in  $\mathcal{S}$  do
5:      $(\beta, N) \leftarrow \text{parameter}$ 
6:      $\text{power} \leftarrow \text{COMPUTE-POWER}(\beta, N, \alpha, \text{sims})$ 
7:      $\mathcal{S}[\text{powers}] \leftarrow \text{power}$ 

```

Algorithm 4: Unsupervised clustering of the power surface with PCA features

```

1: Input: Length of training set- $L$ , model parameter
   space- $\mathcal{S}$  of length  $L$  (each parameter consists of the
   weight vector  $\beta$  and sample size  $N$ ), power sensitivity- $\alpha$ 
   (default 0.05), number of simulations- $\text{sims}$  (default
   200), PCA variance (in %) to be retained- $\text{var}$ , Number
   of clusters- $k$ 
2: procedure POWERCLUSTER( $\mathcal{S}, \alpha, \text{sims}$ )
3:   Output: Power Surface Graph
4:    $\mathcal{S}^* \leftarrow \text{GENERATE-DATA}(\mathcal{S}, \alpha, \text{sims})$ 
5:    $\mathcal{S}_{\text{PCA}} \leftarrow \text{PCA-FIT-TRANSFORM}(\mathcal{S}, \text{variance}=\text{var})$ 
6:    $C_k \leftarrow \text{KMeans}(\mathcal{S}_{\text{PCA}}, \text{num\_clusters} = k)$ 
7:   Return clusters  $C_k$  derived from KMeans

```

(Akiba et al. 2019) over the choice of the learning rate, batch size, and the regularization parameter. We choose the *Adam* optimizer since it consistently outperforms the rest. We use an 80/20 split for training and testing data. We note that careful tuning *PNN* is essential to obtain high-performing networks.

Results

We note that both approaches are incredibly versatile and work in any simulation-based power analysis scenario. *P-CLUSTER* is a quick way of discarding low power regions

Algorithm 5: Predicting power surface with PCA features

```

1: Input: Length of training set- $L$ , model parameter
   space- $\mathcal{S}$  of length  $L$  (each parameter consists of the
   weight vector  $\beta$  and sample size  $N$ ), power sensitivity- $\alpha$ 
   (default 0.05), number of simulations- $\text{sims}$  (default
   200), PCA variance (in %) to be retained- $\text{var}$ 
2: procedure POWNETWORK( $\mathcal{S}, \alpha, \text{sims}$ )
3:   Output: Power Surface Graph
4:    $\mathcal{S}^* \leftarrow \text{GENERATE-DATA}(\mathcal{S}, \alpha, \text{sims})$ 
5:    $\mathcal{S}_{\text{PCA}} \leftarrow \text{PCA-FIT-TRANSFORM}(\mathcal{S}, \text{variance}=\text{var})$ 
6:   Train Neural Network Model  $\mathcal{M}$  on  $(\mathcal{S}^* \cup \mathcal{S}_{\text{PCA}})$ 
7:   Return trained model  $\mathcal{M}$ 

```



Figure 3: Performance of *P-CLUSTER* with increase in model complexity. We increase complexity by either adding more predictors or using a more complicated base model.

as it encodes partial information about the model-based on just the model weights and the sample size. Note that *P-CLUSTER* requires no calls to the *COMPUTE-POWER* function and essentially runs in constant time.

PNN, on the other hand, is quickly able to capture information about the underlying model to predict power with high accuracy. *PNN* requires some training data (obtained by calls to *COMPUTE-POWER*). However, it is significantly less than the data needed for a brute-force approach. A significant limitation of *PNN* is that it continually requires more training points as the complexity of the underlying model rises.

Layer	Parameters
Dense (input)	64 units + ReLU
Dense (hidden)	32 units + ReLU
Dense (output)	1 unit + Sigmoid

Table 1: *PNN* Architecture: We report the fully connected layers. The input dimensions depend on the number of predictors and additional PCA features.

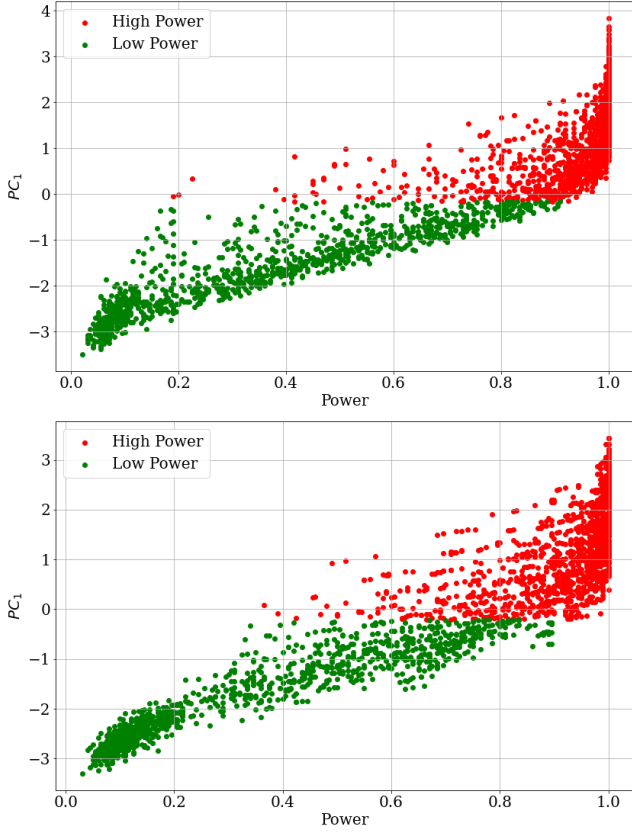


Figure 4: A simple unsupervised learning algorithm can identify high and low power domains. (top) Results for partial f-test performed on a 3-predictor *REG* model. (bottom) Results for partial f-test performed on a 5-predictor *REG* model.

D_{X_1}	D_{X_2}	D_{X_3}	D_{X_4}	D_{X_5}
CAT[-1, 1]	$\mathcal{N}(0, 1)$	$X_1 \times X_2$	$\mathcal{N}(0, 2)$	$X_4 \times X_2$
D_{X_6}	D_{X_7}	D_{X_8}	D_{X_9}	$D_{X_{10}}$
CAT[0, 1, 2]	$\mathcal{N}(0, 2)$	$X_6 \times X_7$	$\mathcal{N}(0, 1)$	$X_2 \times X_6$

Table 2: Distribution of model features. For predictors $k < 10$, we only select the first k predictors. Note that CAT refers to a categorical variable while $\mathcal{N}(\mu, \sigma)$ is the Normal Distribution with mean μ and standard deviation σ .

Conclusion

In this work, we show that PCA can significantly aid in exploring the power surface for any study at a fractional cost (less than 7% of the time required by the brute force approach). Even though unsupervised learning provides a good prediction of the power surface for simple models, it fails to generalize to complex models since they lack information about the *true* power. We, however, provide a universal solution by leveraging a simple neural network on top of our feature engineering with PCA, which consistently predicts with an accuracy above 95%. Our supervised methods require a much smaller fraction of the total samples usually

needed for a brute force approach. We want to explore strategies to bridge the gap between unsupervised learning and supervised methods in future work.

Model	REG-3	REG-5	REG-7	RMANOVA-3
Dataset Size	2376	2592	3888	1296
Collection Time (secs)	500	631	1027	5640
Hypothesis (Zero β)	β_1, β_3	$\beta_1, \beta_3, \beta_5$	$\beta_1, \beta_3, \beta_7$	N/A

Table 3: Dataset properties for the various models. We compute the partial f-test to identify the significance of the parameters mentioned.

References

- Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; and Koyama, M. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2623–2631.
- Baker, D. H.; Vilidaitė, G.; Lygo, F. A.; Smith, A. K.; Flack, T. R.; Gouws, A. D.; and Andrews, T. J. 2021. Power contours: Optimising sample size and precision in experimental psychology and human neuroscience. *Psychological Methods*, 26(3): 295.
- Bakker, M.; Hartgerink, C. H.; Wicherts, J. M.; and van der Maas, H. L. 2016. Researchers’ intuitions about power in psychological research. *Psychological science*, 27(8): 1069–1077.
- Bakker, M.; Van Dijk, A.; and Wicherts, J. M. 2012. The rules of the game called psychological science. *Perspectives on Psychological Science*, 7(6): 543–554.
- Button, K. S.; Ioannidis, J. P.; Mokrysz, C.; Nosek, B. A.; Flint, J.; Robinson, E. S.; and Munafò, M. R. 2013. Power failure: why small sample size undermines the reliability of neuroscience. *Nature reviews neuroscience*, 14(5): 365–376.
- Cohen, J. 1992. Things I have learned (so far). In *Annual Convention of the American Psychological Association, 98th, Aug, 1990, Boston, MA, US; Presented at the aforementioned conference*. American Psychological Association.
- Lane, S. P.; and Hennes, E. P. 2018. Power struggles: Estimating sample size for multilevel relationships research. *Journal of Social and Personal Relationships*, 35(1): 7–31.
- Lane, S. P.; and Hennes, E. P. 2019. Conducting sensitivity analyses to identify and buffer power vulnerabilities in studies examining substance use over time. *Addictive behaviors*, 94: 117–123.
- Maxwell, S. E. 2004. The persistence of underpowered studies in psychological research: causes, consequences, and remedies. *Psychological methods*, 9(2): 147.
- Rast, P.; and Hofer, S. M. 2014. Longitudinal design considerations to optimize power to detect variances and covariances among rates of change: simulation results based on actual longitudinal studies. *Psychological methods*, 19(1): 133.

Szucs, D.; and Ioannidis, J. P. 2017. Empirical assessment of published effect sizes and power in the recent cognitive neuroscience and psychology literature. *PLoS biology*, 15(3): e2000797.

Wold, S.; Esbensen, K.; and Geladi, P. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3): 37–52.