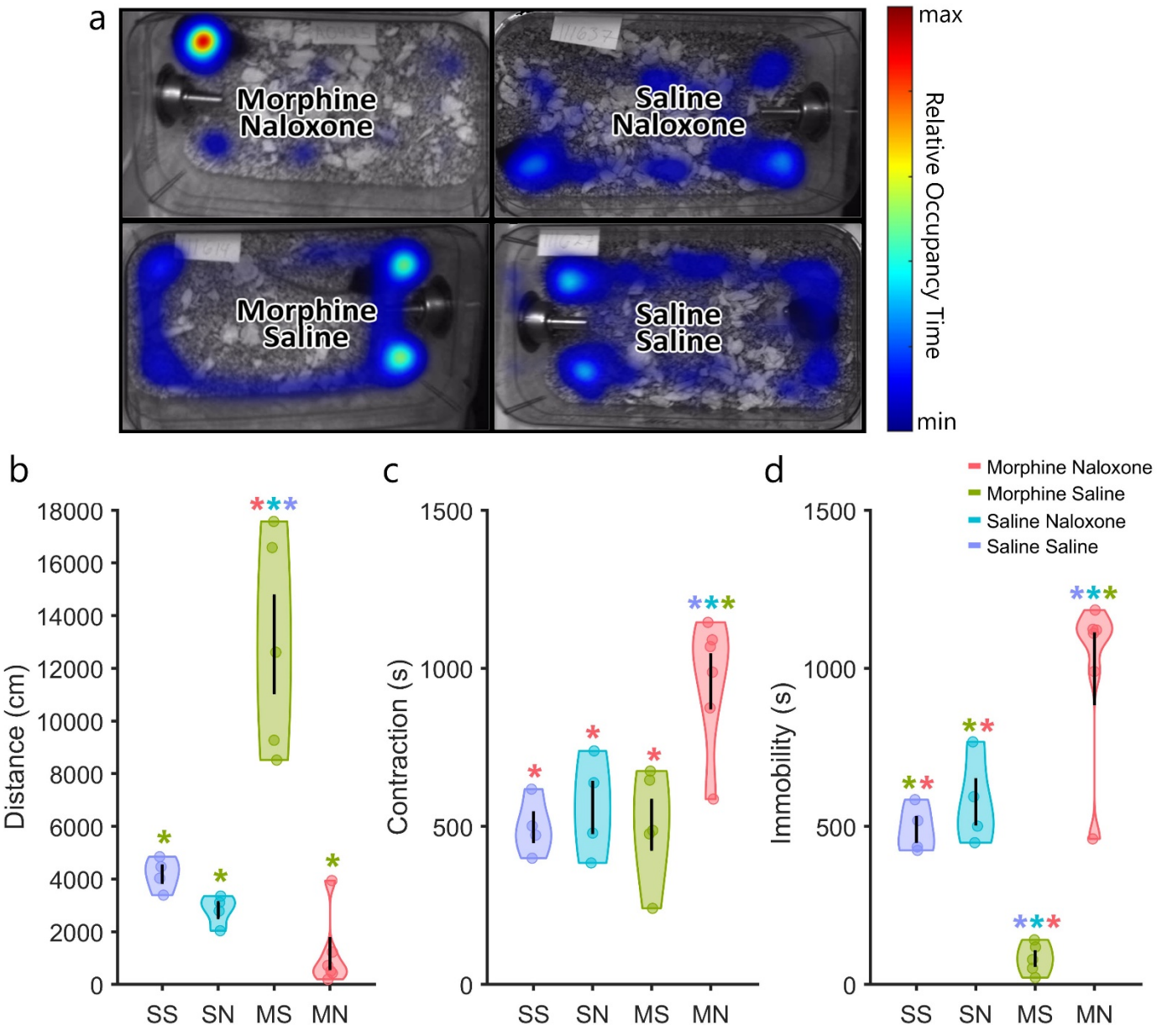
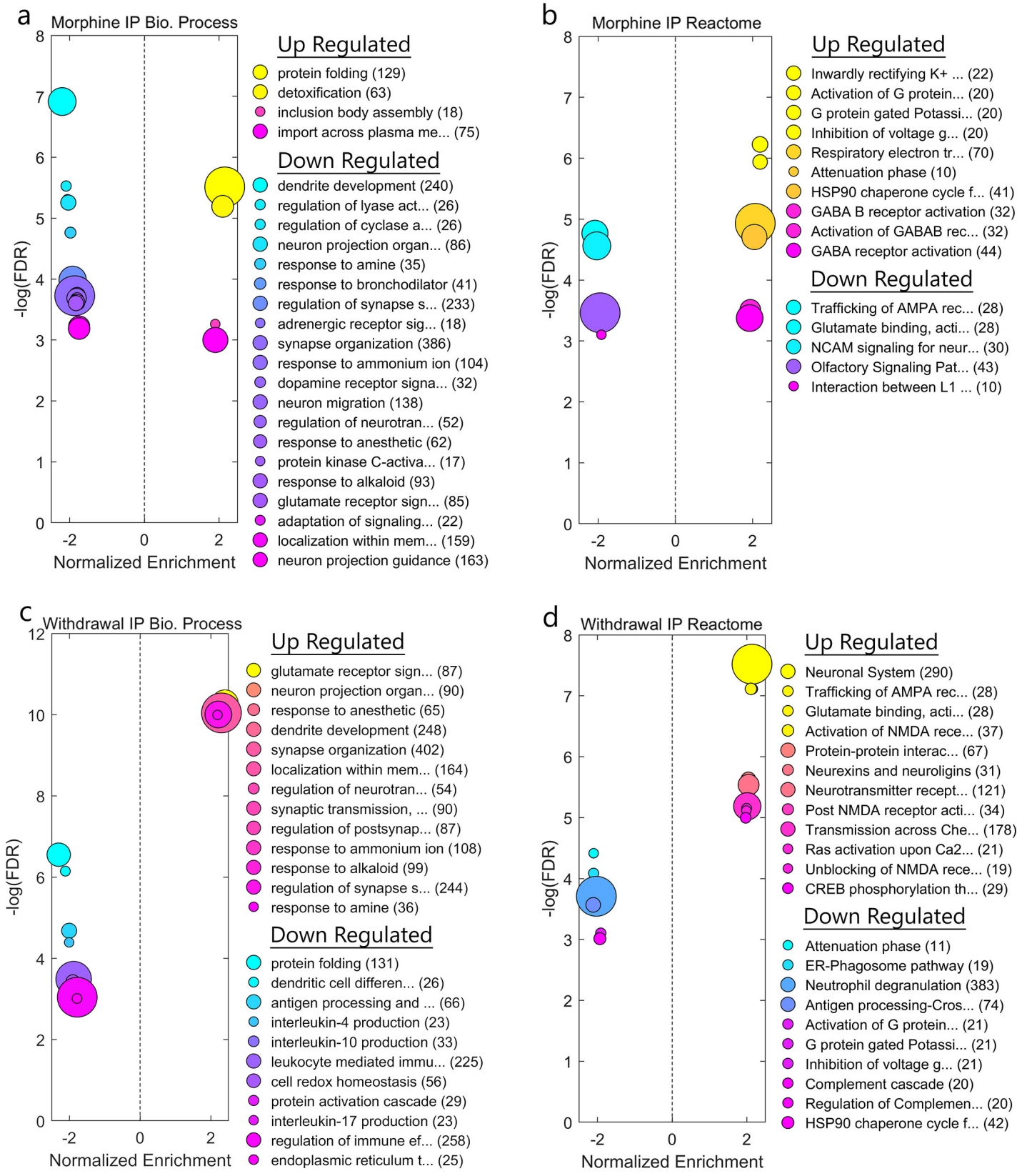
A Cyclic Adenosine Monophosphate Related Gene Network in Microglia Is Inversely Regulated by Morphine Tolerance and Withdrawal

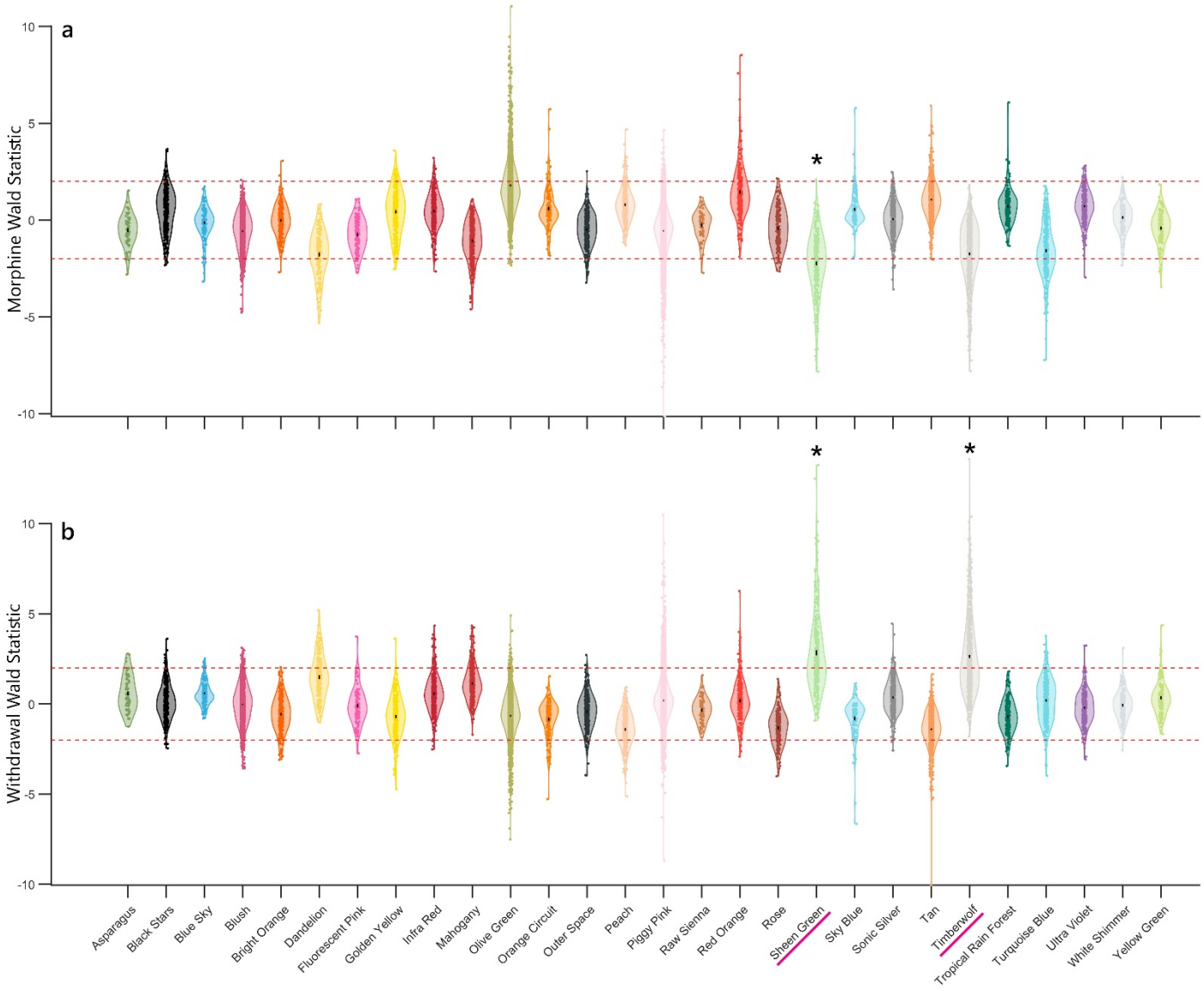
Supplemental Information



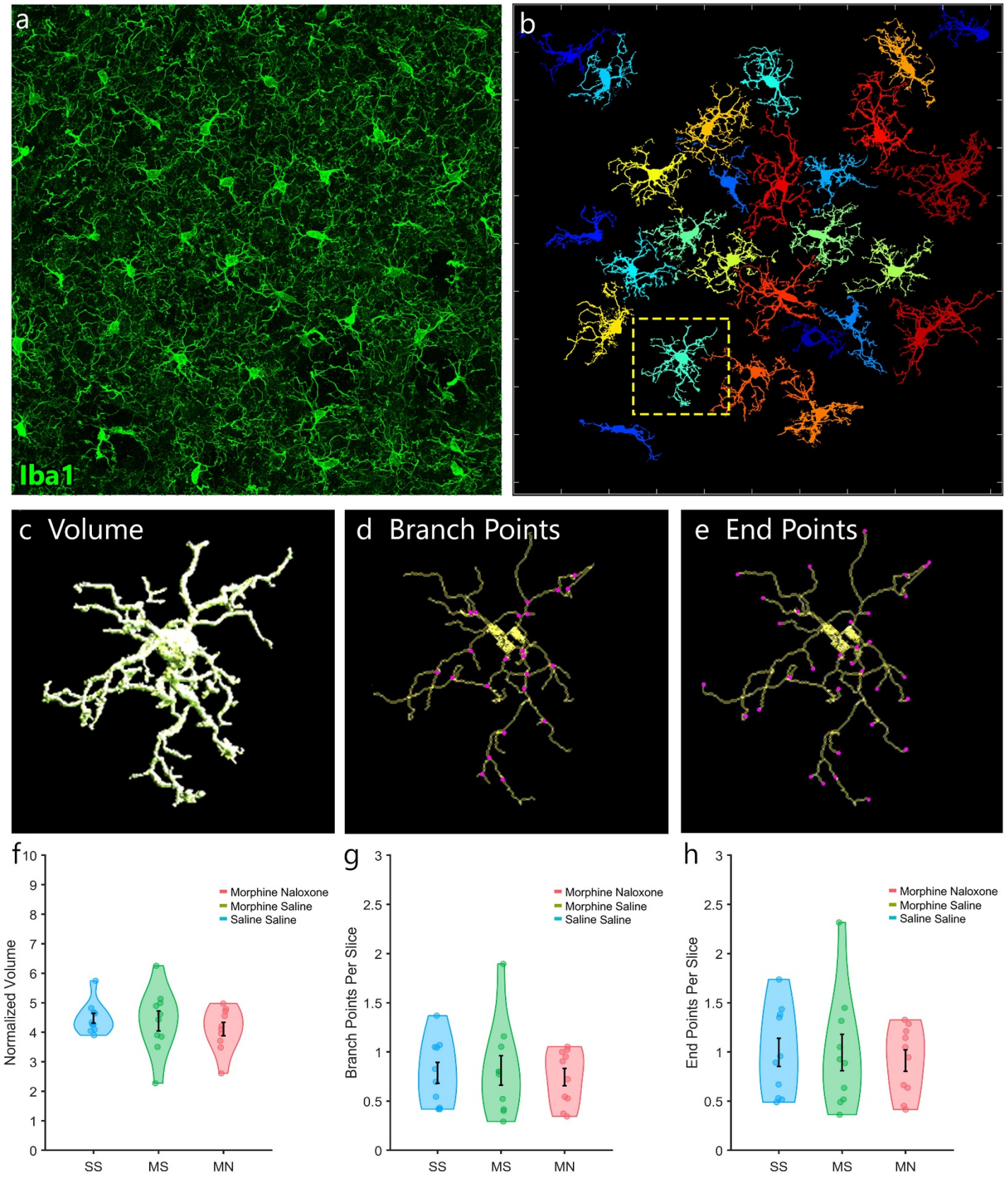
Supplementary Figure S1. Naloxone Precipitates Morphine Withdrawal | a, Automatic scoring of morphine and withdrawal behavior. **b,** Morphine administration to tolerant mice induces locomotor sensitization, while naloxone precipitated withdrawal induces an aversive state that is reflected in **c,** a hunched body posture (contraction) and **d,** increased immobility**.** Saline-saline animals and saline-naloxone animals did not significantly differ in any tested behavior. *p<.05; color indicates comparison.



Supplementary Figure S2. Morphine Tolerance and Withdrawal Induce Inverse Gene Set Enrichment | GSEA reveals many annotated gene sets that are significantly upregulated or downregulated in microglia. **a,b,** For example, morphine tolerance upregulates protein folding genes and inwardly rectifying K+ channels, and downregulates neuronal projection and synaptic organization genes. **c,d,** On the other hand, withdrawal upregulates neuronal projection and synaptic organization genes and CREB phosphorylation genes, and downregulates interleukin production, and G protein gated K+ channels. Gene-sets are sorted by normalized enrichment while color represents –log(FDR).



Supplementary Figure S3. WGCNA Module Significance | Most gene modules are unrelated to morphine tolerance and withdrawal. However, two gene modules, Sheen Green and Timberwolf, are on average **a,** downregulated during morphine tolerance, and **b,** upregulated in withdrawal.



Supplementary Figure S4. Morphine Tolerance and Withdrawal Does Not Affect Gross Microglia Morphology | a, Maximum intensity projection of striatal microglia captured at 40x (30µm thick). **b,** Automated cell segmentation and removal of incomplete (border touching) cells using 3DMorph. **c,** 3D volumetric reconstruction of the highlighted microglia in the above panel. **d,** Automatically calculated branching points generated from a 3D skeletonization of the cell. **e,** Automatically calculated end points generated from a 3D skeletonization of the cell. **f,** There was no effect of morphine administration on increasing cell volume, **g,** process branching, or **h,** terminations (end points).

Supplementary Methods.

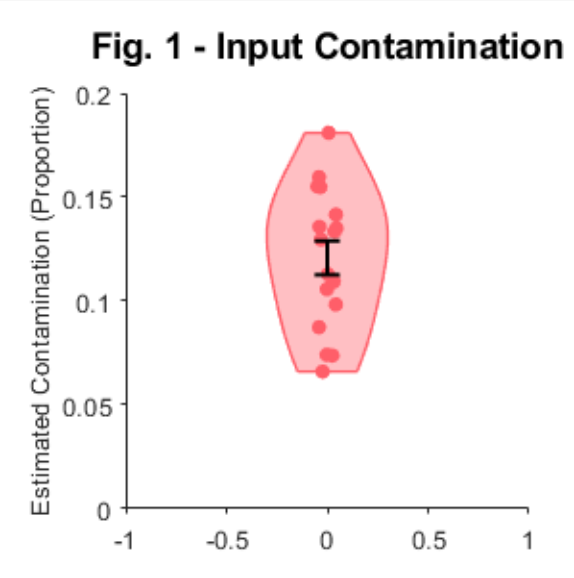
Removing Input Contamination from RiboTag RNA-Seq data

RiboTag-Seq studies often rely on the concept of "enrichment" to determine which genes are specific to the cell type of interest and which are positively differentially expressed in the IP samples versus the Input samples. This methodology is extremely effective at isolating genes that are highly expressed in each cell type. However, this technique should not be used to rule out a gene as expressed and functional in a cell of interest. Just because a gene is expressed at lower levels in one cell type than in the surrounding tissue, does not mean that the gene was detected spuriously, and precluded from functional relevance. To mitigate this issues, we developed a new approach to remove Input contamination from IP samples computationally and to determine which gene changes can reliably be ascribed to microglia. To test the validity of this approach, we modeled a sequencing run of IP and Input samples where gene quantities vary independently between samples. We simulated contamination by mixing the "true" input and IP samples and generating sequencing data from these pooled distributions. Before modeling, we introduced contaminating (*gamma generated*) Input RNA into the IP gene pool. After random sampling from this contaminated gene pool, we tested our contamination removal method and compared the results to a modeled run without contamination. This technique improved the correlation between estimated TPM and "true" reads, even when the estimate of contamination is approximated. Bellow we use a combination of actual data and modeling to show how this technique can be applied, and the efficacy of noise removal even given imperfect assumption about the quantity of contamination. To our knowledge, this is the first method to systematically model and remove Input contamination from RiboTag RNA-Seq data. All code below was written in Matlab 2019a.

Understanding the Level of Contamination

Ribotag in the present study was expressed in microglia by crossing floxed-RiboTag mice with tamoxifen inducible cx3cr1-CRE mice. By omitting tamoxifen administration from 2 animals and running the complete RiboTag protocol in parallel with the true RiboTag animals, we can get a good estimate of non-specific RNA pulldown from our procedures. All RNA is reported in nanograms. In the present study we estimate a range of contamination from 6.6-15.9% with an average of 12%.

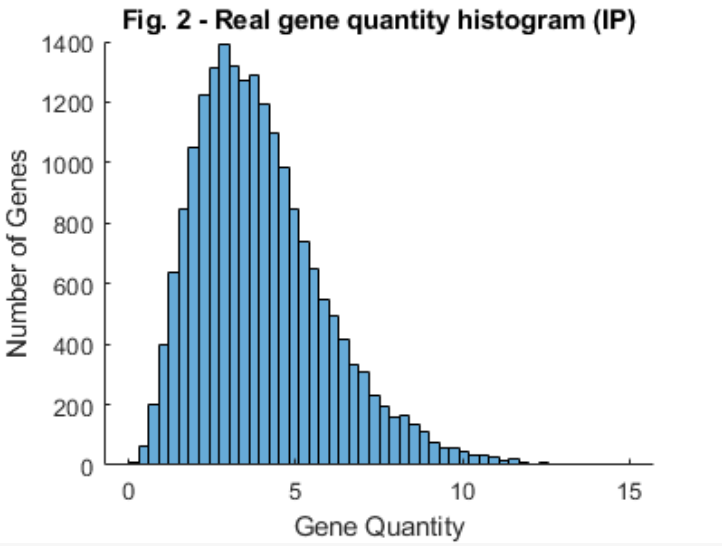
```
RiboTag_RNA=([54.65 60.76 52.94 44.24 80.60 90.52...
        80.97 68.37 37.43 44.03 38.49 56.53...
        33.01 38.58 44.74 46.11 42.19]); % Actual RNA Yield
Negative_Control_RNA=([6.56 5.39]); % Actual RNA Yield
Estimated_Contamination=mean(Negative_Control_RNA)./RiboTag_RNA;
Mean_Contamination=mean(Estimated_Contamination);
figure('Position',[1 1 300 300]);
g=gramm('x',zeros(17,1)','y',Estimated_Contamination);
g.stat_violin('normalization','width','fill','transparent');
g.geom_jitter('width',.1);
g.stat_summary('geom',{'black_errorbar'},'type','sem');
g.axe_property('XLim',[-1 1],'YLim',[0 .20],'tickdir','out');
g.set_names('x',[],'y','Estimated Contamination (Proportion)','color','Gene Length');
g.set_title('Fig. 1 - Input Contamination');
g.draw;
% GENERATE FIGURE 1
```



Generating "True" Gene Counts for Contamination Modeling

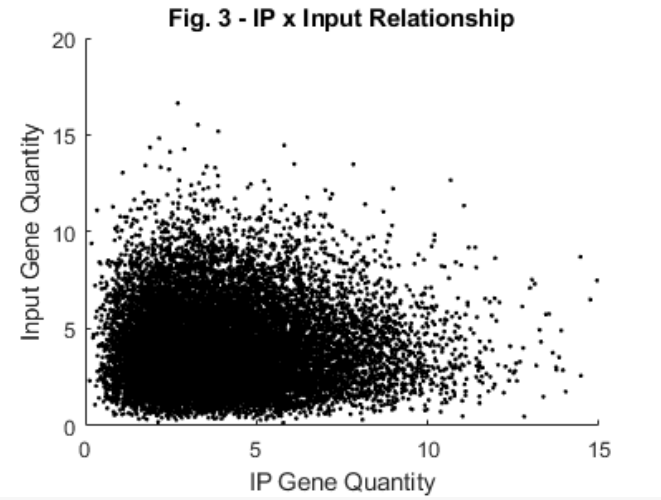
First, we generated a model of "true" gene quantity for the Input and IP samples. These values represent the underlying "true" quantity of each gene and are generated using a gaussian distribution with a mean of 4. Gene quantities in the IP and Input sample are not constrained and may differ greatly for any given gene, as is the case with genes in different cell types.

```
number_of_genes = 20000;
gene_quantity_IN = randg(4,number_of_genes,1);
gene_quantity_IP = randg(4,number_of_genes,1);
figure('Position',[1 1 400 300]);
histogram(gene_quantity_IP);
box off;
xlabel('Gene Quantity');
ylabel('Number of Genes');
title("Fig. 2 - 'True' Gene Quantity Histogram (IP)");
% GENERATE FIGURE 2
```



```
figure('Position',[1 1 400 300]);
scatter(gene_quantity_IP,gene_quantity_IN,4,"black","filled");
box off;
xlabel('IP Gene Quantity');
ylabel('Input Gene Quantity');
title("Fig. 3 - IP x Input Relationship");
```

% GENERATE FIGURE 2



Generating Simulated RNA-Seq Reads and TPM

RNA-Seq is based around technology that reads a finite number of gene fragments. One consequence of this method is that the final data are a proportional representation of the "true" gene counts, where longer genes are more likely to be counted. To control for this, specialized units have been developed such as transcripts per million reads (TPM). When calculating TPM, you normalize for gene length first, and then normalize for sequencing depth second. The sum of all TPMs in each sample are the same (1e6). This makes it easier to compare the proportion of reads that mapped to a gene in each sample.

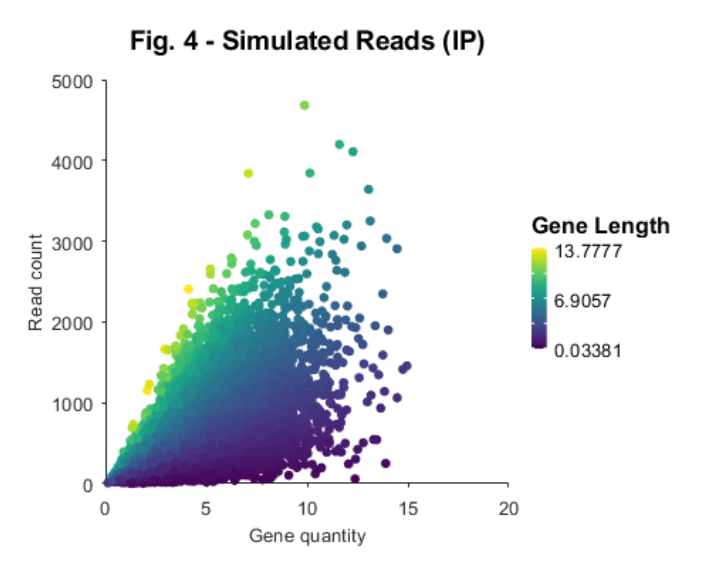
Below we model the RNA-Sequencing bias for long genes and show how calculating TPM can correct this bias and produce values that nearly perfectly match the "true" underlying gene count per million reads (CPM).

```
number_of_reads = 1e7; % Read depth of our modeled sequencing run
gene_length = randg(3,number_of_genes,1); % Random gene length from gamma (mean=3)
% IN
read_probability_IN = gene_quantity_IN .* gene_length;
% Normalized read probability
read_probability_IN = read_probability_IN ./ sum(read_probability_IN);
% Generate 1e7 sequencing counts based on the read probability of each gene
read_count_IN = poissrnd(number_of_reads * read_probability_IN);
TPM_IN = calculateTPM(read_count_IN, gene_length); % Calculate TMP
% IP
read_probability_IP = gene_quantity_IP .* gene_length;
% Normalized read probability
read_probability_IP = read_probability_IP ./ sum(read_probability_IP);
% Generate 1e7 sequencing counts based on the read probability of each genes
read_count_IP = poissrnd(number_of_reads * read_probability_IP);

TPM_IP = calculateTPM(read_count_IP, gene_length); % Calculate TMP
figure('Position',[1 1 500 400]);
g = gramm('x',gene_quantity_IP,'y',read_count_IP, 'color', gene_length);
```

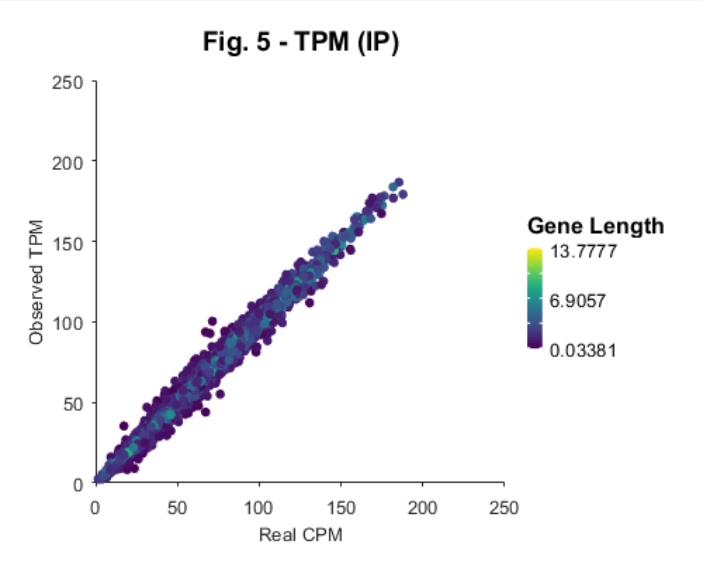
```
g.set_continuous_color;
g.geom_point;
g.set_names('x','Gene quantity','y','Read count','color','Gene Length');
g.set_title('Fig. 4 - Simulated Reads (IP)');
g.axe_property('XLim',[0 20],'YLim',[0 5000],'tickdir','out');
g.draw;
```

% GENERATE FIGURE 4




```
figure('Position',[1 1 500 400]);
g = gramm('x',1e6 * gene_quantity_IP / sum(gene_quantity_IP),'y',TPM_IP, 'color', gene_length);
g.set_continuous_color;
g.geom_point;
g.set_names('x','Real CPM','y','Observed TPM','color','Gene Length');
g.set_title('Fig. 5 - TPM (IP)');
g.axe_property('XLim',[0 250],'YLim',[0 250],'tickdir','out');
g.draw;
```

% GENERATE FIGURE 5



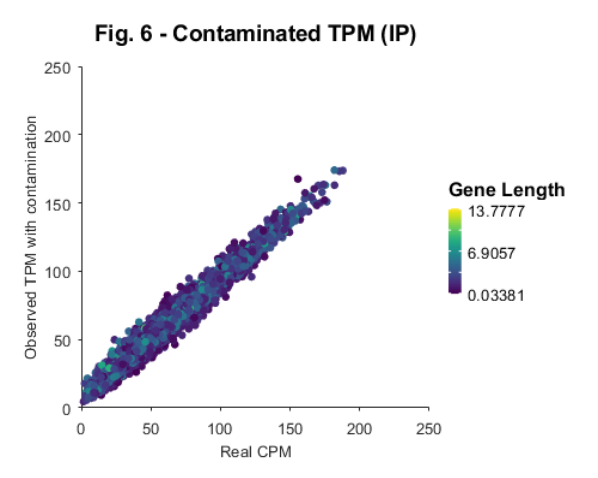

```
r_squared = corr(TPM_IP,gene_quantity_IP)^2;
```

r_squared = 0.9881

Contaminating the IP Sample with Gene Counts from the Input

Now we can contaminate our "true" IP gene counts with gene counts from the Input sample. Below we will assume that the mean level of Input contamination from our actual samples (~12%) is mixed in with the IP genes, and see how it affects the sequencing run and our estimate of TPM.

```
contamination = Mean_Contamination; % Mean observed contamination
% Contaminate the IP sample with Input genes
read_probability_IP_contaminated = read_probability_IP*(1-contamination) + read_probability_IN*contamination;
% Generate 1e7 sequencing counts based on the read probability of each gene
read_count_IP_contaminated = poissrnd(number_of_reads * read_probability_IP_contaminated);
TPM_IP_contaminated = calculateTPM(read_count_IP_contaminated, gene_length);
figure('Position',[1 1 500 400]);
g = gramm('x', 1e6 * gene_quantity_IP / sum(gene_quantity_IP),'y', TPM_IP_contaminated, 'color', gene_length);
g.geom_point;
g.set_names('x','Real CPM','y','Observed TPM with contamination','color','Gene Length');
g.set_title('Fig. 6 - Contaminated TPM (IP)');
g.axe_property('XLim',[0 250],'YLim',[0 250],'tickdir','out');
g.draw;
% GENERATE FIGURE 6
```

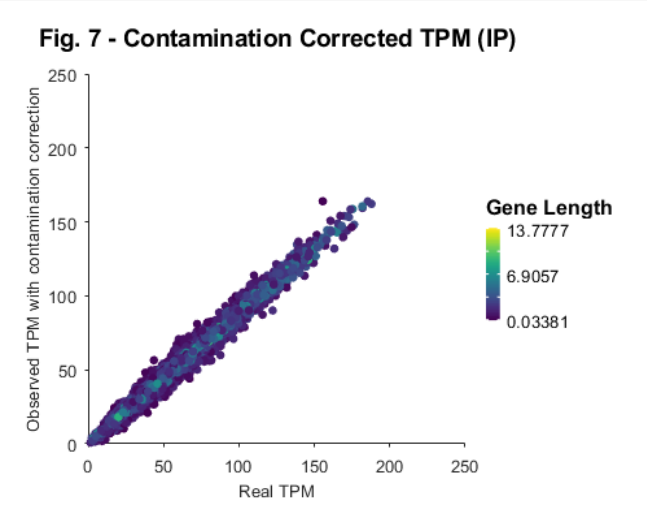



```
r_squared = corr(TPM_IP_contaminated,gene_quantity_IP)^2; r_squared = 0.9669
```

Removing Input Contamination from IP TPM

The contamination in our IP sample means that the TPMs we calculate from our modeled sequencing no longer perfectly reflect the underlying "true" gene counts. To fix this we can sequence the Input sample and simply subtract the proprtion of reads that are present due to non-specific RNA contamination.

```
% Subtract TPM present due to Input contamination.
TPM_IP_contaminated_corrected = TPM_IP_contaminated - contamination*TPM_IN;
figure('Position',[1 1 500 400]);
g = gramm('x', 1e6 * gene_quantity_IP / sum(gene_quantity_IP),...
    'y', TPM_IP_contaminated_corrected, 'color', gene_length);
g.geom_point;
g.set_names('x','Real TPM','y','Observed TPM with contamination correction','color','Gene Length');
g.set_title('Fig. 7 - Contamination Corrected TPM (IP)');
g.axe_property('XLim',[0 250],'YLim',[0 250],'tickdir','out');
g.draw;
% GENERATE FIGURE 7
```

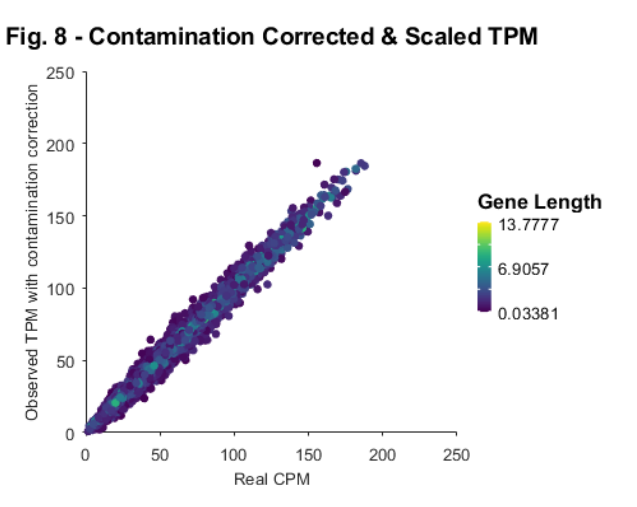


r_squared = corr(TPM_IP_contaminated_corrected,gene_quantity_IP)^2; **r_squared = 0.9847**

Scaling Corrected IP TPM

Pure subtraction of TPM will lower the number of reads bellow 1e6 and some small fraction of TPM will become negative. By setting negative reads to zero, and rescalling TPM to 1e6 reads, we can restore our corrected TPM to a nearly perfect estimate of the true underlying CPM.

```
TPM_IP_contaminated_corrected = TPM_IP_contaminated - contamination*TPM_IN ;
TPM_IP_contaminated_corrected(TPM_IP_contaminated_corrected<0)=0;
TPM_IP_contaminated_corrected = ...
   TPM_IP_contaminated_corrected/sum(TPM_IP_contaminated_corrected)*1000000;
figure('Position',[1 1 500 400]);
g = gramm('x', 1e6 * gene_quantity_IP / sum(gene_quantity_IP),...
   'y', TPM_IP_contaminated_corrected, 'color', gene_length);
g.geom_point;
g.set_names('x','Real CPM','y','Observed TPM with contamination correction','color','Gene Length');
g.set_title('Fig. 8 - Contamination Corrected & Scaled TPM');
g.axe_property('XLim',[0 250],'YLim',[0 250],'tickdir','out');
g.draw;
% GENERATE FIGURE 8
```

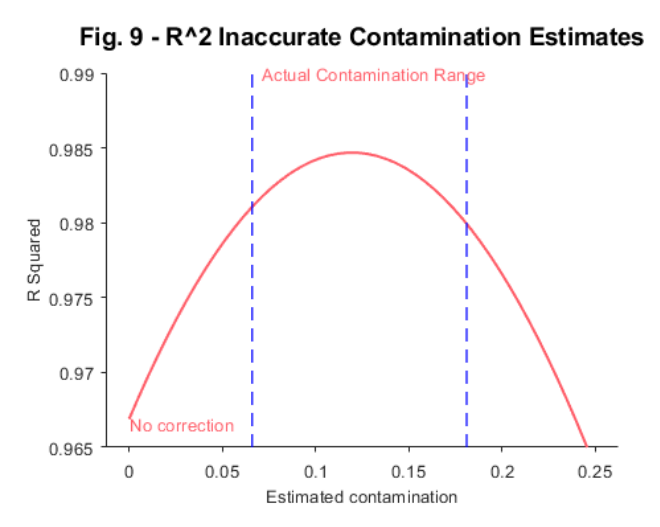


r_squared = corr(TPM_IP_contaminated_corrected,gene_quantity_IP)^2; **r_squared = 0.9847**

Benefit of Correction Given a Contamination Estimate Range

Contamination for each sample is estimated from our negative controls. This means there is a risk of over or under estimating contamination in any given sample. However, correcting for contamination anywhere within our actual range is better than no correction at all. In all RiboTag RNA-Seq experiments, large changes in the Input sample could be responsible for changes in the IP samples. Using our current experiment as an example, if morphine withdrawal causes a 30-fold increase in a particular gene in the Input, a 10% contaminated IP sample would show a 3-fold increase. Using this technique, we can reliably remove known changes in the Input samples from the IP samples, markedly improving the reliability and interpretability of our RiboTag RNA-seq results.

```
r_squared = [];
contamination_values = linspace(0, .25, 50);
for i = 1:length(contamination_values)
   TPM_IP_contaminated_corrected = TPM_IP_contaminated - contamination_values(i)*TPM_IN;
   r_squared(i) = corr(TPM_IP_contaminated_corrected,gene_quantity_IP)^2;
end
figure('Position',[1 1 500 400]);
clear g;
g = gramm('x', contamination_values, 'y', r_squared);
g.geom_line;
g.set_names('x','Estimated contamination','y','R Squared');
g.set_title('Fig. 9 - R^2 Inaccurate Contamination Estimates');
g.geom_vline('xintercept',min(Estimated_Contamination),'style','b--');
g.geom_vline('xintercept',max(Estimated_Contamination),'style','b--');
g.draw;
g.update('x',[0, contamination-.05],'y',[r_squared(1)-.0003,.99],...
   'label',{'No correction','Actual Contamination Range'});
g.axe_property('YLim',[.965 .99],'tickdir','out');
g.geom_label;
g.draw;
% GENERATE FIGURE 9
```



TPM Calculation Function

```
function TPM = calculateTPM(reads, gene_length)
TPM = reads ./ gene_length;
TPM = 1e6 * TPM / sum(TPM);
end
```

Supplementary Table S1. Sheen Green Network Central Genes

Gene Name	Module	Centrality	Full Name	Description
Smpd3	Sheen Green	0.004772383	Sphingomyelin phosphodiesterase 3	Catalyzes the hydrolysis of sphingomyelin to form ceramide and phosphocholine. Ceramide mediates numerous cellular functions, such as apoptosis and growth arrest, and can regulate these 2 cellular events independently.
Cacna2d3	Sheen Green	0.004760259	Calcium Channel, Voltage-Dependent, Alpha 2/Delta Subunit 3	The alpha-2/delta subunit of voltage-dependent calcium channels regulates calcium current density and activation/inactivation kinetics of the calcium channel.
Phactr1	Sheen Green	0.004750367	Phosphatase and Actin Regulator 1	Binds actin monomers (G actin) and plays a role in multiple processes including the regulation of actin cytoskeleton dynamics, actin stress fibers formation, cell motility and survival, formation of tubules by endothelial cells, and regulation of PPP1CA activity. Involved in the regulation of cortical neuron migration and dendrite arborization.
Syndig1l	Sheen Green	0.004750238	Synapse Differentiation Inducing 1 Like	May regulate AMPA receptor content at nascent synapses and have a role in postsynaptic development and maturation. Expressed highly in human microglia.
Arhgef9	Sheen Green	0.004745868	Cdc42 Guanine Nucleotide Exchange Factor 9	The protein encoded by this gene is a Rho-like GTPase that switches between the active (GTP-bound) state and inactive (GDP-bound) state to regulate CDC42 and other genes. This brain-specific protein also acts as an adaptor protein for the recruitment of gephyrin and together these proteins facilitate receptor recruitment in GABAergic and glycinergic synapses.
Pdyn	Sheen Green	0.00474121	Prodynorphin	The protein encoded by this gene is a preproprotein that is proteolytically processed to form the secreted opioid peptides beta-neoendorphin, dynorphin, leu-enkephalin, rimorphin, and leumorphin. These peptides are ligands for the kappa-type of opioid receptor.
Jcad	Sheen Green	0.004740184	Junctional Cadherin 5 Associated	This gene encodes an endothelial cell-to-cell junction protein. Also highly expressed in mouse brain tissue.
Pde10a	Sheen Green	0.004735912	Phosphodiesterase 10A	The protein encoded by this gene belongs to the cyclic nucleotide phosphodiesterase family. It plays a role in signal transduction by regulating the intracellular concentration of cyclic nucleotides. This protein can hydrolyze both cAMP and cGMP to the corresponding nucleoside 5' monophosphate, but has higher affinity for cAMP, and is more efficient with cAMP as substrate.
Itpka	Sheen Green	0.004729103	Inositol-Trisphosphate 3-Kinase A	Regulates inositol phosphate metabolism by phosphorylation of second messenger inositol 1,4,5-trisphosphate to Ins(1,3,4,5)P4. It is also a substrate for the cyclic AMP-dependent protein kinase, calcium/calmodulin- dependent protein kinase II, and protein kinase C in vitro.
Gldc	Sheen Green	0.004728291	glycine decarboxylase	Degradation of glycine is brought about by the glycine cleavage system, which is composed of four mitochondrial protein components: P protein (a pyridoxal phosphate-dependent glycine decarboxylase), H protein (a lipoic acid-containing protein), T protein (a tetrahydrofolate-requiring enzyme), and L protein (a lipoamide dehydrogenase).