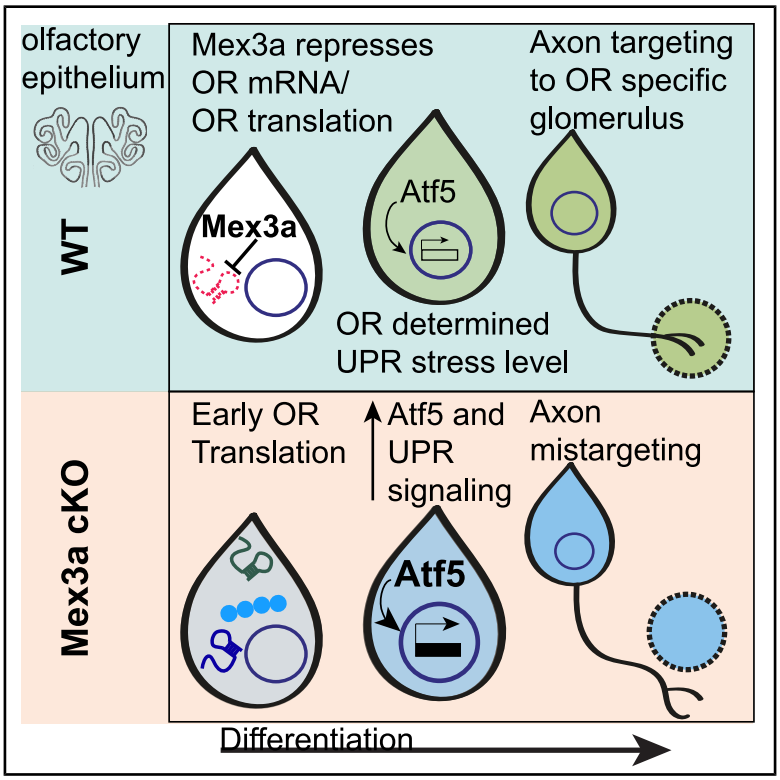
Cell Reports

Article

Mex3a-dependent post-transcriptional silencing ensures olfactory receptor diversity and axon guidance specificity

Graphical abstract



Highlights

- Mex3a is expressed in immature OSNs when multiple ORs are expressed per cell
- Exogenous expression of Mex3a leads to robust repression of OR mRNA and protein
- Mex3a conditional knockout leads to early OR translation and increased UPR signaling
- Mex3a conditional knockout results in skewed OR choice and axon pathfinding defects

Authors

Rachel Duffié, Hani Shayya, Martín Escamilla del Arenal, ..., Eduard Batlle, Marko Jovanovic, Stavros Lomvardas

Correspondence

sl682@cumc.columbia.edu

In brief

Olfactory receptor (OR) expression is refined from multiple to one OR, enabling axon targeting to an OR-specific glomerulus. Polygenic OR translation during differentiation would disrupt precise axonal targeting. We show that Mex3a, an RNA binding and ubiquitin ligase protein, represses ORs in immature neurons preventing axon mistargeting in mature neurons.

Duffié et al., 2025, Cell Reports *44*, 115979
August 26, 2025
https://doi.org/10.1016/j.celrep.2025.115979

CellPress

Cell Reports

CellPress
OPEN ACCESS

Article

Mex3a-dependent post-transcriptional silencing ensures olfactory receptor diversity and axon guidance specificity

Rachel Duffié,[1] Hani Shayya,[1] Martín Escamilla del Arenal,[1] Miao Wang,[1] Jerome Kahiapo,[1] Aileen Ugurbil,[1] Abdurrahman Keskin,[2] Fiona Clowney,[1] Elizaveta V. Bashkirova,[1] Ariel D. Pourmorady,[1] Ira Schieren,[1] Humberto Ibarra Avila,[1] Luke E. Berchowitz,[3] Francisco M. Barriga,[4,5] Eduard Batlle,[5,6,7] Marko Jovanovic,[2] and Stavros Lomvardas[1,8,9,10,*]

[1]Mortimer B. Zuckerman Mind, Brain, and Behavior Institute, Columbia University, New York, NY 10027, USA
[2]Department of Biological Sciences, Columbia University, New York, NY 10027, USA
[3]Department of Genetics and Development, Columbia University, Vagelos College of Physicians and Surgeons, New York, NY 10032, USA
[4]Systems Oncology Program, Vall d'Hebron Institute of Oncology (VHIO), Vall d'Hebron Barcelona Hospital Campus, 08035 Barcelona, Spain
[5]Institute for Research in Biomedicine (IRB Barcelona), The Barcelona Institute of Science and Technology (BIST), Baldiri i Reixac 10, 08028 Barcelona, Spain
[6]Centro de Investigación Biomédica en Red de Cáncer (CIBERONC), Barcelona, Spain
[7]Institució Catalana de Recerca i Estudis Avançats (ICREA), Barcelona, Spain
[8]Kavli Institute for Brain Sciences at Columbia University, New York, NY 10027, USA
[9]Department of Biochemistry and Molecular Biophysics, Vagelos College of Physicians and Surgeons, Columbia University, New York, NY 10032, USA
[10]Lead contact
*Correspondence: sl682@cumc.columbia.edu
https://doi.org/10.1016/j.celrep.2025.115979

SUMMARY

Olfactory sensory neurons (OSNs) use olfactory receptor (OR)-specific patterns of endoplasmic reticulum (ER) stress to transform OR sequence identity into axon guidance precision. However, during neuronal differentiation, OSNs transiently co-express random combinations of OR genes, which could generate unpredictable ER stress signatures that could lead to axon miswiring. Here, we show that post-transcriptional OR silencing by the transiently expressed and cytoplasmic RING and KH domain protein Mex3a, decouples OR transcription from OR protein-induced ER stress, until the onset of singular OR transcription. Consequently, conditional Mex3a deletion results in premature ER stress during the polygenic stage of OR transcription, which biases OR choice toward the OR alleles that are transcribed first and perturbs the specificity of OR-regulated axon targeting, disrupting the glomerular map of odor representation in the olfactory bulb. Our experiments reveal the critical role of post-transcriptional gene regulation in a fundamental cellular pathway that influences the assembly of neuronal circuits.

INTRODUCTION

Vertebrate olfactory systems are governed by the "one receptor per neuron" and the "one receptor per glomerulus" principles that transform singular olfactory receptor (OR) gene choice into axon guidance specificity. OR DNA, mRNA, and protein sequences contain regulatory information that enables both governing principles: OR DNA coding sequences (CDSs) promote their own transcriptional silencing,[1] OR mRNAs enforce transition from polygenic to singular OR transcription,[2] OR proteins stabilize OR gene choice,[3–6] promote olfactory sensory neuron (OSN) differentiation,[3] and instruct OSN axon guidance specificity.[7–9] While OR DNA and nascent OR RNA act in the nucleus, inducing heterochromatic silencing,[10,11] genomic compartmentalization,[12,13] and assembly of multi-chromosomal enhancer hubs,[14–20] OR protein elicits its regulatory function through the endoplasmic reticulum (ER) via Perk induction and eukaryotic translation initiation factor 2A phosphorylation.[3] Any level of Perk induction upon a threshold suffices for Atf5 translation and stabilization of singular OR choice.[3,21] OR sequence-dependent variations of Perk signaling, however, determine axon targeting specificity through Ddit3-dependent regulation of axon guidance genes.[7] Thus, quantitative differences in Perk induction,[7] together with the distinct activity properties of each OR,[22,23] transform stochastic OR gene choice into a stereotypic map of OR representation in the brain.

Any process that leverages the identity of the OR protein to instruct axon guidance precision should rely on singular OR expression or invariant OR co-expression between OSNs that project to the same glomerulus. However, immediate neuronal precursors (INPs) and early immature OSNs (iOSNs) co-transcribe diverse repertoires of OR genes.[18,20,24,25] Thus, neither

Perk signaling[7] nor neuronal activity[22,23] can fine-tune axon guidance in differentiating OSNs, as these will be significantly variable between cells that eventually choose the same OR. Indeed, even though Atf5 is highly transcribed at the earliest stage of OSN differentiation, Perk-dependent Atf5 translation coincides with the transition to singular OR transcription. Thus, a post-transcriptional mechanism must prevent co-transcribed ORs from inducing Perk signaling, to insulate OSN axon guidance from OR protein identity until the transition to transcriptional singularity is complete. On this note, it was previously proposed that unique features of the 5′ and 3′ untranslated regions (UTRs) may subject the OR superfamily to post-transcriptional regulation,[26] while more recently it was shown that mutations in components of the mRNA decay pathway alter the OR repertoire expressed in the mutant main olfactory epithelia (MOEs).[27] A direct demonstration of post-transcriptional OR silencing in the developing MOE, however, has yet to be shown.

Here, we identify Mex3 RNA-binding family member A (Mex3a), a KH and RING domain-containing protein[28] as a post-transcriptional repressor of OR expression. We show that Mex3a interacts with several RNA binding proteins involved in post-transcriptional silencing, mRNA stability, and translational inhibition. While we do not detect direct Mex3a-OR mRNA interactions, we reveal that the OR CDS is required for Mex3a-mediated OR silencing. We also show that efficient OR silencing is dependent upon an intact Mex3a RING domain, suggesting a pleotropic role of Mex3a in OR silencing that could reduce OR mRNA stability, OR mRNA translation, and, possibly, OR protein stability. Furthermore, loss-of-function experiments propose that the endogenous Mex3a protein promotes post-transcriptional OR silencing in INPs and iOSNs, reducing Perk signaling during the polygenic and oligogenic phases of OR transcription. Consequently, Mex3a deletion causes heterochronic induction of the OR protein feedback, which could enable co-transcribed ORs to influence axon guidance. The former alters OR gene choice patterns in the MOE, while the latter perturbs the precision by which like axons converge to distinct glomeruli in the olfactory bulb. Our experiments uncover a post-transcriptional layer of OR gene regulation, demonstrate its importance for the diversification and wiring of olfactory neurons, and propose previously unappreciated regulatory roles for Mex3a.

RESULTS

Mex3a silences OR expression in olfactory neurons

We hypothesized that polygenic OR transcription cannot elicit OR protein feedback or influence OSN axon guidance. OR transcription could be decoupled from OR protein expression if OR mRNA trafficking to the cytoplasm were impaired, if OR mRNA stability were reduced, or if OR mRNA translation were inhibited. We compared single nucleus (sn) and single-cell (sc) RNA sequencing (RNA-seq) experiments[2] to ask whether any of these processes operate in the MOE. Consistent with previous reports, scRNA-seq shows a strong increase in OR mRNA abundance from polygenic (INPs), to oligogenic (iOSNs), and eventually monogenic (mOSNs) stages of OR transcription. In contrast, snRNA-seq detects similar amounts of total nuclear OR RNA between INPs and iOSNs, and only a small increase in mOSNs (Figure 1A). Moreover, bulk RNA-seq experiments from fluorescence-activated cell sorting (FACS)-sorted cells from the MOE reveal that the intron/exon ratio of total OR reads is highest in INP cells and decreases with differentiation (Figure S1A). These observations raise the possibility that OR mRNA stability is reduced in INPs, and less so in iOSNs, compared to mOSNs. We therefore searched for post-transcriptional silencers with strong expression in INPs that are reduced in iOSNs and turned off in mOSNs. Post-transcriptional repressor Mex3a, one of four highly homologous RNA binding proteins,[28] exhibits this pattern, as it has the highest mRNA expression in INPs that declines in iOSNs and is absent in mOSNs (Figure 1B). Immunofluorescence (IF) experiments confirm that Mex3a protein is abundant in the cytoplasm of OSN progenitors and iOSNs but undetectable in Vglut2-expressing mOSNs (Figure 1C).

Mex3a promotes renewal and maintenance of intestinal[29,30] and neuronal stem cells,[31,32] while ectopic Mex3a expression in cancers is associated with poor prognosis.[33] Mex3a expression in the MOE persists in post-mitotic iOSNs, suggesting a wider function than stem cell maintenance. Reasoning that Mex3a downregulation in mOSNs is essential for OR expression, we sought to restore Mex3a expression in mOSNs. We generated a tetracycline-controlled transactivator (tTA)-inducible Mex3a transgene (tetO-Mex3a-t2a-mCherry, referred to as Mex3a Tg henceforth), which we crossed to an OMP-IRES-tTA driver[34] (called OMP-tTA) (Figure 1D). RNA-seq analysis comparing FACS-sorted OSNs that express the Mex3a Tg vs. tetO-GFP[35] (referred to as GFP Tg), revealed widespread downregulation for most OR genes, 42% of which are significantly downregulated (Figure 1E). In aggregate, we find a significant reduction of all the OR mRNAs in Mex3a Tg mOSNs, which is not observed for the rest of the mOSN transcriptome (Figure 1F). Since OR expression influences the expression of numerous mOSN markers,[3,7] transcriptional changes that give Mex3a-expressing OSNs a less differentiated signature (Figure S1B) may be a direct consequence of OR downregulation. IF with antibodies that detect OR proteins Mor28 (Olfr1507), C6 (Olfr49), M71 (Olfr151), and P2 (Olfr17) confirmed the widespread and significant downregulation of OR expression in Mex3a Tg MOEs (Figures 1G and 1H). The few remaining OR-expressing neurons in Mex3a Tg MOEs have significantly lower OR IF signal than control MOEs (Figure S1C). Consistent with this, activation of Mex3a expression earlier in OSN differentiation, with two developmentally overlapping tTA drivers, Gg8-tTA[1] and OMP-tTA, induces even stronger effects on OR transcription, with 72% of OR genes significantly downregulated, and near-complete loss of OR immunoreactivity (Figure 1H and S1D-S1F). Intriguingly, comparison of the effects of ectopic Mex3a expression on OR mRNA vs. OR intron sequences shows significantly stronger downregulation of OR mRNAs than their intronic sequences (Figure 1I), supporting a post-transcriptional and cytoplasmic mechanism for OR mRNA silencing, followed by a secondary transcriptional downregulation caused by depletion of the OR mRNA[17] and protein.[3]

Mex3a interactome identifies proteins with known post-transcriptional roles

To obtain molecular insight into the mechanisms of Mex3a-induced OR silencing, we performed immunopurification

CellPress
OPEN ACCESS

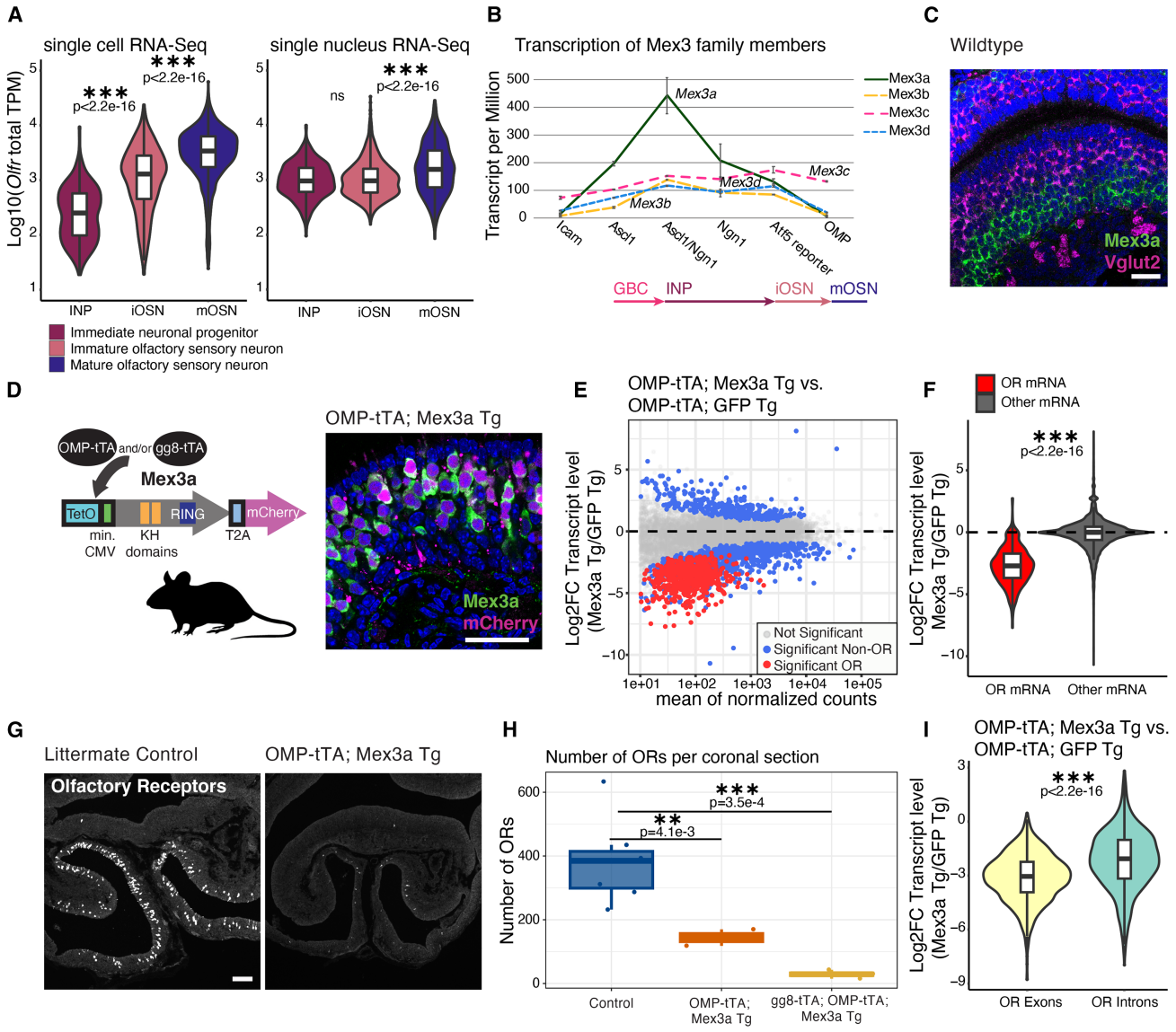


Figure 1. Mex3a silences OR expression in olfactory neurons

(A) OR mRNA levels quantified over olfactory neuronal lineage by scRNA-seq (left) and snRNA-seq (right). Wilcoxon rank-sum test. Data from Pourmorady et al.[17]

(B) Mex3 family members *Mex3a*, *Mex3b*, *Mex3c*, and *Mex3d* expression patterns by bulk RNA-seq of two replicates each of FACS-sorted cells over olfactory neuronal differentiation. Error bars, standard error. See also Table S1.

(C) Representative immunofluorescence (IF) image from >10 biological replicates in wild-type (WT), 4-week-old olfactory epithelium, Mex3a (green), neuronal marker Vglut2 (magenta), and DAPI (blue). Scale bar, 30 μM.

(D) Diagram of Mex3a Tg allele (left). IF in OMP-tTA, Mex3a Tg olfactory epithelium, Mex3a (green), mCherry (magenta), and DAPI (blue) (right). Scale bar, 30 μM.

(E) MA plot of differentially expressed genes from bulk RNA-seq of mCherry or GFP$^+$ sorted cells from OMP-tTA, Mex3a Tg and OMP-tTA, and GFP Tg MOEs, respectively. Significant ORs in red, significant non-OR mRNAs in blue. DESeq2, padj < 0.05, log2 fold change threshold, 0.5849.

(F) Violin plot quantifying all OR mRNAs and non-OR mRNAs (RNA-seq) from OMP-tTA, Mex3a Tg compared to OMP-tTA, and GFP Tg. Welch's two-sample t test.

(G) IF for OR proteins C2, M71, Mor28, and P2 in OMP-tTA; Mex3a Tg (right) and littermate control (left). ORs in white, background fluorescence delineates contours of olfactory tissue. Scale bar, 100 μM.

(H) Number of ORs detected by IF per MOE section. *n* = two or more biological replicates per condition. Welch's two-sample t test.

(I) Violin plot quantifying OR exonic and intronic reads in bulk RNA-seq from OMP-tTA; Mex3a Tg compared to OMP-tTA; GFP Tg. Wilcoxon rank-sum test.

For all statistical tests in this figure: $^*p < 0.05$; $^{**}p < 0.01$; $^{***}p < 0.001$.

See also Figure S1.

followed by mass spectrometry (IP-MS), using lysates from the MOE (Figure 2A). Mex3a is technically challenging to IP,[32] so we compared replicates from multiple experimental conditions to determine high confidence interactors (Figure 2B). We generated an additional Mex3a Tg line, discussed in detail below, with a mutant RING domain and V5 tag[36] to facilitate IP (Figures S2A and S2B). For each experiment we used a Mex3a-specific or V5-specific antibody and an immunoglobulin G (IgG) control and quantified peptide enrichment in each IP (Figures 2B and S2B). To exclude proteins that cross-react with the Mex3a antibody, we generated a conditional Mex3a knockout (Mex3a cKO) (Figure 2C) (Foxg1iresCre[37]; Mex3a fl/fl[38]). Mex3a immunoreactivity was abolished from Mex3a cKO MOEs (Figure 2D), as was detection of a protein band with the expected Mex3a size in western blot from MOE lysates (Figure 2E). For each genotype, we searched for peptides with a log2 fold change of 0.5 or greater enrichment in the Mex3a or V5 IP compared to the respective IgG IP. Putative interacting proteins also exhibited log2 fold change of 0.5 or greater enrichment than the Mex3a IP in Mex3a cKO MOE and had a Wilcoxon p value of less than 0.05 (Figures 2F and S2C). This analysis uncovered numerous RNA binding proteins with known roles in post-transcriptional silencing (Ago2, Tnrc6b), mRNA degradation (Cnot1, Upf1, Edc4), regulation of mRNA translation (Pum1 and 2, Cpeb4, Fmr1, Fxr2, Pabpc4), as well as numerous ribosomal proteins (Rpl and Rps family members). IP-MS in HEK293T cells transiently transfected with Mex3a constructs revealed enrichment for RNA binding proteins and ribosomal proteins compared to non-transfected controls, as observed in mouse. Truncated constructs demonstrate that KH domains are required for these protein-protein interactions (Figure S2D). RNase treatment of protein lysates reveals that ribosomal protein interactions do not depend on RNA interactions, while RNA binding protein interactors may be indirect via shared binding to RNA targets (Figure S2E). The Mex3a interactome is consistent with the hypothesis that Mex3a is involved in post-transcriptional OR silencing affecting both OR mRNA stability and OR mRNA translation, explaining why ectopic Mex3a expression has a stronger effect on exonic OR sequences.

Post-transcriptional silencing of ORs with the Mex3a Tg

To further distinguish between transcriptional and post-transcriptional Mex3a-dependent OR silencing, we devised a genetic strategy that would uncouple the two processes. Specifically, we crossed our Mex3a Tg mice to mice that express the transgenic OR M71, also under the control of the tetO promoter (tetO-M71iresLacZ,[39] referred as M71 Tg). In this scheme both Tgs have a tetO promoter and therefore are activated by the same transcription factor (tTA) (Figure 3A). If OR downregulation is induced by transcriptional OR silencing, the transgenic M71 should not be affected, resulting in frequent co-expression with Mex3a. However, if OR downregulation is post-transcriptional, then Mex3a should silence the co-transcribed M71 Tg. Consistent with the latter, M71 Tg expression frequency is dramatically reduced in the presence of the Mex3a Tg (Figures 3B, 3E, and S3A). Furthermore, the rare mOSNs that still express M71 protein in the double transgenics express greatly reduced levels of the M71 Tg (Figure 3E). Moreover, RNA fluorescence *in situ* hybridization (FISH) and northern blot using a probe against the M71 CDS show that M71 mRNA detection is also strongly reduced in Mex3a-expressing MOEs (Figures S3B–S3D). Mex3a suppresses the expression of another tetO-induced OR, the P2iresGFP knockin allele[17,40] when induced by OMP-tTA, or with both Gg8-tTA and OMP-tTA (Figures 3C–3E, S3E, and S3F). We performed RNA FISH against the intron of P2 in tetO-P2 knockin mice and tetO-P2 knockin Mex3a Tg animals (Figure S3G) and observe fewer nuclear puncta, indicating reduced transcription of the P2 allele in the presence of Mex3a, consistent with a role of the OR mRNA in continuous OR transcription.[2] In contrast to the M71 Tg and tetO-P2 knockin allele, we detect abundant co-expression between Mex3a and the tetO-driven GFP Tg that does not contain OR sequences (Figures 3F, 3G, and S3H), excluding the possibility for competition between transgenes for tTA recruitment as an explanation of the mutually exclusive expression pattern. Finally, it is worth noting that when we induce M71 or P2 by both Gg8-tTA and OMP-tTA, we observe reduced coexpression with endogenous Mex3a protein, which is naturally expressed in $Gg8^{+}$ cells. In contrast, endogenous Mex3a has significantly more frequent co-expression with the GFP Tg (Figures S3K and S3L), further supporting Mex3a selectivity for OR sequences. Thus, in a striking concurrence between transcriptomics, proteomics, and genetics, our data suggest that endogenous and ectopic Mex3a silence OR gene expression post-transcriptionally. Importantly, Mex3a mediated OR silencing has functional consequences on odor detection. We exposed Mex3a Tg mice and littermate controls to the odor acetophenone, which activates a large number of OSNs,[41] and then we performed IF for phosphorylated S6 ribosome protein as a marker of neuronal activity.[41,42] gg8-tTA; OMP-tTA;Mex3a Tg mice showed a dramatic reduction in the number of cells that are positive for p-S6, as well as the p-S6 fluorescence intensity compared to littermate controls that did not express transgenic Mex3a (Figures S6B–S6D), suggesting strong deficits in odor detection by Mex3a-expressing OSNs.

To obtain deeper mechanistic understanding of Mex3a-mediated OR silencing, we modified the only known enzymatic domain of this protein, the RING domain, which has E3 ubiquitin ligase activity.[33] Four distinct point mutations were introduced to the tetO-Mex3a-t2A-mCherry transgene, which disrupt the RING domain's cross-brace structure required for its interaction with zinc.[43] Driving this mutant Mex3a Tg (Figures S2A and S2B) (referred to as $Mex3a_{mut}$ Tg) together with the M71 Tg in mOSNs resulted in abundant expression of M71 and frequent co-expression of the M71 and mutant MEX3A proteins in the same cell (Figures 3H, 3I, and S3I), in striking contrast to the expression pattern for M71 with the wild-type (WT) Mex3a Tg (Figure 3B). In addition, FACS isolation followed by RNA-seq of mOSNs expressing the $Mex3a_{mut}$ Tg confirms a significantly reduced ability to silence OR expression genome-wide compared to the WT Mex3a Tg (Figure 3J). When compared to the GFP Tg, $Mex3a_{mut}$ Tg sorted cells show a downregulation of 18% of OR genes compared to 46% of ORs for the WT Mex3a Tg (Figure S3J compared to Figure 1E). Thus, Mex3a does not simply act as a scaffold protein that recruits known post-transcriptional repressors but also actively contributes to OR silencing via its enzymatic E3 ubiquitin ligase activity. Importantly, IP-MS

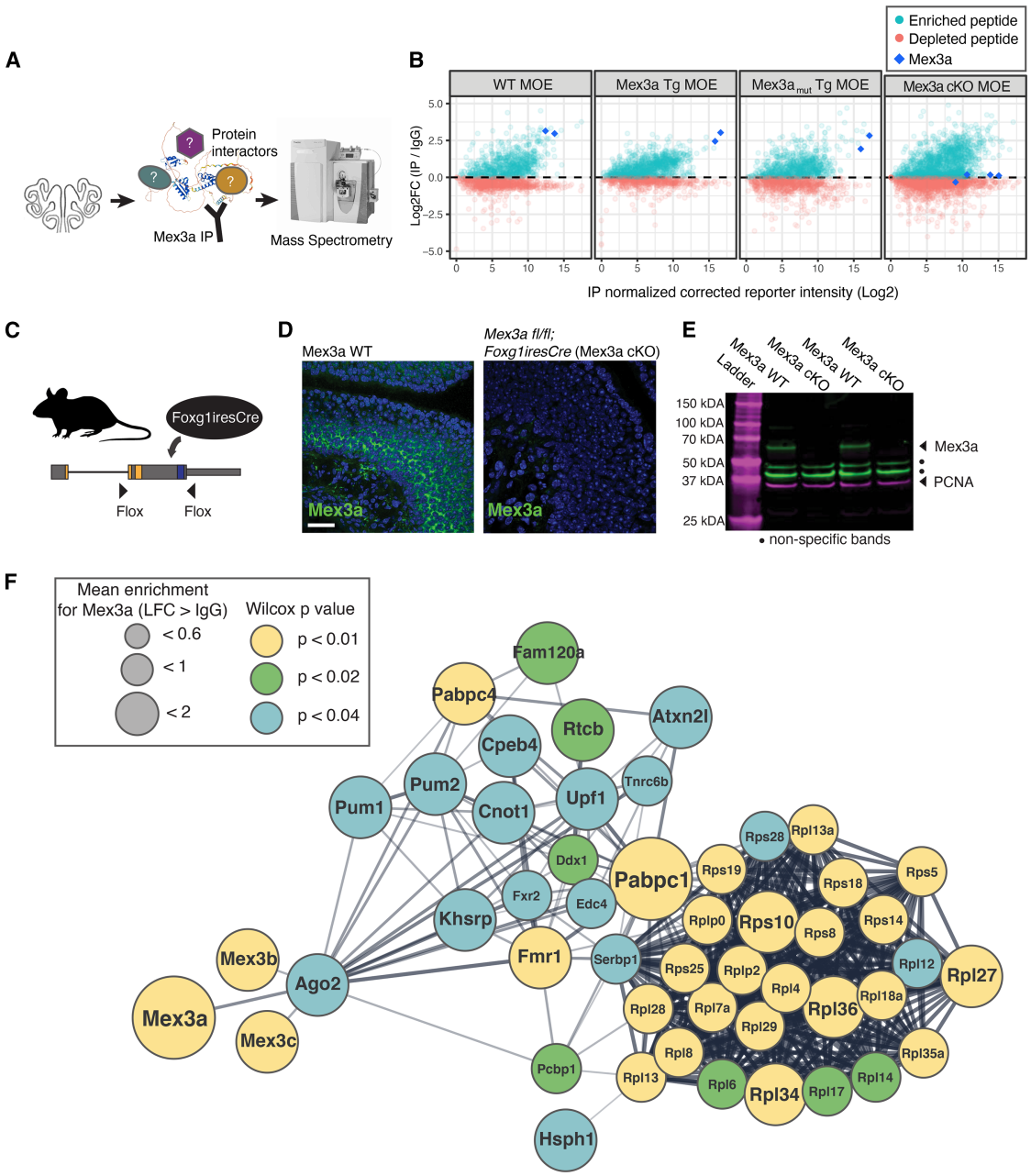


Figure 2. Mex3a interactome identifies proteins with known post-transcriptional roles

(A) Design for immunopurification-mass spectrometry experiment.

(B) Enriched peptides compared to IgG controls from the same protein samples across four genotypes. Two replicates for each genotype except for Mex3a cKO (four replicates). Mex3a peptides in blue.

(C) Mex3a cKO allele. CDS, thicker gray rectangle, 3′ UTR, thin gray rectangle. KH domains in orange and RING domain in blue. The first KH domain spans an intron (thin line). Flox sequences excise the CDS in the second exon.

(D) Representative IF image from >10 biological replicates in Mex3a cKO and littermate control, Mex3a (green). Scale bar, 25 μM.

(E) Western blot of two biological replicates of whole MOE at post-natal day 12 (PN12) for Mex3a (green) and loading control proliferating cell nuclear antigen (PCNA) (magenta). Mex3a predicted at 58 kDa; PCNA predicted at 29 kDa.

(F) STRING representation of Mex3a interacting proteins, generated with Cytoscape software. Log fold enrichment greater than IgG represented by size of circle; color represents *p* value relative to Mex3a cKO; weight of lines between proteins is based on STRING prediction of interaction.

See also Figure S2 and Table S2.

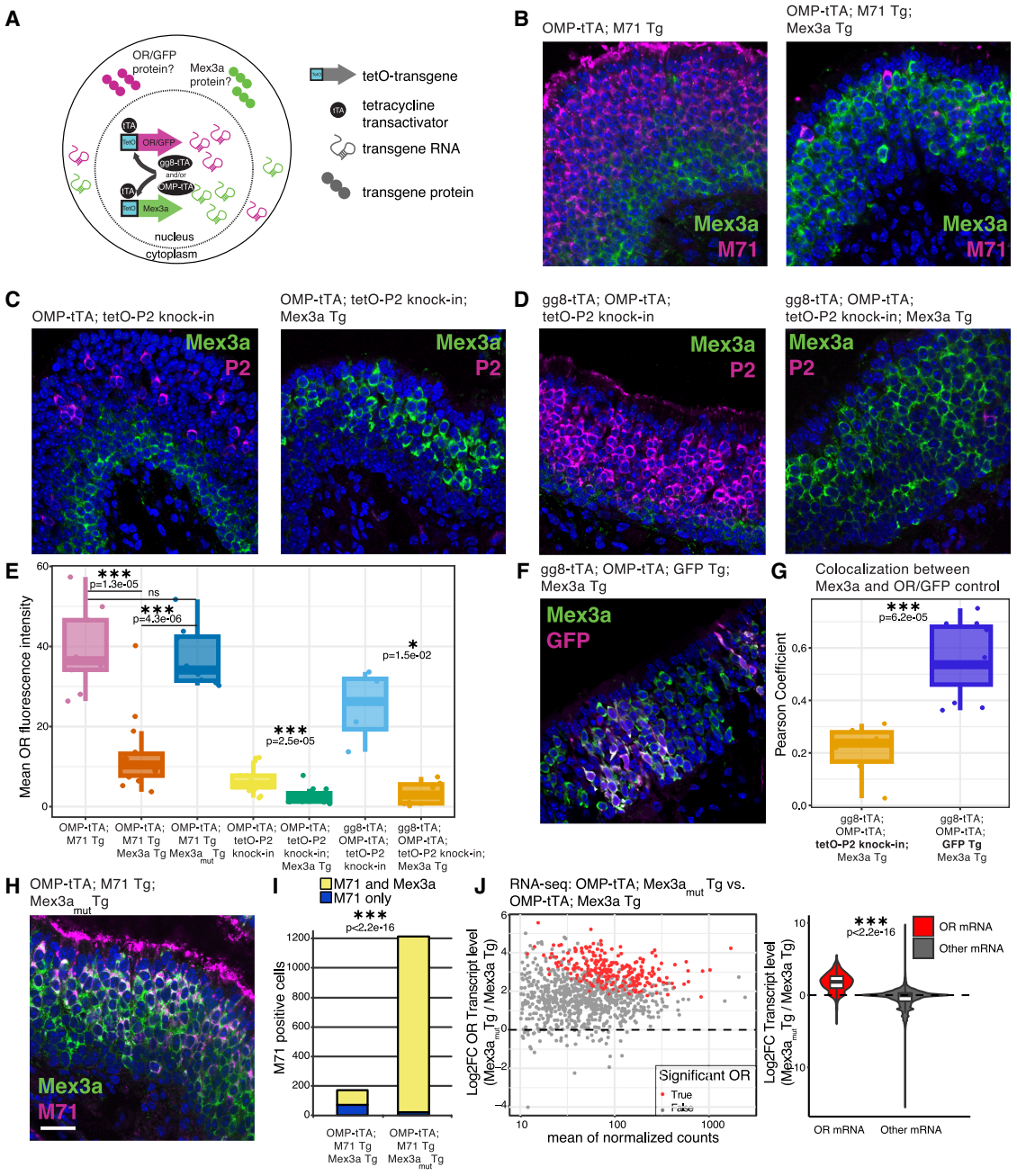


Figure 3. Post-transcriptional silencing of ORs with the Mex3a Tg

(A) Schematic depicting strategy to test post-transcriptional role of Mex3a.

(B) IF for Mex3a (green) and M71 (magenta) in (left) OMP-tTA and M71 Tg, and (right) OMP-tTA, M71 Tg, and Mex3a Tg.

(C) IF for Mex3a (green) and P2 (magenta) in OMP-tTA; tetO-P2 knockin on left, and right, OMP-tTA; tetO-P2 knockin; Mex3a Tg.

(D) Same as (C), except both genotypes include the gg8-tTA and OMP-tTA drivers.

(E) Boxplot of mean OR fluorescence intensity per image. Whiskers extend to most extreme values within 1.5 times the interquartile range; center line is the median. *n* = 2–4 biological replicates per condition, 4–16 images per condition. Welch's two-sample t test.

(F) IF for Mex3a (green) and GFP (magenta) in gg8-tTA, OMP-tTA, GFP Tg, and Mex3a Tg.

(G) Pearson correlations comparing colocalization between Mex3a and P2 fluorescent channels in the gg8-tTA, OMP-tTA, tetO-P2 knockin Mex3a Tg and Mex3a, and GFP fluorescent channels in the gg8-tTA, OMP-tTA, GFP Tg, and Mex3a Tg. Boxplot whiskers as in (E). Welch's two-sample t test.

(H) IF for Mex3a (green) and M71 (magenta) in OMP-tTA, M71 Tg, and Mex3a$_{mut}$ Tg. Scale bar, 25 μM (same scale for all images in the figure).

(legend continued on next page)

with the $Mex3a_{mut}$ Tg does not reveal significant alterations in the interactions between Mex3a and other RNA-binding proteins (Figure S2C), excluding the possibility that RING mutations altered the Mex3a interactome. We cannot exclude, however, the intriguing possibility that these presumed Mex3a co-factors silence OR mRNAs in a ubiquitination-dependent fashion.

Premature OR expression upon loss of Mex3a

Our analyses thus far have demonstrated that ectopic Mex3a expression in mOSNs induces post-transcriptional OR silencing that depends on a functional RING domain and likely occurs in coordination with a complex milieu of other post-transcriptional repressors. However, these results do not provide insight into the physiological role of Mex3a in the OSN progenitors that express the endogenous Mex3a protein. A direct expectation from our observations in the gain-of-function experiments is that loss of Mex3a activity will result in an increase in OR mRNA and protein in OSN progenitors, when Mex3a is expressed.

To test the effects of Mex3a deletion on OR expression, we turned our attention to the Mex3a cKO mice that we described in the IP-MS section (Figures 2C, 2D, and 2E). We first performed scRNA-seq in control and Mex3a cKO MOEs (Figures 4A, S4A, and S4B), with the goal of quantifying changes in OR mRNA abundance across differentiation. Overall, control and Mex3a cKO MOEs have almost identical cell-type composition (Figure S4C), suggesting that Mex3a deletion does not cause gross differentiation defects. However, consistent with the hypothesis that Mex3a represses OR expression during the polygenic state of OR transcription, we detect a significant increase in the number of cells expressing multiple ORs (Figure S4D) and the number of OR genes co-transcribed in Mex3a cKO INPs (Figures 4B and S4E). In addition, we find a significant increase in the number of total OR mRNAs detected in INPs and iOSNs (Figure 4C). These differences dissipate in mOSNs, consistent with a developmentally transient Mex3a role in OR mRNA stability that does not extend to the singular phase of OR transcription.

Because Mex3a interacts with translational regulators (Figure 2F), it is possible that it promotes both OR mRNA degradation and repression of OR mRNA translation. We thus performed ribosome profiling[44] in OSN progenitors and mOSNs from control and Mex3a cKO mice (Figures 4D–4F, S4F, and S4G). To isolate OSN progenitors, we crossed the Mex3a cKO to Ngn1-GFP Tg mice, and to isolate mOSNs, we crossed-in the OMP-GFP knockin allele. Ribosome profiling of sorted Ngn1-GFP$^+$ cells revealed increased OR mRNA translation upon deletion of Mex3a (Figures 4E and 4F), as measured by ribosome sequencing counts in the CDS relative to mRNA-seq counts in the same CDS.[44,45] The increase in translational efficiency observed for OR mRNAs in the Mex3a cKO Ngn1-GFP$^+$ cells is not generalized to the rest of the transcriptome (Figures 4E and 4F). Moreover, similar analysis in FACS-sorted, OMP-GFP$^+$ mOSNs, did not reveal differences in OR translational efficiency between control and Mex3a cKO (Figures S4F and S4G), further supporting a transient role for Mex3a in suppressing OR mRNA translation. IF with a pool of antibodies against the ORs P2 (Olfr17), C6 (Olfr49), and M71 (Olfr151) on MOE sections from control and Mex3a cKO Ngn1-GFP mice (Figure 4G) confirmed the ribosome profiling data as Ngn1-GFP intensity is significantly increased in OR$^+$ cells from the Mex3a cKO MOEs (Figure 4H). Moreover, OR$^+$ cells are detected closer to the basal layer of the MOEs of Mex3a cKO mice, confirming that these are OSN progenitors that initiated OR protein expression earlier in development (Figure 4I).Thus, while Mex3a destabilizes the OR mRNA, it also inhibits the translation of the OR mRNAs that were not degraded, consistent with the role of the first Mex3 protein isolated in *Caenorhabditis elegans*,[46] which blocked translation of its mRNA targets. However, we did not find evidence for direct binding of Mex3a on OR mRNAs, either computationally, by searching for enrichment of human and *C. elegans* MEX3 RNA binding motifs,[47–50] (Figures S3A and S3B) or by RNA IP and proximity labeling approached (data not shown). Thus, Mex3a likely binds OR mRNA via interactions with one of the many RNA-binding proteins it interacts with in the MOE.

Perk-induced transcription factor Atf5 and targets exhibit increased expression in the Mex3a cKO

We then explored the biological significance of post-transcriptional OR silencing by Mex3a. We reasoned that heterochronic OR translation could increase Perk signaling levels before the onset of singular OR transcription. We performed IF for transcription factor Atf5, which represents a bona fide marker for Perk signaling in OSNs.[3] Quantification of Atf5 immunoreactivity in MOE sections revealed a significant increase in Atf5 protein expression in Mex3a cKO mice, compared to control littermates (Figures 5A and 5B). To further explore the effects on Perk signaling, we generated two lists of genes and computed their module scores[51] in cells from our scRNA-seq experiment. We identified 318 genes from bulk RNA-seq experiments of sorted cells whose expression peaks in Atf5 translating cells (Figures S5C and S5D). Computing the module score for these genes, termed "Atf5 translating score," demonstrates a high score in iOSNs, as expected. To determine whether Mex3a cKO results in earlier Perk signaling, we compared the Atf5 translating scores of WT and Mex3a cKO INP cells, before Perk signaling should occur. We detected a significant increase in the Atf5 translating scores of Mex3a cKO INP cells (Figure 5C), supporting the hypothesis that early OR translation induces early Perk signaling. To explore Perk signaling strength further, we used a list of 272 genes that exhibit Atf5 chromatin IP-seq peaks near their promoters and are differentially expressed in iOSNs of Atf5 KO scRNA-seq,[52] termed "Atf5-dependent genes." We observed a significant increase in the expression of this class

(I) Quantification of M71$^+$ cells in mice from (A) and (G), with Mex3a co-expression (yellow) or M71 only (green). Fisher's exact test.
(J) Bulk RNA-seq from OMP-tTA, $Mex3a_{mut}$ Tg compared to OMP-tTA, and Mex3a Tg. (Left) MA plot of OR mRNAs. DESeq2, padj < 0.05, log2 fold change threshold, 0.5849. (Right) Violin plot comparing all OR mRNAs to non-OR mRNAs. Welch's two-sample t test.
For all statistical tests in this figure: *$p < 0.05$; **$p < 0.01$; ***$p < 0.001$.
See also Figure S3.

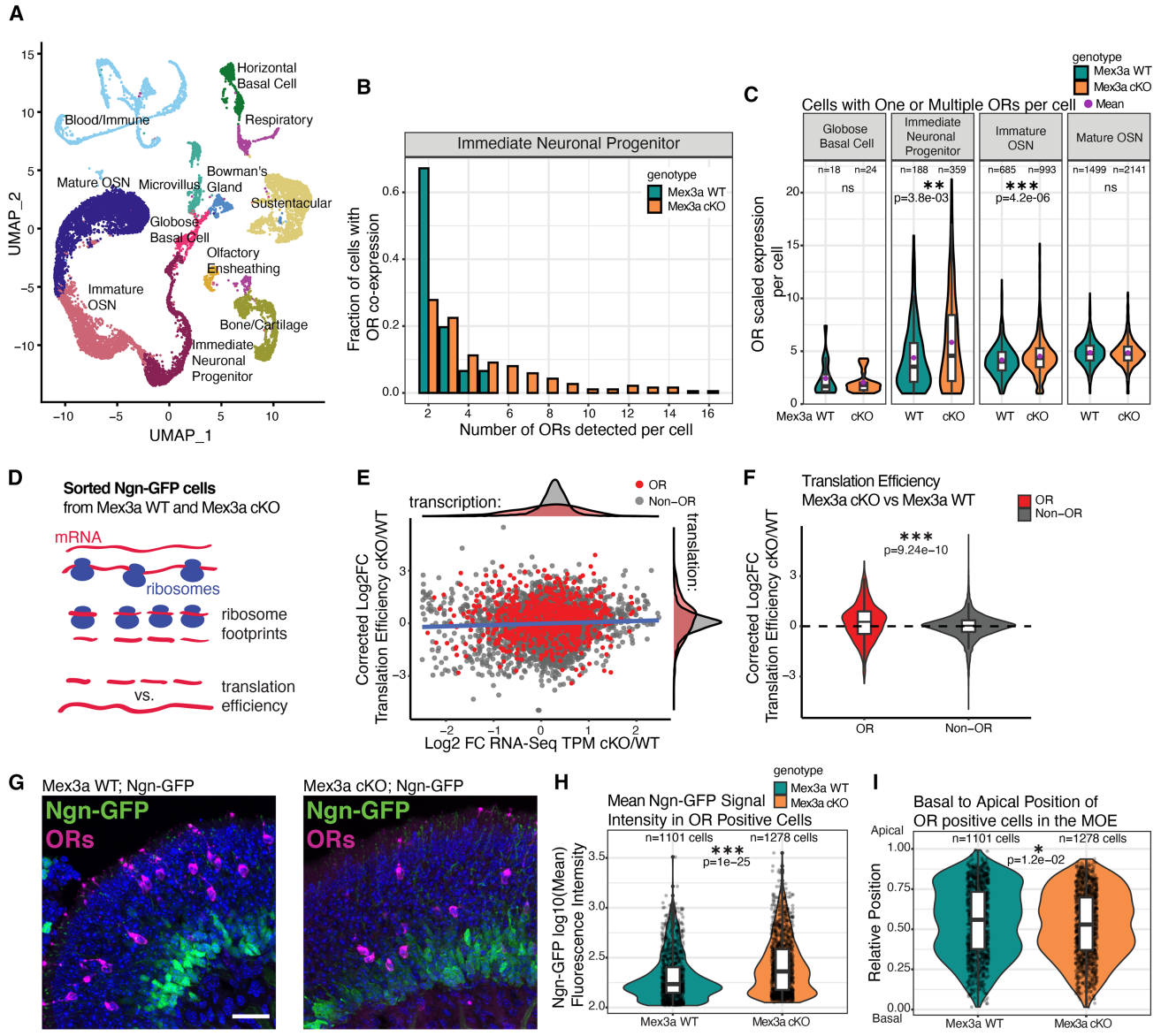


Figure 4. Premature OR expression upon loss of Mex3a

(A) Uniform manifold approximation and projection (UMAP) plot of scRNA-seq data from two replicates of Mex3a cKO and Mex3a WT littermate controls; whole MOE at PN12.

(B) Number of ORs expressed per INP cell expressing more than one OR with a threshold of >1 normalized counts in Mex3a WT and Mex3a cKO cells.

(C) OR mRNA levels in single cells from the neuronal lineage. *n* for each condition = number of cells with >1 normalized count for an OR. Wilcoxon rank-sum test with continuity correction.

(D) Schematic of the ribosome footprinting technique. Ribosomes in blue and mRNA in red.

(E) Scatterplot comparing log2 fold change translational efficiency (*y* axis) and log2 fold change RNA transcripts per million (*x* axis) in two replicates of Mex3a cKO vs. Mex3a WT Ngn1-GFP sorted cells, PN12 aged mice. Red dots, OR RNAs; gray dots, non-OR RNAs. Blue line, linear regression line. Marginal density plots summarize OR gene family (red) or all other genes (gray). Translational efficiency (*y* axis) or RNA expression levels (*x* axis).

(F) Violin plot quantifying translational efficiency of OR mRNAs (red) compared to non-OR mRNAs (gray) in two replicates of Mex3a cKO vs. Mex3a WT Ngn1-GFP sorted cells. Welch's two-sample t test.

(G) Representative IF from three biological replicates at 4 weeks old, Mex3a WT littermate controls, and Mex3a cKO. Endogenous Ngn1-GFP (green), OR protein (magenta), and DAPI (blue). A pool of M71, P2, and C6 OR antibodies were used. Images from Mex3a WT and cKO are from the same zonal position in the MOE. Scale bar, 25 μM.

(legend continued on next page)

of genes across OSN differentiation (Figures 5D and 5E). Conditional Mex3a deletion from the MOE causes increased OR mRNA detection, OR mRNA translation, and OR protein detection, resulting in stronger Perk signaling induction at a stage that precedes singular OR transcription.

Mex3a deletion biases OR gene choice and disrupts axon guidance specificity

The realization that Mex3a deletion accelerates OR protein expression and Perk induction raises questions about the functional consequences of these heterochronic changes. Our scRNA-seq data suggest that Mex3a cKO OSNs transition to singular OR transcription with the same efficiency as control OSNs (Figures 4C, S4D, and S4E). This is not surprising given what we know about the OR-elicited feedback and the process that promotes the transition to singular OR transcription. The OR-elicited feedback shuts off expression of lysine demethylase Lsd1, preventing both the activation of additional OR alleles (via H3K9 demethylation) and the silencing of the already chosen OR (via H3K4 demethylation).[21] Our recent work suggests that transition from polygenic to singular OR transcription is mediated by the non-coding functions of the nascent OR mRNA, and singular OR transcription precedes the OR-protein-elicited feedback signal.[17] Thus, while loss of OR-elicited feedback results in unstable OR gene choice, premature induction of this feedback in INPs or early iOSNs does not interfere with the stability and singularity of OR transcription in mOSNs. What, then, is the biological significance of delaying mRNA OR translation and Perk induction until a single OR is chosen?

We recently showed that OR genes are transcriptionally activated at different stages of OSN differentiation, with ORs expressed in ventral MOE regions (zones 4 and 5) detected in more differentiated cells than ORs with dorsal identities (zones 1–3[53]). We therefore asked whether premature induction of the OR-elicited feedback signal in Mex3a cKO mice disadvantages late-onset ORs. Indeed, scRNA-seq experiments from microdissected ventral MOE zones show that in Mex3a cKO MOEs, ventral ORs are less likely to be activated during differentiation (Figure 6A). Moreover, bulk RNA-seq of FACS-sorted mOSNs from the whole MOE showed reduced frequency of choice of ventral, zone 5 (and less so of zone 4) OR genes (Figure 6B). Thus, premature induction of Perk signaling may narrow the developmental window of OR transcriptional activation, making the transcription and future choice of "delayed" OR genes less likely. ORs in all zones exhibit reduced expression in the Mex3a Tg MOE (Figure S6A), while curiously, ventral zone 5 ORs are more significantly affected than ORs from other zones. These data show that ventral zone 5 ORs are particularly sensitive to proper levels of Mex3a protein during differentiation.

Beyond assuring that the whole OR repertoire can participate in the competitive process of OR gene choice, the precise developmental timing of OR protein expression and Perk induction likely contributes to axon guidance precision. Indeed, Ddit3, the Perk-induced transcription factor that transforms OR protein identity into distinct axon guidance signatures exhibits significantly increased transcription in Mex3a cKO immature OSNs (Figure 6C). This prompted us to analyze axonal projections in Mex3a cKO mice. We bred four OR-IRES-GFP lines (OR: Mor23, Mor28, M71, or P2) into the Mex3a cKO mice (Mex3a fl/fl; Foxg1iresCre; OR-IRES-GFP) and explored the effects of Mex3a deletion on glomerular targeting. For all four of these ORs, we observed a significant increase in abnormal axonal wiring to OR-specific stereotyped glomeruli, including missing glomeruli, mispositioned glomeruli, glomeruli with axons extending to other glomeruli of the same OR, and ectopic/microglomeruli (Figures 6D–6F and S6E). Thus, conditional Mex3a deletion not only results in premature Perk signaling and Ddit3 dysregulation but also in disruption of the peripheral olfactory circuit, highlighting the importance of post-transcriptional OR gene regulation for the proper development and function of the peripheral olfactory circuit.

DISCUSSION

Our experiments assign Mex3a as a post-transcriptional repressor of OR gene expression during mOSN differentiation. Mex3a expression is exquisitely restricted to INP and iOSN stages, when multiple ORs are expressed per cell. Our gain-of-function experiments demonstrate robust repression of OR RNA and protein, even when OR expression is driven with a tetO promoter. Loss-of-function experiments reveal increased OR mRNA per cell, increased OR translational efficiency, and early and more robust Perk signaling.

While we did not observe direct Mex3a binding on OR mRNAs, our experiments suggest that the OR CDS is required for post-transcriptional silencing by Mex3a. These results invite the exciting possibility that Mex3a does not target ORs by binding them directly but selectively represses them via a conserved layer of gene regulation that regulates OR translation. Our data invoke several hypotheses for this mechanism of action. Our proteomics analysis proposes that Mex3a recruits proteins with known functions in post-transcriptional gene silencing and mRNA destabilization to the OR mRNA. Moreover, interactions with inhibitors of mRNA translation and with ribosomal proteins suggest that Mex3a may also target OR mRNAs as they engage in translation, explaining why Mex3a deletion results in increased translational efficiency of ORs. Furthermore, the requirement for an intact RING domain for efficient OR silencing by Mex3a proposes three possible scenarios for the role of lysine ubiquitination. First, the activity of some of the Mex3a interacting proteins that contribute to post-transcriptional silencing may be increased by ubiquitination. Second, ubiquitination may cause

(H) Violin plot quantifying Ngn1-GFP fluorescence intensity in each OR^+ cell (dots) from the Mex3a WT (left) and cKO (right) IF experiment described in (G). n = number of OR^+ cells counted for each genotype. Wilcoxon rank-sum test.
(I) Violin plot quantifying relative position (for each image, the basal position of the MOE is set to 0 on the y axis, apical set to 1) of OR^+ cells (dots). n = number of OR^+ cells counted for each genotype. Wilcoxon rank-sum test.
For all statistical tests in this figure: $^*p < 0.05$; $^{**}p < 0.01$; $^{***}p < 0.001$.
See also Figure S4.

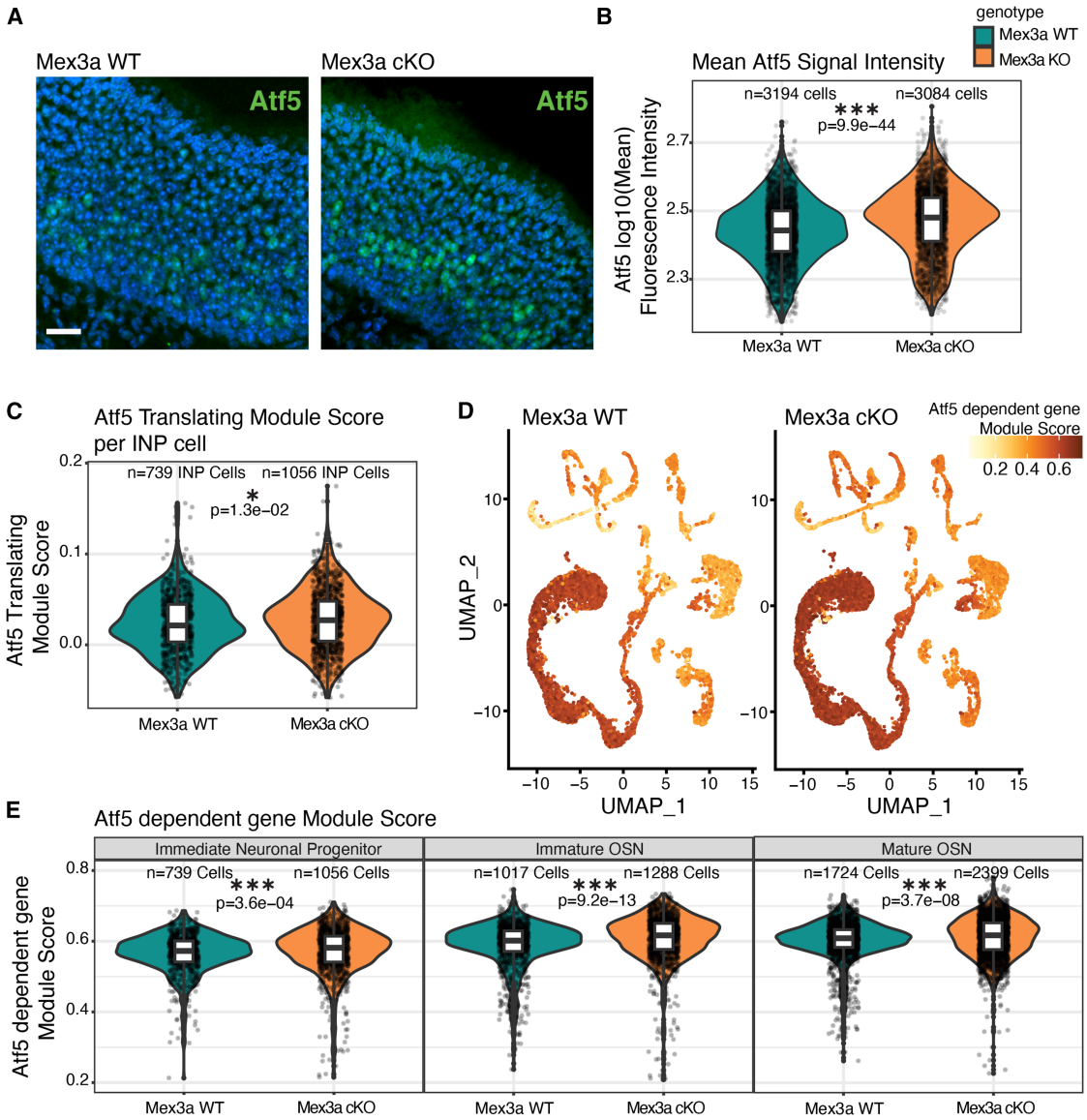


Figure 5. Perk-induced transcription factor Atf5 and targets exhibit increased expression in the Mex3a cKO

(A) Representative IF from three biological replicates at PN12, Mex3a WT littermate controls, and Mex3a cKO. Atf5 in green; DAPI in blue. Images from Mex3a WT and cKO are from the same zonal position in the MOE. Scale bar, 25 μM.

(B) Violin plot depicting mean Atf5 signal intensity of Atf5$^+$ individual cells, quantified from images taken from all zones of the olfactory epithelium. Each dot is a cell. *n* = number of Atf5$^+$ cells counted for each genotype. Three biological replicates per genotype.

(C) Violin plot quantifying "Atf5 translating" module score from scRNA-seq data (described in Figure 4A), INP cells, comparing Mex3a WT and cKO. See also Table S3.

(D) UMAP plot, which splits Mex3a WT (left) and Mex3a cKO (right) cells. Cells are colored based on module score for Atf5-dependent genes. Cells with a higher score have higher transcription of Atf5 target genes. See also Table S4.

(E) Violin plots quantifying Mex3a WT compared to cKO Atf5-dependent gene module scores across the neuronal lineage. *n* = number of cells from each genotype in each cell stage. Wilcoxon rank-sum test.

For all statistical tests in this figure: $^*p < 0.05$; $^{**}p < 0.01$; $^{***}p < 0.001$.

See also Figure S5.

proteosomal degradation of some of the critical ribosomal proteins engaged in OR mRNA translation. Third, Mex3a may also ubiquitinate the OR polypeptides as they are translated by the ribosome, resulting in immediate post-translational degradation by the proteosome.

In Mex3a cKO animals, we observe increased Atf5 nuclear staining as well as increased transcription of Atf5 downstream targets. Increased OR translation in Mex3a cKO MOE may suffice to activate Perk signaling, explaining these findings. An alternative possibility is that instead of directly targeting OR mRNA

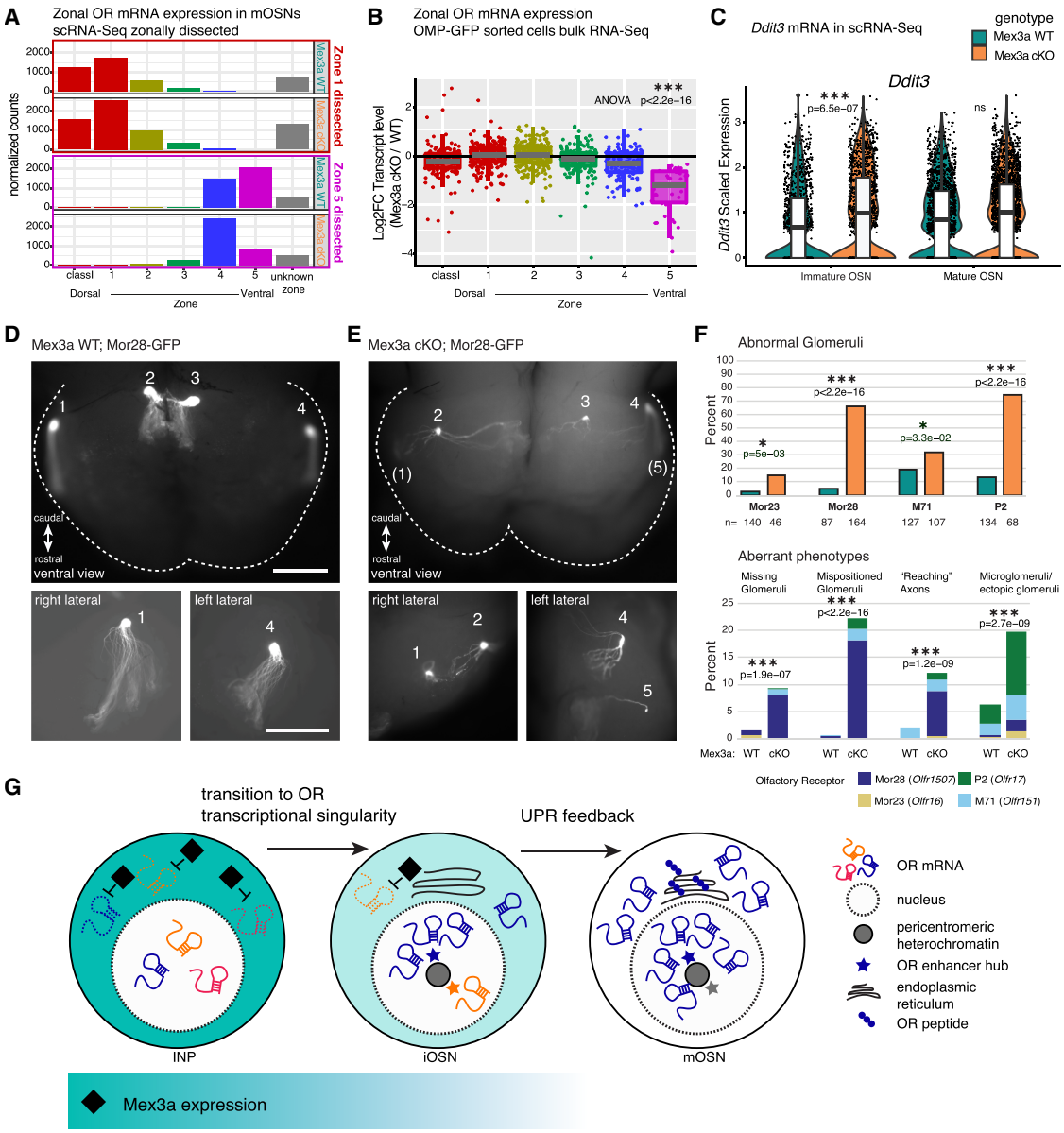


Figure 6. Mex3a deletion biases OR gene choice and disrupts axon guidance specificity

(A) Bar plot quantifying zonal OR expression in scRNA-seq from mOSNs of micro-dissected MOE from two biological replicates of four-week-old Mex3a cKO and Mex3a WT littermate controls. (Rows 1 and 2) scRNA-seq from dorsal dissection of MOE. (Rows 3 and 4) scRNA-seq from ventral dissection of MOE. Mex3a cKO is shown as the second panel in each case.

(B) Boxplot quantifying OR mRNA from three biological replicates of OMP-GFP sorted neurons from Mex3a cKO and Mex3a WT littermate controls. ORs (dots) divided into known zones. Whiskers extend to most extreme values within 1.5 times the interquartile range; center line is the median. ANOVA one-way test.

(C) *Ddit3* mRNA levels measured in single cells (dots) from PN12 scRNA-seq shown in Figure 4A; iOSN and mOSNs only. Wilcoxon rank-sum test (Seurat's FindMarkers).

(D) Representative whole-mount image from 22 biological replicates of an olfactory bulb from Mex3a WT; Mor28 GFP (ventral view). Mor28 glomeruli and axon fibers in white. (Bottom) Magnification of lateral glomeruli.

(E) Representative whole-mount image from 41 biological replicates of an olfactory bulb from Mex3a cKO; Mor28 GFP (ventral view).

(F) (Top) Bar plot quantifying abnormal phenotypes of individual glomeruli from four different OR-IRES-GFP lines mated with Mex3a cKO and WT littermate controls. N equals number of glomeruli. Fisher's exact test. (Bottom) Stacked bar plot depicting proportions of glomeruli from a given OR that exhibit abnormal wiring phenotypes (missing, mispositioned, reaching axons, ectopic, or microglomeruli). N equals number of glomeruli. Fisher's exact test.

(legend continued on next page)

stability or translation, Mex3a acts to repress Perk signaling, which in turn prevents OR translation before singular OR choice has been established. If Mex3a functions by modulating Perk signaling, then other G protein-coupled receptors (GPCRs) or ER-translated proteins may also be targets of Mex3a regulation. Mex3a is expressed in stem cell niches of multiple tissues including the intestinal crypt,[29] and prenatal cerebral cortex,[54] whose differentiated cell types express high levels of transmembrane bound receptors. Reports also demonstrate that the unfolded protein response (UPR) pathway is an integral part of cellular differentiation in these systems.[55,56] Thus, our findings invite the hypothesis that Mex3a may repress transmembrane bound proteins in immature cells of other tissues before terminal differentiation.

The intricate network of post-transcriptional OR regulation that we report here raises an important question: if OSNs have a mechanism in place that ensures absolute transcriptional singularity, why not use it from the very beginning, surpassing any need for post-transcriptional repression? Recent observations suggest that the nucleoprotein apparatus that ensures robust OR transcription, the multi-chromosomal enhancer hub, requires OR transcription for its assembly over any OR allele or for its recruitment to an OR locus. However, because OR promoters have extensive homology with one another,[57] OSN progenitors cannot activate only a single OR at once. Instead, they deploy polygenic transcription to generate nuclear architecture landscapes that are conducive to singular OR transcription through enhancer hub competition[18] and an RNA-mediated symmetry breaking process.[17] Thus, if the differentiating OSNs had to use the identity of the expressed OR protein as a trigger for the feedback signal and a compass for axon targeting, then they would need to deploy a transient post-transcriptional mechanism operating outside of the nucleus to uncouple the obligatory polygenic OR transcription from OR translation until singularity is established. It appears that Mex3a, with a strictly cytoplasmic distribution in the MOE, plays this critical function in the olfactory system.

Our Mex3a loss-of-function experiments reveal what the consequences would be for the developing olfactory system if polygenic OR transcription resulted in more efficient OR protein expression. Specifically, our observations from the Mex3a cKO mice suggest that post-transcriptional OR silencing during polygenic OR transcription prevents premature induction of the OR-elicited feedback signal. This delay ensures that the entire repertoire of ORs has the opportunity to be chosen, even the developmentally delayed zone 4/5 ORs,[53] and insulates OR-directed axon guidance from the conflicting signals of the co-transcribed ORs. Thus, the striking axon guidance defects observed in Mex3a cKO mice may be explained by both a role of zonal restrictions in OSN axon targeting[58–60] and the recently reported link between Perk signaling OSN axon fasciculation and glomerular specificity.[7]

In summary, we have uncovered a distinct regulatory function for Mex3a, a protein studied almost exclusively in the context of stem cell maintenance and renewal in physiological contexts,[30,31] and as a pro-proliferative, oncogenic factor in tumors.[33,38] Persistent Mex3a expression in post-mitotic cells of the MOE and post-transcriptional silencing of the largest mammalian gene family suggest that this protein and its paralogs may have additional functions that extend beyond the regulation of cell-cycle and cell-signaling pathways controlling cell division rates. In fact, our data suggest that Mex3a plays a crucial role in influencing the identity and eventual connectivity of post-mitotic OSN progenitors, a function conceptually similar to that of the *C. elegans* homolog, which specified anterior-posterior asymmetry and the fate of blastomere descendants.[46] If the OR CDS constitutes a generalizable target for post-transcriptional silencing, then many of our observations are likely applicable to other GPCRs with extensive homology to the OR superfamily. In future studies, it will be interesting to explore whether Mex3a post-transcriptionally represses GPCRs or other ER translated proteins in the diverse repertoire of tissues and cancer cell types where it is expressed.

Limitations of the study

We did not find a direct association between Mex3a and OR mRNA or protein, raising the possibility that Mex3a represses ORs in immature neurons via an indirect mechanism.

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources and reagents should be directed to and will be fulfilled by the lead contact, Stavros Lomvardas (sl682@columbia.cumc.edu).

Materials availability

Plasmids for tetO-Mex3a and mutant-RING Mex3a can be found on Addgene (ID number 223244 and 223245).

Data and code availability

- Data are deposited in GEO: GSE271027, GSE271029, and GSE271031 and MassIVE: MSV000095005.
- Code is available in Data S1.
- Any additional information required to reanalyze the data reported in this paper is available from the lead contact upon request.

ACKNOWLEDGMENTS

We thank Lomvardas lab members for critical discussion. We acknowledge the expertise and support from the animal husbandry teams at Columbia University, the Zuckerman Institute Flow Cytometry Core, the Flow Cytometry Core of the Columbia Center for Translational Immunology (CCTI) and Herbert Irving Comprehensive Cancer Center (HICCC). Imaging was performed with support from the Zuckerman Institute's Cellular Imaging platform. Transgenic mice were made with the Genetically Modified Mouse Model Shared Resource

(G) Model for Mex3a regulation of OR genes during OSN differentiation. Mex3a expression peaks during the INP stage, when multiple OR genes are expressed per cell. Mex3a-mediated repression of OR mRNAs/translation in the cytoplasm allows the neuron to undergo changes to nuclear architecture ensuring OR transcriptional singularity without triggering the unfolded protein response. Immature OSNs: as Mex3a expression wanes, OR mRNAs in the cytoplasm cease to be repressed, and the unfolded protein response is activated. In mature OSNs, Mex3a is no longer expressed, and OR translation increases.
For all statistical tests in this figure: $*p < 0.05$; $**p < 0.01$; $***p < 0.001$.
See also Figure S6.

at Columbia University. Single-cell RNA libraries were generated with the help of the Columbia Genome Center and Single Cell Core. This work was supported by a Helen Hay Whitney postdoctoral fellowship to R.D., an R21 DC017823-03 grant, a Kavli Institute for Brain Science grant, R01DC018744 (to S.L.), and NIH/National Institute of General Medical Sciences MIRA grant R35GM152258 to M.J. The CCTI and HICCC were supported by NIH S10OD020056 and S10OD030282.

AUTHOR CONTRIBUTIONS

S.L. and R.D. conceived of the project. S.L., M.J., and R.D. designed the experiments and secured the funding. R.D., H.S., M.E.d.A., M.W., J.K., A.U., A.K., F.C., L.E.B., E.B., A.D.P., I.S., and H.I.A. performed the experiments and the analyses. F.M.B. and E.B. generously gifted the Mex3a flox mouse. S.L. and R.D. wrote the manuscript.

DECLARATION OF INTERESTS

The authors declare no competing interests.

STAR★METHODS

Detailed methods are provided in the online version of this paper and include the following:

- KEY RESOURCES TABLE
- EXPERIMENTAL MODEL AND SUBJECT DETAILS
  - Generation of Mex3a transgenic alleles
  - Cell line
- METHOD DETAILS
  - Main olfactory epithelium imaging
  - Fluorescence-activated cell sorting
  - RNA-sequencing
  - Immunopurification mass spectrometry
  - Mass spectrometry TMT labeling and multiplexing
  - Mass spectrometry LC-MS/MS
  - Mass spectrometry data analysis
  - HEK293T immunopurification mass spectrometry
  - Western blot
  - Northern blot
  - Ribosome profiling
  - Pre-processing of ribosome profiling samples
  - Ribosome footprint isolation
  - Ribosome profiling library preparation/sequencing
  - Bioinformatic analysis of ribosome profiling data
  - Single cell RNA-Sequencing
  - Odor exposure assay
  - Whole mount olfactory bulb and glomerulus analysis
- QUANTIFICATION AND STATISTICAL ANALYSIS

SUPPLEMENTAL INFORMATION

Supplemental information can be found online at https://doi.org/10.1016/j.celrep.2025.115979.

Received: January 9, 2025
Revised: April 20, 2025
Accepted: June 18, 2025
Published: July 15, 2025

REFERENCES

1. Nguyen, M.Q., Zhou, Z., Marks, C.A., Ryba, N.J.P., and Belluscio, L. (2007). Prominent roles for odorant receptor coding sequences in allelic exclusion. Cell *131*, 1009–1017. https://doi.org/10.1016/j.cell.2007.10.050.

2. Pourmorady, A.D., Bashkirova, E.V., Chiariello, A.M., Belagzhal, H., Kodra, A., Duffié, R., Kahiapo, J., Monahan, K., Pulupa, J., Schieren, I., et al. (2024). RNA-mediated symmetry breaking enables singular olfactory receptor choice. Nature *625*, 181–188. https://doi.org/10.1038/s41586-023-06845-4.

3. Dalton, R.P., Lyons, D.B., and Lomvardas, S. (2013). Co-opting the unfolded protein response to elicit olfactory receptor feedback. Cell *155*, 321–332. https://doi.org/10.1016/j.cell.2013.09.033.

4. Shykind, B.M., Rohani, S.C., O'Donnell, S., Nemes, A., Mendelsohn, M., Sun, Y., Axel, R., and Barnea, G. (2004). Gene switching and the stability of odorant receptor gene choice. Cell *117*, 801–815.

5. Serizawa, S., Miyamichi, K., Nakatani, H., Suzuki, M., Saito, M., Yoshihara, Y., and Sakano, H. (2003). Negative feedback regulation ensures the one receptor-one olfactory neuron rule in mouse. Science *302*, 2088–2094. https://doi.org/10.1126/science.1089122.

6. Lewcock, J.W., and Reed, R.R. (2004). A feedback mechanism regulates monoallelic odorant receptor expression. Proc. Natl. Acad. Sci. USA *101*, 1069–1074.

7. Shayya, H.J., Kahiapo, J.K., Duffie, R., Lehmann, K.S., Bashkirova, L., Monahan, K., Dalton, R.P., Gao, J., Jiao, S., Schieren, I., et al. (2022). ER stress transforms random olfactory receptor choice into axon targeting precision. Cell *185*, 3896–3912.e22. https://doi.org/10.1016/j.cell.2022.08.025.

8. Wang, F., Nemes, A., Mendelsohn, M., and Axel, R. (1998). Odorant receptors govern the formation of a precise topographic map. Cell *93*, 47–60.

9. Feinstein, P., Bozza, T., Rodriguez, I., Vassalli, A., and Mombaerts, P. (2004). Axon guidance of mouse olfactory sensory neurons by odorant receptors and the beta2 adrenergic receptor. Cell *117*, 833–846. https://doi.org/10.1016/j.cell.2004.05.013.S0092867404005318.[pii].

10. Magklara, A., Yen, A., Colquitt, B.M., Clowney, E.J., Allen, W., Markenscoff-Papadimitriou, E., Evans, Z.A., Kheradpour, P., Mountoufaris, G., Carey, C., et al. (2011). An epigenetic signature for monoallelic olfactory receptor expression. Cell *145*, 555–570. https://doi.org/10.1016/j.cell.2011.03.040.

11. Lyons, D.B., Magklara, A., Goh, T., Sampath, S.C., Schaefer, A., Schotta, G., and Lomvardas, S. (2014). Heterochromatin-mediated gene silencing facilitates the diversification of olfactory neurons. Cell Rep. *9*, 884–892. https://doi.org/10.1016/j.celrep.2014.10.001.

12. Clowney, E.J., LeGros, M.A., Mosley, C.P., Clowney, F.G., Markenskoff-Papadimitriou, E.C., Myllys, M., Barnea, G., Larabell, C.A., and Lomvardas, S. (2012). Nuclear aggregation of olfactory receptor genes governs their monogenic expression. Cell *151*, 724–737. https://doi.org/10.1016/j.cell.2012.09.043.

13. Le Gros, M.A., Clowney, E.J., Magklara, A., Yen, A., Markenscoff-Papadimitriou, E., Colquitt, B., Myllys, M., Kellis, M., Lomvardas, S., and Larabell, C.A. (2016). Soft X-Ray Tomography Reveals Gradual Chromatin Compaction and Reorganization during Neurogenesis In Vivo. Cell Rep. *17*, 2125–2136. https://doi.org/10.1016/j.celrep.2016.10.060.

14. Lomvardas, S., Barnea, G., Pisapia, D.J., Mendelsohn, M., Kirkland, J., and Axel, R. (2006). Interchromosomal interactions and olfactory receptor choice. Cell *126*, 403–413. https://doi.org/10.1016/j.cell.2006.06.035.

15. Markenscoff-Papadimitriou, E., Allen, W.E., Colquitt, B.M., Goh, T., Murphy, K.K., Monahan, K., Mosley, C.P., Ahituv, N., and Lomvardas, S. (2014). Enhancer interaction networks as a means for singular olfactory receptor expression. Cell *159*, 543–557. https://doi.org/10.1016/j.cell.2014.09.033.

16. Monahan, K., Horta, A., and Lomvardas, S. (2019). LHX2- and LDB1-mediated trans interactions regulate olfactory receptor choice. Nature *565*, 448–453. https://doi.org/10.1038/s41586-018-0845-0.

17. Pourmorady, A.D., Bashkirova, E.V., Chiariello, A.M., Belagzhal, H., Kodra, A., Duffié, R., Kahiapo, J., Monahan, K., Pulupa, J., Schieren, I., et al. (2024). RNA-mediated symmetry breaking enables singular olfactory

receptor choice. Nature *625*, 181–188. https://doi.org/10.1038/s41586-023-06845-4.

18. Wu, H., Zhang, J., Jian, F., Chen, J.P., Zheng, Y., Tan, L., and Sunney Xie, X. (2024). Simultaneous single-cell three-dimensional genome and gene expression profiling uncovers dynamic enhancer connectivity underlying olfactory receptor choice. Nat. Methods *21*, 974–982. https://doi.org/10.1038/s41592-024-02239-0.
19. Tan, L., Xing, D., Daley, N., and Xie, X.S. (2019). Three-dimensional genome structures of single sensory neurons in mouse visual and olfactory systems. Nat. Struct. Mol. Biol. *26*, 297–307. https://doi.org/10.1038/s41594-019-0205-2.
20. Bashkirova, E.V., Klimpert, N., Monahan, K., Campbell, C.E., Osinski, J., Tan, L., Schieren, I., Pourmorady, A., Stecky, B., Barnea, G., et al. (2023). Opposing, spatially-determined epigenetic forces impose restrictions on stochastic olfactory receptor choice. eLife *12*, 87445. https://doi.org/10.7554/eLife.87445.
21. Lyons, D.B., Allen, W.E., Goh, T., Tsai, L., Barnea, G., and Lomvardas, S. (2013). An epigenetic trap stabilizes singular olfactory receptor expression. Cell *154*, 325–336. https://doi.org/10.1016/j.cell.2013.06.039.
22. Imai, T., Suzuki, M., and Sakano, H. (2006). Odorant receptor-derived cAMP signals direct axonal targeting. Science *314*, 657–661. https://doi.org/10.1126/science.1131794.
23. Serizawa, S., Miyamichi, K., Takeuchi, H., Yamagishi, Y., Suzuki, M., and Sakano, H. (2006). A neuronal identity code for the odorant receptor-specific and activity-dependent axon sorting. Cell *127*, 1057–1069. https://doi.org/10.1016/j.cell.2006.10.031.S0092-8674(06)01404-8.
24. Tan, L., Li, Q., and Xie, X.S. (2015). Olfactory sensory neurons transiently express multiple olfactory receptors during development. Mol. Syst. Biol. *11*, 844. https://doi.org/10.15252/msb.20156639.
25. Hanchate, N.K., Kondoh, K., Lu, Z., Kuang, D., Ye, X., Qiu, X., Pachter, L., Trapnell, C., and Buck, L.B. (2015). Single-cell transcriptomics reveals receptor transformations during olfactory neurogenesis. Science *350*, 1251–1255. https://doi.org/10.1126/science.aad2456.
26. Shum, E.Y., Espinoza, J.L., Ramaiah, M., and Wilkinson, M.F. (2015). Identification of novel post-transcriptional features in olfactory receptor family mRNAs. Nucleic Acids Res. *43*, 9314–9326. https://doi.org/10.1093/nar/gkv324.
27. Tan, K., Jones, S.H., Lake, B.B., Chousal, J.N., Shum, E.Y., Zhang, L., Chen, S., Sohni, A., Pandya, S., Gallo, R.L., et al. (2020). The role of the NMD factor UPF3B in olfactory sensory neurons. eLife *9*, e57525. https://doi.org/10.7554/eLife.57525.
28. Buchet-Poyau, K., Courchet, J., Le Hir, H., Séraphin, B., Scoazec, J.Y., Duret, L., Domon-Dell, C., Freund, J.N., and Billaud, M. (2007). Identification and characterization of human Mex-3 proteins, a novel family of evolutionarily conserved RNA-binding proteins differentially localized to processing bodies. Nucleic Acids Res. *35*, 1289–1300. https://doi.org/10.1093/nar/gkm016.
29. Barriga, F.M., Montagni, E., Mana, M., Mendez-Lago, M., Hernando-Momblona, X., Sevillano, M., Guillaumet-Adkins, A., Rodriguez-Esteban, G., Buczacki, S.J.A., Gut, M., et al. (2017). Mex3a Marks a Slowly Dividing Subpopulation of Lgr5+ Intestinal Stem Cells. Cell Stem Cell *20*, 801–816.e7. https://doi.org/10.1016/j.stem.2017.02.007.
30. Pereira, B., Amaral, A.L., Dias, A., Mendes, N., Muncan, V., Silva, A.R., Thibert, C., Radu, A.G., David, L., Máximo, V., et al. (2020). MEX3A regulates Lgr5(+) stem cell maintenance in the developing intestinal epithelium. EMBO Rep. *21*, e48938. https://doi.org/10.15252/embr.201948938.
31. Naef, V., De Sarlo, M., Testa, G., Corsinovi, D., Azzarelli, R., Borello, U., and Ori, M. (2020). The Stemness Gene Mex3A Is a Key Regulator of Neuroblast Proliferation During Neurogenesis. Front. Cell Dev. Biol. *8*, 549533. https://doi.org/10.3389/fcell.2020.549533.
32. Domingo-Muelas, A., Duart-Abadia, P., Morante-Redolat, J.M., Jordán-Pla, A., Belenguer, G., Fabra-Beser, J., Paniagua-Herranz, L., Pérez-Villalba, A., Álvarez-Varela, A., Barriga, F.M., et al. (2023). Post-transcriptional control of a stemness signature by RNA-binding protein MEX3A regulates murine adult neurogenesis. Nat. Commun. *14*, 373. https://doi.org/10.1038/s41467-023-36054-6.
33. Lederer, M., Muller, S., Glass, M., Bley, N., Ihling, C., Sinz, A., and Huttelmaier, S. (2021). Oncogenic Potential of the Dual-Function Protein MEX3A. Biology *10*, 415. https://doi.org/10.3390/biology10050415.
34. Yu, C.R., Power, J., Barnea, G., O'Donnell, S., Brown, H.E.V., Osborne, J., Axel, R., and Gogos, J.A. (2004). Spontaneous neural activity is required for the establishment and maintenance of the olfactory sensory map. Neuron *42*, 553–566. S0896627304002247 [pii].
35. Nguyen, M.Q., Marks, C.A., Belluscio, L., and Ryba, N.J.P. (2010). Early expression of odorant receptors distorts the olfactory circuitry. J. Neurosci. *30*, 9271–9279. https://doi.org/10.1523/JNEUROSCI.1502-10.2010.
36. Hanke, T., Szawlowski, P., and Randall, R.E. (1992). Construction of solid matrix-antibody-antigen complexes containing simian immunodeficiency virus p27 using tag-specific monoclonal antibody and tag-linked antigen. J. Gen. Virol. *73*, 653–660. https://doi.org/10.1099/0022-1317-73-3-653.
37. Kawaguchi, D., Sahara, S., Zembrzycki, A., and O'Leary, D.D.M. (2016). Generation and analysis of an improved Foxg1-IRES-Cre driver mouse line. Dev. Biol. *412*, 139–147. https://doi.org/10.1016/j.ydbio.2016.02.011.
38. Alvarez-Varela, A., Novellasdemunt, L., Barriga, F.M., Hernando-Momblona, X., Canellas-Socias, A., Cano-Crespo, S., Sevillano, M., Cortina, C., Stork, D., Morral, C., et al. (2022). Mex3a marks drug-tolerant persister colorectal cancer cells that mediate relapse after chemotherapy. Nat. Cancer *3*, 1052–1070. https://doi.org/10.1038/s43018-022-00402-0.
39. Fleischmann, A., Shykind, B.M., Sosulski, D.L., Franks, K.M., Glinka, M.E., Mei, D.F., Sun, Y., Kirkland, J., Mendelsohn, M., Albers, M.W., and Axel, R. (2008). Mice with a "monoclonal nose": perturbations in an olfactory map impair odor discrimination. Neuron *60*, 1068–1081. https://doi.org/10.1016/j.neuron.2008.10.046.
40. Fleischmann, A., Abdus-Saboor, I., Sayed, A., and Shykind, B. (2013). Functional interrogation of an odorant receptor locus reveals multiple axes of transcriptional regulation. PLoS Biol. *11*, e1001568. https://doi.org/10.1371/journal.pbio.1001568.
41. Jiang, Y., Gong, N.N., Hu, X.S., Ni, M.J., Pasi, R., and Matsunami, H. (2015). Molecular profiling of activated olfactory neurons identifies odorant receptors for odors in vivo. Nat. Neurosci. *18*, 1446–1454. https://doi.org/10.1038/nn.4104.
42. Knight, Z.A., Tan, K., Birsoy, K., Schmidt, S., Garrison, J.L., Wysocki, R.W., Emiliano, A., Ekstrand, M.I., and Friedman, J.M. (2012). Molecular profiling of activated neurons by phosphorylated ribosome capture. Cell *151*, 1126–1137. https://doi.org/10.1016/j.cell.2012.10.039.
43. Borden, K.L., Lally, J.M., Martin, S.R., O'Reilly, N.J., Solomon, E., and Freemont, P.S. (1996). In vivo and in vitro characterization of the B1 and B2 zinc-binding domains from the acute promyelocytic leukemia protooncoprotein PML. Proc. Natl. Acad. Sci. USA *93*, 1601–1606. https://doi.org/10.1073/pnas.93.4.1601.
44. McGlincy, N.J., and Ingolia, N.T. (2017). Transcriptome-wide measurement of translation by ribosome profiling. Methods *126*, 112–129. https://doi.org/10.1016/j.ymeth.2017.05.028.
45. Ingolia, N.T., Ghaemmaghami, S., Newman, J.R.S., and Weissman, J.S. (2009). Genome-wide analysis in vivo of translation with nucleotide resolution using ribosome profiling. Science *324*, 218–223. https://doi.org/10.1126/science.1168978.
46. Draper, B.W., Mello, C.C., Bowerman, B., Hardin, J., and Priess, J.R. (1996). MEX-3 is a KH domain protein that regulates blastomere identity in early C. elegans embryos. Cell *87*, 205–216. https://doi.org/10.1016/s0092-8674(00)81339-2.
47. Pereira, B., Sousa, S., Barros, R., Carreto, L., Oliveira, P., Oliveira, C., Chartier, N.T., Plateroti, M., Rouault, J.P., Freund, J.N., et al. (2013). CDX2 regulation by the RNA-binding protein MEX3A: impact on intestinal

differentiation and stemness. Nucleic Acids Res. *41*, 3986–3999. https://doi.org/10.1093/nar/gkt087.

48. Yang, K., Chen, G., Yu, F., Fang, X., Zhang, J., Zhang, Z., Shi, Y., and Zhang, L. (2024). Molecular mechanism of specific HLA-A mRNA recognition by the RNA-binding-protein hMEX3B to promote tumor immune escape. Commun. Biol. *7*, 158. https://doi.org/10.1038/s42003-024-05845-y.
49. Yang, L., Wang, C., Li, F., Zhang, J., Nayab, A., Wu, J., Shi, Y., and Gong, Q. (2017). The human RNA-binding protein and E3 ligase MEX-3C binds the MEX-3-recognition element (MRE) motif with high affinity. J. Biol. Chem. *292*, 16221–16234. https://doi.org/10.1074/jbc.M117.797746.
50. Pagano, J.M., Farley, B.M., Essien, K.I., and Ryder, S.P. (2009). RNA recognition by the embryonic cell fate determinant and germline totipotency factor MEX-3. Proc. Natl. Acad. Sci. USA *106*, 20252–20257. https://doi.org/10.1073/pnas.0907916106.
51. Tirosh, I., Venteicher, A.S., Hebert, C., Escalante, L.E., Patel, A.P., Yizhak, K., Fisher, J.M., Rodman, C., Mount, C., Filbin, M.G., et al. (2016). Single-cell RNA-seq supports a developmental hierarchy in human oligodendroglioma. Nature *539*, 309–313. https://doi.org/10.1038/nature20123.
52. Kahiapo, J.K. (2020). Atf5 links olfactory receptor induced stress response to proper neuronal function. PhD thesis (Columbia University).
53. Bashkirova, E.V., Klimpert, N., Monahan, K., Campbell, C.E., Osinski, J. M., Tan, L., Schieren, I., Pourmorady, A., Stecky, B., Barnea, G., et al. (2023). Opposing, spatially-determined epigenetic forces impose restrictions on stochastic olfactory receptor choice. Preprint at bioRxiv. https://doi.org/10.1101/2023.03.15.532726.
54. Weyn-Vanhentenryck, S.M., Feng, H., Ustianenko, D., Duffié, R., Yan, Q., Jacko, M., Martinez, J.C., Goodwin, M., Zhang, X., Hengst, U., et al. (2018). Precise temporal regulation of alternative splicing during neural development. Nat. Commun. *9*, 2189. https://doi.org/10.1038/s41467-018-04559-0.
55. Heijmans, J., van Lidth de Jeude, J.F., Koo, B.K., Rosekrans, S.L., Wielenga, M.C.B., van de Wetering, M., Ferrante, M., Lee, A.S., Onderwater, J.J.M., Paton, J.C., et al. (2013). ER stress causes rapid loss of intestinal epithelial stemness through activation of the unfolded protein response. Cell Rep. *3*, 1128–1139. https://doi.org/10.1016/j.celrep.2013.02.031.
56. Laguesse, S., Creppe, C., Nedialkova, D.D., Prévot, P.P., Borgs, L., Huysseune, S., Franco, B., Duysens, G., Krusy, N., Lee, G., et al. (2015). A Dynamic Unfolded Protein Response Contributes to the Control of Cortical Neurogenesis. Dev. Cell *35*, 553–567. https://doi.org/10.1016/j.devcel.2015.11.005.
57. Clowney, E.J., Magklara, A., Colquitt, B.M., Pathak, N., Lane, R.P., and Lomvardas, S. (2011). High-throughput mapping of the promoters of the mouse olfactory receptor genes reveals a new type of mammalian promoter and provides insight into olfactory receptor gene regulation. Genome Res. *21*, 1249–1259. https://doi.org/10.1101/gr.120162.110.
58. Imai, T., Yamazaki, T., Kobayakawa, R., Kobayakawa, K., Abe, T., Suzuki, M., and Sakano, H. (2009). Pre-target axon sorting establishes the neural map topography. Science *325*, 585–590. https://doi.org/10.1126/science.1173596.
59. Takeuchi, H., Inokuchi, K., Aoki, M., Suto, F., Tsuboi, A., Matsuda, I., Suzuki, M., Aiba, A., Serizawa, S., Yoshihara, Y., et al. (2010). Sequential arrival and graded secretion of Sema3F by olfactory neuron axons specify map topography at the bulb. Cell *141*, 1056–1067. https://doi.org/10.1016/j.cell.2010.04.041.S0092-8674(10)00495-2.
60. Ressler, K.J., Sullivan, S.L., and Buck, L.B. (1994). Information coding in the olfactory system: evidence for a stereotyped and highly organized epitope map in the olfactory bulb. Cell *79*, 1245–1255. 0092-8674(94)90015-9 [pii].
61. Tsue, A.F., Kania, E.E., Lei, D.Q., Fields, R., McGann, C.D., Hershberg, E., Deng, X., Kihiu, M., Ong, S.E., Disteche, C.M., et al. (2023). Oligonucleotide-directed proximity-interactome mapping (O-MAP): A unified method for discovering RNA-interacting proteins, transcripts and genomic loci in situ. Preprint at bioRxiv. https://doi.org/10.1101/2023.01.19.524825.
62. Laureau, R., Dyatel, A., Dursuk, G., Brown, S., Adeoye, H., Yue, J.X., De Chiara, M., Harris, A., Ünal, E., Liti, G., et al. (2021). Meiotic Cells Counteract Programmed Retrotransposon Activation via RNA-Binding Translational Repressor Assemblies. Dev. Cell *56*, 22–35.e7. https://doi.org/10.1016/j.devcel.2020.11.008.
63. Hornstein, N., Torres, D., Das Sharma, S., Tang, G., Canoll, P., and Sims, P.A. (2016). Ligation-free ribosome profiling of cell type-specific translation in the brain. Genome Biol. *17*, 149. https://doi.org/10.1186/s13059-016-1005-1.
64. Ibarra-Soria, X., Levitin, M.O., Saraiva, L.R., and Logan, D.W. (2014). The olfactory transcriptomes of mice. PLoS Genet. *10*, e1004593. https://doi.org/10.1371/journal.pgen.1004593.

STAR★METHODS

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Antibodies		
Mex3a – rabbit polyclonal	Abcam	Cat# ab79046; RRID:AB_2266271
Vglut2 – guinea pig polyclonal	Millipore Sigma	Cat# AB2251-I; RRID:AB_2665454
Mor28 (OR antibody) – guinea pig polyclonal		Gift from Gilead Barnea (Brown University); RRID:AB_2636806
C2 (OR antibody) – rabbit polyclonal		Gift from Gilead Barnea (Brown University)
P2 (OR antibody) – rabbit polyclonal		Gift from Gilead Barnea (Brown University)
M71 (OR antibody) – rabbit polyclonal		Gift from Gilead Barnea (Brown University)
Calmegin – goat polyclonal	Santa Cruz	Cat# sc-49899; RRID:AB_2081804
Atf5 – goat polyclonal	Santa Cruz	Cat# sc-46934; RRID:AB_2058761
Phosphorylated Rps6 (p-s6 ser244/247) – rabbit polyclonal	ThermoFisher	Cat#44-923G; RRID:AB_2533798
V5 tag – rabbit polyclonal	Abcam	Cat# ab15828; RRID:AB_443253
V5 tag – mouse polyclonal	Abcam	Cat# ab27671; RRID:AB_471093
Anti-mouse IgG – rabbit polyclonal	Abcam	Cat# ab46540; RRID:AB_2614925
PE CD54 – mouse monoclonal	Biolegend	Cat# 322707; RRID:AB_535979
PCNA – chicken polyclonal	Abcam	Cat# ab139696; RRID:AB_2894712
Deposited data		
Ribosome profiling sequencing data	This study	GEO: GSE271027
RNA-Sequencing data	This study	GEO: GSE271029
Single Cell RNA-Sequencing data	This study	GEO: GSE271031
Mass Spectrometry data	This study	MassIVE: MSV000095005
Experimental models: Cell lines		
HEK293T	Takara Bio	Cat # 632180
Experimental models: Organisms/strains		
Ngn-GFP, Tg(Neurog1-EGFP)DF148Gsat	MMRRC	RRID:MMRRC_010618-UCD
Ascl1-CRE-ERT2	JAX mice	RRID:IMSR_JAX:012882
Rosa26(LSL-tdtomato), Gt(ROSA) 26Sortm14(CAG-tdTomato)Hze	JAX mice	RRID:IMSR_JAX:007914
Atf5(rep)	Lab stock	PMID: 36167070
Omp(iresGFP) "OMP-GFP"	Donated by PI	PMID: 15186780
TetO-Mex3a WT-tTA-mCherry "Mex3a Tg"	This study	N/A
TetO-GFP-IRES-GFP "GFP Tg"	Donated by PI	PMID: 20610762
TetO-Mex3a mutant RING-V5-t2a-mCherry "Mex3a-mut Tg"	This study	N/A
gg8-tTA	Donated by PI	PMID: 18045541
Omp(irestTA), Omptm1(tTA)Gogo "OMP-tTA"	JAX mice	RRID:IMSR_JAX:017754
Mex3a flox	Donated by PI	PMID: 35773527
Foxg1tm1.1(cre)Ddmo/J	JAX mice	RRID:IMSR_JAX:029690
tetO-M71 "M71 Tg"	Donated by PI	PMID: 19109912
tetO-P2 "tetO-P2 knock-in"	Donated by PI	PMID: 23700388
Mor23(iresGFP) Olfr16tm2Mom	JAX mice	RRID:IMSR_JAX:006643
Mor28(iresGFP)	Donated by PI	PMID: 15186780

(*Continued on next page*)

Continued

REAGENT or RESOURCE	SOURCE	IDENTIFIER
P2(iresGFP)	Donated by PI	PMID: 15186780
M71(iresGFP), Olfr151tm26Mom	JAX mice	RRID:IMSR_JAX:006676
Oligonucleotides		
5′-ACTAACAGGCCAGAGCTCCA-3′	M71-Olfr151 CDS	FISH probe
5′-GGGGAGGATGTCACAGAAGT-3′	M71-Olfr151 CDS	FISH probe
5′-TTGAGACTGGCCTCATGTTG-3′	M71-Olfr151 CDS	FISH probe
5′-GCTGTAGATCAGGGGGTTGA-3′	M71-Olfr151 CDS	FISH probe
5′-TCAAAAGACTTTTCTCCTCAGTG-3’	M71-Olfr151 CDS	Northern probe
5′- ATGACTGCAGAGAATCAATCTAC-3′	M71-Olfr151 CDS	Northern probe
5′-BiotinTEG-GGGGGGATGCGTGCA TTTATCAGATCA-3′	rRNA-derived cDNA	Ribosome profiling
5′-BiotinTEG-TTGGTGACTCTAGAT AACCTCGGGCCGATCGCACG-3′	rRNA-derived cDNA	Ribosome profiling
5′-BiotinTEG-GAGCCGCCTGG ATACCGCAGCTAGGAATAATGGAAT-3′	rRNA-derived cDNA	Ribosome profiling
5′BiotinTEG-TCGTGGGGGGC CCAAGTCCTTCTGATCGAGGCCC-3′	rRNA-derived cDNA	Ribosome profiling
5′-BiotinTEG-GGGGCCGGGCC GCCCCTCCCACGGCGCG-3′	rRNA-derived cDNA	Ribosome profiling
5′-BiotinTEG-CCCAGTGCGCCC CGGGCGTCGTCGCGCCGTCG GGT CCC GGG-3′	rRNA-derived cDNA	Ribosome profiling
5′-BiotinTEG-TCCGCCGAGGGC GCACCACCGGC CCGTCTCGCC-3′	rRNA-derived cDNA	Ribosome profiling
5′-BiotinTEG-AGGGGCTCTCGC TTCTGGCGCCA AGCGT-3′	rRNA-derived cDNA	Ribosome profiling
5′-BiotinTEG-GAGCCTCGGTTGG CCCCGGATAGCCGGGTCCCCGT-3′	rRNA-derived cDNA	Ribosome profiling
5′BiotinTEG-TCCTCCCGGGGCT ACGCCTGTCTGAGCGTCGCT-3′	rRNA-derived cDNA	Ribosome profiling
5′BiotinTEG-TCGCTGCGATCTAT TGAAAGTCAGCCCTCGACACA-3′	rRNA-derived cDNA	Ribosome profiling
Recombinant DNA		
CMV_Ms2coatProtein_ Mex3a_V5_t2a_mCherry	Full length Mex3a	HEK293T Mass Spectrometry
CMV_Ms2coatProtein_ Mex3aRING_V5_t2a_mCherry	Truncated Mex3a: Valine423-Serine653	HEK293T Mass Spectrometry
CMV_TurboGFP_Ms2StemLoop	Turbo GFP	HEK293T Mass Spectrometry
Software		
FIJI	Version 2.1.0	Imaging analysis
NIS-Elements	Version 5.42.06	Imaging analysis
STAR	Version 2.5.3a	RNA-Seq alignment
DESeq2	Version 1.24.0	RNA-Seq analysis
rStudio	Version 1.1.456	Bioinformatics analysis and graphing
Cytoscape	Version 3.10.1	Protein network visualization
10X Genomics Cell Ranger	Version 5.0.1	Single Cell RNA-Seq alignment
Seurat	Version 3.2.3	Single Cell RNA-Seq analysis
MaxQuant	Version 1.6.0.16	Mass Spectrometry Analysis
xCalibur	Version 2.8	Mass Spectrometry data acquisition

EXPERIMENTAL MODEL AND SUBJECT DETAILS

Mouse protocols were approved by the Columbia University IACUC under protocol numbers AC-AAAT2450 and AC-AABG6553. All mice were housed in standard conditions with a 12-h light/dark cycle and access to food and water *ad libitum*. Strains used are indicated in Supplemental Mouse Line table. All animals were on a mixed genetic background and littermate controls were used for comparisons. Animals were sacrificed by decapitation (if younger than postnatal day 14) or CO_2 followed by cervical dislocation, and the main olfactory epithelium (MOE) or olfactory bulb was isolated by dissection. Experiments with Mex3a Transgenic mice were conducted between 3 and 4 weeks of age. Experiments with Mex3a cKO mice were conducted at PN12 (whole MOE scRNA-Seq, ribosome profiling, Immunofluorescence for Atf5), 4 weeks of age (zonally microdissected scRNA-Seq, Immunofluorescence for ORs in Ngn1-GFP positive mice), or 3–10 months old for glomerulus analysis. Careful care was taken to include both males and females for all genotypes as biological replicates for each study.

Generation of Mex3a transgenic alleles

Transgenic Mice were generated with the Genetically Modified Mouse Model Shared Resource at Columbia University. They were constructed with the following plasmids, available on Addgene (IDs 223244 and 223245) tetO-CMV-Mex3a-t2a-mCherry and tetO-CMV-Mex3amutRING-V5-t2a-mCherry. In Fusion cloning (Takara) was used to clone Mex3a that was partially amplified from MOE cDNA, and partially supplied via gene block. The amino acid sequence of *Mus musculus* Mex3a is preserved, however the CG content of the DNA sequence was reduced to facilitate cloning. For the mutant RING transgene, four point-mutations were introduced: Cys 484 Ala; His 486 Ala; Cys 490 Ala; Cys 493 Ala using a gene block (IDT).

Cell line

HEK293T cell line was obtained from Takara Bio (632180) grown at 37C; 5% CO2 in a humidified incubator and regularly screened for mycoplasma. HEK293T cells were maintained in high glucose Dulbecco's modified Eagle's medium (DMEM) (Gibco) supplemented with 10% heat inactivated Fetal Bovine Serum (Corning), 1X Minimum Essential Medium Non-essential amino acids (Gibco) and 1X GlutaMAX (Gibco).

METHOD DETAILS

Main olfactory epithelium imaging

Immunofluorescence

Olfactory Epithelium was dissected and placed in O.C.T. compound (Fisher Scientific) for 20 min before freezing in a dry ice, ethanol bath. If endogenous genetic labeling was visualized (Figures 1D, 3F, 4G, S3E, and S3G), olfactory epithelium was prefixed in 4% PFA (Fisher Scientific), 1X PBS for 8 min on ice, washed 3X with PBS 1X, and incubated in 30% sucrose, PBS 1X overnight at 4C on a rocking platform, then incubated in O.C.T. compound for 20 min before freezing in a dry ice, ethanol bath. Tissue sections were collected at 12um thickness on a Leica cm1950 Cryostat, with control and experimental tissue sections on the same slide to ensure the same experimental conditions throughout. Slides were allowed to come to RT for 10 min before fixing in 4% PFA, PBS 1X at RT for 5 min in a slide mailer. Slides were washed 3X in PBS 1X, then blocked for one hour in IF blocking buffer: PBS 1X, 4% sterile Donkey Serum (Sigma Aldrich), 1% Triton X-100. Slides were assembled in a humid chamber, and 100uL of blocking plus primary antibody (1:200 dilution) solution was added to the slide. A coverslip was placed over slide and incubated overnight at 4C in the humid chamber. The following day, slides were washed in a slide mailer 3 × 5 minutes in PBS 1X, Triton 0.1%, then incubated in secondary antibody solution (Jackson Immunoresearch) for 45 min at RT. Secondary antibody solution was IF blocking buffer with secondary antibodies diluted 1:500, and 1mg/mL DAPI diluted 1:1000. 100uL of secondary solution was applied to the slide and a coverslip was placed on top. Incubations were conducted in the humid chamber. Slides were washed in a slide mailer for 3 × 5 minutes in PBS 1X, Triton 0.1%, then mounted with Vectashield antifade mounting medium (Vector Laboratories). At least two biological replicates were analyzed per condition; details for replicates and statistics for each IF are described in figure legends.

In situ *hybridization*

M71 FISH. Probes were generated using oligonucleotides described in the key resources table to amplify two amplicons 495bp and 397bp from the *Olfr151* gene using MOE tissue cDNA as template. Amplicons were cloned into the TopoTA Dual Promoter vector (Invitrogen), linearized with SpeI (NEB), and *in vitro* transcribed using T7 RNA Pol (NEB) in 1X Sp6/T7 transcription buffer (Roche) with DIG dNTPs (Roche) and SUPERase-IN (Invitrogen). RNA FISH was performed as described in Clowney et al. 2012.[12] Five uM sections of MOE were incubated with RNA probe overnight in hybridization buffer, washed with 0.2X SSC, blocked with TNB buffer, incubated with primary antibody against Digoxigenin conjugated to alkaline phosphatase (Roche), washed with TNT buffer, and labeled with Fast Red secondary antibody (Roche) then counterstained with DAPI and mounted. Images were acquired using same settings for each genotype.

P2 intron FISH

Probes for the tetO-P2 intron were designed by the OligoMiner pipeline using the following settings: blockParse using parameters -l 30 -L 37 -t 42 -T 47 -s 390 -F 40. Bowtie2 parameters are—no-hd -t -k 100—very-sensitive-local. outputClean parameter is -u. Probes were ordered from IDT oPools and were cleaned up using the Oligo Clean & Concentration Kit (Zymo Research).

Fluorophore (647)-conjugated secondary probes were ordered from IDT. Fluorescence *in situ* hybridization for the P2 intron was carried out using the Oligonucleotide-directed proximity-interactome mapping (O-MAP) as described in Tsue et al. 2023[61] without HRP-mediated proximity labeling. Slides were imaged on a SoRa-W1-Yokogawa spinning disk confocal microscope. Images were acquired using same settings for each genotype.

Image acquisition and analysis

Microscopy: Images were taken on a Zeiss LSM 700 Confocal, a W1-Yokogawa spinning disk Confocal or a SoRa-W1-Yokogawa spinning disk confocal and analysis and quantification were performed using FIJI and NIS-Elements software, with detailed code found in Data S1.

Fluorescence-activated cell sorting

The olfactory epithelium was dissected into ice-cold PBS 1X and cells were dissociated with the Worthington Papain Dissociation System (Worthington Biochemical) according to manufacturer's instructions. Cells were resuspended in PBS 1X supplemented with 10% Fetal Bovine Serum and DNaseI provided with Worthington kit. DAPI was added to exclude dead cells during sorting. Cells were filtered and sorted with either a Beckman Coulter MoFlo Astrios EQ Cell Sorter or a BD Influx Cell Sorter. Sorted cells were centrifuged at 500g with a swinging bucket rotor, and the supernatant was removed. Cells used for RNA-Seq were resuspended in 500uL of TRIzol Reagent (Invitrogen). For Ribosome profiling experiments, all solutions beginning with dissection were supplemented with 100μg/mL Cycloheximide (Sigma Aldrich). After sorting, cell pellets were snap frozen in liquid nitrogen and stored at −80C until processed. HEK293T cells were washed with 1X PBS, incubated with trypsin/EDTA (Gibco) at 37C for 3 min, and trypsin was inactivated with FBS-containing medium. HEK293T cells were pipetted to obtain a single cell suspension and cells were filtered and sorted on a BD Influx Cell Sorter.

RNA-sequencing

RNA was extracted from FAC sorted cells using TRIzol reagent (Invitrogen). 1-Bromo-3-chloropropane (Sigma Aldrich) was added to isolate RNA into the upper aqueous phase, and Isopropanol was used to precipitate RNA. GenElute LPA (Sigma Aldrich) was added to facilitate precipitation. The RNA pellet was washed with 75% Ethanol, the pellet was allowed to dry 5 min at RT, and resuspended in H2O. RNA was immediately treated with DNase I (Ambion) at 37C for 30 min, and purified using AMPure XP beads (Beckman Coulter). RNA-Seq libraries were made using the SMARTer Stranded Total RNA-Seq Kit - Pico Input (Takara) or TruSeq Stranded Total RNA Library Prep (Illumina) according to manufacturer's instructions. Comparisons between samples are from the same library preparation kit. Libraries were sequenced on a NextSeq 550 Illumina sequencer. Data were analyzed with DESeq2, with detailed code found in Data S1. Two biological replicates were used for each genotype/sorted cell sample.

Immunopurification mass spectrometry

Two biological replicates were used for each genotype, except for Mex3a cKO (4 replicates). Olfactory epithelium was dissected and lysed in IP lysis buffer (150 mM NaCl, 50mM Tris pH 7.5, 1% NP-40 Igepal Ca-630, 5% glycerol, protease and phosphatase inhibitors) for 30–60 min rotating at 4C. Lysates were centrifuged to remove insoluble material, and protein concentration was estimated with the Pierce BCA protein Assay kit (Thermo Fisher Scientific). 1mg of protein was used per IP except for WT mice for the Mex3a/IgG pulldowns where 4mg of protein were used. 10 μg of antibody (Mex3a, V5, or IgG) were added to lysates (volume brought up to 500uL with lysis buffer), and 100 μL of Protein G Dynabeads beads were added per IP. IPs were rotated overnight at 4C.

Immunopurified material was separated by magnet, washed 2X in IP wash buffer +0.05% NP-40 Igepal (150mM NaCl, 50mM Tris pH 7.5, 5% glycerol) and 2X in IP wash buffer without NP-40. Peptides were eluted from beads by adding 80uL of 2M Urea, 50mM Tris pH7.5, 1mM DTT, 5 μg/mL Trypsin prepared in HPLC water, and shaking for 1 h at RT 1,000 rpm. Eluate was collected, and beads were washed twice with 2M Urea, 50mM Tris pH 7.5. Washes and eluate were combined. Proteins were reduced by adding 4mM DTT and incubating 45 min at RT, shaking 1,000 rpm. Proteins were alkylated by adding 10mM iodoacetamide, protected from light and incubated 45 min RT, shaking 1,000 rpm. Peptides were digested by adding 0.5ug Trypsin and incubating at RT overnight, shaking 700 rpm. The following day, 1% formic acid was added to acidify proteins. C18 stage tips were prepared by packing two disks of Empore 3M C18 material into 200uL tips. Stage tips were equilibrated by sequential washes of 100 μL of 100%MeOH, 80%ACN/0.2%Formic Acid, 2x 3%ACN/0.2%Formic Acid, then the acidified peptides were loaded on the stage tip. Stage tips were washed twice with 3%ACN/0.2% Formic acid, and peptides were eluted with 60% ACN/0.2% Formic Acid, dried in a SpeedVac vacuum concentrator.

Mass spectrometry TMT labeling and multiplexing

The resulting peptides were labeled with the TMT11plex mass tag labeling reagent following the manufacturer's protocol (Thermo Scientific) with slight modifications. In brief, 0.5 units of TMT11plex reagent was used for each IP condition. The dried peptides were dissolved in 30 μL of 50 mM HEPES solution (pH 8.5), and the TMT11plex reagent was added in 12.3 μL of 100% ACN. The labeling reaction was incubated at RT at 600 rpm for 1 h, then quenched with 2.5 μL of 5% hydroxylamine for 15 min at RT. The samples were then pooled according to the mixing scheme and dried to at least three-quarters of their volume to reduce the acetonitrile concentration to less than 5%. The labeled peptides were then acidified with 10% formic acid (pH < 3), followed by C18 clean-up. The

desalted peptides were evaporated to dryness and reconstituted in 12 μL 3%ACN/0.2%Formic Acid for the subsequent liquid chromatography-tandem mass spectrometry (LC-MS/MS) analysis.

Mass spectrometry LC-MS/MS

Around 1 μg of total peptides were analyzed on an ACQUITY UPLC M-class system (Waters) coupled via a 50 cm Thermo Easy-Spray LC column (inner diameter of 75 μm, packed with 2 μm C18 bead stationary phase, 25 cm length, Thermo Fisher Scientific product #: ES803A) to a benchtop Orbitrap Q Exactive HF mass spectrometer (Thermo Fisher Scientific). Peptides were separated at a flow rate of 250 nL/min with a linear 5 min linear gradient from 2% to 5% solvent B (100% acetonitrile, 0.1% formic acid), followed by a 75 min gradient from 5% to 30% solvent B, followed by a linear 20 min gradient from 30 to 60% solvent B, followed by a linear 5 min gradient from 60 to 90% solvent B. Each sample was run for 160 min, including sample loading and column washing and equilibration times. Data were acquired in data dependent mode using Xcalibur 2.8 software. MS1 Spectra were measured with a resolution of 60,000, an AGC target of 3e6 and a mass range from 375 to 2000 m/z. Up to 15 MS2 spectra per duty cycle were triggered at a resolution of 60,000, an AGC target of 2^e5, an isolation window of 1.6 m/z and a normalized collision energy of 36.

Mass spectrometry data analysis

All raw data were analyzed with MaxQuant software version 1.6.0.16 (Cox and Mann, 2008) using an UniProt mouse database (release 2014_07, Mus musculus), and MS/MS searches were performed with the following parameters: TMT-11plex labeling on the MS2 level, oxidation of methionine and protein N-terminal acetylation as variable modifications; carbamidomethylation as fixed modification; Trypsin/P as the digestion enzyme; precursor ion mass tolerances of 20 p.p.m. for the first search (used for nonlinear mass re-calibration) and 4.5 p.p.m. for the main search, and a fragment ion mass tolerance of 20 p.p.m. For identification, we applied a maximum FDR of 1% separately on protein and peptide level. We required 1 or more unique/razor peptides for protein identification.

After identifying peptides that were enriched 0.5 Log Fold greater than IgG IPs, six replicates (first three panels of Figure 2B) were compared to peptides enriched in the Mex3a cKO IPs. Peptides passing a threshold of 0.5 log fold higher than Mex3a cKO, and a Wilcoxon rank-sum test resulting in a p value of 0.05 or lower were considered candidates for Mex3a protein interacting partners.

Further details related to data analysis can be found in Data S1.

HEK293T immunopurification mass spectrometry

Constructs used included CMV_Ms2coatProtein_FullLengthMex3a_V5_t2a_mCherry, CMV_Ms2coatProtein_Mex3aRING_V5_t2a_mCherry (Mex3a Valine423 to Serine653), CMV_TurboGFP_Ms2StemLoop. Ms2 system not used for downstream analysis presented here. Cells were seeded onto 6 well plates and a total of 2.5μg of DNA was transfected using Lipofectamine 3000 (1.25μg each plasmid). Duplicate 6-well transfections were passed into 10cm plates and allowed to grow one day. Cells were FAC-sorted for mCherry two days post-transfection. Untransfected control was trypsinized and processed alongside sorted cells. Cells were lysed in 500uL IP lysis buffer (as described for olfactory epithelium). Lysates were split in two 250uL aliquots, and 1uL of RNase A was added to one of these aliquots. Both aliquots were incubated at 37C for 20 min, then 250uL lysis buffer was added to each aliquot to bring the volume back to 500uL. 50uL Protein A Dynabeads and 50uL Protein G Dynabeads washed in lysis buffer, 5μg of mouse anti V5 antibody and 5μg of rabbit anti V5 antibody were added to each IP. Lysates were incubated end-over-end overnight at 4C and washed and prepared for Mass Spec as described for olfactory epithelium. Two independent experiments were conducted on different days (transient transfections of HEK293T and IP for V5) for each condition.

Western blot

Olfactory epithelium was dissected and minced in a glass dissection dish on ice. Tissue was lysed at 4C, rotating for 30 min in Radio immunoprecipitation assay buffer (RIPA) (the following chemicals from Sigma Aldrich): 150 mM sodium chloride, 1.0% NP-40 Igepal ca-630, 0.5% sodium deoxycholate, 0.1% SDS (sodium dodecyl sulfate), 50 mM Tris, pH 8.0, supplemented with protease inhibitors (Halt Protease inhibitor cocktail, EDTA free, Thermo Fisher Scientific). Lysates were centrifuged to remove insoluble materials, and sonicated using a Covaris Sonicator to sonicate genomic DNA with the settings Duty cycle, 2%; Intensity, 3; Cycles/Burst, 200; Frequency sweeping; Max temp = 6C. Sonicated lysates were quantified using a Pierce BCA protein Assay kit (Thermo Fisher Scientific), and combined with Laemmli 4X loading buffer supplemented with β-Mercaptoethanol. Lysates were loaded onto a 4–20% precast polyacrylamide gel (BioRad) and protein electrophoresis was completed with constant mA. Proteins were transferred to a PVDF membrane, blocked in Intercept (PBS) Blocking Buffer (Licor), and incubated with primary antibodies (Mex3a, 1:3000 dilution, PCNA, 1:3000 dilution) overnight in blocking buffer supplemented with 0.2% Tween 20. The following day, membranes were washed 3 × 7 minutes in PBS 1X; Tween 20 0.1% and incubated for 1 h at RT with IRDye secondary antibodies (Licor) in Intercept PBS blocking buffer supplemented with Tween 20 0.2% and SDS 0.01%. Membranes were washed 3 × 7 minutes with PBS 1X; Tween 20 0.1%, and imaged on a Licor Odyssee Imaging System. Western blot presented is from two biological replicates per genotype, representative of more than five biological replicates.

Northern blot

Northern blot was performed as described in Laureau et al. 2021.[62] Probe was amplified by PCR using NEBNext High-Fidelity 2X Master mix with Q5 DNA polymerase (NEB M0541L) with primers described in key resources table that amplify a 929 bp amplicon spanning the M71 CDS. RNA was extracted from the MOEs of two biological replicates of three-week-old mice that were OMP-tTA; M71 Tg or OMP-tTA M71 Tg; Mex3a Tg. RNA was extracted using Trizol and 1-Bromo-3-chloropropane following standard protocols.

Ribosome profiling

Two replicates of Ngn-GFP+ and OMP-GFP+ cells were collected from olfactory epithelium by FACs from Mex3a WT and Mex3a cKO littermates. Ribosome profiling was performed according to McGlincy and Ignolia 2017[44] as described in greater detail below.

Pre-processing of ribosome profiling samples

To quantify translation, both ribosome profiling and RNA-seq libraries were prepared from the same FAC-sorted cell samples. Prior to FAC sorting, OEs were dissected and dissociated to single cells using a papain dissociation system (Worthington). Cycloheximide (CHX, Sigma, 100ug/mL) was added to all solutions throughout the dissociation and FAC-sorting steps to arrest ribosomes on mRNA transcripts. Following sorting, cells were pelleted for 10 min at 800g and 4°C, supernatant was aspirated, pellets were snap frozen in liquid nitrogen, and pellets were stored at −80°C. When ready to proceed with RNA and ribosome isolation, pellets were thawed and lysed for 10 min at 4°C in RNA lysis buffer (20mM Tris pH 7.5, 150mM NaCl, 5mM MgCl2, 1mM DTT, 1% (v/v) Triton X-100, 25U/mL Turbo DNAse (Thermo), 100ug/mL cycloheximide. RNA concentrations were determined by standard curve using the Quant-IT RiboGreen RNA Assay kit (Thermo Fisher Scientific). Roughly 1ug of RNA was taken forward for footprinting and ribosome profiling library construction; roughly 200ng of RNA was diluted in Trizol and reserved for RNA-seq library construction.

Ribosome footprint isolation

Lysed RNA was diluted to 200uL with polysome buffer (20mM Tris pH 7.5, 150mM NaCl, 5mM MgCl2, 1mM DTT) and treated for 45 min at room temperature with 10U RNase I (Thermo Fisher). Digestion was stopped with 200U SUPERase In (Invitrogen), contents were transferred to 3.5mL quick-seal ultracentrifuge tubes (Beckman Coulter), and a cushion of 1M sucrose and 20U/mL SUPERase In diluted in polysome buffer was underlaid using a blunt fill needle. Ribosomes were pelleted through the sucrose cushion by ultracentrifugation for 4 h at 37,000 rpm and 4°C using a Beckman Sw41.Ti rotor. Supernatants were aspirated, pellets containing ribosome protected fragments were resuspended in 300uL Trizol, and RNA fragments were isolated using the Direct Zol kit (Zymo), eluting in 88uL water. Isopropanol precipitation was then performed overnight at −80°C by adding 2uL glycoblue (Thermo Fisher), 10uL 3M sodium acetate, and 150uL of isopropanol to the eluted RNA. RNA was then thawed on ice, pelleted by centrifugation (20,000g, 30minutes, 4°C), washed twice with 70% ethanol, dried, and resuspended in 10mM Tris. Ribosome protected footprints were then separated from contaminating rRNA by electrophoresis through 15% TBE-Urea gels. Gels were stained with SYBER Gold (Thermo Fisher) and the region between marker RNA oligonucleotides NI-800 and NI-801 was isolated. Gel fragments were crushed in 400uL RNA gel extraction buffer (300mM sodium acetate, 1mM EDTA, 0.25% (v/v) SDS), briefly frozen over an ethanol/dry ice slurry, and thawed to room temperature overnight. Extraction buffer was removed and RNA was precipitated with 500uL of isopropanol and 1uL of glycoblue (Thermo Fisher) overnight at −80°C. The next day, RNA was thawed on ice, 1uL of glycoblue was added, and RNA was pelleted by centrifugation (20,000g, 30minutes, 4°C). Pellets were washed twice in 70% ethanol, dried, and resuspended in 4uL 10mM Tris pH 8. RNA was diluted to 43uL with water, denatured for 90s at 80°C, and end healing was performed by adding 5uL 10X T4 buffer (NEB), 1uL SUPERase-In and 10U T4 PNK (NEB). The reaction was incubated for 1hr at 37°C followed by heat inactivation for 10 min at 70°C. Fragments were then purified by overnight precipitation with sodium acetate, glycoblue and isopropanol as above.

Ribosome profiling library preparation/sequencing

Sequencing libraries were prepared from isolated ribosome protected fragments using the ligation-free method[63] and the SMARTer SmRNA-seq Kit for Illumina (Takara Bio USA). Manufacturer recommendations were followed for polyadenylation (1uL of ATP was added to each reaction) and reverse transcription. After cDNA synthesis was completed, rRNA-derived cDNA was depleted using the biotinylated oligonucleotide pool[63] described in the key resources table (Integrated DNA Technologies). The subtraction oligonucleotide pool was 10uM in each oligo (110uM total); 2uL of this pool was added to 10uL (half) of the cDNA library and annealed (90s at 100°C, −0.1°C/s ramp down to 37°C, hold for 15min at 37°C). rRNA-derived cDNA fragments were then depleted by resuspending 25uL of Dynabeads MyOne Streptavidin C1 (Invitrogen, pre-washed in 20mM Tris, 250mM NaCl, 15mM $MgCl_2$) in the hybridized mixture, incubating for 15min at 37°C, and immobilizing the beads on a magnet. Supernatant containing the depleted library was then used to resuspend a second aliquot of 12.5uL pre-washed Dynabeads, the mixture was incubated for 15min at 37°C, and the final depleted cDNA library was removed after beads were immobilized on a magnet. Indexes were incorporated during a final library amplification step as per manufacturer directions from the SMARTer smRNA-seq kit. Serial 1.8x and 1.2x Ampure XP bead (Beckman) purifications were performed to isolate the final library for sequencing. QC was performed by bioanalyzer (Agilent), regularly showing a major library peak at ∼185bp and a minor artifact of the template switching reaction at ∼145bp. Library concentrations were determined by KAPA assay (Roche) or Qubit HS assay (Thermo Fisher Scientific).

Single end sequencing was performed using the NextSeq 550 system (Illumina), with 20% Phi-X (Illumina) spike-in. 50 cycles of sequencing were performed.

To quantify translation, a corresponding RNA-seq library was constructed for each ribosome profiling sample. Roughly 200ng aliquots of RNA in trizol (see above) were thawed to room temperature and extracted via the Direct Zol RNA kit (Zymo). RNA was eluted in 89uL of nuclease free water, to which 1uL linear polyacrylamide (Sigma), 10uL of 3M sodium acetate, and 150uL of isopropanol were added. Samples were frozen overnight at −80°C. Samples were then thawed to room temperature, RNA was pelleted at 20,000g for 30minutes at 4°C, pellets were washed twice in 75% ethanol, and RNA was resuspended in 17uL of nuclease free water. Genomic DNA was removed by treatment with 1uL TURBO DNAse (Invitrogen) in the corresponding TURBO DNAse buffer for 30 min at 37°C. RNA was purified on 1.8x Ampure Beads (Beckman) and resulting RNA concentrations were determined using the Quant-It RiboGreen RNA Assay kit (Thermo Fisher Scientific). 7ng of RNA was taken forward to RNA-seq library preparation using the SMARTER Stranded Total RNA-Seq Kit- Pico Input Mammalian v2 (Takara Bio USA) and following all manufacturer instructions. Library quality was assessed via bioanalyzer (Agilent) and concentrations were determined by KAPA assay (Roche). Single end sequencing was performed using the NextSeq 550 system (Illumina). 75 cycles of sequencing were performed.

Bioinformatic analysis of ribosome profiling data

Demultiplexed fastq files containing ribosome profiling reads were downloaded from BaseSpace (Illumina). Poly-A tails were removed from each ribosome profiling read using cutadapt v1.17 with the following flags –cut 3 -a "A15" -j 20 –nextseq-trim = 20 –minimum-length 17. Reads aligning to rRNA were removed using bowtie v1.2.2, a genome index built from the mouse 45S pre-rRNA (NCBI reference sequence NR_046233.2), and the flags -v 0 –un, to write all imperfect matches to an intermediate output file. These non-rRNA reads were then compressed using pigz and aligned to the genome using STAR v2.5.3a with the following flags –runMode alignReads –outSAMtype BAM SortedByCoordinate –alignSJoverhangMin 400 –outFilterMismatchNmax 0 –outFilterMatchNmin 15 –outFilterMultimapNmax 1 –readFilesCommand zcat. The genome index for STAR alignment was built in a separate step using STAR –runMode genomeGenerate –sjdbOverhang 29, the mm10 mouse genome assembly, and a custom transcriptome including extended olfactory receptor gene annotations.[64] Aligned BAM files were indexed using samtools v1.7. For all ribosome profiling experiments, $n = 2$ biological replicates were combined into a single sample by concatenating the bam files using samtools merge. Finally, p-offsets were determined for each read length in each merged sample. These offsets define the distance in nt from the 5′-end of each read to the estimated location of the ribosomal p-site and are used to collapse the read to a single point on the genome during downstream read-counting applications. We estimated p-offsets using the psite tool from the plastid package and flags –countfile_format BAM –min_counts 50 –require_upstream –min_length 17 –max_length 35. P-offsets were adjusted as needed by manually evaluating the histogram of read 5′ ends relative to transcription start sites genome-wide, using the intermediate files generated by the psite program.

For the RNA-seq libraries used in the ribosome profiling analysis, single-end reads downloaded from BaseSpace (Illumina) were reverse complemented using the fastx_reverse_complement tool from the fastx toolkit (available at http://hannonlab.cshl.edu/fastx_toolkit), v0.0.13, using the -Q33 flag. Reverse-complemented reads were aligned using STAR flags –runMode alignReads –outSAMtype BAM SortedByCoordinate –outFilterType BySJout –outFilterIntronMotifs RemoveNoncanonicalUnannotated –outSAMstrandField intronMotif –outFilterMultimapNmax 10. The genome for STAR alignment was built from mm10 and the same transcriptome noted above, but with –sjdbOverhang 74 to reflect the longer RNA-seq read length. High quality alignments were extracted with samtools view (-b -q 30) and bam files were indexed using samtools index. As with ribosome profiling libraries, $n = 2$ biological replicates were combined into a single sample for downstream analysis by concatenating the bam files using samtools merge.

Read counting for ribosome profiling and paired RNA-seq libraries was performed using custom python scripts built using tools from the plastid package. For each gene in the custom transcriptome described above, the annotated coding sequences for all transcripts were concatenated together to define a *meta*-CDS. Meta 5′-untranslated regions (5′UTRs) and 3′UTRs were similarly defined by concatenating together the annotated 5′UTRs and 3′UTRs for all transcripts belonging to a given gene, with any regions falling within the *meta*-CDS for that gene being removed. Each ribosome profiling read was reduced to a single count at a length of p-offset nt from the 5′ end of the read. Each RNA-seq read was reduced to a single count at the 5′ end of the read. Ribosome profiling and RNA-seq reads falling within each meta region were then counted and the lengths of each meta-region were tabulated. Translational efficiency was defined as (ribo_reads_meta_cds/[length meta_CDS * total_ribo_reads])/(rna_reads_meta_cds/[length meta_CDS * total_rna_reads]), where total read numbers reflect reads mapping to any meta-region of any transcript. Ribosome profiling reads per kilobase per million (RPKMs) were calculated as ribo_reads_meta_cds/(cds_len/1000 * sum(ribo_reads_meta_cds)) * 1000000. RNAseq reads per kilobase per million (RPKMs) were calculated as rna_reads_meta_cds/(cds_len/1000 *sum(rna_reads_meta_cds)) * 1000000. RNAseq transcripts per million (TPM) were calculated as rnaseq_rpkm/sum(rnaseq_rpkm) * 1000000.

Output count tables generated above were read into R for final analysis and visualization. Only genes expressed >1 RPKM by ribosome profiling and >1 TPM by RNA-seq were considered. Log2 fold changes in translational efficiency and RNA-seq TPM were computed. In all cases, log2 fold changes in translation efficiency were negatively correlated with log2 fold changes in RNA-seq TPM. To better identify outliers against this general trend, we defined the corrected log2 fold change in translation efficiency as the residuals of a linear model of log2 fold change translational efficiency ~ log2 fold change RNA-seq TPM using the predict and lm functions from the base R 'stats' package. Corrected log2 fold changes in translation efficiency were plotted against log2 fold changes in RNA-seq TPM using the ggplot2 and ggMarginal R packages.

Single cell RNA-Sequencing

Olfactory epithelium was dissected and dissociated into single cells using the Worthington Papain dissociation system (Worthington Biochemical). Cells were counted and processed according to manufacturer's instructions using 10X Genomics equipment and reagents from the Chromium Single Cell Gene Expression v3 kit with the help of the Columbia Genome Center. Libraries were sequenced on an Illumina NextSeq 550 Sequencing machine, libraries were aligned with Cell Ranger (10X genomics) with introns included. Data were analyzed using Seurat. Code used for analysis can be found in Data S1. For whole MOE scRNA-Seq RNA, two biological replicates of each genotype were used at PN12. For zonally dissected scRNA-Seq, two biological replicates of each genotype were used at 4 weeks of age.

Odor exposure assay

Mice were placed in a laminar flow hood in cages without bedding and with odor for four hours. 200uL of Acetophenone (Sigma Aldrich) was dotted onto Whatman paper and put into a glass vial. The top of the vial was covered with cheese cloth to allow the odor to permeate the cage. After four hours, mice were sacrificed, MOEs were dissected and embedded in OCT compound and IF was performed for phosphorylated S6 as described above. Whole coronal sections of MOE were imaged at 20X on a Yokogawa Spinning disk confocal microscope, and images were analyzed using Nikon NIS-Elements software with the Segment.ai function. To eliminate ps6 signal in non-OSN parts of tissue, an initial training was conducted on the DAPI dense undulations where OSNs reside in the tissue. Only phosphorylated S6+ cells within those regions were counted and measured for further analysis. Two biological replicates per genotype.

Whole mount olfactory bulb and glomerulus analysis

Mice carrying OR-ires-GFP alleles (Mor23iresGFP, Mor28iresGFP, M71iresGFP, P2iresGFP) were bred into the Mex3a flox; Foxg1iresCre line. Mice were allowed to age and sacrificed between three and ten months of age, averaging five months old at time of experiment, twelve or more mice per OR and genotype. Olfactory bulbs (OBs) were dissected with a portion of the rest of the brain, carefully separating bulbs from the olfactory epithelium. OBs were kept in PBS 1X on ice until imaged the same day. Bulbs were imaged on a Nikon SMZ18 Stereo Microscope. Once whole OB images were taken, OB lobes were hemisected to reveal and image additional glomeruli.

QUANTIFICATION AND STATISTICAL ANALYSIS

Immunofluorescence imaging experiments were quantified with FIJI or NIS Elements. RNA-Seq data were quantified with STAR and DESeq2, and scRNA-Seq data were quantified with 10x Genomics CellRanger and Seurat. Mass Spectrometry data were quantified with XCalibur and MaxQuant. Glomeruli were analyzed by counting images of olfactory bulbs in a Mex3a cKO genotype-blind fashion.

Statistical tests were performed in R and specific tests and *n* values are listed in figure legends, Data S1, and method details (mass spectrometry, ribosome profiling). A *p*-value cutoff of 0.05 was considered statistically significant for all analyses. *p*-values are presented on figures, with asterisks representing confidence as follows: $p < 0.05$ = *, $p < 0.01$ = **, $p < 0.001$ = ***.

Cell Reports, Volume *44*

Supplemental information

Mex3a-dependent post-transcriptional silencing ensures olfactory receptor diversity and axon guidance specificity

Rachel Duffié, Hani Shayya, Martín Escamilla del Arenal, Miao Wang, Jerome Kahiapo, Aileen Ugurbil, Abdurrahman Keskin, Fiona Clowney, Elizaveta V. Bashkirova, Ariel D. Pourmorady, Ira Schieren, Humberto Ibarra Avila, Luke E. Berchowitz, Francisco M. Barriga, Eduard Batlle, Marko Jovanovic, and Stavros Lomvardas

Figure S1

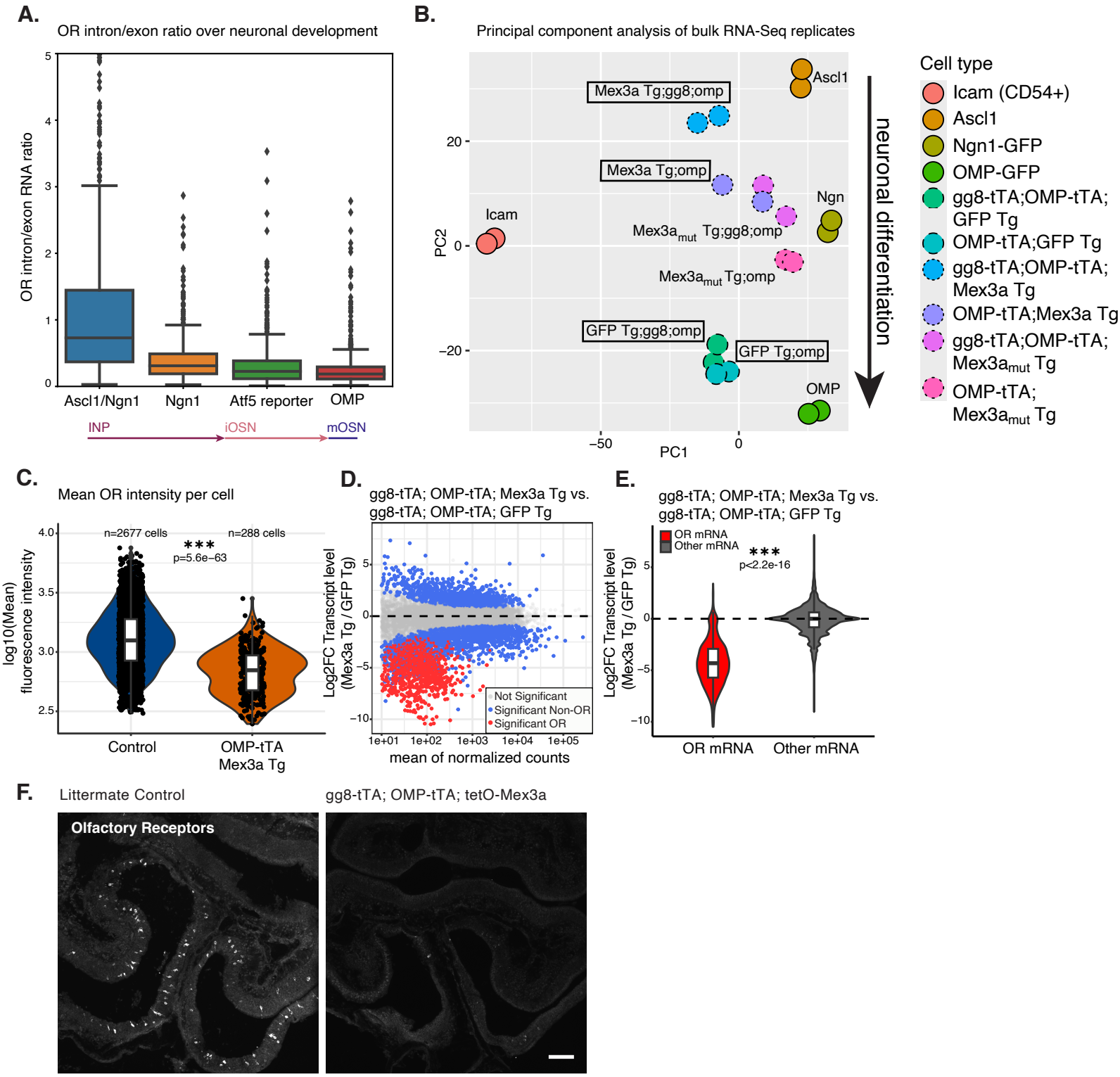

A.
OR intron/exon ratio over neuronal development
OR intron/exon RNA ratio
Ascl1/Ngn1
Ngn1
Atf5 reporter
OMP
INP
iOSN
mOSN
B.
Principal component analysis of bulk RNA-Seq replicates
Ascl1
Mex3a Tg;gg8;omp
Mex3a Tg;omp
Ngn
Icam
Mex3amut Tg;gg8;omp
Mex3amut Tg;omp
GFP Tg;gg8;omp
GFP Tg;omp
OMP
PC1
PC2
neuronal differentiation
Cell type
Icam (CD54+)
Ascl1
Ngn1-GFP
OMP-GFP
gg8-tTA;OMP-tTA; GFP Tg
OMP-tTA;GFP Tg
gg8-tTA;OMP-tTA; Mex3a Tg
OMP-tTA;Mex3a Tg
gg8-tTA;OMP-tTA; Mex3amut Tg
OMP-tTA; Mex3amut Tg
C.
Mean OR intensity per cell
log10(Mean) fluorescence intensity
n=2677 cells
n=288 cells
***
p=5.6e−63
Control
OMP-tTA Mex3a Tg
D.
gg8-tTA; OMP-tTA; Mex3a Tg vs. gg8-tTA; OMP-tTA; GFP Tg
Log2FC Transcript level (Mex3a Tg / GFP Tg)
Not Significant
Significant Non-OR
Significant OR
mean of normalized counts
E.
gg8-tTA; OMP-tTA; Mex3a Tg vs. gg8-tTA; OMP-tTA; GFP Tg
OR mRNA
Other mRNA
***
p<2.2e-16
Log2FC Transcript level (Mex3a Tg / GFP Tg)
OR mRNA
Other mRNA
F.
Littermate Control
gg8-tTA; OMP-tTA; tetO-Mex3a
Olfactory Receptors

Figure S1: Mex3a silences OR expression in olfactory neurons, related to Figure 1

(A) Ratios between OR introns and exons for all ORs that are expressed through the neuronal lineage in WT cells isolated by genetic labeling of distinct olfactory sensory neuron cell types (same RNA-Seq dataset at shown in Figure 1B). Whiskers extend to most extreme value within 1.5 times the inter-quartile range, center line is the median. **(B)** PCA plot of bulk RNA-Seq samples for Mex3a Tg experiments. Two biological replicates per sample. Sorted cells from the MOE using CD54 antibody (Icam) or endogenous labeling (GFP for Ngn1-GFP, OMP-GFP, and GFP Tg; tdTomato for Ascl1, mCherry for Mex3a Tg and $Mex3a_{mut}$ Tg). **(C)** Violin plot quantifying mean OR IF fluorescence intensity per OR^+ cell (dots) in IF images from littermate control vs OMP-tTA; Mex3a Tg genotypes. N is equal to the number of cells quantified for each genotype. Two or more biological replicates per genotype. Statistics, Welsh two sample t-test. **(D)** MA Plot of differentially expressed genes in gg8tTA; OMP-tTA; Mex3a Tg compared to gg8-tTA; OMP-tTA; GFP Tg. Significant ORs (red dots), Significant non-ORs (blue dots). **(E)** Violin Plot quantifying OR mRNA and non-OR mRNA in Mex3a Tg (gg8 and OMP-tTA drivers) compared to GFP Tg (gg8 and OMP-tTA drivers). Statistics, Welch Two Sample t-test. **(F)** Representative immunofluorescence image from four biological replicates per genotype, scale bar 100μM. For all statistical tests in this figure: $P<0.05$ = *, $P<0.01$=**, $P<0.001$=***.

Figure S2

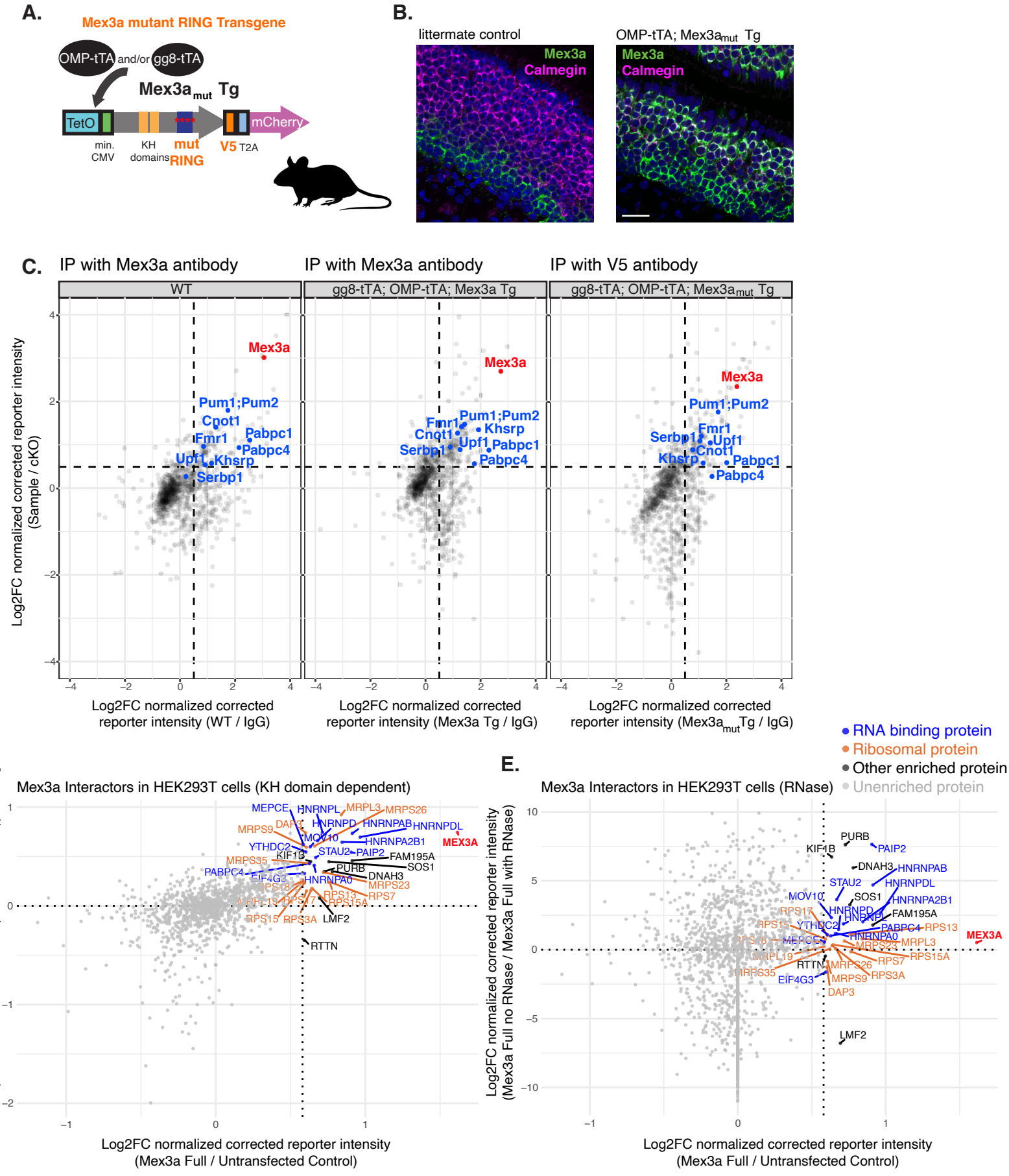

A.
Mex3a mutant RING Transgene
OMP-tTA and/or gg8-tTA
Mex3a mut Tg
TetO
mCherry
min. CMV
KH domains
mut RING
V5 T2A
B.
littermate control
OMP-tTA; Mex3a mut Tg
Mex3a
Calmegin
C.
IP with Mex3a antibody
IP with Mex3a antibody
IP with V5 antibody
WT
gg8-tTA; OMP-tTA; Mex3a Tg
gg8-tTA; OMP-tTA; Mex3a mut Tg
Log2FC normalized corrected reporter intensity (Sample / cKO)
Log2FC normalized corrected reporter intensity (WT / IgG)
Log2FC normalized corrected reporter intensity (Mex3a Tg / IgG)
Log2FC normalized corrected reporter intensity (Mex3a mut Tg / IgG)
Mex3a
Pum1;Pum2
Cnot1
Fmr1
Pabpc1
Pabpc4
Upf1
Khsrp
Serbp1
RNA binding protein
Ribosomal protein
Other enriched protein
Unenriched protein
D.
Mex3a Interactors in HEK293T cells (KH domain dependent)
Log2FC normalized corrected reporter intensity (Mex3a Full / Mex3aRING (no KH domains))
Log2FC normalized corrected reporter intensity (Mex3a Full / Untransfected Control)
E.
Mex3a Interactors in HEK293T cells (RNase)
Log2FC normalized corrected reporter intensity (Mex3a Full no RNase / Mex3a Full with RNase)
Log2FC normalized corrected reporter intensity (Mex3a Full / Untransfected Control)
MEPCE
HNRNPL
MRPL3
MRPS26
DAP3
HNRNPD
HNRNPAB
HNRNPDL
MRPS9
MOV10
HNRNPA2B1
MEX3A
YTHDC2
STAU2
PAIP2
FAM195A
KIF1B
MRPS35
SOS1
PABPC4
PURB
DNAH3
EIF4G3
HNRNPA0
MRPS23
RPS13
RPS7
RPS15A
RPS15
RPS3A
LMF2
RTTN

Figure S2: Mex3a interactome identifies proteins with known post-transcriptional roles, related to Figure 2

(A) Design of the Mex3a$_{mut}$ Tg allele. Four point-mutations in the RING domain render the protein unable to bind the zinc required for ubiquitination activity. A V5 allele was added to facilitate Immunopurification experiments. **(B)** Representative Immunofluorescence image from the gg8-tTA; OMP-tTA; Mex3a$_{mut}$ Tg mouse compared to a littermate control. Mex3a in green, Calmegin (ER marker that is expressed in mOSNs) in magenta, DAPI in blue. Scalebar 25μM. **(C)** Scatterplot showing mean enrichment in Mex3a/V5 Immunopurifications from two biological replicates of 1) WT 2) gg8-tTA; OMP-tTA; Mex3a Tg 3) gg8-tTA; OMP-tTA; Mex3a$_{mut}$ Tg. Mean enrichment compared to IgG controls shown on the x-axis, mean enrichment compared to Mex3a immunopurification from Mex3a cKO samples shown on the y-axis. Each dot is a protein, blue labels highlight known post-transcriptional regulators/RNA-binding proteins. Red label highlights Mex3a. Dotted line show cut-offs used in analysis to generate Figure 2F. **(D)** Scatterplot of peptides quantified after Immuno-purification followed by Mass Spectrometry in transiently transfected HEK293T cells. Cells were transfected with Full length Mex3a-V5 or truncated Mex3a-V5 (Mex3a V423-S653) which harbors the RING domain but no KH domains. Peptides were considered enriched if they were 0.58 Log Fold more enriched in the Mex3a Full length context compared to the untransfected control samples (no V5 tag), depicted on the x-axis (Log2 fold change normalized corrected reporter intensity). MEX3A is colored in red, RNA binding proteins colored in blue, and ribosomal proteins are colored in orange. Other enriched proteins labeled in black. The y-axis shows Log2 Fold change normalized corrected reporter intensity in the Mex3a Full length transfected cells relative to the Mex3a RING (no KH domains) transfected cells. N is equal to two biological replicates for each condition. **(E)** Scatterplot from experiment described in (S2D). The y-axis depicts Log2 Fold change normalized corrected reporter intensity of Mex3a Full length transfected cells with no RNase treatment relative to Mex3a Full length transfected cells with RNase treatment. Peptides are colored as in (d). N is equal to two biological replicates for each condition

Figure S3

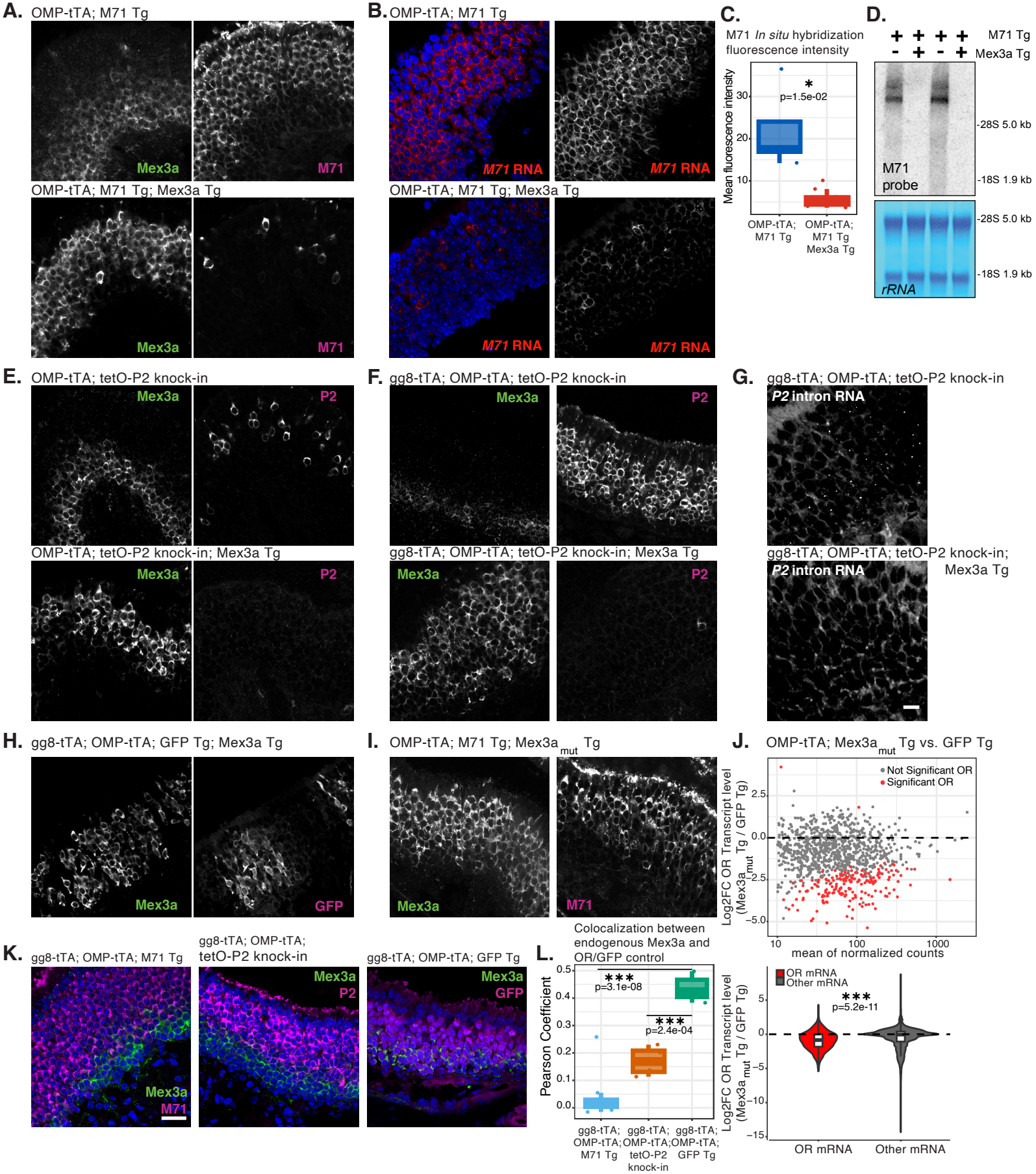

A.
OMP-tTA; M71 Tg
Mex3a
M71
OMP-tTA; M71 Tg; Mex3a Tg
Mex3a
M71
B.
OMP-tTA; M71 Tg
M71 RNA
M71 RNA
OMP-tTA; M71 Tg; Mex3a Tg
M71 RNA
M71 RNA
C.
M71 In situ hybridization fluorescence intensity
Mean fluorescence intensity
30
20
10
*
p=1.5e-02
OMP-tTA; M71 Tg
OMP-tTA; M71 Tg Mex3a Tg
D.
+ + + + M71 Tg
- + - + Mex3a Tg
M71 probe
-28S 5.0 kb
-18S 1.9 kb
rRNA
-28S 5.0 kb
-18S 1.9 kb
E.
OMP-tTA; tetO-P2 knock-in
Mex3a
P2
OMP-tTA; tetO-P2 knock-in; Mex3a Tg
Mex3a
P2
F.
gg8-tTA; OMP-tTA; tetO-P2 knock-in
Mex3a
P2
gg8-tTA; OMP-tTA; tetO-P2 knock-in; Mex3a Tg
Mex3a
P2
G.
gg8-tTA; OMP-tTA; tetO-P2 knock-in
P2 intron RNA
gg8-tTA; OMP-tTA; tetO-P2 knock-in; Mex3a Tg
P2 intron RNA
H.
gg8-tTA; OMP-tTA; GFP Tg; Mex3a Tg
Mex3a
GFP
I.
OMP-tTA; M71 Tg; Mex3a mut Tg
Mex3a
M71
J.
OMP-tTA; Mex3a mut Tg vs. GFP Tg
Not Significant OR
Significant OR
Log2FC OR Transcript level (Mex3a mut Tg / GFP Tg)
2.5
0.0
-2.5
-5.0
10
100
1000
mean of normalized counts
K.
gg8-tTA; OMP-tTA; M71 Tg
Mex3a
M71
gg8-tTA; OMP-tTA; tetO-P2 knock-in
Mex3a
P2
gg8-tTA; OMP-tTA; GFP Tg
Mex3a
GFP
L.
Colocalization between endogenous Mex3a and OR/GFP control
Pearson Coefficient
0.5
0.4
0.3
0.2
0.1
0.0
***
p=3.1e-08
***
p=2.4e-04
gg8-tTA; OMP-tTA; M71 Tg
gg8-tTA; OMP-tTA; tetO-P2 knock-in
gg8-tTA; OMP-tTA; GFP Tg
OR mRNA
Other mRNA
***
p=5.2e-11
Log2FC OR Transcript level (Mex3a mut Tg / GFP Tg)
5
0
-5
-10
-15
OR mRNA
Other mRNA

Figure S3: Post-transcriptional silencing of Olfactory Receptors with the Mex3a Tg, related to Figure3

(A) Grayscale images of M71 and Mex3a channels shown in Figure 3B. **(B)** RNA *In situ* hybridization using a probe for M71 RNA (left, Red, right, grayscale of same image) in OMP-tTA; M71 Tg and OMP-tTA; M71 Tg; Mex3a Tg. **(C)** Boxplot quantifying mean M71 ISH signal per image. N is equal to 5 images (M71 Tg alone) and 9 images (M71 Tg; Mex3a Tg). Whiskers extend to most extreme value within 1.5 times the inter-quartile range, center line is the median. Statistics, Welsh two sample t-test. **(D)** Northern blot using a probe against the M71 CDS in two biological replicates of OMP-tTA M71 Tg or two biological replicates of OMP-tTA M71 Tg; Mex3a Tg. rRNA 18S and 28S bands are shown as control of RNA integrity and as size marker. **(E)** Grayscale images P2 and Mex3a channels shown in Figure 3C. **(F)** Grayscale images P2 and Mex3a channels shown in Figure 3D. **(G)** Representative RNA FISH image against the tetO-P2 knock-in intron in gg8-tTA; OMP-tTA; tetO-P2 knock-in and gg8-tTA; OMP-tTA; tetO-P2 knock-in; Mex3a Tg samples. Scalebar 10μM. **(H)** Grayscale images GFP and Mex3a channels shown in Figure 3F. **(I)** Grayscale images M71 and Mex3a channels shown in Figure 3H. **(J)** Top, MA plot of OR mRNA from RNA-Seq of sorted mCherry or GFP^+ cells from OMP-tTA; Mex3a Mut Tg and OMP-tTA; GFP Tg, respectively. Bottom, violin plot of OR mRNAs and all other mRNAs quantified from the same RNA-Seq experiment. N is equal to two biological replicates. **(K)** Immunofluorescence image of OR/GFP (magenta) and endogenous (non-transgenic) Mex3a (green) for three genotypes 1) gg8-tTA; OMP-tTA; M71 Tg 2) gg8-tTA; OMP-tTA; tetO-P2 knock-in 3) gg8-tTA; OMP-tTA; GFP Tg. Scalebar 25μM, same for all images in this figure except (S3G). **(L)** Pearson correlations measuring colocalization between OR/GFP channels and endogenous Mex3a. Whiskers as for S3C. Statistics, Welsh two sample t-test. For all statistical tests in this figure: $P<0.05$ = *, $P<0.01$=**, $P<0.001$=***.

Figure S4

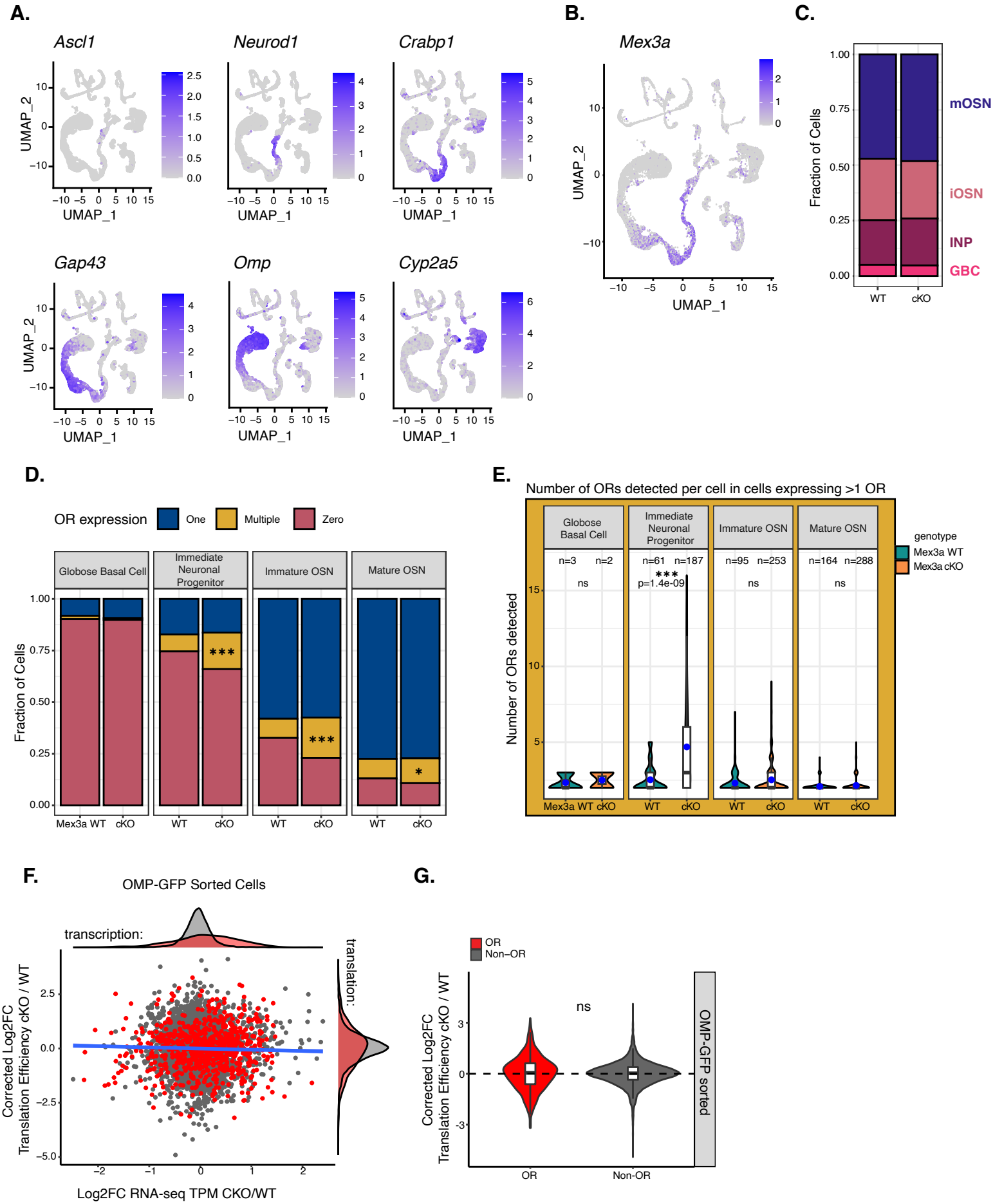

A.
Ascl1
Neurod1
Crabp1
Gap43
Omp
Cyp2a5
UMAP_1
UMAP_2
B.
Mex3a
C.
Fraction of Cells
mOSN
iOSN
INP
GBC
WT
cKO
D.
OR expression
One
Multiple
Zero
Globose Basal Cell
Immediate Neuronal Progenitor
Immature OSN
Mature OSN
Fraction of Cells
Mex3a WT
cKO
E.
Number of ORs detected per cell in cells expressing >1 OR
Number of ORs detected
genotype
Mex3a WT
Mex3a cKO
n=3
n=2
ns
n=61
n=187
p=1.4e-09
n=95
n=253
n=164
n=288
F.
OMP-GFP Sorted Cells
transcription:
translation:
Corrected Log2FC Translation Efficiency cKO / WT
Log2FC RNA-seq TPM CKO/WT
G.
OR
Non–OR
Corrected Log2FC Translation Efficiency cKO / WT
OMP-GFP sorted

Figure S4: Premature OR expression upon loss of Mex3a, related to Figure 4

(A) UMAP plot of single cell RNA-Seq data from two replicates of Mex3a cKO and Mex3a WT littermate controls, whole MOE at PN12. Known markers are highlighted to show cell types (*Ascl1*, globose basal cell; *Neurod1*, immediate neuronal progenitor; *Crabp1*, immediate neuronal progenitor; *Gap43*, immature olfactory sensory neuron; *Omp*, mature olfactory sensory neuron; *Cyp2a5*, sustentacular) **(B)** UMAP projection showing Mex3a expression pattern in single cell RNA-Seq data. **(C)** Stacked barplot classifying Mex3a WT and Mex3a cKO single cells in the neuronal lineage as GBC, INP, iOSN, or mOSN cells **(D)** Stacked barplot classifying GBC, INP, iOSN, and mOSN single cells by whether they are expressing zero, one, or more than one OR per cell. Threshold of >1 normalized count over an OR to be considered expressed. Statistics, Fisher's Exact Test: testing proportion of cells with multiple ORs per cell comparing Mex3a cKO and Mex3a WT. GBC, p=0.65. INP, p= 6.0e-09. iOSN, p= 3.2e-12. mOSN, p= 1.1e-2 **(E)** Quantification of number of ORs detected per cell in cells from the category "multiple OR per cell" from yellow group in (S4D). N equals number of cells expressing more than one OR in each category. **(F)** Scatter plot comparing Translational Efficiency (y-axis) and RNA transcript per million (x-axis) in two replicates of Mex3a cKO vs Mex3a WT OMP-GFP sorted cells. Red dots, OR RNAs, gray dots, non-OR RNAs. Blue line, linear regression line. Marginal density plots summarize OR gene family (red) or all other genes (gray) Translational Efficiency (y-axis) or RNA expression levels (x-axis). **(G)** Violin Plot quantifying translational efficiency of OR mRNAs (red) compared to non-OR mRNAs (gray) in two replicates of Mex3a cKO vs Mex3a WT OMP-GFP sorted cells. Statistics calculated with Welch two sample t-test. For all statistical tests in this figure: $P<0.05$ = *, $P<0.01$=**, $P<0.001$=***.

Figure S5

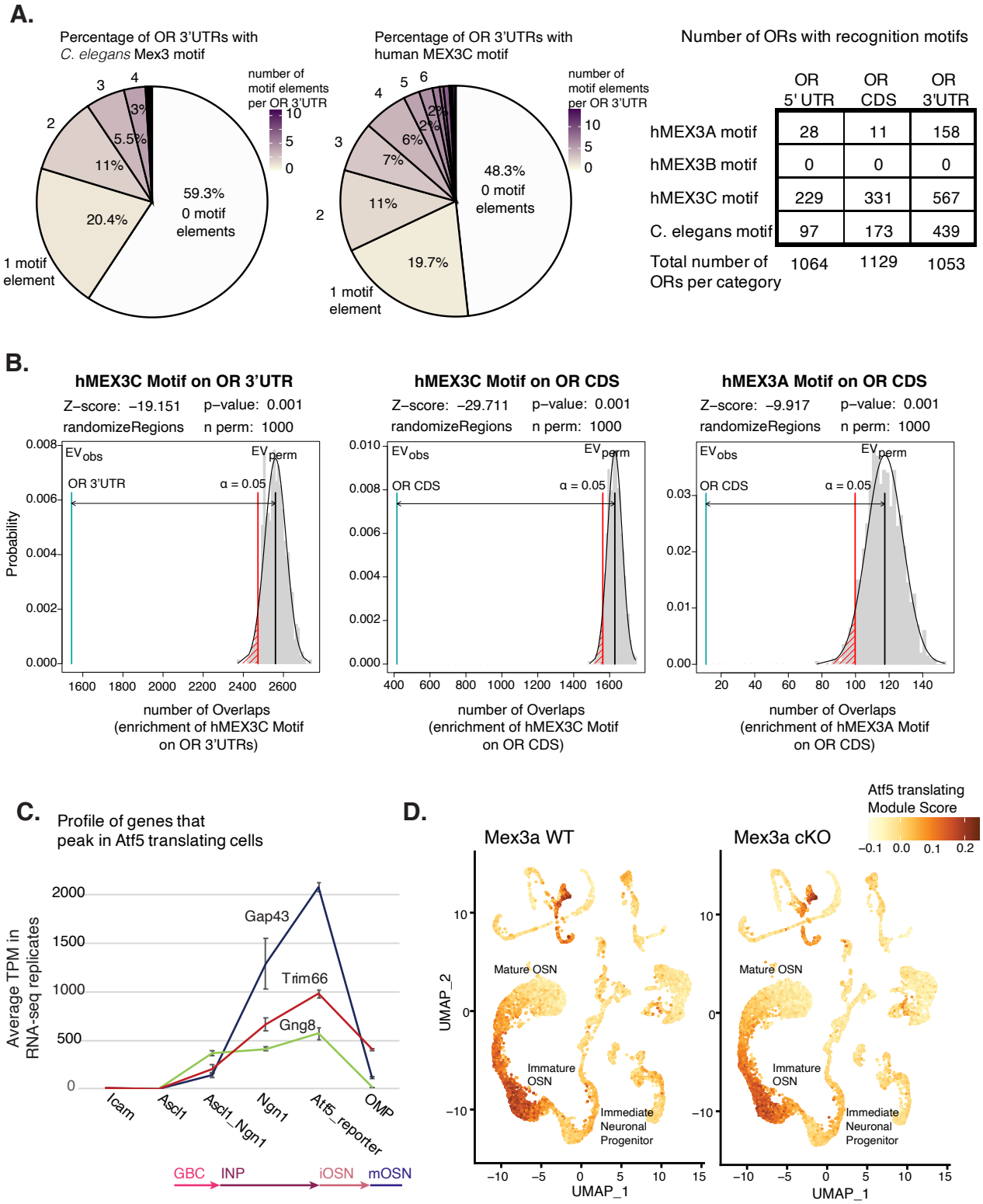
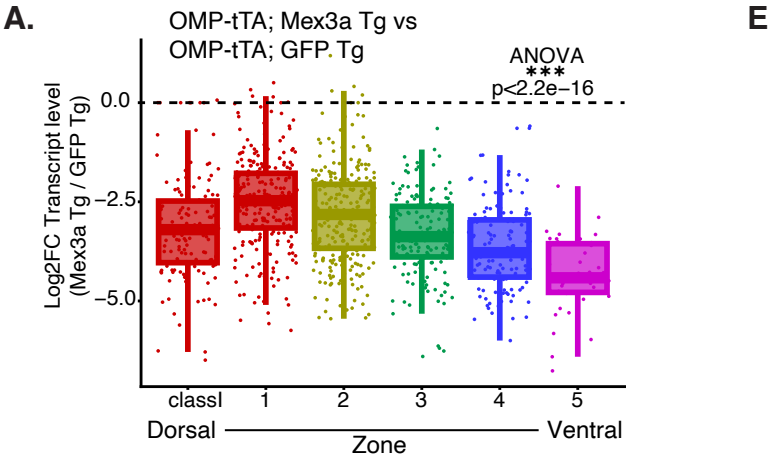
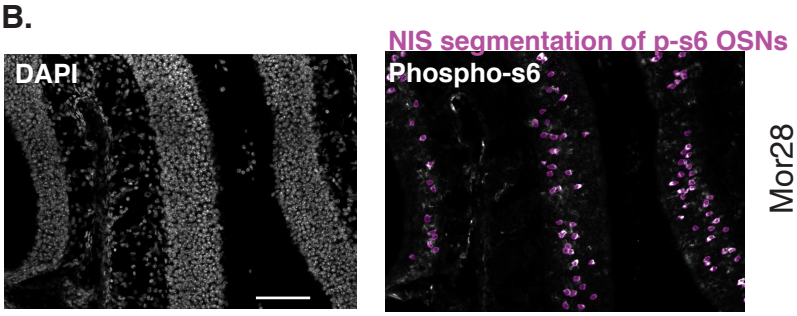

A.
Percentage of OR 3'UTRs with C. elegans Mex3 motif
number of motif elements per OR 3'UTR
59.3% 0 motif elements
20.4% 1 motif element
11%
5.5%
3%
Percentage of OR 3'UTRs with human MEX3C motif
48.3% 0 motif elements
19.7% 1 motif element
11%
7%
6%
2%
2%
Number of ORs with recognition motifs
OR 5' UTR | OR CDS | OR 3'UTR
hMEX3A motif | 28 | 11 | 158
hMEX3B motif | 0 | 0 | 0
hMEX3C motif | 229 | 331 | 567
C. elegans motif | 97 | 173 | 439
Total number of ORs per category | 1064 | 1129 | 1053
B.
hMEX3C Motif on OR 3'UTR
Z–score: –19.151 p–value: 0.001
randomizeRegions n perm: 1000
hMEX3C Motif on OR CDS
Z–score: –29.711 p–value: 0.001
randomizeRegions n perm: 1000
hMEX3A Motif on OR CDS
Z–score: –9.917 p–value: 0.001
randomizeRegions n perm: 1000
Probability
EV obs
EV perm
OR 3'UTR
OR CDS
α = 0.05
number of Overlaps (enrichment of hMEX3C Motif on OR 3'UTRs)
number of Overlaps (enrichment of hMEX3C Motif on OR CDS)
number of Overlaps (enrichment of hMEX3A Motif on OR CDS)
C.
Profile of genes that peak in Atf5 translating cells
Average TPM in RNA-seq replicates
Gap43
Trim66
Gng8
Icam
Ascl1
Ascl1_Ngn1
Ngn1
Atf5_reporter
OMP
GBC
INP
iOSN
mOSN
D.
Atf5 translating Module Score
Mex3a WT
Mex3a cKO
Mature OSN
Immature OSN
Immediate Neuronal Progenitor
UMAP_1
UMAP_2

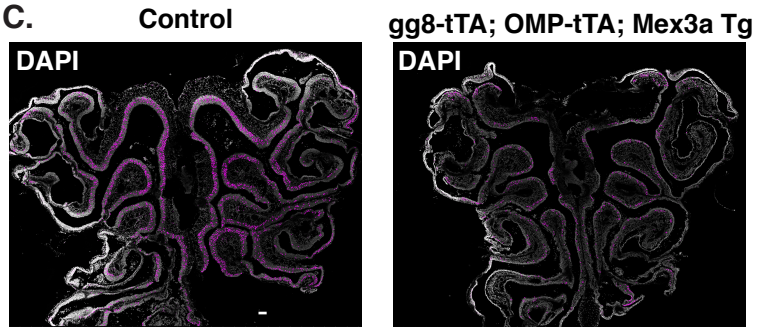
Figure S5: OR genes are depleted for Mex3 family RNA binding motifs, and Perk-induced transcription factor Atf5 and targets exhibit increased expression in the Mex3a cKO. related to Figure5

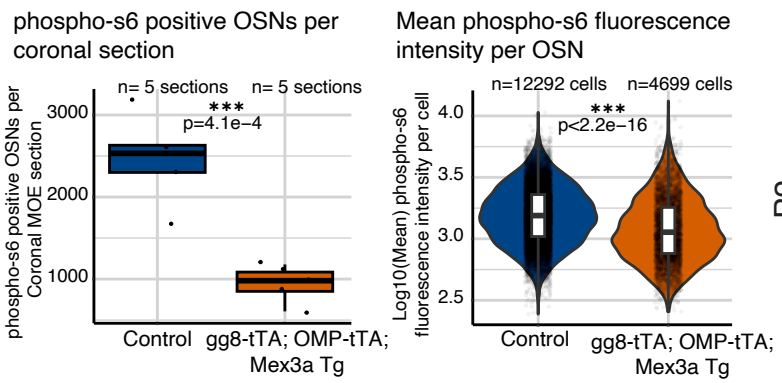
(A) Piecharts depicting number of *C. elegans* Mex3 (left) or human MEX3C (center) RNA binding motifs per OR 3'UTR. Right, chart delineating number of mouse OR sequences containing motifs for human MEX3A, MEX3B, MEX3C, or the C. elegans Mex-3 RNA recognition motifs. EMBOSS fuzznuc was used to search mm10 OR genes. **(B)** RegioneR was used to perform a permutation test assessing the overlap between OR 3'UTRs and OR CDS and human MEX3C or human MEX3A motifs. Overlap between random size-matched reference transcriptome (mm10) fragments and the MEX3C or MEX3A motif used as a negative control. **(C)** Line graph showing three examples of genes whose expression peaks in Atf5 translating cells. DESeq2 was used to identify 318 genes that follow this expression pattern and were used in the "Atf5 translating" module score analysis shown in Figure 5C. Error bars, standard error, two biological replicates per sorted cell sample. See also Table S3. **(D)** UMAP projection of Atf5 translating scores in Mex3a cKO and WT littermate controls. Higher score means the cell is more likely to be actively translating Atf5. As expected, "Atf5 translating" score peaks in the immature OSN stage.

Figure S6


A.
OMP-tTA; Mex3a Tg vs
OMP-tTA; GFP Tg
ANOVA
***
p<2.2e−16
Log2FC Transcript level
(Mex3a Tg / GFP Tg)
0.0
−2.5
−5.0
classI
1
2
3
4
5
Dorsal
Zone
Ventral


B.
NIS segmentation of p-s6 OSNs
DAPI
Phospho-s6


C.
Control
gg8-tTA; OMP-tTA; Mex3a Tg
DAPI
DAPI
NIS segmentation of p-s6 OSNs


D.
phospho-s6 positive OSNs per
coronal section
n= 5 sections
n= 5 sections
***
p=4.1e−4
phospho-s6 positive OSNs per
Coronal MOE section
3000
2000
1000
Control
gg8-tTA; OMP-tTA;
Mex3a Tg
Mean phospho-s6 fluorescence
intensity per OSN
n=12292 cells
n=4699 cells
***
p<2.2e−16
Log10(Mean) phospho-s6
fluorescence intensity per cell
4.0
3.5
3.0
2.5
Control
gg8-tTA; OMP-tTA;
Mex3a Tg

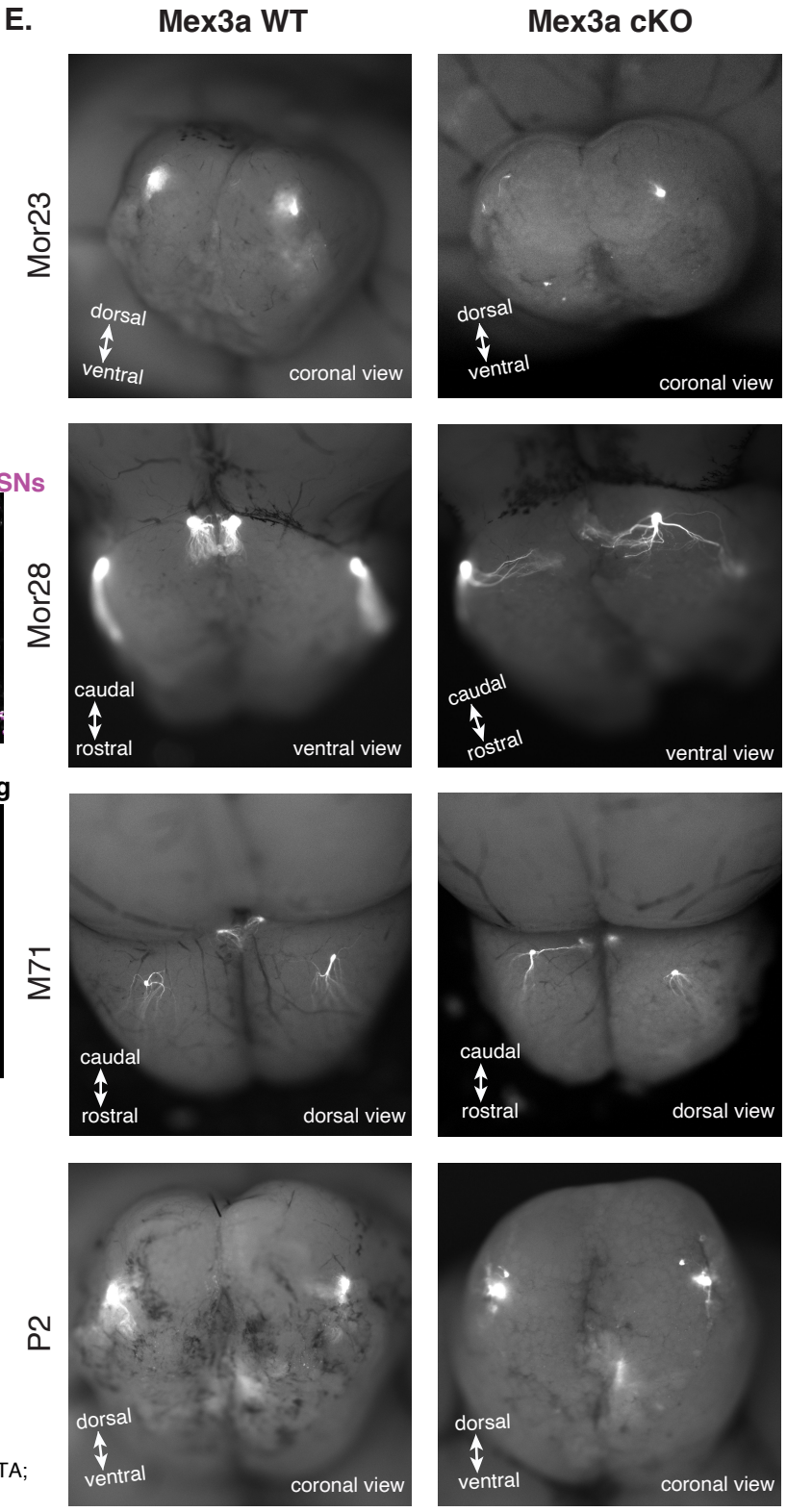

E.
Mex3a WT
Mex3a cKO
Mor23
Mor28
M71
P2
dorsal
ventral
coronal view
caudal
rostral
ventral view
dorsal view

Figure S6: Mex3a cKO biases OR gene choice and disrupts axon guidance specificity, and Mex3a Tg leads to decreased neuronal activity related to Figure 6

(A) Boxplot quantifying zonal OR expression in bulk RNA-Seq from sorted mCherry or GFP^+ cells from OMP-tTA; Mex3a Tg or OMP-tTA; GFP Tg, respectively. Whiskers extend to most extreme value within 1.5 times the inter-quartile range, center line is the median. N is equal to two biological replicates per genotype. Statistics, ANOVA one-way test. **(B)** Representative image of NIS segmentation of phosphorylated S6 ribosomal protein (Ser 244/Ser 247) (p-S6) positive OSNs from mice exposed to the odorant acetophenone. Left, DAPI. Right, immunofluorescence of p-S6 shown in white, with NIS segmentation shown in purple. Scalebar, 100μM **(C)** Coronal sections of MOE with DAPI (white) and NIS segmentation of p-S6 positive OSNs (purple). Littermate control on left and gg8-tTA; OMP-tTA; Mex3a Tg sample shown on right. Scalebar, 100μM. **(D)** Left, boxplot depicting the number of p-S6 positive OSNs per coronal section. N is equal to five coronal sections per genotype, two biological replicates per genotype. Whiskers as Figure S6A. Statistics, Two sample t-test. Right, violin plot quantifying mean fluorescence intensity per p-S6 positive OSN. N is equal to 12,292 cells for control and 4,699 cells for gg8-tTA; OMP-tTA; Mex3a Tg genotype. Statistics, Welch two sample t-test. **(E)** Representative whole mount images of olfactory bulbs from Mex3a WT (left panels) and Mex3a cKO mice (right panels). View of the olfactory bulb is labeled, as OR glomeruli have distinct stereotyped positions. Mor23 Mex3a WT (35 mice), Mex3a cKO (12 mice). Mor28 Mex3a WT (22 mice), Mex3a cKO (41 mice). M71 Mex3a WT (32 mice), Mex3a cKO (27 mice). P2 Mex3a WT (34 mice), Mex3a cKO (17 mice). For all statistical tests in this figure: $P<0.05$ = *, $P<0.01$=**, $P<0.001$=***.

Supplemental Data S1: Code used to generate figures

```
####
#### This code was written by R. Duffié, H. Shayya, and M. Wang
#### It is organized by figure and sometimes refers to a previous figure if similar code was used for multiple figures
################################################################################################
################################################################################################
##########################Figure1A:
################################################################################################
################################################################################################
#single nucleus and single cell RNA Seq, WT genotype only
#SINGLE NUCLEUS data are from Pourmorady et al. 2023 https://doi.org/10.1038/s41586-023-06845-4
#!/usr/bin/env python
# coding: utf-8

###input cell id, R1, cluster 0 is mOSN, cluster 4 is INP and iOSN
import pandas as pd

df1 = pd.read_csv("cluster_cell_R2.csv.gz", compression='gzip')
df1= df1.rename(columns={"Unnamed: 0": "cellID"})
df1['cellID'] = df1['cellID'].replace({'-1' :'.1'},regex=True)

##subset 0 and 4
mOSN = df1.loc[ (df1['x']==2) | (df1['x']==5)]
iOSN =  df1.loc[ (df1['x']==0) ]
INP = df1.loc[ (df1['x']==3) ]
print(mOSN.head())
##cover cell id into a list
mOSN_ID = mOSN['cellID'].tolist()
iOSN_ID = iOSN['cellID'].tolist()
INP_ID = INP['cellID'].tolist()
print(len(mOSN_ID),len(iOSN_ID),len(INP_ID))
#INP_iOSN_ID

##input RNA count matrix
gene_count = pd.read_csv("genecount_matrix_R2.csv.gz", sep=",",compression='gzip')


##change first column name and index
gene_count = gene_count.rename(columns={'Unnamed: 0':"Gene name"})
gene_count = gene_count.set_index("Gene name")


###subset cell id in mOSN
mOSN = gene_count.loc[:,gene_count.columns.isin(mOSN_ID)]
##subset INP/iOSN
iOSN = gene_count.loc[:,gene_count.columns.isin(iOSN_ID)]
INP = gene_count.loc[:,gene_count.columns.isin(INP_ID)]

##input transcript length, keep the longest one when a gene has multiple transcripts
tl = pd.read_csv("transcript_length.txt",sep='\t')
tl = tl.sort_values(by='Transcript length (including UTRs and CDS)', ascending = False )
tl = tl.drop_duplicates(subset=['Gene name'],keep='first')

##add transcript lenght
```

```
mOSN1 = pd.merge(tl, mOSN, on='Gene name')
iOSN1 = pd.merge(tl, iOSN, on='Gene name')
INP1 = pd.merge(tl, INP, on='Gene name')

######Normalize the total number of reads and transcripts length,
# Initialize a dataframe to store TPM values
tpm_values_mOSN = pd.DataFrame()
tpm_values_mOSN['Gene name'] = mOSN1['Gene name']
# Calculate TPM for each sample
for sample in mOSN1.columns[2:]:  # Skip the first column which is gene_id and second column is
transcript length
  # Calculate RPK (Reads Per Kilobase)
  mOSN1['RPK'] = mOSN1[sample] / (mOSN1['Transcript length (including UTRs and CDS)'] / 1000)
# Calculate the sum of all RPK values
sum_rpk = mOSN1['RPK'].sum()
# Calculate TPM (Transcripts Per Million)
tpm_values_mOSN[sample] = (mOSN1['RPK'] / sum_rpk) * 1e6
# Save the result to a new file
tpm_values_mOSN.to_csv('tpm_mOSN.txt', sep='\t', index=False)

# Initialize a dataframe to store TPM values
tpm_values_iOSN = pd.DataFrame()
tpm_values_iOSN['Gene name'] = iOSN1['Gene name']
#print(wt11[2:])
# Calculate TPM for each sample
for sample in iOSN1.columns[2:]:  # Skip the first column which is gene_id and second column is
transcript length
  # Calculate RPK (Reads Per Kilobase)
  iOSN1['RPK'] = iOSN1[sample] / (iOSN1['Transcript length (including UTRs and CDS)'] / 1000)
# Calculate the sum of all RPK values
sum_rpk = iOSN1['RPK'].sum()
# Calculate TPM (Transcripts Per Million)
tpm_values_iOSN[sample] = (iOSN1['RPK'] / sum_rpk) * 1e6
# Save the result to a new file
tpm_values_iOSN.to_csv('tpm_iOSN.txt', sep='\t', index=False)

tpm_values_INP = pd.DataFrame()
tpm_values_INP['Gene name'] = INP1['Gene name']
#print(wt11[2:])
# Calculate TPM for each sample
for sample in INP1.columns[2:]:  # Skip the first column which is gene_id and second column is transcript
length
  # Calculate RPK (Reads Per Kilobase)
  INP1['RPK'] = INP1[sample] / (INP1['Transcript length (including UTRs and CDS)'] / 1000)
# Calculate the sum of all RPK values
sum_rpk = INP1['RPK'].sum()
# Calculate TPM (Transcripts Per Million)
tpm_values_INP[sample] = (INP1['RPK'] / sum_rpk) * 1e6
# Save the result to a new file
tpm_values_INP.to_csv('tpm_INP.txt', sep='\t', index=False)

#subset OR genes
mOSN_OR_rpkm = tpm_values_mOSN[tpm_values_mOSN['Gene name'].str.startswith('Olfr')]
iOSN_OR_rpkm = tpm_values_iOSN[tpm_values_iOSN['Gene name'].str.startswith('Olfr')]
INP_OR_rpkm =  tpm_values_INP[tpm_values_INP['Gene name'].str.startswith('Olfr')]
mOSN_OR_rpkm = mOSN_OR_rpkm.set_index('Gene name')
```

```
iOSN_OR_rpkm = iOSN_OR_rpkm.set_index('Gene name')
INP_OR_rpkm = INP_OR_rpkm.set_index('Gene name')

#sum TPM of all OR in each cell
mOSN_ORrpkm_percell = mOSN_OR_rpkm.sum(axis=0)
mOSN_ORrpkm_percell_df = pd.DataFrame(mOSN_ORrpkm_percell, columns=['Total_ORrpkm'])
##drop index and add cell type
mOSN_ORrpkm_percell_df.reset_index(drop=True, inplace = True)
mOSN_ORrpkm_percell_df['CellType'] = 'mOSN'

###sum RPKM of all OR in each iOSN cell
iOSN_ORrpkm_percell = iOSN_OR_rpkm.sum(axis=0)
iOSN_ORrpkm_percell_df = pd.DataFrame(iOSN_ORrpkm_percell, columns=['Total_ORrpkm'])
##drop index and add cell type
iOSN_ORrpkm_percell_df.reset_index(drop=True, inplace = True)
iOSN_ORrpkm_percell_df['CellType'] = 'iOSN'

#sum TPM of all OR in each INP
INP_ORrpkm_percell = INP_OR_rpkm.sum(axis=0)
INP_ORrpkm_percell_df = pd.DataFrame(INP_ORrpkm_percell, columns=['Total_ORrpkm'])
##drop index and add cell type
INP_ORrpkm_percell_df.reset_index(drop=True, inplace = True)
INP_ORrpkm_percell_df['CellType'] = 'INP'
INP_ORrpkm_percell_df

##merge two type into one dataframe and plot
import numpy as np
result = pd.concat([mOSN_ORrpkm_percell_df,iOSN_ORrpkm_percell_df, INP_ORrpkm_percell_df])
result['log10(Total_OR_TPM)'] = np.log10(result['Total_ORrpkm'].astype(np.float64) + 1)
###remove cell if RPKM of OR is 0
active= result[result['log10(Total_OR_TPM)'] != 0]
active.to_csv("Total_OR_TPM_INP-iOSN-mOSN_R2_drop0.txt", sep='\t',index=False)

#SINGLE CELL data are from this study, WT replicates
#!/usr/bin/env python
# coding: utf-8

##input RNA count matrix
import pandas as pd
wt1 = pd.read_csv("wt1_count.txt", sep="\t")
wt2 = pd.read_csv("wt2_count.txt", sep="\t")
wt1 = wt1.reset_index().rename(columns={'index': 'Gene name'})
wt2 = wt2.reset_index().rename(columns={'index': 'Gene name'})

##input transcript length, keep the longest one when a gene has multiple transcripts
tl = pd.read_csv("transcript_length.txt",sep='\t')
tl = tl.sort_values(by='Transcript length (including UTRs and CDS)', ascending = False )
tl = tl.drop_duplicates(subset=['Gene name'],keep='first')

###add  transcript length
wt11 = pd.merge(tl, wt1, on='Gene name')
wt22 = pd.merge(tl, wt2, on='Gene name')

# Initialize a dataframe to store TPM values
tpm_df = pd.DataFrame()
tpm_df['Gene name'] = wt11['Gene name']
```

```
#print(wt11[2:])
# Calculate TPM for each sample
for sample in wt11.columns[2:]:  # Skip the first column which is gene_id and second column is transcript length
  # Calculate RPK (Reads Per Kilobase)
  wt11['RPK'] = wt11[sample] / (wt11['Transcript length (including UTRs and CDS)'] / 1000)
# Calculate the sum of all RPK values
sum_rpk = wt11['RPK'].sum()
# Calculate TPM (Transcripts Per Million)
tpm_df[sample] = (wt11['RPK'] / sum_rpk) * 1e6

# Save the result to a new file
tpm_df.to_csv('tpm_wt1.txt', sep='\t', index=False)

##wt2
w2_tpm_df = pd.DataFrame()
w2_tpm_df['Gene name'] = wt22['Gene name']
# Calculate TPM for each sample
for sample in wt22.columns[2:]:  # Skip the first column which is gene_id and second column is transcript length
  # Calculate RPK (Reads Per Kilobase)
  wt22['RPK'] = wt22[sample] / (wt22['Transcript length (including UTRs and CDS)'] / 1000)
# Calculate the sum of all RPK values
sum_rpk = wt22['RPK'].sum()
# Calculate TPM (Transcripts Per Million)
w2_tpm_df[sample] = (wt22['RPK'] / sum_rpk) * 1e6

# Save the result to a new file
w2_tpm_df.to_csv('tpm_wt2.txt', sep='\t', index=False)

#subset OR genes
w1_OR_tpm = tpm_df[tpm_df['Gene name'].str.startswith('Olfr')]
w2_OR_tpm= w2_tpm_df[w2_tpm_df['Gene name'].str.startswith('Olfr')]

##change first column to index
w1_OR_tpm = w1_OR_tpm.set_index("Gene name")
w2_OR_tpm = w2_OR_tpm.set_index("Gene name")

#sum TPM of all OR in each cell
ORrpkm_percell = w1_OR_tpm.sum(axis=0)
ORrpkm_percell_df = pd.DataFrame(ORrpkm_percell, columns=['Total_ORrpkm'])
##drop index and add cell type
ORrpkm_percell_df['Group'] = 'WT1'
ORrpkm_percell_df = ORrpkm_percell_df.reset_index().rename(columns={'index': 'cell_id'})
print(ORrpkm_percell_df)
###add cell type
Type=pd.read_csv("cellType_id.txt",sep="\t")
Type['cell_id'] = Type['cell_id'].str.replace('-1', '.1')
Merge1 = ORrpkm_percell_df.merge(Type,on="cell_id")

###merge W2
#sum RPKM of all OR in each cell
print(w2_OR_tpm)
w2_ORrpkm_percell = w2_OR_tpm.sum(axis=0)
w2_ORrpkm_percell_df = pd.DataFrame(w2_ORrpkm_percell, columns=['Total_ORrpkm'])
##drop index and add cell type
```

```
w2_ORrpkm_percell_df['Group'] = 'WT2'
w2_ORrpkm_percell_df = w2_ORrpkm_percell_df.reset_index().rename(columns={'index': 'cell_id'})
Merge2 = w2_ORrpkm_percell_df.merge(Type,on="cell_id")

# In[ ]:

##merge two type into one dataframe and plot
import numpy as np
##Merge w1 and w2
Merge = pd.concat([Merge1,Merge2])
print(Merge)
Merge['log10(Total_ORrpkm)'] = np.log10(Merge['Total_ORrpkm'].astype(np.float64) + 1)
###remove cell if RPKM of OR is 0
active= Merge[Merge['log10(Total_ORrpkm)'] != 0]
active.to_csv("Total_TPMP_OR_INP-iOSN-mOSN_WT1_WT2_drop0.txt", sep='\t',index=False)

################################################################################################
################################################################################################
##########################Figure1B:
################################################################################################
################################################################################################

#Pre-processing RNA-Seq data, $1 is the name of the library, these libraries were made with the Takara
SMARTSeq mammalian pico input total RNA v2 kit:
#!/bin/bash
echo "Running fastqc..."
echo "fastqc $1.fastq.gz"
fastqc $1.R1.fastq.gz
fastqc $1.R2.fastq.gz
cutadapt --minimum-length 18 -a AGATCGGAAGAGCACACGTC -A AGATCGGAAGAGCGTCGTGT -o
$1.ca.R1.fastq.gz -p $1.ca.R2.fastq.gz  $1.R1.fastq.gz $1.R2.fastq.gz > cutadapt-report.$1.txt
fastqc $1.ca.R1.fastq.gz
fastqc $1.ca.R2.fastq.gz
#Align the reads with mm10
STAR --genomeDir /seq/mm10/indexes/STAR/  --runThreadN 8 --readFilesCommand zcat --readFilesIn
$1.ca.R1.fastq.gz $1.ca.R2.fastq.gz --outFileNamePrefix $1. --outFilterIntronMotifs RemoveNoncanonical
--clip5pNbases 6 --outFilterType BySJout --outFilterMultimapNmax 20 --outSAMtype BAM
SortedByCoordinate
#Post-process
#!/bin/bash
echo "Postprocessing aligned data..."
samtools index $1.Aligned.sortedByCoord.out.bam
samtools view -bq 30 $1.Aligned.sortedByCoord.out.bam -o $1.q30.bam
samtools index $1.q30.bam
#normalize to 1,000,000 50PE reads
bam2wig.py -i $1.q30.bam -s /seq/mm10/mm10.chrom.sizes -t 100000000 -o $1.q30.rnaseq -d "1+-,1-
+,2++,2--"
bam2wig.py -i $1.q30.bam -s /seq/mm10/mm10.chrom.sizes -t 100000000 -o $1.rnaseq

#DESeq2 analysis in R
setwd("/path/to/working/directory")
library(dplyr)
library(tidyverse)
library("DESeq2")
library("BiocParallel")
library("GenomicAlignments")
```

```
library("GenomicFeatures")
library("GenomicRanges")
library("Rsamtools")
library("AnnotationDbi")
library("rtracklayer")
library("org.Mm.eg.db")
library("genefilter")
library("ggrepel")
library(reshape2)
library("RColorBrewer")
library("pheatmap")
library(gridExtra)
library(fgsea)
library(cowplot)
library(grid)
theme_set(theme_cowplot())
#Functions
filter_genes <- function(x){
  x <- intersect(x, rownames(dds))
  return(x)
}
remove_ORs <- function(x) {
  x <- setdiff(x,Olfr)
  return(x)
}
firstColToRowName <- function(x) {
  rownames(x) <- x[,1]
  x[,1] <- NULL
  return(x)
}
#Load reference
genes <-import("/path/to/custom/gtf/mm10+additional_Olfr_annotation_from_Ibarra_Soria2017.gtf")
#Additional Olfr information from this paper: https://doi.org/10.7554/eLife.21476
genes_txdb <- makeTxDbFromGRanges(genes)
exonsByGene <- exonsBy(genes_txdb, by="gene")
sampleTable <- read.delim("SampleTable.with.paths.to.bams.txt",header =T)
bamfiles <- filenames <- file.path(sampleTable$data)
bamfiles <- BamFileList(bamfiles)
rna <- summarizeOverlaps(features=exonsByGene, reads=bamfiles, mode="Union", singleEnd=FALSE,
ignore.strand=FALSE, preprocess.reads=invertStrand)
colData(rna) <- DataFrame(sampleTable)
colnames(rna)<-sampleTable$library

# set field for comparing samples
dds <- DESeqDataSet(rna, design = ~ geno)  # adjust field used for design as needed
# Set levels
dds$geno <- factor(dds$geno)
# remove genes with low counts
dds <- dds[ rowSums(counts(dds)) > 0, ]
# differential expressed genes
dds <- DESeq(dds)
#Calculate FPKM and TPM
fpkm.table <- as.data.frame(fpkm(dds, robust = FALSE)) %>% rownames_to_column(var="GeneID")
fpkm <- fpkm.table %>% gather(condition,fpkm,-GeneID) %>%
separate(condition,sep="_",into=c("condition","geno"))
tpm_table <- fpkm.table %>% mutate_at(.vars = vars(-GeneID), .funs = function(x) {x/sum(x) * 1e6})
```

```
write.table(tpm_table, file="Name_of_experiment.txt",quote=FALSE)

##############################################################################################
##############################################################################################
#########################Figure1E:
##############################################################################################
##############################################################################################
### MA plot of Significantly DE genes, highlighting Olfrs
###bulk RNA-Seq libraries from this experiment were generated with the Illumina Truseq Stranded total
RNA kit
#Processing as for Figure 1B, generate dds, now calculate results between replicates of two genotypes
res <- results(dds, alpha=0.05,lfcThreshold = 0.5849,contrast= c("condition", "Mutant_condition",
"Control_condition"))
cutoff <- 0.05
df.res <- data.frame(res)
df.res <- rownames_to_column(df.res, var = 'gene_name')
Olfr <- filter_genes(scan(file = "/path/to/list/of/Olfrs.txt", what = character()))
df.res$Olfr <- ifelse(df.res$gene_name %in% Olfr, 'Olfactory_Receptor','Other')
df.res$significance <- ifelse(df.res$padj < cutoff,'Significant','Not_Significant')
#You want a unique mapping. We can add that as an additional variable
df.res$color_ <- ifelse(df.res$significance == 'Not_Significant', 'Not Significant', ifelse(df.res$Olfr ==
'Olfactory_Receptor', 'Significant OR','Significant Non-OR'))
#Now you can plot.
ggplot(data = df.res, aes(x = baseMean, y = log2FoldChange, color = color_, alpha = color_)) +
          geom_point() +
          theme_bw() +
          scale_color_manual(values = c('Not Significant' = 'gray','Significant OR' = 'firebrick1','Significant
        Non-OR' = 'royalblue2')) +
          scale_alpha_manual(values = c('Not Significant' = 0.1, 'Significant OR' = 1, 'Significant Non-OR'
        = 1))+
          scale_x_log10(name = 'mean of normalized counts', limits = c(10, NA)) +
          geom_hline(yintercept = 0, linetype = 'dashed', color = 'black')

##############################################################################################
##############################################################################################
#########################Figure1F:
##############################################################################################
##############################################################################################
###Figure 1F, Violin plot comparing Olfrs to non-Olfrs in Mex3a Tg compared to GFP Tg (OMPtTA
drivers)
###Make a violin plot that looks at Olfrs vs the rest of the genome. df.res is calculated using DESeq2 as
described for Fig 1E
ggplot(data=df.res,
              aes(x = Olfr, y = log2FoldChange, fill = Olfr)) +
          geom_violin() +
          scale_fill_manual(values=c("#FF0000", "#656565")) +
          geom_boxplot(fill = 'white', width = 0.1,outlier.shape = NA) +
          theme_classic() +
          geom_hline(yintercept = 0, linetype='dashed') +
          xlab('') + ylab('title')
#Calculate significance
var.test(log2FoldChange ~ Olfr, df.res)
#check significance, if p.val is significant, the variance is not equal, do not set var.equal to TRUE
t.test(log2FoldChange ~ Olfr, df.res)
```

```
##############################################################################################
##############################################################################################
#########################Figure1H:
##############################################################################################
##############################################################################################
##Average Number of OR per coronal section

#Read in the Data from FIJI processing
#Cells were circled in FIJI and the following measurements were collected for each OR positive cell:
# Area  Mean    Min      Max     X         Y         IntDen  RawIntDen
######

Files <- list.files(pattern = "*OLFR.csv")
CSVlist <- lapply(Files, read.csv)
names(CSVlist) <- Files
AllDat <- bind_rows(CSVlist, .id = "Origin")

#need to assign genotype
AllDat2 <- AllDat %>% separate(Origin,sep="_",into=c("File","Genotype"))
AllDat3 <- AllDat2 %>% count(Genotype,File)
BlueRedYellow_colorScheme2 <-c("#004488","#D55E00","#DDAA33")

Mean_OR_perSection<-ggplot((AllDat3) %>%
          mutate(Genotype = fct_relevel(Genotype, 'Control', 'TMexF2omp', 'TMexF2gg8omp')),
          mapping = aes(Genotype,n, color=Genotype, fill = Genotype)) + #ylim(-5,5)+
            geom_boxplot(alpha = 0.7, lwd =1.2 , outlier.shape=NA)+
            scale_color_manual(values=BlueRedYellow_colorScheme2) + scale_fill_manual(values =
          BlueRedYellow_colorScheme2) +
            geom_jitter(alpha = 1)+
            #geom_hline(yintercept=0, color="black") +
            theme_bw()+
            theme(legend.position="none",axis.text.y = element_text(colour= "black", size = 12),axis.text.x =
          element_text(colour= "black", size = 14), axis.title.x = element_blank(),axis.title.y =
          element_text(colour= "black", size = 14),plot.title = element_text(hjust = 0.5, size=20))+
            ggtitle("Number of ORs per coronal section")
Mean_OR_perSection

#subset each genotype separately
TMexF2gg8omp_count_per_slide <- AllDat3[AllDat3$Genotype=="TMexF2gg8omp",]
TMexF2gg8omp_count_per_slide <- TMexF2gg8omp_count_per_slide[ -c(2) ]
TMexF2gg8omp_count_per_slide <- TMexF2gg8omp_count_per_slide[ -c(1) ]
TMexF2omp_count_per_slide <- AllDat3[AllDat3$Genotype=="TMexF2omp",]
TMexF2omp_count_per_slide <- TMexF2omp_count_per_slide[ -c(2) ]
TMexF2omp_count_per_slide <- TMexF2omp_count_per_slide[ -c(1) ]
Control_count_per_slide <- AllDat3[AllDat3$Genotype=="Control",]
Control_count_per_slide <- Control_count_per_slide[ -c(2) ]
Control_count_per_slide <- Control_count_per_slide[ -c(1) ]

t.test(Control_count_per_slide,TMexF2omp_count_per_slide)

#Welch Two Sample t-test

#data:  Control_count_per_slide and TMexF2omp_count_per_slide
#t = 4.2705, df = 6.7003, p-value = 0.004088
#alternative hypothesis: true difference in means is not equal to 0
#95 percent confidence interval:
```

```
#  105.2019 371.6553
#sample estimates:
#  mean of x mean of y
#382.4286  144.0000

var.test(Control_count_per_slide,TMexF2omp_count_per_slide, alternative = 'two.sided')
t.test(Control_count_per_slide,TMexF2gg8omp_count_per_slide)

#Welch Two Sample t-test

#data:  Control_count_per_slide and TMexF2gg8omp_count_per_slide
#t = 7.1057, df = 6.1603, p-value = 0.0003469
#alternative hypothesis: true difference in means is not equal to 0
#95 percent confidence interval:
#  232.4850 474.3722
#sample estimates:
#  mean of x mean of y
#382.4286   29.0000


##############################################################################################
##############################################################################################
#########################Figure1I:
##############################################################################################
##############################################################################################
###Comparison between Olfr introns and exons in Mex3a Tg compared to GFP Tg (OMPtTA drivers)
library( "DESeq2" )
library(ggplot2)
library( "gplots" )
library( "RColorBrewer" )
###Deseq2 call significant OR when reads mapped to intron
countData <- read.table('intron_readcounts_TetOMex3a_Deseq2.txt', header = TRUE, sep =
"\t",row.names = 1)
metaData <- read.table('dataframe_tetO.txt', header = TRUE, sep = "\t",row.names = 1 )
dds <- DESeqDataSetFromMatrix(countData=countData, colData=metaData, design = ~ treatment )
#filter and check data replicates
dds <- dds [ rowSums(counts(dds))>2, ]
dds <- DESeq(dds)
rld <- rlog( dds )
plotPCA( rld, intgroup = c( "treatment"))
###DEG
res <- results( dds, independentFiltering=FALSE,contrast = c("treatment", "TetO_Mex3a", "TetO_GFP") )
###intron
df_intron <- as.data.frame(res)
df_intron <- cbind(Geneid = rownames(df_intron), df_intron)
rownames(df_intron) <- NULL
###add gene name based on gene id
id <- read.table("gene_id_name.txt",sep='\t',header=TRUE)
merge_intron <- merge(df_intron,id, on='Geneid')
#subset OR genes
OR_intron <- merge_intron[grepl("Olfr",merge_intron$Gene_name),]
colnames(OR_intron) <- paste0(colnames(OR_intron), "_intron")
names(OR_intron)[names(OR_intron) == "Gene_name_intron"] <- "Gene_name"
##Deseq2 call significant OR when reads mapped to exon
countData_exon <- read.table('exon_readcounts_TetOMex3a_Deseq2.txt', header = TRUE, sep =
"\t",row.names = 1)
```

```
dds <- DESeqDataSetFromMatrix(countData=countData_exon, colData=metaData, design = ~ treatment
)
#filter and check data replicates
dds <- dds [ rowSums(counts(dds))>2, ]
dds <- DESeq(dds)
rld <- rlog( dds )
vsd <- vst(dds, blind=FALSE)
###
plotPCA( rld, intgroup = c( "treatment"))
###DEG
res <- results(dds,independentFiltering=FALSE, contrast = c("treatment","TetO_Mex3a", "TetO_GFP") )
df_exon <- as.data.frame(res)
###rerun from here
df_exon <- read.table("TetOMex3a_vs_Tet0GFP_exon.txt")
head(df_exon)
df_exon <- cbind(Geneid = rownames(df_exon), df_exon)
rownames(df_exon) <- NULL

#change gene id into gene name based on id
merge <- merge(df_exon,id, on='Geneid')
###
OR_exon <- merge[grepl("Olfr",merge$Gene_name),]
OR_exon
colnames(OR_exon) <- paste0(colnames(OR_exon), "_exon")
names(OR_exon)[names(OR_exon) == "Gene_name_exon"] <- "Gene_name"

####merge intron and exon into one dataframe
merge_intron_exon <- merge(OR_intron,OR_exon, on='Gene_name')
df_new <- merge_intron_exon[, c('log2FoldChange_intron','log2FoldChange_exon','Gene_name')]
head(df_new)
library(data.table)
df_reshape <- melt(df_new, id.vars = c("Gene_name"),variable.name = "Type",value.name =
"log2FoldChange")
head(df_reshape)

p3 <- ggplot(df_reshape, aes(x=Type, y=log2FoldChange,fill=Type)) +
            geom_violin() + geom_boxplot(width=0.1,fill='white') +
            theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
                panel.background = element_blank(), axis.line = element_line(colour = "black",size=1),
                axis.text = element_text(colour = "black",size =10,face="bold"), legend.position = "none") +
            scale_fill_brewer(palette="Pastel1") +
            ylab("log2(Fold Change)") + xlab("") +
            scale_x_discrete(labels =c('log2FoldChange_intron' ="Intron", 'log2FoldChange_exon' =
          "Exon"))

pdf("exon vs intron.pdf")
p3
dev.off()
###significant test for log2 fold change in exon and intron group
intron_n <- subset(df_reshape, Type == 'log2FoldChange_intron')
exon_n <- subset(df_reshape, Type == 'log2FoldChange_exon')
wilcox.test(intron_n$log2FoldChange,exon_n$log2FoldChange, alternative = "two.sided")

###set significant threshold padj<0.05 and log2 fold change 0.5849
library('dplyr')
merge_intron_exon <- read.table("merge_intron_exon_update.txt", sep='\t')
```

```
merge_intron_exon <- merge_intron_exon %>% mutate(intron_change =
          case_when((log2FoldChange_intron > 0.5849 & padj_intron < 0.5) ~ "Significant",
           (log2FoldChange_intron < -0.5849 & padj_intron < 0.5) ~ "Significant",
           (log2FoldChange_intron <= 0.5849 & log2FoldChange_intron >= -0.5849 | padj_intron >= 0.5) ~
          "Not Significant"))

merge_intron_exon <- merge_intron_exon %>% mutate(exon_change =
          case_when((log2FoldChange_exon > 0.5849 & padj_exon < 0.5) ~ "Significant",
          (log2FoldChange_exon < -0.5849 & padj_exon < 0.5) ~ "Significant",
          (log2FoldChange_exon <= 0.5849 & log2FoldChange_exon >= -0.5849 | padj_exon >= 0.5) ~
          "Not Significant"))

merge_intron_exon
dt <- read.table('merge_intron_exon_update.txt',sep='\t')
###set significant threshold padj is 0.05 and log2 fold change is 0.5849
library('dplyr')
dt  <- dt %>% mutate(intron_change =
                  case_when((log2FoldChange_intron > 0.5849 & padj_intron < 0.5) ~ "Significant Up",
                    (log2FoldChange_intron < -0.5849 & padj_intron < 0.5) ~ "Significant Down",
                    (log2FoldChange_intron <= 0.5849 & log2FoldChange_intron >= -0.5849 | padj_intron >=
                    0.5) ~ "Not Significant"))

dt <- dt %>% mutate(exon_change =
                 case_when((log2FoldChange_exon > 0.5849 & padj_exon < 0.5) ~ "Significant Up",
                    (log2FoldChange_exon < -0.5849 & padj_exon < 0.5) ~ "Significant Down",
                    (log2FoldChange_exon <= 0.5849 & log2FoldChange_exon >= -0.5849 | padj_exon >=
                    0.5) ~ "Not Significant"))

dt
dt %>% group_by(intron_change) %>% tally()
dt %>% group_by(exon_change) %>% tally()

dt2 <- read.table('count.txt',sep='\t',header = TRUE)
dt2

p8 <- ggplot(dt2, aes(x = Group, y=Count, fill = Change)) +
           geom_col(color="white")  +
           scale_fill_brewer(palette="Set2") +
           ylab("Number of OR") + xlab("") +
           theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
                panel.background = element_blank(), axis.line = element_line(colour = "black",size=1),
                axis.text = element_text(colour = "black",size =10,face="bold"),legend.title =
          element_blank())
p8
dev.off()

##############################################################################################
##############################################################################################
########################Figure2B:
##############################################################################################
##############################################################################################

#data were generated by MaxQuant analysis of multiple Mass Spec runs (two biological replicates per genotype plus four biological replicates of Mex3a cKO IP), using mouse proteome as reference proteome
#Column headers were changed to reflect the experimental name before inputting into R for further normalization and plotting
```

```
### Peptides enriched in Mex3a or V5 IP compared to IgG
#Set up the environment
setwd('/path/to/working/directory')
library(tidyverse)
library(reshape2)
library(GGally)

#Read in the data and sample names
#proteinGroups file is output from MaxQuant
raw_data <- read.delim('proteinGroups_201906Batch.txt', sep = '\t', stringsAsFactors = F)
sample_names <- read_tsv('MexIP_SampleNames_Jan2020.txt', col_names = T) %>%
  mutate_all(.funs = str_replace_all,
             pattern = ' |-|\\.', replace = '_')

#Filter the data
filtered_data <- raw_data %>%
  rename_all(str_replace_all, pattern = ' |-|\\.', replace = '_') %>%
  filter(Reverse != '+' & Potential_contaminant != '+') %>% #remove confounders
  mutate('unique_id' = seq(1, nrow(.))) #assign unique id

#Create Reporter Intensity Table
intensity_table_raw_norm <- filtered_data %>%
  #Collapse vars of interest (ids, corrected intensities) to long form
  melt(id.vars = c('unique_id','Protein_IDs','Majority_protein_IDs', 'Gene_names'),
       measure.vars = names(.) %>% .[str_detect(., '(^Reporter_intensity_corrected_[0-9]+_Mix[0-
9]+$)|(^iBAQ_Mix[0-9]+$)')]) %>%
  #Tidy vars
  tidyr::extract(col = 'variable', into = c('variable','mix'), regex = '(.*)_(Mix[0-9]+)') %>%
  #Project id + mix  ~ corrected intensities + iBAQ
  dcast(...~variable, value.var = 'value') %>%
  #Normalize using Lena's formula
  mutate('sum_intensities' = rowSums(dplyr::select(., contains('Reporter_intensity_corrected')))) %>%
  mutate_at(.vars = vars(contains('Reporter_intensity_corrected')),
            .funs = function(x) {x*.$iBAQ/.$sum_intensities}) %>%
  #Collapse vars of interest (ids, mix, norm intensities) to long form
  melt(id.vars = c('unique_id','Protein_IDs','Majority_protein_IDs', 'Gene_names', 'mix'),
       measure.vars = names(.) %>% .[str_detect(.,'Reporter_intensity_corrected')]) %>%
  #Map sample codes back to sample names
  mutate('MQ_name' = paste(variable, mix, sep = '_')) %>%
  left_join(sample_names, by = 'MQ_name') %>%
  filter(!is.na(Sample_name)) %>% #remove empty lanes
  #Tidy that data
  tidyr::extract(col = 'Sample_name', into = c('genotype', 'sample_type','replicate','mix'),
                 regex = '(.*)_.*(IP|IgG)_rep([0-9]+)_Mix([0-9]+)',
                 remove = F) %>%
  #Normalize by TPM and log2 transform, add pseudocount
  group_by(Sample_name) %>%
  mutate('norm_value' = log2((value/sum(value, na.rm = T) * 10^6)+1)) %>%
  ungroup()

#Compute Per Library Enrichment (IP-IgG)
enrichment_over_igg <- intensity_table_raw_norm %>%
  dplyr::select(unique_id, Protein_IDs, Majority_protein_IDs, Gene_names,
                genotype, sample_type, replicate, norm_value) %>%
  dcast(...~ sample_type, value.var = 'norm_value') %>%
```

```
  mutate('enrichment' = IP-IgG)

#Plot Enrichment vs. IP Value, Facetted by replicates (view in big window for best experience :-P)
#Note, KO is Mex3a cKO, TMexF1 was not used in downstream analysis due to lower Mex3a enrichment
in IP, TMexF2 is Mex3a Tg, TMexF6 is Mex3a(mut) Tg, WT is 1mg of protein for IP (did not work
efficiently, Mex3a is not easy to IP), WT_4mg is 4mg of protein for IP (termed "WT" in downstream
analysis)
ggplot(data =  enrichment_over_igg %>%
                 mutate('experiment' = interaction(genotype, replicate),
                        'identity'=ifelse(enrichment > 0, 'Enriched','Depleted')),
               aes(x = IP, y = IP-IgG, color = identity)) +
            geom_point(alpha = 0.1) +
            facet_wrap(facets = vars(genotype)) +
            theme_bw() +
            geom_hline(yintercept = 0, linetype = 'dashed') +
            geom_point(data = . %>% filter(str_detect(Gene_names,'Mex3a')),
                       color = 'blue', shape = 18, size =2.5) +
            ylim(-5,5)

##############################################################################################
##############################################################################################
#########################Figure2F:
##############################################################################################
##############################################################################################
### code below continues from above and generates a dataframe that was then imported into Cytoscape
to generate the STRING style figure

enrichment_over_igg %>%
            filter(str_detect(genotype, 'WT_4mg|TMexF2|TMexF6')) %>%
            group_by(genotype, replicate) %>%
            summarize()

p_val_table <- enrichment_over_igg %>%
            filter(str_detect(genotype, 'WT_4mg|TMexF2|TMexF6')) %>%
            melt(id.vars = c('unique_id','Gene_names', 'genotype', 'replicate'),
                 measure.vars = c('IgG','IP')) %>%
            group_by(unique_id, Gene_names) %>%
            filter(length(which(!is.na(value[variable == 'IgG']))) >= 2 &
                     length(which(!is.na(value[variable == 'IP']))) >= 2) %>%
            summarize('mean_IP' = mean(value[variable == 'IP'], na.rm = T),
                      'mean_enrichment' = mean_IP - mean(value[variable == 'IgG'], na.rm = T),
                      'p_val_ttest' = t.test(x = value[variable == 'IP'],
                                             y = value[variable == 'IgG'])$p.value,
                      'p_val_wilcox' = wilcox.test(x = value[variable == 'IP'],
                                                   y = value[variable == 'IgG'])$p.value,
                      'p_val_KS' = ks.test(x = value[variable == 'IP'],
                                           y = value[variable == 'IgG'])$p.value) %>%
            ungroup() %>%
            mutate('padj_ttest' = p.adjust(p_val_ttest, method = 'fdr'),
                   'padj_wilcox' = p.adjust(p_val_wilcox, method = 'fdr'),
                   'padj_KS' = p.adjust(p_val_KS, method = 'fdr'))

#WT is all working IPs, the WT_4mg, Mex Tg, and Mex (mut) Tg, total of six replicates
enrichment_wt_vs_KO <-
            enrichment_over_igg %>%
            filter(str_detect(genotype, 'WT_4mg|TMexF2|TMexF6|KO')) %>%
```

```
 mutate('experimental_group' = ifelse(genotype == 'KO','KO','WT')) %>%
 group_by(unique_id, Gene_names) %>%
 filter(length(which(!is.na(enrichment[experimental_group == 'KO']))) >= 2 &
length(which(!is.na(enrichment[experimental_group == 'WT']))) >= 2) %>%
summarize('mean_enrichment_WT' = mean(enrichment[experimental_group == 'WT'], na.rm =
T),
        'mean_enrichment_KO' = mean(enrichment[experimental_group == 'KO'], na.rm = T),
        'p_val_wilcox' = wilcox.test(x = enrichment[experimental_group == 'WT'],
                                     y = enrichment[experimental_group == 'KO'])$p.value) %>%
 ungroup() %>%
 mutate('padj_wilcox' = p.adjust(p_val_wilcox, method = 'fdr'))

enrichment_over_igg %>% write_excel_csv('/path/IP_MS_workingIPs_enrichment_over_igg_Pval.csv')
enrichment_wt_vs_KO %>% write_excel_csv('/path/IP_MS_enrichment_workingIPs_vs_KO_Pval.csv')
#Candidates were 1) enriched relative to IgG by Log2(0.5) or greater. 2) enriched relative to Mex3a cKO
Mex3a IP by Log2(0.5) or greater. 3) Had a wilcox p value of 0.05 or lower


##############################################################################################
##############################################################################################
##########################Figure3E:
##############################################################################################
##############################################################################################

##Values were calculated in NIS software. The mean fluorescence intensity for a given color channel was
calculated for each image. Each dot on the graph represents values for an entire image

library(tidyverse)
library(magrittr)
library(reshape2)
library(ggExtra)
library(gridExtra)
library(dplyr)

setwd('/path/to/working/directory/tetOMex_Analysis')

tetOMex_Mean_pearson_DF <- read.csv('TetOMex_MEAN_PEARSON_DF_ordered.csv',
stringsAsFactors = F)
tetOMex_Mean_pearson_DF_summary <- tetOMex_Mean_pearson_DF %>% count(Genotype)

Okabi_Ito_colors_diagram<-
c("#CC79A7","#D55E00","#0072B2","#F0E442","#009E73","#56B4E9","#E69F00")

mean_plot <-ggplot((tetOMex_Mean_pearson_DF) %>%
            filter(Genotype %in% c('TM71 omp','TmexF2 TM71 omp','TmexF6 TM71 omp','TP2
            omp','TmexF2 TP2 omp','TP2 gg8omp','TmexF2 TP2 gg8omp')) %>%
            mutate('Genotype' = fct_relevel(Genotype,'TM71 omp','TmexF2 TM71 omp','TmexF6 TM71
            omp','TP2 omp','TmexF2 TP2 omp','TP2 gg8omp','TmexF2 TP2 gg8omp')),
            mapping = aes(Genotype,Green_Mean_intensity_value, color=Genotype, fill = Genotype)) +
                geom_boxplot(alpha = 0.7, lwd =1.2 , outlier.shape=NA)+
                scale_color_manual(values=Okabi_Ito_colors_diagram) + scale_fill_manual(values =
                Okabi_Ito_colors_diagram) +
            geom_jitter(alpha = 1)+
            theme_bw() +
```

```
        theme(legend.position="none",axis.text.y = element_text(colour= "black"),axis.text.x =
      element_text(colour= "black", size = 8), axis.title.x = element_blank(),axis.title.y =
      element_text(colour= "black", size = 8),plot.title = element_text(hjust = 0.5, size=12))+
      ggtitle("Mean OR fluorescence intensity")
mean_plot

#Calculate Significance
TM71omp_mean_values <-tetOMex_Mean_pearson_DF[tetOMex_Mean_pearson_DF$Genotype
=="TM71 omp",]
TM71omp_mean_values <- TM71omp_mean_values[-c(1:7)]

TM71omp_TMexF2_mean_values <-
tetOMex_Mean_pearson_DF[tetOMex_Mean_pearson_DF$Genotype =="TmexF2 TM71 omp",]
TM71omp_TMexF2_mean_values <- TM71omp_TMexF2_mean_values[-c(1:7)]

t.test(TM71omp_mean_values,TM71omp_TMexF2_mean_values)

#Welch Two Sample t-test

#data:  TM71omp_mean_values and TM71omp_TMexF2_mean_values
#t = 6.1816, df = 16.043, p-value = 1.301e-05
#alternative hypothesis: true difference in means is not equal to 0
#95 percent confidence interval:
#  17.18827 35.12418
#sample estimates:
#  mean of x mean of y
#39.06637  12.91014

TM71omp_TMexF6_mean_values <-
tetOMex_Mean_pearson_DF[tetOMex_Mean_pearson_DF$Genotype =="TmexF6 TM71 omp",]
TM71omp_TMexF6_mean_values <- TM71omp_TMexF6_mean_values[-c(1:7)]

t.test(TM71omp_TMexF2_mean_values,TM71omp_TMexF6_mean_values)
#Welch Two Sample t-test

#data:  TM71omp_TMexF2_mean_values and TM71omp_TMexF6_mean_values
#t = -6.5959, df = 17.158, p-value = 4.332e-06
#alternative hypothesis: true difference in means is not equal to 0
#95 percent confidence interval:
#  -32.22984 -16.61655
#sample estimates:
#  mean of x mean of y
#12.91014  37.33334

t.test(TM71omp_mean_values,TM71omp_TMexF6_mean_values)

TMP2omp_mean_values <-tetOMex_Mean_pearson_DF[tetOMex_Mean_pearson_DF$Genotype
=="TP2 omp",]
TMP2omp_mean_values <- TMP2omp_mean_values[-c(1:7)]

TP2_TMexF2_omp_mean_values <-
tetOMex_Mean_pearson_DF[tetOMex_Mean_pearson_DF$Genotype =="TmexF2 TP2 omp",]
TP2_TMexF2_omp_mean_values <- TP2_TMexF2_omp_mean_values [-c(1:7)]

t.test(TMP2omp_mean_values,TP2_TMexF2_omp_mean_values)
```

```
#Welch Two Sample t-test

#data:  TMP2omp_mean_values and TP2_TMexF2_omp_mean_values
#t = 4.2648, df = 24.824, p-value = 0.0002535
#alternative hypothesis: true difference in means is not equal to 0
#95 percent confidence interval:
#  2.014260 5.779146
#sample estimates:
#  mean of x mean of y
#6.484718  2.588015

TMP2gg8omp_mean_values <-tetOMex_Mean_pearson_DF[tetOMex_Mean_pearson_DF$Genotype
=="TP2 gg8omp",]
TMP2gg8omp_mean_values <- TMP2gg8omp_mean_values[-c(1:7)]

TP2_TMexF2_gg8omp_mean_values <-
tetOMex_Mean_pearson_DF[tetOMex_Mean_pearson_DF$Genotype =="TmexF2 TP2 gg8omp",]
TP2_TMexF2_gg8omp_mean_values <- TP2_TMexF2_gg8omp_mean_values[-c(1:7)]

t.test(TMP2gg8omp_mean_values,TP2_TMexF2_gg8omp_mean_values)

#Welch Two Sample t-test

#data:  TMP2gg8omp_mean_values and TP2_TMexF2_gg8omp_mean_values
#t = 4.5099, df = 3.4399, p-value = 0.01512
#alternative hypothesis: true difference in means is not equal to 0
#95 percent confidence interval:
#  7.412282 35.839318
#sample estimates:
#  mean of x mean of y
#24.96325   3.33745


##############################################################################################
##############################################################################################
#########################Figure3G:
##############################################################################################
##############################################################################################

##############Colocalization
Pearson_colors_diagram<-c("#E69F00","#3621DA")


coloc_plot_main <- ggplot((tetOMex_Mean_pearson_DF)%>%
                                    filter(Genotype %in% c('TmexF2 TP2 gg8omp','TmexF2 TGFP
                  gg8omp')) %>%
                                     mutate('Genotype' = fct_relevel(Genotype,'TmexF2 TP2 gg8omp','TmexF2
                  TGFP gg8omp')),
                                  mapping = aes(Genotype,Pearson_Coefficient, color=Genotype, fill =
                  Genotype)) + #ylim(-5,5)+
                    geom_boxplot(alpha = 0.7, lwd =1.2 , outlier.shape=NA)+
                    scale_color_manual(values=Pearson_colors_diagram) + scale_fill_manual(values =
                  Pearson_colors_diagram) +
                    geom_jitter(alpha = 1)+
                    theme_bw() +
```

```
                  theme(legend.position="none",axis.text.y = element_text(colour= "black", size =
                12),axis.text.x = element_text(colour= "black", size = 14), axis.title.x =
                element_blank(),axis.title.y = element_text(colour= "black", size = 14),plot.title =
                element_text(hjust = 0.5, size=20))+
                  ggtitle("Pearson Colocalization Coefficient")
coloc_plot_main


TP2_TMex_gg8omp_pearson_values <-
tetOMex_Mean_pearson_DF[tetOMex_Mean_pearson_DF$Genotype =="TmexF2 TP2 gg8omp",]
TP2_TMex_gg8omp_pearson_values <- TP2_TMex_gg8omp_pearson_values[-c(1:5,7,8)]

TGFP_TMexF2_gg8omp_pearson_values <-
tetOMex_Mean_pearson_DF[tetOMex_Mean_pearson_DF$Genotype =="TmexF2 TGFP gg8omp",]
TGFP_TMexF2_gg8omp_pearson_values <- TGFP_TMexF2_gg8omp_pearson_values[-c(1:5,7,8)]

t.test(TP2_TMex_gg8omp_pearson_values,TGFP_TMexF2_gg8omp_pearson_values)

#Welch Two Sample t-test

#data:  TP2_TMex_gg8omp_pearson_values and TGFP_TMexF2_gg8omp_pearson_values
#t = -5.7933, df = 13.015, p-value = 6.216e-05
#alternative hypothesis: true difference in means is not equal to 0
#95 percent confidence interval:
#  -0.4739809 -0.2165188
#sample estimates:
#  mean of x mean of y
#0.2082872 0.5535370


##############################################################################################
##############################################################################################
##########################Figure3J:
##############################################################################################
##############################################################################################

#Same code as Fig 1E and Fig 1F, this time comparing Mex3a (mut) Tg to Mex3a Tg (OMPtTA drivers)
### get results for specific contrasts with p value set to 0.05
res <- results(dds, alpha=0.05,lfcThreshold = 0.5849,contrast= c("condition", "TetOmutMex_omp",
"TetOMex_omp"))
cutoff <- 0.05
df.res <- data.frame(res)
df.res <- rownames_to_column(df.res, var = 'gene_name')
#Olfr subset for MA plot
Olfr <- filter_genes(scan(file = "/path/to/list/of/Olfrs.txt", what = character()))
df.res.Olfr <- res[Olfr,]
df.res.Olfr <- data.frame(res[Olfr,])
df.res.Olfr <- rownames_to_column(df.res.Olfr, var = 'gene_name')
df.res.Olfr$significant <- ifelse((df.res.Olfr$padj< cutoff),TRUE,FALSE)
MA.res.Olfr <- data.frame(mean = df.res.Olfr$baseMean, Fold = df.res.Olfr$log2FoldChange,
sig=df.res.Olfr$significant)
ggplot(data = MA.res.Olfr, aes(x = mean, y = Fold, color = sig)) +
                  geom_point()+
                  theme_bw() +
                  scale_color_manual(values = c('FALSE' = 'gray','TRUE' = 'firebrick1')) +
                  scale_alpha_manual(values = c('FALSE' = 1, 'TRUE' = 1))+
```

```
                    scale_x_log10(name = 'mean of normalized counts', limits = c(10, NA)) +
                    geom_hline(yintercept = 0, linetype = 'dashed', color = 'black')
#Violin plot
df.res$Olfr <- ifelse(df.res$gene_name %in% Olfr, 'Olfactory_Receptor','Other')
ggplot(data=df.res,
           aes(x = Olfr, y = log2FoldChange, fill = Olfr)) +
          geom_violin() +
          scale_fill_manual(values=c("#FF0000", "#656565")) +
          geom_boxplot(fill = 'white', width = 0.1,outlier.shape = NA) +
          theme_classic() +
          geom_hline(yintercept = 0, linetype='dashed') +
          xlab('') + ylab('Log2FC tetO-Mex3a Mutant RING vs tetO-Mex3a WT')
var.test(log2FoldChange ~ Olfr, df.res)
#for the Mex3a mutant RING Tg the variance was not significant, so I set var.equal to TRUE
t.test(log2FoldChange ~ Olfr, df.res, var.equal=TRUE)

##############################################################################################
##############################################################################################
#########################Figure4A:
##############################################################################################
##############################################################################################
#Align the single cell reads with 10X Genomics CellRanger
#gtf is the same as described for Figure 1B
nohup /path/to/cellranger-5.0.1/cellranger count --id=SAMPLE_NAME --transcriptome=/path/to/custom/gtf
--fastqs=/path/to/fastqs --sample=SAMPLE_name --localcores=16 --include-introns

####Seurat was used for single cell RNA Seq analysis. This resource has been described in multiple
publications found here: https://cran.r-project.org/web/packages/Seurat/citation.html
#especially Satija et al. 2015 https://doi.org/10.1038/nbt.3192.

#Analysis in R with Seurat
library(Seurat)
library(tidyverse)
library(reshape2)
library(sctransform)
library(dplyr)
library(umap)
library(ggplot2)
library(cowplot)

setwd('/path/to/working/directory')
#Return a list of transformed datasets
files_ls <- c(
  '/path/to/cellranger/outs/KO1/outs/filtered_feature_bc_matrix',
  '/path/to/cellranger/outs/KO2/outs/filtered_feature_bc_matrix',
  '/path/to/cellranger/outs/WT1/outs/filtered_feature_bc_matrix',
  '/path/to/cellranger/outs/WT2/outs/filtered_feature_bc_matrix') %>%
  map(.f = function(filepath_) {
    extracted_name <- str_extract(filepath_, '(WT|KO)[12]')
    Read10X(data.dir = filepath_) %>%
      CreateSeuratObject(project = extracted_name, min.cells = 3, min.features = 200) %>%
      SCTransform ()
  })
#Do the integration
features <- SelectIntegrationFeatures(object.list = files_ls, nfeatures = 3000)
files_ls <- PrepSCTIntegration(object.list = files_ls, anchor.features = features)
```

```
anchors_ <- FindIntegrationAnchors(object.list = files_ls, normalization.method = "SCT",
                                   anchor.features = features)
Mex.integrated.sct <- IntegrateData(anchorset = anchors_, normalization.method = "SCT")
Mex.integrated.sct <- RunPCA(Mex.integrated.sct, verbose = FALSE)
Mex.integrated.sct <- RunUMAP(Mex.integrated.sct, reduction = "pca", dims = 1:15)
#normalize RNA
DefaultAssay(Mex.integrated.sct) <- "RNA"
Mex.integrated.sct <- NormalizeData(Mex.integrated.sct, verbose = FALSE)
#Adding Seurat_Clusters
DefaultAssay(Mex.integrated.sct) <- "integrated"
Mex.integrated.sct <- FindNeighbors(Mex.integrated.sct, dims = 1:15, verbose = FALSE)
Mex.integrated.sct <- FindClusters(Mex.integrated.sct, verbose = TRUE)
DimPlot(Mex.integrated.sct, label = TRUE) + NoLegend()
ClusterMarkers_KOWT.sct.integrated.15dimensions <- FindAllMarkers(object = Mex.integrated.sct)
write_csv(ClusterMarkers_KOWT.sct.integrated.15dimensions,
"ClusterMarkers_KOWT.sct.integrated.15dimensions.12may22.csv")
#Manually went through each cluster, searched highest genes and assigned an identity to clusters based
on literature. Some clusters are combined into one identity!
#to reset to original seurat clusters
Idents(Mex.integrated.sct) <- 'seurat_clusters'
levels(Mex.integrated.sct)
#cluster 0 = "Mature OSN", cluster 1 = "Sustentacular" etc...
new.cluster.ids <- c("Mature OSN","Sustentacular","Immature OSN","Mature OSN","Mature OSN",
                     "Immature OSN","Bone/Cartilage","Sustentacular","Immediate Neuronal
Progenitor","Bone/Cartilage",
                     "Immediate Neuronal Progenitor","Blood/Immune","Immediate Neuronal
Progenitor","Globose Basal Cell","Microvillus",
                     "Blood/Immune","Horizontal Basal Cell","Immature OSN","Blood/Immune","Olfactory
Ensheathing","Horizontal Basal Cell",
                     "Blood/Immune","Blood/Immune","Sustentacular","Blood/Immune","Microvillus",
                     "Bowman's Gland","Respiratory","Immediate Neuronal
Progenitor","Respiratory","Blood/Immune","Mature
OSN","Blood/Immune","Respiratory","Sustentacular","Bowman's Gland")
names(new.cluster.ids) <- levels(Mex.integrated.sct)
Mex.integrated.sct <- RenameIdents(Mex.integrated.sct, new.cluster.ids)
Mex.integrated.sct[["new.cluster.ids"]] <- Idents(object = Mex.integrated.sct)
Tol_muted_plus_three2 <- c("#332288","#DDCC77","#CC6677", "#999933", "#882255",
"#88CCEE","#EE3377", "#44AA99","#117733", "#DDAA33","#5289C7", "#AA4499")
DimPlot(Mex.integrated.sct,
      cols = Tol_muted_plus_three2, label=TRUE , repel=TRUE)

##############################################################################################
##############################################################################################
#########################Figure4B:
##############################################################################################
##############################################################################################

#builds of off code from Fig 4A
all_OR_data_INP <- Mex.integrated.sct@meta.data %>%
            rownames_to_column(var = 'cell_id') %>%
            filter(new.cluster.ids %in% c("Immediate Neuronal Progenitor")) %>%
            pull(cell_id) %>%
            {Mex.integrated.sct[['RNA']]@data[str_detect(rownames(Mex.integrated.sct[['RNA']]@data),
          'Olfr'), .]} %>%
            as.data.frame() %>%
            rownames_to_column(var = 'gene_name') %>%
```

```
  melt(id.vars = 'gene_name', variable.name = 'cell_id') %>%
  left_join(rownames_to_column(Mex.integrated.sct@meta.data, var = 'cell_id'), by = 'cell_id')
#add the metadata

all_OR_data_INP$geno <- factor(x=all_OR_data_INP$geno, levels = c("WT","KO"))

ggplot(data = all_OR_data_INP %>%
       group_by(cell_id, new.cluster.ids, geno) %>%
                summarize('OR_scaled_expression' = sum(value),
                          'Number_ORs_detected' = length(which(value > 1))) %>%
                ungroup() %>%
                mutate('genotype' = fct_relevel(geno,'WT','KO')) %>%
                filter(Number_ORs_detected > 1) %>%
                group_by(new.cluster.ids, geno, Number_ORs_detected) %>%
                summarize(n=n()) %>%
                mutate('Fraction_of_Total_Cells_With_Coexpression' = n/sum(n)),
              aes(x = Number_ORs_detected, y = Fraction_of_Total_Cells_With_Coexpression, fill =
geno)) +
  geom_col(position = 'dodge', color = 'black') +
  theme_bw() +
  facet_wrap(facets = vars(new.cluster.ids), nrow = 1)

##############################################################################################
##############################################################################################
#########################Figure4C:
##############################################################################################
##############################################################################################

all_OR_data_neuronal_lineage <- Mex.integrated.sct@meta.data %>%
  rownames_to_column(var = 'cell_id') %>%
  filter(new.cluster.ids %in% c('Mature OSN','Immature OSN','Immediate Neuronal
Progenitor','Globose Basal Cell')) %>%
  pull(cell_id) %>%
  {Mex.integrated.sct[['RNA']]@data[str_detect(rownames(Mex.integrated.sct[['RNA']]@data),
'Olfr'), .]} %>%
  as.data.frame() %>%
  rownames_to_column(var = 'gene_name') %>%
  melt(id.vars = 'gene_name', variable.name = 'cell_id') %>%
  left_join(rownames_to_column(Mex.integrated.sct@meta.data, var = 'cell_id'), by = 'cell_id')

all_OR_data_neuronal_lineage$geno <- factor(x=all_OR_data_neuronal_lineage$geno, levels =
c("WT","KO"))
all_OR_data_neuronal_lineage$new.cluster.ids <-
factor(x=all_OR_data_neuronal_lineage$new.cluster.ids, levels = c('Globose Basal Cell','Immediate
Neuronal Progenitor','Immature OSN','Mature OSN'))

data = all_OR_data_neuronal_lineage %>% #I made a df that looks like the all_OR_data_INP above but
has the four cell types below
  filter(new.cluster.ids %in% c("Globose Basal Cell","Immediate Neuronal Progenitor","Immature
OSN","Mature OSN")) %>%
  group_by(cell_id, new.cluster.ids, geno) %>%
  summarize('OR_scaled_expression' = sum(value),
            'Number_ORs_detected' = length(which(value > 1))) %>%
  ungroup() %>%
  mutate('OR_expression' = ifelse(Number_ORs_detected == 0, 'Zero',
ifelse(Number_ORs_detected == 1, 'One','Multiple'))) %>%
```

```
        group_by(new.cluster.ids, geno, OR_expression)

#Violin Plot
ggplot(data = data %>%
                filter(OR_expression %in% c("One","Multiple")),
            aes(x = geno, y = OR_scaled_expression)) +
        geom_violin(aes(fill = geno), color = 'black') +
        scale_fill_manual(values=c("#12908E", "#F98F45")) +
        geom_boxplot(fill = 'white', width = 0.2, outlier.shape = NA) +
        facet_wrap(facets = vars(new.cluster.ids), nrow = 1) +
        stat_summary(fun.y=mean, geom="point", shape=20, size=2, color="blue", fill="blue") +
        geom_text(data = . %>% group_by(new.cluster.ids,geno) %>%
                    summarize(n=n()),
                  aes(y = 22, label = paste0('n=',n, ' Cells'))) +
        theme_bw() +
        ggtitle('Cells with One or Multiple ORs per cell')

#statistics
test_vals <- filter(data, OR_expression != "Zero" & geno == 'KO' & new.cluster.ids == 'Globose Basal
Cell')$OR_scaled_expression
control_vals <- filter(data, OR_expression != "Zero" & geno == 'WT' & new.cluster.ids == 'Globose Basal
Cell')$OR_scaled_expression
wilcox.test(control_vals, test_vals)

#Same code repeated replacing new.cluster.ids == 'Immediate Neuronal Progenitor', new.cluster.ids ==
'Immature OSN',new.cluster.ids == 'Mature OSN'

################################################################################################
################################################################################################
#########################Figure4E:
################################################################################################
################################################################################################

#Custom code from Hani Shayya, more information available upon request.
#Ribosome Profiling of NgnGFP sorted cells, two replicates Mex3a cKO and WT at PN12
#Example shell script demonstrating critical steps of pre-processing/alignment for ribosome profiling
experiments
#This script is NOT meant to be run as a stand-alone script. It is simply to show the commands that were
used.
#Cutadapt, Bowtie, STAR, Samtools, wigtoBigWig are standard bioinformatics tools
#Psite, make_wiggle, metagene, phase_by_size are tools from the plastid software package (Dunn, JG
and Weissman JS, BMC Genomics, 2016)
#Plastid software is available at https://plastid.readthedocs.io
#featurecounts_1_4.py is a custom read counting software provided at the end of this section

################################################################################################
######                    Pre-processing, alignment and post-processing of ribosome profiling libraries
################################################################################################
#RiboSeq_Analysis.sh
#Ribo-seq data in its own folder
cd path/to/data/4_16_Ribo_MexWT_KO_Ngn

#Find lane-split fastq files in file structure output from bcl2fastq and move to top level
array=()
while IFS=  read -r -d $'\0'; do
array+=("$REPLY")
```

```
done < <(find . -type f -regex ".*fastq.gz" -print0)

for i in "${array[@]}"
do
mv $i ./
  done

#Clean the empty file tree
rm -r 4_16_Ribo_MexWT_KO_Ngn-128425299
rm -r Unindexed_Reads-94202108

#Concatenate lane-split samples together with a useful name. Only a subset of ribo-seq libraries shown
here.
samples=("Ribo-MexKO-Ngn-1" "Ribo-MexKO-Ngn-2" "Ribo-MexWT-Ngn-1" "Ribo-MexWT-Ngn-2"
"Undetermined")
for i in ${samples[@]}
do
echo "Working on Sample: "$i
echo "Found the following files for concatenation: "
find -iname "*$i*"
find -iname "*$i*" -print0 | xargs -0 cat >> "$i".fastq.gz
echo "Concatenated files to: $i.fastq.gz"
done
#note that $i.fastq.gz created first in the pipe so cat tells you it ignores it
#this is fine, see https://superuser.com/questions/703887/cat-input-file-is-output-file-while-trying-to-
combine-multiple-files-into-one

#Clean up non-concatenated files
rm *R1_001*

  #Basic Pipeline for ribo-seq read pre-processing and alignment.
  #Note bowtie index was previously prepared from fasta file of 45S rRNA sequence, provided.
  #Note STAR index was previously prepared from provided GTF, see methods in paper.
  function riboprof_basic_align {
    echo Received Input Fastq: "$1".fastq.gz
    echo Using $2 processors
    echo Pre-processing input...
    zcat "$1".fastq.gz | cutadapt --cut 3 -a "A{15}" -j 20 --nextseq-trim=20 --minimum-length 17 -
2>cutadaptlog_$1.log | bowtie -p $2 -v 0 --un "$1"_unalig.fq
/path/to/alignment/45SrRNA_bowtie_index/45SrRNA - >/dev/null 2>bowtielog_$1.log
    echo "done!"

    echo Compressing non-rRNA reads using $2 cores...
    pigz "$1"_unalig.fq -p $2

    echo Creating directory ./STARout_$1 and aligning to mm10 using $2 cores...
    mkdir ./STARout_$1
    STAR --runMode alignReads --runThreadN $2 --genomeDir /path/to/alignment/STARDir_Coding30nt --
outFileNamePrefix ./STARout_$1/"$1"_ --outSAMtype BAM SortedByCoordinate --alignSJoverhangMin
400 --outFilterMismatchNmax 0 --outFilterMatchNmin 15 --outFilterMultimapNmax 1 --readFilesCommand
zcat --readFilesIn "$1"_unalig.fq.gz
    echo "done!"
  }

#Run the alignment
for i in ${samples[@]}
```

```
do
riboprof_basic_align $i 20
done

#clean up logs
mkdir ./logs
mv *log ./logs

#Clean up raw fastq
mkdir ./zipped_fq
mv *gz ./zipped_fq/

  #Find all the aligned bam files for post-processing
  for_postproc=()
while IFS=  read -r -d $'\0'; do
for_postproc+=("$REPLY")
done < <(find . -type f -iname "*Aligned.sortedByCoord.out.bam" ! -path "./STARout_wholeOE-
pool1_initial/*" -print0)

#Pipeline for initial steps of post-processing.
#Provided the roi file for psite, make_wiggle
for i in ${for_postproc[@]}
do
echo "Found File: $i"
echo "Indexing..."
samtools index $i
echo "done!"
my_path=$(echo $i | grep -Po ".*(?=_Aligned.sortedByCoord.out.bam)")
echo "Running psite script. Outfile Name Set as $my_path"
psite --count_files $i --countfile_format BAM --min_counts 50 --require_upstream --min_length 17 --
max_length 35 /path/to/alignment/Plastid_Metagene/mm10_CDSStart_50bp_rois.txt $my_path
echo "done!"
read -p "Please edit the P Offset file located in folder $my_path, then press enter to continue"
echo Making Wiggle Files. Outfile Name Set as $my_path. Using offset file $my_path"_p_offsets.txt"
make_wiggle --count_files $i --countfile_format BAM --min_length 17 --fiveprime_variable --offset
$my_path"_p_offsets.txt" -o $my_path
echo "done!"
done

#Convert wig to bigwig for easy vizualization in genome browser
for_bw=()
while IFS=  read -r -d $'\0'; do
for_bw+=("$REPLY")
done < <(find . -type f -iname "*wig" ! -path "./STARout_wholeOE-pool1_initial/*" -print0)

for i in ${for_bw[@]}
do
wigToBigWig "$i" /path/to/alignment/mm10annotation_ensembl/mm10.chrom.sizes "$i.bw"
done

#Clean up large wig files.
#for i in ${for_bw[@]}; do rm $i; done

#Run the final QC steps of the plastid package
for_QC=()
while IFS=  read -r -d $'\0'; do
```

```
for_QC+=("$REPLY")
done < <(find . -type f -iname "*Aligned.sortedByCoord.out.bam" -print0)

for i in ${for_QC[@]}
do
echo "Found File: $i"
my_path=$(echo $i | grep -Po ".*(?=_Aligned.sortedByCoord.out.bam)")
echo "Running Metagene script. Outfile Name Set as $my_path"
metagene count --count_files $i --fiveprime_variable --offset $my_path"_p_offsets.txt" --normalize_over
20 50 --min_counts 50 /path/to/alignment/Plastid_Metagene/mm10_CDSStart_50bp_rois.txt $my_path
echo "..done!"
echo "Running Phase by Size Script. Outfile Name Set as $my_path"
phase_by_size --count_files $i --countfile_format BAM --fiveprime_variable --offset
$my_path"_p_offsets.txt" /path/to/alignment/Plastid_Metagene/mm10_CDSStart_50bp_rois.txt $my_path
echo "...done!"
done

#Merging biological replicates and post-process
mkdir ./merged_MexWT_Ngn
samtools merge -O BAM --threads 20 ./merged_MexWT_Ngn/merged_MexWT_Ngn.bam
/path/to/4_16_Ribo_MexWT_KO_Ngn/STARout_Ribo-MexWT-Ngn-1/Ribo-MexWT-Ngn-
1_Aligned.sortedByCoord.out.bam /path/to/4_16_Ribo_MexWT_KO_Ngn/STARout_Ribo-MexWT-Ngn-
2/Ribo-MexWT-Ngn-2_Aligned.sortedByCoord.out.bam

mkdir ./merged_MexKO_Ngn
samtools merge -O BAM --threads 20 ./merged_MexKO_Ngn/merged_MexKO_Ngn.bam
/path/to/4_16_Ribo_MexWT_KO_Ngn/STARout_Ribo-MexKO-Ngn-1/Ribo-MexKO-Ngn-
1_Aligned.sortedByCoord.out.bam /path/to/4_16_Ribo_MexWT_KO_Ngn/STARout_Ribo-MexKO-Ngn-
2/Ribo-MexKO-Ngn-2_Aligned.sortedByCoord.out.bam

merged_postproc=()
while IFS=  read -r -d $'\0'; do
merged_postproc+=("$REPLY")
done < <(find . -type f -iname "*merged*bam" -print0)

for i in ${merged_postproc[@]}
do
echo "Found File: $i"
echo "Indexing..."
samtools index $i
echo "done!"
my_path=$(echo $i | grep -Po ".*(?=.bam)")
echo "Running psite script. Outfile Name Set as $my_path"
psite --count_files $i --countfile_format BAM --min_counts 50 --require_upstream --min_length 17 --
max_length 35 /path/to/alignment/Plastid_Metagene/mm10_CDSStart_50bp_rois.txt $my_path
echo "done!"
read -p "Please edit the P Offset file located in folder $my_path, then press enter to continue"
echo Making Wiggle Files. Outfile Name Set as $my_path. Using offset file $my_path"_p_offsets.txt"
make_wiggle --count_files $i --countfile_format BAM --min_length 17 --fiveprime_variable --offset
$my_path"_p_offsets.txt" -o $my_path
echo "done!"
echo "Preparing Bigwig files"
wigToBigWig $my_path"_fw.wig" /path/to/alignment/mm10annotation_ensembl/mm10.chrom.sizes
$my_path"_fw.wig.bw"
wigToBigWig $my_path"_rc.wig" /path/to/alignment/mm10annotation_ensembl/mm10.chrom.sizes
$my_path"_rc.wig.bw"
```

```
echo "Removing Intermediate Wig Files"
rm $my_path"_fw.wig"
rm $my_path"_rc.wig"
echo "Running Metagene script. Outfile Name Set as $my_path"
metagene count --count_files $i --fiveprime_variable --offset $my_path"_p_offsets.txt" --normalize_over
20 50 --min_counts 50 /path/to/alignment/Plastid_Metagene/mm10_CDSStart_50bp_rois.txt $my_path
echo "..done!"
echo "Running Phase by Size Script. Outfile Name Set as $my_path"
phase_by_size --count_files $i --countfile_format BAM --fiveprime_variable --offset
$my_path"_p_offsets.txt" /path/to/alignment/Plastid_Metagene/mm10_CDSStart_50bp_rois.txt $my_path
echo "...done!"
done

##############################################################################################
######                    Pre-processing, alignment and post-processing of RNA-seq libraries
##############################################################################################

#RNA data in a separate folder
cd /path/to/6_18_RNA_NgnAtfOmp_P2_M71

#Clean up bcl2fastq output by finding all the fastq files in the file tree and move to the top level
array=()
while IFS=  read -r -d $'\0'; do
array+=("$REPLY")
done < <(find . -type f -regex ".*fastq.gz" -print0)

for i in "${array[@]}"
do
mv $i ./
  done

#Clean the empty file tree
rm -r 6_18_RNA_NgnAtfOmp_P2_M71-137389256

#Concatenate samples together with a useful name
samples=("RNA-MexWT-Ngn-1" "RNA-MexWT-Ngn-2""RNA-MexKO-Ngn-1" "RNA-MexKO-Ngn-2")
for i in ${samples[@]}
do
echo "Working on Sample: "$i
echo "Found the following files for concatenation to Read 1. Order is respected"
find . -maxdepth 1 -iregex ".*$i.*R1.*" -print0 | sort -z | xargs -0
find . -maxdepth 1 -iregex ".*$i.*R1.*" -print0 | sort -z | xargs -0 cat >> "$i".R1.fastq.gz
echo "Concatenated Read 1 files to: $i.R1.fastq.gz"
done
#Again see above note, $i.fastq.gz created first in the pipe so cat tells you it ignores it (see
https://superuser.com/questions/703887/cat-input-file-is-output-file-while-trying-to-combine-multiple-files-
into-one )

#Clean up non-concatenated files
rm *R1_001* #these are ALL saved in the raw form with native file structure on my external hard drive,
raw fastq folder.

  #Run FastQC
  mkdir fastqc_out
fastqc -o ./fastqc_out *gz
```

```
#Identify the R1 Files (the idea here is to keep things flexible- obviously we could hard code R1 into stuff
and just use the names)
fq_files=()
while IFS=  read -r -d $'\0'; do
fq_files+=("$REPLY")
done < <(find . -type f -regex ".*R1.fastq.gz" -print0)

#Align the Data
for i in ${fq_files[@]}; do
echo Identified File $i
my_path=$(echo $i | grep -Po ".*(?=.R1.fastq.gz)")
my_name=$(echo $i | grep -Po "(?<=./).*(?=.R1.fastq.gz)")
echo Extracted name $my_name
echo Preparing Intermediate RevComp File at $my_path"REVCOMP.R1.fastq"
zcat $my_path".R1.fastq.gz" | fastx_reverse_complement -Q33 > $my_path"REVCOMP.R1.fastq"
#careful of disk size here, but avoid time to re-compress
echo Created directory ./STARout_$my_name
mkdir ./STARout_$my_name
echo Initated Star Mapping to ./STARout_$my_name"/"$my_name"_"
STAR --runMode alignReads --runThreadN 20 --genomeDir /path/to/alignment/STARDir_Coding75nt --
outFileNamePrefix ./STARout_$my_name"/"$my_name"_" --outSAMtype BAM SortedByCoordinate --
outFilterType BySJout --outFilterIntronMotifs RemoveNoncanonicalUnannotated --outSAMstrandField
intronMotif --outFilterMultimapNmax 10 --readFilesIn $my_path"REVCOMP.R1.fastq"
done

#Remove the reverse complement
rm *REVCOMP*

  #Extract Q30, Create Bigwigs and Count
  rna_bams=()
while IFS=  read -r -d $'\0'; do
rna_bams+=("$REPLY")
done < <(find . -iname *bam -type f -print0)

for i in ${rna_bams[@]}; do
echo Working on file $i
my_path=$(echo $i | grep -Po ".*(?=_Aligned.sortedByCoord.out.bam)")
samtools view -b -q 30 --threads 20 -o $my_path"_q30.bam" $i
echo Indexing file $my_path"_q30.bam"
samtools index $my_path"_q30.bam"
echo Generating wiggle files with output $my_path"_q30"
make_wiggle --count_files $my_path"_q30.bam" --countfile_format BAM --min_length 17 --fiveprime -o
$my_path"_q30"
echo Generating bigwigs...
wigToBigWig $my_path"_q30_fw.wig" /path/to/alignment/mm10annotation_ensembl/mm10.chrom.sizes
$my_path"_q30_fw.wig.bw"
wigToBigWig $my_path"_q30_rc.wig" /path/to/alignment/mm10annotation_ensembl/mm10.chrom.sizes
$my_path"_q30_rc.wig.bw"
echo Removing wiggles...
rm $my_path"_q30_fw.wig"
rm $my_path"_q30_rc.wig"
echo '...done!'
done

#Clean up Raw fastq
mkdir raw_fastq
```

```
mv *gz ./raw_fastq

#Merge biological replicates
mkdir ./Merged_RNA_Bams

#Some libraries were sequenced across two separate RNA-seq runs in order to get the proper number of reads
#First needed to combine these two runs. The following code automates this process:
#First, look at the libraries in the current sequencing replicate
libs=()
while IFS=  read -r -d $'\0'; do
libs+=("$REPLY")
done < <(find . -type f -regex ".*q30.bam" -print0)

#For each library in the current set, find the corresponding bam from the first round of sequencing and merge
for current_lib in ${libs[@]}; do
#Parse File Info
echo Received file $current_lib
lib_name=$(echo $current_lib | grep -Po "(?<=RNA-).*(?=/)")
echo Extracted file name $lib_name
#
#Find possible matches for old file
old_file=()
while IFS=  read -r -d $'\0'; do
old_file+=("$REPLY")
done < <(find ../6_10_Rnaseq_NgnAtfOmp_P2/ -iname *$lib_name*q30.bam -type f -print0)
#
#Report Back on Matches
echo Found ${#old_file[@]} "Old File(s), listed below:"
  for i in ${old_file[@]}; do echo $i; done
  #
  #Merge Matches
  if [ ${#old_file[@]} -eq 1 ];
    then
    echo Merging $current_lib and ${old_file[0]} to output ./Merged_RNA_Bams/$lib_name"_merged.q30.bam"
    samtools merge -O BAM --threads 20 ./Merged_RNA_Bams/$lib_name"_merged.q30.bam" $current_lib ${old_file[0]}
    echo Done Merging...
    fi
    echo .....done.....
    done

    #Now that multiple runs of the samples have been combined, merge biological replicates
    cd ./Merged_RNA_Bams
    samtools merge -O BAM --threads 20 MexKO-Ngn-AllMerged.q30.bam MexKO-Ngn-1_merged.q30.bam MexKO-Ngn-2_merged.q30.bam
    samtools merge -O BAM --threads 20 MexWT-Ngn-AllMerged.q30.bam /path/to/6_10_Rnaseq_NgnAtfOmp_P2/STARout_RNA-MexWT-Ngn-1/RNA-MexWT-Ngn-1_Aligned.sortedByCoord.out.bam /path/to/6_10_Rnaseq_NgnAtfOmp_P2/STARout_RNA-MexWT-Ngn-2/RNA-MexWT-Ngn-2_Aligned.sortedByCoord.out.bam
    cd ..

    #Index the merged files
    bams=()
```

```
    while IFS=  read -r -d $'\0'; do
    bams+=("$REPLY")
    done < <(find ./Merged_RNA_Bams/ -type f -regex ".*bam" -print0)

    for i in ${bams[@]}; do samtools index $i; done


################################################################################
######                                  Final Read counting
################################################################################

    #Featurecounts_1_4 is my custom code (provided). Please do not confuse with the actual
featurecounts software!
    #Necessary .obj files have been provided for the sake of allowing this to be run.
    #Please ignore the uORF annotations part of this code, this element of translational regulation was not
ultimately studied in this paper and is thus not described in the methods.

    #MexWT Ngn
    featurecounts_1_4.py --run_mode both --transcript_annotations
/path/to/alignment/mm10annotation_ensembl/IS_merged_CodingTranscriptsHS_sorted_segmentchainlist
.obj --transcript_format Obj --uORF_annotations
/path/to/10_31_Ribo_WholeOE_TN_Har/STARout_harring/SVM_predict/uORFs_OE_crossvalidated_SV
M.obj --uORF_format Obj --path_to_ribprof_bam
/path/to/4_16_Ribo_MexWT_KO_Ngn/merged_MexWT_Ngn/merged_MexWT_Ngn.bam --path_to_psite
/path/to/4_16_Ribo_MexWT_KO_Ngn/merged_MexWT_Ngn/merged_MexWT_Ngn_p_offsets.txt --
path_to_rnaseq /path/to/6_18_RNA_NgnAtfOmp_P2_M71/Merged_RNA_Bams/MexWT-Ngn-
AllMerged.q30.bam --rnamapfactory fiveprime --nthreads 22 --outfile_prefix
./Merged_RNA_Bams/MexWT-Ngn-AllMerged_TEcounts

    #Mex KO Ngn
    featurecounts_1_4.py --run_mode both --transcript_annotations
/path/to/alignment/mm10annotation_ensembl/IS_merged_CodingTranscriptsHS_sorted_segmentchainlist
.obj --transcript_format Obj --uORF_annotations
/path/to/10_31_Ribo_WholeOE_TN_Har/STARout_harring/SVM_predict/uORFs_OE_crossvalidated_SV
M.obj --uORF_format Obj --path_to_ribprof_bam
/path/to/4_16_Ribo_MexWT_KO_Ngn/merged_MexKO_Ngn/merged_MexKO_Ngn.bam --path_to_psite
/path/to/4_16_Ribo_MexWT_KO_Ngn/merged_MexKO_Ngn/merged_MexKO_Ngn_p_offsets.txt --
path_to_rnaseq /path/to/6_18_RNA_NgnAtfOmp_P2_M71/Merged_RNA_Bams/MexKO-Ngn-
AllMerged.q30.bam --rnamapfactory fiveprime --nthreads 22 --outfile_prefix
./Merged_RNA_Bams/MexKO-Ngn-AllMerged_TEcounts

#####featurecounts_1_4.py code
    #!/usr/bin/env python

    #Import Relevent Packages
    from __future__ import division
    from plastid import GTF2_TranscriptAssembler, BAMGenomeArray, VariableFivePrimeMapFactory,
FivePrimeMapFactory, CenterMapFactory
    import twobitreader as twobit
    import numpy as np
    import re
    from collections import Counter, defaultdict
    import itertools
    from plastid.util.io.filters import CommentReader

    #Import Additional Relevent Packages
```

```
    import dill
    import pandas as pd
    from pathos.multiprocessing import ProcessingPool as Pool
    import copy
    import os
    import argparse
#############################################################################################
######                                    Define Functions
#############################################################################################
    #These functions are copied from WholeOE_uORFquant_script_12_9.py as I am going to write some RNA-seq functions here and I want to be able to just import all of the analysis functions from one place

    def meta_roi(segmentchain_list):
      """An old function that I wrote and have adopted here. Takes a list of segment chains, breaks them apart into constituent genomic segments, and throws them back together into a segment chain representing a meta-window.

    Function was updated on 12/14/18 to deepcopy starter segment for construcitng the meta window. This ensures that input segmentchain list is not altered by the function.

    --Input--
    segmentchain_list: a list of plastid segmentchains or plastid transcripts.

    --Output--
    A single plastid segmentchain representing the union of all input segmentchains. Attributes are propagated from the first segmentchain in the input list.
    """
    if type(segmentchain_list) is str:
      return(segmentchain_list)

    elif (len(segmentchain_list) == 1):
      uORF_meta = segmentchain_list[0]
    return uORF_meta

    else:
      # Figure out how to chain the genomic segments together
      segments_list = []
    for i in segmentchain_list[1:len(segmentchain_list)]:
      segments_list.append(i.segments)

    segments_flattened = list(itertools.chain.from_iterable(segments_list))
    uORF_meta = copy.deepcopy(segmentchain_list[0]) #implemented deepcopy approach here which is necessary if you want to call this again.
    uORF_meta.add_segments(*segments_flattened)
    return uORF_meta

    def get_counts_and_lengths_masked(input_segment_chain, mapped_read_array, masked_logical, keep_true):
      """A basic method for getting counts and lengths from a plastid segmentchain and a plastid BAMGenome Array. Advantage over get_counts is that it provides methods for handling masked regions and functionality for keeping either masked OR unmasked region read/length counts.

    --Input--
    input_segment_chain: plastid segment_chain_object to calculate parameters over
    mapped_read_array: plastid BAMGenomeArray with mapping set, containing reads that need to be counted
```

```
    masked_logical: (True|False) do you want to look at masks that exist on the input segment_chain? Masks must have been previously added with .add_masks method
    keep_true: (string: 'yes'|'no') applicable only when masked_logical=True. Do you want to count reads/length for masked region (yes) or the NON-masked region (no)

    --Output--
    a tuple: (counts, length) for input_segment_chain given mapped_read_array
    """
    if masked_logical:
      masked_counts = input_segment_chain.get_masked_counts(mapped_read_array)
    if keep_true == 'yes':
      roi_masked_counts = input_segment_chain.get_counts(mapped_read_array)[masked_counts.mask] #keep only reads in masked region
    my_counts = np.nansum(roi_masked_counts)
    my_length = len(roi_masked_counts)
    #
    else:
      roi_masked_counts = input_segment_chain.get_counts(mapped_read_array)[np.invert(masked_counts.mask)] #Trick b/c get_masked_counts doesnt count the masked regions
    my_counts = np.nansum(roi_masked_counts)
    my_length = len(roi_masked_counts)
    #
    #
    else:
      my_counts = np.nansum(input_segment_chain.get_counts(mapped_read_array))
    my_length = input_segment_chain.get_length()
    #
    return my_counts, my_length

    def genewise_counter_riboprof(run_mode, gene_id, full_transcript_list, uORF_list, path_to_ribprof_bam, path_to_psite, path_to_rnaseq, rnamapfactory):
      """A genewise counter function for ribosome profiling and RNA-seq data. When run in rnaseq mode, returns cds and full transcript counts as well as lengths. When run in riboprof mode, returns ribosome profiling counts and lengths for various regions of the meta-transcript (UTR/CDS/uORFs) under various sets of assumptions as detailed below. When run in 'both' mode, returns both ribosome profiling and RNAseq read counts, along with a translational efficiency that is calculated internally.

    --Input--
    run_mode: str('riboprof' | 'rnaseq' | 'both') run mode to use for counting. See output below.
    gene_ID: string representing gene ID to be quantified
    full_transcript_list: list of complete plastid transcript objects to be concatenated for the given gene_id. Ensure these are complete objects, the UTR/CDS parsing will happen internally.
    uORF_list: list of all uORFs to be included in for the given gene_id. These should be plastid segmentchains. The meta roi will be constructed internally. If there are no uORFs, you must pass an empty list to this paramater (ie. uORF_list = [])
    path_to_ribprof_bam: string giving path to ribosome profiling CHX reads. Required if mode is "riboprof" or "both". Otherwise pass 'None'.
    path_to_psite: string giving path to file containing table of read lengths:psite offsets (generated by plastid psite script). Required if run_mode is "riboprof" or "both". Otherwise pass 'None'.
    path_to_rnaseq: string giving path to mRNA-seq reads. Required if mode is "rnaseq" or "both". Otherwise pass 'None'.
    rnamapfactory: str('fiveprime'|'center') detailing which map factory to use for counting rnaseq data. 'fiveprime' will use plastid's FivePrimeMapFactory which will assign a value of 1 to the 5'-most base of the read. 'center' will use plastid's CenterMapFactory which assigns a value of 1/L to every position on read of length L. Use 'fiveprime' for tools like DESeq2.
```

```
    --Output--
    'rnaseq' mode: a list of length 6- [gene_id, gene_name, reads in meta-cds, length of meta-cds, reads in
meta-transcript, length of meta-transcript]. Gene name is taken from attributes of input transcript list,
meta-transcript and meta-cds regions are constructed via internal calls to meta-roi using the full transcript
list and the annotated CDS from that full transcript list respectively.
    'riboprof' mode: a list of length 12- [gene_id, gene_name, utr5_counts, utr5_len, cds_counts, cds_len,
utr3_counts, utr3_len, meta_uORF_counts, meta_uORF_len, meta_uORF_counts_in5utronly_noCDS,
meta_uORF_len_in5utronly_noCDS].
    utrs are defined as the meta-region constructed from 5utr and 3utr annotations of input transcript list.
These regions are masked by the meta-CDS, to ensure that reads are not counted twice.
    'meta_uORF' parameters refer to meta-regions constructed via a call to meta-uORF with input
uORF_list. No masks are applied.
    'meta_uORF_in5utronly_noCDS' parameters are the result of masking meta-uORF region with the
5'utr, which is itself masked by the CDS. These parameters refer to the region of the meta-uORF which is
in the 5'UTR and does not overlap CDS on any transcript for that gene.
    'both' mode: a list of length 15- [gene_id, gene_name, ribosome profiling utr5 counts, utr5_length,
ribosome profiling cds_counts, cds_len, ribosome profiling utr3 counts, utr3_len, ribosome profiling
meta_uORF_counts, meta_uORF_len, ribosome profiling meta_uORF_counts_in5utronly_noCDS,
meta_uORF_len_in5utronly_noCDS, rnaseq counts cds, rnaseq counts full meta-transcript, length full
meta-transcript]
    ribosome profiling parameters are precisely as documented above for 'riboprof' mode
    rnaseq parameters are as described above for 'rnaseq' mode
    full transcript length gives the length of the full meta-transcript constructed via a call to meta-roi using
the input full_transcript_list. This is useful for calculating rpkm for the full transcript.
    """

    print 'Working on ' + str(gene_id)
    #
    #Test run mode. Now implemented at top level in wrapper function.
    #if run_mode not in ['riboprof', 'rnaseq', 'both']:
    #    raise Exception("Did not understand the value for run_mode")
    #
    #Set up meta-window annotations for given gene using full transcript annotations
    CDS_list = [i.get_cds() for i in full_transcript_list]
    utr5_list = [i.get_utr5() for i in full_transcript_list]
    utr3_list = [i.get_utr3() for i in full_transcript_list]
    meta_CDS = meta_roi(CDS_list)
    meta_transcript = meta_roi(full_transcript_list)

    #Import the reads for rna
    if run_mode == 'rnaseq' or run_mode == 'both':
      rnaseq_reads = BAMGenomeArray(path_to_rnaseq)
    if rnamapfactory == 'fiveprime':
      rnaseq_reads.set_mapping(FivePrimeMapFactory())
    elif rnamapfactory == 'center':
      rnaseq_reads.set_mapping(CenterMapFactory())
    else:
      raise Exception("Did not understand the value for rnamapfactory.")
    #
    #

    #Import the reads for ribosome profiling
    if run_mode == 'riboprof' or run_mode == 'both':
      CHX_reads = BAMGenomeArray(path_to_ribprof_bam)
    CHX_reads.set_mapping(VariableFivePrimeMapFactory.from_file(open(path_to_psite)))
```

```
    #

    #RNA-seq read counting
    if run_mode == 'rnaseq' or run_mode == 'both':
      cds_rnaseq_counts = np.nansum(meta_CDS.get_counts(rnaseq_reads))
    full_transcript_rnaseq_counts = np.nansum(meta_transcript.get_counts(rnaseq_reads))
    full_transcript_meta_length = meta_transcript.get_length()
    if run_mode == 'rnaseq':
      output_line = [gene_id, full_transcript_list[0].attr['gene_name'], cds_rnaseq_counts,
meta_CDS.get_length(), full_transcript_rnaseq_counts, full_transcript_meta_length]
    print '...done!'
    return output_line
    #
    #

    #Ribosome profiling analysis, used for run_mode = 'both' or 'riboprof'
    meta_utr5 = meta_roi(utr5_list)
    meta_utr3 = meta_roi(utr3_list)

    #Mask CDS overlap of 5'UTR and 3'UTR windows. For calculating 5'UTR/CDS/3'UTR ratios most
efficiently
    meta_utr5.add_masks(*meta_CDS.segments)
    meta_utr3.add_masks(*meta_CDS.segments)

    #Import Reads and Set mapping
    CHX_reads = BAMGenomeArray(path_to_ribprof_bam)
    CHX_reads.set_mapping(VariableFivePrimeMapFactory.from_file(open(path_to_psite))) #doesn't read
string like it should, so my workaround is to give it the open file handle instead

    #Calculate Everything Except for uORF Parameters
    utr5_counts_maskedbyCDS, utr5_len_maskedbyCDS =
get_counts_and_lengths_masked(input_segment_chain=meta_utr5, mapped_read_array=CHX_reads,
masked_logical=True, keep_true='no')
    cds_counts, cds_len = get_counts_and_lengths_masked(input_segment_chain=meta_CDS,
mapped_read_array=CHX_reads, masked_logical=False, keep_true='no')
    utr3_counts_maskedbyCDS, utr3_len_maskedbyCDS =
get_counts_and_lengths_masked(input_segment_chain=meta_utr3, mapped_read_array=CHX_reads,
masked_logical=True, keep_true='no')

    #Build in if statements b/c may not have uORFs
    if len(uORF_list)!= 0:
      meta_uORF = meta_roi(uORF_list)
    meta_uORF_counts, meta_uORF_len =
get_counts_and_lengths_masked(input_segment_chain=meta_uORF, mapped_read_array=CHX_reads,
masked_logical=False, keep_true='no') #whole meta_uORF used for uORF/CDS ratio count.
    meta_uORF.add_masks(*meta_utr5.segments)
    meta_uORF_in5utr = meta_uORF.get_masks_as_segmentchain() #takes the intersection of uORF and
5'UTR
    meta_uORF_in5utr.add_masks(*meta_CDS.segments) #masks the CDs
    meta_uORF_counts_in5utronly_noCDS, meta_uORF_len_in5utronly_noCDS =
get_counts_and_lengths_masked(input_segment_chain=meta_uORF_in5utr,
mapped_read_array=CHX_reads, masked_logical=True, keep_true='no') #used for uORF/5'UTR
    else:
      meta_uORF_counts, meta_uORF_len = (np.nan, np.nan)
    meta_uORF_counts_in5utronly_noCDS, meta_uORF_len_in5utronly_noCDS = (np.nan, np.nan)
    #
```

```
    #

    #Output for 'riboprof mode'
    if run_mode == 'riboprof':
      output_line = [gene_id, full_transcript_list[0].attr['gene_name'], utr5_counts_maskedbyCDS,
utr5_len_maskedbyCDS, cds_counts, cds_len, utr3_counts_maskedbyCDS, utr3_len_maskedbyCDS,
meta_uORF_counts, meta_uORF_len, meta_uORF_counts_in5utronly_noCDS,
meta_uORF_len_in5utronly_noCDS]
    print '...done!'
    return output_line
    #
    if run_mode == 'both':
      output_line = [gene_id, full_transcript_list[0].attr['gene_name'], utr5_counts_maskedbyCDS,
utr5_len_maskedbyCDS, cds_counts, cds_len, utr3_counts_maskedbyCDS, utr3_len_maskedbyCDS,
meta_uORF_counts, meta_uORF_len, meta_uORF_counts_in5utronly_noCDS,
meta_uORF_len_in5utronly_noCDS, cds_rnaseq_counts, full_transcript_rnaseq_counts,
full_transcript_meta_length]
    print '...done!'
    return output_line
    #

    def feature_counter(run_mode, list_of_transcripts, list_of_uORFs, path_to_ribprof_bam, path_to_psite,
path_to_rnaseq, rnamapfactory, nthreads):
      """Wrapper function for genewise_counter_riboprof that parses input transcripts/uORF into groups by
gene, generates genewise lists that are fed into genewise_counter_riboprof in a multithreaded way, and
creates a clean pandas dataframe with count paramaters for all genes analyzed. Calculates relevent
parameters like rpkm for rnaseq and translational efficiency, which require all data in aggreate.

    --Input--
    run_mode: str('riboprof' | 'rnaseq' | 'both') run mode to use for counting. Passed to
genewise_counter_riboprof internally. See output below.
    list_of_transcripts: list of all annotated transcripts in genome to be quantified (eg. all coding transcripts
from mm10). Transcripts should be represented as plastid transcript objects, with a 'gene_id' attribute that
will be used to group transcripts by gene for meta-region construction and analysis. Transcripts should
have a 'gene_name' attribute which will be included in the output table (assuming that gene_id is a UCSC
or Ensembl ID that is not particuarily helpful).
    list_of_uORFs: list of all uORFs to be included in the analysis. uORFs should be plastid
segmentchains, with a gene_id that can be cross-referenced with the list_of_transcripts.
    path_to_ribprof_bam: string giving path to ribosome profiling CHX reads. Required if mode is "riboprof"
or "both". Otherwise pass 'None'.
    path_to_psite: string giving path to file containing table of read lengths:psite offsets (generated by
plastid psite script). Required if run_mode is "riboprof" or "both". Otherwise pass 'None'.
    path_to_rnaseq: string giving path to mRNA-seq reads. Required if mode is "rnaseq" or "both".
Otherwise pass 'None'.
    rnamapfactory: str('fiveprime'|'center') detailing which map factory to use for counting rnaseq data.
'fiveprime' will use plastid's FivePrimeMapFactory which will assign a value of 1 to the 5'-most base of the
read. 'center' will use plastid's CenterMapFactory which assigns a value of 1/L to every position on read
of length L. Use 'fiveprime' for tools like DESeq2.
    nthreads: number of threads to be used for multiprocessing.

    --Output--
    'rnaseq' mode: a pandas dataframe of shape (number_genes_input, 7). Columns: [gene_id,
gene_name, reads in meta-cds, length of meta-cds, reads in meta-transcript, length of meta-transcript,
rpkm].
    Gene name is taken from attributes of input transcript list
```

Meta-transcript and meta-cds regions are constructed via internal calls to meta-roi using the full transcript list and the annotated CDS from that full transcript list respectively.

RPKM is definied for the total meta-transcript: (reads in meta-transcript) / (length of meta-transcript in kb * total number of reads in any annotated meta-transcript) * 1,000,000.

'riboprof' mode: a pandas dataframe of shape (number_genes_input, 12). Columns: [gene_id, gene_name, utr5_counts, utr5_len, cds_counts, cds_len, utr3_counts, utr3_len, meta_uORF_counts, meta_uORF_len, meta_uORF_counts_in5utronly_noCDS, meta_uORF_len_in5utronly_noCDS].

utrs are defined as the meta-region constructed from 5utr and 3utr annotations of input transcript list. These regions are masked by the meta-CDS, to ensure that reads are not counted twice.

'meta_uORF' parameters refer to meta-regions constructed via a call to meta-uORF with input uORF_list. No masks are applied.

'meta_uORF_in5utronly_noCDS' parameters are the result of masking meta-uORF region with the 5'utr, which is itself masked by the CDS. These parameters refer to the region of the meta-uORF which is in the 5'UTR and does not overlap CDS on any transcript for that gene.

'both' mode: a pandas dataframe of shape (number_genes_input, 16). Columns: [gene_id, gene_name, ribosome profiling utr5 counts, utr5_length, ribosome profiling cds_counts, cds_len, ribosome profiling utr3 counts, utr3_len, ribosome profiling meta_uORF_counts, meta_uORF_len, ribosome profiling meta_uORF_counts_in5utronly_noCDS, meta_uORF_len_in5utronly_noCDS, rnaseq counts cds, rnaseq counts full meta-transcript, length full meta-transcript, translation_efficiency]

ribosome profiling parameters are precisely as documented above for 'riboprof' mode

rnaseq parameters are as described above for 'rnaseq' mode

full transcript length gives the length of the full meta-transcript constructed via a call to meta-roi using the input full_transcript_list.

translational efficiency follows the definition of Ingolia et al, 2011: (riboprof_reads_meta_cds / [length meta_CDS in kb * total number of ribosome reads in any region of any annotated transcript]) / (rnaseq_reads_meta_cds / [length meta_CDS in kb * total number of rnaseq reads in any region of any annotated transcript])

Note that the Ingolia definition of TE normalizes to reads in any region of any annotated transcript (UTR or CDS), rather than the total number of reads that STAR could align. This avoids confounding by rRNA or small ncRNA, which STAR can align, but may differ between experiments.

"""

```
#Test run mode.
if run_mode not in ['riboprof', 'rnaseq', 'both']:
  raise Exception("Did not understand the value for run_mode")
#
print "Creating Dictionaries..."
#Create full transcript dictionary
full_transcript_dict = defaultdict(list)
for transcript in list_of_transcripts:
  full_transcript_dict[transcript.get_gene()].append(transcript)
#
#Create meta-uORF dictionary
uORF_dict = defaultdict(list)
for uORF in list_of_uORFs:
  uORF_dict[uORF.attr['gene_id']].append(uORF)
#
#Merged dictionary has structure gene_id: list of 2 [full transcripts list, uORFs list]
gene_id_list = []
full_transcript_list_of_lists = []
uORF_list_of_lists = []
for gene_id in full_transcript_dict.iterkeys():
  gene_id_list.append(gene_id)
full_transcript_list_of_lists.append(full_transcript_dict[gene_id])
try:
  uORF_list_of_lists.append(uORF_dict[gene_id])
except KeyError:
```

```
      uORF_list_of_lists.append([])
    #

    #Run the loop
    print "Counting Features..."
    output_list = Pool(nthreads).map(genewise_counter_riboprof, itertools.repeat(run_mode), gene_id_list,
full_transcript_list_of_lists, uORF_list_of_lists, itertools.repeat(path_to_ribprof_bam),
itertools.repeat(path_to_psite), itertools.repeat(path_to_rnaseq), itertools.repeat(rnamapfactory))

    print "Concatenating Dataframes..."
    if run_mode == 'rnaseq':
      output_df = pd.DataFrame(output_list, columns = ["gene_id", "gene_name", "rnaseq_cds_counts",
"cds_len", "rnaseq_total_counts", "total_len"])
    output_df['rnaseq_total_rpkm'] = output_df.rnaseq_total_counts / ((output_df.total_len/1000) *
np.nansum(output_df.rnaseq_total_counts)) * 1000000
    print "...done!"
    return output_df
    elif run_mode == 'riboprof':
      output_df = pd.DataFrame(output_list, columns = ["gene_id", "gene_name", "ribo_utr5_counts",
"utr5_len", "ribo_cds_counts", "cds_len", "ribo_utr3_counts", "utr3_len", "ribo_uORF_total_counts",
"uORF_total_len", "ribo_uORF_utr5_counts", "uORF_utr5_len"])
    print "...done!"
    return output_df
    else:
      output_df = pd.DataFrame(output_list, columns = ["gene_id", "gene_name", "ribo_utr5_counts",
"utr5_len", "ribo_cds_counts", "cds_len", "ribo_utr3_counts", "utr3_len", "ribo_uORF_total_counts",
"uORF_total_len", "ribo_uORF_utr5_counts", "uORF_utr5_len", "rnaseq_cds_counts",
"rnaseq_total_counts", "total_transcript_len"])
    total_ribo_reads = np.nansum(output_df.ribo_utr5_counts) + np.nansum(output_df.ribo_cds_counts) +
np.nansum(output_df.ribo_utr3_counts)
    output_df['translation_efficiency'] = (output_df.ribo_cds_counts / total_ribo_reads) /
(output_df.rnaseq_cds_counts / np.nansum(output_df.rnaseq_total_counts)) #cds_length divides out
    print '...done!'
    return output_df
    #

    def main():
      """
    Description
    ---------------------------------------------------------------------
    A read counting program for RNA-seq or ribosome profiling applications.

    --'rnaseq' mode --
    Returns a tab-delimited table of shape (number_genes_input, 7). Columns: [gene_id, gene_name,
reads in meta-cds, length of meta-cds, reads in meta-transcript, length of meta-transcript, rpkm].
    Meta-transcripts are compiled for a given gene_id (attribute of plastid obj or field 9 of GTF) by grouping
all full transcripts (--transcript_annotations) with that gene_id and collapsing into a single maximal
spanning window.
    RPKM is definied for the total meta-transcript: (reads in meta-transcript) / (length of meta-transcript in
kb * total number of reads in any annotated meta-transcript) * 1,000,000.

    --'riboprof' mode--
    Returns a tab-delimited table of shape (number_genes_input, 12). Columns: [gene_id, gene_name,
utr5_counts, utr5_len, cds_counts, cds_len, utr3_counts, utr3_len, meta_uORF_counts, meta_uORF_len,
meta_uORF_counts_in5utronly_noCDS, meta_uORF_len_in5utronly_noCDS].
```

utrs are defined as the meta-region constructed from 5utr and 3utr annotations of input transcript list. These regions are masked by the meta-CDS, to ensure that reads are not counted twice.

'meta_uORF' parameters refer to meta-regions constructed via a call to meta-uORF with input uORF_list. No masks are applied.

'meta_uORF_in5utronly_noCDS' parameters are the result of masking meta-uORF region with the 5'utr, which is itself masked by the CDS. These parameters refer to the region of the meta-uORF which is in the 5'UTR and does not overlap CDS on any transcript for that gene.

--'both' mode--

Returns a tab-delimited table of shape (number_genes_input, 16). Columns: [gene_id, gene_name, ribosome profiling utr5 counts, utr5_length, ribosome profiling cds_counts, cds_len, ribosome profiling utr3 counts, utr3_len, ribosome profiling meta_uORF_counts, meta_uORF_len, ribosome profiling meta_uORF_counts_in5utronly_noCDS, meta_uORF_len_in5utronly_noCDS, rnaseq counts cds, rnaseq counts full meta-transcript, length full meta-transcript, translation_efficiency]

ribosome profiling parameters are precisely as documented above for 'riboprof' mode

rnaseq parameters are as described above for 'rnaseq' mode

full transcript length gives the length of the full meta-transcript constructed via a call to meta-roi using the input full_transcript_list.

translational efficiency follows the definition of Ingolia et al, 2011: (riboprof_reads_meta_cds / [length meta_CDS in kb * total number of ribosome reads in any region of any annotated transcript]) / (rnaseq_reads_meta_cds / [length meta_CDS in kb * total number of rnaseq reads in any region of any annotated transcript])

Note that the Ingolia definition of TE normalizes to reads in any region of any annotated transcript (UTR or CDS), rather than the total number of reads that STAR could align. This avoids confounding by rRNA or small ncRNA, which STAR can align, but may differ between experiments.

"""

```
parser = argparse.ArgumentParser(description=main.__doc__,
formatter_class=argparse.RawDescriptionHelpFormatter)
parser.add_argument('--run_mode', required=True, choices=['riboprof', 'rnaseq', 'both'], help='Required. str(riboprof|rnaseq|both). Run mode to use for counting.')
parser.add_argument('--transcript_annotations', help='Required. Path to file containing transcript annotations in GTF2 format or as a list of plastid transcript objects that have been pickled using dill. The gene_id attribute (column 9 of GTF) will be used to group transcripts by gene- so please ensure this is correct. Ensembl annotations are preferred for this reason.', required=True)
parser.add_argument('--transcript_format', choices=['GTF2','Obj'], help='Required. str(GTF2|Obj) specifying if transcript annotations are in GTF2 format or a serialized list of plastid transcripts. Usual security concerns apply when using pickled objects', required=True)
parser.add_argument('--uORF_annotations', default=[], help='Path to file containing uORF annotations in GTF2 format or as a list of plastid segmentchain objects. Special keyword "none" (default) can be used if no uORF annotations are desired. If using a GTF file, be especially careful that the gene_id attribute from column 9 of the GTF is correct and agrees with the gene ID of the appropriate transcripts in your transcript_annotations, as this will be used to group the features and pair uORFs/full transcript annotations')
parser.add_argument('--uORF_format', choices = ['GTF2','Obj'], help='Required only if uORF_annotations != "none", str(GTF2|Obj) specifying if uORF annotations are in GTF2 format or are a serilarized list of plastid segmentchain objects. Usual security concerns apply with de-pickling so be careful.')
parser.add_argument('--path_to_ribprof_bam', default=None, help='String specifying path to BAM file containing ribosome profiling read alignments. Required if mode is "riboprof" or "both".')
parser.add_argument('--path_to_psite', default=None, help='String specifying path to string giving path to file containing table of read lengths:psite offsets (generated by plastid psite script). Required if run_mode is "riboprof" or "both".')
parser.add_argument('--path_to_rnaseq', default=None, help='String specifying path to BAM file containing RNA-seq read alignments. Required if mode is "rnaseq" or "both".')
```

```
    parser.add_argument('--rnamapfactory', default='fiveprime', choices=['fiveprime','center'],
help="str('fiveprime'|'center') detailing which map factory to use for counting rnaseq data. 'fiveprime'
(default) will use plastid's FivePrimeMapFactory, assigning a count of 1 to the 5'-most base of the read.
'center' will use plastid's CenterMapFactory which assigns a value of 1/L to every position on read of
length L. You must use 'fiveprime' for tools that require integer values for counts, like DESeq2.")
    parser.add_argument('--nthreads', default= 4, type=int, help="Number of threads to be used for
multiprocessing. Default is 4. For optimal performance set as max processors - 1. ")
    parser.add_argument('--outfile_prefix', required=True, help='Required. Prefix for disk location where
output data table should be saved. The final .tsv will be appended by the program.')

    #Parse Args into Namespace object
    args = parser.parse_args()

    #Check that specified file output can safely be written.
    if os.path.exists(args.outfile_prefix + '.tsv'):
      raise ValueError('Output file already exists and will not be overwritten. Please specify a new
filename.')

    #Write arguements to output file as commented lines.
    fout = open(args.outfile_prefix + '.tsv', 'w')
    for arg, val in vars(args).iteritems():
      fout.write('#' + str(arg) + ': ' + str(val) + '\n')

    #Build Transcript annotations list from input GTF or load from serialized object
    if args.transcript_format == 'GTF2':
      print 'Assembling Transcripts from GTF2 File: ' + args.transcript_annotations
    annotation_list = list(GTF2_TranscriptAssembler(args.transcript_annotations))
    print '...done!'
    else:
      annotation_list = dill.load(open(args.transcript_annotations))
    print 'Successfullly assembled transcripts from serialized object: ' + args.transcript_annotations

    #Build uORF annotations list. uORF_annotations are not required, default will be list of zero. If string
(ie. filepath) specified, parse based on format.
    if isinstance(args.uORF_annotations,(str,)):
      if args.uORF_format == 'GTF2':
      print 'Assembling uORFs from GTF2 File: ' + args.uORF_annotations
    list_of_uORFs = list(GTF2_TranscriptAssembler(args.uORF_annotations))
    print '...done!'
    else:
      list_of_uORFs = dill.load(open(args.uORF_annotations))
    print 'Successfullly assembled uORFs from serilarized object: ' + args.uORF_annotations
    else:
      list_of_uORFs = args.uORF_annotations

    #Run the program
    output_frame = feature_counter(run_mode=args.run_mode, list_of_transcripts=annotation_list,
list_of_uORFs=list_of_uORFs, path_to_ribprof_bam=args.path_to_ribprof_bam,
path_to_psite=args.path_to_psite, path_to_rnaseq=args.path_to_rnaseq,
rnamapfactory=args.rnamapfactory, nthreads=args.nthreads)

    #Save file
    output_frame.to_csv(fout, sep='\t', header=True, index=False)
    fout.close()

    print 'Program Complete.'
```

```
    if __name__ == '__main__':
      main()
################################################################################################
######                                        R code to analyze aligned data
################################################################################################

    library(tidyverse)
    library(magrittr)
    library(reshape2)
    library(ggExtra)
    library(gridExtra)
    setwd('/path/to/data/8_22_Ribo_MexKO_Omp_OmpSinglePos/')

    #Read in the data
all_ribo_list <- list.files('../6_18_RNA_NgnAtfOmp_P2_M71/Merged_RNA_Bams', pattern = '.*tsv',
full.names = T) %>%
      c(., list.files('./', pattern = 'Merged_TEcounts.tsv', full.names = T, recursive = T)) %>%
      .[str_detect(.,'Mex(WT|KO)')]

raw_data <- lapply(all_ribo_list, function(filepath) {
      lib_name <- str_extract(filepath, '(?<=/)Atf5|OMP-MexWT|Mex(WT|KO)(-|_)(Ngn|Omp)|P2(?=-)')
      parsed_input <- read_tsv(filepath, col_names = T, comment = '#') %>%
        mutate('ribo_rpkm' = ribo_cds_counts/(cds_len/1000 * sum(ribo_cds_counts)) * 1000000,
               'rnaseq_fpkm' = rnaseq_cds_counts/(cds_len/1000 * sum(rnaseq_cds_counts)) * 1000000,
               'rnaseq_tpm' = rnaseq_fpkm/sum(rnaseq_fpkm) * 1000000) %>%
        reshape2::melt(id.vars = c('gene_id','gene_name')) %>%
        mutate('identity' = rep(lib_name, nrow(.)))
      return(parsed_input)
    }) %>%
      bind_rows()

    #Filter the Data
rachel_df <- raw_data %>%
      filter(str_detect(identity, 'Mex')) %>%
      mutate('genotype' = str_extract(.$identity, 'Mex(WT|KO)'),
             'cell_type' = str_extract(.$identity, 'Ngn|Omp|OMP')) %>%
      mutate('cell_type' = fct_recode(.$cell_type , Omp = "OMP"))

rachel_df %>%
      filter(variable %in% c('cds_len','ribo_cds_counts', 'rnaseq_cds_counts',
                             'ribo_rpkm','rnaseq_tpm', 'translation_efficiency')) %>%
      dcast(gene_id + gene_name ~ cell_type + genotype + variable, value.var = 'value') %>%
      write_tsv('/path/to/TE_RNASeq_Measurements_Raw.tsv')

#Will use cutoffs (empirically) of 1rpkm ribo-seq, 1tpm RNA-seq
high_confidence_rachel <- rachel_df %>%
      filter(variable %in% c('ribo_rpkm', 'rnaseq_tpm')) %>%
      group_by(gene_id, identity) %>%
      filter(all(value > 1)) %>%
      summarize(n=n()) %>%
      {semi_join(rachel_df, ., by = c('gene_id','identity'))} %>%
      filter(variable %in% c('translation_efficiency', 'rnaseq_tpm')) %>%
      dplyr::select(-identity) %>%
      dcast(...~ genotype, value.var = 'value') %>%
      mutate('log2FC_KO_vs_WT' = log2(MexKO/MexWT))
```

```
#Notice a dramatic anti-correlation log2FC TE vs. log2FC RNA
#Use linear regression to remove the anticorrelation and extract genes with highest residuals
reg_high_confidence_rachel <-
    high_confidence_rachel %>%
    melt(measure.vars = c('MexKO','MexWT','log2FC_KO_vs_WT'),
        variable.name = 'genotype') %>%
    dcast(gene_id + gene_name + cell_type ~ variable + genotype) %>%
    group_by(cell_type) %>%
    nest() %>%
    mutate('data' = map(data, function(df) {
      df %>%
        filter_at(.vars = vars(contains("log2FC")), .vars_predicate = all_vars(is.finite(.))) %>%
        mutate('reg_log2FC_TE' = predict(lm(translation_efficiency_log2FC_KO_vs_WT ~
                                    rnaseq_tpm_log2FC_KO_vs_WT, data = .))) %>%
        mutate('corrected_log2FC_TE' = translation_efficiency_log2FC_KO_vs_WT - reg_log2FC_TE)
    })) %>%
    unnest()

reg_high_confidence_rachel_data_subset <- filter(reg_high_confidence_rachel,
rnaseq_tpm_log2FC_KO_vs_WT < 2.5 & rnaseq_tpm_log2FC_KO_vs_WT > -2.5)
   lapply(levels(factor(reg_high_confidence_rachel_data_subset$cell_type)),
        function(x) {
          my_plot <- ggplot(data = reg_high_confidence_rachel_data_subset %>%
                      ungroup() %>%
                      filter(complete.cases(.) & cell_type == x) %>%
                      mutate('identity' = str_detect(.$gene_name, 'Olfr')) %>%
                      mutate('identity'= ifelse(.$identity, 'Olfr', 'Non-Olfr')) %>%
                      mutate('identity' = fct_relevel(identity,'Olfr', 'Non-Olfr')),
                    aes(x = rnaseq_tpm_log2FC_KO_vs_WT, y = corrected_log2FC_TE)) +
          geom_point(aes(color = identity), alpha = 0.3) +
          scale_color_manual(values=c("#FF0000","#656565")) +
          geom_smooth(method = 'lm', formula = y~x, se = T, fill = 'blue') +
          theme_classic() +
                xlab('Log2 Fold Change RNAseq TPM KO/WT') + ylab('Corrected Log2 Fold Change
          Translation Efficiency KO/WT') +
          ggtitle(label = paste0(x, '-Sorted Cells'))

          ggMarginal(my_plot, type = 'density', margins = 'both', groupFill = T)
        })

##############################################################################################
##############################################################################################
#########################Figure4F:
##############################################################################################
##############################################################################################
#subset first half of dataframe which is all Ngn cell type
Ngn_data <- reg_high_confidence_rachel[1:11987,]


ggplot(data=Ngn_data %>%
          mutate('identity' = str_detect(.$gene_name, 'Olfr')) %>%
          mutate('identity'= ifelse(.$identity, 'Olfr', 'Non-Olfr')) %>%
          mutate('identity' = fct_relevel(identity,'Olfr', 'Non-Olfr')),
        aes(x = identity, y = corrected_log2FC_TE, fill = identity)) +
    geom_violin() +
```

```
    scale_fill_manual(values=c("#FF0000", "#656565")) +
    geom_boxplot(fill = 'white', width = 0.1,outlier.shape = NA) +
    theme_classic() +
    geom_hline(yintercept = 0, linetype='dashed') +
    xlab('') + ylab('Corrected Log2FC Translation Efficiency KO/WT') +
    facet_grid(rows = vars(cell_type))
#get the Olfr identity on this Ngn_data df
Ngn_data <- Ngn_data %>%
    mutate('identity' = str_detect(.$gene_name, 'Olfr')) %>%
    mutate('identity'= ifelse(.$identity, 'Olfr', 'Non-Olfr')) %>%
    mutate('identity' = fct_relevel(identity,'Olfr', 'Non-Olfr'))

#statistics
t.test(corrected_log2FC_TE ~ identity, Ngn_data)

##############################################################################################
##############################################################################################
##########################Figure4H:
##############################################################################################
##############################################################################################

#ImageJ Analysis
#20X Images were taken of a 12uM coronal section of the entire olfactory epithelium, septum bone is
medial
#Next, about 260uM by 260uM (800x800) tiles were generated covering the entire epithelium. The
800x800 square was duplicated (all three channels) and saved as a tile.
#Tiles were opened individually in ImageJ and each Olfr positive cell was circled with the freehand
selection tool and added to the ROI manager
#The ROI manager was saved as a Roi.zip and used in the following imageJ code to measure intensity in
Olfr and Ngn channels, and determine the relative position from basal to apical (which is set by the user
with the DAPI channel)

#THIS SCRIPT RUNS IN IMAGEJ
   // Set Working Directory
   work_dir="/path/to/working/directory/"

   //Split Channels and name result
   orig_title=getTitle(); //get the title
   run("Split Channels");
   run("Duplicate...", " ");
   rename(orig_title);
   run("Put Behind [tab]");
   rename("DAPI");
   run("Put Behind [tab]");
   rename("NgnGFP");
   run("Put Behind [tab]");
   rename("Olfr");
   run("Put Behind [tab]");

   //Select the Thresholded Image
   image_ids=newArray(nImages)
   for (i=1; i<=nImages; i++){
     selectImage(i);
     image_ids[i-1]=getImageID();
   }
   Array.show(image_ids)
```

```
selectWindow("DAPI")
image_ids=Array.delete(image_ids, getImageID())
selectWindow("NgnGFP")
image_ids=Array.delete(image_ids, getImageID())
selectWindow("Olfr")
image_ids=Array.delete(image_ids, getImageID())
selectImage(image_ids[0])

//Get the Name from the Thresholded Image and Set Outputs
filename=getTitle(); //get its name
filename=split(filename, "(.tif)") //Thresholded image is of form C3-title-1.czi
filename=filename[0] //parse title

output_NgnGFP=work_dir+filename+"-NgnGFP.csv"
output_Olfr=work_dir+filename+"-Olfr.csv"
output_basal=work_dir+filename+"-Basal.txt"
output_apical=work_dir+filename+"-Apical.txt"
output_NgnApical=work_dir+filename+"-NgnApical.txt"

waitForUser("Parsed Name: " + filename + ". Click Ok to continue")

//Run the NgnGFP Measurement
count=roiManager("count");
final_array=newArray(count);
for(i=0; i<count;i++) {
  final_array[i] = i;
}
final_array = Array.slice(final_array, 0);
roiManager("Select", final_array);
selectWindow("NgnGFP")
run("Set Measurements...", "area mean centroid integrated redirect=None decimal=3");
run("Clear Results");
roiManager("measure")
saveAs("Results", output_NgnGFP);

//Run the Olfr Measurement
count=roiManager("count");
final_array=newArray(count);
for(i=0; i<count;i++) {
  final_array[i] = i;
}
final_array = Array.slice(final_array, 0);
roiManager("Select", final_array);
selectWindow("Olfr")
run("Set Measurements...", "area mean centroid integrated redirect=None decimal=3");
run("Clear Results");
roiManager("measure")
saveAs("Results", output_Olfr);

//Measure Apical and Basal Surfaces
selectWindow("DAPI")
roiManager("Show None");
setTool("freeline");
waitForUser("Delimit the Basal Surface, then click OK to save.")
saveAs("XY Coordinates",output_basal);
```

```
    waitForUser("Delimit the Apical Surface, then click OK to save.")
    saveAs("XY Coordinates",output_apical);

    //Delineate Ngn Apical surface
    selectWindow("NgnGFP")
    roiManager("Show None");
    setTool("freeline");
    waitForUser("Delimit the Apical Surface of Ngn layer, then click OK to save.")
    saveAs("XY Coordinates",output_NgnApical);

    //Cleanup
    for (i=1; i<=nImages; i++){
      selectImage(i);
      close();
    }
    roiManager("delete")

##########R ANALYSIS
    library(dplyr)
    library(tidyverse)
    library(magrittr)
    library(reshape2)
    setwd('/working/directory')

    compute_min_distance <- function(input_x, input_y, possible_x_vec, possible_y_vec){
      #' Computes minimum distance between centroid of cell (input_x, input_y) and an XY surface (x1, y1)
...(xn,yn)
      #'
      #' @param input_x (int) providing x coordinate for cell
      #' @param input_y (int) providing y coordinate for cell
      #' @param possible_x_vec (vec) providing x coordinates for points on surface (x1...xn)
      #' @param possible_y_vec (vec) providing y coordinates for points on surface (y1...yn)
      #' @return (int) specifying minimum euclidean distance to the surface.
      x_dist <- possible_x_vec - input_x
      y_dist <- possible_y_vec - input_y
      euc_dist <- sqrt(x_dist^2 + y_dist^2)
      return(min(euc_dist))
    }

    process_image <- function(path_ROI_out, path_basal_surface, path_apical_surface) {
      #' Processes Image from ImageJ output ROI measurements and TSV files specifying basal and apical
surfaces.
      #' @param path_ROI_out (str) specifying path to ROI out file.
      #' This file should be a CSV with (minimally) col 1 = ROI ID, a column entitled 'X' containing X
coordinate for ROI centroid and a column entitled 'Y' containing Y coordinate for ROI centroid
      #' Tested Workflow: Open Lhx2 IF -> Split Channels -> Use DAPI to Threshold -> Watershed ->
Analyze Particles (8um-100um, 0.5-1 circ)
      #' Combine all the cell ROIS -> single large ROI
      #' Create ROI that selects only cells of interest (ie. not sus or stem cells).Intersect this with all cells
ROI
      #' Split the resulting ROI -> single cell ROIs
      #' Measure centroid, mean, min, max, IntDen and RawIntDen
      #' Export Paths as XY surfaces
      #' @param path_basal_surface (str) specifying path to tsv (no header) where each row takes the form
(X_coord'\\t'Ycoord) describing basal surface
      #' @param path_apical_surface (str) as above, but for apical surface
```

```
    #' @return (data.frame) which adds basal and apical euclidian distances to whatever else is in the
provided ROI information
    cell_table <- read.csv(path_ROI_out)
    names(cell_table)[1]<-'ID'

    basal_surface <- read.table(path_basal_surface, header=F, col.names = c('X','Y'))
    apical_surface <- read.table(path_apical_surface, header=F, col.names = c('X','Y'))

    distances_df <- apply(cell_table, MARGIN = 1, FUN = function(my_cell) {
      basal_distance <- compute_min_distance(my_cell['X'], my_cell['Y'],
                                             basal_surface$X, basal_surface$Y)
      apical_distance <- compute_min_distance(my_cell['X'], my_cell['Y'],
                                              apical_surface$X, apical_surface$Y)
      return(data.frame('dist_basal_surface' = basal_distance,
                        'dist_apical_surface' = apical_distance))
    })
    return(bind_cols(cell_table, bind_rows(distances_df)))
  }

  #Read in the Data from HandDrawnResults
input_files <- data.frame('path' = list.files(path =
'/path/to/csv/files/from/imageJ/fluorescence/measurements/in/both/channels/for/circled/ROIs', full.names
= T)) %>%
    mutate('parsed_name' = str_extract(.$path, '(?<=/ROIs/).*(?=\\.)')) %>%
    tidyr::extract(col = 'parsed_name', into = c('channel','location','filetype'),
                   regex = '([0-9]+)_(.*)\\-((?:Apical|Basal|Ngn|Olfr).*)') %>%
    group_by(location) %>%
    dcast(location ~ filetype, value.var = 'path') %>%
    melt(measure.vars = c('Olfr','NgnGFP'), variable.name = 'Channel', value.name = 'Cells')

  parsed_data <- apply(X = input_files, MARGIN = 1, FUN = function(x) {
    process_image(path_ROI_out=x['Cells'], path_basal_surface = x['Basal'], path_apical_surface =
x['Apical'])
  })

  all_data_pilot <- mapply(FUN = function(df, location, channel) {
    my_df <- df %>% mutate('location' = rep(location, nrow(.)),
                           'channel' = rep(channel, nrow(.)))
    return(my_df)
  }, df = parsed_data, location = input_files$location, channel = input_files$Channel, SIMPLIFY = F)
%>%
    bind_rows() %>%
    mutate('identity' = str_extract(location,'KO|WT'),
           'size_OE' = dist_basal_surface + dist_apical_surface,
           'rel_position' = dist_basal_surface/size_OE)

  #Subset NgnGFP positive cells
  NgnGFP_cells <- all_data_pilot[c(#row_number_NgnStart:row_number_NgnEnd),]

  #Look at Intensities of Signal By Genotype (Ngn Data)
  ggplot(subset(NgnGFP_cells, identity %in% c("WT", "KO")) %>%
                      mutate(genotype = str_extract(identity, 'KO|WT'),
                             genotype = fct_relevel(genotype, 'WT')),
                    aes(x = genotype, y = log10(Mean))) +
               geom_point(alpha = 0.1, position = position_jitter(width = 0.01)) +
               geom_violin(aes(fill = genotype), alpha = 0.1) +
```

```
            geom_boxplot(fill = 'white', width = 0.1) +
            geom_text(data = . %>%
                        summarize('pval' = wilcox.test(x = log10(Mean)[genotype == 'WT'],
                                                       y = log10(Mean)[genotype == 'KO'])$p.value),
                      aes(label = paste0('p=',formatC(pval, digits = 2))),
                      x = Inf, y = -Inf, hjust = 1, vjust = 0) +
            theme_bw() +
            geom_text(data = . %>% group_by(genotype) %>%
                        summarize(n=n()),
                      aes(y = 3.7, label = paste0('n=',n, 'cells'))) +
            ggtitle('Avg NgnGFP Signal Intensity in Olfr Positive Cells')

##############################################################################################
##############################################################################################
#########################Figure4I:
##############################################################################################
##############################################################################################

#Subset Olfr measurements
Olfr_cells <- all_data_pilot[c(#row_number_Olfr_start:row_number_Olfr_End),]

#Look at the Ratios by Genotype
ggplot(data = Olfr_cells %>%
             mutate(size_OE = dist_basal_surface + dist_apical_surface,
                    rel_position = dist_basal_surface/size_OE,
                    genotype = str_extract(identity, 'KO|WT'),
                    genotype = fct_relevel(genotype, 'WT')),
           aes(x = genotype, y = rel_position)) +
          geom_point(alpha = 0.3, position = position_jitter(width = 0.1)) +
          geom_violin(aes(fill = genotype), alpha = 0.1) +
          geom_boxplot(fill = 'white', width = 0.1) +
          geom_text(data = . %>%
                      summarize('pval' = wilcox.test(x = rel_position[genotype == 'WT'],
                                                     y = rel_position[genotype == 'KO'])$p.value),
                    aes(label = paste0('p=',formatC(pval, digits = 2))),
                    x = Inf, y = -Inf, hjust = 1, vjust = 0) +
          theme_bw() +
          geom_text(data = . %>% group_by(genotype) %>%
                      summarize(n=n()),
                    aes(y = 1.05, label = paste0('n=',n, ' cells'))) +
          ggtitle('Relative Position')

##############################################################################################
##############################################################################################
#########################Figure5B:
##############################################################################################
##############################################################################################
#Same workflow as Figure 4H, replace NgnGFP with "Atf5", Cells with nuclear Atf5 staining were circled
instead of OR positive cells


##############################################################################################
##############################################################################################
#########################Figure5C:
##############################################################################################
##############################################################################################
```

```
#AddModuleScore is taken from Tirosh et al, Science (2016) doi: 10.1126/science.aad0501.

#This code builds off of code for scRNA-Seq that was described in Figure 4A
###These 318 genes are upregulated in Atf5 translating bulk RNA Seq compared to both Ngn GFP bulk RNA Seq and OMP GFP bulk RNA Seq, using DESeq2 to calculate differentially expressed genes#

Atf5_high <- read_tsv('/path/to/Atf5_high.txt',
                  col_names = c('GeneID'))

#Seurat's addmodulescore code
Mex.integrated.sct <- AddModuleScore(Mex.integrated.sct,
                               features = Atf5_high_list,
                               name="Atf5_high")
#make a dataframe to plot these data
Atf5_df_INP<-
  Mex.integrated.sct@meta.data$Atf5_high1%>%
  as.data.frame()

filter_INP <- Mex.integrated.sct@meta.data$new.cluster.ids %in% c('Immediate Neuronal Progenitor')
Atf5_df_INP$INP <-filter_INP
filter_KO <- Mex.integrated.sct@meta.data$geno %in% c('KO')
Atf5_df_INP$KO <-filter_KO
colnames(Atf5_df_INP)[colnames(Atf5_df_INP) == '.'] <- 'Atf5_Score'
Atf5_df_INP['INP'][Atf5_df_INP['INP'] == 'TRUE'] <- 'INP'
Atf5_df_INP['KO'][Atf5_df_INP['KO'] == 'TRUE'] <- 'KO'
Atf5_df_INP['KO'][Atf5_df_INP['KO'] == 'FALSE'] <- 'WT'

Atf5_df_INP_INP=Atf5_df_INP[Atf5_df_INP$INP %in%'INP', ]


Atf5_df_INP_INP$KO <- factor(x=Atf5_df_INP_INP$KO, levels = c("WT","KO"))
colnames(Atf5_df_INP_INP)[3] <- 'Geno'

ggplot(Atf5_df_INP_INP, aes(x=Geno, y=Atf5_Score)) +
          geom_violin(aes(fill = Geno), alpha = 1) +
          scale_fill_manual(values=c("#12908E", "#F98F45")) +
          geom_point(alpha = 0.2, position = position_jitter(width = 0.1)) +
          geom_boxplot(fill = 'white', width = 0.1) +
          geom_text(data = . %>%
                      summarize('pval' = wilcox.test(x = Atf5_Score[Geno == 'WT'],
                                                     y = Atf5_Score[Geno == 'KO'])$p.value),
                    aes(label = paste0('p=',formatC(pval, digits = 2))),
                    x = Inf, y = -Inf, hjust = 1, vjust = 0) +
          theme_bw() +
          geom_text(data = . %>% group_by(Geno) %>%
                      summarize(n=n()),
                    aes(y = 0.19, label = paste0('n=',n, ' Cells'))) +
          ggtitle('Atf5 Translating Score per INP cell')


##############################################################################################
##############################################################################################
#########################Figure5D:
##############################################################################################
##############################################################################################
```

```
#This is using the AddModuleScore call from Seurat on a different set of genes (AddModule originally described here: Tirosh et al, Science (2016) doi: 10.1126/science.aad0501.)
#Genes from Kahiapo, Jerome K. (2020). Atf5 Links Olfactory Receptor Induced Stress Response to Proper Neuronal Function [Doctoral dissertation, Columbia University]. Columbia University ProQuest Dissertations Publishing: number 28148970.

#these are differentially expressed in iOSNs (scRNA-Seq) between Atf5KO and Atf5WT + they are also Atf5 ChIP targets ("Atf5-dependent").
# (272 genes)

Atf5_dependent_genes <- read_tsv('/path/to/Atf5_dependent_genes.txt',
                         col_names = c('GeneID'))

Mex.integrated.sct <- AddModuleScore(Mex.integrated.sct,
                              features = Atf5_dependent_genes,
                              name="Atf5_dependent_genes")

YlOrBr <- c("#FFFFE5", "#FFF7BC", "#FEE391",
          "#FEC44F", "#FB9A29", "#EC7014",
          "#CC4C02", "#993404", "#662506")

FeaturePlot(object = Mex.integrated.sct, features = 'Atf5_dependent_genes1', split.by = 'geno',
keep.scale = NULL) & scale_colour_gradientn(colours = YlOrBr) & theme(legend.position = "bottom")


##############################################################################################
##############################################################################################
##########################Figure5E:
##############################################################################################
##############################################################################################

#Three separate dataframes were made from the Mex.integrated.sct Seurat object after running the code for figure 5D
Atf5_dependent_df<-
  Mex.integrated.sct@meta.data$Atf5_dependent_genes1%>%
  as.data.frame()

filter_INP <- Mex.integrated.sct@meta.data$new.cluster.ids %in% c('Immediate Neuronal Progenitor')
Atf5_dependent_df$INP <-filter_INP
filter_KO <- Mex.integrated.sct@meta.data$geno %in% c('KO')
Atf5_dependent_df$KO <-filter_KO
colnames(Atf5_dependent_df)[colnames(Atf5_dependent_df) == '.'] <- 'Atf5_dependent_Score'
Atf5_dependent_df['INP'][Atf5_dependent_df['INP'] == 'TRUE'] <- 'INP'
Atf5_dependent_df['KO'][Atf5_dependent_df['KO'] == 'TRUE'] <- 'KO'
Atf5_dependent_df['KO'][Atf5_dependent_df['KO'] == 'FALSE'] <- 'WT'

Atf5_dependent_df_INP=Atf5_dependent_df[Atf5_dependent_df$INP %in%'INP', ]
colnames(Atf5_dependent_df_INP)[3] <- 'Geno'
Atf5_dependent_df_INP$Geno <- factor(x=Atf5_dependent_df_INP$Geno, levels = c("WT","KO"))

ggplot(Atf5_dependent_df_INP, aes(x=Geno, y=Atf5_dependent_Score)) +
          geom_violin(aes(fill = Geno), alpha = 1) +
          scale_fill_manual(values=c("#12908E", "#F98F45")) +
          geom_point(alpha = 1, position = position_jitter(width = 0.1)) +
          geom_boxplot(fill = 'white', width = 0.075) +
```

```
            geom_text(data = . %>%
                        summarize('pval' = wilcox.test(x = Atf5_dependent_Score[Geno == 'WT'],
                                                       y = Atf5_dependent_Score[Geno == 'KO'])$p.value),
                      aes(label = paste0('p=',formatC(pval, digits = 2))),
                      x = Inf, y = -Inf, hjust = 1, vjust = 0) +
            theme_bw() +
            geom_text(data = . %>% group_by(Geno) %>%
                        summarize(n=n()),
                      aes(y = 0.78, label = paste0('n=',n, ' Cells'))) +
            ggtitle('Atf5 dependent Score per INP cell')

#The same code was repeated for iOSNs (named "Immature OSN" in Seurat object)
#The same code was repeated for mOSNs (named "Mature OSN" in Seurat object)

##############################################################################################
##############################################################################################
########################Figure6A:
##############################################################################################
##############################################################################################
#Tissue was zonally dissected into ventral zones (zone4-5) and dorsal zones (segment of zone 1) from
two biological replicates for each genotype at 4 weeks old
#scRNA-Seq data was processed as described for Figure 4A
#Identified mature OSN seurat clusters and focused analysis on these cells only

all_OR_data_mOSN <- ZonalMex.integrated.sct@meta.data %>%
            rownames_to_column(var = 'cell_id') %>%
            filter(seurat_clusters %in% c(37,11,0,16,2,20,1,3)) %>%
            pull(cell_id) %>%

          {ZonalMex.integrated.sct[['RNA']]@data[str_detect(rownames(ZonalMex.integrated.sct[['RNA']]@
          data), 'Olfr'), .]} %>%
            as.data.frame() %>%
            rownames_to_column(var = 'gene_name') %>%
            melt(id.vars = 'gene_name', variable.name = 'cell_id') %>%
            left_join(rownames_to_column(ZonalMex.integrated.sct@meta.data, var = 'cell_id'), by =
          'cell_id') #add the metadata

min_OR_enrichment <- 2
OR_ID_mOSN <- all_OR_data_mOSN %>%
            group_by(cell_id) %>%
            filter(sum(value) > 0) %>%
            summarize('chosen_OR' = gene_name[which.max(value)],
                      'chosen_OR_cts' = max(value),
                      'total_OR_cts' = sum(value),
                      'total_ORs_detected' = length(which(value>0)),
                      'frac_top_OR' = chosen_OR_cts/total_OR_cts,
                      'frac_second_OR' = sort(value, decreasing = T)[2]/total_OR_cts) %>%
            mutate('identity' = ifelse(frac_top_OR/frac_second_OR > min_OR_enrichment |
          total_ORs_detected == 1, 'Consensus Reached','No Consensus'))

#Add Metadata
OR_ID_withorig_mOSN <- ZonalMex.integrated.sct@meta.data %>%
  rownames_to_column(var = 'cell_id') %>%
  right_join(OR_ID_mOSN, by = 'cell_id')
```

```
#zonal_anno_2020_fromLisa.txt: these data were generously provided to the lab by Sunney Xie and assign Olfr genes into one of 5 anatomical zones
zonal_annotation <- read_tsv('/path/to/zonal_anno_2020_fromLisa.txt',
                         col_names = c('chosen_OR','zone'))

OR_ID_withorig_mOSN_zone <- left_join(OR_ID_withorig_mOSN, zonal_annotation, by = "chosen_OR")

#Consensus Reached filters to include only mOSNs that are expressing one OR predominantly
ggplot(data = OR_ID_withorig_mOSN_zone %>%
                 filter(identity == 'Consensus Reached' & !is.na(zone)),
               aes(x = orig.ident, fill = zone)) +
          geom_bar(color = 'black', position = 'fill') +
          theme_bw()

##############################################################################################
##############################################################################################
#########################Figure6B:
##############################################################################################
##############################################################################################
#Bulk RNA-Seq from three replicates of OMP-GFP sorted cells, either Mex3a cKO background or littermate controls. Processed with Takara SMARTer Stranded Total RNA-Seq Kit - Pico Input Mammalian.
#RNA-Seq is processed as Figure 1B

#dds is the output from DESeq2, make fpkm, tpm tables.
fpkm.table <- as.data.frame(fpkm(dds, robust = FALSE)) %>% rownames_to_column(var="GeneID")
fpkm <- fpkm.table %>% gather(condition,fpkm,-GeneID) %>%
separate(condition,sep="_",into=c("condition","replicate"))
fpkm.mean <- fpkm %>% group_by(condition,GeneID) %>% summarize(mean_fpkm=mean(fpkm))
fpkm.mean.table <- spread(fpkm.mean,condition,mean_fpkm)

tpm_table <- fpkm.table %>% mutate_at(.vars = vars(-GeneID), .funs = function(x) {x/sum(x) * 1e6})
write.table(tpm_table, file="MexWT_MexKO_OMPGFP_Clonetech_tpm_3Nov2021.txt",quote=FALSE)
tpm <- tpm_table %>% gather(condition,tpm,-GeneID) %>%
separate(condition,sep="_",into=c("condition","replicate"))
tpm.mean <- tpm %>% group_by(condition,GeneID) %>% summarize(mean_tpm = mean(tpm))
tpm.mean.table <- spread(tpm.mean,condition,mean_tpm)
#import zonal annotation
zonal_anno_fine_tb<-read_tsv("/path/to/zonal_anno_2020_fine.tsv", col_names =
c("name","chr","start","stop","strand","zone"))
zonal_anno_fine_tb<-zonal_anno_fine_tb %>% dplyr::select(name,zone)

tpm.mean.table.zone<-right_join(tpm.mean.table,zonal_anno_fine_tb,by=c("GeneID" = "name"))
tpm.mean.table.zone$zone<-factor(tpm.mean.table.zone$zone, levels=c("classI","1","2","3","4","5"))

#Fish Olfrs (Class I) and Zone1 Olfrs are physically in the same anatomical space, so they get the same color
zonal_colors_diagram<-c("#CC0000","#CC0000","#999900","#00994C","#3333FF","#CC00CC")

#Calculate Log2FC between Mex3a cKO and WT for each Olfr
tpm.mean.table.zone.log2.MexKO<-dplyr::select(tpm.mean.table.zone,GeneID,MexKO,MexWT,zone)
%>%
  mutate('KOvsWT_log2'=log((MexKO+0.01)/(MexWT+0.01)),2)

#plot
```

```
KOvsWT_log2<-ggplot((tpm.mean.table.zone.log2.MexKO), mapping = aes(zone,KOvsWT_log2,
color=zone, fill = zone)) + #ylim(-5,5)+
          geom_boxplot(alpha = 0.7, lwd =1.2 , outlier.shape=NA)+
          scale_color_manual(values=zonal_colors_diagram) + scale_fill_manual(values =
        zonal_colors_diagram) +
          geom_jitter(alpha = 0.7)+
          geom_hline(yintercept=0, color="black") +
          theme(legend.position="none",axis.text.y = element_text(colour= "black", size = 12),axis.text.x =
        element_text(colour= "black", size = 14), axis.title.x = element_blank(),axis.title.y =
        element_text(colour= "black", size = 14),plot.title = element_text(hjust = 0.5, size=20))+
          ggtitle("Mex3a cKO vs WT")
        KOvsWT_log2


################################################################################################
################################################################################################
#########################Figure6C:
################################################################################################
################################################################################################

#back to data from experiment shown in Figure 4A: PN12 Mex3a cKO and PN12 Mex3a WT littermate
controls
#this code builds on the code from 4A

VlnPlot(Mex.integrated.sct, features <- c('Ddit3'), cols = "#F98F45","#12908E", idents = c("Mature OSN",
"Immature OSN"),  group.by= "new.cluster.ids", split.by = "geno", pt.size = 0.1) +geom_boxplot(width=0.2,
position = position_dodge(0.9))

#In order to get the statistics for this Violin plot, run FindMarkers, as below
#The steps required to see how many cells in each of the original experiments
colnames(Mex.integrated.sct[[]])
Idents(Mex.integrated.sct) <- 'orig.ident'
head(Idents(Mex.integrated.sct))
##Levels: KO1  KO2  WT1  WT2
table(Mex.integrated.sct$orig.ident)
# KO1  KO2  WT1  WT2
#4865 5119  3196 4637

#this considers the two replicates for each geno the same genotype (KO or WT)
Mex.integrated.sct@meta.data$geno <- rep(c('KO','KO','WT','WT'), times = c(4865,5119,3196,4637))

Mex.integrated.sct@meta.data$final_ident <- ifelse(Mex.integrated.sct@meta.data$new.cluster.ids %in%
c("Immature OSN"), Mex.integrated.sct@meta.data$geno, 'other') #you only want to compare WT vs. KO
in the specified clusters. Set all other cells to "other"
Mex.integrated.sct <- SetIdent(Mex.integrated.sct, value = 'final_ident')
markers_KO_vs_WT_iOSN <- FindMarkers(Mex.integrated.sct, ident.1 = 'KO', ident.2 = 'WT')
markers_KO_vs_WT_iOSN<-tibble::rownames_to_column(markers_KO_vs_WT_iOSN, "GeneID")
write_csv(markers_KO_vs_WT_mOSN, "DiffExpress_KOvsWT_mOSN.csv")

Mex.integrated.sct@meta.data$final_ident <- ifelse(Mex.integrated.sct@meta.data$new.cluster.ids %in%
c("Mature OSN"), Mex.integrated.sct@meta.data$geno, 'other') #you only want to compare WT vs. KO in
the specified clusters. Set all other cells to "other"
Mex.integrated.sct <- SetIdent(Mex.integrated.sct, value = 'final_ident')
markers_KO_vs_WT_mOSN <- FindMarkers(Mex.integrated.sct, ident.1 = 'KO', ident.2 = 'WT')
markers_KO_vs_WT_mOSN<-tibble::rownames_to_column(markers_KO_vs_WT_mOSN, "GeneID")
write_csv(markers_KO_vs_WT_mOSN, "DiffExpress_KOvsWT_mOSN.csv")
```

```
#These CSVs contain Wilcox stats for Ddit3


##############################################################################################
##############################################################################################
#########################Figure6F:
##############################################################################################
##############################################################################################

#Glomeruli were scored as abnormal or normal/ and if abnormal, which of four phenotypes

##Considering each OR of the four we studied:
#####Mor23######
dat_M23_glom <- data.frame(
             "normal_no" = c(4, 7),
             "normal_yes" = c(136, 39),
             row.names = c("M23_WT", "M23_KO"),
             stringsAsFactors = FALSE
           )
           colnames(dat_M23_glom) <- c("AbnormalGlomeruli","NormalGlomeruli")

dat_M23_glom

mosaicplot(dat_M23_glom,
          main = "Mosaic plot",
          color = TRUE
)

chisq.test(dat_M23_glom)$expected

fisher.test(dat_M23_glom)
#Fisher's Exact Test for Count Data

#data:  dat_M23_glom
#p-value = 0.005553
#alternative hypothesis: true odds ratio is not equal to 1
#95 percent confidence interval:
#0.03381045 0.69178790
#sample estimates:
#odds ratio
#0.1659385

##########Mor28#########
dat_M28_glom <- data.frame(
             "normal_no" = c(4, 109),
             "normal_yes" = c(83, 55),
             row.names = c("M28_WT", "M28_KO"),
             stringsAsFactors = FALSE
           )
           colnames(dat_M28_glom) <- c("AbnormalGlomeruli","NormalGlomeruli")

dat_M28_glom

mosaicplot(dat_M28_glom,
          main = "Mosaic plot",
```

```
        color = TRUE
)

chisq.test(dat_M28_glom)$expected
#AbnormalGlomeruli NormalGlomeruli
#M28_WT         39.16733        47.83267
#M28_KO         73.83267        90.16733

chisq.test(dat_M28_glom)
#Pearson's Chi-squared test with Yates' continuity correction

#data:  dat_M28_glom
#X-squared = 85.416, df = 1, p-value < 2.2e-16

fisher.test(dat_M28_glom)
#Fisher's Exact Test for Count Data

#data:  dat_M28_glom
#p-value < 2.2e-16
#alternative hypothesis: true odds ratio is not equal to 1
#95 percent confidence interval:
# 0.006261759 0.070714708
#sample estimates:
#odds ratio
#0.02468745

#################M71###############
dat_M71_glom <- data.frame(
            "normal_no" = c(24, 34),
            "normal_yes" = c(103, 73),
            row.names = c("M71_WT", "M71_KO"),
            stringsAsFactors = FALSE
           )
           colnames(dat_M71_glom) <- c("AbnormalGlomeruli","NormalGlomeruli")

dat_M71_glom

mosaicplot(dat_M71_glom,
        main = "Mosaic plot",
        color = TRUE
)

chisq.test(dat_M71_glom)$expected
#AbnormalGlomeruli NormalGlomeruli
#M71_WT         31.47863        95.52137
#M71_KO         26.52137        80.47863

chisq.test(dat_M71_glom)
#Pearson's Chi-squared test with Yates' continuity correction

#data:  dat_M71_glom
#X-squared = 4.4984, df = 1, p-value = 0.03393
fisher.test(dat_M71_glom)

#Fisher's Exact Test for Count Data
```

```
#data:  dat_M71_glom
#p-value = 0.03285
#alternative hypothesis: true odds ratio is not equal to 1
#95 percent confidence interval:
# 0.2608611 0.9530061
#sample estimates:
#odds ratio
# 0.5017917
#######P2####################
dat_P2_glom <- data.frame(
              "normal_no" = c(18, 51),
              "normal_yes" = c(116, 17),
              row.names = c("P2_WT", "P2_KO"),
              stringsAsFactors = FALSE
            )
            colnames(dat_P2_glom) <- c("AbnormalGlomeruli","NormalGlomeruli")

dat_P2_glom

mosaicplot(dat_P2_glom,
          main = "Mosaic plot",
          color = TRUE
)
chisq.test(dat_P2_glom)$expected
#AbnormalGlomeruli NormalGlomeruli
#P2_WT         45.77228        88.22772
#P2_KO         23.22772        44.77228

chisq.test(dat_P2_glom)
#Pearson's Chi-squared test with Yates' continuity correction

#data:  dat_P2_glom
#X-squared = 73.313, df = 1, p-value < 2.2e-16

fisher.test(dat_P2_glom)
#Fisher's Exact Test for Count Data

#data:  dat_P2_glom
#p-value < 2.2e-16
#alternative hypothesis: true odds ratio is not equal to 1
#95 percent confidence interval:
# 0.02302448 0.11489430
#sample estimates:
#odds ratio
#0.05287508
#

################# Individual Phenotypes ###############
#Missing Glomeruli

dat_missing_glom <- data.frame(
              "missing_yes" = c(8, 36),
              "missing_no" = c(480, 349),
              row.names = c("WT", "KO"),
              stringsAsFactors = FALSE
            )
```

```
          colnames(dat_missing_glom) <- c("MissingGlomeruli","NormalGlomeruli")

dat_missing_glom

mosaicplot(dat_missing_glom,
         main = "Mosaic plot",
         color = TRUE
)
chisq.test(dat_missing_glom)$expected
chisq.test(dat_missing_glom)

#
#Pearson's Chi-squared test with Yates' continuity correction

#data:  dat_missing_glom
#X-squared = 25.152, df = 1, p-value = 5.299e-07

fisher.test(dat_missing_glom)
#Fisher's Exact Test for Count Data

#data:  dat_missing_glom
#p-value = 1.947e-07
#alternative hypothesis: true odds ratio is not equal to 1
#95 percent confidence interval:
#  0.06419697 0.35977371
#sample estimates:
#  odds ratio
#0.1618701

###Mispositioned#########

dat_mispositioned_glom <- data.frame(
             "mispositioned_yes" = c(3, 86),
             "mispositioned_no" = c(485, 299),
             row.names = c("WT", "KO"),
             stringsAsFactors = FALSE
           )

colnames(dat_mispositioned_glom) <- c("MispositionedGlomeruli","CorrectPositionGlomeruli")

dat_mispositioned_glom
mosaicplot(dat_mispositioned_glom,
         main = "Mosaic plot",
         color = TRUE
)
chisq.test(dat_mispositioned_glom)$expected
chisq.test(dat_mispositioned_glom)
#Pearson's Chi-squared test with Yates' continuity correction

#data:  dat_mispositioned_glom
#X-squared = 108.56, df = 1, p-value < 2.2e-16

fisher.test(dat_mispositioned_glom)
#Fisher's Exact Test for Count Data

#data:  dat_mispositioned_glom
```

```
#p-value < 2.2e-16
#alternative hypothesis: true odds ratio is not equal to 1
#95 percent confidence interval:
# 0.004333953 0.066162888
#sample estimates:
#odds ratio
#0.02157259

#####Reaching Axons#####

dat_reaching_glom <- data.frame(
  "reaching_yes" = c(10, 47),
  "reaching_no" = c(478, 338),
  row.names = c("WT", "KO"),
  stringsAsFactors = FALSE
)

colnames(dat_reaching_glom) <- c("ReachingAxons","IsolatedGlomeruli")

dat_reaching_glom
mosaicplot(dat_reaching_glom,
           main = "Mosaic plot",
           color = TRUE
)
chisq.test(dat_reaching_glom)$expected
chisq.test(dat_reaching_glom)
#Pearson's Chi-squared test with Yates' continuity correction

#data:  dat_reaching_glom
#X-squared = 34.746, df = 1, p-value = 3.757e-09
fisher.test(dat_reaching_glom)
#Fisher's Exact Test for Count Data

#data:  dat_reaching_glom
#p-value = 1.151e-09
#alternative hypothesis: true odds ratio is not equal to 1
#95 percent confidence interval:
#  0.06695655 0.30765325
#sample estimates:
#  odds ratio
#0.1507418

####Microglomeruli/ectopic glomeruli#####
dat_micro_glom <- data.frame(
              "microglom_yes" = c(31, 76),
              "microglom_no" = c(457, 309),
              row.names = c("WT", "KO"),
              stringsAsFactors = FALSE
            )

colnames(dat_micro_glom) <- c("Microglomeruli","No_microglomeruli")

dat_micro_glom
mosaicplot(dat_micro_glom,
                    main = "Mosaic plot",
                    color = TRUE
```

```
          )
chisq.test(dat_micro_glom)$expected
chisq.test(dat_micro_glom)
#Pearson's Chi-squared test with Yates' continuity correction

#data:  dat_micro_glom
#X-squared = 34.633, df = 1, p-value = 3.98e-09

fisher.test(dat_micro_glom)
#data:  dat_micro_glom
#p-value = 2.746e-09
#alternative hypothesis: true odds ratio is not equal to 1
#95 percent confidence interval:
#  0.1713574 0.4363570
#sample estimates:
#  odds ratio
#0.2762079

####################################SUPPLEMENTARY DATA#########################

#################################################################################################
#################################################################################################
########################SUPPLEMENTARY FigureS1A:
#################################################################################################
#################################################################################################

#!/usr/bin/env python
# coding: utf-8


####input exon and inton, merge two dataframe
import pandas as pd
import numpy as np

df1 = pd.read_csv("intron_5cell.txt",sep='\t')
df2 = pd.read_csv("exon_5cell.txt",sep='\t')


def calculate_fpkm_for_samples(df, length_col, raw_count_cols):
    for raw_count_col in raw_count_cols:
        # Calculate total mapped reads for the sample
        total_mapped_reads = df[raw_count_col].sum()
        # Create the FPKM column name for the sample
        fpkm_col_name = f'FPKM_{raw_count_col.split("_")[0]}'
        # Calculate FPKM
        df[raw_count_col] = (df[raw_count_col] * 1e9) / (df[length_col] * total_mapped_reads)
    return df


raw_count_columns_df1 = df1.iloc[:,-12:]
# Apply the function to the DataFrame
df_FPKM_intron = calculate_fpkm_for_samples(df1, 'Length_intron', raw_count_columns_df1)

raw_count_columns_df2 = df2.iloc[:,-12:]
# Apply the function to the DataFrame
df_FPKM_exon = calculate_fpkm_for_samples(df2, 'Length_exon', raw_count_columns_df2)
```

```
###merge exon and intron, subset OR gene, plot correalation
df3 = df_FPKM_exon.merge(df_FPKM_intron,on='Geneid')

def calculate_ratios(df, prefix):
    # Identify columns with the specified prefix
    columns = [col for col in df.columns if col.startswith(prefix)]
    # Calculate ratio for each pair of columns
    df[f'{prefix}'] = df[f'{prefix}_intron'] / df[f'{prefix}_exon']


calculate_ratios(df3, 'MashNgn_rep1')
calculate_ratios(df3, 'MashNgn_rep2')
calculate_ratios(df3, 'Ngn_rep1')
calculate_ratios(df3, 'Ngn_rep2')
calculate_ratios(df3, 'Atf5_rep2')
calculate_ratios(df3, 'Atf5_rep3')
calculate_ratios(df3, 'OMP_rep1')
calculate_ratios(df3, 'OMP_rep2')


f2  = open("OR_id.txt")
lines2 = f2.readlines()
OR_id = []
for line in lines2:
    words = line.strip().split("\t")
    OR_id.append(words[0])
OR_df3 = df3[df3['Geneid'].isin(OR_id)]
df4 =pd.concat([OR_df3.iloc[:,0], OR_df3.iloc[:,-8:]],axis=1)


###take average of Rep1 and Rep2
def calculate_average(df, col1, col2):
    # Remove the last 5 characters from col1 to create the new column name
    average_col_name = col1[:-5] if len(col1) > 5 else 'average'
    df[average_col_name] = df[[col1, col2]].mean(axis=1)
    return df


calculate_average(df4,'MashNgn_rep1', 'MashNgn_rep2')
calculate_average(df4,'Ngn_rep1', 'Ngn_rep2')
calculate_average(df4,'Atf5_rep3', 'Atf5_rep2')
calculate_average(df4,'OMP_rep1', 'OMP_rep2')


##only take average columns and OR has expression in each cell type
df4 =pd.concat([df4.iloc[:,0], df4.iloc[:,-4:]],axis=1)
filtered_df = df4[~df4.isin([0, np.nan, np.inf]).any(axis=1)]
df_long = pd.melt(filtered_df, id_vars=['Geneid'], var_name='CellType', value_name='intron/exon')

import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages

plt.figure(figsize=(8, 8))
```

```
sns.boxplot(x="CellType", y="intron/exon", data=df_long)
plt.ylim(0,5)
plt.savefig("intron_exon_ratio.pdf", format="pdf", bbox_inches="tight")
plt.show()
plt.close()

################################################################################################
################################################################################################
#########################SUPPLEMENTARY FigureS1B:
################################################################################################
################################################################################################

##dds was generated as output using DESeq2 as described in greater detail in Fig1B
# generate normalized counts for QC
# Normalizing transformation of count data and export normalized counts (normalized counts per gene)
rld <- rlog(dds, blind =T)
rlogMat <- assay(rld)
rlog_rna <- as.data.frame(rlogMat)

#### QC
##PCA analysis
plotPCA(rld, intgroup=c("condition")) #uses top  500 diff expressed genes by default

################################################################################################
################################################################################################
#########################SUPPLEMENTARY FigureS1C:
################################################################################################
################################################################################################
#OR positive cells were circled by hand in FIJI and measurements were calculated for each cell
#Area   Mean    Min      Max      X          Y          IntDen   RawIntDen

#Read in the Data from FIJI processing
Files <- list.files(pattern = "*OLFR.csv")
CSVlist <- lapply(Files, read.csv)
names(CSVlist) <- Files
AllDat <- bind_rows(CSVlist, .id = "Origin")
AllDat2 <- AllDat %>% separate(Origin,sep="_",into=c("File","Genotype"))

#remove gg8tTA; OMPtTA; Mex3a Tg genotype
AllDat_OMP <- AllDat2[AllDat2$Genotype!="TMexF2gg8omp",]

ggplot(data = AllDat_OMP %>%
                 mutate(Genotype = fct_relevel(Genotype, 'Control')),
               aes(x = Genotype, y = log10(Mean))) +
            geom_violin(aes(fill = Genotype), alpha = 1) +
            geom_point(alpha = 1, position = position_jitter(width = 0.1)) +
            scale_fill_manual(values=c("#004488", "#D55E00")) +
            geom_boxplot(fill = 'white', width = 0.1) +
            geom_text(data = . %>%
                      summarize('pval' = wilcox.test(x = log10(Mean)[Genotype == 'Control'],
                                                     y = log10(Mean)[Genotype == 'TMexF2omp'])$p.value),
                      aes(label = paste0('p=',formatC(pval, digits = 2))),
                      x = Inf, y = -Inf, hjust = 1, vjust = 0) +
            theme_bw() +
            geom_text(data = . %>% group_by(Genotype) %>%
                      summarize(n=n()),
```

```
            aes(y = 4, label = paste0('n=',n, 'cells'))) +
    ggtitle('Mean OR intensity per cell')


################################################################################
################################################################################
########################SUPPLEMENTARY FigureS1D:
################################################################################
################################################################################
```

#Code as for Figure 1E, except comparing gg8tTA; OMPtTA; Mex3a Tg to gg8tTA; OMPtTA; GFP Tg (added the gg8tTA driver here)

```
################################################################################
################################################################################
########################SUPPLEMENTARY FigureS1E:
################################################################################
################################################################################
```

#Code as for Figure 1F, except comparing gg8tTA; OMPtTA; Mex3a Tg to gg8tTA; OMPtTA; GFP Tg (added the gg8tTA driver here)

```
################################################################################
################################################################################
########################SUPPLEMENTARY FigureS2C:
################################################################################
################################################################################
##This code continues directly from code described in Fig 2B and 2F##

enrichment_wt_vs_KO_WT4mg_subset <- select(enrichment_wt_vs_KO,
unique_id,Gene_names,mean_enrichment_WT_4mg,mean_enrichment_KO)
enrichment_wt_vs_KO_WT4mg_subset <- enrichment_wt_vs_KO_WT4mg_subset %>%
mutate(Log2FCWTvsKO = mean_enrichment_WT_4mg - mean_enrichment_KO)

ggplot(data = enrichment_wt_vs_KO_WT4mg_subset,
            aes(x = mean_enrichment_WT_4mg, y = Log2FCWTvsKO)) +
    geom_point(alpha = 0.1)+
    theme_bw() +
    geom_point(data = . %>% filter(str_detect(Gene_names, 'Mex3a')),
            aes(label = Gene_names), color = 'red') +
    geom_point(data = . %>% filter(str_detect(Gene_names, 'Pabpc4')),
            aes(label = Gene_names), color = 'blue') +
    geom_point(data = . %>% filter(str_detect(Gene_names, 'Pabpc1')),
            aes(label = Gene_names), color = 'blue') +
    geom_point(data = . %>% filter(str_detect(Gene_names, 'Fmr1')),
            aes(label = Gene_names), color = 'blue') +
    geom_point(data = . %>% filter(str_detect(Gene_names, 'Cnot1')),
            aes(label = Gene_names), color = 'blue') +
    geom_point(data = . %>% filter(str_detect(Gene_names, 'Upf1')),
            aes(label = Gene_names), color = 'blue') +
    geom_point(data = . %>% filter(str_detect(Gene_names, 'Pum1')),
            aes(label = Gene_names), color = 'blue') +
    geom_point(data = . %>% filter(str_detect(Gene_names, 'Khsrp')),
            aes(label = Gene_names), color = 'blue') +
    geom_point(data = . %>% filter(str_detect(Gene_names, 'Serbp1')),
            aes(label = Gene_names), color = 'blue') +
```

```
        geom_text(data = . %>% filter(str_detect(Gene_names, 'Mex3a')),
                  aes(label = Gene_names), color = 'red',  check_overlap = TRUE) +
        geom_text(data = . %>% filter(str_detect(Gene_names, 'Pabpc4')),
                  aes(label = Gene_names), color = 'blue', check_overlap = TRUE) +
        geom_text(data = . %>% filter(str_detect(Gene_names, 'Pabpc1')),
                  aes(label = Gene_names), color = 'blue', check_overlap = TRUE) +
        geom_text(data = . %>% filter(str_detect(Gene_names, 'Fmr1')),
                  aes(label = Gene_names), color = 'blue', check_overlap = TRUE) +
        geom_text(data = . %>% filter(str_detect(Gene_names, 'Cnot1')),
                  aes(label = Gene_names), color = 'blue', check_overlap = TRUE) +
        geom_text(data = . %>% filter(str_detect(Gene_names, 'Upf1')),
                  aes(label = Gene_names), color = 'blue',check_overlap = TRUE) +
        geom_text(data = . %>% filter(str_detect(Gene_names, 'Pum1')),
                  aes(label = Gene_names), color = 'blue', check_overlap = TRUE) +
        geom_text(data = . %>% filter(str_detect(Gene_names, 'Khsrp')),
                  aes(label = Gene_names), color = 'blue', check_overlap = TRUE) +
        geom_text(data = . %>% filter(str_detect(Gene_names, 'Serbp1')),
                  aes(label = Gene_names), color = 'blue', check_overlap = TRUE) +
        ylim(-4,4) +
        xlim(-4,4) +
        geom_hline(yintercept=0.5,linetype=2) +
        geom_vline(xintercept=0.5,linetype=2)

#######Mex3a Tg panel
enrichment_wt_vs_KO_TMexF2_subset <- select(enrichment_wt_vs_KO,
unique_id,Gene_names,mean_enrichment_TMexF2,mean_enrichment_KO)
enrichment_wt_vs_KO_TMexF2_subset <- enrichment_wt_vs_KO_TMexF2_subset %>%
mutate(Log2FC_TMexF2_vsKO = mean_enrichment_TMexF2 - mean_enrichment_KO)


ggplot(data = enrichment_wt_vs_KO_TMexF2_subset,
           aes(x = mean_enrichment_TMexF2, y = Log2FC_TMexF2_vsKO)) +
        geom_point(alpha = 0.1)+
        theme_bw() +
        geom_point(data = . %>% filter(str_detect(Gene_names, 'Mex3a')),
                   aes(label = Gene_names), color = 'red') +
        geom_point(data = . %>% filter(str_detect(Gene_names, 'Pabpc4')),
                   aes(label = Gene_names), color = 'blue') +
        geom_point(data = . %>% filter(str_detect(Gene_names, 'Pabpc1')),
                   aes(label = Gene_names), color = 'blue') +
        geom_point(data = . %>% filter(str_detect(Gene_names, 'Fmr1')),
                   aes(label = Gene_names), color = 'blue') +
        geom_point(data = . %>% filter(str_detect(Gene_names, 'Cnot1')),
                   aes(label = Gene_names), color = 'blue') +
        geom_point(data = . %>% filter(str_detect(Gene_names, 'Upf1')),
                   aes(label = Gene_names), color = 'blue') +
        geom_point(data = . %>% filter(str_detect(Gene_names, 'Pum1')),
                   aes(label = Gene_names), color = 'blue') +
        geom_point(data = . %>% filter(str_detect(Gene_names, 'Khsrp')),
                   aes(label = Gene_names), color = 'blue') +
        geom_point(data = . %>% filter(str_detect(Gene_names, 'Serbp1')),
                   aes(label = Gene_names), color = 'blue') +
        geom_text(data = . %>% filter(str_detect(Gene_names, 'Mex3a')),
                  aes(label = Gene_names), color = 'red', check_overlap = TRUE) +
        geom_text(data = . %>% filter(str_detect(Gene_names, 'Pabpc4')),
                  aes(label = Gene_names), color = 'blue', check_overlap = TRUE) +
```

```
        geom_text(data = . %>% filter(str_detect(Gene_names, 'Pabpc1')),
                  aes(label = Gene_names), color = 'blue', check_overlap = TRUE) +
        geom_text(data = . %>% filter(str_detect(Gene_names, 'Fmr1')),
                  aes(label = Gene_names), color = 'blue', check_overlap = TRUE) +
        geom_text(data = . %>% filter(str_detect(Gene_names, 'Cnot1')),
                  aes(label = Gene_names), color = 'blue', check_overlap = TRUE) +
        geom_text(data = . %>% filter(str_detect(Gene_names, 'Upf1')),
                  aes(label = Gene_names), color = 'blue', check_overlap = TRUE) +
        geom_text(data = . %>% filter(str_detect(Gene_names, 'Pum1')),
                  aes(label = Gene_names), color = 'blue', check_overlap = TRUE) +
        geom_text(data = . %>% filter(str_detect(Gene_names, 'Khsrp')),
                  aes(label = Gene_names), color = 'blue', check_overlap = TRUE) +
        geom_text(data = . %>% filter(str_detect(Gene_names, 'Serbp1')),
                  aes(label = Gene_names), color = 'blue', check_overlap = TRUE) +
        ylim(-4,4) +
        xlim(-4,4) +
        geom_hline(yintercept=0.5,linetype=2) +
        geom_vline(xintercept=0.5,linetype=2)

#######Mutant Mex3a Tg panel

enrichment_wt_vs_KO_TMexF6_subset <- select(enrichment_wt_vs_KO,
unique_id,Gene_names,mean_enrichment_TMexF6,mean_enrichment_KO)
enrichment_wt_vs_KO_TMexF6_subset <- enrichment_wt_vs_KO_TMexF6_subset %>%
mutate(Log2FC_TMexF6_vsKO = mean_enrichment_TMexF6 - mean_enrichment_KO)
ggplot(data = enrichment_wt_vs_KO_TMexF6_subset,
           aes(x = mean_enrichment_TMexF6, y = Log2FC_TMexF6_vsKO)) +
        geom_point(alpha = 0.1)+
        theme_bw() +
        geom_point(data = . %>% filter(str_detect(Gene_names, 'Mex3a')),
                   aes(label = Gene_names), color = 'red') +
        geom_point(data = . %>% filter(str_detect(Gene_names, 'Pabpc4')),
                   aes(label = Gene_names), color = 'blue') +
        geom_point(data = . %>% filter(str_detect(Gene_names, 'Pabpc1')),
                   aes(label = Gene_names), color = 'blue') +
        geom_point(data = . %>% filter(str_detect(Gene_names, 'Fmr1')),
                   aes(label = Gene_names), color = 'blue') +
        geom_point(data = . %>% filter(str_detect(Gene_names, 'Cnot1')),
                   aes(label = Gene_names), color = 'blue') +
        geom_point(data = . %>% filter(str_detect(Gene_names, 'Upf1')),
                   aes(label = Gene_names), color = 'blue') +
        geom_point(data = . %>% filter(str_detect(Gene_names, 'Pum1')),
                   aes(label = Gene_names), color = 'blue') +
        geom_point(data = . %>% filter(str_detect(Gene_names, 'Khsrp')),
                   aes(label = Gene_names), color = 'blue') +
        geom_point(data = . %>% filter(str_detect(Gene_names, 'Serbp1')),
                   aes(label = Gene_names), color = 'blue') +
        geom_text(data = . %>% filter(str_detect(Gene_names, 'Mex3a')),
                  aes(label = Gene_names), color = 'red', check_overlap = TRUE) +
        geom_text(data = . %>% filter(str_detect(Gene_names, 'Pabpc4')),
                  aes(label = Gene_names), color = 'blue', check_overlap = TRUE) +
        geom_text(data = . %>% filter(str_detect(Gene_names, 'Pabpc1')),
                  aes(label = Gene_names), color = 'blue', check_overlap = TRUE) +
        geom_text(data = . %>% filter(str_detect(Gene_names, 'Fmr1')),
                  aes(label = Gene_names), color = 'blue', check_overlap = TRUE) +
        geom_text(data = . %>% filter(str_detect(Gene_names, 'Cnot1')),
```

```
        aes(label = Gene_names), color = 'blue', check_overlap = TRUE) +
    geom_text(data = . %>% filter(str_detect(Gene_names, 'Upf1')),
        aes(label = Gene_names), color = 'blue', check_overlap = TRUE) +
    geom_text(data = . %>% filter(str_detect(Gene_names, 'Pum1')),
        aes(label = Gene_names), color = 'blue', check_overlap = TRUE) +
    geom_text(data = . %>% filter(str_detect(Gene_names, 'Khsrp')),
        aes(label = Gene_names), color = 'blue', check_overlap = TRUE) +
    geom_text(data = . %>% filter(str_detect(Gene_names, 'Serbp1')),
        aes(label = Gene_names), color = 'blue',check_overlap = TRUE) +
    ylim(-4,4) +
    xlim(-4,4) +
    geom_hline(yintercept=0.5,linetype=2) +
    geom_vline(xintercept=0.5,linetype=2)


################################################################################
################################################################################
#########################SUPPLEMENTARY FigureS2D:
################################################################################
################################################################################
#data were generated by MaxQuant analysis of two independent Mass Spec runs (two biological
replicates), using human proteome as reference proteome
#Column headers were changed to reflect the experimental name before inputting into R for further
normalization and plotting

library(dplyr)
library(tidyverse)
library(magrittr)
library(reshape2)
library(tidyr)

setwd('/media/storageE/rachel/MassSpec_2024/HEK293T_proteomics/HEK293T_V5_IP_Ms2_assay_20
17_2018/')
raw_data <- read.csv('4Mar25_HEK_minus_plus_RNase_IPs.csv', stringsAsFactors = F)
raw_data <- raw_data %>% dplyr::select(-matches("empty"))
raw_data <- raw_data %>% dplyr::filter(!grepl("CON_", Protein.IDs))
raw_data <- raw_data %>% dplyr::filter(!grepl("REV_", Protein.IDs))


# Calculate the sum of each sample column (columns 11 to 28)
sum_row <- raw_data %>%
    dplyr::select(11:38) %>% # Select only sample columns
    summarise(across(everything(), sum, na.rm = TRUE)) %>% # Sum each column separately
    pivot_longer(
      cols = everything(),
      names_to = "Sample_Info",
      values_to = "Sum_of_library"
    ) %>%
    separate(
      col = Sample_Info,
      into = c("Replicate", "IP_number", "Sample", "Condition"),
      sep = "_",
      fill = "right"
    ) %>%
    mutate(Condition = replace_na(Condition, "minusRNase"))
```

```
# Reshape, separate, and replace missing Condition values
long_data_all <- raw_data %>%
            pivot_longer(
              cols = -c(1:10),  # Exclude columns 1 to 12 from reshaping
              names_to = "Sample_Info",  # Temporary column holding sample names
              values_to = "Value"  # Measurement values
            ) %>%
            separate(
              col = Sample_Info,  # Column to separate
              into = c("Replicate", "IP_number", "Sample", "Condition"),  # New variables
              sep = "_",
              fill = "right"  # Fill missing values to the right with NA
            ) %>%
            mutate(
              Condition = replace_na(Condition, "minusRNase")  # Replace NAs with "minusRNase"
            ) %>%
            group_by(Protein.IDs)  # Group by Protein.IDs for analysis

#Join Sum_of_library based on all four separate columns
long_data_all <- long_data_all %>%
            left_join(sum_row, by = c("Replicate", "IP_number", "Sample", "Condition")) %>%
            mutate(Sum_of_library = as.numeric(Sum_of_library))

#Correct discrepancies in naming
long_data_all <- long_data_all %>%
            mutate(Sample = Sample %>%
                  str_replace_all("Mex3aMutRing", "Mex3aMutRING") %>%
                  str_replace_all("Mex3aRing", "Mex3aRING") %>%
                  str_replace_all("Mex3aRINGDomain", "Mex3aRING"))

# Add the new normalized column
long_data_all <- long_data_all %>%
            mutate(Normalized_Value = (Value / Sum_of_library) * 1000000)

# Create Pseudocount column
long_data_all <- long_data_all %>%
            mutate(
              Pseudocount = Normalized_Value + 1
 )

# Create Log2 column
long_data_all <- long_data_all %>%
  mutate(
    Log2 = log2(Pseudocount)
  )

#Calculate the average for Replicate 1 and Replicate 2 for each group
rep_average <- long_data_all %>%
  group_by(Protein.IDs, Sample, Condition, IP_number) %>%
  summarise(
    Rep_Average = mean(Log2, na.rm = TRUE), # Average between Rep1 and Rep2
    .groups = "drop"  # Prevent grouping in the result
  )

long_data_all <- long_data_all %>%
```

```
  left_join(rep_average, by = c("Protein.IDs", "Sample", "Condition", "IP_number"))

#Calculate the Control value for each Protein.IDs, Sample, and Condition
control_values <- long_data_all %>%
  filter(Sample == "Control") %>%  # Filter only for Control samples
  group_by(Protein.IDs, Condition) %>%  # Group by Protein.IDs and Condition
  summarise(Control_Rep_Avg = mean(Rep_Average, na.rm = TRUE), .groups = "drop")

# Join the Control_Rep_Avg to the original dataframe for all rows of the same Protein.IDs and Condition
long_data_all <- long_data_all %>%
  left_join(control_values, by = c("Protein.IDs", "Condition"))

#Calculate the Relative_to_Control column
long_data_all <- long_data_all %>%
  mutate(
    Relative_to_Control = Rep_Average - Control_Rep_Avg
  )

#Subset to Mex3aFull and Mex3aRING (no KH domains) samples
long_data_all_Full_RING <- long_data_all %>%
  filter(Sample %in% c("Mex3aFull", "Mex3aRING"))

#Planning to plot from the Rep_average column, which is redundant between Rep1 and Rep2. Removing
Rep2 rows so we only plot one dot for each Protein.IDs
long_data_all_Full_RING_rep1 <- long_data_all_Full_RING %>%
  filter(Replicate %in% c("Rep1"))

#only considering minusRNAse condition here
long_data_all_Full_RING_rep1_minusRNase <- long_data_all_Full_RING_rep1 %>%
  filter(Condition %in% c("minusRNase"))

#calculate difference between Mex3aFull and Mex3aRING samples
long_data_all_Full_RING_rep1_minusRNase <- long_data_all_Full_RING_rep1_minusRNase %>%
  group_by(Protein.IDs) %>%
  mutate(
    Rel_to_Mex3aRING = Rep_Average[Sample == "Mex3aFull"] - Rep_Average[Sample ==
"Mex3aRING"]
  ) %>%
  ungroup()


##now only keep the Mex3aFull data and plot relative to Mex3aRING

long_data_all_Full_RING_rep1_minusRNase_noRING <- long_data_all_Full_RING_rep1_minusRNase
%>%
  filter(Sample %in% c("Mex3aFull"))

#pre-identified proteins that were enriched in Mex3aFull relative to control

ribosomal_proteins <- c("MRPL3", "RPS7", "RPS13", "MRPS23", "MRPS26", "RPS15A", "RPS3A",
"MRPS35", "RPS17",
                        "MRPS9", "DAP3", "RPS15", "RPS18", "MRPL19")
rna_binding_proteins <- c("MEX3A", "HNRNPDL", "HNRNPAB", "PAIP2", "HNRNPA2B1", "HNRNPL",
"STAU2", "HNRNPA0",
                          "MOV10", "PABPC4", "YTHDC2", "EIF4G3", "MEPCE", "HNRNPD")
```

```
# Modify the df to include the new color mapping for Point_Label_Color
long_data_all_Full_RING_rep1_minusRNase_noRING <-
long_data_all_Full_RING_rep1_minusRNase_noRING %>%
  mutate(Point_Label_Color = case_when(
    Gene.names %in% ribosomal_proteins ~ "#E26928",              # Ribosomal proteins
    Gene.names %in% rna_binding_proteins ~ "blue",           # RNA binding proteins
    Relative_to_Control > 0.58 ~ "black",                  # Other proteins with Relative_to_Control > 0.58
    TRUE ~ "gray"                                    # Default to black for all other cases
  ))


ggplot(long_data_all_Full_RING_rep1_minusRNase_noRING,
           aes(x = Relative_to_Control, y = Rel_to_Mex3aRING)) +
             geom_point(aes(color = Point_Label_Color), alpha = 0.8) +  # Scatter points with color
             geom_text_repel(
               aes(label = ifelse(Relative_to_Control > 0.58, Gene.names, NA), color = Point_Label_Color),
               max.overlaps = Inf,
               size = 4,
               box.padding = 0.3,
               min.segment.length = 0
             ) +
             geom_hline(yintercept = 0, linetype = "dotted", color = "black") +  # Horizontal line
             geom_vline(xintercept = 0.58, linetype = "dotted", color = "black") +  # Vertical line
             labs(
               x = "Relative to Untransfected Control (Log2)",
               y = "Abundance compared to Mex3aRING (no KH domains) (Log2)",
               title = "Mex3a Interactors in HEK293T cells",
               color = "Condition"
             ) +
             scale_color_identity() +  # Apply pre-defined custom colors
             theme_minimal() +
             theme(text = element_text(size = 14))



################################################################################
################################################################################
########################SUPPLEMENTARY FigureS2E:
################################################################################
################################################################################

#This code continues from above, but starts with the long_data_all df generated at the beginning of code
for Supplementary Figure 2D

#Subset Mex3aFull condition
long_data_all_Mex3aFull <- long_data_all %>%
  filter(Sample == "Mex3aFull")

#Planning to plot from the Rep_average column, which is redundant between Rep1 and Rep2. Removing
Rep2 rows so we only plot one dot for each Protein.IDs
long_data_all_Mex3aFull_Rep1 <- long_data_all_Mex3aFull %>%
  filter(Replicate == "Rep1")

#calculate difference between RNase and no RNase
long_data_all_Mex3aFull_Rep1 <- long_data_all_Mex3aFull_Rep1 %>%
  group_by(Protein.IDs) %>%
```

```
  mutate(Relative_RNase = ifelse(Condition == "minusRNase",
                                 Rep_Average - Rep_Average[Condition == "plusRNase"],
                                 NA)) %>%
  ungroup()

#plotting how minusRNase compares to plusRNAse, remove plusRNase from df for plotting
long_data_all_Mex3aFull_Rep1_RNase <- long_data_all_Mex3aFull_Rep1 %>%
  filter(Condition == "minusRNase")

# List of ribosomal proteins and RNA binding proteins (preselected from those proteins that are enriched
in Mex3aFull relative to Control (>Log 0.58))
ribosomal_proteins <- c("MRPL3", "RPS7", "RPS13", "MRPS23", "MRPS26", "RPS15A", "RPS3A",
"MRPS35", "RPS17",
                        "MRPS9", "DAP3", "RPS15", "RPS18", "MRPL19")
rna_binding_proteins <- c("MEX3A", "HNRNPDL", "HNRNPAB", "PAIP2", "HNRNPA2B1", "HNRNPL",
"STAU2", "HNRNPA0",
                          "MOV10", "PABPC4", "YTHDC2", "EIF4G3", "MEPCE", "HNRNPD")

# Modify the df to include the new color mapping for Point_Label_Color
long_data_all_Mex3aFull_Rep1_RNase <- long_data_all_Mex3aFull_Rep1_RNase %>%
  mutate(Point_Label_Color = case_when(
    Gene.names %in% ribosomal_proteins ~ "#E26928",              # Ribosomal proteins
    Gene.names %in% rna_binding_proteins ~ "blue",           # RNA binding proteins
    Relative_to_Control > 0.58 ~ "black",                  # Other proteins with Relative_to_Control > 0.58
    TRUE ~ "gray"                                    # Default to black for all other cases
  ))

# Plot with points colored as per the Gene.names lists
ggplot(long_data_all_Mex3aFull_Rep1_RNase, aes(x = Relative_to_Control, y = Relative_RNase)) +
            geom_point(aes(color = Point_Label_Color), alpha = 0.8) +  # Scatter points with color
            geom_text_repel(
              aes(label = ifelse(Relative_to_Control > 0.58, Gene.names, NA), color = Point_Label_Color),
              max.overlaps = Inf,
              size = 4,
              box.padding = 0.3,
              min.segment.length = 0
            ) +
            geom_hline(yintercept = 0, linetype = "dotted", color = "black") +  # Horizontal line
            geom_vline(xintercept = 0.58, linetype = "dotted", color = "black") +  # Vertical line
            labs(
              x = "Relative to Untransfected Control (Log2)",
              y = "Sensitive to RNase (Log2)",
              title = "Mex3a Interactors in HEK293T cells",
              color = "Condition"
            ) +
            scale_color_identity() +
            theme_minimal() +
            theme(text = element_text(size = 14))


##############################################################################################
##############################################################################################
#########################SUPPLEMENTARY FigureS3C:
##############################################################################################
##############################################################################################
```

```
#data are taken from NIS analysis of the MEAN INTENSITY for the red channel in each image for each genotype

tetOMex_M71ISH_Mean_pearson_DF <- read.csv('TetOMex_M71_ISH_MEAN_PEARSON_DF.csv', stringsAsFactors = F)

M71ISH_colors_diagram2 <- c("#005AB5","#DC3220")

mean_plot_ISH <-ggplot((tetOMex_M71ISH_Mean_pearson_DF) %>%
                         mutate('Genotype' = fct_relevel(Genotype,'TM71 omp','TmexF2 TM71 omp')),
                       mapping = aes(Genotype,M71_ISH_Mean_Intensity, color=Genotype, fill = Genotype)) +
  geom_boxplot(alpha = 0.7, lwd =1.2 , outlier.shape=NA)+
  scale_color_manual(values=M71ISH_colors_diagram2) + scale_fill_manual(values = M71ISH_colors_diagram2) +
  geom_jitter(alpha = 1)+
  theme_bw() +
  theme(legend.position="none",axis.text.y = element_text(colour= "black"),axis.text.x = element_text(colour= "black", size = 8), axis.title.x = element_blank(),axis.title.y = element_text(colour= "black", size = 8),plot.title = element_text(hjust = 0.5, size=12))+
  ggtitle("Mean M71 ISH fluorescence intensity")
mean_plot_ISH

TM71omp_omp_mean_values <- tetOMex_M71ISH_Mean_pearson_DF[tetOMex_M71ISH_Mean_pearson_DF$Genotype =="TM71 omp",]
TM71omp_omp_mean_values <- TM71omp_omp_mean_values[-c(1:2)]


TM71_TMexF2_omp_mean_values <- tetOMex_M71ISH_Mean_pearson_DF[tetOMex_M71ISH_Mean_pearson_DF$Genotype =="TmexF2 TM71 omp",]
TM71_TMexF2_omp_mean_values <- TM71_TMexF2_omp_mean_values[-c(1:2)]
t.test(TM71omp_omp_mean_values,TM71_TMexF2_omp_mean_values)

##############################################################################################
##############################################################################################
##########################SUPPLEMENTARY FigureS3J:
##############################################################################################
##############################################################################################

#MA plot and violin plot generated as described in Fig 1E and 1F, comparing OMPtTA; Mex3a RING Mutant Transgene sorted mCherry+ cells to OMPtTA; GFP Tg sorted GFP+ cells

##############################################################################################
##############################################################################################
##########################SUPPLEMENTARY FigureS3L:
##############################################################################################
##############################################################################################

#this code continues from code elaborated in Figure 3E, starting with the same tetOMex_Mean_pearson_DF generated there.
#Pearson_Coefficient value calculated for each image using NIS software which determined colocalization between red and green channels

Pearson_colors_diagram2<-c("#56B4E9","#D55E00","#009E73")
```

```
coloc_plot_supplemental <- ggplot((tetOMex_Mean_pearson_DF)%>%
                    filter(Genotype %in% c('TM71 gg8omp','TP2 gg8omp', 'TGFP gg8omp')) %>%
                    mutate('Genotype' = fct_relevel(Genotype,'TM71 gg8omp','TP2 gg8omp', 'TGFP
          gg8omp')),
                  mapping = aes(Genotype,Pearson_Coefficient, color=Genotype, fill = Genotype)) +
            geom_boxplot(alpha = 0.7, lwd =1.2 , outlier.shape=NA)+
            scale_color_manual(values=Pearson_colors_diagram2) + scale_fill_manual(values =
          Pearson_colors_diagram2) +
            geom_jitter(alpha = 1)+
            theme_bw() +
            theme(legend.position="none",axis.text.y = element_text(colour= "black", size = 12),axis.text.x =
          element_text(colour= "black", size = 14), axis.title.x = element_blank(),axis.title.y =
          element_text(colour= "black", size = 14),plot.title = element_text(hjust = 0.5, size=20))+
            ggtitle("Pearson Colocalization Coefficient")
coloc_plot_supplemental

TM71_gg8omp_pearson_values <-tetOMex_Mean_pearson_DF[tetOMex_Mean_pearson_DF$Genotype
=="TM71 gg8omp",]
TM71_gg8omp_pearson_values <- TM71_gg8omp_pearson_values[-c(1:5,7,8)]

TP2_gg8omp_pearson_values <-tetOMex_Mean_pearson_DF[tetOMex_Mean_pearson_DF$Genotype
=="TP2 gg8omp",]
TP2_gg8omp_pearson_values <- TP2_gg8omp_pearson_values[-c(1:5,7,8)]

TGFP_gg8omp_pearson_values <-tetOMex_Mean_pearson_DF[tetOMex_Mean_pearson_DF$Genotype
=="TGFP gg8omp",]
TGFP_gg8omp_pearson_values <- TGFP_gg8omp_pearson_values[-c(1:5,7,8)]

t.test(TM71_gg8omp_pearson_values,TGFP_gg8omp_pearson_values)

#Welch Two Sample t-test

#data:  TM71_gg8omp_pearson_values and TGFP_gg8omp_pearson_values
#t = -12.093, df = 12.432, p-value = 3.065e-08
#alternative hypothesis: true difference in means is not equal to 0
#95 percent confidence interval:
#  -0.4677956 -0.3254290
#sample estimates:
#  mean of x mean of y
#0.0383699 0.4349822

t.test(TP2_gg8omp_pearson_values,TGFP_gg8omp_pearson_values)

#Welch Two Sample t-test

#data:  TP2_gg8omp_pearson_values and TGFP_gg8omp_pearson_values
#t = -7.7049, df = 6.0104, p-value = 0.0002484
#alternative hypothesis: true difference in means is not equal to 0
#95 percent confidence interval:
#  -0.3485924 -0.1806015
#sample estimates:
#  mean of x mean of y
#0.1703852 0.4349822
```

```
################################################################################################
################################################################################################
#########################SUPPLEMENTARY FigureS4A:
################################################################################################
################################################################################################
#This continues from code in Figure 4A (aligning scRNA Seq reads, importing Cell Ranger aligned data
into a Seurat object, normalizing etc)

FeaturePlot(Mex.integrated.sct, features = c('Ascl1','Neurod1','Crabp1','Gap43','Omp','Cyp2a5'), label = F)

################################################################################################
################################################################################################
#########################SUPPLEMENTARY FigureS4B:
################################################################################################
################################################################################################
#This continues from code in Figure 4A (aligning scRNA Seq reads, importing Cell Ranger aligned data
into a Seurat object, normalizing etc)

FeaturePlot(Mex.integrated.sct, features = c('Mex3a'), label = F)

################################################################################################
################################################################################################
#########################SUPPLEMENTARY FigureS4C:
################################################################################################
################################################################################################
#This continues from code in Figure 4A (aligning scRNA Seq reads, importing Cell Ranger aligned data
into a Seurat object, normalizing etc)

> Mex.integrated.sct@meta.data$geno <- rep(c('KO','KO','WT','WT'), times = c(4865,5119,3196,4637))
#cells in each library
> DefaultAssay(Mex.integrated.sct) <- "RNA"

four_color_palette <- c("#332288","#CC6677", "#882255", "#EE3377")
ggplot(data = Mex.integrated.sct@meta.data %>%
                filter(new.cluster.ids %in% c("Globose Basal Cell","Immediate Neuronal
        Progenitor","Immature OSN","Mature OSN")) %>%
                mutate('genotype' = fct_relevel(geno, 'WT', 'KO'),
                       'new.cluster.ids' = as.character(new.cluster.ids),
                       'new.cluster.ids' = fct_relevel(new.cluster.ids,'Mature OSN','Immature OSN','Immediate
        Neuronal Progenitor','Globose Basal Cell')),
              aes(x = geno, fill = new.cluster.ids)) +
          geom_bar(position = 'fill', color = 'black') +
          scale_fill_manual(values=four_color_palette) +
          theme_bw() + theme(legend.position = 'bottom')

################################################################################################
################################################################################################
#########################SUPPLEMENTARY FigureS4D:
################################################################################################
################################################################################################

#This figure uses the all_OR_data_neuronal_lineage df generated as described in Fig 4C


dat_ <-all_OR_data_neuronal_lineage %>%
```

```
  filter(new.cluster.ids %in% c("Globose Basal Cell","Immediate Neuronal Progenitor","Immature
OSN","Mature OSN")) %>%
  group_by(cell_id, new.cluster.ids, geno) %>%
  summarize('OR_scaled_expression' = sum(value),
            'Number_ORs_detected' = length(which(value > 1))) %>%
  ungroup() %>%
  mutate('geno' = fct_relevel(geno,'WT','KO'),
         'new.cluster.ids' = as.character(new.cluster.ids),
         'new.cluster.ids' = fct_relevel(new.cluster.ids,'Globose Basal Cell','Immediate Neuronal
Progenitor','Immature OSN','Mature OSN'),
         'OR_expression' = ifelse(Number_ORs_detected == 0, 'Zero',
ifelse(Number_ORs_detected == 1, 'One','Multiple')),
         'OR_expression' = fct_relevel(OR_expression,'One','Multiple','Zero')) %>%
  group_by(new.cluster.ids, geno, OR_expression) %>%
  summarize(n=n())


mycolors <- c("#004488", "#DDAA33", "#BB5566")
ggplot(data = dat_, aes(x = geno, y = n, fill=OR_expression)) +
         geom_col(position = 'fill', color = 'black') +
         scale_fill_manual(values=mycolors) +
         theme_bw() + theme(legend.position = 'bottom') +
         facet_wrap(facets = vars(new.cluster.ids), nrow = 1)


#F test to compare two variances
###Enter data for mOSN
dat_mOSN <- data.frame(
         "multiple_yes" = c(164, 288),
         "multiple_no" = c(1560, 2111),
         row.names = c("WT", "KO"),
          stringsAsFactors = FALSE
         )
colnames(dat_mOSN) <- c("Multiple_ORs","Zero_or_One")
dat_mOSN
  Multiple_ORs Zero_or_One
WT          164        1560
KO          288        2111
mosaicplot(dat_mOSN,
         main = "Mosaic plot",
         color = TRUE
)
chisq.test(dat_mOSN)$expected
  Multiple_ORs Zero_or_One
WT     189.0002        1535
KO     262.9998        2136
chisq.test(dat_mOSN)

         Pearson's Chi-squared test with Yates' continuity correction

data:  dat_mOSN
X-squared = 6.1304, df = 1, p-value = 0.01329

fisher.test(dat_mOSN)

         Fisher's Exact Test for Count Data
```

```
data:  dat_mOSN
p-value = 0.01153
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 0.6251853 0.9475538
sample estimates:
odds ratio
  0.770624

###Enter data for iOSN
dat_iOSN <- data.frame (
              "multiple_yes" = c(95, 253),
             "multiple_no" = c(922, 1035),
             row.names = c("WT", "KO"),
             stringsAsFactors = FALSE
            )
colnames(dat_iOSN) <- c("Multiple_ORs","Zero_or_One")
dat_iOSN
   Multiple_ORs Zero_or_One
WT           95         922
KO          253        1035
mosaicplot(dat_iOSN,
                     main = "Mosaic plot",
                     color = TRUE
           )
chisq.test(dat_iOSN)$expected
   Multiple_ORs Zero_or_One
WT     153.5427    863.4573
KO     194.4573   1093.5427
> chisq.test(dat_iOSN)

          Pearson's Chi-squared test with Yates' continuity correction

data:  dat_iOSN
X-squared = 46.249, df = 1, p-value = 1.041e-11

> fisher.test(dat_iOSN)

          Fisher's Exact Test for Count Data

#data:  dat_iOSN
#p-value = 3.222e-12
#alternative hypothesis: true odds ratio is not equal to 1
#95 percent confidence interval:
# 0.3240709 0.5452974
#sample estimates:
#odds ratio
# 0.4216632

#data:  dat_iOSN
#p-value = 3.222e-12
#alternative hypothesis: true odds ratio is not equal to 1
#95 percent confidence interval:
#   0.3240709 0.5452974
#sample estimates:
```

```
#   odds ratio
#0.4216632

###Enter data for INP
dat_INP <- data.frame (
              "multiple_yes" = c(61, 187),
              "multiple_no" = c(678, 869),
              row.names = c("WT", "KO"),
              stringsAsFactors = FALSE
            )
colnames(dat_INP) <- c("Multiple_ORs","Zero_or_One")
dat_INP
   Multiple_ORs Zero_or_One
WT           61         678
KO          187         869
mosaicplot(dat_INP,
           main = "Mosaic plot",
          color = TRUE
           )
chisq.test(dat_INP)$expected
   Multiple_ORs Zero_or_One
WT     102.1014    636.8986
KO     145.8986    910.1014
chisq.test(dat_INP)

           Pearson's Chi-squared test with Yates' continuity correction

data:  dat_INP
X-squared = 31.844, df = 1, p-value = 1.671e-08

fisher.test(dat_INP)
#Fisher's Exact Test for Count Data

#data:  dat_INP
#p-value = 6.028e-09
#alternative hypothesis: true odds ratio is not equal to 1
#95 percent confidence interval:
# 0.3026585 0.5718342
#sample estimates:
#odds ratio
# 0.4183142


#Enter data for GBC
dat_GBC <- data.frame (
              "multiple_yes" = c(3, 2),
              "multiple_no" = c(181, 235),
              row.names = c("WT", "KO"),
              stringsAsFactors = FALSE
            )

colnames(dat_GBC) <- c("Multiple_ORs","Zero_or_One")
dat_GBC
mosaicplot(dat_GBC,
                    main = "Mosaic plot",
                    color = TRUE
```

```
          )
chisq.test(dat_GBC)$expected
chisq.test(dat_GBC)
fisher.test(dat_GBC)

#Fisher's Exact Test for Count Data

#data:  dat_GBC
#p-value = 0.6573
#alternative hypothesis: true odds ratio is not equal to 1
#95 percent confidence interval:
#0.220344 23.503835
#sample estimates:
#odds ratio
#1.944379


##############################################################################################
##############################################################################################
#########################SUPPLEMENTARY FigureS4E:
##############################################################################################
##############################################################################################

#This figure uses the all_OR_data_neuronal_lineage df generated as described in Fig 4C

all_OR_data_neuronal_lineage$geno <- factor(x=all_OR_data_neuronal_lineage$geno, levels =
c("WT","KO"))
all_OR_data_neuronal_lineage$new.cluster.ids <-
factor(x=all_OR_data_neuronal_lineage$new.cluster.ids, levels = c('Globose Basal Cell','Immediate
Neuronal Progenitor','Immature OSN','Mature OSN'))
data = all_OR_data_neuronal_lineage %>% #I made a df that looks like the all_OR_data_INP above but
has the four cell types below
            filter(new.cluster.ids %in% c("Globose Basal Cell","Immediate Neuronal Progenitor","Immature
          OSN","Mature OSN")) %>%
            group_by(cell_id, new.cluster.ids, geno) %>%
            summarize('OR_scaled_expression' = sum(value),
                      'Number_ORs_detected' = length(which(value > 0))) %>%
            ungroup() %>%
            mutate('OR_expression' = ifelse(Number_ORs_detected == 0, 'Zero',
          ifelse(Number_ORs_detected == 1, 'One','Multiple'))) %>%
            group_by(new.cluster.ids, geno, OR_expression)


ggplot(data = data %>%
                 filter(OR_expression %in% c("Multiple")),
               aes(x = geno, y = Number_ORs_detected)) +
            geom_violin(aes(fill = geno), color = 'black') +
            scale_fill_manual(values=c("#12908E", "#F98F45")) +
            geom_boxplot(fill = 'white', width = 0.2, outlier.shape = NA) +
            stat_summary(fun.y=mean, geom="point", shape=20, size=2, color="blue", fill="blue") +
            facet_wrap(facets = vars(new.cluster.ids), nrow = 1) +
            theme_bw()

test_vals <- filter(data, OR_expression == "Multiple" & geno == 'KO' & new.cluster.ids == 'Immediate
Neuronal Progenitor')$Number_ORs_detected
```

```
control_vals <- filter(data, OR_expression == "Multiple" & geno == 'WT' & new.cluster.ids == 'Immediate Neuronal Progenitor')$Number_ORs_detected
wilcox.test(control_vals, test_vals)

#	Wilcoxon rank sum test with continuity correction

#data:  control_vals and test_vals
#W = 67558, p-value = 0.00002
#alternative hypothesis: true location shift is not equal to 0

test_vals <- filter(data, OR_expression == "Multiple" & geno == 'KO' & new.cluster.ids == 'Globose Basal Cell')$Number_ORs_detected
control_vals <- filter(data, OR_expression == "Multiple" & geno == 'WT' & new.cluster.ids == 'Globose Basal Cell')$Number_ORs_detected
wilcox.test(control_vals, test_vals)

#Wilcoxon rank sum test with continuity correction

#data:  control_vals and test_vals
#W = 1191.5, p-value = 0.8799
#alternative hypothesis: true location shift is not equal to 0

test_vals <- filter(data, OR_expression == "Multiple" & geno == 'KO' & new.cluster.ids == 'Immature OSN')$Number_ORs_detected
control_vals <- filter(data, OR_expression == "Multiple" & geno == 'WT' & new.cluster.ids == 'Immature OSN')$Number_ORs_detected
wilcox.test(control_vals, test_vals)

#Wilcoxon rank sum test with continuity correction

#data:  control_vals and test_vals
#W = 131900, p-value = 0.1884
#alternative hypothesis: true location shift is not equal to 0

test_vals <- filter(data, OR_expression == "Multiple" & geno == 'KO' & new.cluster.ids == 'Mature OSN')$Number_ORs_detected
control_vals <- filter(data, OR_expression == "Multiple" & geno == 'WT' & new.cluster.ids == 'Mature OSN')$Number_ORs_detected
wilcox.test(control_vals, test_vals)
#Wilcoxon rank sum test with continuity correction

#data:  control_vals and test_vals
#W = 527830, p-value = 4.726e-06
#alternative hypothesis: true location shift is not equal to 0

################################################################################
################################################################################
##########################SUPPLEMENTARY FigureS4F:
################################################################################
################################################################################

#code is the same as code for Fig4E, comparing the bulk RNA-Seq from sorted OMP-GFP+ cells in Mex3a cKO mice to sorted OMP-GFP+ cells in Mex3a WT mice.

################################################################################
################################################################################
```

##########SUPPLEMENTARY FigureS4G:
##########
##########

#code is the same as code for Fig4F, comparing the bulk RNA-Seq from sorted OMP-GFP+ cells in Mex3a cKO mice to sorted OMP-GFP+ cells in Mex3a WT mice.

##########
##########
##########SUPPLEMENTARY FigureS5A:
##########
##########

#Fuzznuc was used to search motif through the mm10 genome and 3'UTR of ORs: Rice P., Longden I. and Bleasby A. EMBOSS: The European Molecular Biology Open Software Suite. Trends in Genetics. 2000 16(6):276-277

###The same code was used for the C_elegans motif and the human Mex3c motif after identifying hits using Fuzznuc
####C_Elegans RNA recognition motif in 3'UTR of ORs
#we checked if there was any zonal bias (see Fig6) for RNA recognition motifs - there were none (data not shown).

```
zonal_data <- read.csv('zonal_anno_2020_fine.csv', stringsAsFactors = FALSE)
raw_data_CElegans <- read.delim('3UTR_Mex3a_Elegans_hitCount-2.txt', stringsAsFactors = FALSE)
raw_data_CElegans <- raw_data_CElegans %>%
  left_join(zonal_data, by = c("Olfr_name" = "Gene"))

# Replace NA in Zone with "undetermined"
raw_data_CElegans <- raw_data_CElegans %>%
  mutate(Zone = replace_na(Zone, "undetermined"))

# Full join to keep all Olfrs from zonal_data
raw_data_full <- full_join(zonal_data, raw_data_CElegans, by = c("Gene" = "Olfr_name"))

# Replace NA HitCounts with 0
raw_data_full <- raw_data_full %>%
  mutate(HitCount = ifelse(is.na(HitCount), 0, HitCount))

# Quantify what percentage of ORs has specific number of hits for RNA recognition motif in the 3'UTR
pie_data <- raw_data_full %>%
  group_by(HitCount) %>%
  summarise(Frequency = n()) %>%
  mutate(Percentage = Frequency / sum(Frequency) * 100)

#Plot the pie chart with sequential color mapped to HitCount
ggplot(pie_data, aes(x = "", y = Percentage, fill = HitCount)) +
        geom_bar(stat = "identity", width = 1, color = "black") +
        coord_polar(theta = "y") +
        labs(
          title = "Distribution of Hit Counts",
          fill = "HitCount"
        ) +
        theme_void() +
        theme(text = element_text(size = 14)) +
```

```
  scale_fill_gradient(low = "#FFFFE5", high = "#440154") +  # Viridis sequential gradient (light to
dark)
  geom_text(aes(label = paste0(round(Percentage, 1), "%")),
            position = position_stack(vjust = 0.5),
            color = "black", size = 4)

################################################################################
################################################################################
#########################SUPPLEMENTARY FigureS5B:
################################################################################
################################################################################

library(GenomicFeatures)
library(regioneR,lib="/media/summer/miao/packages/R_lib")
library("BSgenome.Hsapiens.UCSC.hg19",lib="/media/summer/miao/packages/R_lib")
library("txdbmaker")
library(regioneR,lib="/media/summer/miao/packages/R_lib")
library(XVector, lib.loc = "/media/summer/miao/packages/R_lib")
library(data.table)
library(parallel)

motif <- fread("hMEX3C_mm10.bed",sep='\t')
trascript_OR <- fread("OR_3UTR_3.bed",sep='\t')

#Pre-allocate vectors
chr <- motif$V1
start <- motif$V2
end <- motif$V3
strand <- motif$V4

# Create GRanges object
A <- GRanges(
  seqnames = chr,
  ranges = IRanges(start = start, end = end),
  strand = strand
)

B <- GRanges(
  seqnames = trascript_OR$V1,
  ranges = IRanges(start = trascript_OR$V2, end = trascript_OR$V3),
  strand = trascript_OR$V4
)

# Extract transcript regions
# Create TxDb object from UCSC for mouse genome (mm10)
txdb <- makeTxDbFromUCSC(genome = "mm10", tablename = "refGene")

# Extract transcript regions as a GRanges object
transcripts <- transcripts(txdb)

pt <- overlapPermTest(A = A, B = B, ntimes = 1000, universe = transcripts, ignore.strand =
FALSE,mc.cores = 16)

pdf("hMEX3C_enrich.pdf")
plot(pt)
dev.off()
```

```
################################################################################
################################################################################
#########################SUPPLEMENTARY FigureS5D:
################################################################################
################################################################################
```

#these feature plots, generated in Seurat, use similar code to that described in Main Fig 5D, except it plots the AddModuleScore for each cell for the genes identified to be most highly expressed in the Atf5 translating cells (by bulk RNA Seq on sorted cells, see Supplementary Fig5C). 318 Genes were used in this analysis.

```
################################################################################
################################################################################
#########################SUPPLEMENTARY FigureS6A:
################################################################################
################################################################################
```

#this box plot uses similar code to that described in Main Fig 6B, except it compares OMP-tTA Mex3a Tg sorted mCherry+ cells to OMP-tTA GFP Tg GFP+ cells

```
################################################################################
################################################################################
#########################SUPPLEMENTARY FigureS6B:
################################################################################
################################################################################
```

#Nikon Instruments Software (NIS-Elements) with the segment.ai function was used to select and measure ps6 + cells. To eliminate ps6 signal in non-OSN parts of tissue, an initial training was conducted on the DAPI dense undulations where OSNs reside in the tissue. Only ps6+ cells within those regions were counted and measured for further analysis.

```
################################################################################
################################################################################
#########################SUPPLEMENTARY FigureS6D:
################################################################################
################################################################################
```

```
# Load required libraries
library(dplyr)
library(readr)

# Set working directory
setwd("/path/to/working/directory/6Mar25_ps6_TMex")

# Define a function to read files and add metadata
read_and_add_metadata <- function(file_path) {
  # Extract metadata from the filename (using "_" as the separator)
  file_info <- strsplit(basename(file_path), "_")[[1]]

  # Assign metadata to variables
  File_number <- file_info[1]
  Mouse_number <- file_info[2]
  Genotype <- file_info[3]
  antibody <- file_info[4]
```

```
  section_number <- file_info[5]
  secondary_color <- file_info[6]

  # Read the .txt file into a dataframe
  temp_data <- read.delim(file_path, stringsAsFactors = FALSE)

  # Rename the ObjectId by appending file metadata
  temp_data <- temp_data %>%
    mutate(
      ObjectId = paste(ObjectId, File_number, sep = "_"),  # Add File_number as a suffix to ObjectId
      File_number = File_number,
      Mouse_number = Mouse_number,
      Genotype = Genotype,
      antibody = antibody,
      section_number = section_number,
      secondary_color = secondary_color
    )

  return(temp_data)
}

# Loop through each file and read it into a list of dataframes
all_data <- lapply(file_list, read_and_add_metadata)

# Combine all dataframes into one large dataframe
combined_data <- bind_rows(all_data)

#####Number of Cells per coronal section
          ggplot(file_number_counts, aes(x = Genotype, y = count_per_file, fill = Genotype)) +  # Use fill for
          coloring boxes
            geom_boxplot(outlier.shape = NA, color = "black") +  # Boxplot with red outliers and black
          borders
            scale_fill_manual(values=c("#004488", "#D55E00")) +
            geom_jitter(width = 0.2, alpha = 1)+ # Custom color scheme for Genotypes
            labs(
              x = "Genotype",
              y = "ps6 positive OSNs per Coronal MOE section",
              title = "ps6 positive OSNs per coronal section",
              color = "Genotype"
            ) +
            theme_classic() +  # Classic theme
            theme(
              text = element_text(size = 14),
              panel.grid.major.y = element_line(color = "lightgray", size = 0.5),
              panel.grid.minor.y = element_line(color = "lightgray", size = 0.25),
              panel.grid.major.x = element_line(color = "lightgray", size = 0.5) # Add light gray vertical
          gridlines
            )

#Calculate significance
var.test(count_per_file ~ Genotype, file_number_counts)
#check significance, if p.val is significant, the variance is not equal, do not set var.equal to TRUE
#p-value 0.1126
t.test(count_per_file ~ Genotype, file_number_counts, var.equal = TRUE)
```

```
#####Mean fluorescence intensity for each ps6+ cell
ggplot(combined_data, aes(x = Genotype, y = log_MeanObjectIntensity, fill = Genotype)) +
            geom_violin() +
            scale_fill_manual(values=c("#004488", "#D55E00")) +
            geom_jitter(width = 0.1, alpha = 0.05) + # Boxplot inside the violin with white fill
            geom_boxplot(fill = 'white', width = 0.1,outlier.shape = NA) +
            labs(
              x = "Genotype",
              y = "Log-Transformed Mean Object Intensity",
              title = "ps6 Log-Transformed Mean Object Intensity per OSN",
              fill = "Genotype"
            ) +
            theme_classic() +  # Classic theme
            theme(
              text = element_text(size = 14),
              panel.grid.major.y = element_line(color = "lightgray", size = 0.5),
              panel.grid.minor.y = element_line(color = "lightgray", size = 0.25),
              panel.grid.major.x = element_line(color = "lightgray", size = 0.5) # Add light gray vertical
          gridlines
            )

#Calculate significance
var.test(log_MeanObjectIntensity ~ Genotype, combined_data)
#check significance, if p.val is significant, the variance is not equal, do not set var.equal to TRUE
#p-value < 2.2e-16
t.test(log_MeanObjectIntensity ~ Genotype, combined_data)
```