Emergence of neuronal diversity during vertebrate brain development

Bushra Raj[1,2]*, Jeffrey A. Farrell[1,3], Jialin Liu[2,4], Jakob El Kholtei[2,4], Adam Carte[1,4,5], Joaquin Navajas Acedo[2,4], Lucia Y Du[2,4], Aaron McKenna[6], Đorđe Relić[3,7], Jessica M. Leslie[1], and Alexander F. Schier[1,2,4,8,9,10]*

[1] Department of Molecular and Cellular Biology, Harvard University, Cambridge, Massachusetts, USA
[2] Allen Discovery Center for Cell Lineage Tracing, Seattle, Washington, USA
[3] Unit on Cell Specification and Differentiation, National Institute of Child Health and Human Development, NIH, Bethesda, Maryland, USA
[4] Biozentrum, University of Basel, Switzerland
[5] Systems, Synthetic, and Quantitative Biology Program, Harvard University, Cambridge, Massachusetts, USA
[6] Department of Molecular and Systems Biology, Dartmouth Geisel School of Medicine, Lebanon, New Hampshire, USA
[7] Swiss Institute of Bioinformatics (SIB), Basel, Switzerland
[8] Broad Institute of MIT and Harvard, Cambridge, Massachusetts, USA
[9] Harvard Stem Cell Institute, Harvard University, Cambridge, Massachusetts, USA
[10] Center for Brain Science, Harvard University, Cambridge, Massachusetts, USA

*Co-corresponding authors. Email: bushranraj@gmail.com (B.R.); alex.schier@unibas.ch (A.F.S.)

ABSTRACT

Neurogenesis comprises many steps from progenitor proliferation to neuronal differentiation and maturation. These processes are highly regulated, but the landscape of transcriptional changes underlying brain development are poorly characterized. Here, we describe a developmental single-cell RNA-seq catalog of ~220,000 zebrafish brain cells encompassing 12 stages from 12 hours post-fertilization to 15 days post-fertilization. We characterize known and novel gene markers for ~800 clusters and provide an overview of the diversification of neurons and progenitors across these timepoints. We also introduce an optimized version of the GESTALT lineage recorder that enables higher expression and recovery of Cas9-edited barcodes to query lineage segregation. Cell type characterization indicates that most embryonic neural progenitor states are transitory and transcriptionally distinct from neural progenitors of post-embryonic stages. Reconstruction of cell specification trajectories reveals that late-stage retinal neural progenitors transcriptionally overlap cell states observed in the embryo. The zebrafish brain development atlas provides a resource to define and manipulate specific subsets of neurons and to uncover the molecular mechanisms underlying vertebrate neurogenesis.

INTRODUCTION

The vertebrate brain develops from a limited pool of embryonic neural progenitor cells that cycle through rounds of proliferation, diversification, and terminal differentiation into an extensive catalogue of distinct neuronal and glial cell types. A central goal in developmental neurobiology is to investigate how neuronal complexity arises through molecular specification and commitment by studying the origins and fates of cells during development. Fundamental insights into these processes have been gained via classic approaches using genetic markers, perturbations and fate mapping (Cepko, 2014; Kretzschmar and Watt, 2012; Ma et al., 2017; Wamsley and Fishell, 2017; Wilson et al., 2002; Woo and Fraser, 1995; Woodworth et al., 2017). These approaches have recently been complemented by single-cell genomics technologies in the developing nervous system, including the spinal cord (Delile et al., 2019; Rosenberg et al., 2018); cortex (Nowakowski et al., 2017; Zhong et al., 2018); olfactory system (H. Li et al., 2017); cerebellum (Carter et al., 2018; Tambalo et al., 2020); retina (Clark et al., 2019; Hu et al., 2019; Xu et al., 2020); and whole animal (Farnsworth et al., 2020). These studies have provided transcriptome-level views of the rich heterogeneous states that cells progress through as they proliferate, migrate and differentiate. Nevertheless, existing datasets are limited in their scope as they focus on specific brain regions, survey limited timepoints or do not enrich for neural cell types, thereby missing transitions and cellular diversity. Thus, there is a need for a large-scale neurodevelopmental single-cell resource that profiles whole brain development across a range of closely-spaced embryonic and post-embryonic stages. In addition, such an atlas would help address fundamental questions about the dynamics of brain development. For example, it is poorly understood how embryonic neural progenitors are molecularly related to post-embryonic neural progenitors. Furthermore, the transcriptional programs that are activated or suppressed as neural progenitors become fate-restricted and differentiate are largely unknown.

Here we present resources to obtain global views of neurogenesis, cell type heterogeneity, specification trajectories and lineage relationships in the developing zebrafish brain. We generated a single-cell RNA-seq (scRNA-seq) atlas consisting of ~220,000 cells from 12 hours post fertilization (hpf) to 15 days post fertilization (dpf). We also created a new version of the scGESTALT CRISPR-Cas9 lineage recorder (Raj et al., 2018b) with improved barcode capture and used it to query early lineage decisions. Using the cell type atlas, we analyzed the expansion of neuronal diversity, the loss of transitory embryonic progenitors, and the maintenance of distinct larval progenitor states. We reconstructed cell specification trajectories of the zebrafish retina and hypothalamus, revealing gene expression cascades and distinct specification programs. Collectively, the zebrafish brain development atlas reveals molecular and cellular changes at an unprecedented scale and resolution, and lays the foundation for the detailed analysis of neuronal diversification.

RESULTS

Building a developmental atlas of the zebrafish brain with single-cell transcriptomics

To reveal the landscape of cell states and cell types during brain development, we profiled 223,037 cells across 12 stages of zebrafish embryonic and larval development using the 10X Chromium scRNA-seq platform. Samples spanned from 12 hpf (shortly after gastrulation), when the embryo is undergoing early developmental patterning, to 15 dpf, when larvae are mature, exhibit complex behaviors, and are expected to exhibit substantial cell type diversity (Figure 1A). To enrich for brain cell types, we dissected the heads of animals from 12 hpf to 3 dpf, and the brains and eyes from 5 dpf to 15 dpf (Figure 1B). To determine cell type diversity in the head and brain of zebrafish, data from each stage was analyzed individually using Louvain clustering (Figure 1C and Sup Figure 1). This approach identified a total of 815 cell clusters across all 12 timepoints (Sup Table). To classify each cluster, we compared enriched gene markers with existing gene expression annotations in the ZFIN database and literature, as described previously (Raj et al., 2018b). Plotting expression of known cell type markers identified clusters corresponding to neural progenitors (*sox19a*), dozens of neuron subtypes (*elavl3*, *gad2*, *slc17a6b*), eye cells (*foxg1b*, *lim2.4*, *pmela*, *ca14*, *gnat1*, *opn1mw1*), radial glia (*mfge8a*, *s100b*), neural crest (*sox10*), oligodendrocytes (*mbpa*), blood cells (*cahz*, *etv2, cd74a*), cartilage (*matn4, col9a2*), pharyngeal arches (*pmp22a*, *prrx1b*, *barx1*), sensory placodes (*dlx3b*, *six1b*), and epidermal cells (*epcam*, *cldni*), among others. As expected, cell type complexity increased with developmental time. We validated new marker expression across several cell types identified in our dataset, such as *sdpra* in the trigeminal placode, *sox1a* in the hypothalamus, and *ompa* in the retina (Figure 1D-F). Our analysis also revealed groups of embryonic clusters that were absent or transcriptionally distinct from larval clusters, suggesting that many embryonic cell states are transitory. Several of these transitions are known developmental changes (e.g. loss of placodes and rhombomeres), but changes in neural progenitor cell states are poorly understood (see below).

To enable direct comparison of cell types across our time course, we subsetted the 12 hpf dataset to only comprise neural populations and blood cells found in the brain, eliminating non-relevant head cells from earlier stages, such as mesoderm, placodes, and periderm. This approach resulted in an initial set of 21 clusters at 12 hpf (Figure 2A) that diversified into 98 clusters by 15 dpf (Figure 2C). Notably, most clusters could be uniquely identified using a minimal group of 2-3 enriched gene markers (Figure 2B, 2D). For example, at 12 hpf, the optic vesicle is identified by expression of *rx2* and *rx3*; hindbrain rhombomeres 5/6 by *hoxb3a* and *eng2b*; and ventral diencephalon by *nkx2.4a* and *dbx1a*. Similarly, at 15 dpf, the cerebellar granule cells are marked by expression of *oprd1b* and *zic2a*; optic tectum by *pax7a* and *tal1*; and a new retinal cell type by *kidins220a*, *foxg1b* (exclusively detected in retinal cells) and *tbx3a*. We did not find unique gene combinations for cycling progenitors, differentiating progenitors and newly born neurons, as many of these subtypes had similar expression signatures of pan neuronal or pan progenitor marker genes, such as *elavl3* and *tubb5* in neurons, and *rpl5a* and *npm1a* in progenitors (Figure 2D, grey box).

At 12 hpf, the early demarcation of multiple brain regions is already apparent and by 15 dpf these regions expand and diversify further. For example, the optic vesicle at 12 hpf is defined by one cluster and is the origin of 18 retinal cell types at 15 dpf. Similarly, a single cluster of ventral diencephalon cells (expressing *shha*, *nkx2.4a*, *nkx2.1*, *rx3*) at 12 hpf develops into 7 major hypothalamus cell types at 15 dpf. An exception to this diversification is the loss of rhombomeres (r1-r7) in the hindbrain (Moens and Prince, 2002).

To further explore brain neuronal subtypes at 15 dpf, we analyzed the expression of transcription factors, neuropeptides and their receptors, and genes involved in neuronal physiology (e.g. neurotransmitters, transporters, receptors, and channels) (R. Chen et al., 2017; Pandey et al., 2018; Tiklová et al., 2019; Zeisel et al., 2018). Our results indicate that nearly all identified neuron subtypes can be distinguished from one another via the expression of individual or combinations of genes belonging to these categories (Figure 3A-C). For example, cluster 2 and 84 neurons are GABAergic forebrain neurons that express *dlx2a* and *dlx5a*, while cluster 84 neurons additionally express *six3b*, *gria1a* and *gria2b*.

We next asked if neuron clusters detected at 15 dpf are found in the earlier larval stages, when most behavioral experiments are performed (Sup Figure 2). 68% (23/34 clusters) and 74% (25/34 clusters) of 15 dpf neuron clusters have a closely matching counterpart at 5 dpf and 8 dpf (based on enriched marker gene expression), respectively (Figure 3D). Sampling issues might have prevented the identification of additional overlapping clusters, but our data indicate a large overlap between identified cell types from 5 to 15 dpf. These results suggest that the zebrafish brain already has considerable cell type diversity at early larval stages. Furthermore, 97% (33/34) of 15 dpf clusters overlapped with clusters identified in our previously described 23-25 dpf juvenile brain dataset (Raj et al., 2018b). Thus, by 15 dpf late larval stage, nearly all of the brain cell types that persist into the early juvenile stage have already been established. Notably, among cell types that are “missing” or under-represented at 15 dpf but readily detected at 23-25 dpf are cell types in the optic tectum, cerebellum and the torus longitudinalis, suggesting that these structures undergo further diversification after 15 dpf. In contrast, many cell types in the pallium, habenula (Pandey

et al., 2018), hypothalamus and preoptic area are detected across these stages, suggesting that they develop earlier.

In summary, we generated a zebrafish brain development cell type atlas spanning 12 stages of brain organogenesis. The complete dataset can be explored using the accompanying app: https://github.com/brlauuu/zf_brain.

Neurogenic expansion during brain development

During development, cell composition shifts from predominantly progenitor populations to more differentiated cell types (Schmidt et al., 2013). To better characterize how differentiation varies during neuronal development, we first asked if our dataset captured the two neurogenic phases (primary and secondary) before and after 2 dpf that have been traditionally defined through histological analyses (Allende and Weinberg, 1994; Korzh et al., 1998; Mueller and Wullimann, 2003). We considered neural progenitors as non-differentiated neuronal precursor cells that may or may not be proliferating, and express a subset of classical progenitor markers e.g. *sox19a*, *dla*, *s100b*, and cell cycle genes. Since the brain is undergoing substantial molecular changes during these developmental windows, we defined the transcriptional programs and cells that exhibit these programs as progenitor cell states. We calculated the percentage of the dataset that corresponds to neural progenitor cells, neurons (expressing markers such as *elavl3*, *elavl4*) or other cell types across each timepoint in our dataset. Since the earlier stages (12 hpf to 3 dpf) contained non-brain and non-eye cell types, while later stages were restricted to these tissues, we subsetted the early timepoints to only brain and eye cells. With increasing developmental time, we observed a progressive decrease in the fraction of the dataset comprising neural progenitor cells (from 53.8% to 18.3%) with a concomitant increase in neurons (from 4.5% to 58%) (Figure 4A). For example, we observed an initial increase in the number of distinct progenitor clusters from 12 hpf to 18 hpf (early embryo stages), while the number of neuron clusters remained low (Figure 4A, right panels). From 20 hpf to 3 dpf (intermediate stages), the total progenitor clusters decreased while neuron clusters started to increase. For example, neuronal clusters expanded from 11 at 20 hpf to 23 at 36 hpf. This burst coincides with the presumed timing of late-stage primary neurogenesis in zebrafish (Mueller and Wullimann, 2003). Notably, by 5 to 15 dpf (late larva stages), a second expansion of neuronal populations, corresponding to the secondary neurogenic phase (Mueller and Wullimann, 2003), had occurred (53 neuronal subtypes at 5 dpf). At 5 dpf, we detected cell types identified as early as 36 hpf (e.g. *tal1*$^{+}$, *gata3*$^{+}$ neurons in the optic tectum, and *tfap2e*$^{+}$, *barhl2*$^{+}$ neurons in the thalamus), as well as subtypes only observed during the second phase, such as *nrgnb+ prkcda+* neurons in the forebrain and cone bipolar cell subtypes in the retina. Collectively, our dataset captures both phases of neurogenesis and reveals the diversification of neurons in multiple brain structures.

Dampening of spatial and developmental signatures during the transition from embryonic to larval neural progenitors

We next analyzed our dataset to determine how cell states change during the transition from the embryonic to post-embryonic brain. The zebrafish brain undergoes lifetime constitutive

neurogenesis due to the persistence of neural progenitor pools distributed along the brain's axis (Schmidt et al., 2013). However, the embryonic origins and transcriptional programs that underlie their development are poorly understood. Furthermore, how the molecular identities of embryonic and post-embryonic neural progenitor cell states compare have not been well characterized. To address these questions, we asked how neural progenitor gene expression signatures globally change from embryo to larva. Based on the results described above, we defined early embryonic brain progenitors as neural cell transcriptional states from 12 hpf to 18 hpf, intermediate stage brain progenitors as neural cell transcriptional states from 20 hpf to 3 dpf, and larval brain progenitors as neural cell transcriptional states from 5 dpf to 15 dpf (Figure 4B, Sup Figure 3). We determined the greatest sources of variation within these populations. For embryonic brain progenitors we found that the top 3 principal components comprise genes implicated in spatial and developmental patterning (Gibbs et al., 2017; Moens and Prince, 2002; Wilson et al., 2002; Wilson and Rubenstein, 2000). Cells exhibit characteristic anteroposterior and dorsoventral axial signatures (Figure 4C, top panel). For example, the telencephalon (anterior forebrain) is marked by *foxg1a* and *emx3a* expression, the midbrain by *pax2a* and *eng2a*, and the hindbrain is segmented into rhombomeres marked by distinct combinatorial patterns of *egr2b* and *hox* gene expression. Furthermore, all cells are in a highly proliferative state with strong expression of cell cycle genes such as *pcna*, *mki67* and *cdca7a*. Collectively, the expression signatures are reflective of a developmental state during which the embryo is orchestrating a rapid expansion of neural progenitor populations concurrent with their acquisition of positional information and overt absence of differentiation (Schmidt et al., 2013; Stigloher et al., 2008).

In contrast, larval neural progenitors comprised two major groups: proliferating (expressing cell cycle genes *pcna* and *top2a*) and non-proliferating (depleted expression of cell cycle markers) (Figure 4D, bottom panel). Indeed, the top 3 principal components in the larval progenitors comprised genes that mark stem cells (PC1, PC3) and differentiation (PC2). The non-proliferating group is subdivided into radial glia (stem cells) and *her2*$^{+}$ neural progenitors expressing proneural genes *insm1b* and *scrt2*. The proliferating group is subdivided into *her2*$^{+}$ and *scrt2*$^{-}$ neural progenitor cells, *her2*$^{-}$ progenitors, *her2*$^{+}$ and *neurod1*$^{+}$ progenitor cells, and upper rhombic lip progenitors (localized to cerebellum) expressing *atoh1c* and *oprd1b*.

Strikingly, most larval progenitors were characterized by a reduced spatial signature (except for the cerebellar upper rhombic lip pool), such that cells were less enriched in region-specific transcription factors relative to embryonic progenitors (Figure 4D, top panel). For example, radial glia exist in multiple pools along the brain axis (Than-Trong and Bally-Cuif, 2015), but they formed a single cluster in our dataset (marked by expression of *fabp7a*, *cx43*, *s100b* and *aqp1a.1*). This result suggests that radial glia are largely transcriptionally similar. Although some expression of region-specific transcription factors was detected in larval progenitor clusters, these signatures were not sufficiently strong to resolve clusters as they were during embryonic stages.

To explore the apparent dearth of spatial signatures further, we calculated pairwise correlation scores for 79 transcription factors and signaling proteins with known spatial expression patterns in the forebrain and midbrain based on previously described histological analysis (ZFIN), and which were identified as gene markers for neuronal clusters in our dataset. These genes showed

strongest correlations in embryonic progenitors, followed by intermediate stage progenitors, and were weakly correlated in larval progenitors (Figure 4E).

Since spatial signatures are encoded by a combinatorial code of genes with overlapping expression patterns, we asked whether the same subsets of genes co-varied with each of the 79 spatial markers across embryonic, intermediate, and larval neural progenitors. We found that intermediate stage progenitors showed overlap in co-varying genes with both embryonic and larval progenitors. For example, 44/79 genes had >40% overlap in their top 20 co-varying genes between embryonic and intermediate stage progenitors, and 23/79 genes had >40% overlap between intermediate and larval stage progenitors. In contrast, we found low overlap across embryonic and larval stages (3/79 genes had >40% overlap in their top 20 co-varying genes). Additionally, when we searched for genes that strongly co-varied with these 79 spatial markers (Pearson correlation >0.4), we found 38 genes during embryonic stages, 17 genes during intermediate stages, but only 4 genes during larval stages (Figure 4F).

Taken together, these results demonstrate that intermediate stage progenitors resemble a hybrid of early embryonic and late larval progenitor signatures. Furthermore, the overall spatial code between embryonic and larval progenitors are distinct, and the embryonic spatial code involves a larger collection of genes. Notably, the signatures of larval progenitors resemble juvenile neural progenitor pools (Raj et al., 2018b), indicating developmental switches in neural progenitor identities from embryo to larva that are maintained to at least juvenile stages. Thus, embryonic states that existed in early progenitors are largely altered in late-stage progenitors: while spatial patterning signals are the greatest source of variation between embryonic neural progenitors, these signals are dampened in post-embryonic neural progenitors.

An optimized scGESTALT lineage recorder

A long-term goal in developmental neurobiology is to understand the lineage relationships of neurons. As a first step to derive lineage relationships of the cell types identified in the brain development atlas, we performed lineage recording experiments with scGESTALT. This lineage recorder enables simultaneous cell type and cell lineage identification by combining scRNA-seq with CRISPR-Cas9 barcode editing (McKenna et al., 2016; Raj et al., 2018b). To enable higher recovery of edited barcodes from single cells, we optimized the design and library preparation of the lineage recording cassette, including barcode editing of a transgene coding region and compatibility with the 10x platform (see Methods). To test the performance of this new recording cassette, we barcoded early embryonic lineage relationships by injecting Cas9 protein and target guide RNAs into 1-cell embryos (Figure 5A) and then isolated four 15 dpf larval brains. We recovered barcodes and transcriptional profiles of 5,794 cells total (barcode recovery rate 30-75% compared to 6-28% of our previous scGESTALT version (Raj et al., 2018b)). Edited barcodes showed no overlap between animals, displayed a diverse spectrum of repair products that spanned single and multiple sites, and were of varying clone sizes (Figure 5B-D, Sup Figure 4A). These features closely resembled the editing patterns obtained with our previous recorders (McKenna et al., 2016; Raj et al., 2018b). Using the recovered barcodes and associated transcriptomes, we reconstructed lineage trees representing cell lineage segregations formed

during early embryogenesis (for one example see Sup Figure 4B). These lineage trees accompany our transcriptional cell type atlas and are available to explore at https://scgestalt.mckennalab.org/

Since the injection of editing reagents into 1-cell embryos saturates editing within 4-6 hours (McKenna et al., 2016) , we expected early lineage divergences to be overrepresented in our dataset. We first asked if our recorder captured diverse multi-lineage tissue origins of the eye, which is derived from neuroectoderm, surface ectoderm and mesoderm (Figure 5E). Eye cell types were identified as clusters that contained cells from scRNA-seq samples comprising eye tissue exclusively. Retinal cell types were defined as clusters expressing the pan-retinal marker *foxg1b* (Figure 1F), whereas non-retinal cell types were depleted in *foxg1b*. We performed pairwise comparisons of all eye clusters with at least 4 independent barcodes (each with at least 2 cells). Since <1% of all barcoded cells were captured by scRNA-seq, we asked if there is cell type-specific barcode enrichment greater than expected by chance ("lineage segregation" in Figure 5E). For cluster pairs where we did not observe significant lineage segregation, we asked if this was due to a lack of sampling ("lineage status undefined") or true lack of cell type-specific barcode enrichment ("no lineage segregation"). The latter case would indicate that two cell types shared a more recent common ancestor than cell types that segregated earlier. We found that multiple retinal and non-retinal cell types segregated from each other, as would be expected due to early separation of their tissue origins. Interestingly, however, a few non-retinal cell types (e.g. clusters 34, 44, 49) did not fully segregate from retinal cell types, suggesting that they shared a common progenitor. Furthermore, there was extensive lineage segregation between various non-retinal cell types (e.g. clusters 45, 47, 86). In contrast, we did not observe lineage segregation between the different retinal cell types, likely due to the termination of barcode editing prior to terminal divisions. The exception was cluster 28 (cones), which segregated from clusters 15 and 32 (cone bipolar cells) and 28 (retinal ganglion cells). Thus, lineage splits between retinal and non-retinal cell types, and within non-retinal subtypes preceded most splits within retinal subtypes.

Next, we asked if our recorder captured lineage divergences between neurons across brain regions and the retina. Although the hindbrain and retina formed distinct lineages early in development, forebrain and midbrain neurons continued to share progenitors across the same barcoding period (Figure 5F). Pairwise comparisons of all forebrain and midbrain clusters revealed examples of emerging segregation along multiple spatial axes (Figure 5G). For example, we saw evidence of dorsal-ventral split: cluster 9 pallium (dorsal) separated from cluster 25 sub-pallium (ventral). Furthermore, barcode enrichments confirmed rostral-caudal splits: cluster 64 habenula separated from clusters 9 and 25 pallium (telencephalon, rostral) and clusters 0 and 13 optic tectum (caudal). Overall, the lineage segregations agreed with classic fate mapping experiments (Woo and Fraser, 1995) and correlate with the anteroposterior and dorsoventral gene expression signatures of early progenitors (Figure 4).

To query the lineage relationships of brain progenitor cell types, we performed pairwise comparisons of progenitor clusters at 15 dpf (Figure 5H). Notably, the upper rhombic lip (URL) progenitors (cluster 12) formed a separate lineage from all progenitor classes except cluster 74, a cycling progenitor subtype expressing *pif1*. Since URL progenitors give rise to granule cells in

the cerebellum, we asked if the two cell types shared barcodes. We found that the proportion of barcode overlap was highest between granule cells and URL progenitors (Figure 5I). The URL progenitors formed a distinct cluster as early as 12 hpf (cluster 9) in our transcriptional dataset. Thus, URL progenitors become discrete in both lineage and transcriptional signature relatively early in development.

In summary, we present an optimized scGESTALT cassette with improved lineage barcode expression and recovery by scRNA-seq. The barcodes display high sequence diversity, which is important for generating large-scale distinct labels in a developing animal. The scGESTALT transgenic line is available as a resource for the community and can be paired with other transgenic lines for temporal, spatial or cell-type specific control of barcode editing (see Discussion).

Cell specification trajectories in the retina and hypothalamus

With the exception of a few model systems (Clark et al., 2019; Delile et al., 2019; Guo and J. Y. H. Li, 2019; Holguera and Desplan, 2018; Kim et al., 2019; Tambalo et al., 2019), little is known about gene expression cascades that accompany the development of progenitors into terminally differentiated neurons. To address how different neuronal populations become molecularly specialized, we reconstructed gene expression trajectories from 12 hpf to 15 dpf. We first tested our approach on the subsetted retina dataset in which cell types expand from a single cluster at 12 hpf to 18 clusters at 15 dpf (Figure 2). UMAP embedding of the subsetted dataset revealed progressive paths from the embryonic state to defined cell types at 15 dpf (Figure 6A, Sup Figure 5A). One outlier cluster that expressed *kidins220a* and whose progenitor state may not have been captured in our timepoints, was excluded from further analysis. Although UMAP represents continuity in the data, it does not order individual cells according to their relative developmental time (i.e. pseudotime). Therefore, we also used URD (Farrell et al., 2018) to construct a branching specification tree that represents the developmental trajectories in the retina at a higher resolution (Figure 6B, Sup Figure 5B, Sup Figure 6A-B). Many of the major branching features agreed with the UMAP representation. For example, the trajectories revealed the early segregation of RPE, shared branching of photoreceptor cells, a path towards multiple cone bipolar cell subtypes, and a common branchpoint between amacrine and retinal ganglion cells (RGC).

Plotting gene expression of known early regulators of eye development and terminal cell type markers on the URD tree supported the inferred specification branches (Figure 6C, Sup Figure 7). For example, *pax6a* was most enriched in the amacrine and RGC branches, and *vsx1* marked cone bipolar cells with *fezf2* marking one specific subtype. Notably, our analysis also revealed previously unknown markers and characteristics of horizontal and amacrine cells. Zebrafish horizontal cells are GABAergic (*gad2*$^{+}$, *gad1b*$^{+}$), but unlike mammals where these cells do not express GABA membrane uptake transporters (Deniz et al., 2011), zebrafish cells expressed *slc6a1l* (likely a duplication of *slc6a1* involved in GABA uptake from the synaptic cleft), suggesting that they may be capable of uptake. Additionally, whereas *slc32a1* GABA transporter is expressed in mouse horizontal and amacrine cells (Cueva et al., 2002), we observed restriction of *slc32a1*

to amacrine cells and *slc6a1l* to horizontal cells. Finally, we detected several novel horizontal cell markers such as *ompa* and *prkacaa* (Figure 1F).

To discover the gene expression trajectories from precursors to different retinal cell types, we used differential gene expression approaches that characterize pseudotime-ordered molecular trajectories. This analysis revealed known and novel regulatory steps (Figure 6D, Sup Figure 8). For example, RGC specification trajectories confirmed several known differentiation regulators including *sox11a*, *sox11b*, *sox6*, *irx4a*, and *pou4f2* (Rheaume et al., 2018). Similarly, known regulators of photoreceptor differentiation such as *isl2a* (Fischer et al., 2011), *prdm1a* (Brzezinski et al., 2010), *otx5* (Viczian et al., 2003), and *crx* (Shen and Raymond, 2004) were expressed early in our photoreceptor trajectories, while known regulators of cone versus rod fate, such as *six7* (Ogawa et al., 2015), *nr2f1b* (Satoh et al., 2009), and *nr2e3* (J. Chen et al., 2005) were expressed as those trajectories diverged. Furthermore, our analysis revealed novel transcription factors within the gene expression cascades. For example, we detected *runx1t1*, *foxp1b*, *mef2aa* in the RGC pathway; *tfap2a* in horizontal cell trajectory; and *tbx3a* and *tbx2a* in amacrine cell branches. Interestingly, among signaling pathways, we found that both apelin receptors (*aplnra*, *aplnrb*) were expressed in photoreceptor progenitors, while one of their ligands (*apln*) was expressed in differentiating cones; this suggests a potential cell autonomous role for apelin signaling in photoreceptor cells in addition to its role in preventing photoreceptor degeneration via vascular remodeling (McKenzie et al., 2012).

A surprising result from this analysis was that a Muller glia pathway was detected earlier in zebrafish than expected based on studies in mouse, where these cells are detected late (Centanin and Wittbrodt, 2014; Clark et al., 2019). We found a cluster of cells as early as 20 hpf (cluster 50) that expresses markers (e.g. *cahz*, *rlbp1a*) that are shared with the Muller glia cluster (cluster 33) at 15 dpf (Sup Table). smFISH analysis of Muller glia markers validated their expression at 36 hpf and 2 dpf (Sup Figure 9). Similarly, in our transcriptional trajectories (Figure 6B), the Muller glia expression program is the earliest non-epithelial retinal program to diverge, commencing with the expression of several *her*-family transcription factors (*her4*, *her12*, and *her15*), then proceeding through a cascade of intermediate overlapping expression states such as onset of *fabp7a, s100a10b,* and later connexin genes that are characteristic of Muller glia fate (Sup Figure 8). Cells from all timepoints can already be found in the early part of the Muller glia branch. These observations suggest that cells early in development transition from a naive progenitor state to a Muller glia-like transcriptional state, and do so continually during larval development.

To extend our analysis to a central brain region, we reconstructed specification trajectories and expression cascades for hypothalamic neurons. These cells expanded from a single ventral diencephalon cluster at 12 hpf to 7 clusters at 15 dpf (Figure 6E-H, Sup Figure 6C-D, Sup Figure 10). The earliest branchpoint denoted segregation of *prdx1*$^{+}$ and *prdx1*$^{-}$ cells. Committed hypothalamic progenitors in the *prdx1*$^{-}$ trajectory gave rise to neuronal precursors expressing proneural transcription factors such as *ascl1a*, *scrt2*, *insm1a* and *elavl3* (early neuron fate marker) (Sup Figure 11). The specified cell types then matured over time and were characterized by expression of neuronal maturation markers such as *tubb5*, *gap43*, *ywhag2*, *snap25a*, *scg2b* and *elavl4*. The *prdx1*$^{-}$ group further diverged into two major groups: *nrgna*$^{+}$ and *nrgna*$^{-}$ trajectories

(Figure 6F). The *nrgna*$^{+}$ branch segregated into GABAergic *tac1*$^{+}$, *synpr*$^{-}$ subtype and GABAergic *tac1*$^{+}$, *synpr*$^{+}$ positive subtype. The *nrgna*$^{-}$ branch subdivided into glutamatergic *pdyn*$^{+}$ neurons and a GABAergic branch that further resolved to *sst1.1*$^{+}$ and *tph2*$^{+}$ neuron subtypes. We detected expression of known regulators of hypothalamus development in the early branches such as *shha*, *rx3*, *nkx2.4b*. We also identified new candidate regulators in later branches including *nrgna* in the *synpr*$^{+}$ and *synpr*$^{-}$ trajectories, and *sox1a*, *sox1b* and *sox14* in the *pdyn*$^{+}$ trajectory (Sup Figure 12, Figure 1E). The results in the retina and hypothalamus demonstrate that the brain development atlas can be used to reconstruct neuronal differentiation trajectories and define the underlying gene expression cascades

Differences in progenitor specification strategies between retina and hypothalamus

Pseudotime analysis represents cell trajectories in relative but not absolute time (Bendall et al., 2014; Trapnell et al., 2014). Therefore, comparing the developmental and pseudotime age of cells can define whether molecular states are unique to a given developmental stage or persist through development (Figure 6B, 6F). For example, mapping RGC and *pdyn*$^{+}$ neurons from different developmental stages onto the pseudotime trajectory showed the expected maturation of these cell types with developmental age (Sup Figure 13). In addition, even at 15 dpf some RGC and *pdyn*$^{+}$ neurons were still in an immature state, consistent with the continuous growth and differentiation in the zebrafish retina and brain (Centanin and Wittbrodt, 2014; Schmidt et al., 2013).

To systematically analyze the relationships of pseudotime state and developmental stage, we mapped differentiated cells, precursors and progenitors found in different pseudotime windows to their origin in developmental time. We found that the proportion of differentiated cells increased, whereas the number of early progenitors in both retina and hypothalamus decreased with developmental age. In contrast, precursor cells from an intermediate pseudotime window were present in embryo and larva. These precursor cells expressed genes that were an intermediate of progenitor (e.g. *insm1a*, *her4.1* in hypothalamus (Xie and Dorsky, 2017); *hes2.2*, *rx2* in retina) and early differentiation genes (e.g. *tubb5*, *gap43* in hypothalamus; *foxg1b* in retina). In addition, a second class of retinal progenitors mapped to an earlier pseudotime trajectory but was also present from embryonic to late larval stages (Figure 7, Sup Figure 14). Comparison of these progenitors between 24-36 hpf and 15 dpf identified only 71 differentially expressed genes. The majority of these genes (56/71) increased in all cells of the retina between these stages, while a few (15/71) were only upregulated in the 15 dpf group. A similar population was not detected in the hypothalamus. These observations suggest that as the retina grows, some progenitor cell states observed in the embryo persist later in development without extensive maturation.

DISCUSSION

As the brain develops, embryonic neural progenitor pools transition through many cellular states as they become more committed, diversify into post-embryonic neural progenitors, and undergo

terminal differentiation. Although regulators and transcriptional changes of this process have been identified (e.g. using specific driver lines and *in situ* detection of select genes), the global transcriptional networks mediating the sequential activation and maturation of neurogenic programs from embryo to later stages are largely unknown. To help address this question, we used scRNA-seq to generate a zebrafish brain development atlas. This resource supports the identification of marker genes, the comparison of cell types, and the dissection of cell specification and differentiation trajectories during vertebrate brain development.

Our data address how the transcriptional programs of neural progenitors vary and contribute to fate-restriction during development. Different models to explain these processes have been proposed. For example, neural progenitors of the medial and lateral mouse ganglionic eminence, which give rise to cortical interneurons, have been found to converge to a shared mitotic signature regardless of their region of origin, followed by expression of cardinal fate-specific transcription factors post-mitotically (Mayer et al., 2018). In contrast, the spinal cord has dedicated pools of domain-specific neural progenitors that retain domain-specific signatures (Delile et al., 2019; Jessell, 2000; Lee and Pfaff, 2001; Sagner and Briscoe, 2019). Our results indicate that early embryonic neural progenitors in the brain are transcriptionally distinct from late larval neural progenitors. Gene expression profiles of neural progenitors switch from strong spatially segregated signatures in early embryos to proliferative and non-proliferative states in late larvae. These cell state changes might reflect developmental shifts from an establishment program during gastrulation, where strong spatial patterning cues set up regional boundaries, to a maintenance program at late stages, where progenitors are geographically confined and express dampened regional restriction signatures. Although expression of some spatially-enriched transcription factors (e.g. *pax6a*, *eng2a*, *nkx2.4a*) and signaling proteins detected in embryonic progenitors are also detected in late progenitors, the overall signatures are different, as these factors co-vary with different sets of genes in larva relative to embryo.

The expression of pan-progenitor markers at larval stages raises the question of how neural progenitor pools remain or become fate restricted. There are several different scenarios that might address this question. First, it is conceivable that embryo and larva share a minimal core set of regionally-restricted transcription factors that are sufficient to ensure spatial restriction, despite differences in their relative expression levels and downstream targets. Spatial genes that are highly expressed in the embryo may be lowly expressed in the larva, and be sufficient to maintain regionally-restricted cell states. Second, cell-type specific transcription factors rather than spatially defined regulators might guide specification and differentiation at these stages, independent of positional information. Such signatures would be difficult to analyze via scRNA-seq, which is biased towards recovering highly expressed genes. Third, it is also possible that restrictions at the genomic level, such as chromatin accessibility, may ensure that cells maintain the signature of their spatial origin. Fate mapping experiments of early and late neural progenitors, profiling open chromatin states of neural progenitors, and transcriptome analyses that recover lowly expressed genes will provide further insight into these questions.

Our reconstruction of specification trajectories for cell types in the retina and hypothalamus revealed several findings. First, our data supports a multipotent progenitor model whereby

multiple differentiated cell types can be traced to common post-embryonic progenitors. For example, all retinal neurons can be traced to an early pseudotime progenitor branch containing cells from larval stages, consistent with multipotency and fate stochasticity of zebrafish retinal progenitors (Boije et al., 2015; He et al., 2012). The early emergence of Muller glia observed in both the time course atlas and eye trajectory reconstruction is particularly interesting in light of clonal analyses. For example, single retinal progenitor cells in zebrafish give rise to clones comprised of neurons and one Muller glia cell (Rulands et al., 2018). This observation has been interpreted as evidence for a progenitor that first gives rise to neurons and then differentiates into a Muller glia cell. However, it is also conceivable based on our data that an early common progenitor divides, with one daughter expanding to give rise retinal neurons while the other daughter forms Muller glia. Second, our results reveal that whereas progenitor cell types in the rest of the brain appear molecularly distinct between the embryo and larva, there are progenitor cell states in the eye that are maintained from the embryo to larva (Figure 4 and Figure 7). A subset of 15 dpf retinal progenitors have similar transcriptional states as observed in the embryonic eye. This observation raises the possibility that a subset of long-term retinal progenitors may be "frozen" in an embryonic phase that could possibly underlie the multi-fate potential of these cells. An independent study of zebrafish retinal stem cells has proposed a similar conclusion (Xu et al., 2020). Collectively, these findings highlight differences in neurogenic programs in the central nervous system, and underscore the power of investigating multiple specification trajectories simultaneously.

Our results also highlight differences between zebrafish and mammalian neurogenesis. For example, we detected pan-neuronal transcriptional signatures (e.g. *neurod1*, *ascl1a*, *insm1a*, *neurog1*) in zebrafish radial glia and other progenitors at late stages of development, suggesting that neurons remain the principal output of these cells. This is consistent with fate mapping studies that have shown that zebrafish radial glia persist into adulthood and contribute to neurogenesis (Schmidt et al., 2013). In contrast, radial glia progenitor cells in the developing embryonic mouse brain shift from neurogenic to gliogenic programs (Mission et al., 1991; Schmechel and Rakic, 1979).

While developmental atlases and trajectories can help identify cellular differentiation paths, a full understanding of cell type specification requires lineage tracing experiments. To catalyze such approaches we introduced improvements to scGESTALT through a redesigned recorder cassette for optimized mRNA expression and library compatibility with the 10X Chromium scRNA-seq platform. The resulting higher recovery of barcodes allows more dense reconstruction of lineage trees. Our analysis revealed differences between the timing of segregation between different brain regions: neuronal lineages in the retina and hindbrain diverged earlier than the forebrain and midbrain. These results complement classic zebrafish fate maps of brain compartmentalization (Woo and Fraser, 1995) and recent analysis of clonal cells in forebrain and midbrain (Solek et al., 2017). Furthermore, our findings support early transcriptional and lineage segregation of cerebellar upper rhombic lip progenitors relative to other classes of progenitor cells. To query additional lineage divergences and combine with cellular trajectories, our optimized recorder can be readily adapted for barcoding lineages at developmental windows that correspond to different

branches of the specification trees (Raj et al., 2018b) or combined with cell- or tissue-specific Cas9 driver lines to introduce lineage labels in populations of interest.

The resources presented here lay the groundwork for characterizing lineage histories and transcriptional changes underlying the development and diversification of the vertebrate brain. Future extensions include the generation of transgenic reporters to select populations of interest and perform deeper analyses of cell type heterogeneity and differentiation (Pandey et al., 2018). Cell specification trajectories can be extended to include additional subregions of the brain to generate increasingly complex trees and combined with other zebrafish scRNA-seq datasets (Cosacak et al., 2019; Farnsworth et al., 2020; Farrell et al., 2018; Lange et al., 2020; Pandey et al., 2018; Tambalo et al., 2020; Wagner et al., 2018; Xu et al., 2020) to trace complete trajectories from gastrulation to adulthood. Finally, it will be interesting to perform comparative studies by using our atlas in conjunction with data described in a recent preprint (La Manno et al., 2020).

METHODS

Zebrafish husbandry

All vertebrate animal work was performed at the facilities of Harvard University, Faculty of Arts & Sciences (HU/FAS). This study was approved by the Harvard University/Faculty of Arts & Sciences Standing Committee on the Use of Animals in Research & Teaching under Protocol No. 25–08. The HU/FAS animal care and use program maintains full AAALAC accreditation, is assured with OLAW (A3593-01), and is currently registered with the USDA.

Chromogenic in situ hybridization

Embryos were dechorionated with forceps and then fixed in 4% PFA in 1X PBS (pH 7.4) overnight at 4°C. After fixation, embryos were dehydrated in methanol series (0%, 25%, 50%, 75% and 100% MetOH in PBSTween 0.3% (PBST)) and stored in 100% methanol at −20°C. Embryos were rehydrated by reversing the methanol series for 10 min in each step at room temperature (RT) and washed 2 × 5 min in PBST. To bleach pigment in 2 dpf fish, larvae were incubated for 10 min in bleaching solution (3% H_2O_2/0.5% KOH in ddH_2O) at room temperature and washed 3 x 5 min in PBST (Thisse et al., 2004). For permeabilization, 2 dpf larvae were incubated with Proteinase K (10 µg/ml in PBST) for 2 min at RT and postfixed in 4% PFA in 1X PBS for 30 min at RT. Afterwards, embryos were washed 3 x 5 min in PBST at RT, prehybridized in HYB^+ solution (50% Deionized Formamide (Amresco), 5X SSC (Ambion), 0.1% Tween-20, 5mg/ml Torula RNA (Sigma) in ddH2O) for 3 hours at 69°C, and hybridized overnight with the antisense probes diluted in HYB^+ at 69°C. The rest of the steps were performed as described previously, by hand (Navajas Acedo et al., 2019). Before imaging, embryos were cleared using an increasing MetOH series. For imaging of 12 hpf embryos, the yolk was dissected away, and the embryos were flat mounted on a microscope slide and covered with a cover slip. Larvae were photographed on a Zeiss AxioZoom.V16.

The antisense probes were synthetized from DNA fragments amplified from TLAB zebrafish cDNA using the following primers: *klf17* (Fw GAAGGAAAGACTGCATCCTGAC; Rv CTGCTGTCCCAAAATAGGAGTT), *ptgs2a* (Fw CGAGGACTATGTTCAGCACTTG; Rv

TGCACATCGATCACAATACAAA), *tp63* (Fw TGCTTTGCTAAATTGTGCTGTC; Rv ATTGCCGCTTATGAGAATCAAG), *cavin2a* (Fw GAGCCTTCTCGTGCTAACAAGT; Rv CAGGCATTTCAGTTCAATTTCA), *sox1a* (Fw AATCAAGACCGCGTAAAGAGAC; Rv TTTGGTGGAGTGTTTCTGAATG), *pdyn* (Fw AAGAGAACGCCATACTGAAAGG; Rv GCAGTTACGAATTGCCATGATA), *dlx1a* (Fw AAGGAGGAGAGGTTCGTTTCA; Rv AGTGTGTGTCAGCAGGTGTCTT).

smFISH staining and imaging

Single-molecule FISH probe sets were generated as previously described and coupled to either Atto 647N NHS ester (Millipore Sigma #18373) (*foxg1b*, *cahz*) or Atto 550 NHS ester (Millipore Sigma # 92835) (*ompa*, *rlbp1a*) (Lord et al., 2019). Sectioned larvae were affixed to polylysine-coated #1.5 coverslips, and staining was carried out as previously described (Lord et al., 2019), with each coverslip contained in a well of a plastic 6-well plate. During the probe hybridization step, coverslips were placed upside-down onto a 100µl droplet of probe solution on Parafilm (Farack and Itzkovitz, 2020). Sample mounting was performed as previously described (Lord et al., 2019). Mounted samples were imaged on an Olympus spinSR spinning disk microscope fitted with a UPLAPO 60X/1.5 oil immersion objective using 0.3µm slices.

smFISH image processing

All image processing was performed in Fiji (Schindelin et al., 2012). Rolling-ball background subtraction (radius 25 pixels) was performed on smFISH channels before maximum intensity projections were produced from 30 slices (Figure 1F) or 50 slices (Sup Figure 9) of processed z-stacks. Channels were scaled individually, maximizing for visibility.

Optimization of scGESTALT lineage cassette

In our previous iteration of scGESTALT, the barcode capture rate by scRNA-seq was 6-28%. (Raj et al., 2018b), thereby limiting the density of lineage tree reconstruction. To improve recovery we adapted a different transgenic cassette (Yoshinari et al., 2012) for lineage recording. This cassette has the following modifications compared to our previous recorder: (1) The heat-shock inducible (hsp70l) promoter of the previous version is now replaced with a constitutive ubiquitous promoter (medaka beta-actin) to drive strong widespread expression of the barcode mRNA. Expression of the cassette was confirmed by fluorescence and the signal was more intense than that obtained with the heat shock promoter. Furthermore, this version eliminates the requirement to heat shock edited animals to express the barcode prior to scRNA-seq experiments. (2) We adapted the 3' end of the DsRed open reading frame as a lineage recorder cassette with up to 8 sgRNA target sites positioned next to each other. This vastly improved expression of the construct compared to our previous version where the recording cassette was placed downstream of the DsRed open reading frame. (3) We made library preparation compatible with the 10X Genomics platform.

To generate scGESTALT.2 barcode founder fish, one-cell embryos were injected with zebrafish codon optimized Tol2 mRNA and pT2Olactb:loxP-dsR2-loxP-EGFP vector (gift from Atsushi Kawakami (Yoshinari et al., 2012)). Potential founder fish were screened for widespread DsRed

expression and grown to adulthood. Adult founder transgenic fish were identified by outcrossing to wild type fish and screening clutches of embryos for ubiquitous DsRed expression. Single copy scGESTALT.2 F1 transgenics were identified using qPCR, as described previously (McKenna et al., 2016; Pan et al., 2013; Raj et al., 2018b).

SgRNAs specific to sites 1-8 of the scGESTALT.2 array were generated by in vitro transcription as previously described (Raj et al., 2018a). To initiate early barcode editing, single copy scGESTALT.2 F1 male transgenic adults were crossed to wildtype female adults and one-cell embryos were injected with 1.5 nl of Cas9 protein (NEB) and sgRNAs 1-8 in salt solution (8 µM Cas9, 100 ng/µl pooled sgRNAs, 50 mM KCl, 3 mM $MgCl_2$, 5 mM Tris HCl pH 8.0, 0.05% phenol red). Since editing results in loss of DsRed signal, transgenic animals were distinguished from wild type animals by amplifying the scGESTALT.2 barcode by PCR using genomic DNA from the tail fin at 15 dpf. In the experiments presented in this study, early lineage decisions were barcoded by injecting reagents at the one-cell stage. It is worth noting that the scGESTALT.2 barcode can be readily paired with a two-step barcoding protocol. This would require the establishment of a second stable transgenic line for in vivo expression of Cas9 and a subset of sgRNAs matching the target sequences of the new barcode cassette to enable sequential barcoding at early and late stages. Such a line can be established using a similar step-by-step guidance that is detailed in (Raj et al., 2018a).

Processing of samples for scRNA-seq time course

Wild type embryos (12 hpf, 14 hpf, 16 hpf, 18 hpf, 20 hpf, 24 hpf, 36 hpf) and larvae (2 dpf, 3 dpf, 5 dpf, 8 dpf) were used for scRNA-seq analysis. Samples for 15 dpf had a mix of wild type and barcode edited larvae. Two of the 15 dpf samples consisted of only eye cells (no brain). Embryos from 12 hpf to 36 hpf were first de-chorionated by incubating in 1 mg/ml pronase (Sigma-Aldrich) at 28 C for 6-7 min until chorions began to blister, and then washed three times in ~200 ml of zebrafish embryo medium (5 mM NaCl, 0.17 mM KCl, 0.33 mM $CaCl_2$, 0.33 mM $MgSO_4$, 0.1% methylene blue) in a glass beaker. Embryos were de-yolked using two pairs of watchmaker forceps, and the heads were chopped just anterior of the spinal cord. All processing steps were done using 100 mm Petri dishes coated with Sylgard (Raj et al., 2018a). Samples from 2 and 3 dpf were processed similarly to the embryos, except they were not de-chorionated as they had hatched out of the chorions. Larvae from 5 dpf to 15 dpf were dissected to remove whole brains and eyes as described previously (Raj et al., 2018a). The following numbers of embryos and larvae were used for each timepoint: 12 hpf – ~20 embryos; 14 hpf – ~20 embryos; 16 hpf – ~18 embryos; 18 hpf – ~18 embryos; 20 hpf – ~30 embryos; 24 hpf – ~30 embryos; 36 hpf – ~15 embryos; 2 dpf – ~30 larvae; 3 dpf – ~30 larvae; 5 dpf – ~25 larvae; 8 dpf – ~ 25 larvae; 15 dpf – ~15 larvae. Tissues were dissociated into single cells using the Papin Dissociation Kit (Worthington) as described previously (Raj et al., 2018a). Cells were resuspended in 50 µl to 150 µl of DPBS (Life Technologies) depending on anticipated amount of material, and counted using a hemocytometer. Samples were run on the 10X Genomics scRNA-seq platform according to the manufacturer's instructions (Single Cell 3' v2 kit). Libraries were processed according to the manufacturer's instructions. Transcriptome libraries were sequenced using NextSeq 75 cycle kits.

scGESTALT.2 library prep

To generate scGESTALT.2 libraries, lineage edited 15 dpf samples post cDNA amplification and prior to fragmentation were split into two halves. One half was processed for transcriptome libraries as instructed by the manufacturer. The other half was processed for lineage libraries as follows. To enrich for scGESTALT.2 lineage barcodes, 5 µl of the whole transcriptome cDNA was PCR amplified using Phusion polymerase (NEB) and 10XPCR1_F (CTACACGACGCTCTT CCGATCT) and GP10X2_R (GTGACTGGAGTTCAGACGTGTGCTCTTCCGATCT GCTGCTTC ATCTACAAGGTGAAG). The reaction (98 C, 30 s; [98 C, 10 s; 67 C, 25 s; 72 C, 30 s] x 14-15 cycles; 72 C, 2 min) was cleaned up with 0.6X AMPure beads and eluted in 20 ul EB buffer (Omega). Finally, adapters and sample indexes were incorporated in another PCR reaction using Phusion polymerase and 10XP5Part1long (AATGATACGGCGACCACCGA GATCTACACTCTTTCC CTACACGACGCTCTTCCGATCT) and 10XP7Part2Ax (CAAGCAGAAGACGGCATACGAGAT-xxxxxxxx-GTGACTGGAGTTCAGACGTGT), where x represents index bases. These include A1: GGTTTACT; A2: TTTCATGA; A3: CAGTACTG; A4: TATGATTC. Thus, up to 4 scGESTALT.2 samples were multiplexed in a sequencing run. Libraries were sequenced using MiSeq 300 cycle kits and 20% PhiX spike-in. Sequencing parameters: Read1 250 cycles, Read2 14 cycles, Index1 8 cycles, Index2 8 cycles. Standard sequencing primers were used.

Bioinformatic processing of raw sequencing data and cell type clustering analysis

Transcriptome sequencing data were processed using Cell Ranger 2.1.0 according to the manufacturer's guidelines. scGESTALT.2 sequencing data were processed with a custom pipeline (https://github.com/aaronmck/SC_GESTALT) as previously described (Raj et al., 2018b). The scGESTALT.2 barcode for each cell was matched to its corresponding cell type (tSNE cluster membership) assignment using the cell identifier introduced during transcriptome capture. Cells with fewer than 500 expressed genes, greater than 9% mitochondrial content or very high numbers of UMIs and gene counts that were outliers of a normal distribution (likely doublets/multiplets) were removed from further analysis. Clustering analysis was performed using the Seurat v2.3.4 package (Butler et al., 2018) as described previously (Raj et al., 2018b). For Figure 3 and Sup Figure 2, we selected the list of transcription factors, neuropeptides and their receptors, and genes involved in neuron electrophysiology from our enriched marker analysis and previous literature (R. Chen et al., 2017; Pandey et al., 2018; Tiklová et al., 2019; Zeisel et al., 2018).

Construction of lineage trees from GESTALT barcodes.

All unique barcodes were then encoded into an event matrix and weights file, as described previously (McKenna et al., 2016; Raj et al., 2018b), and were processed using PHYLIP mix with Camin-Sokal maximum parsimony (Felsenstein, 1989). Individual cells were then grafted onto the leaves matching their barcode sequence. After the subtrees were attached, we repeatedly eliminated unsupported internal branching by recursively pruning parent-child nodes that had identical barcodes. Cell annotations are then added to the corresponding leaves. The resulting tree was converted to a JSON object, annotated with cluster membership, and visualized with custom tools using the D3 software framework.

Lineage segregation analysis between cell types

We combined all barcodes obtained from 4 fish. For our analysis, we only considered barcodes with at least two cells, and we only analyzed cell types with at least 4 barcodes. To test segregation between any two cell types/clusters, we first retrieved all barcodes that were present in at least one of the two cell types. Then, we split these barcodes into two categories: “shared barcode” or “specific barcode”. A shared barcode was defined as one that contains cells from both cell types. In contrast, a specific barcode was defined as one that only contains cells from one of the two cell types. Our null hypothesis is that the two cell types come from the same ancestor at the time of Cas9 editing. Thus, we asked whether the number of observed specific barcodes can be explained by chance under the null hypothesis. If it cannot be explained by chance, it indicates that the two cell types have segregated.

To do so, we performed a randomization test as below:

1. We generated a pool of cells. The size of the pool is the total number of cells from the two cell types. The ratio of the two cell types in the pool is equal to the ratio observed in the real data. Under the null hypothesis, the pool of cells come from the same ancestor, so they would share the same barcode.
2. For each barcode, we randomly sampled the same number of cells of this barcode from the pool of cells.
3. We repeated this for all the barcodes, and then calculated the number of barcodes that only contain one cell type (i.e. “specific barcode”).
4. We repeated steps 2 and 3 5000 times.
5. We calculated how many times (for example n times) the number of specific barcodes from the random sampling process is greater than or equal to the number of specific barcodes from the real data.
6. The probability that the number of specific barcodes can be explained by chance under the null hypothesis is n/5000.
7. If the probability < 0.01 (*pvalue* < 0.01), we rejected the null hypothesis.

Next, for each cell type we split its corresponding pairwise comparison cell types into two categories: “with segregation” or “other”. For the “other” category, we considered two interpretations. First, it could signify that there is no segregation between the two cell types. Second, it could suggest that we did not recover enough cells with barcode information, such that there is not enough power to detect lineage segregation (low sampling). To distinguish between the two scenarios, for each cell type in the two categories, we calculated the ratio between the number of cells with barcodes and the number of all cells from scRNA-seq. If the ratio of one cell type from the “other” category is greater than or equal to the smallest ratio from the first category (“with segregation”), it indicates this cell type did not have low sampling issues. Thus, it supports the interpretation that there is no segregation between the queried cell types. Otherwise, we assign the cell type pair as “undefined” (i.e. insufficient sampling power to query lineage segregation).

Granule cell analysis

For each progenitor cell type, we used barcodes that did not include any cells from the other nine progenitor cell types. The Jaccard Index between each progenitor cell type and granule cell was calculated as below:

$$\text{Jaccard Index} = \frac{\text{the number of shared barcodes between the two cell types}}{\text{the number of barcodes in either cell type}}$$

Analyzing dampened spatial correlations in progenitors

Progenitors were isolated by subsetting the data to include clusters expressing markers such as *sox19a*, *her* genes, *pcna*, *mki67*, *fabp7a*, *gfap*, *id1*, etc (Supplementary Table). Cells from 12 hpf – 18 hpf were considered embryonic progenitors, cells from 20 hpf – 3 dpf were considered intermediate progenitors, and cells from 5 dpf – 15 dpf were considered larval progenitors. Variable genes were calculated for embryonic, intermediate and larval progenitors separately using the FindVariableGenes function from Seurat v2.3.4 with parameters: *x.low.cutoff* = 0.015, *x.high.cutoff* = 3, *y.cutoff* = 0.7. Then, a list of 79 transcription factors with known spatial signatures was assembled by consulting previously described histological analysis (ZFIN) together with those that were identified as gene markers for neuronal clusters in our dataset.. Separately in the three progenitor groups, the pairwise Pearson correlation was calculated pairwise between all genes detected as variable in each progenitor group. For several thresholds between 0.2–0.8, the number of genes that correlated more strongly than the threshold with any of the 79 spatial transcription factors (excluding self-correlation) were determined. The strongest correlations were observed in the embryonic population, followed by the intermediate population, and for any threshold, more genes correlated with the spatial TFs in the embryonic progenitors than the larval progenitors.

Construction and analysis of branching transcriptional trajectories using URD

We built branching transcriptional trajectories from cells of the retina and hypothalamus to determine the molecular events that occur as cells diversify and differentiate in these tissues. First, cells from the retina and hypothalamus were isolated from each stage by determining clusters that belonged to these tissues by expression of marker genes.

Determination of variable genes

For URD trajectory analyses, a more restrictive set of variable genes was calculated on each subset of the data, as previously described (Farrell et al., 2018; Pandey et al., 2018) using the URD *findVariableGenes* function, with parameter *diffCV.cutoff* = 0.3. Briefly, a curve was fit that related each gene's coefficient of variation to its mean expression level and represents the expected coefficient of variation resulting from technical noise, given a gene's mean expression value; genes with much higher coefficients of variation likely encode biological variability and were used downstream.

Removal of outliers

Poorly connected outliers can disrupt diffusion map calculation and so were removed from the data. A *k*-nearest neighbor network was calculated between cells (Euclidean distance in variable genes) with 100 nearest neighbors. Cells were then removed based on either unusually high distance to their nearest neighbor or unusually high distance to their 20th nearest neighbor, given

their distance to their nearest neighbor using the URD function *knnOutliers* (retina: *x.max* = 40, *slope.r* = 1.05, *int.r* = 4.3, *slope.b* = 0.75, *int.b* = 11.5; hypothalamus: *x.max* = 40, *slope.r* = 1.1, *int.r* = 3, *slope.b* = 0.66, *int.b* = 11.5).

Removal of doublets by NMF modules

To remove putative cell doublets (i.e. where two cells are encapsulated into a single droplet and processed as one cell), which can disrupt trajectory relationships, we removed cells that expressed multiple NMF (non-negative matrix factorization) modules characteristic of different expression programs, as previously described (Siebert et al., 2019). NMF modules were computed using a previously published NMF framework (https://github.com/YiqunW/NMF) (Farrell et al., 2018). The analysis was performed on log-normalized read count data for a set of variable genes using the *run_nmf.py* script with the following parameters: -rep 5 -scl “false” -miter 10000 -perm True -run_perm True -tol 1e-6 -a 2 -init “random” -analyze True. Several *k* parameters were evaluated for each tissue, and k was chosen to maximize the number of modules, while minimizing the proportion of modules defined primarily by a single gene (retina, *k* = 45; hypothalamus, *k* =). Modules were used downstream that (a) had a ratio between their top-weighted and second-highest weighted gene of < 5, and (b) exhibited a strong cell-type signature, as determined by plotting on a UMAP representation and looking for spatial restriction. Pairs of modules that were appropriate for using to remove doublets (and that did not define transition states) were determined using the URD function *NMFDoubletsDefineModules* with parameters *module.thresh.high* = 0.4, and *module.thresh.low* = 0.15. Putative doublets were identified using the URD function *NMFDoubletsDetermineCells* with parameters *frac.overlap.max* = 0.03, *frac.overlap.diff.max* = 0.1, *module.expressed.thresh* = 0.33 and were then removed.

Choice of root and tips

Branching transcriptional trajectories in the retina and hypothalamus were constructed using URD 1.1.1 (Farrell 2018). Briefly, cells from the first stage of the time course (12 hpf) were selected as the ‘root’ or starting point for the tree. Terminal cell types comprised the clusters at 15 dpf from these tissues, with the exception of clusters that were clearly progenitor or precursors based on known gene expression (retina: 29, 39, 43). Additionally, in the retina, one cluster (96) was excluded because it did not seem that any related cell types had been recovered in previous stages.

Construction of branching transcriptional trajectories

A diffusion map was calculated using *destiny* (Haghverdi et al., 2015; 2016), using 140 (retina) or 100 (hypothalamus) nearest neighbors (approximately the square root of the number of cells in the data), and with a globally-defined sigma of 14 (retina) or 8 (hypothalamus) — slightly smaller than the suggested sigma from *destiny*. Pseudotime was then computed using the simulated ‘flood’ procedure previously described (Farrell et al., 2018), using the following parameters: *n* = 100*, minimum.cells.flooded* = 2. Biased random walks were performed to determine the cells visited from each terminal population in the data as previously described (Farrell et al., 2018), using the following parameters: *optimal.cells.forward* = 40*, max.cells.back* = 80*, n.per.tip* = 50000*, end.visits* = 1. The branching tree was then constructed using URD’s *buildTree* function with the following parameters: *divergence.method* = "ks" (hypothalamus) or *divergence.method* =

"preference" (retina), *save.all.breakpoint.info* = TRUE, *cells.per.pseudotime.bin* = 40, *bins.per.pseudotime.window* = 5, *p.thresh* = 0.0001 (hypothalamus) or , *p.thresh* = 0.01 (retina), and *min.cells.per.segment* = 10. The resulting trees were then evaluated using known marker genes and branch regulators.

Finding genes that vary during differentiation

Genes were selected for inclusion in gene cascades based on their differential expression relative to other cell types in the tissue. See the Supplementary Analysis for the full set of commands used. Within each tissue, cells were first compared in large populations that defined major cell types (retina: cone bipolar cells, photoreceptors, amacrine cells, retinal ganglion cells, horizontal cells, Muller glia, retinal pigmented epithelium; hypothalamus: *prdx1*+ neurons, *pdyn*+ neurons, GABAergic *dlx*+ neurons, *nrgna*+ neurons). Comparisons were performed pairwise, and genes were considered differential in a population if they were upregulated compared to at least 2 (hypothalamus) or 3 (retina) other groups. Genes were considered differentially expressed based on their expression fold-change (retina: ≥1.32-fold change, hypothalamus: ≥ 1.41-fold change) and their performance as a precision-recall classifier for the two cell populations compared (≥ 1.1-fold better than a random classifier). Additionally, the *aucprTestAlongTree* function from URD was used to select additional genes by performing pairwise comparisons, starting from a terminal cell type and comparing at each branchpoint along the way, back to the root (Farrell et al., 2018). Genes were selected based on expression fold-change between branchpoints (hypothalamus: ≥1.74-fold upregulated; hypothalamus, populations with small cell numbers (GABAergic *dlx*+ cells): ≥1.51-fold upregulated; retina: ≥1.32-fold upregulated), their function as a precision-recall classifier between branchpoints (hypothalamus: ≥1.2-fold better than a random classifier; hypothalamus, populations with small cell numbers (GABAergic *dlx*+ cells): ≥1.15-fold better than a random classifier; retina: ≥1.1-fold better than a random classifier), their function as a precision recall classifier globally (i.e. between the entire trajectory leading to a cell type and the rest of the tissue): ≥1.03-fold better than a random classifier, and their upregulation globally (i.e. between the entire trajectory leading to a cell type and the rest of the tissue): ≥1.07-fold upregulated. Mitochondrial, ribosomal, and tandem duplicated genes were excluded. Cells were ordered according to pseudotime, split into groups of at least 25 cells that differ at least 0.005 in pseudotime, and the mean expression was determined with a 5-group moving window. A spline curve was fit to the mean expression vs. pseudotime relationship of selected genes, using the *smooth.spline* function from R's *stats* package, with the parameter *spar* = 0.5. Genes were then sorted according to their peak expression in pseudotime, normalized to their max expression observed in the tissue, and plotted on a heatmap.

Analyzing progenitor populations

To determine whether retinal progenitors mature transcriptionally over time, we looked for genes that were differentially expressed between young and old progenitors. We chose cells that occupied the same region of the URD tree from either early (24 / 36 hpf) or late (15 dpf) stages. We looked for genes that were differentially expressed in 15 dpf progenitors that: (1) were 1.1-fold better as a precision-recall classifier than random, (2) changed ≥1.32-fold in expression, (3) were expressed in at least 20% of progenitors, (4) had a mean expression value ≥ 0.8, and (5)

were more differentially expressed than equally sized cell populations chosen at random at least 99% of the time.

To determine whether cells were found in progenitor or precursor states long-term, we first defined progenitor and precursor states by cells' assignment in the URD tree, cross-referenced with the expression of progenitor / precursor markers. We then determined how many cells from different stages fell into each of these different states.

ACKNOWLEDGEMENTS

We thank members of the Schier lab for discussion and advice, the Bauer Core Facility (Harvard) and the Molecular Biology Core Facility (Dana Farber Cancer Institute) for sequencing services, the Harvard zebrafish facility staff for technical support, and the Imaging Core Facility of the Biozentrum for microscopy resources. We thank M. Shafer and J. Gagnon for comments on the manuscript, and H. Boije for comments on retinal lineages. This work was supported by a postdoctoral fellowship from the Canadian Institutes of Health Research and 1K99HD098298 to B.R., 1K99HD091291 to J.A.F., R00HG010152 to A.M., R01HD85905, DP1HD094764, ERC 834788, an Allen Discovery Center grant, and a McKnight Foundation Technological Innovations in Neuroscience Award to A.F.S.

AUTHOR CONTRIBUTIONS

B.R. and A.F.S. conceived and designed the study. B.R., J.A.F., J.L., J.E.K, and A.F.S. interpreted the data. B.R., J.A.F. and A.F.S. wrote the manuscript. B.R. and J.L.L. generated transgenic lines. B.R. performed scRNA-seq and scGESTALT experiments and data processing. J.L. analyzed scGESTALT data with assistance from B.R. and J.E.K. L.Y.D. generated violin plots of neuron subtype diversity. J.N.A. performed chromogenic in situs. A.N.C. and J.E.K. performed smFISH experiments. J.A.F. performed URD trajectory analysis with assistance from B.R. A.M. generated lineage trees. Đ.R. developed the R Shiny app for scRNA-seq data exploration.

REFERENCES

Allende, M.L., Weinberg, E.S., 1994. The expression pattern of two zebrafish achaete-scute homolog (ash) genes is altered in the embryonic brain of the cyclops mutant. Dev. Biol. 166, 509–530. doi:10.1006/dbio.1994.1334

Bendall, S.C., Davis, K.L., Amir, E.-A.D., Tadmor, M.D., Simonds, E.F., Chen, T.J., Shenfeld, D.K., Nolan, G.P., Pe'er, D., 2014. Single-cell trajectory detection uncovers progression and regulatory coordination in human B cell development. Cell 157, 714–725. doi:10.1016/j.cell.2014.04.005

Boije, H., Rulands, S., Dudczig, S., Simons, B.D., Harris, W.A., 2015. The Independent Probabilistic Firing of Transcription Factors: A Paradigm for Clonal Variability in the Zebrafish Retina. Developmental Cell 34, 532–543. doi:10.1016/j.devcel.2015.08.011

961 Brzezinski, J.A., Lamba, D.A., Reh, T.A., 2010. Blimp1 controls photoreceptor versus bipolar
962 cell fate choice during retinal development. Development 137, 619–629.
963 doi:10.1242/dev.043968
964 Butler, A., Hoffman, P., Smibert, P., Papalexi, E., Satija, R., 2018. Integrating single-cell
965 transcriptomic data across different conditions, technologies, and species. Nat Biotechnol
966 36, 411–420. doi:10.1038/nbt.4096
967 Carter, R.A., Bihannic, L., Rosencrance, C., Hadley, J.L., Tong, Y., Phoenix, T.N., Natarajan, S.,
968 Easton, J., Northcott, P.A., Gawad, C., 2018. A Single-Cell Transcriptional Atlas of the
969 Developing Murine Cerebellum. Current Biology 28, 2910–2920.e2.
970 doi:10.1016/j.cub.2018.07.062
971 Centanin, L., Wittbrodt, J., 2014. Retinal neurogenesis. Development 141, 241–244.
972 doi:10.1242/dev.083642
973 Cepko, C., 2014. Intrinsically different retinal progenitor cells produce specific types of progeny.
974 Nat. Rev. Neurosci. 15, 615–627. doi:10.1038/nrn3767
975 Chen, J., Rattner, A., Nathans, J., 2005. The rod photoreceptor-specific nuclear receptor Nr2e3
976 represses transcription of multiple cone-specific genes. J. Neurosci. 25, 118–129.
977 doi:10.1523/JNEUROSCI.3571-04.2005
978 Chen, R., Wu, X., Jiang, L., Zhang, Y., 2017. Single-Cell RNA-Seq Reveals Hypothalamic Cell
979 Diversity. Cell Rep 18, 3227–3241. doi:10.1016/j.celrep.2017.03.004
980 Clark, B.S., Stein-O'Brien, G.L., Shiau, F., Cannon, G.H., Davis-Marcisak, E., Sherman, T.,
981 Santiago, C.P., Hoang, T.V., Rajaii, F., James-Esposito, R.E., Gronostajski, R.M., Fertig,
982 E.J., Goff, L.A., Blackshaw, S., 2019. Single-Cell RNA-Seq Analysis of Retinal Development
983 Identifies NFI Factors as Regulating Mitotic Exit and Late-Born Cell Specification. Neuron
984 102, 1111–1126.e5. doi:10.1016/j.neuron.2019.04.010
985 Cosacak, M.I., Bhattarai, P., Reinhardt, S., Petzold, A., Dahl, A., Zhang, Y., Kizil, C., 2019.
986 Single-Cell Transcriptomics Analyses of Neural Stem Cell Heterogeneity and Contextual
987 Plasticity in a Zebrafish Brain Model of Amyloid Toxicity. Cell Rep 27, 1307–1318.e3.
988 doi:10.1016/j.celrep.2019.03.090
989 Cueva, J.G., Haverkamp, S., Reimer, R.J., Edwards, R., Wässle, H., Brecha, N.C., 2002.
990 Vesicular gamma-aminobutyric acid transporter expression in amacrine and horizontal cells.
991 J. Comp. Neurol. 445, 227–237. doi:10.1002/cne.10166
992 Delile, J., Rayon, T., Melchionda, M., Edwards, A., Briscoe, J., Sagner, A., 2019. Single cell
993 transcriptomics reveals spatial and temporal dynamics of gene expression in the developing
994 mouse spinal cord. Development 146, dev173807. doi:10.1242/dev.173807
995 Deniz, S., Wersinger, E., Schwab, Y., Mura, C., Erdelyi, F., Szabó, G., Rendon, A., Sahel, J.-A.,
996 Picaud, S., Roux, M.J., 2011. Mammalian retinal horizontal cells are unconventional
997 GABAergic neurons. J. Neurochem. 116, 350–362. doi:10.1111/j.1471-4159.2010.07114.x
998 Farack, L., Itzkovitz, S., 2020. Protocol for Single-Molecule Fluorescence In Situ Hybridization
999 for Intact Pancreatic Tissue. STAR Protocols 1, 100007. doi:10.1016/j.xpro.2019.100007
1000 Farnsworth, D.R., Saunders, L.M., Miller, A.C., 2020. A single-cell transcriptome atlas for
1001 zebrafish development. Dev. Biol. 459, 100–108. doi:10.1016/j.ydbio.2019.11.008
1002 Farrell, J.A., Wang, Y., Riesenfeld, S.J., Shekhar, K., Regev, A., Schier, A.F., 2018. Single-cell
1003 reconstruction of developmental trajectories during zebrafish embryogenesis. Science 360,
1004 eaar3131. doi:10.1126/science.aar3131
1005 Felsenstein, J., 1989. PHYLIP - Phylogeny Inference Package (Version 3.2). Cladistics, Vol. 5
1006 (1989), pp. 164-166 5, 164–166.
1007 Fischer, A.J., Bongini, R., Bastaki, N., Sherwood, P., 2011. The maturation of photoreceptors in
1008 the avian retina is stimulated by thyroid hormone. Neuroscience 178, 250–260.
1009 doi:10.1016/j.neuroscience.2011.01.022

1010 Gibbs, H.C., Chang-Gonzalez, A., Hwang, W., Yeh, A.T., Lekven, A.C., 2017. Midbrain-
1011 Hindbrain Boundary Morphogenesis: At the Intersection of Wnt and Fgf Signaling. Front.
1012 Neuroanat. 11, 64. doi:10.3389/fnana.2017.00064
1013 Guo, Q., Li, J.Y.H., 2019. Defining developmental diversification of diencephalon neurons
1014 through single cell gene expression profiling. Development 146, dev174284.
1015 doi:10.1242/dev.174284
1016 Haghverdi, L., Buettner, F., Theis, F.J., 2015. Diffusion maps for high-dimensional single-cell
1017 analysis of differentiation data. Bioinformatics 31, 2989–2998.
1018 doi:10.1093/bioinformatics/btv325
1019 Haghverdi, L., Büttner, M., Wolf, F.A., Buettner, F., Theis, F.J., 2016. Diffusion pseudotime
1020 robustly reconstructs lineage branching. Nat. Methods 13, 845–848.
1021 doi:10.1038/nmeth.3971
1022 He, J., Zhang, G., Almeida, A.D., Cayouette, M., Simons, B.D., Harris, W.A., 2012. How variable
1023 clones build an invariant retina. Neuron 75, 786–798. doi:10.1016/j.neuron.2012.06.033
1024 Holguera, I., Desplan, C., 2018. Neuronal specification in space and time. Science 362, 176–
1025 180. doi:10.1126/science.aas9435
1026 Hu, Y., Wang, X., Hu, B., Mao, Y., Chen, Y., Yan, L., Yong, J., Dong, J., Wei, Y., Wang, W.,
1027 Wen, L., Qiao, J., Tang, F., 2019. Dissecting the transcriptome landscape of the human
1028 fetal neural retina and retinal pigment epithelium by single-cell RNA-seq analysis. PLoS
1029 Biol. 17, e3000365. doi:10.1371/journal.pbio.3000365
1030 Jessell, T.M., 2000. Neuronal specification in the spinal cord: inductive signals and
1031 transcriptional codes. Nat. Rev. Genet. 1, 20–29. doi:10.1038/35049541
1032 Kim, D.W., Washington, P.W., Wang, Z.Q., Lin, S., Sun, C., Jiang, L., Blackshaw, S., 2019.
1033 Single cell RNA-Seq analysis identifies molecular mechanisms controlling hypothalamic
1034 patterning and differentiation. bioRxiv 83, 657148. doi:10.1101/657148
1035 Korzh, V., Sleptsova, I., Liao, J., He, J., Gong, Z., 1998. Expression of zebrafish bHLH genes
1036 ngn1 and nrd defines distinct stages of neural differentiation. Dev. Dyn. 213, 92–104.
1037 doi:10.1002/(SICI)1097-0177(199809)213:1<92::AID-AJA9>3.0.CO;2-T
1038 Kretzschmar, K., Watt, F.M., 2012. Lineage tracing. Cell 148, 33–45.
1039 doi:10.1016/j.cell.2012.01.002
1040 La Manno, G., Siletti, K., Furlan, A., Gyllborg, D., Vinsland, E., Langseth, C.M., Khven, I.,
1041 Johnsson, A., Nilsson, M., Lönnerberg, P., Linnarsson, S., 2020. Molecular architecture of
1042 the developing mouse brain. bioRxiv 135C, 2020.07.02.184051.
1043 doi:10.1101/2020.07.02.184051
1044 Lange, C., Rost, F., Machate, A., Reinhardt, S., Lesche, M., Weber, A., Kuscha, V., Dahl, A.,
1045 Rulands, S., Brand, M., 2020. Single cell sequencing of radial glia progeny reveals the
1046 diversity of newborn neurons in the adult zebrafish brain. Development 147, dev185595.
1047 doi:10.1242/dev.185595
1048 Lee, S.-K., Pfaff, S.L., 2001. Transcriptional networks regulating neuronal identity in the
1049 developing spinal cord. Nat. Neurosci. 4, 1183–1191. doi:10.1038/nn750
1050 Li, H., Horns, F., Wu, B., Xie, Q., Li, J., Li, T., Luginbuhl, D.J., Quake, S.R., Luo, L., 2017.
1051 Classifying Drosophila Olfactory Projection Neuron Subtypes by Single-Cell RNA
1052 Sequencing. Cell 171, 1206–1220.e22. doi:10.1016/j.cell.2017.10.019
1053 Lord, N.D., Carte, A.N., Abitua, P.B., Schier, A.F., 2019. The pattern of Nodal morphogen
1054 signaling is shaped by co-receptor expression. bioRxiv 125, 2019.12.30.891101.
1055 doi:10.1101/2019.12.30.891101
1056 Ma, J., Shen, Z., Yu, Y.-C., Shi, S.-H., 2017. Neural lineage tracing in the mammalian brain.
1057 Curr. Opin. Neurobiol. 50, 7–16. doi:10.1016/j.conb.2017.10.013
1058 Mayer, C., Hafemeister, C., Bandler, R.C., Machold, R., Batista Brito, R., Jaglin, X., Allaway, K.,
1059 Butler, A., Fishell, G., Satija, R., 2018. Developmental diversification of cortical inhibitory
1060 interneurons. Nature 555, 457–462. doi:10.1038/nature25999

McKenna, A., Findlay, G.M., Gagnon, J.A., Horwitz, M.S., Schier, A.F., Shendure, J., 2016. Whole-organism lineage tracing by combinatorial and cumulative genome editing. Science 353, aaf7907. doi:10.1126/science.aaf7907

McKenzie, J.A.G., Fruttiger, M., Abraham, S., Lange, C.A.K., Stone, J., Gandhi, P., Wang, X., Bainbridge, J., Moss, S.E., Greenwood, J., 2012. Apelin is required for non-neovascular remodeling in the retina. Am. J. Pathol. 180, 399–409. doi:10.1016/j.ajpath.2011.09.035

Mission, J.P., Takahashi, T., Caviness, V.S., 1991. Ontogeny of radial and other astroglial cells in murine cerebral cortex. Glia 4, 138–148. doi:10.1002/glia.440040205

Moens, C.B., Prince, V.E., 2002. Constructing the hindbrain: insights from the zebrafish. Dev. Dyn. 224, 1–17. doi:10.1002/dvdy.10086

Mueller, T., Wullimann, M.F., 2003. Anatomy of neurogenesis in the early zebrafish brain. Brain Res. Dev. Brain Res. 140, 137–155. doi:10.1016/s0165-3806(02)00583-7

Navajas Acedo, J., Voas, M.G., Alexander, R., Woolley, T., Unruh, J.R., Li, H., Moens, C., Piotrowski, T., 2019. PCP and Wnt pathway components act in parallel during zebrafish mechanosensory hair cell orientation. Nat Comms 10, 3993–17. doi:10.1038/s41467-019-12005-y

Nowakowski, T.J., Bhaduri, A., Pollen, A.A., Alvarado, B., Mostajo-Radji, M.A., Di Lullo, E., Haeussler, M., Sandoval-Espinosa, C., Liu, S.J., Velmeshev, D., Ounadjela, J.R., Shuga, J., Wang, X., Lim, D.A., West, J.A., Leyrat, A.A., Kent, W.J., Kriegstein, A.R., 2017. Spatiotemporal gene expression trajectories reveal developmental hierarchies of the human cortex. Science 358, 1318–1323. doi:10.1126/science.aap8809

Ogawa, Y., Shiraki, T., Kojima, D., Fukada, Y., 2015. Homeobox transcription factor Six7 governs expression of green opsin genes in zebrafish. Proceedings of the Royal Society B: Biological Sciences 282, 20150659. doi:10.1098/rspb.2015.0659

Pan, Y.A., Freundlich, T., Weissman, T.A., Schoppik, D., Wang, X.C., Zimmerman, S., Ciruna, B., Sanes, J.R., Lichtman, J.W., Schier, A.F., 2013. Zebrabow: multispectral cell labeling for cell tracing and lineage analysis in zebrafish. Development 140, 2835–2846. doi:10.1242/dev.094631

Pandey, S., Shekhar, K., Regev, A., Schier, A.F., 2018. Comprehensive Identification and Spatial Mapping of Habenular Neuronal Types Using Single-Cell RNA-Seq. 28, 1052–1065.e7. doi:10.1016/j.cub.2018.02.040

Raj, B., Gagnon, J.A., Schier, A.F., 2018a. Large-scale reconstruction of cell lineages using single-cell readout of transcriptomes and CRISPR-Cas9 barcodes by scGESTALT. Nat Protoc 13, 2685–2713. doi:10.1038/s41596-018-0058-x

Raj, B., Wagner, D.E., McKenna, A., Pandey, S., Klein, A.M., Shendure, J., Gagnon, J.A., Schier, A.F., 2018b. Simultaneous single-cell profiling of lineages and cell types in the vertebrate brain. Nat Biotechnol 36, 442–450. doi:10.1038/nbt.4103

Rheaume, B.A., Jereen, A., Bolisetty, M., Sajid, M.S., Yang, Y., Renna, K., Sun, L., Robson, P., Trakhtenberg, E.F., 2018. Single cell transcriptome profiling of retinal ganglion cells identifies cellular subtypes. Nat Comms 9, 1–17. doi:10.1038/s41467-018-05134-3

Rosenberg, A.B., Roco, C.M., Muscat, R.A., Kuchina, A., Sample, P., Yao, Z., Graybuck, L.T., Peeler, D.J., Mukherjee, S., Chen, W., Pun, S.H., Sellers, D.L., Tasic, B., Seelig, G., 2018. Single-cell profiling of the developing mouse brain and spinal cord with split-pool barcoding. Science 360, 176–182. doi:10.1126/science.aam8999

Rulands, S., Iglesias-Gonzalez, A.B., Boije, H., 2018. Deterministic fate assignment of Müller glia cells in the zebrafish retina suggests a clonal backbone during development. Eur. J. Neurosci. 48, 3597–3605. doi:10.1111/ejn.14257

Sagner, A., Briscoe, J., 2019. Establishing neuronal diversity in the spinal cord: a time and a place. Development 146, dev182154. doi:10.1242/dev.182154

Satoh, S., Tang, K., Iida, A., Inoue, M., Kodama, T., Tsai, S.Y., Tsai, M.-J., Furuta, Y., Watanabe, S., 2009. The spatial patterning of mouse cone opsin expression is regulated by

bone morphogenetic protein signaling through downstream effector COUP-TF nuclear receptors. J. Neurosci. 29, 12401–12411. doi:10.1523/JNEUROSCI.0951-09.2009
Schindelin, J., Arganda-Carreras, I., Frise, E., Kaynig, V., Longair, M., Pietzsch, T., Preibisch, S., Rueden, C., Saalfeld, S., Schmid, B., Tinevez, J.-Y., White, D.J., Hartenstein, V., Eliceiri, K., Tomancak, P., Cardona, A., 2012. Fiji: an open-source platform for biological-image analysis. Nat. Methods 9, 676–682. doi:10.1038/nmeth.2019
Schmechel, D.E., Rakic, P., 1979. A Golgi study of radial glial cells in developing monkey telencephalon: morphogenesis and transformation into astrocytes. Anat. Embryol. 156, 115–152. doi:10.1007/BF00300010
Schmidt, R., Strähle, U., Scholpp, S., 2013. Neurogenesis in zebrafish - from embryo to adult. Neural Development 8, 3. doi:10.1186/1749-8104-8-3
Shen, Y.-C., Raymond, P.A., 2004. Zebrafish cone-rod (crx) homeobox gene promotes retinogenesis. Dev. Biol. 269, 237–251. doi:10.1016/j.ydbio.2004.01.037
Siebert, S., Farrell, J.A., Cazet, J.F., Abeykoon, Y., Primack, A.S., Schnitzler, C.E., Juliano, C.E., 2019. Stem cell differentiation trajectories in Hydra resolved at single-cell resolution. Science 365, eaav9314. doi:10.1126/science.aav9314
Solek, C.M., Feng, S., Perin, S., Weinschutz Mendes, H., Ekker, M., 2017. Lineage tracing of dlx1a/2a and dlx5a/6a expressing cells in the developing zebrafish brain. Dev. Biol. 427, 131–147. doi:10.1016/j.ydbio.2017.04.019
Stigloher, C., Chapouton, P., Adolf, B., Bally-Cuif, L., 2008. Identification of neural progenitor pools by E(Spl) factors in the embryonic and adult brain. Brain Res. Bull. 75, 266–273. doi:10.1016/j.brainresbull.2007.10.032
Tambalo, M., Mitter, R., Wilkinson, D.G., 2020. A single cell transcriptome atlas of the developing zebrafish hindbrain. Development 147, dev184143. doi:10.1242/dev.184143
Tambalo, M., Mitter, R., Wilkinson, D.G., 2019. A single cell transcriptome atlas of the developing zebrafish hindbrain. bioRxiv 124, 745141. doi:10.1101/745141
Than-Trong, E., Bally-Cuif, L., 2015. Radial glia and neural progenitors in the adult zebrafish central nervous system. Glia 63, 1406–1428. doi:10.1002/glia.22856
Thisse, B., Heyer, V., Lux, A., Alunni, V., Degrave, A., Seiliez, I., Kirchner, J., Parkhill, J.-P., Thisse, C., 2004. Spatial and temporal expression of the zebrafish genome by large-scale in situ hybridization screening. Methods Cell Biol. 77, 505–519. doi:10.1016/s0091-679x(04)77027-2
Tiklová, K., Björklund, Å.K., Lahti, L., Fiorenzano, A., Nolbrant, S., Gillberg, L., Volakakis, N., Yokota, C., Hilscher, M.M., Hauling, T., Holmström, F., Joodmardi, E., Nilsson, M., Parmar, M., Perlmann, T., 2019. Single-cell RNA sequencing reveals midbrain dopamine neuron diversity emerging during mouse brain development. Nat Comms 10, 581–12. doi:10.1038/s41467-019-08453-1
Trapnell, C., Cacchiarelli, D., Grimsby, J., Pokharel, P., Li, S., Morse, M., Lennon, N.J., Livak, K.J., Mikkelsen, T.S., Rinn, J.L., 2014. The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. Nat Biotechnol 32, 381–386. doi:10.1038/nbt.2859
Viczian, A.S., Vignali, R., Zuber, M.E., Barsacchi, G., Harris, W.A., 2003. XOtx5b and XOtx2 regulate photoreceptor and bipolar fates in the Xenopus retina. Development 130, 1281–1294. doi:10.1242/dev.00343
Wagner, D.E., Weinreb, C., Collins, Z.M., Briggs, J.A., Megason, S.G., Klein, A.M., 2018. Single-cell mapping of gene expression landscapes and lineage in the zebrafish embryo. Science 360, 981–987. doi:10.1126/science.aar4362
Wamsley, B., Fishell, G., 2017. Genetic and activity-dependent mechanisms underlying interneuron diversity. Nat. Rev. Neurosci. 18, 299–309. doi:10.1038/nrn.2017.30
Wilson, S.W., Brand, M., Eisen, J.S., 2002. Patterning the zebrafish central nervous system. Results Probl Cell Differ 40, 181–215.

Wilson, S.W., Rubenstein, J.L., 2000. Induction and dorsoventral patterning of the telencephalon. Neuron 28, 641–651. doi:10.1016/s0896-6273(00)00171-9
Woo, K., Fraser, S.E., 1995. Order and coherence in the fate map of the zebrafish nervous system. Development 121, 2595–2609.
Woodworth, M.B., Girskis, K.M., Walsh, C.A., 2017. Building a lineage from single cells: genetic techniques for cell lineage tracking. Nat. Rev. Genet. 18, 230–244. doi:10.1038/nrg.2016.159
Xie, Y., Dorsky, R.I., 2017. Development of the hypothalamus: conservation, modification and innovation. Development 144, 1588–1599. doi:10.1242/dev.139055
Xu, B., Tang, X., Jin, M., Zhang, H., Du, L., Yu, S., He, J., 2020. Unifying developmental programs for embryonic and postembryonic neurogenesis in the zebrafish retina. Development 147, dev185660. doi:10.1242/dev.185660
Yoshinari, N., Ando, K., Kudo, A., Kinoshita, M., Kawakami, A., 2012. Colored medaka and zebrafish: Transgenics with ubiquitous and strong transgene expression driven by the medaka β-actinpromoter. Develop. Growth Differ. 54, 818–828. doi:10.1073/pnas.94.8.3789
Zeisel, A., Hochgerner, H., Lönnerberg, P., Johnsson, A., Memic, F., van der Zwan, J., Häring, M., Braun, E., Borm, L.E., La Manno, G., Codeluppi, S., Furlan, A., Lee, K., Skene, N., Harris, K.D., Hjerling-Leffler, J., Arenas, E., Ernfors, P., Marklund, U., Linnarsson, S., 2018. Molecular Architecture of the Mouse Nervous System. Cell 174, 999–1014.e22. doi:10.1016/j.cell.2018.06.021
Zhong, S., Zhang, S., Fan, X., Wu, Q., Yan, L., Dong, J., Zhang, H., Li, L., Le Sun, Pan, N., Xu, X., Tang, F., Zhang, J., Qiao, J., Wang, X., 2018. A single-cell RNA-seq survey of the developmental landscape of the human prefrontal cortex. Nature 555, 524–528. doi:10.1038/nature25980

Figure 1. Developmental compendium of zebrafish head and brain cell types

A. Schematic of the developmental stages profiled. Red hatched line represents head regions that were selected for enrichment of brain cells in early development. Samples from 5 to 15 dpf were dissected to obtain brain and eye specifically. h, hours post fertilization; d, days post fertilization

B. Schematic of scRNA-seq using 10X Genomics platform.

C. Cell type heterogeneity within each stage. Clusters at each stage were assigned to a region or tissue type based on known markers and color coded to reflect their classification. tSNE implementations: Barnes-Hut (12h to 3d), Fourier transform (5d and 15d).

D. In situ hybridization for novel markers in the trigeminal placode at 12 hpf. *klf17* is expressed on the anterior polster and ventral mesoderm, delineating the border of the embryo. Trigeminal ganglia markers *ptgs2a*, *tp63* and *sdpra* (*cavin2a*) are expressed bilaterally (asterisks) posterior to the eye. Eyes are delineated by dotted lines. A: Anterior; P: Posterior. Scale bar equals 100 µm.

E. In situ hybridization validation of novel marker *sox1a* in the hypothalamus at 2 dpf. Top panels, lateral view of brain; Bottom panels, ventral view of brain. *dlx1a* and *pdyn* are known hypothalamus. Eyes are delineated by dotted lines. VHyp: Ventral Hypothalamus; TVZ: Telencephalic Ventricular Zone; ADi: Anterior Diencephalon; AFb: Anterior Forebrain; VDi: Ventral Diencephalon; Le: Lens. Scale bar equals 200 µm.

F. smFISH validation of novel marker *ompa* in horizontal cells of the retina at 5 dpf. Left panel, retina section stained with DAPI (grey), pan-retinal *foxg1b* (cyan) and *ompa* (yellow). Strong yellow signal in photoreceptors represent autofluorescence. White box indicates area that was zoomed in for the right panels. Dotted lines indicate the horizontal cell layer. PR, photoreceptor cells; HC, horizontal cells; BC, bipolar cells; AC, amacrine cells; RGC, retinal ganglion cells

Figure 2. Brain cell type diversification from 12 hpf to 15 dpf

A. tSNE plot of 12 hpf dataset. Only clusters corresponding to neural and blood cell types are shown. Inferred identities of each cluster are described.
B. Dot plot of gene expression pattern of select marker genes (columns) for each cluster (row). Dot size indicates the percentage of cells expressing the marker; color represents the average scaled expression level.
C. tSNE plot of 15 dpf dataset. Inferred identities of each cluster are described.
D. Dot plot of gene expression patterns of select marker genes for each cluster. Layout is same as (**B**). Grey box represents generic neuronal and progenitor genes.
tSNE implementations: Barnes-Hut (**A**), Fourier transform (**C**)

Figure 3. Neuron subtype diversity at 15 dpf

A-C. Violin plots of select marker gene expression in identified brain neuron subtypes from 15 dpf. Retina neurons and nascent neurons are omitted from the analysis. Cluster numbers are indicated at the bottom along with their inferred spatial location in the brain. Cluster 76 has unknown spatial location. Detailed cluster descriptions are in Supplementary Table 1 and can be explored interactively in the accompanying app at https://github.com/brlauuu/zf_brain.
A. Expression of transcription factors.
B. Expression of neuropeptides and their receptors.
C. Expression of genes involved in neuron electrophysiology including neurotransmitters, transporters, receptors, and channels.
D. Matrix showing whether neuron subtypes identified at 15 dpf are also detected in earlier larval (5 and 8 dpf) and later juvenile (25 dpf (Raj et al., 2018b)) stages. Clusters were matched across stages by comparing marker gene expression. The cluster number at 15 dpf is shown and an orange circle indicates that the subtype is detected in another stage.

Figure 4. Developmental diversification of neurons and progenitors

A. Area plot of the percentage of dataset at each timepoint corresponding to neural progenitors, neurons, and other cell types. Right panels, Total number of clusters of progenitors and neurons at each stage of brain development.
B. tSNE plot of embryonic, intermediate and larval neural progenitors. All progenitor cells were analyzed together after subsetting from the whole dataset.
C-D. Heatmaps of select gene expression in early embryonic (**C**) and late larval (**D**) brain neural progenitors. Top panel, genes enriched in embryonic progenitors. Bottom panel, genes enriched in larval progenitors. Embryonic progenitors have a strong spatial signature (forebrain, midbrain, hindbrain) and are depleted in genes that distinguish larval progenitor subtypes (**C**). Larval progenitors segregate into non-proliferative and proliferative groups that can be resolved into additional subtypes characterized by expression of various gene combinations (**D**). TF, transcription factor. *pax6a is expressed in multiple regions
E. Heatmap of Pearson correlation values of 79 spatial markers in embryonic, intermediate and larval neural progenitors. Spatial markers were selected based on existing literature. Groups of co-varying genes in the midbrain and forebrain are highlighted with dashed boxes.
F. Plot showing number of highly variable genes that co-vary with any of the selected 79 spatial markers in embryonic and larval progenitors. Co-variation was determined by Pearson correlation, with several thresholds (from stringent to relaxed) displayed along the x-axis.

Figure 5. Optimization of scGESTALT lineage recorder for better barcode recovery

A. Schematic overview of CRISPR-Cas9 lineage recording. Optimized scGESTALT comprises a barcode cassette in the 3'end of DsRed transgene (single copy) and the medaka beta-actin promoter. Embryos are injected with Cas9 protein and DsRed sgRNAs and animals are profiled at 15 dpf by scRNA-seq.
B. Pairwise comparisons using cosine dissimilarity of barcode edit patterns from four (ZF1-4) edited 15 dpf larval brains.

C. Chord diagram of the nature and frequency of deletions within and between target sites. Each colored sector represents a target site. Links between target sites represent inter-site deletions; self-links represent intra-site deletions. Link widths are proportional to the edit frequencies.
D. Type of edit at each target site within the barcode from edited ZF1-4 larval brains.
E. Heat map of lineage relationships between non-retinal and retinal cell types in the eye. All clusters with >3 cells and all barcodes with >1 cell were used to determine if there is enrichment of cell type-specific barcodes across each cluster pair. Blue indicates significant enrichment and lineage segregation. Purple indicates no significant enrichment and no lineage segregation. Grey indicates insufficient sampling power and undefined lineage status. Cluster numbers are indicated (e.g. C45) and either cell type gene markers (e.g. *cldna*$^{+}$) or the exact name of the cell type (e.g. cone bipolar cells) are indicated along the rows. Along the columns, the numbers within the brackets indicate the number of barcodes and number of cells, respectively, for that cluster.
F. Heat map of lineage relationships between brain regions and the retina. Neuron clusters that could be pseudospatially assigned to the each region were used (see Supplementary Table). Analysis, layout and color code are same as in E.
G. Heat map of lineage relationships between neuronal cell types in the forebrain and midbrain. Analysis, layout and color code are same as in E. The brain region each cluster belongs to is indicated (e.g. pallium, hypothalamus), and for clusters where a more precise location could be inferred a gene marker is indicated (e.g. *pitx2*$^{+}$).
H. Heat map of lineage relationships between brain progenitor clusters. Analysis, layout and color code is same as in E. Cell type marker genes are indicated along with the cluster number. URL, upper rhombic lip
I. Bar plot of the proportion (based on Jaccard Index) of granule cell (cerebellum neurons) barcodes that are shared with each brain progenitor cluster. Cluster numbers are the same as in H.

Figure 6. Cell specification trajectories in the retina and hypothalamus

A. UMAP visualization of retinal cell types. Retinal cells (based on clustering analysis) from 12 hpf to 15 dpf were subsetted from the full dataset and analyzed together. Cells are color coded by stage.
B. Cell specification tree of zebrafish retinal development. Trajectories were generated by URD and visualized as a branching tree. Cells are color coded by stage. 12 hpf cells were assigned as the root and 15 dpf differentiated cells were assigned as tips. CBP, cone bipolar cells (6 subtypes are numbered); RGC, retinal ganglion cells; RPE, retinal pigment epithelium
C. Expression of select genes are shown on the retina specification tree.
D. Heat maps of gene expression cascades of photoreceptor cell trajectories and retinal ganglion cell trajectories. Cells were selected based on high expression along trajectories leading to these cell types, compared to expression along opposing branchpoints. Red, high expression. Yellow, low expression
E. UMAP visualization of hypothalamus cell types. Hypothalamus cells (based on clustering analysis) from 12 hpf to 15 dpf were subsetted from the full dataset and analyzed together. Cells are color coded by stage.
F. Cell specification tree of zebrafish hypothalamus development. Trajectories were generated by URD and visualized as a branching tree. Cells are color coded by stage. 12 hpf cells were assigned as the root and 15 dpf differentiated cells were assigned as tips.
G. Expression of select genes are shown on the hypothalamus specification tree.
H. Heat map of gene expression cascade of *nrgna*+ cell trajectories. Red, high expression. Yellow, low expression

Figure 7. Progenitor differences between retina and hypothalamus

Retinal and hypothalamus cells were divided into progenitor (purple), precursor (orange), and differentiated (blue) cells, as shown on the URD tree. The fraction of cells in each of these transcriptional states was then determined for three developmental periods (12–24 hpf, 36 hpf – 3 dpf, and 5–15 dpf). In the retina, cells can be found in a progenitor state (light purple) that persists post-embryonically.

Figure1

FIGURE 1

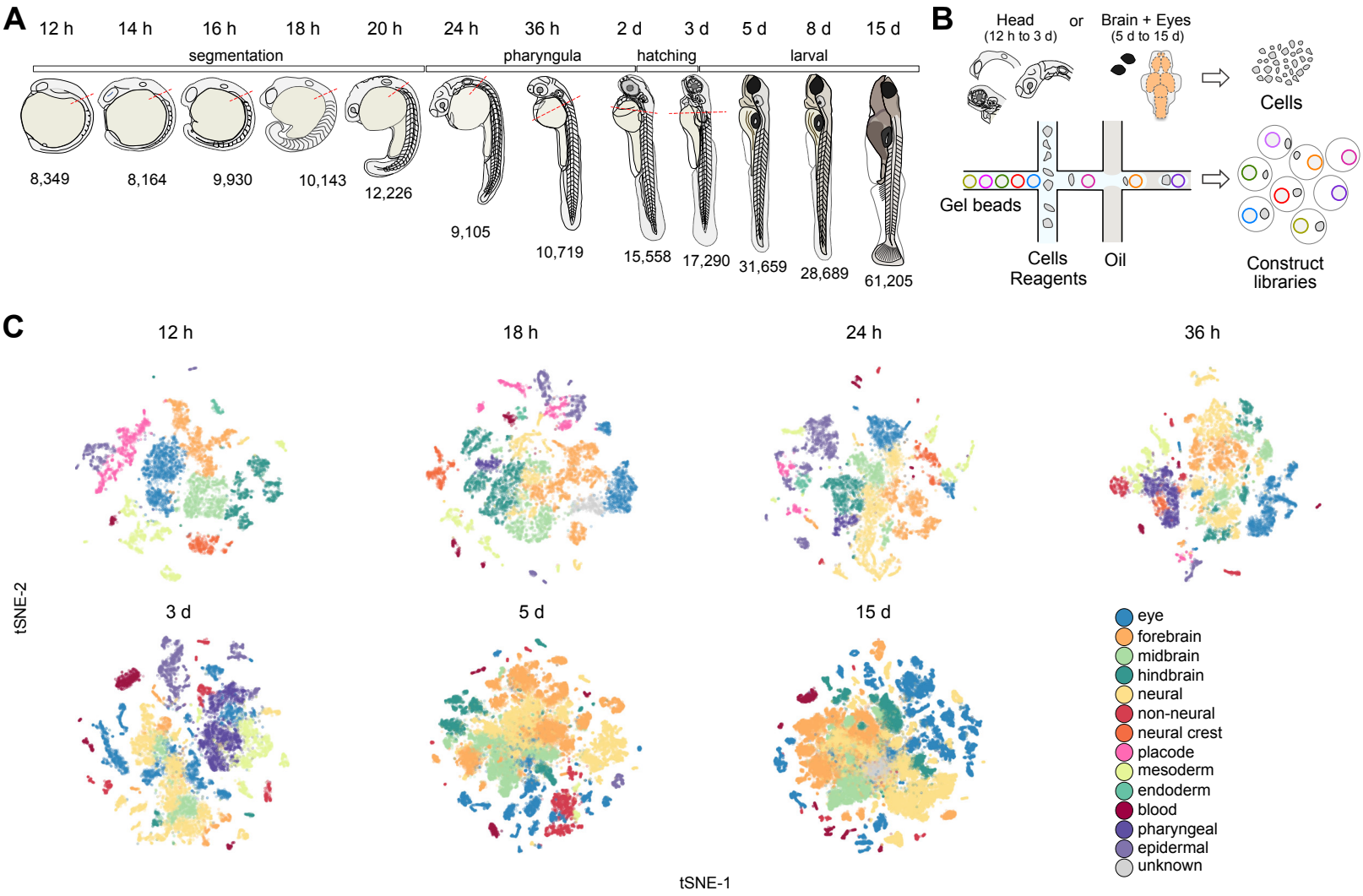
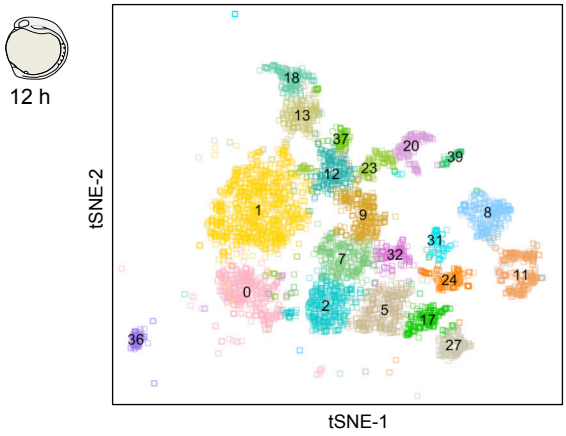


Figure2

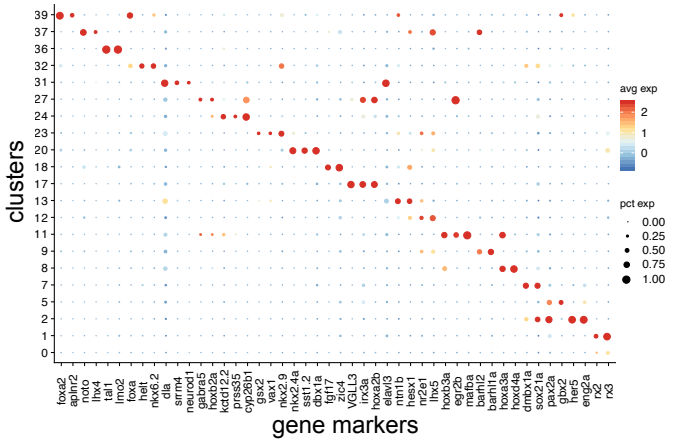
FIGURE 2

A

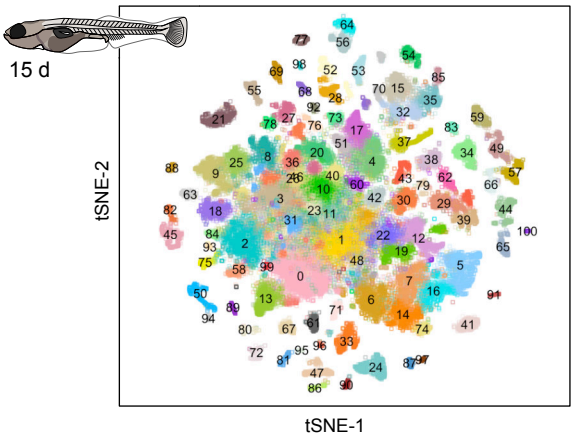


0 - optic vesicle
1 - optic vesicle
2 - midbrain
5 - mid/hindbrain boundary
7 - midbrain
8 - hindbrain r7
9 - diencephalon
11 - hindbrain r5/6
12 - dorsal diencephalon
13 - telencephalon
17 - hindbrain r1/2
18 - telencephalon
20 - ventral diencepahalon
23 - ventral diencephalon, optic stalk
24 - hindbrain r4
27 - hindbrain r3
31 - neurons
32 - ventral midbrain
36 - rostral blood island
37 - diencephalon
39 - floor plate

B



C



0,13 - optic tectum
2,58 - ventral forebrain
4,17,73 - granule cells
5 - radial glia
6,7,14,48 - progenitors
8,40,46,67 - diencephalon
9,25 - pallium
10,20,92 - thalamus
12 - URL progenitors
13,84 - telencephalon
15,32,35,54,70,85 - cone bipolar cells
16 - ventral progenitors
18 - tuberculum
19,74 - cycling progenitors
21,88 - erythrocytes
24 - microglia
27,31 - hindbrain/cranial nerve
28 - cones
29 - retina progenitors
30,79 - amacrine cells
26,36 - midbrain
33 - muller glia
34 - cornea
37 - lens
38,62 - retinal ganglion cells
39 - retina progenitors
41 - ventral habenula
42,68,91 - oligodendrocytes
43 - photoreceptor precursors
44,49,66 - cornea epithelium
45,82,83 - eye cells
47,86 - eye epidermis
50 - RPE
51 - torus longitudinalis
52 - rods
53 - glia
55 - purkinje cells
56,64 - dorsal habenula
57 - eye cartilage
59 - eye glia
61,63,75,93,99 - hypothalamus
65 - cornea epithelial progenitors
69 - horizontal cells
71 - OPC
72 - neutrophils
76 - ganglion neurons
77 - lens epithelium
78 - hindbrain
80 - metaphocytes
81 - myeloid cells
87,94 - pigment cells
89,90 - blood/immune cells
95 - perivascular FGP cells
96 - retina neurons
97,98 - neural crest pigment cells
100 - glia progenitors

D

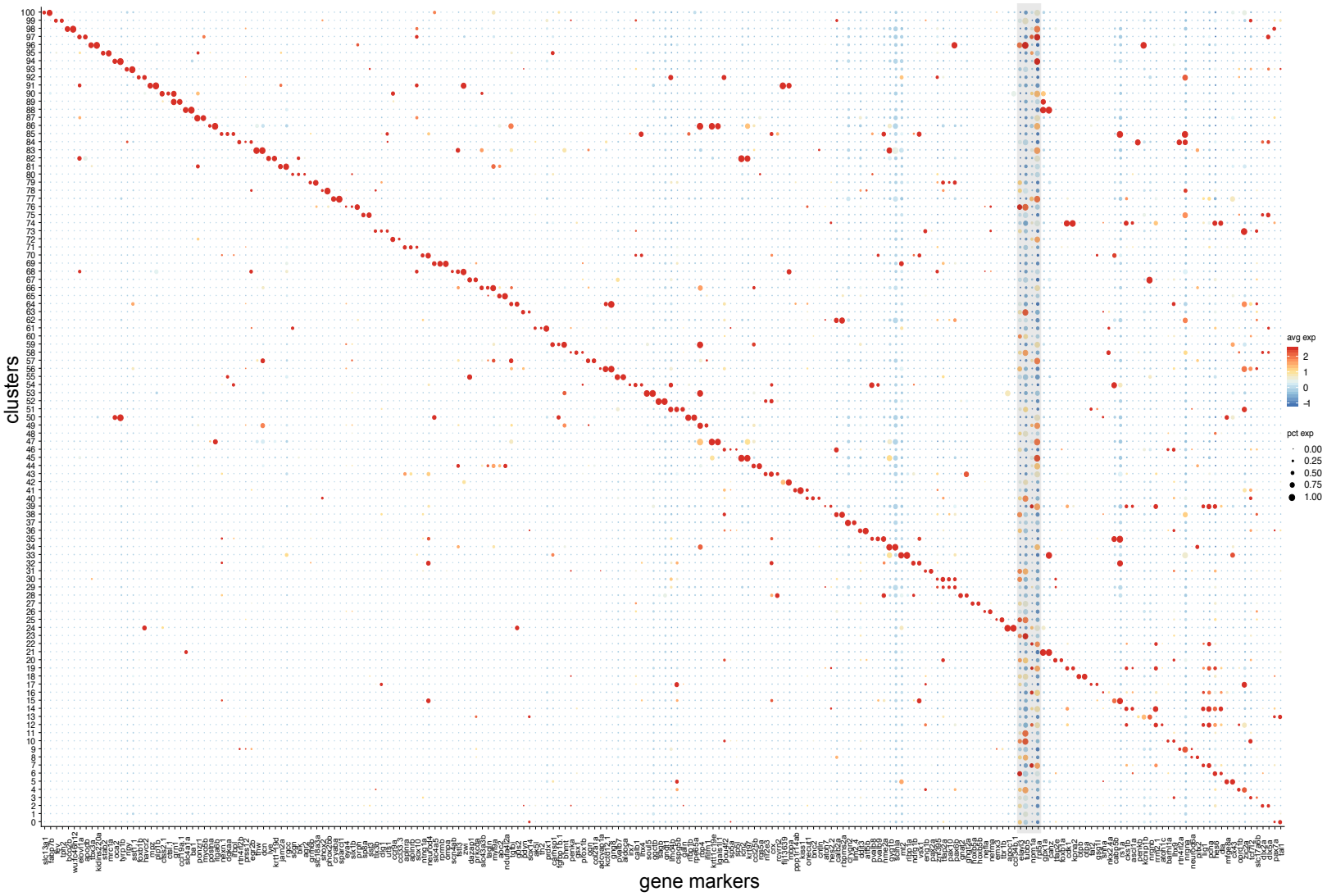


Figure3

FIGURE 3

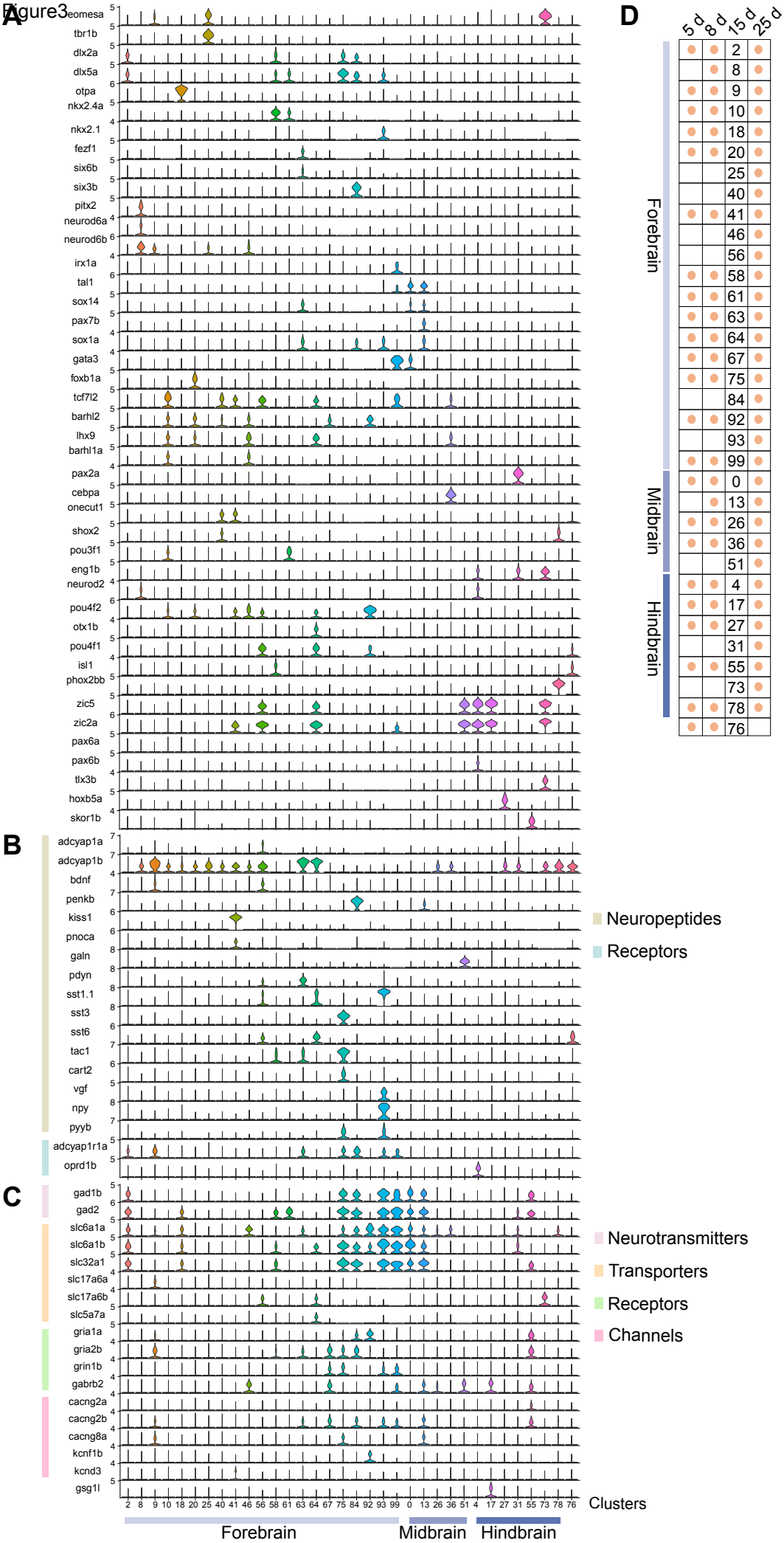


FIGURE 4

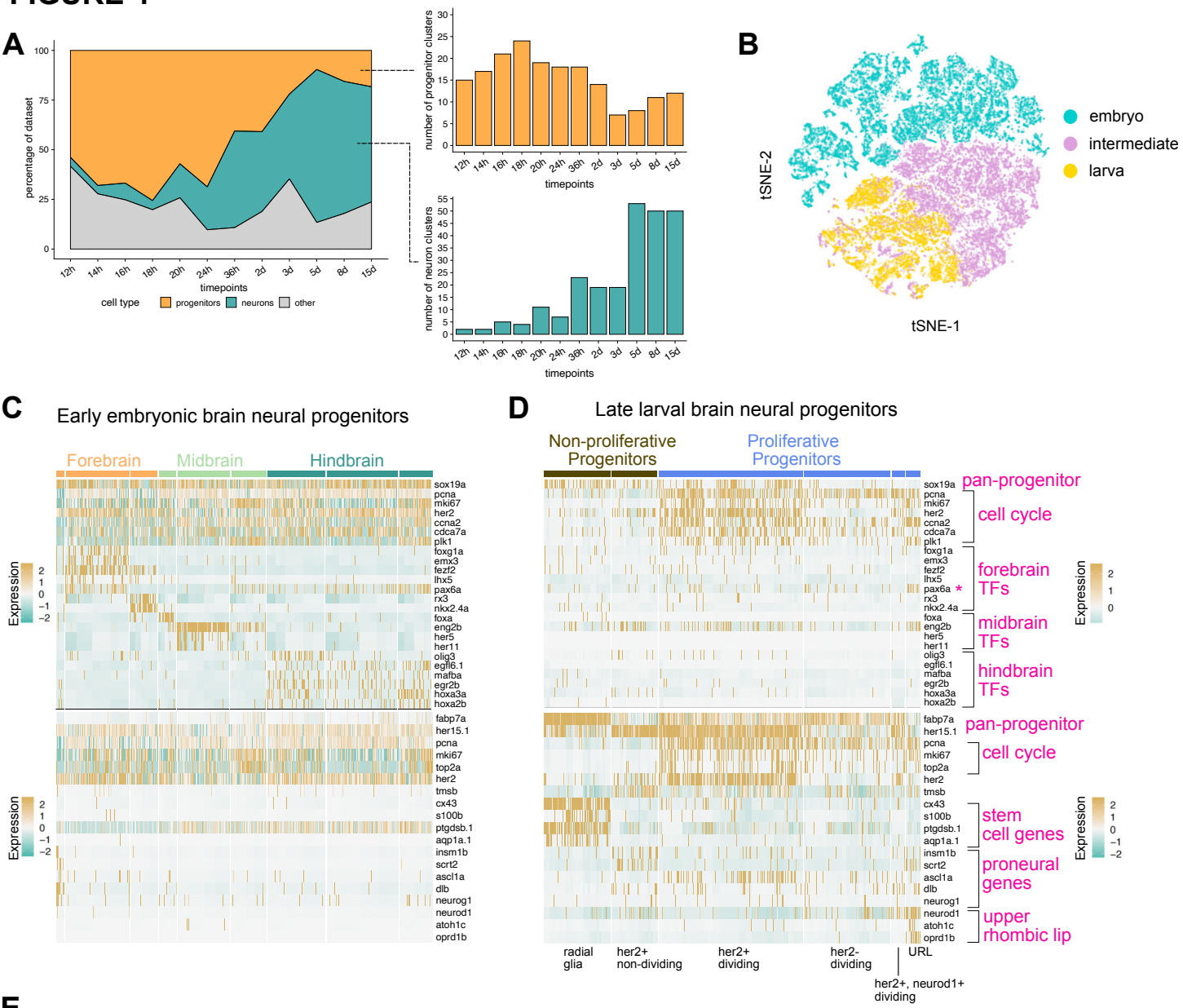
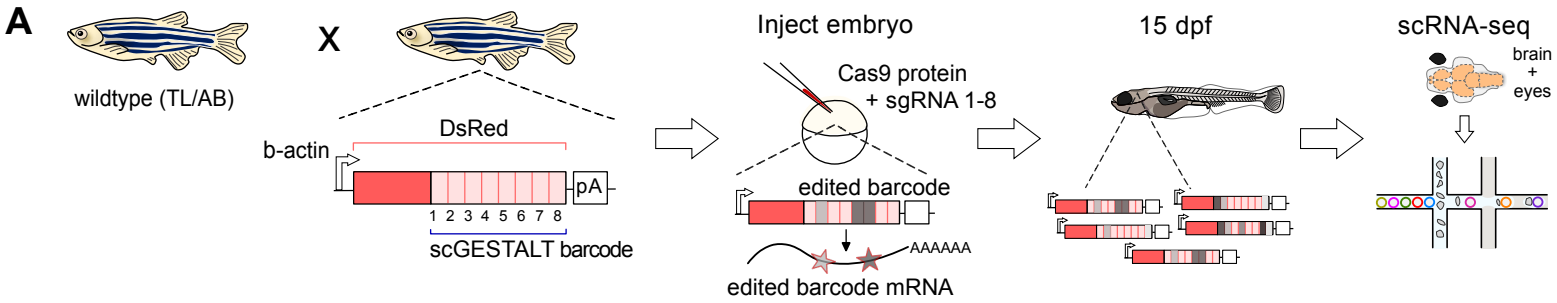
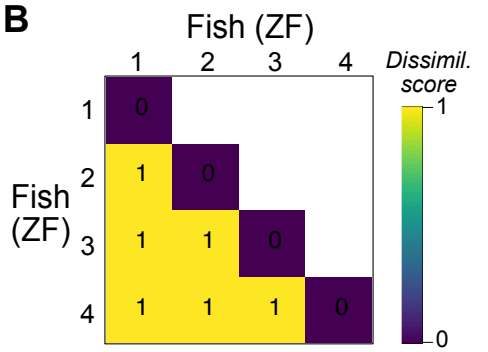
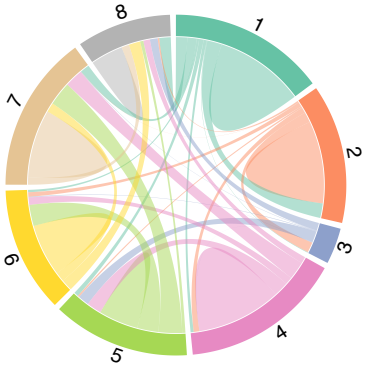
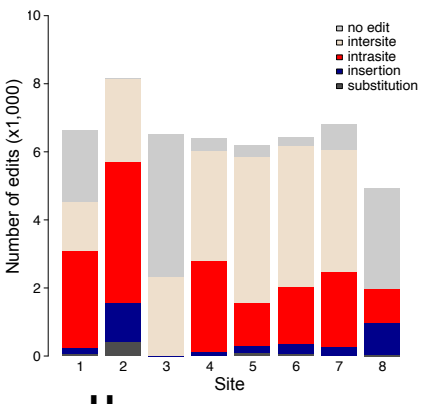

A
percentage of dataset
timepoints
cell type
progenitors
neurons
other
number of progenitor clusters
number of neuron clusters
B
tSNE-2
tSNE-1
embryo
intermediate
larva
C
Early embryonic brain neural progenitors
Forebrain
Midbrain
Hindbrain
Expression
D
Late larval brain neural progenitors
Non-proliferative Progenitors
Proliferative Progenitors
pan-progenitor
cell cycle
forebrain TFs
midbrain TFs
hindbrain TFs
stem cell genes
proneural genes
upper rhombic lip
radial glia
her2+ non-dividing
her2+ dividing
her2- dividing
URL
her2+, neurod1+ dividing
E
Embryo
Intermediate
Larva
midbrain
forebrain
F
Number of genes correlated with 79 spatial TFs
Pearson correlation threshold
embryo
intermediate
larva

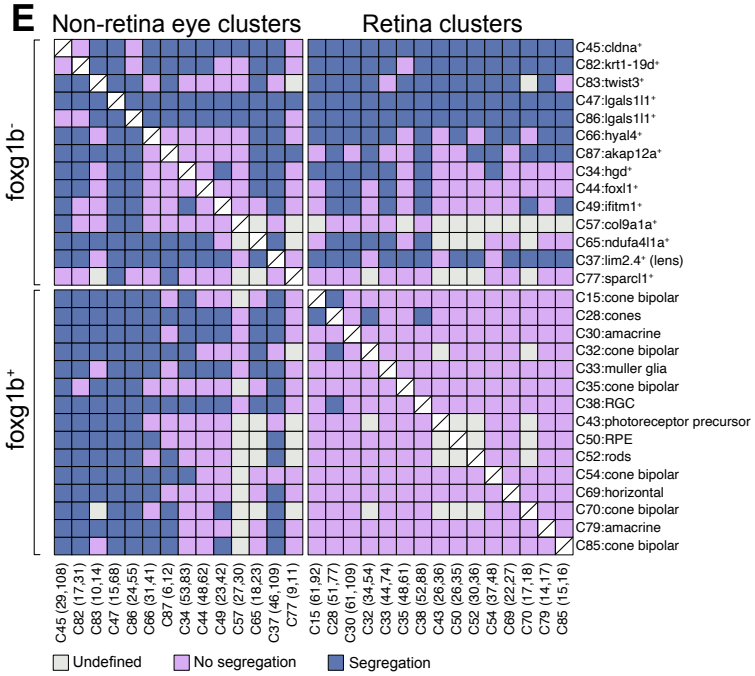
FIGURE 5

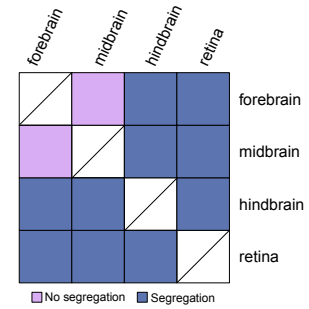

A
wildtype (TL/AB)
X
DsRed
b-actin
pA
1 2 3 4 5 6 7 8
scGESTALT barcode
Inject embryo
Cas9 protein
+ sgRNA 1-8
edited barcode
AAAAAA
edited barcode mRNA
15 dpf
scRNA-seq
brain
+
eyes

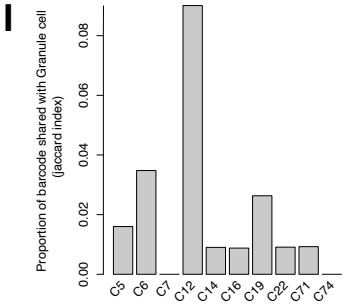

B
Fish (ZF)
1 2 3 4
Dissimil. score
1
0
Fish (ZF)
1 0
1 1 0
1 1 1 0

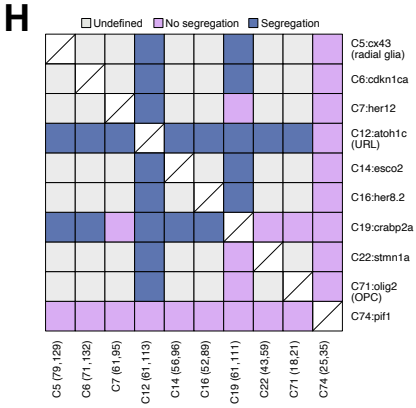

C
1
2
3
4
5
6
7
8


D
Number of edits (x1,000)
Site
no edit
intersite
intrasite
insertion
substitution


E
Non-retina eye clusters
Retina clusters
foxg1b-
foxg1b+
C45:cldna+
C82:krt1-19d+
C83:twist3+
C47:lgals1l1+
C86:lgals1l1+
C66:hyal4+
C87:akap12a+
C34:hgd+
C44:foxl1+
C49:ifitm1+
C57:col9a1a+
C65:ndufa4l1a+
C37:lim2.4+ (lens)
C77:sparcl1+
C15:cone bipolar
C28:cones
C30:amacrine
C32:cone bipolar
C33:muller glia
C35:cone bipolar
C38:RGC
C43:photoreceptor precursor
C50:RPE
C52:rods
C54:cone bipolar
C69:horizontal
C70:cone bipolar
C79:amacrine
C85:cone bipolar
Undefined
No segregation
Segregation


F
forebrain
midbrain
hindbrain
retina
No segregation
Segregation


I
Proportion of barcode shared with Granule cell (Jaccard index)
0.00
0.02
0.04
0.06
0.08
C5
C6
C7
C12
C14
C16
C19
C22
C71
C74


H
Undefined
No segregation
Segregation
C5:cx43 (radial glia)
C6:cdkn1ca
C7:her12
C12:atoh1c (URL)
C14:esco2
C16:her8.2
C19:crabp2a
C22:stmn1a
C71:olig2 (OPC)
C74:pif1

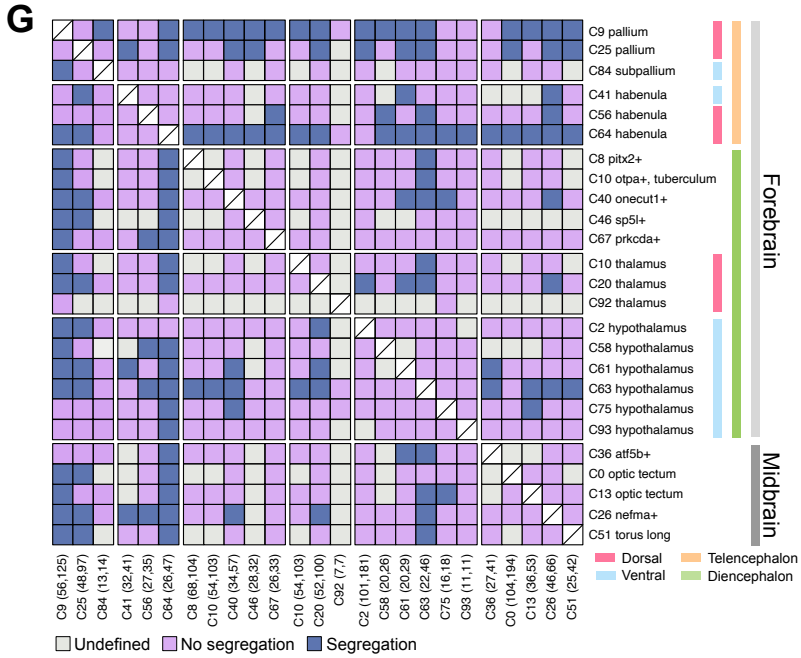

G
C9 pallium
C25 pallium
C84 subpallium
C41 habenula
C56 habenula
C64 habenula
C8 pitx2+
C10 otpa+, tuberculum
C40 onecut1+
C46 sp5l+
C67 prkcda+
C10 thalamus
C20 thalamus
C92 thalamus
C2 hypothalamus
C58 hypothalamus
C61 hypothalamus
C63 hypothalamus
C75 hypothalamus
C93 hypothalamus
C36 atf5b+
C0 optic tectum
C13 optic tectum
C26 nefma+
C51 torus long
Forebrain
Midbrain
Dorsal
Ventral
Telencephalon
Diencephalon
Undefined
No segregation
Segregation

Figure6

FIGURE 6

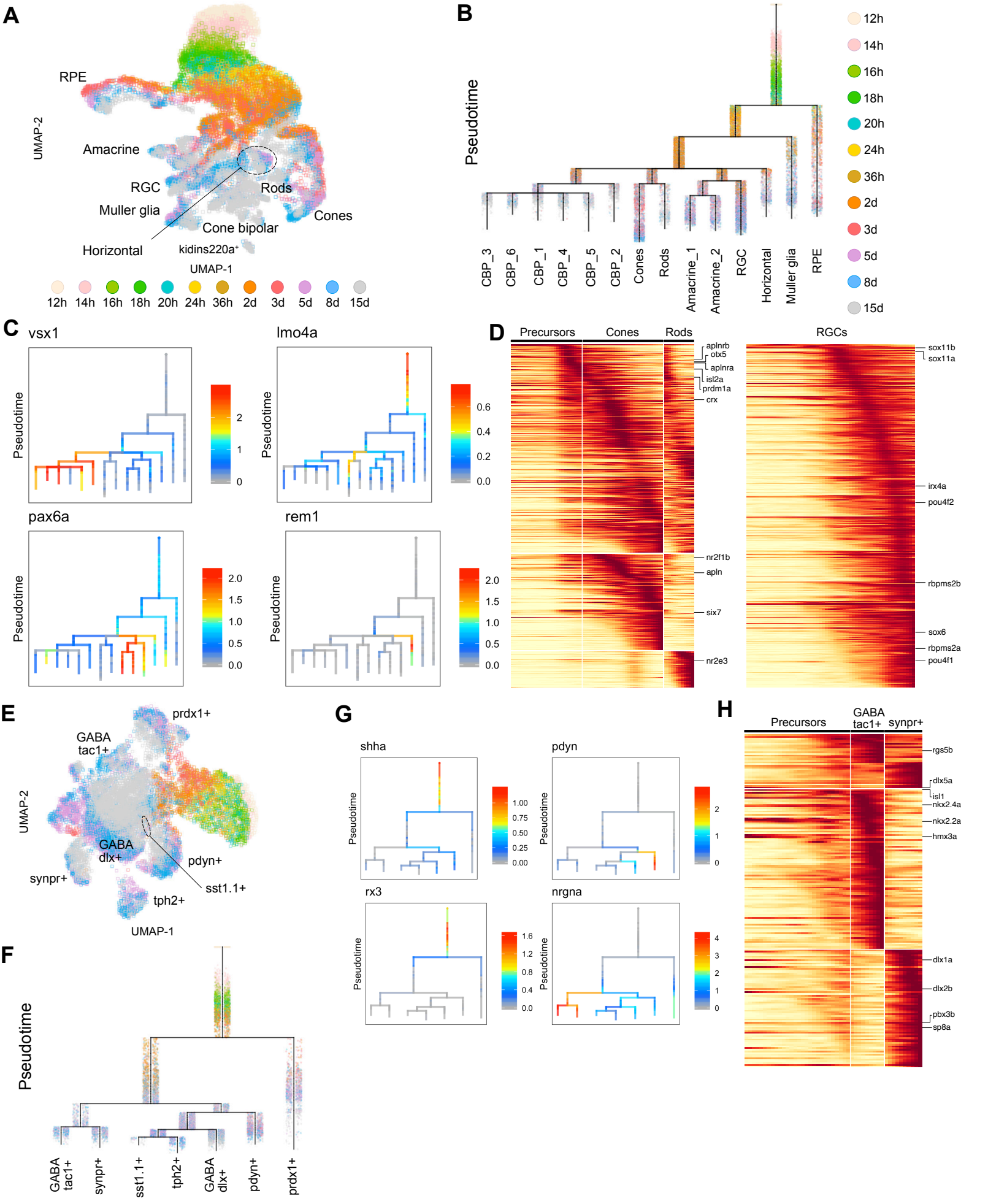
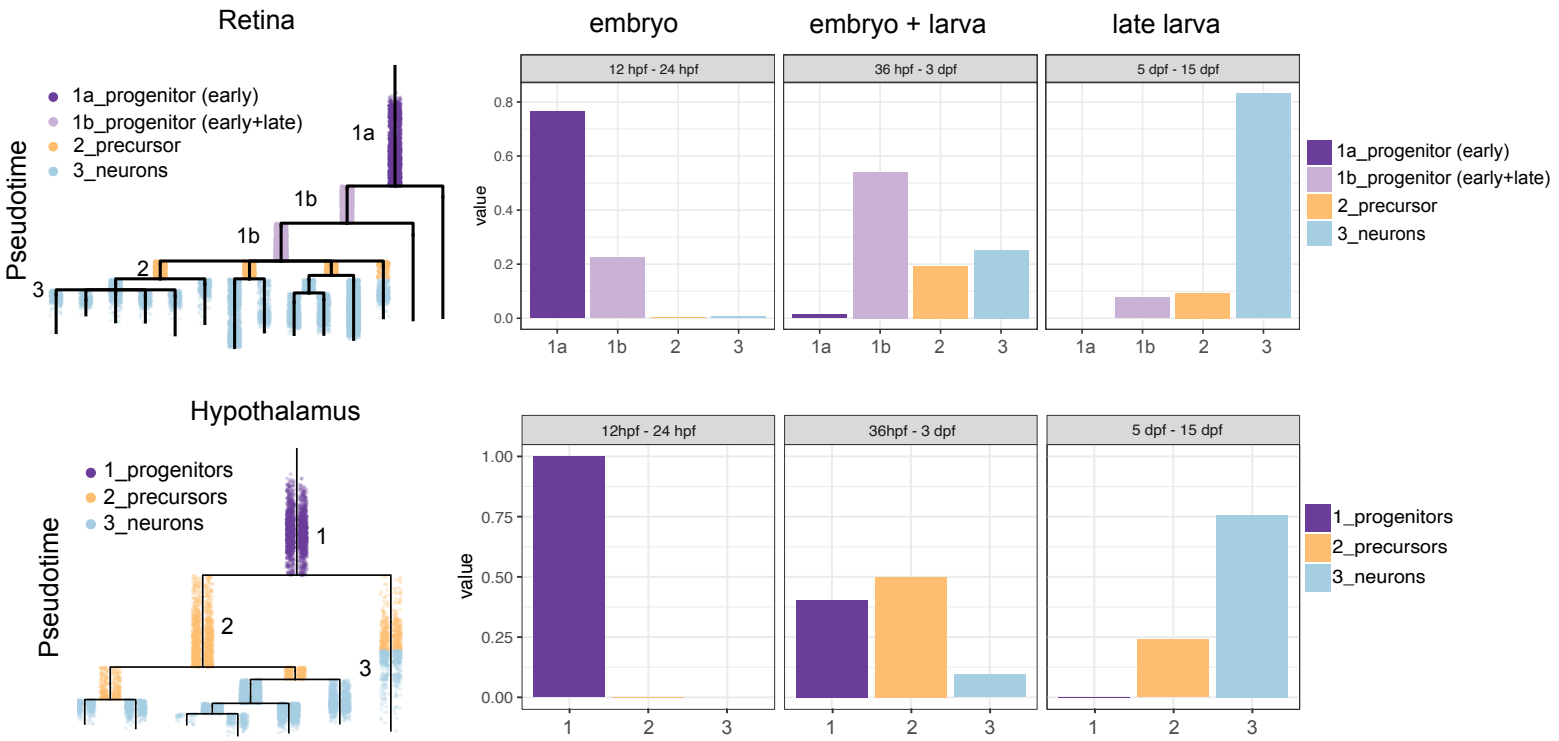


Figure7

FIGURE 7



Supplemental Information

Emergence of neuronal diversity during vertebrate brain development

Bushra Raj[1,2]*, Jeffrey A. Farrell[1,3], Jialin Liu[2,4], Jakob El Kholtei[2,4], Adam Carte[1,4,5], Joaquin Navajas Acedo[2,4], Lucia Y Du[2,4], Aaron McKenna[6], Đorđe Relić[3,7], Jessica M. Leslie[1], and Alexander F. Schier[1,2,4,8,9,10]*

[1] Department of Molecular and Cellular Biology, Harvard University, Cambridge, Massachusetts, USA
[2] Allen Discovery Center for Cell Lineage Tracing, Seattle, Washington, USA
[3] Unit on Cell Specification and Differentiation, National Institute of Child Health and Human Development, NIH, Bethesda, Maryland, USA
[4] Biozentrum, University of Basel, Switzerland
[5] Systems, Synthetic, and Quantitative Biology Program, Harvard University, Cambridge, Massachusetts, USA
[6] Department of Molecular and Systems Biology, Dartmouth Geisel School of Medicine, Lebanon, New Hampshire, USA
[7] Swiss Institute of Bioinformatics (SIB), Basel, Switzerland
[8] Broad Institute of MIT and Harvard, Cambridge, Massachusetts, USA
[9] Harvard Stem Cell Institute, Harvard University, Cambridge, Massachusetts, USA
[10] Center for Brain Science, Harvard University, Cambridge, Massachusetts, USA

*Co-corresponding authors. Email: bushranraj@gmail.com (B.R.); alex.schier@unibas.ch (A.F.S.)

Sup Figure 1. Zebrafish brain cell types identified at each stage of time course

tSNE plots of cell types at each stage of the time course. Cells are color coded by stage.
tSNE implementations: Barnes-Hut (12 hpf to 3 dpf, 8 dpf), Fourier transform (5 dpf and 15 dpf). Cluster numbers are indicated on each plot. Detailed cluster descriptions are in Supplementary Table 1 and can be explored interactively in the accompanying app at https://github.com/brlauuu/zf_brain.

Sup Figure 2. Neuron subtype diversity at 5 and 8 dpf

Violin plots of select marker gene expression in identified brain neuron subtypes from 5 dpf (left) and 8 dpf (right). Retina neurons and nascent (immature) neurons are omitted from the analysis. Cluster numbers are indicated at the bottom along with their inferred spatial location in the brain. Clusters 69 (5 dpf) and 51 (8 dpf) have unknown spatial location. Detailed cluster descriptions are in Supplementary Table 1 and can be explored interactively in the accompanying app at https://github.com/brlauuu/zf_brain.

Sup Figure 3. Embryonic, intermediate and larval stage neural progenitor populations

tSNE plots showing embryonic (12 hpf – 18 hpf), intermediate (20 hpf – 3 dpf) and larval (5 dpf – 15 dpf) stage progenitor clusters that were subsetted from the dataset. Cluster numbers match plots shown in Sup Figure 1. Detailed cluster descriptions are in Supplementary Table 1 and can be explored interactively in the accompanying app at https://github.com/brlauuu/zf_brain.

Sup Figure 4. Optimized scGESTALT lineage recorder enables reconstruction of more dense lineage trees

A. Size and diversity of clones from edited ZF1-4 larval brains. Each colored rectangle represents a unique clone and the area represents the size of the clone.
B. A reconstructed scGESTALT brain lineage tree from one zebrafish 15 dpf brain. 302 barcodes recovered from ZF1 were assembled into a lineage tree. Barcode edits are represented as red (deletions), blue (insertions), and black (substitutions). Associated cells are color coded by cell type and region. Interactive trees are presented at: https://scgestalt.mckennalab.org/. Each tip on the tree has an associated cell type assignment (color coded), a lineage barcode schematic, and a cluster number. For reasons of space, the tree is split into multiple columns and dashed lines connect subsections of the tree together.

Sup Figure 5.
Retinal cell type marker expression and trajectory analysis

A. UMAP plots highlighting expression of select genes enriched in retinal cell types. *rx3*, *vsx2*, *hes2.2* are enriched in early embryonic retinal progenitors; *foxg1b* is enriched in differentiated cells; *pax6a* is enriched in progenitors, retinal ganglion cells (RGC) and amacrine cells; *crx* is enriched in photoreceptor cells and cone bipolar cells; *gngt2a* is enriched in cones; *gnat1* is enriched in rods; *ompa* is enriched in horizontal cells; *tfap2a* is enriched in RGCs and horizontal cells; *apoeb* is enriched in early progenitors and muller glia; *rbpms2a* is enriched in amacrine cells; *vsx1* is enriched in cone bipolar cells; *cabp5b* is enriched in cone bipolar cells; *rpe65a* is enriched in retinal pigment epithelium; *kidins220a* is enriched in new retinal subtype.
B. tSNE plot of 15 dpf brain and eye cell types. Retinal cell types used as the endpoint cell types (tips) for URD analysis are color coded. Cluster number and cell type description are indicated on the legend. Cluster 96 was discarded from all analysis, see Results.

Sup Figure 6.
Retina and hypothalamus URD trajectory analysis

A. Cell specification tree of zebrafish retinal development generated with URD, reproduction of Figure 6B for comparison. CBP, cone bipolar cells (6 subtypes are numbered); RGC, retinal ganglion cells; RPE, retinal pigment epithelium
B. Assessing robustness of cell assignment on URD retina trees. Trees were recalculated with random subsets of 50% of the cells from the original retinal dataset (sampled per stage so that proportions of cells from each stage remained

constant). The parameters used were the same, except the number of nearest neighbors used was reduced to reflect the smaller dataset. The position of cells on trees was compared to the original tree. Cells assigned the same segment are colored blue ("Same"), cells that moved to a parent or child segment ("Parent/Child") are colored green, cells that changed assignment to a different location are colored red ("Changed"), and cells that were not assigned a location in the original tree are colored grey. 82.1% cells retained their original assignment, 12.5% cells shifted to either parent or child segments (reflects small shifts in pseudotime of branchpoint assignment), and 5.5% cells shifted to a different location.

C. Cell specification trees of zebrafish hypothalamus development generated with URD, reproduction of Figure 6F for comparison.

D. Assessing robustness of cell assignment on URD hypothalamus trees. Trees were recalculated with random subsets of 50% of the cells from the original hypothalamus dataset (sampled per stage so that proportions of cells from each stage remained constant). The parameters used were the same, except the number of nearest neighbors used was reduced to reflect the smaller dataset. The position of cells on trees was compared to the original tree. Cells assigned the same segment are colored blue ("Same"), cells that moved to a parent or child segment ("Parent/Child") are colored green, cells that changed assignment to a different location are colored red ("Changed"), and cells that were not assigned a location in the original tree are colored grey. 80.1% cells retained their original assignment, 13.6% cells shifted to either parent or child segments (reflects small shifts in pseudotime of branchpoint assignment or small changes in tree structure), and 6.3% cells shifted to a different location.

Sup Figure 7. Gene expression along retina cell trajectories

Expression of select genes are shown on the retina URD specification tree. Cell types: 1. Cone bipolar cell (CBP_3); 2. Cone bipolar cell (CBP_6); 3. Cone bipolar cell (CBP_1); 4. Cone bipolar cell (CBP_4); 5. Cone bipolar cell (CBP_5); 6. Cone bipolar cell (CBP_2); 7. Cones; 8. Rods; 9. Amacrine cells (Amacrine_1); 10. Amacrine cells (Amacrine_2); 11. Retinal ganglion cells (RGC); 12. Horizontal cells; 13. Muller glia; 14. Retinal pigment epithelium (RPE)

Sup Figure 8. Gene expression cascades of retinal cell trajectories

Heat maps of gene expression cascades of photoreceptor cell, amacrine cell, retinal ganglion cell, muller glia, horizontal cell and retinal pigment epithelium cell trajectories. Cells were selected based on high expression along trajectories leading to these cell types, compared to expression along opposing branchpoints. Red, high expression. Yellow, low expression. X-axis represents cell states along the cascade progression.

Sup Figure 9. Müller glia-like cells are detected early in zebrafish retina development

Detection of Muller glia markers *cahz* and *rlbp1a* in the retina at **A.** 36 hpf and **B.** 2 dpf

A. Left panel, retina section stained with DAPI (grey), *cahz* (cyan) and rlbp1a (yellow). White box indicates area that was zoomed in for the right panels.

B. Left panel, retina section stained with DAPI (grey), *cahz* (cyan) and rlbp1a (yellow). White and red boxes indicate area that were zoomed in for the middle and right panels, respectively. Middle panels denote cells that co-express *rlbp1a* and *cahz*. Right panels denote cells that are *rlbp1a*$^{+}$ and *cahz*$^{-}$

Sup Figure 10. Hypothalamus cell type marker expression and trajectory analysis

A. UMAP plots highlighting expression of select genes enriched in hypothalamus cell types. *dbx1a*, *fezf2*, *rx3* are enriched in early embryonic hypothalamus progenitors; *fezf1* is enriched in *pdyn*$^{+}$ subtype; *nrgna* is enriched in two subtypes (*synpr*$^{+}$ and *synpr*$^{-}$*), dlx2a* is expressed in several subtypes; *nkx2.4a* is expressed in early progenitors and *prdx1*$^{+}$ subtype; *synpr, npy, tph2, pdyn and prdx1 are enriched in specific subtypes*

B. tSNE plot of 15 dpf brain and eye cell types. Hypothalamus cell types used as the endpoint cell types (tips) for URD analysis are color coded. Cluster number and cell type description are indicated on the legend.

Sup Figure 11. Gene expression along hypothalamus cell trajectories

Expression of select genes are shown on the hypothalamus specification tree. Cell types: 1. GABA tac1+, nrgna+; 2. synpr+; nrgna+; 3. sst1.1+; 4. tph2+; 5. GABA dlx+; 6. pdyn+; 7. prdx1+

Sup Figure 12. Gene expression cascades of hypothalamus cell trajectories

Heat maps of gene expression cascades of profiled hypothalamus cell trajectories. Cells were selected based on high expression along trajectories leading to these cell types, compared to expression along opposing branchpoints. Red, high expression. Yellow, low expression. X-axis represents cell states along the cascade progression.

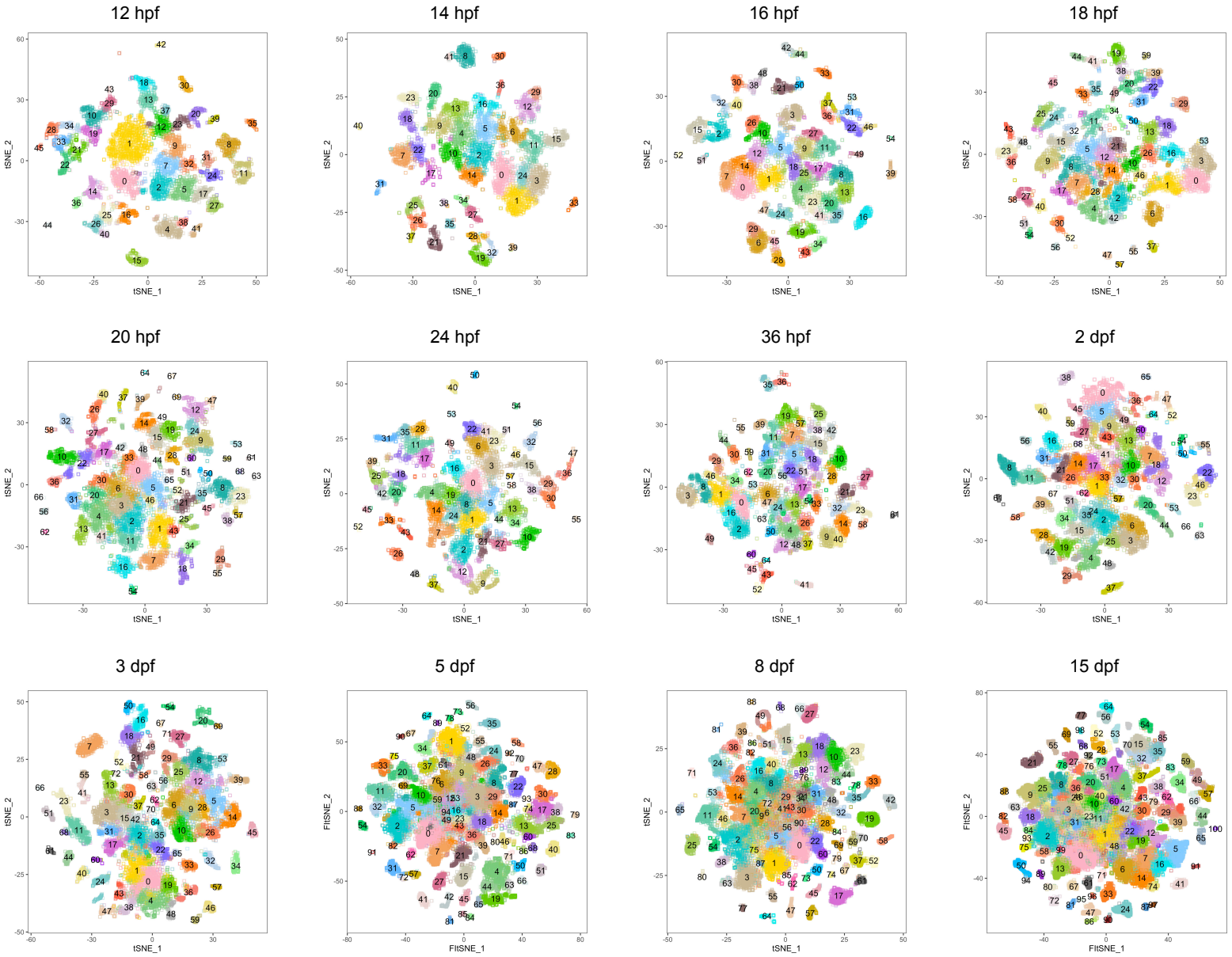
Sup Figure 13. Cell type maturation along URD trajectories

Retinal ganglion cells **(A)** and *pdyn*+ hypothalamic neurons **(B)** cells were plotted (red circles) on URD cell specification trajectories across the stages indicated. The cell types matured with developmental age, as expected. Additionally, later stages also contained immature cell states (early pseudotime) consistent with continuous neurogenesis.

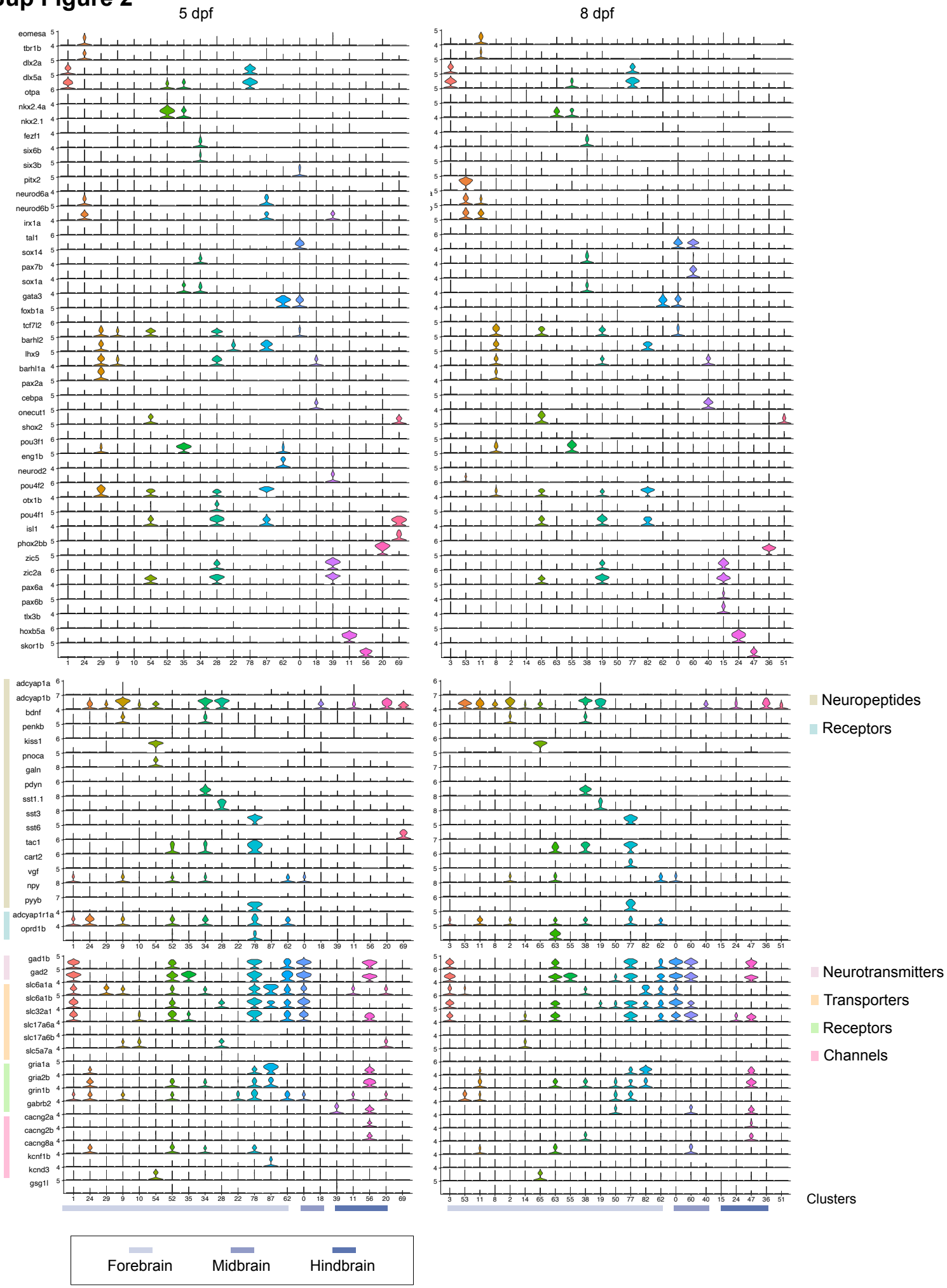
Sup Figure 14. Gene markers of embryonic and larval progenitors in retina and hypothalamus

Dot plot of gene expression pattern of select marker genes that were used to define progenitor and precursor states (rows) for segments (columns) of the retina (left) or hypothalamus (right) cell specification trees (see Figure 7E). Dot size indicates the percentage of cells expressing the marker; color represents the average scaled expression level.

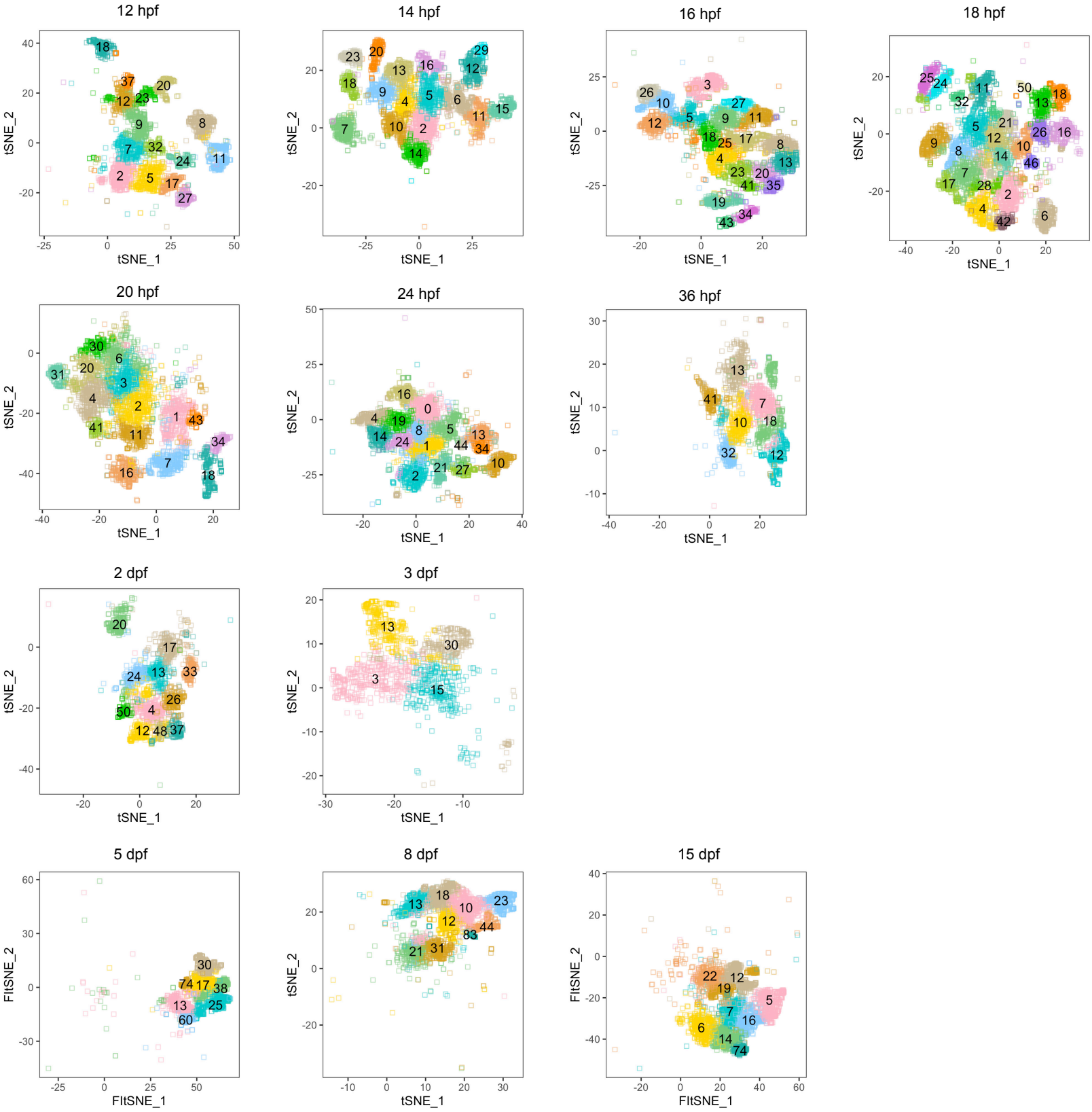
Sup Figure 1


12 hpf
14 hpf
16 hpf
18 hpf
20 hpf
24 hpf
36 hpf
2 dpf
3 dpf
5 dpf
8 dpf
15 dpf
tSNE_1
tSNE_2
FItSNE_1
FItSNE_2

Sup Figure 2

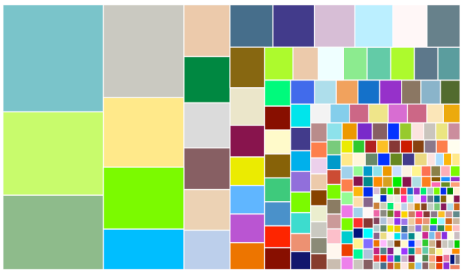

5 dpf
8 dpf
eomesa
tbr1b
dlx2a
dlx5a
otpa
nkx2.4a
nkx2.1
fezf1
six6b
six3b
pitx2
neurod6a
neurod6b
irx1a
tal1
sox14
pax7b
sox1a
gata3
foxb1a
tcf7l2
barhl2
lhx9
barhl1a
pax2a
cebpa
onecut1
shox2
pou3f1
eng1b
neurod2
pou4f2
otx1b
pou4f1
isl1
phox2bb
zic5
zic2a
pax6a
pax6b
tlx3b
hoxb5a
skor1b
adcyap1a
adcyap1b
bdnf
penkb
kiss1
pnoca
galn
pdyn
sst1.1
sst3
sst6
tac1
cart2
vgf
npy
pyyb
adcyap1r1a
oprd1b
gad1b
gad2
slc6a1a
slc6a1b
slc32a1
slc17a6a
slc17a6b
slc5a7a
gria1a
gria2b
grin1b
gabrb2
cacng2a
cacng2b
cacng8a
kcnf1b
kcnd3
gsg1l
Neuropeptides
Receptors
Neurotransmitters
Transporters
Receptors
Channels
Clusters
Forebrain
Midbrain
Hindbrain

Sup Figure 3


12 hpf
14 hpf
16 hpf
18 hpf
20 hpf
24 hpf
36 hpf
2 dpf
3 dpf
5 dpf
8 dpf
15 dpf
tSNE_1
tSNE_2
FItSNE_1
FItSNE_2

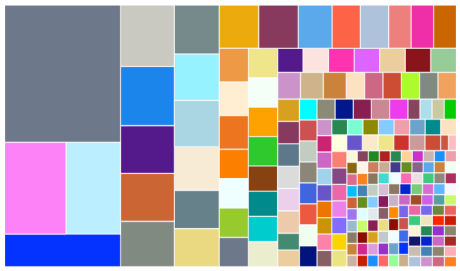
Sup Figure 4

A



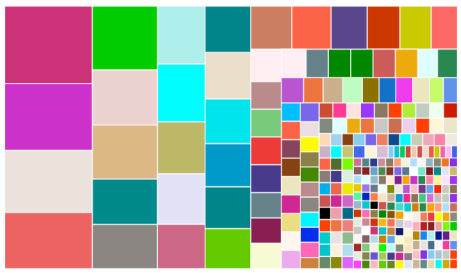
ZF1

302 clones
2,331 cells
largest clone = 207 cells



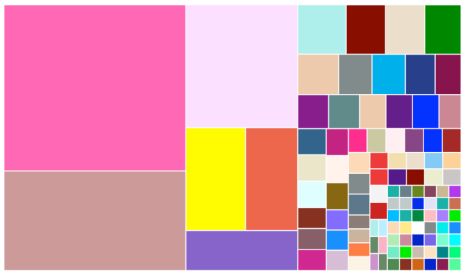
ZF2

216 clones
968 cells
largest clone = 130 cells



ZF3

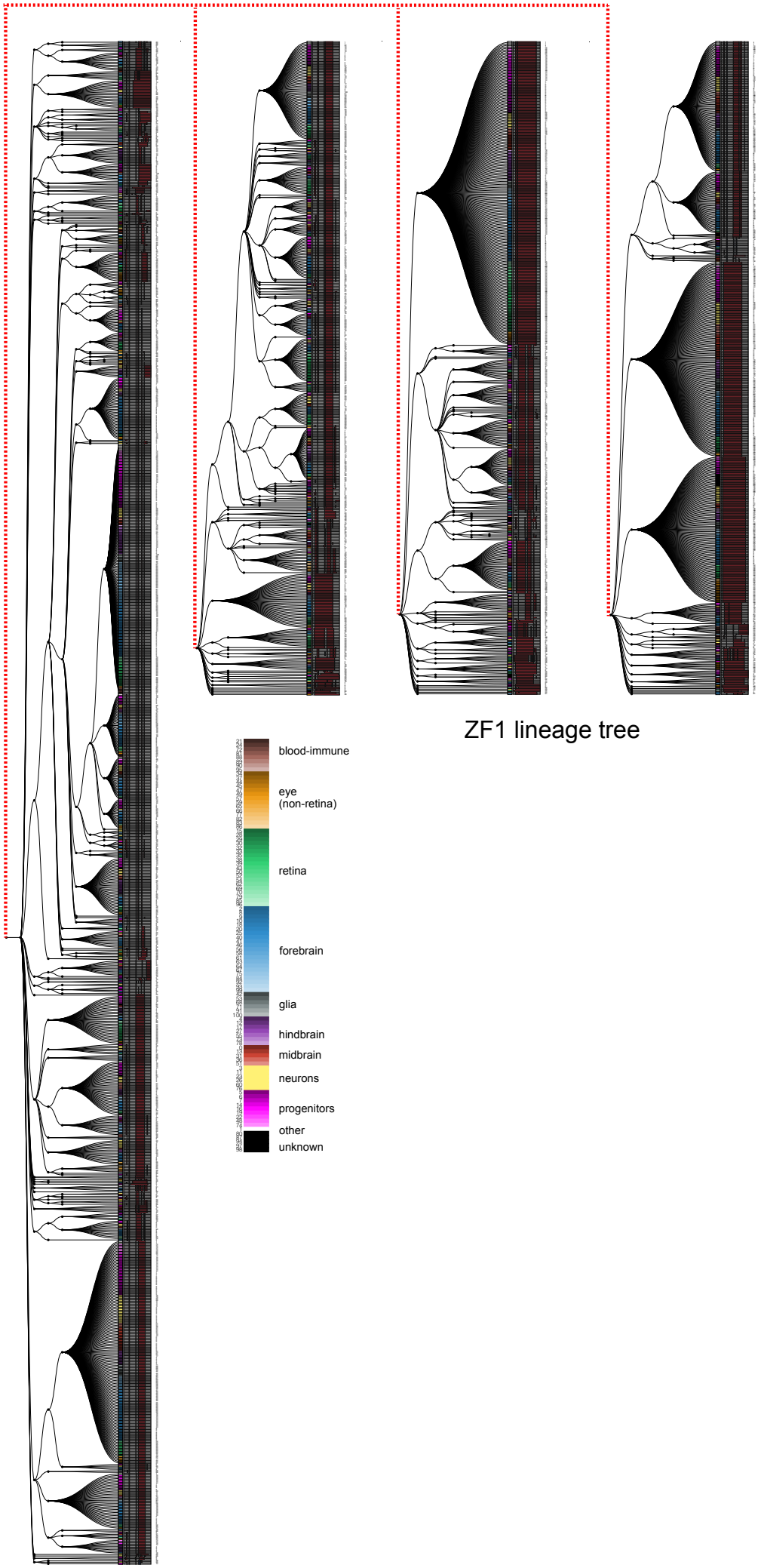
312 clones
1,685 cells
largest clone = 96 cells



ZF4

106 clones
812 cells
largest clone = 202 cells

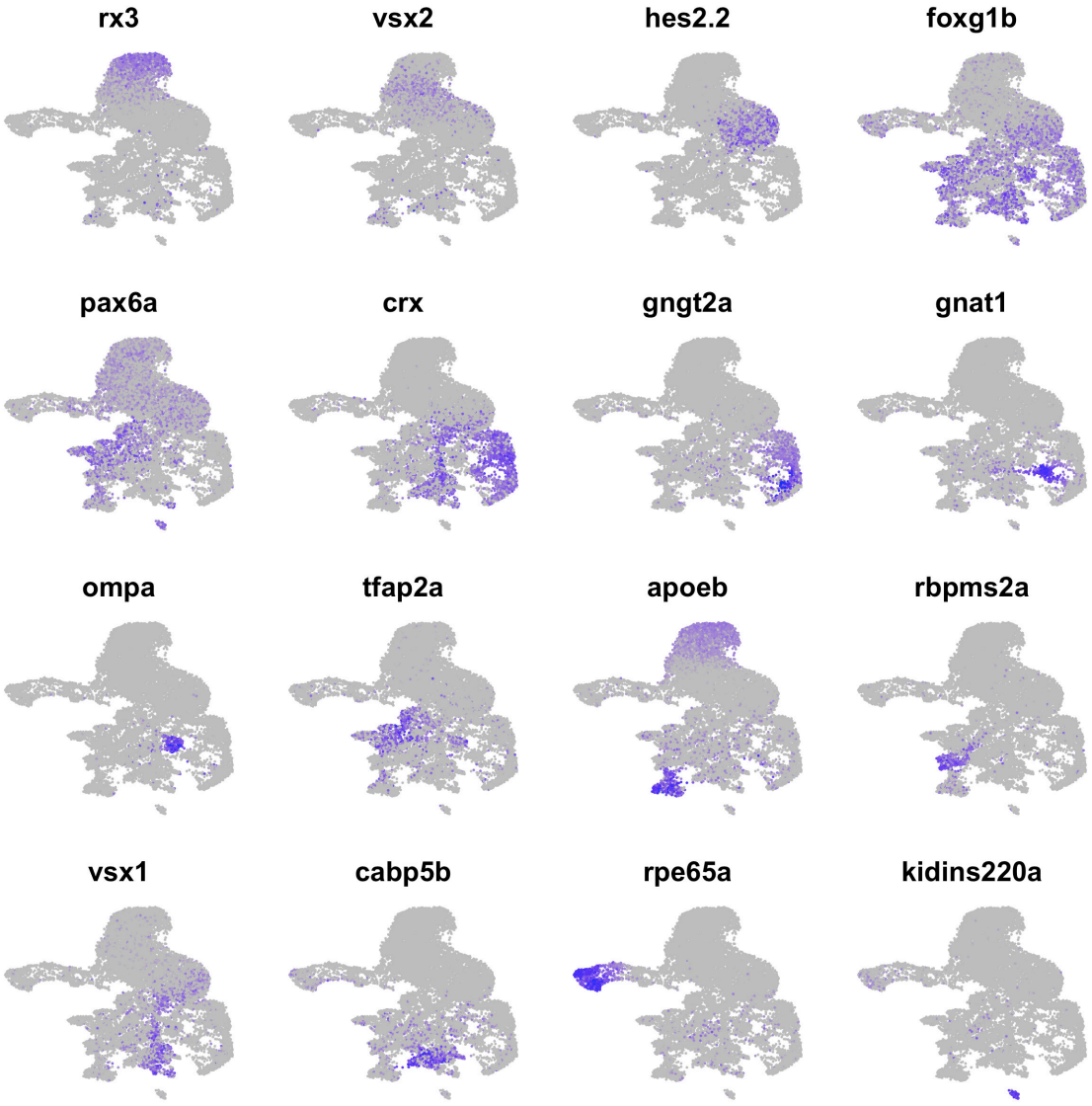
B



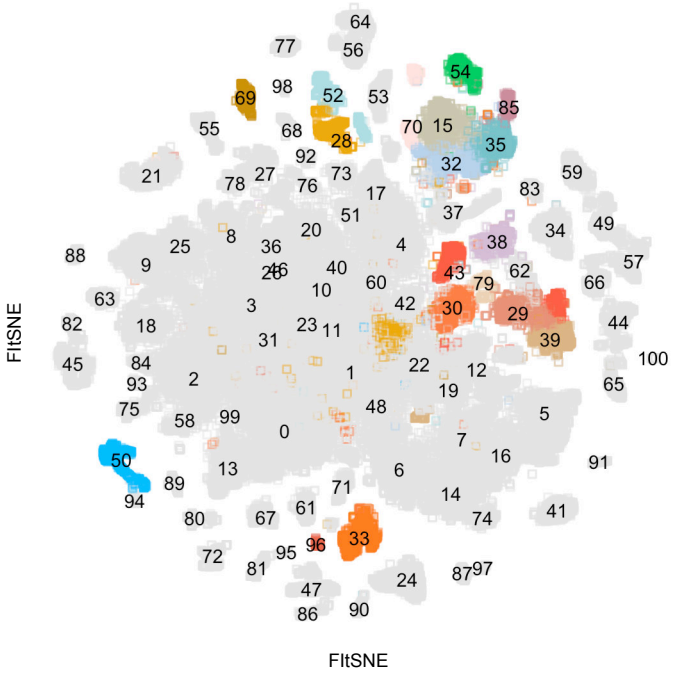
ZF1 lineage tree

A

Sup Figure 5

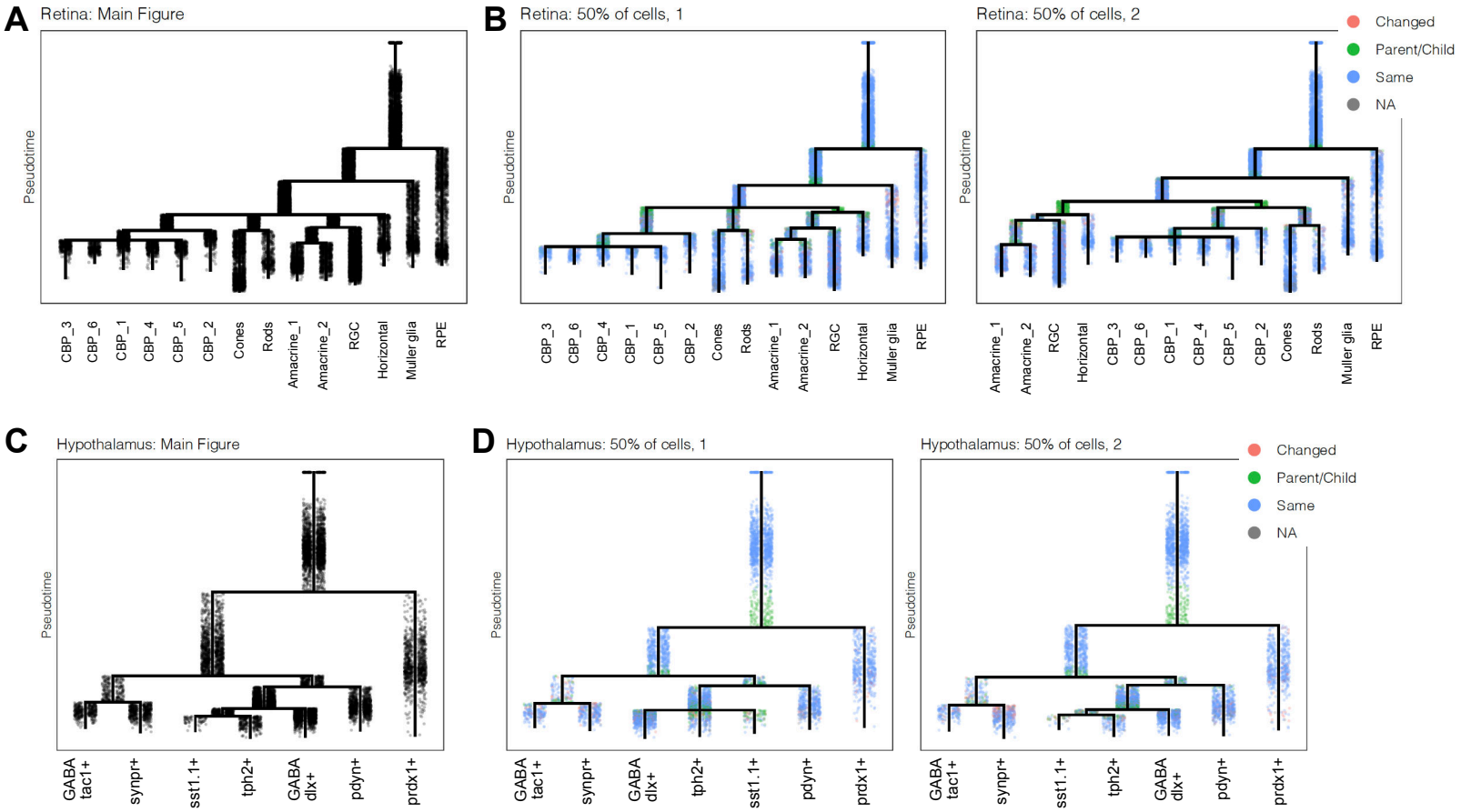


B

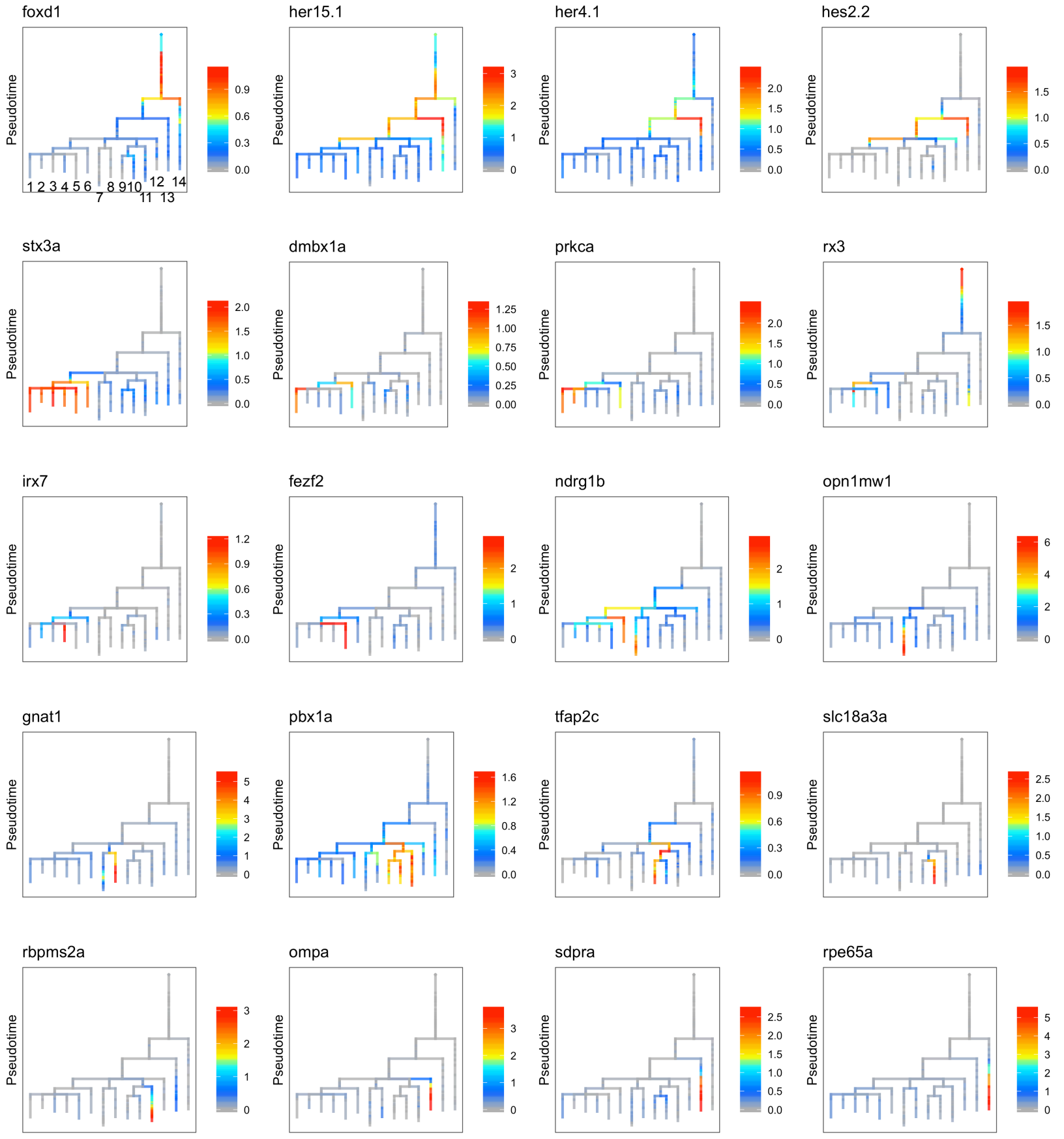


15	cone bipolar cells (CBP_1)
28	cones
29	pax6a+ cells (not used as tip)
30	amacrine cells (amacrine_1)
32	cone bipolar cells (CBP_2)
33	muller glia
35	cone bipolar cells (CBP_3)
38	RGC
39	retinal neural prgenitors (not used as tip)
43	photoreceptor precursor cells (not used as tip)
50	RPE
52	rods
54	cone bipolar cells (CBP_4)
69	horizontal cells
70	cone bipolar cells (CBP_5)
79	starburst amacrine cells (amacrine_2)
85	cone bipolar cells (CBP_6)
96	kidins220a+ cells (not used in analysis)

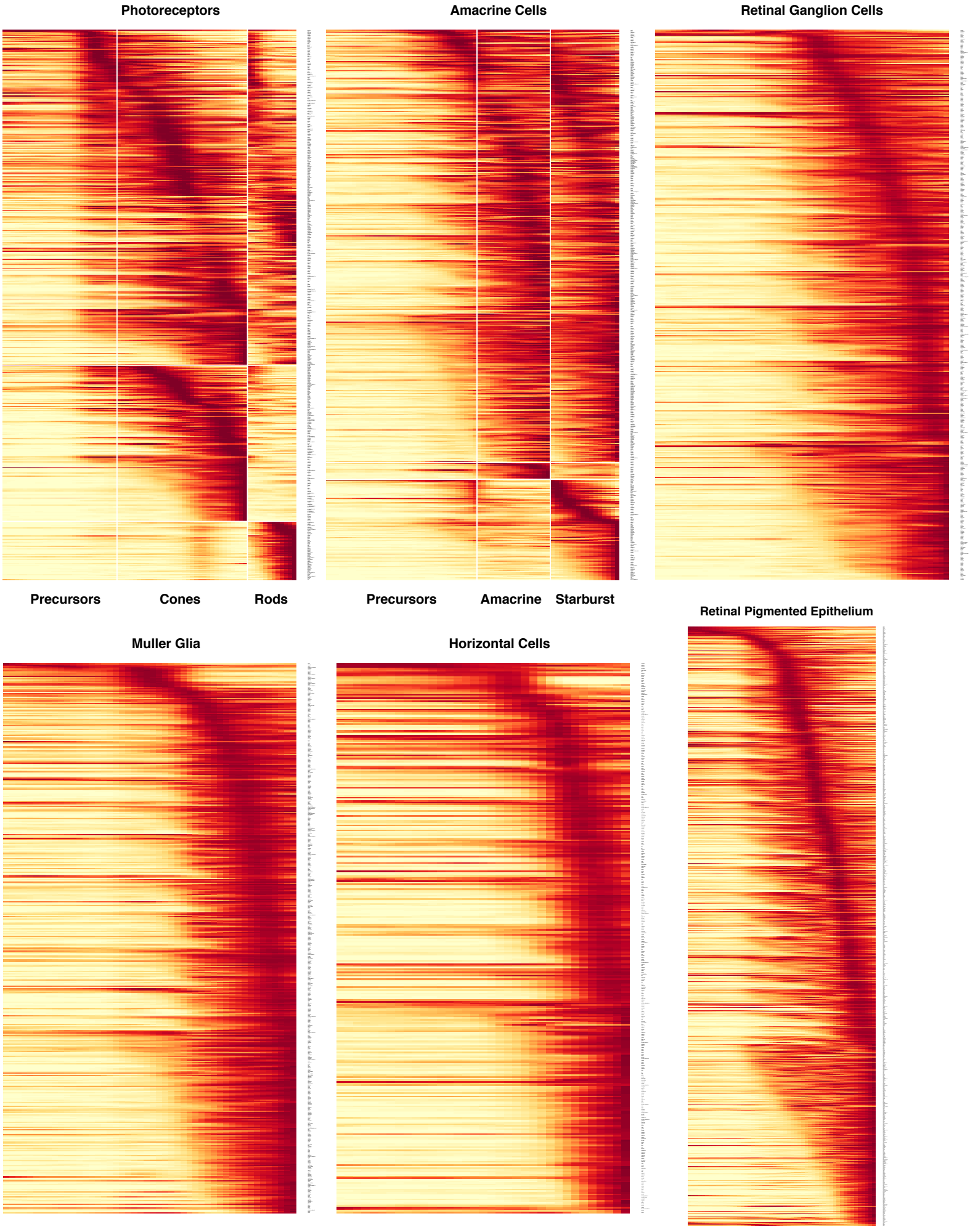
Sup Figure 6


A
Retina: Main Figure
B
Retina: 50% of cells, 1
Retina: 50% of cells, 2
Changed
Parent/Child
Same
NA
Pseudotime
CBP_3
CBP_6
CBP_1
CBP_4
CBP_5
CBP_2
Cones
Rods
Amacrine_1
Amacrine_2
RGC
Horizontal
Muller glia
RPE
C
Hypothalamus: Main Figure
D
Hypothalamus: 50% of cells, 1
Hypothalamus: 50% of cells, 2
GABA tac1+
synpr+
sst1.1+
tph2+
GABA dlx+
pdyn+
prdx1+

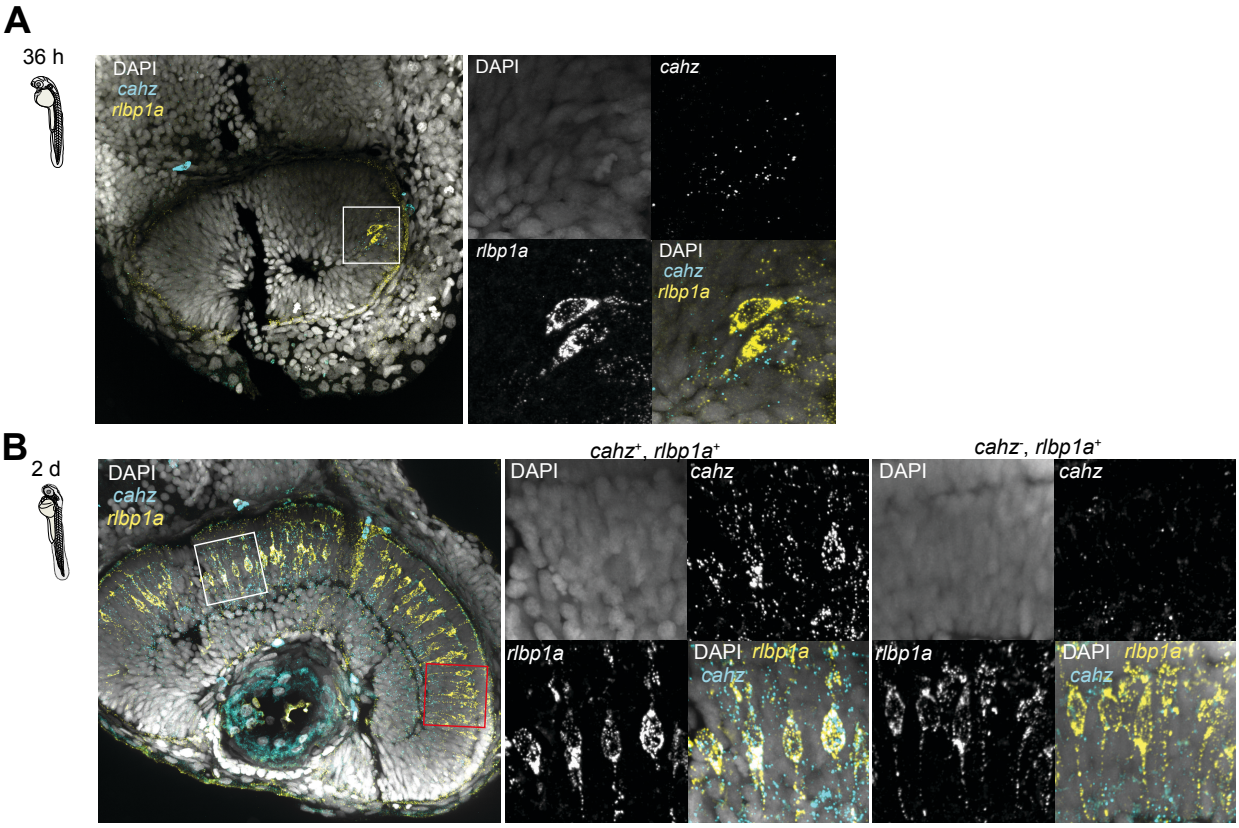
Sup Figure 7


foxd1
her15.1
her4.1
hes2.2
stx3a
dmbx1a
prkca
rx3
irx7
fezf2
ndrg1b
opn1mw1
gnat1
pbx1a
tfap2c
slc18a3a
rbpms2a
ompa
sdpra
rpe65a
Pseudotime
1 2 3 4 5 6 7 8 9 10 11 12 13 14

Sup Figure 8

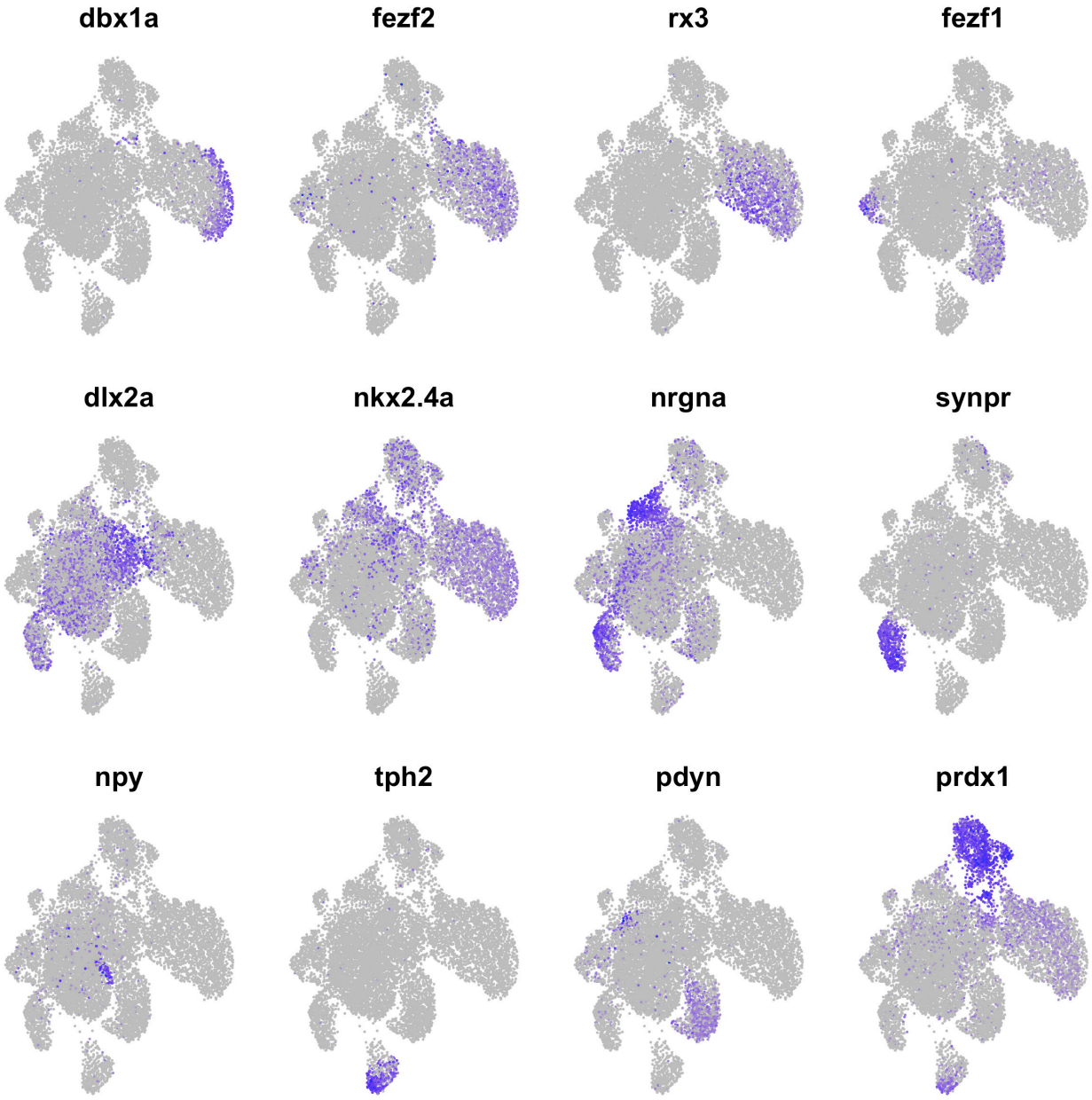

Photoreceptors
Precursors
Cones
Rods
Amacrine Cells
Precursors
Amacrine
Starburst
Retinal Ganglion Cells
Muller Glia
Horizontal Cells
Retinal Pigmented Epithelium

Sup Figure 9

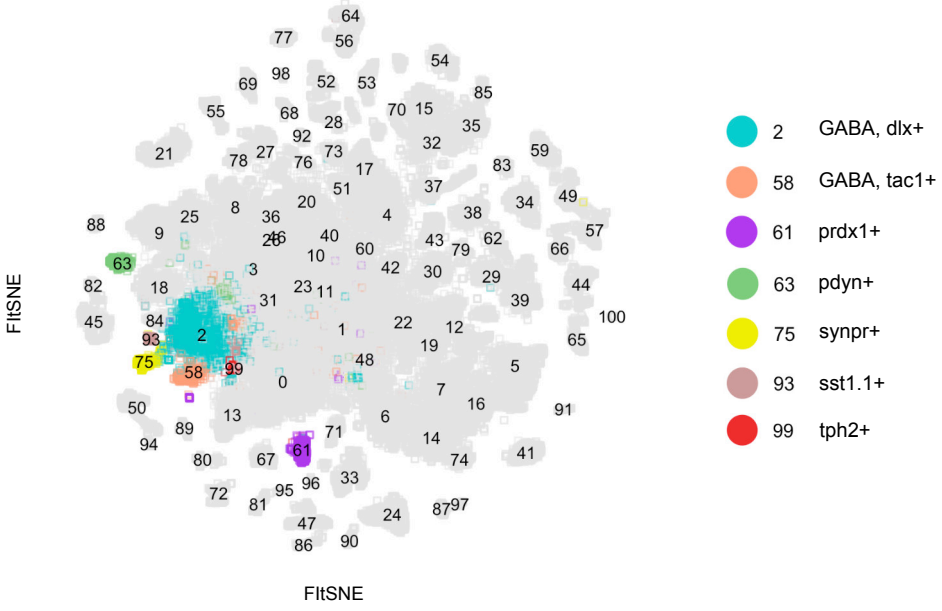

A
36 h
DAPI
cahz
rlbp1a
DAPI
cahz
rlbp1a
DAPI
cahz
rlbp1a
B
2 d
cahz+, rlbp1a+
cahz-, rlbp1a+
DAPI
cahz
rlbp1a
DAPI
cahz
rlbp1a
DAPI
rlbp1a
cahz
DAPI
cahz
rlbp1a
DAPI
rlbp1a
cahz
DAPI
rlbp1a
cahz

Sup Figure 10

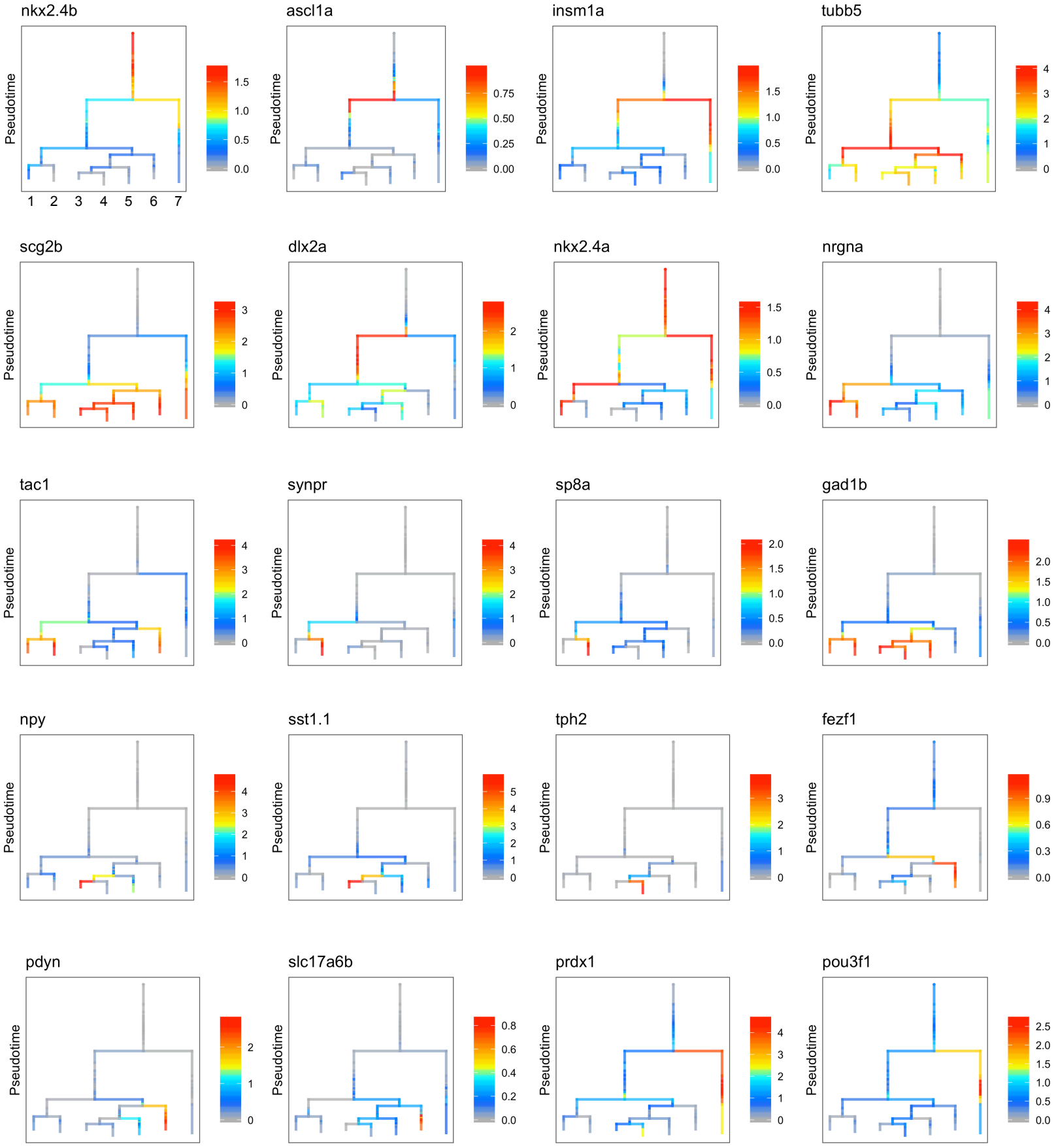
A



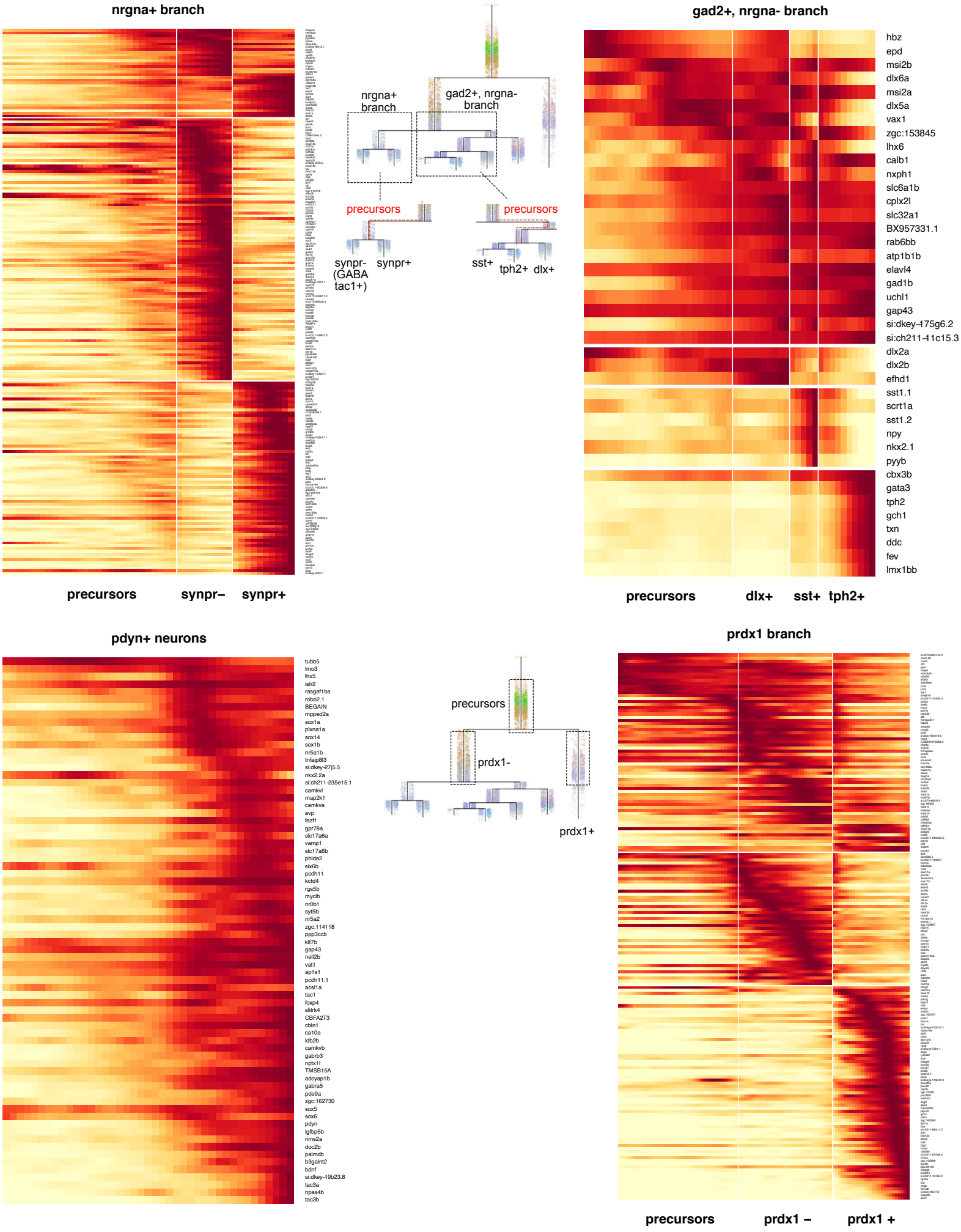
B



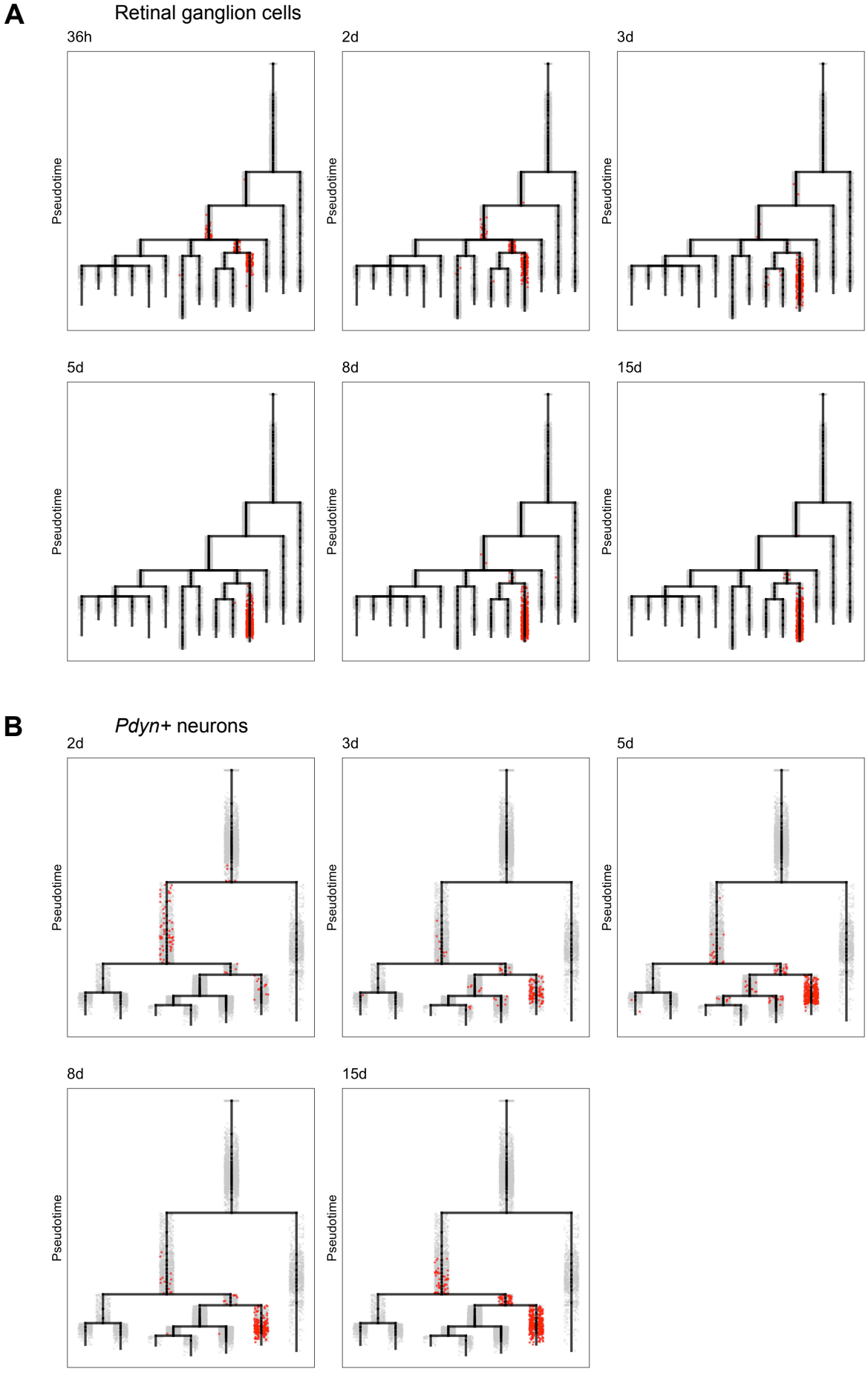
Sup Figure 11



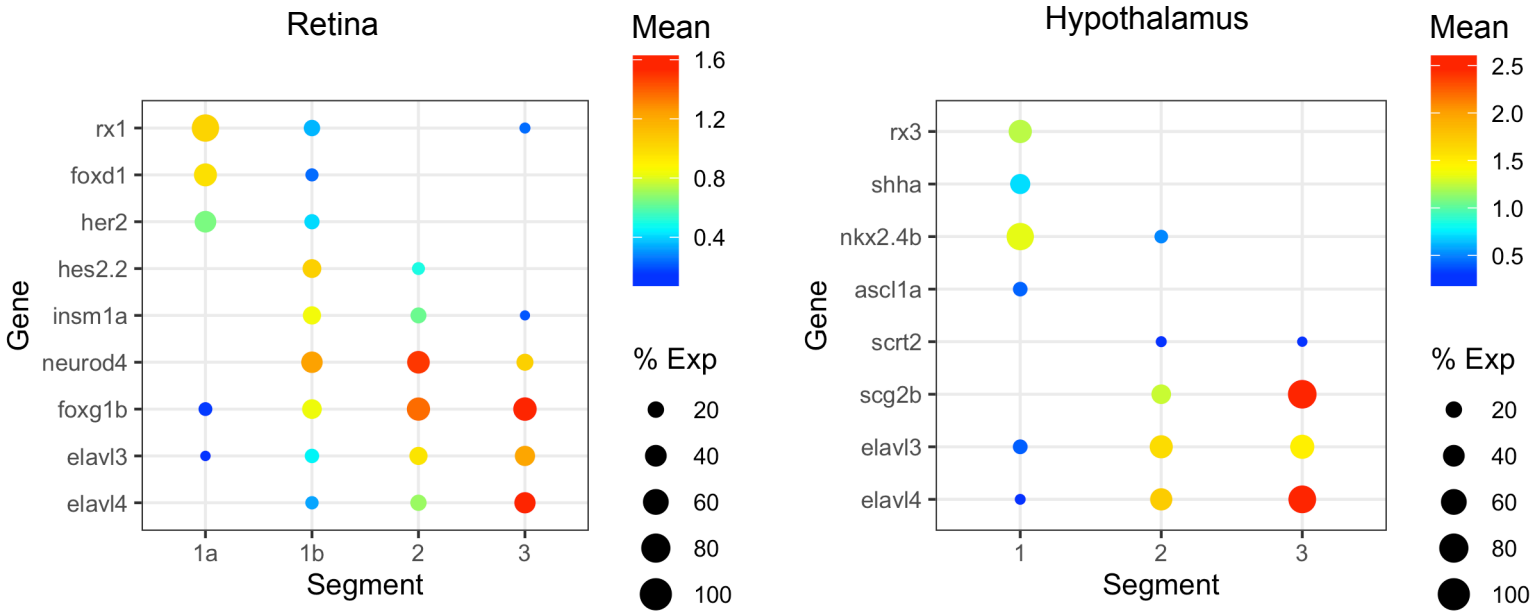
Sup Figure 12



Sup Figure 13



Sup Figure 14



Retina: 1 - URD object & doublet removal

Jeff Farrell

8/22/2019

Contents

Import data into URD

Convert Seurat object to URD

We first loaded a Seurat object that contained just cells from the clusters that belonged to the hypothalamus from each stage.

```
suppressPackageStartupMessages(library(URD))
suppressPackageStartupMessages(library(Seurat))
```

```
base.path <- "~/urd-cluster-bushra/"

# Load Seurat object that has been cropped to hypothalamus cells
object.seurat <- readRDS(paste0(base.path, "obj/retina.new_seurat.rds"))

# Convert to URD object
suburd <- seuratToURD(object.seurat)
```

Combined individual stage clustering

Bushra had performed individual clusterings across each stage with different resolutions. Here, it was better to create a single identifier that included stage + cluster information to combine all those clusterings (while preventing any overlap).

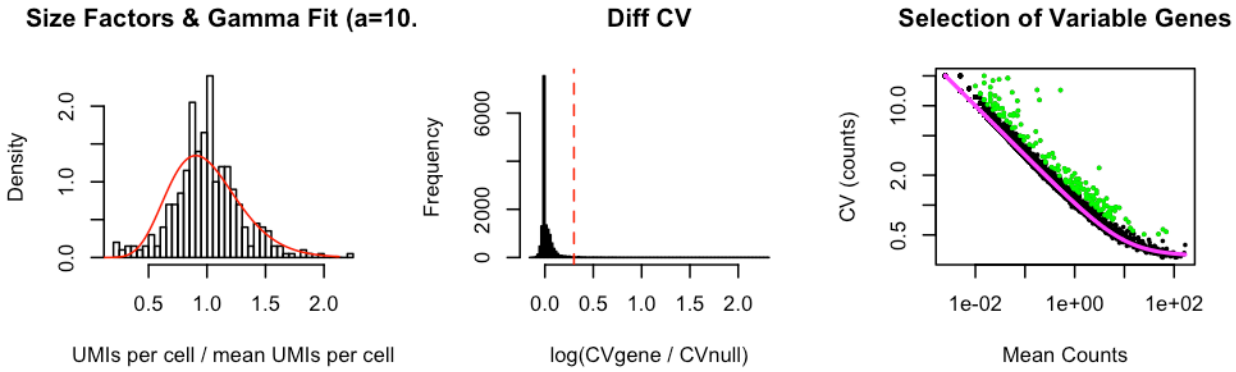
```
stages <- sort(unique(suburd@meta$stage))
clust.res.used <- paste0("res.", c("4.5", "4", "5", "5", "4.5", "5", "6",
    "6", "6", "5.5", "6", "5"))
names(clust.res.used) <- stages
suburd@group.ids$cluster <- NA
for (stage in stages) {
    suburd@group.ids[cellsInCluster(suburd, "stage", stage), "cluster"] <- paste0(stage,
        "-", suburd@group.ids[cellsInCluster(suburd, "stage", stage), clust.res.used[stage]])
}
```

Calculate highly variable genes

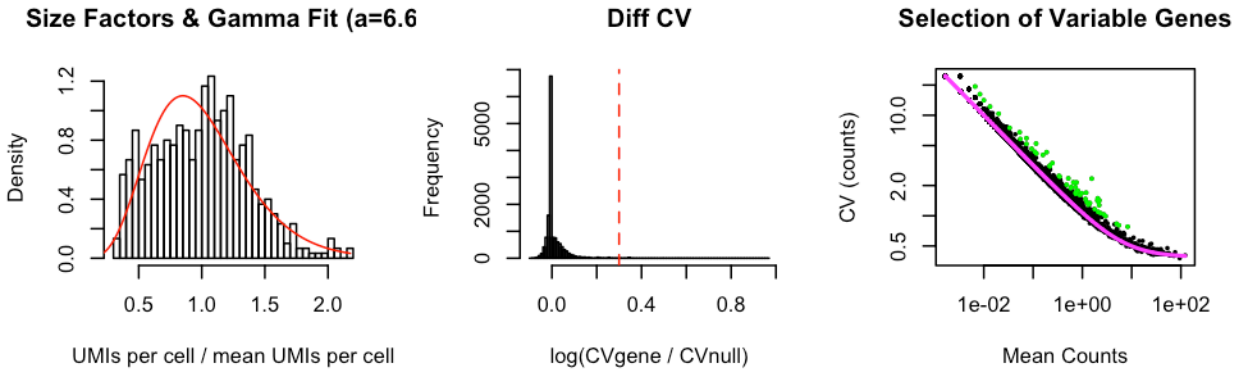
We calculated highly variable genes for each stage, used genes that were found as highly in at least two stages, but were not mitochondrial, ribosomal, heat-shock protein, or tandem duplicated genes.

```
# Calculated on each stage separaely, final gene list was all genes
# that were 'variable' in at least two stages NB: For a couple of
# stages, the gamma fit was poor -- the library size distribution
# seemed bimodal. Have seen this before in 10X data, but not sure what
# it means.
var.genes.by.stage <- lapply(stages, function(stage) {
    findVariableGenes(suburd, cells.fit = cellsInCluster(suburd, "stage",
        stage), set.object.var.genes = F, diffCV.cutoff = 0.3, main.use = stage,
        do.plot = T)
})
```

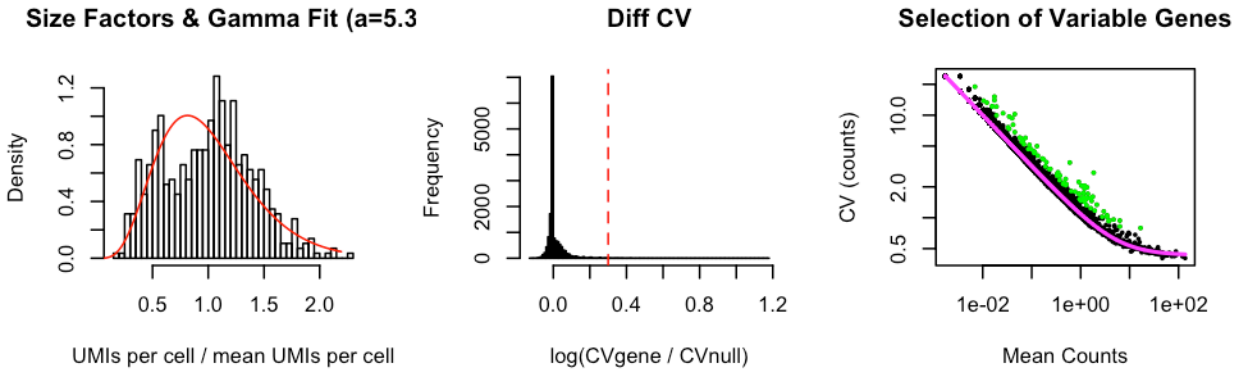
01-12h



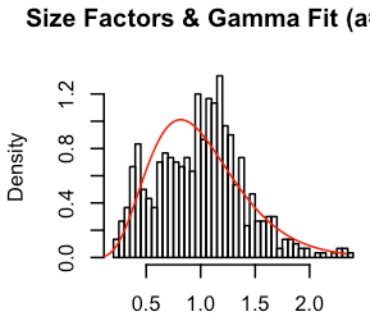
02-14h

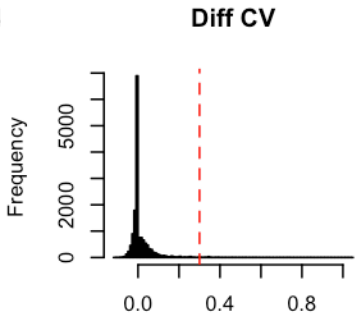


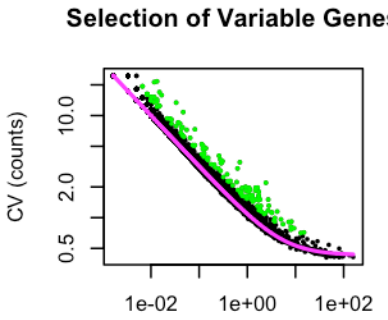
03-16h



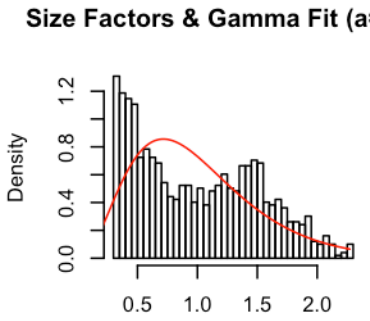
04-18h

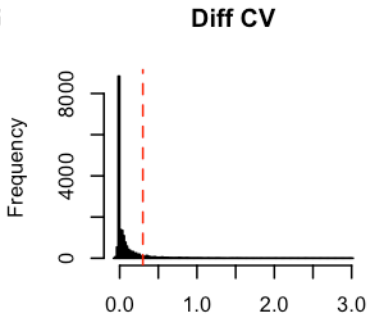


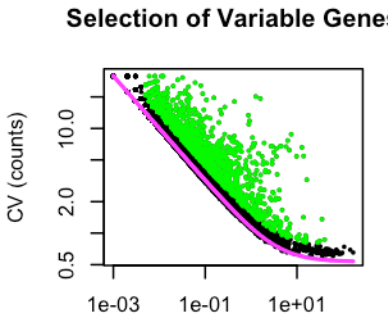




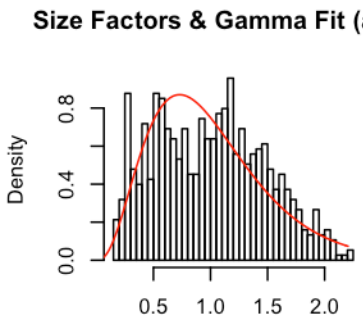
05-20h

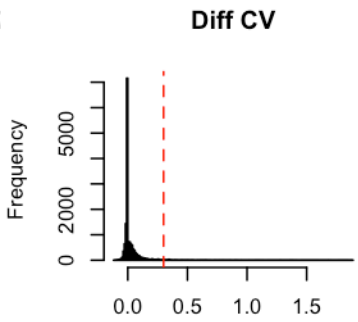


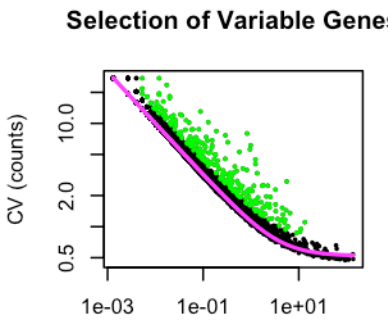




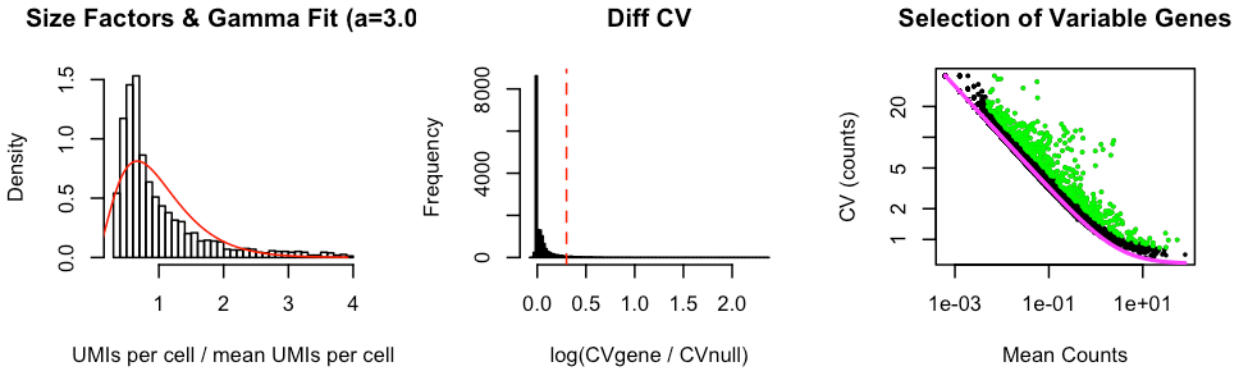
06-24h



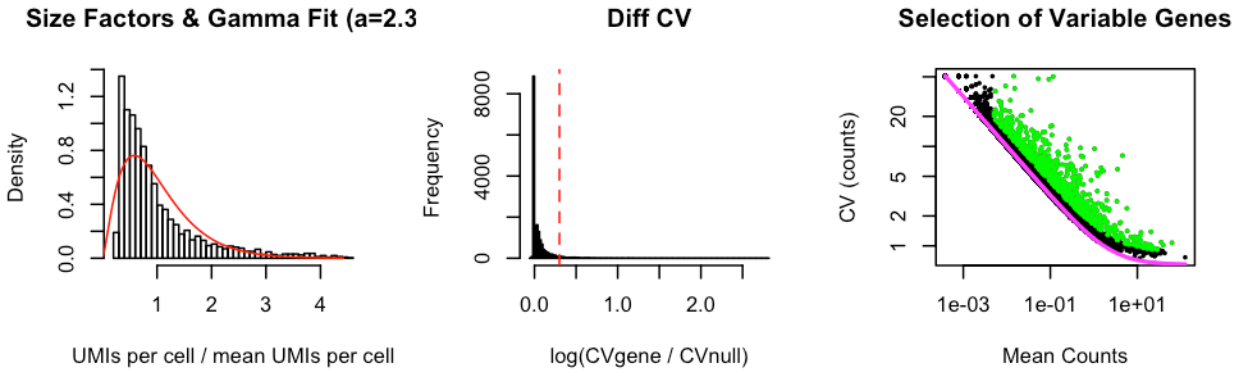




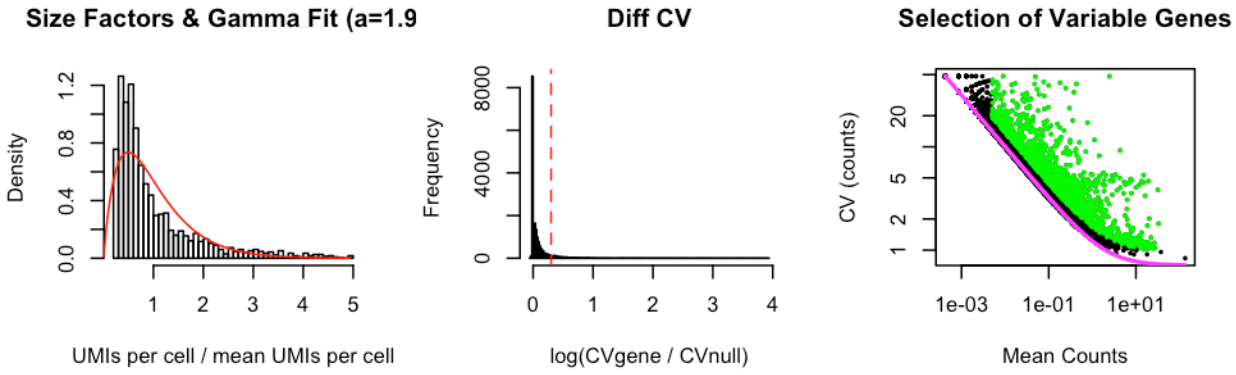
07-36h

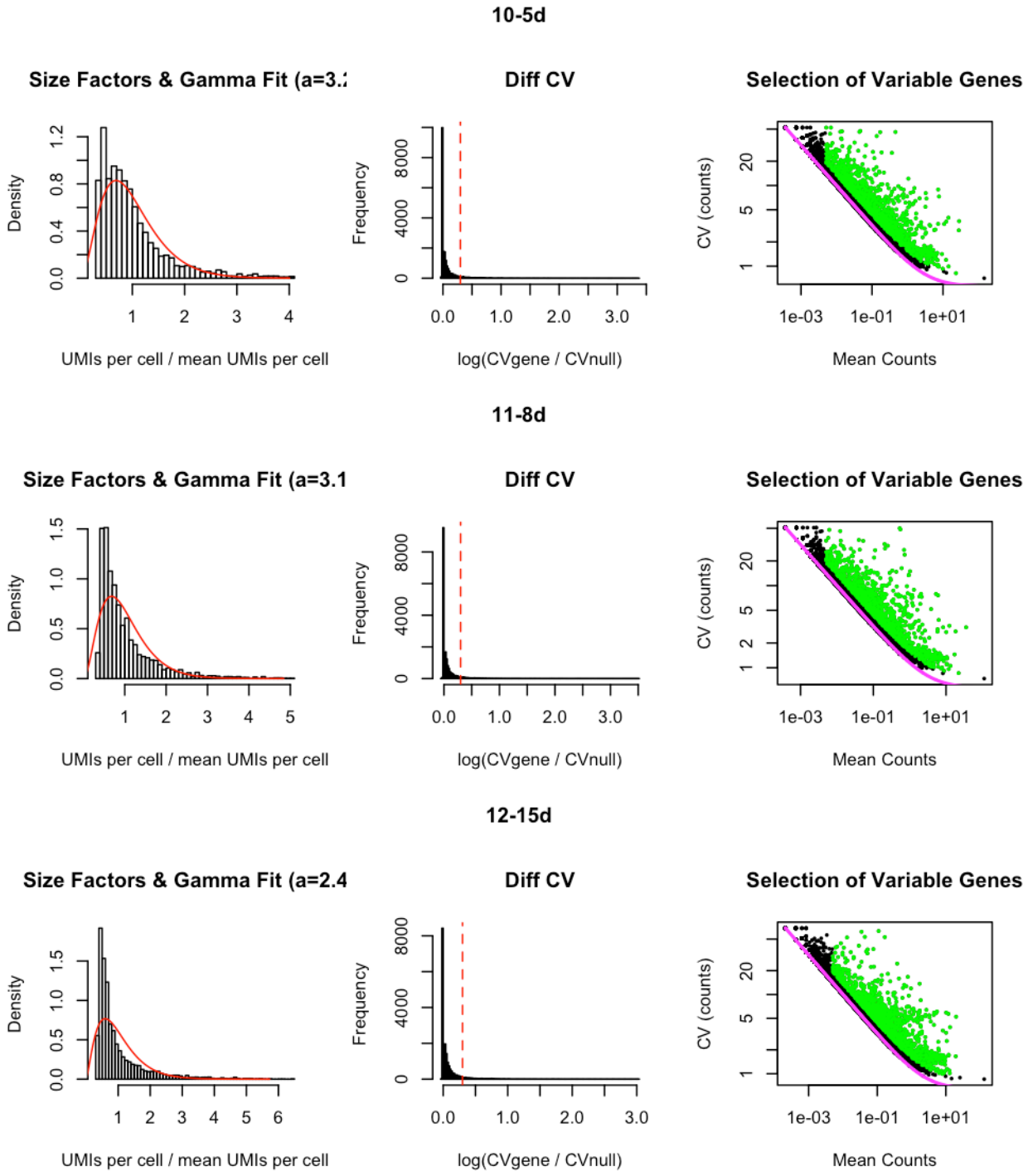


08-2d



09-3d






```
names(var.genes.by.stage) <- stages
var.genes <- sort(unique(unlist(var.genes.by.stage)))
print(paste0("Length of variable genes is ", length(var.genes)))
```

```
## [1] "Length of variable genes is 2636"
```

```
var.genes.twice <- names(which(table(unlist(var.genes.by.stage)) >= 2))
print(paste0("Length of variable genes shared across at least 2 stages is ",
    length(var.genes.twice)))
```

```
## [1] "Length of variable genes shared across at least 2 stages is 1724"
```

```
# Remove mitochondrial genes
var.mito <- grep("^mt-|^AC0", var.genes.twice, value = T)
# Remove ribosomal genes
var.ribo <- grep("^rps|^rpl", var.genes.twice, value = T)
# Remove hsp genes
var.hsp <- grep("^hsp", var.genes.twice, value = T)
# Remove genes with duplicates
var.dups <- grep("of many", var.genes.twice, value = T)
suburd@var.genes <- setdiff(var.genes.twice, c(var.mito, var.ribo, var.hsp,
    var.dups))
print(paste0("Length of final variable genes list (after removing mito, ribo, hsp genes) is ",
    length(suburd@var.genes)))
```

```
## [1] "Length of final variable genes list (after removing mito, ribo, hsp genes) is 1595"
```

To prevent downstream problems, we also removed any cells from the data that had the exact same expression of the variable genes (i.e. cells with completely duplicated coordinates in the high-dimensional space we would use for analysis downstream).

```
# Check for duplicate data points - cells with exact same expression of
# variable genes
vg.dups <- duplicated(as.data.frame(as.matrix(t(suburd@logupx.data[suburd@var.genes,
    ]))))
if (length(which(vg.dups)) > 0) {
    print(paste("Removing", length(which(vg.dups)), "cell(s) with duplicated variable gene expression."))
    not.dup.cells <- colnames(suburd@logupx.data)[!vg.dups]
    suburd <- urdSubset(suburd, not.dup.cells)
}
```

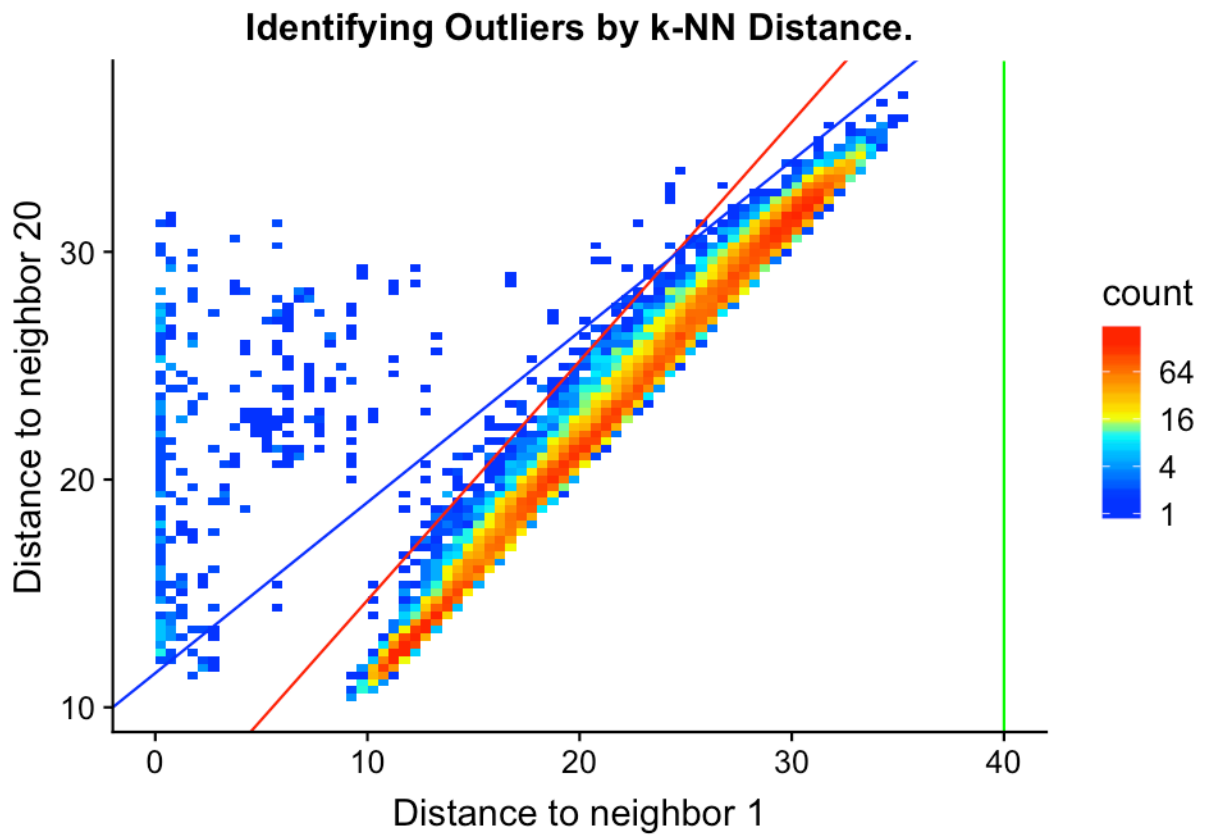
```
## [1] "Removing 6 cell(s) with duplicated variable gene expression."
```

Calculate KNN graph and remove outliers

We then calculated a k-nearest neighbor graph and removed cells that had unusual distance to their nearest neighbor, or unusual distance to their 20th nearest neighbor (given their distance to their nearest neighbor). These sorts of outliers often cause problems or skew diffusion maps (used downstream).

```
# Calculate k-nn
suburd <- calcKNN(suburd)
```

```
# Check what the outliers are
outliers <- knnOutliers(suburd, nn.1 = 1, nn.2 = 20, x.max = 40, slope.r = 1.05,
    int.r = 4.2, slope.b = 0.75, int.b = 11.5, title = "Identifying Outliers by k-NN Distance.")
```




```
length(outliers)
```

```
## [1] 521
```

```
suburd <- urdSubset(suburd, cells.keep = setdiff(colnames(suburd@logupx.data),
    outliers))
```

Remove ***kidins220a+*** population

A cell cluster was observed in 15 dpf that was positive for expression of kidins220a, and foxg1b (which is exclusive to the retina). However, no similar clusters were observed in other stages, suggesting that we did not recover the progenitors of this population, so we excluded it from the URD analysis.

```
suburd <- urdSubset(suburd, cells.keep = setdiff(colnames(suburd@logupx.data),
    cellsInCluster(suburd, "cluster", "12-15d-96")))
```

Remove cell type doublets

Add UMAP projection

While not strictly required, a UMAP projection can make it easier to assess the expression of NMF modules and whether thresholds for overlap are set correctly.

```
## add UMAP command

# Load pre-calculated UMAP
umap <- readRDS(paste0(base.path, "/umap/umap_retina.rds"))

# Add projection to URD object
suburd@tsne.y <- umap[colnames(suburd@logupx.data), ]
```

Load NMF results and import into object

NMF results were calculated by providing `suburd@logupx.data` to an external NMF pipeline written in Python. The output results are imported here, scaled, and added to the URD object.

```
# Load the NMF results
load(paste0(base.path, "/NMF/retina/result_tbls.Robj"))

# The results object contains NMF runs for several K values.  k=45 was
# chosen for this tissue, so this extracts the results for that
# particular parameter
k.use <- "45"
nmf.cells <- result_obj[[paste0("K=", k.use)]][[1]]$C
rownames(nmf.cells) <- paste0("nmf", 1:nrow(nmf.cells))
colnames(nmf.cells) <- gsub("\\.", "-", colnames(nmf.cells))
nmf.genes <- result_obj[[paste0("K=", k.use)]][[1]]$G
colnames(nmf.genes) <- paste0("nmf", 1:nrow(nmf.cells))

# Some stages were subsampled in the original object and accidentally
# cropped out cells that had scGESTALT barcodes.  Those were added back
# in, and their expression was decomposed with original NMF gene matrix
# to give an additional NMF cell matrix for those cells.
new.nmf.c <- read.csv(paste0(base.path, "/NMF/retina/retina_new_nmfC_k45.csv"),
    row.names = 1)
rownames(new.nmf.c) <- paste0("nmf", 1:nrow(new.nmf.c))
colnames(new.nmf.c) <- gsub("\\.", "-", colnames(new.nmf.c))

# Combine old and new NMF results
nmf.cells <- cbind(nmf.cells, new.nmf.c)

# Trim NMF results to match cells in current object
nmf.cells <- nmf.cells[, colnames(suburd@logupx.data)]

# Scale NMF results 0-1
nmf.cells.scaled <- sweep(nmf.cells, 1, apply(nmf.cells, 1, max), "/")

# Add scaled NMF results to the URD object
suburd@nmf.c1 <- as(t(as.matrix(nmf.cells.scaled)), "dgCMatrix")
```

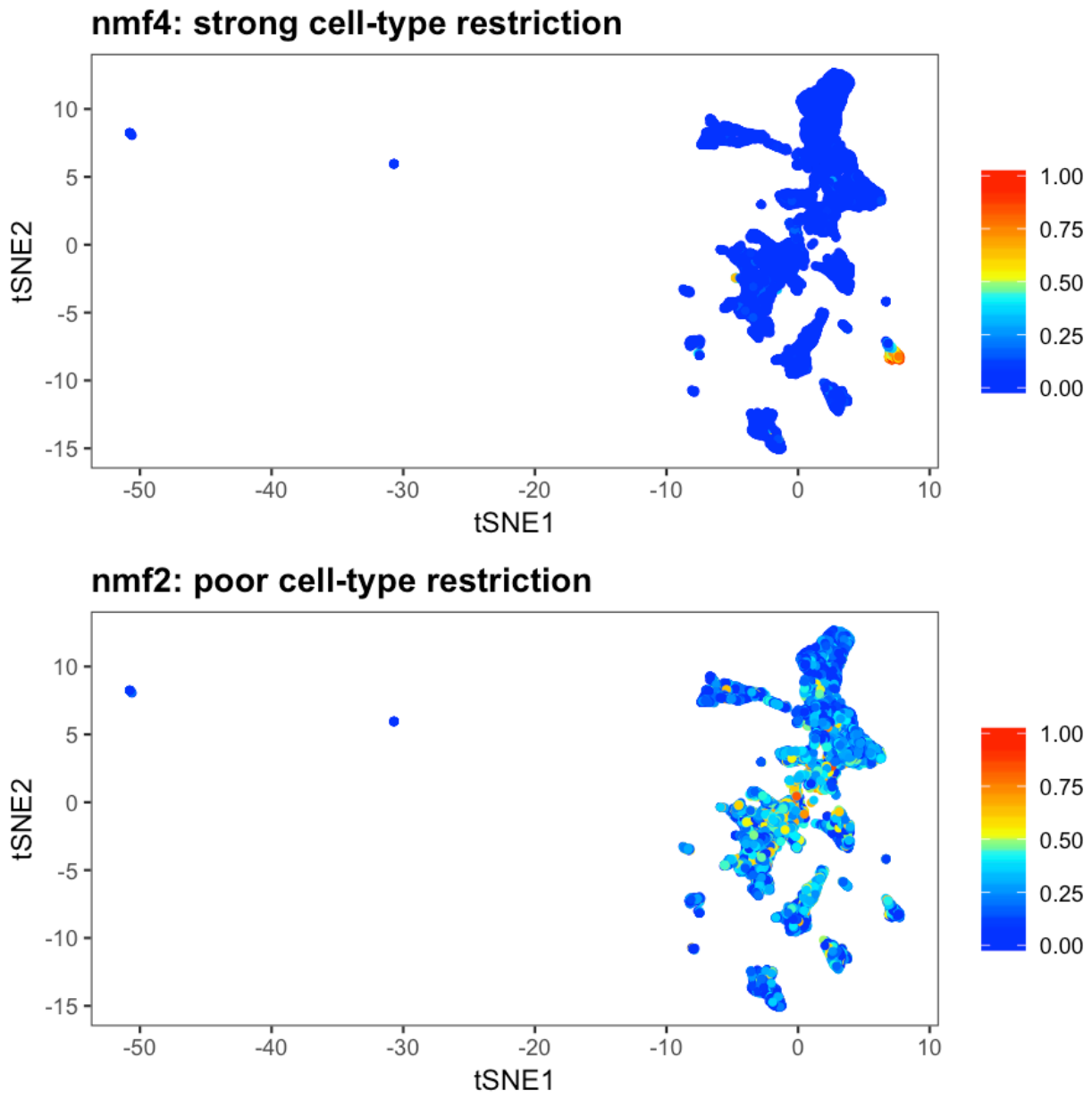
Select cell-type specific modules

Several NMF modules will be poor markers of cell types — these are often modules driven mostly be the expression of 1-2 genes (where the gene loading of the first gene is much greater than that

of the fourth gene, for instance), or modules that don't exhibit any restriction in a tSNE or UMAP projection.

```
# Plot size parameters
plot.height = 8
plot.width = 8
dpi = 150

# Plot every module to determine which exhibit cell-type specificity
# This saves directly to the hard drive: two example plots are shown
# below.

# for (n in colnames(suburd@nmf.c1)) { png(paste0(path, '/doublets/',
# subset, '-plots/', n, '.png'), width=dpi*plot.width,
# height=dpi*plot.height) plot(plotDim(suburd, n)) dev.off() }

gridExtra::grid.arrange(grobs = list(plotDim(suburd, "nmf4", plot.title = "nmf4: strong cell-type restriction"),
    plotDim(suburd, "nmf2", plot.title = "nmf2: poor cell-type restriction")),
    ncol = 1)
```

```
## Warning: Removed 520 rows containing missing values (geom_point).

## Warning: Removed 520 rows containing missing values (geom_point).
```

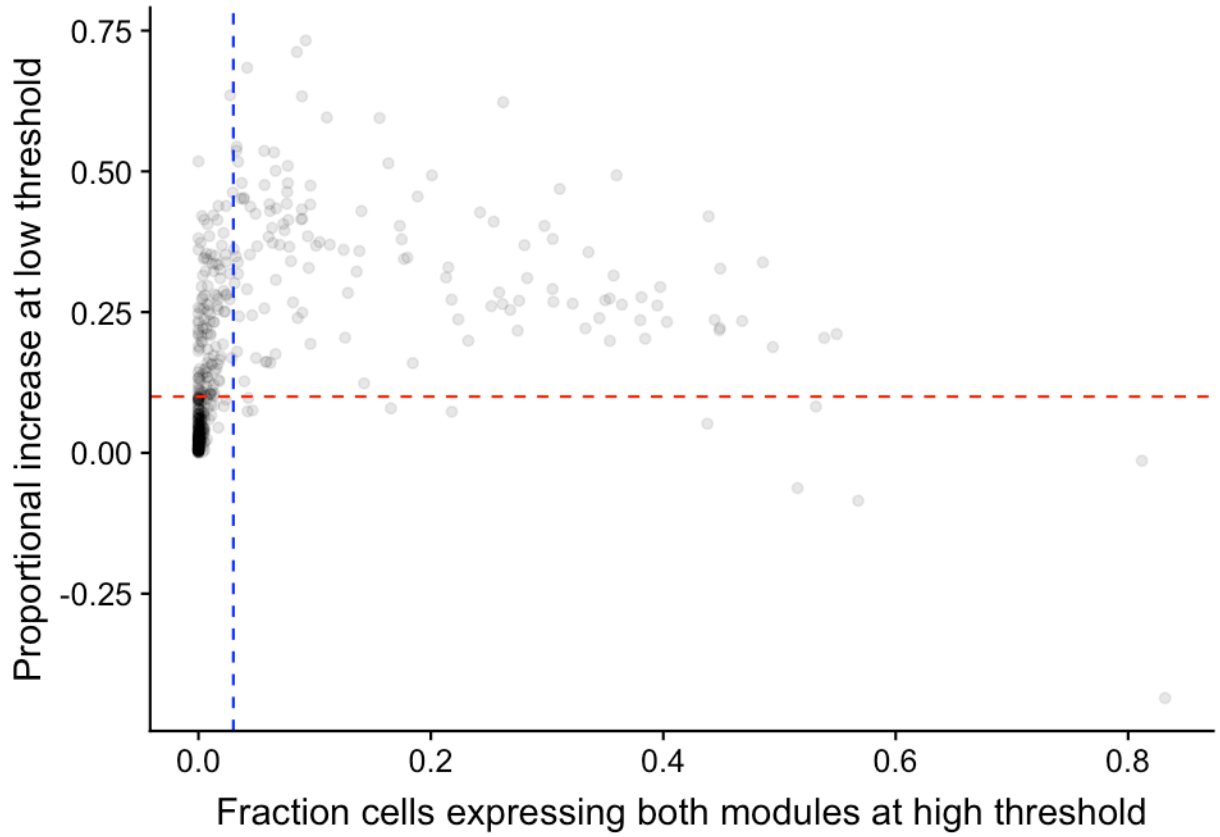



```
# Module Gene 1 : Gene 4 Ratios
top.genes <- result_obj[[paste0("K=", k.use)]][[1]]$top30genes
top.weights <- top.genes[, grep("Weights", colnames(top.genes), value = T)]
colnames(top.weights) <- paste0("nmf", 1:nrow(nmf.cells))
top.weights.ratio <- top.weights[1, ]/top.weights[4, ]

# Which modules exhibit cell-type restriction?
modules.ok.ratio <- names(top.weights.ratio)[which(top.weights.ratio <
    5)]
restricted.modules <- paste0("nmf", c(4:5, 8:13, 15:18, 21:29, 31:35, 37:39,
    41:44))
good.modules <- intersect(modules.ok.ratio, restricted.modules)
```

Determine which module pairs to use for doublet removal

We consider NMF modules pairwise and only use those pairs that don't are non-overlapping in the data. (In other words, NMF modules that are mutually exclusive in the majority of the data.) Here, we determine thresholds for selecting those module pairs.

```
# Determine overlaps between module pairs
nmf.doublet.combos <- NMFDoubletsDefineModules(suburd, modules.use = good.modules,
    module.thresh.high = 0.4, module.thresh.low = 0.15)

# Determine thresholds for NMF modules
frac.overlap.max = 0.03
frac.overlap.diff.max = 0.1
module.expressed.thresh = 0.33

# Determine which module pairs to use for doublets
NMFDoubletsPlotModuleThresholds(nmf.doublet.combos, frac.overlap.max = frac.overlap.max,
    frac.overlap.diff.max = frac.overlap.diff.max)
```

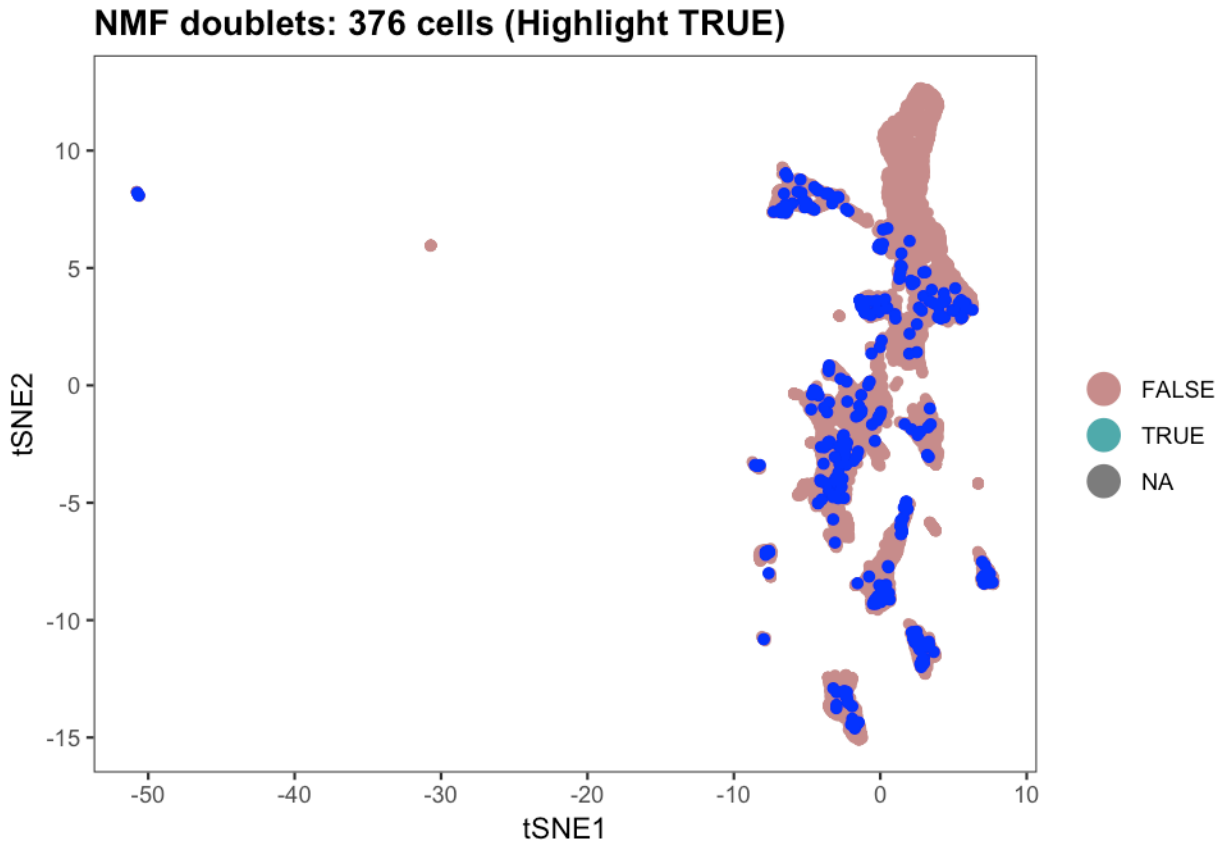



```
# These commands save plots directly to the hard-drive.

# Make plots to see how your thresholds are
NMFDoubletsPlotModuleCombos(suburd, path = paste0(path, "/doublets/", subset,
    "-doublet-combos/"), module.combos = nmf.doublet.combos, module.expressed.thresh = module.expressed.thresh,
    frac.overlap.max = frac.overlap.max, frac.overlap.diff.max = frac.overlap.diff.max,
    boundary = "pass", sort = "near", n.plots = 25)
NMFDoubletsPlotModuleCombos(suburd, path = paste0(path, "/doublets/", subset,
```

```
    "-ok-combos/"), module.combos = nmf.doublet.combos, module.expressed.thresh = module.expressed.thresh,
    frac.overlap.max = frac.overlap.max, frac.overlap.diff.max = frac.overlap.diff.max,
    boundary = "discarded", sort = "near", n.plots = 25)
```

```
# Define doublet cells
nmf.doublets <- NMFDoubletsDetermineCells(suburd, nmf.doublet.combos, module.expressed.thresh = module.expressed.
    frac.overlap.max = frac.overlap.max, frac.overlap.diff.max = frac.overlap.diff.max)  # 376 cells / 19837 cell

# Plot doublet cells on the UMAP
suburd <- groupFromCells(suburd, "nmf.doublets", cells = nmf.doublets)
plot(plotDimHighlight(suburd, clustering = "nmf.doublets", cluster = "TRUE",
    plot.title = paste0("NMF doublets: ", length(nmf.doublets), " cells"),
    point.size = 2, highlight.color = "blue"))
```

```
## Warning: Removed 520 rows containing missing values (geom_point).
```




```
# Crop object to exclude doublets
suburd.cropped <- urdSubset(suburd, cells.keep = setdiff(colnames(suburd@logupx.data),
    nmf.doublets))
```

And then save the completed object for use downstream in building a tree using URD.

```
saveRDS(suburd.cropped, file = paste0(base.path, "/obj/URD_retina_ND.rds"))
```

Retina: 2 - URD tree

Jeff Farrell

9/07/2019

Contents

Load data

```
suppressPackageStartupMessages(library(URD))
suppressPackageStartupMessages(library(Seurat))
```

```
base.path <- "~/urd-cluster-bushra/"

# Load procesed URD object
object <- readRDS(paste0(base.path, "obj/URD_retina_ND.rds"))
```

Processed on the cluster

Most of the following steps were run on a computing cluster. These individual tissue subsets can be run on a modern, well-equipped laptop. The use of a computing cluster allows multiple parameter choices to be tried in parallel, and also allows further parallelization of the random walk procedure, speeding it up. Below, we should the commands that one would run on their laptop, and then generally load the pre-processed results from the cluster that were used in the paper. If you want to parallelize your own processing on a compute cluster, the scripts we used will be available at http://github.com/farrellja/URD/cluster/

Calculate diffusion map and pseudotime

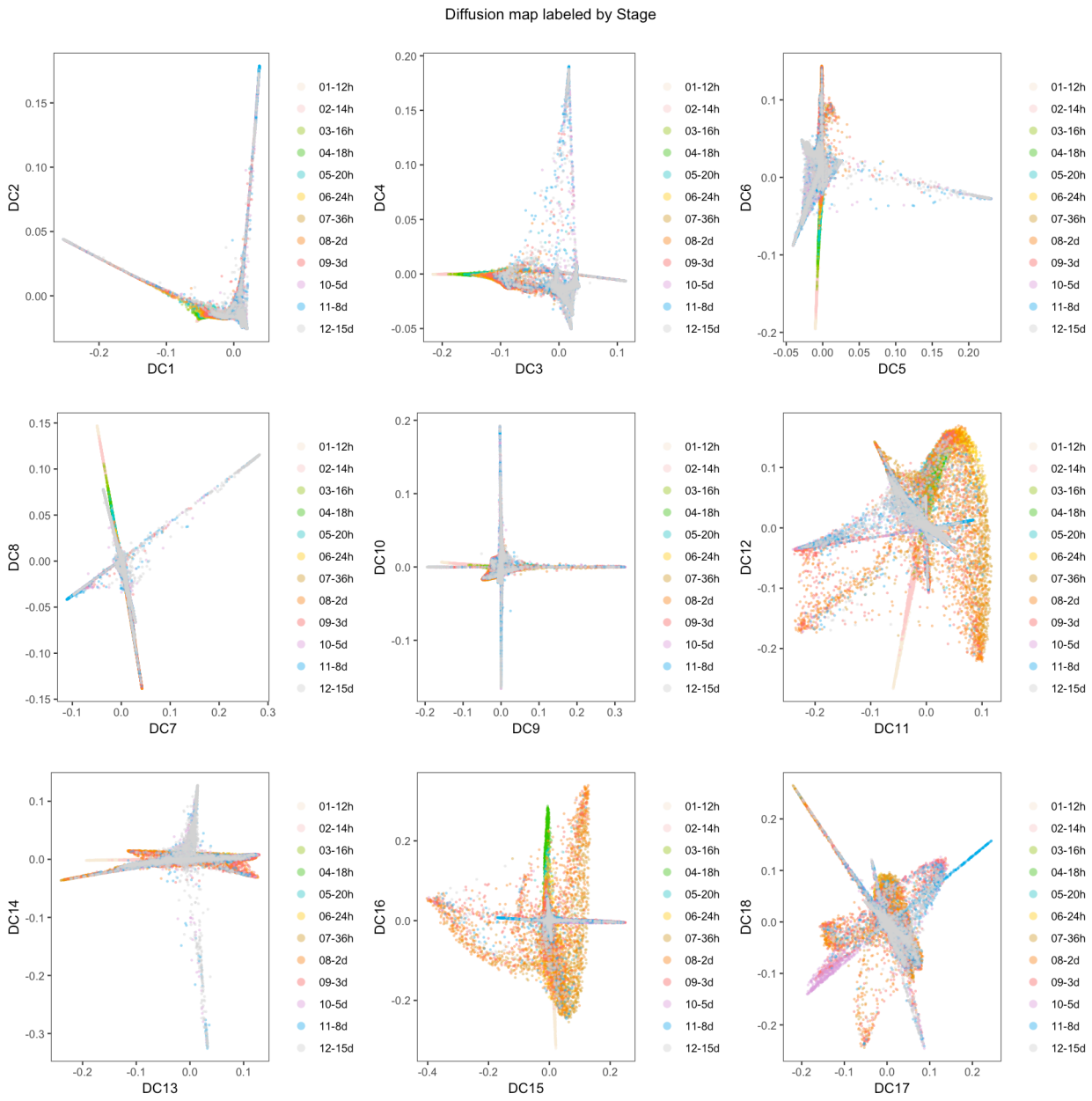
These two steps are run in the cluster script URD-DM-PT.R.

Calculate diffusion map

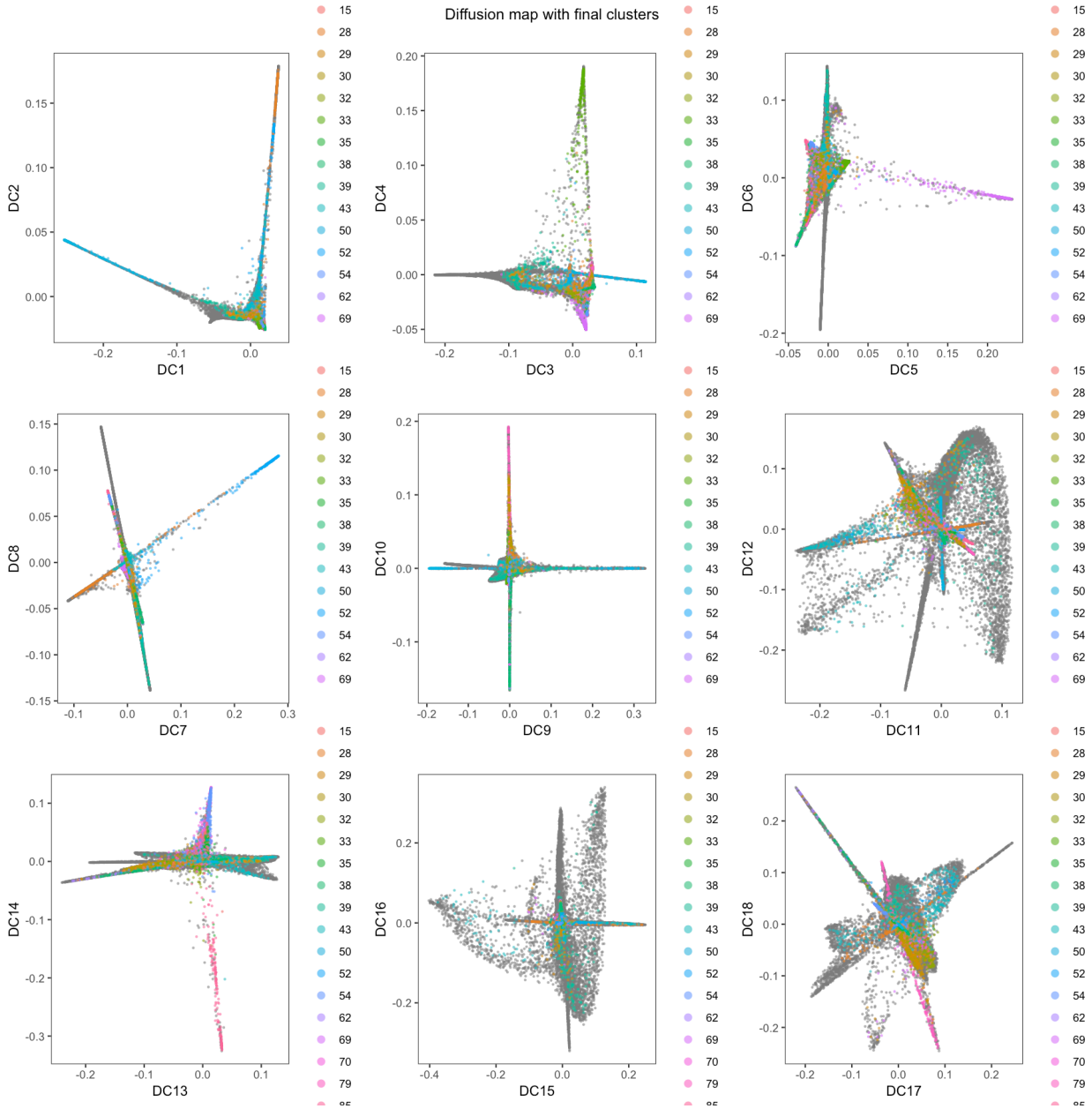
```
# To run locally: Calculate a diffusion map projection
object <- calcDM(object, knn = 140, sigma.use = 14)
```

```
# Or: Load a pre-computed diffusion map projection
dm <- readRDS(paste0(base.path, "dm/dm_retinanewnokND_knn-140_sigma-14.rds"))
object <- importDM(object, dm)
```

```
# Plot diffusion maps
stage.colors <- c("antiquewhite", "#FFCCCC", "#99CC00", "#33CC00", "cyan3",
    "gold", "goldenrod", "darkorange", "indianred1", "plum", "deepskyblue2",
    "lightgrey")

# Plot by stage
plotDimArray(object = object, reduction.use = "dm", dims.to.plot = 1:18,
    label = "stage", plot.title = "", outer.title = "Diffusion map labeled by Stage",
    legend = T, alpha = 0.45, discrete.colors = stage.colors)
```




```
# Plot with final cell types labeled
object@group.ids$final.cluster <- NA
object@group.ids[cellsInCluster(object, "stage", "12-15d"), "final.cluster"] <- object@group.ids[cellsInCluster(o
    "stage", "12-15d"), "res.5"]
plotDimArray(object = object, reduction.use = "dm", dims.to.plot = 1:18,
    label = "final.cluster", plot.title = "", outer.title = "Diffusion map with final clusters",
    legend = T, alpha = 0.6)
```

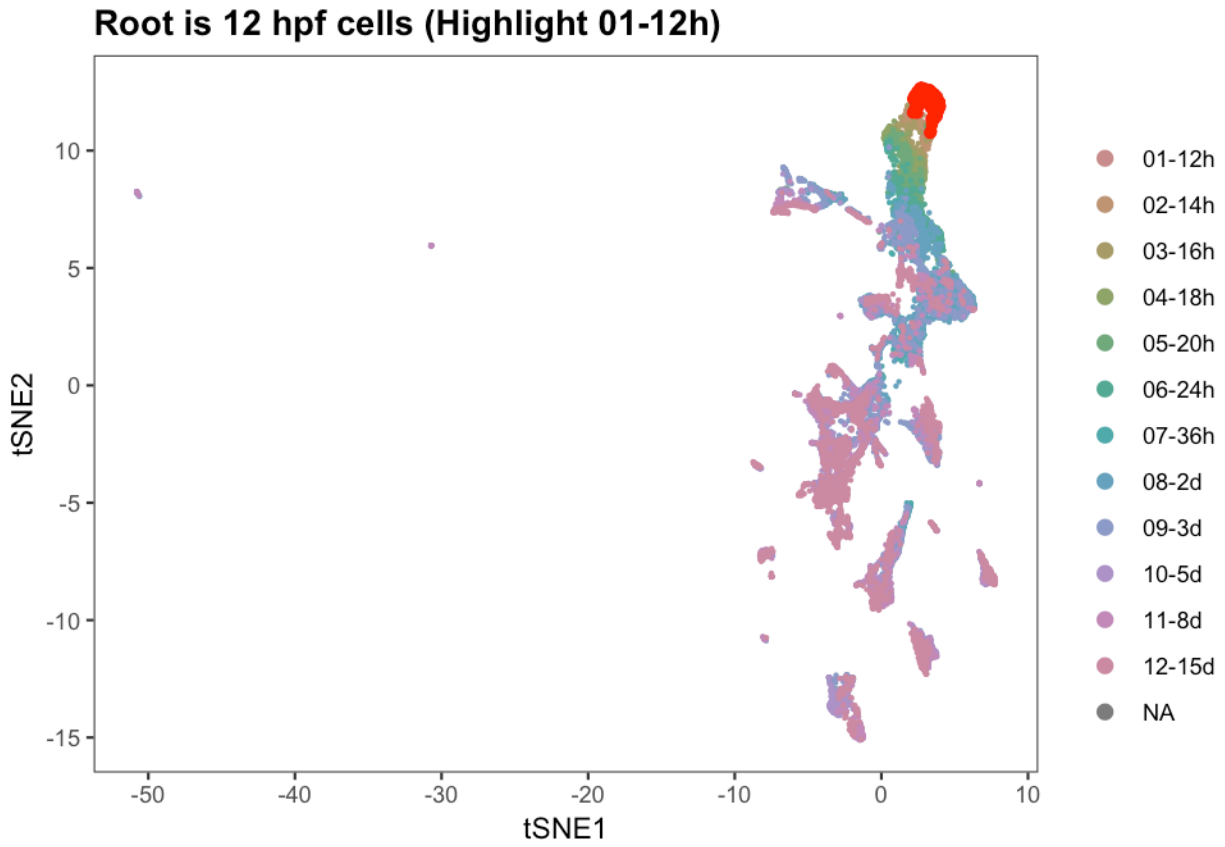


Calculate pseudotime

URD requires a starting point or 'root' for determining pseudotime. Here, we used all cells from the first timepoint (i.e. 12 hpf) as the root.

```
# Here, we used all cells from the first timepoint (i.e. 12 hours) as
# the root.
root.cells <- cellsInCluster(object, "stage", "01-12h")
plotDimHighlight(object, "stage", "01-12h", plot.title = "Root is 12 hpf cells")
```

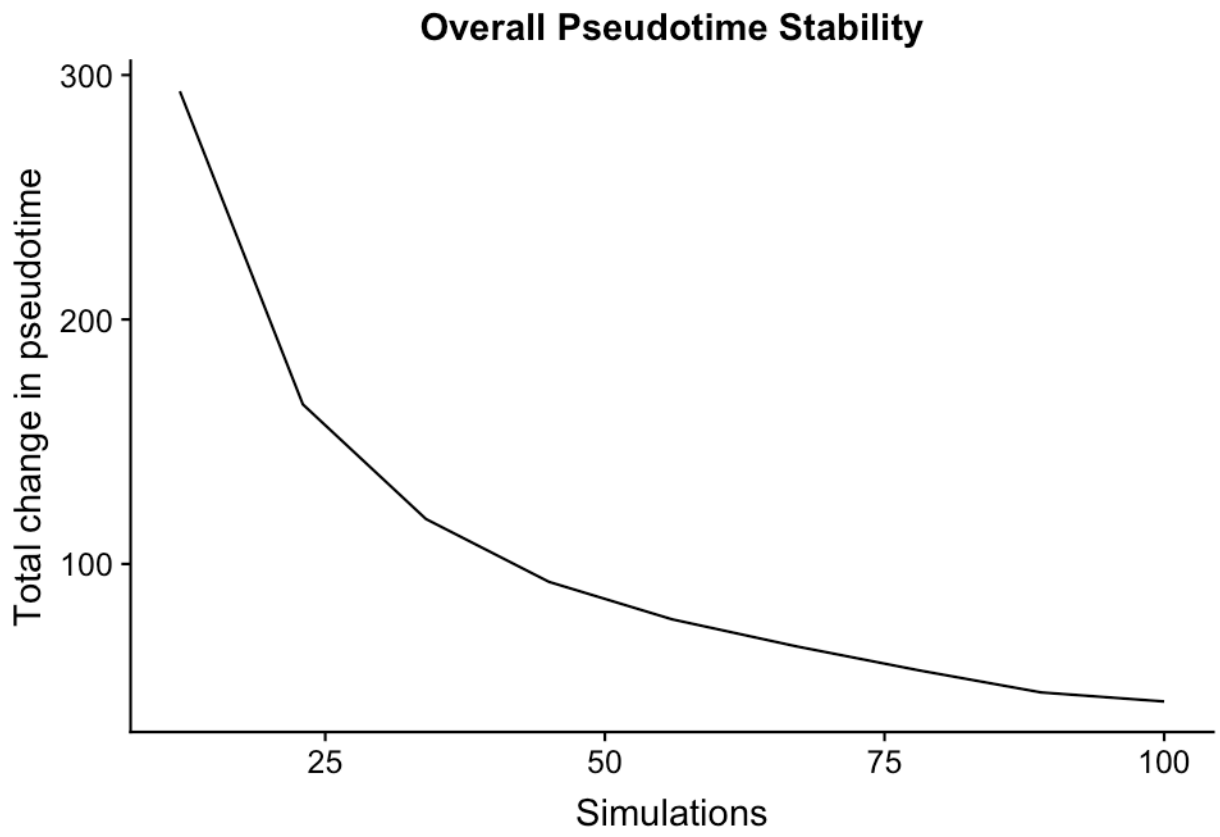
```
## Warning: Removed 500 rows containing missing values (geom_point).
```



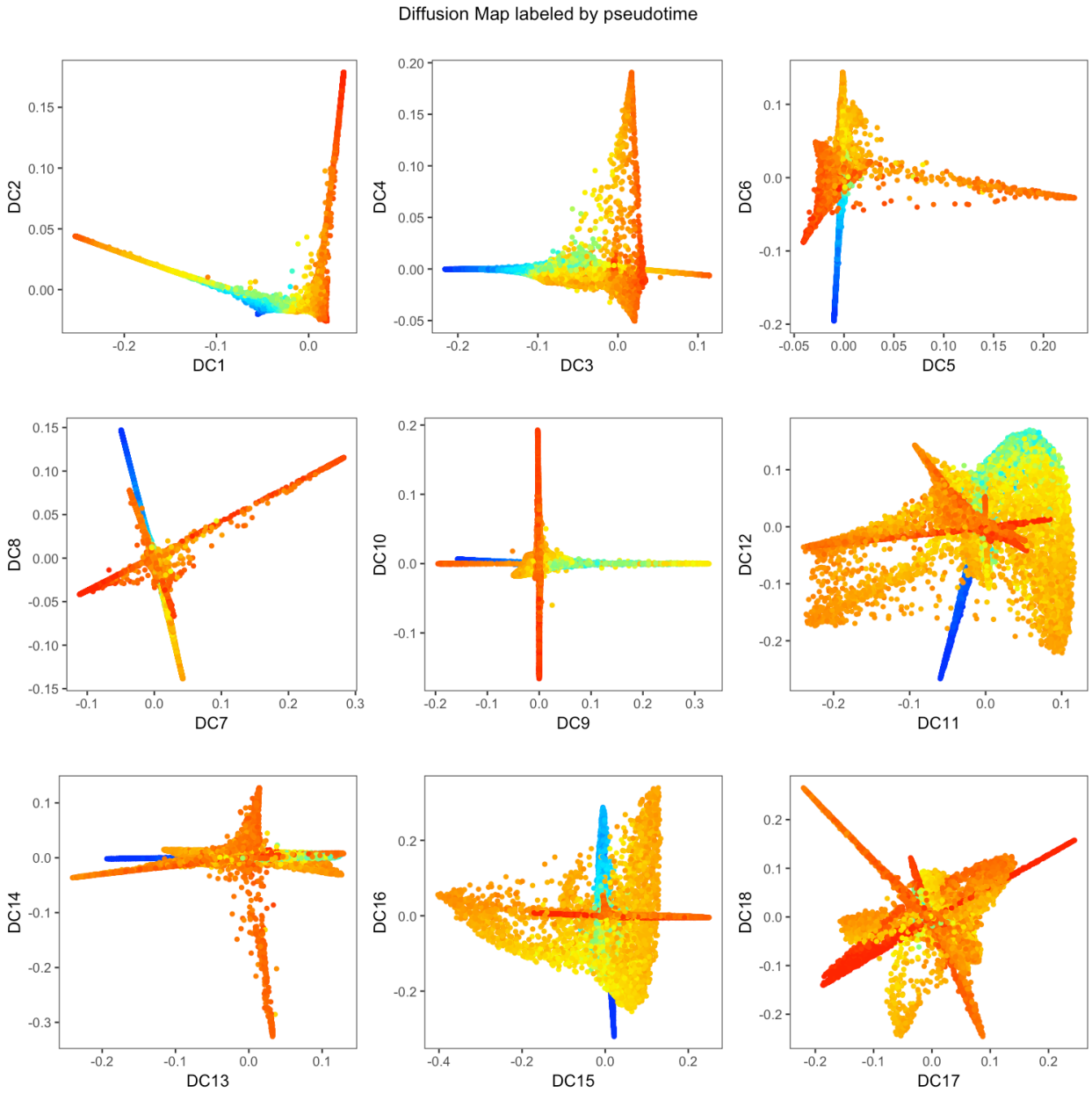

```
# To run locally: Run graph-search simulations to determine pseudotime
flood.result <- floodPseudotime(object, root.cells = root.cells, n = 100,
    minimum.cells.flooded = 2, verbose = T)
```

```
# Or load a pre-computed graph-search simulation result
flood.result <- readRDS(paste0(base.path, "flood/flood_retinanewnokND_knn-140_sigma-14.rds"))
```

```
# Process the graph-search simulations to determine the pseudotime of
# each cell
object <- floodPseudotimeProcess(object, flood.result, floods.name = "pseudotime",
    max.frac.NA = 0.4, pseudotime.fun = mean, stability.div = 10)

# If enough simulations have been run, then as additional simulations
# are added, the overall change in pseudotime of cells should reach an
# asymptote. If it does not, then floodPseudotime should be run with a
# higher n.
pseudotimePlotStabilityOverall(object)
```

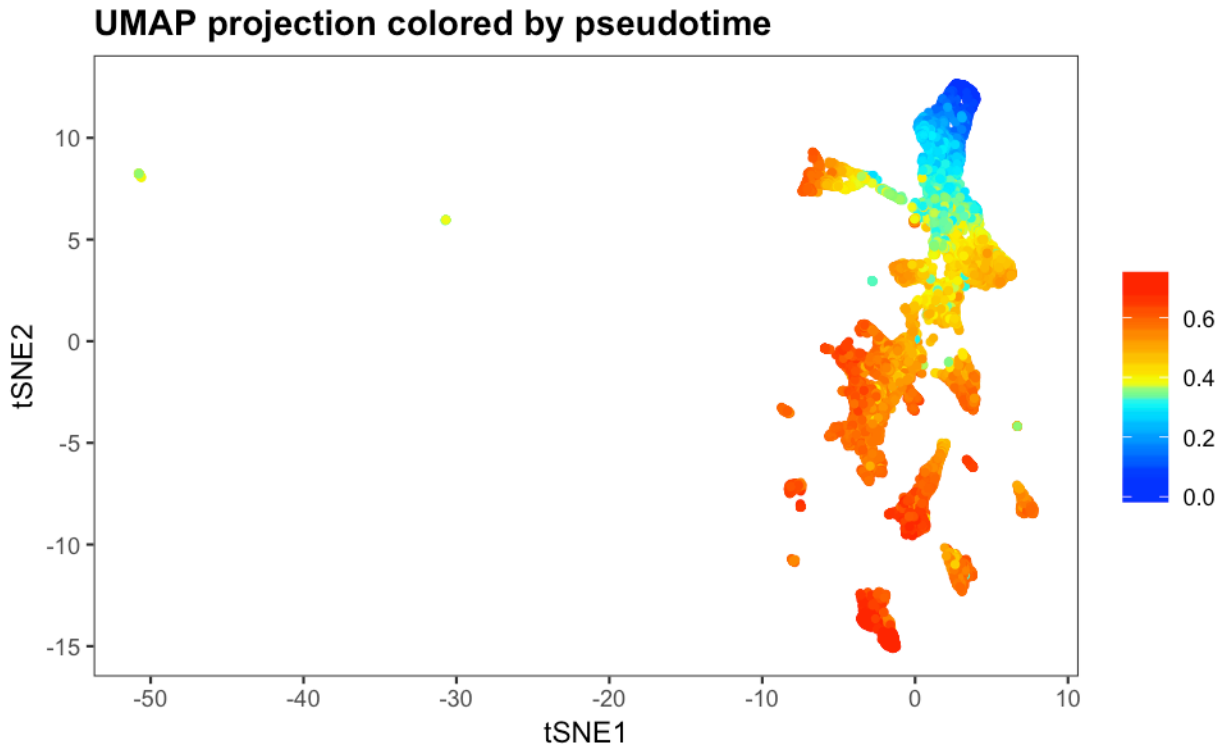



```
plotDimArray(object = object, reduction.use = "dm", dims.to.plot = 1:18,
    label = "pseudotime", plot.title = "", outer.title = "Diffusion Map labeled by pseudotime",
    legend = F, alpha = 0.4)
```

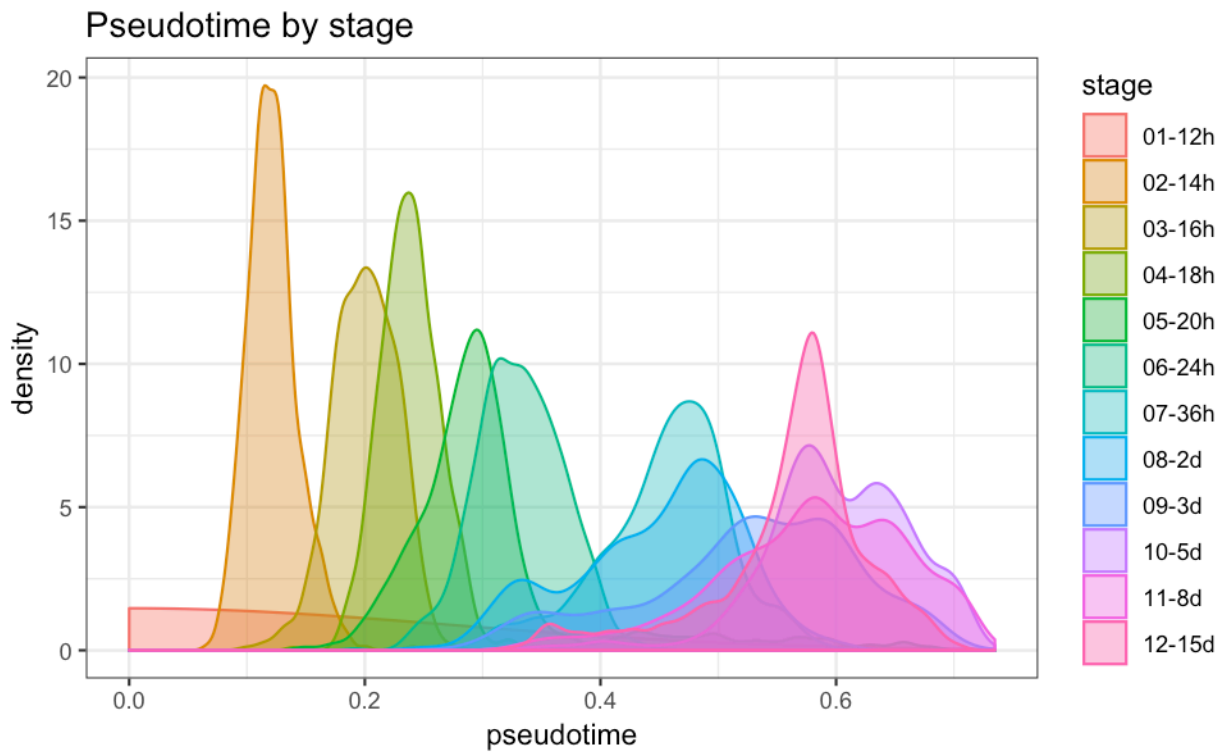



```
plotDim(object, "pseudotime", plot.title = "UMAP projection colored by pseudotime")
```

```
## Warning: Removed 500 rows containing missing values (geom_point).
```



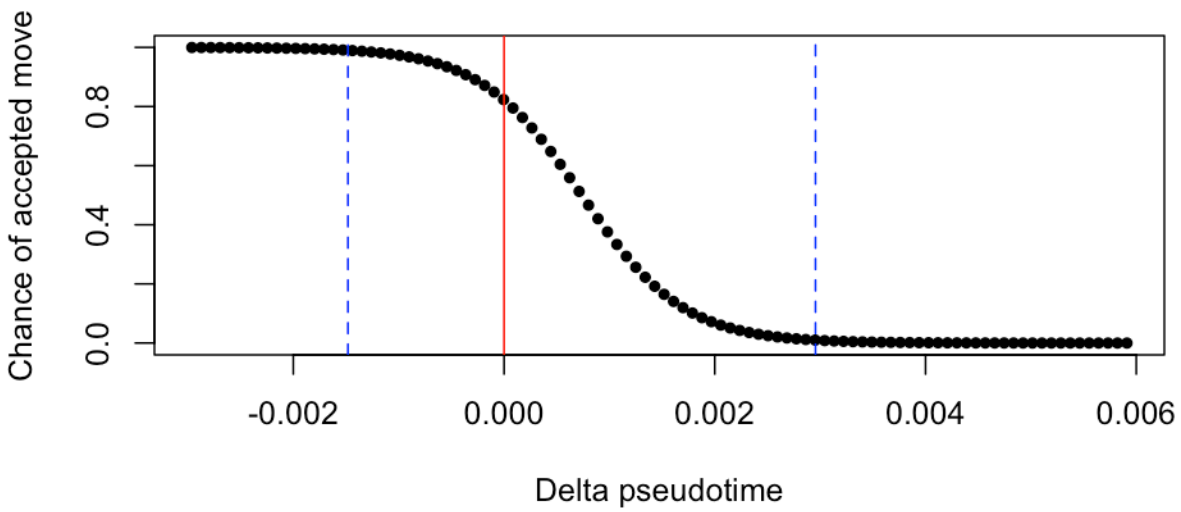

```
plotDists(object, "pseudotime", "stage", plot.title = "Pseudotime by stage")
```



Calculate biased transition matrix

In order to perform biased random walks, we must first bias the transition matrix to ensure that walks proceed towards the root and do not turn into other differentiated cell types. This is performed in the cluster script URD-TM.R.

```
# Calculate parameters for biasing the transition matrix.
diffusion.logistic <- pseudotimeDetermineLogistic(object, "pseudotime",
    optimal.cells.forward = 40, max.cells.back = 80, pseudotime.direction = "<",
    do.plot = T, print.values = T)
```




```
## [1] "Mean pseudotime back (~80 cells) 0.00295569803650678"
## [1] "Chance of accepted move to equal pseudotime is 0.822024945232085"
## [1] "Mean pseudotime forward (~40 cells) -0.00148141113789574"
```

```
# Calculate the biased matrix.
biased.tm <- pseudotimeWeightTransitionMatrix(object, pseudotime = "pseudotime",
    logistic.params = diffusion.logistic, pseudotime.direction = "<")
```

Perform biased random walks

Then, we perform biased walks starting from each tip. Visited cells are inferred to lie along the trajectory that connects the root to each cell type. This is performed in the cluster script URD-Walk.R.

Determine tips

We used clusters from 15 dpf as the tips for performing biased random walks. Here we define the cells belonging to each of those clusters.

```
# All clusters at 15 days
clusters.15day <- unique(object@group.ids[grep("15d", object@group.ids$stage),
    "res.5"])
# All cells at 15 days
cells.15day <- rownames(object@group.ids)[grep("15d", object@group.ids$stage)]
# Cell lists of each cluster at 15dpf
cells.15dpf.clusters <- lapply(clusters.15day, function(clust) intersect(cells.15day,
    cellsInCluster(object, "res.5", clust)))
names(cells.15dpf.clusters) <- paste0("15d-", clusters.15day)
```

We also load a .csv file that contains information about the tips. It has four columns:

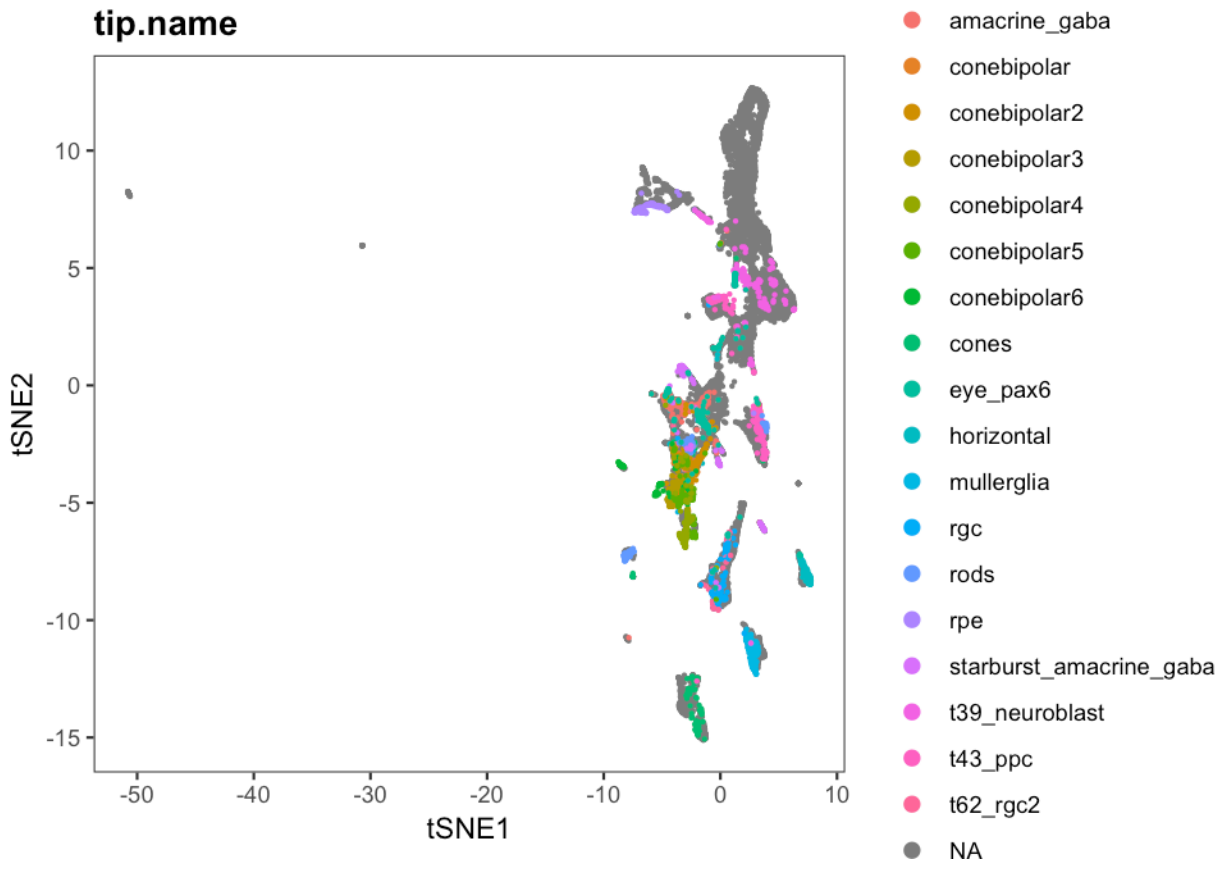
- id: Cluster ID for the tip
- use: Whether this cluster should be used when building the tree
- name: The name for this tip, which will be used on 2D plots
- short.name: The 'short' name for this tip, which would be used on 3D plots (though we did not use that feature in this study).

```
# Load CSV
tip.names <- read.csv(paste0(base.path, "tips/tip_names_retinanewnokND.csv"),
    header = F, stringsAsFactors = F, colClasses = c("character", "logical",
        "character", "character"))

# Name columns and rows
names(tip.names) <- c("id", "use", "name", "short.name")
rownames(tip.names) <- gsub("_", "-", tip.names$id)

# Sort alphabetically
tip.names <- tip.names[order(rownames(tip.names)), ]
```

These are the tips that were considered during the construction of the retina URD tree (some were excluded during tree construction later in the `buildTree` command).

```
# Define a 'tips' clustering
object@group.ids$tip <- NA
object@group.ids$tip.id <- NA
object@group.ids$tip.name <- NA

# If the tip will be used in the tree, define its cells in the
# clustering
for (i in 1:nrow(tip.names)) {
    tip.cells <- cells.15dpf.clusters[[rownames(tip.names)[i]]]
    object@group.ids[tip.cells, "tip"] <- as.character(i)
    object@group.ids[tip.cells, "tip.id"] <- rownames(tip.names)[i]
    object@group.ids[tip.cells, "tip.name"] <- as.character(tip.names[i,
        "name"])
}

# Plot the tips
plotDim(object, "tip.name")
```

```
## Warning: Removed 500 rows containing missing values (geom_point).
```



Perform the biased random walks

Biased random walks then need to be run starting from each tip. This can be performed on a laptop, but is an ideal candidate for parallelization on a cluster. (The walks from each tip can be run as a separate job.)

```
## IF RUNNING LOCALLY

# Loop through each cluster
walks <- lapply(rownames(tip.names), function(c) {
    # Exclude any tip cells that for whatever reason didn't end up in the
    # biased TM (e.g. maybe not assigned a pseudotime).
    tip.cells <- intersect(cells.15dpf.clusters[[c]], rownames(biased.tm))
    # Perform the random walk simulation
    this.walk <- simulateRandomWalk(start.cells = tip.cells, transition.matrix = biased.tm,
        end.cells = root.cells, n = 50000, end.visits = 1, verbose.freq = 1000,
        max.steps = 5000)
    return(this.walk)
})
names(walks) <- rownames(tip.names)

# Alternatively, this loop is automated by the function
# simulateRandomWalksFromTips
```

Alternatively, a set of pre-calculated walks can be loaded. Since the walks are a simulation (and

therefore not deterministic), this is particularly crucial for reproducing results.

```
## IF LOADING PRE-CALCULATED WALKS

# Get list of files in the walks directory
walks.files <- list.files(paste0(base.path, "/walks/retinanewnokND/"),
    pattern = ".rds")

# Load the walks previously performed for each cluster
walks <- lapply(rownames(tip.names), function(c) {
    walk.file <- grep(pattern = paste0("_tip-", c, "_"), x = walks.files,
        value = T)[1]
    return(readRDS(paste0(base.path, "/walks/retinanewnokND/", walk.file)))
})
names(walks) <- rownames(tip.names)
```

Process the random walks

The walks are then converted to visitation frequency by importing them into the URD object.

```
for (i in 1:nrow(tip.names)) {
    # Load the individual walk visitation frequencies into the object
    object <- processRandomWalks(object, walks = walks[[i]], walks.name = i,
        n.subsample = 1, verbose = F)
}
```

Build the URD tree

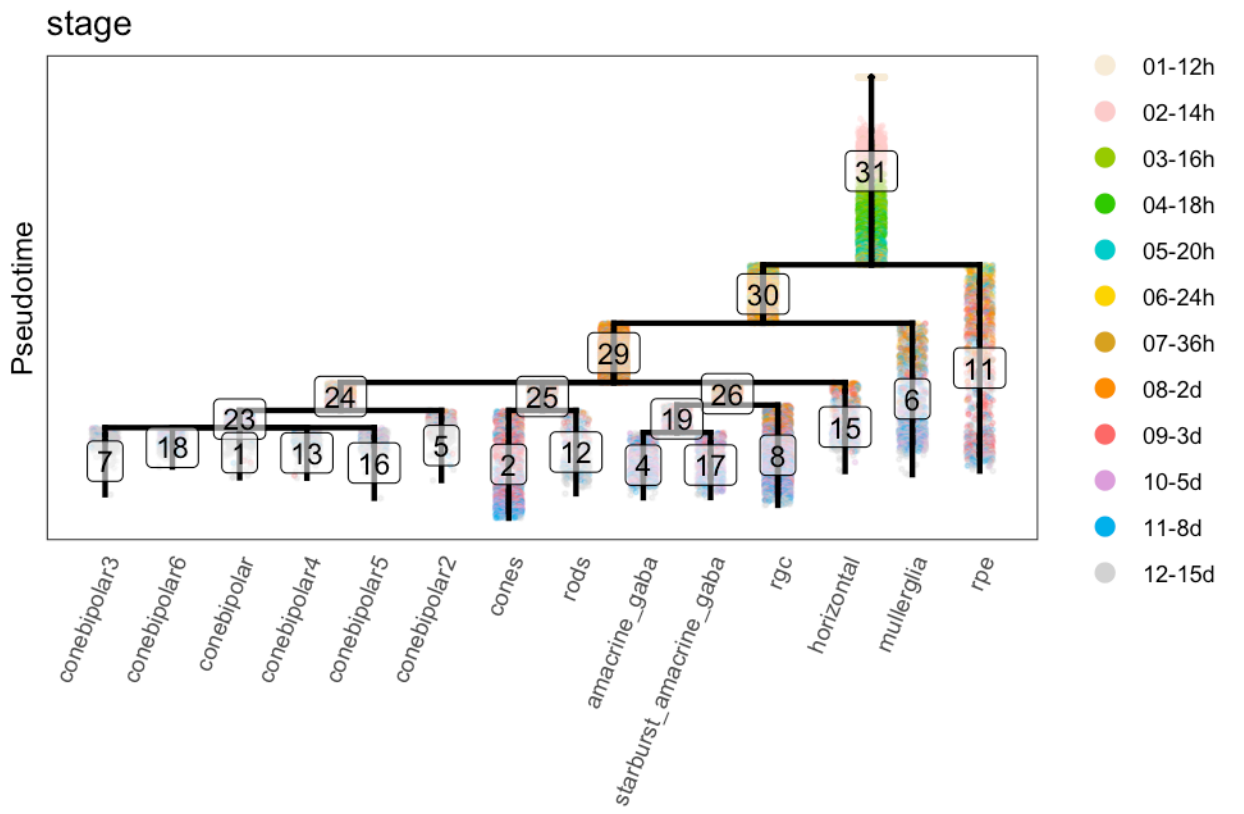
Then, a branching tree is constructed, by joining trajectories in an agglomerative fashion when cells are highly visited by walks from multiple tips. The following steps were performed in the cluster script URD-Tree.R.

```
# Tree building is destructive, so create a copy of the object
object.tree <- object

# Load tip cells
object.tree <- loadTipCells(object.tree, "tip")

# Determine tips to use
tips.to.use <- which(tip.names$use)
```

```
# Build the tree
object.tree <- buildTree(object.tree, pseudotime = "pseudotime", divergence.method = "preference",
    cells.per.pseudotime.bin = 40, bins.per.pseudotime.window = 5, save.all.breakpoint.info = T,
    p.thresh = 0.01, verbose = F, tips.use = as.character(tips.to.use))
```

```
# Name the tips of the tree
object.tree <- nameSegments(object.tree, segments = tips.to.use, segment.names = as.character(tip.names[tips.to.u
    "name"], short.names = as.character(tip.names[tips.to.use, "short.name"])))
```

```
plotTree(object.tree, "stage", discrete.colors = stage.colors, label.segments = T)
```



Save the URD tree

The tree is then saved for use in downstream analysis, and can easily be loaded for further perusal.

```
saveRDS(object.tree, file = paste0(base.path, "tree/URD-Tree-Retina.rds"))
```

Retina: 3 - URD Cascades and Figures

Jeff Farrell

10/08/2019, updated 07/30/2020

Contents

Load data

```
# Load URD
library(URD)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: Matrix
```

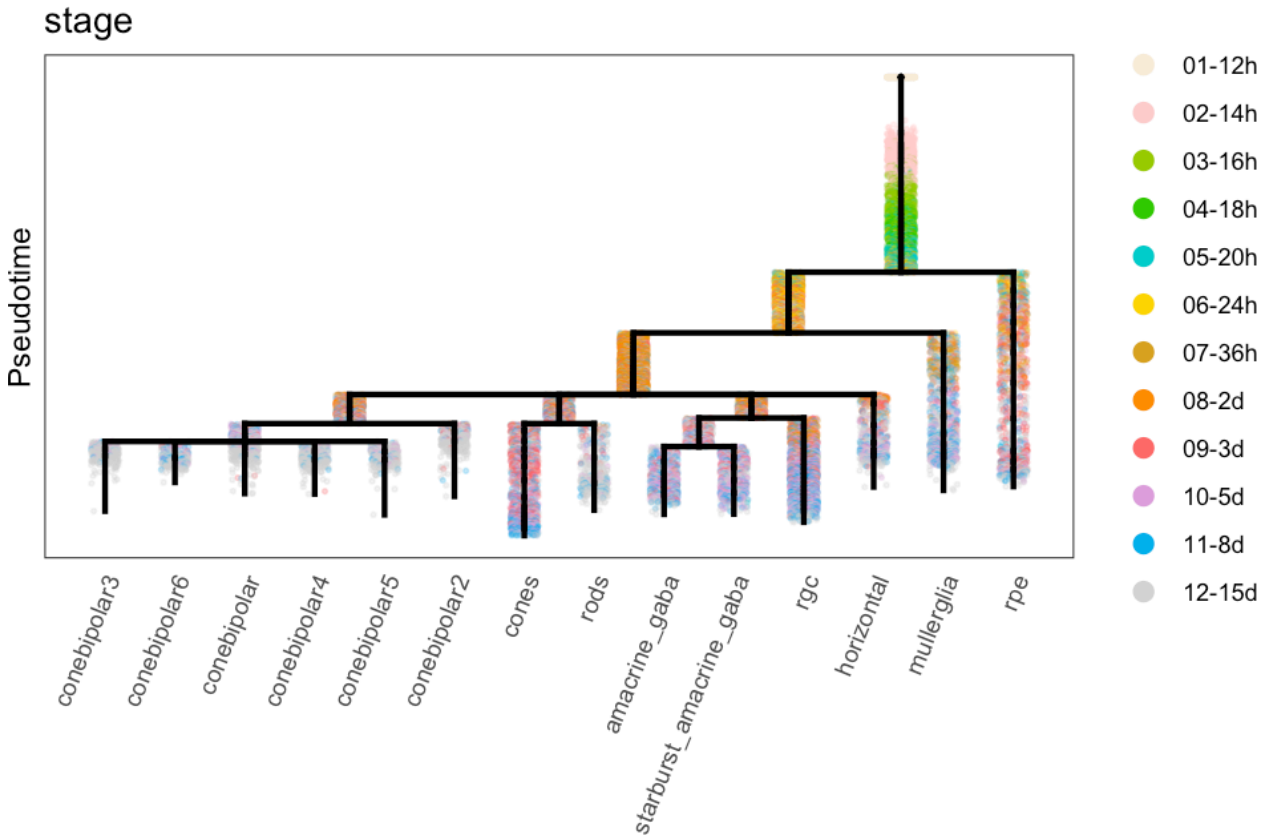
```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo
```

```
# Basic location
base.path <- "~/Documents/R sessions/urd-cluster-bushra/"

# Load completed retina tree object
obj.path <- paste0(base.path, "tree/retinanewnokND/tree-retinanewnokND_knn-140_sigma-14_40F-80B_NO-15d-29-15d-39-
obj <- readRDS(obj.path)
```

Plot gene expression on the tree

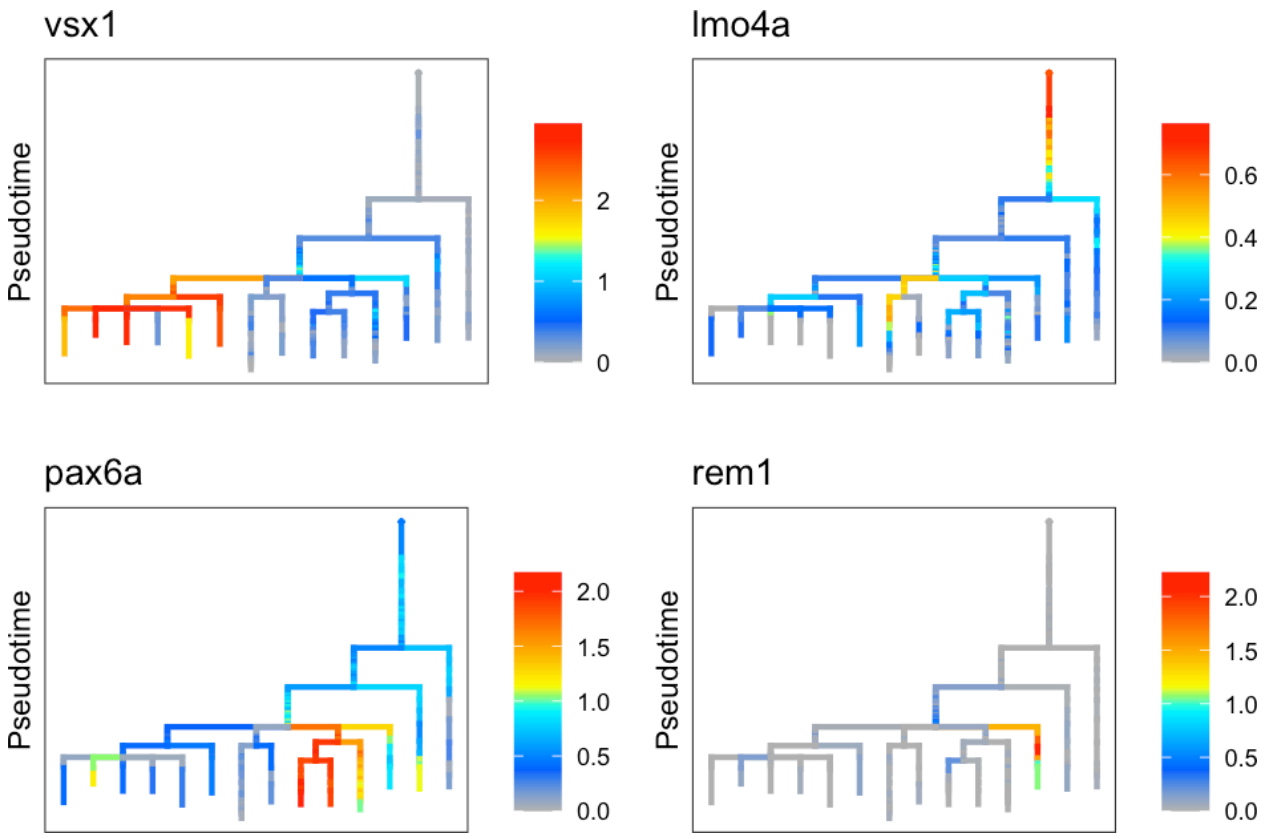
Plot tree by stage

```
stage.colors <- c("antiquewhite", "#FFCCCC", "#99CC00", "#33CC00", "cyan3", "gold",
    "goldenrod", "darkorange", "indianred1", "plum", "deepskyblue2", "lightgrey")

plotTree(obj, "stage", label.type = "group", discrete.colors = stage.colors)
```



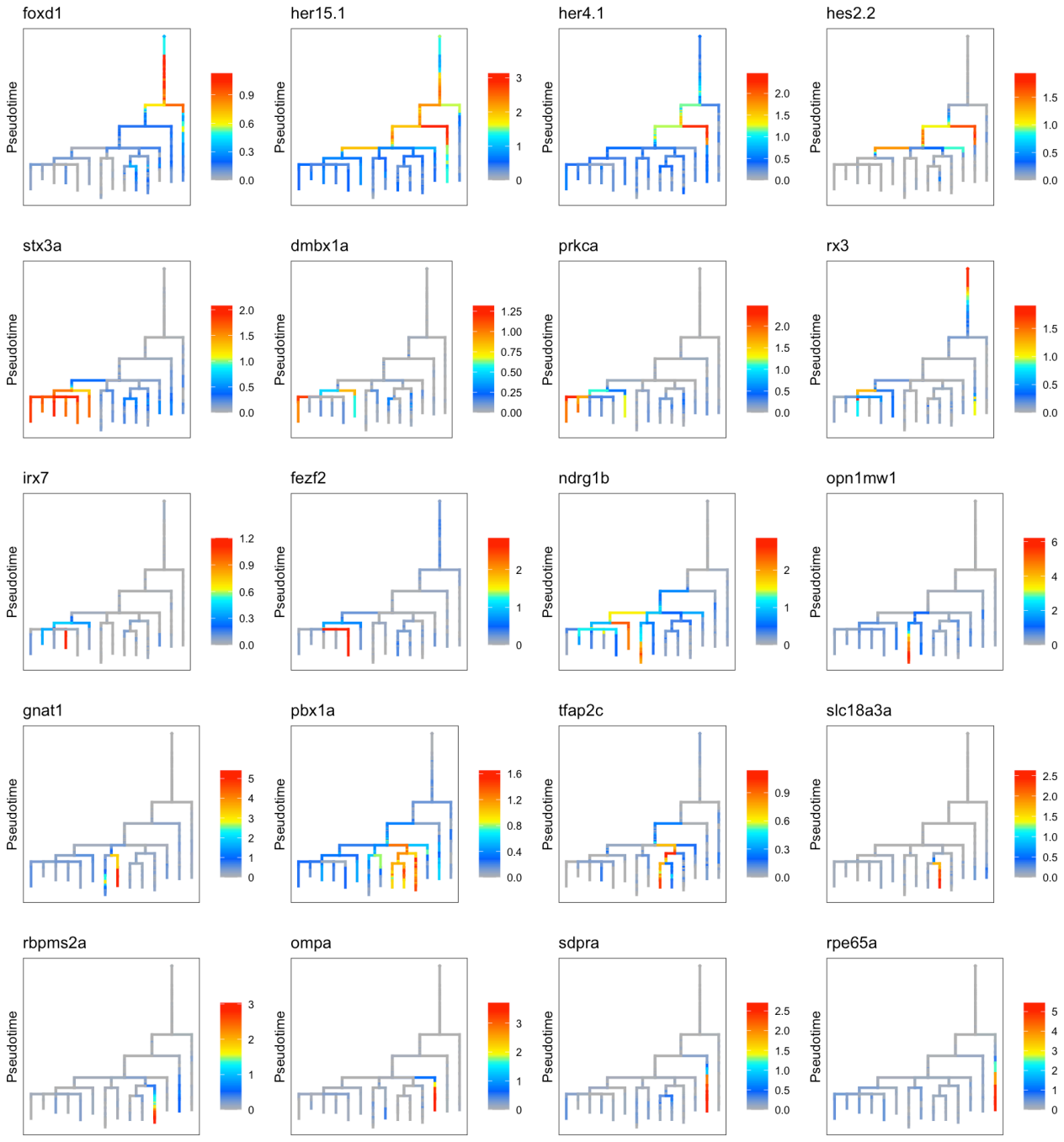
Plot tree with gene expression: main figures

```
gridExtra::grid.arrange(grobs = lapply(c("vsx1", "lmo4a", "pax6a", "rem1"), plotTree,
    object = obj, label.x = F, plot.cells = F), ncol = 2)
```



Plot tree with gene expression: supplemental figures

```
gridExtra::grid.arrange(grobs = lapply(c("foxd1", "her15.1", "her4.1", "hes2.2",
    "stx3a", "dmbx1a", "prkca", "rx3", "irx7", "fezf2", "ndrg1b", "opn1mw1", "gnat1",
    "pbx1a", "tfap2c", "slc18a3a", "rbpms2a", "ompa", "sdpra", "rpe65a"), plotTree,
    object = obj, label.x = F, plot.cells = F), ncol = 4)
```



Determine genes enriched in trajectories to particular cell types

Comparison between major cell types

We took each major group ("clade") of branches from the end of the tree as a single entity (i.e. cone bipolar cells, photoreceptors, amacrine cells, retinal ganglion cells, horizontal cells) and compared them against each other pairwise to look for differentially expressed genes.

```
# Get the parent segment of each clade to consider as a group
combined.tips <- c("24", "25", "19", "8", "15")
```

```r
# Get the cells in that segment and all child segments
cells.combined.tips <- lapply(combined.tips, function(t) whichCells(obj, label = "segment",
    value = segChildrenAll(obj, t, include.self = T)))
names(cells.combined.tips) <- combined.tips

# Loop through each of these clades and look for differentially expressed genes
combined.markers <- lapply(combined.tips, function(tip) {
    # Find all of the other clades
    opposing.tips <- setdiff(combined.tips, tip)
    # Perform pairwise comparisons to each other clade
    m.o <- lapply(opposing.tips, function(tip.opposing) {
        # message(paste0(Sys.time(), ': Comparing tip ', tip, ' to ', tip.opposing, '.'))
        # Find differentially expressed genes between the pair of clades
        ma <- markersAUCPR(object = obj, cells.1 = cells.combined.tips[[tip]], cells.2 = cells.combined.tips[[tip
            effect.size = 0.4, auc.factor = 1.1)
        # In order to facilitate combining all of the results later, add columns about
        # which two clades were compared and also a duplicate entry of the name of each
        # gene that's recovered.
        ma$gene <- rownames(ma)
        ma$tip1 <- tip
        ma$tip2 <- tip.opposing
        return(ma)
    })
    names(m.o) <- opposing.tips
    return(m.o)
})
names(combined.markers) <- combined.tips

# Require that genes are markers against at least 3 other clades
combined.markers.beatmult <- lapply(combined.markers, function(m) {
    names(which(table(unlist(lapply(m, rownames))) >= 3))
})

# Since genes might be a marker in a comparison to several other clades, combine
# the results into a single table, where each gene is listed only once with the
# info from the pairwise comparison where it had the strongest differential
# expression.
combined.markers.best <- lapply(1:length(combined.markers.beatmult), function(i) {
    cm <- do.call("rbind", combined.markers[[i]])
    cm <- cm[cm$gene %in% combined.markers.beatmult[[i]], ]
    cmb <- do.call("rbind", lapply(combined.markers.beatmult[[i]], function(g) {
        cmr <- cm[cm$gene == g, ]
        return(cmr[which.max(cmr$AUCPR.ratio), ])
    }))
    rownames(cmb) <- cmb$gene
    cmb <- cmb[order(cmb$AUCPR.ratio, decreasing = T), ]
    cmb$exp.global <- apply(obj@logupx.data[rownames(cmb), unlist(obj@tree$cells.in.segment)],
        1, mean.of.logs)
    cmb$exp.global.fc <- cmb$nTrans_1 - cmb$exp.global
    return(cmb)
})
names(combined.markers.best) <- combined.tips
```

AUCPR along tree

We also used the AUCPRTestAlongTree function to ask for genes that are differential markers of a lineage using URD's tree structure. This makes a comparison at each branchpoint from a particular cell type up to the root.

```
# Get all of the tips from the tree
tips.in.tree <- as.character(obj@tree$tips)

# Tree segments to use as root for each particular cell population.
roots <- rep("29", length(tips.in.tree))
names(roots) <- tips.in.tree
roots["11"] <- "31"
roots["6"] <- "30"
roots[c("4", "17", "8")] <- "26"

# Perform a loop of tests with each tip.
markers <- lapply(tips.in.tree, function(t) {
    this.root <- roots[t]
    # message(paste0(Sys.time(), ': Starting tip ', t, ' and root ', this.root))
    these.markers <- aucprTestAlongTree(obj, pseudotime = "pseudotime", tips = as.character(t),
        genes.use = NULL, must.beat.sibs = 0.6, report.debug = F, root = this.root,
        auc.factor = 1.1, log.effect.size = 0.4)
    these.markers$gene <- rownames(these.markers)
    these.markers$tip <- t
    return(these.markers)
})
names(markers) <- tips.in.tree
```

Functions for curating differential expression results

We further curated those differentially expressed genes using the following functions:

threshold.tree.markers

Function to threshold markers from a markersAUCPRAlongTree test with additional criteria

- markers: list of results from markersAUCPRAlongTree tests
- tip: which tip (or element of the list to pursue)
- global.fc: fold.change that gene must have along the trajectory pursued vs. rest of the data
- aucpr.ratio.all: classifier score that gene must exhibit along trajectory test vs. rest of the data
- branch.fc: fold.change that gene must have (in best case) vs. the opposing branch at any branchpoint along the trajectory.
- Returns `markers` with only a subset of rows retained.

```
threshold.tree.markers <- function(markers, tip, global.fc = 0.1, branch.fc = 0.4,
    aucpr.ratio.all = 1.03) {
    m <- markers[[tip]]
    # First off -- lose global FC < x
    bye.globalfc <- rownames(m)[m$expfc.all < global.fc]
    # Second -- get rid of branch FC < x
    bye.branchfc <- rownames(m)[m$expfc.maxBranch < branch.fc]
    # Third -- get rid of stuff essentially worse than random classification on
    # global level
```

```
    bye.badglobalaucpr <- rownames(m)[m$AUCPR.ratio.all < aucpr.ratio.all]
    bye.all <- unique(c(bye.globalfc, bye.branchfc, bye.badglobalaucpr))
    m.return <- m[setdiff(rownames(m), bye.all), ]
    return(m.return)
}
```

threshold.clade.markers

Function to threshold markers of particular clades (see "Combined major branch families") using additional criteria

- markers: result of `markersAUCPR`
- global.fc: fold.change that gene must have along the trajectory pursued vs. rest of the data (during testing, branches were compared pairwise. This compares one branch to all others together.)
- Returns `markers` with a subset of rows retained

```
threshold.clade.markers <- function(markers, global.fc = 0.1) {
    m <- markers
    # First off -- lose global FC < x
    bye.globalfc <- rownames(m)[m$exp.global.fc < global.fc]
    m.return <- m[setdiff(rownames(m), bye.globalfc), ]
    return(m.return)
}
```

divide.branches

Function to compare genes between two branches. Use this on a compiled list of markers to do a final selection of genes that are specific to one branch or another or markers of both (i.e. when making photoreceptor heatmap, use to divide into photoreceptor, cone, and rod markers)

- object: An URD object
- genes: (Character vector) Genes to test
- clust.1: (Character) Cluster 1
- clust.2: (Character) Cluster 2
- clustering: (Character) Clustering to pull from
- exp.fc: (Numeric) Minimum expression fold-change between branches to consider different
- exp.thresh: (Numeric) Minimum fraction of cells in order to consider gene expressed in a branch
- exp.diff: (Numeric) Minimum difference in fraction of cells expressing to consider gene differential
- Returns list of gene names ("specific.1" = specific to clust.1, "specific.2" = specific to clust.2, "markers" = all genes tested)

```
divide.branches <- function(object, genes, clust.1, clust.2, clustering = "segment",
    exp.fc = 0.4, exp.thresh = 0.1, exp.diff = 0.1) {
    # Double check which markers are unique to one or the other population
    mcomp <- markersAUCPR(object, clust.1 = clust.1, clust.2 = clust.2, clustering = clustering,
        effect.size = -Inf, auc.factor = 0, genes.use = genes, frac.min.diff = 0,
        frac.must.express = 0)
    specific.b <- rownames(mcomp)[abs(mcomp$exp.fc) > exp.fc & mcomp[, 4] < exp.thresh &
        mcomp[, 5] > pmin((mcomp[, 4] + exp.diff), 1)]
    specific.a <- rownames(mcomp)[abs(mcomp$exp.fc) > exp.fc & mcomp[, 5] < exp.thresh &
        mcomp[, 4] > pmin((mcomp[, 5] + exp.diff), 1)]
```

```
    r <- list(specific.a, specific.b, mcomp)
    names(r) <- c("specific.1", "specific.2", "markers")
    return(r)
}
```

Functions for heatmap generation

These functions were used in the production of heatmaps:

Color scale

Generate color scale to use with heatmaps.

```
cols <- (scales::gradient_n_pal(RColorBrewer::brewer.pal(9, "YlOrRd")))(seq(0, 1,
    length.out = 50))
```

determine.timing

Determines order to plot genes in heatmap. "Expression" is defined as 20% higher expression than the minimum observed value. "Peak" expression is defined as 50% higher expression than minimum observed value. The two longest stretches of "peak" expression are found, and then the later one is used. The onset time of the stretch of expression that contains that peak is also determined. Genes are then ordered by the pseudotime at which they enter "peak" expression, leave "peak" expression, start "expression", and leave "expression".

- **s**: result from `geneSmoothFit`
- **genes**: genes to order; default is all genes that were fit.
- Returns `s` but with an additional list entry (`$timing`) of the order to plot genes

```
determine.timing <- function(s, genes = rownames(s$mean.expression)) {
    s$timing <- as.data.frame(do.call("rbind", lapply(genes, function(g) {
        sv <- as.numeric(s$scaled.smooth[g, ])
        pt <- as.numeric(colnames(s$scaled.smooth))
        # Figure out baseline expression & threshold for finding peaks
        min.val <- max(min(sv), 0)
        peak.val <- ((1 - min.val)/2) + min.val
        exp.val <- ((1 - min.val)/5) + min.val
        # Run-length encoding of above/below the peak-threshold
        peak.rle <- rle(sv >= peak.val)
        peak.rle <- data.frame(lengths = peak.rle$lengths, values = peak.rle$values)
        peak.rle$end <- cumsum(peak.rle$lengths)
        peak.rle$start <- head(c(0, peak.rle$end) + 1, -1)
        # Run-length encoding of above/below the expressed-threshold
        exp.rle <- rle(sv >= exp.val)
        exp.rle <- data.frame(lengths = exp.rle$lengths, values = exp.rle$values)
        exp.rle$end <- cumsum(exp.rle$lengths)
        exp.rle$start <- head(c(0, exp.rle$end) + 1, -1)
        # Take top-two longest peak RLE & select later one.  Find stretches that are
        # above peak value
        peak <- which(peak.rle$values)
        # Order by length and take 1 or 2 longest ones
        peak <- peak[order(peak.rle[peak, "lengths"], decreasing = T)][1:min(2, length(peak))]
```

```
        # Order by start and take latest one.
        peak <- peak[order(peak.rle[peak, "start"], decreasing = T)][1]
        # Identify the actual peak value within that stretch
        peak <- which.max(sv[peak.rle[peak, "start"]:peak.rle[peak, "end"]]) + peak.rle[peak,
            "start"] - 1
        # Identify the start and stop of the expressed stretch that contains the peak
        exp.start <- exp.rle[which(exp.rle$end >= peak & exp.rle$start <= peak),
            "start"]
        exp.end <- exp.rle[which(exp.rle$end >= peak & exp.rle$start <= peak), "end"]
        # Identify values of expression at start and stop
        smooth.start <- sv[exp.start]
        smooth.end <- sv[exp.end]
        # Convert to pseudotime?
        exp.start <- pt[exp.start]
        exp.end <- pt[exp.end]
        peak <- pt[peak]
        # Return a vector
        v <- c(exp.start, peak, exp.end, smooth.start, smooth.end)
        names(v) <- c("pt.start", "pt.peak", "pt.end", "exp.start", "exp.end")
        return(v)
    }))
    rownames(s$timing) <- genes

    # Decide on ordering of genes
    s$gene.order <- rownames(s$timing)[order(s$timing$pt.peak, s$timing$pt.start,
        s$timing$pt.end, s$timing$exp.end, decreasing = c(F, F, F, T), method = "radix")]

    return(s)
}
```

filter.heatmap.genes

Removes undesired (mitochondrial, ribosomal, tandem duplicated genes) from heatmaps for presentation purposes.

- genes: (Character vector) genes to check
- Returns `genes` with undesired genes removed.

```
filter.heatmap.genes <- function(genes) {
    mt.genes <- grep("^mt-", ignore.case = T, genes, value = T)
    many.genes <- grep("\\(1 of many\\)", ignore.case = T, genes, value = T)
    ribo.genes <- grep("^rpl|^rps", ignore.case = T, genes, value = T)
    cox.genes <- grep("^cox", ignore.case = T, genes, value = T)
    return(setdiff(genes, c(mt.genes, many.genes, ribo.genes, cox.genes)))
}
```

Heatmaps of gene cascades

Using the genes that were determined as differentially expressed along the way to particular cell types, we generated expression cascades and plotted them as heatmaps.

Photoreceptors

Prepare cascade

```
## PHOTORECEPTORS: Seg 25 -> Cones (Seg 2) + Rods (Seg 12)

# Get markers from the two approaches:

# Lineage markers from above the combined clades
t25 <- threshold.clade.markers(combined.markers.best[["25"]], global.fc = 0.05)
# Cone markers from aucprTestAlongTree
m2 <- threshold.tree.markers(markers, "2", global.fc = 0.6)
# Rod markers from aucprTestAlongTree
m12 <- threshold.tree.markers(markers, "12", global.fc = 0.6)
pr.markers <- unique(c(rownames(t25), rownames(m2), rownames(m12)))

## Pseudotime for rods and cones is very different; for heatmaps, would like to
## normalize these, so that spline curves that consider both of them are not out
## of sync. Need to stretch pseudotime of cells in segment 12 / rods.

# Make a duplicate of the pseudotime measurement (pseudotime.212)
obj@pseudotime$pseudotime.212 <- obj@pseudotime$pseudotime
# Grab pseudotime of branchpoint
pt.start.212 <- as.numeric(obj@tree$segment.pseudotime.limits["2", "start"])
# Figure out lengths (and ratio) of the two branches in pseudotime
pt.end.212 <- as.numeric(obj@tree$segment.pseudotime.limits[c("2", "12"), "end"]) -
    pt.start.212
pt.ratio.212 <- pt.end.212[1]/pt.end.212[2]
# For cells in the shorter branch (12), subtract the starting pseudotime,
# multiply by the ratio of branch lengths, then add the starting pseudotime back
# in order to stretch the branch.
obj@pseudotime[cellsInCluster(obj, "segment", "12"), "pseudotime.212"] <- (obj@pseudotime[cellsInCluster(obj,
    "segment", "12"), "pseudotime.212"] - pt.start.212) * pt.ratio.212 + pt.start.212

# Calculate spline curves Using segments 29, 25, 2, and 12. Calculating a curve
# using only 29/25/2 for cone-specific genes, 29/25/12 for rod-specific genes,
# and 29/25/2+12 for genes that mark both. Should work now that pseudotimes are
# aligned.
spline.2 <- geneSmoothFit(obj, pseudotime = "pseudotime.212", cells = cellsInCluster(obj,
    "segment", c("29", "25", "2")), genes = pr.markers, method = "spline", moving.window = 5,
    cells.per.window = 25, pseudotime.per.window = 0.005, spar = 0.5, verbose = F)
spline.12 <- geneSmoothFit(obj, pseudotime = "pseudotime.212", cells = cellsInCluster(obj,
    "segment", c("29", "25", "12")), genes = pr.markers, method = "spline", moving.window = 5,
    cells.per.window = 25, pseudotime.per.window = 0.005, spar = 0.5, verbose = F)
spline.212 <- geneSmoothFit(obj, pseudotime = "pseudotime.212", cells = cellsInCluster(obj,
    "segment", c("29", "25", "2", "12")), genes = pr.markers, method = "spline",
    moving.window = 5, cells.per.window = 25, pseudotime.per.window = 0.005, spar = 0.5,
    verbose = F)

# Want to plot a heatmap that shows expression in photoreceptor progenitors and
# then each branch (i.e. rods, cones) as separate columns. Going to crop each
# spline fit to the correct pseudotime range and then combine them into a single
# one that can be plotted as a three-column heatmap.
```

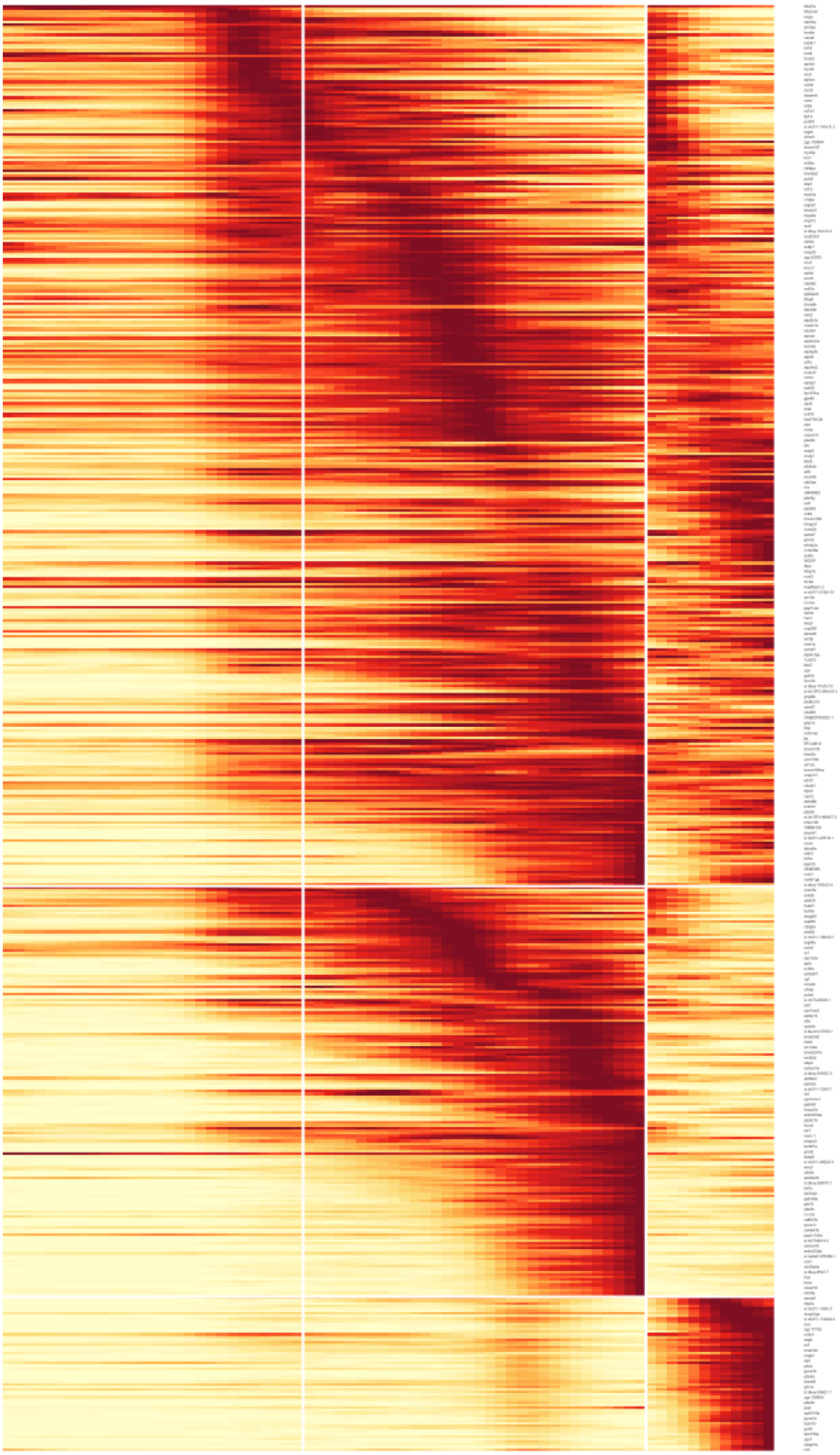
```
pt.2v12 <- obj@tree$segment.pseudotime.limits["2", "start"]  # pseudotime where the crop should happen
splines.pr <- list(cropSmoothFit(spline.212, pt.min = -Inf, pt.max = pt.2v12), cropSmoothFit(spline.2,
    pt.min = pt.2v12, pt.max = Inf), cropSmoothFit(spline.12, pt.min = pt.2v12, pt.max = Inf))
names(splines.pr) <- c("Photoreceptor Progenitors", "Rods", "Cones")
splines.pr.hm <- combineSmoothFit(splines.pr)  # Combine into a single one

# Calculate gene expression timing for ordering rows
spline.212 <- determine.timing(s = spline.212)
spline.2 <- determine.timing(s = spline.2)
spline.12 <- determine.timing(s = spline.12)

# Decide which markers are specific to one cell type or both
d2v12 <- divide.branches(obj, pr.markers, clust.1 = "2", clust.2 = "12", exp.fc = 0.4,
    exp.thresh = 0.2, exp.diff = 0.1)

# Generate gene ordering based on timing & specificity
order.212 <- filter.heatmap.genes(setdiff(spline.212$gene.order, c(d2v12$specific.1,
    d2v12$specific.2)))
order.2 <- filter.heatmap.genes(intersect(spline.2$gene.order, d2v12$specific.1))
order.12 <- filter.heatmap.genes(intersect(spline.12$gene.order, d2v12$specific.2))
gene.order <- c(order.212, order.2, order.12)

# Output gene table
table.save <- data.frame(gene = gene.order, marks = c(rep("both", length(order.212)),
    rep("cone", length(order.2)), rep("rod", length(order.12))), stringsAsFactors = F)
table.save$clade.AUCPR.ratio <- t25[table.save$gene, "AUCPR.ratio"]
table.save$clade.exp.fc <- t25[table.save$gene, "exp.fc"]
table.save$clade.exp.fc.global <- t25[table.save$gene, "exp.global.fc"]
table.save$cone.AUCPR.ratio.all <- m2[table.save$gene, "AUCPR.ratio.all"]
table.save$cone.AUCPR.ratio.maxBranch <- m2[table.save$gene, "AUCPR.ratio.maxBranch"]
table.save$cone.exp.fc.all <- m2[table.save$gene, "expfc.all"]
table.save$cone.exp.fc.best <- m2[table.save$gene, "expfc.maxBranch"]
table.save$rod.AUCPR.ratio.all <- m12[table.save$gene, "AUCPR.ratio.all"]
table.save$rod.AUCPR.ratio.maxBranch <- m12[table.save$gene, "AUCPR.ratio.maxBranch"]
table.save$rod.exp.fc.all <- m12[table.save$gene, "expfc.all"]
table.save$rod.exp.fc.best <- m12[table.save$gene, "expfc.maxBranch"]
write.csv(table.save, quote = F, file = paste0(base.path, "/heatmaps/retina-photoreceptor.csv"))
```

Generate heatmap: all genes

```
# Make sure any values <0 in the spline curves get set to 0 so that the heatmap
# scale doesn't get messed up.
splines.pr.hm$scaled.smooth[splines.pr.hm$scaled.smooth < 0] <- 0
# Determine where to place column separators (i.e. how many columns will each
# cell type occupy in the heatmap )
colsep <- cumsum(as.numeric(head(unlist(lapply(splines.pr, function(x) ncol(x$scaled.smooth))),
    -1)))
# Determine where to place row separators (i.e. how many common markers, and
# markers are specific to each cell type)
rowsep <- cumsum(c(length(order.212), length(order.2)))
# Open a PDF and generate the heatmap pdf(paste0(base.path,
# '/heatmaps/retina-photoreceptor.pdf'), width=6, height=10)
```

```
gplots::heatmap.2(x = as.matrix(splines.pr.hm$scaled.smooth[gene.order, ]), Rowv = F,
    Colv = F, dendrogram = "none", col = cols, trace = "none", density.info = "none",
    key = F, cexCol = 0.8, cexRow = 0.15, margins = c(8, 8), lwid = c(0.3, 4), lhei = c(0.3,
        4), labCol = NA, colsep = colsep, rowsep = rowsep, sepwidth = c(0.1, 0.2))
title(main = "Photoreceptors")
title(main = "Precursors", line = -41, adj = 0)
title(main = "Cones", line = -41, adj = 0.45)
title(main = "Rods", line = -41, adj = 0.76)
```

Photoreceptors

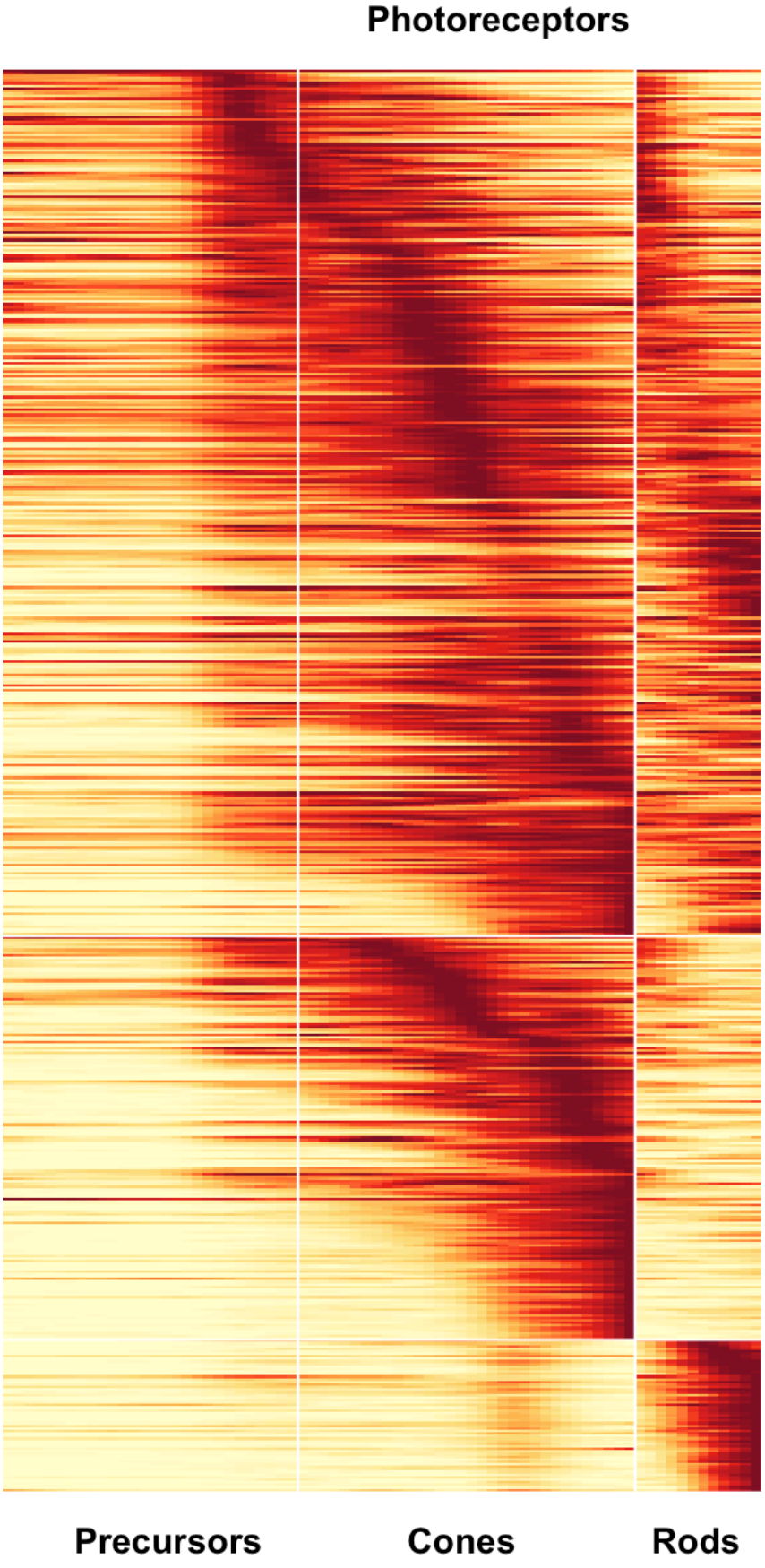


Precursors Cones Rods

```
# dev.off()
```

Generate heatmap: main figure

```
## Generate heatmap with only particular genes labeled for main figure
genes.to.plot <- c("isl2a", "prdm1a", "otx5", "crx", "six7", "nr2f1b", "nr2e3", "aplnrb",
    "aplnra", "apln")
rownames.to.plot <- gene.order
rtp <- rownames.to.plot %in% genes.to.plot
rownames.to.plot[!rtp] <- ""
rownames.to.plot[rtp] <- paste0("- ", rownames.to.plot[rtp])
# Open a PDF and generate the heatmap pdf(paste0(base.path,
# '/heatmaps/retina-photoreceptor-mainfig.pdf'), width=6, height=10)
gplots::heatmap.2(x = as.matrix(splines.pr.hm$scaled.smooth[gene.order, ]), Rowv = F,
    Colv = F, dendrogram = "none", col = cols, trace = "none", density.info = "none",
    key = F, cexCol = 0.8, cexRow = 1.8, margins = c(8, 8), lwid = c(0.3, 4), lhei = c(0.3,
        4), labCol = NA, colsep = colsep, rowsep = rowsep, sepwidth = c(0.1, 0.2),
    labRow = rownames.to.plot)
title(main = "Photoreceptors")
title(main = "Precursors", line = -41, adj = 0)
title(main = "Cones", line = -41, adj = 0.45)
title(main = "Rods", line = -41, adj = 0.76)
```


Photoreceptors
Precursors
Cones
Rods

```
# dev.off()
```

Amacrine cells

Prepare cascade

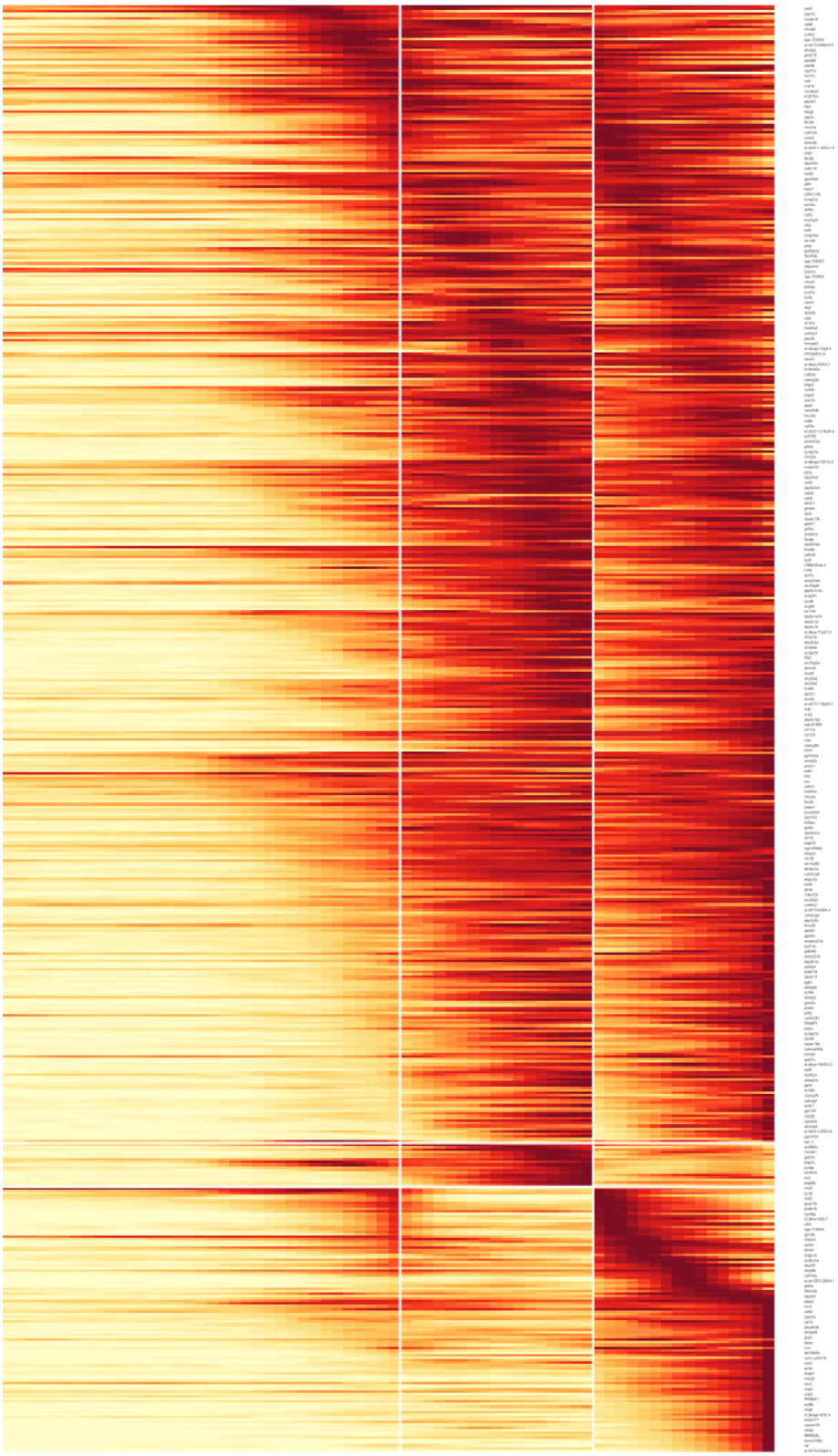
```
## AMACRINE CELLS: Seg 19 -> Amarcine (Seg 4) + Starburst Amacrine (Seg 17)

# Get markers from the two approaches:

# Lineage markers from above the combined clades
t19 <- threshold.clade.markers(combined.markers.best[["19"]], global.fc = 0.05)
# Amacrine markers from aucprTestAlongTree
m4 <- threshold.tree.markers(markers, "4", global.fc = 0.6)
# Starburst amacrine markers from aucprTestAlongTree
m17 <- threshold.tree.markers(markers, "17", global.fc = 0.6)
am.markers <- unique(c(rownames(t19), rownames(m4), rownames(m17)))

## These have pretty equivalent pseudotimes, so don't need to worry about
## stretching them to match or anything.

# Calculate spline curves Using segments 29, 26, 19, and 4/17. Calculating a
# curve using only 29/26/19/4 for amacrine-specific genes, 29/26/19/4 for
# starburst-specific genes, and 29/26/19/4+17 for genes that mark both amacrine
# populations.
spline.4 <- geneSmoothFit(obj, pseudotime = "pseudotime", cells = cellsInCluster(obj,
    "segment", c("29", "26", "19", "4")), genes = am.markers, method = "spline",
    moving.window = 5, cells.per.window = 25, pseudotime.per.window = 0.005, spar = 0.5,
    verbose = F)
spline.17 <- geneSmoothFit(obj, pseudotime = "pseudotime", cells = cellsInCluster(obj,
    "segment", c("29", "26", "19", "17")), genes = am.markers, method = "spline",
    moving.window = 5, cells.per.window = 25, pseudotime.per.window = 0.005, spar = 0.5,
    verbose = F)
spline.417 <- geneSmoothFit(obj, pseudotime = "pseudotime", cells = cellsInCluster(obj,
    "segment", c("29", "26", "19", "4", "17")), genes = am.markers, method = "spline",
    moving.window = 5, cells.per.window = 25, pseudotime.per.window = 0.005, spar = 0.5,
    verbose = F)

# Want to plot a heatmap that shows expression in amacrine progenitors and then
# each branch (i.e. amacrine_gaba, starburst_amacrine) as separate columns. Going
# to crop each spline fit to the correct pseudotime range and then combine them
# into a single one that can be plotted as a three-column heatmap.

pt.4v17 <- obj@tree$segment.pseudotime.limits["4", "start"]  # pseudotime where the crop should happen
splines.am <- list(cropSmoothFit(spline.417, pt.min = -Inf, pt.max = pt.4v17), cropSmoothFit(spline.4,
    pt.min = pt.4v17, pt.max = Inf), cropSmoothFit(spline.17, pt.min = pt.4v17, pt.max = Inf))
names(splines.am) <- c("Amacrine Precursors", "Amacrine", "Starburst Amacrine")
splines.am.hm <- combineSmoothFit(splines.am)  # Combine into a single one

# Calculate gene expression timing for ordering rows
spline.417 <- determine.timing(s = spline.417)
spline.4 <- determine.timing(s = spline.4)
spline.17 <- determine.timing(s = spline.17)
```

```
# Decide which markers are specific to one cell type or both
d4v17 <- divide.branches(obj, am.markers, clust.1 = "4", clust.2 = "17", exp.fc = 0.4,
    exp.thresh = 0.2, exp.diff = 0.1)

# Generate gene ordering based on timing & specificity
order.417 <- filter.heatmap.genes(setdiff(spline.417$gene.order, c(d4v17$specific.1,
    d4v17$specific.2)))
order.4 <- filter.heatmap.genes(intersect(spline.4$gene.order, d4v17$specific.1))
order.17 <- filter.heatmap.genes(intersect(spline.17$gene.order, d4v17$specific.2))
gene.order <- c(order.417, order.4, order.17)

# Output gene table
table.save <- data.frame(gene = gene.order, marks = c(rep("both", length(order.417)),
    rep("amacrine", length(order.4)), rep("starburst", length(order.17))), stringsAsFactors = F)
table.save$clade.AUCPR.ratio <- t19[table.save$gene, "AUCPR.ratio"]
table.save$clade.exp.fc <- t19[table.save$gene, "exp.fc"]
table.save$clade.exp.fc.global <- t19[table.save$gene, "exp.global.fc"]
table.save$am.AUCPR.ratio.all <- m4[table.save$gene, "AUCPR.ratio.all"]
table.save$am.AUCPR.ratio.maxBranch <- m4[table.save$gene, "AUCPR.ratio.maxBranch"]
table.save$am.exp.fc.all <- m4[table.save$gene, "expfc.all"]
table.save$am.exp.fc.best <- m4[table.save$gene, "expfc.maxBranch"]
table.save$star.AUCPR.ratio.all <- m17[table.save$gene, "AUCPR.ratio.all"]
table.save$star.AUCPR.ratio.maxBranch <- m17[table.save$gene, "AUCPR.ratio.maxBranch"]
table.save$star.exp.fc.all <- m17[table.save$gene, "expfc.all"]
table.save$star.exp.fc.best <- m17[table.save$gene, "expfc.maxBranch"]
write.csv(table.save, quote = F, file = paste0(base.path, "/heatmaps/retina-amacrine.csv"))
```

Generate heatmap: all genes

```
# Make sure any values <0 in the spline curves get set to 0 so that the heatmap
# scale doesn't get messed up.
splines.am.hm$scaled.smooth[splines.am.hm$scaled.smooth < 0] <- 0
# Determine where to place column separators (i.e. how many columns will each
# cell type occupy in the heatmap )
colsep <- cumsum(as.numeric(head(unlist(lapply(splines.am, function(x) ncol(x$scaled.smooth))),
    -1)))
# Determine where to place row separators (i.e. how many common markers, and
# markers are specific to each cell type)
rowsep <- cumsum(c(length(order.417), length(order.4)))
# Open a PDF and generate the heatmap pdf(paste0(base.path,
# '/heatmaps/retina-amacrine.pdf'), width=6, height=10)
gplots::heatmap.2(x = as.matrix(splines.am.hm$scaled.smooth[gene.order, ]), Rowv = F,
    Colv = F, dendrogram = "none", col = cols, trace = "none", density.info = "none",
    key = F, cexCol = 0.8, cexRow = 0.15, margins = c(8, 8), lwid = c(0.3, 4), lhei = c(0.3,
        4), labCol = NA, colsep = colsep, rowsep = rowsep, sepwidth = c(0.05, 0.2))
title(main = "Amacrine Cells")
title(main = "Precursors", line = -41, adj = 0.05)
title(main = "Amacrine", line = -41, adj = 0.475)
title(main = "Starburst", line = -41, adj = 0.75)
```

Amacrine Cells



Precursors Amacrine Starburst

```
# dev.off()
```

Retinal ganglion cells

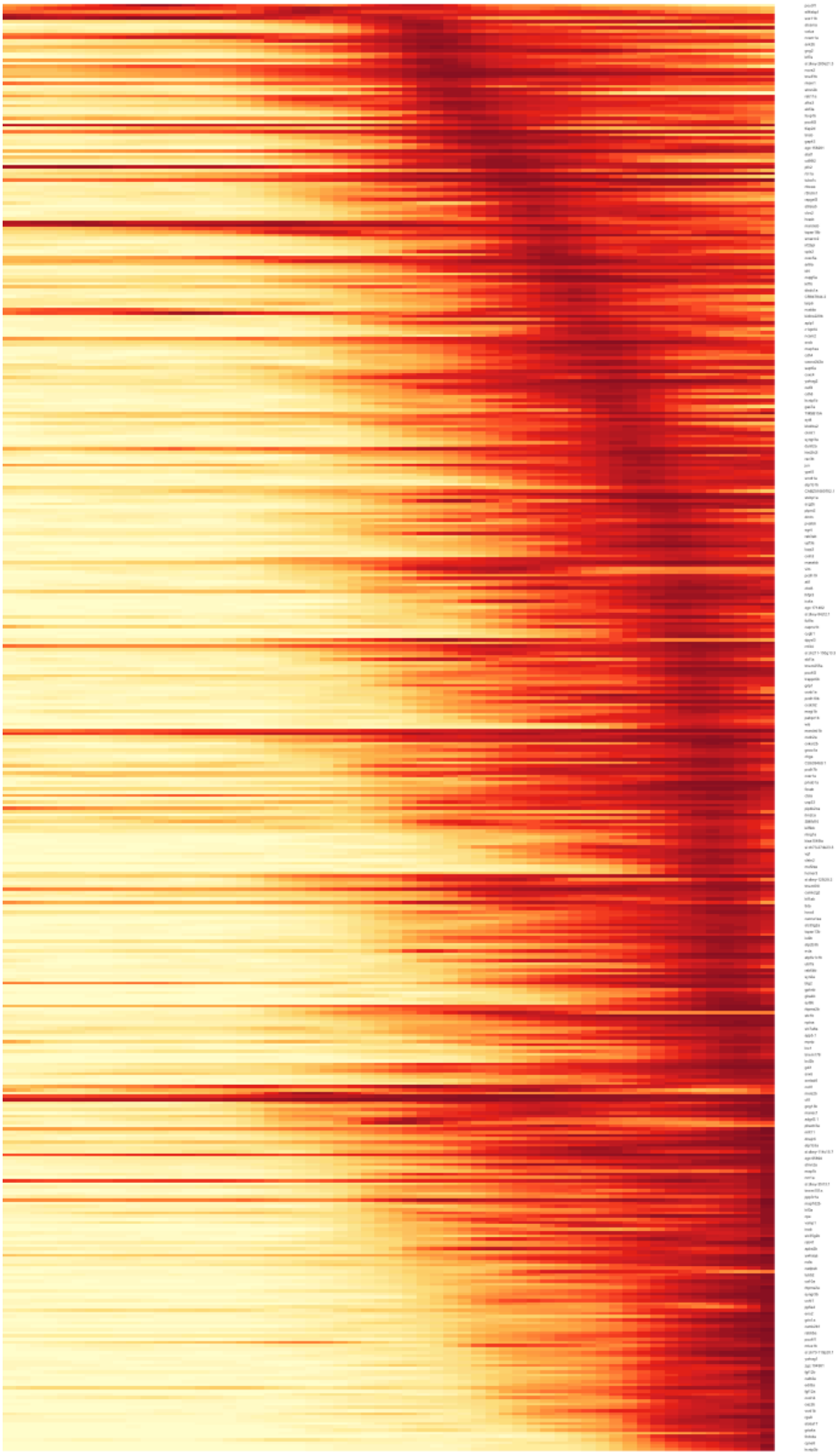
Prepare cascade

```
## RGCs: Seg 8

# Get markers from the two approaches:

# Lineage markers from above the combined clades
t8 <- threshold.clade.markers(combined.markers.best[["8"]], global.fc = 0.05)
# RGC markers from aucprTestAlongTree
m8 <- threshold.tree.markers(markers, "8", global.fc = 0.6)
rgc.markers <- unique(c(rownames(t8), rownames(m8)))

# Calculate spline curves Using segments 29, 26, and 8.
spline.8 <- geneSmoothFit(obj, pseudotime = "pseudotime", cells = cellsInCluster(obj,
    "segment", c("29", "26", "8")), genes = rgc.markers, method = "spline", moving.window = 5,
    cells.per.window = 25, pseudotime.per.window = 0.005, spar = 0.5, verbose = F)

# Calculate gene expression timing for ordering rows
spline.8 <- determine.timing(s = spline.8)
order.8 <- filter.heatmap.genes(spline.8$gene.order)

# Output gene table
table.save <- data.frame(gene = order.8, stringsAsFactors = F)
table.save$clade.AUCPR.ratio <- t8[table.save$gene, "AUCPR.ratio"]
table.save$clade.exp.fc <- t8[table.save$gene, "exp.fc"]
table.save$clade.exp.fc.global <- t8[table.save$gene, "exp.global.fc"]
table.save$rgc.AUCPR.ratio.all <- m8[table.save$gene, "AUCPR.ratio.all"]
table.save$rgc.AUCPR.ratio.maxBranch <- m8[table.save$gene, "AUCPR.ratio.maxBranch"]
table.save$rgc.exp.fc.all <- m8[table.save$gene, "expfc.all"]
table.save$rgc.exp.fc.best <- m8[table.save$gene, "expfc.maxBranch"]
write.csv(table.save, quote = F, file = paste0(base.path, "/heatmaps/retina-rgc.csv"))
```

Generate heatmap: all genes

```
# Make sure any values <0 in the spline curves get set to 0 so that the heatmap
# scale doesn't get messed up.
spline.8$scaled.smooth[spline.8$scaled.smooth < 0] <- 0
# Open a PDF and generate the heatmap pdf(paste0(base.path,
# '/heatmaps/retina-rgc.pdf'), width=6, height=10)
gplots::heatmap.2(x = as.matrix(spline.8$scaled.smooth[order.8, ]), Rowv = F, Colv = F,
    dendrogram = "none", col = cols, trace = "none", density.info = "none", key = F,
    cexCol = 0.8, cexRow = 0.15, margins = c(8, 8), lwid = c(0.3, 4), lhei = c(0.3,
        4), labCol = NA)
title(main = "Retinal Ganglion Cells")
```

Retinal Ganglion Cells

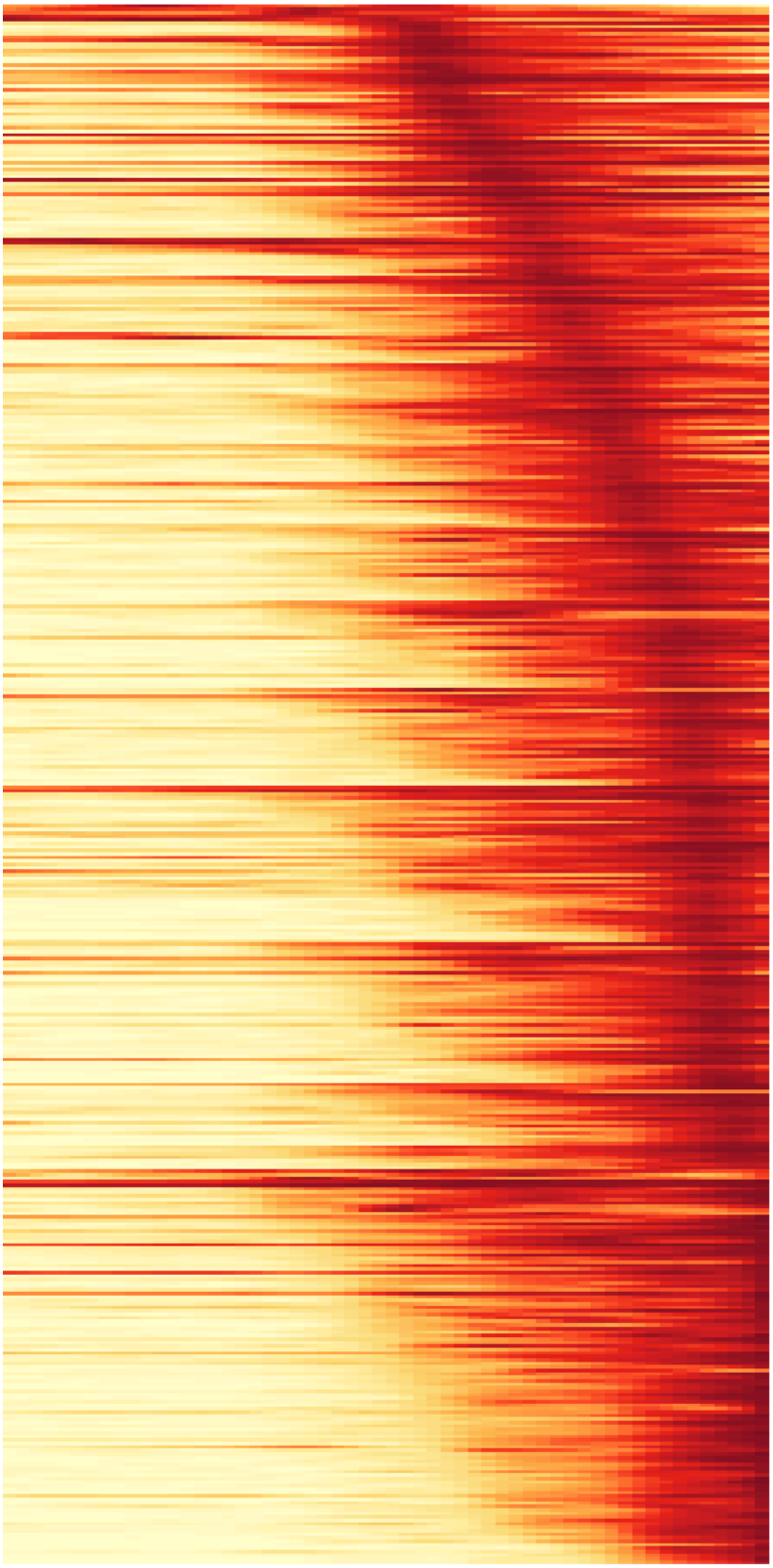


```
# dev.off()
```

Generate heatmap: main figure

```
genes.to.plot <- c("sox11a", "sox11b", "sox6", "irx4a", "pou4f2", "pou4f1", "rbpms2b",
    "rbpms2a")
rownames.to.plot <- order.8
rtp <- rownames.to.plot %in% genes.to.plot
rownames.to.plot[!rtp] <- ""
rownames.to.plot[rtp] <- paste0("- ", rownames.to.plot[rtp])
# Open a PDF and generate the heatmap pdf(paste0(base.path,
# '/heatmaps/retina-rgc-mainfig.pdf'), width=6, height=10)
gplots::heatmap.2(x = as.matrix(spline.8$scaled.smooth[order.8, ]), Rowv = F, Colv = F,
    dendrogram = "none", col = cols, trace = "none", density.info = "none", key = F,
    cexCol = 0.8, cexRow = 1.8, margins = c(8, 10), lwid = c(0.3, 4), lhei = c(0.3,
        4), labCol = NA, labRow = rownames.to.plot)
title(main = "Retinal Ganglion Cells")
```

Retinal Ganglion Cells



- sox11b

```
# dev.off()
```

Horizontal Cells

Prepare cascade

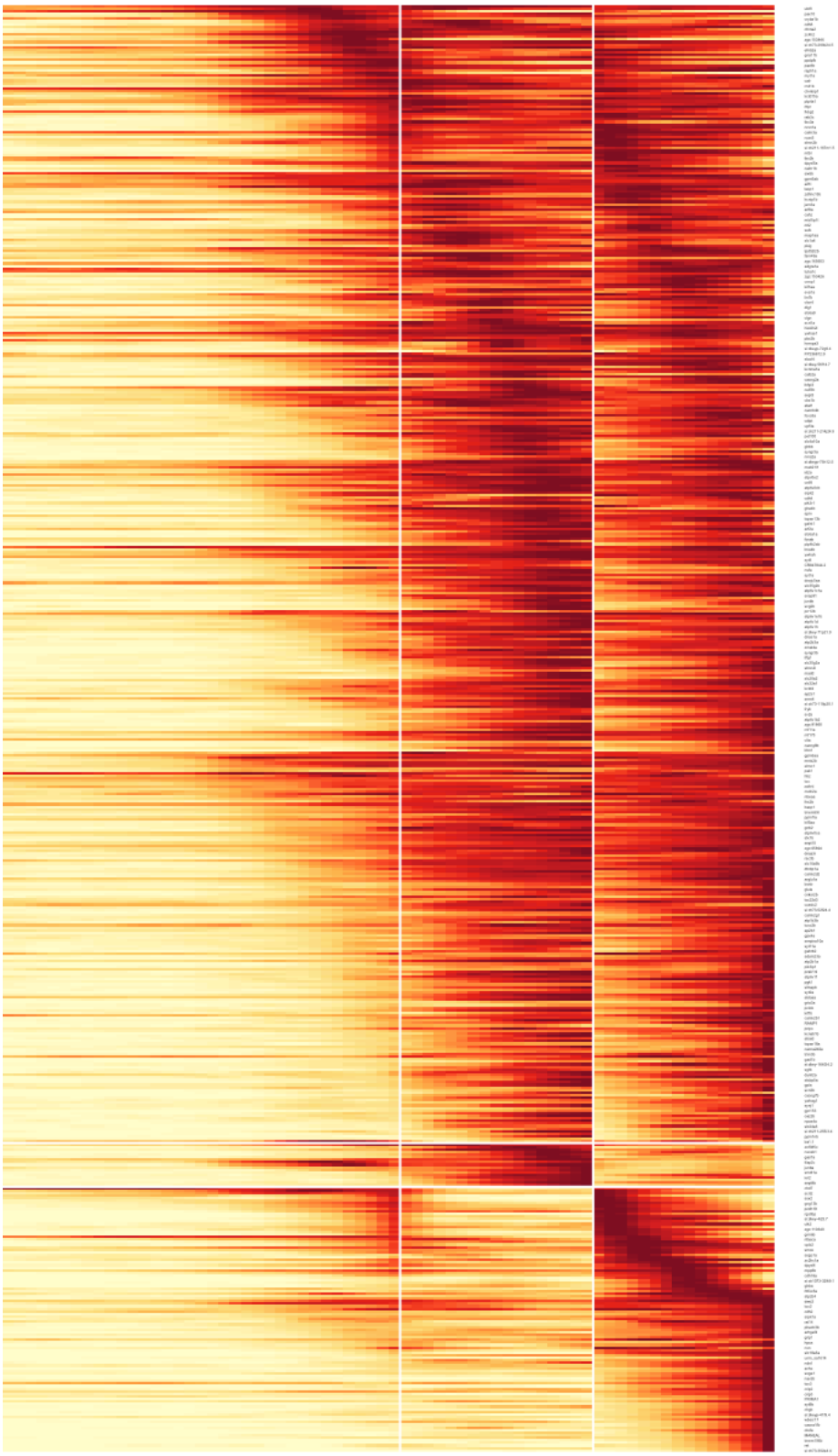
```
## Horizontal Cells: Seg 15

# Get markers from the two approaches:

# Lineage markers from above the combined clades
t15 <- threshold.clade.markers(combined.markers.best[["15"]], global.fc = 0.05)
# Horizontal Cell markers from aucprTestAlongTree
m15 <- threshold.tree.markers(markers, "15", global.fc = 0.6)
horiz.markers <- unique(c(rownames(t15), rownames(m15)))

# Calculate spline curves Using segments 29 and 15.
spline.15 <- geneSmoothFit(obj, pseudotime = "pseudotime", cells = cellsInCluster(obj,
    "segment", c("29", "15")), genes = horiz.markers, method = "spline", moving.window = 5,
    cells.per.window = 25, pseudotime.per.window = 0.005, spar = 0.5, verbose = F)

# Calculate gene expression timing for ordering rows
spline.15 <- determine.timing(s = spline.15)
order.15 <- filter.heatmap.genes(spline.15$gene.order)

# Output gene table
table.save <- data.frame(gene = order.15, stringsAsFactors = F)
table.save$clade.AUCPR.ratio <- t15[table.save$gene, "AUCPR.ratio"]
table.save$clade.exp.fc <- t15[table.save$gene, "exp.fc"]
table.save$clade.exp.fc.global <- t15[table.save$gene, "exp.global.fc"]
table.save$horiz.AUCPR.ratio.all <- m15[table.save$gene, "AUCPR.ratio.all"]
table.save$horiz.AUCPR.ratio.maxBranch <- m15[table.save$gene, "AUCPR.ratio.maxBranch"]
table.save$horiz.exp.fc.all <- m15[table.save$gene, "expfc.all"]
table.save$horiz.exp.fc.best <- m15[table.save$gene, "expfc.maxBranch"]
write.csv(table.save, quote = F, file = paste0(base.path, "/heatmaps/retina-horiz.csv"))
```

Generate heatmap: all genes

```
# Make sure any values <0 in the spline curves get set to 0 so that the heatmap
# scale doesn't get messed up.
splines.am.hm$scaled.smooth[splines.am.hm$scaled.smooth < 0] <- 0
# Determine where to place column separators (i.e. how many columns will each
# cell type occupy in the heatmap )
colsep <- cumsum(as.numeric(head(unlist(lapply(splines.am, function(x) ncol(x$scaled.smooth))),
    -1)))
# Determine where to place row separators (i.e. how many common markers, and
# markers are specific to each cell type)
rowsep <- cumsum(c(length(order.417), length(order.4)))
# Open a PDF and generate the heatmap pdf(paste0(base.path,
# '/heatmaps/retina-amacrine.pdf'), width=6, height=10)
gplots::heatmap.2(x = as.matrix(splines.am.hm$scaled.smooth[gene.order, ]), Rowv = F,
    Colv = F, dendrogram = "none", col = cols, trace = "none", density.info = "none",
```

```
    key = F, cexCol = 0.8, cexRow = 0.15, margins = c(8, 8), lwid = c(0.3, 4), lhei = c(0.3,
        4), labCol = NA, colsep = colsep, rowsep = rowsep, sepwidth = c(0.05, 0.2))
title(main = "Amacrine Cells")
title(main = "Precursors", line = -41, adj = 0.05)
title(main = "Amacrine", line = -41, adj = 0.475)
title(main = "Starburst", line = -41, adj = 0.75)
```

Amacrine Cells



Precursors Amacrine Starburst

```
# dev.off()
```

Muller Glia

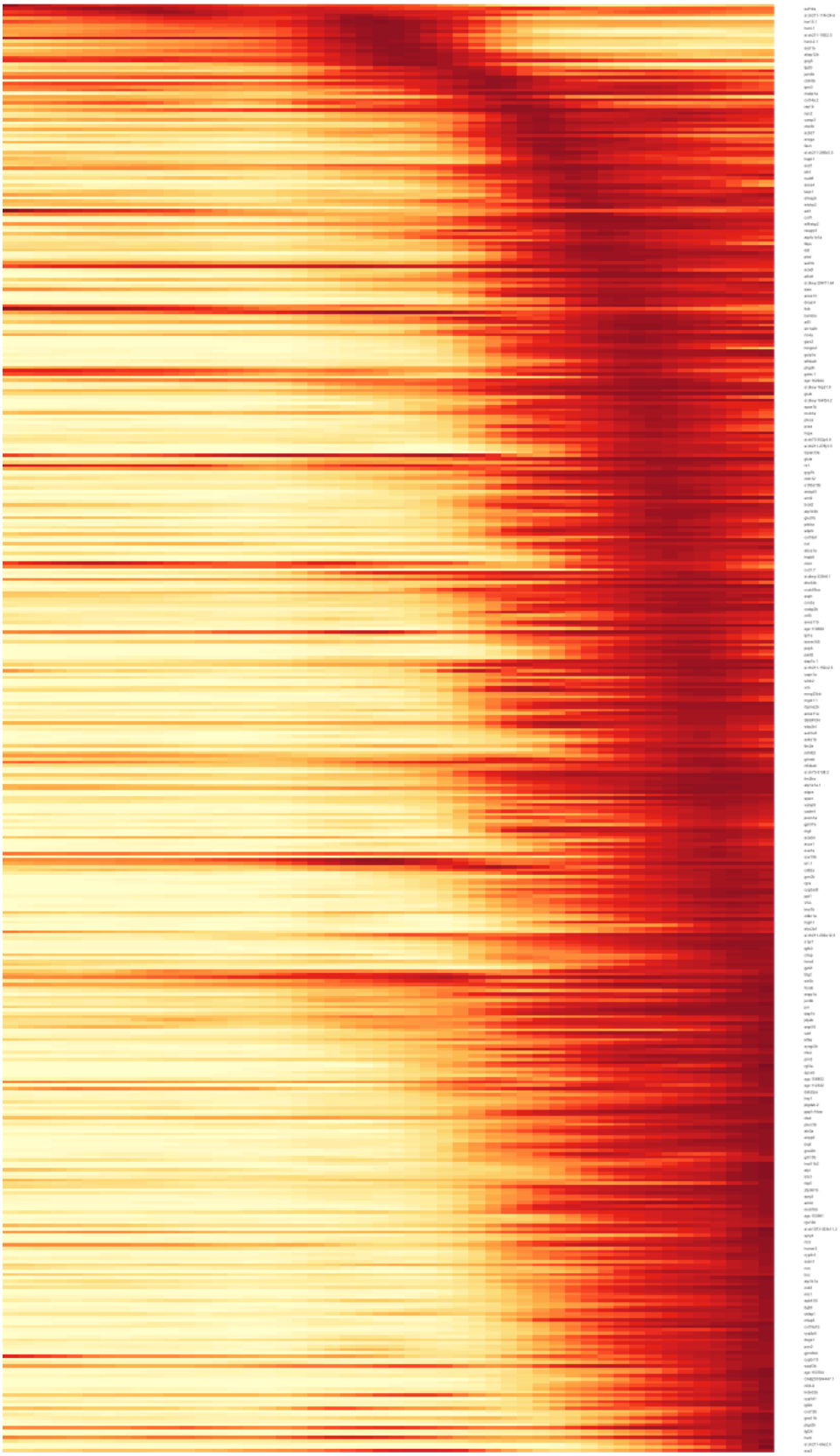
Prepare cascade

```
## Muller Glia: Seg 6

# Get markers from the two approaches
m6 <- threshold.tree.markers(markers, "6", global.fc = 0.6)  # Muller Glia markers from aucprTestAlongTree
muller.markers <- rownames(m6)

# Calculate spline curves Using segments 29 and 15.
spline.6 <- geneSmoothFit(obj, pseudotime = "pseudotime", cells = cellsInCluster(obj,
    "segment", c("30", "6")), genes = muller.markers, method = "spline", moving.window = 5,
    cells.per.window = 25, pseudotime.per.window = 0.005, spar = 0.5, verbose = F)

# Calculate gene expression timing for ordering rows
spline.6 <- determine.timing(s = spline.6)
order.6 <- filter.heatmap.genes(spline.6$gene.order)

# Output gene table
table.save <- data.frame(gene = order.6, stringsAsFactors = F)
table.save$muller.AUCPR.ratio.all <- m6[table.save$gene, "AUCPR.ratio.all"]
table.save$muller.AUCPR.ratio.maxBranch <- m6[table.save$gene, "AUCPR.ratio.maxBranch"]
table.save$muller.exp.fc.all <- m6[table.save$gene, "expfc.all"]
table.save$muller.exp.fc.best <- m6[table.save$gene, "expfc.maxBranch"]
write.csv(table.save, quote = F, file = paste0(base.path, "/heatmaps/retina-muller.csv"))
```

Generate heatmap: all genes

```
# Make sure any values <0 in the spline curves get set to 0 so that the heatmap
# scale doesn't get messed up.
spline.6$scaled.smooth[spline.6$scaled.smooth < 0] <- 0
# Open a PDF and generate the heatmap pdf(paste0(base.path,
# '/heatmaps/retina-muller.pdf'), width=6, height=10)
gplots::heatmap.2(x = as.matrix(spline.6$scaled.smooth[order.6, ]), Rowv = F, Colv = F,
    dendrogram = "none", col = cols, trace = "none", density.info = "none", key = F,
    cexCol = 0.8, cexRow = 0.15, margins = c(8, 8), lwid = c(0.3, 4), lhei = c(0.3,
        4), labCol = NA)
title(main = "Muller Glia")
```

Muller Glia



```
# dev.off()
```

Retinal Pigmented Epithelium

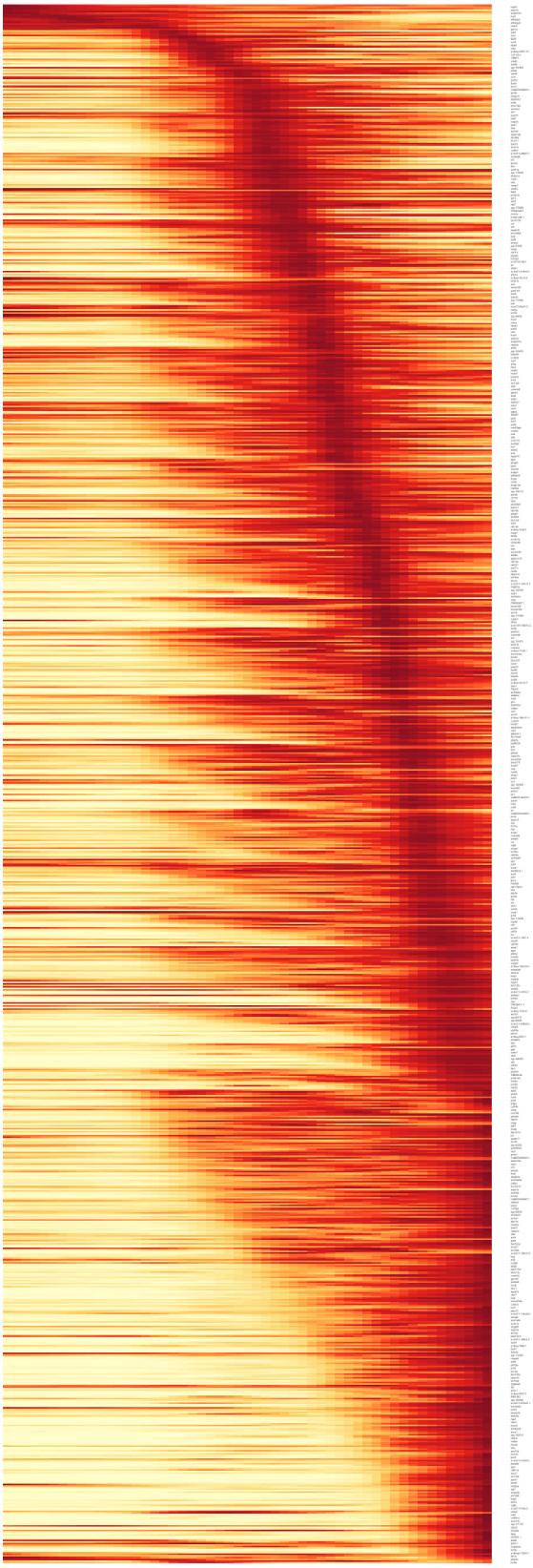
Prepare cascade

```
## RPE: Seg 11

# Get markers from the two approaches
m11 <- threshold.tree.markers(markers, "11", global.fc = 0.6)  # RPE markers from aucprTestAlongTree
rpe.markers <- rownames(m11)

# Just want to plot part of cells from upstream segment 31, which is very long.
# Going to use cells from segment 11 and from segment 31 with pseudotime > 0.23
cells.rpe <- unique(c(whichCells(obj, "pseudotime", c(0.23, 0.30308134)), cellsInCluster(obj,
    "segment", "11")))

# Calculate spline curves
spline.11 <- geneSmoothFit(obj, pseudotime = "pseudotime", cells = cells.rpe, genes = rpe.markers,
    method = "spline", moving.window = 5, cells.per.window = 25, pseudotime.per.window = 0.005,
    spar = 0.5, verbose = F)

# Calculate gene expression timing for ordering rows
spline.11 <- determine.timing(s = spline.11)
order.11 <- filter.heatmap.genes(spline.11$gene.order)

# Output gene table
table.save <- data.frame(gene = order.11, stringsAsFactors = F)
table.save$rpe.AUCPR.ratio.all <- m11[table.save$gene, "AUCPR.ratio.all"]
table.save$rpe.AUCPR.ratio.maxBranch <- m11[table.save$gene, "AUCPR.ratio.maxBranch"]
table.save$rpe.exp.fc.all <- m11[table.save$gene, "expfc.all"]
table.save$rpe.exp.fc.best <- m11[table.save$gene, "expfc.maxBranch"]
write.csv(table.save, quote = F, file = paste0(base.path, "/heatmaps/retina-rpe.csv"))
```

Generate heatmap: all genes

```
# Make sure any values <0 in the spline curves get set to 0 so that the heatmap
# scale doesn't get messed up.
spline.11$scaled.smooth[spline.11$scaled.smooth < 0] <- 0
# Open a PDF and generate the heatmap pdf(paste0(base.path,
# '/heatmaps/retina-rpe.pdf'), width=6, height=16)
gplots::heatmap.2(x = as.matrix(spline.11$scaled.smooth[order.11, ]), Rowv = F, Colv = F,
    dendrogram = "none", col = cols, trace = "none", density.info = "none", key = F,
    cexCol = 0.8, cexRow = 0.15, margins = c(8, 8), lwid = c(0.3, 4), lhei = c(0.3,
        4), labCol = NA)
title(main = "Retinal Pigmented Epithelium")
```

Retinal Pigmented Epithelium



```
# dev.off()
```

Continuous differentiation

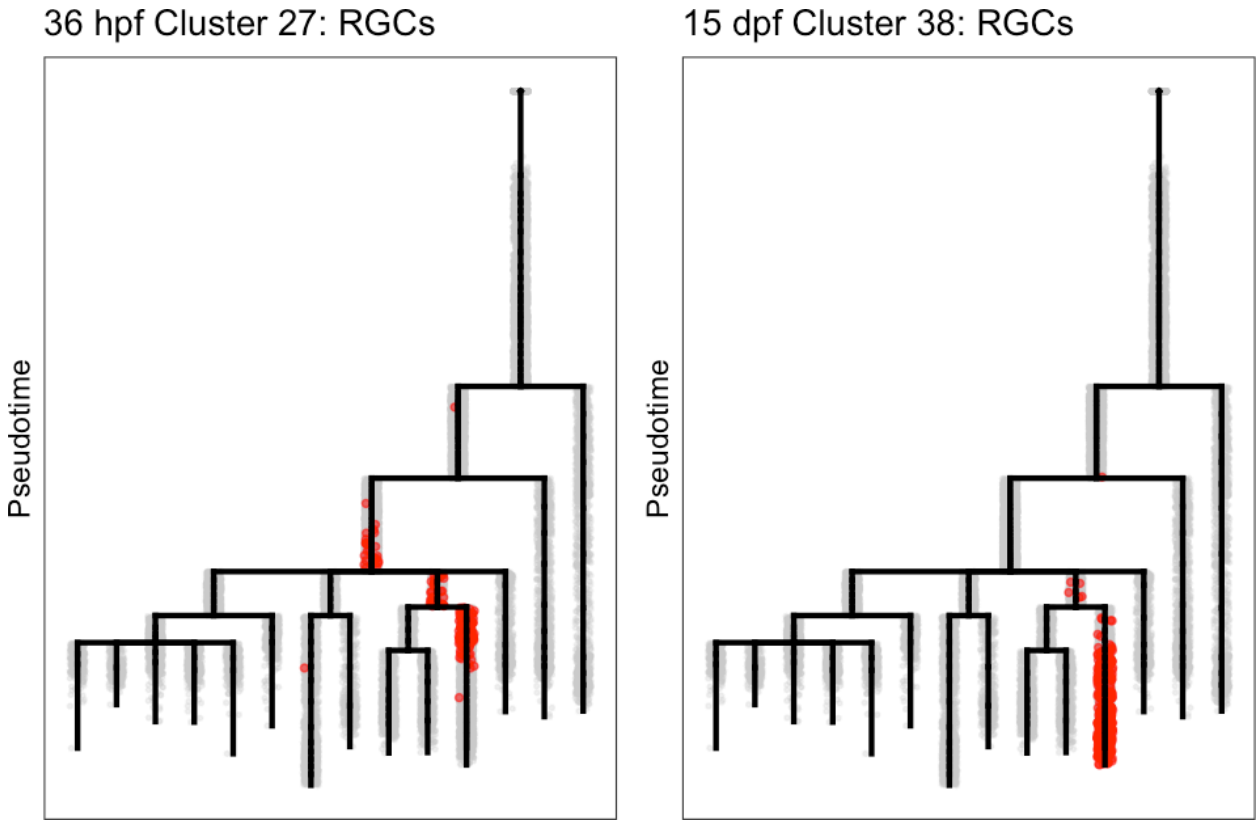
Retinal cell types were often found with similar molecular states across many stages of development. This reflects that pseudotime accurately represents the asynchrony introduced by continuous differentation.

RGC cells

```
gridExtra::grid.arrange(grobs = list(plotTreeHighlight(obj, label.name = "clus.orig",
    label.value = "7-36h_27", highlight.size = 1, title = "36 hpf Cluster 27: RGCs",
    label.x = F), plotTreeHighlight(obj, label.name = "clus.orig", label.value = "12-15d_38",
    highlight.size = 1, title = "15 dpf Cluster 38: RGCs", label.x = F)), ncol = 2)
```

```
## Warning: Removed 5 rows containing missing values (geom_point).
```

```
## Warning: Removed 17 rows containing missing values (geom_point).
```



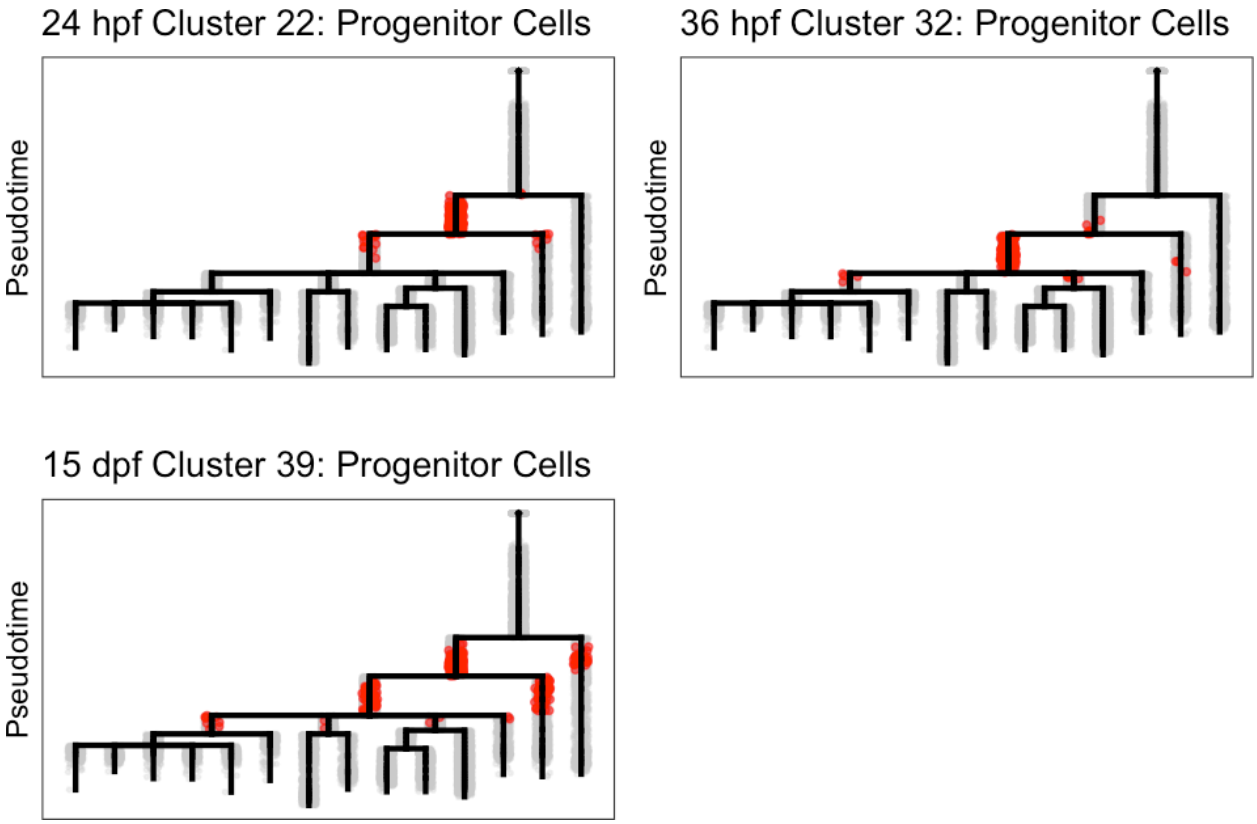
Progenitor cells

```
gridExtra::grid.arrange(grobs = list(plotTreeHighlight(obj, label.name = "clus.orig",
    label.value = "6-24h_22", highlight.size = 1, title = "24 hpf Cluster 22: Progenitor Cells",
    label.x = F), plotTreeHighlight(obj, label.name = "clus.orig", label.value = "7-36h_32",
    highlight.size = 1, title = "36 hpf Cluster 32: Progenitor Cells", label.x = F),
```

```
    plotTreeHighlight(obj, label.name = "clus.orig", label.value = "12-15d_39", highlight.size = 1,
        title = "15 dpf Cluster 39: Progenitor Cells", label.x = F)), ncol = 2)
```

```
## Warning: Removed 9 rows containing missing values (geom_point).
```

```
## Warning: Removed 4 rows containing missing values (geom_point).
```



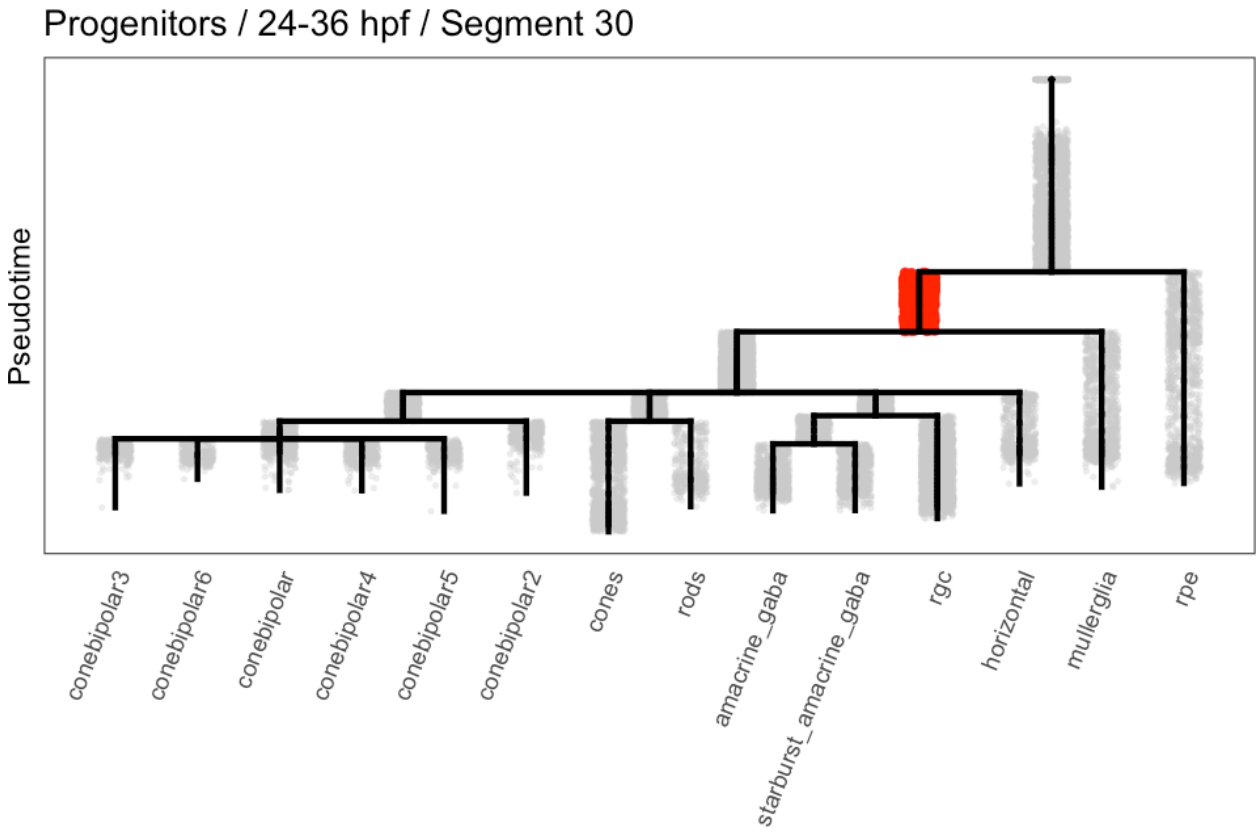
Progenitors over time

Retinal progenitors with similar transcriptional states are found across many different time points. We wanted to know whether there were significant transcriptional changes within those progenitors between early stages and late stages.

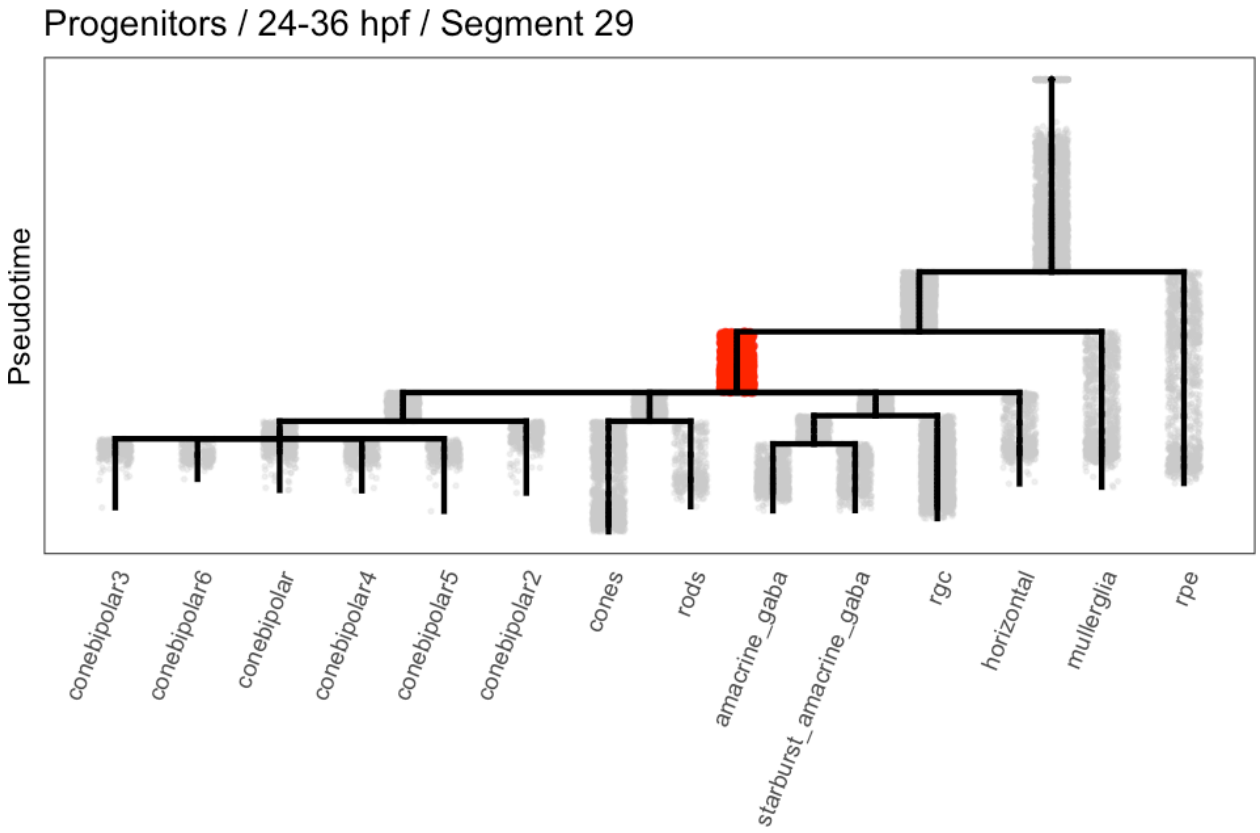
Identify populations

First we grabbed early (24 / 36 hpf) and late (15 dpf) progenitors from two sections of the tree.

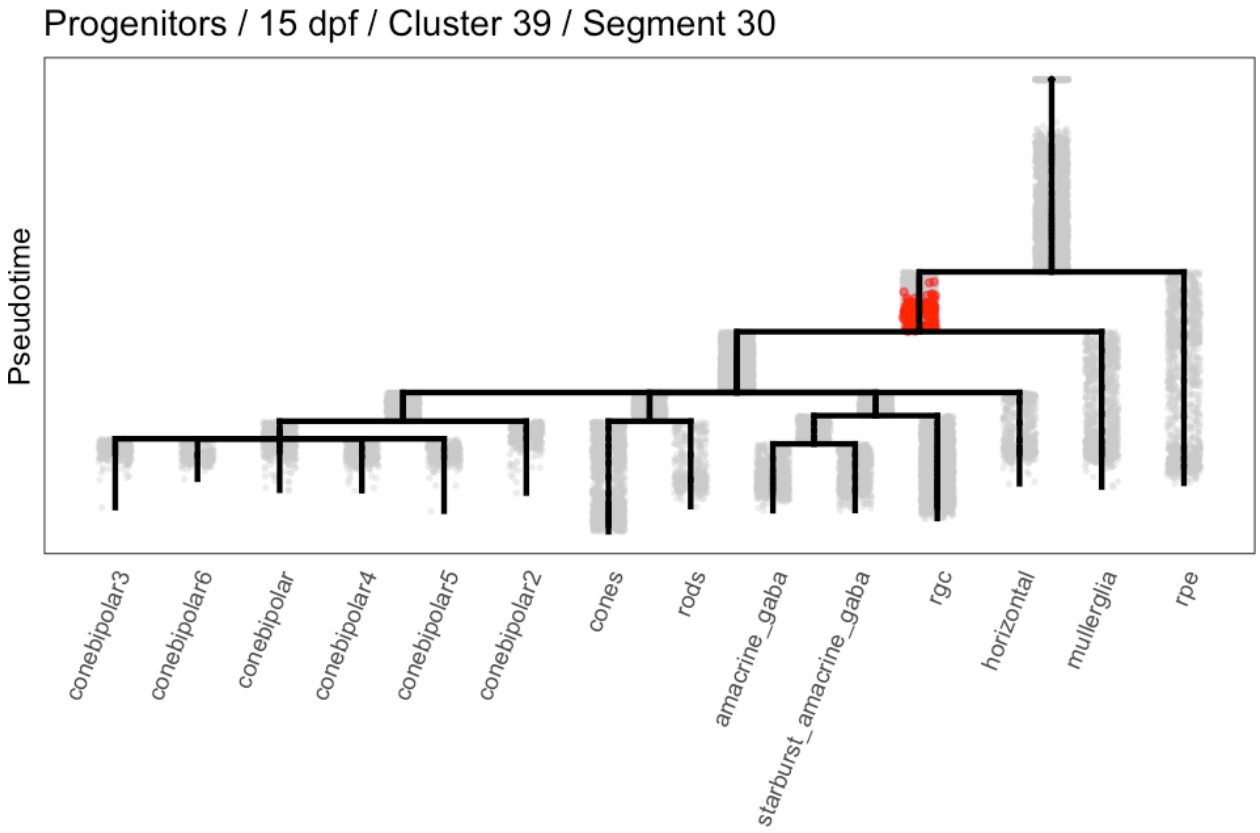
```
# Progenitors / 24-36 hpf / Segment 30
prog.early.s30 <- intersect(cellsInCluster(obj, "stage", c("06-24h", "07-36h")),
    cellsInCluster(obj, "segment", "30"))
obj <- groupFromCells(obj, group.id = "prog.early.s30", cells = prog.early.s30)
plotTreeHighlight(obj, "prog.early.s30", "TRUE", highlight.size = 1, highlight.alpha = 0.5,
    title = "Progenitors / 24-36 hpf / Segment 30")
```



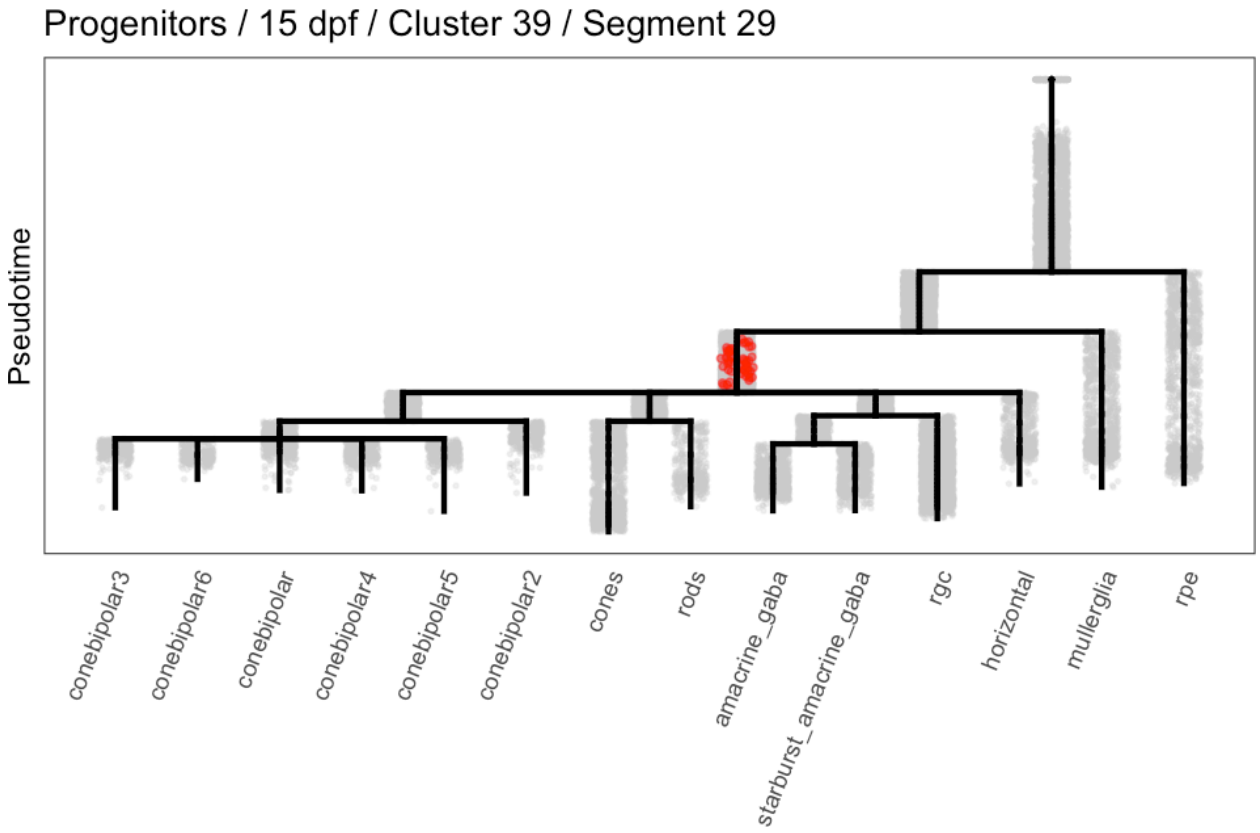

```
# Progenitors / 24-36 hpf / Segment 29
prog.early.s29 <- intersect(cellsInCluster(obj, "stage", c("06-24h", "07-36h")),
    cellsInCluster(obj, "segment", "29"))
obj <- groupFromCells(obj, group.id = "prog.early.s29", cells = prog.early.s29)
plotTreeHighlight(obj, "prog.early.s29", "TRUE", highlight.size = 1, highlight.alpha = 0.5,
    title = "Progenitors / 24-36 hpf / Segment 29")
```




```
# Progenitors / 15 dpf / Cluster 39 / Segment 30
prog.15d.c39.s30 <- intersect(cellsInCluster(obj, "clus.orig", "12-15d_39"), cellsInCluster(obj,
    "segment", "30"))
obj <- groupFromCells(obj, group.id = "prog.15d.c39.s30", cells = prog.15d.c39.s30)
plotTreeHighlight(obj, "prog.15d.c39.s30", "TRUE", highlight.size = 1, highlight.alpha = 0.5,
    title = "Progenitors / 15 dpf / Cluster 39 / Segment 30")
```




```
# Progenitors / 15 dpf / Cluster 39 / Segment 29
prog.15d.c39.s29 <- intersect(cellsInCluster(obj, "clus.orig", "12-15d_39"), cellsInCluster(obj,
    "segment", "29"))
obj <- groupFromCells(obj, group.id = "prog.15d.c39.s29", cells = prog.15d.c39.s29)
plotTreeHighlight(obj, "prog.15d.c39.s29", "TRUE", highlight.size = 1, highlight.alpha = 0.5,
    title = "Progenitors / 15 dpf / Cluster 39 / Segment 29")
```



Differential expression between neural progenitor populations

Then we determined what was differentially expressed between them.

```
# Compare 15d ('late') vs. early - S29
markers.nb.lve.s29 <- markersAUCPR(obj, cells.1 = prog.15d.c39.s29, cells.2 = prog.early.s29,
    auc.factor = 1.1, effect.size = 0.4)

# Compare 15d ('late') vs. early - S30
markers.nb.lve.s30 <- markersAUCPR(obj, cells.1 = prog.15d.c39.s30, cells.2 = prog.early.s30,
    auc.factor = 1.1, effect.size = 0.4)
```

boot.fc

- **object**: An URD object
- **cells.1**: Cells from group 1 of the differential expression
- **cells.2**: Cells from group 2 of the differential expression
- **cells.segment**: All cells in the segment that can be pulled for bootstrapping
- **genes.test**: Genes to test in the bootstrapping
- **exp.fc**: Exp.fc from the original differential expression test to compare for bootstrap
- **exp.data**: Can pre-calculated un-logged expression data to pass to the function (`getUPXData`)
- **n**: (Numeric) Number of bootstrap simulations to run
- Returns list: `p` is the empirical p-value for each differential expression, `boot.fc` contains all of the test information.

```
# Function to bootstrap fold-change
boot.fc <- function(object, cells.1, cells.2, cells.segment, genes.test, exp.fc,
```

```
    exp.data = NULL, n = 1000) {

    # Pull random populations of equivalent sizes
    l1 <- length(cells.1)
    l2 <- length(cells.2)
    random.pops <- lapply(1:1000, function(i) {
        y <- sample(x = cells.segment, size = l1 + l2, replace = F)
        return(list(a = y[1:l2], b = y[(l2 + 1):(l1 + l2)]))
    })

    # Get un-logged expression data, if not provided
    if (is.null(exp.data))
        exp.data <- getUPXData(object)

    # Calculate the expression fold-change for each random population
    fc.boot <- as.data.frame(lapply(1:n, function(i) {
        exp.a <- exp.data[genes.test, random.pops[[i]][["a"]]]
        exp.b <- exp.data[genes.test, random.pops[[i]][["b"]]]
        exp.fc <- log2((rowMeans(exp.a)/rowMeans(exp.b)) + 1)
        return(exp.fc)
    }))
    names(fc.boot) <- paste0("rep", 1:n)

    # Figure out p-value (proportion of these that beat provided exp.fc)
    beat.boot <- sweep(fc.boot, 1, exp.fc, ">")
    p.boot <- rowSums(beat.boot)/n

    # Return information
    return(list(p = p.boot, boot.fc = fc.boot))
}
```

Empirical p-value

Because these are relatively small populations, there's a decent chance that (due to the variability and noise inherent in scRNAseq data) that choosing any two similarly sized populations would find a number of differentially expressed genes also. Thus, we used an empirically-determined p-value to limit ourselves to differentially expressed genes that probably wouldn't arise by chance. We asked that our real comparison had a greater expression fold-change than two populations from a given segment of the same size chosen at random at least 99% of the time (i.e. $p < 0.01$).

```
# Try a bootstrapping approach to determine which markers are real, vs. which
# ones would arise just from small number of compared cells.  Going to just do it
# on expression fc, so that the computation is reasonably fast.

# Isolate cells from each segment
cells.seg.29 <- cellsInCluster(obj, "segment", "29")
cells.seg.30 <- cellsInCluster(obj, "segment", "30")

# Get un-logged expression data to pass to the function
exp.data <- getUPXData(obj)

# Run the actual bootstrapping.
boot.s30.lve <- boot.fc(object, cells.1 = prog.15d.c39.s30, cells.2 = prog.early.s30,
    cells.segment = cells.seg.30, genes.test = rownames(markers.nb.lve.s30), exp.fc = markers.nb.lve.s30$exp.fc,
```

```
    exp.data = exp.data, n = 1000)

boot.s29.lve <- boot.fc(object, cells.1 = prog.15d.c39.s29, cells.2 = prog.early.s29,
    cells.segment = cells.seg.29, genes.test = rownames(markers.nb.lve.s29), exp.fc = markers.nb.lve.s29$exp.fc,
    exp.data = exp.data, n = 1000)

# Limit markers to those that pass the bootstrap test
markers.nbb.lve.s30 <- markers.nb.lve.s30[which(boot.s30.lve$p <= 0.01), ]
markers.nbb.lve.s29 <- markers.nb.lve.s29[which(boot.s29.lve$p <= 0.01), ]
```

Tissue-specific changes

We also then divided genes based on whether they changed in all cells between 24/36 hpf and 15 dpf, or specifically in progenitors. Genes that change in all cells could represent either (a) global transcriptional changes in the tissue, or (b) changes in ambient RNA that is included with most cells based on highly expressed genes during different stages.

```
# Add global stage information to these - genes must change more in progenitors
# than just generally.

# Figure out early/late cells
cells.early <- cellsInCluster(obj, "stage", c("06-24h", "07-36h"))
cells.late <- cellsInCluster(obj, "stage", "12-15d")

# Calculate markers across stages generally with no restrictions
markers.nbball.lve <- markersAUCPR(object = obj, cells.1 = cells.late, cells.2 = cells.early,
    effect.size = -Inf, frac.must.express = 0, auc.factor = 0, genes.use = unique(c(rownames(markers.nbb.lve.s30)
        rownames(markers.nbb.lve.s29))))

# Transfer information to NBB comparisons
markers.nbb.lve.s30$exp.fc.stage <- markers.nbball.lve[rownames(markers.nbb.lve.s30),
    "exp.fc"]
markers.nbb.lve.s30$posFrac_stage1 <- markers.nbball.lve[rownames(markers.nbb.lve.s30),
    "posFrac_1"]

markers.nbb.lve.s29$exp.fc.stage <- markers.nbball.lve[rownames(markers.nbb.lve.s29),
    "exp.fc"]
markers.nbb.lve.s29$posFrac_stage1 <- markers.nbball.lve[rownames(markers.nbb.lve.s29),
    "posFrac_1"]

# Calculate ratios (i.e. how much more does a gene change in progenitors than in
# the entire tissue)
markers.nbb.lve.s30$exp.fc.ratio <- pmin(markers.nbb.lve.s30$exp.fc, 1000) - pmin(markers.nbb.lve.s30$exp.fc.stag
    1000)
markers.nbb.lve.s29$exp.fc.ratio <- pmin(markers.nbb.lve.s29$exp.fc, 1000) - pmin(markers.nbb.lve.s29$exp.fc.stag
    1000)

markers.nbb.lve.s30$posFrac.ratio <- markers.nbb.lve.s30$posFrac_1/markers.nbb.lve.s30$posFrac_stage1
markers.nbb.lve.s29$posFrac.ratio <- markers.nbb.lve.s29$posFrac_1/markers.nbb.lve.s29$posFrac_stage1
```

Limit to well-expressed

We also limited ourselves to genes that had a decent level of expression. (In this case, they were detected in at least 20% of progenitor cells, and had a mean expression of at least 0.8.)

```
# All genes that change in segment 30
markers.nbbexp.lve.s30 <- markers.nbb.lve.s30[Reduce(intersect, list(which(markers.nbb.lve.s30$posFrac_1 >=
    0.2), which(markers.nbb.lve.s30$nTrans_1 >= 0.8))), ]

# All genes that change in segment 30
markers.nbbexp.lve.s29 <- markers.nbb.lve.s30[Reduce(intersect, list(which(markers.nbb.lve.s29$posFrac_1 >=
    0.2), which(markers.nbb.lve.s29$nTrans_1 >= 0.8))), ]

# Genes that change in segment 30 more than in the entire tissue
markers.nbbselect.lve.s30 <- markers.nbb.lve.s30[Reduce(intersect, list(which(markers.nbb.lve.s30$posFrac_1 >=
    0.2), which(markers.nbb.lve.s30$nTrans_1 >= 0.8), which(markers.nbb.lve.s30$exp.fc.ratio >=
    1.2), which(markers.nbb.lve.s30$posFrac.ratio >= 1.1))), ]

# Genes that change in segment 29 more than in the entire tissue
markers.nbbselect.lve.s29 <- markers.nbb.lve.s29[Reduce(intersect, list(which(markers.nbb.lve.s29$posFrac_1 >=
    0.2), which(markers.nbb.lve.s29$nTrans_1 >= 0.8), which(markers.nbb.lve.s29$exp.fc.ratio >=
    1.2), which(markers.nbb.lve.s29$posFrac.ratio >= 1.1))), ]
```

Result

That recovered a total of 71 genes that vary in progenitors between 24/36 hpf and 15 dpf, of which 16 change more in neural progenitors than the rest of the tissue.

```
# All genes that change in progenitors
unique(c(rownames(markers.nbbexp.lve.s29), rownames(markers.nbbexp.lve.s30)))
```

```
##  [1] "hbbe2"             "hbz"               "ba1.1"
##  [4] "rho"               "crabp1a"           "si:ch211-251b21.1"
##  [7] "hbaa1"             "pde6h"             "tsc22d3"
## [10] "si:xx-by187g17.1"  "ba1"               "rpe65a"
## [13] "zgc:153704"        "arr3a"             "lin7a"
## [16] "crygm1"            "gnat1"             "ptgdsb.1"
## [19] "gngt1"             "rgs16"             "cabp2a"
## [22] "junba"             "crygm2b"           "zgc:112320"
## [25] "si:dkey-183i3.5"   "krt91"             "cabp5a"
## [28] "sagb"              "crygmx"            "scinla"
## [31] "rbp4l"             "gngt2b"            "rs1a"
## [34] "mt2"               "fosab"             "cebpd"
## [37] "snap25b"           "CNDP1"             "crabp2a"
## [40] "cryba4"            "jdp2b"             "cst3"
## [43] "higd1a"            "mt-nd3"            "si:dkey-16p21.8"
## [46] "crybb1"            "crygn2"            "gadd45ba"
## [49] "gapdhs"            "eno1a"             "mif"
## [52] "ggctb"             "ckbb"              "glula"
## [55] "tsc22d1"           "sod2"              "btg2"
## [58] "sod1"              "stmn1b"            "si:dkey-238o13.4"
## [61] "fabp11a"           "mdkb"              "gstp1"
## [64] "slc3a2b"           "si:dkey-238c7.12"  "CABZ01102240.1"
## [67] "atp5ia"            "atpif1b"           "cadm3"
## [70] "h1f0"
```

```
# Genes that change in progenitors more than the rest of the tissue
unique(c(rownames(markers.nbbselect.lve.s29), rownames(markers.nbbselect.lve.s30)))
```

```
##  [1] "si:ch211-114n24.6" "rps29"             "rrm2.1"
##  [4] "si:ch211-193l2.6"  "si:dkey-238o13.4"  "crabp1a"
##  [7] "si:ch211-251b21.1" "CNDP1"             "junba"
## [10] "crabp2a"           "cryba4"            "crybb1"
## [13] "crygn2"            "fabp11a"           "si:dkey-238c7.12"
## [16] "cadm3"
```

Preservation of embryonic molecular profiles in larval progenitors

We were looking to compare the molecular profiles of embryonic and post-embryonic progenitors. Here, in the retina, we find progenitors at larval stages whose molecular signatures are preserved from embryonic stages. For comparison, in the hypothalamus, we find that progenitors at larval stages are transcriptionally different from embryonic progenitors (see Hypothalamus 3).

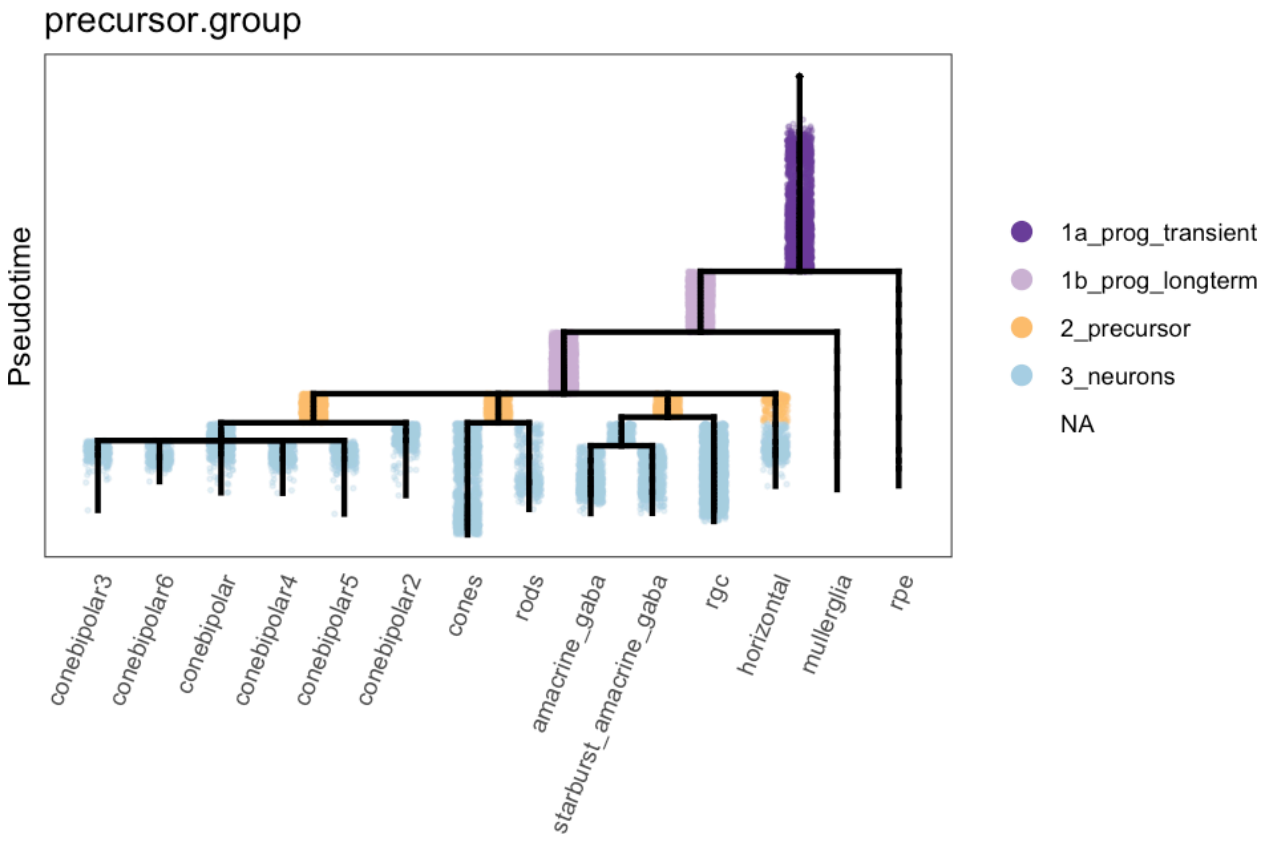
Identify populations to compare

We defined progenitor / precursor / neuron populations based on their location in the tree, cross-referenced with the expression of markers of each of these types

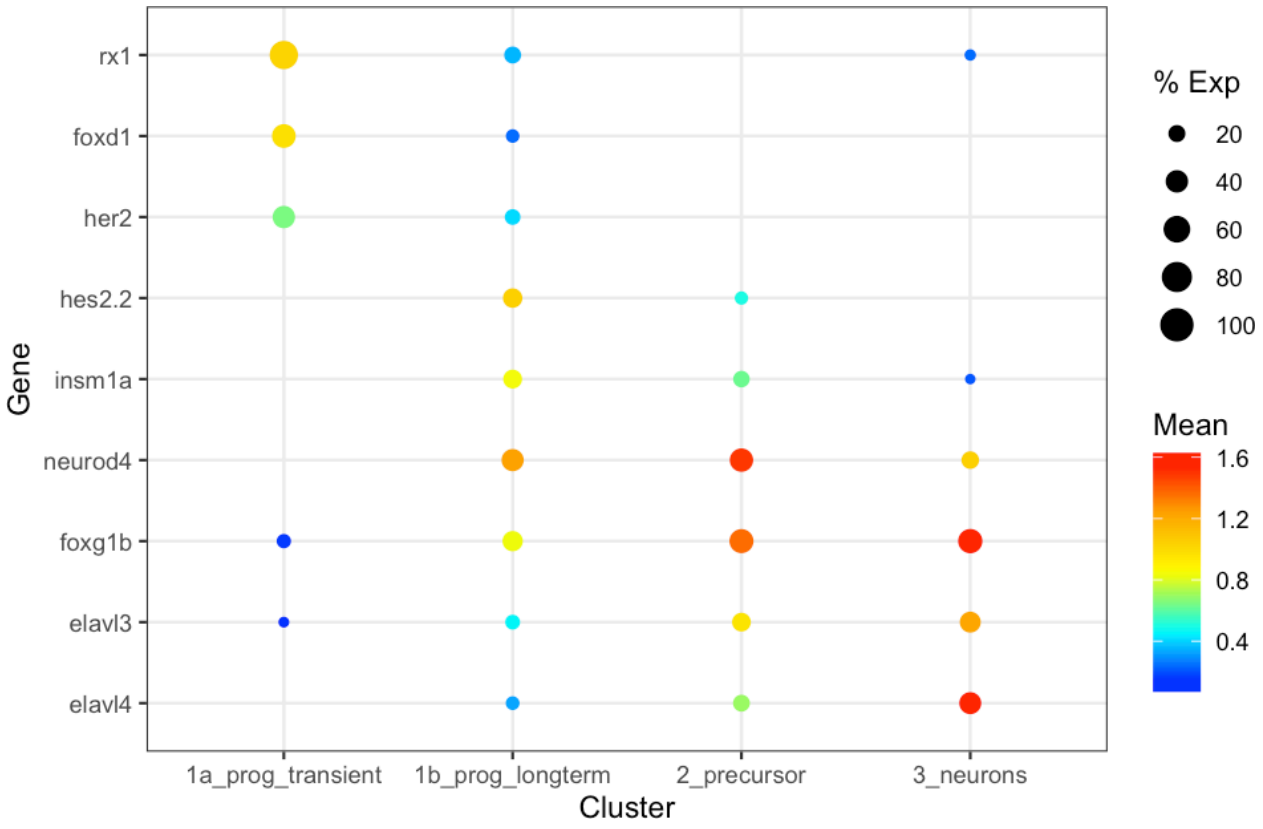
```
obj@group.ids$precursor.group <- NA

cells.s31 <- intersect(cellsInCluster(obj, "segment", "31"), whichCells(obj, "pseudotime",
    c(0.05, 1)))
obj@group.ids[cells.s31, "precursor.group"] <- "1a_prog_transient"

cells.prog.late <- cellsInCluster(obj, "segment", c("30", "29"))
obj@group.ids[cells.prog.late, "precursor.group"] <- "1b_prog_longterm"

cells.precursor <- intersect(cellsInCluster(obj, "segment", c("24", "25", "26", "15")),
    whichCells(obj, "pseudotime", c(0, 0.535)))
obj@group.ids[cells.precursor, "precursor.group"] <- "2_precursor"

cells.neurons <- setdiff(whichCells(obj, "pseudotime", c(0.535, 1)), cellsInCluster(obj,
    "segment", c("6", "11")))
obj@group.ids[cells.neurons, "precursor.group"] <- "3_neurons"
```

```
# Colors for ggplot
stage.colors <- RColorBrewer::brewer.pal(12, "Paired")[c(10, 9, 7, 1)]

# Plot tree to show where the groups are
plotTree(obj, "precursor.group", discrete.colors = stage.colors)
```

```
## Warning: Removed 2530 rows containing missing values (geom_point).
```




```
# Plot genes in each group
plotDot(obj, genes = c("rx1", "foxd1", "her2", "hes2.2", "insm1a", "neurod4", "foxg1b",
    "elavl3", "elavl4"), clustering = "precursor.group", scale.by = "area") + theme_bw()
```



Determine proportion of cells in each state

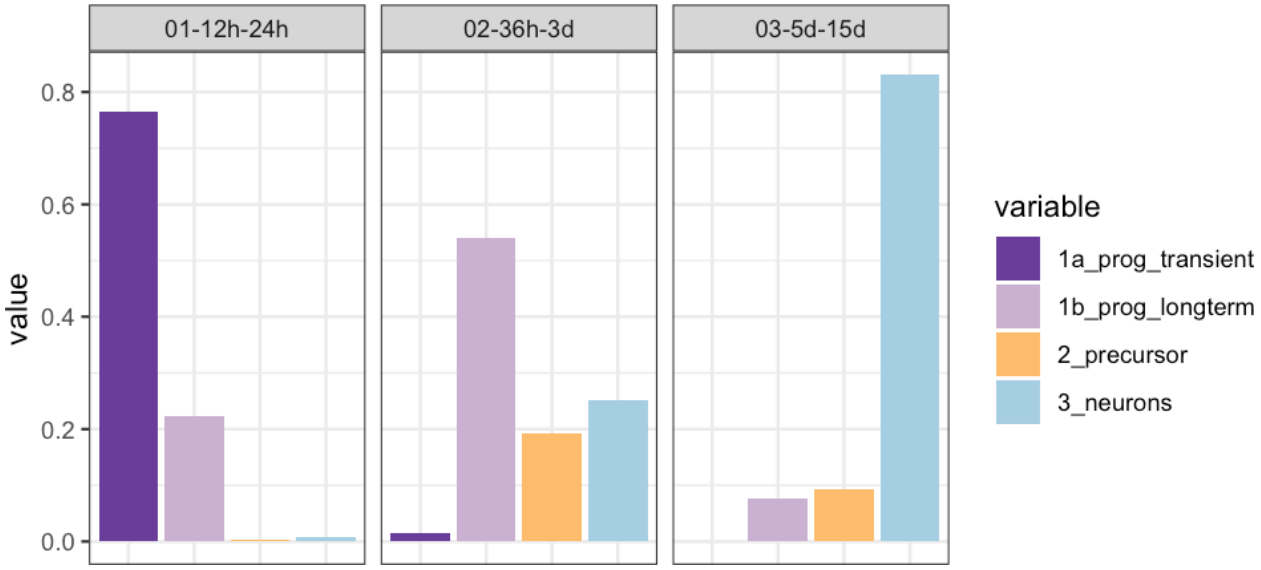
We then determined the proportion of cells from different stages that fell into each of these transcriptional states.

```
# We combined stages to reduce number of plots
obj@group.ids$stage.collapsed <- plyr::mapvalues(x = obj@group.ids$stage, from = c("01-12h",
    "02-14h", "03-16h", "04-18h", "05-20h", "06-24h", "07-36h", "08-2d", "09-3d",
    "10-5d", "11-8d", "12-15d"), to = c(rep("01-12h-24h", 6), rep("02-36h-3d", 3),
    rep("03-5d-15d", 3)))

# Count number of cells from each stage group in each precursor group
stage.group.count <- plyr::count(obj@group.ids, vars = c("stage.collapsed", "precursor.group"))

# Remove cells that weren't part of a precursor group
stage.group.count <- stage.group.count[complete.cases(stage.group.count), ]

# Cast into a data frame and convert NA to 0 (no cells of that type observed)
stage.group.df <- reshape2::dcast(stage.group.count, formula = stage.collapsed ~
    precursor.group)
```

```
## Using freq as value column: use value.var to override.
```

```
stage.group.df[is.na(stage.group.df)] <- 0

# Normalize by the number of precursors from each stage group
stage.group.df[, 2:5] <- sweep(stage.group.df[, 2:5], 1, rowSums(stage.group.df[,
    2:5]), "/")
```

```r
# Melt for ggplot
stage.group.df.melt <- reshape2::melt(stage.group.df, id.vars = "stage.collapsed")

# Plot proportions
ggplot(stage.group.df.melt, aes(x = variable, y = value, group = stage.collapsed,
    fill = variable)) + geom_bar(stat = "identity") + facet_wrap(~stage.collapsed) +
    theme_bw() + scale_fill_manual(values = stage.colors) + theme(axis.title.x = element_blank(),
    axis.text.x = element_blank(), axis.ticks.x = element_blank())
```



Hypothalamus: 1 - URD object & doublet removal

Jeff Farrell

8/22/2019

Contents

Import data into URD

Convert Seurat object to URD

We first loaded a Seurat object that contained just cells from the clusters that belonged to the hypothalamus from each stage.

```
suppressPackageStartupMessages(library(URD))
suppressPackageStartupMessages(library(Seurat))
```

```
base.path <- "~/urd-cluster-bushra/"

# Load Seurat object that has been cropped to hypothalamus cells
object.seurat <- readRDS(paste0(base.path, "obj/hypo_seurat.rds"))

# Convert to URD object
suburd <- seuratToURD(object.seurat)
```

Combined individual stage clustering

Bushra had performed individual clusterings across each stage with different resolutions. Here, it was better to create a single identifier that included stage + cluster information to combine all those clusterings (while preventing any overlap).

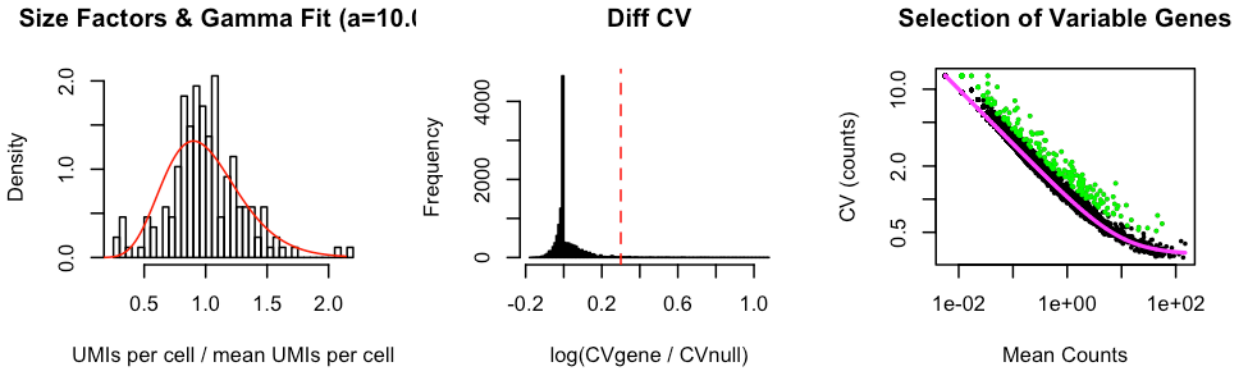
```
stages <- sort(unique(suburd@meta$stage))
clust.res.used <- paste0("res.", c("4.5", "4", "5", "5", "4.5", "5", "6",
    "6", "6", "5.5", "6", "5"))
names(clust.res.used) <- stages
suburd@group.ids$cluster <- NA
for (stage in stages) {
    suburd@group.ids[cellsInCluster(suburd, "stage", stage), "cluster"] <- paste0(stage,
        "-", suburd@group.ids[cellsInCluster(suburd, "stage", stage), clust.res.used[stage]])
}
```

Calculate highly variable genes

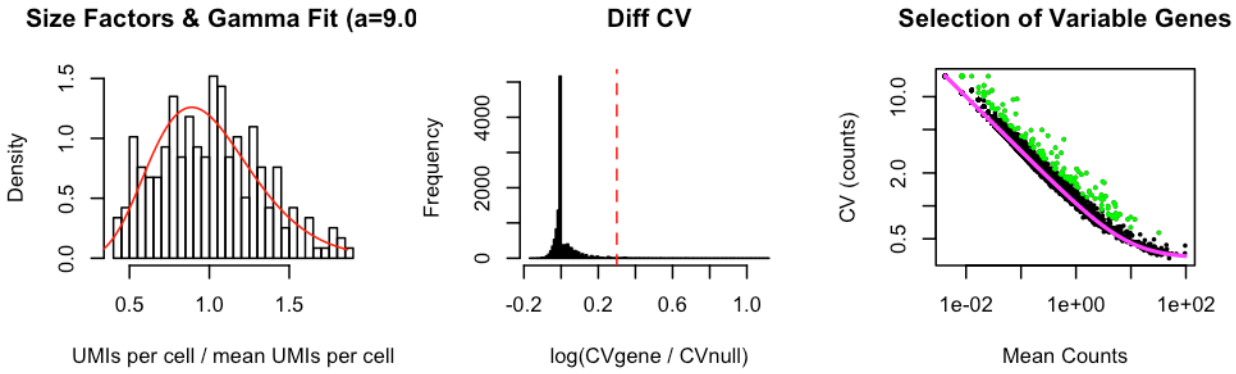
We calculated highly variable genes for each stage, used genes that were found as highly in at least two stages, but were not mitochondrial, ribosomal, heat-shock protein, or tandem duplicated genes.

```
# Calculated on each stage separaely, final gene list was all genes
# that were 'variable' in at least two stages NB: For a couple of
# stages, the gamma fit was poor -- the library size distribution
# seemed bimodal. Have seen this before in 10X data, but not sure what
# it means.
var.genes.by.stage <- lapply(stages, function(stage) {
    findVariableGenes(suburd, cells.fit = cellsInCluster(suburd, "stage",
        stage), set.object.var.genes = F, diffCV.cutoff = 0.3, main.use = stage,
        do.plot = T)
})
```

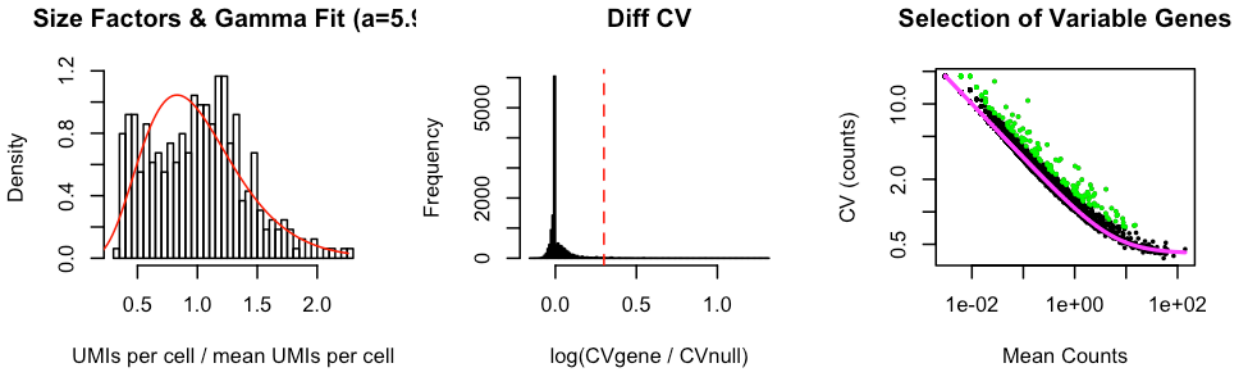
01-12h



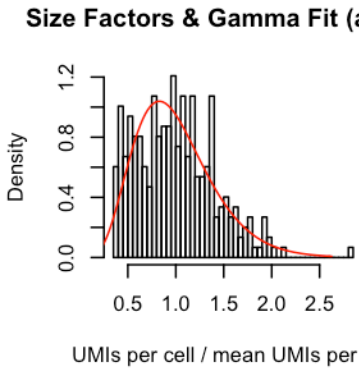
02-14h

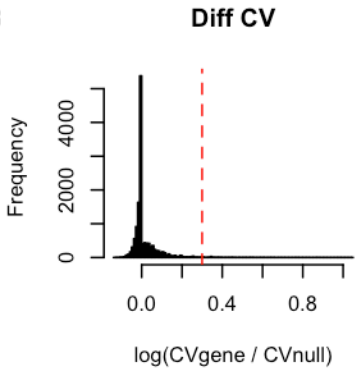


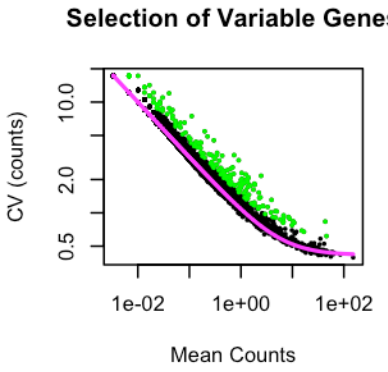
03-16h



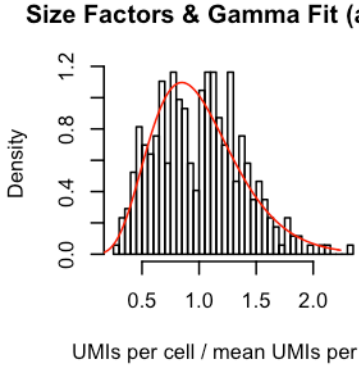
04-18h

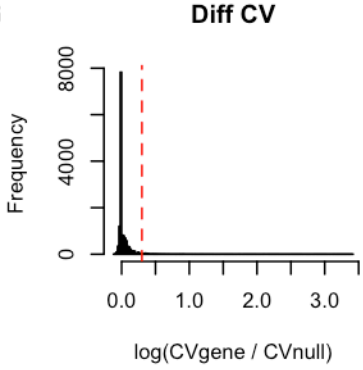


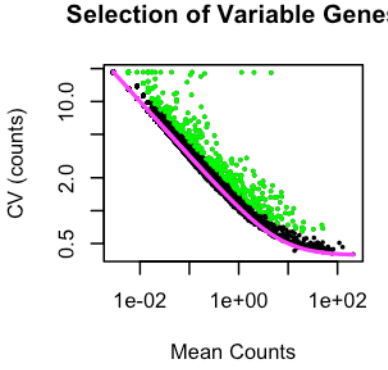




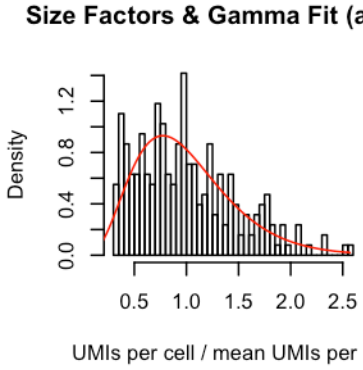
05-20h

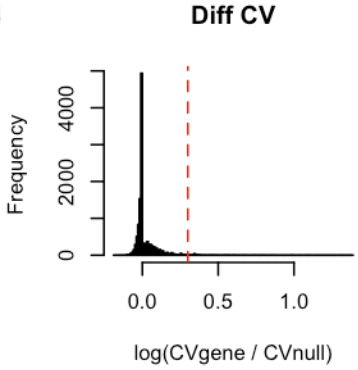


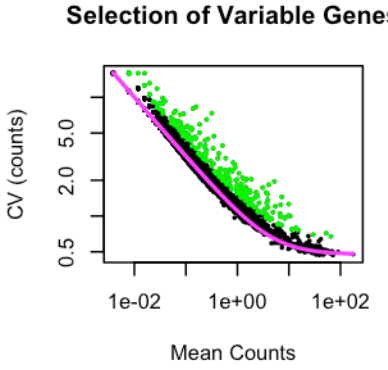




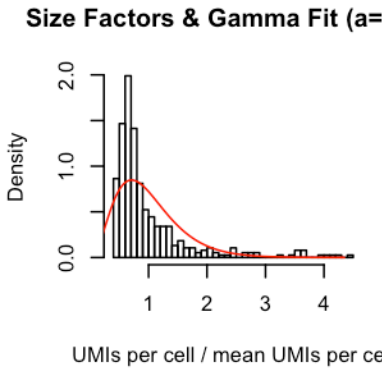
06-24h

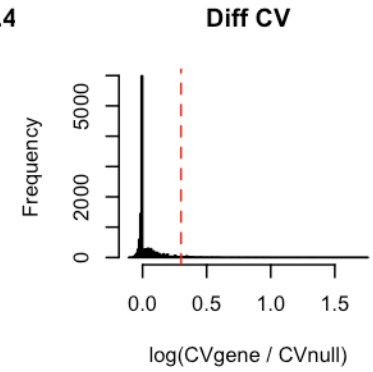


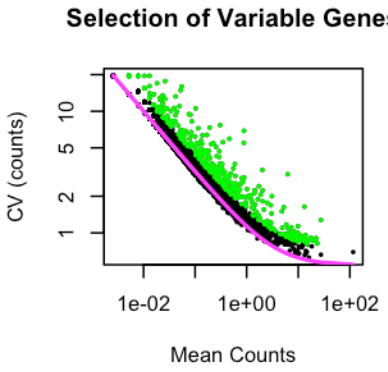




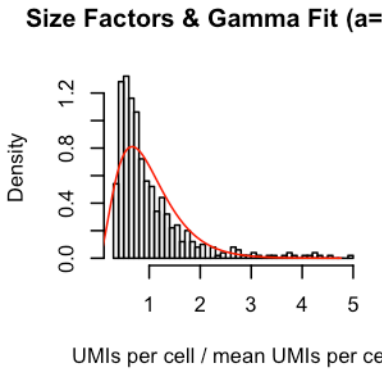
07-36h

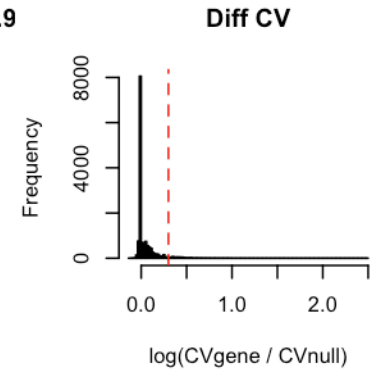


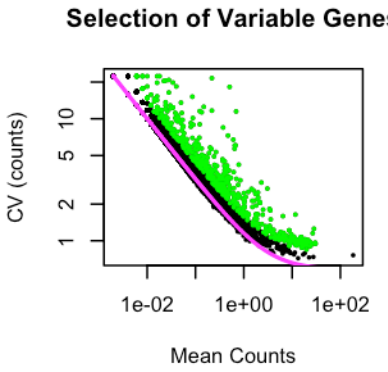




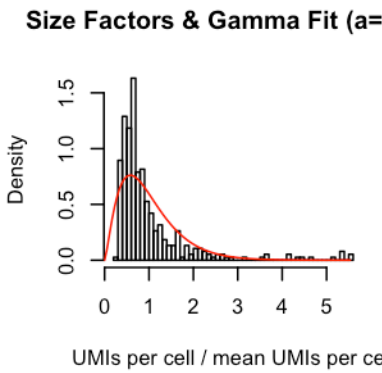
08-2d

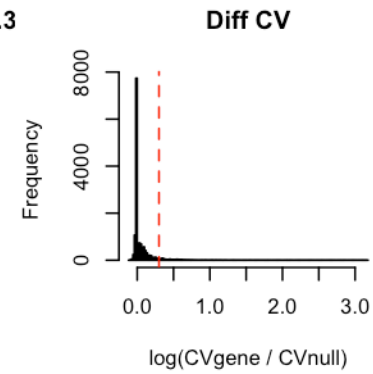


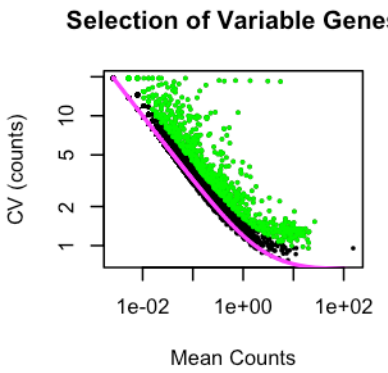


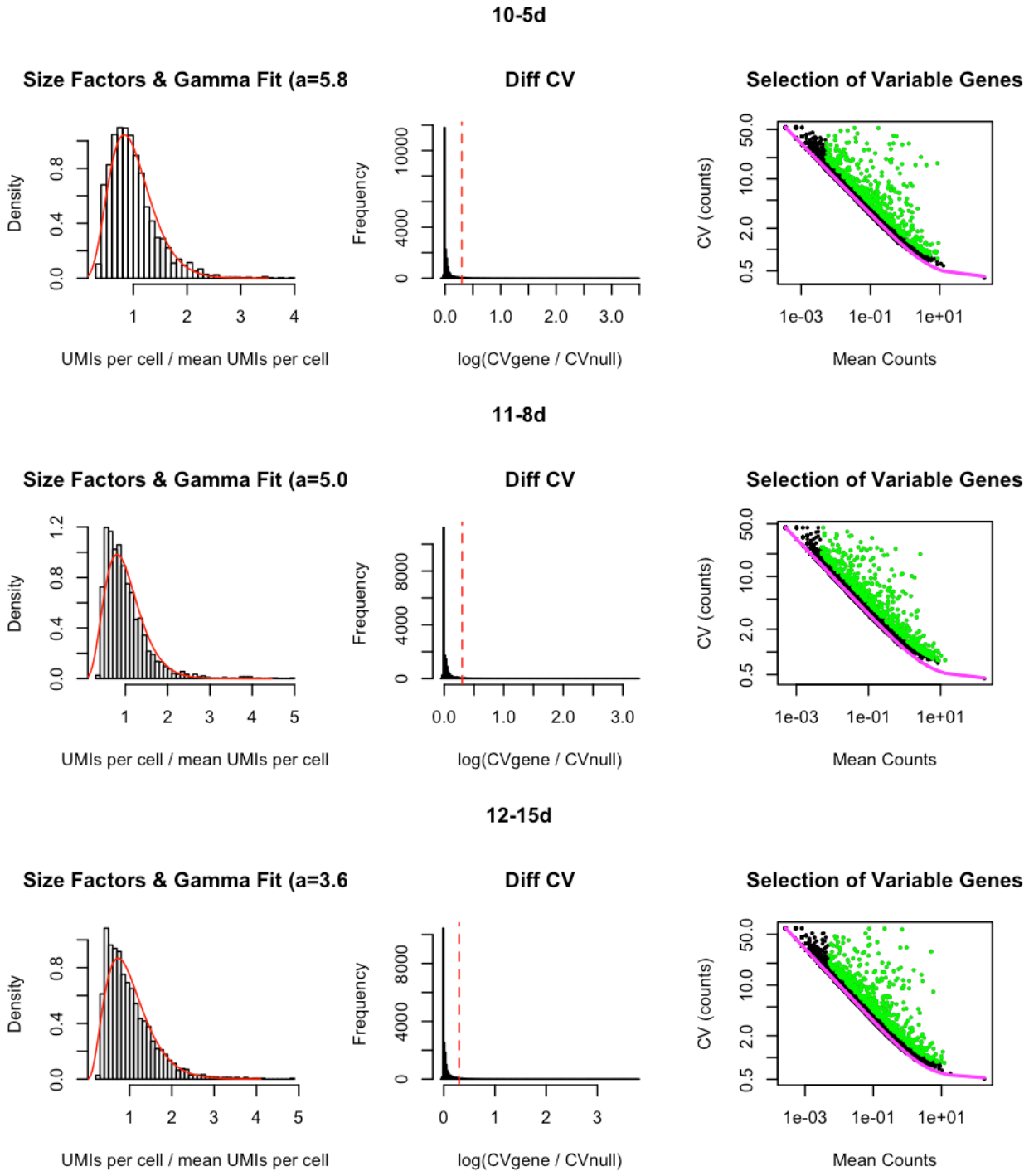


09-3d










```
names(var.genes.by.stage) <- stages
var.genes <- sort(unique(unlist(var.genes.by.stage)))
print(paste0("Length of variable genes is ", length(var.genes)))
```

```
## [1] "Length of variable genes is 1783"
```

```
var.genes.twice <- names(which(table(unlist(var.genes.by.stage)) >= 2))
print(paste0("Length of variable genes shared across at least 2 stages is ",
    length(var.genes.twice)))
```

```
## [1] "Length of variable genes shared across at least 2 stages is 957"
```

```
# Remove mitochondrial genes
var.mito <- grep("^mt-|^AC0", var.genes.twice, value = T)
# Remove ribosomal genes
var.ribo <- grep("^rps|^rpl", var.genes.twice, value = T)
# Remove hsp genes
var.hsp <- grep("^hsp", var.genes.twice, value = T)
# Remove genes with duplicates
var.dups <- grep("of many", var.genes.twice, value = T)
suburd@var.genes <- setdiff(var.genes.twice, c(var.mito, var.ribo, var.hsp,
    var.dups))
print(paste0("Length of final variable genes list (after removing mito, ribo, hsp genes) is ",
    length(suburd@var.genes)))
```

```
## [1] "Length of final variable genes list (after removing mito, ribo, hsp genes) is 856"
```

To prevent downstream problems, we also removed any cells from the data that had the exact same expression of the variable genes (i.e. cells with completely duplicated coordinates in the high-dimensional space we would use for analysis downstream).

```
# Check for duplicate data points - cells with exact same expression of
# variable genes
vg.dups <- duplicated(as.data.frame(as.matrix(t(suburd@logupx.data[suburd@var.genes,
    ]))))
if (length(which(vg.dups)) > 0) {
    print(paste("Removing", length(which(vg.dups)), "cell(s) with duplicated variable gene expression."))
    not.dup.cells <- colnames(suburd@logupx.data)[!vg.dups]
    suburd <- urdSubset(suburd, not.dup.cells)
}
```

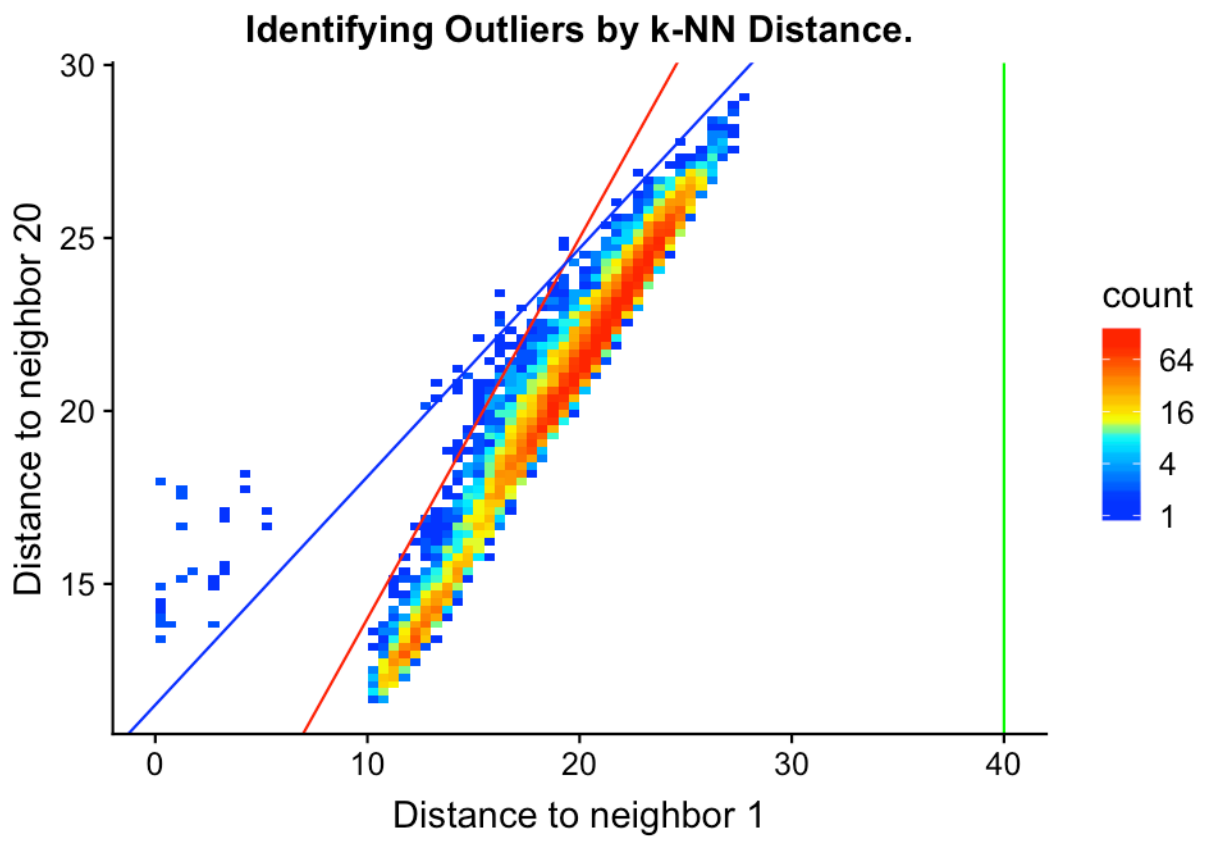
```
## [1] "Removing 1 cell(s) with duplicated variable gene expression."
```

Calculate KNN graph and remove outliers

We then calculated a k-nearest neighbor graph and removed cells that had unusual distance to their nearest neighbor, or unusual distance to their 20th nearest neighbor (given their distance to their nearest neighbor). These sorts of outliers often cause problems or skew diffusion maps (used downstream).

```
# Calculate k-nn
suburd <- calcKNN(suburd)
```

```
# Check what the outliers are
outliers <- knnOutliers(suburd, nn.1 = 1, nn.2 = 20, x.max = 40, slope.r = 1.1,
    int.r = 3, slope.b = 0.66, int.b = 11.5, title = "Identifying Outliers by k-NN Distance.")
```




```
length(outliers)
```

```
## [1] 87
```

```
suburd <- urdSubset(suburd, cells.keep = setdiff(colnames(suburd@logupx.data),
    outliers))
```

Remove cell type doublets

Add UMAP projection

While not strictly required, a UMAP projection can make it easier to assess the expression of NMF modules and whether thresholds for overlap are set correctly.

```
## add UMAP command

# Load pre-calculated UMAP
umap <- readRDS(paste0(base.path, "/umap/umap_hypo.rds"))

# Add projection to URD object
suburd@tsne.y <- umap
```

Load NMF results and import into object

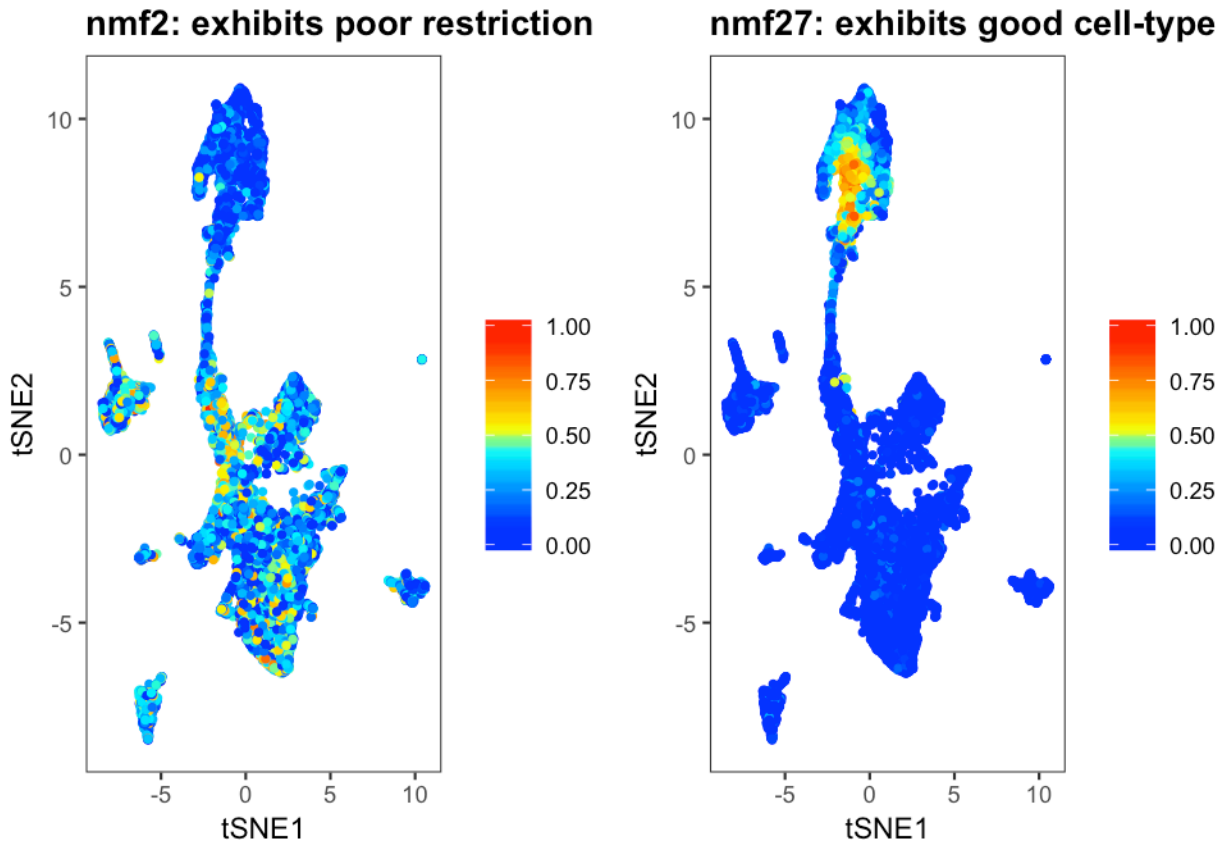
NMF results were calculated by providing `suburd@logupx.data` to an external NMF pipeline written in Python. The output results are imported here, scaled, and added to the URD object.

```
# Load the NMF results
load(paste0(base.path, "/NMF/hypo/result_tbls.Robj"))

# The results object contains NMF runs for several K values.  k=28 was
# chosen for this tissue, so this extracts the results for that
# particular parameter
k.use <- "28"
nmf.cells <- result_obj[[paste0("K=", k.use)]][[1]]$C
rownames(nmf.cells) <- paste0("nmf", 1:nrow(nmf.cells))
colnames(nmf.cells) <- gsub("\\.", "-", colnames(nmf.cells))
nmf.genes <- result_obj[[paste0("K=", k.use)]][[1]]$G
colnames(nmf.genes) <- paste0("nmf", 1:nrow(nmf.cells))

# Scale NMF results 0-1
nmf.cells.scaled <- sweep(nmf.cells, 1, apply(nmf.cells, 1, max), "/")

# Add scaled NMF results to the URD object
suburd@nmf.c1 <- as(t(as.matrix(nmf.cells.scaled)), "dgCMatrix")
```

Select cell-type specific modules

Several NMF modules will be poor markers of cell types — these are often modules driven mostly be the expression of 1-2 genes (where the gene loading of the first gene is much greater than that of the fourth gene, for instance), or modules that don't exhibit any restriction in a tSNE or UMAP projection.

```
# Plot size parameters
plot.height = 6
plot.width = 16
dpi = 150

# Plot every module to determine which exhibit cell-type specificity
# This saves directly to the hard drive: two example plots are shown
# below.

# for (n in colnames(suburd@nmf.c1)) { png(paste0(path, '/doublets/',
# subset, '-plots/', n, '.png'), width=dpi*plot.width,
# height=dpi*plot.height) plot(plotDim(suburd, n)) dev.off() }

gridExtra::grid.arrange(grobs = list(plotDim(suburd, "nmf2", plot.title = "nmf2: exhibits poor restriction"),
    plotDim(suburd, "nmf27", plot.title = "nmf27: exhibits good cell-type")),
    ncol = 2)
```



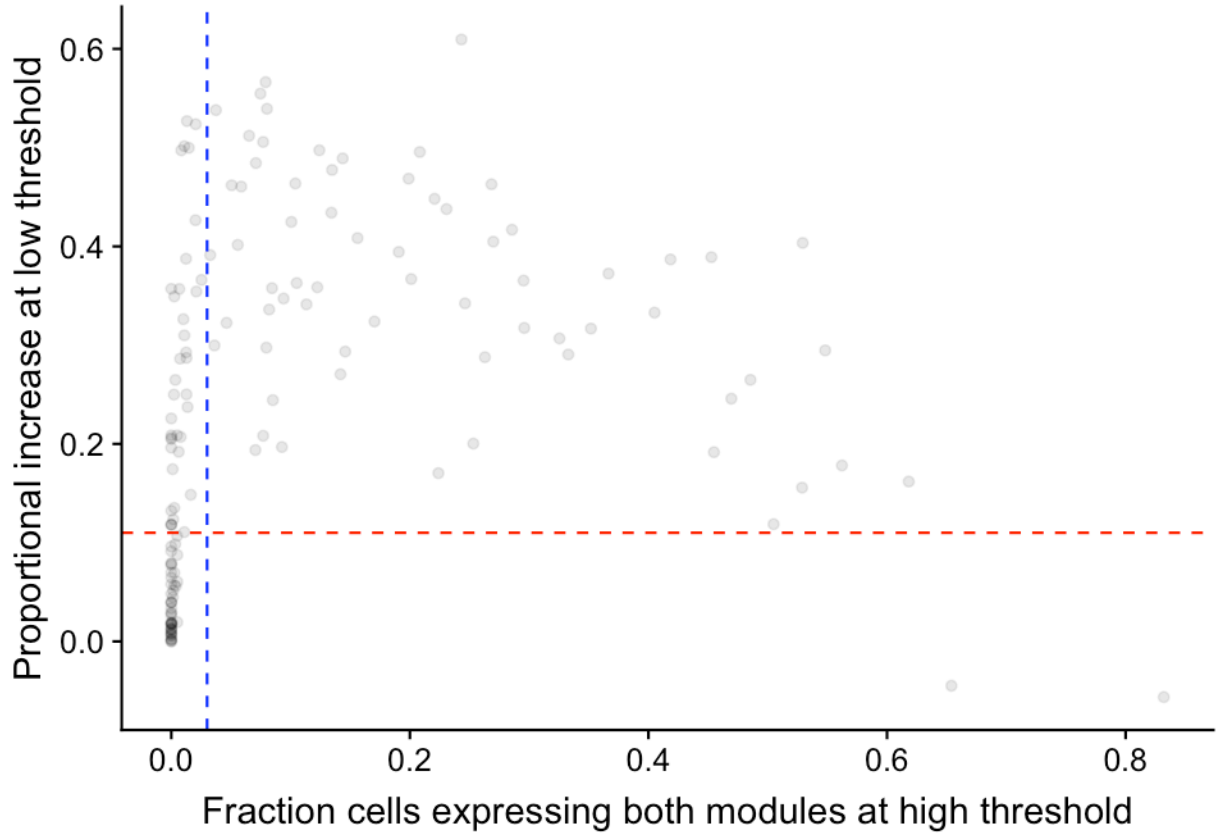

```
# Module Gene 1 : Gene 4 Ratios
top.genes <- result_obj[[paste0("K=", k.use)]][[1]]$top30genes
top.weights <- top.genes[, grep("Weights", colnames(top.genes), value = T)]
colnames(top.weights) <- paste0("nmf", 1:nrow(nmf.cells))
top.weights.ratio <- top.weights[1, ]/top.weights[4, ]

# Which modules exhibit cell-type restriction?
modules.bad.ratio <- names(top.weights.ratio)[which(top.weights.ratio >
    3)]
unrestricted.modules <- paste0("nmf", c("12", "19", "24", "28"))
good.modules <- setdiff(colnames(suburd@nmf.c1), c(modules.bad.ratio, unrestricted.modules))
```

Determine which module pairs to use for doublet removal

We consider NMF modules pairwise and only use those pairs that don't are non-overlapping in the data. (In other words, NMF modules that are mutually exclusive in the majority of the data.) Here, we determine thresholds for selecting those module pairs.

```
# Determine overlaps between module pairs
nmf.doublet.combos <- NMFDoubletsDefineModules(suburd, modules.use = good.modules,
    module.thresh.high = 0.4, module.thresh.low = 0.15)

# Determine thresholds for NMF modules
frac.overlap.max = 0.03
frac.overlap.diff.max = 0.11
```

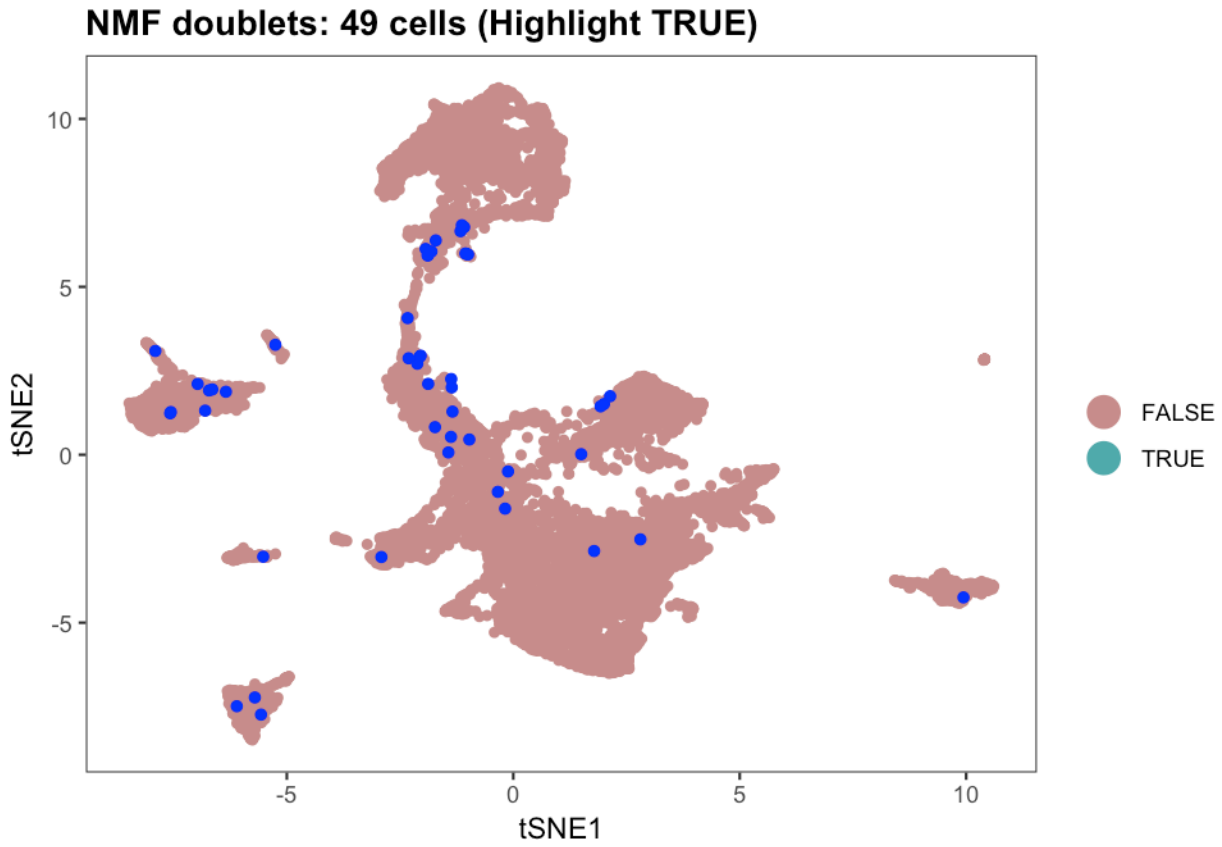
```
module.expressed.thresh = 0.33

# Determine which module pairs to use for doublets
NMFDoubletsPlotModuleThresholds(nmf.doublet.combos, frac.overlap.max = frac.overlap.max,
    frac.overlap.diff.max = frac.overlap.diff.max)
```




```
# These commands save plots directly to the hard-drive.

# Make plots to see how your thresholds are
NMFDoubletsPlotModuleCombos(suburd, path = paste0(path, "/doublets/", subset,
    "-doublet-combos/"), module.combos = nmf.doublet.combos, module.expressed.thresh = module.expressed.thresh,
    frac.overlap.max = frac.overlap.max, frac.overlap.diff.max = frac.overlap.diff.max,
    boundary = "pass", sort = "near", n.plots = 25)
NMFDoubletsPlotModuleCombos(suburd, path = paste0(path, "/doublets/", subset,
    "-ok-combos/"), module.combos = nmf.doublet.combos, module.expressed.thresh = module.expressed.thresh,
    frac.overlap.max = frac.overlap.max, frac.overlap.diff.max = frac.overlap.diff.max,
    boundary = "discarded", sort = "near", n.plots = 25)
```

```
# Define doublet cells
nmf.doublets <- NMFDoubletsDetermineCells(suburd, nmf.doublet.combos, module.expressed.thresh = module.expressed.
    frac.overlap.max = frac.overlap.max, frac.overlap.diff.max = frac.overlap.diff.max)  # 49 cells / 11307 cells

# Plot doublet cells on the UMAP
suburd <- groupFromCells(suburd, "nmf.doublets", cells = nmf.doublets)
plot(plotDimHighlight(suburd, clustering = "nmf.doublets", cluster = "TRUE",
    plot.title = paste0("NMF doublets: ", length(nmf.doublets), " cells"),
```

```
    point.size = 2, highlight.color = "blue"))
```




```
# Crop object to exclude doublets
suburd.cropped <- urdSubset(suburd, cells.keep = setdiff(colnames(suburd@logupx.data),
    nmf.doublets))
```

And then save the completed object for use downstream in building a tree using URD.

```
saveRDS(suburd.cropped, file = paste0(base.path, "/obj/URD_hypo_ND.rds"))
```

Hypothalamus: 2 - URD tree

Jeff Farrell

9/07/2019

Contents

Load data

```
suppressPackageStartupMessages(library(URD))
suppressPackageStartupMessages(library(Seurat))
```

```
base.path <- "~/urd-cluster-bushra/"

# Load procesed URD object
object <- readRDS(paste0(base.path, "obj/URD_hypo_ND.rds"))
```

Processed on the cluster

Most of the following steps were run on a computing cluster. These individual tissue subsets can be run on a modern, well-equipped laptop. The use of a computing cluster allows multiple parameter choices to be tried in parallel, and also allows further parallelization of the random walk procedure, speeding it up. Below, we should the commands that one would run on their laptop, and then generally load the pre-processed results from the cluster that were used in the paper. If you want to parallelize your own processing on a compute cluster, the scripts we used will be available at http://github.com/farrellja/URD/cluster/

Calculate diffusion map and pseudotime

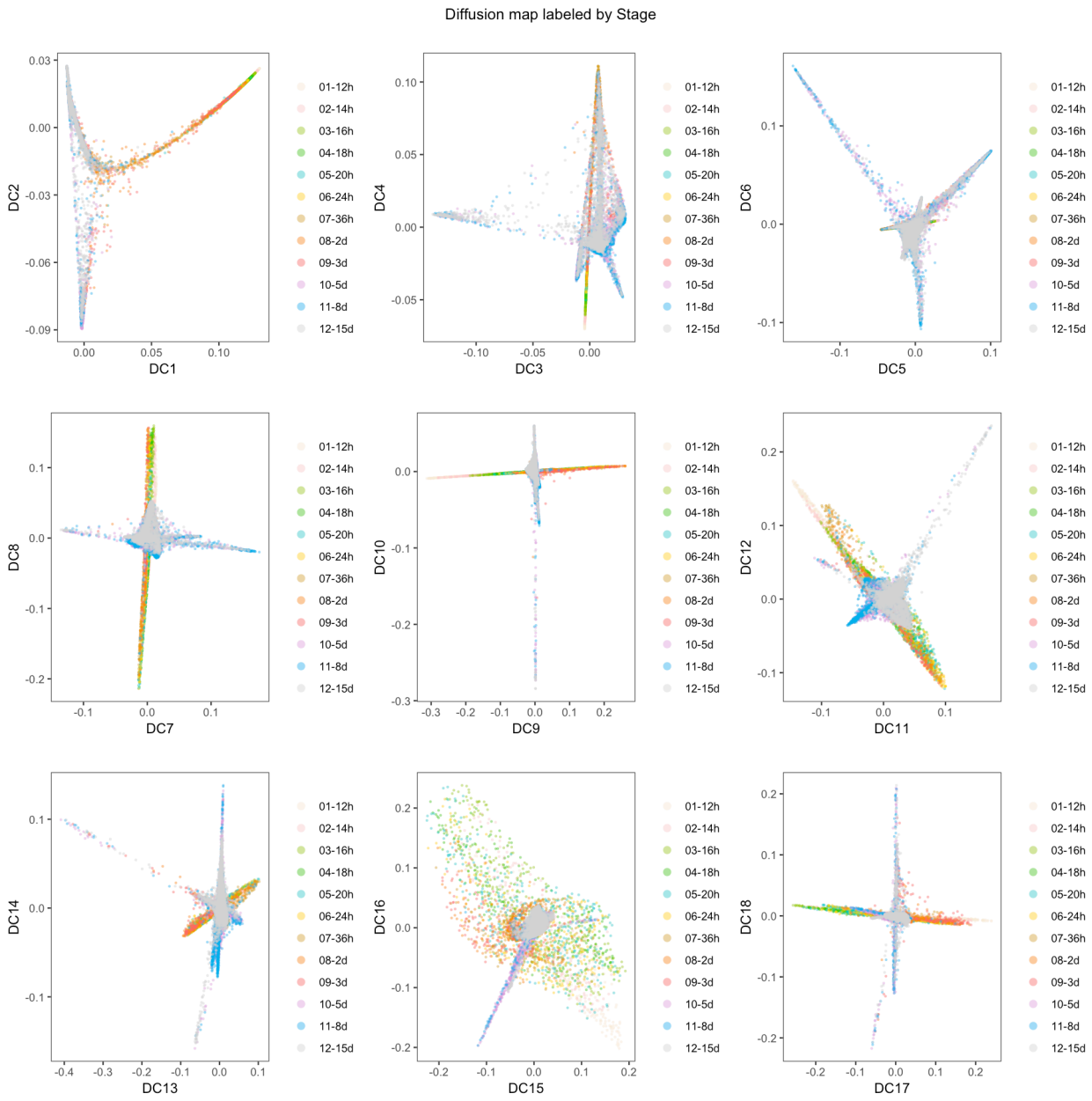
These two steps are run in the cluster script URD-DM-PT.R.

Calculate diffusion map

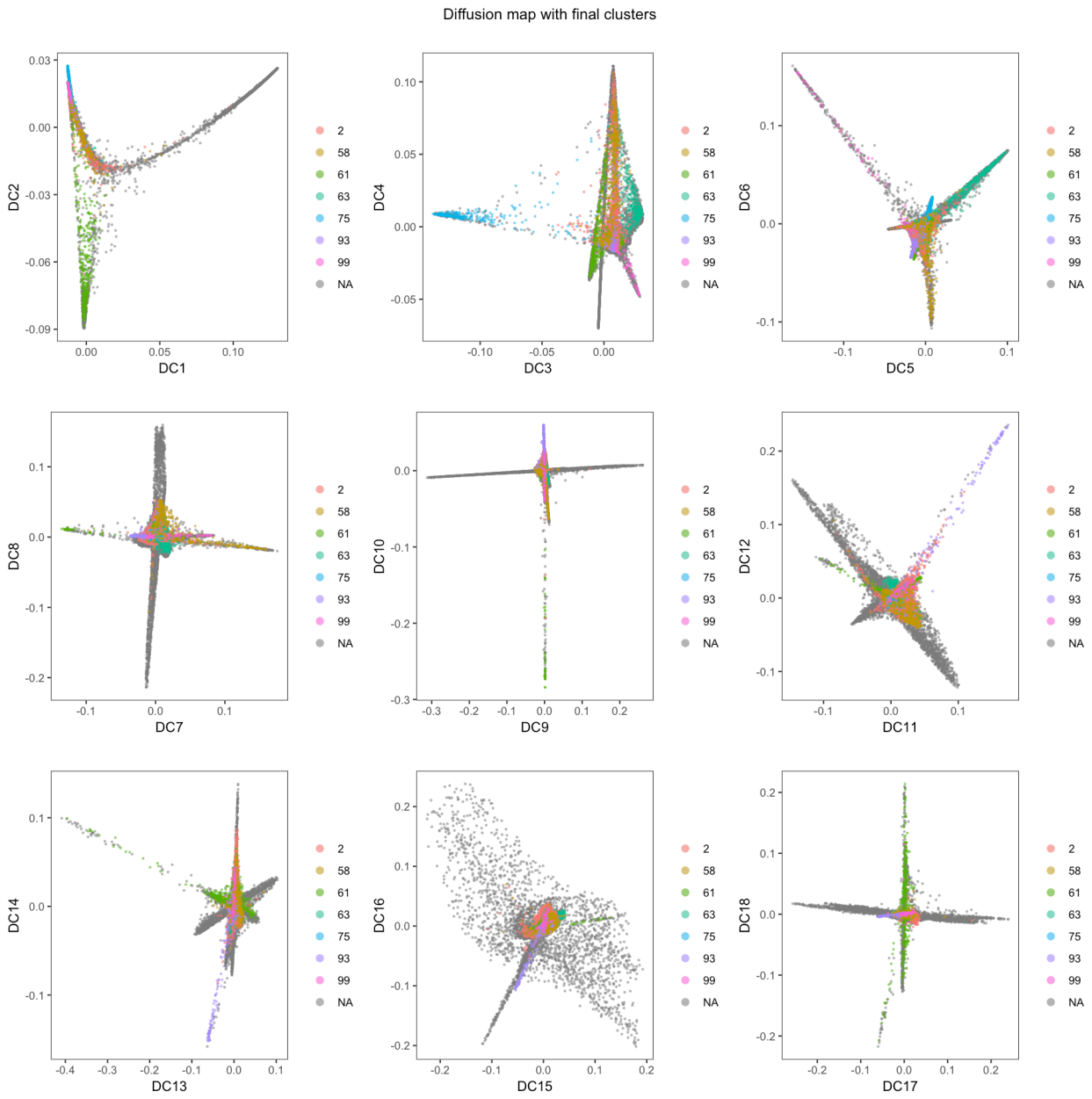
```
# To run locally: Calculate a diffusion map projection
object <- calcDM(object, knn = 100, sigma.use = 8)
```

```
# Or: Load a pre-computed diffusion map projection
dm <- readRDS(paste0(base.path, "dm/dm_hypoND_knn-100_sigma-8.rds"))
object <- importDM(object, dm)
```

```
# Plot diffusion maps
stage.colors <- c("antiquewhite", "#FFCCCC", "#99CC00", "#33CC00", "cyan3",
    "gold", "goldenrod", "darkorange", "indianred1", "plum", "deepskyblue2",
    "lightgrey")

# Plot by stage
plotDimArray(object = object, reduction.use = "dm", dims.to.plot = 1:18,
    label = "stage", plot.title = "", outer.title = "Diffusion map labeled by Stage",
    legend = T, alpha = 0.45, discrete.colors = stage.colors)
```



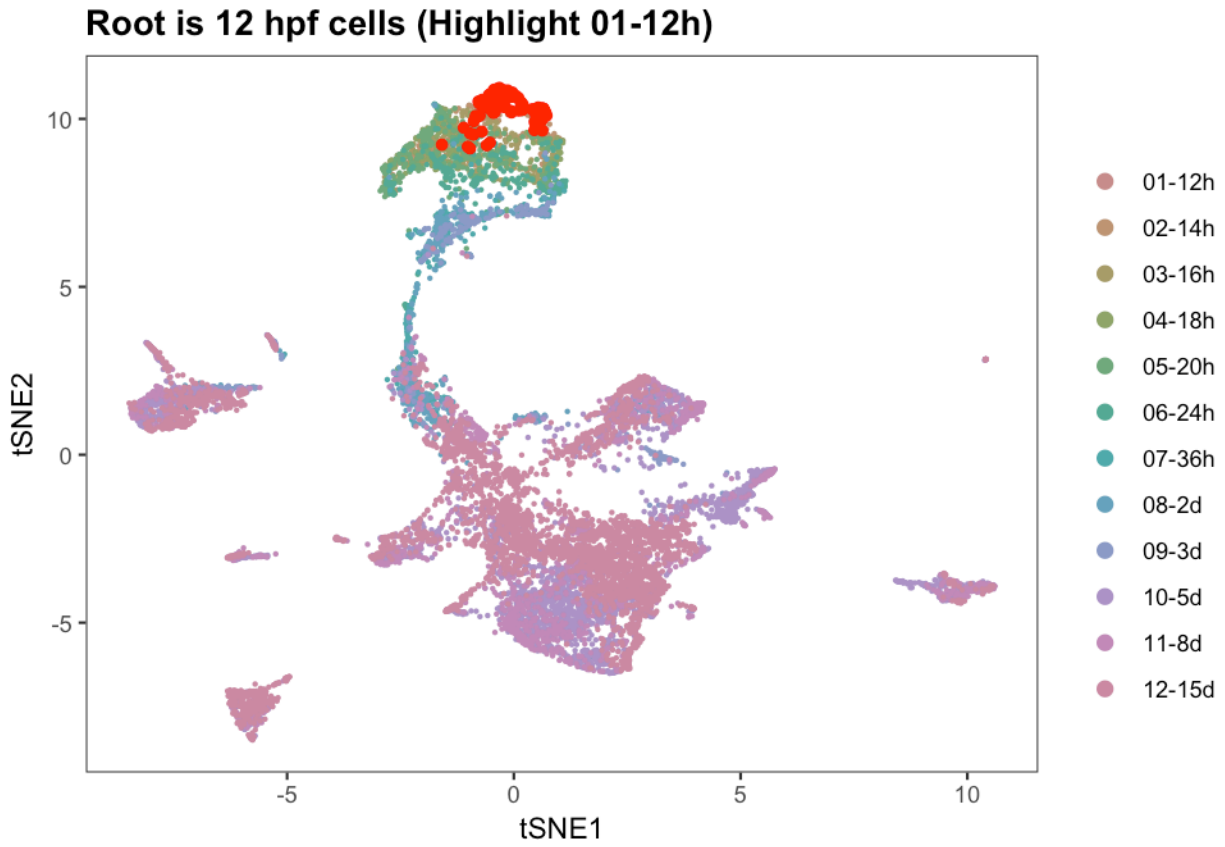

```
# Plot with final cell types labeled
object@group.ids$final.cluster <- NA
object@group.ids[cellsInCluster(object, "stage", "12-15d"), "final.cluster"] <- object@group.ids[cellsInCluster(o
    "stage", "12-15d"), "res.5"]
plotDimArray(object = object, reduction.use = "dm", dims.to.plot = 1:18,
    label = "final.cluster", plot.title = "", outer.title = "Diffusion map with final clusters",
    legend = T, alpha = 0.6)
```



Calculate pseudotime

URD requires a starting point or 'root' for determining pseudotime. Here, we used all cells from the first timepoint (i.e. 12 hpf) as the root.

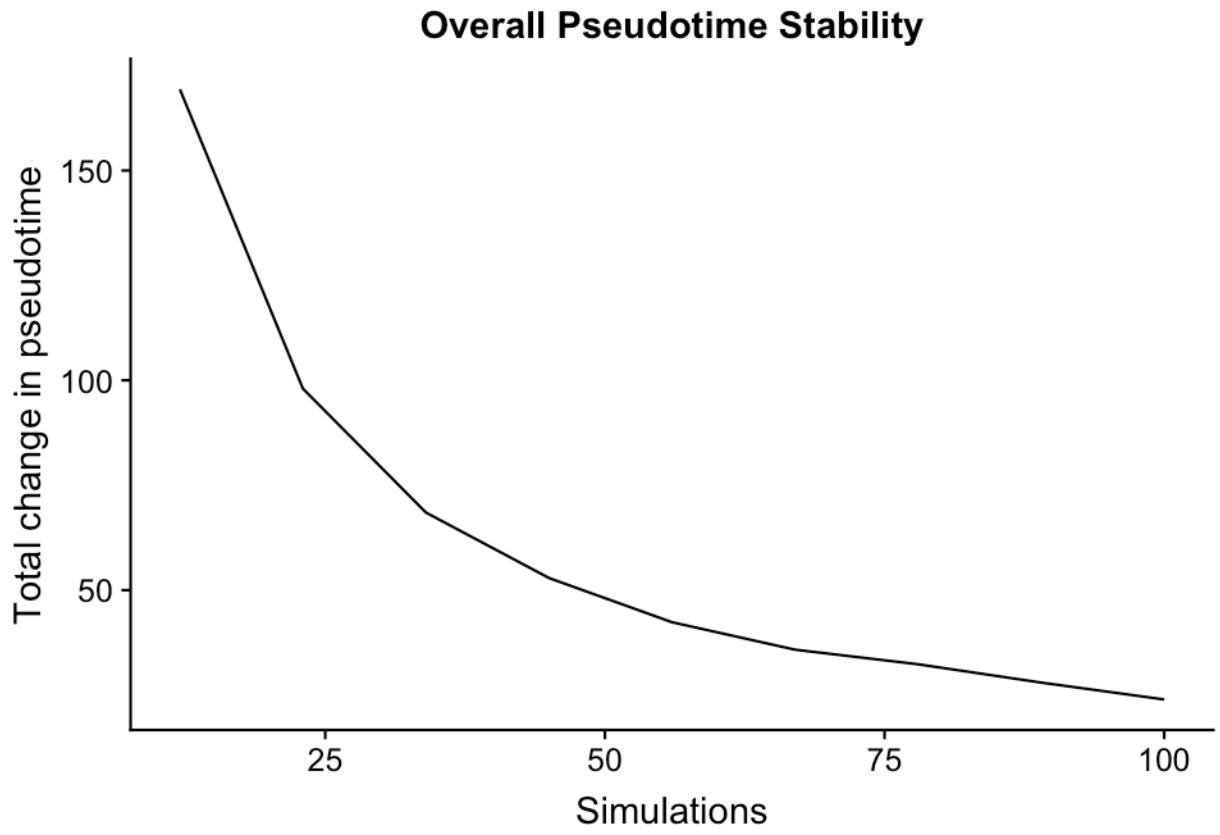
```
# Here, we used all cells from the first timepoint (i.e. 12 hours) as
# the root.
root.cells <- cellsInCluster(object, "stage", "01-12h")
plotDimHighlight(object, "stage", "01-12h", plot.title = "Root is 12 hpf cells")
```



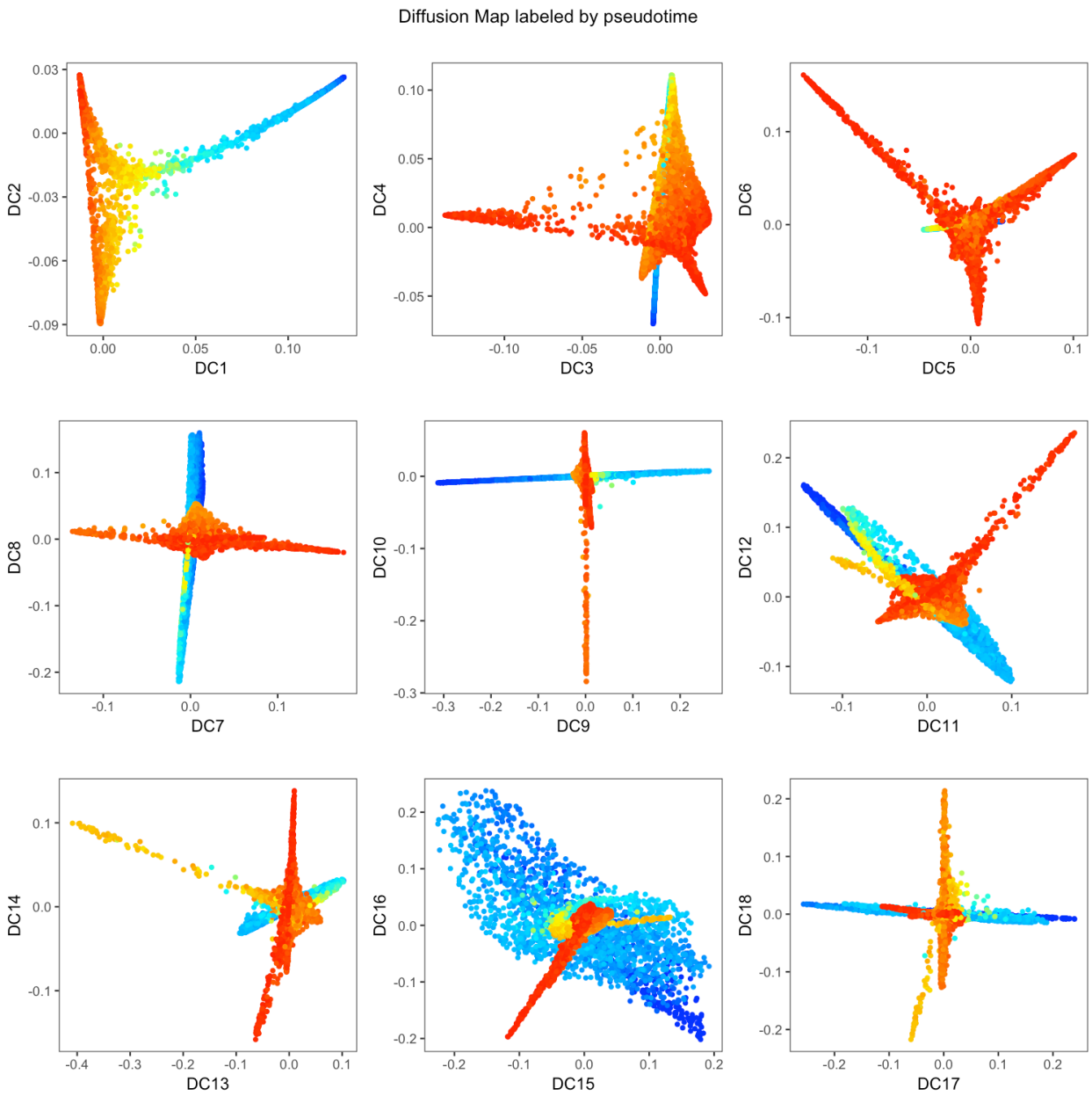

```
# To run locally: Run graph-search simulations to determine pseudotime
flood.result <- floodPseudotime(object, root.cells = root.cells, n = 100,
    minimum.cells.flooded = 2, verbose = T)
```

```
# Or load a pre-computed graph-search simulation result
flood.result <- readRDS(paste0(base.path, "flood/flood_hypoND_knn-100_sigma-8.rds"))
```

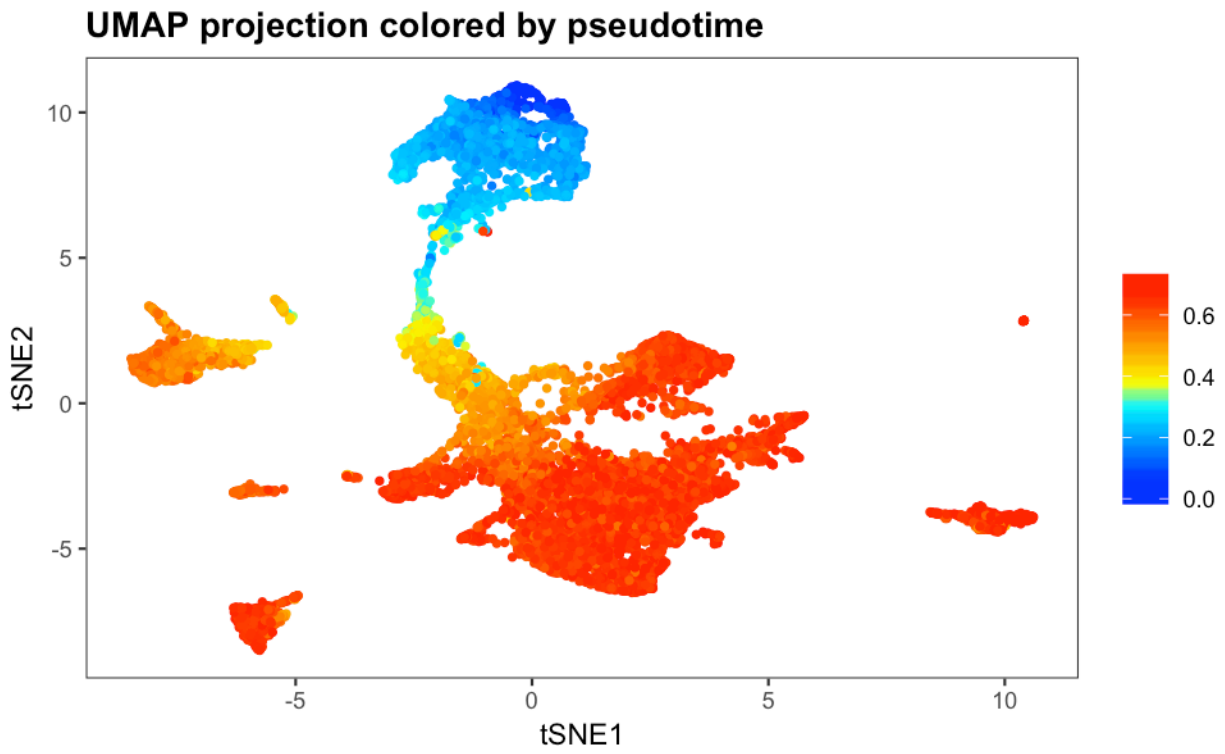
```
# Process the graph-search simulations to determine the pseudotime of
# each cell
object <- floodPseudotimeProcess(object, flood.result, floods.name = "pseudotime",
    max.frac.NA = 0.4, pseudotime.fun = mean, stability.div = 10)

# If enough simulations have been run, then as additional simulations
# are added, the overall change in pseudotime of cells should reach an
# asymptote. If it does not, then floodPseudotime should be run with a
# higher n.
pseudotimePlotStabilityOverall(object)
```



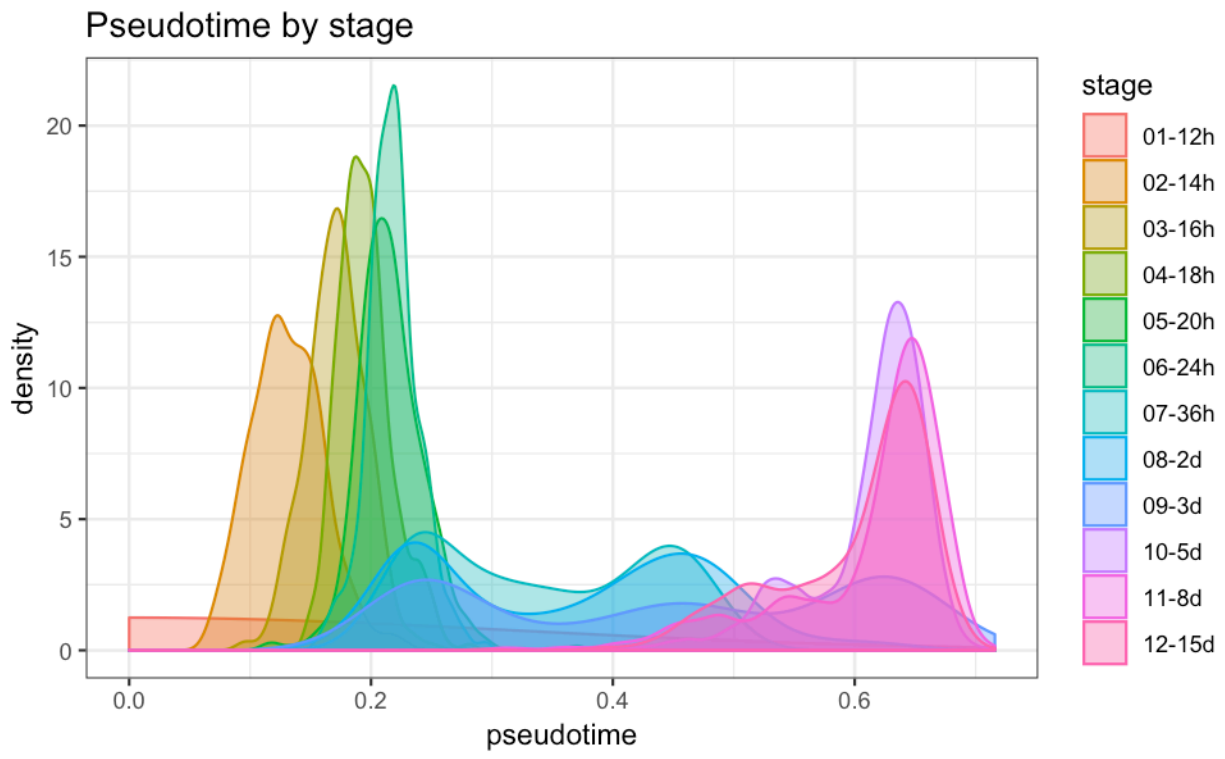

```
plotDimArray(object = object, reduction.use = "dm", dims.to.plot = 1:18,
    label = "pseudotime", plot.title = "", outer.title = "Diffusion Map labeled by pseudotime",
    legend = F, alpha = 0.4)
```




```
plotDim(object, "pseudotime", plot.title = "UMAP projection colored by pseudotime")
```



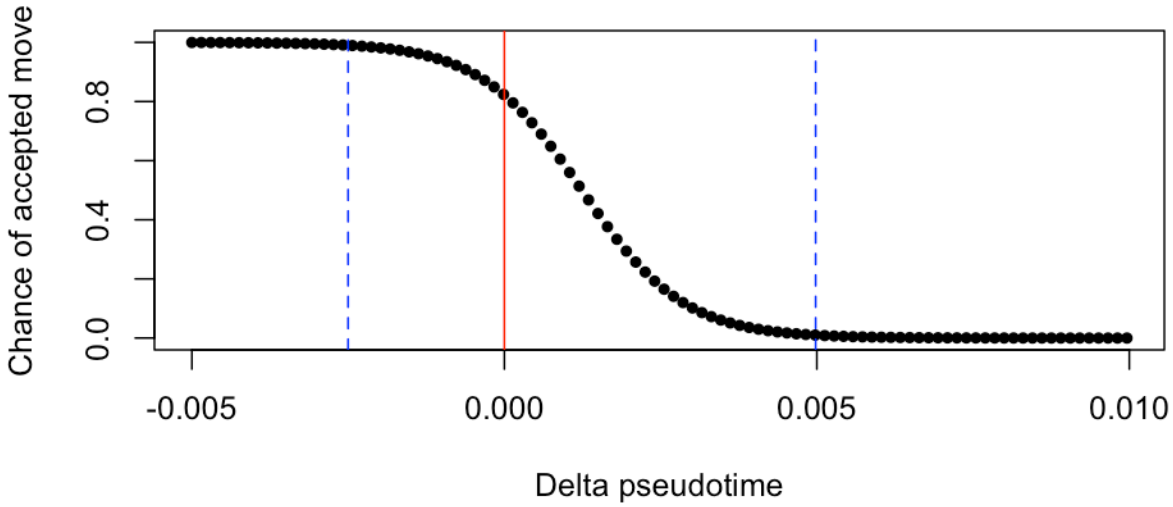

```
plotDists(object, "pseudotime", "stage", plot.title = "Pseudotime by stage")
```



Calculate biased transition matrix

In order to perform biased random walks, we must first bias the transition matrix to ensure that walks proceed towards the root and do not turn into other differentiated cell types. This is performed in the cluster script URD-TM.R.

```
# Calculate parameters for biasing the transition matrix.
diffusion.logistic <- pseudotimeDetermineLogistic(object, "pseudotime",
    optimal.cells.forward = 40, max.cells.back = 80, pseudotime.direction = "<",
    do.plot = T, print.values = T)
```




```
## [1] "Mean pseudotime back (~80 cells) 0.00498088811519173"
## [1] "Chance of accepted move to equal pseudotime is 0.821561374686937"
## [1] "Mean pseudotime forward (~40 cells) -0.00250030667341253"
```

```
# Calculate the biased matrix.
biased.tm <- pseudotimeWeightTransitionMatrix(object, pseudotime = "pseudotime",
    logistic.params = diffusion.logistic, pseudotime.direction = "<")
```

Perform biased random walks

Then, we perform biased walks starting from each tip. Visited cells are inferred to lie along the trajectory that connects the root to each cell type. This is performed in the cluster script URD-Walk.R.

Determine tips

We used clusters from 15 dpf as the tips for performing biased random walks. Here we define the cells belonging to each of those clusters.

```
# All clusters at 15 days
clusters.15day <- unique(object@group.ids[grep("15d", object@group.ids$stage),
    "res.5"])
# All cells at 15 days
cells.15day <- rownames(object@group.ids)[grep("15d", object@group.ids$stage)]
# Cell lists of each cluster at 15dpf
cells.15dpf.clusters <- lapply(clusters.15day, function(clust) intersect(cells.15day,
    cellsInCluster(object, "res.5", clust)))
names(cells.15dpf.clusters) <- paste0("15d-", clusters.15day)
```

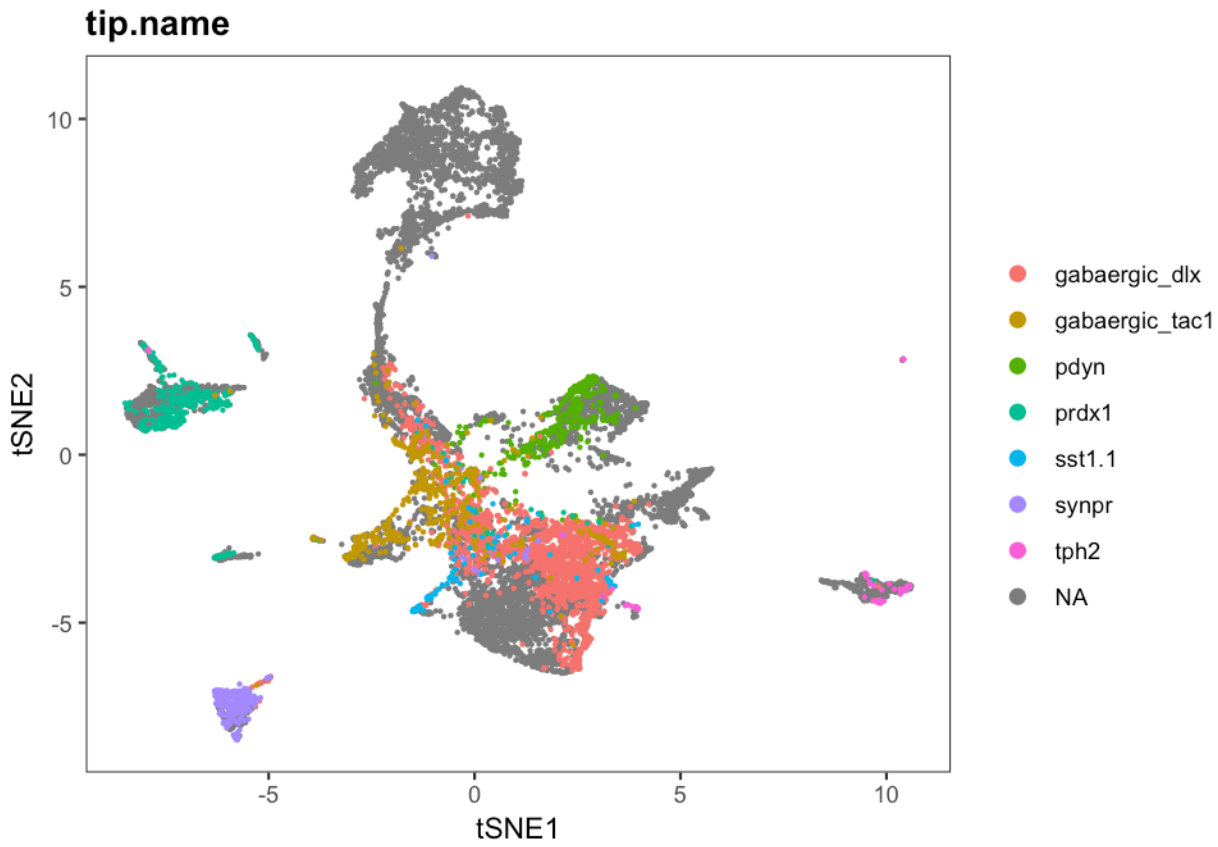
We also load a .csv file that contains information about the tips. It has four columns:

- id: Cluster ID for the tip
- use: Whether this cluster should be used when building the tree
- name: The name for this tip, which will be used on 2D plots
- short.name: The 'short' name for this tip, which would be used on 3D plots (though we did not use that feature in this study).

```
# Load CSV
tip.names <- read.csv(paste0(base.path, "tips/tip_names_hypoND.csv"), header = F,
    stringsAsFactors = F, colClasses = c("character", "logical", "character",
        "character"))

# Name columns and rows
names(tip.names) <- c("id", "use", "name", "short.name")
rownames(tip.names) <- gsub("_", "-", tip.names$id)

# Sort alphabetically
tip.names <- tip.names[order(rownames(tip.names)), ]
```

These are the tips that were considered during the construction of the retina URD tree (some were excluded during tree construction later in the `buildTree` command).

```
# Define a 'tips' clustering
object@group.ids$tip <- NA
object@group.ids$tip.id <- NA
object@group.ids$tip.name <- NA

# If the tip will be used in the tree, define its cells in the
# clustering
for (i in 1:nrow(tip.names)) {
    tip.cells <- cells.15dpf.clusters[[rownames(tip.names)[i]]]
    object@group.ids[tip.cells, "tip"] <- as.character(i)
    object@group.ids[tip.cells, "tip.id"] <- rownames(tip.names)[i]
    object@group.ids[tip.cells, "tip.name"] <- as.character(tip.names[i,
        "name"])
}

# Plot the tips
plotDim(object, "tip.name")
```



Perform the biased random walks

Biased random walks then need to be run starting from each tip. This can be performed on a laptop, but is an ideal candidate for parallelization on a cluster. (The walks from each tip can be run as a separate job.)

```
## IF RUNNING LOCALLY

# Loop through each cluster
walks <- lapply(rownames(tip.names), function(c) {
    # Exclude any tip cells that for whatever reason didn't end up in the
    # biased TM (e.g. maybe not assigned a pseudotime).
    tip.cells <- intersect(cells.15dpf.clusters[[c]], rownames(biased.tm))
    # Perform the random walk simulation
    this.walk <- simulateRandomWalk(start.cells = tip.cells, transition.matrix = biased.tm,
        end.cells = root.cells, n = 50000, end.visits = 1, verbose.freq = 1000,
        max.steps = 5000)
    return(this.walk)
})
names(walks) <- rownames(tip.names)

# Alternatively, this loop is automated by the function
# simulateRandomWalksFromTips
```

Alternatively, a set of pre-calculated walks can be loaded. Since the walks are a simulation (and

therefore not deterministic), this is particularly crucial for reproducing results.

```
## IF LOADING PRE-CALCULATED WALKS

# Get list of files in the walks directory
walks.files <- list.files(paste0(base.path, "/walks/hypoND/"), pattern = ".rds")

# Load the walks previously performed for each cluster
walks <- lapply(rownames(tip.names), function(c) {
    walk.file <- grep(pattern = paste0("_tip-", c, "_"), x = walks.files,
        value = T)[1]
    return(readRDS(paste0(base.path, "/walks/hypoND/", walk.file)))
})
names(walks) <- rownames(tip.names)
```

Process the random walks

The walks are then converted to visitation frequency by importing them into the URD object.

```
for (i in 1:nrow(tip.names)) {
    # Load the individual walk visitation frequencies into the object
    object <- processRandomWalks(object, walks = walks[[i]], walks.name = i,
        n.subsample = 1, verbose = F)
}
```

Build the URD tree

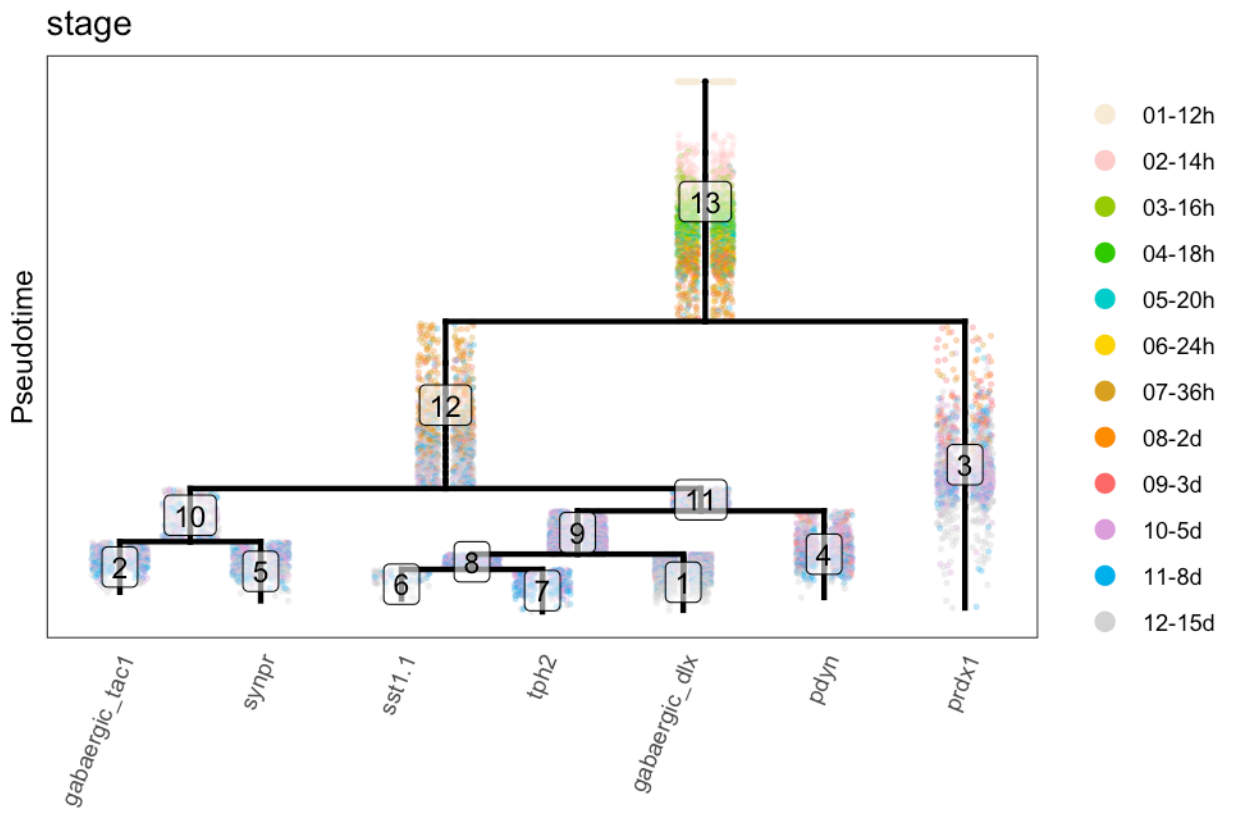
Then, a branching tree is constructed, by joining trajectories in an agglomerative fashion when cells are highly visited by walks from multiple tips. The following steps were performed in the cluster script URD-Tree.R.

```
# Tree building is destructive, so create a copy of the object
object.tree <- object

# Load tip cells
object.tree <- loadTipCells(object.tree, "tip")

# Determine tips to use
tips.to.use <- which(tip.names$use)
```

```
# Build the tree
object.tree <- buildTree(object.tree, pseudotime = "pseudotime", divergence.method = "ks",
    cells.per.pseudotime.bin = 40, bins.per.pseudotime.window = 5, save.all.breakpoint.info = T,
    p.thresh = 1e-04, verbose = F, tips.use = as.character(tips.to.use))
```

```
# Name the tips of the tree
object.tree <- nameSegments(object.tree, segments = tips.to.use, segment.names = as.character(tip.names[tips.to.u
    "name"], short.names = as.character(tip.names[tips.to.use, "short.name"])))
```

```
plotTree(object.tree, "stage", discrete.colors = stage.colors, label.segments = T)
```



Save the URD tree

The tree is then saved for use in downstream analysis, and can easily be loaded for further perusal.

```
saveRDS(object.tree, file = paste0(base.path, "tree/URD-Tree-Hypo.rds"))
```

Hypothalamus: 3 - URD Cascades and Figures

Jeff Farrell

10/08/2019, updated 07/30/2020

Contents

Load data

```
# Load URD
library(URD)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: Matrix
```

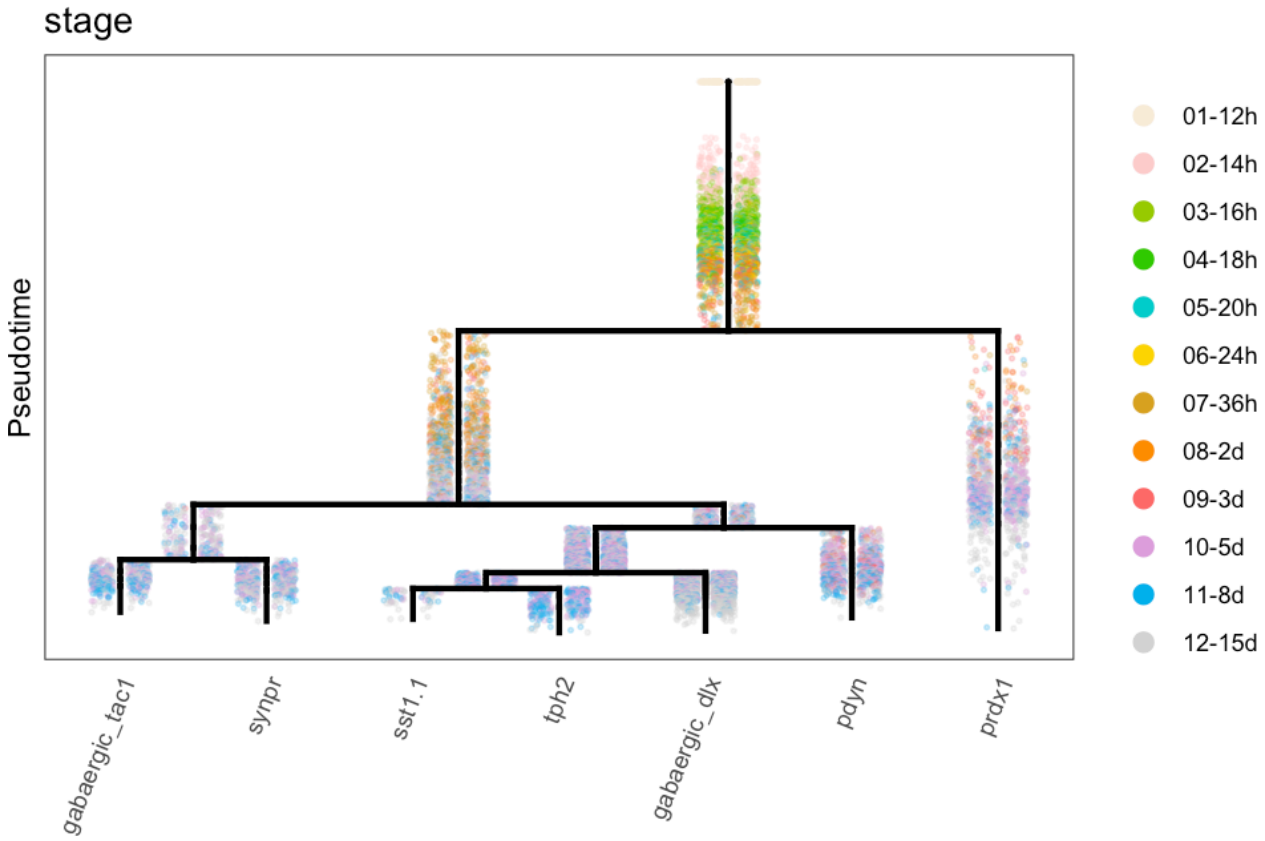
```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo
```

```
# Basic location
base.path <- "~/Documents/R sessions/urd-cluster-bushra/"

# Load completed hypothalamus tree object
obj.path <- paste0(base.path, "tree/hypoND/tree-hypoND_knn-100_sigma-8_40F-80B_NO-_ks_0001.rds")
obj <- readRDS(obj.path)
```

Plot gene expression on the tree

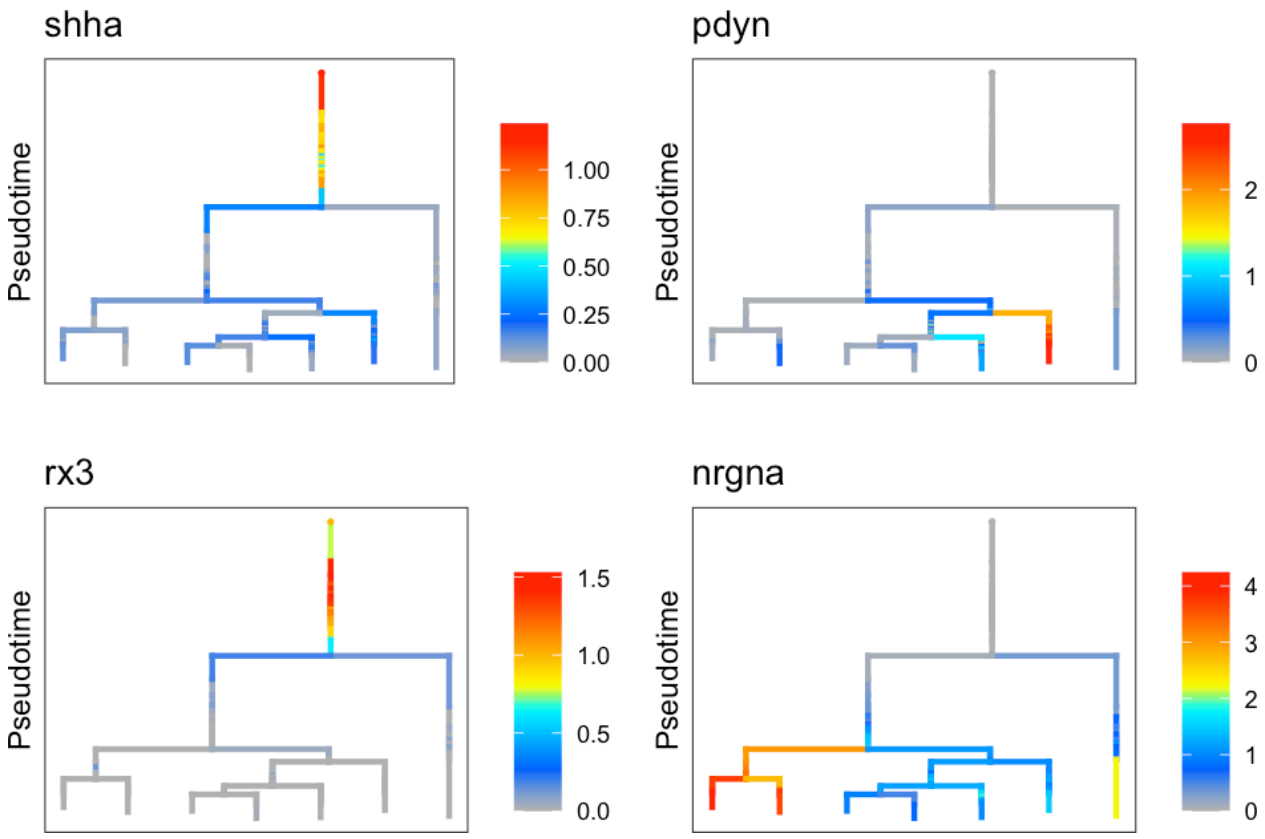
Plot tree by stage

```
stage.colors <- c("antiquewhite", "#FFCCCC", "#99CC00", "#33CC00", "cyan3",
    "gold", "goldenrod", "darkorange", "indianred1", "plum", "deepskyblue2",
    "lightgrey")

plotTree(obj, "stage", label.type = "group", discrete.colors = stage.colors)
```

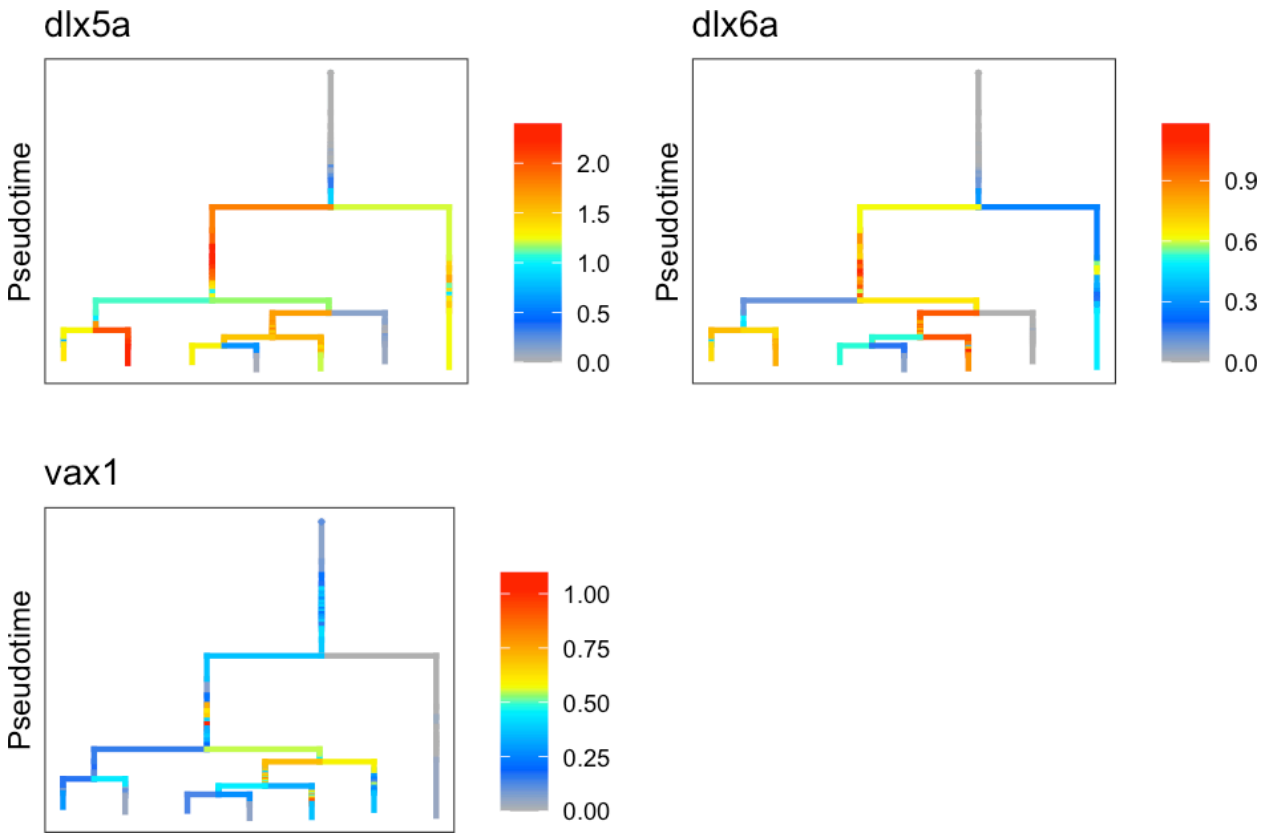


Plot tree with gene expression: main figures

```
gridExtra::grid.arrange(grobs = lapply(c("shha", "pdyn", "rx3", "nrgna"),
    plotTree, object = obj, label.x = F, plot.cells = F), ncol = 2)
```

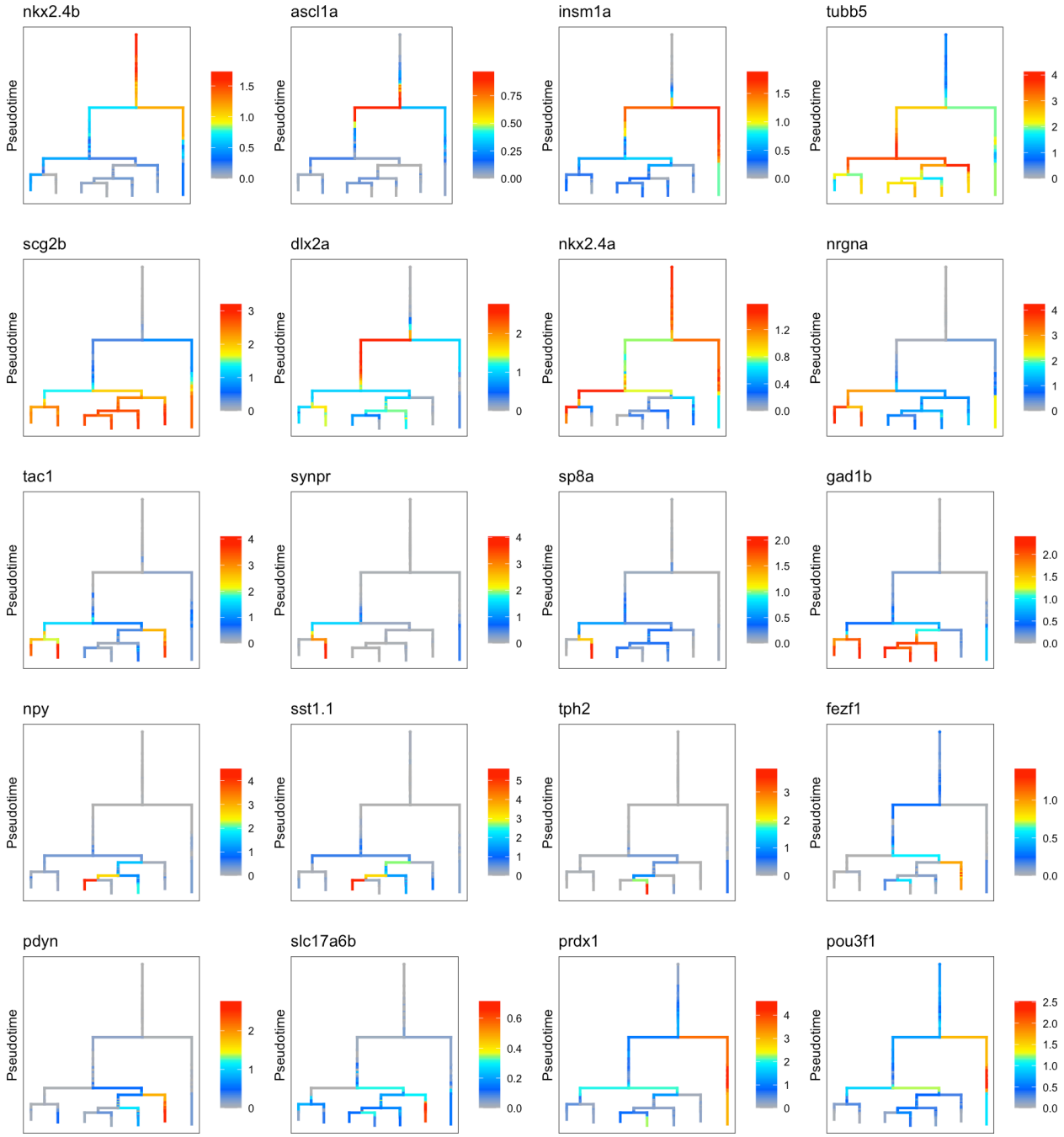



```
gridExtra::grid.arrange(grobs = lapply(c("dlx5a", "dlx6a", "vax1"), plotTree,
    object = obj, label.x = F, plot.cells = F), ncol = 2)
```



Plot tree with gene expression: supplemental figures

```
gridExtra::grid.arrange(grobs = lapply(c("nkx2.4b", "ascl1a", "insm1a",
    "tubb5", "scg2b", "dlx2a", "nkx2.4a", "nrgna", "tac1", "synpr", "sp8a",
    "gad1b", "npy", "sst1.1", "tph2", "fezf1", "pdyn", "slc17a6b", "prdx1",
    "pou3f1"), plotTree, object = obj, label.x = F, plot.cells = F), ncol = 4)
```



Determine genes enriched in trajectories to particular cell types

Comparison between major cell types

We took each major group ("clade") of branches from the end of the tree as a single entity (i.e. prdx1+ neurons, pdyn+ neurons, GABAergic dlx+ neurons, nrgna+ neurons) and compared them against each other to look for differentially expressed genes.

```
# Get the parent segment of each clade to consider as a group
combined.tips <- c("3", "4", "9", "10")
```

```r
# Get the cells in that segment and all child segments
cells.combined.tips <- lapply(combined.tips, function(t) whichCells(obj,
    label = "segment", value = segChildrenAll(obj, t, include.self = T)))
names(cells.combined.tips) <- combined.tips

# Loop through each of these clades and look for differentially
# expressed genes
combined.markers <- lapply(combined.tips, function(tip) {
    # Find all of the other clades
    opposing.tips <- setdiff(combined.tips, tip)
    # Perform pairwise comparisons to each other clade
    m.o <- lapply(opposing.tips, function(tip.opposing) {
        # message(paste0(Sys.time(), ': Comparing tip ', tip, ' to ',
        # tip.opposing, '.')) Find differentially expressed genes between the
        # pair of clades
        ma <- markersAUCPR(object = obj, cells.1 = cells.combined.tips[[tip]],
            cells.2 = cells.combined.tips[[tip.opposing]], effect.size = 0.5,
            auc.factor = 1.1)
        # In order to facilitate combining all of the results later, add
        # columns about which two clades were compared and also a duplicate
        # entry of the name of each gene that's recovered.
        if (nrow(ma) > 0) {
            ma$gene <- rownames(ma)
            ma$tip1 <- tip
            ma$tip2 <- tip.opposing
        }
        return(ma)
    })
    names(m.o) <- opposing.tips
    return(m.o)
})
names(combined.markers) <- combined.tips

# Require that genes are markers against at least 2 other clades
combined.markers.beatmult <- lapply(combined.markers, function(m) {
    names(which(table(unlist(lapply(m, rownames))) >= 2))
})

# Since genes might be a marker in a comparison to several other
# clades, combine the results into a single table, where each gene is
# listed only once with the info from the pairwise comparison where it
# had the strongest differential expression.
combined.markers.best <- lapply(1:length(combined.markers.beatmult), function(i) {
    cm <- do.call("rbind", combined.markers[[i]])
    cm <- cm[cm$gene %in% combined.markers.beatmult[[i]], ]
    cmb <- do.call("rbind", lapply(combined.markers.beatmult[[i]], function(g) {
        cmr <- cm[cm$gene == g, ]
        return(cmr[which.max(cmr$AUCPR.ratio), ])
    }))
    rownames(cmb) <- cmb$gene
    if (!is.null(cmb)) {
        cmb <- cmb[order(cmb$AUCPR.ratio, decreasing = T), ]
        cmb$exp.global <- apply(obj@logupx.data[rownames(cmb), unlist(obj@tree$cells.in.segment)],
```

```
            1, mean.of.logs)
        cmb$exp.global.fc <- cmb$nTrans_1 - cmb$exp.global
    }
    return(cmb)
})
names(combined.markers.best) <- combined.tips
```

AUCPR along tree

We also used the AUCPRTestAlongTree function to ask for genes that are differential markers of a lineage using URD's tree structure. This makes a comparison at each branchpoint from a particular cell type up to the root.

```
# Get all of the tips from the tree
tips.in.tree <- as.character(obj@tree$tips)

# Tree segments to use as root
roots <- rep("12", length(tips.in.tree))
names(roots) <- tips.in.tree
roots["3"] <- "13"

# Define parameters to use for calculation Used more permissive values
# in the sst1.1+ / tph2+ / gabaergic dlx+ neuronal comparisons due to
# the small number of cells in these populations
auc.use <- rep(1.2, length(tips.in.tree))
names(auc.use) <- tips.in.tree
auc.use[c("1", "6", "7")] <- 1.15
log.effect.use <- rep(0.8, length(tips.in.tree))
names(log.effect.use) <- tips.in.tree
log.effect.use[c("1", "6", "7")] <- 0.6

# Perform a loop of tests with each tip.
markers <- lapply(tips.in.tree, function(t) {
    this.root <- roots[t]
    this.auc <- auc.use[t]
    this.log <- log.effect.use[t]
    # message(paste0(Sys.time(), ': Starting tip ', t, ' and root ',
    # this.root, ' with params ', this.auc, ' AUC and ', this.log, ' effect
    # size.'))
    these.markers <- aucprTestAlongTree(obj, pseudotime = "pseudotime",
        tips = as.character(t), genes.use = NULL, must.beat.sibs = 0.6,
        report.debug = F, root = this.root, auc.factor = this.auc, log.effect.size = this.log)
    these.markers$gene <- rownames(these.markers)
    these.markers$tip <- t
    return(these.markers)
})
names(markers) <- tips.in.tree
```

Markers of the prdx1- neuron clade

```
# Calculate from segment 12 against segment 3 specifically
nonprdx.markers <- markersAUCPR(obj, clust.1 = "12", clust.2 = "3", clustering = "segment",
```

```
    effect.size = 0.8, auc.factor = 1.2)
# Also look at segment 12 vs. rest of the hypothalamus with lower
# thresholds
nonprdx.markers.global <- markersAUCPR(obj, clust.1 = "12", clust.2 = as.character(c(1:11,
    13)), clustering = "segment", effect.size = 0.4, auc.factor = 1.1)
```

```
## Warning in names(genes.data)[4:7] <- paste(c("posFrac", "posFrac", "nTrans", :
## number of items to replace is not a multiple of replacement length
```

Functions for curating differential expression results

We further curated those differentially expressed genes using the following functions:

threshold.tree.markers

Function to threshold markers from a markersAUCPRAlongTree test with additional criteria

- **markers**: list of results from markersAUCPRAlongTree tests
- **tip**: which tip (or element of the list to pursue)
- **global.fc**: fold.change that gene must have along the trajectory pursued vs. rest of the data
- **aucpr.ratio.all**: classifier score that gene must exhibit along trajectory test vs. rest of the data
- **branch.fc**: fold.change that gene must have (in best case) vs. the opposing branch at any branchpoint along the trajectory.
- Returns `markers` with only a subset of rows retained.

```
threshold.tree.markers <- function(markers, tip, global.fc = 0.1, branch.fc = 0.4,
    aucpr.ratio.all = 1.03) {
    m <- markers[[tip]]
    # First off -- lose global FC < x
    bye.globalfc <- rownames(m)[m$expfc.all < global.fc]
    # Second -- get rid of branch FC < x
    bye.branchfc <- rownames(m)[m$expfc.maxBranch < branch.fc]
    # Third -- get rid of stuff essentially worse than random
    # classification on global level
    bye.badglobalaucpr <- rownames(m)[m$AUCPR.ratio.all < aucpr.ratio.all]
    bye.all <- unique(c(bye.globalfc, bye.branchfc, bye.badglobalaucpr))
    m.return <- m[setdiff(rownames(m), bye.all), ]
    return(m.return)
}
```

divide.branches

Function to compare genes between two branches. Use this on a compiled list of markers to do a final selection of genes that are specific to one branch or another or markers of both (i.e. when making photoreceptor heatmap, use to divide into photoreceptor, cone, and rod markers)

- **object**: An URD object
- **genes**: (Character vector) Genes to test
- **clust.1**: (Character) Cluster 1
- **clust.2**: (Character) Cluster 2
- **clustering**: (Character) Clustering to pull from
- **exp.fc**: (Numeric) Minimum expression fold-change between branches to consider different

- **exp.thresh**: (Numeric) Minimum fraction of cells in order to consider gene expressed in a branch
- **exp.diff**: (Numeric) Minimum difference in fraction of cells expressing to consider gene differential
- Returns list of gene names ("specific.1" = specific to clust.1, "specific.2" = specific to clust.2, "markers" = all genes tested)

```
divide.branches <- function(object, genes, clust.1, clust.2, clustering = "segment",
    exp.fc = 0.4, exp.thresh = 0.1, exp.diff = 0.1) {
    # Double check which markers are unique to one or the other population
    mcomp <- markersAUCPR(object, clust.1 = clust.1, clust.2 = clust.2,
        clustering = clustering, effect.size = -Inf, auc.factor = 0, genes.use = genes,
        frac.min.diff = 0, frac.must.express = 0)
    specific.b <- rownames(mcomp)[abs(mcomp$exp.fc) > exp.fc & mcomp[,
        4] < exp.thresh & mcomp[, 5] > pmin((mcomp[, 4] + exp.diff), 1)]
    specific.a <- rownames(mcomp)[abs(mcomp$exp.fc) > exp.fc & mcomp[,
        5] < exp.thresh & mcomp[, 4] > pmin((mcomp[, 5] + exp.diff), 1)]
    r <- list(specific.a, specific.b, mcomp)
    names(r) <- c("specific.1", "specific.2", "markers")
    return(r)
}
```

divide.branches.triple

Function to compare genes between three branches. Use this on a compiled list of markers to do a final selection of genes that are specific to one branch or another or markers of both (i.e. when making gad2+ heatmap, use to divide into general, dlx+, sst+, and tph2+ markers).

- **object**: An URD object
- **genes**: (Character vector) Genes to test
- **clust.1**: (Character) Cluster 1
- **clust.2**: (Character) Cluster 2
- **clust.3**: (Character) Cluster 3
- **clustering**: (Character) Clustering to pull from
- **exp.fc**: (Numeric) Minimum expression fold-change between branches to consider different
- **exp.thresh**: (Numeric) Minimum fraction of cells in order to consider gene expressed in a branch
- **exp.diff**: (Numeric) Minimum difference in fraction of cells expressing to consider gene differential
- Returns list of gene names ("nonspecific" = genes not in specific.1/2/3, "specific.1" = specific to clust.1, "specific.2" = specific to clust.2, "specific.3" = specific to clust.3, each pairwise comparison, and "markers" = all genes tested)

```
divide.branches.triple <- function(object, genes, clust.1, clust.2, clust.3,
    clustering = "segment", exp.fc = 0.4, exp.thresh = 0.1, exp.diff = 0.1) {
    # Double check which markers are unique to one or the other population
    mcomp12 <- markersAUCPR(object, clust.1 = clust.1, clust.2 = clust.2,
        clustering = clustering, effect.size = -Inf, auc.factor = 0, genes.use = genes,
        frac.min.diff = 0, frac.must.express = 0)
    mcomp23 <- markersAUCPR(object, clust.1 = clust.2, clust.2 = clust.3,
        clustering = clustering, effect.size = -Inf, auc.factor = 0, genes.use = genes,
        frac.min.diff = 0, frac.must.express = 0)
    mcomp13 <- markersAUCPR(object, clust.1 = clust.1, clust.2 = clust.3,
        clustering = clustering, effect.size = -Inf, auc.factor = 0, genes.use = genes,
        frac.min.diff = 0, frac.must.express = 0)
```

```
    specific.1v2 <- rownames(mcomp12)[abs(mcomp12$exp.fc) > exp.fc & mcomp12[,
        5] < exp.thresh & mcomp12[, 4] > pmin((mcomp12[, 5] + exp.diff),
        1)]
    specific.2v1 <- rownames(mcomp12)[abs(mcomp12$exp.fc) > exp.fc & mcomp12[,
        4] < exp.thresh & mcomp12[, 5] > pmin((mcomp12[, 4] + exp.diff),
        1)]

    specific.2v3 <- rownames(mcomp23)[abs(mcomp23$exp.fc) > exp.fc & mcomp23[,
        5] < exp.thresh & mcomp23[, 4] > pmin((mcomp23[, 5] + exp.diff),
        1)]
    specific.3v2 <- rownames(mcomp23)[abs(mcomp23$exp.fc) > exp.fc & mcomp23[,
        4] < exp.thresh & mcomp23[, 5] > pmin((mcomp23[, 4] + exp.diff),
        1)]

    specific.1v3 <- rownames(mcomp13)[abs(mcomp13$exp.fc) > exp.fc & mcomp13[,
        5] < exp.thresh & mcomp13[, 4] > pmin((mcomp13[, 5] + exp.diff),
        1)]
    specific.3v1 <- rownames(mcomp13)[abs(mcomp13$exp.fc) > exp.fc & mcomp13[,
        4] < exp.thresh & mcomp13[, 5] > pmin((mcomp13[, 4] + exp.diff),
        1)]

    specific.1 <- unique(setdiff(c(specific.1v2, specific.1v3), c(specific.2v3,
        specific.3v2)))
    specific.2 <- unique(setdiff(c(specific.2v1, specific.2v3), c(specific.1v3,
        specific.3v1)))
    specific.3 <- unique(setdiff(c(specific.3v2, specific.3v1), c(specific.2v1,
        specific.1v2)))
    nonspecific <- setdiff(genes, c(specific.1, specific.2, specific.3))

    markers.comp <- list(mcomp12, mcomp13, mcomp23)
    names(markers.comp) <- c("1v2", "1v3", "2v3")

    r <- list(nonspecific, specific.1, specific.2, specific.3, specific.1v2,
        specific.1v3, specific.2v1, specific.2v3, specific.3v1, specific.3v2,
        markers.comp)
    names(r) <- c("nonspecific", "specific.1", "specific.2", "specific.3",
        "specific.1v2", "specific.1v3", "specific.2v1", "specific.2v3",
        "specific.3v1", "specific.3v2", "markers")
    return(r)
}
```

Functions for heatmap generation

These functions were used in the production of heatmaps:

Color scale

Generate color scale to use with heatmaps.

```
cols <- (scales::gradient_n_pal(RColorBrewer::brewer.pal(9, "YlOrRd")))(seq(0,
    1, length.out = 50))
```

determine.timing

Determines order to plot genes in heatmap. "Expression" is defined as 20% higher expression than the minimum observed value. "Peak" expression is defined as 50% higher expression than minimum observed value. The two longest stretches of "peak" expression are found, and then the later one is used. The onset time of the stretch of expression that contains that peak is also determined. Genes are then ordered by the pseudotime at which they enter "peak" expression, leave "peak" expression, start "expression", and leave "expression".

- **s**: result from `geneSmoothFit`
- **genes**: genes to order; default is all genes that were fit.
- Returns `s` but with an additional list entry (`$timing`) of the order to plot genes

```
determine.timing <- function(s, genes = rownames(s$mean.expression)) {
    s$timing <- as.data.frame(do.call("rbind", lapply(genes, function(g) {
        sv <- as.numeric(s$scaled.smooth[g, ])
        pt <- as.numeric(colnames(s$scaled.smooth))
        # Figure out baseline expression & threshold for finding peaks
        min.val <- max(min(sv), 0)
        peak.val <- ((1 - min.val)/2) + min.val
        exp.val <- ((1 - min.val)/5) + min.val
        # Run-length encoding of above/below the peak-threshold
        peak.rle <- rle(sv >= peak.val)
        peak.rle <- data.frame(lengths = peak.rle$lengths, values = peak.rle$values)
        peak.rle$end <- cumsum(peak.rle$lengths)
        peak.rle$start <- head(c(0, peak.rle$end) + 1, -1)
        # Run-length encoding of above/below the expressed-threshold
        exp.rle <- rle(sv >= exp.val)
        exp.rle <- data.frame(lengths = exp.rle$lengths, values = exp.rle$values)
        exp.rle$end <- cumsum(exp.rle$lengths)
        exp.rle$start <- head(c(0, exp.rle$end) + 1, -1)
        # Take top-two longest peak RLE & select later one.  Find stretches
        # that are above peak value
        peak <- which(peak.rle$values)
        # Order by length and take 1 or 2 longest ones
        peak <- peak[order(peak.rle[peak, "lengths"], decreasing = T)][1:min(2,
            length(peak))]
        # Order by start and take latest one.
        peak <- peak[order(peak.rle[peak, "start"], decreasing = T)][1]
        # Identify the actual peak value within that stretch
        peak <- which.max(sv[peak.rle[peak, "start"]:peak.rle[peak, "end"]]) +
            peak.rle[peak, "start"] - 1
        # Identify the start and stop of the expressed stretch that contains
        # the peak
        exp.start <- exp.rle[which(exp.rle$end >= peak & exp.rle$start <=
            peak), "start"]
        exp.end <- exp.rle[which(exp.rle$end >= peak & exp.rle$start <=
            peak), "end"]
        # Identify values of expression at start and stop
        smooth.start <- sv[exp.start]
        smooth.end <- sv[exp.end]
        # Convert to pseudotime?
        exp.start <- pt[exp.start]
        exp.end <- pt[exp.end]
        peak <- pt[peak]
```

```
        # Return a vector
        v <- c(exp.start, peak, exp.end, smooth.start, smooth.end)
        names(v) <- c("pt.start", "pt.peak", "pt.end", "exp.start", "exp.end")
        return(v)
    })))
    rownames(s$timing) <- genes

    # Decide on ordering of genes
    s$gene.order <- rownames(s$timing)[order(s$timing$pt.peak, s$timing$pt.start,
        s$timing$pt.end, s$timing$exp.end, decreasing = c(F, F, F, T),
        method = "radix")]

    return(s)
}
```

filter.heatmap.genes

Removes undesired (mitochondrial, ribosomal, tandem duplicated genes) from heatmaps for presentation purposes.

- genes: (Character vector) genes to check
- Returns `genes` with undesired genes removed.

```
filter.heatmap.genes <- function(genes) {
    mt.genes <- grep("^mt-", ignore.case = T, genes, value = T)
    many.genes <- grep("\\(1 of many\\)", ignore.case = T, genes, value = T)
    ribo.genes <- grep("^rpl|^rps", ignore.case = T, genes, value = T)
    cox.genes <- grep("^cox", ignore.case = T, genes, value = T)
    hsp.genes <- grep("^hsp", ignore.case = T, genes, value = T)
    return(setdiff(genes, c(mt.genes, many.genes, ribo.genes, cox.genes,
        hsp.genes)))
}
```

Heatmaps of gene cascades

Using the genes that were determined as differentially expressed along the way to particular cell types, we generated expression cascades and plotted them as heatmaps.

Pdyn+ neurons

Prepare cascade

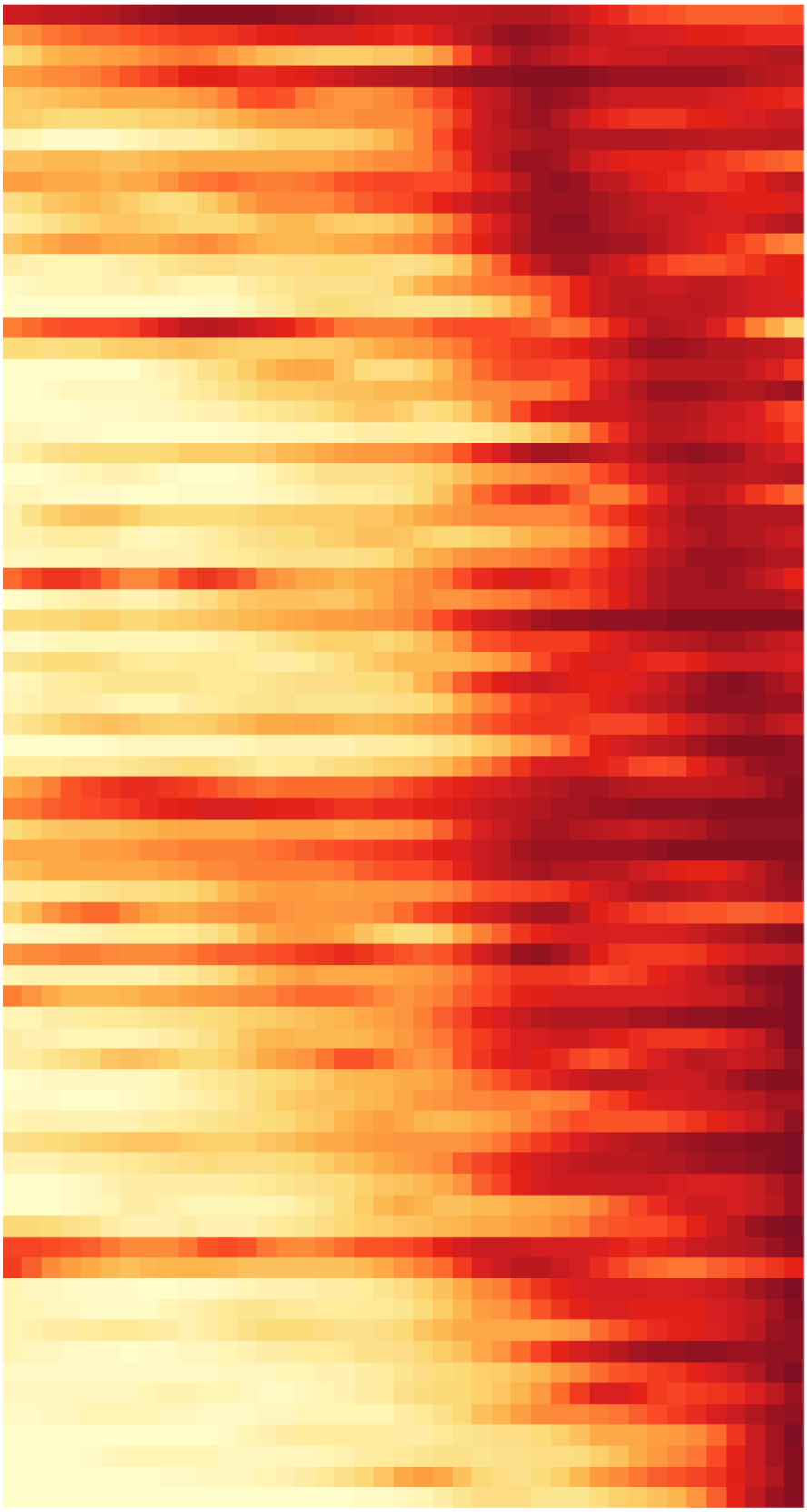
```
## Pdyn+ neurons: Seg 4

# Get markers from the two approaches
t <- combined.markers.best[["4"]]  # pdyn+ markers from above the combined clades
t <- t[t$exp.global.fc >= 0.8, ]  # limited to those with good global parameters
m <- threshold.tree.markers(markers, "4", global.fc = 0.6)  # Pdyn+ Cell markers from aucprTestAlongTree
pdyn.markers <- unique(c(rownames(t), rownames(m)))

# Just want to plot part of cells from upstream segment 12, which is
# very long. Going to use cells from segments 4, 11, and from segment
```

```
# 12 with pseudotime > 0.23
cells.plot <- unique(c(intersect(cellsInCluster(obj, "segment", "12"),
    whichCells(obj, "pseudotime", c(0.45, Inf))), cellsInCluster(obj, "segment",
    c("11", "4"))))

# Calculate spline curve
spline.plot <- geneSmoothFit(obj, pseudotime = "pseudotime", cells = cells.plot,
    genes = pdyn.markers, method = "spline", moving.window = 5, cells.per.window = 25,
    pseudotime.per.window = 0.005, spar = 0.5, verbose = F)

# Calculate gene expression timing for ordering rows
spline.plot <- determine.timing(s = spline.plot)
order.plot <- filter.heatmap.genes(spline.plot$gene.order)

# Output gene table
table.save <- data.frame(gene = order.plot, stringsAsFactors = F)
table.save$clade.AUCPR.ratio <- t[table.save$gene, "AUCPR.ratio"]
table.save$clade.exp.fc <- t[table.save$gene, "exp.fc"]
table.save$clade.exp.fc.global <- t[table.save$gene, "exp.global.fc"]
table.save$pdyn.AUCPR.ratio.all <- m[table.save$gene, "AUCPR.ratio.all"]
table.save$pdyn.AUCPR.ratio.maxBranch <- m[table.save$gene, "AUCPR.ratio.maxBranch"]
table.save$pdyn.exp.fc.all <- m[table.save$gene, "expfc.all"]
table.save$pdyn.exp.fc.best <- m[table.save$gene, "expfc.maxBranch"]
write.csv(table.save, quote = F, file = paste0(base.path, "/heatmaps/hypo-pdyn.csv"))
```

Generate heatmap: all genes

```
## Generate heatmap Make sure any values <0 in the spline curves get set
## to 0 so that the heatmap scale doesn't get messed up.
spline.plot$scaled.smooth[spline.plot$scaled.smooth < 0] <- 0
# Open a PDF and generate the heatmap pdf(paste0(base.path,
# '/heatmaps/hypo-pdyn.pdf'), width=6, height=10)
gplots::heatmap.2(x = as.matrix(spline.plot$scaled.smooth[order.plot, ]),
    Rowv = F, Colv = F, dendrogram = "none", col = cols, trace = "none",
    density.info = "none", key = F, cexCol = 0.8, cexRow = 0.6, margins = c(8,
        8), lwid = c(0.3, 4), lhei = c(0.3, 4), labCol = NA)
title(main = "pdyn+ Neurons")
```

pdyn+ Neurons



```
# dev.off()
```

Prdx1+ neurons vs. other neurons

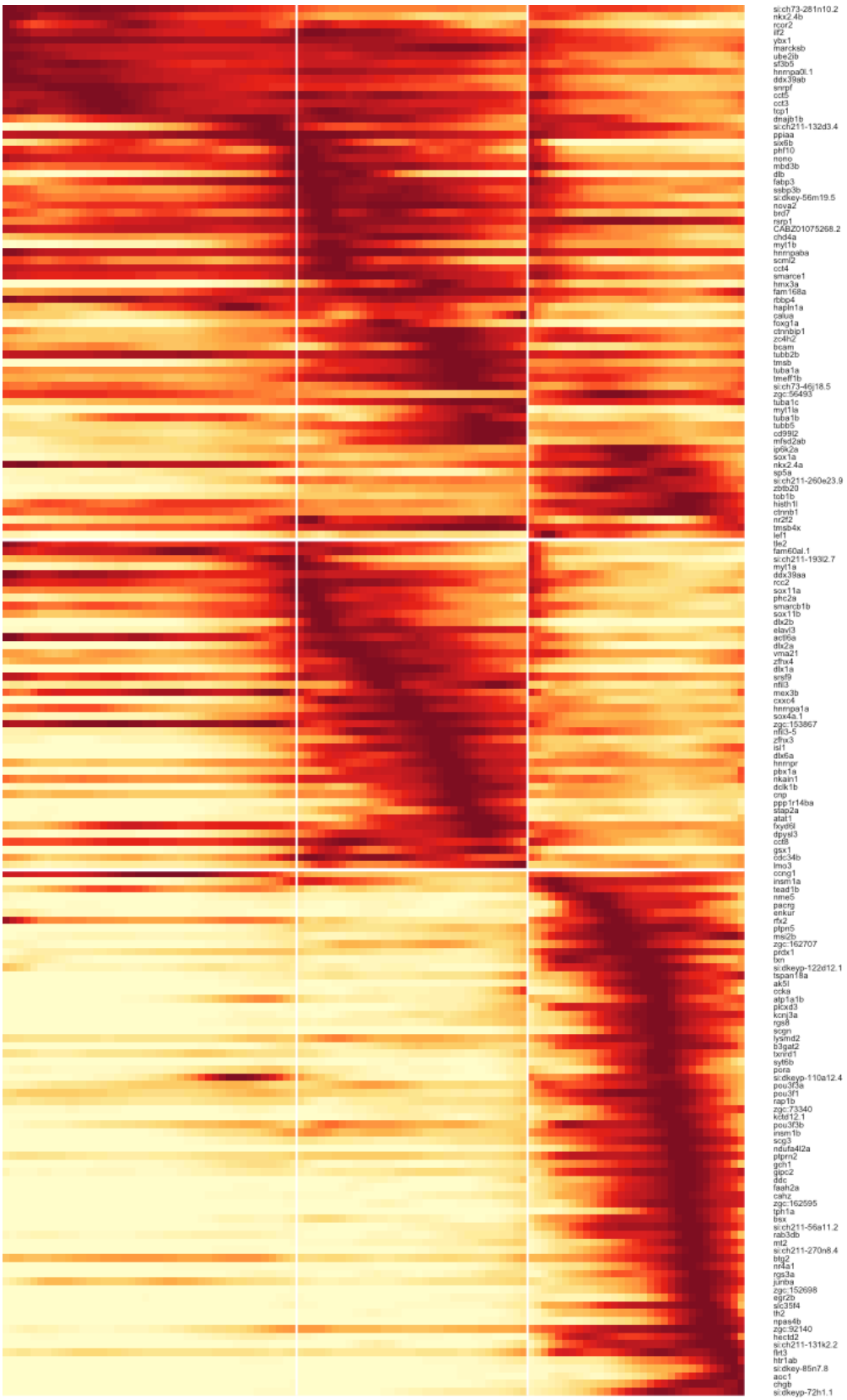
Prepare cascade

```
## Prdx1+ neurons: Seg 3

# Get markers from the two approaches
t <- combined.markers.best[["3"]]  # prdx1+ markers from above the combined clades
t <- t[t$exp.global.fc >= 0.8, ]  # limited to those with good global parameters
m <- threshold.tree.markers(markers, "3", global.fc = 0.6)  # prdx1+ Cell markers from aucprTestAlongTree
prdx.markers <- unique(c(rownames(t), rownames(m)))

# Get markers for the opposing segment
opposing.prdx.markers <- intersect(rownames(nonprdx.markers), rownames(nonprdx.markers.global))

prdx.hm.markers <- unique(c(prdx.markers, opposing.prdx.markers))

# Calculate spline curves Using segments 13 and 12 or 3.
spline.12 <- geneSmoothFit(obj, pseudotime = "pseudotime", cells = cellsInCluster(obj,
    "segment", c("13", "12")), genes = prdx.hm.markers, method = "spline",
    moving.window = 5, cells.per.window = 25, pseudotime.per.window = 0.005,
    spar = 0.5, verbose = F)
spline.3 <- geneSmoothFit(obj, pseudotime = "pseudotime", cells = cellsInCluster(obj,
    "segment", c("13", "3")), genes = prdx.hm.markers, method = "spline",
    moving.window = 5, cells.per.window = 25, pseudotime.per.window = 0.005,
    spar = 0.5, verbose = F)
spline.123 <- geneSmoothFit(obj, pseudotime = "pseudotime", cells = cellsInCluster(obj,
    "segment", c("13", "12", "3")), genes = prdx.hm.markers, method = "spline",
    moving.window = 5, cells.per.window = 25, pseudotime.per.window = 0.005,
    spar = 0.5, verbose = F)

# Want to plot a heatmap that shows expression in most upstream
# progenitors and then each branch (i.e. prdx1- vs. prdx1+ neurons) as
# separate columns. Going to crop each spline fit to the correct
# pseudotime range and then combine them into a single one that can be
# plotted as a three-column heatmap.

pt.12v3 <- obj@tree$segment.pseudotime.limits["3", "start"]  # pseudotime where the crop should happen
splines.prdx <- list(cropSmoothFit(spline.123, pt.min = -Inf, pt.max = pt.12v3),
    cropSmoothFit(spline.12, pt.min = pt.12v3, pt.max = Inf), cropSmoothFit(spline.3,
        pt.min = pt.12v3, pt.max = Inf))
names(splines.prdx) <- c("Hypo Precursors", "Prdx1-", "Prdx1+")
splines.prdx.hm <- combineSmoothFit(splines.prdx)  # Combine into a single one

# Calculate gene expression timing for ordering rows
spline.12 <- determine.timing(s = spline.12)
spline.3 <- determine.timing(s = spline.3)
spline.123 <- determine.timing(s = spline.123)

# Decide which markers are specific to one cell type or both
d12v3 <- divide.branches(obj, prdx.hm.markers, clust.1 = "12", clust.2 = "3",
```

```
    exp.fc = 0.4, exp.thresh = 0.2, exp.diff = 0.1)

# Generate gene ordering based on timing & specificity
order.123 <- filter.heatmap.genes(setdiff(spline.123$gene.order, c(d12v3$specific.1,
    d12v3$specific.2)))
order.12 <- filter.heatmap.genes(intersect(spline.12$gene.order, d12v3$specific.1))
order.3 <- filter.heatmap.genes(intersect(spline.3$gene.order, d12v3$specific.2))
gene.order <- c(order.123, order.12, order.3)

# Output gene table
table.save <- data.frame(gene = gene.order, marks = c(rep("both", length(order.123)),
    rep("prdx1-", length(order.12)), rep("prdx1+", length(order.3))), stringsAsFactors = F)
table.save$clade.AUCPR.ratio <- t[table.save$gene, "AUCPR.ratio"]
table.save$clade.exp.fc <- t[table.save$gene, "exp.fc"]
table.save$clade.exp.fc.global <- t[table.save$gene, "exp.global.fc"]
table.save$non.AUCPR.ratio <- nonprdx.markers[table.save$gene, "AUCPR.ratio"]
table.save$non.exp.fc <- nonprdx.markers[table.save$gene, "exp.fc"]
table.save$non.exp.fc.global <- nonprdx.markers.global[table.save$gene,
    "exp.fc"]
table.save$prdx.AUCPR.ratio.all <- m[table.save$gene, "AUCPR.ratio.all"]
table.save$prdx.AUCPR.ratio.maxBranch <- m[table.save$gene, "AUCPR.ratio.maxBranch"]
table.save$prdx.exp.fc.all <- m[table.save$gene, "expfc.all"]
table.save$prdx.exp.fc.best <- m[table.save$gene, "expfc.maxBranch"]
write.csv(table.save, quote = F, file = paste0(base.path, "/heatmaps/hypo-prdx1-vs-non.csv"))
```

Generate heatmap: all genes

```
# Make sure any values <0 in the spline curves get set to 0 so that the
# heatmap scale doesn't get messed up.
splines.prdx.hm$scaled.smooth[splines.prdx.hm$scaled.smooth < 0] <- 0
# Determine where to place column separators (i.e. how many columns
# will each cell type occupy in the heatmap )
colsep <- cumsum(as.numeric(head(unlist(lapply(splines.prdx, function(x) ncol(x$scaled.smooth))),
    -1)))
# Determine where to place row separators (i.e. how many common
# markers, and markers are specific to each cell type)
rowsep <- cumsum(c(length(order.123), length(order.12)))
# Open a PDF and generate the heatmap pdf(paste0(base.path,
# '/heatmaps/hypo-prdx-vs-non.pdf'), width=6, height=10)
gplots::heatmap.2(x = as.matrix(splines.prdx.hm$scaled.smooth[gene.order,
    ]), Rowv = F, Colv = F, dendrogram = "none", col = cols, trace = "none",
    density.info = "none", key = F, cexCol = 0.8, cexRow = 0.35, margins = c(8,
        8), lwid = c(0.3, 4), lhei = c(0.3, 4), labCol = NA, colsep = colsep,
    rowsep = rowsep, sepwidth = c(0.05, 0.2))
title(main = "Precursors", line = -41, adj = 0)
title(main = "prdx1 -", line = -41, adj = 0.425)
title(main = "prdx1 +", line = -41, adj = 0.725)
```



Precursors **prdx1 -** **prdx1 +**

```
# dev.off()
```

nrgna+ neurons

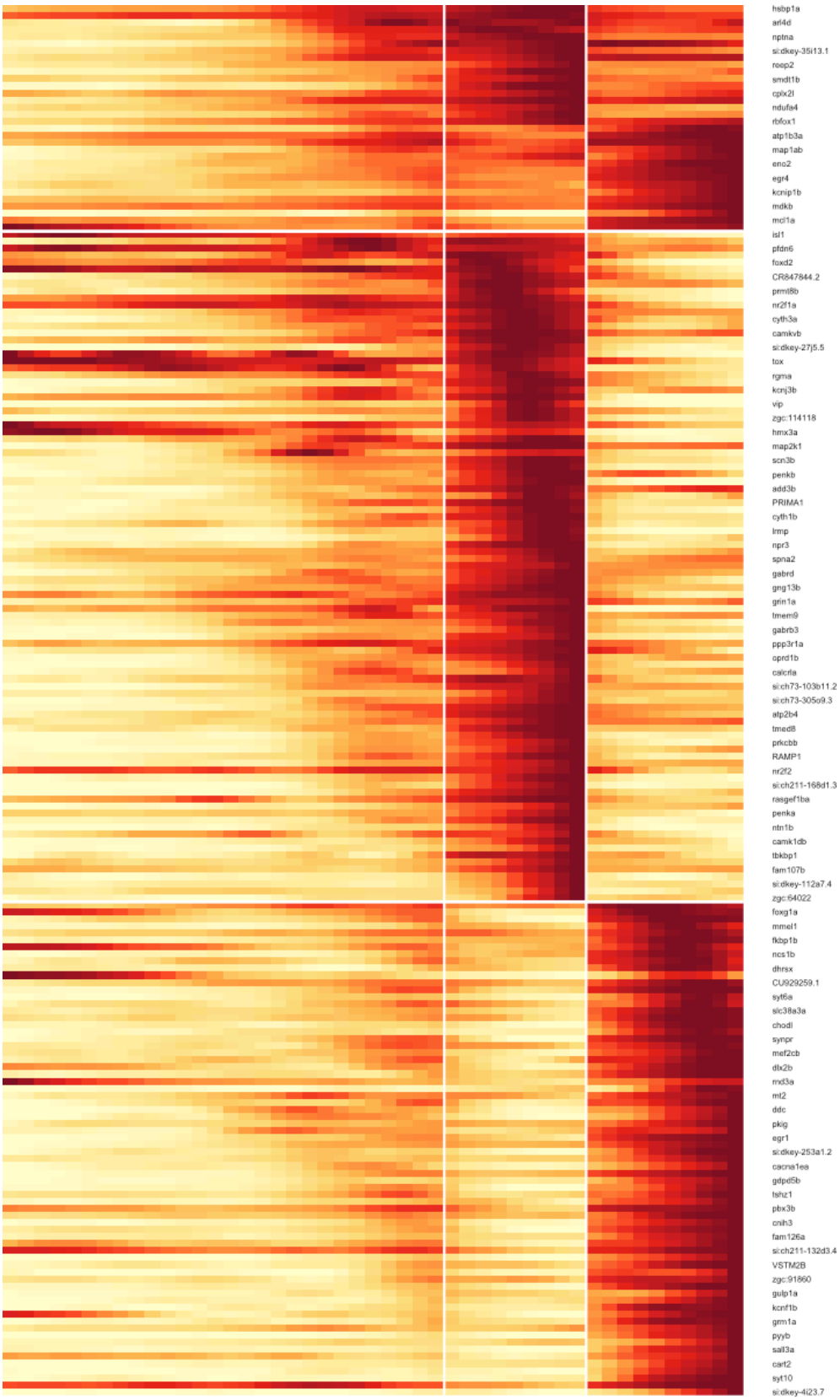
Prepare cascade

```
# Get markers from the two approaches
t <- combined.markers.best[["10"]]  # Markers from above the combined clades
t <- t[t$exp.global.fc >= 0.8, ]  # limited to those with good global parameters
m2 <- threshold.tree.markers(markers, "2", global.fc = 0.6)  # Markers from aucprTestAlongTree
m5 <- threshold.tree.markers(markers, "5", global.fc = 0.6)  # Markers from aucprTestAlongTree
tacsyn.markers <- unique(c(rownames(t), rownames(m2), rownames(m5)))

# Just want to plot part of cells from upstream segment 12, which is
# very long. Going to use cells from segments 2 or 5, 10, and from
# segment 12 with pseudotime > 0.23
cells.plot.2 <- unique(c(intersect(cellsInCluster(obj, "segment", "12"),
    whichCells(obj, "pseudotime", c(0.45, Inf))), cellsInCluster(obj, "segment",
    c("10", "2"))))
cells.plot.5 <- unique(c(intersect(cellsInCluster(obj, "segment", "12"),
    whichCells(obj, "pseudotime", c(0.45, Inf))), cellsInCluster(obj, "segment",
    c("10", "5"))))
cells.plot.25 <- unique(c(intersect(cellsInCluster(obj, "segment", "12"),
    whichCells(obj, "pseudotime", c(0.45, Inf))), cellsInCluster(obj, "segment",
    c("10", "2", "5"))))

# Calculate spline curves
spline.2 <- geneSmoothFit(obj, pseudotime = "pseudotime", cells = cells.plot.2,
    genes = tacsyn.markers, method = "spline", moving.window = 5, cells.per.window = 25,
    pseudotime.per.window = 0.005, spar = 0.5, verbose = F)
spline.5 <- geneSmoothFit(obj, pseudotime = "pseudotime", cells = cells.plot.5,
    genes = tacsyn.markers, method = "spline", moving.window = 5, cells.per.window = 25,
    pseudotime.per.window = 0.005, spar = 0.5, verbose = F)
spline.25 <- geneSmoothFit(obj, pseudotime = "pseudotime", cells = cells.plot.25,
    genes = tacsyn.markers, method = "spline", moving.window = 5, cells.per.window = 25,
    pseudotime.per.window = 0.005, spar = 0.5, verbose = F)

# Want to plot a heatmap that shows expression in upstream progenitors
# and then each branch (i.e. gabaergic_tac1 vs. synpr+) as separate
# columns. Going to crop each spline fit to the correct pseudotime
# range and then combine them into a single one that can be plotted as
# a three-column heatmap.

pt.2v5 <- obj@tree$segment.pseudotime.limits["2", "start"]  # pseudotime where the crop should happen
splines.tacsyn <- list(cropSmoothFit(spline.25, pt.min = -Inf, pt.max = pt.2v5),
    cropSmoothFit(spline.2, pt.min = pt.2v5, pt.max = Inf), cropSmoothFit(spline.5,
        pt.min = pt.2v5, pt.max = Inf))
names(splines.tacsyn) <- c("Precursors", "Synpr-", "Synpr+")
splines.tacsyn.hm <- combineSmoothFit(splines.tacsyn)  # Combine into a single one

# Calculate gene expression timing for ordering rows
spline.2 <- determine.timing(s = spline.2)
spline.5 <- determine.timing(s = spline.5)
```

```
spline.25 <- determine.timing(s = spline.25)

# Decide which markers are specific to one cell type or both
d2v5 <- divide.branches(obj, tacsyn.markers, clust.1 = "2", clust.2 = "5",
    exp.fc = 0.4, exp.thresh = 0.2, exp.diff = 0.1)

# Generate gene ordering based on timing & specificity
order.25 <- filter.heatmap.genes(setdiff(spline.25$gene.order, c(d2v5$specific.1,
    d2v5$specific.2)))
order.2 <- filter.heatmap.genes(intersect(spline.2$gene.order, d2v5$specific.1))
order.5 <- filter.heatmap.genes(intersect(spline.5$gene.order, d2v5$specific.2))
gene.order <- c(order.25, order.2, order.5)

# Output gene table
table.save <- data.frame(gene = gene.order, marks = c(rep("both", length(order.25)),
    rep("synpr-", length(order.2)), rep("synpr+", length(order.5))), stringsAsFactors = F)
table.save$clade.AUCPR.ratio <- t[table.save$gene, "AUCPR.ratio"]
table.save$clade.exp.fc <- t[table.save$gene, "exp.fc"]
table.save$clade.exp.fc.global <- t[table.save$gene, "exp.global.fc"]
table.save$nonsynpr.AUCPR.ratio.all <- m2[table.save$gene, "AUCPR.ratio.all"]
table.save$nonsynpr.AUCPR.ratio.maxBranch <- m2[table.save$gene, "AUCPR.ratio.maxBranch"]
table.save$nonsynpr.exp.fc.all <- m2[table.save$gene, "expfc.all"]
table.save$nonsynpr.exp.fc.best <- m2[table.save$gene, "expfc.maxBranch"]
table.save$synpr.AUCPR.ratio.all <- m5[table.save$gene, "AUCPR.ratio.all"]
table.save$synpr.AUCPR.ratio.maxBranch <- m5[table.save$gene, "AUCPR.ratio.maxBranch"]
table.save$synpr.exp.fc.all <- m5[table.save$gene, "expfc.all"]
table.save$synpr.exp.fc.best <- m5[table.save$gene, "expfc.maxBranch"]
write.csv(table.save, quote = F, file = paste0(base.path, "/heatmaps/hypo-nrgna.csv"))
```

Generate heatmap: all genes

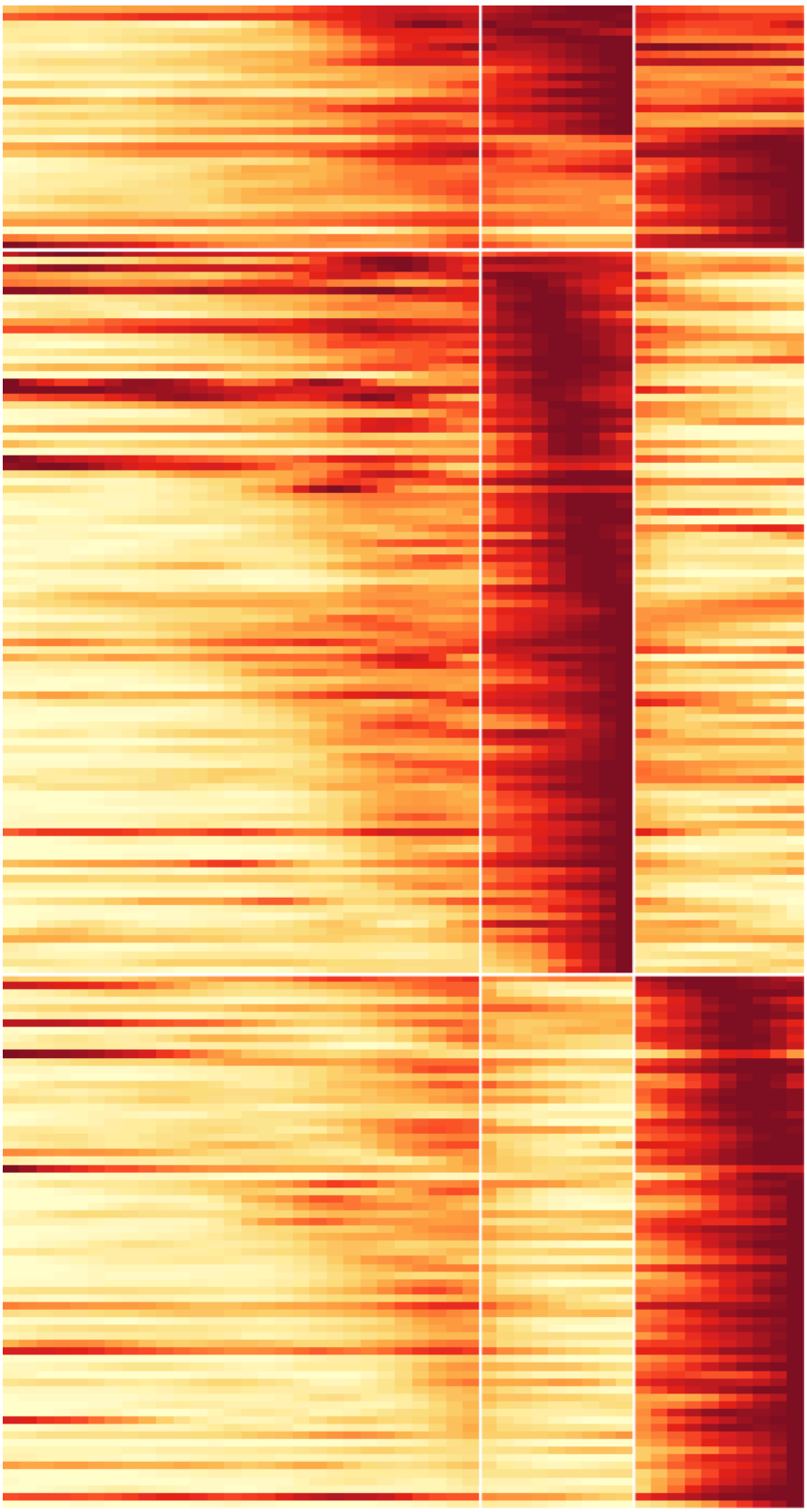
```
# Make sure any values <0 in the spline curves get set to 0 so that the
# heatmap scale doesn't get messed up.
splines.tacsyn.hm$scaled.smooth[splines.tacsyn.hm$scaled.smooth < 0] <- 0
# Determine where to place column separators (i.e. how many columns
# will each cell type occupy in the heatmap )
colsep <- cumsum(as.numeric(head(unlist(lapply(splines.tacsyn, function(x) ncol(x$scaled.smooth))),
    -1)))
# Determine where to place row separators (i.e. how many common
# markers, and markers are specific to each cell type)
rowsep <- cumsum(c(length(order.25), length(order.2)))
# Open a PDF and generate the heatmap pdf(paste0(base.path,
# '/heatmaps/hypo-nrgna.pdf'), width=6, height=10)
gplots::heatmap.2(x = as.matrix(splines.tacsyn.hm$scaled.smooth[gene.order,
    ]), Rowv = F, Colv = F, dendrogram = "none", col = cols, trace = "none",
    density.info = "none", key = F, cexCol = 0.8, cexRow = 0.35, margins = c(8,
        8), lwid = c(0.3, 4), lhei = c(0.3, 4), labCol = NA, colsep = colsep,
    rowsep = rowsep, sepwidth = c(0.05, 0.2))
title(main = "synpr-", line = -41, adj = 0.535)
title(main = "synpr+", line = -41, adj = 0.75)
```


hsbp1a
arl4d
nptna
si:dkey-35i13.1
reep2
smdt1b
cplx2l
ndufa4
rbfox1
atp1b3a
map1ab
eno2
egr4
kcnip1b
mdkb
mcl1a
isl1
pfdn6
foxd2
CR847844.2
prmt8b
nr2f1a
cyth3a
camkvb
si:dkey-27j5.5
tox
rgma
kcnj3b
vip
zgc:114118
hmx3a
map2k1
scn3b
penkb
add3b
PRIMA1
cyth1b
lrmp
npr3
spna2
gabrd
gng13b
grin1a
tmem9
gabrb3
ppp3r1a
oprd1b
calcrla
si:ch73-103b11.2
si:ch73-305o9.3
atp2b4
tmed8
prkcbb
RAMP1
nr2f2
si:ch211-168d1.3
rasgef1ba
penka
ntn1b
camk1db
tbkbp1
fam107b
si:dkey-112a7.4
zgc:64022
foxg1a
mmel1
fkbp1b
ncs1b
dhrsx
CU929259.1
syt6a
slc38a3a
chodl
synpr
mef2cb
dlx2b
rnd3a
mt2
ddc
pkig
egr1
si:dkey-253a1.2
cacna1ea
gdpd5b
tshz1
pbx3b
cnih3
fam126a
si:ch211-132d3.4
VSTM2B
zgc:91860
gulp1a
kcnf1b
grm1a
pyyb
sall3a
cart2
syt10
si:dkey-4i23.7
synpr-
synpr+

```
# dev.off()
```

Generate heatmap: main figure

```
# Plot heatmap with only certain genes labeled for main figure

genes.to.plot <- c("rgs5b", "dlx5a", "isl1", "nkx2.4a", "nkx2.2a", "hmx3a",
    "dlx1a", "dlx2b", "sp8a", "pbx3b")
rownames.to.plot <- gene.order
rtp <- rownames.to.plot %in% genes.to.plot
rownames.to.plot[!rtp] <- ""
rownames.to.plot[rtp] <- paste0("- ", rownames.to.plot[rtp])

# Open a PDF and generate the heatmap pdf(paste0(base.path,
# '/heatmaps/hypo-nrgna-mainfig.pdf'), width=6, height=10)
gplots::heatmap.2(x = as.matrix(splines.tacsyn.hm$scaled.smooth[gene.order,
    ]), Rowv = F, Colv = F, dendrogram = "none", col = cols, trace = "none",
    density.info = "none", key = F, cexCol = 0.8, cexRow = 1.8, margins = c(8,
        8), lwid = c(0.3, 4), lhei = c(0.3, 4), labCol = NA, colsep = colsep,
    rowsep = rowsep, sepwidth = c(0.05, 0.2), labRow = rownames.to.plot)
title(main = "synpr-", line = -41, adj = 0.535)
title(main = "synpr+", line = -41, adj = 0.75)
```


synpr-
synpr+

```
# dev.off()
```

GABAergic neurons

Prepare cascade

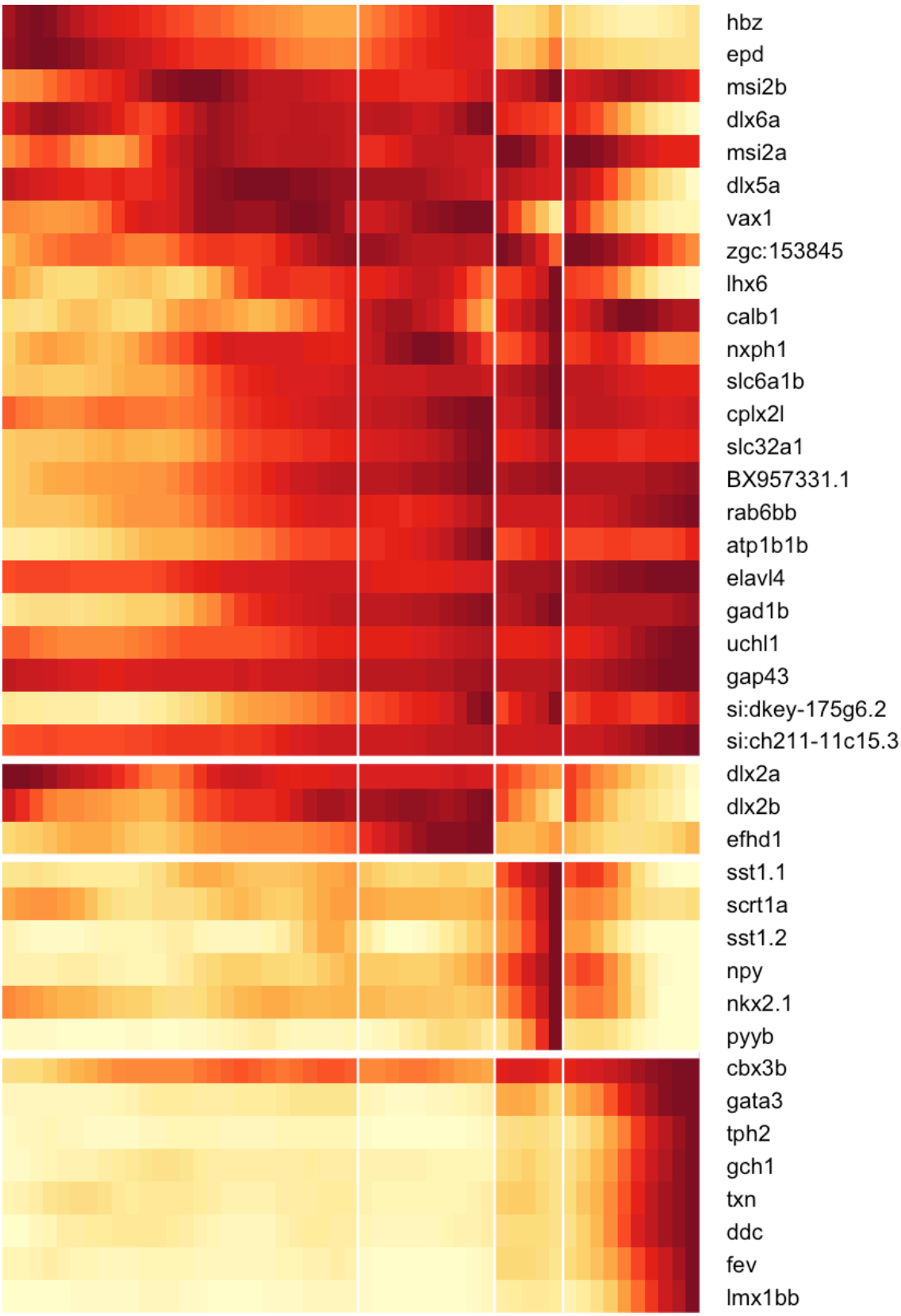
```
# Get markers from the two approaches
t <- combined.markers.best[["9"]]  # Markers from above the combined clades
m1 <- threshold.tree.markers(markers, "1", global.fc = 0.6)  # Markers from aucprTestAlongTree
m6 <- threshold.tree.markers(markers, "6", global.fc = 0.6)  # Markers from aucprTestAlongTree
m7 <- threshold.tree.markers(markers, "7", global.fc = 0.6)  # Markers from aucprTestAlongTree
gaba.markers <- unique(c(rownames(t), rownames(m1), rownames(m6), rownames(m7)))

# Defining cell populations to use in the heatmap Just want to use a
# tiny bit of segment 12 and then the rest of the gabaergic clade
cells.plot.1 <- unique(c(intersect(cellsInCluster(obj, "segment", "12"),
    whichCells(obj, "pseudotime", c(0.5, Inf))), cellsInCluster(obj, "segment",
    c("11", "9", "1"))))
cells.plot.6 <- unique(c(intersect(cellsInCluster(obj, "segment", "12"),
    whichCells(obj, "pseudotime", c(0.5, Inf))), cellsInCluster(obj, "segment",
    c("11", "9", "8", "6"))))
cells.plot.7 <- unique(c(intersect(cellsInCluster(obj, "segment", "12"),
    whichCells(obj, "pseudotime", c(0.5, Inf))), cellsInCluster(obj, "segment",
    c("11", "9", "8", "7"))))
cells.plot.167 <- unique(c(intersect(cellsInCluster(obj, "segment", "12"),
    whichCells(obj, "pseudotime", c(0.5, Inf))), cellsInCluster(obj, "segment",
    c("11", "9", "8", "1", "6", "7"))))

# Calculate spline curves
spline.1 <- geneSmoothFit(obj, pseudotime = "pseudotime", cells = cells.plot.1,
    genes = gaba.markers, method = "spline", moving.window = 5, cells.per.window = 25,
    pseudotime.per.window = 0.005, spar = 0.5, verbose = F)
spline.6 <- geneSmoothFit(obj, pseudotime = "pseudotime", cells = cells.plot.6,
    genes = gaba.markers, method = "spline", moving.window = 5, cells.per.window = 25,
    pseudotime.per.window = 0.005, spar = 0.5, verbose = F)
spline.7 <- geneSmoothFit(obj, pseudotime = "pseudotime", cells = cells.plot.7,
    genes = gaba.markers, method = "spline", moving.window = 5, cells.per.window = 25,
    pseudotime.per.window = 0.005, spar = 0.5, verbose = F)
spline.167 <- geneSmoothFit(obj, pseudotime = "pseudotime", cells = cells.plot.167,
    genes = gaba.markers, method = "spline", moving.window = 5, cells.per.window = 25,
    pseudotime.per.window = 0.005, spar = 0.5, verbose = F)

# Want to plot a heatmap that shows expression in upstream progenitors
# and then each branch as separate columns. Going to crop each spline
# fit to the correct pseudotime range and then combine them into a
# single one that can be plotted as a three-column heatmap.

pt.1 <- obj@tree$segment.pseudotime.limits["1", "start"]  # pseudotime where the crop should happen
splines.gaba <- list(cropSmoothFit(spline.167, pt.min = -Inf, pt.max = pt.1),
    cropSmoothFit(spline.1, pt.min = pt.1, pt.max = Inf), cropSmoothFit(spline.6,
        pt.min = pt.1, pt.max = Inf), cropSmoothFit(spline.7, pt.min = pt.1,
        pt.max = Inf))
names(splines.gaba) <- c("Precursors", "dlx+", "sst1.1+", "tph2+")
```

```
splines.gaba.hm <- combineSmoothFit(splines.gaba)  # Combine into a single one

# Calculate gene expression timing for ordering rows
spline.1 <- determine.timing(s = spline.1)
spline.6 <- determine.timing(s = spline.6)
spline.7 <- determine.timing(s = spline.7)
spline.167 <- determine.timing(s = spline.167, genes = setdiff(gaba.markers,
    "sst1.2"))

# Decide which markers are specific to one cell type or not specific
d <- divide.branches.triple(obj, gaba.markers, clust.1 = "1", clust.2 = "6",
    clust.3 = "7", exp.fc = 0.4, exp.thresh = 0.2, exp.diff = 0.1)

# Generate gene ordering based on timing & specificity
order.167 <- filter.heatmap.genes(intersect(spline.167$gene.order, d$nonspecific))
order.1 <- filter.heatmap.genes(intersect(spline.1$gene.order, d$specific.1))
order.6 <- filter.heatmap.genes(intersect(spline.6$gene.order, d$specific.2))
order.7 <- filter.heatmap.genes(intersect(spline.7$gene.order, d$specific.3))
gene.order <- c(order.167, order.1, order.6, order.7)

# Output gene table
table.save <- data.frame(gene = gene.order, marks = c(rep("multiple", length(order.167)),
    rep("dlx+", length(order.1)), rep("sst+", length(order.6)), rep("tph2+",
        length(order.7))), stringsAsFactors = F)
table.save$clade.AUCPR.ratio <- t[table.save$gene, "AUCPR.ratio"]
table.save$clade.exp.fc <- t[table.save$gene, "exp.fc"]
table.save$clade.exp.fc.global <- t[table.save$gene, "exp.global.fc"]
table.save$dlx.AUCPR.ratio.all <- m1[table.save$gene, "AUCPR.ratio.all"]
table.save$dlx.AUCPR.ratio.maxBranch <- m1[table.save$gene, "AUCPR.ratio.maxBranch"]
table.save$dlx.exp.fc.all <- m1[table.save$gene, "expfc.all"]
table.save$dlx.exp.fc.best <- m1[table.save$gene, "expfc.maxBranch"]
table.save$sst.AUCPR.ratio.all <- m6[table.save$gene, "AUCPR.ratio.all"]
table.save$sst.AUCPR.ratio.maxBranch <- m6[table.save$gene, "AUCPR.ratio.maxBranch"]
table.save$sst.exp.fc.all <- m6[table.save$gene, "expfc.all"]
table.save$sst.exp.fc.best <- m6[table.save$gene, "expfc.maxBranch"]
table.save$tph.AUCPR.ratio.all <- m7[table.save$gene, "AUCPR.ratio.all"]
table.save$tph.AUCPR.ratio.maxBranch <- m7[table.save$gene, "AUCPR.ratio.maxBranch"]
table.save$tph.exp.fc.all <- m7[table.save$gene, "expfc.all"]
table.save$tph.exp.fc.best <- m7[table.save$gene, "expfc.maxBranch"]
write.csv(table.save, quote = F, file = paste0(base.path, "/heatmaps/hypo-gaba.csv"))
```

Generate heatmap: all genes

```
# Make sure any values <0 in the spline curves get set to 0 so that the
# heatmap scale doesn't get messed up.
splines.gaba.hm$scaled.smooth[splines.gaba.hm$scaled.smooth < 0] <- 0
# Determine where to place column separators (i.e. how many columns
# will each cell type occupy in the heatmap )
colsep <- cumsum(as.numeric(head(unlist(lapply(splines.gaba, function(x) ncol(x$scaled.smooth))),
    -1)))
# Determine where to place row separators (i.e. how many common
# markers, and markers are specific to each cell type)
```

```
rowsep <- cumsum(c(length(order.167), length(order.1), length(order.6)))
# Open a PDF and generate the heatmap pdf(paste0(base.path,
# '/heatmaps/hypo-gaba.pdf'), width=6, height=10)
gplots::heatmap.2(x = as.matrix(splines.gaba.hm$scaled.smooth[gene.order,
    ]), Rowv = F, Colv = F, dendrogram = "none", col = cols, trace = "none",
    density.info = "none", key = F, cexCol = 0.8, cexRow = 1, margins = c(8,
        8), lwid = c(0.3, 4), lhei = c(0.3, 4), labCol = NA, colsep = colsep,
    rowsep = rowsep, sepwidth = c(0.05, 0.2))
title(main = "dlx+", line = -41, adj = 0.45)
title(main = "sst+", line = -41, adj = 0.61)
title(main = "tph2+", line = -41, adj = 0.75)
```


hbz
epd
msi2b
dlx6a
msi2a
dlx5a
vax1
zgc:153845
lhx6
calb1
nxph1
slc6a1b
cplx2l
slc32a1
BX957331.1
rab6bb
atp1b1b
elavl4
gad1b
uchl1
gap43
si:dkey-175g6.2
si:ch211-11c15.3
dlx2a
dlx2b
efhd1
sst1.1
scrt1a
sst1.2
npy
nkx2.1
pyyb
cbx3b
gata3
tph2
gch1
txn
ddc
fev
lmx1bb
dlx+
sst+
tph2+

```
# dev.off()
```

Embryonic molecular profiles are not found in larval progenitors

We were looking to compare the molecular profiles of embryonic and post-embryonic progenitors. In the retina, we found progenitors at larval stages whose molecular signatures were preserved from embryonic stages (see Retina 3). Here, we find that, in the hypothalamus, progenitors at larval stages are transcriptionally different from embryonic progenitors.

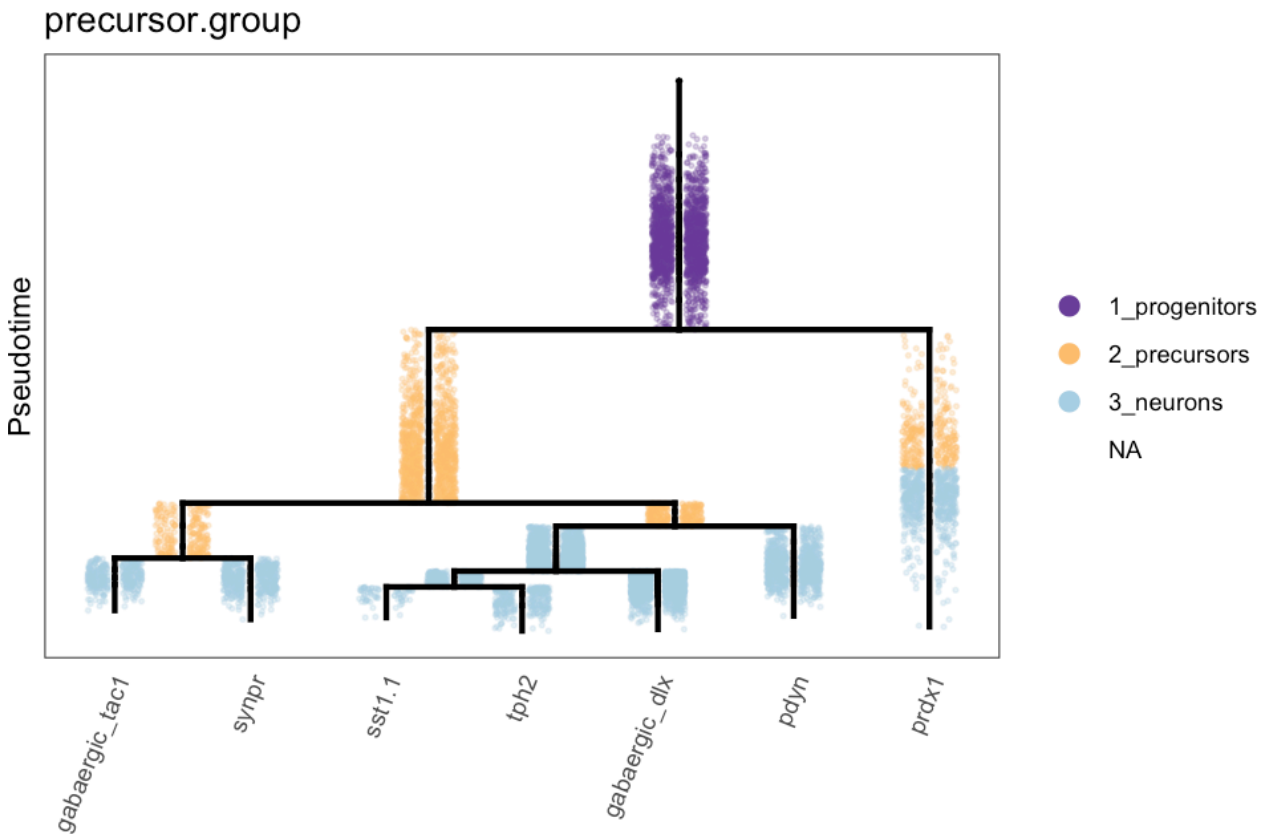
Identify populations to compare

We defined progenitor / precursor / neuron populations based on their location in the tree, cross-referenced with the expression of markers of each of these types.

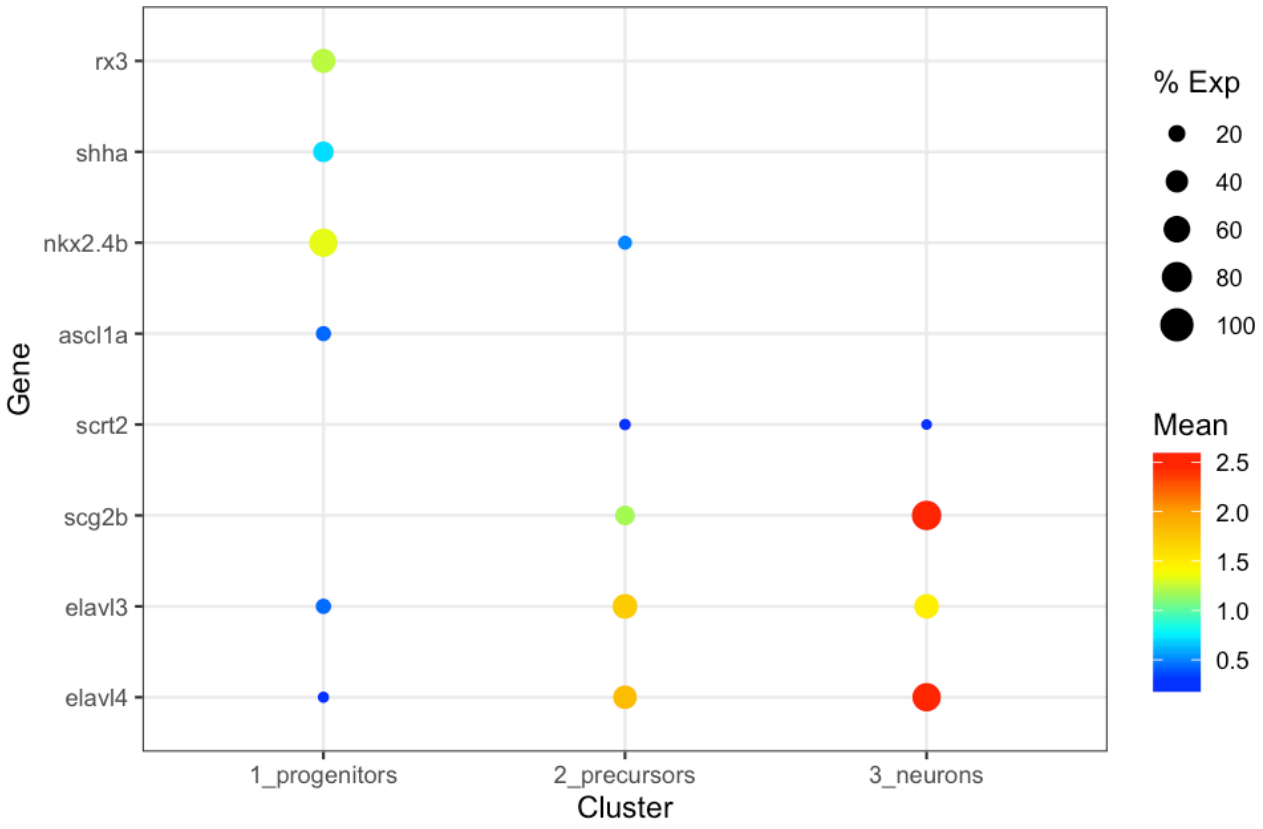
```
obj@group.ids$precursor.group <- NA

cells.s13 <- intersect(cellsInCluster(obj, "segment", "13"), whichCells(obj,
    "pseudotime", c(0.05, 1)))

cells.s3neurons <- intersect(cellsInCluster(obj, "segment", "3"), whichCells(obj,
    "pseudotime", c(0.5, 1)))

cells.s3precursors <- intersect(cellsInCluster(obj, "segment", "3"), whichCells(obj,
    "pseudotime", c(0.3, 0.5)))

obj@group.ids[cells.s13, "precursor.group"] <- "1_progenitors"

obj@group.ids[c(cells.s3precursors, cellsInCluster(obj, "segment", c("12",
    "11", "10"))), "precursor.group"] <- "2_precursors"

obj@group.ids[c(cells.s3neurons, cellsInCluster(obj, "segment", c("9",
    "1", "4", "8", "6", "7", "2", "5"))), "precursor.group"] <- "3_neurons"
```

```
# Colors for ggplot
stage.colors <- RColorBrewer::brewer.pal(12, "Paired")[c(10, 7, 1)]

# Plot tree to show where the groups are
plotTree(obj, "precursor.group", discrete.colors = stage.colors)
```

```
## Warning: Removed 175 rows containing missing values (geom_point).
```




```
# Plot genes in each group
plotDot(obj, genes = c("rx3", "shha", "nkx2.4b", "ascl1a", "scrt2", "scg2b",
    "elavl3", "elavl4"), clustering = "precursor.group", scale.by = "area") +
    theme_bw()
```



Determine proportion of cells in each state

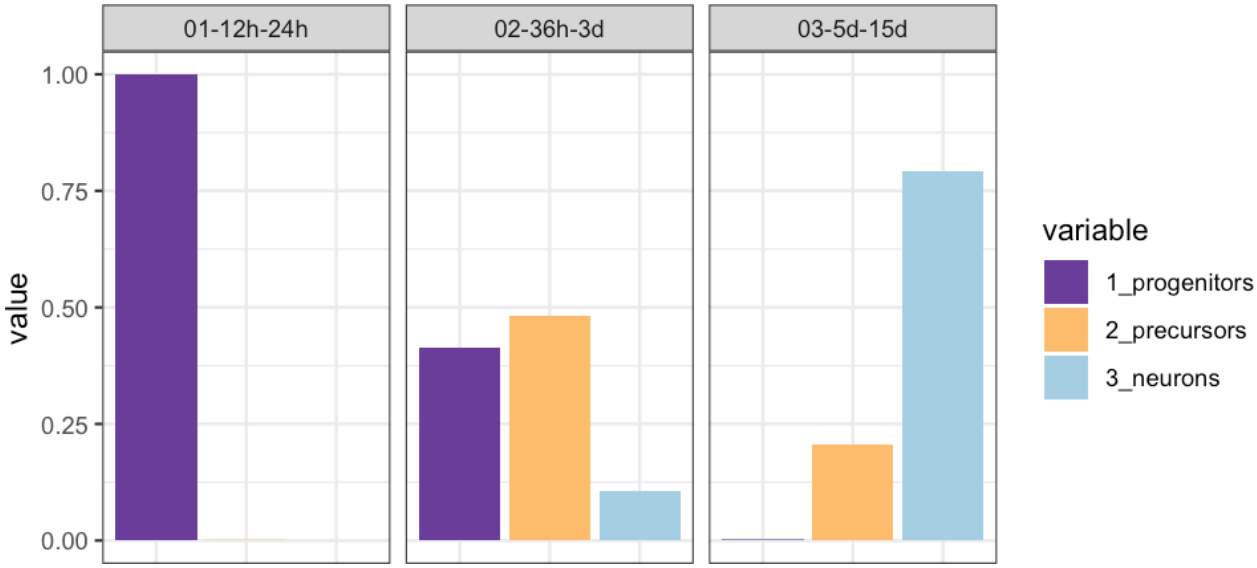
We then determined the proportion of cells from different stages that fell into each of these transcriptional states.

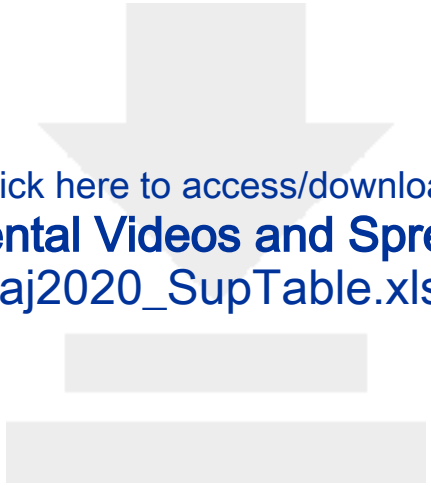
```
# Combine stages to reduce number of plots
obj@group.ids$stage.collapsed <- plyr::mapvalues(x = obj@group.ids$stage,
    from = c("01-12h", "02-14h", "03-16h", "04-18h", "05-20h", "06-24h",
        "07-36h", "08-2d", "09-3d", "10-5d", "11-8d", "12-15d"), to = c(rep("01-12h-24h",
        6), rep("02-36h-3d", 3), rep("03-5d-15d", 3)))

# Count number of cells from each stage group in each precursor group
stage.group.count <- plyr::count(obj@group.ids, vars = c("stage.collapsed",
    "precursor.group"))

# Remove cells that weren't part of a precursor group
stage.group.count <- stage.group.count[complete.cases(stage.group.count),
    ]

# Cast into a data frame and convert NA to 0 (no cells of that type
# observed)
stage.group.df <- reshape2::dcast(stage.group.count, formula = stage.collapsed ~
    precursor.group)
```

```
## Using freq as value column: use value.var to override.
```

```
stage.group.df[is.na(stage.group.df)] <- 0
```

```r
# Normalize by the number of precursors from each stage group
stage.group.df[, 2:4] <- sweep(stage.group.df[, 2:4], 1, rowSums(stage.group.df[,
    2:4]), "/")

# Melt for ggplot
stage.group.df.melt <- reshape2::melt(stage.group.df, id.vars = "stage.collapsed")

# Plot proportions
ggplot(stage.group.df.melt, aes(x = variable, y = value, group = stage.collapsed,
    fill = variable)) + geom_bar(stat = "identity") + facet_wrap(~stage.collapsed) +
    theme_bw() + scale_fill_manual(values = stage.colors) + theme(axis.title.x = element_blank(),
    axis.text.x = element_blank(), axis.ticks.x = element_blank())
```





Click here to access/download

Supplemental Videos and Spreadsheets

Raj2020_SupTable.xlsx