Appendix - Single-cell transcriptomics identifies Gadd45b as a regulator of herpesvirus-reactivating neurons

Aligned data were subsequently imported into Seurat v3.0 for additional quality filtering, identification of highly variable genes, data integration, dimensionality reduction, unsupervised clustering and integrated differential gene expression analysis using MAST. Full details of these analysis are recorded in R Markdown files.

Appendix Table S1

Pathway name	Entities					Reactions			Species name	-log10(FDR)
	found	Total	ratio	pValue	FDR	found	total	ratio		
Cellular responses to stress	9	691	0.047	7.67E-04	9.97E-03	20	227	0.018	Homo sapiens	2.00
HSF1 activation	6	43	0.003	1.29E-08	2.07E-07	1	7	0.001	Homo sapiens	6.68
Cellular response to heat stress	9	135	0.009	1.45E-09	2.91E-08	12	29	0.002	Homo sapiens	7.54
Regulation of HSF1-mediated heat shock response	9	113	0.008	3.11E-10	8.40E-09	7	14	0.001	Homo sapiens	8.08
HSF1-dependent transactivation	8	59	0.004	5.16E-11	2.11E-09	4	8	0.001	Homo sapiens	8.68
Attenuation phase	8	47	0.003	8.64E-12	7.17E-10	3	5	0	Homo sapiens	9.14

Pathway analysis of differentially expressed genes reported in Dataset EV1 identified six cellular pathways impacted by HSV-1 reactivation

Appendix Table S2

Sample ID	Simple name	Experiment accession	Run accession	Chemistry	Batch	Estimated number of cells [post filtering]
Uninfected BioRep #1	Uninf_V2_rep_1	ERX4858799	ERR5052726	10X Genomics Chromium Single Cell 3' v2	1	3066
Latent BioRep #1	Lat_V2_rep_1	ERX4858240	ERR5051908			1058
Latent BioRep #2	Lat_V2_rep_2	ERX4858255	ERR5051923		2	3871
Reactivation (20 hrs) BioRep #1	Reac20_V2_rep_1	ERX4858418	ERR5052302			5675
Reactivation (48 hrs) BioRep #1	Reac48_V2_rep_1	ERX4858600	ERR5052485			2727
Latent BioRep #3	Lat_V2_rep_3	ERX4858416	ERR5052300		3	2079
Uninfected BioRep #2	Uninf_V2_rep_2	ERX4858776	ERR5052682		4	2832
Uninfected (20 hrs LY treatment) BioRep #1	UninfLY_V2_rep_3	ERX4858740	ERR5052625			1661
Reactivation (20 hrs) BioRep #2	Reac20_V3_rep_1	ERX4858599	ERR5052484	10X Genomics Chromium Single Cell 3' v3	5	1195
Reactivation (48 hrs) BioRep #2	Reac48_V3_rep_1	ERX4858735	ERR5052620			1700
Reactivation (72 hrs) BioRep #1	Reac72_V3_rep_1	ERX4858741	ERR5052626			2163

List of sequence datasets generated as part of this study. Libraries were sequenced in paired-end mode on either an Illumina HiSeq 4000 or Illumina NovaSeq 6000.

All data are available via the European Nucleotide Archive under project accession: PRJEB39022

Appendix Table S3

Transcription Unit	HSV-1 transcripts
LAT	LAT*
RL1	RL1*
RL2	RL2*
RS1	RS1*
TU1	UL1; UL2; UL3
TU10	US8; US8A; US9
TU11	UL4; UL5
TU12	UL8; UL9
TU13	UL11; UL12; UL13; UL14
TU14	UL16; UL17
TU15	UL18; UL19; UL20
TU16	UL27; UL28
TU17	UL31; UL32
TU18	UL46; UL47
TU19	UL49; UL49A
TU2	UL6; UL7
TU20	US10; US11gfp; US12
TU3	UL24; UL25; UL26
TU4	UL33; UL34; UL35
TU5	UL38; UL39; UL40
TU6	UL43; UL44; UL45
TU7	UL52; UL53; UL54; UL55
TU8	US3; US4
TU9	US5; US6; US7
UL10	UL10
UL15	UL15
UL21	UL21
UL22	UL22
UL23	UL23
UL29	UL29
UL30	UL30
UL36	UL36
UL37	UL37
UL41	UL41
UL42	UL42
UL50	UL50
UL51	UL51
UL56	UL56
US1	US1
US12	US12
US2	US2

HSV-1 RNAs were grouped into a set of 20 transcription units (TU1-20) for the purposes of mapping and analysis.

*** data aggregated between both copies**

List of HSV-1 ORFs and transcription units (TUs) included in this study. Transcription units comprise multiple ORFs encoded on overlapping mRNAs.

scRNASeq-V2-analysis

```
### Load required libraries
library(Seurat)
library(cowplot)
library(dplyr)
library(Matrix)
library(MAST)

### Import Cell Ranger Gene Expression Matrices
dmso1.data <- Read10X(data.dir = "/path/to/scRNASeq-V2data/Uninf_V2_rep_1/")
dmso2.data <- Read10X(data.dir = "/path/to/scRNASeq-V2data/Uninf_V2_rep_2/")
ly.data <- Read10X(data.dir = "/path/to/scRNASeq-V2data/UninfLY_V2_rep_3/")
latent1.data <- Read10X(data.dir = "/path/to/scRNASeq-V2data/Lat_V2_rep_1/")
latent2.data <- Read10X(data.dir = "/path/to/scRNASeq-V2data/Lat_V2_rep_2/")
latent3.data <- Read10X(data.dir = "/path/to/scRNASeq-V2data/Lat_V2_rep_3/")
ly20.data <- Read10X(data.dir = "/path/to/scRNASeq-V2data/Reac20_V2_rep_1/")
ly48.data <- Read10X(data.dir = "/path/to/scRNASeq-V2data/Reac20_V2_rep_2/")

### Setup Seurat Objects
dmso1 <- CreateSeuratObject(counts = dmso1.data)
dmso2 <- CreateSeuratObject(counts = dmso2.data)
ly <- CreateSeuratObject(counts = ly.data)
latent1 <- CreateSeuratObject(counts = latent1.data)
latent2 <- CreateSeuratObject(counts = latent2.data)
latent3 <- CreateSeuratObject(counts = latent3.data)
ly20 <- CreateSeuratObject(counts = ly20.data)
ly48 <- CreateSeuratObject(counts = ly48.data)

### Data exploration to determine filtering parameters
ly20[["percent.mt"]] <- PercentageFeatureSet(object = ly20, pattern = "^Mt-")
v2<-VlnPlot(object = ly20, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"),
ncol = 3)
plot1 <- FeatureScatter(object = ly20, feature1 = "nCount_RNA", feature2 =
"percent.mt")
plot2 <- FeatureScatter(object = ly20, feature1 = "nCount_RNA", feature2 =
"nFeature_RNA")
plot_grid(plot1,plot2,v2)

ly48[["percent.mt"]] <- PercentageFeatureSet(object = ly48, pattern = "^Mt-")
v3<-VlnPlot(object = ly48, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"),
ncol = 3)
plot1 <- FeatureScatter(object = ly48, feature1 = "nCount_RNA", feature2 =
"percent.mt")
plot2 <- FeatureScatter(object = ly48, feature1 = "nCount_RNA", feature2 =
"nFeature_RNA")
plot_grid(plot1,plot2,v3)

ly[["percent.mt"]] <- PercentageFeatureSet(object = ly, pattern = "^Mt-")
v4<-VlnPlot(object = ly, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"),
ncol = 3)
plot1 <- FeatureScatter(object = ly, feature1 = "nCount_RNA", feature2 =
"percent.mt")
```

```
plot2 <- FeatureScatter(object = ly, feature1 = "nCount_RNA", feature2 =
"nFeature_RNA")
plot_grid(plot1,plot2,v4)

dmso1[["percent.mt"]] <- PercentageFeatureSet(object = dmso1, pattern = "^Mt-")
v5<-VlnPlot(object = dmso1, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"),
ncol = 3)
plot1 <- FeatureScatter(object = dmso1, feature1 = "nCount_RNA", feature2 =
"percent.mt")
plot2 <- FeatureScatter(object = dmso1, feature1 = "nCount_RNA", feature2 =
"nFeature_RNA")
plot_grid(plot1,plot2,v5)

dmso2[["percent.mt"]] <- PercentageFeatureSet(object = dmso2, pattern = "^Mt-")
v6<-VlnPlot(object = dmso2, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"),
ncol = 3)
plot1 <- FeatureScatter(object = dmso2, feature1 = "nCount_RNA", feature2 =
"percent.mt")
plot2 <- FeatureScatter(object = dmso2, feature1 = "nCount_RNA", feature2 =
"nFeature_RNA")
plot_grid(plot1,plot2,v6)

latent1[["percent.mt"]] <- PercentageFeatureSet(object = latent1, pattern = "^Mt-")
v7<-VlnPlot(object = latent1, features = c("nFeature_RNA", "nCount_RNA",
"percent.mt"), ncol = 3)
plot1 <- FeatureScatter(object = latent1, feature1 = "nCount_RNA", feature2 =
"percent.mt")
plot2 <- FeatureScatter(object = latent1, feature1 = "nCount_RNA", feature2 =
"nFeature_RNA")
plot_grid(plot1,plot2,v7)

latent2[["percent.mt"]] <- PercentageFeatureSet(object = latent2, pattern = "^Mt-")
v12<-VlnPlot(object = latent2, features = c("nFeature_RNA", "nCount_RNA",
"percent.mt"), ncol = 3)
plot1 <- FeatureScatter(object = latent2, feature1 = "nCount_RNA", feature2 =
"percent.mt")
plot2 <- FeatureScatter(object = latent2, feature1 = "nCount_RNA", feature2 =
"nFeature_RNA")
plot_grid(plot1,plot2,v12)

latent3[["percent.mt"]] <- PercentageFeatureSet(object = latent3, pattern = "^Mt-")
v9<-VlnPlot(object = latent3, features = c("nFeature_RNA", "nCount_RNA",
"percent.mt"), ncol = 3)
plot1 <- FeatureScatter(object = latent3, feature1 = "nCount_RNA", feature2 =
"percent.mt")
plot2 <- FeatureScatter(object = latent3, feature1 = "nCount_RNA", feature2 =
"nFeature_RNA")
plot_grid(plot1,plot2,v9)
```

Filter datasets

```
ly20$stim <- "ly20"
ly20 <- subset(x = ly20, subset = nFeature_RNA > 500 & nCount_RNA < 60000 &
percent.mt < 25)
ly20 <- NormalizeData(object = ly20, verbose = FALSE)
```

```
ly20 <- FindVariableFeatures(object = ly20, selection.method = "vst", nfeatures =
2000)

ly48$stim <- "ly48"
ly48 <- subset(x = ly48, subset = nFeature_RNA > 500 & nCount_RNA < 60000 &
percent.mt < 25)
ly48 <- NormalizeData(object = ly48, verbose = FALSE)
ly48 <- FindVariableFeatures(object = ly48, selection.method = "vst", nfeatures =
2000)

ly$stim <- "ly"
ly <- subset(x = ly, subset = nFeature_RNA > 500 & nCount_RNA < 60000 & percent.mt <
25)
ly <- NormalizeData(object = ly, verbose = FALSE)
ly <- FindVariableFeatures(object = ly, selection.method = "vst", nfeatures = 2000)

dmso1$stim <- "dmso1"
dmso1 <- subset(x = dmso1, subset = nFeature_RNA > 500 & nCount_RNA < 60000 &
percent.mt < 25)
dmso1 <- NormalizeData(object = dmso1, verbose = FALSE)
dmso1 <- FindVariableFeatures(object = dmso1, selection.method = "vst", nfeatures =
2000)

dmso2$stim <- "dmso2"
dmso2 <- subset(x = dmso2, subset = nFeature_RNA > 500 & nCount_RNA < 60000 &
percent.mt < 25)
dmso2 <- NormalizeData(object = dmso2, verbose = FALSE)
dmso2 <- FindVariableFeatures(object = dmso2, selection.method = "vst", nfeatures =
2000)

latent1$stim <- "latent1"
latent1 <- subset(x = latent1, subset = nFeature_RNA > 500 & nCount_RNA < 60000 &
percent.mt < 25)
latent1 <- NormalizeData(object = latent1, verbose = FALSE)
latent1 <- FindVariableFeatures(object = latent1, selection.method = "vst", nfeatures
= 2000)

latent2$stim <- "latent2"
latent2 <- subset(x = latent2, subset = nFeature_RNA > 500 & nCount_RNA < 60000 &
percent.mt < 25)
latent2 <- NormalizeData(object = latent2, verbose = FALSE)
latent2 <- FindVariableFeatures(object = latent2, selection.method = "vst", nfeatures
= 2000)

latent3$stim <- "latent3"
latent3 <- subset(x = latent3, subset = nFeature_RNA > 500 & nCount_RNA < 60000 &
percent.mt < 25)
latent3 <- NormalizeData(object = latent3, verbose = FALSE)
latent3 <- FindVariableFeatures(object = latent3, selection.method = "vst", nfeatures
= 2000)

### Find Integration Anchors
reactivation.anchors <- FindIntegrationAnchors(object.list = list(ly20, ly48, ly,
dmso1, dmso2, latent1, latent2, latent3), dims = 1:25)
reactivation.combined <- IntegrateData(anchorset = reactivation.anchors, dims = 1:25)
```

```
### Report number of cells present in each dataset post-filtering
table(reactivation.combined@meta.data$stim)

### Generate UMAP plot for combined dataset
DefaultAssay(object = reactivation.combined) <- "integrated"

# Run the standard workflow for visualization and clustering
reactivation.combined <- ScaleData(object = reactivation.combined, verbose = FALSE)
reactivation.combined <- RunPCA(object = reactivation.combined, npcs = 30, verbose =
FALSE)

# t-SNE and Clustering
reactivation.combined <- RunUMAP(object = reactivation.combined, reduction = "pca",
dims = 1:15)
reactivation.combined <- FindNeighbors(object = reactivation.combined, reduction =
"pca", dims = 1:15)
reactivation.combined <- FindClusters(reactivation.combined, resolution = 0.2)

# Visualization
p1 <- DimPlot(object = reactivation.combined, reduction = "umap", group.by = "stim")
p2 <- DimPlot(object = reactivation.combined, reduction = "umap", label = TRUE)
plot_grid(p1, p2)

### Generate individual UMAP plots for each dataset
DimPlot(reactivation.combined, reduction = "umap", split.by = "stim", ncol = 3)
table(Idents(reactivation.combined), reactivation.combined$stim)

### Generate plots showing expression levels of select genes across cell population
FeaturePlot(object = reactivation.combined, features = c("Prph", "Tubb3", "Sox10",
"S100b", "Aif1", "Cd68", "Col1a1", "Fn1", "Sox2", "percent.mt", "Gadd45b", "Th"),
min.cutoff = "q9", cols = c("grey", "red"))

### Assign cell type identities (based on expression patterns observed in above
plots)
reactivation.combined <- RenameIdents(object = reactivation.combined, `0` =
"neurons", `1` = "neurons", `2` = "fibroblasts", `3` = "neurons", `4` = "Satellite
glial cells", `5` = "Schwann cells", `6` = "neurons", `7` = "neurons", `8` =
"neurons", `9` = "fibroblasts", `10` = "neurons", `11` = "neurons")

DimPlot(object = reactivation.combined, label = FALSE, cols = c("blue", "tomato3",
"limegreen", "goldenrod1"), split.by="stim")

table(Idents(reactivation.combined), reactivation.combined$stim)

### Subset data to focus on neurons only
neurons2.combined <- subset(reactivation.combined, idents = "neurons")
Idents(neurons2.combined) <- "stim"

# t-SNE and Clustering
neurons2.combined <- RunTSNE(object = neurons.combined, reduction = "pca", dims =
1:15)
neurons2.combined <- FindNeighbors(object = neurons.combined, reduction = "pca", dims
= 1:15)
neurons2.combined <- FindClusters(neurons.combined, resolution = 0.2)
```

```
# Visualization
p1 <- DimPlot(object = neurons2.combined, reduction = "umap", group.by = "stim")
p2 <- DimPlot(object = neurons2.combined, reduction = " umap ", label = TRUE)
plot_grid(p1, p2)

### Generate plots showing expression levels of select genes across neuronal
population
FeaturePlot(object = neurons.combined, features = c("Npy", 'Dbh', 'Th',
"percent.mt"), min.cutoff = "q9", cols = c("grey", "blue"), ncol = 3)
```

scRNASeq-V3-analysis

```
### Load libraries
library(rlang)
library(Seurat)
library(cowplot)
library(dplyr)
library(Matrix)
library(MAST)
library(EnhancedVolcano)

### Load CellRanger Gene Expression Matrices
ly20.data <- Read10X(data.dir = "/path/to/scRNASeq-V2data/Reac20_V3_rep_1/")
ly48.data <- Read10X(data.dir = "/path/to/scRNASeq-V2data/Reac48_V3_rep_1/")
ly72.data <- Read10X(data.dir = "/path/to/scRNASeq-V2data/Reac72_V3_rep_1/")

### Setup Seurat objects and perform simple data exploration of (i) number of
features, (ii) feature counts, (iii), fraction of mitochondrial reads per cell
ly20 <- CreateSeuratObject(counts = ly20.data)
ly48 <- CreateSeuratObject(counts = ly48.data)
ly72 <- CreateSeuratObject(counts = ly72.data)

ly20[["percent.mt"]] <- PercentageFeatureSet(object = ly20, pattern = "^Mt-")
v1<-VlnPlot(object = ly20, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"),
ncol = 3)
plot1 <- FeatureScatter(object = ly20, feature1 = "nCount_RNA", feature2 =
"percent.mt")
plot2 <- FeatureScatter(object = ly20, feature1 = "nCount_RNA", feature2 =
"nFeature_RNA")
plot_grid(plot1,plot2,v1)

ly48[["percent.mt"]] <- PercentageFeatureSet(object = ly48, pattern = "^Mt-")
v2<-VlnPlot(object = ly48, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"),
ncol = 3)
plot1 <- FeatureScatter(object = ly48, feature1 = "nCount_RNA", feature2 =
"percent.mt")
plot2 <- FeatureScatter(object = ly48, feature1 = "nCount_RNA", feature2 =
"nFeature_RNA")
plot_grid(plot1,plot2,v2)

ly72[["percent.mt"]] <- PercentageFeatureSet(object = ly72, pattern = "^Mt-")
v3<-VlnPlot(object = ly72, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"),
ncol = 3)
plot1 <- FeatureScatter(object = ly72, feature1 = "nCount_RNA", feature2 =
"percent.mt")
plot2 <- FeatureScatter(object = ly72, feature1 = "nCount_RNA", feature2 =
"nFeature_RNA")
plot_grid(plot1,plot2,v3)


### Filter datasets to exclude possible doublets and cells with a higher fraction of
mitochondrial reads (>25%)
ly20$stim <- "ly20"
```

```
ly20 <- subset(x = ly20, subset = nFeature_RNA > 500 & nCount_RNA < 150000 &
percent.mt < 25)
ly20 <- NormalizeData(object = ly20, verbose = FALSE)
ly20 <- FindVariableFeatures(object = ly20, selection.method = "vst", nfeatures =
2000)

ly48$stim <- "ly48"
ly48 <- subset(x = ly48, subset = nFeature_RNA > 500 & nCount_RNA < 150000 &
percent.mt < 25)
ly48 <- NormalizeData(object = ly48, verbose = FALSE)
ly48 <- FindVariableFeatures(object = ly48, selection.method = "vst", nfeatures =
2000)

ly72$stim <- "ly72"
ly72 <- subset(x = ly72, subset = nFeature_RNA > 500 & nCount_RNA < 150000 &
percent.mt < 25)
ly72 <- NormalizeData(object = ly72, verbose = FALSE)
ly72 <- FindVariableFeatures(object = ly72, selection.method = "vst", nfeatures =
2000)

### Integrate datasets using first 20 dimensions + report number of cells in each
dataset post-filtering
reactivation.anchors <- FindIntegrationAnchors(object.list = list(ly20, ly48, ly72),
dims = 1:20) #latent
reactivation.combined <- IntegrateData(anchorset = reactivation.anchors, dims = 1:20)
table(reactivation.combined@meta.data$stim)

### Scale and cluster data + Generate UMAP plots.
DefaultAssay(object = reactivation.combined) <- "integrated"
reactivation.combined <- ScaleData(object = reactivation.combined, verbose = FALSE)
reactivation.combined <- RunPCA(object = reactivation.combined, npcs = 30, verbose =
FALSE)
reactivation.combined <- RunUMAP(object = reactivation.combined, reduction = "pca",
dims = 1:25)
reactivation.combined <- FindNeighbors(object = reactivation.combined, reduction =
"pca", dims = 1:25)
reactivation.combined <- FindClusters(reactivation.combined, resolution = 0.2)
p1 <- DimPlot(object = reactivation.combined, reduction = "umap", group.by = "stim")
p2 <- DimPlot(object = reactivation.combined, reduction = "umap", label = TRUE)
plot_grid(p1, p2)

### Separate UMAP plots by dataset & determine number of cells per identity
DimPlot(reactivation.combined, reduction = "umap", split.by = "stim", ncol = 3)
table(Idents(reactivation.combined), reactivation.combined$stim)

### Generate plots for canonical markers to aid in cell type identification
# Sympathetic neurons: Prph, Tubb3, Snap25
# Schwann cells: Sox10, S100b
# Satellite glial cells: Aif1, Cd68
# Fibroblasts: Fn1, Col3a1, Col1a1
FeaturePlot(object = reactivation.combined, features = c("Prph", "Tubb3", "Snap25",
"Sox10", "S100b", "nes", "Aif1", "Cd68", "Col3a1", "Fn1", "Col1a1", "percent.mt"),
min.cutoff = "q9") #"nCount_RNA", "nFeature_RNA" percent.mt
```

```
### Generate plots for viral RNA markers to identify cells/clusters with reactivating
virus
FeaturePlot(object = reactivation.combined, features = c("TU1", "TU2", "TU3", "TU4",
"TU5", "TU6", "TU7", "TU8", "TU9", "TU10", "TU11", "TU12", "TU13", "TU14", "TU15",
"TU16", "TU17", "TU18", "TU19", "TU20"), min.cutoff = "q9")

### Merge similar clusters and label with cell identity + generate counts of each
cell type per condition
reactivation.combined <- RenameIdents(object = reactivation.combined, `0` =
"neurons", `1` = "Schwann cell", `2` = "Fibroblasts", `3` = "Schwann cell", `4` =
"neurons", `5` = "Schwann cell", `6` = "Schwann cell", `7` = "Schwann cell", `8` =
"neurons")
DimPlot(object = reactivation.combined, label = FALSE)
table(Idents(reactivation.combined), reactivation.combined$stim)

### Subset data to focus on neurons only
neurons.combined <- subset(reactivation.combined, idents = "neurons")
Idents(neurons.combined) <- "stim"

# t-SNE and Clustering
neurons.combined <- RunTSNE(object = neurons.combined, reduction = "pca", dims =
1:20)
neurons.combined <- FindNeighbors(object = neurons.combined, reduction = "pca", dims
= 1:20)
neurons.combined <- FindClusters(neurons.combined, resolution = 0.2)

# Visualization
p1 <- DimPlot(object = neurons.combined, reduction = "umap", group.by = "stim") #umap
rather than tsne?
p2 <- DimPlot(object = neurons.combined, reduction = "umap", label = TRUE)
plot_grid(p1, p2)

### Check whether HSV-1 RNAs are enriched in one or more neuronal subsets
FeaturePlot(object = neurons.combined, features = c("TU1", "TU2", "TU3", "TU4",
"TU5", "TU6", "TU7", "TU8", "TU9", "TU10", "TU11", "TU12", "TU13", "TU14", "TU15",
"TU16", "TU17", "TU18", "TU19", "TU20"), min.cutoff = "q9")

### Label neurons according to expression level of HSV-1 transcripts
neurons.combined <- RenameIdents(object = neurons.combined, `0` = "neurons (low/no
HSV-1 expression)", `1` = "neurons (low/no HSV-1 expression)", `2` = "neurons (low/no
HSV-1 expression)", `3` = "neurons (low/no HSV-1 expression)", `4` = "neurons (low/no
HSV-1 expression)", `5` = "neurons (high HSV-1 expression)")
DimPlot(object = neurons.combined, label = FALSE)

### Perform DGE between clusters using MAST
dds <- FindMarkers(neurons.combined, ident.1 = "neurons (high HSV-1 expression)",
ident.2 = "neurons (low/no HSV-1 expression)", test.use="MAST")

### Generate Volcano Plot, and export data.
EnhancedVolcano(dds, lab = rownames(dds), x = 'avg_logFC', y = 'p_val_adj',
ylim=c(0,350), xlim = c(-3, 8), title = 'Regulated during HSV-1 reactivation',
pCutoff = 0.05, FCcutoff = 1.5, transcriptLabSize = 4.0,
col=c('darkgrey','darkgrey','darkgrey','darkred'), transcriptPointSize = 1.5,
cutoffLineCol = 'blue', colAlpha = 0.9)
```

```
EnhancedVolcano(dds, lab = rownames(dds), x = 'avg_logFC', y = 'p_val_adj',
ylim=c(0,350), xlim = c(-3, 8), pCutoff = 0.05, FCcutoff = 1.5, transcriptLabSize =
4.0, col=c('darkgrey','darkgrey','darkgrey','darkred'), transcriptPointSize = 1.5,
cutoffLineCol = 'blue', colAlpha = 0.9, drawConnectors = TRUE, widthConnectors = 0.2,
colConnectors = "grey")
EnhancedVolcano(dds, lab = rownames(dds), x = 'avg_logFC', y = 'p_val_adj',
ylim=c(0,350), xlim = c(-3, 8), pCutoff = 0.05, FCcutoff = 1.5,
col=c('darkgrey','darkgrey','darkgrey','darkred'), transcriptPointSize = 1.5,
cutoffLineCol = 'blue', colAlpha = 0.9, selectLab = "TU3")
write.table(dds,file="MAST-DGE.csv")


### Generate heatmap comparing clusters
# Set up list of viral transcription units to plot
VirusList <- c("TU1", "TU2", "TU3", "TU4", "TU5", "TU6", "TU7", "TU8", "TU9", "TU10",
"TU11", "TU12", "TU14", "TU15", "TU16", "TU17", "TU18", "TU19", "TU20", "US1",
"US12", "RS1", "RL1", "RL2")
# Set up list of significantly upregulated cellular genes to plot
HostUpList <-
c("AABR07030834.1","Cdkn1c","Gadd45b","Gadd45g","Galns","Hspa1b","Hspa2","Idi1","Igf2
","Ing1","Ldah","LOC690422","Mgp","Nefh","Nefm","Nusap1","RGD1562673","Srsf2","Tchh",
"Zdbf2")
# Set up list of randomly sampled cellular genes that are not differentially
expressed
HostNoChangeList <-
c("Bag3","Chp1","Ddit3","Fdft1","Gadd45a","Ngb","Rgs17","S100a1","Serpinb1a","Vip")
# Set up list of significantly downregulated cellular genes to plot
HostDownList <- c("Basp1", "Hspb1", "Mmd")
# Randomly subsample 100 cells from each  population (to make plot manageable)
sample<-subset(neurons.combined, downsample=100)

### Generated plots
DoHeatmap(sample, features = VirusList, size = 3) + scale_fill_gradientn(colors =
rev(RColorBrewer::brewer.pal(n = 7, name = "RdBu")))
DoHeatmap(sample, features = HostUpList, size = 3) + scale_fill_gradientn(colors =
rev(RColorBrewer::brewer.pal(n = 7, name = "RdBu")))
DoHeatmap(sample, features = HostDownList, size = 3) + scale_fill_gradientn(colors =
rev(RColorBrewer::brewer.pal(n = 7, name = "RdBu")))
DoHeatmap(sample, features = HostNoChangeList, size = 3) +
scale_fill_gradientn(colors = rev(RColorBrewer::brewer.pal(n = 7, name = "RdBu")))
```