smFISH analysis

Lorena Garcia-Perez (lorena.garcia-perez@crick.ac.uk)

01 August, 2018, 18:04

Main R script to determine: RNA copy number per cell RNA density number per cell Transcriptional properties Based on the output .csv table obtained with the Python script

Load libraries

```
library(ggplot2)
library (ggbeeswarm)
require(plyr) # Am I using this in the end?
library(tidyverse)
# library(tibble) #for add_column etc
library(knitr)
library(rmarkdown)
library(pander)
```

```
opts_chunk$set(panderOptions('table.split.table', 80),
               panderOptions('table.alignment.default', 'left'),
               panderOptions('table.alignment.rownames', 'right'))
```

Read data and calculate RNA copy number and concentration (== density)

Another option is to execute the following: args = c("Diff26_c2_data.csv", "26_c2_copy_den_plots.pdf", "26_c2_TX_plots.pdf", "26_c2_heter_plots.pdf", "26_c2_copyN.csv", "26_c2_TX.csv", "26_c2_heter.csv")
#
#The first argument is the data to be read,
#arguments 2-4 are the names of the pdf files to be created containing the plots,
#arguments 5-7 are the names of the csv files to be created cotaining the extracted data

```
wd <- getwd()
setwd("..")
parent <- getwd()

file <- read.csv("./input/Diff26_c2_data.csv")

setwd(wd)

# To remove any row where the area = 0 due to clicking on a point during cell segmentation
file <- file[file$cell.area != 0, ]
pander(head(file), style = "rmarkdown")
```

Table 1: Table continues below

channel	day	image	cell.id	cell.area	spot.id	x	y
0	4	10	148	3899	1	226	258
0	4	10	148	3899	2	223.5	258
0	4	10	148	3899	3	224	256
0	4	10	148	3899	4	234	255.3
0	4	10	148	3899	5	228	255
0	4	10	148	3899	6	220	254

area	max.int	mean.int
1	1947	1947
2	1979	1968
1	2030	2030
3	2421	2212
1	2049	2049
1	2369	2369

```
# Colors
cbbPalette <- c("#000000", "#009E73", "#e79f00", "#9ad0f3", "#0072B2", "#D55E00",
                "#CC79A7", "#F0E442")

days <- c(4,5,6)
days.names <- c("day 4", "day 5", "day 6")

channels <- c(0, 1, 2)
genes <- c("Irx3","Sox2","Olig2")

progenitors.gene <- genes[2]

# For spot selection as TX spot
MaxInt.alpha <- 2 # This value should lay between 1.5 to 3
area.alpha <- 2 # This value should lay between 1.5 to 3


# With 60X objective in BX61 microscope and previous ANDOR camera:
# Pixel size = 0.1107x0.1107 µm^2 = 0.01225 µm^2
pixel.size <- 0.01225


ploidy <- 3

#Correction factor that takes into account
#probe localization over gene:
#Irx3 reverse starnd transcribed, Sox2 and Olig2
correction.factor <- c(0.34, 0.5, 0.34)


genes.length <- c(4822, 3311, 4400) # channel 0, 1, 2 (Irx3, Sox2, Olig2)
vPol.chosen = 2040 # bp/min, as in Bahar Halpern
```

```
copy.density.cell <-
  file %>%
  mutate(cell.area = cell.area * pixel.size) %>%
    # Convert cell area in pixels to cell area in µm^2
  group_by(channel, day, image, cell.id, cell.area) %>%
  summarise(copy = n()) %>% # RNA copy number per cell
  mutate(density = round(copy / cell.area, 3))
    # RNA densities in terms of RNA molecules per µm^2
pander(head(copy.density.cell), style = "rmarkdown")
```

channel	day	image	cell.id	cell.area	copy	density
0	4	3	36	24.9	7	0.281

channel	day	image	cell.id	cell.area	copy	density
0	4	3	37	30.26	15	0.496
0	4	3	38	19.33	14	0.724
0	4	3	39	37.71	42	1.114
0	4	3	40	39.62	32	0.808
0	4	3	41	32.89	13	0.395

To determine which spots can be considered transcription spots, we need to know the area and maximum intensity values for all the spots we normalize according to each channel, day and image

```
TX.threholds <-
  file %>%
  group_by(channel, day, image) %>%
    # Choose threholds specifically for each image and channel
  summarise(MaxInt.median = median(max.int), MaxInt.mad = mad(max.int),
            area.median = median(area), area.mad = mad(area)) %>%
  mutate(MaxInt.thres = MaxInt.median + 3 * MaxInt.alpha * MaxInt.mad,
         area.thres = area.median + 3 * area.alpha * area.mad)
pander(head(TX.threholds), style = "rmarkdown")
```

Table 4: Table continues below

channel	day	image	MaxInt.median	MaxInt.mad	area.median
0	4	3	1894	282.4	2
0	4	4	1547	198.7	2
0	4	5	1755	296.5	2
0	4	6	1727	209	2
0	4	7	1454	222.4	2
0	4	9	1628	247.6	2

area.mad	MaxInt.thres	area.thres
1.483	3588	10.9
1.483	2739	10.9
1.483	3534	10.9
1.483	2981	10.9
1.483	2788	10.9
1.483	3114	10.9

```
TX.number.cell <-
  file %>%
  group_by(channel, day, image) %>%
  left_join(TX.threholds) %>%
  mutate(is.TX.spot = max.int > MaxInt.thres & area > area.thres) %>%
  group_by(channel, day, image, cell.id) %>%
  summarise(TX.number = sum(is.TX.spot))
```

```
## Joining, by = c("channel", "day", "image")
```

```
pander(head(TX.number.cell), style = "rmarkdown")
```

channel	day	image	cell.id	TX.number
0	4	3	36	0
0	4	3	37	0
0	4	3	38	0
0	4	3	39	0
0	4	3	40	0
0	4	3	41	0

```
TX.mol <-
  file %>%
  group_by(channel, day, image) %>%
  left_join(TX.threholds) %>%
  mutate(is.TX.spot = max.int > MaxInt.thres & area > area.thres) %>%
  mutate(meanInt.median = median(mean.int)) %>%
  filter(is.TX.spot == TRUE) %>%
  mutate(probe.loc = channel) %>%
  mutate(probe.loc = as.numeric(recode(probe.loc,
                            "0"=correction.factor[1],
                            "1"=correction.factor[2],
                            "2"=correction.factor[3]))) %>%
  mutate(TX.mol = mean.int * area / (probe.loc * meanInt.median * area.median))
```

```
## Joining, by = c("channel", "day", "image")
```

```
TX.mol$channel <- as.factor(TX.mol$channel)
levels(TX.mol$channel) <- genes
pander(head(TX.mol), style = "rmarkdown")
```

Table 7: Table continues below

channel	day	image	cell.id	cell.area	spot.id	x	y
Irx3	5	0	1287	2372	28	195.2	316.4
Irx3	5	0	1314	1584	8	258	108.2
Irx3	5	0	1316	1436	25	140.5	107.5
Irx3	5	1	1338	1401	11	258.5	305.5
Irx3	5	1	1327	2463	19	137.3	327.3
Irx3	5	2	1415	1920	25	383.2	101.6

Table 8: Table continues below

area	max.int	mean.int	MaxInt.median	MaxInt.mad	area.median
5	4837	3606	2168	329.9	1
4	4412	3444	2168	329.9	1
4	5530	5109	2168	329.9	1
4	3615	3212	1740	266.9	1
3	3410	2680	1740	266.9	1
5	4024	3125	1826	344	1

Table 9: Table continues below

area.mad	MaxInt.thres	area.thres	is.TX.spot	meanInt.median
0	4147	1	TRUE	2145
0	4147	1	TRUE	2145
0	4147	1	TRUE	2145
0	3341	1	TRUE	1710
0	3341	1	TRUE	1710
0	3890	1	TRUE	1810

probe.loc	TX.mol
0.34	24.72
0.34	18.89
0.34	28.02
0.34	22.09
0.34	13.82
0.34	25.4

2. Main datasets for single cell data:

```
single.cell.data <-
  inner_join(copy.density.cell, TX.number.cell) %>%
  ungroup()
```

```
## Joining, by = c("channel", "day", "image", "cell.id")
```

```
single.cell.data$channel <- as.factor(single.cell.data$channel)
levels(single.cell.data$channel) <- genes
pander(head(single.cell.data), style = "rmarkdown")
```

channel	day	image	cell.id	cell.area	copy	density	TX.number
Irx3	4	3	36	24.9	7	0.281	0
Irx3	4	3	37	30.26	15	0.496	0
Irx3	4	3	38	19.33	14	0.724	0
Irx3	4	3	39	37.71	42	1.114	0
Irx3	4	3	40	39.62	32	0.808	0
Irx3	4	3	41	32.89	13	0.395	0

```
single.cell.expressing.data <-
  single.cell.data %>%
  filter(density >= 1)
pander(head(single.cell.expressing.data), style = "rmarkdown")
```

channel	day	image	cell.id	cell.area	copy	density	TX.number
Irx3	4	3	39	37.71	42	1.114	0
Irx3	4	3	42	26.88	46	1.712	0
Irx3	4	3	46	27.8	58	2.087	0
Irx3	4	3	49	25	25	1	0
Irx3	4	3	51	24.92	33	1.324	0

channel	day	image	cell.id	cell.area	copy	density	TX.number
Irx3	4	3	82	29.67	34	1.146	0

```
single.cell.sox2.expressing.data <-
  # This could be combined within single.cell.progenitors.data with filter_if()?
  single.cell.data %>%
  filter(channel == progenitors.gene) %>%
  filter(density >= 1) # Require sox2 density >= 1 molecule / µm^2
pander(head(single.cell.sox2.expressing.data), style = "rmarkdown")
```

channel	day	image	cell.id	cell.area	copy	density	TX.number
Sox2	4	3	36	24.9	61	2.449	0
Sox2	4	3	37	30.26	76	2.512	0
Sox2	4	3	38	19.33	48	2.483	0
Sox2	4	3	39	37.71	81	2.148	0
Sox2	4	3	40	39.62	94	2.373	0
Sox2	4	3	41	32.89	66	2.007	0

```
single.cell.progenitors.data <-
  # Select progenitors according to sox2 density >= 1 molecule / µm^2
  semi_join(single.cell.data, single.cell.sox2.expressing.data,
            by = c("day", "image", "cell.id", "cell.area"))
pander(head(single.cell.progenitors.data), style = "rmarkdown")
```

channel	day	image	cell.id	cell.area	copy	density	TX.number
Irx3	4	3	36	24.9	7	0.281	0
Irx3	4	3	37	30.26	15	0.496	0
Irx3	4	3	38	19.33	14	0.724	0
Irx3	4	3	39	37.71	42	1.114	0
Irx3	4	3	40	39.62	32	0.808	0
Irx3	4	3	41	32.89	13	0.395	0

```
# This is the more complete, non subseted, dataset
# But, if a cell has two TX spots, it will appear on two rows and so on
withTX.single.cell.data <-
  TX.mol %>%
  select (channel, day, image, cell.id, spot.id, TX.mol) %>%
  full_join(single.cell.data) %>%
  arrange(channel, day, image, cell.id) %>%
  replace_na(list(TX.mol = 0))
```

```
## Joining, by = c("channel", "day", "image", "cell.id")
```

```
pander(head(withTX.single.cell.data), style = "rmarkdown")
```

Table 15: Table continues below

channel	day	image	cell.id	spot.id	TX.mol	cell.area	copy
Irx3	4	3	36	NA	0	24.9	7
Irx3	4	3	37	NA	0	30.26	15

channel	day	image	cell.id	spot.id	TX.mol	cell.area	copy
Irx3	4	3	38	NA	0	19.33	14
Irx3	4	3	39	NA	0	37.71	42
Irx3	4	3	40	NA	0	39.62	32
Irx3	4	3	41	NA	0	32.89	13

density	TX.number
0.281	0
0.496	0
0.724	0
1.114	0
0.808	0
0.395	0

```
two.alleles.data <-
  TX.mol %>%
  select(channel, day, image, cell.id, TX.mol) %>%
  group_by(channel, day, image, cell.id) %>%
  mutate(freq = n()) %>%
  ungroup() %>%
  filter(freq == 2) %>%
  select(-freq) %>%
  mutate(TX.id = rep(c(1,2), length.out = length(.$channel))) %>%
    # Create a column with appropiate number of rows
  spread(TX.id, TX.mol) %>%
  rename(TX.id.1 = "1", TX.id.2 = "2")
two.alleles.data$channel <- as.factor(two.alleles.data$channel )
levels(two.alleles.data$channel) <- genes
pander(head(two.alleles.data), style = "rmarkdown")
```

channel	day	image	cell.id	TX.id.1	TX.id.2
Irx3	5	8	1171	32.79	23.44
Irx3	5	9	1207	28.36	22.51
Irx3	6	9	2083	29.24	26.39
Sox2	4	5	136	22.51	27.95
Sox2	4	5	146	17.37	31.55
Sox2	4	5	152	18.81	11.57

Mean properties

single.cell.progenitors.data is used, but results with single.cell.data are quite similar

Trancription probability

```
prob.TX.fn <- function(data, day.chosen, gene.chosen, ploidy) {
  # day.chosen and gene.schosen must be input as character (for example, "4"
  # and "Olig2")
  # ploidy must be input as numeric
  day.gene.TX.subset <-
```

```
    pull(filter(data, day == day.chosen & channel == gene.chosen), TX.number)
  return(sum(day.gene.TX.subset)/(ploidy*length(day.gene.TX.subset)))
}
```

Mean RNA copy number

```
mean.copy.fn <- function(data, day.chosen, gene.chosen) {
  # day.chosen and gene.schosen must be input as character (for example, "4"
  # and "Olig2")
  day.gene.copy.subset <-
    pull(filter(data, day == day.chosen & channel == gene.chosen), copy)
  return(mean(day.gene.copy.subset))
}
```

Mean number of RNA pol = mean number of molecules at TX spot

```
mean.nPol.fn <- function(data, day.chosen, gene.chosen) {
  # day.chosen and gene.schosen must be input as character (for example, "4"
  # and "Olig2")
  day.gene.nPol.subset <-
    pull(filter(data, day == day.chosen & channel == gene.chosen), TX.mol)
  return(mean(day.gene.nPol.subset))
}
```

Chose white or dark themes as part of the plotting funtion inputs

```
basicTheme <-list(
  theme(plot.title = element_text(size = 15, hjust = 0.5, face = "bold",
                                  margin = margin(t = 0, r = 0, b = 30, l = 0)),
        axis.text = element_text(size = 15),
        axis.title = element_text(size = 15),
        axis.title.x =
          element_text(margin = margin(t = 20, r = 0, b = 0, l = 0)),
        axis.title.y =
          element_text(margin = margin(t = 0, r = 20, b = 0, l = 0)),
        axis.line = element_line(size = 1.5),
        axis.ticks = element_line(size = 1.5),
        axis.ticks.length = unit(0.15,"cm"),
        strip.text.x = element_text(size = 15, face = "bold"),
        panel.spacing = unit(1, "lines"),
        panel.background = element_rect(fill = "transparent"),
        plot.background = element_rect(fill = "transparent", colour = NA),
        legend.background = element_rect(fill = "transparent"),
        legend.box.background = element_rect(fill = "transparent"))
)

whiteTheme <- list(theme())

darkTheme <- list(
  theme(axis.text = element_text(colour = "lightgrey"),
        axis.title = element_text(colour = "lightgrey"),
        axis.line = element_line(colour = "lightgrey"),
        axis.ticks = element_line(colour = "lightgrey"),
        strip.text.x = element_text(colour = "lightgrey"))
)
```

RNA copy number and RNA concentration (== density) plots

1. Box plots

```
copy.box.plot <-
  function(data, Theme = whiteTheme) {
    data$day <- as.factor(data$day)
    levels(data$day) <- days.names
    plots <-
      ggplot(data, aes(day, copy, group=day)) +
        geom_boxplot(outlier.shape = NA, fill = NA,
                     colour = cbbPalette[2], lwd = 1) +
        geom_quasirandom(shape = 21, fill = "lightgrey", colour = "black",
                         size = 0.75, alpha = 0.4) +
        ylab("RNAs per cell") +
        xlab("") +
        ylim(0, max(data$copy)) +
        labs (title = "RNA copy number") +
        theme_classic() +
        basicTheme +
        Theme +
        theme(plot.title=element_text(colour = cbbPalette[2]),
              strip.background = element_rect(fill = cbbPalette[2],
                                              colour = cbbPalette[2])) +
        facet_wrap(~ channel)
    ggsave(paste("./figures/copy_box_plot_",
                 deparse(substitute(Theme)), ".pdf", sep=""),
           plots, width = 10, height = 5)
    print(plots)
  }

concentration.box.plot <-
  function(data, Theme = whiteTheme) {
    data$day <- as.factor(data$day)
    levels(data$day) <- days.names
    plots <-
      ggplot(data, aes(day, density, group=day)) +
      geom_boxplot(outlier.shape = NA, fill = NA,
                   colour = cbbPalette[3], lwd = 1) +
      geom_quasirandom(shape = 21, fill = "lightgrey", colour = "black",
                       size = 0.75, alpha = 0.4) +
      ylab(bquote("RNAs/" ~  mu*m^2)) +
      xlab("") +
      ylim(0, max(data$density)) +
      labs (title = "RNA concentration") +
      theme_classic() +
      basicTheme +
      Theme +
      theme(plot.title=element_text(colour = cbbPalette[3]),
            strip.background = element_rect(fill = cbbPalette[3],
                                            colour = cbbPalette[3])) +
      facet_wrap(~ channel)
    ggsave(paste("./figures/concentration_box_plot_",
                 deparse(substitute(Theme)), ".pdf", sep=""),
           plots, width = 10, height = 5)
```

```
  print(plots)
}
```

2. Scatter and density contours plots:

channel.1 and channel.2 must be supplied as strings
data must have channel as factor

```
copy.pairwise.scatter.plot <-
  # Could try to make axis limit specific for pair of genes being plot each time
  function(data, channel.1, channel.2, Theme = whiteTheme) {
    data.subset <-
      data %>%
      select(channel, day, image, cell.id, cell.area, copy) %>%
      spread(channel, copy)
    plots <-
      ggplot(data.subset, aes_string(channel.2, channel.1)) +
        geom_point(shape = 21, fill = "lightgrey", colour = "black",
                   size = 1.5, alpha = 0.4) +
        geom_density_2d(size = 0.5, colour = cbbPalette[2], bins = 10) +
        labs (title = "RNA copy number") +
        scale_x_continuous(name = bquote(.(channel.2) ~ " RNAs per cell"),
                           breaks = seq(0, max(data$copy), 50),
                           limits = c(0, max(data$copy))) +
        scale_y_continuous(name = bquote(.(channel.1) ~ " RNAs per cell"),
                           breaks = seq(0, max(data$copy), 50),
                           limits = c(0, max(data$copy))) +
        theme_classic() +
        basicTheme +
        Theme +
        theme(plot.title=element_text(colour = cbbPalette[2]),
              strip.background = element_rect(fill = cbbPalette[2],
                                              colour = cbbPalette[2])) +
        facet_wrap(~ day)
    ggsave(paste("./figures/copy_pairwise_scatter_plot_",
                 deparse(substitute(Theme)),"_",
                 channel.2,"_",channel.1, ".pdf", sep=""),
           plots, width = 10, height = 5)
    print(plots)
  }

concentration.pairwise.scatter.plot <-
  # Could try to make axis limit specific for pair of genes being plot each time
    function(data, channel.1, channel.2, Theme = whiteTheme) {
    data.subset <-
      data %>%
      select(channel, day, image, cell.id, cell.area, density) %>%
      spread(channel, density)
    plots <-
      ggplot(data.subset, aes_string(channel.2, channel.1)) +
      geom_point(shape = 21, fill = "lightgrey", colour = "black",
                 size = 1.5, alpha = 0.4) +
      geom_density_2d(size = 0.5, colour = cbbPalette[3], bins = 10) +
      labs (title = "RNA cconcentration") +
      scale_x_continuous(name = bquote(.(channel.2) ~ " RNAs/" ~  mu*m^2),
```

```
                          breaks = seq(0, max(data$density), 2),
                          limits = c(0, max(data$density))) +
      scale_y_continuous(name = bquote(.(channel.1) ~ " RNAs/" ~  mu*m^2),
                          breaks = seq(0, max(data$density), 2),
                          limits = c(0, max(data$density))) +
      theme_classic() +
      basicTheme +
      Theme +
      theme(plot.title=element_text(colour = cbbPalette[3]),
            strip.background = element_rect(fill = cbbPalette[3],
                                            colour = cbbPalette[3])) +
      facet_wrap(~ day)
    ggsave(paste("./figures/concentration_pairwise_scatter_plot_",
                 deparse(substitute(Theme)),"_",
                 channel.2,"_",channel.1,".pdf", sep=""),
           plots, width = 10, height = 5)
    print(plots)
  }
```

```
mean.properties.data <-
  tibble(day = character(), gene = character(), prob.TX = as.numeric(),
         mean.copy = as.numeric(), mean.nPol = as.numeric())

for (d in days) {
  for (g in genes) {
    mean.properties.data <-
      add_row(mean.properties.data,
              day = d, gene = g,
              prob.TX =
                round(prob.TX.fn(single.cell.progenitors.data, d, g, ploidy),3),
              mean.copy =
                round(mean.copy.fn(single.cell.progenitors.data, d, g),3),
              mean.nPol =
                round(mean.nPol.fn(TX.mol, d, g),3))
  }
}
mean.properties.data <- mean.properties.data %>% replace_na(list(mean.nPol = 0))

mean.properties.data <-
  mean.properties.data %>%
  mutate(vPol = vPol.chosen, gene.length = rep(genes.length,3),
         ploidy = ploidy) %>%
  mutate(TX.rate = mean.nPol * vPol / gene.length) %>%
  mutate(deg.rate = prob.TX * TX.rate * ploidy / mean.copy) %>%
  mutate(half.life = log(2)/deg.rate) #in min
pander(head(mean.properties.data), style = "rmarkdown")
```

Table 18: Table continues below

day	gene	prob.TX	mean.copy	mean.nPol	vPol	gene.length
4	Irx3	0	25.17	0	2040	4822
4	Sox2	0.041	58.19	18.89	2040	3311
4	Olig2	0.008	7.899	38.7	2040	4400
5	Irx3	0.015	25.28	23.27	2040	4822

day	gene	prob.TX	mean.copy	mean.nPol	vPol	gene.length
5	Sox2	0.04	38.23	16.76	2040	3311
5	Olig2	0.02	12.21	33.23	2040	4400

ploidy	TX.rate	deg.rate	half.life
3	0	0	Inf
3	11.64	0.0246	28.17
3	17.94	0.05451	12.72
3	9.846	0.01753	39.55
3	10.32	0.03241	21.39
3	15.4	0.07567	9.161

4.

```
prob.TX.pairs.data <-
  mean.properties.data %>%
  select(day, gene, prob.TX) %>%
  rename(gene.1 = gene, prob.TX.1 = prob.TX) %>%
  mutate(gene.2 = gene.1) %>%
  expand(nesting(day, gene.1, prob.TX.1), gene.2) %>%
  group_by(day) %>%
  mutate(prob.TX.2 = rep(unique(prob.TX.1), times = 3)) %>%
  filter(gene.1 < gene.2) # To get unique combinations
pander(head(prob.TX.pairs.data), style = "rmarkdown")
```

day	gene.1	prob.TX.1	gene.2	prob.TX.2
4	Irx3	0	Olig2	0.008
4	Irx3	0	Sox2	0.041
4	Olig2	0.008	Sox2	0.041
5	Irx3	0.015	Olig2	0.02
5	Irx3	0.015	Sox2	0.04
5	Olig2	0.02	Sox2	0.04

TO BE FINISHED...

for (d in days) { for (g in genes) { prob.TX.pairs.data$prob.TX.both <- add_row(prob.TX.pairs.data$prob.TX.both, prob.TX.pairs.fn())

```
prob.TX.pairs.fn <-
  function(data, day.chosen, gene.chosen.1, gene.chosen.2,ploidy) {
  # day.chosen and gene.schosen must be input as character (for example, "4"
  # and "Olig2")
  # ploidy must be input as numeric
  day.gene.TX.subset <-
    filter(data, day == day.chosen &
                 (channel == gene.chosen.1 | channel == gene.chosen.2))
  dual.TX <-
    day.gene.TX.subset %>%
```

```
    select(channel, image, day, cell.id, TX.number) %>%
    filter (TX.number != 0) %>%
    group_by(day, image, cell.id) %>%
    mutate(freq = n()) %>%
    ungroup() %>%
    filter(freq == 2) %>%
    select(-freq) %>%
    spread(channel, TX.number) %>%

  return(nrow(dual.TX)/(nrow(day.gene.TX.subset)))
    # Here I do not make use of the ploidy concept...
}
```

two.alleles.data$channel <- as.factor(two.alleles.data$channel) levels(two.alleles.data$channel) <- genes

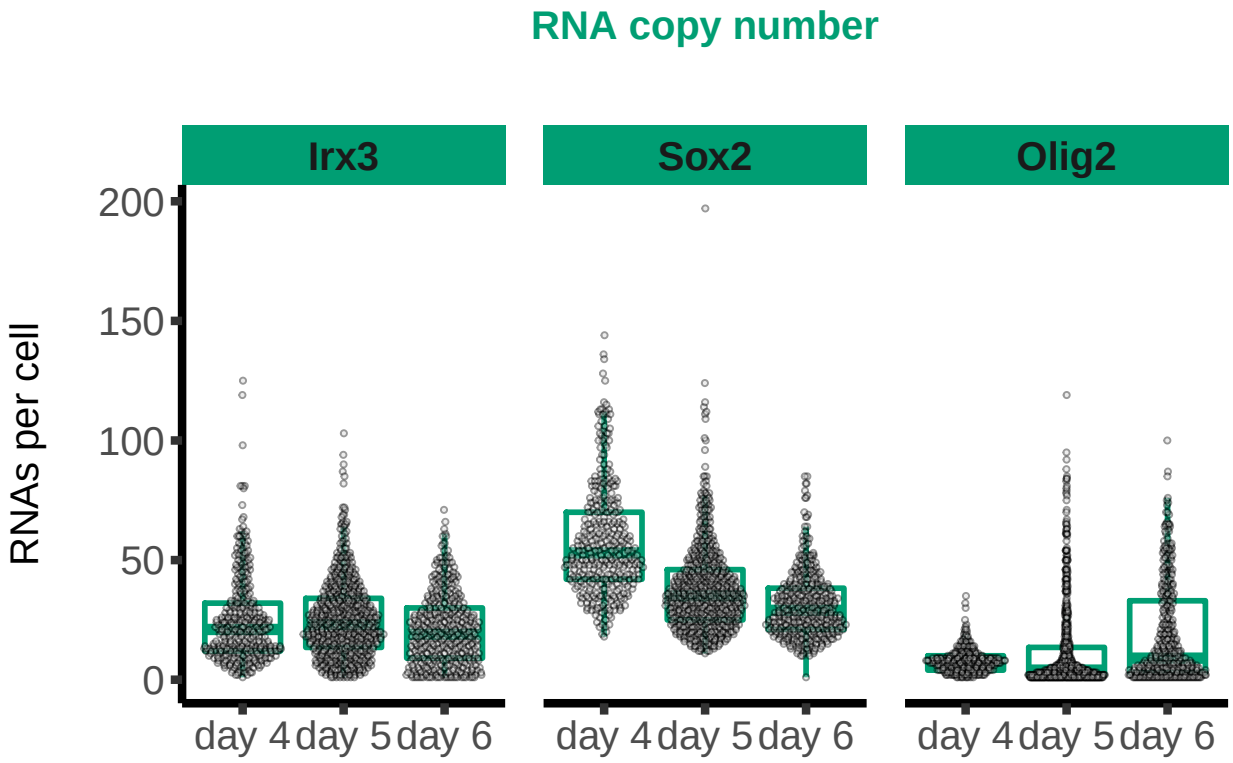
TX.probability.all[,6] <- TX.probability.all$TX.prob.pair/(TX.probability.all$TX.prob.oneC * TX.probability.all$TX.prob.oth
colnames(TX.probability.all)[6] <- c("extrinsic.noise")

To measure the influence of extrinsic noise:

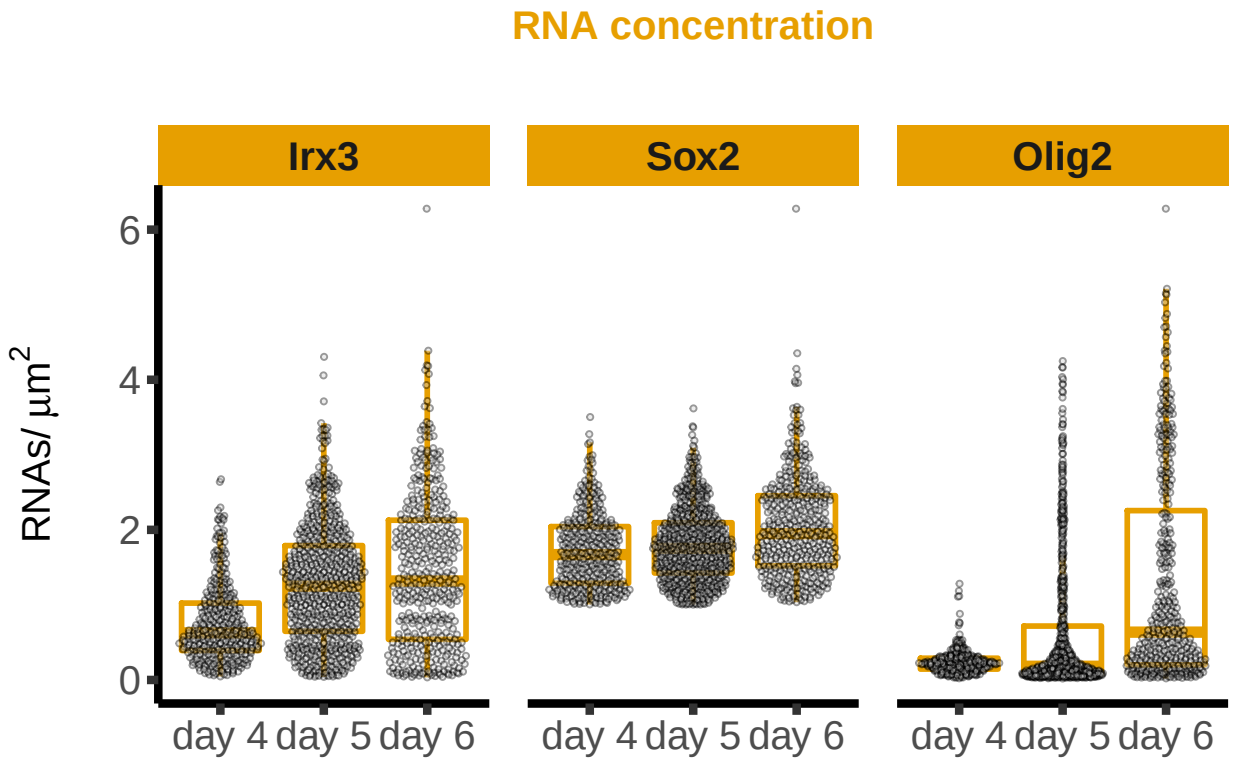
THIS SHOULD BE DONE WITH A CONSTITUTIVE GENE TO COMPARE!!!

```
genes.pairwise <-
  tibble(genes) %>%
  rename(gene.1 = genes) %>%
  mutate(gene.2 = gene.1) %>%
  expand(gene.1, gene.2) %>%
  filter(gene.1 < gene.2) # To get unique combinations
```

```
setwd(parent)

copy.box.plot(single.cell.progenitors.data)
```

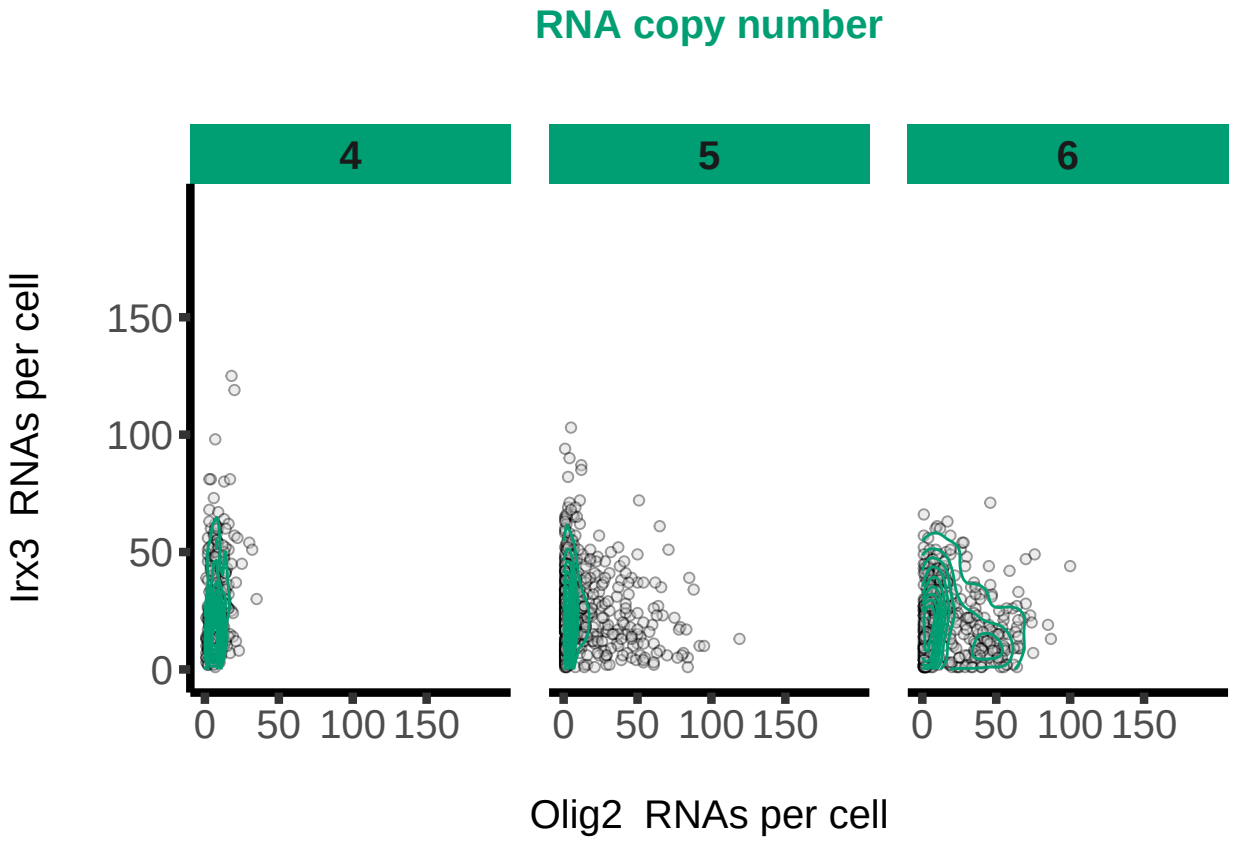


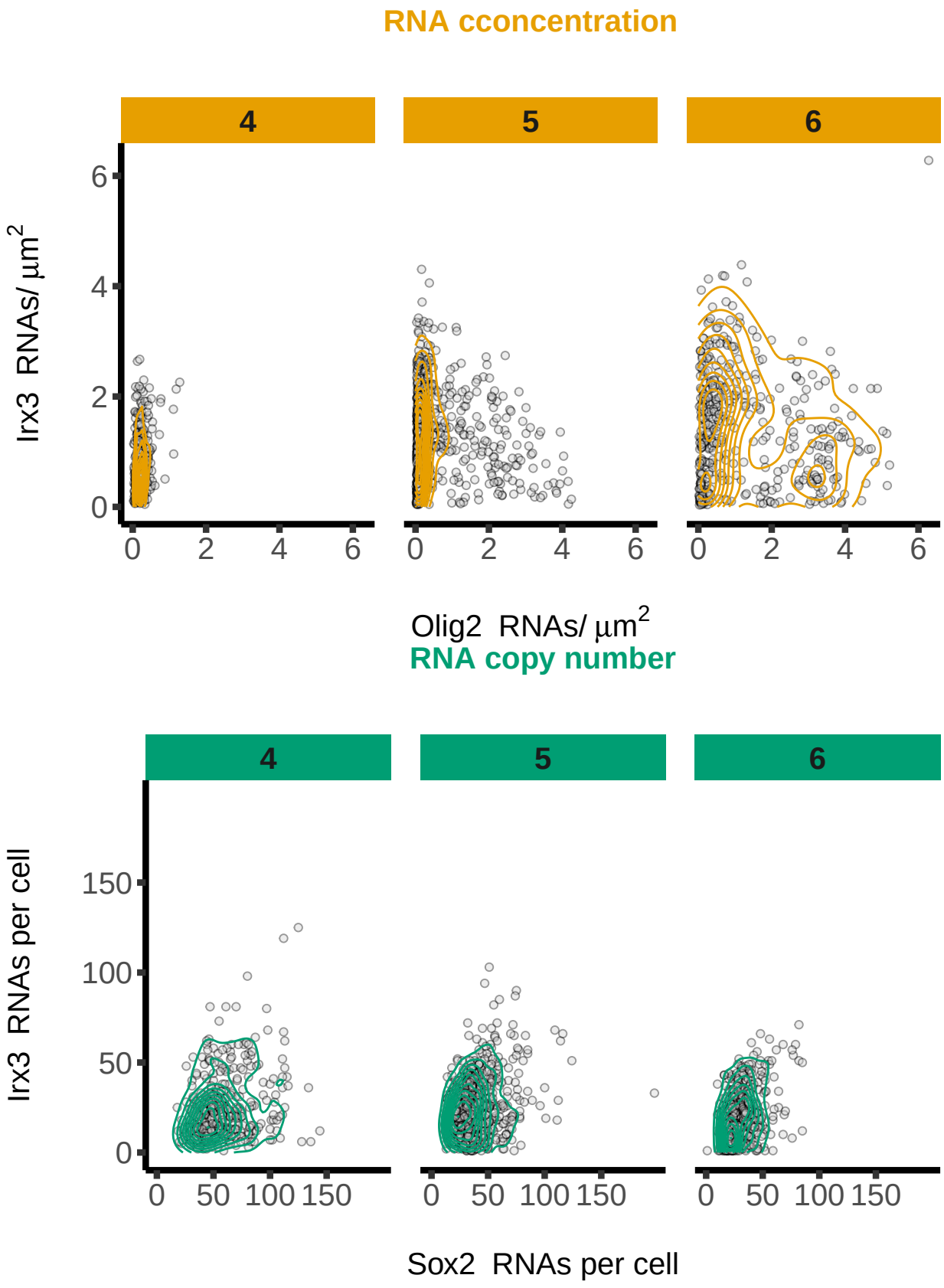

```
#copy.box.plot(single.cell.progenitors.data, darkTheme)
concentration.box.plot(single.cell.progenitors.data)
```

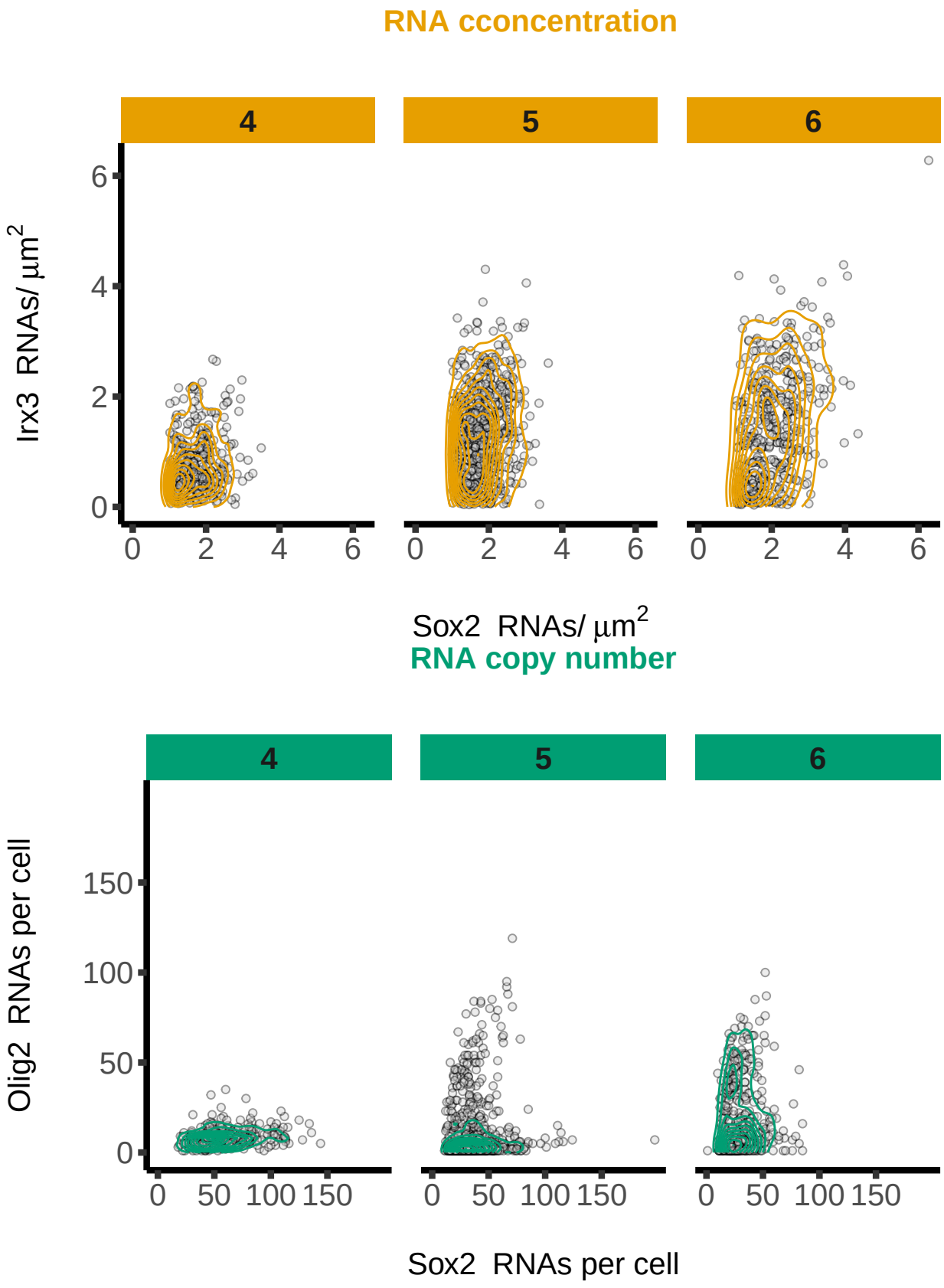


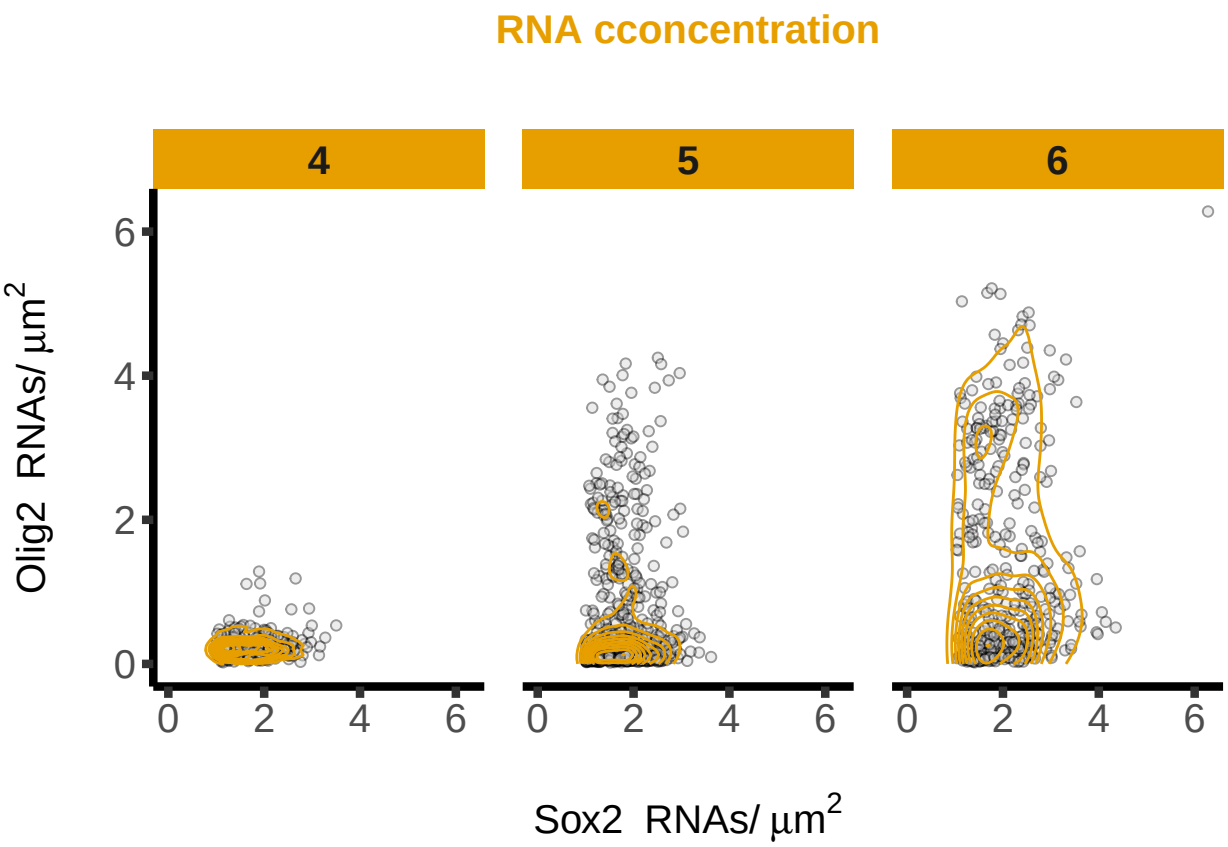

```
#concentration.box.plot(single.cell.progenitors.data, darkTheme)

for (g in c(1,2,3)) {
```

```
  copy.pairwise.scatter.plot(
    single.cell.progenitors.data,
    genes.pairwise$gene.1[g], genes.pairwise$gene.2[g],
    whiteTheme)
#  copy.pairwise.scatter.plot(
#    single.cell.progenitors.data,
#    genes.pairwise$gene.1[g], genes.pairwise$gene.2[g],
#    darkTheme)
  concentration.pairwise.scatter.plot(
    single.cell.progenitors.data,
    genes.pairwise$gene.1[g], genes.pairwise$gene.2[g],
    whiteTheme)
#  concentration.pairwise.scatter.plot(
#    single.cell.progenitors.data,
#    genes.pairwise$gene.1[g], genes.pairwise$gene.2[g],
#    darkTheme)
}
```




RNA cconcentration
4
5
6
Irx3 RNAs/ µm2
Olig2 RNAs/ µm2
RNA copy number
Irx3 RNAs per cell
Sox2 RNAs per cell


RNA cconcentration
4
5
6
Irx3 RNAs/ µm2
0
2
4
6
Sox2 RNAs/ µm2
RNA copy number
Olig2 RNAs per cell
0
50
100
150
Sox2 RNAs per cell



source("26_cell_area.R")

file$channel <- as.factor(file$channel) levels(file$channel) <- genes

rect <- data.frame(xmin=threshold.area, xmax=Inf, ymin=threshold.MaxInt, ymax=Inf, channel=c(0,1,2)) rect$channel <- as.factor(rect$channel) levels(rect$channel) <- genes

source("26_spot_area_maxInt.R")

source("26_TXnumber_vs.R")

source("26_barTXprob.R")

source("26_barTXprobExtrinsic.R")

nested <- total %>% group_by(day, image, cell.id, cell.area, channel, copy, density, TX.number) %>% nest() %>% group_by(day, image, cell.id, cell.area, channel) %>% nest() %>% group_by(day, image, cell.id, cell.area) %>% nest() %>% group_by(day) %>% nest()

myjson <- toJSON(nested, pretty = TRUE, auto_unbox = TRUE)

write(myjson, "single-RNAs_TX-RNAs_cell.json")

library(rjson) json_data <- fromJSON(file="single-RNAs_TX-RNAs_cell.json")

source("26_boxTXmolecules.R")

source("26_TX2alleles.R")

plots <- c(hist.spots.area, hist.spots.MaxInt, p.spots.AreaVsMaxInt, p.TX.numberVsCopy, p.TX.numberVsDensity, p.TX.numberVsCellArea, bar.TX.prob, bar.TX.prob.extrinsic, box.TX.molecules, p.TX.2alleles.molecules.final)

Heterogeneity analysis

var(spots.cell*copy.C*1)/*mean*(*spots.cell*copy.C1)

Fano factor = variance / mean = sigma^2 / mean

FF.expressing <- data.frame(FF.C0=as.numeric(), FF.C1=as.numeric(), FF.C2=as.numeric()) i <- 1 for (d in days) { temp <- spots.cell.expressing[[1]] temp2 <- spots.cell.expressing[[2]] temp3 <- spots.cell.expressing[[3]] FF.expressing[i,] <- c(var(temp*copy.C*0[*temp*day==d]) / mean(temp*copy.C*0[*temp*day==d]), var(temp2*copy.C*1[*temp*2day==d]) / mean(temp2*copy.C*1[*temp*2day==d]), var(temp3*copy.C*2[*temp*3day==d]) / mean(temp3*copy.C*2[*temp*3day==d])) i <- i + 1 }

Save extracted data to file

write.csv(spots.cell, "spots_cell.csv", row.names=FALSE) write.csv(spots.cell.2, "spots_cell2.csv", row.names=FALSE)

write.csv(spots.cell.pro, "spots_cell_pro.csv", row.names=FALSE) write.csv(spots.cell.2.pro, "spots_cell2_pro.csv", row.names=FALSE)

write.csv(TX.molecules, "TX_molecules.csv", row.names=FALSE)