Final Project: In situ hybridization analysis by Larissa Erben

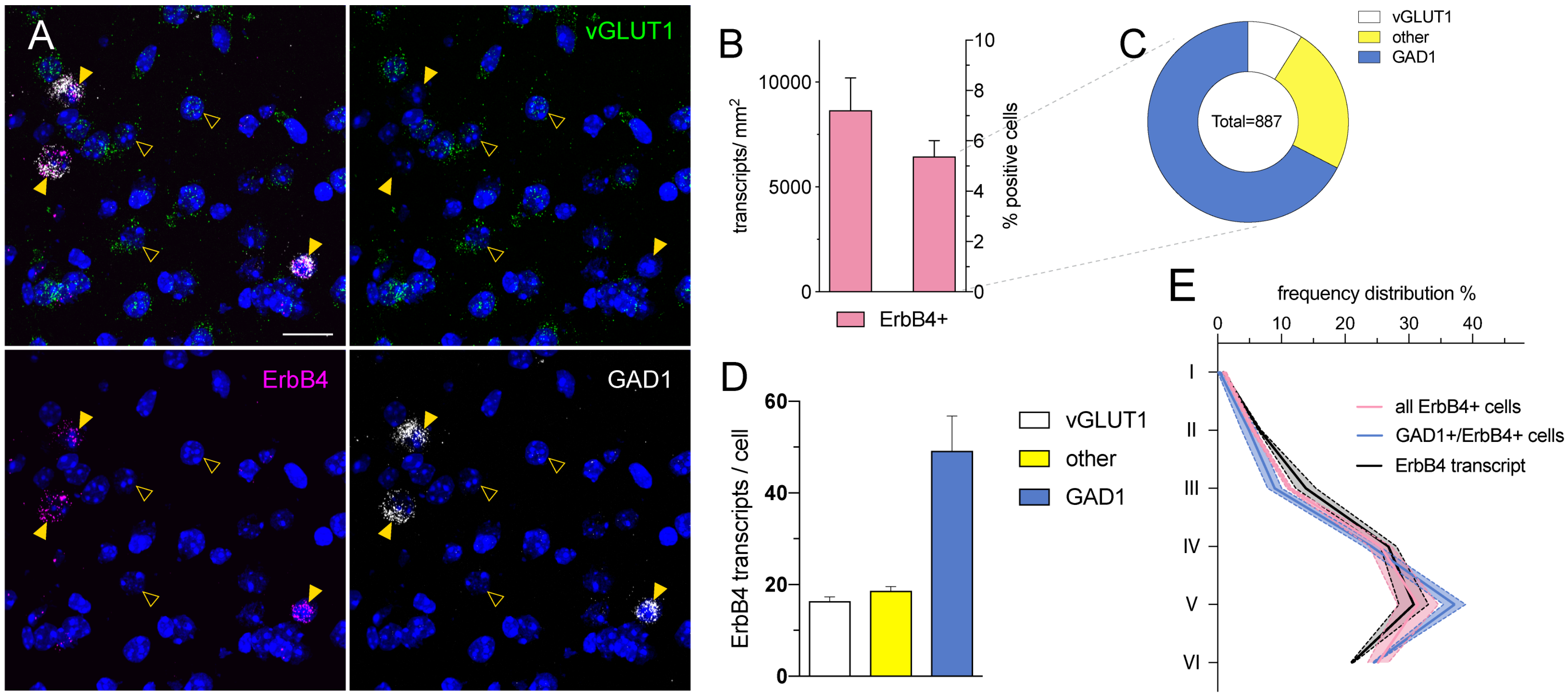
erbenlm

12/12/2018

Background & project aim:

RNAscope is a multiplex fluorescent in situ hybridization approach that allows for gene expression analysis of multiple genes at the time (Wang et al. 2012). We routinely analyze RNAscope signal using custom-made CellProfiler pipelines (Erben et al. 2018). CellProfiler (Carpenter et al. 2006) is a Matlab-based free open-source software for image analysis which outputs are several csv-files that I have analyzed manually in the past.

The aim of this project is to write a script that allows for automated downstream analysis and plotting of the data generated with CellProfiler. I am using a dataset of a recent manuscript (Erben & Buonanno, 2019, Current Protocols of Neurocscience, in revision) and recreating some of graphs in that manuscript.



RNAscope example & quantification

Data:

This in situ hybridization dataset explores the expression of the tyrosine kinase receptor ErbB4 by different cell types in the mouse primary somatosensory cortex. Expression of ErbB4 in the cortex was previously characterized to be confined to inhibitory GABAergic interneurons and absent from excitatory glutamatergic neurons (Vullhorst et al. 2009). This examplary RNAscope hybridization was done using probes for ErbB4 in

Channel 1 detected in red fluorescence, vGLUT1 as glutamatergic marker in Channel 2 visualized with green fluorescence and Gad1 as GABAergic marker in Channel 3 in the far-red channel (denoted as white). The analysis was done bilaterally in two samples.

1. Install relevant packages

```
install.packages('tidyverse',repos='http://cran.r-project.org')
```

```
##
## The downloaded binary packages are in
##  /var/folders/ts/xt7r92890wndz4_d5y2t27znb7q8z1/T//Rtmp6GQFAD/downloaded_packages
```

```
library (tidyverse)
```

```
## ── Attaching packages ──────────────────────────────────── tidyverse 1.2.1 ──
```

```
## ✔ ggplot2 3.1.0     ✔ purrr   0.2.5
## ✔ tibble  1.4.2     ✔ dplyr   0.7.8
## ✔ tidyr   0.8.2     ✔ stringr 1.3.1
## ✔ readr   1.2.1     ✔ forcats 0.3.0
```

```
## ── Conflicts ──────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
install.packages('ggplot2',repos='http://cran.r-project.org')
```

```
##
## The downloaded binary packages are in
##  /var/folders/ts/xt7r92890wndz4_d5y2t27znb7q8z1/T//Rtmp6GQFAD/downloaded_packages
```

```
library(ggplot2)
install.packages('ggExtra',repos='http://cran.r-project.org')
```

```
##
## The downloaded binary packages are in
##  /var/folders/ts/xt7r92890wndz4_d5y2t27znb7q8z1/T//Rtmp6GQFAD/downloaded_packages
```

```
library(ggExtra)
install.packages('readxl', repos='http://cran.r-project.org')
```

```
##
## The downloaded binary packages are in
##  /var/folders/ts/xt7r92890wndz4_d5y2t27znb7q8z1/T//Rtmp6GQFAD/downloaded_packages
```

```
library(readxl)
```

2. Import & explore data:

The summary of the analysis is stored in a csv file with the name _Image. Information about the area analyzed is obtained in ImageJ and stored in a excel file (Area). Additionally, single object information of different subpopulations of ErbB4 positive cells (RedCells, RedandWhite, RedOnly, RedandGreen) as well as the ErbB4 transcript itself (RedTranscript) is loaded. There are two files of each type, one for each sample analyzed.

```
data1<-read_csv('WT1_Image.csv') ## loads files
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   FileName_DAPI = col_character(),
##   FileName_green = col_character(),
##   FileName_red = col_character(),
##   FileName_white = col_character(),
##   ImageSet_ImageSet = col_character(),
##   MD5Digest_DAPI = col_character(),
##   MD5Digest_green = col_character(),
##   MD5Digest_red = col_character(),
##   MD5Digest_white = col_character(),
##   PathName_DAPI = col_character(),
##   PathName_green = col_character(),
##   PathName_red = col_character(),
##   PathName_white = col_character(),
##   ProcessingStatus = col_character(),
##   URL_DAPI = col_character(),
##   URL_green = col_character(),
##   URL_red = col_character(),
##   URL_white = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
data2<-read_csv('WT2_Image.csv')
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   FileName_DAPI = col_character(),
##   FileName_green = col_character(),
##   FileName_red = col_character(),
##   FileName_white = col_character(),
##   ImageSet_ImageSet = col_character(),
##   MD5Digest_DAPI = col_character(),
##   MD5Digest_green = col_character(),
##   MD5Digest_red = col_character(),
##   MD5Digest_white = col_character(),
##   PathName_DAPI = col_character(),
##   PathName_green = col_character(),
##   PathName_red = col_character(),
##   PathName_white = col_character(),
##   ProcessingStatus = col_character(),
##   URL_DAPI = col_character(),
##   URL_green = col_character(),
##   URL_red = col_character(),
##   URL_white = col_character()
## )
## See spec(...) for full column specifications.
```

```
areaIJ<-read_excel('Area.xlsx') ## loads file with information about image size
RedCells1<-read_csv('WT1_RedCells.csv')
```

```
## Parsed with column specification:
## cols(
##   .default = col_double()
## )
## See spec(...) for full column specifications.
```

```
RedCells2<-read_csv('WT2_RedCells.csv')
```

```
## Parsed with column specification:
## cols(
##   .default = col_double()
## )
## See spec(...) for full column specifications.
```

```
RedTranscript1<-read_csv('WT1_Red.csv')
```

```
## Parsed with column specification:
## cols(
##   .default = col_double()
## )
## See spec(...) for full column specifications.
```

```
RedTranscript2<-read_csv('WT2_Red.csv')
```

```
## Parsed with column specification:
## cols(
##   .default = col_double()
## )
## See spec(...) for full column specifications.
```

```
RedandWhite1<-read_csv('WT1_RedandWhiteCells.csv')
```

```
## Parsed with column specification:
## cols(
##   ImageNumber = col_double(),
##   ObjectNumber = col_double(),
##   Children_Red_Count = col_double(),
##   Location_Center_X = col_double(),
##   Location_Center_Y = col_double(),
##   Location_Center_Z = col_double(),
##   Mean_Red_Location_Center_X = col_double(),
##   Mean_Red_Location_Center_Y = col_double(),
##   Mean_Red_Location_Center_Z = col_double(),
##   Mean_Red_Number_Object_Number = col_double(),
##   Number_Object_Number = col_double(),
##   Parent_RedCells = col_double()
## )
```

```
RedandWhite2<-read_csv('WT2_RedandWhiteCells.csv')
```

```
## Parsed with column specification:
## cols(
##   ImageNumber = col_double(),
##   ObjectNumber = col_double(),
##   Children_Red_Count = col_double(),
##   Location_Center_X = col_double(),
##   Location_Center_Y = col_double(),
##   Location_Center_Z = col_double(),
##   Mean_Red_Location_Center_X = col_double(),
##   Mean_Red_Location_Center_Y = col_double(),
##   Mean_Red_Location_Center_Z = col_double(),
##   Mean_Red_Number_Object_Number = col_double(),
##   Number_Object_Number = col_double(),
##   Parent_RedCells = col_double()
## )
```

```
RedOnly1<-read_csv('WT1_RedOnlyCells.csv')
```

```
## Parsed with column specification:
## cols(
##   ImageNumber = col_double(),
##   ObjectNumber = col_double(),
##   Children_Red_Count = col_double(),
##   Location_Center_X = col_double(),
##   Location_Center_Y = col_double(),
##   Location_Center_Z = col_double(),
##   Mean_Red_Location_Center_X = col_double(),
##   Mean_Red_Location_Center_Y = col_double(),
##   Mean_Red_Location_Center_Z = col_double(),
##   Mean_Red_Number_Object_Number = col_double(),
##   Number_Object_Number = col_double(),
##   Parent_RedCells = col_double()
## )
```

```
RedOnly2<-read_csv('WT2_RedOnlyCells.csv')
```

```
## Parsed with column specification:
## cols(
##   ImageNumber = col_double(),
##   ObjectNumber = col_double(),
##   Children_Red_Count = col_double(),
##   Location_Center_X = col_double(),
##   Location_Center_Y = col_double(),
##   Location_Center_Z = col_double(),
##   Mean_Red_Location_Center_X = col_double(),
##   Mean_Red_Location_Center_Y = col_double(),
##   Mean_Red_Location_Center_Z = col_double(),
##   Mean_Red_Number_Object_Number = col_double(),
##   Number_Object_Number = col_double(),
##   Parent_RedCells = col_double()
## )
```

```
RedandGreen1<-read_csv('WT1_RedandGreenCells.csv')
```

```
## Parsed with column specification:
## cols(
##   ImageNumber = col_double(),
##   ObjectNumber = col_double(),
##   Children_Red_Count = col_double(),
##   Location_Center_X = col_double(),
##   Location_Center_Y = col_double(),
##   Location_Center_Z = col_double(),
##   Mean_Red_Location_Center_X = col_double(),
##   Mean_Red_Location_Center_Y = col_double(),
##   Mean_Red_Location_Center_Z = col_double(),
##   Mean_Red_Number_Object_Number = col_double(),
##   Number_Object_Number = col_double(),
##   Parent_RedCells = col_double()
## )
```

```
RedandGreen2<-read_csv('WT2_RedandGreenCells.csv')
```

```
## Parsed with column specification:
## cols(
##   ImageNumber = col_double(),
##   ObjectNumber = col_double(),
##   Children_Red_Count = col_double(),
##   Location_Center_X = col_double(),
##   Location_Center_Y = col_double(),
##   Location_Center_Z = col_double(),
##   Mean_Red_Location_Center_X = col_double(),
##   Mean_Red_Location_Center_Y = col_double(),
##   Mean_Red_Location_Center_Z = col_double(),
##   Mean_Red_Number_Object_Number = col_double(),
##   Number_Object_Number = col_double(),
##   Parent_RedCells = col_double()
## )
```

The summmary data (data1 and data2) are wide dataframes with many columns and two rows per hemisphere analyzed. The data of each subpopulation contains information about of each single object (cell or transcript) identified (number of rows).

```
dim(data1) ## dimensions of data
```

```
## [1]   2 621
```

```
dim(data2)
```

```
## [1]   2 621
```

```
names (data1[1:20]) ## column names
```

```
##  [1] "Channel_DAPI"                  "Channel_green"
##  [3] "Channel_red"                   "Channel_white"
##  [5] "Count_Cells"                   "Count_Green"
##  [7] "Count_GreenCells"              "Count_Nuclei"
##  [9] "Count_NucleiArea"              "Count_Red"
## [11] "Count_RedCells"                "Count_RedDotsonRedandGreenCells"
## [13] "Count_RedOnlyCells"            "Count_RedandGreenCells"
## [15] "Count_RedandWhiteCells"        "Count_RedonRedOnly"
## [17] "Count_RedonRedandWhite"        "Count_RedonTriple"
## [19] "Count_TripleCells"             "Count_White"
```

```
dim(RedCells1)
```

```
## [1] 549  26
```

```
dim(RedCells2)
```

```
## [1] 442  26
```

```
names(RedCells1[1:20])
```

```
##  [1] "ImageNumber"                    "ObjectNumber"
##  [3] "Children_Green_Count"           "Children_RedOnlyCells_Count"
##  [5] "Children_Red_Count"             "Children_RedandGreenCells_Count"
##  [7] "Children_RedandWhiteCells_Count" "Children_TripleCells_Count"
##  [9] "Children_White_Count"           "Location_Center_X"
## [11] "Location_Center_Y"              "Location_Center_Z"
## [13] "Mean_Green_Location_Center_X"   "Mean_Green_Location_Center_Y"
## [15] "Mean_Green_Location_Center_Z"   "Mean_Green_Number_Object_Number"
## [17] "Mean_Red_Location_Center_X"     "Mean_Red_Location_Center_Y"
## [19] "Mean_Red_Location_Center_Z"     "Mean_Red_Number_Object_Number"
```

```
names(RedTranscript1[1:20])
```

```
##  [1] "ImageNumber"              "ObjectNumber"
##  [3] "AreaShape_Area"           "AreaShape_Center_X"
##  [5] "AreaShape_Center_Y"       "AreaShape_Center_Z"
##  [7] "AreaShape_Compactness"    "AreaShape_Eccentricity"
##  [9] "AreaShape_EulerNumber"    "AreaShape_Extent"
## [11] "AreaShape_FormFactor"     "AreaShape_MajorAxisLength"
## [13] "AreaShape_MaxFeretDiameter" "AreaShape_MaximumRadius"
## [15] "AreaShape_MeanRadius"     "AreaShape_MedianRadius"
## [17] "AreaShape_MinFeretDiameter" "AreaShape_MinorAxisLength"
## [19] "AreaShape_Orientation"    "AreaShape_Perimeter"
```

3. Manipulation & Tidying of data

Selection of relevant columns of the summary data

```
WT1<-data1 %>% select(Count_Cells:Count_WhiteCells) ##selects columns of interests
WT2<-data2 %>% select(Count_Cells:Count_WhiteCells)
```

Conversion of area size in pixel into um2 (factor needs to be adjusted with different magnification images - here 40X)

```
areaIJ<-areaIJ %>%
  mutate(um2=Area*0.043083972) %>%  ## calculates area measured in pixels into um2
  select(um2) ## selects this single column
```

The information of the area is added to the main table with cbind function.

```
areaIJ1<-areaIJ[1:2,] ## splits the tabl for the two animals analyzed
areaIJ2<-areaIJ[3:4,]
WT1<-cbind(WT1, areaIJ1) ## adds the area information to the main table
WT2<-cbind(WT2, areaIJ2)
```

The data of the two hemispheres is sumed, and the tables of the two samples combined.

```
WT1_sum<-WT1 %>% summarize_all(sum) ##sums the data of the two hemisphere
WT2_sum<-WT2 %>% summarize_all(sum)
Image<-rbind(WT2_sum, WT1_sum) ## combines the two tables, first one will be second i
n the combined dataset
```

Additional variables necessary for the analysis are calculated such as the percentage of positive cells and the mean expression levels per positive cells.

```
Image<-Image %>% ## calculates missing variables
  mutate(Redxp=Count_RedCells/Count_Cells*100) %>% ##percentage of positive cells per
all cells
  mutate(Greenxp=Count_GreenCells/Count_Cells*100) %>%
  mutate(Whitexp=Count_WhiteCells/Count_Cells*100) %>%
  mutate(RealRed=Count_RedandGreenCells+Count_RedandWhiteCells+Count_RedOnlyCells) %>
% ##excludes triple positives (biologically unpossible)
  mutate(RedOnlyxp = Count_RedOnlyCells/RealRed*100) %>% ## percentage of positive ce
lls per all red cells
  mutate(RedandGreenxp = Count_RedandGreenCells/RealRed*100) %>%
  mutate(RedandWhitexp = Count_RedandWhiteCells/RealRed*100) %>%
  mutate(Redxsignal=Count_Red/um2*1000000) %>% ##signal per area, in mm2
  mutate(Greenxsignal=Count_Green/um2*1000000) %>%
  mutate(Whitexsignal=Count_White/um2*1000000) %>%
  mutate(RedandWhitexsignal=Count_RedonRedandWhite/um2*1000000) %>%
  mutate(RedandGreenxsignal=Count_RedDotsonRedandGreenCells/um2*1000000) %>%
  mutate(RedOnlyxsignal=Count_RedonRedOnly/um2*1000000) %>%
  mutate(Triplexsignal=Count_RedonTriple/um2*1000000) %>%
  mutate(Redxsignalpc=Count_Red/Count_RedCells) %>% ## dots per positive cell
  mutate(Greenxsignalpc=Count_Green/Count_GreenCells) %>%
  mutate(Whitexsignalpc=Count_White/Count_WhiteCells) %>%
  mutate(Triplexsignalpc=Count_RedonTriple/Count_TripleCells) %>%
  mutate(RedOnlyxsignalpc=Count_RedonRedOnly/Count_RedOnlyCells) %>% ## dots subtype
of red cells per cell
  mutate(RedandGreenxsignalpc=Count_RedDotsonRedandGreenCells/Count_RedandGreenCells)
%>%
  mutate(RedandWhitexsignalpc=Count_RedonRedandWhite/Count_RedandWhiteCells)
```

For downstream manipulations I changed some column names.

```
Image_tidy<-Image %>% select(Count_Cells:Count_GreenCells,Count_Red:Count_WhiteCells,
Redxp:Whitexp, RedOnlyxp:RedandWhitexsignalpc) ##selects the columns of interest
Image_tidy<-Image_tidy %>% rename (CellsxCount=Count_Cells,GreenxCount = Count_Green,
GreenxCells = Count_GreenCells,
                                   RedxCount = Count_Red,RedandGreenxCells = Count_Re
dandGreenCells, RedandWhitexCells = Count_RedandWhiteCells,
                                   RedxCells = Count_RedCells, RedandGreenxCount = Co
unt_RedDotsonRedandGreenCells,
                                   RedOnlyxCells = Count_RedOnlyCells, RedandWhitexCo
unt = Count_RedonRedandWhite,
                                   RedOnlyxCount = Count_RedonRedOnly, TriplexCount =
Count_RedonTriple,
                                   TriplexCells = Count_TripleCells, WhitexCount = Co
unt_White,
                                   WhitexCells = Count_WhiteCells)
```

The Mean and SEM for each variable are calculated and the dataframe is gathered and spread to obtain a table summarizing the information per cell type.

```
Image_tidy_sums<-Image_tidy %>%
  summarize_all(funs(Mean = mean(., na.rm=T),
                     SEM = sd(., na.rm=T)/sqrt(n()))) %>% ##calculates Mean+SEM from
each colunm
  gather(variable, value) %>%  ## gathers data to table with variable and value
  separate(variable, c('CellType',"Measurement"), sep='x') %>% ## separates different
measurements by cell type
  spread(Measurement, value) ## summarizes per cell type
```

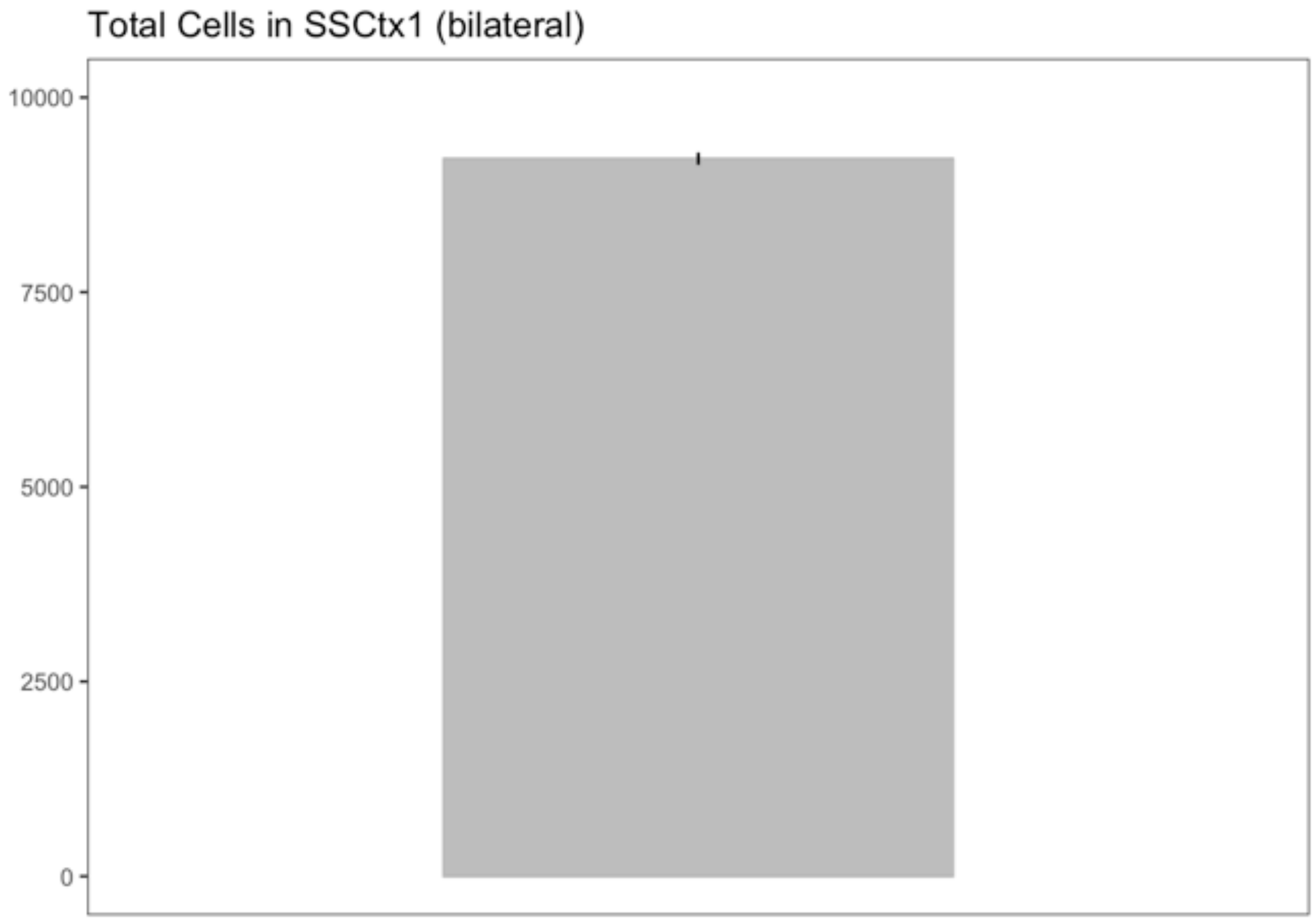
This table is split into several tables with different information about all cells detected (Cells), general information about the three different cell types (Red, Green, White; RGW) and the subpopulation of ErbB4 positive cells (Red) for plotting these different subsets of data.

```
RGW<-Image_tidy_sums %>%filter(CellType %in% c('Green', 'Red', 'White'))
Cells_df<-Image_tidy_sums %>% filter (CellType == 'Cells')
Red<-Image_tidy_sums %>% filter (CellType %in% c('RedandGreen','RedandWhite','RedOnly
','Triple'))
```

4. Exploratory data plotting

For each of the tables, I did an exploratory graph which are exported as a pdf file.

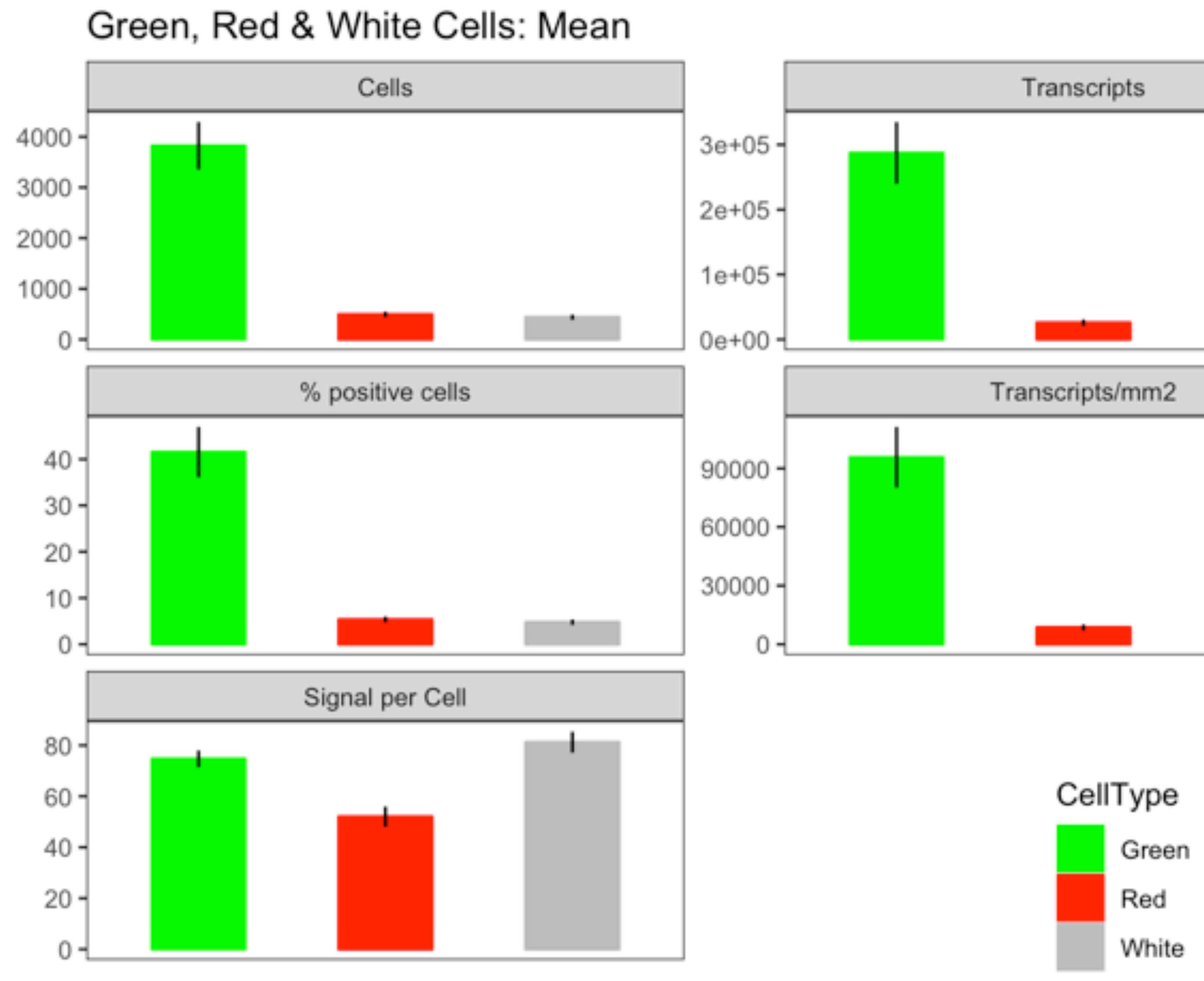
```
Cells_plot<-ggplot(Cells_df, aes(x=CellType, y=Count_Mean, ymin=Count_Mean-Count_SEM,
ymax=Count_Mean+Count_SEM, fill=CellType)) +
  geom_col(colour="grey",width=0.5) + ##column blot
  geom_linerange() + ##adds errorbars defined with ymin and ymax
  scale_fill_manual(values="grey")+ ##fills the bar in grey scale colors
  ylim(0,10000) + ##sets y-axis limits
  ggtitle ("Total Cells in SSCtx1 (bilateral)") + ## title of blot
  xlab ("") + ##no x title
  ylab ("") +  ##no y title
  theme_test()+
  theme(legend.position="none") + ## no legend
  theme(axis.text.x = element_blank(), axis.ticks.x = element_blank())
Cells_plot
```



```
pdf(file="cells.pdf", width=4, height=4) ##prints pdf
print(Cells_plot)
dev.off()
```

```
## quartz_off_screen
##                 2
```

```
group.colors<-c('Green'='green','Red'='red','White'='grey')
RGW_sum<-RGW %>%
  gather(variable, value, -CellType) %>%
  separate (variable,c('Variable','Stat'), sep='_') %>%
  spread (Stat, value) %>%
  mutate (lcb = Mean - SEM, ucb = Mean + SEM)
subplot_names = c('Cells' = 'Cells', 'Count' = 'Transcripts', 'p' = '% positive cells
', 'signal' = 'Transcripts/mm2',
                  'signalpc' = 'Signal per Cell')
RGW_sum_blot<-ggplot(RGW_sum, aes(x=CellType, y= Mean, ymin=lcb, ymax=ucb, fill=CellT
ype, color=CellType))+
  facet_wrap(~Variable, scales = 'free_y',ncol=2, labeller = as_labeller(subplot_name
s)) + ##allows to blot multiple variables
  geom_col(width=0.5)+ ##line colors
  scale_color_manual(values=group.colors)+
  scale_fill_manual(values=group.colors)+ ##fill colors
  geom_linerange(colour="black")+ ## errorbars, needs ymin and ymax
  labs (x='', y='') + ##labels x and y axis
  theme_test()+ ## changes theme
  ggtitle ('Green, Red & White Cells: Mean')+ ## title of blot
  theme(legend.position =c(0.8,0.1)) +## position of legend
  theme(axis.text.x = element_blank(), axis.ticks.x = element_blank()) ## Removes axi
s labels
RGW_sum_blot
```



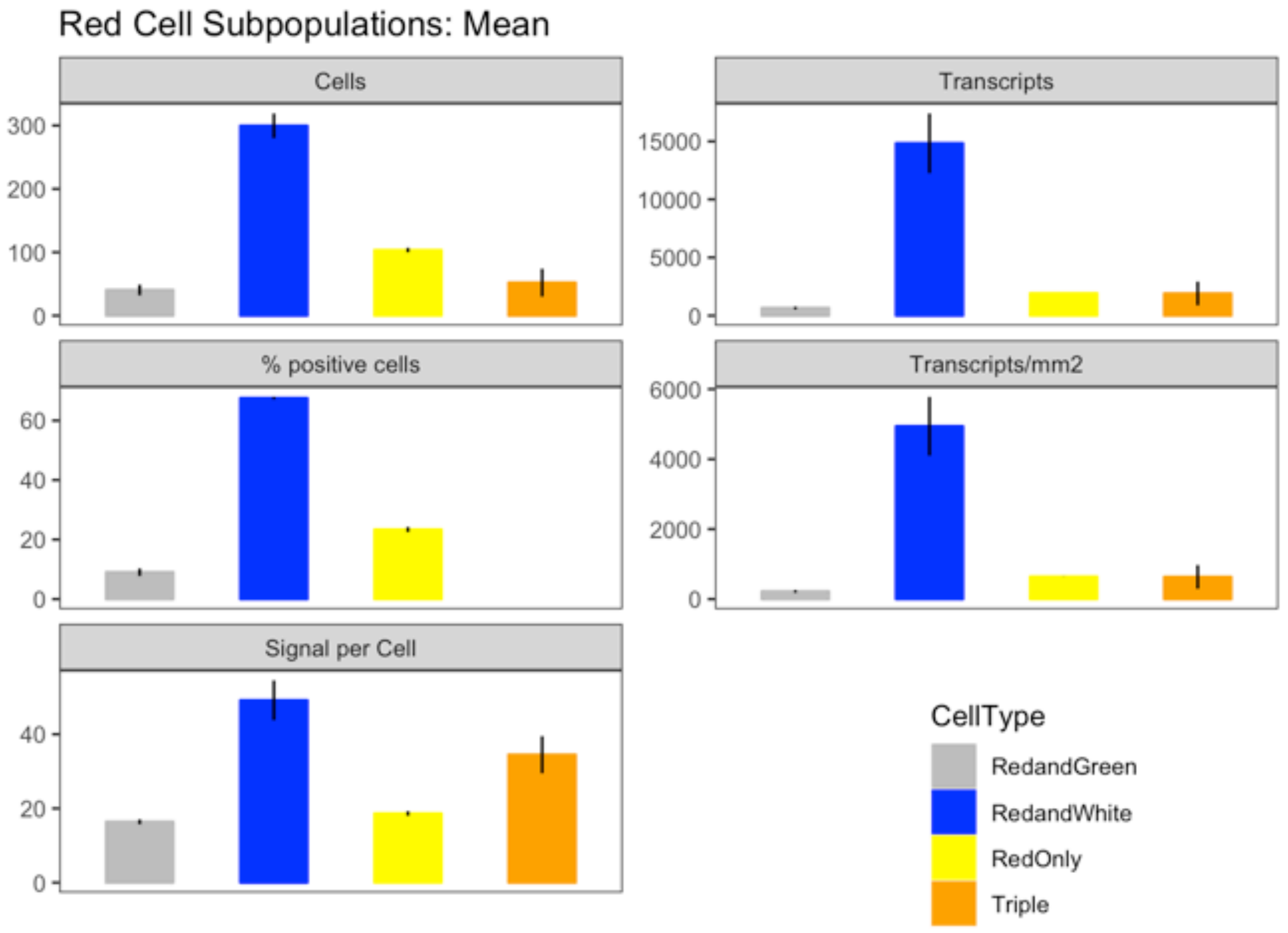
```
pdf(file="RGW facetwrap.pdf", width=4, height=4) ##prints pdf
print(RGW_sum_blot)
dev.off()
```

```
## quartz_off_screen
##                 2
```

```r
Red_sum<-Red %>%
  gather(variable, value, -CellType) %>%
  separate (variable,c('Variable','Stat'), sep='_') %>%
  spread (Stat, value) %>%
  mutate (lcb = Mean - SEM, ucb = Mean + SEM)
group.colors2<-c('RedandGreen'='grey','RedandWhite'='blue','RedOnly'='yellow','Triple
'='orange')
Red_sum_blot<-ggplot(Red_sum, aes(x=CellType, y= Mean, ymin=lcb, ymax=ucb, fill=CellT
ype, color=CellType))+
  facet_wrap(~Variable, scales = 'free_y',ncol=2, labeller = as_labeller(subplot_name
s)) + ##allows to blot multiple variables
  geom_col(width=0.5)+
  scale_fill_manual(values=group.colors2)+ ##fill colors
  scale_color_manual(values=group.colors2)+ ##line colors
  geom_linerange(colour='black')+
  labs (x='', y='') + ##labels x and y axis
  theme_test()+ ## changes theme
  ggtitle ('Red Cell Subpopulations: Mean')+ ## title of blot
  theme(legend.position=c(0.8,0.1))+
  theme(axis.text.x = element_blank(), axis.ticks.x = element_blank())
Red_sum_blot
```

```
## Warning: Removed 1 rows containing missing values (position_stack).
```

```
## Warning: Removed 1 rows containing missing values (geom_linerange).
```



```
pdf(file="Red facetwrap.pdf", width=4, height=4) ##prints pdf
print(Red_sum_blot)
```

```
## Warning: Removed 1 rows containing missing values (position_stack).

## Warning: Removed 1 rows containing missing values (geom_linerange).
```

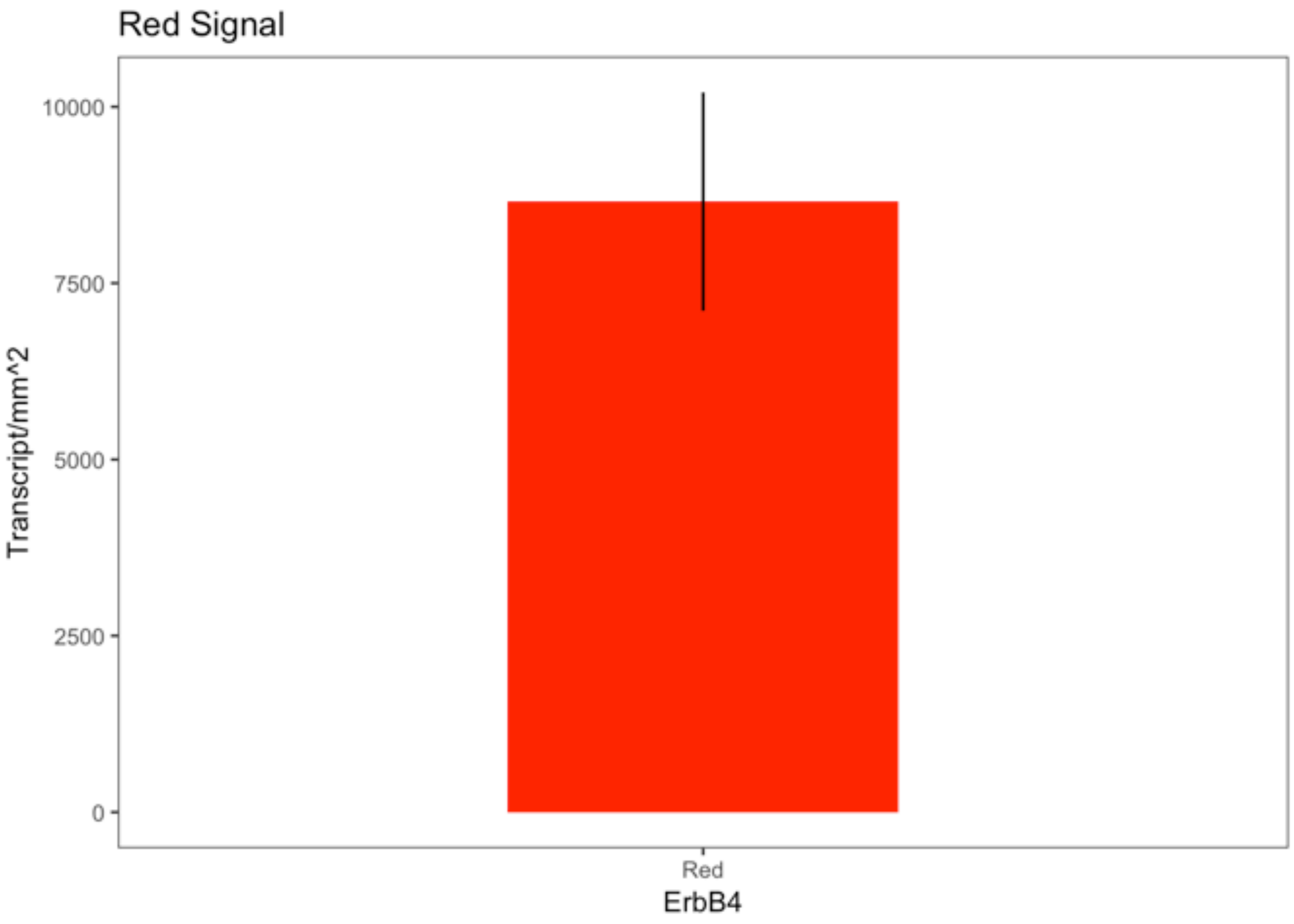
```
dev.off()
```

```
## quartz_off_screen
##                 2
```

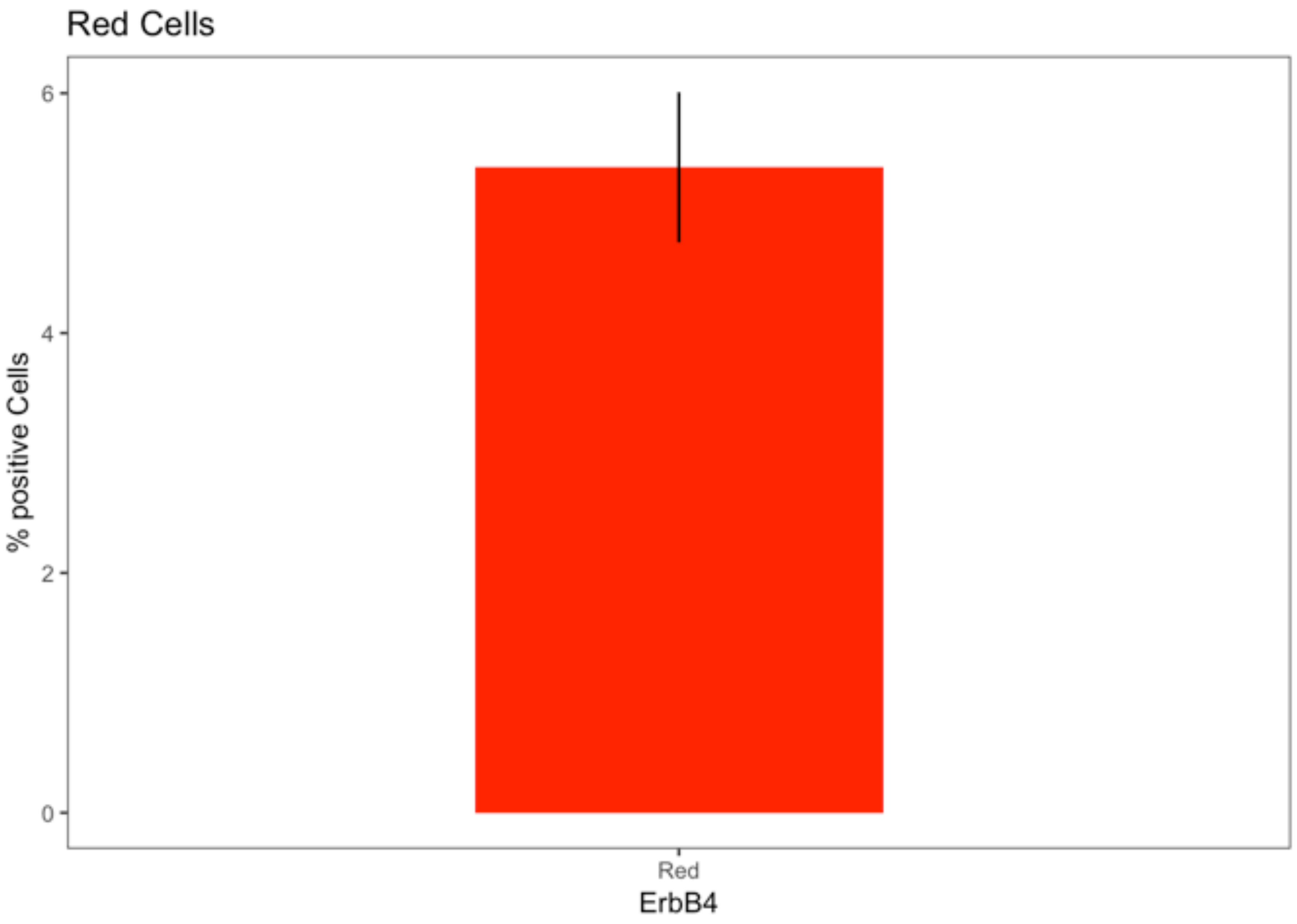
5. Graphs of the summary data

As in our manuscript, from this summary data I plotted the expression of the red ErbB4 transcript per area, the percentage of ErbB4 positive cells, a pie plot of the subpoplutaions of ErbB4 positive cells and the mean ErbB4 transceipt levels in these cell populations.

```
RGW_Red<-RGW %>%filter(CellType %in% c('Red'))## picks the Red data only for the plot
s
Redtranscript<-ggplot(RGW_Red,aes(x=CellType,y=signal_Mean, ymin=signal_Mean-signal_S
EM, ymax=signal_Mean+signal_SEM, fill=CellType))+ ##figure for red expression levels
  geom_col(width=0.4)+ ## bargraph, smaller width
  geom_linerange()+ ##errorbar needs ymin and ymax
  theme_test()+ ## theme for white background no lines
  ggtitle('Red Signal')+ ## blot title
  labs(x="ErbB4",y="Transcript/mm^2")+ ## axis titles
  theme(legend.position="none") + ## no legends
  scale_fill_manual(values=c("red")) ## fill color red
Redtranscript ##display
```



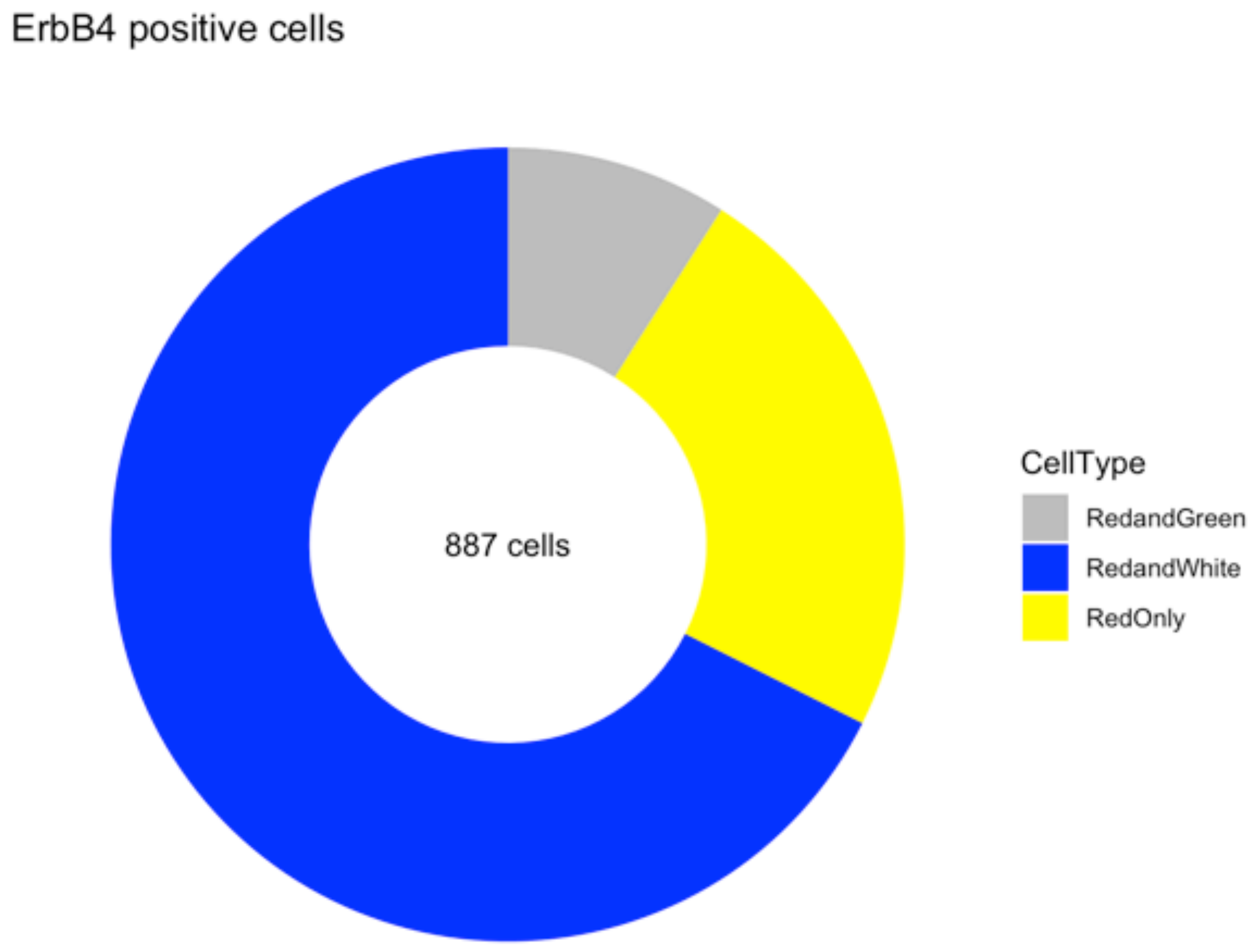
```
Redpositive<-ggplot(RGW_Red,aes(x=CellType, y=p_Mean, ymin=p_Mean-p_SEM, ymax=p_Mean+
p_SEM, fill=CellType))+ ##similar figure for percentage of postive cells
  geom_col(width=0.4)+
  geom_linerange()+
  theme_test()+
  ggtitle('Red Cells')+
  labs(x='ErbB4', y='% positive Cells')+
  theme(legend.position="none")+
  scale_fill_manual(values=c("red"))
Redpositive ##display
```



```
pdf(file="ErbB4expression.pdf", width=4, height=4, onefile=TRUE) ##prints pdf ##print
two figures in one file
print(Redtranscript)
print(Redpositive)
dev.off()
```

```
## quartz_off_screen
##                  2
```

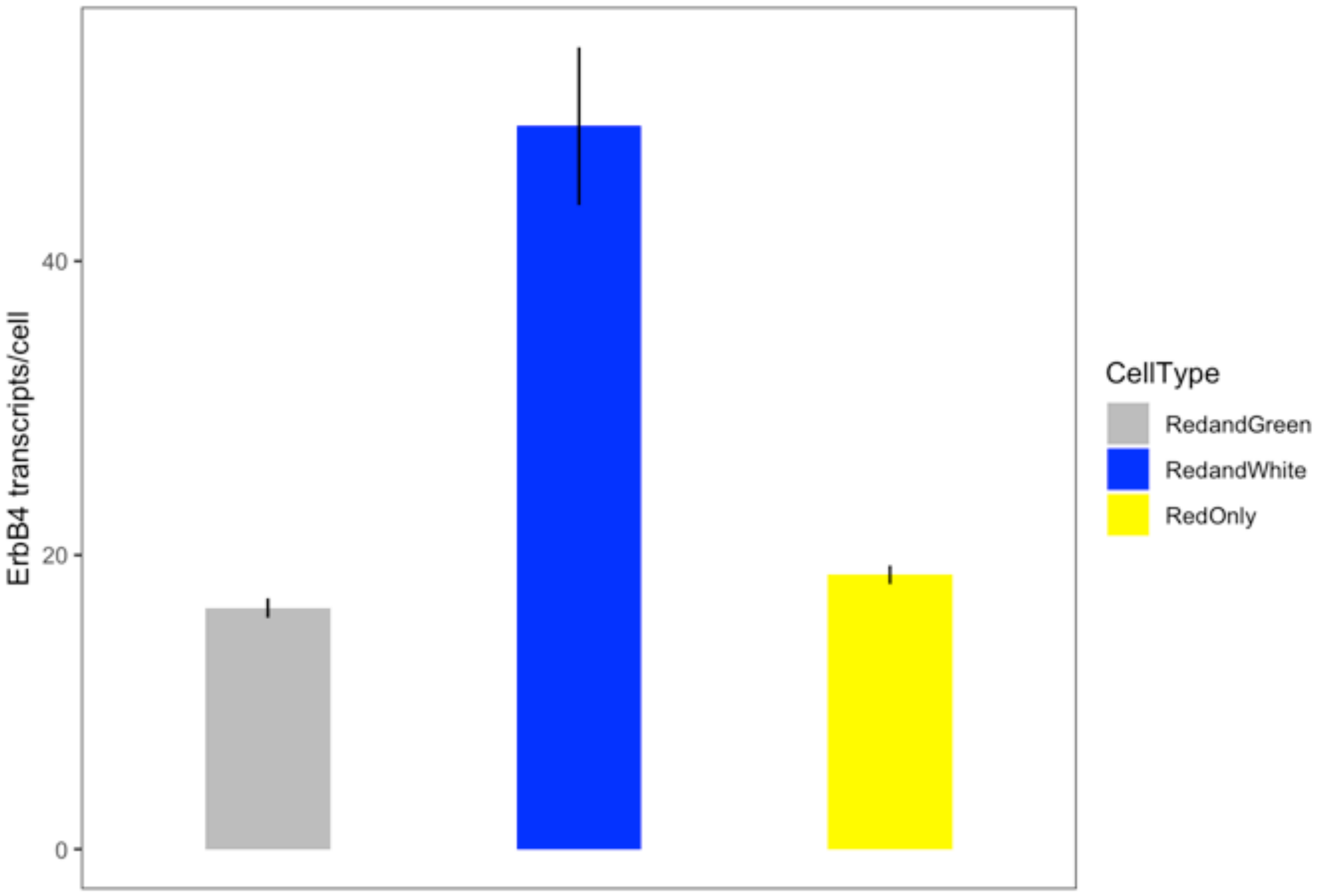
```
CellType<-Red[,1] ## picks the data needed for the pie
Red_pie2<-Red[,6:7]
Red_pie3<-cbind(CellType,Red_pie2)
Red_pie<-Red_pie3[1:3,]
Red_pie<-Red_pie %>% ## mutates data for the pie_chart
  mutate(fraction=p_Mean/100)
Red_pie=Red_pie[order(Red_pie$fraction),]
Red_pie$ymax=cumsum(Red_pie$fraction)
Red_pie$ymin = c(0, head(Red_pie$ymax, n=-1))
pie<-ggplot(Red_pie, aes(fill=CellType, ymax=ymax, ymin=ymin, xmax=4, xmin=2)) + ##pl
ots Pie chart
  geom_rect() +
  coord_polar(theta="y") +
  xlim(c(0, 4)) +
  labs(title="")+
  theme_void()+ ##liked this one better this time
  annotate("text", x = 0, y = 0, label = "887 cells") +
  scale_fill_manual(values=c("grey","blue","yellow"))+
  ggtitle('ErbB4 positive cells')
pie
```



```
pdf(file="ErbB4 pie.pdf", width=4, height=4) ##prints pdf
print(pie)
dev.off()
```

```
## quartz_off_screen
##                 2
```

```
Red_noTriple<-Red[1:3,] ## excludes triple positive cells
transcriptspcell<-ggplot(Red_noTriple,aes(x=CellType, y=signalpc_Mean, ymin=signalpc_
Mean-signalpc_SEM, ymax=signalpc_Mean+signalpc_SEM, fill=CellType))+
  geom_col(width=0.4)+
  geom_linerange()+
  theme_test()+
  scale_fill_manual(values=c("grey","blue","yellow"))+
  labs(x="",y="ErbB4 transcripts/cell")+
  theme(axis.text.x = element_blank(), axis.ticks.x = element_blank())
transcriptspcell
```



```
pdf(file="Transcripts per cell.pdf", width=4, height=4) ##prints pdf
print(transcriptspcell)
dev.off()
```

```
## quartz_off_screen
##                 2
```

6. Per cell expression

Next, I am looking into expression per cell and distribution of the signal which has been quite challenging in my previous manual analysis due to the size of the tables. First, columns are selected and a column specifying the CellType is added.

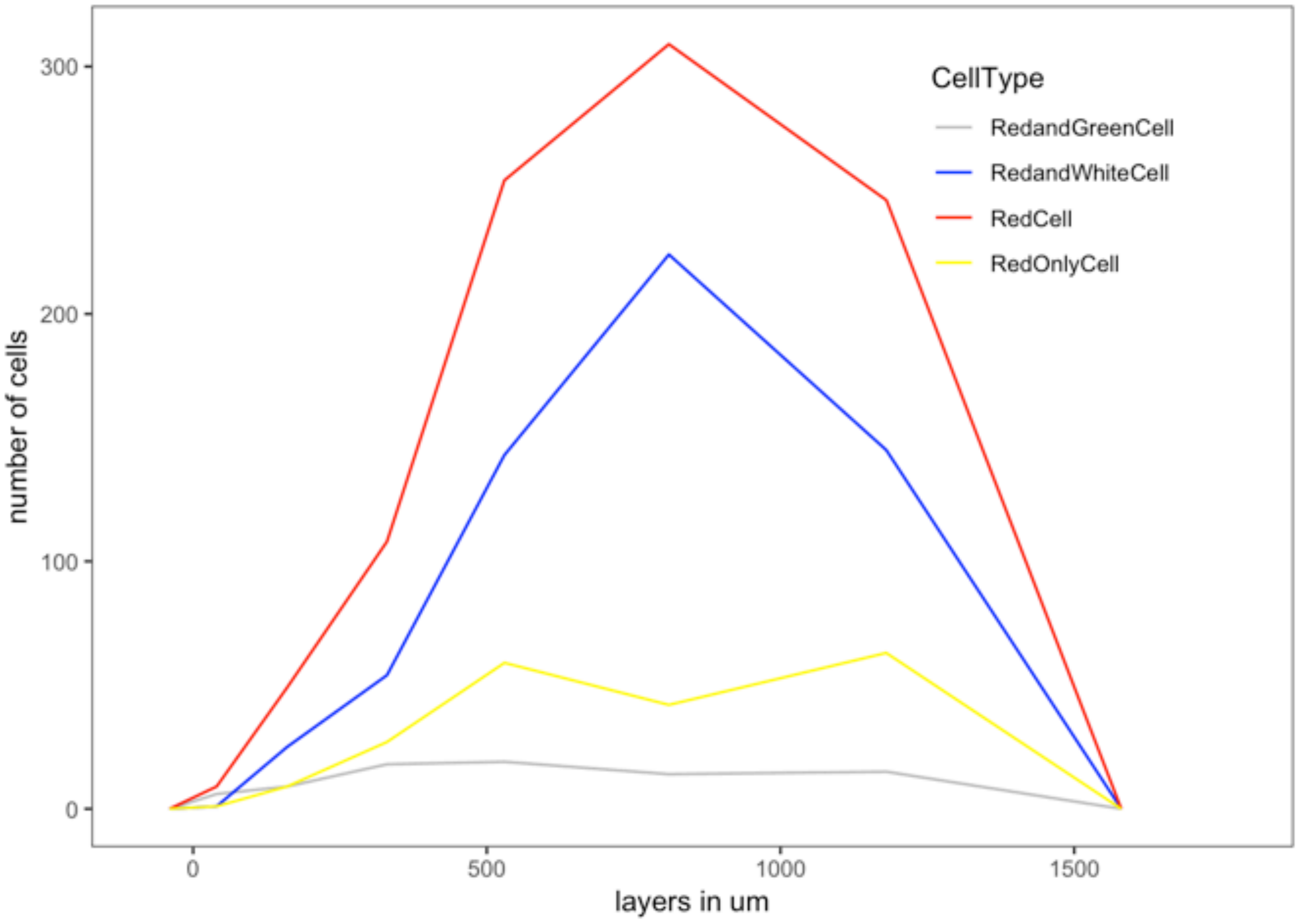
```
RedCells1<-RedCells1 %>% select(ImageNumber, ObjectNumber, Children_Red_Count, Locati
on_Center_X,Location_Center_Y) %>%  ##selects columns of interests
  mutate(CellType='RedCell') ##adds column that specifies cell type so tables can be
merged
RedCells2<-RedCells2 %>% select(ImageNumber, ObjectNumber, Children_Red_Count, Locati
on_Center_X,Location_Center_Y) %>%
  mutate(CellType='RedCell') %>%
  mutate(ImageNumber = case_when(ImageNumber %in% c(1) ~ 3, ImageNumber %in% c(2) ~ 4
)) ##changes ImageNumbers to 3,4, so when combined with first sample no duplicates
RedTranscript1<-RedTranscript1 %>% select(ImageNumber, ObjectNumber, Location_Center_
X,Location_Center_Y) %>%
  mutate(CellType='RedTranscript') %>%
  mutate(Children_Red_Count='NA')
RedTranscript2<-RedTranscript2 %>% select(ImageNumber, ObjectNumber, Location_Center_
X,Location_Center_Y) %>%
  mutate(CellType='RedTranscript') %>%
  mutate(Children_Red_Count='NA') %>%
  mutate(ImageNumber = case_when(ImageNumber %in% c(1) ~ 3, ImageNumber %in% c(2) ~ 4
))
RedandWhite1<-RedandWhite1 %>% select(ImageNumber, ObjectNumber, Children_Red_Count,
Location_Center_X,Location_Center_Y) %>%
  mutate(CellType='RedandWhiteCell')
RedandWhite2<-RedandWhite2 %>% select(ImageNumber, ObjectNumber, Children_Red_Count,
Location_Center_X,Location_Center_Y) %>%
  mutate(CellType='RedandWhiteCell') %>%
  mutate(ImageNumber = case_when(ImageNumber %in% c(1) ~ 3, ImageNumber %in% c(2) ~ 4
))
RedOnly1<-RedOnly1 %>% select(ImageNumber, ObjectNumber, Children_Red_Count, Location
_Center_X,Location_Center_Y) %>%
  mutate(CellType='RedOnlyCell')
RedOnly2<-RedOnly2 %>% select(ImageNumber, ObjectNumber, Children_Red_Count, Location
_Center_X,Location_Center_Y) %>%
  mutate(CellType='RedOnlyCell') %>%
  mutate(ImageNumber = case_when(ImageNumber %in% c(1) ~ 3, ImageNumber %in% c(2) ~ 4
))
RedandGreen1<-RedandGreen1 %>% select(ImageNumber, ObjectNumber, Children_Red_Count,
Location_Center_X,Location_Center_Y) %>%
  mutate(CellType='RedandGreenCell')
RedandGreen2<-RedandGreen2 %>% select(ImageNumber, ObjectNumber, Children_Red_Count,
Location_Center_X,Location_Center_Y) %>%
  mutate(CellType='RedandGreenCell') %>%
  mutate(ImageNumber = case_when(ImageNumber %in% c(1) ~ 3, ImageNumber %in% c(2) ~ 4
))
combined<-rbind(RedCells1, RedCells2, RedTranscript1, RedTranscript2, RedandWhite1, R
edandWhite2, RedOnly1, RedOnly2, RedandGreen1, RedandGreen2)
```

Position in pixels need to be converted to um and cortical thickness is normalized between images.

```
combined<-combined %>%
  mutate(Xum=Location_Center_X/8404*1744.39) %>% ## converting pixels into um (here v
alue for 40X)
  mutate(Xum_norm = case_when(ImageNumber %in% c(1) ~ Xum*0.9911047, ImageNumber %in%
c(2) ~ Xum*1.01774873, ImageNumber %in% c(3) ~ Xum*1.01776363, ImageNumber %in% c(4)
~Xum*0.97473494)) %>% ##normalizing images to same size to adjust for small differenc
es in thickness of cortex
  mutate(Yum=Location_Center_Y/8404*1744.39) %>%
  mutate(Yum_norm = case_when(ImageNumber %in% c(1) ~ Yum*1, ImageNumber %in% c(2) ~
Yum*1, ImageNumber %in% c(3) ~ Yum*1.00645161, ImageNumber %in% c(4) ~Yum*1))
cellsonly<-combined %>% filter (!CellType %in% c('RedTranscript'))
cellsonly$Children_Red_Count<-as.numeric(cellsonly$Children_Red_Count)
```

There are six cortical layers. Layer cell distribution is defined with bins which were measured in ImageJ and then plotted as frequency plot. For the inclusion of the ErbB4 transcript data required a normalized density plot to be visualized.

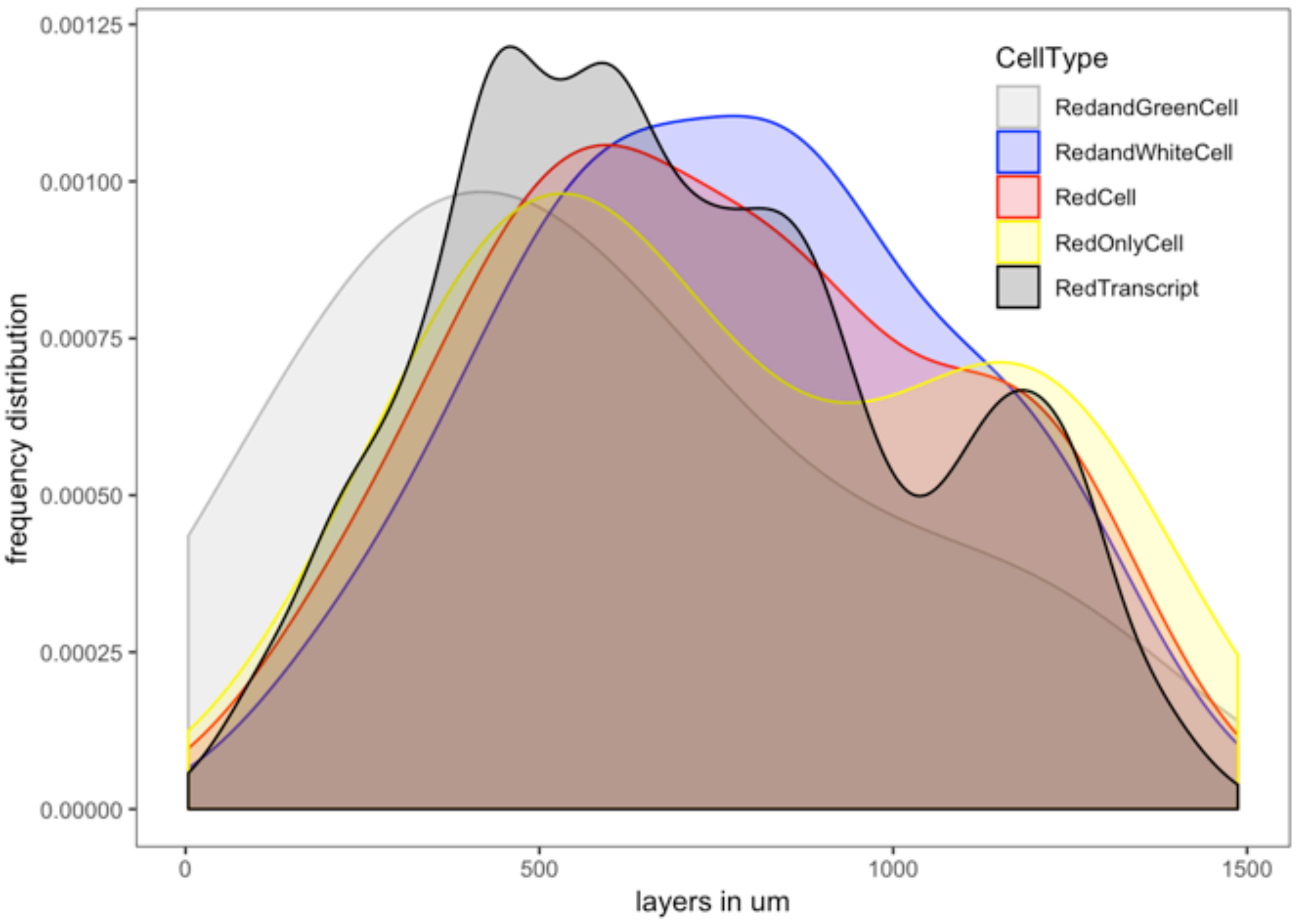
```
bins<-c(0,80,240,420,640,980,1380) ## determines six cortical layers in um
cellcolors<-c('RedTranscript'='black','RedandWhiteCell'='blue','RedandGreenCell'='gre
y','RedOnlyCell'='yellow', 'RedCell'='red')
p1=ggplot(cellsonly, aes(x=Xum_norm, group=CellType, color=CellType, fill=CellType))
+
  geom_freqpoly(aes(x=Xum_norm), breaks=bins) +
  scale_color_manual(values=cellcolors) +
  scale_fill_manual(values=cellcolors)+
  labs (x='layers in um', y='number of cells') + ##labels x and y axis
  theme_test() +
  theme(legend.position=c(0.8,0.8))
p1
```



```
pdf(file="layer1.pdf", width=8, height=6) ##prints pdf
print(p1)
dev.off()
```

```
## quartz_off_screen
##                  2
```

```
p2=ggplot(combined, aes(x=Xum_norm, group=CellType, color=CellType, fill=CellType)) +
  geom_density(adjust=1.5, alpha=0.2)+
  theme_test()+
  scale_color_manual(values=cellcolors)+
  scale_fill_manual(values=cellcolors) +
  labs (x='layers in um', y='frequency distribution') +
  theme(legend.position=c(0.85,0.8))
p2
```

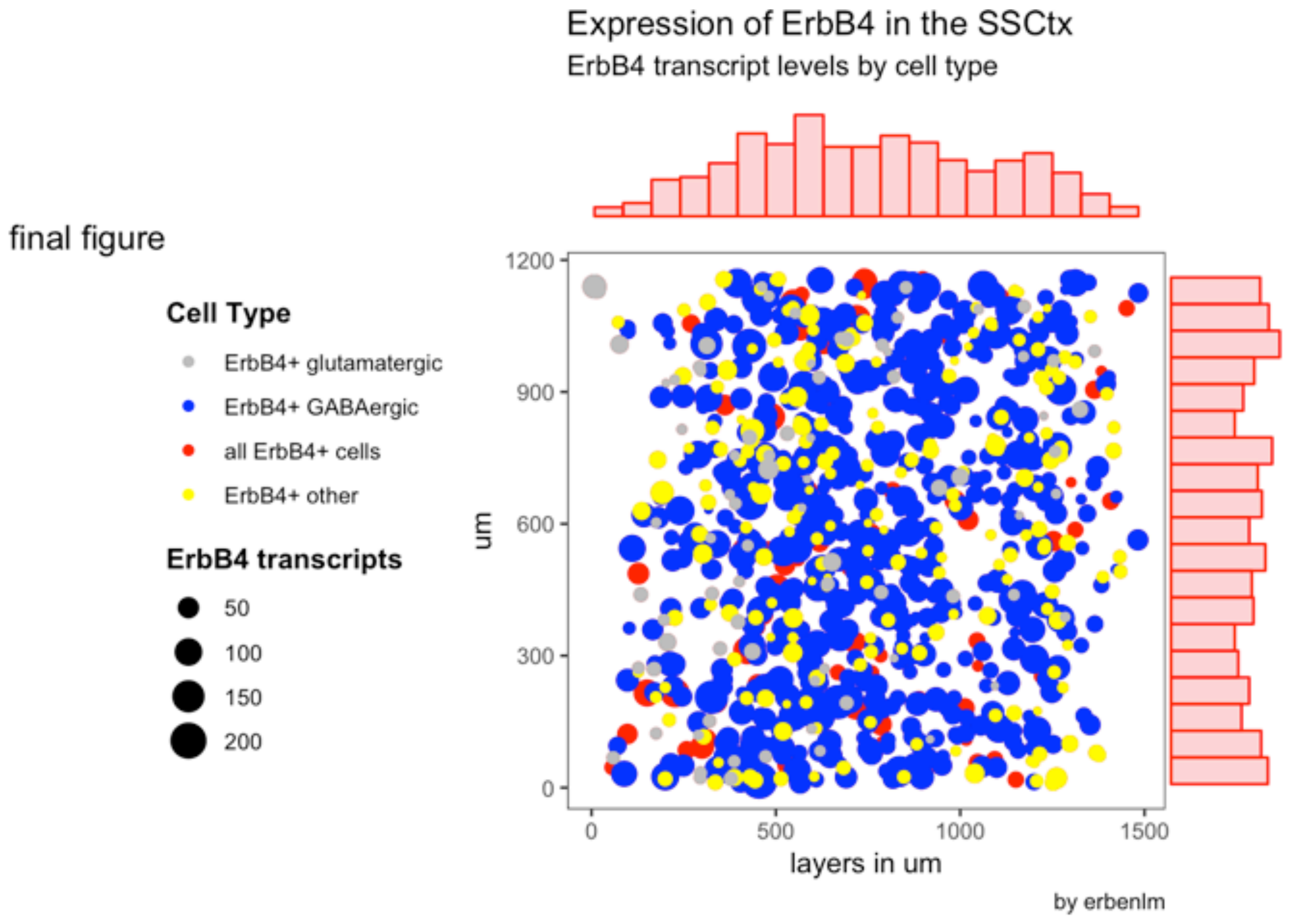


```
pdf(file="layer2.pdf", width=8, height=6) ##prints pdf
print(p2)
dev.off()
```

```
## quartz_off_screen
##                 2
```

I also found this option with ggMarginal in the ggExtra package that allows to plot a main plot and small plots at the axes. It contains a lot of information at the same time: position of cells by cell type in the cortex, histogram distribution per layer and transcript levels per cell. Preivoulsy, I plotted a similar graph where size of the dot varied with the transcripts expressed. However, I had to adjust dot size individually in Prism and therefore was only able to do this for a very small area.

```
celllabels<-c('RedandWhiteCell'='ErbB4+ GABAergic','RedandGreenCell'='ErbB4+ glutamat
ergic','RedOnlyCell'='ErbB4+ other', 'RedCell'='all ErbB4+ cells')
p=ggplot(cellsonly,aes(x=Xum_norm,y=Yum_norm, color=CellType, size=Children_Red_Count
))+
  geom_point()+
  scale_color_manual(values=cellcolors, labels=celllabels)+
  theme_test()+
  labs(x='layers in um', y='um')+
  theme(legend.position='left')+
  theme(legend.title = element_text (face = "bold")) +
  labs(caption='by erbenlm')+
  ggtitle('Expression of ErbB4 in the SSCtx')+
  labs(subtitle='ErbB4 transcript levels by cell type')+
  labs(color='Cell Type')+
  labs(size='ErbB4 transcripts')+
  labs(tag='final figure')
fencyplot=ggMarginal(p,type='histogram', color='red', alpha=0.2, fill='red', bins=20)
fencyplot
```



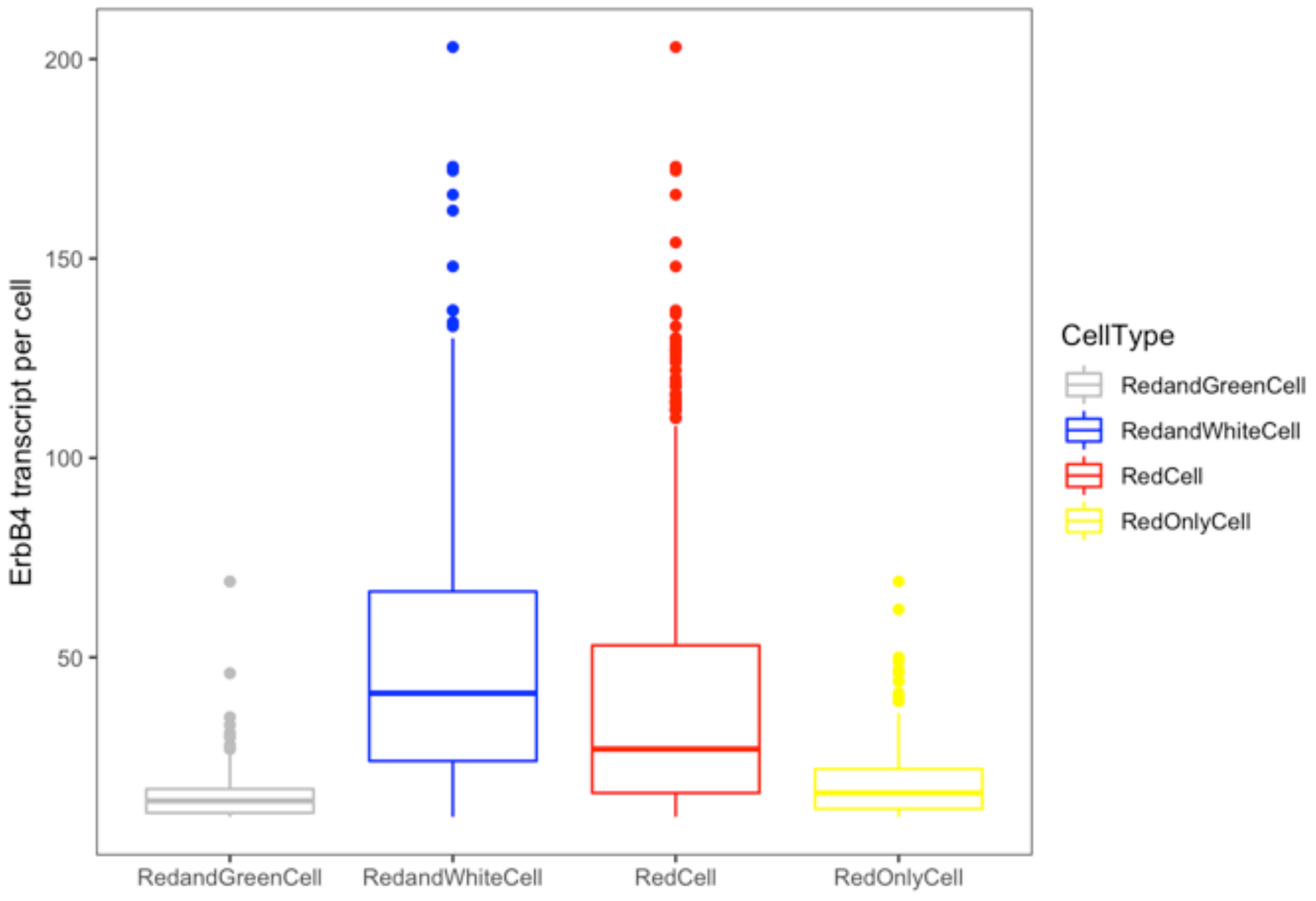
```
pdf(file="Expression in xy.pdf", width=8, height=6) ##prints pdf
print(fencyplot)
dev.off()
```

```
## quartz_off_screen
##                 2
```

7. Statistical test

I am testing if ErbB4+ GABAergic interneurons (RedandWhiteCell) do express more ErbB4 transcript than glutamatergic neurons found to be ErbB4+ (RedandGreenCell). I first tested for normality, since the data was not normally ditributed, a non-parametric t-test should be used. The expression levels are signficantly different (p<0.05).

```
p3=ggplot(cellsonly, aes(x = CellType, y = Children_Red_Count, color=CellType)) +
  geom_boxplot() +
  theme_test() +
  scale_color_manual(values=cellcolors) +
  labs(x='', y='ErbB4 transcript per cell')
p3
```



```
pdf(file="expressionpercell.pdf", width=8, height=6) ##prints pdf
print(p3)
dev.off()
```

```
## quartz_off_screen
##                 2
```

```
shapiro.test(rnorm(cellsonly$Children_Red_Count)) ##test for normal distribution
```

```
##
##  Shapiro-Wilk normality test
##
## data:  rnorm(cellsonly$Children_Red_Count)
## W = 0.99927, p-value = 0.6923
```

```
cellsonly_filtered<-cellsonly %>%  filter (CellType %in% c('RedandGreenCell','RedandW
hiteCell'))
wilcox.test(Children_Red_Count ~ CellType, data=cellsonly_filtered) ##non-parametric
for different medians, more appropriate in this case since data not normally distribu
ted
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  Children_Red_Count by CellType
## W = 5271, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```