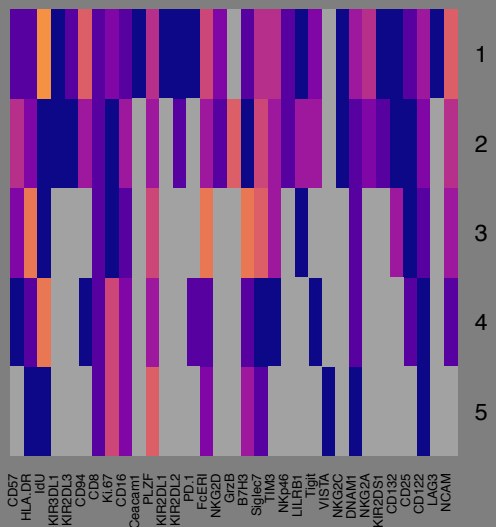
Nuffield Department of Clinical Neurosciences

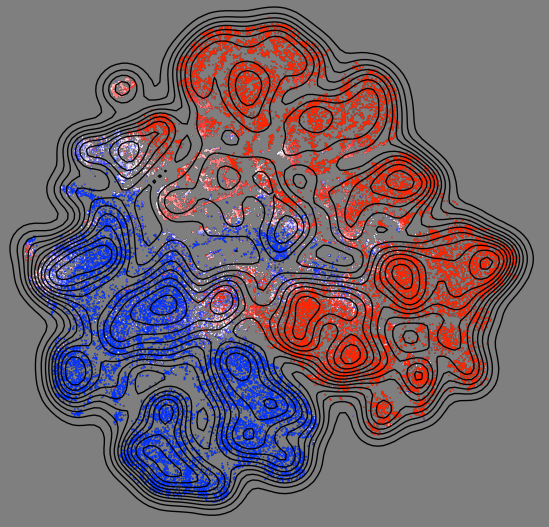
UNIVERSITY OF OXFORD

Building and publishing R packages -lessons learnt when building DepecheR

Jakob Theorell

Date 2019-09-16





UNIVERSITY OF OXFORD

Questions to you

- Who writes functions (in any language)?
- Who considers building a package?
- Who has built a package?
- Who has published a package (CRAN/BioConductor etc)

- Why build a package?

- Why publish a package?

UNIVERSITY OF OXFORD

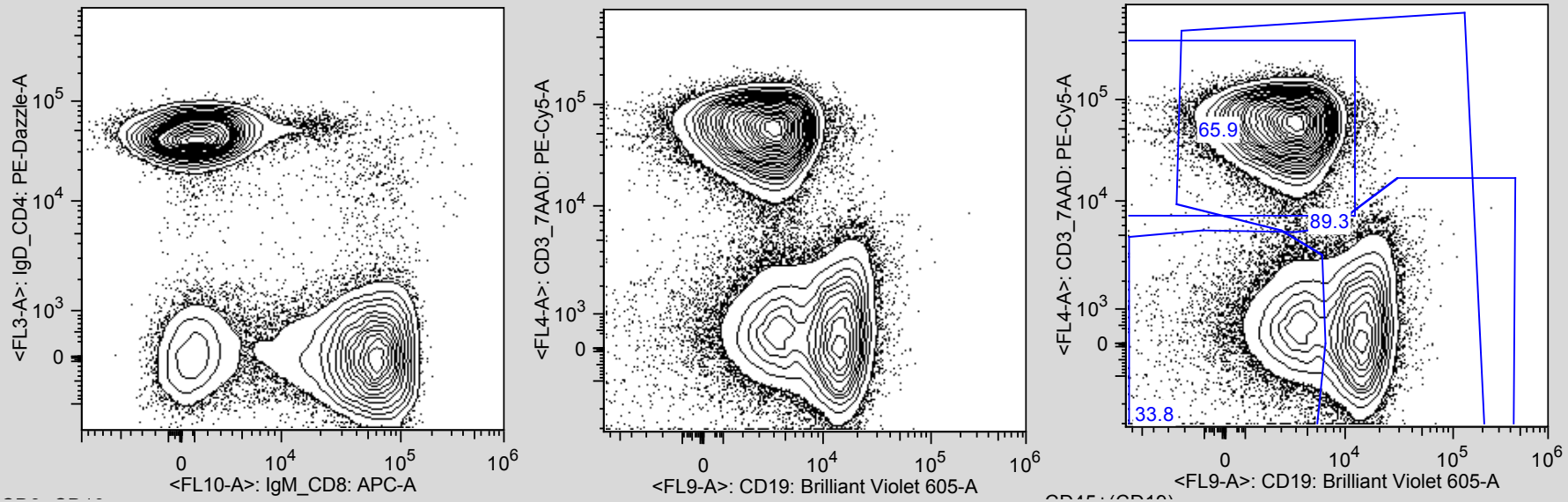
Why build a package

1. Store/keep track of functions
2. Traceability (git)
3. Structure
    1. Separation of data types
    2. Documentation
4. To increase reproducibility for publications

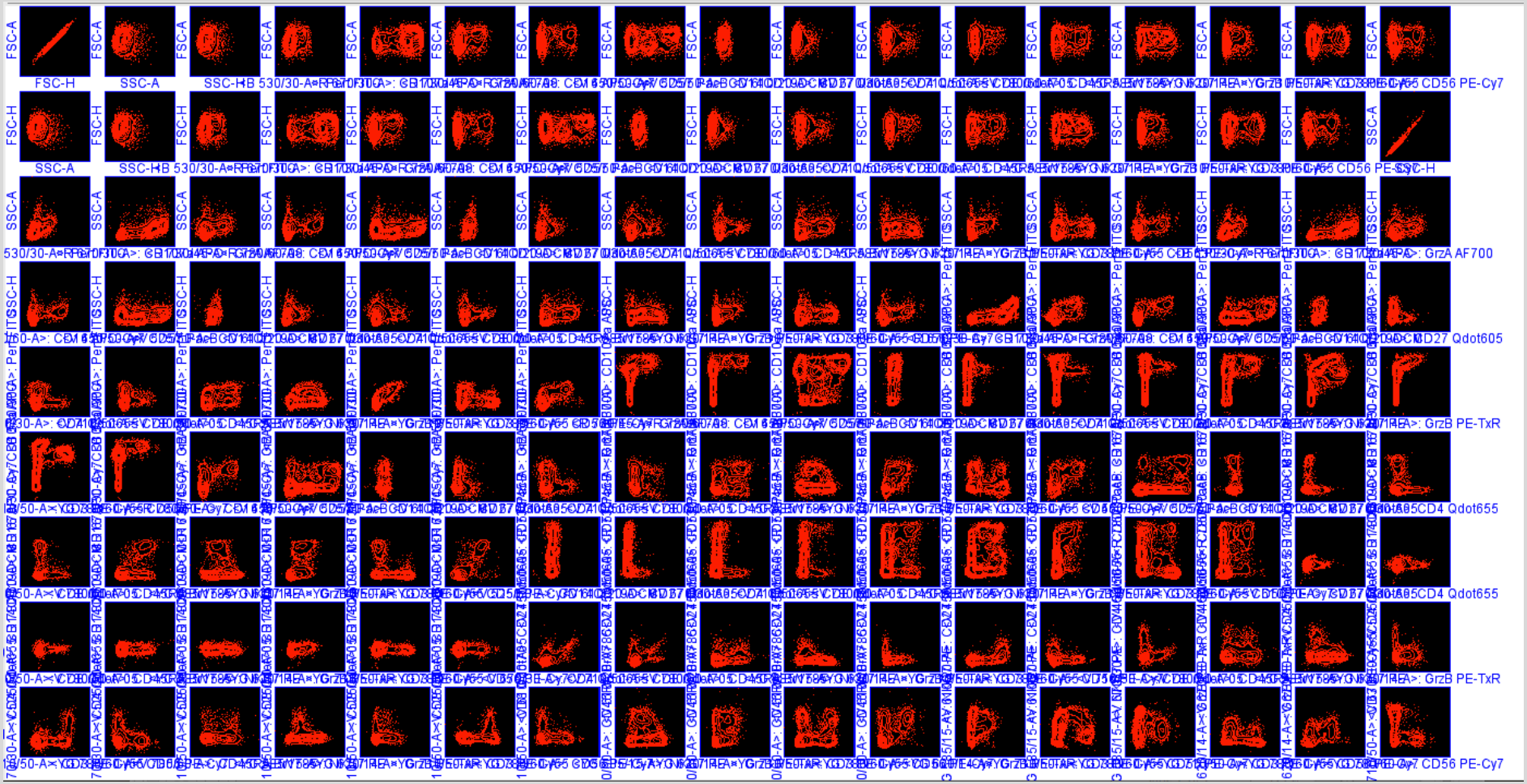
Why publish a package (with BioConductor)

1. External quality control
2. Forcing further testing/documentation/vinjettes
3. A quality stamp

Background - cytometry



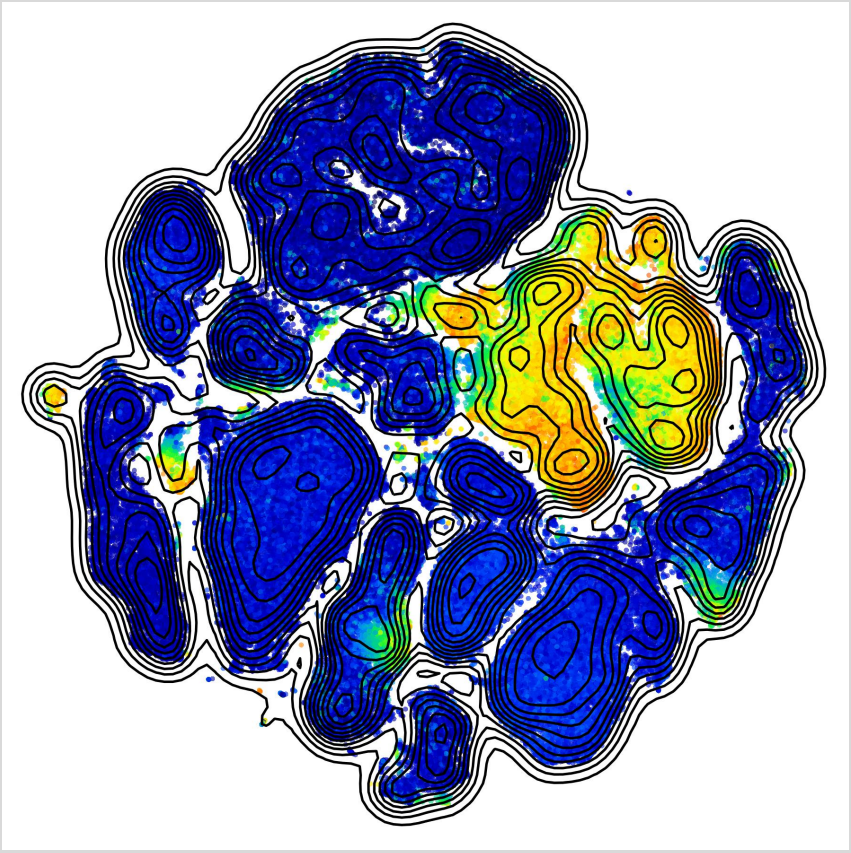
Background: the 2D problem



t-distributed stochastic neighbor embedding

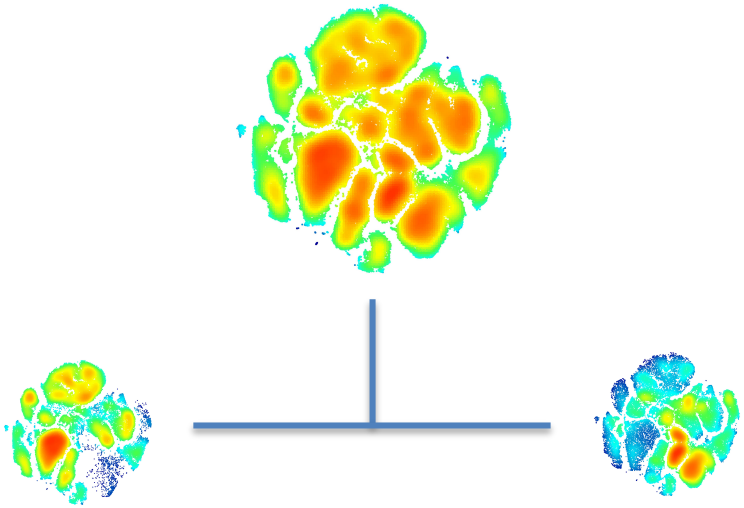
by Laurens van der Maaten

2013

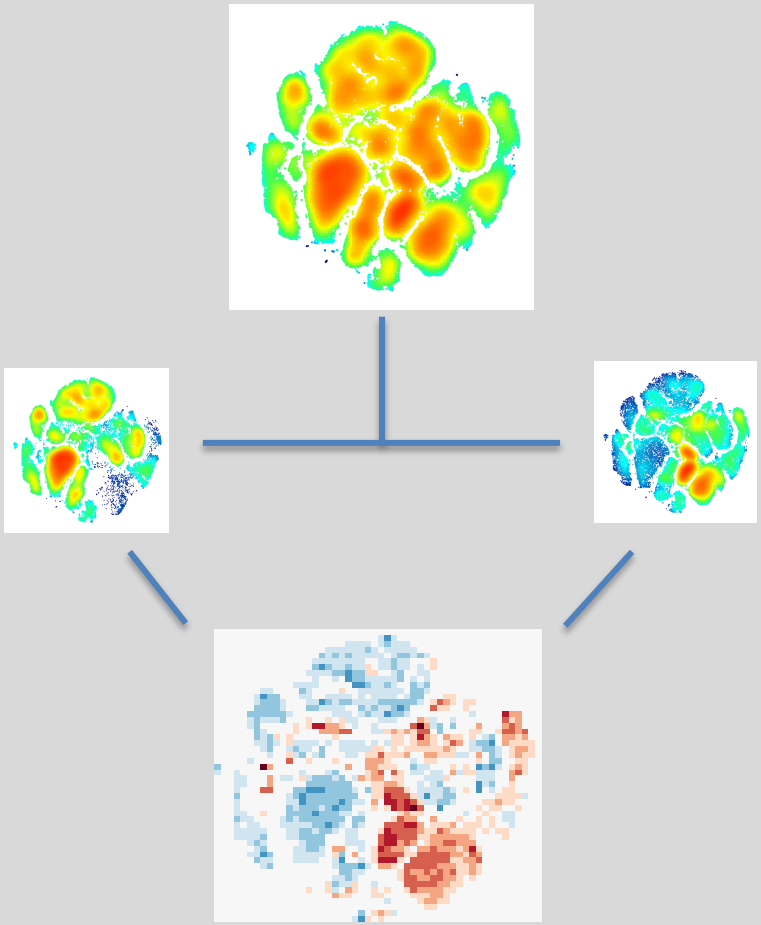


```
#11 Make contour plot of one parameter to see if the settings are correct

fname3 = paste("CD57",'.jpeg')

p <- ggplot(Full_all_data,aes(x=V1,y=V2)) +
      geom_point(col=(1+0.98*(102-Full_all_data$CD57)), size=1.3, alpha=0.6)     + geom_density2d(col="black", size=I(0.6)) + xlim(-5, 105) + ylim(-5, 105) +
      theme (line = element_blank(),
                  text = element_blank(),
                  line = element_blank(),
                  title = element_blank(),
                        panel.background = element_rect(fill = "white"))
gt <- ggplot_gtable(ggplot_build(p))
ge <- subset(gt$layout, name == "panel")
grid.draw(gt[ge$t:ge$b, ge$l:ge$r])

#12 Make contour plot of all the populations with color representing each csv column
for (i in 1:length(Full_all_data)){

    cat(i,"\n")
    graphics.off()

    #graphs
    fname3 = paste(colnames(Full_all_data[i]),'.jpeg')

    p <- ggplot(Full_all_data,aes(x=V1,y=V2)) +
      geom_point(col=(1 + 0.98*(102-Full_all_data[[i]])), size=1.2, alpha=0.6)    + geom_density2d(col="black", size=I(0.6)) + xlim(-5, 105) + ylim(-5, 105) +
      theme (line = element_blank(),
                  text = element_blank(),
                  line = element_blank(),
                  title = element_blank(),
                        panel.background = element_rect(fill = "white"))
      gt <- ggplot_gtable(ggplot_build(p))
      ge <- subset(gt$layout, name == "panel")

      grid.draw(gt[ge$t:ge$b, ge$l:ge$r])

    #Save file
    ggsave(filename = fname3, dpi=300)
}
```

Taking groups into consideration



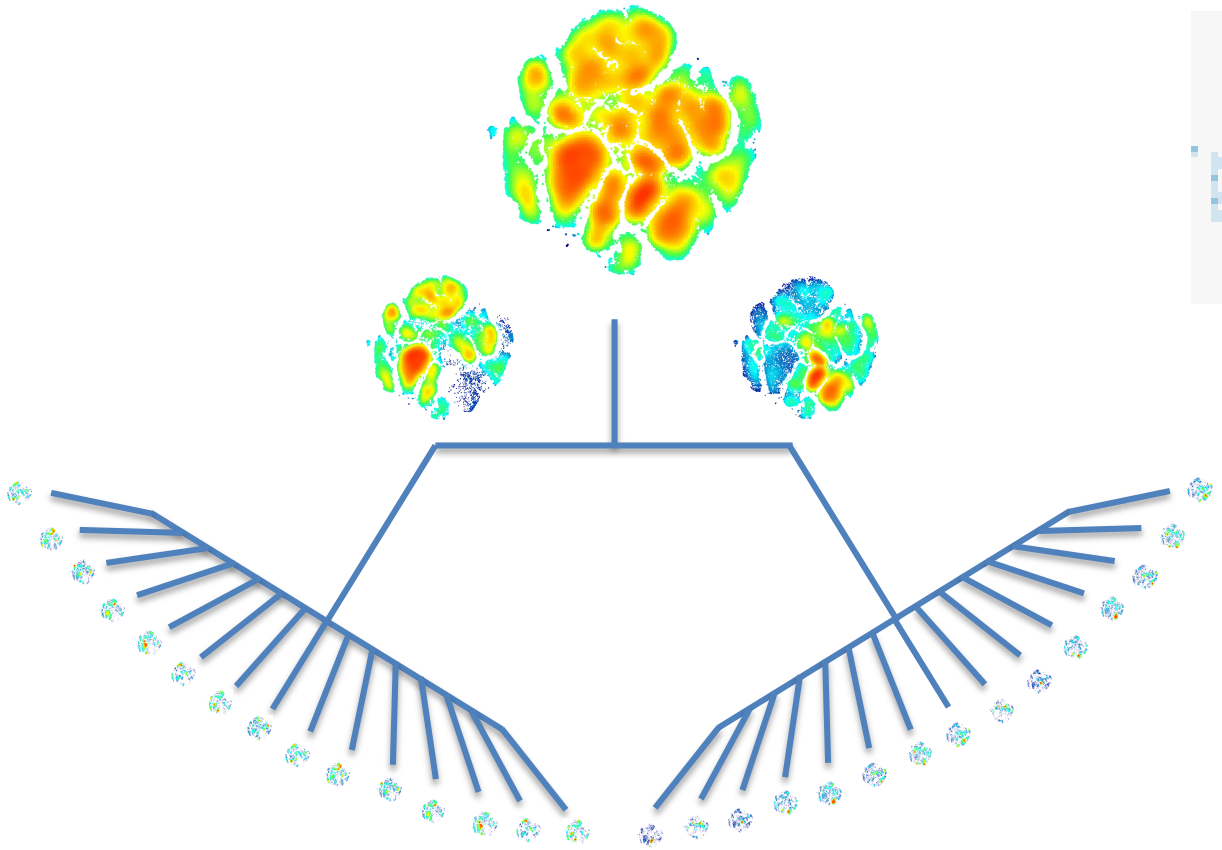
Plotting residuals

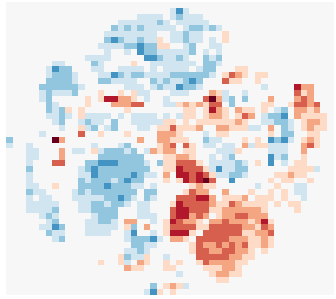


2014

```
#This script first makes a 2D histogram and then compares two 2Dhistograms to each other.
library(RColorBrewer)

setwd("/Users/Samuelchiang/Desktop/")
Sample  <- read.csv("Sample.csv", header=TRUE)
Control <- read.csv("Control.csv", header=TRUE)

#For PBMC
nbins <- 50
x.bin <- seq(floor(min(Control$V1)), ceiling(max(Control$V2)), length=nbins)
y.bin <- seq(floor(min(Control$V1)), ceiling(max(Control$V2)), length=nbins)

freq <-  as.data.frame(table(findInterval(Control$V1, x.bin),findInterval(Control$V2, y.bin)))
freq[,1] <- as.numeric(freq[,1])
freq[,2] <- as.numeric(freq[,2])

freq2DControl <- diag(nbins)*0
freq2DControl[cbind(freq[,1], freq[,2])] <- freq[,3]

#For Liver
nbins <- 50
x.bin <- seq(floor(min(Sample$V1)), ceiling(max(Sample$V1)), length=nbins)
y.bin <- seq(floor(min(Sample$V2)), ceiling(max(Sample$V2)), length=nbins)

freq <-  as.data.frame(table(findInterval(Sample$V1, x.bin),findInterval(Sample$V2, y.bin)))
freq[,1] <- as.numeric(freq[,1])
freq[,2] <- as.numeric(freq[,2])

freq2DSample <- diag(nbins)*0
freq2DSample[cbind(freq[,1], freq[,2])] <- freq[,3]

#Normalizing the matrices to each other so that their sizes are comparable.

sum.freq2D <-sum(freq2DSample)+sum(freq2DControl)

constantSample <- sum(freq2DSample)/sum.freq2D
constantControl <- (sum(freq2DControl))/sum.freq2D

Sample2D <- freq2DSample*constantSample
```

Taking individuals into consideration

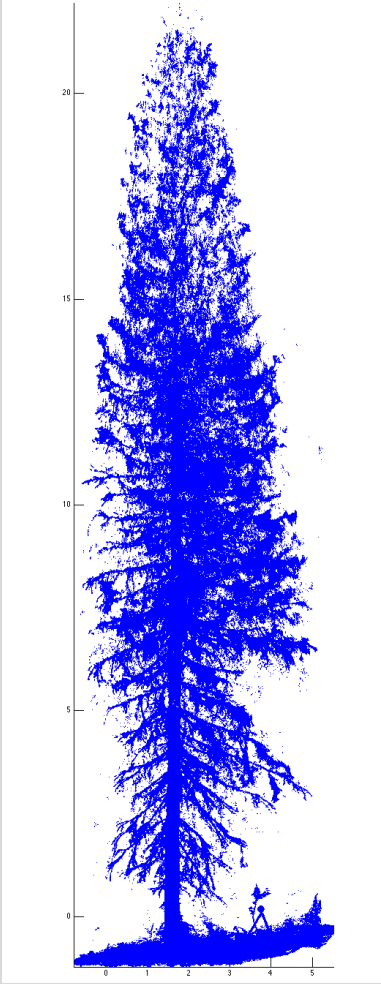


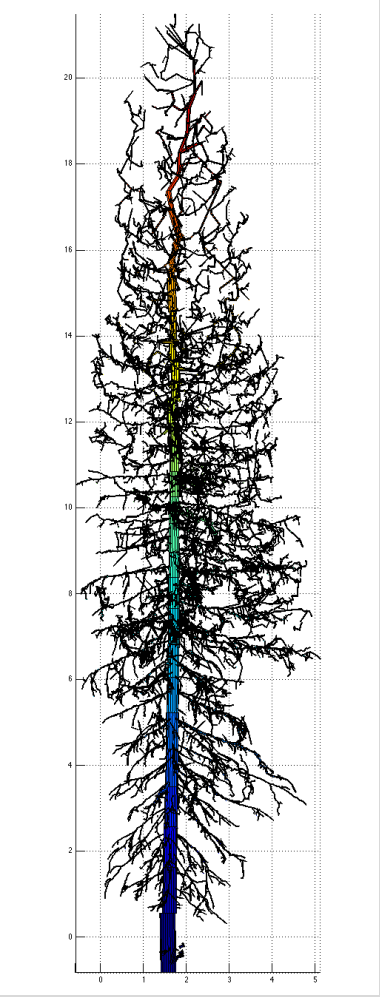


2016

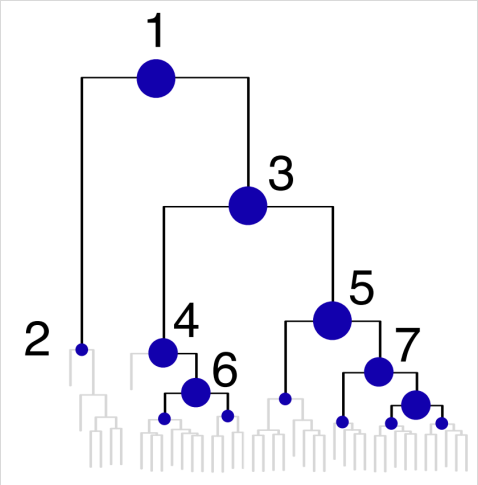
```
#Code to most functions
source('/users/jakobtheorell/Desktop/Labbet/R/Pre_post_SNE_functions_versions/160209_functions_pre_post_SNE.r')
```

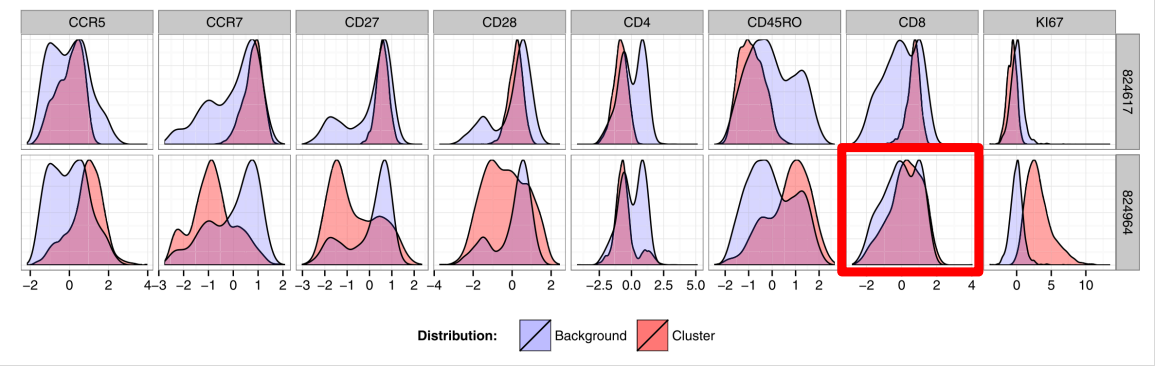
Result collaboration 1





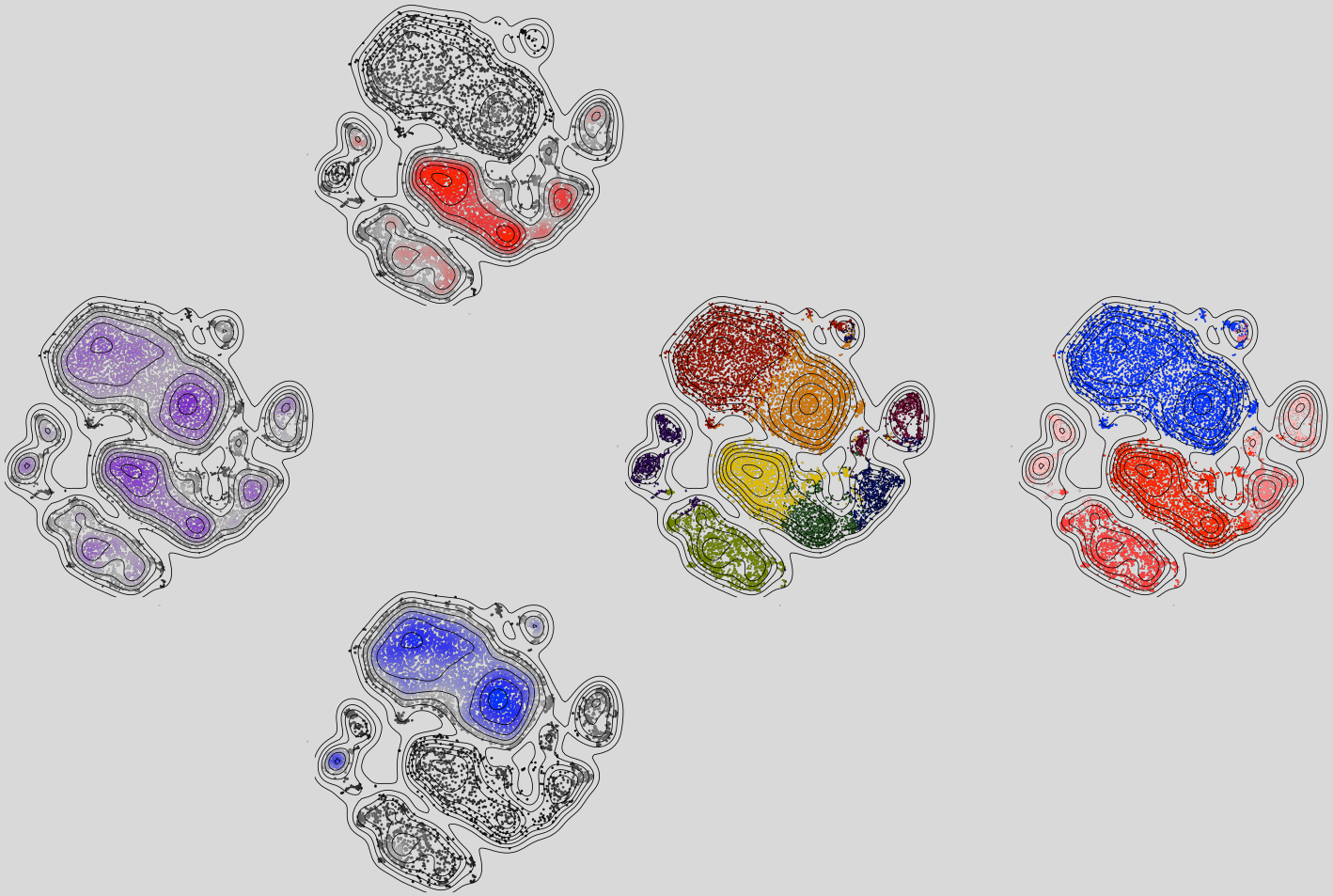
Previous solutions

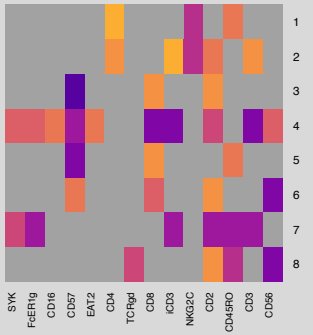


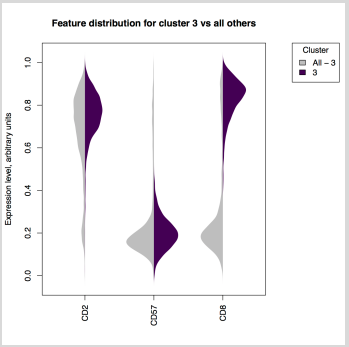


CITRUS, Bruggner et al, PNAS 2014

Our solution



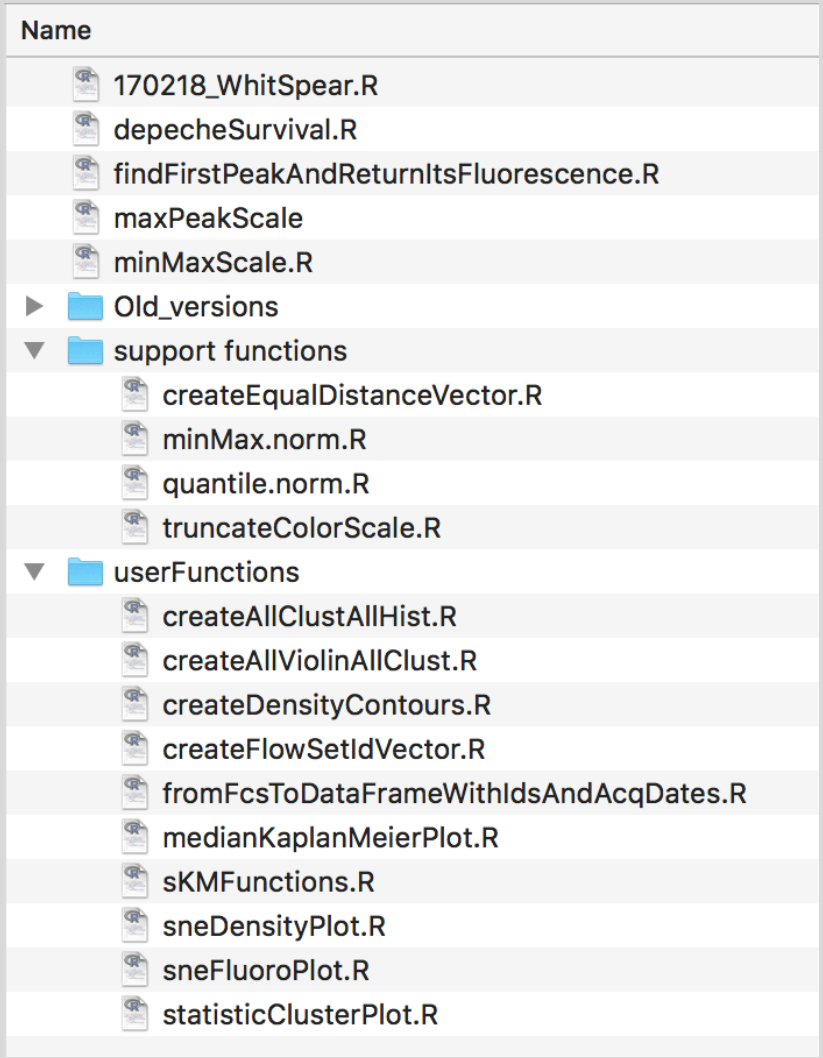




Theorell J, Theorell, A:
DepecheR, BioConductor, 2019

Time to start building a package?

2017:




```
statisticClusterPlot2 <- function(data, sneData, densContourPresent=TRUE, densContour, sneDataForDensCountour, name, method=c("coxph"), title=FALSE, dotsize=400/sqrt(length(data)),
 highestAbsValues=max(abs(data)), bandColor="black"){

directory <- getwd()
dir.create(paste(directory, "/statisticClusterPlot", sep=""))
setwd(paste(directory, "/statisticClusterPlot", sep=""))

normSneData <- minMax.norm.df(sneData)
colnames(normSneData) <- c("V1", "V2")

data.df <- as.data.frame(data)

#Make a color vector with the same length as the data

#make a breaks vector to define each bin for the colors
brks <- with(data.df, seq(-highestAbsValues, highestAbsValues, length.out = 22))

#assign each value to a bin
grps <- with(data.df, cut(data.df[,1], breaks = brks, include.lowest = TRUE))

colors <- colorRampPalette(c("#FF0000",  "white","#0000FF"))(21)

normSneData$col <- rev(colors)[grps]

#If there is no matrix present to construct the contour lines, create the density matrix for all_data to make them.
if(densContourPresent==FALSE){
densContour <- createDensityContours(sneDataForDensCountour)
}

if(title==TRUE){
    jpeg(paste(name,'.jpeg', sep=""), width = 2500, height = 2500, units = "px")
plot(V2~V1, data=normSneData, main=name, pch=20, cex=dotsize, cex.main=5, col=col, xlim=c(-0.05, 1.05), ylim=c(-0.05, 1.05), axes=FALSE, xaxs="i", yaxs="i")
par(fig=c(0,1,0,1), mar=c(6,4.5,4.5,2.5), new=TRUE)
contour(x=densContour$x, y=densContour$y, z=densContour$z, xlim=c(-0.05, 1.05), ylim=c(-0.05, 1.05), nlevels=10, col=bandColor, lwd=8, drawlabels = FALSE, axes=FALSE, xaxs="i", yaxs="i")

dev.off()
}

if(title==FALSE){
    jpeg(paste(name,'.jpeg', sep=""), width = 2500, height = 2500, units = "px")
plot(V2~V1, data=normSneData, main="", pch=20, cex=dotsize, cex.main=5, col=col, xlim=c(-0.05, 1.05), ylim=c(-0.05, 1.05), axes=FALSE, xaxs="i", yaxs="i")
par(fig=c(0,1,0,1), mar=c(6,4.5,4.5,2.5), new=TRUE)
contour(x=densContour$x, y=densContour$y, z=densContour$z, xlim=c(-0.05, 1.05), ylim=c(-0.05, 1.05), nlevels=10, col=bandColor, lwd=8, drawlabels = FALSE, axes=FALSE, xaxs="i", yaxs="i")

dev.off()
}



#Create a color legend with text
if(method=="coxph"){
    yname <- "Wald values"
    topText <- "Higher risk"
    bottomText <- "Lower risk"
    legendTitle <- paste("Color scale for", name, ".pdf", sep="")
}
pdf(legendTitle)
par(fig=c(0.35,0.65,0,1), xpd=NA)
z=matrix(1:21,nrow=1)
x=1
y=seq(-highestAbsValues,highestAbsValues,len=21)
image(x,y,z,col=rev(colors),axes=FALSE,xlab="",ylab=yname)
axis(2)
text(1,highestAbsValues*1.13, labels=topText, cex=1.1)
text(1,highestAbsValues*-1.13, labels=bottomText, cex=1.1)
box()
dev.off()

}
```

UNIVERSITY OF OXFORD

Let us build one!

Speeding up code

1. Before going to parallelization, check if any processes are inefficient:
    1. Check if there is a vectorized version of the function you plan to use.
    2. for loops are inefficient in R
    3. No iteration of identical processes within loop
    4. Keep objects within apply as small as possible
    5. Read Hadley Wickams “Advanced R”-chapters on the subject
    6. Check package lineprof by Hadley Wickam

#	Source code	t	r	a	d
1	`f <- function() {`				
2	`  pause(0.1)`				
3	`  g()`				
4	`  h()`				
5	`}`				
6	`g <- function() {`				
7	`  pause(0.1)`				
8	`  h()`				
9	`}`				
10	`h <- function() {`				
11	`  pause(0.1)`				
12	`}`				

UNIVERSITY OF OXFORD

Parallelization options

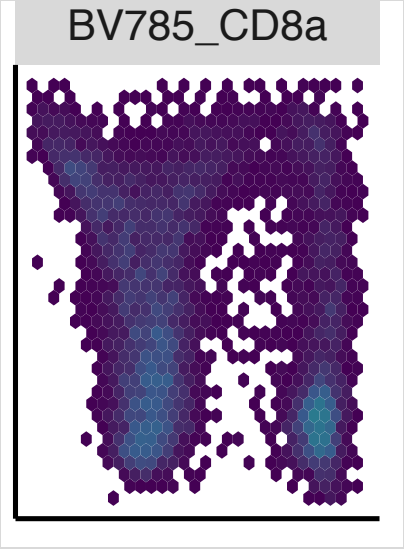
1. Many alternatives
    1. Experiences?
2. BiocParallel function bplapply preferred (by BioC).
    - Experiences?
- Foreach might be cleaner
    - forks only for Unix
        - complicated?

UNIVERSITY OF OXFORD

Publishing a package with BioConductor

1. In addition to the CMD check, BiocCheck:
   - Formatting
   - No errors or warnings
2. Upon upload, system checks all platforms.
   - Generally; if one fails, all fails.
3. Formal review
   - Coding practice
   - Usage of S4 classes
   - If package adds to BioConductor
   - If tests are implemented

A call for help!



$$\mathcal{D} = 2\mathbf{j}^T(\mathbf{r} \circ \log(\mathbf{r} \circ \overline{\mathbf{M}\boldsymbol{\alpha}}) - (\mathbf{r} - \mathbf{M}\boldsymbol{\alpha})),$$

Oxford Autoimmune Neurology Group

UNIVERSITY OF OXFORD

Thank you all!