Sample crack analysis for a soil from spain

Liu et al. 2019

Table of Contents

Version from Mon Dec 09 19:44:03 2019

This document is written in rmd format (R markup language), if you want to test the program use the following procedure:

# Software requirements

The programming environment requirement for this analysis is:

* R (current version 3.6.2)
* R Studio
* the libraries defined in this document

To reproduce the analyse from the paper carry out the following steps

* install the software
* load crack\_analysis.rmd in RStudio and run it (“knit”).
* compare the results with the ones from the archive

To apply the code to your own problems proceed as follows:

* take the pictures are described in the paper
* adjust the parameters manually until the structure of the cracks is clearly visible

# Crack analysis

## Prepare data filenames and data structures

# load filesnames  
fnames = dir(path = "./", pattern = "\*.jpg") # get jpg filenames  
#t3 = gsub("\_", "-", fnames) # replace \_ by -  
#t3=fnames  
  
# split names into components  
# part 1 is soil (or any text)  
# part 2 is date  
# part 3 is time  
# part 4 is anything like repetition  
  
t = matrix(unlist(strsplit(fnames, "-")), ncol = length(fnames))  
t2 = as.data.frame(t(t))  
  
# extract number of soils, only one soil   
nsamples = levels(t2$V1)  
fnames = as.data.frame(fnames)  
fnames$Sample = t2$V1  
  
# prepare the empty data structure  
pores = data.frame(  
 s.area = numeric(),  
 s.perimeter = numeric(),  
 s.radius.mean = numeric(),  
 s.radius.sd = numeric(),  
 s.radius.min = numeric(),  
 s.radius.max = numeric(),  
 soil = numeric(),  
 fname = character(),  
 date = character()  
)  
soilsample = pores  
  
image\_list = droplevels(fnames$fnames)  
date\_list = as.character(t2$V2)  
t5 = data.frame(date\_list)  
# convert text date to POSIXct  
date = as.POSIXct(date\_list, format = "%Y%m%d")  
#vlist = t2

## Loop for analysis

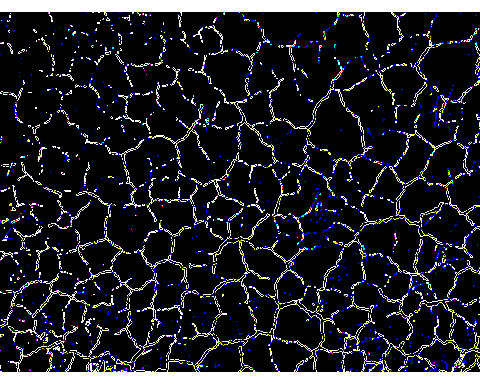
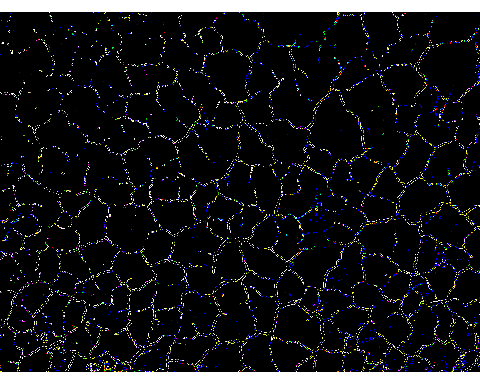
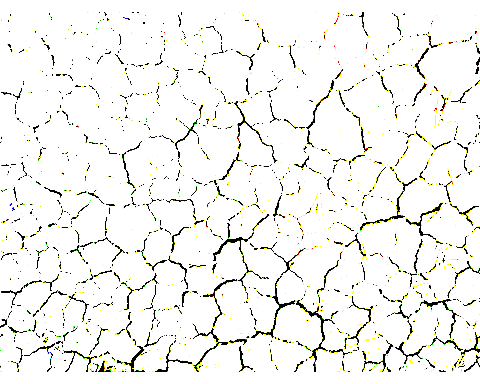
This is the loop where the images are analysed - currently only for one soil, but the data structure is already prepared for several soils. The pictures have different quality to show also the limits of the method.

Practical hints

* the ‘display’ and ‘writeImage’ functions are useful for debugging but the pictures need a lot of space. Uncomment if everything is running smoothly
* the final versions of the commands are executed, the comment lines contain parameters used in other analysis as an indicator of the possible range.

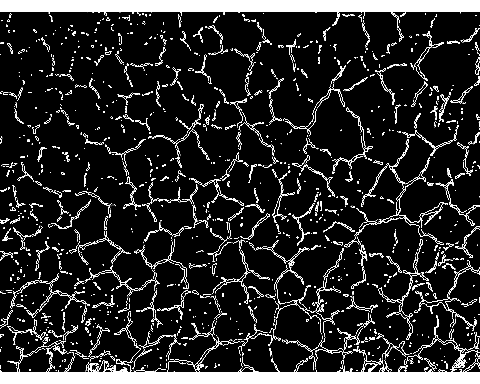
for (j in image\_list) {  
 # j=image\_list[1] # for debugging single images  
 print(j)  
 k = paste("./", j, sep = "")  
 img = readImage(k)  
 display(img, method = "raster")  
 # nuc\_gblur = gblur(img ^ 0.2, sigma = 5) # original value  
 # nuc\_gblur = gblur(img ^ 0.6, sigma = 3) # original value  
   
 nuc\_gblur = gblur(img ^ 0.76, sigma = 3) # original value  
   
 #nuc\_gblur = gblur(img ^ 0.1, sigma = 2)  
 # display(nuc\_gblur, method = "raster")  
 outname = sub('\\.jpg$', '', as.character(j))  
 outname = paste("results/", outname, sep = "")  
   
 writeImage(nuc\_gblur, paste(outname, "-img\_gblur.jpg", sep =  
 ""))  
 # select range for crack  
 # original value: 0.85 China  
 img\_cut = (nuc\_gblur > 0.43)  
 display(img\_cut, method = "raster")  
 writeImage(img\_cut, paste(outname, "-img\_cut.jpg", sep = ""))  
   
 # create binary mask from image  
 # y = thresh(img\_cut, 10, 10, 0.05) original  
 # y = thresh(img\_cut, 5, 5, 0.05)  
 y = thresh(img\_cut, 3, 3, 0.05)  
   
 display(y, title = 'Binary mask for cracks')  
 writeImage(y, paste(outname, "-binary.jpg", sep = ""))  
   
 brushno = 5  
 b1 = makeBrush(brushno, shape = 'disc')  
 y2 = dilate(y, b1)  
 # display(y2, title = 'Binary mask for cracks 1')  
 writeImage(y2, paste(outname, "-bin2.jpg", sep = ""))  
   
 t3 = bwlabel(y2)  
 display(t3, title = 'Binary mask for cracks 2')  
 writeImage(t3, paste(outname, "-bin3.jpg", sep = ""))  
   
 max(t3)  
 y = channel(t3, "luminance")  
 # y = computeFeatures.shape(y)# to greyscale  
 # display(y, title = "Cell nuclei")  
 writeImage(y, paste(outname, "-nuclei.jpg", sep = ""))  
 display(y, title = 'Greyscale')  
   
 fts = computeFeatures.shape(y)  
 # fts is the basic result matrix of EBImage  
 # output is in a weird matrix format  
 # convert to data frame and add factor variables for each image  
 prop = data.frame(fts)  
 prop$file = as.character(j)  
 prop$date = date[1]  
 date = date[-1] # remove first line  
 #vlist = vlist[-1, ] # rest variable  
 soilsample = rbind(soilsample, prop)  
 nprop = melt(soilsample, id.vars = c("file", "date"))  
 header = paste("Soil No. 1")  
 qplot(data = nprop, x = value) +  
 facet\_grid(as.factor(date) ~ variable, scales = "free") +  
 ggtitle(header)  
 ggsave(paste("results/charts/", j, "-pic2.jpg", sep = ""))  
   
 gp = soilsample[, c(1:6, 8)]  
 gp$date = as.factor(gp$date)  
   
 k1 = group\_by(nprop, date, variable)  
 k2 = summarize(  
 k1,  
 mean = mean(value),  
 median = median(value),  
 q95 = quantile(value, 0.95),  
 q10 = quantile(value, 0.1)  
 )  
   
 k3 = melt(k2, id.var = c("date", "variable"))  
 names(k3)[2] = "parameter"  
   
 qplot(  
 data = k3,  
 y = value,  
 x = date,  
 col = variable,  
 geom = "point"  
 ) +  
 facet\_wrap(~parameter, scales = "free",ncol=2) +  
 ggtitle(header)  
 ggsave(  
 paste("results/charts/", j, "-summary.jpg", sep = ""),  
 unit = "mm", width = 200, height = 200  
 )  
   
 qplot(data = soilsample, x = s.area) +  
 facet\_grid(date ~ .)  
   
 pores = rbind(pores, soilsample)  
}

## [1] "IMG-20180925-110428-1.jpg"

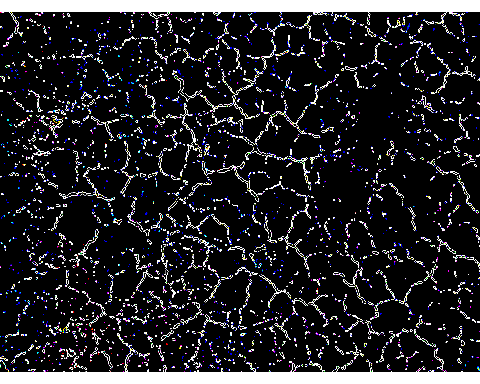
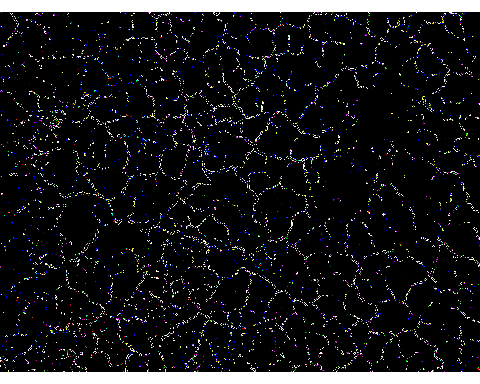
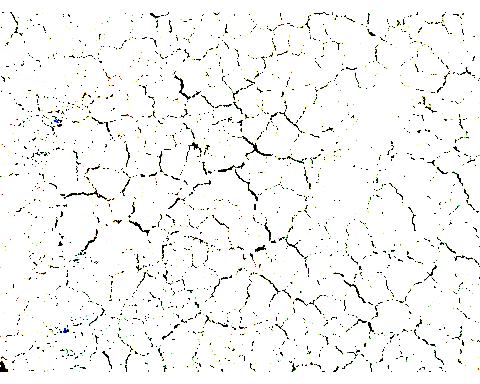


## Saving 5 x 4 in image

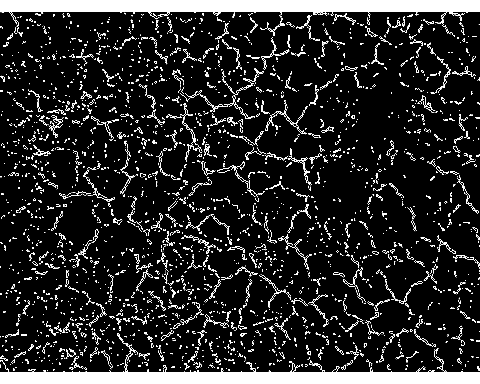
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



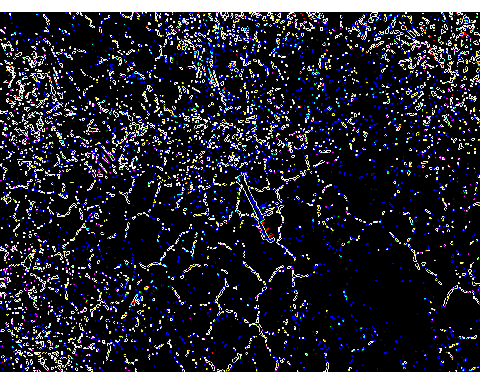
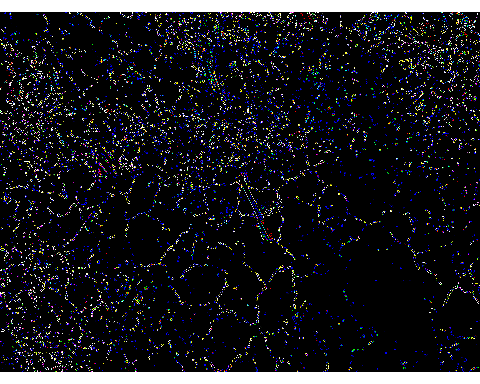
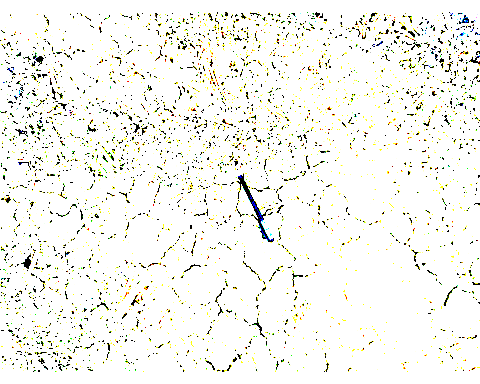
## [1] "IMG-20180925-110444-1.jpg"



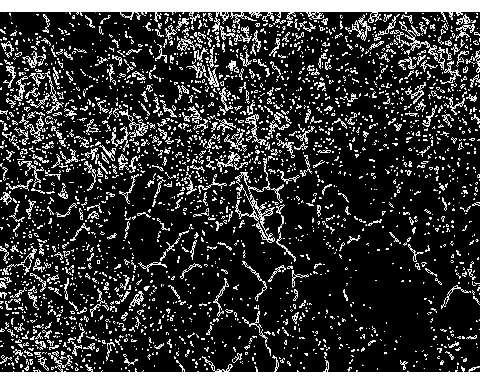
## Saving 5 x 4 in image  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



## [1] "IMG-20180925-110634-1.jpg"



## Saving 5 x 4 in image  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



save(pores, file = "results/pores.rdata")  
write\_xlsx(pores, path = "results/pores.xlsx")  
  
#### end loop ####

# cleaning up the pictures  
rm(img)  
rm(img\_cut)  
rm(nuc\_gblur)  
rm(t3,y2,y)

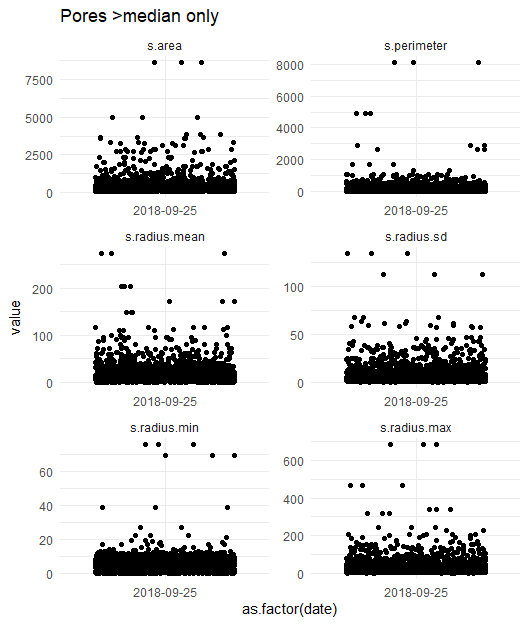
# Analysis of all results

The analysis from here on is just a suggestion, adapt to your own needs.

#### analysis of pore values ####  
# delete small pores below median, we only want cracks/big pores  
# attention: currently no adjustment for images sizes!  
# delete small pores  
limit = quantile(pores$s.area, 0.5)  
big\_only = pores[pores$s.area > limit, ]

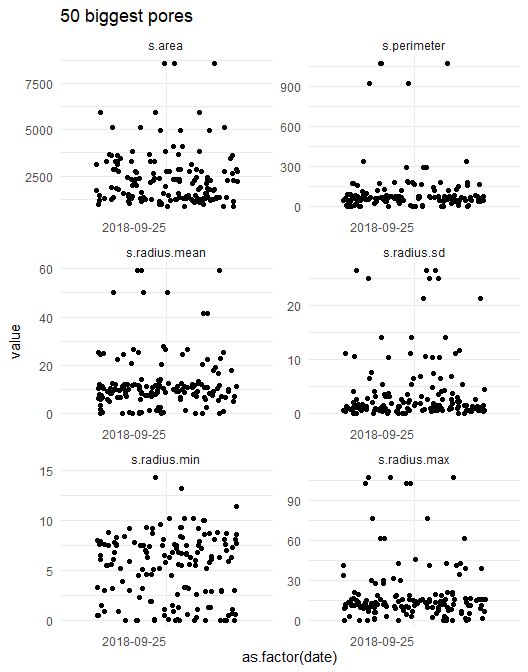
The definition of cracks used here is max. radius / min. radius >1.5.

#get cracks only  
cracks = big\_only[(big\_only$s.radius.max / big\_only$s.radius.min) > 1.5, ]  
  
cracksn = melt(cracks, id.vars = c("date", "file"))  
cracksn = cracksn[!is.na(cracksn$value),]  
  
qplot(data = cracksn, x = as.factor(date), y = value, geom="jitter") +  
 facet\_wrap(~ variable, scales = "free",ncol = 2) +  
 ggtitle("Pores >median only")



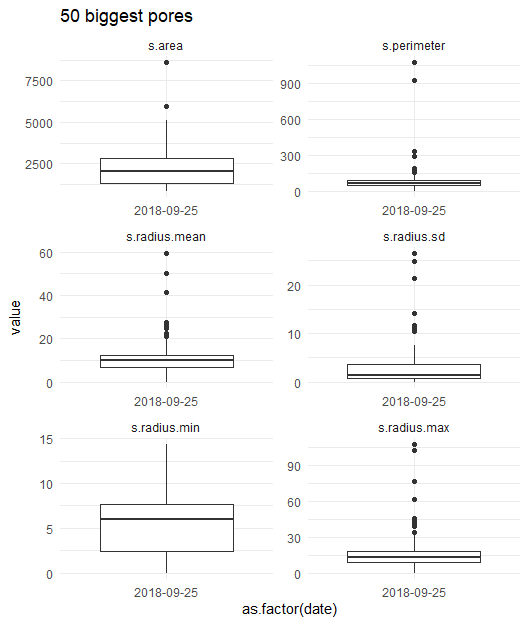
ggsave("results/charts/all\_crack\_only\_allvars.jpg",  
 unit = "mm",  
 width = 200,  
 height = 200)

#### get the biggest cracks for each file ####   
   
top50 = pores[0, ] # copy structure  
  
for (i in levels(as.factor(pores$file))) {  
 temp = pores[pores$file == i, ]  
 temp = temp[order(temp$s.area, decreasing = TRUE), ]  
 temp = temp[1:50, ] # 50 biggest pores  
 top50 = rbind(top50, temp)  
}  
  
top50n=melt(top50,id.vars=c("date","file"))  
  
qplot(  
 data = top50n,  
 x = as.factor(date),  
 y = value,  
 geom = "jitter"  
) +  
 facet\_wrap(~variable, scales = "free", ncol=2) +  
 ggtitle("50 biggest pores") +  
 theme(axis.text.x = element\_text(  
 angle = 0,  
 hjust = 1,  
 vjust = 0.5  
 ))



ggsave("results/charts/all\_crack\_top50\_allvars1.jpg",  
 unit = "mm",  
 width = 300,  
 height = 200)

qplot(data = top50n, x = as.factor(date), y = value, geom = "boxplot") +  
 facet\_wrap(~variable, scales = "free", ncol = 2) +  
 ggtitle("50 biggest pores")



# theme(axis.text.x = element\_text(angle = 0, hjust = 1, vjust = 0.5))  
  
ggsave("results/charts/all\_crack\_top50\_allvars2.jpg",  
 unit = "mm",  
 width = 500,  
 height = 300  
)

# Analysis of file IMG-20180925-110428-1.jpg

Sample analysis of a single file. One of the important parameters is the relation between max. and min. radius, calls “crackfactor” in the code.

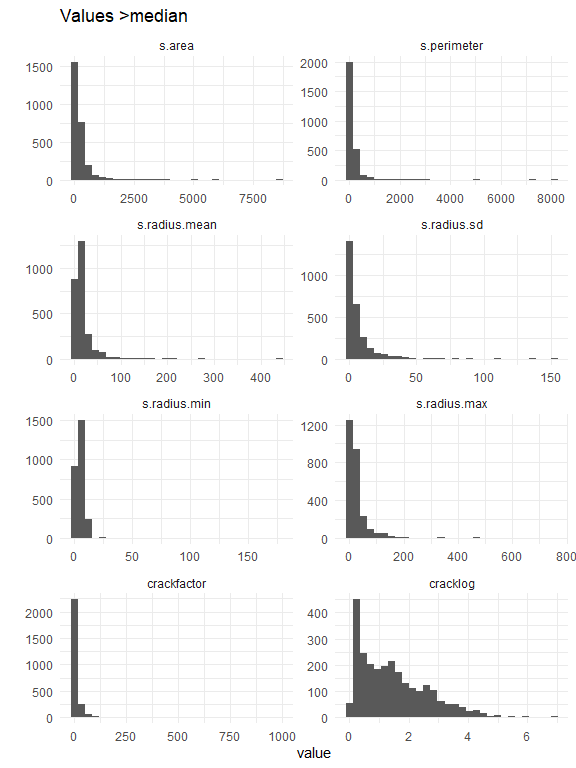
pores$crackfactor <- pores$s.radius.max / pores$s.radius.min  
pores$cracklog <- log(pores$s.radius.max / pores$s.radius.min)  
  
t1 <- pores[pores$file == "IMG-20180925-110428-1.jpg", ]  
t1 <- t1[, c(-8, -7)]

# plot all factors together  
  
t2 <- melt(t1)

## No id variables; using all as measure variables

pcomp <- t1 # for comparison  
pcomp$source <- "all"  
qplot(data = t2, x = value) +  
 facet\_wrap(~variable, scales = "free", ncol = 2)+  
 ggtitle("Values >median")

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

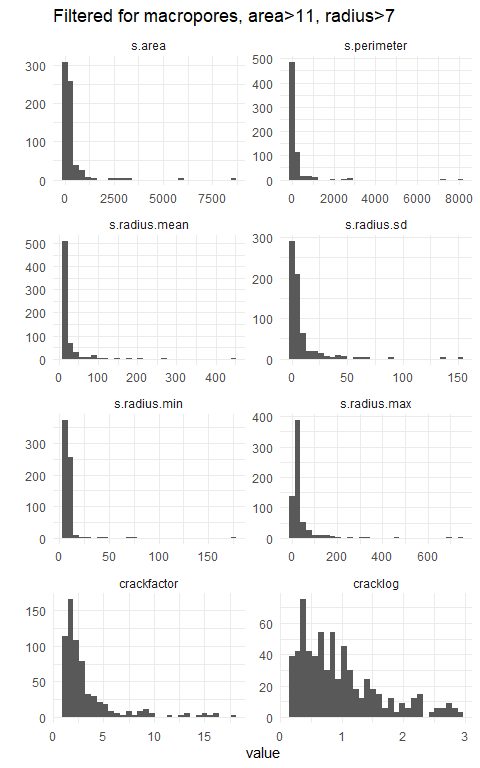


# filter for area > 11 and min.radius > 7  
t2\_filtered <- t1[(t1$s.area > 11) & (t1$s.radius.min > 7), ]  
t3 <- t2\_filtered  
t3$source <- "filtered"  
pcomp <- rbind(pcomp, t3)  
t2\_filtered <- melt(t2\_filtered)

## No id variables; using all as measure variables

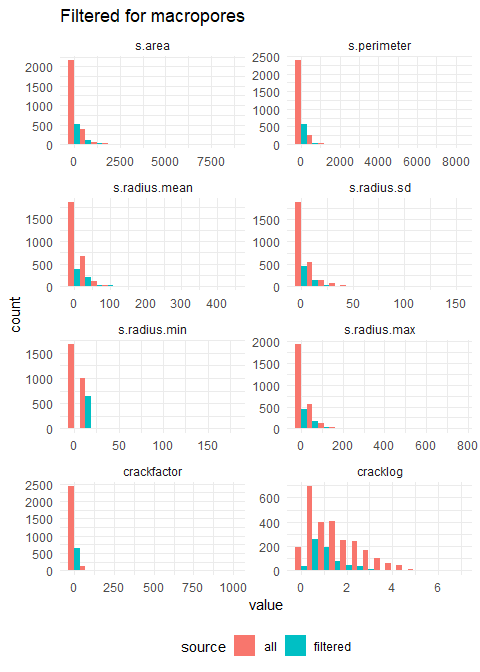
qplot(data = t2\_filtered, x = value) +  
 facet\_wrap(~variable, scales = "free", ncol = 2) +  
 ggtitle("Filtered for macropores, area>11, radius>7")

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



Now compare the properties of cracks and the other pores.

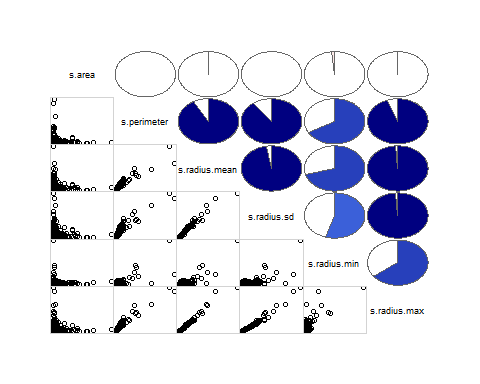
pcomp\_n <- melt(pcomp, id.vars = "source")  
  
ggplot() +  
 geom\_histogram(  
 data = pcomp\_n, aes(x = value, fill = source),  
 position = "dodge", bins = 15  
 ) +  
 facet\_wrap(~variable, scales = "free", ncol = 2) +  
 ggtitle("Filtered for macropores") +  
 theme(legend.position = "bottom")



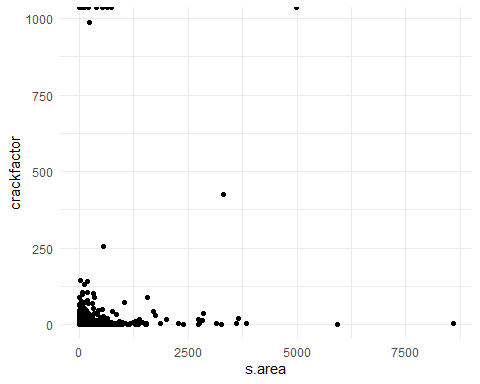
## Correlation analysis

Analyse the correlations between the variables, this part is work in progress.

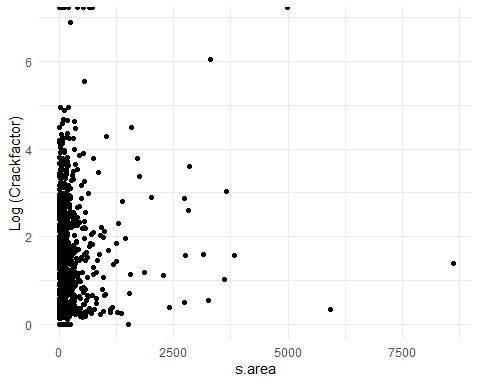
library(corrgram)  
t1 = t1[, c(-8, -7)]  
corrgram(t1, lower.panel = panel.pts, upper.panel = panel.pie)



t1 = pores[pores$file == "IMG-20180925-110428-1.jpg", ] # only one file  
t1 = t1[, c(-8, -7)]  
  
qplot(data = t1,  
 x = s.area,  
 y = crackfactor,  
 geom = "point")



qplot(data = t1,  
 x = s.area,  
 y = cracklog,ylab = "Log (Crackfactor)",  
 geom = "point")



Now calculate numeric summaries.

c = sum(t1$s.area)  
all = 4608 \* 3456 # size of image in pixel  
  
percent = c / all \* 100  
percent # percent crack area

## [1] 4.484389

t3 = group\_by(pcomp\_n, source, variable)  
summarise(t3, sum = sum(value, na.rm = TRUE) / all \* 100)

## # A tibble: 16 x 3  
## # Groups: source [2]  
## source variable sum  
## <chr> <fct> <dbl>  
## 1 all s.area 4.48   
## 2 all s.perimeter 2.82   
## 3 all s.radius.mean 0.292   
## 4 all s.radius.sd 0.111   
## 5 all s.radius.min 0.0909   
## 6 all s.radius.max 0.498   
## 7 all crackfactor Inf   
## 8 all cracklog Inf   
## 9 filtered s.area 1.34   
## 10 filtered s.perimeter 1.06   
## 11 filtered s.radius.mean 0.104   
## 12 filtered s.radius.sd 0.0356   
## 13 filtered s.radius.min 0.0436   
## 14 filtered s.radius.max 0.175   
## 15 filtered crackfactor 0.0142   
## 16 filtered cracklog 0.00402

Compare distribution of all values with pores >2mm.

cdata <- pcomp[, c(1, 8, 9)]  
cdata$source <- as.factor(cdata$source)  
f1 <- ggplot() +  
 geom\_histogram(  
 data = cdata, aes(x = s.area, fill = source),  
 position = "dodge", bins = 10  
 ) +  
 scale\_fill\_manual(name = "Filter", values = c("grey", "black"), breaks = c("all", "filtered"), labels = c("No filter", ">2mm")) +  
 theme\_minimal() +  
 xlab("Area (px)") +  
 theme(legend.position = "none")  
  
f2 <- ggplot() +  
 geom\_histogram(  
 data = cdata, aes(x = cracklog, fill = source),  
 position = "dodge", bins = 10  
 ) +  
 xlab("LOG (max/min width)") +  
 scale\_fill\_manual(name = "Filter", values = c("grey", "black"), breaks = c("all", "filtered"), labels = c("No filter", ">2mm")) +  
 theme\_minimal()  
# theme(legend.position = "bottom")  
  
grid.arrange(f1, f2, ncol = 2)

