Assignment 10

Lydia Strebe

May 7, 2019

Problem 1: PCA

Reconstructing a grayscale image using different number of principal components

```
library(imager)
```

```
## Warning: package 'imager' was built under R version 3.5.3
```

```
## Loading required package: magrittr
```

```
## Warning: package 'magrittr' was built under R version 3.5.2
```

```
##
## Attaching package: 'imager'
```

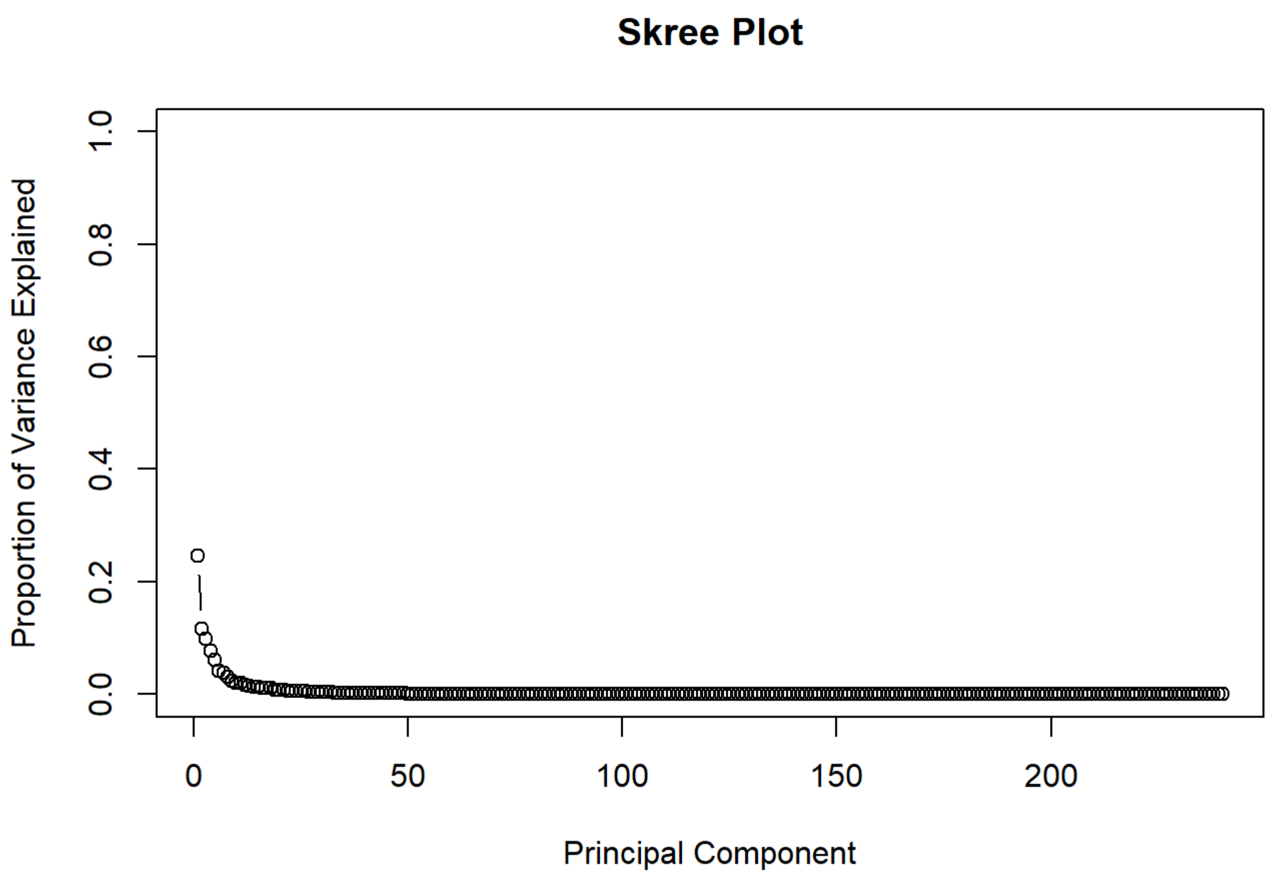
```
## The following object is masked from 'package:magrittr':
##
##     add
```

```
## The following objects are masked from 'package:stats':
##
##     convolve, spectrum
```

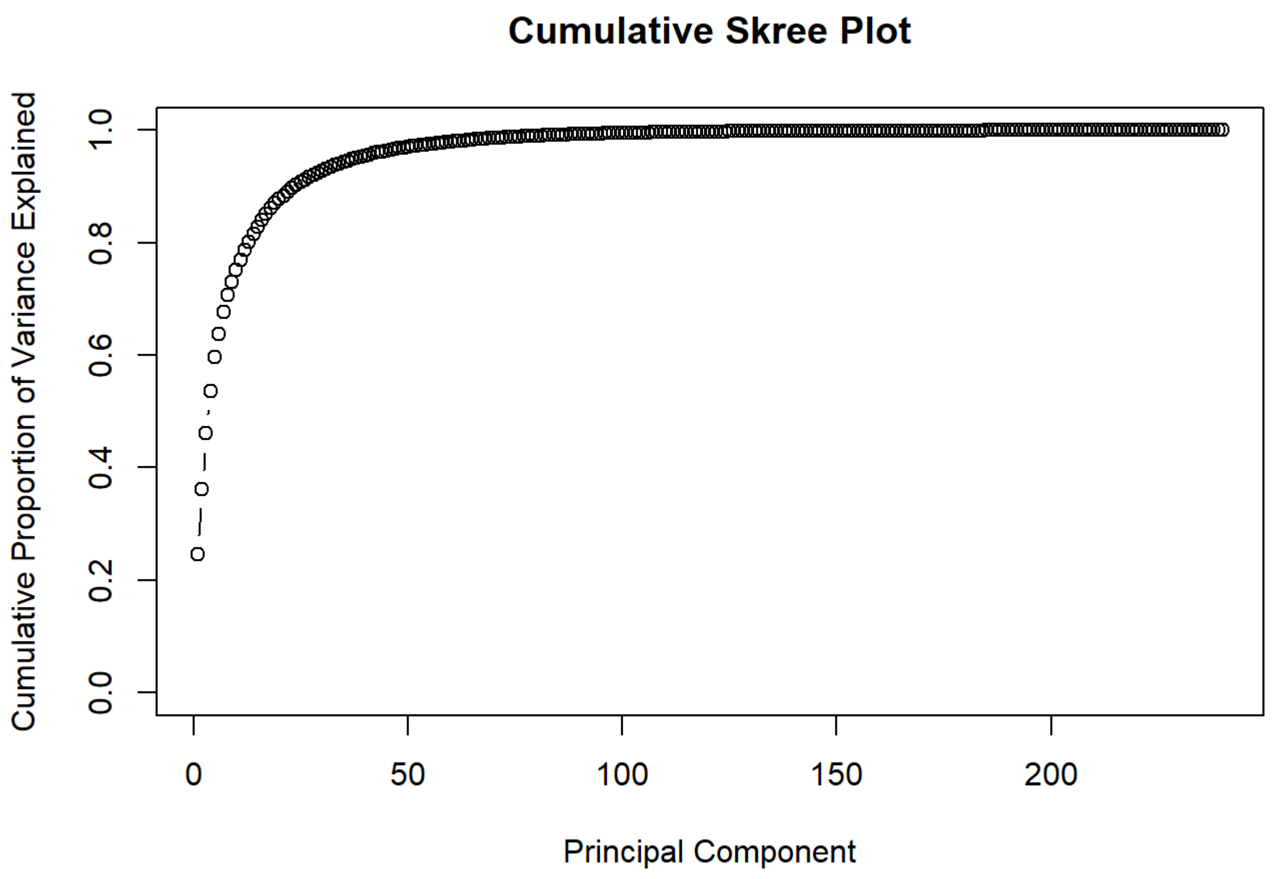
```
## The following object is masked from 'package:graphics':
##
##     frame
```

```
## The following object is masked from 'package:base':
##
##     save.image
```

```
#Load image
courtyard=load.image("C:/Users/lydia/Desktop/Lydia/IE 5561 - Data Driven Decision Making/Assignm
ents/Assignment10/courtyard.jpg")

#Grayscale image
gray_courtyard=grayscale(courtyard)

#PCA
pc_gray=prcomp(gray_courtyard, scale=TRUE)

#Calculate variance
pc_var=pc_gray$sdev^2

#Percent variance explained by each PC
pve=pc_var/sum(pc_var)

#Skree plot
plot(pve, xlab="Principal Component", ylab="Proportion of Variance Explained", ylim=c(0,1),type=
'b',main="Skree Plot")
```



```
#Cumulative skree plot
plot(cumsum(pve), xlab="Principal Component", ylab="Cumulative Proportion of Variance Explained"
, ylim=c(0,1),type='b',main="Cumulative Skree Plot")
```



```
#Choose number of PCs to explain at least 99% of variance
which(cumsum(pve)>=0.99)
```

```
##   [1]  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95
##  [18]  96  97  98  99 100 101 102 103 104 105 106 107 108 109 110 111 112
##  [35] 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129
##  [52] 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146
##  [69] 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163
##  [86] 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
## [103] 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197
## [120] 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214
## [137] 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231
## [154] 232 233 234 235 236 237 238 239 240
```

```
#Need 79 PCs
sum(pve[1:79])
```

```
## [1] 0.9900272
```

```
#Reconstruct data using only 79 components
pc_gray_recon = pc_gray$x[,1:79] %*% t(pc_gray$rotation[,1:79])

#Add the center (and re-scale) back to data
if(pc_gray$scale != FALSE){
  pc_gray_recon = scale(pc_gray_recon, center = FALSE , scale=1/pc_gray$scale)
}
```

```
## Warning in if (pc_gray$scale != FALSE) {: the condition has length > 1 and
## only the first element will be used
```

```
if(pc_gray$center != FALSE){
  pc_gray_recon = scale(pc_gray_recon, center = -1 * pc_gray$center, scale=FALSE)
}
```

```
## Warning in if (pc_gray$center != FALSE) {: the condition has length > 1 and
## only the first element will be used
```

```
#Half as many PCs
#Reconstructing data using only 40 components
pc_gray_recon_half = pc_gray$x[,1:40] %*% t(pc_gray$rotation[,1:40])

#Add the center (and re-scale)
if(pc_gray$scale != FALSE){
  pc_gray_recon_half = scale(pc_gray_recon_half, center = FALSE , scale=1/pc_gray$scale)
}
```

```
## Warning in if (pc_gray$scale != FALSE) {: the condition has length > 1 and
## only the first element will be used
```

```
if(pc_gray$center != FALSE){
  pc_gray_recon_half = scale(pc_gray_recon_half, center = -1 * pc_gray$center, scale=FALSE)
}
```

```
## Warning in if (pc_gray$center != FALSE) {: the condition has length > 1 and
## only the first element will be used
```

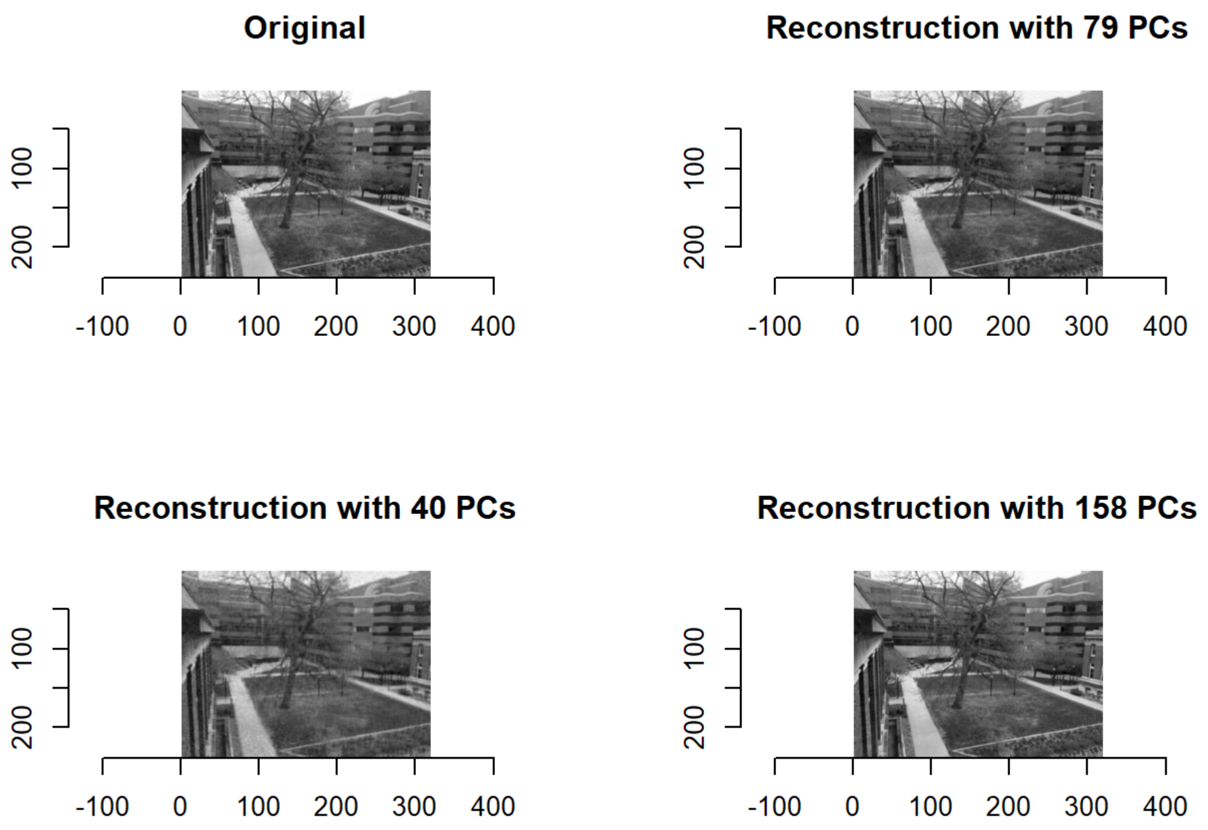
```
#Twice as many PCs
#Reconstruct data using 158 components
pc_gray_recon_twice = pc_gray$x[,1:158] %*% t(pc_gray$rotation[,1:158])
#Add the center (and re-scale)
if(pc_gray$scale != FALSE){
  pc_gray_recon_twice = scale(pc_gray_recon_twice, center = FALSE , scale=1/pc_gray$scale)
}
```

```
## Warning in if (pc_gray$scale != FALSE) {: the condition has length > 1 and
## only the first element will be used
```

```
if(pc_gray$center != FALSE){
  pc_gray_recon_twice = scale(pc_gray_recon_twice, center = -1 * pc_gray$center, scale=FALSE)
}
```

```
## Warning in if (pc_gray$center != FALSE) {: the condition has length > 1 and
## only the first element will be used
```

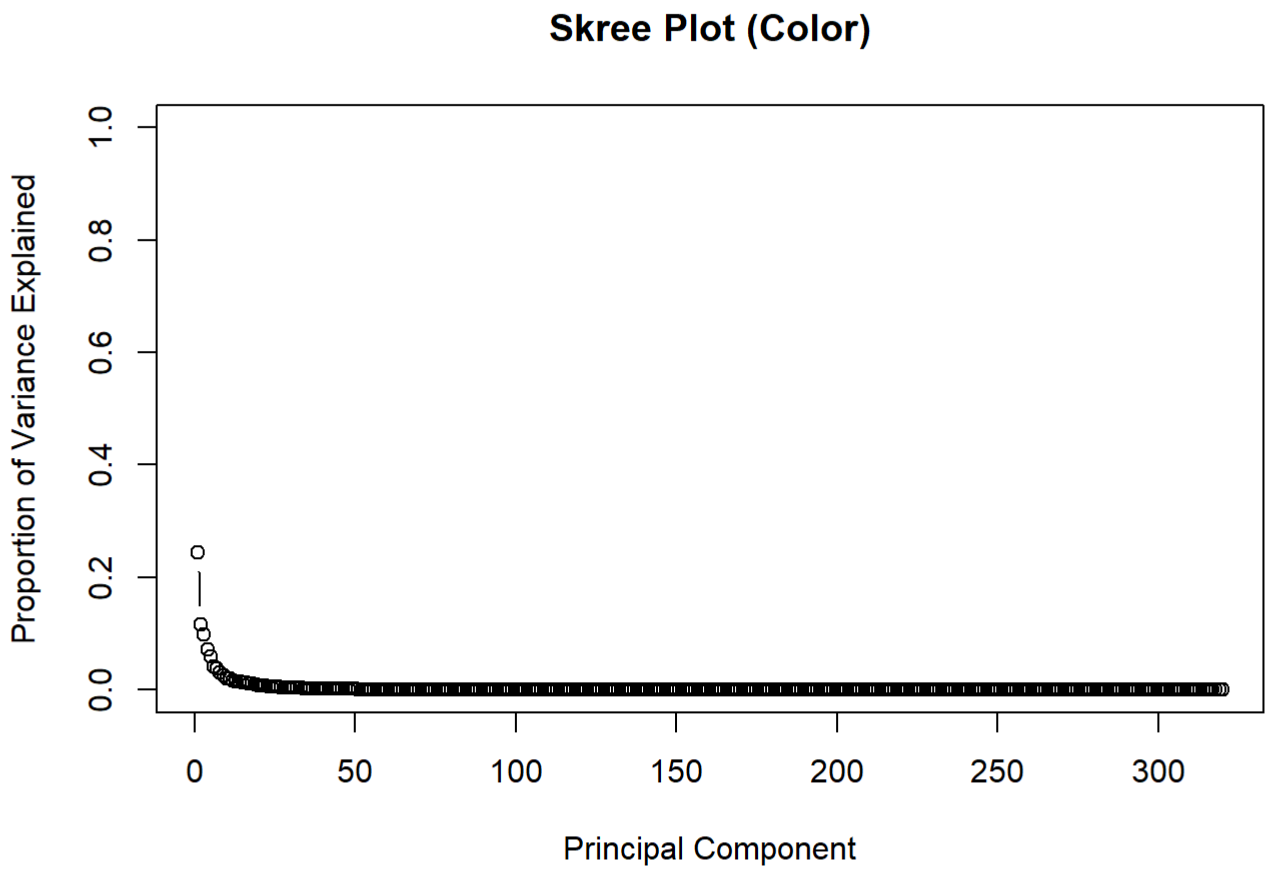
```
#Plot images
par(mfrow=c(2,2))
plot(gray_courtyard,main="Original")
plot(as.cimg(pc_gray_recon),main="Reconstruction with 79 PCs")
plot(as.cimg(pc_gray_recon_half),main="Reconstruction with 40 PCs")
plot(as.cimg(pc_gray_recon_twice),main="Reconstruction with 158 PCs")
```


Original
Reconstruction with 79 PCs
Reconstruction with 40 PCs
Reconstruction with 158 PCs
100
200
-100
0
100
200
300
400

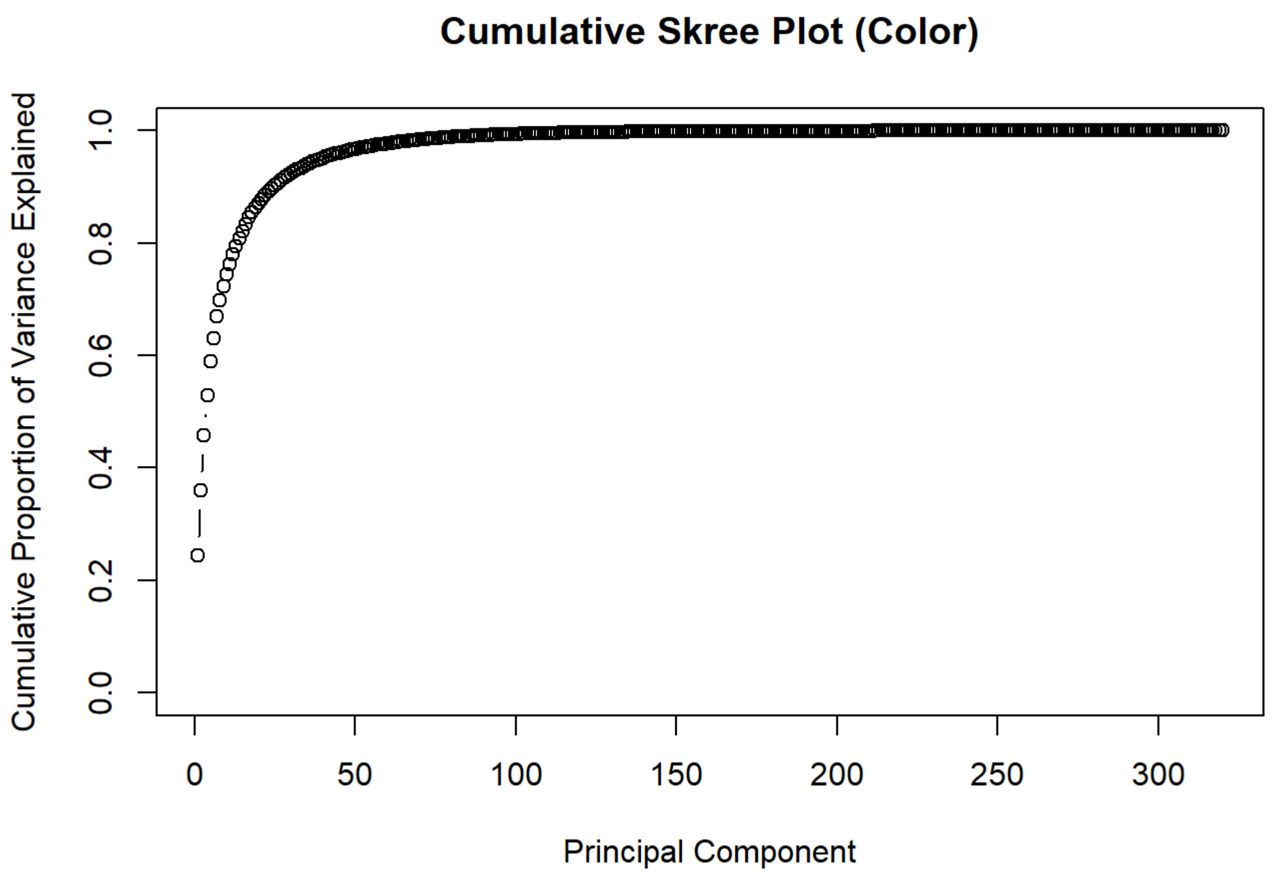
I personally cannot tell the difference between 79 PCs, 158 PCs, and the original picture. 40 PCs is definitely fuzzy (but not terrible).

Reconstructing a color image using different number of principal components

```
#Convert image to data frame
courtyard_col_df=as.data.frame(courtyard)

#Restructure data frame as matrix
new_court_col_mat=matrix(nrow=320,ncol=720)
i=1
for(j in 1:720){
  i_values=seq(i,i+319,1)
  new_court_col_mat[,j]=courtyard_col_df$value[i_values]
  i=i+320
}

#PCA
pc_color=prcomp(new_court_col_mat, scale=TRUE)

#Calculate variance
pc_var_col=pc_color$sdev^2

#Percent variance explained by each PC
pve_col=pc_var_col/sum(pc_var_col)

#Skree plot
plot(pve_col, xlab="Principal Component", ylab="Proportion of Variance Explained", ylim=c(0,1),t
ype='b',main="Skree Plot (Color)")
```



```
#Cumulative skree plot
plot(cumsum(pve_col), xlab="Principal Component", ylab="Cumulative Proportion of Variance Explai
ned", ylim=c(0,1),type='b',main="Cumulative Skree Plot (Color)")
```



```
#Choose number of PCs to explain at least 99% of variance
which(cumsum(pve_col)>=0.99)
```

```
##   [1]  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##  [18] 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117
##  [35] 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134
##  [52] 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151
##  [69] 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168
##  [86] 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185
## [103] 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202
## [120] 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219
## [137] 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236
## [154] 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253
## [171] 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270
## [188] 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287
## [205] 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304
## [222] 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320
```

```
#Need 84 PCs
sum(pve_col[1:84])
```

```
## [1] 0.9902368
```

```
#Reconstruct data using only 84 components
pc_col_recon = pc_color$x[,1:84] %*% t(pc_color$rotation[,1:84])

#Add the center (and re-scale) back to data
if(pc_color$scale != FALSE){
  pc_col_recon = scale(pc_col_recon, center = FALSE , scale=1/pc_color$scale)
}
```

```
## Warning in if (pc_color$scale != FALSE) {: the condition has length > 1 and
## only the first element will be used
```

```
if(pc_color$center != FALSE){
  pc_col_recon = scale(pc_col_recon, center = -1 * pc_color$center, scale=FALSE)
}
```

```
## Warning in if (pc_color$center != FALSE) {: the condition has length > 1
## and only the first element will be used
```

```
#Put data into an array (Flatten out reconstructed data)
pc_col_recon_array=array(data=pc_col_recon,dim=230400)

#Replace values column in data frame
courtyard_col_df$value <- pc_col_recon_array

#Convert to cimg
pc_col_recon_image=as.cimg(courtyard_col_df)
```

```
## Warning in as.cimg.data.frame(courtyard_col_df): Guessing image dimensions
## from maximum coordinate values
```

```
#Half as many PCs
#Reconstruct data using only 42 components
pc_col_recon_half = pc_color$x[,1:42] %*% t(pc_color$rotation[,1:42])

#Add the center (and re-scale)
if(pc_color$scale != FALSE){
  pc_col_recon_half = scale(pc_col_recon_half, center = FALSE , scale=1/pc_color$scale)
}
```

```
## Warning in if (pc_color$scale != FALSE) {: the condition has length > 1 and
## only the first element will be used
```

```
if(pc_color$center != FALSE){
  pc_col_recon_half = scale(pc_col_recon_half, center = -1 * pc_color$center, scale=FALSE)
}
```

```
## Warning in if (pc_color$center != FALSE) {: the condition has length > 1
## and only the first element will be used
```

```
#Put data into an array (flatten out reconstructed data)
pc_col_recon_array_half=array(data=pc_col_recon_half,dim=230400)

#Replace values column in data frame
courtyard_col_df$value <- pc_col_recon_array_half

#Convert to cimg
pc_col_recon_img_half=as.cimg(courtyard_col_df)
```

```
## Warning in as.cimg.data.frame(courtyard_col_df): Guessing image dimensions
## from maximum coordinate values
```

```
#Twice as many PCs
#Reconstruct data using 168 components
pc_col_recon_twice = pc_color$x[,1:168] %*% t(pc_color$rotation[,1:168])

#Add the center (and re-scale)
if(pc_color$scale != FALSE){
  pc_col_recon_twice = scale(pc_col_recon_twice, center = FALSE , scale=1/pc_color$scale)
}
```

```
## Warning in if (pc_color$scale != FALSE) {: the condition has length > 1 and
## only the first element will be used
```

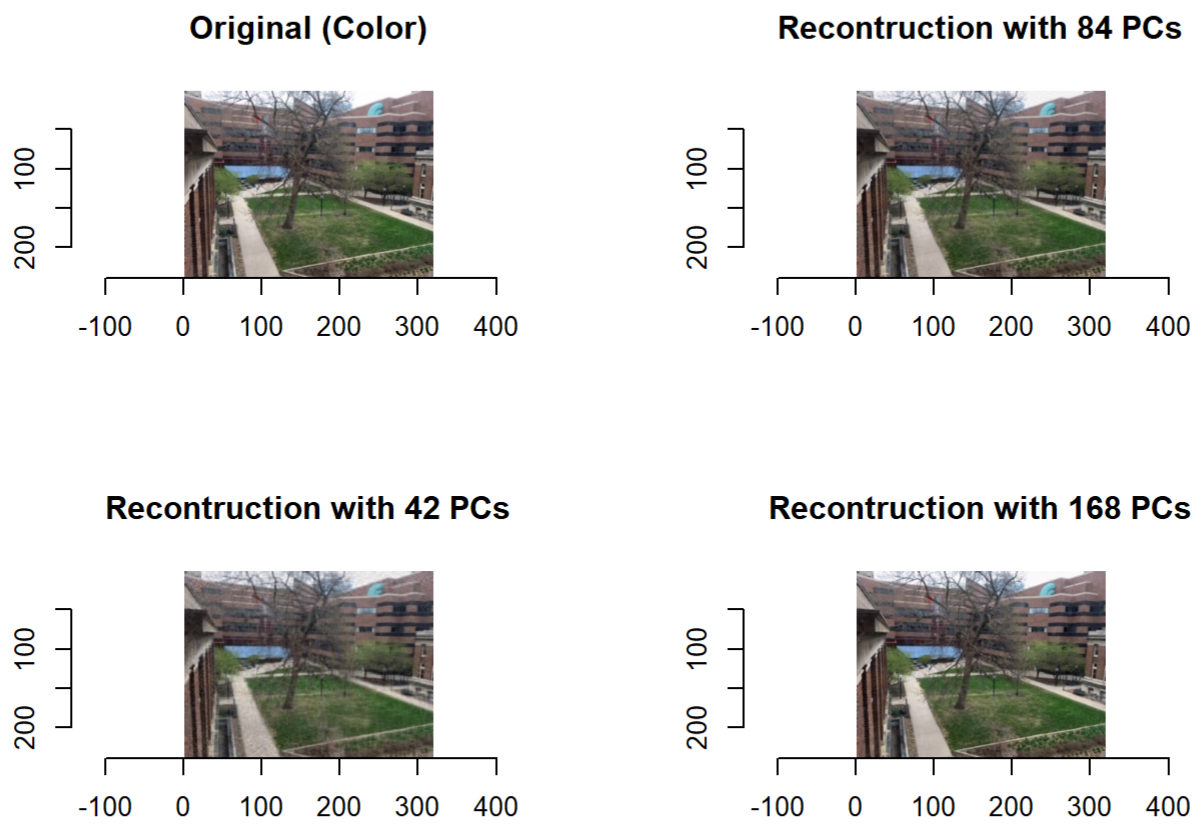
```
if(pc_color$center != FALSE){
  pc_col_recon_twice = scale(pc_col_recon_twice, center = -1 * pc_color$center, scale=FALSE)
}
```

```
## Warning in if (pc_color$center != FALSE) {: the condition has length > 1
## and only the first element will be used
```

```
#Put data into an array (flatten out reconstructed data)
pc_col_recon_array_twice=array(data=pc_col_recon_twice,dim=230400)

#Replace values column in data frame
courtyard_col_df$value <- pc_col_recon_array_twice

#Convert to cimg
pc_col_recon_img_twice=as.cimg(courtyard_col_df)
```

```
## Warning in as.cimg.data.frame(courtyard_col_df): Guessing image dimensions
## from maximum coordinate values
```

```
#Plot images
par(mfrow=c(2,2))
plot(courtyard,main="Original (Color)")
plot(pc_col_recon_image,main="Recontruction with 84 PCs")
plot(pc_col_recon_img_half,main="Recontruction with 42 PCs")
plot(pc_col_recon_img_twice,main="Recontruction with 168 PCs")
```


Original (Color)
Recontruction with 84 PCs
Recontruction with 42 PCs
Recontruction with 168 PCs
100
200
-100
0
100
200
300
400

Again, I see very little difference between my reconstruction with 84 PCs, 168 PCs, and the original. The PCA reconstructions are, perhaps, a little darker. However, the reconstruction with 42 PCs is noticably darker and fuzzier. I would say it worked about as well as the grayscale version although I needed to use a few more principal components to get to 99% variation explained.