grouping-plotting-data.R

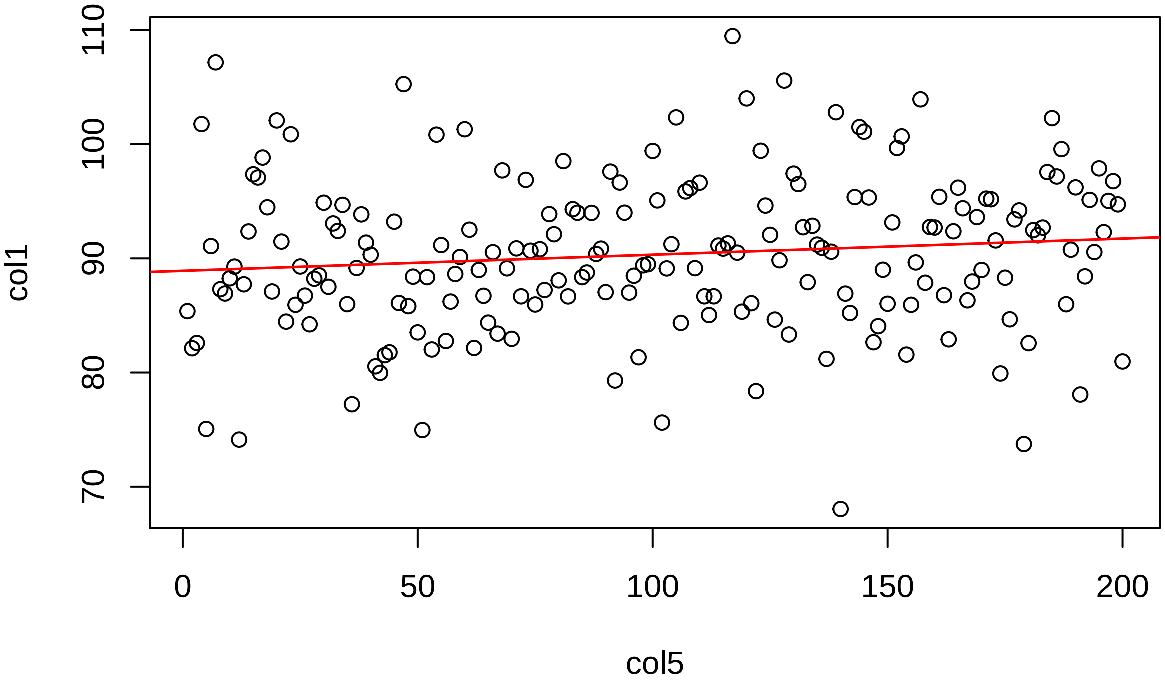
julian

Tue Jun 2 16:04:11 2015

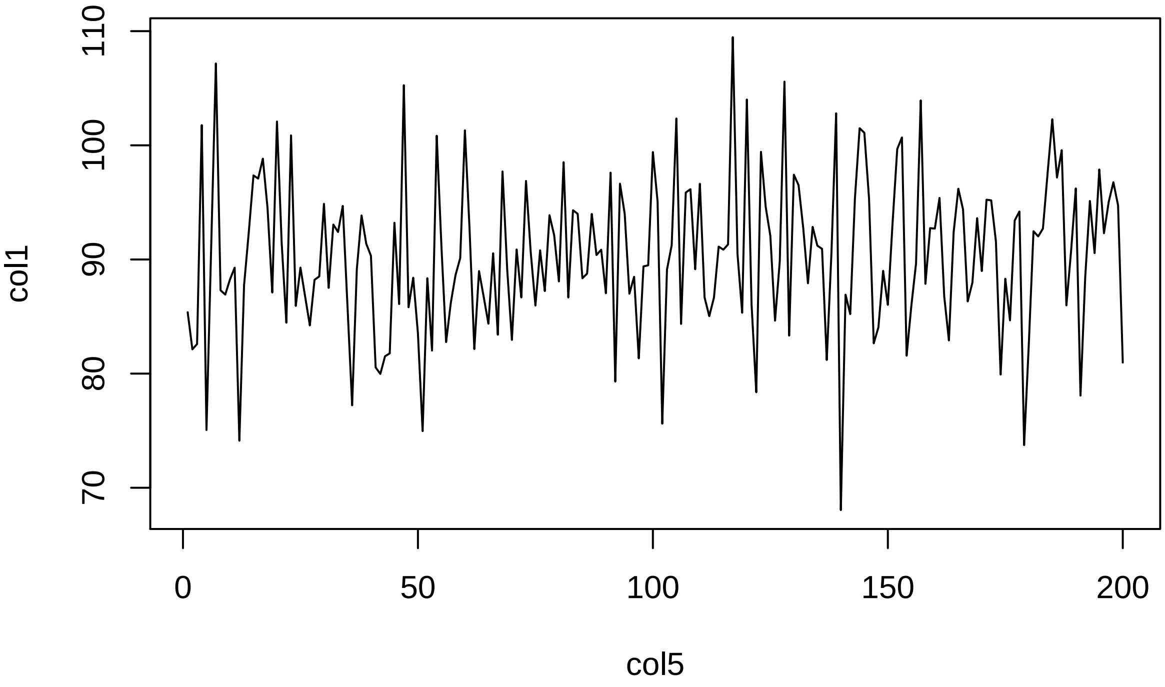
```r
#!/usr/bin/env Rscript

# describes how to make plots in base R
# introduction to ggplot


target.dir <- '~/GitHub/reproducible-research/Day-3/datasets'
target.file <- 'grouping-plotting-data-r-examples.txt'
sink(file = file.path(target.dir, target.file))


# grouping data ---------------------------------------------------------

# will use both plyr and dplyr
# when importing both, load plyr, then dplyr

library(plyr)
library(dplyr)

RNGkind('Mersenne-Twister')
set.seed(86519883)
old.seed <- .Random.seed

col1 = rlnorm(n = 200, meanlog = 4.5, sdlog = 0.08)
col2 = rnorm(n = 200, mean = 2.3, sd = 0.57)
col3 = runif(n = 200, min = 0, max = 2000)
col4 = rgeom(n = 200, prob = 0.34)
col5 = seq(1, 200, 1)

plot.df <- data.frame(col1, col2, col3, col4, col5)

head(plot.df)
```

```
##        col1     col2      col3 col4 col5
## 1  85.37802 1.679782 1864.0727    1    1
## 2  82.12614 1.959824  309.1779    2    2
## 3  82.58994 2.799975 1509.0907    3    3
## 4 101.76460 2.999126 1451.9434    3    4
## 5  75.05743 2.307869  156.7992    0    5
## 6  91.07610 3.439199 1208.1128    0    6
```

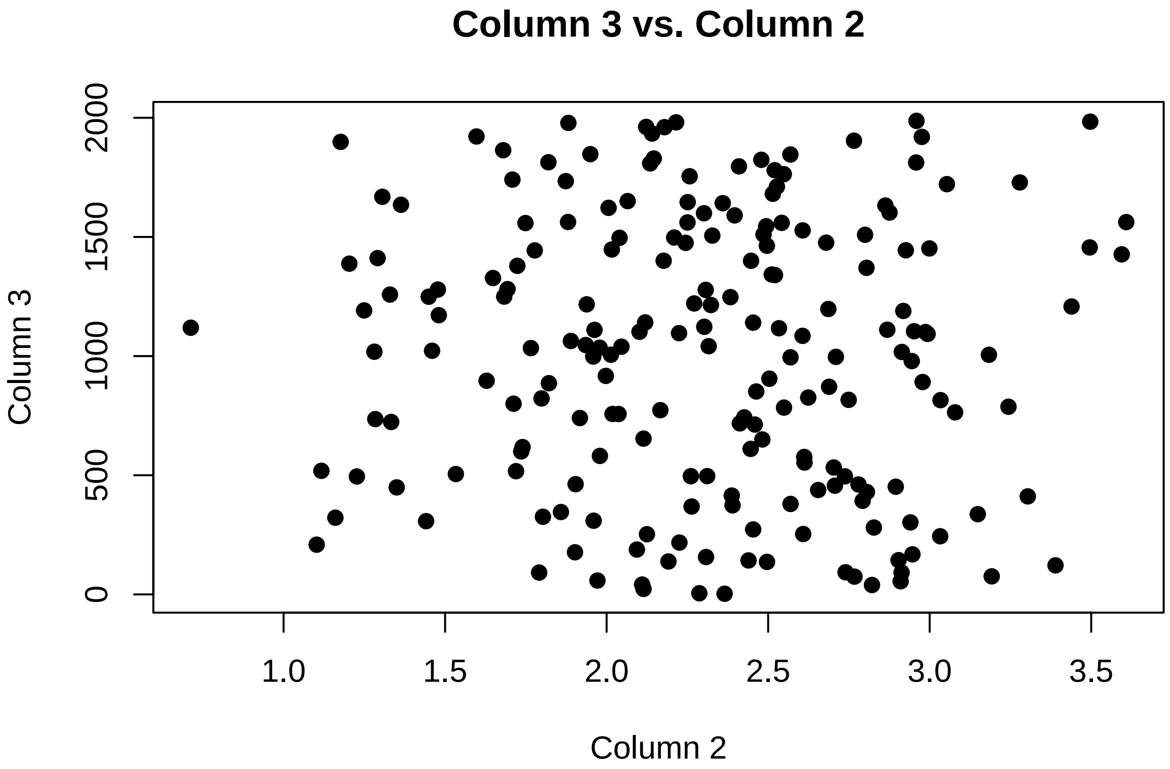
```r
# formula notation for a plot;
# uses form y ~ x
# scatter plot by default
plot(col1 ~ col5, data = plot.df)
abline(lm(col1 ~ col5, data = plot.df), col = 'red', lwd = 1.3)
```



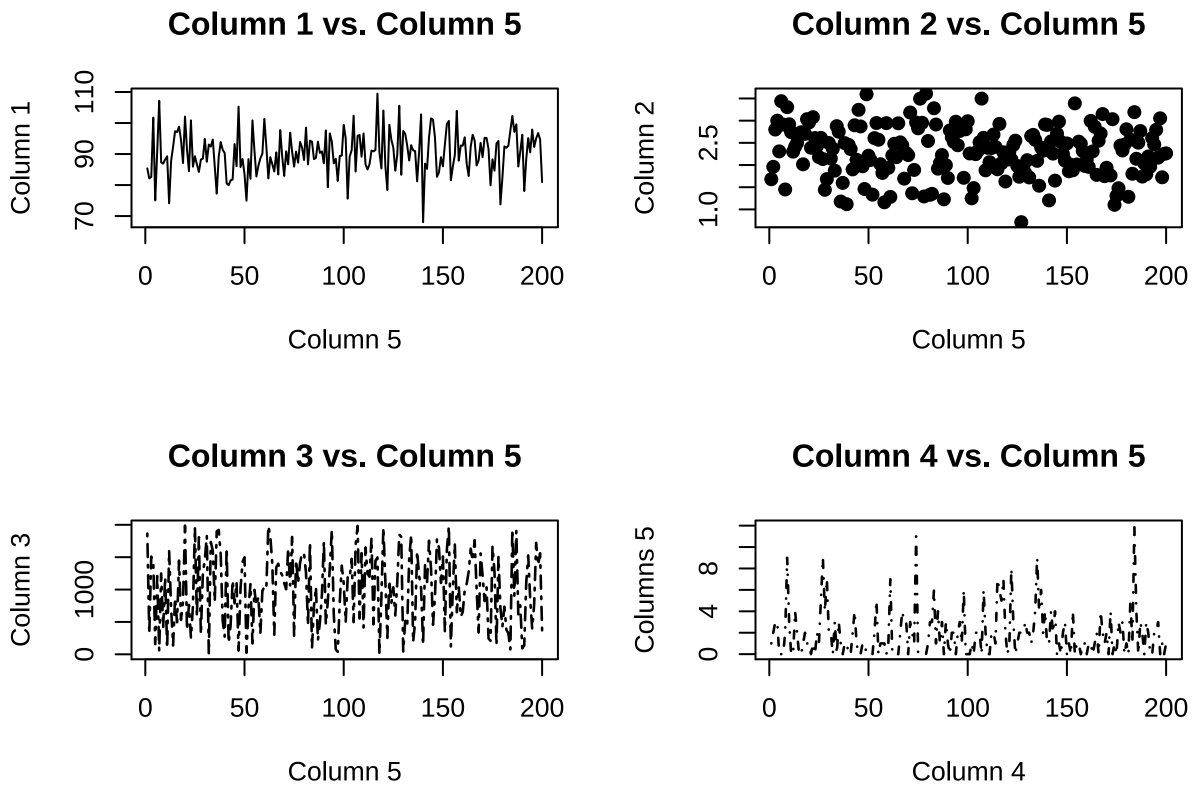
```
plot(col1 ~ col5, data = plot.df, type = 'l')
```



```
# scatter plot without the default plot shape and size
plot(col3 ~ col2, data = plot.df, pch = 19,
     xlab = 'Column 2', ylab = 'Column 3',
     main = 'Column 3 vs. Column 2')
```



```r
# adding subplots and different types
par(mfrow = c(2, 2))

plot(col1 ~ col5, data = plot.df, type = 'l',
     ylab = 'Column 1', xlab = 'Column 5',
     main = 'Column 1 vs. Column 5')

plot(col2 ~ col5, data = plot.df, pch = 19,
     xlab = 'Column 5', ylab = 'Column 2',
     main = 'Column 2 vs. Column 5')

plot(col3 ~ col5, data = plot.df,
     type = 'l', lty = 6, lwd = 1.3,
     ylab = 'Column 3', xlab = 'Column 5',
     main = 'Column 3 vs. Column 5')

plot(col4 ~ col5, data = plot.df,
     type = 'l', lty = 4, lwd = 1.3,
     ylab = 'Columns 5', xlab = 'Column 4',
     main = 'Column 4 vs. Column 5')
```



```r
# histograms and kernel density estimates --------------------------------

# create categorical variables

random.nums <- runif(n = nrow(plot.df), min = 0, max = 1)

high <- random.nums > 0.85
middling <- random.nums <= 0.85 & random.nums > 0.5
just.ugh <- random.nums <= 0.5

# creates null vector
plot.df$Category <- NA

plot.df[high, 'Category'] <- 'high'
plot.df[middling, 'Category'] <- 'middling'
plot.df[just.ugh, 'Category'] <- 'terrible'

head(plot.df)
```

```
##        col1     col2      col3 col4 col5 Category
## 1  85.37802 1.679782 1864.0727    1    1 terrible
## 2  82.12614 1.959824  309.1779    2    2 terrible
## 3  82.58994 2.799975 1509.0907    3    3 terrible
## 4 101.76460 2.999126 1451.9434    3    4 terrible
## 5  75.05743 2.307869  156.7992    0    5 terrible
## 6  91.07610 3.439199 1208.1128    0    6 middling
```

```r
summary(plot.df)
```

```
##       col1             col2             col3               col4
##  Min.   : 68.05   Min.   :0.7127   Min.   :   2.705   Min.   : 0.000
##  1st Qu.: 86.09   1st Qu.:1.9308   1st Qu.: 495.794   1st Qu.: 0.000
##  Median : 90.44   Median :2.3137   Median :1040.492   Median : 1.000
##  Mean   : 90.33   Mean   :2.3004   Mean   :1007.463   Mean   : 1.875
##  3rd Qu.: 94.70   3rd Qu.:2.7075   3rd Qu.:1496.675   3rd Qu.: 3.000
##  Max.   :109.47   Max.   :3.6085   Max.   :1987.244   Max.   :12.000
##       col5          Category
##  Min.   :  1.00   Length:200
##  1st Qu.: 50.75   Class :character
##  Median :100.50   Mode  :character
##  Mean   :100.50
##  3rd Qu.:150.25
##  Max.   :200.00
```

```r
str(plot.df)
```

```
## 'data.frame':    200 obs. of  6 variables:
##  $ col1    : num  85.4 82.1 82.6 101.8 75.1 ...
##  $ col2    : num  1.68 1.96 2.8 3 2.31 ...
##  $ col3    : num  1864 309 1509 1452 157 ...
##  $ col4    : int  1 2 3 3 0 0 0 2 9 3 ...
##  $ col5    : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ Category: chr  "terrible" "terrible" "terrible" "terrible" ...
```

```r
glimpse(plot.df)
```

```
## Observations: 200
## Variables:
## $ col1     (dbl) 85.37802, 82.12614, 82.58994, 101.76460, 75.05743, 91...
## $ col2     (dbl) 1.679782, 1.959824, 2.799975, 2.999126, 2.307869, 3.4...
## $ col3     (dbl) 1864.07274, 309.17789, 1509.09074, 1451.94336, 156.79...
## $ col4     (int) 1, 2, 3, 3, 0, 0, 0, 2, 9, 3, 0, 0, 4, 2, 0, 0, 1, 2,...
## $ col5     (dbl) 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16...
## $ Category (chr) "terrible", "terrible", "terrible", "terrible", "terr...
```

```r
# here is a good chance to introduce various additional summaries

min(plot.df$col1)
```

```
## [1] 68.04582
```

```r
max(plot.df$col2)
```

```
## [1] 3.608546
```

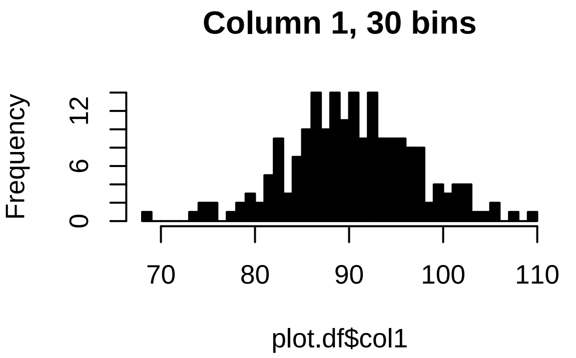
```r
range(plot.df$col4)
```

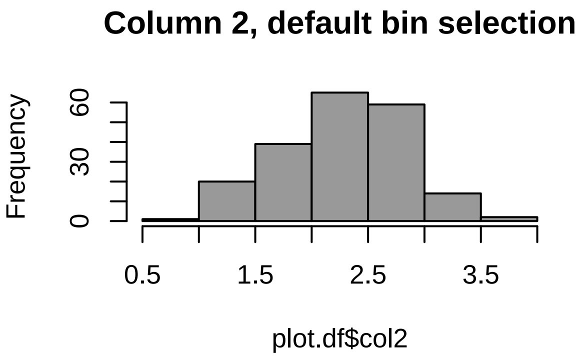
```
## [1]  0 12
```

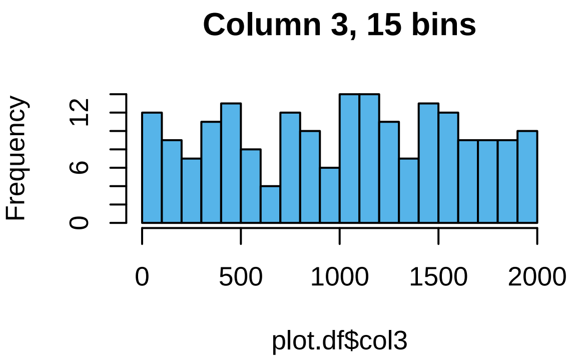
```r
fivenum(plot.df$col5)
```

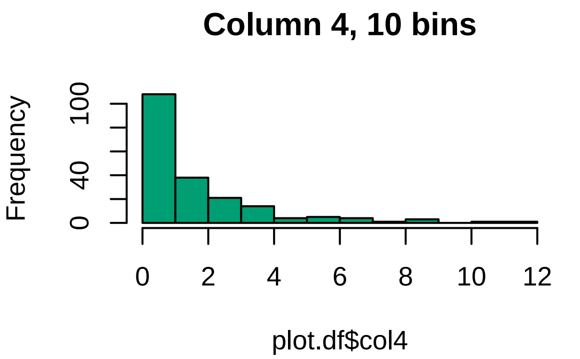
```
## [1]   1.0  50.5 100.5 150.5 200.0
```

```r
# basic histograms

hist(plot.df$col1, col = '#000000', main = 'Column 1, 30 bins',
     breaks = 30)

hist(plot.df$col2, col = '#999999', main = 'Column 2, default bin selection')

hist(plot.df$col3, col = '#56B4E9', main = 'Column 3, 15 bins',
     breaks = 15)

hist(plot.df$col4, col = '#009E73', main = 'Column 4, 10 bins',
     breaks = 10)
```

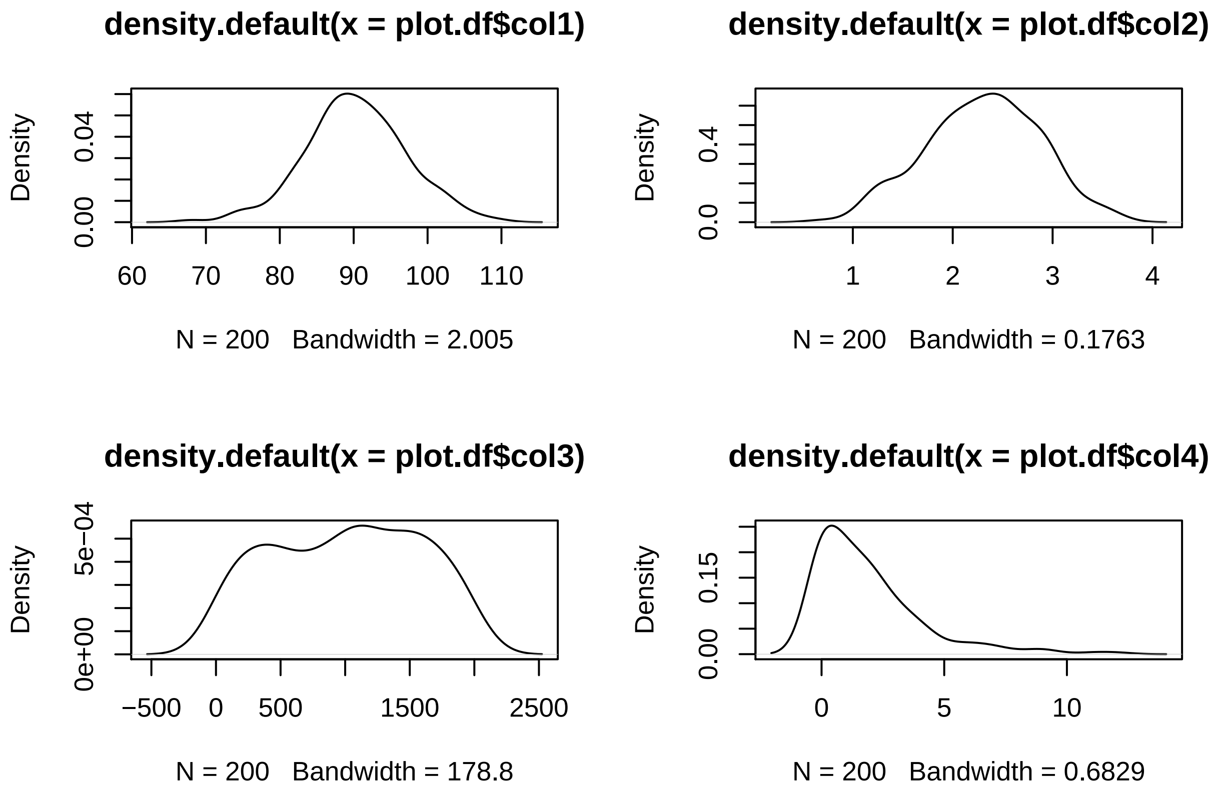








```r
# basic kernel density estimates

plot(density(plot.df$col1))
plot(density(plot.df$col2))
plot(density(plot.df$col3))
plot(density(plot.df$col4))
```



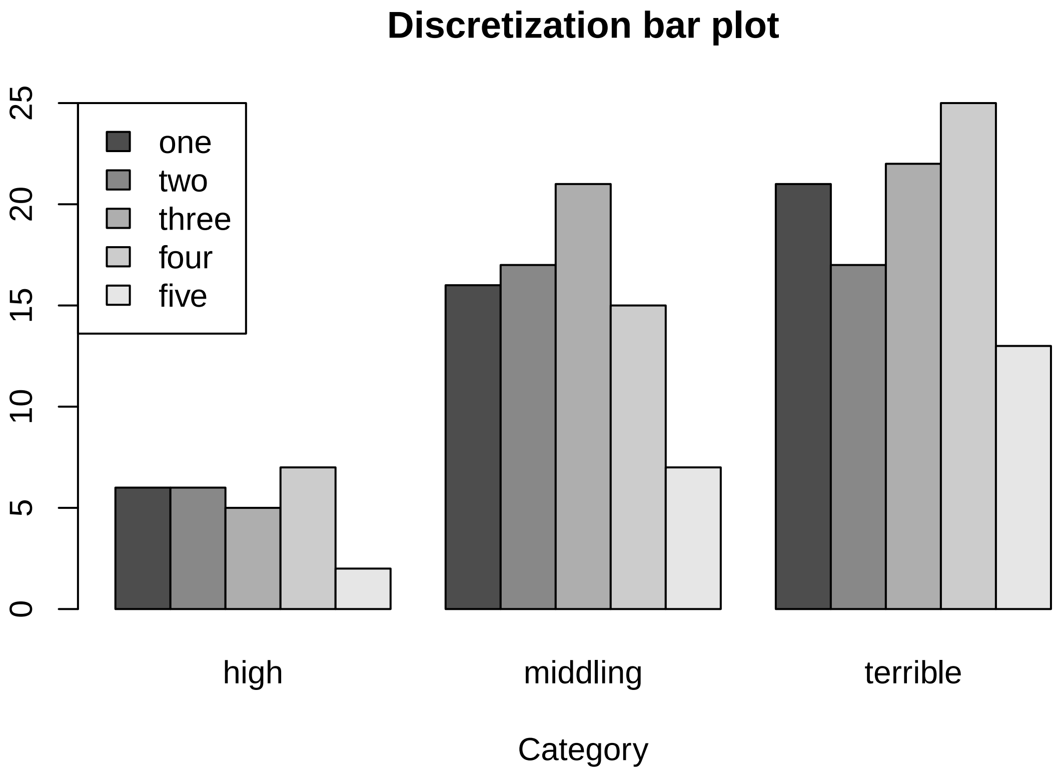
```
# binning columns

col3.bins <- seq(1, 2220, 440)
col3.labels = c('one', 'two', 'three', 'four', 'five')

plot.df$col3Cut <-
  cut(plot.df$col3, breaks = col3.bins, labels = col3.labels,
      include.lowest = TRUE, right = TRUE)

head(plot.df)
```

```
##        col1     col2      col3 col4 col5 Category col3Cut
## 1  85.37802 1.679782 1864.0727    1    1 terrible    five
## 2  82.12614 1.959824  309.1779    2    2 terrible     one
## 3  82.58994 2.799975 1509.0907    3    3 terrible    four
## 4 101.76460 2.999126 1451.9434    3    4 terrible    four
## 5  75.05743 2.307869  156.7992    0    5 terrible     one
## 6  91.07610 3.439199 1208.1128    0    6 middling   three
```

```
# cross-tabulation is similar to initial Python example (get counts)
# notice that the order of columns and rows is different

plot.df.xtabs <- xtabs( ~ col3Cut + Category, data = plot.df)
plot.df.xtabs
```

```
##        Category
## col3Cut high middling terrible
##    one       6        16        21
##    two       6        17        17
##    three     5        21        22
##    four      7        15        25
##    five      2         7        13
```

```
par(mfrow = c(1,1))
barplot(plot.df.xtabs, beside = TRUE, main = 'Discretization bar plot',
        legend.text = TRUE, xlab = 'Category',
        args.legend = list(x = 'topleft'))
```



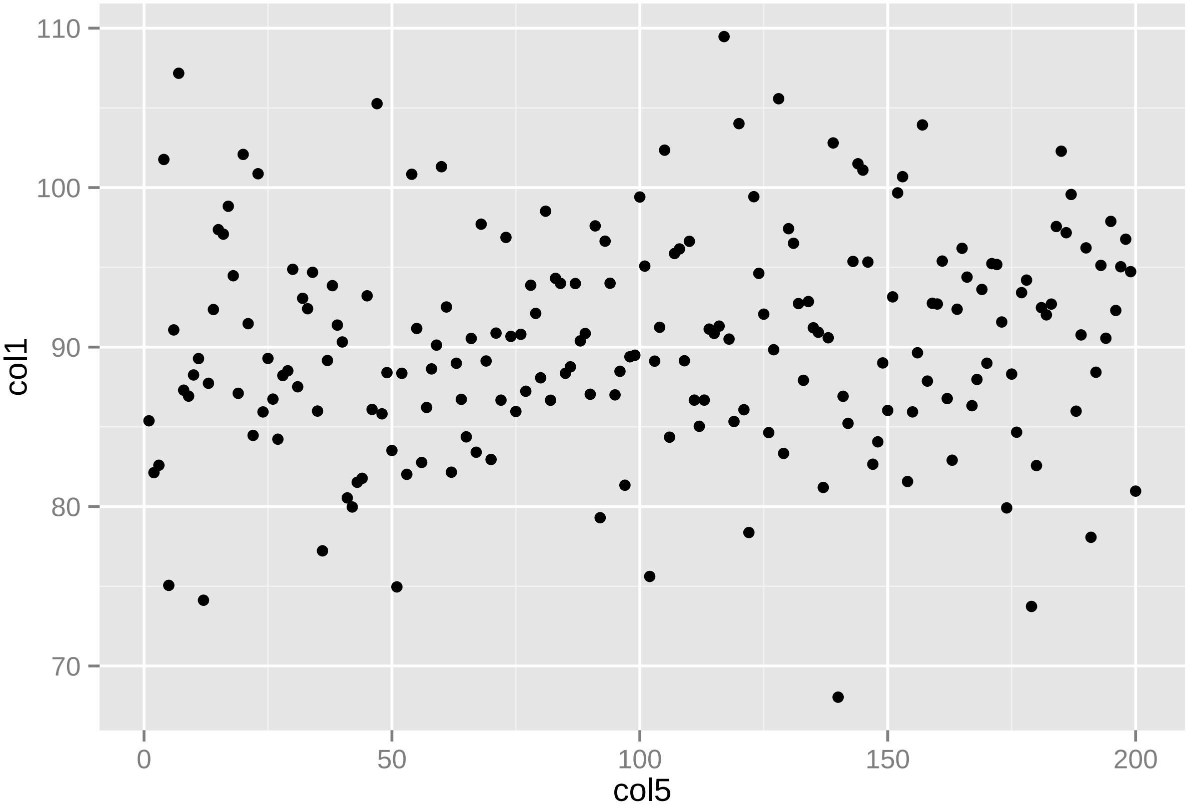
```
# grouping columns
# two ways to group: use plyr or dplyr

# grouping with plyr and dplyr

# using plyr
plot.df.plyr <-
  ddply(plot.df, .(Category), summarize,
        meanCol1 = mean(col1, na.rm = TRUE),
        meanCol2 = mean(col2, na.rm = TRUE),
        meanCol3 = mean(col3, na.rm = TRUE),
        meanCol4 = mean(col4),
        varCol1 = var(col1, na.rm = TRUE),
        varCol2 = var(col2, na.rm = TRUE),
        varCol3 = var(col3, na.rm = TRUE),
        varCol4 = var(col4))

plot.df.plyr
```

```
##   Category meanCol1 meanCol2  meanCol3 meanCol4  varCol1   varCol2
## 1     high 90.22889 2.422299 1032.3790 1.307692 27.05617 0.3274840
## 2 middling 90.01248 2.181181  971.7555 1.763158 49.99122 0.3250632
## 3 terrible 90.60285 2.360568 1028.5452 2.112245 49.29412 0.3010934
##    varCol3  varCol4
## 1 309238.6 3.741538
## 2 299709.9 3.489825
## 3 361064.0 6.018199
```

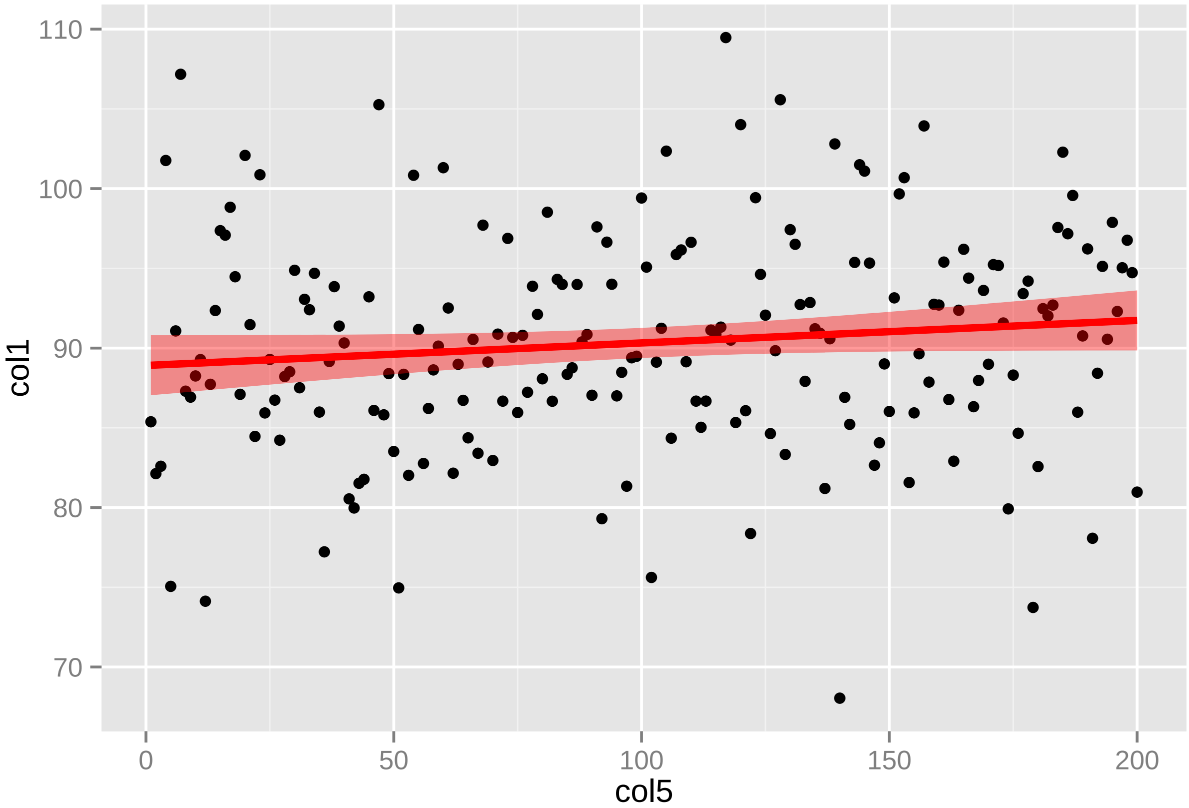
```r
# using dplyr
# notice the similarities with pandas
plot.df.dplyr <-
  group_by(plot.df, Category) %>%
  summarize(meanCol1 = mean(col1, na.rm = TRUE),
            meanCol2 = mean(col2, na.rm = TRUE),
            meanCol3 = mean(col3, na.rm = TRUE),
            meanCol4 = mean(col4),
            varCol1 = var(col1, na.rm = TRUE),
            varCol2 = var(col2, na.rm = TRUE),
            varCol3 = var(col3, na.rm = TRUE),
            varCol4 = var(col4))

plot.df.dplyr
```

```
## Source: local data frame [3 x 9]
##
##   Category meanCol1 meanCol2  meanCol3 meanCol4  varCol1   varCol2
## 1     high 90.22889 2.422299 1032.3790 1.307692 27.05617 0.3274840
## 2 middling 90.01248 2.181181  971.7555 1.763158 49.99122 0.3250632
## 3 terrible 90.60285 2.360568 1028.5452 2.112245 49.29412 0.3010934
## Variables not shown: varCol3 (dbl), varCol4 (dbl)
```

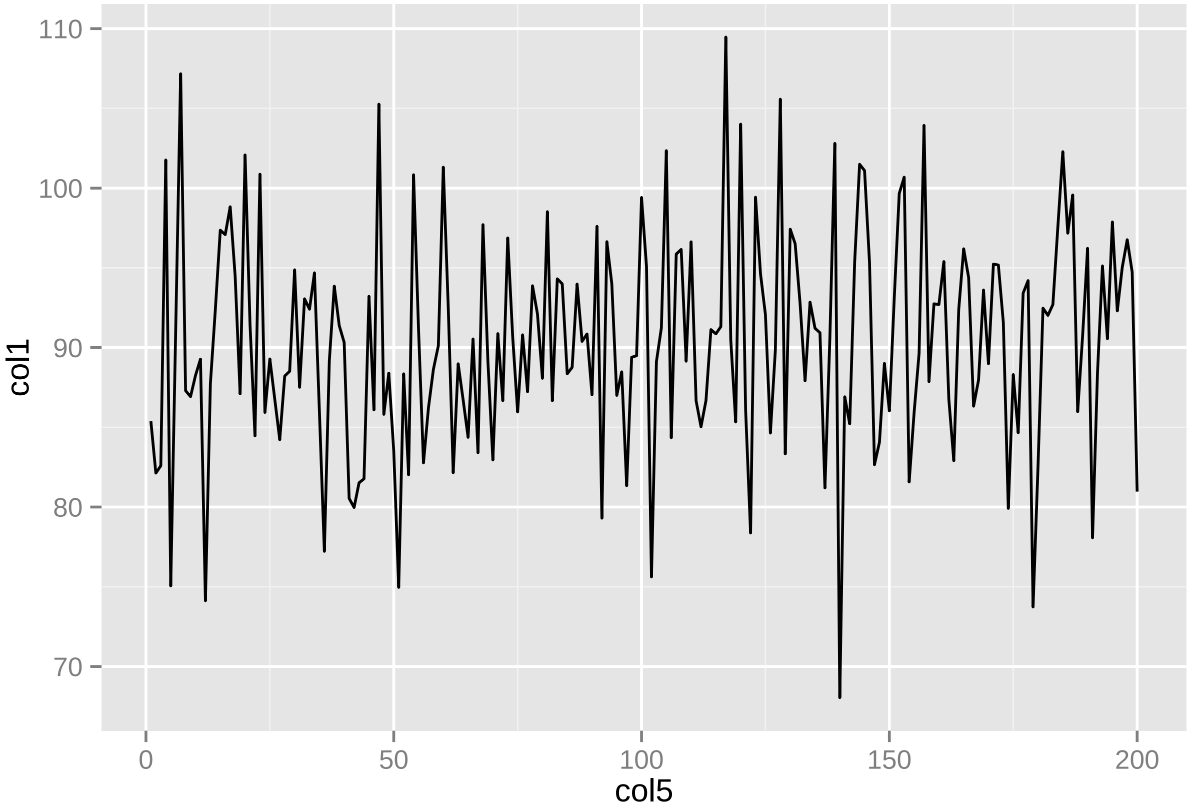
```r
# introduction to ggplot2 -------------------------------------------------

# see also library(help = 'ggplot2') and the RStudio cheatsheet
# for what can be dome with ggplot2

library(ggplot2)

ggplot(plot.df, aes(x = col5, y = col1)) +
  geom_point()
```



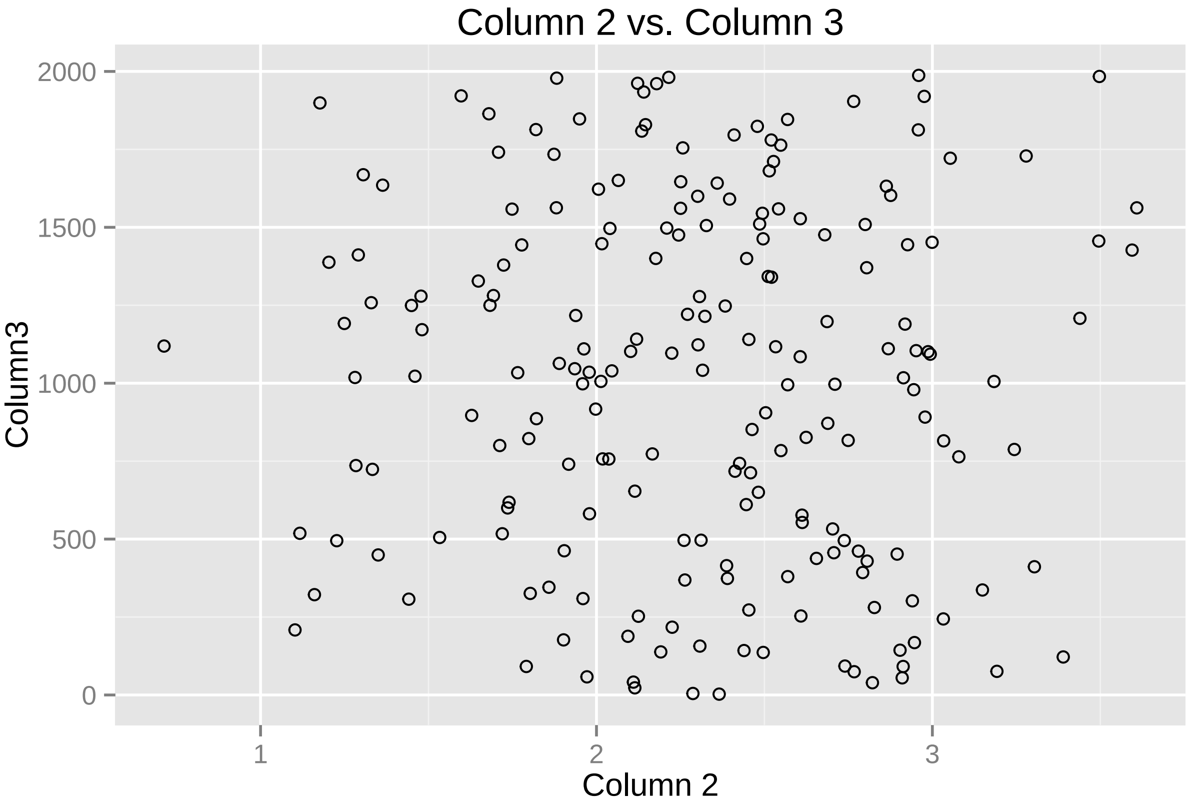
```
ggplot(plot.df, aes(x = col5, y = col1)) +
  geom_point() +
  geom_smooth(method = 'lm', size = 1.3, colour = 'red', fill = 'red')
```



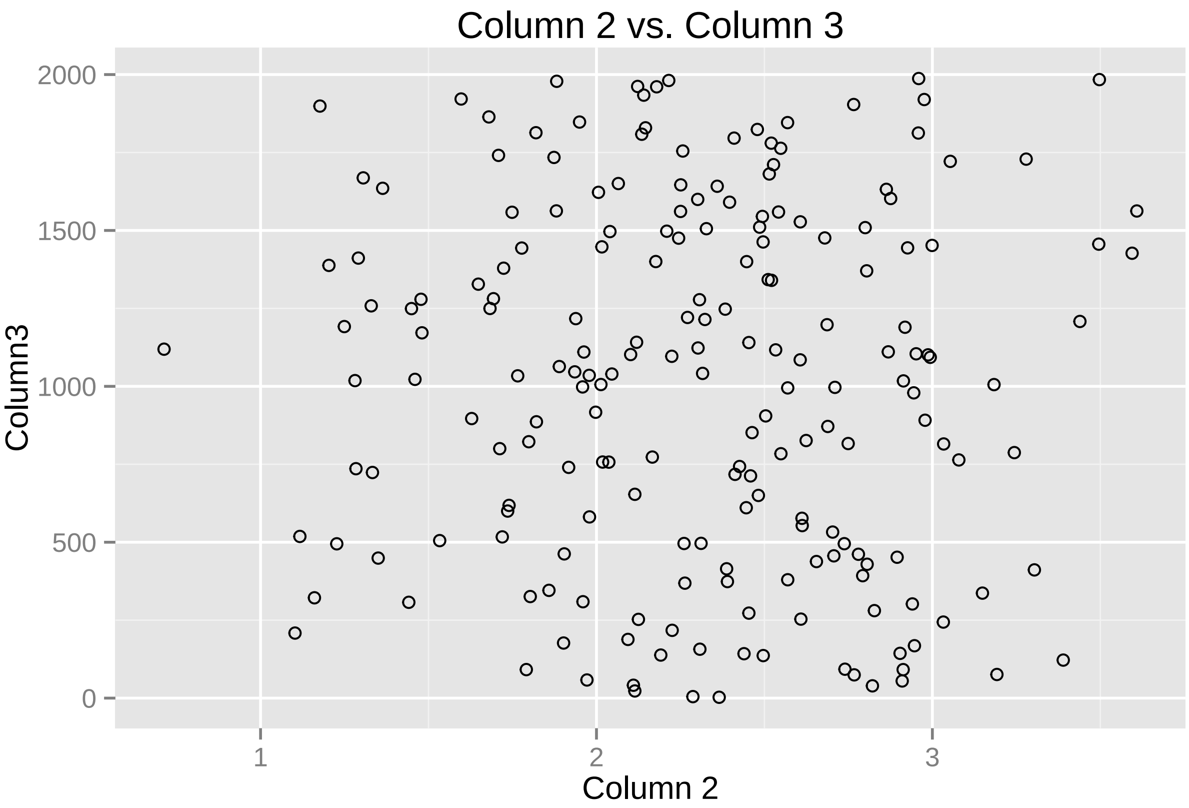
```
ggplot(plot.df, aes(x = col5, y = col1)) +
  geom_line()
```



```
# scatter plot without the default plot shape and size
# note that filled circle (pch = 19) is the default in ggplot

ggplot(plot.df, aes(x = col2, y = col3)) +
  geom_point(shape = 21) +
  xlab('Column 2') + ylab('Column3') +
  ggtitle('Column 2 vs. Column 3')
```

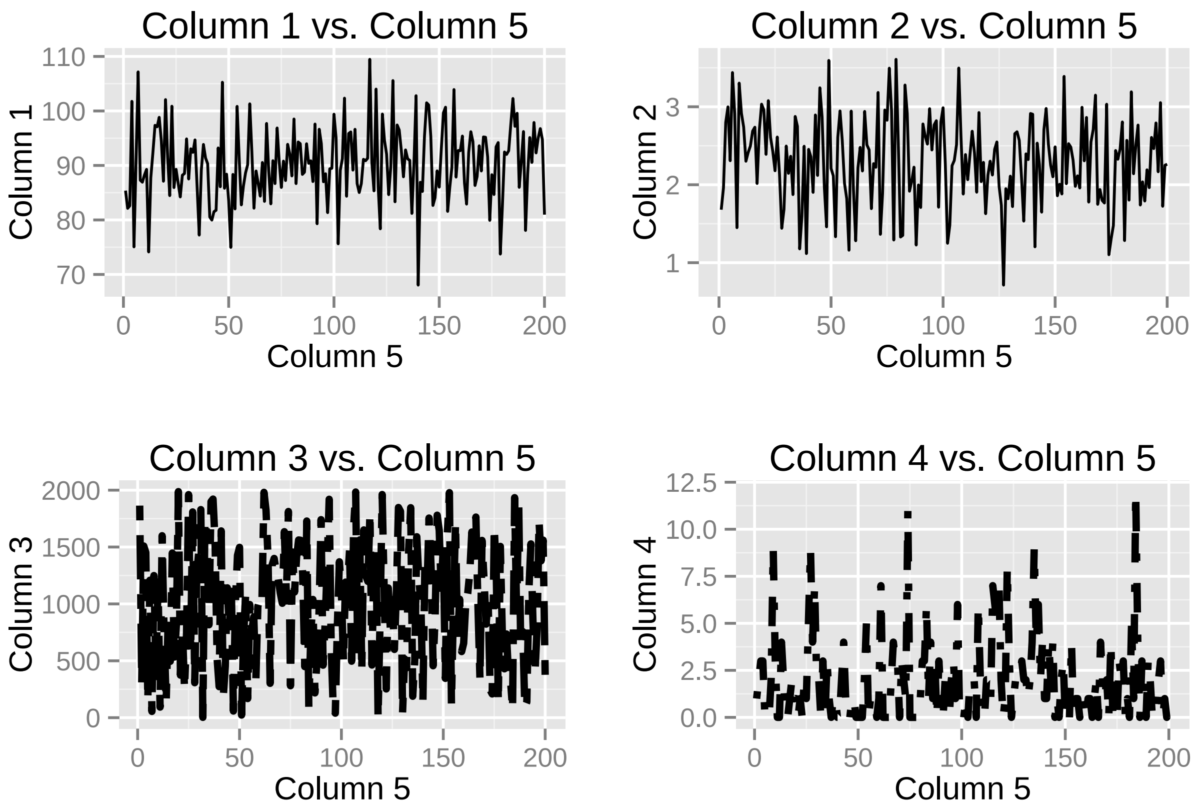


```
# could also do this
ggplot(plot.df, aes(x = col2, y = col3)) +
  geom_point(shape = 21) +
  labs(x = 'Column 2', y = 'Column3', title = 'Column 2 vs. Column 3')
```

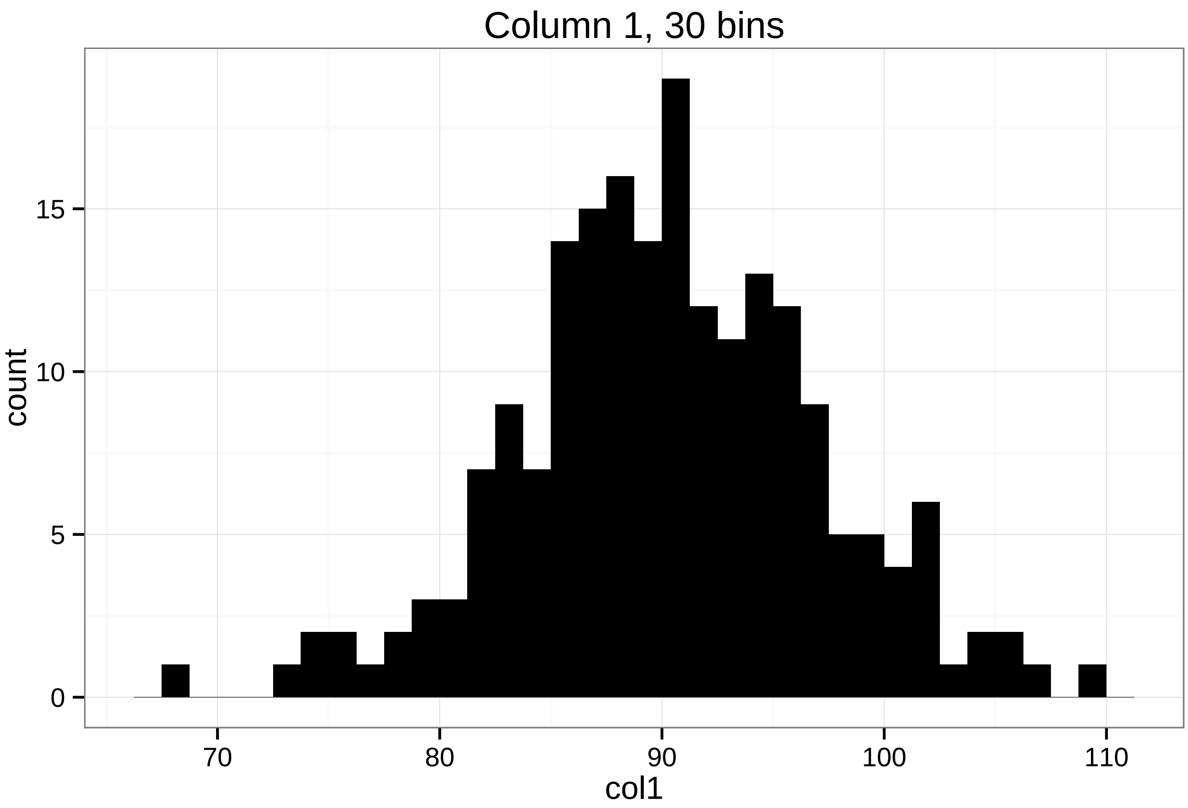


```
# adding subplots and different types
# to add subplots with ggplot2, need to use the grid package

# see also the multiplot function by Winston Chang

library(grid)

grid.newpage()
pushViewport(viewport(layout = grid.layout(2,2)))

# can crete objects with ggplot commands
# use these to print to specific part of grid

col1.vs.col5.plot <-
  ggplot(plot.df, aes(x = col5, y = col1)) +
  geom_line() +
  labs(title = 'Column 1 vs. Column 5', x = 'Column 5', y = 'Column 1')

col2.vs.col5.plot <-
  ggplot(plot.df, aes(x = col5, y = col2)) +
  geom_line() +
  labs(title = 'Column 2 vs. Column 5', x = 'Column 5', y = 'Column 2')

col3.vs.col5.plot <-
  ggplot(plot.df, aes(x = col5, y = col3)) +
  geom_line(linetype = 6, size = 1.3) +
  labs(title = 'Column 3 vs. Column 5', x = 'Column 5', y = 'Column 3')
```

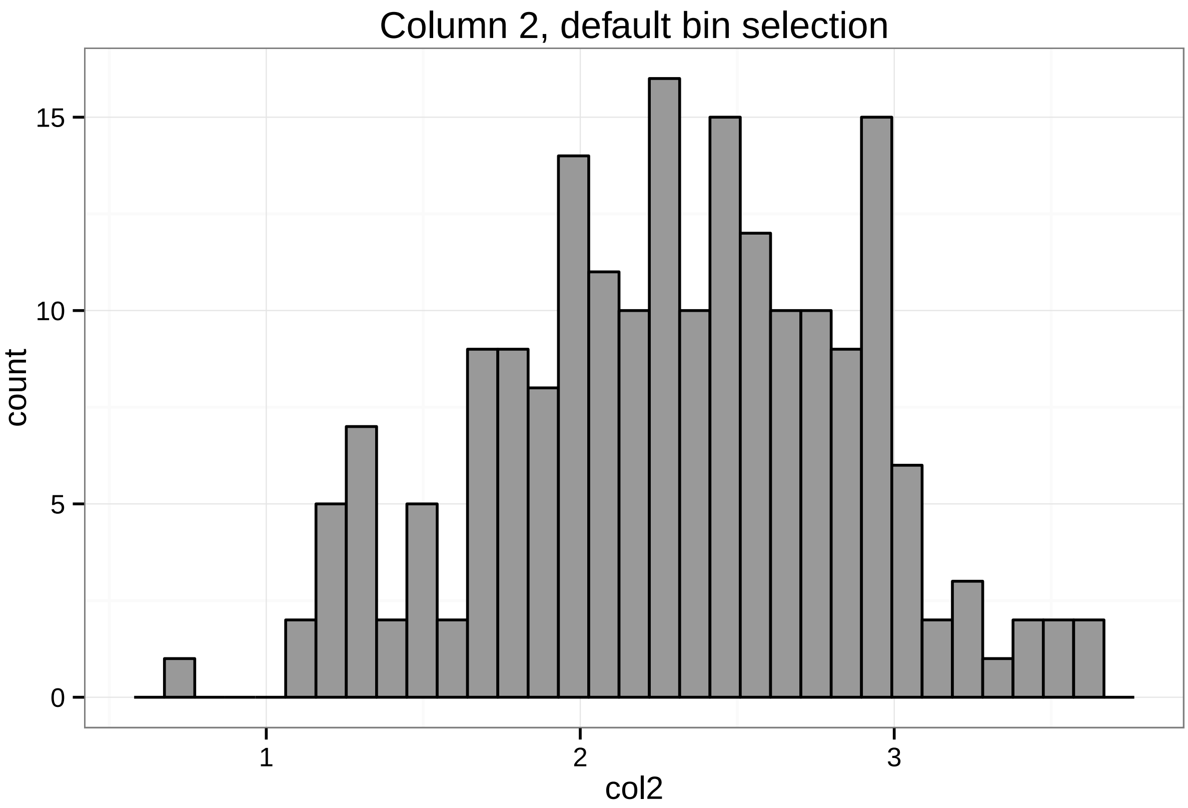
```
col4.vs.col5.plot <-
  ggplot(plot.df, aes(x = col5, y = col4)) +
  geom_line(linetype = 'dotdash', size = 1.3) +
  labs(title = 'Column 4 vs. Column 5', x = 'Column 5', y = 'Column 4')

print(col1.vs.col5.plot,
      vp = viewport(layout.pos.row = 1, layout.pos.col = 1))

print(col2.vs.col5.plot,
      vp = viewport(layout.pos.row = 1, layout.pos.col = 2))

print(col3.vs.col5.plot,
      vp = viewport(layout.pos.row = 2, layout.pos.col = 1))

print(col4.vs.col5.plot,
      vp = viewport(layout.pos.row = 2, layout.pos.col = 2))
```



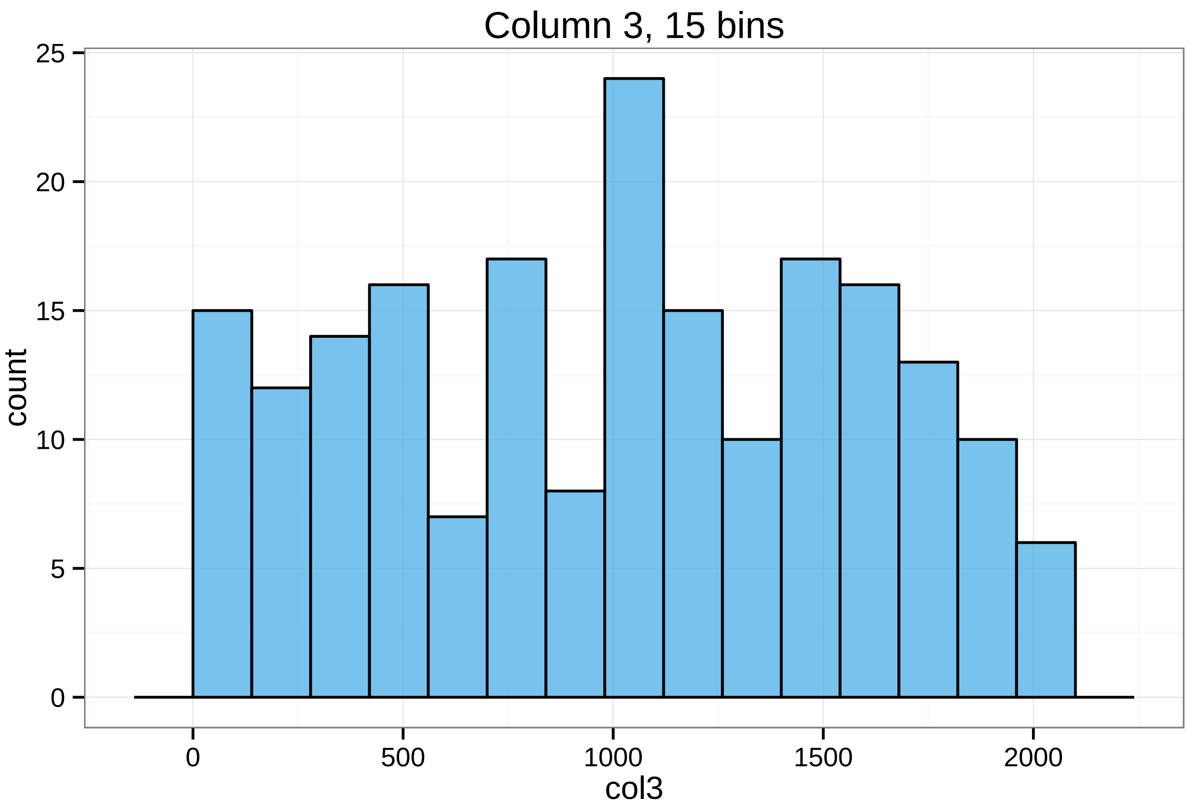
```
# histograms and kernel density estimates in ggplot2 ----------------------

# histograms

ggplot(plot.df, aes(x = col1)) +
  geom_histogram(fill = '#000000', binwidth = 1.25) +
  labs(title = 'Column 1, 30 bins') +
  theme_bw()
```



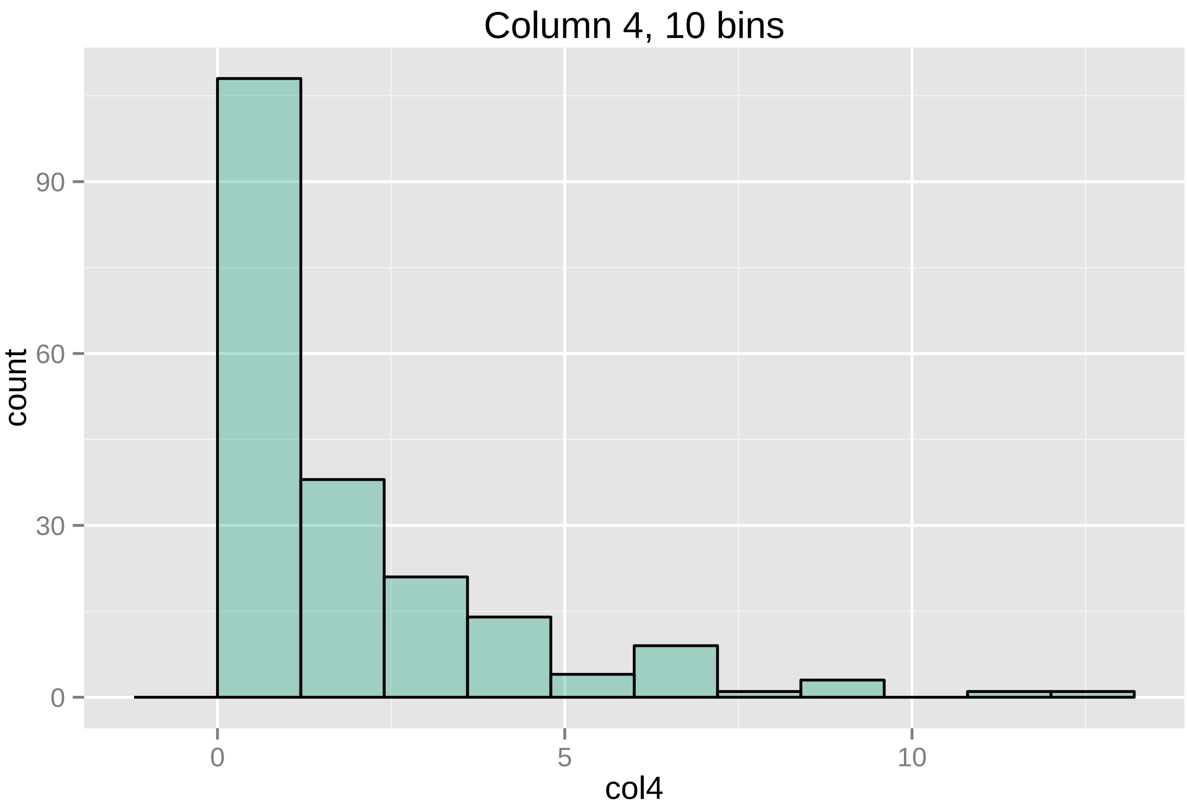
```
# fill controls inside bins, colour the outside
ggplot(plot.df, aes(x = col2)) +
  geom_histogram(fill = '#999999', colour = '#000000') +
  labs(title = 'Column 2, default bin selection') +
  theme_bw()
```



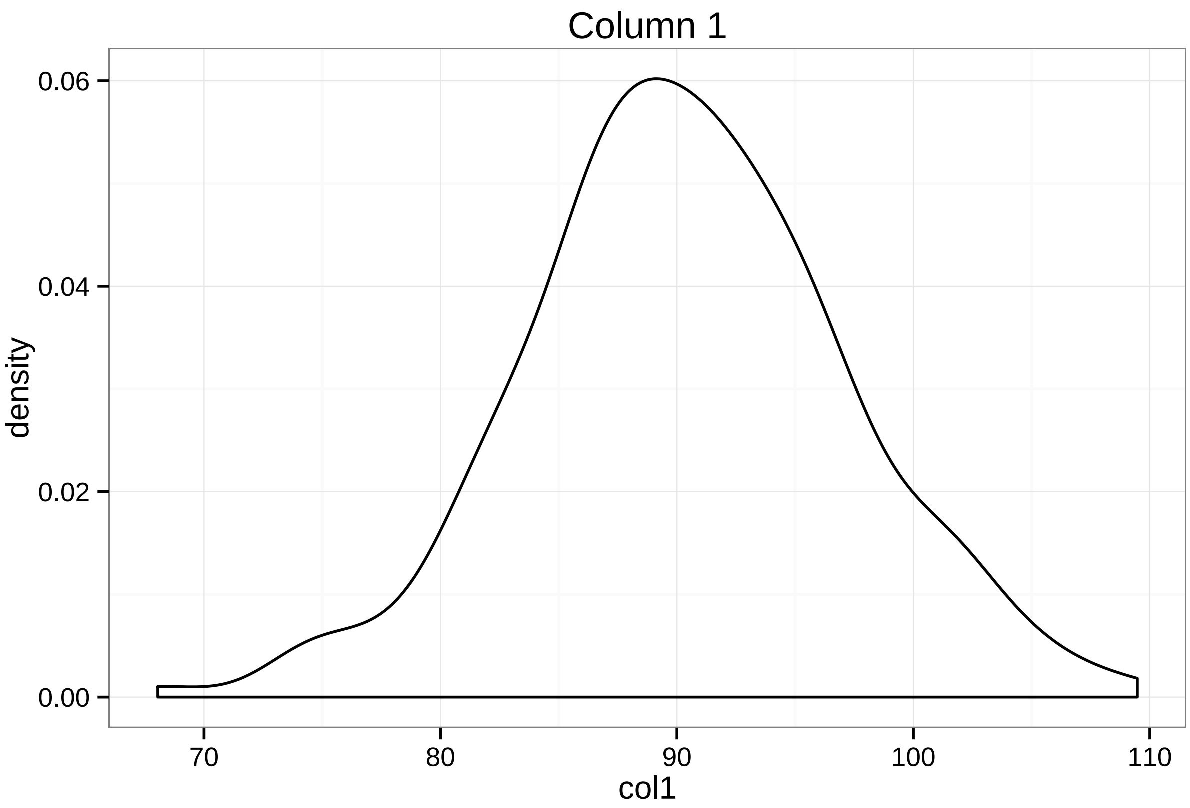
```
ggplot(plot.df, aes(x = col3)) +
  geom_histogram(fill = '#56B4E9', colour = '#000000',
                 binwidth = 140, alpha = 0.8) +
  labs(title = 'Column 3, 15 bins') +
  theme_bw()
```



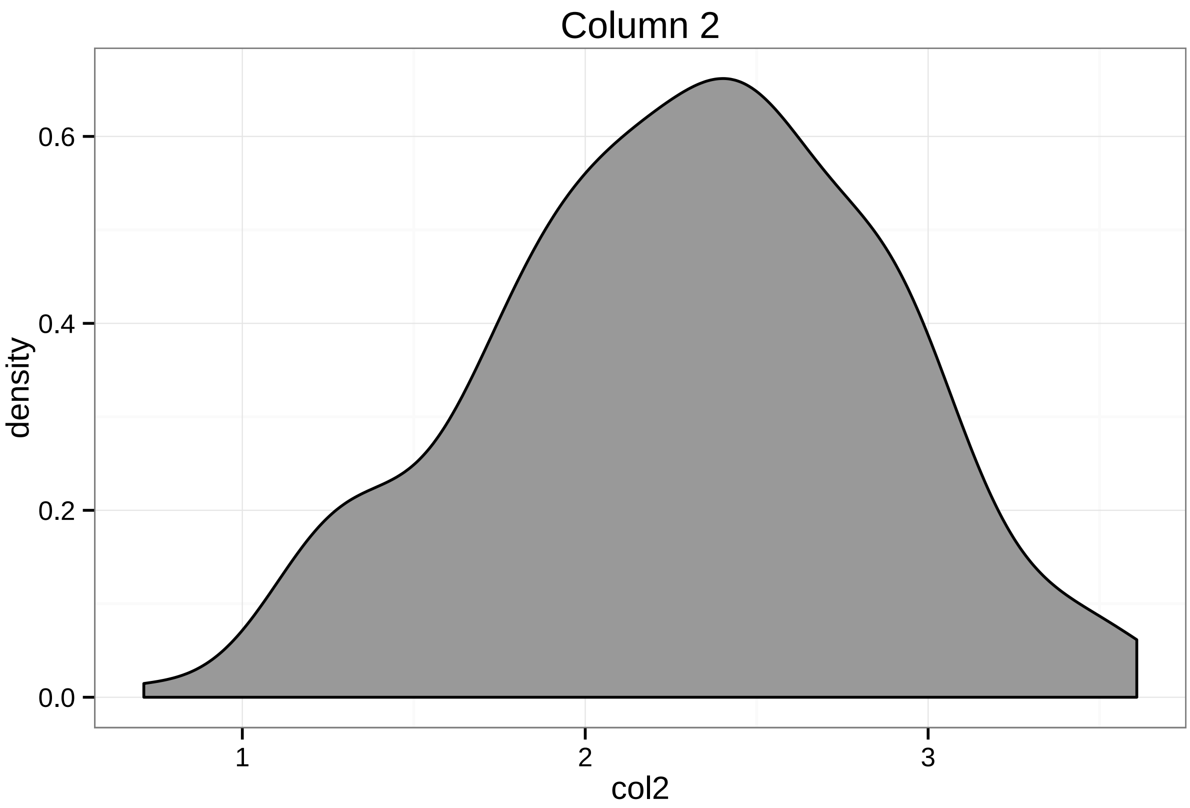
```
ggplot(plot.df, aes(x = col4)) +
  geom_histogram(fill = '#009E73', colour = '#000000',
                 binwidth = 1.2, alpha = 0.3) +
  labs(title = 'Column 4, 10 bins')
```



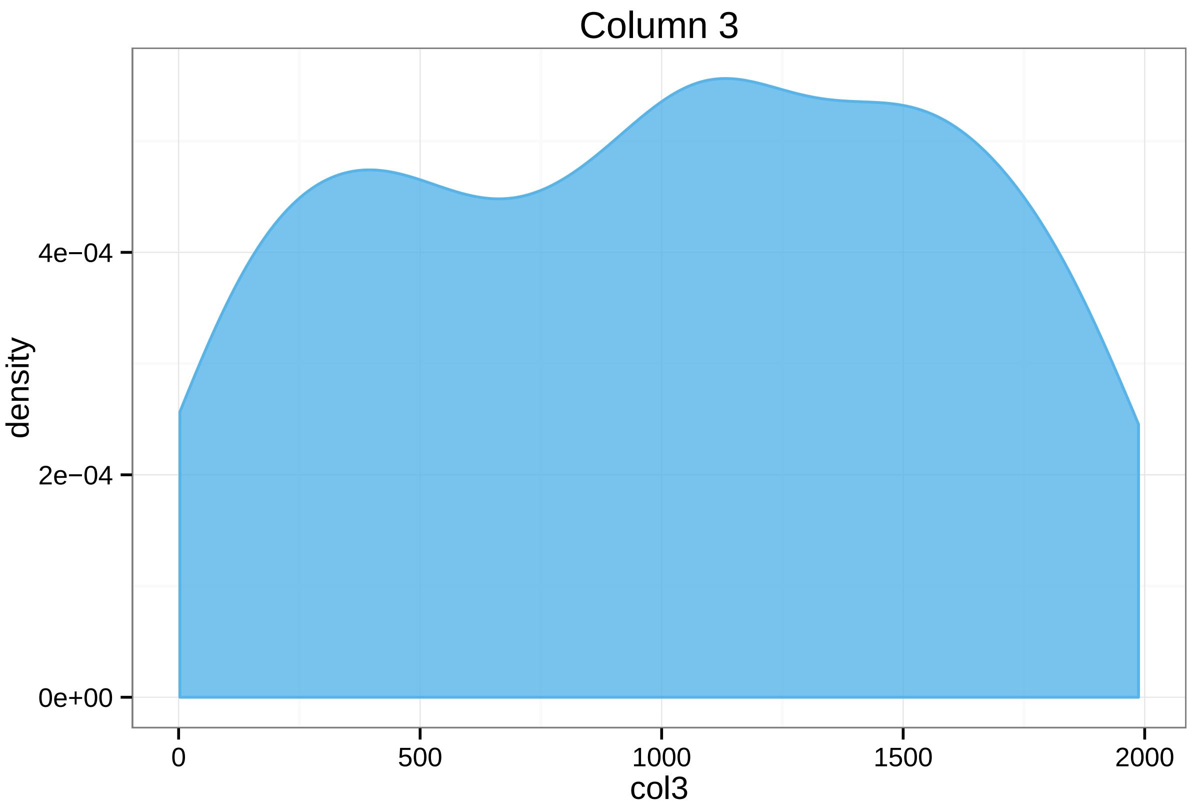
```
# kernel density estimates
# not that R/ggplot picks a different rule of thumb to determine bandwidth
# than pandas

ggplot(plot.df, aes(x = col1)) +
  geom_density(colour = '#000000') +
  labs(title = 'Column 1') +
  theme_bw()
```



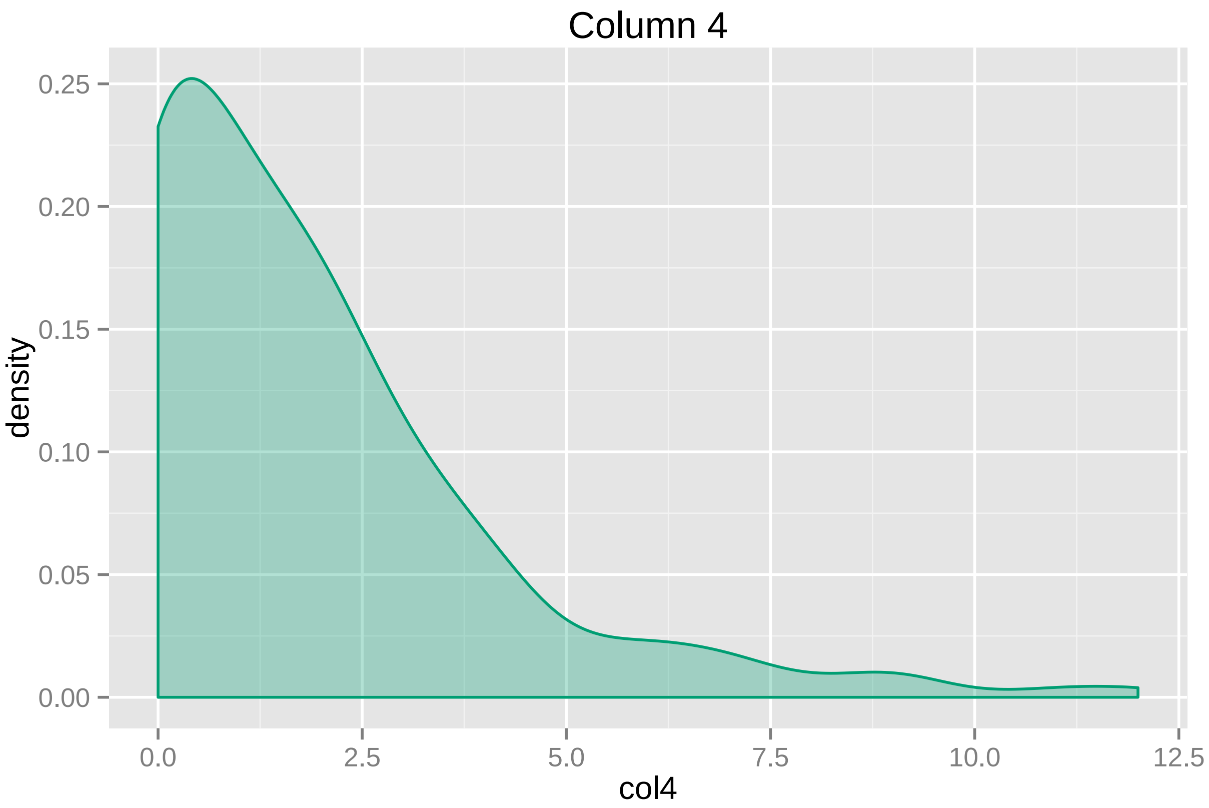
```
# fill controls inside estimate, colour the outside
ggplot(plot.df, aes(x = col2)) +
  geom_density(fill = '#999999', colour = '#000000') +
  labs(title = 'Column 2') +
  theme_bw()
```



```
ggplot(plot.df, aes(x = col3)) +
  geom_density(fill = '#56B4E9', colour = '#56B4E9', alpha = 0.8) +
  labs(title = 'Column 3') +
  theme_bw()
```



```
# notice that ggplot2 by defauly ends the KDE at the min and max values
ggplot(plot.df, aes(x = col4)) +
  geom_density(fill = '#009E73', colour = '#009E73', alpha = 0.3) +
  labs(title = 'Column 4')
```



```
# saving data ------------------------------------------------------------

# reusing data for advanced plotting script

save.path <- '~/GitHub/reproducible-research/Day-3/datasets'
save.image(file.path(save.path, '/basic-grouping-plotting.rda'))

sink()
```