Chapter 19 Practice

Jim Bang

Summarizing Data

Summarizing an Entire Data Frame

- Calculate the summary statistics for the *salary*, *sales*, *roe*, and *ros* variables.

Summaries with Categorical Variables

R handles factors much more intuitively than *Stata* (the original source of the data). The *factor()* function will create a factor from a variable from a variable with discrete values, but there is a slight three-step trick (involving matrix algebra) that you have to do first. - Step 1: Use *cbind* to combine the four mutually-exclusive industry variables into a matrix, *A*. - Step 2: Create a vector, *f*, that takes a discrete number of values equal to the number of factor categories (columns in *A*). We'll use (1, 2, 3, 4), but you could just as validly use four random numbers. - Step 3: Cross multiply *A* and *f*.

Summary by Categorical Group

Now, summarize your entire dataset *by industry*, and *exclude* the following:

1. The four individual-industry indicator variables (use the factor variable from the previous step instead).
2. The log of salary and the log of sales.

Use two places after the decimal place for all continuous variables.

Summarizing Relationships between Variables

Calculate the correlation matrix among **only** *salary*, *sales*, *roe*, and *ros*.

Other "corrgram" Options

If we want to make it look cleaner, we can set the color to black using the *color.regions* function and the *colorRampPalette()* function. Do this for the numerical correlation matrix for the *ceosal1* data that we started in class by adding the appropriate option(s).

Factor Variable Tables

Do the following: - Generate the factor variables "haskids" and "marriage" for kids and ratemarr labels with labels "no"/"yes" and "very unhappy"/"unhappy"/"average"/"happy"/"very happy," respectively. - Create a proportions contingency table of whether or not a couple has kids conditional on marital happiness. - Create a proportions contingency table of marital happiness conditional on whether or not a couple has kids.

Attractive Crosstab Tables

- Replicate the previous tables using "tbl_cross" (using percent to generate percentages instead of proportions) to generate a more attractive layout that you can save as html.
- Combine counts and percentages in one table with each cell displaying percentages in parentheses next to its count.

Base Graphics

Add the following to your plots from the in-class examples:

- Set the number of breaks or bins to 30;
- A *dashed* normal curve over the histogram with mean equal to $\bar{x}_{salary}$ and standard deviation equal to $\hat{\sigma}_{salary}$;
- *Dotted* vertical and horizontal lines over the scatterplot at the means of CEO salaries and ROE, respectively.

Make sure your plots have descriptive (English) titles: "Histogram of Salary," "Salary versus ROE," "Salary," and "ROE."

As you can see, CEO salaries are very abnormal! WWWD (what-would-Wooldridge-do)? Take the logs?

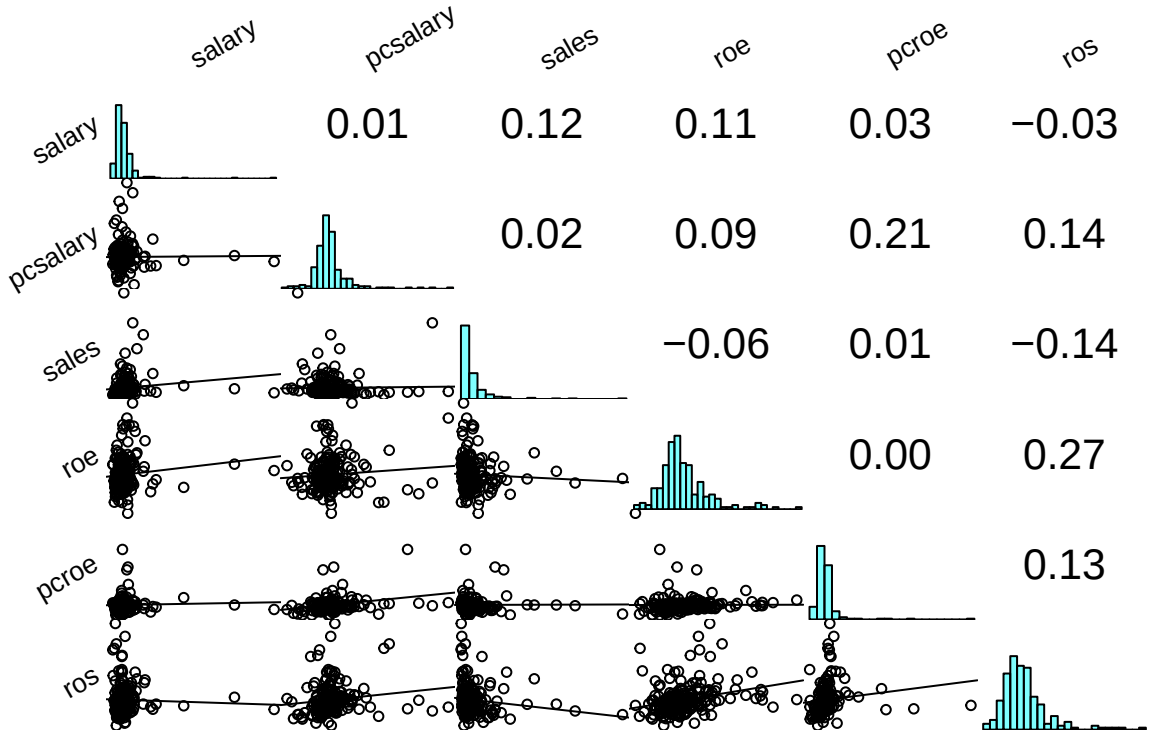
Other "corrgram" Options

To really up your game with pairs of variables, you can put everything in a corrgram() plot. Did you notice how the upper-right triangular half of the correlation matrix is the same as the lower-left? What if we could put coefficients in the upper half, scatterplots in the lower half, and histograms (almost) down the diagonal? Let's do that! (And keep the fonts black, please!)

This uses the built-in 'panel.density' option to approximate the distribution of the x's. For a 'true' histogram, we would have to specify our own function, panel.hist. We can also specify functions to add a regression line, or fancy "spline" fits.

```
# Create a user-defined function, panel.hist, to make a histogram along the diagonals.
panel.hist <- function(x, ...) {
    usr <- par("usr")
    on.exit(par(usr))
    par(usr = c(usr[1:2], 0, 1.5))
    his <- hist(x, plot = FALSE, breaks = 30)
    breaks <- his$breaks
    nB <- length(breaks)
    y <- his$counts
```

```
    y <- y/max(y)
    rect(breaks[-nB], 0, breaks[-1], y, col = rgb(0, 1, 1, alpha = 0.5), ...)
}
# Create a user-defined function, panel.lm, to add a quick regression line to a panel.plot.
panel.lm <- function (x, y, corr = NULL, col.regions, cor.method, ...) {
    if (!is.null(corr))
        return()
    plot.xy(xy.coords(x, y), type = "p", ...)
    abline(lm(y ~ x)) # The lm command runs a simple linear regression.
}
# Create a user-defined function, panel.spline, to add a fancy kinked linear spline to a panel.plot.
panel.spline <- function (x, y, corr = NULL, col.regions, cor.method, ...) {
    if (!is.null(corr))
        return()
    plot.xy(xy.coords(x, y), type = "p", ...)
    lines(smooth.spline(x, y), type = 'l')
}
corrgram(ceosal1[, -which(names(ceosal1) %in%
                          c('indus', 'finance', 'consprod', 'utility', 'lsalary', 'lsales'))],
         upper.panel = panel.cor,
         lower.panel = panel.lm,
         diag.panel = panel.hist,
         text.panel = NULL,
         cex.cor = 2, # This line gives an error, but does what you want!
         outer.labels = list(top  = list(names(ceosal1), cex = 1.25, srt = 30),
                             left = list(names(ceosal1), cex = 1.25, srt = 30)),
         col.regions = colorRampPalette(c('black')))
```


salary
pcsalary
sales
roe
pcroe
ros
0.01
0.12
0.11
0.03
−0.03
0.02
0.09
0.21
0.14
−0.06
0.01
−0.14
0.00
0.27
0.13