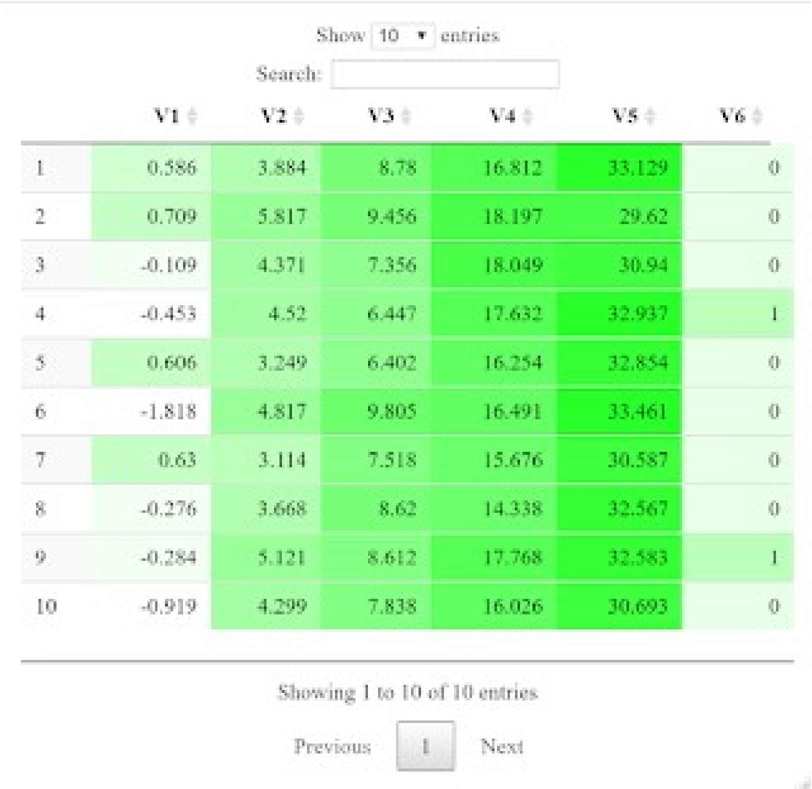
Here we generate data similar to that in Example 3, but with average values growing by column

```
set.seed(12345)
df = as.data.frame(
  cbind(round(rnorm(10, mean = 0), 3),
  round(rnorm(10, mean = 4), 3),
  round(rnorm(10, mean = 8), 3),
  round(rnorm(10, mean = 16), 3),
  round(rnorm(10, mean = 32), 3),
  sample(0:1, 10, TRUE)))
```

Using the code in the example — modified to green — the darker values naturally appear in columns **V4** and **V5**.

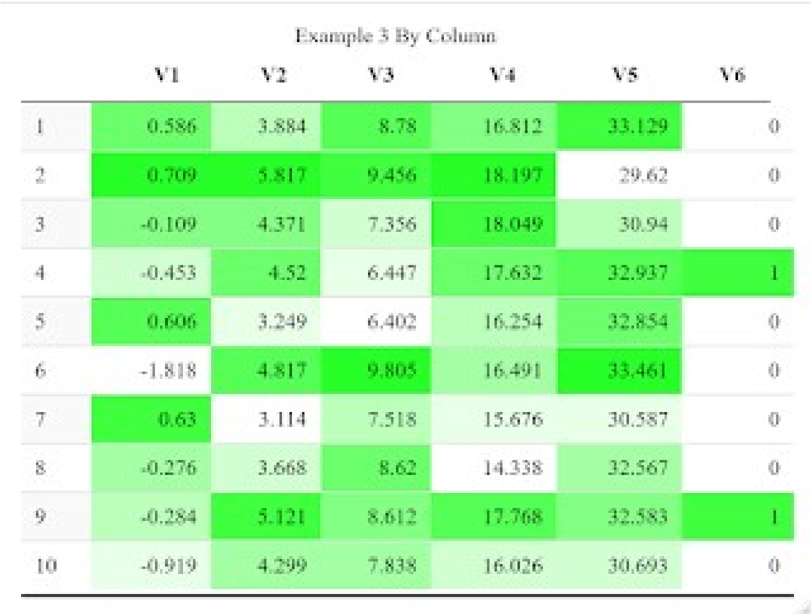


	V1	V2	V3	V4	V5	V6
1	0.586	3.884	8.78	16.812	33.129	0
2	0.709	5.817	9.456	18.197	29.62	0
3	-0.109	4.371	7.356	18.049	30.94	0
4	-0.453	4.52	6.447	17.632	32.937	1
5	0.606	3.249	6.402	16.254	32.854	0
6	-1.818	4.817	9.805	16.491	33.461	0
7	0.63	3.114	7.518	15.676	30.587	0
8	-0.276	3.668	8.62	14.338	32.567	0
9	-0.284	5.121	8.612	17.768	32.583	1
10	-0.919	4.299	7.838	16.026	30.693	0

Showing 1 to 10 of 10 entries

But that's not what we want.
For each column to have it's own scale, simply apply RStudio's algorithm to each column of `df` in a loop. The trick to notice is that `formatStyle` wants a `datatable` object as its first argument, and produces a `datatable` object as its result. Therefore, start off with a plain-Jane `datatable` and successively format each column, saving the result each time. Almost like building a **ggplot**. At the end, view the final result.

```
# Start with a (relatively) plain, unformatted datatable object
dt <- DT::datatable(df,
                    options = list(dom = 't', ordering = FALSE),
                    caption = "Example 3 By Column")
# Loop through the columns formatting according to that column's distribution
for (j in seq_along(df)) {
  # Create breaks for shading column values high to low
  brks <- stats::quantile(x <- df[[j]], probs = seq(.05, .95, .05), na.rm =
TRUE)
  # Create shades of green for backgrounds
  y <- round(seq(255, 40, length.out = length(brks) + 1), 0)
  clrs <- paste0("rgb(", y, ", 255,", y, ")")
  # Format cells in j-th column
  dt <- DT::formatStyle(dt, j, backgroundColor = DT::styleInterval(brks, clrs))
}
dt
```



Example 3 By Column

	V1	V2	V3	V4	V5	V6
1	0.586	3.884	8.78	16.812	33.129	0
2	0.709	5.817	9.456	18.197	29.62	0
3	-0.109	4.371	7.356	18.049	30.94	0
4	-0.453	4.52	6.447	17.632	32.937	1
5	0.606	3.249	6.402	16.254	32.854	0
6	-1.818	4.817	9.805	16.491	33.461	0
7	0.63	3.114	7.518	15.676	30.587	0
8	-0.276	3.668	8.62	14.338	32.567	0
9	-0.284	5.121	8.612	17.768	32.583	1
10	-0.919	4.299	7.838	16.026	30.693	0

Actuaries in the crowd might recognize the image at the top of the post as the table of link ratios from the `GenIns` dataset…