Heatmaps in R

Kayvan Jalali & Carlos Puerta

9/27/2023

1. What is a heatmap?
2. Pheatmap Package
3. Applying Heatmaps

Section 1

What is a heatmap?

What is a heatmap?

A heatmap is graphic that will display your data in a colorful grid. This is great for seeing trends and patterns in your data.

Generally, heatmaps are used to represent data where you have 2 categorical variables with a third continuous variable, though this is not explicitly necessary.

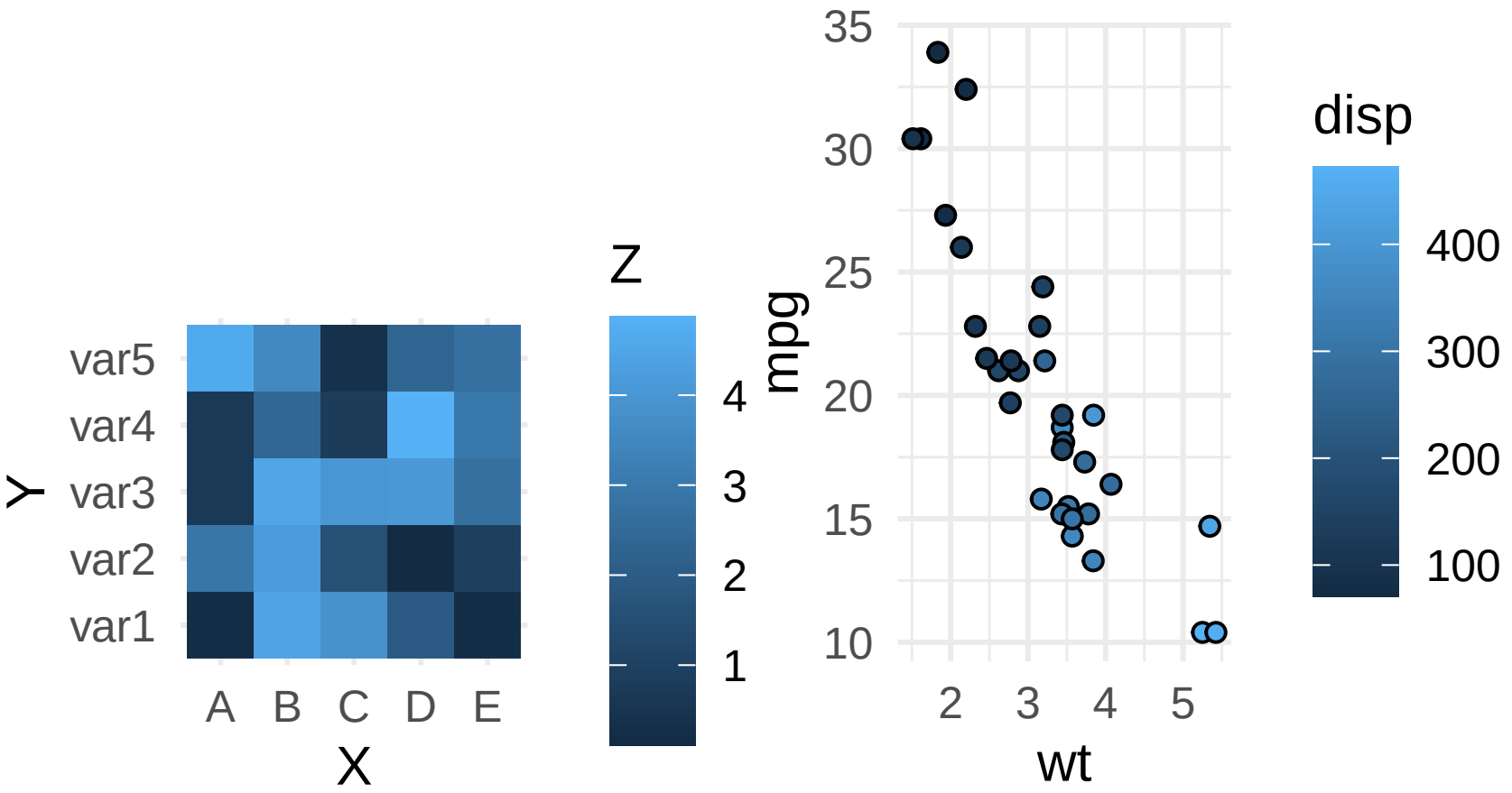
Advantages

- Adds another variable
- Easier to interpret
- Color is easy to parse

Disadvantages

- Need right choice of colors and data
- Only useful if clear
- A different plot may be better

Examples


var5
var4
var3
var2
var1
Y
A
B
C
D
E
X
Z
4
3
2
1
mpg
35
30
25
20
15
10
2
3
4
5
wt
disp
400
300
200
100

Heatmap Colors

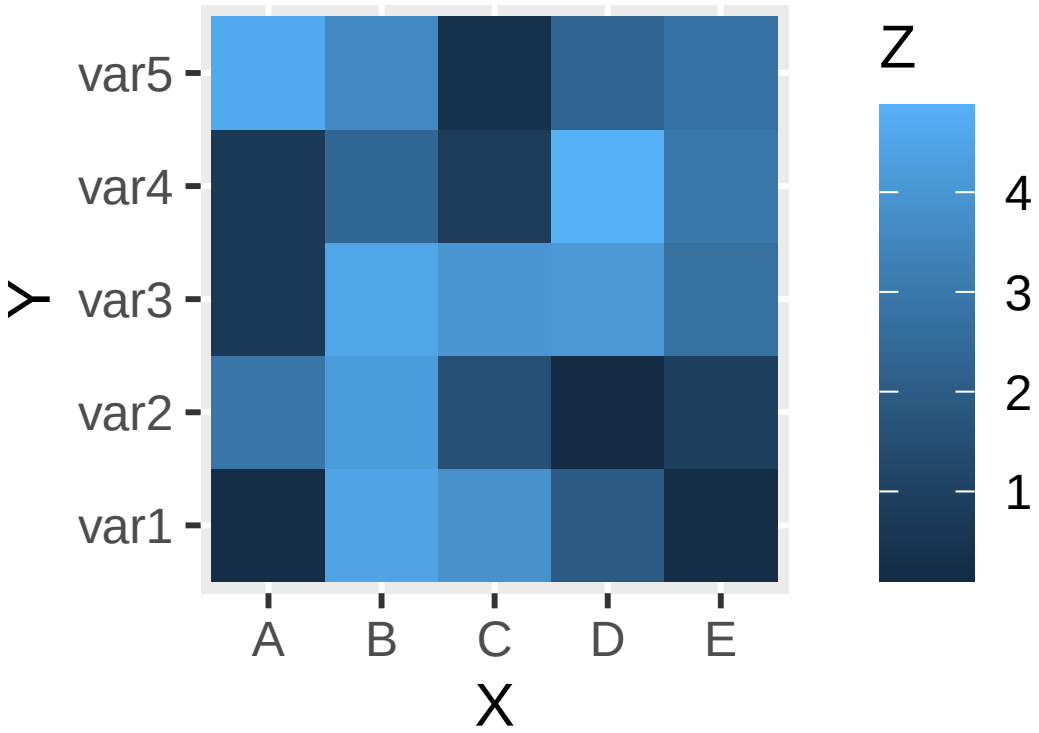
Heatmaps are applicable to both continuous and discrete variables. When working with discrete variables, distinguishing between similar colors can be challenging. To improve visibility, it is recommended to use a color palette of no more than nine colors, such as ROYGBIV, black, and white.

In the case of continuous variables, you have the option to use a sequential color scale, transitioning from a lighter hue to a darker one or vice versa. Alternatively, you can create a divergent color scale, transitioning between two distinct colors. A divergent color scale may be more useful when the center and ends of the value range are meaning full.

ggplot2

You can create a simple and quick heatmap using the geom_tile() function in ggplot2, but these heatmaps are very limited in their functionality.

```
ggplot(data, aes(X, Y, fill = Z)) +
  geom_tile() + coord_equal()
```



Section 2

Pheatmap Package

Pheatmap Package

Pheatmap (Pretty Heatmaps) is a package for R that supercharges heatmaps and allows you to create incredibly complex and helpful heatmaps. Although it is slightly more difficult to use when compared to ggplot, it is highly specialized and focused. It can plot more than 3 variables in a single heatmap, and more importantly, it groups rows and/or columns.

The grouping specifically can be incredibly helpful for visually spotting trends and similarities.

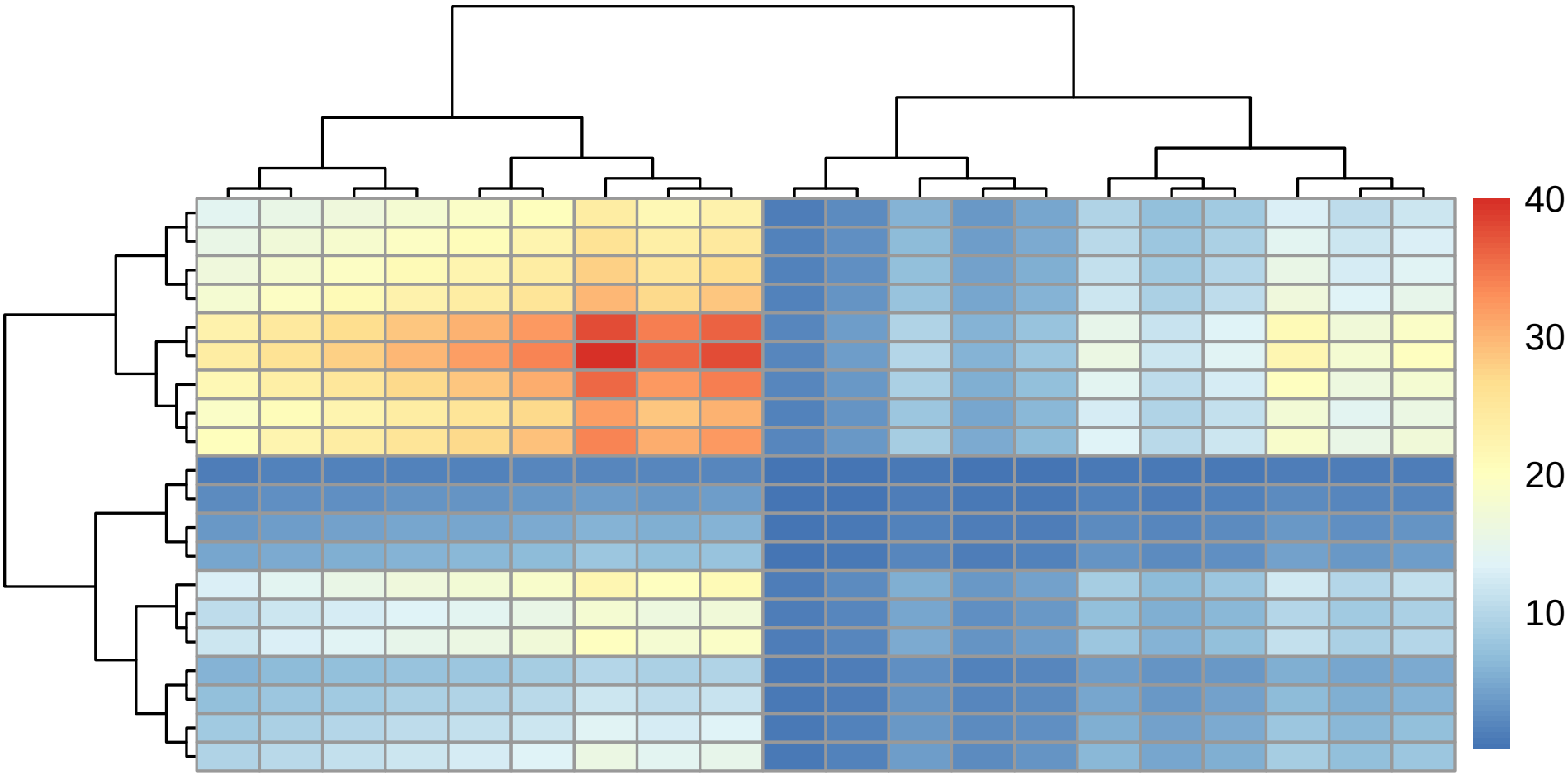
Dendrogram

Dendrograms, create a higherarchy of similar groups. Eventhough they can be standalone, and are not always seen with heatmaps, pheatmap includes dendrograms by default. Groups that are most similar get pooled together until there are no more groupings possible. The distance between groups directly reflects the difference between groups. By default these groupings are made using the rows or columns euclidean distance.

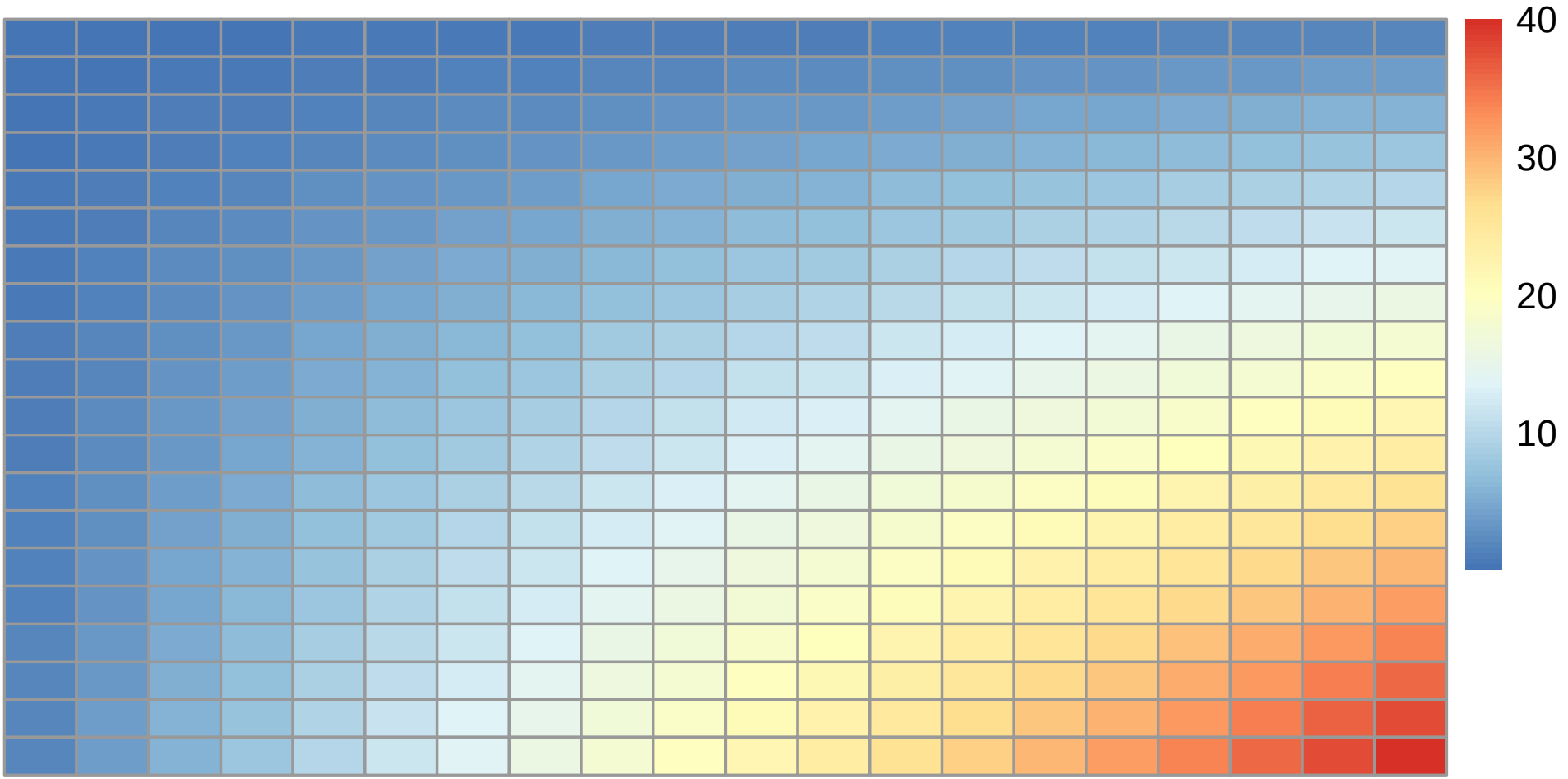
A Bad Tournament

You can think of it as a tournament bracket where seeding determines your matchups, your initial matches will be the strongest teams against each other and the weakest teams against each other. The distance between the connections symbolizes the overall difference in skill.

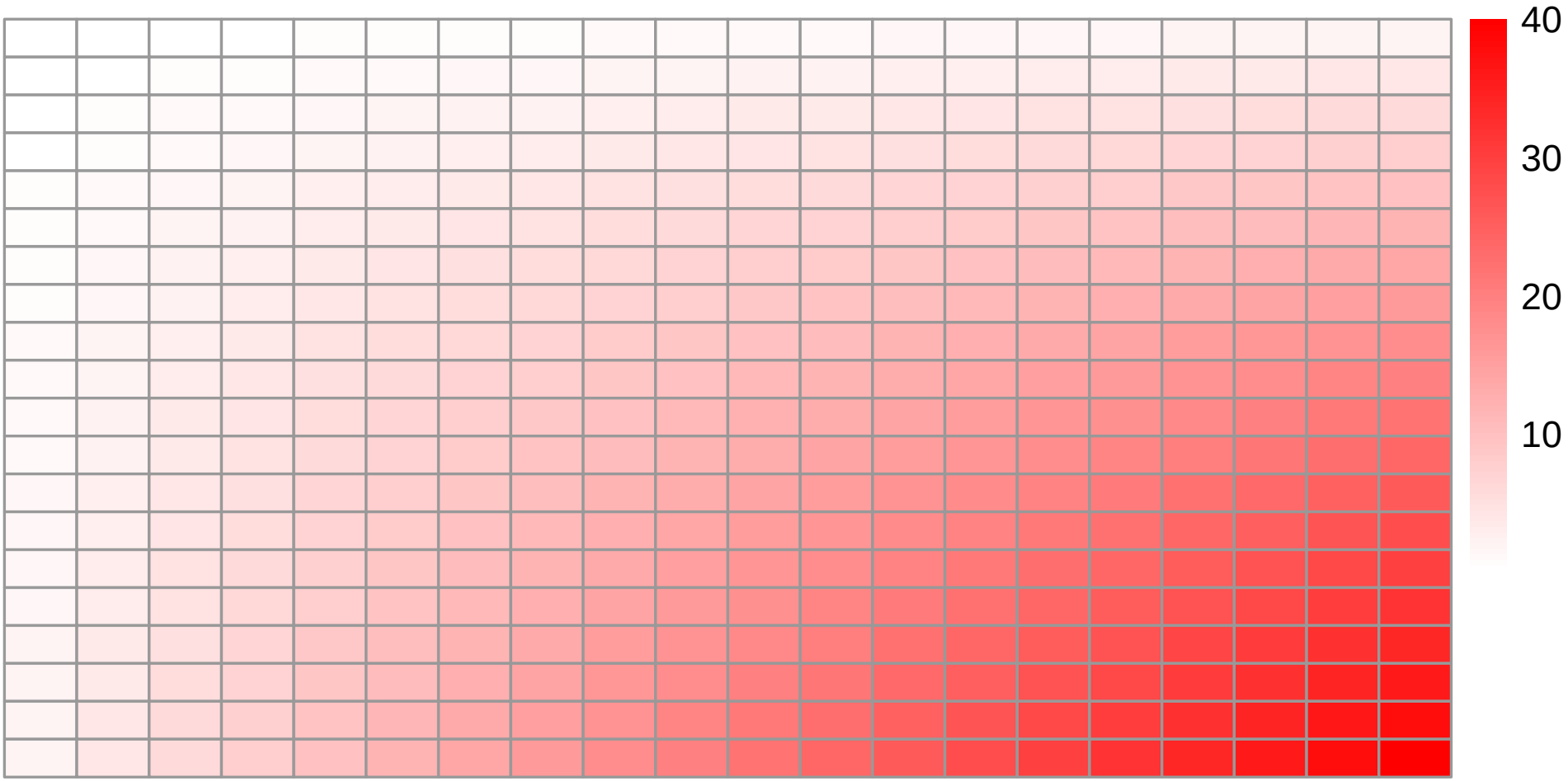
Just Pheatmap



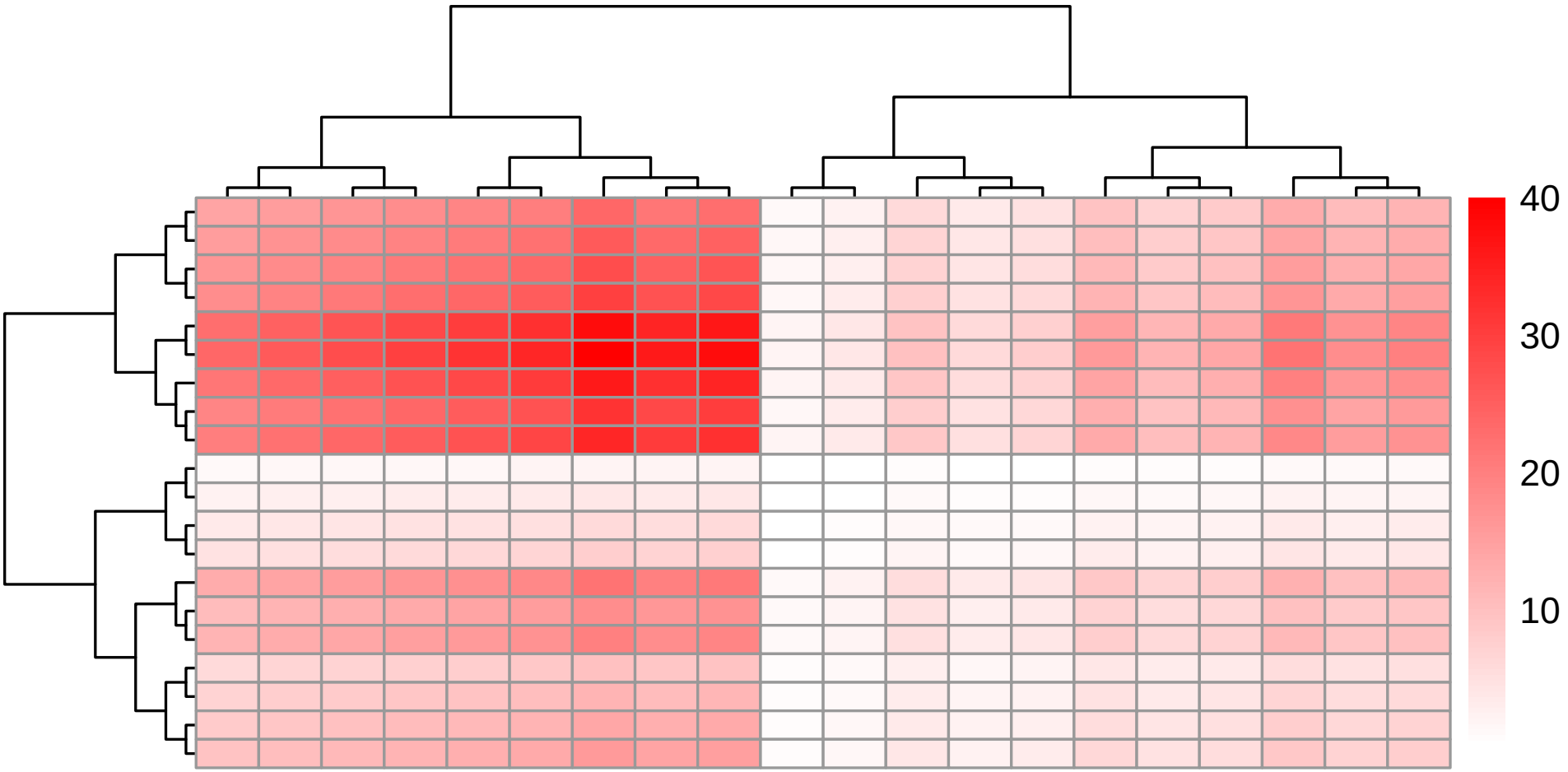
No Dendrogram



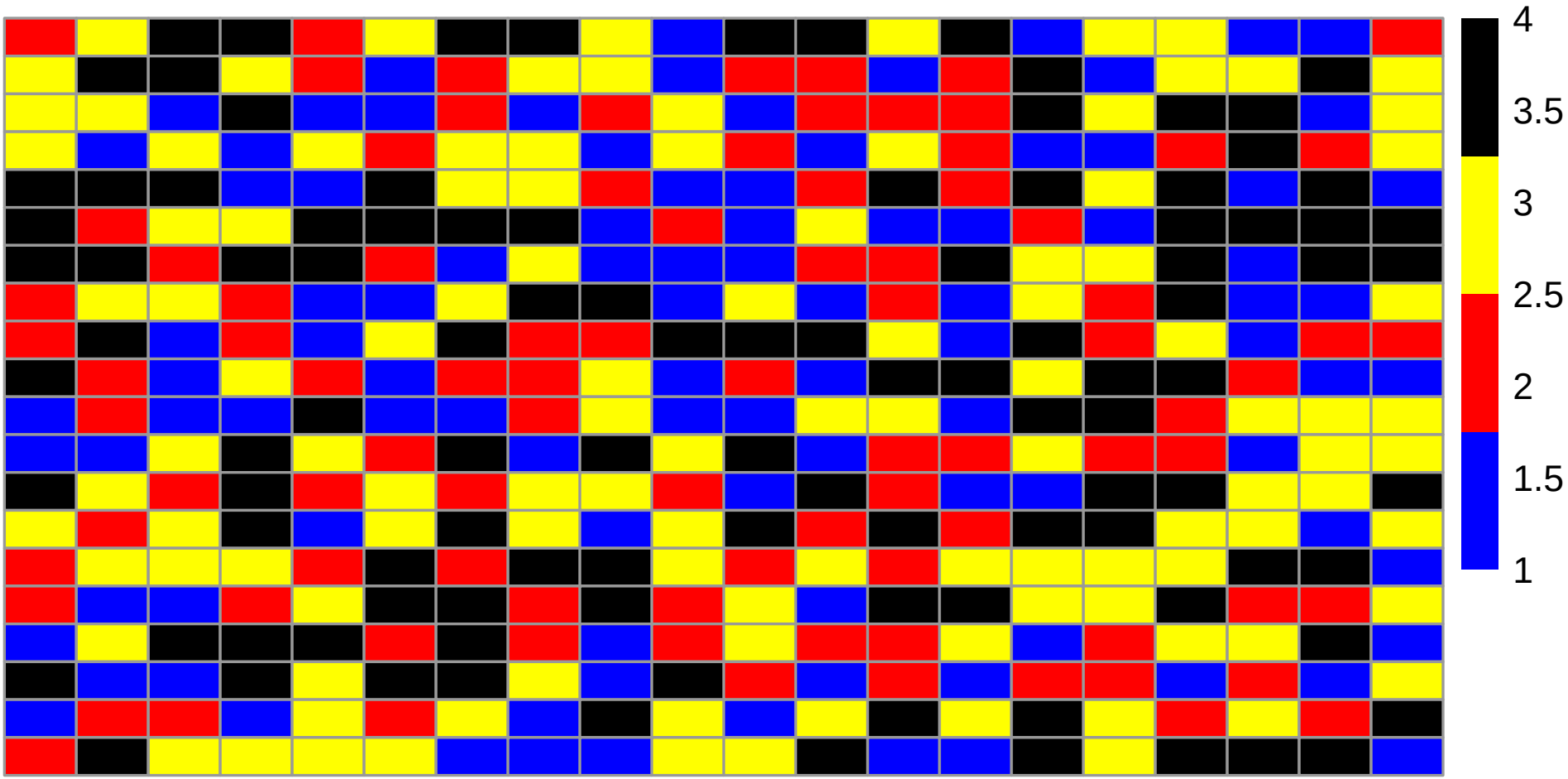
Color



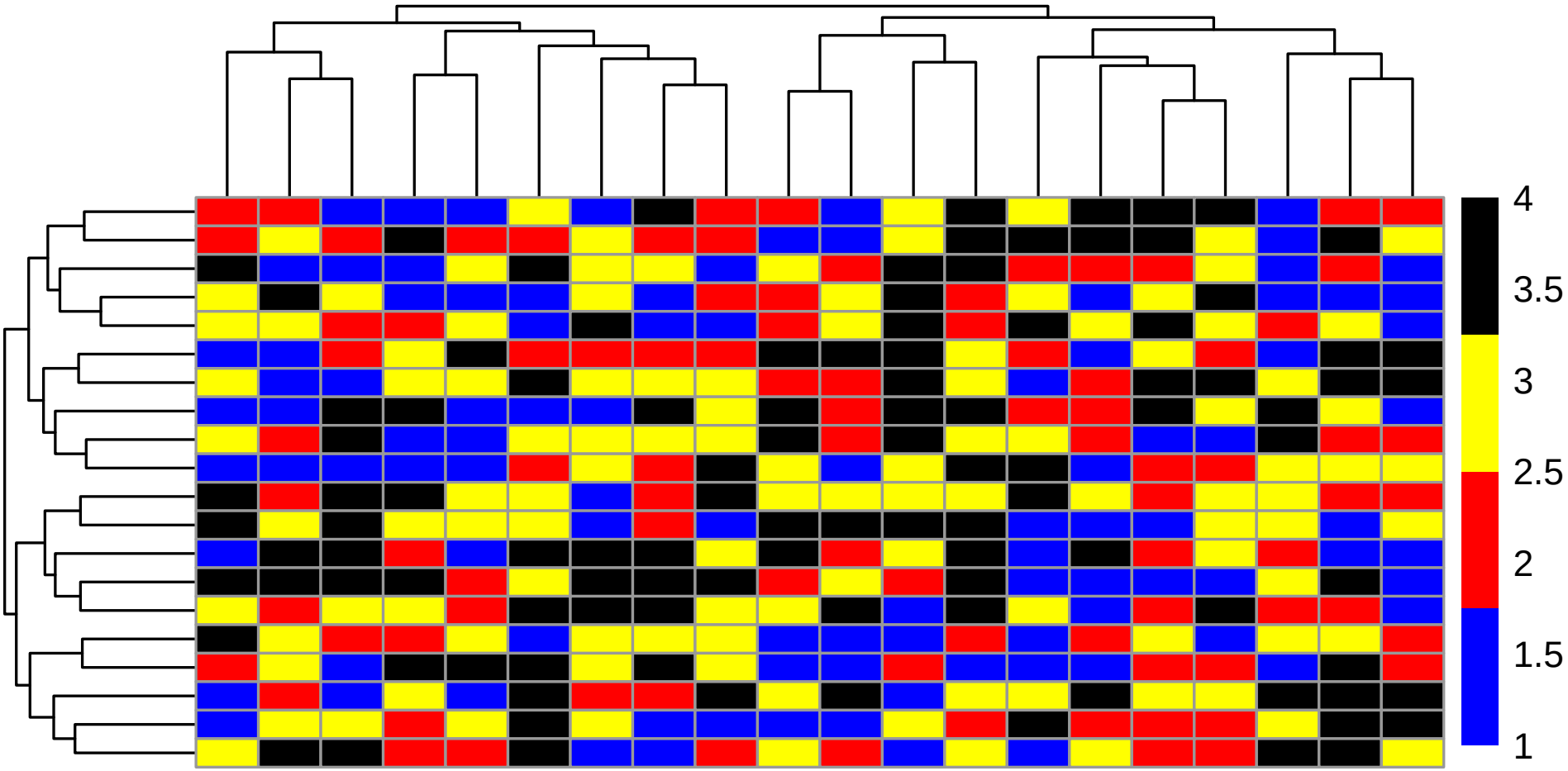
Pheatmap example



Pheatmap example



Pheatmap example



Section 3

Applying Heatmaps

Transform Diamonds Data

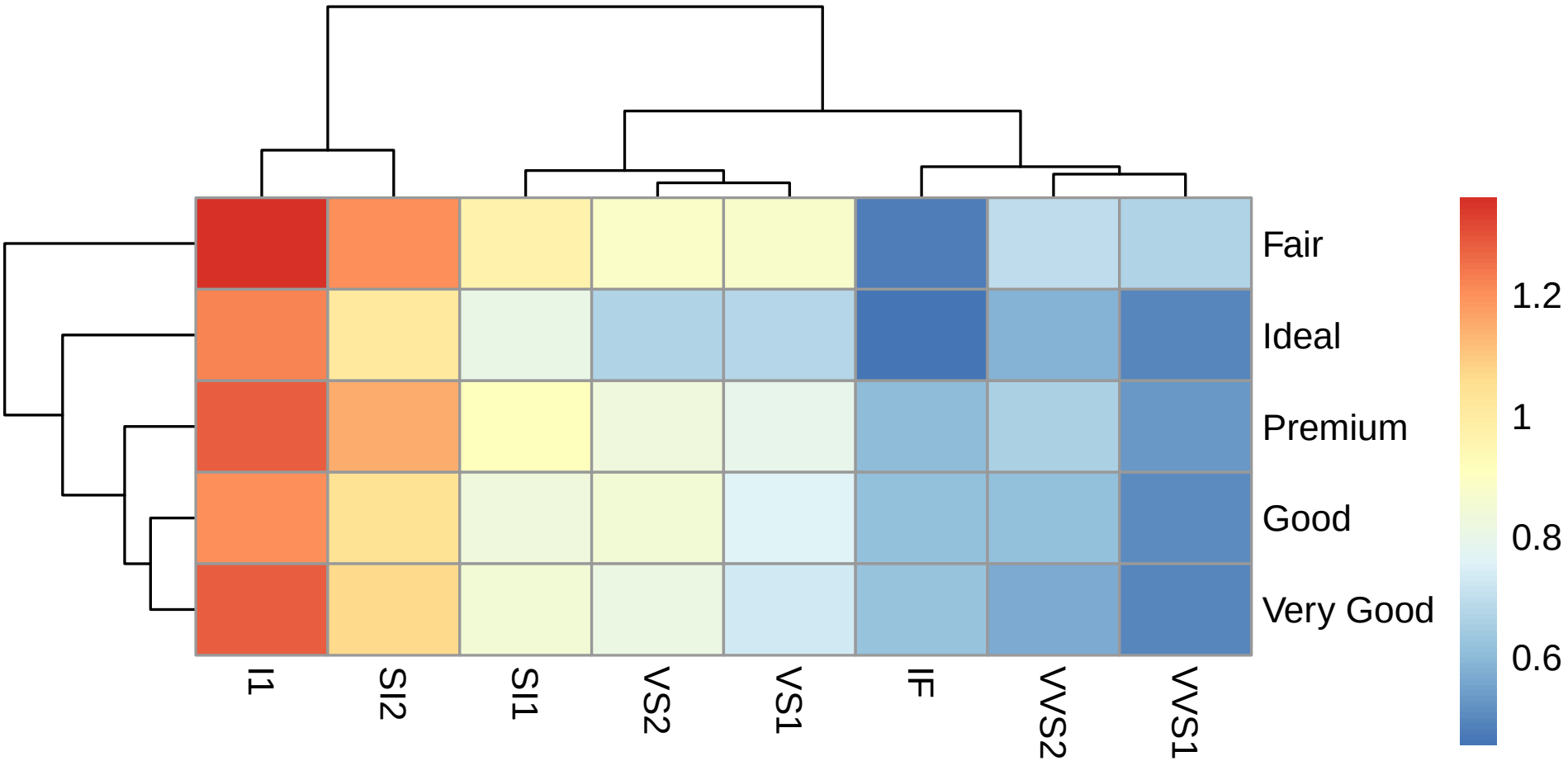
```
df <- diamonds %>%
  group_by(cut, clarity) %>%
  summarise(avg_carat =  mean(carat)) %>%
  pivot_wider(names_from = clarity, values_from = avg_carat) %>%
  column_to_rownames("cut")
```

Transform Diamonds Data

```
##                  I1       SI2       SI1       VS2       VS1
## Fair       1.361000 1.203841 0.9646324 0.8852490 0.8798235
## Good       1.203021 1.035227 0.8303974 0.8507873 0.7576852
## Very Good  1.281905 1.064338 0.8459784 0.8111810 0.7333070
## Premium    1.287024 1.144161 0.9086014 0.8337742 0.7933082
## Ideal      1.222671 1.007925 0.8018076 0.6705660 0.6747144
```

Normal Heatmap

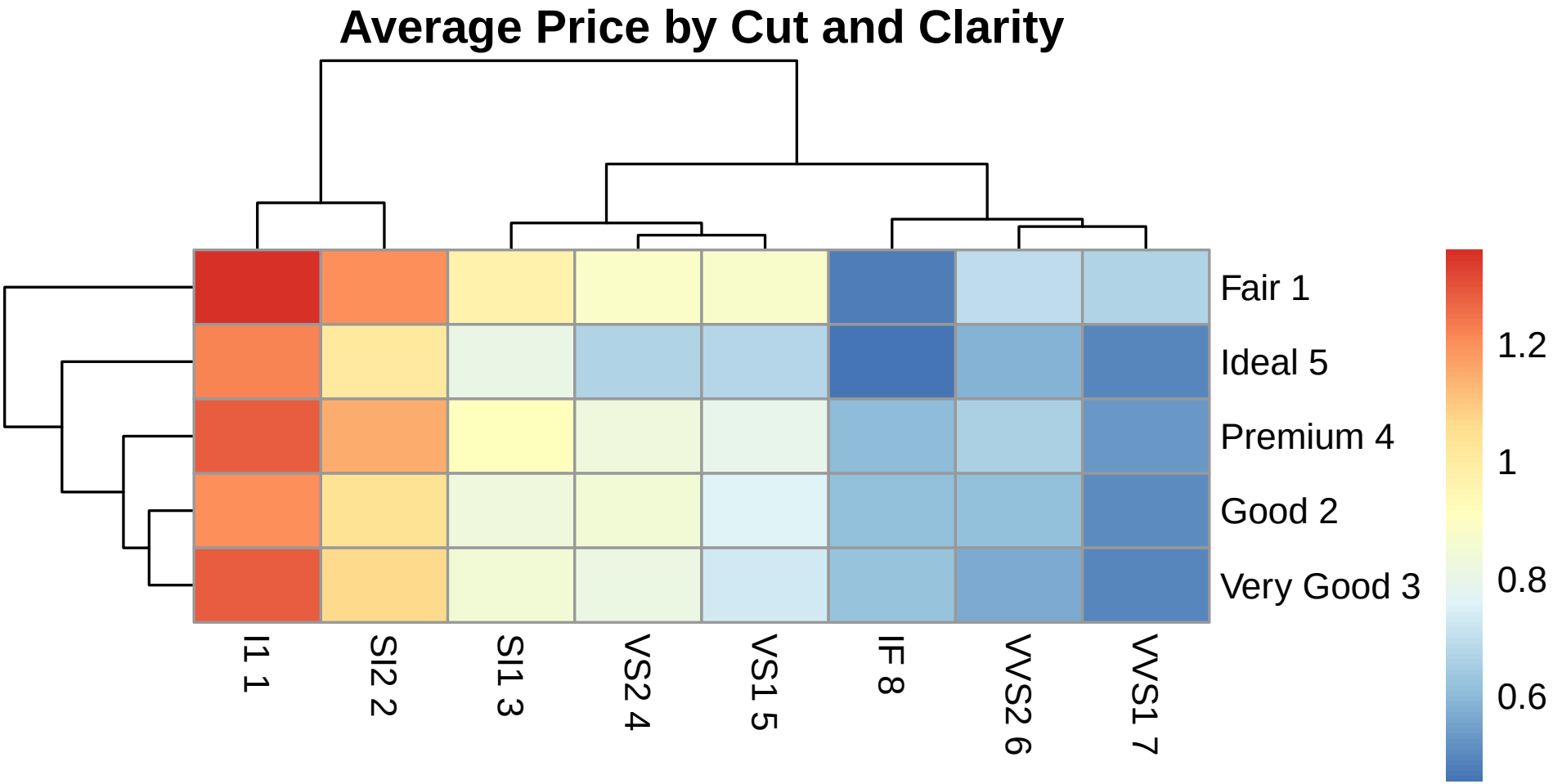
```
pheatmap(df)
```



Labels

```
pheatmap(df,
  labels_row = paste(rownames(df), 1:5),
  labels_col = paste(colnames(df), 1:8),
  main = "Average Price by Cut and Clarity",
  show_rownames = TRUE,
  show_colnames = TRUE,
)
```

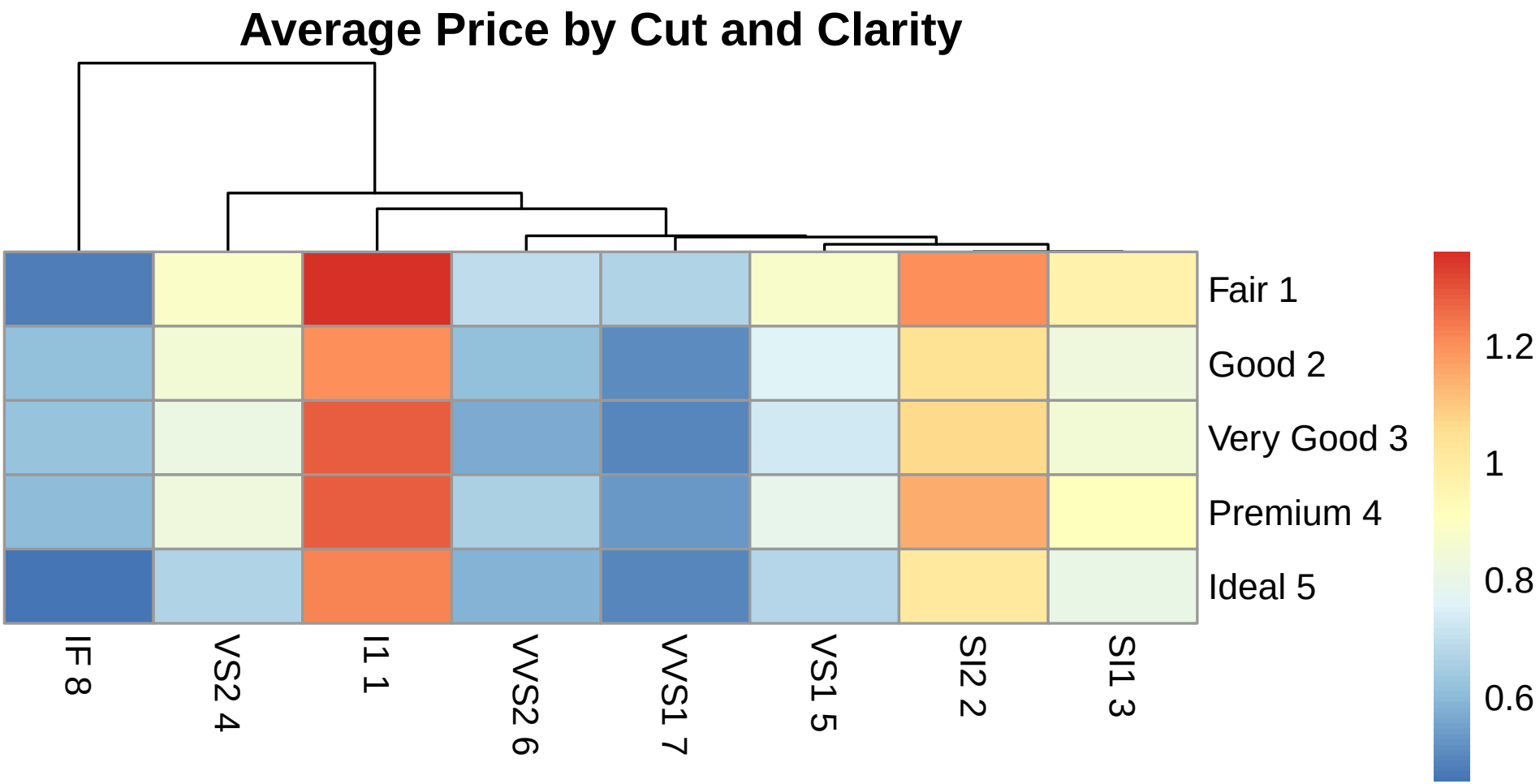
Labels


Average Price by Cut and Clarity
Fair 1
Ideal 5
Premium 4
Good 2
Very Good 3
I1 1
SI2 2
SI1 3
VS2 4
VS1 5
IF 8
VVS2 6
VVS1 7
1.2
1
0.8
0.6

Dendrogram

```
pheatmap(df,
  labels_row = paste(rownames(df), 1:5),
  labels_col = paste(colnames(df), 1:8), # overlay
  main = "Average Price by Cut and Clarity",
  cluster_rows = FALSE,
  cluster_cols = TRUE,
  clustering_distance_cols = "correlation"
)
```

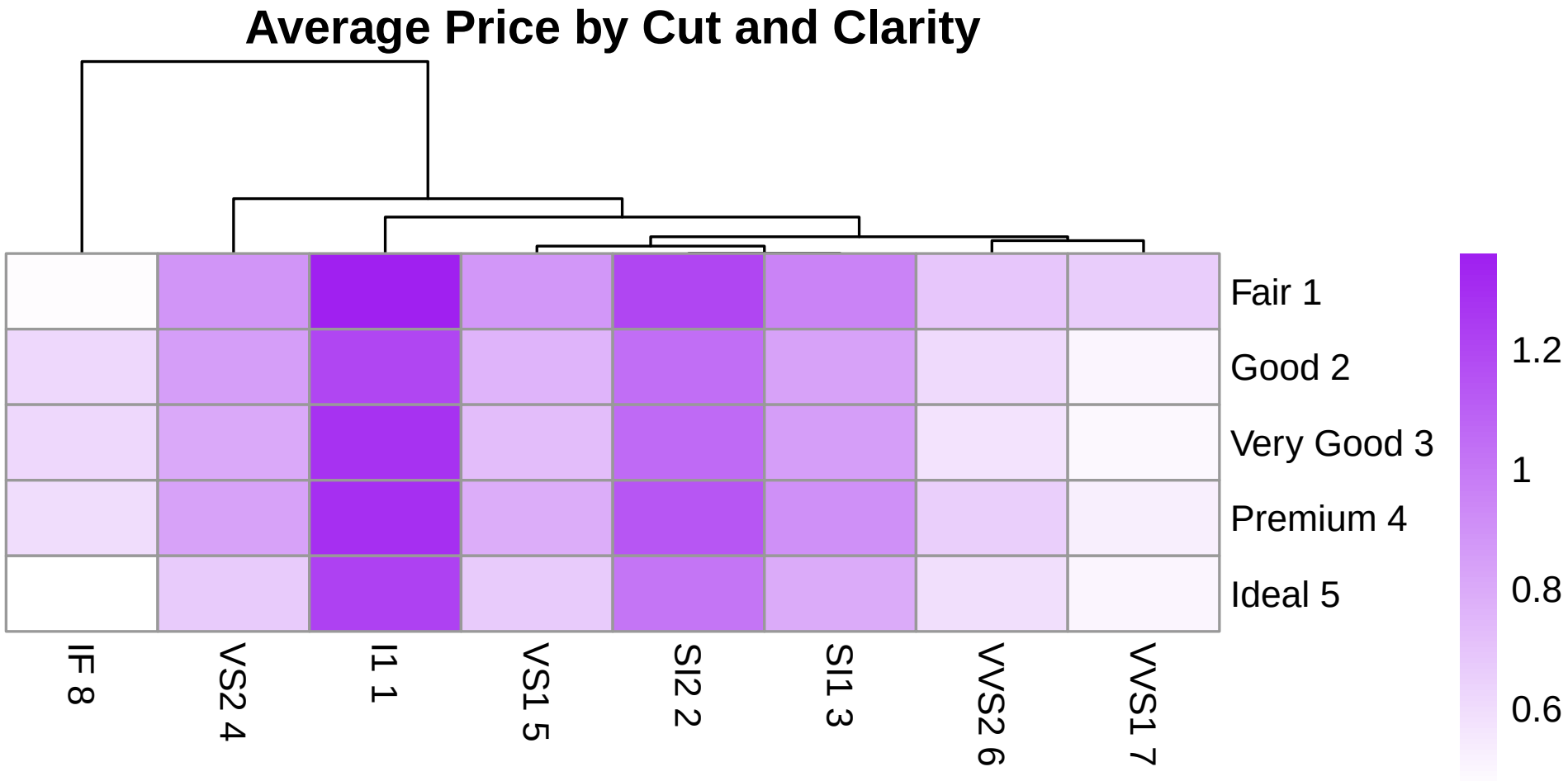
Dendrogram


Average Price by Cut and Clarity
Fair 1
Good 2
Very Good 3
Premium 4
Ideal 5
IF 8
VS2 4
I1 1
VVS2 6
VVS1 7
VS1 5
SI2 2
SI1 3
1.2
1
0.8
0.6

Color

```
pheatmap(df,
  labels_row = paste(rownames(df), 1:5),
  labels_col = paste(colnames(df), 1:8), # overlay
  main = "Average Price by Cut and Clarity",
  cluster_rows = FALSE,
  clustering_distance_rows = "correlation",
  color = colorRampPalette(c("white", "purple"))(100)
)
```

Color


Average Price by Cut and Clarity
Fair 1
Good 2
Very Good 3
Premium 4
Ideal 5
IF 8
VS2 4
I1 1
VS1 5
SI2 2
SI1 3
VVS2 6
VVS1 7
1.2
1
0.8
0.6

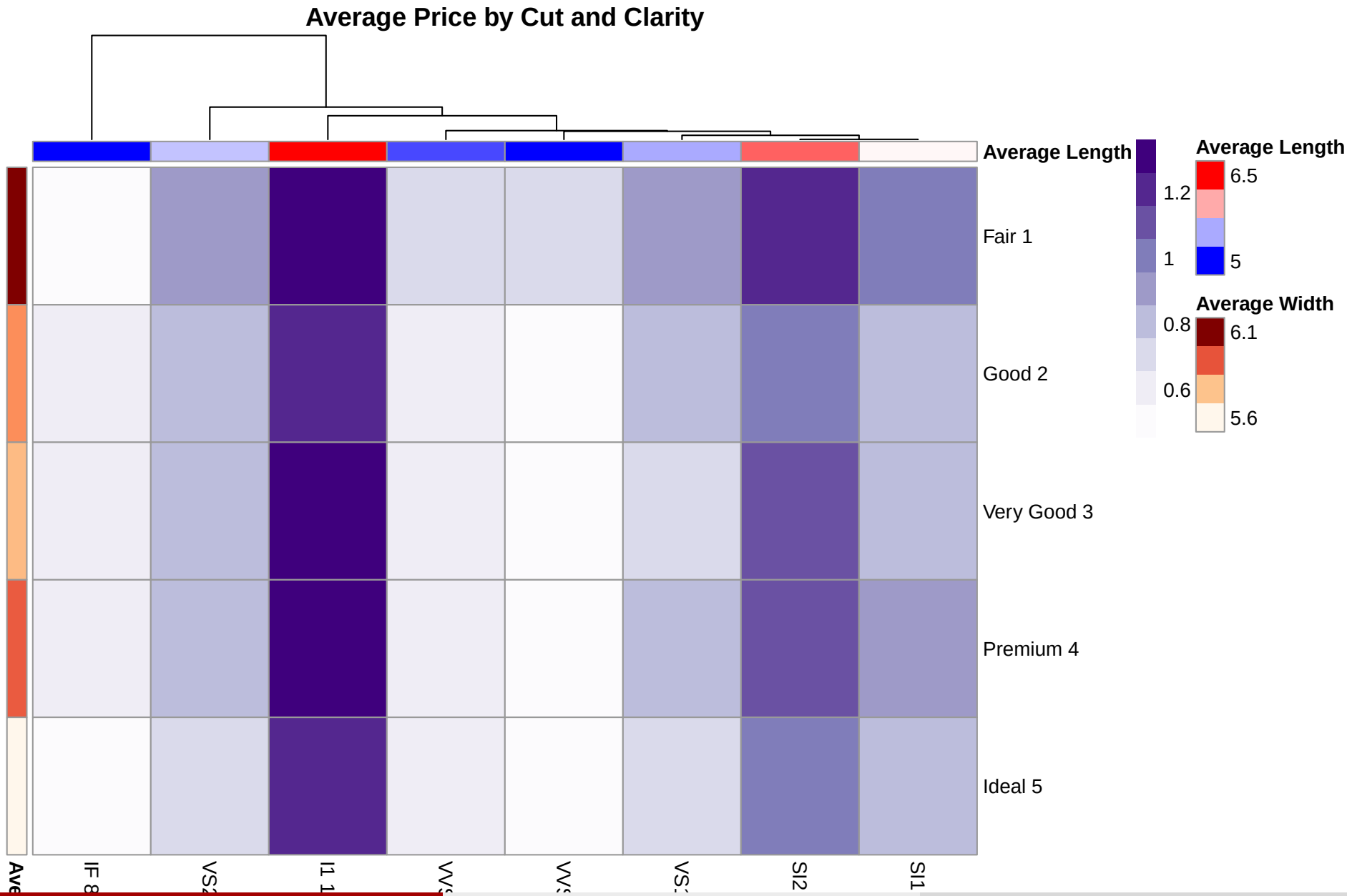
Make Annotations

```
# Annotations
ann1 <- diamonds %>%
  group_by(clarity) %>%
  summarise(avg_len =  mean(x)) %>%
  column_to_rownames("clarity")
colnames(ann1) <- "Average Length"

ann2 <- diamonds %>%
  group_by(cut) %>%
  summarise(avg_wid =  mean(y)) %>%
  column_to_rownames("cut")
colnames(ann2) <- "Average Width"

# Annotations colors
ann_colors <- list(
  `Average Length` = c("blue",'white',"red"),
  `Average Width` = c(brewer.pal(9, "OrRd"))
)
```

Annotations

```
pheatmap(df,
  labels_row = paste(rownames(df), 1:5),
  labels_col = paste(colnames(df), 1:8), # overlay
  main = "Average Price by Cut and Clarity",
  cluster_rows = FALSE,
  clustering_distance_cols = "correlation",
  color = brewer.pal(9, "Purples"), # brewer.pal.info
  annotation_col = ann1,
  annotation_row = ann2,
  annotation_colors = ann_colors
)
```

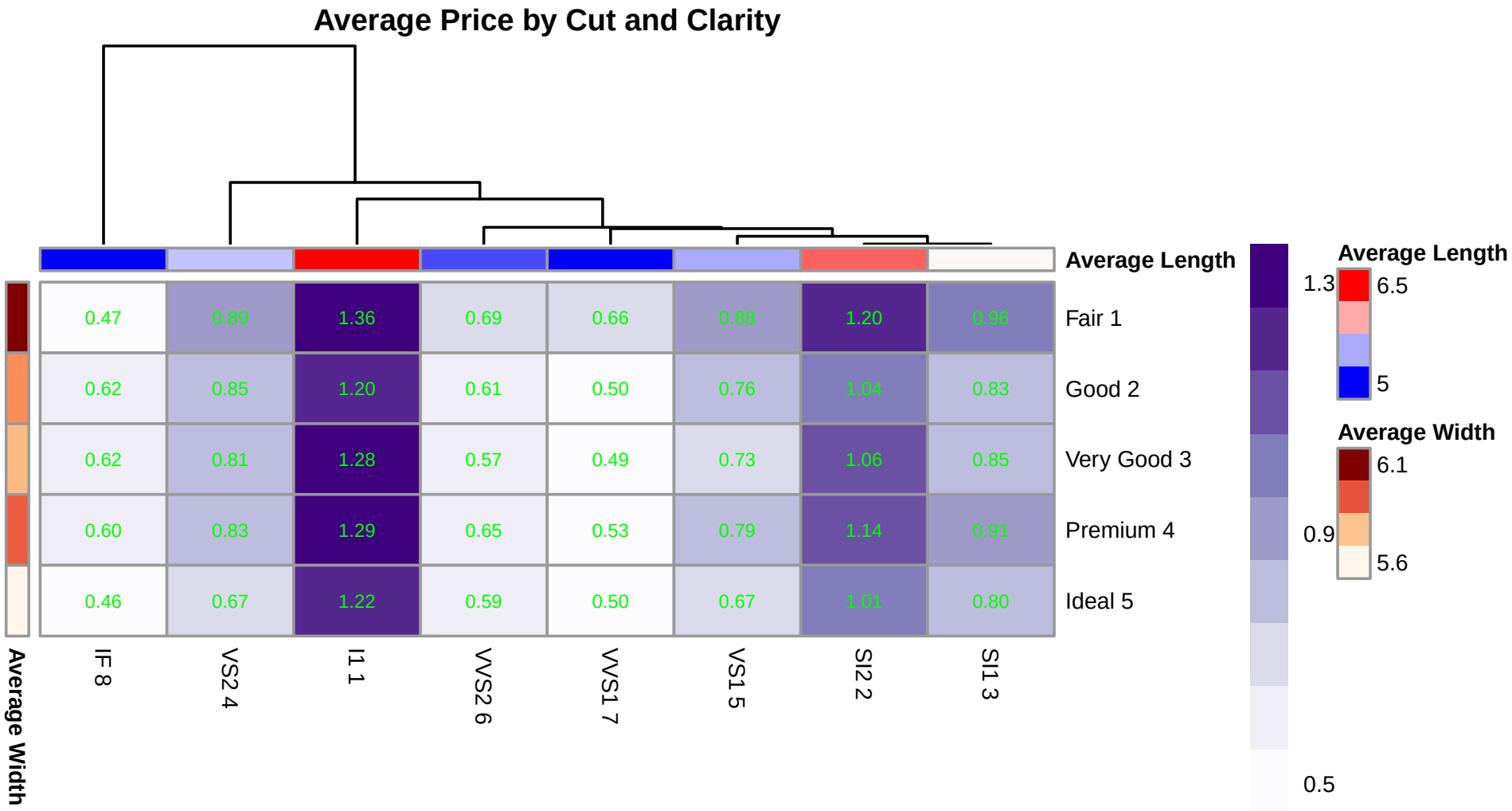
Annotations



Other

```
pheatmap(df,
  labels_row = paste(rownames(df), 1:5),
  labels_col = paste(colnames(df), 1:8), # overlay
  main = "Average Price by Cut and Clarity",
  cluster_rows = FALSE,
  cluster_cols = TRUE,
  clustering_distance_cols = "correlation",
  color = brewer.pal(9, "Purples"), # brewer.pal.info
  annotation_col = ann1,
  annotation_row = ann2,
  annotation_colors = ann_colors,
  legend_breaks = c(.5, .9, 1.3),
  display_numbers = TRUE,
  number_color = "green",
  fontsize = 6,
)
```

Other


Average Price by Cut and Clarity
Average Length
0.47 0.89 1.36 0.69 0.66 0.88 1.20 0.96 Fair 1
0.62 0.85 1.20 0.61 0.50 0.76 1.04 0.83 Good 2
0.62 0.81 1.28 0.57 0.49 0.73 1.06 0.85 Very Good 3
0.60 0.83 1.29 0.65 0.53 0.79 1.14 0.91 Premium 4
0.46 0.67 1.22 0.59 0.50 0.67 1.01 0.80 Ideal 5
IF 8
VS2 4
I1 1
VVS2 6
VVS1 7
VS1 5
SI2 2
SI1 3
Average Width
Average Length
1.3
6.5
5
Average Width
6.1
0.9
5.6
0.5

NYC13Flights

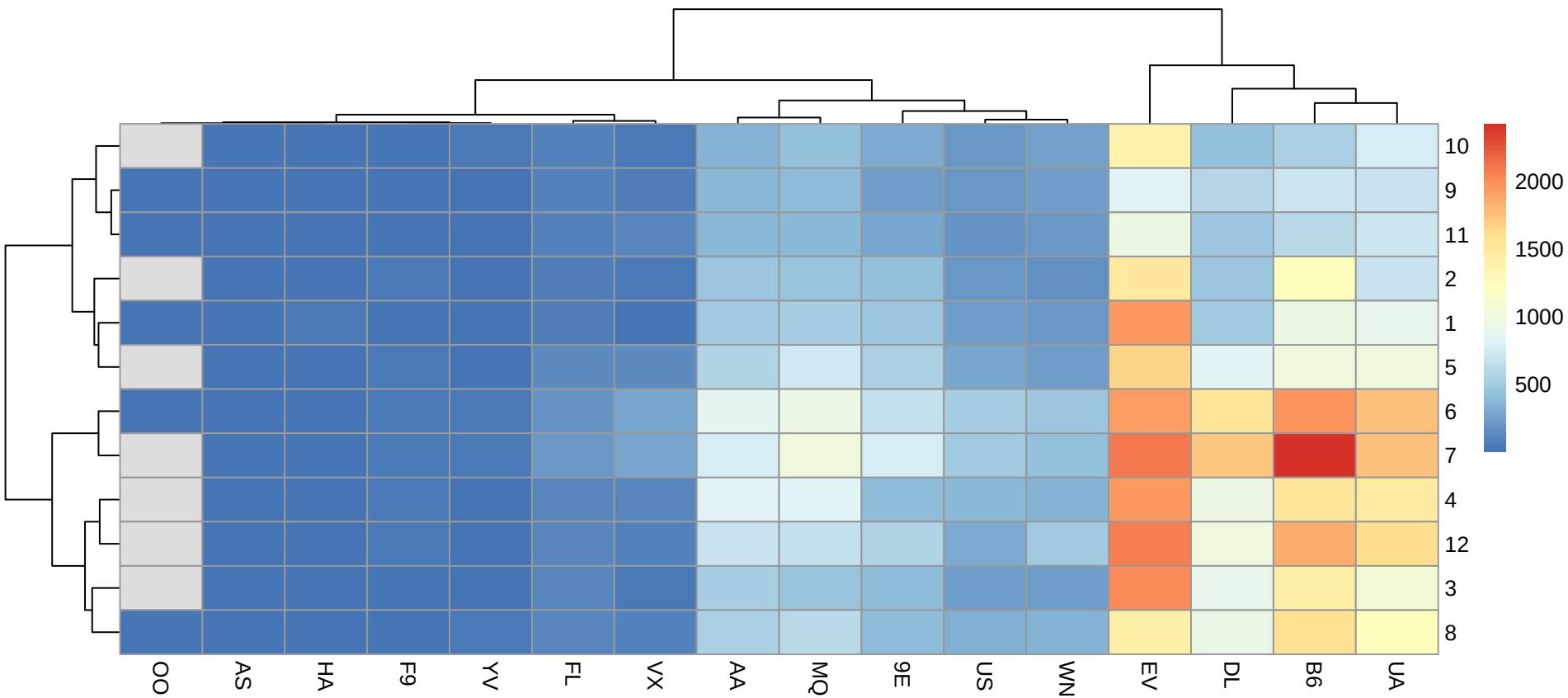
We want to plot the total number of hours flights were late arrivals each month, broken up by each carrier.

```
flights2 <- flights %>%
  filter(arr_delay > 0) %>%
  group_by(month, carrier) %>%
  summarize(total_delay = sum(arr_delay / 60)) %>%
  spread(carrier, total_delay) %>%
  column_to_rownames("month")
```

NYC13Flights

We can now plot it.

```
pheatmap(flights2)
```



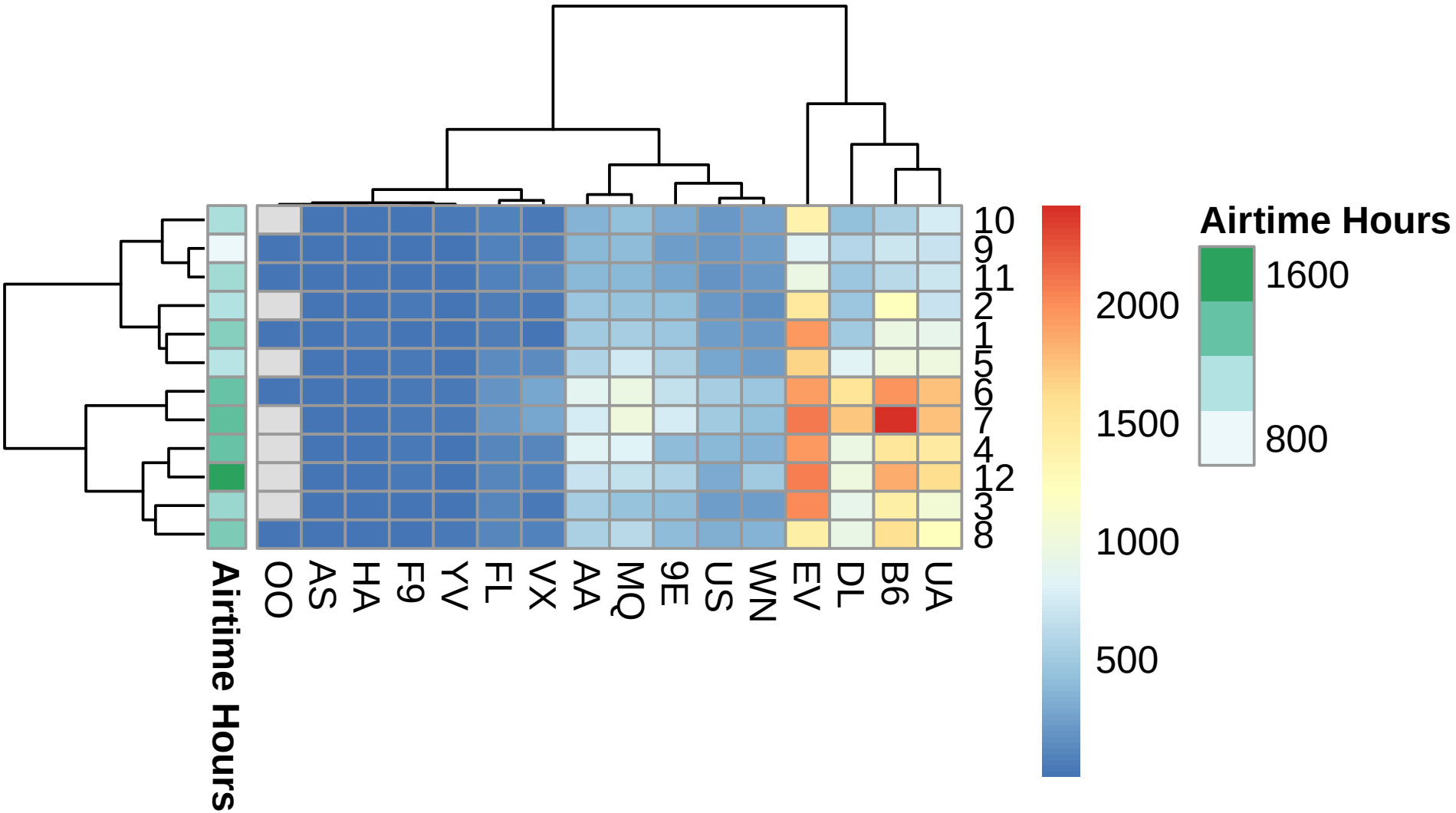
NYC13Flights

Additionally, we would like to overlay the total airtime (in days) for each month. We can do this like as well.

```
airtime <- flights %>%
  filter(arr_delay > 0) %>%
  group_by(month) %>%
  summarize(airtime = sum(air_time / (60 * 24))) %>%
  column_to_rownames("month")
colnames(airtime) <- "Airtime Hours"
```

```
pheatmap(flights2, annotation_row = airtime)
```

NYC13Flights



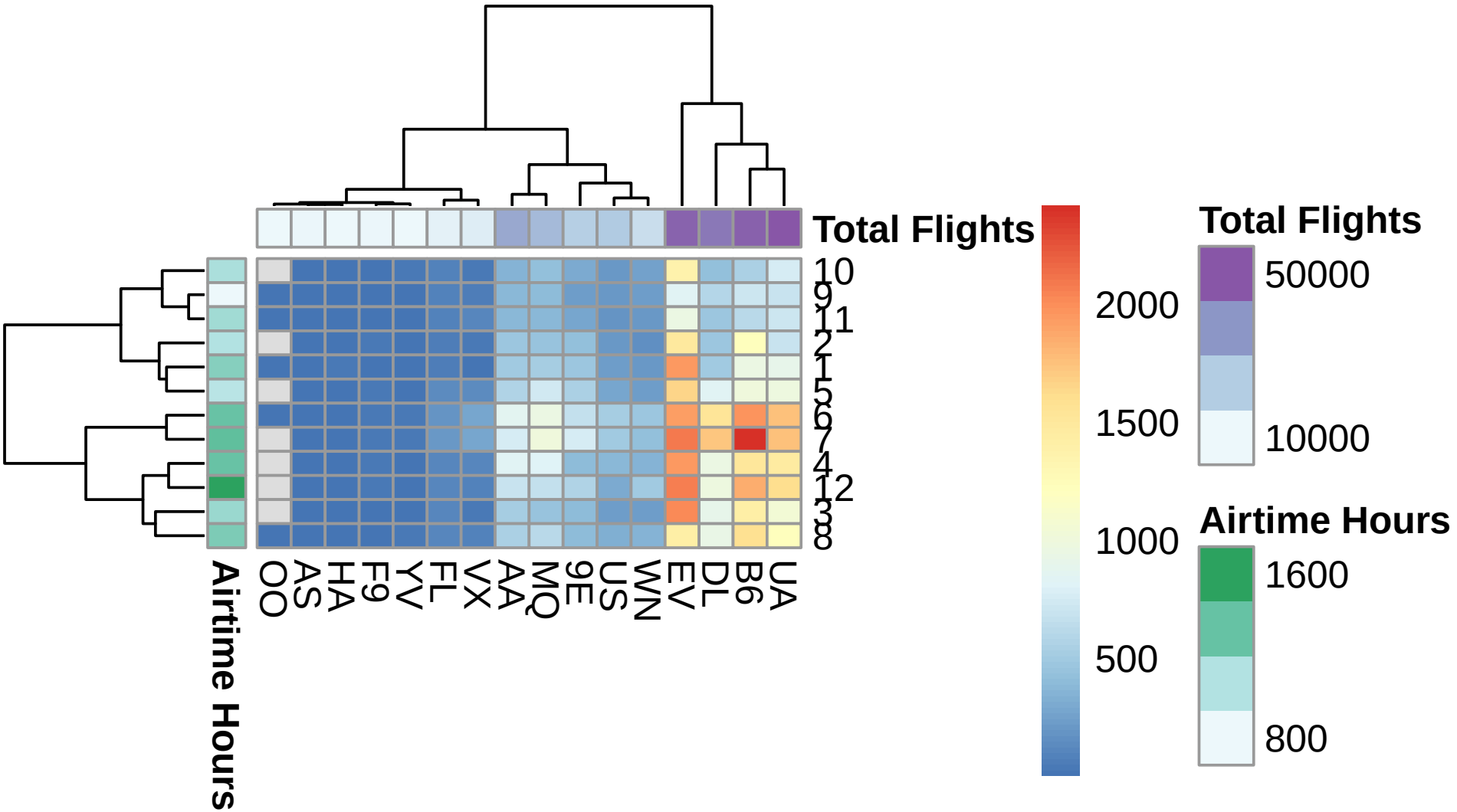
NYC13Flights

We also know certain airlines are preferred, so we would like to see the total number of flights by airline.

```
flightcounts <- flights %>%
  group_by(carrier) %>%
  summarize(total_flights = n()) %>%
  column_to_rownames("carrier")

colnames(flightcounts) <- "Total Flights"
```

```
pheatmap(
  flights2,
  annotation_row = airtime,
  annotation_col = flightcounts
)
```

NYC13Flights

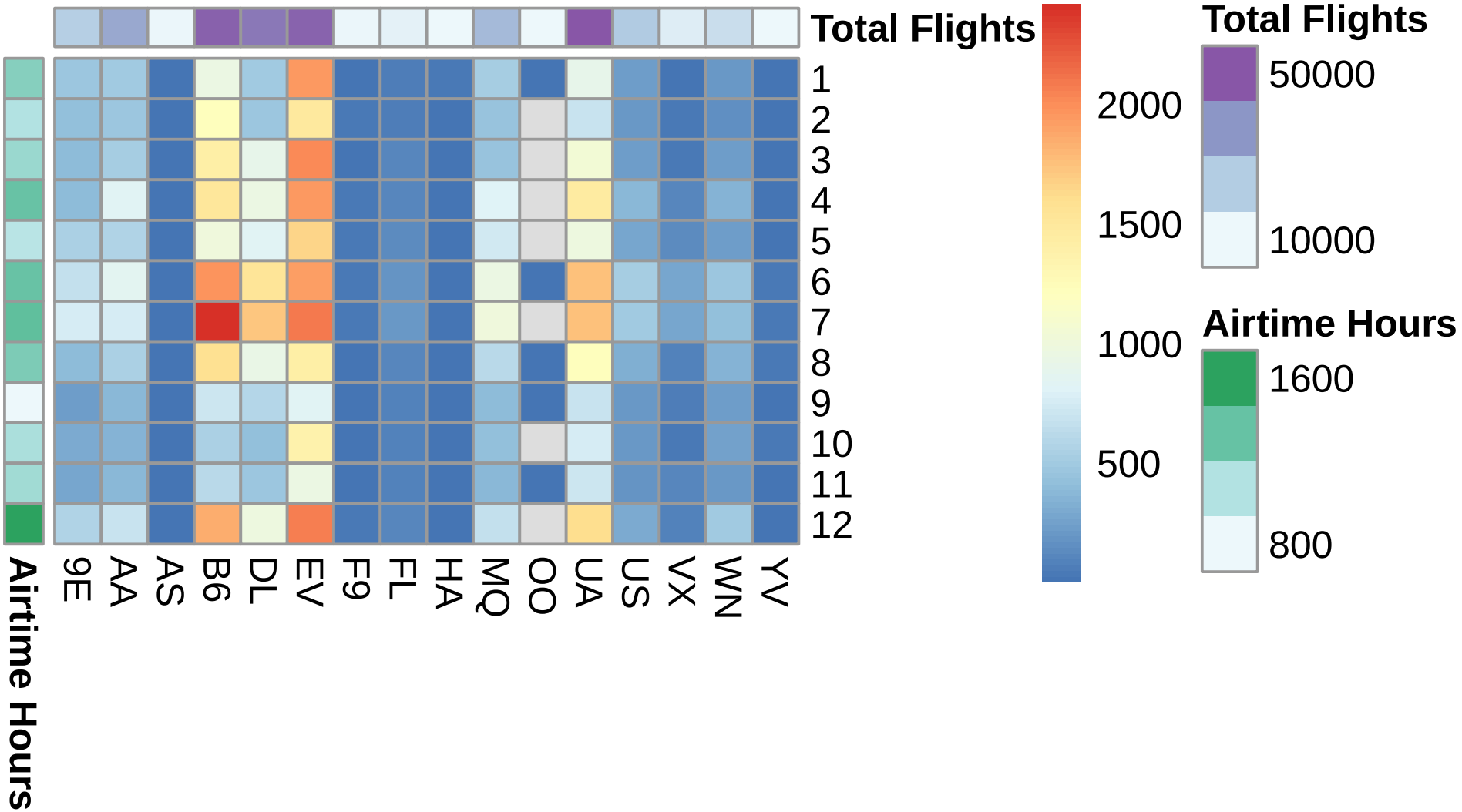


NYC13Flights

If we're not interested in the dendrogram, we can remove it easily.

```
pheatmap(flights2,
  annotation_row = airtime,
  annotation_col = flightcounts,
  cluster_row = FALSE,
  cluster_cols = FALSE
)
```

NYC13Flights

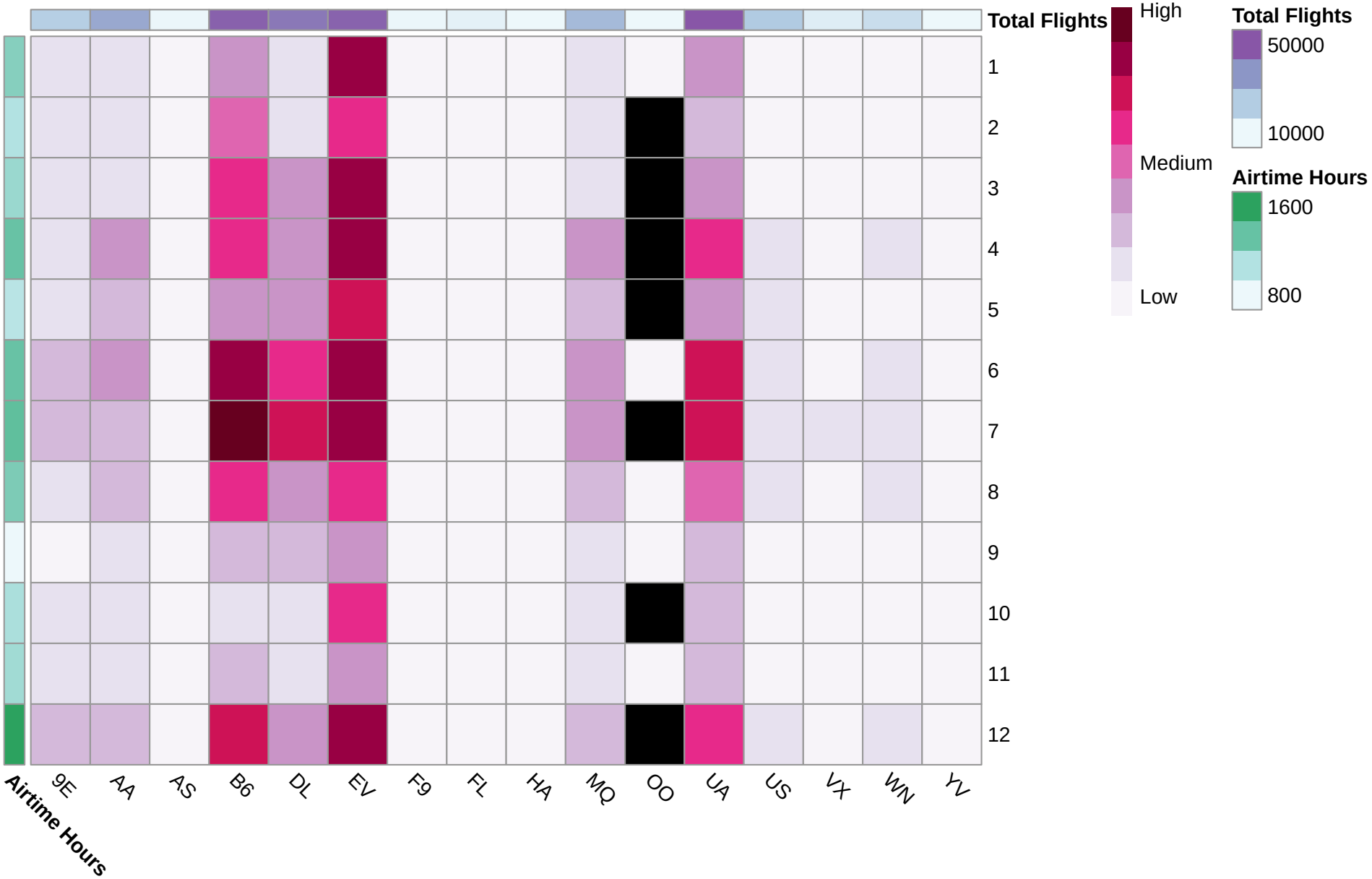


NYC13Flights

Finally, we can clean up our main legend, change the color scheme, and very clearly show missing values for our final figure.

```
pheatmap(flights2,
  annotation_row = airtime,
  annotation_col = flightcounts,
  cluster_row = FALSE,
  cluster_cols = FALSE,
  legend_breaks = c(150, 1200, 2400),
  legend_labels = c("Low", "Medium", "High"),
  color = RColorBrewer::brewer.pal(9, "PuRd"),
  na_col = "black",
  angle_col = 315
)
```

NYC13Flights



Wrapping up

- Heatmaps are applicable to both continuous and discrete variables.
- When working with discrete variables, distinguishing between similar colors can be challenging.
- Use no more than nine unique colors, a divergent color scale, or a sequential colorscale.
- Use Pheatmap to supercharge your heatmaps

References

Bedre, Renesh. 2022. "Pheatmap: Create Annotated Heatmaps in r (Detailed Guide)." *RS Blog*, October. http://www.reneshbedre.com/blog/heatmap-with-pheatmap-package-r.html.

Kayvan Jalali, Carlos Puerta. n.d. "607-Presentation." https://github.com/psweet/607-presentation.

Kolde, Raivo. n.d. "Package 'Pheatmap'." https://cran.r-project.org/web/packages/pheatmap/pheatmap.pdf.

"Pheatmap: A Function to Draw Clustered Heatmaps." n.d. https://rdrr.io/bioc/COMPASS/man/pheatmap.html.

Susan Holmes, Wolfgang Huber. n.d. "Modern Statistics for Modern Biology." https://www.huber.embl.de/msmb/03-chap.html#heatmaps.

Wickham, Hadley, Romain François, Lionel Henry, Kirill Müller, and Davis Vaughan. 2023. "Dplyr: A Grammar of Data Manipulation." 2023.