Heatmaps in R

Kayvan Jalali & Carlos Puerta

9/20/2023

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

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.

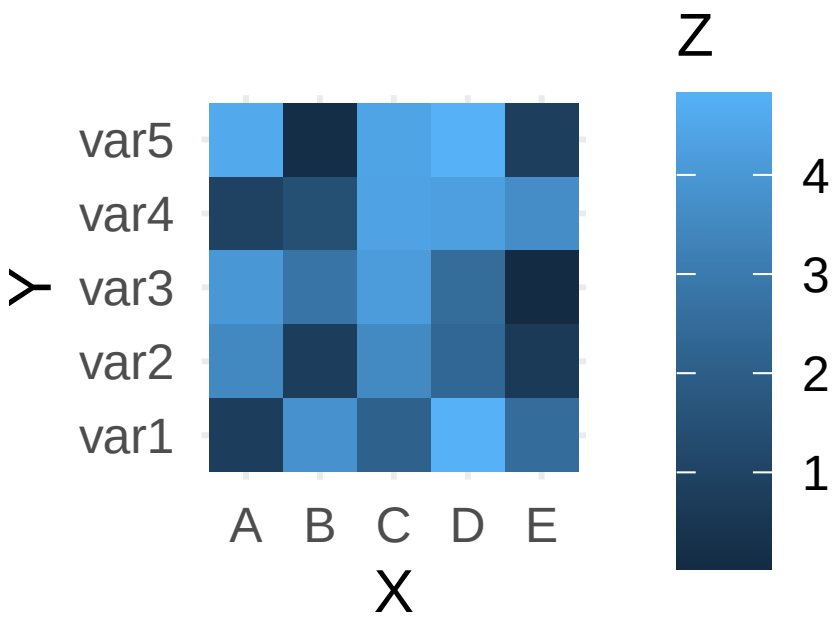
Advantages

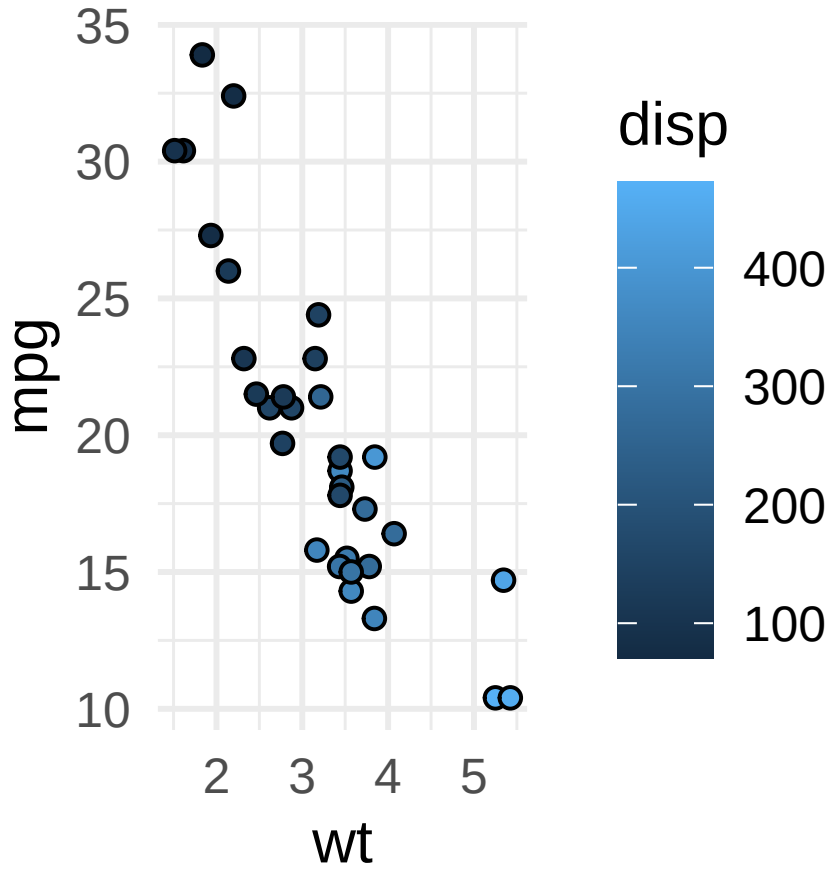
- Adds another variable
- Easier to interpret
- Color is easy to parse

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


35
30
25
20
15
10
mpg
2 3 4 5
wt
disp
400
300
200
100

When to use a heatmap

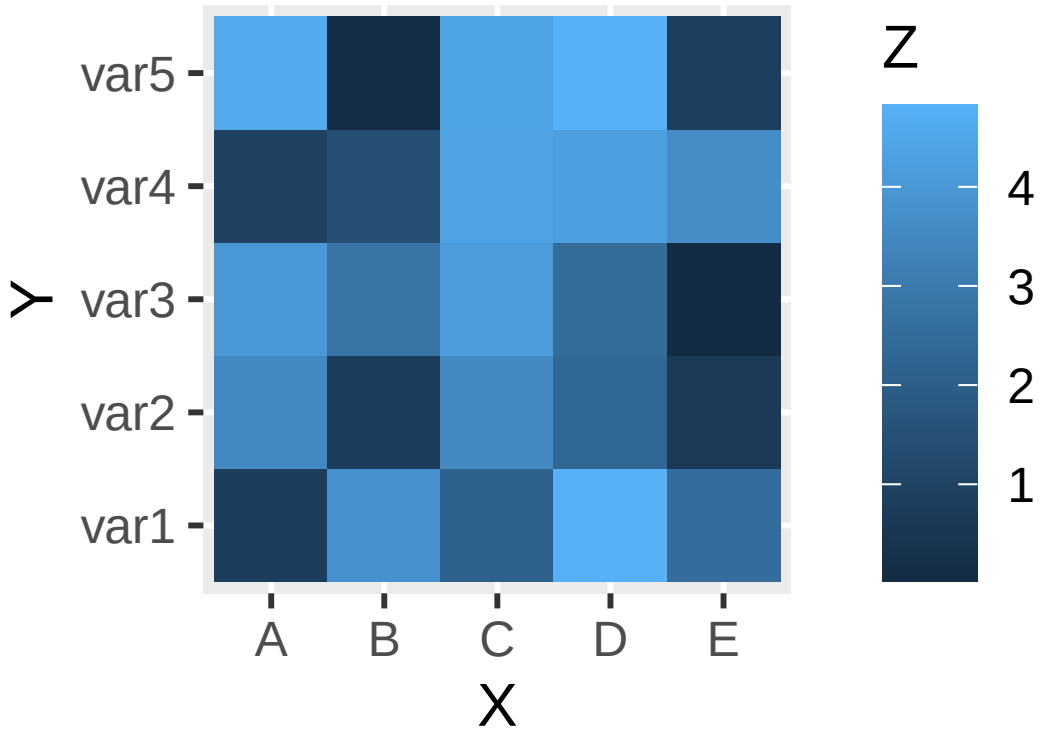
Heatmaps are applicable to both continuous and discrete variables. When working with discrete variables, distinguishing between similar colors can be challenging. To improve visibility, it is recommended to use a color palette of no more than nine colors, such as ROYGBIV, black, and white.

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.

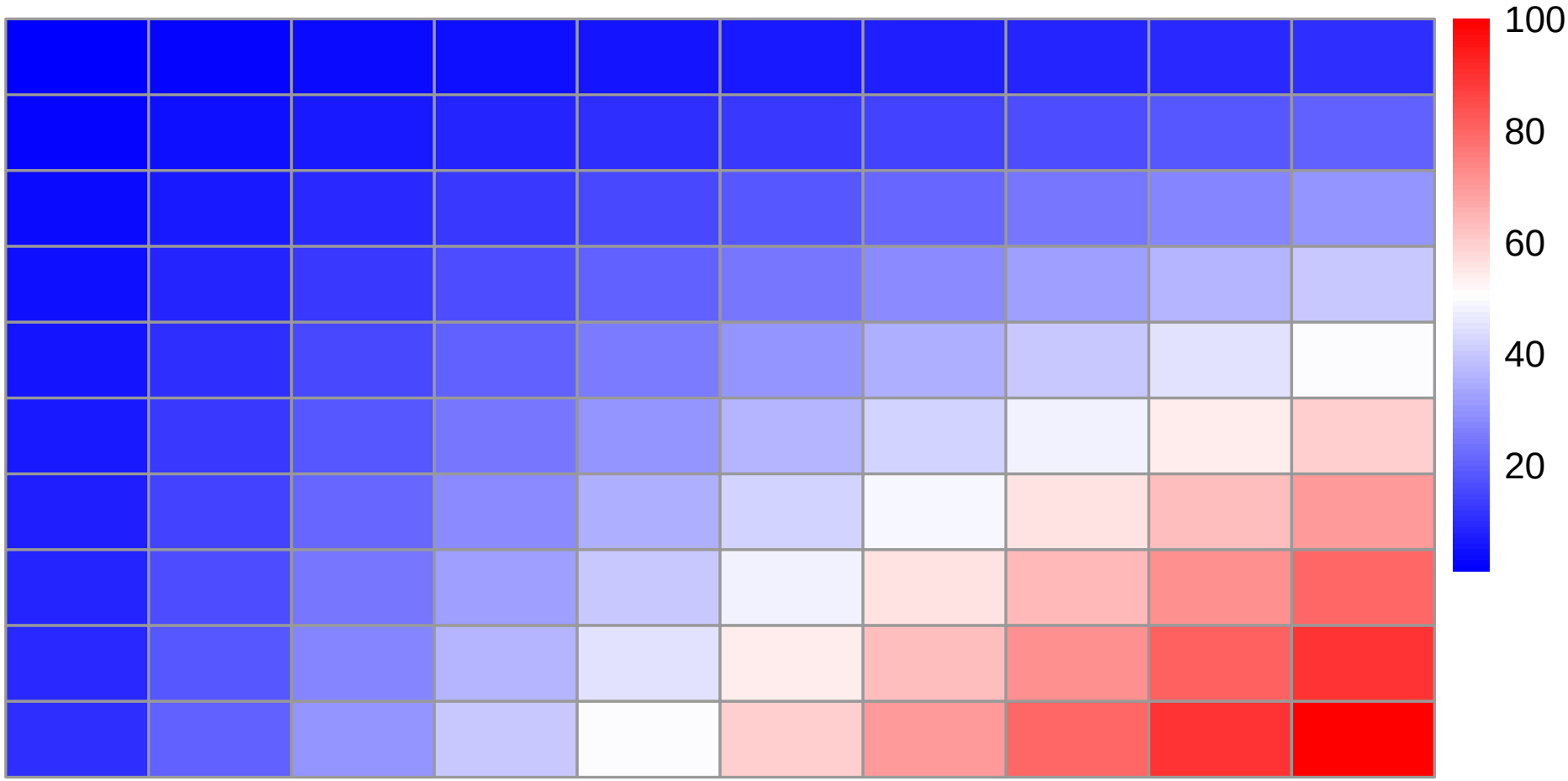
Dendrogram

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.

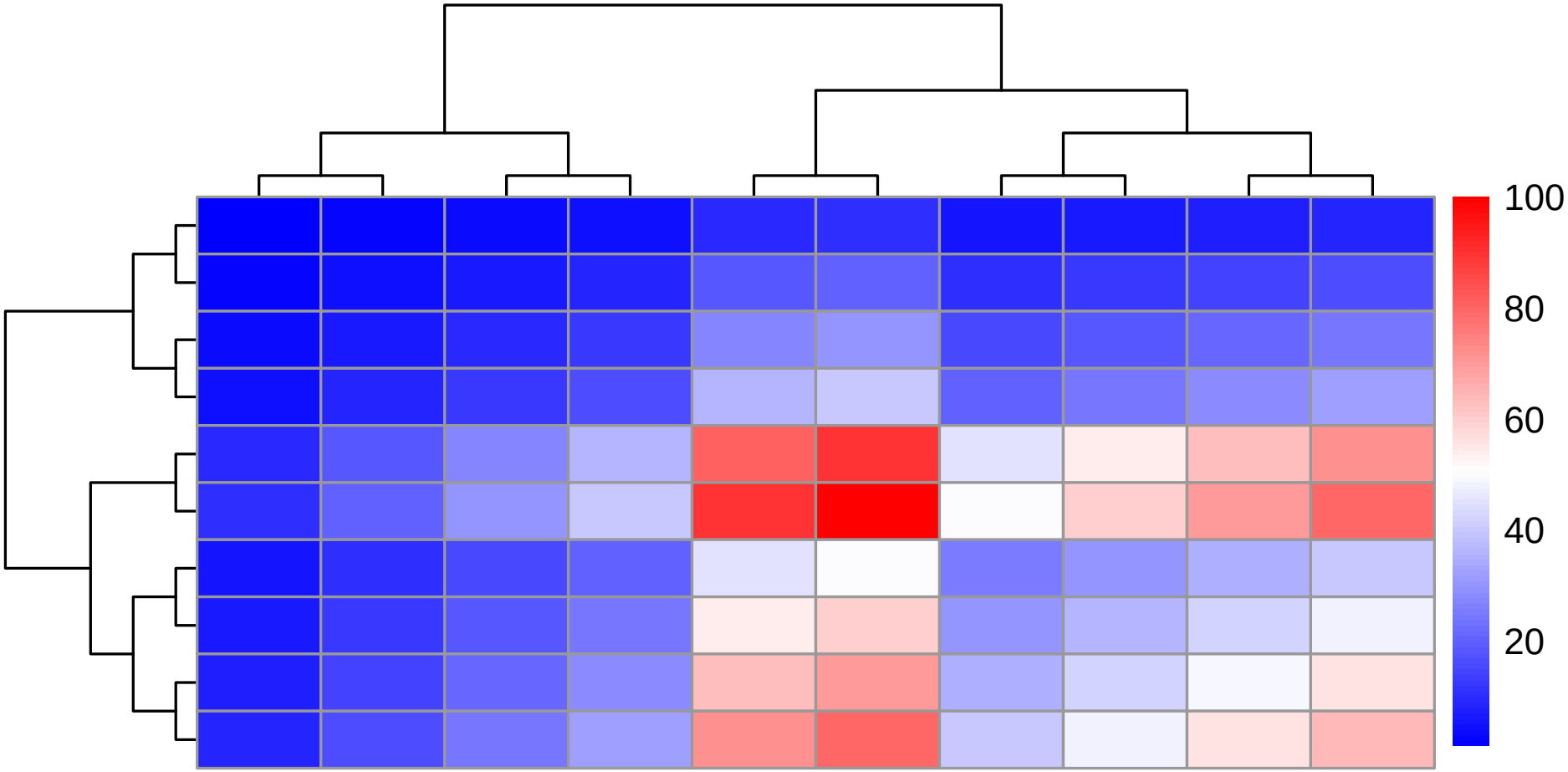
A Bad Tournament

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.

Pheatmap example

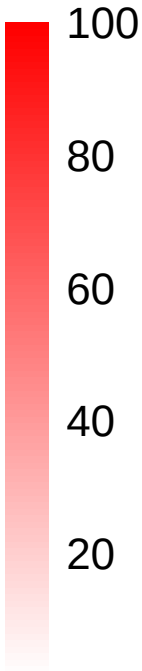


Pheatmap example

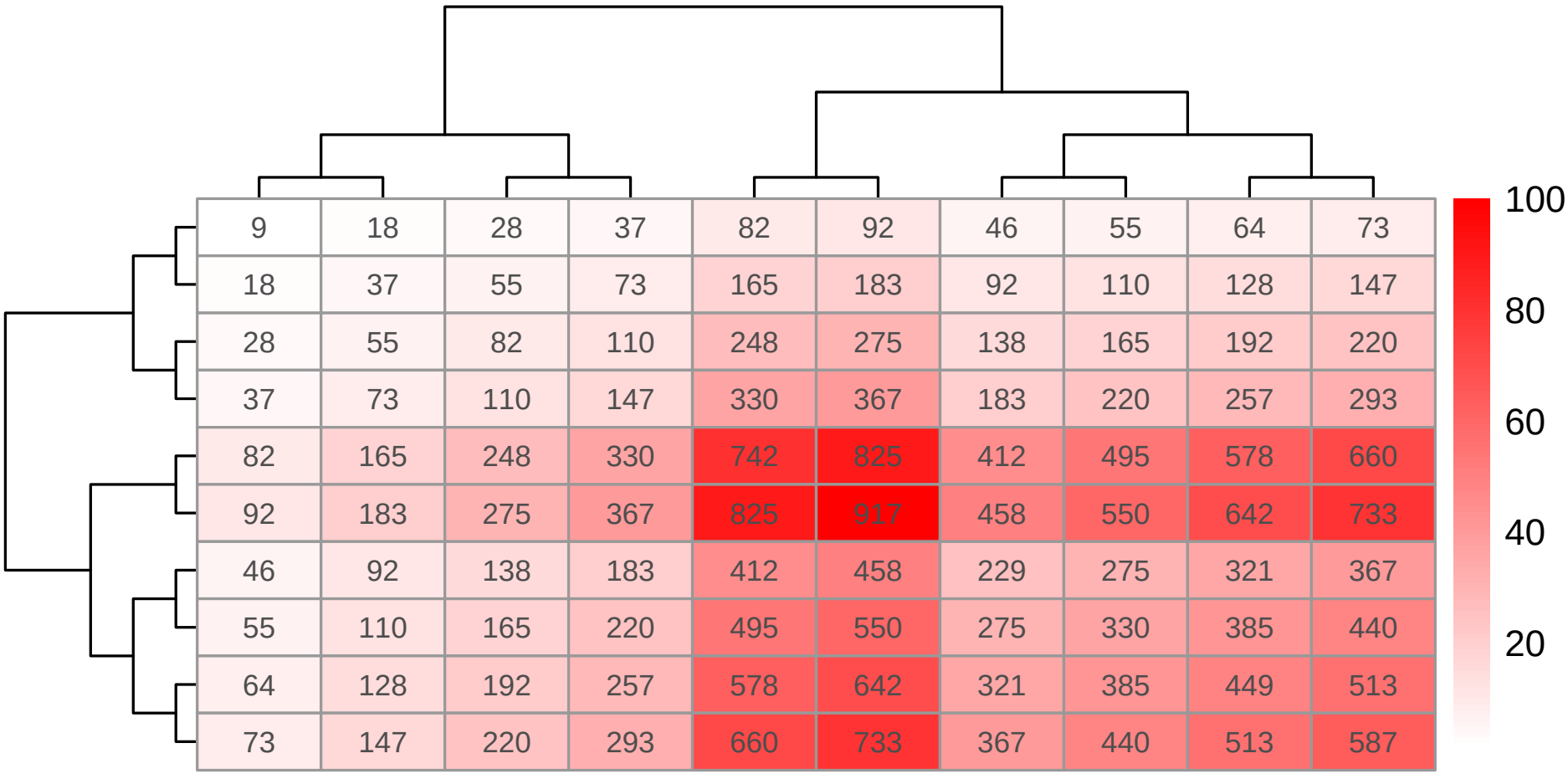


Pheatmap example

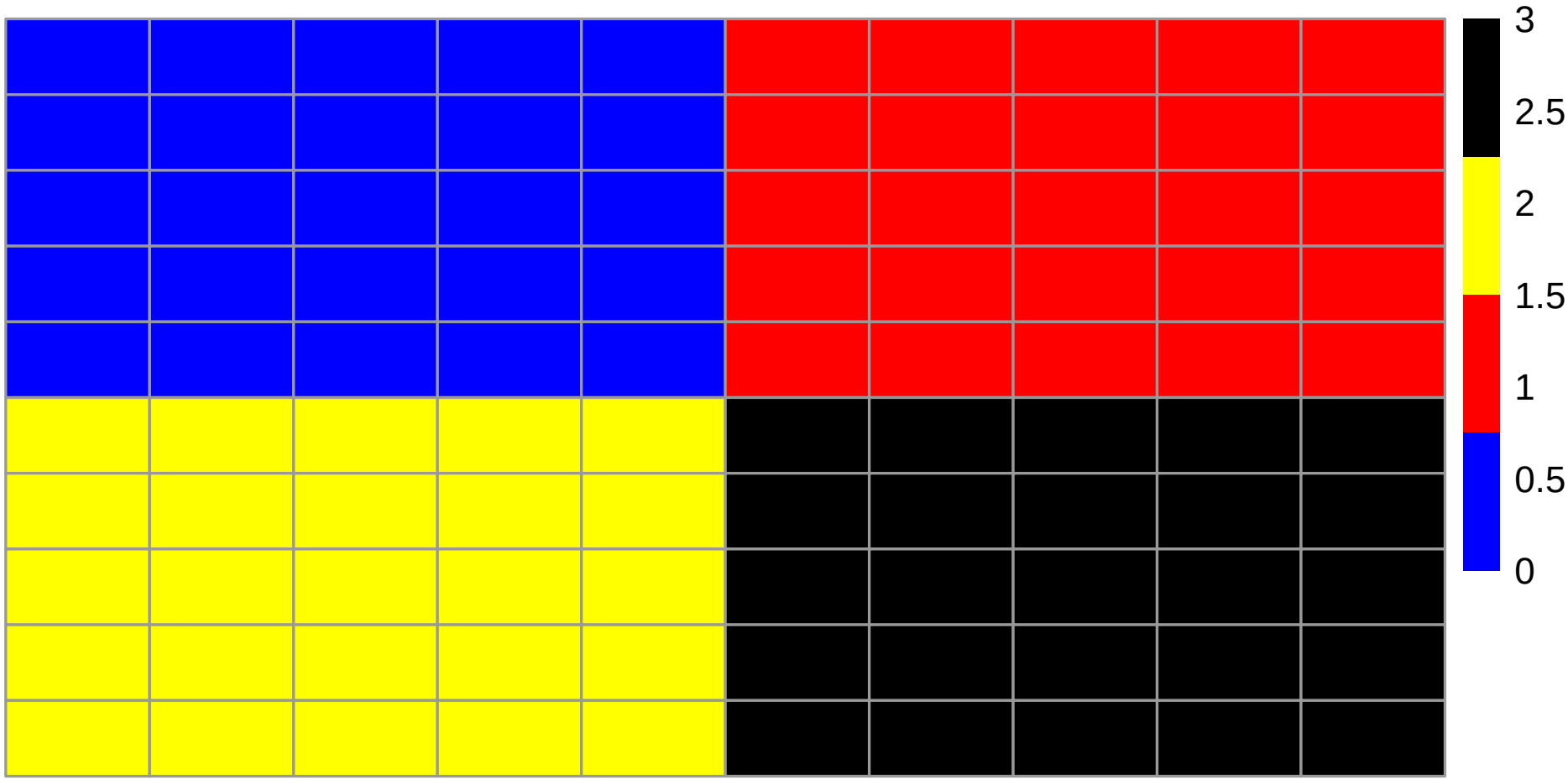
9	18	28	37	46	55	64	73	82	92
18	37	55	73	92	110	128	147	165	183
28	55	82	110	138	165	192	220	248	275
37	73	110	147	183	220	257	293	330	367
46	92	138	183	229	275	321	367	412	458
55	110	165	220	275	330	385	440	495	550
64	128	192	257	321	385	449	513	578	642
73	147	220	293	367	440	513	587	660	733
82	165	248	330	412	495	578	660	742	825
92	183	275	367	458	550	642	733	825	917



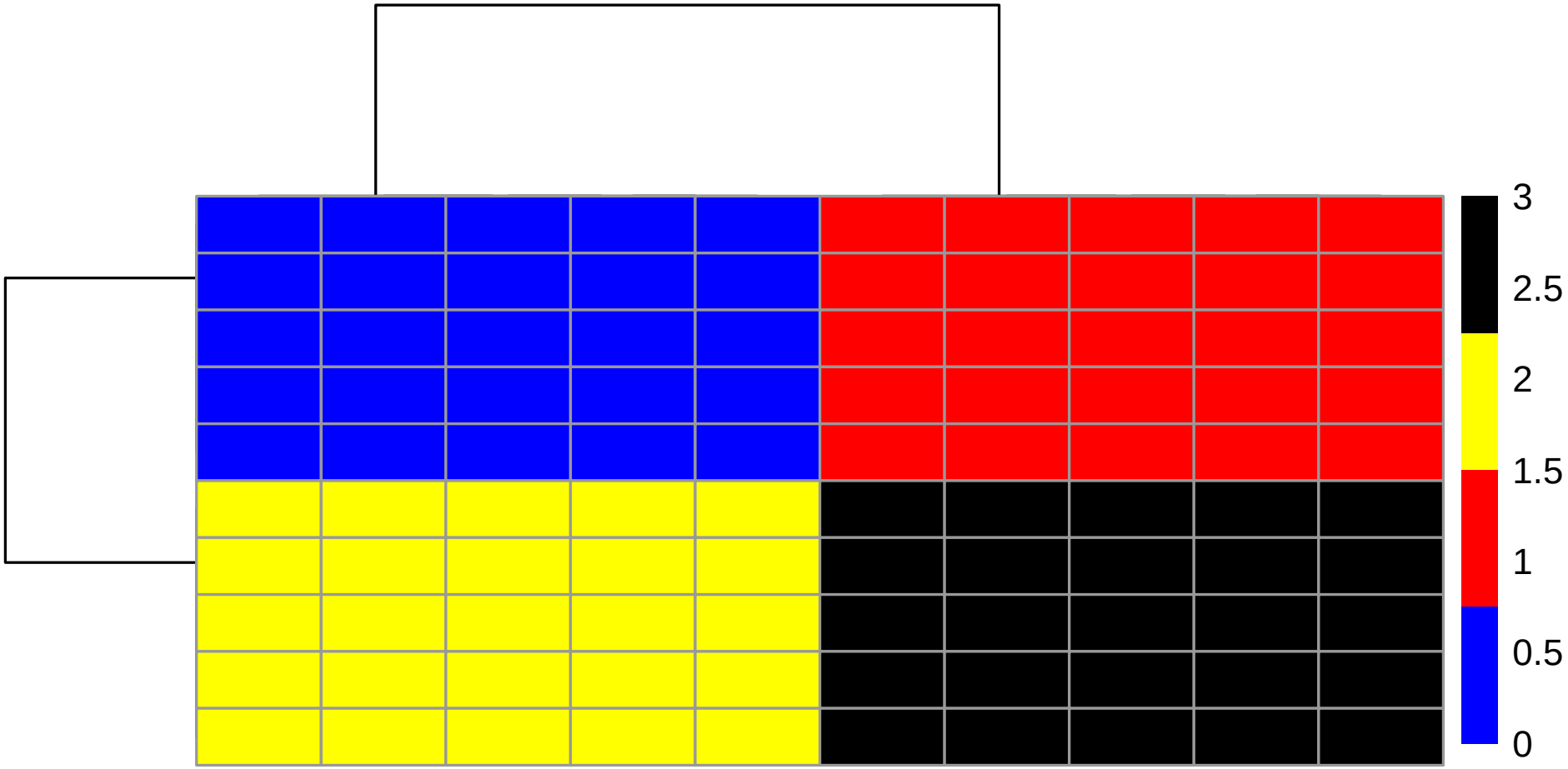
Pheatmap example



Pheatmap example

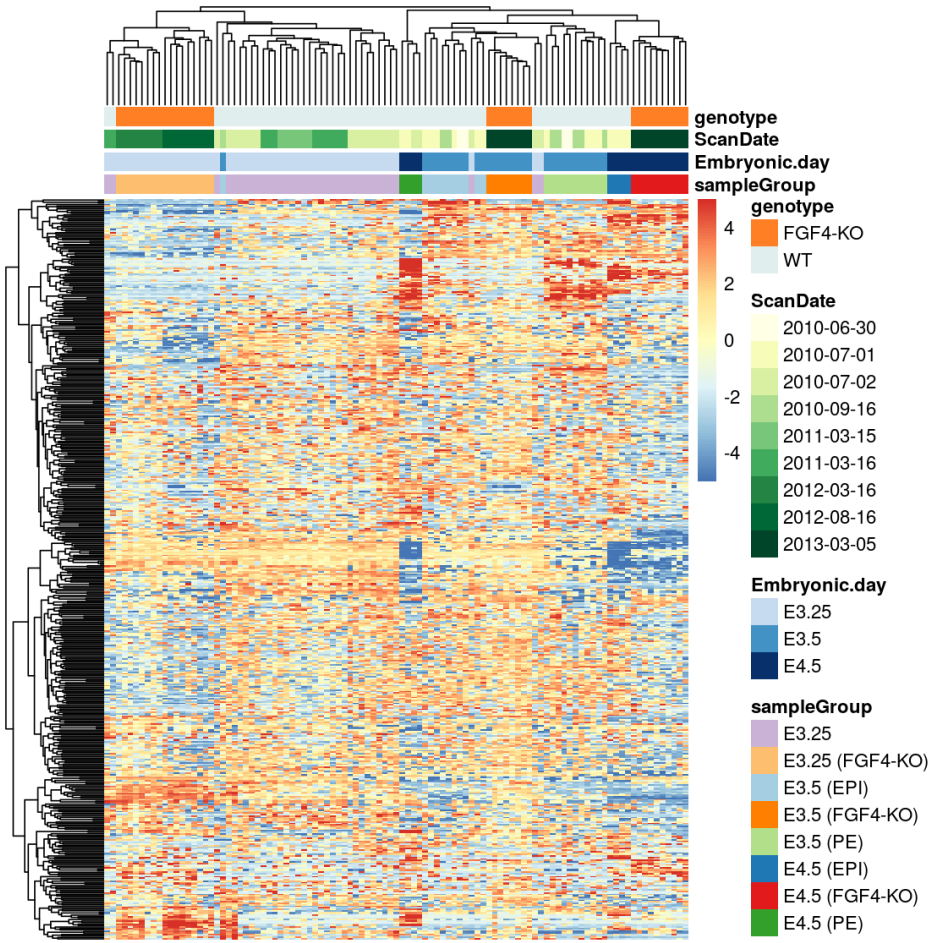


Pheatmap example



More Complex Example

Genes


genotype
ScanDate
Embryonic.day
sampleGroup
4
2
0
-2
-4
genotype
FGF4-KO
WT
ScanDate
2010-06-30
2010-07-01
2010-07-02
2010-09-16
2011-03-15
2011-03-16
2012-03-16
2012-08-16
2013-03-05
Embryonic.day
E3.25
E3.5
E4.5
sampleGroup
E3.25
E3.25 (FGF4-KO)
E3.5 (EPI)
E3.5 (FGF4-KO)
E3.5 (PE)
E4.5 (EPI)
E4.5 (FGF4-KO)
E4.5 (PE)

Code for Genes

```
topGenes = order(rowVars(Biobase::exprs(x)),
                 decreasing = TRUE)[1:500]
rowCenter = function(x) { x - rowMeans(x) }
pheatmap(rowCenter(dfx[topGenes, ]),
  show_rownames = FALSE,
  show_colnames = FALSE,
  breaks = seq(-5, +5, length = 101),
  annotation_col = pData(x)[, c("sampleGroup", "Embryonic.day",
                                "ScanDate", "genotype")],
  annotation_colors = list(
    sampleGroup = groupColor,
    genotype = c(`FGF4-KO` = "chocolate1", `WT` = "azure2"),
    Embryonic.day = setNames(brewer.pal(9, "Blues")[c(3, 6, 9)],
                             c("E3.25", "E3.5", "E4.5")),
    ScanDate = setNames(brewer.pal(nlevels(x$ScanDate), "YlGn"),
                        levels(x$ScanDate))
  )
)
```

Section 3

Applying Heatmaps

MPG

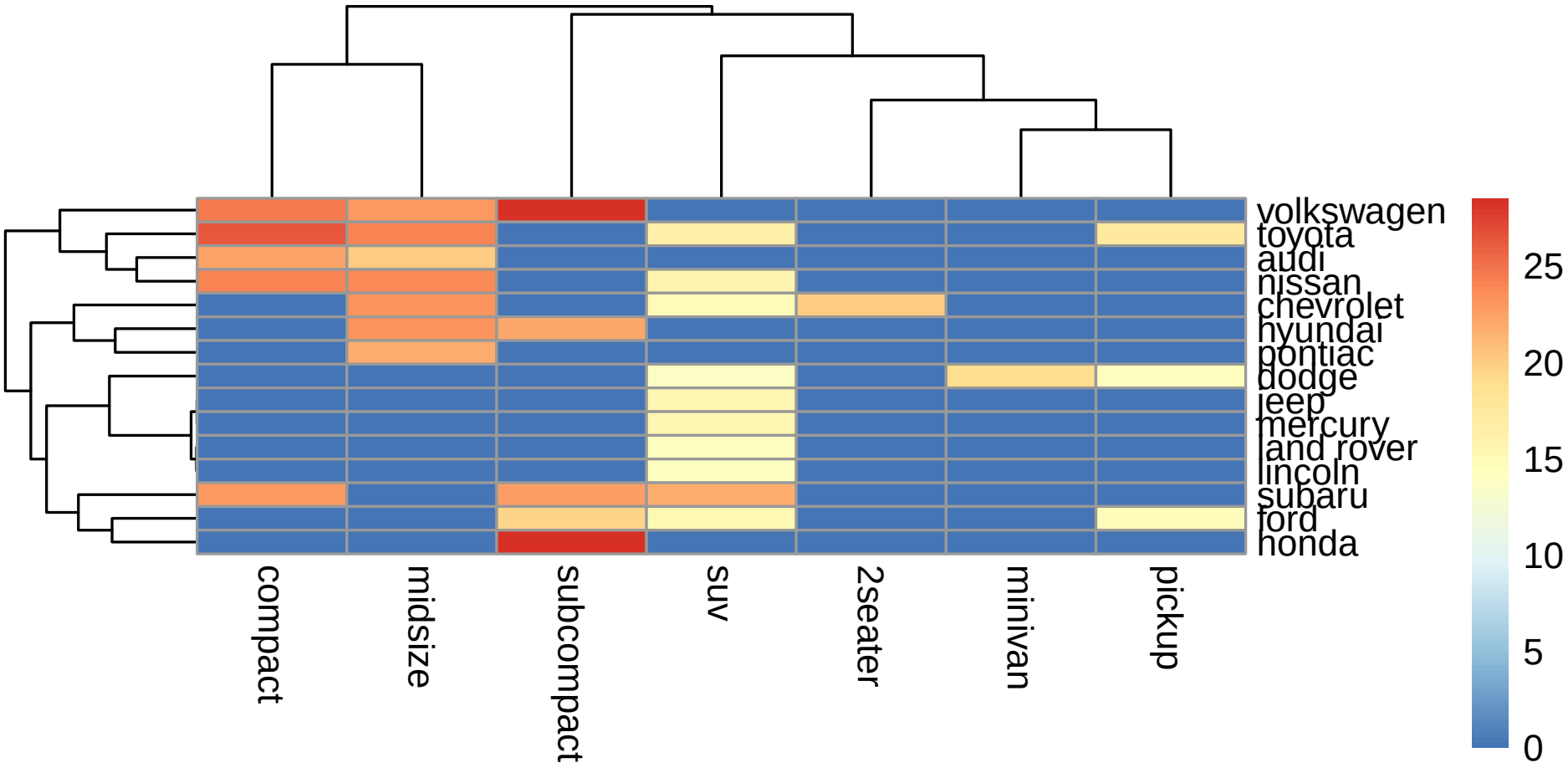
Let's do it with a simple dataset we're familiar with.

```
mpg2 <- mpg %>%
  group_by(manufacturer, class) %>%
  mutate(avg_mpg = (cty + hwy) / 2) %>%
  summarize(avg_mpg = mean(avg_mpg)) %>%
  spread(class, avg_mpg) %>%
  column_to_rownames("manufacturer")
mpg2[is.na(mpg2)] = 0
mpg2[1:5, 1:6]
```

```
##           2seater  compact midsize  minivan   pickup subcompact
## audi          0.0 22.43333    20.0  0.00000  0.00000    0.00000
## chevrolet    20.1  0.00000    23.2  0.00000  0.00000    0.00000
## dodge         0.0  0.00000     0.0 19.09091 14.07895    0.00000
## ford          0.0  0.00000     0.0  0.00000 14.71429   19.55556
## honda         0.0  0.00000     0.0  0.00000  0.00000   28.50000
```

MPG

```
pheatmap(mpg2)
```

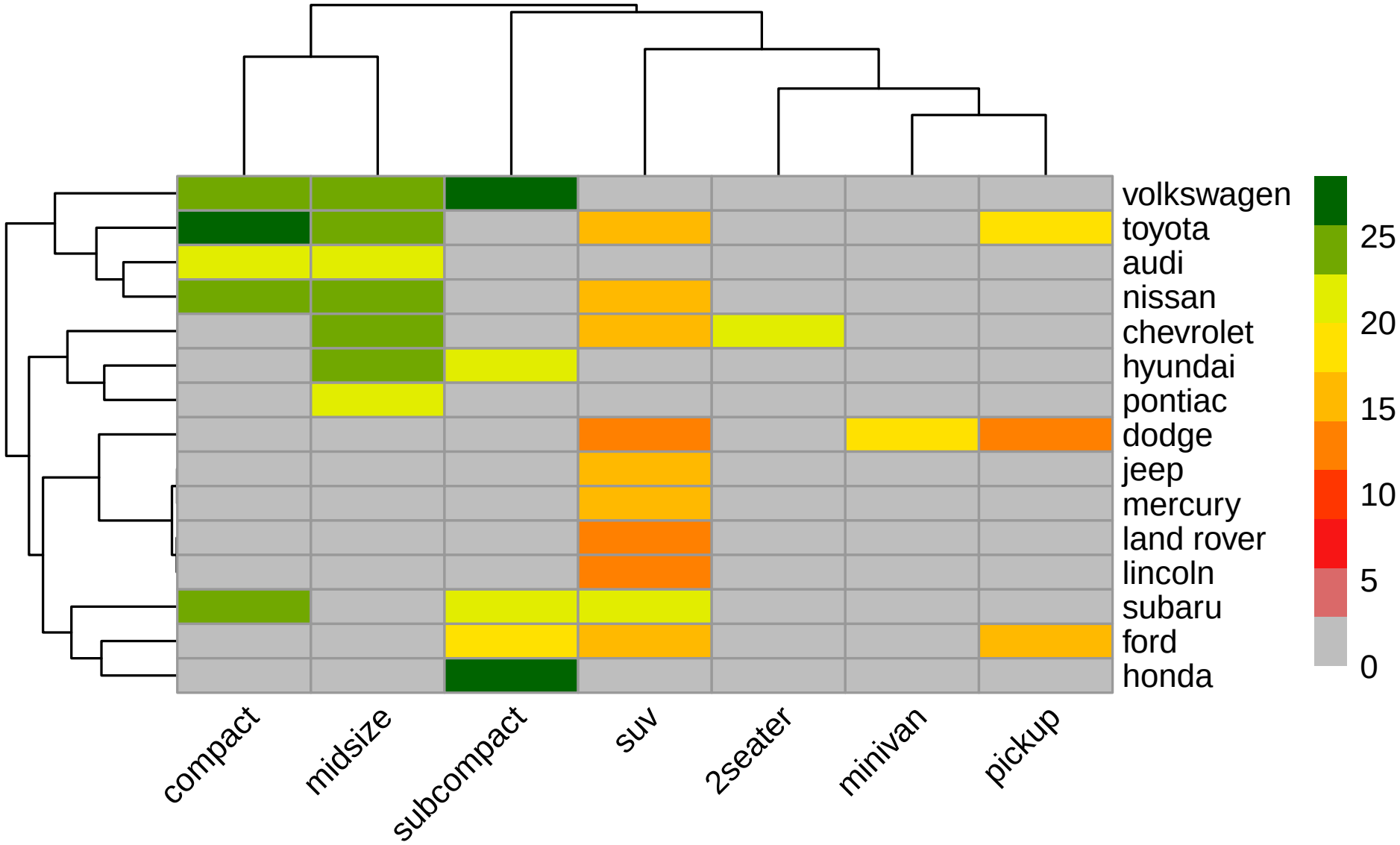


MPG

That didn't look great. Let's clean it up.

```
pheatmap(mpg2,
  angle_col = 45,
  color = colorRampPalette(
    c("gray", "red", "orange", "yellow", "darkgreen")
  )(10),
)
```

MPG


volkswagen
toyota
audi
nissan
chevrolet
hyundai
pontiac
dodge
jeep
mercury
land rover
lincoln
subaru
ford
honda
compact
midsize
subcompact
suv
2seater
minivan
pickup
25
20
15
10
5
0

NYC13Flights

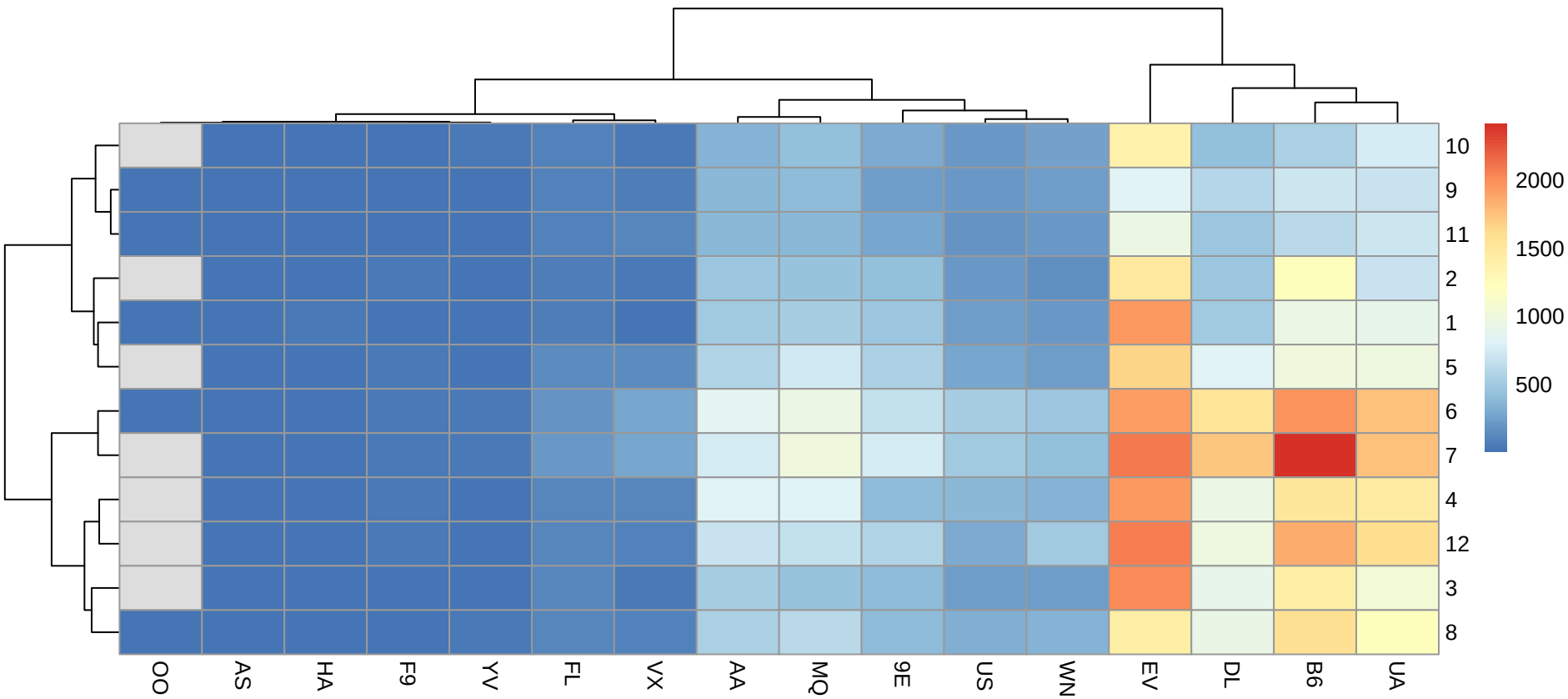
We want to plot the total number of hours flights were late arrivals each month, broken up by each carrier.

```
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)
```



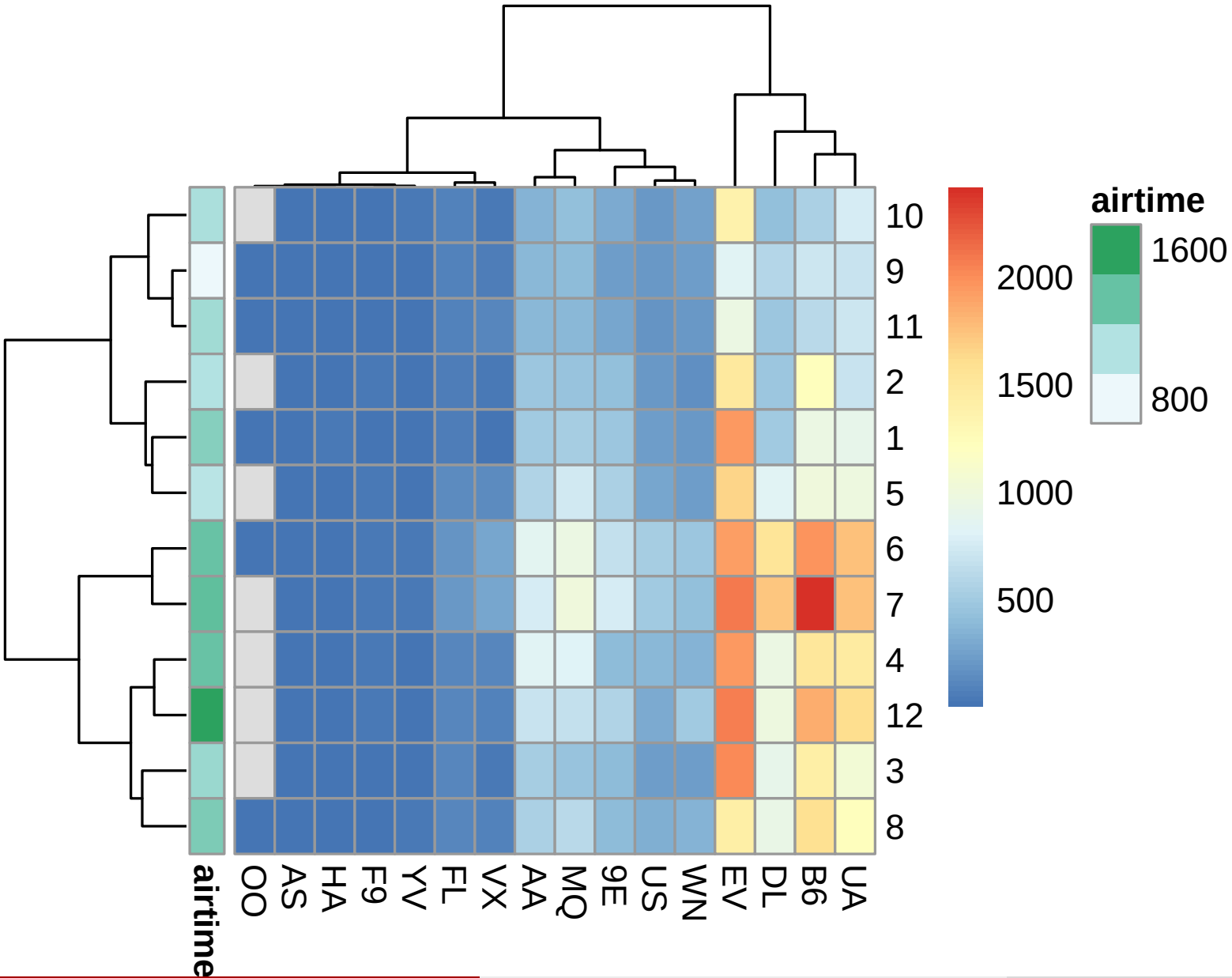
NYC13Flights

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")
```

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

NYC13Flights



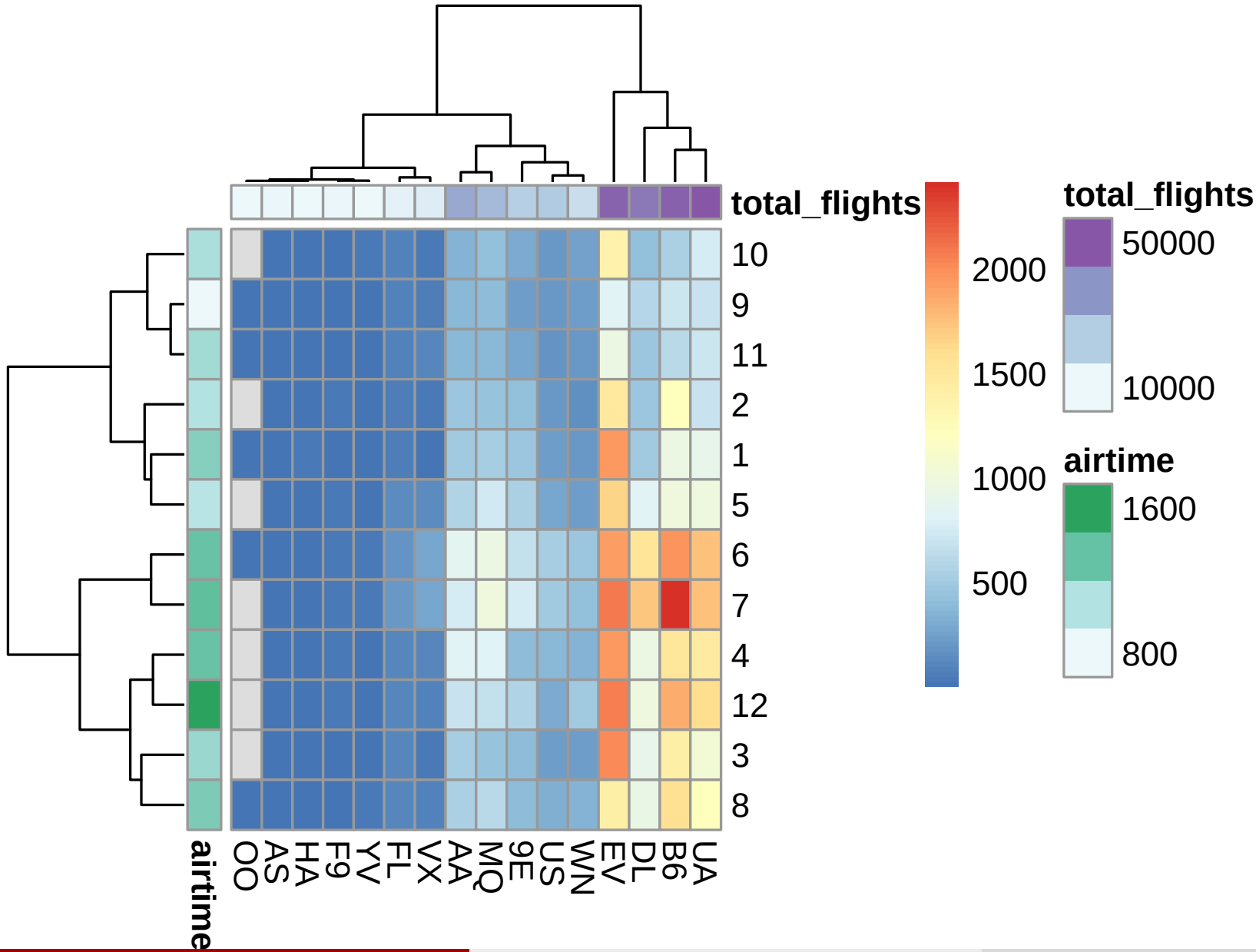
NYC13Flights

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")
```

```
pheatmap(flights2, annotation_row = airtime, annotation_col = flight
```

NYC13Flights

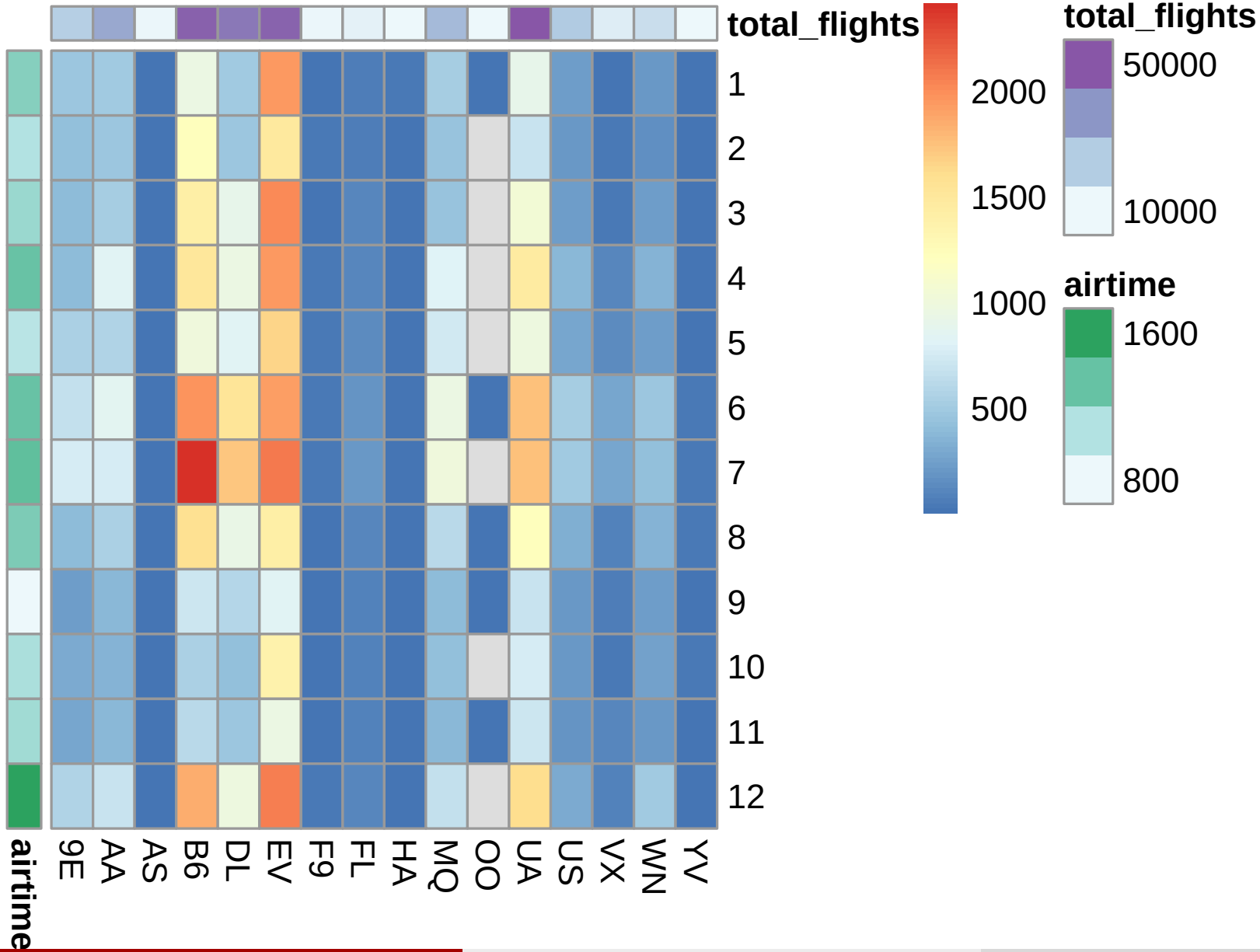


NYC13Flights

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

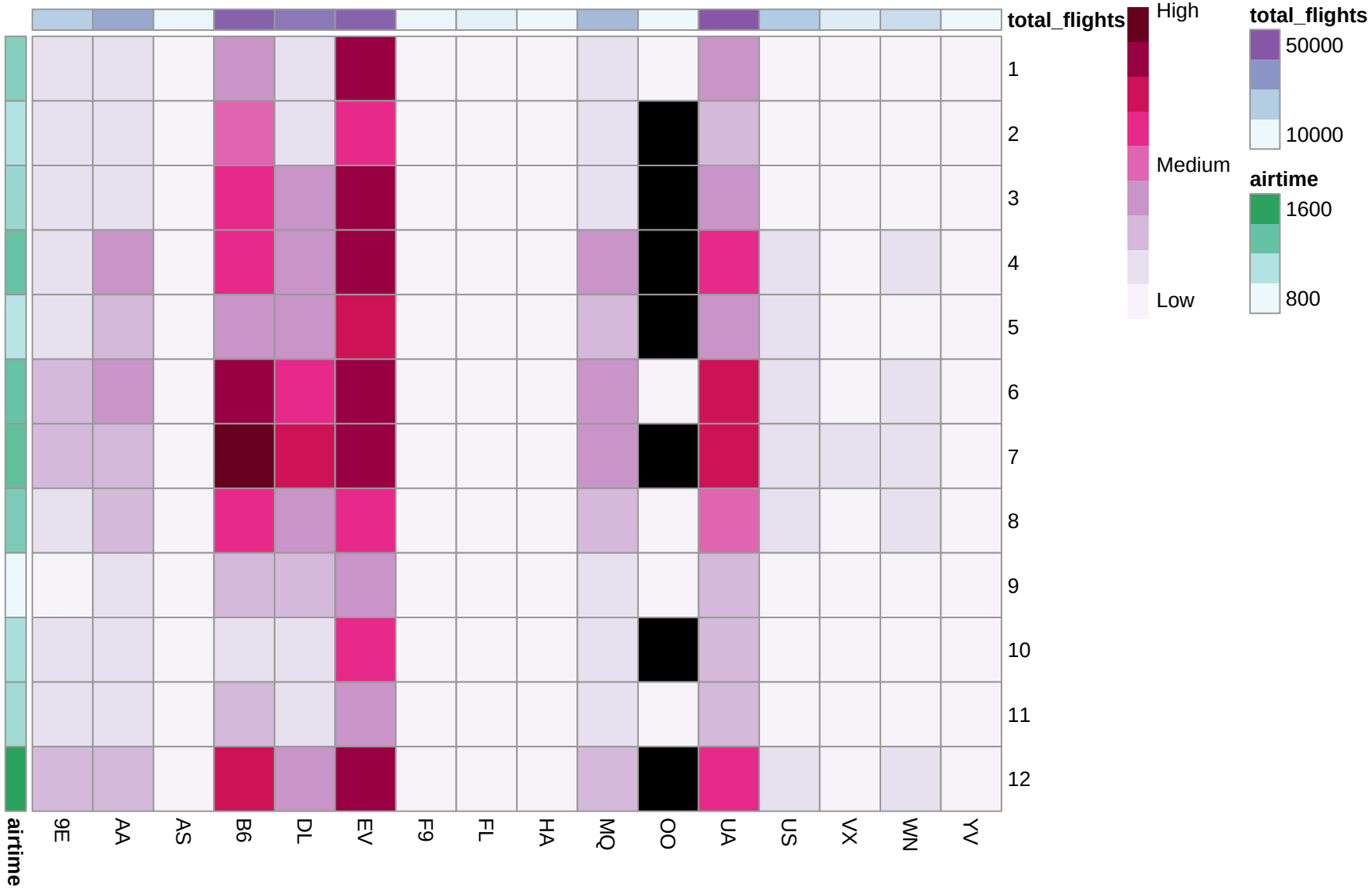


NYC13Flights

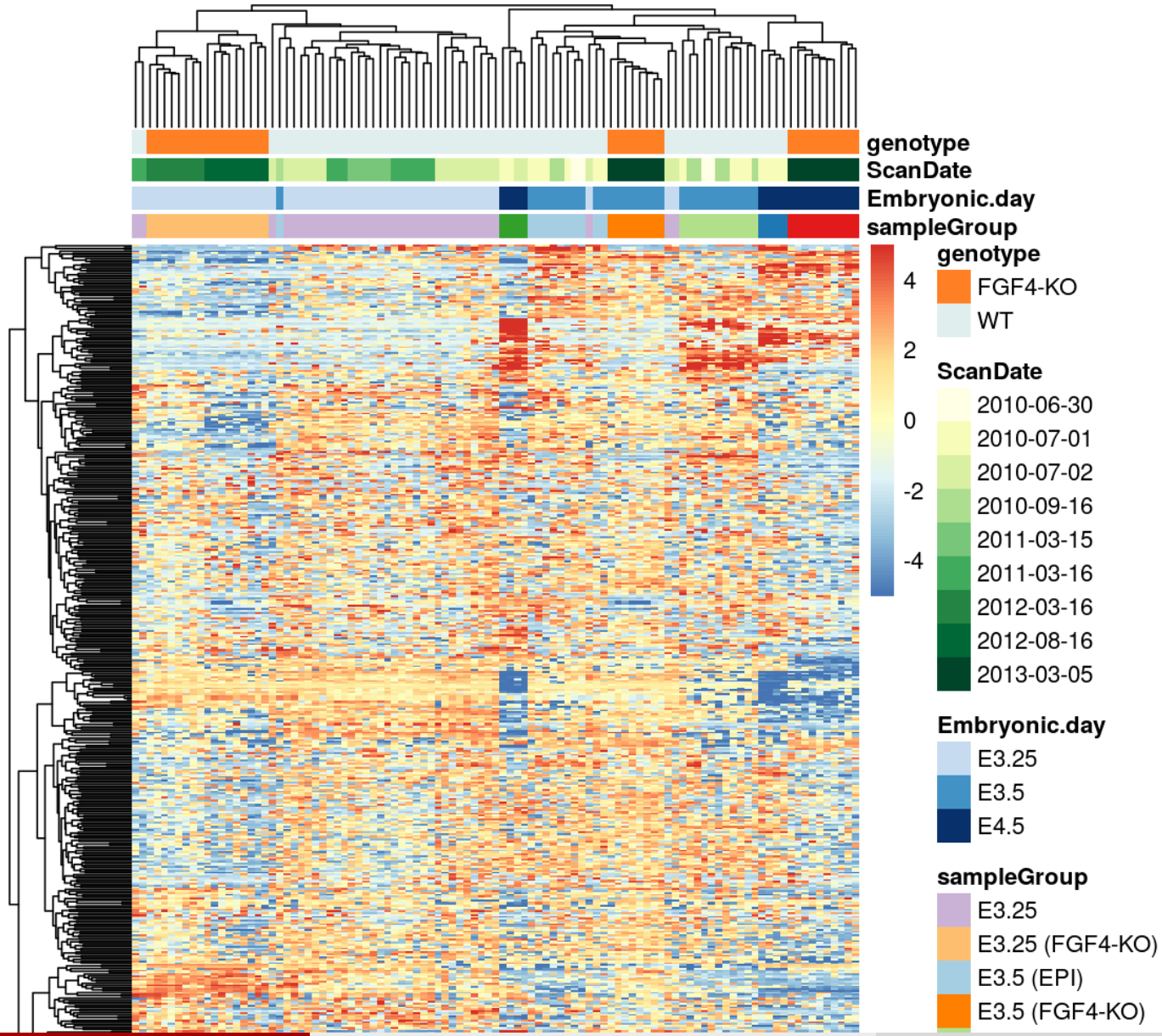
Finally, we can clean up our main legend, change the color scheme, and very clearly show missing values for our final figure.

```
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"
)
```

NYC13Flights



You can also choose to go bananas



Section 4

Wrapping up