In [1]:

```
library(tidyverse)
```

```
── Attaching packages ─────────────────────────────────────── tidyverse 1.2.1 ──
✔ ggplot2 2.2.1     ✔ purrr   0.2.4
✔ tibble  1.4.1     ✔ dplyr   0.7.4
✔ tidyr   0.7.2     ✔ stringr 1.2.0
✔ readr   1.1.1     ✔ forcats 0.2.0
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
```

STATS 306

Problem set 1: plotting with `ggplot`

Part 1

Factors

Factors are used in R to represent discrete or categorical variables. We will discuss factors at greater length later in the course. This exercise is designed to show you how `ggplot` handles factors when plotting data.

Use the `tribble()` command to re-create the following small data table. (See the lecture 02 notes for an example of how to use this commmand.) Store your table in a variable called `toy_table`:

```
# A tibble: 4 x 2
      x     y
  <chr> <dbl>
1     a     1
2     b     2
3     c     4
4     d     5
```

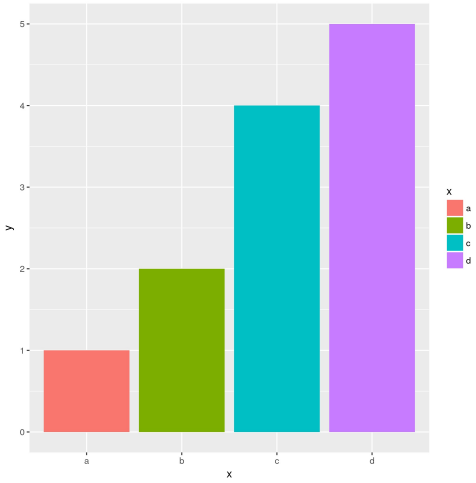
In [2]:

```
### BEGIN SOLUTION
toy_table = tribble(
~x, ~y,
'a', 1,
'b', 2,
'c', 4,
'd', 5)
### END SOLUTION
```

In [3]:

```
### BEGIN HIDDEN TESTS
soln = tribble(
~x, ~y,
'a', 1,
'b', 2,
'c', 4,
'd', 5)
stopifnot(all.equal(toy_table, soln))
### END HIDDEN TESTS
```

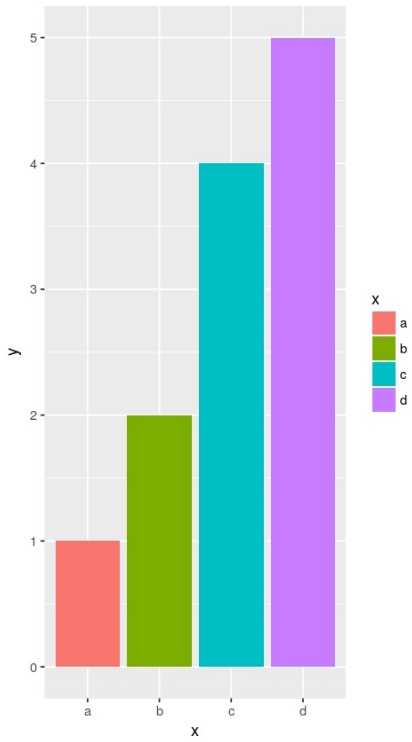
Use `ggplot` to generate the following plot from `toy_table`:



In [4]:

```
### BEGIN SOLUTION
ggplot(toy_table) + geom_col(aes(x=x, y=y, fill=x))
ggsave('toy_table.png')
### END SOLUTION
```

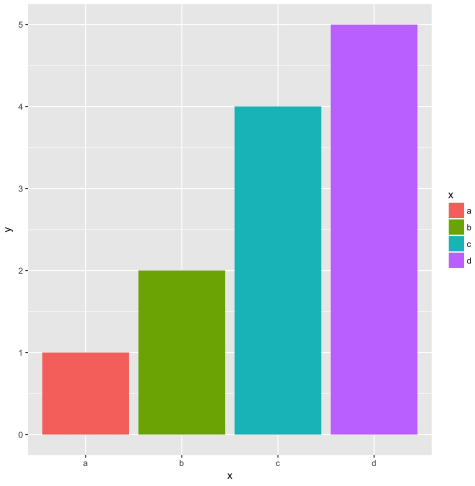
```
Saving 6.67 x 6.67 in image
```



Suppose that we want to change the ordering of the x-axis to be d c b a. To do this we need to convert x to a factor and reorder its levels. The syntax for doing this is

```
{r}
factor(x, levels=c('d','c','b','a'))
```

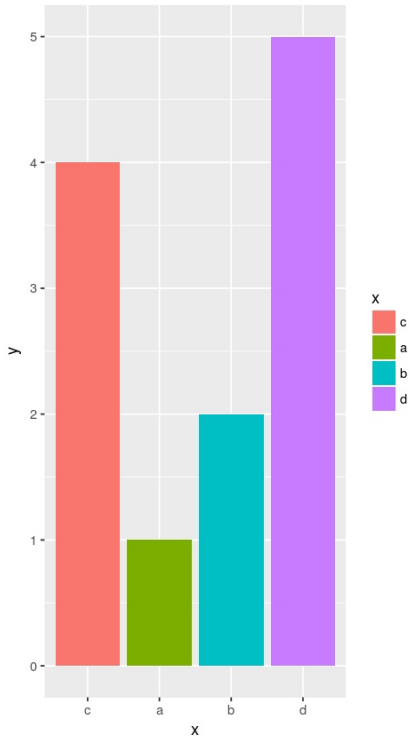
This produces the following plot:



Use this technique to produce a plot where the bars are ordered `c,a,b,d':

In [5]:

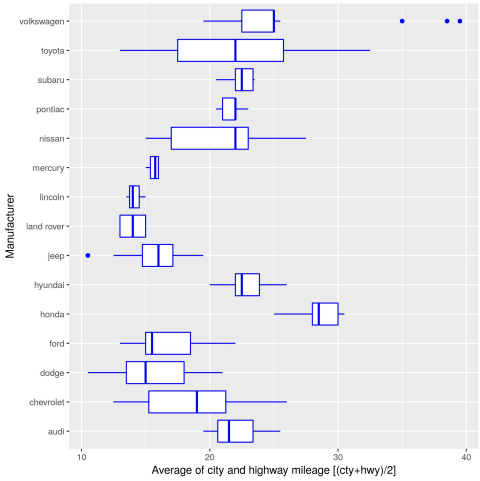
```
### BEGIN SOLUTION
toy_table = mutate(toy_table, x=factor(x, levels=c('c','a','b','d')))
ggplot(toy_table) + geom_col(aes(x=x, y=y, fill=x))
### END SOLUTION
```



Part 2

For each of the plots shown below, enter the R code in the cell provided which *exactly reproduces* the plot. (Your could should print this plot in the notebook.)

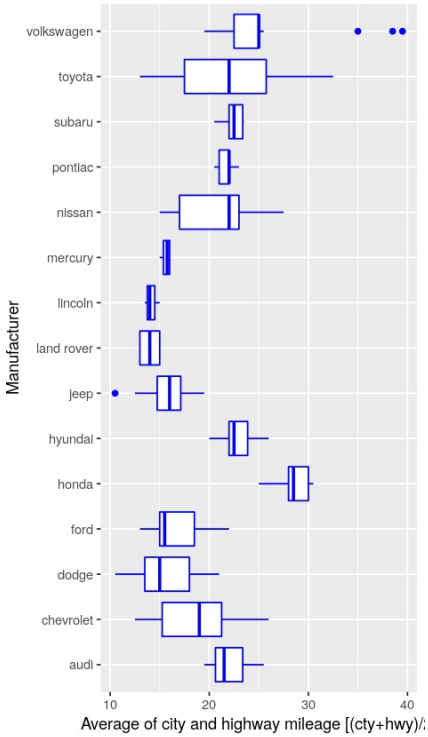
Plot 1



In [6]:

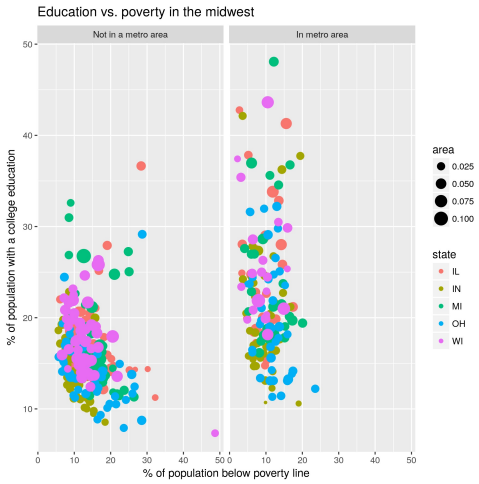
```
library(tidyverse)
ggplot(mpg) + geom_boxplot(aes(x=manufacturer, y=(cty + hwy)/2), color="blue") + coord_flip() + xlab("Manufa
cturer") + ylab("Average of city and highway mileage [(cty+hwy)/2]")
ggsave('p1.png')
```

```
Saving 6.67 x 6.67 in image
```



Plot 2

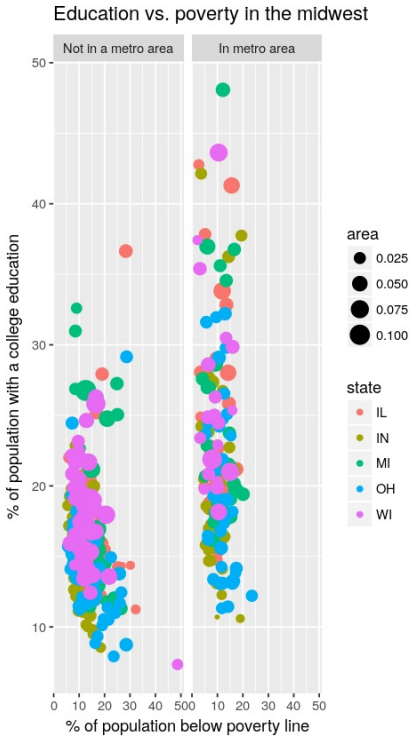
This is using the `midwest` data set which we discussed at the end of lecture 02.



In [7]:

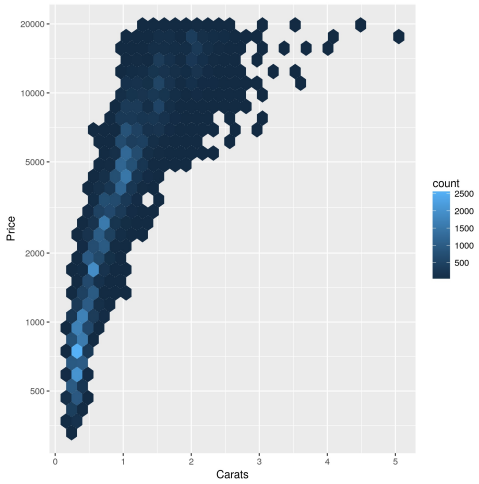
```
### BEGIN SOLUTION
data(midwest)
midwest = midwest %>% mutate(inmetro = factor(inmetro, labels=c("Not in a metro area", "In metro area")))
ggplot(midwest) + geom_point(mapping=aes(x=percbelowpoverty,y=percollege,size=area,color=state)) +
                  facet_wrap(~ inmetro) + ylab("% of population with a college education") + xlab("% of popu
lation below poverty line") +
                  ggtitle("Education vs. poverty in the midwest")
ggsave('p2.png')
### END SOLUTION
```

```
Saving 6.67 x 6.67 in image
```



Plot 3

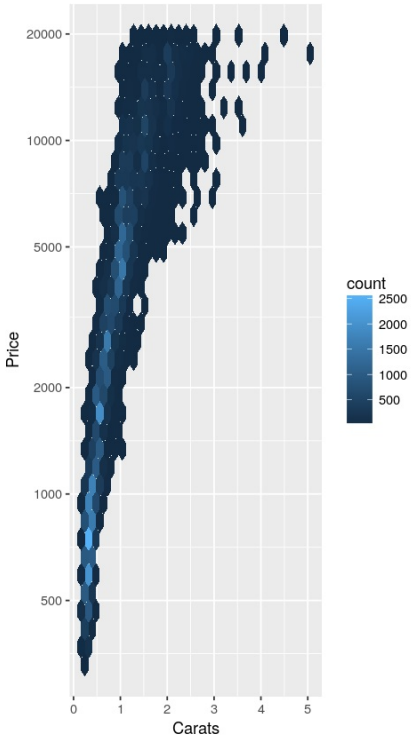
(Hint: for this plot, you will need to manually manipulate the ticks on the y-axis using the `breaks=` option of the appropriate `scale_y` command.)



In [8]:

```
### BEGIN SOLUTION
ggplot(diamonds) + geom_hex(aes(x=carat, y=price)) + scale_y_log10(breaks=c(5e2,1e3,2e3,5e3,1e4,2e4)) +
                   ylab("Price") + xlab("Carats")
ggsave('p3.png')
### END SOLUTION
```

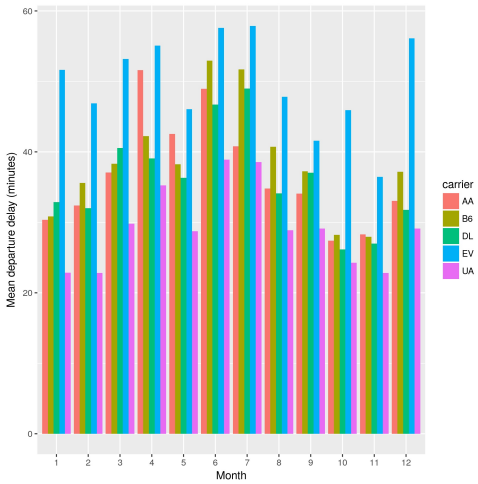
```
Saving 6.67 x 6.67 in image
```



Plot 4

In lecture 00 we looked at a database on flights in New York City in 2013. This data set is not part of the `tidyverse` package. You may need to install it using the command:

```
{r}
install.packages('nycflights13')
```



This plots is average delay time by carrier for the top five carriers. Because we have not yet covered the material needed to compute such an average, the code that does this has been provided for you. Use the resulting dataset, `flights_top5`, to generate the plot.

In [9]:

```
library(nycflights13)
top5_carriers = group_by(flights, carrier) %>% summarize(departures=n()) %>%
                ungroup() %>% top_n(5)
flights_top5 =  filter(flights, carrier %in% top5_carriers$carrier) %>%
                filter(dep_delay>0) %>% group_by(carrier, month) %>%
                summarize(mean_dep_delay=mean(dep_delay))
```

```
Selecting by departures
```

In [10]:

```
### BEGIN SOLUTION
library(nycflights13)
ggplot(flights_top5) + geom_col(aes(x=factor(month), y=mean_dep_delay, fill=carrier), position="dodge") +
xlab("Month") + ylab("Mean departure delay (minutes)")
ggsave("p4.png")
### END SOLUTION
```

```
Saving 6.67 x 6.67 in image
```