Introduction to Data Visualization in R

Introduction to Data Visualization in R

In this notebook, we will explore how to create basic visualizations using R. This notebook builds on the lessons you have learned in using the dplyr. We will focus on plotting capabilities in base R as well as in and the `ggplot2` library, which is part of the tidyverse.

Python vs. R for data visualization

Both Python and R offer robust data visualization capabilities, with each language having its own strengths and challenges. Python's ecosystem is diverse and well-suited for integration with web-based applications, while R's ggplot2 package offers a powerful and expressive syntax for creating publication-quality graphics. Ultimately, the choice between Python and R for data visualization depends on factors such as personal preference, project requirements, and existing expertise in each language.

Examples of Basic Visualizations

In this notebook, we will explore bar charts, histograms, scatter plots, and line plots. We will look at how to do this with base R as well as with `ggplot2`.

Loading the dataset

Let's load the colleges dataset - the same one we looked at for Python data viz.

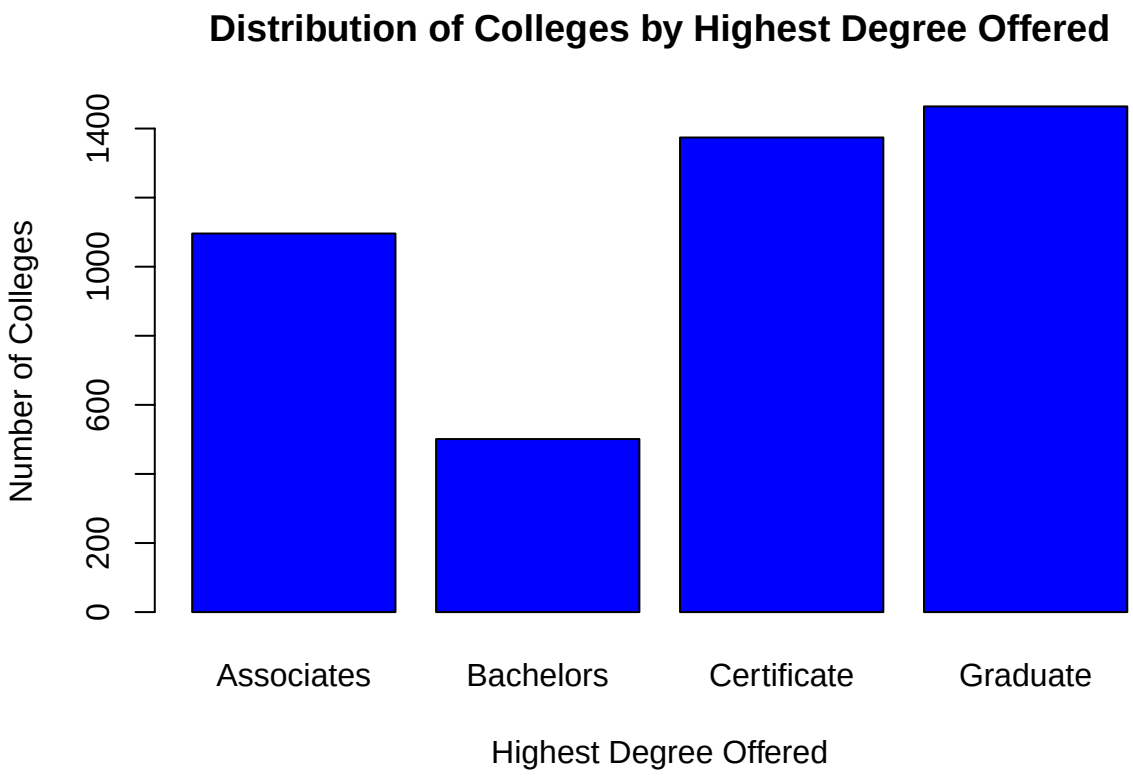
```
collegesDat <- read_csv('data/colleges.csv')
```

```
## Rows: 4435 Columns: 26
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (9): name, city, state, region, highest_degree, ownership, locale, hbcu...
## dbl (17): OPEID, median_debt, default_rate, admit_rate, SAT_avg, enrollment,...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

(1) Bar Charts

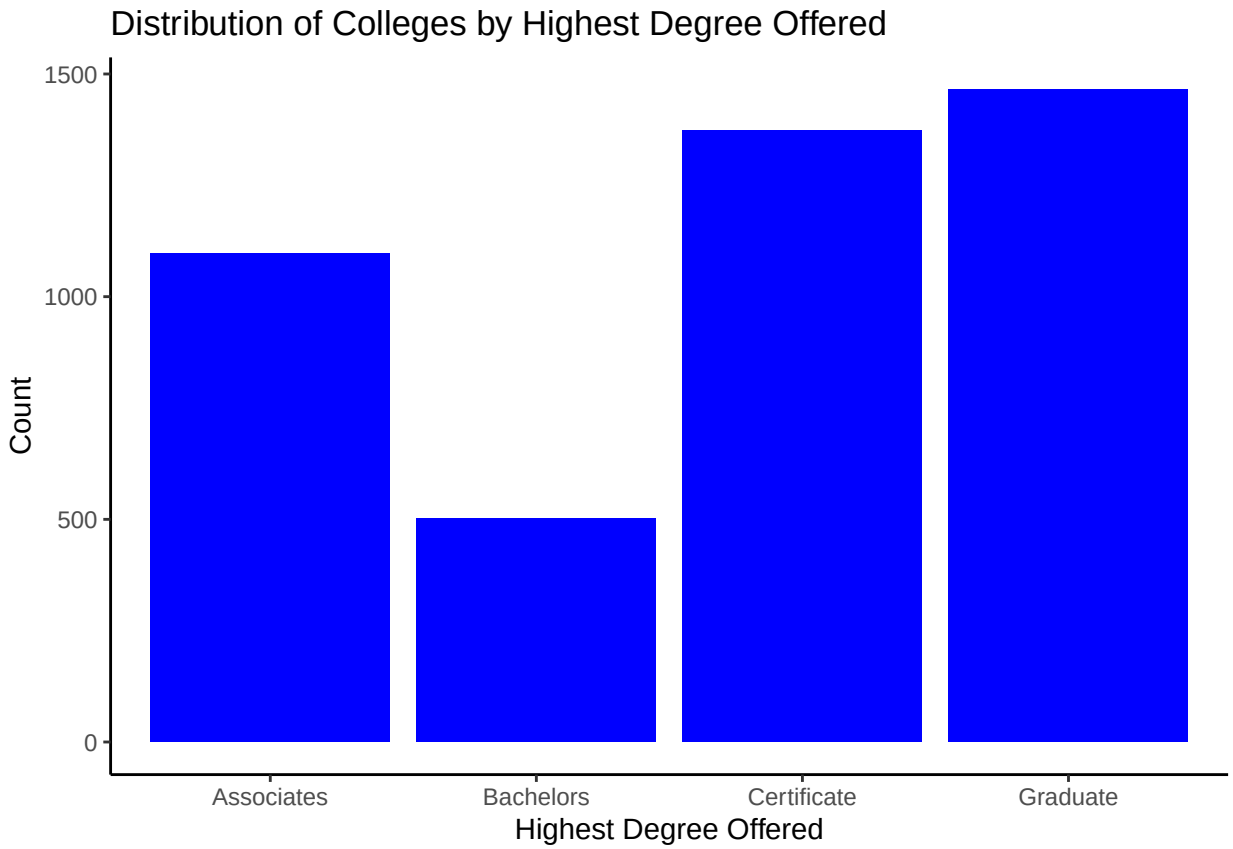
Basic R In base R, we use the `barplot()` function. The help documentation provides lots of assistance with different specifications to make your plot more readable.

```
highest_degree_counts <- table(collegesDat$highest_degree)
barplot(highest_degree_counts,
        main="Distribution of Colleges by Highest Degree Offered",
        xlab="Highest Degree Offered",
        ylab="Number of Colleges",
        col="blue")
```



ggplot2 Here is how we make the same plot using `ggplot2`.

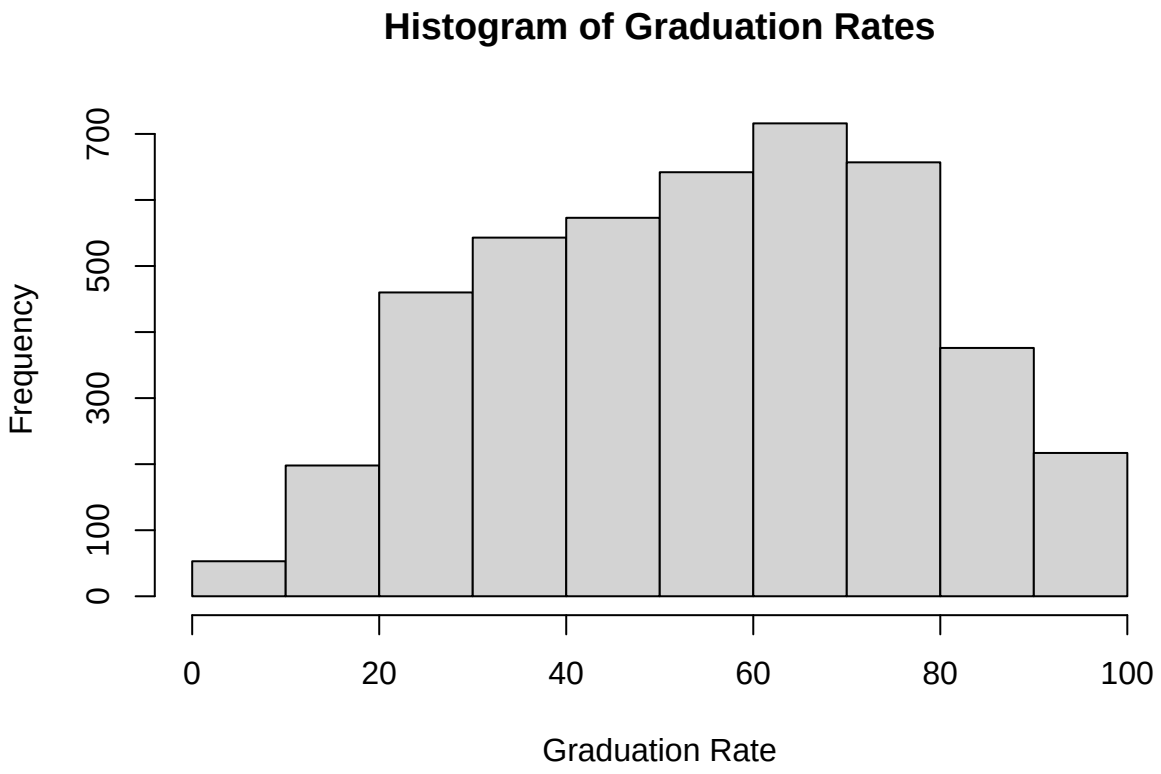
```
ggplot(collegesDat, aes(x=highest_degree)) +
  geom_bar(fill="blue") + # this specifies to create a bar chart
  theme_classic() +  # this specifies a set of formatting options
  labs(title="Distribution of Colleges by Highest Degree Offered", x="Highest Degree Offered", y="Count
```



(2) Histograms

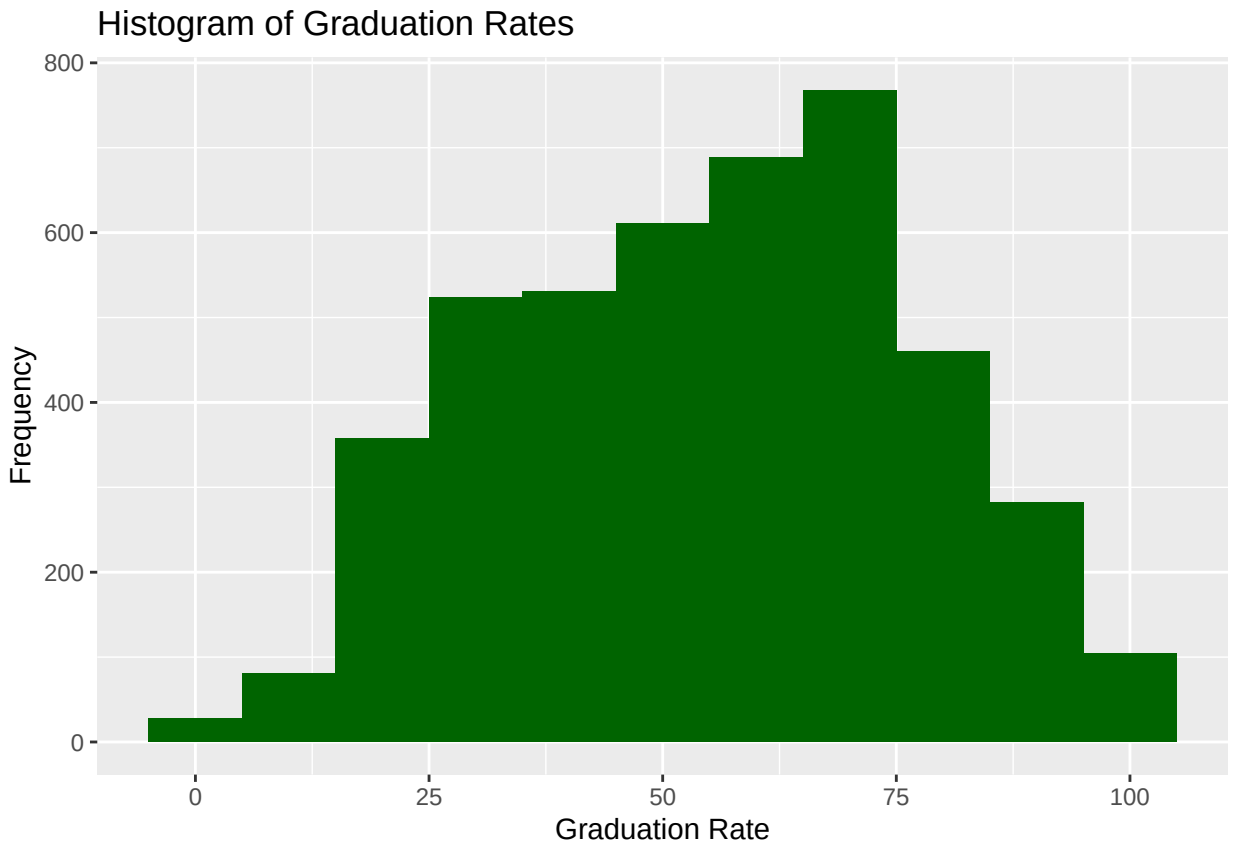
Basic R The hist function in R creates a histogram. Again see `help(hist)` for info on specifications for tailoring your plot.

```
hist(collegesDat$grad_rate, main="Histogram of Graduation Rates", xlab="Graduation Rate", breaks=10)
```



ggplot2 In ggplot2, we start the same way we did for the bar plot (the first line is exactly the same), but now we use `geom_histogram()`.

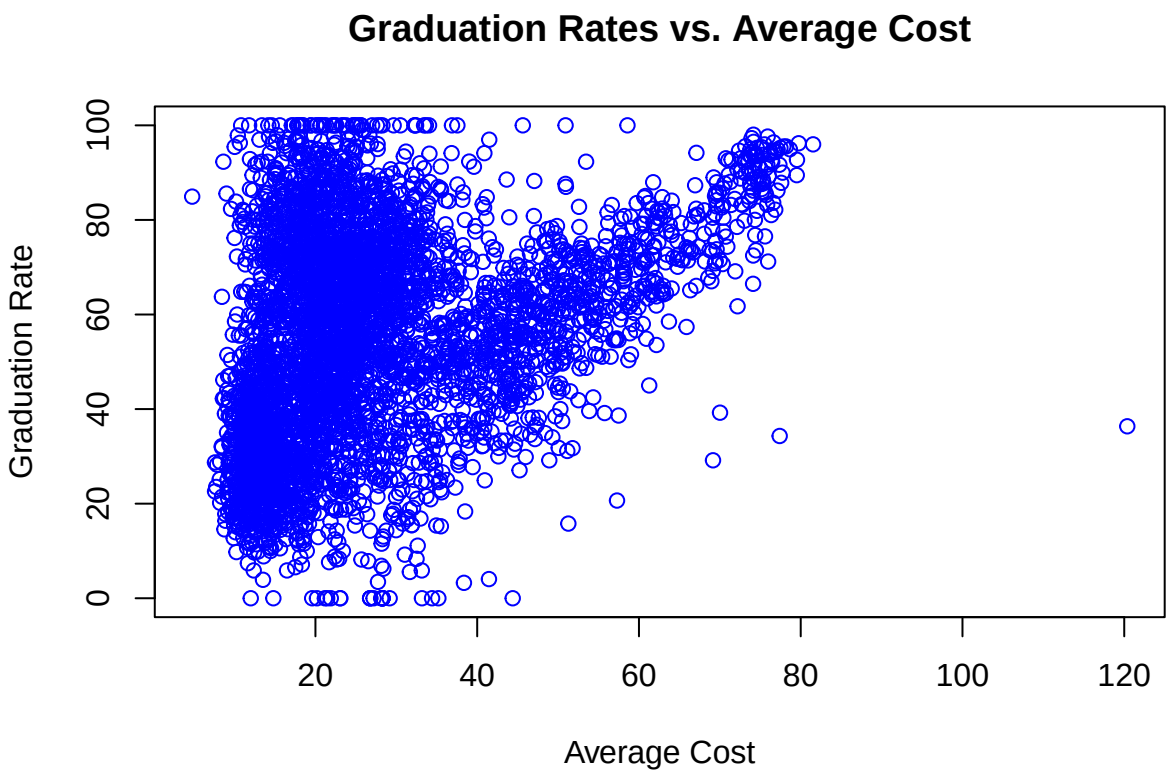
```
ggplot(collegesDat, aes(x=grad_rate)) +
  geom_histogram(binwidth=10,fill="dark green") +
  labs(title="Histogram of Graduation Rates", x="Graduation Rate", y="Frequency")
```



(3) Scatter Plots

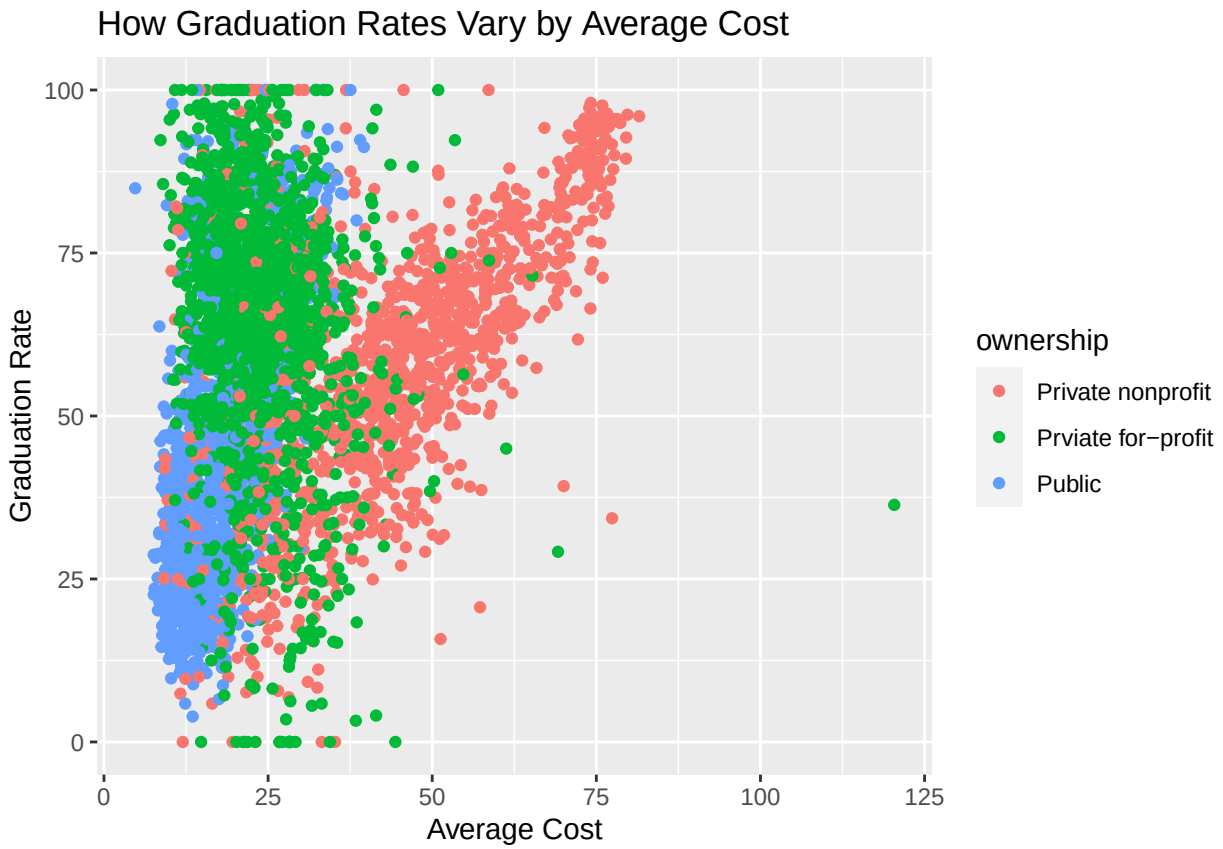
Basic R For a quick plot in base R, simply use the `plot()` function.

```
plot(collegesDat$avg_cost, collegesDat$grad_rate, main="Graduation Rates vs. Average Cost", xlab="Averag
```



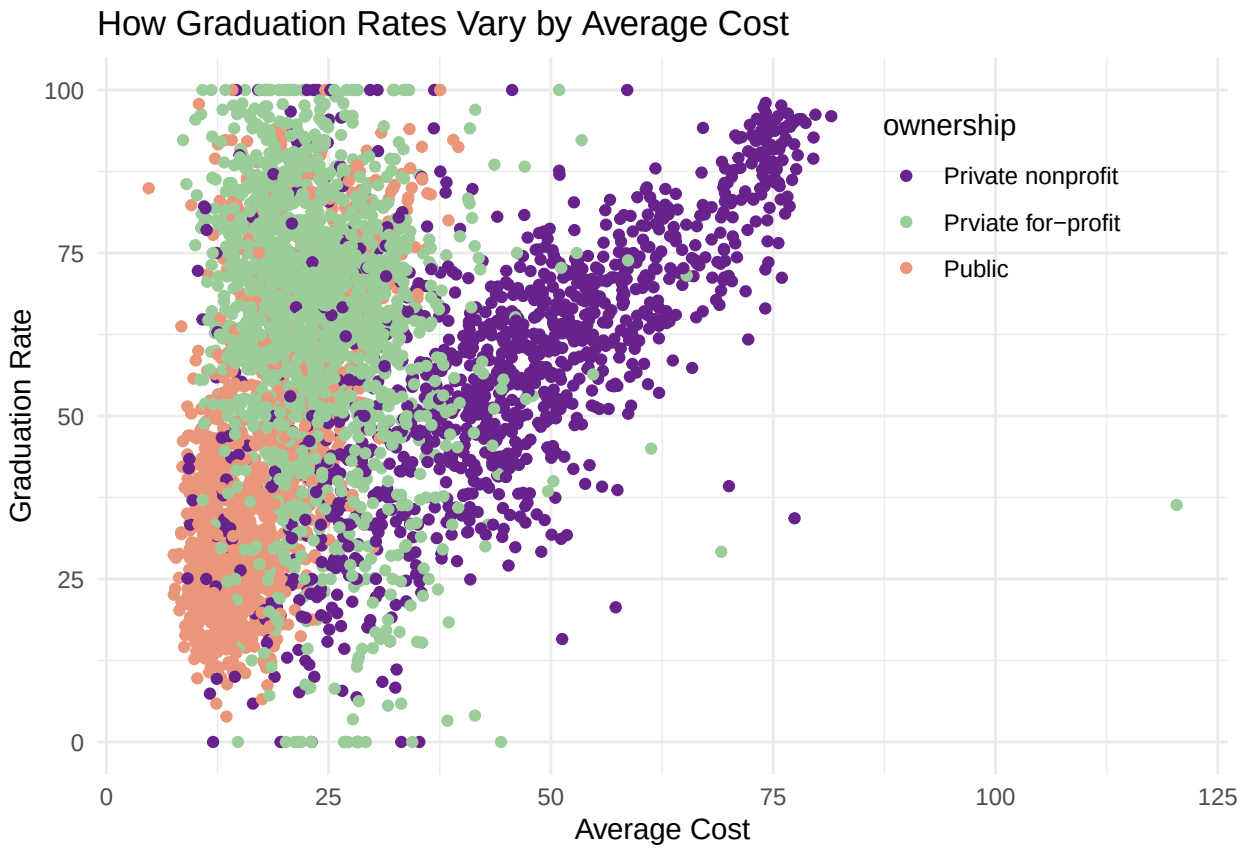
ggplot2 But here, let's apply the grammar of graphics so that we can see differences by ownership of the colleges. We will add this as a third aesthetic, assigning college ownership to color. Here, I am going to assign my plot to an object. This will allow me to add to it later, without having to repeat everything. When I do this, I need to include a print command.

```
scatter1 <- ggplot(collegesDat, aes(x=avg_cost, y=grad_rate, color=ownership)) +
  geom_point() +
  labs(title="How Graduation Rates Vary by Average Cost", x="Average Cost", y="Graduation Rate")
print(scatter1)
```



So let's try to make it prettier. Here are some steps towards that. Note, you can find a set of color options here[https://r-graph-gallery.com/ggplot2-color.html].

```
scatter2 <- scatter1 +
            theme_minimal() + # using theme_minimal to change look of plot
            theme(legend.position = c(0.8, 0.8)) + # moves the legend to the upper right
            scale_color_manual(values = c("Private nonprofit" = "darkorchid4", "Prviate for-profit" = "
print(scatter2)
```

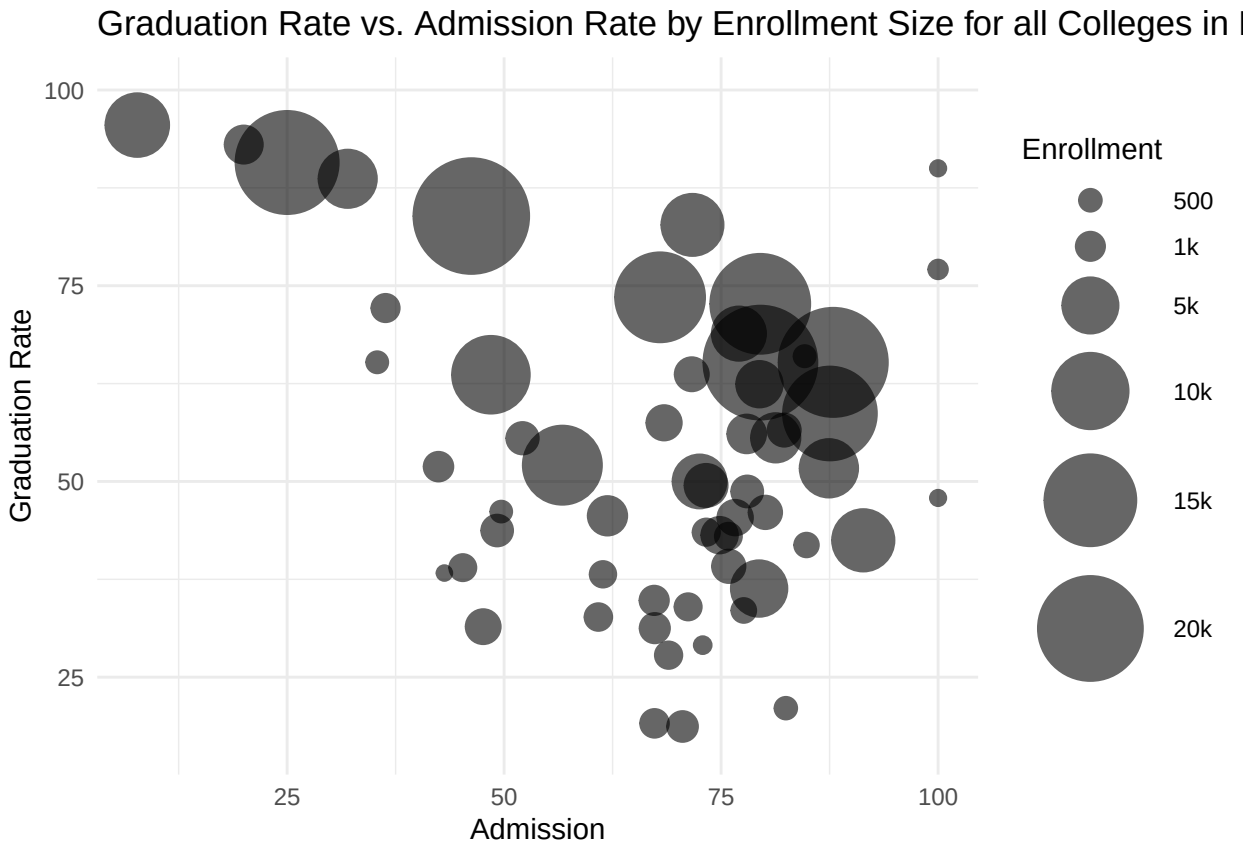


Here I want to add a new variation of the scatter plot - a bubble plot. This simply uses the SIZE of the points to add a third variable's value to the plot. For example, we could use the size to indicate the enrollment. Here's an example:

```
# Let's restrict our analysis just to private schools in which the highest degree is a Bachelors degree
filterDat <- collegesDat %>%
  filter(state == "NC")

# Bubble plot
ggplot(filterDat, aes(x = admit_rate, y = grad_rate, size = enrollment)) +
  geom_point(alpha = 0.6) +
  scale_size_continuous(range = c(1, 20)) +
  labs(title = "Graduation Rate vs. Admission Rate by Enrollment Size for all Colleges in North Carolina
       x = "Admission",
       y = "Graduation Rate",
       size = "Enrollment") +
  theme_minimal() +
  theme(legend.position = "right") +
  scale_size_continuous(range = c(1, 20),
                        breaks = c(500, 1000, 5000, 10000, 15000, 20000), # Adjust breaks to suitable e
                        labels = c("500", "1k", "5k", "10k", "15k", "20k"))  # Label adjustments for re
```

```
## Scale for size is already present.
## Adding another scale for size, which will replace the existing scale.
```

```
## Warning: Removed 60 rows containing missing values (`geom_point()`).
```



(4) Line Plots

Here we will use the UFO data. Like we did in python, we need to first get counts of sightings by hear. We can do this with `dplyr`.

```
UFOdat <- read_csv("data/ufo_sightings.csv")
```

```
## Rows: 60632 Columns: 16
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (5): Location.City, Location.State, Location.Country, Data.Shape, Data....
## dbl (11): Data.Encounter duration, Location.Coordinates.Latitude, Location.C...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

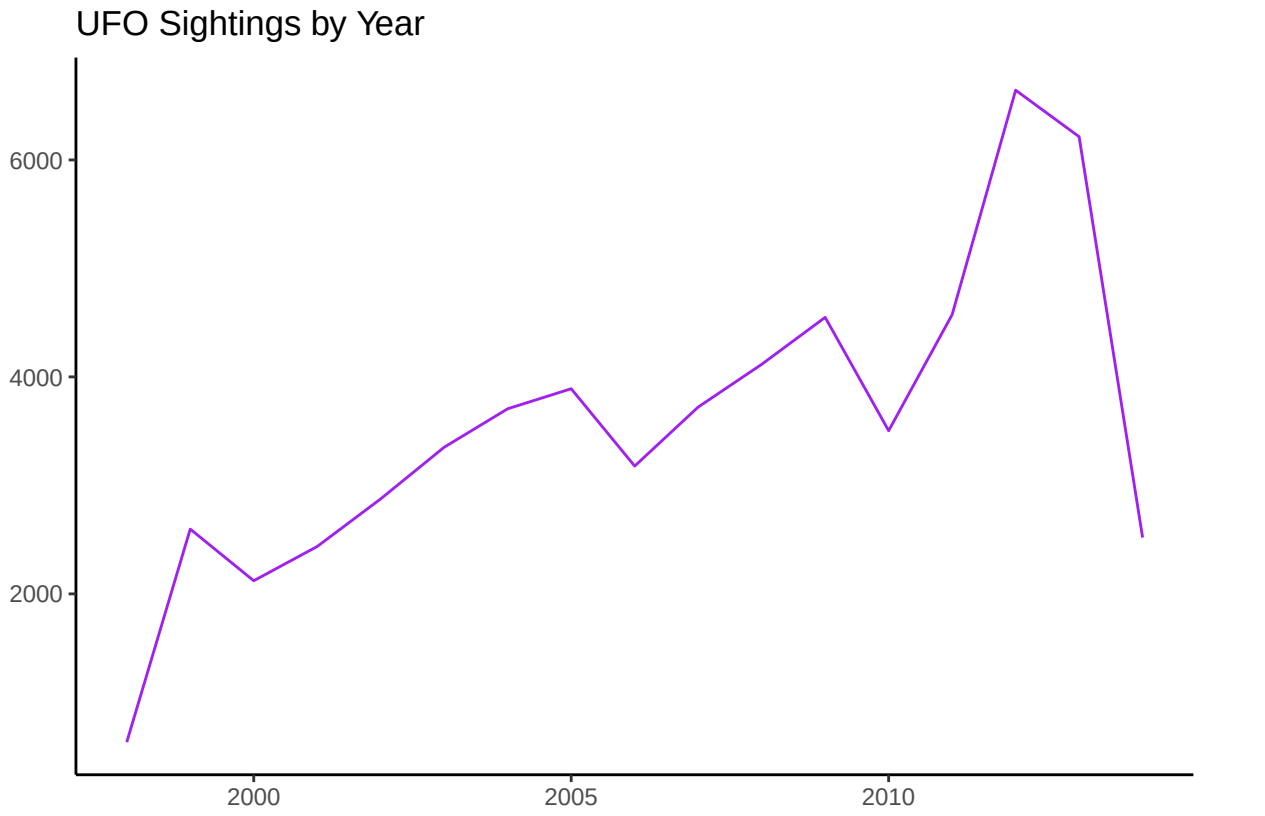
```
sightings_by_year <- UFOdat %>%
  group_by(Dates.Documented.Year) %>%
  summarise(Count = n())

# To view the result
print(sightings_by_year)
```

```
## # A tibble: 17 x 2
##    Dates.Documented.Year Count
##                    <dbl> <int>
##  1                  1998   633
##  2                  1999  2597
```

```
##  3                   2000  2121
##  4                   2001  2437
##  5                   2002  2876
##  6                   2003  3352
##  7                   2004  3706
##  8                   2005  3891
##  9                   2006  3178
## 10                   2007  3722
## 11                   2008  4116
## 12                   2009  4549
## 13                   2010  3504
## 14                   2011  4574
## 15                   2012  6643
## 16                   2013  6214
## 17                   2014  2519
```

Next, let's plot with `ggplot2`.

```
ggplot(sightings_by_year, aes(x=Dates.Documented.Year, y=Count)) +
  geom_line(color="purple") +
  theme_classic() +
  labs(title="UFO Sightings by Year", x="", y="")
```



Finally, let's replicate what we did in Python to show a line plot for each of the states, highlight just one state to compare with the rest.

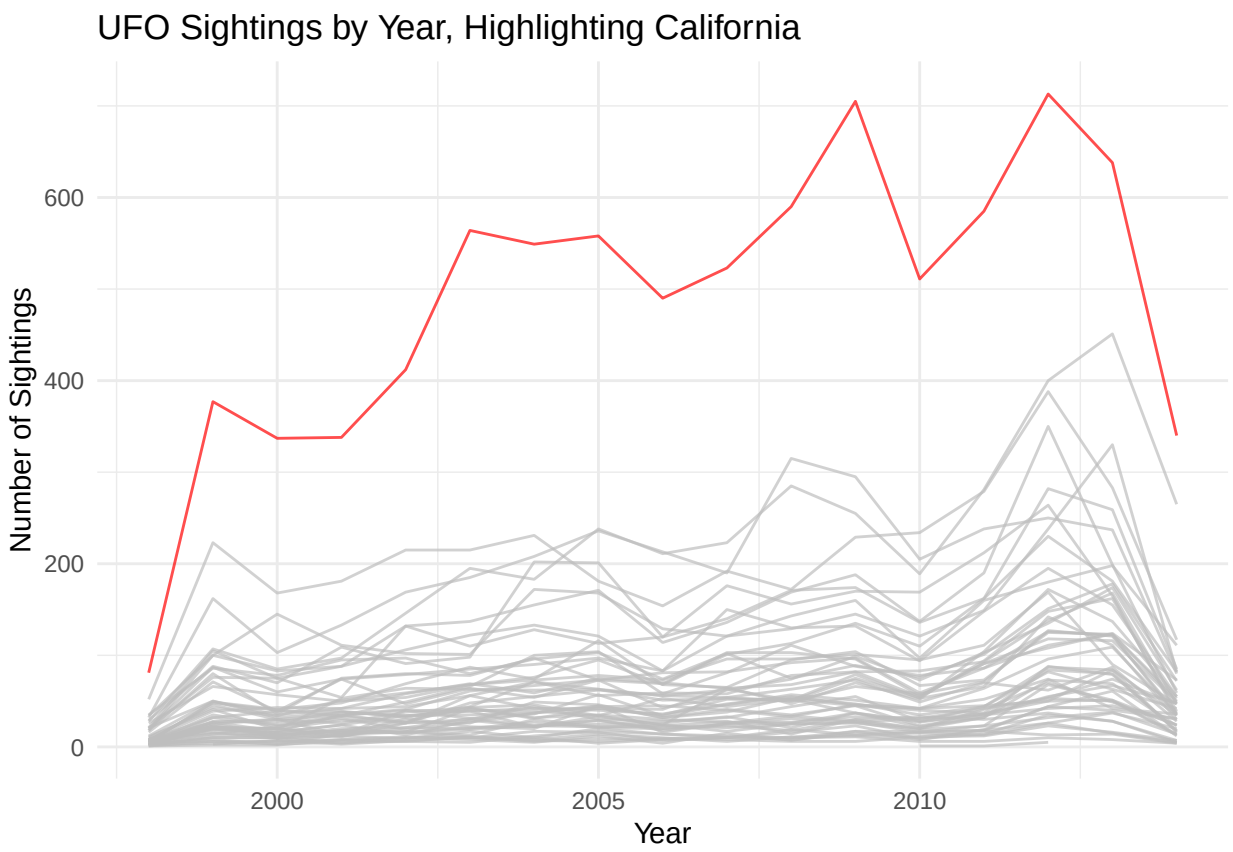
```
# First, we need sightings by year and state
sightings_by_year_state <- UFOdat %>%
  group_by(Dates.Documented.Year, Location.State) %>%
  summarise(Count = n())
```

```
## `summarise()` has grouped output by 'Dates.Documented.Year'. You can override
## using the `.groups` argument.
```

```
# Let's check our output
head(sightings_by_year_state)
```

```
## # A tibble: 6 x 3
## # Groups:   Dates.Documented.Year [1]
##   Dates.Documented.Year Location.State Count
##                   <dbl> <chr>          <int>
## 1                  1998 AK                 8
## 2                  1998 AL                 2
## 3                  1998 AR                 2
## 4                  1998 AZ                34
## 5                  1998 CA                81
## 6                  1998 CO                18
```

```
# Let's pick a state to highlight
highlight_state <- 'CA'

# And plot note, we use the aes of group
ggplot(sightings_by_year_state, aes(x = Dates.Documented.Year, y = Count, group = Location.State)) +
  geom_line(aes(color = Location.State == highlight_state), alpha = 0.7) +
  scale_color_manual(values = c("FALSE" = "gray", "TRUE" = "red")) +
  labs(title = "UFO Sightings by Year, Highlighting California",
       x = "Year",
       y = "Number of Sightings") +
  theme_minimal() +
  theme(legend.position = "none") # Hide the legend
```


UFO Sightings by Year, Highlighting California
Number of Sightings
600
400
200
0
2000
2005
2010
Year