ds2

Matthew Robinson

library(ggplot2)  
library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.4 ✔ readr 2.1.5  
## ✔ forcats 1.0.0 ✔ stringr 1.5.1  
## ✔ lubridate 1.9.3 ✔ tibble 3.2.1  
## ✔ purrr 1.0.2 ✔ tidyr 1.3.1  
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

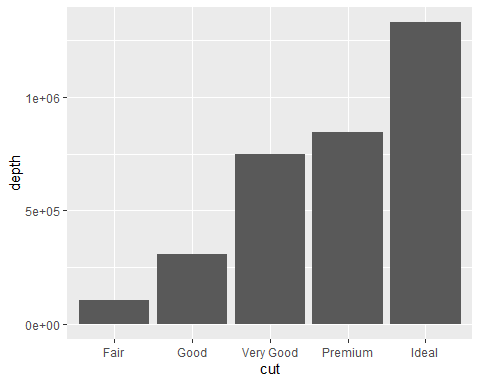
library(nycflights13)

# Section 3.7.1

# Exercise 2

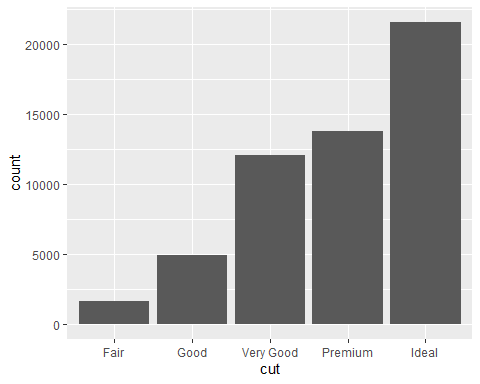
What does geom\_col() do?

ggplot(data = diamonds) + geom\_col(mapping = aes(x = cut, y = depth))



The geom\_bar() function creates a bar chart with the specified x and y values.  
How is it different to geom\_bar()?

ggplot(data = diamonds) + geom\_bar(mapping = aes(x = cut))

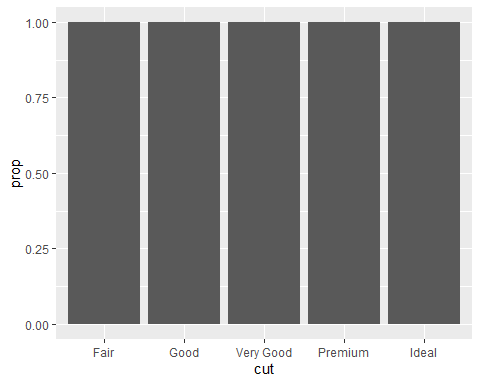


The geom\_bar function does not require a y value and defaults to count.

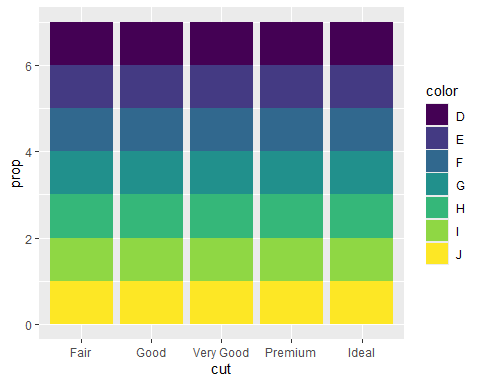
# Exercise 5

In our proportion bar chart, we need to set group = 1. Why? In other words what is the problem with these two graphs?

ggplot(data = diamonds) + geom\_bar(mapping = aes(x = cut, y = after\_stat(prop)))

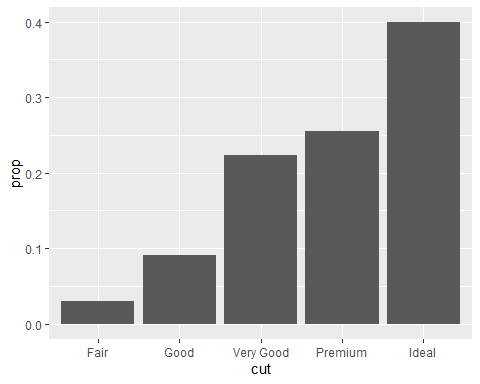


ggplot(data = diamonds) + geom\_bar(mapping = aes(x = cut, fill = color, y = after\_stat(prop)))



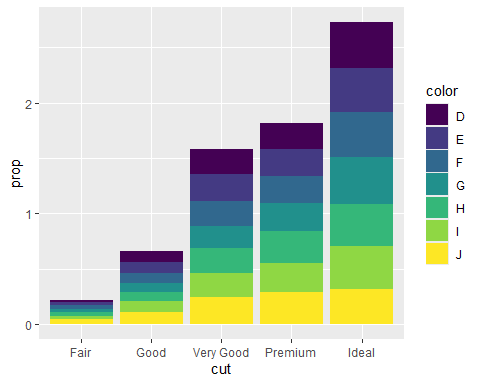
The above graphs are are being graphed based on count as opposed to proportion.

ggplot(data = diamonds) + geom\_bar(mapping = aes(x = cut, y = after\_stat(prop), group = 1))



For the first graph, you must add the “group = 1” command as shown above to get the correct graph.

ggplot(data = diamonds) + geom\_bar(mapping = aes(x = cut, fill = color, y = after\_stat(prop), group = color))



For the second graph, you must add the “group = color” command as shown above to get the correct graph.

# Section 3.8.1

# Exercise 1

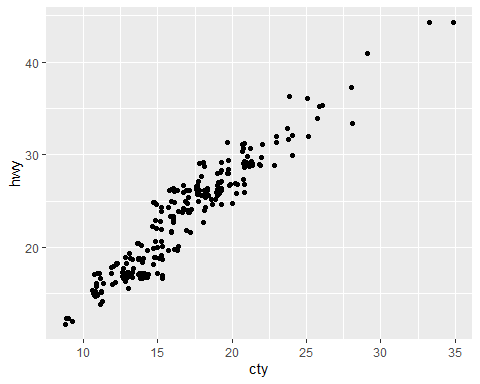
What is the problem with this plot?

ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) + geom\_point()



Many of the points on the above plot overlap each other which makes it hard to see where most of the data is.  
How could you improve it?

ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) + geom\_jitter()



This can be improved by using geom\_jitter instead of geom\_point as shown above.

# Exercise 3

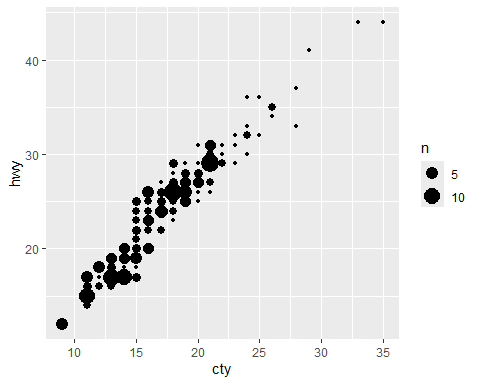
Compare and contrast geom\_jitter() with geom\_count().

ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) + geom\_jitter()



The geom\_jitter function scatters the points, so they do not overlap.

ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) + geom\_count()



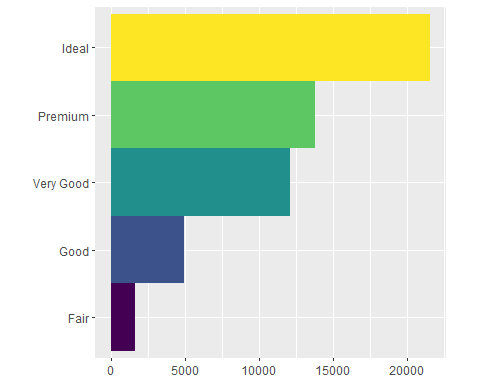
The geom\_count function makes the points bigger when there are multiple points in the same place.

# Section 3.9.1

# Exercise 1

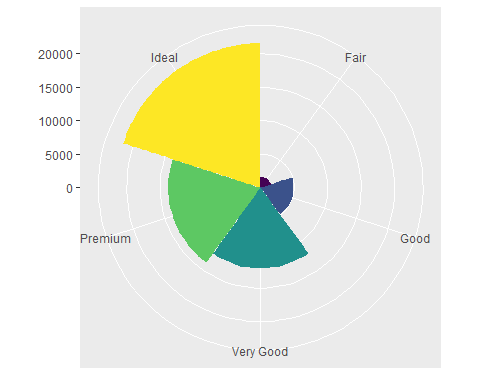
Turn a stacked bar chart into a pie chart using coord\_polar().

ggplot(data = diamonds) + geom\_bar(mapping = aes(x = cut, fill = cut), show.legend = FALSE, width = 1) + theme(aspect.ratio = 1) + labs(x = NULL, y = NULL) + coord\_flip()



The coord\_flip() command is used to create a stacked bar chart.

ggplot(data = diamonds) + geom\_bar(mapping = aes(x = cut, fill = cut), show.legend = FALSE, width = 1) + theme(aspect.ratio = 1) + labs(x = NULL, y = NULL) + coord\_polar()

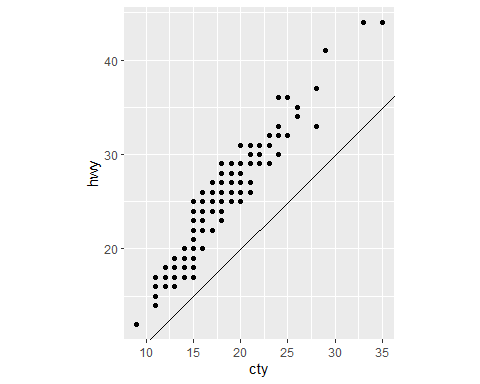


The coord\_polar command is used to create a pie chart.

# Exercise 4

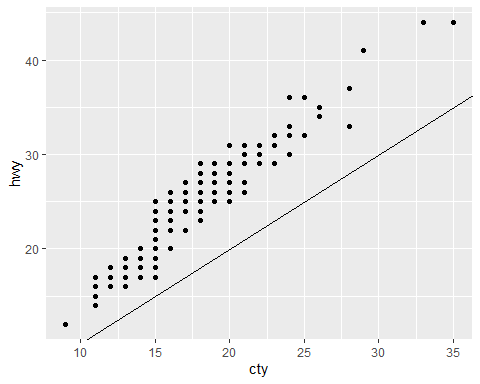
What does the plot below tell you about the relationship between city and highway mpg?

ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) + geom\_point() + geom\_abline() + coord\_fixed()



Highway mpg is generally better than city mpg.  
Why is coord\_fixed() important?

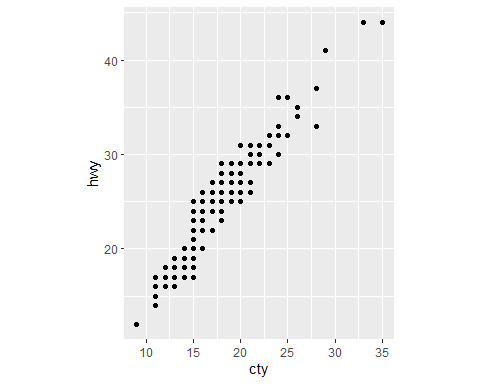
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) + geom\_point() + geom\_abline()



The coord\_fixed() function keeps ensures that x and y values are the same distance apart on each axis.

What does geom\_abline() do?

ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) + geom\_point() + coord\_fixed()



The geom\_abline() function adds a line that shows the trend of the data.

# Exercise 5.2.4

# Exercise 1

Find all flights that had an arrival delay of two or more hours.

filter(flights, arr\_delay >= 120)

## # A tibble: 10,200 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 811 630 101 1047 830  
## 2 2013 1 1 848 1835 853 1001 1950  
## 3 2013 1 1 957 733 144 1056 853  
## 4 2013 1 1 1114 900 134 1447 1222  
## 5 2013 1 1 1505 1310 115 1638 1431  
## 6 2013 1 1 1525 1340 105 1831 1626  
## 7 2013 1 1 1549 1445 64 1912 1656  
## 8 2013 1 1 1558 1359 119 1718 1515  
## 9 2013 1 1 1732 1630 62 2028 1825  
## 10 2013 1 1 1803 1620 103 2008 1750  
## # ℹ 10,190 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

Find all flights that flew to Houston (IAH or HOU).

filter(flights, dest == "IAH" | dest == "HOU")

## # A tibble: 9,313 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 623 627 -4 933 932  
## 4 2013 1 1 728 732 -4 1041 1038  
## 5 2013 1 1 739 739 0 1104 1038  
## 6 2013 1 1 908 908 0 1228 1219  
## 7 2013 1 1 1028 1026 2 1350 1339  
## 8 2013 1 1 1044 1045 -1 1352 1351  
## 9 2013 1 1 1114 900 134 1447 1222  
## 10 2013 1 1 1205 1200 5 1503 1505  
## # ℹ 9,303 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

Find all flights that were operated by United, American, or Delta.

filter(flights, carrier == "UA" | carrier == "AA" | carrier == "DL")

## # A tibble: 139,504 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 542 540 2 923 850  
## 4 2013 1 1 554 600 -6 812 837  
## 5 2013 1 1 554 558 -4 740 728  
## 6 2013 1 1 558 600 -2 753 745  
## 7 2013 1 1 558 600 -2 924 917  
## 8 2013 1 1 558 600 -2 923 937  
## 9 2013 1 1 559 600 -1 941 910  
## 10 2013 1 1 559 600 -1 854 902  
## # ℹ 139,494 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

Find all flights that departed in summer (July, August, and September).

filter(flights, month == 7 | month == 8 | month == 9)

## # A tibble: 86,326 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 7 1 1 2029 212 236 2359  
## 2 2013 7 1 2 2359 3 344 344  
## 3 2013 7 1 29 2245 104 151 1  
## 4 2013 7 1 43 2130 193 322 14  
## 5 2013 7 1 44 2150 174 300 100  
## 6 2013 7 1 46 2051 235 304 2358  
## 7 2013 7 1 48 2001 287 308 2305  
## 8 2013 7 1 58 2155 183 335 43  
## 9 2013 7 1 100 2146 194 327 30  
## 10 2013 7 1 100 2245 135 337 135  
## # ℹ 86,316 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

Find all flights that arrived more than two hours late, but didn’t leave late.

filter(flights, arr\_delay > 120 & dep\_delay <= 0)

## # A tibble: 29 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 27 1419 1420 -1 1754 1550  
## 2 2013 10 7 1350 1350 0 1736 1526  
## 3 2013 10 7 1357 1359 -2 1858 1654  
## 4 2013 10 16 657 700 -3 1258 1056  
## 5 2013 11 1 658 700 -2 1329 1015  
## 6 2013 3 18 1844 1847 -3 39 2219  
## 7 2013 4 17 1635 1640 -5 2049 1845  
## 8 2013 4 18 558 600 -2 1149 850  
## 9 2013 4 18 655 700 -5 1213 950  
## 10 2013 5 22 1827 1830 -3 2217 2010  
## # ℹ 19 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

Find all flights that were delayed by at least an hour, but made up over 30 minutes in flight.

filter(flights, dep\_delay >= 60 & arr\_delay <= dep\_delay - 30)

## # A tibble: 2,074 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 1716 1545 91 2140 2039  
## 2 2013 1 1 2205 1720 285 46 2040  
## 3 2013 1 1 2326 2130 116 131 18  
## 4 2013 1 3 1503 1221 162 1803 1555  
## 5 2013 1 3 1821 1530 171 2131 1910  
## 6 2013 1 3 1839 1700 99 2056 1950  
## 7 2013 1 3 1850 1745 65 2148 2120  
## 8 2013 1 3 1923 1815 68 2036 1958  
## 9 2013 1 3 1941 1759 102 2246 2139  
## 10 2013 1 3 1950 1845 65 2228 2227  
## # ℹ 2,064 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

Find all flights that departed between midnight and 6am (inclusive).

filter(flights, dep\_time <= 600)

## # A tibble: 9,344 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 542 540 2 923 850  
## 4 2013 1 1 544 545 -1 1004 1022  
## 5 2013 1 1 554 600 -6 812 837  
## 6 2013 1 1 554 558 -4 740 728  
## 7 2013 1 1 555 600 -5 913 854  
## 8 2013 1 1 557 600 -3 709 723  
## 9 2013 1 1 557 600 -3 838 846  
## 10 2013 1 1 558 600 -2 753 745  
## # ℹ 9,334 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

# Exercise 3

How many flights have a missing dep\_time? What other variables are missing? What might these rows represent?

filter(flights, is.na(dep\_time))

## # A tibble: 8,255 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 NA 1630 NA NA 1815  
## 2 2013 1 1 NA 1935 NA NA 2240  
## 3 2013 1 1 NA 1500 NA NA 1825  
## 4 2013 1 1 NA 600 NA NA 901  
## 5 2013 1 2 NA 1540 NA NA 1747  
## 6 2013 1 2 NA 1620 NA NA 1746  
## 7 2013 1 2 NA 1355 NA NA 1459  
## 8 2013 1 2 NA 1420 NA NA 1644  
## 9 2013 1 2 NA 1321 NA NA 1536  
## 10 2013 1 2 NA 1545 NA NA 1910  
## # ℹ 8,245 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

8,255 flights do not have a departure time. These flights are also missing departure delays and arrival times. Since these flights never departed, they were never delayed and never arrived.

# Section 5.3.1

# Exercise 2

Sort flights to find the most delayed flights.

arrange(flights, desc(dep\_delay))

## # A tibble: 336,776 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 9 641 900 1301 1242 1530  
## 2 2013 6 15 1432 1935 1137 1607 2120  
## 3 2013 1 10 1121 1635 1126 1239 1810  
## 4 2013 9 20 1139 1845 1014 1457 2210  
## 5 2013 7 22 845 1600 1005 1044 1815  
## 6 2013 4 10 1100 1900 960 1342 2211  
## 7 2013 3 17 2321 810 911 135 1020  
## 8 2013 6 27 959 1900 899 1236 2226  
## 9 2013 7 22 2257 759 898 121 1026  
## 10 2013 12 5 756 1700 896 1058 2020  
## # ℹ 336,766 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

Find the flights that left earliest.

arrange(flights, dep\_delay)

## # A tibble: 336,776 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 12 7 2040 2123 -43 40 2352  
## 2 2013 2 3 2022 2055 -33 2240 2338  
## 3 2013 11 10 1408 1440 -32 1549 1559  
## 4 2013 1 11 1900 1930 -30 2233 2243  
## 5 2013 1 29 1703 1730 -27 1947 1957  
## 6 2013 8 9 729 755 -26 1002 955  
## 7 2013 10 23 1907 1932 -25 2143 2143  
## 8 2013 3 30 2030 2055 -25 2213 2250  
## 9 2013 3 2 1431 1455 -24 1601 1631  
## 10 2013 5 5 934 958 -24 1225 1309  
## # ℹ 336,766 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

# Exercise 4

Which flights traveled the farthest?

arrange(flights, desc(distance))

## # A tibble: 336,776 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 857 900 -3 1516 1530  
## 2 2013 1 2 909 900 9 1525 1530  
## 3 2013 1 3 914 900 14 1504 1530  
## 4 2013 1 4 900 900 0 1516 1530  
## 5 2013 1 5 858 900 -2 1519 1530  
## 6 2013 1 6 1019 900 79 1558 1530  
## 7 2013 1 7 1042 900 102 1620 1530  
## 8 2013 1 8 901 900 1 1504 1530  
## 9 2013 1 9 641 900 1301 1242 1530  
## 10 2013 1 10 859 900 -1 1449 1530  
## # ℹ 336,766 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

Which traveled the shortest?

arrange(flights, distance)

## # A tibble: 336,776 × 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 7 27 NA 106 NA NA 245  
## 2 2013 1 3 2127 2129 -2 2222 2224  
## 3 2013 1 4 1240 1200 40 1333 1306  
## 4 2013 1 4 1829 1615 134 1937 1721  
## 5 2013 1 4 2128 2129 -1 2218 2224  
## 6 2013 1 5 1155 1200 -5 1241 1306  
## 7 2013 1 6 2125 2129 -4 2224 2224  
## 8 2013 1 7 2124 2129 -5 2212 2224  
## 9 2013 1 8 2127 2130 -3 2304 2225  
## 10 2013 1 9 2126 2129 -3 2217 2224  
## # ℹ 336,766 more rows  
## # ℹ 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

# Section 5.4.1

# Exercise 2

What happens if you include the name of a variable multiple times in a select() call?

select(flights, carrier)

## # A tibble: 336,776 × 1  
## carrier  
## <chr>   
## 1 UA   
## 2 UA   
## 3 AA   
## 4 B6   
## 5 DL   
## 6 UA   
## 7 B6   
## 8 EV   
## 9 B6   
## 10 AA   
## # ℹ 336,766 more rows

This is the result of the select statement for flights on carrier.

select(flights, carrier, carrier)

## # A tibble: 336,776 × 1  
## carrier  
## <chr>   
## 1 UA   
## 2 UA   
## 3 AA   
## 4 B6   
## 5 DL   
## 6 UA   
## 7 B6   
## 8 EV   
## 9 B6   
## 10 AA   
## # ℹ 336,766 more rows

As shown above, including the same variable multiple times does not affect the results.

# Exercise 3

What does the any\_of() function do? Why might it be helpful in conjunction with this vector?

vars <- c("year", "month", "day", "dep\_delay", "arr\_delay")

The any\_of() function allows you to specify all the variables in a vector.

select(flights, any\_of(vars))

## # A tibble: 336,776 × 5  
## year month day dep\_delay arr\_delay  
## <int> <int> <int> <dbl> <dbl>  
## 1 2013 1 1 2 11  
## 2 2013 1 1 4 20  
## 3 2013 1 1 2 33  
## 4 2013 1 1 -1 -18  
## 5 2013 1 1 -6 -25  
## 6 2013 1 1 -4 12  
## 7 2013 1 1 -5 19  
## 8 2013 1 1 -3 -14  
## 9 2013 1 1 -3 -8  
## 10 2013 1 1 -2 8  
## # ℹ 336,766 more rows

When the any\_of() function is applied to the previous vector in a select statement, it gives the results shown above without needing to type everything out again.

# Section 5.5.2

# Exercise 1

Currently dep\_time and sched\_dep\_time are convenient to look at, but hard to compute with because they’re not really continuous numbers. Convert them to a more convenient representation of number of minutes since midnight.

flights1 <- flights  
transmute(flights1, dep\_time = 60 \* dep\_time %/% 100 + dep\_time %% 100, sched\_dep\_time = 60 \* sched\_dep\_time %/% 100 + sched\_dep\_time %% 100)

## # A tibble: 336,776 × 2  
## dep\_time sched\_dep\_time  
## <dbl> <dbl>  
## 1 317 315  
## 2 333 329  
## 3 342 340  
## 4 344 345  
## 5 354 360  
## 6 354 358  
## 7 355 360  
## 8 357 360  
## 9 357 360  
## 10 358 360  
## # ℹ 336,766 more rows

The above formula can be used to convert dep\_time and sched\_dep\_time to minutes.

# Exercise 3

Compare dep\_time, sched\_dep\_time, and dep\_delay. How would you expect those three numbers to be related?

flights1 <- flights  
transmute(flights1, dep\_time, sched\_dep\_time, dep\_delay, dep\_time - sched\_dep\_time)

## # A tibble: 336,776 × 4  
## dep\_time sched\_dep\_time dep\_delay `dep\_time - sched\_dep\_time`  
## <int> <int> <dbl> <int>  
## 1 517 515 2 2  
## 2 533 529 4 4  
## 3 542 540 2 2  
## 4 544 545 -1 -1  
## 5 554 600 -6 -46  
## 6 554 558 -4 -4  
## 7 555 600 -5 -45  
## 8 557 600 -3 -43  
## 9 557 600 -3 -43  
## 10 558 600 -2 -42  
## # ℹ 336,766 more rows

You would expect dep\_delay to always match dep\_time - sched\_dep\_time. However, this is not the case because dep\_time and sched\_dep\_time are not listed in minutes.

flights1 <- flights  
transmute(flights1, dep\_time = 60 \* dep\_time %/% 100 + dep\_time %% 100, sched\_dep\_time = 60 \* sched\_dep\_time %/% 100 + sched\_dep\_time %% 100, dep\_delay, dep\_time - sched\_dep\_time)

## # A tibble: 336,776 × 4  
## dep\_time sched\_dep\_time dep\_delay `dep\_time - sched\_dep\_time`  
## <dbl> <dbl> <dbl> <dbl>  
## 1 317 315 2 2  
## 2 333 329 4 4  
## 3 342 340 2 2  
## 4 344 345 -1 -1  
## 5 354 360 -6 -6  
## 6 354 358 -4 -4  
## 7 355 360 -5 -5  
## 8 357 360 -3 -3  
## 9 357 360 -3 -3  
## 10 358 360 -2 -2  
## # ℹ 336,766 more rows

The above formula fixes this issue by converting dep\_time and sched\_dep\_time to minutes.

# Section 5.6.7

# Exercise 3

Our definition of cancelled flights (is.na(dep\_delay) | is.na(arr\_delay) ) is slightly suboptimal. Why? Which is the most important column?

reframe(flights, is.na(dep\_delay) | is.na(arr\_delay))

## # A tibble: 336,776 × 1  
## `is.na(dep\_delay) | is.na(arr\_delay)`  
## <lgl>   
## 1 FALSE   
## 2 FALSE   
## 3 FALSE   
## 4 FALSE   
## 5 FALSE   
## 6 FALSE   
## 7 FALSE   
## 8 FALSE   
## 9 FALSE   
## 10 FALSE   
## # ℹ 336,766 more rows

It is unnecessary to check if both dep\_delay and arr\_delay are null because if dep\_delay is null then arr\_delay will also be null since a flight that never departed cannot arrive.

reframe(flights, is.na(dep\_delay))

## # A tibble: 336,776 × 1  
## `is.na(dep\_delay)`  
## <lgl>   
## 1 FALSE   
## 2 FALSE   
## 3 FALSE   
## 4 FALSE   
## 5 FALSE   
## 6 FALSE   
## 7 FALSE   
## 8 FALSE   
## 9 FALSE   
## 10 FALSE   
## # ℹ 336,766 more rows

It is only necessary to check if dep\_delay is null as this will give the same result.

# Exercise 6

What does the sort argument to count() do. When might you use it?

flights %>% count(dest)

## # A tibble: 105 × 2  
## dest n  
## <chr> <int>  
## 1 ABQ 254  
## 2 ACK 265  
## 3 ALB 439  
## 4 ANC 8  
## 5 ATL 17215  
## 6 AUS 2439  
## 7 AVL 275  
## 8 BDL 443  
## 9 BGR 375  
## 10 BHM 297  
## # ℹ 95 more rows

Count will default to sorting by alphabetical order.

flights %>% count(dest, sort = TRUE)

## # A tibble: 105 × 2  
## dest n  
## <chr> <int>  
## 1 ORD 17283  
## 2 ATL 17215  
## 3 LAX 16174  
## 4 BOS 15508  
## 5 MCO 14082  
## 6 CLT 14064  
## 7 SFO 13331  
## 8 FLL 12055  
## 9 MIA 11728  
## 10 DCA 9705  
## # ℹ 95 more rows

After adding the “sort = TRUE” argument, count will sort in numerical order. This is useful if you want to see what the most common occurrences are.

# Section 5.7.1

# Exercise 2

Which plane (tailnum) has the worst on-time record?

flights %>% filter(arr\_delay > 0) %>% count(tailnum, sort = TRUE)

## # A tibble: 3,874 × 2  
## tailnum n  
## <chr> <int>  
## 1 N725MQ 215  
## 2 N228JB 192  
## 3 N258JB 191  
## 4 N713MQ 185  
## 5 N711MQ 184  
## 6 N723MQ 180  
## 7 N531MQ 174  
## 8 N190JB 173  
## 9 N353JB 173  
## 10 N534MQ 172  
## # ℹ 3,864 more rows

The plane with tailnum N725MQ has the worst on-time record.

# Exercise 3

What time of day should you fly if you want to avoid delays as much as possible?

flights %>% filter(dep\_delay < 0) %>% count(sched\_dep\_time, sort = TRUE)

## # A tibble: 1,008 × 2  
## sched\_dep\_time n  
## <int> <int>  
## 1 600 5097  
## 2 700 3760  
## 3 630 3575  
## 4 900 3399  
## 5 1200 3090  
## 6 800 2856  
## 7 1700 2305  
## 8 1300 2289  
## 9 1000 2138  
## 10 1600 2107  
## # ℹ 998 more rows

You should fly at 6am if you want to avoid delays as much as possible.

# Midwest Demographics

This data set contains data on the demographics of each county in five midwestern states: Illinois, Indiana, Michigan, Ohio, and Wisconsin. The following table shows the top ten most populous counties in these states along with what state they are in, their population, and whether or not they are in a metropolitan area.

select(arrange(midwest, desc(poptotal)), county, state, poptotal, inmetro)

## # A tibble: 437 × 4  
## county state poptotal inmetro  
## <chr> <chr> <int> <int>  
## 1 COOK IL 5105067 1  
## 2 WAYNE MI 2111687 1  
## 3 CUYAHOGA OH 1412140 1  
## 4 OAKLAND MI 1083592 1  
## 5 FRANKLIN OH 961437 1  
## 6 MILWAUKEE WI 959275 1  
## 7 HAMILTON OH 866228 1  
## 8 MARION IN 797159 1  
## 9 DU PAGE IL 781666 1  
## 10 MACOMB MI 717400 1  
## # ℹ 427 more rows

When looking at this plot, it is clear that the most populated and most densely populated counties are all in metropolitan areas. This makes sense because metropolitan areas are centered around large cities with large populations. If a county is not close enough to one of these cities to be in a metropolitan area, it will probably not have a large population. To get a closer look at this, the following table shows these same counties along with their county seat and what metropolitan and combined statistical areas they are in.

| County | County Seat | Metro Area | CSA |
| --- | --- | --- | --- |
| Cook | Chicago | Chicago | Chicago |
| Wayne | Detroit | Detroit | Detroit |
| Cuyahoga | Cleveland | Cleveland | Cleveland |
| Oakland | Pontiac | Detroit | Detroit |
| Franklin | Columbus | Columbus | Columbus |
| Milwaukee | Milwaukee | Milwaukee | Milwaukee |
| Hamilton | Cincinnati | Cincinnati | Cincinnati |
| Marion | Indianapolis | Indianapolis | Indianapolis |
| Du Page | Wheaton | Chicago | Chicago |
| Macomb | Mt. Clemens | Detroit | Detroit |

As shown in the above table, half of the ten most populous counties in these states fall into either the Chicago or Detroit Metropolitan Areas. This makes sense as these are the two most populous metropolitan areas in the region. However, the other half of these counties each falls into a different metropolitan area. To get a closer look at this divide, we will look at each of these five states individually.

# Illinois

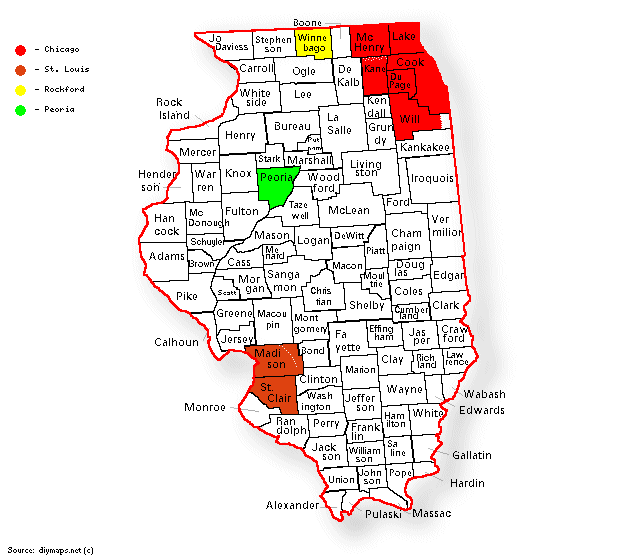
select(arrange(filter(midwest, state == "IL"), desc(poptotal)), county, poptotal, inmetro)

## # A tibble: 102 × 3  
## county poptotal inmetro  
## <chr> <int> <int>  
## 1 COOK 5105067 1  
## 2 DU PAGE 781666 1  
## 3 LAKE 516418 1  
## 4 WILL 357313 1  
## 5 KANE 317471 1  
## 6 ST CLAIR 262852 1  
## 7 Winnebago 252913 1  
## 8 MADISON 249238 1  
## 9 MCHENRY 183241 1  
## 10 PEORIA 182827 1  
## # ℹ 92 more rows

As shown in the above table, the ten most populous counties in Illinois are all in a metropolitan area. A breakdown of these ten counties can be seen in the following table.

| County | County Seat | Metro Area | CSA |
| --- | --- | --- | --- |
| Cook | Chicago | Chicago | Chicago |
| Du Page | Wheaton | Chicago | Chicago |
| Lake | Waukegan | Chicago | Chicago |
| Will | Joliet | Chicago | Chicago |
| Kane | Geneva | Chicago | Chicago |
| St. Clair | Belleville | St. Louis | St. Louis |
| Winnebago | Rockford | Rockford | Rockford |
| Madison | Edwardsville | St. Louis | St. Louis |
| McHenry | Woodstock | Chicago | Chicago |
| Peoria | Peoria | Peoria | Peoria |

As shown in the above table, six of the ten most populous counties in Illinois, including all of the top five, are a part of the Chicago Metropolitan Area. These six counties, along with three more in other states, make Chicago the most common metropolitan area among the fifty counties listed. Interestingly, two of the top ten most populous counties in Illinois are a part of the St. Louis Metropolitan Area. These are the only two counties in the fifty counties listed to be a part of a metropolitan area anchored outside of these five states. This makes sense because St. Louis borders Illinois and is larger than every metropolitan area in the region besides the previously mentioned Chicago and Detroit Metropolitan Areas.



Illinois Counties

# Indiana

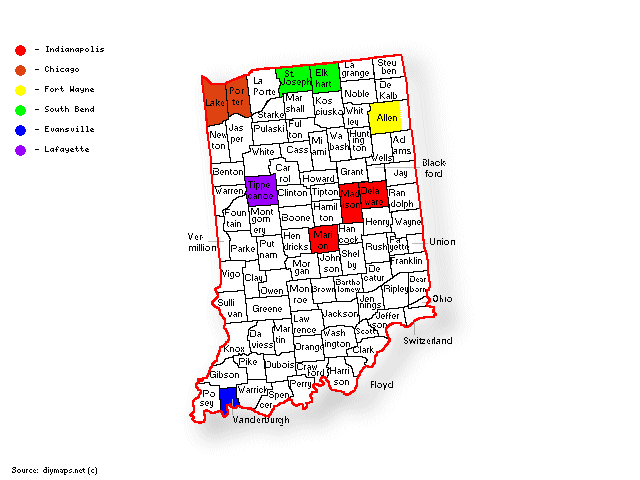
select(arrange(filter(midwest, state == "IN"), desc(poptotal)), county, poptotal, inmetro)

## # A tibble: 92 × 3  
## county poptotal inmetro  
## <chr> <int> <int>  
## 1 MARION 797159 1  
## 2 LAKE 475594 1  
## 3 ALLEN 300836 1  
## 4 ST JOSEPH 247052 1  
## 5 VANDERBURGH 165058 1  
## 6 ELKHART 156198 1  
## 7 MADISON 130669 1  
## 8 TIPPECANOE 130598 1  
## 9 PORTER 128932 1  
## 10 DELAWARE 119659 1  
## # ℹ 82 more rows

As shown in the above table, the ten most populous counties in Indiana are all in a metropolitan area. A breakdown of these ten counties can be seen in the following table.

| County | County Seat | Metro Area | CSA |
| --- | --- | --- | --- |
| Marion | Indianapolis | Indianapolis | Indianapolis |
| Lake | Crown Point | Chicago | Chicago |
| Allen | Fort Wayne | Fort Wayne | Fort Wayne |
| St. Joseph | South Bend | South Bend | South Bend |
| Vanderburgh | Evansville | Evansville | Evansville |
| Elkhart | Goshen | Elkhart | South Bend |
| Madison | Anderson | Indianapolis | Indianapolis |
| Tippecanoe | Lafayette | Lafayette | Lafayette |
| Porter | Valparaiso | Chicago | Chicago |
| Delaware | Muncie | Muncie | Indianapolis |

As shown in the table above, two of the ten most populous counties in Indiana are a part of the Indianapolis Metropolitan Area with a third being included in the Indianapolis Combined Statistical Area. This makes sense as Indianapolis is the capital and largest city in Indiana. The Chicago Metropolitan Area also includes two of the ten most populous counties in Indiana despite being anchored in a different state. Although none of the other counties share a metropolitan area, St. Joseph and Elkhart counties do share the South Bend Combined Statistical Area.



Indiana Counties

# Michigan

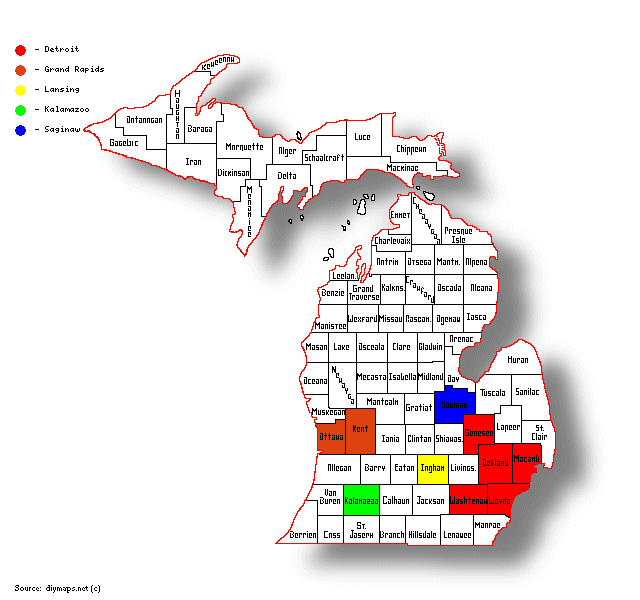
select(arrange(filter(midwest, state == "MI"), desc(poptotal)), county, poptotal, inmetro)

## # A tibble: 83 × 3  
## county poptotal inmetro  
## <chr> <int> <int>  
## 1 WAYNE 2111687 1  
## 2 OAKLAND 1083592 1  
## 3 MACOMB 717400 1  
## 4 KENT 500631 1  
## 5 GENESEE 430459 1  
## 6 WASHTENAW 282937 1  
## 7 INGHAM 281912 1  
## 8 KALAMAZOO 223411 1  
## 9 SAGINAW 211946 1  
## 10 OTTAWA 187768 1  
## # ℹ 73 more rows

As shown in the above table, the ten most populous counties in Michigan are all in a metropolitan area. A breakdown of these ten counties can be seen in the following table.

| County | County Seat | Metro Area | CSA |
| --- | --- | --- | --- |
| Wayne | Detroit | Detroit | Detroit |
| Oakland | Pontiac | Detroit | Detroit |
| Macomb | Mt. Clemens | Detroit | Detroit |
| Kent | Grand Rapids | Grand Rapids | Grand Rapids |
| Genesee | Flint | Flint | Detroit |
| Washtenaw | Ann Arbor | Ann Arbor | Detroit |
| Ingham | Mason | Lansing | Lansing |
| Kalamazoo | Kalamazoo | Kalamazoo | Kalamazoo |
| Saginaw | Saginaw | Saginaw | Saginaw |
| Ottawa | Grand Haven | Grand Rapids | Grand Rapids |

As shown in the table above, the top three most populous counties in Michigan are a part of the Detroit Metropolitan Area with two more being included in the Detroit Metropolitan Area. This makes sense as Detroit is the largest city in the state. The Grand Rapids Metropolitan Area also includes two of the ten most populous counties in Michigan.



Michigan Counties

# Ohio

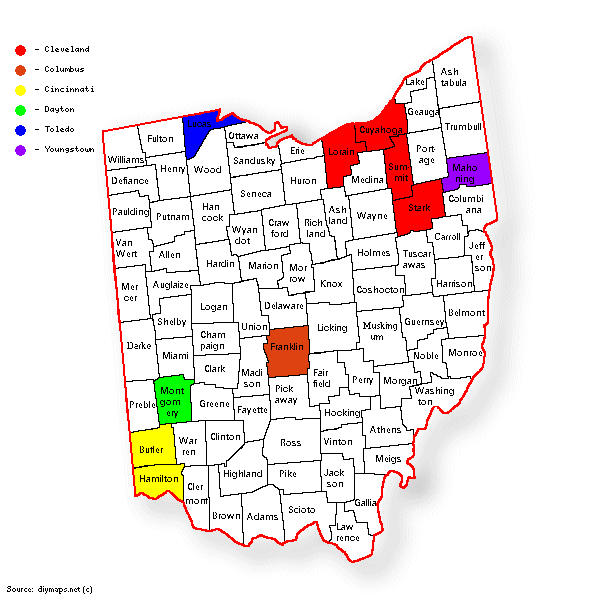
select(arrange(filter(midwest, state == "OH"), desc(poptotal)), county, poptotal, inmetro)

## # A tibble: 88 × 3  
## county poptotal inmetro  
## <chr> <int> <int>  
## 1 CUYAHOGA 1412140 1  
## 2 FRANKLIN 961437 1  
## 3 HAMILTON 866228 1  
## 4 MONTGOMERY 573809 1  
## 5 SUMMIT 514990 1  
## 6 LUCAS 462361 1  
## 7 STARK 367585 1  
## 8 BUTLER 291479 1  
## 9 LORAIN 271126 1  
## 10 MAHONING 264806 1  
## # ℹ 78 more rows

As shown in the above table, the ten most populous counties in Ohio are all in a metropolitan area. A breakdown of these ten counties can be seen in the following table.

| County | County Seat | Metro Area | CSA |
| --- | --- | --- | --- |
| Cuyahoga | Cleveland | Cleveland | Cleveland |
| Franklin | Columbus | Columbus | Columbus |
| Hamilton | Cincinnati | Cincinnati | Cincinnati |
| Montgomery | Dayton | Dayton | Dayton |
| Summit | Akron | Akron | Cleveland |
| Lucas | Toledo | Toledo | Toledo |
| Stark | Canton | Canton | Cleveland |
| Butler | Hamilton | Cincinnati | Cincinnati |
| Lorain | Elyria | Cleveland | Cleveland |
| Mahoning | Youngstown | Youngstown | Youngstown |

As shown in the table above, two of the ten most populous counties in Ohio are a part of the Cleveland Metropolitan Area with two more being included in the Cleveland Combined Statistical Area. This makes sense as the Cleveland Combined Statistical Area is the largest combined statistical area in Ohio. The Cincinnati Metropolitan Area also includes two of the ten most populous counties in Ohio.



Ohio Counties

# Wisconsin

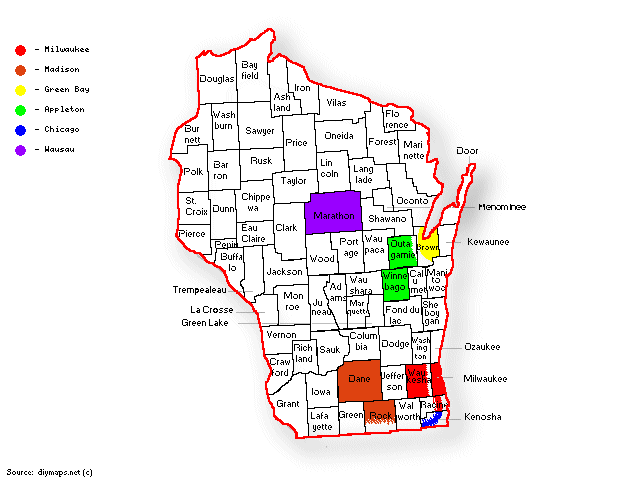
select(arrange(filter(midwest, state == "WI"), desc(poptotal)), county, poptotal, inmetro)

## # A tibble: 72 × 3  
## county poptotal inmetro  
## <chr> <int> <int>  
## 1 MILWAUKEE 959275 1  
## 2 DANE 367085 1  
## 3 WAUKESHA 304715 1  
## 4 BROWN 194594 1  
## 5 RACINE 175034 1  
## 6 OUTAGAMIE 140510 1  
## 7 WINNEBAGO 140320 1  
## 8 ROCK 139510 1  
## 9 KENOSHA 128181 1  
## 10 MARATHON 115400 1  
## # ℹ 62 more rows

As shown in the above table, the ten most populous counties in Wisconsin are all in a metropolitan area. A breakdown of these ten counties can be seen in the following table.

| County | County Seat | Metro Area | CSA |
| --- | --- | --- | --- |
| Milwaukee | Milwaukee | Milwaukee | Milwaukee |
| Dane | Madison | Madison | Madison |
| Waukesha | Waukesha | Milwaukee | Milwaukee |
| Brown | Green Bay | Green Bay | Green Bay |
| Racine | Racine | Racine | Milwaukee |
| Outagamie | Appleton | Appleton | Appleton |
| Winnebago | Oshkosh | Oshkosh | Appleton |
| Rock | Janesville | Janesville | Madison |
| Kenosha | Kenosha | Chicago | Chicago |
| Marathon | Wausau | Wausau | Wausau |

As shown in the table above, two of the ten most populous counties in Wisconsin are a part of the Milwaukee Metropolitan Area with a third being included in the Milwaukee Combined Statistical Area. This makes sense as Milwaukee is the largest city in the state. Although none of the other counties share a metropolitan area, the Madison and Appleton Combined Statistical Areas each include two of the ten most populous counties in Wisconsin. Interestingly, Kenosha County is a part of the Chicago Metropolitan Area which makes sense as it borders Illinois.



Wisconsin Counties

# Conclusion

Based on this data, it is safe to say that the most populous counties tend to be in the largest metropolitan areas. This is especially evident in Illinois and Michigan which are dominated by the Chicago and Detroit Metropolitan Areas respectively. However, the other states included here have more spread out populations. It is interesting to see this divide between different states, and it could be worth looking into how this plays out in more states.