parker\_dsc520\_week4

2022-12-19

Test Scores A professor has recently taught two sections of the same course with only one difference between the sections. In one section, he used only examples taken from sports applications, and in the other section, he used examples taken from a variety of application areas. The sports themed section was advertised as such; so students knew which type of section they were enrolling in. The professor has asked you to compare student performance in the two sections using course grades and total points earned in the course. You will need to import the Scores.csv dataset that has been provided for you. Use the appropriate R functions to answer the following questions: What are the observational units in this study?

Identify the variables mentioned in the narrative paragraph and determine which are categorical and quantitative?

sports applications:categorical  
application areas: categorical course grades: quantitative total points: quantitative

Create one variable to hold a subset of your data set that contains only the Regular Section and one variable for the Sports Section.

sports = subset(score\_data,Section == "Sports")  
   
regular = subset(score\_data,Section == "Regular")  
print(sports)

## # A tibble: 19 × 3  
## Count Score Section  
## <dbl> <dbl> <chr>   
## 1 10 200 Sports   
## 2 10 205 Sports   
## 3 20 235 Sports   
## 4 10 240 Sports   
## 5 10 250 Sports   
## 6 30 285 Sports   
## 7 20 300 Sports   
## 8 10 305 Sports   
## 9 10 310 Sports   
## 10 10 315 Sports   
## 11 10 325 Sports   
## 12 10 330 Sports   
## 13 30 335 Sports   
## 14 10 340 Sports   
## 15 10 360 Sports   
## 16 20 365 Sports   
## 17 10 370 Sports   
## 18 10 375 Sports   
## 19 10 395 Sports

print(regular)

## # A tibble: 19 × 3  
## Count Score Section  
## <dbl> <dbl> <chr>   
## 1 10 265 Regular  
## 2 10 275 Regular  
## 3 10 295 Regular  
## 4 10 300 Regular  
## 5 10 305 Regular  
## 6 10 310 Regular  
## 7 20 320 Regular  
## 8 10 305 Regular  
## 9 20 320 Regular  
## 10 10 325 Regular  
## 11 20 330 Regular  
## 12 10 335 Regular  
## 13 20 340 Regular  
## 14 30 350 Regular  
## 15 20 360 Regular  
## 16 20 365 Regular  
## 17 10 370 Regular  
## 18 20 375 Regular  
## 19 20 380 Regular

Use the Plot function to plot each Sections scores and the number of students achieving that score. Use additional Plot Arguments to label the graph and give each axis an appropriate label. Once you have produced your Plots answer the following questions:

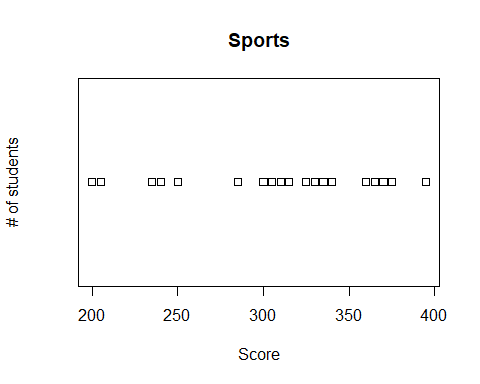
sports\_scores = sports[,2]  
regular\_scores = regular[,2]  
#regular\_scores <- regular["Score"]  
print(sports\_scores)

## # A tibble: 19 × 1  
## Score  
## <dbl>  
## 1 200  
## 2 205  
## 3 235  
## 4 240  
## 5 250  
## 6 285  
## 7 300  
## 8 305  
## 9 310  
## 10 315  
## 11 325  
## 12 330  
## 13 335  
## 14 340  
## 15 360  
## 16 365  
## 17 370  
## 18 375  
## 19 395

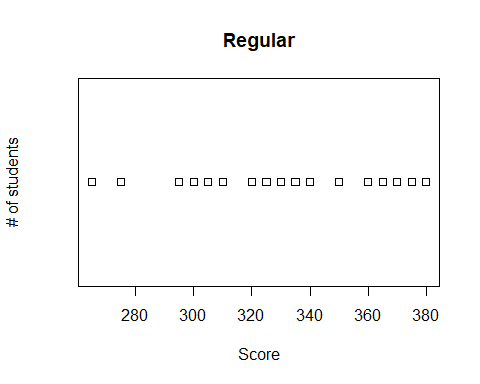
print(regular\_scores)

## # A tibble: 19 × 1  
## Score  
## <dbl>  
## 1 265  
## 2 275  
## 3 295  
## 4 300  
## 5 305  
## 6 310  
## 7 320  
## 8 305  
## 9 320  
## 10 325  
## 11 330  
## 12 335  
## 13 340  
## 14 350  
## 15 360  
## 16 365  
## 17 370  
## 18 375  
## 19 380

#ggplot(sports\_scores,aes(x = "students",y="score")) + geom\_point()  
  
#ggplot(sports\_scores,aes(sample = y)) + stat\_qq()  
plot(sports\_scores, xlab="Score", ylab="# of students", main="Sports")



plot(regular\_scores, xlab="Score", ylab="# of students", main="Regular")



#I tried several different graphs here but could not get it to turn out how it should. I need to figure out how to swap the x and y data and that should resolve the biggest issue. Attempted to ask other students but could not get help either. I have found what the graphs should look like so I can still answer the rest of the questions.

Comparing and contrasting the point distributions between the two section, looking at both tendency and consistency: Can you say that one section tended to score more points than the other? Justify and explain your answer.

#It appears that the sports section has a higher overall points. It reaches a total of 395 while the regular section only hits 380. However the sports section also has the lowest points at 200, while the regular only goes to 260.

Did every student in one section score more points than every student in the other section? If not, explain what a statistical tendency means in this context.

#No, the students in the sports section scored the highest and lowest scores. If you look at the mean, the regular students actually had a higher overall average. The medial score is also higher among the regular students. So, even though the sports section had the highest score, they didn't have the highest overall score.   
  
s\_mean <- mean(sports[["Score"]], trim = 0, na.rm = FALSE)  
r\_mean <- mean(regular[["Score"]], trim = 0, na.rm = FALSE)  
  
print(s\_mean)

## [1] 307.3684

print(r\_mean)

## [1] 327.6316

s\_median <- median(sports[["Score"]], trim = 0, na.rm = FALSE)  
r\_median <- median(regular[["Score"]], trim = 0, na.rm = FALSE)  
  
print(s\_median)

## [1] 315

print(r\_median)

## [1] 325

What could be one additional variable that was not mentioned in the narrative that could be influencing the point distributions between the two sections?

#one thing we didn’t look at was the age of the students. If the students in one section are older, than they might have a higher chance of getting a better score.

We interact with a few datasets in this course, one you are already familiar with, the 2014 American Community Survey and the second is a Housing dataset, that provides real estate transactions recorded from 1964 to 2016. For this exercise, you need to start practicing some data transformation steps – which will carry into next week, as you learn some additional methods. For this week, using either dataset (or one of your own – although I will let you know ahead of time that the Housing dataset is used for a later assignment, so not a bad idea for you to get more comfortable with now!), perform the following data transformations:

setwd("C:/Users/brean/OneDrive/Desktop/NucampFolder/projects/dsc520-1")  
housing\_data = read\_excel("data/week-7-housing.xlsx")  
print(housing\_data)

## # A tibble: 12,865 × 24  
## `Sale Date` Sale Pric…¹ sale\_…² sale\_…³ sale\_…⁴ sitet…⁵ addr\_…⁶ zip5  
## <dttm> <dbl> <dbl> <dbl> <chr> <chr> <chr> <dbl>  
## 1 2006-01-03 00:00:00 698000 1 3 <NA> R1 17021 … 98052  
## 2 2006-01-03 00:00:00 649990 1 3 <NA> R1 11927 … 98052  
## 3 2006-01-03 00:00:00 572500 1 3 <NA> R1 13315 … 98052  
## 4 2006-01-03 00:00:00 420000 1 3 <NA> R1 3303 1… 98052  
## 5 2006-01-03 00:00:00 369900 1 3 15 R1 16126 … 98052  
## 6 2006-01-03 00:00:00 184667 1 15 18 51 R1 8101 2… 98053  
## 7 2006-01-04 00:00:00 1050000 1 3 <NA> R1 21634 … 98053  
## 8 2006-01-04 00:00:00 875000 1 3 <NA> R1 21404 … 98053  
## 9 2006-01-04 00:00:00 660000 1 3 <NA> R1 7525 2… 98053  
## 10 2006-01-04 00:00:00 650000 1 3 <NA> R1 17703 … 98052  
## # … with 12,855 more rows, 16 more variables: ctyname <chr>, postalctyn <chr>,  
## # lon <dbl>, lat <dbl>, building\_grade <dbl>, square\_feet\_total\_living <dbl>,  
## # bedrooms <dbl>, bath\_full\_count <dbl>, bath\_half\_count <dbl>,  
## # bath\_3qtr\_count <dbl>, year\_built <dbl>, year\_renovated <dbl>,  
## # current\_zoning <chr>, sq\_ft\_lot <dbl>, prop\_type <chr>, present\_use <dbl>,  
## # and abbreviated variable names ¹​`Sale Price`, ²​sale\_reason,  
## # ³​sale\_instrument, ⁴​sale\_warning, ⁵​sitetype, ⁶​addr\_full

Use the apply function on a variable in your dataset

apply(housing\_data[14], 2, sum)

## square\_feet\_total\_living   
## 32670747

Use the aggregate function on a variable in your dataset

aggregate(building\_grade ~ year\_built, data = housing\_data, mean)

## year\_built building\_grade  
## 1 1900 6.500000  
## 2 1903 6.000000  
## 3 1905 9.000000  
## 4 1906 7.000000  
## 5 1909 5.000000  
## 6 1910 5.000000  
## 7 1912 7.000000  
## 8 1913 6.500000  
## 9 1914 8.000000  
## 10 1915 5.000000  
## 11 1916 7.000000  
## 12 1918 6.666667  
## 13 1919 6.000000  
## 14 1920 6.500000  
## 15 1922 6.750000  
## 16 1923 5.000000  
## 17 1924 6.750000  
## 18 1925 7.000000  
## 19 1926 6.333333  
## 20 1927 8.250000  
## 21 1928 7.000000  
## 22 1929 8.000000  
## 23 1930 6.333333  
## 24 1931 5.000000  
## 25 1932 6.600000  
## 26 1933 6.333333  
## 27 1934 7.000000  
## 28 1935 5.666667  
## 29 1936 7.333333  
## 30 1937 6.333333  
## 31 1938 6.500000  
## 32 1939 5.500000  
## 33 1940 6.777778  
## 34 1941 6.200000  
## 35 1942 6.500000  
## 36 1943 6.400000  
## 37 1944 6.500000  
## 38 1945 4.750000  
## 39 1946 8.000000  
## 40 1947 6.111111  
## 41 1948 6.800000  
## 42 1949 6.750000  
## 43 1950 6.222222  
## 44 1951 7.833333  
## 45 1952 7.333333  
## 46 1953 7.000000  
## 47 1954 6.444444  
## 48 1955 7.250000  
## 49 1956 7.153846  
## 50 1957 7.538462  
## 51 1958 7.052632  
## 52 1959 7.304348  
## 53 1960 7.388889  
## 54 1961 7.214286  
## 55 1962 7.102041  
## 56 1963 7.310345  
## 57 1964 7.269231  
## 58 1965 7.382716  
## 59 1966 7.201439  
## 60 1967 7.290503  
## 61 1968 7.358491  
## 62 1969 7.212871  
## 63 1970 6.846154  
## 64 1971 7.222222  
## 65 1972 7.632075  
## 66 1973 7.750000  
## 67 1974 8.000000  
## 68 1975 7.785340  
## 69 1976 7.600000  
## 70 1977 7.639257  
## 71 1978 7.768456  
## 72 1979 7.941964  
## 73 1980 7.685714  
## 74 1981 7.792308  
## 75 1982 7.952381  
## 76 1983 7.942731  
## 77 1984 8.175074  
## 78 1985 8.485714  
## 79 1986 8.173469  
## 80 1987 8.633484  
## 81 1988 9.286517  
## 82 1989 9.497238  
## 83 1990 9.857143  
## 84 1991 9.381443  
## 85 1992 8.640351  
## 86 1993 9.175258  
## 87 1994 9.398148  
## 88 1995 8.916667  
## 89 1996 8.871166  
## 90 1997 9.054054  
## 91 1998 9.441176  
## 92 1999 9.887324  
## 93 2000 8.915789  
## 94 2001 8.178218  
## 95 2002 8.017467  
## 96 2003 8.252101  
## 97 2004 8.325301  
## 98 2005 8.447273  
## 99 2006 8.681583  
## 100 2007 8.579592  
## 101 2008 8.230552  
## 102 2009 8.038462  
## 103 2010 8.077731  
## 104 2011 8.275510  
## 105 2012 8.460411  
## 106 2013 8.435443  
## 107 2014 8.767857  
## 108 2015 8.834951  
## 109 2016 8.540984

Use the plyr function on a variable in your dataset – more specifically, I want to see you split some data, perform a modification to the data, and then bring it back together

library(dplyr)  
  
a <- select(housing\_data, square\_feet\_total\_living, bedrooms)  
  
b <- housing\_data %>% select(starts\_with('sale'))  
print(b)

## # A tibble: 12,865 × 5  
## `Sale Date` `Sale Price` sale\_reason sale\_instrument sale\_warning  
## <dttm> <dbl> <dbl> <dbl> <chr>   
## 1 2006-01-03 00:00:00 698000 1 3 <NA>   
## 2 2006-01-03 00:00:00 649990 1 3 <NA>   
## 3 2006-01-03 00:00:00 572500 1 3 <NA>   
## 4 2006-01-03 00:00:00 420000 1 3 <NA>   
## 5 2006-01-03 00:00:00 369900 1 3 15   
## 6 2006-01-03 00:00:00 184667 1 15 18 51   
## 7 2006-01-04 00:00:00 1050000 1 3 <NA>   
## 8 2006-01-04 00:00:00 875000 1 3 <NA>   
## 9 2006-01-04 00:00:00 660000 1 3 <NA>   
## 10 2006-01-04 00:00:00 650000 1 3 <NA>   
## # … with 12,855 more rows

colnames(b)[2] = "sale\_price"  
print(b)

## # A tibble: 12,865 × 5  
## `Sale Date` sale\_price sale\_reason sale\_instrument sale\_warning  
## <dttm> <dbl> <dbl> <dbl> <chr>   
## 1 2006-01-03 00:00:00 698000 1 3 <NA>   
## 2 2006-01-03 00:00:00 649990 1 3 <NA>   
## 3 2006-01-03 00:00:00 572500 1 3 <NA>   
## 4 2006-01-03 00:00:00 420000 1 3 <NA>   
## 5 2006-01-03 00:00:00 369900 1 3 15   
## 6 2006-01-03 00:00:00 184667 1 15 18 51   
## 7 2006-01-04 00:00:00 1050000 1 3 <NA>   
## 8 2006-01-04 00:00:00 875000 1 3 <NA>   
## 9 2006-01-04 00:00:00 660000 1 3 <NA>   
## 10 2006-01-04 00:00:00 650000 1 3 <NA>   
## # … with 12,855 more rows

colnames(housing\_data)[2] <- colnames(b)[2]  
  
print(housing\_data)

## # A tibble: 12,865 × 24  
## `Sale Date` sale\_price sale\_r…¹ sale\_…² sale\_…³ sitet…⁴ addr\_…⁵ zip5  
## <dttm> <dbl> <dbl> <dbl> <chr> <chr> <chr> <dbl>  
## 1 2006-01-03 00:00:00 698000 1 3 <NA> R1 17021 … 98052  
## 2 2006-01-03 00:00:00 649990 1 3 <NA> R1 11927 … 98052  
## 3 2006-01-03 00:00:00 572500 1 3 <NA> R1 13315 … 98052  
## 4 2006-01-03 00:00:00 420000 1 3 <NA> R1 3303 1… 98052  
## 5 2006-01-03 00:00:00 369900 1 3 15 R1 16126 … 98052  
## 6 2006-01-03 00:00:00 184667 1 15 18 51 R1 8101 2… 98053  
## 7 2006-01-04 00:00:00 1050000 1 3 <NA> R1 21634 … 98053  
## 8 2006-01-04 00:00:00 875000 1 3 <NA> R1 21404 … 98053  
## 9 2006-01-04 00:00:00 660000 1 3 <NA> R1 7525 2… 98053  
## 10 2006-01-04 00:00:00 650000 1 3 <NA> R1 17703 … 98052  
## # … with 12,855 more rows, 16 more variables: ctyname <chr>, postalctyn <chr>,  
## # lon <dbl>, lat <dbl>, building\_grade <dbl>, square\_feet\_total\_living <dbl>,  
## # bedrooms <dbl>, bath\_full\_count <dbl>, bath\_half\_count <dbl>,  
## # bath\_3qtr\_count <dbl>, year\_built <dbl>, year\_renovated <dbl>,  
## # current\_zoning <chr>, sq\_ft\_lot <dbl>, prop\_type <chr>, present\_use <dbl>,  
## # and abbreviated variable names ¹​sale\_reason, ²​sale\_instrument,  
## # ³​sale\_warning, ⁴​sitetype, ⁵​addr\_full

Check distributions of the data

stat.desc(housing\_data)

## Sale Date sale\_price sale\_reason sale\_instrument  
## nbr.val 1.286500e+04 1.286500e+04 1.286500e+04 1.286500e+04  
## nbr.null 0.000000e+00 0.000000e+00 2.000000e+00 3.000000e+00  
## nbr.na 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00  
## min 1.136246e+09 6.980000e+02 0.000000e+00 0.000000e+00  
## max 1.481846e+09 4.400000e+06 1.900000e+01 2.700000e+01  
## range 3.456000e+08 4.399302e+06 1.900000e+01 2.700000e+01  
## sum 1.687715e+13 8.500391e+09 1.994100e+04 4.731400e+04  
## median 1.321488e+09 5.930000e+05 1.000000e+00 3.000000e+00  
## mean 1.311866e+09 6.607377e+05 1.550019e+00 3.677730e+00  
## SE.mean 9.130786e+05 3.565217e+03 2.358588e-02 2.918881e-02  
## CI.mean.0.95 1.789770e+06 6.988354e+03 4.623183e-02 5.721441e-02  
## var 1.072571e+16 1.635241e+11 7.156721e+00 1.096081e+01  
## std.dev 1.035650e+08 4.043811e+05 2.675205e+00 3.310712e+00  
## coef.var 7.894483e-02 6.120145e-01 1.725917e+00 9.002051e-01  
## sale\_warning sitetype addr\_full zip5 ctyname postalctyn  
## nbr.val NA NA NA 1.286500e+04 NA NA  
## nbr.null NA NA NA 0.000000e+00 NA NA  
## nbr.na NA NA NA 0.000000e+00 NA NA  
## min NA NA NA 9.805200e+04 NA NA  
## max NA NA NA 9.807400e+04 NA NA  
## range NA NA NA 2.200000e+01 NA NA  
## sum NA NA NA 1.261446e+09 NA NA  
## median NA NA NA 9.805200e+04 NA NA  
## mean NA NA NA 9.805254e+04 NA NA  
## SE.mean NA NA NA 1.494488e-02 NA NA  
## CI.mean.0.95 NA NA NA 2.929417e-02 NA NA  
## var NA NA NA 2.873389e+00 NA NA  
## std.dev NA NA NA 1.695107e+00 NA NA  
## coef.var NA NA NA 1.728774e-05 NA NA  
## lon lat building\_grade square\_feet\_total\_living  
## nbr.val 1.286500e+04 1.286500e+04 1.286500e+04 1.286500e+04  
## nbr.null 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00  
## nbr.na 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00  
## min -1.221643e+02 4.745635e+01 2.000000e+00 2.400000e+02  
## max -1.219499e+02 4.773255e+01 1.300000e+01 1.354000e+04  
## range 2.144216e-01 2.761993e-01 1.100000e+01 1.330000e+04  
## sum -1.570549e+06 6.134492e+05 1.060130e+05 3.267075e+07  
## median -1.221003e+02 4.768742e+01 8.000000e+00 2.420000e+03  
## mean -1.220792e+02 4.768358e+01 8.240420e+00 2.539506e+03  
## SE.mean 4.603069e-04 2.271998e-04 9.633091e-03 8.726704e+00  
## CI.mean.0.95 9.022698e-04 4.453453e-04 1.888229e-02 1.710564e+01  
## var 2.725867e-03 6.640879e-04 1.193826e+00 9.797388e+05  
## std.dev 5.220984e-02 2.576990e-02 1.092624e+00 9.898176e+02  
## coef.var -4.276718e-04 5.404356e-04 1.325932e-01 3.897677e-01  
## bedrooms bath\_full\_count bath\_half\_count bath\_3qtr\_count  
## nbr.val 1.286500e+04 1.286500e+04 1.286500e+04 1.286500e+04  
## nbr.null 1.900000e+01 5.100000e+01 5.177000e+03 7.457000e+03  
## nbr.na 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00  
## min 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00  
## max 1.100000e+01 2.300000e+01 8.000000e+00 8.000000e+00  
## range 1.100000e+01 2.300000e+01 8.000000e+00 8.000000e+00  
## sum 4.475300e+04 2.313700e+04 7.891000e+03 6.355000e+03  
## median 4.000000e+00 2.000000e+00 1.000000e+00 0.000000e+00  
## mean 3.478663e+00 1.798445e+00 6.133696e-01 4.939759e-01  
## SE.mean 7.724356e-03 5.737733e-03 4.639903e-03 5.731102e-03  
## CI.mean.0.95 1.514088e-02 1.124681e-02 9.094899e-03 1.123381e-02  
## var 7.675990e-01 4.235361e-01 2.769668e-01 4.225578e-01  
## std.dev 8.761273e-01 6.507965e-01 5.262763e-01 6.500444e-01  
## coef.var 2.518575e-01 3.618662e-01 8.580085e-01 1.315944e+00  
## year\_built year\_renovated current\_zoning sq\_ft\_lot prop\_type  
## nbr.val 1.286500e+04 1.286500e+04 NA 1.286500e+04 NA  
## nbr.null 0.000000e+00 1.269600e+04 NA 0.000000e+00 NA  
## nbr.na 0.000000e+00 0.000000e+00 NA 0.000000e+00 NA  
## min 1.900000e+03 0.000000e+00 NA 7.850000e+02 NA  
## max 2.016000e+03 2.016000e+03 NA 1.631322e+06 NA  
## range 1.160000e+02 2.016000e+03 NA 1.630537e+06 NA  
## sum 2.563998e+07 3.376330e+05 NA 2.859705e+08 NA  
## median 1.998000e+03 0.000000e+00 NA 7.965000e+03 NA  
## mean 1.993003e+03 2.624431e+01 NA 2.222857e+04 NA  
## SE.mean 1.518212e-01 2.005595e+00 NA 5.019511e+02 NA  
## CI.mean.0.95 2.975921e-01 3.931264e+00 NA 9.838986e+02 NA  
## var 2.965342e+02 5.174832e+04 NA 3.241400e+09 NA  
## std.dev 1.722017e+01 2.274826e+02 NA 5.693329e+04 NA  
## coef.var 8.640314e-03 8.667883e+00 NA 2.561267e+00 NA  
## present\_use  
## nbr.val 1.286500e+04  
## nbr.null 9.000000e+00  
## nbr.na 0.000000e+00  
## min 0.000000e+00  
## max 3.000000e+02  
## range 3.000000e+02  
## sum 8.488000e+04  
## median 2.000000e+00  
## mean 6.597746e+00  
## SE.mean 2.663628e-01  
## CI.mean.0.95 5.221105e-01  
## var 9.127604e+02  
## std.dev 3.021192e+01  
## coef.var 4.579128e+00

Identify if there are any outliers

#The sale price has more than 3 standard deviations, and so does the sale\_instrument, square\_feet\_total\_living, bedrooms, bath counts, and sq\_ft\_lot. anything beyond 3 standard deviations indicates an outlier.

Create at least 2 new variables

housing\_data[c("A", "B")] <- NA  
print(housing\_data)

## # A tibble: 12,865 × 26  
## `Sale Date` sale\_price sale\_r…¹ sale\_…² sale\_…³ sitet…⁴ addr\_…⁵ zip5  
## <dttm> <dbl> <dbl> <dbl> <chr> <chr> <chr> <dbl>  
## 1 2006-01-03 00:00:00 698000 1 3 <NA> R1 17021 … 98052  
## 2 2006-01-03 00:00:00 649990 1 3 <NA> R1 11927 … 98052  
## 3 2006-01-03 00:00:00 572500 1 3 <NA> R1 13315 … 98052  
## 4 2006-01-03 00:00:00 420000 1 3 <NA> R1 3303 1… 98052  
## 5 2006-01-03 00:00:00 369900 1 3 15 R1 16126 … 98052  
## 6 2006-01-03 00:00:00 184667 1 15 18 51 R1 8101 2… 98053  
## 7 2006-01-04 00:00:00 1050000 1 3 <NA> R1 21634 … 98053  
## 8 2006-01-04 00:00:00 875000 1 3 <NA> R1 21404 … 98053  
## 9 2006-01-04 00:00:00 660000 1 3 <NA> R1 7525 2… 98053  
## 10 2006-01-04 00:00:00 650000 1 3 <NA> R1 17703 … 98052  
## # … with 12,855 more rows, 18 more variables: ctyname <chr>, postalctyn <chr>,  
## # lon <dbl>, lat <dbl>, building\_grade <dbl>, square\_feet\_total\_living <dbl>,  
## # bedrooms <dbl>, bath\_full\_count <dbl>, bath\_half\_count <dbl>,  
## # bath\_3qtr\_count <dbl>, year\_built <dbl>, year\_renovated <dbl>,  
## # current\_zoning <chr>, sq\_ft\_lot <dbl>, prop\_type <chr>, present\_use <dbl>,  
## # A <lgl>, B <lgl>, and abbreviated variable names ¹​sale\_reason,  
## # ²​sale\_instrument, ³​sale\_warning, ⁴​sitetype, ⁵​addr\_full