problem_set_2a.R

Aaryan Agarwal

2023-10-15

```r
# PROBLEM SET 2A
# This assignment is done in a group of three
# Prajwal Kaushal, Sumukha Sharma, Aaryan Agarwal

##################################################################################
#PART 1: R QUESTIONS
##################################################################################
#Question 1: Data Transformation
##################################################################################
# Filtering
library(nycflights13)
sum(is.na(flights$dep_time))
```

```
## [1] 8255
```

```r
colSums(is.na(flights))
```

```
##           year          month            day       dep_time sched_dep_time
##              0              0              0           8255              0
##      dep_delay       arr_time sched_arr_time      arr_delay        carrier
##           8255           8713              0           9430              0
##         flight        tailnum         origin           dest       air_time
##              0           2512              0              0           9430
##       distance           hour         minute      time_hour
##              0              0              0              0
```

```r
# we can see that 8255 flights have missing dep_time
# There are other variables that have missing values like - dep_delay, arr_time,
# arr_delay, tailnum, air_time
# Rows with missing dep_time probably represent cancelled flights or flights that didn't
# get recorded properly.

# 2.

# NA ^ 0 gives 1 because anything raised to the power of 0 is 1.
#NA | TRUE gives TRUE because OR plus anything with TRUE is always TRUE.
#FALSE & NA gives FALSE because AND plus anything with FALSE is always FALSE.
#General rule:

# For arithmetic operations: If the result of the operation is determinable
```

```r
# irrespective of the actual value of the missing data (NA), R will return that determinate value.
# For example:
# Any number to the power of zero is 1, so NA ^ 0 returns 1, irrespective of what NA represents.
# Multiplication involving 0 will always result in 0, irrespective of the other operand.
# therefore, NA * 0 gives 0.
# For logical operations:
# If an operation involves OR and one of the values is TRUE, the outcome is definitely
# TRUE irrespective of the other operand's value. So, NA | TRUE gives TRUE.
# If an operation involves AND and one of the values is FALSE, the outcome is definitely FALSE
# irrespective of the other operand's value. Thus, FALSE & NA gives FALSE.

# ARRANGE DATA:
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
# 1.
dplyr::arrange(nycflights13::flights, desc(is.na(dep_time)))
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_de~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013     1     1       NA       1630      NA      NA    1815      NA EV
##  2  2013     1     1       NA       1935      NA      NA    2240      NA AA
##  3  2013     1     1       NA       1500      NA      NA    1825      NA AA
##  4  2013     1     1       NA        600      NA      NA     901      NA B6
##  5  2013     1     2       NA       1540      NA      NA    1747      NA EV
##  6  2013     1     2       NA       1620      NA      NA    1746      NA EV
##  7  2013     1     2       NA       1355      NA      NA    1459      NA EV
##  8  2013     1     2       NA       1420      NA      NA    1644      NA EV
##  9  2013     1     2       NA       1321      NA      NA    1536      NA EV
## 10  2013     1     2       NA       1545      NA      NA    1910      NA AA
## # ... with 336,766 more rows, 9 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, and abbreviated variable names
## #   1: sched_dep_time, 2: dep_delay, 3: arr_time, 4: sched_arr_time,
## #   5: arr_delay
```

```r
# 2.
# For the longest flight
longest_flight <- dplyr::arrange(nycflights13::flights, desc(distance)) %>% dplyr::slice(1)
longest_flight
```

```
## # A tibble: 1 x 19
##    year month   day dep_time sched_dep~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier
##   <int> <int> <int>    <int>       <int>   <dbl>   <int>   <int>   <dbl> <chr>
## 1  2013     1     1      857         900      -3    1516    1530     -14 HA
## # ... with 9 more variables: flight <int>, tailnum <chr>, origin <chr>,
## #   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dttm>, and abbreviated variable names 1: sched_dep_time,
## #   2: dep_delay, 3: arr_time, 4: sched_arr_time, 5: arr_delay
```

```r
# For the shortest flight
shortest_flight <- dplyr::arrange(nycflights13::flights, distance) %>% dplyr::slice(1)
shortest_flight
```

```
## # A tibble: 1 x 19
##    year month   day dep_time sched_dep~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier
##   <int> <int> <int>    <int>       <int>   <dbl>   <int>   <int>   <dbl> <chr>
## 1  2013     7    27       NA         106      NA      NA     245      NA US
## # ... with 9 more variables: flight <int>, tailnum <chr>, origin <chr>,
## #   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dttm>, and abbreviated variable names 1: sched_dep_time,
## #   2: dep_delay, 3: arr_time, 4: sched_arr_time, 5: arr_delay
```

```r
# SELECT COLUMNS:

# 1.

dplyr::select(nycflights13::flights, year, year, year)
```

```
## # A tibble: 336,776 x 1
##     year
##    <int>
##  1  2013
##  2  2013
##  3  2013
##  4  2013
##  5  2013
##  6  2013
##  7  2013
##  8  2013
##  9  2013
## 10  2013
## # ... with 336,766 more rows
```

```r
# as we can see herethe column will only be selected once



# Create new variables
# 1.


flights$difference <- flights$arr_time - flights$dep_time
```

```r
selected_flights <- flights[, c("air_time", "dep_time", "arr_time", "difference")]
head(selected_flights)
```

```
## # A tibble: 6 x 4
##   air_time dep_time arr_time difference
##      <dbl>    <int>    <int>      <int>
## 1      227      517      830        313
## 2      227      533      850        317
## 3      160      542      923        381
## 4      183      544     1004        460
## 5      116      554      812        258
## 6      150      554      740        186
```

```r
# air_time is the time spent in the air,
# while arr_time - dep_time is a rough measure of the total duration of the flight.
# However, this isn't a perfect comparison because arr_time and dep_time are in the HHMM format,
# while ait_time is in minutes,
# so direct subtraction will lead to incorrect calculations.
# To fix this, we need to convert these times into a format that correctly represents the duration.
# something like this:
# Computing the minutes since midnight for departure and arrival
flights$dep_minutes <- (flights$dep_time %/% 100) * 60 + (flights$dep_time %% 100)
flights$arr_minutes <- (flights$arr_time %/% 100) * 60 + (flights$arr_time %% 100)

# 2.

selected_columns <- flights[, c("dep_time", "sched_dep_time", "dep_delay")]
head(selected_columns)
```

```
## # A tibble: 6 x 3
##   dep_time sched_dep_time dep_delay
##      <int>          <int>     <dbl>
## 1      517            515         2
## 2      533            529         4
## 3      542            540         2
## 4      544            545        -1
## 5      554            600        -6
## 6      554            558        -4
```

```r
# dep_delay is the difference between dep_time and sched_dep_time.
# So, dep_time = sched_dep_time + dep_delay.

# 3.

1:3 + 1:10
```

```
## Warning in 1:3 + 1:10: longer object length is not a multiple of shorter object
## length
```

```
##  [1]  2  4  6  5  7  9  8 10 12 11
```

```r
# This gives a warning.
# The reason is that R is trying to repeat the shorter vector
# to match the length of the longer one.
# It repeats 1:3 until it reaches the length of 1:10


# Grouped summaries

# 1.

# I am assuming that the dep_time in na for cancelled flights
# Grouping by year, month, and day
grouped_flights <- group_by(flights, year, month, day)
daily_summary <- summarise(grouped_flights,
                           cancelled_flights = sum(is.na(dep_time)),
                           avg_delay = mean(dep_delay, na.rm = TRUE))
```

```
## `summarise()` has grouped output by 'year', 'month'. You can override using the
## `.groups` argument.
```

```r
print(daily_summary, n=100)
```

```
## # A tibble: 365 x 5
## # Groups:   year, month [12]
##      year month   day cancelled_flights avg_delay
##     <int> <int> <int>             <int>     <dbl>
##   1  2013     1     1                 4    11.5
##   2  2013     1     2                 8    13.9
##   3  2013     1     3                10    11.0
##   4  2013     1     4                 6     8.95
##   5  2013     1     5                 3     5.73
##   6  2013     1     6                 1     7.15
##   7  2013     1     7                 3     5.42
##   8  2013     1     8                 4     2.55
##   9  2013     1     9                 5     2.28
##  10  2013     1    10                 3     2.84
##  11  2013     1    11                11     2.82
##  12  2013     1    12                 6     1.60
##  13  2013     1    13                16    19.9
##  14  2013     1    14                 2     2.79
##  15  2013     1    15                13     0.124
##  16  2013     1    16                46    24.6
##  17  2013     1    17                 9     7.65
##  18  2013     1    18                10     6.77
##  19  2013     1    19                 1     3.48
##  20  2013     1    20                 4     6.78
##  21  2013     1    21                 8     7.83
##  22  2013     1    22                 5    12.5
##  23  2013     1    23                 9    10.6
##  24  2013     1    24                14    19.5
##  25  2013     1    25                35    21.9
##  26  2013     1    26                 9     7.21
```

```
## 27 2013     1    27                  16    8.38
## 28 2013     1    28                  64   15.1
## 29 2013     1    29                  13    2.50
## 30 2013     1    30                  98   28.6
## 31 2013     1    31                  85   28.7
## 32 2013     2     1                  15   10.9
## 33 2013     2     2                   2    5.42
## 34 2013     2     3                  19    7.02
## 35 2013     2     4                  10   10.9
## 36 2013     2     5                  16    5.32
## 37 2013     2     6                   8    5.62
## 38 2013     2     7                   4    6.50
## 39 2013     2     8                 472   14.9
## 40 2013     2     9                 393   18.5
## 41 2013     2    10                  26   15.2
## 42 2013     2    11                  73   39.1
## 43 2013     2    12                   6    4.67
## 44 2013     2    13                  13    3.66
## 45 2013     2    14                   4    5.62
## 46 2013     2    15                   6    5.94
## 47 2013     2    16                   1    7.01
## 48 2013     2    17                  16   10.3
## 49 2013     2    18                   3    8.53
## 50 2013     2    19                  15   17.4
## 51 2013     2    20                  13    9.46
## 52 2013     2    21                  22   10.6
## 53 2013     2    22                  17   12.1
## 54 2013     2    23                   3   11.6
## 55 2013     2    24                   9    6.80
## 56 2013     2    25                  13    6.20
## 57 2013     2    26                  31    7.80
## 58 2013     2    27                  41   37.8
## 59 2013     2    28                  10    5.55
## 60 2013     3     1                  14   11.0
## 61 2013     3     2                  11    8.03
## 62 2013     3     3                   3    6.07
## 63 2013     3     4                  18    4.75
## 64 2013     3     5                  29    5.02
## 65 2013     3     6                 180   21.0
## 66 2013     3     7                  98   20.4
## 67 2013     3     8                 180   83.5
## 68 2013     3     9                   9   11.3
## 69 2013     3    10                   5   10.7
## 70 2013     3    11                   3    6.91
## 71 2013     3    12                  60   26.3
## 72 2013     3    13                   8    6.46
## 73 2013     3    14                   1   12.0
## 74 2013     3    15                   8   12.4
## 75 2013     3    16                   2   10.2
## 76 2013     3    17                   9    7.93
## 77 2013     3    18                  40   30.1
## 78 2013     3    19                  38   23.6
## 79 2013     3    20                  13    8.44
## 80 2013     3    21                   7   10.7
```

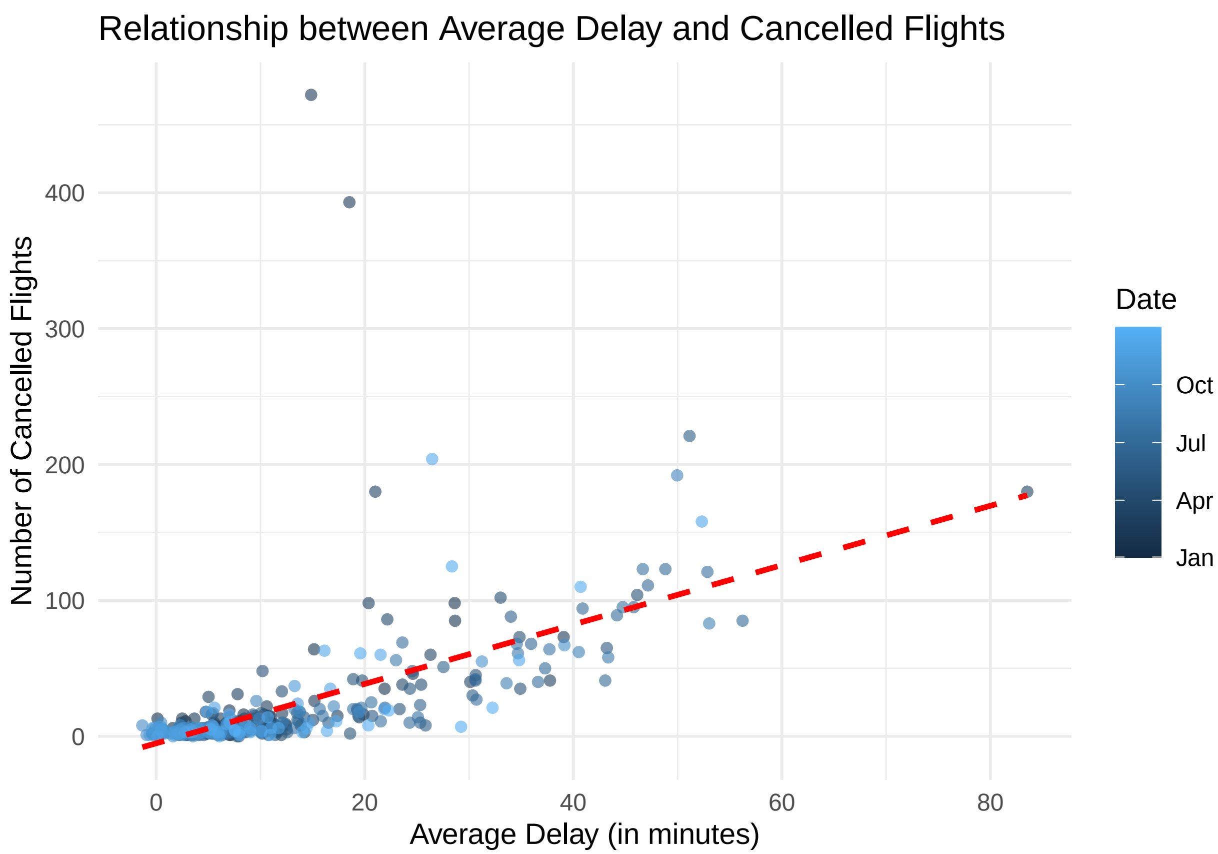
```
## 81 2013 3 22 4 10.6
## 82 2013 3 23 3 14.2
## 83 2013 3 24 12 15.0
## 84 2013 3 25 86 22.2
## 85 2013 3 26 6 2.17
## 86 2013 3 27 2 2.79
## 87 2013 3 28 4 6.24
## 88 2013 3 29 4 2.93
## 89 2013 3 30 1 2.18
## 90 2013 3 31 3 5.61
## 91 2013 4 1 9 12.4
## 92 2013 4 2 3 8.26
## 93 2013 4 3 2 3.45
## 94 2013 4 4 5 6.96
## 95 2013 4 5 1 5.91
## 96 2013 4 6 2 4.95
## 97 2013 4 7 3 2.86
## 98 2013 4 8 10 2.42
## 99 2013 4 9 15 9.43
## 100 2013 4 10 102 33.0
## # ... with 265 more rows
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.2.2
```

```
ggplot(daily_summary, aes(x = avg_delay, y = cancelled_flights)) +
  geom_point(aes(color = as.Date(paste(year, month, day, sep = "-"))), alpha = 0.6) +  # color by date
  geom_smooth(method = "lm", se = FALSE, color = "red", linetype = "dashed") +  # adds a linear regress
  labs(title = "Relationship between Average Delay and Cancelled Flights",
       x = "Average Delay (in minutes)",
       y = "Number of Cancelled Flights",
       color = "Date") +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
# we can see that on many days where there's a high number of cancelled flights,
# there's also a high average delay. Therefore, there seems to be a relationship
# between the proportion of cancelled flights and the average delay.
# with the plot we can say that it seems to be kind of linear.

# 2.
# carriers with worst delays
carrier_delays <- flights %>%
  group_by(carrier) %>%
  summarise(
    avg_delay = mean(arr_delay, na.rm = TRUE)
  ) %>%
  arrange(desc(avg_delay))

head(carrier_delays)
```

```
## # A tibble: 6 x 2
##   carrier avg_delay
##   <chr>       <dbl>
## 1 F9           21.9
## 2 FL           20.1
## 3 EV           15.8
## 4 YV           15.6
## 5 OO           11.9
## 6 MQ           10.8
```

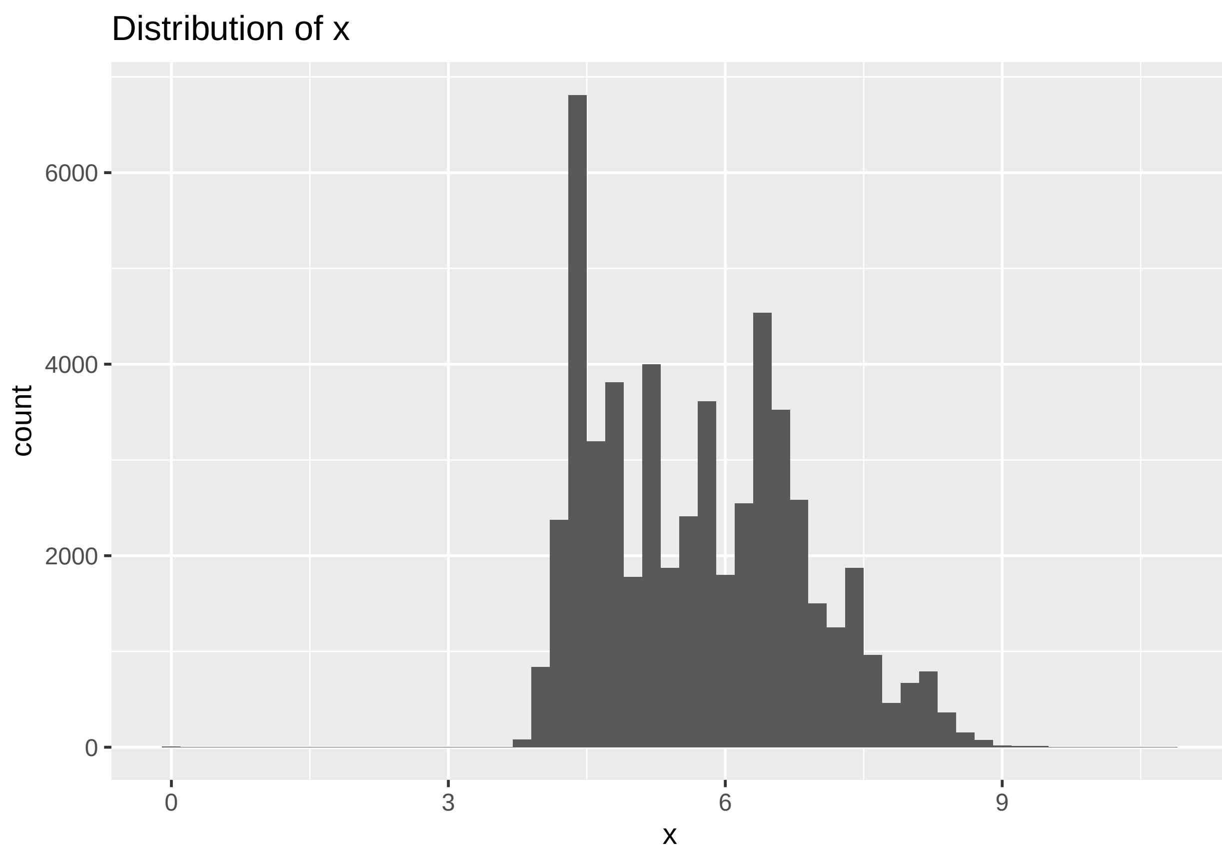
```
# Challenge
carrier_airport_counts <- flights %>%
  group_by(carrier, dest) %>%
  summarise(
    flights_count = n(),
    avg_delay = mean(arr_delay, na.rm = TRUE)
  ) %>%
  arrange(desc(avg_delay))
```

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

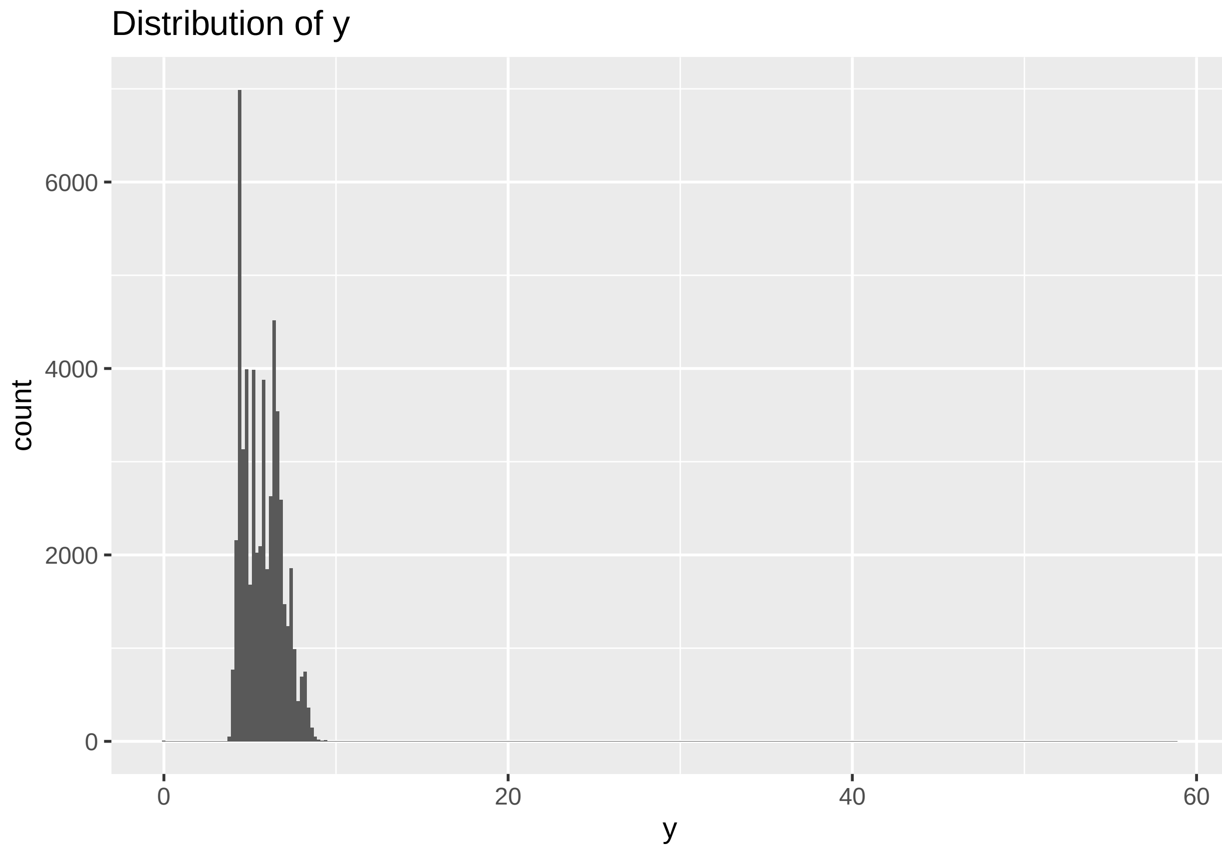
```
head(carrier_airport_counts)
```

```
## # A tibble: 6 x 4
## # Groups:   carrier [3]
##   carrier dest  flights_count avg_delay
##   <chr>   <chr>         <int>     <dbl>
## 1 UA      STL               2     110
## 2 OO      ORD               1     107
## 3 OO      DTW               2      68.5
## 4 UA      RDU               1      56
## 5 EV      CAE             113      42.8
## 6 EV      TYS             323      41.2
```

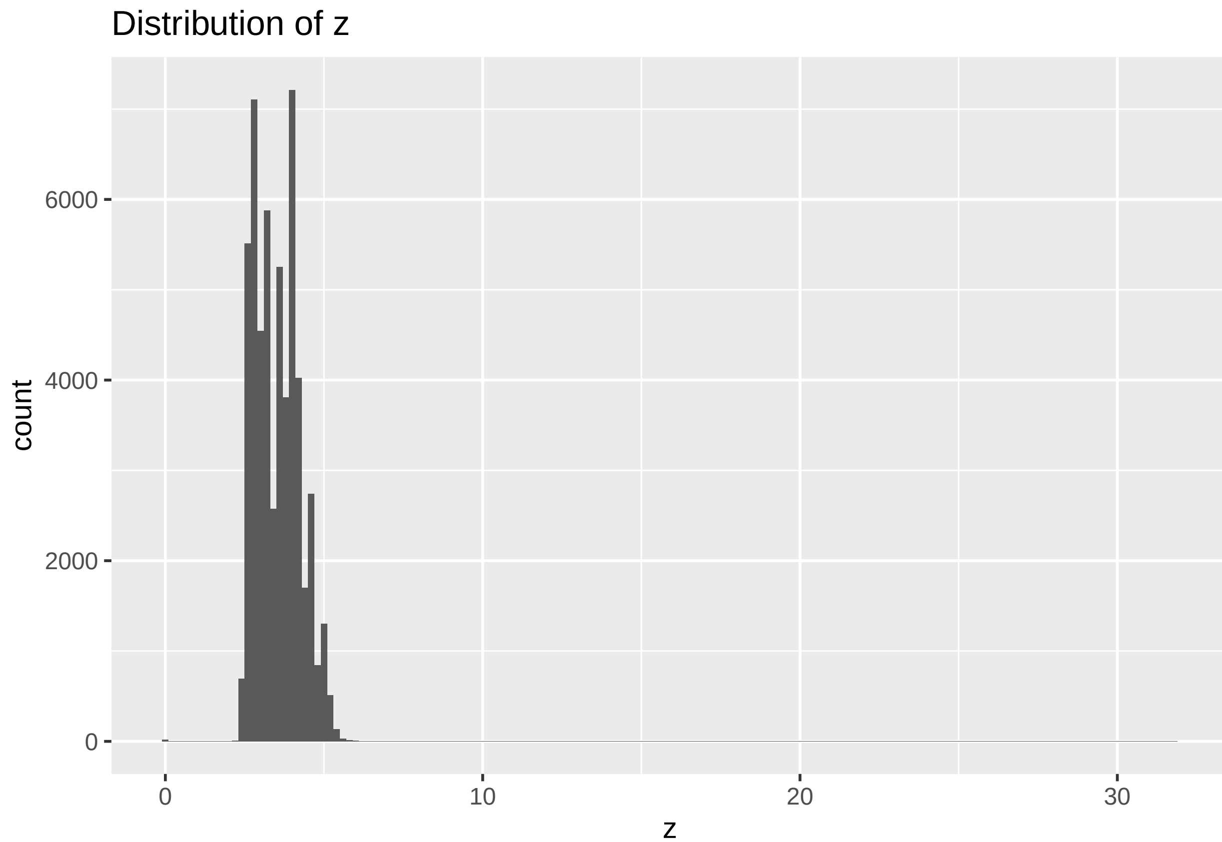
```
# from this grouped summary we ccan seee which combination of carriers and destinations
# have the most flights and the average delay.
# we have to think about these two questions:
# Are there carriers that consistently have bad delays irrespective of the destinations?
# Are there destinations that consistently have delays irrespective of the carrier?


###############################################################################
#Question 2: Exploratory Data Analysis
###############################################################################

# Typical and atypical values

library(ggplot2)

# 1.

ggplot(diamonds, aes(x=x)) + geom_histogram(binwidth=0.2) + ggtitle("Distribution of x")
```



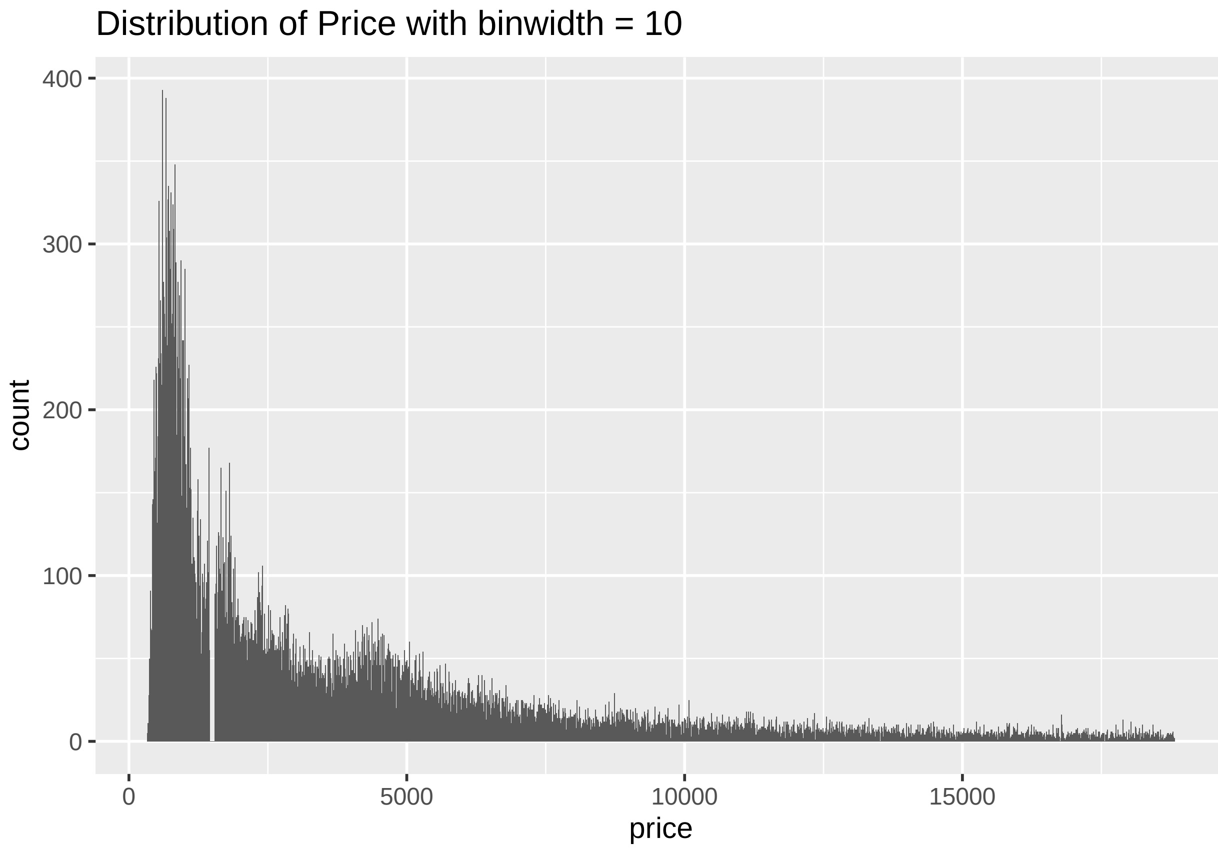

```
ggplot(diamonds, aes(x=y)) + geom_histogram(binwidth=0.2) + ggtitle("Distribution of y")
```



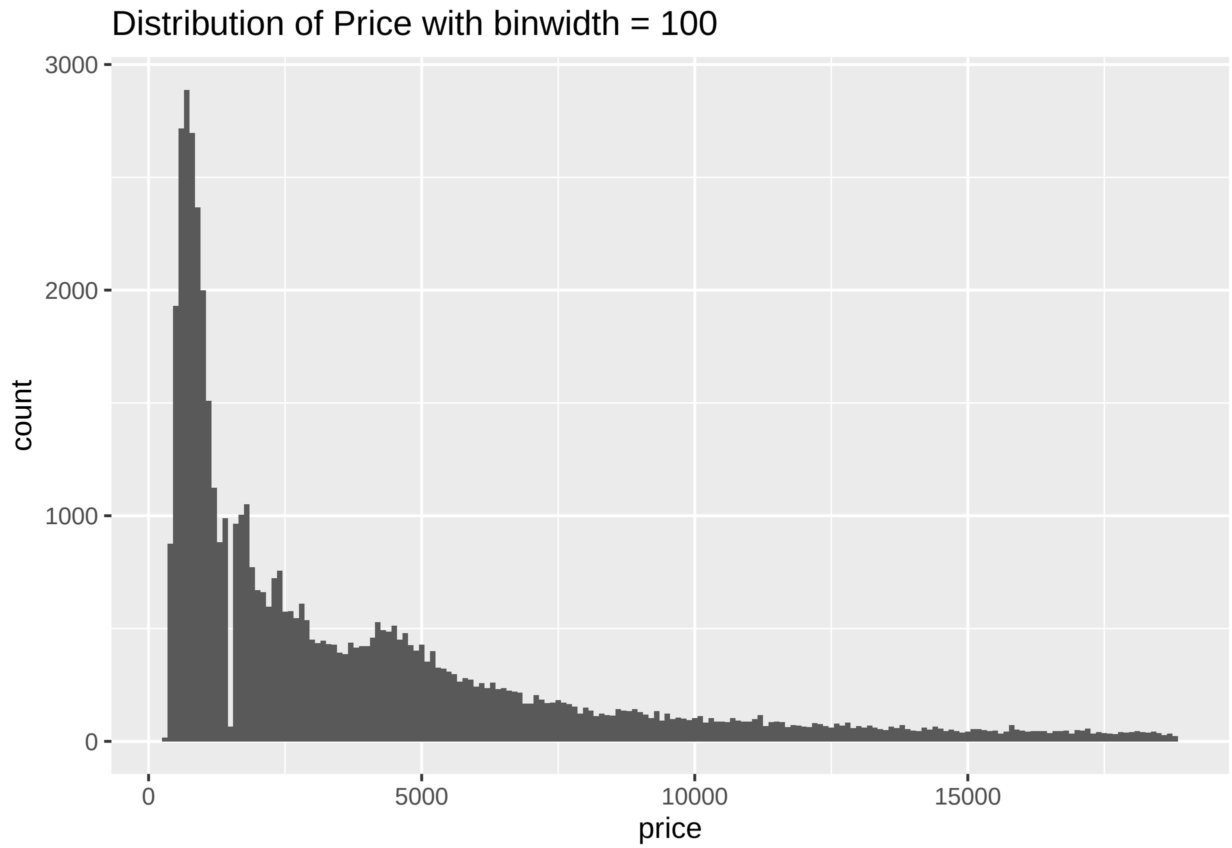

```
ggplot(diamonds, aes(x=z)) + geom_histogram(binwidth=0.2) + ggtitle("Distribution of z")
```



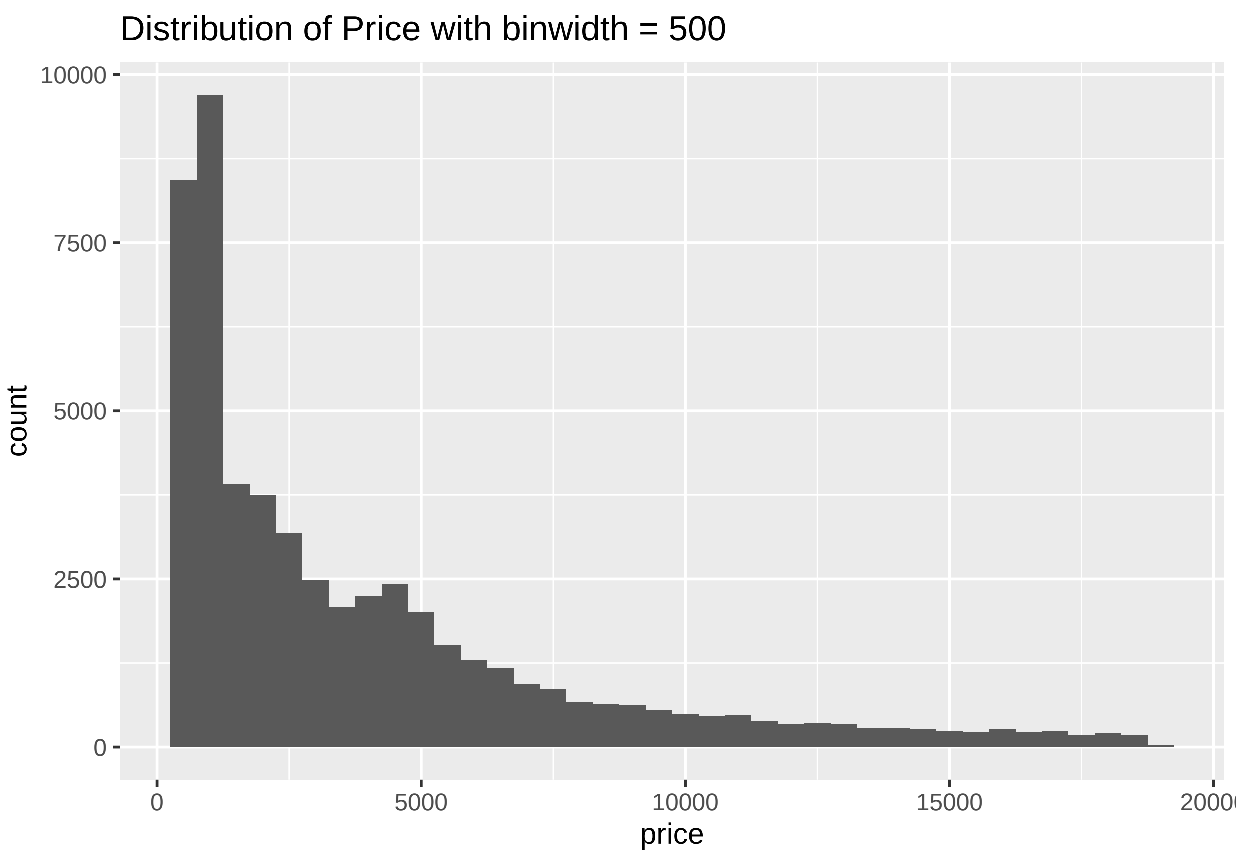

```
# 2.

ggplot(diamonds, aes(x=price)) + geom_histogram(binwidth=10) +
  ggtitle("Distribution of Price with binwidth = 10")
```




```r
ggplot(diamonds, aes(x=price)) + geom_histogram(binwidth=100) +
  ggtitle("Distribution of Price with binwidth = 100")
```




```
ggplot(diamonds, aes(x=price)) + geom_histogram(binwidth=500) +
  ggtitle("Distribution of Price with binwidth = 500")
```




```
# By adjusting the binwidth, we can observe the granularity of the distribution.
# we can see a huge spike between 500-800
# there is almost no bar around 1400 price

# 3.
sum(diamonds$carat == 0.99)
```

```
## [1] 23
```

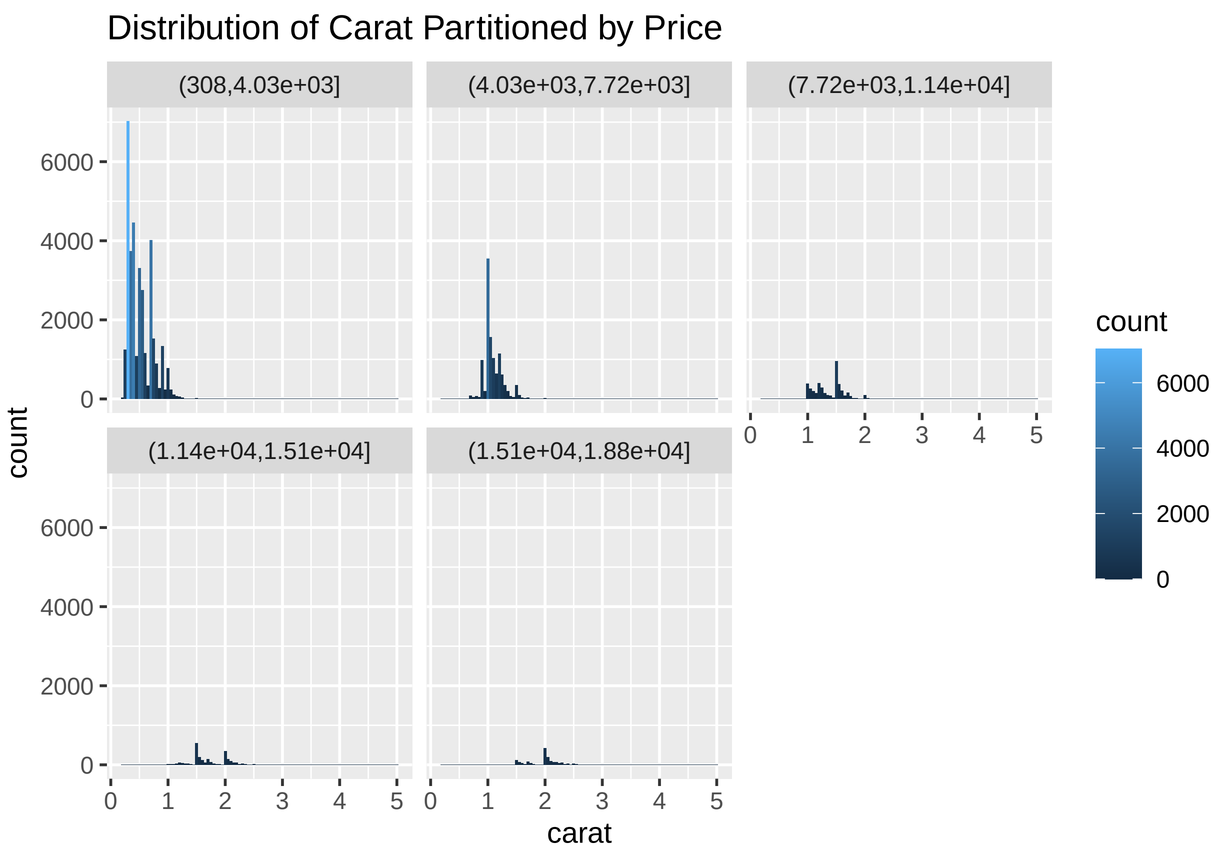
```
sum(diamonds$carat == 1)
```

```
## [1] 1558
```

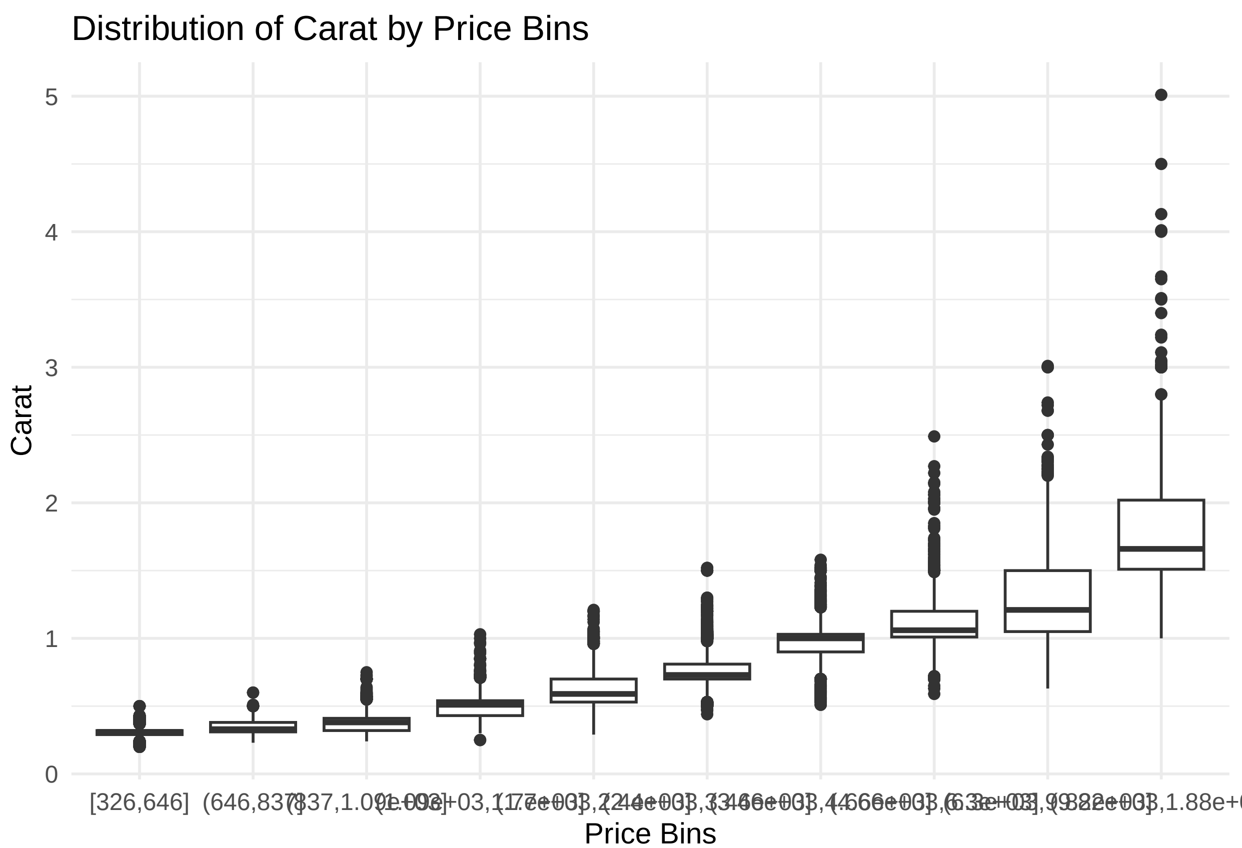
```
# A 1-carat diamond might be seen as more prestigious than a 0.99-carat diamond,
# influencing purchasing and selling behavior.

# Missing values

# 1.

# Missing values are ignored in histograms and won't be represented in the bins.
# In bar charts, NA is considered as just another category and they could get their own bar.

# 2.
```

```
# This removes NA values from the vector before calculating the mean and sum


# Covariance

# 1.

# using hisstogram
ggplot(diamonds, aes(x=carat)) + geom_histogram(aes(fill=..count..), binwidth=0.05) + facet_wrap(~cut(pr
```

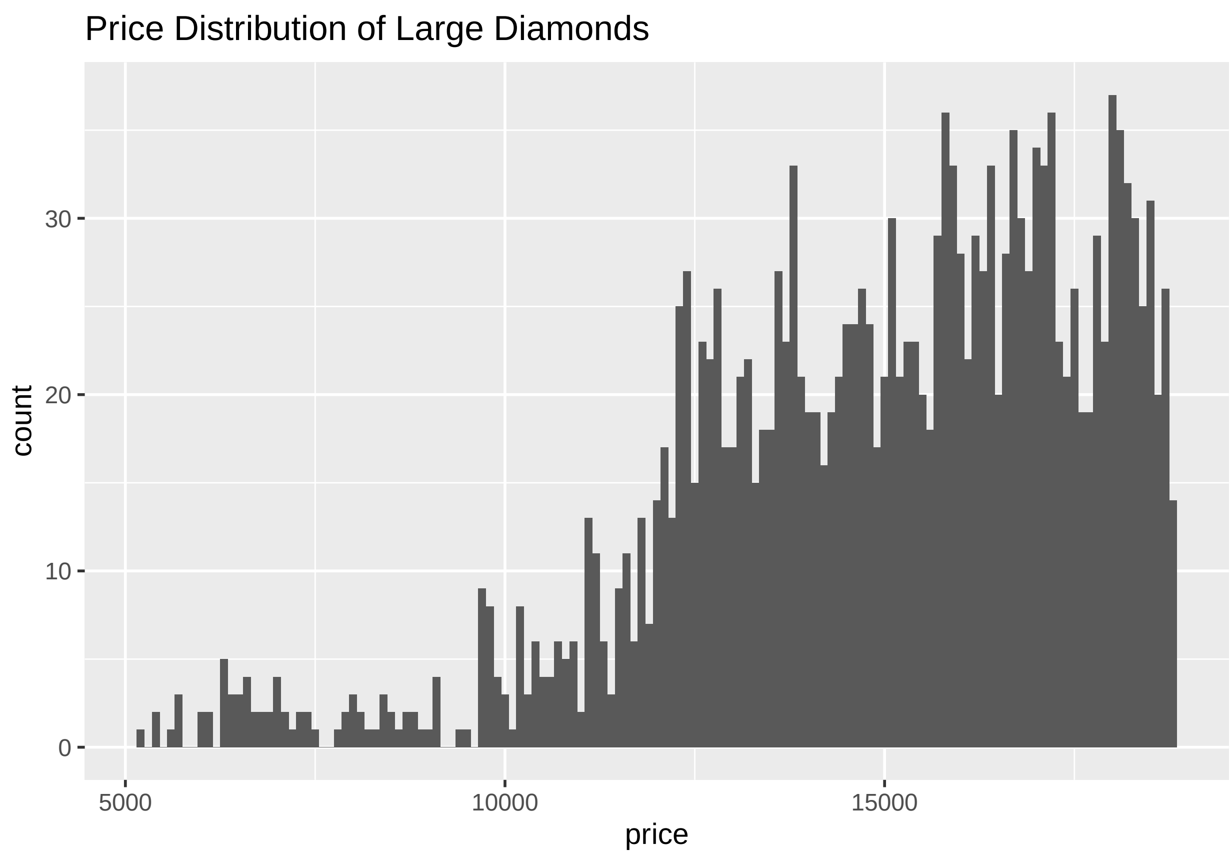
```
## Warning: The dot-dot notation (`..count..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(count)` instead.
```



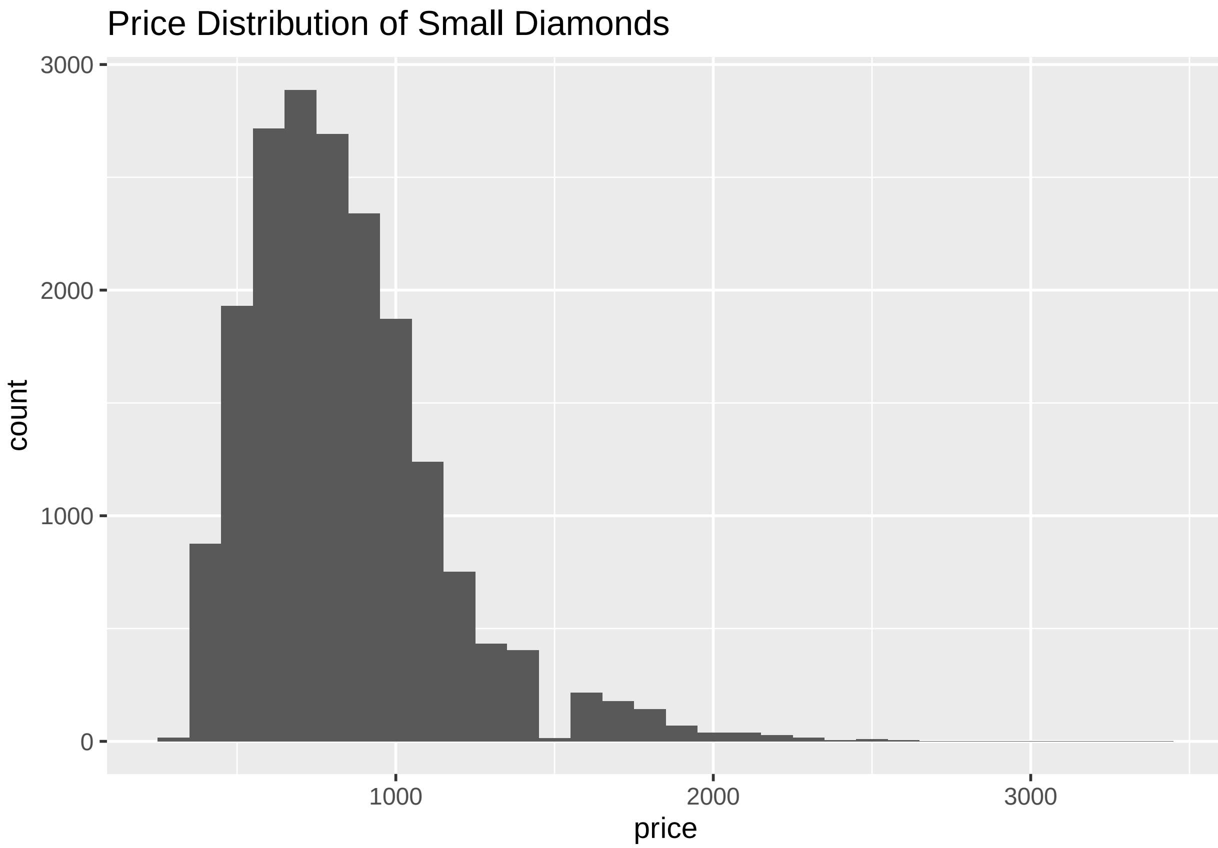

```
# This visualizes how the distribution of carat changes across different price segments

# using boxplot
ggplot(diamonds, aes(x = cut_number(price, 10), y = carat)) +
  geom_boxplot() +
  labs(title = "Distribution of Carat by Price Bins",
       x = "Price Bins",
       y = "Carat") +
  theme_minimal()
```




```r
# 2.

large_diamonds <- diamonds[diamonds$carat > 2, ]
small_diamonds <- diamonds[diamonds$carat <= 0.5, ]

ggplot(large_diamonds, aes(x=price)) + geom_histogram(binwidth=100) + ggtitle("Price Distribution of La
```




```
ggplot(small_diamonds, aes(x=price)) + geom_histogram(binwidth=100) + ggtitle("Price Distribution of Sma
```




```
# The distribution of large diamonds is variable
# The of large diamond is higher.


################################################################################
#PART 2: Project
################################################################################

#An interesting data set we came across was the General Social Survey (GSS)
#dataset. It is a high-quality survey which gathers data on American society
#and opinions, and it conducted since 1972. The data set can be accessed in
#R using the library infer. The dataset present in R is a sample of the
#original dataset of 500 entries from the original with a span of years
#1973-2018. It includes demographic markers and some economic variables.
#It contains of 11 variables namely year (year the respondent was surveyed),
#age (age of the respondent at the time of the survey), sex (gender of the
#respondent which is self-identified by them), college
#(whether the respondent has a valid college degree or no),
#partyid (respondents political party affiliation),
#hompop (number of people in the respondents house),
#hours (number of hours the respondent works while he was being surveyed),
#income (total family income of the respondent), class
#(subjective socioeconomic class identification), finrela
#(opinion of family income) and weight (survey weight). The data set consists
#of just 500 rows of data.
#We can use this dataset to generate the average number of people living
```

```
#in each household in a certain year. We can chart out the slope of the '
#increase or the decrease in the number of people in each household.
#We can determine how much an average worker works each week and
#the average salary they get for each hour. We can group the previous
#result based on the class of the individual. We can determine which political
#party is likely to succeed in that area during a specific year. The literacy
#rate of the area can be determined on whether a person has achieved a degree
#or not. Many such inferences can be made through this dataset by various
#statistical methods. We can group the dataset based upon the years by
#splitting the dataset and can determine many inferences according to the year.
#Same can be done by splitting the dataset by class or political party
#preferences.
#Its good data because we can infer many different conditions as given above
#and it gives us a lot of potential. The original dataset is available on the
#gss website and should be easily accessible. A shorter format is available
#in the infer library in R if for some reason we are not able to process the
#data.
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
```

```
## v tibble  3.1.8     v purrr   0.3.5
## v tidyr   1.2.1     v stringr 1.5.0
## v readr   2.1.3     v forcats 0.5.2
```

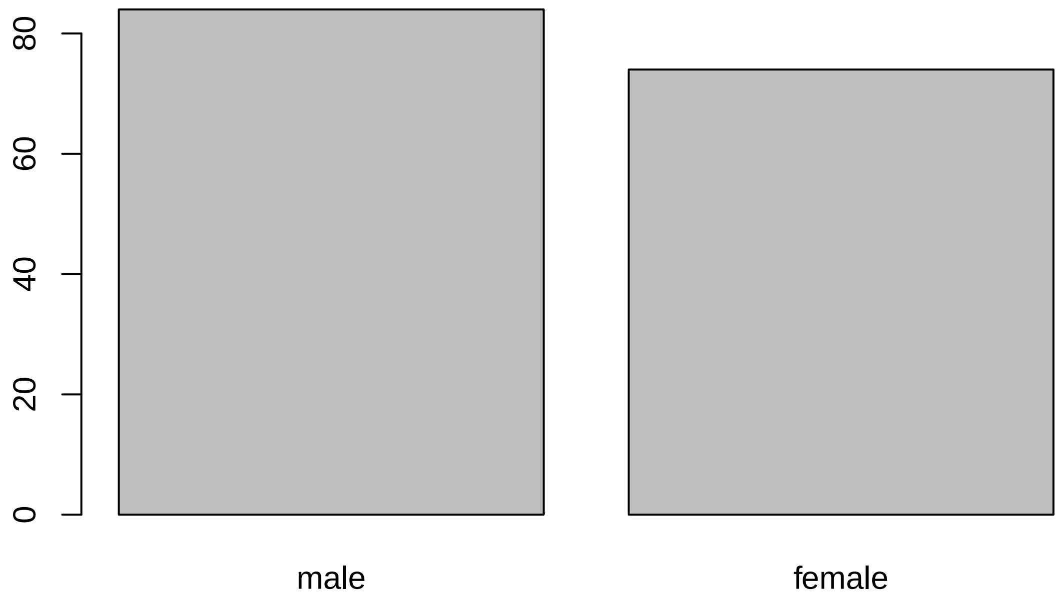
```
## Warning: package 'stringr' was built under R version 4.2.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

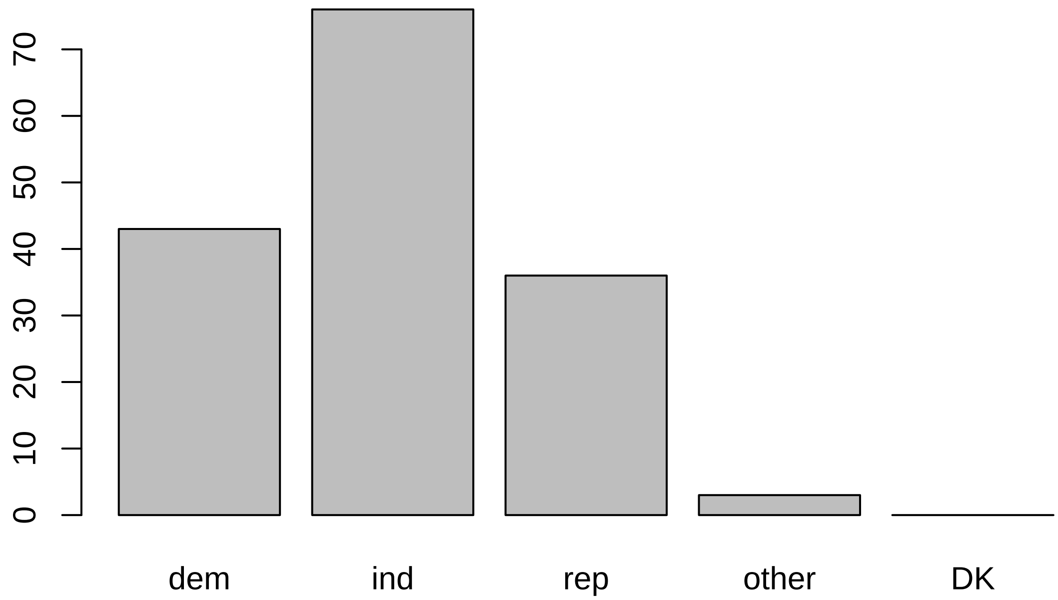
```
library(infer)
```

```
## Warning: package 'infer' was built under R version 4.2.2
```

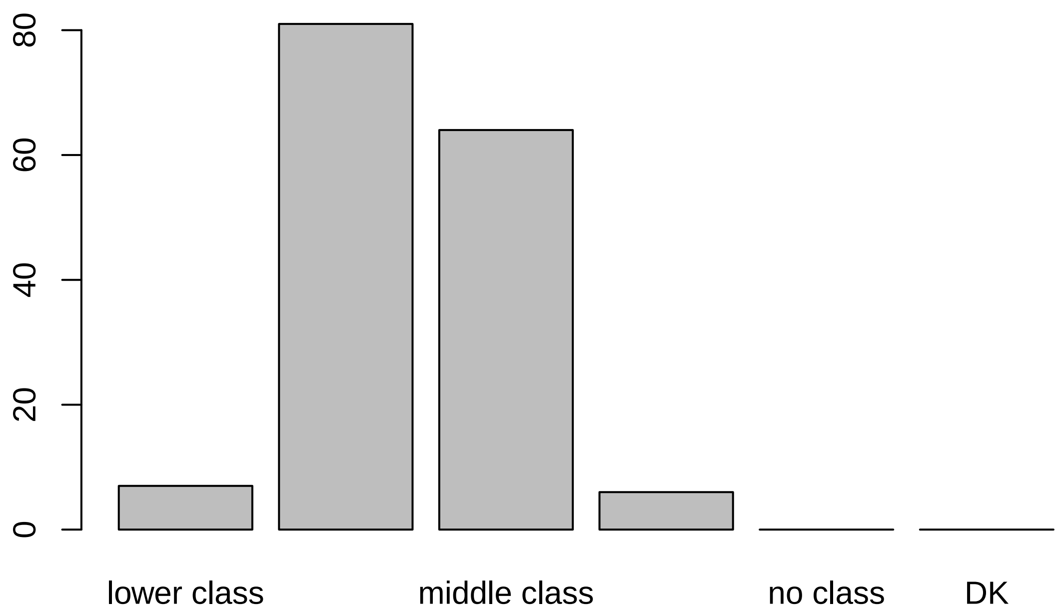
```
library(ggplot2)
data<-gss
newdata1<-filter(data,year>2000)
#Data collected after the year 2000
plot(newdata1['sex'])
```




```r
plot(newdata1['partyid'])
```

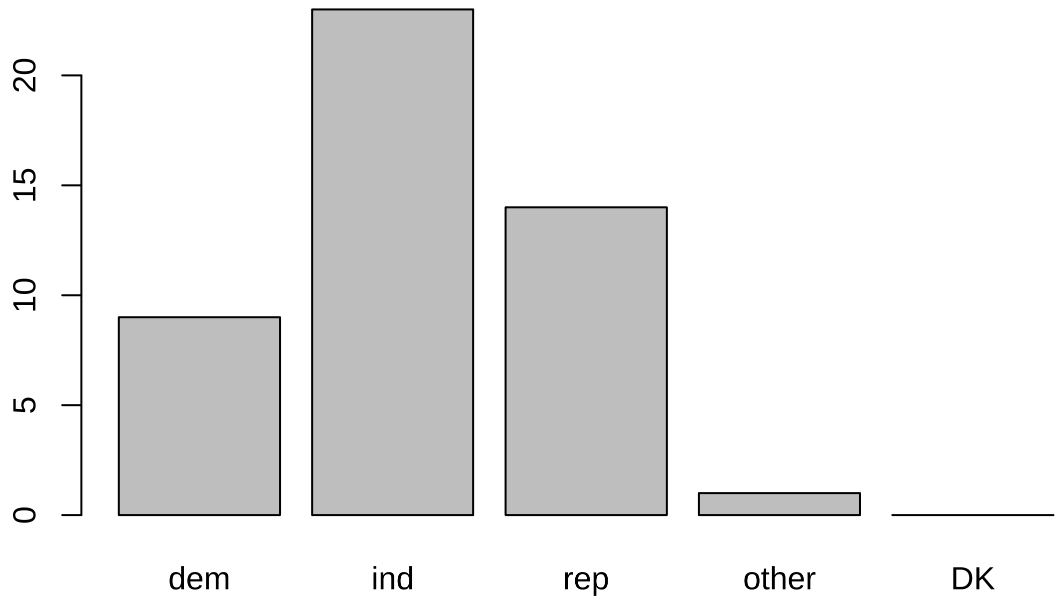



```
plot(newdata1['class'])
```

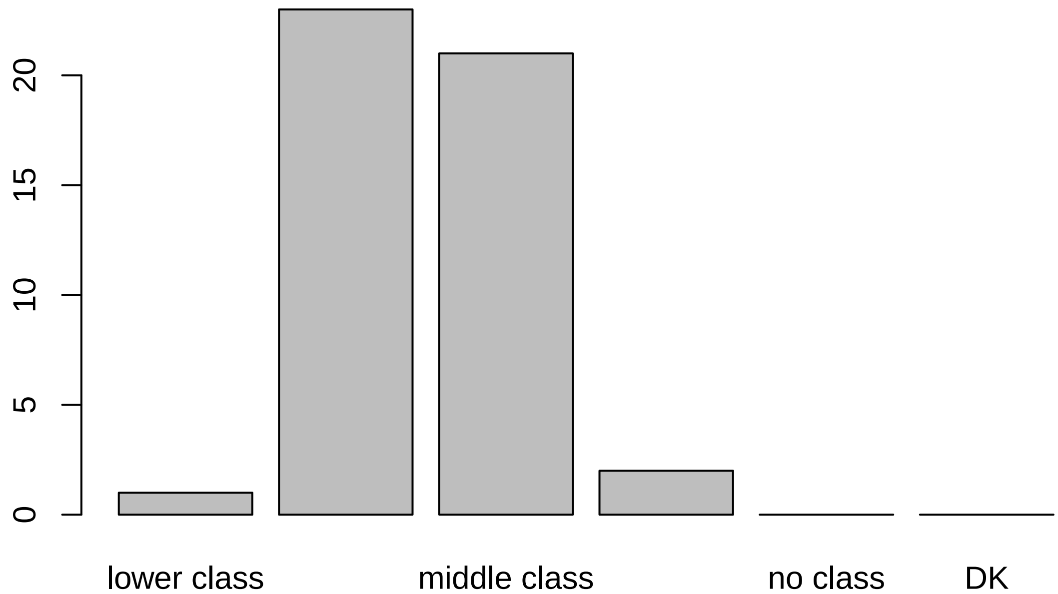



```r
newdata2<-filter(data,year>2000 & weight>1)
#Data collected after the year 2000 and the survey weight is greater than 1
plot(newdata2['sex'])
```

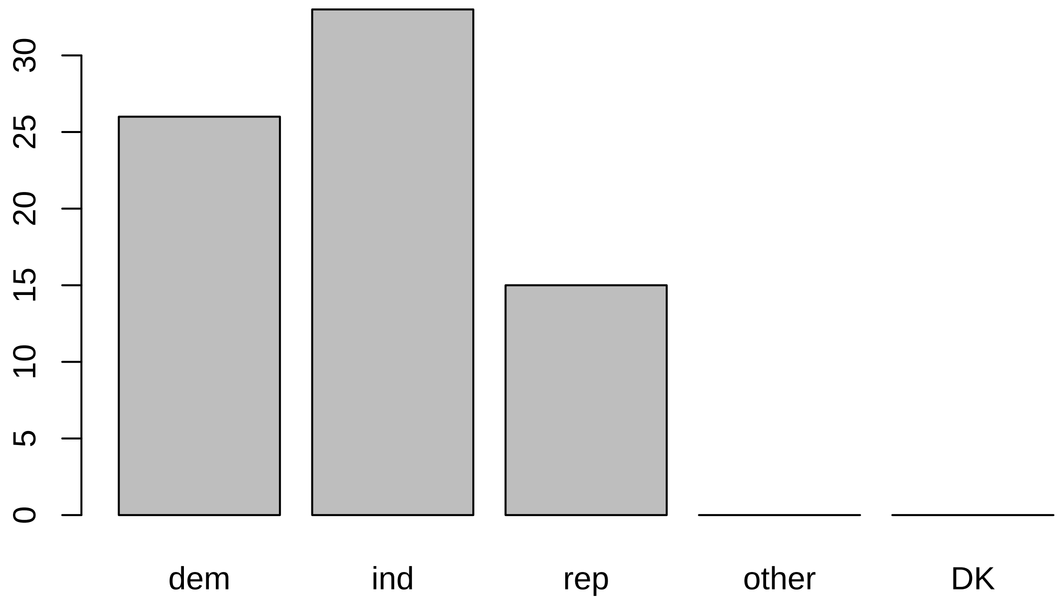

```
plot(newdata2['partyid'])
```



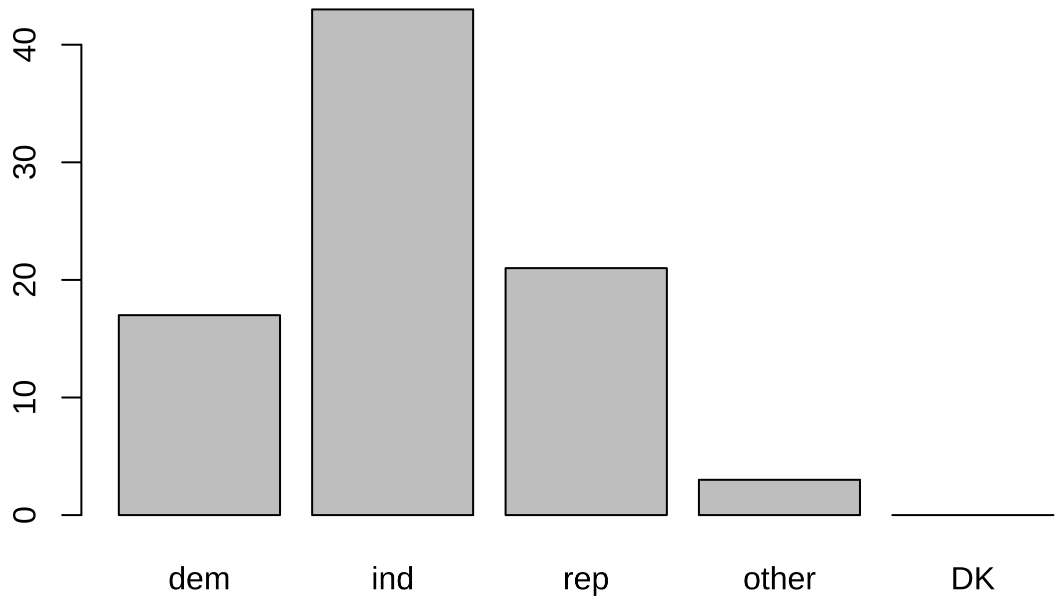

```
plot(newdata2['class'])
```



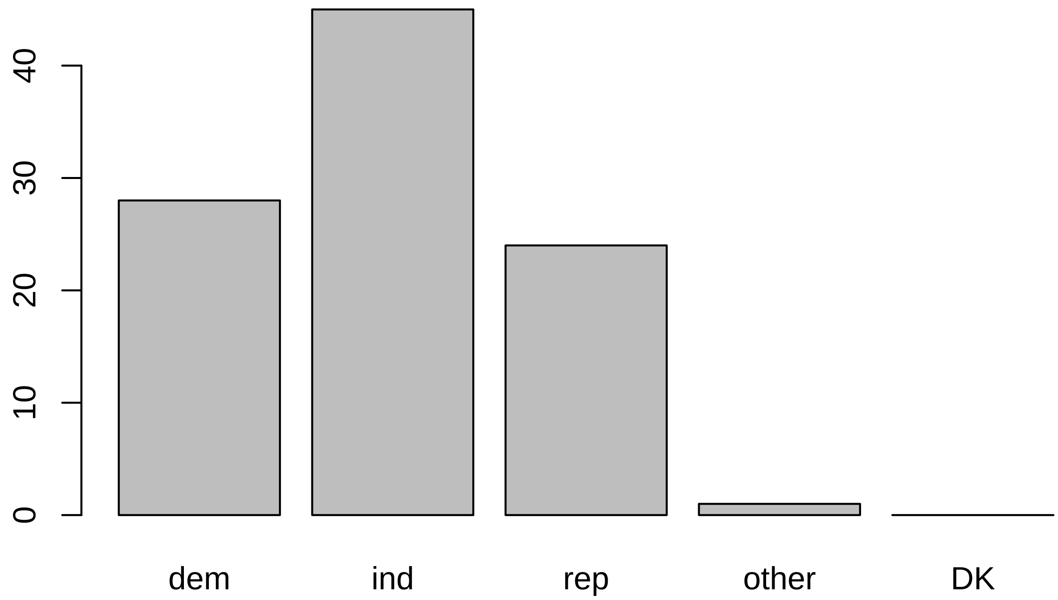

```
#We can see that the females have a higher survey weight than the men
#Data collected from men and women respectively
newdata3<-filter(data,year>2000 & sex=='female')
plot(newdata3['partyid'])
```



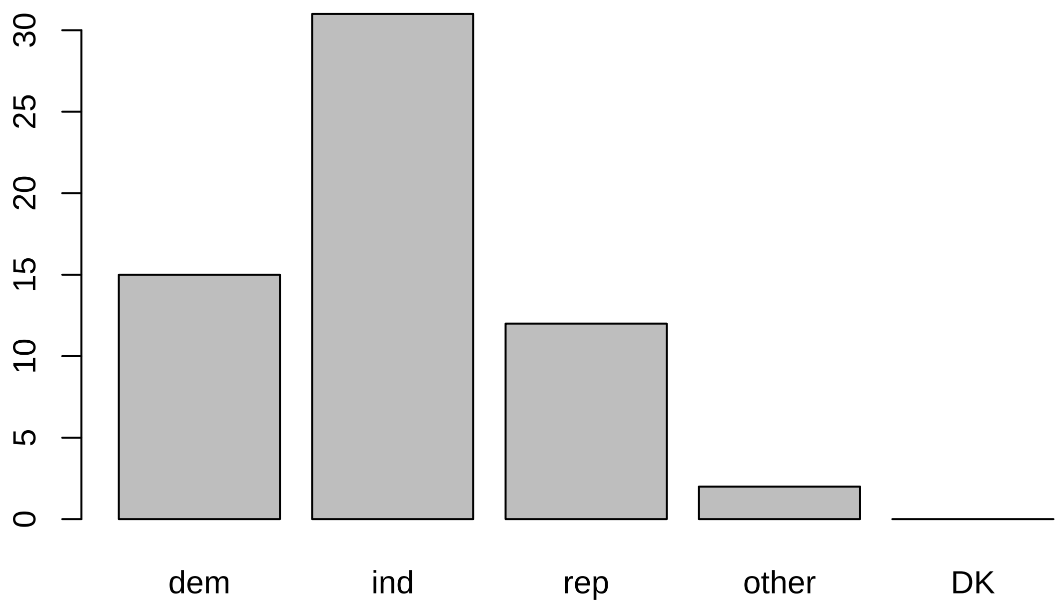

```
newdata4<-filter(data,year>2000 & sex=='male')
plot(newdata4['partyid'])
```




```
#We can see that the females tend to vote for the democratic party
#less than the males
#Data collected from people above and below the age of 35 respectively
newdata5<-filter(data,year>2000 & age>35)
plot(newdata5['partyid'])
```




```
newdata6<-filter(data,year>2000 & age<=35)
plot(newdata6['partyid'])
```




```
#We can see that the people below the age of 35 have less confidence
#in the democratic party than the people above the age of 35
```