Harvesting Data

Dr. Jeffrey Strickland

8/30/2018

Introduction

In my conception of harvesting, which I freely borrow from industries such as farming and fishing, data harvesting includes everything required to "take the data to market." In Table 1, I use the analogy of harvesting fish.

Table 1: Fishing for Data

Fish Harvesting	Data Harvesting	Description
Hatching (without regard for the reproductive process)	Data discovery	Searching for an finding (new[1]) data
Development/growth	Data development	Data growth or allowing enough data to accumulate so as to be worthwhile
Maturation/attaining harvesting age	Data maturation	Allowing data to mature or reach a usable state
Fishing/harvesting	Data harvesting	Collecting the data
Preprocessing: cleaning, cutting, etc.	Data preprocessing	Cleaning the data
Processing: preserving, etc.	Data processing	Reading the data for accessibility
Fish market	Datamart-ing	Storing the data for accessing
Fresh baked Atlantic Cod for dinner	Data consumption	Using the data to obtain/ derive information
Fishery management	Data management	Managing the known "population" of data

Just as fishing should result in nourishment of the body, so for data to be useful it must contain or lead to actionable information. The analogy is neither perfect, nor the science pure.

radiant.data

Radiant[2] is an open-source platform-independent browser-based interface for business analytics in R, which can be run locally or on a server. When code is rendered, it has the appearance as code from the scripting window. All output is preceded by ## and is included so that you can try to duplicate what I have done.

---

[1] New data maybe data that is "new" to you or actual new data.

[2] Radiant was developed by Vincent Nijs.

Summary method for Pivot Tables with pivotr()

Pivot tables are powerful tools for exploring data. They are tables that summarizes data from another table and is made by applying an operation such as sorting, averaging, or summing to data in the first table, typically including grouping of the data.

Pivot tables are used in data processing and are found in data visualization programs such as spreadsheets or business intelligence software. Such programs can automatically sort, count, total or average the data stored in one table or spreadsheet, displaying the results in a second table — the pivot table — showing the summarized data.

This "rotation" or pivoting of the summary table gives the concept its name. Also called cross-tabulation, as we look across the table's pivot (diagonal) and look for the strength of relationship.

Instruction

Perform each step in this lab, being careful to read the content that surrounds the code.

```
library(radiant.data)

pivotr(diamonds, cvars = "cut") %>% summary(chi2 = TRUE)
```

```
## Pivot table
## Data        : diamonds
## Categorical : cut
##
##        cut n_obs
##       Fair   101
##       Good   275
##  Very Good   677
##    Premium   771
##      Ideal 1,176
##      Total 3,000
##
## Chi-squared: 1202.62 df(4), p.value < .001
## 0.0% of cells have expected values below 5
```

```
pivotr(diamonds, cvars = "cut", tabsort = "desc(n_obs)") %>% summary()
```

```
## Pivot table
## Data        : diamonds
## Table sorted: desc(n_obs)
## Categorical : cut
##
##        cut n_obs
##      Ideal 1,176
##    Premium   771
##  Very Good   677
##       Good   275
```

```
##       Fair   101
##      Total 3,000
```

```
pivotr(diamonds, cvars = "cut", tabfilt = "n_obs > 700") %>% summary()
```

```
## Pivot table
## Data        : diamonds
## Table filter: n_obs > 700
## Categorical : cut
##
##      cut n_obs
##  Premium   771
##    Ideal 1,176
##    Total 3,000
```

```
pivotr(diamonds, cvars = "cut:clarity", nvar = "price") %>% summary()
```

```
## Pivot table
## Data        : diamonds
## Categorical : cut clarity
## Numeric     : price
## Function    : mean
##
##  clarity      Fair      Good Very_Good   Premium     Ideal     Total
##       I1 2,730.167 4,333.500 3,864.167 4,932.231 6,078.200 4,194.775
##      SI2 5,893.964 5,280.919 5,045.621 5,568.019 4,435.673 5,100.189
##      SI1 4,273.069 3,757.022 4,277.544 4,113.811 3,758.125 3,998.577
##      VS2 3,292.000 3,925.481 3,950.947 4,522.914 3,306.290 3,822.967
##      VS1 5,110.769 3,740.697 3,889.475 4,461.333 3,189.362 3,789.181
##     VVS2 2,030.500 4,378.167 2,525.193 3,580.581 3,665.181 3,337.820
##     VVS1 6,761.500 3,889.333 1,945.875 1,426.692 2,960.594 2,608.460
##       IF 3,205.000   817.250 4,675.867 2,361.333 1,961.344 2,411.697
##    Total 4,505.238 4,130.433 3,959.916 4,369.409 3,470.224 3,907.186
```

Summary method for the explore function explore()

Launch a Radiant data window in R Studio

To open aa separate interactive Radiant data window from R Studio, type:

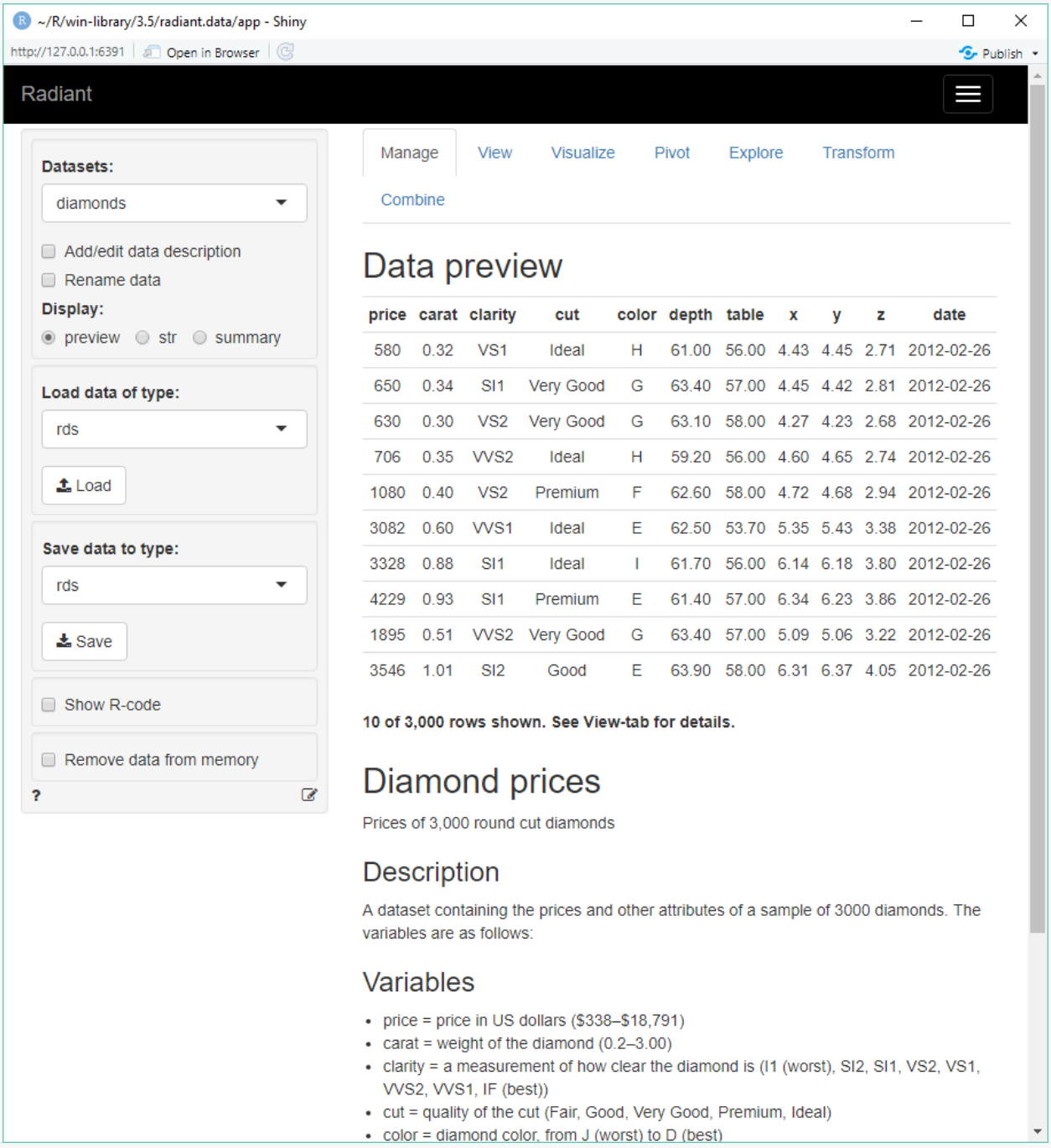
```
radiant.data_window()
```

To open aa separate interactive Radiant data window in the R Studio Viewer, type:

```
radiant.data_viewer()
```

To open aa separate interactive Radiant data window in your web browser from R Studio, type:

```
radiant.data()
```



pivotr: Create a pivot table

```
pivotr(diamonds, cvars = "cut") %>% str()
```

```
## List of 13
##  $ tab_freq   :Classes 'tbl_df', 'tbl' and 'data.frame': 6 obs. of  2
variables:
##   ..$ cut  : Factor w/ 6 levels "Fair","Good",..: 1 2 3 4 5 6
##   ..$ n_obs: int [1:6] 101 275 677 771 1176 3000
##  $ tab        :'data.frame': 6 obs. of  2 variables:
##   ..$ cut  : Factor w/ 6 levels "Fair","Good",..: 1 2 3 4 5 6
##   ..$ n_obs: int [1:6] 101 275 677 771 1176 3000
##   ..- attr(*, "nrow")= num 5
##  $ df_name    : chr "diamonds"
##  $ fill       : int 0
##  $ vars       : chr "cut"
##  $ cvars      : chr "cut"
##  $ nvar       : chr "n_obs"
##  $ fun        : chr "mean"
##  $ normalize  : chr "None"
##  $ tabfilt    : chr ""
##  $ tabsort    : chr ""
##  $ nr         : NULL
##  $ data_filter: chr ""
##  - attr(*, "class")= chr [1:2] "pivotr" "list"
```

```
pivotr(diamonds, cvars = "cut")$tab
```

```
##         cut n_obs
## 1      Fair   101
## 2      Good   275
## 3 Very Good   677
## 4   Premium   771
## 5     Ideal  1176
## 6     Total  3000
```

```
pivotr(diamonds, cvars = c("cut","clarity","color"))$tab
```

```
##    clarity color Fair Good Very_Good Premium Ideal Total
## 1       I1     D    0    1         0       3     0     4
## 2       I1     E    1    1         2       1     0     5
## 3       I1     F    2    1         2       2     4    11
## 4       I1     G    1    1         1       2     0     5
## 5       I1     H    3    0         1       3     1     8
## 6       I1     I    4    0         0       1     0     5
## 7       I1     J    1    0         0       1     0     2
## 8      SI2     D    8   14        18      13    15    68
## 9      SI2     E    5   16        30      25    30   106
## 10     SI2     F    1   11        24      23    34    93
## 11     SI2     G    5    7        29      35    21    97
## 12     SI2     H    4    7        14      31    30    86
## 13     SI2     I    2    2         7      21    14    46
## 14     SI2     J    3    5        10       9     6    33
## 15     SI1     D    4    9        21      38    39   111
## 16     SI1     E    5   22        37      47    36   147
```

```
## 17     SI1     F    4   14        40      25    42   125
## 18     SI1     G    5   10        23      26    31    95
## 19     SI1     H    9   13        21      31    43   117
## 20     SI1     I    1   14        18      20    23    76
## 21     SI1     J    1   11        11       9    18    50
## 22     VS2     D    2    3        17      12    55    89
## 23     VS2     E    2   15        26      41    55   139
## 24     VS2     F    4   12        23      36    53   128
## 25     VS2     G    2    8        17      42    50   119
## 26     VS2     H    2   10        20      25    31    88
## 27     VS2     I    2    2        16      17    19    56
## 28     VS2     J    0    2        12      12    16    42
## 29     VS1     D    0    3        10      13    24    50
## 30     VS1     E    0    5         9      13    25    52
## 31     VS1     F    4    8        25      17    29    83
## 32     VS1     G    2    7        22      30    51   112
## 33     VS1     H    3    3        19      23    22    70
## 34     VS1     I    2    5        11      14    18    50
## 35     VS1     J    2    2         3      10     8    25
## 36    VVS2     D    0    4         8       9    20    41
## 37    VVS2     E    1    1        25       7    22    56
## 38    VVS2     F    0    0        14       6    29    49
## 39    VVS2     G    1    3        21      10    43    78
## 40    VVS2     H    0    3        12       5    13    33
## 41    VVS2     I    0    1         1       4    14    20
## 42    VVS2     J    0    0         2       2     3     7
## 43    VVS1     D    1    1         2       4     9    17
## 44    VVS1     E    0    2         7       7    21    37
## 45    VVS1     F    1    7         4       4    29    45
## 46    VVS1     G    0    3        16       9    42    70
## 47    VVS1     H    0    1         6      11    14    32
## 48    VVS1     I    0    1         4       3    12    20
## 49    VVS1     J    0    0         1       1     1     3
## 50      IF     D    0    0         1       0     1     2
## 51      IF     E    0    0         4       3     5    12
## 52      IF     F    1    2         4       6    18    31
## 53      IF     G    0    1         2       2    16    21
## 54      IF     H    0    0         2       3    15    20
## 55      IF     I    0    1         2       3     5    11
## 56      IF     J    0    0         0       1     1     2
## 57   Total Total  101  275       677     771  1176  3000
```

```
pivotr(diamonds, cvars = "cut:clarity", nvar = "price")$tab
```

```
##   clarity     Fair     Good Very_Good  Premium    Ideal    Total
## 1      I1 2730.167 4333.500  3864.167 4932.231 6078.200 4194.775
## 2     SI2 5893.964 5280.919  5045.621 5568.019 4435.673 5100.189
## 3     SI1 4273.069 3757.022  4277.544 4113.811 3758.125 3998.577
## 4     VS2 3292.000 3925.481  3950.947 4522.914 3306.290 3822.967
## 5     VS1 5110.769 3740.697  3889.475 4461.333 3189.362 3789.181
```

```
## 6    VVS2 2030.500 4378.167  2525.193 3580.581 3665.181 3337.820
## 7    VVS1 6761.500 3889.333  1945.875 1426.692 2960.594 2608.460
## 8      IF 3205.000  817.250  4675.867 2361.333 1961.344 2411.697
## 9   Total 4505.238 4130.433  3959.916 4369.409 3470.224 3907.186
```

```
pivotr(diamonds, cvars = "cut", nvar = "price")$tab
```

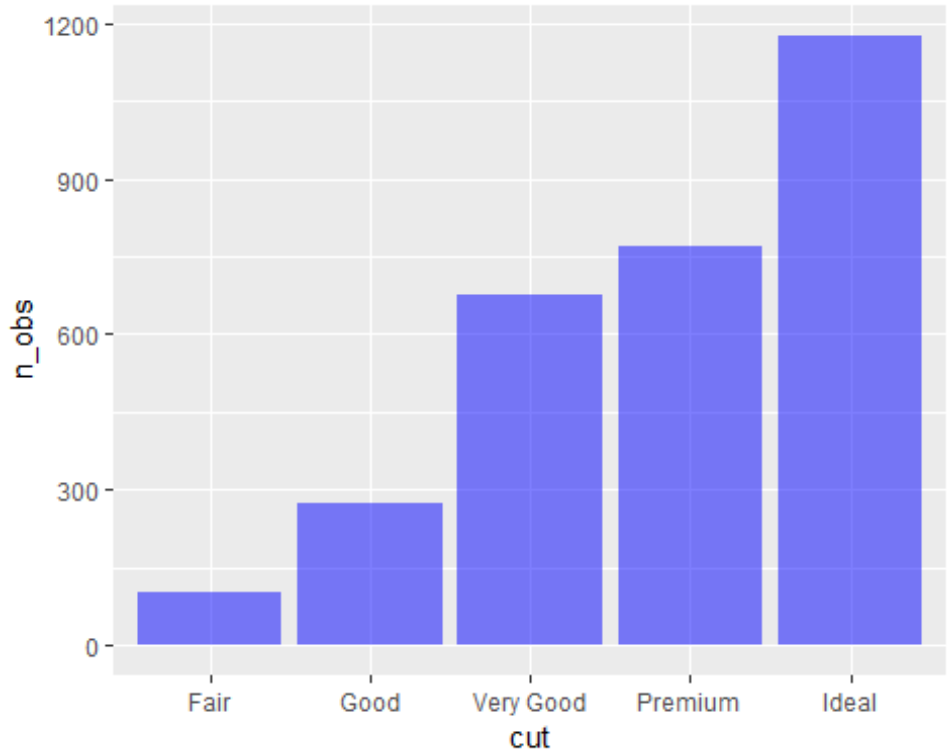
```
##         cut    price
## 1      Fair 4505.238
## 2      Good 4130.433
## 3 Very Good 3959.916
## 4   Premium 4369.409
## 5     Ideal 3470.224
## 6     Total 3907.186
```

```
pivotr(diamonds, cvars = "cut", normalize = "total")$tab
```

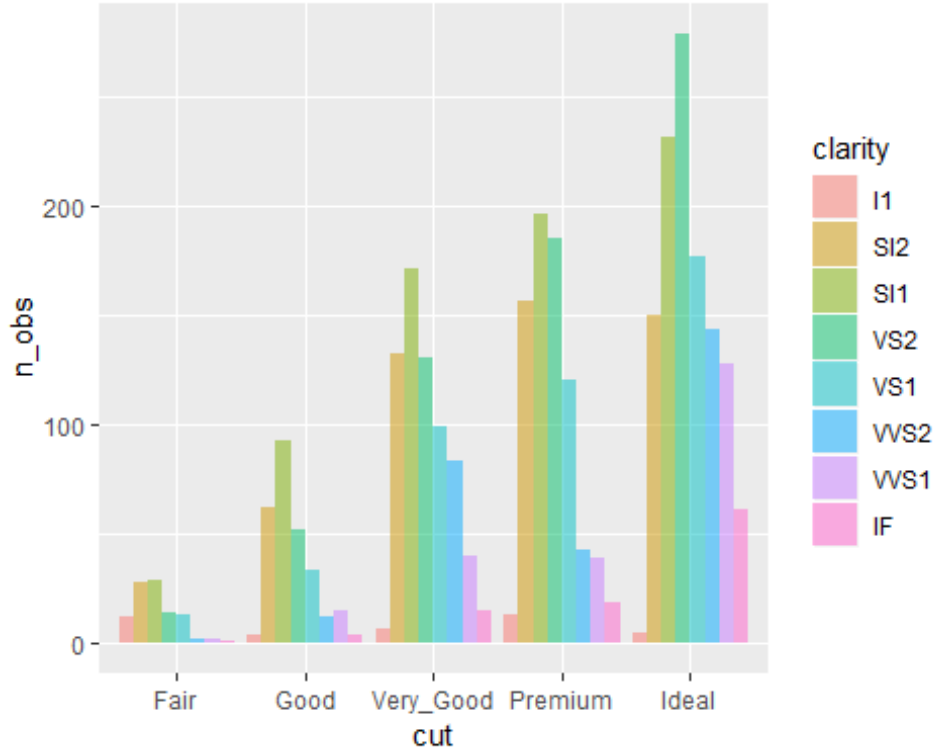
```
##         cut      n_obs
## 1      Fair 0.03366667
## 2      Good 0.09166667
## 3 Very Good 0.22566667
## 4   Premium 0.25700000
## 5     Ideal 0.39200000
## 6     Total 1.00000000
```

plot.pivotr: Plot method for the pivotr function

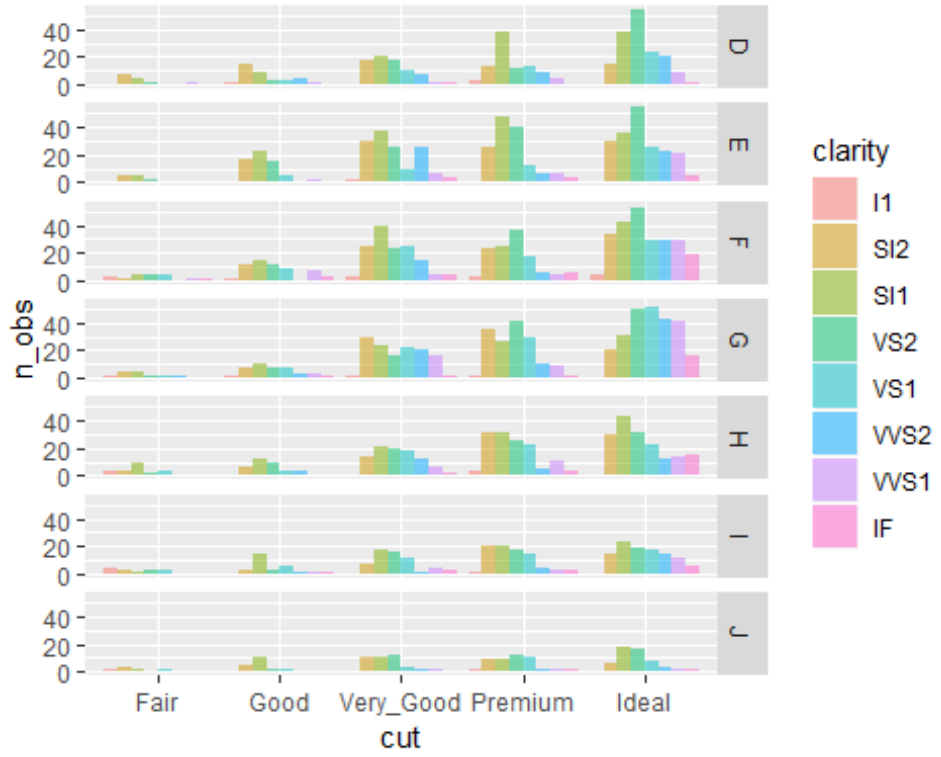
```
pivotr(diamonds, cvars = "cut") %>% plot()
```



```
pivotr(diamonds, cvars = c("cut", "clarity")) %>% plot()
```




```
pivotr(diamonds, cvars = c("cut", "clarity", "color")) %>% plot()
```



explore: Explore and summarize data

```
explore(diamonds, c("price", "carat")) %>% str()
```

```
## List of 10
##  $ tab        :'data.frame': 2 obs. of  3 variables:
##   ..$ variable: Factor w/ 2 levels "price","carat": 1 2
##   ..$ mean    : num [1:2] 3907.186 0.794
##   ..$ sd      : num [1:2] 3956.915 0.474
##   ..- attr(*, "nrow")= int 2
##  $ df_name    : chr "diamonds"
##  $ vars       : chr [1:2] "price" "carat"
##  $ byvar      : chr ""
##  $ fun        : chr [1:2] "mean" "sd"
##  $ top        : chr "fun"
##  $ tabfilt    : chr ""
##  $ tabsort    : chr ""
##  $ nr         : NULL
##  $ data_filter: chr ""
##  - attr(*, "class")= chr [1:2] "explore" "list"
```

```
explore(diamonds, "price:x")$tab
```

```
##   variable         mean           sd
## 1    price 3.907186e+03 3956.9154001
## 2    carat 7.942833e-01    0.4738263
## 3  clarity 1.333333e-02    0.1147168
## 4      cut 3.366667e-02    0.1803998
## 5    color 1.273333e-01    0.3334016
## 6    depth 6.175267e+01    1.4460279
## 7    table 5.746533e+01    2.2411022
## 8        x 5.721823e+00    1.1240545
```

```
explore(diamonds, c("price","carat"), byvar = "cut", fun = c("n_missing",
"skew"))$tab
```

```
## Warning: 'glue::collapse' is deprecated.
## Use 'glue_collapse' instead.
## See help("Deprecated") and help("glue-deprecated").
```

```
##          cut variable n_missing      skew
## 1       Fair    price         0 1.5741334
## 2       Fair    carat         0 0.9285670
## 3       Good    price         0 1.4885765
## 4       Good    carat         0 1.0207909
## 5  Very Good    price         0 1.6007752
## 6  Very Good    carat         0 0.9370738
## 7    Premium    price         0 1.4131786
## 8    Premium    carat         0 0.9299567
## 9      Ideal    price         0 1.7986601
## 10     Ideal    carat         0 1.3654745
```

dtab.pivotr: Make an interactive pivot table

```
pivotr(diamonds, cvars = "cut") %>% dtab()
```

cut	n_obs
All	All
Fair	101
Good	275
Very Good	677
Premium	771
Ideal	1,176
Total	**3,000**

```
pivotr(diamonds, cvars = c("cut","clarity")) %>% dtab(format = "color_bar")
```

	cut					
clarity	Fair	Good	Very_Good	Premium	Ideal	Total
All	All	All	All	All	All	All
I1	12	4	6	13	5	**40**
SI2	28	62	132	157	150	**529**
SI1	29	93	171	196	232	**721**
VS2	14	52	131	185	279	**661**
VS1	13	33	99	120	177	**442**
VVS2	2	12	83	43	144	**284**
VVS1	2	15	40	39	128	**224**
IF	1	4	15	18	61	**99**
Total	**101**	**275**	**677**	**771**	**1,176**	**3,000**

```
pivotr(diamonds, cvars = c("cut","clarity"), normalize = "total") %>%
dtab(format = "color_bar", perc = TRUE)
```

	cut					
clarity	Fair	Good	Very_Good	Premium	Ideal	Total
All	All	All	All	All	All	All
I1	0.400%	0.133%	0.200%	0.433%	0.167%	**1.333%**
SI2	0.933%	2.067%	4.400%	5.233%	5.000%	**17.633%**
SI1	0.967%	3.100%	5.700%	6.533%	7.733%	**24.033%**
VS2	0.467%	1.733%	4.367%	6.167%	9.300%	**22.033%**
VS1	0.433%	1.100%	3.300%	4.000%	5.900%	**14.733%**
VVS2	0.067%	0.400%	2.767%	1.433%	4.800%	**9.467%**
VVS1	0.067%	0.500%	1.333%	1.300%	4.267%	**7.467%**
IF	0.033%	0.133%	0.500%	0.600%	2.033%	**3.300%**
Total	**3.367%**	**9.167%**	**22.567%**	**25.700%**	**39.200%**	**100.000%**

dtab.data.frame: Create an interactive table to view, search, sort, and ???lter data

```
dtab(mtcars)
```

Show 10 entries

Search:

mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
All	All	All	All	All	All	All	All	All	All	All
21.000	6.000	160.000	110.000	3.900	2.620	16.460	0.000	1.000	4.000	4.000
21.000	6.000	160.000	110.000	3.900	2.875	17.020	0.000	1.000	4.000	4.000
22.800	4.000	108.000	93.000	3.850	2.320	18.610	1.000	1.000	4.000	1.000
21.400	6.000	258.000	110.000	3.080	3.215	19.440	1.000	0.000	3.000	1.000
18.700	8.000	360.000	175.000	3.150	3.440	17.020	0.000	0.000	3.000	2.000
18.100	6.000	225.000	105.000	2.760	3.460	20.220	1.000	0.000	3.000	1.000
14.300	8.000	360.000	245.000	3.210	3.570	15.840	0.000	0.000	3.000	4.000
24.400	4.000	146.700	62.000	3.690	3.190	20.000	1.000	0.000	4.000	2.000
22.800	4.000	140.800	95.000	3.920	3.150	22.900	1.000	0.000	4.000	2.000
19.200	6.000	167.600	123.000	3.920	3.440	18.300	1.000	0.000	4.000	4.000

Showing 1 to 10 of 32 entries

Previous 1 2 3 4 Next

combine_data: Combine datasets using dplyr’s bind and join functions

```
library(dplyr)
data("superheroes")
```

```
data("avengers")
avengers %>% combine_data(superheroes, type = "bind_cols")
```

name <chr>	alignment <chr>	gender <chr>	publisher <chr>	name1 <chr>	alignment1 <chr>	gender1 <chr>
Thor	good	male	Marvel	Magneto	bad	male
Iron Man	good	male	Marvel	Storm	good	female
Hulk	good	male	Marvel	Mystique	bad	female
Hawkeye	good	male	Marvel	Batman	good	male
Black Widow	good	female	Marvel	Joker	bad	male
Captain America	good	male	Marvel	Catwoman	bad	female
Magneto	bad	male	Marvel	Hellboy	good	male

7 rows | 1-7 of 8 columns

```
combine_data(avengers, superheroes, type = "bind_cols")

##              name alignment gender publisher    name1 alignment1 gender1
## 1            Thor      good   male    Marvel  Magneto        bad    male
## 2        Iron Man      good   male    Marvel    Storm       good  female
## 3            Hulk      good   male    Marvel Mystique        bad  female
## 4         Hawkeye      good   male    Marvel   Batman       good    male
## 5     Black Widow      good female    Marvel    Joker        bad    male
## 6 Captain America      good   male    Marvel Catwoman        bad  female
## 7         Magneto       bad   male    Marvel  Hellboy       good    male
##          publisher1
## 1            Marvel
## 2            Marvel
## 3            Marvel
## 4                DC
## 5                DC
## 6                DC
## 7 Dark Horse Comics

avengers %>% combine_data(superheroes, type = "bind_rows")

##               name alignment gender         publisher
## 1             Thor      good   male            Marvel
## 2         Iron Man      good   male            Marvel
## 3             Hulk      good   male            Marvel
## 4          Hawkeye      good   male            Marvel
## 5      Black Widow      good female            Marvel
## 6  Captain America      good   male            Marvel
## 7          Magneto       bad   male            Marvel
## 8          Magneto       bad   male            Marvel
## 9            Storm      good female            Marvel
## 10        Mystique       bad female            Marvel
## 11          Batman      good   male                DC
```

```
## 12           Joker       bad   male                DC
## 13        Catwoman       bad female                DC
## 14         Hellboy      good   male Dark Horse Comics
```

```
avengers %>% combine_data(superheroes, add = "publisher", type = "bind_rows")
```

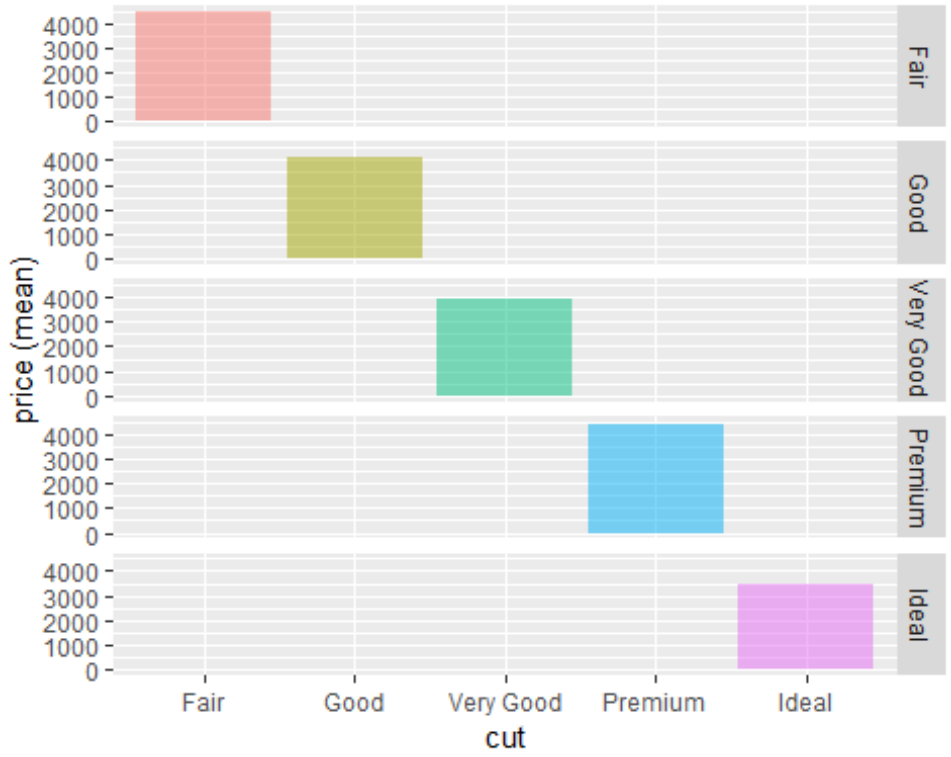
```
##                name alignment gender         publisher
## 1              Thor      good   male            Marvel
## 2         Iron Man      good   male            Marvel
## 3              Hulk      good   male            Marvel
## 4           Hawkeye      good   male            Marvel
## 5       Black Widow      good female            Marvel
## 6  Captain America      good   male            Marvel
## 7           Magneto       bad   male            Marvel
## 8           Magneto       bad   male            Marvel
## 9             Storm      good female            Marvel
## 10         Mystique       bad female            Marvel
## 11           Batman      good   male                DC
## 12           Joker       bad   male                DC
## 13        Catwoman       bad female                DC
## 14         Hellboy      good   male Dark Horse Comics
```

choose_files: Choose files interactively

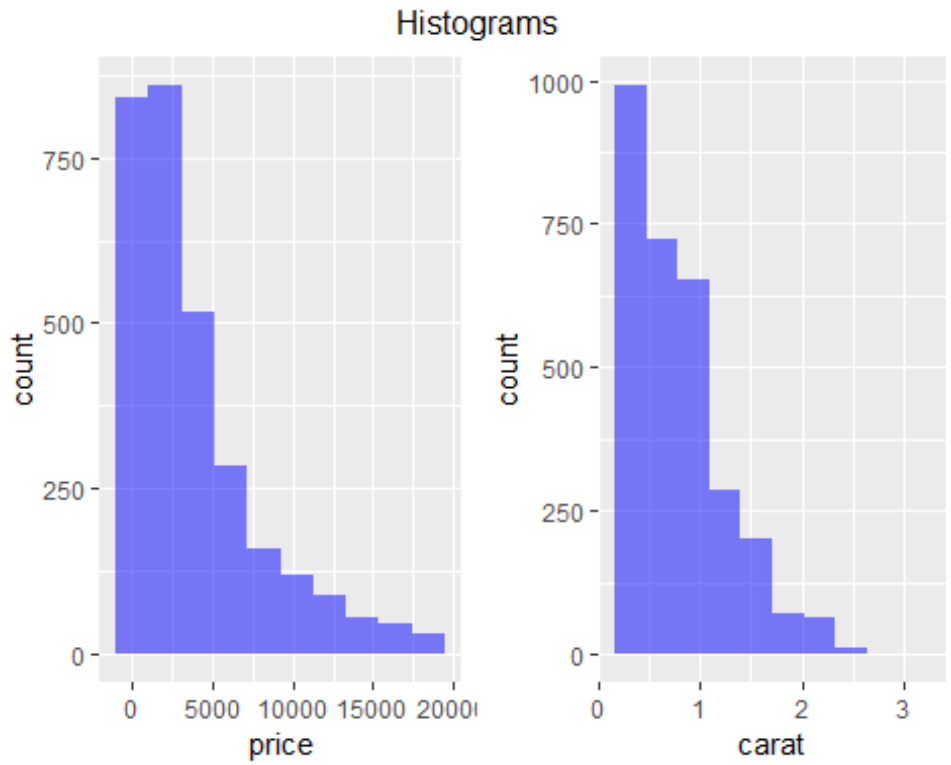
```
choose_files("csv")
```

visualize: Visualize data using ggplot2 http://ggplot2.tidyverse.org

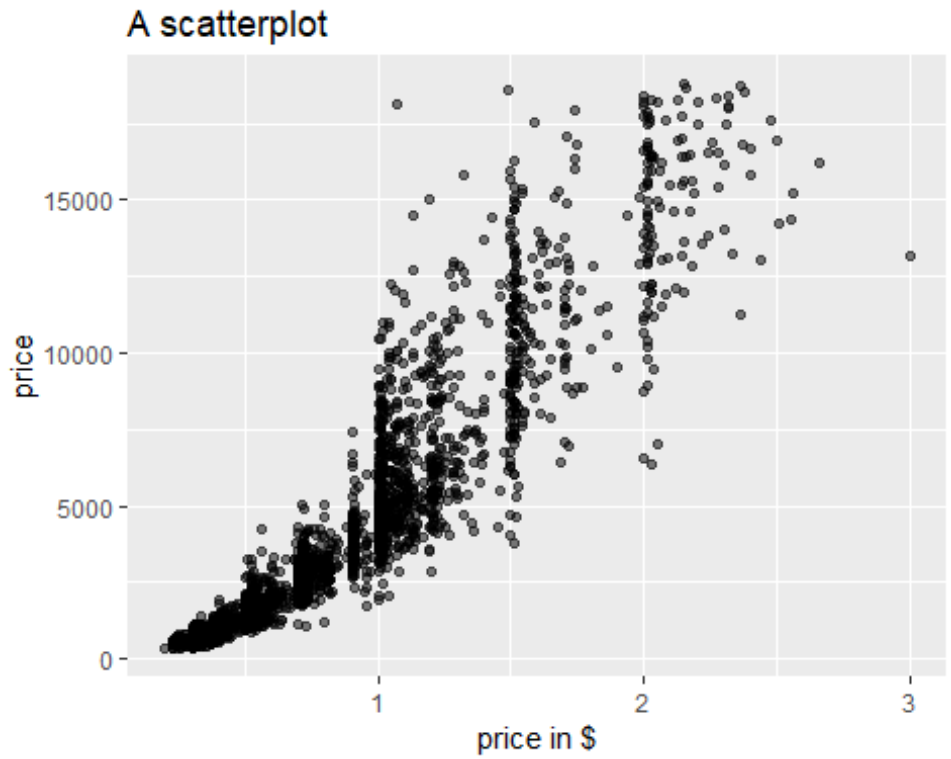
```
visualize(diamonds, xvar = "cut", yvar = "price", type = "bar", facet_row =
"cut", fill = "cut")
```



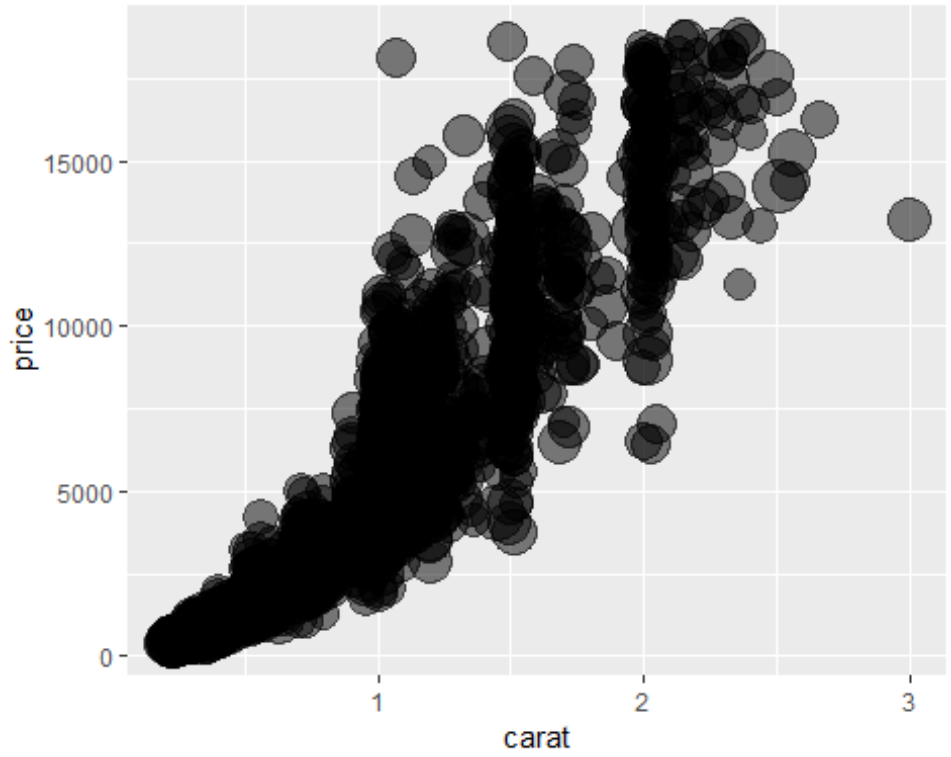

```
visualize(diamonds, xvar = "price:carat", custom = TRUE) %>%
gridExtra::grid.arrange(grobs = ., top = "Histograms", ncol = 2)
```



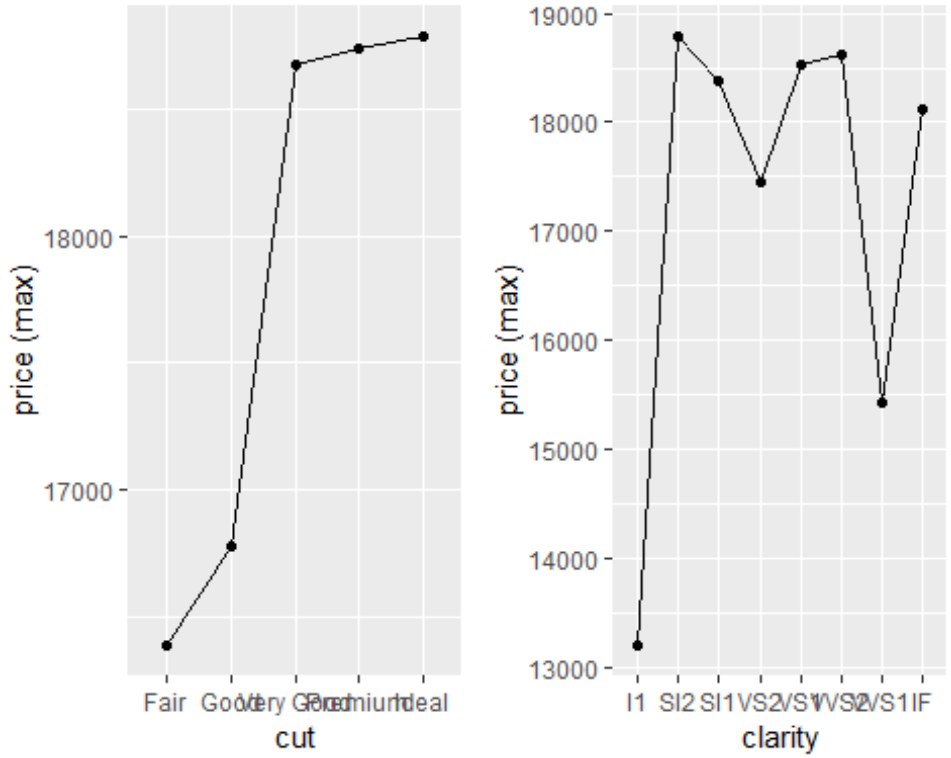
```
visualize(diamonds, yvar = "price", xvar = "carat", type = "scatter", custom
= TRUE) + labs(title = "A scatterplot", x = "price in $")
```



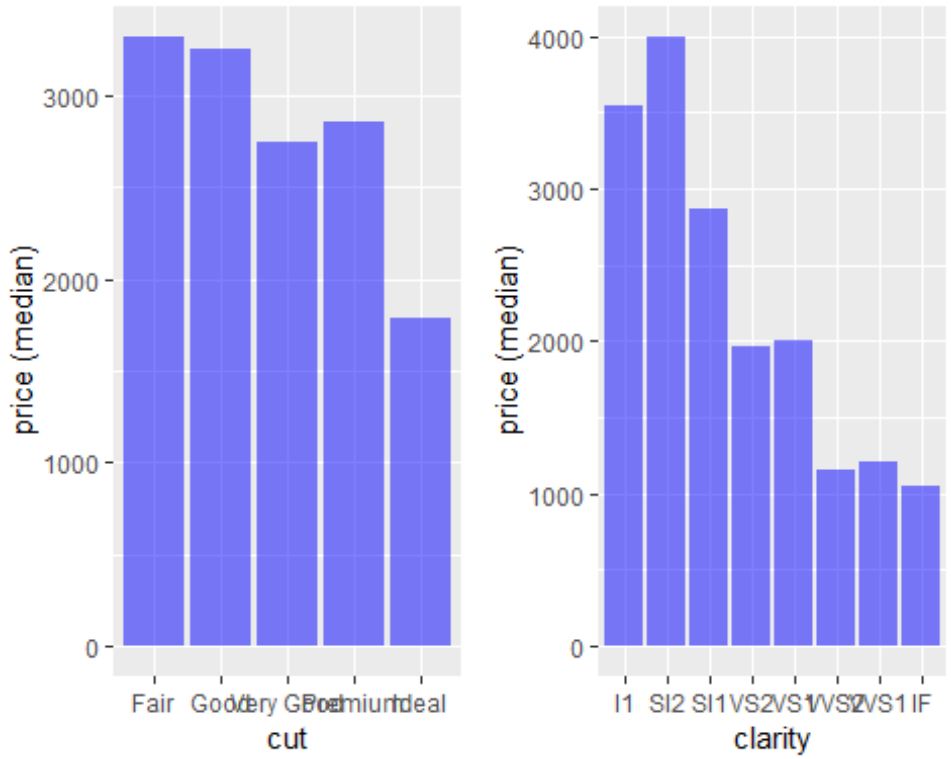

```
visualize(diamonds, yvar = "price", xvar = "carat", type = "scatter", size =
"table", custom = TRUE) + scale_size(range=c(1,10), guide = "none")
```



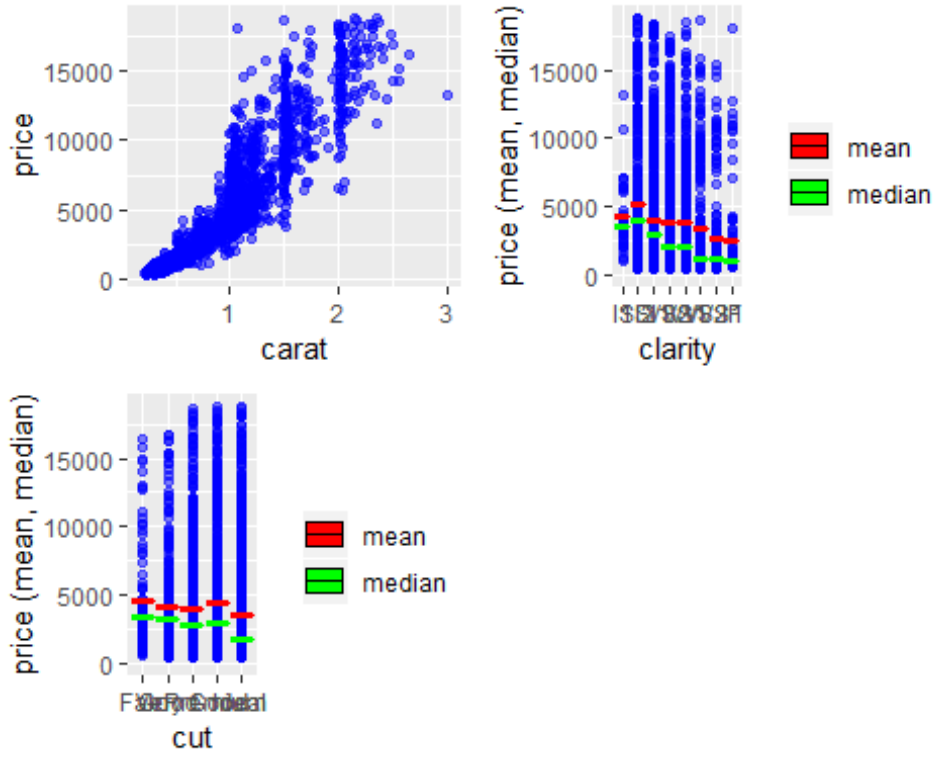

```
visualize(diamonds, yvar = "price", xvar = c("cut","clarity"), type = "line",
fun = "max")
```



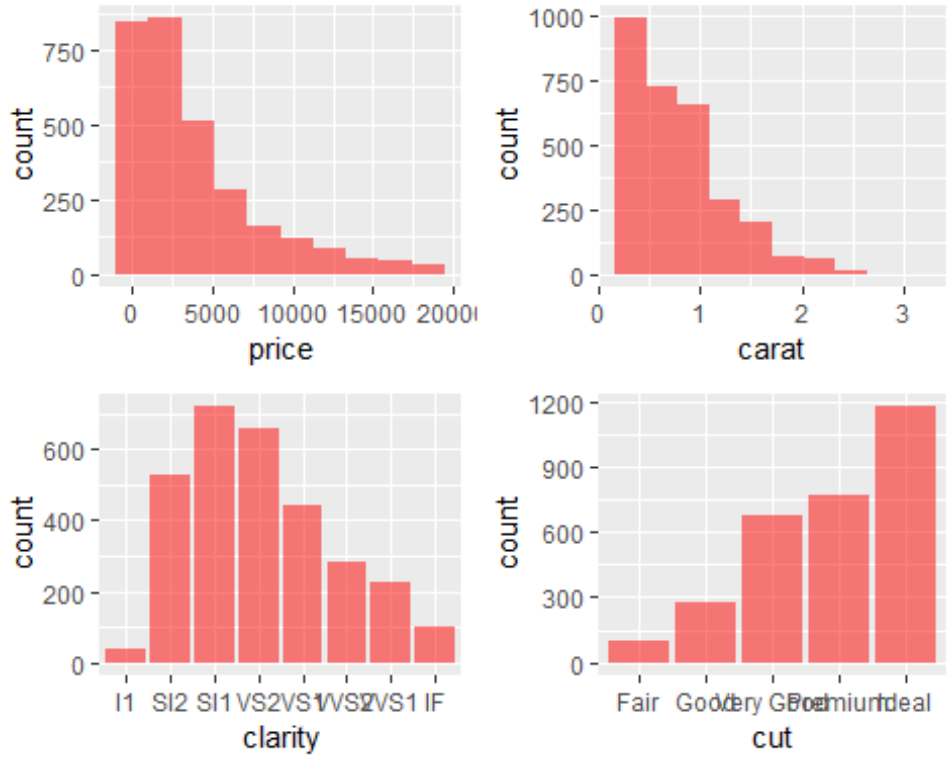
```
visualize(diamonds, yvar = "price", xvar = c("cut","clarity"), type = "bar",
fun = "median")
```




```
visualize(diamonds, "carat:cut", yvar = "price", type = "scatter", pointcol =
"blue", fun = c("mean", "median"), linecol = c("red","green"))
```




```
visualize(diamonds, "price:cut", type = "dist", fillcol = "red")
```



view_data: View data in a shiny-app

```
view_data(mtcars)
```

table2data: Create data.frame from a table

```
df<-data.frame(price = c("$200","$300"), sale = c(10, 2)) %>% table2data()

head(df)

##   price
## 1  $200
## 2  $200
## 3  $200
## 4  $200
## 5  $200
## 6  $200

library(readr)
library(plyr)

## ------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first,
then dplyr:
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------

##
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following object is masked from 'package:lubridate':
##
##     here

path <- "c:\\Users\\jeff\\Documents\\Crime
Analysis\\India_data\\chennai_crimes.csv"
df <- read_csv(path)

## Parsed with column specification:
## cols(
##   IncidntNum = col_integer(),
##   Category = col_character(),
##   Descript = col_character(),
##   DayOfWeek = col_character(),
##   Date = col_character(),
```

```
##   Time = col_time(format = ""),
##   PdDistrict = col_character(),
##   Resolution = col_character(),
##   Address = col_character(),
##   X = col_double(),
##   Y = col_double(),
##   Location = col_character(),
##   PdId = col_double()
## )

head(df)

## # A tibble: 6 x 13
##   IncidntNum Category Descript DayOfWeek Date  Time  PdDistrict Resolution
##        <int> <chr>    <chr>    <chr>     <chr> <tim> <chr>      <chr>
## 1  150098210 ROBBERY  ROBBERY~ Sunday    2/1/~ 15:45 Zone4      NONE
## 2  150098210 AGG_ASS~ AGGRAVA~ Sunday    2/1/~ 15:45 Zone4      NONE
## 3  150098260 LARCENY~ PETTY T~ Saturday  1/31~ 17:00 Zone3      ARREST
## 4  150098345 LARCENY~ PETTY T~ Sunday    2/1/~ 14:00 Zone5      ARREST
## 5  150098367 ROBBERY  ROBBERY~ Sunday    2/1/~ 16:20 Zone5      ARREST
## 6  150098395 LARCENY~ PETTY T~ Sunday    2/1/~ 14:30 Zone7      ARREST
## # ... with 5 more variables: Address <chr>, X <dbl>, Y <dbl>,
## #   Location <chr>, PdId <dbl>

summary(df$Resolution)

##    Length     Class      Mode
##    499365 character character

BB<-baseball
head(BB)

##            id year stint team lg  g  ab  r  h X2b X3b hr rbi sb cs bb so
## 4   ansonca01 1871     1  RC1    25 120 29 39  11   3  0  16  6  2  2  1
## 44  forceda01 1871     1  WS3    32 162 45 45   9   4  0  29  8  0  4  0
## 68  mathebo01 1871     1  FW1    19  89 15 24   3   1  0  10  2  1  2  0
## 99  startjo01 1871     1  NY2    33 161 35 58   5   1  1  34  4  2  3  0
## 102 suttoez01 1871     1  CL1    29 128 35 45   3   7  3  23  3  1  1  0
## 106 whitede01 1871     1  CL1    29 146 40 47   6   5  1  21  2  2  4  1
##     ibb hbp sh sf gidp
## 4    NA  NA NA NA   NA
## 44   NA  NA NA NA   NA
## 68   NA  NA NA NA   NA
## 99   NA  NA NA NA   NA
## 102  NA  NA NA NA   NA
## 106  NA  NA NA NA   NA
```

Difference between mutate and transform

```
system.time(mutate(baseball, avg_ab = ab / g))
```

```
##    user  system elapsed
##       0       0       0

system.time(transform(baseball, avg_ab = ab / g))

##    user  system elapsed
##    0.01    0.00    0.01
```

Text manipulation

Load necessary libraries
```
library(readr)
library(broom)
library(dplyr)
library(tm)
library(tidyverse)
library(tidytext)
library(tidyr)
library(tmap)
library(purrr)
library(readr)
library(stringr)
library(tibble)
library(reshape2)
library(wordcloud)
library(wordcloud2)
```

Which of the libraries above are required in order to run the following code?

```
mypath =
("C:/Users/jeff/Documents/VIT_Course_Material/Data_Analytics_2018/data
/ind_phil/Indian_Philosophy_I_and_II.txt")

ind_text <- read_csv(mypath)
summary(ind_text, n=1, showmeta=TRUE, tolower=TRUE)
```

Will "mypath" (the way is is definedd) pose problems for you?

What does the following code do?

```
read_folder <- function(infolder) {
  data_frame(file = dir(infolder, full.names = TRUE)) %>%
    mutate(text = map(file, read_lines)) %>%
    transmute(id = basename(file), text) %>%
    unnest(text)
```

```
}

ind_text <-
read_folder("users/jeff/Documents/VIT_Cource_Material/data/ind_phil")
summary(ind_text)
```