chapter10 : Data Transformation

```
library(ggplot2)
library(gridExtra)
library(dplyr)
```

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

```
## The following object is masked from 'package:gridExtra':
##
##     combine
```

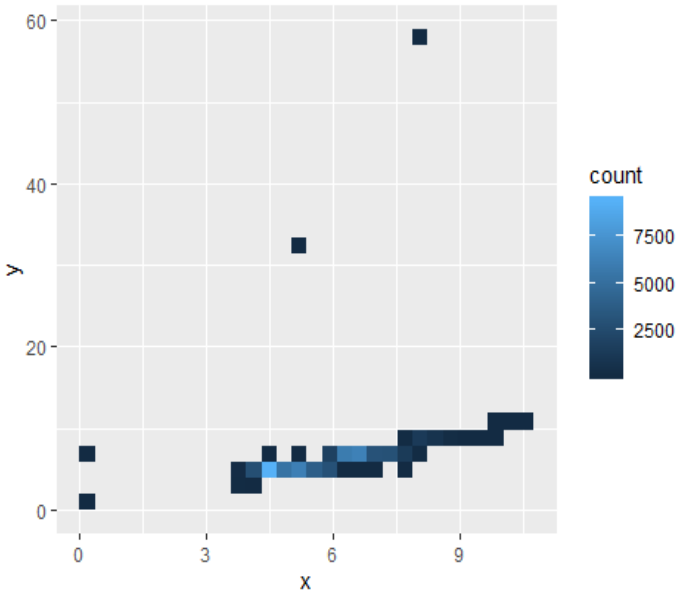
```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

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

10.1 Introduction

10.2 Filter observations

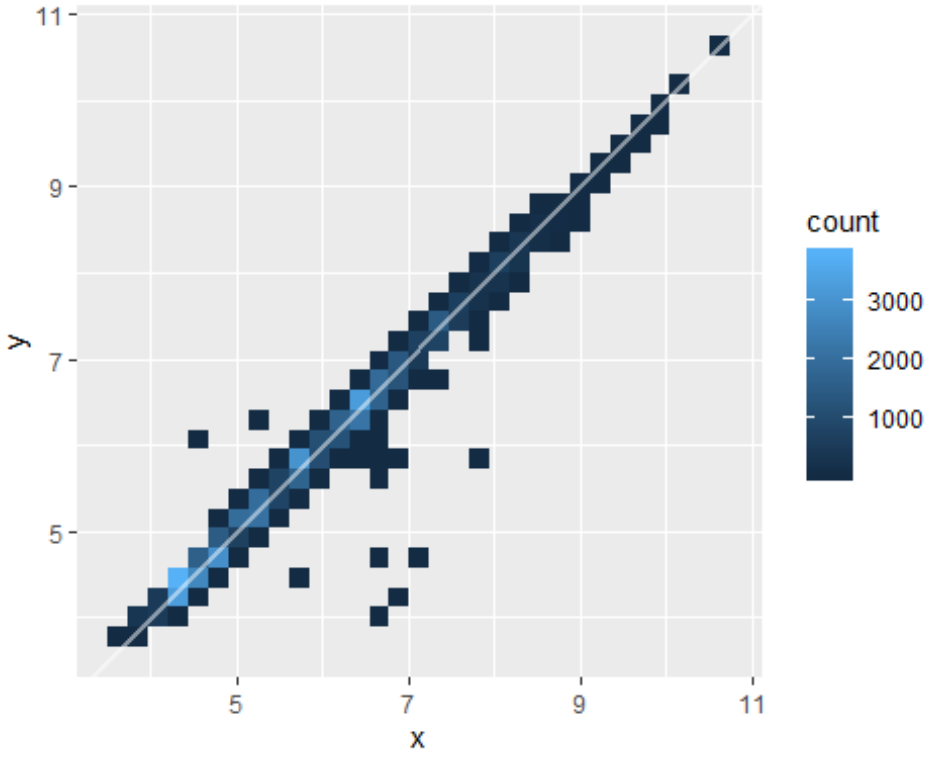
```
ggplot(diamonds, aes(x, y)) +
  geom_bin2d()
```



```r
filter(diamonds, x==0 | y == 0)
```

```
## # A tibble: 8 x 10
##   carat cut       color clarity depth table price     x     y     z
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  1.07 Ideal     F     SI2      61.6    56  4954     0  6.62     0
## 2  1    Very Good H     VS2      63.3    53  5139     0  0        0
## 3  1.14 Fair      G     VS1      57.5    67  6381     0  0        0
## 4  1.56 Ideal     G     VS2      62.2    54 12800     0  0        0
## 5  1.2  Premium   D     VVS1     62.1    59 15686     0  0        0
## 6  2.25 Premium   H     SI2      62.8    59 18034     0  0        0
## 7  0.71 Good      F     SI2      64.1    60  2130     0  0        0
## 8  0.71 Good      F     SI2      64.1    60  2130     0  0        0
```

```r
diamonds_ok <- filter(diamonds, x > 0, y > 0, y < 20)
ggplot(diamonds_ok, aes(x, y)) +
  geom_bin2d() +
  geom_abline(slope = 1, colour = "white", size = 1, alpha = 0.5)
```



10.2.1 Useful Tools

- first argument to *filter()* is a data frame.
- second and subsequent arguments must be logical vectors.

: filter() selects every row where all the logical expressions are *TRUE* !!

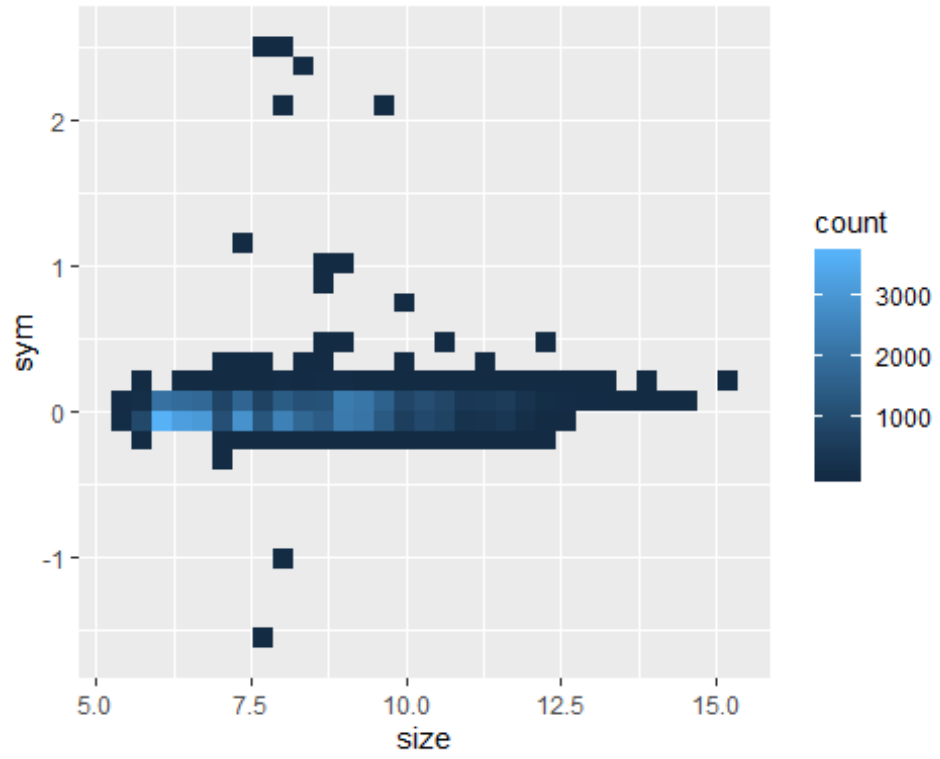
10.2.2 Missing Values

- in *filter()*, *NA* values are automatically dropped.

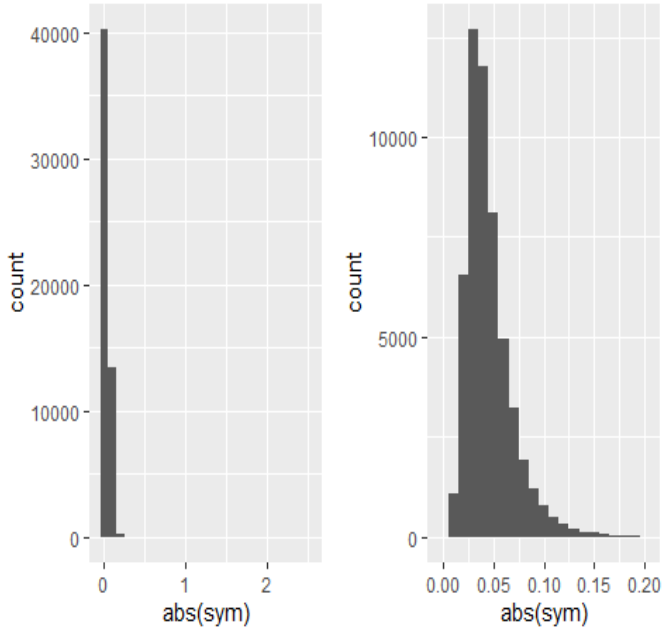
```
x <- c(1, NA, 2)
is.na(x)

## [1] FALSE  TRUE FALSE
```

10.3 Create New Variables

```
diamonds_ok2 <- mutate(diamonds_ok,
  sym = x - y,
  size = sqrt(x ^ 2 + y ^ 2)
)
diamonds_ok2

## # A tibble: 53,930 x 12
##    carat cut    color clarity depth table price     x     y     z      sym
  size
##    <dbl> <ord>  <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>    <dbl>
 <dbl>
##  1 0.23  Ideal  E     SI2      61.5    55   326  3.95  3.98  2.43 -0.0300
  5.61
##  2 0.21  Premi… E     SI1      59.8    61   326  3.89  3.84  2.31  0.05
  5.47
##  3 0.23  Good   E     VS1      56.9    65   327  4.05  4.07  2.31 -0.02
  5.74
##  4 0.290 Premi… I     VS2      62.4    58   334  4.2   4.23  2.63 -0.03
  5.96
##  5 0.31  Good   J     SI2      63.3    58   335  4.34  4.35  2.75 -0.01000
  6.14
##  6 0.24  Very … J     VVS2     62.8    57   336  3.94  3.96  2.48 -0.02
  5.59
##  7 0.24  Very … I     VVS1     62.3    57   336  3.95  3.98  2.47 -0.0300
  5.61
##  8 0.26  Very … H     SI1      61.9    55   337  4.07  4.11  2.53 -0.04
  5.78
##  9 0.22  Fair   E     VS2      65.1    61   337  3.87  3.78  2.49  0.09
  5.41
## 10 0.23  Very … H     VS1      59.4    61   338  4     4.05  2.39 -0.0500
  5.69
## # … with 53,920 more rows

ggplot(diamonds_ok2, aes(size, sym)) +
  stat_bin2d()
```




```
g1 = ggplot(diamonds_ok2, aes(abs(sym))) +
  geom_histogram(binwidth = 0.10)
diamonds_ok3 <- filter(diamonds_ok2, abs(sym) < 0.20)
g2 = ggplot(diamonds_ok3, aes(abs(sym))) +
  geom_histogram(binwidth = 0.01)
grid.arrange(g1,g2,ncol=2)
```



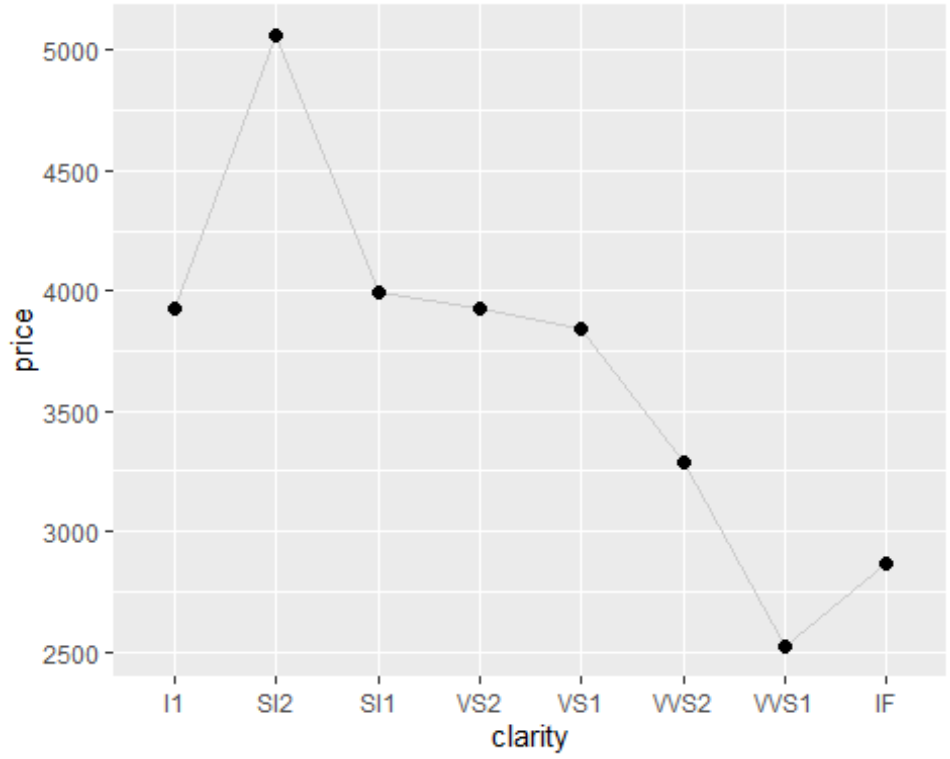
10.4 Group-wise Summaries

dplyr does summaries in two steps: 1. Define the grouping variables with *group by()* 2. Describe how to summarise each group with a single row with *summarise()*

```
by_clarity <- group_by(diamonds, clarity)
sum_clarity <- summarise(by_clarity, price = mean(price))
head(sum_clarity)
```

```
## # A tibble: 6 x 2
##   clarity price
##   <ord>   <dbl>
## 1 I1      3924.
## 2 SI2     5063.
## 3 SI1     3996.
## 4 VS2     3925.
## 5 VS1     3839.
## 6 VVS2    3284.
```

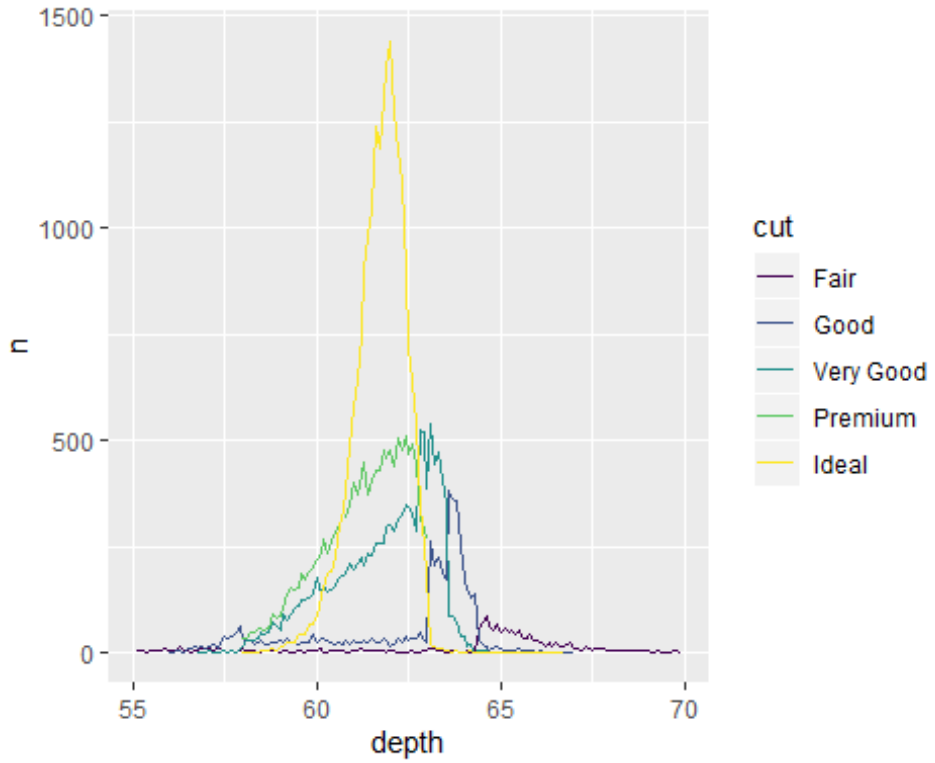
```
ggplot(sum_clarity, aes(clarity, price)) +
  geom_line(aes(group = 1), colour = "grey80") +
  geom_point(size = 2)
```



```r
cut_depth <- summarise(group_by(diamonds, cut, depth), n = n())
cut_depth <- filter(cut_depth, depth > 55, depth < 70)
head(cut_depth)
```

```
## # A tibble: 6 x 3
## # Groups:   cut [1]
##   cut   depth     n
##   <ord> <dbl> <int>
## 1 Fair   55.1     3
## 2 Fair   55.2     6
## 3 Fair   55.3     5
## 4 Fair   55.4     2
## 5 Fair   55.5     3
## 6 Fair   55.6     4
```

```r
ggplot(cut_depth, aes(depth, n, colour = cut)) +
  geom_line()
```



10.4.1 Useful Tools

summarise() needs to be used with functions that take a vector of n values and always return a single value - Counts: n(), n distinct(x) - Middle: mean(x), median(x) - Spread: sd(x), mad(x), IQR(x) - Extremes: quartile(x), min(x), max(x) - Positions: first(x), last(x), nth(x, 2)

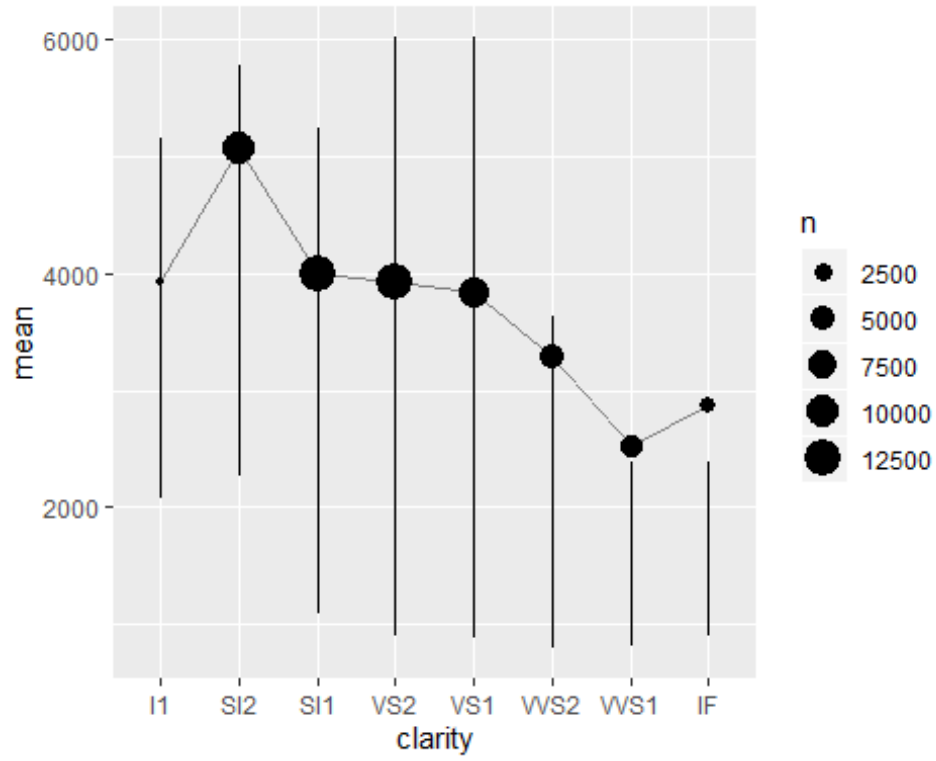
10.4.2 Statistical Considerations

The following example extends our previous summary of the average price by clarity to also include the number of observations in each group, and the upper and lower quartiles. It suggests the mean might be a bad summary for this data - the distributions of price are so highly skewed that the mean is higher than the upper quartile for some of the groups!

```
by_clarity = diamonds %>%
  group_by(clarity) %>%
  summarise(
    n = n(),
    mean = mean(price),
    lq = quantile(price, 0.25),
    uq = quantile(price, 0.75)
  )
by_clarity
```

```
## # A tibble: 8 x 5
##   clarity     n  mean    lq    uq
##   <ord>   <int> <dbl> <dbl> <dbl>
## 1 I1        741 3924. 2080  5161
## 2 SI2      9194 5063. 2264  5777.
## 3 SI1     13065 3996. 1089  5250
## 4 VS2     12258 3925.  900  6024.
## 5 VS1      8171 3839.  876  6023
## 6 VVS2     5066 3284.  794. 3638.
## 7 VVS1     3655 2523.  816  2379
## 8 IF       1790 2865.  895  2388.
```

```
ggplot(by_clarity, aes(clarity, mean)) +
  geom_linerange(aes(ymin = lq, ymax = uq)) +
  geom_line(aes(group = 1), colour = "grey50") +
  geom_point(aes(size = n))
```



10.5 Transformation Pipelines

```
cut_depth <- group_by(diamonds, cut, depth)
cut_depth <- summarise(cut_depth, n = n())
cut_depth <- filter(cut_depth, depth > 55, depth < 70)
cut_depth <- mutate(cut_depth, prop = n / sum(n))
```

Alternative approach with the pipe %>% !!

```
cut_depth <- diamonds %>%
  group_by(cut, depth) %>%
  summarise(n = n()) %>%
  filter(depth > 55, depth < 70) %>%
  mutate(prop = n / sum(n))
```