Exercises-1: ggplot2

Nathan Green, Imperial College London

07/09/2019

In this practical we are going to explore the functionality available in `ggplot2`.

First read in the data.

```
library(dataPakistan)
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.5.3
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.5.3
```

```
##
## 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
```

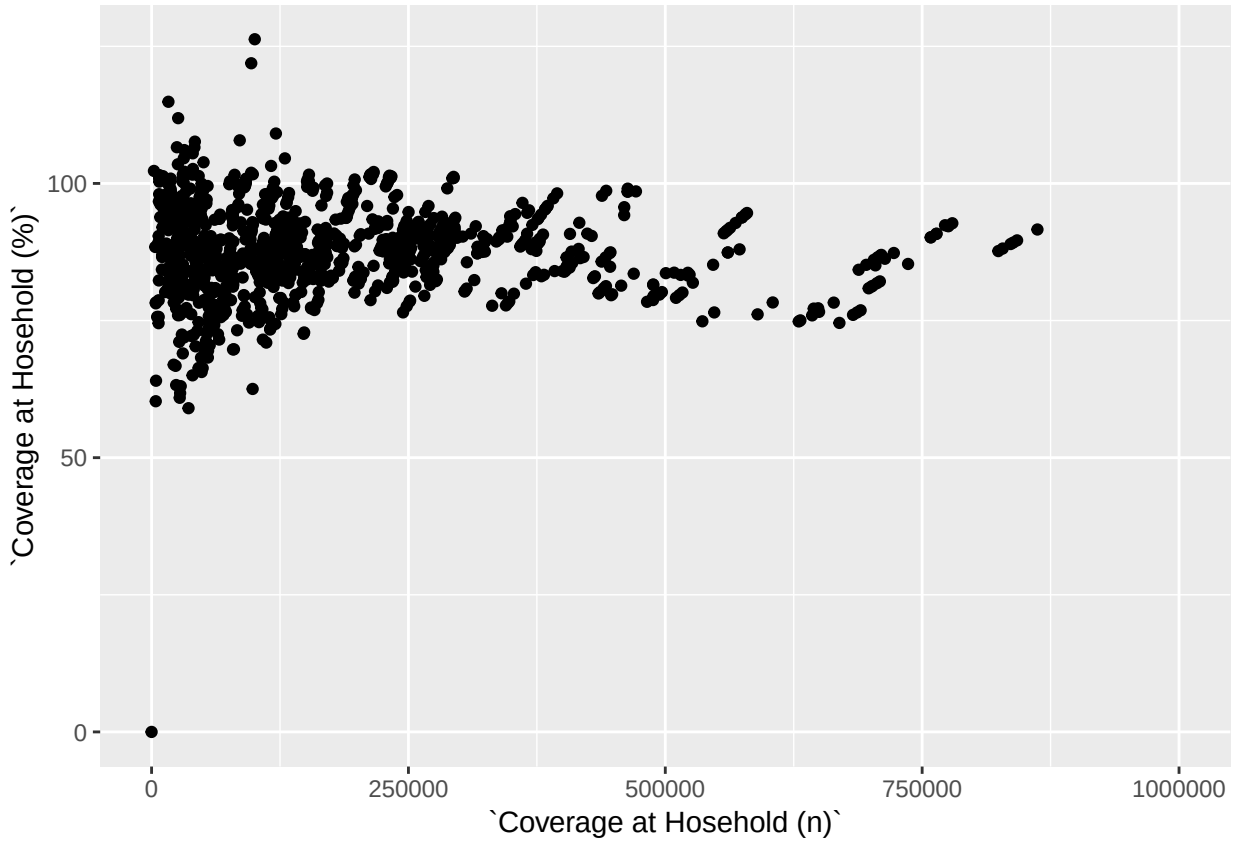
```
file_location <- system.file("extdata", package = "dataPakistan")
Admin <- readxl::read_excel(path = paste0(file_location, "/Admin-datasheet-year2018.xlsx"))
AFP <- readxl::read_excel(path = paste0(file_location, "/List of AFP Cases 2015-2019.xlsx"),
                          sheet = "Data")
pop <- readxl::read_excel(path =
              paste0(file_location, "/Population under 15-estimates 2018-19 as 190510.xlsx"))
```

Scatter plot

Create a scatter plot with defined x-axis limits.

```
ggplot(data = Admin, mapping = aes(x = `Coverage at Hosehold (n)`,
                                   y = `Coverage at Hosehold (%)`)) + geom_point() + xlim(0, 1e+6)
```

```
## Warning: Removed 21 rows containing missing values (geom_point).
```



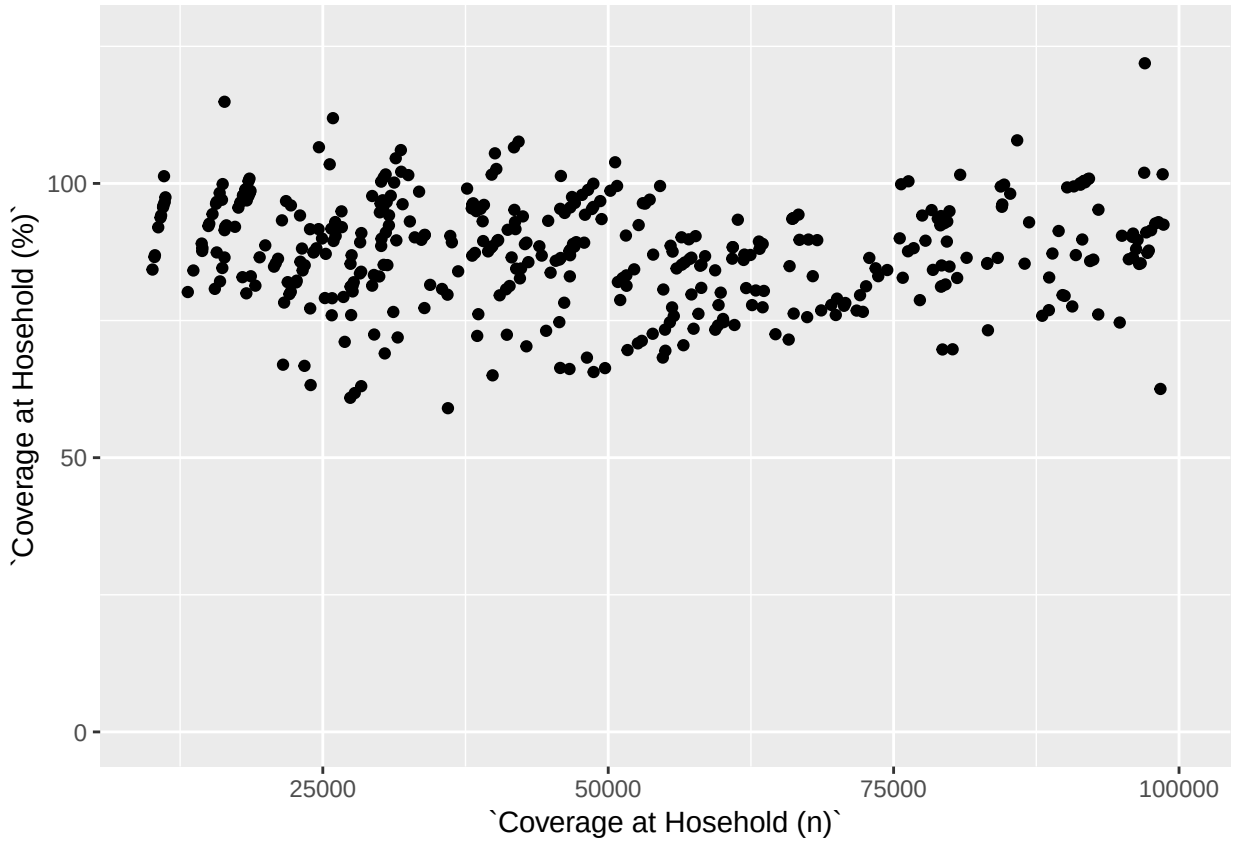
We can assign the plot to a variable in the same way as any other value. This means we can then add to and plot it again later.

```
my_plot <- ggplot(data = Admin, mapping = aes(x = `Coverage at Hosehold (n)`,
                                              y = `Coverage at Hosehold (%)`))
```

For example, if we wanted to zoom in on a part of the plot.

```
my_plot + geom_point() + xlim(1e+4, 1e+5)
```

```
## Warning: Removed 724 rows containing missing values (geom_point).
```



Time series

Lets use the AFP cases data. First, aggregate the data by year of onset month and district. This gives the total number of cases for each.

```
AFP_agg <-
  AFP %>%
  group_by(YRONSET, MTHONSET, DISTRICT) %>%
  summarise(cases = n())
```

Now create a new column of combined month and year of onset so that we can order the points in time.

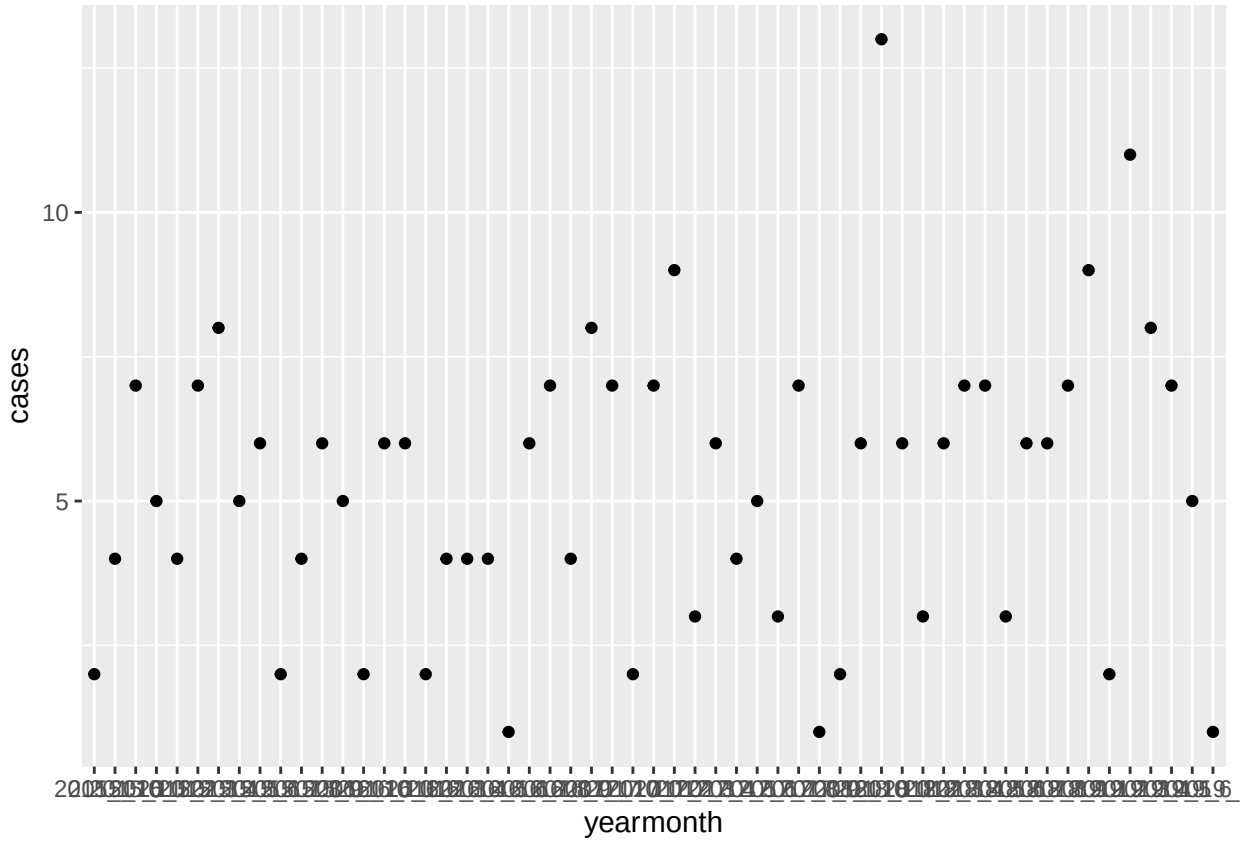
```
AFP_agg$yearmonth <- paste(AFP_agg$YRONSET, AFP_agg$MTHONSET, sep = "_")
```

So that we can clearly see the data, lets look at a single District e.g. Attock.

```
AFP_ATTOCK <- AFP_agg[AFP_agg$DISTRICT == "ATTOCK", ]
```

Now we are ready to plot. Plot number of cases by time as a scatter plot.

```
ggplot(AFP_ATTOCK, aes(x=yearmonth, y=cases)) + geom_point()
```

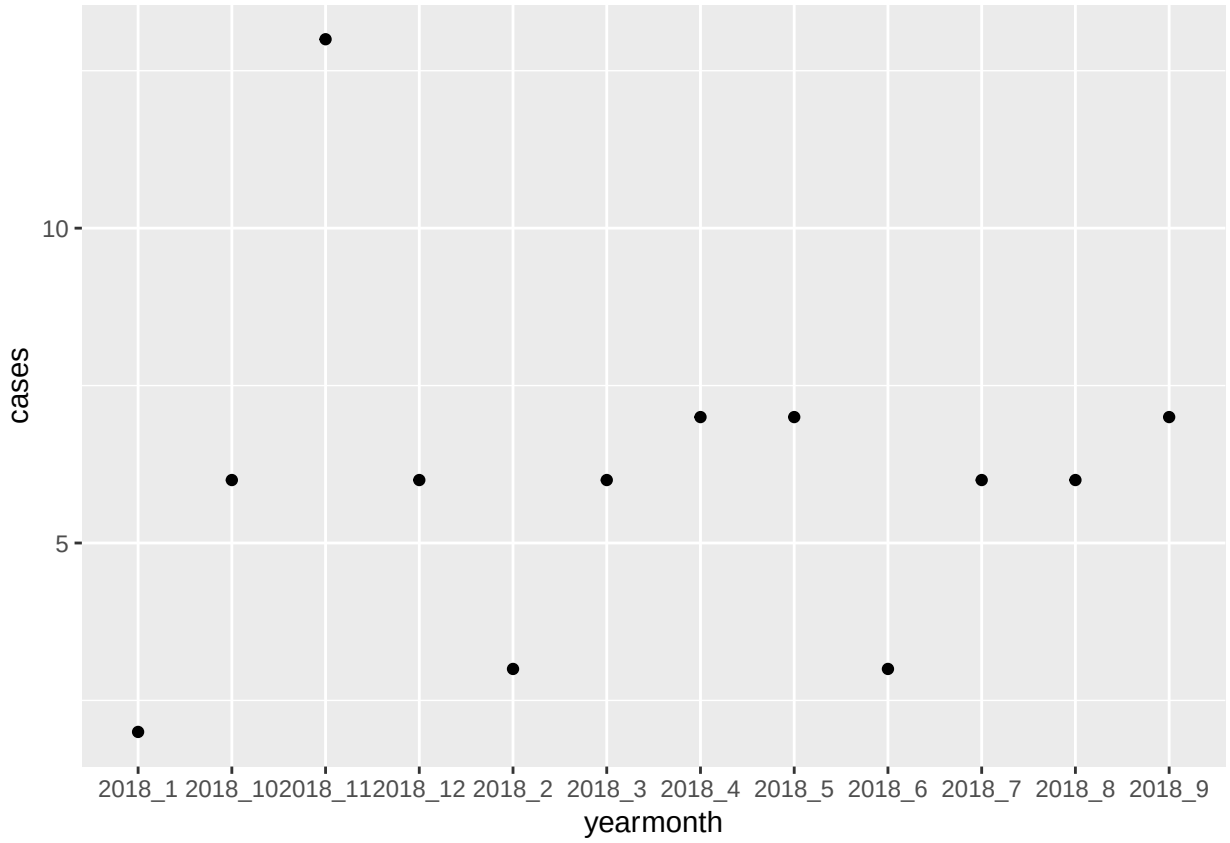


For exposition lets looks at a single year e.g. 2018. Subset the data set to just this year.

```
AFP_ATTOCK2018 <- AFP_ATTOCK[AFP_ATTOCK$YRONSET == 2018, ]
```

Now plot the scatter plot again.

```
ggplot(AFP_ATTOCK2018, aes(x=yearmonth, y=cases)) + geom_point()
```



Notice that the months are not in the correct sequence. This is another case of ordering the levels of factors. Lets create a vector of what order we want them to have.

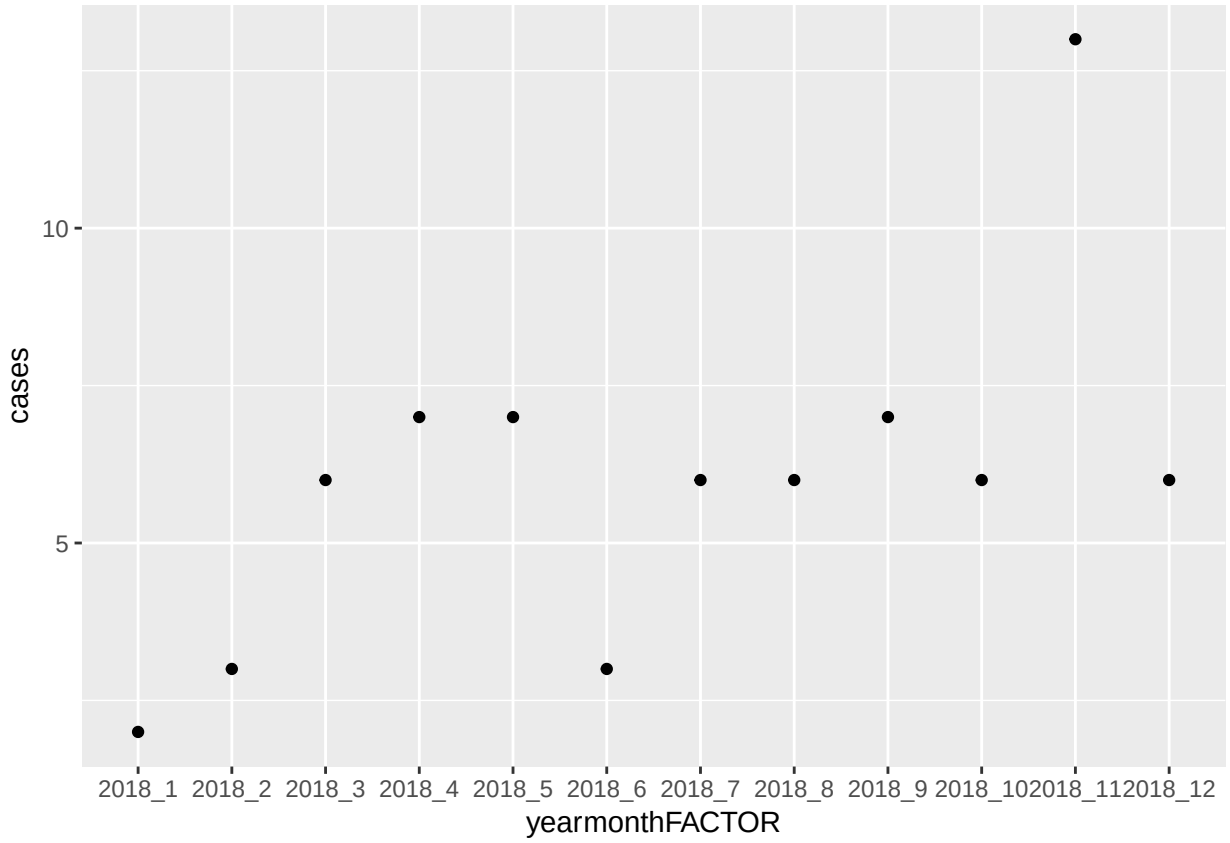
```
LEVELS <-
  c(paste("2015", 1:12, sep = "_"),
    paste("2016", 1:12, sep = "_"),
    paste("2017", 1:12, sep = "_"),
    paste("2018", 1:12, sep = "_"),
    paste("2019", 1:12, sep = "_"))
```

And then make this the order in the data.

```
AFP_ATTOCK2018$yearmonthFACTOR <- factor(AFP_ATTOCK2018$yearmonth, levels = LEVELS)
```

Finally, plot again to check.

```
ggplot(AFP_ATTOCK2018, aes(x=yearmonthFACTOR, y=cases)) + geom_point()
```

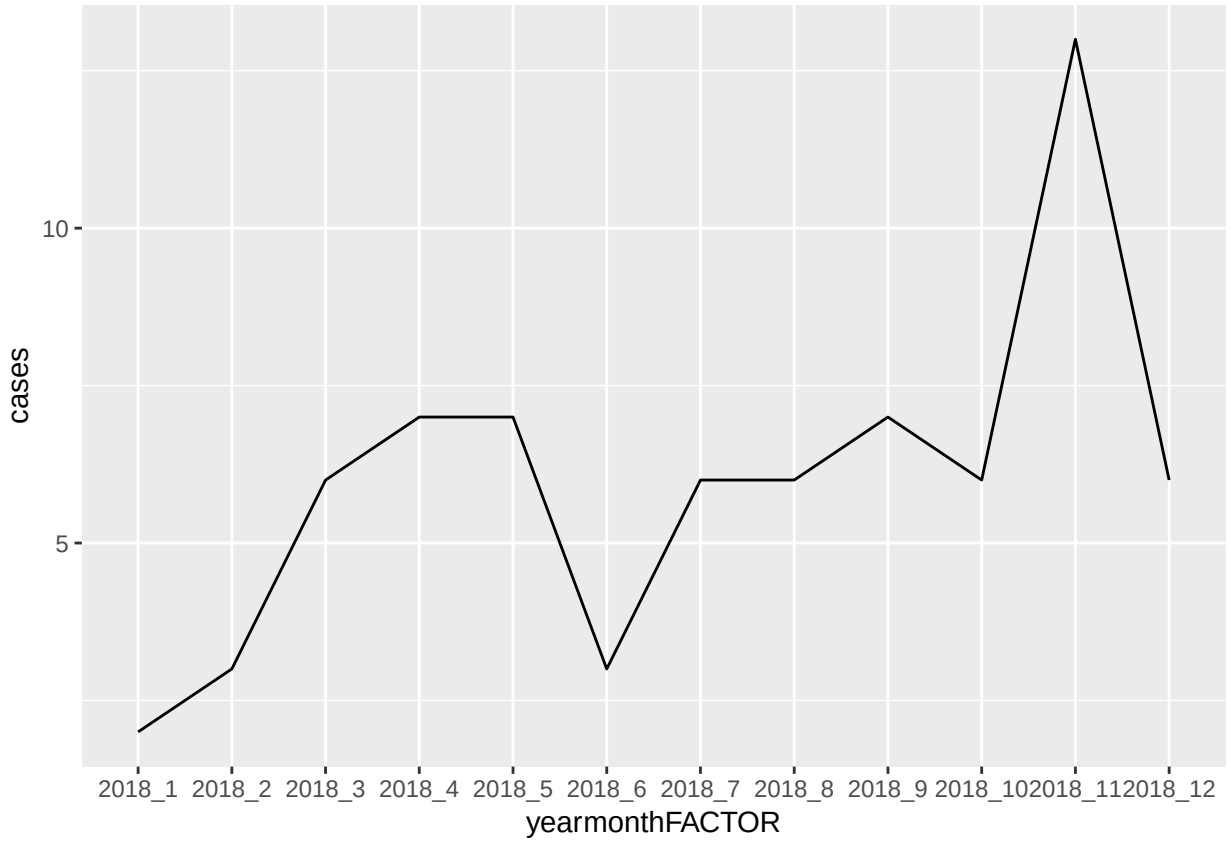


Adding a line

Use the `geom_line()` function.

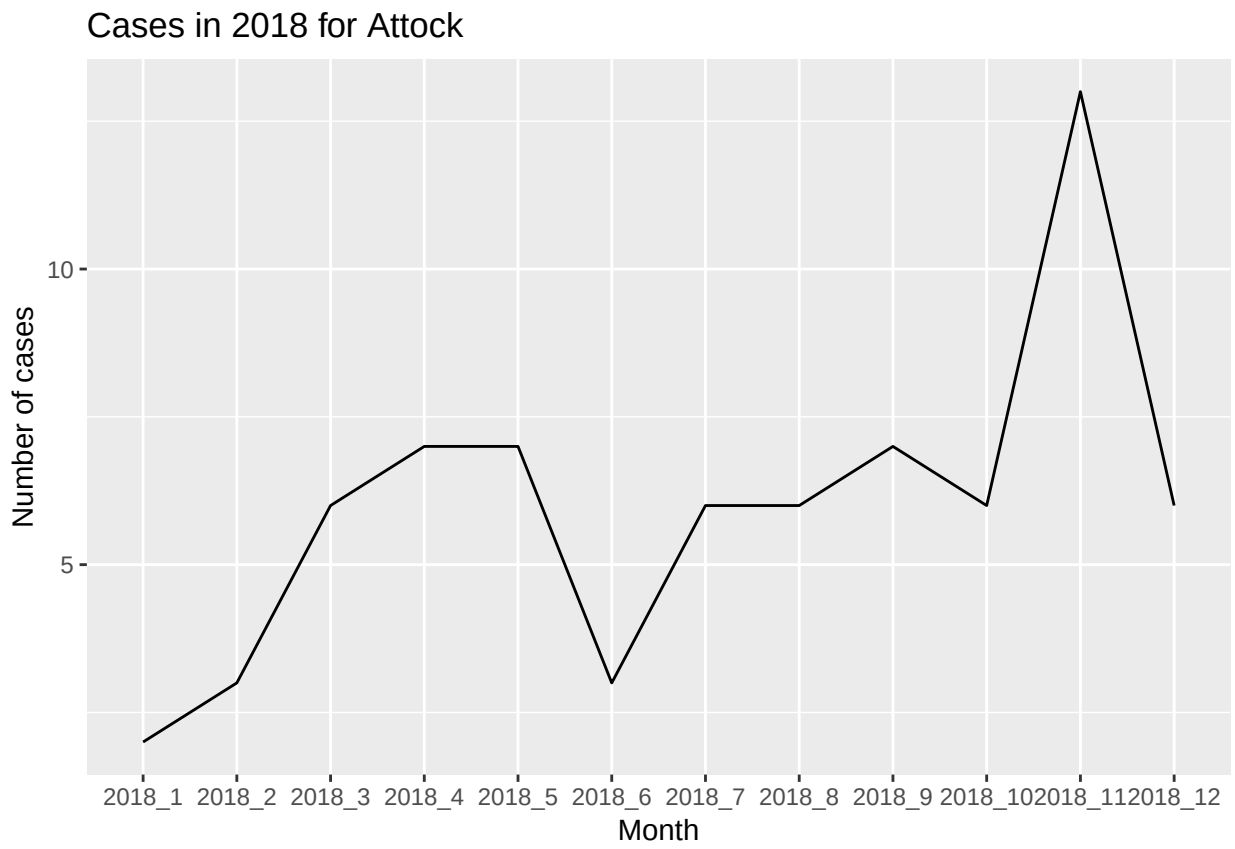
```
my_plot <-
  ggplot(AFP_ATTOCK2018, aes(x=yearmonthFACTOR, y=cases, group=1)) + geom_line()

my_plot
```



Adding titles and labelling

```
my_plot + ggtitle("Cases in 2018 for Attock") +
  xlab("Month") + ylab("Number of cases")
```



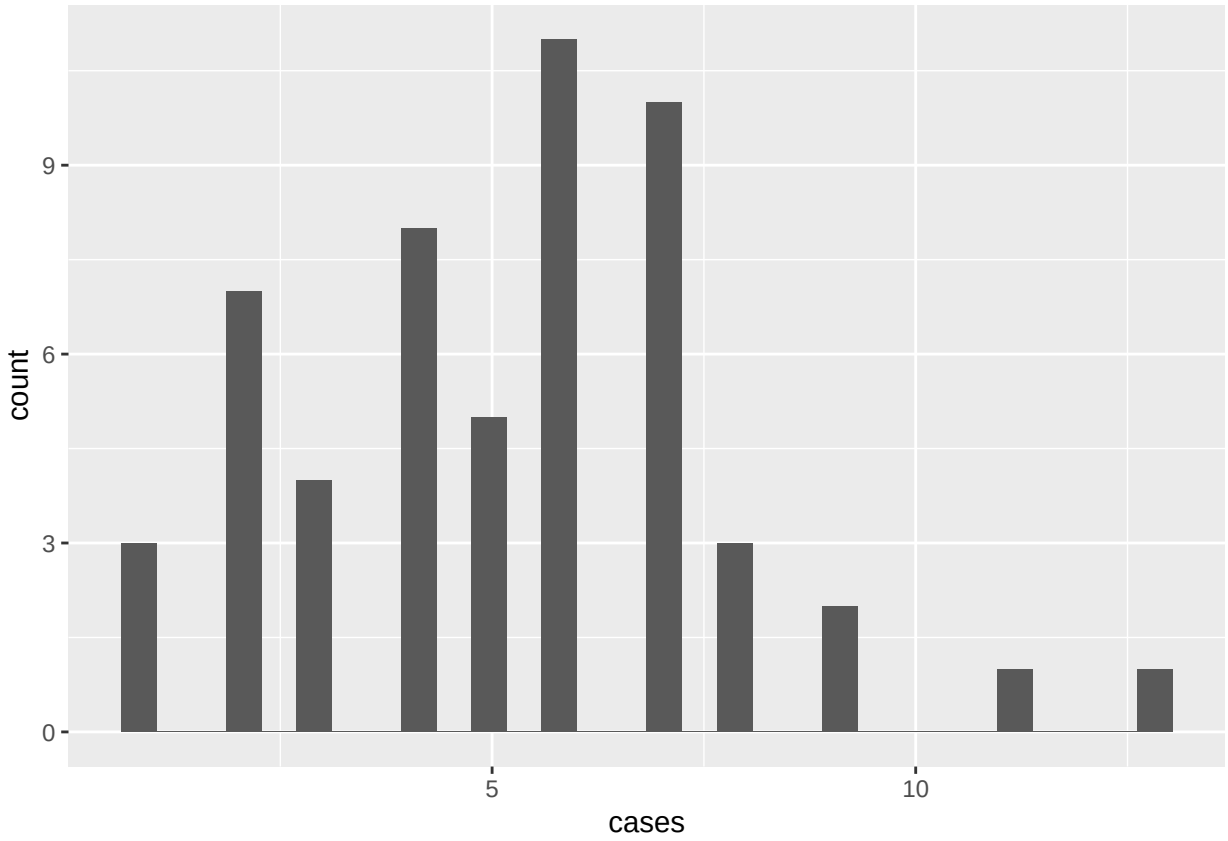
Changing plot type

Creating a histogram uses the `geom_histogram`.

```
plot_hist <-
  ggplot(AFP_ATTOCK, aes(x=cases)) + geom_histogram()

plot_hist
```

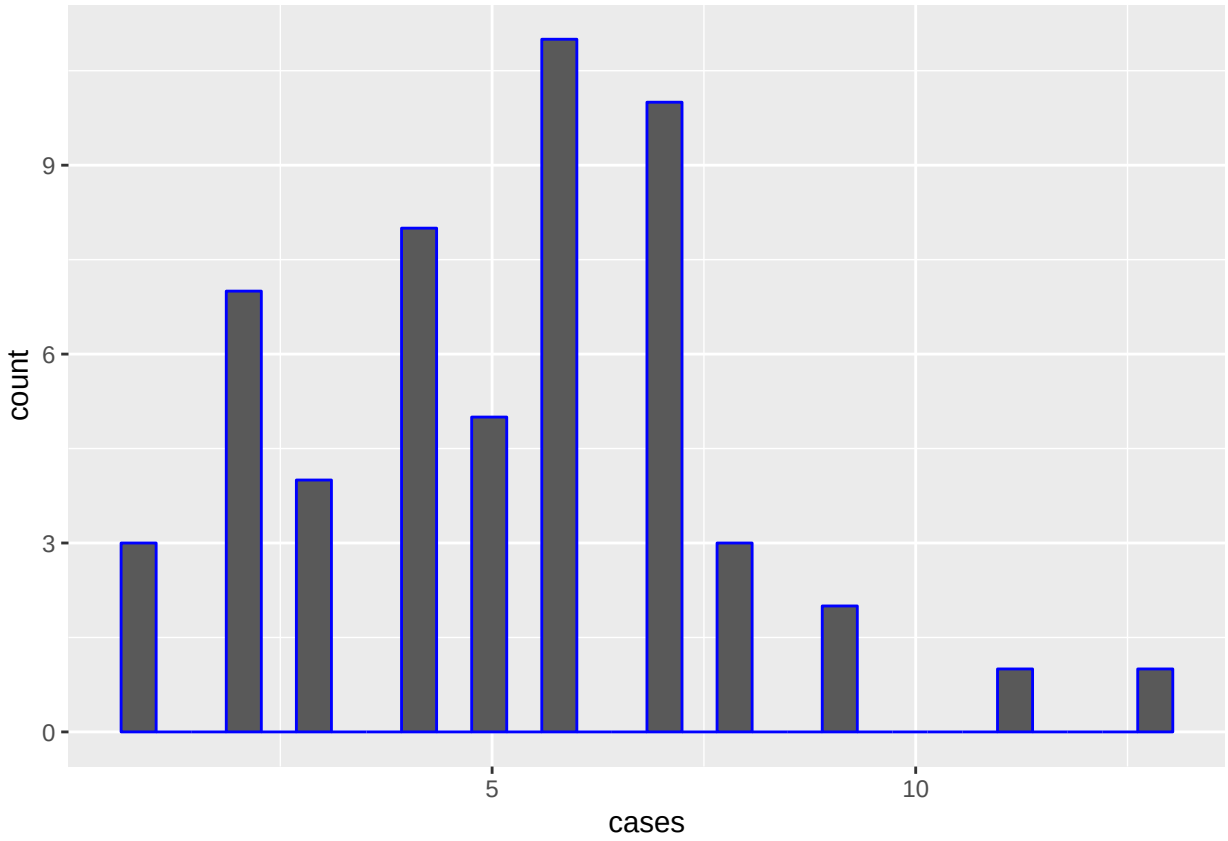
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



We can change the colours of the outlines, bar fill in the same way as for base R.

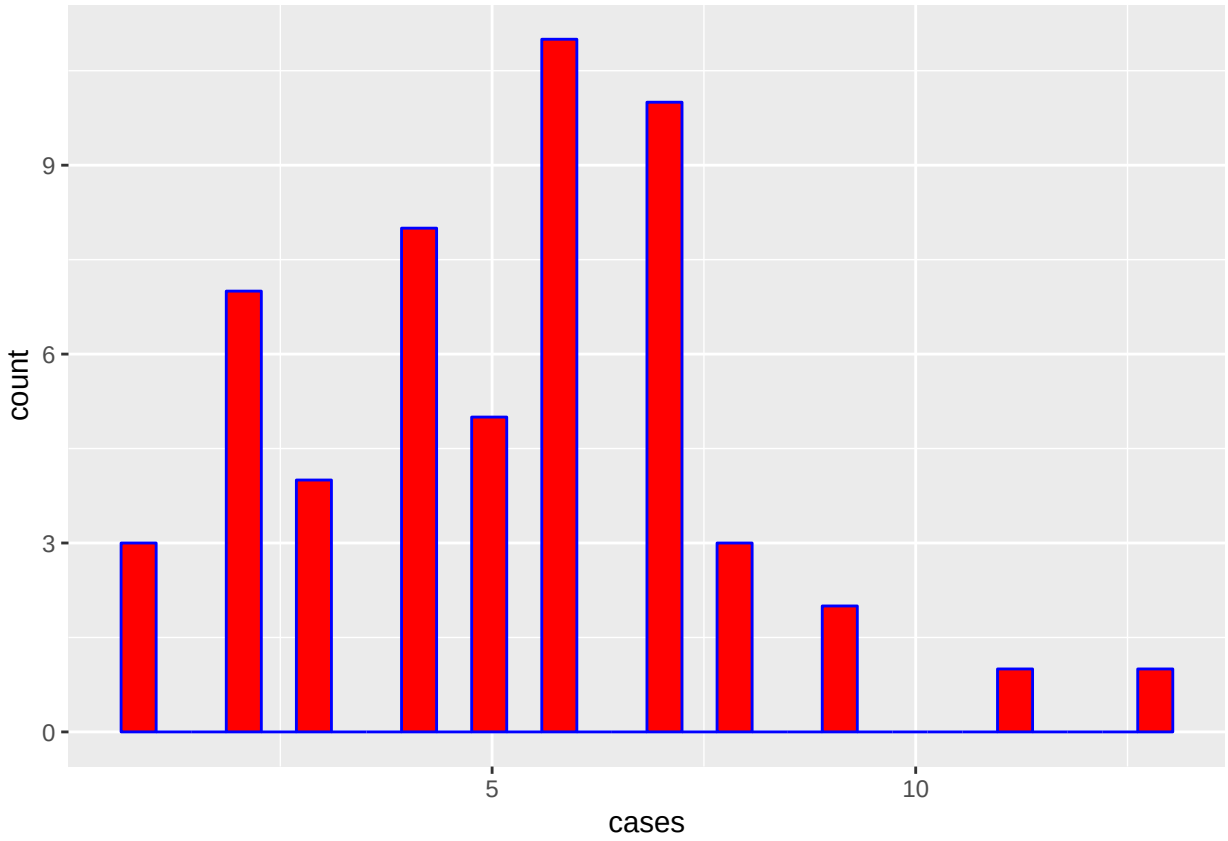
```
ggplot(AFP_ATTOCK, aes(x=cases)) + geom_histogram(colour = "blue")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



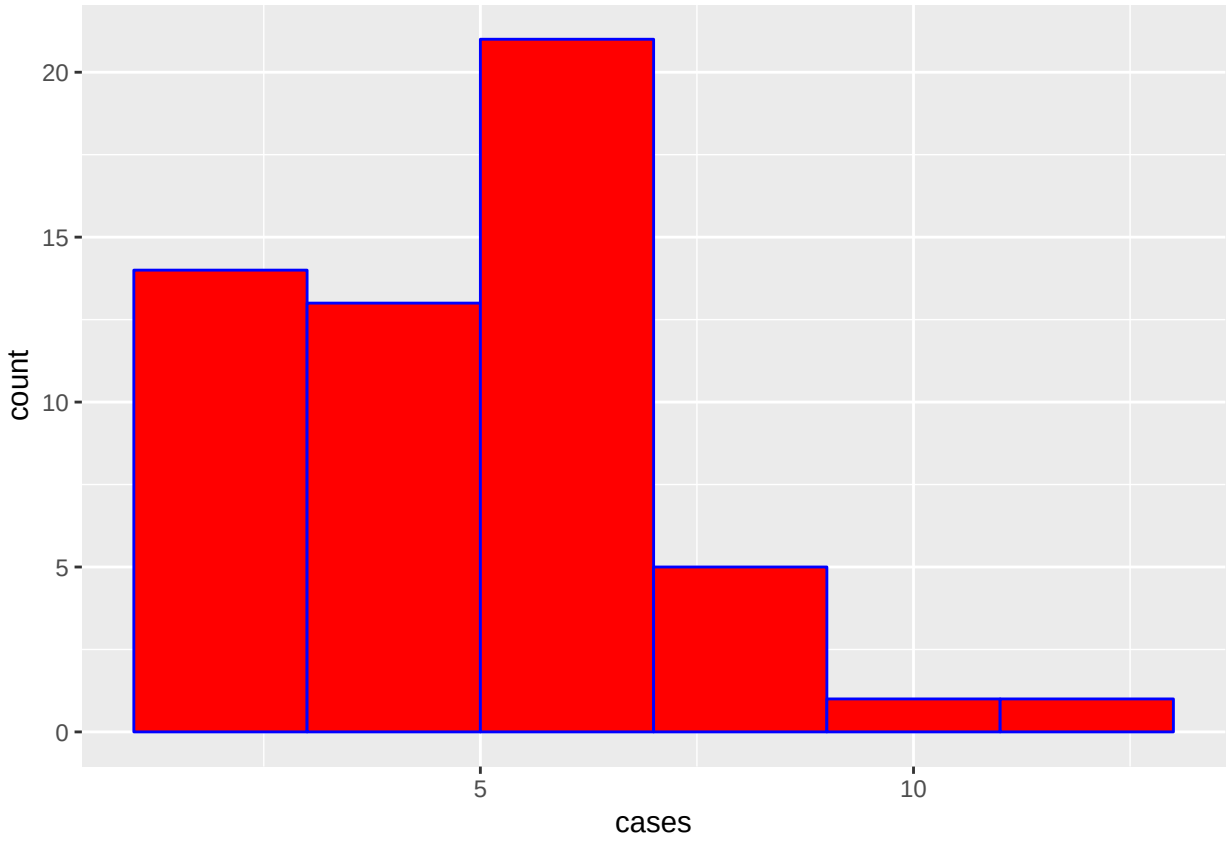

```
ggplot(AFP_ATTOCK, aes(x=cases)) + geom_histogram(colour = "blue", fill = "red")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



We can also change the bin sizes that the histogram uses.

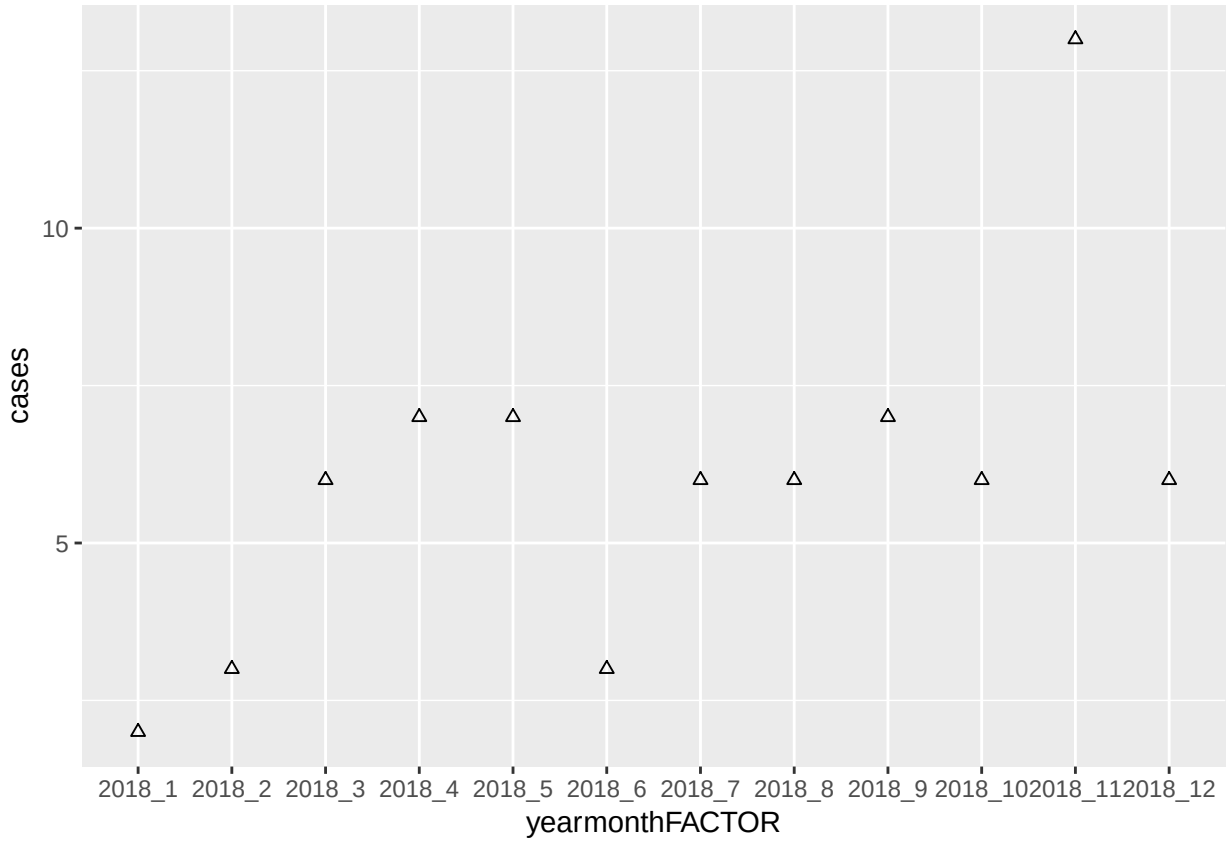
```
ggplot(AFP_ATTOCK, aes(x=cases)) + geom_histogram(colour = "blue", fill = "red", binwidth = 2)
```



Change point style

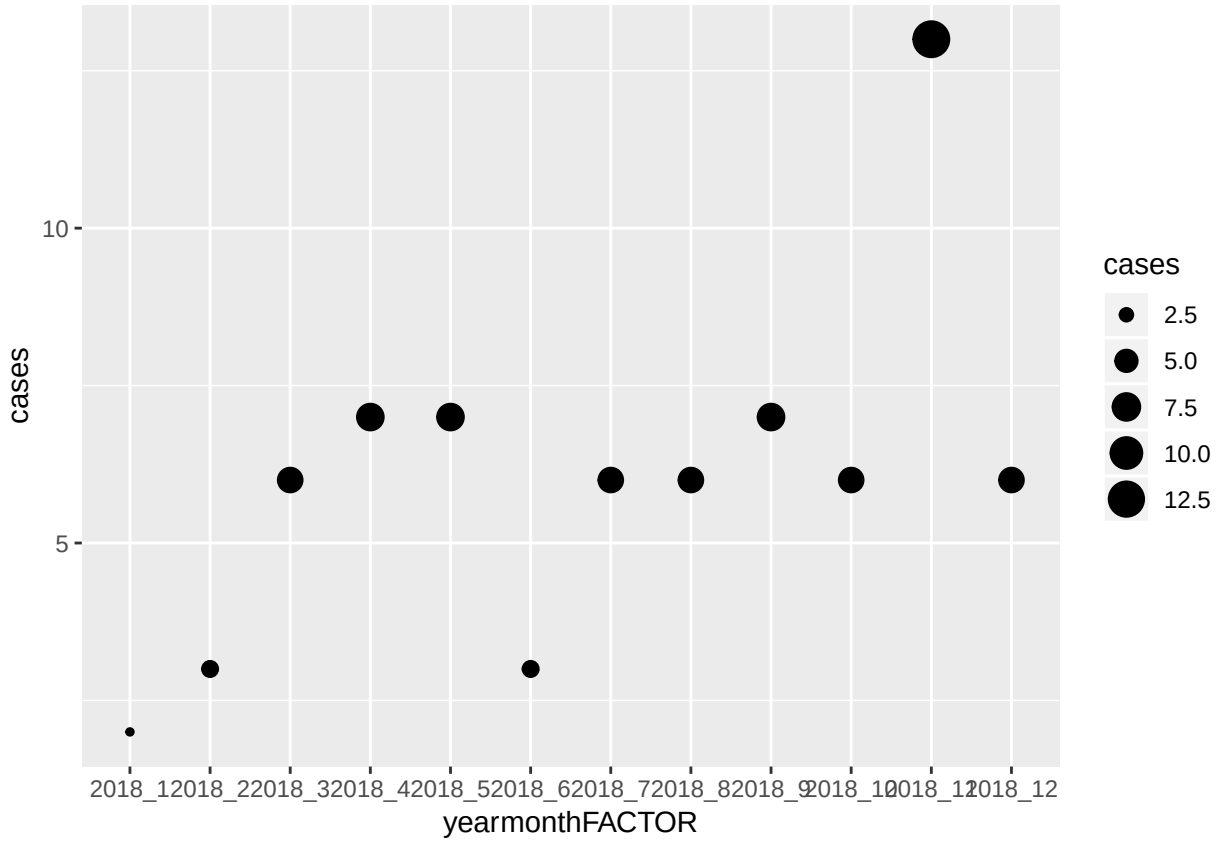
Change the point to triangles.

```
ggplot(AFP_ATTOCK2018, aes(x=yearmonthFACTOR, y=cases, group=1)) + geom_point(shape = 2)
```



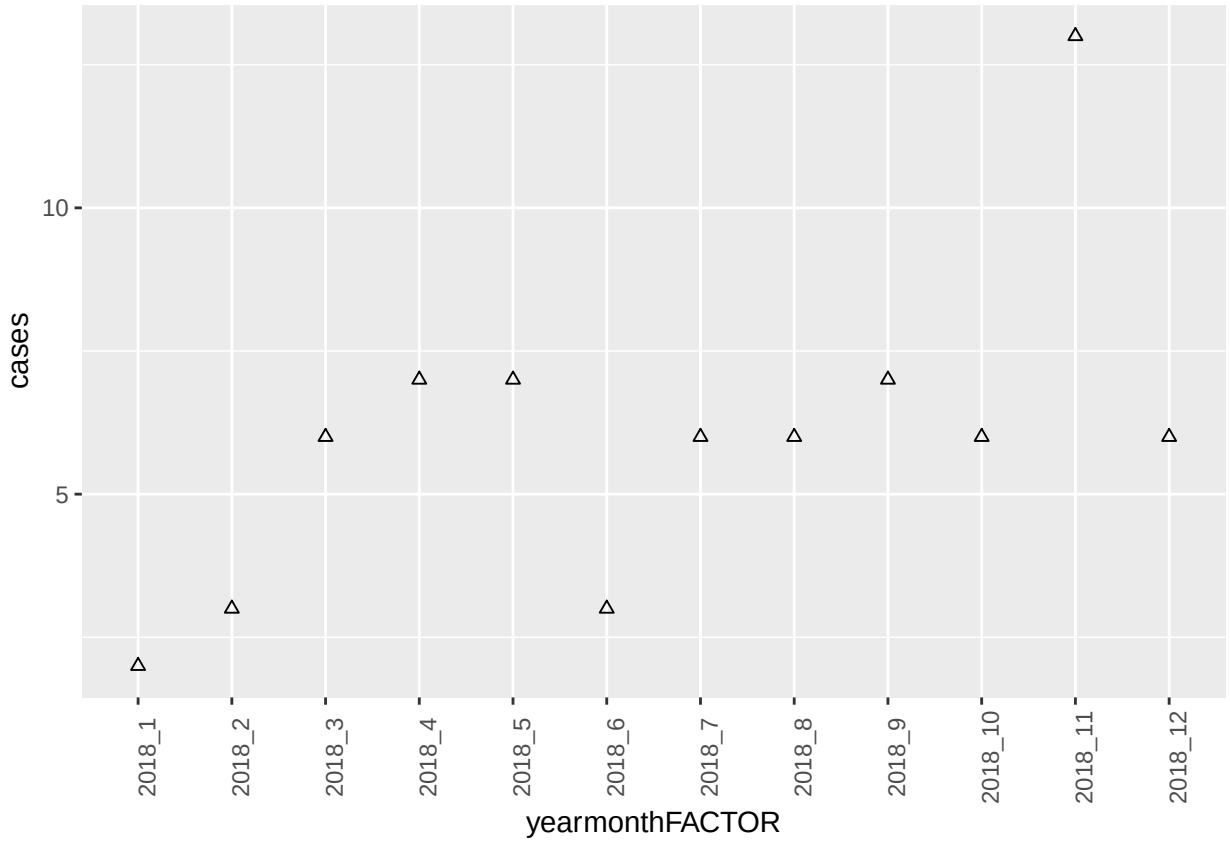
Chnge the sizes of the points.

```
ggplot(AFP_ATTOCK2018, aes(x=yearmonthFACTOR, y=cases, group=1)) + geom_point(aes(size = cases))
```



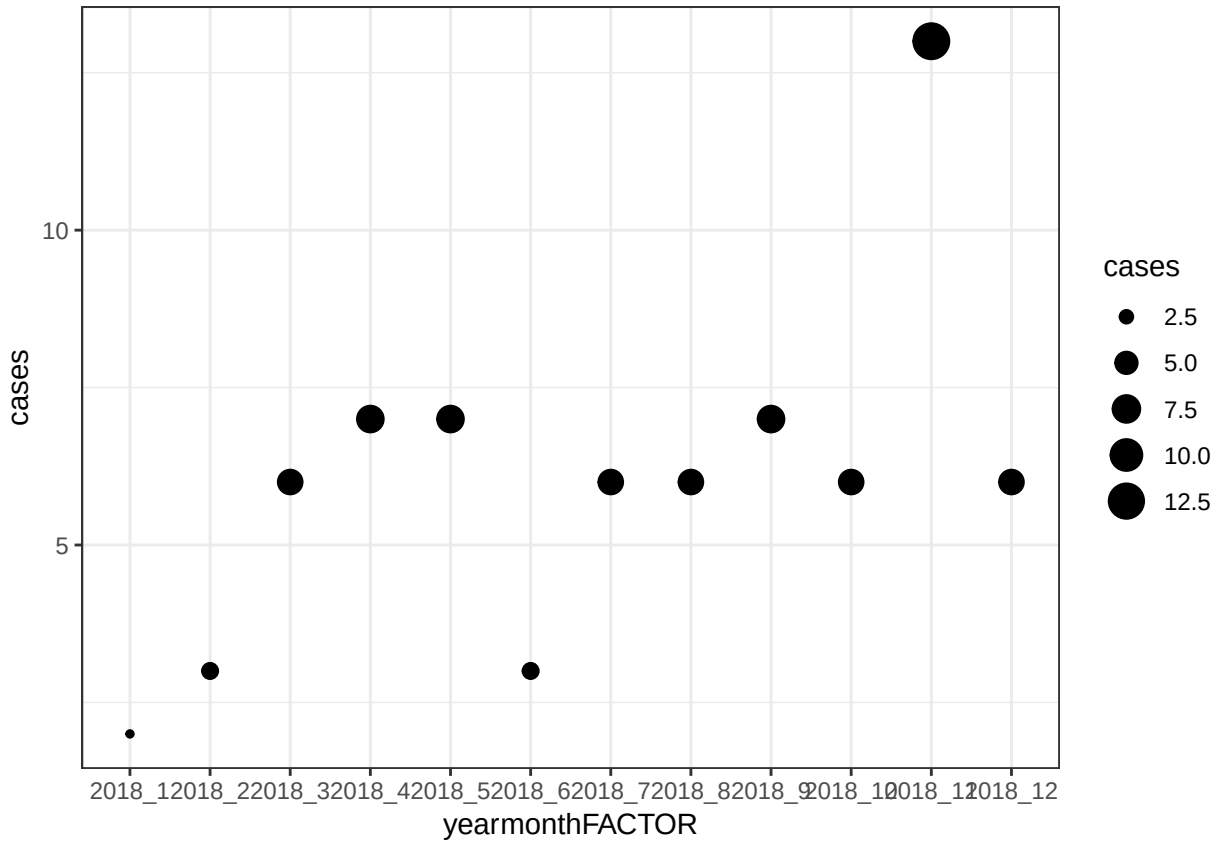
Rotate axis labels

```
ggplot(AFP_ATTOCK2018, aes(x=yearmonthFACTOR, y=cases, group=1)) + geom_point(shape = 2) +
  theme(axis.text.x=element_text(angle=90))
```



There are lots of themes to choose from. The most common have their own function e.g. `theme_bw()`.

```
ggplot(AFP_ATTOCK2018, aes(x=yearmonthFACTOR, y=cases, group=1)) +
  geom_point(aes(size = cases)) +
  theme_bw()
```



Saving plots

Use the name of the plot you've assigned to save it.

```
ggsave("my_plot.png", my_plot)
```

```
## Saving 6.5 x 4.5 in image
```