Graphics with ggplot2 (solution)

Data Science Lab, University of Copenhagen

20 May, 2022

Loading the core tidyverse packages and the 'readxl' package for data import from .xlsx.

```
library(tidyverse)
library(readxl)
```

Importing the climate data from **climate.xlsx**[1]. (Change the path to the Excel file below so that it matches the path to the file saved on your own computer, or use *Import Dataset* in RStudio to obtain the relevant code.)

```
climate <- read_excel("climate.xlsx")
climate
```

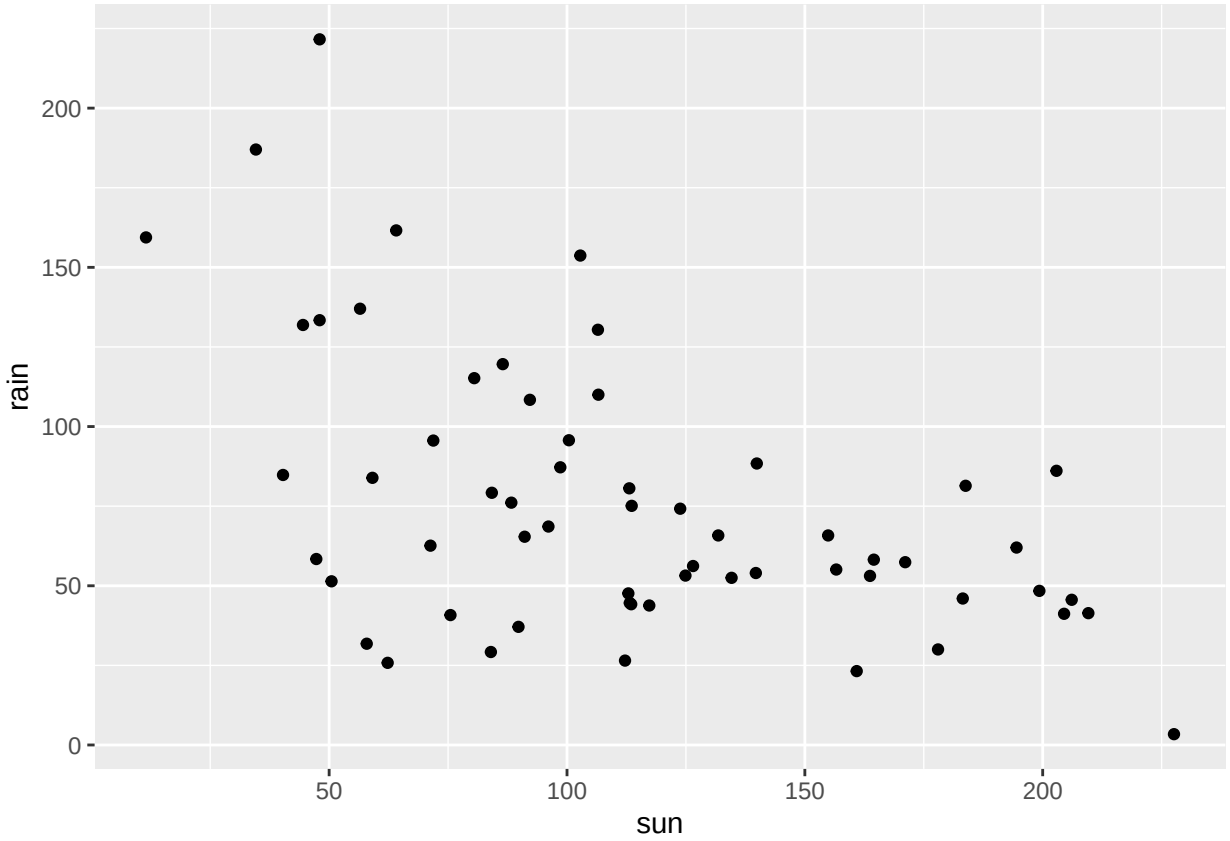
```
## # A tibble: 60 x 7
##    station  year month    af  rain   sun device
##    <chr>   <dbl> <dbl> <dbl> <dbl> <dbl> <chr>
##  1 armagh   2016     1     5 132.   44.5 Campbell Stokes
##  2 armagh   2016     2    10  62.6  71.3 Campbell Stokes
##  3 armagh   2016     3     4  43.8 117.  Campbell Stokes
##  4 armagh   2016     4     5  54   140.  Campbell Stokes
##  5 armagh   2016     5     0  41.4 210.  Campbell Stokes
##  6 armagh   2016     6     0  75.1 114.  Campbell Stokes
##  7 armagh   2016     7     0  80.6 113.  Campbell Stokes
##  8 armagh   2016     8     0  52.5 135.  Campbell Stokes
##  9 armagh   2016     9     0  65.4  91.1 Campbell Stokes
## 10 armagh   2016    10     0  37.1  89.8 Campbell Stokes
## # ... with 50 more rows
```

Scatter plot I

1. Make a scatter plot of **rain** against **sun**. HINT: Try `geom_point`.

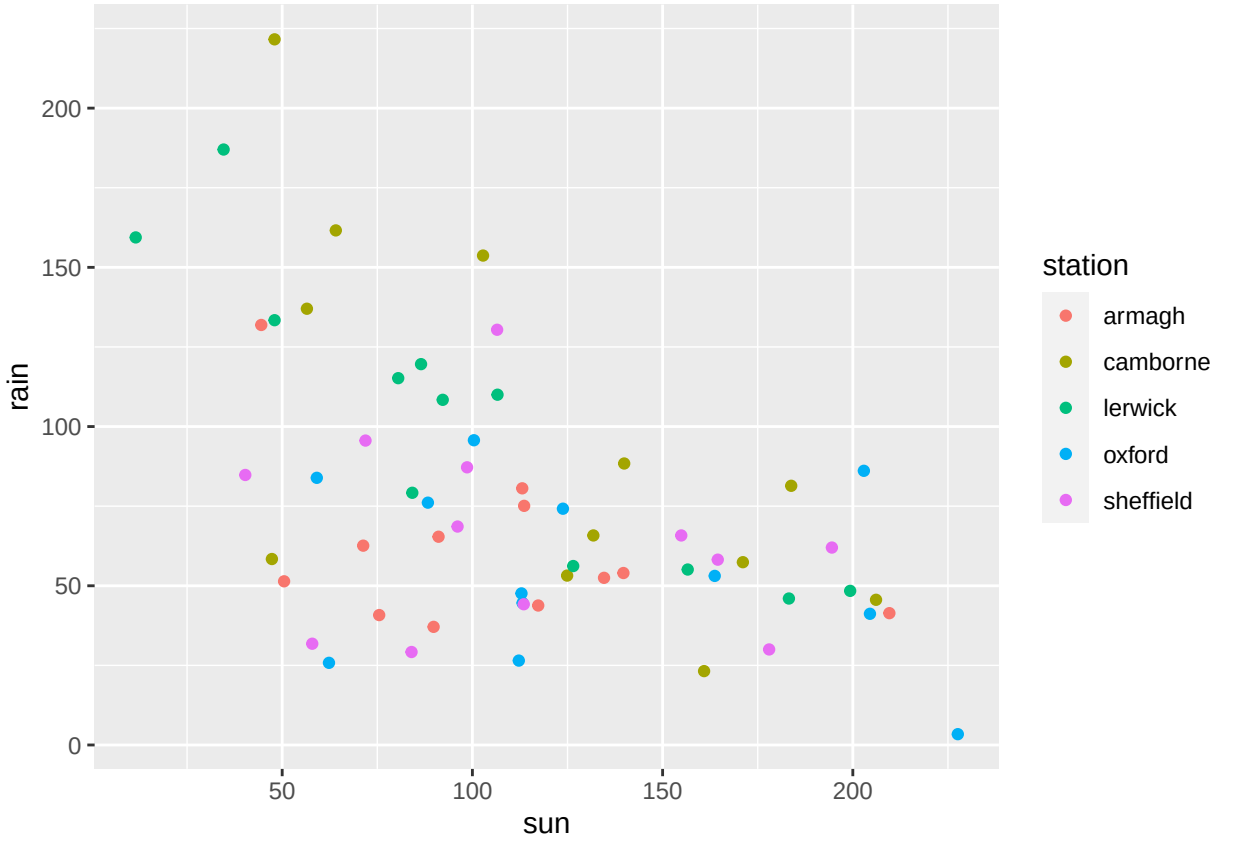
```
ggplot(climate, aes(x = sun, y = rain)) +
  geom_point()
```

[1] Contains public sector information licensed under the Open Government Licence v3.0.



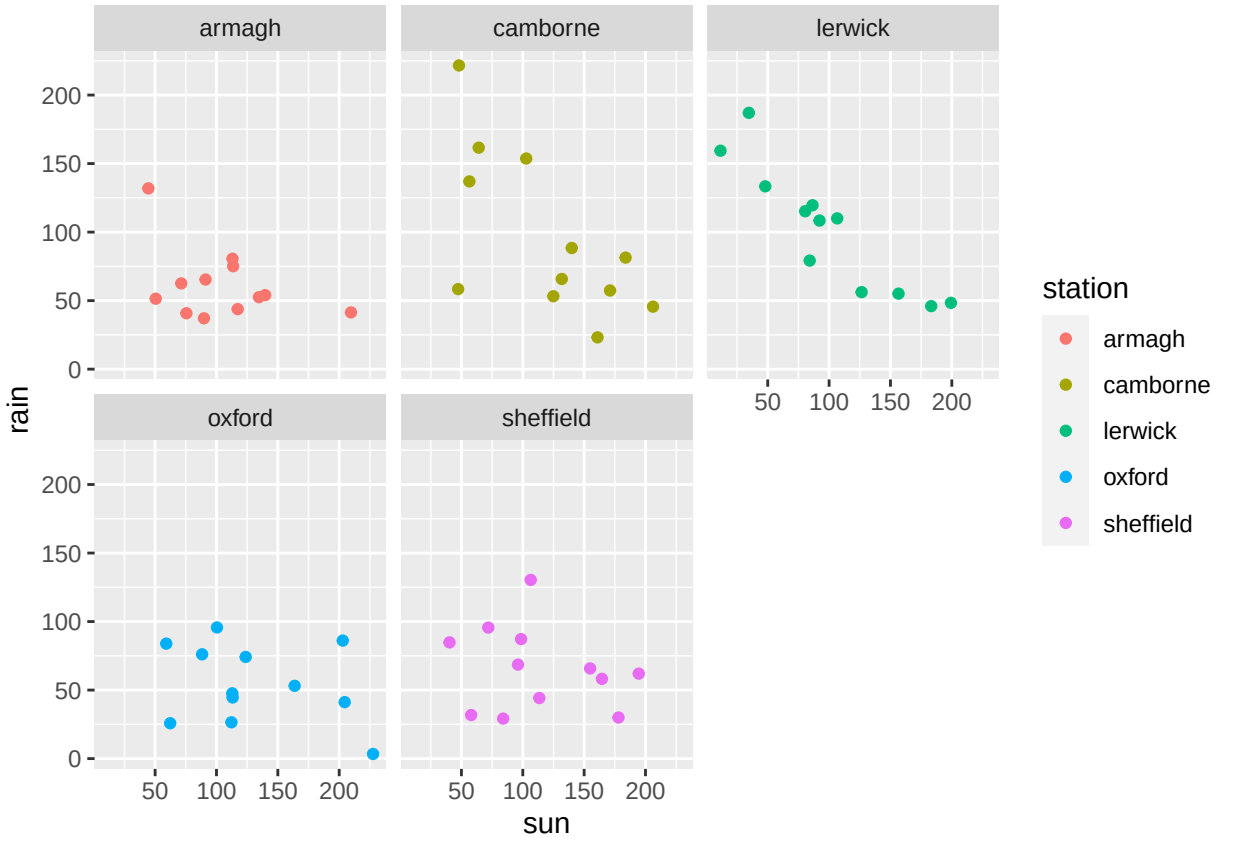
2. Colour the points in the scatter plot according to weather station. Save the plot in an object, for example `p`.

```
p <- ggplot(climate, aes(x = sun, y = rain, colour = station)) +
  geom_point()
p
```



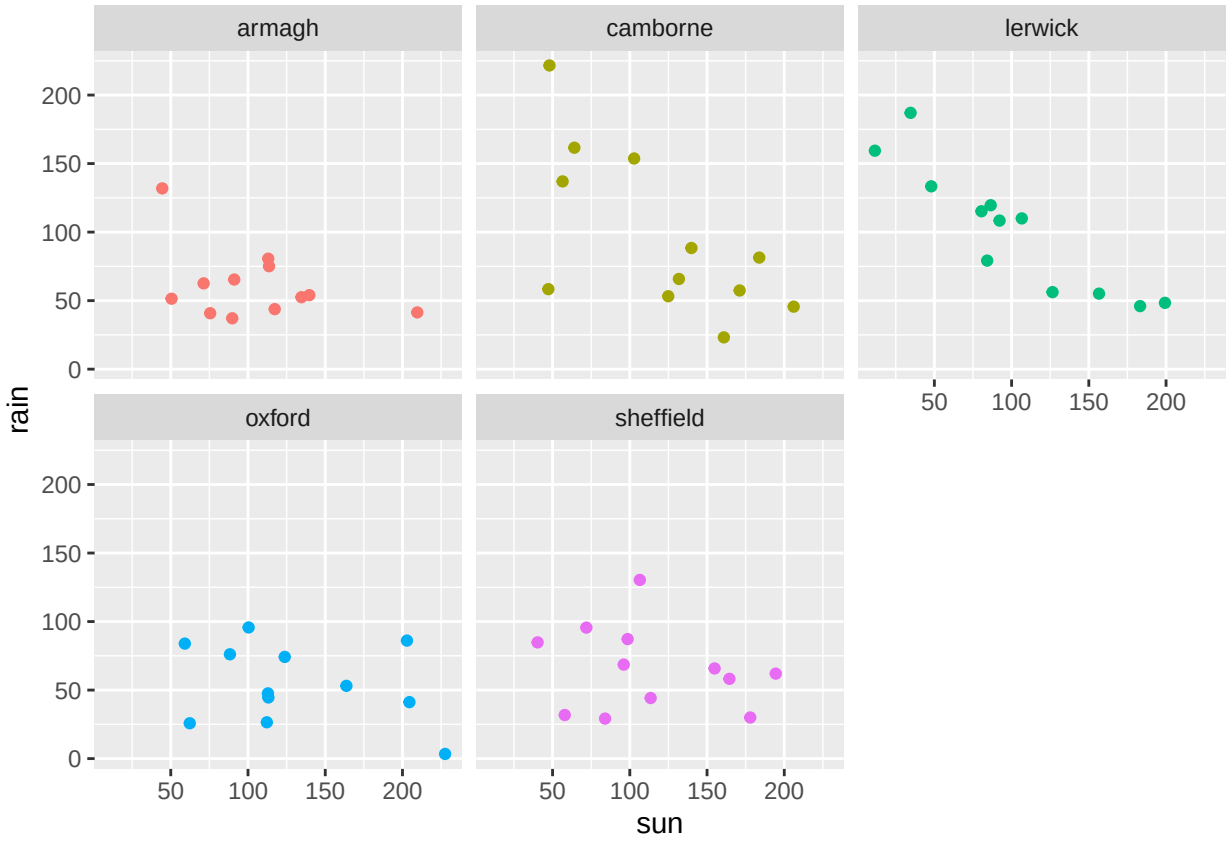
3. Add the command `facet_wrap(~station)` to the saved plot object from above, and update the plot. New instructions are added with `+`, see the ggplot lecture. What happens?

```
p + facet_wrap(~station)
```



4. Is it necessary to have a legend in the faceted plot? You can remove the legend by adding `theme(legend.position = "none")` to your code.

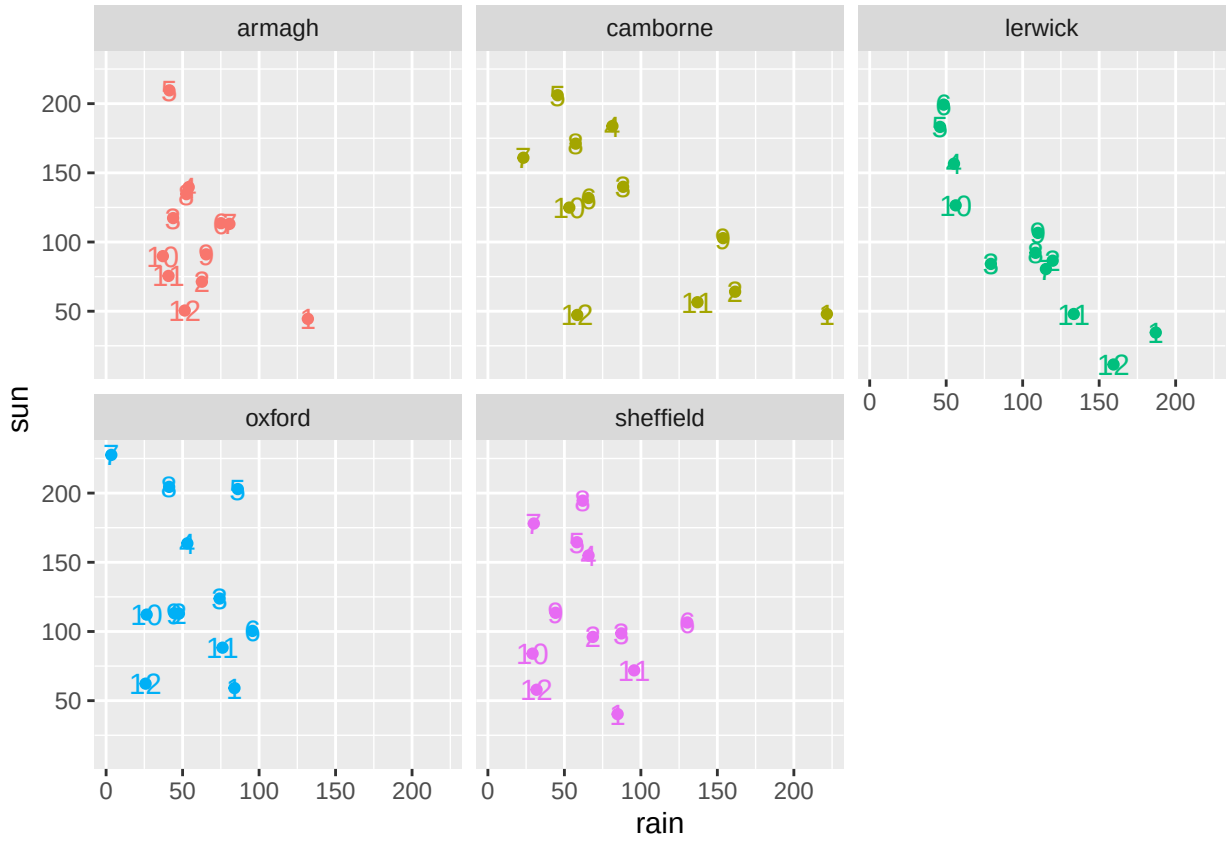
```
p <- ggplot(climate, aes(x = sun, y = rain, colour = station)) +
  geom_point() +
  facet_wrap(~station) +
  theme(legend.position = "none")
p
```



Scatter plot II: labels

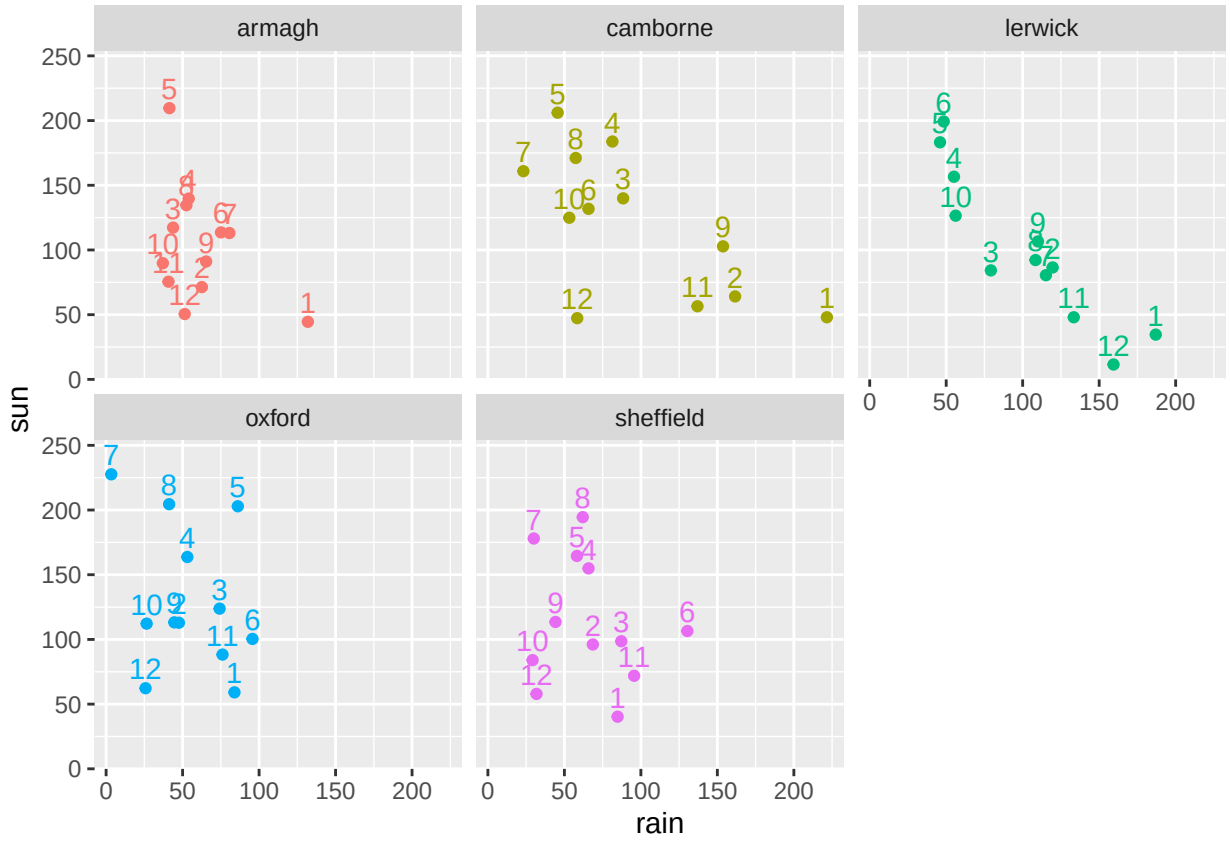
5. Now, let's label each point with the month it corresponds to. This is done by using the geom `geom_text`.

```
ggplot(climate, aes(x=rain, y = sun, color = station)) +
  geom_point() + facet_wrap(~station) + theme(legend.position = "none") +
  geom_text(aes(label = month))
```



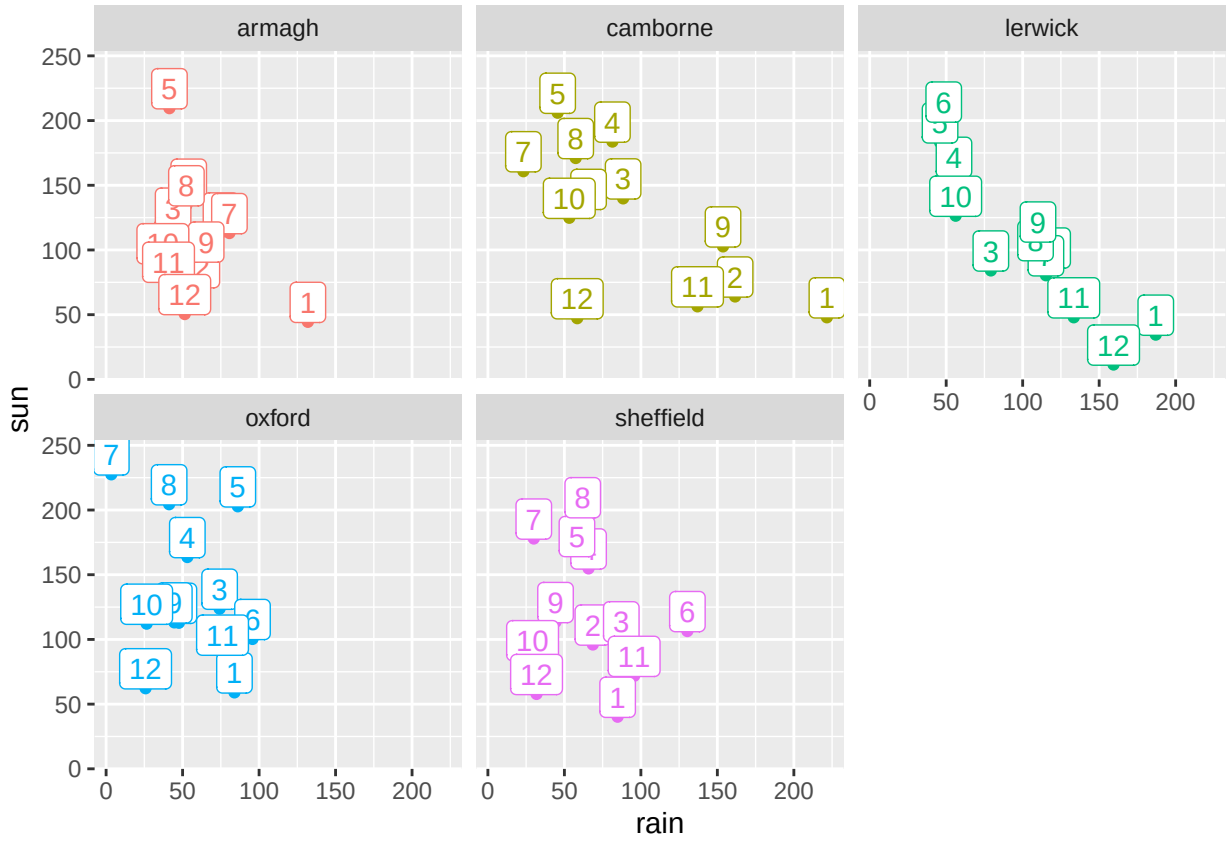
6. You might find that your labels are on top of your points and not very readable. Let's put them a bit higher by adding `nudge_y = 15` to the geom_text.

```
ggplot(climate, aes(x=rain, y = sun, color = station)) +
  geom_point() + facet_wrap(~station) + theme(legend.position = "none") +
  geom_text(aes(label = month), nudge_y = 15)
```



7. Change `geom_text` to `geom_label`.

```
ggplot(climate, aes(x=rain, y = sun, color = station)) +
  geom_point() + facet_wrap(~station) + theme(legend.position = "none") +
  geom_label(aes(label = month), nudge_y = 15)
```



Graphic files

```
ggsave(file="weather.jpeg")

ggsave(file="weather.png",width=10,height=8,units="cm")
```

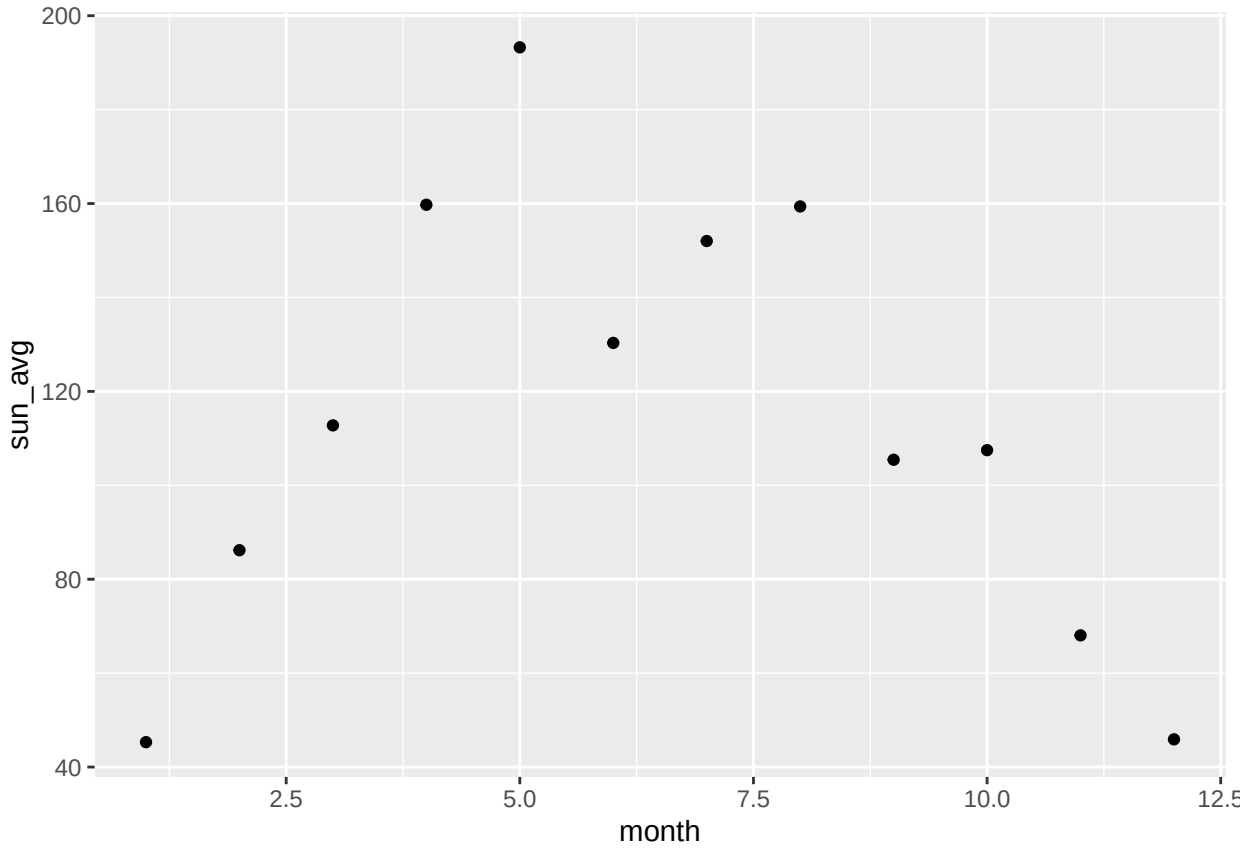
Scatter plot III: error bars

10. Calculate the average and standard deviation for sunshine in each month and save it to a table called `summary_stats`. You will need group_by and summarize.

```
summary_stats <- climate %>%
  group_by(month) %>%
  summarize(sun_avg = mean(sun), sun_sd = sd(sun))
```

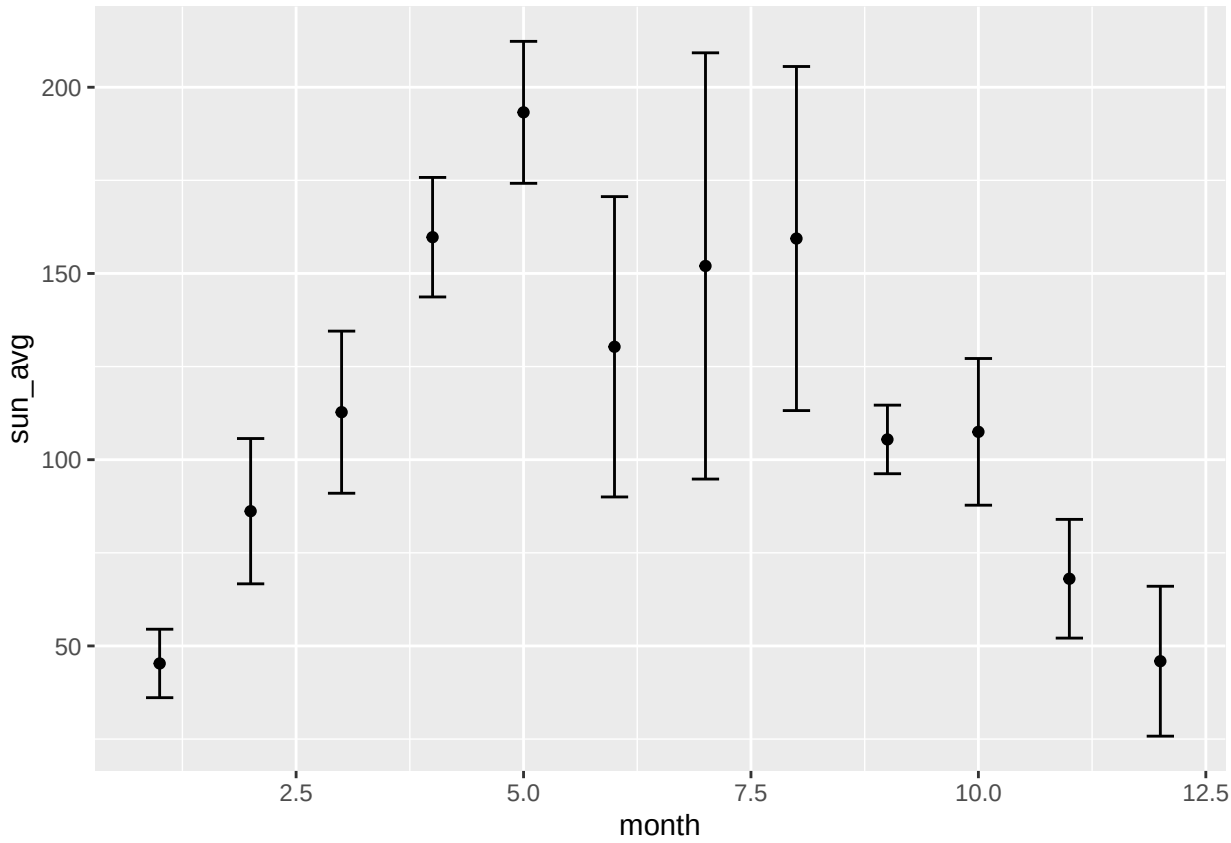
11. Make a scatter plot of the summary_stats with month on the x-axis, and the average number of sunshine hours on the y-axis.

```
p <- ggplot(summary_stats, aes(x = month, y = sun_avg)) +
  geom_point()
p
```



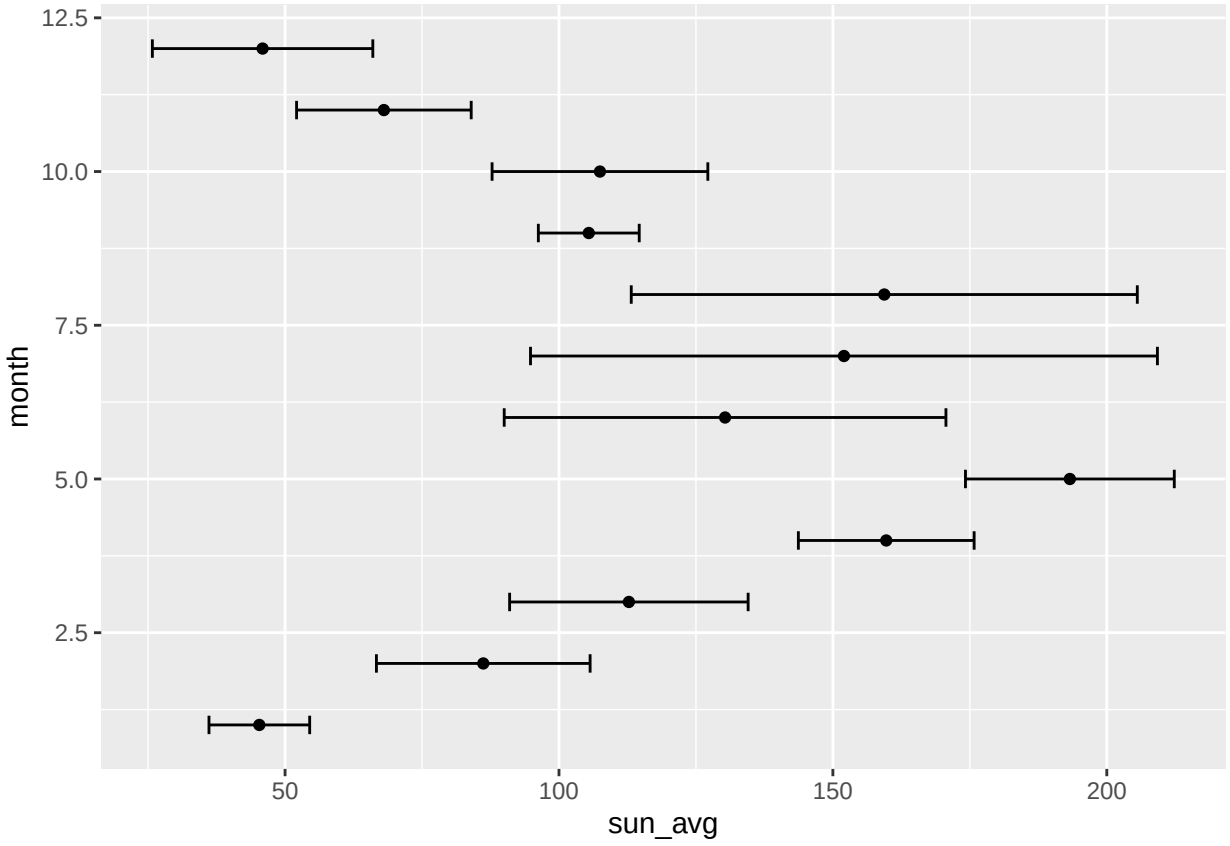
12. Add error bars to the plot, which represent the average number of sunshine hours plus/minus the standard deviation of the observations. The relevant geom is called `geom_errorbar`.

```
p <- ggplot(summary_stats, aes(x = month, y = sun_avg)) +
  geom_point() +
  geom_errorbar(aes(ymin = sun_avg - sun_sd, ymax = sun_avg + sun_sd), width = 0.3)
p
```



13. Could you have made a plot with horizontal error bars instead? How?

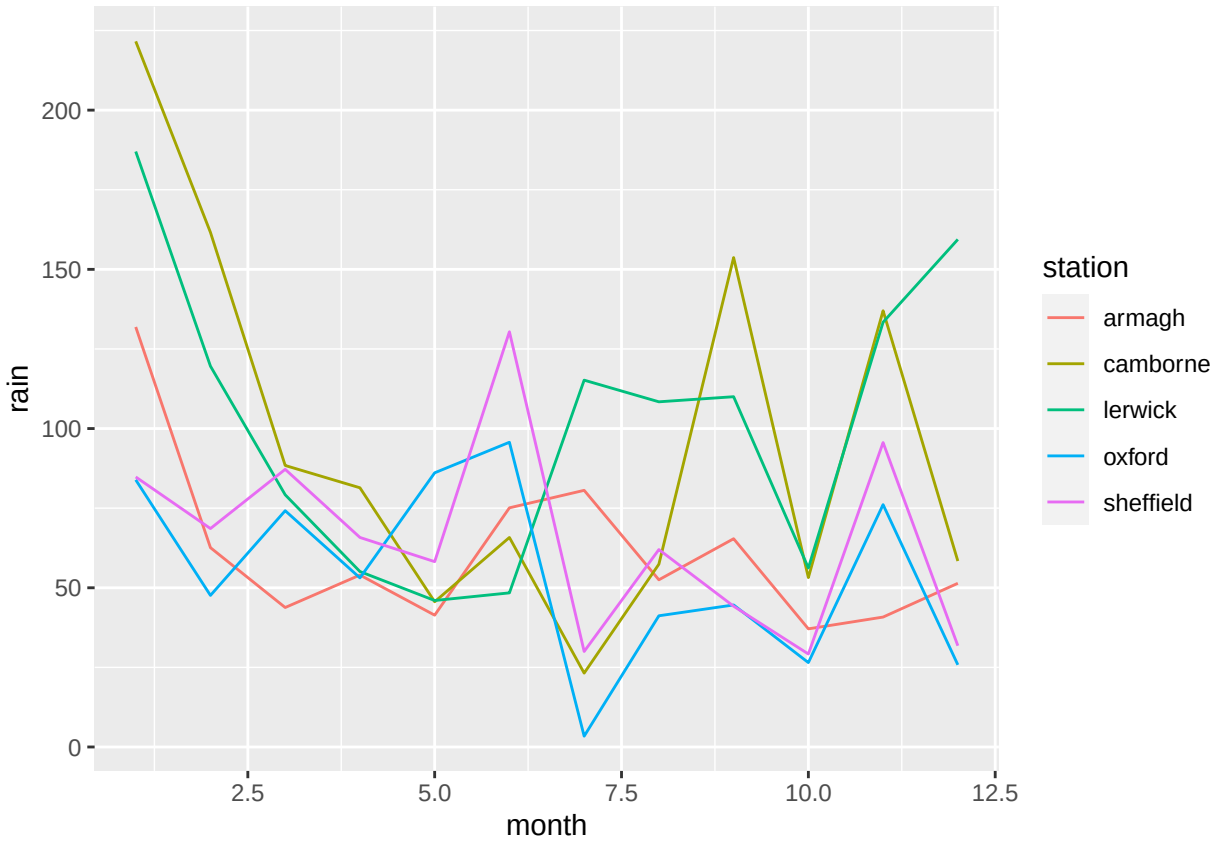
```
p + coord_flip()
```



Line plot

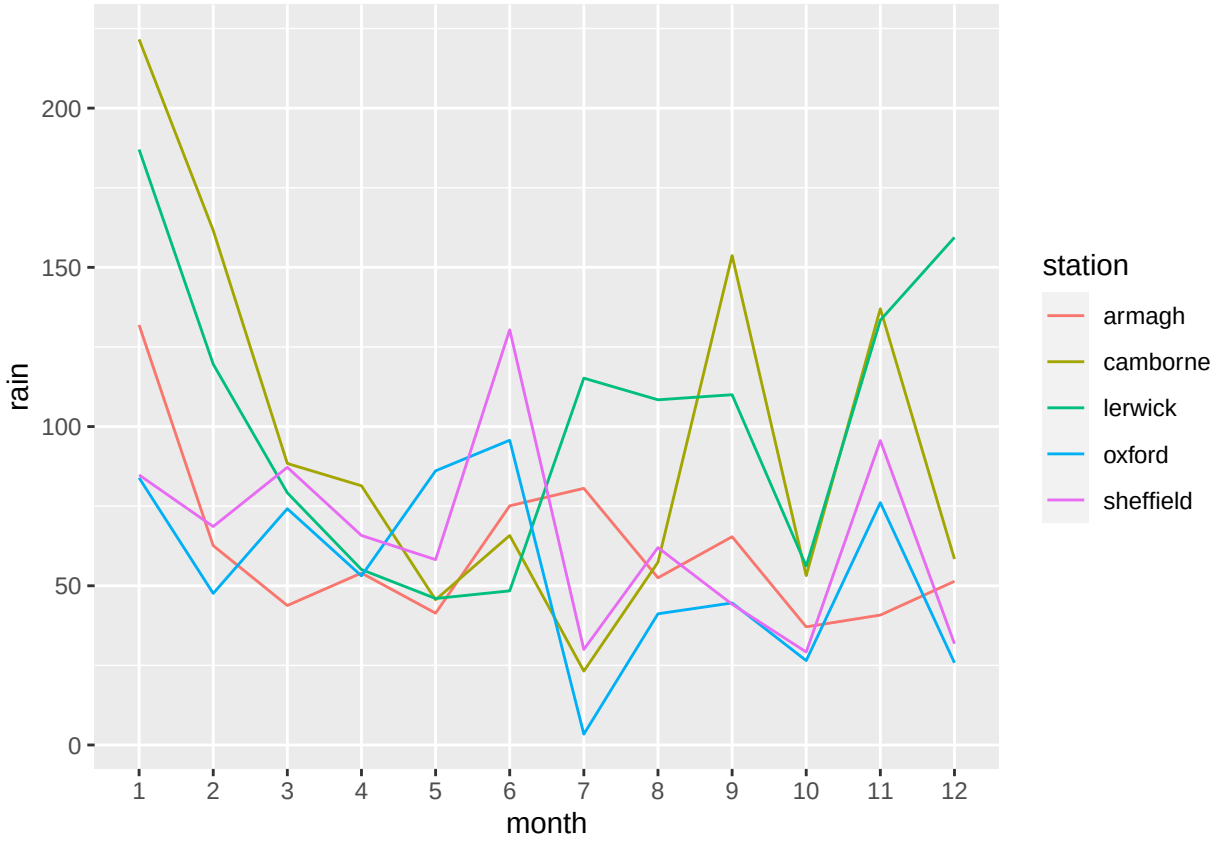
14. Make a line plot of the rainfall observations over time, such that observations from the same station are connected in one line. If you have trouble with this, have a look at the ggplot lecture. Put month on the x-axis. Colour the lines according to weather station as well.

```
ggplot(climate, aes(x = month, y = rain, group = station, colour = station)) +
  geom_line()
```



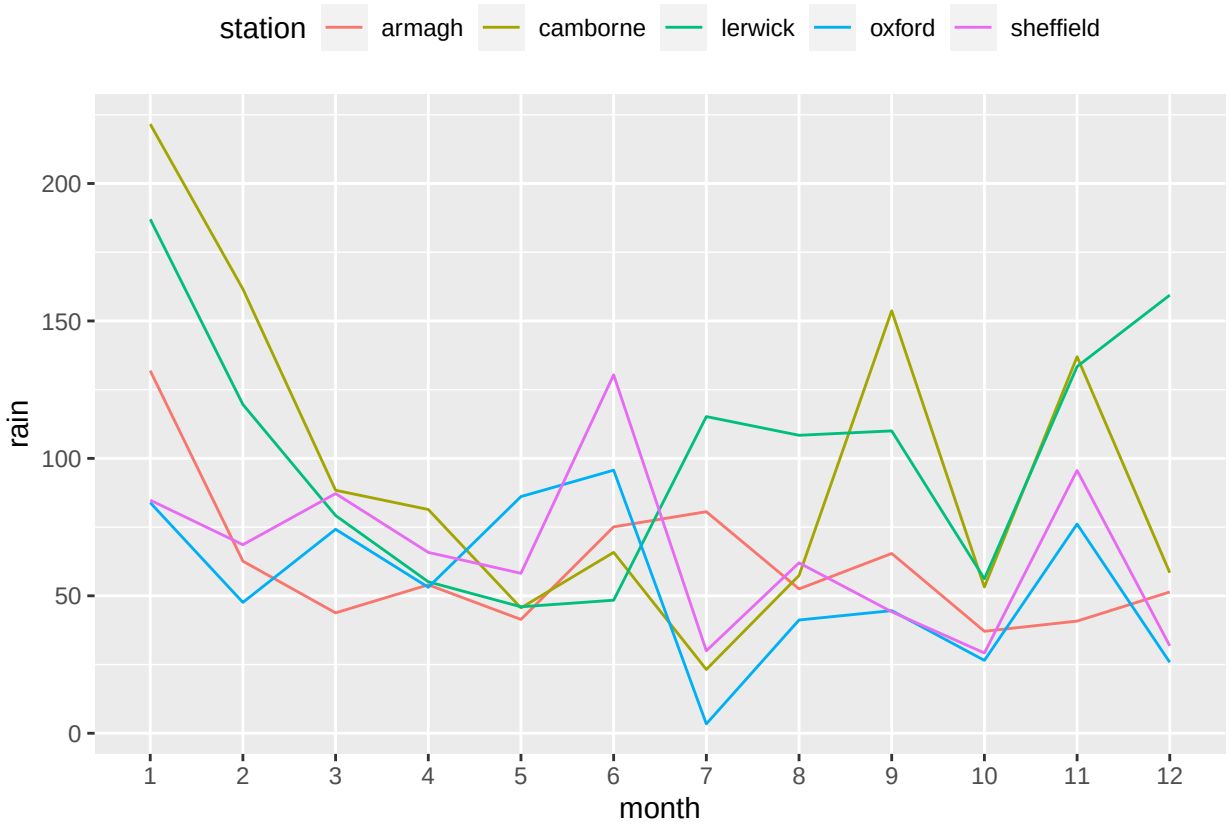
15. Change month to a factor. Make the line plot again.

```
climate <- mutate(climate, month = factor(month))

ggplot(climate, aes(x = month, y = rain, group = station, colour = station)) +
  geom_line()
```



16. Use `theme(legend.position = "top")` to move the colour legend to the top of the plot.

```
ggplot(climate, aes(x = month, y = rain, group = station, colour = station)) +
  geom_line() + theme(legend.position = "top")
```

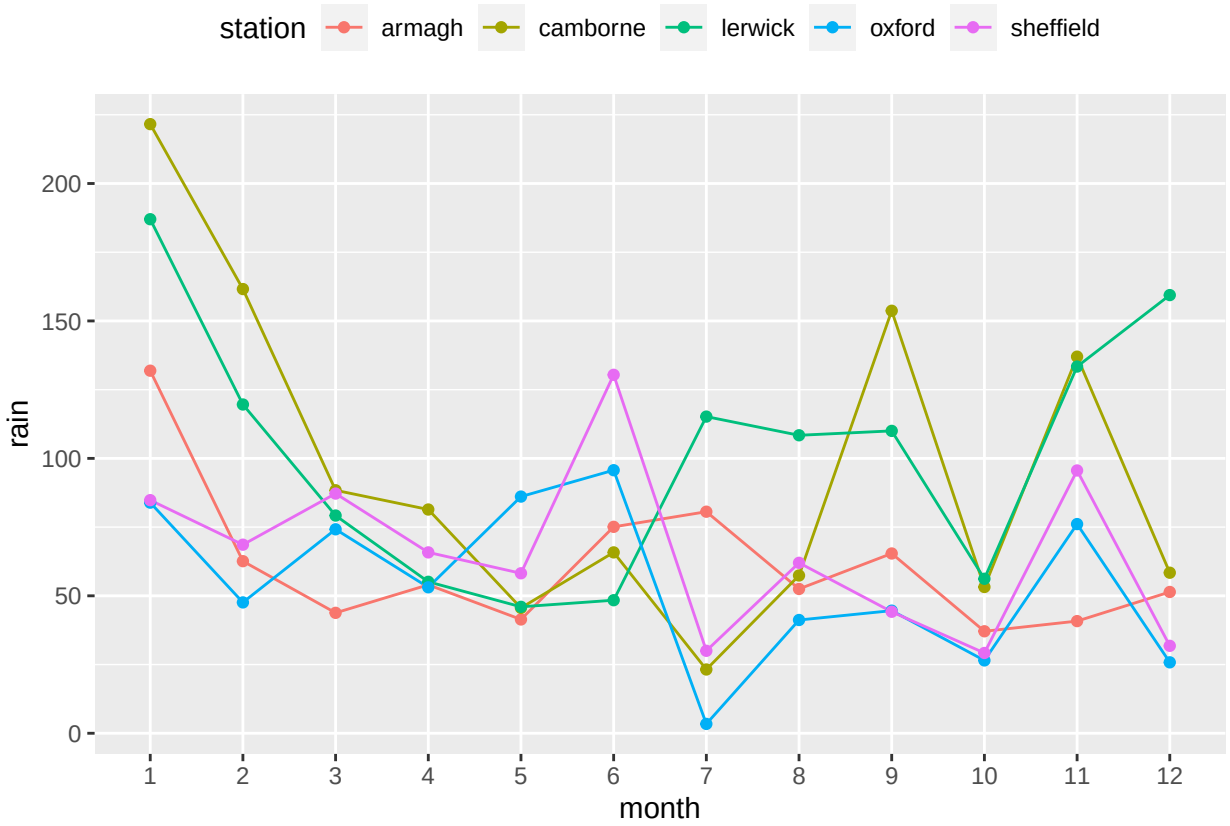


Layering

We can add several geoms to the same plot to show several things at once.

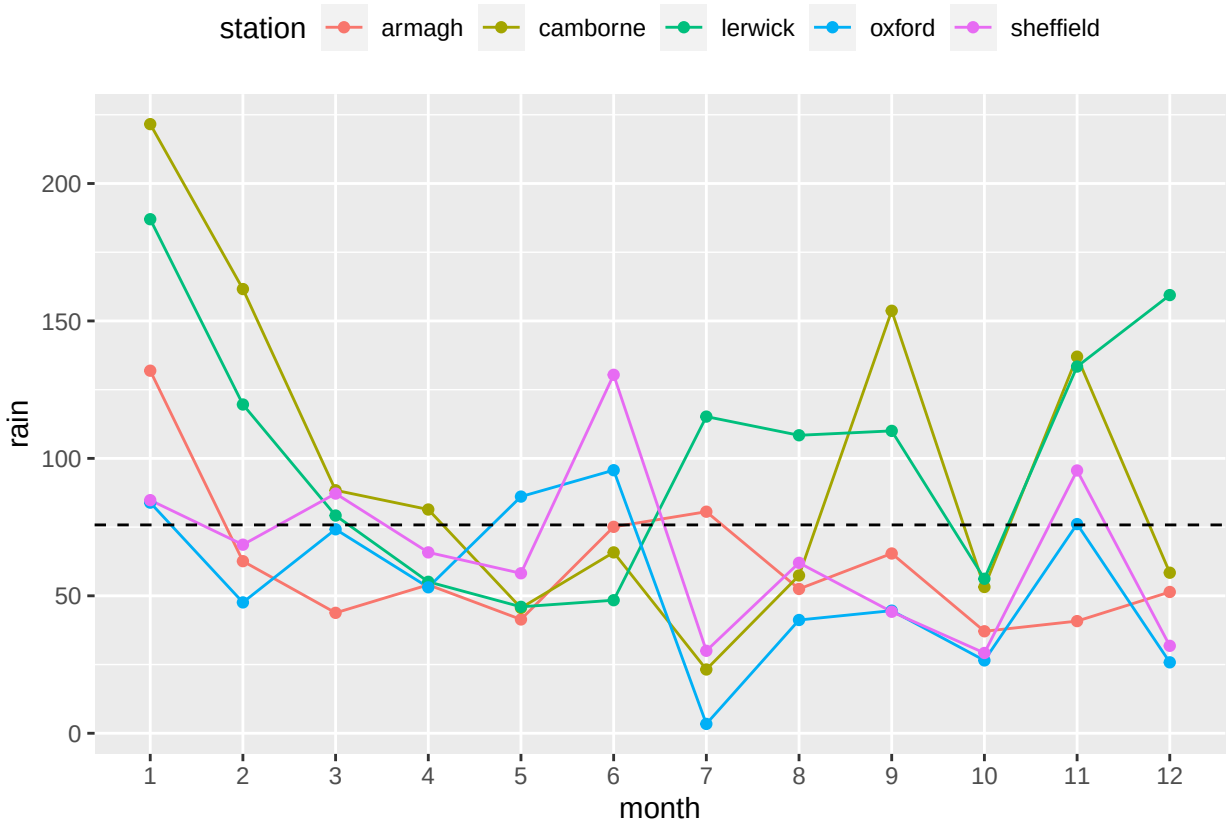
17. Use `geom_point()` to add the monthly rainfall data points to the line plot above.

```
ggplot(climate, aes(x = month, y = rain, group = station, colour = station)) +
  geom_line() +
  geom_point() +
  theme(legend.position = "top")
```



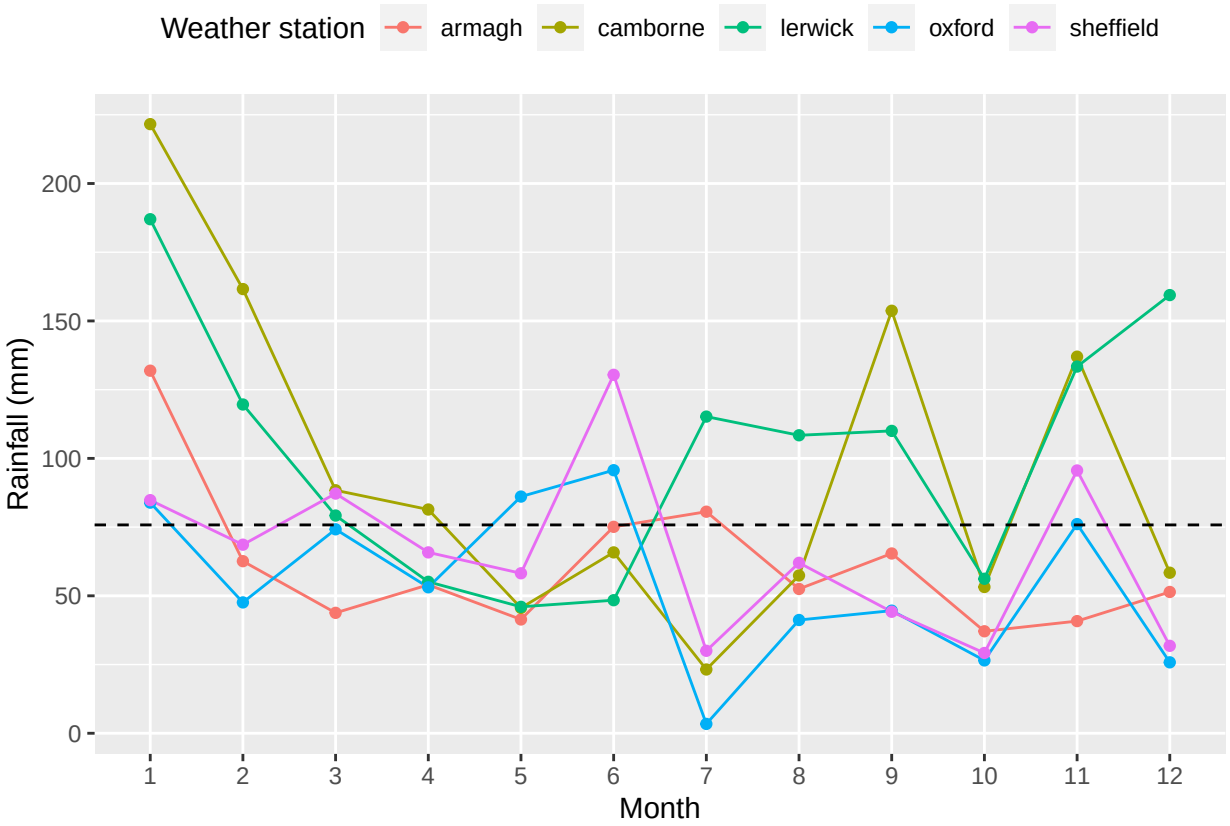
18. Now, insert `geom_hline(yintercept = mean(climate$rain), linetype = "dashed")` at the end of your code for the line plot, and update the plot again. What have you just added to the plot?

```
ggplot(climate, aes(x = month, y = rain, group = station, colour = station)) +
  geom_line() +
  geom_point() +
  theme(legend.position = "top") +
  geom_hline(yintercept = mean(climate$rain), linetype = "dashed")
```



19. Labels:

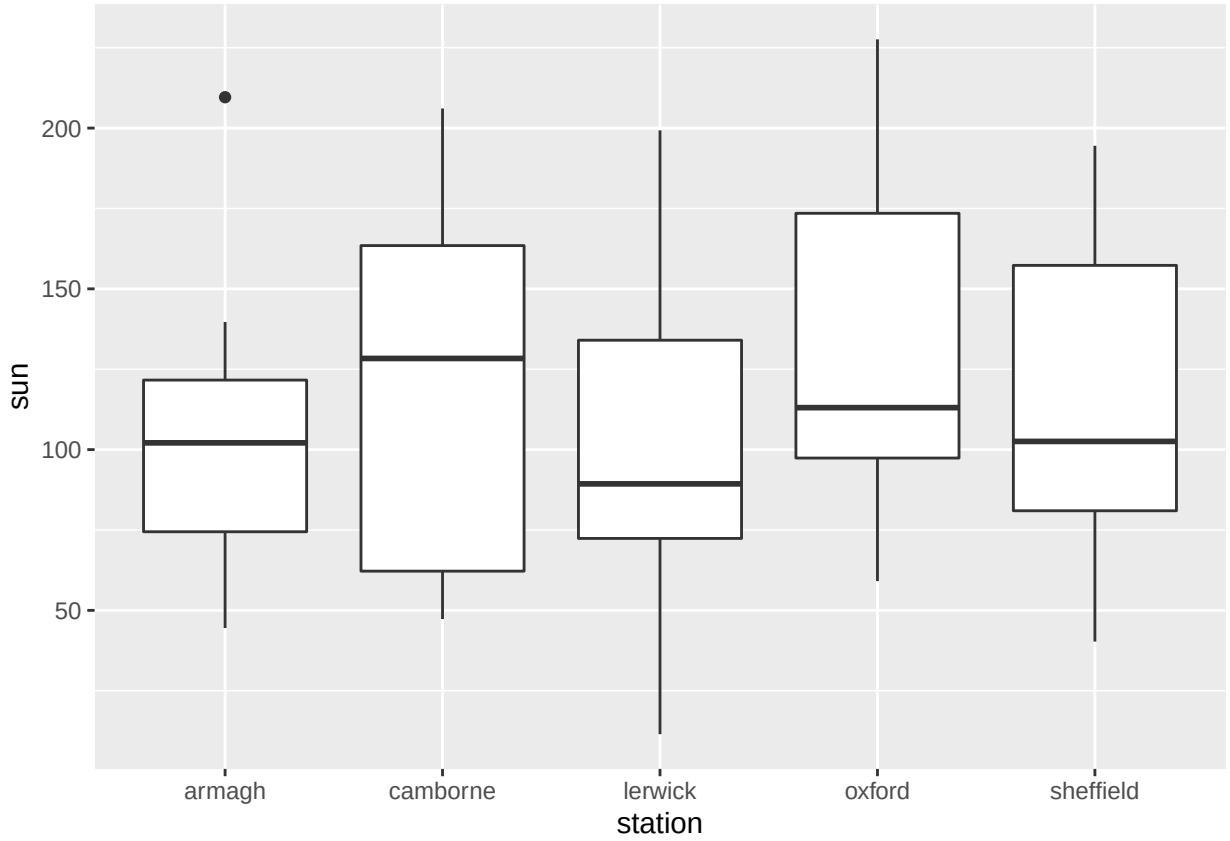
```
ggplot(climate, aes(x = month, y = rain, group = station, colour = station)) +
  geom_line() +
  geom_point() +
  theme(legend.position = "top") +
  geom_hline(yintercept = mean(climate$rain), linetype = "dashed") +
  labs(x = "Month", y = "Rainfall (mm)", colour = "Weather station")
```



Box plot I

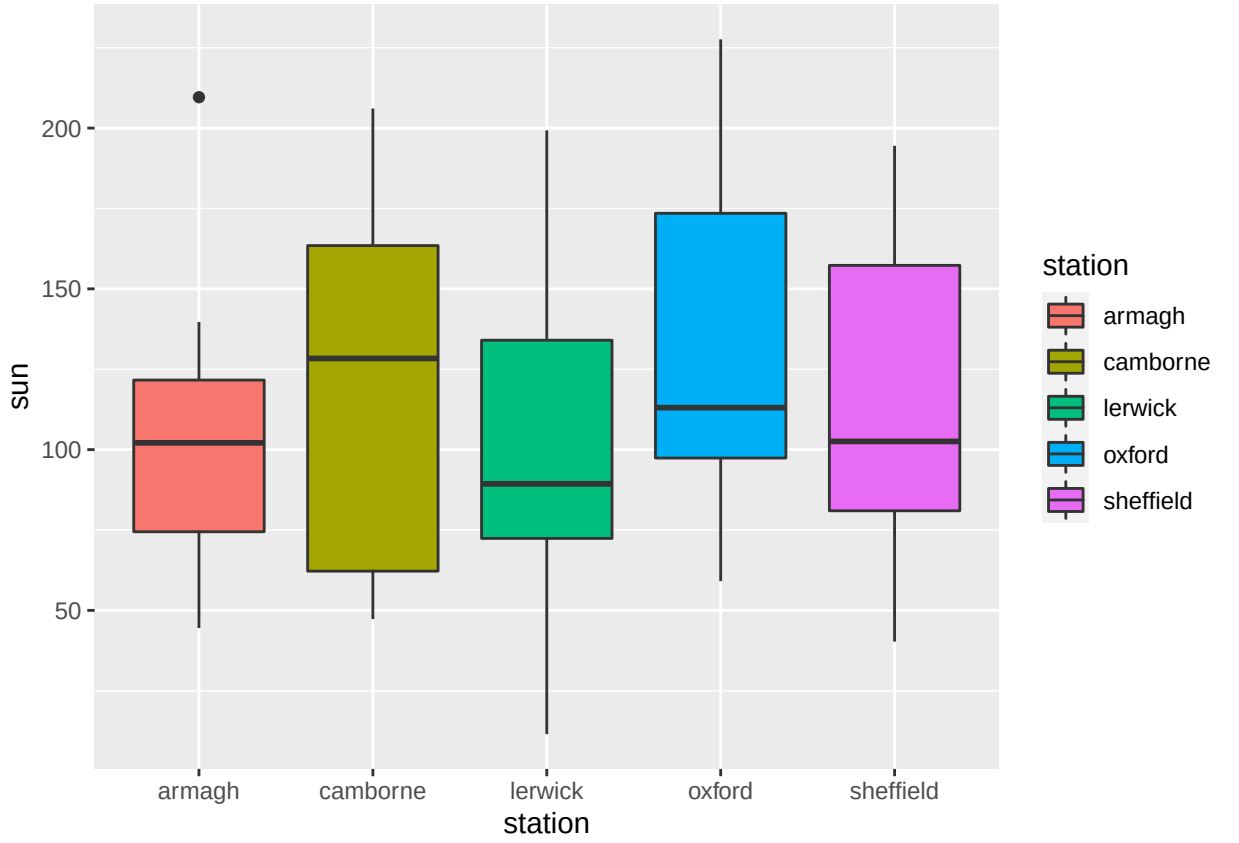
22. Make a box plot of the sunshine observations by weather station.

```
ggplot(climate, aes(x = station, y = sun)) +
  geom_boxplot()
```



23. Colour the boxes according to weather station.

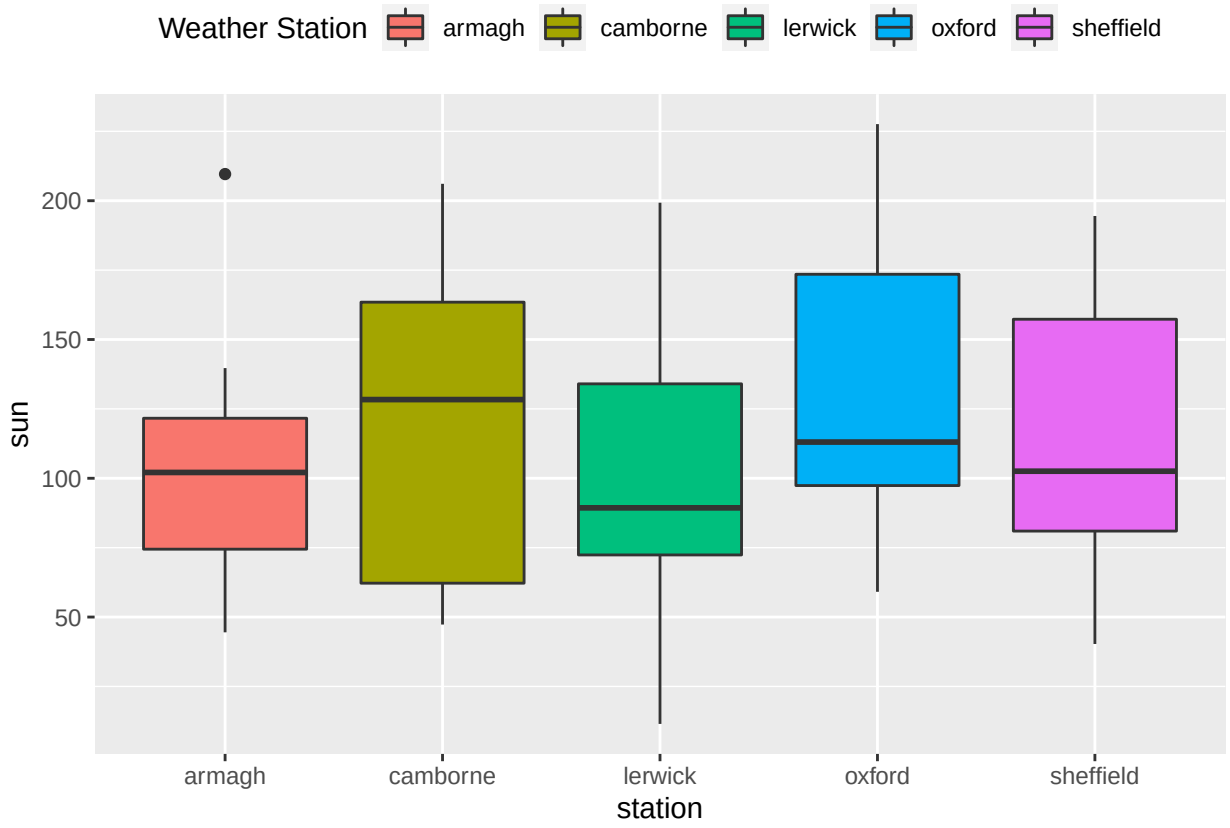
```
ggplot(climate, aes(x = station, y = sun, fill = station)) +
  geom_boxplot()
```



Box plot II - Aesthetics

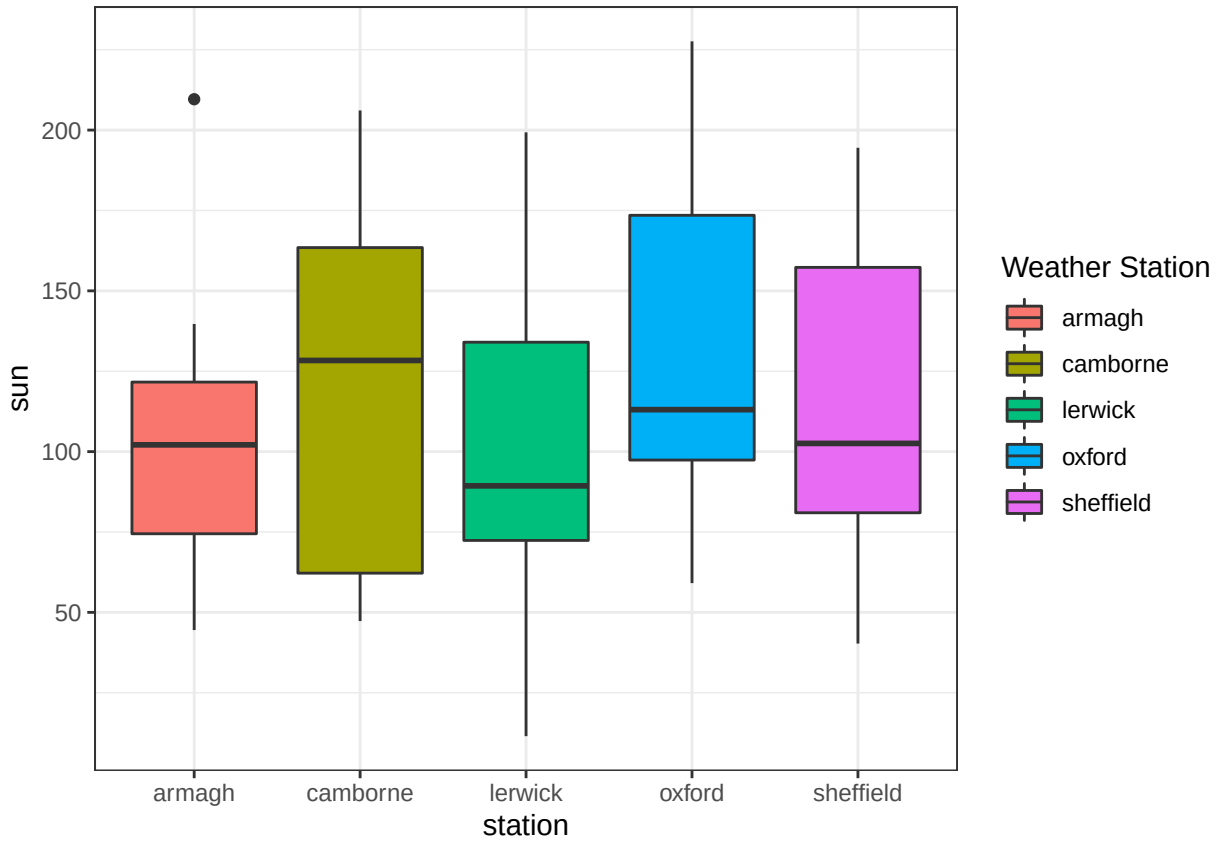
24. Move the legend with `theme(legend.position="top")` and add a different legend title with `labs(fill = "Custom Title")`.

```
ggplot(climate, aes(x = station, y = sun, fill = station)) +
  geom_boxplot() +
  theme(legend.position="top") +
  labs(fill = "Weather Station")
```



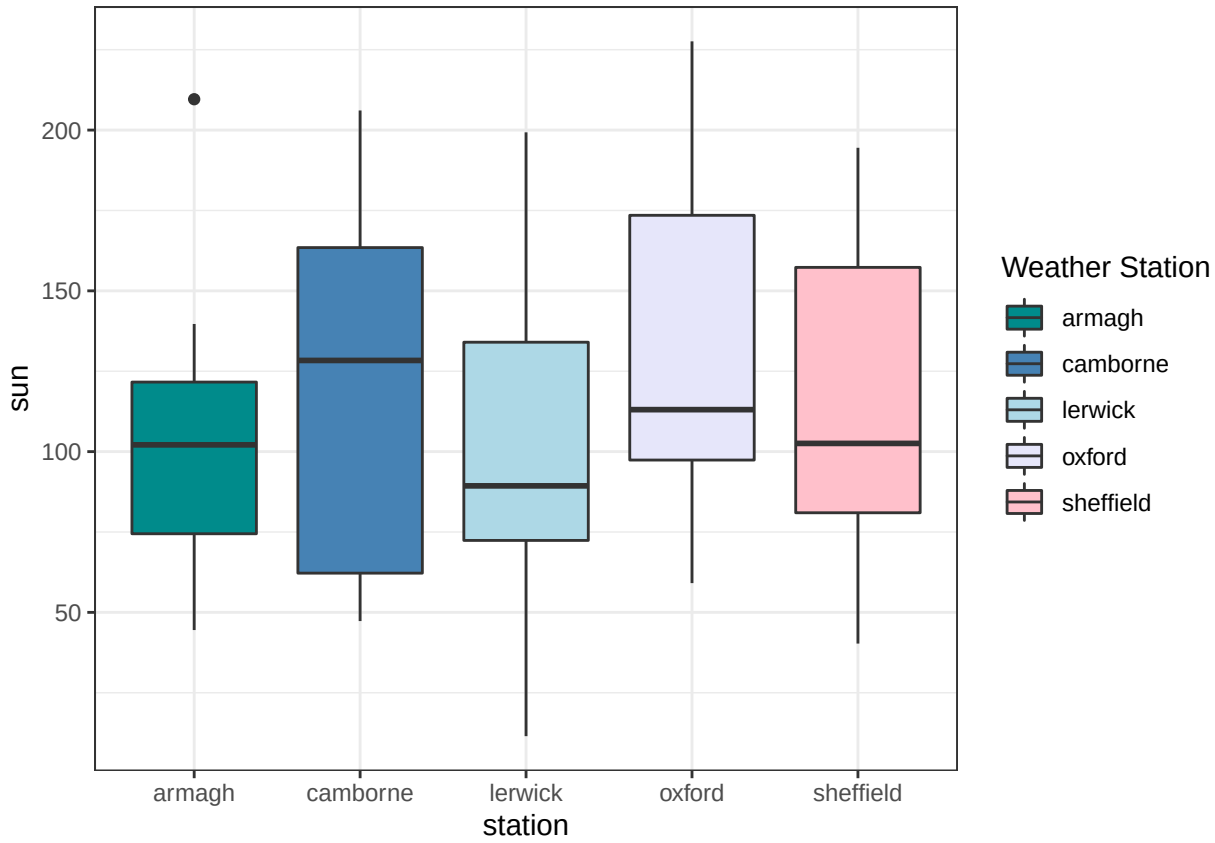
25. Change the theme of the ggplot grid. Suggestions: `theme_minimal()`, `theme_bw()`, `theme_dark()`, `theme_void()`.

```
ggplot(climate, aes(x = station, y = sun, fill = station)) +
  geom_boxplot() +
  theme(legend.position="top") +
  labs(fill = "Weather Station") +
  theme_bw()
```



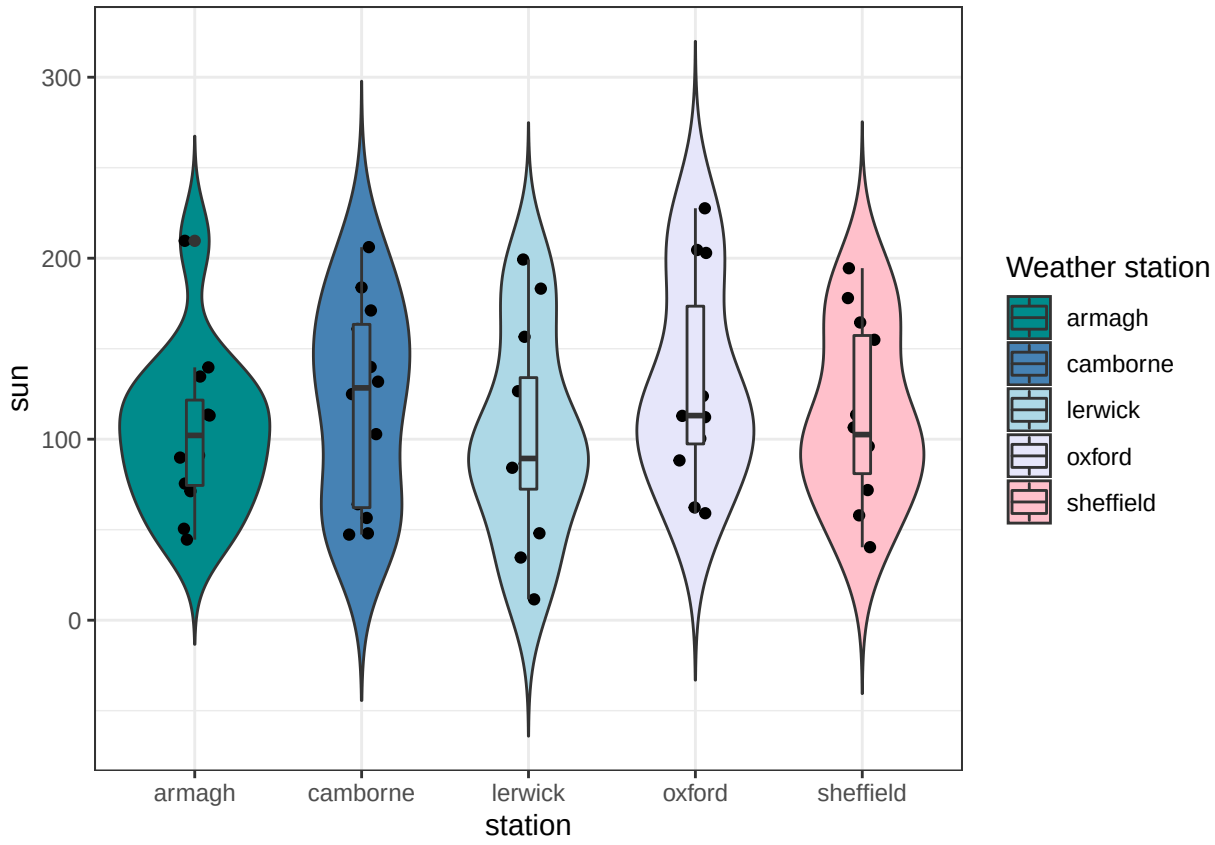
26. Instead of the colours automatically chosen when you use `fill = station`, pick your own colors. Use the `scale_fill_manual()`. You will need five colors, one for each station. What happens if you choose too few colours?

```
ggplot(climate, aes(x = station, y = sun, fill = station)) +
  geom_boxplot() +
  theme(legend.position="top") +
  labs(fill = "Weather Station") +
  theme_bw() +
  scale_fill_manual(values = c("darkcyan","steelblue", "lightblue", "lavender", "pink"))
```



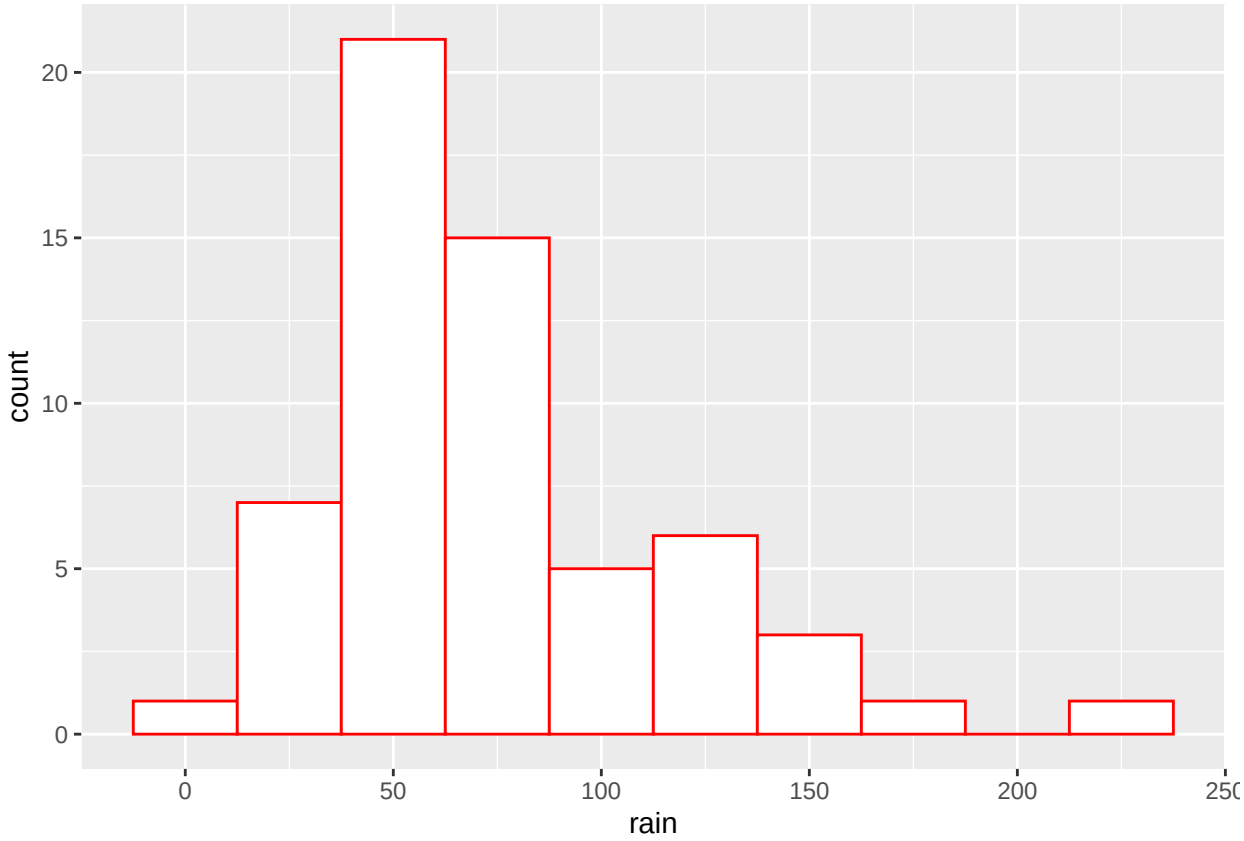
27. Change the boxplot to a **violin plot**. Add the sunshine observations as scatter points to the plot. Include a boxplot inside the violin plot, use `geom_boxplot(width=.1)` to do this.

```
ggplot(climate, aes(x=station, y=sun, fill=station)) +
  #Change the boxplot to a violin plot
  geom_violin(trim=FALSE) +
  #Add the sunshine observations as scatter points to the plot.
  geom_point(position = position_jitter(seed = 1, width = 0.1)) +
  #Include a boxplot inside the violin plot
  geom_boxplot(width=.1) +
  scale_fill_manual(values = c("darkcyan","steelblue", "lightblue", "lavender", "pink")) +
  labs(fill = "Weather station") + theme(legend.position="top") + theme_bw()
```



Histogram

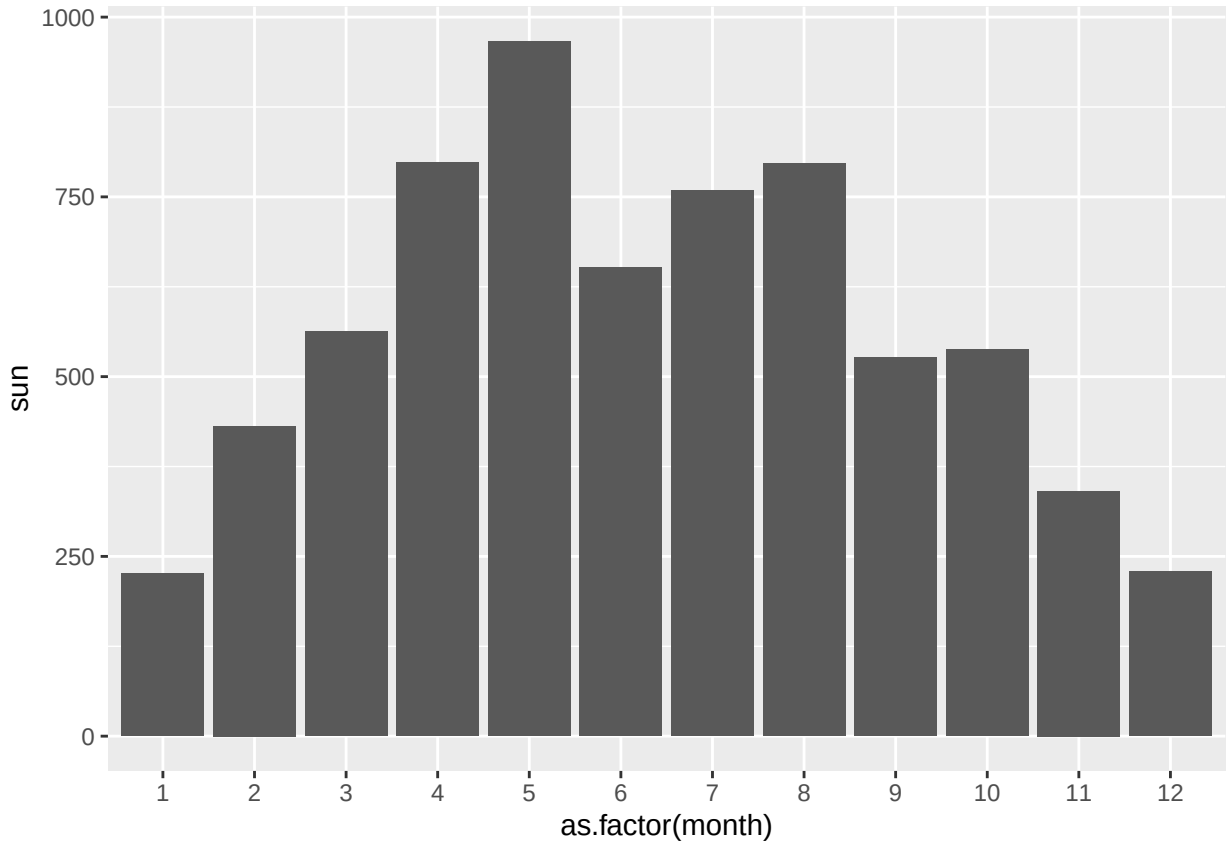
```
ggplot(climate, aes(x = rain)) +
  geom_histogram(binwidth = 25, colour = "red", fill = "white")
```



Bar chart I

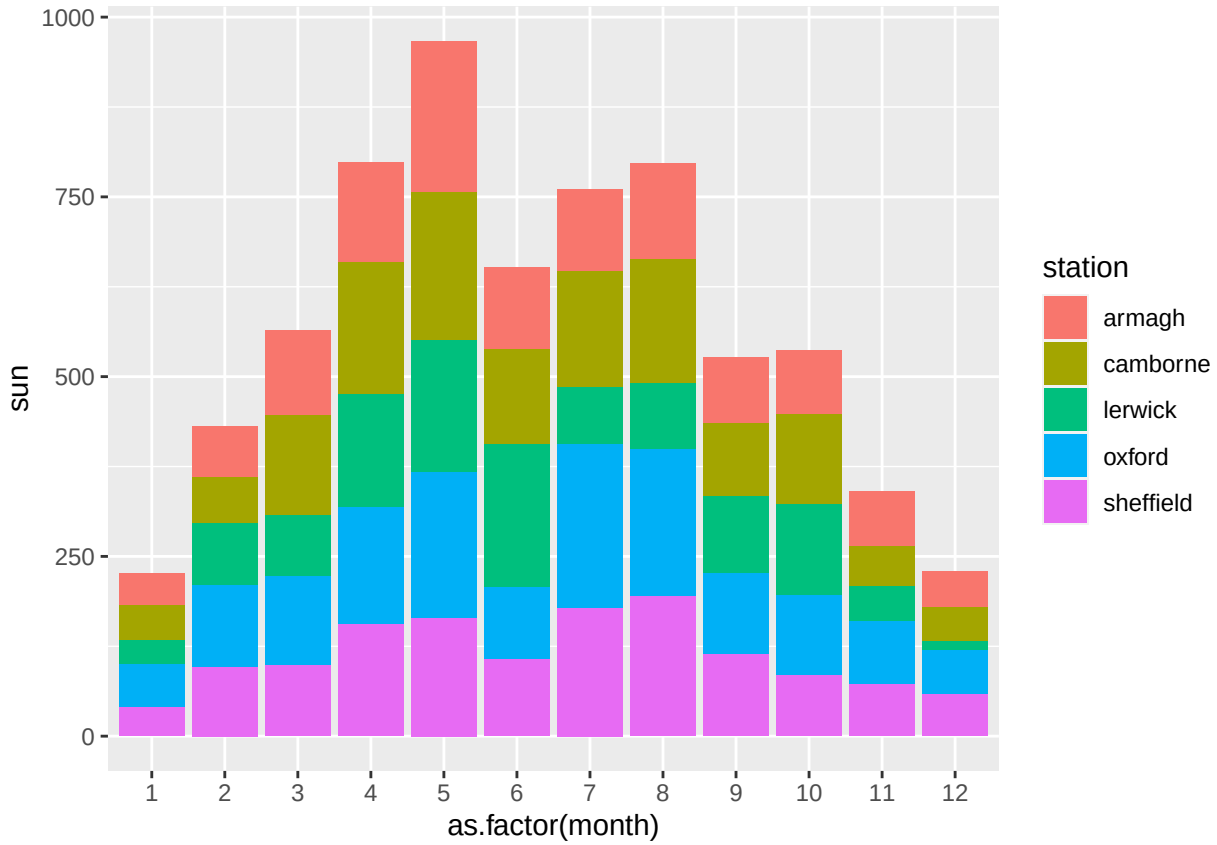
32. Make a bar chart which visualizes the sunshine hours per month.

```
ggplot(climate, aes(x=as.factor(month), y=sun)) +
  geom_col()
```



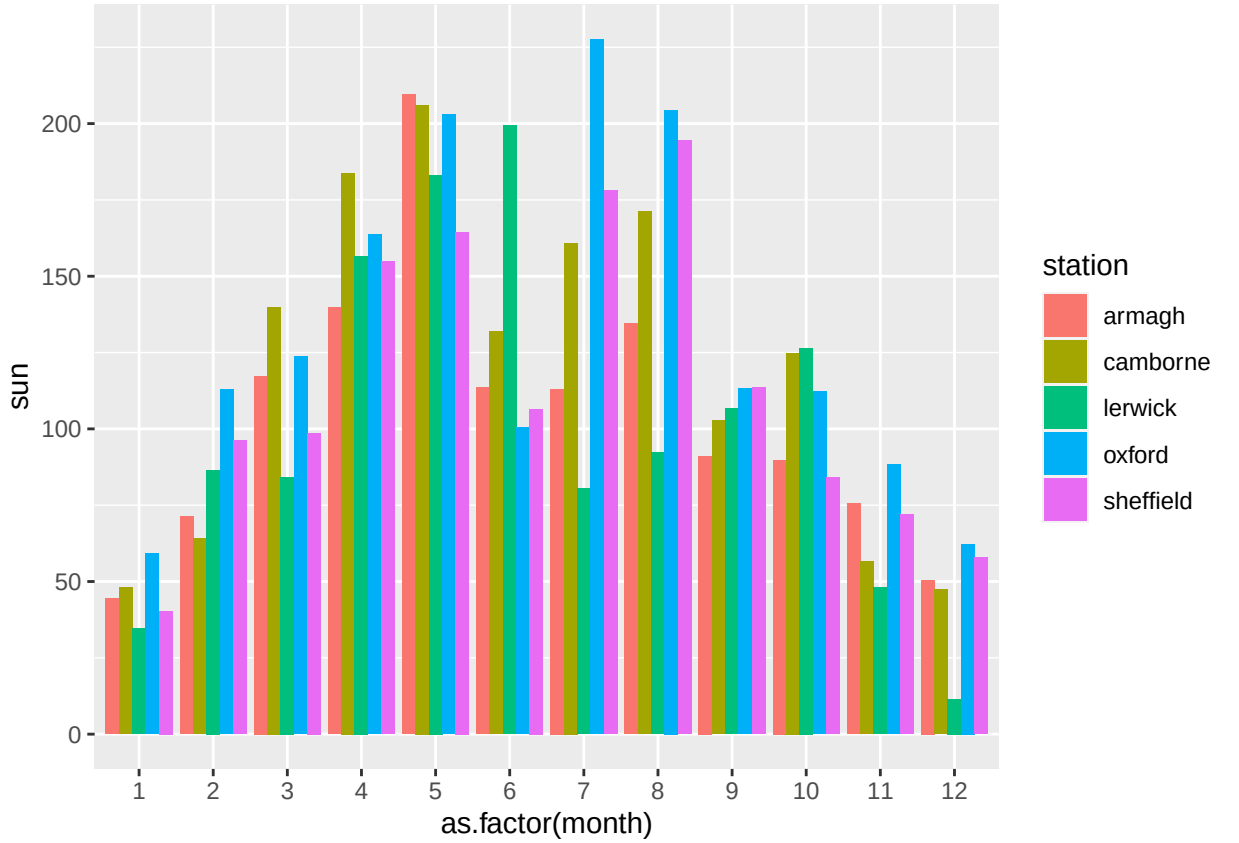
33. Colour, i.e. divide the bars according to weather station.

```
ggplot(climate, aes(x = as.factor(month), y = sun, fill = station)) +
  geom_col()
```



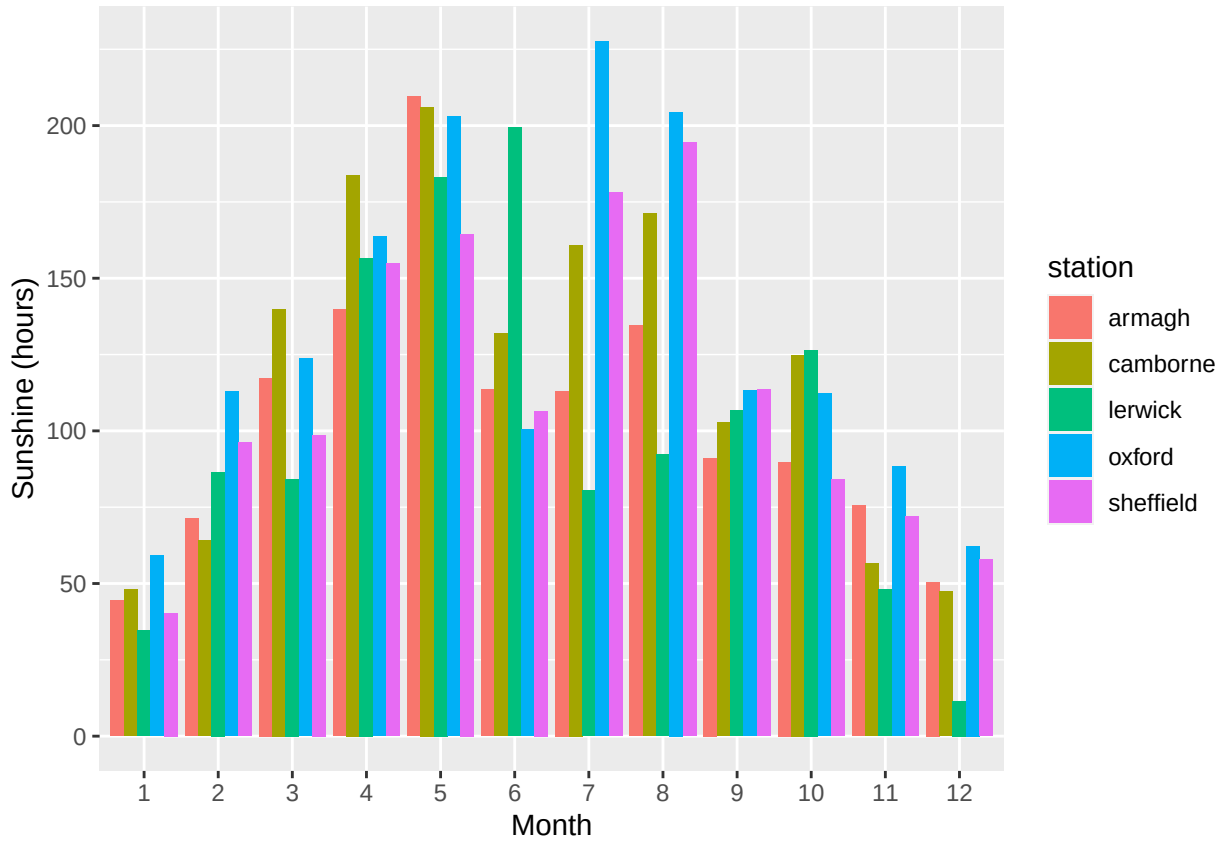
34. For better comparison, place the bars for each station next to each other instead of stacking them.

```
ggplot(climate, aes(x=as.factor(month), y=sun, fill = station)) +
  geom_col(position = 'dodge')
```



35. Make the axis labels and legend title of the plot more informative by customizing them like you did for the line plot above.

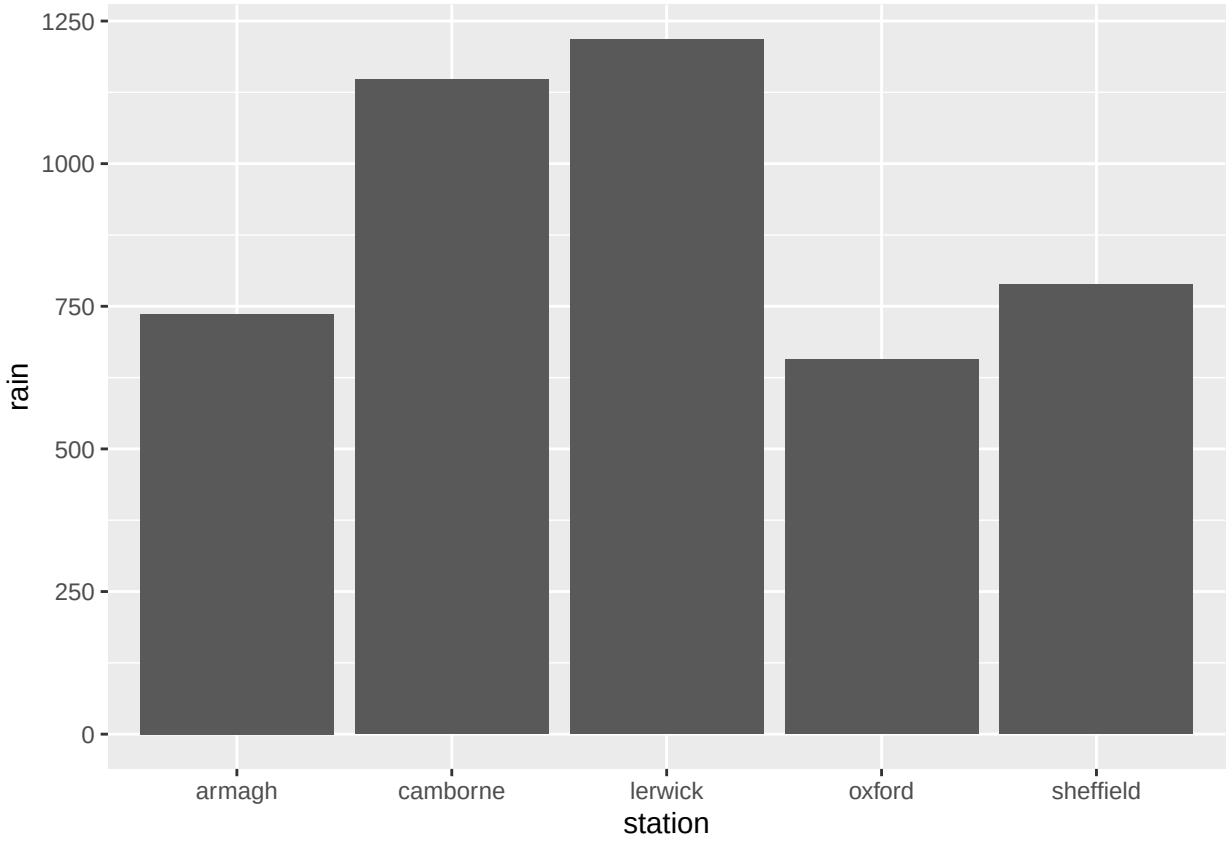
```
ggplot(climate, aes(x=as.factor(month), y=sun, fill = station)) +
  geom_col(position = 'dodge') +
  labs(x = "Month", y = "Sunshine (hours)", colour = "Weather station")
```



Bar chart II: Sorting bars

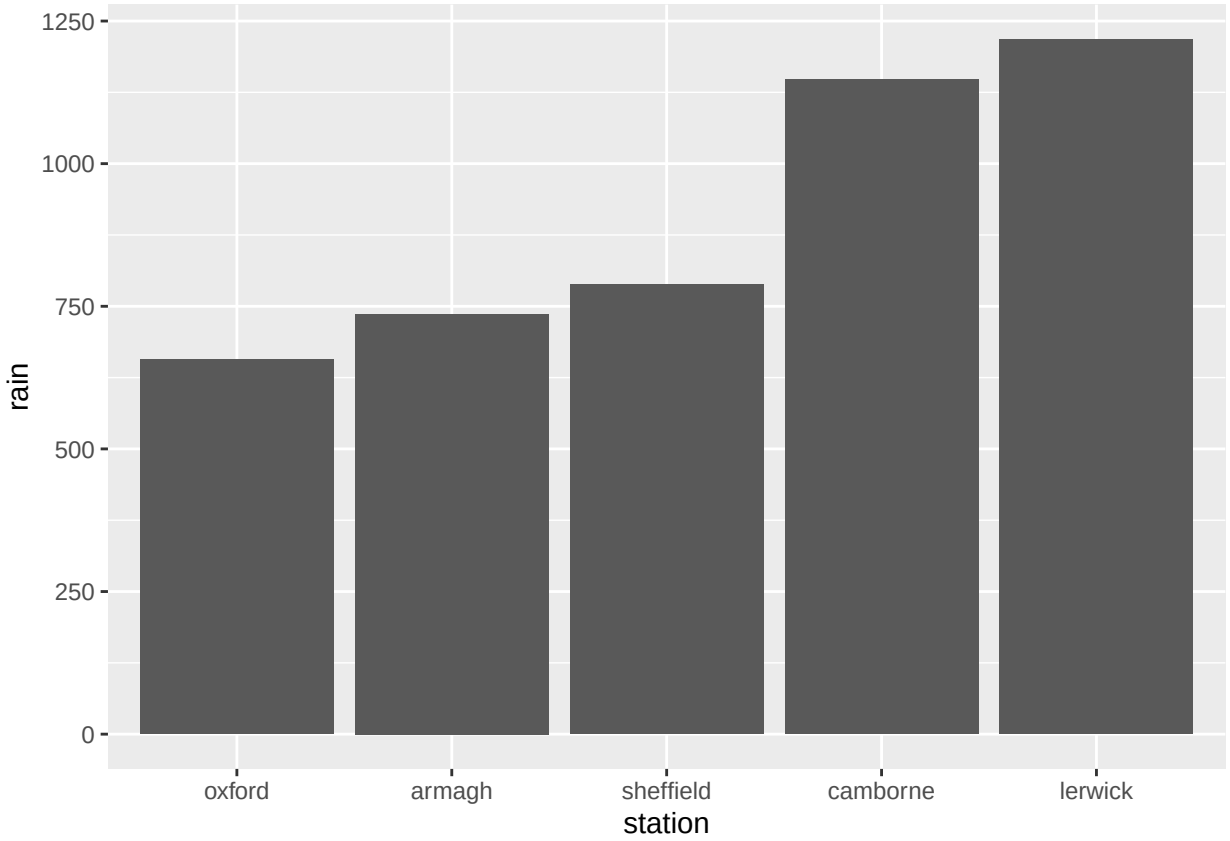
39. Make a new bar chart showing the annual rainfall recorded at each weather station.

```
ggplot(climate, aes(x = station, y = rain)) +
  geom_col()
```



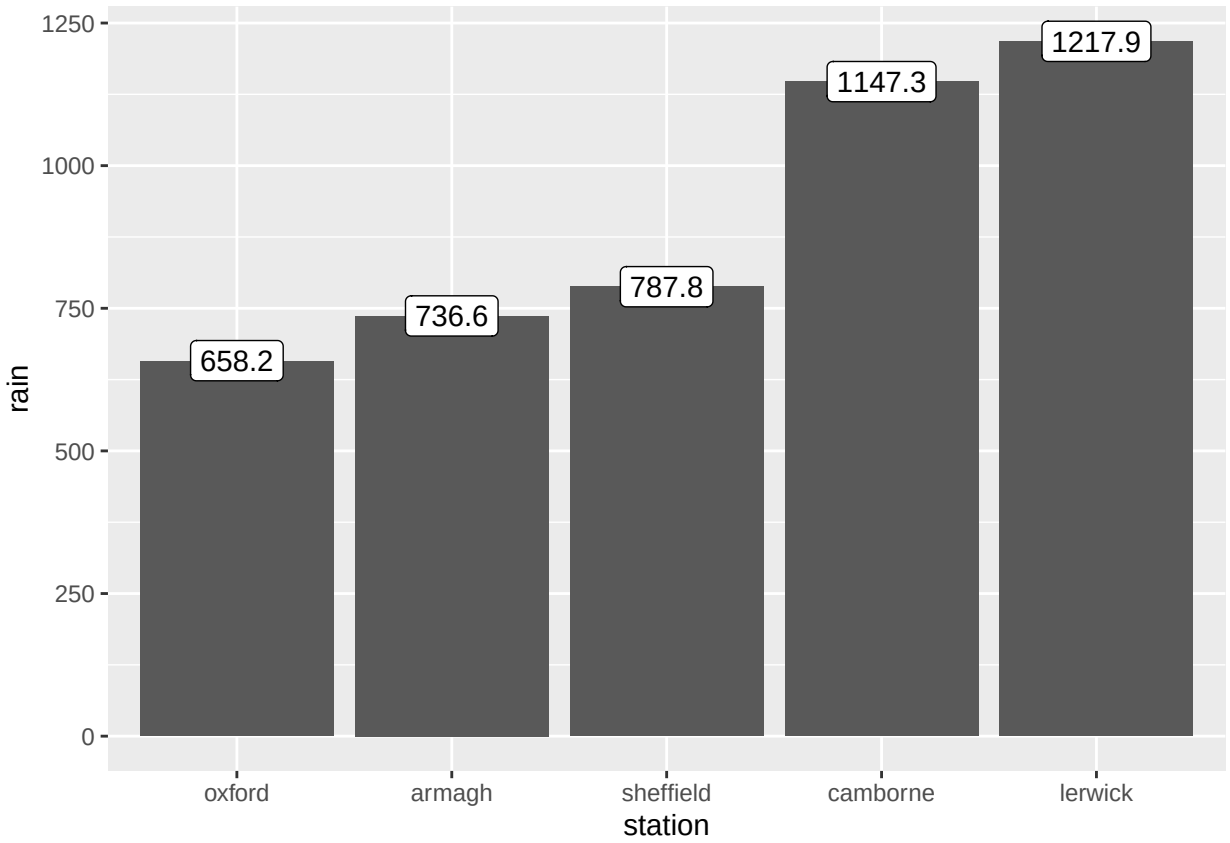
40. Sort the stations in accordance to rainfall, either ascending or descending. This was shown in the ggplot lecture.

```
annual_rain <-
  climate %>%
  group_by(station) %>%
  summarize(rain = sum(rain)) %>%
  arrange(rain)

climate <- mutate(climate, station = factor(station, levels = annual_rain$station))

ggplot(climate, aes(x = station, y = rain)) +
  geom_col()
```



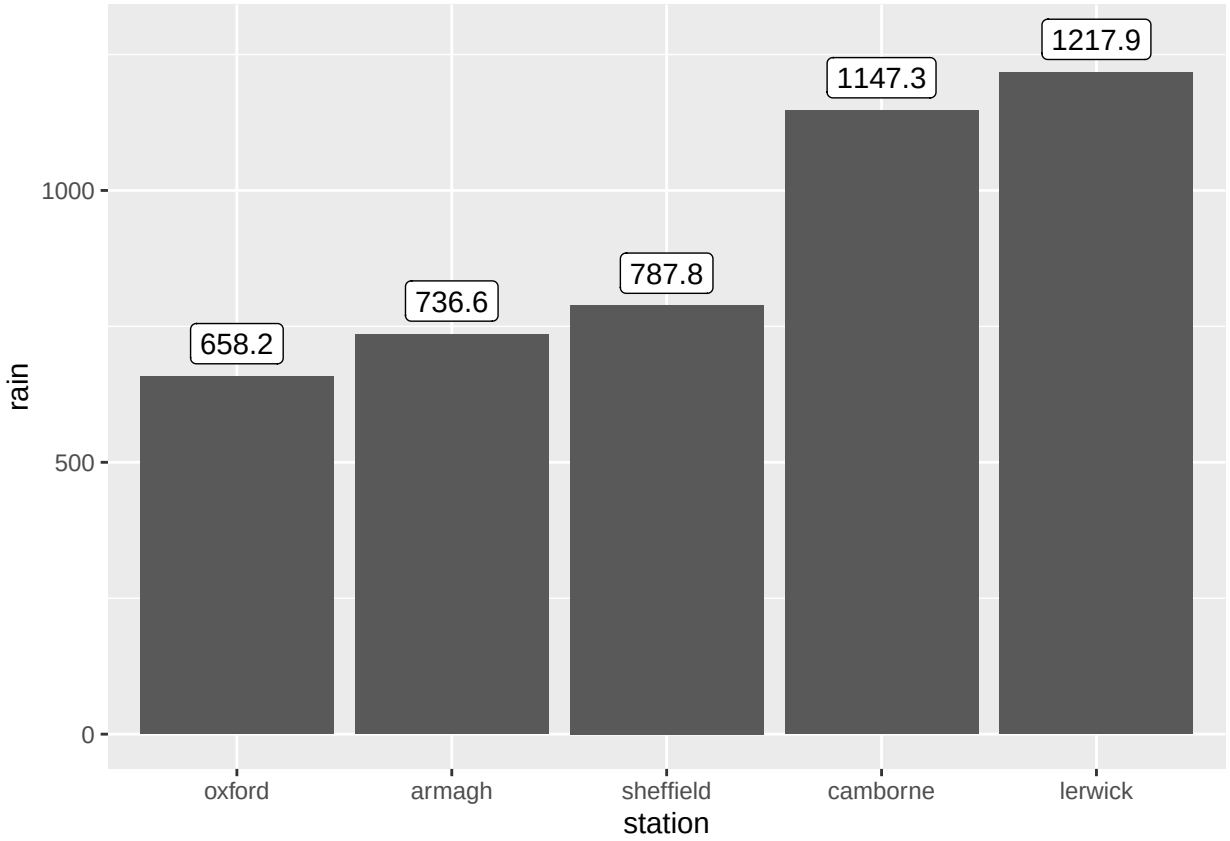
41. Add labels to each bar that state the sum of the rainfall. You can do this by using the summarized dataframe created above and passing it to `geom_text` as data, together with a suitable aes mapping. I.e. if you named it `annual_rain` you can write:

```
ggplot(climate, aes(x = station, y = rain)) +
  geom_col() +
  geom_label(mapping = aes(x = station, y = rain, label = rain), data = annual_rain)
```



42. Adjust the label positions so that the labels are positioned immediately above the bars.

```
ggplot(climate, aes(x = station, y = rain)) +
  geom_col() +
  geom_label(mapping = aes(x = station, y = rain, label = rain), data = annual_rain, nudge_y = 60)
```


1217.9
1147.3
787.8
736.6
658.2
1000
500
0
rain
oxford
armagh
sheffield
camborne
lerwick
station