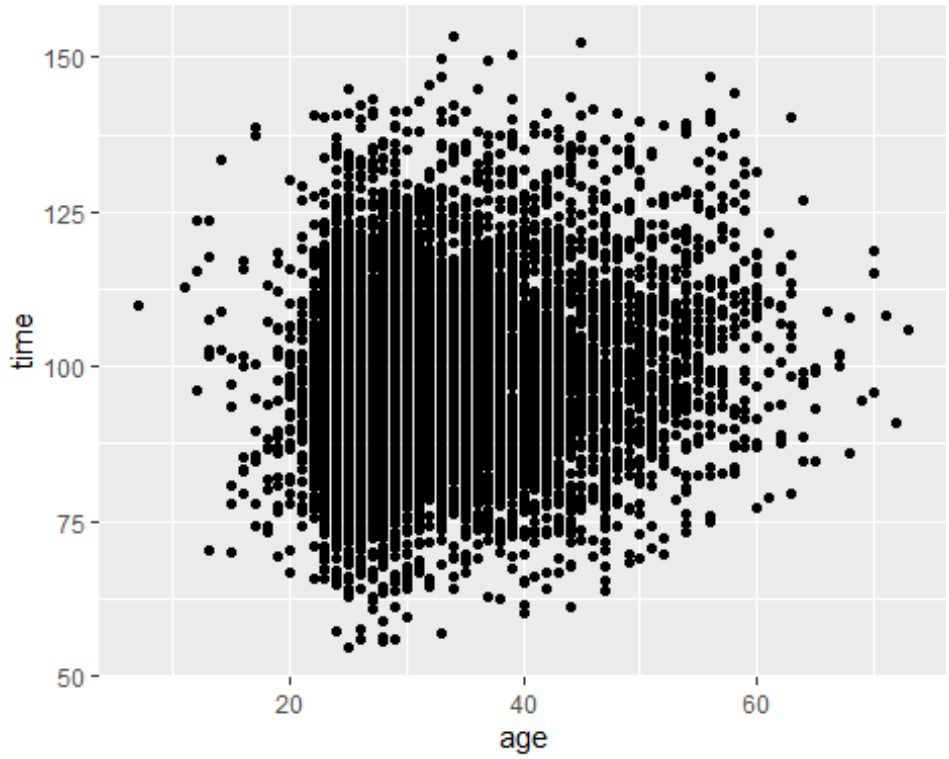
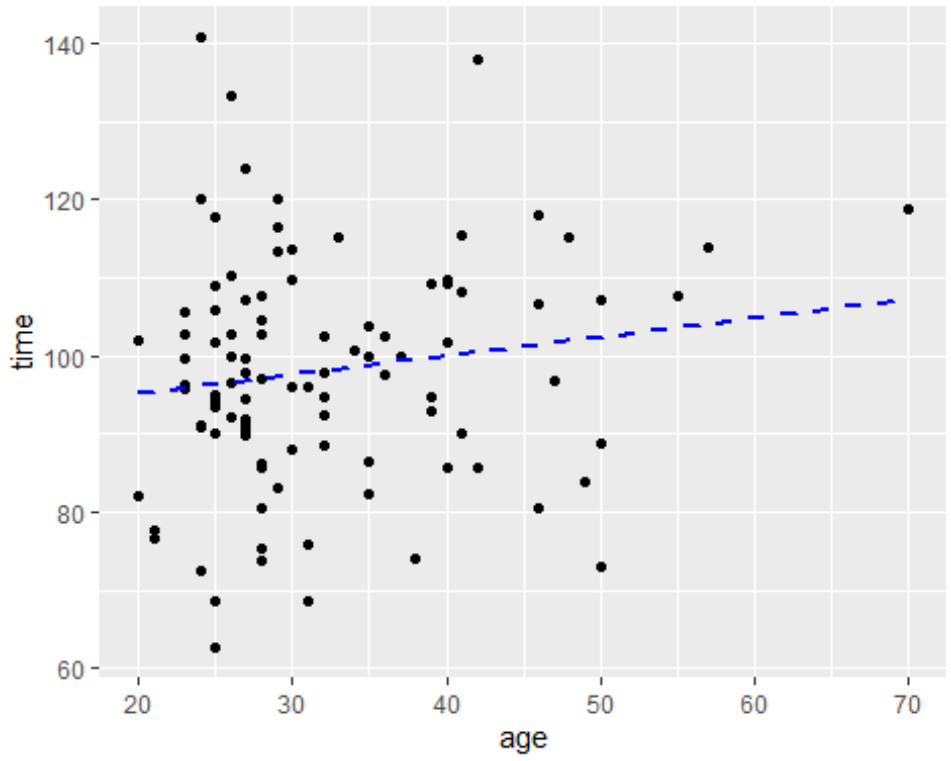
Two-Variable Graphs

```
# As usual, we load the mosaic and tidyverse packages.
library(mosaic)
library(tidyverse)

# We make use of the run09 data frame, which is included in the
# cherryblossom package.
library(cherryblossom)
head(run09)

## # A tibble: 6 x 14
##   place  time net_time  pace   age gender first  last  city  state country
div
##   <int> <dbl>    <dbl> <dbl> <int> <fct>  <fct>  <fct> <fct> <fct> <fct>
<int>
## 1     1  53.5     53.5  5.37    21 F      Lineth Chep~ Kenya NR    KEN
2
## 2     2  53.9     53.9  5.4     21 F      Belia~ Gebre Ethi~ NR    ETH
2
## 3     3  54.0     54.0  5.4     22 F      Teyba  Naser Ethi~ NR    ETH
2
## 4     4  54.4     54.4  5.45    19 F      Abebu  Gelan Ethi~ NR    ETH
1
## 5     5  54.4     54.4  5.45    36 F      Cathe~ Nder~ Kenya NR    KEN
5
## 6     6  54.5     54.5  5.47    28 F      Olga   Roma~ Russ~ NR    RUS
3
## # ... with 2 more variables: div_place <int>, div_tot <int>

# Let's see how many rows there are.
nrow(run09)

## [1] 14974

# That's a lot!  Let's get a smaller data set to work with.  We'll
# "filter" the data frame to obtain just the female runners from the
# United States.  We'll also select just some of the columns and combine
# some of the divisions
cb09data <- run09 %>%
  filter(gender == "F" & country == "USA" & div != "NA") %>%
  select(place, time = net_time, age, state, div, div_place) %>%
  mutate(div = ifelse(div < 11, div, 10))
head(cb09data)

## # A tibble: 6 x 6
##   place  time   age state   div div_place
##   <int> <dbl> <int> <fct> <dbl>     <int>
## 1     7  54.6    25 NR        3         2
```

```
## 2    14  55.7    28 CO        3         5
## 3    15  55.8    26 NR        3         6
## 4    16  55.9    29 NR        3         7
## 5    17  56.3    28 NR        3         8
## 6    19  56.8    33 TX        4         2

# Scatter Plots.
# To produce a scatter plot, use gf_point() or ggplot() along with
# geom_point().  Let's make a scatter plot of time vs. age for the
# female runners.
gf_point(time ~ age, data = cb09data)
```



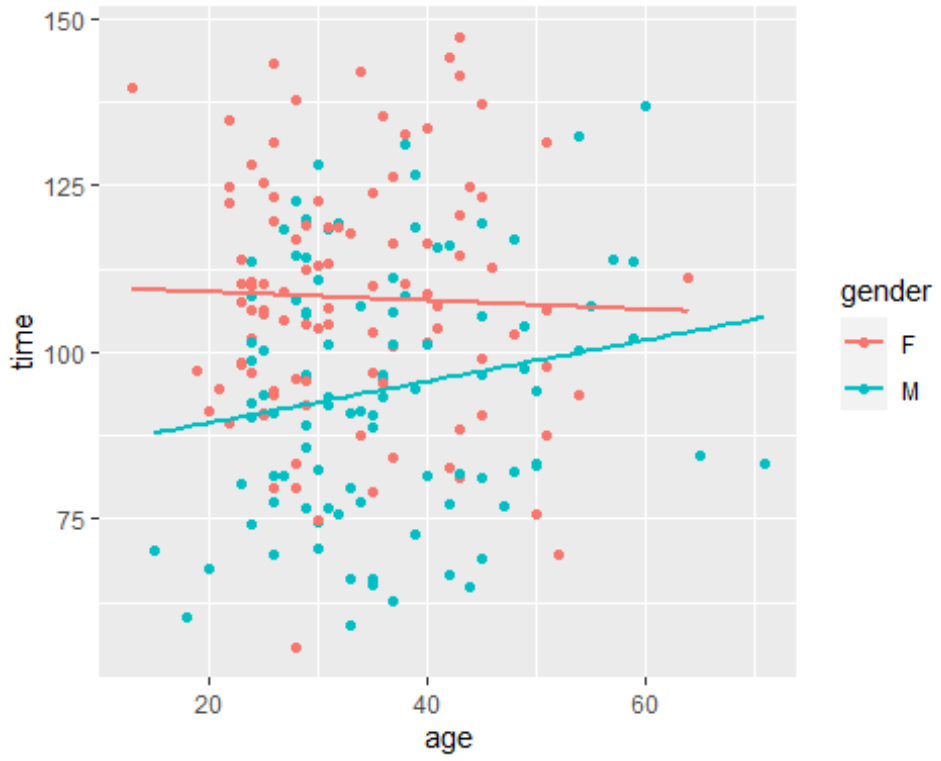

```
# That's still a lot of points!  Let's plot a sample of size 100.
# Also, we can add a regression line using gm_lm().
set.seed(0122)
gf_point(time ~ age, data = sample(cb09data,100)) %>%
  gf_lm(color = "blue", linetype = "dashed", size = 1)
```



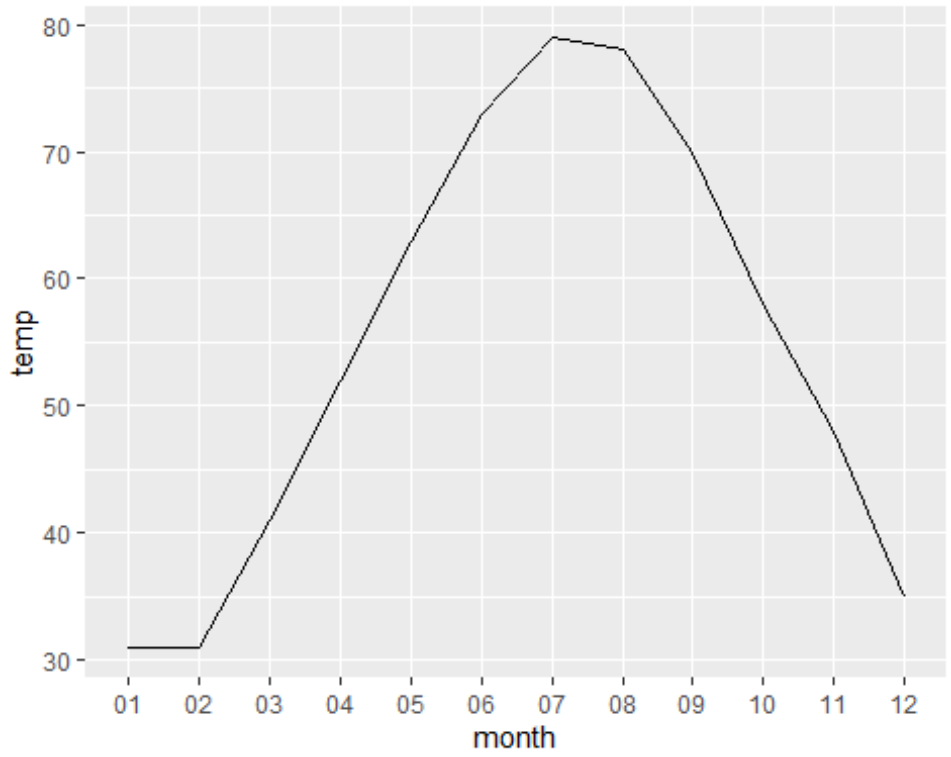

```
# Do help("gf_point") to see what options are available.  For
# example, there are options to control the color, size, and
# shape of the points.
gf_point(time ~ age, data = sample(cb09data,100),
         color = "red", size = 2, shape = "triangle")
```


```
# An example using ggplot() and geom_point().  We use a sample of
# size 200 from run09, and color the points and according to gender.
# Also, we add a regression line for each gender.
ggplot(data = sample(run09,200), aes(x = age, y = time)) +
  geom_point(aes(color = gender)) +
  geom_smooth(method = "lm", se = 0, aes(color = gender))
```

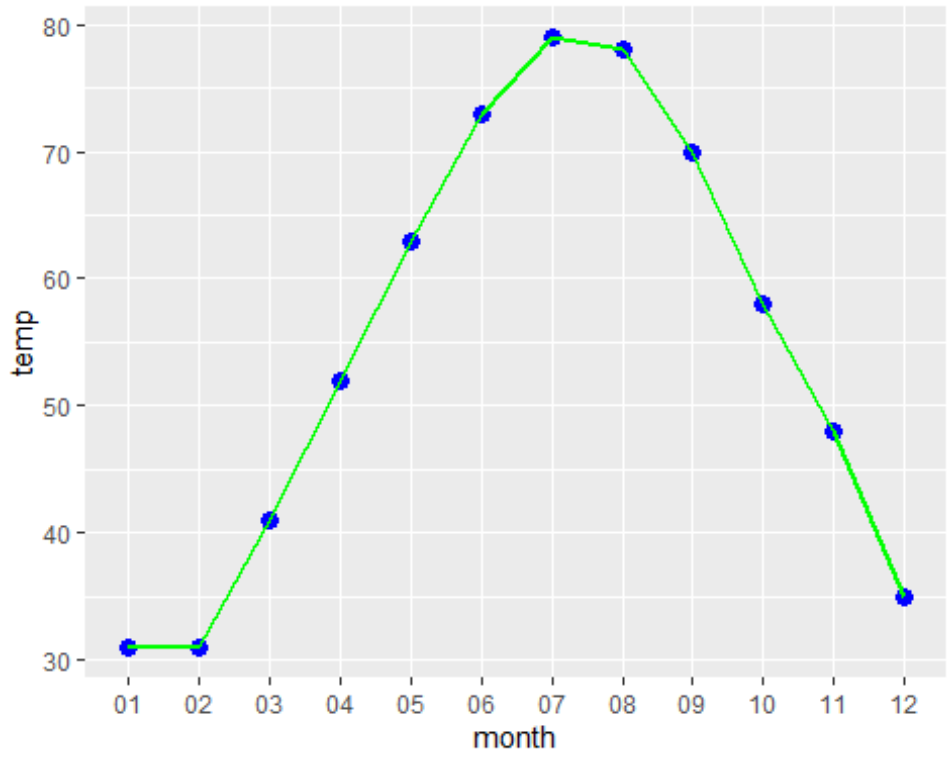
```
## `geom_smooth()` using formula 'y ~ x'
```




```
# Line Graphs.
# Let's use gf_line() to plot a line graph of average high
# temperature by month for Buffalo, NY.
month <- c("01","02","03","04","05","06",
           "07","08","09","10","11","12")
temp <- c(31,31,41,52,63,73,79,78,70,58,48,35)
df <- tibble(month,temp)
gf_line(temp ~ month, data = df, group = 1)
```



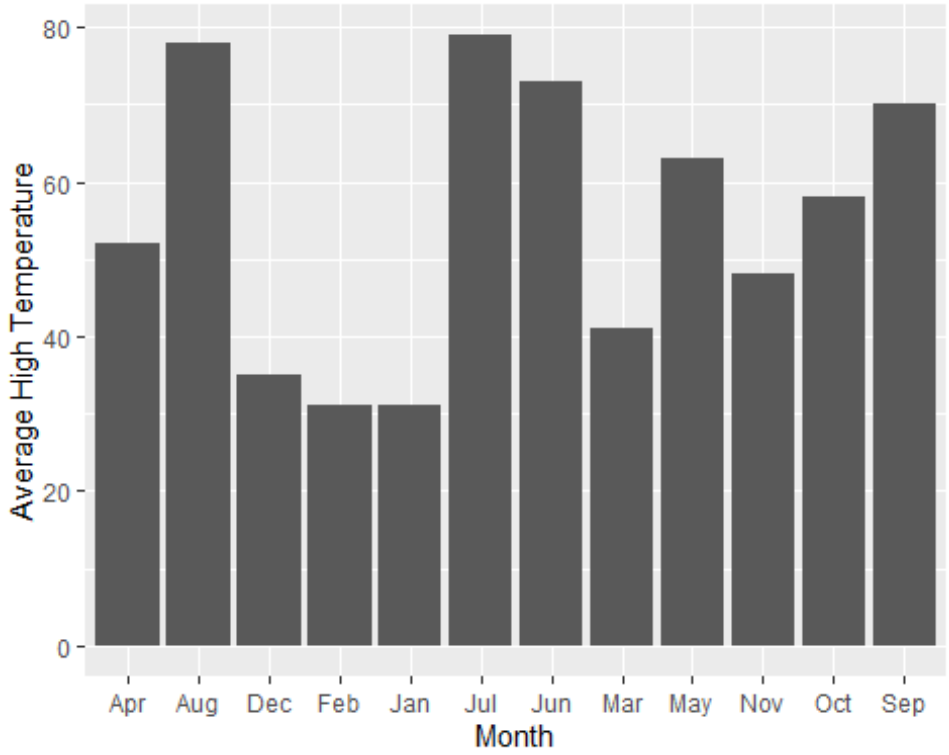

```
# Using ggplot(), along with geom_point() and geom_line(),
# to produce the graph below.
# Exercise: Experiment with different colors and styles!
ggplot(data = df, aes(x = month, y = temp)) +
  geom_point(size = 3, color = "blue") +
  geom_line(size = 1, color = "green", group = 1)
```



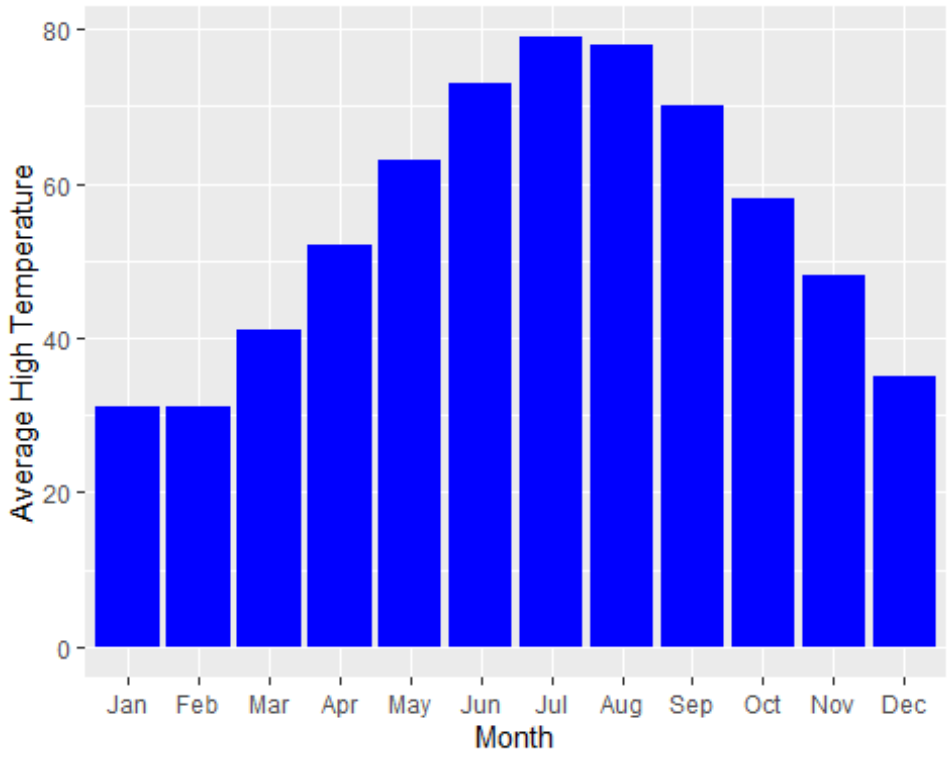

```
# Column Graphs.
# We can produce a column graph using ggplot() and geom_col().
# For example, let's produce a column graph showing the average
# high temperature for each month in Buffalo.
# First, we add a column to df with the names of the months.
month_name = c("Jan","Feb","Mar","Apr","May","Jun",
               "Jul","Aug","Sep","Oct","Nov","Dec")
df2 <- df
df2[,3] <- month_name
names(df2)[3] <- "month_name"
df2
```

```
## # A tibble: 12 x 3
##    month  temp month_name
##    <chr> <dbl> <chr>
##  1 01       31 Jan
##  2 02       31 Feb
##  3 03       41 Mar
##  4 04       52 Apr
##  5 05       63 May
##  6 06       73 Jun
##  7 07       79 Jul
##  8 08       78 Aug
##  9 09       70 Sep
## 10 10       58 Oct
## 11 11       48 Nov
## 12 12       35 Dec
```

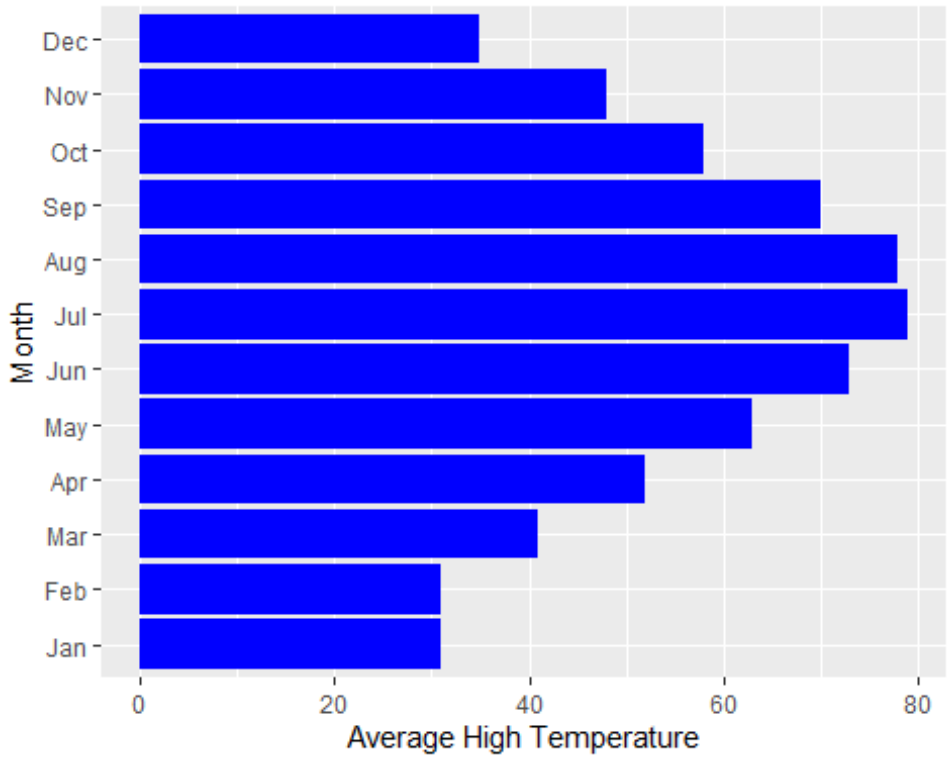
```
ggplot(data = df2, aes(x = month_name, y = temp)) +
  geom_col() +
  labs(x = "Month", y = "Average High Temperature")
```




```
# How can we get the months to display in the correct order?
# Here's one solution.
ggplot(data = df2, aes(x = reorder(month_name,as.integer(month)),
                       y = temp)) +
  geom_col(fill = "blue") +
  labs(x = "Month", y = "Average High Temperature")
```




```
# One can make the columns horizontal by interchanging "x" and
# "y".
ggplot(data = df2, aes(y = reorder(month_name,as.integer(month)),
                       x = temp)) +
  geom_col(fill = "blue") +
  labs(y = "Month", x = "Average High Temperature")
```




```
# Exercise: Figure out how to reverse the order of the months
# in the above plot, so that "Jan" is at the top and "Dec" is
# at the bottom.
```