Chapter 2

Introduction to ggplot2

This section provides an brief overview of how the ggplot2 package works. If you are simply seeking code to make a specific type of graph, feel free to skip this section. However, the material can help you understand how the pieces fit together.

2.1 A worked example

The functions in the ggplot2 package build up a graph in layers. We'll build a a complex graph by starting with a simple graph and adding additional elements, one at a time.

The example uses data from the 1985 Current Population Survey to explore the relationship between wages (*wage*) and experience (*expr*).

```
# load data
data(CPS85 , package = "mosaicData")
```

In building a ggplot2 graph, only the first two functions described below are required. The other functions are optional and can appear in any order.

2.1.1 ggplot

The first function in building a graph is the ggplot function. It specifies the

- data frame containing the data to be plotted
- the mapping of the variables to visual properties of the graph. The mappings are placed within the aes function (where *aes* stands for aesthetics).

```
# specify dataset and mapping
library(ggplot2)
ggplot(data = CPS85,
       mapping = aes(x = exper, y = wage))
```

Why is the graph empty? We specified that the *exper* variable should be mapped to the *x*-axis and that the *wage* should be mapped to the *y*-axis, but we haven't yet specified what we wanted placed on the graph.

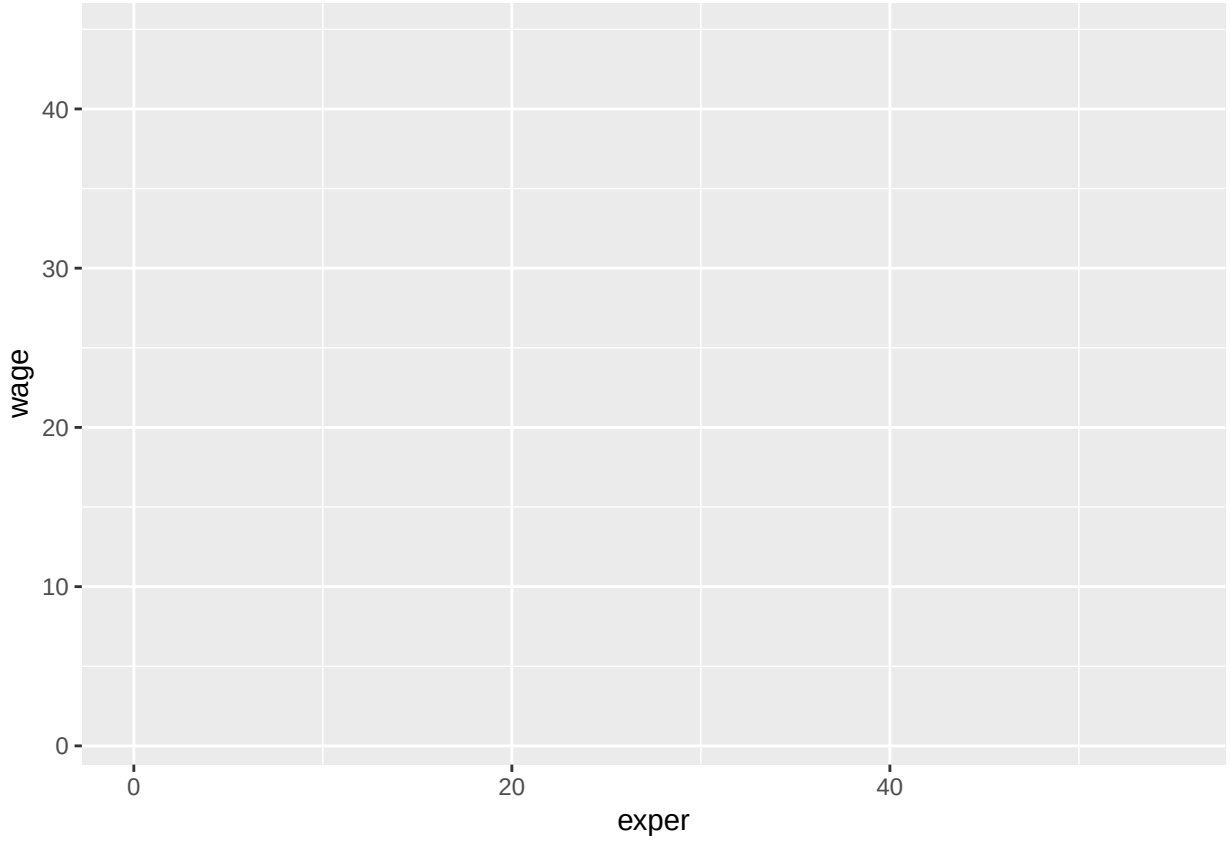


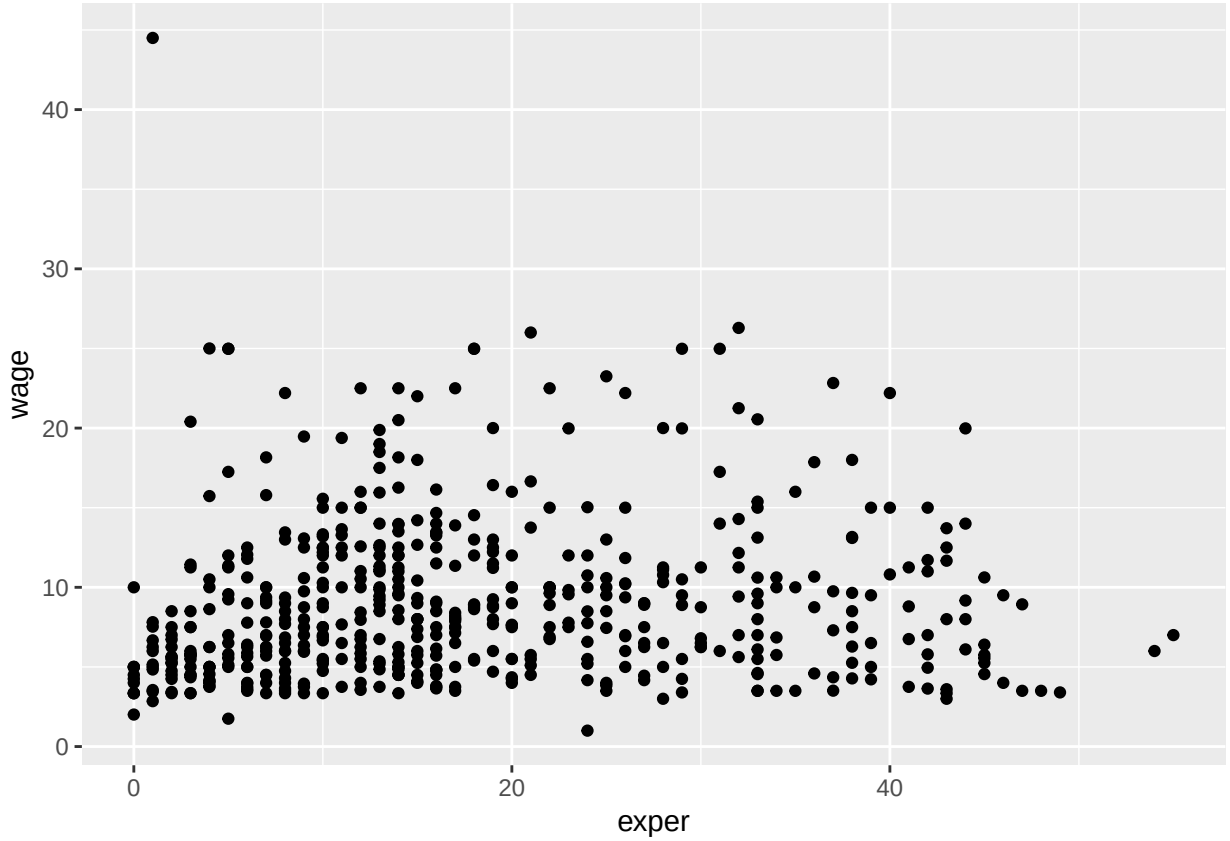
Figure 2.1: Map variables

2.1.2 geoms

Geoms are the geometric objects (points, lines, bars, etc.) that can be placed on a graph. They are added using functions that start with `geom_`. In this example, we'll add points using the `geom_point` function, creating a scatterplot.

In `ggplot2` graphs, functions are chained together using the `+` sign to build a final plot.

```
# add points
ggplot(data = CPS85,
       mapping = aes(x = exper, y = wage)) +
  geom_point()
```



The graph indicates that there is an outlier. One individual has a wage much higher than the rest. We'll delete this case before continuing.

```
# delete outlier
library(dplyr)
plotdata <- filter(CPS85, wage < 40)

# redraw scatterplot
ggplot(data = plotdata,
       mapping = aes(x = exper, y = wage)) +
  geom_point()
```

A number of parameters (options) can be specified in a `geom_` function. Options for the `geom_point` function include `color`, `size`, and `alpha`. These control the point color, size, and transparency, respectively. Trans-

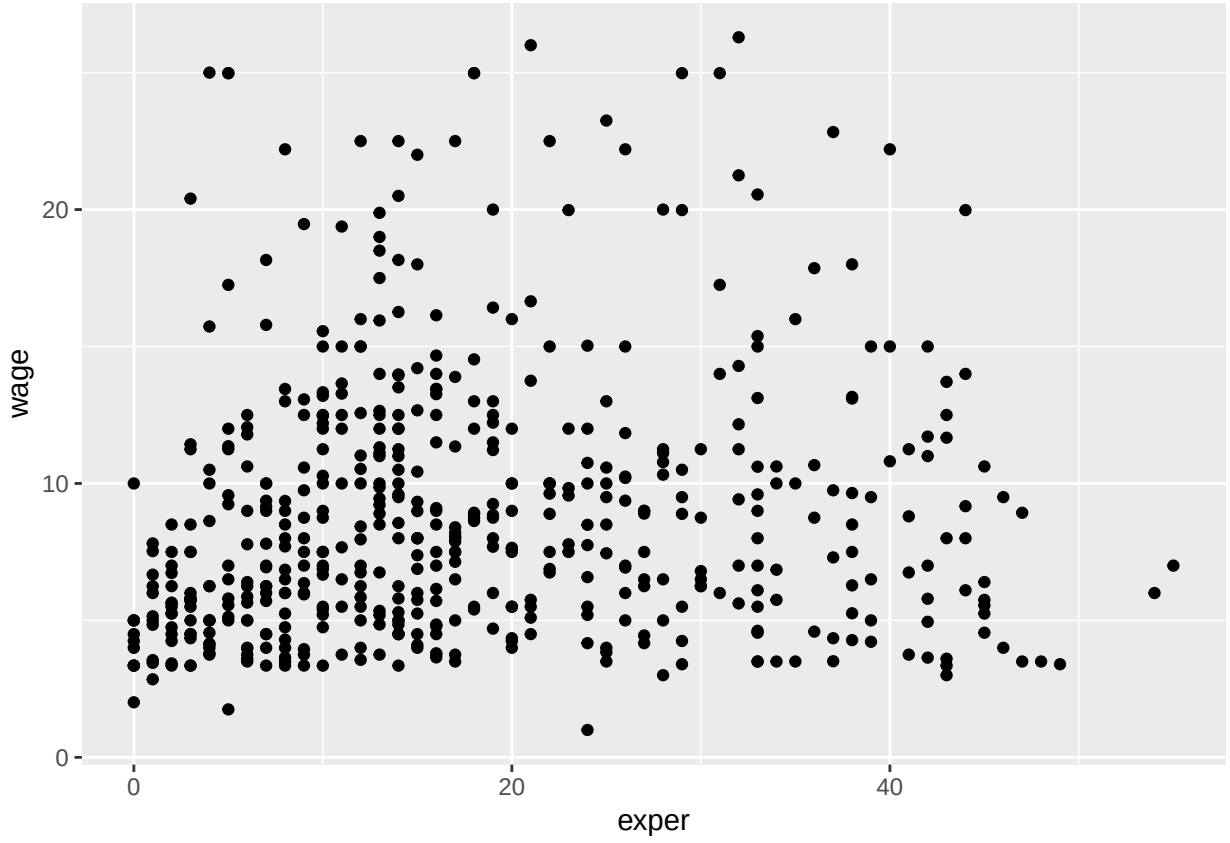


Figure 2.2: Remove outlier

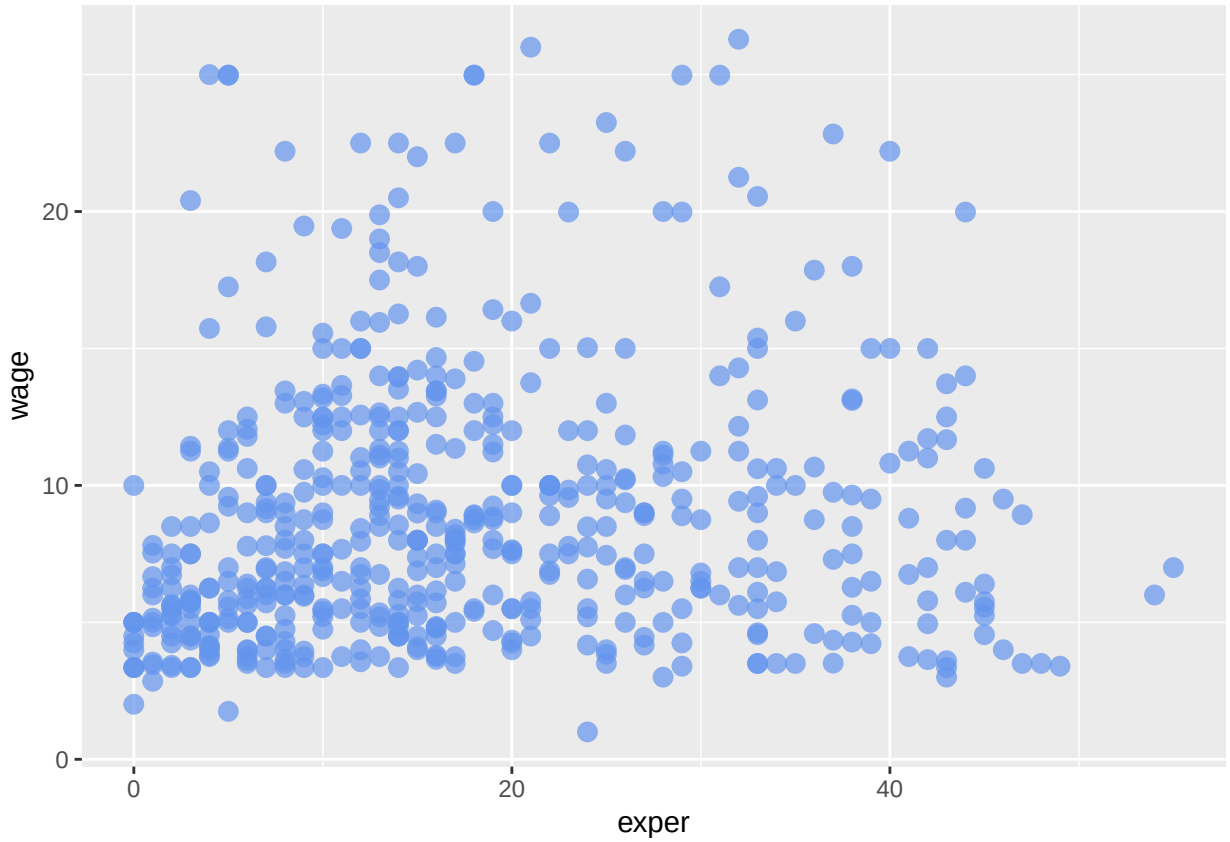


Figure 2.3: Modify point color, transparency, and size

parency ranges from 0 (completely transparent) to 1 (completely opaque). Adding a degree of transparency can help visualize overlapping points.

```
# make points blue, larger, and semi-transparent
ggplot(data = plotdata,
       mapping = aes(x = exper, y = wage)) +
  geom_point(color = "cornflowerblue",
             alpha = .7,
             size = 3)
```

Next, let's add a line of best fit. We can do this with the `geom_smooth` function. Options control the type of line (linear, quadratic, nonparametric), the thickness of the line, the line's color, and the presence or absence of a confidence interval. Here we request a linear regression (`method = lm`) line (where *lm* stands for linear model).

```
# add a line of best fit.
ggplot(data = plotdata,
       mapping = aes(x = exper, y = wage)) +
  geom_point(color = "cornflowerblue",
             alpha = .7,
             size = 3) +
  geom_smooth(method = "lm")
```

Wages appears to increase with experience.

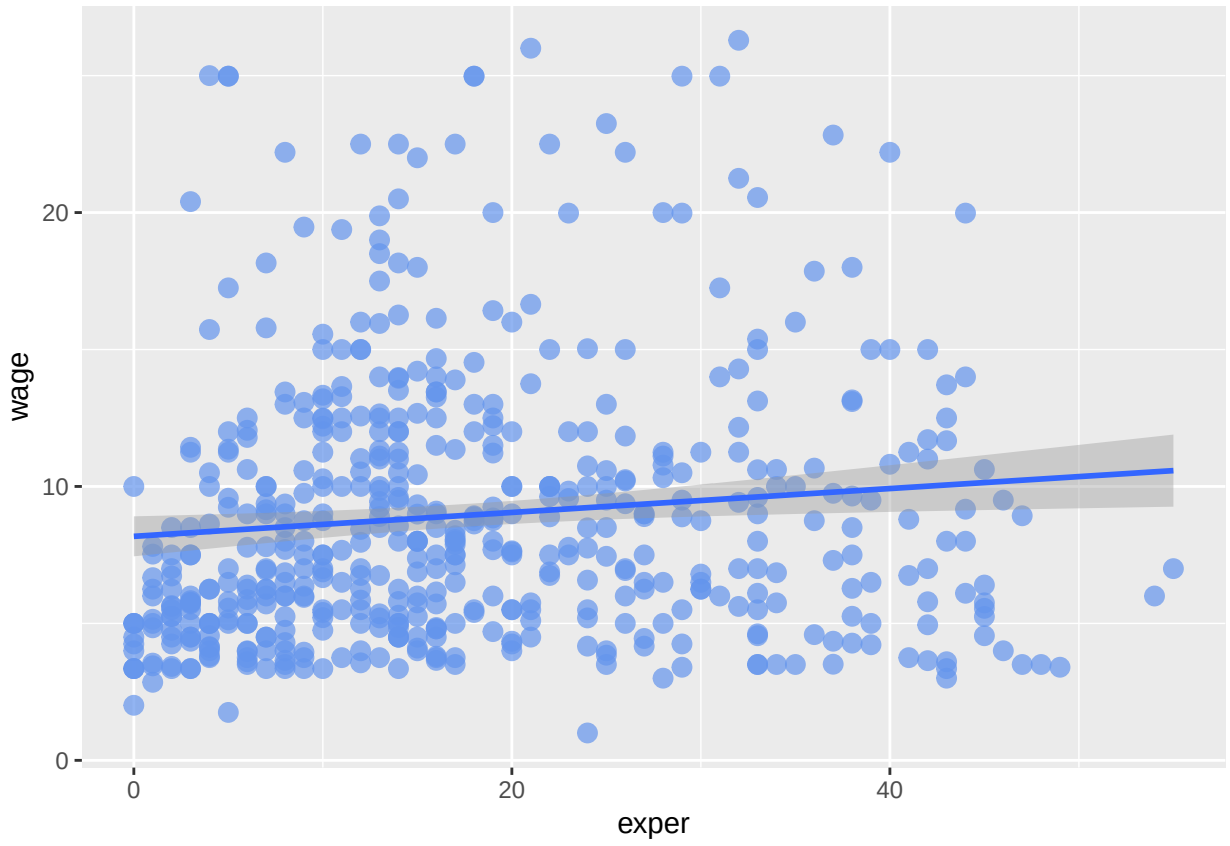


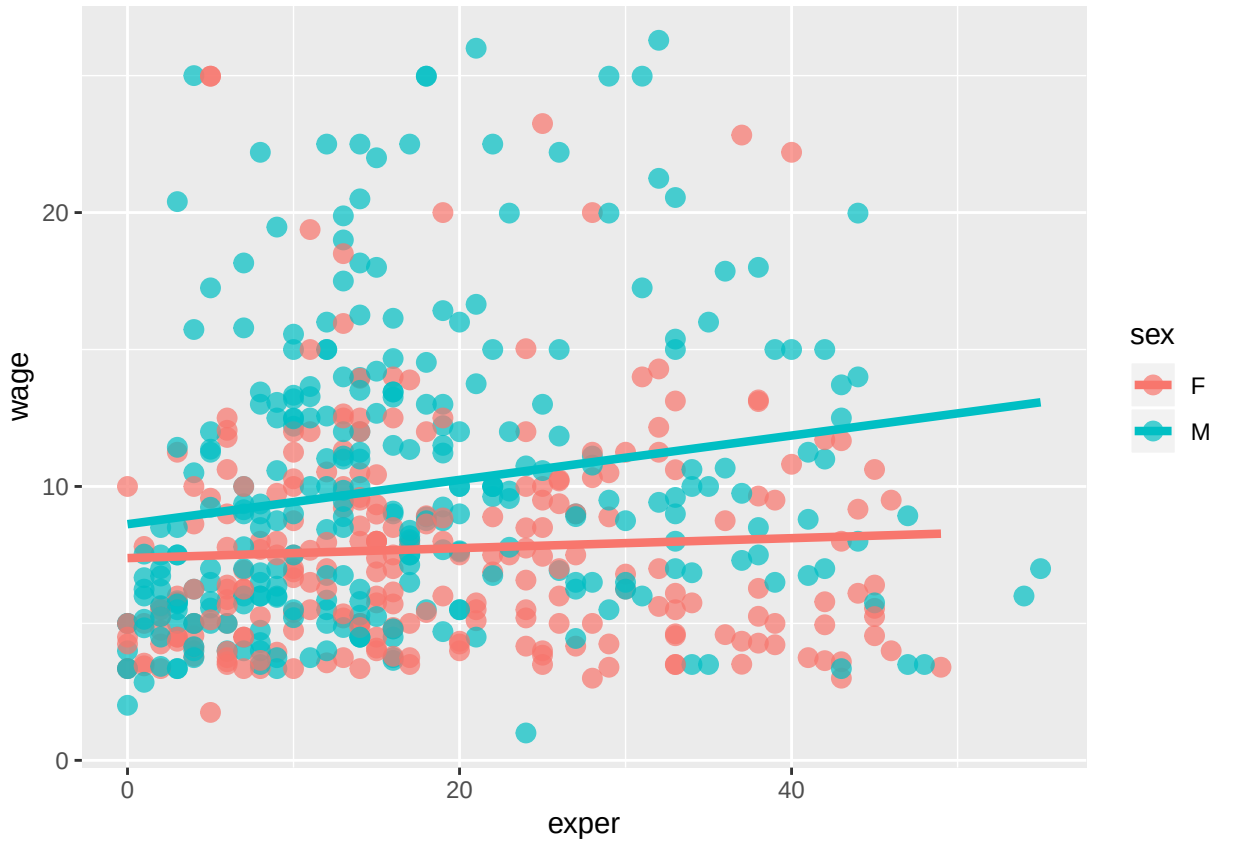
Figure 2.4: Add line of best fit

2.1.3 grouping

In addition to mapping variables to the x and y axes, variables can be mapped to the color, shape, size, transparency, and other visual characteristics of geometric objects. This allows groups of observations to be superimposed in a single graph.

Let's add sex to the plot and represent it by color.

```
# indicate sex using color
ggplot(data = plotdata,
       mapping = aes(x = exper,
                     y = wage,
                     color = sex)) +
  geom_point(alpha = .7,
             size = 3) +
  geom_smooth(method = "lm",
              se = FALSE,
              size = 1.5)
```



The `color = sex` option is placed in the `aes` function, because we are mapping a variable to an aesthetic. The `geom_smooth` option (`se = FALSE`) was added to suppresses the confidence intervals.

It appears that men tend to make more money than women. Additionally, there may be a stronger relationship between experience and wages for men than than for women.

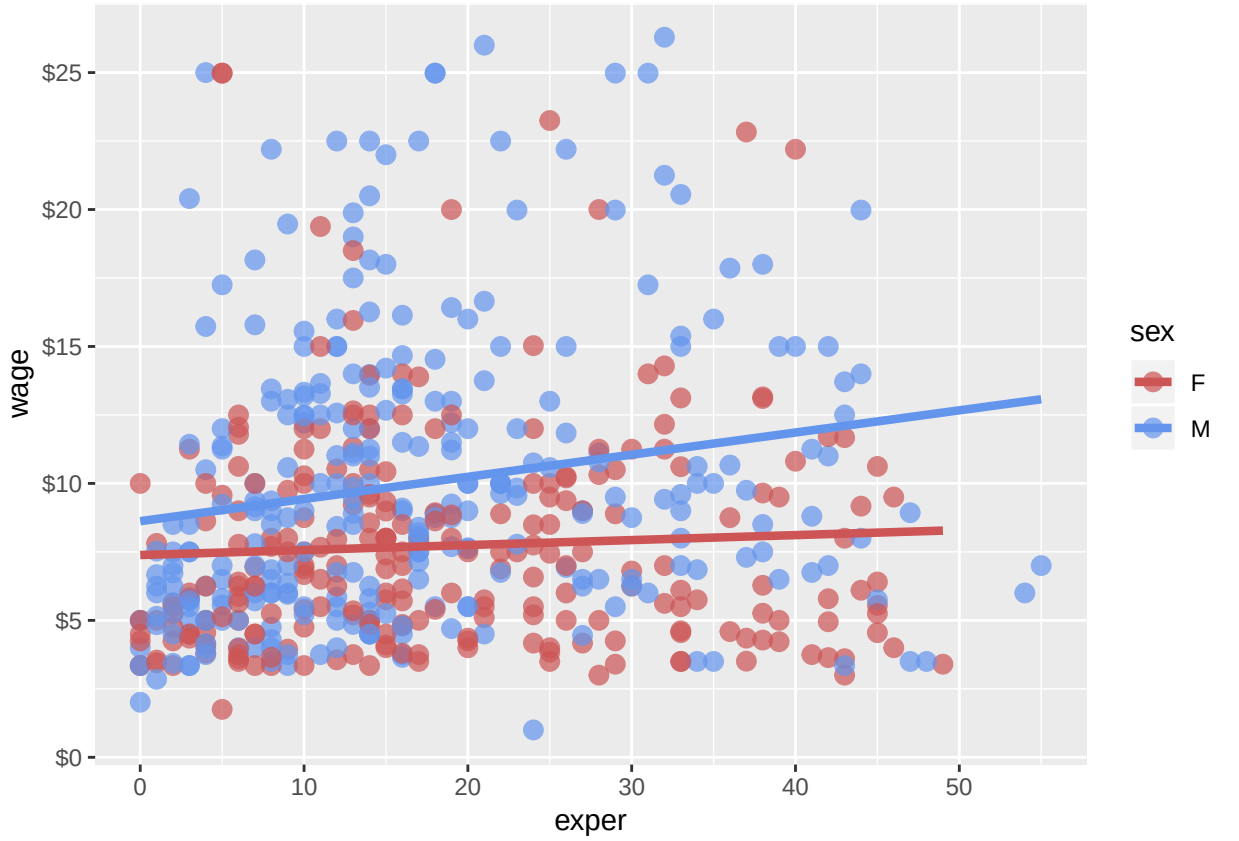


Figure 2.5: Change colors and axis labels

2.1.4 scales

Scales control how variables are mapped to the visual characteristics of the plot. Scale functions (which start with `scale_`) allow you to modify this mapping. In the next plot, we'll change the x and y axis scaling, and the colors employed.

```
# modify the x and y axes and specify the colors to be used
ggplot(data = plotdata,
       mapping = aes(x = exper,
                     y = wage,
                     color = sex)) +
  geom_point(alpha = .7,
             size = 3) +
  geom_smooth(method = "lm",
              se = FALSE,
              size = 1.5) +
  scale_x_continuous(breaks = seq(0, 60, 10)) +
  scale_y_continuous(breaks = seq(0, 30, 5),
                     label = scales::dollar) +
  scale_color_manual(values = c("indianred3",
                                "cornflowerblue"))
```

We're getting there. The numbers on the x and y axes are better, the y axis uses dollar notation, and the

colors are more attractive (IMHO).

Here is a question. Is the relationship between experience, wages and sex the same for each job sector? Let's repeat this graph once for each job sector in order to explore this.

2.1.5 facets

Facets reproduce a graph for each level a given variable (or combination of variables). Facets are created using functions that start with `facet_`. Here, facets will be defined by the eight levels of the *sector* variable.

```
# reproduce plot for each level of job sector
ggplot(data = plotdata,
       mapping = aes(x = exper,
                     y = wage,
                     color = sex)) +
  geom_point(alpha = .7) +
  geom_smooth(method = "lm",
              se = FALSE) +
  scale_x_continuous(breaks = seq(0, 60, 10)) +
  scale_y_continuous(breaks = seq(0, 30, 5),
                     label = scales::dollar) +
  scale_color_manual(values = c("indianred3",
                                "cornflowerblue")) +
  facet_wrap(~sector)
```

It appears that the differences between mean and women depend on the job sector under consideration.

2.1.6 labels

Graphs should be easy to interpret and informative labels are a key element in achieving this goal. The `labs` function provides customized labels for the axes and legends. Additionally, a custom title, subtitle, and caption can be added.

```
# add informative labels
ggplot(data = plotdata,
       mapping = aes(x = exper,
                     y = wage,
                     color = sex)) +
  geom_point(alpha = .7) +
  geom_smooth(method = "lm",
              se = FALSE) +
  scale_x_continuous(breaks = seq(0, 60, 10)) +
  scale_y_continuous(breaks = seq(0, 30, 5),
                     label = scales::dollar) +
  scale_color_manual(values = c("indianred3",
                                "cornflowerblue")) +
  facet_wrap(~sector) +
  labs(title = "Relationship between wages and experience",
       subtitle = "Current Population Survey",
       caption = "source: http://mosaic-web.org/",
       x = " Years of Experience",
       y = "Hourly Wage",
       color = "Gender")
```

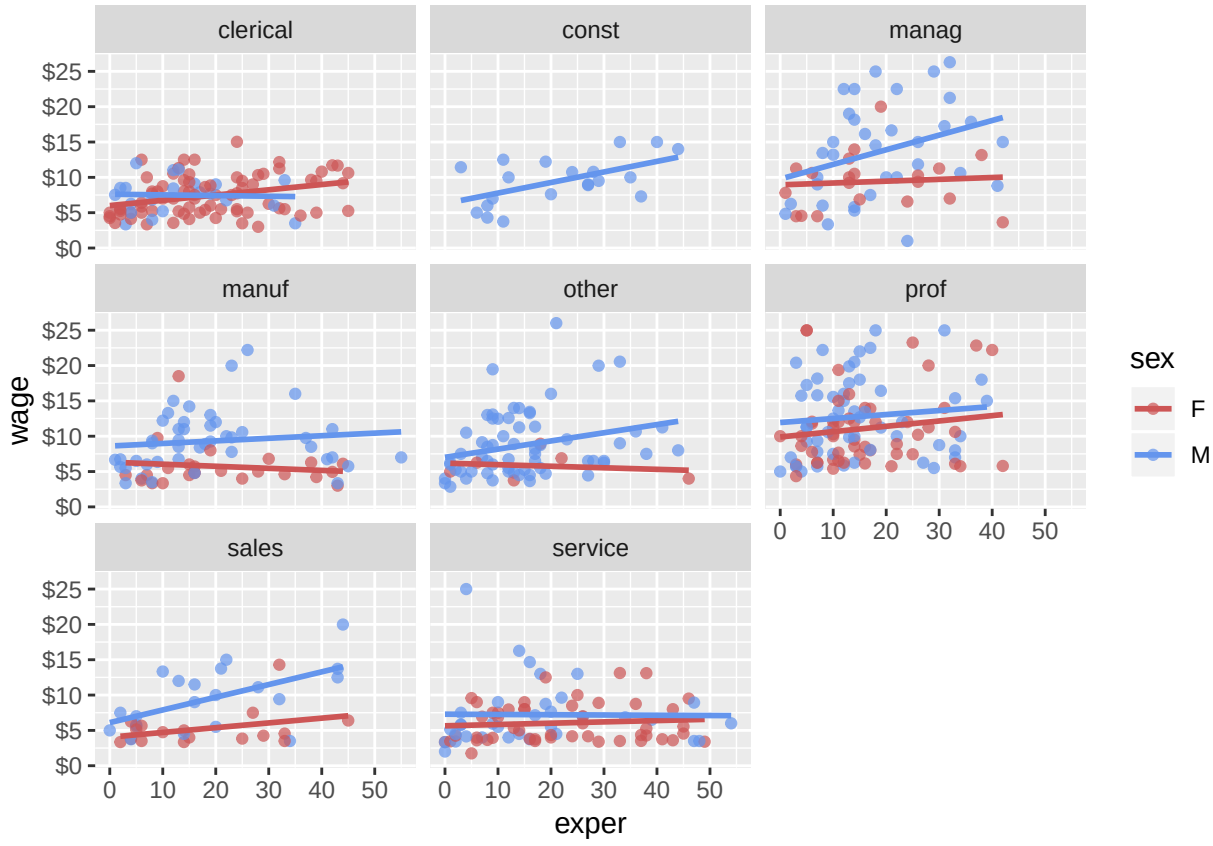
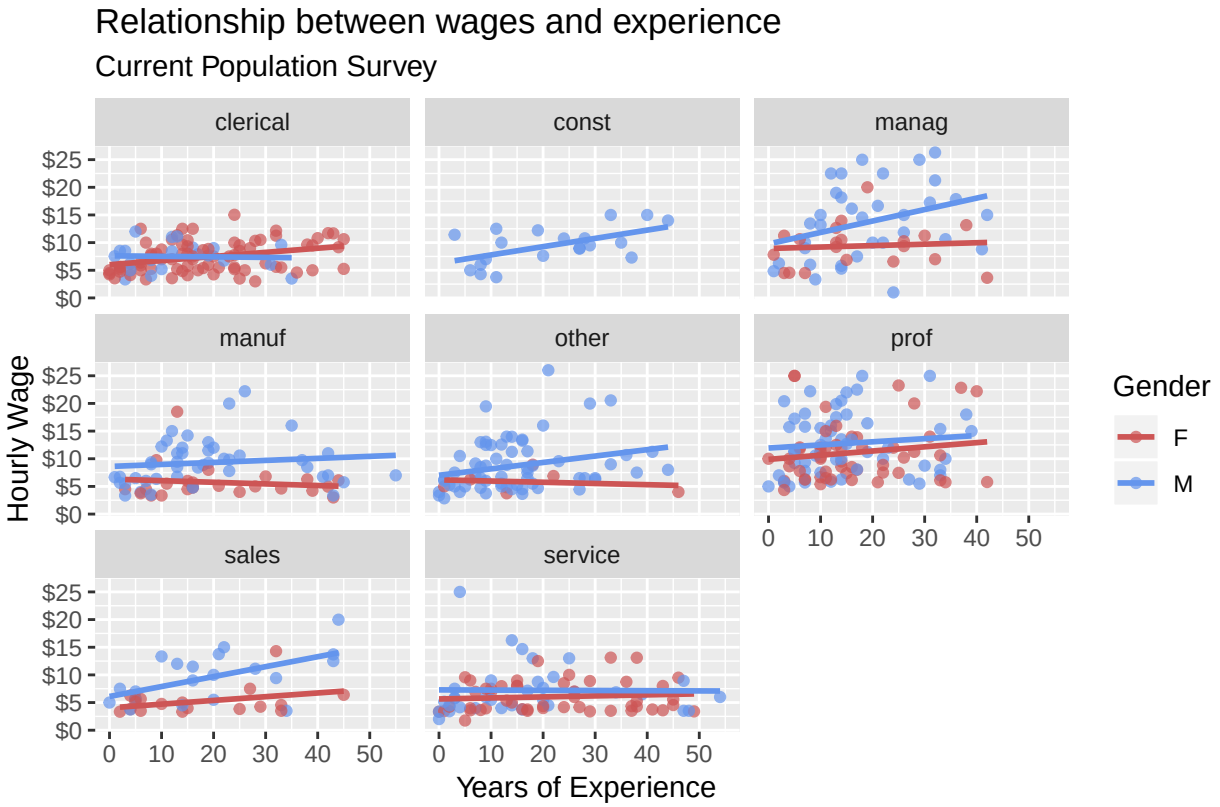


Figure 2.6: Add job sector, using faceting



Now a viewer doesn't need to guess what the labels *expr* and *wage* mean, or where the data come from.

2.1.7 themes

Finally, we can fine tune the appearance of the graph using themes. Theme functions (which start with `theme_`) control background colors, fonts, grid-lines, legend placement, and other non-data related features of the graph. Let's use a cleaner theme.

```
# use a minimalist theme
ggplot(data = plotdata,
       mapping = aes(x = exper,
                     y = wage,
                     color = sex)) +
  geom_point(alpha = .6) +
  geom_smooth(method = "lm",
              se = FALSE) +
  scale_x_continuous(breaks = seq(0, 60, 10)) +
  scale_y_continuous(breaks = seq(0, 30, 5),
                     label = scales::dollar) +
  scale_color_manual(values = c("indianred3",
                                "cornflowerblue")) +
  facet_wrap(~sector) +
  labs(title = "Relationship between wages and experience",
       subtitle = "Current Population Survey",
       caption = "source: http://mosaic-web.org/",
       x = " Years of Experience",
```

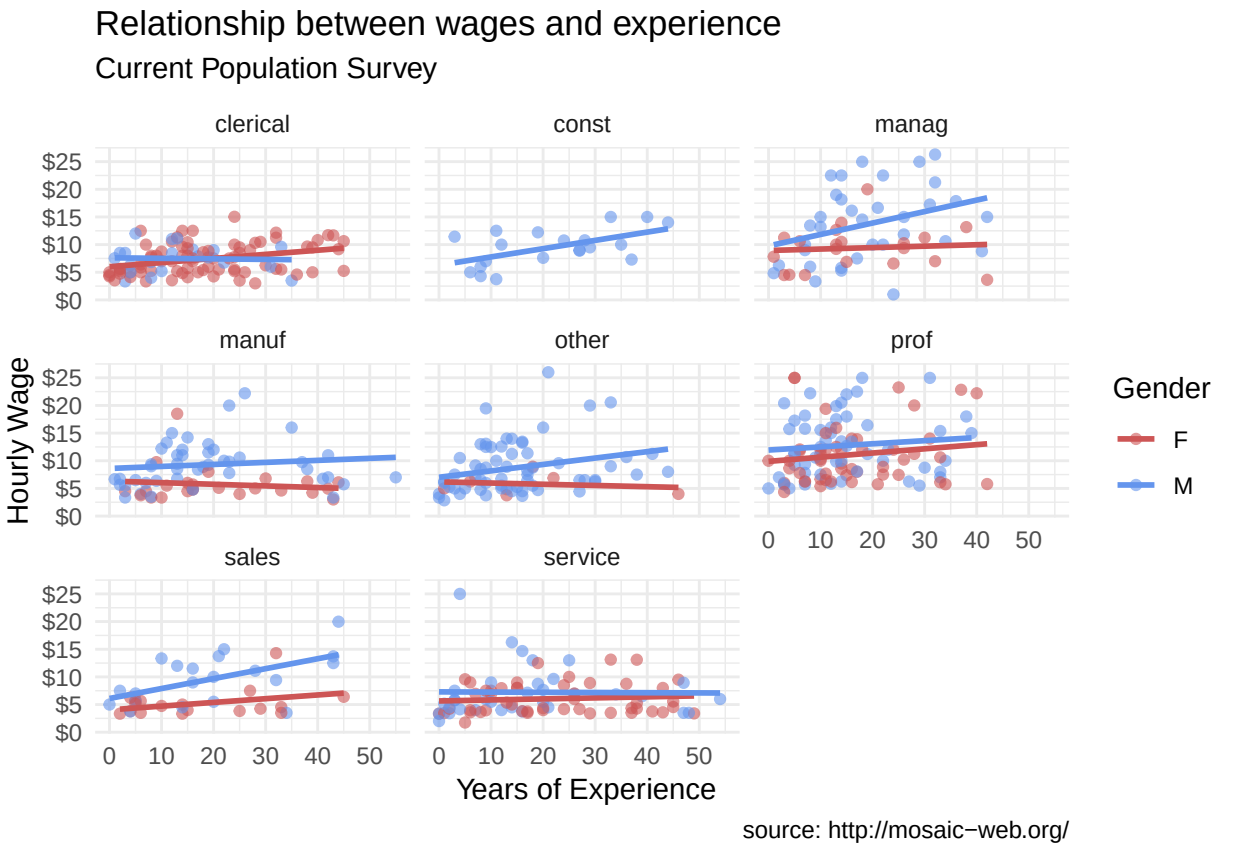


Figure 2.7: Use a simpler theme

```
       y = "Hourly Wage",
       color = "Gender") +
  theme_minimal()
```

Now we have something. It appears that men earn more than women in management, manufacturing, sales, and the "other" category. They are most similar in clerical, professional, and service positions. The data contain no women in the construction sector. For management positions, wages appear to be related to experience for men, but not for women (this may be the most interesting finding). This also appears to be true for sales.

Of course, these findings are tentative. They are based on a limited sample size and do not involve statistical testing to assess whether differences may be due to chance variation.

2.2 Placing the `data` and `mapping` options

Plots created with `ggplot2` always start with the `ggplot` function. In the examples above, the `data` and `mapping` options were placed in this function. In this case they apply to each `geom_` function that follows.

You can also place these options directly within a `geom`. In that case, they only apply only to that specific geom.

Consider the following graph.

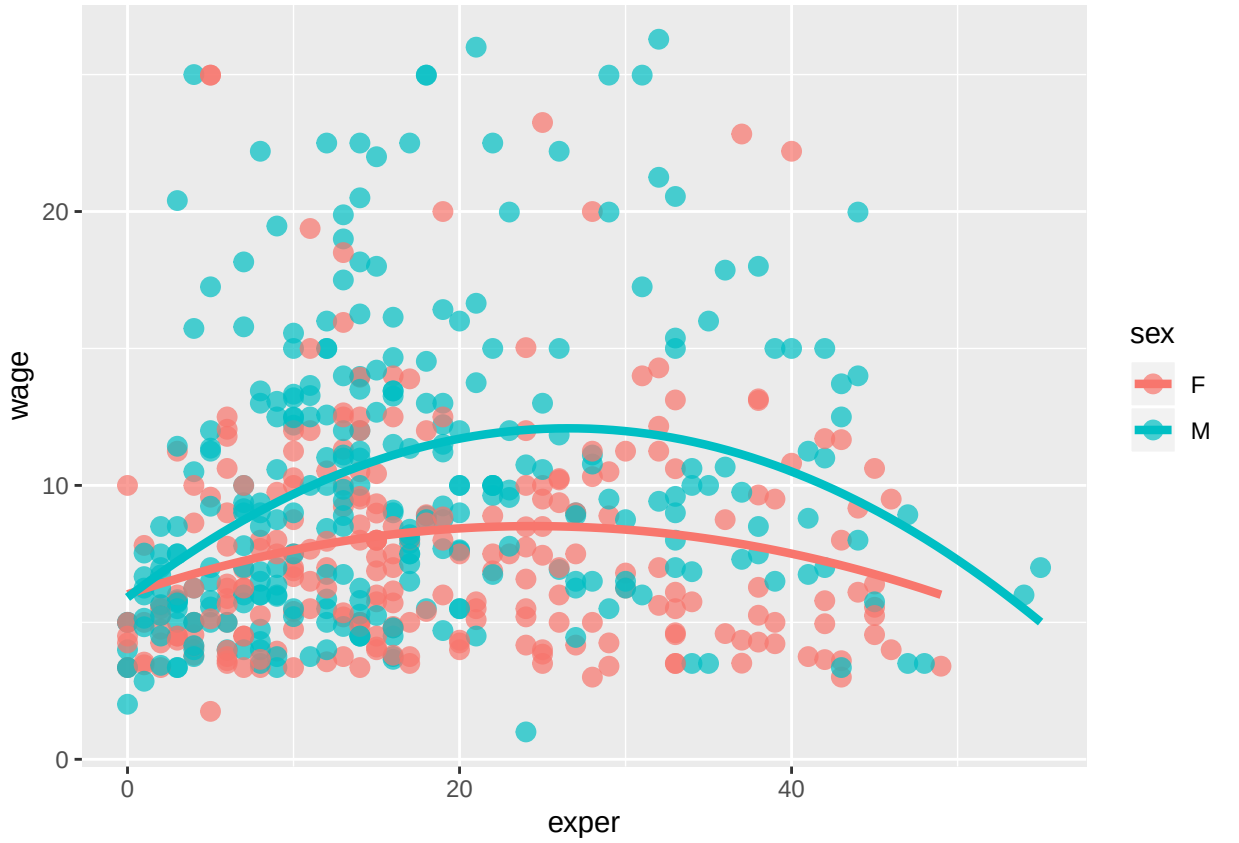


Figure 2.8: Color mapping in ggplot function

```
# placing color mapping in the ggplot function
ggplot(plotdata,
       aes(x = exper,
           y = wage,
           color = sex)) +
  geom_point(alpha = .7,
             size = 3) +
  geom_smooth(method = "lm",
              formula = y ~ poly(x,2),
              se = FALSE,
              size = 1.5)
```

Since the mapping of sex to color appears in the `ggplot` function, it applies to *both* `geom_point` and `geom_smooth`. The color of the point indicates the sex, and a separate colored trend line is produced for men and women. Compare this to

```
# placing color mapping in the geom_point function
ggplot(plotdata,
       aes(x = exper,
           y = wage)) +
  geom_point(aes(color = sex),
             alpha = .7,
             size = 3) +
```

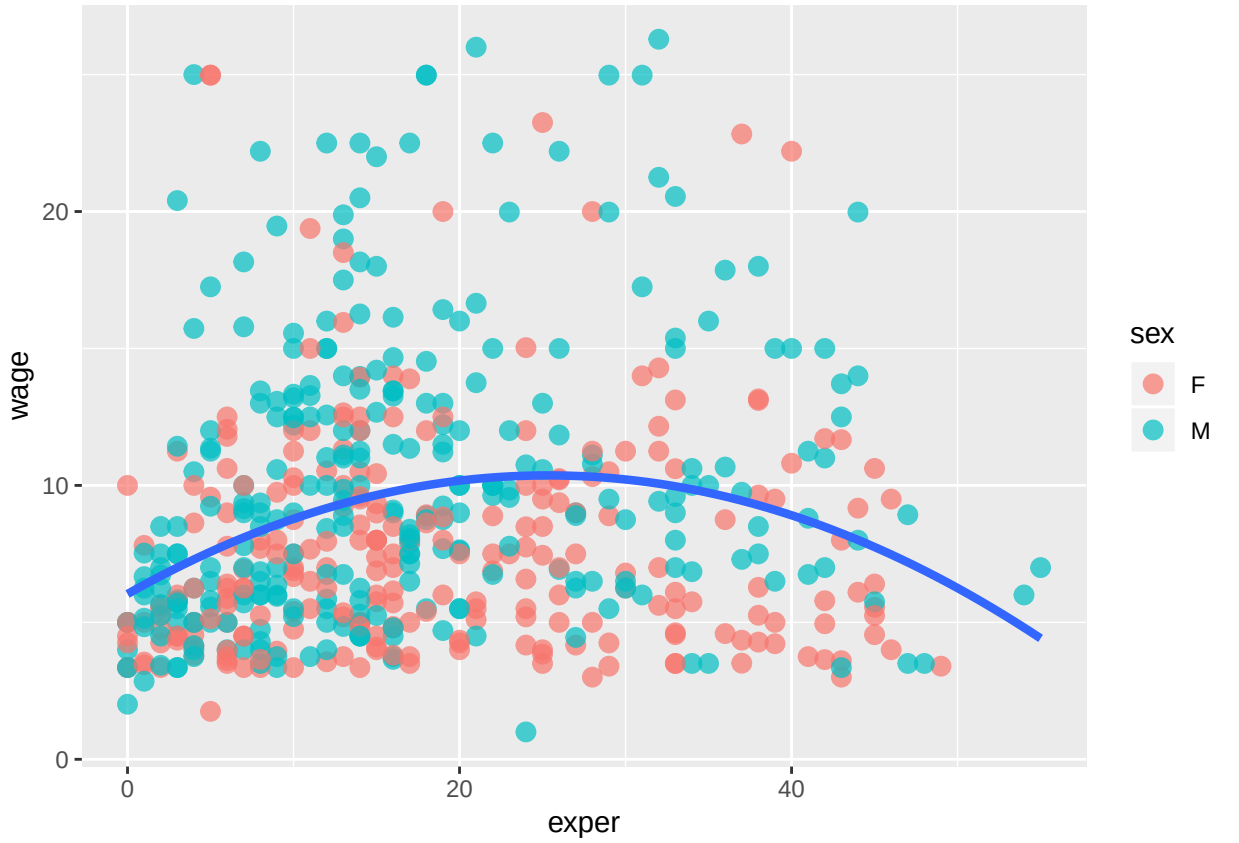


Figure 2.9: Color mapping in ggplot function

```
  geom_smooth(method = "lm",
              formula = y ~ poly(x,2),
              se = FALSE,
              size = 1.5)
```

Since the sex to color mapping only appears in the `geom_point` function, it is only used there. A single trend line is created for all observations.

Most of the examples in this book place the data and mapping options in the `ggplot` function. Additionally, the phrases *data=* and *mapping=* are omitted since the first option always refers to data and the second option always refers to mapping.

2.3 Graphs as objects

A `ggplot2` graph can be saved as a named R object (like a data frame), manipulated further, and then printed or saved to disk.

```
# prepare data
data(CPS85 , package = "mosaicData")
plotdata <- CPS85[CPS85$wage < 40,]
```

```
# create scatterplot and save it
myplot <- ggplot(data = plotdata,
                 aes(x = exper, y = wage)) +
            geom_point()

# print the graph
myplot

# make the points larger and blue
# then print the graph
myplot <- myplot + geom_point(size = 3, color = "blue")
myplot

# print the graph with a title and line of best fit
# but don't save those changes
myplot + geom_smooth(method = "lm") +
  labs(title = "Mildly interesting graph")

# print the graph with a black and white theme
# but don't save those changes
myplot + theme_bw()
```

This can be a real time saver (and help you avoid carpal tunnel syndrome). It is also handy when saving graphs programmatically.

Now it's time to try out other types of graphs.