Plotting with ggplot2

Andrew Zieffler

```
# Load the riverside.csv data
> city = read.csv("~/Google Drive/andy/epsy-8251/data/riverside.csv")


# Examine the data
> head(city)

  edu income senior gender party
1   8  26430      9      0     1
2   8  37449      7      1     0
3  10  34182     16      0     1
4  10  25479      1      0     2
5  10  47034     14      1     0
6  12  37656     14      1     0
```

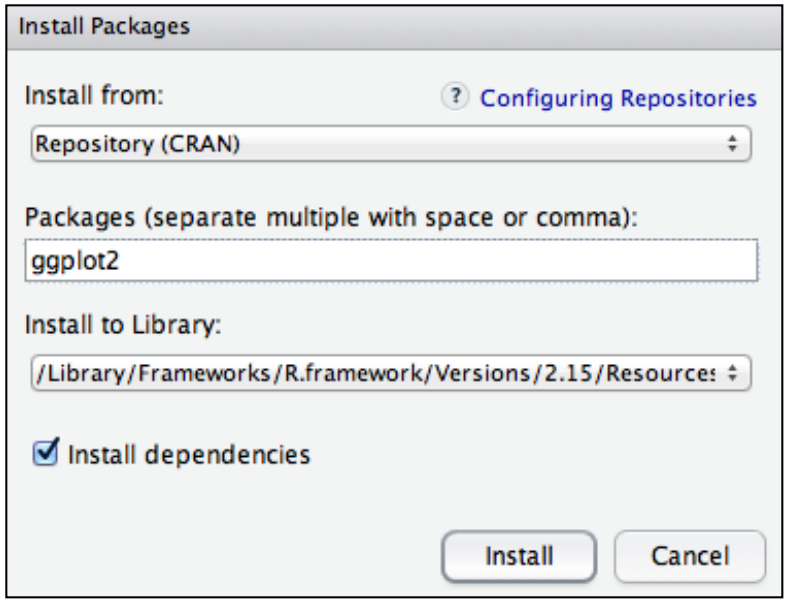
```
> summary(city)

      edu            income          senior          gender           party      
 Min.   : 8   Min.   :25479   Min.   : 1.00   Min.   :0.000   Min.   :0.000  
 1st Qu.:12   1st Qu.:44512   1st Qu.: 9.75   1st Qu.:0.000   1st Qu.:0.000  
 Median :16   Median :55830   Median :15.00   Median :0.000   Median :1.000  
 Mean   :16   Mean   :53742   Mean   :14.81   Mean   :0.438   Mean   :0.906  
 3rd Qu.:20   3rd Qu.:62717   3rd Qu.:20.25   3rd Qu.:1.000   3rd Qu.:1.000  
 Max.   :24   Max.   :82726   Max.   :27.00   Max.   :1.000   Max.   :2.000  
```

Install the **ggplot2** Package

Using the RStudio GUI...

- Click the **Packages** tab.
- Click **Install Packages**.
- Enter *ggplot2* in the text box.
- Click **Install**.



...or directly from the R command line...

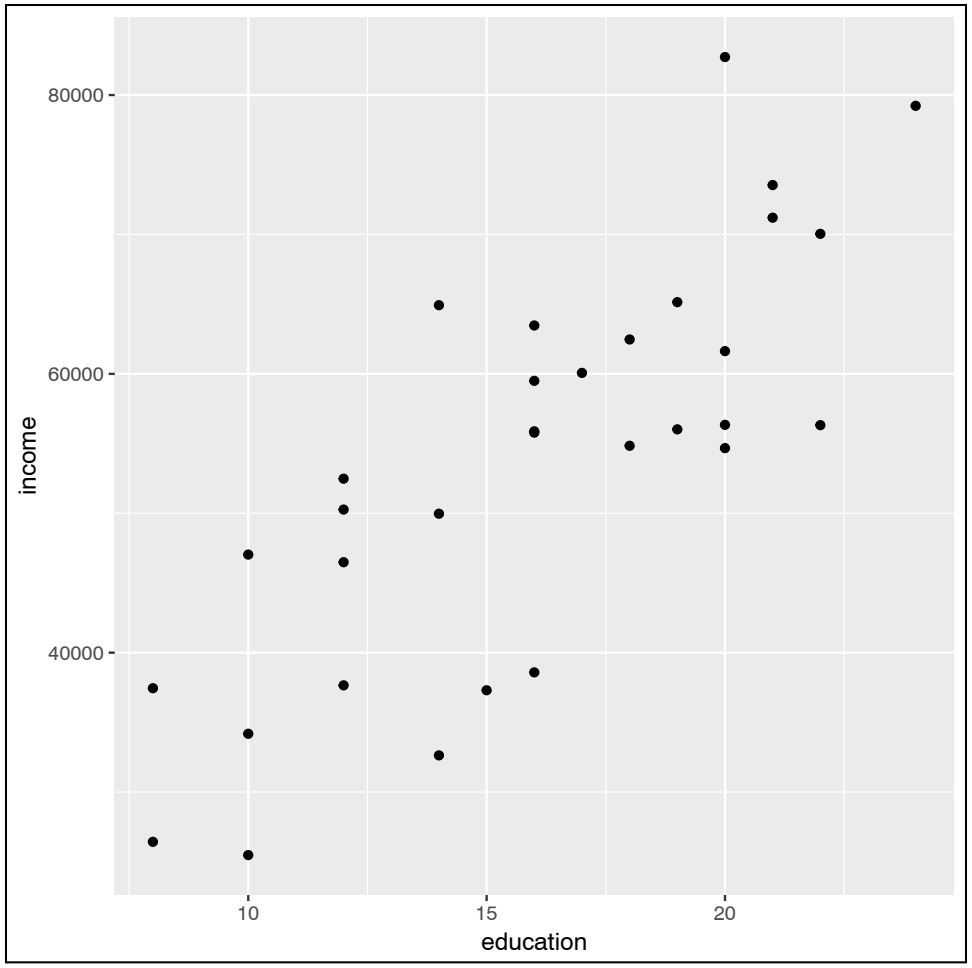
```
> install.packages("ggplot2", dependencies = TRUE)
```

The `library()` function loads the package so that the functions in the package are accessible. Libraries need to be loaded *every* R session.

```
# Load the ggplot2 library
> library(ggplot2)
```

Understanding the Basic Syntax

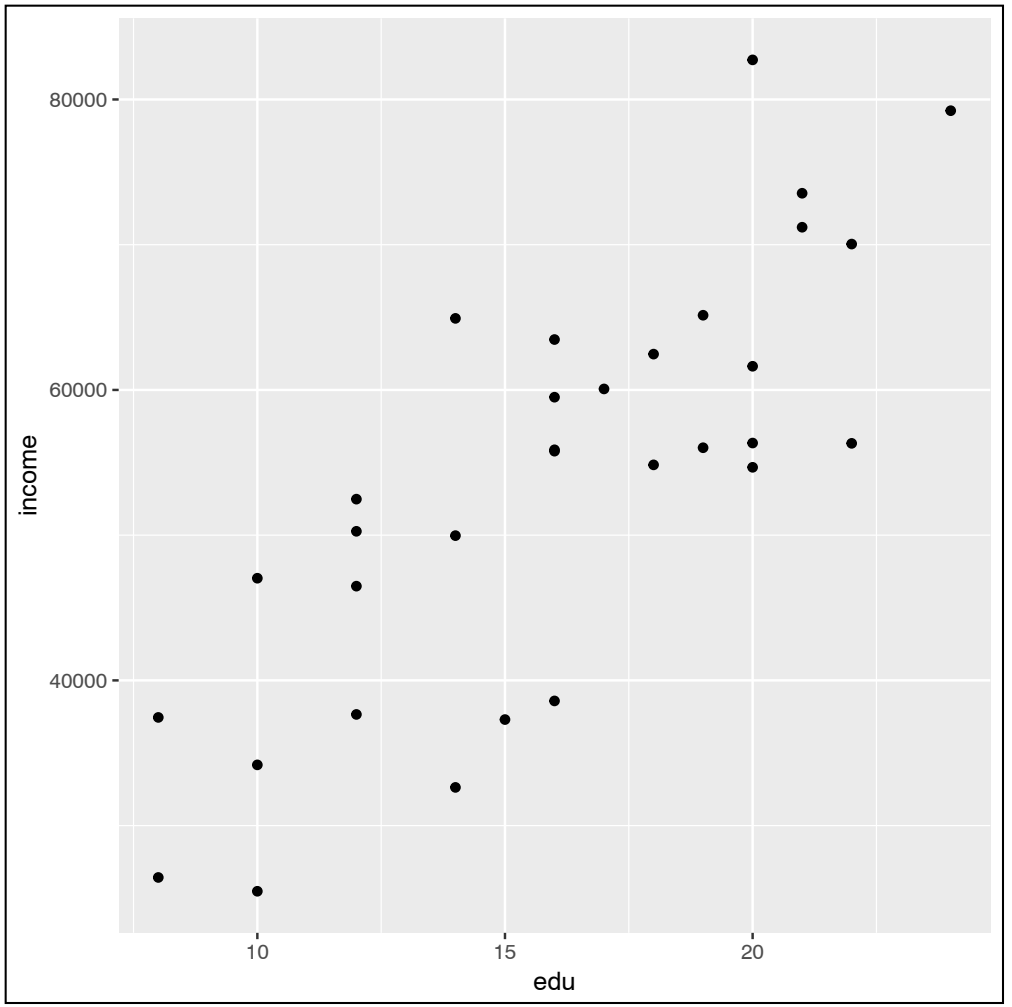
```
> ggplot(data = city, aes(x = education, y = income)) + geom_point()
```



Understanding the Basic Syntax

```
> ggplot(data = city, aes(x = education, y = income)) + geom_point()
```

Plots are built by layering graphical components. In the syntax, the layers are literally *summed* together to form the plot.



Global Layer

Aesthetic mappings given in the `ggplot()` layer are applied to all layers in the plot

```
> ggplot(data = city, aes(x = education, y = income)) +
```

The `data=` argument indicates the source data frame.

The `aes=` argument sets the aesthetic mapping(s).

The first layer is always `ggplot()`. It contains reference to the source data (data frame) and ***global*** **aesthetic mappings**.

The first layer only sets up the plot, it doesn't actually plot anything. In the subsequent layers, we add geometric objects (e.g., points, boxplots).

Aesthetic mappings describe how **variables in the data are mapped to visual properties** (aesthetics) of geoms. They are used to define position (x-dimension y-dimension), size, color, fill, groupings, etc.

- Aesthetics can be set **globally**—in `ggplot()` layer—or **locally** (only used in a specific geom layer)
- Each aesthetic can be **variable** or **fixed**
  - If the aesthetic is variable it needs to be specified in the `aes()` function
  - If the aesthetic is fixed it should be specified outside the `aes()` function

Adding Geometric Objects

```
> ggplot(data = city, aes(x = education, y = income)) + geom_point()
```

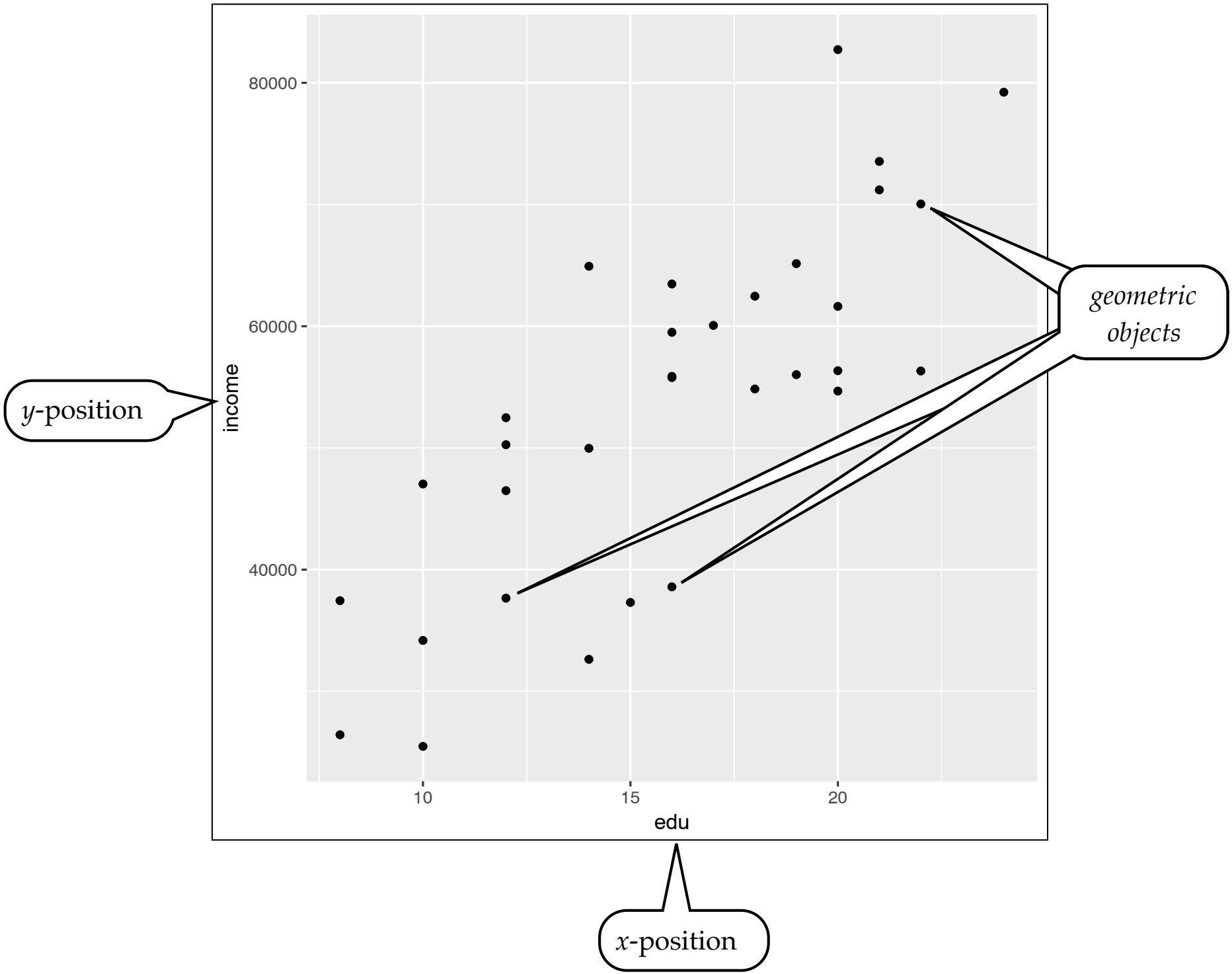
The `+` adds another layer.

The `geom_point()` function adds the **geometric object** of points using the global data and aesthetic mapping.

Geometric objects, or *geoms*, are features that are actually drawn on plot (e.g., lines, points). They are specified using the prefix `geom_` and a suffix that names the feature to be plotted.

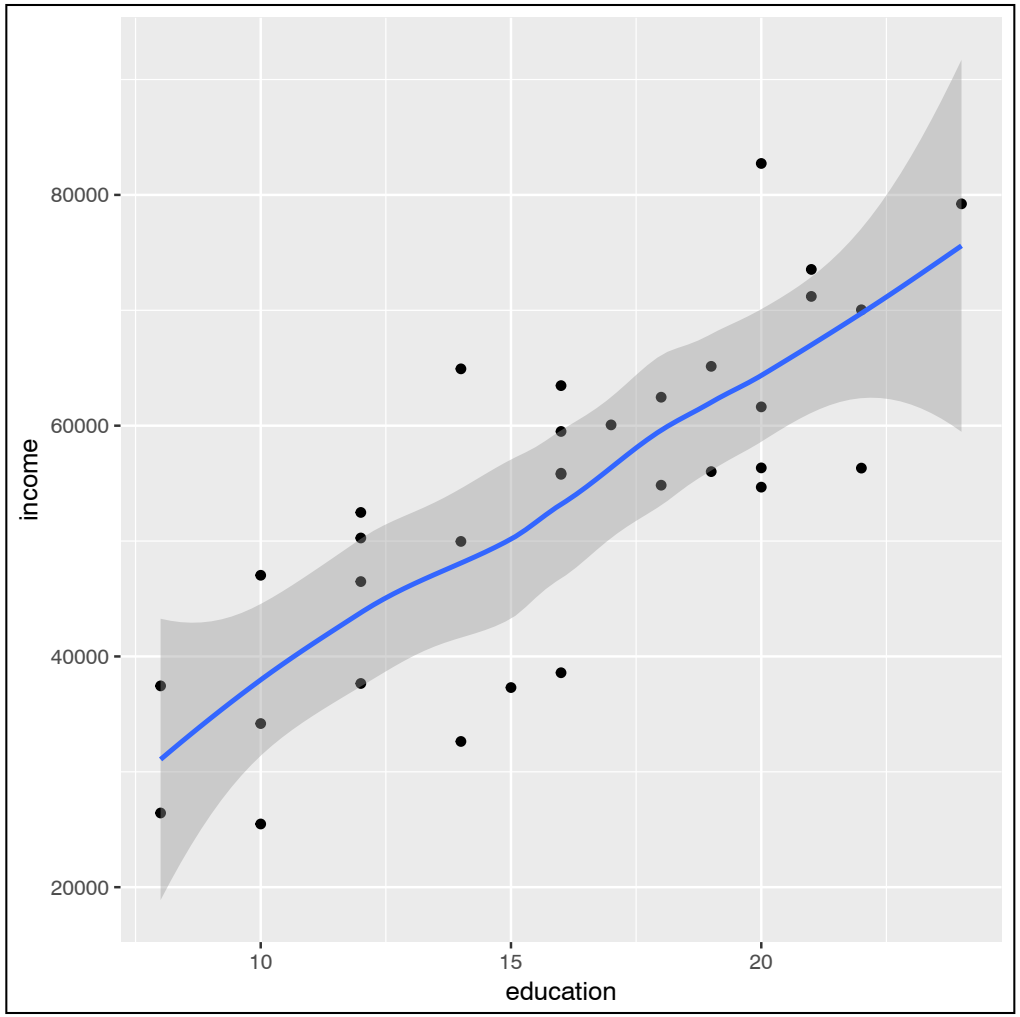
- **Points** specified with `geom_point()`
- **Jittered points** specified with `geom_jitter()`
- **Lines** specified with `geom_line()`
- **Boxplots** specified with `geom_boxplot()`

The geometric objects (e.g., points, boxplots). are plotted based on the aesthetics specified in the ggplot layer. For example, the syntax above draws points at the ordered pairs of employees' education (x-position) and incomes (y-position).


80000
60000
40000
income
10
15
20
edu
y-position
geometric objects
x-position

When layers are added they are "stacked" on top of previous layers. Imagine drawings on separate transparencies, and then those transparencies are stacked.

```
> ggplot(data = city, aes(x = education, y = income)) + geom_point() + geom_smooth()
```



The + adds another layer.

The `geom_smooth()` function adds the geometric object of a loess smoother using the global data and aesthetic mapping.

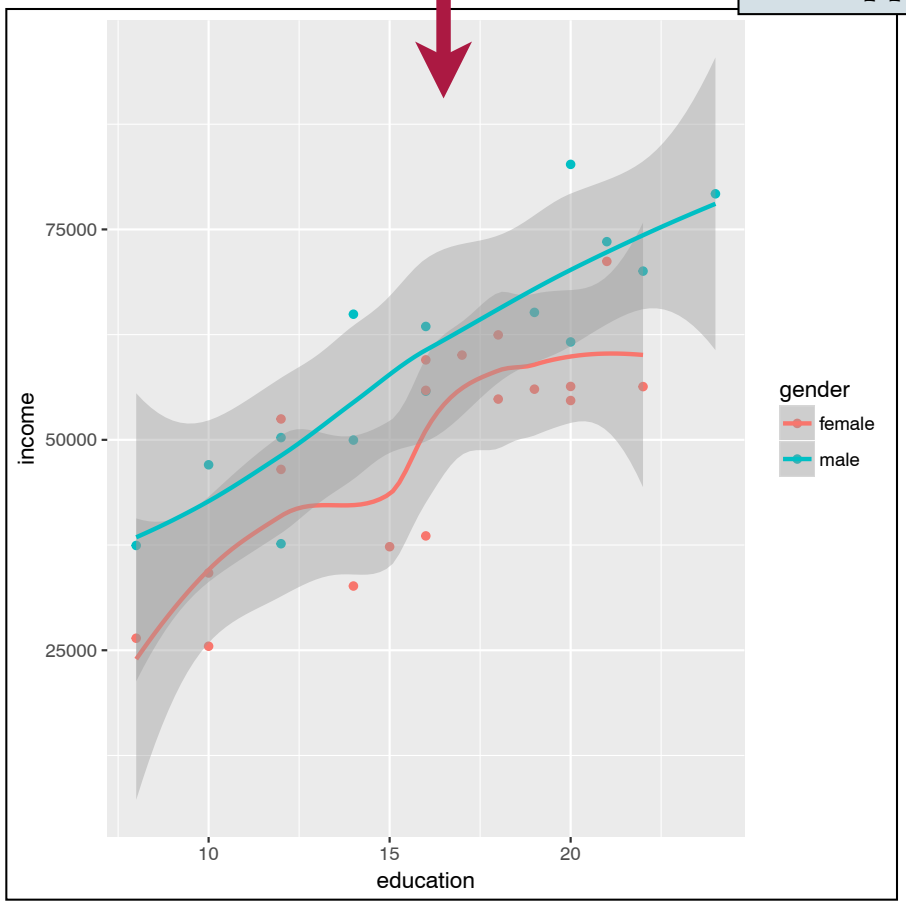
As we add more layers, it is better to use multiple lines for the syntax. Generally we put one layer on each line. The + sign needs to be at the end of the line (not at the beginning). To run the syntax, highlight ALL layers in the plot and click Run.

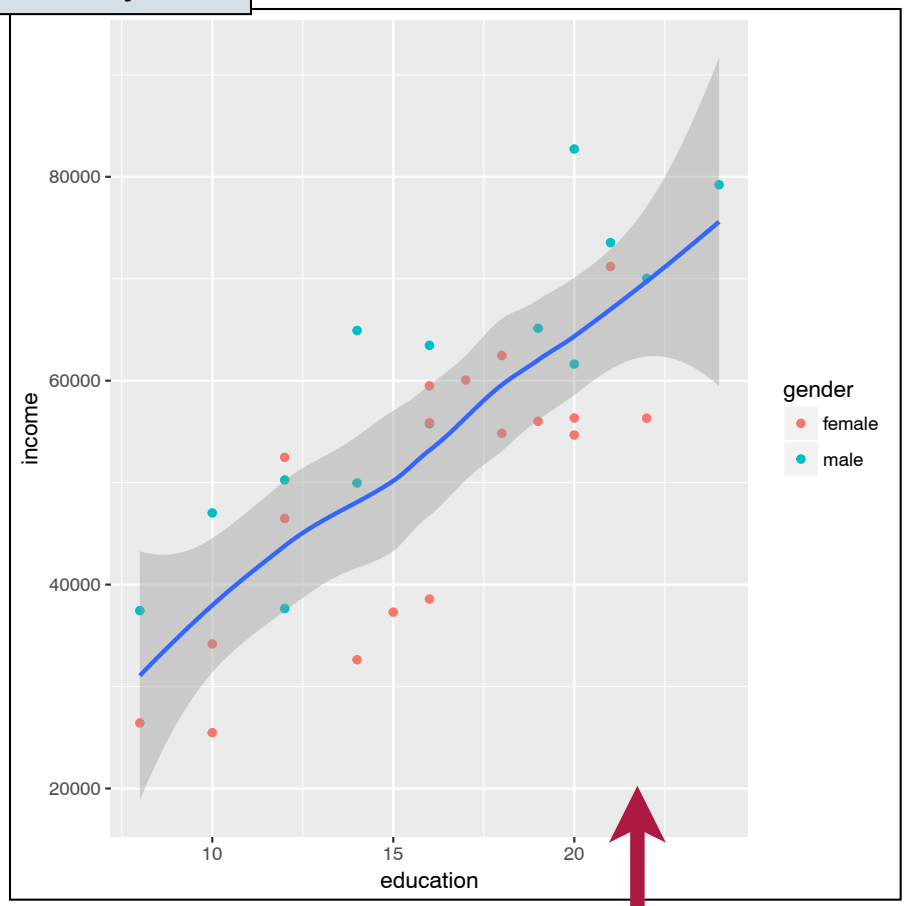
```
> ggplot(data = city, aes(x = education, y = income)) +
    geom_point() +
    geom_smooth()
```

The + is included at the end of each layer.

```
> ggplot(data = city, aes(x = education, y = income, color = gender)) +
    geom_point() +
    geom_smooth()
```

Global aesthetic mappings are applied to *all* layers.






```
> ggplot(data = city, aes(x = education, y = income)) +
    geom_point(aes(color = gender)) +
    geom_smooth()
```

Local aesthetic mappings (in a particular layer) are only applied to that layer.

Fixed vs Variable Aesthetics

```
> ggplot(data = city, aes(x = education, y = income, color = gender)) +
    geom_point() +
    geom_smooth(color = "yellow", fill = "darkblue")
```

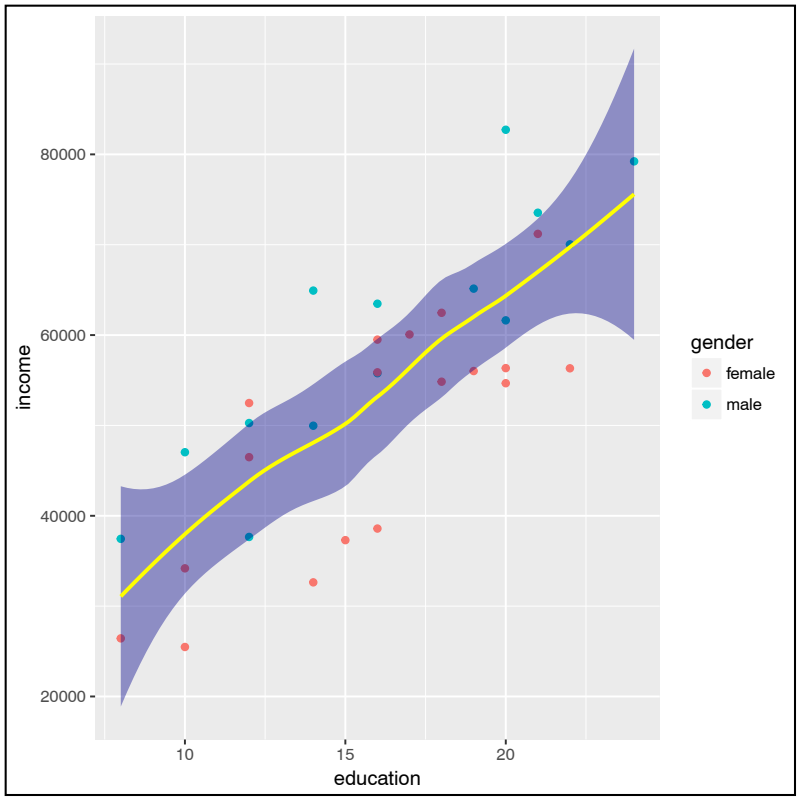
The `color=` argument sets the color for this layer (in this case the line).

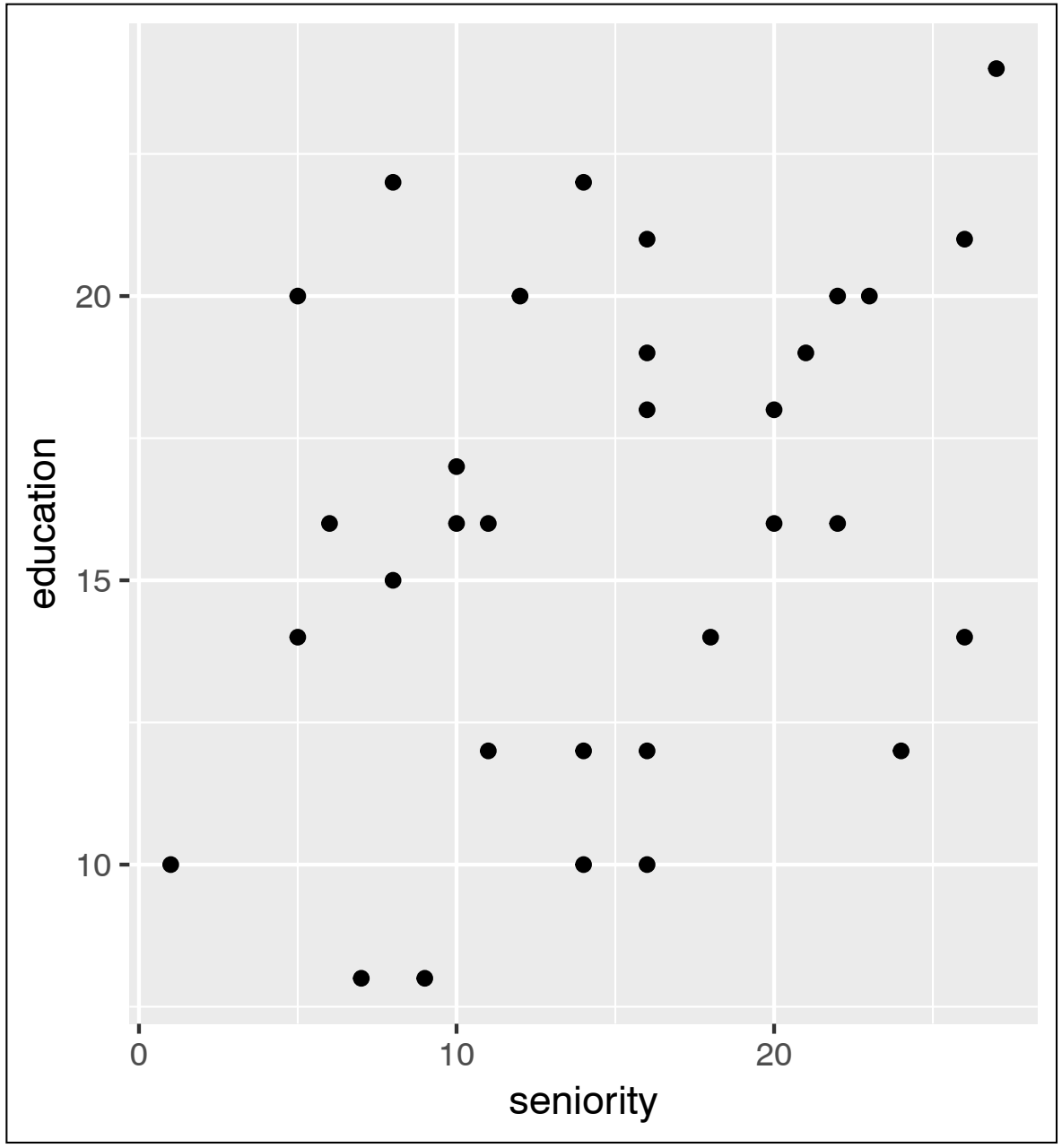
The `fill=` argument sets the fill color for this layer.

Notice the quotation marks...color names are character strings.

Aesthetic mappings that are fixed to a particular value (do not vary) do **not** need to be enclosed in the `aes()` function.

Note also that the local aesthetics override the global aesthetics





Your Turn

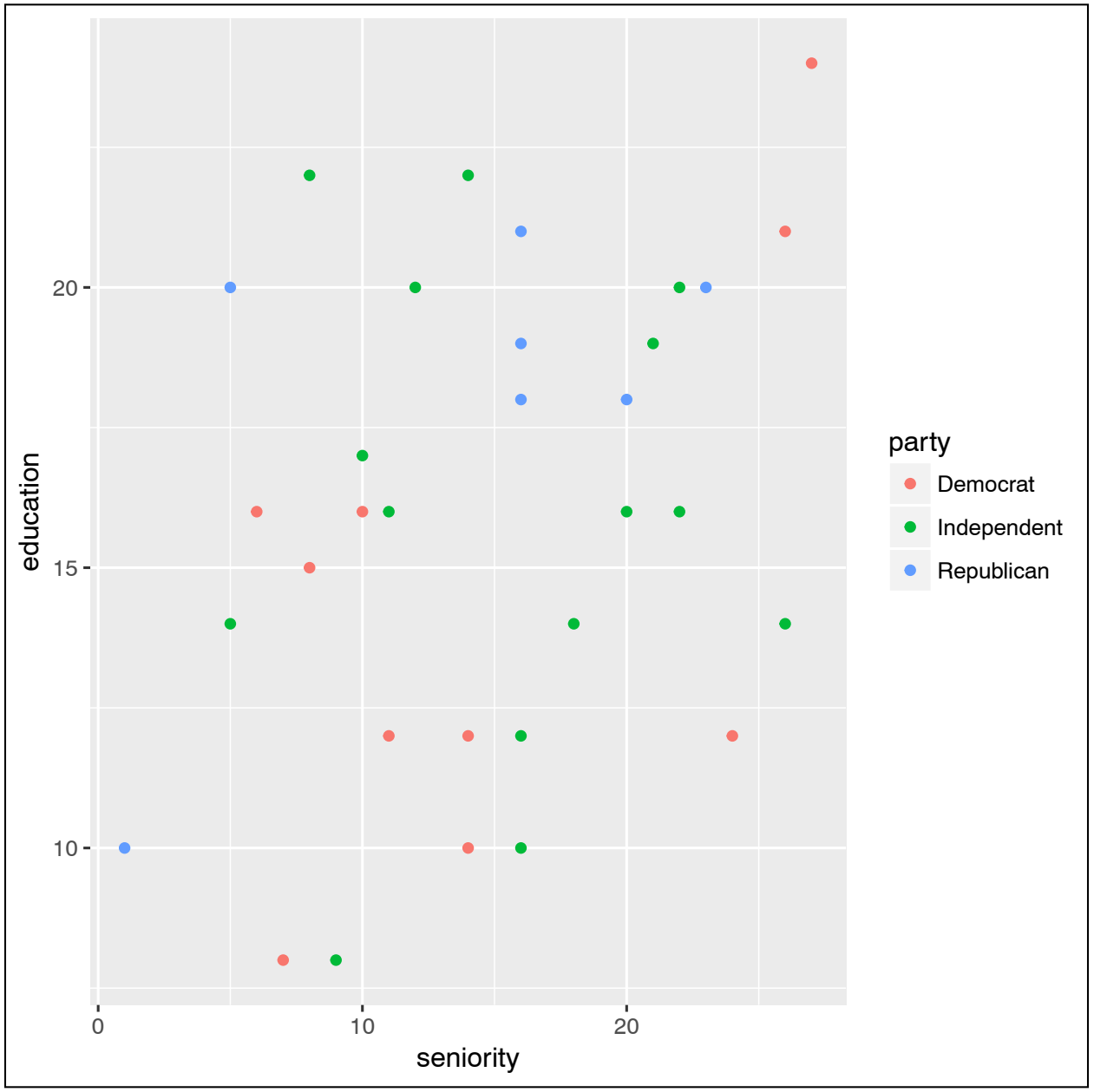
Write the syntax to create this scatterplot.

How would we color the points by political party?

```
# Create the plot
> ggplot(data = city, aes(x = seniority, y = education)) +
    geom_point()
```

```
# Color the points by department (Option 1)
> ggplot(data = city, aes(x = seniority, y = education, color = party)) +
    geom_point()
```

```
# Color the points by department (Option 2)
> ggplot(data = city, aes(x = seniority, y = education)) +
    geom_point(aes(color = party))
```



When we use non-positional aesthetics (e.g., color) ggplot will add a legend to our plot.

Point Aesthetics

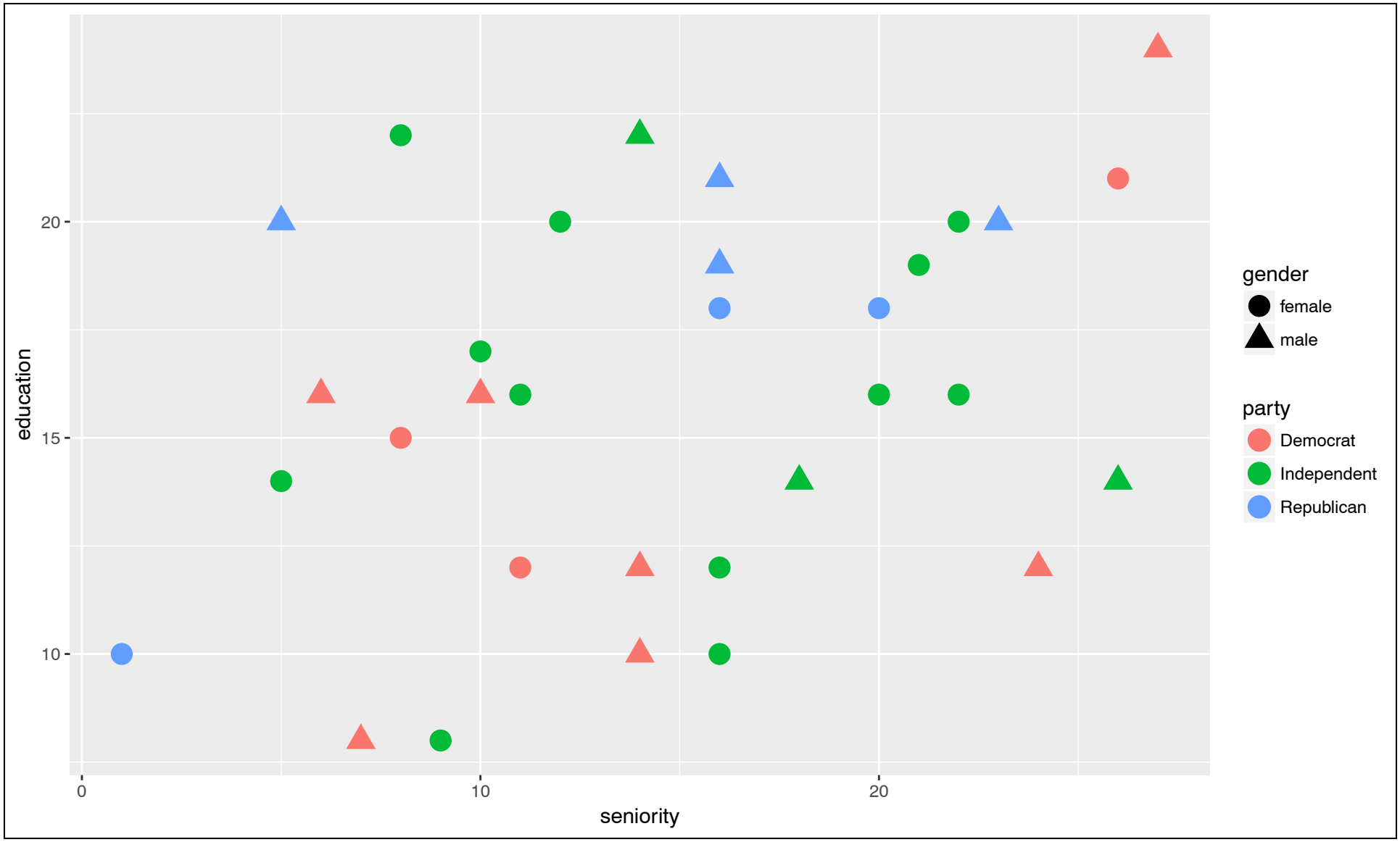
Two other useful aesthetics for points are `pch=` and `size=` for plotting character and point size, respectively.

```
> ggplot(data = city, aes(x = seniority, y = education)) +
    geom_point(aes(color = party, pch = gender), size = 5)
```

The `pch=` argument sets the plotting character.

The `size=` argument sets the point size. (The default size is 4.)

Describe the resulting plot based on the syntax above.



Note: EVERY non-positional aesthetics gets added to the legend.

Faceting

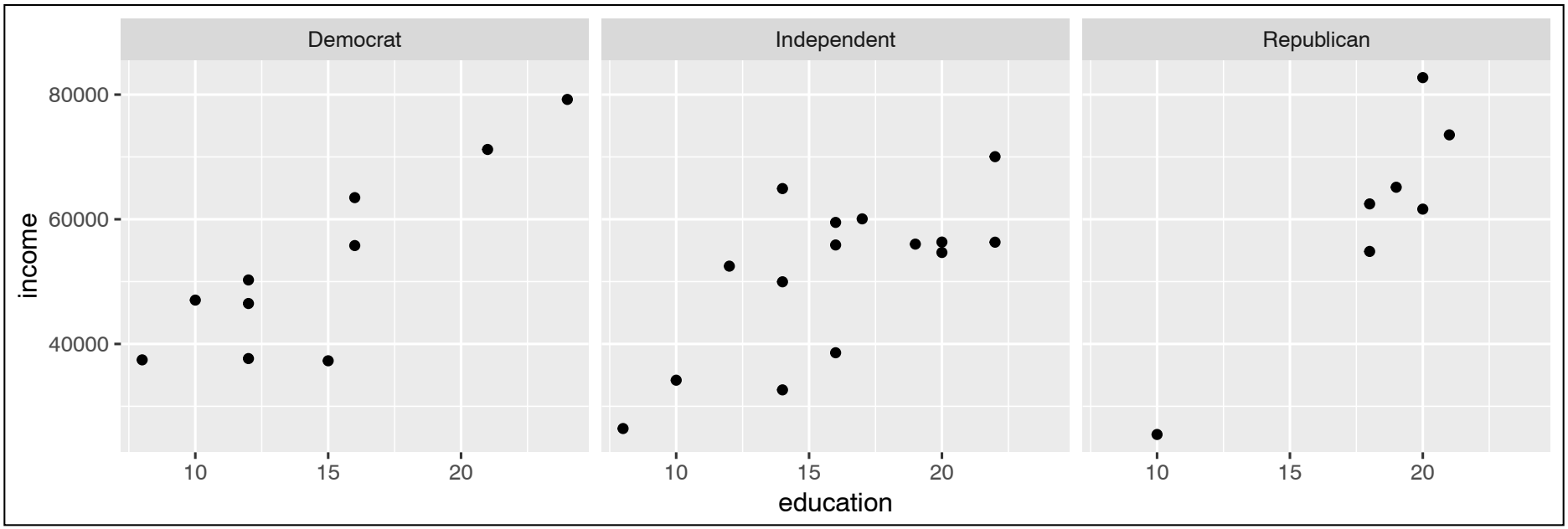
Faceting creates a separate plot for each subgroup declared

- `facet_wrap()` displays the plots conditioned on a **single predictor**
- `facet_grid()` displays the plots conditioned on **multiple predictors**

```
> ggplot(data = city, aes(x = education, y = income)) +
    geom_point() +
    facet_wrap(~ party)
```

~ sets the predictor for conditioning

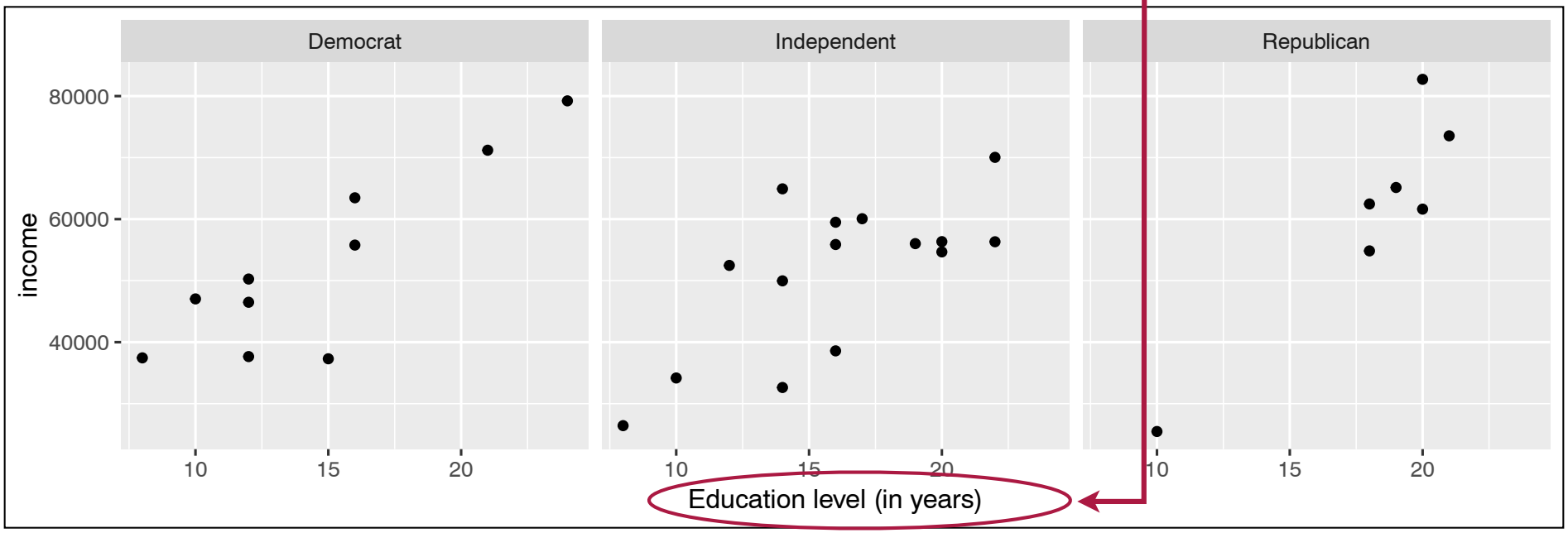
The scatterplots show the relationship between education level and income separated by political party.



Changing the Axis Label

```
> ggplot(data = city, aes(x = education, y = income)) +
    geom_point() +
    facet_wrap(~ party) +
    xlab("Education level (in years)")
```

`xlab()` can be used to change the label on the x-axis, and `ylab()` is used to change the label on the y-axis.



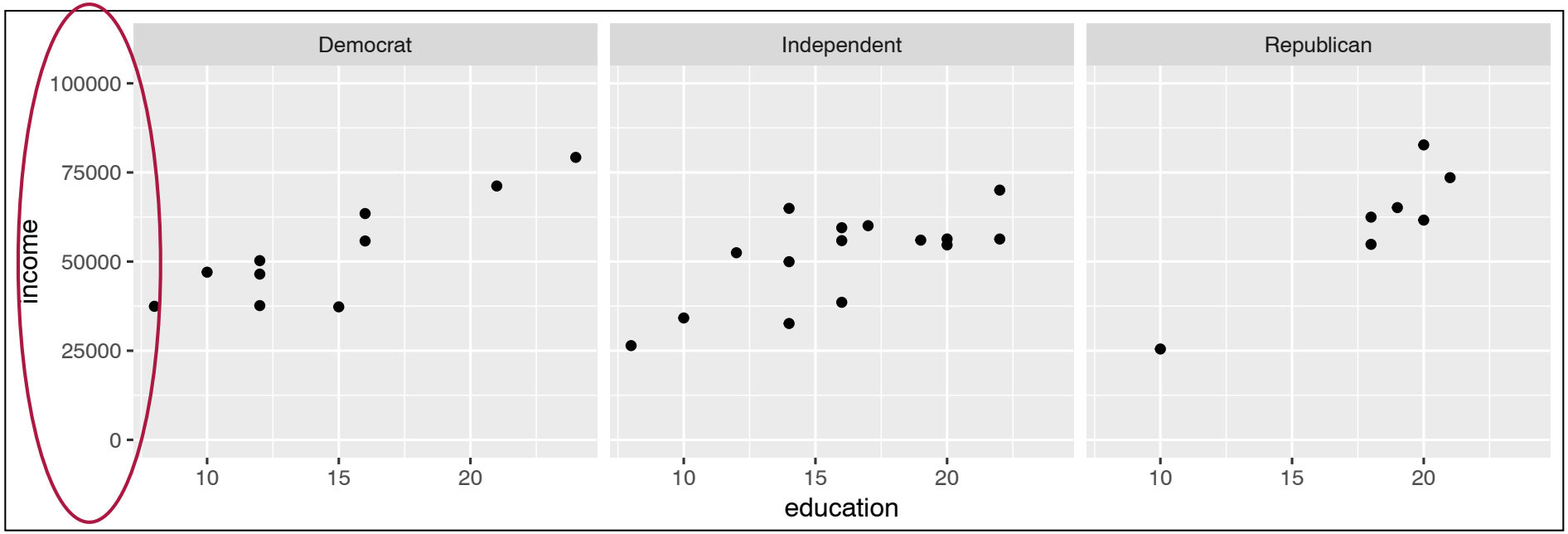
Changing the Axis Limits

```
> ggplot(data = city, aes(x = education, y = income)) +
    geom_point() +
    facet_wrap(~ party) +
    ylim(0, 100000)
```

The first value is the minimum.

The second value is the maximum.

`xlim()` and `ylim()` are used to set the limits on the x-axis and y-axis respectively.



Fine-Tuning Axis Scales

The `xlab()` and `ylim()` functions we used are shortcuts to using scaling layers. The use of scaling layers allows much more fine-tuning and control of the axis scales.

There are four different scaling functions you can use depending on which axis (*x* or *y*) you want to control and whether the variable plotted along that axis is *continuous* or *discrete*. The four functions are (1) `scale_x_continuous()`, (2) `scale_x_discrete()`, (2) `scale_y_continuous()`, and (4) `scale_y_discrete()`.

```
> ggplot(data = city, aes(x = education, y = income)) +
    geom_point() +
    facet_wrap(~ party) +
    scale_x_continuous(
        name = "Education level (in years)",
        breaks = c(10, 12, 14, 16, 18, 20, 22, 24)
    )
```

The `name=` option labels the scale (it is the same as the `ylab()` layer in this case). The `breaks=` option adds break lines on the axis. There are several other options including `labels=` for labelling the break lines, etc.

Prettying Up the Scales

We can get other options for labeling using the **scales** package. For example, we can add commas to separate by thousands in long values, or add the $ for monetary values.

```
> library(scales)

> gplot(data = city, aes(x = education, y = income)) +
    geom_point() +
    facet_wrap(~ party) +
    scale_y_continuous(
      name = "Annual income",
      labels = dollar
      )
```

The `labels=dollar` option is a built-in formatter from the **scales** package that adds the dollar sign and commas to the labels on a specified axis. Read more at http://www.rdocumentation.org/packages/scales/versions/0.4.0

Customizing the Color

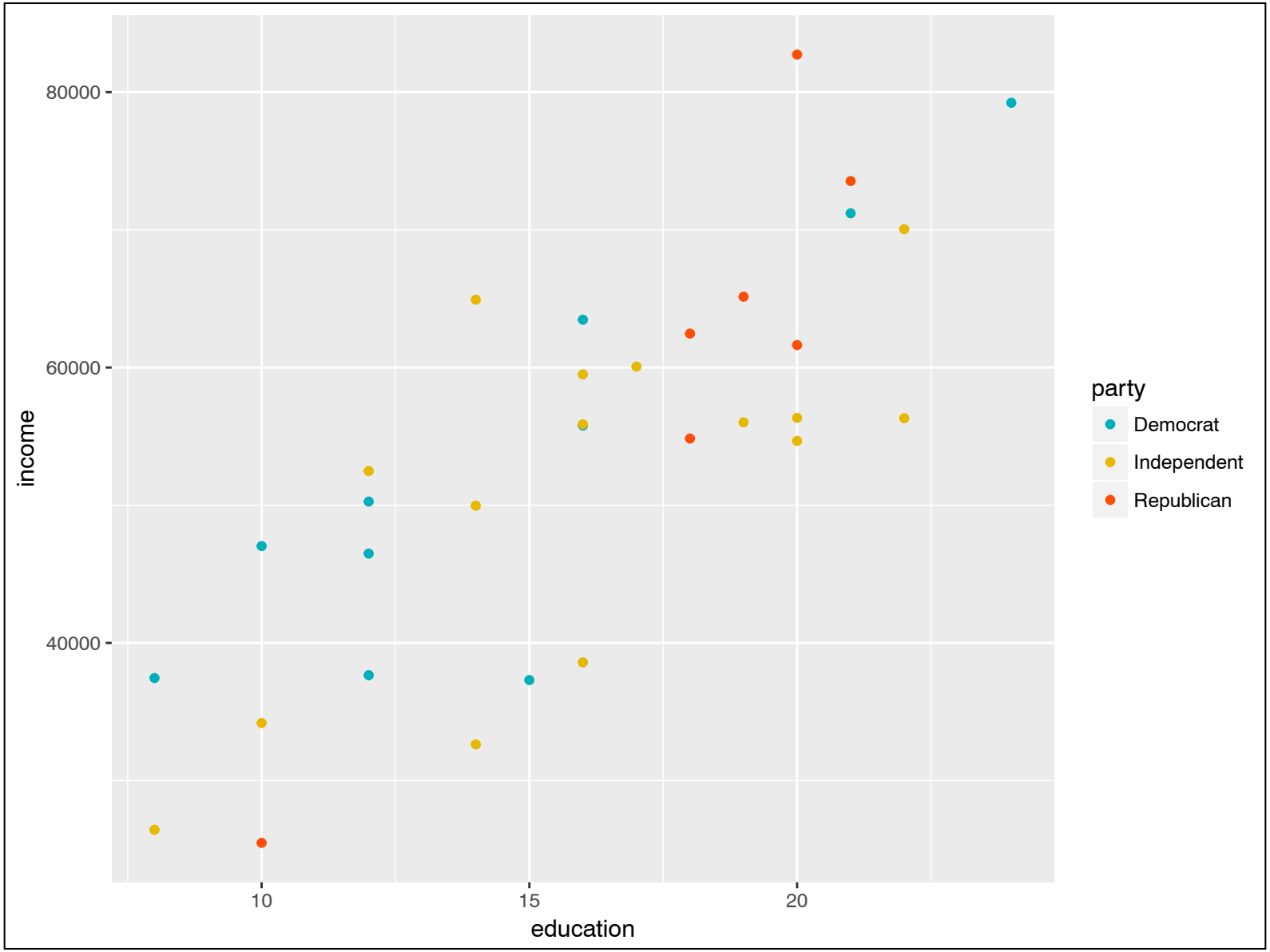
Scaling functions can also be used to fine-tune colors and fills. For these you need to specify either *color* or *fill*, and also the palette you want to use. For example, `scale_color_manual()` can be used to manually set the colors when the `color=` argument is used.

```
> ggplot(data = city, aes(x = seniority, y = education)) +
    geom_point(aes(color = party)) +
    scale_color_manual(
        values = c("#00afbb", "#e7b800", "#fc4e07")
        )
```

`scale_color_manual()` allows you to manually set the attributes associated with the fill aesthetic.

The `values=` argument sets the color values for each level of the factor.

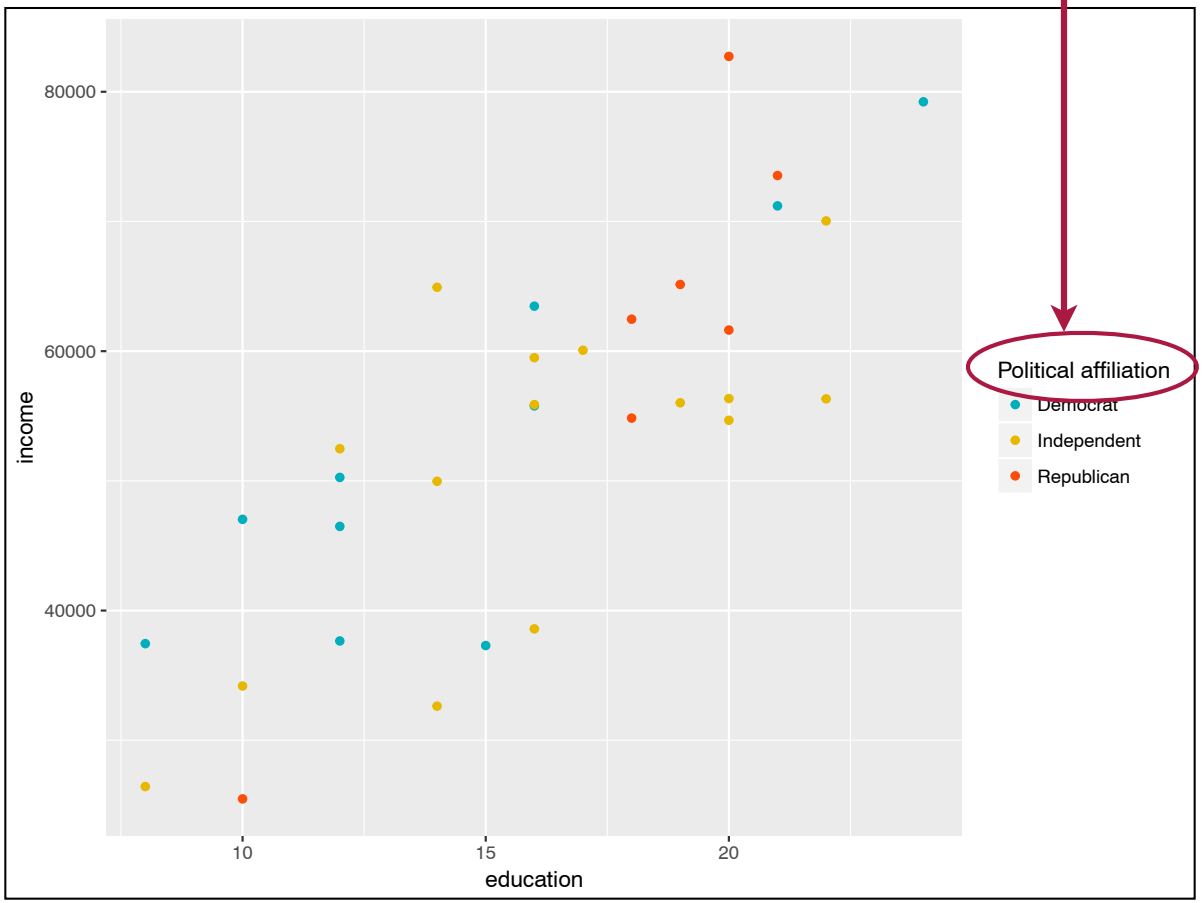
Named colors or HEX values (both given as quoted character strings) can be used in `values=` argument of `scale_color_manual()` or `scale_fill_manual()`.


80000
60000
40000
income
10
15
20
education
party
Democrat
Independent
Republican

`scale()` functions can also be used to change the name and labels in the legend.

```
> ggplot(data = city, aes(x = education, y = income)) +
    geom_point(aes(color = party)) +
    scale_color_manual(
        values = c("#00afbb", "#e7b800", "#fc4e07"),
        name = "Political affiliation"
        )
```

The `name=` argument changes the title of the legend.



Choosing a Color Palette

`colors()` will provide a list of all the **named colors** available in R.

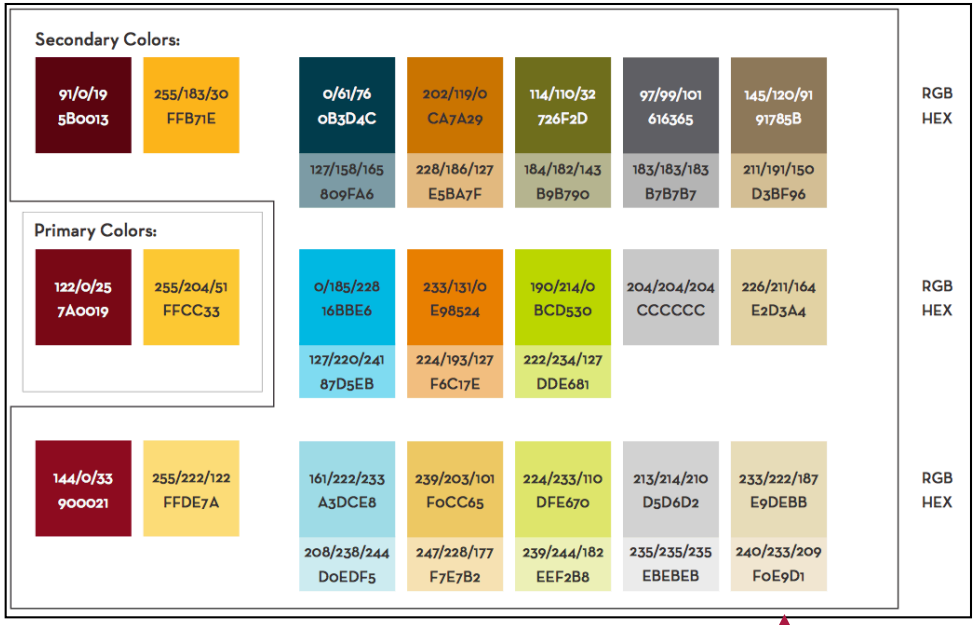
```
> colors()

  [1] "white"           "aliceblue"           "antiquewhite"
  [4] "antiquewhite1"   "antiquewhite2"       "antiquewhite3"
  [7] "antiquewhite4"   "aquamarine"          "aquamarine1"
[10] "aquamarine2"      "aquamarine3"         "aquamarine4"
       ⋮                   ⋮                     ⋮
```

Most universities have official colors. The University of Minnesota's two official colors in HEX (for electronic display) are:

- `#ffcc33` (gold)
- `#7a0019` (maroon)

See more at: https://www.ur.umn.edu/brand/requirements-and-guidelines/color-and-type/



The U of M also has an entire palette of secondary colors available at:
https://www.ur.umn.edu/brand/assets/pdf/secondary_colors_rgb.pdf

Pre-Selected Color Palettes

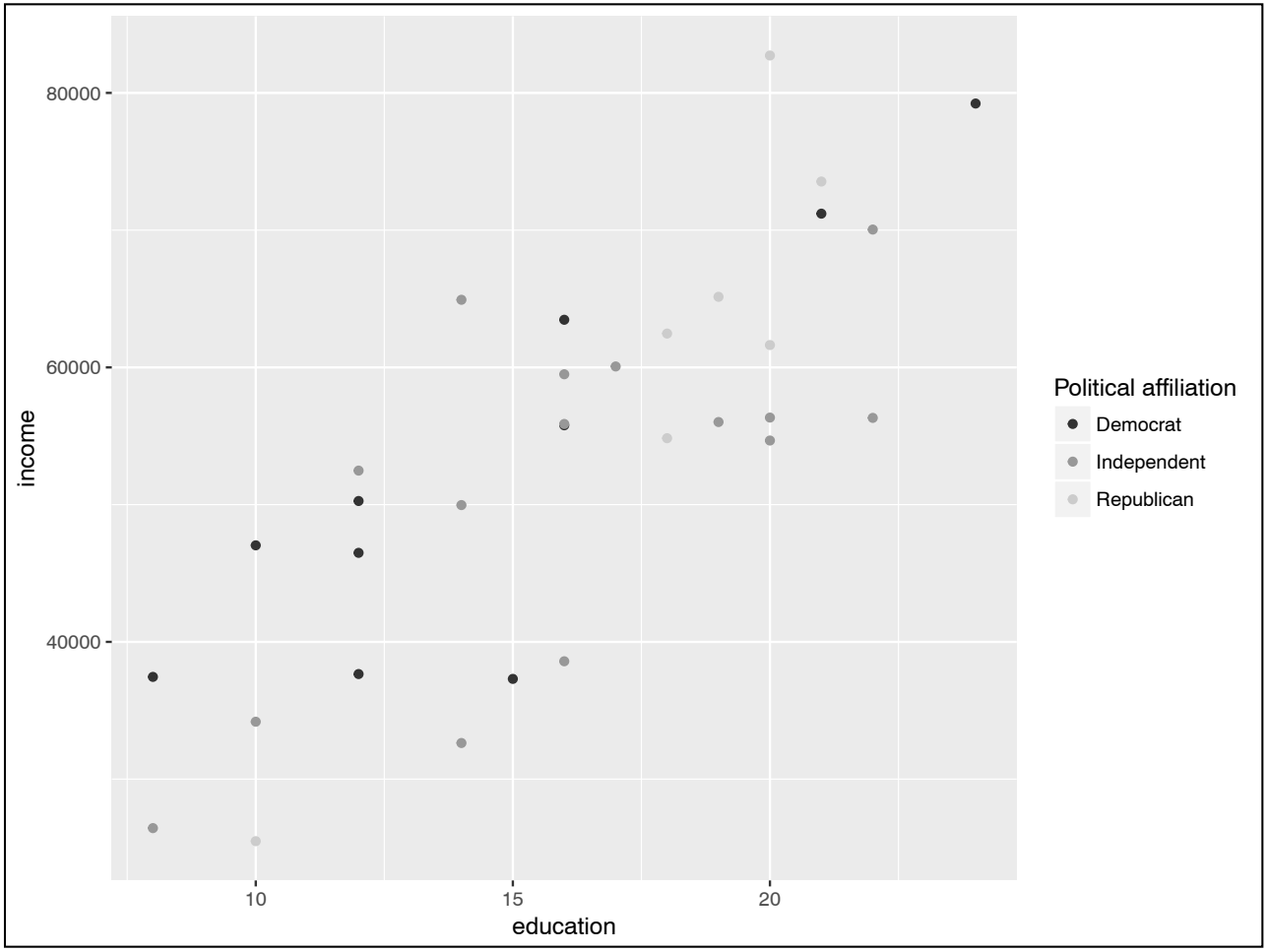
There are several "built-in" color palettes available for use in ggplot

Fill Scale	Color Scale	Description
`scale_fill_hue()`	`scale_color_hue()`	Colors evenly spaced around the color wheel
`scale_fill_grey()`	`scale_color_grey()`	Grey scale palette
`scale_fill_brewer()`	`scale_color_brewer()`	ColorBrewer palettes

Grey Scale Color Palette

```
> ggplot(data = city, aes(x = education, y = income)) +
    geom_point(aes(color = party)) +
    scale_color_grey(name = "Political affiliation")
```

The `scale_color_grey()` and `scale_fill_grey()` functions use a greyscale color palette. This is a useful palette if you are printing in black-and-white.



Color Brewer

Cynthia Brewer chose color palettes that not only are aesthetically pleasing, but also based on how humans perceive the colors that are displayed.

http://www.colorbrewer2.org

She has palettes for three different types of data

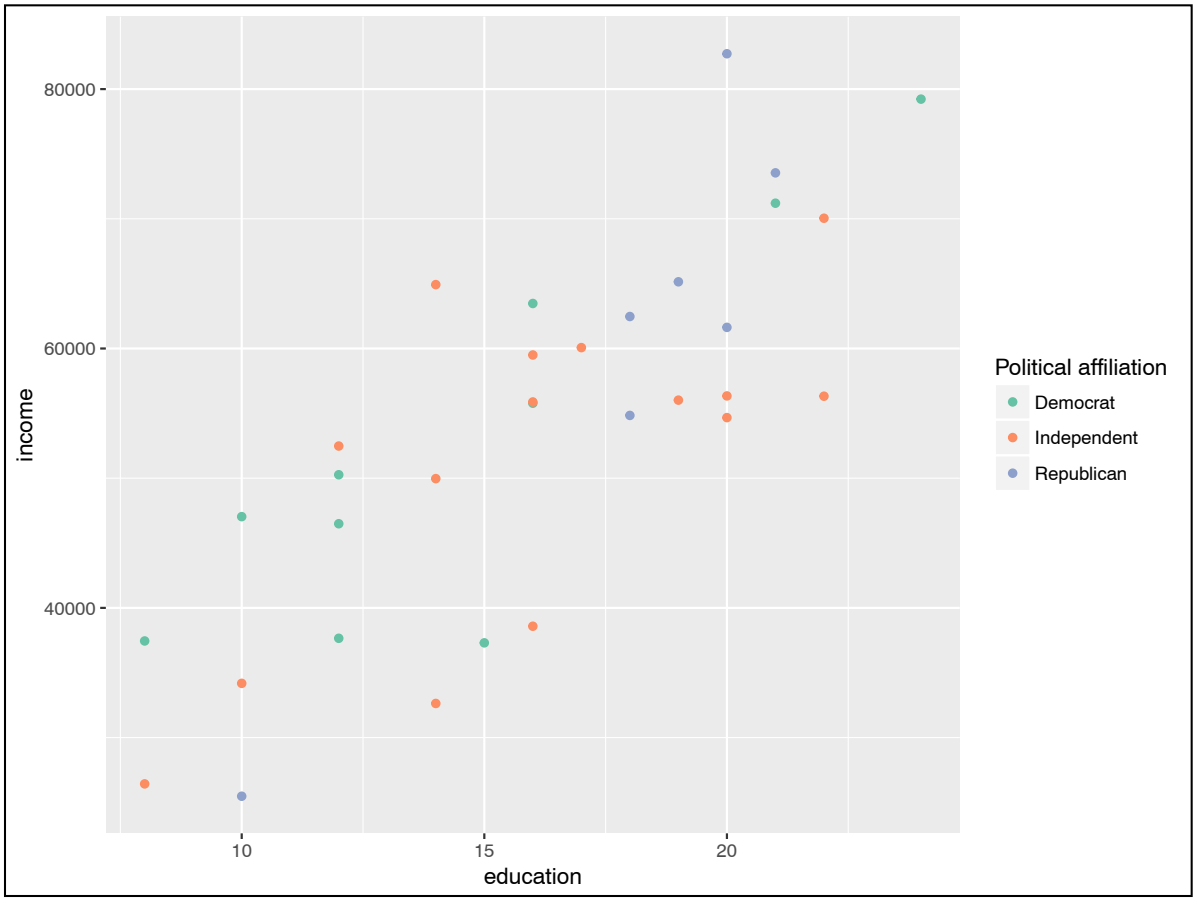
- **Qualitative/Categorical**—colors do not have a perceived order
- **Sequential**—colors have a *perceived order* and perceived difference between successive colors is uniform
- **Diverging**—two back-to-back sequential palettes starting from a common color (e.g., for Likert scale data)

There is a very readable introduction to color brewer palettes at http://mkweb.bcgsc.ca/brewer/

Brewer Color Palette

```
> ggplot(data = city, aes(x = education, y = income)) +
    geom_point(aes(color = party)) +
    scale_color_brewer(
        name = "Political affiliation",
        palette = "Set2"
        )
```

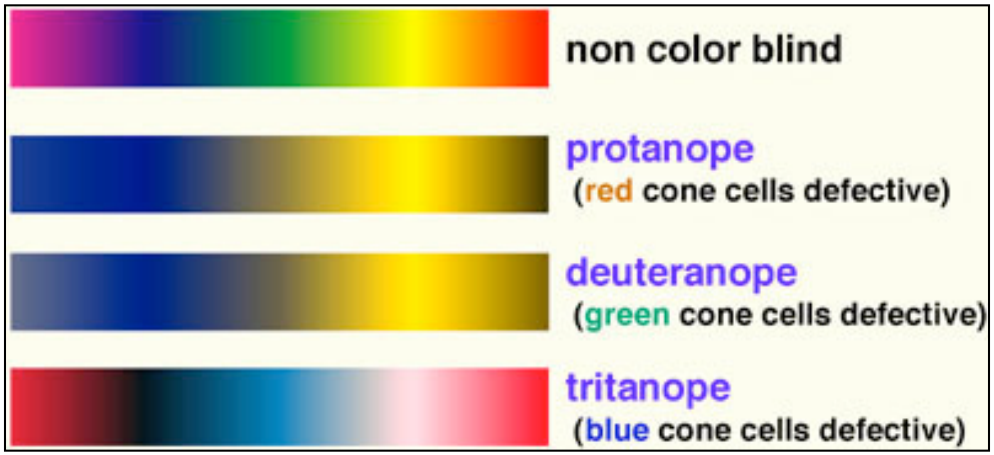
The `scale_color_brewer()` and `scale_fill_brewer()` functions use a Cynthia Brewer's color palettes. You need to specify a palette using the `palette=` argument.



Palettes for Color-Blindness

About 8% of males and ½% of females have some form of color vision deficiency (good chance that someone in your audience will be one of these people)

Color *and* grey-scale palettes have been developed for use with people that have the more common forms of color-blindness



There is more information related to color-blindness and the creation of suitable color palettes for scientific figures at http://jfly.iam.u-tokyo.ac.jp/color/

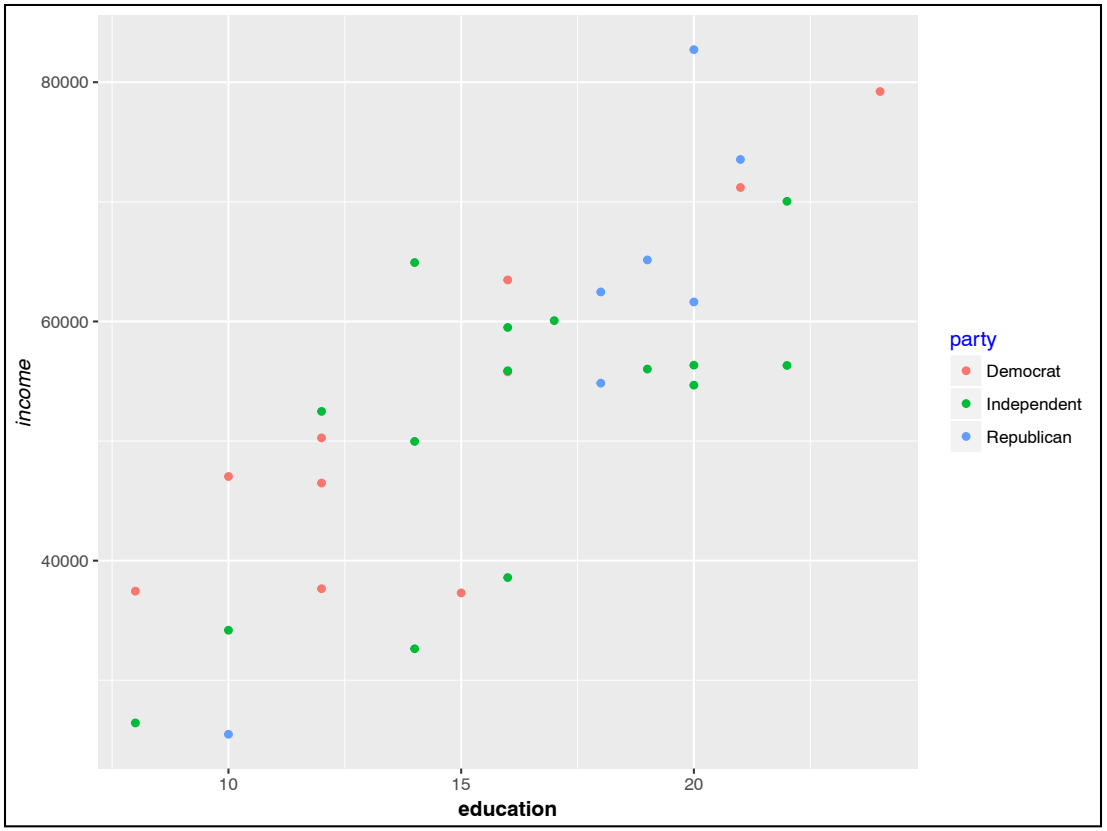
There is a large body of research literature related to the creation of suitable color palettes for figures. As a starting point,

Lumley, T. (2006). Color-coding and color blindness in statistical graphics. *Statistical computing and graphics newsletter*. http://www.amstat-online.org/sections/graphics/newsletter/Volumes/v172.pdf

Fine-Tuning the Theme

```
> ggplot(data = city, aes(x = education, y = income)) +
    geom_point(aes(color = party)) +
    theme(
         axis.title.x = element_text(face = "bold"),
         axis.title.y = element_text(face = "italic"),
         legend.title = element_text(color = "blue")
         )
```

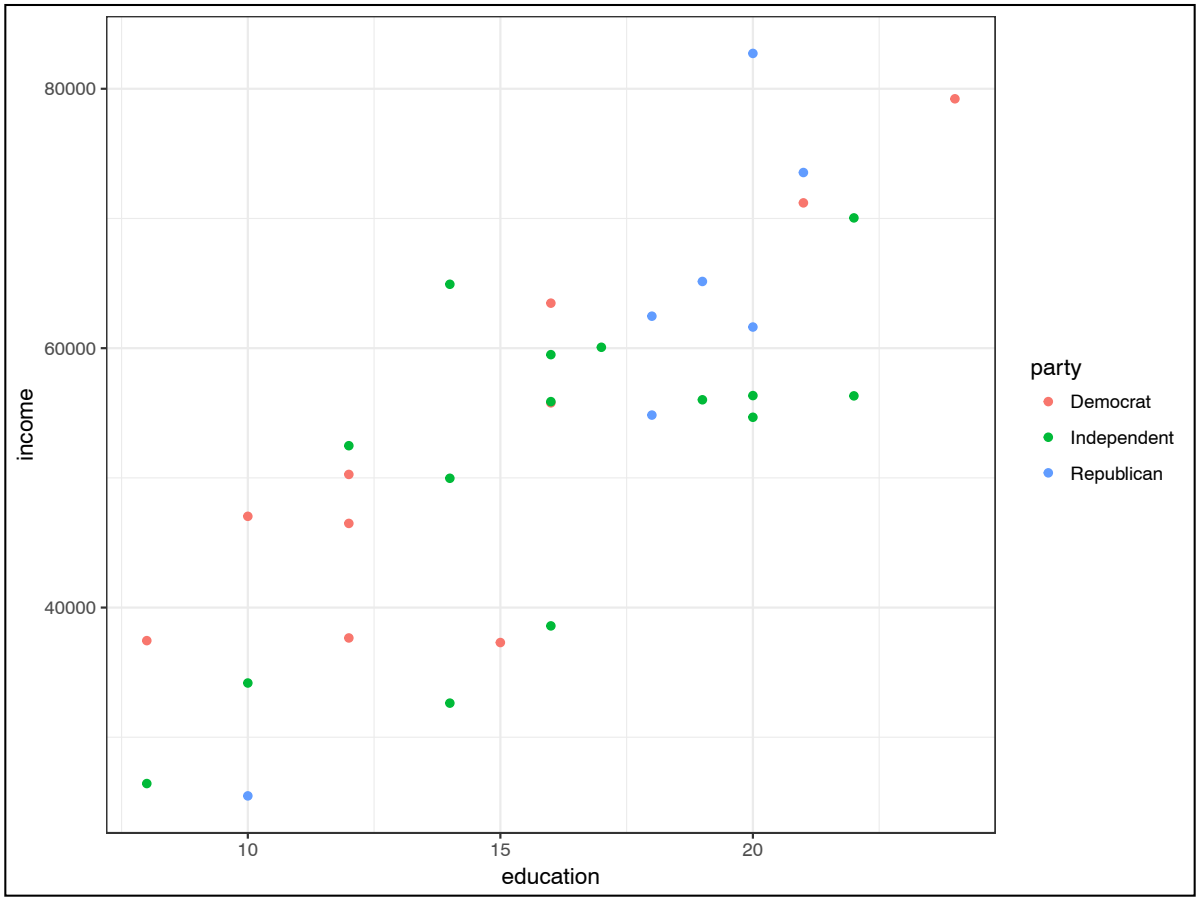
The `theme()` function can be used to change *every* element in the plot (e.g., grid lines, font, color, etc.). See http://docs.ggplot2.org/current/theme.html



Using "Built-In" Themes

```
> ggplot(data = city, aes(x = education, y = income)) +
    geom_point(aes(color = party)) +
    theme_bw()
```

The `theme_bw()` function is a "built-in" theme that uses a black-and-white background (rather than grey).

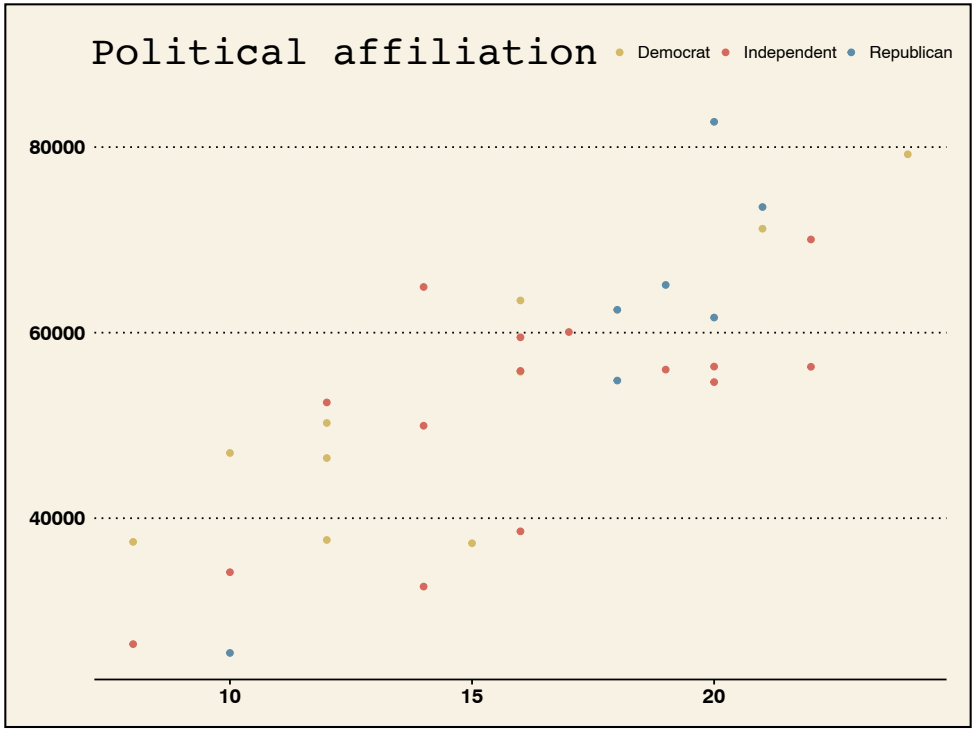


There are many other themes available

- http://drunks-and-lampposts.com/2012/10/02/clegg-vs-pleb-an-xkcd-esque-chart/
- https://github.com/jrnold/ggthemes

```
# Install the ggthemes library then load it
> library(ggthemes)

> ggplot(data = city, aes(x = education, y = income)) +
    geom_point(aes(color = party)) +
    theme_wsj() +
    scale_color_wsj(name = "Political affiliation", palette = "rgby")
```



You can also build your own themes and use them.

Putting It All Together

```
> ggplot(data = city, aes(x = education, y = income)) +
    geom_point(aes(color = party)) +
    scale_color_manual(
      name = "Political affiliation",
      values = c("#00afbb", "#e7b800", "#fc4e07")
      ) +
    xlab("Education level (in years)") +
    scale_y_continuous(
      name = "Annual income",
      labels = dollar
      ) +
    theme_bw() +
    facet_wrap(~ gender)
```

Make a rough sketch of the plot you think this syntax will produce.

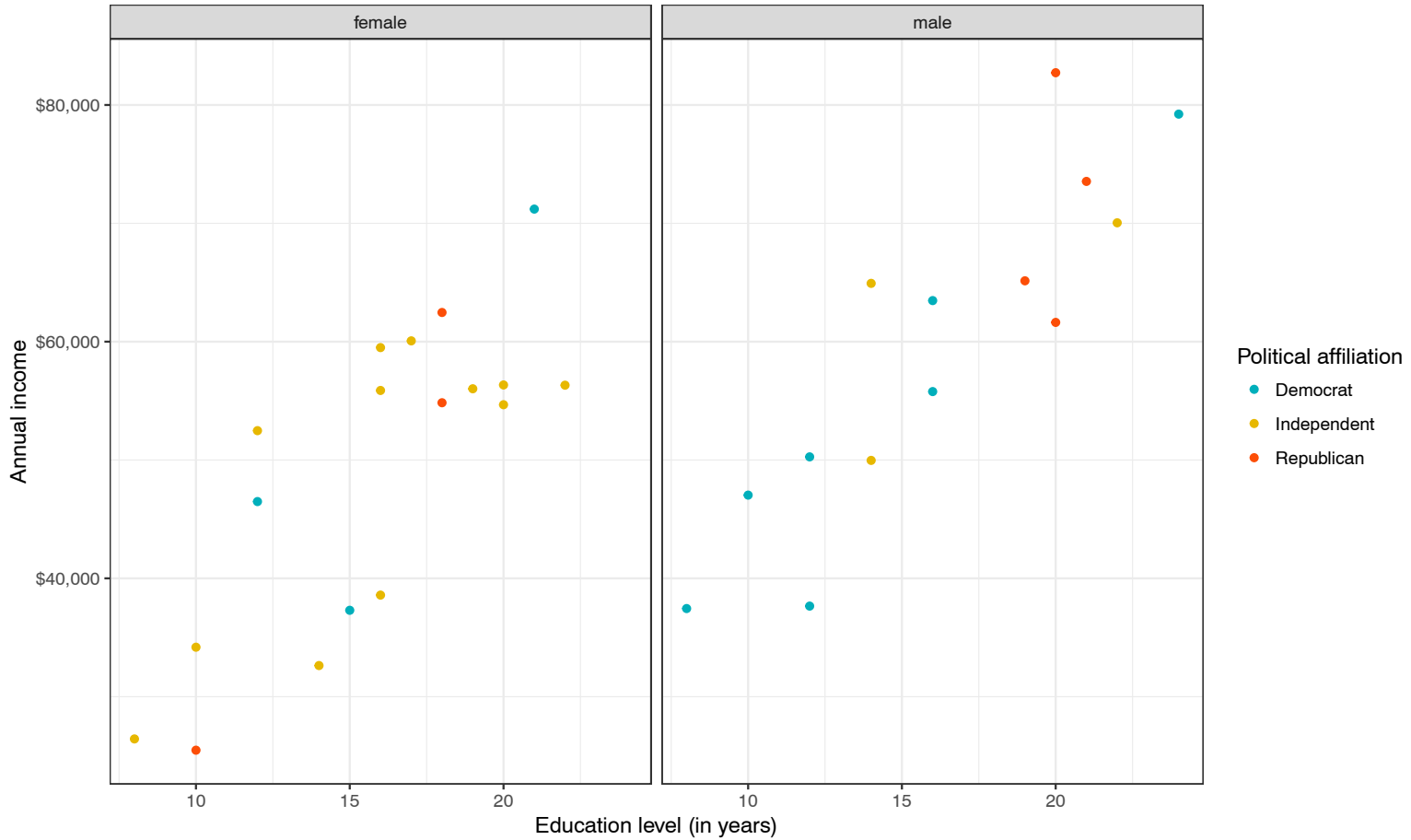


Figure 1. Relationship between annual income (in U.S. dollars) and years of education for $n = 32$ City of Riverside employees. This relationship is shown for both males and females.

`#protip`**:** It is easier to use a word-processor (e.g., Word) to add the figure title and caption than to try and get it formatted correctly using R.

`#protip`**:** When you only have a few colors, include them in the caption rather than as a legend if you have space limits.

ggplot Resources

- **ggplot2 Cheatsheet**: A one-page (front and back) cheatsheet of ggplot2 syntax with pictures https://www.rstudio.com/wp-content/uploads/2015/08/ggplot2-cheatsheet.pdf

- **ggplot2 Extensions**: Third-party and user contributed extensions for some pretty cool plots http://www.ggplot2-exts.org/index.html

- **Cookbook for R:** Web-based version of Winston Chang's R Graphics Cookbook **http://www.cookbook-r.com/Graphs/** (The UMN library has electronic access to the actual book. Just search for "R Graphics Cookbook" and log-in with your x500.)

- **extrafonts package:** Use almost any font on your computer in your plots. http://blog.revolutionanalytics.com/2012/09/how-to-use-your-favorite-fonts-in-r-charts.html

`#protip`**:** Use Google to find out how to do just about anything with ggplot.