Bar Plots to Illustrate Means

James M. Murray, Ph.D.
University of Wisconsin - La Crosse

Updated: September 20, 2017

PDF file location: http://www.murraylax.org/rtutorials/barplots.pdf

HTML file location: http://www.murraylax.org/rtutorials/barplots.html

---

Note on required packages: The following code requires the packages `tidyverse`, `scales`, `stringr`, `Hmisc`, and `ggthemes`.

- The `tidyverse` package contains many packages that allow you to organize, summarize, and plot data.
- We use the `scales` library to customize the scales of our axes.
- The `stringr` package allows us to manipulate strings, which we use to manipulate string labels.
- The `Hmisc` provides mathematical and statistical functions to use with our plots.
- The `ggthemes` package provides multiple themes, which are combinations of parameters to change a plots look and feel

If you have not already done so, download, install, and load the libraries with the following code:

```
install.packages("tidyverse") # This only needs to be executed once for your machine

install.packages("scales") # This only needs to be executed once for your machine

install.packages("stringr") # This only needs to be executed once for your machine

install.packages("Hmisc") # This only needs to be executed once for your machine

install.packages("ggthemes") # This only needs to be executed once for your machine

library("ggplot2") # This needs to be executed every time you load R

library("scales") # This needs to be executed every time you load R

library("stringr") # This needs to be executed every time you load R

library("Hmisc") # This needs to be executed every time you load R

library("ggthemes") # This needs to be executed every time you load R
```

---

1 Introduction

The bar plots in this tutorial visually describe the mean of a numerical variable across different factors of a categorical variable. We also expand the bar plots to include visuals for the margins of error for the estimates of the means.

Data set: The following examples use a sample from the 2016 Current Population Survey which is a monthly survey conducted by the U.S. Census Bureau and Bureau of Labor Statistics. The data is used, among other things, to compute employment statistics such related to earnings, hours employed, and unemployment. This particular sample includes 1,552 observations and includes only head-of-households.

The line below downloads and loads the data set.

```r
load(url("http://murraylax.org/datasets/cps2016.RData"))
```

The data is in a `data.frame` object called `df` and a description of the variables is given in another `data.frame` object called `df.desc`. You can familiarize yourself with the data set by opening these data frames in *Rstudio*. Alternatively, you can get a short description of the data frame `df` with a call to the `str()` function.

```r
str(df)
```

```
## 'data.frame':    1552 obs. of  16 variables:
##  $ age        : num  46 37 35 38 66 28 50 49 63 43 ...
##  $ incwage    : num  24000 86000 22000 35000 50000 24000 75000 52000 40000 30000 ...
##  $ sex        : Factor w/ 2 levels "Female","Male": 1 2 1 2 2 1 1 2 2 2 ...
##  $ race       : Factor w/ 5 levels "Asian/Pacific Islander",..: 5 5 2 5 5 5 5 5 5 5 ...
##  $ empstat    : Factor w/ 4 levels "Armed Forces",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ inlf       : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ unempl     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ industry   : Factor w/ 7 levels "Agriculture, Forestry, and Fishing",..: 7 6 5 3 1 5 5 6 5 7 ...
##  $ usualhrs   : num  32 40 NA 40 60 40 45 40 45 40 ...
##  $ ureason    : Factor w/ 4 levels "Job Leaver","Job Loser",..: NA NA NA NA NA NA NA NA NA NA ...
##  $ vetran     : num  0 0 0 0 1 0 0 0 0 0 ...
##  $ usualhrearn: num  14.4 41.2 NA 16.8 16 ...
##  $ edu        : Ord.factor w/ 5 levels "Less than high school"<..: 2 4 3 1 3 3 5 3 3 3 ...
##  $ medoop     : num  1000 3300 3570 1500 1730 ...
##  $ insprem    : num  0 1000 1800 0 480 6500 0 2000 16000 4000 ...
##  $ totmed     : num  1000 4300 5370 1500 2210 ...
```

2 Creating the Bar Plot

In this example, we compare the mean usual hourly earnings (`usualhrearn`) across different industries (`industry`). Usual hourly earnings is a numerical variable expressed in dollars. Industry is a categorical variable with descriptive factors.

2.1 Data and Aesthetics Layers

We start with a call to `ggplot()` that specifies the data and aesthetics layers. We set the data parameter equal to the data frame, `df`, and set the mapping with a call to the `aes()` function that maps the x-axis to `industry` and the y-axis to `usualhrearn`.

```r
bar.base <- ggplot(data=df, mapping=aes(x=industry, y=usualhrearn))
```

This sets up the first two layers of the plot which is a useful base to save. We still do not have a visual plot as we have not completed the minimal number of layers. We still have to specify a geometry layer to have some kind of shape to draw. We do this indirectly in the next subsection.

2.2 Statistics and Geometry Layers

In the case of a bar plot, we do not wish to graph the raw data, but rather a *statistic* that summarizes the data. Our statistic is the *mean*. We wish to compute the mean of the usual hourly earnings for each industry, and plot rectangular bars with a height equal to the mean. To do this, we will not specify a geometry layer, but rather a **statistics layer** which will both compute the statistic in which we are interested and specify the *bar* geometry.

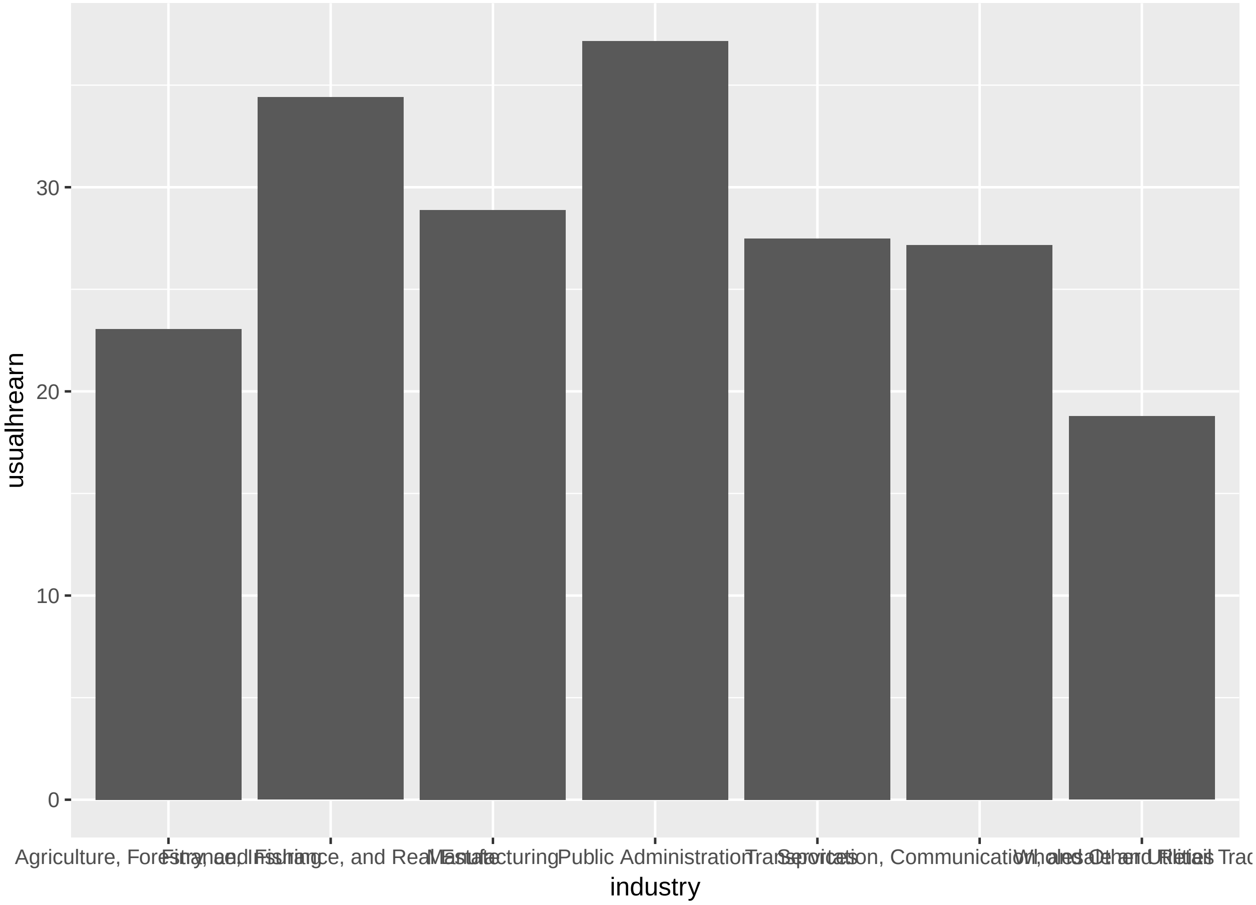
```
bar.p <- bar.base + stat_summary(fun.data=mean_sdl, geom="bar")
```

The call to `stat_summary()` specifies our statistics and geometry layer. The first parameter `fun.data=mean_sdl` tells `stat_summary()` what function to call to calculate the desired statistics. The function `mean_sdl()` takes a single variable as a parameter and computes the mean and lower and upper limits of a confidence interval. The second parameter, `geom="bar"`, tells `stat_summary()` to use the bar geometry for the statistics it calculates.

Let us look at the plot:

```
bar.p
```

```
## Warning: Removed 271 rows containing non-finite values (stat_summary).
```



We get a warning that many observations were not included because there are a number of people in our sample that are not employed so they did not have usual weekly hours and/or wage or salary income. We still have over 1000 data points, so this is not problematic.

2.3. Fix Labels - Text Wrap

The labels for the industries are long and overlap with one another. To address this, we can wrap our x-axis labels onto new lines so that they do not overwrite each other. To do this, we will change the text for the levels of the factor variable, `industry`. Let us remind ourselves what are these levels:

```
levels(df$industry)
```

```
## [1] "Agriculture, Forestry, and Fishing"
## [2] "Finance, Insurance, and Real Estate"
## [3] "Manufacturing"
## [4] "Public Administration"
## [5] "Services"
## [6] "Transportation, Communication, and Other Utilities"
## [7] "Wholesale and Retail Trade"
```

The code below calls the function `str_wrap()` to re-write the levels for `industry`:

```
levels(df$industry) <- str_wrap( levels(df$industry), width=12 )
```

The function `str_wrap()` takes as parameters a string or vector of strings and a desired width in characters. We pass the vector of strings that `levels(df$industry)` returns, we specify a maximum width of 12 characters, and `str_wrap()` outputs a new vector of strings that includes new lines as necessary to assure that no string goes over 12 characters in a single line. We save that output to `levels(df$industry)`, which overrides the existing levels.

Now let us view our levels.

```
levels(df$industry)
```

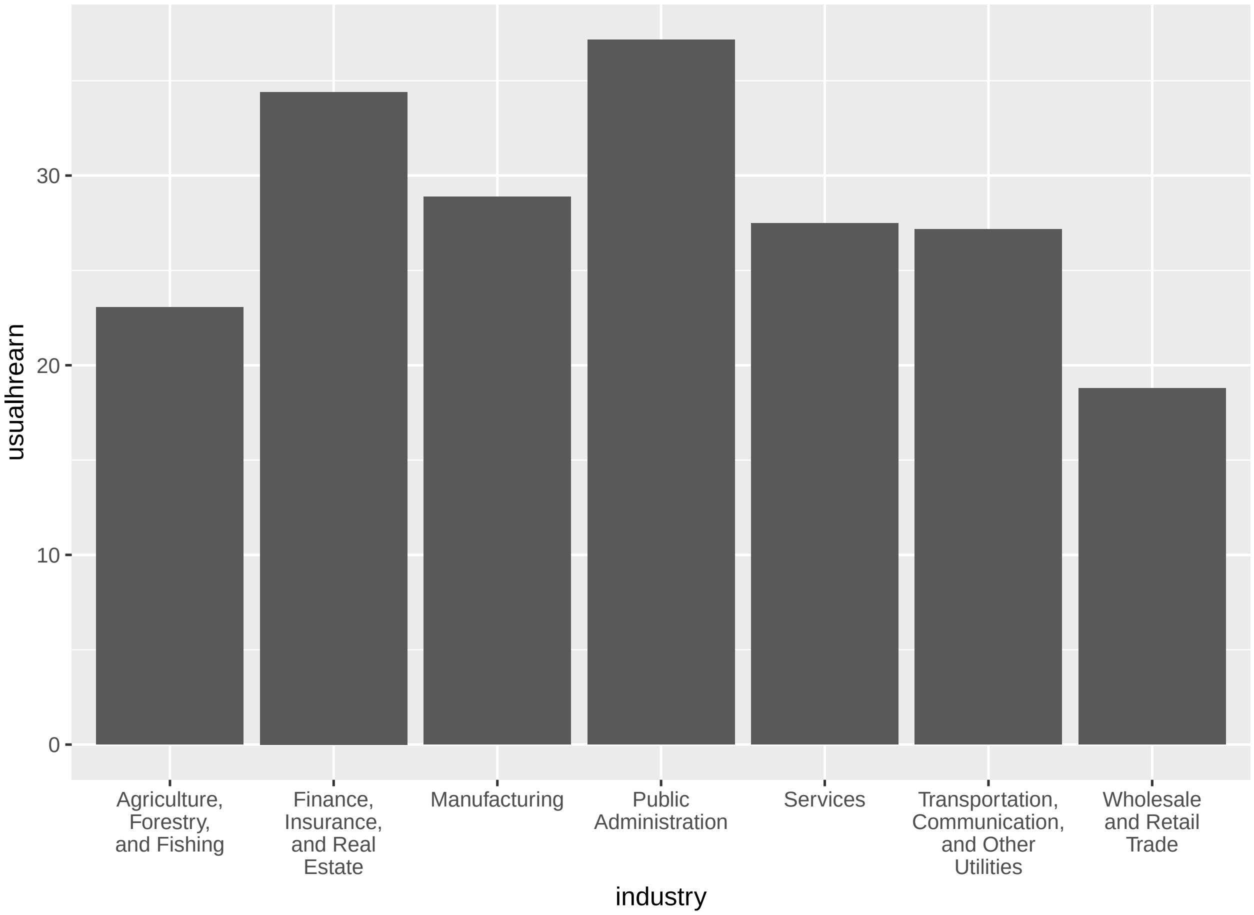
```
## [1] "Agriculture,\nForestry,\nand Fishing"
## [2] "Finance,\nInsurance,\nand Real\nEstate"
## [3] "Manufacturing"
## [4] "Public\nAdministration"
## [5] "Services"
## [6] "Transportation,\nCommunication,\nand Other\nUtilities"
## [7] "Wholesale\nand Retail\nTrade"
```

The newline characters, `\n`, indicate where a new line will appear.

Unfortunately, we need to recreate the plot from scratch now. We changed the definition of one of our variables (the factor variable `industry` is different, with different levels), which entered into the aesthetics layer in the very first call to `ggplot()`. The code below puts everything together, and is constructed by simply copying and pasting code from above to recreate the plot.

```
bar.p <- ggplot(data=df, mapping=aes(x=industry, y=usualhrearn)) +
  stat_summary(fun.data=mean_sdl, geom="bar")
bar.p
```

```
## Warning: Removed 271 rows containing non-finite values (stat_summary).
```



2.4. Labels and Titles

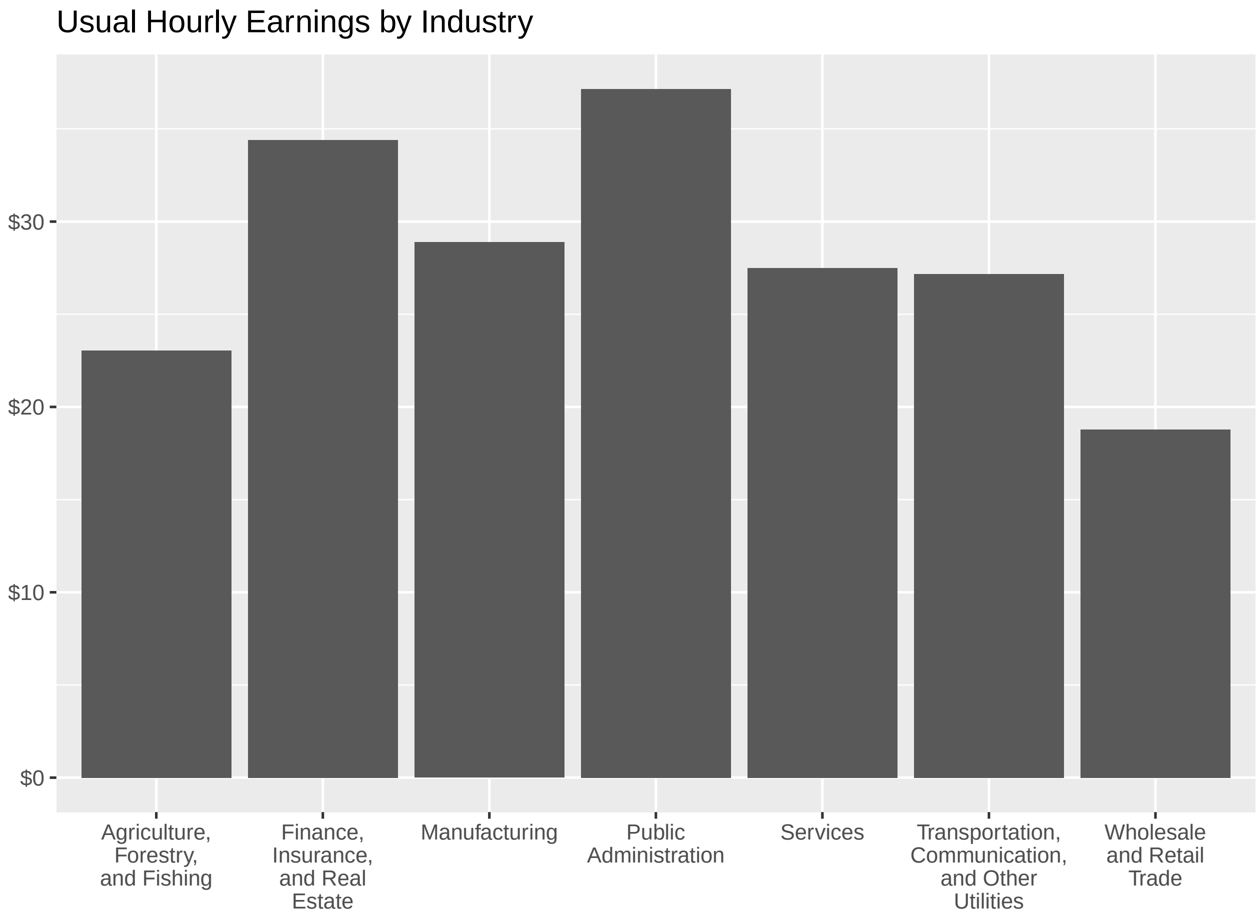
The vertical axis is the mean usual hourly earnings expressed in dollars. So that this is clear to the reader, let us specify that the labels on the scale be expressed in dollars. We can do this with a call to `scale_y_continuous()` which allows us to customize the scale on the y-axis. We set the optional parameter `labels` equal to the *function* dollar() which takes on a single numerical value for a parameter and returns a string with that same number with a dollar sign and commas for every third digit left of the decimal point.

```
bar.p <- bar.p + scale_y_continuous(labels=dollar)
```

Next, we use the function `labs()` to add a descriptive title for plot and we remove the titles for the horizontal and vertical axes since those will be obvious from the title of the plot and the labels on the scales.

```
bar.p <- bar.p + labs(title="Usual Hourly Earnings by Industry", x="", y="")
bar.p
```

```
## Warning: Removed 271 rows containing non-finite values (stat_summary).
```



2.5. Reordering Bars by Mean

It is often useful for visual communication if the bars are ordered by height (mean). This allows the reader to determine at a glance which industries earn the highest and lowest wages and where a particular industry falls in the distribution.

To order the bars by means, we need to create a new industry variable that specifies an order for the levels (categories). Currently, industry is a factor (categorical) variable whose levels are given in alphabetical order. We can verify this with a call to `levels()`.

```r
levels(df$industry)
```

```
## [1] "Agriculture,\nForestry,\nand Fishing"
## [2] "Finance,\nInsurance,\nand Real\nEstate"
## [3] "Manufacturing"
## [4] "Public\nAdministration"
## [5] "Services"
## [6] "Transportation,\nCommunication,\nand Other\nUtilities"
## [7] "Wholesale\nand Retail\nTrade"
```

What we need to do is order the levels of industry according to a statistic (the mean) of another variable (`usualhrearn`). Unfortunately, the code necessary to do this is beyond the scope of this tutorial. Fortunately, I do provide a function that provides this code. You can download it and call it into memory with the following call to `source()`:

```
source("http://murraylax.org/code/R/orderfactor.R")
```

This file provides a function called `order.factor.by()` that takes four parameters: (1) the factor variable to order levels, (2) the numerical variable to compute the mean for each level (3) the function we wish to use for ordering - we will pass along `mean`, but you can potentially pass along another function like `median` or `sum`, and (4) the data frame to find the variables. The function returns a new factor variable whose levels are ordered appropriately, but it is otherwise equal to the original factor.

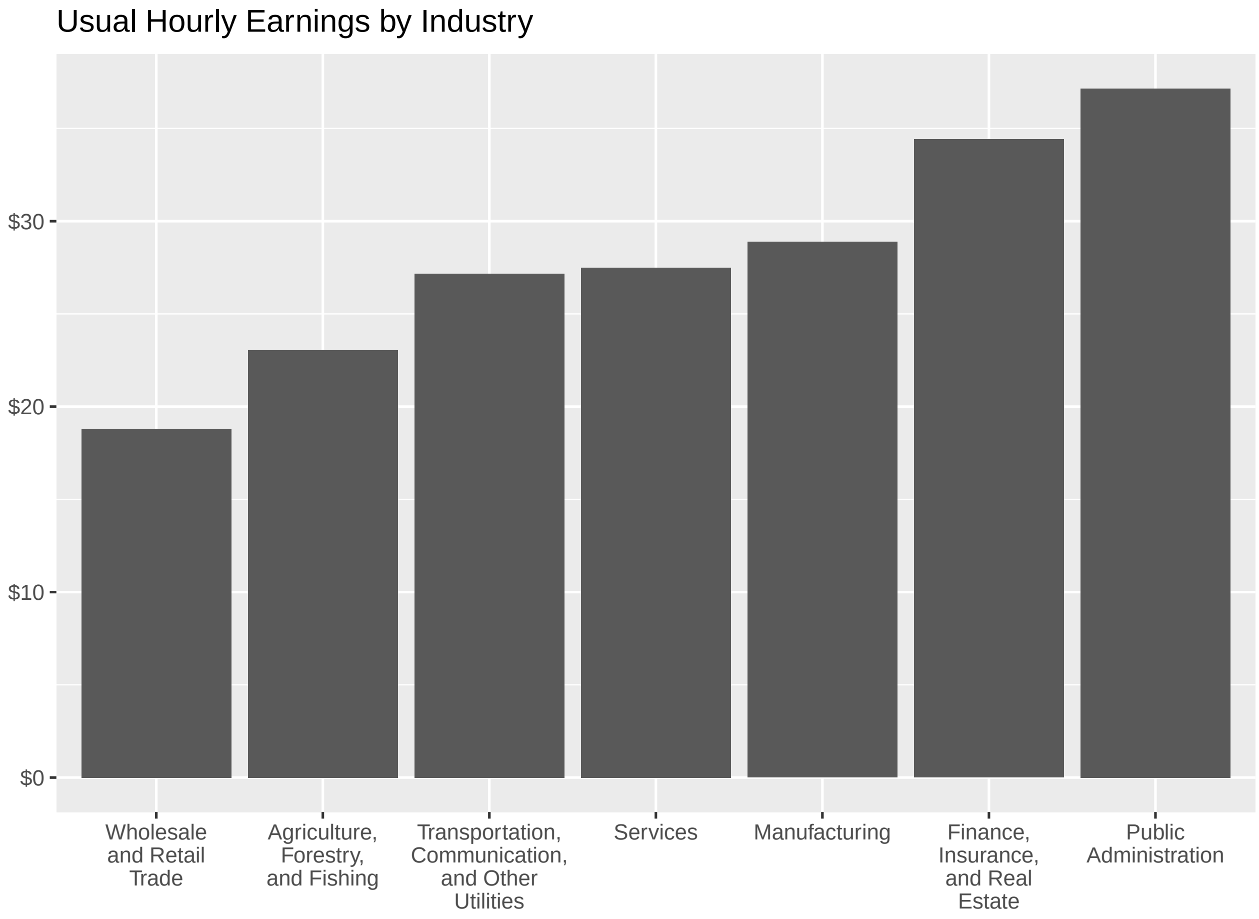
In the code below, we call `order.factor.by()` and let the output overwrite our current `industry` variable in data frame `df`.

```
df$industry <- order.factor.by(industry, usualhrearn, mean, df)
```

Now we can recreate our bar plot, and the industries will be presented left to right in increasing order.

```
bar.ord.p <- ggplot(data=df, mapping=aes(x=industry, y=usualhrearn)) +
  stat_summary(fun.y=mean, geom="bar") +
  scale_y_continuous(labels=dollar) +
  labs(title="Usual Hourly Earnings by Industry", x="", y="")

bar.ord.p
```

```
## Warning: Removed 271 rows containing non-finite values (stat_summary).
```

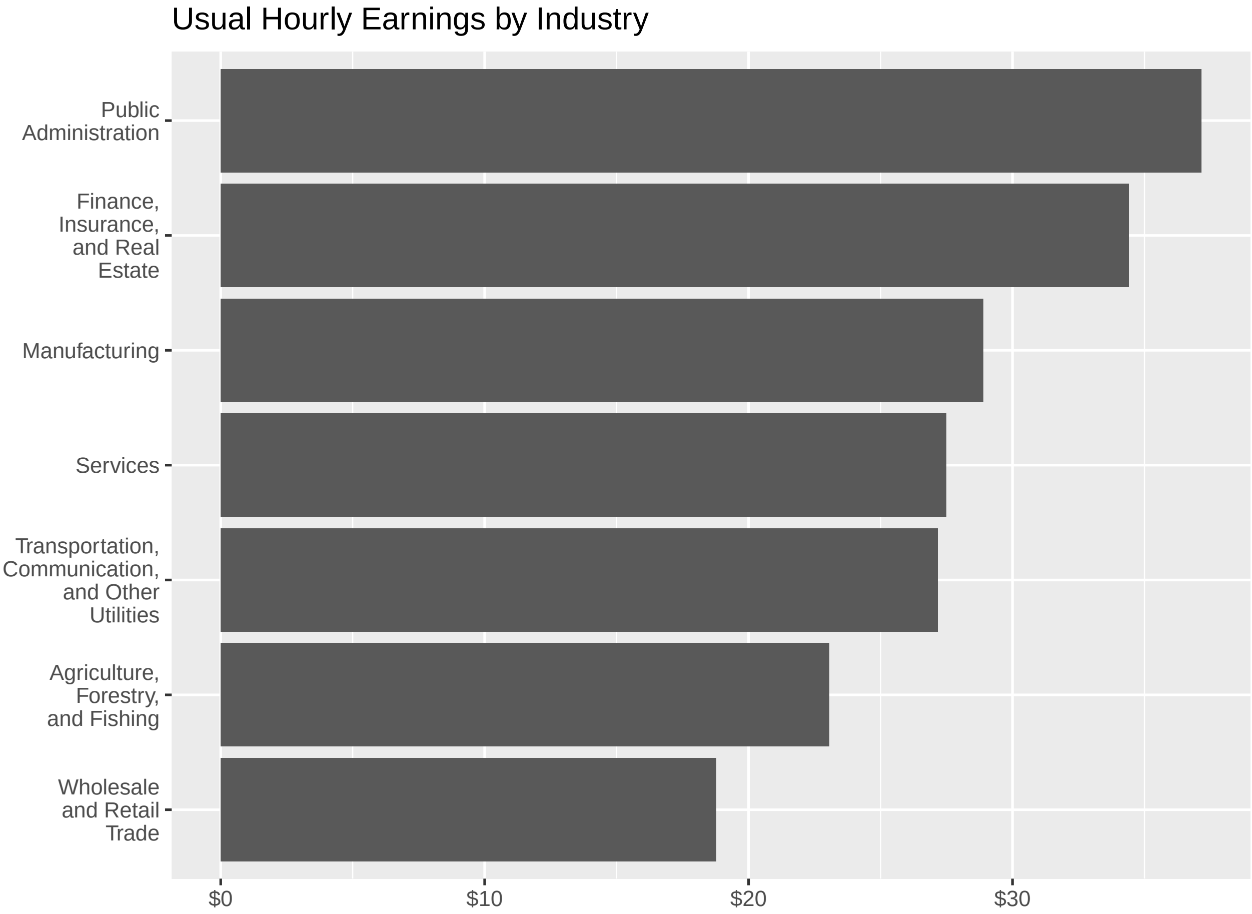


2.6. Flip Coordinates

It is sometimes more convenient to display horizontal bars rather than vertical bars, especially if there are a lot of categories. We can flip the horizontal and vertical axes with a call to `coord_flip()` which is one of many functions that customizes the coordinate layer. In the call below, we preserve our most recent plot that is saved in `bar.ord.p` and add the coordinate layer.

```
bar.flip.p <- bar.ord.p + coord_flip()
bar.flip.p
```

```
## Warning: Removed 271 rows containing non-finite values (stat_summary).
```



2.7. Using Color to Highlight One Category

Generally speaking, it is best practice to keep all of the bars the same color unless a change in color communicates something meaningful.

Let us suppose that our bar plot is part of a study with a focus on the manufacturing industry and the purpose of the bar plot is to highlight how usual hourly earnings in the manufacturing industry compares to other industries. So that our visualization helps achieve this purpose, we wish to make the bar for manufacturing a different color than the other bars.

Color is visual attribute that can be mapped to a variable in the *aesthetics layer*. Let us first create a dummy variable called `manu` that is equal to 1 if the industry variable is equal to “Manufacturing” and 0 otherwise.

```
df$manu <- (df$industry == "Manufacturing")
```

The expression `(df$industry == "Manufacturing")` will compare every observation in the `industry` variable and return a value Boolean `TRUE` if the expression is true and `FALSE` if the value is false.

Next, we build our plot again from the beginning, this time including another mapping in the aesthetics layer.

```
bar.manu.base.p <- ggplot(data=df, mapping=aes(x=industry, y=usualhrearn, fill=manu))
```

The call to `aes()` now includes `fill=manu` which maps the fill visual attribute (which refers to the color of the bars) to the dummy variable `manu`.

Let us finish up the graph with the statistics, theme, and coordinates layers as we have done above.

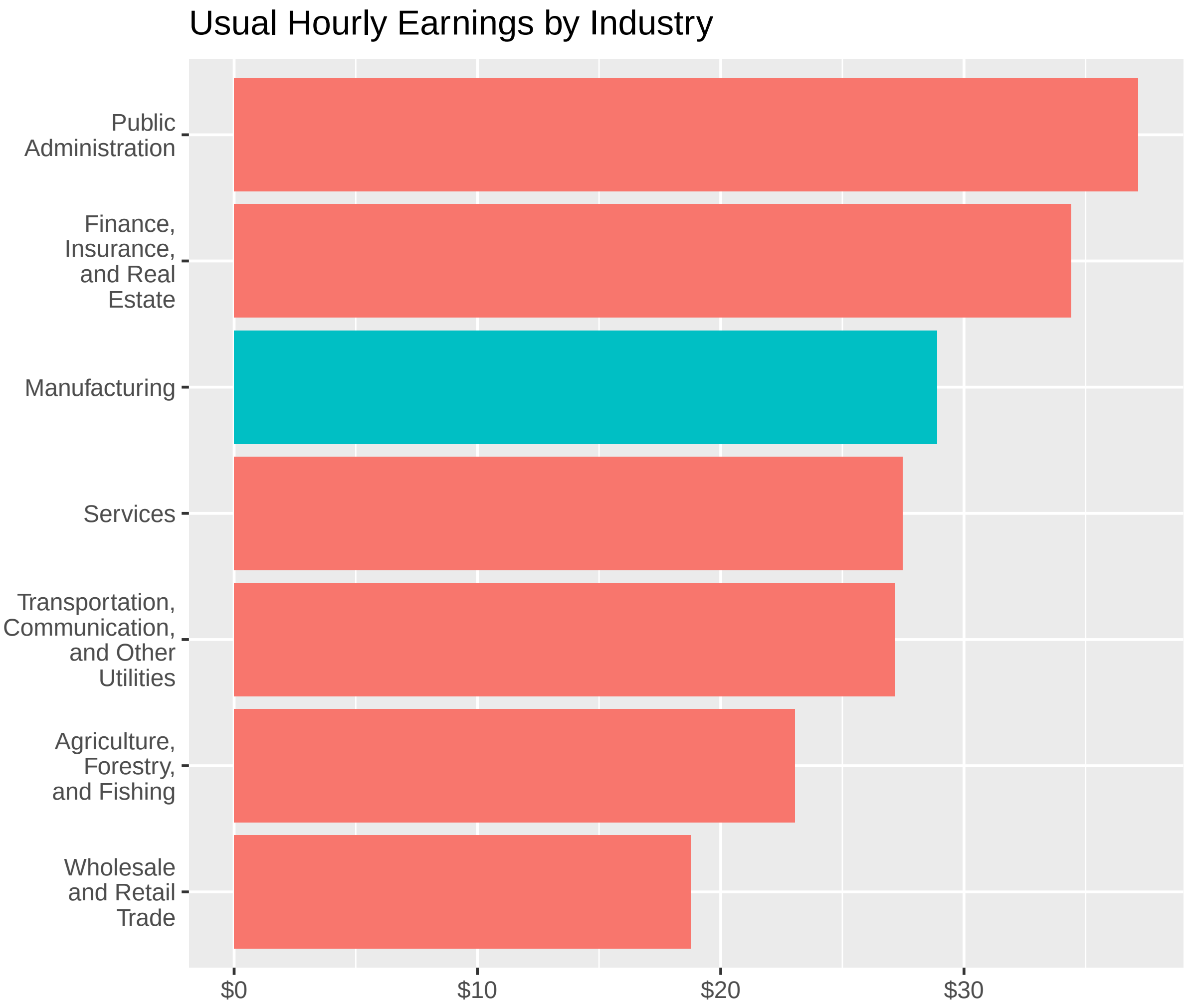
```
bar.manu.p <- bar.manu.base.p +
  stat_summary(fun.y=mean, geom="bar") +
  scale_y_continuous(labels=dollar) +
  labs(title="Usual Hourly Earnings by Industry", x="", y="") +
  coord_flip() +
  theme(legend.position="none")
```

By default, ggplot will provide a legend that describes the color mapping to the dummy variable `manu`. The legend is not necessary, as it is obvious from the graph that only the manufacturing bar is a different color than the others. We remove the legend with the following call to alter the theme layer, `theme(legend.position="none")`.

Let us look at the plot:

```
bar.manu.p
```

```
## Warning: Removed 271 rows containing non-finite values (stat_summary).
```

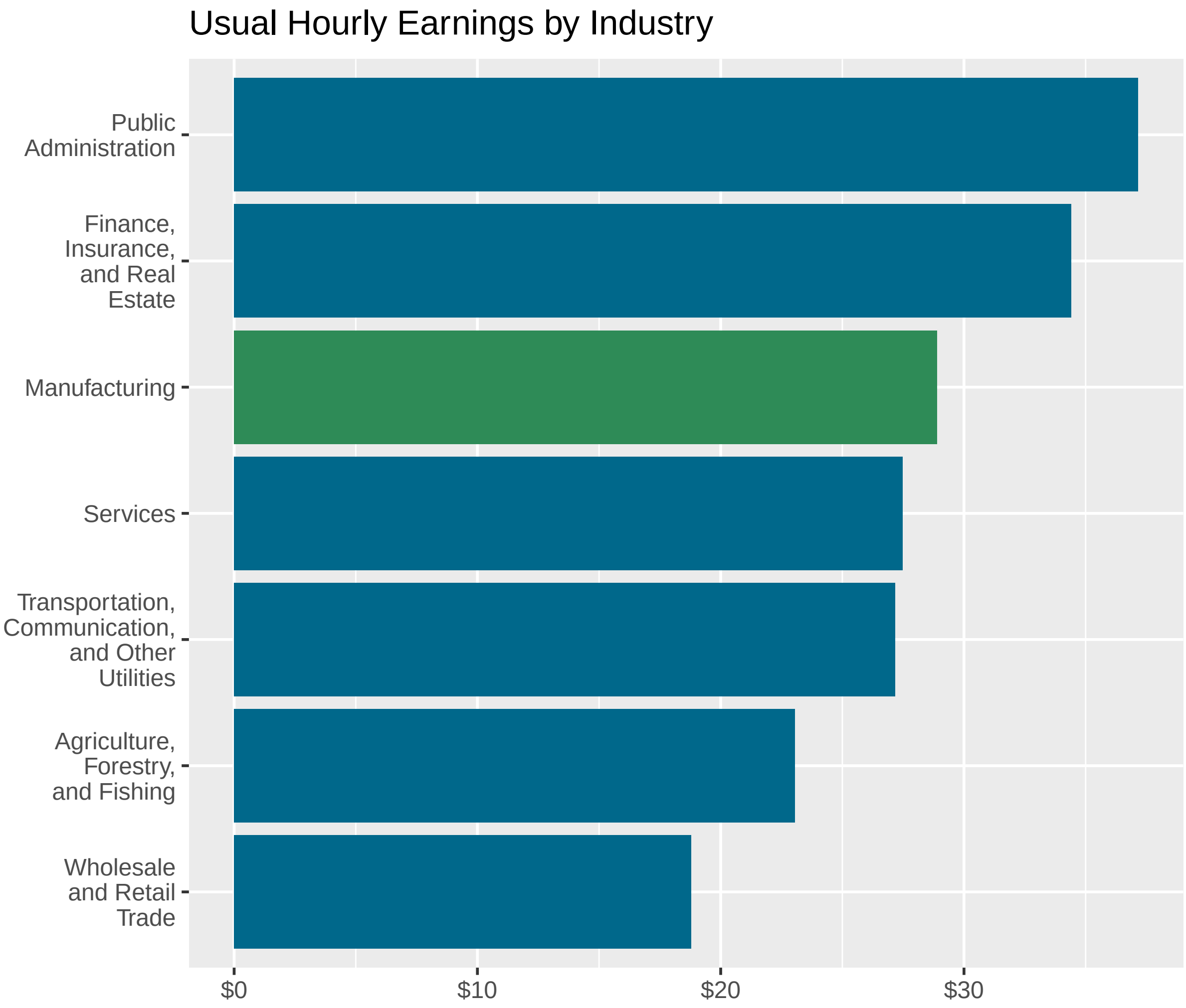


If you prefer different colors, you can alter the fill color scale. The color scale is a visual scale just like the vertical and horizontal axes on the Cartesian plane, and the method for customizing the scale is similar. Since we only have two levels for color (`TRUE` or `FALSE` on whether industry is manufacturing) we only need a color scale with two colors.

The code below defines our binary color scale with colors of our choosing and augments the `bar.manu.p` object with our new scale. I chose the colors “deepskyblue4” and “seagreen”. You can view all your predefined color choices at http://sape.inf.usi.ch/quick-reference/ggplot2/colour. It is also possible to make up your own colors by specifying values for the quantities of red, green, and blue.

```
mycols <- c("deepskyblue4", "seagreen")
bar.manu.p <- bar.manu.p + scale_fill_manual(values=mycols)
bar.manu.p
```

```
## Warning: Removed 271 rows containing non-finite values (stat_summary).
```



3. Error Bars

The purpose of statistics (and data visualization is visual communication of statistics) is to use sample data to make inferences about a whole population. The researcher and reader likely want to know if differences in bar heights are due to actual true differences in usual hourly earnings between industries, or are the differences merely due to statistical chance based on random sampling error. The bar plots without error bars, like those above, do not allow for statistical inference. Moreover, the bar plots can be deceptive if large differences in bar heights are due large degrees of sampling error.

We can add error bars to the bar plots that visually describe the confidence intervals for the mean for each category. We can compute the confidence intervals by adding another statistics layer.

3.1. Confidence Interval with a Normal Distribution

The code below adds a statistics layer, which makes a call to the function `mean_cl_normal()` which computes the mean confidence interval for each category, assuming the sampling distribution for the mean is normal. Below we use our existing bar plot and add on the appropriate statistics layer.

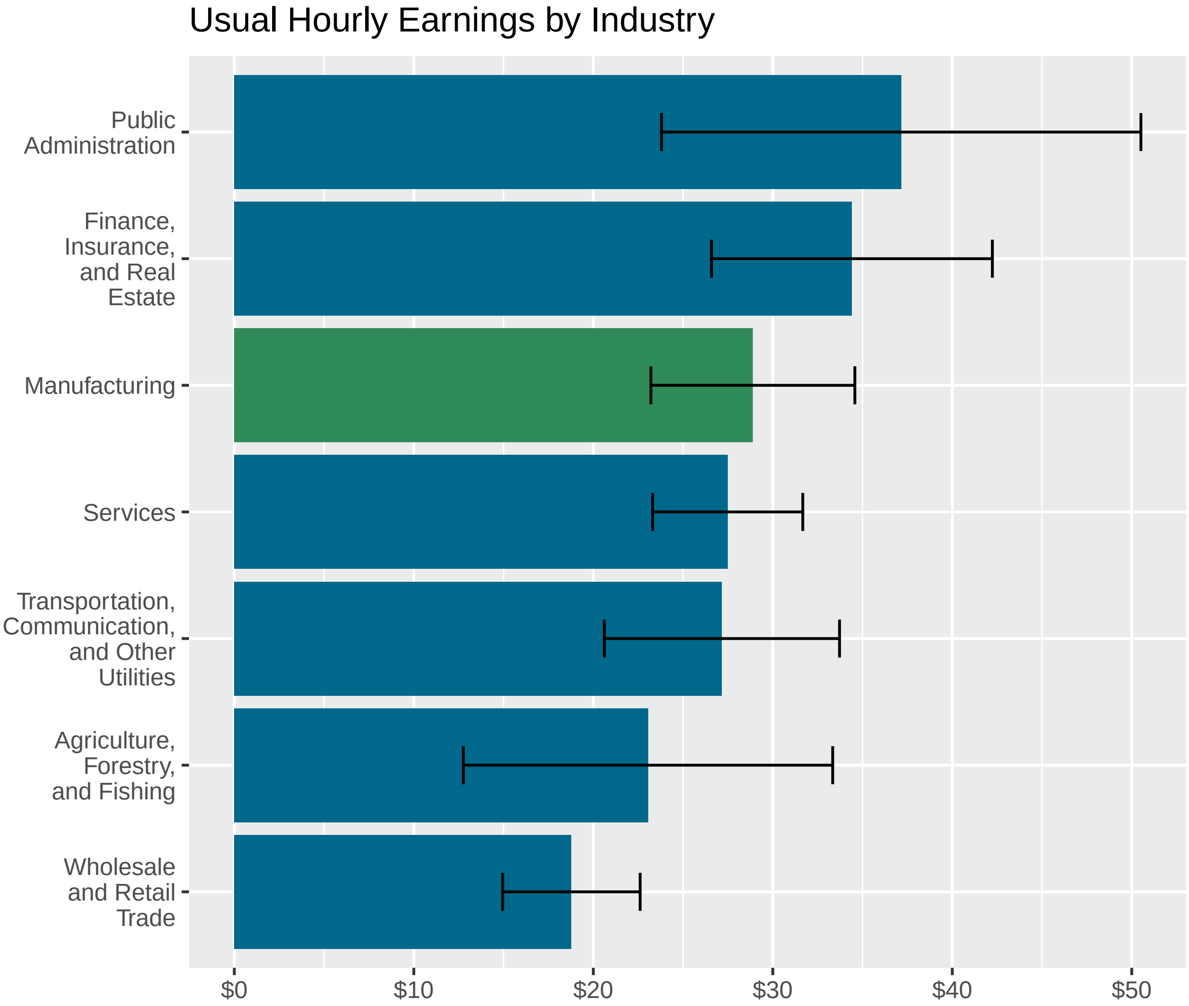
```
bar.error.p <- bar.manu.p + stat_summary(fun.data=mean_cl_normal, geom="errorbar", width=0.3)
```

The line `stat_summary(fun.data=mean_normal_cl, geom="errorbar")` adds another statistics and geometry layer to our existing graph. We set the parameter `fun.data` equal to `mean_cl_normal` so that the `mean_cl_normal()` function is called to compute the confidence interval. The parameter `geom="errorbar"` specifies the error bar geometry. The `width=0.3` parameter sets the width of the lines in the error bar.

Let us view the new bar graph:

```
bar.error.p
```

```
## Warning: Removed 271 rows containing non-finite values (stat_summary).

## Warning: Removed 271 rows containing non-finite values (stat_summary).
```



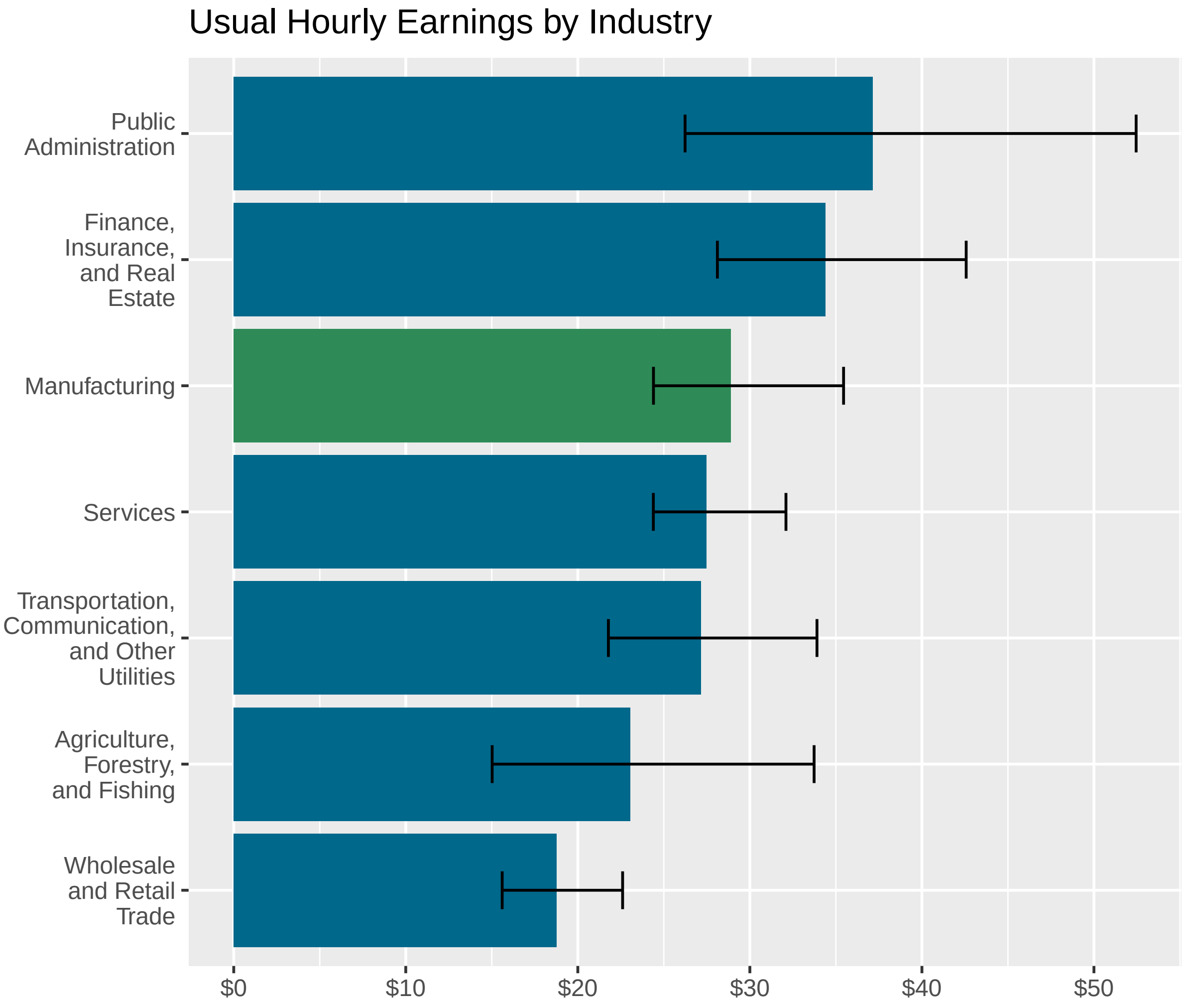
3.2. Confidence Interval with Bootstrapped Errors

It is possible to conduct statistical inference without assuming a normal distribution, especially with the ability for computers to use simulation methods to estimate the appropriate distribution. While these methods are usually beyond the scope of many statistics classes, it is not difficult to conduct this analysis in R with ggplot.

The code below recreates the plot. When specifying the statistics layer, we instead call the function `mean_cl_boot()` which computes a 95% confidence interval based on a **bootstrapped** estimate of the sampling distribution for the mean. Bootstrapping is a simulation method that takes advantage of the computing power of modern computers. The method involves simulating thousands of random samples from our one sample and for every simulated random sample, making note of the statistic of interest (in this case the mean). What results is a distribution for the statistic which can be used to create a confidence interval. The function `mean_cl_boot()` creates 1,000 simulated samples, computes the mean for each, and uses the 2.5 percentile and 97.5 percentile from this distribution of means for the confidence interval.

```
bar.error.p <- bar.manu.p + stat_summary(fun.data=mean_cl_boot, geom="errorbar", width=0.3)
bar.error.p
```

```
## Warning: Removed 271 rows containing non-finite values (stat_summary).

## Warning: Removed 271 rows containing non-finite values (stat_summary).
```

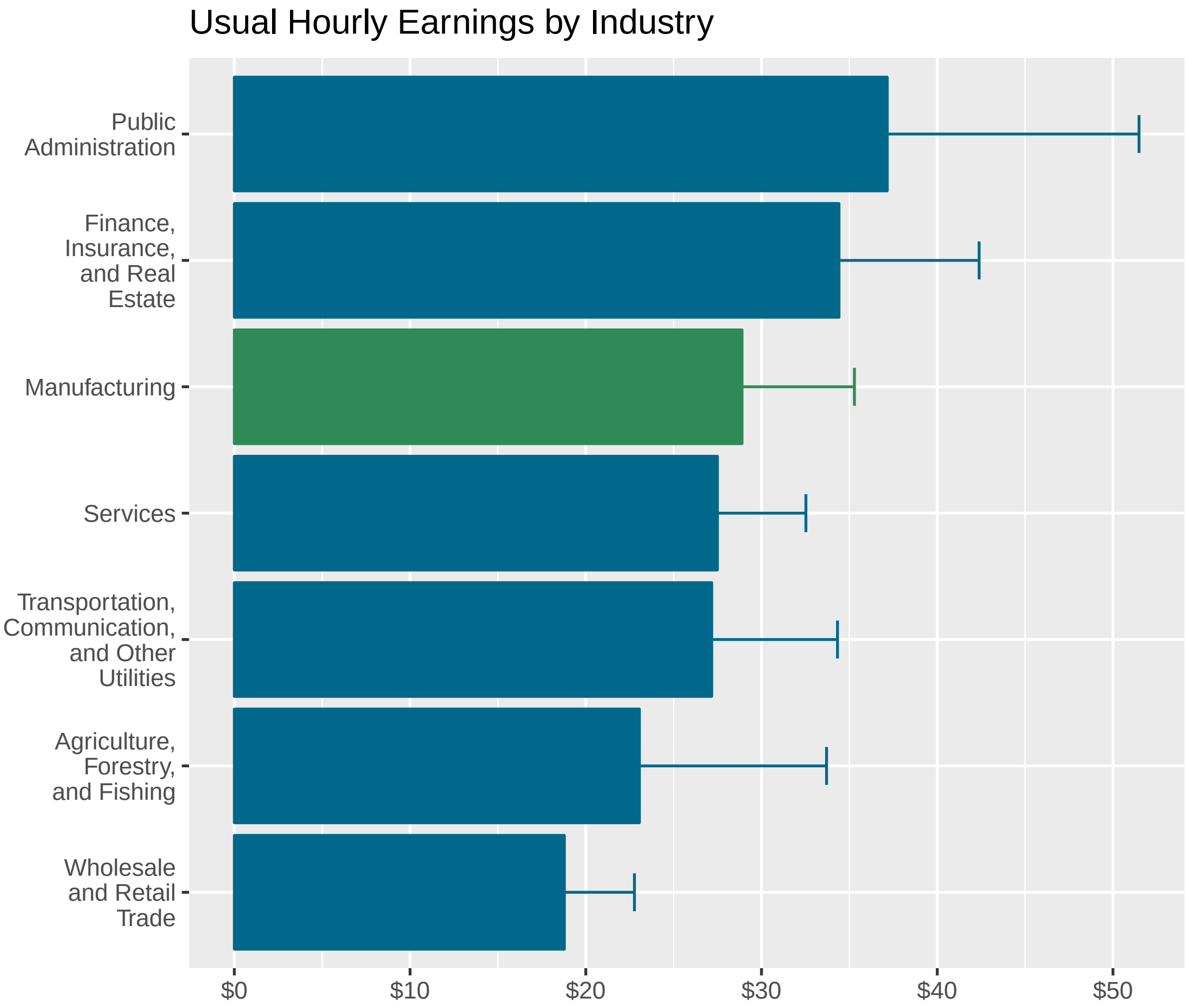


3.3. Set the Color of the Error Bars

One more customization can make our plot more visually appealing. Instead of having black error bars, we can make the color of the bars the same color as the fill. We can do this by adding a `col=manu` mapping to the aesthetics layer. Since we need to change the aesthetics layer, we recreate the entire graph from scratch again and make our final two changes.

```
bar.error.p <- ggplot(data=df, mapping=aes(x=industry, y=usualhrearn, fill=manu, col=manu)) +
  stat_summary(fun.y=mean, geom="bar") +
  stat_summary(fun.data=mean_cl_boot, geom="errorbar", width=0.3) +
  scale_y_continuous(labels=dollar) +
  coord_flip() +
  labs(title="Usual Hourly Earnings by Industry", x="", y="") +
  theme(legend.position="none")  +
  scale_fill_manual(values=mycols) +
  scale_color_manual(values=mycols)

bar.error.p
```

```
## Warning: Removed 271 rows containing non-finite values (stat_summary).

## Warning: Removed 271 rows containing non-finite values (stat_summary).
```



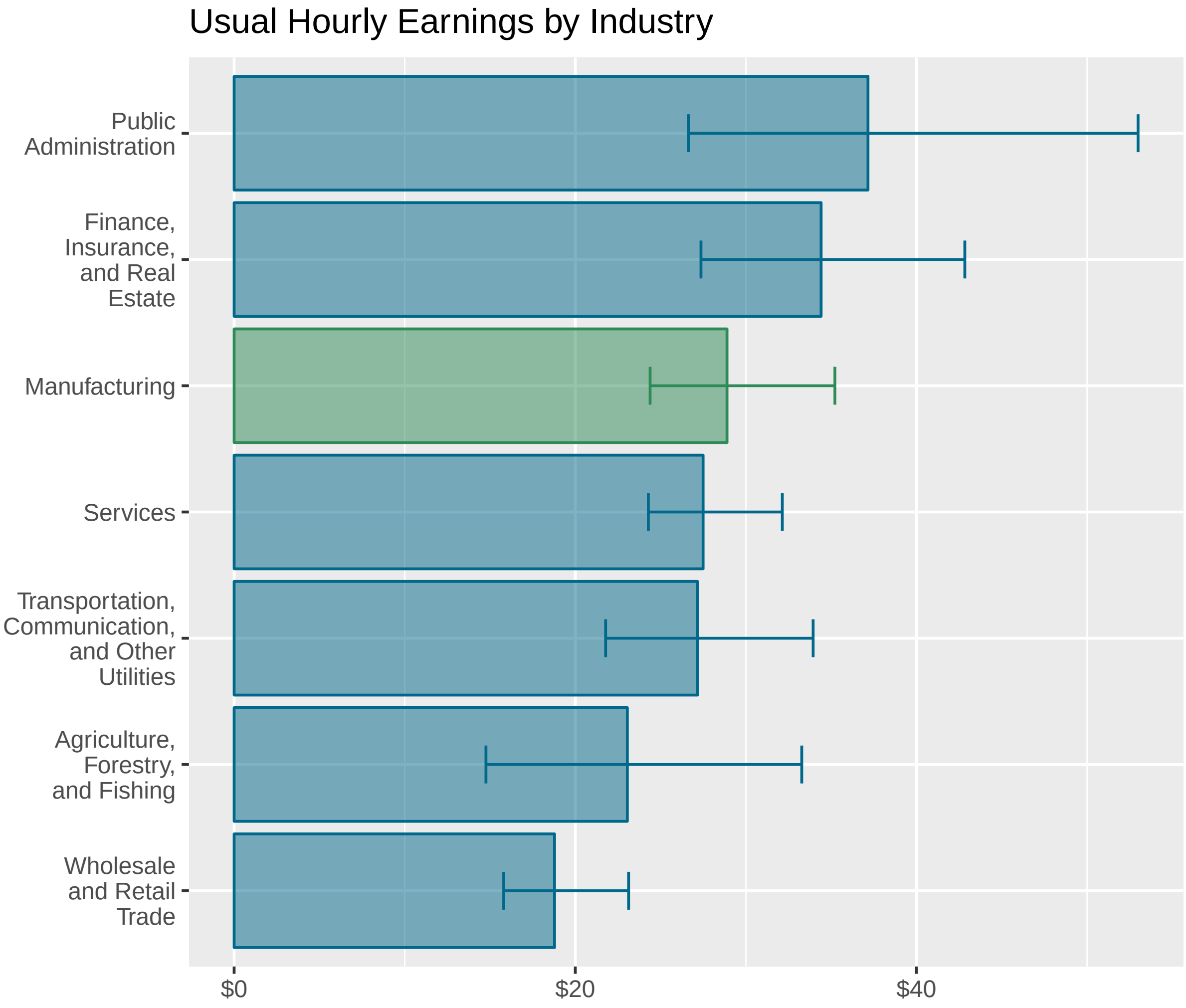
The above code differs from what we have previously done in this tutorial in only two respects. First, the call to `aes()` in the first line now includes the parameter `col=munu`. This maps the color of *lines* (as in, the lines in the confidence whiskers) to manufacturing. Second, we set the color scale for the lines to our custom colors with `scale_color_manual(values=mycols)`. Color for the lines and color for the fill are separate aesthetics, so we also need to set each scale.

Notice that the lower ends of the error bars overlap with the bars, which masks the error bars when they are exactly the same color. So that we can see the error bars and the filled bars "underneath", we will create a little **transparency** in the bars by setting `alpha=0.5` in the `stat_summary()` that creates the bars for the mean.

We again recreate the entire graph from scratch, copying and pasting most of what we have above, but adding the parameter `alpha=0.5` to the call to `stat_summary()` that creates the bars.

```
bar.error.p <- ggplot(data=df, mapping=aes(x=industry, y=usualhrearn, fill=manu, col=manu)) +
  stat_summary(fun.y=mean, geom="bar", alpha=0.5) +
  stat_summary(fun.data=mean_cl_boot, geom="errorbar", width=0.3) +
  scale_y_continuous(labels=dollar) +
  coord_flip() +
  labs(title="Usual Hourly Earnings by Industry", x="", y="") +
  theme(legend.position="none")  +
  scale_fill_manual(values=mycols) +
  scale_color_manual(values=mycols)

bar.error.p
```

```
## Warning: Removed 271 rows containing non-finite values (stat_summary).

## Warning: Removed 271 rows containing non-finite values (stat_summary).
```



3.4. Making Inferences

We can now make inferences based on our most recent plot that includes bars for means and error bars. We can see that many of these confidence bounds overlap, which likely indicates a lack of statistical evidence that the mean usual hourly earnings is different across these different industries. We can see that the confidence bounds for the bottom industry, wholesale and trade, does not overlap with the top four industries, Public Administration, Finance Insurance and Real Estate, Manufacturing, and Services. This indicates there is statistical evidence that workers in the Wholesale and Trade Industry earn less on average than workers in these other industries.

A word of caution regarding making statistical inferences based on graphical visualizations: The confidence bounds in the plots are based on univariate (single-variable) confidence intervals, not on hypothesis tests or confidence intervals on the *difference* (two-variables) between means. When the categories are independent (different observations in different groups), non-overlapping confidence intervals does indicate statistically significant differences in means. When independent categories do have univariate confidence intervals overlap, but may be close together, there still may be enough statistical evidence for differences in the means. These questions can be answered by conducting the appropriate bivariate hypothesis tests.

4. Multiple Categorical Variables

In past sections we created visualizations that communicated usual hourly earnings by a single categorical variable, industry. In this section we expand our bar plot to include information on how our numerical dependent variable, usual hourly earnings, depends on multiple categorical variables. In the examples below, we look at how usual hourly earnings depends on `industry` and `sex`.

In the code below we create a vertical bar plot in which we map usual hourly earnings to the vertical axis, industry to the horizontal axis, sex to the fill color of the bar, and sex again to the color of lines (will be used for error bars).

```
bar.two.base.p <- ggplot(data=df, mapping=aes(x=industry, y=usualhrearn, fill=sex, col=sex))
```

Let us add to the base plot the many visual improvements we have made above, including expressing the numbers on the vertical axis in dollars, rotating the text labels on the horizontal axis, creating a title, and clearing the axes titles.

```
bar.two.base.p <- bar.two.base.p +
  scale_y_continuous(labels=dollar) +
  labs(title="Usual Hourly Earnings by Industry", x="", y="")
```

4.1 Multiple Bars

We still do not have a visualization to show because we have not added the statistics or geometry layers. In the next step, we call `stat_summary()` as before, but add a value for the `position` parameter which tells ggplot that for each industry (the x aesthetic), make multiple bars that do not overlap (they dodge each other). This will create two bars for each industry, one for each possible outcome of the color aesthetic (male and female).

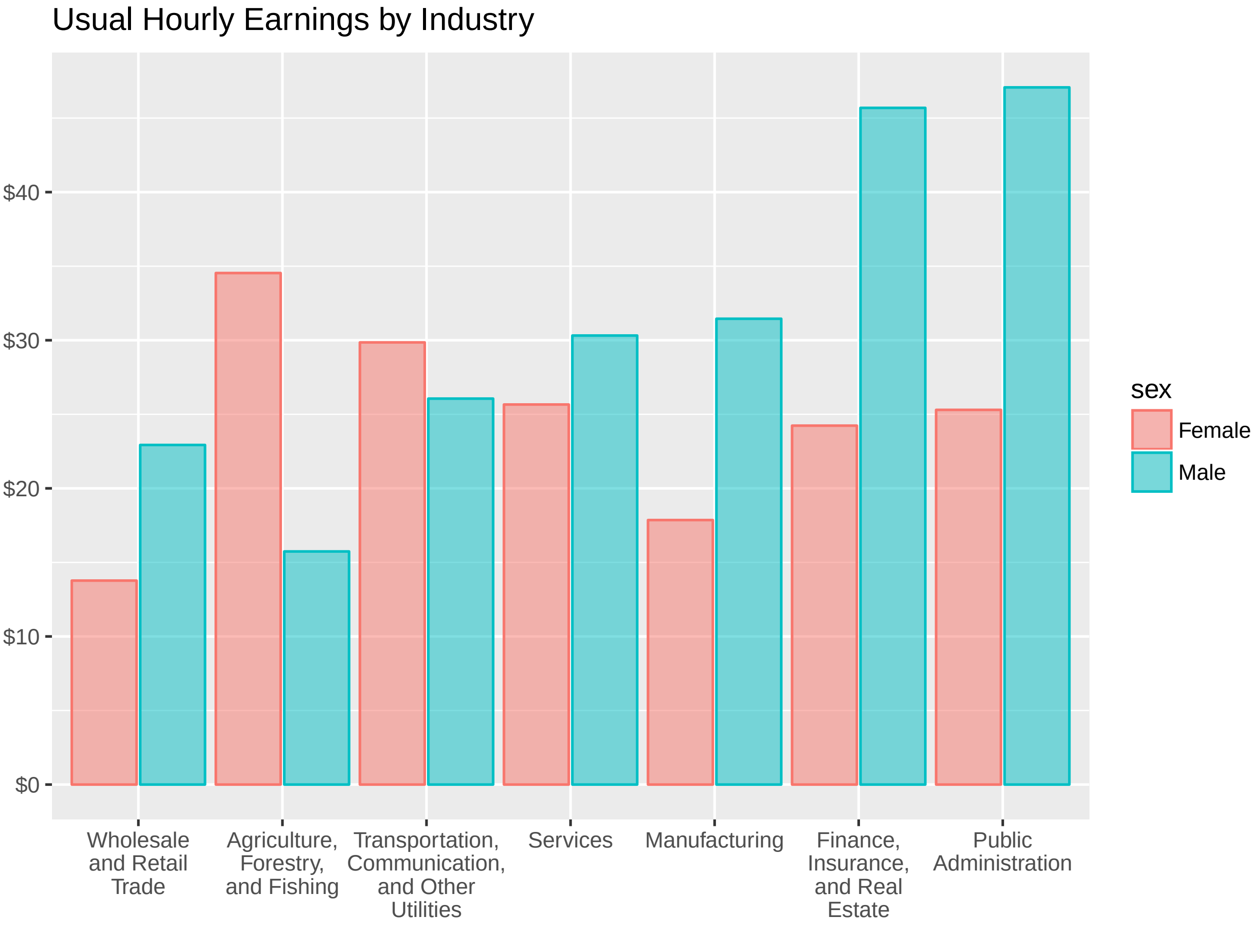
First we create a position object called `posn.d` using the `position_dodge()` function which tells ggplot how far apart to separate the centers of the geometries. With some experimentation, one can find a `width=0.95` creates bars that do not overlap and do not have space in between.

```
posn.d <- position_dodge(width=0.95)
```

Next we create the plot, adding to the base plot we created above and saving the new plot in an object called `base.two.p`.

```
base.two.p <- bar.two.base.p +
  stat_summary(fun.data=mean_sdl, geom="bar", position=posn.d, alpha=0.5)
base.two.p
```

```
## Warning: Removed 271 rows containing non-finite values (stat_summary).
```

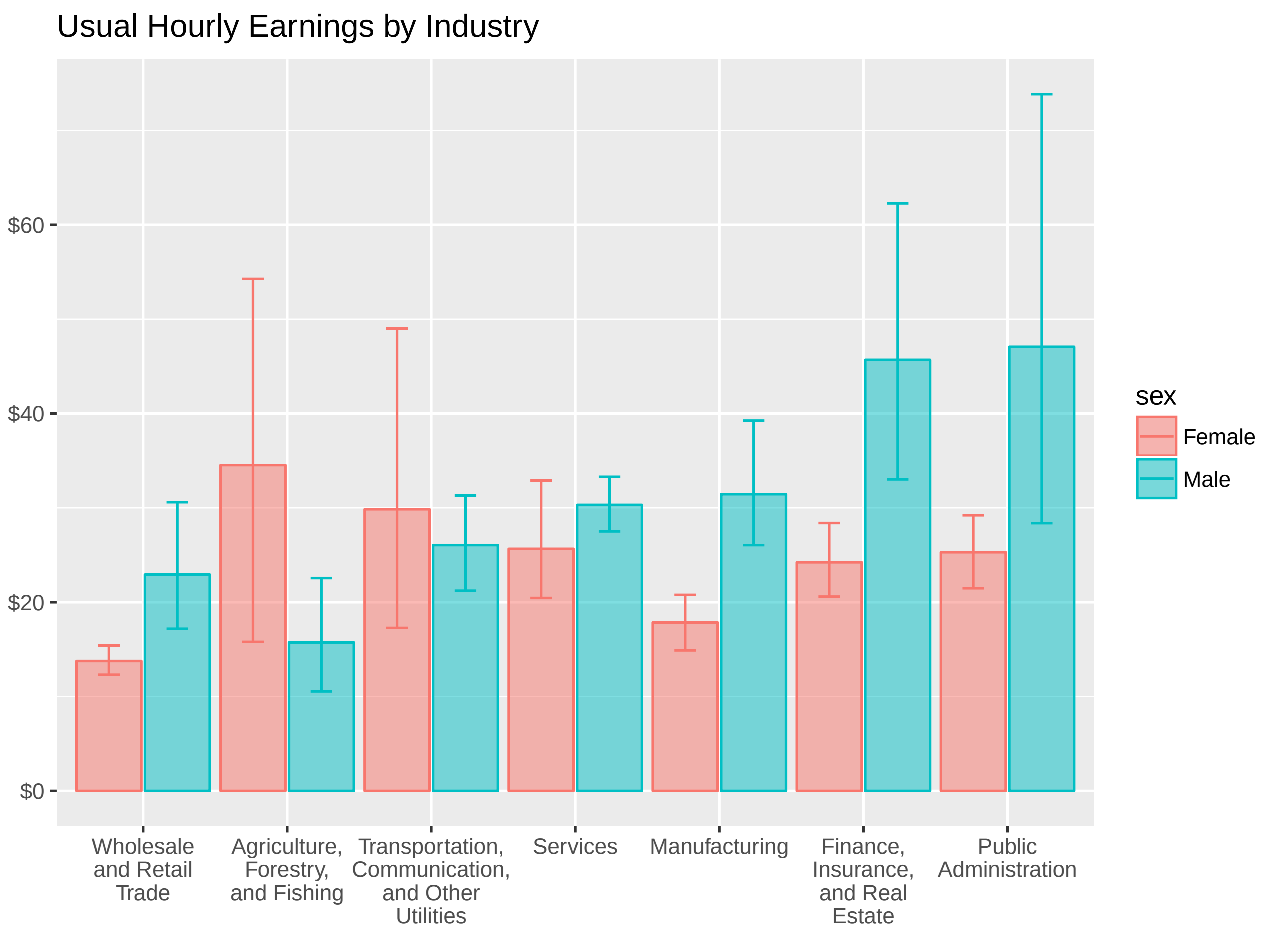


Besides the position parameter in our call to stat_summary() we also specify the height of the bar to be equal to the mean summary statistic using the parameter fun.data=mean_sdl and geometry to be a bar with the geom="bar" parameter. Furthermore, we give the bars some transparency using alpha=0.5 in anticipation of drawing error bars later and wanting to be able to see both the bars and the error bars where they overlap.

We can add our error bars as before.

```
base.two.p <- base.two.p +
  stat_summary(fun.data=mean_cl_boot, geom="errorbar", position=posn.d, width=0.3)
base.two.p
```

```
## Warning: Removed 271 rows containing non-finite values (stat_summary).

## Warning: Removed 271 rows containing non-finite values (stat_summary).
```



4.2 Multiple Facets

Instead of separating males and females by color, we can create multiple facets with the **facets layer**, one for each sex. Facets are multiple plots, usually with a matching scale, placed next to each other for comparison.

Before we add facets, let us first build our base plot. Read the code that follows and assure yourself that you understand every line.

```
# Set a longer length string for the industries, 20 characters
levels(df$industry) <- str_wrap( levels(df$industry), width=30 )

# Create Grid Plot
bar.grid.base.p <-
  ggplot(data=df, mapping=aes(x=industry, y=usualhrearn, fill=manu, col=manu)) +
  stat_summary(fun.y=mean, geom="bar", alpha=0.5) +
  stat_summary(fun.data=mean_cl_boot, geom="errorbar", width=0.3) +
  labs(title="Usual Hourly Earnings by Industry", x="", y="") +
  theme(legend.position="none") +
  coord_flip() +
  scale_y_continuous(labels=dollar) +
  scale_fill_manual(values=mycols) +
  scale_color_manual(values=mycols)
```

The first line of the `ggplot()` call above sets the data and aesthetics layer. In the aesthetics layer, we map the x-axis to `industry`, y-axis to `usualhrearn`, and to highlight manufacturing, we map both fill color and line color to `manu`.

In the second and third lines of code we add statistics and geometry layers to create bars to visualize the means, and error bars to visualize bootstrapped confidence intervals, respectively. We set some transparency in the bars so that we can see both the error bars and mean bars when they overlap.

In the forth line we set the title and erase the titles for the horizontal and vertical axes.

In the fifth line we use the theme layer to turn off the legend, as we do not need to have a legend explaining the color for the manufacturing bar.

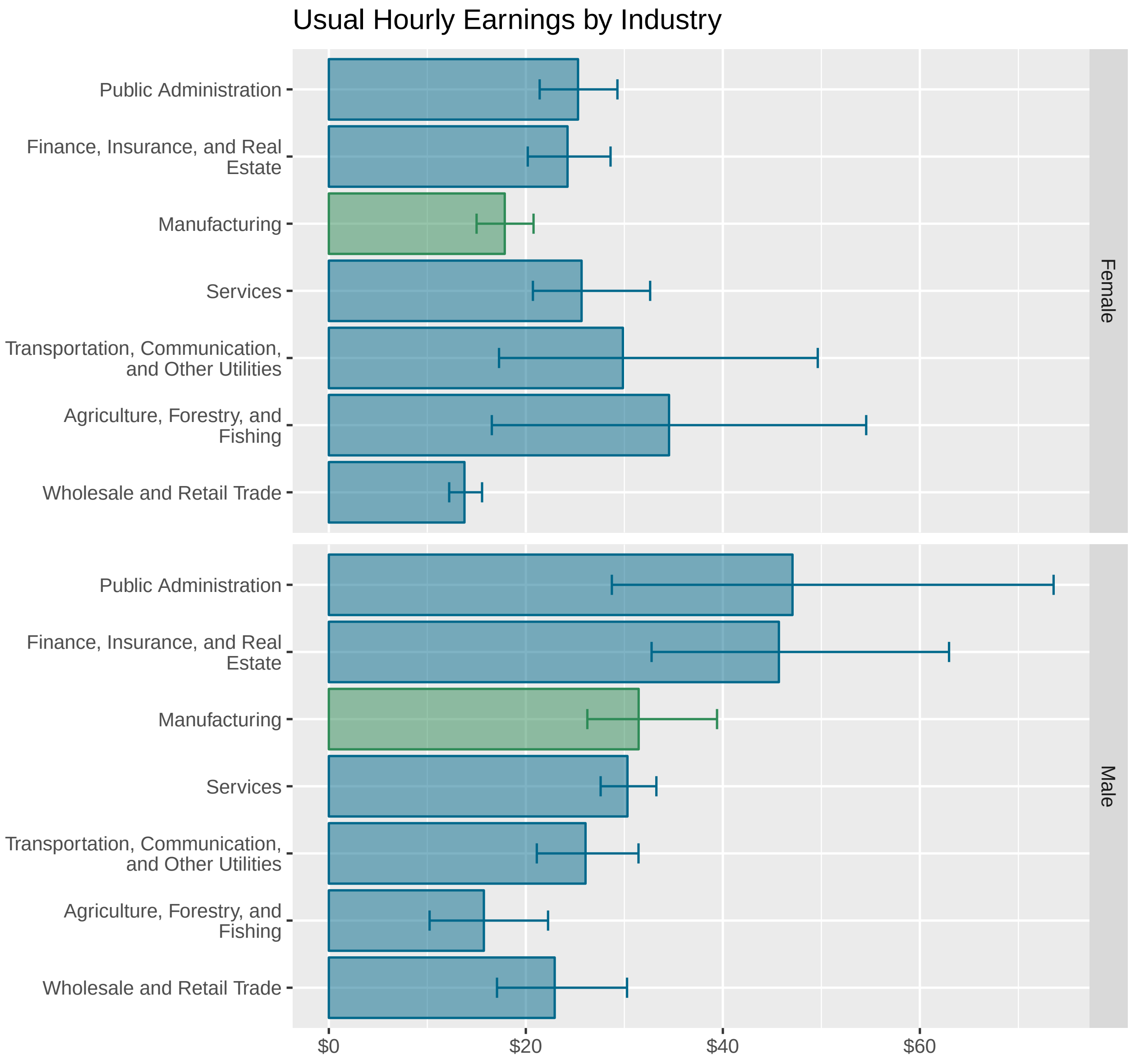
In the sixth line we flip the coordinates so that we have horizontal bars.

In the last three lines we manipulate our scales. In the seventh line, we set the labels for the y-axis scale to be expressed in dollars. In the eighth line we set the color scale for the fill color of the bars equal to our custom colors. In the ninth and final line we set the color scale for the line color equal to our same custom color scale.

We can add a facets layer with a call to `facet_grid()` which will create a grid of graphs based on different levels of categorical variables. The function expects a *formula* in the form `row_factor ~ col_factor`. Here we finish our grid bar plot

```r
bar.grid.p <- bar.grid.base.p + facet_grid(sex~.)
bar.grid.p
```

```
## Warning: Removed 271 rows containing non-finite values (stat_summary).

## Warning: Removed 271 rows containing non-finite values (stat_summary).
```

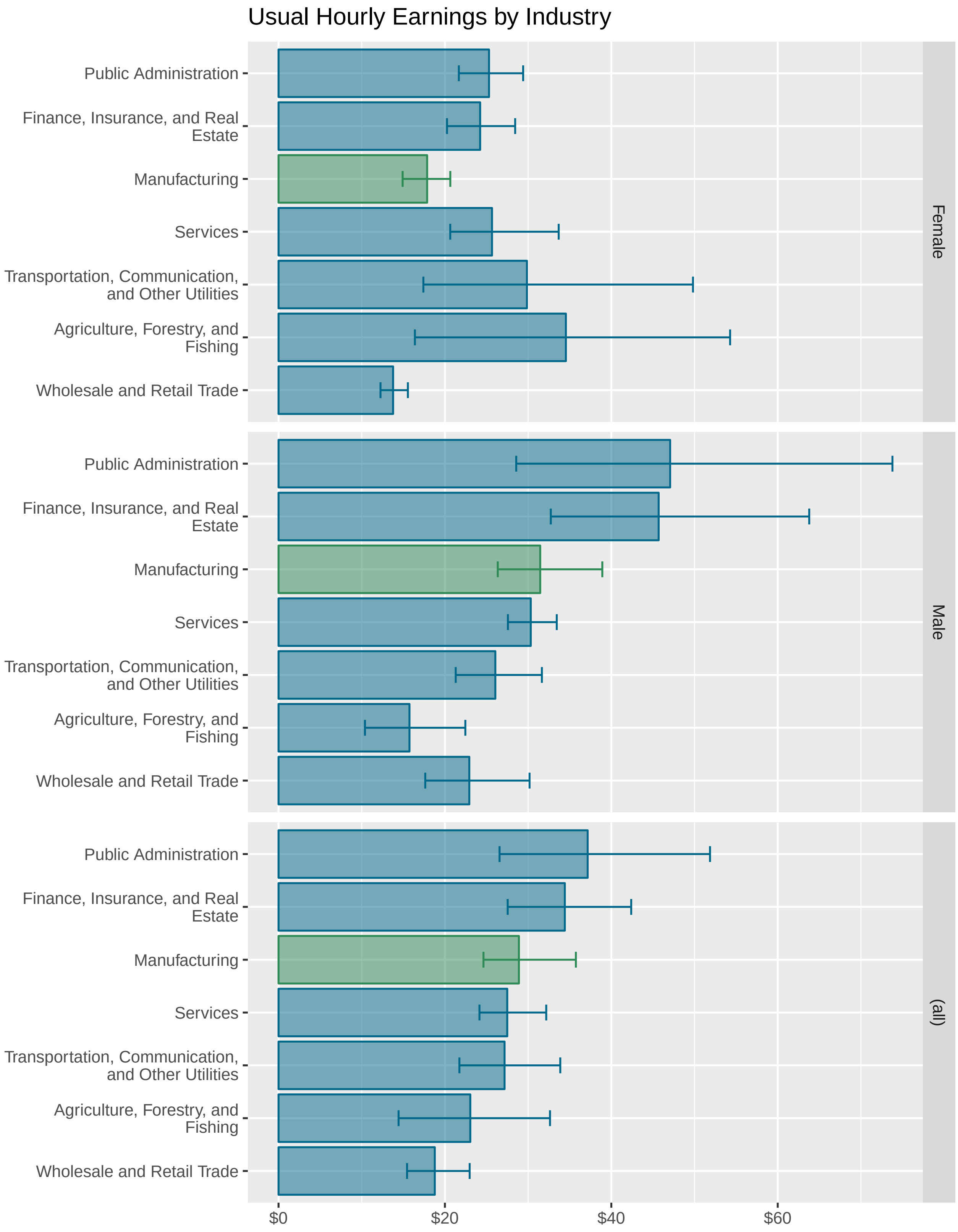


The call `facet_grid(sex~.)` sets the row variable equal to sex and no column variable. With two levels for sex (male and female), we have two rows of bar plots, one for males and another for females. Note that the top and bottom bar plots have exactly the same scale which is presented only once on the bottom.

If you want to display bar plots for the levels men and women and for men and women overall, add the `margins=TRUE` parameter to the `facet_grid()` function call.

```
bar.grid.base.p + facet_grid(sex~., margins=TRUE)
```

```
## Warning: Removed 542 rows containing non-finite values (stat_summary).

## Warning: Removed 542 rows containing non-finite values (stat_summary).
```


Usual Hourly Earnings by Industry
Public Administration
Finance, Insurance, and Real Estate
Manufacturing
Services
Transportation, Communication, and Other Utilities
Agriculture, Forestry, and Fishing
Wholesale and Retail Trade
Female
Male
(all)
$0
$20
$40
$60

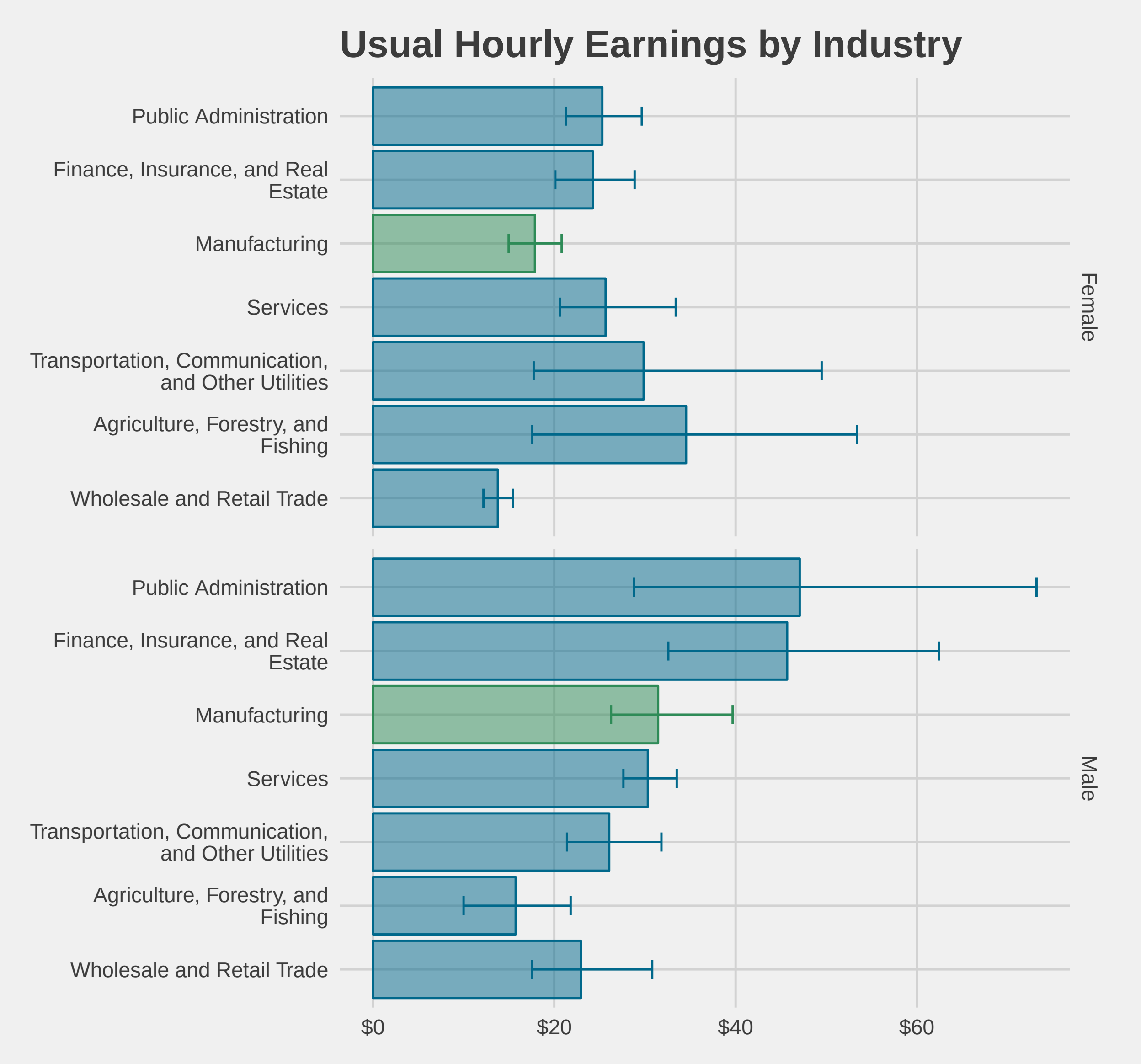
4.3. Pretty Themes

Everything looks prettier with a nice theme. You can view several available predefined themes at: https://cran.r-project.org/web/packages/ggthemes/vignettes/ggthemes.html

How about a plot using a look and feel resembling http://fivethirtyeight.com/?

```
bar.grid.p + theme_fivethirtyeight() + theme(legend.position="none")
```

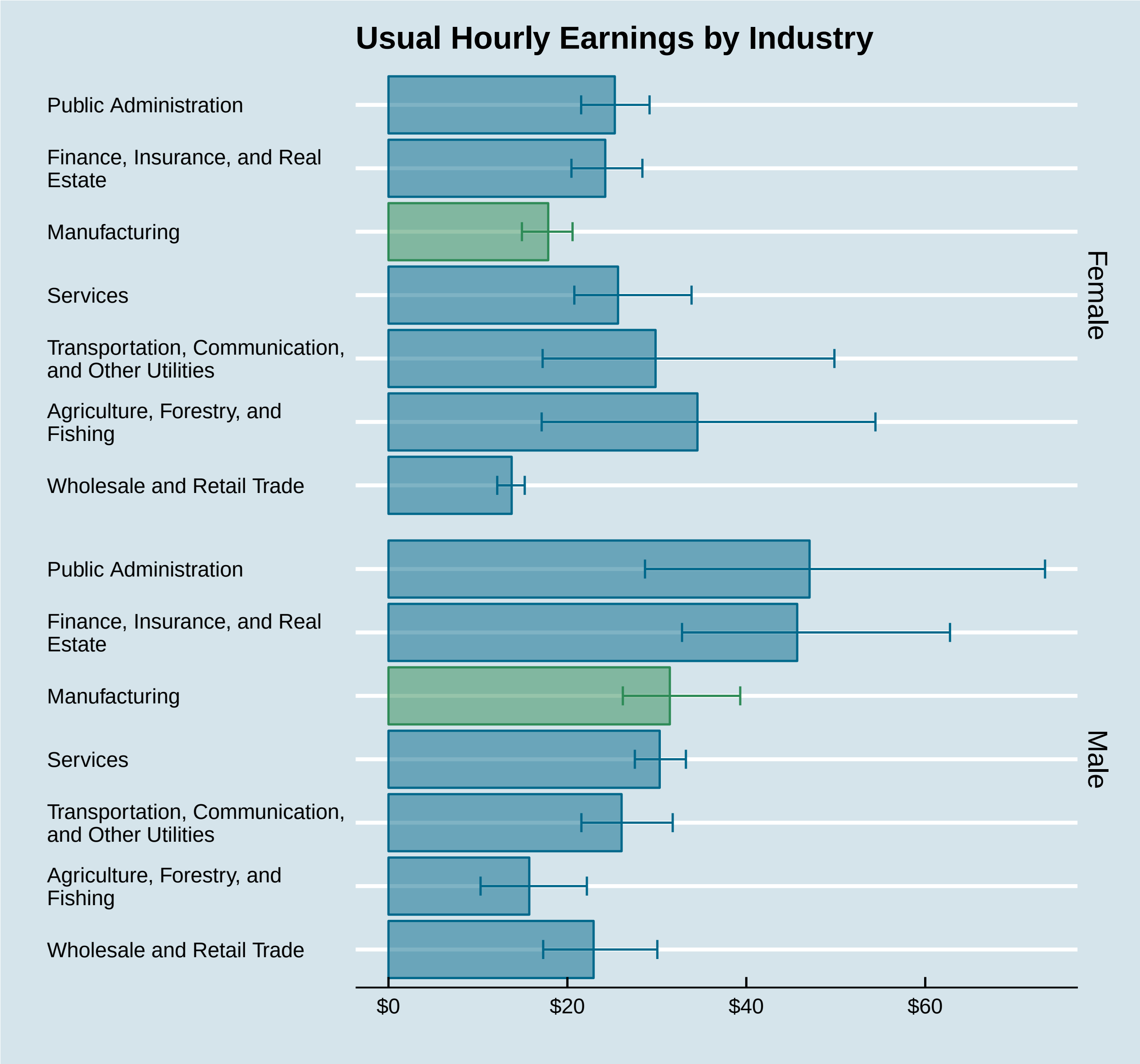
```
## Warning: Removed 271 rows containing non-finite values (stat_summary).

## Warning: Removed 271 rows containing non-finite values (stat_summary).
```



Maybe you like the look and feel of the *Economist* https://www.economist.com/?

```
bar.grid.p + theme_economist() + theme(legend.position="none")
```

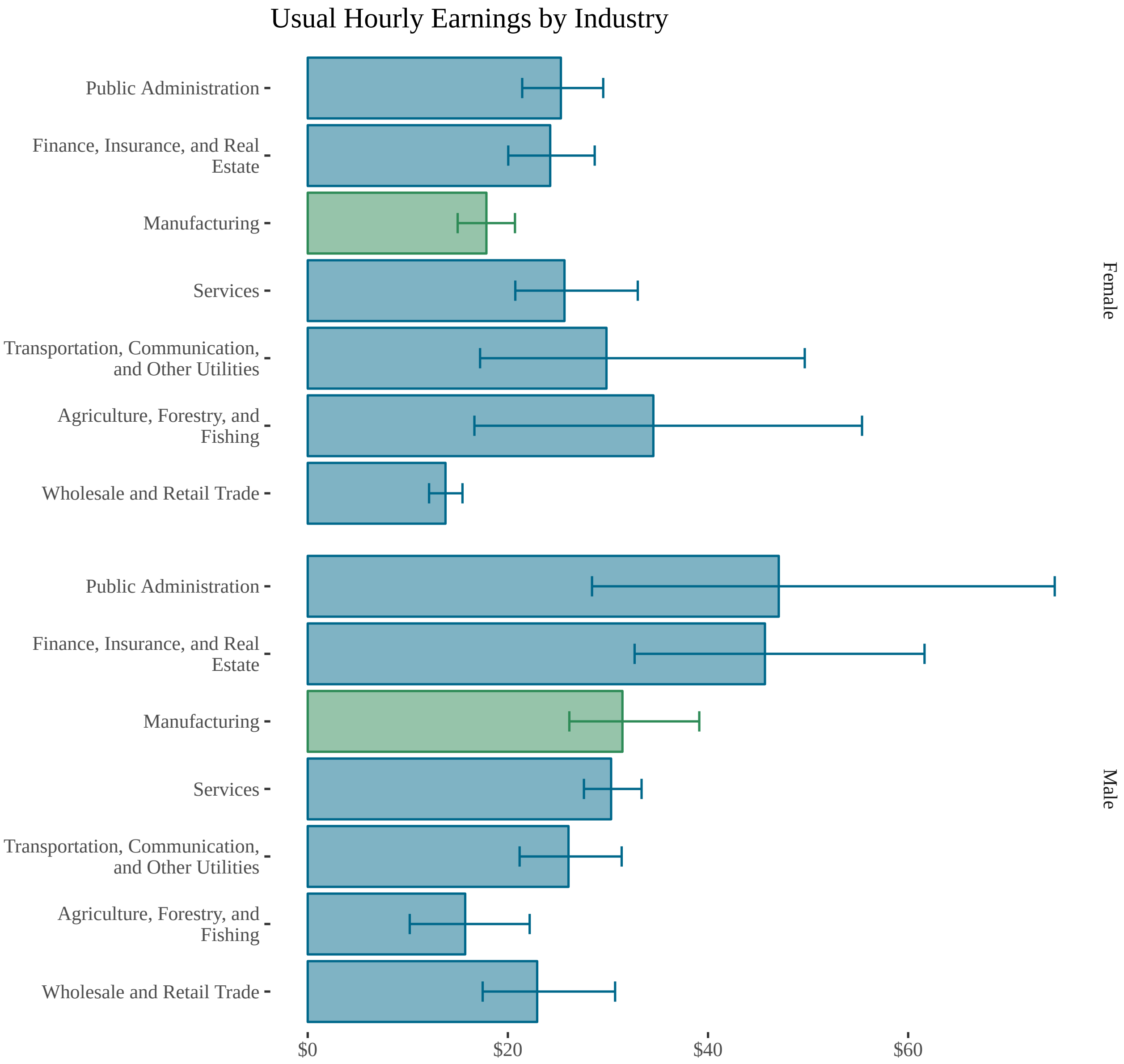
```
## Warning: Removed 271 rows containing non-finite values (stat_summary).
## Warning: Removed 271 rows containing non-finite values (stat_summary).
```



Maybe you want a more simple theme?

```
bar.grid.p + theme_tufte() + theme(legend.position="none")
```

```
## Warning: Removed 271 rows containing non-finite values (stat_summary).

## Warning: Removed 271 rows containing non-finite values (stat_summary).
```



5 Exercises

Use the data frame in this tutorial and create bar charts as described in the following problems.

1. Create a bar chart illustrating average annual income by education level. Create the plot so that the vertical scale is expressed in dollars. Also give your plot a title, no labels for the horizontal or vertical axes, and make it so that the horizontal labels do not overlap.
2. Recreate the plot in (1) and add bootstrapped error bars to illustrate the margins of error associated with each mean. Also, flip the coordinates so that the bars are horizontal and education is on the vertical axes.
3. Based on the plot in (2), what can you say about statistical evidence for differences in annual income between different levels of education?

4. Use the plot you created in (2) and create bar plots illustrating annual wage and salary income by sex. Differentiate sex by the fill color of the bar, and put male and female bars next to each other for each level of education.

5. Based on your plot in (4), highlight the levels of educational attainment where there is statistical evidence for differences in wage and salary income by sex.

6. Recreate the plot in (2) and highlight the bar associated with the baccalaureate degree with a different color. Do not show a legend.

7. Use the plot you created in (6) and create a grid of plots illustrating annual wage and salary income by sex and the overall.