Chapter 4 Notes

Table of contents

# Chapter 4: Exploring US Census data with visualization

* **ggplot2** by Hadley Wickham is the core visualization framework for **tidyverse**
* Chapter goes over the following chart types:
  + faceted (“small multiples”)
  + population pyramids
  + Margin of Error plots
* Chapter goes over **plotly** for interactive vizualization

```{r setup}  
library(extrafont)  
```

Registering fonts with R

```{r setup}  
library(tidycensus)  
library(tidyverse)  
```

── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
✔ dplyr 1.1.2 ✔ readr 2.1.4  
✔ forcats 1.0.0 ✔ stringr 1.5.0  
✔ ggplot2 3.4.2 ✔ tibble 3.2.1  
✔ lubridate 1.9.2 ✔ tidyr 1.3.0  
✔ purrr 1.0.2

── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
✖ dplyr::filter() masks stats::filter()  
✖ dplyr::lag() masks stats::lag()  
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

```{r setup}  
library(scales)  
```

Attaching package: 'scales'  
  
The following object is masked from 'package:purrr':  
  
 discard  
  
The following object is masked from 'package:readr':  
  
 col\_factor

```{r setup}  
loadfonts(device = 'win')  
```

Agency FB already registered with windowsFonts().  
Aladin already registered with windowsFonts().  
Algerian already registered with windowsFonts().  
AquilineTwo already registered with windowsFonts().  
Arial Black already registered with windowsFonts().  
Arial already registered with windowsFonts().  
Arial Narrow already registered with windowsFonts().  
Arial Rounded MT Bold already registered with windowsFonts().  
Avalon Quest already registered with windowsFonts().  
Bahnschrift already registered with windowsFonts().  
Baskerville Old Face already registered with windowsFonts().  
Bauhaus 93 already registered with windowsFonts().  
Bell MT already registered with windowsFonts().  
Berlin Sans FB already registered with windowsFonts().  
Berlin Sans FB Demi already registered with windowsFonts().  
Bernard MT Condensed already registered with windowsFonts().  
Blackadder ITC already registered with windowsFonts().  
Blackout already registered with windowsFonts().  
Blackout 2AM already registered with windowsFonts().  
Blackout Midnight already registered with windowsFonts().  
Blackout Sunrise already registered with windowsFonts().  
Bodoni MT already registered with windowsFonts().  
Bodoni MT Black already registered with windowsFonts().  
Bodoni MT Condensed already registered with windowsFonts().  
Bodoni MT Poster Compressed already registered with windowsFonts().  
Book Antiqua already registered with windowsFonts().  
Bookman Old Style already registered with windowsFonts().  
Bookshelf Symbol 7 already registered with windowsFonts().  
Bradley Hand ITC already registered with windowsFonts().  
Bridgnorth already registered with windowsFonts().  
Britannic Bold already registered with windowsFonts().  
Broadway already registered with windowsFonts().  
Brush Script MT already registered with windowsFonts().  
Calibri already registered with windowsFonts().  
Calibri Light already registered with windowsFonts().  
Californian FB already registered with windowsFonts().  
Calisto MT already registered with windowsFonts().  
Cambria already registered with windowsFonts().  
Candara already registered with windowsFonts().  
Candara Light already registered with windowsFonts().  
Cascadia Code already registered with windowsFonts().  
Cascadia Mono already registered with windowsFonts().  
Castellar already registered with windowsFonts().  
Centaur already registered with windowsFonts().  
Century already registered with windowsFonts().  
Century Gothic already registered with windowsFonts().  
Century Schoolbook already registered with windowsFonts().  
Chiller already registered with windowsFonts().  
ChunkFive already registered with windowsFonts().  
Colonna MT already registered with windowsFonts().  
Comic Sans MS already registered with windowsFonts().  
Consolas already registered with windowsFonts().  
Constantia already registered with windowsFonts().  
Cooper Black already registered with windowsFonts().  
Copperplate Gothic Bold already registered with windowsFonts().  
Copperplate Gothic Light already registered with windowsFonts().  
Corbel already registered with windowsFonts().  
Corbel Light already registered with windowsFonts().  
Courier New already registered with windowsFonts().  
Curlz MT already registered with windowsFonts().  
Dubai already registered with windowsFonts().  
Dubai Light already registered with windowsFonts().  
Dubai Medium already registered with windowsFonts().  
Ebrima already registered with windowsFonts().  
Edwardian Script ITC already registered with windowsFonts().  
Elephant already registered with windowsFonts().  
Engravers MT already registered with windowsFonts().  
Eras Bold ITC already registered with windowsFonts().  
Eras Demi ITC already registered with windowsFonts().  
Eras Light ITC already registered with windowsFonts().  
Eras Medium ITC already registered with windowsFonts().  
Fanwood already registered with windowsFonts().  
Fanwood Text already registered with windowsFonts().  
Far East already registered with windowsFonts().  
Felix Titling already registered with windowsFonts().  
Footlight MT Light already registered with windowsFonts().  
Forte already registered with windowsFonts().  
Franklin Gothic Book already registered with windowsFonts().  
Franklin Gothic Demi already registered with windowsFonts().  
Franklin Gothic Demi Cond already registered with windowsFonts().  
Franklin Gothic Heavy already registered with windowsFonts().  
Franklin Gothic Medium already registered with windowsFonts().  
Franklin Gothic Medium Cond already registered with windowsFonts().  
Freestyle Script already registered with windowsFonts().  
French Script MT already registered with windowsFonts().  
Gabriola already registered with windowsFonts().  
Gadugi already registered with windowsFonts().  
Gaeilge 1 already registered with windowsFonts().  
Garamond already registered with windowsFonts().  
Georgia already registered with windowsFonts().  
Gigi already registered with windowsFonts().  
Gill Sans Ultra Bold already registered with windowsFonts().  
Gill Sans Ultra Bold Condensed already registered with windowsFonts().  
Gill Sans MT already registered with windowsFonts().  
Gill Sans MT Condensed already registered with windowsFonts().  
Gill Sans MT Ext Condensed Bold already registered with windowsFonts().  
Gloucester MT Extra Condensed already registered with windowsFonts().  
Goudy Bookletter 1911 already registered with windowsFonts().  
GoudyMedieval already registered with windowsFonts().  
Goudy Old Style already registered with windowsFonts().  
Sorts Mill Goudy already registered with windowsFonts().  
Goudy Stout already registered with windowsFonts().  
Haettenschweiler already registered with windowsFonts().  
Harlow Solid Italic already registered with windowsFonts().  
Harrington already registered with windowsFonts().  
High Tower Text already registered with windowsFonts().  
HoloLens MDL2 Assets already registered with windowsFonts().  
IM FELL English already registered with windowsFonts().  
Impact already registered with windowsFonts().  
Imprint MT Shadow already registered with windowsFonts().  
Informal Roman already registered with windowsFonts().  
Ink Free already registered with windowsFonts().  
Javanese Text already registered with windowsFonts().  
Jokerman already registered with windowsFonts().  
Juice ITC already registered with windowsFonts().  
Junction Bold already registered with windowsFonts().  
Junction Light already registered with windowsFonts().  
Junction Regular already registered with windowsFonts().  
Knewave already registered with windowsFonts().  
Knewave Outline already registered with windowsFonts().  
Kristen ITC already registered with windowsFonts().  
Kunstler Script already registered with windowsFonts().  
Wide Latin already registered with windowsFonts().  
League Script Thin already registered with windowsFonts().  
League Spartan Black already registered with windowsFonts().  
League Spartan already registered with windowsFonts().  
League Spartan ExtraBold already registered with windowsFonts().  
League Spartan ExtraLight already registered with windowsFonts().  
League Spartan Light already registered with windowsFonts().  
League Spartan Medium already registered with windowsFonts().  
League Spartan SemiBold already registered with windowsFonts().  
Leelawadee already registered with windowsFonts().  
Leelawadee UI already registered with windowsFonts().  
Leelawadee UI Semilight already registered with windowsFonts().  
Linden Hill already registered with windowsFonts().  
Lucida Bright already registered with windowsFonts().  
Lucida Calligraphy already registered with windowsFonts().  
Lucida Console already registered with windowsFonts().  
Lucida Fax already registered with windowsFonts().  
Lucida Handwriting already registered with windowsFonts().  
Lucida Sans already registered with windowsFonts().  
Lucida Sans Typewriter already registered with windowsFonts().  
Lucida Sans Unicode already registered with windowsFonts().  
Magneto already registered with windowsFonts().  
Maiandra GD already registered with windowsFonts().  
Malgun Gothic already registered with windowsFonts().  
Malgun Gothic Semilight already registered with windowsFonts().  
Marlett already registered with windowsFonts().  
MasonSerifAltBold already registered with windowsFonts().  
MasonSerifAltReg already registered with windowsFonts().  
MasonSerifBold already registered with windowsFonts().  
MasonSerifRegular already registered with windowsFonts().  
MasonSerifSuper already registered with windowsFonts().  
MasonSerifSuperBold already registered with windowsFonts().  
Matura MT Script Capitals already registered with windowsFonts().  
Microsoft Himalaya already registered with windowsFonts().  
Microsoft Yi Baiti already registered with windowsFonts().  
Microsoft New Tai Lue already registered with windowsFonts().  
Microsoft PhagsPa already registered with windowsFonts().  
Microsoft Sans Serif already registered with windowsFonts().  
Microsoft Tai Le already registered with windowsFonts().  
Microsoft Uighur already registered with windowsFonts().  
Middle Ages already registered with windowsFonts().  
Mistral already registered with windowsFonts().  
Modern No. 20 already registered with windowsFonts().  
Mongolian Baiti already registered with windowsFonts().  
Monotype Corsiva already registered with windowsFonts().  
MS Outlook already registered with windowsFonts().  
MS Reference Sans Serif already registered with windowsFonts().  
MS Reference Specialty already registered with windowsFonts().  
MT Extra already registered with windowsFonts().  
MV Boli already registered with windowsFonts().  
Myanmar Text already registered with windowsFonts().  
Niagara Engraved already registered with windowsFonts().  
Niagara Solid already registered with windowsFonts().  
Nirmala UI already registered with windowsFonts().  
Nirmala UI Semilight already registered with windowsFonts().  
OCR A Extended already registered with windowsFonts().  
OFL Sorts Mill Goudy TT already registered with windowsFonts().  
Old English Text MT already registered with windowsFonts().  
Onyx already registered with windowsFonts().  
Orbitron already registered with windowsFonts().  
Ostrich Sans already registered with windowsFonts().  
Ostrich Sans Inline already registered with windowsFonts().  
Ostrich Sans Rounded already registered with windowsFonts().  
Palace Script MT already registered with windowsFonts().  
Palatino Linotype already registered with windowsFonts().  
Papyrus already registered with windowsFonts().  
Parchment already registered with windowsFonts().  
Perpetua already registered with windowsFonts().  
Perpetua Titling MT already registered with windowsFonts().  
Playbill already registered with windowsFonts().  
Poor Richard already registered with windowsFonts().  
Pristina already registered with windowsFonts().  
Prociono already registered with windowsFonts().  
Prociono TT already registered with windowsFonts().  
Rage Italic already registered with windowsFonts().  
Raleway Black already registered with windowsFonts().  
Raleway already registered with windowsFonts().  
Raleway ExtraBold already registered with windowsFonts().  
Raleway ExtraLight already registered with windowsFonts().  
Raleway Light already registered with windowsFonts().  
Raleway Medium already registered with windowsFonts().  
Raleway SemiBold already registered with windowsFonts().  
Raleway Thin already registered with windowsFonts().  
Ravie already registered with windowsFonts().  
Rockwell already registered with windowsFonts().  
Rockwell Condensed already registered with windowsFonts().  
Rockwell Extra Bold already registered with windowsFonts().  
Sans Serif Collection already registered with windowsFonts().  
Script MT Bold already registered with windowsFonts().  
Segoe Fluent Icons already registered with windowsFonts().  
Segoe MDL2 Assets already registered with windowsFonts().  
Segoe Print already registered with windowsFonts().  
Segoe Script already registered with windowsFonts().  
Segoe UI already registered with windowsFonts().  
Segoe UI Light already registered with windowsFonts().  
Segoe UI Semibold already registered with windowsFonts().  
Segoe UI Semilight already registered with windowsFonts().  
Segoe UI Black already registered with windowsFonts().  
Segoe UI Historic already registered with windowsFonts().  
Segoe UI Symbol already registered with windowsFonts().  
Segoe UI Variable already registered with windowsFonts().  
Shanghai already registered with windowsFonts().  
Showcard Gothic already registered with windowsFonts().  
SimSun-ExtB already registered with windowsFonts().  
Sitka Text already registered with windowsFonts().  
Snap ITC already registered with windowsFonts().  
Sniglet already registered with windowsFonts().  
Stencil already registered with windowsFonts().  
Sylfaen already registered with windowsFonts().  
Symbol already registered with windowsFonts().  
Tahoma already registered with windowsFonts().  
Tempus Sans ITC already registered with windowsFonts().  
Times New Roman already registered with windowsFonts().  
Tolkien already registered with windowsFonts().  
Trebuchet MS already registered with windowsFonts().  
Tw Cen MT already registered with windowsFonts().  
Tw Cen MT Condensed already registered with windowsFonts().  
Tw Cen MT Condensed Extra Bold already registered with windowsFonts().  
Verdana already registered with windowsFonts().  
Viner Hand ITC already registered with windowsFonts().  
Vivaldi already registered with windowsFonts().  
Vladimir Script already registered with windowsFonts().  
Webdings already registered with windowsFonts().  
Wingdings already registered with windowsFonts().  
Wingdings 2 already registered with windowsFonts().  
Wingdings 3 already registered with windowsFonts().  
Worn Manuscript already registered with windowsFonts().  
League Spartan Extralight already registered with windowsFonts().

## 4.1 Basic Census visualization with ggplot2

* visualization is the examination of patterns and trends in data graphically
* visualization is especially key for exploratory analysis
* Below we’ll pull in the data we’ll work through - 2020 5-year Georgia median household income and median age by county, in a wide table format

```{r mhi-georgia}  
ga\_wide <- get\_acs(  
 geography = "county",  
 state = "Georgia",  
 variables = c(medinc = "B19013\_001",  
 medage = "B01002\_001"),  
 output = "wide",  
 year = 2020  
)  
```

Getting data from the 2016-2020 5-year ACS

```{r mhi-georgia}  
ga\_wide  
```

# A tibble: 159 × 6  
 GEOID NAME medincE medincM medageE medageM  
 <chr> <chr> <dbl> <dbl> <dbl> <dbl>  
 1 13001 Appling County, Georgia 37924 4761 39.9 1.7  
 2 13003 Atkinson County, Georgia 35703 5493 35.9 1.5  
 3 13005 Bacon County, Georgia 36692 3774 36.5 1   
 4 13007 Baker County, Georgia 34034 9879 52.2 4.8  
 5 13011 Banks County, Georgia 50912 4278 41.5 1.1  
 6 13013 Barrow County, Georgia 62990 2562 36 0.3  
 7 13017 Ben Hill County, Georgia 32077 4008 39.5 1.4  
 8 13021 Bibb County, Georgia 41317 1220 36.3 0.3  
 9 13023 Bleckley County, Georgia 46992 6279 36 1.5  
10 13027 Brooks County, Georgia 37516 4438 43.6 0.9  
# ℹ 149 more rows

### Getting started with ggplot2

1. Initalize **ggplot2** visualizations as a plot object using ggplot()
   * First argument is a dataset to be visualized
   * Second argument is default aes() *(aesthetic)* mappings for the plot
     + mappings can be applied to elements such as data axes, fill, color, etc.

After initializing the plot object, we layer plot elements onto the object

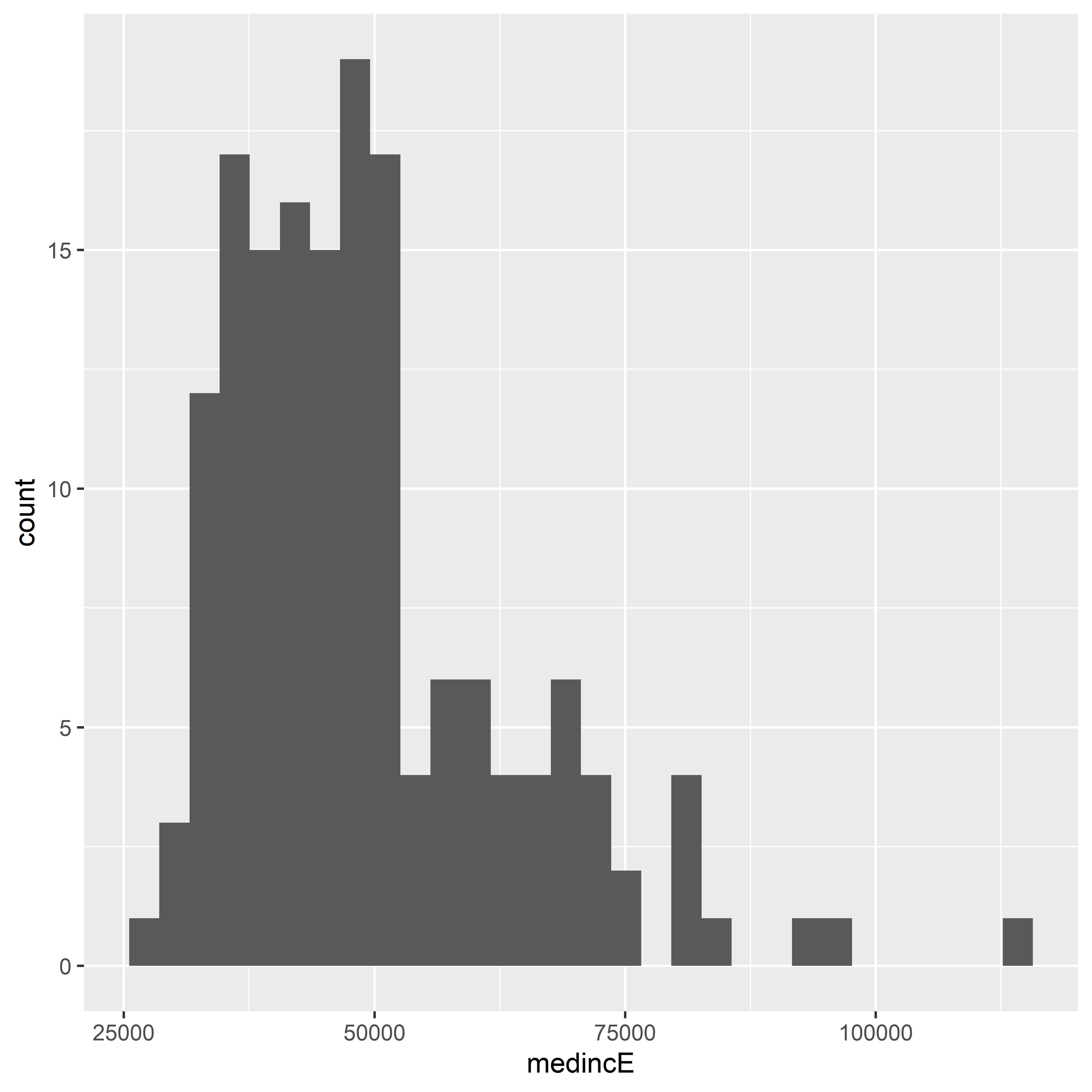
* Main factor here is the geom, determining the chart type to be used.
  + geom\_bar() - bar plots
  + geom\_line() - line plots
  + geom\_point() - point plots
  + geom\_histogram() - histograms
  + geom\_boxplot() - box and whisker plots
* To add a geom layer, use the + operator

#### Histograms

We’ll start with a histogram to look at data distribution

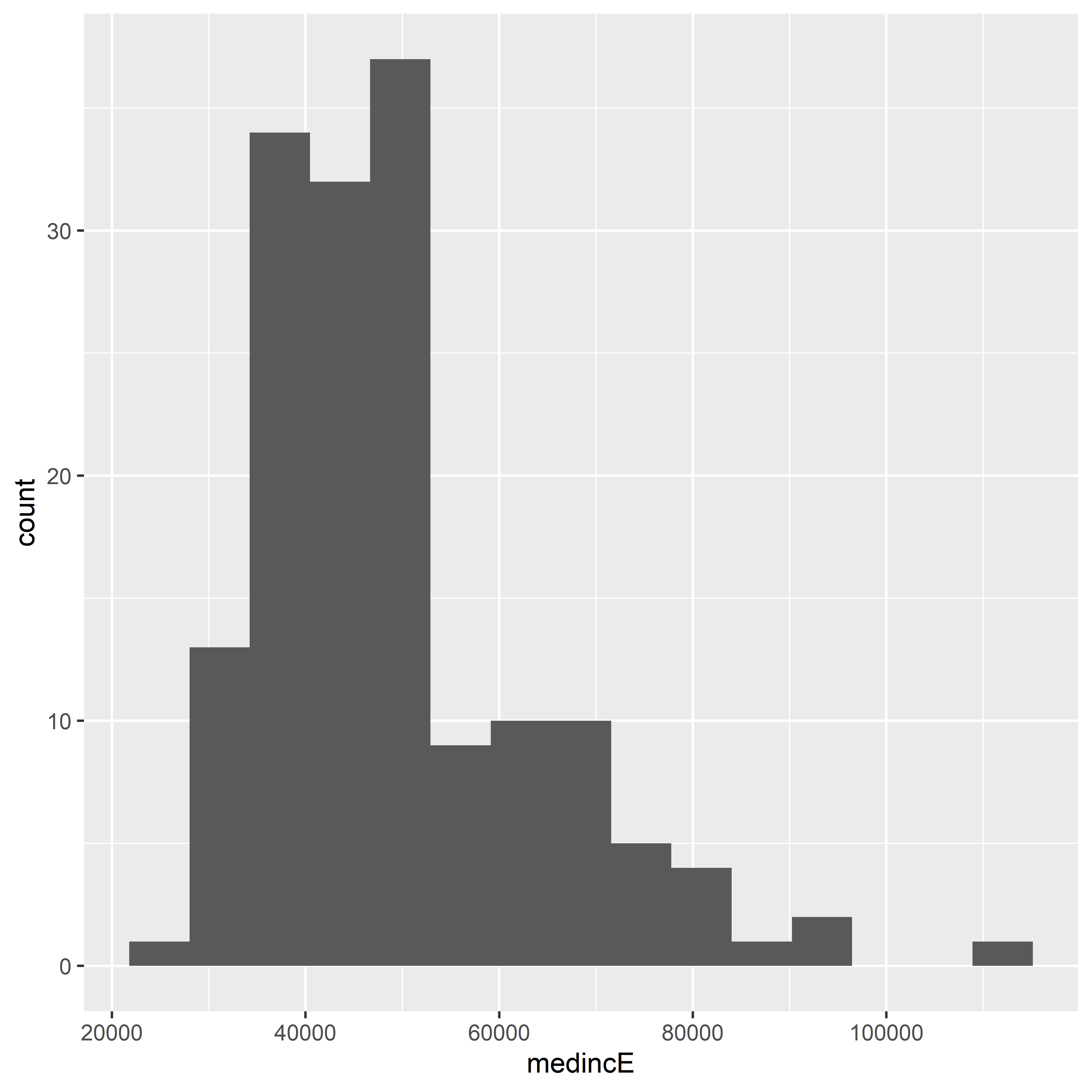
```{r georgia-histo}  
options(scipen = 999) # optional call; avoid scientific notation  
  
ggplot(ga\_wide, aes(x = medincE)) +  
 geom\_histogram()  
```

`stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



* A histogram displays the distribution of data values
  + the horizontal axis represents the range of values from least to greatest
  + the vertical axis represents the count, or number quantity of data points with a given value
    - a given range of values is ‘binned’ into a box whose height represent the count of values that may be classed in said box
    - by default, **ggplot2** organizes the data values into 30 bins
      * this can be changed via the bins parameter in geom\_histogram(), demonstrated below

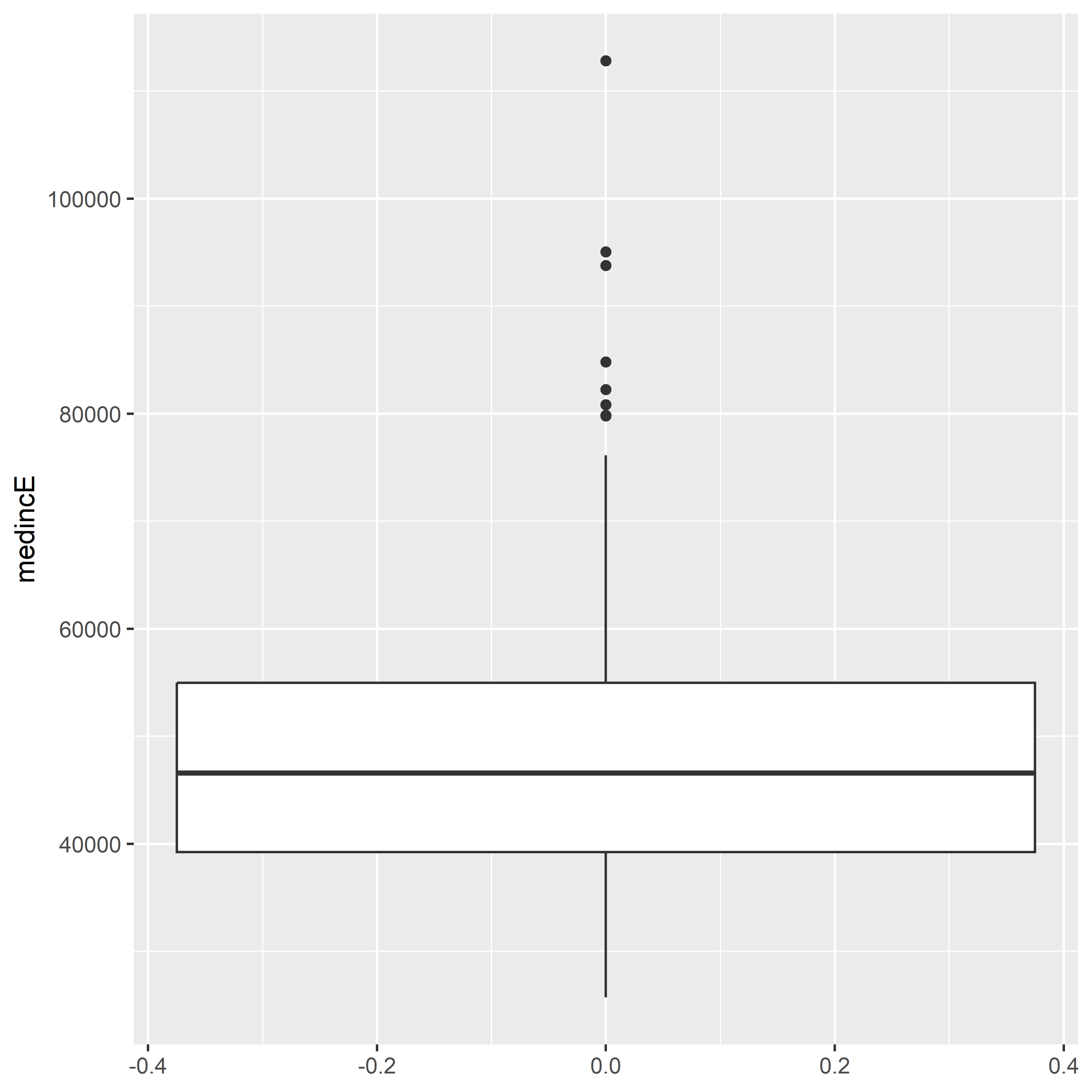
```{r georgia-15bin-histo}  
ggplot(ga\_wide, aes(x = medincE)) +  
 geom\_histogram(bins = 15)  
```



#### Box-and-Whisker Plot

* The box-and-whisker plot is a common alternative to the histogram
  + The central box of a box plot shows the *interquartile range (IQR)* or those values between the 25th and 75th percentile
    - The central line of the box represents the distribution median
  + the *whiskers* demonstrate values outside the IQR as follows:
    - they may stretch to the min/max values of the distribution, OR
    - they may represent 1.5x the IQR in either, or both, directions
  + values plotted as points plotted beyond the whiskers are *outliers*

```{r georgia-boxplot}  
ggplot(ga\_wide, aes(y = medincE)) +  
 geom\_boxplot()  
```

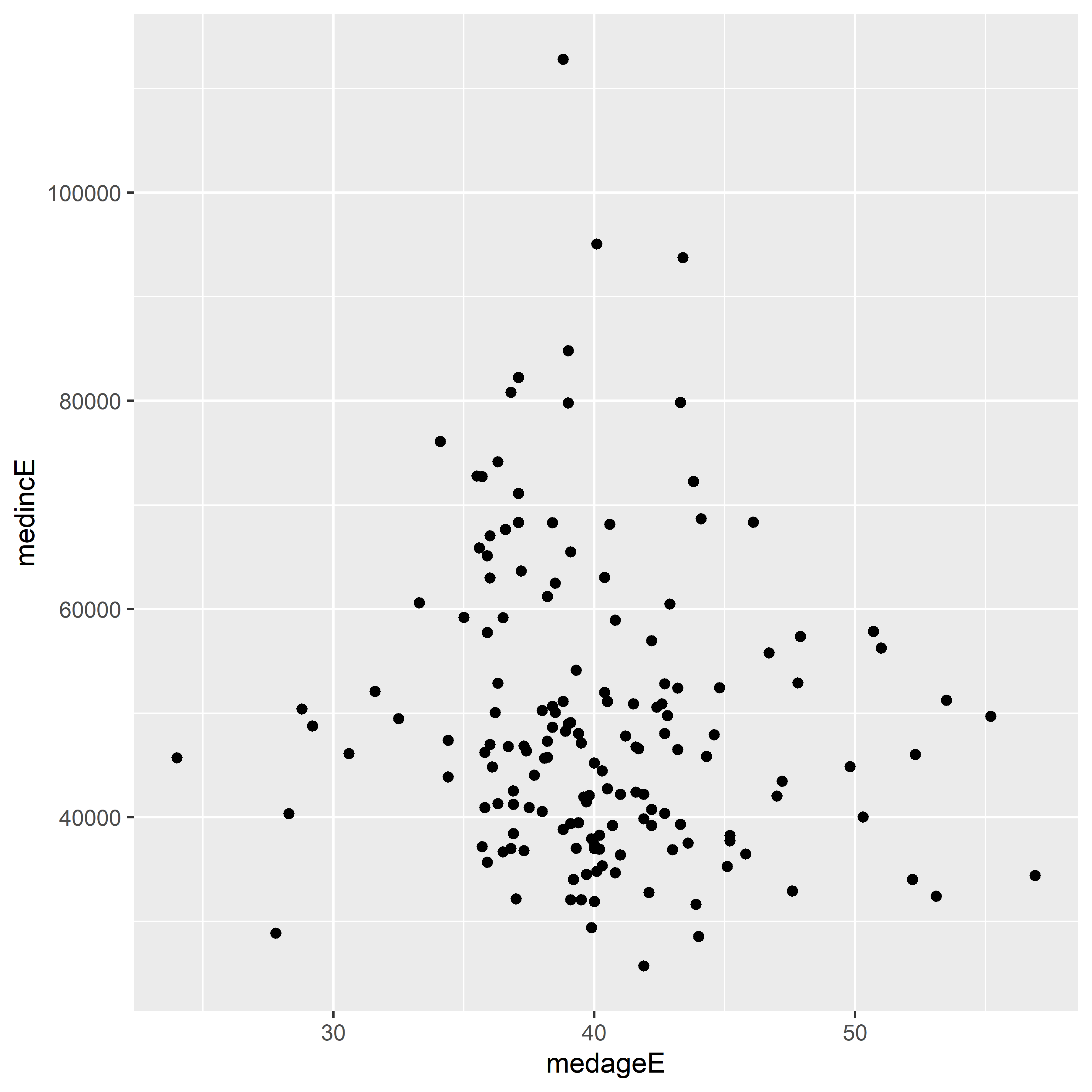


* in the above boxplot:
  + the aes() call maps the plot to the y axis, creating a vertical boxplot
  + the lower whiskers stretches to the minimum value
  + the upper whiskers stretches to 1.5x the IQR
  + we see several points plotted as outliers above the upper bound

#### (Scatter plots) Visualizing multivariate relationships

* We may want to explore/visualize interrelationship between two (or more) variables
* with two numeric variables, we may explore relationships using *scatter plots*
  + one column of values is mapped to the X-axis
  + the other is mapped to the Y-axis
  + geom\_point() makes scatter plots
    - two columns are needed for the aes() mapping

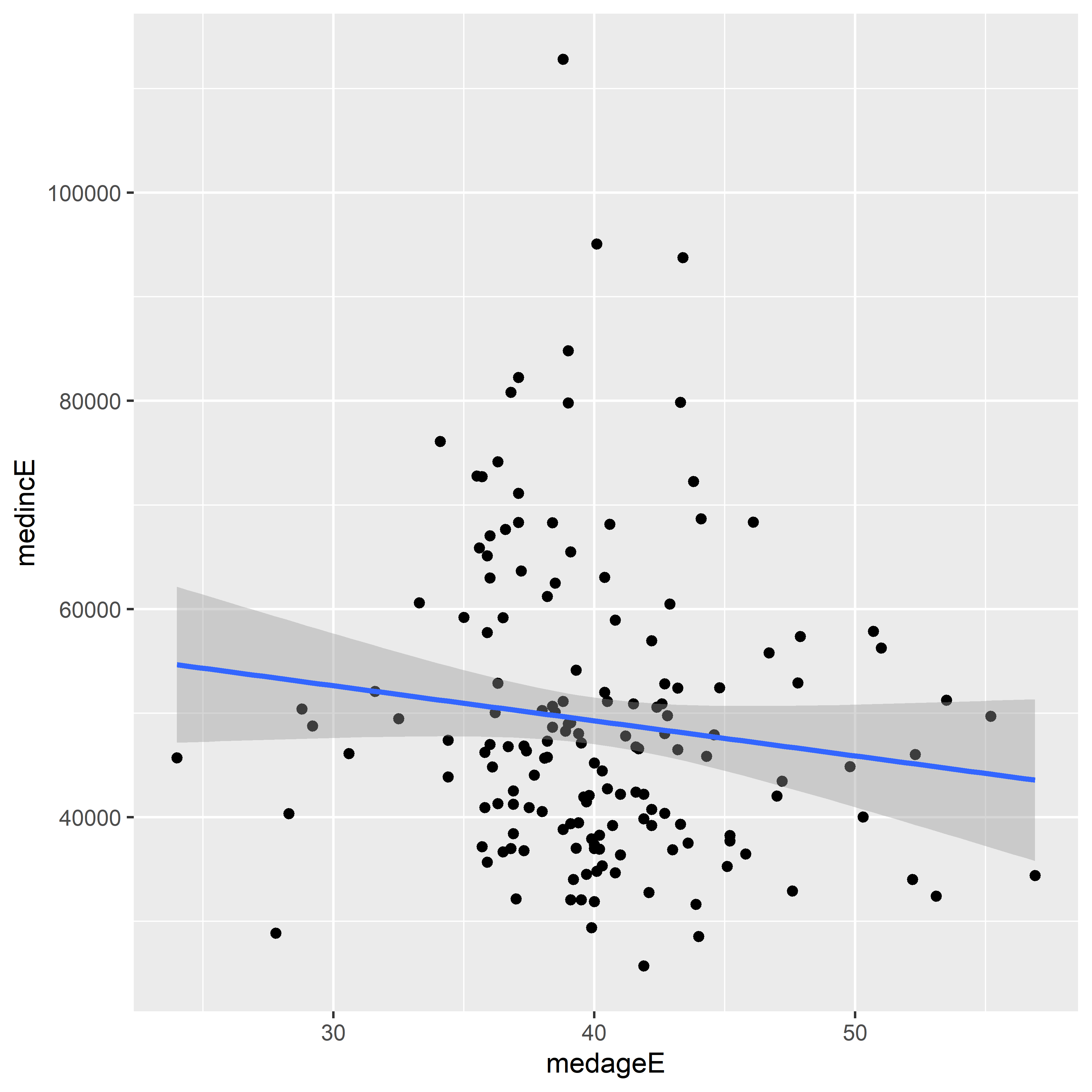
```{r georgia-medage-medinc-scatter}  
ggplot(ga\_wide, aes(x = medageE, y = medincE)) +  
 geom\_point()  
```



* Sometimes, a plot like above clearly demonstrates the existence of a relationship between variables (*correlation)*
* Sometimes, the plot fails to make such correlation immediately clear
  + in these cases, we may add further elements to clarify
  + geom\_smooth(), for instance, draws a fitted line representing the relationship between the plotted columns
    - method = "lm" uses the linear model to fit a line
    - method = "loess" uses Local Polynomial Regression Fitting to fit smoothed relationships

```{r georgia-medage-medinc-scatter-lm-smooth}  
ggplot(ga\_wide, aes(x = medageE, y = medincE)) +  
 geom\_point() +  
 geom\_smooth(method = "lm")  
```

`geom\_smooth()` using formula = 'y ~ x'



* the regression line has a light downward movement from left to right, implying somewhat of a negative relationship between the two variables
  + we might interpret this as median household income declining slightly as median age increases

## 4.2 Customizing ggplot2 visualizations

* **ggplot2** provides useful and attractive defaults, but analysts will want to customize visualizations for presenting to diverse audiences
* The subsection follows an example of visualization prep for data showing percent of commuters who use urban public transit
  + we take this data from 2019 1-year ACS Data Profile
  + The variable we use is DP03\_0021P - “Percent!!COMMUTING TO WORK!!Workers 16 years and over!!Public transportation (excluding taxicab)”
  + We’ll find the 20 largest metropolitan areas by pop using slice\_max()

```{r metro-pt-commute}  
metros <- get\_acs(  
 geography = "cbsa",  
 variables = "DP03\_0021P",  
 summary\_var = "B01003\_001",  
 survey = "acs1",  
 year = 2019  
) %>%   
 slice\_max(summary\_est, n = 20)  
```

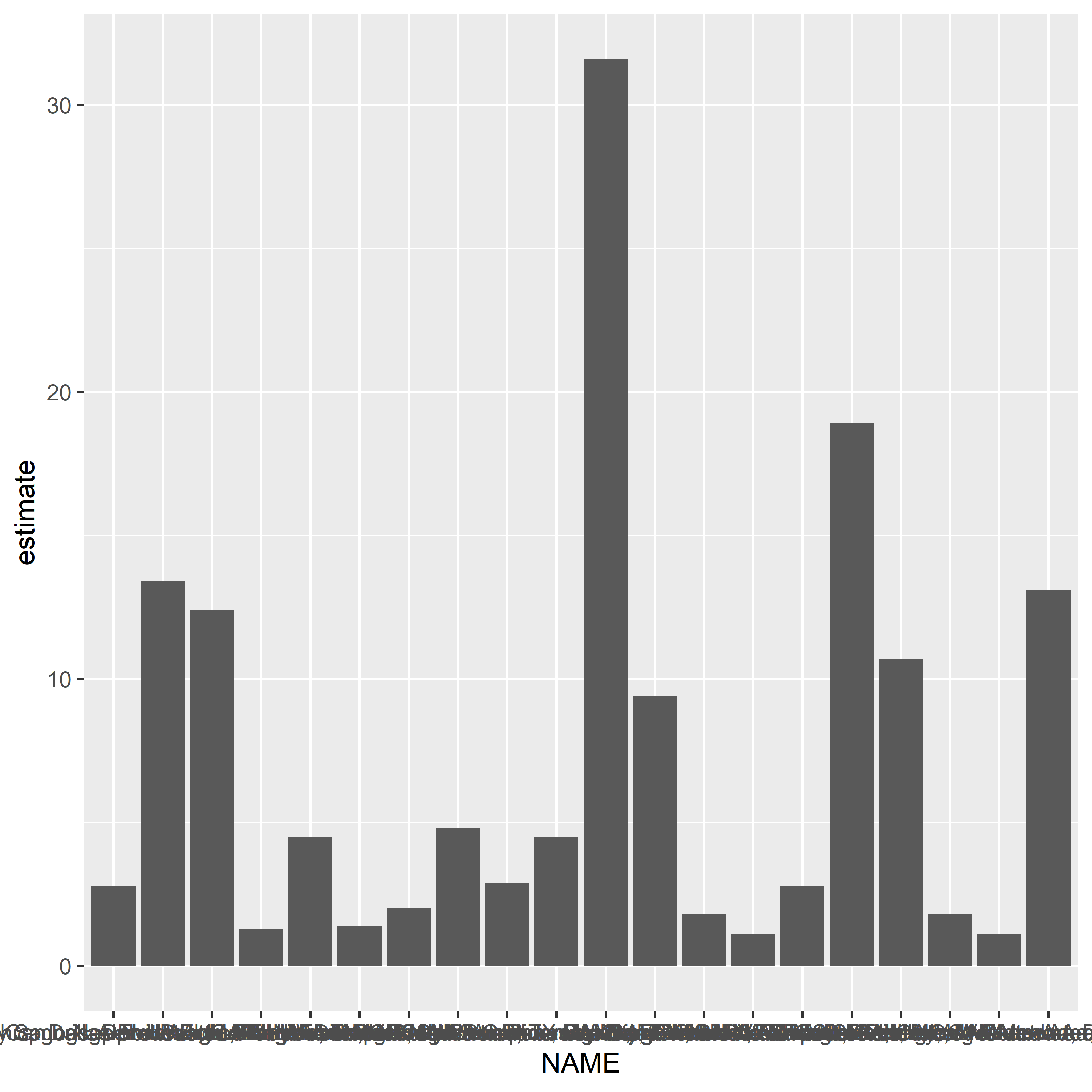
Getting data from the 2019 1-year ACS

The 1-year ACS provides data for geographies with populations of 65,000 and greater.

Using the ACS Data Profile

* data returned is 20x7, with two extra columns from the summary\_var and its associated summary\_moe, and the rows being the slice\_max return
* below, we create a bar plot comparing public transit commute share between metro areas

```{r initial-commute-chart}  
ggplot(metros, aes(x = NAME, y = estimate)) +  
 geom\_col()  
```



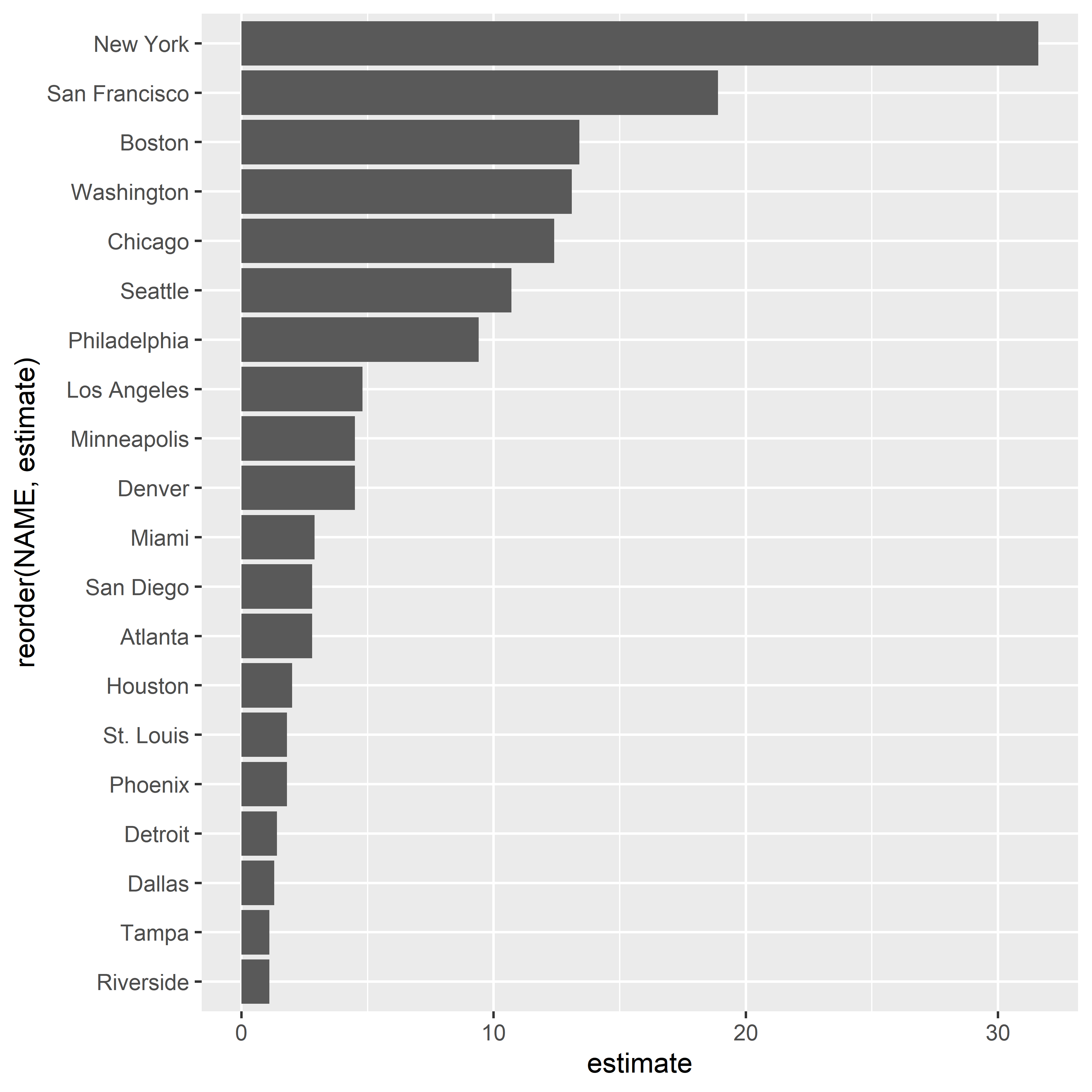
* the above plot is hard to interpret without fine-tuning the formatting
  + x-axis overlap, making reading impossible
  + axis titles are not intuitive
  + data isn’t sorted

### Improving legibility

* we can perform the above cleaning
  + **ggplot2** works with **magrittr** piping and **tidyverse** functions, meaning we can combine manipulation and visualization tasks

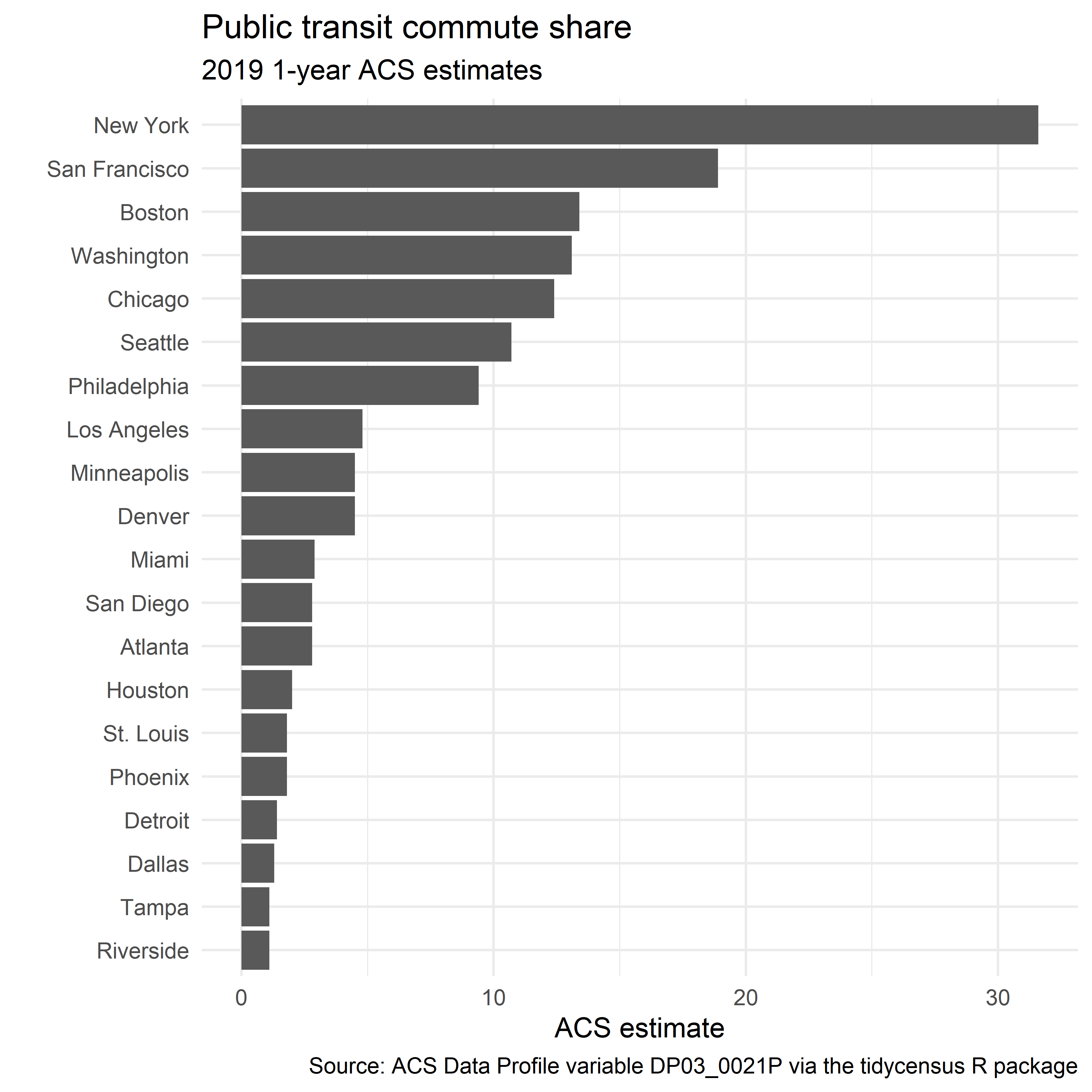
1. We will format NAME to be more intuitive using **stringr** info nested in mutate() calls - str\_remove() lets us use regex to remove unneeded parts of the NAME text for each area - the mutate calls are piped into ggplot() and ggplot() infers the dataset from the preceding pipe chains
2. We will improve legibility by putting metro name on y-axis and ACS estimate on x-axis
3. We will order data in descending order of estimated values using reorder() inside aes()
4. We plot it as a column plot

```{r cleaned-commute-chart-a}  
metros %>%   
 mutate(NAME = str\_remove(NAME, "-.\*$")) %>%   
 mutate(NAME = str\_remove(NAME, ",.\*$")) %>%   
 ggplot(aes(y = reorder(NAME, estimate), x = estimate)) +  
 geom\_col()  
```



* the above is much cleaner, but we need to fix the labels to make the chart interpretable
  + We can finish cleaning up by using labs() to specify labels
    - In this instance, we will include a title label, a subtitle label, we will leave the y axis label empty, and we will add a caption label containing data credits
  + We can also include themes (in this case, using theme\_minimal()

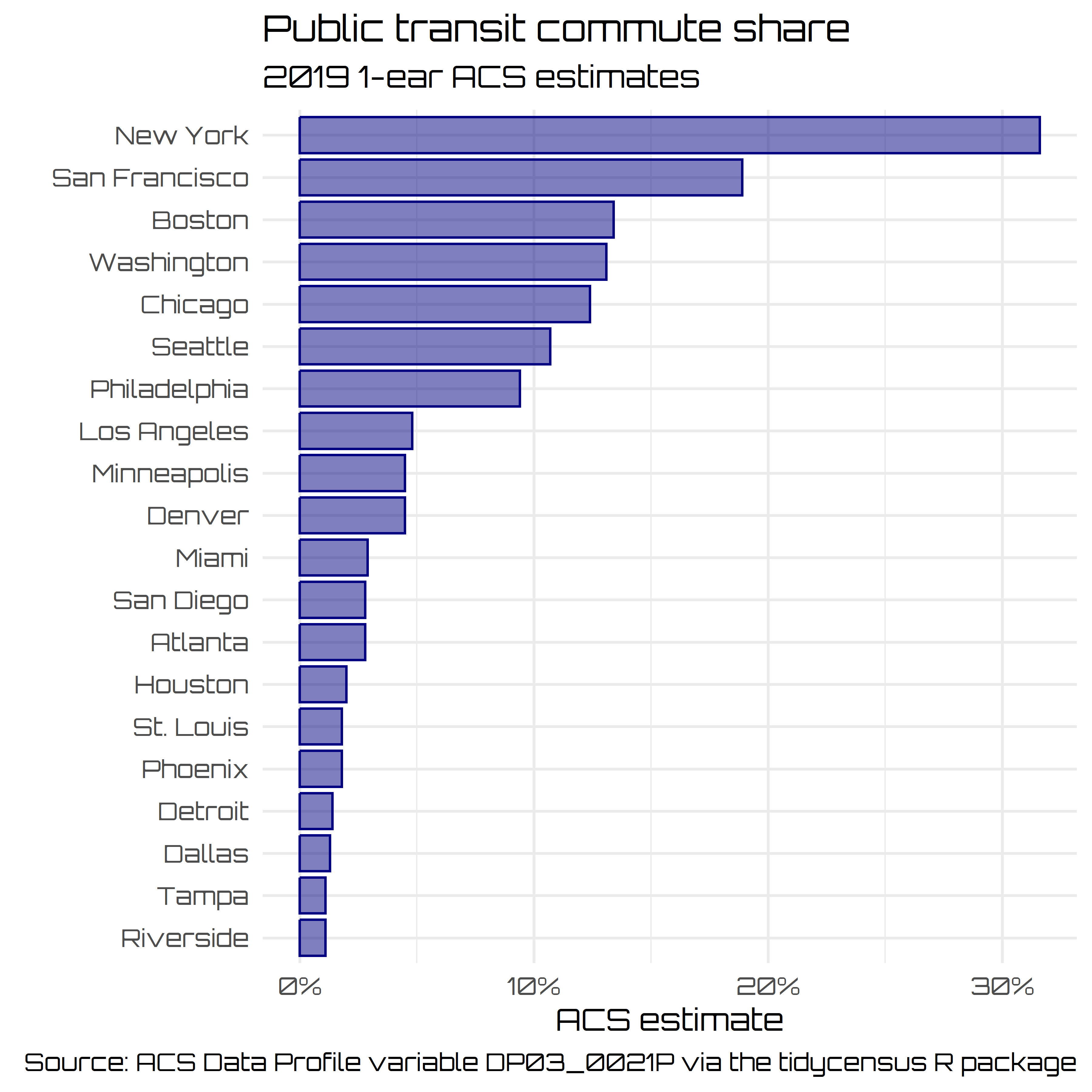
```{r final-commute-chart}  
metros %>%   
 mutate(NAME = str\_remove(NAME, "-.\*$")) %>%   
 mutate(NAME = str\_remove(NAME, ",.\*$")) %>%   
 ggplot(aes(y = reorder(NAME, estimate), x = estimate)) +  
 geom\_col() +  
 theme\_minimal() +  
 labs(title = "Public transit commute share",  
 subtitle = "2019 1-year ACS estimates",  
 y = "",  
 x = "ACS estimate",  
 caption = "Source: ACS Data Profile variable DP03\_0021P via the tidycensus R package")  
```



### Custom styling of ggplot2 charts

**ggplot2** maintains a wide scope for customization. Below, we demonstrate several style changes to color, transparency, font, and tick labels.

```{r customized-commute-chart}  
library(scales)  
  
  
metros %>%   
 mutate(NAME = str\_remove(NAME, "-.\*$")) %>%   
 mutate(NAME = str\_remove(NAME, ",.\*$")) %>%   
 ggplot(aes(y = reorder(NAME, estimate), x = estimate)) +  
 geom\_col(color = "navy", fill = "navy",  
 alpha = 0.5, width = 0.85) +  
 theme\_minimal(base\_size = 12, base\_family = "Orbitron") +  
 scale\_x\_continuous(labels = label\_percent(scale = 1)) +  
 labs(title = "Public transit commute share",  
 subtitle = "2019 1-ear ACS estimates",  
 y = "",  
 x = "ACS estimate",  
 caption = "Source: ACS Data Profile variable DP03\_0021P via the tidycensus R package")  
```



* The above code used the following modifications:
  + geom\_cols() had aes() mappings modified locally, rather than at the global level of the object initialization
    - the bar was given a color (the outline color) and fill (the fill color)
    - transparency was modified using alpha argument
    - width = 0.85 increased the space between bars by reducing the visual width of each bar to 85% of the default value
  + theme\_minimal() uses the base\_size and base\_family parameters, respectively
    - base\_size controls the font size of plot elements, defaulting to 11
    - base\_family designates the font family to use
      * in this case, the “Verdana” plot is used
      * any font family accessible from the operating system may be used
      * Check installed fonts with systemfonts::system\_fonts()
  + scale\_x\_continuous() gives us a customized X-axis
    - labels parameter can take a function or formula to programmatically style tick labels
    - **scales** packages has formatting functions to apply:
      * label\_percent()
      * label\_dollar()
      * label\_date()

### Exporting data visualizations from R

* Once you get the design you want, you have several ways of exporting
  + **Export > Save as Image** will let you save individual plots
  + ggsave() saves the last plot generated to an image file stored in a given directory (working directory by default)

```{r export-demo-1}  
ggsave("metro\_transit.png")  
```

Saving 6 x 6 in image

* ggsave() has a lot of options:
  + width and height can control the image dimensions
  + dpi controls image resolution (dots per inch)
  + path specifies where to save the image

```{r export-demo-2}  
ggsave(  
 filename = "metro\_transit.png",  
 path = "./images",  
 width = 8,  
 height = 5,  
 units = "in",  
 dpi = 300  
)  
```

## 4.3 Visualizing margins of error

* we often want to visualize margins of error to get a sense of uncertainty around an estimate
  + this is especially important when making comparative analyses between small geographies

### Data setup

* We will look at median household income for Main Counties from the 2016-2020 ACS

```{r maine-setup}  
maine <- get\_decennial(  
 state = "Maine",  
 geography = "county",  
 variables = c(totalpop = "P1\_001N"),  
 year = 2020  
) %>%   
 arrange(desc(value))  
```

Getting data from the 2020 decennial Census

Using the PL 94-171 Redistricting Data Summary File

Note: 2020 decennial Census data use differential privacy, a technique that  
introduces errors into data to preserve respondent confidentiality.  
ℹ Small counts should be interpreted with caution.  
ℹ See https://www.census.gov/library/fact-sheets/2021/protecting-the-confidentiality-of-the-2020-census-redistricting-data.html for additional guidance.  
This message is displayed once per session.

```{r maine-setup}  
maine  
```

# A tibble: 16 × 4  
 GEOID NAME variable value  
 <chr> <chr> <chr> <dbl>  
 1 23005 Cumberland County, Maine totalpop 303069  
 2 23031 York County, Maine totalpop 211972  
 3 23019 Penobscot County, Maine totalpop 152199  
 4 23011 Kennebec County, Maine totalpop 123642  
 5 23001 Androscoggin County, Maine totalpop 111139  
 6 23003 Aroostook County, Maine totalpop 67105  
 7 23017 Oxford County, Maine totalpop 57777  
 8 23009 Hancock County, Maine totalpop 55478  
 9 23025 Somerset County, Maine totalpop 50477  
10 23013 Knox County, Maine totalpop 40607  
11 23027 Waldo County, Maine totalpop 39607  
12 23023 Sagadahoc County, Maine totalpop 36699  
13 23015 Lincoln County, Maine totalpop 35237  
14 23029 Washington County, Maine totalpop 31095  
15 23007 Franklin County, Maine totalpop 29456  
16 23021 Piscataquis County, Maine totalpop 16800

* Maine has 16 counties
  + these range in population from a max of 303,069 to a min of 16,800
  + smaller counties will have relatively larger margins of error
* We will compare median household incomes to demonstrate
  1. get the data
  2. clean up NAME with str\_remove() to remove redundant info

```{r income-cleanup}  
maine\_income <- get\_acs(  
 state = "Maine",  
 geography = "county",  
 variables = c(hhincome = "B19013\_001"),  
 year = 2020  
) %>%   
 mutate(NAME = str\_remove(NAME, " County, Maine"))  
```

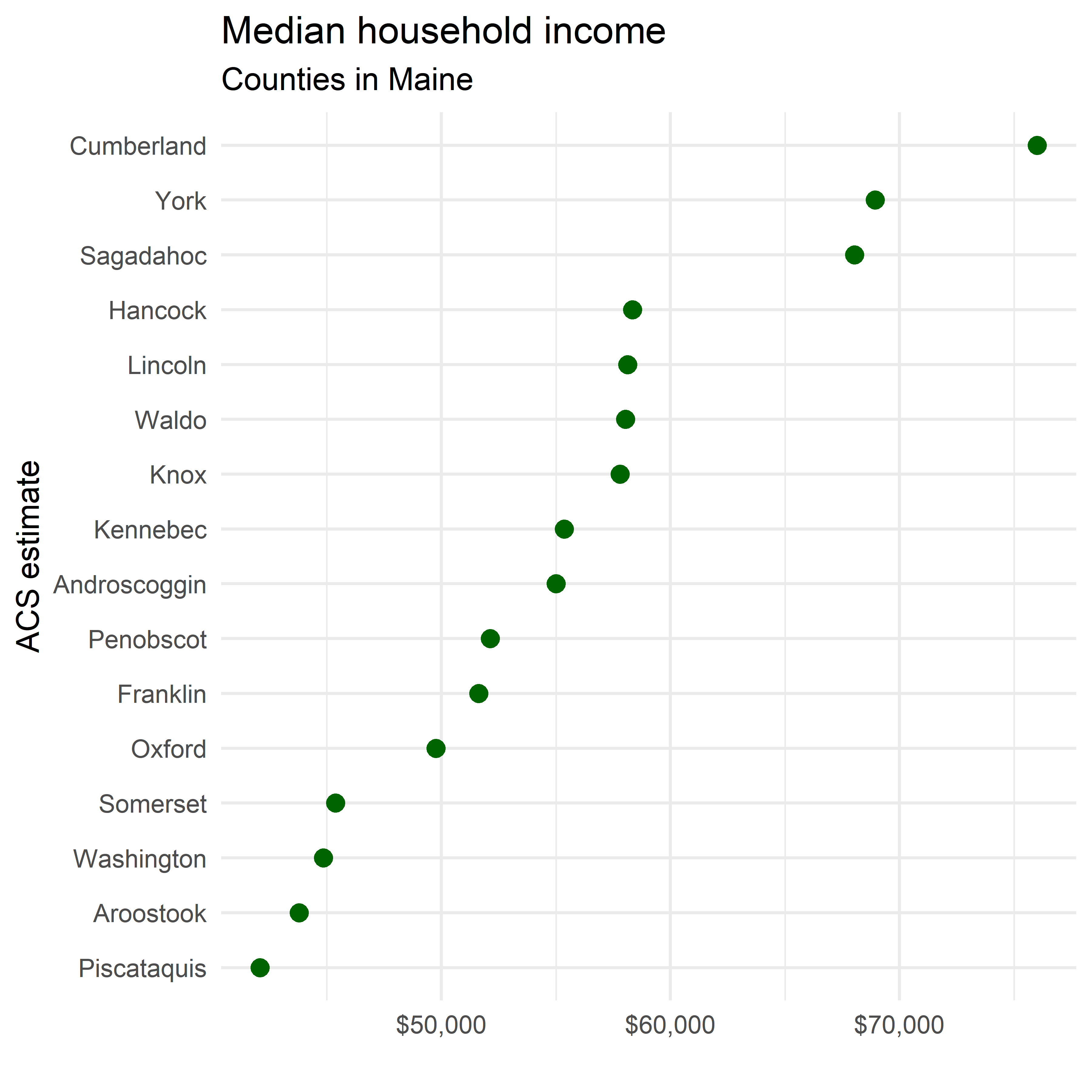
Getting data from the 2016-2020 5-year ACS

```{r income-cleanup}  
maine\_income  
```

# A tibble: 16 × 5  
 GEOID NAME variable estimate moe  
 <chr> <chr> <chr> <dbl> <dbl>  
 1 23001 Androscoggin hhincome 55002 2003  
 2 23003 Aroostook hhincome 43791 2306  
 3 23005 Cumberland hhincome 76014 1563  
 4 23007 Franklin hhincome 51630 2948  
 5 23009 Hancock hhincome 58345 2593  
 6 23011 Kennebec hhincome 55368 2112  
 7 23013 Knox hhincome 57794 2528  
 8 23015 Lincoln hhincome 58125 3974  
 9 23017 Oxford hhincome 49761 2380  
10 23019 Penobscot hhincome 52128 1836  
11 23021 Piscataquis hhincome 42083 2883  
12 23023 Sagadahoc hhincome 68039 4616  
13 23025 Somerset hhincome 45382 2694  
14 23027 Waldo hhincome 58034 3482  
15 23029 Washington hhincome 44847 2292  
16 23031 York hhincome 68932 2239

1. Now we style rank the counties and stylize the plot

```{r mhhincome-maine-rank}  
ggplot(maine\_income,  
 aes(x = estimate,  
 y = reorder(NAME, estimate))) + # take counties in NAME and reorder by estimate values; note we can use -estimate to invert the order  
 geom\_point(size = 3,  
 color = "darkgreen") +  
 labs(title = "Median household income",  
 subtitle = "Counties in Maine",  
 x = "",  
 y = "ACS estimate") +  
 theme\_minimal(base\_size = 12.5) +  
 scale\_x\_continuous(labels = label\_dollar()) # format label as dollars  
```



* The chart seems to hint at a Wealthiest > Poorest ranking, but note that it is more nuanced than our Metro example
  + The Metro examples all had large populations, with smaller relative MOEs
  + these smaller geographies may have more uncertainty due to the smaller population to sample from

### Using error bars for margins of error

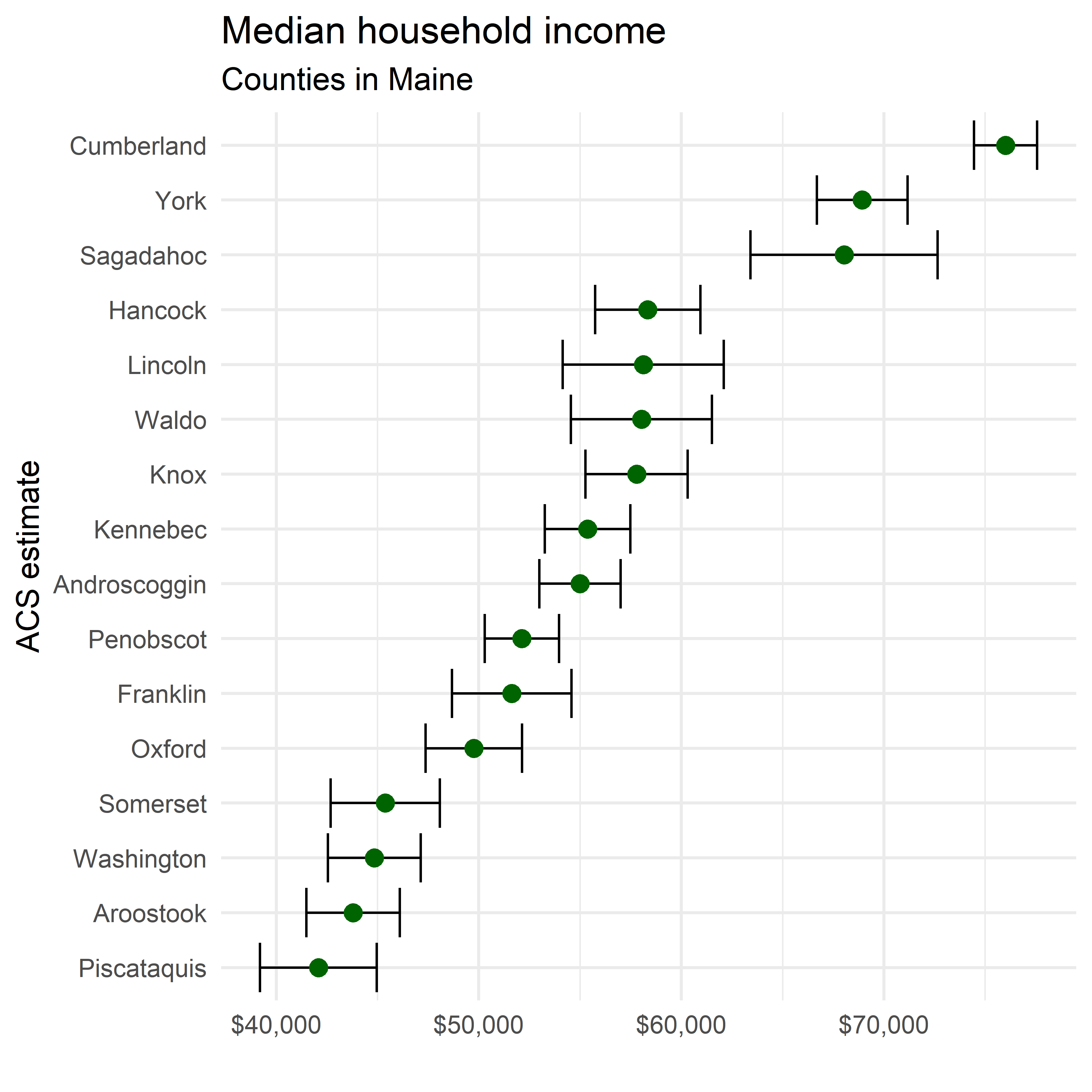
* given how close many of the values are, margins of error may prevent misleading interpretation for close values

```{r maine-income-moes}  
maine\_income %>%   
 arrange(desc(moe))  
```

# A tibble: 16 × 5  
 GEOID NAME variable estimate moe  
 <chr> <chr> <chr> <dbl> <dbl>  
 1 23023 Sagadahoc hhincome 68039 4616  
 2 23015 Lincoln hhincome 58125 3974  
 3 23027 Waldo hhincome 58034 3482  
 4 23007 Franklin hhincome 51630 2948  
 5 23021 Piscataquis hhincome 42083 2883  
 6 23025 Somerset hhincome 45382 2694  
 7 23009 Hancock hhincome 58345 2593  
 8 23013 Knox hhincome 57794 2528  
 9 23017 Oxford hhincome 49761 2380  
10 23003 Aroostook hhincome 43791 2306  
11 23029 Washington hhincome 44847 2292  
12 23031 York hhincome 68932 2239  
13 23011 Kennebec hhincome 55368 2112  
14 23001 Androscoggin hhincome 55002 2003  
15 23019 Penobscot hhincome 52128 1836  
16 23005 Cumberland hhincome 76014 1563

* there is a wide range of MOEs
  + note areas where MOE are larger than the differences between adjacently ranked estimates
    - this means the ranking has uncertainty worth noting
  + this means the dot visualization can be misleading, and we should pursue alternatives
* So we’ll add some error bars
  + geom\_errorbarh() plots horizontal bars around the points, showing the moe range around the value
  + where error bars overlap in whole or in part, this should be interpreted as having enough uncertainty that alternative rankings may be just as viable

```{r maine-error-bars}  
ggplot(maine\_income,  
 aes(x = estimate,  
 y = reorder(NAME, estimate))) + # take counties in NAME and reorder by estimate values; note we can use -estimate to invert the order  
 geom\_errorbarh(aes(xmin = estimate - moe, xmax = estimate + moe)) +  
 geom\_point(size = 3,  
 color = "darkgreen") +  
 labs(title = "Median household income",  
 subtitle = "Counties in Maine",  
 x = "",  
 y = "ACS estimate") +  
 theme\_minimal(base\_size = 12.5) +  
 scale\_x\_continuous(labels = label\_dollar()) # format label as dollars  
```



## 4.4 Visualizing ACS estimates over time

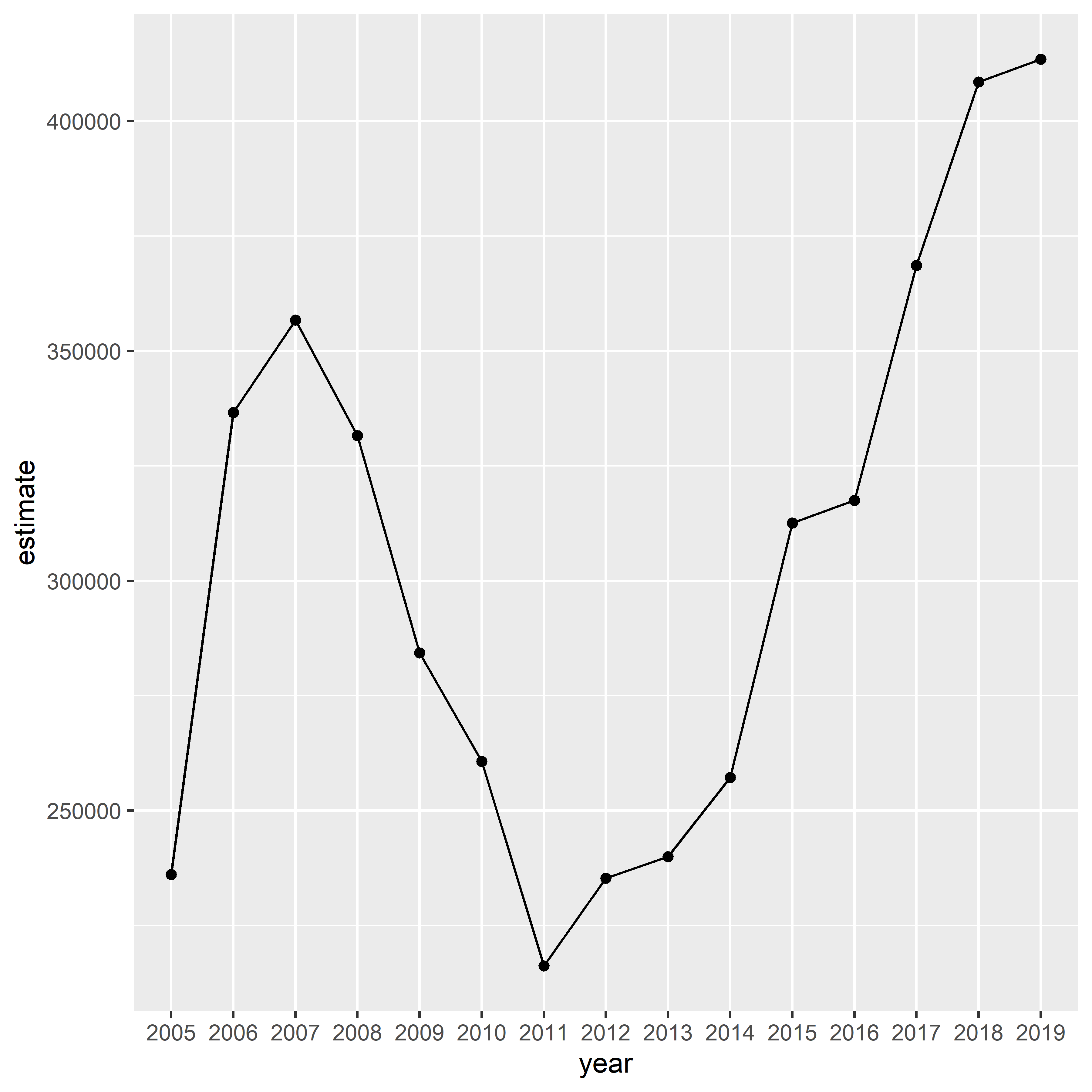
* line charts are often used to visualize time-series information for analyzing change over time
  + **ggplot2** provides the geom\_line() geom for line plots
* This demonstrates the use of 1-year ACS data for each year 2005-2019
* We look at median home value in Deschutes County, Oregon within the years 2005-2019, drawing from the 1-year ACS data for each year using the following steps:

1. We set up a named vector of years
2. we use map\_dfr() to iterate over the vector of years, pulling down the ACS data we want for each year

* ```{r iterate-over-years}  
  years <- 2005:2019  
  names(years) <- years  
    
  deschutes\_value <- map\_dfr(years, ~{  
   get\_acs(  
   geography = "county",  
   variables = "B25077\_001",  
   state = "OR",  
   county = "Deschutes",  
   year = .x,  
   survey = "acs1"  
   )  
  }, .id = "year")  
  ```
* Getting data from the 2005 1-year ACS
* The 1-year ACS provides data for geographies with populations of 65,000 and greater.
* Getting data from the 2006 1-year ACS
* The 1-year ACS provides data for geographies with populations of 65,000 and greater.
* Getting data from the 2007 1-year ACS
* The 1-year ACS provides data for geographies with populations of 65,000 and greater.
* Getting data from the 2008 1-year ACS
* The 1-year ACS provides data for geographies with populations of 65,000 and greater.
* Getting data from the 2009 1-year ACS
* The 1-year ACS provides data for geographies with populations of 65,000 and greater.
* Getting data from the 2010 1-year ACS
* The 1-year ACS provides data for geographies with populations of 65,000 and greater.
* Getting data from the 2011 1-year ACS
* The 1-year ACS provides data for geographies with populations of 65,000 and greater.
* Getting data from the 2012 1-year ACS
* The 1-year ACS provides data for geographies with populations of 65,000 and greater.
* Getting data from the 2013 1-year ACS
* The 1-year ACS provides data for geographies with populations of 65,000 and greater.
* Getting data from the 2014 1-year ACS
* The 1-year ACS provides data for geographies with populations of 65,000 and greater.
* Getting data from the 2015 1-year ACS
* The 1-year ACS provides data for geographies with populations of 65,000 and greater.
* Getting data from the 2016 1-year ACS
* The 1-year ACS provides data for geographies with populations of 65,000 and greater.
* Getting data from the 2017 1-year ACS
* The 1-year ACS provides data for geographies with populations of 65,000 and greater.
* Getting data from the 2018 1-year ACS
* The 1-year ACS provides data for geographies with populations of 65,000 and greater.
* Getting data from the 2019 1-year ACS
* The 1-year ACS provides data for geographies with populations of 65,000 and greater.

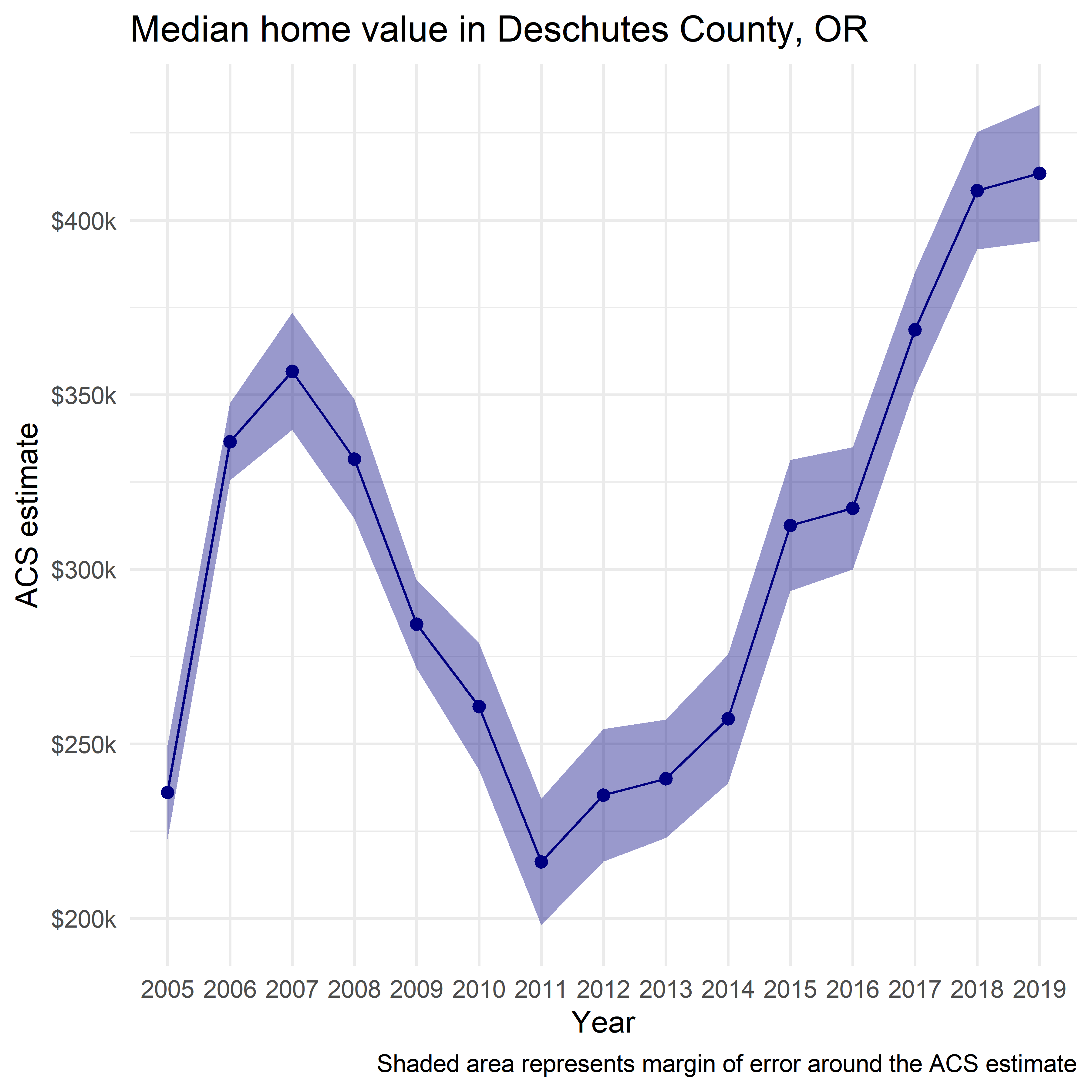
1. We can now plot it using geom\_line()

```{r deschutes-median-home-value-initial-plot}  
ggplot(deschutes\_value,   
 aes(x = year, y = estimate, group = 1)) +   
 geom\_line() +   
 geom\_point()  
```



1. Then we can clean the chart up, and add a margin of error ribbon to visualized uncertainty around the values

```{r final-median-home-value-chart}  
ggplot(deschutes\_value,   
 aes(x = year, y = estimate, group = 1)) +   
 geom\_ribbon(  
 aes(ymax = estimate + moe, ymin = estimate - moe),   
 fill = "navy",  
 alpha = 0.4) +   
 geom\_line(color = "navy") +   
 geom\_point(color = "navy", size = 2) +   
 theme\_minimal(base\_size = 12) +   
 scale\_y\_continuous(labels = label\_dollar(  
 scale = .001,   
 suffix = "k")) +   
 labs(title = "Median home value in Deschutes County, OR",  
 x = "Year",  
 y = "ACS estimate",  
 caption = "Shaded area represents margin of error around the ACS estimate")  
```



## 4.5 Exploring age and sex structure with population pyramids

* Population pyramids visualize population proportions on the x-axis
  + age cohort on the y-axis
  + sex as a categorical variable mirrored across the central axis

### Preparing data from Pop Estimates API

* Using Utah as a demo case

```{r utah-demo}  
utah <- get\_estimates(  
 geography = "state",  
 state = "UT",  
 product = "characteristics",  
 breakdown = c("SEX", "AGEGROUP"),  
 breakdown\_labels = TRUE,  
 year = 2019  
)  
  
utah  
```

# A tibble: 96 × 5  
 GEOID NAME value SEX AGEGROUP   
 <chr> <chr> <dbl> <chr> <fct>   
 1 49 Utah 3205958 Both sexes All ages   
 2 49 Utah 247803 Both sexes Age 0 to 4 years   
 3 49 Utah 258976 Both sexes Age 5 to 9 years   
 4 49 Utah 1614917 Male All ages   
 5 49 Utah 132868 Male Age 5 to 9 years   
 6 49 Utah 1591041 Female All ages   
 7 49 Utah 126108 Female Age 5 to 9 years   
 8 49 Utah 23039 Female Age 80 to 84 years  
 9 49 Utah 267985 Both sexes Age 10 to 14 years  
10 49 Utah 137940 Male Age 10 to 14 years  
# ℹ 86 more rows

* We won’t need "Both sexes" or "All ages"
  + we use a NOT EQUAL conditional in the filter (!=) to remove rows where SEX has “Both sexes” values
* We need to isolate 5 year range bands from the other rows which will not be used
  + We use the regex expression "^Age" below to select all values of AGEGROUP that start with the word Age, as these are the 5 year range rows
* We mutate values where SEX equals "Male" to be negative, so that they can be mirrored across a central axis on 0.

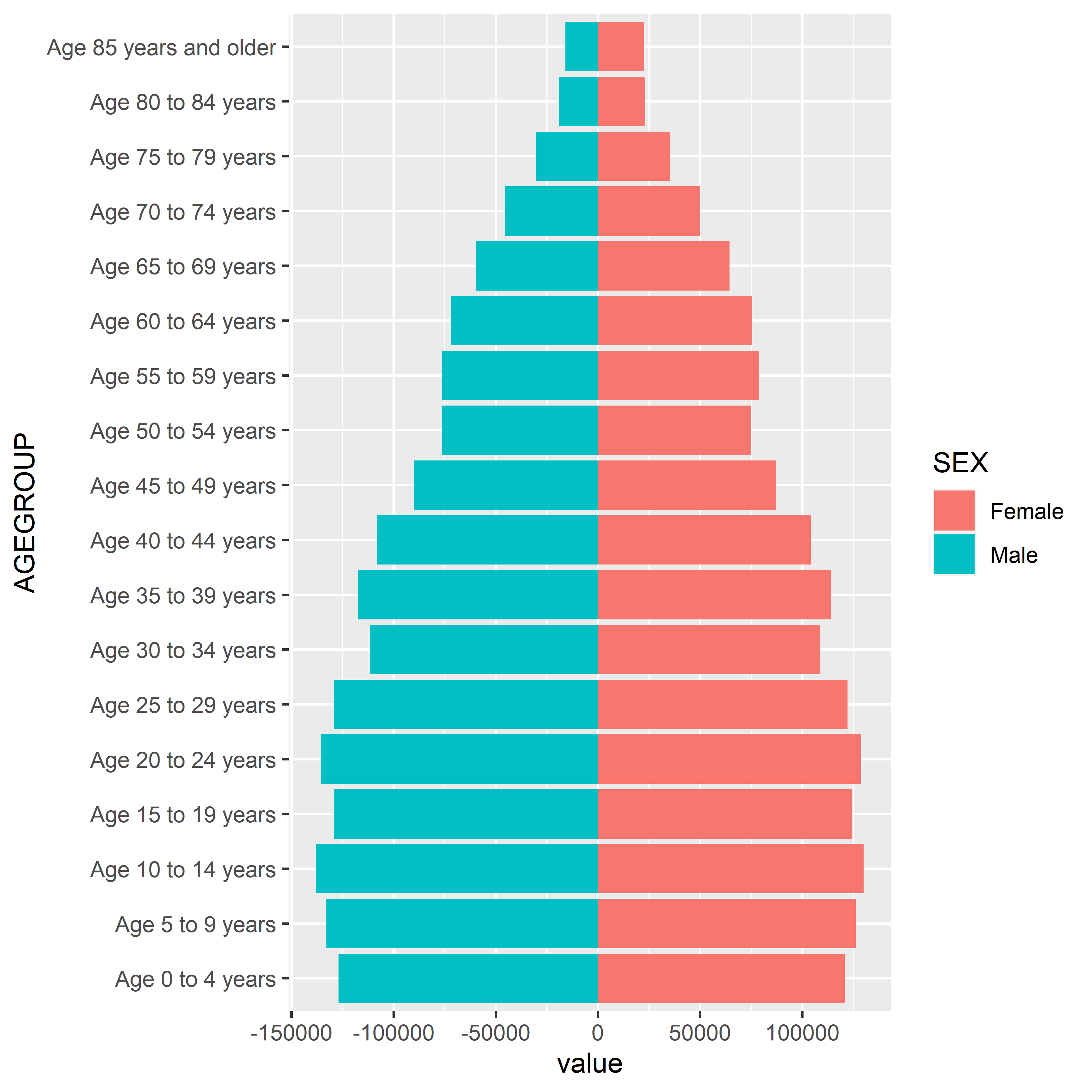
```{r utah-tidying}  
utah\_filtered <- filter(utah, str\_detect(AGEGROUP, "^Age"),  
 SEX != "Both sexes") %>%   
 mutate(value = ifelse(SEX == "Male", -value, value))  
```

### Designing and styling the pop pyramid

We can make a rough draft with the following code

* Note:
  + x axis has the value; negative for males, positive for females
  + y axis has the age bands in 5 year increments
  + bars are filled with color based on the categorical value of “Male” or “Female” in SEX

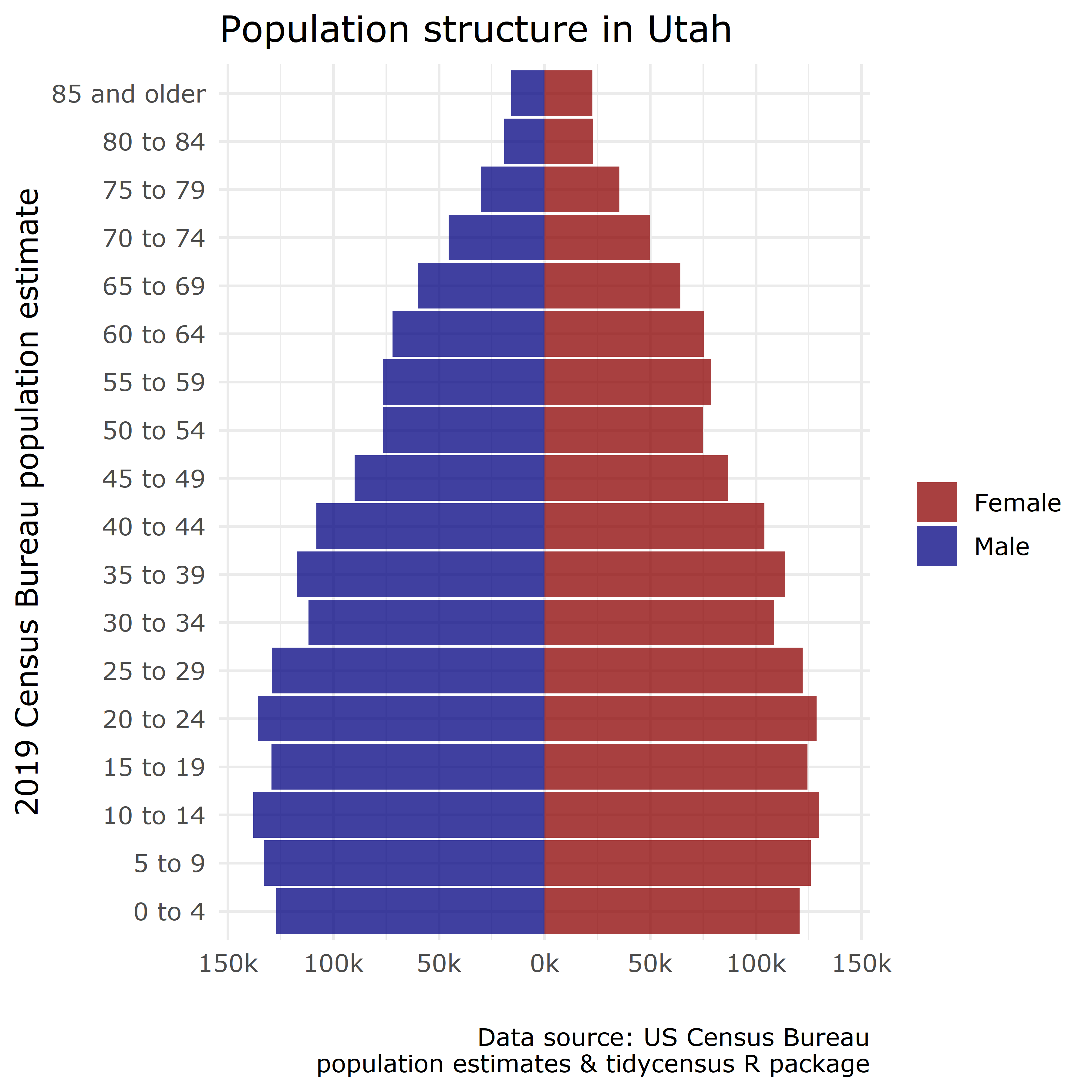
```{r utah-design}  
ggplot(utah\_filtered, aes(x = value, y = AGEGROUP, fill = SEX)) +  
 geom\_col()  
```



We can clean this up for clarity and aesthetics

* We will remove redundancy in labels
* We will adjust the scale of axis ticks for easier interpretability

```{r clean-pyramid}  
utah\_pyramid <- ggplot(utah\_filtered,  
 aes(x = value,  
 y = AGEGROUP,  
 fill = SEX)) +   
 geom\_col(width = 0.95, alpha = 0.75) +  
 theme\_minimal(base\_family = "Verdana",   
 base\_size = 12) +  
 scale\_x\_continuous(  
 labels = ~number\_format(scale = .001, suffix = "k")(abs(.x)),  
 limits = 140000 \* c(-1,1)  
 ) +  
 scale\_y\_discrete(labels = ~str\_remove\_all(.x, "Age\\s|\\syears")) +  
 scale\_fill\_manual(values = c("darkred", "navy")) +  
 labs(x = "",  
 y = "2019 Census Bureau population estimate",  
 title = "Population structure in Utah",  
 fill = "",  
 caption = "Data source: US Census Bureau\npopulation estimates & tidycensus R package")  
  
utah\_pyramid  
```



## 4.6 Visualizing group-wise comparisons

* faceted plots (aka small multiples) allow for the visualization of multiple sub-groups of data in such a way that they can be compared to each other, side by side
* We will use census tract geography and median home values in Clackamas, Columbia, Marion, Multnomah, Washington, and Yamhill counties, Oregon.

```{r Oregon-area-mhv}  
housing\_val <- get\_acs(  
 geography = "tract",  
 variables = "B25077\_001",  
 state = "OR",  
 county = c("Multnomah", "Clackamas",  
 "Washington", "Yamhill",  
 "Marion", "Columbia"),  
 year = 2020  
)  
```

Getting data from the 2016-2020 5-year ACS

```{r Oregon-area-mhv}  
housing\_val  
```

# A tibble: 513 × 5  
 GEOID NAME variable estimate moe  
 <chr> <chr> <chr> <dbl> <dbl>  
 1 41005020101 Census Tract 201.01, Clackamas County, … B25077\_… 666700 131453  
 2 41005020102 Census Tract 201.02, Clackamas County, … B25077\_… 909000 130787  
 3 41005020201 Census Tract 202.01, Clackamas County, … B25077\_… 897400 97893  
 4 41005020202 Census Tract 202.02, Clackamas County, … B25077\_… 821200 93103  
 5 41005020302 Census Tract 203.02, Clackamas County, … B25077\_… 565600 32555  
 6 41005020303 Census Tract 203.03, Clackamas County, … B25077\_… 560000 34669  
 7 41005020304 Census Tract 203.04, Clackamas County, … B25077\_… 646600 71342  
 8 41005020401 Census Tract 204.01, Clackamas County, … B25077\_… 525000 45504  
 9 41005020403 Census Tract 204.03, Clackamas County, … B25077\_… 749800 67248  
10 41005020404 Census Tract 204.04, Clackamas County, … B25077\_… 677300 26384  
# ℹ 503 more rows

* NAME can be split into several useful columns using separate()

```{r}  
housing\_val2 <- separate(  
 housing\_val,  
 NAME,  
 into = c("tract", "county", "state"),  
 sep = ", "  
)  
  
housing\_val2  
```

# A tibble: 513 × 7  
 GEOID tract county state variable estimate moe  
 <chr> <chr> <chr> <chr> <chr> <dbl> <dbl>  
 1 41005020101 Census Tract 201.01 Clackamas Cou… Oreg… B25077\_… 666700 131453  
 2 41005020102 Census Tract 201.02 Clackamas Cou… Oreg… B25077\_… 909000 130787  
 3 41005020201 Census Tract 202.01 Clackamas Cou… Oreg… B25077\_… 897400 97893  
 4 41005020202 Census Tract 202.02 Clackamas Cou… Oreg… B25077\_… 821200 93103  
 5 41005020302 Census Tract 203.02 Clackamas Cou… Oreg… B25077\_… 565600 32555  
 6 41005020303 Census Tract 203.03 Clackamas Cou… Oreg… B25077\_… 560000 34669  
 7 41005020304 Census Tract 203.04 Clackamas Cou… Oreg… B25077\_… 646600 71342  
 8 41005020401 Census Tract 204.01 Clackamas Cou… Oreg… B25077\_… 525000 45504  
 9 41005020403 Census Tract 204.03 Clackamas Cou… Oreg… B25077\_… 749800 67248  
10 41005020404 Census Tract 204.04 Clackamas Cou… Oreg… B25077\_… 677300 26384  
# ℹ 503 more rows

* tidyverse lets us use a series of pipes, groupings, and summaries to do group-wise comparison across all 6 counties:

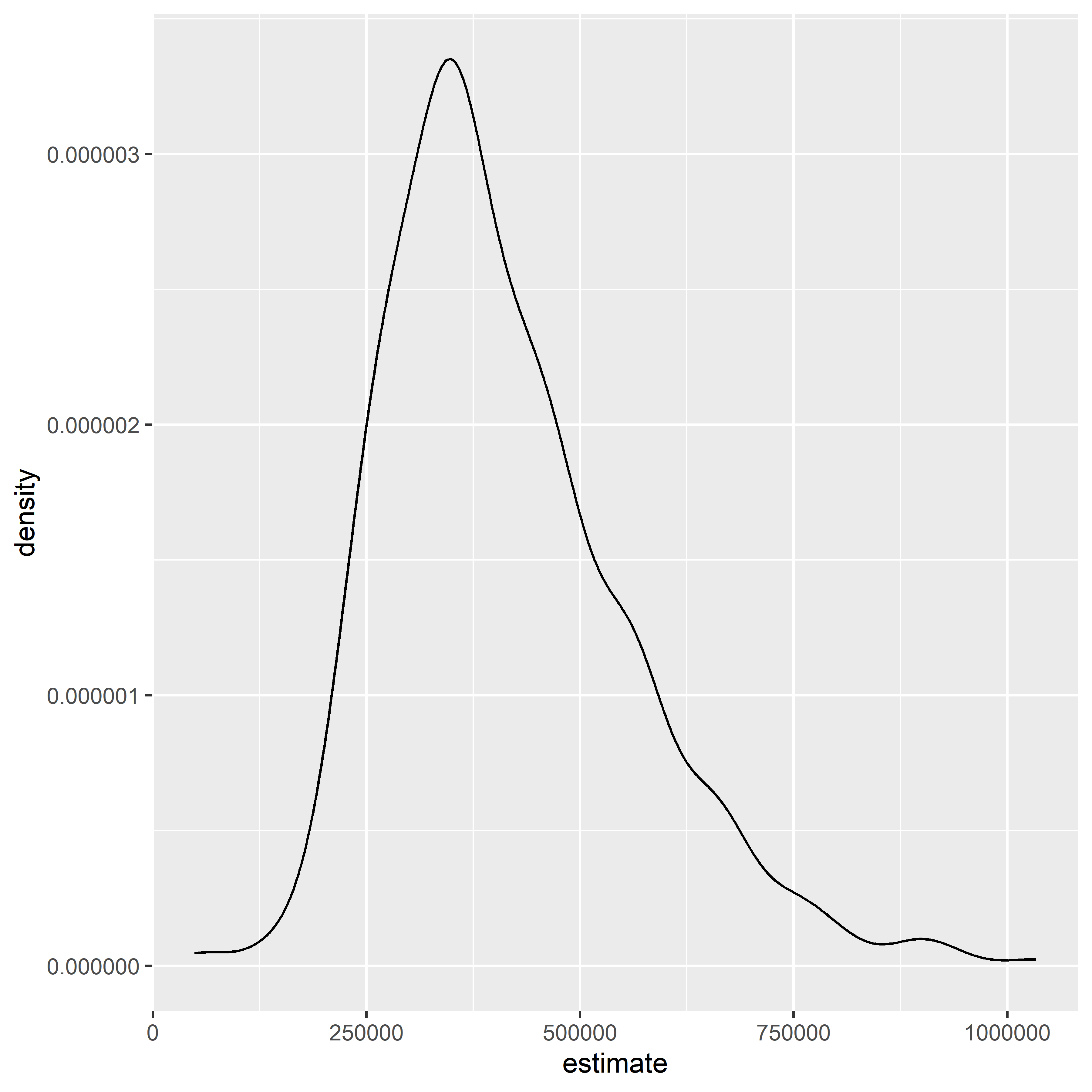
```{r}  
housing\_val2 %>%   
 group\_by(county) %>%   
 summarize(min = min(estimate, na.rm = TRUE),  
 mean = mean(estimate, na.rm = TRUE),  
 median = median(estimate, na.rm = TRUE),  
 max = max(estimate, na.rm = TRUE))  
```

# A tibble: 6 × 5  
 county min mean median max  
 <chr> <dbl> <dbl> <dbl> <dbl>  
1 Clackamas County 62800 449941. 426700 909000  
2 Columbia County 218100 277591. 275900 362200  
3 Marion County 48700 270969. 261200 483500  
4 Multnomah County 192900 455706. 425950 1033500  
5 Washington County 221900 419618. 406100 769700  
6 Yamhill County 230000 333816. 291100 545500

* We can now plot the (statistical) distribution of values
  + We’ll see a simple density plot of all values, and then the use of facet\_wrap() to make a group of density plots for each county

```{r total-distribution}  
ggplot(housing\_val2, aes(x = estimate)) +   
 geom\_density()  
```

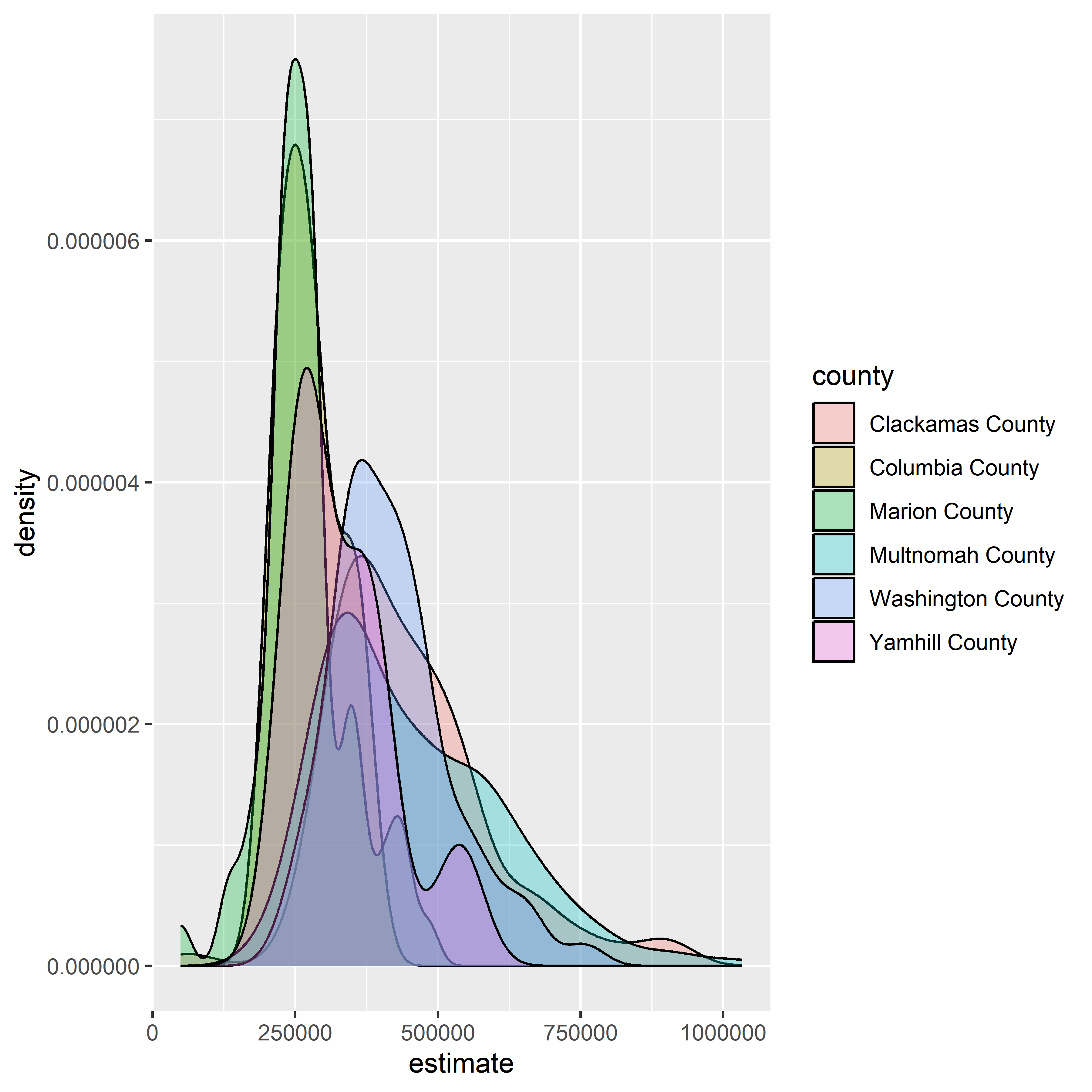
Warning: Removed 9 rows containing non-finite values (`stat\_density()`).



* We can do an overlay across the variables using fill

```{r}  
ggplot(housing\_val2, aes(x = estimate, fill = county)) +   
 geom\_density(alpha = 0.3)  
```

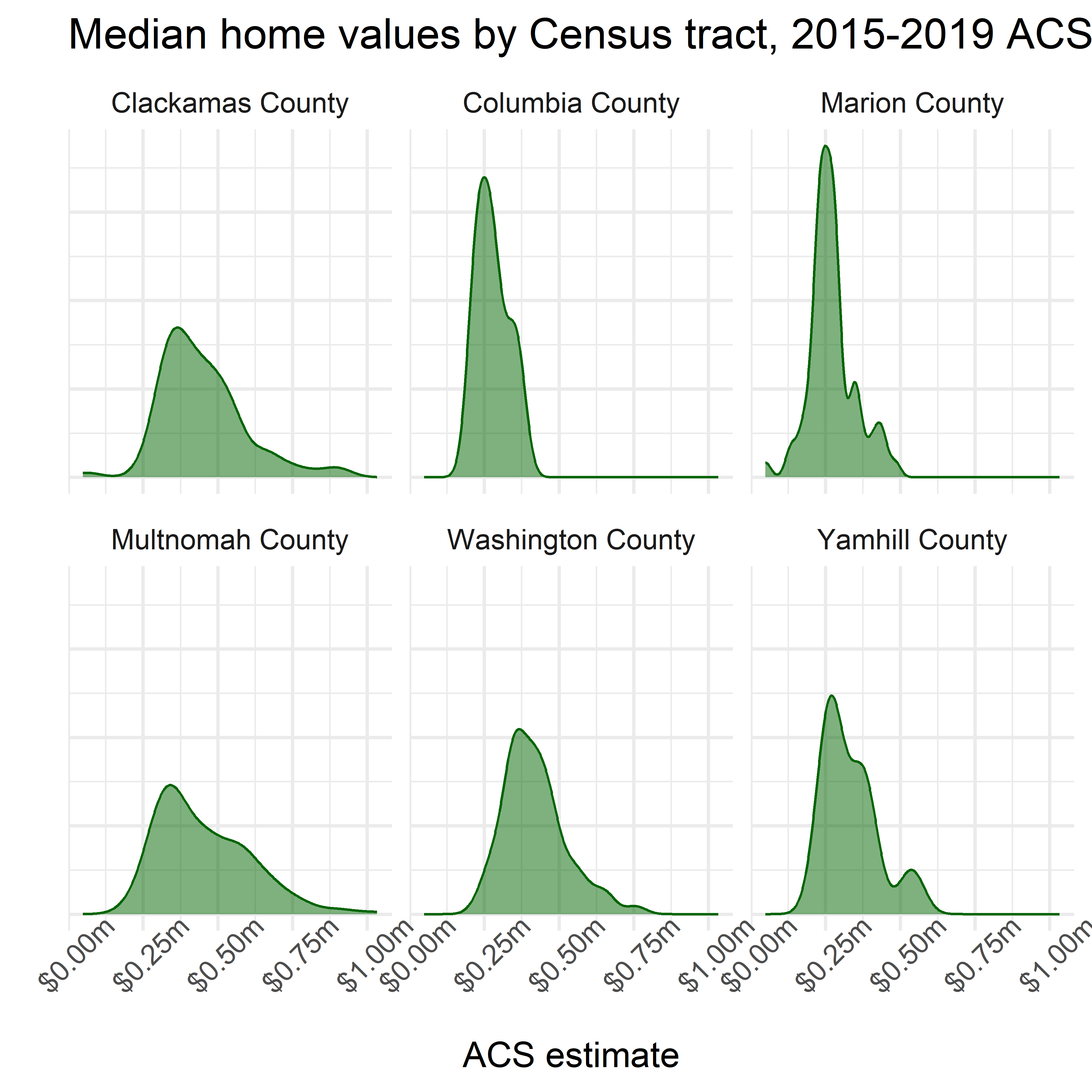
Warning: Removed 9 rows containing non-finite values (`stat\_density()`).



* facet\_wrap() instead breaks each overlayed plot into its own individual plot

```{r}  
ggplot(housing\_val2, aes(x = estimate)) +   
 geom\_density(fill = "darkgreen", color = "darkgreen", alpha = 0.5) +  
 facet\_wrap(~county) +   
 scale\_x\_continuous(labels = dollar\_format(scale = 0.000001,  
 suffix = "m")) +  
 theme\_minimal(base\_size = 14) +  
 theme(axis.text.y = element\_blank(),  
 axis.text.x = element\_text(angle = 45)) +  
 labs(x = "ACS estimate",  
 y = "",  
 title = "Median home values by Census tract, 2015-2019 ACS")  
```

Warning: Removed 9 rows containing non-finite values (`stat\_density()`).



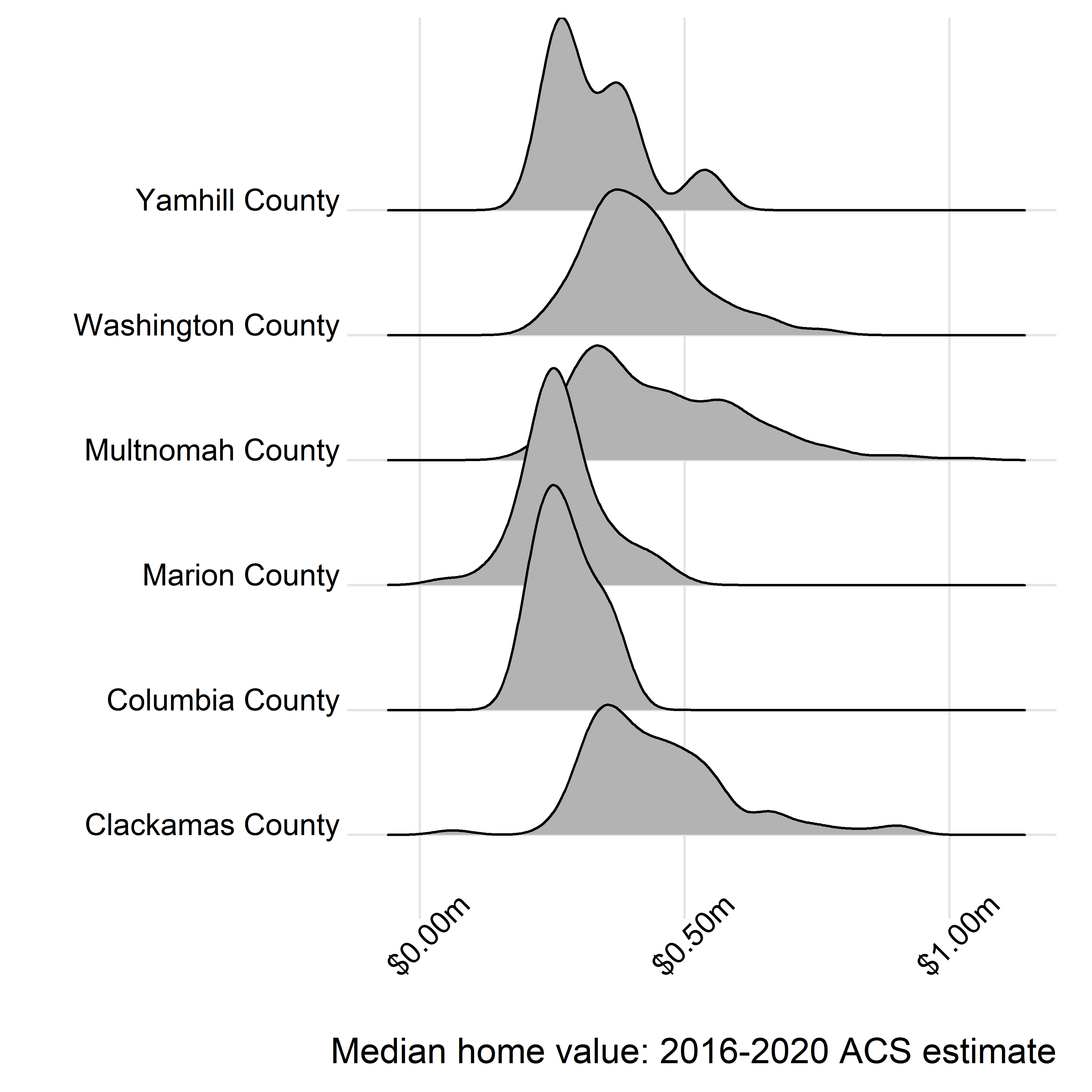
## 4.7 Advanced visualization with ggplot2 extensions

* there are a variety of ggplot2 extensions developed by folks outside the ggplot2 core team
  + examples can be found [here](https://exts.ggplot2.tidyverse.org/)
* The author walks us through 4:
  + **ggridges** which generates ridgeline plots - a density plot visualization with plots overlapping each other as though silhouetted ridges (also called a “joyplot” in reference to the iconic Joy Division album cover)

```{r}  
library(ggridges)  
  
ggplot(housing\_val2, aes(x = estimate,  
 y = county)) +  
 geom\_density\_ridges() +  
 theme\_ridges() +  
 labs(x = "Median home value: 2016-2020 ACS estimate",  
 y = "") +  
 scale\_x\_continuous(labels = label\_dollar(scale = .000001, suffix = "m"),  
 breaks = c(0, 500000, 1000000)) +  
 theme(axis.text.x = element\_text(angle = 45))  
```

Picking joint bandwidth of 36200

Warning: Removed 9 rows containing non-finite values (`stat\_density\_ridges()`).



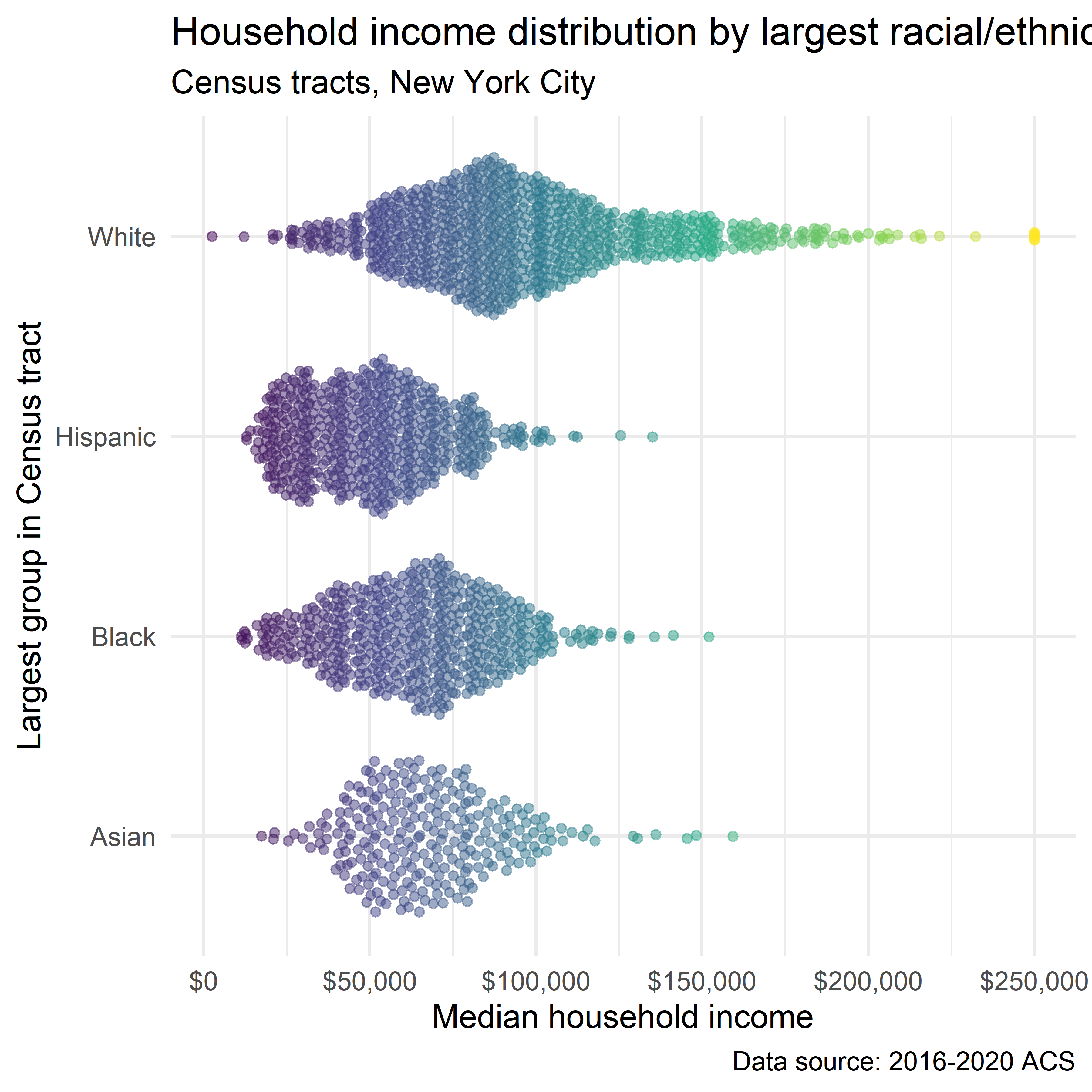
* **ggbeeswarm** which generates beeswarm plots - clouds of points jittered/dispersed to show overall density
  + can be useful for visualizing multiple variables on a chart

```{r}  
library(ggbeeswarm)  
  
ny\_race\_income <- get\_acs(  
 geography = "tract",  
 state = "NY",  
 county = c("New York", "Bronx", "Queens", "Richmond", "Kings"),  
 variables = c(White = "B03002\_003",  
 Black = "B03002\_004",  
 Asian = "B03002\_006",  
 Hispanic = "B03002\_012"),  
 summary\_var = "B19013\_001",  
 year = 2020  
) %>%   
 group\_by(GEOID) %>%   
 filter(estimate == max(estimate, na.rm = TRUE)) %>%   
 ungroup() %>%   
 filter(estimate != 0)  
```

Getting data from the 2016-2020 5-year ACS

```{r}  
ggplot(ny\_race\_income, aes(x = variable,   
 y = summary\_est,  
 color = summary\_est)) +  
 geom\_quasirandom(alpha = 0.5) +  
 coord\_flip() +  
 theme\_minimal(base\_size = 13) +  
 scale\_color\_viridis\_c(guide = "none") +  
 scale\_y\_continuous(labels = label\_dollar()) +  
 labs(x = "Largest group in Census tract",  
 y = "Median household income",  
 title = "Household income distribution by largest racial/ethnic group",  
 subtitle = "Census tracts, New York City",  
 caption = "Data source: 2016-2020 ACS")  
```

Warning: Removed 35 rows containing missing values (`position\_quasirandom()`).

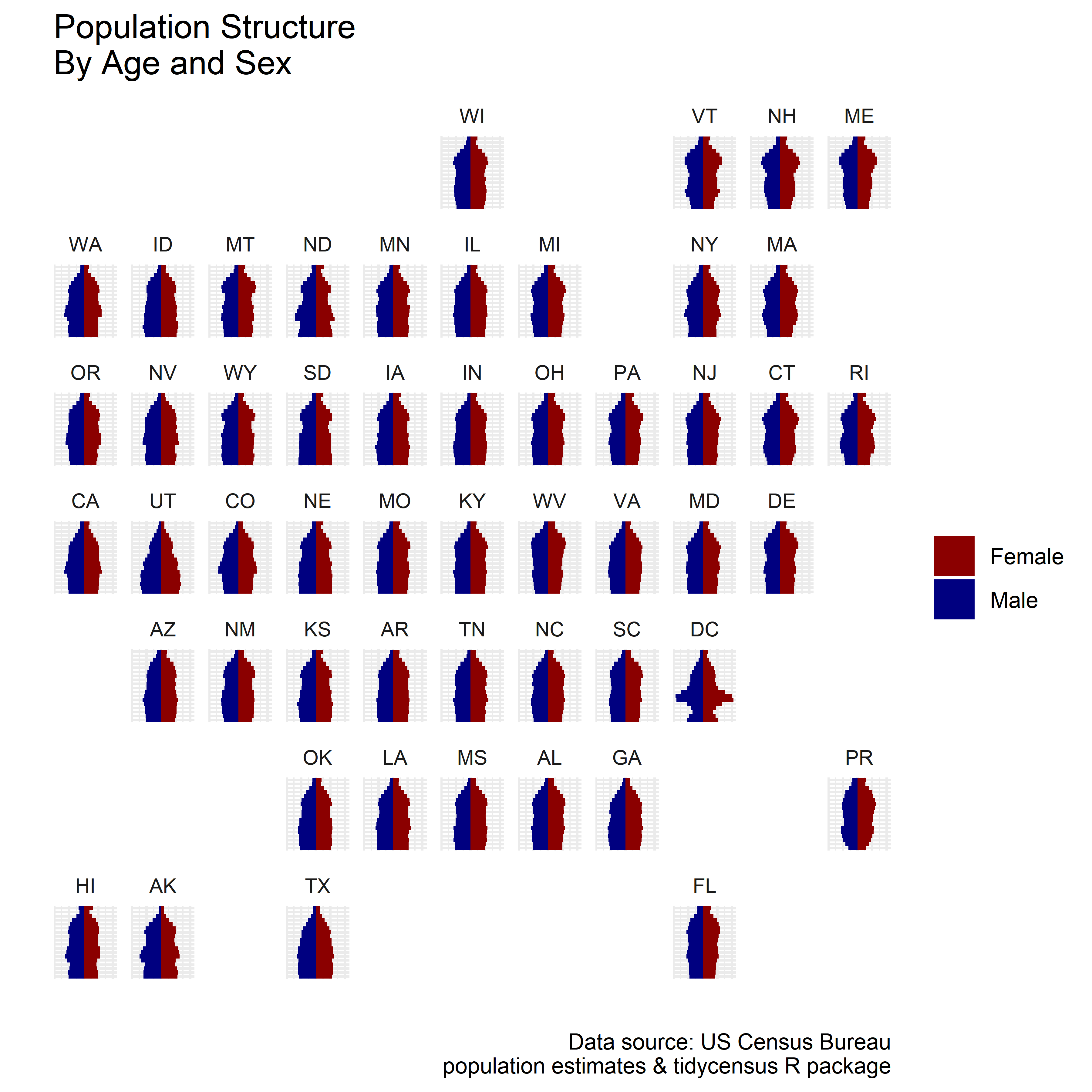


* **geofacet** plots for faceted representation of information in a spatial format without more complex geospatial tools
  + “The key is to use a column that can map correctly to information in the geofaceted grid you are using”

```{r}  
library(geofacet)  
```

The legacy packages maptools, rgdal, and rgeos, underpinning the sp package,  
which was just loaded, will retire in October 2023.  
Please refer to R-spatial evolution reports for details, especially  
https://r-spatial.org/r/2023/05/15/evolution4.html.  
It may be desirable to make the sf package available;  
package maintainers should consider adding sf to Suggests:.  
The sp package is now running under evolution status 2  
 (status 2 uses the sf package in place of rgdal)

```{r}  
us\_pyramid\_data <- get\_estimates(  
 geography = "state",  
 product = "characteristics",  
 breakdown = c("SEX", "AGEGROUP"),  
 breakdown\_labels = TRUE,  
 year = 2019  
) %>%  
 filter(str\_detect(AGEGROUP, "^Age"),  
 SEX != "Both sexes") %>%  
 group\_by(NAME) %>%  
 mutate(prop = value / sum(value, na.rm = TRUE)) %>%  
 ungroup() %>%  
 mutate(prop = ifelse(SEX == "Male", -prop, prop))  
  
ggplot(us\_pyramid\_data, aes(x = prop, y = AGEGROUP, fill = SEX)) +   
 geom\_col(width = 1) +   
 theme\_minimal() +   
 scale\_fill\_manual(values = c("darkred", "navy")) +   
 facet\_geo(~NAME, grid = "us\_state\_with\_DC\_PR\_grid2",  
 label = "code") +   
 theme(axis.text = element\_blank(),  
 strip.text.x = element\_text(size = 8)) +   
 labs(x = "",   
 y = "",   
 title = "Population Structure\nBy Age and Sex",   
 fill = "",   
 caption = "Data source: US Census Bureau\npopulation estimates & tidycensus R package")  
```



* **plotly** - a library for interactive visualization drawing on the **htmlwidgets** package and the Plotly JavaScript data visualization library
  + with **plotly**’s ggplotly() function, you can make any existing ggplot() object into an interactive chart

```{r message=TRUE, warning=TRUE}  
library(plotly)  
```

Attaching package: 'plotly'

The following object is masked from 'package:ggplot2':  
  
 last\_plot

The following object is masked from 'package:stats':  
  
 filter

The following object is masked from 'package:graphics':  
  
 layout

```{r message=TRUE, warning=TRUE}  
#ggplotly(utah\_pyramid)  
```

## 4.8 Learning more about visualization

Author recommends the following readings:

* Munzer. 2014. *Visualization Analysis and Design.* CRC Press. <https://doi.org/10.1201/b17511>
* Knaflic. 2015. “Storytelling with Data,” October. <https://doi.org/10.1002/9781119055259>
* Healy. 2019. *Data Visualization: A Practical Introduction.* Princeton University Press. <https://socviz.co/>
* Wilke. 2019. *Fundamentals of Data Visualization.* O’Reilly Media. <https://clauswilke.com/datavis/>

## 4.9 Exercises

* Choose a different variable in the ACS and/or a different location and create a margin of error visualization of your own.

```{r}  
years <- 2005:2019  
names(years) <- years  
  
la\_value <- map\_dfr(years, ~{  
 get\_acs(  
 geography = "county",  
 variables = "B25077\_001",  
 state = "CA",  
 county = "Los Angeles",  
 year = .x,  
 survey = "acs1"  
 )  
}, .id = "year")  
```

Getting data from the 2005 1-year ACS

The 1-year ACS provides data for geographies with populations of 65,000 and greater.

Getting data from the 2006 1-year ACS

The 1-year ACS provides data for geographies with populations of 65,000 and greater.

Getting data from the 2007 1-year ACS

The 1-year ACS provides data for geographies with populations of 65,000 and greater.

Getting data from the 2008 1-year ACS

The 1-year ACS provides data for geographies with populations of 65,000 and greater.

Getting data from the 2009 1-year ACS

The 1-year ACS provides data for geographies with populations of 65,000 and greater.

Getting data from the 2010 1-year ACS

The 1-year ACS provides data for geographies with populations of 65,000 and greater.

Getting data from the 2011 1-year ACS

The 1-year ACS provides data for geographies with populations of 65,000 and greater.

Getting data from the 2012 1-year ACS

The 1-year ACS provides data for geographies with populations of 65,000 and greater.

Getting data from the 2013 1-year ACS

The 1-year ACS provides data for geographies with populations of 65,000 and greater.

Getting data from the 2014 1-year ACS

The 1-year ACS provides data for geographies with populations of 65,000 and greater.

Getting data from the 2015 1-year ACS

The 1-year ACS provides data for geographies with populations of 65,000 and greater.

Getting data from the 2016 1-year ACS

The 1-year ACS provides data for geographies with populations of 65,000 and greater.

Getting data from the 2017 1-year ACS

The 1-year ACS provides data for geographies with populations of 65,000 and greater.

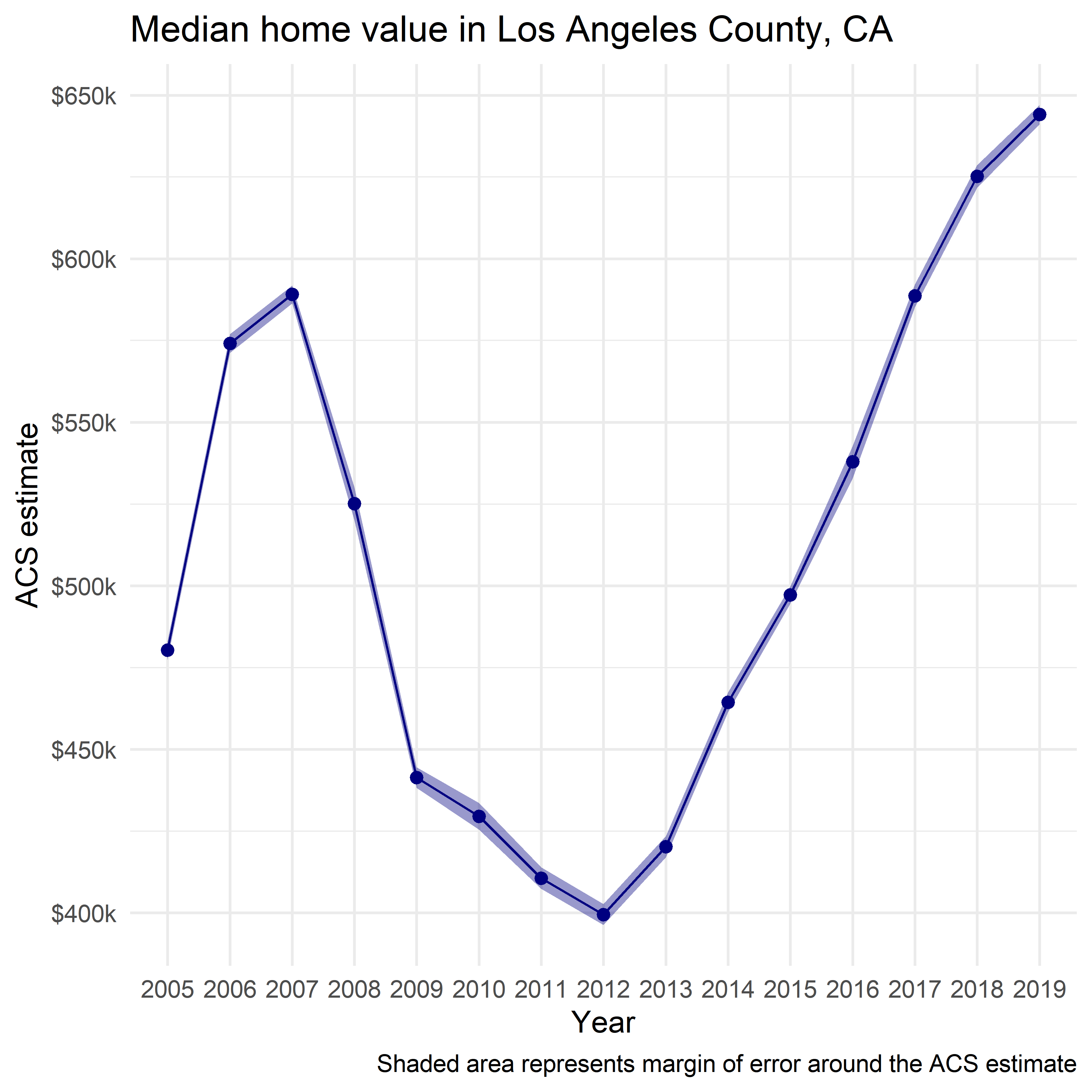
Getting data from the 2018 1-year ACS

The 1-year ACS provides data for geographies with populations of 65,000 and greater.

Getting data from the 2019 1-year ACS

The 1-year ACS provides data for geographies with populations of 65,000 and greater.

```{r}  
ggplot(la\_value, aes(x = year, y = estimate, group = 1)) +   
 geom\_ribbon(aes(ymax = estimate + moe, ymin = estimate - moe),   
 fill = "navy",  
 alpha = 0.4) +   
 geom\_line(color = "navy") +   
 geom\_point(color = "navy", size = 2) +   
 theme\_minimal(base\_size = 12) +   
 scale\_y\_continuous(labels = label\_dollar(scale = .001, suffix = "k")) +   
 labs(title = "Median home value in Los Angeles County, CA",  
 x = "Year",  
 y = "ACS estimate",  
 caption = "Shaded area represents margin of error around the ACS estimate")  
```



* Modify the population pyramid code to create a different, customized population pyramid. You can choose a different location (state or county), different colors/plot design, or some combination!

```{r}  
la\_age\_sex <- get\_estimates(  
 geography = "county",  
 state = "CA",  
 county = "Los Angeles",  
 product = "characteristics",  
 breakdown = c("SEX", "AGEGROUP"),  
 breakdown\_labels = TRUE,  
 year = 2019  
)  
  
la\_age\_sex\_filtered <- filter(la\_age\_sex, str\_detect(AGEGROUP, "^Age"),  
 SEX != "Both sexes") %>%   
 mutate(value = ifelse(SEX == "Male", -value, value))  
  
la\_pyramid <- ggplot(la\_age\_sex\_filtered,  
 aes(x = value,  
 y = AGEGROUP,  
 fill = SEX,  
 text = paste0(  
 "POPULATION: ",  
 label\_comma()  
 (abs(value))  
 )  
 )  
 ) +   
 geom\_col(width = 0.90, alpha = 0.75) +  
 theme\_minimal(base\_family = "Verdana", base\_size = 12) +  
 scale\_x\_continuous(  
 labels = ~label\_number(scale = 0.001, suffix = "k")(abs(.x)),  
 breaks = c(-400000, -200000, 200000, 400000),  
 limits = 500000 \* c(-1, 1)) +  
 scale\_y\_discrete(labels = ~str\_remove\_all(.x, "Age\\s|\\syears")) +  
 scale\_fill\_manual(values = c("darkred", "navy")) +  
 labs(x = "",  
 y = "2019 Census Bureau population estimate",  
 title = "Population structure\nLos Angeles County, CA",  
 fill = "",  
 caption = "Data source: US Census Bureau\npopulation estimates & tidycensus R package")  
  
  
la\_pyramid # %>% ggplotly(tooltip = c("AGEGROUP", "SEX", "text"))  
```

