Module 1

Day 2

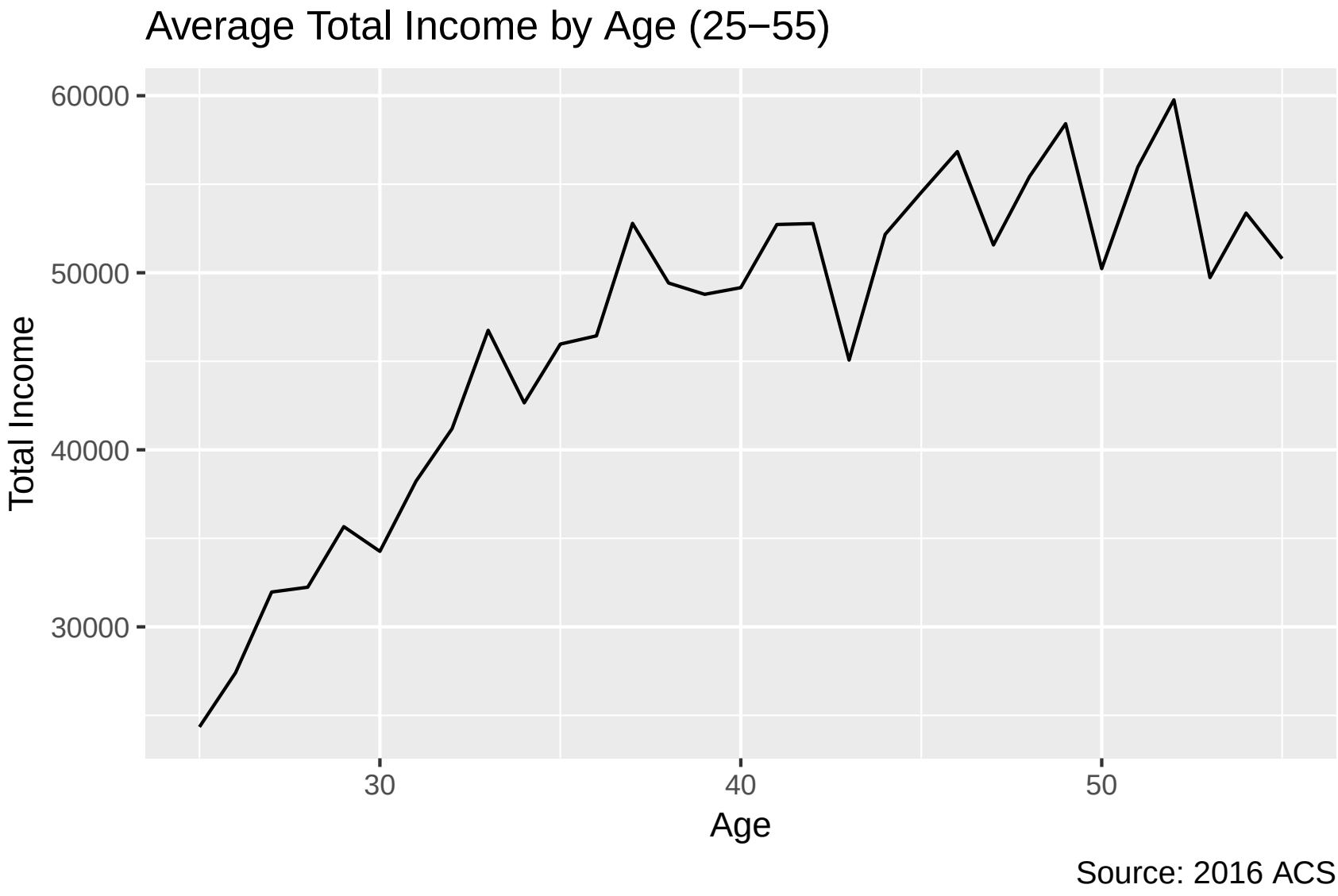
Recap last week

- `read_csv()`
- `ggplot()`, `aes()`, `geom_line()`, `geom_point()`, `geom_density()`
- `filter()`, `select()`, `mutate()`

Recap Exercise 1

- Set your working directory
- Read in the data file "acs_2016_age.csv"
- Create a new data frame filtered to ages between 25-55
- Make a line plot with AGE on the x axis and INCTOT on the y axis
- Label your chart appropriately

Recap last week



Recap Exercise 2

- Add a variable to the original data frame that is the gap between total and wage income
- Make a line plot with AGE on the x axis and inc_gap on the y axis
- Label your chart appropriately

Recap last week

Goals for Today

Economics:

- Discover a relationship between wages and education

Goals for Today

R:

- Renaming variables
- Recoding categorical variables
- The %>% operator
- Sorting data
- Sequences
- group_by() and summarise()

rename()

- Start by reading in "acs_2016_sample.csv"

```
acs_2016_sample <- read_csv("acs_2016_sample.csv")
```

- Using the glimpse(), head(), or str() function look at the variable names.
- Are they intuitive? Easy to understand?

Exercise: rename()

```
df <- rename(df, new_variable_name = old_variable_name)
```

Create a new data frame called acs_2016_cleaned and change the names:

- STATEFIP = state_code
- SEX = sex
- AGE = age
- RACE = race
- HISPAN = hispanic
- EDUC = education
- EMPSTAT = employment_status
- INCTOT = total_income
- INCWAGE = wage_income
- UHRSWORK = hrs_worked
- WKSWORK2 = weeks_worked

Now that we've renamed the variables, let's clean up the data frame!

Exercise: select()

From acs_2016_cleaned select:

- state_code
- sex
- age
- race
- hispanic
- education
- employment_status
- total_income
- wage_income
- hrs_worked
- weeks_worked

Exercise: filter()

Filter acs_2016)cleaned to observations where:

- total_income <= 1,000,000
- wage_income <= 1,000,000
- age >= 25 & age <= 55
- hrs_worked > 0

We can begin to get an understanding of the variables in our data frame using the summary() function.

Exercise: mutate(), ifelse()

Using mutate() and ifelse() recode the sex variable so that

- 1 -> “Male”
- 2 -> “Female”

Exercise: Mean Income by Sex

Now that we've cleaned our data frame, we can run some summary statistics.

A simple question: What is the average income for men?

- Create a new data frame that only contains observations for men called male_data
- select() the column total_income
- find the mean() and median()

Mean Income by Sex

```
mean(male_data$total_income, na.rm = T)
```

```
## [1] 66555.44
```

```
median(male_data$total_income, na.rm = T)
```

```
## [1] 47000
```

- What does it mean that our average income and our median income are so different?
- Why is the average income so much higher?

%>%

To write more efficient code we can use nested functions f(g(x))

```
nested_data <- select(filter(acs_2016_cleaned,
                             sex == "Male"), total_income)

mean(nested_data$total_income, na.rm = TRUE)
```

```
## [1] 66555.44
```

We get the same answer, but this is still not great.

%>%

Luckily we have another solution, the pipe operator: %>%

```
pipe_data <-
  acs_2016_cleaned %>%
  filter(sex == "Male") %>%
  select(total_income)

mean(nested_data$total_income, na.rm = T)
```

```
## [1] 66555.44
```

%>%

The pipe operator says: "Take the left side of the pipe and make it the first argument on the right side"

```
filter(acs_2016_cleaned, sex == "Male")
```

is the same as

```
acs_2016_cleaned %>%
  filter(sex == "Male")
```

- By default the %>% will put the left side as the first argument
- We can control this with the dot `(.)` as a placeholder

%>%

Let's use seq() as an example:

```
seq(start, end, interval)
```

```
seq(25, 30, 1)
```

```
## [1] 25 26 27 28 29 30
```

```
25 %>% seq(30, 1) # 25 is read as the first argument
```

```
## [1] 25 26 27 28 29 30
```

```
30 %>% seq(25, ., 1) # 30 is directed by the placeholder
```

```
## [1] 25 26 27 28 29 30
```

Exercise: %>%

Create a new data frame called female_data and use the %>% operator calculate the mean income for women

```
mean(female_data$total_income, na.rm = T)
```

```
## [1] 44048.09
```

summarise()

- We want to be able to calculate summary statistics for groups of data (i.e. men/women, white/non-white)
- What are the arguments to the summarise() function?

```
summary_df <- summarise(df, new_variable_name = mean(variab
```

summarise()

- Here we are using the summarise() function to calculate the mean and median income for the male_data data frame.

```
male_data %>%
  summarise(mean_inc = mean(total_income, na.rm = T),
            median_inc = median(total_income, na.rm = T))
```

```
## # A tibble: 1 x 2
##   mean_inc median_inc
##      <dbl>      <dbl>
## 1    66555      47000
```

Exercise: summarise()

- Now do the same calculation but for the female_data data frame.

```
## # A tibble: 1 x 2
##   mean_inc median_inc
##      <dbl>      <int>
## 1    44048      33300
```

group_by(), summarise()

- This was a very cumbersome process for calculating the mean and median income for two subgroups of data.
- Lucky for us dplyr knows how to understand groups!
- We can tell dplyr what our group variables are by piping group_by() to summarise()

```
group_by(variable_name)
```

- What grouping variable should we use to find median and mean income for males *and* females?

group_by(), summarise()

```
acs_2016_cleaned %>%
  group_by(sex) %>%
  summarise(mean_inc = mean(total_income, na.rm = T),
            median_inc = median(total_income, na.rm = T))
```

```
## # A tibble: 2 x 3
##   sex    mean_inc median_inc
##   <chr>     <dbl>      <dbl>
## 1 Female    44048      33300
## 2 Male      66555      47000
```

mutate(), ifelse(), case_when()

Now let's go back and clean up our data for race and education so that we can calculate some more interesting summary stats.

- Use the codebook to see what the numbers mean for Educational attainment [general version]
- How do we want to break up the education variable into categories?

mutate(), ifelse(), case_when()

I propose:

- education <= 5 ~ "No HS Diploma"
- education == 6 ~ "HS Diploma"
- education %in% c(7,8,9) ~ "Some College"
- education == 10 ~ "College Degree"
- education == 11 ~ "Graduate Degree"

mutate(), ifelse(), case_when()

Using ifelse statements in this case can get messy very quickly! Better to use case_when() because it does not require an else statement:

```
acs_2016_cleaned <-
  acs_2016_cleaned %>%
  mutate(education =
           case_when(education <= 5 ~ "No HS Diploma",
                     education == 6 ~ "HS Diploma",
                     education %in% c(7,8,9) ~ "Some College
                     education == 10 ~ "College Degree",
                     education == 11 ~ "Graduate Degree"))
```

Exercise: mutate(), ifelse(), case_when()

- Let's use the same function to recode the race variable!
- The codebook has the definitions for Hispanic origin [general version] and Race [general version]

I propose the following categories:

- race == 1 & hispanic == 0 ~ "Non-Hispanic White"
- race == 1 & hispanic %in% c(1,2,3,4) ~ "Hispanic White"
- race == 2 ~ "Black"
- race == 3 ~ "Native American"
- race %in% c(4,5,6) ~ "Asian"
- race %in% c(7,8,9) ~ "Other/Multiracial"

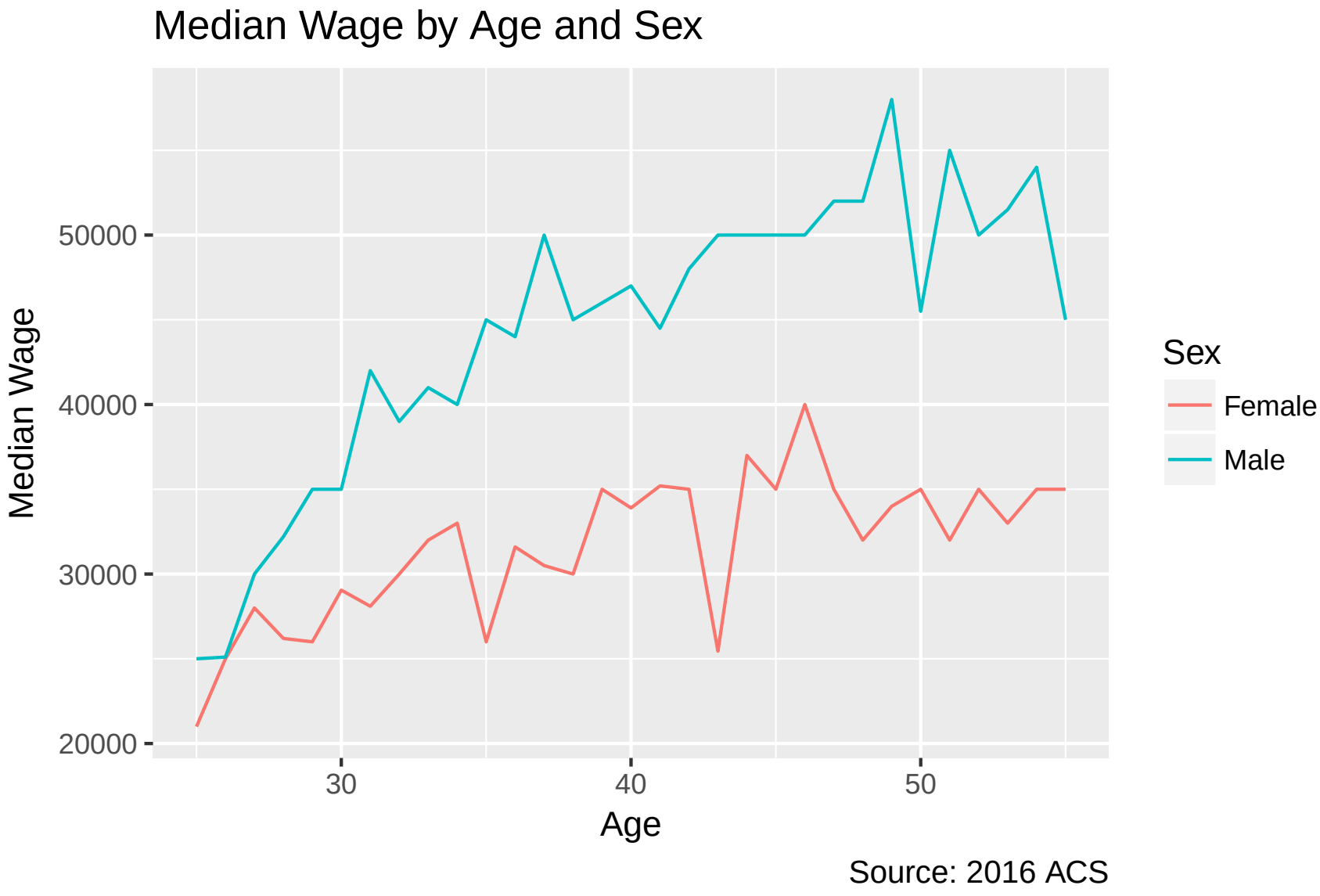
Code up these categories using case_when()

Exercise: group_by(), summarise() 1

Now that we have our data all cleaned up, we can calculate some summary data and plot it:

- Create a data frame called wage_age that contains the median wage income by age and sex
- What grouping variables should you use?
- Make a line plot of the data with age on the x axis, median_wage on the y axis and sex on the color axis
- Be sure to label your chart appropriately

group_by(), summarise()



Exercise: group_by(), summarise() 2

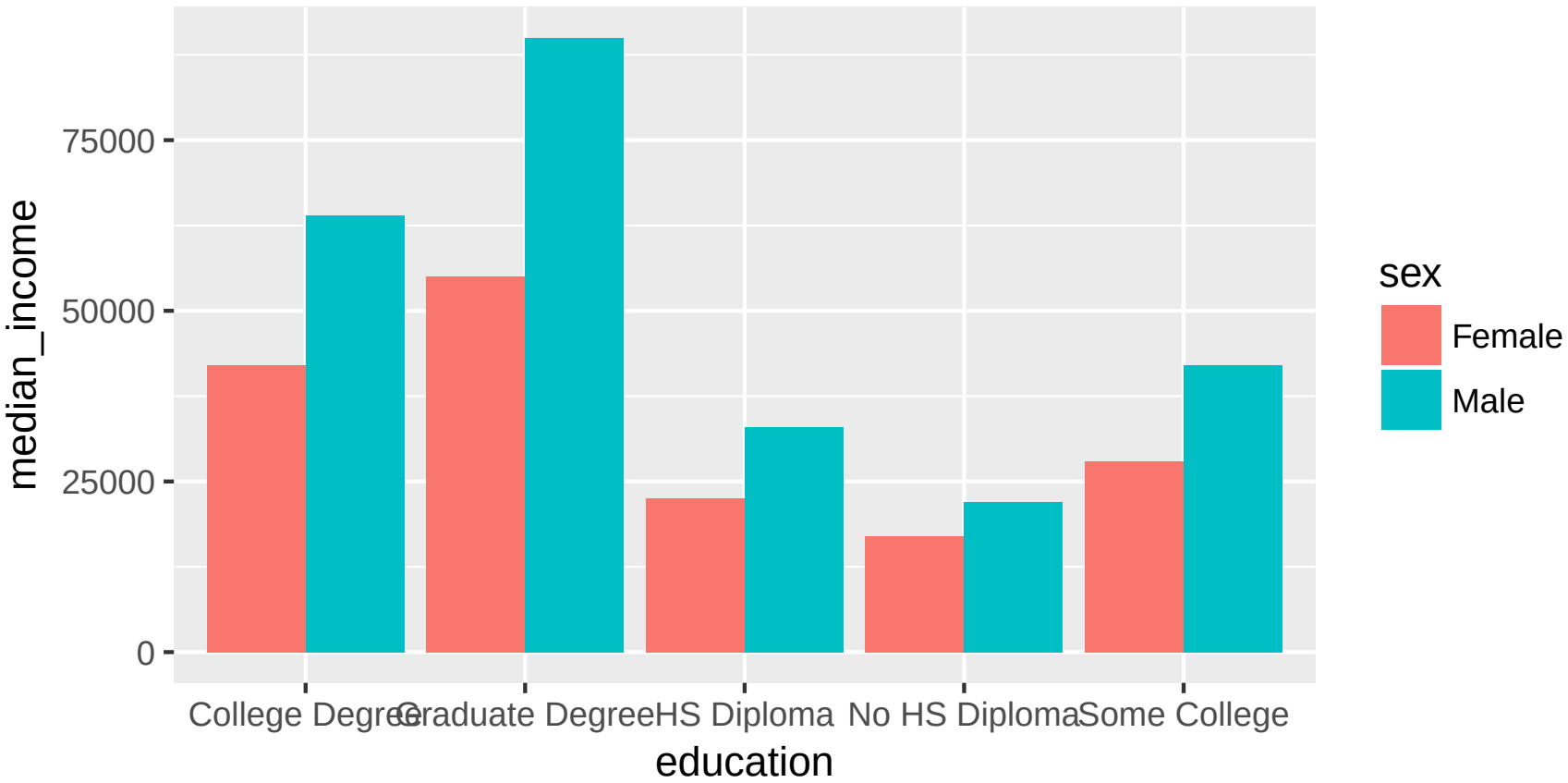
We can also do the same with education and gender

- Create a data frame called wage_education that contains the median wage income by education level and sex
- What grouping variables should you use?

This time we want to make a bar plot of this data. Which geom should we use?

geom_col()

```
gds_plot <- wage_eduction %>%
    ggplot(aes(x = education, y = median_income, fill = sex
    geom_col(position = "dodge")
gds_plot
```

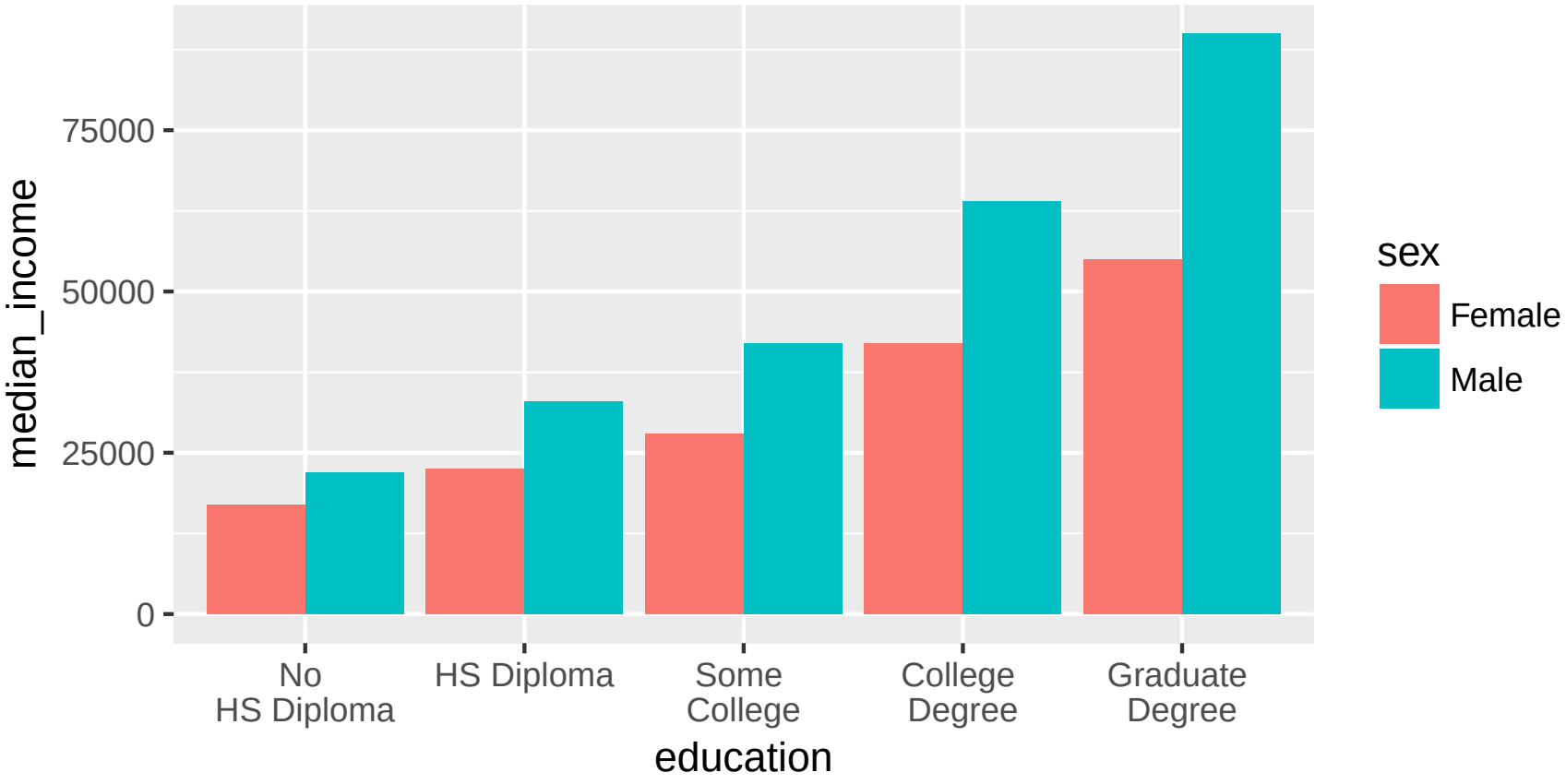


- What are some things we will want to change about this chart?

scale_x_discrete()

- OUR X AXIS LABELS ARE TERRIBLE!
- We want them to go in some sort of order and not run into each other
- How can we fix this? using the `scale_x_discrete` function!
- We can change the order of the variables using the limits argument and the labels using the labels argument.

scale_x_discrete()

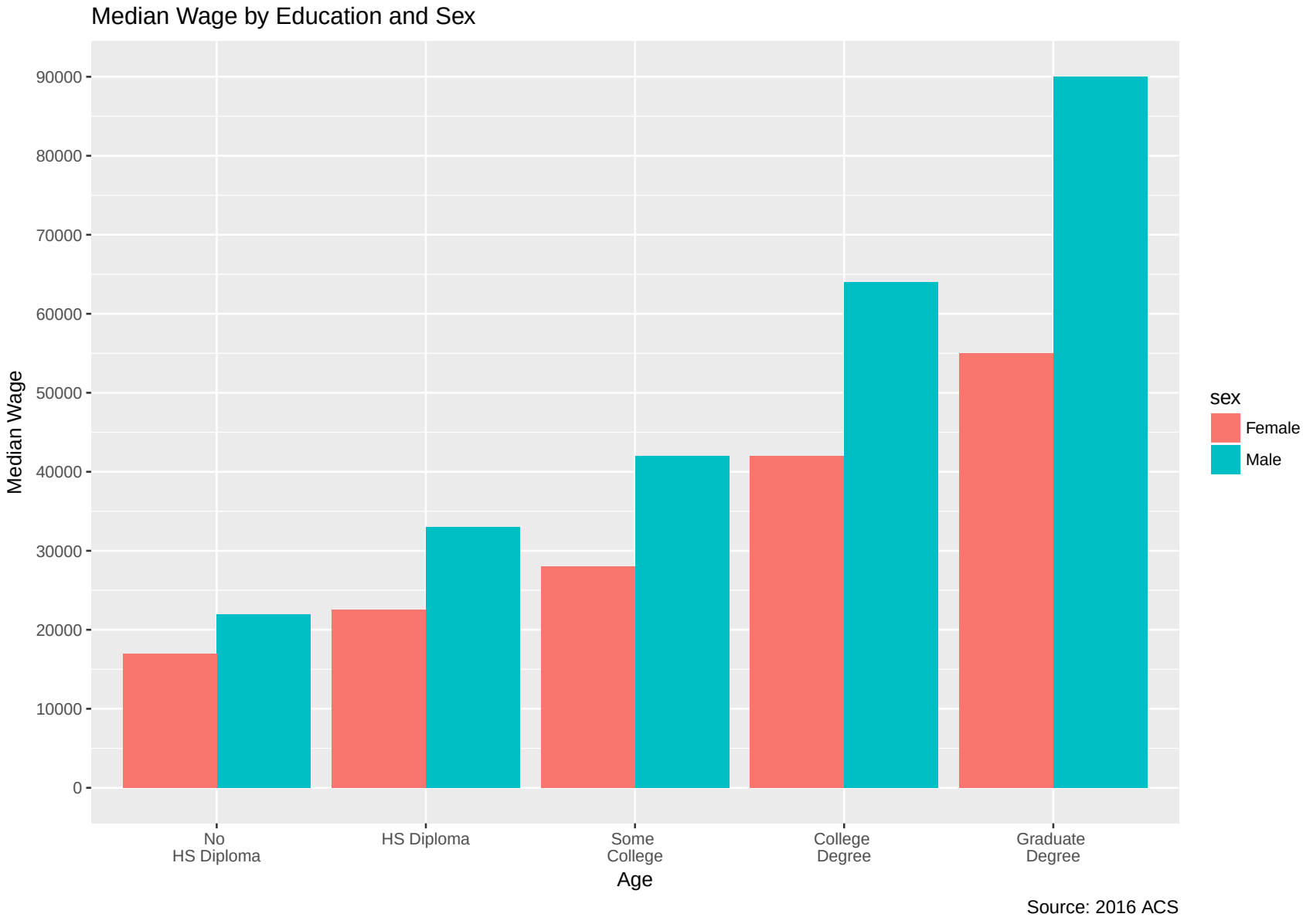


scale_y_continuous()

Now I want to change the y scale to go in $10,000 increments instead of $25,000 increments. How?

- Using the `scale_y_continuous()` function
- The argument we want to use is breaks which we want to set to a sequence
- Please also add appropriate axis labels and a title

scale_y_continuous()



Recap Exercise

Make the same plot but instead of using sex as the color variable use race. You will need to:

- Calculate the median wage per hour for each racial group at each education level using mutate(), group_by() and summarise()
- Assume a person works their average number of
- Use geom_col() to create a bar chart
- Use scale_x_discrete(), scale_fill_discrete(), and scale_y_continuous() to fix up the axis
- Be sure to include labels and a title

Recap Exercise