Introduction to Dplyr

Federal Reserve Board of Governors
Howard University

Introduction

- Last week - visualization with ggplot2
- Today - Dplyr
  - Data Analysis
  - Data Manipulation
- More background on this topic can be found in *R 4 Data Science* chapter 5 http://r4ds.had.co.nz/transform.html

Today's Question

- American Community Survey for the state of California
  - income, education, age, gender,
  - 2010-2014
- How does income differ for men and women?
- How does income differ by age?

Data

- `ca_acs.csv`
  - Read in this file as a data.frame and call it: “acs”
- Disclaimer: Data are taken from a random 20,000 subsample of the 5-year ACS Public Use data for California; results should be considered only as instructional exercises

Examining our Data

- How many columns?
- Names of our columns?
- Data is messy
- Install and attach the dplyr package

Guiding Question

- Easier to analyze data if you have a question/goal
- How do wages and income differ for men and women in California by age?
  - Line plot with age on the x-axis and dollars on the y-axis
- First step?

Analyzing our columns

- names are cryptic acronyms
  - Data dictionaries
  - Census provides a data dictionary for the ACS at www2.census.gov/programs-surveys/acs/tech_docs/pums/data_dict/PUMS_Data_Dictionary_2011-2015.pdf
- How do wages and income differ for men and women in California by age

Selecting our Columns

- Age (AGEP), Gender (SEX), wages (WAGP), income (PINCP)
  - inflation adjustment factor (ADJINC)
  - weighting factor (PWGTP)

dplyr::select

- `Select()` allows us to subset columns from data.frames
  - `select(.data, ...)`
  - `.data` = data.frame
  - `...` = columns
- `Select()` example

Deselecting Columns

- Putting the minus sign, -, in front of a column name deselects it
  - As output you would get every column except for the one you deselect

```
names(income_data)
```

```
Output: [1] "AGEP"   "SEX"    "WAGP"   "WKHP"   "PINCP"
Output: [6] "ADJINC" "PWGTP"  "PUMA10"
```

```
deselected_data <- select(income_data, -PUMA10, -AGEP)
names(deselected_data)
```

```
Output: [1] "SEX"    "WAGP"   "WKHP"   "PINCP"  "ADJINC"
Output: [6] "PWGTP"
```

In class exercise: Select

- Now it's your turn, create a data.frame called `exercise_1_data` of the following columns from `acs`: `SEX`, `WKHP`, `ADJINC`, `PWGTP`, `WAGP`

Filtering our data

- Remove unusable observations
  - rows with NA values in them
- `filter()`
  - `filter(.data, ...)`
    - `.data` = data.frame
    - `...` = logical conditions (>, <, <=, >=, etc...)
- `Filter()` example 1

Boolean Logic

- Comparison statements that evalute to `TRUE` or `FALSE`
  - string multiple together with `&` (and) and `|` (or) operators
  - $3 < 4$ evaluates to TRUE,
  - $3 < 4$ & $5 > 7$ evaluates to FALSE
    - When making a statement with `&` remember: TRUE & TRUE $=$ TRUE while any other combination $=$ FALSE
  - However, the statement $3 < 4 \mid 5 > 7$ evaluates to TRUE
    - Only one statement must be TRUE when using `|` for the whole statement to be TRUE

Boolean logic: exercises

- Remember that == is "equal to" in R, since = also has assignment powers in R, like <-
- `%in%` is the "in" operator, for example, 2 %in% c(1, 2, 3) is TRUE
  - `filter(acs, AGEP < 30)`
  - `filter(acs, AGEP < 30 & SEX == 1)`
  - `filter(acs, AGEP < 25 | AGEP > 65)`
  - `filter(acs, Occ %in% c("Computer/Math", "Legal"))`

Back to Filter

- `filter()` can take multiple criteria
  - Example above use |
- Adding more conditions with , is the same as &

```
identical(filter(income_data, AGEP == 24 & SEX == 1),
    filter(income_data, AGEP == 24, SEX == 1))
```

`Output: [1] TRUE`

What do you think happens if you do not provide a search criterion to `filter()`?

In class exercise: Filter

- Re-assign your exercise_1_data data.frame to exercise_2_data and filter out any `NA` values in the columns
- Additionally, make sure that your WKHP variable is greater or equal to 10 and less than or equal to 60
- Your data should have the following dimensions:

8598, 5

Creating a Workflow

- Need to execute multiple steps
- One method: create new objects

```
object1 <- select(acs, AGEP, SEX, WAGP,
                  PINCP, ADJINC, PWGTP)

object2 <- filter(object1,
                  !(is.na(AGEP) | is.na(PINCP)))
```

Combining statements

- Creating new objects each time
  - Time consuming and confusing
- Nesting functions
  - $f(g(x)) = 5$

Nested Functions example

```
income_data_nest <- filter(select(acs, AGEP, SEX, WAGP,
                    PINCP, ADJINC, PWGTP),
                    !(is.na(AGEP) | is.na(PINCP)))

identical(object2, income_data_nest)
```

```
Output: [1] TRUE
```

Also confusing!

Pipe Operators

- %>% operator from dplyr
  - output of the left side becomes input on the right side
  - x %>% f() == f(x)

```
income_data_pipe <- acs %>%
    select(AGEP, SEX, WAGP, PINCP, ADJINC, PWGTP) %>%
    filter(!(is.na(AGEP) | is.na(PINCP)))

identical(income_data_nest, income_data_pipe)
```

```
Output: [1] TRUE
```

Pipes Explained

- By default the pipe operator takes the output from the left function and passes it to the right function **as the first argument**
  - Can change manually with the `.` operator

```
25 %>% seq(30, by = 1)
```

```
Output: [1] 25 26 27 28 29 30
```

```
25 %>% seq(30, ., by = -1)
```

```
Output: [1] 30 29 28 27 26 25
```

- The first example is `seq(25, 30, by = 1)` while the second is `seq(30, 25, by = -1)`

Mutating our data

- Adjust our income and wage data to be in constant 2014 USD
  - Move the decimal in ADJINC over
  - Recode our gender column to be "Male" and "Female"
  - Create buckets for our weekly hours column
- `mutate()`
  - calculate new columns, overwrite current columns, or delete columns
- `mutate()` example
  - Can combine mutations into a single `mutate()` call

Updating Income and Wages

- Can refer to a column created earlier in a `mutate()` call within that same function call!
- `mutate()` example 2

Updating Gender

- Use a function within `mutate()`
  - ifelse()
  - plyr::round_any()
    - Need plyr installed
    - package::function notation

```
Output:   SEX gender WKHP Hours
Output: 1   2 Female   40    40
Output: 2   1   Male   40    40
Output: 3   2 Female   40    40
Output: 4   1   Male   60    60
Output: 5   2 Female   30    30
Output: 6   2 Female   45    45
```

case_when() a close cousin of ifelse

- Ifelse is good when there are two options (i.e. male or female)
- What do you do when you have many different possibilities?

```
adjusted_data %>%
    mutate(gender = case_when(SEX == 2 ~ "Female",
                              SEX == 1 ~ "Male")) %>%
    select(SEX, gender) %>%
    head(3)
```

```
Output:   SEX gender
Output: 1   2 Female
Output: 2   1   Male
Output: 3   2 Female
```

In class Exercise: Mutate

- Now it's time to update your data: exercise_2_data
- Update SEX, ADJINC, and WKHP as seen previously
- Update WAGP to account for the adjustment factor

```
Output:    SEX WKHP   ADJINC PWGTP     WAGP Gender Hours
Output: 1    2   40 1.024037    15 23552.85 Female    40
Output: 2    1   40 1.024037    22 15360.56   Male    40
Output: 3    2   40 1.024037     7 38913.41 Female    40
Output: 4    1   60 1.008425    53 25412.31   Male    60
Output: 5    2   30 1.094136    10  5470.68 Female    30
Output: 6    2   45 1.008425    19 93783.52 Female    45
```

Deleting your columns

- using `mutate()` set your column = NULL

```
"SEX" %in% names(gender_hours_data)
```

```
Output: [1] TRUE
```

```
gender_hours_data <- gender_hours_data %>%
    mutate(SEX = NULL,
           WKHP = NULL)

"SEX" %in% names(gender_hours_data)
```

```
Output: [1] FALSE
```

What is the difference between this method and deselecting data?

The fun part: Summarize

- Generate our summary statistics
- Calculate functions over entire column(s) in a data.frame.
  - mean, median, etc...
- Summarise example

```
Output:      income    wages
Output: 1 54836.58 48077.21
```

Grouping

- Summarize by itself isn't all that useful
- Income for each gender/age combination.
- `group_by()`
  - When used in conjunction with summarize or other functions allows us to calculate statistics within different groups
- `ungroup()` resets our data

```r
income_data_summary <- gender_hours_data %>%
    group_by(AGEP, gender) %>%
    dplyr::summarise(income = weighted.mean(PINCP,
                                              PWGTP),
              wages = weighted.mean(WAGP, PWGTP))
head(income_data_summary, 3)
```

```
Output: # A tibble: 3 x 4
Output: # Groups:   AGEP [2]
Output:    AGEP gender   income     wages
Output:   <int>  <chr>    <dbl>     <dbl>
Output: 1    20 Female 10694.49  9342.122
Output: 2    20   Male 10492.21 10178.223
Output: 3    21 Female 11375.18 11053.704
```

dplyr::summarise()? weigthed.mean() instead of mean()?

In class exercise: group_by and summarize

- Using the `group_by()` and `summarize()` functions find the mean values for each gender/hour combination
- save your output as `exercise_4_data`
- Your data should look like the following:

```
Output: # A tibble: 6 x 3
Output: # Groups:   Gender [1]
Output:   Gender Hours     wages
Output:    <chr> <dbl>     <dbl>
Output: 1 Female    10  4640.393
Output: 2 Female    15 10555.757
Output: 3 Female    20 11856.598
Output: 4 Female    25 16512.445
Output: 5 Female    30 21234.293
Output: 6 Female    35 25792.914
```

Finding the top and bottom with arrange

- sorts rows
  - The default method is ascending sorting
  - descending is possible with the `desc()` function
- `arrange()` examples

In Class Exercise: arrange

- Using your `exercise_4_data` find the bottom five observations for average wages
  - You do not have to save your output to a variable
  - what is their gender, the hours worked?

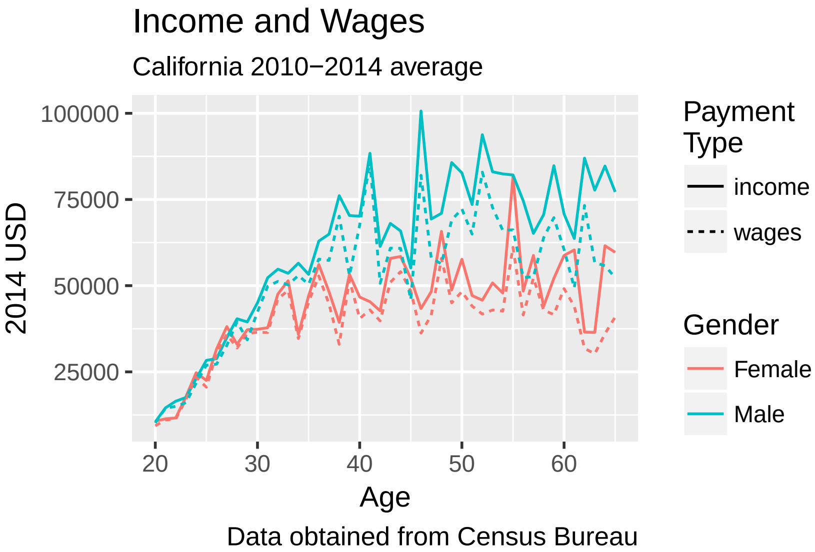
```
Output: # A tibble: 5 x 3
Output: # Groups:   Gender [2]
Output:   Gender Hours     wages
Output:    <chr> <dbl>     <dbl>
Output: 1 Female    10  4640.393
Output: 2   Male    10  5108.755
Output: 3   Male    15  8083.622
Output: 4   Male    20 10326.199
Output: 5 Female    15 10555.757
```

Reshaping our Data

- Long data is easier to use with ggplot2
- Tidyr::gather() and Tidyr::spread()
  - gather takes multiple columns and collapses into key/value pairs (wide -> long)
    - duplicates other data as needed
  - spread takes key-value pairs and spreads the data across multiple columns (long -> wide)
- gather() example

Answering our Question

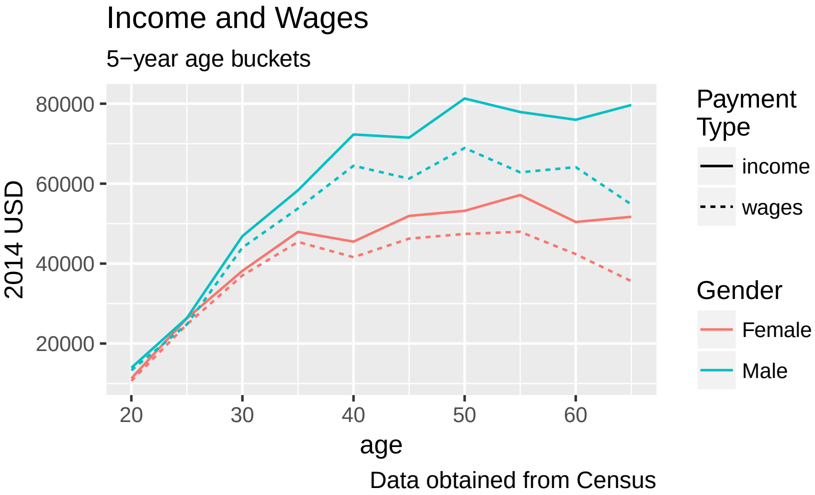
- Use ggplot to analyze our output



Understanding your analysis

- Difference between income and wages?
- Men and women?
  - What do we not take into account?
- Jagged peaks and valleys?
- Is this accurate?

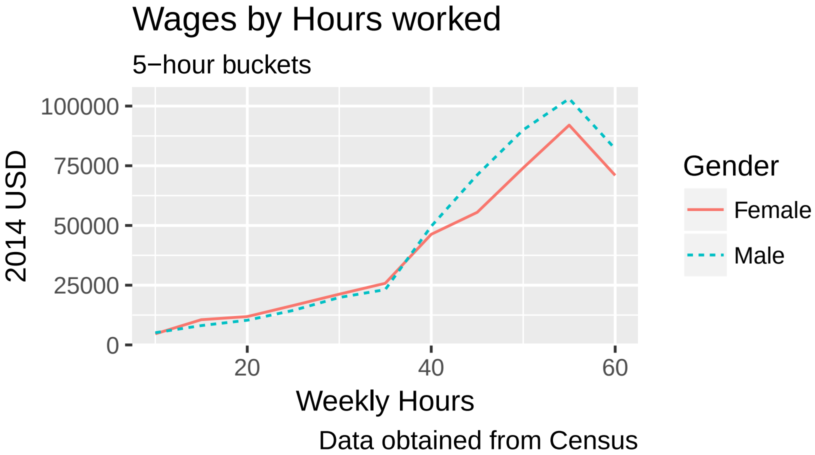
Widening our age ranges



What is the message of this chart compared with our first chart using only 1-year age buckets?

In Class Exercise

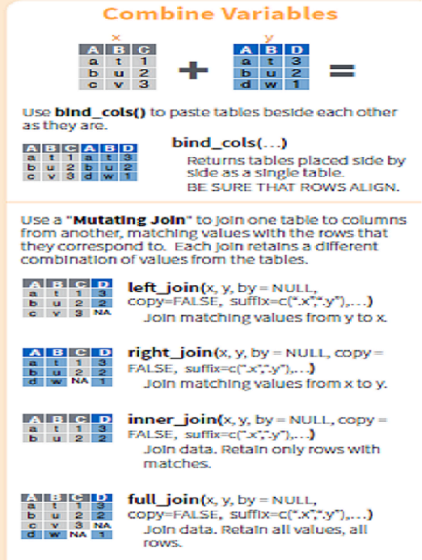
- Using the `exercise_3_data` create a chart showing mean wages for men and women by hours worked



What message does this chart show about wage inequality compared with our previous chart? Any other questions?

Joins

- Need multiple datasets to talk to each other



New Questions

- In what county do men have the highest wages? Women?
- In what county do men and women have the largest income ratio?
- Need to add county level data to our acs data
  - `puma10_county_xwalk.csv`

```
Output:   puma12   county
Output: 1    101 Alameda
Output: 2    102 Alameda
Output: 3    103 Alameda
```

Joining our Data

- We want to use an inner join

```
Output: [1] "AGEP"   "WAGP"   "PINCP"  "ADJINC" "PWGTP"
Output: [6] "PUMA10" "gender" "Hours"  "county"
```

County Level Statistics

- Which county has the highest male wages?

```
Output: # A tibble: 6 x 3
Output:             county    Female     Male
Output: *            <chr>     <dbl>    <dbl>
Output: 1            Marin 114325.96 69574.81
Output: 2           Placer  70621.52       NA
Output: 3 San Francisco    52621.21 98917.28
Output: 4       San Mateo  58002.22 93219.94
Output: 5     Santa Clara  56311.29 77299.68
Output: 6         Ventura        NA 70940.12
```

County Income Ratio

- Male/Female income ratio (male wages/female wages)

```
Output: # A tibble: 6 x 2
Output: # Groups:   county [6]
Output:             county    ratio
Output:              <chr>    <dbl>
Output: 1         Imperial  2.644490
Output: 2 San Luis Obispo  2.471640
Output: 3          Ventura  2.376310
Output: 4           Merced  2.292796
Output: 5             Yolo  2.120836
Output: 6             Lake  1.967070
```

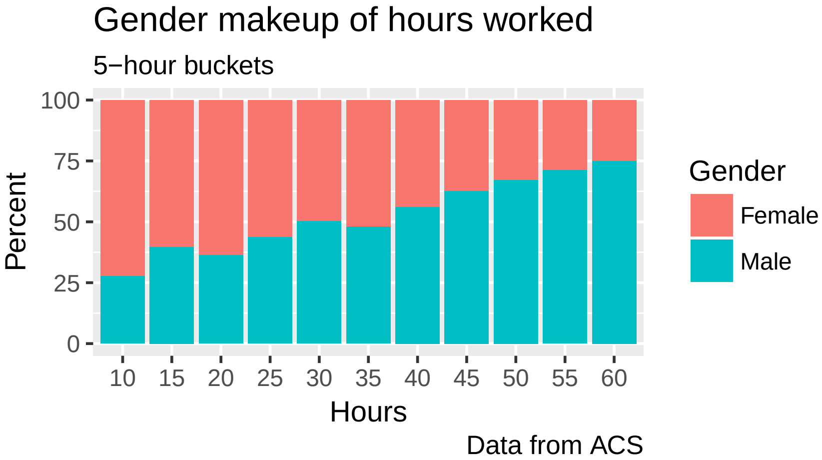
Let's now go back to how hours worked impacts wages

Challenge exercise 1

- Wages seem to increase dramatically with hours worked
- Next step: bar chart showing the percentage of men and women in each 5 hour bucket

1. Find the total number of observations in each hour/gender combination
   - group_by() and dplyr::summarise()
2. Change your grouping to just the hour buckets
   - use mutate to calculate the total number of observations in each bucket and the the percentage of men/women in each bucket
3. Plot with `ggplot()` and `geom_bar()` etc..., your `y` value should be the percentage
   - in the `geom_bar()` make sure to include `stat = "identity"`

Challenge Exercise 1 Answer

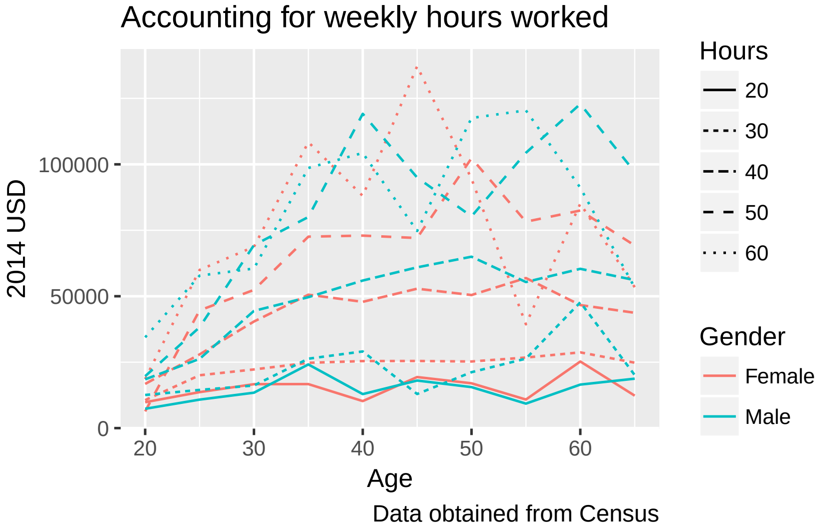


New insights? Further questions?

Challenge Exercise 2

- Using dplyr and ggplot analyze how wage (WAGP) changes over age between men and women
  - control for number of hours worked
    - 10 hour buckets instead of 5
    - Limit to people who work between 20 and 60 hours per week
  - Bucket sizes for age should be 5 years
- PWGTP
- You will need to start with the original ACS data to answer this

Challenge Exercise 2: Answer



Insights? Further Questions? Is this a “good” chart?