Introduction to data wrangling 2

dplyr and friends

Daniel Anderson
Week 3, Class 2

UNIVERSITY OF OREGON

Slides available at: http://www.datalorax.com/vita/ds/ds1-slides/w3p2/

Agenda

- Questions
- Workflows (quickly)
- Importing data
- A little more dplyr
- Lab

Reading in data

- We'll use the {here} package
  - Allows you to not worry about paths so much
  - Helps reproducibility

Reading in data

- We'll use the {here} package
  - Allows you to not worry about paths so much
  - Helps reproducibility
- In your R Studio project directory
  - Create a new folder called "data"
  - Place the data for today there

Look at what data files are there

```
library(here)
list.files(here("data"))
```

```
## [1] "ais.xlsx"                  "dirty_data.xlsx"
## [3] "ecls-k_samp.sav"           "exam1.csv"
## [5] "Fatality.txt"              "Project_Reads_Scores.csv"
## [7] "seda.csv"                  "sesame13.sav"
## [9] "star.csv"
```

{rio}

- Super nice package - most of the time, it just works, regardless of the source file type.
- (If this isn't astounding to you, you obviously haven't struggled for hours to read in data properly)

Example: these all work! Try it out and verify for yourself!

```
library(rio)
exam1 <- import(here("data", "exam1.csv"))
eclsk <- import(here("data", "ecls-k_samp.sav"))
fatality <- import(here("data", "Fatality.txt"))
```

Read directly from the web!

```
pubschls <- import("https://github.com/datalorax/ncme_18/raw/master/data/pub
head(pubschls)
```

```
##         CDSCode NCESDist NCESSchool StatusType  County
## 1 1100170000000   691051         NA     Active Alameda
## 2 1100170109835   691051      10546     Closed Alameda
## 3 1100170112607   691051      10947     Active Alameda
## 4 1100170118489   691051      12283     Closed Alameda
## 5 1100170123968   691051      12844     Active Alameda
## 6 1100170124172   691051      12901     Active Alameda
##                               District
## 1 Alameda County Office of Education
## 2 Alameda County Office of Education
## 3 Alameda County Office of Education
## 4 Alameda County Office of Education
## 5 Alameda County Office of Education
## 6 Alameda County Office of Education
##                                          School
## 1
## 2                            FAME Public Charter
## 3        Envision Academy for Arts & Technology
## 4 Aspire California College Preparatory Academy
## 5       Community School for Creative Education
```

Export just as easily!

```
export(exam1, here("data", "exam1.sav"))
export(exam1, here("data", "exam1.txt"))
export(exam1, here("data", "exam1.dta"))
```

`convert()`

- Another really useful feature is `convert()`, which just takes a file of one type and converts it to another.

convert()

- Another really useful feature is `convert()`, which just takes a file of one type and converts it to another.
- Say your advisor uses SPSS, but her/his colleague uses Stata. They might use some proprietary (and expensive) software like SAS/CONNECT. Instead, just run this one line of code and voila!

```
convert(here("data", "ecls-k_samp.sav"),
        here("data", "ecls-k_samp.dta"))
```

How is this all working?

The full list of supported formats is below:

Format	Typical Extension	Import Package	Export Package	Installed by Default
Comma-separated data	.csv	data.table	data.table	Yes
Pipe-separated data	.psv	data.table	data.table	Yes
Tab-separated data	.tsv	data.table	data.table	Yes
SAS	.sas7bdat	haven	haven	Yes
SPSS	.sav	haven	haven	Yes
Stata	.dta	haven	haven	Yes
SAS XPORT	.xpt	haven		Yes
SPSS Portable	.por	haven		Yes
Excel	.xls	readxl		Yes
Excel	.xlsx	readxl	openxlsx	Yes
R syntax	.R	base	base	Yes
Saved R objects	.RData, .rda	base	base	Yes
Serialized R objects	.rds	base	base	Yes
Epiinfo	.rec	foreign		Yes
Minitab	.mtp	foreign		Yes
Systat	.syd	foreign		Yes
"XBASE" database files	.dbf	foreign	foreign	Yes
Weka Attribute-Relation File Format	.arff	foreign	foreign	Yes
Data Interchange Format	.dif	utils		Yes
Fortran data	no recognized extension	utils		Yes
Fixed-width format data	.fwf	utils	utils	Yes
gzip comma-separated data	.csv.gz	utils	utils	Yes
CSVY (CSV + YAML metadata header)	.csvy	csvy	csvy	No
Feather R/Python interchange format	.feather	feather	feather	No
Fast Storage	.fst	fst	fst	No
JSON	.json	jsonlite	jsonlite	No
Matlab	.mat	rmatio	rmatio	No
OpenDocument Spreadsheet	.ods	readODS	readODS	No
HTML Tables	.html	xml2	xml2	No
Shallow XML documents	.xml	xml2	xml2	No
YAML	.yml	yaml	yaml	No
Clipboard	default is tsv	clipr	clipr	No
Google Sheets	as Comma-separated data			

Looking at the import documentation

```
?import
```

import {rio} R Documentation

Import

Description

Read in a data.frame from a file

Usage

```
import(file, format, setclass, which, ...)
```

So… let's go look at the original packages more!

Arguments

`file`	A character string naming a file, URL, or single-file .zip or .tar archive.
`format`	An optional character string code of file format, which can be used to override the format inferred from `file`. Shortcuts include: "," (for comma-separated values), ";" (for semicolon-separated values), and "|" (for pipe-separated values).
`setclass`	An optional character vector specifying one or more classes to set on the import. By default, all the return object is always a "data.frame". Allowed values for this might be "tbl_df", "tbl", or "tibble" (if using dplyr) or "data.table" (if using data.table). Other values are ignored such that a data.frame is returned.
`which`	This argument is used to control import from multi-object files; as a rule `import` only ever returns a single data frame. (Use `import_list` to import multiple data frames from a multi-object file.) If `file` is a compressed directory, `which` can be either a character string specifying a filename or an integer specifying which file (in locale sort order) to extract from the compressed directory. For Excel spreadsheets, this can be used to specify a sheet number. For .Rdata files, this can be an object name. For HTML files, which table to exract (from document order). Ignored otherwise. A character string value will be used as a regular expression, such that the extracted file is the first match of the regular expression against the file names in the archive.
`...`	Additional arguments passed to the underlying import functions. For example, this can control column classes for delimited file types, or control the use of haven for Stata and SPSS or readxl for Excel (.xlsx) format. See details below.

Maintaining labels

- In SPSS, numeric data are often encoded with labels
- rio (and {haven}) allow you to transform the data into the character/factor verion

```
library(tidyverse)
eclsk %>%
    select(child_id:ethnic) %>%
    head()
```

```
##   child_id teacher_id school_id k_type school_type sex ethnic
## 1 0842021C    0842T02      0842      1           0   0      2
## 2 0905002C    0905T01      0905      1           1   0      5
## 3 0150012C    0150T01      0150      1           1   1      2
## 4 0556009C    0556T01      0556      1           1   1      4
## 5 0089013C    0089T04      0089      1           0   0      1
## 6 1217001C    1217T13      1217      0           0   1      6
```

```
eclsk %>%
    characterize() %>%
    select(child_id:ethnic) %>%
    head()
```

```
##   child_id teacher_id school_id   k_type school_type    sex
## 1 0842021C    0842T02      0842 full-day      public   male
## 2 0905002C    0905T01      0905 full-day     private   male
## 3 0150012C    0150T01      0150 full-day     private female
## 4 0556009C    0556T01      0556 full-day     private female
## 5 0089013C    0089T04      0089 full-day      public   male
## 6 1217001C    1217T13      1217 half-day      public female
##                                     ethnic
## 1 BLACK OR AFRICAN AMERICAN, NON-HISPANIC
## 2                                    ASIAN
## 3 BLACK OR AFRICAN AMERICAN, NON-HISPANIC
## 4            HISPANIC, RACE NOT SPECIFIED
## 5                   WHITE, NON-HISPANIC
## 6 NATIVE HAWAIIAN, OTHER PACIFIC ISLANDER
```

- Run the following line of code - what do you get?
- Note the `setclass` argument

```
library(rio)
reads <- import(here("data", "Project_Reads_Scores.csv"),
                setclass = "tbl_df")
```

Project Reads data

```
head(reads)
```

```
## # A tibble: 6 x 25
##   `Test Year`         `Test Type` `Test Site` `Student ID` `Pre-Test Score`
##   <chr>               <chr>       <chr>       <chr>                   <int>
## 1 06/01/2016 12:00:… YEAR END    VIRDEN      Virden 1                   43
## 2 06/01/2016 12:00:… YEAR END    VIRDEN      Virden 2                   46
## 3 06/01/2016 12:00:… YEAR END    VIRDEN      Virden 3                   39
## 4 06/01/2016 12:00:… YEAR END    VIRDEN      Virden 4                   35
## 5 06/01/2016 12:00:… YEAR END    VIRDEN      Virden 5                   46
## 6 06/01/2016 12:00:… YEAR END    VIRDEN      Virden 6                   35
## # ... with 20 more variables: `Pre-Test (%)` <chr>, `Post-Test
## #   Score` <int>, `Post-Test (%)` <chr>, `Percentage Change` <chr>, `Unit
## #   1 Score` <int>, `Unit 1 (%)` <chr>, `Unit 2 Score` <int>, `Unit 2
## #   (%)` <chr>, `Unit 3 Score` <int>, `Unit 3 (%)` <chr>, `Unit 4
## #   Score` <int>, `Unit 4 (%)` <chr>, `Unit 5/6 Score` <int>, `Unit 5/6
## #   (%)` <chr>, `Unit 7 Score` <int>, `Unit 7 (%)` <chr>, `Unit 8
## #   Score` <int>, `Unit 8 (%)` <chr>, `Total Score` <int>, `Total (%)
## #   Correct` <chr>
```

Clean the names

```
library(magrittr)
reads  %<>%
    janitor::clean_names()

head(reads)
```

```
## # A tibble: 6 x 25
##   test_year test_type test_site student_id pre_test_score pre_test_percent
##   <chr>     <chr>     <chr>     <chr>                <int> <chr>
## 1 06/01/20… YEAR END  VIRDEN    Virden 1                43 29%
## 2 06/01/20… YEAR END  VIRDEN    Virden 2                46 31%
## 3 06/01/20… YEAR END  VIRDEN    Virden 3                39 26%
## 4 06/01/20… YEAR END  VIRDEN    Virden 4                35 23%
## 5 06/01/20… YEAR END  VIRDEN    Virden 5                46 31%
## 6 06/01/20… YEAR END  VIRDEN    Virden 6                35 23%
## # ... with 19 more variables: post_test_score <int>,
## #   post_test_percent <chr>, percentage_change <chr>, unit_1_score <int>,
## #   unit_1_percent <chr>, unit_2_score <int>, unit_2_percent <chr>,
## #   unit_3_score <int>, unit_3_percent <chr>, unit_4_score <int>,
## #   unit_4_percent <chr>, unit_5_6_score <int>, unit_5_6_percent <chr>,
## #   unit_7_score <int>, unit_7_percent <chr>, unit_8_score <int>,
## #   unit_8_percent <chr>, total_score <int>, total_percent_correct <chr>
```

Dataset summary

```
install.packages("skimr")
```

```
library(skimr)
```

(demo)

Check out the structure

```
dim(reads)
```

```
## [1] 48 25
```

```
str(reads)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    48 obs. of  25 variables:
##  $ test_year             : chr  "06/01/2016 12:00:00 AM" "06/01/2016 12:00:00 AM
##  $ test_type             : chr  "YEAR END" "YEAR END" "YEAR END" "YEAR END" ...
##  $ test_site             : chr  "VIRDEN" "VIRDEN" "VIRDEN" "VIRDEN" ...
##  $ student_id            : chr  "Virden 1" "Virden 2" "Virden 3" "Virden 4" ...
##  $ pre_test_score        : int  43 46 39 35 46 35 40 39 40 45 ...
##  $ pre_test_percent      : chr  "29%" "31%" "26%" "23%" ...
##  $ post_test_score       : int  92 104 75 115 85 91 96 74 90 86 ...
##  $ post_test_percent     : chr  "61%" "69%" "50%" "77%" ...
##  $ percentage_change     : chr  "32%" "38%" "24%" "54%" ...
##  $ unit_1_score          : int  3 5 4 4 2 5 5 4 6 4 ...
##  $ unit_1_percent        : chr  "12%" "20%" "16%" "16%" ...
##  $ unit_2_score          : int  4 5 4 4 5 5 5 5 4 5 ...
##  $ unit_2_percent        : chr  "16%" "20%" "16%" "16%" ...
##  $ unit_3_score          : int  6 6 6 6 6 7 6 5 5 5 ...
##  $ unit_3_percent        : chr  "24%" "24%" "24%" "24%" ...
##  $ unit_4_score          : int  8 8 9 10 7 7 7 8 8 7 ...
```


dplyr

Oddity

```
reads %>%
 count(student_id)
```

```
## # A tibble: 46 x 2
##    student_id                  n
##    <chr>                   <int>
##  1 All Students (Average)      3
##  2 Jones 1                     1
##  3 Jones 10                    1
##  4 Jones 11                    1
##  5 Jones 12                    1
##  6 Jones 13                    1
##  7 Jones 14                    1
##  8 Jones 15                    1
##  9 Jones 2                     1
## 10 Jones 3                     1
## # ... with 36 more rows
```

filter

Let's remove the summary rows at the end

```
reads %<>%
    filter(student_id != "All Students (Average)")

reads %>%
 count(student_id)
```

```
## # A tibble: 45 x 2
##    student_id     n
##    <chr>      <int>
##  1 Jones 1        1
##  2 Jones 10       1
##  3 Jones 11       1
##  4 Jones 12       1
##  5 Jones 13       1
##  6 Jones 14       1
##  7 Jones 15       1
##  8 Jones 2        1
##  9 Jones 3        1
## 10 Jones 4        1
## # ... with 35 more rows
```

`select()` helper funs

- `starts_with()`
- `ends_with()`
- `contains()`

starts_with

Let's use the Project Reads data

```
reads %>%
    select(starts_with("test"))
```

```
## # A tibble: 45 x 3
##    test_year                test_type test_site
##    <chr>                    <chr>     <chr>
##  1 06/01/2016 12:00:00 AM YEAR END  VIRDEN
##  2 06/01/2016 12:00:00 AM YEAR END  VIRDEN
##  3 06/01/2016 12:00:00 AM YEAR END  VIRDEN
##  4 06/01/2016 12:00:00 AM YEAR END  VIRDEN
##  5 06/01/2016 12:00:00 AM YEAR END  VIRDEN
##  6 06/01/2016 12:00:00 AM YEAR END  VIRDEN
##  7 06/01/2016 12:00:00 AM YEAR END  VIRDEN
##  8 06/01/2016 12:00:00 AM YEAR END  VIRDEN
##  9 06/01/2016 12:00:00 AM YEAR END  VIRDEN
## 10 06/01/2016 12:00:00 AM YEAR END  VIRDEN
## # ... with 35 more rows
```

ends_with

```
reads %>%
    select(ends_with("score"))
```

```
## # A tibble: 45 x 10
##    pre_test_score post_test_score unit_1_score unit_2_score unit_3_score
##             <int>           <int>        <int>        <int>        <int>
##  1             43              92            3            4            6
##  2             46             104            5            5            6
##  3             39              75            4            4            6
##  4             35             115            4            4            6
##  5             46              85            2            5            6
##  6             35              91            5            5            7
##  7             40              96            5            5            6
##  8             39              74            4            5            5
##  9             40              90            6            4            5
## 10             45              86            4            5            5
## # ... with 35 more rows, and 5 more variables: unit_4_score <int>,
## #   unit_5_6_score <int>, unit_7_score <int>, unit_8_score <int>,
## #   total_score <int>
```

contains

```
reads %>%
    select(contains("test"))
```

```
## # A tibble: 45 x 7
##    test_year           test_type test_site pre_test_score pre_test_percent
##    <chr>               <chr>     <chr>               <int> <chr>
##  1 06/01/2016 12:00:0… YEAR END  VIRDEN                 43 29%
##  2 06/01/2016 12:00:0… YEAR END  VIRDEN                 46 31%
##  3 06/01/2016 12:00:0… YEAR END  VIRDEN                 39 26%
##  4 06/01/2016 12:00:0… YEAR END  VIRDEN                 35 23%
##  5 06/01/2016 12:00:0… YEAR END  VIRDEN                 46 31%
##  6 06/01/2016 12:00:0… YEAR END  VIRDEN                 35 23%
##  7 06/01/2016 12:00:0… YEAR END  VIRDEN                 40 27%
##  8 06/01/2016 12:00:0… YEAR END  VIRDEN                 39 26%
##  9 06/01/2016 12:00:0… YEAR END  VIRDEN                 40 27%
## 10 06/01/2016 12:00:0… YEAR END  VIRDEN                 45 30%
## # ... with 35 more rows, and 2 more variables: post_test_score <int>,
## #   post_test_percent <chr>
```

Mix types

You can mix types and helper funs. You can also use select to rearrange your columns.

```
reads %>%
    select(student_id, 1, starts_with("total"))
```

```
## # A tibble: 45 x 4
##    student_id test_year              total_score total_percent_correct
##    <chr>      <chr>                        <int> <chr>
##  1 Virden 1   06/01/2016 12:00:00 AM         207 41%
##  2 Virden 2   06/01/2016 12:00:00 AM         224 45%
##  3 Virden 3   06/01/2016 12:00:00 AM         193 39%
##  4 Virden 4   06/01/2016 12:00:00 AM         223 45%
##  5 Virden 5   06/01/2016 12:00:00 AM         198 40%
##  6 Virden 6   06/01/2016 12:00:00 AM         210 42%
##  7 Virden 7   06/01/2016 12:00:00 AM         211 42%
##  8 Virden 8   06/01/2016 12:00:00 AM         180 36%
##  9 Virden 9   06/01/2016 12:00:00 AM         206 41%
## 10 Virden 10  06/01/2016 12:00:00 AM         206 41%
## # ... with 35 more rows
```

Summarizing and plotting

- Calculate means by test site

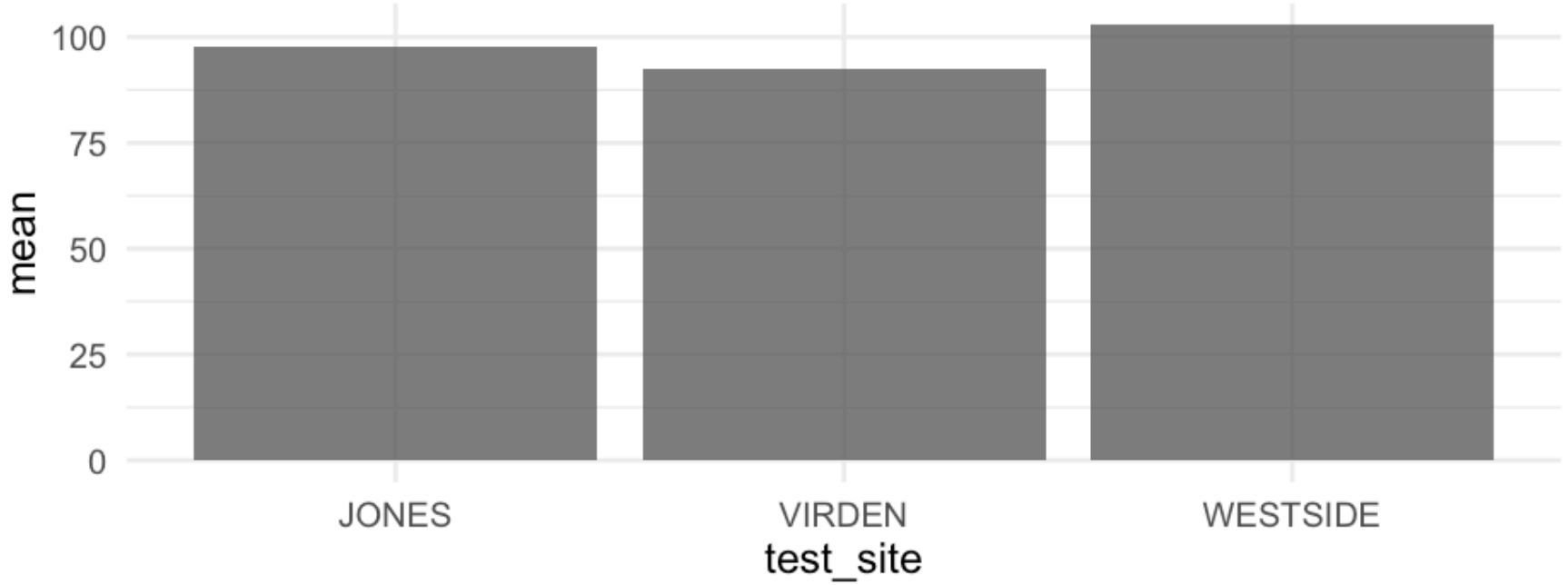
```
reads %>%
    group_by(test_site) %>%
    summarize(mean = mean(post_test_score))
```

```
## # A tibble: 3 x 2
##   test_site  mean
##   <chr>     <dbl>
## 1 JONES      97.8
## 2 VIRDEN     92.5
## 3 WESTSIDE  103.
```

Plot it

```
theme_set(theme_minimal(base_size = 25))

reads %>%
    group_by(test_site) %>%
    summarize(mean = mean(post_test_score)) %>%
    ggplot(aes(test_site, mean)) +
        geom_col(alpha = 0.8)
```



Multi-group

Let's look at *ecslsk*

```
eclsk %<>%
    characterize() %>%
    janitor::clean_names() %>%
    tbl_df()

head(eclsk)
```

```
## # A tibble: 6 x 33
##   child_id teacher_id school_id k_type  school_type sex   ethnic  famtype
##   <chr>    <chr>      <chr>     <chr>   <chr>       <chr> <chr>   <chr>
## 1 0842021C 0842T02    0842      full-d… public      male  BLACK … BIOLOGI…
## 2 0905002C 0905T01    0905      full-d… private     male  ASIAN   BIOLOGI…
## 3 0150012C 0150T01    0150      full-d… private     fema… BLACK … BIOLOGI…
## 4 0556009C 0556T01    0556      full-d… private     fema… HISPAN… BIOLOGI…
## 5 0089013C 0089T04    0089      full-d… public      male  WHITE,… BIOLOGI…
## 6 1217001C 1217T13    1217      half-d… public      fema… NATIVE… BIOLOGI…
## # ... with 25 more variables: numsibs <dbl>, ses_cont <dbl>,
## #   ses_cat <chr>, age <dbl>, t1rscale <dbl>, t1mscale <dbl>,
## #   t1gscale <dbl>, t2rscale <dbl>, t2mscale <dbl>, t2gscale <dbl>,
## #   ir_treadgain <dbl>, ir_tmathgain <dbl>, ir_tgkgain <dbl>,
## #   t1arslit <dbl>, t1arsmat <dbl>, t1arsgen <dbl>, t2arslit <dbl>,
```

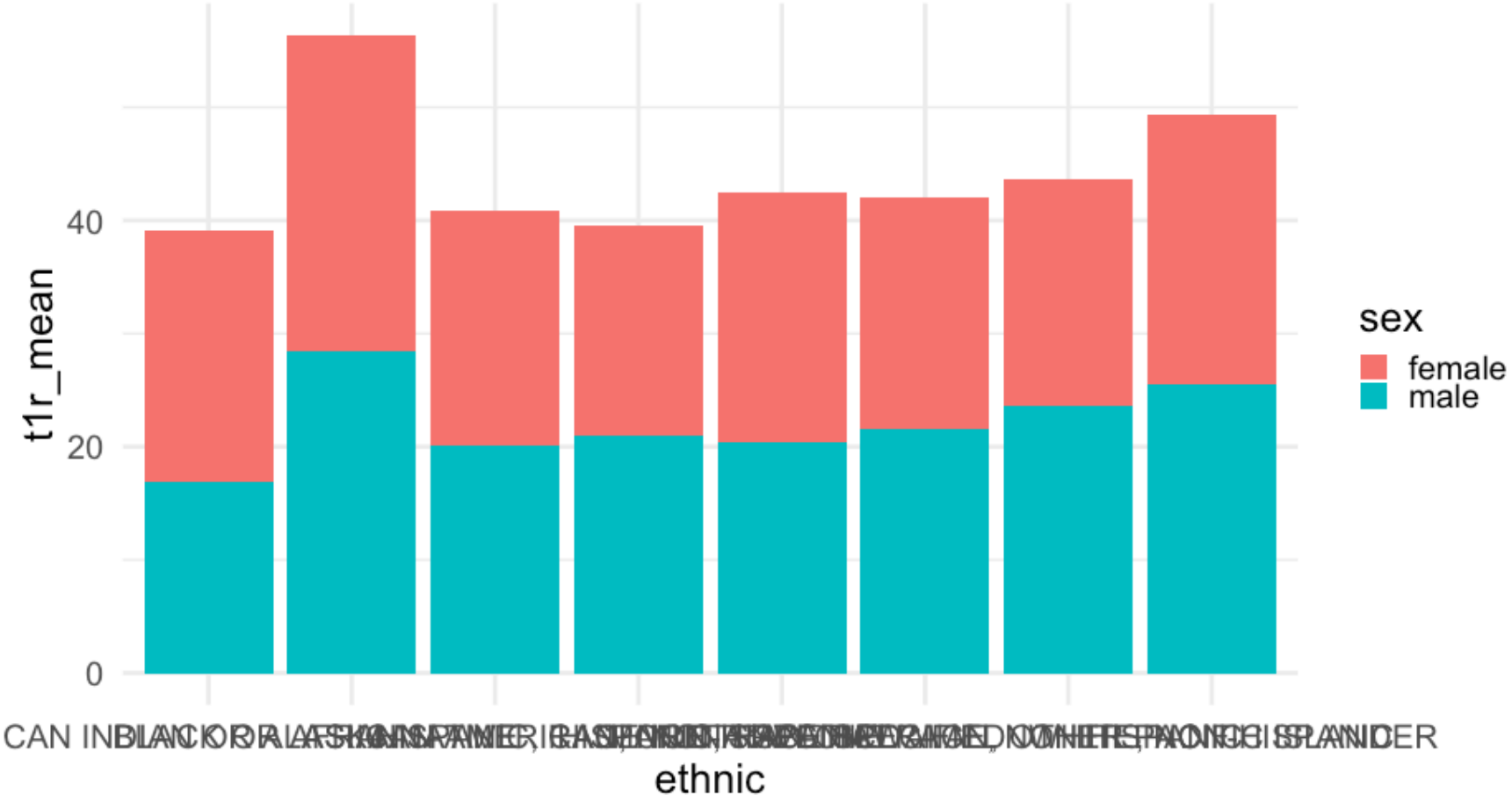
Summarize

```
ecls_smry <- eclsk %>%
    group_by(sex, ethnic) %>%
    summarize(t1r_mean = mean(t1rscale))
ecls_smry
```

```
## # A tibble: 16 x 3
## # Groups:   sex [?]
##    sex    ethnic                                    t1r_mean
##    <chr>  <chr>                                        <dbl>
##  1 female AMERICAN INDIAN OR ALASKA NATIVE              22.2
##  2 female ASIAN                                        28.0
##  3 female BLACK OR AFRICAN AMERICAN, NON-HISPANIC       20.8
##  4 female HISPANIC, RACE NOT SPECIFIED                  18.7
##  5 female HISPANIC, RACE SPECIFIED                      22.2
##  6 female MORE THAN ONE RACE, NON HISPANIC              20.3
##  7 female NATIVE HAWAIIAN, OTHER PACIFIC ISLANDER       20.1
##  8 female WHITE, NON-HISPANIC                           23.8
##  9 male   AMERICAN INDIAN OR ALASKA NATIVE              16.9
## 10 male   ASIAN                                        28.4
## 11 male   BLACK OR AFRICAN AMERICAN, NON-HISPANIC       20.1
## 12 male   HISPANIC, RACE NOT SPECIFIED                  21.0
## 13 male   HISPANIC, RACE SPECIFIED                      20.3
## 14 male   MORE THAN ONE RACE, NON HISPANIC              21.6
## 15 male   NATIVE HAWAIIAN, OTHER PACIFIC ISLANDER       23.6
```

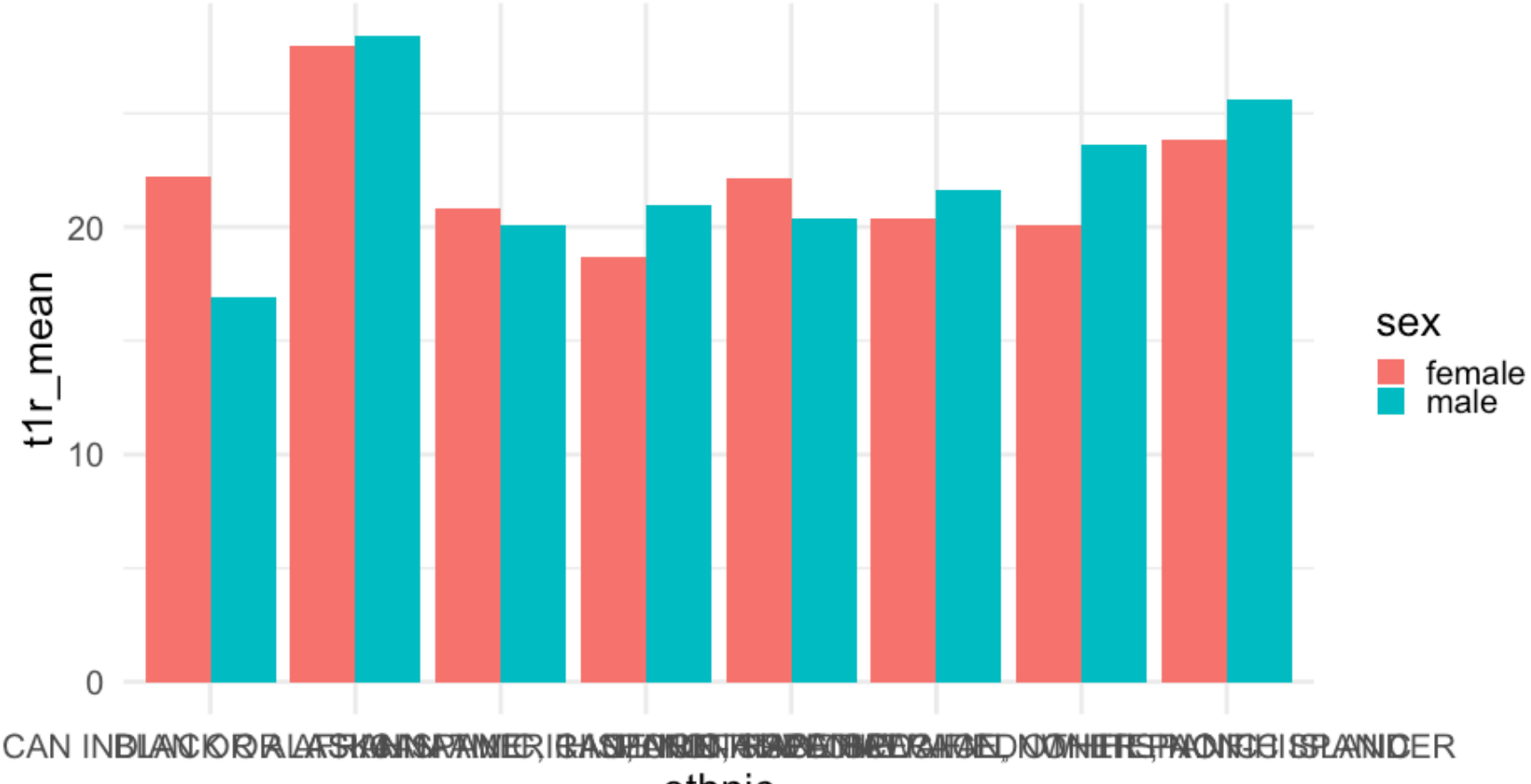
Look for gender effects

```
ggplot(ecls_smry, aes(ethnic, t1r_mean)) +
    geom_col(aes(fill = sex))
```



Look for gender effects

```
ggplot(ecls_smry, aes(ethnic, t1r_mean)) +
    geom_col(aes(fill = sex),
             position = "dodge")
```



Rotating labels

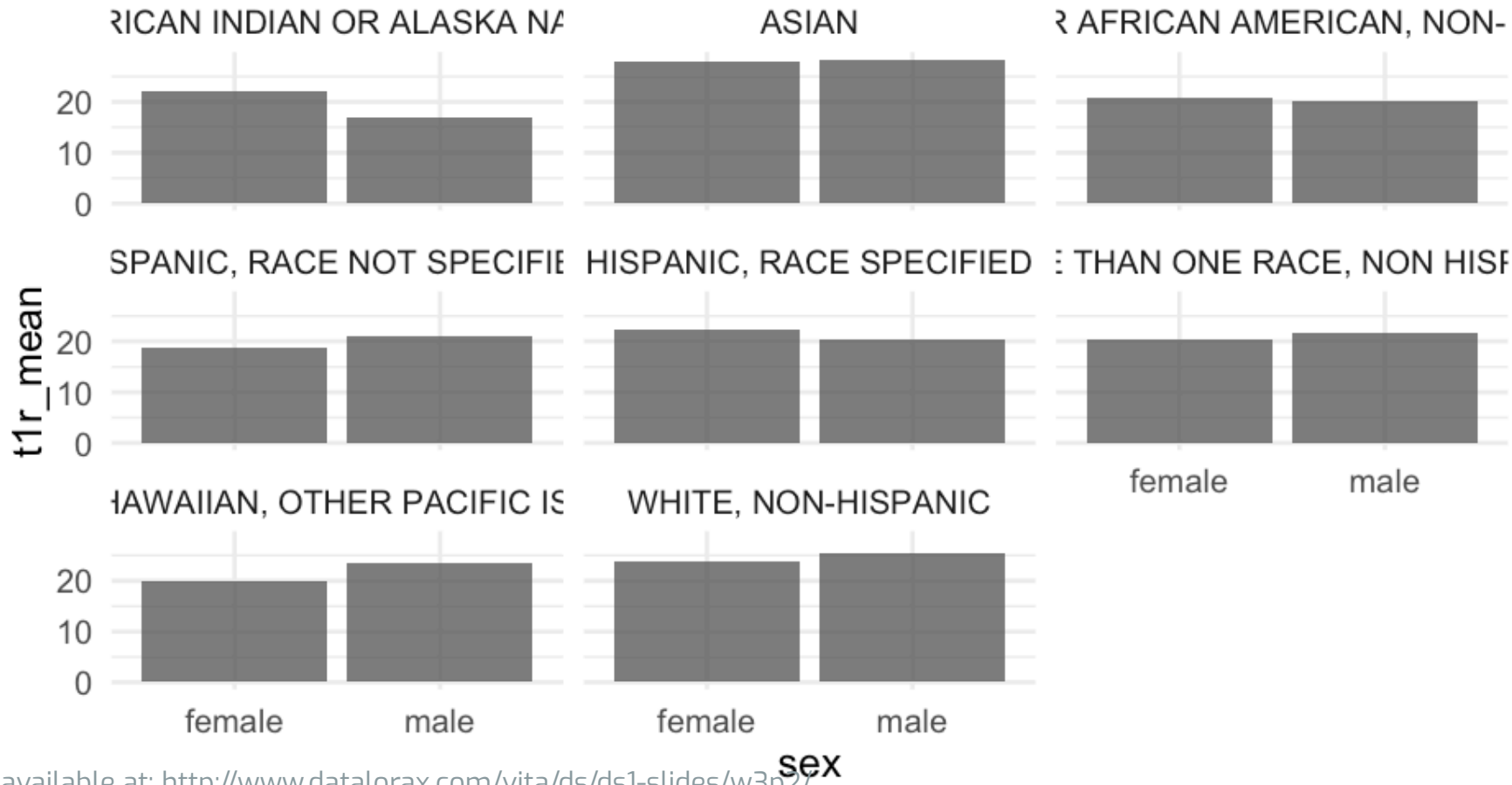
Doesn't show up well on the slide but the relevant bit of code is

```
theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

I have to look this up just about every time (SO answer here)

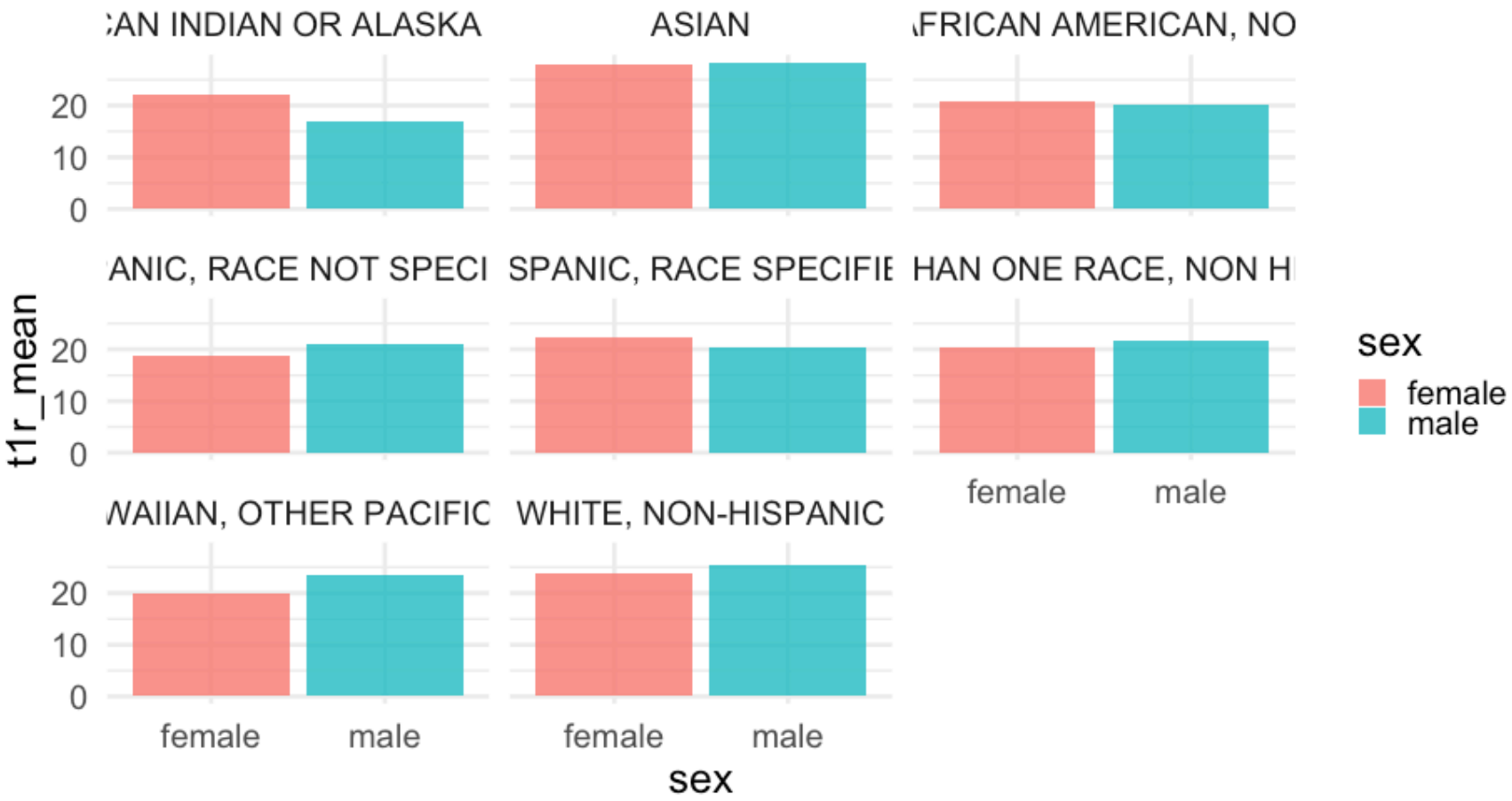
An alternative

```
ggplot(ecls_smry, aes(sex, t1r_mean)) +
    geom_col(alpha = 0.8) +
    facet_wrap(~ethnic)
```



Occasionally doubling up works well, I think

```
ggplot(ecls_smry, aes(sex, t1r_mean, fill = sex)) +
    geom_col(alpha = 0.8) +
    facet_wrap(~ethnic)
```



Final notes on {dplyr}

- We'll be using it all term long
- Verbs can help you gain fluency
- There are also conditional and all-inclusive versions of `mutate`, `select`, and `summarize`.
  - For example `mutate_if(is.character, as.numeric)`, `select_if(is.numeric)`, etc.

Lab