Review & Random

You've already learned *a lot*!

- Objects
- Classes
- Indexing 1-d vectors and 2-d data.frames
- Logical operators & functions
- How to look at the help documentation
- Installing/loading packages
- Making new script files
- Importing your data etc.

This time

- Lots of practice sets; letting it all come together

- Some random tidbits to make your life easier

MIDUS

For this portion, we will use the `midus.csv` dataset.

- Please download this file and save it somewhere you can easily find it.

You should do the exercises in this tutorial *locally* (on your own computer).

MIDUS

ICPSR Find & Analyze Data

Log In/Create Account

FIND DATA   SEARCH/COMPARE VARIABLES   DATA-RELATED PUBLICATIONS   RESOURCES FOR STUDENTS   HELP

Midlife in the United States (MIDUS 2), 2004-2006 (ICPSR 4652)

Version Date: Nov 20, 2017 ❓   Cite this study | Share this page

Principal Investigator(s): ❓
Carol Ryff, University of Wisconsin-Madison; David M. Almeida, Pennsylvania State University; John Ayanian, Harvard University; Deborah S. Carr, University of Wisconsin-Madison; Paul D. Cleary, Harvard University; Christopher Coe, University of Wisconsin-Madison; Richard Davidson, University of Wisconsin-Madison; Robert F. Krueger, University of Minnesota; Marge E. Lachman, Brandeis University; Nadine F. Marks, University of Wisconsin-Madison; Daniel K. Mroczek, Purdue University; Teresa Seeman, University of California-Los Angeles; Marsha Mailick Seltzer, University of Wisconsin-Madison; Burton H. Singer, Princeton University; Richard P. Sloan, Columbia University; Patricia A. Tun, Brandeis University; Maxine Weinstein, Georgetown University; David Williams, University of Michigan

Series:

- Midlife in the United States (MIDUS) Series

https://doi.org/10.3886/ICPSR04652.v7

Version V7

Download ▾	📈 Analyze Online

| At A Glance | Data & Documentation | Variables | Data-related Publications | Export Metadata |

▼ **Project Description**

67,742
Downloads *
Usage Report

742
Data-related
Publications

* past three years

About the MIDUS dataset

Project Description: "The purpose of the study was to investigate the role of behavioral, psychological, and social factors in understanding age-related differences in physical and mental health." Variables available in this data file:

- **Demographic variables**: age, sex
- **Physical health variables**: self-rated physical health, heart problems, father had heart attack, BMI
- **Mental health variables**: self-rated meantal health, self-esteem, life satisfaction *(life overall, work, health, relationship with spouse/partner, relationship with children)*, hostility *(stress reactivity & agression)*

Exercise 1

1. Open a *new* `.R` script file

2. Load the following packages using **code** in your new `.R` file:

   - psych
   - dplyr
   - ggplot2

3. `Get` your working directory

4. If you haven't already, download `midus.csv` from above. Drag and drop the file into a directory that you want to continue using for this class.

5. Import `midus.csv` into R. You can keep the name **midus**.

6. Make sure that the import code ends up in your SCRIPT (`.R`) file.

7. **Save** your new `.R` script file in the same working directory as `midus.csv`. You can name the script whatever you want.

Exercise 1...continued!

1. Check the classes of your variables. Below is a list of each variable and the class they *should* be. Change the class of any that don't match what you see.

   - `ID` should be a factor
   - `sex` should be a factor
   - `age` should be an integer
   - `BMI` should be numeric
   - `physical_health_self` & `mental_health_self` should be integers
   - `self_esteem`, `life_satisfaction`, and `hostility` should all be numeric
   - `heart_self` and `heart_father` should be factors

2. Save your script again. What happens to the text on the tab in RStudio?

Cleaning your global environment

Sometimes, your Global Environment (top right) can fill up with stuff that you don't need anymore. You can clean this!

A) Delete EVERYTHING using the broom icon (next to the Import Dataset tab)

B) Switch to GRID view, check boxes of individual objects you DON'T want, then press the broom icon. *Be sure to switch back to LIST view when finished*

Exercise 1...continued again!

Now...

Clear your global environment, select your entire script file, and RUN it!

...what happens? what does this mean?

Indexing...continued!
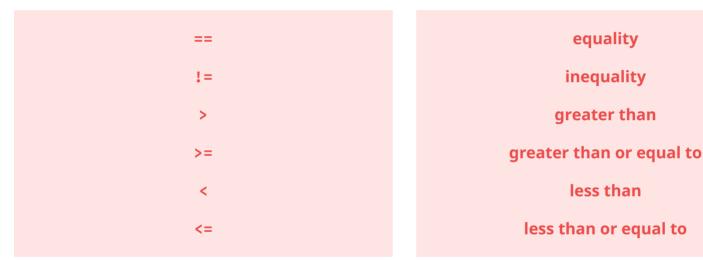
Indexing a **vector** (1 dimension):

- `midus$age[1]`
- The number `1` gives us the **1st** item in the **age** vector that is part of the **midus** data.frame

Indexing a data.frame (2 dimensions):

- `midus[1, 2:3]`
- This gives us the **1st row** and the **2nd thru 3rd columns** of the **midus** data.frame

Logical Operator Review

Returns a value of TRUE or FALSE

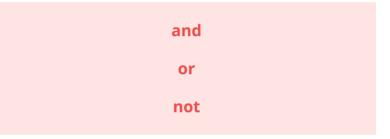|	
==	equality
!=	inequality
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to

A = 3 - *"A is an objec that stores the number 3."*

A == 3 - *"Is A equal to 3?"*

Multiple Logical Statements

You can combine logical operators to simultaneously see if multiple conditions have been met.

&

|

!

and

or

not

Multiple Logical Statements

Is this true? **AND** Is this true?

```
(10 < 100 & 24 == 23 + 1)
```

```
## [1] TRUE
```

```
(5 > 4 & 5 > 10)
```

```
## [1] FALSE
```

Is this true? **OR** Is this true?

```
(5 > 4 | 5 > 10)
```

```
## [1] TRUE
```

```
# Could also be something like:
# (A == B | C != D)
```

Subset your data

Sometimes, you will want to perform functions on only some of your data points.

You can **subset** your data to identify subjects in a certain subgroup.

Use the `subset()` function:

```
subset(x = empire, subset = (mass > 100))
```

```
## # A tibble: 2 x 6
##   name          height  mass sex    homeworld species
##   <chr>          <int> <dbl> <chr>  <chr>     <fct>
## 1 Darth Vader      202   136 male   Tatooine  Human
## 2 Chewbacca        228   112 male   Kashyyyk  Wookiee
```

Notes:

- Some functions have a `data =` or `x =` argument, where you name your data.frame. Because you've named in within the function's arguments, you do not need to use the dollar sign `$` to access the vector.
- `( )` around `mass > 100` are not necessary, but as logical expressions become more complicated, the extra parentheses can be helpful for keeping it straight.
- `subset()` is the name of the **function** we're using, but one of it's **arguments** is also called `subset =`

What if I don't remember the arguments?

Check the R documentation! `?subset`

When in doubt, you can always search the "Help" tab or the internet

Functions within functions

Often, you can wrap functions within functions. This is called **nesting**.

Example:

```
round(x = sqrt(86), digits = 1)
```

```
## [1] 9.3
```

How does it do this? Inner to outer!

sqrt(86) **--> 9.2736185 -->** round(x = 9.2736185, digits = 1) **--> 9.3**

A million ways to do 1 thing

Add a varible to a data.frame

```r
midus[,9] <- rep(x = "Wave2", times = 40)
midus$wave <- rep(x = "Wave2", times = 40)
midus["wave"] <- rep(x = "Wave2", times = 40)
```

Remove a variable from a data.frame

```r
midus <- midus[,-9]
midus <- subset(midus, select = -wave)
```

Removing rows from a data.frame

```r
midus <- midus[-c(1:10),]
midus <- subset(x = midus, subset = ID != "10175")
```

Exercise 2

Use nested functions to create a `subset` of the `midus` data.frame which contains only participants who have `self_esteem` scores above the `mean`.

Hints:

- Name this new data.frame something different from `midus`
- Use only complete cases (no missing) to get the `mean`. Where can you look to find the arguments of the `mean` function?

A note on missing values

- There are MANY ways to deal with missing data

- Decisions regarding missing data are best made by you and your colleagues/advisors since it will depend on your research question

- For now, we're going to use **listwise deletion** aka using complete cases

- To overwrite the `midus` data.frame so that it only shows participants that have a data point for every variable, we can use the following line of code:

    - `midus <- na.omit(midus)`

Dimensions

Sometimes it's useful to better understand the dimensions of your data.frame. Comes up in regards to error messages! *For more, details on this, see the "Examples of a few functions" section at the end of the 5: Functions practice set.*

```
dim(empire) # for both dimensions
```

```
## [1] 10  6
```

```
nrow(empire) # for rows
```

```
## [1] 10
```

```
ncol(empire) # for columns
```

```
## [1] 6
```

Saving data

So you've manipulated and cleaned your dataset, and now you want to save it...

```
write.csv(x = data.frame,
          file = "newFile.csv",
          row.names = FALSE)
```

Exploring data that is stored within a list obect (i.e., regression, ANOVA, t-test results) is really hard.

In these cases, I suggest pulling out the pieces of information that you actually care about, making it your own data.frame, and then writing out that data.frame. There are other ways, though (you'll need to Google it)!

Recoding variables

- Check out the `recode()` function from the `dplyr` package
- If there is something systematic you want to add or remove, you can use `gsub()`
  - EX: What if you wanted to replace all the underscores (_) with periods (.) in my column names?

```
colnames(midus) <- gsub(pattern = "_",
                        replacement = ".",
                        x = colnames(midus))
```

- EX: What if I wanted to remove everything after the underscore(_)?

```
colnames(midus) <- gsub(pattern = "_",
                        replacement = "",
                        x = colnames(midus))
```

Naming Conventions

Camel Case 🐫

- camelCase
- `meanAgeControls`

Pascal Case

- Similar to camelCase, but first letter is also capitalized
- `MeanAgeControls`

Dot Case

- `mean.age.controls`
- I personally don't like this because if you decide to save an object, the periods can get in the way of the file path and make it confusing (ex: `mean.age.controls.csv`)

File Names

- Underscores and hypens OK.
- DO NOT USE SPECIAL CHARACTERS, INCLUDING SPACES!!!!!

Habits

Get into good coding habits **now**, so you don't hate yourself *later*!

Pro tip:

- Write code that you can understand now
- Write code that you can understand 6 months from now
- Write code that someone else can understand 6 months from now

What does this mean?

- **COMMENT YOUR CODE**
- Use consistent naming conventions
- Use descriptive object names (who cares if they're long when you have tab-complete)
- Organize your files
- Think *computer & human readable*
- Cloud-based storage
- Version control

See next video for remaining tips within RStudio

Answers for Exercise 1

Your R script should look something like this:

```r
library(psych)
library(dplyr)
library(ggplot2)

getwd()

# set your working directory to the folder you need...if not there alrea
# setwd("folder/with/midus.csv")

midus <- read.csv("folder/midus.csv")

# change 4 character variables into factors
# see the practice for 5: Functions for more on how to do this!
midus$ID <- factor(midus$ID)
midus$sex <- factor(midus$sex)
midus$heart_self <- factor(midus$heart_self)
midus$heart_father <- factor(midus$heart_father)
```

Answers for Exercise 2

Your code should look something like this:

```
midusMini <- subset(x = midus,
                    subset = self_esteem > mean(self_esteem, na.rm = TRU
```

To look up the arguments of the `mean()` function, use `?mean`. Many functions have an argument called `na.rm` or similar to it that asks if you want the function to remove missing variables. Here, we're saying, *"yes, please remove missing variables before calculating the mean"*