# coding etiquette for (non-coder) Social Scientists

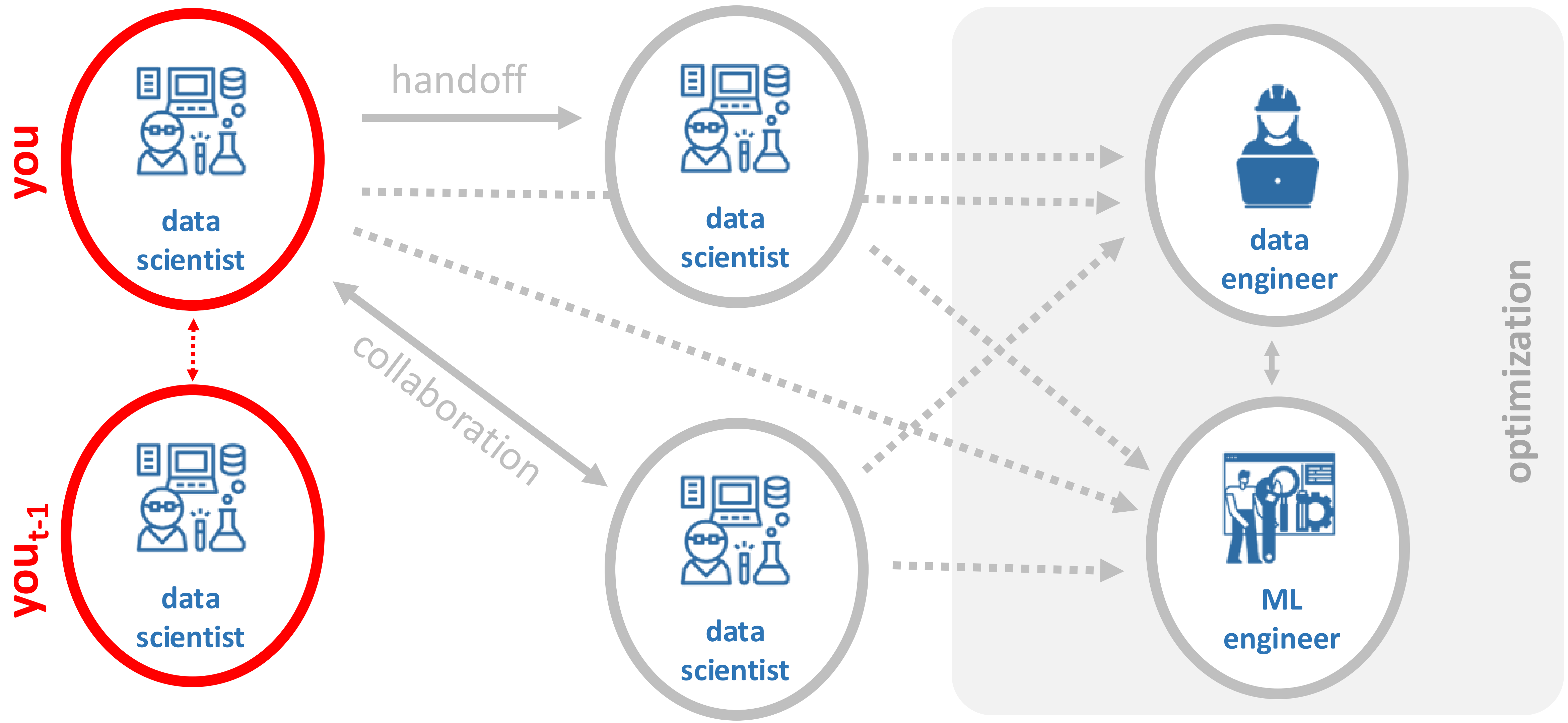
Marco Morales
marco.morales@columbia.edu

Nana Yaw Essuman
nanayawce@gmail.com

GR5069: Applied Data Science
for Social Scientists

Spring 2025
Columbia University

# **recap**: workflow collaboration in Data Science



## recap: iteration to build Data Products

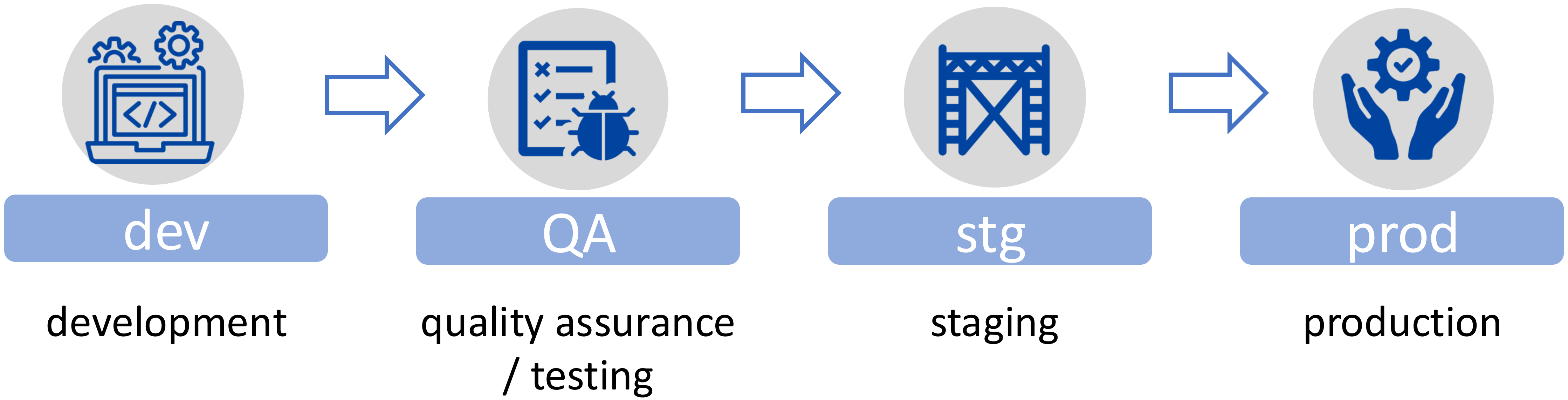
start small
(MVP)

fail fast

iterate

scale up

# **recap**: working environments



## **recap**: operational concepts in Data Science

**portability**

anyone should be able to **pick up where you left off** from any machine

**replicability**

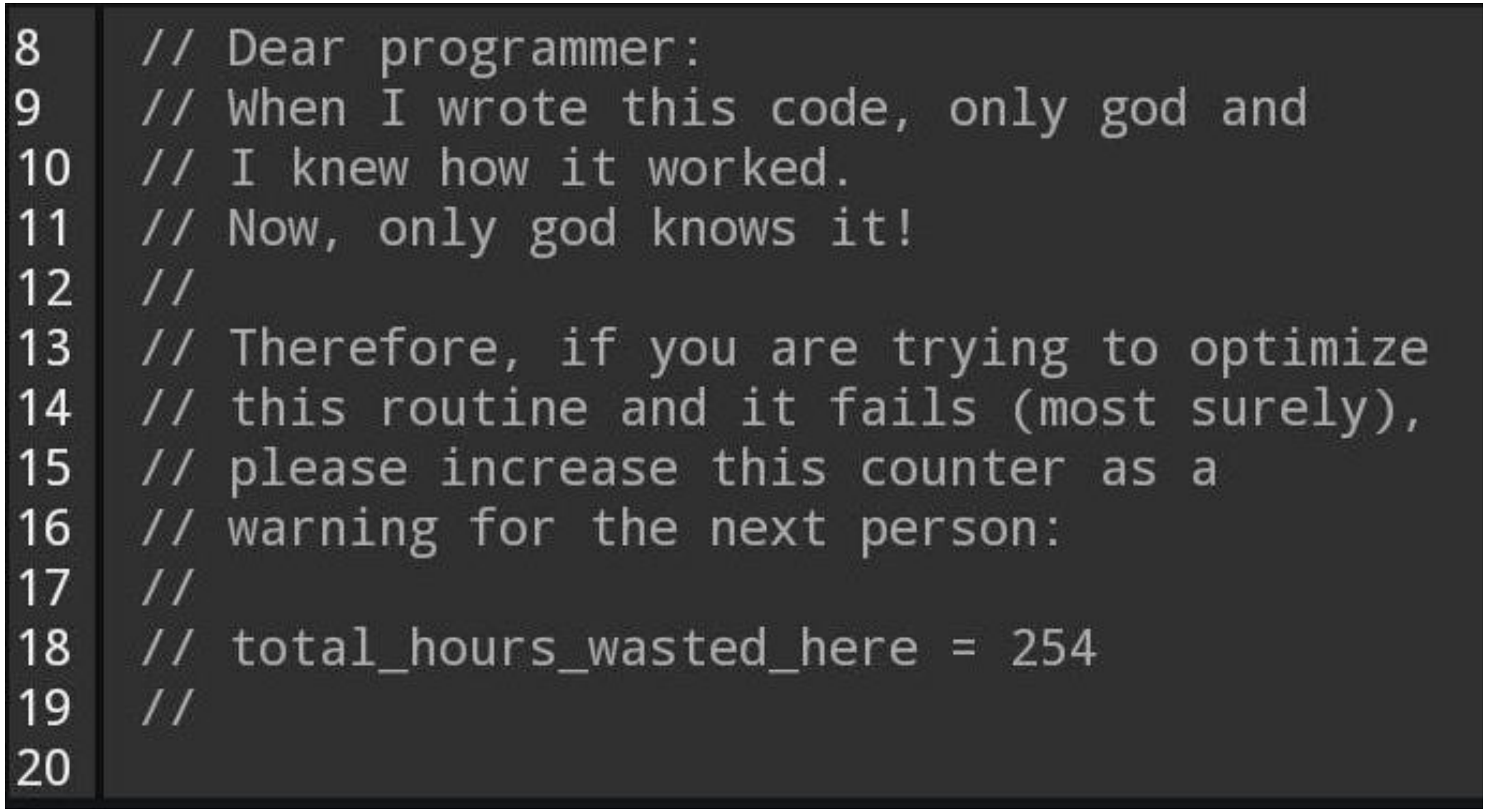
anyone should be able to arrive at your **same results**

**scalability**

your prototype should also work for **larger data sets** and/or be on the path of **automation**

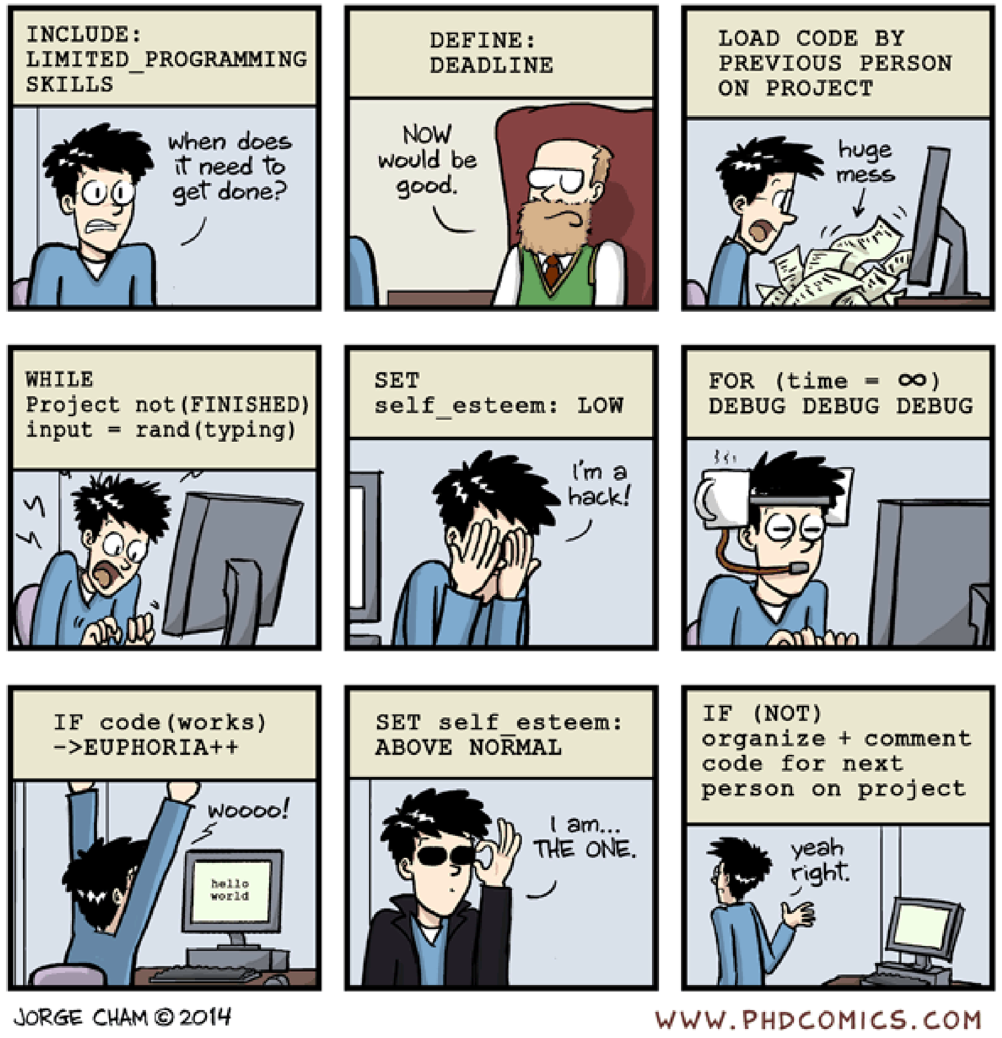
## our focus today:

## writing scalable code



```
// Dear programmer:
// When I wrote this code, only god and
// I knew how it worked.
// Now, only god knows it!
//
// Therefore, if you are trying to optimize
// this routine and it fails (most surely),
// please increase this counter as a
// warning for the next person:
//
// total_hours_wasted_here = 254
//

```

# and how to do better code handoffs

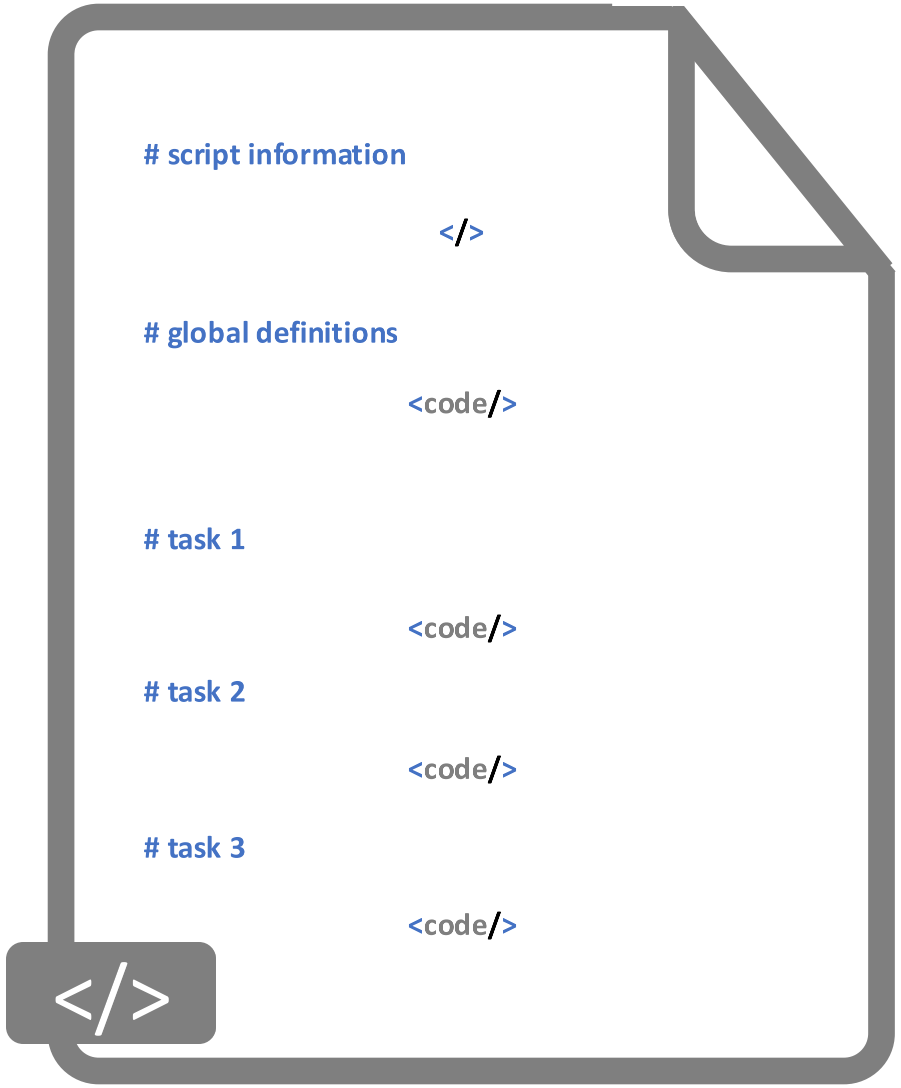

INCLUDE: LIMITED_PROGRAMMING SKILLS
when does it need to get done?
DEFINE: DEADLINE
NOW would be good.
LOAD CODE BY PREVIOUS PERSON ON PROJECT
huge mess
WHILE Project not(FINISHED) input = rand(typing)
SET self_esteem: LOW
I'm a hack!
FOR (time = ∞) DEBUG DEBUG DEBUG
IF code(works) ->EUPHORIA++
wooo!
hello world
SET self_esteem: ABOVE NORMAL
I am... THE ONE.
IF (NOT) organize + comment code for next person on project
yeah right.
JORGE CHAM © 2014
WWW.PHDCOMICS.COM

## scripts

# purpose of your (pseudo) code

- (Markdown / Jupyter) **notebooks** are great for **sharing** work and (code) review
  - nice sandbox to **develop / test** code
  - nice way to **review code + outputs** without having to run it
  - (usually) terrible for scaling!
- **scripts** are preferred for **running processes**
  - scripts can be run directly from source
  - you may need to extract your code from a notebook if you developed there
- **define the purpose of your code** early on!
  - avoid doing the same task twice!

# create structured scripts

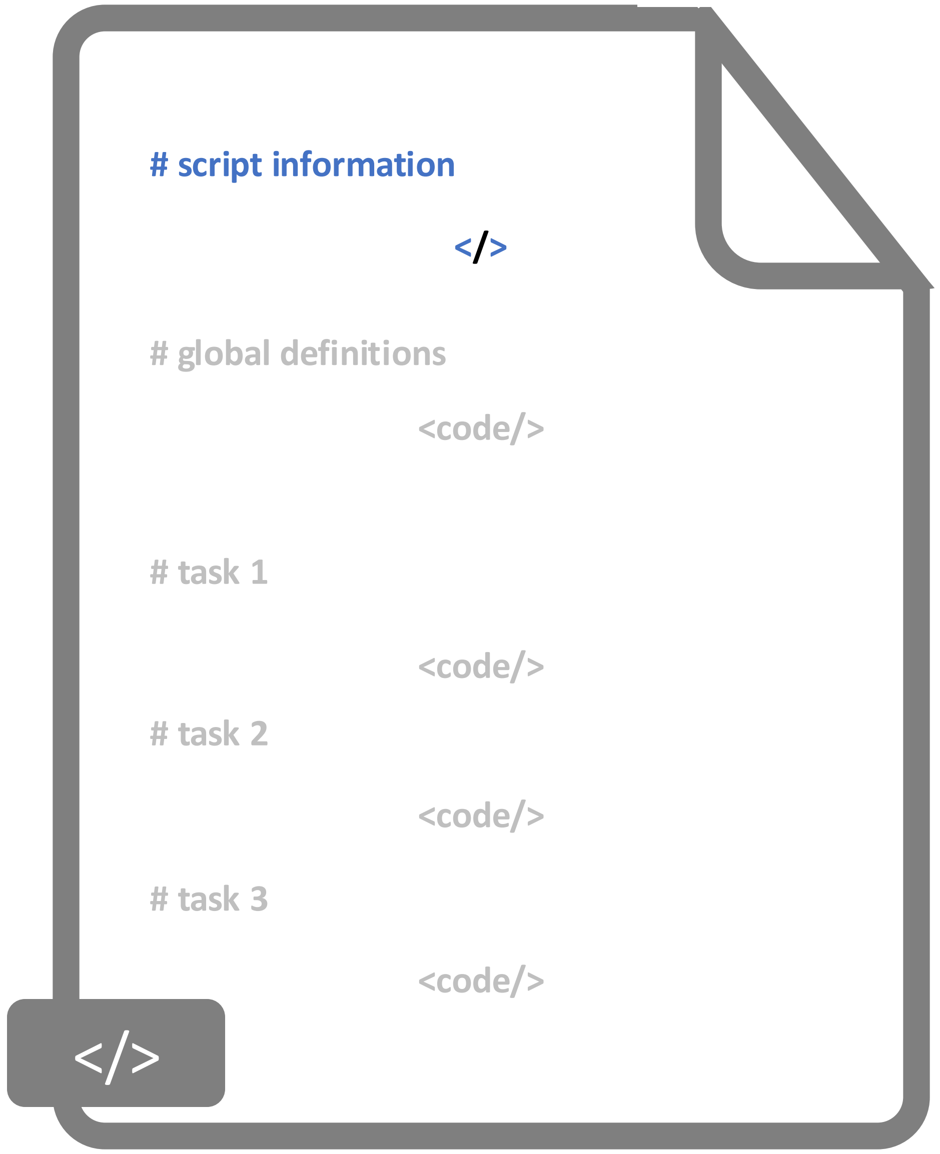


## create structured scripts

- each script should perform **only one task**
  - useful to **call additional scripts** from your script if/when needed
  - create a **global parameters** script if/when needed
  - if too many functions, create a **separate script defining all functions**
  - separate data manipulation from ML in different scripts

- your code should be **as simple as possible**
  - being clever can - and will! - come back to haunt you when sharing or revisiting code

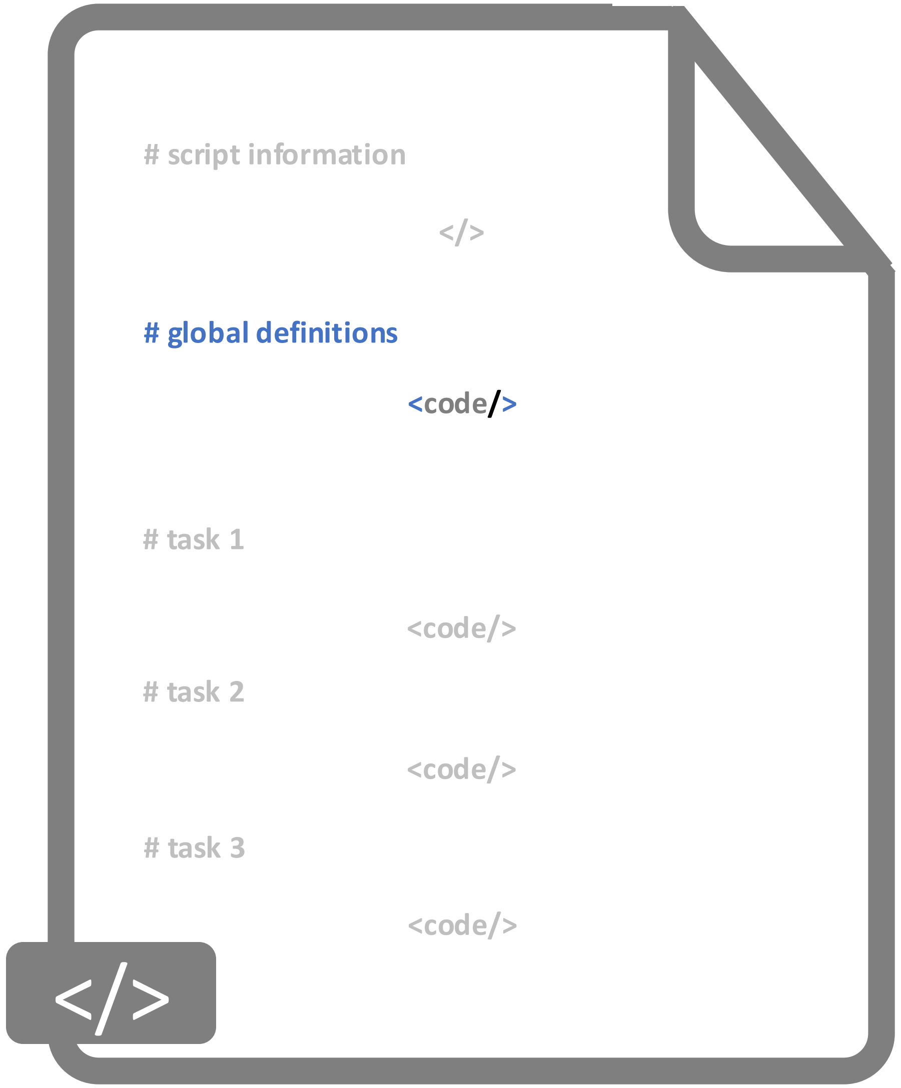
## start with a meaningful script info section



## start with a meaningful script info section

```
#   ####################################################################
#        File-Name:      MakeGraphs_CongressRollCall_160603.R
#        Version:        R 3.3.1
#        Date:           June 03, 2016
#        Author:         MM
#        Purpose:        Exploratory graphs of congressional roll call
#                        data for the 112th US Congress. Simple initial
#                        visualizations to find patterns and outliers.
#        Input Files:    ProcessedRollCall_160225.csv
#        Output Files:   Graph_RollCall_112Congress.gif
#        Data Output:    NONE
#        Previous files: MakeGraphs_CongressRollCall_160524.R
#        Dependencies:   GatherData_CongressRollCall_160222.R
#        Required by:    NONE
#        Status:         IN PROGRESS
#        Machine:        personal laptop
#   ####################################################################


library(ggplot2)
library(dplyr)
```

## add a global definitions section



## add a global definitions section

- place all **important definitions** for the project in a **single section**

```
# ::::::: SOME GLOBAL DEFINITIONS ::::::::::::::::::::::::::


# packages to load
library(tidiverse)
library(here)


# additional scripts to call
source(modeling_functions.R)


# objects to use in the script
raw_data_confrontations  <- here("data", "raw", "A-E.xlsx")
equivalence_table        <- here("data", "external", "ARCH535.csv")
```

## add a global definitions section

- load **all packages/libraries** from a single location

```
# packages to load
library(tidiverse)
library(here)
```

- call **additional scripts** from a single location

```
# additional scripts to call
source(modeling_functions.R)
```

- **always** use **relative paths** when defining locations and files

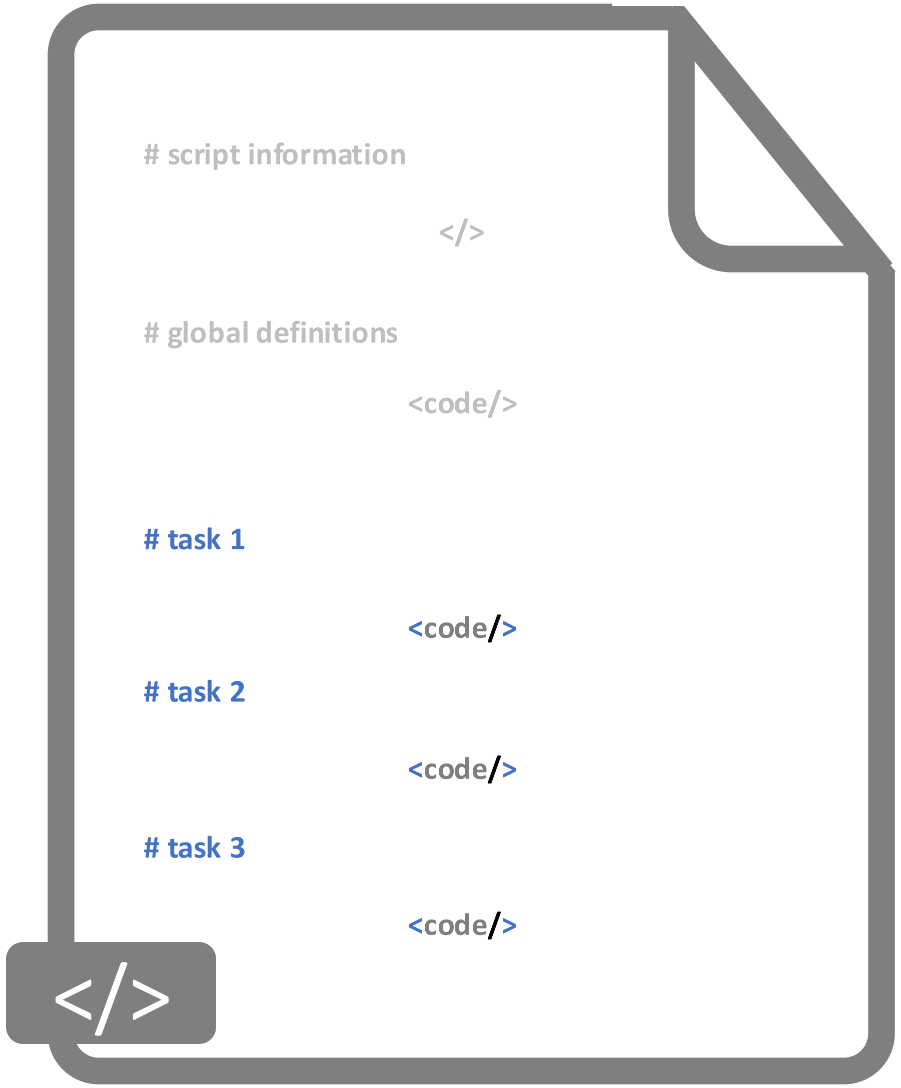
```
# objects to use in the script
raw_data_confrontations  <- here("data", "raw", "A-E.xlsx")
equivalence_table        <- here("data", "external", "ARCH535.csv")
```

## add a global definitions section

***ProTips:***

- do not add definitions manually at different places in the code!
- place at beginning of the script if using a single short script
- place on separate script if working on a larger project

## separate tasks in sections



## separate tasks in sections

- each **section** should perform a **single task**

```
# :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
# :::::::::::::::::::::: LOADS DATA :::::::::::::::::::::::::::::::::::::::::::


confrontations <- read_excel(
                             raw_data_confrontations,
                             sheet = 1,
                             na = "9999"   # converting sentinel value to null
)


# ::::::::::::::::::::: SOME DATA PROCESSING ::::::::::::::::::::::::::::::::::


forces_confrontations <- WrangleTable(forces_table_confrontations,
                                       forces_name_lookup)

forces_aggressions <- WrangleTable(forces_table_aggressions,
                                    forces_name_lookup)
```

## syntax

## generate readable code

- improve the readability of your code with **spaces,** though never before a comma

```
#Good
inner_join(forces_table, by = c("event_id" = "ID")

#Bad
inner_join(forces_table,by=c("event_id"="ID")
```

- **indent and align** your code to enhance readability

```
confrontations <- read_excel(
                             raw_data_confrontations,
                             sheet = 1,
                             na = "9999"
                             )
```

## generate readable code

***ProTip:*** **never mix spaces and tabs** to indent your code

## commenting code

## ALWAYS comment your code!!

- always start your comments with **# followed by space**
- separate your code into distinguishable chunks using visually distinct characters like : , - or =

```
# :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
# :::::::::::::::::::::: LOAD DATA ::::::::::::::::::::::::::::::::::::::::::::

raw_deaths_data <- read_csv(raw_confrontations)

# :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

## ALWAYS comment your code!!

- include **comments before each block of code** describing its **purpose**

```
# :::::: LOADING NAME CONVERSION TABLE
# the original file treats numeric codes as strings, must convert to integers
# upon loading. Also, names of municipalities are in Spanish, so must specify
# the encoding as the file is read

name_table <- read_csv(conversion_table,
                      col_types = cols(
                          CVE_ENT = col_integer(),
                          NOM_ENT = col_character(),
                          NOM_ABR = col_character(),
                          CVE_MUN = col_integer(),
                          NOM_MUN = col_character()
                          ),
                      locale = locale(encoding = "ISO-8859-1")
                      )
```

## ALWAYS comment your code!!

- **comment your functions** thoroughly, including **inputs** & **outputs**

```
MungeData <- function(baseEventData, StateNames, ForcesTable, SourceString){

    # ::::::::: DESCRIPTION
    #
    # The function performs the following transformations in the data to
    # produce the desired output data:
    #
    # 1. add actual names of states and municipalities from a Census table;
    #    currently the database only has their numeric codes
    # 2. rename columns from Spanish to English (not everyone speaks both languages)
    # 3. adding a new variable that indicates the armed force involved in the
    #    confrontation event
    # 4. replace all missing values with 0; this will come in handy as we start to
    #    explore the data futher
    #
    # ::::: INPUTS
    #
    # i)   BaseEventData - the raw database to be munged
    # ii)  StatesName - a table with State/Municipality names
    # iii) ForcesTable - a table that identifies armed forces involved in the event
    # iv)  SourceString  - a string that will identify origin of the table
    #
    #:::::: OUTPUT
    #
    # the function returns a dataframe
```

## ALWAYS comment your code!!

- include comments for any line of code **if meaning would be ambiguous** to someone other than yourself

```
# filling in NAs with zeros, to facilitate graphing and basic computations
# replace_na() requires a list of columns and rules to apply. Code below
# provides that
replace_na(
    # creates an object with numeric column names
    setNames(
        lapply(          # applies a function that links numeric column names
                         # with the asignment of 0
           vector("list", length(select_if(., is.numeric))), # creates list len= 25
           function(x) x <- 0),   # defines assignment of 0 to numeric col names
        names(select_if(., is.numeric))) # provides numeric column names
)
```

## ALWAYS comment your code!!

- ***ProTip:*** if your code needs too many comments, you probably will have to simplify it when cleaning it up

## code validation

## your code should do what you think it does

- verify that **transformed variables** resemble what you intended

```
# create a new global unique ID
processed_data %<>%
+     mutate(
+           global_id = 1:nrow(.)
+     )

# verify there are no duplicates
length(processed_data$global_id) ==
+     length(unique(processed_data$global_id))
[1] TRUE

# a quick look to see the distribution of the variable
summary(processed_data$global_id)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      1    1350    2698    2698    4047    5396
```

## your code should do what you think it does

- verify that **missing data is handled correctly** on any recode or creation of a new variable

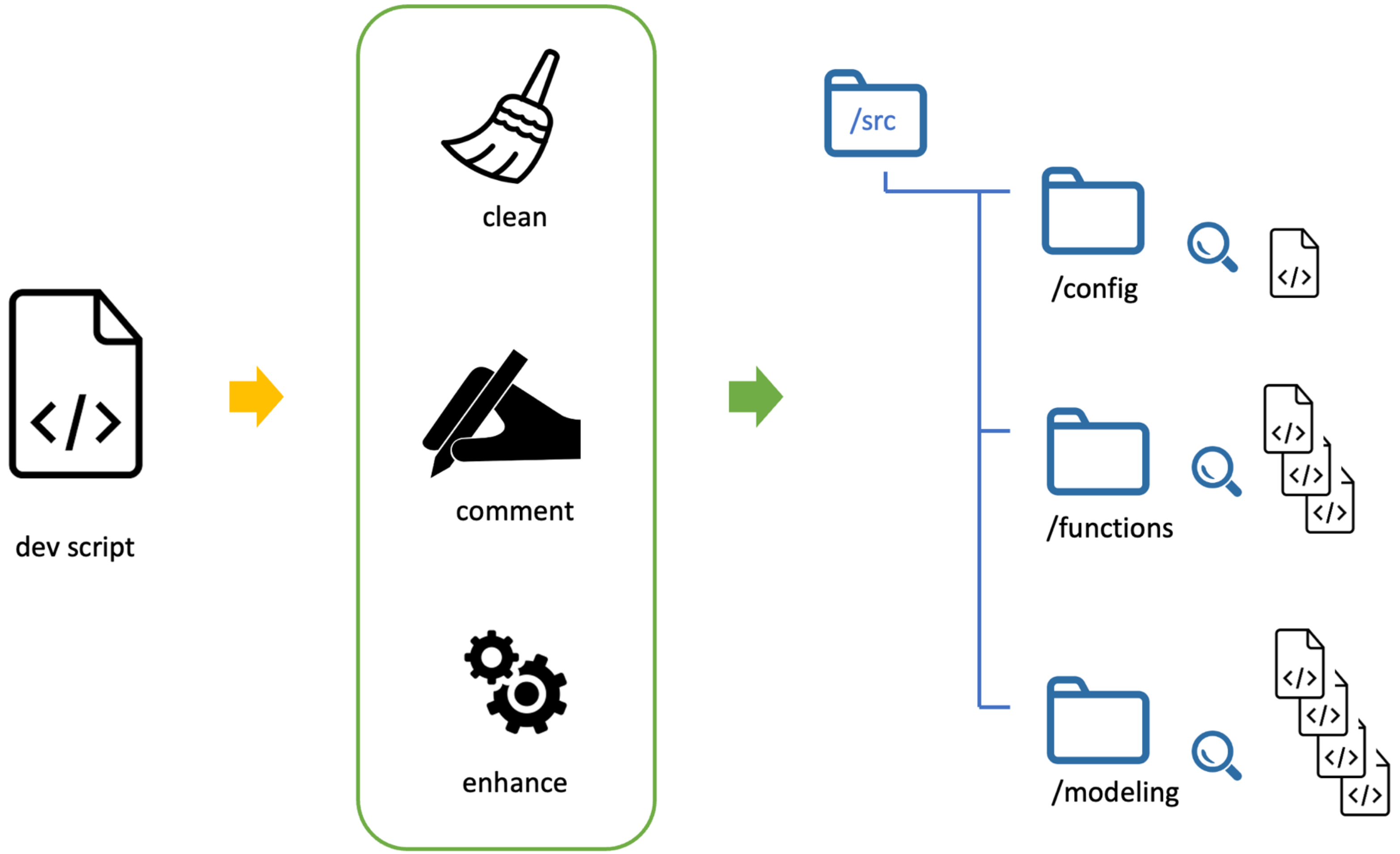
```
# computes lethality indices
processed_data %<>%
+   mutate(organized_crime_lethality =
+               organized_crime_dead /
+               organized_crime_wounded
+   )

# exploration to identify undefined values
summary(processed_data$organized_crime_lethality)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
      0       1     Inf     Inf     Inf     Inf    3090
```

## workflow principles

## general workflow principles

- **80 characters** should be the maximum length of any line in your code
- if you find an error in your code, **correct it exactly where it happened**
  - NEVER fix it from a later chunk of code
- when you are done with your project, go back and:
  - **clean up** your code
  - **add comments** where appropriate (for the you of the future)
  - perform **stress tests** with as many **edge cases** as you can imagine
  - make sure to **document future enhancements** (especially to scale up)

# general workflow principles



## commit messages in git

## commit with informative messages

- **remember**: commit **small chunks of logically grouped changes**
  - you may want to undo a change, but only that change
- message summarizes **what changed**
  - use imperative mood
    - [*this commit will*] Rename income variable
  - start with a **capital letter** and **do not end with a period**
  - maximum length: **50 characters**
- if you need to provide more detail on the **what** and **why**:
  - add a **body** by adding a **blank line**
  - add a **paragraph** that wraps text at **72 characters**

# coding etiquette for (non-coder) Social Scientists

Marco Morales
marco.morales@columbia.edu

Nana Yaw Essuman
nanayawce@gmail.com

GR5069: Applied Data Science
for Social Scientists

Spring 2025
Columbia University