THE IOP POLICY PROGRAM

Resource and Style Guide

Chairs:
Alex Grayson
Amisha Kambath

Membership Director & Primary Contributor:
Amy Zhou

Summer 2020

TABLE OF CONTENTS

RESEARCHING AND WRITING TIPS

Don't know where to start?

Many of our researchers have varied research workflows, but if you're not sure how to approach an assignment, here is a sample research pathway that you could take.

STEP 1: Do a broad, Google search of your research question.This will generally pull up a lot of news articles or articles from non-academic sources. These usually are easier to read and can help you to narrow down what you want to focus on or identify any discrepancies that may indicate a deeper issue at play.

STEP 2: Look at academic research papers/do a more in depth examination of the issue. Given what you found in your initial search, now you can type your narrowed topic or question into Hollis or Google Scholar to find more formal sources on the topic.

STEP 3: For each source write down a few main points.

STEP 4: When you're writing up your research you can synthesize the main points of these sources into conclusions and policy recommendations. At this point if you notice gaps in the research or discrepancies, you can go back and search for sources to address these.

LESS IS MORE!

No one wants to read a hundred paged policy paper, and most of our commisioners won't have the time either. So be concise!

Other tips:

- Keep up with your edits! Don't leave them all to the end of the semester (it will not be fun).
- Communicate to your teammates so you don't turn in redundant research
- Keep up with citations!

STYLE GUIDE

Abbreviations and Contractions

- Only use abbreviations when they improve readability; if a term is used only once, you do not need to introduce its abbreviation. Otherwise, abbreviations and acronymns should be introduced the first time that they are used.
  For Example: According to the Legal Services Corporation (LSC)...
- Avoid contractions unless in quotations

Numbers

- Numbers 1 - 10 should be written out
- Money should be written as $4.5 Billion and not $4.5 Billion dollars
- Always use numerals for age
- Always spell out percent
- Spell out a number if it is the first word in a sentence

Dates

- Be as specific as possible with dates
- Format is Month, Day, Year : August 4, 2020

Sensitivity

- Use gender neutral professional titles when possible ie. ‘spokesperson’ not ‘spokeswoman’
- When possible refer to a person’s country of origin rather than broad sweeping ‘Asian-American’ or Latino/Latina: ie. ‘Cuban’, ‘Korean American’
- ‘African-American’ is acceptable for description of an American Black person of African descent, as is ‘Black,’ but the terms are not necessarily interchangeable.
- When referring to drug use, avoid terms that treat it as an identity such as ‘junkie,’ ‘addict’, etc.
- Avoid referring to incarcerated individuals as ‘criminals.’

CITATIONS

The IOP Policy Program uses Chicago Style citations. This means that you will have to generate a few different types of citations to be used in a various situations. ***It is extremely important that you use the correct one in the right place.***

Footnotes

Footnotes go at the bottom of each page of the report and correlate to every claim cited by the paper. There are three types of footnotes that you will need.

FULL NOTE: Use full notes the first time that a source is referenced. These contain mostly the same information as bibliography entries but are formatted in a different way.

Format: Henry David Thoreau, "Walking," in *The Making of the American Essay*, ed. John D'Agata (Minneapolis: Graywolf Press, 2016), 177–78.

SHORTENED NOTE: Use in subsequent, non-consecutive uses of a source.

Format: Thoreau, "Walking," 182.

IBID: Although not formally a "type" of footnote, it means "in the same source," so you will use Ibid. for subsequent, consecutive uses of a source.

Bibliography Entries

At the very end of the report, there will be a bibliography containing all of the sources referenced in the report.

Format: Thoreau, Henry David. "Walking." *In The Making of the American Essay*, edited by John D'Agata, 167–95. Minneapolis: Graywolf Press, 2016.

HELPFUL WEBSITES

Indiana Wesleyan University Chicago Style Guide

Purdue Owl Chicago Style Guide

CITATION TOOLS

Citation Machine

Zotero

IMPORTANT NOTES AND TIPS

- **You must include a footnote EVERY time you make a claim. This means that most of your sentences will have a footnote attached to it.**
- **However, if you are concluding something from synthesizing various source information, you do not need to include a footnote.**
- **Cite as you go! Create properly formatted footnotes and a bibliography with every assignment you turn in. It's tempting to procrastinate and put this off, but DON'T DO IT!! Keeping up with your citations will make you and your chairs' lives so much easier when the end of the semester comes around and you don't have to go through citing tons and tons of sources (it's terrible, and sometimes you have to spend hours tracking down a source because you forgot what it was called after weeks of not using it).**

COMMON ERROR

One of the most common errors that people make is using the wrong citation type, so make sure you use the correct footnote type.

Furthermore, full notes and bibliography entries look similar and contain similar information but they ARE NOT the same.

The easiest way to tell the difference is by looking at the seperators and author names.

- **Full notes use commas to separate each piece of information and bibliography entries use periods.**
- **In a full note, the author name is listed as First M. Last; in a bibliography entry the author name is listed as Last, First M.**

USING ZOTERO

Many people find Zotero to be an extremely helpful tool when doing citations and also organizing sources. It can take a bit to figure out its quirks, but once you have a handle on the tool it is invaluable, and will make your life a thousand times easier.

Set Up

The Harvard Libraries have a great guide to setting up Zotero both for Word and for Google Docs. They also have a great guide on how to use Zotero.

Collecting and Citing

For policy paper purposes the main steps to using Zotero are:

STEP 1: Save articles, websites and files to your library using the Zotero Google Chome extension.

STEP 2: To add a citation for one of the items, go to the Zotero tab in your word processor. Click add/edit citation. For your first note choose Chicago Full Note (under footnotes). For shortened note, click Chicago Note (usually Zotero will put shortened notes where appropriate automatically).

STEP 3: At the end of your document, you can add your bibliography by clicking Add/Edit Bibliography. Select Chicago Full Note for the style.

OR

STEP 1: You can cite directly from the Zotero app to create citations. First, go into the folder, right click on an entry and choose "Create Bibliography from item."

STEP 2: For footnotes choose your Output Mode as 'Notes'. For the first footnote of a source, choose Chicago Full Note as the style and for later uses, choose Chicago note.

STEP 3: Once you want to create your bibliography, choose your output mode as 'Bibliography.' Choose Copy to Clipboard as output method.

* Click the icon for the Zotero quick start guide

NOTE:
Zotero will only let you cite sources that you've used so usually to add additional sources and make it easier for your chairs, please unlink your document from Zotero when you're done citing!

WORKING WITH DATA

SET UP

Install R:
- Windows
- Mac

Install RStudio:
- Windows
- Mac

OPTIONAL

Github Sign-Up
- Link

Install Git
- Windows
- Mac (Follow Instructions Below)
  - Step 1: Open your terminal (Spotlight Search OR Find in utilities folder in Applications)
  - Step 2: Type: git --version
  - Step 3: If you already have it, you are set. If it asks you to install, click on install

CREATING NEW PROJECTS

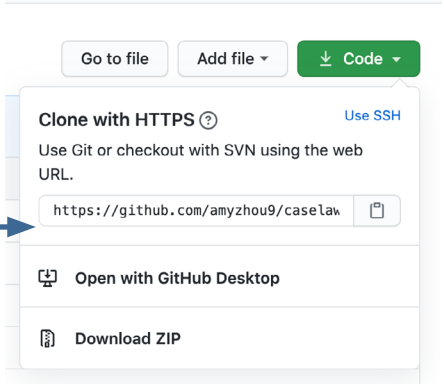
R Projects are folders that will hold all of your raw data, your R files, any graphics that you create and an .RProj file that designates this folder as the working directory. Clicking on the .RProj file will open that project in R.

CONNECTED TO GITHUB

If you connect your projects and files to your GitHub, your code and data will be backed up online. This is useful as it can be easily shared with collaborators, but if your data is given confidentially to you from your commissioners, you should make your repository private or store it locally.

STEP 1: Go to Github.com. Create a new repository on your Github account and name it what you want your project to be named. It's best practice to initialize the repository with a README.

STEP 2: Go to your repository and click the green code button. Copy the URL.



STEP 3: In RStudio go to the File tab. Click New Project --> Version Control --> Git then paste the repository URL into the repository URL slot. You can choose where this project saves on your computer by changing the subdirectory option.

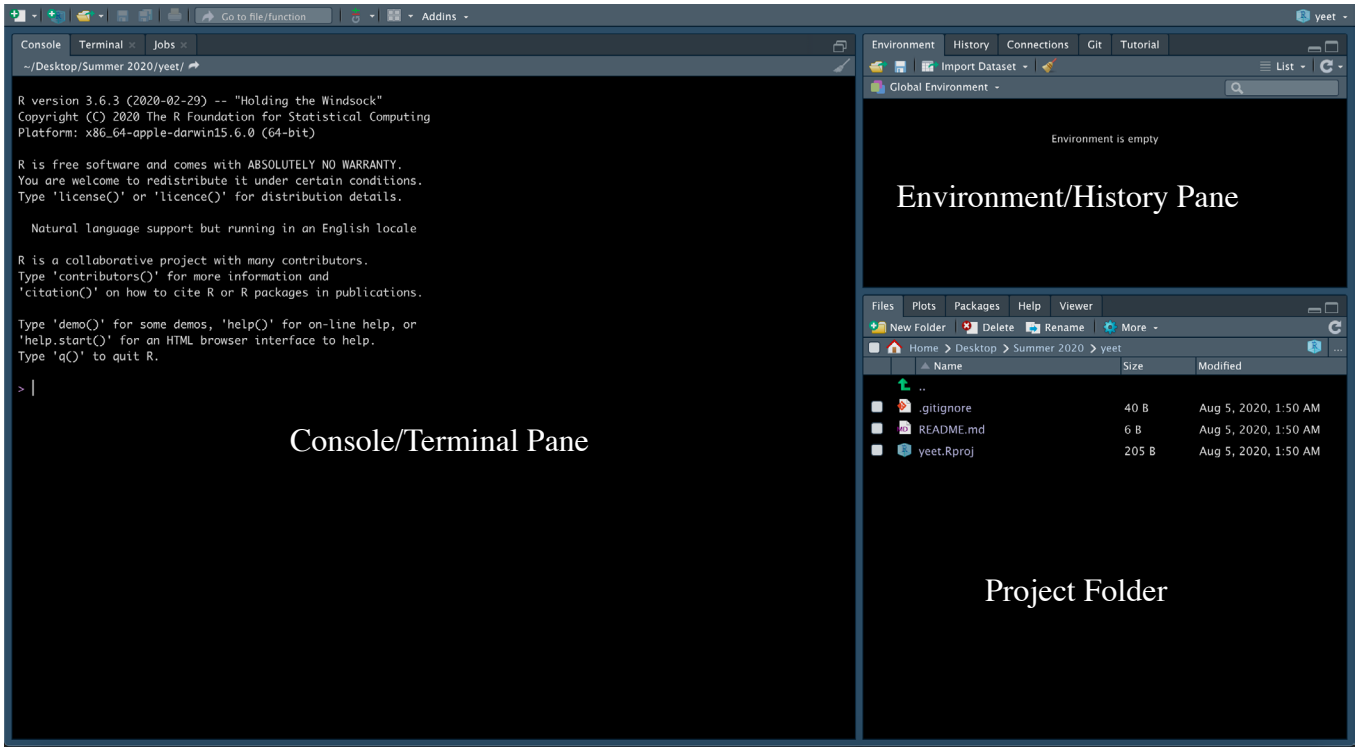
Your new project's Environment/History pane should have a tab labeled "Git."

LOCALLY

If you don't have a Github account or don't intend on sharing your code publicly, creating your project locally is a good option.

STEP 1: In RStudio go to the File tab. Click New Project --> New Directory --> New Project. Give the new project a name and choose where to save the project by changing the subdirectory option.

Congrats! You have created a new project and you should see something like this:



What do these do?

CONSOLE/TERMINAL PANE: You can enter R commands directly into the console and this is where you communicate to RStudio. In fact without creating any files you can code everything in the console (this is annoying and not recommended). The terminal is where you communicate to your computer.

ENVIRONMENT/HISTORY PANE: Shows what objects you have created and stored in your environment during your current R session; makes accessing these objects much faster and easier. This is also where you will push and pull changes to Github (if you created the project through Github, there will be a Git tab).

PROJECT FOLDER: Shows all of the files in your R project

CREATING NEW FILES

Once you have created an R Project, you can create new R files within the project.

There are two types of files that you will probably be coding in for the Policy Program: **R Markdown files (.rmd)** and **R Script files (.R)**. R Markdowns can be knitted into a nicely formatted document that can be published. R Script files are just pure code.

To create either one go to the File in RStudio and go to New File and select the kind of file that you want to create. Make sure that you are creating the file in the correct project/directory that you want.

When you create an R Markdown, there will be a lot of filler code so you can delete everything until only these sections are left: the header chunk and the setup chunk. Your file should then look something like this:

```
---
title: "oldestPeople"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
```

BACKING UP FILES TO GITHUB

If your project is connected to your Github, you should regularly back up your code to your repository.

STEP 1: Go to the Git tab in your Environment/History Tab

STEP 2: If you have made edits to your project, you should see the files that you have edited in the pane.

STEP 3: Check the files that you want to back up, and click commit. Commit is essentially saying that you are approving the changes that you made to these files and want to save these changes to the cloud. It is good practice to give commits names so that in the future you know what changes you made with each commit.

STEP 4: Once you have these changes committed, you can click the green upward arrow. This is called pushing your changes to Github. If you go to your repository on Github, you should be able to see the changes that you pushed.

DATA ANALYSIS IN R

There are various ways to do analysis and code in R, so if you've taken coursework in R you may have learned how to use R in a different way. This is a brief guide on how to do analysis and visualization using the Tidyverse. If you want to learn it more in depth, Gov 50 is a great course to do that; you will be guaranteed to have a super in depth understanding of how to work with data in R.

PACKAGES

There are thousands of R packages available and are the key to using R. Each package is a set of functions geared towards a common goal or theme. For example, the memer package allows you to make memes and the janitor package is used for cleaning datasets.

Common packages that you'll need are:

- **tidyverse**
- **dplyr**
- **janitor**
- **readxl**
- **writexl**
- **readr**
- **gplot2**
- **gt**
- **broom**
- **lubridate**
- **stringr**

If you want to do something with your data that you're not sure how to, for example make a wordcloud or do text mining, there's probably a package for it, so Google it!

How to use packages?

First you will need to install the package into R Studio, and you only need to do this once. In your console tab in your Console/Terminal pane (bottom left), use the install.packages() function. For example to install the tidyverse package I would write:

install.packages("tidyverse")

*Note: I put the name of the package in quotation marks, as without these, R will not recognize this as a package name.

Then, every time to create a new R Script or R Markdown, you will need to call the packages that you will need in that file. Usually, I will put these calls in the setup chunk. To call the library/load the library into your file type:

library(tidyverse)

*Note: This time, the package name does not need to be in quotation marks.

```
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
library(tidyverse)
library(ggplot2)
library(janitor)
```
```

CODING IN THE TIDYVERSE: BASICS

<- use a left arrow to assign code/data on the right of the arrow into a new object/dataframe given by the name of the left of the arrow.
For example: x <- code... creates an object called x containing all of the code on the right.

%>% this is a pipe operator which is used to connect functions. It tells the function on the right of the operator to act upon the data piped to it from the left of the pipe operator.

Cmd + Shift + M OR Ctrl + Shift + M

Code is organized by code chunks which are just chunks of related code. To create a new code chunk:

Cmd + Option + I OR Ctrl + Alt + I

You can run code by running the whole document, by chunk, or by line.
To run the whole document click the Run button at the top right of the document and select Run All. To run a code chunk, click the green arrow at the top right of the chunk. To run a line, go to that line and use **Cmd + Enter OR Ctrl + Enter.**

READING IN DATA

To read in data there are various options. Generally, when you get data you should try to download them one of the following forms:

- Excel, CSV, .Rdata, .Rds, Stata

Many dataforms can be read in using just the base package, but for other forms such as Excel you will need to install and load other packages. Below are a selection of the basic functions you can use to read in datasets in these formats.

Excel (.xlsx)
library(readxl)
new_object_name <-
read_excel("filepath")

.csv
No packages needed
new_object_name <-
read.csv("filepath")

.Rdata
No packages needed
load("filepath")

.Rds
No packages needed
new_object_name <-
readRDS("filepath")

Stata (.dta)
library(foreign)
new_object_name <-
read.dta("filepath")

VIEWING AND CLEANING DATA

Now that you have your data loaded into your environment, you can clean it up a bit. Best practice wise, it is smart to take a look at your data to see if there's any weird rows or columns that you want to get rid of or fix. For example, many excel sheets with have weird header rows or total rows which are difficult to work with in R.

There are a few easy ways to view your data:

- Simply typing the name of the object you created will show you the entire dataframe
- summary(name of object) will give you a lot of information about the mean, median and quartiles of each of the variable
- glimpse(name of object) will flip your data so you can easily see all of the variables at once

If you have to get rid of rows, you can either skip them when you read the data in (this depends on the function you use to read the data-- look to see if there's a skip argument in the documentation)

IMPORTANT, SUPER COOL, RSTUDIO HACK

If you're not sure how to use a function, typing ?name_of_function {??name_of_function if the package the function is in has not been loaded) in the console will pull up the documentation for the function including all of its arguments.

For example, ?read_excel will pull up the documentation for the read_excel function, showing us that there is a skip function that we could use to skip line when reading in data. The na = " " argument can be used to assign certain values to be coded as NA

```
read_excel(path, sheet = NULL, range = NULL, col_names = TRUE,
  col_types = NULL, na = "", trim_ws = TRUE, skip = 0,
  n_max = Inf, guess_max = min(1000, n_max),
  progress = readxl_progress(), .name_repair = "unique")
```

Other useful functions for selecting certain regions of data are:
head(n) which will take the first n rows of the data, **tail(n)** which takes the last n rows of the data, and **slice(n:q)** which takes data from the nth row to the qth row.

The janitor package is also useful for cleaning data. In particular the **clean_names()** function can be used to standardize variable names into a certain case ie. snake, camel, etc.
If analysis can be more efficiently done with a pivoted table, you can pivot the date with **pivot_longer()** and **pivot_wider()**

ANALYZING DATA

This guide cannot even attempt to cover all of the functions that R has for analyzing and modelling data. Thus it will provide just a few basic functions that you can use to conduct basic analysis with. If you want to conduct more advanced analysis, Google is your friend!

Sorting and Filtering

filter(conditional_statement)
Selects rows that make the conditional statement true.

arrange(name_of_variable)
library(dplyr)
Sorts the data by the given variable, often can be paired ith head(), tail(), slice()

Basic Data Manipulation

mutate(name_of_new_variable = new column entries)
library(dplyr)
Create new variable in your data

group_by(variable)
library(dplyr)
All following code will be run once for each group within the group variable(s)

summarize(function(variable))
library(dplyr)
Often used with group_by, will collapse all the given rows in a variable according to the function ie. mean, sum, etc

select(variables)
Select variables to keep
subset()
subset rows and columns

map(vector, function)
library(purrr)
Iterates over every element in a vector and conducts the function on them

ifelse(), case_when()
Conditional functions
ifelse(condtional, action, action)
case_when(condition ~ action)

Saving Data Into Other Forms

write.csv(object, "filepath")
Saves your dataframe object into a csv

save(object, "filepath.rdata")
Saves your dataframe object into an .Rdata file

saveRDS(object, "filepath")
Saves your dataframe object into an .rds file

write_xlsx(object, "filepath.xlsx")
Saves your dataframe object into an Excel file.

Statistics and Regressions

t_test()
library(infer)
Conducts a two sample t-test

prop.test()
library(stats)
Conducts a two proportion z-test

lm(y ~ x, data)
library(tidypredict)
Returns a linear regression model between two variables

For multiple regressions:
lm(y ~ x + x2 + x3 ..., data)

tidy(model, conf.int = TRUE)
library(broom)
Can tidy up your regression models. Conf.int give you confidence intervals.

corr()
library(boot)
Find correlation coefficient between two variables

Other helpful functions:

grepl() - does regular expressions to match patterns, useful for filtering

sample() & rep_sample_n() - randomly samples for a vector of values/objects

full_join(), inner_join(), left_join(), right_join() - joins separate data sets together on a common column/variable. Full join keeps everything, inner join keeps things that both datasets have, left join keeps everything on the left hand side and right join keeps everything on the right hand side.

bind_cols(), bind_rows() - bind the columns or rows of two datasets together

Additional Resources:

Google!! - There are so many really great tutorials on cool things that R can do. For example here is one on textmining. Many others are only a Google search a way!

Gov50 Textbook - Really great, comprehensive textbook that is constantly being improved by Gov 50 course staff.

RStudio Community - If you make an RStudio community account, the RStudio community is often a great resource to turn to for some crowd sourced help.

When you ask for help, make sure you make a reprex of your code so other can recreate it. You can do so with the reprex() function and package.

Congrats on your acceptance into the Policy Program!!

Happy, researching, writing and analyzing! Go out and make change!