Spring 2023

Gregory Bruich, Ph.D.
Dept. of Economics, Harvard University

Lab 8: Bias in Algorithms

Methods/concepts: algorithmic bias; choice of "labels" vs. "predictors"

LAB DESCRIPTION

In this lab, we will dive deeper into bias in algorithms, following Obermeyer, Powers, Vogeli, and Mullainathan (2019). We will train several prediction algorithms, some including the patient's race and others explicitly leaving out the patient's race. We will see how the choice of "label" – either patient costs or patient health – affects the performance of the models. Finally, we will examine the racial composition of patients predicted to have high risk according to the algorithms. A list and description of each of the Stata and R commands needed for questions 6 through 9 on this lab are contained in Table 2 and Table 3, respectively.

QUESTIONS

1. Start by randomly splitting the 48,748 patients included in the **health.dta** data set into a **10%** training data set and a **90%** test data set. Table 1 describes the data. There are two reasons we are using such a small fraction of the data to train the models. First, estimating random forests on a larger fraction of the data would be prohibitively time consuming. Second, we require a large number of observations in the test data set so that we can study differences in risk score by race.

2. Estimate the following statistical models using the training data set:

    a. Random forest to predict the "label" of patient costs (`cost_t`) using the full set of predictors consisting of all variables starting with `tm1_`, but *excluding* the patient's race

    b. Random forest to predict the "label" of patient costs (`cost_t`) using the full predictor set, now *including* the patient's race

    c. Random forest to predict the "label" of patient health (`gagne_sum_t`) using the full predictor set, *excluding* the patient's race

    d. Random forest to predict the "label" of patient health (`gagne_sum_t`) using the full predictor set, *including* the patient's race

    Note that random forests with lots of observations and predictors (150) will take a long time to run. You should therefore only use around 100 trees in your forests.

3. Calculate and compare the root mean squared prediction error for your models that include patient race vs. those that exclude patient race in the **training sample**.

4. Calculate and compare the root mean squared prediction error for your models that include patient race vs. those that exclude patient race in the **test sample**.

5. Export a data set with **the test data** and your predictions as a .dta file. If you are in a Stata lab, you can exit Python and load this file into Stata for further analysis.

6. As in Lab 1 and Lab 2, convert the predictions in the test sample from each of your prediction algorithms into percentile ranks, normalized so that the top rank is equal to 100. The percentile rank is the "risk score" from the algorithm.

7. Now consider a program that makes patients eligible for extra resources if their "risk score" is above the 55th percentile. (This corresponds to the top 45 percent of risk scores).

    a. As on lab 1 and 2, start by defining four new indicator variables corresponding to whether the risk score from each model is strictly greater than 55.

    b. What fraction of all Black patients would be eligible for the program using each of the four algorithms? To answer this question, report the means of the indicator variables you created in (a) after subsetting the data to Black patients (i.e., `tm1_dem_black == 1`).

    c. Among patients eligible for the program, what fraction are Black? To answer this question, report the mean of the indicator variable `tm1_dem_black` after subseting the data to patients eligible for the program for model 1. Then repeat for models 2, 3, and 4.

8. Now we will replicate the key figures from Obermeyer, Powers, Vogeli, and Mullainathan (2019). Produce binned scatter plots of patient costs and patient health vs. the percentile rank "risk score" from each algorithm, with White and Black patients plotted separately. This is a total of 8 graphs: 4 models x 2 outcomes.

    In Stata, use a connected line type in `binscatter`, which is controlled by the option `linetype(connect)`. To plot Black and White patients separately, use the `by(race)` option:

    ```
    binscatter outcome_variable percentile_rank, by(race) linetype(connect)
    ```

    In R, you could do the same in `ggplot` by using the `geom="line"` option and `geom="point"` option in `stat_binmean` from the `statar` package, and set the `color` option to the race variable to plot Black and White patients separately:

    ```
    ggplot(dat, aes(x = percentile_rank , y = outcome_variable, color = race)) +
     stat_binmean(n = 20, geom = "line") +
     stat_binmean(n = 20, geom = "point")
    ```

9. In the pre-recorded video for this lab, Professor Ziad Obermeyer said that it is the left-hand side variable (i.e., the "label" or target parameter) that is the source of bias in algorithms, not the right-hand side variables (i.e., the predictors). Explain what he meant, and evaluate whether you agree with him using your binned scatters above.

10. The files to submit for this lab are:

    a. Your well annotated .do/.pynb /.R/.rmd file(s) replicating all your analyses above (with enough comments that a principal investigator on a research project would be able to follow and understand what each step of the code is doing). Please submit these files to Gradescope.

    b. For the Stata/Python labs, please submit a log-file with the log showing the output generated by your final do-file for questions 6-8. Please submit this file to the same gradescope assignment as the do-file. (Please do not submit a .smcl file: we can only read .log files in gradescope).

    c. A PDF version of the solutions to the above questions. For graphs, save them as .png files and insert them into the document. Please submit this file to the same gradescope assignment as the code. (Please do not submit a word document: we can only read PDFs in gradescope. Using

R Markdown is never required; but if you have chosen to use it, you can *knit* the file to generate the PDF).

DATA DESCRIPTION, FILE: health.dta

The data consist of 48,784 patient records. Variables that start with `tm1_` were measured in the prior year (time $t-1$). Variable that end with `_t` are measured in the current year. For more details on the construction of the variables included in this data set, please see Obermeyer, Powers, Vogeli, and Mullainathan (2019).

TABLE 1
Variable Definitions

Variable (1)	Description (2)	mean (3)	sd (4)	min (5)	max (6)
patient_id	Patient identification number	n/a	n/a	n/a	n/a
gagne_sum_t	Total number of active chronic illnesses	1.354	1.942	0	17
cost_t	Total medical expenditures, rounded to the nearest 100	7,660	17,990	0	550,500
cost_avoidable_t	Total avoidable (emergency + inpatient) medical expenditures, rounded to nearest	2,435	12,058	0	642,700
race	String variable containing the words "black" and "white"	n/a	n/a	n/a	n/a
tm1_dem_black	1 = Black 0 = White	0.114	0.318	0	1
tm1_dem_female	1 = Female 0 = Male	0.631	0.483	0	1
tm1_dem_age_band_1824	Indicator for patient age between 18-24	0.0369	0.188	0	1
tm1_dem_age_band_2534	Indicator for patient age between 25-34	0.110	0.313	0	1
tm1_dem_age_band_3544	Indicator for patient age between 35-44	0.194	0.396	0	1
tm1_dem_age_band_4554	Indicator for patient age between 45-54	0.239	0.427	0	1
tm1_dem_age_band_5564	Indicator for patient age between 55-64	0.197	0.397	0	1
tm1_dem_age_band_6574	Indicator for patient age between 65-74	0.142	0.349	0	1
tm1_dem_age_band_75	Indicator for patient age 75+	0.0703	0.256	0	1
tm1_alcohol_elixhauser	Indicator for alcohol abuse	0.00892	0.0940	0	1
tm1_anemia_elixhauser	Indicator for deficiency anemia	0.0636	0.244	0	1
tm1_arrhythmia_elixhauser	Indicator for arrhythmia	0.0922	0.289	0	1
tm1_arthritis_elixhauser	Indicator for arthritis	0.0466	0.211	0	1

tm1_bloodlossanemia_elixhauser	Indicator for blood loss anemia	0.00246	0.0495	0	1
tm1_coagulopathy_elixhauser	Indicator for coagulopathy	0.0115	0.107	0	1
tm1_compdiabetes_elixhauser	Indicator for diabetes, complicated	0.0217	0.146	0	1
tm1_depression_elixhauser	Indicator for depression	0.0621	0.241	0	1
tm1_drugabuse_elixhauser	Indicator for drug abuse	0.00623	0.0787	0	1
tm1_electrolytes_elixhauser	Indicator for electrolyte disorder	0.0329	0.178	0	1
tm1_hypertension_elixhauser	Indicator for hypertension	0.332	0.471	0	1
tm1_hypothyroid_elixhauser	Indicator for hypothyroid	0.0938	0.292	0	1
tm1_liver_elixhauser	Indicator for liver disease	0.0159	0.125	0	1
tm1_neurodegen_elixhauser	Indicator for neurodegenerative disease	0.0280	0.165	0	1
tm1_obesity_elixhauser	Indicator for obesity	0.0929	0.290	0	1
tm1_paralysis_elixhauser	Indicator for paralysis	0.000574	0.0240	0	1
tm1_psychosis_elixhauser	Indicator for psychoses	0.0325	0.177	0	1
tm1_pulmcirc_elixhauser	Indicator for pulmonary circulation disorders	0.00558	0.0745	0	1
tm1_pvd_elixhauser	Indicator for peripheral vascular disorders	0.0263	0.160	0	1
tm1_renal_elixhauser	Indicator for renal failure	0.0367	0.188	0	1
tm1_uncompdiabetes_elixhauser	Indicator for diabetes, uncomplicated	0.0987	0.298	0	1
tm1_valvulardz_elixhauser	Indicator for valvular disease	0.0315	0.175	0	1
tm1_wtloss_elixhauser	Indicator for weight loss	0.00139	0.0373	0	1
tm1_cerebrovasculardz_romano	Indicator for cerebrovascular disease	0.0283	0.166	0	1
tm1_chf_romano	Indicator for congestive heart failure	0.0319	0.176	0	1
tm1_dementia_romano	Indicator for dementia	0.00949	0.0970	0	1
tm1_hemiplegia_romano	Indicator for hemiplegia	0.00266	0.0516	0	1
tm1_hivaids_romano	Indicator for HIV/AIDS	0.00305	0.0552	0	1
tm1_metastatic_romano	Indicator for metastasis	0.00613	0.0780	0	1
tm1_myocardialinfarct_romano	Indicator for myocardial infarction	0.0169	0.129	0	1
tm1_pulmonarydz_romano	Indicator for pulmonary disease	0.102	0.302	0	1
tm1_tumor_romano	Indicator for tumor	0.0944	0.292	0	1
tm1_ulcer_romano	Indicator for ulcer	0.00480	0.0691	0	1
tm1_cost_dialysis	Total costs for dialysis, rounded to nearest 10	26.72	976.6	0	63,410
tm1_cost_emergency	Total costs for emergency, rounded to nearest 10	423.7	1,572	0	67,090
tm1_cost_home_health	Total costs for home health, rounded to nearest 10	220.5	1,396	0	56,830
tm1_cost_ip_medical	Total costs for inpatient medical, rounded to nearest 10	638.8	4,570	0	282,300

tm1_cost_ip_surgical	Total costs for inpatient surgical, rounded to nearest 10	978.5	6,575	0	279,930
tm1_cost_laboratory	Total costs for laboratory, rounded to nearest 10	330.9	949.4	-490	62,720
tm1_cost_op_primary_care	Total costs for outpatient primary care, rounded to nearest 10	473.9	1,872	0	240,290
tm1_cost_op_specialists	Total costs for outpatient specialists, rounded to nearest 10	866.2	1,546	0	41,720
tm1_cost_op_surgery	Total costs for outpatient surgery, rounded to nearest 10	846.6	2,659	0	75,790
tm1_cost_other	Total other costs, rounded to nearest 100	1,569	4,639	0	193,200
tm1_cost_pharmacy	Total costs for pharmacy, rounded to nearest 10	342.5	3,995	-10	153,250
tm1_cost_physical_therapy	Total costs for physical therapy, rounded to nearest 10	167.2	534.0	0	10,240
tm1_cost_radiology	Total costs for radiology, rounded to nearest 10	241.1	580.8	0	20,710
tm1_lasix_dose_count	Number of Lasix doses	0.0182	0.228	0	9
tm1_lasix_min_daily_dose	Minimum daily dose of Lasix	0.353	4.370	0	200
tm1_lasix_mean_daily_dose	Mean daily dose of Lasix	0.378	4.535	0	160
tm1_lasix_max_daily_dose	Maximum daily dose of Lasix	0.418	5.247	0	200
tm1_cre_tests	Number of c-reatinine tests	1.237	3.396	0	166
tm1_crp_tests	Number of c-reactive protein tests	0.000471	0.0226	0	2
tm1_esr_tests	Number of erythrocyte sedimentation rate tests	0.113	0.538	0	13
tm1_ghba1c_tests	Number of GHbA1c tests	0.385	0.748	0	9
tm1_hct_tests	Number of hematocrit tests	1.089	3.140	0	164
tm1_ldl_tests	Number of LDL tests	0.520	0.701	0	10
tm1_nt_bnp_tests	Number of BNP tests	0.0305	0.257	0	10
tm1_sodium_tests	Number of sodium tests	1.156	3.237	0	122
tm1_trig_tests	Number of triglycerides tests	0.483	0.681	0	12
tm1_cre_minlow	Indicator for low (< 0.84) minimum creatinine test result	0.222	0.416	0	1
tm1_cre_minhigh	Indicator for high (> 1.21) minimum creatinine test result	0.0391	0.194	0	1
tm1_cre_minnormal	Indicator for normal minimum creatinine test result	0.236	0.424	0	1
tm1_cre_meanlow	Indicator for low (< 0.84) mean creatinine test result	0.200	0.400	0	1

tm1_cre_meanhigh	Indicator for high (> 1.21) mean creatinine test result	0.0512	0.220	0	1
tm1_cre_meannormal	Indicator for normal mean creatinine test result	0.245	0.430	0	1
tm1_cre_maxlow	Indicator for low (< 0.84) maximum creatinine test result	0.178	0.383	0	1
tm1_cre_maxhigh	Indicator for high (> 1.21) maximum creatinine test result	0.0674	0.251	0	1
tm1_cre_maxnormal	Indicator for normal maximum creatinine test result	0.252	0.434	0	1
tm1_crp_minlow	Indicator for low (< 1) minimum c-reactive protein test result	0.000164	0.0128	0	1
tm1_crp_minhigh	Indicator for high (> 3) minimum c-reactive protein test result	0.000164	0.0128	0	1
tm1_crp_minnormal	Indicator for normal minimum c-reactive protein test result	6.15e-05	0.00784	0	1
tm1_crp_meanlow	Indicator for low (< 1) mean c-reactive protein test result	0.000164	0.0128	0	1
tm1_crp_meanhigh	Indicator for high (> 3) mean c-reactive protein test result	0.000164	0.0128	0	1
tm1_crp_meannormal	Indicator for normal mean c-reactive protein test result	6.15e-05	0.00784	0	1
tm1_crp_maxlow	Indicator for low (< 1) maximum c-reactive protein test result	0.000164	0.0128	0	1
tm1_crp_maxhigh	Indicator for high (> 3) maximum c-reactive protein test result	0.000164	0.0128	0	1
tm1_crp_maxnormal	Indicator for normal maximum c-reactive protein test result	6.15e-05	0.00784	0	1
tm1_esr_minlow	Indicator for low (< 1) minimum erythrocyte sedimentation rate test result	0	0	0	0
tm1_esr_minhigh	Indicator for high (> 20) minimum erythrocyte sedimentation rate test result	0.0218	0.146	0	1
tm1_esr_minnormal	Indicator for normal minimum erythrocyte sedimentation rate test result	0.0514	0.221	0	1
tm1_esr_meanlow	Indicator for low (< 1) mean erythrocyte sedimentation rate test result	0	0	0	0

tm1_esr_meanhigh	Indicator for high (> 20) mean erythrocyte sedimentation rate test result	0.0245	0.155	0	1
tm1_esr_meannormal	Indicator for normal mean erythrocyte sedimentation rate test result	0.0487	0.215	0	1
tm1_esr_maxlow	Indicator for low (< 1) maximum erythrocyte sedimentation rate test result	0	0	0	0
tm1_esr_maxhigh	Indicator for high (> 20) maximum erythrocyte sedimentation rate test result	0.0265	0.161	0	1
tm1_esr_maxnormal	Indicator for normal maximum erythrocyte sedimentation rate test result	0.0470	0.212	0	1
tm1_ghba1c_minlow	Indicator for low (< 4) minimum GHbA1c test result	4.10e-05	0.00640	0	1
tm1_ghba1c_minhigh	Indicator for high (> 5.7) minimum GHbA1c test result	0.123	0.329	0	1
tm1_ghba1c_minnormal	Indicator for normal minimum GHbA1c test result	0.146	0.353	0	1
tm1_ghba1c_meanlow	Indicator for low (< 4) mean GHbA1c test result	4.10e-05	0.00640	0	1
tm1_ghba1c_meanhigh	Indicator for high (> 5.7) mean GHbA1c test result	0.130	0.336	0	1
tm1_ghba1c_meannormal	Indicator for normal mean GHbA1c test result	0.140	0.347	0	1
tm1_ghba1c_maxlow	Indicator for low (< 4) maximum GHbA1c test result	4.10e-05	0.00640	0	1
tm1_ghba1c_maxhigh	Indicator for high (> 5.7) maximum GHbA1c test result	0.133	0.339	0	1
tm1_ghba1c_maxnormal	Indicator for normal maximum GHbA1c test result	0.137	0.344	0	1
tm1_hct_minlow	Indicator for low (< 35.5) minimum hematocrit test result	0.0639	0.245	0	1
tm1_hct_minhigh	Indicator for high (> 48.6) minimum hematocrit test result	0.00679	0.0821	0	1
tm1_hct_minnormal	Indicator for normal minimum hematocrit test result	0.375	0.484	0	1
tm1_hct_meanlow	Indicator for low (< 35.5) mean hematocrit test result	0.0424	0.202	0	1

tm1_hct_meanhigh	Indicator for high (> 48.6) mean hematocrit test result	0.00787	0.0884	0	1
tm1_hct_meannormal	Indicator for normal mean hematocrit test result	0.396	0.489	0	1
tm1_hct_maxlow	Indicator for low (< 35.5) maximum hematocrit test result	0.0242	0.154	0	1
tm1_hct_maxhigh	Indicator for high (> 48.6) maximum hematocrit test result	0.0119	0.109	0	1
tm1_hct_maxnormal	Indicator for normal maximum hematocrit test result	0.410	0.492	0	1
tm1_ldl_minlow	Indicator for low (< 50) minimum LDL test result	0.0155	0.124	0	1
tm1_ldl_minhigh	Indicator for high (> 99) minimum LDL test result	0.204	0.403	0	1
tm1_ldl_minnormal	Indicator for normal minimum LDL test result	0.198	0.398	0	1
tm1_ldlmeanlow	Indicator for low (< 50) mean LDL test result	0.0127	0.112	0	1
tm1_ldlmeanhigh	Indicator for high (> 99) mean LDL test result	0.211	0.408	0	1
tm1_ldlmeannormal	Indicator for normal mean LDL test result	0.134	0.340	0	1
tm1_ldl_maxlow	Indicator for low (< 50) maximum LDL test result	0.0117	0.108	0	1
tm1_ldl_maxhigh	Indicator for high (> 99) maximum LDL test result	0.218	0.413	0	1
tm1_ldl_maxnormal	Indicator for normal maximum LDL test result	0.127	0.333	0	1
tm1_nt_bnp_minlow	Indicator for low (< 100) minimum BNP test result	0.00488	0.0697	0	1
tm1_nt_bnp_minhigh	Indicator for high (> 450) minimum BNP test result	0.00980	0.0985	0	1
tm1_nt_bnp_minnormal	Indicator for normal minimum BNP test result	0.00543	0.0735	0	1
tm1_nt_bnp_meanlow	Indicator for low (< 100) mean BNP test result	0.00668	0.0815	0	1
tm1_nt_bnp_meanhigh	Indicator for high (> 450) mean BNP test result	0.0103	0.101	0	1
tm1_nt_bnp_meannormal	Indicator for normal minimum BNP test result	0.00344	0.0586	0	1
tm1_nt_bnp_maxlow	Indicator for low (< 100) maximum BNP test result	0.00646	0.0801	0	1
tm1_nt_bnp_maxhigh	Indicator for high (> 450) maximum BNP test result	0.0106	0.102	0	1
tm1_nt_bnp_maxnormal	Indicator for normal minimum BNP test result	0.00344	0.0586	0	1

tm1_sodium_minlow	Indicator for low (< 135) minimum sodium test result	0.0403	0.197	0	1
tm1_sodium_minhigh	Indicator for high (> 145) minimum sodium test result	0.000615	0.0248	0	1
tm1_sodium_minnormal	Indicator for normal minimum sodium test result	0.438	0.496	0	1
tm1_sodium_meanlow	Indicator for low (< 135) mean sodium test result	0.0196	0.139	0	1
tm1_sodium_meanhigh	Indicator for high (> 145) mean sodium test result	0.000861	0.0293	0	1
tm1_sodium_meannormal	Indicator for normal mean sodium test result	0.459	0.498	0	1
tm1_sodium_maxlow	Indicator for low (< 135) maximum sodium test result	0.0109	0.104	0	1
tm1_sodium_maxhigh	Indicator for high (> 145) maximum sodium test result	0.00515	0.0715	0	1
tm1_sodium_maxnormal	Indicator for normal maximum sodium test result	0.464	0.499	0	1
tm1_trig_minlow	Indicator for low (< 50) minimum triglycerides test result	0.0318	0.176	0	1
tm1_trig_minhigh	Indicator for high (> 150) minimum triglycerides test result	0.0901	0.286	0	1
tm1_trig_minnormal	Indicator for normal minimum triglycerides test result	0.262	0.440	0	1
tm1_trig_meanlow	Indicator for low (< 50) mean triglycerides test result	0.0289	0.167	0	1
tm1_trig_meanhigh	Indicator for high (> 150) mean triglycerides test result	0.0972	0.296	0	1
tm1_trig_meannormal	Indicator for normal mean triglycerides test result	0.256	0.436	0	1
tm1_trig_maxlow	Indicator for low (< 50) maximum triglycerides test result	0.0279	0.165	0	1
tm1_trig_maxhigh	Indicator for high (> 150) maximum triglycerides test result	0.107	0.309	0	1
tm1_trig_maxnormal	Indicator for normal maximum triglycerides test result	0.251	0.434	0	1
tm1_gagne_sum	Total number of active illnesses	1.443	2.049	0	18

TABLE 2
Stata Commands

STATA command	Description
*clear the workspace clear all version 17 *change working directory and open data cd "C:\Users\gbruich\Ec 50\Lab 8\" use lab8_2023_results_python.dta, clear *Display all variables in the data describe *Report detailed information on all variables codebook	This code shows how to clear the workspace, change the working directory, and open a Stata data file. To change directories on either a mac or windows PC, you can use the drop down menu in Stata. Go to file -> change working directory -> navigate to the folder where your data is located. The command to change directories will appear; it can then be copied and pasted into your .do file. The describe and codebook commands will report information on what is included in the data set loaded into memory.
*Create variable in percentile ranks *Drop previously defined scalar max_rank cap scalar drop max_rank *Start by rank ordering the data based on yvar egen yvar_rank = rank(yvar) *Get maximum rank, automatically stored as r(max) sum yvar_rank *Store maximum rank as a scalar variable scalar max_rank = r(max) *Normalize rank so that maximum is 100 replace yvar_rank = 100* yvar_rank / max_rank	These commands show how to convert a variable yvar into percentile ranks, normalized so that the highest rank is 100. The line "cap scalar drop max_rank" will clear from memory any saved scalar variables with the name max_rank, and otherwise proceed to the next line. As in Labs 1 and 2, we start using egen and the rank() function to generate a new variable that rank orders yvar. Then to normalize the variable, we divide it by the maximum rank and multiply by 100. The maximum rank is saved temporarily as r(max) after the sum command. I store it as a "scalar variable" called max_rank, and use that variable in the denominator in the last line.
*Create new indicator variable gen dvar= 0 replace dvar = 1 if wvar > 55	These commands show how to generate a new indicator variable called dvar. In the example, dvar equals 1 if another variable wvar is greater than 55.
*Summary stats *Observations with dvar equal to 1 sum yvar if dvar == 1	These commands report means and standard deviations for *yvar* for observations meeting certain criteria: when another variable in the data is equal to 1. To report summary statistics for multiple variables, list them next to each other with no commas: *sum yvar wvar xvar*
*install bin scatter command ssc install binscatter *Bin scatter plot – connected dots binscatter yvar xvar, linetype(connect) graph export figure2_connected.png, replace *Bin scatter plot – connected dots for two groups binscatter yvar xvar, linetype(connect) **by(race)** graph export figure2_connected_race.png, replace	These commands show how to create binned scatter plots. The first command installs binscatter, which only has to be done once. The second block of code shows how to create a binned scatter plot where a variable yvar is along the y-axis and a variable xvar is along the x-axis. It will connect the dots with a line. The third block of code shows how to draw a binned scatter plot for two groups defined by the variable "race." We do this by adding the by(race) option. The commands beginning with "graph export" save the graphs as .png files.

*close any possibly open log-files cap log close *start a log file log using lab8.log, replace *commands go here *close and save log file log close	These commands show how to start and close a log file, which will save a text file of all the commands and output that appears on the command window in stata. The first line is short for "capture log close" which will close any open log files, and otherwise just proceed to the next step. Then the "log using lab8.log, replace" starts the log file and changes the default in two ways. First, it changes the file type to have a .log file extension, which creates a plain text log file (which is readable in Gradesope so is important!). Second, it also adds the ", replace" option which will save over any other log file that has the same name. This is usually what you want. The rest of your lab code can go below the "log using lab8.log, replace" line. At the end of your do-file you can include the last line which is "log close" which will close and save the log-file.

TABLE 3
R Commands

R command	Description

#clear the workspace
rm(list=ls()) # removes all objects from the environment
cat('\014') # clears the console
 	
#Install and load haven package
if (!require(haven)) install.packages("haven"); library(haven)
 	
#Change working directory and load stata data set
setwd("C:/Users/gbruich/Ec 50/Lab 8")
dat <- read_dta("lab8_2023_results.dta")
 	
#Report detailed information on all variables
summary(dat)
	This sequence of commands shows how to open Stata datasets in R. The first block of code clears the work space. The second block of code installs and loads the "haven" package. The third block of code changes the working directory to the location of the data and loads in lab8_2023_results.dta. To change the working directory in R Studio, you can also use the drop down menu. Go to session -> set working directory -> choose working directory.
	
	The easiest way to open a Stata data set in R Studio is to use the drop down menu. Go to file, then import data set, and finally browse to locate the file you want to open. This option will be available after you install the haven package.
	
	The summary command will report information on what is included in the data set loaded into memory, including information on the number of missing observations NAs for each variable.

#Create variable in percentile ranks
#Start by rank ordering the data based on yvar
dat$yvar_rank <- rank(dat$yvar)
 	
#Store the maximum rank
max_rank <- max(dat$yvar_rank)
 	
#Normalize rank so that maximum is 100
dat$yvar_rank <- 100*dat$yvar_rank / max_rank
 	
# Create Function that will Calculate Percentile Ranks with NAs 	
 	
#Define function for percentile ranking
percentile_rank<-function(variable){
 	
#Convert to ranks, taking care of potential missing values
r <- ifelse(is.na(variable), NA, rank(variable, ties.method = "average"))
 	
#Return percentile rank = rank normalized so max is 100
100*r/max(r, na.rm = T)
}
 	
#Example using Function to Define ranks
dat$yvar_rank <-with(dat, percentile_rank(yvar))
	We used these commands in Lab 1 to convert a variable yvar into percentile ranks, normalized so that the highest rank is 100. We start using the rank() function to generate a new variable that rank orders yvar. Then to normalize the variable, we divide it by the maximum rank and multiply by 100. The code uses the max() function in R in the denominator to do the normalization.
	
	Unfortunately, the rank() function does not work as desired for data with missing values (NAs). But we can create our own function to do what we want that will work as intended in more complex data sets. This second block of code shows how to define a new function called percentile_rank() that will generate percentile ranks that assign missing values to NAs, and returns the percentile rank normalized to have a maximum rank of 100.
	
	The last line shows how to use the function to create the variable yvar_rank. The with() function in R takes two arguments: a data frame and an expression. The data frame argument is dat and the expression applies the new function we wrote to the variable yvar: percentile_rank(yvar).

#Create new indicator variable called dvar dat$dvar <- 0 dat$dvar[which(dat$xvar > 55)] <- 1 #Alternatively, use ifelse() function dat$dvar <- ifelse(dat$xvar > 55, 1, 0)	These commands illustrate an example of how to generate an indicator that equals 1 when xvar is greater than 55. The first two lines show how to start a variable that always equals 0 and then replace it equal to 1 if a logical condition is satisfied (i.e., xvar > 55). An alternative way to do it uses the ifelse() function, which takes three arguments: the logical condition, the value if the condition is satisfied, and the value of the condition is not satisfied.

#Summary stats for one variable
mean(dat$yvar, na.rm=TRUE)
 	
#Summary stats for observations with dvar==1
#Subset data
new_df <- subset(dat, dvar == 1)
 	
#Report mean
mean(new_df$yvar, na.rm=TRUE)
 	
#Alternatively, do it all at once using the with() function
with(subset(dat, dvar == 1), mean(yvar, na.rm=TRUE))
	We used these commands in Lab 1. These commands report mean of *yvar*. The first line calculates these statistics across the full sample.
	
	The other lines illustrate how to calculate these statistics for observations meeting certain criteria: when another variable in the data is equal to 1.
	
	The subset() function will pick out only the observations in a data frame that meet certain criteria. One way to proceed is to create a new data frame and then apply the mean() function to yvar in this new data frame. The second way to proceed is to do it all at once using the with() function. The with() function in R takes two arguments: a data frame and an expression. The data frame argument is dat and the expression applies the mean() function to the variable yvar: mean(yvar).

#install ggplot and statar packages
if (!require(tidyverse)) install.packages("tidyverse"); library(tidyverse)
if (!require(ggplot2)) install.packages("ggplot2"); library(ggplot2)
if (!require(statar)) install.packages("statar"); library(statar)
 	
#Bin scatter plot – connected dots
ggplot(dat, aes(x = xvar , y = yvar)) +
stat_binmean(n = 20, geom = "line") +
stat_binmean(n = 20, geom = "point")
 	
#Save graph
ggsave("binscatter_connected.png")
 	
#Bin scatter plot – connected dots
ggplot(dat, aes(x = xvar , y = yvar, **color = race**)) +
stat_binmean(n = 20, geom = "line") +
stat_binmean(n = 20, geom = "point")
 	
#Save graph
ggsave("binscatter_connected_race.png")
	We used these commands in Labs 1 and 2 to create binned scatter plots. The first lines install packages, including the statar package so that we can use the stat_binmean() function with ggplot.
	
	The second block of code shows how to create a binned scatter plot where a variable yvar is along the y-axis and a variable xvar is along the x-axis. It will connect the dots with a line.
	
	The third block of code shows how to do this separately by The third block of code shows how to draw a binned scatter plot for two groups defined by the variable "race." We do this by adding the "color = race" option.