```
/* Activity 3 */

/* This code assumes patient- and site-level information follow the configuration of the Excel data upload template and are stored in a data file labeled "patient_data_A3_child.xlsx". The Excel file should contain six sheets titled: (1) Survey Information-CHILDREN+ADO, (2) Configuration, (3) National Information, (4) ART Clinic Information, (5) Survey Participants, and (6) Participant Treatments-ChildAdo.*/

/* HIV drug resistance sequences are assumed to be in FASTA file format and stored in a data file labeled "FASTA_A3_child.xlsx". Sequence identification numbers and eligible case specimen identifiers must be identical and follow WHO convention. */

/* Both data files should be in the same directory, e.g., "C:/Documents" */



/* Clear any previous output */

clear


/* Set working directory to the directory containing all data files. For example, if directory is "C:/Documents", run the following code */

cd "C:/Documents"


/* Import the "National Information" sheet of the Excel Upload Template, storing the first row as headings and changing all header names to uppercase. Save as a file named "NATIONAL_INFORMATION_CHILD_ADO.dta". */

import excel using "patient_data_A3_child.xlsx", sheet("National Information") cellrange(A2) firstrow case(upper)

save "NATIONAL_INFORMATION_CHILD_ADO", replace

clear


/* Import the "ART Clinic Information" sheet of the Excel Upload Template, storing the first row as headings and changing all header names to uppercase. Save as a file named "ART_CLINIC_INFORMATION_CHILD_ADO.dta". */

import excel using "patient_data_A3_child.xlsx", sheet("ART Clinic Information") cellrange(A2) firstrow case(upper)
```

```
save "ART_CLINIC_INFORMATION_CHILD_ADO", replace

clear


/* Import the "Survey Participants" sheet of the Excel Upload Template, storing the first row as headings and changing all header names to uppercase. Save as a file named "SURVEY_PARTICIPANTS_CHILD_ADO.dta". */

import excel using "patient_data_A3_child.xlsx", sheet("Survey Participants") firstrow case(upper)

save "SURVEY_PARTICIPANTS_CHILD_ADO", replace

clear


/* Import the "Participant Treatments-ChildAdo" sheet of the Excel Upload Template, storing the first row as headings and changing all header names to uppercase. Save as a file named "PARTICIPANT_TREATMENTS_CHILD_ADO.dta". */

import excel using "patient_data_A3_child.xlsx", sheet("Participant Treatments-ChildAdo") firstrow case(upper)

save "PARTICIPANT_TREATMENTS_CHILD_ADO", replace

clear



/* This code prepares the resistance data imported from the Stanford HIVdb database */


/* Import the HIVdb resistance data, storing the first row as heading and changing all header names to uppercase. Given the file name "FASTA_A3_child.xlsx", run */

import excel using "FASTA_A3_child.xlsx", sheet("ResistanceSummary") firstrow case(upper)

/* Rename SEQUENCENAME as PARTICIPANTID */

rename SEQUENCENAME PARTICIPANTID

/* Drop all cells without a subject ID */

drop if missing(PARTICIPANTID)

/* Drop all unnecessary variables */

drop *SCORE ALGORITHM* STRAIN GENES PI* NRTI* NNRTI* INSTI*

/* Replace "NA" and "None" with the missing symbol */
```

```
destring, ignore("NA" "None") replace

/* For each of the resistance level variables, classify as a binary resistance indicator, with levels 1-2 corresponding to susceptible (no HIVDR), and levels 3-5 corresponding to HIVDR. */

ds *LEVEL

local plist = r(varlist)

foreach i of local plist {

        replace `i' = 0 if `i' < 3 & !missing(`i')

        replace `i' = 1 if `i' >= 3 & !missing(`i')

}

/* Rename resistance type variables and save changes to the modified dataset */

rename *LEVEL *_RES

/* Generate variable for DTG-specific resistance */

gen DTG_ADR = DTG_RES

/* Generate variable for Any Boosted PI resistance, defined as any resistance to ATV_r, DRV_r, or LPV_r */

egen ANY_PI = rowmax(ATVR_RES DRVR_RES LPVR_RES)

/* Generate variable for Any NRTI resistance */

egen ANY_NRTI = rowmax(ABC_RES AZT_RES D4T_R DDI_RES FTC_RES TDF_RES)

/* Generate variable for Any NNRTI resistance, defined as any resistance to NVP or EFV */

egen ANY_NNRTI = rowmax(EFV_RES NVP_RES)

/* Generate variable for Any INI resistance */

egen ANY_INI = rowmax(BIC_RES DTG_RES EVG_RES RAL_RES)

/* Generate variable for Any ADR resistance, as defined in the protocol */

egen ANY_ADR = rowmax(ANY_PI ANY_NRTI ANY_NNRTI ANY_INI)


save RESISTANCE_SUMMARY, replace



/* This code prepares the national-level data */
```

```
/* Remove previous dataset, then load the national-level data*/

clear

use NATIONAL_INFORMATION_CHILD_ADO.dta

/* Rename variable for total number of clinics providing ART to adults */

rename NADULTCLINICSSAMPLINGTABLE C_ADULT

/* Rename variable for total number of adults receiving ART */

rename NADULTSARTSAMPLINGTABLE N_ADULT

/* Rename variable for number of clinics serving adults to be sampled */

rename NADULTCLINICSSAMPLED SAMPLE_C_ADULT

/* Rename variable for proportion of adult patients on DTG-containing regimens */

rename PROPADULTSONDTG P_DTG_ADULT

/* Rename variable for total number of clinics providing ART to children/adolescents, excluding clinics selected for the adult survey */

rename NCHILDADOCLINICSSAMPTABLE C_CHILD

/* Rename variable for total number of children/adolescents receiving ART, excluding children/adolescents selected for the adult survey */

rename NCHILDADOARTSAMPLINGTABLE N_CHILD

/* Rename variable for number of clinics serving children/adolescents to be sampled */

rename NCHILDADOCLINICSSAMPLED SAMPLE_C_CHILD

/* Rename variable for proportion of child/adolescent patients on DTG-containing regimens */

rename PROPCHILDADOONDTG P_DTG_CHILD

/* Save changes */

save NATIONAL_INFORMATION_CHILD_ADO.dta, replace


/* This code prepares the clinic-level data */

/* Remove previous dataset, then load the clinic-level data */

clear

use ART_CLINIC_INFORMATION_CHILD_ADO.dta

/* Rename the variables for site code, adult clinic size, child clinic size, and clinics serving both adults and children/adolescents */
```

```
rename UNIQUE3LETTERCLINICCODE SITECODE

rename N_ADULTCLINIC_SIZE ADULT_CLINIC_SIZE

rename N_CHILDADOCLINIC_SIZE CHILD_CLINIC_SIZE

rename CLINIC_BOTHADULTCHILDADO_ART CLINIC_ADULT_CHILD_ART

/* Save changes */

save ART_CLINIC_INFORMATION_CHILD_ADO, replace



/* This code prepares the patient-level data on ARV treatment regimen */


/* Remove previous dataset, then load the treatment regimen data */

clear

use PARTICIPANT_TREATMENTS_CHILD_ADO.dta

/* Exclude observations missing a subject ID or belonging to the adult population */

drop if missing(PARTICIPANTID) | substr(PARTICIPANTID, 23, 1) != "c"

/* Rename ARV drug types so that all variable names start with a letter */

replace ARVDRUG = "ARV_" + ARVDRUG

/* Rename prior ARV drug types to differentiate them from current drug types and then drop the
current ART indicator variable */

replace ARVDRUG = "PRIOR_" + ARVDRUG if CURRENTARTYN == "N"

drop CURRENTARTYN

/* Generate binary DTG variable corresponding to 1 if patient is on a DTG-containing regimen and 0 if
patient is on a non-DTG-containing regimen */

gen TEMP_DTG = cond(inlist(ARVDRUG, "ARV_DTG", "ARV_TLD", "ARV_JUL"), 1, 0)

by PARTICIPANTID, sort: egen DTG = max(TEMP_DTG)

drop TEMP_DTG

/* Reformat the ARVDRUG variable so that each ARV drug type corresponds to a new binary variable,
set to 1 if the patient is taking that drug, and set to 0

if not */

gen ON = 1
```

```
reshape wide ON, i(PARTICIPANTID) j(ARVDRUG) string

rename ON* *

/* Save changes */

save PARTICIPANT_TREATMENTS_CHILD_ADO, replace



/* This code prepares the patient-level data on other variables */


/* Remove previous dataset, then load the patient-level data */

clear

use SURVEY_PARTICIPANTS_CHILD_ADO.dta

/* Recode unknown values as the missing symbol */

recode DATE* (9999 = .)

recode AGE (-9 = .)

foreach var of varlist BREASTFEEDINGSTATUS PREGNANCYSTATUS CURRENTART {

        replace `var' = "." if `var' == "UNK"

}

/* Drop lab specimen code variable */

drop LABSPECIMENCODE

/* Drop if missing subject ID or patient is in the adult population */

drop if missing(PARTICIPANTID) | substr(PARTICIPANTID, 23, 1) != "c"

/* Convert VL copies variable from string to integer */

destring VIRALLOADVALUECOPIESML, replace

/* Generate viral suppression variable (<1000 copies/mL) */

gen VIRAL_SUPPRESSION = cond(VIRALLOADVALUECOPIESML < 1000, 1, 0) if
VIRALLOADVALUECOPIESML != .

/* Save changes */

save SURVEY_PARTICIPANTS_CHILD_ADO.dta, replace
```

```
/* Merge all the datasets and save the combined data into one .dta file */


/* Remove previous dataset, then load the patient-level data */

clear

use SURVEY_PARTICIPANTS_CHILD_ADO.dta

/* Generate a variable for total number of adults on ART to match on when merging */

gen N_ADULT = .

/* Merge in the national data, dropping variables and observations that are unnecessary */

append using NATIONAL_INFORMATION_CHILD_ADO, keep(N_ADULT C_ADULT SAMPLE_C_ADULT
P_DTG_ADULT N_CHILD C_CHILD SAMPLE_C_CHILD P_DTG_CHILD)

drop if missing(PARTICIPANTID) & missing(N_ADULT)

/* Replace the necessary variables with the values provided in the last row. */

foreach var of varlist N_ADULT C_ADULT SAMPLE_C_ADULT P_DTG_ADULT N_CHILD C_CHILD
SAMPLE_C_CHILD P_DTG_CHILD {

        replace `var' = `var'[_N]

}

/* Drop if missing subject ID or patient is in the adult population */

drop if missing(PARTICIPANTID) | substr(PARTICIPANTID, 23, 1) != "c"

/* Use a many-to-one merge to merge the ART clinic data on the site code */

merge m:1 SITECODE using ART_CLINIC_INFORMATION_CHILD_ADO, keep(match) nogenerate

/* Merge in the treatment regimen data by subject ID */

merge 1:1 PARTICIPANTID using PARTICIPANT_TREATMENTS_CHILD_ADO, keep(match) nogenerate

/* Merge in the HIVDR data by subject ID */

merge 1:1 PARTICIPANTID using RESISTANCE_SUMMARY, nogenerate

/* save merged data */

save ALL_DATA.dta, replace


/* This code is for defining survey weights, declaring survey design and analyzing data for viral
suppression and HIVDR */
```

```
/* Remove previous dataset and load in the combined data */

clear

use ALL_DATA.dta

/* Generate the first stage site sampling weight, calculated as described in the protocol, with the form depending on if the clinic serves both adults and children/adolescents or only children/adolescents */

by SITECODE, sort: gen P_SELECT_ADULT_CHILD = (SAMPLE_C_ADULT*ADULT_CLINIC_SIZE)/N_ADULT + ((SAMPLE_C_CHILD*CHILD_CLINIC_SIZE)/N_CHILD) * (1-((SAMPLE_C_ADULT*ADULT_CLINIC_SIZE)/N_ADULT)) if CLINIC_ADULT_CHILD_ART == "Y"

by SITECODE, sort: gen P_SELECT_CHILD_ONLY = (SAMPLE_C_CHILD*CHILD_CLINIC_SIZE)/N_CHILD if CLINIC_ADULT_CHILD_ART == "N"

by SITECODE, sort: gen WEIGHT1 = 1/P_SELECT_ADULT_CHILD if CLINIC_ADULT_CHILD_ART == "Y"

by SITECODE, sort: replace WEIGHT1 = 1/P_SELECT_CHILD_ONLY if CLINIC_ADULT_CHILD_ART == "N"


/* Create the variable for stratification by regimen (DTG vs. non-DTG) */

egen STRATA = group(DTG)

/* Generate the variable for total DTG and non-DTG case specimens per sampled clinic. This corresponds to the second stage populations. */

gen CLINIC_POP = cond(DTG == 1, N_CHILD_ADO_DTG, N_CHILD_ADO_NONDTG)

/* Generate the second stage sampling weight, calculated as the number of patients on ART observed during the survey period, divided by the number of specimens sampled and with successful viral load testing */

by SITECODE DTG, sort: gen WEIGHT2 = CLINIC_POP/_N

/* Generate the sampling weight as the product of the first and second stage sampling weights */

gen WEIGHTS = WEIGHT1 * WEIGHT2

/* Set the stratified two-stage clustered survey design with finite population correction. Standard errors for single-unit strata will be set to the average of the standard errors for other strata */

svyset SITECODE [pweight = WEIGHTS], fpc(C_CHILD) || _n, strata(STRATA) fpc(CLINIC_POP) singleunit(scaled)


/* Analyses for Outcomes 1 & 2 */
```

```
/* Obtain estimates and confidence intervals for the prevalence of viral suppression among all patients */

svy: proportion VIRAL_SUPPRESSION


/* If Wald confidence intervals are desired instead, uncomment the following */

/* svy: proportion VIRAL_SUPPRESSION, citype(wald)*/


/* To obtain the design effect for this estimate, uncomment the following */

/* estat effects */


/* Obtain estimates and confidence intervals for the prevalence of viral suppression among patients on DTG-containing regimens */

svy, subpop(if DTG==1): proportion VIRAL_SUPPRESSION

/* Store estimated prevalence of viral suppression among patients receiving DTG-containing regimens for use in ADR estimates */

gen P_DTG_VS = e(b)[1,2]


/* Obtain estimates and confidence intervals for the prevalence of viral suppression among patients on non-DTG-containing regimens */

svy, subpop(if DTG==0): proportion VIRAL_SUPPRESSION

/* Store estimated prevalence of viral suppression among patients receiving non-DTG-containing regimens for use in ADR estimates */

gen P_NONDTG_VS = e(b)[1,2]



/* Analyses for Outcome 3*/


/* Example: Obtain prevalence and variance estimates of viral suppression among men */

svy, subpop(if GENDER == "M"): proportion VIRAL_SUPPRESSION
```

```
/* Example: Obtain prevalence and variance estimates of viral suppression among individuals who are breastfeeding */

svy, subpop(if BREASTFEEDINGSTATUS == "Y"): proportion VIRAL_SUPPRESSION



/* Analyses for Outcomes 4 and 5 */


/* Run cluster bootstrap to estimate standard errors*/

/* Install the 'bsweights' package to run bootstrap variance estimation in complex survey data */

net install bsweights, from(http://staskolenikov.net/stata)

/* Generate a byte variable as a placeholder for the strata variable */

generate byte _one=1


/* Generate the weight that accounts for stratification by regimen */

/* Obtain the total number of individuals observed during the survey period, for each regimen and sampled across clinics. */

by DTG, sort: egen REG_TOTAL = total(CLINIC_POP)

/* Multiply by the estimated regimen-specific viral non-suppression prevalence using Outcomes 1 and 2 */

gen REG_TOTAL_VNS = REG_TOTAL*cond(DTG==1, 1-P_DTG_VS, 1-P_NONDTG_VS)

/* Total number of sampled individuals with VNS, for each regimen */

by DTG, sort: egen REG_SAMPLED_TOTAL_VNS = total(VIRAL_SUPPRESSION == 0) if VIRAL_SUPPRESSION == 0

/* Stratification weight is the regimen-specific total number with viral non-suppression by the regimen-specific number of sampled individuals with viral non-suppression */

gen STRAT_WT = REG_TOTAL_VNS / REG_SAMPLED_TOTAL_VNS


/* Generate non-response weight to account for genotyping failure */

/* Total number of sampled individuals with VNS, for each regimen and each clinic*/
```

```
by SITECODE DTG, sort: egen REG_SAMPLED_VNS = total(VIRAL_SUPPRESSION == 0) if
VIRAL_SUPPRESSION == 0

/* Total number of sampled individuals with VNS and successful genotyping, for each regimen and each
clinic*/

by SITECODE DTG, sort: egen REG_SAMPLED_VNS_GENO = total(ANY_ADR != .) if ANY_ADR != .

gen NON_RESPONSE_WT = REG_SAMPLED_VNS / REG_SAMPLED_VNS_GENO

replace NON_RESPONSE_WT = 1 if VIRAL_SUPPRESSION == 1


/* Generate the sampling weight as the product of the stratification and non-response weights. Clinic
sampling weights are ignored. */

gen ADR_WT = STRAT_WT*NON_RESPONSE_WT


/* Set the initial survey design with linearized variance */

svyset SITECODE [pw=ADR_WT], strata(_one)|| _n, strata(STRATA) fpc(CLINIC_POP) singleunit(scaled)

/* Obtain bootstrap replication weights with 100 replications and n-1 resampled units from each
stratum */

bsweights bsw, reps(100) n(-1)

/* Define new survey design with bootstrap replication weights and cluster bootstrap variance. Finite
population correction cannot be used with cluster bootstrap. */

svyset SITECODE [pweight=ADR_WT], strata(_one) bsrweight(bsw*) vce(bootstrap) || _n,
strata(STRATA) singleunit(scaled)


/* Obtain estimates and confidence intervals for the prevalence of overall drug resistance among all
patients with VNS */

svy, subpop(if VIRAL_SUPPRESSION==0): proportion ANY_ADR


/* To obtain the design effect for this estimate, uncomment the following */

/* estat effects */


/* Obtain estimates and confidence intervals for the prevalence of DTG-specific drug resistance among
individuals taking DTG-containing regimens and with VNS */

svy, subpop(if DTG==1 & VIRAL_SUPPRESSION==0): proportion DTG_ADR
```