

Stata Cheat Sheet

Large Scale Data - Midterm Review

Example: Hospital Discharge Data

This cheat sheet uses **SPARCS hospital discharge data** as the running example:

- **Observations:** Individual hospital admissions (e.g., 150,000 records)
- **Variables:** Age, sex, admission type, length of stay, total charges
- **Geographic:** Multiple counties with socioeconomic data
- **Outcome:** Total costs (continuous), expensive stay flag (binary)
- **Key challenges:** Missing values (999/9999), top-coding, strings

PART 1: CONCEPTS & LOGIC

1. Data Types: Pros, Cons, & Use Cases

Cross-sectional Surveys

Examples: BRFSS, NHANES, NHIS

Advantages: Relatively low cost, quick large samples, ideal for prevalence estimates, nationally representative, population-level health estimates

Disadvantages: Cannot track individual changes, difficult causality (associations only), compare different people not same person, declining response rates, cannot control unmeasured time-invariant characteristics

Best for: “What is diabetes prevalence?” “Insurance coverage by income?” “Physical activity guideline adherence?”

Longitudinal Surveys

Examples: MEPS, National Longitudinal Survey of Youth

Advantages: Track same individuals (within-person), support fixed effects, stronger causal inference, observe temporal sequences, control time-invariant characteristics

Disadvantages: High cost, attrition bias, long collection period, learning effects, complex analysis

Best for: “Does having baby reduce sleep?” “Does retirement improve mental health?” “Job loss effect on health?”

Claims/Hospital Discharge Data

Examples: SPARCS (NY), Medicare Claims

Advantages: Very large samples (population-level), actual costs/utilization, tracks across providers, less missing data, readmission tracking, covers entire populations

Disadvantages: No clinical details (labs, vitals), inaccurate diagnoses (billing focus), excludes non-users, limited socioeconomic info, coding errors, cannot capture out-of-network care

Best for: “Hospitalization costs by county?” “90-day readmission rate?” “Length of stay patterns?” “High-cost predictors?”

Quality Metrics: **Good:** Readmission, LOS, utilization, mortality — **Bad:** BP/HbA1c control (need clinical data)

Electronic Health Records (EHR)

Examples: Hospital system EHR, Epic, Cerner

Advantages: Rich clinical info (labs, vitals, meds, imaging), longitudinal visits, real-world data, unstructured notes, clinical decisions

Disadvantages: Single system only (selection bias), variable quality, missing external visits, hard to standardize, local practice patterns (not generalizable)

Best for: “HbA1c control in our system?” “Real-world drug effectiveness?” “Clinical reminders impact?”

Quality Metrics: **Good:** HbA1c/BP control, medication adherence — **Bad:** Cross-system readmissions

Text-based Data

Examples: Physician notes, chief complaints, radiology reports

Advantages: Rich unstructured info, details not in structured fields, clinical context

Disadvantages: Requires NLP, complex analysis, inconsistent quality, time-intensive

2. Core Regression Concepts

Survey Weights

Why? (1) Design oversampling (minorities for adequate N), (2) Differential response rates by group, (3) Non-representative samples

Understanding: Weight = how many people each respondent represents. Weight=1.5 means represents 1.5 people. Allows generalization to population. Adjusts for unequal selection & non-response.

When: Survey data (BRFSS, NHANES, MEPS) for population-representative results

Stata: svyset [pweight=wt]; svy: reg y x

Categorical vs Continuous

Categorical: Use i. prefix, Stata creates dummies, first = reference (omitted), each coef = difference from reference. Examples: race, admission type, education

Continuous: Use c. or direct name, coef = effect of 1-unit increase, assumes linearity. Examples: age, BMI, income, costs

Choose categorical when: Want separate effects, non-linear relationship, meaningful groups, unequal intervals

Logistic vs Linear Regression

For binary outcomes, both are acceptable but interpret differently!

| Feature | Logistic | Linear |
|----------------|---------------------|------------------------|
| Command | logistic or logit | reg |
| Output | Odds Ratios (OR) | Coefficients |
| Interpretation | Fold-change in odds | Percentage point diff. |

Logistic Interpretation Example:

OR = 1.5: “Each \$1000 increase in county income is associated with 50% increase in the *odds* of expensive stay (odds multiply by 1.5)”

Linear Interpretation Example:

Coef = 0.04: “Each \$1000 increase in county income is associated with 4 percentage point increase in *probability* of expensive stay (e.g., from 10% to 14%)”

Interaction Terms

Purpose: Test whether the relationship between X and Y varies depending on the value of variable Z.

Research Question Examples:

- Does the effect of income on health vary by sex?
- Does drug effectiveness vary by age group?
- Do hospitalization costs respond differently to county income depending on admission type?

Stata syntax:

```
// ## includes main effects + interaction
reg outcome vari1##vari2

// Categorical x Continuous (MUST use c.!)
reg outcome i.sex##c.age

// Categorical x Categorical
reg outcome i.sex##i.race
```

Individual Fixed Effects

When to use:

- Have longitudinal/panel data (same person, multiple time points)
- Want to control for time-invariant individual factors
- Research question focuses on “within-person changes”
- Concerned about unmeasured confounding from stable traits

Advantages:

- Controls for ALL time-invariant characteristics (observed or unobserved)
- Stronger foundation for causal inference
- Each person serves as their own control
- Eliminates selection bias from stable individual traits

Example: Does having a baby reduce sleep? Compare same person before vs after baby.

Stata commands:

```
// Set panel data structure
xtset person_id time_var

// Fixed effects regression
xtreg outcome predictors, fe
```

Robust Standard Errors

Why use?

- **Heteroskedasticity:** Error variance differs across observations
- Very common in large samples and cross-sectional data
- Makes standard errors more accurate and reliable
- Improves inference (p-values, confidence intervals)

Stata syntax:

```
reg outcome predictors, robust
// or shorthand:
reg outcome predictors, r
```

Disciplinary differences:

- **Economics:** Almost always uses robust SE
- **Biostatistics:** Not always standard practice
- **Exam:** Won’t lose points for not using, but understand why it’s useful

3. Statistical vs Practical Significance

Key Distinction:

| Statistical Significance | Practical Significance |
|--|---------------------------------------|
| $p < 0.05$ | Is difference large enough to matter? |
| Affected by sample size | NOT affected by sample size |
| Technical judgment | Substantive judgment |
| Can detect tiny effects in large samples | Focuses on real-world importance |

Detailed Example:

In BRFSS survey of 500,000 people, we find Gen Z and Millennials differ by 3 minutes in daily exercise (20 min vs 23 min), $p=0.007$.

Analysis:

- **Statistically significant:** $p < 0.05$
- **? Practically significant:** Debatable
 - 3 minutes may be too small for meaningful health benefits
 - But represents 15% relative difference (15%)
 - In large samples, even tiny differences achieve significance (“overpowered”)

Key principle: ALWAYS discuss BOTH statistical and practical significance in your interpretations! Don’t just report p-values.

4. Variable Types & Handling

Binary/Dummy Variables (0/1)

Definition: Flag variable coded as 0 or 1

Common uses:

- Calculate percentages (e.g., vaccination rate = mean of binary)
- As outcome variable (logistic or linear regression)
- As control variable in regression models
- Create subgroup indicators for analysis

Creation method:

```
// Standard approach
gen flag = 1 if condition
replace flag = 0 if opposite_condition

// Example: elderly patient flag
gen elderly = 1 if age >= 65 & age != .
replace elderly = 0 if age < 65
```

Verification (CRITICAL):

```
summ flag
// Check: mean 0-1, min=0, max=1

tab original_var flag
// Verify cross-tabulation is correct
```

Categorical Variables

Examples: Race, education level, admission type, age groups
Handling methods:

```
// Method 1: encode (string to numeric)
encode string_var, gen(categorical_var)

// Method 2: Manual creation
gen cat = 1 if condition1
replace cat = 2 if condition2
replace cat = 3 if condition3

// Use with i. prefix in regression
reg outcome i.categorical_var
```

Continuous Variables

Examples: Age, BMI, income, length of stay
Common handling:

```
// Top-coding (cap extreme outliers)
gen var_clean = var
replace var_clean = 1000000 ///
    if var > 1000000 & var != .

// Log transformation (right-skewed data)
gen log_var = log(var + 1)
// +1 to avoid log(0)
```

String Variables

Common operations:

```
// Convert to numeric (force = ignore errors)
destring string_var, gen(numeric_var) force

// Check what couldn't be converted
tab string_var if numeric_var == .

// String matching/searching
gen flag = strpos(string_var, "keyword") > 0
```

Missing Values

CRITICAL PRINCIPLE: Always include & var != . in conditions!
Why? Stata treats missing (.) as very large number (infinity).
Correct approach:

```
// CORRECT - protects missing values
replace var_clean = 100 if var > 100 & var != .

// WRONG - will set missing to 100!
replace var_clean = 100 if var > 100
```

Identifying missing:

```
// Recode special values to missing
replace var = . if var == 99
replace var = . if var == 999
replace var = . if var < 0
```

5. Data Management Operations

Merging Data

Four merge types:

- 1:1 Merge
- Scenario: Both datasets have 1 row per ID
 - Example: Merge NHANES questionnaire to demographics (both 1 row/person)
 - Syntax: merge 1:1 id using "file.dta"
- 1:M Merge
- Scenario: Master has 1 row per ID, Using has many rows per ID
 - Example: Merge visit data to medications per visit
 - Syntax: merge 1:m id using "file.dta"
- M:1 Merge (MOST COMMON!)
- Scenario: Master has many rows per ID, Using has 1 row per ID
 - Example: Merge county income to hospital records (many admissions/county)
 - Syntax: merge m:1 county using "file.dta"

Post-merge check (CRITICAL):

```
tab _merge
/*
_merge == 1: master only (no match in using)
_merge == 2: using only (no match in master)
_merge == 3: matched successfully
*/

// Usually keep master records and matched
```

```
keep if _merge == 1 | _merge == 3
drop _merge
```

Reshape/Transpose

Wide to Long:

```
reshape long var_prefix, i(id) j(time)
```

Long to Wide:

```
reshape wide var_name, i(id) j(time)
```

Append

Scenario: Combine datasets with same structure (stack rows vertically)

```
append using "second_dataset.dta"
```

PART 2: STATA CODING

A. Data Management
Setup & Import
Standard Setup

```
// Clear memory and set options
clear all
set more off

// Set working directory
cd "path/to/folder"

// Start logging
capture log close
log using "analysis.log", replace
```

Import Data


```
// Example: Load hospital discharge data
use "sparcs_hospital_data.dta", clear

// Load Stata file (general)
use "filename.dta", clear

// Import from SAS XPT (e.g., SPARCS source)
import sasxport5 "sparcs_raw.xpt", clear

// Import from CSV (hospital subset)
import delimited "hospital_subset.csv", ///
    clear firstrow

// Save processed data
save "sparcs_processed.dta", replace
```

 **Warning**
Always use clear or , clear option to avoid "no; data in memory" error.

Data Exploration
View & Describe

```
// View first 10 hospital admissions
list age los totalcosts county in 1/10

// View high-cost stays (>$100k)
list if totalcosts > 100000

// Describe all variables
describe

// Explore age and admission type
codebook age admission_type
```

Summary Statistics

```
// Summarize hospital costs
summ totalcosts

// Detailed summary with percentiles
summ totalcosts los age, d
// Shows: min, max, mean, median, p25, p75

// Summary by group (admission type)
bysort admission_type: summ totalcosts
```

Frequency Tables

```
// Admission type frequencies (show missing)
tab admission_type, m

// Age group distribution
tab age_group

// Cross-tabulation: admission type x sex
tab admission_type sex, row

// Expensive stay by admission type
tab expensive_stay admission_type, col

// Two-way table with both row & col %
tab admission_type admission_source, ///
    row col
```

Check Data Size

```
// Number of observations
display _N

// Number of variables
describe, short
```

Variable Creation
Basic Generation

```
// Create empty variable
gen newvar = .

// Create with value
gen age_squared = age^2

// Create constant
gen constant = 1
```

Replace Values

```
// Replace all values
replace varname = new_value

// Conditional replace
replace var = value if condition
```

Warning
CRITICAL: Use if var != . to protect missing values!

Binary Variables (0/1 Flags)

```
// Method 1: Standard approach
gen flag = 1 if condition
replace flag = 0 if !condition

// Method 2: One-liner
gen flag = (condition) if var != .

// Example: Elderly patient (age >= 65)
gen elderly = 1 if age >= 65 & age != .
replace elderly = 0 if age < 65

// Example: Expensive stay (>$100,000)
gen expensive_stay = 1 if totalcosts ///
> 100000 & totalcosts != .
replace expensive_stay = 0 if ///
totalcosts <= 100000

// VERIFY binary variable
summ flag
// mean should be 0-1, min=0, max=1
```

Categorical Variables

```
// Method 1: Manual creation
gen category = .
replace category = 1 if condition1
replace category = 2 if condition2
replace category = 3 if condition3

// Add value labels
label define cat_lbl 1 "Low" ///
2 "Medium" 3 "High"
label values category cat_lbl

// Method 2: recode
recode age (0/29=1) (30/49=2) ///
(50/max=3), gen(age_group)

// Method 3: encode string variable
encode string_var, gen(numeric_var)
```

Top-coding (Capping Outliers)

```
// Create clean version
gen cost_clean = cost

// Top-code at $1,000,000
replace cost_clean = 1000000 ///
if cost > 1000000 & cost != .

// Verify
summ cost, d
summ cost_clean, d
// Check: max of clean version = cap
```

Logarithmic Transformation

```
// Handle right-skewed data
// Step 1: Add small value to avoid log(0)
gen cost_plus1 = cost + 1

// Step 2: Take log
gen log_cost = log(cost_plus1)

// Alternative: only for positive values
gen log_cost = log(cost) if cost > 0
```

EGEN - Extended Generation

```
// Row operations
egen total = rowtotal(var1 var2 var3)
egen mean = rowmean(var1 var2 var3)
egen max = rowmax(var1 var2 var3)
egen min = rowmin(var1 var2 var3)

// Grouped statistics
egen mean_by_group = mean(var), ///
by(group_var)
```

```
// Example: County-level averages
egen avg_income_county = mean(income), ///
by(county_name)

// Count non-missing
egen count_nonmiss = count(var), ///
by(group)

// String concatenation
egen fullname = concat(first last), ///
punct(" ")
```

Missing Values
Identify & Recode

```
// Count missing
count if varname == .

// Summary shows N
summ varname
// Total N vs variable N

// Show missing in table
tab varname, m
```

Recode Missing Values

```
// Set specific values to missing
replace var = . if var == 99
replace var = . if var == 999
replace var = . if var < 0

// Example from BRFSS
replace height = . if height == 7777
replace height = . if height == 9999
```

Warning
Wrong: replace var = 100 if var > 100 (sets missing to 100!)
Right: replace var = 100 if var > 100 & var != .

Create Missing Indicator

```
// Flag for missing
gen miss_flag = (varname == .)

// Or explicitly
gen miss_flag = 0
replace miss_flag = 1 if varname == .
```

String Variables
Type Conversion

```
// Basic conversion (force ignores errors)
destring string_var, gen(num_var) force

// Check what couldn't be converted
tab string_var if num_var == .

// Example: Handle "120 +"
destring lengthofstay, gen(los_num) force
replace los_num = 120 ///
if lengthofstay == "120 +"
```

Numeric to String

```
// Convert number to string
gen str_var = string(numeric_var)

// With formatting
gen str_var = string(num_var, "%9.2f")
```

String to Categorical (encode)

```
// Create numeric with labels
encode string_var, gen(categorical_var)

// Example: Admission type
encode typeofadmission, ///
gen(admission_type_num)

// Verify
tab typeofadmission admission_type_num
```

String Manipulation

```
// Case conversion
gen upper = strupper(string_var)
gen lower = strlower(string_var)
gen proper = strproper(string_var)

// Extract substring (pos starts at 1!)
gen first5 = substr(string_var, 1, 5)
gen char2to4 = substr(string_var, 2, 3)

// Find substring position
gen pos = strpos(string_var, "keyword")
// Returns 0 if not found

// String length
gen length = strlen(string_var)

// Replace text
gen new = subinstr(string_var, ///
"old", "new", ..)
// Last argument: . = replace all

// Split string
split string_var, gen(part) parse("-")
// Creates: part1, part2, part3, ...
```

⚠ Warning

substr() starts at 1: substr(str, 1, 2) = first 2 chars

Data Merging

Merge Types (1:1, 1:M, M:1)

```
// 1:1 - Both datasets: 1 row per ID
merge 1:1 id_var using "file.dta"

// 1:M - Master: 1 row/ID, Using: many rows/ID
merge 1=m id_var using "file.dta"

// M:1 - Master: many rows/ID, Using: 1 row/ID
merge m=1 id_var using "file.dta"

// M:M - Both: many rows/ID (rare, avoid)
// Use joinby instead if needed
```

M:1 Merge Example (Most Common)

```
// Merge county data to individual records
// Main: Individual hospitalizations
//       (many records per county)
// Using: County characteristics
//       (one record per county)

use "hospital_data.dta", clear

// Rename if needed to match
rename hospitalcounty County_Name

// Perform M:1 merge
merge m=1 County_Name ///
      using "county_data.dta"

// CHECK merge results
tab _merge
/*
_merge values:
  1 = master only (hospital records
    with no county match)
  2 = using only (counties with no
    hospital records)
  3 = matched successfully
*/

// Keep what you want
keep if _merge == 1 | _merge == 3
// Keeps all hospital records

// Clean up
drop _merge
```

Merge Workflow

```
// Step 1: Check merge variable exists
describe merge_var

// Step 2: Check if unique (if "1" side)
duplicates report merge_var
// Should show 0 duplicates

// Step 3: Ensure variable names match
// If not, rename in one dataset
rename old_name new_name

// Step 4: Check variable types match
describe merge_var
// Both should be numeric or string

// Step 5: Perform merge
merge type merge_var using "file.dta"

// Step 6: Always check _merge!
tab _merge

// Step 7: Keep desired records
keep if _merge == 1 | _merge == 3

// Step 8: Drop _merge
drop _merge
```

💡 Tip

M:1 → Master(M):Using(1) — 1:M → Master(1):Using(M) — 1:1 → Master(1):Using(1)

Append (Stack Datasets)

```
// Combine datasets with same structure
use "data2020.dta", clear
append using "data2021.dta"
append using "data2022.dta"

// All rows are kept, stacked vertically
```

Data Reshaping

Wide ↔ Long

```
// Wide format:
// id  bp1  bp2  bp3  hr1  hr2  hr3
// 1   120  118  115  72   70   68

reshape long bp hr, i(id) j(round)

// Long format:
// id  round  bp  hr
// 1   1      120 72
// 1   2      118 70
// 1   3      115 68
```

Long to Wide

```
// Long format:
// id  round  bp  hr
// 1   1      120 72
// 1   2      118 70

reshape wide bp hr, i(id) j(round)

// Wide format:
// id  bp1  bp2  hr1  hr2
// 1   120  118  72   70
```

⚠ Warning

Always save before reshape!

Regression Analysis

Linear Regression

```
// Simple regression
reg outcome predictor

// Multiple regression
reg y x1 x2 x3

// With robust standard errors
reg y x1 x2, robust
// or
reg y x1 x2, r
```

Categorical Variables in Regression

```
// Use i. prefix for categorical
reg outcome continuous_var i.category

// Example
reg totalcosts lengthofstay ///
      i.agegroup i.admission_type

// Stata automatically:
// - Creates dummy variables
// - Omits first category (reference)
// - Shows each category coefficient
```

Continuous Variables

```
// Default: continuous
reg y x1 x2

// Explicit: c. prefix (optional)
reg y c.x1 c.x2
```

Interaction Terms

```
// Categorical x Categorical
reg y i.var1##i.var2
// ## includes main effects + interaction

// Categorical x Continuous
// IMPORTANT: Use c. for continuous!
reg y i.category##c.continuous

// Example: Do age effects vary by sex?
reg totalcosts i.sex##c.age ///
      i.admission_type, robust

// Only interaction (no main effects)
reg y i.var1#i.var2
```

⚠ Warning

MUST use c. for continuous: i.sex##c.age (not i.sex##age)

Logistic Regression

```
// Binary outcome (0/1)
logistic binary_y x1 x2 i.category
// Reports Odds Ratios (OR)

// Alternative: logit (reports log-odds)
logit binary_y x1 x2 i.category

// Example
logistic expensive_stay ///
      County_Income lengthofstay ///
      i.agegroup i.ED_flag
```

Linear vs Logistic Interpretation

```
// LINEAR regression on binary outcome
reg expensive_stay County_Income, r
// Coefficient: Percentage point difference
// Example: coef = 0.04
// Interpretation: "County income increase
// of $1000 associated with 4 percentage
// point increase in probability of
// expensive stay (e.g., 10% to 14%)"

// LOGISTIC regression
logistic expensive_stay County_Income
// Coefficient: Odds Ratio
// Example: OR = 1.02
// Interpretation: "County income increase
// of $1000 associated with 2% increase
// in the odds of expensive stay"
```

Survey Weights

```
// Set survey design
svyset [pweight = weight_var]
```

```
// Weighted regression
svy: reg y x1 x2 i.category

// Weighted logistic
svy: logistic binary_y x1 x2

// Why use weights?
// - Make results representative
// - Account for survey design
// - Adjust for non-response
```

Post-Regression Commands

```
// Test joint significance
reg y x1 x2 x3
test x1 x2
// Tests:  $\alpha_1 = \alpha_2 = 0$ 

// Predicted values
predict yhat

// Residuals
predict resid, residuals

// Margins (adjusted predictions)
reg y x1 i.group
margins group
// Shows predicted y for each group
```

Individual Fixed Effects

```
// For panel/longitudinal data
// Controls for all time-invariant
// individual characteristics

// Set panel structure
xtset person_id time_var

// Fixed effects regression
xtreg y x1 x2, fe

// Why use FE?
// - Within-person analysis
// - Control for unmeasured confounders
// - Stronger causal inference
```

Verification & Validation

Verify Binary Variables

```
// Create binary flag
gen flag = 1 if cost > 50000 & cost != .
replace flag = 0 if cost <= 50000

// CHECK 1: Summary statistics
summ flag
// mean: 0-1, min: 0, max: 1, N correct?

// CHECK 2: Cross-tabulation
tab flag, m
// Should show: 0, 1, and . only

// CHECK 3: Verify cutoff
summ cost if flag == 1
// min should be > 50000
summ cost if flag == 0
// max should be <= 50000
```

Verify Categorical Variables

```
// After encoding or recoding
tab old_var new_var
// Check mapping is correct

// With percentages
tab old_var new_var, row col
```

Verify Continuous Variables

```
// After transformation
summ original_var, d
summ clean_var, d
// Compare: mean, min, max, N

// Grouped summary
bysort group: summ var
// or
summ var if group == 1
summ var if group == 0
```

Check for Missing

```
// Count missing
count if var == .

// Identify observations with missing
list id var if var == .

// Missing by group
tab group, m
bysort group: count if var == .
```

Verify Merge Success

```
// After merge
tab _merge

// List unmatched from master
list id if _merge == 1

// List unmatched from using
list id if _merge == 2
```

```
// Check merged variable
summ merged_var if _merge == 3
// Should have valid values
```

Common Workflows

Clean Outcome Variable

```
// Step 1: Explore
codebook outcome
summ outcome, d
tab outcome, m

// Step 2: Identify issues
// - Missing values?
// - Outliers?
// - Correct range?

// Step 3: Create clean version
gen outcome_clean = outcome

// Step 4: Handle missing
replace outcome_clean = . if outcome == 99
replace outcome_clean = . if outcome < 0

// Step 5: Handle outliers (top-code)
replace outcome_clean = 1000000 ///
    if outcome > 1000000 & outcome != .

// Step 6: Verify
summ outcome_clean, d
tab outcome outcome_clean, m
```

Prepare Covariates

```
// Continuous variable
// - Check range
summ age, d
// - Handle missing
replace age = . if age == 99
// - Create squared term if needed
gen age_squared = age^2

// Categorical variable (numeric)
// - Check values
tab category, m
// - Create labeled version
label define cat_lbl 1 "A" 2 "B" 3 "C"
label values category cat_lbl

// Categorical variable (string)
// - Encode to numeric
encode string_var, gen(category_num)
// - Verify
tab string_var category_num

// Binary flag
// - Create 0/1
gen flag = 1 if condition & var != .
replace flag = 0 if !condition
// - Verify
summ flag
tab flag, m
```

Complete Analysis Example

```
// Research Q: County income effect on
// hospitalization costs?

// Step 1: Load and check data
use "hospital_data.dta", clear
describe
summ totalcosts, d

// Step 2: Clean outcome
gen cost_clean = totalcosts
replace cost_clean = 1000000 ///
    if totalcosts > 1000000 & totalcosts != .

// Step 3: Clean covariates
encode agegroup, gen(age_num)
gen ED_flag = (ed_indicator == "Y") ///
    if ed_indicator != ""

// Step 4: Merge county data
rename county County_Name
merge m:1 County_Name ///
    using "county_income.dta"
keep if _merge == 1 | _merge == 3
drop _merge

// Step 5: Check merged data
summ County_Income, d
// Report min and max

// Step 6: Run regression
reg cost_clean County_Income ///
    lengthofstay i.age_num i.ED_flag, r

// Step 7: Interpret
// "Controlling for length of stay, age,
// and ED status, each $1000 increase in
// county income is associated with
// $XX increase in hospital costs.
// This is statistically significant
// (p<0.05)."
```

Additional Commands

Duplicates

```
// Check for duplicates
duplicates report stay_id

// List duplicate examples
duplicates examples stay_id
```

```
// Tag duplicates (create flag)
duplicates tag stay_id, gen(dup_flag)

// Drop exact duplicates (keep one)
duplicates drop stay_id, force

// Keep first observation per group
bysort subject_id: keep if _n == 1
```

Bysort Techniques

```
// _n = observation number within group
// _N = total observations in group

// Count admissions per patient
bysort subject_id: gen n_admissions = _N

// Sequence number within group
bysort subject_id: gen admission_seq = _n

// Keep first observation per group
bysort subject_id: keep if _n == 1

// Keep last observation per group
bysort subject_id: keep if _n == _N

// Flag first observation
bysort county_name: gen first = (_n == 1)
```

Date/Time Handling

```
// String to date
gen date_var = date(date_string, "MDY")
format date_var %td

// String to datetime
gen datetime_var = clock(dt_string, ///
    "YMD hms")
format datetime_var %tc

// Date calculations (days)
gen los = discharge_date - admit_date

// Extract components
gen year_num = year(date_var)
gen month_num = month(date_var)
gen day_num = day(date_var)

// Common date formats:
// "MDY" - Month/Day/Year (1/15/2023)
// "YMD" - Year/Month/Day (2023-01-15)
// "hms" - hours:minutes:seconds
```

Test (Hypothesis Testing)

```
// After regression
reg outcome x1 x2 i.category

// Test single coefficient
test x1

// Test multiple coefficients jointly
test x1 x2

// Test categorical variable
test 2.category 3.category

// Test interaction term
reg y i.sex#c.age
test sex#c.age
```

Tabstat

```
// Basic usage
tabstat variable, stat(n mean sd)

// By groups
tabstat bmi, by(age_group) ///
    stat(n mean sd min p50 max)

// Multiple variables
tabstat bmi weight height, ///
    by(gender) stat(n mean sd)

// Format output
tabstat var, by(group) stat(n mean sd) ///
    format(%9.2f)
```

Preserve & Restore

```
// Save current data state
preserve

// Make temporary changes
keep if age >= 65
collapse (mean) var, by(state)

// Restore original data
restore

// Use for temporary exploration
// without losing original dataset
```

Collapse

```
// WARNING: Permanently changes data!
// Save original first!
save original_data, replace

// Calculate mean by group
collapse (mean) flu_shot, by(month_year)

// Multiple statistics
collapse (mean) mean_los=los ///
    (sd) sd_los=los ///
    (count) n=los, by(provider_group)

// Restore original
use original_data, clear
```

Key Reminders & Tips
Common Mistakes

⚠ Warning

Top Mistakes: Missing values (& var != .) — substr() starts at 1 — Use c. for continuous — Always tab _merge — Binary: check summ shows 0-1

Statistical Significance vs Practical

```
// Example: p=0.007, diff=3 minutes
// Statistical: YES (p<0.05)
// Practical: MAYBE
// - 3 min might be too small
// - But >10% relative difference
// - Large sample = "overpowered"
//   (can detect tiny differences)

// Always discuss BOTH in interpretation
```

Regression Interpretation

```
// Linear regression coefficient:
// "Each 1-unit increase in X is
// associated with beta-unit change in Y,
// controlling for other variables."

// Logistic regression OR:
// "Each 1-unit increase in X is
// associated with ORx change in the
// odds of Y, controlling for others."

// Binary outcome + linear regression:
// "Each 1-unit increase in X is
// associated with beta percentage point
// change in probability of Y."
```

When to Use What

- Data Types:**
- Cross-sectional survey: Prevalence, associations
 - Longitudinal survey: Within-person changes, causality
 - Claims data: Utilization, costs, readmission
 - EHR data: Clinical details, single system
- Merge Types:**
- M:1: Individual records + area-level data
 - 1:M: Visits + medications per visit
 - 1:1: Same IDs in both datasets

- Regression Types:**
- Linear: Continuous outcome
 - Linear: Binary outcome (percentage points)
 - Logistic: Binary outcome (odds ratios)
 - Fixed effects: Panel data, within-person

Cheat Sheet Usage Tips

- 💡 Tip**
- During the exam:**
1. Start with exploration (summ, tab, describe)
 2. Create clean versions of variables
 3. Verify each step before moving on
 4. Check merge with tab _merge
 5. Verify binary variables with summ
 6. Write complete interpretations

Quick Reference

| Task | Command |
|---------------|--|
| Load data | use "file.dta", clear |
| Import CSV | import delimited "file.csv" |
| Summary stats | summ var, d |
| Frequency | tab var, m |
| Create var | gen newvar = expr |
| Binary flag | gen flag = (condition) |
| Top-code | replace var = cap if var <i>i</i> cap & var != . |
| Encode string | encode str, gen(num) |
| String to num | destring str, gen(num) force |
| M:1 merge | merge m:1 id using "file.dta" |
| Check merge | tab _merge |
| Linear reg | reg y x1 x2, r |
| With category | reg y x1 i.cat |
| Interaction | reg y i.cat1##c.cont |
| Logistic | logistic binary_y x1 x2 |
| Verify binary | summ flag (should be 0-1) |
| Cross-check | tab oldvar newvar |