KING'S *College* LONDON

7QQMM818 Statistical Software for Finance

with Application to Dissertation

Haris Karagiannakis
haris.karagiannakis@kcl.ac.uk

Module Description

- This module focuses on the use of Stata for empirical analysis and data analytics in finance and economics. First, it teaches students how to read data in the software, how to manipulate these data (using sort, merge, implement wide-to-long transformations etc). Then, it teaches how to use Stata to conduct econometric analysis and interpret the results, including plain OLS regressions, standard time-series methods and ARCH/GARCH models, and finally (if time permits) moves on to techniques for panel data. Emphasis is placed on practical examples. We will also discuss important data sources for financial and economic data (including certain WRDS products, Yahoo Finance, Google Finance, the Fed of St. Louis’ FRED database).

There will be no graded assessment for this course. Occasionally we will do in-class exercises.

Module Description

After completing the module, you will:

- Have a working knowledge of Stata.
- Be able to import and manipulate data in Stata.
- Be able to extract summary statistics and conduct a variety of econometric analyses.
- Be able to interpret econometric output.
- Be able to create your own scripts and loops in Stata to address applied research questions that require more complicated procedures.
- Be familiar with numerous data sources.

Course Outline

Applications

We will see a mixture of **Stata,** financial & economic **datasets,** and **econometric applications**:

- **Time-series** dataset example, using the **CRSP** stock return data & the **FF** risk-factors.
  - Application 1: Asset Pricing Models
    - Standard CAPM Model
    - Fama-French (FF) 5-factor Model
  - Application 2: Conditional Mean Models (Univariate)
    - Autoregressive Integrated Moving Average (ARIMA) models
  - Application 3: Conditional Variance Models (Univariate)
    - Autoregressive Conditional Heteroskedasticity (ARCH) Model and extensions
- **Panel** dataset examples, using Wages and Property Prices
  - Application 1: Determinants of real wages using the US National Longitudinal Survey
  - Application 2: Hedonic Price Regression (and Hedonic Price Indices)

Course Outline

Stata Commands

We will explore numerous commands and procedures, including:

- cd, help, use, save, import, export, clear, cls, list browse, describe, format, quietly
- copy, unzipfile, erase, ssc install, view browse, merge , append, set obs
- tabulate, summarize, drop, keep, generate, replace, egen, rename, preserve/restore, reshape
- corr, pwcorr
- regress, arima, arch, xtreg, predict, estimates store, test, testparm
- tsset, xtset, and time-series operators (L.var, F.var, D.var)
- return, ereturn
- estimates table, estout, outreg2, putexcel
- Plotting commands: line, histogram, scatter
- In-line if-statement, prefixes (xi, bysort, rolling), Loops (foreach, forvalues, while) and local’s
- Various functions for string, numerical data and SIF datetimes: Nsplit, tostring

Teaching Arrangements

- Lectures will take place online, every Friday 12:00-15:00, with a 30-minute break

Lecture 1: 10th March

Lecture 2: 17th March

Lecture 3: 24th March

Lecture 4: 31st March

Lecture 5: 7th April

Lecture 6: 14th April

Lecture 7: 21st April

Lecture 8: 28th April

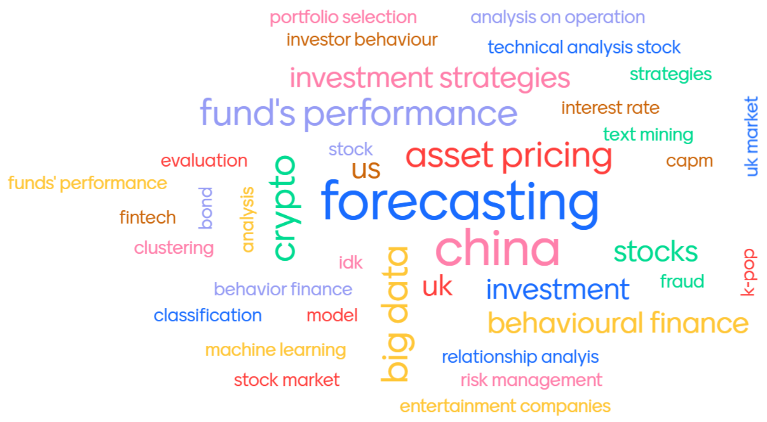
Survey to get to know you

- Please head to www.menti.com and answer the short survey.

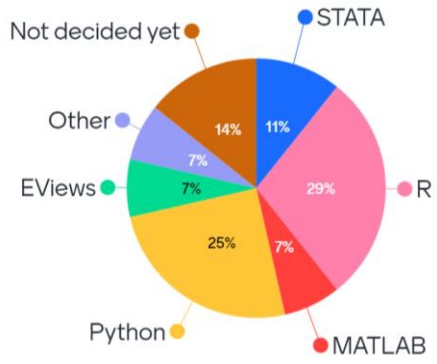
Survey to get to know you

- Please head to www.menti.com and answer the short survey. Some results from last year:

Dissertation topic(s) you are interested in



Which statistical package(s) / software you will use for your dissertation?



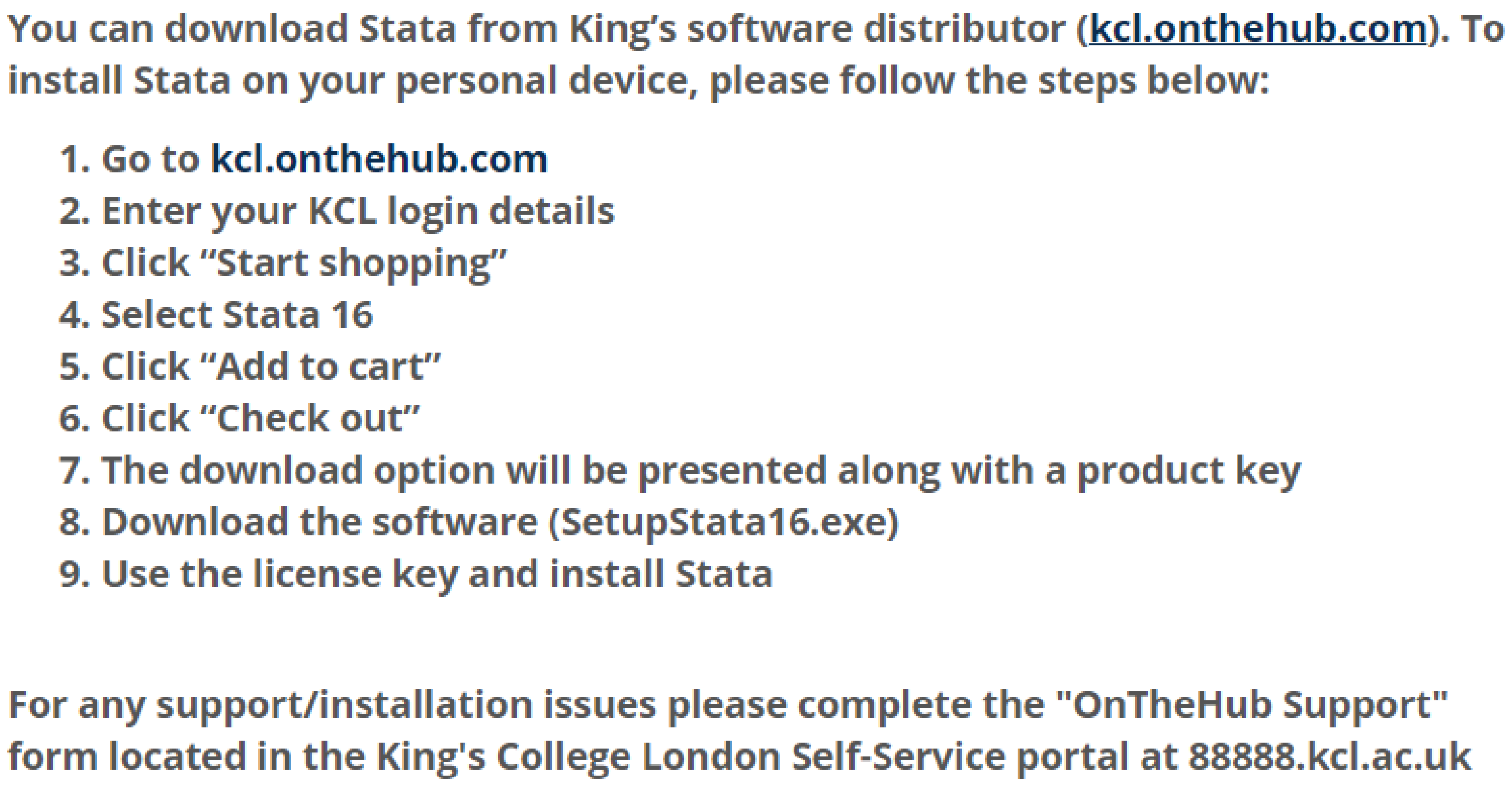
How to pronounce Stata?

- Stata is an invented word. Some pronounce it with a long a as in day (Stay-ta); some pronounce it with a short a as in flat (Sta-ta); and some pronounce it with a long a as in ah (Stah-ta). The correct English pronunciation must remain a mystery, except that personnel of StataCorp use the first of these.

Source: https://www.statalist.org/forums/faq#pronounce

Where can we download Stata?

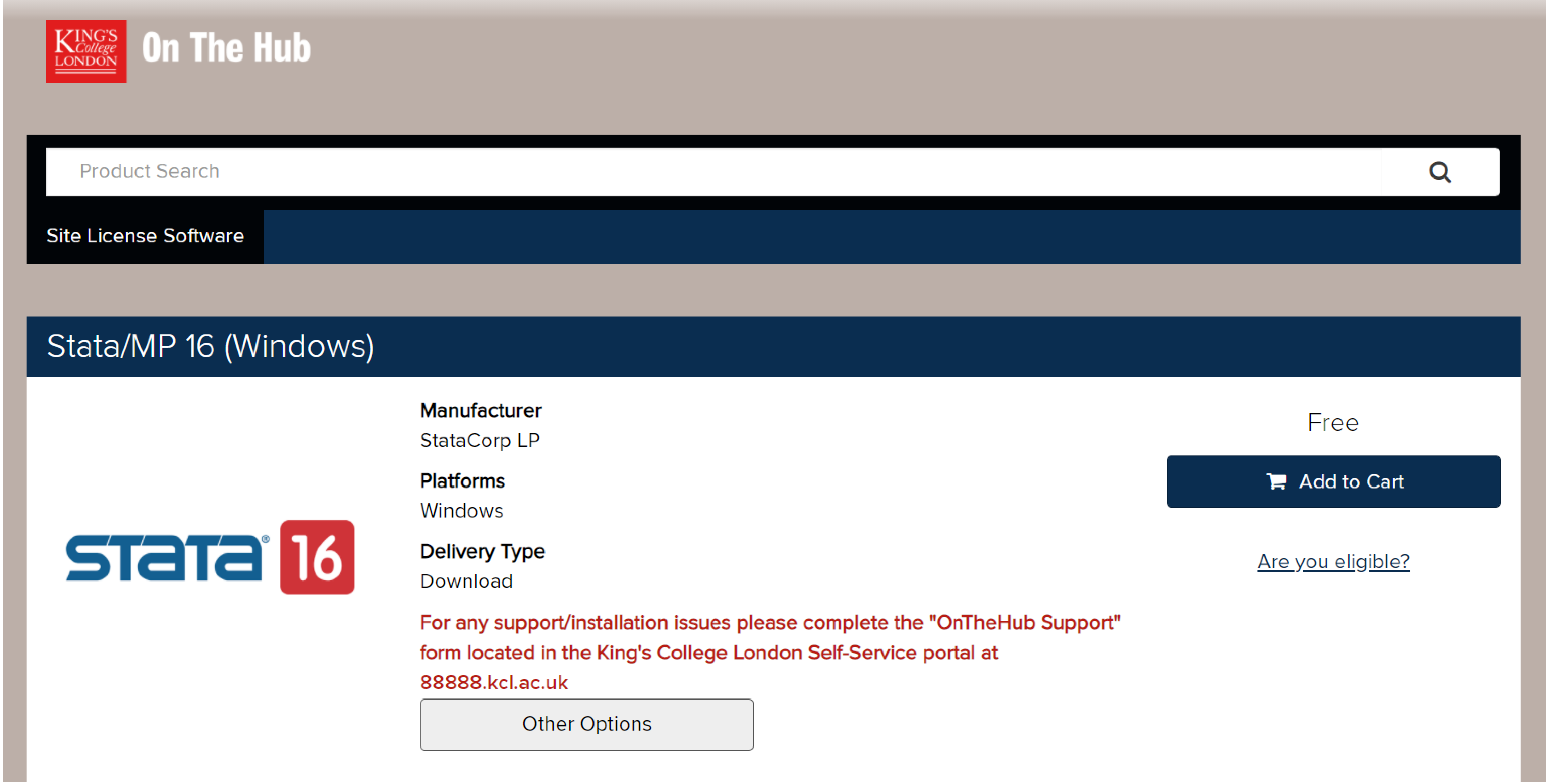
- KBS Software 21-22 in KEATS → https://keats.kcl.ac.uk/enrol/index.php?id=81667



You can download Stata from King's software distributor (kcl.onthehub.com). To install Stata on your personal device, please follow the steps below:

1. Go to **kcl.onthehub.com**
2. Enter your KCL login details
3. Click "Start shopping"
4. Select Stata 16
5. Click "Add to cart"
6. Click "Check out"
7. The download option will be presented along with a product key
8. Download the software (SetupStata16.exe)
9. Use the license key and install Stata

For any support/installation issues please complete the "OnTheHub Support" form located in the King's College London Self-Service portal at 88888.kcl.ac.uk



Stata/MP 16 (Windows)

Manufacturer
StataCorp LP

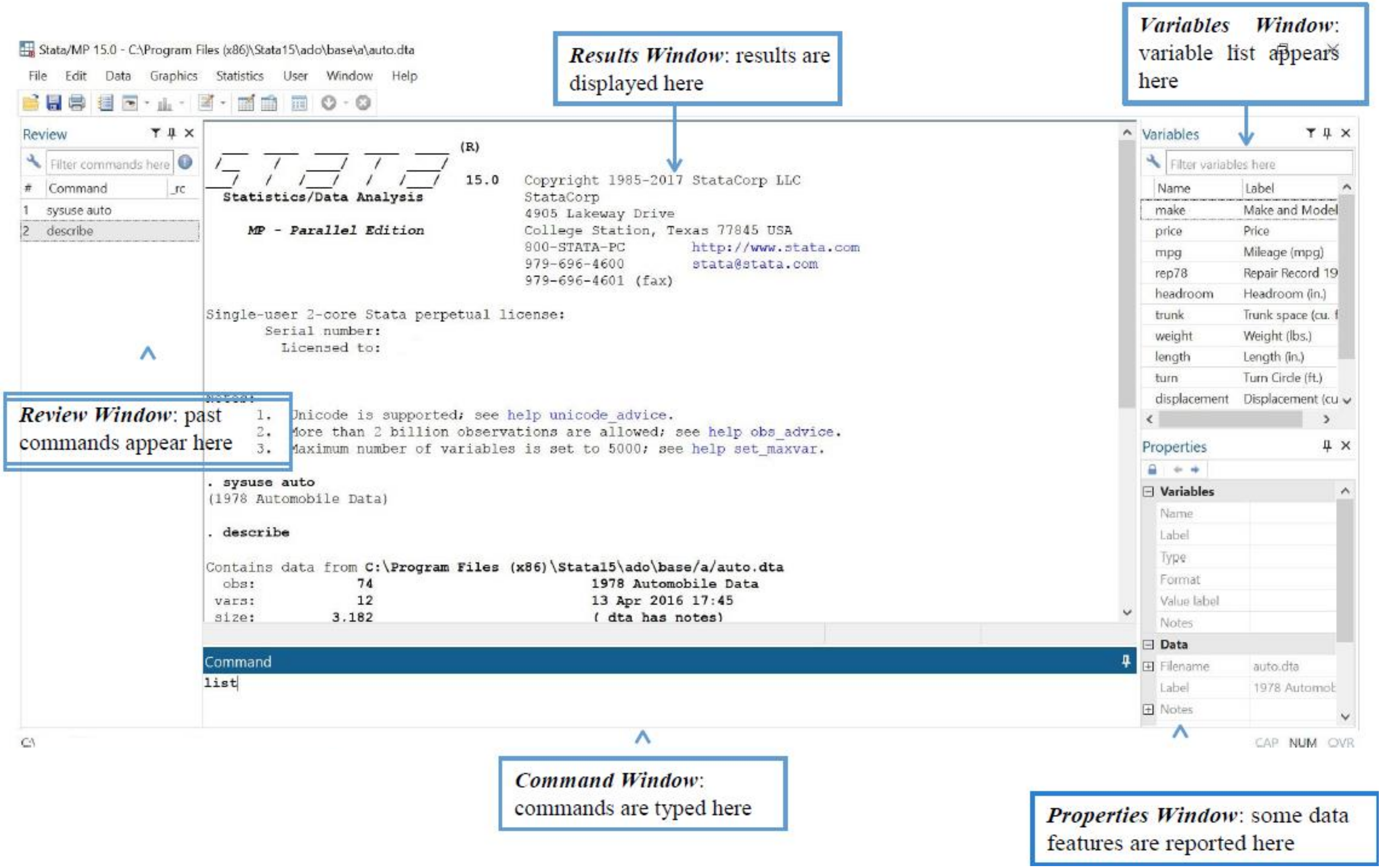
Platforms
Windows

Delivery Type
Download

Free

For any support/installation issues please complete the "OnTheHub Support" form located in the King's College London Self-Service portal at 88888.kcl.ac.uk

Stata’s User Interface



Results Window: results are displayed here

Variables Window: variable list appears here

Review Window: past commands appear here

Command Window: commands are typed here

Properties Window: some data features are reported here

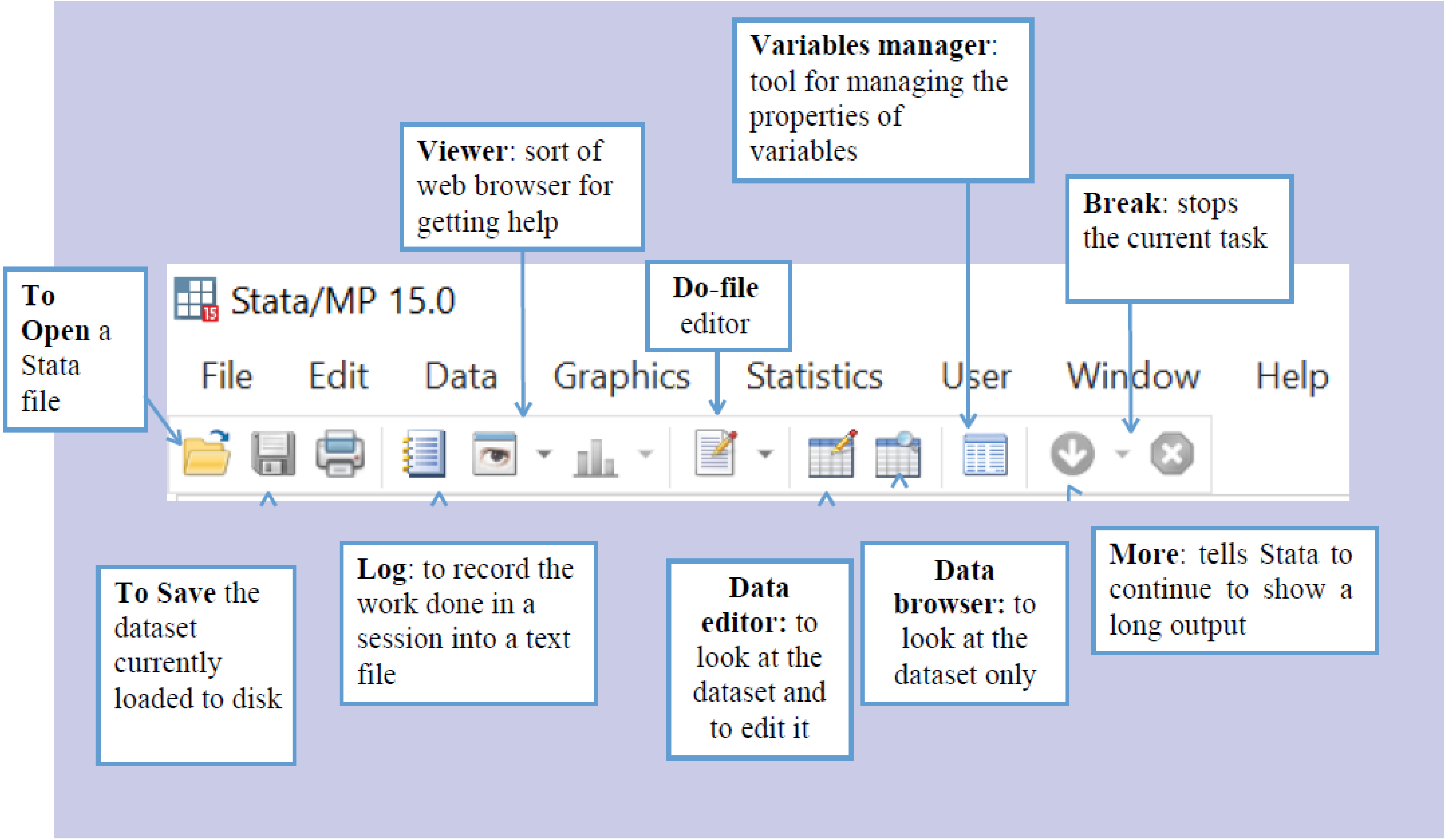
#	Command	_rc
1	sysuse auto	
2	describe	

```
                                        (R)
 Statistics/Data Analysis          15.0   Copyright 1985-2017 StataCorp LLC
                                          StataCorp
                                          4905 Lakeway Drive
   MP - Parallel Edition                  College Station, Texas 77845 USA
                                          800-STATA-PC        http://www.stata.com
                                          979-696-4600        stata@stata.com
                                          979-696-4601 (fax)

Single-user 2-core Stata perpetual license:
       Serial number:
         Licensed to:


Notes:
      1.  Unicode is supported; see help unicode_advice.
      2.  More than 2 billion observations are allowed; see help obs_advice.
      3.  Maximum number of variables is set to 5000; see help set_maxvar.

. sysuse auto
(1978 Automobile Data)

. describe

Contains data from C:\Program Files (x86)\Stata15\ado\base/a/auto.dta
  obs:            74                          1978 Automobile Data
 vars:            12                          13 Apr 2016 17:45
 size:         3,182                          ( dta has notes)
```

```
list
```

Name	Label
make	Make and Model
price	Price
mpg	Mileage (mpg)
rep78	Repair Record 19
headroom	Headroom (in.)
trunk	Trunk space (cu. f
weight	Weight (lbs.)
length	Length (in.)
turn	Turn Circle (ft.)
displacement	Displacement (cu

Variables	
Name	
Label	
Type	
Format	
Value label	
Notes	
Data	
Filename	auto.dta
Label	1978 Automob
Notes	



To Open a Stata file

Viewer: sort of web browser for getting help

Do-file editor

Variables manager: tool for managing the properties of variables

Break: stops the current task

To Save the dataset currently loaded to disk

Log: to record the work done in a session into a text file

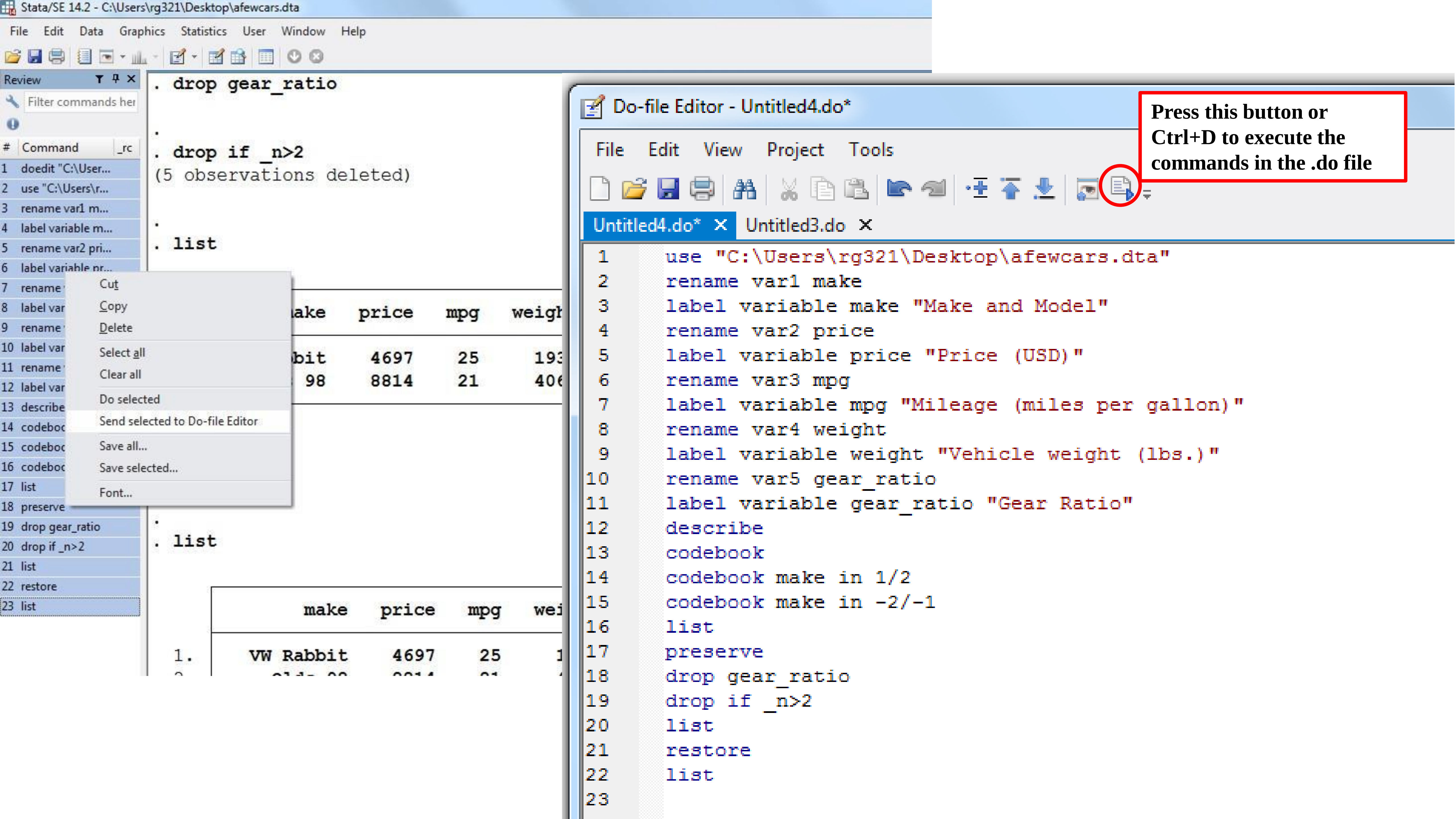
Data editor: to look at the dataset and to edit it

Data browser: to look at the dataset only

More: tells Stata to continue to show a long output

Stata’s Interface

- *Data*, *Graphics*, *Statistics* tabs give you access to Stata's GUI (i.e. dialog boxes), replacing the command line
- Create a **.smcl file** (log file) keeping track of your output (i.e. anything in the Results window)
  > File > Log > Begin
- Start a new **.do file** where you can keep your commands to re-run later
  > 'New Do-file editor' button on the toolbox
  The Do-file is known as 'script' in other software
- **Data editor** (command: 'browse y x1 x2'   OR   'edit y x1 x2’)
- Change each variable's properties (Name, Format, Label)
  > Data > Variables Manager
  > Same functionality also via the 'Properties window' (NB: Unlock using the little lock)
- Shortcut for running selected lines in Do-file: **Ctrl+D**
- Files with a **.dta** extension are Stata datasets.



```
1  doedit "C:\User...
2  use "C:\Users\r...
3  rename var1 m...
4  label variable m...
5  rename var2 pri...
6  label variable pr...
7  rename
8  label var
9  rename
10 label var
11 rename
12 label var
13 describe
14 codeboo
15 codeboo
16 codeboo
17 list
18 preserve
19 drop gear_ratio
20 drop if _n>2
21 list
22 restore
23 list
```

```
. drop gear_ratio

.
. drop if _n>2
(5 observations deleted)

.
. list

    ake    price   mpg   weigh
    bit     4697    25     193
     98     8814    21     406

.
. list

          make    price   mpg   wei

  1.  VW Rabbit    4697    25     1
```

```
use "C:\Users\rg321\Desktop\afewcars.dta"
rename var1 make
label variable make "Make and Model"
rename var2 price
label variable price "Price (USD)"
rename var3 mpg
label variable mpg "Mileage (miles per gallon)"
rename var4 weight
label variable weight "Vehicle weight (lbs.)"
rename var5 gear_ratio
label variable gear_ratio "Gear Ratio"
describe
codebook
codebook make in 1/2
codebook make in -2/-1
list
preserve
drop gear_ratio
drop if _n>2
list
restore
list
```

Press this button or Ctrl+D to execute the commands in the .do file

Commenting in Stata

- You can write comments inside the .do file to:
  - separate different parts/sections of your codes,
  - keep track of what each section, or specific commands are doing,
  - make your code more readable in general.
- Comments, in any programming/scripting language, are ignored when you are executing your script files.
- In Stata, commenting can be done in a few different ways:
  1. Double forward slash: //
     Use at the end of the command to comment-out the rest of that line
  2. Asterisk: *
     Can only be used at the beginning of each line
  3. A forward slash and an asterisk: /* comment */
     Creates a block within which everything is a comment (for writing more conveniently multi-line comments).

Anything in green inside a .do file corresponds to a comment.

Syntax of Stata Commands

```
[prefix:] command [varlist] [=exp] [if] [in range] [weight] [,options]
```

[=exp] specifies the value to be assigned to a variable and is most often used with the commands ***generate*** and ***replace***

The **[if]** qualifier uses a ***logical expression*** to determine which observations to use
If the expression is true, the observation is **used** in the command, otherwise, it is **skipped**

[varlist]: list of variable names. **Most commands** do not require that you explicitly type a ***varlist***.

Thus if **no *varlist*** appears, these commands are **run** on **all** the **variables** in the dataset (see the previous examples of describe and codebook)

Some commands take a [***varname***] rather than a [***varlist]***
(i.e. they can be run only on one variable at a time). A ***varname*** refers to exactly one variable

[prefix:] examples:
xi, bysort

command examples:
generate, summarize

Datasets

CRSP, Compustat, Fama-French and Other Useful Resources

CRSP U.S. Stock database

The CRSP data is accessible through Wharton Research Data Services (WRDS).

The CRSP U.S. Stock database contains end-of-day and month-end prices on primary listings for the NYSE, NYSE MKT (AMEX), NASDAQ, and Arca exchanges.

- NYSE: Series begin on December 31, 1925.
- NYSE MKT (AMEX): Series begin on July 2, 1962.
- NASDAQ: Series begin on December 14, 1972.
- Arca: Series begin on March 8, 2006.

The CRSP U.S. Stock database only includes securities for international (i.e. non-U.S.) companies if they are **ADRs**, **cross-listed,** or traded on the **major stock exchanges** mentioned above. **Bonds**, **preferred shares**, and **shares traded in regional exchanges or OTC**, are not included in the CRSP U.S. Stock database.

CRSP U.S. Stock database

Share Code (variable: SHRCD) in CRSP database is a 2-digit numerical variable that classifies all traded securities according to the issue’s type. The **first digit** gives the primary security class, while the **second digit** gives further information on the security’s type.

SHRCD Code digits		**Definition**
First digit	**1**	Ordinary Common Shares
	2	Certificates
	3	American Depository Receipts (ADR’s)
	4	Shares of Beneficial Interest (SBI’s)
	7	Units (Depository Units, Units of Beneficial Interest, Units of Limited Partnership Interest, Depository Receipts, etc.)
Second digit	**0**	Securities which have not been further defined.
	1	Securities which need not be further defined.
	2	Companies incorporated outside the US
	3	Americus Trust Components (Primes and Scores).
	4	Closed-end funds.
	5	Closed-end fund companies incorporated outside the US
	8	Real Estate Investment Trusts (REIT’s)

CRSP U.S. Stock database

SHRCD Code	Market Value (bn $)	# of Securi ties	Share of Total MV	Share of Securities Number	Examples of Securities within each SHRCD Group (Snapshot of trading day 30/11/17)
11	$27,049	3624	73.2%	51.3%	Microsoft Corp (MSFT), Facebook Inc. (FB), Blackrock Inc. (BLK), Coca Cola Co (KO)
12	$3,212	522	8.7%	7.4%	Fiat Chrysler Automobiles NV (FCAU), Coca Cola European Partners Plc (CCE)
14	$171.9	265	0.5%	3.8%	Blackrock Corporate High Yield Fund (HYT), Blackrock Muniyield Quality Fund III (MYI)
15	$3.5	2	0.0%	0.0%	Central Fund Canada Ltd (CEF), ASA Gold & Precious Metals Ltd (ASA)
18	$995.7	182	2.7%	2.6%	Gladstone Land Corp (LAND), Columbia Property Trust Inc. (CXP)
21	$3.2	1	0.0%	0.0%	Texas Pacific Land Trust (TPL)
31	$1,394	334	3.8%	4.7%	BP Plc (BP), Barclays Plc (BCS), Toyota Motor (TM), Coca Cola FEMSA, SAB de CV (KOF)
41	$1.0	2	0.0%	0.0%	Innsuites Hospitality Trust (IHT), Compass Diversified Holdings (CODI)
44	$139.7	294	0.4%	4.2%	Blackrock Taxable Municipal Bond Trust (BBN), PIMCO Dynamic Credit & Mortgage Income Fund (PCI)
48	$145.5	39	0.4%	0.6%	Whitestone REIT (WSR), Corporate Office Properties Trust (OFC)
71	$409.4	131	1.1%	1.9%	Spectra Energy Partners LP (SEP), Terra Nitrogen Co LP (TNH)
72	$68.4	16	0.2%	0.2%	Brookfield Property Partners LP (BPY), Granite Real Estate Investment Trust (GRPU)
73	$3,334.5	1870	9.0%	26.5%	SPDR S&P 500 ETF Trust (SPY), iShares Core S&P 500 ETF Trust (IVV), Vanguard Total Stock Market ETF Index Fund (VTI)
74	$4.2	25	0.0%	0.4%	Teucrium Commodity Trust (CANE), United States Natural Gas FD LP (UNG)
75	$3.2	3	0.0%	0.0%	Sprott Physical Silver Trust (PSLV), Sprott Physical Platinum and Palladium Trust (SPPP)
Total	$36,937	7060	100%	100%	

CRSP U.S. Stock database

Following Fama and French (1993; 2015) the **market return** is the value-weighted return of all CRSP firms incorporated in the U.S. and listed on the NYSE, AMEX, or NASDAQ that have a CRSP share code (SHRCD) of 10 or 11, while the risk-free rate is the one-month Treasury bill rate (from Ibbotson Associates).

Exchange or Market Segment	Market Value	# of Securities	Share of Total MV	Share of Total Securities Number
NYSE	$21.91	2,491	62.0%	35.3%
AMEX	$0.12	307	0.3%	4.4%
NASDAQ	$10.54	2,935	29.8%	41.6%
ARCA	$2.76	1,323	7.8%	18.8%
Total	$35.32	7,056	100%	100%
Total (ex. ARCA)	$32.56	5,733	92.2%	81.3%
S&P 500	$23.49	505	66.5%	7.2%

Notes: Data as of 30/11/17. Due to CRSP's reporting scheme, market value (MV) figures exclude ADR securities while securities' number also counts ADR's. Market value in trillion (tn) U.S. dollars.
Source: CRSP file on ‘S&P 500 Indexes/Market Cap. - Daily’ for all but the S&P 500 statistics, and CRSP file on ‘Stock File Indexes/Index File on S&P 500’ for the Standard & Poor's 500 Composite Index.

Stock & Company Identifiers

CRSP

CRSP database

Company identifier

- PERMCO: permanent company/issuer identifier.
- Company Name

Security identifier

- PERMNO: permanent security/issue identifier. All securities associated with a single company (regardless of changes in name) will have the same PERMCO. Therefore, multiple PERMNOs can be associated with a single PERMCO code.
- Historical and current CUSIP:
  - 8-digit alphanumeric security/issue identifier; first 6 characters identify the **issuer** (i.e. company/government agency), next 2 identify the **issue** (equity/debt)
  - Within the CRSP database, historical CUSIP (variable: **ncusip**) is referred to as CUSIP, while current CUSIP (variable: **cusip**), is referred to as CUSIP Header.
- Ticker

Stock & Company Identifiers

Compustat

Compustat database

Company identifier

- GVKEY: permanent company/issuer identifier.
- Company Name

Security identifier

- Current CUSIP:
  - 9-digit alphanumeric security/issue identifier;
    The format is the same as CRSP CUSIP, but 9th character is used for accuracy checks.
  - Compustat only reports the latest (variable: cusip) CUSIP for the respective company's security.
- Ticker

Stock & Company Identifiers

- CRSP’s **PERMNO security identifier, PERMCO company identifier** and Compustat’s **GVKEY company identifier** are permanent identifiers, meaning that they do not change over time for a particular security and company, respectively.
- On the other hand, **CUSIP** and **ticker** are not permanent security identifiers. CUSIP may change following non-fundamental events such as when the name or the capital structure of a company changes, or when there are splits. Furthermore, regarding retired stock tickers, those are often reused after an issue ceases trading. Unlike ticker though, CUSIP is **not** a **reusable** security identifier.

Combining CRSP & Compustat

- Assume we would like to combine (annual/quarterly) company financial statement data from Compustat to (monthly/daily) security data from CRSP.
- There are two methods:
  1. Using the common (security) identifier found in both databases, namely CUSIP.
  2. Using another WRDS dataset known as: CRSP/Compustat Merged Database (CCM).
- Resources:

Comparison of the 2 methods:

https://wrds-www.wharton.upenn.edu/pages/support/applications/linking-databases/linking-crsp-and-compustat/

https://wrds-www.wharton.upenn.edu/pages/support/data-overview/ccm-overview-crspcompustat-merged-database/

SAS Code for method 1:

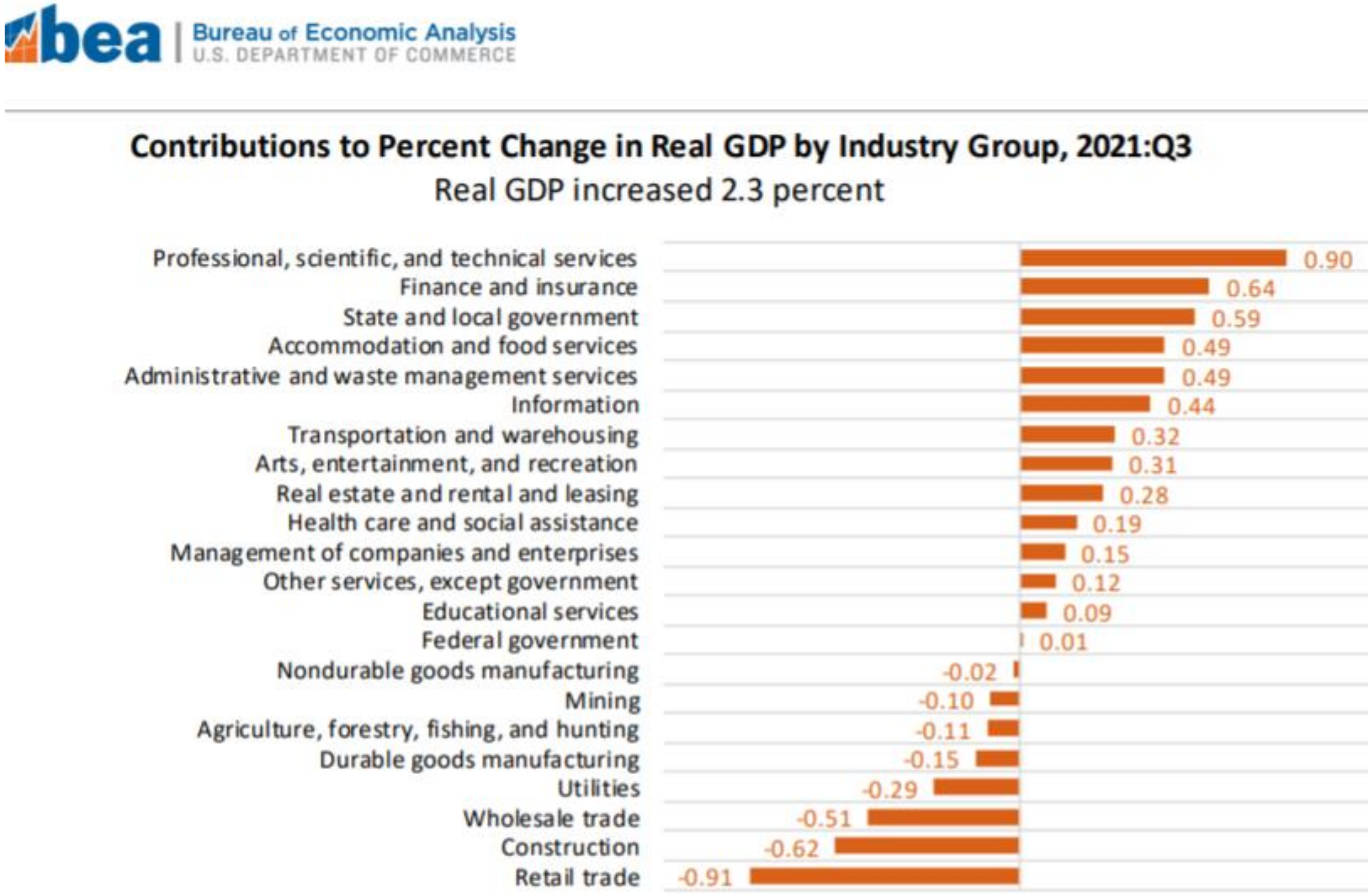
https://wrds-www.wharton.upenn.edu/pages/support/sample-programs/crsp/program-merge-crsp-and-compustat-using-cusip/

Industrial Classifications

- Various industrial classification systems exist to taxonomize industries. These systems are primarily used by government agencies to categorize companies by the nature of their business for the purposes of collecting and presenting a large range of statistical data according to economic activities (e.g. disaggregated production or employment data, and national accounts – GVA's).
- For example, at the lowest disaggregation level an industrial classification system would distinguish between: (1) Industry, (2) Retail, (3) Services, (4) Construction etc.
- Further dis-aggregations, could separate the Industrial sector further into: (1.i) Car Manufacturing, (1.ii) Truck Manufacturing, (1.iii) Clothes Manufacturing and so on.
- The most widely used Industrial Classification systems are:

1. SIC
2. NAICS
3. NACE
4. GICS
5. Fama-French

Industrial Classifications



bea | Bureau of Economic Analysis
U.S. DEPARTMENT OF COMMERCE

Contributions to Percent Change in Real GDP by Industry Group, 2021:Q3

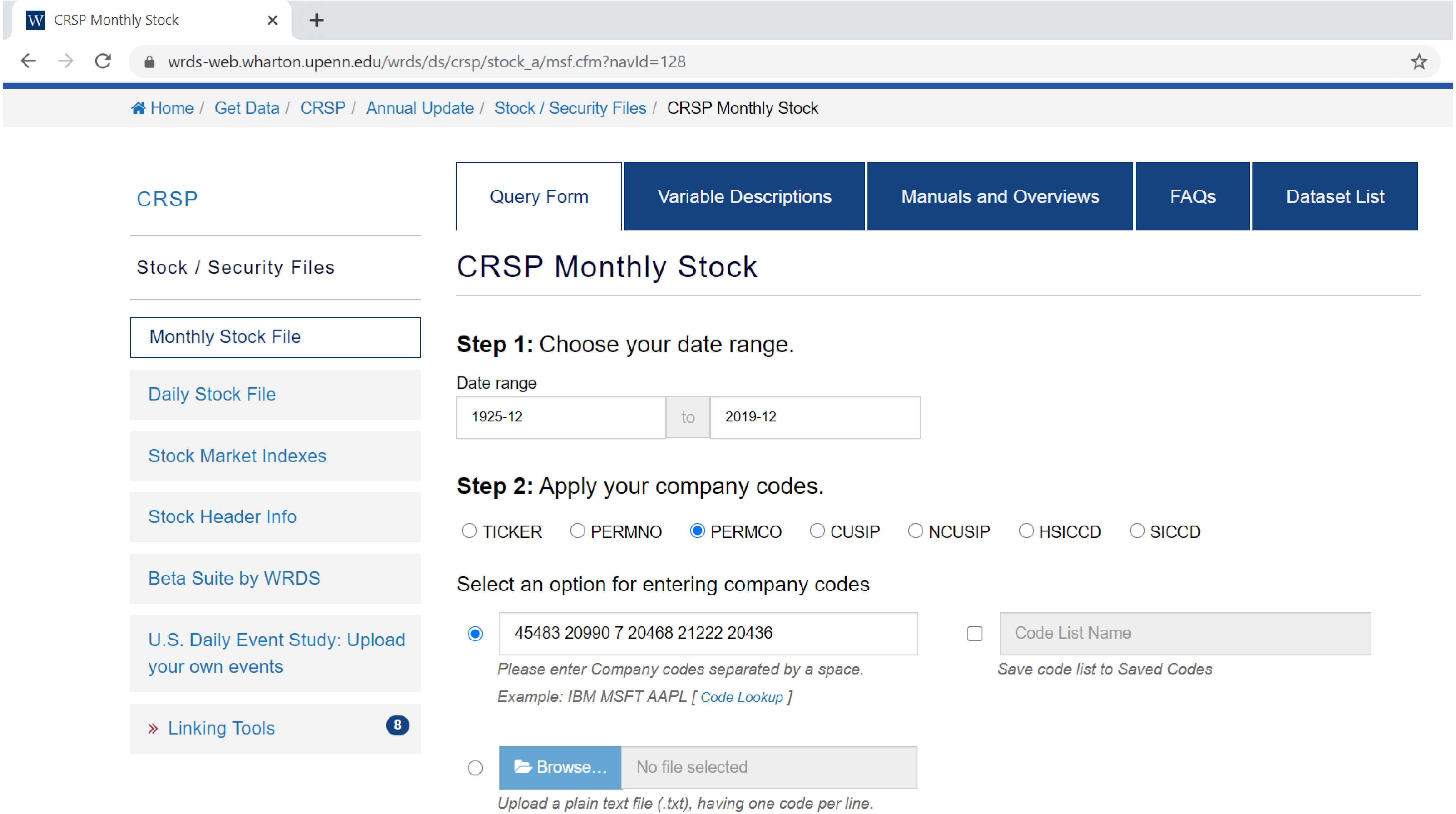
Real GDP increased 2.3 percent

Industrial Classifications

- If you find multiple classification schemes in your data sources (e.g. SIC and NAICS), it is always best to consult the literature before deciding which one to use, because using different classifications, would usually yield different results.
  - Generally, I would suggest using NAICS instead of SIC, since the latter was replaced by the former to provide an updated classification scheme reflecting changes in the overall economic activity.

1^{st} Dataset
WRDS-CRSP Stock Return Data & Fama-French Factors

- Coverage:
  - December 1925 – December 2019,
  - U.S. equities: Apple (AAPL), Coca Cola (KO), Google's parent company Alphabet (GOOG, GOOGL), IBM (IBM), JP Morgan (CHL, CMB, JPM).
- Monthly stock returns, adjusted for stock splits and dividends, from Center for Research in Security Prices (CRSP).
- Factor portfolios and risk-free rate from Kenneth French's website.
- Market return is calculated based on the CRSP value-weighted market index which includes all non-ADR securities (CRSP share code 10 or 11) listed in NYSE, AMEX and NASDAQ exchanges.
- Excess returns calculated with respect to the one-month US Treasury bill rate.



CRSP Monthly Stock

Step 1: Choose your date range.

Date range

1925-12 to 2019-12

Step 2: Apply your company codes.

○ TICKER ○ PERMNO ● PERMCO ○ CUSIP ○ NCUSIP ○ HSICCD ○ SICCD

Select an option for entering company codes

● 45483 20990 7 20468 21222 20436 □ Code List Name

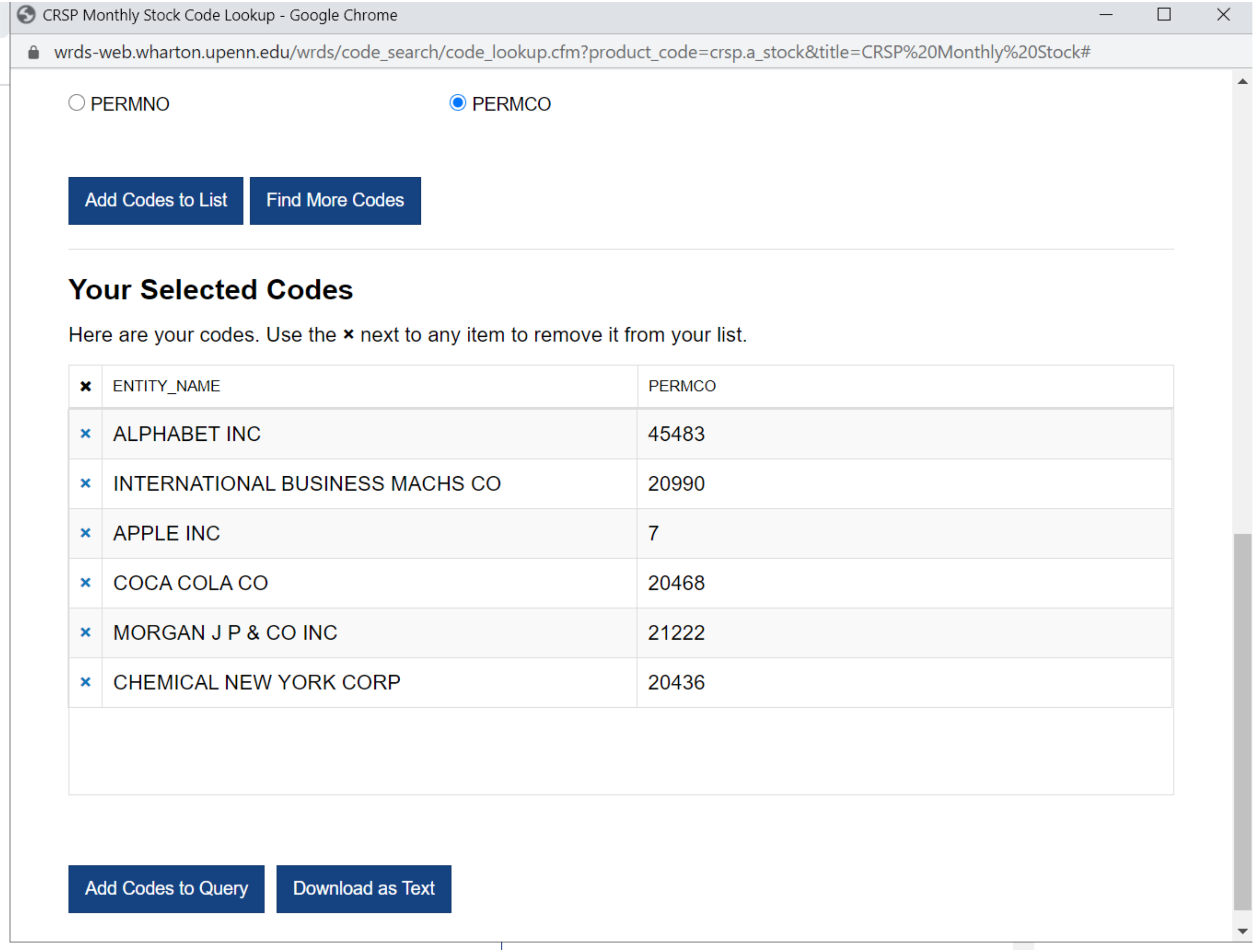
Please enter Company codes separated by a space.

Example: IBM MSFT AAPL [Code Lookup]

Save code list to Saved Codes

○ Browse... No file selected

Upload a plain text file (.txt), having one code per line.



CRSP Monthly Stock Code Lookup - Google Chrome
wrds-web.wharton.upenn.edu/wrds/code_search/code_lookup.cfm?product_code=crsp.a_stock&title=CRSP%20Monthly%20Stock#

○ PERMNO ● PERMCO

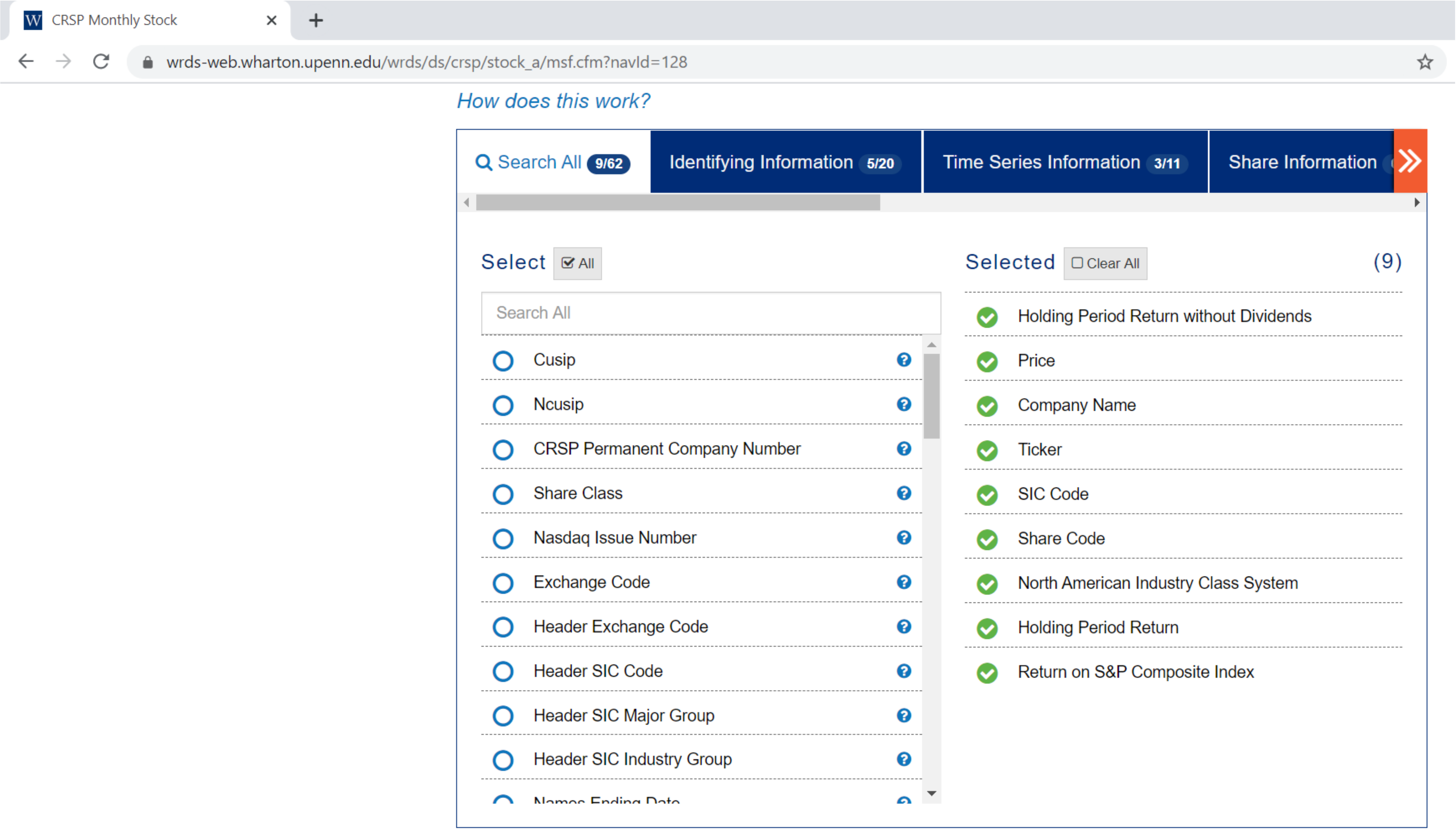
Add Codes to List
Find More Codes

Your Selected Codes

Here are your codes. Use the × next to any item to remove it from your list.

×	ENTITY_NAME	PERMCO
×	ALPHABET INC	45483
×	INTERNATIONAL BUSINESS MACHS CO	20990
×	APPLE INC	7
×	COCA COLA CO	20468
×	MORGAN J P & CO INC	21222
×	CHEMICAL NEW YORK CORP	20436

Add Codes to Query
Download as Text



CRSP Monthly Stock
wrds-web.wharton.upenn.edu/wrds/ds/crsp/stock_a/msf.cfm?navId=128

How does this work?

Search All 9/62
Identifying Information 5/20
Time Series Information 3/11
Share Information

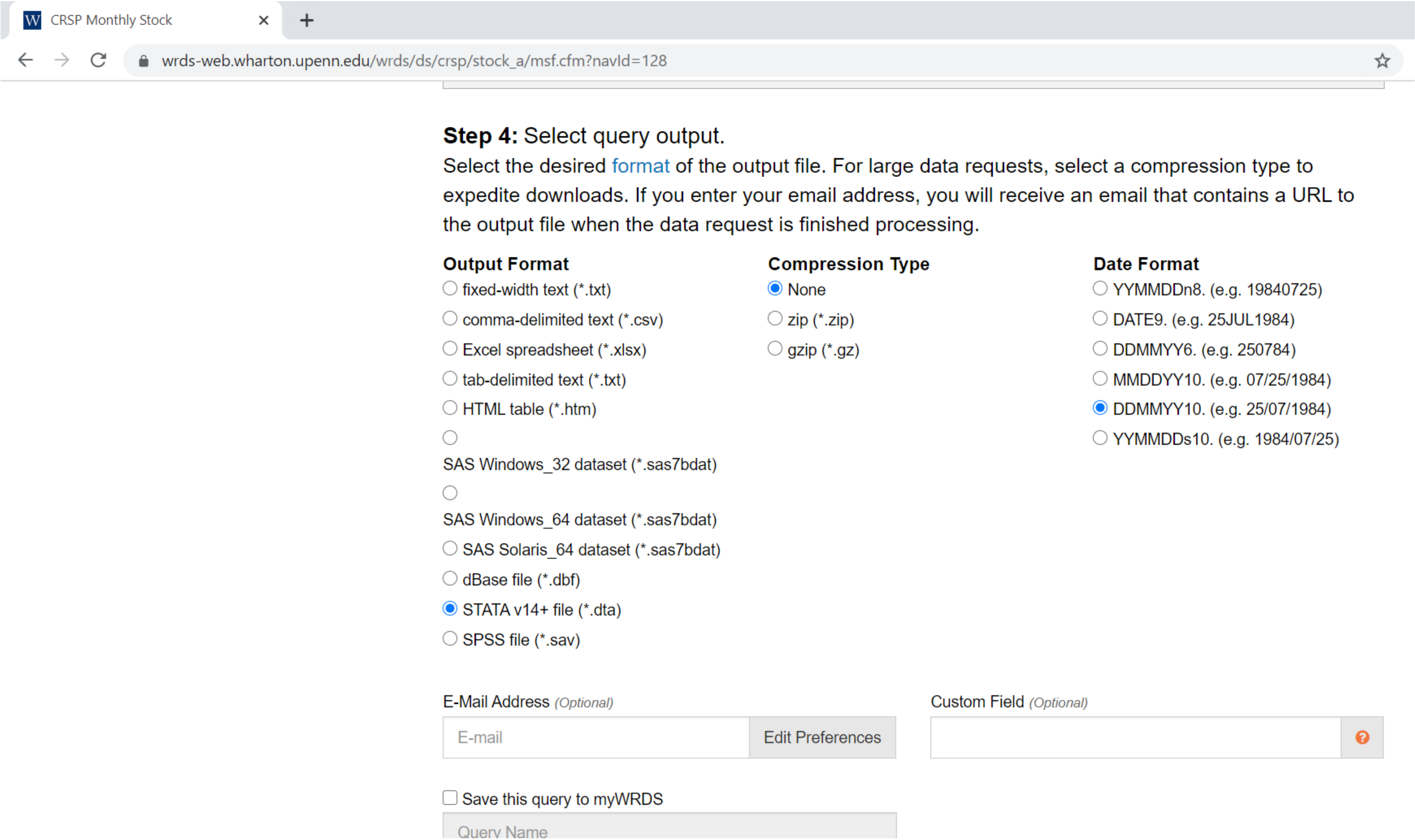
Select ☑ All

Search All

- ○ Cusip
- ○ Ncusip
- ○ CRSP Permanent Company Number
- ○ Share Class
- ○ Nasdaq Issue Number
- ○ Exchange Code
- ○ Header Exchange Code
- ○ Header SIC Code
- ○ Header SIC Major Group
- ○ Header SIC Industry Group
- ○ Names Ending Date

Selected □ Clear All (9)

- ✓ Holding Period Return without Dividends
- ✓ Price
- ✓ Company Name
- ✓ Ticker
- ✓ SIC Code
- ✓ Share Code
- ✓ North American Industry Class System
- ✓ Holding Period Return
- ✓ Return on S&P Composite Index



CRSP Monthly Stock

wrds-web.wharton.upenn.edu/wrds/ds/crsp/stock_a/msf.cfm?navId=128

Step 4: Select query output.

Select the desired format of the output file. For large data requests, select a compression type to expedite downloads. If you enter your email address, you will receive an email that contains a URL to the output file when the data request is finished processing.

Output Format

- ○ fixed-width text (*.txt)
- ○ comma-delimited text (*.csv)
- ○ Excel spreadsheet (*.xlsx)
- ○ tab-delimited text (*.txt)
- ○ HTML table (*.htm)
- ○ SAS Windows_32 dataset (*.sas7bdat)
- ○ SAS Windows_64 dataset (*.sas7bdat)
- ○ SAS Solaris_64 dataset (*.sas7bdat)
- ○ dBase file (*.dbf)
- ● STATA v14+ file (*.dta)
- ○ SPSS file (*.sav)

Compression Type

- ● None
- ○ zip (*.zip)
- ○ gzip (*.gz)

Date Format

- ○ YYMMDDn8. (e.g. 19840725)
- ○ DATE9. (e.g. 25JUL1984)
- ○ DDMMYY6. (e.g. 250784)
- ○ MMDDYY10. (e.g. 07/25/1984)
- ● DDMMYY10. (e.g. 25/07/1984)
- ○ YYMMDDs10. (e.g. 1984/07/25)

E-Mail Address *(Optional)*

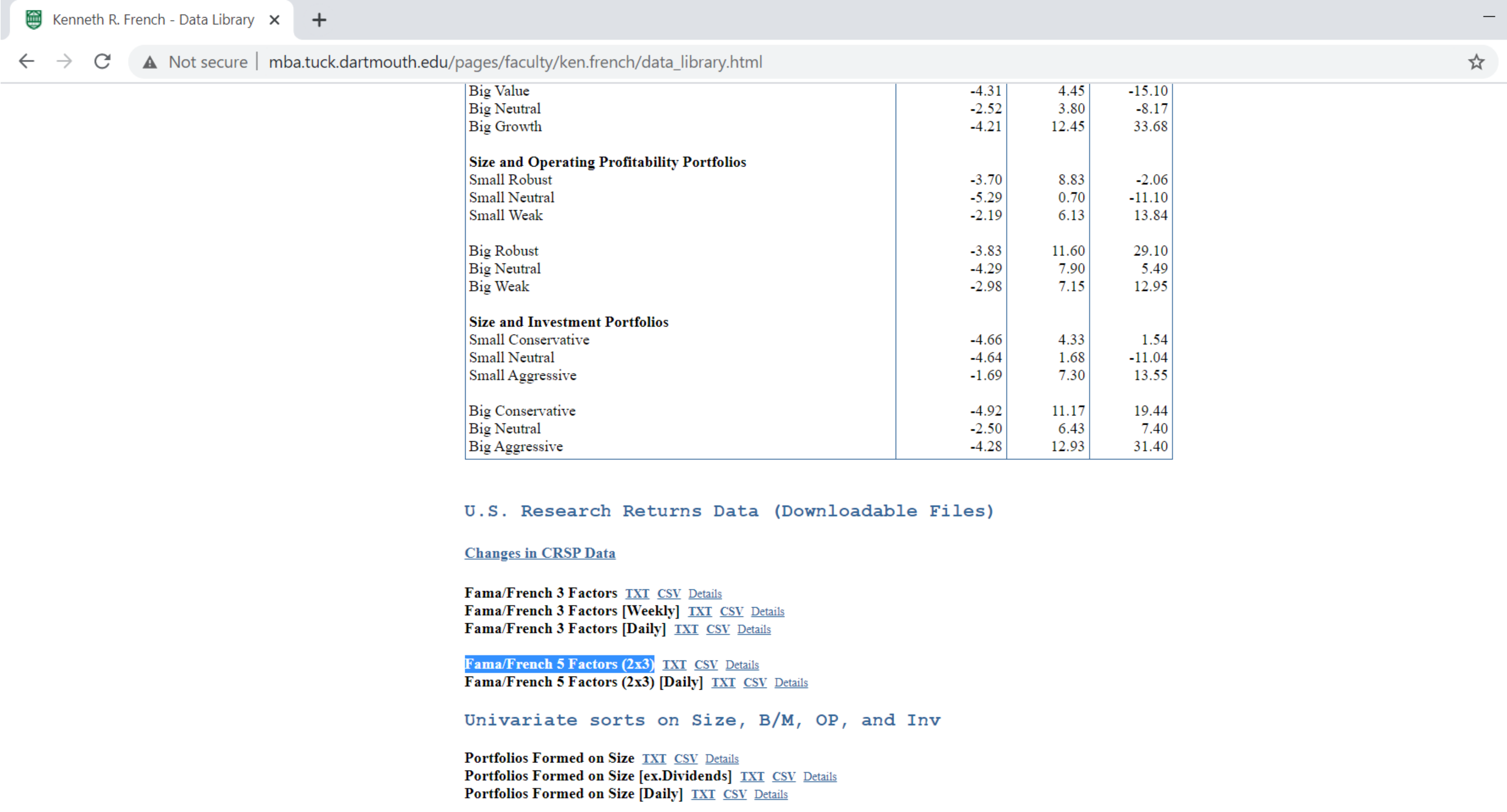
E-mail

Edit Preferences

Custom Field *(Optional)*

□ Save this query to myWRDS

Query Name


Kenneth R. French - Data Library
Not secure | mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html

Big Value	-4.31	4.45	-15.10
Big Neutral	-2.52	3.80	-8.17
Big Growth	-4.21	12.45	33.68
Size and Operating Profitability Portfolios			
Small Robust	-3.70	8.83	-2.06
Small Neutral	-5.29	0.70	-11.10
Small Weak	-2.19	6.13	13.84
Big Robust	-3.83	11.60	29.10
Big Neutral	-4.29	7.90	5.49
Big Weak	-2.98	7.15	12.95
Size and Investment Portfolios			
Small Conservative	-4.66	4.33	1.54
Small Neutral	-4.64	1.68	-11.04
Small Aggressive	-1.69	7.30	13.55
Big Conservative	-4.92	11.17	19.44
Big Neutral	-2.50	6.43	7.40
Big Aggressive	-4.28	12.93	31.40

U.S. Research Returns Data (Downloadable Files)

Changes in CRSP Data

Fama/French 3 Factors TXT CSV Details
Fama/French 3 Factors [Weekly] TXT CSV Details
Fama/French 3 Factors [Daily] TXT CSV Details

Fama/French 5 Factors (2x3) TXT CSV Details
Fama/French 5 Factors (2x3) [Daily] TXT CSV Details

Univariate sorts on Size, B/M, OP, and Inv

Portfolios Formed on Size TXT CSV Details
Portfolios Formed on Size [ex.Dividends] TXT CSV Details
Portfolios Formed on Size [Daily] TXT CSV Details

Quick recap

Week 1

- We downloaded and installed Stata
- Overview of the Stata interface: results window, command window, variables window etc.
- Syntax of Stata commands
- What is a **.do** file, an **.smcl** file, and a **.dta** file
- Saw how to query CRSP data and downloaded the timeseries of returns for a set of securities
- Saw the structure of the CRSP database (stock/company permanent/non-permanent identifiers) and discussed the different variables
- We discussed the different industrial classifications (GICS, NAICS, SIC, NACE, FF) and their main uses
- Imported data by copying and pasting in the data editor
- Changed our working directory (`cd`)

Data Processing in Stata

- Conditional statements & Logical Operators (to use with in-line if-statements):
  - | → 'or' (apply command to obs that satisfy EITHER of the conditions)
  - & → 'and' (apply command to obs that satisfy BOTH conditions)

  ```
  reg y x1 x2 if year=>2000 & year<=2008
  ```
- Deleting VARIABLES (i.e. columns): `drop x1 x2 x3`
- Deleting OBSERVATIONS (i.e. rows): `drop if x1<15` or `drop in 2/5`

Data Processing in Stata

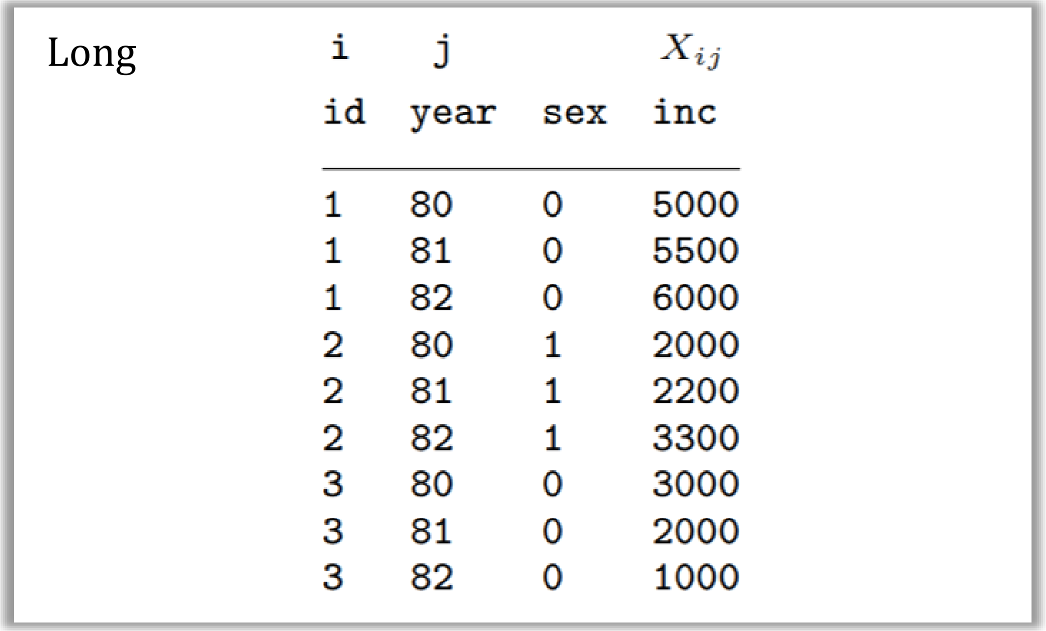
Reshape Dataset

Data in the upper (lower) panel are held in ‘long (wide) form’.

- Go from long to wide:
  `reshape wide inc, i(id) j(year)`
- Go from wide to long:
  `reshape long inc, i(id) j(year)`

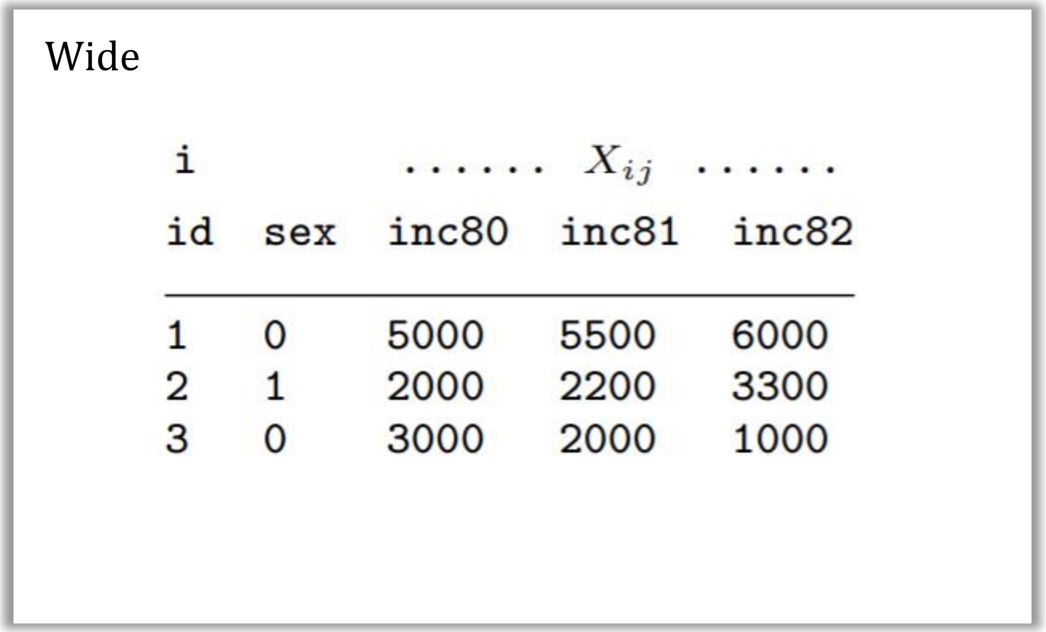
If instead we wanted to have our dataset in a *timeseries* ‘wide’ format (where different rows correspond to different years):
`reshape wide inc, i(year) j(id)`

The ‘long’ format is how we usually structure *panel* datasets.



Long

i	j		X_{ij}
id	year	sex	inc
1	80	0	5000
1	81	0	5500
1	82	0	6000
2	80	1	2000
2	81	1	2200
2	82	1	3300
3	80	0	3000
3	81	0	2000
3	82	0	1000



Wide

i		......	X_{ij}	......
id	sex	inc80	inc81	inc82
1	0	5000	5500	6000
2	1	2000	2200	3300
3	0	3000	2000	1000

Useful Financial & Macro Data Sources

- Google Finance (API via Google Docs): https://docs.google.com/spreadsheets/d/1y_N1mP39jjlpGf3flXHUK_TKvr8RAv8qOnQMaA8g-fo/edit?usp=sharing
- Yahoo Finance
- FRED (Federal Reserve Bank of St. Louis)
- Investing.com
- Quantdl (accessible via the Nasdaq Data Link platform: data.nasdaq.com)

Useful Financial & Macro Data Sources

Depending on the frequency, starting date and type (e.g. prices, volumes etc) of the series you are looking for, there might be different sources you could use to download your data. For example, for **gold**, you can find related series:

- FRED: bullion auction price, starting 1968, daily but with some missing (-use as weekly or monthly, or interpolate missing) https://fred.stlouisfed.org/series/GOLDPMGBD228NLBM
- Investing.com: rolling future prices, daily, starting 1999 https://uk.investing.com/commodities/gold-historical-data
- Quantdl: rolling future price daily, starting 1974 https://www.quandl.com/data/CHRIS/CME_GC1-Gold-Futures-Continuous-Contract-1-GC1-Front-Month

(To use Quantdl you normally need to create a free account, however for this series you can simply go to the table (located next to the graph tab) and copy and past the data to excel)

Quick recap

Week 2

Last week we saw how to:

- Download financial timeseries data via Yahoo Finance and Google Finance
- Load & save Stata (.dta) datasets from and to your disk using `use` and `save`
- Export/save Stata datasets in Excel
- Create tables with frequency and summary statistics for (subsets of) your variables using `tab` and `sum`
- Delete variables (columns) and observations (rows)
- Write commands targeting specific observations using conditional statements (`if`, `in`) & logical operators (`|` `&`) at the end of the command
- Change current working directory (`cd`), and print list of files within a directory (`ls`)
- Rename variables, create new variables (`gen`), and replace values in existing variables (`replace`)
- reshape our dataset from **long** to **wide**
- Explore/print your dataset programmatically using `browse` and `list`
- Temporarily save a ’screenshot’ of your dataset on your computer’s memory using `preserve`, and then restore back to that if needed, using `restore`
- Clear your dataset (using `clear`), and clear the command’s window (`cls`).

Stata Data Types
Type & Format Properties

Variables can be of different **types**. For instance:

- NUMERICAL variables → numerical types: float, double, int, etc.
- STRING variables → string types: str, strL, etc.

NB: The numeric type that stores the most digits is 'double', though at the expense of additional memory.

Variables also have a **format** property which dictates only the appearance of the variable.

To see the Type & Format properties of the variables in your workspace:

- Go to 'Variable Properties' window and look at the 'Type'
- Execute command `describe` or `des`

Stata Data Types

Dates/SIF

Stata's internal data-type for dates is called **SIF** (**Stata internal form**). A SIF is a numerical variable masked (or in other words, using a special format) to appear as a string. In the Stata manual, **SIF dates and times** are also referred to as '**datetimes**’. There are 8 SIF types (see table below).

The (masked) number behind each element of a SIF variable measures the time elapsed (e.g. hours, days, weeks, months, etc) since **00:00:00 AM January 1, 1960**. As a result, the value 0 means the first millisecond, day, week, month, quarter of 1960. SIF variables allow us to make calculations and create conditions, which would not otherwise be possible if the date variable was represented as a string.

datetime/c SIF	milliseconds since 01jan1960 00:00:00.000
datetime/C SIF	milliseconds since 01jan1960 00:00:00.000, adjusted for leap seconds
date/daily SIF	days since 01jan1960 (01jan1960 = 0)
weekly SIF	weeks since 1960w1
monthly SIF	months since 1960m1
quarterly SIF	quarters since 1960q1
half-yearly SIF	half-years since 1960h1
yearly SIF	years since 0000

Stata Data Types

Working with Dates

- Stata provides the following sets of functions to process dates:

A) Convert **string** date variable to SIF variable

B) Convert **numerical** (disaggregated date components) variables into SIF.

C) Convert from one SIF type to another.

D) Extract (numerical) components (year, month, day etc.) from SIF's.

E) Pseudofunctions to conveniently translate dates into SIFs.

Stata Data Types

A) Convert string date variable to SIF variable

```
gen DTs = clock(datestr, "mask")         /* milliseconds from 1jan1960 */
gen DTd = date(datestr, "mask")          /* days from 1jan1960 */
gen DTw = weekly(datestr, "mask")        /* weeks from 1jan1960 */
gen DTm = monthly(datestr, "mask")       /* months from 1jan1960 */
gen DTq = quarterly(datestr, "mask")     /* quarters from 1jan1960 */
```

use `clock()` when the string contains both a **date** & a **time** e.g. '01 Dec 2006 14:22'.
use `date()` when the string contains **only** a **date** e.g. '22/7/2010'.
use `weekly()` when the string contains a **year** and a **week** number (1–52) e.g. '2020 week-1'.
use `monthly()` when the string contains a **year** and a **month** number (1–12) e.g. '2020-06'.
use `quarterly()` when the string contains a **year** and a **quarter** number (1–4) e.g. '2020Q1'.

"mask" specifies the order of the date components (year, month, day, hour, etc). Examples:

- datestr = "2010.07.12 14:32" → use "YMDhm" with clock()
- datestr = "2006-12-01 14:22:43" → use "YMDhms" with clock()
- datestr = "July 12, 2010 2:32 PM“ → use "MDYhm" with clock()
- datestr = "22/7/2010" → use "DMY" with date()

Stata Data Types

B) Convert numerical (disaggregated date components) variables into SIF

```
gen DTs = mdyhms(M, D, Y, h, m, s) /* milliseconds elapsed since 01jan1960 00:00:00.0 */
gen DTd = mdy(M,D,Y)               /* days elapsed since 01jan1960 until M, D, Y */
gen DTm = ym(Y,M)         /* months elapsed since 1960m1 until year Y, month M */
gen DTq = yq(Y,Q)         /* quarters elapsed since 1960q1 until year Y, quarter Q */
gen DTw = yw(Y,W)         /* weeks elapsed since 1960w1 until year Y, week W */
```

Stata Data Types

C) Convert from one SIF type to another

```
gen DTw = wofd(DTd)           /* From DAYS elapsed to WEEKS elapsed */
gen DTm = mofd(DTd)           /* From DAYS elapsed to MONTHS elapsed */
gen DTq = qofd(DTd)           /* From DAYS elapsed to QUARTERS elapsed */
gen DTs = cofd(DTd)           /* From DAYS elapsed to MILLISECONDS elapsed */
gen DTd = dofm(DTm)           /* From MONTHS elapsed to DAYS elapsed */
gen DTd = dofq(DTq)           /* From QUARTERS elapsed to DAYS elapsed */
gen DTd = dofc(DTs)           /* From MILLISECONDS elapsed to DAYS elapsed */
```

You can also use combinations of the functions above:

```
gen DTq = qofd(dofm(DTm))  /* From MONTHS elapsed to QUARTERS elapsed */
```

Stata Data Types

D) Extract (numerical) components (year, month, day etc.) from daily SIF’s

```
gen y = year(DTd)
gen m = month(DTd)
gen d = day(DTd)
gen day = dow(DTd)   /* 0 for Sunday, 1 for Monday, ..., 6 for Saturday */
```

- Note:

These functions only accept **daily SIF** as an input. To extract components from monthly SIF, first convert the monthly SIF to daily SIF:

```
gen y = year(dofm(DTm))
```

- Example:

If we have a daily SIF (DTd) that contains the end-of-month e.g. 2019/03/31, 2019/04/30 etc., we can create a new daily SIF variable with the same year/month as in DTd but taken back to the 1st day of the month:

```
gen DTd_2 = mdy(month(DTd), 1, year(DTd))
```

Stata Data Types

E) Pseudofunctions to conveniently translate dates into SIFs

SIF pseudofunctions allow you to easily translate a desired date into a SIF. They only require that you specify the date and time components in the expected order:

```
keep if TDs >= tc(day-month-year hh:mm:ss)   /* for datetime SIFs */
keep if TDd >= td(day-month-year)            /* for date SIFs */
keep if TDw >= tw(year-week)                 /* for weekly SIFs */
keep if TDm >= tm(year-month)                /* for monthly SIFs */
keep if TDq >= tq(year-quarter)               /* for quarterly SIFs */
```

- Examples:

Drop/keep/choose specific rows:

```
keep if TDd >= td(30-3-2018)
keep if TDd >= td(30mar2018)
```

Create period-specific dummy variables (i.e. =1 when the condition is satisfied, 0 otherwise)

```
gen v1 = (TDd < td(1,1,2008))
gen v2 = (TDd < td(1jan2008))
gen v3 = (TDd < date("January 1 2008","MDY"))   /* Same as above, using functions in A)*/
gen v4 = (TDm < tm(2008,1))
gen v5 = (TDm < tm(2008m1))
```

Quick recap
Week 3

Last week we saw:

- 2 different types of 'primary' data that Stata (and other packages/languages) can process:
  - Numerical data
  - Strings (which are collections of characters & letters)
- The **format**, **type** and **label** properties of variables, and how to change these.
- What is a SIF and how to work with dates in Stata. To do so, we introduced 5 sets of functions to process dates:
  A. Convert **string** date variable to SIF variable
  B. Combine **numerical** variables (containing disaggregated date components e.g. year, month, day etc.) into a SIF variable.
  C. Convert from one SIF type to another (e.g. from daily SIF to monthly SIF variable).
  D. Extract components (year, month, day etc.) from daily SIF's.
  E. Pseudofunctions to conveniently translate dates into SIF's (-most commonly used together with in-line 'if' statements).
- Finally, we created 2 examples of timeseries datasets in Excel and used the above functions to prepare the dates (-convert them to SIF) for the analysis.

Linear Regression Models

Assumptions & Hypothesis Testing

Classical Linear Regression Model
Setup

- Linear Regression Model

The **true model** that describes the population is given by:

$$y_t = x_t{}'\beta + \varepsilon_t$$

$$\text{with } \; t = 1,2,..,T \text{ and } \; x_t' = [x_{t1}, x_{t2}, \dots, x_{tK}],$$

or equivalently in matrix notation:

$$\boldsymbol{Y} = \boldsymbol{X\beta} + \boldsymbol{\varepsilon}$$

$$\text{where } \boldsymbol{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_T \end{bmatrix}, \; \boldsymbol{X} = \begin{bmatrix} x_{11} & \cdots & x_{1\mathrm{K}} \\ x_{21} & \cdots & x_{2K} \\ \vdots & \ddots & \vdots \\ x_{T1} & \cdots & x_{TK} \end{bmatrix}, \boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_K \end{bmatrix}, \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_T \end{bmatrix}$$

Classical Linear Regression Model

Assumptions

- The Classical Linear Regression Model (CLRM) assumptions are:

(1) Linearity of Y and X, i.e. $\boldsymbol{Y} = \boldsymbol{X\beta} + \boldsymbol{\varepsilon}$

(2) No perfect multi-collinearity, $\rho(\boldsymbol{X'X}) = K$,

i.e. the square matrix $\boldsymbol{X'X}$ is full rank, implying that $\boldsymbol{X'X}$ is invertible

(3) Exogeneity of error (e.g. no omitted variables): $\mathrm{E}(\varepsilon_t/\boldsymbol{X}) = 0$

(4) Spherical error variance:

a. Homoscedasticity: $\mathrm{Var}(\varepsilon_t/\boldsymbol{X}) = \mathrm{E}(\varepsilon_t^2/\boldsymbol{X}) = \sigma_\varepsilon^2$

b. No Autocorrelation: $\mathrm{Cov}(\varepsilon_t, \varepsilon_\tau/\boldsymbol{X}) = \mathrm{E}(\varepsilon_t, \varepsilon_\tau/\boldsymbol{X}) = 0$

(5) **Normality** of the true error term

$\varepsilon_t \sim N(0, \sigma_\varepsilon^2)$

or $\boldsymbol{\varepsilon} \sim N(\boldsymbol{0}, \sigma_\varepsilon^2 I_T)$

- Under assumptions (1)-(3) OLS estimator is **unbiased**.
- Under assumptions (1)-(4) OLS is also **efficient** (-the Gauss-Markov Theorem), and OLS is said to be BLUE (Best Linear Unbiased Estimator).
- Under all five CLRM assumptions, the PDF of y_t/X and $\hat{\beta}_{ols}/X$ is the **Gaussian** distribution, allowing us to:

(i) calculate the **likelihood function** and estimate the parameters via MLE, (ii) conduct **hypothesis testing**.

Classical Linear Regression Model

Solution: OLS Estimator

The **ordinary least squares** (**OLS**) solution proposes to find the *best-fitting curve* to a given set of points by minimizing the **sum of the squares of the residuals** (RSS) of the points from the curve.

$$\hat{\beta}_{ols} = \arg\min_{\hat{\beta}} RSS$$

Effectively, we minimize the variance of the residuals

where $RSS = \sum_{t=1}^{T} \hat{\varepsilon}_t^2 = \sum_{t=1}^{T} (y_t - x_t{'}\hat{\beta})^2$ or in matrix notation $RSS = \hat{\varepsilon}' \ \hat{\varepsilon} \ = (Y - X\hat{\beta})'(Y - X\hat{\beta})$.
In the simple case where we have a constant and a single variable (i.e. $K = 2$):

$$\hat{\beta}_{2,ols} = \frac{\sum_{t=1}^{T} y_t x_t / T \ - \bar{y}\bar{x}}{\sum_{t=1}^{T} x_t^2 / T \ - \ \bar{x}^2} \text{ and } \hat{\beta}_{1,ols} = \bar{y} - \hat{\beta}_{2,ols}\bar{x}$$

The solution for **any *K***, can be conveniently written in matrix notation:

$$\hat{\beta}_{ols} = (X'X)^{-1}X'Y$$

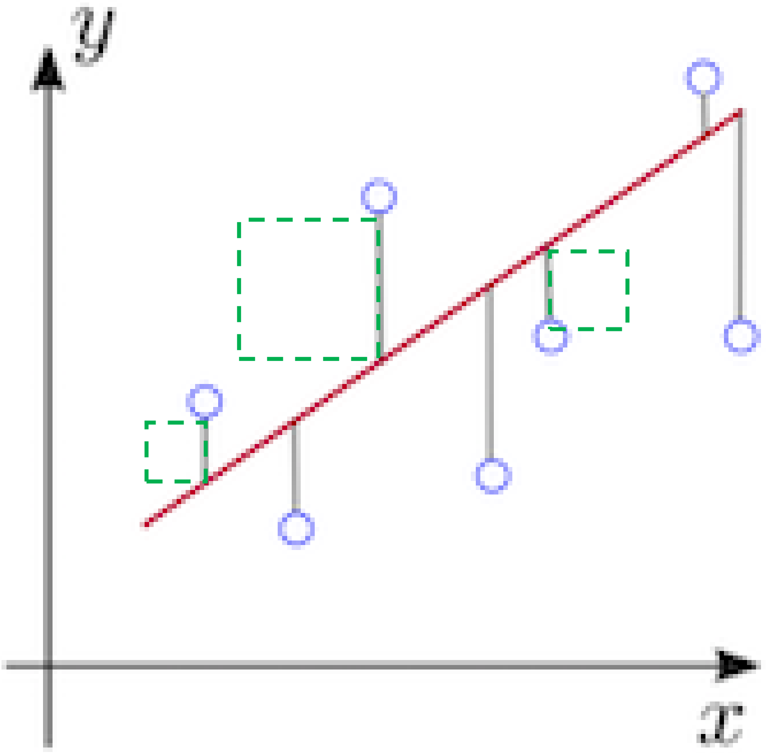
An alternative *loss criterion* in the objective function would be to use the **residual absolute values** instead of the squares of the residuals. This would result in an alternative estimator for β_{true} known as **least absolute deviations** (**LAD**). Why OLS is a preferable estimator? Because OLS is…

B.L.U.E.

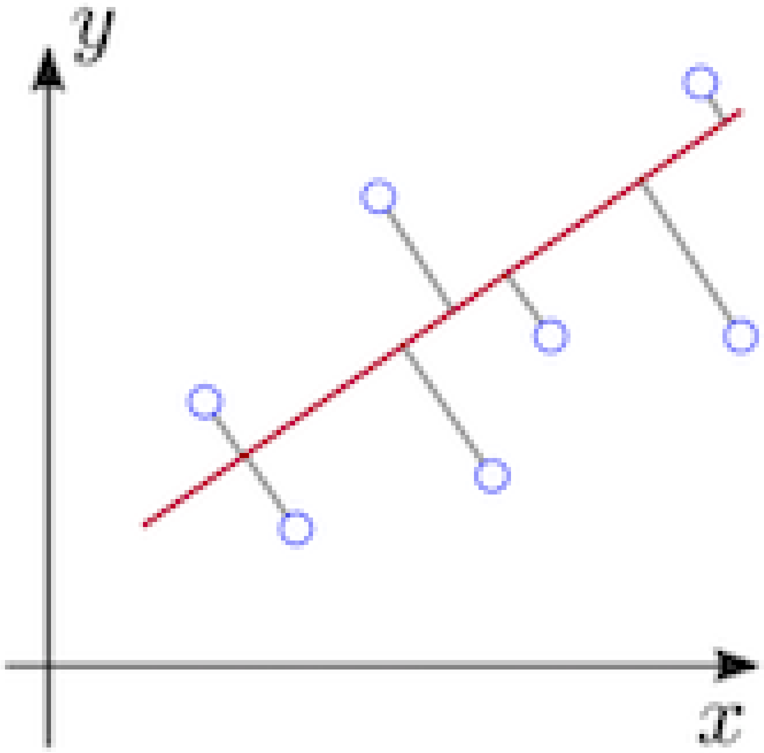
Classical Linear Regression Model

Visualizing the Objective (with 1 RHS variable and a constant)

OLS



LAD



Classical Linear Regression Model

Solution: OLS Estimator

Note that while we derive $\hat{\beta}_{ols}$ from the F.O.C. of the **minimization of the *RSS*** (i.e. $\partial RSS/\partial\hat{\beta} = 0$), $\hat{\sigma}^2_{ols}$ is derived from the F.O.C. of the **maximum likelihood** optimization problem. Specifically, the F.O.C. of the ML optimization problem yields:

$$\hat{\sigma}^2_{mle} = \frac{RSS_{mle}}{T}$$

but this can be shown to be a **biased** estimate of the **true error variance**. Dividing by T-K, instead, corrects the MLE estimator, making it **unbiased**. And this is what statistical packages use as the OLS estimator for the error variance.

$$\hat{\sigma}^2_{ols} = \frac{RSS_{ols}}{T-K} = \frac{\hat{\varepsilon}'_{ols}\hat{\varepsilon}_{ols}}{T-K}$$

Consequently, we can derive the **variance-covariance** matrix of $\hat{\beta}_{ols}$

$$VCOV(\hat{\beta}_{ols}/X) = \sigma^2_{\varepsilon,true}(X'X)^{-1} \longrightarrow$$

Think of this as an unknown 'constant'

Given assumption (A5), $\varepsilon_{true} \sim N\left(0, \sigma^2_{\varepsilon,true} I_T\right)$, we then have the distribution of $\hat{\beta}_{ols}$

$$\hat{\beta}_{ols}/X \sim N\left(\beta_{true}, \sigma^2_{\varepsilon,true}(X'X)^{-1}\right)$$

And therefore, the **estimator for the variance-covariance** is:

$$\widehat{VCOV}(\hat{\beta}_{ols}/X) = \hat{\sigma}^2_{ols}(X'X)^{-1} \longrightarrow$$

This is a random variable (i.e. has a distribution itself)

Classical Linear Regression Model

Hypothesis Testing

- We start with the simple **sample mean estimator**, which is equivalent to the linear model with just a constant. It is easy then to generalize to the multivariate problem. You can think of the true model being:

$$\left.\begin{array}{c} x_i = \mu_{true} + \varepsilon_{i,true} \\ \text{for } \; i = 1,2,..,N \text{ with } \varepsilon_{i,true} \sim N\left(0,\sigma^2_{\varepsilon,true}\right) \end{array}\right\} \quad x_i \sim N\left(\mu_{true},\sigma^2_{\varepsilon,true}\right)$$

- It turns out that the solution to the **least-squares optimization** problem for the **estimator** of the true (population) mean μ_{true}, is the *sample average*.

$$\hat{\mu}_{ls} = \frac{\sum_{i=1}^{N} x_i}{N}, \text{ and it can also be shown that: } \hat{\mu}_{ls} \sim N\left(\mu_{true}, \frac{\sigma^2_{\varepsilon,true}}{N}\right)$$

- Like before, the estimator for the error variance $\sigma^2_{\varepsilon,true}$ can be derived from the **maximum-likelihood** problem. It will be biased, but we can correct it by adjusting (the df's in) the denominator.

$$\hat{\sigma}^2_{mle} = \frac{\sum_{i=1}^{N}(x_i - \hat{\mu}_{ls})^2}{N}, \text{ but we will use: } \hat{\sigma}^2_{ls} = \frac{\sum_{i=1}^{N}(x_i - \hat{\mu}_{ls})^2}{N-1}$$

Classical Linear Regression Model

Hypothesis Testing

- We start with the simple **sample mean estimator**, which is equivalent to the linear model with just a constant. It is easy then to generalize to the multivariate problem. You can think of the true model being:

$$\left.\begin{aligned} x_i &= \mu_{true} + \varepsilon_{i,true} \\ \text{for } i &= 1,2,..,N \text{ with } \varepsilon_{i,true} \sim N\left(0, \sigma^2_{\varepsilon,true}\right) \end{aligned}\right\} \quad x_i \sim N\left(\mu_{true}, \sigma^2_{\varepsilon,true}\right)$$

- It turns out that the solution to the **least-squares optimization** problem for the **estimator** of the true (population) mean μ_{true}, is the *sample average*.

$$\hat{\mu}_{ls} = \frac{\sum_{i=1}^{N} x_i}{N}, \text{ and it can also be shown that: } \hat{\mu}_{ls} \sim N\left(\mu_{true}, \frac{\sigma^2_{\varepsilon,true}}{N}\right)$$

- Like before, the estimator for the error variance $\sigma^2_{\varepsilon,true}$ can be derived from the **maximum-likelihood** problem. It will be biased, but we can correct it by adjusting (the df's in) the denominator.

$$\boxed{\hat{\sigma}^2_{mle} = \frac{\sum_{i=1}^{N} (x_i - \hat{\mu}_{ls})^2}{N}}, \text{ but we will use: } \boxed{\hat{\sigma}^2_{ls} = \frac{\sum_{i=1}^{N} (x_i - \hat{\mu}_{ls})^2}{N-1}}$$

And now you know the difference between Excel's =STDEV.P() and =STDEV.S()! 👏

Classical Linear Regression Model

Hypothesis Testing

$$\hat{\sigma}^2_{ls} = \frac{\sum_{i=1}^{N}(x_i - \hat{\mu}_{ls})^2}{N-1}$$

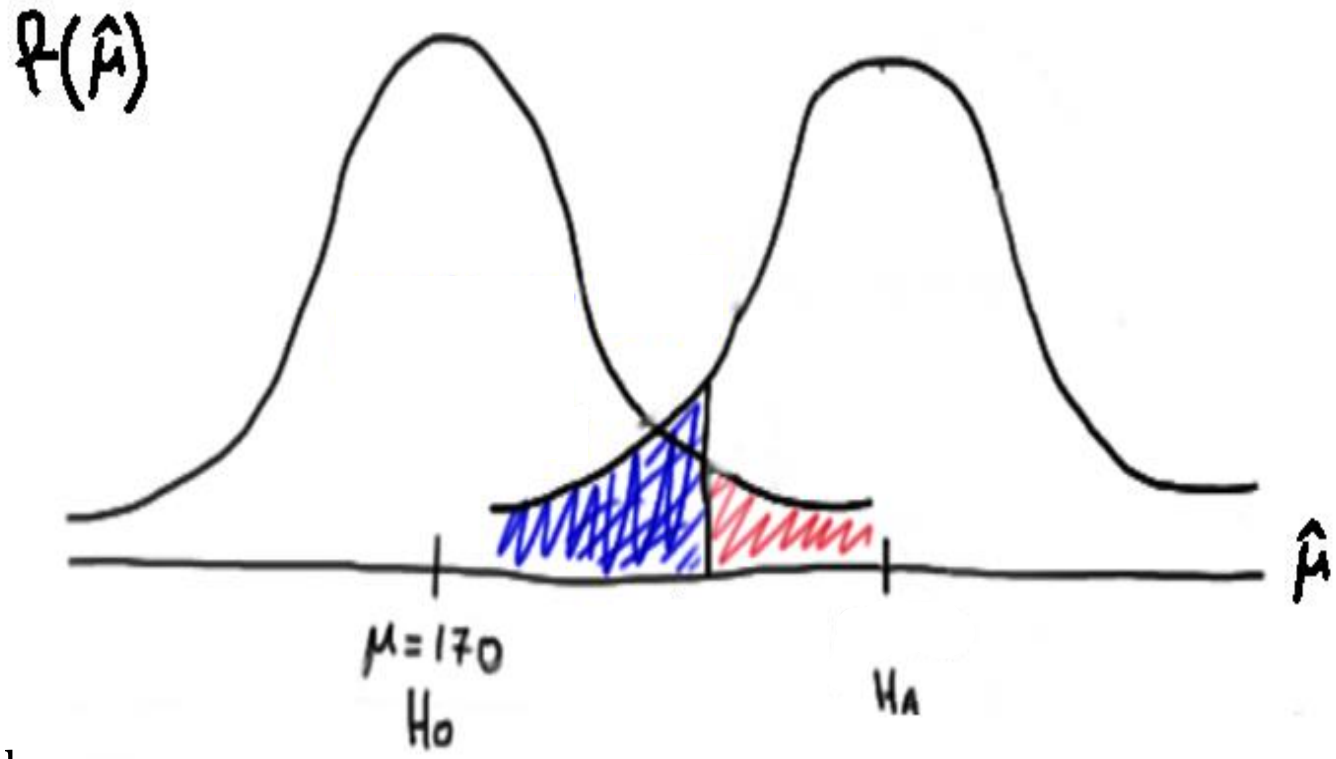
- Assume we have a sample containing information about the *height* x_i in cm for 25 individuals, and the **average** individual in the sample is $\hat{\mu}_{ls} = 172$ cm.
- Recall that $\hat{\mu}_{ls} \sim N\left(\mu_{true}, \frac{\sigma^2_{\varepsilon,true}}{N}\right)$, which implies $\frac{\hat{\mu}_{ls} - \mu_{true}}{\sqrt{\sigma^2_{\varepsilon,true}/N}} \sim N(0,1)$, and $\frac{\hat{\mu}_{ls} - \mu_{true}}{\sqrt{\hat{\sigma}^2_{ls}/N}} \sim T(N-1)$.
- For the purposes of the example that follows, we will assume that the *true* error variance $\sigma^2_{\varepsilon,true}$ is *known* and equal to 10. The only difference in what follows is that we will then need to use the **z-statistic**, instead of the **t-statistic** (which is what we use in practise).
- To test whether $\hat{\mu}_{ls}$ is **statistically significant**, would be to test if the **true mean** could be equal to 0 (H_0: $\mu_{true} = 0$). But this would not be an interesting hypothesis to test, for this sample. Why?
- A more interesting hypothesis would be to test if the true mean could be argued to be equal to 170 (H_0: $\mu_{true} = 170$), against the alternative that the true mean is higher (H_A: $\mu_{true} > 170$).

Classical Linear Regression Model

Hypothesis Testing

- $\hat{\mu}_{ls} \sim N\left(\mu_{true}, \frac{\sigma^2_{\varepsilon,true}}{N}\right)$
- Assuming the true error variance is known, we can then use the Normal distribution:

  $\mathrm{Var}(\hat{\mu}_{ls}) = \frac{\sigma^2_{\varepsilon,true}}{N} = \frac{100}{25}$
- $z = \frac{\hat{\mu}_{ls} - \mu_{true}}{\sqrt{\sigma^2_{\varepsilon,true}/N}} \sim N(0,1)$
- H_0: $\mu_{true} = 170$
- H_A: $\mu_{true} > 170$

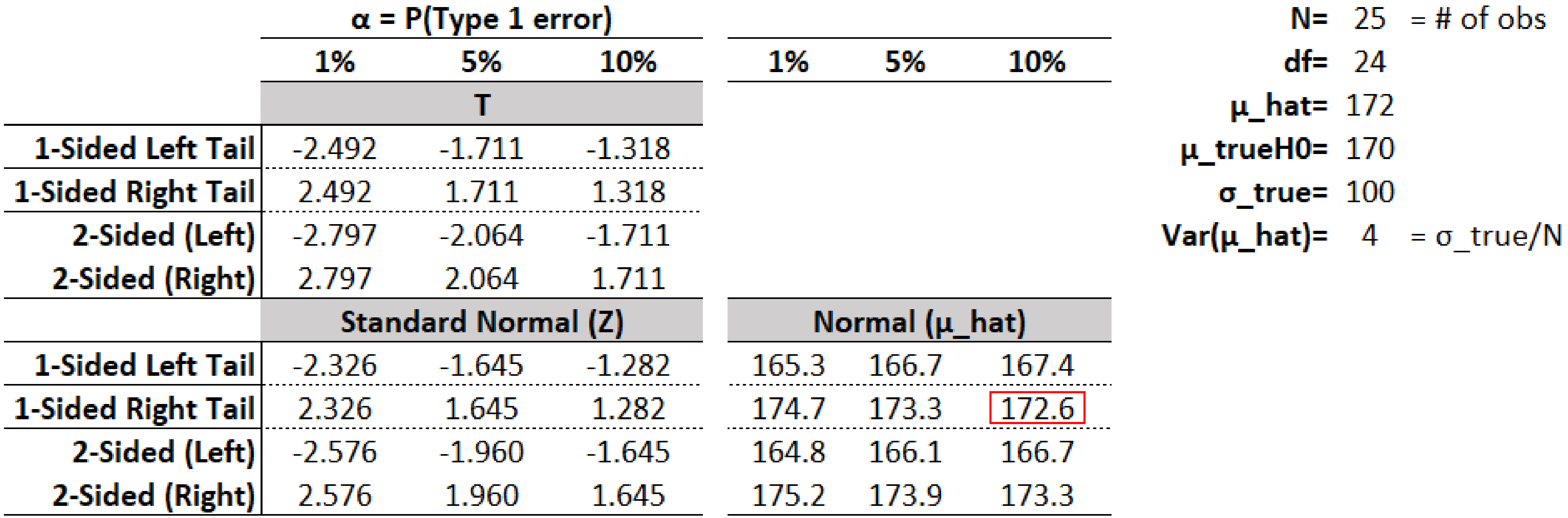


α = P(Type 1 error) = P of rejecting true H0 → red area

β = P(Type 2 error) = P of accepting false H0 → blue area

Classical Linear Regression Model

Hypothesis Testing



	α = P(Type 1 error)					
	1%	5%	10%	1%	5%	10%
	T					
1-Sided Left Tail	-2.492	-1.711	-1.318			
1-Sided Right Tail	2.492	1.711	1.318			
2-Sided (Left)	-2.797	-2.064	-1.711			
2-Sided (Right)	2.797	2.064	1.711			
	Standard Normal (Z)			Normal (μ_hat)		
1-Sided Left Tail	-2.326	-1.645	-1.282	165.3	166.7	167.4
1-Sided Right Tail	2.326	1.645	1.282	174.7	173.3	172.6
2-Sided (Left)	-2.576	-1.960	-1.645	164.8	166.1	166.7
2-Sided (Right)	2.576	1.960	1.645	175.2	173.9	173.3

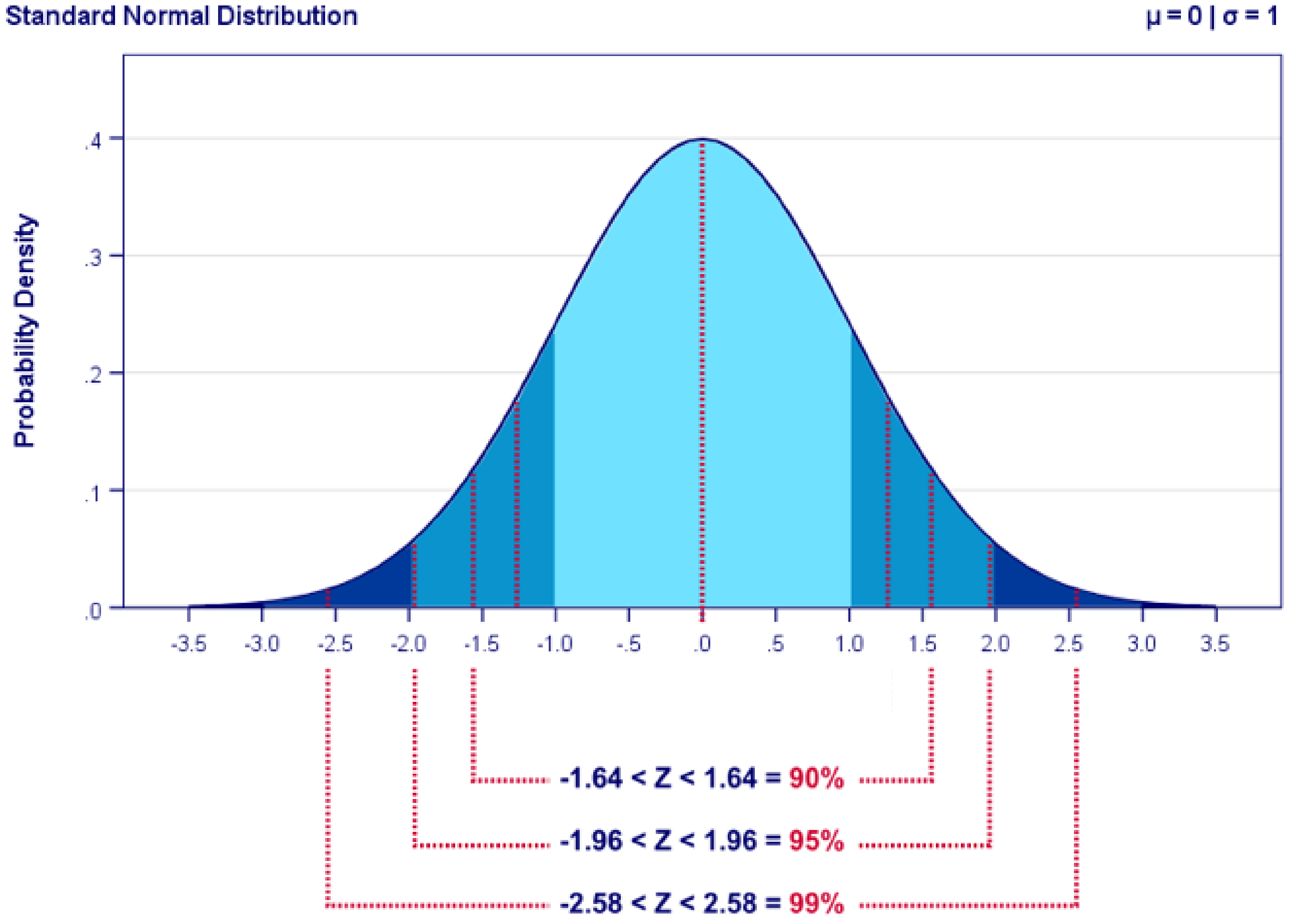
N=	25	= # of obs
df=	24	
μ_hat=	172	
μ_trueH0=	170	
σ_true=	100	
Var(μ_hat)=	4	= σ_true/N

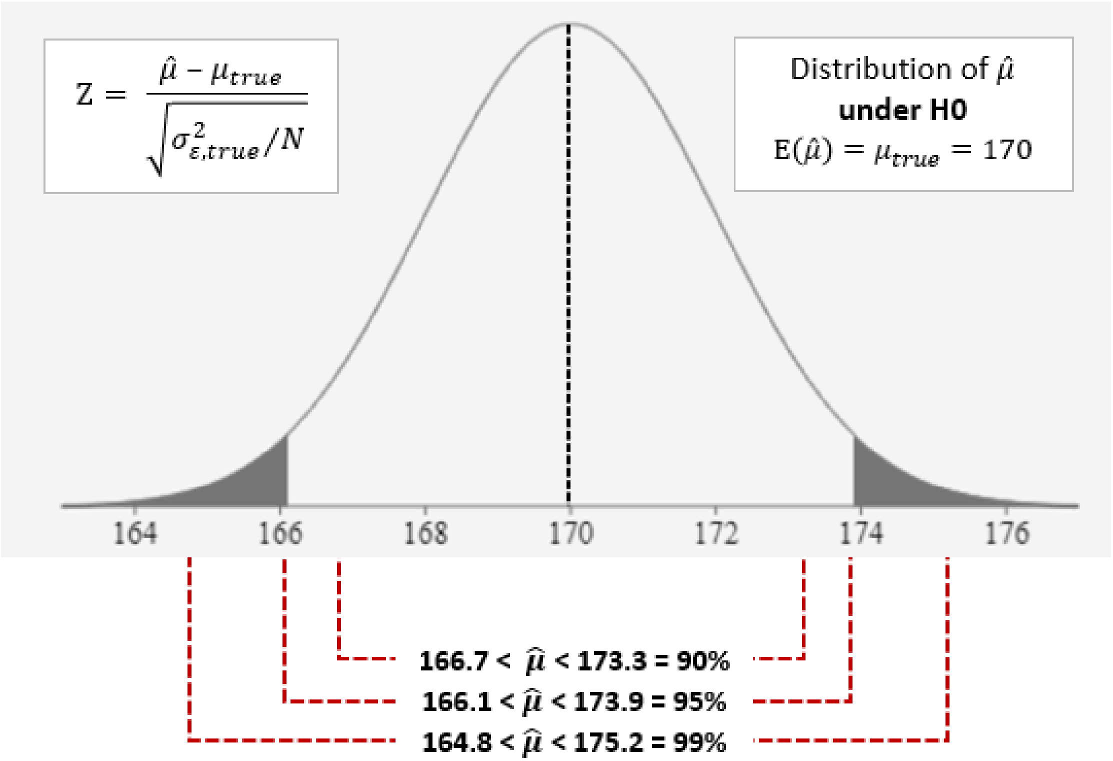
Verdict:

We **fail to reject the <u>null</u>** of $\mu_{true} = 170$ at the 10% level of significance (against the **1-sided <u>alternative</u>**)

Classical Linear Regression Model

Hypothesis Testing





Classical Linear Regression Model

Hypothesis Testing

CONCLUSION based on the **sample**'s value

		H0 Accepted **(Fail 2 Reject)**	**H0 Rejected**	
The entire **population**'s **TRUE** value	**H0 True** (Ha False)	Correct (1-α) TRUE Positive	Type I Error: P(error)=α FALSE Positive	→ **α = Significance Level**; 1-α = Test Specificity → reject true H0
	H0 False (Ha True)	Type II Error: P(error)=β FALSE Negative	Correct (1-β) TRUE Negative	→ Test Power (aka Test Sensitivity) = P(reject H0|Ha True) = 1-β
		↳ accept false H0	↳ reject false H0	

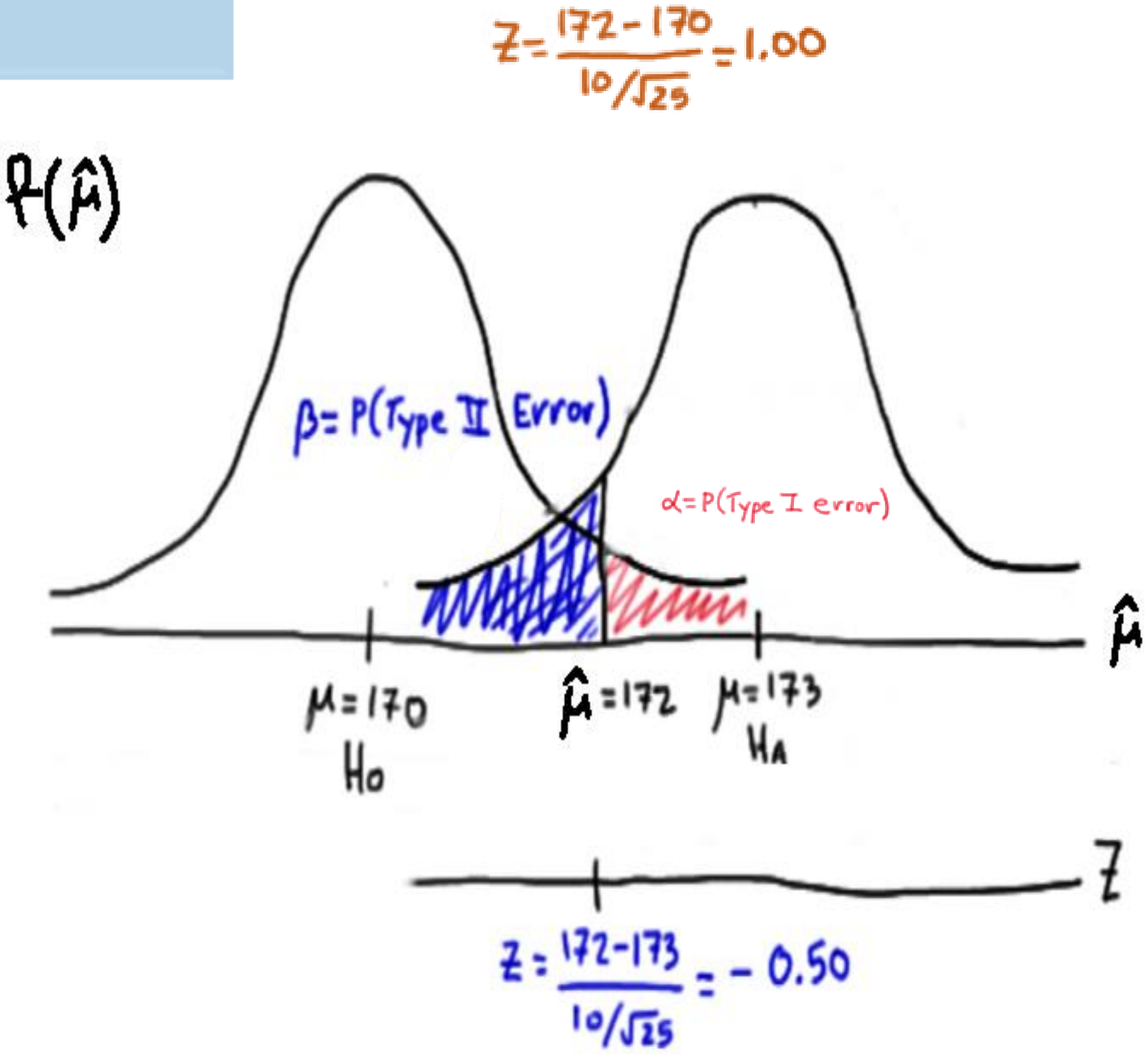
Classical Linear Regression Model

Hypothesis Testing

- Let's say that we know that the **true** population **mean** is 173 and the **true error variance** is 100.
  - $\mu_{true} = 173, \;\; \sigma^2_{\varepsilon,true} = 100$
- The **sample mean** from 25 randomly chosen individuals came out to be 172.
  - $\hat{\mu}_{ls} = \bar{x} = 172$
- The **hypothesis of interest** (H_0) is whether the true population is equal to 170.
  - $\mu^{H_0}_{true} = 170$
- We can then calculate the:
  - Prob. of committing a **Type I error**:

    $P(\hat{\mu}_{ls} \geq 172 \mid \mu_{true} = 170) = P(z \geq 1) = 0.158$
  - Prob. of committing a **Type II error**:

    $P(\hat{\mu}_{ls} \leq 172 \mid \mu_{true} = 173) = \mathrm{P}(\mathrm{z} < -0.5) = 0.308$



$Z = \frac{172-170}{10/\sqrt{25}} = 1.00$
$f(\hat{\mu})$
β= P(Type II Error)
α=P(Type I error)
μ=170
H₀
$\hat{\mu}$=172
μ=173
H_A
$\hat{\mu}$
Z
$z = \frac{172-173}{10/\sqrt{25}} = -0.50$

Classical Linear Regression Model

Hypothesis Testing: F-test

We may wish to know whether **two or more variables** are **jointly significant** in a regression. It may be that two or more variables have statistically **insignificant** t-test scores but are jointly significant. This may occur if two or more variables are collinear with one another (i.e. have relatively high levels of correlation with one another).

The F-test can be employed to test the joint significance of any number of coefficients. An interesting case is to test whether **all betas** are **jointly equal to 0**. In this special case, the p-value for the F-test is actually the probability of observing such a large R^2 value (-the % of variation explained), as you found with your data, or larger, if there had been no actual association in the population.

Classical Linear Regression Model
Hypothesis Testing: F-test

The **null hypothesis (H0)** in the F-test is that all coefficient estimates, except for the intercept term, are zero. In other words, the null hypothesis claims that **there is no predictive relationship between the X variables and Y,** implying that **Y is pure randomness**.

The **alternative hypothesis (Ha)** claims that there is some predictive relationship between the X variables and Y, or in other words, **Y must depend <u>on at least one</u> of the X variables**.

Assuming, the model:

$$y_t = \alpha + \beta_1 x_{1,t} + \beta_2 x_{2,t} + \beta_3 x_{3,t} + \varepsilon_t$$

We are interested in testing the null:

$$H_0 \colon \beta_{1,true} = \beta_{2,true} = \beta_{3,true} = 0$$

$$H_0 \colon \; y_t = \alpha + \varepsilon_t$$

Classical Linear Regression Model

Hypothesis Testing: F-test

The F-test statistic is the result of a **Likelihood Ratio (LR) test** (where both the restricted & the unrestricted models can be estimated optimally via OLS).

$$\frac{(RSS_R - RSS_U)/(K-1)}{RSS_U/(T-K)} \sim F(K-1, T-K)$$

Where RSS_U and RSS_R denote the residual sum of squares, from the **unrestricted** and **restricted** models, respectively, with the restricted model containing only the constant. Because the restricted model contains only the constant, we have that $\hat{y}_t = \bar{y}$, and $RSS_R = \sum_{t=1}^{T} (y_t - \bar{y})^2 = TSS_U$, hence:

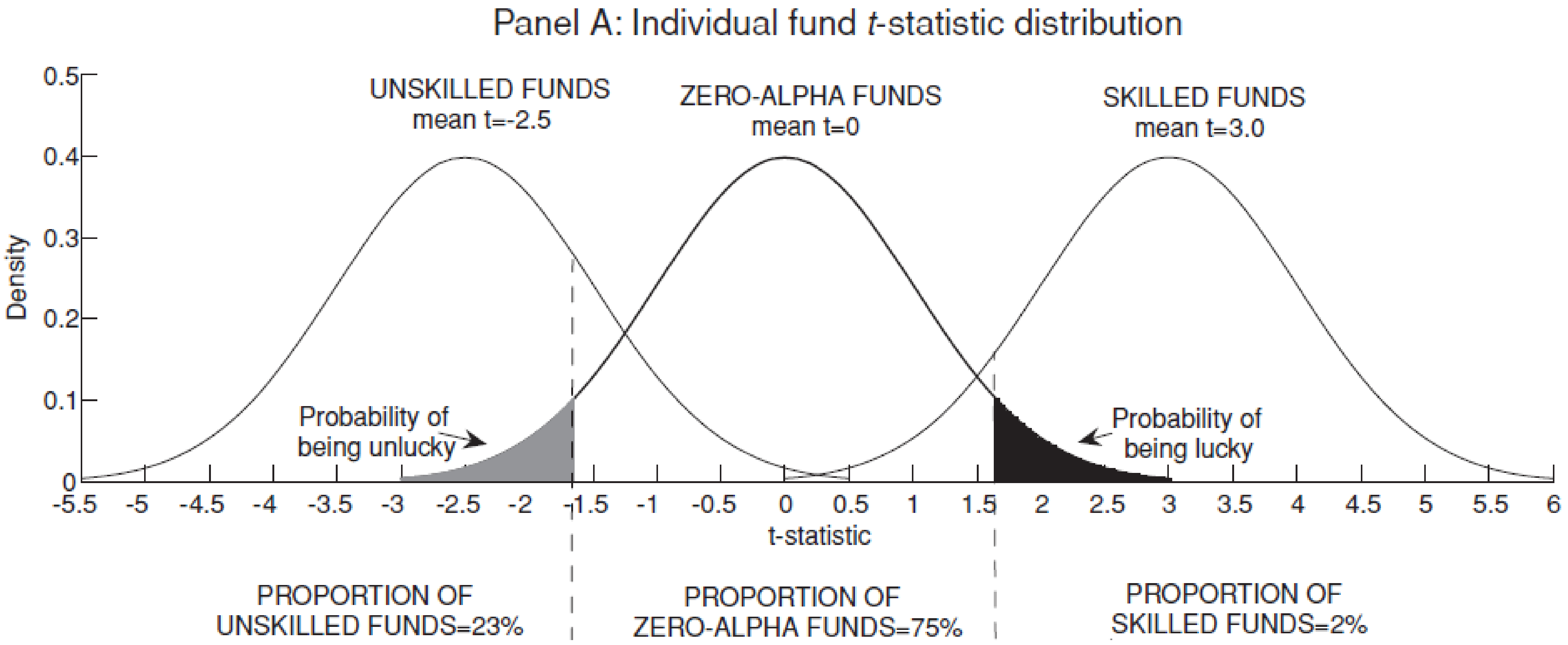
$$\frac{(TSS_U - RSS_U)/(K-1)}{RSS_U/(T-K)}$$

Since TSS = MSS + RSS, the F-test for the case where all betas are equal to 0 under the null, becomes:

$$\frac{MSS_U/(K-1)}{RSS_U/(T-K)} \sim F(K-1, T-K)$$

Application: Mutual Fund Performance

Skilled vs Lucky -and- Unskilled vs Unlucky



Threshold = +1.65

Barras L., Scaillet O., & Wermers R. (2010). False discoveries in mutual fund performance: Measuring luck in estimated alphas. The Journal of Finance 65, 179–216.

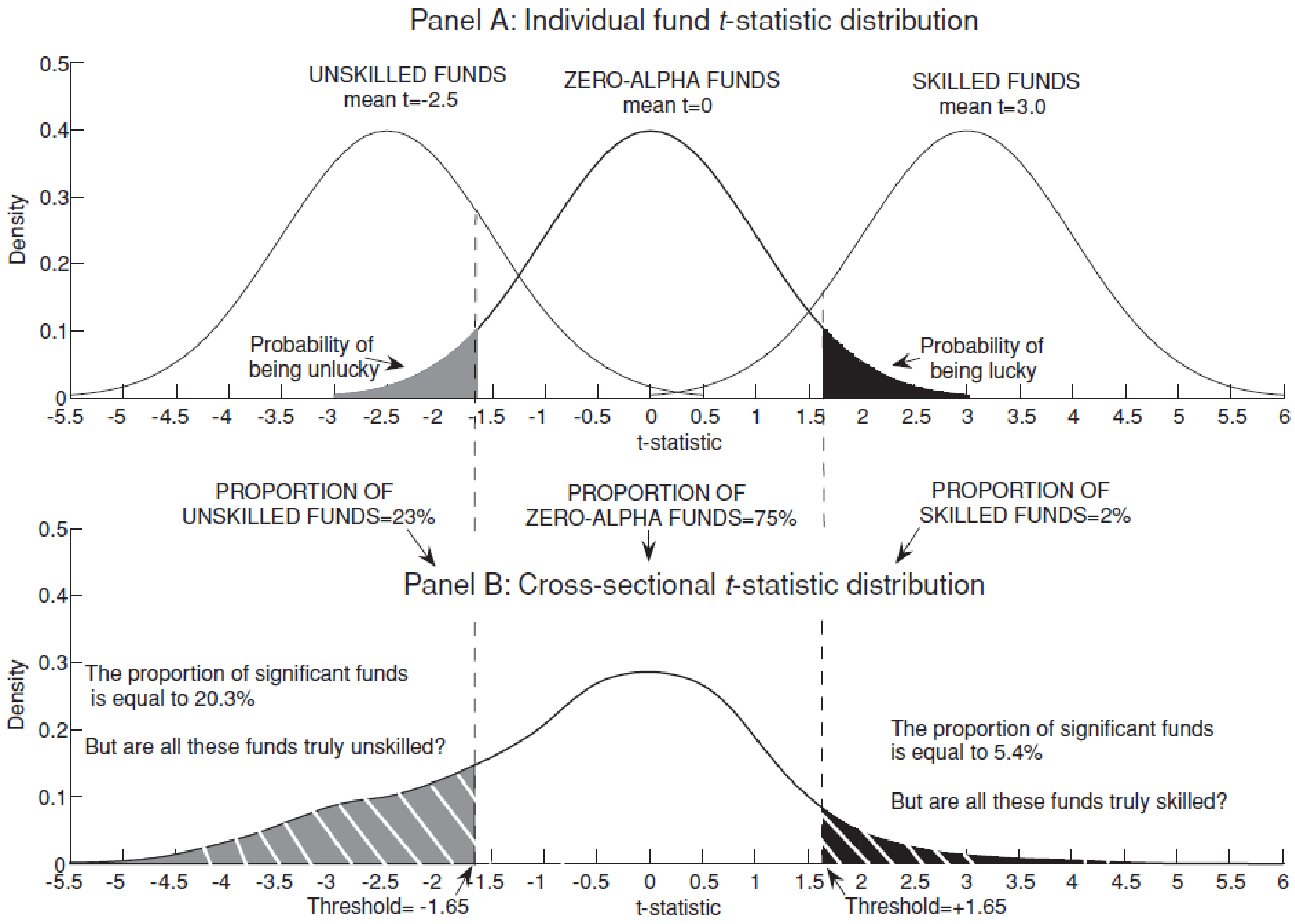
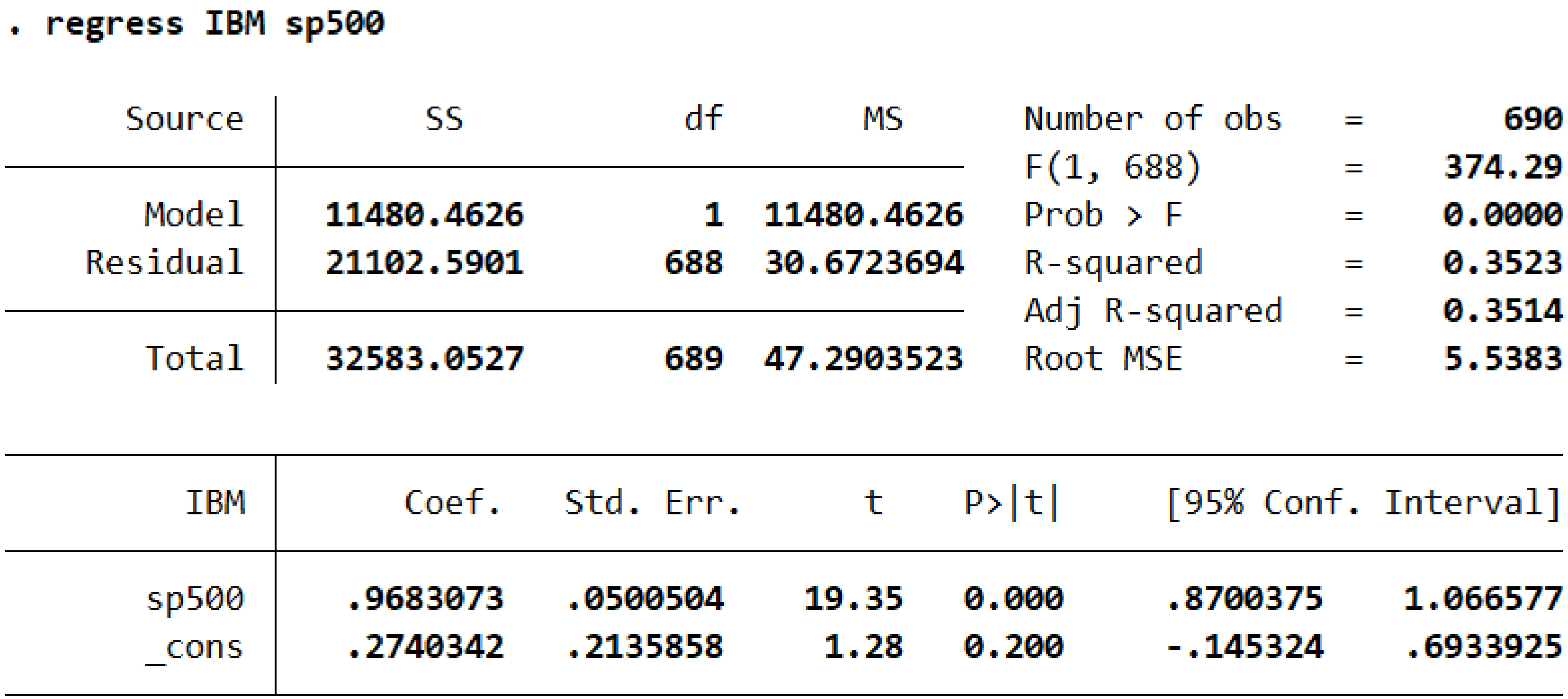


Figure 1. Outcome of the multiple performance test. Panel A shows the distribution of the fund *t*-statistic across the three skill groups (zero-alpha, unskilled, and skilled funds). We set the true four-factor alpha equal to -3.2% and $+3.8\%$ per year for the unskilled and skilled funds (implying that the *t*-statistic distributions are centered at -2.5 and $+3$). Panel B displays the cross-sectional *t*-statistic distribution. It is a mixture of the three distributions in Panel A, where the weight on each distribution depends on the proportion of zero-alpha, unskilled, and skilled funds in the population (π_0, π_A^-, and π_A^+). In this example, we set $\pi_0 = 75\%$, $\pi_A^- = 23\%$, and $\pi_A^+ = 2\%$ to match our average estimated values over the final 5 years of our sample.

Back to Stata…

OLS Regression in Stata

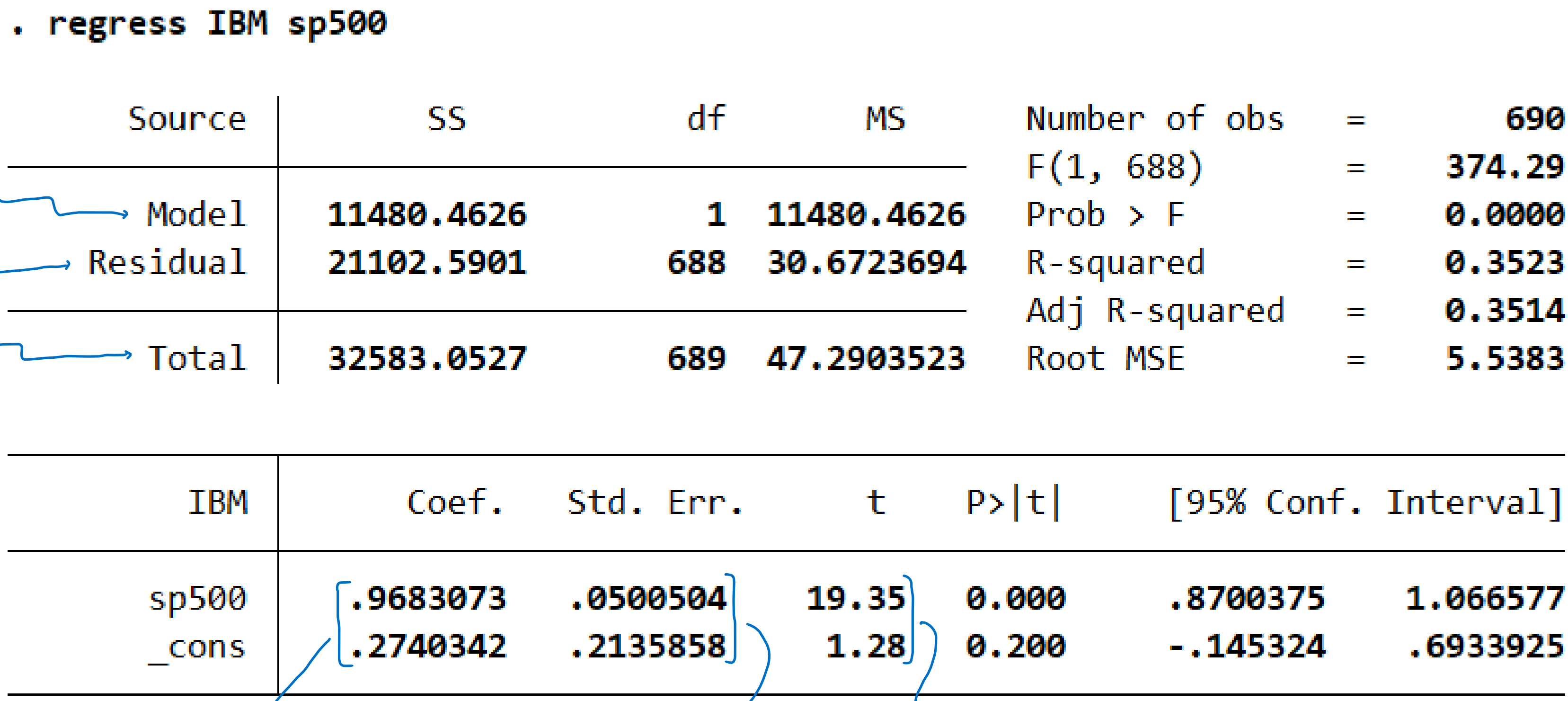
Output



```
. regress IBM sp500

      Source |       SS           df       MS      Number of obs   =       690
-------------+----------------------------------   F(1, 688)       =    374.29
       Model |  11480.4626         1  11480.4626   Prob > F        =    0.0000
    Residual |  21102.5901       688  30.6723694   R-squared       =    0.3523
-------------+----------------------------------   Adj R-squared   =    0.3514
       Total |  32583.0527       689  47.2903523   Root MSE        =    5.5383

------------------------------------------------------------------------------
         IBM |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       sp500 |   .9683073   .0500504    19.35   0.000     .8700375    1.066577
       _cons |   .2740342   .2135858     1.28   0.200     -.145324    .6933925
------------------------------------------------------------------------------
```

OLS Regression in Stata

Output



```
. regress IBM sp500

      Source |       SS           df       MS      Number of obs   =       690
-------------+----------------------------------   F(1, 688)       =    374.29
       Model |  11480.4626         1  11480.4626   Prob > F        =    0.0000
    Residual |  21102.5901       688  30.6723694   R-squared       =    0.3523
-------------+----------------------------------   Adj R-squared   =    0.3514
       Total |  32583.0527       689  47.2903523   Root MSE        =    5.5383

------------------------------------------------------------------------------
         IBM |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       sp500 |   .9683073   .0500504    19.35   0.000     .8700375    1.066577
       _cons |   .2740342   .2135858     1.28   0.200     -.145324    .6933925
------------------------------------------------------------------------------
```

$$MSS = \sum_{t=1}^{T} (\hat{y}_t - \bar{y}_t)^2$$

$$RSS = \sum_{t=1}^{T} (y_t - \hat{y}_t)^2$$

$$TSS = MSS + RSS = \sum_{t=1}^{T} (y_t - \bar{y}_t)^2$$

$$F = \frac{MSS/(K-1)}{RSS/(T-K)}$$

$$R^2 = 1 - RSS/TSS$$

$$R^2_{adj} = 1 - \frac{RSS/(T-K)}{TSS/(T-1)}$$

$$\hat{\sigma}_{ols} = \sqrt{\frac{RSS_{ols}}{T-K}}$$

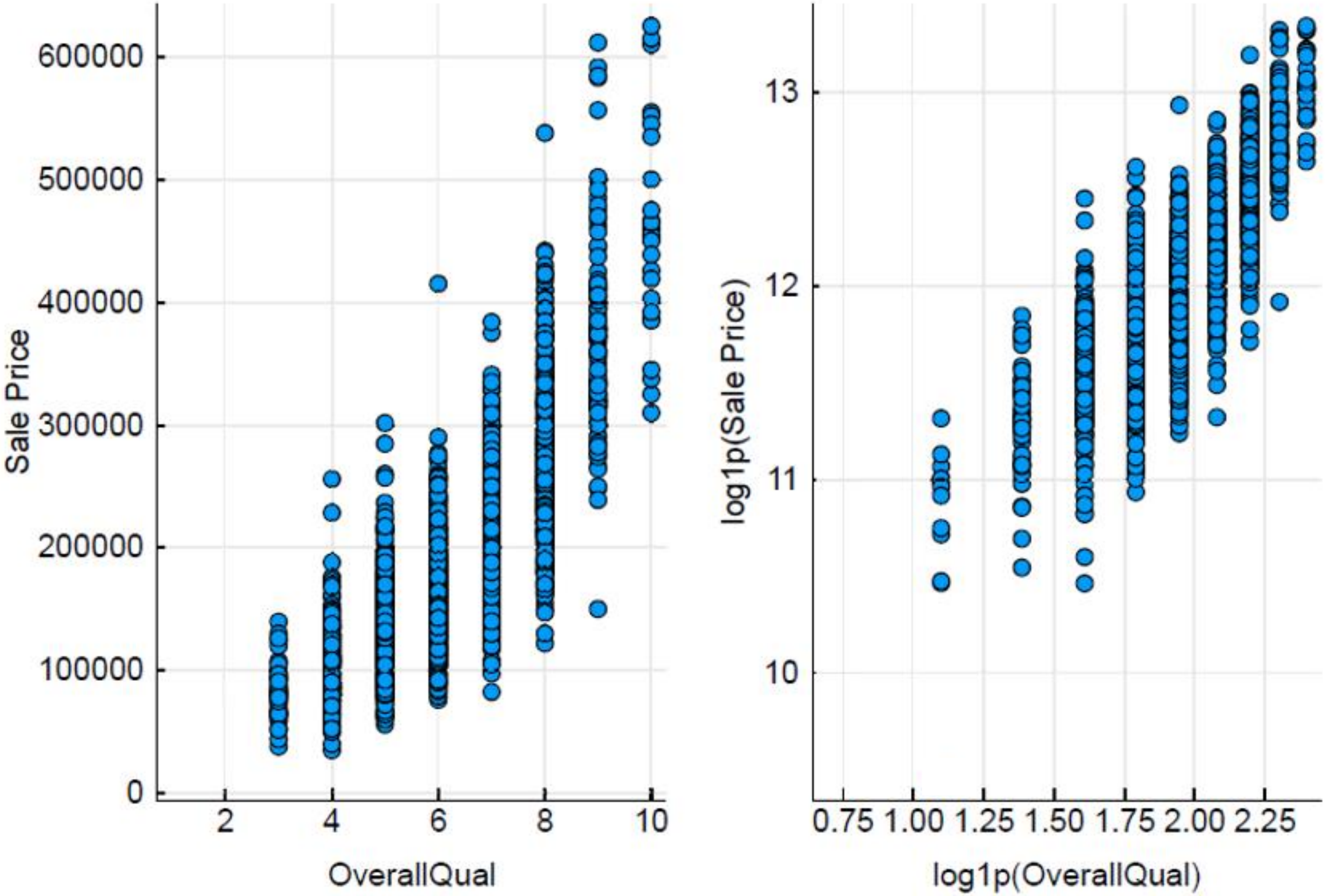
$$\hat{\beta}_{ols} = (X'X)^{-1}X'Y$$

$$t = \frac{\hat{\beta}_{i,ols} - \beta^{H0}_{i,true}}{\widehat{SE}(\hat{\beta}_{i,ols})}$$

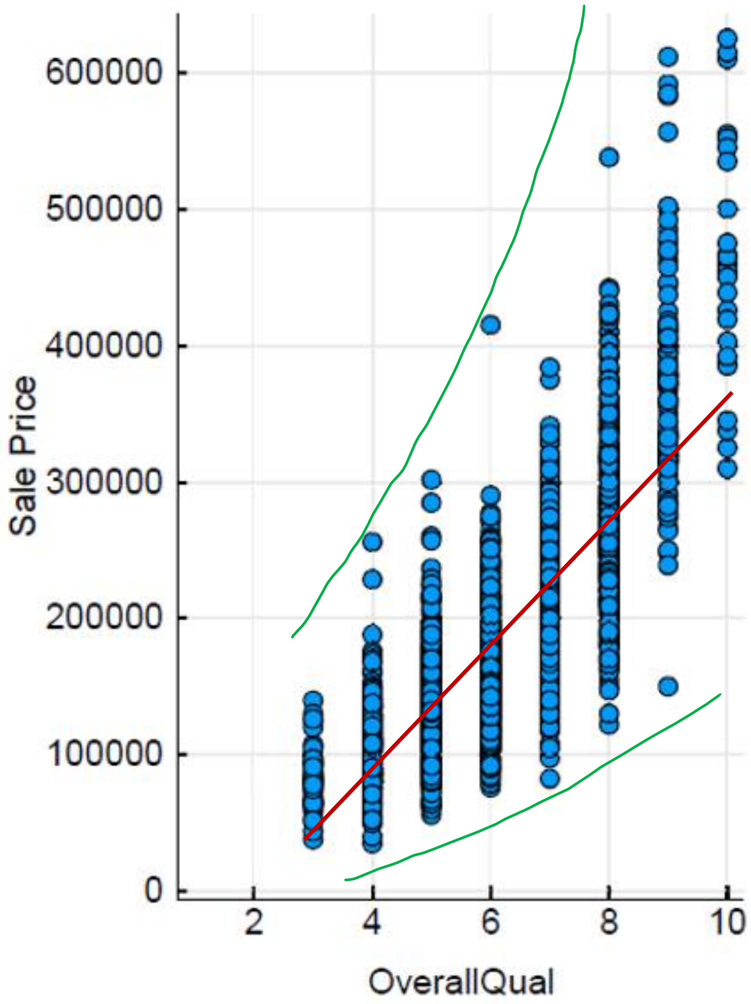
$$\widehat{SE}(\hat{\beta}_{ols}) = \text{sqrt}(\text{diag}(\widehat{VCOV}(\hat{\beta}_{ols})))$$

where $\widehat{VCOV}(\hat{\beta}_{ols}) = \hat{\sigma}^2_{ols}(X'X)^{-1}$

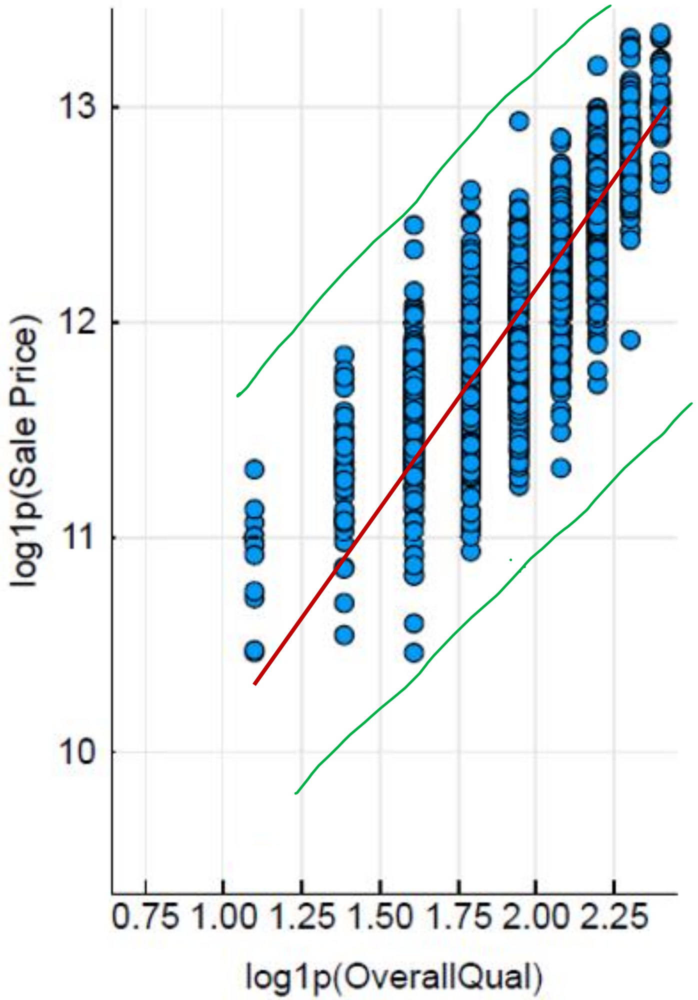
Why do we ln() variables?



Why do we ln() variables?



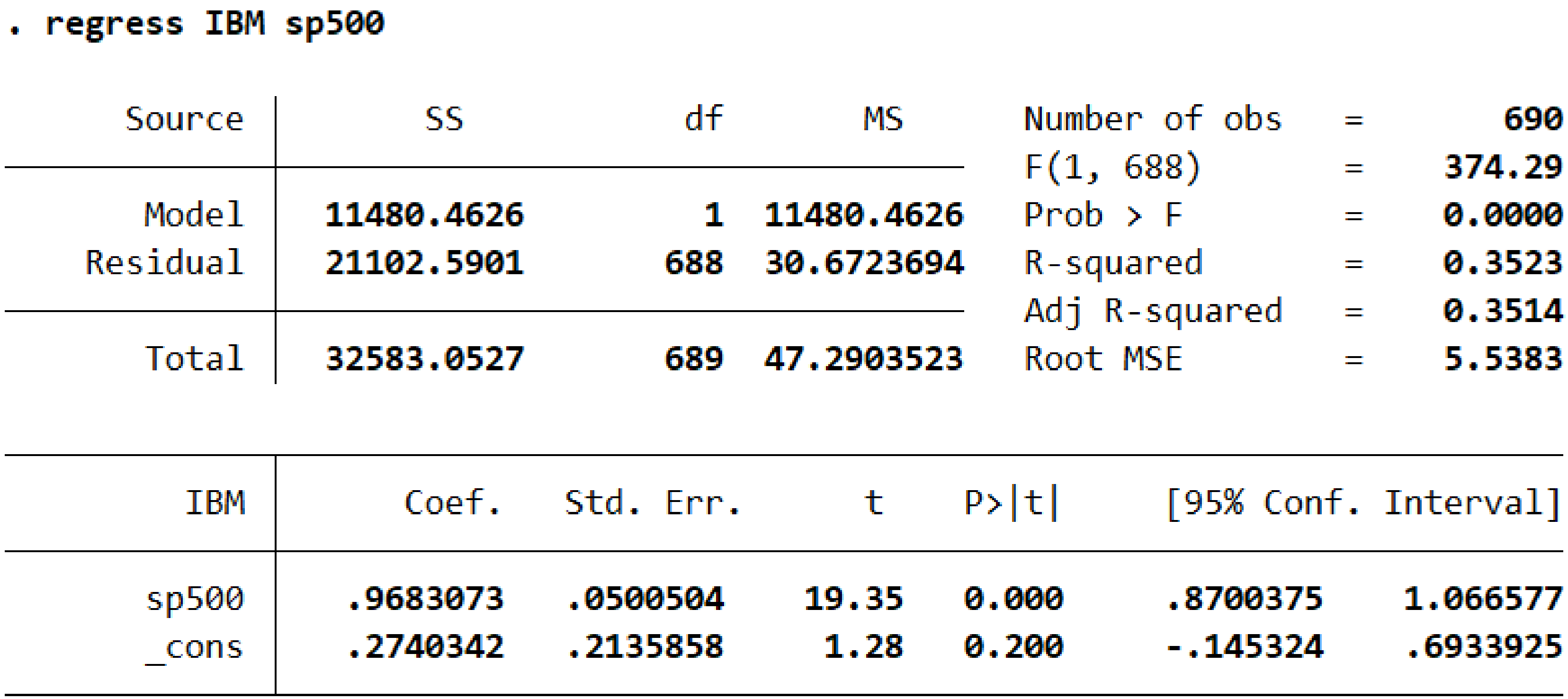
Signs of
Heteroskedastic
Error variance
(**σ increases** with X)



Error variance
more stable
(**σ constant** across X's)

OLS Regression in Stata

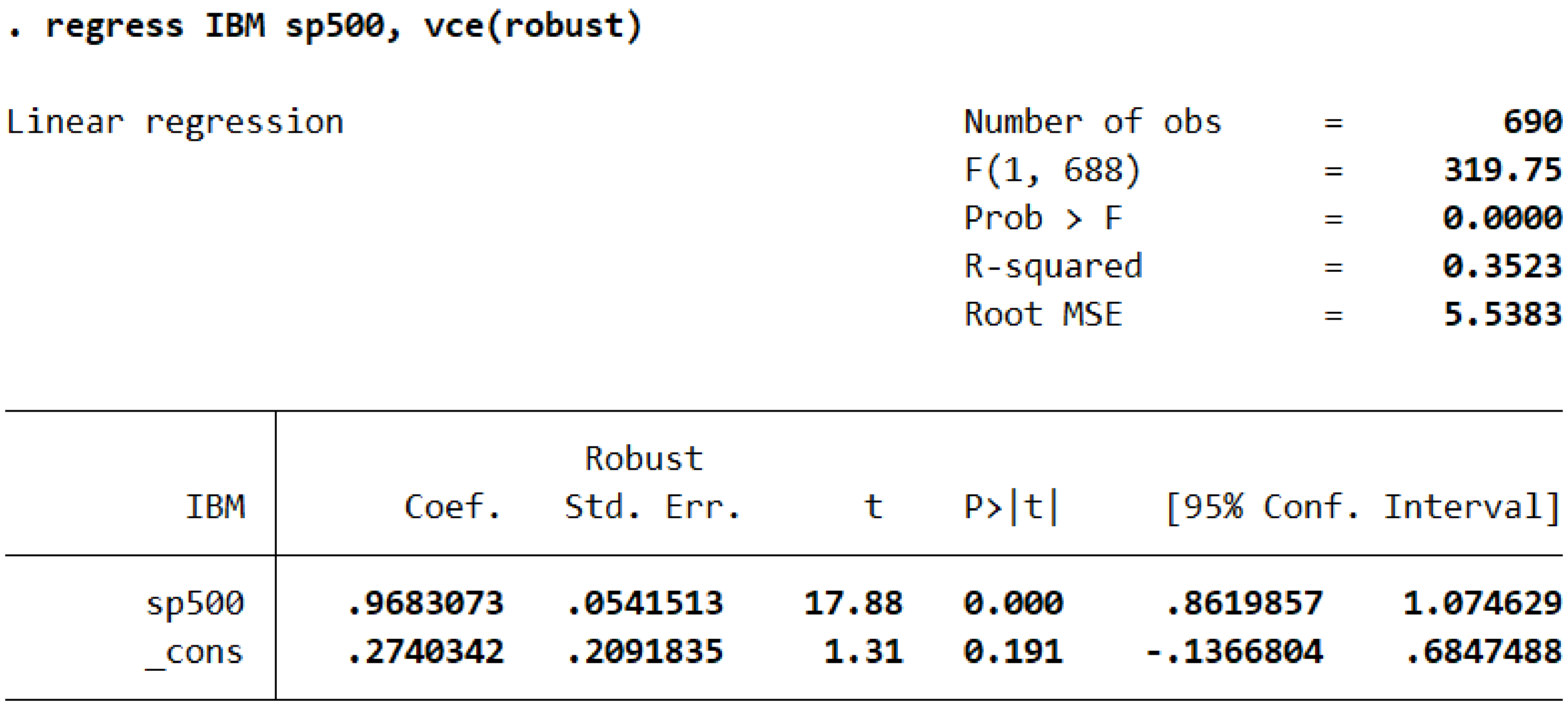
Assuming Homoskedasticity..



```
. regress IBM sp500

      Source |       SS           df       MS      Number of obs   =       690
-------------+----------------------------------   F(1, 688)       =    374.29
       Model |  11480.4626         1  11480.4626   Prob > F        =    0.0000
    Residual |  21102.5901       688  30.6723694   R-squared       =    0.3523
-------------+----------------------------------   Adj R-squared   =    0.3514
       Total |  32583.0527       689  47.2903523   Root MSE        =    5.5383

------------------------------------------------------------------------------
         IBM |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       sp500 |   .9683073   .0500504    19.35   0.000     .8700375    1.066577
       _cons |   .2740342   .2135858     1.28   0.200     -.145324    .6933925
------------------------------------------------------------------------------
```

OLS Regression in Stata

Heteroskedasticity-Robust standard errors



```
. regress IBM sp500, vce(robust)

Linear regression                               Number of obs     =        690
                                                F(1, 688)         =     319.75
                                                Prob > F          =     0.0000
                                                R-squared         =     0.3523
                                                Root MSE          =     5.5383

------------------------------------------------------------------------------
             |               Robust
         IBM |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       sp500 |   .9683073   .0541513    17.88   0.000     .8619857    1.074629
       _cons |   .2740342   .2091835     1.31   0.191    -.1366804    .6847488
------------------------------------------------------------------------------
```

OLS Regression in Stata
Post-estimation commands

After execution of any estimation command such as `regress`, `logit`, `xtreg` etc. Stata allows the direct calculation of certain quantities of interest using a set of postestimation commands. For example we can calculate:

- Fitted values (in-sample predictions)

```
predict y_hat, xb           /* y_hat = X*b_ols */
```

- Residuals

```
predict epsilon, residuals  /* e_hat = y - y_hat */
```

- Akaike's and Schwarz's Bayesian information criteria (AIC and BIC)

```
estat ic
```

Accessing Stored Results

Stata commands that produce statistical results (such as `regress`) store these results, in **r()** and **e()**. Results stored in **e()** correspond to 'estimation' output, whereas **r()** contains everything else. Note that estimation commands store results, until the next estimation command is executed.

Following command execution, we can request the list of the saved results using:
A. `ereturn list` → for the list of all **e()** results
B. `return list` → for the list of all **r()** results

Data stored in **r-** and **e-**classes results are categorized in the following categories:

- scalars (such as F-statistic, R-squared, Number of obs)
- macros (i.e. strings corresponding to the estimation performed)
- matrices (such as the beta_hat matrix, VCOV matrix)
- functions (i.e. e(sample): which is a logical indicator taking the value 1 if the observation was used in the previous estimation, 0 otherwise).

Saving and Processing Stored Results

Scalars & matrices

We can create **variables** (matrices, scalars; as opposed to columns in the data-editor) to save **e-, r-class results** and subsequently perform calculations.

- Save **an e-, r-class scalar** result in a **scalar variable**

```
scalar x1 = e(r2)     /* Save the R^2 following command regress */
```

- Print the value of the x1 variable

`scalar list x1` or `display x1`

- Print a list with all variables in memory

```
scalar list
```

- Similarly, we can save an **e-, r-class matrix/vector** result in a **matrix**

```
matrix vita = e(b)   /* Save the OLS betas vector following command regress */
```

- Print the value of the matrix vita

```
matrix list vita
```

Saving and Processing Stored Results

Local Macros

- You can save copies of **r()** and **e()** class stored results, and perform calculations using either **scalars**, or **local macros** (aka locals). Stata defines macros as *variables of Stata programs*.
- One main use of local macros is for capturing the iteration index variable when creating loops (`foreach`, `forvalues` and `while`).

```
* Using Local Macros *

summarize sp500
local mean1 = r(mean)
summarize KO
local mean2 = r(mean)
local diff = `mean1' - `mean2'
gen v1 = `diff' in 1/10
display `diff'
display "The Difference is: " `diff'
```

```
* Using Scalars *

summarize sp500
scalar m1 = r(mean)
summarize KO
scalar m2 = r(mean)
scalar df = m1 - m2
gen v2 = df in 1/10
display df
display "The Difference is: " df
```

Quick recap

Week 4

Last week we:

- Reviewed the Classical Linear Regression Model (CLRM) framework (-assumptions for OLS to be BLUE, OLS estimator derivation, and hypothesis testing).
- Saw how to estimate standard **OLS** and **Heteroskedasticity-Robust standard errors** (- using the White VCOV estimator) in Stata, using command `regress` and option `vce(robust)`.
- Explored postestimation commands to retrieve **fitted values**, **residuals**, and **information criteria** among other things.
- Saw that after we execute a command, Stata stores estimation/calculation results in r() and e(), which can then be printed using `return list` and `ereturn list`, respectively.
- Saw the differences between **matrices**, **scalars** and **local macros** and how to use each of these types of *program-variables* to store information and perform calculations.
    - `matrix x = e(b)`
    - `scalar tstat = r(mean)/sqrt(r(Var)/r(N))`
    -
      ```
      local m = r(mean)
      local var = r(Var)
      local Nobs = r(N)
      local tstat = `m'/sqrt(`var'/`Nobs')
      ```

One-liner:

```
local tstat = `r(mean)'/sqrt(`r(Var)'/`r(N)')
```

Time-Series Models

Univariate Models, ARMA(p,q)

- ARMA(p,q) or autoregressive moving-average of order (p,q), is a univariate time-series model containing the first ***p* lags** of **autocorrelations** and the first ***q* lags** of **moving averages**.

$$y_t = \alpha + \sum_{i=1}^{p} \rho_i y_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t$$

which is commonly written using **lag operator notation**:

$$\rho(L^p) y_t = \theta(L^q) \varepsilon_t$$

with

$$\rho(L^p) = 1 - \rho_1 L - \rho_2 L^2 - \cdots - \rho_p L^p$$
$$\theta(L^q) = 1 + \theta_1 L + \theta_2 L^2 + \cdots + \theta_q L^q$$

and $L^p y_t = y_{t-p}$.

Time-Series Models

Univariate Models, AR(p) and MA(q)

- AR (p): $y_t = \alpha + \sum_{i=1}^{p} \rho_i y_{t-i} + \varepsilon_t$
- MA (q): $y_t = \alpha + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t$

How can we **choose hyperparameters** p and q?

1. The parameters can be set to standard values (having as an objective to capture possible time dependencies and *leave the residuals 'well behaving')*
   e.g. set $p = q = 1$, or for quarterly data to capture a year of lags: $p = 4, q = 0$
2. Using **hyperparameter optimization techniques**, comparing the pseudo out-of-sample (POOS) forecast error (FE), using the MSE/MAE/MAD metrics:
   - K-fold Cross-Validation (CV) → separate sample into *train* & *test* sets
   - Classical fixed/expanding-window POOS forecasting **performance evaluation** exercise

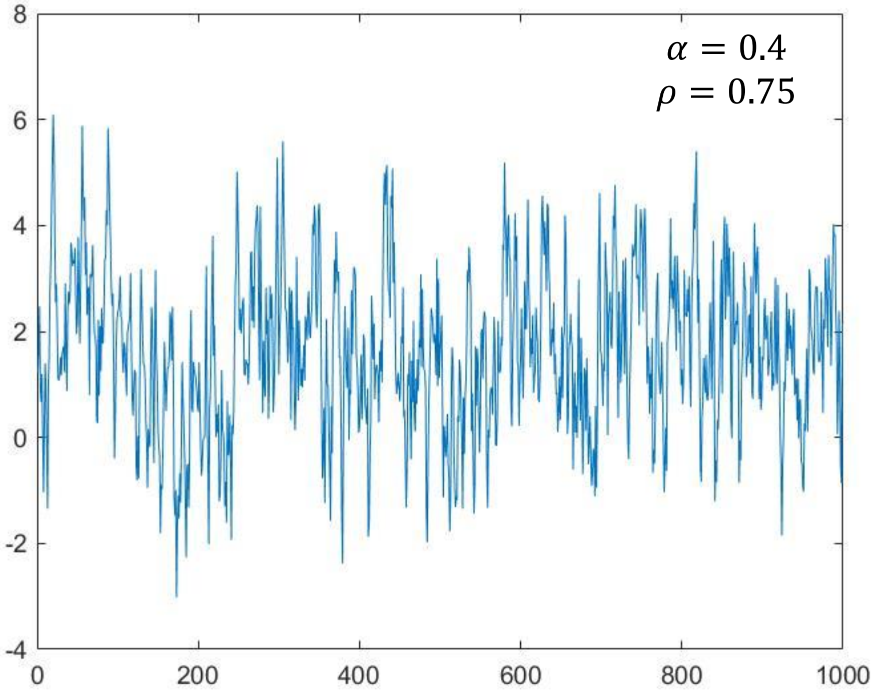
   Here the objective is to select the hyperparameters that *produce the best (oos) forecasts*.
3. Using **Information Criteria**, that capture how well the model fits the data (comparing the maximized log-likelihood) after penalizing for model complexity:
   - AIC, BIC → calculated in-sample

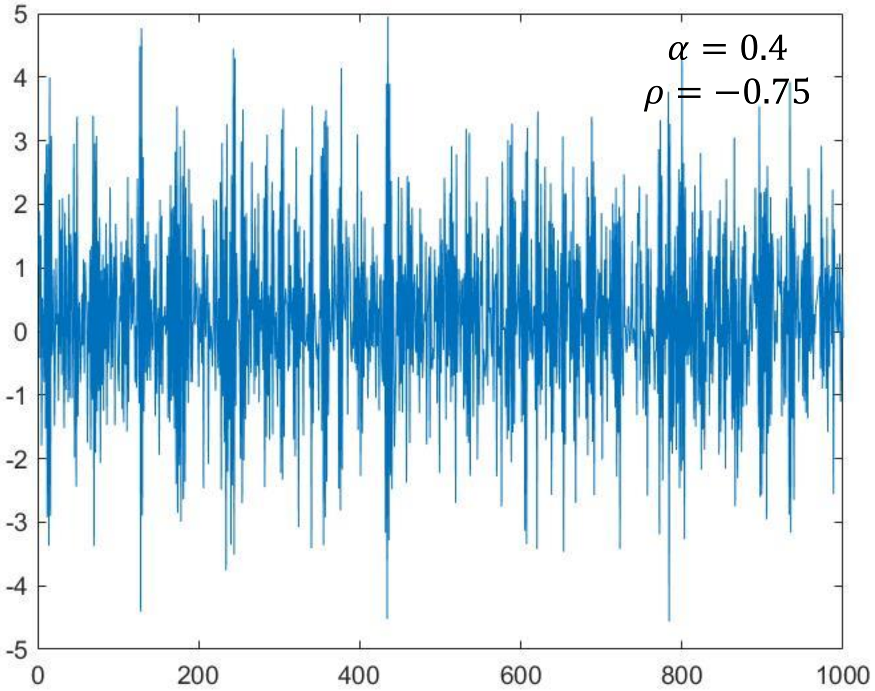
   Here the objective is to select the hyperparameters that *produce the best in-sample fit*.

Time-Series Models

Mean-reverting property of AR(p) process

- A **stationary** AR process is **mean-reverting**. This means that the series will always fluctuate around its **unconditional mean** (sometimes also called **long-term mean** or **steady state**).
- For AR(1): $y_t = \alpha + \rho y_{t-1} + \varepsilon_t$ the unconditional mean is equal to $\mathrm{E}(y_t) = \alpha/(1-\rho)$.





Time-Series Models in Stata

- In order to be able to easily define and use time-series operators (lags, leads, differences), we first need to `tsset` our dataset. By doing so we instruct Stata that we have a timeseries dataset and we define the time dimension.

  `tsset date, monthly` or simply `tsset date`

- We can then define:
  - Lags using: `L(1/p).X`
  - Leads are defined using: `F(1/p).X`
  - Differences using: `D.X`, `D2.X`
  - Combination of lags/leads & differences: `L(1/p).D.X`
- Examples:
  - `reg Y L(0/4).X`
  - `reg Y L(1 2 8).X          /* Selecting only specific lags */`
  - `reg D.cpi L(1/2).D.cpi    /* If you believe prices are I(1) */`
  - `reg D2.cpi L(1/2).D2.cpi  /* If you believe prices are I(2) */`
  - `arima D.cpi ar(1/2)`

Time-Series Models

Ljung–Box test

- The **Ljung–Box Q test** which is also referred to as **Portmanteau (or Q) test** for ***white noise***, was developed by Box and Pierce (1970) and refined by Ljung and Box (1978).
- **H0**: The data are independently distributed (i.e. the **(auto)correlations** in the population from which the sample is taken **are 0**, so that any observed correlations in the data result from randomness of the sampling process). The data are **white noise**. $H_0: \rho_1 = \rho_2 = \ \dots \ = \rho_m = 0$.
  Ha: The data are not independently distributed; they exhibit **serial correlation**; they are **autocorrelated**.
- The test statistic is:

$$\mathrm{Q} = T(T+2)\sum_{j=1}^{m}\frac{1}{T-j}\hat{\rho}_j^2 \sim \boldsymbol{\chi}^2_{(m)}$$

where m denotes the number of autocorrelations calculated (i.e. the number of lags being tested), $\hat{\rho}_j$ is the estimated sample autocorrelation for lag j, and T the sample size.

- The test is commonly used in time-series modelling (after an ARIMA, AR etc. model has been fitted), and is applied to the **residuals** (instead of the **original** series) to test the hypothesis of whether the residuals from the specification employed are **serially correlated**. This relates to CLRM assumptions **A4b** & **A5**.

Time-Series Models

Ljung–Box test

- The **autocovariance** function is defined for $|j| < T$:

$$\hat{R}(j) = \frac{1}{T}\sum_{i=1}^{T-|j|}(x_i - \bar{x})(x_{i-j} - \bar{x})$$

where $\bar{x}$ denotes the sample mean. The **autocorrelation** function is then given by:

$$\hat{\rho}_j \quad = \frac{\hat{R}(j)}{\hat{R}(0)}$$

What is a Moving Average?

- K-period simple moving average (SMA or MA) is the unweighted mean of the previous K data.

$$MA_K = \frac{\sum_{\mathrm{i}=0}^{K-1} y_{t-i}}{K}$$

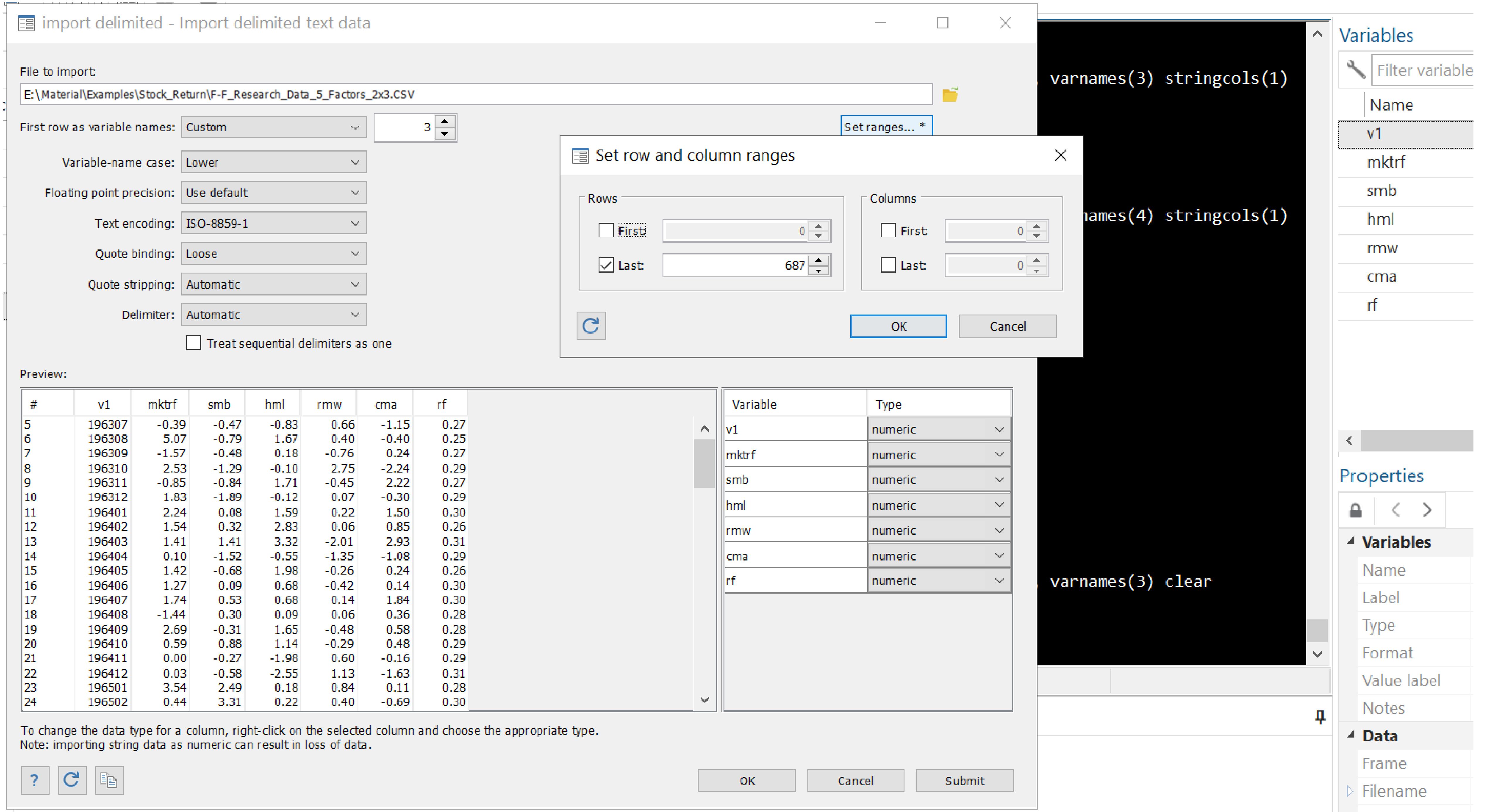
- A moving average is used to **smooth** series that are particularly volatile. It is also a common technical analysis indicator in security trading.

In class exercise (5 min):

Calculate the **6-period MA** of the S&P 500 (without generating multiple variables and then adding them. There is another storage-efficient way!). Make sure you do the required adjustment for the beginning of the sample.

Back to Stata…

Downloading the Fama-French data & merging with the CRSP data



import delimited - Import delimited text data

File to import: E:\Material\Examples\Stock_Return\F-F_Research_Data_5_Factors_2x3.CSV

First row as variable names: Custom 3

Variable-name case: Lower

Floating point precision: Use default

Text encoding: ISO-8859-1

Quote binding: Loose

Quote stripping: Automatic

Delimiter: Automatic

□ Treat sequential delimiters as one

Set row and column ranges

Rows

□ First: 0

☑ Last: 687

Columns

□ First: 0

□ Last: 0

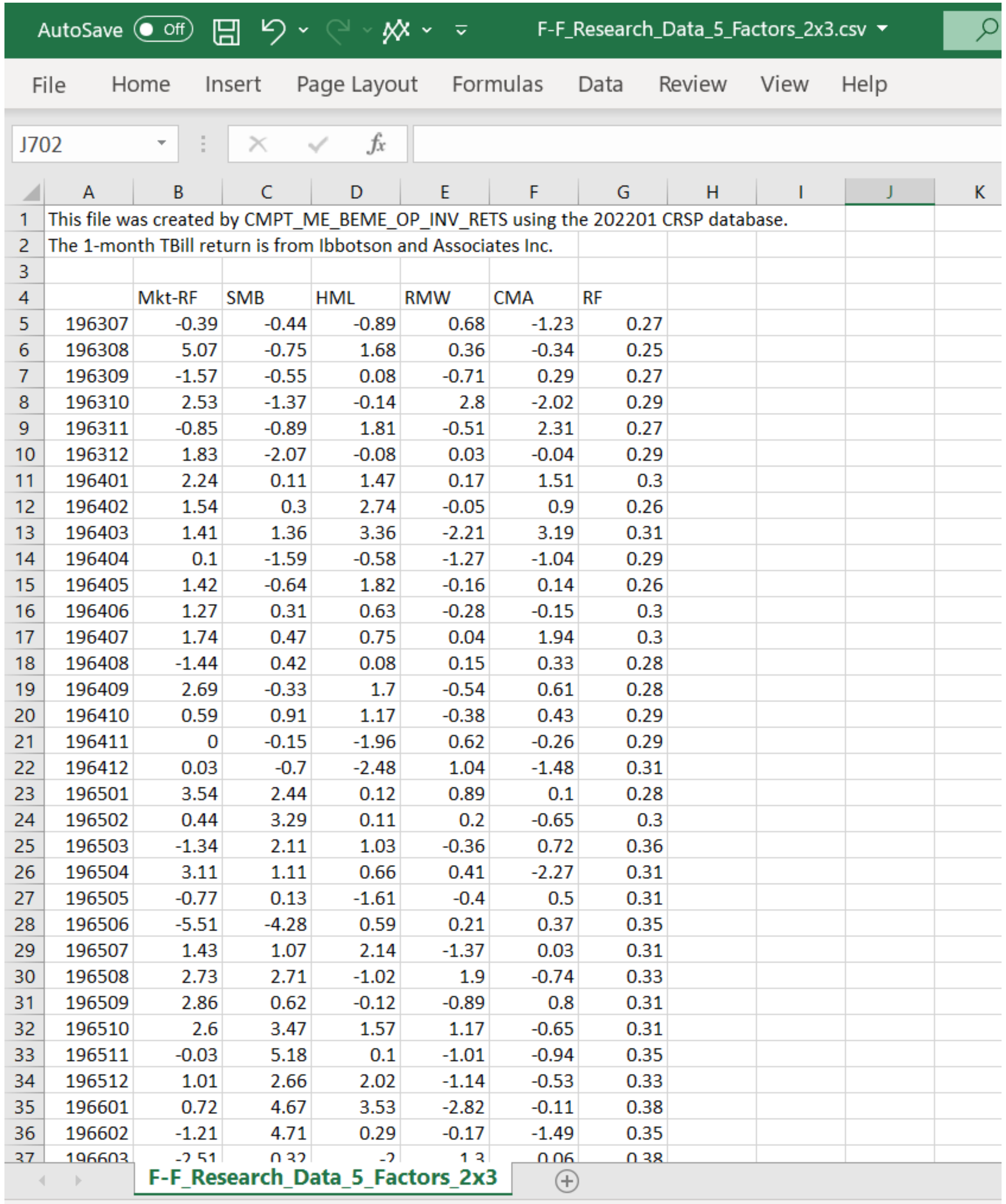
Preview:

#	v1	mktrf	smb	hml	rmw	cma	rf
5	196307	-0.39	-0.47	-0.83	0.66	-1.15	0.27
6	196308	5.07	-0.79	1.67	0.40	-0.40	0.25
7	196309	-1.57	-0.48	0.18	-0.76	0.24	0.27
8	196310	2.53	-1.29	-0.10	2.75	-2.24	0.29
9	196311	-0.85	-0.84	1.71	-0.45	2.22	0.27
10	196312	1.83	-1.89	-0.12	0.07	-0.30	0.29
11	196401	2.24	0.08	1.59	0.22	1.50	0.30
12	196402	1.54	0.32	2.83	0.06	0.85	0.26
13	196403	1.41	1.41	3.32	-2.01	2.93	0.31
14	196404	0.10	-1.52	-0.55	-1.35	-1.08	0.29
15	196405	1.42	-0.68	1.98	-0.26	0.24	0.26
16	196406	1.27	0.09	0.68	-0.42	0.14	0.30
17	196407	1.74	0.53	0.68	0.14	1.84	0.30
18	196408	-1.44	0.30	0.09	0.06	0.36	0.28
19	196409	2.69	-0.31	1.65	-0.48	0.58	0.28
20	196410	0.59	0.88	1.14	-0.29	0.48	0.29
21	196411	0.00	-0.27	-1.98	0.60	-0.16	0.29
22	196412	0.03	-0.58	-2.55	1.13	-1.63	0.31
23	196501	3.54	2.49	0.18	0.84	0.11	0.28
24	196502	0.44	3.31	0.22	0.40	-0.69	0.30

Variable	Type
v1	numeric
mktrf	numeric
smb	numeric
hml	numeric
rmw	numeric
cma	numeric
rf	numeric

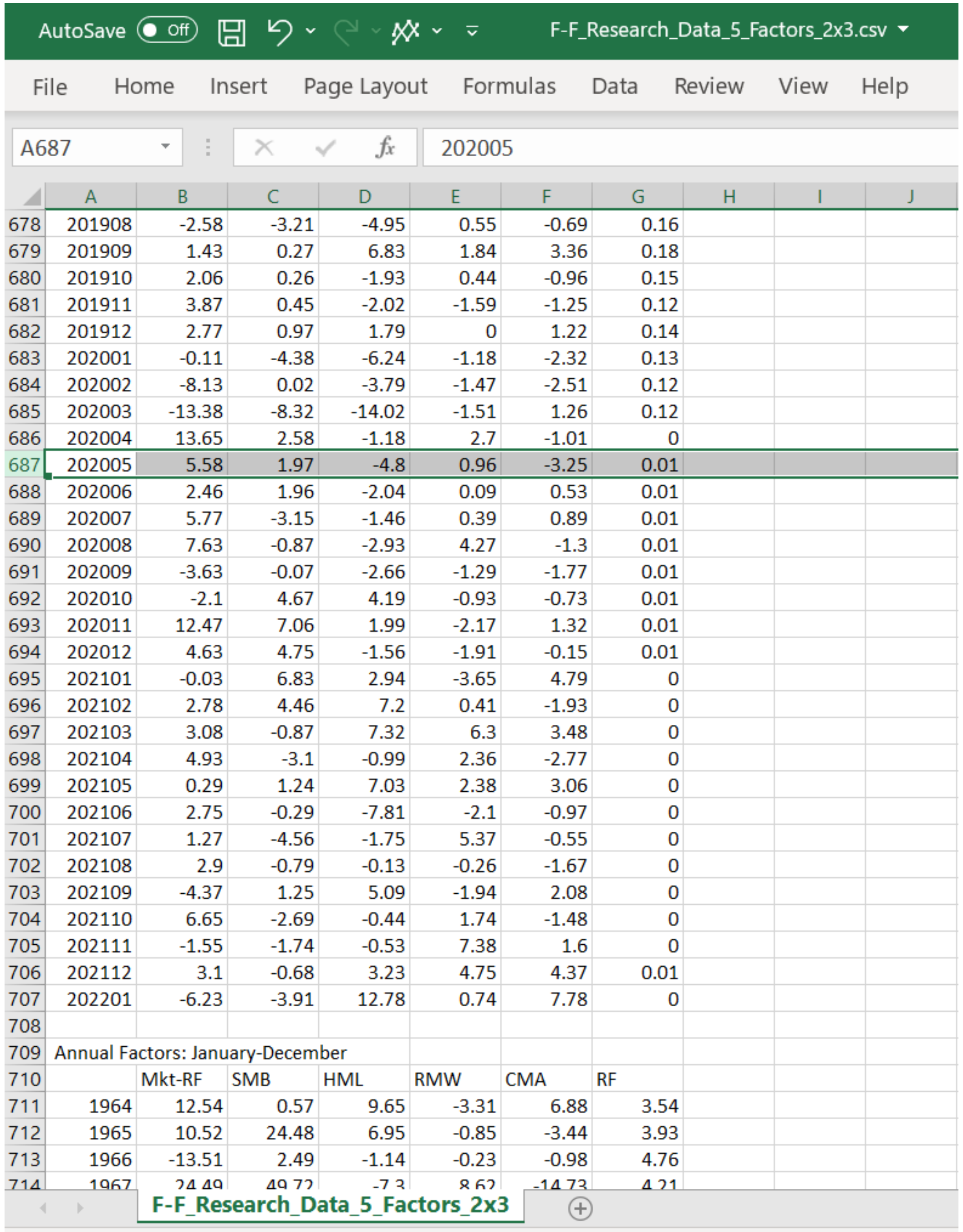
To change the data type for a column, right-click on the selected column and choose the appropriate type.
Note: importing string data as numeric can result in loss of data.

```
varnames(3) stringcols(1)
ames(4) stringcols(1)
varnames(3) clear
```


AutoSave Off
F-F_Research_Data_5_Factors_2x3.csv
File
Home
Insert
Page Layout
Formulas
Data
Review
View
Help
F-F_Research_Data_5_Factors_2x3

J702

	A	B	C	D	E	F	G
1	This file was created by CMPT_ME_BEME_OP_INV_RETS using the 202201 CRSP database.						
2	The 1-month TBill return is from Ibbotson and Associates Inc.						
3							
4		Mkt-RF	SMB	HML	RMW	CMA	RF
5	196307	-0.39	-0.44	-0.89	0.68	-1.23	0.27
6	196308	5.07	-0.75	1.68	0.36	-0.34	0.25
7	196309	-1.57	-0.55	0.08	-0.71	0.29	0.27
8	196310	2.53	-1.37	-0.14	2.8	-2.02	0.29
9	196311	-0.85	-0.89	1.81	-0.51	2.31	0.27
10	196312	1.83	-2.07	-0.08	0.03	-0.04	0.29
11	196401	2.24	0.11	1.47	0.17	1.51	0.3
12	196402	1.54	0.3	2.74	-0.05	0.9	0.26
13	196403	1.41	1.36	3.36	-2.21	3.19	0.31
14	196404	0.1	-1.59	-0.58	-1.27	-1.04	0.29
15	196405	1.42	-0.64	1.82	-0.16	0.14	0.26
16	196406	1.27	0.31	0.63	-0.28	-0.15	0.3
17	196407	1.74	0.47	0.75	0.04	1.94	0.3
18	196408	-1.44	0.42	0.08	0.15	0.33	0.28
19	196409	2.69	-0.33	1.7	-0.54	0.61	0.28
20	196410	0.59	0.91	1.17	-0.38	0.43	0.29
21	196411	0	-0.15	-1.96	0.62	-0.26	0.29
22	196412	0.03	-0.7	-2.48	1.04	-1.48	0.31
23	196501	3.54	2.44	0.12	0.89	0.1	0.28
24	196502	0.44	3.29	0.11	0.2	-0.65	0.3
25	196503	-1.34	2.11	1.03	-0.36	0.72	0.36
26	196504	3.11	1.11	0.66	0.41	-2.27	0.31
27	196505	-0.77	0.13	-1.61	-0.4	0.5	0.31
28	196506	-5.51	-4.28	0.59	0.21	0.37	0.35
29	196507	1.43	1.07	2.14	-1.37	0.03	0.31
30	196508	2.73	2.71	-1.02	1.9	-0.74	0.33
31	196509	2.86	0.62	-0.12	-0.89	0.8	0.31
32	196510	2.6	3.47	1.57	1.17	-0.65	0.31
33	196511	-0.03	5.18	0.1	-1.01	-0.94	0.35
34	196512	1.01	2.66	2.02	-1.14	-0.53	0.33
35	196601	0.72	4.67	3.53	-2.82	-0.11	0.38
36	196602	-1.21	4.71	0.29	-0.17	-1.49	0.35
37	196603	-2.51	0.32	-2	1.3	0.06	0.38

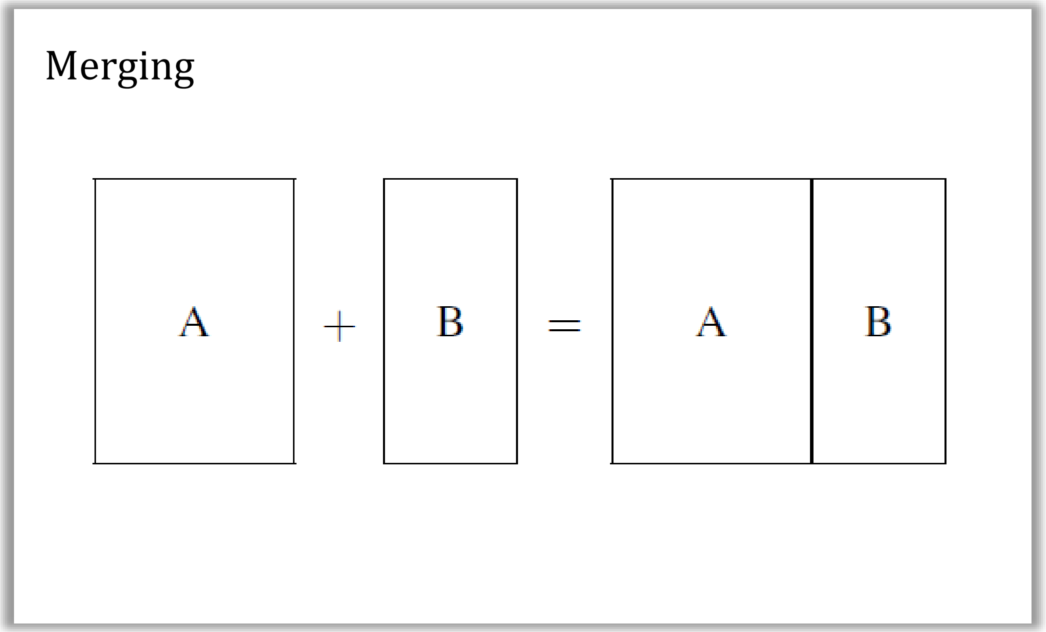

AutoSave Off
F-F_Research_Data_5_Factors_2x3.csv
File
Home
Insert
Page Layout
Formulas
Data
Review
View
Help
F-F_Research_Data_5_Factors_2x3

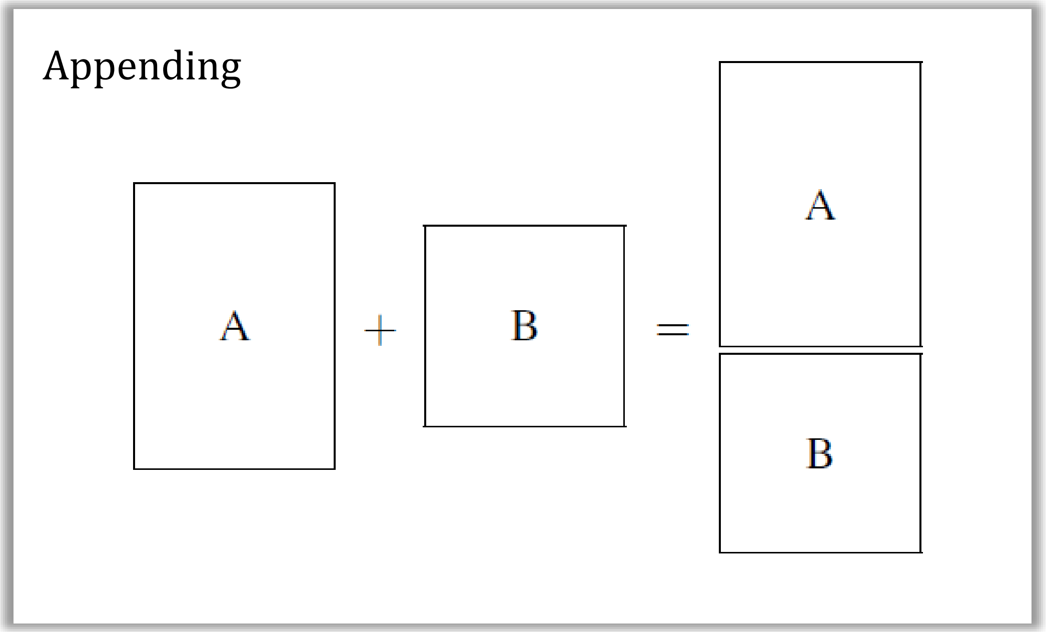
A687 202005

	A	B	C	D	E	F	G
678	201908	-2.58	-3.21	-4.95	0.55	-0.69	0.16
679	201909	1.43	0.27	6.83	1.84	3.36	0.18
680	201910	2.06	0.26	-1.93	0.44	-0.96	0.15
681	201911	3.87	0.45	-2.02	-1.59	-1.25	0.12
682	201912	2.77	0.97	1.79	0	1.22	0.14
683	202001	-0.11	-4.38	-6.24	-1.18	-2.32	0.13
684	202002	-8.13	0.02	-3.79	-1.47	-2.51	0.12
685	202003	-13.38	-8.32	-14.02	-1.51	1.26	0.12
686	202004	13.65	2.58	-1.18	2.7	-1.01	0
687	202005	5.58	1.97	-4.8	0.96	-3.25	0.01
688	202006	2.46	1.96	-2.04	0.09	0.53	0.01
689	202007	5.77	-3.15	-1.46	0.39	0.89	0.01
690	202008	7.63	-0.87	-2.93	4.27	-1.3	0.01
691	202009	-3.63	-0.07	-2.66	-1.29	-1.77	0.01
692	202010	-2.1	4.67	4.19	-0.93	-0.73	0.01
693	202011	12.47	7.06	1.99	-2.17	1.32	0.01
694	202012	4.63	4.75	-1.56	-1.91	-0.15	0.01
695	202101	-0.03	6.83	2.94	-3.65	4.79	0
696	202102	2.78	4.46	7.2	0.41	-1.93	0
697	202103	3.08	-0.87	7.32	6.3	3.48	0
698	202104	4.93	-3.1	-0.99	2.36	-2.77	0
699	202105	0.29	1.24	7.03	2.38	3.06	0
700	202106	2.75	-0.29	-7.81	-2.1	-0.97	0
701	202107	1.27	-4.56	-1.75	5.37	-0.55	0
702	202108	2.9	-0.79	-0.13	-0.26	-1.67	0
703	202109	-4.37	1.25	5.09	-1.94	2.08	0
704	202110	6.65	-2.69	-0.44	1.74	-1.48	0
705	202111	-1.55	-1.74	-0.53	7.38	1.6	0
706	202112	3.1	-0.68	3.23	4.75	4.37	0.01
707	202201	-6.23	-3.91	12.78	0.74	7.78	0
708							
709	Annual Factors: January-December						
710		Mkt-RF	SMB	HML	RMW	CMA	RF
711	1964	12.54	0.57	9.65	-3.31	6.88	3.54
712	1965	10.52	24.48	6.95	-0.85	-3.44	3.93
713	1966	-13.51	2.49	-1.14	-0.23	-0.98	4.76
714	1967	24.49	49.72	-7.3	8.62	-14.73	4.21

Combining Datasets

- Combining two or more datasets **horizontally** is called **merging,** whereas **vertical** concatenation of two or more datasets is commonly referred to as **appending**.





Merging Datasets

In Stata's jargon, when merging, there are 2 types of datasets:
1. **'master' dataset**, which is the one you currently have in memory (i.e. the one currently open), and
2. **'using' dataset**, which is the one (or many) you have saved in your root directory (-say MYdta.dta).

The main types of merging in Stata are:

1. **One-to-one merge** (1:1): When the key variable 'id' uniquely identifies each row in both datasets

```
merge 1:1 id using MYdta
```

2. **Many-to-one merge** (m:1): When multiple rows from 'master' correspond to single row from 'using' (e.g. 'master' is students.dta, and 'using' is universties.dta)

```
merge m:1 id using MYdta
```

3. **One-to-many merge**(1:m): When each row from master corresponds to multiple rows from 'using’ (e.g. 'master' is universities.dta, and 'using' is students.dta)

```
merge 1:m id using MYdta
```

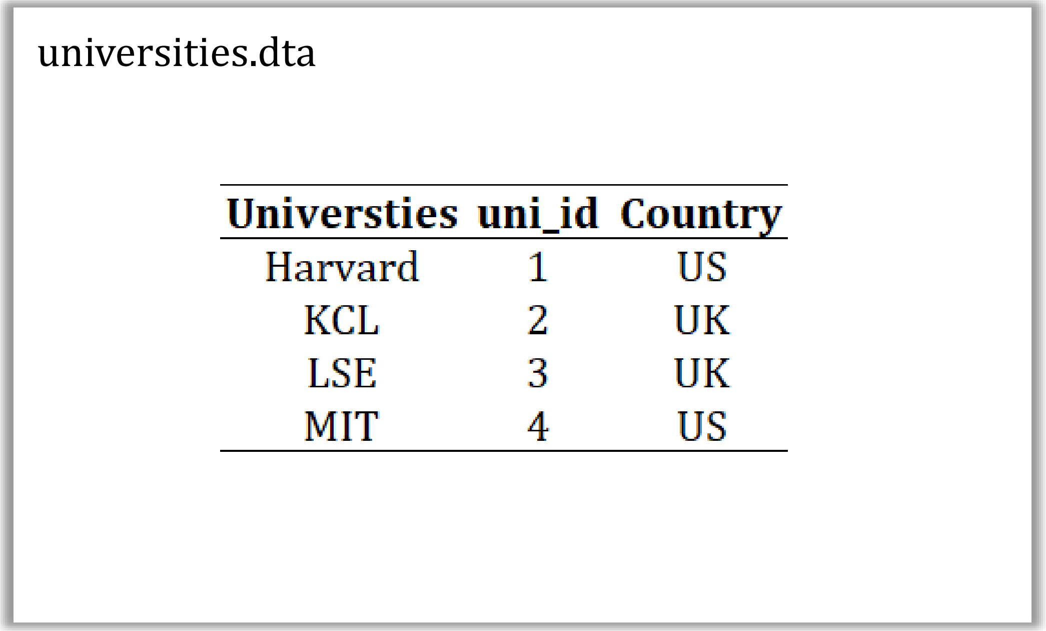
Merging Datasets

Any datasets that can be merged using an **m:1 merge** may be merged using a **1:m merge** by reversing the roles of the **master** and **using** datasets. It is just a matter of which dataset is currently in memory and which is not.

Once merge is executed, a new variable called '**_merge**' will be created telling us from which of the two datasets, each 'id' observation (and subsequently, each row), came from. The values that _merge can take, are the following:

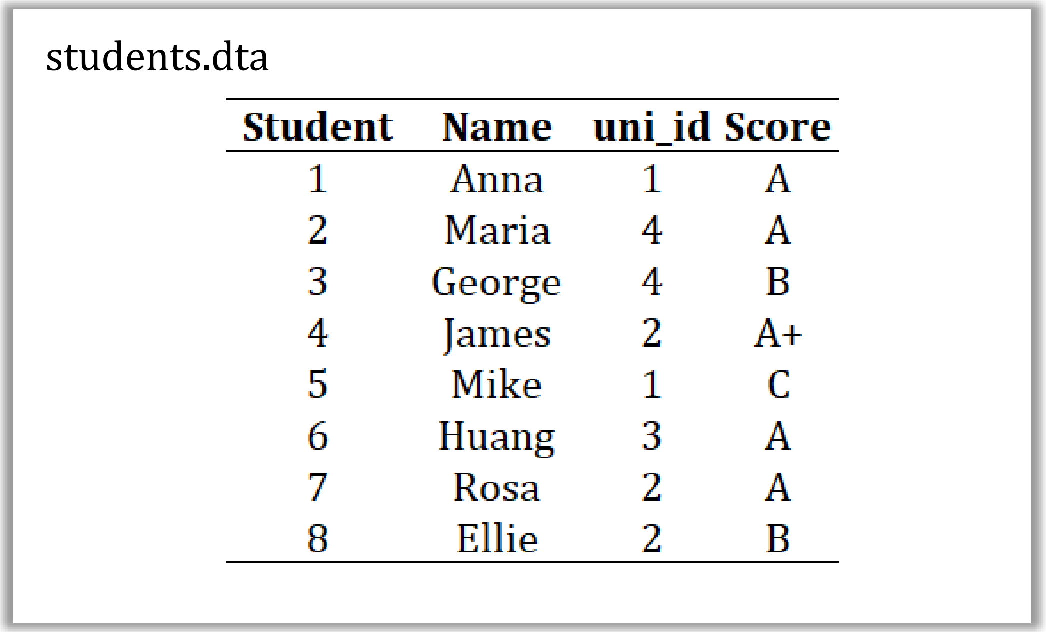
- _merge=1 → row (or id) originally appeared in '**master**' dataset only
- _merge=2 → row (or id) originally appeared in '**using**' dataset only
- _merge=3 → row (or id) originally appeared in both '**master**' & '**using**' datasets

Merging Datasets



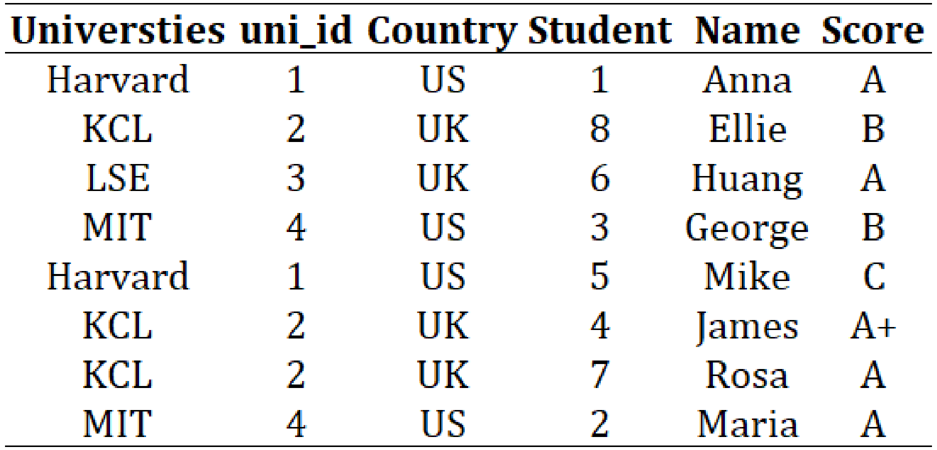
universities.dta

Universties	uni_id	Country
Harvard	1	US
KCL	2	UK
LSE	3	UK
MIT	4	US



students.dta

Student	Name	uni_id	Score
1	Anna	1	A
2	Maria	4	A
3	George	4	B
4	James	2	A+
5	Mike	1	C
6	Huang	3	A
7	Rosa	2	A
8	Ellie	2	B



Universties	uni_id	Country	Student	Name	Score
Harvard	1	US	1	Anna	A
KCL	2	UK	8	Ellie	B
LSE	3	UK	6	Huang	A
MIT	4	US	3	George	B
Harvard	1	US	5	Mike	C
KCL	2	UK	4	James	A+
KCL	2	UK	7	Rosa	A
MIT	4	US	2	Maria	A

```
merge m:1 uni_id using universities
merge 1:m uni_id using students
```

References

- G. M. Ljung; G. E. P. Box (1978). "On a Measure of a Lack of Fit in Time Series Models". Biometrika. 65 (2): 297–303.
- Box, G. E. P.; Pierce, D. A. (1970). "Distribution of Residual Autocorrelations in Autoregressive-Integrated Moving Average Time Series Models". Journal of the American Statistical Association. 65 (332): 1509–1526.

Quick recap
Week 5

Last week we:

- Saw how to `tsset` our data in order to be able to easily apply time-series operators (lags, leads, differences) using L, F, D in our syntaxes. We estimated AR(p) and ARIMA(p,D,0) models via both OLS and MLE, using commands `regress` and `arima`, respectively.
- Saw how to test for serial correlation in the residuals using the **Ljung–Box test** (`wntestq`).
- Programmatically downloaded, unzipped and imported the FF5 monthly data. Then, we **merged** them with the stock return timeseries from CRSP.
- Saw how to **append** datasets and how to draw observations from the uniform distribution.
- Downloaded and installed 3rd-party functions using `ssc install`.
- Created a **categorical** variable with 5 categories, and converted it to a set of (0/1) **dummy** variables using commands:

  ```
  • xi i.Xcateg                         /* omits the dummy for the 1st group */
  • tabulate Xcateg, generate(Dvars)    /* creates dummies for ALL groups */
  ```

- Saw how to automatically incorporate sets of **dummy variables** and **interactions** into any regression command (without having to create the dummies and/or interaction terms beforehand):

  ```
  • xi: reg Y X1 i.Xcateg     /* dummies for Xcateg */
  • xi: reg Y X1 i.Xcateg*Z   /* dummies for Xcateg and interactions with Z */
  ```

Asset Pricing Framework

CAPM & Factor Models

Asset-Pricing Models

CAPM & Factor Models

Asset-pricing models try to explain the timeseries of portfolio's p excess returns (i.e. in excess of the risk-free rate) using the contemporaneous common market-risk factor, as well as other asset-pricing risk factors pertaining to identified long-lasting asset-pricing anomalies (i.e. inefficiencies, in the EMH sense).

(1) Standard Capital Asset Pricing Model (**CAPM**)

$$r_{p,t} - r_{f,t} = \alpha_p + \beta_{p,1}(r_{m,t} - r_{f,t}) + \epsilon_{p,t}$$

(2) Asset Pricing Factor Models: The **three-**, **four-**, and the two **five-factor** models

$$r_{p,t} - r_{f,t} = \alpha_p + \beta_{p,1}(r_{m,t} - r_{f,t}) + \beta_{p,2} SMB_t + \beta_{p,3} HML_t + \epsilon_{p,t}$$

$$r_{p,t} - r_{f,t} = \alpha_p + \beta_{p,1}(r_{m,t} - r_{f,t}) + \beta_{p,2} SMB_t + \beta_{p,3} HML_t + \beta_{p,4} UMD_t + \epsilon_{p,t}$$

$$r_{p,t} - r_{f,t} = \alpha_p + \beta_{p,1}(r_{m,t} - r_{f,t}) + \beta_{p,2} SMB_t + \beta_{p,3} HML_t + \beta_{p,4} UMD_t + \beta_{p,5} LIQ_t + \epsilon_{p,t}$$

$$r_{p,t} - r_{f,t} = \alpha_p + \beta_{p,1}(r_{m,t} - r_{f,t}) + \beta_{p,2} SMB_t + \beta_{p,3} HML_t + \beta_{p,4} RMW_t + \beta_{p,5} CMA_t + \epsilon_{p,t}$$

Asset-Pricing Models

Factor Explanation and Literature

Fama/French (1993)

- **Size:** Small company stocks outperform large
  - Measured as: small-minus-big (SMB)
- **Book-to-market:** ‘Value’ stocks (high BE/ME) outperform ‘growth’ stocks (low BE/ME)
  - Measured as: high-minus-low (HML)

Carhart (1997)

- **Momentum:** Stocks that have been doing well tend to outperform those that have been doing poorly
  - Measured as: up-minus-down (UMD)

Pastor/ Stambaugh (2003)

- **Liquidity:** Stocks sensitive to liquidity shocks outperform those that are less sensitive
  - Measured as: traded liquidity factor (LIQ)

Fama/French (2015)

- **Profitability:** Stocks of highly profitable companies outperform those of less profitable companies
  - Measured as: robust-minus-weak (RMW)
- **Investment patterns:** Firms with conservative investment portfolio outperform those with aggressive investment portfolio
  - Measured as: conservative-minus-aggressive (CMA)

Asset-Pricing Models

Portfolio Return Decomposition

Asset pricing models can be used to **evaluate the performance** of portfolio p**.** The **constant** $\hat{\alpha}_p$ of the timeseries OLS regression of any of the above-mentioned factor models provides a measure of **risk-adjusted returns**, in the sense that it is the ‘abnormal return’ after accounting for the portfolio’s realized exposures to market, size, momentum, liquidity, operating profitability, and investment aggressiveness factors.

Statistically significant ***alphas*** suggest that there is a component in the excess returns of portfolio p, that cannot be explained by the standard risk factors, and that component persist across the entire period covered in our sample.

Decomposition of (excess) returns using the Fama-French (2015) 5-factor model:

$$\underbrace{r_{p,t} - r_{f,t}}_{\text{Portfolio return}} = \underbrace{\alpha_p}_{\text{Abnormal return}} + \underbrace{\beta_{p,1}(r_{m,t} - r_{f,t})}_{\substack{\text{Component attributed to} \\ \text{overall market movement} \\ \text{(i.e. market risk)}}} + \underbrace{\beta_{p,2}SMB_t + \beta_{p,3}HML_t + \beta_{p,4}RMW_t + \beta_{p,5}CMA_t}_{\substack{\text{Component attributed to} \\ \text{the primary asset-pricing} \\ \text{anomalies}}} + \underbrace{\epsilon_{p,t}}_{\text{Residual return}}$$

Asset-Pricing Models

The Minimum Variance Optimal Hedge Portfolio

Assume we estimate the CAPM regression for a portfolio p that follows our proprietary investment strategy

$$r_{p,t} - r_{f,t} = \alpha_p + \beta_p(r_{m,t} - r_{f,t}) + \epsilon_{p,t}$$

We then use **<u>CAPM</u> to construct the Minimum Variance Optimal Hedge (MVOH) portfolio** (by shorting the S&P500) leaving us with just the alpha.

$$(r_{p,t} - r_{f,t}) - \hat{\beta}_p(r_{m,t} - r_{f,t}) = \hat{\alpha}_p$$

However, even though by construction the market loading of the MVOH portfolio ($\hat{\beta}_{\text{MVOH}}$) is set to zero, we are still exposed to the various remaining systematic risk factors. In other words, when you hold the MVOH portfolio, you hold the alpha, but you also hold the various risk-factors (except the market risk factor).

To extract the alpha, **construct the MVOH portfolio using the <u>factor models</u>** instead of the CAPM.

$$r_{p,t} - r_{f,t} = \alpha_p + \beta_{p,1}(r_{m,t} - r_{f,t}) + \beta_{p,2}SMB_t + \beta_{p,3}HML_t + \beta_{p,4}UMD_t + \beta_{p,5}LIQ_t + \epsilon_{p,t}$$

$$(r_{p,t} - r_{f,t}) - \left(\hat{\beta}_p(r_{m,t} - r_{f,t}) + \beta_{p,2}SMB_t + \beta_{p,3}HML_t + \beta_{p,4}UMD_t + \beta_{p,5}LIQ_t\right) = \hat{\alpha}_p$$

We now only hold the alpha.

Annualizing & Compounding Returns

A Note on **annualizing**, **compounding** and **averaging** returns (and growth rates). When having monthly:

1. **cc/log returns** $r_t=\ln(P_t/P_{t-1})$
   - Annualization: $12*100*r_t$
   - 12-month Compounding: $100*\ln(P_t/P_{t-12}) = 100*\ln(P_t/P_{t-1} \dots P_{t-11}/P_{t-12}) = [r_t + r_{t-1} + \dots + r_{t-11}]*100$
   - Averaging (arithmetic-average): $100*(r_t + r_{t-1} + \dots + r_{t-11})/12$
2. **simple returns** $r_t=(P_t-P_{t-1})/P_{t-1}$
   - Annualization: $((r_t + 1)^{12} - 1)*100$
   - 12-month Compounding: $100*(P_t - P_{t-12})/P_{t-12} = [(r_t + 1)(r_{t-1} + 1) \dots (r_{t-11} + 1) - 1]*100$
   - Averaging (geometric-average): $[(r_t + 1)(r_{t-1} + 1) \dots (r_{t-11} + 1)]^{1/12} - 1]*100$

Annualizing & Compounding Returns

Denote log-returns using r_t=ln(P_t/P_{t-1}) and simple-returns by s_t=(P_t-P_{t-1})/P_{t-1}.

- Since with **log-returns**, compounding/accumulating over periods is simply done by summing (r_t + r_{t-1} + …), it becomes a lot easier to work out the **distributions** of compounding returns. For example, assuming the **daily log-return** distribution $r_t \sim N(\mu, \sigma^2)$ then the **annual log-return** distribution is $r_t(252) = r_t + r_{t+1} + \cdots + r_{t+251} \sim N(252\mu, 252\sigma^2)$. Hence the **annualized volatility** is calculated as $\sqrt{252}\sigma$.
- We use log-returns because they simplify calculations but remember to convert figures back to simple returns, because investors realize the 'simple returns' not the 'log returns'. To convert from log- to simple-returns: exp(r_t)-1.
- Similarly, if you have a time-series of simple-returns and you want to convert into log-returns for easier calculations (when compounding, averaging etc), you can do so by: ln(s_t+1).
- When having simple-returns, <u>never</u> *multiply* the (daily, monthly, quarterly) *simple-returns* by a *constant* (252/12/4) to **annualize**. It's wrong! Instead, use (2) from earlier slide.

Annualizing & Compounding Returns

Relation between log-returns (r_t) and simple-returns (s_t):

$r_t = \ln(s_t+1)$.

- Using the *Taylor approximation*, for small values of s_t we have that:

$\ln(s_t+1) \simeq s_t$

- Therefore, for small s_t values we can assume that:

$r_t \simeq s_t$

Annualizing & Compounding Returns

A nice illustrative example for understanding why one should NEVER:

- take the **arithmetic average** of simple-returns, and/or
- compound simple-returns **by summing,**

is the following. Assume that you have a portfolio that gives you the following (actual/simple) returns over 8 consecutive months: -0.5 , 1, -0.5 , 1, -0.5 , 1, -0.5 , 1. This means that the portfolio's value started, say, at $100, and after 8 months its value is still the same, $100.

Conceptually, given these figure we would expect that the **average 8-period return** should be 0%, and the **compound/cumulative 8-period return** should also be 0%. However, if you:

- take the arithmetic average of the simple-returns it will give you a 25% **average return**,
- compound simple-returns by summing them you will get an **8-month cumulative return** of 200%.

Both calculations are nonsensical as illustrated by this example.

Annualizing & Compounding Returns

- When we **annualize** simple-returns, essentially, we go from a **high-frequency** return (daily/monthly/quarterly) to the equivalent **low-frequency** return (yearly).
- In business valuation we often come across the concept of **CAGR** (**compound average growth rate**). **CAGR** is the opposite of **annualization**, in the sense that we go from a **low-frequency** return (e.g. 3- or 5-years return) to a **high-frequency** return (-the equivalent annual return).

- Example:

In 1964 the world GDP was $2 trillion. In 2014 the global economy had increased to $79 trillion. What is the compound annual growth rate of the global economy over these 50 years?

$$\text{CAGR} = \left(\frac{GDP_{2014}}{GDP_{1964}}\right)^{\frac{1}{2014-1964}} - 1 = \left(\frac{79}{2}\right)^{\frac{1}{50}} - 1 = 7.6\%$$

References

- Carhart, M. (1997), "On persistence in mutual fund performance", *Journal of Finance* 52, 57-82.
- Fama, E.F. and French, K.R. (1993), "Common risk factors in the returns on stocks and bonds", *Journal of Financial Economics* 33, 3-56.
- Fama E.F., French, K. R. (2015), "A five-factor asset pricing model", Journal of Financial Economics 116(1): 1-22
- Pastor L., and Stambaugh R. (2003), "Liquidity risk and expected stock returns", Journal of Political Economics, 111, pp. 642-685.

Quick recap

Week 6

Last week we:

- Reviewed the Asset Pricing framework, and listed the risk-factor portfolios (SMB, HML, UMD, LIQ, RMW, and CMA) that correspond to the primary asset-pricing anomalies identified by the literature.
- Estimated 4 asset pricing regression models (CAPM, Fama-French 5-factor model and 2 extensions) and we **stored** the results using `estimates store`
- Saw 2 commands to **print** formatted regression output tables in the **Results Window**:
  - `estimates table`
  - `estout`
- Explored 2 commands to **export** regression tables & selected output **in Excel**:
  - `outreg2`
  - `putexcel`

NB: Regression output first needs to be stored using command `estimates store` before using any of the following commands to combine all the models in a single table: `estimates table`, `estout`, `outreg2`

- Saw how to plot various types of graphs: Line charts, Scatter plots, Histograms (with Normal density and/or fitted Kernel density),
- Saw how to interactively change graphs using the '**Graph Editor**': File > Start Graph Editor

Iterative Estimation

with Fixed & Expanding Windows

Iterative Regressions

Rolling & Recursive

Suppose we have a timeseries dataset with 100 observations and we want to iteratively estimate our OLS regression model.

A. **Fixed-window regression** (aka **rolling-window**)

Regress Y on X using periods 1-20, store the regression coefficients (_b), then run the regression using periods 2-21, and so on, finishing with a regression using periods 81-100 (the last 20 obs). Sample size remains FIXED at each iteration.

```
rolling _b, window(20): regress Y X
```

B. **Expanding-window regression** (aka **recursive estimation**)

Regress Y on X using observations 1-20, store the coefficients, run the regression using observations 1-21, observations 1-22, and so on, finishing with a regression using all 100 observations. In other words, at each iteration we ADD (and therefore, sample size INCREASES by) 1 observation.

```
rolling _b, window(20) recursive: regress Y X
```

NOTE: After `rolling` has finished running, the data in your data editor will have been replaced with the resulted estimated rolling parameters & statistics.

‘for’ and ‘while’ Loops in Stata

Example 1: Use for- and while-loops and print the iteration index variables (at each iteration) as well as other own-defined local macros.

Example 2: Monte Carlo simulation - Generate 10 variables with 30 obs each, drawn from the Standard Normal distribution. Then, create a second loop that changes the label of every other variable created.

Example 3: Loop over a number of different variables (in our data editor) and create a set of new variables containing the (period-specific) average.

Example 4: Estimate the 5-factor Fama-French model and store the (exponentiated) beta coefficients in a new variable called ‘h_0’, inside the data editor. Store the (alpha) constant in the 1st row of ‘h_0’.

Example 5: Estimate AR(p) for p=1,…,5 for the Coca Cola (KO) return timeseries. Retrieve and save in an excel file (LOOPputexcel.xlsx), the following statistics: F-test and p-value, Adj. R2, Log-Likelihood, AIC, BIC, # of Obs, the beta, the S.E. and the p-value of the 1st autoregressive lag.

Example 6: Use 'Infinite’ while-loop, until condition is met and `continue, break` is reached.

Quick recap

Week 7

Last week we:

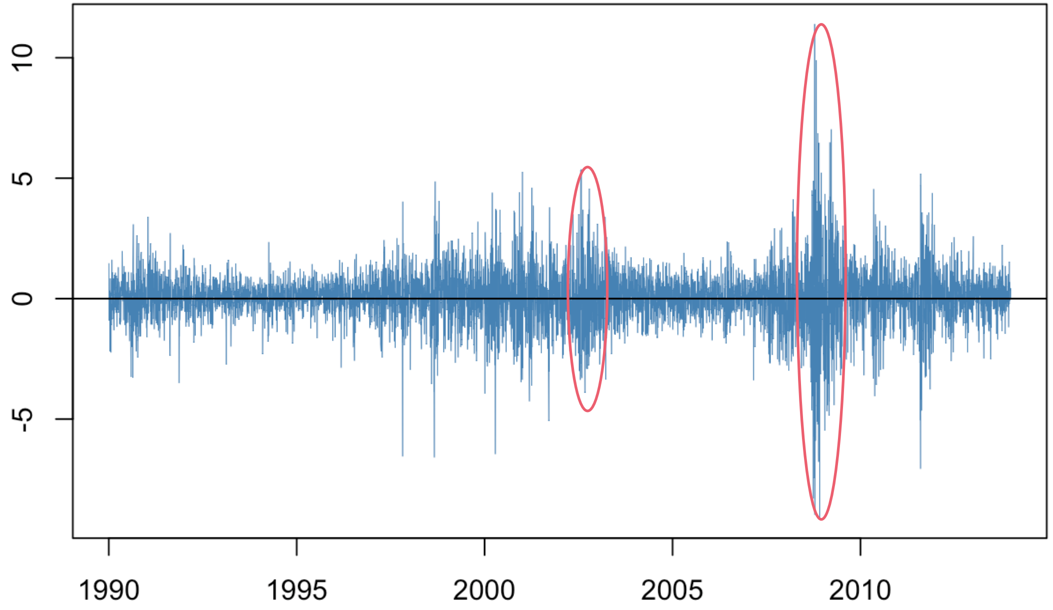
- Saw what **Iterative/Rolling estimation** is, and where it can be useful (- Backtesting/Evaluating competing predictive models, whether the objective is to predict a timeseries of returns or economic activity such as GDP or inflation etc).
- Saw how to implement different versions of rolling estimation in Stata. Specifically:
  - Fixed-window estimation (aka rolling-window) → sample size remains the same
  - Expanding-window estimation (aka recursive estimation) → sample size increases by 1
- Saw how to implement **loops** in Stata. Specifically:
  - FOR-loops:
    - `forvalues` to loop over consecutive values (e.g. `i = 1(2)10`)
    - `foreach` to loop over lists of variables (in the data editor), and non-consecutive numbers
  - WHILE-loops:
    - Standard while-loops: `` while `j' <=100 {..} ``.
    - Infinite while-loops: `while (1) {..}` together with `continue, break` to terminate

Conditional Variance Models

ARCH, GARCH and Extensions

Volatility Clustering & Time-varying Sample Moments

- Daily returns, for most stocks and other assets, fluctuate around zero. However, return *volatility* (i.e. the *standard deviation* of returns) is not constant across time, as most linear regression models would assume. In periods of economic unrest, volatility is significantly higher (and persists). This is know as ***volatility clustering***.
- Both a time-series graph and a **rolling-variance** procedure can help us visually inspect the sample moments of our series and see whether they vary over time or not.



Conditional Variance Models

- To model the fact that the volatility of certain series varies through time, we turn to the ARCH (autoregressive conditional heteroscedasticity) family of models. Let us start with a general model nesting several ARCH-type specifications. The model is given by the following **two** equations:

a) Conditional Mean Model: $y_t = \mu + x_t{}'\beta + \varepsilon_t, \qquad t = 1, \ldots, T,$

with $\varepsilon_t = \sigma_t z_t$ where $z_t \sim N(0,1)$ and $\varepsilon_t \sim N(0, \underbrace{\boldsymbol{\sigma_t^2}}_{\text{The ‘H’ in (G)ARCH}})$.

b) Conditional Variance Model:

$$\sigma_t^2 = \gamma_0 + \gamma_1 \varepsilon_{t-1}^2 + \gamma_2 \sigma_{t-1}^2 + \gamma_3 \varepsilon_{t-1}^2 d_{t-1}^+$$

$$\mathrm{Var}(\varepsilon_t / I_{t-1}) = \mathrm{Var}(y_t / I_{t-1}) = \sigma_t^2$$

where d_{t-1}^+ is an indicator variable taking the value one when $\varepsilon_{t-1} > 0$, and zero otherwise, I_{t-1} is all the available information at time $t-1$. Notice that the *conditional variance model*, describes the evolution of the **variance of the current error term**. However, in certain cases the conditional variance of the error term is also equal to the conditional variance of the timeseries y_t itself.

Conditional Variance Models

a) Conditional Mean Model:

$$y_t = \mu + x_t{}'\beta + \overset{\text{ARCH in-mean}}{\psi\sigma_t^2} + \varepsilon_t, \text{ with } \varepsilon_t \sim N(0, \sigma_t^2).$$

b) Conditional Variance Model:

$$\sigma_t^2 = \gamma_0 + \overset{\text{ARCH}}{\gamma_1\varepsilon_{t-1}^2} + \overset{\text{GARCH}}{\gamma_2\sigma_{t-1}^2} + \overset{\text{Threshold ARCH}}{\gamma_3\varepsilon_{t-1}^2 d_{t-1}^+}$$

- Four important specifications that can be obtained by restricting the above general model:

(1) ARCH(1) (imposing constraints $\psi = 0$ and $\gamma_2 = 0, \gamma_3 = 0$),
(2) GARCH(1,1) (imposing constraints $\psi = 0$ and $\gamma_3 = 0$),
(3) GJR-GARCH, aka threshold ARCH (imposing constraint $\psi = 0$),
(4) ARCH-in-mean (imposing constraint $\gamma_3 = 0$). In the ARCH-in-mean model, the conditional mean model is also influenced by the conditional variance.

- The ARCH model is due to Engle (1982), the GARCH due to Bollerslev (1986), ARCH-in-mean due to Engle, Lilien, and Robins (1987), and the GJR-ARCH due to Glosten, Jagannathan, and Runkle (1993).

References

- Bollerslev, T. (1986) Generalized autoregressive conditional heteroskedasticity. Journal of Econometrics 31: 307–327.
- Engle, R. F. (1982) Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. Econometrica 50: 987–1007.
- Engle, R. F., D. M. Lilien, and R. P. Robins (1987) Estimating time varying risk premia in the term structure: The ARCH-M model. Econometrica 55: 391–407.
- Glosten, L. R., R. Jagannathan, and D. E. Runkle (1993) On the relation between the expected value and the volatility of the nominal excess return on stocks. Journal of Finance 48: 1779–1801.

Quick recap
Week 8

Today we discussed:

- 3 stylized facts for return timeseries (observed primarily in daily, but also monthly frequency):
  1. Volatility Clustering: large changes (in returns) tend to be followed by large changes, of either sign, and small changes tend to be followed by small changes
  2. Time-varying return volatility
  3. Returns are **uncorrelated**, but **absolute or squared returns** have a positive (slowly decaying) autocorrelation function (-relation to the Momentum investment strategies?)
- Numerous **Conditional Variance** specifications and how to estimate those in Stata:
  - (G)ARCH, GJR-GARCH (aka threshold ARCH), ARCH-in-mean
- Estimation commands for **Panel datasets** in Stata
  - How to `xtset` our data
  - Estimated (`xtreg`) the **determinants of real wages** using the US National Longitudinal Survey (`webuse nlswork`)
  - Fixed Effects = OLS with dummies (for the panel dimension) in order to capture (group-specific) unquantifiable characteristics and avoid potential omitted variable bias (i.e. endogeneity) issues.
- 4 Methods to obtain the OLS estimator in Excel (-see here). Using:
  - Excel’s built-in formula =LINEST()
  - Excel's Solver (Optimization Toolkit) by minimizing the RSS (or maximizing the log likelihood –i.e. the MLE estimator)
  - Linear algebra and Excel's functions for linear algebra operations such as TRANSPOSE(), MMULT(), MINVERSE()