R

R

- Introduction
  - data structures, IO, commands
  - running regressions, output
- Applications
  - Machine learning
  - Text mining
  - GIS

Useful references

- R for Data Science
  - *Chapter 3 (5 in new book): Data transformation with dplyr*
  - *Download at https://r4ds.hadley.nz/*
    Code available at https://github.com/hadley/r4ds
- https://www.rstudio.com/resources/cheatsheets/
- R for Stata users: http://www.matthieugomez.com/statar/
- An Introduction to Statistical Learning with Applications in R
  - Labs in Chapters 2 and 3.
  - http://www-bcf.usc.edu/~gareth/ISL/

Install

- download R from CRAN (*comprehensive R archive network*)
- download Rstudio from *http://www.rstudio.com/download*

Common data types

- int stands for integers.
- dbl stands for doubles, or real numbers.
- chr stands for character vectors, or strings.
- dttm stands for date-times (a date + a time).
- lgl stands for logical, vectors that contain only TRUE or FALSE.
- fctr stands for factors, which R uses to represent categorical variables with fixed possible values.
- date stands for dates.

Logical and comparison operators same as Stata

- Comparison operators

  >, >=, <, <=, !=, ==

- Logical operators

  &, |, !

  also has xor

- Arithmetic operators

  +, -, *, /, ^

Missing values

- NA
  - empty character ”” is not a missing value
  - Use is.na to test for missing values

```
is.na(NA)
#> [1] 1
```

```
is.na("")
#> [1] FALSE
```

R
sum(is.na(mydata) # Number of missing in dataset
rowSums(is.na(data)) # Number of missing per variable
rowMeans(is.na(data))*length(data)# No. of missing per row
complete.cases() # returns a logical vector indicating which cases are complete
na.omit() # returns the object with listwise deletion of missing values.
newdata <- na.omit(mydata) # create new dataset without missing data

Basics

R	Stata
getwd() # Shows the working directory (wd)	pwd
setwd("C:/myfolder/data")	cd c:\myfolder\data
install.packages("ABC") # Install the package on your computer	ssc install abc
library(ABC) # Load the package –-ABC-– to your workspace in R	
rm(list=ls())	clear
c() #concatenate. Any numbers inside () are joined.	
You can use <- or = to generate things.	

```
> x <- c(1,3,2,5)
> x
[1] 1 3 2 5
```

```
> x = c(1,6,2)
> x
[1] 1 6 2
```

Macros

replace	R	Stata
scalars	x <- 10 slice(df, 1:x)	scalar x = 10 keep if _n <= `x'
strings	x <- "MYFILE" read_csv(paste("mydir/", x, ".csv", sep = ""))	local x MYFILE import delimited mydir/`x'.csv
functions	x <- mean df %>% group_by(id) %>% mutate(v1_mean = x(v1))	local x mean egen v1_mean = `x'(v1), by(id)
formulas	formula <- y ~ x lm(formula, df) or formula <- as.formula("y ~ x + id")	

While Stata automatically concatenates strings such as mydir/`x'.csv, in R you have to use the paste function.

Read/write data files (df is a data frame)

Package	R	Stata
rio	df <-import("filepath.csv")	import delimited "filepath.csv"
	read.csv("filepath.csv",header=TRUE)	
	read.table(("filepath.txt", header=TRUE, sep="\t", na.strings = "-9")	import delimited "filepath.txt", delimiter(tab)
rio	import("filepath.xls") # rio package	import excel "filepath.xls"
rio	import("filepath.dta")	use filepath.dta
	load("mydata.rda")	
rio	export(df, "newfilepath.csv")	
	write.table(df, file = "newfilepath.txt", sep = "\t")	
rio	export(df, "newfilepath.dta")	
foreign	write.dta(df, file = "newfilepath.dta")	

Exploring data

R		Stata
str(mydata)	# Provides the structure of the dataset	describe
summary(mydata)	# Provides basic descriptives	summarize
names(mydata)	# Lists variables in the dataset	ds
View(mydata)	# View data in RStudio	browse

Because R loads multiple tables simultaneously, you have to refer tables

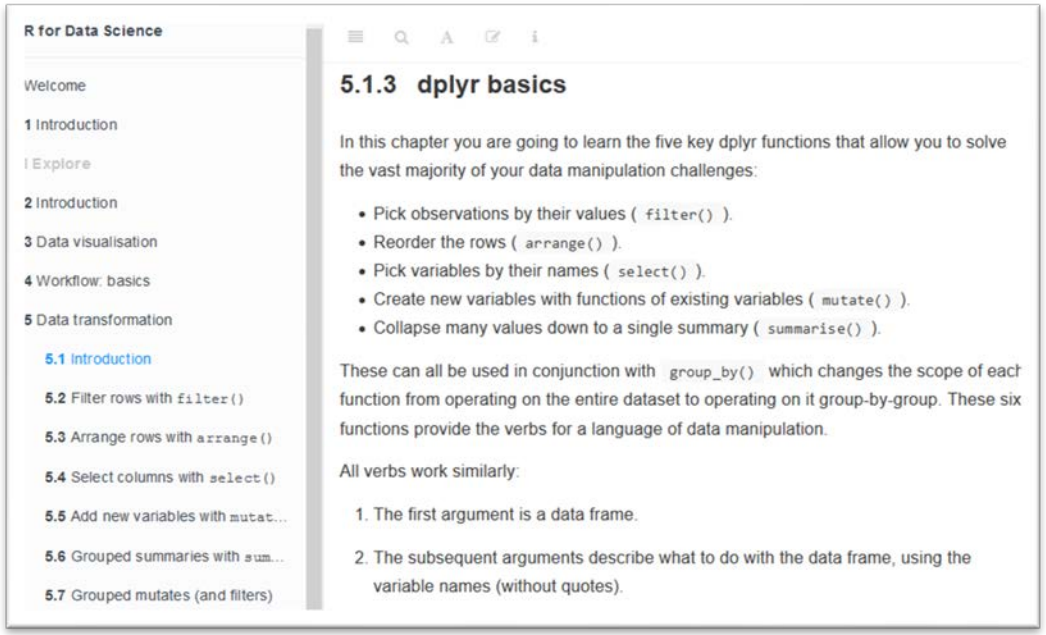
R	Stata
mydata$var1 <- mydata$var2 + mydata$var3	gen var1 = var2 + var3
mydata$total <- dim(mydata)[1]	gen total = _N
mydata$id <- seq(dim(mydata)[1])	gen id = _n

Loops

R	Stata
for (character in c("a", "b", "c")) {	foreach character in a b c {
for (i in 1:100) {	foreach i of numlist 1/100 {
if (2 > 1) {	if 2 > 1 {
} else {	} else {

dplyr

- *R for Data Science*, Chapter 5



- Key functions
  - filter, arrange, select, mutate, summarize
  - used conjunction with group_by
- Syntax
  - The first argument is a data frame.
  - The subsequent arguments describe what to do with the data frame, using the variable names (without quotes).
  - The result is a new data frame.

Select rows

- filter(df, condition) only selects rows where the condition evaluates to TRUE.

R	Stata
df <-slice(df, 1:100)	keep in 1/100
df <-filter(df, v1 >= 2)	keep if v1 >= 2
df <-filter(df, id %in% c("id01", "id02"))	keep if inlist(id, "id01", "id02")
df <-filter(df, between(v2, 3, 5))	keep if inrange(v1, 3, 5)
df <-filter(df, v1 == max(v1))	egen temp = max(v1) keep if v1 == temp
df <-filter(df, !is.na(y))	keep if y!=.
df <-filter(df, complete.cases(df))	select rows without missing observations for any variable

Arrange (sort) rows

- arrange(df, list of variable names).

R	Stata
arrange(df, id, v1)	sort id v1
arrange(df, id, desc(v1))	gsort id -v1

Select columns

- select(df, list of variable names).
  - use dplyr helper functions to select varlist, similar to wildcards.

R	Stata
select(db,id, v1)	keep id v1
select(db, -v1)	drop v1
select(df, id:v2)	keep id-v2
select(df, starts_with(v))	keep v*
select(df, ends_with("v"))	keep *v
select(df, contains("v"))	keep *v*
select(df, matches("^v.$"))	select all variables that matches a regular expression
select(df, v1, everything())	order(v1)

Generate new variables (mutate)

R	Stata
mutate(db, new = 1)	gen new=1
mutate(db, x = …, y=…, z= x / y)	you can refer to columns that you've just created
transmutate(db, new = 1)	If you only want to keep the new variables, use transmute()

Replace

R	Stata
mutate(db, v1 = ifelse(id == "id01", 0, v1))	replace v1 = 0 if id =="id01" or replace v1 = cond(id =="id01" ,0,v1)
mutate(db, v1_share =v1/sum(v1))	gen v1_sum=sum(v1) gen v1_share=v1/v1_sum[_N] drop v1_sum
mutate_at(db, vars(v1, v2), funs(as.character)) #multiple variables	tostring v1 v2, replace force
mutate_at(vars(v1, v2), funs(as.character, mean))	multiple variables & functions (alternative to looping over variables and functions)

Other useful commands

R	Stata
rename(db, id1 = id)	rename id id1
lead(x)	F1.x
lag(x)	L1.x
cusum(x)	sum(x)
x %/% y (example: 517 %/% 100 = 5)	int(x/y)
x %% y (example: 517 %% 100 = 17)	mod(x,y)

Grouped summaries (Stata: egen, collapse)

	year	month	day	dep_time	sched_dep_time	dep_delay	a
1	2013	1	1	517	515	2	
2	2013	1	1	533	529	4	
3	2013	1	1	542	540	2	
4	2013	1	1	544	545	-1	
5	2013	1	1	554	600	-6	
6	2013	1	1	554	558	-4	
7	2013	1	1	555	600	-5	

```
> summarize(flights, delay = mean(dep_delay, na.rm = TRUE))
# A tibble: 1 x 1
  delay
  <dbl>
1  12.6
```

- group_by()
  - changes the unit of analysis from complete dataset to individual groups
- ungroup()
  - changes the unit of analysis to complete dataset

```
> by_day <- group_by(flights, year, month, day)
> summarize(by_day, delay = mean(dep_delay, na.rm = TRUE))
# A tibble: 365 x 4
# Groups:   year, month [12]
    year month   day delay
   <int> <int> <int> <dbl>
 1  2013     1     1 11.5
 2  2013     1     2 13.9
 3  2013     1     3 11.0
 4  2013     1     4  8.95
 5  2013     1     5  5.73
```

```
by_day <-ungroup(by_day)
summarize(by_day, delay = mean(dep_delay, na.rm = TRUE))
A tibble: 1 x 1
delay
<dbl>
 12.6
```

Grouped summaries

R	Stata

by_day <- group_by(df, year, month, day) #creates grouped df
summarize(by_day, delay = mean(dep_delay, na.rm = **TRUE**))
	collapse delay, by(year, month, day)

by_day <- group_by(df, year, month, day) df_by_day<-summarize(by_day, count = sum(!is.na(delay)), dist = mean(distance, na.rm = **TRUE**), delay = mean(dep_delay, na.rm = **TRUE**)))	collapse (count) count=delay (mean) distance delay, by(year, month,day)
ungroup()	remove grouping, and return to operations on ungrouped data

The Pipe "%>%"

- Output df from previous command is implied input in next.
  - x %>% f(y) turns into f(x, y), and
  - x %>% f(y)%>% g(z) turns into g(f(x, y), z), etc.

```r
by_dest <- group_by(flights, dest)
delay <- summarize(by_dest,
  count = n(),
  dist = mean(distance, na.rm = TRUE),
  delay = mean(arr_delay, na.rm = TRUE)
)
delay <- filter(delay, count > 20, dest != "HNL")
```

```r
delays <- flights %>%
  group_by(dest) %>%
  summarize(
    count = n(),
    dist = mean(distance, na.rm = TRUE),
    delay = mean(arr_delay, na.rm = TRUE)
  ) %>%
  filter(count > 20, dest != "HNL")
```

Stata

```
collapse (count) count=delay (mean)
distance delay if count>20 & dest !="HNL",
by(year, month,day)
```

Useful summary functions

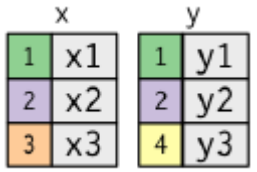
R	
mean(x), median(x),	
spread: sd(x), IQR(x), mad(x)	
rank: min(x), quantile(x, 0.25), max(x)	
position: first(x), nth(x, 2), last(x) (or x[1], x[2], and x[length(x)])	x[1], x[2],x[_N]
Counts and proportions of logical values sum(x > 10), mean(y == 0)	
n(), sum(!is.na(x))	_N, egen (count) x
n_distinct(x)	Number of distinct values
count(df,x)	collapse (count) x, by(x)

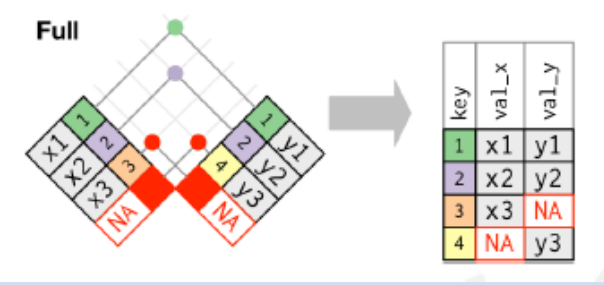
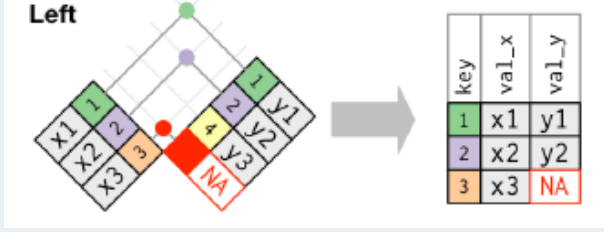
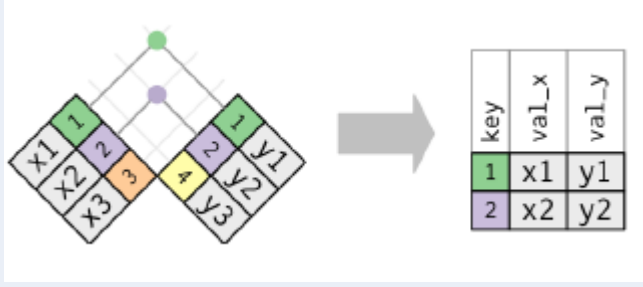
Apply functions within groups (egen)

- The difference to ”collapse” is that you don’t use summarize()

R	Stata
df %>% group_by(id) %>% mutate(v1mean = mean(v1))	egen v1mean = mean(v1), by(id)
df %>% group_by(id) %>% mutate(v1 = v1[1])	by id : replace v1 = v1[1]
Compare:	
df<- group_by(id) %>% summarize(v1mean = mean(v1)))	collapse v1mean=v1, by(id)

Merge



Base R	R dplyr	Stata	
merge(df1, df2, by = "v1", all.x = TRUE, all.y = TRUE)	full_join(df1, df2, by = "v1")	merge v1 using df2, keep(master matched using)	
merge(df1, df2, by = "v1", all.x = TRUE, all.y = FALSE)	left_join(df1, df2, by = "v1")	merge v1, keep(master matched)	
merge(df1, df2, by = "v1", all.x = FALSE, all.y = TRUE)	right_join(df1, df2, by = "v1")	merge v1, keep(matched using)	
merge(df1, df2, by = "v1", all.x = FALSE, all.y = FALSE)	inner_join(df1, df2, by = "v1")	merge v1, keep(matched)	

Merge

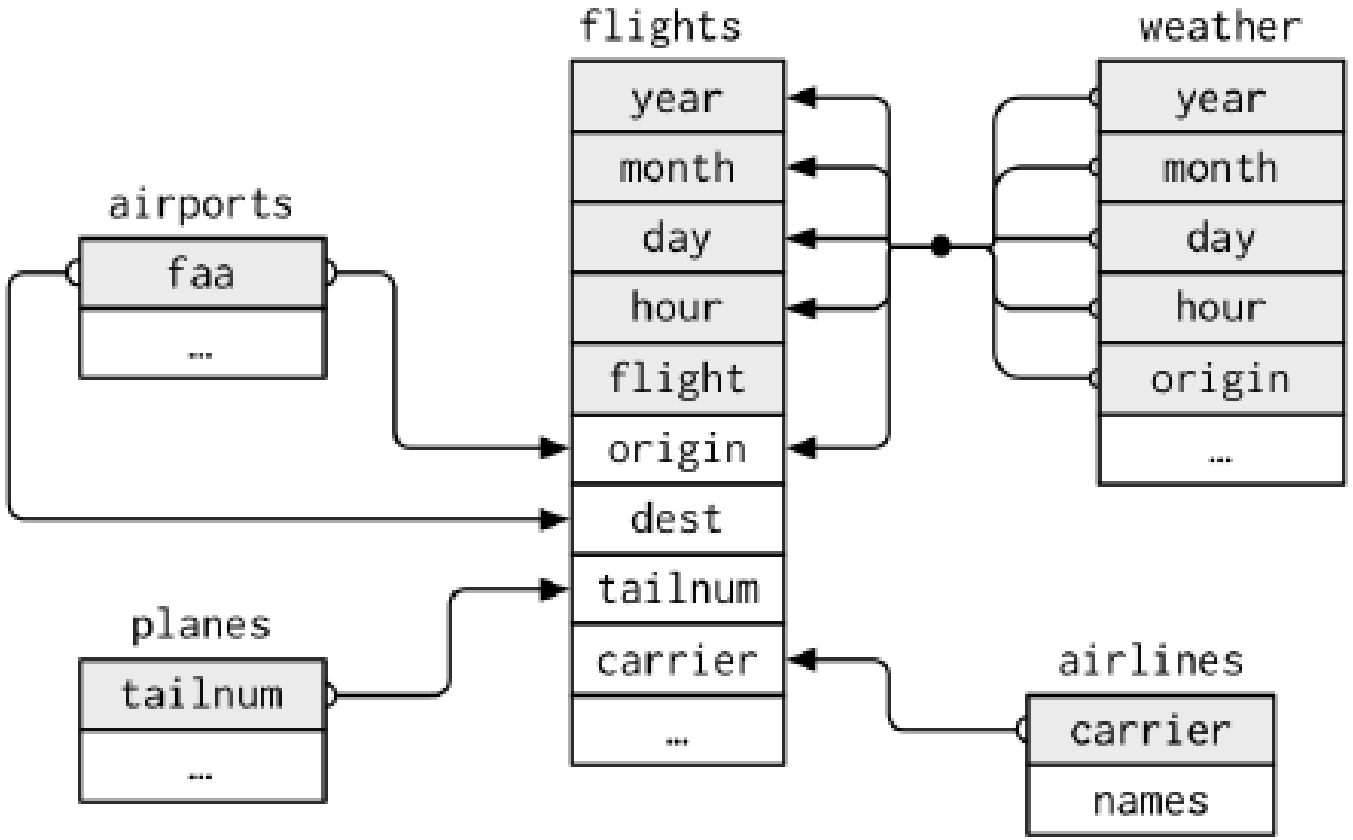
- By
  - When the option by is not specified, merges are based on variables with common names.
  - Different variable names datasets: by.x = id_var1, by.y = id_var2
- Duplicate keys
  - When there are multiple matches both in the master and using datasets, R functions create all combinations of rows, similarly to joinby (and contrary to merge m:m).
  - R does not have 1:1, 1:m, m:m notation.
  - You have to check yourself whether key is unique
    - df %>%
      - count(id_var) %>%
      - filter(n > 1)

A relational database

- Stored in normal form
  - Unique keys

```
planes %>%
  count(tailnum) %>%
  filter(n > 1)
A tibble: 0 x 2
... with 2 variables: tailnum <chr>, n <int>
```

  - Foreign keys



	carrier	name
1	9E	Endeavor Air Inc.
2	AA	American Airlines Inc.
3	AS	Alaska Airlines Inc.
4	B6	JetBlue Airways
5	DL	Delta Air Lines Inc.
6	EV	ExpressJet Airlines Inc.
7	F9	Frontier Airlines Inc.
8	FL	AirTran Airways Corporation
9	HA	Hawaiian Airlines Inc.
10	MQ	Envoy Air
11	OO	SkyWest Airlines Inc.

	year	month	day	hour	origin	dest	tailnum	carrier
1	2013	1	1	5	EWR	IAH	N14228	UA
2	2013	1	1	5	LGA	IAH	N24211	UA
3	2013	1	1	5	JFK	MIA	N619AA	AA
4	2013	1	1	5	JFK	BQN	N804JB	B6
5	2013	1	1	6	LGA	ATL	N668DN	DL
6	2013	1	1	5	EWR	ORD	N39463	UA
7	2013	1	1	6	EWR	FLL	N516JB	B6
8	2013	1	1	6	LGA	IAD	N829AS	EV
9	2013	1	1	6	JFK	MCO	N593JB	B6
10	2013	1	1	6	LGA	ORD	N3ALAA	AA
11	2013	1	1	6	JFK	PBI	N793JB	B6

```
flights2 %>%
  select(-origin, -dest) %>%
  left_join(airlines, by = "carrier")
A tibble: 336,776 x 7
   year month   day  hour tailnum carrier name
  <int> <int> <int> <dbl> <chr>   <chr>   <chr>
1  2013     1     1     5 N14228  UA      United Air Lines Inc.
2  2013     1     1     5 N24211  UA      United Air Lines Inc.
3  2013     1     1     5 N619AA  AA      American Airlines Inc.
4  2013     1     1     5 N804JB  B6      JetBlue Airways
5  2013     1     1     6 N668DN  DL      Delta Air Lines Inc.
6  2013     1     1     5 N39463  UA      United Air Lines Inc.
7  2013     1     1     6 N516JB  B6      JetBlue Airways
8  2013     1     1     6 N829AS  EV      ExpressJet Airlines Inc.
9  2013     1     1     6 N593JB  B6      JetBlue Airways
0  2013     1     1     6 N3ALAA  AA      American Airlines Inc.
... with 336,766 more rows
```

Append and reshape

R dplyr	Stata
row_binds(df1, df2) #tidyr	append using "using.dta"
gather(dfwide, variable, value, starts_with("stub"))	reshape long stub, i(i) j(variable) string
gather(dfwide, variable, value, starts_with("stub"))	reshape long stub, i(id) j(variable) string rename stub value
spread(dflong, variable, value)	reshape wide value, i(i) j(variable) string

- When the option by is not specified,
  merges are based on variables with common names
- When there are multiple matches both in the master and using datasets,
  R functions create all combinations of rows,
  similarly to joinby (and contrary to merge m:m).

Panel data (package statar)

R	Stata
df %>% group_by(id) %>% mutate(value_l = lag(value, n = 1, order_by = date))	by id : gen value_l = value[_n-1]
df %>% group_by(id) %>% mutate(value_l = tlag(value, n = 1, date))	tsset id date value_l = L.value
lag and tlag differ when the previous date is missing. In this case, the function lag returns the value in the most recent date while the function tlag returns a missing value.	

R script editor

R script editor	
Cmd/Ctrl-Enter	executes the current R expression in the console
Cmd/Ctrl-Shift-S	execute the complete script

Regression commands

- See Witten, Hastie and Tibshirani(2015), 3.6 Lab: Linear Regression.
- Ch3.R

R	Stata
lm(y ~ x ,data=df) or formula <- y ~ x or formula <- as.formula("y ~ x") + lm(formula,data=df)	reg y x
ivreg(y~x+w|w+z, data) # AER package	ivregress 2sls y=x w (x=z)

Regression formulas

R	Stata
y ~ x1 x2	y x1 x2
y ~ 0 + x1	y x1, nocons
log(y) ~ log(x)	gen ylog = log(y) ; gen x2log = log(x2) ; ylog x2log
y ~ I(x1 + x3)	gen x3 = x1 + x2 ; y x3

Factor variables and interactions

R	Stata
y ~ as.factor(x1)	y i.x1
y ~ x1*x2 /*includes main effects*/	y c.x1#c.x2
y ~ x1:x2	y c.x1##c.x2
y ~ x1*as.factor(x2)	y c.x1##i.x2

Factor variables in large data sets

R: package lfe	Stata: reghdfe
felm(y ~ x1 | id1 | 0 | id1, df, weight = x3)) felm(formula|partial out vars|IV|cluster|,data,options)	areg y x1 [w=x3], a(id1) cl(id1)
felm(y ~ x2 | x3:id1 + id1|0|id1+id2, df)	reghdfe y x2, a(c.x3#i.id1 id1) cl(id1 id2)
felm(y ~ x3 | id1 | (x2 ~ x1) | id1 + id2, df)	reghdfe y x3 (x2 = x1), a(id1) cl(id1 id2)
felm(y ~ x2 | x3:id1 + id1, df)	reghdfe y x2, a(c.x3#i.id1 id1) cl(id1 id2)

reghdfe is a generalization of areg (and xtreg,fe, xtivreg,fe) for multiple levels of fixed effects .

areg and felm do not do a degrees of freedom correction to the standard errors (in contrast to xtreg).
This can be done manually.
Check also feols in r package fixest.

Object orientation

- **In most other econometrics packages:**
  - An analysis leads to a large amount of output containing information on estimation, model diagnostics, specification tests, etc.
- **In R:**
  - Analysis is broken down into a series of steps.
  - Intermediate results are stored in objects.
  - Minimal output at each step (often none).
  - Objects can be manipulated and interrogated to obtain the information required (e.g., print(), summary(), plot()).
- **Fundamental design principle:**
  - “Everything is an object.”
  - **Examples:** Vectors and matrices are objects, but also fitted model objects, functions, and even function calls) facilitates programming tasks.

Object orientation

`print()`	simple printed display with coefficients
`summary()`	standard regression summary
`plot()`	diagnostic plots
`coef()`	extract coefficients
`vcov()`	associated covariance matrix
`predict()`	(different types of) predictions for new data
`fitted()`	fitted values for observed data
`residuals()`	extract (different types of) residuals
`terms()`	extract terms
`model.matrix()`	extract model matrix (or matrices)
`nobs()`	extract number of observations
`df.residual()`	extract residual degrees of freedom
`logLik()`	extract fitted log-likelihood

Post-estimation commands in R

- An estimation function returns a list that contains the
  - estimates, covariance matrix and often the
  - residuals, the predicted values, or the original variables used in the estimation.
  - Apply the names function to examine the result:

```
result <- felm(y ~ x2, df)
names(result)
#>  [1] "coefficients"  "badconv"       "Pp"            "N"             "p"
#>  [6] "inv"           "beta"          "response"      "fitted.values" "residuals"
#> [11] "r.residuals"   "terms"         "cfactor"       "numrefs"       "df"
#> [16] "df.residual"   "rank"          "exactDOF"      "vcv"           "robustvcv"
#> [21] "clustervcv"    "cse"           "ctval"         "cpval"         "clustervar
"
#> [26] "se"            "tval"          "pval"          "rse"           "rtval"
#> [31] "rpval"         "xp"            "call"
pryr::object_size(result)
#> [1] 88 MB
```

Post-estimation commands in R

- Applying summary prints a table similar to Stata output

```
summary(result)
#> Call:
#>    felm(formula = y ~ x2, data = df)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -48.834 -23.175  -5.028  25.222  50.939
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 48.746112   0.064228 758.949   <2e-16 ***
#> x2           0.001997   0.001059   1.886   0.0593 .
#> ---
#> Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
#>
#> Residual standard error: 29.91 on 999998 degrees of freedom
#> Multiple R-squared: 3.556e-06   Adjusted R-squared: 1.556e-06
#> F-statistic:3.556 on 1 and 999998 DF, p-value: 0.05934
```

Post-estimation commands in R

- The package stargazer combines several regression results in a table:

```
stargazer(result, type = "text")
#> ================================================
#>                          Dependent variable:
#>                      ---------------------------
#>                                    y
#> ------------------------------------------------
#> x2                                -0.0004
#>                                   (0.001)
#>
#> Constant                         50.315***
#>                                   (0.064)
#>
#> ------------------------------------------------
#> Observations                     1,000,000
#> R2                                0.00000
#> Adjusted R2                      -0.00000
#> Residual Std. Error    29.707 (df = 999998)
#> ================================================
#> Note:                *p<0.1; **p<0.05; ***p<0.01
```

Saving output

- Logfile
  - sink(”outfile.txt", append = FALSE)
  - regfit.fwd$xnames[regfit.fwd$vorder]
  - sink()
- Print a plot to a pdf file:

pdf (" Figure .pdf ")

plot(x,y,col =" green ")

dev.off ()

Graphics in R

Basically, a graph is composed of three distinct parts:

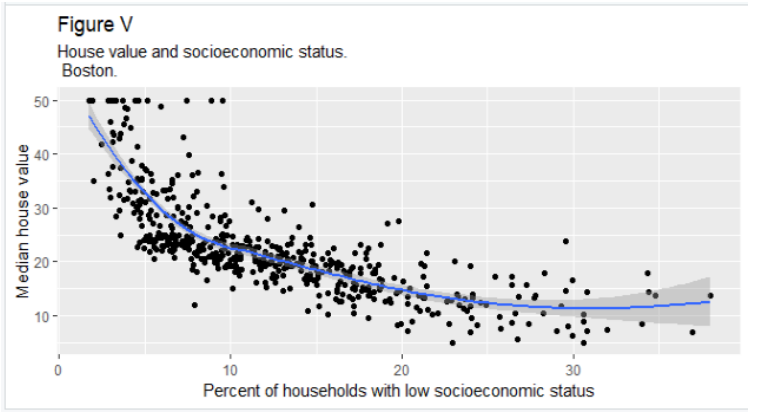
- Aesthetics that maps variables (columns) to axis or colors
- stats that transform the data
- geoms that plot an aesthetic

- R for Data Science Ch 3

Graphics in R using ggplot

R	Stata
ggplot(data=Boston) + geom_point(mapping=aes(x=lstat, y=medv))	twoway scatter x y
ggplot(data=Boston, mapping=aes(x=lstat, y=medv)) + geom_point() + geom_smooth()	twoway (scatter x y) (lpoly x y)
geom_point, geom_line, geom_text, etc.	scatter, line, etc
geom_histogram, geom_density(kernel = "gaussian")	Histogram, density,

```
ggplot(data=Boston, mapping=aes(x=lstat, y=medv) ) +
  geom_point() +
  geom_smooth() +
  labs(
    title="Figure V",
    subtitle="House value and socioeconomic status. \n Boston.",
    x="Percent of households with low socioeconomic status",
    y="Median house value"
  )
```



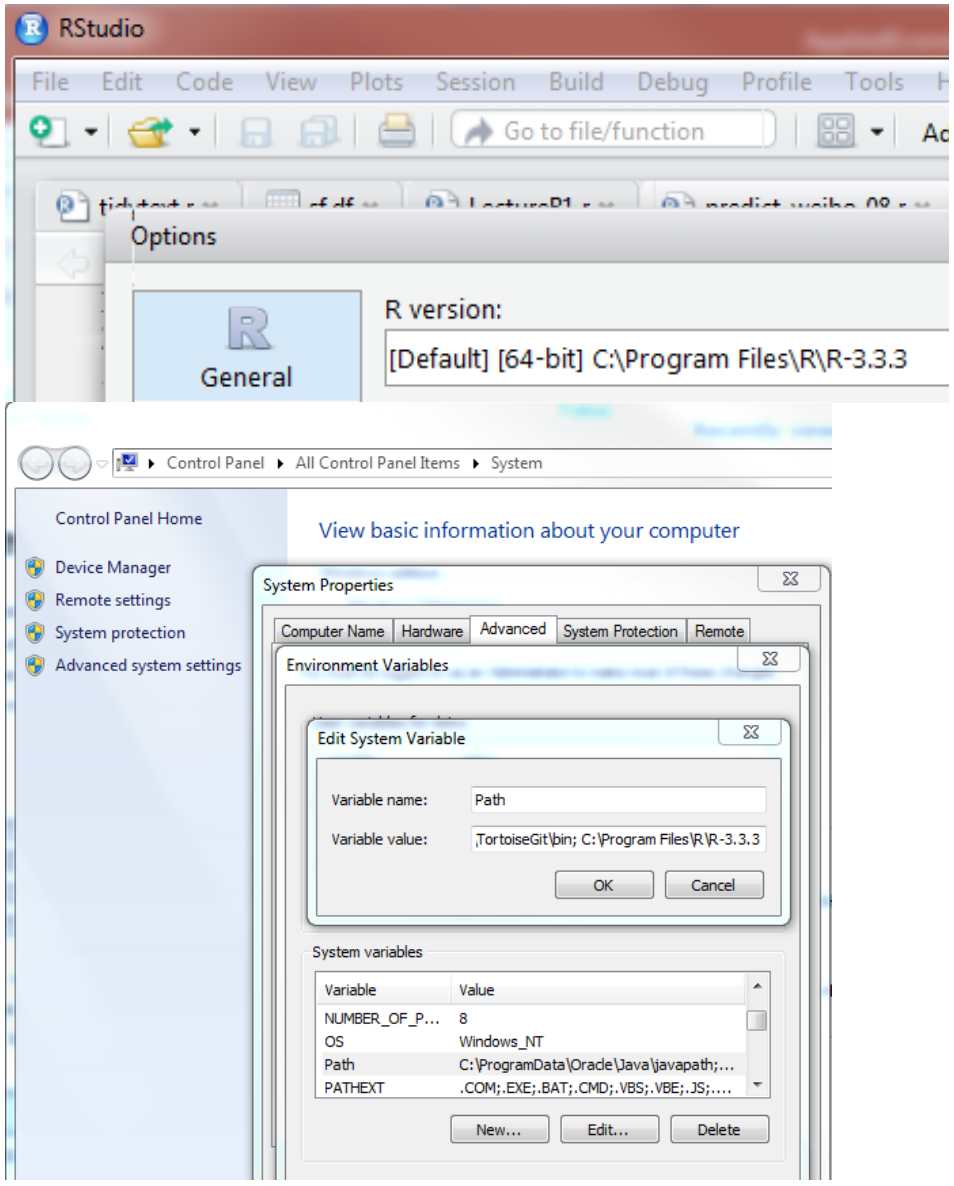
Examples

- ch3.R (i R\RexamplesHastie)
- TableIV_data.r (i Examples\AK91\Build\Code)
- TableIV.r
- FigureV.r
- ch6.r

Executing R scripts from the command promts (or Stata)

- Add the path to R to your Windows path (where Windows looks for executables).

1. Identify your R-exe path eg. in Rstudio >Tools> Global Options
2. Right-click on Computer > Properties >Advanced system settings > Environment variables > choose Path, click edit
3. At the end of Variable value add the path of your R program,
   in my case ” ; C:\Program Files\R\R-3.3.3”.



Executing R scripts from your master file

- In Stata master file: call Rscript with project folder path as argument.

```
* Set global macro
global rootdir "E:/c_old/DavidD/Courses/AppliedEmpirical/Examples/AK91"

* Do analysis in R
! Rscript $rootdir/Build/Code/TableIV_data.r $rootdir
! Rscript $rootdir/Build/Code/TableIV.r      $rootdir
! Rscript $rootdir/Build/Code/FigureV.r      $rootdir
```

- In R: read project folder path. first line same as "args rootdir" in Stata

```
rootdir <- commandArgs(trailingOnly = TRUE)
setwd(rootdir)
```

Task 2d: Refresh R-skills

1. AK91 in R
    - Install R and Rstudio.
    - Replicate your Angrist and Krueger (1991) analysis in R.
    - Replicate Figure V in R.