Intro to Cross-Sectional Time-Series Data in R for Students of IR & Comparative Politics

Justin Murphy

November 21, 2014

This document demonstrates how to perform several routine tasks in the analysis of state-level economic and political data for multiple countries over time, using the programming language R. It is aimed toward early-career graduate students with little or no experience with R. The .Rmd file which generated this document may prove especially useful as a template from which students may wish to get started. All one needs to do is download and install R, RStudio, and LaTex, open this .Rmd file in Rstudio, and modify/extend the examples already coded here.

This document downloads from the web two widely used data sources (the World Bank's World Development Indicators and the Polity IV measure of democracy), merges them, cleans and transforms some variables, visualizes several variables in commonly recurring types of graphs, runs simple models, and generates automatically formatted tables.

Download packages (unless you already have them!)

```
install.packages("WDI")
install.packages("countrycode")
install.packages("plm")
install.packages("ggplot2")
install.packages("reporttools")
install.packages("stargazer")
install.packages("psData")
```

Introducing the World Development Indicators API in R

1. Data collected by the World Bank.
2. Hundreds of state-level variables.
3. From 1960 to 2012.
4. Import to R directly.

```
require("WDI")  # load WDI package
```

- To find the variable codes to call, first search with the `WDIsearch()` function.
- Then call data with the `WDI()` function.
- Super easy!

Simple demonstration of WDI package in R

- Let's say we're interested in GDP growth across countries. First we'll search for "gdp growth" to see if the World Bank has a variable matching this search term.

```r
### Use this function to find indicator codes for things that interest you
WDIsearch("gdp growth")
```

```
##      indicator           name
## [1,] "NV.AGR.TOTL.ZG"    "Real agricultural GDP growth rates (%)"
## [2,] "NY.GDP.MKTP.KD.ZG" "GDP growth (annual %)"
```

- To download the data straight from the web, we use the `WDI()` function, specifying which countries, variables, and years we're interested in.

```r
### Insert as many countries and variables as you want, within this range.
### Let's try getting GDP growth for US and France from 1960-2012
wb <- WDI(country=c("US", "FR"), indicator=c("NY.GDP.MKTP.KD.ZG"), start=1960, end=2012)
```

Make a real research dataset

- Download a set of commonly used variables for all countries between 1960 and the most recent update of the WDI. Setting `extra=TRUE` retrieves some extra variables such as region, geocodes, and additional country codes. The additional country code turns out to merge better with other datasets (see below).

```
wb <- WDI(country="all", indicator=c("SE.XPD.TOTL.GB.ZS", "GC.DOD.TOTL.GD.ZS",
                                     "BN.CAB.XOKA.GD.ZS", "SP.POP.DPND", "AG.LND.TOTL.K2",
                                     "NY.GDP.MKTP.CD", "NY.GDP.MKTP.PP.CD",
                                     "NY.GDP.MKTP.KD", "NY.GDP.PCAP.CD", "SP.POP.TOTL",
                                     "IT.PRT.NEWS.P3", "IT.RAD.SETS", "NE.IMP.GNFS.ZS",
                                     "NE.EXP.GNFS.ZS", "NE.TRD.GNFS.ZS",
                                     "BX.KLT.DINV.WD.GD.ZS", "BN.KLT.PRVT.GD.ZS",
                                     "NE.CON.GOVT.ZS", "SL.IND.EMPL.ZS", "NV.IND.TOTL.ZS",
                                     "NY.GDP.MKTP.KD.ZG", "NY.GDP.PCAP.KD"),
          start=1960,
          end=2012,
          extra=TRUE)
```

Clean & transform the data

```
# Assignment operator "<-" will add a new variable to the dataframe "wb"
wb$debt<-wb$GC.DOD.TOTL.GD.ZS      # central government debt as share of gdp
wb$curracct<-wb$BN.CAB.XOKA.GD.ZS   # current account balance as share of gdp
wb$dependency<-wb$SP.POP.DPND  # dependency-age population as share of total population
wb$land<-wb$AG.LND.TOTL.K2    # total land in square kilometers
wb$pop<-wb$SP.POP.TOTL        # total population
wb$radioscap<-(wb$IT.RAD.SETS/wb$pop)*1000  # radios per 1000 people
wb$logradioscap<-log(wb$radioscap + 1)  # logarithm of radios per capita, because it's skewed
wb$newspaperscap<-wb$IT.PRT.NEWS.P3  # newspapers in circulation per 1000 people
wb$gdp<-wb$NY.GDP.MKTP.KD     # gross domestic product, constant 2000 USD
wb$gdp2<-wb$NY.GDP.MKTP.CD  # gross domestic product, current USD, millions
wb$gdp3<-wb$NY.GDP.MKTP.PP.CD # gross domestic product, purchasing-power parity, current
wb$gdpcap<-wb$NY.GDP.PCAP.KD      # gdp per capita constant 2000 USD
wb$gdpgrowth<-wb$NY.GDP.MKTP.KD.ZG   # change in gross domestic product
wb$imports<-wb$NE.IMP.GNFS.ZS # imports of goods and services as share of gdp
wb$exports<-wb$NE.EXP.GNFS.ZS # exports of goods and services as share of gdp
wb$trade<-wb$NE.TRD.GNFS.ZS   # total trade as share of gdp
wb$fdi<-wb$BX.KLT.DINV.WD.GD.ZS  # foreign-direct investment as share of gdp
wb$privatecapital<-wb$BN.KLT.PRVT.GD.ZS   # private capital flows as share of gdp
wb$spending<-wb$NE.CON.GOVT.ZS     # government consumption expenditure as share of gdp
wb$industry<-wb$SL.IND.EMPL.ZS     # employment in industry as share of total employment
wb$industry2<-wb$NV.IND.TOTL.ZS    # value added in industry as share of gdp
```

Subset the data

Take a peak at the "head" (the first six rows) using the `head()` function. This is "panel" or "pooled" cross-sectional, time-series data. This is also called "country-year" format.

```r
head(wb[,1:7])   # limit to columns 1 through 7
```

```
##   iso2c country year SE.XPD.TOTL.GB.ZS GC.DOD.TOTL.GD.ZS BN.CAB.XOKA.GD.ZS
## 1        Africa 1960                NA                NA                NA
## 2        Africa 1961                NA                NA                NA
## 3        Africa 1962                NA                NA                NA
## 4        Africa 1963                NA                NA                NA
## 5        Africa 1964                NA                NA                NA
## 6        Africa 1965                NA                NA                NA
##   SP.POP.DPND
## 1          NA
## 2          NA
## 3          NA
## 4          NA
## 5          NA
## 6          NA
```

The World Bank keeps data not only on countries, but on certain aggregates of countries. Aggregate units are distinguished by the category "Aggregates" in the factor variable "region" in the `wb` object (a dataframe). Then, each of the different aggregates are specified by the "country" variable. To keep things tidy, let's break the `wb` dataframe into a few separate dataframes. We'll make one for all the aggregates, one for regions in particular (or any other set of aggregates you're interested in; try replacing the region categories below with other categories in the "country" variable of the `aggregate` subset we're about to make…), and one for the good-old fashioned states.

```r
# make a subset of only the region/world aggregate measures
aggregates<-subset(wb, region=="Aggregates")

# make a subset of only global-level aggregate measures
world<-subset(aggregates, country=="World")

# Make a subset of only region aggregates
regions<-subset(aggregates, country=="East Asia & Pacific (all income levels)" |
                            country=="Middle East & North Africa (all income levels)" |
                            country=="Sub-Saharan Africa (all income levels)" |
                            country=="North America" |
                            country=="Latin America & Caribbean (all income levels)" |
                            country=="Europe & Central Asia (all income levels)" |
                            country=="South Asia")

wb<-subset(wb, region!="Aggregates")
```

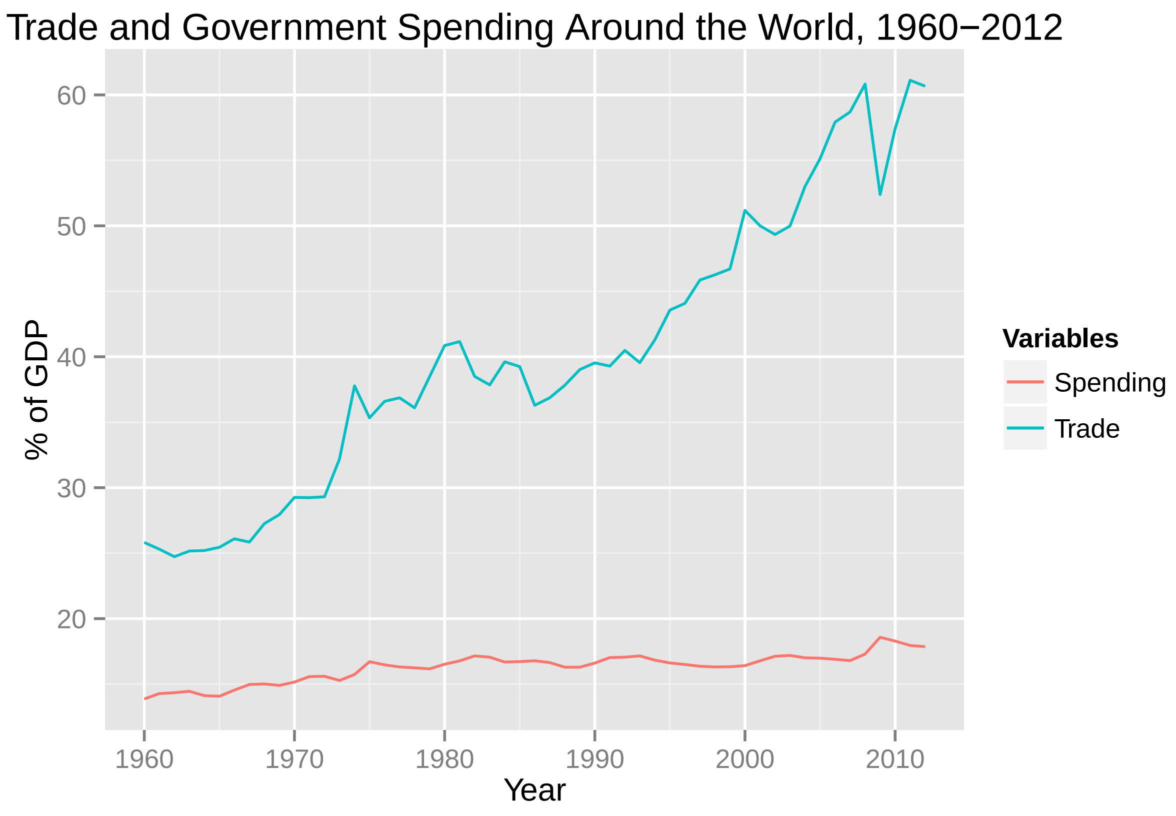
Visualizing panel data

Below we investigate our data using one of the most powerful graphing packages in R: ggplot2. The formulas below are a quick tour of plots suited especially for the demands of panel data.

Multiple variables (on a single scale) for one unit over time

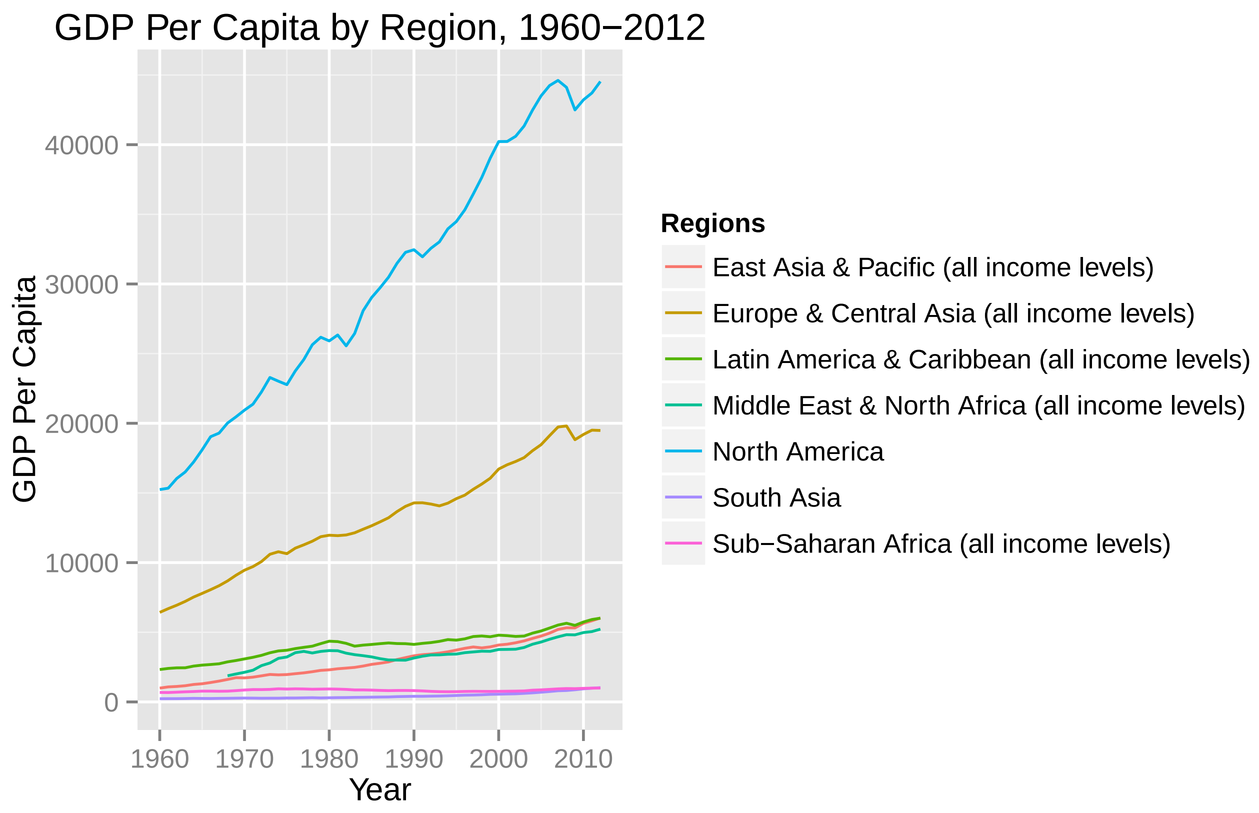
```r
# load ggplot2 package, must install.packages("ggplot2") if you haven't previously.
require(ggplot2)

ggplot(world, aes(x = year)) +
  geom_line(aes(y = trade, colour = "Trade")) +
  geom_line(aes(y = spending, colour = "Spending")) +
  scale_colour_discrete(name = "Variables") +
  xlab("Year") +
  ylab("% of GDP") +
  ggtitle("Trade and Government Spending Around the World, 1960-2012")
```



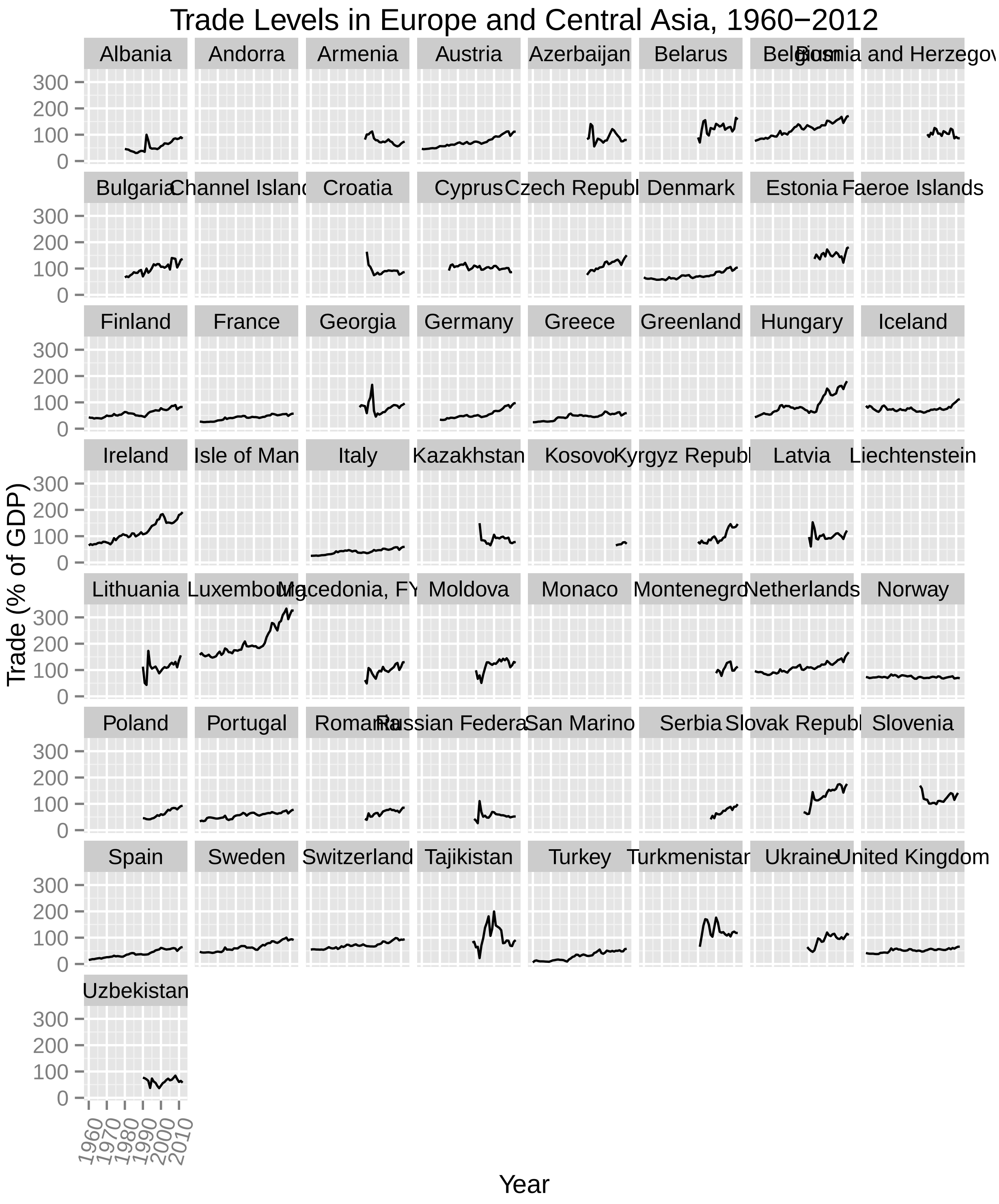
One time-series plot for one variable and one line per unit

```r
ggplot(regions, aes(x = year, y = gdpcap, colour=country)) +
  geom_line() +
  scale_colour_discrete(name = "Regions") +
  xlab("Year") +
  ylab("GDP Per Capita") +
  ggtitle("GDP Per Capita by Region, 1960-2012")
```



Panel of time-series plots for one variable and one plot per unit

```r
ggplot(wb[wb$region=="Europe & Central Asia (all income levels)",], aes(x=year, y=trade)) +
  geom_line(aes(group=country)) +
  facet_wrap(~ country) +
  xlab("Year") +
  ylab("Trade (% of GDP)") +
  ggtitle("Trade Levels in Europe and Central Asia, 1960-2012") +
  theme(axis.text.x = element_text(angle = 75, hjust = 1))
```


Trade Levels in Europe and Central Asia, 1960−2012
Albania
Andorra
Armenia
Austria
Azerbaijan
Belarus
Belgium
Bosnia and Herzegov
Bulgaria
Channel Islanc
Croatia
Cyprus
Czech Republ
Denmark
Estonia
Faeroe Islands
Finland
France
Georgia
Germany
Greece
Greenland
Hungary
Iceland
Ireland
Isle of Man
Italy
Kazakhstan
Kosovo
Kyrgyz Republ
Latvia
Liechtenstein
Lithuania
Luxembourg
Macedonia, FY
Moldova
Monaco
Montenegro
Netherlands
Norway
Poland
Portugal
Romania
Russian Federa
San Marino
Serbia
Slovak Republ
Slovenia
Spain
Sweden
Switzerland
Tajikistan
Turkey
Turkmenistan
Ukraine
United Kingdom
Uzbekistan
Trade (% of GDP)
0
100
200
300
1960
1970
1980
1990
2000
2010
Year

Import Polity IV

```
require(psData)
polity <- PolityGet(vars = "polity2")
polity<-subset(polity, year>=1960 & year<=2012)
head(polity)                # peek at the dataframe
```

```
##     iso2c     country year polity2
## 161    AF Afghanistan 1960     -10
## 162    AF Afghanistan 1961     -10
## 163    AF Afghanistan 1962     -10
## 164    AF Afghanistan 1963     -10
## 165    AF Afghanistan 1964      -7
## 166    AF Afghanistan 1965      -7
```

Easily deal with country codes using the `countrycode` package

```
# load the "countrycode" package for converting codes
require(countrycode)

# convert COW code to ISO 3-character code as in WDI
polity$cowcode<-countrycode(polity$iso2c, "iso2c", "cown")

# peek at the dataframe again
head(polity)
```

```
##     iso2c     country year polity2 cowcode
## 161    AF Afghanistan 1960     -10     700
## 162    AF Afghanistan 1961     -10     700
## 163    AF Afghanistan 1962     -10     700
## 164    AF Afghanistan 1963     -10     700
## 165    AF Afghanistan 1964      -7     700
## 166    AF Afghanistan 1965      -7     700
```

Subset variables of interest and merge with WDI

```r
# make subset of key variables
dem<-subset(polity, select=c("iso2c", "year", "polity2"))

df<-merge(wb, dem, by=c("iso2c", "year"))      # 7,634 country-years matched, yay
```

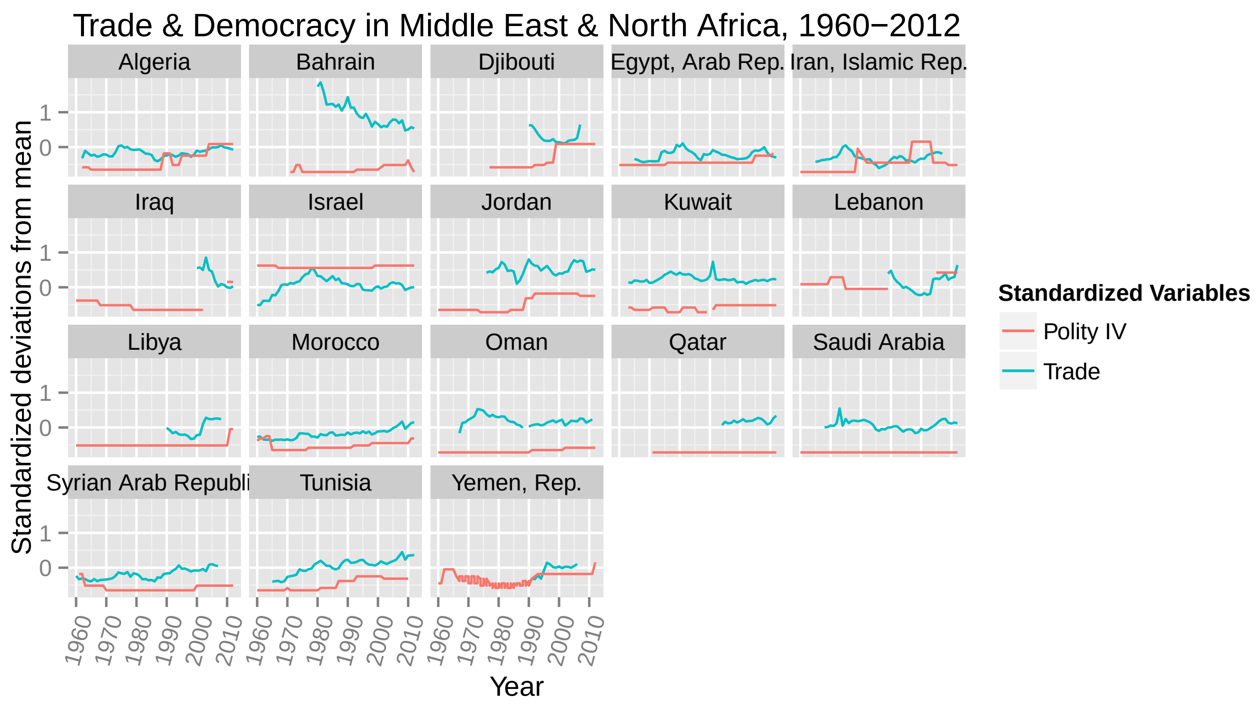
Standardize time-series on different scales

```r
require(arm)

# subtract out the mean and divide by two standard deviations
df$dem.z<-rescale(df$polity2)
df$trade.z<-rescale(df$trade)
```

Visualize standardized time-series by country

```r
ggplot(df[df$region=="Middle East & North Africa (all income levels)",], aes(x=year)) +
  geom_line(aes(y=trade.z, group=country, colour="Trade")) +
  geom_line(aes(y=dem.z, group=country, colour="Polity IV")) +
  facet_wrap(~ country) +
  scale_colour_discrete(name = "Standardized Variables") +
  xlab("Year") +
  ylab("Standardized deviations from mean") +
  ggtitle("Trade & Democracy in Middle East & North Africa, 1960-2012") +
  theme(axis.text.x = element_text(angle = 75, hjust = 1))
```



Modeling pooled data

```r
require(plm)
model<-plm(spending ~ trade + polity2 + dependency + gdpgrowth + lag(spending),
           index = c("iso3c","year"),
           model="within",
           effect="twoways",
           data=df)
ls(model)    # see the contents of the dataframe
```

```
## [1] "args"         "call"         "coefficients" "df.residual"
## [5] "formula"      "model"        "residuals"    "vcov"
```

```r
summary(model)  # see main model output
```

```
## Twoways effects Within Model
##
## Call:
## plm(formula = spending ~ trade + polity2 + dependency + gdpgrowth +
##     lag(spending), data = df, effect = "twoways", model = "within",
##     index = c("iso3c", "year"))
##
## Unbalanced Panel: n=158, T=5-52, N=6008
##
## Residuals :
##     Min.  1st Qu.   Median  3rd Qu.     Max.
## -31.1000  -0.7540  -0.0645   0.6360  25.9000
##
## Coefficients :
##               Estimate Std. Error t-value Pr(>|t|)
## trade          0.00738    0.00159    4.65  3.3e-06 ***
## polity2        0.01961    0.00793    2.47    0.013 *
## dependency     0.00604    0.00386    1.57    0.117
## gdpgrowth     -0.05459    0.00522  -10.45  < 2e-16 ***
## lag(spending)  0.79578    0.00693  114.77  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    95500
## Residual Sum of Squares: 27900
## R-Squared      :  0.708
##       Adj. R-Squared :  0.683
## F-statistic: 2810.31 on 5 and 5794 DF, p-value: <2e-16
```

Panel-corrected standard errors

```r
require(lmtest)
coeftest(model, vcov=function(x) vcovBK(x, type="HC1", cluster="time"))
```

```
##
## t test of coefficients:
##
##                Estimate Std. Error t value Pr(>|t|)
## trade           0.00738    0.00233    3.17   0.0015 **
## polity2         0.01961    0.00951    2.06   0.0392 *
## dependency      0.00604    0.00364    1.66   0.0971 .
## gdpgrowth      -0.05459    0.00725   -7.53    6e-14 ***
## lag(spending)   0.79578    0.02035   39.10   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Generate summary tables

These functions will generate publication-quality Latex tables of summary statistics of all the variables in your dataframe. To use these tables in a paper, you simply have to copy and paste into a Latex document what these functions print for you in the R console. These functions generate Latex code, which you can then typeset with Latex into an elegant paper.

```r
require(reporttools)
summaryvars<-subset(df, select=c("lending", "income", "imports", "exports", "industry"))
tableContinuous(summaryvars[,sapply(summaryvars, is.numeric)], font.size=12,
                longtable=FALSE, comment=FALSE, timestamp=FALSE,
                cap="Example Table for Continous Variables") # numeric variables only
```

Variable	**n**	**Min**	q_1	$\widetilde{x}$	$\bar{x}$	q_3	**Max**	**s**	**IQR**	**#NA**
imports	6576	0.1	22.3	32.2	38.6	48.0	424.8	27.3	25.7	1058
exports	6576	0.2	17.2	27.7	33.3	42.8	230.3	24.3	25.6	1058
industry	2403	2.1	19.8	24.3	24.7	29.9	59.6	7.8	10.1	5231

Table 1: Example Table for Continous Variables

```r
tableNominal(summaryvars[,!sapply(summaryvars, is.numeric)], font.size=12,
             longtable=FALSE, comment=FALSE, timestamp=FALSE,
             cap="Example Table for Categorical Variables") # only non-numeric variables
```

Variable	**Levels**	**n**	**%**	$\sum$**%**
lending	Aggregates	0	0.0	0.0
	Blend	488	6.4	6.4
	IBRD	2627	34.4	40.8
	IDA	2635	34.5	75.3
	Not classified	1884	24.7	100.0
	all	7634	100.0	
income	Aggregates	0	0.0	0.0
	High income: nonOECD	464	6.1	6.1
	High income: OECD	1494	19.6	25.6
	Low income	1676	21.9	47.6
	Lower middle income	2088	27.4	75.0
	Not classified	0	0.0	75.0
	Upper middle income	1912	25.1	100.0
	all	7634	100.0	

Table 2: Example Table for Categorical Variables

Generate publication-quality results tables

The `stargazer` function in the package of the same name will turn your model object into a nicely formatted table of results. It works just as the reporttools functions above. A nice little trick is to simply write the Latex code to a file, which will automatically save to your working directory. After running these lines, check your working directory for you Latex file called "model.tex". Typset this file in a Latex editor.

```r
require(stargazer)
model<-lm(spending ~ trade + polity2 + dependency + gdpgrowth + lag(spending), data=df)
stargazer(model,
          header=FALSE,
          title="Table of Regression Results",
          digits = 2,
          style = "apsr")
```

Table 3: Table of Regression Results

	spending
trade	-0.00^{***}
	(0.00)
polity2	-0.00^{***}
	(0.00)
dependency	0.00
	(0.00)
gdpgrowth	0.00^{***}
	(0.00)
lag(spending)	1.00^{***}
	(0.00)
Constant	-0.00^{***}
	(0.00)
N	6,094
R^2	1.00
Adjusted R^2	1.00
Residual Std. Error	0.00 (df = 6088)
F Statistic	$1{,}784{,}599{,}984{,}481{,}631{,}618{,}244{,}476{,}630{,}204{,}416.00^{***}$ (df = 5; 6088)

$^{*}p < .1$; $^{**}p < .05$; $^{***}p < .01$

Save your dataset to a file

Finally, this code chunk will save your dataframe "df" to a comma-separated text file, so you can easily open it in Stata, SPSS, Excel, etc.

```r
#set your working directory to whatever folder you wish
write.csv(df, "exported_data.csv")
```