Brute Force Regression

Analysis of UNESCAP Online Statistical Database

Jeffery M. Cooper, MAS
jefferymcooper@yahoo.com

Introduction

The following project is a demonstration of programming and analytic skills. This is in regards to a current UNESCAP opening for Statistician in the Statistics Division (Opening Number : 16-STT-ESCAP-54478-R-BANGKOK(G), closing June 12th 2016), that I am applying for.

My project entails scraping the entire database from the UNESCAP Online Statistical Database for UNESCAP countries (http://www.unescap.org/stat/data/statdb/DataExplorer.aspx) using **C#**, storing the data locally for analysis in **SQL Server** and finally running simple regression models on all combinations of Indicators using **R**.

Sample source code is included in the appendix, as well as code snippets through the analysis.

Jeffery M. Cooper, MAS
jefferymcooper@yahoo.com
+66 086 046 0091
Bangkok, Thailand
June 1st, 2016

Data Retrieval (C#)

The UNESCAP Online Statistical Database contains **1,026** Indicators, collected across **58** Areas (Countries), as you know. Downloading all Indicators from the website would be a tedious exercise, even with the option of dowloading multiple Indicators at a time. The solution is to "scrape" the data. That is automate this process by cycling through all combinations of Areas/Indicators - passing the HTTP requests and capturing the rerturned data. This can be done in many languages. I chose C# which has a rich set of libraries for crafting HTTP requests as well as JSON parsing.

This process is aided by the fact that UNESCAP exposes the data via a web service :

http://www.unescap.org/stat/data/statdb/StatWorksDataService.svc/GetData

We only need to find (1) the signature of the web service request and (2) the definitive lists of Areas and Indicators (ID's) to pass to t he service.

The web service signature is easily obtained with an HTTP sniffer such as Fiddler. We can monitor the the HTTP request when calling the webservice to see the shape of the parameter/value signature, for example :

GetData?indicatorIDsUnitIDs=[{"ID":"2406","ID2":"880"}]&areaIDs=["54"]&includeMetaData=false

As for the list of Indicator ID's and "Measure ID's" (what I call the **ID2** field above), these can be obtained from the main aspx page, **DataExplorer.aspx** parsing the html input nodes (partially shown) :

<input name="**indicatorsXml**" type="hidden" id="indicatorsXml" style="visibility:collapse" value="<Indicators><tvn text="Demographic trends" … >
<input name="**serieslistXml**" type="hidden" id="serieslistXml" style="visibility:collapse" value="<Series><item text="Afghanistan" id=" ... >

Data Retrieval (C#) cont.

These input lists are parsed into C# data structures (ArrayList's). We then loop through each Indicator (keeping the fixed list of all ESCAP countries constant) and make a call to the web service. In C# we encapsulate this process in a method like so :

```
public static void CallGetDataWebService(string id, string uid, string[] areas, string indText)
{

  string jsonFile = id + "-" + uid + ".json";

  string qq = ((char)34).ToString(); // double quote

  string url = "http://www.unescap.org/stat/data/statdb/StatWorksDataService.svc/GetData";

  string indList = WebUtility.UrlEncode("[{\"ID\":" + qq + id.ToString() + qq + ",\"ID2\":" + qq + uid.ToString() + qq + "}]");
  string areaList = WebUtility.UrlEncode("[" + qq + String.Join(qq + "," + qq, areas) + qq + "]");
  string query = String.Format("indicatorIDsUnitIDs={0}&areaIDs={1}&includeMetaData=false", indList, areaList);
  string uri = url + "?" + query;

  string json = GetJson(uri);
  bool success = SaveToDisk(json, jsonFile);
  if (success) { Console.WriteLine("{1} -> {0}", jsonFile, indText); }

}
```

Each call returns a chunk of JSON. We save this to disk for the next step of processing. We could parse the JSON at runtime, but because of the complexity of the parsing involved it's better to save locally and not keep hitting the UNESCAP web server while testing ! We end up with N = 1,026 JSON files, one for each Indicator.

Note : the Indicator hierarchy is contained in the HTML input list (e.g. **Population size [Thousands]** is a child of the folder **/Demographic trends/Population/Composition**). This is maintained and saved to a SQL structure for later use.

Data Retrieval (C#) cont.

After saving all JSON files (Indicator data for all UNESCAP countries) to disk, we need to import this data, along with the Area and Indicator lists, to SQL Server tables for easier processing. Once inside SQL, we can access the stats data from R scripts and run our regression models programatically.

We simply deserialize each JSON file int o a C# class which parallels the internal JSON structure, e.g. :

```
public class Datum
{
 [JsonProperty("__type")]
 public string Type;

 [JsonProperty("AreaID")]
 public int AreaID;

 [JsonProperty("DPs")]
 public DP[] DPs;

 [JsonProperty("IndID")]
 public int IndID;

 [JsonProperty("UnitID")]
 public int UnitID;
}
```

then flatten these values and create SQL insert statements (partial code) :

```
  JStats stats = JsonConvert.DeserializeObject<JStats>(json);
  JStats.D2 d = stats.D;
  .
  .
  SqlCommand cmd = new SqlCommand("INSERT INTO un.Stats (AreaID, TimeID, KpiID, MeasureID, Value) VALUES (@AreaID, @TimeID, @KpiID, @MeasureID,
    @Value)");
  cmd.CommandType = System.Data.CommandType.Text;
  cmd.Connection = connection;
  cmd.Parameters.AddWithValue("@AreaID", datum.AreaID);
  cmd.Parameters.AddWithValue("@TimeID", dp.TimeID);
  cmd.Parameters.AddWithValue("@KpiID", kpiID);
  cmd.Parameters.AddWithValue("@MeasureID", measID);
  cmd.Parameters.AddWithValue("@Value", dp.Value);
```

Data Retrieval (C#) cont.

Note that at the same time we parse the stats data from the JSON file we also parse the period of the Indicator :

```
public class Time
{
   [JsonProperty("__type")]
   public string Type;

   [JsonProperty("Chd")]
   public object Chd;

   [JsonProperty("ID")]
   public int ID;

   [JsonProperty("ID2")]
   public object ID2;

   [JsonProperty("Name")]
   public string Name;

   [JsonProperty("data")]
   public bool Data;
}
```

.. and insert into a SQL table as well :

```
SqlCommand cmd = new SqlCommand("INSERT INTO un.Time (KpiID, MeasureID, TimeID, Label) VALUES (@KpiID, @MeasureID, @TimeID, @Label)");
cmd.CommandType = System.Data.CommandType.Text;
cmd.Connection = connection;
cmd.Parameters.AddWithValue("@KpiID", kpiID);
cmd.Parameters.AddWithValue("@MeasureID", measID);
cmd.Parameters.AddWithValue("@TimeID", time.ID);
cmd.Parameters.AddWithValue("@Label", time.Name);
```

The time is not a single value, but a range of Years in most case (2000, 2001,…) with a corresponding Indicator value. Some Indicators do not use single years but a contiguous sets of years e.g. 1990-1995, 1995-2000, 2000-2005, … etc. This will be important when we run our regession models later.

Data Storage (SQL Server) cont.

Once inside SQL Server, we can create Views and Procedures to facilitate easier processing by R. I expect my SQL Server tables are similar to what you have at ESCAP (different object names of course). Data "sanity" checks can be performed as well to gauge what exact Indicators will be used in our Regression models (not al Indicators are reported every year as you know, so the Time we choose for our model will affect our processing).

I call the Indicator table **KPI** in my database for brevity. From C# I simply created an INSERT script and populate the corresponding SQL table. Here is a sample of the table (this is actually a view on the table called **v_KPI** since we just want to look at KPI's not folders) :

KMID	KpiID	MeasureID	LongName	ShortName	MeasureName	PathID	Path	ParentFolderID	ParentFolder
4	2406	880	Population size [Thousands]	Population size	Thousands	12	/Demographic trends/Population/Composition	3	Demographic trends
5	442	2527	Population growth [% change per annum]	Population growth	% change per annum	12	/Demographic trends/Population/Composition	3	Demographic trends
6	1072	880	Population: men [Thousands]	Population: men	Thousands	12	/Demographic trends/Population/Composition	3	Demographic trends
7	1071	880	Population: women [Thousands]	Population: women	Thousands	12	/Demographic trends/Population/Composition	3	Demographic trends
8	735	2514	Population sex ratio [Males per 100 ...	Population sex ...	Males per 100 females	12	/Demographic trends/Population/Composition	3	Demographic trends
9	736	2515	Child sex ratio [Boys per 100 girls]	Child sex ratio	Boys per 100 girls	12	/Demographic trends/Population/Composition	3	Demographic trends

with columns :

KMID - a unique ID (a proxy for KpiID-MeasureID to make processing easier)
KpiID - the **ID** field from source data
MeasureID - the **ID2** field from the source data (also called UnitID)
LongName - the **text** field from the source data
ShortName - (calculated) LongName minus the suffix * *KPI's Only (NULL for folders)*
MeasureName - (calculated) suffix from LongName (e.g. "Thousands") * *KPI's Only (NULL for folders)*
PathID/Path - (calculated) entire parent path
ParentFolderID/ParentFolder - (calculated) top-level parent folder

Data Storage (SQL Server) cont.

The rest of the tables are imported as is and look very much like their source data (partial data) :

Stats

AreaID	TimeID	KpiID	MeasureID	Value
4	319	1070	4065	21.7
4	323	1070	4065	26.9
5	317	1070	4065	7.2
5	324	1070	4065	7.9
22	325	1070	4065	11.4
28	321	1070	4065	9.1
28	327	1070	4065	9.2
29	318	1070	4065	13
29	323	1070	4065	10.9
33	318	1070	4065	21.7
33	320	1070	4065	28.6
33	323	1070	4065	27.6
33	327	1070	4065	39

Area

AreaID	Name
15	Afghanistan
18	American Samoa
29	Armenia
24	Australia
22	Azerbaijan
28	Bangladesh
33	Bhutan
41	Brunei Darussalam
5	Cambodia
54	China
60	Cook Islands
119	DPR Korea

Time

KpiID	MeasureID	TimeID	Label
1524	880	327	2013
1524	880	328	2014
1524	880	329	2015
1524	880	330	2016
1524	880	331	2017
1524	880	332	2018
1524	880	333	2019
1524	880	334	2020
1524	880	335	2021
1524	880	336	2022
1524	880	337	2023
1524	880	338	2024

Table **Stats** is large as expected, N = 1,218,019 records (each KpiID/MeasureID x TimeID x Area). Since we have ~1,000 KPI's we can have potentially n * (n-1) / 2 regression models => 499,500 (for a single fixed Period). However many KPI's are not collected every year. Additionally we only process models with n >= 30 obs for significance. As you will see this brings our number of models way down. We have one more SQL table called **Reg** where our regression results will be stored. This is detailed in the next section.

Data Analysis (R)

Now that we have all our stats and corresponding lookup data handy in SQL (Area, Kpi, Time) we can use R to calculate simple regression on all KPI pairs. The results of the regression analysis will be saved back to a SQL Server table. There's no clean way to save this in R (R's data structures called "frames" do not get materialized to disk). We could write these out to text files, but it makes sense to save them to SQL Server where we can query the 1000's of models easily.

First off we can calculate the available number of KPI's by Year (TimeID). We just want to foucs on a single year's data for this pilot project. We don't want year ranges either (e.g. Time = 1995-2000). Using some SQL magic we arrive at this summary :

Year	HasInsuffData	KpiOK	Total
2005	364	399	763
2006	377	393	770
2007	370	394	764
2008	368	398	766
2009	375	399	774
2010	372	424	796
2011	275	400	675
2012	297	405	703
2013	267	369	637
2014	239	267	507
2015	74	80	154

Where **HasInsuffData →** KPI's with < 30 observations for year, and **KpiOK →** KPI's with >= 30 (non-zero) observations for year.

We exclude any "subsetted" KPI's (e.g. **Population, female, 0-4 years [Thousands]**) which are simply a subset of an existing KPI. Most of these exist in path **/Demographic trends/Population/Age structure (5 years range)**.

Clearly 2015 is pretty lean with only 154 KPI's with valid data. Thus, we will use Year = 2010 as it contains a good amount of valid KPI's (N=424) and is fairly recent (I expect there are census level KPI's in 2010).

Note that the actual data actually goes as far back as 1948 for some KPI's (and as far ahead as 2050), but you already know this.

Using this list of valid KPI's (N = 424) we can then calculate the total number of regression models, using the following criteria (these are filters applied **after** we cross each KPI with another):

* No need to run regression on y~x and x~y (only y~x)
* Do not run model if both KPI's belong to the same folder (Path)

Using this criteria we arrive at **N = 86,634** regression models.

Data Analysis (R) cont.

After producing this list of KPI pairs (regression models) our plan of attack in **R** is :

* Set up loop to cycle through all combinations of KPI's
* Feed each KPI pair to a regression function, this is **lm()** in R
* Capture regression output and save back to SQL Server (stats and parameter estimates)

R can easily read/write to SQL using it's RODBC library. The trick is "flattenting" out the regression output so we can save in a relational table. R (like SAS, SPSS …) displays its regression results as a chunk of formatted text, e.g. :

```
lm(formula = YValue ~ XValue, data = datx)

Residuals:
     Min       1Q   Median       3Q      Max
-2691727  -212329   108971   161709  2824867

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.648e+05  1.347e+05  -1.224    0.229
XValue       3.025e+01  8.716e-01  34.699   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 787900 on 37 degrees of freedom
Multiple R-squared:  0.9702,     Adjusted R-squared:  0.9694
F-statistic:  1204 on 1 and 37 DF,  p-value: < 2.2e-16
```

Luckily, R makes it straight forward to extract individual statistics into local variables :

```
# get reg stats
mod = lm(YValue ~ XValue, data = dat2)
rsquared = summary(mod)$r.squared
arsquared = summary(mod)$adj.r.squared
```

Data Analysis (R) cont.

We create a wrapper function around **lm()** which flattens out all our desired statisitics and parameter estimates into a single row (a vector in R terminology) . Here is a partial listing of the code :

```
RegressionIt <- function(regdat, xcol, ycol, trimper = NULL)
{
  out <- vector("list", length(15))
  err_code = 0

  # trim obs, if passed
  if(!is.null(trimper)){
    n = length(regdat[,1])
    trim = round(length(regdat[,1]) * trimper)
    # re-order by X
    regdat <- regdat[order(regdat[, xcol]),]
    # remove top and bottom  trimper % of obs
    regdat <- regdat[(trim+1):(n-trim),]
  }

  # run simple regression model : y = ax + b
  mod <- lm(regdat[, ycol] ~ regdat[, xcol], data = regdat)

  # get resids for later testing
  stdres = rstandard(mod)

  # Collinear Model
  if(!is.null(alias(mod)$Complete[1])){
    out[15] = -1 # err_code
    return(out)
  }

  # get reg stats
  rsquared = summary(mod)$r.squared
  arsquared = summary(mod)$adj.r.squared
  coeffs <- unname(summary(mod)$coefficients)

  # wilks test
  shap <- shapiro.test(stdres)
  wilk = unname(shap$statistic)
  wpval = unname(shap$p.value)
.
.
  out <- c(rsquared, arsquared, coeffs[1,1], coeffs[1,2], coeffs[1,3], coeffs[1,4], coeffs[2,1], coeffs[2,2], coeffs[2,3], coeffs[2,4], wilk,
  wpval, f, f_pval, err_code)
  return(out)

}
```

Data Analysis (R) cont.

The output of function **RegressionIt()** is a data frame (with corresponding SQL table **Reg**):

R

```
rows <- data.frame(
  #x_name = character()
  #, y_name = character()
  x_kmid = integer()
  , y_kmid = integer()
  , timeid = integer()
  , n = integer()
  , rsq = double()
  , arsq = double()
  , b_est = double()
  , b_se = double()
  , b_tval = double()
  , b_pval = double()
  , a_est = double()
  , a_se = double()
  , a_tval = double()
  , a_pval = double()
  , wilks = double()
  , wilks_pval = double()
  , f = double()
  , f_pval = double()
  , stringsAsFactors = FALSE
)
```

SQL

```
CREATE TABLE un.Reg
(
  x_kmid INT
  , y_kmid INT
  , timeid INT
  , n INT
  , rsq FLOAT
  , arsq FLOAT
  , b_est FLOAT
  , b_se FLOAT
  , b_tval FLOAT
  , b_pval FLOAT
  , a_est FLOAT
  , a_se FLOAT
  , a_tval FLOAT
  , a_pval FLOAT
  , wilks FLOAT
  , wilks_pval FLOAT
  , f FLOAT
  , f_pval FLOAT
  , model VARCHAR(20)
  , error_desc varchar(50) NULL
)
```

Our R function simply exports the frame data to SQL as a single row. Sample SQL data from table **Reg** (not all columns shown) :

x_kmid	y_kmid	timeid	n	rsq	arsq	b_est	b_se	b_tval	b_pval
4	149	328	51	0.00773339140024835	-0.0125169475507672	13.9740358527981	1.44983895640595	9.63833658287041	6.73979033093197E-13
4	446	324	49	0.59832994658515	0.589783775235897	-143.718007778013	295.299115804887	-0.486686210984025	0.62874264772341
17	929	324	46	0.225614865550667	0.208015203404091	38278.927820381	8551.28050083512	4.47639716842906	5.31583812656546E-05
4	160	328	45	0.00193856345499283	-0.0212721676274492	89.316862745147	14.2262008185409	6.27833557844487	1.43886602743 31E-07
108	982	324	43	0.000548778389518924	-0.0238280806741513	21.1509108408516	3.73336681793774	5.6653717334251	1.29336176258087E-06
114	528	324	30	0.00390755213526799	-0.0316671781456155	161727.088375298	548428.952793136	0.294891594529475	0.770252184698165
114	531	324	35	0.00892839559382054	-0.0211040772669728	8348.64403534316	18672.0262498702	0.447120410159086	0.657709104425562
114	532	324	33	0.0151957789175538	-0.0165720991818801	892.05669295592	9016.50532321369	0.0989359692007557	0.92182611242092

Data Analysis (R) cont.

After running our ~86K regression models (around 3 hours on my laptop at 7 seconds or so per model), we can now analyze our data. Note that some models still fell short of N = 30 observations (even thought we pre-screend the KPI's). This is because when we construct KPI pairs, it's possible a pair doesn't join at all data points (across AreaID). This was a small percentage of models however (11,302/86,634 or 13%). So we end up with N' = **75,332** valid models.

The goal of this exercise is to numerically identify "good" models according to our collected statistics (without initially using the usual x-y plots, which need eyeballed. Not an easy thing to do with 73K models). Among these stats are the F statistic/p-value (to look at the model fit) as well as Shapiro-WIlks test for normality along with it's p-value. Looking at only models with a high R^2 (>=0.95), with Wilks p-value > 0 (value rounded to 4 places), we find N = **24** models. Ordering by Wilks p-value (descending) :

	X	Y	x_kmid	y_kmid	N	R-Squared	F (p-val)	Wilks (p-val)
1	Average annual GDP (2005 US dollars) growth rate [% c...	Average annual GDP per capita (2005 US dollars) gro...	781	825	54	0.9764	0.0000	0.4064
2	Medium size cities (More than 300 000 people in 2014)...	GDP by activity: Agriculture, hunting, forestry, fi...	118	784	35	0.9896	0.0000	0.0078
3	Medium size cities (More than 300 000 people in 2014)...	GDP by activity: Agriculture, hunting, forestry, fi...	118	786	35	0.9896	0.0000	0.0078
4	Methane emissions (CH4) from agriculture [Thousand to...	Agricultural area irrigated [Thousand hectares]	440	532	39	0.9560	0.0000	0.0034
5	Nitrous oxide (N2O) emissions [Thousand tons]	GDP by activity: Agriculture, hunting, forestry, fi...	443	785	40	0.9869	0.0000	0.0019
6	Nitrogen consumption in nutrients [metric tons of nut...	Total employment [Thousands]	512	874	39	0.9822	0.0000	0.0018
7	Gross national income in current prices [US dollars p...	Labour productivity [Constant 2005 US dollar]	818	877	34	0.9627	0.0000	0.0014
8	Medium size cities (More than 300 000 people in 2014)...	Imports of merchandise from Africa [Million US doll...	118	917	31	0.9798	0.0000	0.0010
9	GDP by activity: Agriculture, hunting, forestry, fish...	Imports of merchandise from Africa [Million US doll...	784	917	42	0.9790	0.0000	0.0007
10	GDP by activity: Agriculture, hunting, forestry, fish...	Imports of merchandise from Africa [Million US doll...	786	917	42	0.9790	0.0000	0.0007
11	Chemicals and related products, n.e.s. (Imports SITC ...	Air transport passengers carried [Number]	948	1137	33	0.9551	0.0000	0.0006
12	Medium size cities (More than 300 000 people in 2014)...	Total employment [Thousands]	118	874	36	0.9682	0.0000	0.0004
13	Medium size cities (More than 300 000 people in 2014)...	Nitrous oxide (N2O) emissions [Thousand tons]	118	443	35	0.9541	0.0000	0.0003
14	Medium size cities (More than 300 000 people in 2014)...	GDP by activity: Agriculture, hunting, forestry, fi...	118	785	35	0.9768	0.0000	0.0003
15	Population: women [Thousands]	Phosphate consumption in nutrients [metric tons of ...	7	513	38	0.9727	0.0000	0.0003
16	Gross Domestic Product (GDP) [2005 US dollars per cap...	Labour productivity [Constant 2005 US dollar]	824	877	34	0.9873	0.0000	0.0003
17	Population: men [Thousands]	Phosphate consumption in nutrients [metric tons of ...	6	513	38	0.9766	0.0000	0.0002
18	Population: women [Thousands]	Agricultural area irrigated [Thousand hectares]	7	532	39	0.9635	0.0000	0.0001
19	Population size [Thousands]	Phosphate consumption in nutrients [metric tons of ...	4	513	39	0.9748	0.0000	0.0001
20	Methane emissions (CH4) from agriculture [Thousand to...	Phosphate consumption in nutrients [metric tons of ...	440	513	38	0.9552	0.0000	0.0001
21	Phosphate consumption in nutrients [metric tons of nu...	Road traffic deaths [Number]	513	1139	36	0.9715	0.0000	0.0001
22	Greenhouse gas (GHG) emissions from agriculture [Thou...	Agricultural area irrigated [Thousand hectares]	412	532	39	0.9535	0.0000	0.0001
23	Greenhouse gas (GHG) emissions from agriculture [Thou...	Nitrogen consumption in nutrients [metric tons of n...	412	512	40	0.9828	0.0000	0.0001
24	Population size [Thousands]	Agricultural area irrigated [Thousand hectares]	4	532	39	0.9654	0.0000	0.0001

Data Analysis (R) cont.

Note 3 of the top 5 models have the same response KPI, but at different measures (units). As such they all have similar statistics.

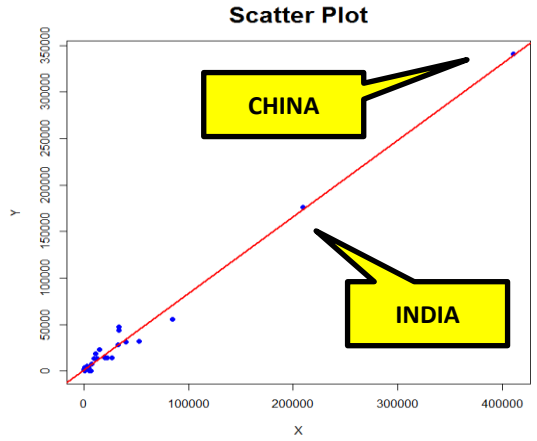
Kpiid	MeasureID	LongName
463	2312	GDP by activity: Agriculture, hunting, forestry, fishing [National currency, millions (constant 2005)]
463	454	GDP by activity: Agriculture, hunting, forestry, fishing [Million US dollars]
463	2311	GDP by activity: Agriculture, hunting, forestry, fishing [Million 2005 US dollars]

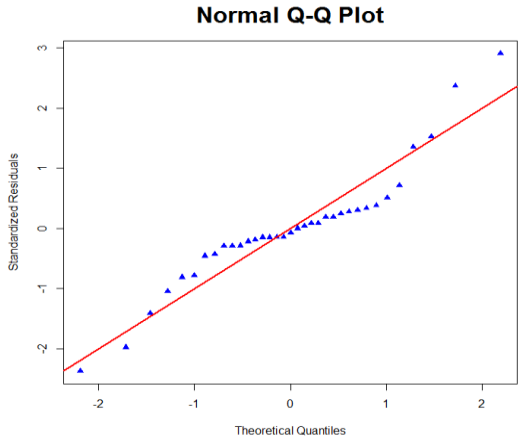
Additionally, the top model's X & Y variables appear to be derived from each other although they are different KPI's :

KpiID	MeasureID	LongName	Path
751	2527	Average annual GDP (2005 US dollars) growth rate [% change per annum]	/Economy/GDP by expenditure
753	2561	Average annual GDP per capita (2005 US dollars) growth rate [% change per capita per annum]	/Economy/GDP per capita

We exclude this model. All remaining models shown have very small p-values for their F-statistic. Note we don't have any models whose residuals indicate normality. The highest Shapiro-Wilks statistic is w= 0.008, which would reject the null hypotheses of H_0 : *Residuals are normally distributed*, even at p = 0.01. Looking at the scatter plots, it's obvious that outliers (very large countries in particular) are affecting the residuals normality assumption. Takling a look at the 2nd model in the list :

GDP by activity: Agriculture, hunting, forestry, fishing [National currency, millions (constant 2005)] ~
***f* (Medium size cities (More than 300 000 people in 2014) [Thousand people])**

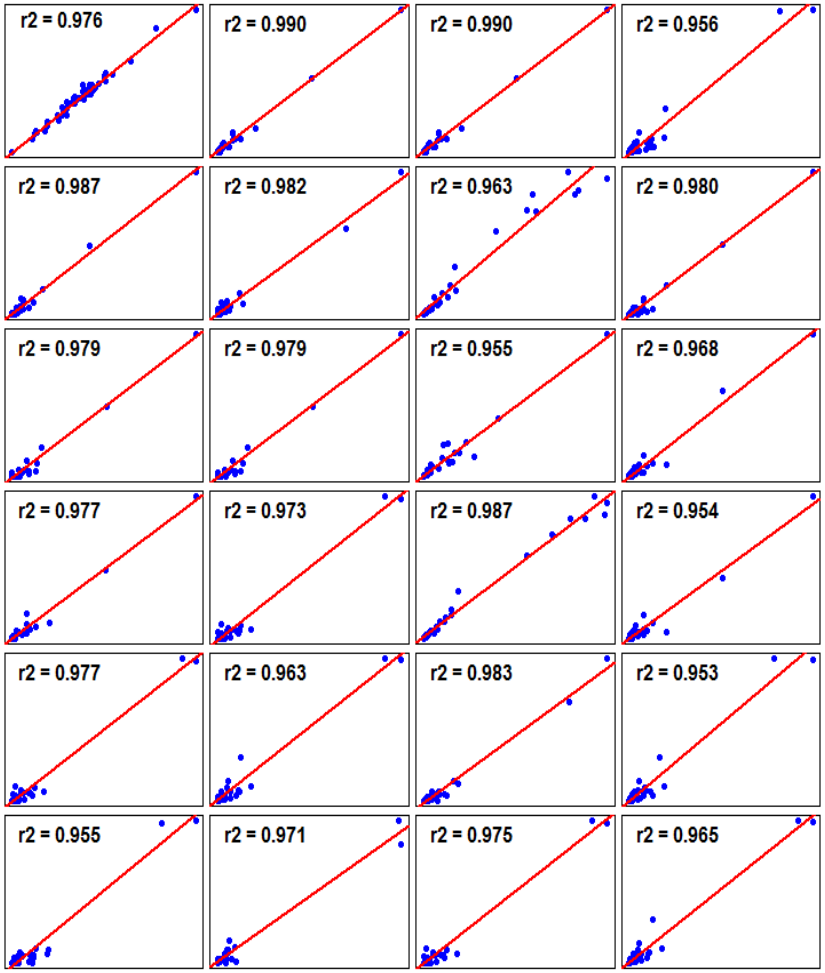




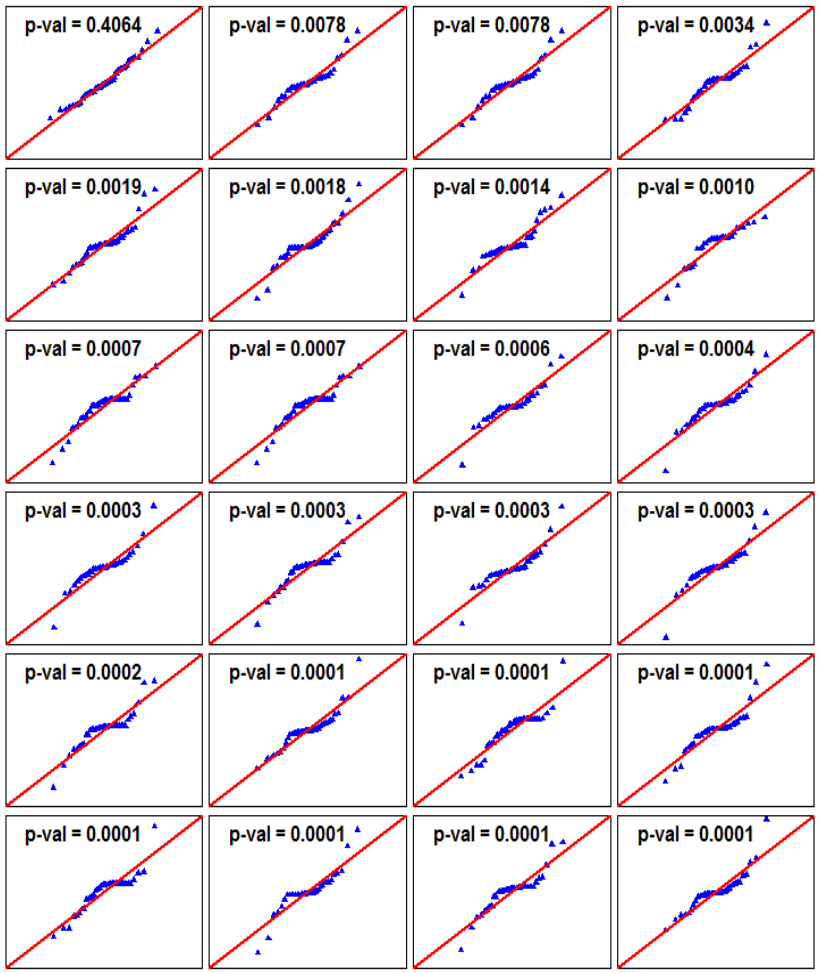
Data Analysis (R) cont.

We can see this pattern in the QQ Plot for all 24 models (left-to-right, top-to-bottom according to Wilks p-value descending) :

X-Y Scatter Plot



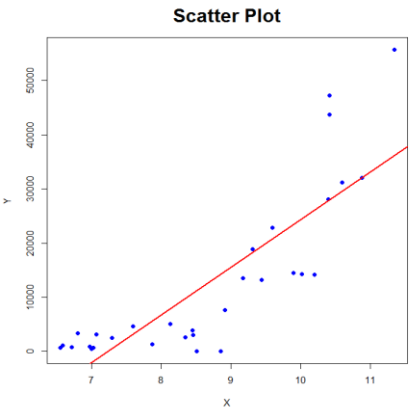
QQ Normal Plot

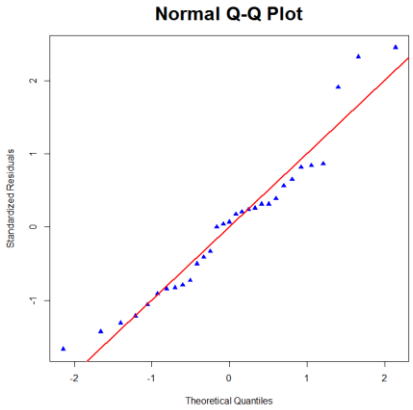


Data Analysis (R) cont.

We can dampen the outlier affect by excluding the larger countries (India, China) as well as a transformation such as the natural log :

GDP by activity: Agriculture, hunting, forestry, fishing [National currency, millions (constant 2005)] ~ $\log_n$ (Medium size cities (More than 300 000 people in 2014) [Thousand people]) *





*** India & China exlcuded**

This lowers our R^2 = **0.6954**, but increases our Wilks p-value = **0.1289** (which doesn't reject the hypotheses of normality on the residuals). This kind of manipluation of the model is dangerous however - we are just tweaking the data set to create a better-fitting model. Since there is an obvious effect from the size of the country, we probably need to include another term in all of our models (multiple regression). The obvious candidate is using the population size, which is in fact one of our KPI's : **Population size [Thousands]** (shows up in many of our models , see previous list).

We didn't find many significant models unfortunately. Extending this "brute force regression" idea to transformed models or multiple regression models (with some proxy for country size as an independent variable) would not be difficult with the existing framework. For multiple regression (just 2 independent variables) we would be dealing with millions of models (a factor of 1026 choose 2). This is still feasible, although a faster computer would be in order (e.g. multiple processor server).

APPENDIX

Sample Code - C#

```
        public static void ParseStatsFromJson(string fileName, SqlConnection connection)
        {

            string json = "";
            bool success = ReadFromDisk(ref json, fileName, "..\\..\\json"); // dec data

            int kpiID;
            int measID;
            Match match = Regex.Match(fileName, @"(\d+)-(\d+)\.json");
            kpiID = Convert.ToInt32(match.Groups[1].Value);
            measID = Convert.ToInt32(match.Groups[2].Value);

            int recCount = 0;

            JStats stats = JsonConvert.DeserializeObject<JStats>(json);
            JStats.D2 d = stats.D;

            // data!
            JStats.Datum[] data = stats.D.Data;
            for (int i = 0; i <= data.GetUpperBound(0); i++)
            {
                JStats.Datum datum = data[i];
                 JStats.DP[] dps = datum.DPs;
                for (int j = 0; j <= dps.GetUpperBound(0); j++)
                {

                    JStats.DP dp = dps[j];

                    SqlCommand cmd = new SqlCommand("INSERT INTO un.Stats (AreaID, TimeID, KpiID, MeasureID, Value) VALUES (@AreaID, @TimeID,
@KpiID, @MeasureID, @Value)");
                    cmd.CommandType = System.Data.CommandType.Text;
                    cmd.Connection = connection;
                    cmd.Parameters.AddWithValue("@AreaID", datum.AreaID);
                    cmd.Parameters.AddWithValue("@TimeID", dp.TimeID);
                    cmd.Parameters.AddWithValue("@KpiID", kpiID);
                    cmd.Parameters.AddWithValue("@MeasureID", measID);
                    cmd.Parameters.AddWithValue("@Value", dp.Value);
                    cmd.ExecuteNonQuery();

                    recCount++;

                }

        }
```

APPENDIX

Sample Code - C#

```
            Console.WriteLine("{0} records written to un.Stats", recCount.ToString());
            recCount = 0;

            // times
            for (int k = 0; k <= d.Times.GetUpperBound(0); k++)
            {

                JStats.Time time = stats.D.Times[k];
                //Console.WriteLine(time.ID + " -> " + time.Name);

                SqlCommand cmd = new SqlCommand("INSERT INTO un.Time (KpiID, MeasureID, TimeID, Label) VALUES (@KpiID, @MeasureID, @TimeID,
@Label)");
                cmd.CommandType = System.Data.CommandType.Text;
                cmd.Connection = connection;
                cmd.Parameters.AddWithValue("@KpiID", kpiID);
                cmd.Parameters.AddWithValue("@MeasureID", measID);
                cmd.Parameters.AddWithValue("@TimeID", time.ID);
                cmd.Parameters.AddWithValue("@Label", time.Name);
                cmd.ExecuteNonQuery();

                recCount++;

            }

            Console.WriteLine("{0} records written to un.Stats", recCount.ToString());

            Console.WriteLine("File {0} processed\n", fileName);

        }
```

APPENDIX

Sample Code - R

```
PlotIt <- function(type)
{

                # open sql server conn'n
                conn <-odbcConnect("LOCAL_2014EXP_X64", uid="dev", pwd="mom5069!")

                sql = "
                select
                  x_kmid
                  , y_kmid, timeid
                  , CAST(ROUND(rsq, 3) AS DEC(4,3)) as [rsq]
                  , CAST(ROUND(wilks_pval, 4) AS DEC(5,4)) as [wilks_pval]
                from DEV.un.Reg r
                where
                  error_desc = ''
                  and TimeID = 324
                  and model = 'y~x'
                  and rsq >= 0.95
                  and r.wilks_pval > 0.00009
                order by
                  wilks_pval DESC"

                # replace newline & tab
                sql = gsub("[\n\t]", " ", sql)
                loop <- sqlQuery(conn, sql, errors = TRUE)

                # we have 24 plots, so create an 6x4 grid
                par(mfrow=c(6,4), mar=c(0.25,0.25,0.25,0.25))

                for(i in 1:length(loop[,1]))
                {
                                m_kmid1 = loop[i,1]
                                m_kmid2 = loop[i,2]
                                m_timeid = loop[i,3]
                                m_rsq = round(loop[i,4], 3)
                                m_wilksp = round(loop[i,5], 4)

                                # debug
                                #print(sprintf("%d -> %d (%d)", m_kmid1, m_kmid2, m_timeid))
```

APPENDIX

Sample Code - R

```
                                sql = "SET NOCOUNT ON\nEXEC DEV.un.usp_GetRegData %d, %d, %d"
                                xsql = sprintf(sql, m_kmid1, m_kmid2, m_timeid)
                                dat <- sqlQuery(conn, xsql, errors = TRUE)
                                mod <- lm(YValue ~ XValue, dat) # regn

                                if(type == "qq"){
                                                lbl = sprintf("p-val = %f", m_wilksp)
                                                lbl = substr(lbl, 1, nchar(lbl) - 2) # sprintf is padding numerics with zeros!
                                                qqnorm(rstandard(mod), xlab = "", ylab = "", main = "", pch=17, col="blue", xaxt='n', yaxt='n',
ann=FALSE, xlim=c(-4.0,4.0), ylim=c(-4.0,4.0))
                                                abline(a=0,b=1, lwd=2, col="red")
                                                text(-3.75, 3.25, lbl, col="black", cex=1.5, pos=4, font=2) #bold
                                }
                                else{
                                                lbl = sprintf("r2 = %f", m_rsq)
                                                lbl = substr(lbl, 1, nchar(lbl) - 3) # sprintf is padding numerics with zeros!
                                                plot(dat$XValue, dat$YValue, xlab = "", ylab = "", main = "", pch=16, col="blue", xaxt='n',
yaxt='n', ann=FALSE)
                                                abline(mod, lwd=2, col="red")
                                                text(min(dat$XValue) * 0.95, max(dat$YValue) * 0.90, lbl, col="black", cex=1.5, pos=4, font=2)
                                }

                                # test
                                #if(i==9) {break}
                }

                close(conn)

}
```

Errata

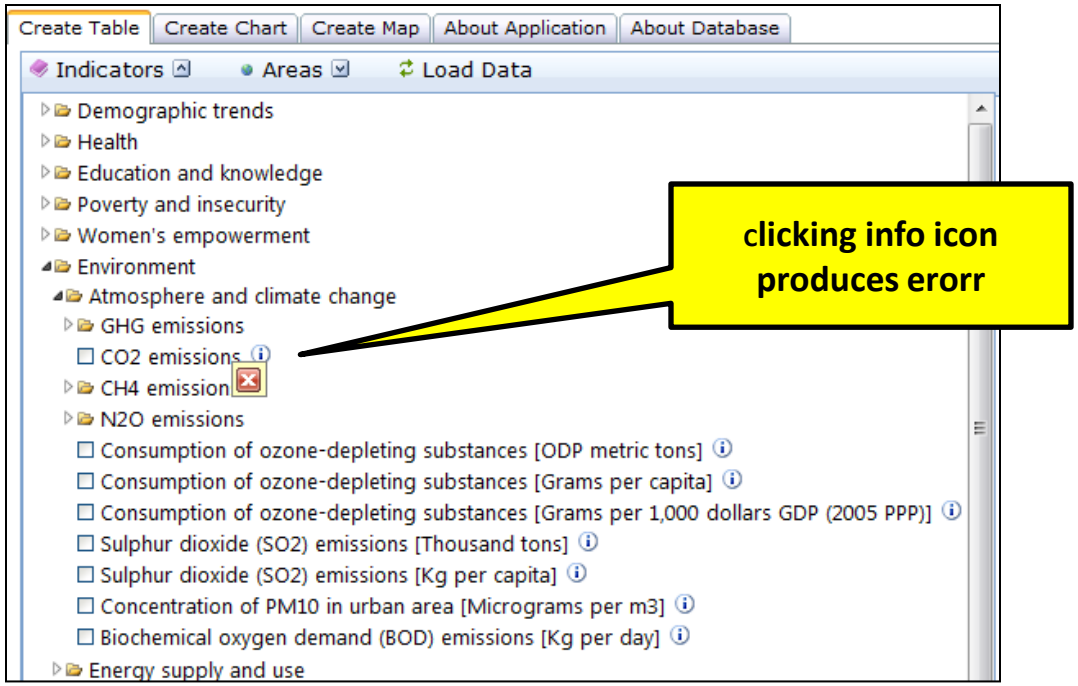
While developing this project I became very aquainted with your data. As such I stumbled upon a few anomalies (you may be aware of these already, if so please ingore!). Specifically I found (1) some bad “nodes” in the Indicator hierarchy and (2) ~~some duplicate Indicator values (duplicated for Indicator/Unit/Area/Time~~). **<CORRECTED ON WEBSITE AS OF 06/01>**

Indicator Hierarchy

	ID	ID2	Text	Path
1	3870	NULL	CO2 emissions	/Environment/Atmosphere and climate change
2	861	NULL	GDP (2005 PPP dollars)	/Economy/GDP (PPP)
3	3699	NULL	Share of persons in informal employment in total non-agricultural employment, total	/Economy/Employment/Employment and labour productivity
4	3701	NULL	Share of persons in informal employment in total non-agricultural employment, female	/Economy/Employment/Employment and labour productivity
5	3700	NULL	Share of persons in informal employment in total non-agricultural employment, male	/Economy/Employment/Employment and labour productivity
6	3801	NULL	ODA (Commitments) by purpose, total	/Economy/International financing/ODA/ODA by Purpose
7	3786	NULL	ODA to Social Infrastructure & Services	/Economy/International financing/ODA/ODA by Purpose
8	3787	NULL	ODA to Education	/Economy/International financing/ODA/ODA by Purpose
9	3788	NULL	ODA to Health and Population	/Economy/International financing/ODA/ODA by Purpose
10	3789	NULL	ODA to Economic Infrastructure & Services	/Economy/International financing/ODA/ODA by Purpose
11	3790	NULL	ODA to Energy	/Economy/International financing/ODA/ODA by Purpose
12	3791	NULL	ODA to Transport and Communication	/Economy/International financing/ODA/ODA by Purpose
13	3792	NULL	ODA to Production Sectors	/Economy/International financing/ODA/ODA by Purpose
14	3793	NULL	ODA to Agriculture, Forestry, Fishing	/Economy/International financing/ODA/ODA by Purpose
15	3794	NULL	ODA to Industry, Mining, Construction	/Economy/International financing/ODA/ODA by Purpose
16	3795	NULL	ODA to Trade & Tourism	/Economy/International financing/ODA/ODA by Purpose
17	3796	NULL	ODA to Multisector	/Economy/International financing/ODA/ODA by Purpose
18	3797	NULL	ODA to Food Aid	/Economy/International financing/ODA/ODA by Purpose
19	3798	NULL	ODA to Action Relating to Debt - - - - -	/Economy/International financing/ODA/ODA by Purpose
20	3799	NULL	ODA to Humanitarian Aid	/Economy/International financing/ODA/ODA by Purpose
21	3800	NULL	ODA to Other & Unallocated/Unspecified	/Economy/International financing/ODA/ODA by Purpose

These above Indicators (ID) have no Unit (ID2). On the web UI these will display an error when clicking on the “information” icon

Errata cont.



The web service fails as well when you try and “Load Data”.

~~Duplicate Values~~ ***<CORRECTED ON WEBSITE AS OF 06/01>***

Luckily I only found dupes for only 1 Indicator : **Contributing family workers, Female [% of employed females]** (ID/ID2 = **1603 / 2523**). You can see this clearly in the JSON file (the dupes span multiple countries, here is Thailand’s data) :

Errata cont.

http://www.unescap.org/stat/data/statdb/StatWorksDataService.svc/GetData?indicatorIDsUnitIDs=[{"ID":"1603","ID2":"2523"}]&areaIDs=["4"]&includeMetaData=false

We can see from a JSON sample :


```
{"__type":"DP:#SilverlightClient.WebSite","Dec":1,"MID":0,"TimeID":303,"Value":54.6},
{"__type":"DP:#SilverlightClient.WebSite","Dec":1,"MID":0,"TimeID":304,"Value":58.1},
{"__type":"DP:#SilverlightClient.WebSite","Dec":1,"MID":0,"TimeID":304,"Value":58.1},
{"__type":"DP:#SilverlightClient.WebSite","Dec":1,"MID":0,"TimeID":305,"Value":58.5},
{"__type":"DP:#SilverlightClient.WebSite","Dec":1,"MID":0,"TimeID":305,"Value":58.5},
{"__type":"DP:#SilverlightClient.WebSite","Dec":1,"MID":0,"TimeID":11,"Value":56.1},
{"__type":"DP:#SilverlightClient.WebSite","Dec":1,"MID":0,"TimeID":11,"Value":56.1},
{"__type":"DP:#SilverlightClient.WebSite","Dec":1,"MID":0,"TimeID":12,"Value":53.2},
{"__type":"DP:#SilverlightClient.WebSite","Dec":1,"MID":0,"TimeID":12,"Value":53.2},
.
.
```