Student Growth Percentiles: Introduction to R & the SGP Software Package.

Adam VanIwaarden
Ph.D. Candidate, University of Colorado at Boulder

An introduction to R…

What it R?

- Interpreted programming language tailored for statistics.
  - Instructions/commands are entered at a prompt, interpreted and executed before moving on to the next
  - In contrast to C, C++, FORTRAN, etc. that are compiled before execution. Some R functions use these languages to provide more efficiency.
- Graphical engine.
  - Produce plots and reports such as those shown here.
- Data manipulation and management to a lesser extent.

What it R?

- Object oriented language. Everything is an object.
  - Single characters or numbers.
  - Functions (which are often composed of other, more ‘primitive’ functions – R is mostly written in R).
  - Data
    - Data structures include vectors, matrices, arrays, data frames (an important variation of this is the data.table)
    - User can have multiple datasets open at one time unlike other stats software.
  - Results (model summaries, predictions, residuals). Probably the most important. In the end we want some product (statistics, graphics, etc.).

What it R?

- Objects can be classified by Mode or Class.
    - Mode is how R stores the object (function, list, numeric, character, factor, logical).
    - Class determines how functions treat the object. As data, graphical object, etc. Hundreds of classes (even SGP!).

- For those interested in a full evaluation of the R language, here is a detailed article:
    - http://www.cs.purdue.edu/homes/jv/pubs/ecoop12.pdf

Packages (Libraries)

- A “package” is a collection of functions, documentation, data and other stuff that are often tailored for a particular purpose. A user loads a package into R in order to use these functions. A collection of packages makes a “library”.
- R has a BASE package of statistical and graphical functions upon which many more can be added.
- Many R packages depend on other packages in order to work. That is, many R functions are built upon other R functions. This is one of the main advantages of open source R. You are free to build on, change or borrow other peoples work in order to create your own software.

Helpful Reference Sites

- http://cran.r-project.org/
- Quick R: http://www.statmethods.net/
- http://www.ats.ucla.edu/stat/r/
- http://www.statmethods.net/
- http://stackoverflow.com/questions/192369/books-for-learning-the-r-language
- R Community and blogs:
  - http://www.r-bloggers.com/
  - http://www.inside-r.org/howto/how-learn-r/
- R Courses (I've never taken one, but many praise them):
  - https://www.coursera.org/ (Free)
  - http://www.statistics.com/ (Paid)

The SGP Package

- First version available on CRAN in October, 2008.
- Now uses functions from 13 other R packages.
  - No longer "depends" on or "suggests" these packages (only the **`SGPdata`** package and the native **`parallel`** package)
  - Most essential dependency is **`rq`** function from **`quantreg`** – the quantile regression package developed by R. Koenker.
- Stable version is available on CRAN:

  http://cran.r-project.org/web/packages/SGP/index.html

- Development version (and visible code) is available on GitHub:
  - http://schoolview.github.com/SGP/
  - https://github.com/SchoolView/SGP

Installing the development version of SGP Package

- To install from Github you might need:
  - Windows: Rtools (http://cran.r-project.org/bin/windows/Rtools/),
  - OS X: xcode (from the app store)
  - Linux: apt-get install r-base-dev (or similar)

```
install.packages("devtools")

require(devtools)

install_github("SGP", "SchoolView", args="--byte-compile")

require(SGP)
```

The SGP Package

- Over 50 functions in the package.
  - Two key functions are **`studentGrowthPercentiles`** and `studentGrowthProjections`.
  - “Wrapper” functions for analyzing data. Higher level functions that do a lot of data management and bookkeeping, but are largely built around the two key lower level functions and their output.
  - Utility and convenience functions that perform tasks that are often repeated.   These help with the bookkeeping mentioned above, and are largely only used internally (they are not “exported” functions).
  - Visualization and data export functions.

The SGP Package

- Two “exemplar” data sets (**`sgpData`** and **`sgpData_LONG`**) and meta-data for state level assessment programs - **`SGPstateData`**.

Recent SGP Changes

- Mostly changes that let us run multiple content areas as priors.
  - spline matrix class
- Data set classes are now all character.
- Externalized many of the utility functions
- Visualizations: png output functionality (moving towards web based documentation)

SGP Central Functions

- **Basic SGP Analysis (Lower Level Functions)**
  - `studentGrowthPercentiles:` Function to calculate student growth percentiles using large scale assessment data. Outputs growth percentiles for each student and supplies various options as function arguments.
  - `studentGrowthProjections:` Function to calculate percentile growth projections/trajectories using large scale assessment data and results derived from student growth percentile calculation and a set of predefined scale score cut.
  - The SGP Primer (sgpPrimer.pdf) provides an introduction to these functions.
  - Data Note: These functions require a WIDE data format. The column order is critical for use with these functions. ID, grades, scale scores in that order (see **`sgpData`** as an exemplar). Or you can also use the **`panel.data.vnames`** argument to specify the column names in the correct order.

SGP Advanced Functions

- **Four-Step Analysis Functions**
  - **`prepareSGP:`** Data "preparation" (assumes data cleaning and formatting has already been done – **`sgpData_LONG`** is the ideal).
  - **`analyzeSGP:`** Data analysis – student growth percentiles and projections (also "baseline" reference group analyses).
  - **`combineSGP:`** Merge SGP results back into the LONG data file
  - **`summarizeSGP:`** Summarize the SGP results (Median SGPs aggregated by institutions, demographic groups, bootstrap confidence intervals, etc.).
  - Data Note: These functions require a LONG data format (see **`sgpData_LONG`** as an exemplar).

SGP Advanced Functions

- **Visualization and Output Functions**
  - **`visualizeSGP:`** Produce student growth reports, bubble plots and growth achievement plots.
  - **`outputSGP:`** Export long data (@Data) in either long or wide format. Export summary tables for use in the Adobe Air visualization tool. All files exported as a pipe delimited flat file (compressed).

SGP Object

- S4 object composed of 6 'slots'. Each slot contains other objects and information.
  - **`Version`** – meta-data about what version of the SGP package was used to create the object and when.
  - **`Names`** – meta-data about the variable names included in the LONG data file. Used to produce summaries.
  - **`Data`** – The LONG data file
  - **`Data_Supplementary`** – potentially many things (room for future expansion), but so far this is where teacher – student link data is stored.
  - **`SGP`** – the analysis results (`SGPercentiles` and `SGProjections` lists), as well as coefficient matrices, Goodness of Fit graphical objects, as well as the Knots, Boundaries and cutscore meta-data.
  - **`Summary`** – the summary tables produced.

Pre-Step 1:
Data Formatting and Prep

- There is nothing more important than setting up your data in the proper format.
  - True for using the 5 step functions, or with the basic studentGrowthPercentiles/Projections functions.
- See the SGP Primer for a description of an exemplar LONG data set for use with the 5 step functions.
- Most recent versions of SGP (>1.0-0.0) will now convert all variables to a character class.

Pre-Step 1: Common issues to look for

- Duplicate cases
  - Is there a business rule to choose one over another (e.g. highest scale score or other indicator)
- Unique Identifiers (student ID or school, teacher, etc.)
- Incorrect Achievement (Performance) Level assignment.
- Outliers
  - Non-tested grades (out of grade testing)
  - Scores above/below the highest/lowest obtainable scale score

Step 1:
Prepare an SGP Object.

```
> library(SGP)

> Demonstration_SGP <- prepareSGP(sgpData_LONG)



> class(Demonstration_SGP)

> slotNames(Demonstration_SGP)

> names(Demonstration_SGP@Data)

> class(Demonstration_SGP@Data)

> summary(Demonstration_SGP@Data)
```

Step 2: Analyze Data

- First lets take a look at the function arguments.
  - `?analyzeSGP`
  - `args(analyzeSGP)`
- `sgp_object` is the object we just created.
- The state argument tells the function where to look in the `SGPstateData` object for test related information.
  - `SGPstateData[[‘DEMO’]]`

Step 2 Interlude: SGPstateData

- State Assessment Data Includes
  - Knots and Boundaries. Set in advance and should not be changed once an analysis run. When these change, the results will change too!
    - Generally obtained from either empirical analysis. Default is to take the 20th, 40th, 60th and 80th percentiles of the distribution for the knots. Boundaries are ~10% above/below the highest/lowest obtainable scale score.
  - Cutscores. Performance level cuts determined by state/assessment vendor/etc. Outside knowledge and standard setting procedures, not empirical.
  - Assessment and reporting facts, information and other meta-data.
  - Misc. – Baseline coefficient matrices, variable name lookup tables, IRT CSEMs (conditional standard errors of measurement), etc.

Step 2: Analyze Data

- The `years`, `content_areas`, and `grades` arguments all are ‘optional’. They help construct the `sgp.config` list (another option below), If these arguments are not provided and you don’t provide the `sgp.config` list, the function will try to construct that list for you.
- For more complex analyses I like to specify the list myself. Here’s what one might look like…

Step 2: Analyze Data

Custom sgp.config list example:

```
ALGEBRA_I.2012_2013 = list(
        sgp.content.areas=c('MATHEMATICS', 'ALGEBRA_I'),
        sgp.panel.years=c('2011_2012', '2012_2013'),
        sgp.grade.sequences=list(c(8, 'EOCT')),
        sgp.exact.grade.progression=TRUE)
```

Required Element

Optional/context dependent Element

Step 2: Analyze Data

- `sgp.percentiles … sgp.projections.lagged.baseline`
  - Tells the function whether (`TRUE`) or not (`FALSE`) to run a particular type of SGP analysis.
    - Baseline analyses use multiple years of students to form the cohort (rather than a single year). May provide information on how/whether a system is changing over time (rather than re-norming year after year).
    - Lagged projections are used for growth-to-standard analyses (i.e. adequate growth). The current year of scores is removed (lagged), and projections are calculated using the current years' coefficient matrices.
    - `sgp.percentiles` must be run first before ANY projections can be run.

Step 2: Analyze Data

- `parallel.config` is an advanced feature for running analyses on multi-core workstations or multi-node clusters. This requires AMPLE amounts of memory and newer processors (and, god willing, NOT Window OS). Argument is a nested list:
  - `BACKEND` –one of two choices: `FOREACH or PARALLEL`. These are all parallel processing packages.
    - The `SNOW` implementation in the `PARALLEL` package works on Windows (kind of – Windows is not ideal)
    - *Everything* will work on Linux/Mac.
    - Second sub-list is either 1) `TYPE` = `SOCK` or `MPI` when SNOW/PARALLEL backend) or `TYPE` = `doParallel` for FOREACH.
  - `WORKERS`–a list specifying the number of workers used for each analysis type: `PERCENTILES=8`, `PROJECTIONS=6`, etc.

Step 2: Analyze Data, Let’s give it a go!

```
Demonstration_SGP <- analyzeSGP(
        Demonstration_SGP,
        sgp.config=my.config,
        sgp.percentiles.baseline = FALSE,
        sgp.projections.baseline = FALSE,
        sgp.projections.lagged.baseline = FALSE,
        simulate.sgps=FALSE,
        parallel.config=list(
                BACKEND="SNOW", TYPE="SOCK”,
                WORKERS=list(PERCENTILES=8,
                PROJECTIONS=8,
                LAGGED_PROJECTIONS=8)))
```

Step 2: Analyze Data, Some results

```
> names(Demonstration_SGP@SGP$Coefficient_Matrices$READING.2009_2010)
[1] "qrmatrix_4_1" "qrmatrix_5_1" "qrmatrix_6_1"

> names(Demonstration_SGP@SGP$SGPercentiles$READING.2009_2010)
[1] "ID"        "SGP"       "SGP_LEVEL”

> dim(Demonstration_SGP@SGP$SGPercentiles$READING.2009_2010)
[1] 12042     3

> summary(Demonstration_SGP@SGP$SGPercentiles$READING.2009_2010)
       ID               SGP            SGP_LEVEL
 Min.   :1000372   Min.   : 1.00   Very Low :2364
 1st Qu.:3159980   1st Qu.:25.00   Low      :2409
 Median :5406502   Median :50.00   Typical  :2525
 Mean   :5437710   Mean   :49.89   High     :2410
 3rd Qu.:7723936   3rd Qu.:75.00   Very High:2334
 Max.   :9999399   Max.   :99.00
```

Step 2 Interlude 2

- The `analyzeSGP` function is basically just a function that performs data management, reshaping and other ‘bookkeeping’ necessary to perform a series of analyses similar to what is presented in the SGP Primer included in the training materials and source code.
- `sgpData` is basically a re-shaped subset of `sgpData_LONG` for a single year/subject (2011 Reading) combination.

Step 3: Combine Results into @Data slot

- Much more simple function ☺
- Look at the names and dimension of data before and after (as well as summary)

```
names(Demonstration_SGP@Data)
dim(Demonstration_SGP@Data)

Demonstration_SGP <- combineSGP(Demonstration_SGP)

names(Demonstration_SGP@Data)
dim(Demonstration_SGP@Data)
summary(Demonstration_SGP@Data$SGP)
```

Step 3: Combine Results

- Most arguments are self-explanatory (hopefully)

- One important exception may be the last one:
  - `max.lagged.sgp.target.years.forward=4`
  - "A integer indicating the number of years forward from the lagged (last year's) score to project forward for growth to standard calculations. Default is 4 years from last year or 3 years from present, which is the standard in most growth to standard calculations used by state departments of education."
  - Only ends up making a difference for kids in lower grades that have (many) more years of testing ahead of them.
  - `head(Demonstration_SGP@SGP$SGProjections$READING.2010_2011.LAGGED)`
    - Only 3 possible years with the my.config list! 3rd – 6th grade.
  - `SGPstateData$DEMO$SGP_Configuration$max.order.for.projection`
    - Can specify this list element for your state in `SGPstateData` to force `analyzeSGP` to only project a given numbers of years forward.

Step 4: Summarize Results

- The LEAST intuitive of the SGP steps (we're working on it!).
- Important Concepts
  - Summaries
  - Groups
    - Summary
    - Confidence Interval
  - Variable Name Lookup
    - @Names slot
    - Recent addition as of version 0.9-0.0
    - The following slides show details needed to manually specify arguments that this feature replaces.

Step 4: Summarize Results, Parallel Processing

- As of version 0.9-0.0, use a parallel.config argument similar to other functions. For example, on Windows:

```
parallel.config=list(
  BACKEND="PARALLEL", TYPE="SOCK”,
  WORKERS=list(SUMMARY=8))
```

Step 5: Visualize Results

- Three types of visuals
  - Bubble Plots
  - Growth Achievement Plots
  - Student Growth Reports

- The later two can be produced in parallel. E.g.:

```
parallel.config=list(
        BACKEND="PARALLEL", TYPE="SOCK",
        WORKERS=list(GA_PLOTS=8, SG_PLOTS=8)))
```

Step 5: Visualize Results Bubble Plots

- Various “Styles” to choose from
  - Highlight particular schools, districts or teachers
  - Highlight levels of particular demographic subgroups (e.g. % of FRL or students with disabilities in a school, class, etc.)

  - Styles 1:3 are state level plots
  - Styles 10 & 11 are district level plots
  - Styles 50, 53, 57, & 59 and teacher/classroom level
  - Styles 100, 150, and 153 are student level plots (150 and 153 at by teacher / classroom)

Step 5: Visualize Results

Growth Achievement and Student Growth Reports

- Print or Presentation formats
- Student Growth reports for 2 or 3 subjects at this time
  - Hope to add functionality to do just 1 subject or more at a time
- Straight projections (**`sgp.projections=TRUE`** in **`analyzeSGP`**) are required for the "fans" for projected future growth.

All 5 in One!

- abcSGP is a wrapper function that can (in theory) run all 5 steps in one call:

```
Demonstration_SGP <- sgpData_LONG
Demonstration_SGP <- abcSGP(Demonstration_SGP,
       parallel.config=list(
              BACKEND="PARALLEL", TYPE="SOCK",
              WORKERS=list(
              PERCENTILES=8, BASELINE_PERCENTILES=8,
              PROJECTIONS=8, LAGGED_PROJECTIONS=8,
              SUMMARY=8, GA_PLOTS=8, SG_PLOTS=8)))
```

- Not sure I recommend this … ☺

Package Development

- The raw source code for the SGP package can be downloaded or browsed from the Github site. This provides a good sample of a package's components.
  - https://github.com/CenterForAssessment/SGP
  - This raw source code can be modified and built into your own private package.
  - If you have a Github account you can fork the repository and join our development effort (preferred!)

Debugging R & SGP Package

- Generally I use one of two processes:
  - options(error=recover)
    - Allows you to pick a point in the process where the error occurred and browse that portion of the “stack”.
  - debug(…)
    - Don’t need an error to bring up the browser.
    - Step though each line of the functions code:
      - Lets you see exactly what is being done (objects produced/modified) at each step.
      - Can be annoyingly long process to get where you want to be in the code…