DATA 698 Midterm Report: Executive Order 88 Energy Analytics & Data Cleansing

Dan Smilowitz

November 3, 2017

Background

On December 28, 2012, New York Governor Andrew Cuomo issued Executive Order 88 requiring that

> By April 1, 2020 . . . State Entities shall collectively reduce the average EUI [Energy Use Intensity] . . . by at least 20% from a baseline of the average EUI . . . for State fiscal year 2010/2011

The New York Power Authority (NYPA) was charged with establishing a management and implementation team to administer the executive order, directed to ensure agencies' compliance by implementing reporting requirements to document progress toward the target. In order to track performance against baselines, NYPA's BuildSmart NY team collects utility bill information for all fuels (e.g. electricity, natural gas, and water) for all covered facilities as compared to the square footage of each facility. This information is reported by each agency and submitted to NYPA in the form of an Excel spreadsheet template.

Challenges

Agencies are responsible for performing their own data validation — the performance of this task has proved incomplete. Many facilities have missing data for the baseline year, as well as showing large spikes or dips in reported usage believed to be due to data entry errors. NYPA must ensure accurate data is used for the establishment of baselines and the tracking and reporting of performance.

EO88 reporting data has been hosted by a software vendor responsible for the the creation and maintenance of NYPA's New York Energy Manager (NYEM) platform. The current structure of the data provides a great deal of redundancy and unused columns, harming the efficiency of computations based on this data. Finally, the vendor's platform has been unable to properly handle the quality issues of reported data.

Finally, the NYEM platform and the annual EO88 report produced by the BuildSmart NY team identify performance at agency and facility levels, but do not sufficiently highlight trends in the data, nor serve to identify performance by facility characteristics to allow for the development of new energy service offerings to help customer meet their required energy reductions.

Expected Approaches & Required Delivery

In order to enable expanded analysis, the data will be converted to a "tidy" format and stored in a properly normalized SQL database, as the volume of data may make in-memory storage impractical. The database may be operationalized by NYPA; Microsoft SQL Server will be utilized to match NYPA's enterprise architecture. The analyses for this project (conducted in R), may be leveraged into enterprise data integration and visualization softwares to allow automated ingestion into a cloud platform.

The method for handling of missing data has not yet been determined, but it is expected that values will be imputed for based on other reported values for each facility, in combination with data for similar facilities and weather data (possibly that provided through NYEM). A method for flagging possibly-abberant data will also be established, likely based on change from previous months.

Data Processing

The data provided from the above-mentioned software vendor was provided in two files, one containing building data and the other containing reported consumption data. These files are read into R:

```
library(tidyverse)

bldg <- read_csv("data/2017-10-25_nyem-eo88_bldg-all-sfy.csv")
eo88 <- read_csv("data/2017-10-26_nyem-eo88_consumption-all-sfy.csv")
```

Building Data

The structure of the building information dataset is investigated:

```
# function to print summary of variable types & missing values
library(pander)
var_summary <- function(df) {
  cat(paste(deparse(substitute(df)), ":\n",
            nrow(df), "observatons of", ncol(df), "variables\n",
            sum(is.na(df)), "/", nrow(df) * ncol(df), "values missing\n"))
  pander(data.frame(class = sapply(df, typeof),
                    Missing = sapply(df, function(x) sum(is.na(x))),
                    row.names = names(df)),
         split.table = Inf, split.cells = 65)
}
var_summary(bldg)
```

```
bldg :
 442 observatons of 126 variables
 28526 / 55692 values missing
```

	class	Missing
ESP Location ID	integer	0
NYEM System ID	character	12
Status	character	0
Agency Name	character	0
Sub Agency	character	252
Campus	character	370
Building ID	character	56
Building Name	character	0
Building Address	character	2
Building City	character	1
Building Zip Code	character	0
Year Built	integer	256
NYS NOAA Climate Region	character	0
Comments	character	335
SFY2010-11 Gross Floor Area (ft2)	double	3
SFY2011-12 Gross Floor Area (ft2)	double	3
SFY2012-13 Gross Floor Area (ft2)	double	3
SFY2013-14 Gross Floor Area (ft2)	double	3
SFY2014-15 Gross Floor Area (ft2)	double	3
SFY2015-16 Gross Floor Area (ft2)	double	3
SFY2016-17 Gross Floor Area (ft2)	double	3
SFY2010-11 Property Type	character	79

	class	Missing
SFY2011-12 Property Type	character	79
SFY2012-13 Property Type	character	79
SFY2013-14 Property Type	character	79
SFY2014-15 Property Type	character	79
SFY2015-16 Property Type (BEDES)	character	91
SFY2016-17 Property Type (ES)	character	91
SFY2010-11 Percent of Gross Floor Area Heated	double	218
SFY2011-12 Percent of Gross Floor Area Heated	double	218
SFY2012-13 Percent of Gross Floor Area Heated	double	218
SFY2013-14 Percent of Gross Floor Area Heated	double	218
SFY2014-15 Percent of Gross Floor Area Heated	double	218
SFY2015-16 Percent of Gross Floor Area Heated	double	218
SFY2016-17 Percent of Gross Floor Area Heated	double	218
SFY2010-11 Percent of Gross Floor Area Cooled	double	219
SFY2011-12 Percent of Gross Floor Area Cooled	double	219
SFY2012-13 Percent of Gross Floor Area Cooled	double	219
SFY2013-14 Percent of Gross Floor Area Cooled	double	219
SFY2014-15 Percent of Gross Floor Area Cooled	double	219
SFY2015-16 Percent of Gross Floor Area Cooled	double	219
SFY2016-17 Percent of Gross Floor Area Cooled	double	219
SFY2010-11 Peak Total Occupants	double	309
SFY2011-12 Peak Total Occupants	double	309
SFY2012-13 Peak Total Occupants	double	309
SFY2013-14 Peak Total Occupants	double	309
SFY2014-15 Peak Total Occupants	double	309
SFY2015-16 Peak Total Occupants	double	309
SFY2016-17 Peak Total Occupants	double	311
SFY2010-11 Weekly Operating Hours (Hours/Week)	double	115
SFY2011-12 Weekly Operating Hours (Hours/Week)	double	115
SFY2012-13 Weekly Operating Hours (Hours/Week)	double	115
SFY2013-14 Weekly Operating Hours (Hours/Week)	double	115
SFY2014-15 Weekly Operating Hours (Hours/Week)	double	115
SFY2015-16 Weekly Operating Hours (Hours/Week)	double	115
SFY2016-17 Weekly Operating Hours (Hours/Week)	double	115
SFY2010-11 Property Type 1	character	2
SFY2011-12 Property Type 1	character	2
SFY2012-13 Property Type 1	character	2
SFY2013-14 Property Type 1	character	2
SFY2014-15 Property Type 1	character	2
SFY2015-16 Property Type 1	character	2
SFY2016-17 Property Type 1	character	2
SFY2010-11 Property Type 1 Percent of Total Gross Floor Area	double	1
SFY2011-12 Property Type 1 Percent of Total Gross Floor Area	double	1
SFY2012-13 Property Type 1 Percent of Total Gross Floor Area	double	1
SFY2013-14 Property Type 1 Percent of Total Gross Floor Area	double	1
SFY2014-15 Property Type 1 Percent of Total Gross Floor Area	double	1
SFY2015-16 Property Type 1 Percent of Total Gross Floor Area	double	1
SFY2016-17 Property Type 1 Percent of Total Gross Floor Area	double	1
SFY2010-11 Property Type 2	character	332
SFY2011-12 Property Type 2	character	332
SFY2012-13 Property Type 2	character	332
SFY2013-14 Property Type 2	character	332

	class	Missing
SFY2014-15 Property Type 2	character	332
SFY2015-16 Property Type 2	character	332
SFY2016-17 Property Type 2	character	332
SFY2010-11 Property Type 2 Percent of Total Gross Floor Area	double	332
SFY2011-12 Property Type 2 Percent of Total Gross Floor Area	double	332
SFY2012-13 Property Type 2 Percent of Total Gross Floor Area	double	332
SFY2013-14 Property Type 2 Percent of Total Gross Floor Area	double	332
SFY2014-15 Property Type 2 Percent of Total Gross Floor Area	double	332
SFY2015-16 Property Type 2 Percent of Total Gross Floor Area	double	332
SFY2016-17 Property Type 2 Percent of Total Gross Floor Area	double	332
SFY2010-11 Property Type 3	character	351
SFY2011-12 Property Type 3	character	351
SFY2012-13 Property Type 3	character	351
SFY2013-14 Property Type 3	character	351
SFY2014-15 Property Type 3	character	351
SFY2015-16 Property Type 3	character	351
SFY2016-17 Property Type 3	character	351
SFY2010-11 Property Type 3 Percent of Total Gross Floor Area	double	351
SFY2011-12 Property Type 3 Percent of Total Gross Floor Area	double	351
SFY2012-13 Property Type 3 Percent of Total Gross Floor Area	double	351
SFY2013-14 Property Type 3 Percent of Total Gross Floor Area	double	351
SFY2014-15 Property Type 3 Percent of Total Gross Floor Area	double	351
SFY2015-16 Property Type 3 Percent of Total Gross Floor Area	double	351
SFY2016-17 Property Type 3 Percent of Total Gross Floor Area	double	351
SFY2010-11 Property Type 4	character	394
SFY2011-12 Property Type 4	character	394
SFY2012-13 Property Type 4	character	394
SFY2013-14 Property Type 4	character	394
SFY2014-15 Property Type 4	character	394
SFY2015-16 Property Type 4	character	394
SFY2016-17 Property Type 4	character	394
SFY2010-11 Property Type 4 Percent of Total Gross Floor Area	double	394
SFY2011-12 Property Type 4 Percent of Total Gross Floor Area	double	394
SFY2012-13 Property Type 4 Percent of Total Gross Floor Area	double	394
SFY2013-14 Property Type 4 Percent of Total Gross Floor Area	double	394
SFY2014-15 Property Type 4 Percent of Total Gross Floor Area	double	394
SFY2015-16 Property Type 4 Percent of Total Gross Floor Area	double	394
SFY2016-17 Property Type 4 Percent of Total Gross Floor Area	double	394
SFY2010-11 Property Type 5	character	394
SFY2011-12 Property Type 5	character	394
SFY2012-13 Property Type 5	character	394
SFY2013-14 Property Type 5	character	394
SFY2014-15 Property Type 5	character	394
SFY2015-16 Property Type 5	character	394
SFY2016-17 Property Type 5	character	394
SFY2010-11 Property Type 5 Percent of Total Gross Floor Area	double	394
SFY2011-12 Property Type 5 Percent of Total Gross Floor Area	double	394
SFY2012-13 Property Type 5 Percent of Total Gross Floor Area	double	394
SFY2013-14 Property Type 5 Percent of Total Gross Floor Area	double	394
SFY2014-15 Property Type 5 Percent of Total Gross Floor Area	double	394
SFY2015-16 Property Type 5 Percent of Total Gross Floor Area	double	394
SFY2016-17 Property Type 5 Percent of Total Gross Floor Area	double	394

The bldg data frame contains 126 columns, many of which correspond to reporting fields for each fiscal year. There also a fair number of fields corresponding to building information independent of fiscal year. To make more efficient use of storage, the data is converted using Hadley Wickham's tidy data principles:

```
# tidy data that changes by SFY
bldg <- bldg %>%
  gather(Field, Value,
         `SFY2010-11 Gross Floor Area (ft2)`:
           `SFY2016-17 Property Type 5 Percent of Total Gross Floor Area`)

# separate FY from actual measure
library(stringr)
bldg <- bldg %>%
  mutate(SFY = str_sub(Field, end = 10),
         Field = str_sub(Field, start = 12))

# remove duplicate entries per FY & ESP ID (unique identifier)
bldg <- bldg %>% distinct(`ESP Location ID`, SFY, Field, .keep_all = TRUE)

# spread FY-based fields
bldg <- bldg %>%
  spread(Field, Value)

# clean up names for easier manipulation
names(bldg) <- str_replace_all(names(bldg), " ", "")
```

Building metadata that remains constant and building data that may change by fiscal year are separated into two tables to join like operations and further optimize use of storage. The primary key for each building is ESPLocationID; as such, this field is contained in both tables:

```
# create two dfs -- one with fixed data & hierarchy; other with SFY data
bldg_meta <- bldg %>% select(ESPLocationID:NYSNOAAClimateRegion)
bldg_sfy  <- bldg %>% select(ESPLocationID, SFY:`WeeklyOperatingHours(Hours/Week)`)
```

Fixed Building Metadata

The building metadata requires only minor cleanup – removing duplicate entries (arising from the repeating of these fields across fiscal years) and the renaming of fields

```
# remove duplicates
bldg_meta <- distinct(bldg_meta)

# rename & reorder fields
bldg_meta <- bldg_meta %>%
  rename(Name = BuildingName,
         Agency = AgencyName,
         ClimateRegion = NYSNOAAClimateRegion,
         Address = BuildingAddress,
         City = BuildingCity,
         ZipCode = BuildingZipCode,
         Included = Status)

# convert included field to boolean
bldg_meta$Included <- bldg_meta$Included == "included"
```

This table is now far more compact:

```
bldg_meta :
 441 observatons of 13 variables
 947 / 5733 values missing
```

	class	Missing
ESPLocationID	integer	0
NYEMSystemID	character	12
Included	logical	0
Agency	character	0
SubAgency	character	251
Campus	character	369
BuildingID	character	56
Name	character	0
Address	character	2
City	character	1
ZipCode	character	0
YearBuilt	integer	256
ClimateRegion	character	0

Variable Building Data

Building data varying across fiscal years requires additional cleanup, as the names of fields are lengthy and not all variables are stored as the correct data type due to earlier transformations.

```
# remove redundant "SFY" from SFY field
bldg_sfy$SFY <- str_sub(bldg_sfy$SFY, 4)

# shorten long names & remove any parentheses
bldg_sfy <- bldg_sfy %>%
  rename(FloorArea = `GrossFloorArea(ft2)`,
         PeakOccupants = PeakTotalOccupants,
         CooledPct = PercentofGrossFloorAreaCooled,
         HeatedPct = PercentofGrossFloorAreaHeated,
         Type_BEDES = `PropertyType(BEDES)`,
         Type_ES = `PropertyType(ES)`,
         WeeklyHours = `WeeklyOperatingHours(Hours/Week)`)

# shorten property & percentage
names(bldg_sfy) <- str_replace_all(names(bldg_sfy), "Property", "")
names(bldg_sfy) <- str_replace_all(names(bldg_sfy), "PercentofTotalGrossFloorArea", "Pct")

# drop empty "Type" field
bldg_sfy <- bldg_sfy %>% select(-Type)

# convert fy-reported fields from character to numeric
bldg_sfy <- mutate_at(bldg_sfy,
  vars(FloorArea, PeakOccupants, CooledPct, HeatedPct, Type1Pct, Type2Pct,
       Type3Pct, Type4Pct, Type5Pct, WeeklyHours),
  parse_number)
```

This table is far more reasonable for processing, and is properly tided and normalized:

```
bldg_sfy :
 3087 observatons of 19 variables
 32032 / 58653 values missing
```

	class	Missing
ESPLocationID	integer	0
SFY	character	0
FloorArea	double	14
PeakOccupants	double	2158
CooledPct	double	1526
HeatedPct	double	1519
Type_BEDES	character	2736
Type_ES	character	2736
Type1	character	7
Type1Pct	double	0
Type2	character	2317
Type2Pct	double	2317
Type3	character	2450
Type3Pct	double	2450
Type4	character	2751
Type4Pct	double	2751
Type5	character	2751
Type5Pct	double	2751
WeeklyHours	double	798

Creation of Database

With the building data properly handled, the SQL Server database is created and populated with the two tables:

```
library(DBI)
library(odbc)

# connect to local SQL Server instance
con <- dbConnect(odbc(), Driver = "{SQL Server}", Server = Sys.getenv("USERDOMAIN"))

# create EO88 db
dbSendQuery(con, "DROP DATABASE IF EXISTS EO88;")
dbSendQuery(con, "CREATE DATABASE EO88;")
dbSendQuery(con, "USE EO88;")

# write two building tables
dbWriteTable(con, "building_fixed", bldg_meta, row.names = FALSE)
dbWriteTable(con, "building_variable", bldg_sfy, row.names = FALSE)

# disconnect from server for sanitation while working on eo88 data
dbDisconnect(con)
```

Consumption Data

The structure of the EO88 reported consumption data is investigated:

```
var_summary(eo88)
```

```
eo88 :
 176886 observatons of 16 variables
 713488 / 2830176 values missing
```

	class	Missing
ESP Location ID	integer	0
Parent ESP Location ID	integer	0
Agency Name	character	0
Sub Agency	character	61543
Campus	character	81973
BuildingId	character	176886
Building Name	character	73028
Fuel Type (units)	character	0
Billing Period Start	double	0
Billing Period End	double	0
Use	double	5475
Cost	double	5547
Utility Provider	character	0
Account Number	character	0
Demand (kW)	double	132150
Rate/Service Classification	character	176886

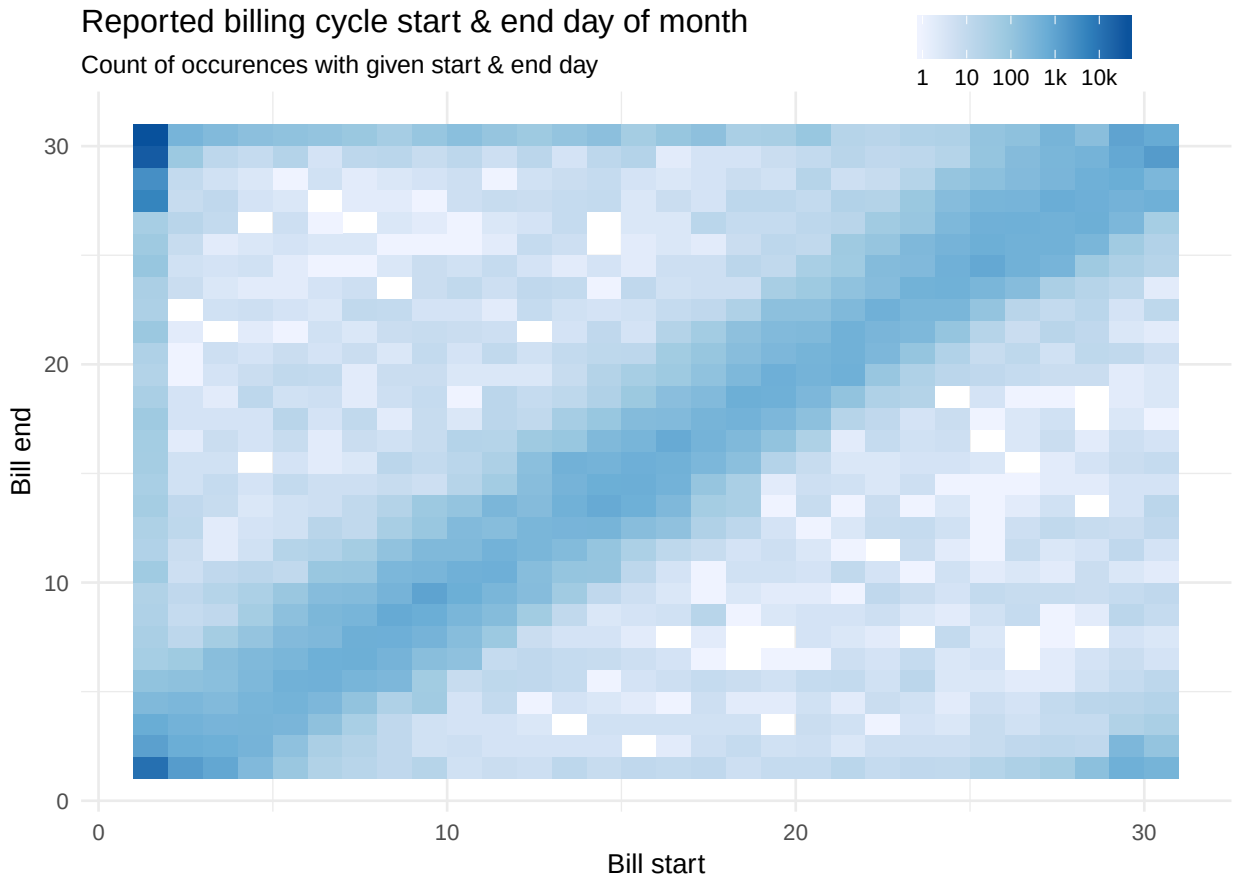
The data frame contains roughly 177,000 rows and 16 columns – the first seven of these contain data also included in the `bldg_meta` dataframe – these are removed to enforce normalization, with the exception of ESP Location ID, which is the unique identifier used to join the two tables. The fuel and units measures are also split into separate columns:

```
# clean up eo88 names
names(eo88) <- str_replace_all(names(eo88), " ", "")
# rename fields
eo88 <- eo88 %>%
  rename(Parent = ParentESPLocationID,
         FuelUnits = `FuelType(units)`,
         Start = BillingPeriodStart,
         End = BillingPeriodEnd,
         Utility = UtilityProvider,
         Demand = `Demand(kW)`)
# split fuel type & units
eo88 <- eo88 %>%
  mutate(Fuel = str_split(FuelUnits, " \\(", simplify = TRUE)[, 1],
         Units = str_split(FuelUnits, " \\(", simplify = TRUE)[, 2],
         Units = str_replace_all(Units, "\\)", "")) %>%
  select(-FuelUnits)
# remove duplicated bldg data (& empty rate/sc) -- retrieve using esp id
eo88 <- eo88 %>% select(ESPLocationID, Parent, Utility, AccountNumber,
                        Start, End, Fuel, Units, Use, Demand, Cost)
# remove missing usage rows
eo88 <- drop_na(eo88, Use)
```

Next Steps

Date Normalization

An investigation of billing start & end dates shows that while an overwhelming number of bills start at the beginning of the month and end at the end of the month, many bills start and/or end at other points in the month, with a strong trend towards roughly matching start and end days:



Before the EO88 reporting data can be added to the database and used for analysis, the billing dates will need to be converted to match calendar months – the month-to-month values may not make a tremendous difference, but values must be able to be matched to state fiscal years, so mapping to calendar months will be essential. This will be attempted using the map family of functions from the purrr package to apply a function that extracts, for each reported value:

- Each calendar month the value encompasses
- The number of days the billed value overlaps each month encompassed
- The prorated share of usage per month, assuming constant usage

Source EUI Calculation

Following this mapping, the energy use intensity can be calculated for each agency for each fiscal year using a simple formula:

$$EUI = \frac{Energy\ Use}{Area}$$

In order to allow for a common view of all fuels, they will all need to be converted to a single unit of measure. The units of measure reported are viewed by fuel and frequency:

```
eo88 %>%
  group_by(Fuel, Units) %>%
  summarize(Count = n()) %>%
  arrange(desc(Count)) %>%
  pander::pander()
```

Fuel	Units	Count
Electricity, grid purchase	kWh	74109
Natural Gas	therms	57318
Natural Gas	ccf	14456
Fuel Oil #2	gallons	11947
Propane and Liquid Propane	gallons	5933
Steam	thousand lbs	2042
Steam, onsite cogeneration	thousand lbs	1225
Electricity, onsite cogeneration	kWh	1105
Fuel Oil #5 and #6	gallons	723
Diesel	gallons	687
Electricity, onsite solar PV	kWh	588
Fuel Oil #4	gallons	263
Electricity, onsite hydro	kWh	231
Coal - bituminous	tons	154
Chilled Water	ton-hours	146
Wood	tons	137
Kerosene	gallons	113
Electricity, other	kWh	85
Fuel Oil #1	gallons	57
Hot Water	therms	30
Electricity, onsite wind	kWh	26
Chilled Water, onsite cogeneration	ton-hours	24
Hot Water, onsite cogeneration	therms	12

The seven types of measurement units present will be converted to thousand British thermal units (kBtu) using Energy Star Portfolio Manager's extensive Thermal Energy Converstions technical reference.

The progress towards EO88 goals is measured using *source EUI*, which takes into account the total energy needed to make the energy used available (i.e. the amount of raw fuel needed to meet the consumption need). Therefore, the above formula will have to be modified to handle the conversion from site energy use to source energy use. This conversion is a fairly straightforward, though the conversion factor varies by fuel; the reference guide created by Portfolio Manager will be used to perform the calculations:

Energy Type	U.S. Ratio	Canadian Ratio
Electricity (Grid Purchase)	3.14	2.05
Electricity (on-Site Solar or Wind Installation)	1.00	1.00
Natural Gas	1.05	1.02
Fuel Oil (1,2,4,5,6,Diesel, Kerosene)	1.01	1.01
Propane & Liquid Propane	1.01	1.03
Steam	1.20	1.20
Hot Water	1.20	1.20
Chilled Water	1.00	0.71
Wood	1.00	1.00
Coal/Coke	1.00	1.00
Other	1.00	1.00

Figure 1: Portfolio Manager Source-Site Ratios

Confidentiality

It is unknown if the data used in this report will need to be treated as confidential (due to the ongoing status of fiscal year 2016-17) – to account for this, data may need to be anonymized to avoid any potential issues. To account for this, a brief anonymization function has been developed, which will scramble the letters of and remove any punctuation from any potentially identifying fields (agency, campus, building, address, etc.):

```
# scramble letters in replicable (but masked) fashion
set.seed(getOption("letter_seed"))
name_sub <- c(sample(LETTERS, 26), " ")

# continue scrambling to increase security
for (i in 1:getOption("number_scrambles")) {
  set.seed(getOption("letter_seed"))
  name_sub <- c(sample(name_sub[1:26], 26), " ")
}

anonymize <- function(old_vector) {
  old_vector <- str_to_upper(old_vector)
  new_vector <- character(0)
  for (i in 1:length(old_vector)) {
    new_str <- NULL
    for (j in 1:str_length(old_vector[i])) {
      old_ltr <- substr(old_vector[i], j, j)
      new_ltr <- name_sub[which(c(LETTERS, " ") == old_ltr)]
      new_str <- paste0(new_str, new_ltr)
    }
    new_vector <- c(new_vector, new_str)
  }
  new_vector
}
```