Technical Discussion Questions - J D Freeman

Please answer each question to the best of your ability and let us know approximately how much time you took. You can share your code as plain text files, a zip or tar.gz archive, or a link to a public GitHub repository.

Please refer to http://github.com/JD-Freeman/Milliman. This is henceforth referred to as "the GitHub repository" throughout this document.

Please note that my time estimates do not include any ancillary activities, such as typing within this document, downloading QlikView, committing my response (and its supporting material) to the GitHub repository.

(1) Describe the largest data engineering project you have worked on. By "data engineering" we mean a software project where you had to parse, transform, compute, model, store, or query a large amount of data. What technologies were involved? What was your role? What was challenging about the project? Include metrics to help us understand the magnitude of the project.

Recent Project:

Working as both the only data scientist and the acting project manager for a software-as-a-service start-up with product features that were heavily reliant upon predictive analytics, and targeting at the K-20 educational market, I was embedded for 7 months within a real live school district as an alpha testing black site.

This district was composed of 4 elementary schools, 1 middle school, 1 senior high school, over 3,500 students, and over 500 faculty and staff. Its departments included District Administration, Curriculum, Business, Transportation, Technology, Food Services, Health Services, Facilities, Special Education, and Campus Police. Teachers were found to have an average of 53 logins to manage for separate systems, none of which spoke to each other, including the Student Information System, Learning Management System, Attendance, and Gradebook.

Our task was to unify all of these into our platform, which would serve as a single source of truth, under a single login. This was built on PostgreSQL and arranged in a snowflake schema that I specified. Further, all student information was regulated under multiple federal laws (COPPA, FERPA & CIPA) and all data assets were required to reside behind a state-regulated, G-rated firewall. Iframes were not permitted to pass ads from 3rd party internet brokers inside the firewall.

As the intent was to run a freemium business model, wherein the core was provided to schools free-of-charge, and revenues were to be generated with premium app sales and targeted advertising, data tasks included conversion and scrubbing, report generation (for both state and federal requirements), and analysis to identify career aptitudes and students at-risk of falters (suicidality, academic weakness, etc.), as well as ad targeting and screening (for appropriateness).

In additional to analytics that were internal to the product, I developed a large interactive financial model and dashboard with over a hundred variables that venture capitalists could alter, each of which populated through the first 5 years' worth of projected user acquisition curves, staffing, cash flow, balance sheets and income statements.

Mid-Career Project:

I was put in charge of a team of 3 part-time SQL programmers as part of a position custom-created to house my Six Sigma greenbelt project. We brought legacy data out of an AS400/MANMAN system and brought it into a custom module within Oracle.

I personally scrubbed the bills-of-material (a document listing which components make up a product, in what quantities, and at what cost) and routers (a document outlining the labor steps to make a product, and at what cost) for errors, duplicates, etc. I then performed correlation analysis to determine what features, as specified by a smart part numbering system, called for which parts, and which parts were assembled in which order.

This allowed us to create a parametric design algorithm wherein the customer would specify the product features they desired via web interface, and our custom module would create a smart part number, bill-of-material, and routing for the custom product. This cut our lead time on custom product from 12 weeks to 4 weeks, essentially eliminating 95% of the time spent waiting on the engineering department (of which I was a member). The remaining 5% of engineering time was left in the process deliberately as a human review prior to production release, which only verified the stability of the solution, and also fulfilled some regulatory compliance requirements.

All told, the project took 5 months, and was deemed critical by our marketing department in retaining market share of custom product within an increasingly competitive environment.

Early-Career Project:

I supervised a team working for 8 months on a data warehousing task that also involved machine imaging.

We took a library of controlled engineering drawings, some of which were subject to military regulations on document control from both US and foreign agencies, scanned them first as raster, scrubbed the raster for noise, vectorized them, then corrected for microscopically disjointed intersections and proper parametric assignment of line lengths.

These newly digitized drawings were then archived as dual raster and vector files, with a smart numbering file nomenclature. The final digital archive came to approximately 1.2TB. The project took about 2/3rds of the time and ½ the staff that would have been required to simply re-draft the entire collection in vector by hand. Further, the scanning document clerks made less money than we would have paid drafters, and this alone essentially offset the cost of the large-format scanner and software.

(2) Write an original function in the python language that finds outliers in an arbitrary list of data. Try to come up with your own definition of outliers without consulting a reference. If you consult a reference, let us know where you looked. Write some additional code to test your function. If you don't know python, then you may use any language that you know, though python is pretty easy to learn so try to do it in python first.

My practical definition of an outlier is largely dependent on context.

In the case of supervised machine learning, for example, an outlier is a datum that tips the bias/variance trade-off toward risk of overfitting.

In the case of a normal probability distribution within a 6σ framework, it is a point beyond ±3σ from the mean.

Regardless, the outlier has the potential to either reveal a new class/segment to us, or to cause our analysis to be less useful/meaningful/accurate if allowed to remain in the same class/segment as the remainder of the data, from which is it removed by a distance too large to be responsibly ignored.

Please note that neither of these definitions carries a presumption of wrongness…outliers should be investigated to determine whether they are the result of error, or whether they indicate something potentially meaningful. In the case of public health, for example, isolating an immune carrier amidst an epidemic could prove very useful in developing a treatment that could save millions of lives, and that individual's immunity, while rare, could be the result of natural genetic variation.

Note: I have used the Anaconda 2.2.0 distribution of Python 2.7 and interpreted it in Project Jupyter notebook, "Milliman Outlier Detection.ipynb" which is in the GitHub repository. There is also a pdf with screen captures of the notebook while running.

References: http://scikit-learn.org/stable/modules/outlier_detection.html

Assumptions:

As if I did NOT generate the data myself:

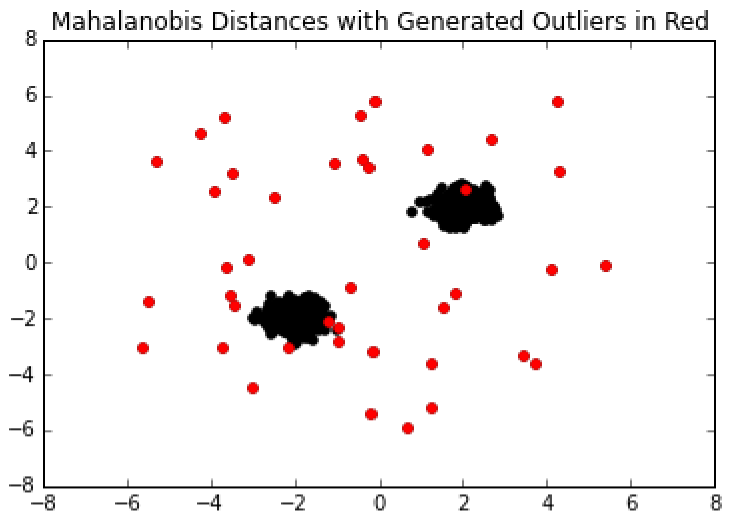
1) I do not know the distribution of the arbitrary data.
2) I do not know the concentration of outliers, if any, within the arbitrary data.

Time: About 1 hour and 15 minutes was spent on the analysis, mostly fiddling around on graphics or making details flow through cobbled-together sections of code to form a united whole.

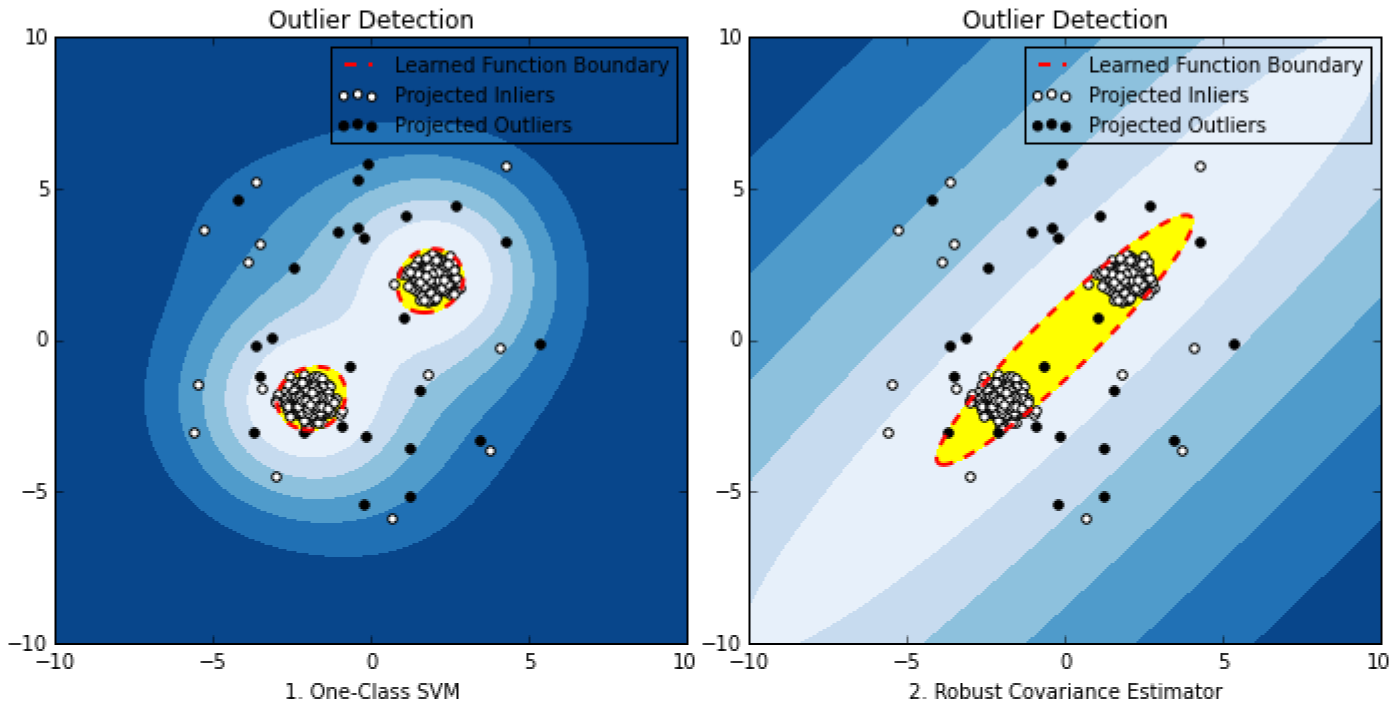
Approach:

1) Generate random data with a random outlier contamination rate and varying cluster separation.

2) Use Mahalanobis distances to estimate outlier contamination while disregarding any fore knowledge from the data generation process (as if I was NOT the data author).



3) Take that estimate and then use it as an underlying assumption for both a single class support vector machine and a robust covariance estimator.



(3) Download any dataset from the UCI Machine Learning Repository. Pose a basic but interesting question about the dataset and write some code that answers the question. Don't overthink this problem, just use this as an opportunity to show that you can write code that works with data.

Time: No additional time was spent on analysis to answer question 3, as it was previously accomplished in an entirely separate context.

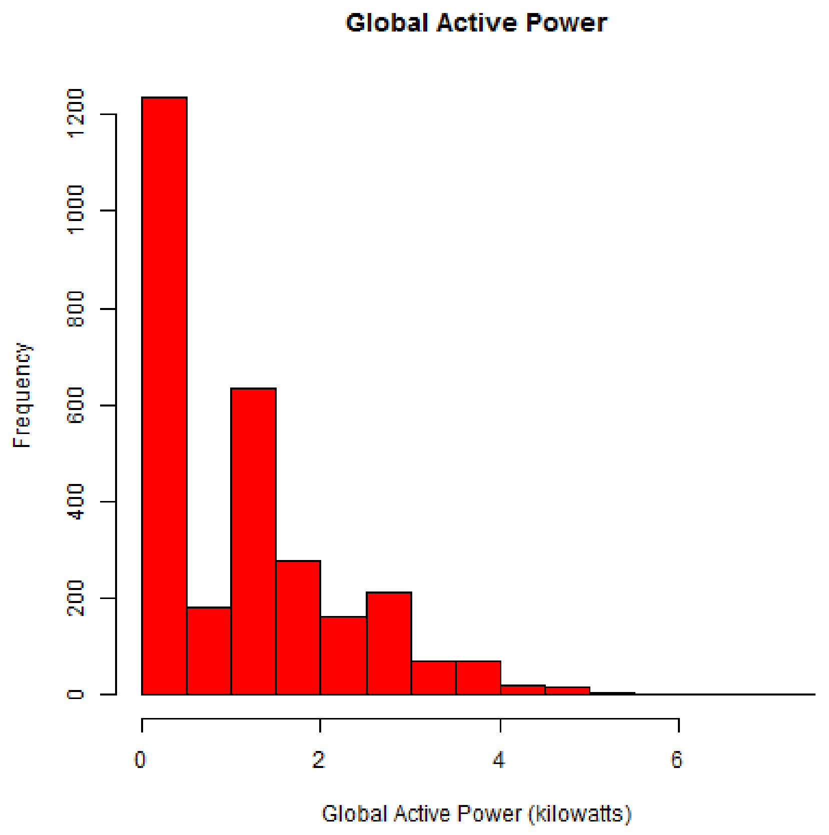
Dataset: "Individual household electric power consumption" from the UC Irvine Machine Learning Repository. This contains measurements of electric power consumption in one household with a one-minute sampling rate over a period of almost 4 years. The dataset has 2,075,259 observations across 9 variables, and the file size exceeds the limit for a public GitHub repository. As you will see in the R scripts, it was saved to the working directory at "household_power_consumption.txt". I am only be using data from the dates 1/2/2007 (Feb 1, 2007 – dd/mm/yyyy) and 2/2/2007 (Feb 2, 2007 – dd/mm/yyyy) in that analysis, and you will see the subset being made in the R scripts. Note that missing values are coded as "?".

The following descriptions of the 9 variables in the dataset are taken from the UCI web site:

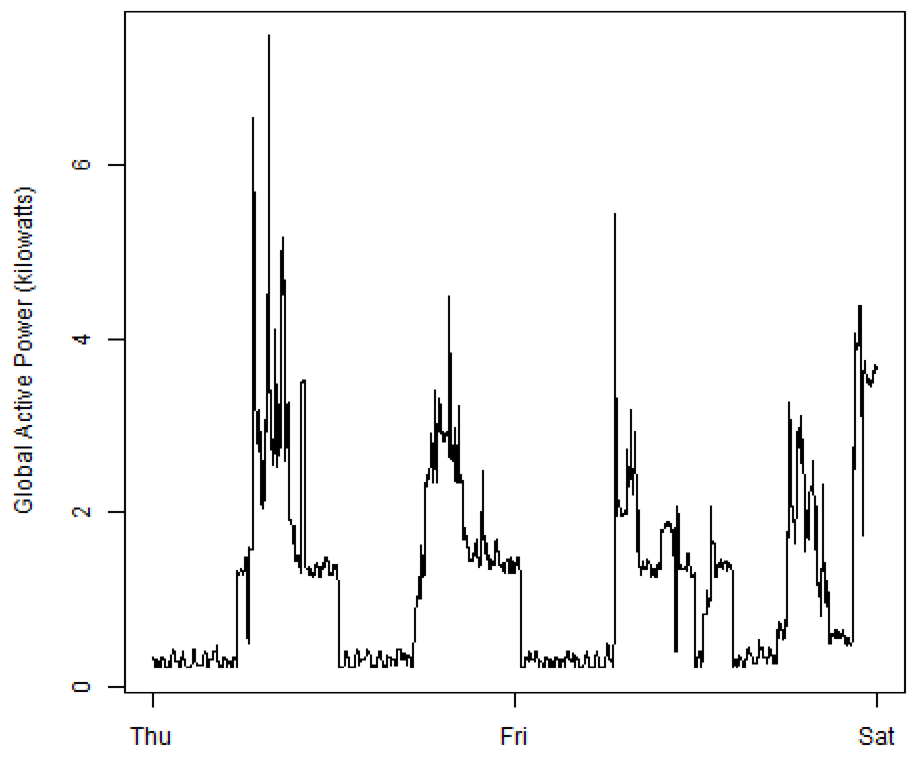
1. **Date**: Date in format dd/mm/yyyy
2. **Time**: time in format hh:mm:ss
3. **Global_active_power**: household global minute-averaged active power (in kilowatt)
4. **Global_reactive_power**: household global minute-averaged reactive power (in kilowatt)
5. **Voltage**: minute-averaged voltage (in volt)
6. **Global_intensity**: household global minute-averaged current intensity (in ampere)
7. **Sub_metering_1**: energy sub-metering No. 1 (in watt-hour of active energy). It corresponds to the kitchen, containing mainly a dishwasher, an oven and a microwave (hot plates are not electric but gas powered).
8. **Sub_metering_2**: energy sub-metering No. 2 (in watt-hour of active energy). It corresponds to the laundry room, containing a washing-machine, a tumble-drier, a refrigerator and a light.
9. **Sub_metering_3**: energy sub-metering No. 3 (in watt-hour of active energy). It corresponds to an electric water-heater and an air-conditioner.

I examine how household energy usage varies over the first 2-day period in February, 2007. Four plots follow, each constructed using the base plotting system in R. PNG files of each image, and the R script files that generated them, are in the GitHub repository.

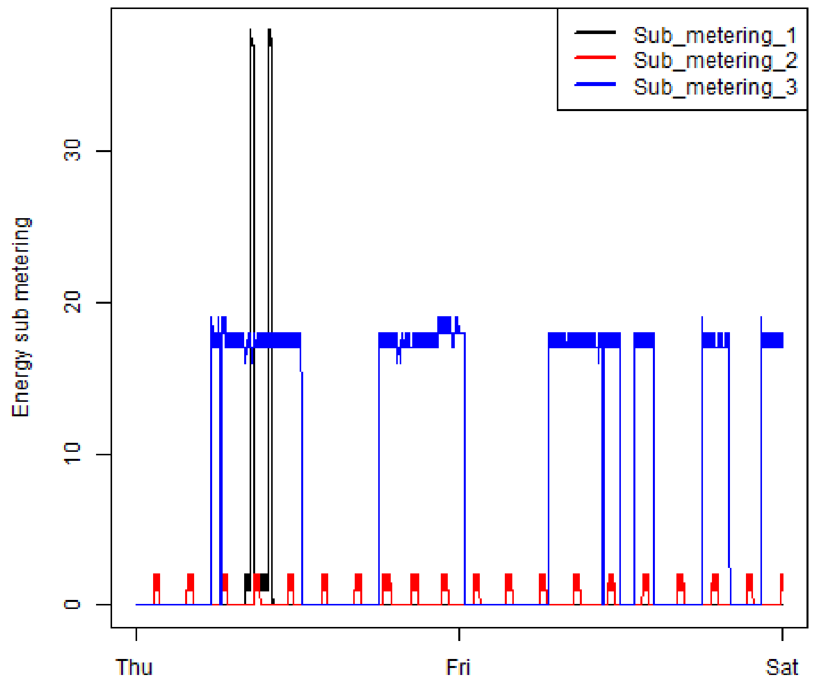
Plot 1:


Global Active Power
Frequency
0
200
400
600
800
1000
1200
0
2
4
6
Global Active Power (kilowatts)

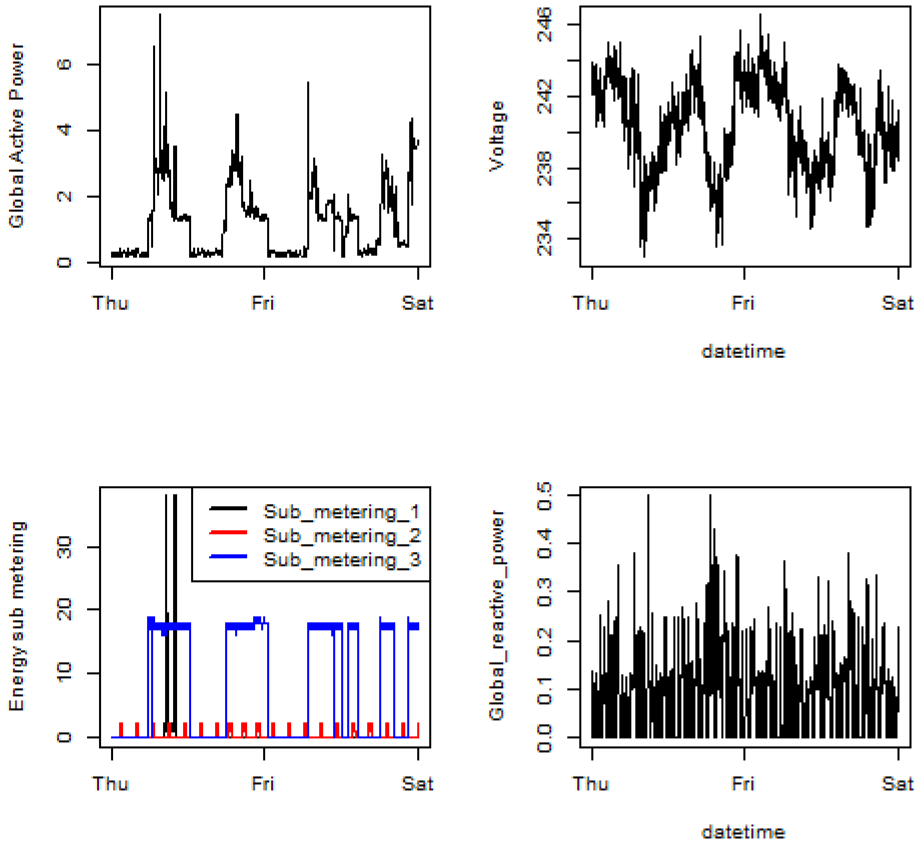
Plot 2:


Global Active Power (kilowatts)
0
2
4
6
Thu
Fri
Sat

Plot 3:


Energy sub metering
0
10
20
30
Thu
Fri
Sat
Sub_metering_1
Sub_metering_2
Sub_metering_3

Plot 4:


Global Active Power
0
2
4
6
Thu
Fri
Sat
Voltage
234
238
242
246
datetime
Energy sub metering
0
10
20
30
Sub_metering_1
Sub_metering_2
Sub_metering_3
Global_reactive_power
0.0
0.1
0.2
0.3
0.4
0.5

(4) Describe some bugs that typically come up when writing code. How do you ensure the quality of your code?

Common Bugs:

- Intolerant/Not Robust in the face of unexpected input (lacks exception handling)
- Hangs/Catches in Infinite Loop
- Fails to compile (if coded in compiled language)
- Fails to perform as expected (more common in open-source languages without commercial support)
- One function fails to pass/receive to/from another
- Conflicts with operating system or environment
- Packages have dissimilar functions called by same name, and one fails to mask
- Function has specific requirement for input of a certain atomic datatype and input must first be coerced (example: factors in R must sometimes be dually coerced {as.numeric(as.character(factor))} before math can be performed with them).
- Metacharacter conflict
- Terminating character not provided at end of line/function call where required
- Improper indenting (Python)
- Scoping (lexical/static vs. dynamic) was ill-considered (searches in wrong order to bind variables)
- Missing string designators (something should be in quotes, but isn't)
- Case sensitivity (something should be capitalized/minuscule, but isn't)

I test my code in stepwise fashion, feeding component functions a variety of inputs, both expected and unexpected, and providing exception handling where necessary. These test inputs often include bounding values, empty sets, wrong data type, etc.

The process of step-wise testing sometimes requires commenting out all sections of the program not under scrutiny, to ensure each piece behaves correctly, in its turn. The programming environment may also offer tracing tools.

(5) What are some of the data elements associated with health care claims and health care clinical data? How are they used in the industry?

Time: Approximately 7 minutes of research went into the response that follows.

Example 1 – Data Elements of Health Care Claims by Outpatient Providers

Form CMS-1500 (revised February 2012), Centers for Medicare and Medicaid Services, for use by outpatient providers such as physicians, radiologists and other non-institutional providers:

- Subscriber's/patient's plan ID number (field 1a);
- Patient's name (field 2);
- Patient's date of birth and gender (field 3);
- Subscriber's name (field 4);
- Patient's address (street or P.O. Box, city, zip) (field 5);
- Patient's relationship to subscriber (field 6);
- Subscriber's address (street or P.O. Box, City, Zip Code) (field 7);
- Whether patient's condition is related to employment, auto accident, or other accident (field 10);
- Subscriber's policy number (field 11);
- Subscriber's birth date and gender (field 11a);
- HMO or preferred provider carrier name (field 11c);
- Disclosure of any other health benefit plans (field 11d);
- Patient's or authorized person's signature or notation that the signature is on file with the physician or provider (field 12)
- Subscriber's or authorized person's signature or notation that the signature is on file with the physician or provider (field 13);
- Date of current illness, injury, or pregnancy (field 14);
- First date of previous, same or similar illness (field 15);
- Name of Referring Provider or Other Source (field 17);
- Referring Provider NPI Number (field 17b);
- Diagnosis codes or nature of illness or injury (current ICD-9 codes are required) (field 21);
- Date(s) of service (field 24A);
- Place of service codes (field 24B);
- EMG (field 24C);
- Procedure/modifier code (current CPT or HCPCS codes are required) (field 24D);
- Diagnosis code (ICD-9) by specific service (field 24E);
- Charge for each listed service (field 24F);
- Number of days or units (field 24G);
- Rendering provider NPI (field 24J);
- Physician's or provider's federal taxpayer ID number (field 25);
- Total charge (field 28);
- Signature of physician or provider that rendered service, including indication of professional license (e.g., MD, LCSW, etc.) or notation that the signature is on file with the HMO or preferred provider carrier (field 31);
- Name and address of facility where services rendered (if other than home or office) (field 32);
- The service facility Type 1 NPI (if different from main or billing NPI) (field 32a);
- Physician's or provider's billing name and address (field 33); and
- Main or billing Type 1 NPI number (field 33a).

Example 2 – Data Elements of Health Care Claims (Intermediaries)

The UB-04 form (previously known as the UB-92 and CMS-1450 claim forms), Centers for Medicare and Medicaid Services, for use by Medicare fiscal intermediaries, Medicaid state agencies and health plans/insurers:

- Provider’s name, address and telephone number (field 1);
- Patient control number (field 3);
- Type of bill code (field 4);
- Provider’s federal tax ID number (field 5);
- Statement period (beginning and ending date of claim period) (field 6);
- Patient’s name (field 8);
- Patient’s address (field 9);
- Patient’s date of birth (field 10);
- Patient’s gender (field 11);
- Date of admission (field 12);
- Admission hour (field 13);
- Type of admission (e.g. emergency, urgent, elective, newborn) (field 14);
- Source of admission code (field 15);
- Patient-status-at-discharge code (field 17);
- Value code and amounts (fields 39-41);
- Revenue code (field 42);
- Revenue/service description (field 43);
- HCPCS/Rates (current CPT or HCPCS codes are required) (field 44);
- Service date (field 45), (required for each date of facility-based non-inpatient services or itemization in a separate attachment is required);
- Units of service (field 46);
- Total charge (field 47);
- HMO or preferred provider carrier name (field 50);
- Type 2 main NPI number (field 56);
- Subscriber’s name (field 58);
- Patient’s relationship to subscriber (field 59);
- Insured’s Unique ID (field 60);
- Principal diagnosis code (current ICD-9 codes are required) (field 67);
- Rendering provider Type 1 NPI (field 76-79); and
- Attending physician ID (field 76-79).

Example 3 – Situationally Applicable Data Elements of Health Care Claims

These additional data elements may also be required for claims on a situational basis:

(1) Other insured's or enrollee's name (CMS-1500, field 9), is applicable if patient is covered by more than one health benefit plan. If the essential data element specified in CMS-1500, field 11d, "disclosure of any other health benefit plans", is answered yes, this is applicable.
(2) Other insured's or enrollee's policy/group number (CMS-1500, field 9a), is applicable if patient is covered by more than one health benefit plan. If the essential data element specified in paragraph CMS-1500, field 11d, "disclosure of any other health benefit plans", is answered yes, this is applicable.
(3) Other insured or enrollee date of birth (CMS-1500, field 9b), is applicable if patient is covered by more than one health benefit plan. If the essential data element specified in paragraph CMS-1500, field 11d, "disclosure of any other health benefit plans", is answered yes, this is applicable.
(4) Other insured or enrollee plan name (employer, school, etc.) (CMS-1500, field 9c), is applicable if patient is covered by more than one health benefit plan. If the essential data element specified in CMS-1500, field 11d, "disclosure of any other health benefit plans", is answered yes, this is applicable.
(5) Other insured or enrollee HMO or insurer name. If the essential data element specified in CMS-1500, field 11d, "disclosure of any other health benefit plans", is answered yes, this is applicable.
(6) Subscriber's plan name (employer, school, etc.) (CMS-1500, field 11b) is applicable if the health benefit plan is a group plan;
(7) Prior authorization number (CMS-1500, field 23), is applicable when prior authorization is required;
(8) Whether assignment was accepted (CMS-1500, field 27), is applicable when assignment has been accepted;
(9) Amount paid (CMS-1500, field 29), is applicable if an amount has been paid to the physician or provider submitting the claim by the patient or subscriber, or on behalf of the patient or subscriber or by a primary plan (Commercial or Medicare). When applicable, a copy of the primary plan's EOB is required;
(10) Balance due (CMS-1500, field 30), is applicable if an amount has been paid to the physician or provider submitting the claim by the patient or subscriber, or on behalf of the patient or subscriber;
(11) Discharge hour (UB-04, field 16), is applicable if the patient was an inpatient, or was admitted for outpatient observation;
(12) Condition codes (UB-04, fields 18-28 are applicable if the CMS UB-04 manual contains a condition code appropriate to the patient's condition;
(13) Occurrence codes and dates (UB-04, fields 31-34), are applicable if the CMS UB-04 manual contains an occurrence code appropriate to the patient's condition;
(14) Occurrence span code, from and through dates (UB-04, field 36), is applicable if the CMS UB-04 manual contains an occurrence span code appropriate to the patient's condition;
(15) HCPCS/Rates (UB-04, field 44), is applicable if Revenue Code description used does not adequately describe service provided or if Medicare is a primary or secondary payer;
(16) Prior payments – payer and patient (UB-04, field 54), is applicable if payments have been made to the physician or provider by the patient or another payer or subscriber, on behalf of the patient or subscriber, or by a primary plan;
(17) Diagnoses codes other than principle diagnosis code (UB-04, fields 67), is applicable if there are diagnoses other than the principle diagnosis and ICD-9 code is required;
(18) Ambulance trip report, is applicable for itemizing a covered ambulance service; and
(19) Anesthesia report is applicable to report time spent on anesthesia services.

Example 4 - Health Care Clinical Data

The complete data element catalog for an electronic health record, according to the National Institutes of Health, has 1921 entries. I find no purpose in pasting all of that into this document. The catalog is available here: http://www.nlm.nih.gov/healthit/dec/dec_complete_052015.xlsx

Uses of Data by the Industry

Insurance companies would use claim data to determine the fair market prices for each service, and limit their payouts accordingly. They would track prevalence of conditions and use this to inform their risk models, and to assign a premium to risk while establishing rates. Temporal data could also be used to track the progress of a claim through the process, improve that process for efficiency, or even to identify where early treatment can be leveraged to reduce costs (and maybe even improve health care outcomes).

Regarding clinical data:

- Doctor's Notes (unstructured text)
    - Specifically because ICD-9 codes can be vague, investigators may well turn to the doctor's notes for additional insight. Here they will run into all of the issues common to unstructured text:
        - misspelling
        - errant capitalization
        - superfluous whitespaces
        - junk Unicode characters
        - spurious punctuation
        - poor grammar
        - synonyms/substitutes (e.g. Lab Name=BRAF Mutations; Substitutes: V600E mutation, V600K mutation, BRAF gene, KRAS gene, DNA, colon cancer, colorectal cancer, cetuximab, panitumumab, melanoma, vemurafenib)
        - hierarchal relationships among terms (e.g. cardiovascular disease -> heart disease -> congestive heart failure -> left sided CHF)
    - There is currently a massive effort underway at the University of Washington health system, expected to take the better part of the next decade, which will include to vectorizing these notes and then clustering them by Hamming distance, in an attempt to make them useful for massive clinical research projects.

- Patient Identifiers
    - Owing to HIPPA and similar privacy safeguards, public health investigators most often wind up working with anonymized data.
    - Otherwise, they have to go through review boards that regulate specifically what they are allowed to have, and must prove to these boards what is related to the research question they are trying to answer, in advance, without first having the benefit access to the data for exploratory analysis.

Finally, use of clinical data could easily vary among stakeholders. For example:

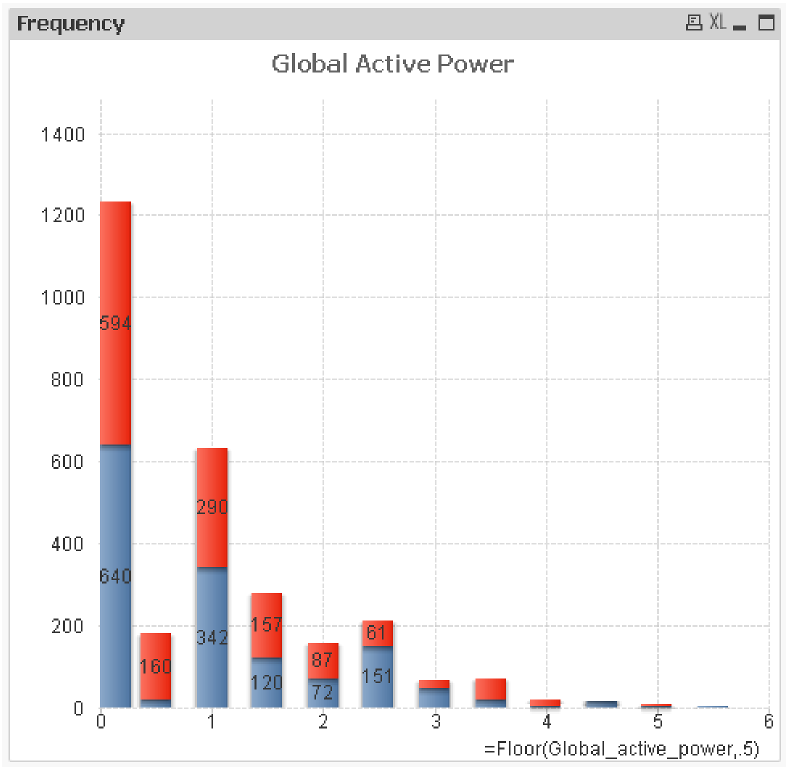
- Care Managers may need to prioritize those most in need.
- Administrators may hunt for treatment pathways with lower costs and better outcomes.
- Clinicians may seek to understand trends in admission and re-admission.
- Researchers may seek out cohorts for experimentation.
- Compliance Officers could monitor whether forms are being completed.

(6) Download the free version of QlikView from here: http://www.qlik.com/us/explore/products/qlikview/free-download. Use QlikView to make a simple analytics visualization to tell a story about your data from question 3 above.

Time: Approximately 12 minutes were spent on this analysis. This occurred only after additional time was used on the download and getting familiar with the interface, as I had no prior experience with QlikView, but rather, had been trained and experienced in Tableau (a somewhat similar software product). Because QlikView does not have a native histogram function like the one called by the Plot1.R script, I spent about half of those 12 minutes finding online references on how to accomplish the binning.

References:
https://community.qlik.com/blogs/qlikviewdesignblog/2014/08/13/recipe-for-a-histogram
https://community.qlik.com/message/337447#337447

Here I reproduce Plot 1 from Question 3, only this time in QlikView:



The QlikView file is also in the GitHub repository. Once again, the dataset itself is too large for the GitHub size limit for a single file in a public repository. “Milliman.qvw” is the project file and “read data.qvw” is the script I created to read the “household_power_consumption.txt” file into QlikView.

Story Narrative:

During the first two days of February 2007, household global active power consumption, averaged over each minute of each day, never exceeded a minute-average of 6 kilowatts. Greater than 70% of the consumption during this same period never exceeded a minute-average of 1.5 kilowatts.