Supplemental Digital Content 2
Retrieval of Denominator Neuromuscular Blocking Agent Data

Version 1.13

1. Synopsis

The following databases were interrogated:

1. The SAFERsleep[*] database at Auckland District Health Board (ADHB);
2. The SAFERsleep database at Waitemata District Health Board (WDHB);
3. The patient administration database "Content Management System (CMS) database" at ADHB;
4. A simple, customised database ('unity'), used to amalgamate and query the data obtained from the two SAFERsleep databases. This database, created using the MySQL relational database management system (Oracle Corporation, Redwood City, CA), is described in detail below.

Methodology is described in Section 2. After reconciliation of minor National Health Index (NHI) numbers, data were sanitised (Section 3) and then queried to determine:

1. Number of patients who received various neuromuscular blocking agents (NMBAs);
2. Number of anesthetics in which various NMBAs were used;
3. Total usage of the various NMBAs;
4. Repeat doses of NMBAs within one anesthetic;
5. Use of NMBA infusions;
6. Scanning versus manual entry of drug identities.

The data thus obtained are summarised and discussed in the accompanying paper, and are presented in detail below in Section 4. Section 5 contains listings of the more complex SQL code.

Potential risks for data analysis are the need to exclude "test" data present within the databases, missing data, inconsistencies within the data, and variable/varied use of codes. It is reassuring that just a small percentage of records suffered from these issues (0.2% of NHIs were defective; 0.86% of entries were excluded; 0.3% of entries refer to a 'generic' NMBA, and 0.5% of entries were altered), as explored in depth in Sections 3 and 4 of this document. The fact that New Zealand National Health Index (NHI) numbers have a built in checksum[†] was invaluable in excluding defective records.

The single most important part of this analysis is the table on page 10, as it lists the number of patients from the two District Health Boards (DHBs) as identified by their NHIs, who were exposed to the various neuromuscular blocking agents used during the interval examined. These data provide the denominator for our calculation of rates of occurrence of anaphylaxis to NMBAs.

Section 6 explains how anyone with access to a MySQL database can use the full, anonymized data set provided with this paper to duplicate almost all of the queries used in this document (Queries labelled `-- [1]-[24]` in the text).

[*] The SAFERsleep anesthetic safety system captures anesthesia-related data and details of intra-operative management of patients.

[†] Described at: http://www.health.govt.nz/system/files/documents/pages/nhi_validation_routine_0.doc Last accessed 21 August 2014.

2. Methodology

2.1 Initial queries:

At ADHB, data for SAFERsleep version 6.0.5 are stored in Microsoft SQL server Enterprise Edition (V9.00.4035.00). This database was interrogated using Microsoft SQL Server 2008 R2 Management Studio (V10.50.1600.1)[‡]. Listing 1 in Section 5 shows the query used.

The principal objective of the code in Listing 1 is to identify all drug records where the class of drug is an NMBA (`Cla.Description LIKE 'Muscle%'`) and the record creation time is between `'2006-01-01 00:00:00' and '2012-12-31 23:59:59'`. The test to identify NMBAs has been checked and found to be necessary and sufficient in this database. Further, minor details of operation of the listed code accompany the listing.

A similar query was used to interrogate the database at WDHB; due to the different SAFERsleep version, two extra fields were retrieved from the `DrAdm` table (`DrugPresentationId` and `IsGeneric`). These two fields are populated with '0' values in the ADHB data (as is clear from Listing 1) and were not used in any analyses. Retrieved data were stored as comma-delimited (CSV) files.

The following files were created and used for data importation:

1) From WDHB[§]:
   a) *WDHB_data.csv* — WDHB data up to June 30, 2012;
   b) *WDHB_late2012.csv* — WDHB data for the remainder of 2012.[¶]
2) From ADHB: *ADHB_data_all.csv* — data for the period January 1, 2006, until December 31, 2012.

In order to amalgamate the data for simple analysis, a flat-file database was created using MySQL, (Final version used for these analyses 5.5.31 running Apache/2/2/22) using the SQL code shown in Listing 2 (Section 5). Data were then imported from the CSV files created above. The MySQL console statements used to perform this task are shown in Listing 3. The insertion statements differ for the two databases, because data retrieved at WDHB were provided to the author with New Zealand date formatting. All MySQL queries were checked on two operating systems (Windows XP SP3 (Microsoft Corporation, Redmond, WA); Ubuntu Linux 12.10, (Canonical, London, United Kingdom)).

There were several initial, minor errors on importation. A backslash included in a surname (line 144 of the main WDHB CSV data, and lines 51 and 52 of the ancillary data) causes a frame misread. These problems were corrected by manual editing of the files. Added blank characters terminating the NHIs in the WDHB data required removal. Two birth dates contained overt typographical errors and were treated as invalid. Similarly, with the ADHB data, backslash characters were altered in the comma-delimited (CSV) data at physical lines 388, 619–625, 693, 1,524–1,527, 2,083, 2,281, 3,047–3,048, 3,248–3,252, 3,272, 3,273, 3,915, 67,075, 147,418–147,419, and 160,142–160,144. Ectopic quote characters were removed at lines 14,642–14,643 and 123,777.

[‡] With Microsoft Data Access Components 3.85.1132.
[§] Thanks to Richard Roger for providing the WDHB data, and running the appropriate query
[¶] Availability of data for the whole of 2012 (not initially present as the project was conceived in mid-2012) motivated completion of the record for this year.

2.2 Reconciliation of Minor NHIs:

On admission to hospital, if a patient is not identified as already having an NHI (for example, if the patient is unable to communicate, incorrect details are provided, or an administrative error has been made) then a duplicate NHI will be created within the national NHI database. Subsequently, most of these NHIs will be reconciled on a national basis with existing (primary) NHIs, and the second NHI will be linked to the primary record and designated a “minor” NHI.

At ADHB the local patient administration database (CMS database) contains current information about major and minor NHIs, and details of the association. The query used to reconcile minor NHIs present in the SAFERsleep database with primary NHIs in CMS is shown in Listing 4. This query identified 788 minor NHIs for the 66,012 valid NHIs at ADHB (1.2%). These associations were imported into a table created using the code in Listing 4, and then reconciled using the remaining code in that listing.

Note that there was no direct reconciliation of minor NHIs at WDHB as the author does not have access to their patient management system (which differs from that at ADHB) or direct access to the national NHI system. A query of all NHIs from the WDHB SAFERsleep database against the CMS database at ADHB however showed that of the 28,412 distinct NHIs, 22,680 matched major NHIs visible at ADHB; there were just 21 minor NHIs among all of those from WDHB.[**] This query may have missed patients who have no representation in the ADHB CMS database, permitting a worst-case number of minor NHIs in the WDHB data of 28,412-22,680= 5,732 entries (20.2% of these NHIs, or ~6.2% of all NHIs) however the ADHB data suggest a rate of approximately 1.2%.

It should be noted that the presence of a minor NHI will only be relevant where the major NHI is also present in the same data set, *i.e.*, there are two entries for the same patient. As noted, examination of the ADHB data set revealed 788 minor NHIs but, if prior to updating the minor NHIs, we say:

```
select count(1) from FLAT inner join minority on FLAT.NHI = minority.major
```

... we obtain just 298 such entries that relate to 97 distinct NHIs. If we confine our attentions to ADHB (*i.e.*, `AND DHB=1`), then the numbers drop to 275 and 88. In other words, without correction for minor NHIs, we determine that for the ADHB data set, the number of NHIs in the denominator was inflated by just 88/(66,260-88), that is 0.13%.

[**] The query was crude, as follows:

```
SELECT count(distinct a1.sAliasIdCode) as MajorCount
FROM prodPatient.dbo.Alias a1
WHERE a1.lAliasTypeId = 1 -- Primary NHI
AND a1.sAliasIdCode IN  ( -- vast list of quoted NHIs here
 );
```

In creating the list of NHIs for insertion in the above, you will also need to replace double quotes with single ones. Under Windows try, *e.g.* `perl -p -i.bak -e "s/\"/'/mg" WDHB_NHIs.csv ;` To create a new csv with carriage returns replaced by commas, say, *e.g.* `perl -p -e "s/\n/,/mg" single.csv > wnhi.csv`

3. Data examination & sanitisation

In addition to the amendments already described at data importation, the following approach was used to 'sanitise' the data:

1. The bad_nhi and suspect_data fields were used to identify defective/suspect data;
2. Records in the database were examined in detail if:
    2.1. The date of birth was defective or missing;
    2.2. Gender was absent;
    2.3. Surnames were missing;
    2.4. Timestamps were defective or appeared inappropriate;
    2.5. Entries were documented as “not administered”.

The results of this investigation are as follows.

3.1 Data characteristics

3.1.1 Number of records

```
SELECT count(1) FROM `FLAT` WHERE 1;  -- [1]
```

There are 237,469 records. The patient count (based on distinct NHIs) is 93,109; the anesthetic count 87,564 for ADHB and 34,440 for WDHB.

```
SELECT count(DISTINCT NHI) FROM `FLAT` WHERE 1;  -- [2]
SELECT
  DHB
  ,count(DISTINCT AnaestheticId)
 FROM `FLAT`
GROUP BY DHB;  -- [3]
```

Table 1. Gender

Sex	DHB	Number
NULL	0	32
F	0	17,159
M	0	11,222
NULL	1	160
F	1	33,180
M	1	32,833

3.1.2 Gender distribution

```
SELECT
  Sex
  ,DHB
  ,count(distinct NHI) as Number
FROM FLAT
GROUP BY
  DHB
  ,Sex
ORDER BY DHB, Sex;
```

The results of this query are shown in table 1. Values where the sex is reported as “NULL” are further examined under below.

3.2 Defective NHIs

There are 449 records (0.2%) with NHIs flagged as defective (the checksum is invalid), reflecting 191 distinct NHIs (0.2%).

```
SELECT *
FROM FLAT
WHERE bad_nhi = 1
ORDER BY NHI;  -- [4]
```

Visual inspection of these records reveals that some are test data from WDHB (notably 70 with the NHI commencing with “ZZZ”), others appear to represent mis-keying of the NHI (Earlier versions of SAFERsleep did not validate the NHI using the checksum). In any case, relative to the total number of patients, anesthetics and drug administrations, this number is small, and these records have subsequently been excluded.

3.3 'Suspect' data

The suspect_data field identifies records where the automated checks described in the documentation for Listing 1 suggest a problem. As the criteria were broad and fairly non-specific, the number of records here is predictably larger at 5,267 (1.9%), with 1,868 distinct NHIs (2.0%), and 737 and 2,014 distinct anesthetics at WDHB and ADHB respectively.

```
SELECT count(1)
FROM FLAT
WHERE suspect_data = 1
ORDER BY NHI;  -- [5]

SELECT count(distinct NHI)
FROM FLAT
WHERE suspect_data = 1;  -- [6]

SELECT
  DHB
  ,count(distinct Anaestheticid)
FROM FLAT
WHERE suspect_data = 1
GROUP BY DHB;  -- [7]
```

The following queries all exclude records where the NHI is incorrect. This Leaves 4,978 records labelled “suspect”.

```
SELECT count(1)
FROM FLAT
WHERE suspect_data = 1
AND bad_nhi <> 1;  -- [8]
```

Of these, 438 have null gender; 1,357 have a null date of birth; 104 have a null surname;[††] the remainder we selected on more 'fuzzy' criteria noted above. Most of these data appear otherwise normal on visual inspection, unless known test NHIs are used (*e.g.*, those with a leading `Z', HUX8660, and PRP1660).

Based on the preceding analysis, it seems reasonable to:

1. Exclude all records with defective NHIs. The number is small, and these records are often defective;
2. Also exclude all records associated with known "test" NHIs. These are listed below;
3. Retain all other records for further analysis.

An initial strategy to identify defective records with apparently valid NHIs is:

```
SELECT DHB, NHI, Firstname, LastName, OperationDescription
FROM FLAT
WHERE bad_nhi <> 1
AND
  (NHI IN ('HUX8660', 'PRP1660', 'SMF4820', 'HTL5755', 'PRG4091',
  'RBX1098', 'SBE2699', 'JBX3656', 'JBX3648', 'QNE2039', 'RMQ4825',
  'RLT4436', 'QES3450', 'PFA2279', 'LYJ6754')
   OR NHI LIKE 'Z%')
ORDER BY NHI;
```

This identifies 115 records; together with the defective NHIs, we exclude a total of 564 rows. We are left with 236,905 rows, retaining 99.76% of all records. However, most of the known test NHIs are not present in the data set, permitting a simpler query to identify defective records:

```
SELECT DHB, NHI, Firstname, LastName, OperationDescription
FROM FLAT
WHERE bad_nhi = 1
  OR NHI IN ('HUX8660', 'PRP1660')
  OR NHI LIKE 'Z%'
ORDER BY NHI;
```

We can conveniently exclude such records by:[‡‡]

```
ALTER TABLE FLAT ADD Excluded INTEGER DEFAULT 0;
UPDATE FLAT SET Excluded = 1 WHERE bad_nhi = 1;
UPDATE FLAT SET Excluded = 1 WHERE NHI IN ('HUX8660', 'PRP1660')
  OR NHI LIKE 'Z%';
```

3.4 Defective timestamps

A reasonable first test is the difference in time between WhenCreated and CreationTime. The former refers to the creation of a drug administration record, and the latter to the creation of the anesthetic record.

```
SELECT *
FROM FLAT
WHERE Excluded = 0
AND TIMESTAMPDIFF(HOUR, CreationTime, WhenCreated) < 0;   -- [9]
```

This reassuringly provides just five additional records. We can add these to our tally of defects:

[††] Subsequent to data entry, it is possible for users to amend or delete the patient surname in the SAFERsleep database.
[‡‡] SQL statements that alter the database structure or contents are shown in red.

```
UPDATE FLAT SET Excluded = 1
WHERE TIMESTAMPDIFF(HOUR, CreationTime, WhenCreated) < 0;
```

We next examine records where the recorded time of drug administration is far in excess of the time of creation of the anesthetic record (over 12 h):

```
SELECT
  NHI
  ,CreationTime
  ,WhenCreated
  ,TIMESTAMPDIFF(HOUR, CreationTime, WhenCreated)  as Delta
  ,NotAdministeredReasonId
  ,AdministrationTime
  ,DHB
  ,OperationDescription
FROM FLAT
WHERE Excluded = 0
HAVING Delta >12
ORDER BY Delta DESC;
```

There are 104 such records. Forty-nine of these have excessive durations (24 h or more) and visual inspection of these and the remainder suggested that all but a few represent erroneous timestamps without other pathology. When we examine the cases where the time difference (`Delta`) is between 9 and 12 h, these are consistently complex and long operations.

In subsequent analyses I have excluded records with such times that are clearly in error.[§§]

```
UPDATE FLAT
  SET Excluded = 1
WHERE TIMESTAMPDIFF(HOUR, CreationTime, WhenCreated) > 12;
```

We now reconcile two other time fields: WhenCreated and AdministrationTime, using a similar query:

```
SELECT
 *
FROM FLAT
WHERE Excluded = 0
AND TIMESTAMPDIFF(HOUR, AdministrationTime, WhenCreated) < 0;
```

There are 43 such records. There is clustering of timestamp differences around -1 h and -23 h, suggesting that these merely represent *post-hoc* data entry, or defects in manual entry of the date. I have not excluded any records based on this minor field.

[§§] It is reasonable to argue that these should be left in, as the administration of NMBAs is documented as having occurred, despite the timestamp errors; however the inaccuracy of the timestamps invites the possibility that the records were generated outside the chosen date range for records. Either way, these records won't substantially influence our denominator data.

3.5 Not administered

It's clearly important to exclude records where the stated drug was not actually administered. These can be determined as follows:

```
SELECT count(1)
FROM FLAT
WHERE NotAdministeredReasonId IS NOT NULL
  AND Excluded = 0;
```

There are 1,375 such records. We will add these records to the Excluded group, but to distinguish these from the preceding records, use the identifier 2 rather than 1:

```
UPDATE FLAT
  SET Excluded = 2
WHERE
  Excluded = 0
  AND NotAdministeredReasonId IS NOT NULL;  -- [10]
```

Any nonzero value in `Excluded` reflects an exclusion. The breakdown of reasons for nonadministration is shown in table 2:

```
SELECT
   NotAdministeredReasonId
   ,count(1) as Number
FROM FLAT
WHERE Excluded = 2
GROUP BY  NotAdministeredReasonId
ORDER BY Number DESC;  -- [11]
```

Table 2. Nonadministration (Reasons)

Reason	**Id**	**Number**
Inadvertent double scan	2	1,069
Decision changed	4	154
Other	6	74
Drug allergy or CI	5	40
Clinical condition changed	3	34
Near miss	1	4

I have not further examined the remaining fields, notably minor details of the process like `IsSoundOff`, `IsGeneric` and `IsDrugExpired` as these are only peripherally relevant to our task. Important fields considered in the following section are `DrugName, Dose, DoseUnit, IsInfusion, InfusionRate, IsBolus` and `WasRepeated`.

3.6 Drug doses

Although not directly relevant to the current analysis, please note that a small number of the recorded drug doses are implausible. These defects appear related to errors on manual entry of data into the SAFERsleep user interface. See Query [19] on pages 12 and 25.

4. Results

In the preceding section, we identified records that could reasonably be excluded, and added the field `Excluded`, which if nonzero indicates an exclusion. Table 3 summarises these exclusions (and the included records) by record count, NHI count, and number of anesthetics.

Table 3. Summary of Data for Excluded Records

Metric	Total	Included	Excluded	% with exclusion
Records	237,469	235,421	2,048	0.86
NHIs	93,109	92,858	1,491	1.58
Anesthetics	112,834	111,395	1,439	1.28

```
SELECT
  count(1) as Records
  ,count(DISTINCT NHI) as NHIs
FROM FLAT
WHERE Excluded = 0;  -- use <> 0 for exclusions  [12],[13]
```

Because `AnaestheticId`s are not unique across DHBs, total anesthetics must be calculated as the sum of anesthetics for the two DHBs.[¶] The number of accepted records, anesthetics and NHIs broken down by DHB (WDHB = 0, ADHB = 1) is shown in table 4.

Table 4. Breakdown of Inclusions by DHB

DHB	Records	Anesthetics	NHIs
0	70,023	34,326	28,320
1	165,398	87,310	66,012

One NHI (The invalid, test NHI 'ZZZ0000') was excluded at both DHBs, and 1,474 NHIs share valid records at both DHBs. For the latter query, an index will speed things:

```
CREATE INDEX flat_nhi_idx ON FLAT(NHI);
SELECT count(DISTINCT A.NHI)
FROM FLAT as A
  INNER JOIN FLAT as B
    ON A.NHI = B.NHI
WHERE A.Excluded = 0
  AND B.Excluded = 0
  AND A.DHB = 1
  AND B.DHB = 0;   -- [14]
```

This query is significant as it demonstrates how patients are generally localised to a particular DHB, at least in terms of our surgical population, with just 2.0–4.5% cross-over.

We can now proceed to our principal queries:
(a) Number of patients who received various neuromuscular blocking agents (NMBAs);
(b) Number of anesthetics in which these NMBAs were used;
(c) Total usage of the various NMBAs;
(d) Repeat doses of NMBAs within one anesthetic;
(e) Use of NMBA infusions.

[¶] SELECT DHB, COUNT(DISTINCT AnaestheticId) FROM FLAT WHERE Excluded = 0 GROUP BY DHB;

Table 5. Patient Exposure (as NHIs) to NMBAs

DrugName	Patients	Percentage of NHIs
Atracurium	67,354	72.5
Mivacurium	1,658	1.8
Muscle Relaxant	332	0.4
Pancuronium	3,799	4.1
Rocuronium	14,995	16.1
Succinylcholine	24,960	26.9
Vecuronium	9,585	10.3

4.1 Patients who received NMBAs

The following query is central to our analysis:

```
SELECT
  DrugName
  ,count(DISTINCT NHI) as Patients
FROM FLAT
WHERE Excluded = 0
GROUP BY
  DrugName
ORDER BY
  DrugName;  -- [15]
```

The above query spans the two hospitals; results are presented in table 5. The denominator for the calculation of percentages is the included NHIs (92,858) from table 3. As some patients were exposed to more than one NMBA (commonly succinylcholine and another agent), the percentage data from table 5 add up to more than 100%. It is clear that the predominant drug used was atracurium, followed by rocuronium, succinylcholine and vecuronium. The use of the unhelpful generic term “Muscle Relaxant” in a small number of cases is explored in Section 4.6.

Although it is possible to analyse the denominator data by number of anesthetics (and these data are indeed presented below) we chose number of distinct patients as we believe this provides a more reasonable denominator. (In any case, use of the former does not substantially alter the major outcomes of the study).

Breakdown by DHB

It is also possible to break down exposure of patients (distinct NHIs) to the NMBAs by DHB.

```
SELECT
  DrugName ,DHB ,count(DISTINCT NHI) as Patients
FROM FLAT
WHERE Excluded = 0
GROUP BY DrugName ,DHB
ORDER BY DrugName ,DHB;  -- [16]
```

The results of this query are presented in table 6. Note however that this query does not accommodate the relatively small number of patients who attended both DHBs, as they are represented in both columns.

Table 6. Patient Exposure to NMBAs, by DHB

DrugName	WDHB	%	ADHB	%
Atracurium	21,943	77.5	46,268	70.1
Mivacurium	588	2.1	1,073	1.6
Muscle Relaxant	29	0.1	303	0.5
Pancuronium	18	0.1	3,781	5.7
Rocuronium	3,740	13.2	11,290	17.1
Succinylcholine	10,024	35.4	15,055	22.8
Vecuronium	1,632	5.8	7,979	12.1

The analysis presented in table 7 is similar to that of table 5, but provides breakdown by *anesthetic* and DHB. The percentages refer to total nonexcluded anesthetics for the two DHB hospitals, i.e. with respective denominators of 34,326 and 87,310 derived from table 4. The predominant drug is atracurium, followed by succinylcholine and rocuronium, with little use of the other agents. Again, as more than one agent may have been used during the same anesthetic, the percentages add up to over 100%.

```
SELECT
  DrugName
  ,DHB
  ,count(DISTINCT AnaestheticId) as Anaesthetics
FROM FLAT
WHERE Excluded = 0
GROUP BY
  DrugName
  ,DHB
ORDER BY
  DrugName
  ,DHB;  -- [17]
```

Table 7. NMBAs and Anesthetic Count

DrugName	WDHB	%	ADHB	%	Total	%
Atracurium	25,227	73.5	58,825	67.4	84,052	69.1
Mivacurium	607	1.8	1,095	1.3	1,702	1.4
Muscle Relaxant	29	0.1	305	0.3	334	0.3
Pancuronium	18	0.1	3,948	4.5	3,966	3.3
Rocuronium	3,915	11.4	12,470	14.3	16,385	13.5
Succinylcholine	11,512	33.5	16,271	18.6	27,783	22.8
Vecuronium	1,689	4.9	8,729	10.0	10,418	8.6

Table 8. Total NMBA Usage (Grams)

DrugName	WDHB	ADHB	Total_grams
Atracurium	1,084	2,469	3,553
Mivacurium	8	13	21
“Muscle Relaxant”	0	(8)	(8)
Pancuronium	0	32	32
Rocuronium	204	811	1,015
Succinylcholine	1,106	1,462	2,569
Vecuronium	15	85	100

4.3 Total NMBA usage

As each administration record should be associated with a dosage, it should also be possible to estimate total consumption of each drug, as the sum of these doses. There are some inconsistencies within the DoseUnit field:

```
SELECT
  DoseUnit
  ,count(1) as Entries
FROM FLAT
WHERE Excluded = 0
GROUP BY DoseUnit;  -- [18]
```

In 235,335 instances the value is “mg”; 8 have “mg/h” or “mg/hr”; 37 have “mcg”, 1 “ugm” (presumably micrograms/min), 3 state “ml” and 37 have null (or place-holder) entries. The few aberrant values reflect the 'configurability' of SAFERsleep. The following query examines the records with “mg” unit values.

```
SELECT
  DHB
  ,DrugName
  ,SUM(Dose)/1000 as Total_grams
FROM FLAT
WHERE Excluded = 0
    AND DoseUnit = 'mg'
    AND Dose < 200
GROUP BY
  DHB
  ,DrugName
ORDER BY
  DHB,
  DrugName;  -- [19]
```

The results of the query are shown in table 8. (Rounding accounts for the inexact sums). The crude limit (Dose < 200) compensates for extreme deviations introduced by manual data entry errors as noted in Section 3.6. More precise refinements introduce little further correction.

Table 9. Repeat Doses of NMBAs

DrugName	WDHB	ADHB
Atracurium	10,365	22,671
Mivacurium	108	117
Muscle Relaxant	2	61
Pancuronium	1	1,376
Rocuronium	1,692	5,865
Succinylcholine	87	123
Vecuronium	870	3,709

4.4 Repeat doses of NMBAs

I here determine the number of anesthetics in which more than one dose of a particular agent was given (Contrasting table 4 and table 5 suggests the magnitude of this repeat usage).

```
SELECT
  AnaestheticId ,DHB ,DrugName
  ,count(1) as Doses
FROM FLAT
WHERE Excluded = 0
GROUP BY
  DHB ,AnaestheticId ,DrugName
HAVING Doses > 1
ORDER BY DHB,DrugName;  -- [20]
```

This provides 47,047 such repeat doses for the various agents. Count the numbers for each agent:

```
SELECT
  REPEATS.DrugName
  ,REPEATS.DHB
  ,count(1)
FROM
(SELECT
  AnaestheticId ,DHB ,DrugName
  ,count(1) as Doses
FROM FLAT
WHERE Excluded = 0
GROUP BY
  DHB ,AnaestheticId ,DrugName
HAVING Doses > 1
ORDER BY
  DHB
  ,DrugName) as REPEATS
GROUP BY
  REPEATS.DrugName
  ,REPEATS.DHB
ORDER BY
  REPEATS.DrugName
  ,REPEATS.DHB;  -- [21]
```

The results of this query are presented in table 9, with the DHBs in columns. Note that these numbers reflect *at least one* repeat dose.

Table 10. NMBA Infusions

DrugName	Anesthetics
Atracurium	527
Mivacurium	22
Pancuronium	10
Rocuronium	24
Succinylcholine	2
Vecuronium	21

4.5 NMBA infusions

The `IsInfusion` field was used to identify use of NMBA infusions. (An idiosyncrasy of SAFERsleep is that this Boolean value is a text field containing either 'FALSE' or 'TRUE').

```
SELECT
  DrugName
  ,count(DISTINCT AnaestheticId) as Anaesthetics
FROM FLAT
WHERE Excluded = 0
  AND IsInfusion = 'TRUE'
GROUP BY
  DrugName
ORDER BY
  DrugName;  -- [22]
```

The results are presented in table 10. It's worth noting that for several of these anesthetics, more than one “IsInfusion” entry is present (easily determined by replacing “DISTINCT AnaestheticId” in the above query with “1”). Of further interest is infusion of Succinylcholine. The `OperationDescription` fields for these data record “Laparoscopy” and “Crash LSCS”; in neither case is an `InfusionRate` recorded.

This invites further examination of the InfusionRate field:

```
SELECT
  DrugName
  ,Count(1) as Instances
FROM FLAT
WHERE Excluded = 0
  AND isInfusion = 'TRUE'
  AND (InfusionRate = 0
    OR InfusionRate IS NULL)
GROUP BY DrugName
ORDER BY DrugName;  -- [23]
```

There are 166 instances of zero/null infusions for atracurium, the corresponding values for mivacurium are 16, rocuronium 10, vecuronium 6, pancuronium 5 and succinylcholine 2. These values suggest inaccurate data capture for these instances.

Table 11. Doses: Scanned Id v Actual Drug

	Scanned identifier						Total
DrugName	**NULL**	**%**	**Matching**	**%**	**Altered**	**%**	
Atracurium	60,705	40.6	88,134	58.9	796	0.5	149,635
Mivacurium	1,267	63.0	710	35.3	34	1.7	2,011
Muscle Relaxant	9	2.1	420	97.9	0	0	429
Pancuronium	2,842	49.1	2,907	50.2	44	0.8	5,793
Rocuronium	15,545	50.5	15,201	49.4	36	0.1	30,782
Succinylcholine	14,639	52.2	13,283	47.4	98	0.3	28,020
Vecuronium	10,517	56.1	8,179	43.6	55	0.3	18,751

4.6 “Muscle Relaxant” and name matching

Data entry of drug administration into the SAFERsleep system can be by barcode scanning or manual entry. The small number of generic “Muscle Relaxant” entries (in 0.3% of anesthetics, as per table 5) suggests that identification of the individual agents worked well.

Anesthetic practices can be further examined by looking at the concordance between the scanned barcode number and the final agent used as represented in the DrugName field. (The anesthetist can manually override the scanned identifier).

```
SELECT
  DrugName
  ,ScannedBarCodeNumber
  ,count(1) as Count
FROM FLAT
WHERE Excluded = 0
GROUP BY
  DrugName
  ,ScannedBarCodeNumber
ORDER BY
  DrugName
  ,ScannedBarCodeNumber;  -- [24]
```

Table 11 reflects the results of this query, appropriately formatted. Scanning appears to have been used in 54.7% of instances, manual entry (absent scan identifier) in 44.8%, and alterations were made to the remaining 0.5% of entries.

5. Listing of code

The following listings have been removed to this section, as they are fairly lengthy and therefore disrupt the flow of the preceding text.

Listing 1. Query used to extract data from SQL Server database at ADHB.

```
SELECT
  Drg.DrugName
  ,Pt.NHI
  ,case
    when EVIL.invalid = 1 then 1
    else 0
   end as bad_nhi
  , case
      when dodgy.IsDodgy = 1
         OR Pt.LastName IS NULL
         OR Pt.DOB < '1900-01-01'
         OR Pt.DOB IS NULL
         OR LEN(Pt.LastName) < 2
         OR Pt.Sex IS NULL
         OR Pt.LastName = 'demo'
         OR Pt.NHI = 'HUX8660' -- identify Mickey Mouse :-(
         OR Pt.LastName LIKE '%xx%'
      then 1
    else
      0
    end as suspect_data
  ,Pt.Sex
  ,Pt.DOB
  , Pt.Firstname
  , Pt.LastName
  ,An.CreationTime
  ,replace(replace(cast(An.OperationDescription as
varchar(256)),char(10),''),char(13),'') as OperationDescription
  ,DrAdm.DrugAdministrationId
  ,DrAdm.AnaestheticId
  ,0 as DrugPresentationId -- was: DrAdm.DrugPresentationId
  ,DrAdm.NotAdministeredReasonId
  ,DrAdm.AdministrationTime
  ,DrAdm.ScannedBarCodeNumber
  ,DrAdm.IsSoundOff
  ,DrAdm.IsDrugExpired
  ,DrAdm.Dose
  ,DrAdm.DoseUnit
  ,0 as IsGeneric -- was: DrAdm.IsGeneric
  ,DrAdm.IsInfusion
  ,DrAdm.InfusionRate
  ,DrAdm.InfusionRateUnit
  ,DrAdm.IsInfusionPurge
  ,DrAdm.IsBolus
  ,DrAdm.Comment
  ,replace(replace(cast(DrAdm.NotAdministeredComment as
varchar(64)),char(10),''),char(13),'')
  ,DrAdm.WasRepeated
  ,DrAdm.WhenCreated
  ,1 as DHB -- signal ADHB
FROM [SaferSleep].[dbo].[Class] as Cla
  -- use Cla.Description to identify neuromuscular blocking agents
```

```
   INNER JOIN [SaferSleep].[dbo].[DrugClass] as DrgCla
     ON Cla.ClassId = DrgCla.ClassId
   INNER JOIN [SaferSleep].[dbo].[DrugProductComponent] as DrPrCo
     ON DrgCla.DrugId = DrPrCo.DrugId
   INNER JOIN [SaferSleep].[dbo].[DrugAdministration] as DrAdm -- the key table
     ON DrPrCo.DrugProductId = DrAdm.DrugProductId
   INNER JOIN [SaferSleep].[dbo].[Anaesthetic] as An
     ON An.AnaestheticId = DrAdm.AnaestheticId
   INNER JOIN [SaferSleep].[dbo].[Drug] as Drg
     ON DrPrCo.DrugId = Drg.DrugId
   INNER JOIN [SaferSleep].[dbo].[AnaestheticPatient] as AnPt
     ON An.AnaestheticId = AnPt.AnaestheticId
   INNER JOIN [SaferSleep].[dbo].[Patient] as Pt
     ON AnPt.PatientId = Pt.PatientId
   LEFT OUTER JOIN
   -- clumsy = validate NHIs in SAFERsleep, make list, join...
   (SELECT
   OUTX.NHI BADNHI
   ,1 invalid
 FROM
 (SELECT
    X.NHI
    ,case
      when X.c1 = 0
        or X.c2 = 0
        or X.c3 = 0
        then -1
      when (7*X.c1 + 6*X.c2 + 5*X.c3 + 4*X.d1 + 3*X.d2 + 2*X.d3)  % 11 = 0
        then 0
      when (11 -  (7*X.c1 + 6*X.c2 + 5*X.c3 + 4*X.d1 + 3*X.d2 + 2*X.d3)%11 ) % 10
= X.chk
        then 1
      else 0
    end as valid
 FROM
   (SELECT
    Pt.NHI as NHI
     , case
         when ascii (substring(Pt.NHI, 1, 1)) - 64 BETWEEN 1 AND 8
           then ascii (substring(Pt.NHI, 1, 1)) - 64
         when ascii (substring(Pt.NHI, 1, 1)) - 64 BETWEEN 10 AND 14
           then ascii (substring(Pt.NHI, 1, 1)) - 65
         when ascii (substring(Pt.NHI, 1,1)) - 64 BETWEEN 16 AND 26
           then ascii (substring(Pt.NHI, 1, 1)) - 66
         else 0
       end as c1
     , case
         when ascii (substring(Pt.NHI, 2, 1)) - 64 BETWEEN 1 AND 8
           then ascii (substring(Pt.NHI, 2, 1)) - 64
         when ascii (substring(Pt.NHI, 2, 1)) - 64 BETWEEN 10 AND 14
           then ascii (substring(Pt.NHI, 2, 1)) - 65
         when ascii (substring(Pt.NHI, 2, 1)) - 64 BETWEEN 16 AND 26
           then ascii (substring(Pt.NHI, 2, 1)) - 66
        else 0
       end as c2
     , case
         when ascii (substring(Pt.NHI, 3, 1)) - 64 BETWEEN 1 AND 8
           then ascii (substring(Pt.NHI, 3, 1)) - 64
         when ascii (substring(Pt.NHI, 3, 1)) - 64 BETWEEN 10 AND 14
           then ascii (substring(Pt.NHI, 3, 1)) - 65
         when ascii (substring(Pt.NHI, 3, 1)) - 64 BETWEEN 16 AND 26
```

```
          then ascii (substring(Pt.NHI, 3, 1)) - 66
        else 0
       end as c3
     ,convert (integer, substring(Pt.NHI, 4, 1)) as d1
     ,convert (integer, substring(Pt.NHI, 5, 1)) as d2
     ,convert (integer, substring(Pt.NHI, 6, 1)) as d3
     ,convert (integer, substring(Pt.NHI, 7, 1)) as chk
  FROM [SAFERsleep].[dbo].[Patient] Pt
  ) as X
) as OUTX
WHERE OUTX.valid <> 1) as EVIL
  ON Pt.NHI = EVIL.BADNHI
  -- end clumsy
  -- third check = possible duplicates based on DOB match
 LEFT OUTER JOIN
 (SELECT distinct
  PtA.PatientId as PtId  -- must be DISTINCT
  , 1 as IsDodgy
FROM
  [SAFERsleep].[dbo].[patient] as PtA
INNER JOIN
  [SAFERsleep].[dbo].[patient] as PtB
    ON PtA.DOB = PtB.DOB   -- match on DOB
    WHERE
      PtA.NHI <> PtB.NHI   -- match both ways
        AND PtA.Sex = PtB.Sex
        AND (  ( UPPER(PtA.FirstName) = UPPER(PtB.FirstName)
               AND  PtA.Sex='F'  -- with women try to identify change in Surname
               )
             OR UPPER(PtA.LastName) = UPPER(PtB.LastName)  -- UPPER is redundant
with SQL server, sigh.
            )
  ) dodgy
    ON dodgy.PtId = Pt.PatientId
  -- end third check
WHERE
  Cla.Description LIKE 'Muscle%'
  AND An.CreationTime BETWEEN '2006-01-01 00:00:00' and '2012-12-31 23:59:59'
ORDER BY
  bad_nhi desc,
  suspect_data desc,
  NHI,
  DrAdm.AdministrationTime;
```

Operational notes.

The query to retrieve data at ADHB was run on 23 September 2013.[***]

In addition to fulfilling the primary objective of retrieving the relevant records, the listed code also does the following:

1. Carriage returns (0x0D) and line feeds (0x0A) within the comment that describes the reason for nonadministration of a drug (DrAdm.NotAdministeredComment) are replaced with blanks;
2. Defective NHIs are flagged (The last digit of the NHI is a check-digit, and the above code implements this check);
3. Suspect records are identified on the basis of:
   - 3.1. Records where the date of birth is the same as the date of birth for another patient in the SAFERsleep database, and the NHI differs, and the gender is the same, and the surname is the same (or the forename is the same and the gender is female);
   - 3.2. Absent surname;
   - 3.3. Unusual date of birth;
   - 3.4. Absent gender;
   - 3.5. Known test records (The database prominently contains data attached to the NHI 'HUX8660' or “Mickey Mouse”);

Condition 3.1 is tested by …

```
FROM
  [SAFERsleep].[dbo].[patient] as PtA
INNER JOIN
  [SAFERsleep].[dbo].[patient] as PtB
    ON PtA.DOB = PtB.DOB   -- match on DOB
    WHERE
      PtA.NHI <> PtB.NHI   -- match both ways
        AND PtA.Sex = PtB.Sex
        AND (   ( UPPER(PtA.FirstName) = UPPER(PtB.FirstName)
               AND  PtA.Sex='F'
               )
             OR UPPER(PtA.LastName) = UPPER(PtB.LastName)
```

… and the latter several questions (3.2 – 3.5) by:

```
when dodgy.IsDodgy = 1
         OR Pt.LastName IS NULL
         OR Pt.DOB < '1900-01-01'
         OR Pt.DOB IS NULL
         OR LEN(Pt.LastName) < 2
         OR Pt.Sex IS NULL
         OR Pt.LastName = 'demo'
         OR Pt.NHI = 'HUX8660'
         OR Pt.LastName LIKE '%xx%'
```

These queries actually turned out not to be very helpful, likely a reflection of the accuracy of the NHIs in identifying patients.

[***] This observation is relevant as it is possible for users to, *e.g.*, alter a surname inappropriately, changing the results of the “IsDodgy” query. I have noted such an occurrence in one case, over the course of a week between queries.

Listing 2. SQL statement used to create MySQL flat-file database for analysis

```
create schema unity CHARACTER SET=utf8 COLLATE=utf8_general_ci;
use unity;

CREATE TABLE FLAT
(  DrugName varchar(64)
  ,NHI varchar(7)
  ,bad_nhi integer
  ,suspect_data integer
  ,Sex varchar(1)
  ,DOB datetime default NULL
  ,Firstname varchar(32)
  ,LastName varchar(64)
  ,CreationTime datetime default NULL
  ,OperationDescription varchar(256)
  ,DrugAdministrationId integer
  ,AnaestheticId integer
  ,DrugPresentationId integer
  ,NotAdministeredReasonId integer
  ,AdministrationTime datetime default NULL
  ,ScannedBarCodeNumber varchar(20)
  ,IsSoundOff varchar(5)
  ,IsDrugExpired varchar(5)
  ,Dose integer
  ,DoseUnit varchar(16)
  ,IsGeneric varchar(5)
  ,IsInfusion varchar(5)
  ,InfusionRate integer
  ,InfusionRateUnit varchar(16)
  ,IsInfusionPurge varchar(5)
  ,IsBolus  varchar(5)
  ,Comment varchar(256)
  ,NotAdministeredComment varchar(256)
  ,WasRepeated  varchar(5)
  ,WhenCreated  datetime default NULL
  ,DHB integer default 0
);
```

As noted in Section 2.1, the `DrugPresentationId` and `IsGeneric` fields are not used.

Listing 3. MySQL command line statements used to populate database.

```
-- from WDHB:
set max_error_count=65535;
load data local infile 'P:/projects/anaphylaxis/WDHB_data.csv'
-- load data local infile '/home/jvs/Projects/unity/WDHB_DATA.csv'
   into table FLAT
 fields terminated by ','
 enclosed by '"'
 lines terminated by '\n'
 ignore 1 lines
 (DrugName ,@NHI ,bad_nhi ,suspect_data ,Sex ,@DOB ,Firstname
  ,LastName ,@CreationTime ,OperationDescription ,DrugAdministrationId
  ,AnaestheticId ,DrugPresentationId ,NotAdministeredReasonId
  ,@AdministrationTime ,ScannedBarCodeNumber ,IsSoundOff
  ,IsDrugExpired ,Dose ,DoseUnit ,IsGeneric ,IsInfusion
  ,InfusionRate ,InfusionRateUnit ,IsInfusionPurge ,IsBolus
  ,Comment ,NotAdministeredComment ,WasRepeated ,@WhenCreated)
SET DOB = STR_TO_DATE(@DOB, '%d/%m/%Y %H:%i'),
 CreationTime = STR_TO_DATE(@CreationTime, '%d/%m/%Y %H:%i'),
 AdministrationTime = STR_TO_DATE(@AdministrationTime, '%d/%m/%Y %H:%i'),
 WhenCreated = STR_TO_DATE(@WhenCreated, '%d/%m/%Y %H:%i'),
 NHI = RTRIM(@NHI);
```

The header line is ignored. The CSV file contained 64,720 data lines, and resulted in this number of row insertions into the database. Warnings are related to NULL timestamps (mostly for the WhenCreated field).
A similar query is used for the ancillary file from WDHB that contains 5,955 data lines, altering the first line of the above query to:

```
load data local infile 'P:/projects/anaphylaxis/WDHB_late2012.csv'
-- load data local infile '/home/jvs/Projects/unity/WDHB_late2012.csv'
```

```
-- from ADHB:
load data local infile 'P:/projects/anaphylaxis/ADHB_data_all.csv'
-- load data local infile '/home/jvs/Projects/unity/ADHB_DATA_all.csv'
   into table FLAT
 fields terminated by ','
 enclosed by '"'
 lines terminated by '\n'
 ignore 1 lines
 (DrugName ,NHI ,bad_nhi ,suspect_data ,Sex ,DOB ,Firstname
  ,LastName ,CreationTime ,OperationDescription ,DrugAdministrationId
  ,AnaestheticId ,DrugPresentationId ,NotAdministeredReasonId
  ,AdministrationTime ,ScannedBarCodeNumber ,IsSoundOff
  ,IsDrugExpired ,Dose ,DoseUnit ,IsGeneric ,IsInfusion
  ,InfusionRate ,InfusionRateUnit ,IsInfusionPurge ,IsBolus
  ,Comment ,NotAdministeredComment ,WasRepeated ,WhenCreated,DHB);
```

I stored the query here with a header, so I ignore the first of the 166,795 lines.[†††] Importation details are discussed in Section 2.1. The total number of lines imported should be 166,794 + 64,720 + 5,955 = 237,469 lines: `select count(1) from FLAT;`

[†††] To store the CSV with quoted strings in MS SQL Server Management Studio, under Query | Query options | Results, check the box that says “Quote strings ...”. The “Include column headers” option is here too.

Listing 4. Reconciliation of minor NHIs at ADHB

```
SELECT
  distinct Pt.NHI    as Minor
   , a2.sAliasIdCode as Major
   -- , a1.lAliasTypeId
FROM [SaferSleep].[dbo].[Class] as Cla  -- use Cla.Description to identify
neuromuscular blocking agents
  INNER JOIN [SaferSleep].[dbo].[DrugClass] as DrgCla
    ON Cla.ClassId = DrgCla.ClassId
  INNER JOIN [SaferSleep].[dbo].[DrugProductComponent] as DrPrCo
    ON DrgCla.DrugId = DrPrCo.DrugId
  INNER JOIN [SaferSleep].[dbo].[DrugAdministration] as DrAdm -- the key table
    ON DrPrCo.DrugProductId = DrAdm.DrugProductId
  INNER JOIN [SaferSleep].[dbo].[Anaesthetic] as An
    ON An.AnaestheticId = DrAdm.AnaestheticId
  INNER JOIN [SaferSleep].[dbo].[Drug] as Drg
    ON DrPrCo.DrugId = Drg.DrugId
  INNER JOIN [SaferSleep].[dbo].[AnaestheticPatient] as AnPt
    ON An.AnaestheticId = AnPt.AnaestheticId
  INNER JOIN [SaferSleep].[dbo].[Patient] as Pt
    ON AnPt.PatientId = Pt.PatientId
 INNER JOIN prodPatient.dbo.Alias a1
   ON a1.sAliasIdCode = Pt.NHI
 INNER JOIN prodPatient.dbo.Alias a2 ON a1.lPatientId = a2.lPatientId
WHERE a1.lAliasTypeId <>1
  AND a2.lAliasTypeId = 1 -- Primary NHI
 AND Cla.Description LIKE 'Muscle%'
 AND An.CreationTime BETWEEN '2006-01-01 00:00:00' and '2012-12-31 23:59:59';
```

This is exported as *ADHB_minor_major.csv* and can then be imported into the unity database as follows:

```
create table minority(
  minor varchar(7)
  ,major varchar(7)
  );
  CREATE INDEX minority_minor_idx ON minority(minor);
  CREATE INDEX minority_major_idx ON minority(major);
```

Then import the exported csv (788 rows):

```
load data local infile 'P:/projects/anaphylaxis/ADHB_minor_major.csv'
-- load data local infile '/home/jvs/Projects/unity/ADHB_minor_major.csv'
   into table minority
 fields terminated by ','
 enclosed by '"'
 lines terminated by '\n'
 ignore 1 lines
 (minor, major);
```

Finally we replace minor NHIs throughout the database with primary NHIs, with 2,276 alterations.

```
update FLAT, minority
set FLAT.NHI = minority.major
where FLAT.NHI = minority.minor;
```

6. Anonymized data

The data set is available as an anonymised CSV table (*anonymous2.csv*). To recreate the anonymized data set on within a recent version of MySQL on your local machine, from the MySQL command line enter the following commands:

```
CREATE DATABASE X;

USE X;

CREATE TABLE ANON
(  DrugName varchar(64)
  ,NHI integer -- anonymise NHI
  ,bad_nhi integer
  ,suspect_data integer
  ,CreationTime integer  -- year only
  ,AnaestheticId integer -- anonymise AnaestheticId
  ,NotAdministeredReasonId integer
  ,Delta integer -- time difference as above
  ,ScannedBarCodeNumber varchar(20)
  ,Dose integer
  ,DoseUnit varchar(16)
  ,IsInfusion varchar(5)
  ,InfusionRate integer
  ,InfusionRateUnit varchar(16)
  ,IsBolus  varchar(5)
  ,WasRepeated  varchar(5)
  ,DHB integer
  ,Excluded integer);

CREATE INDEX anon_nhi_idx ON ANON(NHI);

load data local infile 'C:/anaphylaxis/anonymous2.csv'
   into table ANON
 fields terminated by ','
 enclosed by '"'
 lines terminated by '\n'
 ignore 1 lines

(DrugName ,NHI ,bad_nhi ,suspect_data ,CreationTime ,AnaestheticId  ,NotAdminist
eredReasonId ,Delta ,ScannedBarCodeNumber ,Dose ,DoseUnit  ,IsInfusion  ,Infusio
nRate ,InfusionRateUnit ,IsBolus ,WasRepeated ,DHB ,Excluded);
```

You will need to alter the path name from C:/anaphylaxis/.. to specify where you stored the CSV file *anonymous2.csv*. Although it is anticipated that many current users will be working in a Microsoft Windows environment, the above statements will work under Linux, with appropriate alterations to path names.

The details of how the anonymous data set was derived from the main database are described in Section 6.2.

6.1 Querying the anonymous data

The following queries can now be performed to confirm the corresponding (numbered) queries on the FLAT database in the main body of this document.

```
SELECT count(1) FROM `ANON` WHERE 1;  -- [1]

SELECT count(DISTINCT NHI) FROM `ANON` WHERE 1;  -- [2]

SELECT DHB ,count(DISTINCT AnaestheticId) FROM `ANON` GROUP BY DHB;  -- [3]

SELECT * FROM ANON WHERE bad_nhi = 1 ORDER BY NHI;  -- [4]

SELECT count(1) FROM ANON WHERE suspect_data = 1 ORDER BY NHI;  -- [5]

SELECT count(distinct NHI) FROM ANON WHERE suspect_data = 1;  -- [6]

SELECT DHB ,count(distinct Anaestheticid) FROM ANON
WHERE suspect_data = 1 GROUP BY DHB;  -- [7]

SELECT count(1) FROM ANON WHERE suspect_data = 1 AND bad_nhi <> 1;  -- [8]

SELECT * FROM ANON WHERE DELTA < 0;  -- [9]

SELECT count(1) FROM ANON WHERE NotAdministeredReasonId IS NOT NULL
  AND Excluded = 2;  -- [10]

SELECT  NotAdministeredReasonId ,count(1) as Number FROM ANON
WHERE Excluded = 2 GROUP BY  NotAdministeredReasonId
ORDER BY Number DESC;  -- [11]

SELECT count(1) as Records ,count(DISTINCT NHI) as NHIs
FROM ANON WHERE Excluded = 0;  -- [12]

SELECT count(1) as Records ,count(DISTINCT NHI) as NHIs
FROM ANON WHERE Excluded <> 0;  -- [13]

SELECT count(DISTINCT A.NHI) FROM ANON as A INNER JOIN ANON as B
    ON A.NHI = B.NHI
WHERE A.Excluded = 0 AND B.Excluded = 0 AND A.DHB = 1 AND B.DHB = 0;   -- [14]

SELECT DrugName ,count(DISTINCT NHI) as Patients
FROM ANON WHERE Excluded = 0 GROUP BY DrugName ORDER BY DrugName;  -- [15]

SELECT DrugName ,DHB ,count(DISTINCT NHI) as Patients
FROM ANON WHERE Excluded = 0 GROUP BY DrugName ,DHB
ORDER BY DrugName ,DHB;  -- [16]
```

```sql
SELECT DrugName ,DHB ,count(DISTINCT AnaestheticId) as Anaesthetics
FROM ANON WHERE Excluded = 0 GROUP BY DrugName ,DHB
ORDER BY DrugName ,DHB;  -- [17]

SELECT DoseUnit ,count(1) as Entries FROM ANON
WHERE Excluded = 0 GROUP BY DoseUnit;  -- [18]

SELECT DHB ,DrugName ,SUM(Dose)/1000 as Total_grams
FROM ANON WHERE Excluded = 0 AND DoseUnit = 'mg'
AND Dose < 200 GROUP BY DHB, DrugName ORDER BY DHB, DrugName;  -- [19]

SELECT AnaestheticId ,DHB ,DrugName ,count(1) as Doses
FROM ANON WHERE Excluded = 0 GROUP BY  DHB ,AnaestheticId ,DrugName
HAVING Doses > 1 ORDER BY DHB,DrugName;  -- [20]

SELECT REPEATS.DrugName ,REPEATS.DHB ,count(1)
FROM (SELECT AnaestheticId ,DHB ,DrugName
  ,count(1) as Doses FROM ANON WHERE Excluded = 0
  GROUP BY DHB ,AnaestheticId ,DrugName HAVING Doses > 1) as REPEATS
GROUP BY REPEATS.DrugName ,REPEATS.DHB ORDER BY REPEATS.DrugName
  ,REPEATS.DHB;  -- [21]

SELECT DrugName  ,count(DISTINCT AnaestheticId) as Anaesthetics
FROM ANON WHERE Excluded = 0 AND IsInfusion = 'TRUE'
GROUP BY DrugName ORDER BY DrugName;  -- [22]

SELECT   DrugName ,Count(1) as Instances
FROM ANON WHERE Excluded = 0 AND isInfusion = 'TRUE'
  AND (InfusionRate = 0 OR InfusionRate IS NULL)
GROUP BY DrugName ORDER BY DrugName;  -- [23]

SELECT DrugName ,ScannedBarCodeNumber ,count(1) as Count
FROM ANON WHERE Excluded = 0
GROUP BY DrugName ,ScannedBarCodeNumber
ORDER BY DrugName ,ScannedBarCodeNumber;  -- [24]
```

6.2 Details of anonymization

In the anonymized data, the following fields have been stripped from the primary database:

1. Forename and surname;
2. Gender;
3. Birth date,
4. Time of administration of the dose, and time of creation of drug entry;
5. DrugAdministrationId, and the redundant fields DrugPresentationId and IsGeneric.

The time of creation of the record has been reduced to a year only. In addition the NHI and Anesthetic ID have each been randomly assigned a unique number (a number that is in no way related to the original identifier). The following method of randomization was used:

1. Rows in the FLAT table were each assigned a unique identifier:
```
ALTER TABLE FLAT ADD ID INT auto_increment primary key;
```
2. Two extra rows were added to FLAT to accommodate random values:
```
ALTER TABLE FLAT ADD rndA int;
ALTER TABLE FLAT ADD rndB int;
```
3. A true random number generator[‡‡‡] was used to generate over 2*237,469 random values.
4. These random values were imported into a separate table:
```
CREATE TABLE RNDM (ID int auto_increment primary key, rndm int);
 load data local infile 'C:/anaphylaxis/uniform_int.csv'
    into table RNDM lines terminated by '\n' ignore 10 lines (rndm);
```
5. Random values were copied into rndA and rndB:
```
update FLAT, RNDM set FLAT.rndA = RNDM.rndm where FLAT.id = RNDM.id;
update FLAT, RNDM set FLAT.rndB = RNDM.rndm where RNDM.id =
237469+FLAT.id;
```
6. NHIs were then randomly mapped to integers from 1–93109 as follows:
```
CREATE TABLE NHI (NHI varchar(7), ID int auto_increment,
 constraint primary key(ID));
 CREATE INDEX nhi_nhi_idx ON NHI(NHI);
INSERT INTO NHI(NHI)
   SELECT NHI from FLAT GROUP BY NHI order by min(rndA);
```
7. AnaestheticID was similarly mapped to integers from 1–122004, taking DHB into account, by:
```
CREATE TABLE ANID (AnaestheticId int, DHB int, ID int auto_increment,
 constraint primary key(ID));
 CREATE INDEX anid_id_idx ON ANID(AnaestheticId);
 INSERT INTO ANID(AnaestheticId, DHB)
   SELECT AnaestheticId, DHB FROM FLAT
 GROUP BY AnaestheticId, DHB order by min(rndB);
```

I next created the ANON table as described above, and then said:

[‡‡‡] Not a pseudorandom number generator, but one based on random noise acquired from a reverse-biased zener diode, suitably processed to convert a normal distribution into a uniform one. Both the generator and the specific random sequence used met criteria for randomness on testing with the Dieharder suite, version 3.31.1, available from http://www.phy.duke.edu/~rgb/General/dieharder.php (tested under Ubuntu Linux version 12.10, Last accessed August 21, 2014).

```
INSERT INTO ANON
 (DrugName, NHI, bad_nhi, suspect_data,
  CreationTime, AnaestheticId, NotAdministeredReasonId,
  Delta,
  ScannedBarCodeNumber, Dose, DoseUnit, IsInfusion,
  InfusionRate, InfusionRateUnit, IsBolus, WasRepeated, DHB, Excluded)
SELECT DrugName, NHI.ID, bad_nhi, suspect_data,
  YEAR(CreationTime), ANID.ID, NotAdministeredReasonId,
  TIMESTAMPDIFF(HOUR, CreationTime, WhenCreated),
  ScannedBarCodeNumber, Dose, DoseUnit, IsInfusion,
  InfusionRate, InfusionRateUnit, IsBolus, WasRepeated, FLAT.DHB, Excluded
FROM FLAT
  INNER JOIN NHI on FLAT.NHI = NHI.NHI
  INNER JOIN ANID
    on FLAT.AnaestheticId = ANID.AnaestheticId AND FLAT.DHB=ANID.DHB;

         SELECT DrugName ,NHI ,bad_nhi ,suspect_data
         ,CreationTime ,AnaestheticId ,NotAdministeredReasonId,
         Delta ,ScannedBarCodeNumber ,Dose ,DoseUnit ,IsInfusion
         ,InfusionRate ,InfusionRateUnit ,IsBolus ,WasRepeated ,DHB ,Excluded
         FROM ANON
         INTO OUTFILE 'C:/anaphylaxis/anonymous2.csv'
         FIELDS TERMINATED BY ','
         ENCLOSED BY '"'
         LINES TERMINATED BY '\n';
```

Potential for de-anonymization

This appears slight, for the following reasons:

1. Important patient identifiers and identifying properties have been stripped off (Name, gender, date of birth, timestamps for interventions) or replaced by random values (NHI, internal anesthetic record identifier).
2. The date of record creation has been reduced to a year;
3. There are sufficiently large numbers for identification of the DHB to be irrelevant to patient identification;
4. The ‘Exclusions’ field simply refers to records excluded as described previously;
5. The remaining data fields (ScannedBarcodeNumber, Dose, DoseUnits, IsInfusion, InfusionRate, InfusionRateUnit, IsBolus, WasRepeated, NotAdministeredReasonId, Delta) refer to the associated drug usage, and there are multiple instances of usage of the drugs for each year. Even with ‘unusual modalities’ such as infusions of various agents, the number of instances in a year appears to limit association of a given instance with a particular individual, *e.g.*

```
select creationtime, count(1) from ANON where isinfusion = 'TRUE'
group by creationtime;
```

The sole residual concern that I can identify is that it is conceivable that an individual who knows the number of anesthetics they underwent in the period might identify their anesthetics, provided the number is large (over 23) and unique.[§§§]

[§§§] select NHI, count(distinct AnaestheticId) as Anesthetics from ANON group by NHI order by Anesthetics desc.