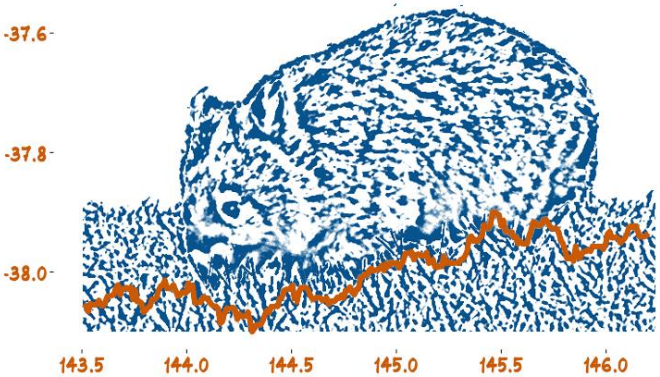


Real Data – Real Issues

what they never told me at Uni

Phil Brierley

WOMBAT

18th February 2016

Data Prep v Predictive Modelling

Time wise…

80%	- Data Prep
5%	- Model Building
50%	- Explaining results

Data Scientists like to over deliver!

- Model building is now becoming a point and click commodity
- ‘Correct’ data preparation will never be this
- Rubbish in – Rubbish out
- Caveat Emptor

  let the buyer beware

- Knowing your data is the most important thing

- Don’t listen to ‘expert opinion’

- The data contains all the questions

  the ‘experts’ may have the answers

This talk is nothing to do with Maths, Statistics or Algorithms

Its to do with the 90% of your time you will spend getting your data 'Algorithm Ready'

4 Steps

1. Getting the Data ✓
   - Receiving the data
2. Sanity Checking the Data ✓
   - Is it consistent?
3. Preprocessing the Data ✓
   - Don’t introduce issues
4. Predictive Modelling ✗
   - Last 5 mins of project timeline & budget!

Getting the data ‘Algorithm Ready’

But we’ll talk a little on this

1. Getting the Data

A long process
- but can be made less painful

Getting the Data

- Can involve 2 or 3 iterations if its not extracted correctly

- This can be avoided by specifying exactly how you want it

Getting the Data

- Raw data only
  - We’ll do any aggregation
  - Quicker for us and you
  - We need to know what has gone on

- All the data (size permitting)
  - We’ll decide what populations not to use
  - Maybe only filter on time
  - Much quicker, storage is cheap
  - Not saving us or you any time by doing filtering your end

Getting the Data

- Database dump if possible
  - Detach database, we’ll reattach
  - Ensures our ‘solution’ can be run in your production environment (*thinking ahead!*)

- Delimited Text Files
  - Pipe delimited (|)
  - No quotes around text fields

Getting the Data

- EXCEL
  - Excel generally means humans involved – BAD!
  - Hard to replicate exactly what has gone on (see above)
  - Excel does weird stuff (see later)
  - Source data won’t be Excel (hopefully)
  - Putting it in Excel to ‘help us’ is not actually helping (the first thing we do is ***try*** get it out of Excel)

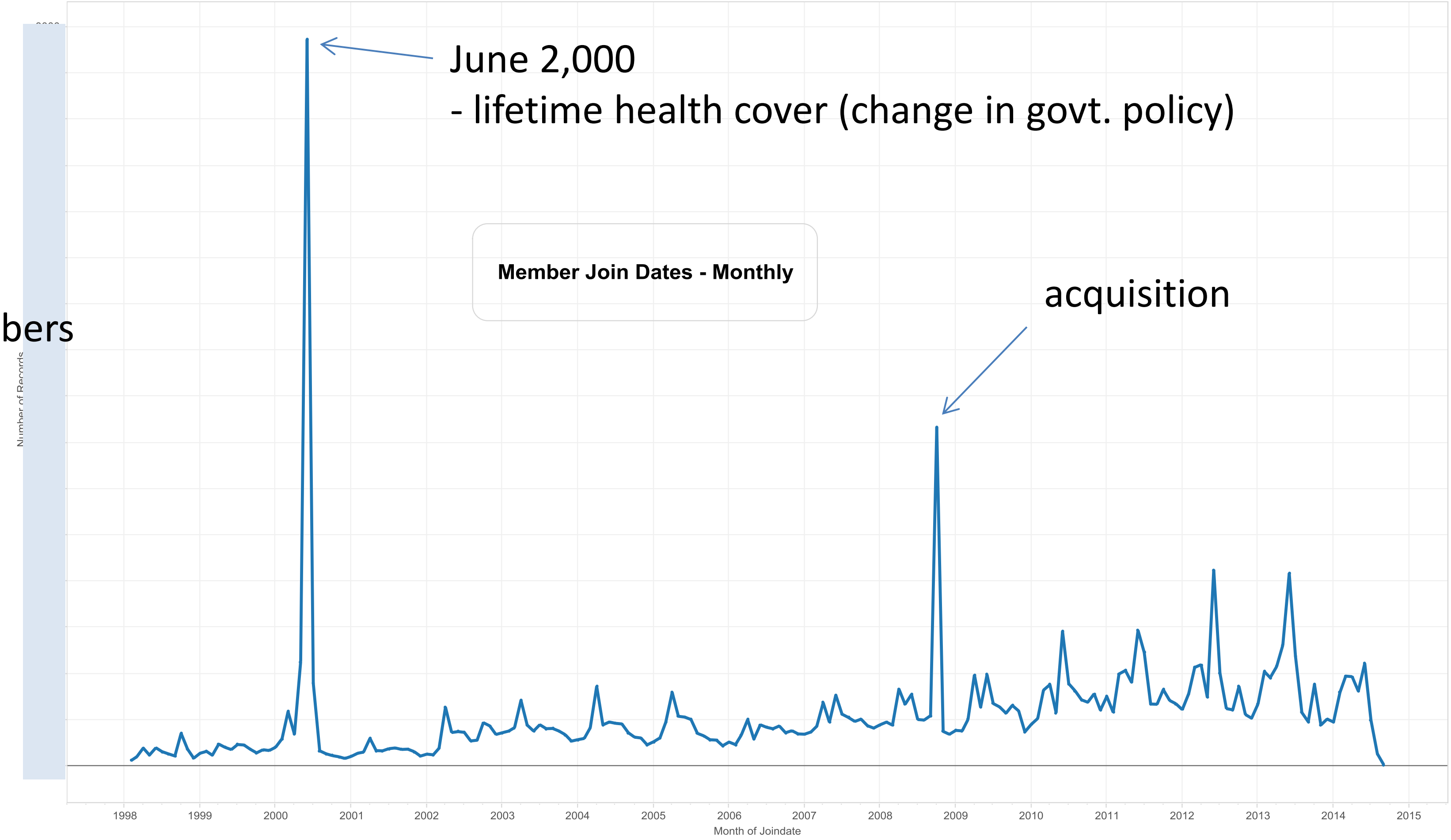
2. Sanity Checking

Identifying Systematic Data Issues

Data driven predictive modelling assumes the future will be like the past – we need to make sure the past is like the past

1. Acquisitions and Events

Health Insurer



June 2,000
- lifetime health cover (change in govt. policy)

acquisition

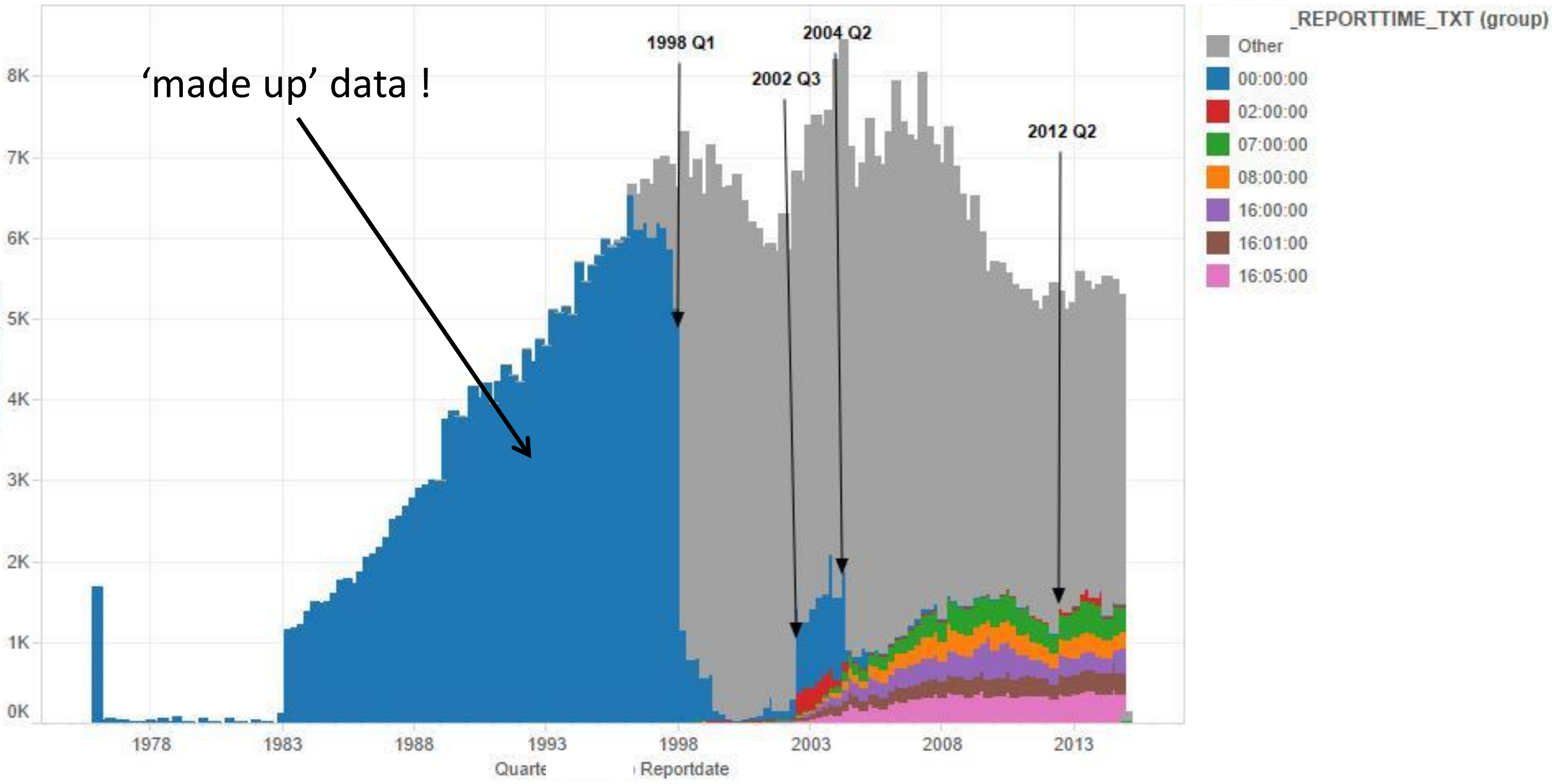
New Members

Time (months) >>

1. Acquisitions and Events

Company Financial Statement Dates

Number of Reports

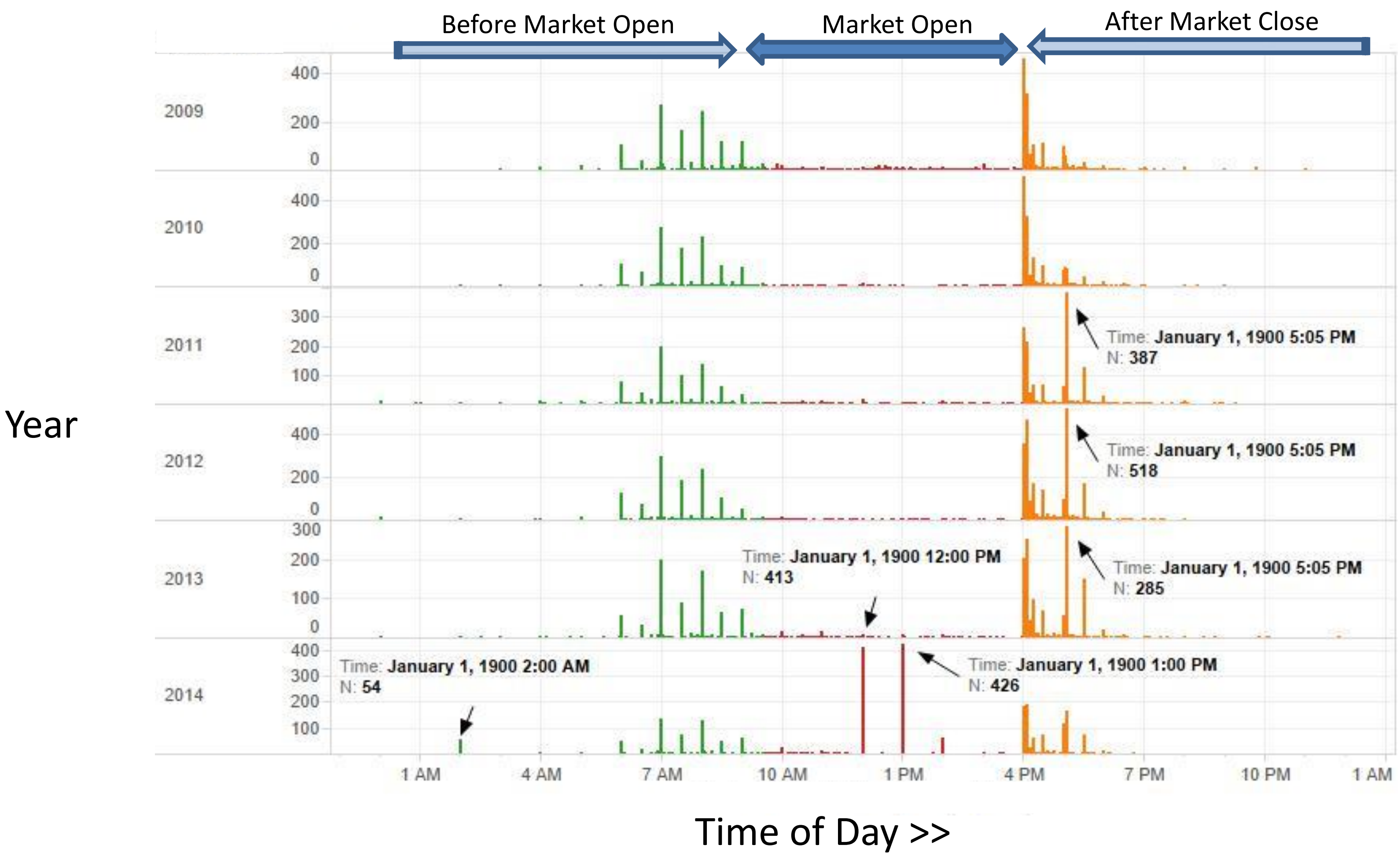


‘made up’ data !

Date >>

2. Made up Dates

Company Financial Statement Times



Before Market Open

Market Open

After Market Close

Time: **January 1, 1900 5:05 PM**
N: **387**

Time: **January 1, 1900 5:05 PM**
N: **518**

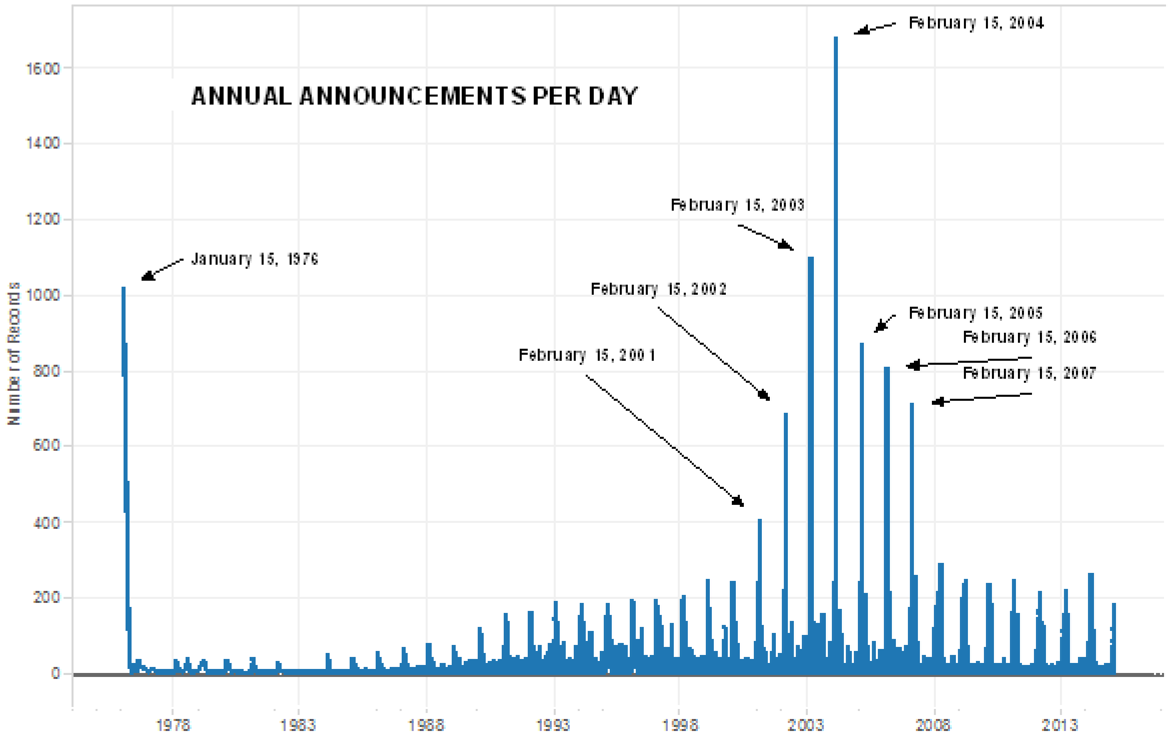
Time: **January 1, 1900 12:00 PM**
N: **413**

Time: **January 1, 1900 5:05 PM**
N: **285**

Time: **January 1, 1900 2:00 AM**
N: **54**

Time: **January 1, 1900 1:00 PM**
N: **426**

2. Made up Dates



Date of Announcement >>

2. Made up Dates

Sometimes fudged !

Electricity Consumption

2am clock change

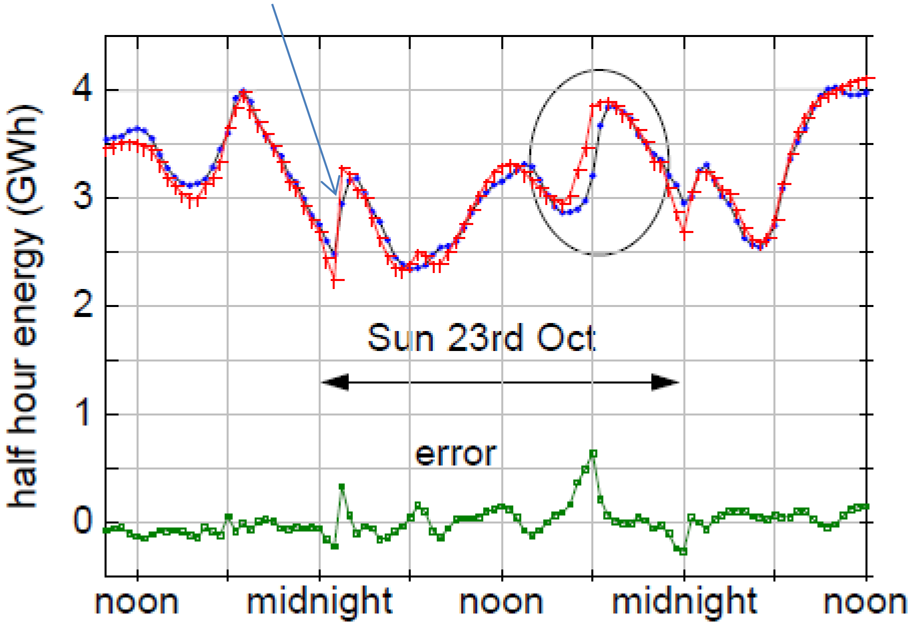


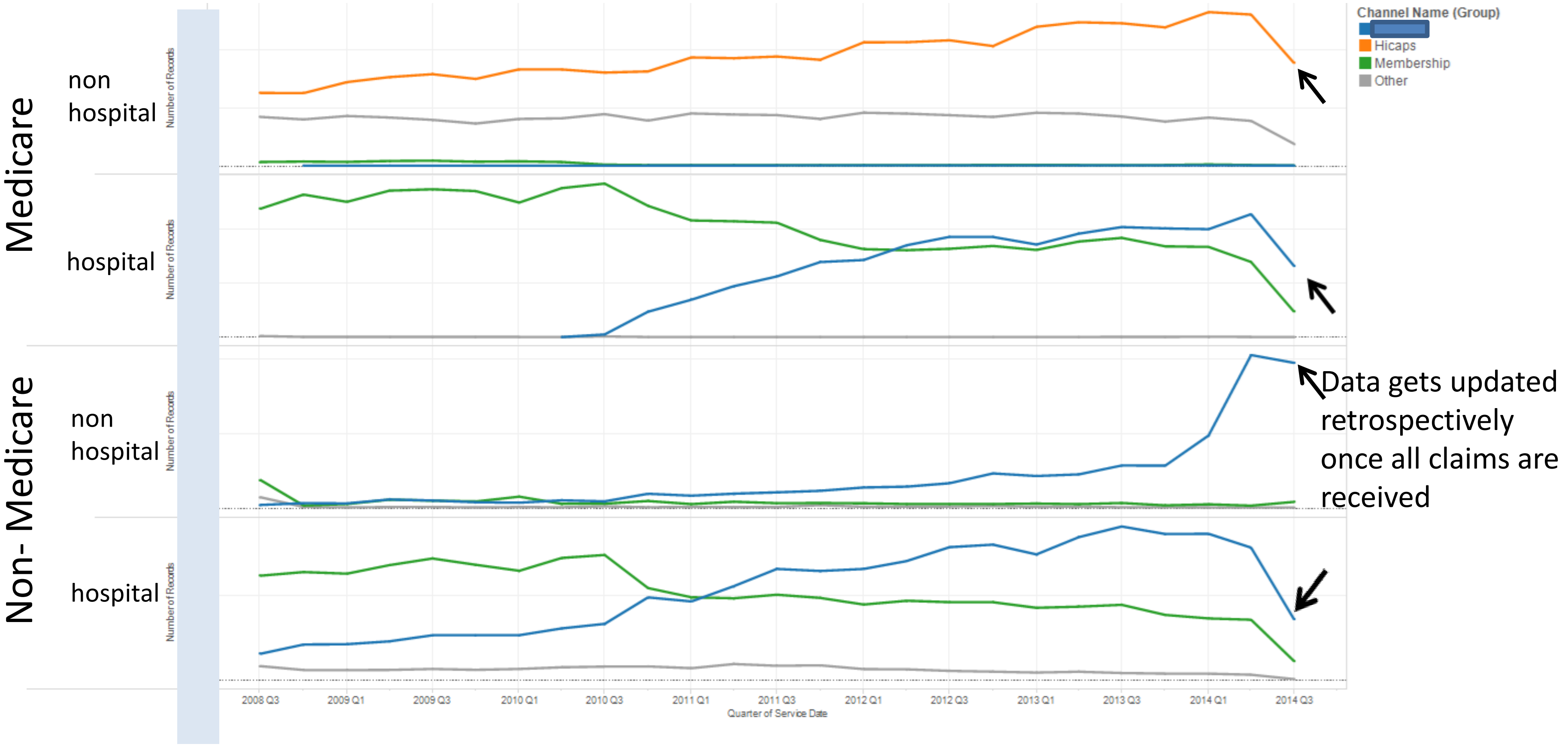
Fig 3-42 *An anomaly the day the clocks change*

2. Made up Dates

- Always look at date distributions
- Dates usually cannot be ‘null’ in a database
- Thus common to see system default dates
- Happens when data sources are ‘merged’ or dates are unknown

3. Moving Definitions

Health Insurance Claims – time lags

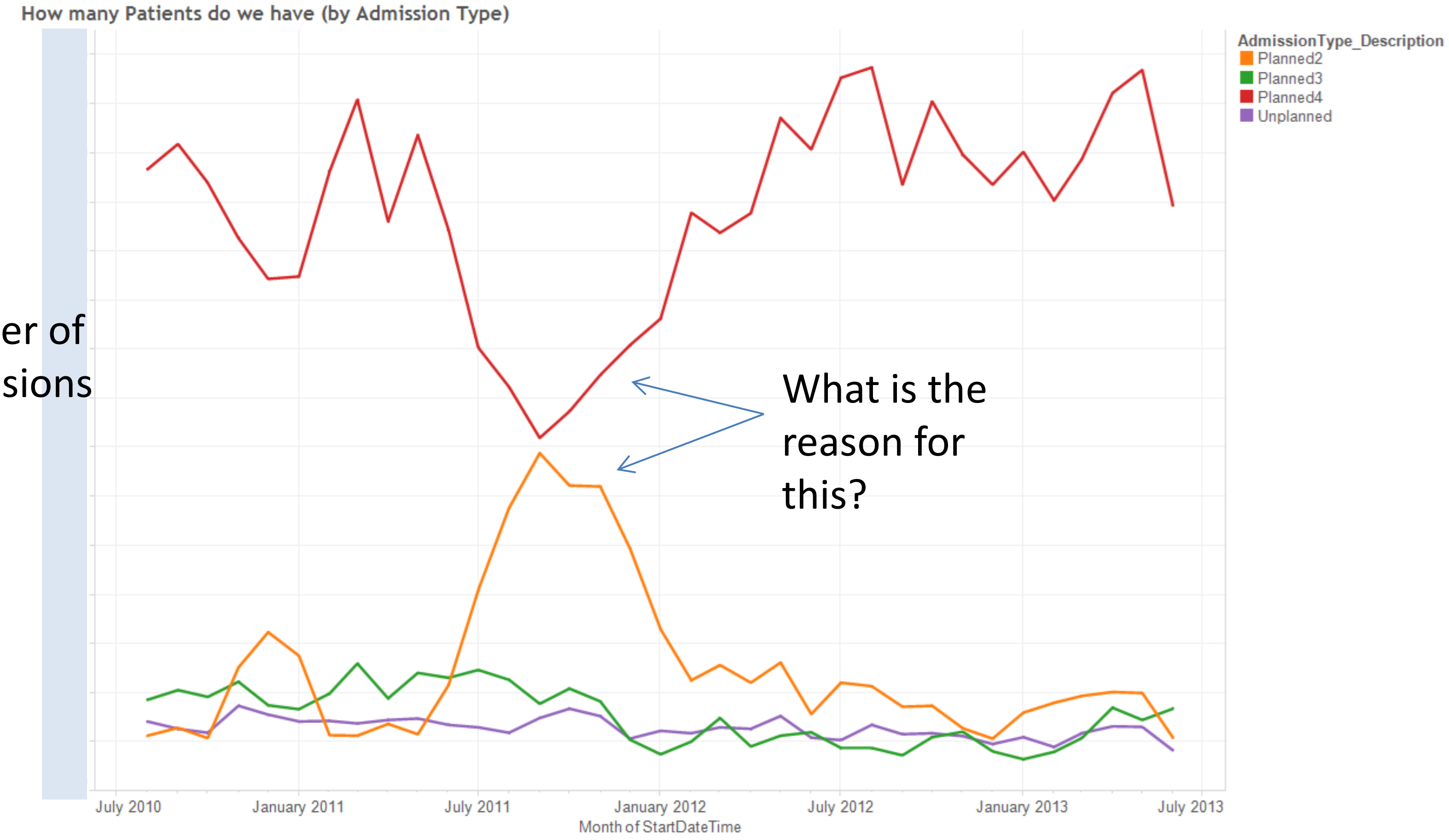


Data gets updated retrospectively once all claims are received

Date >>

3. Moving Definitions

Hospital Admissions



Number of Admissions

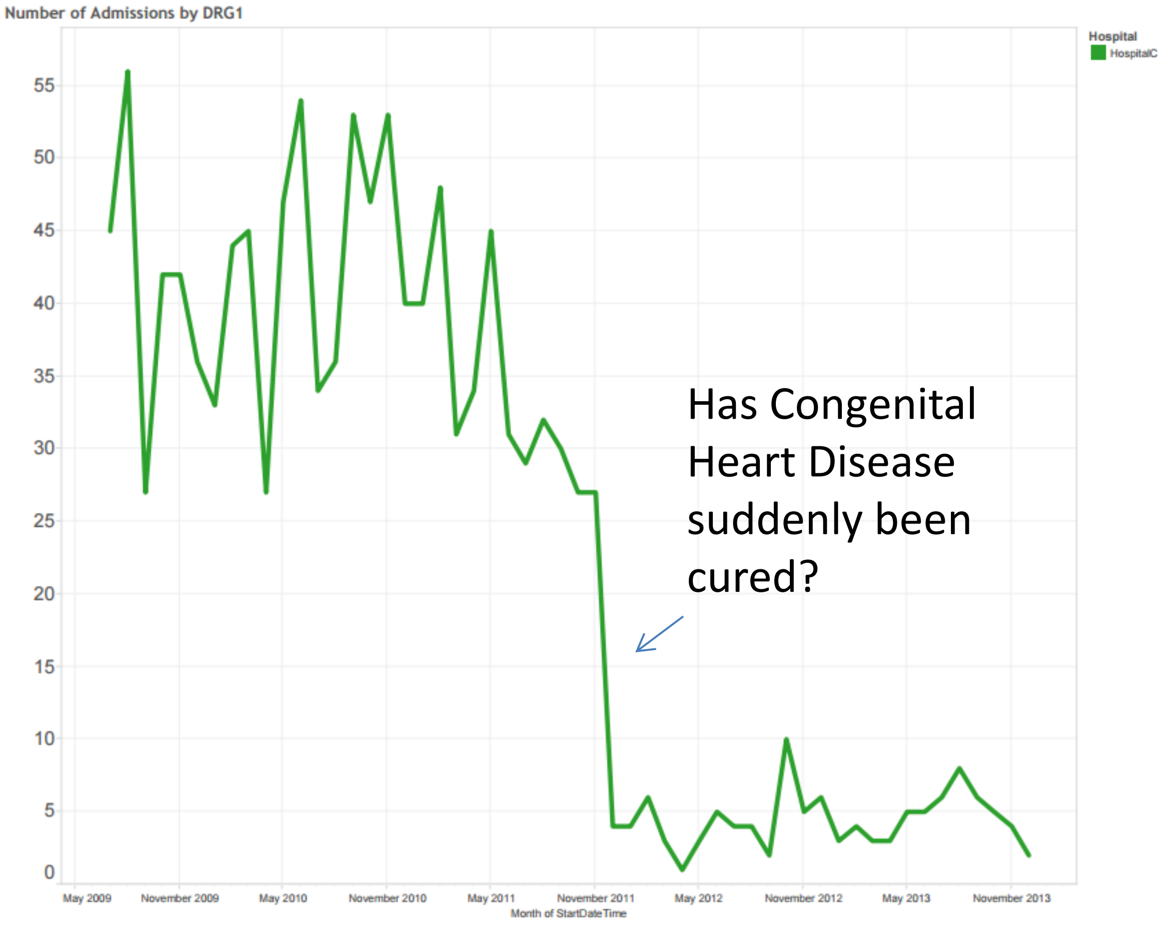
What is the reason for this?

Date >>

3. Moving Definitions

Number of Admissions for Congenital Heart Disease

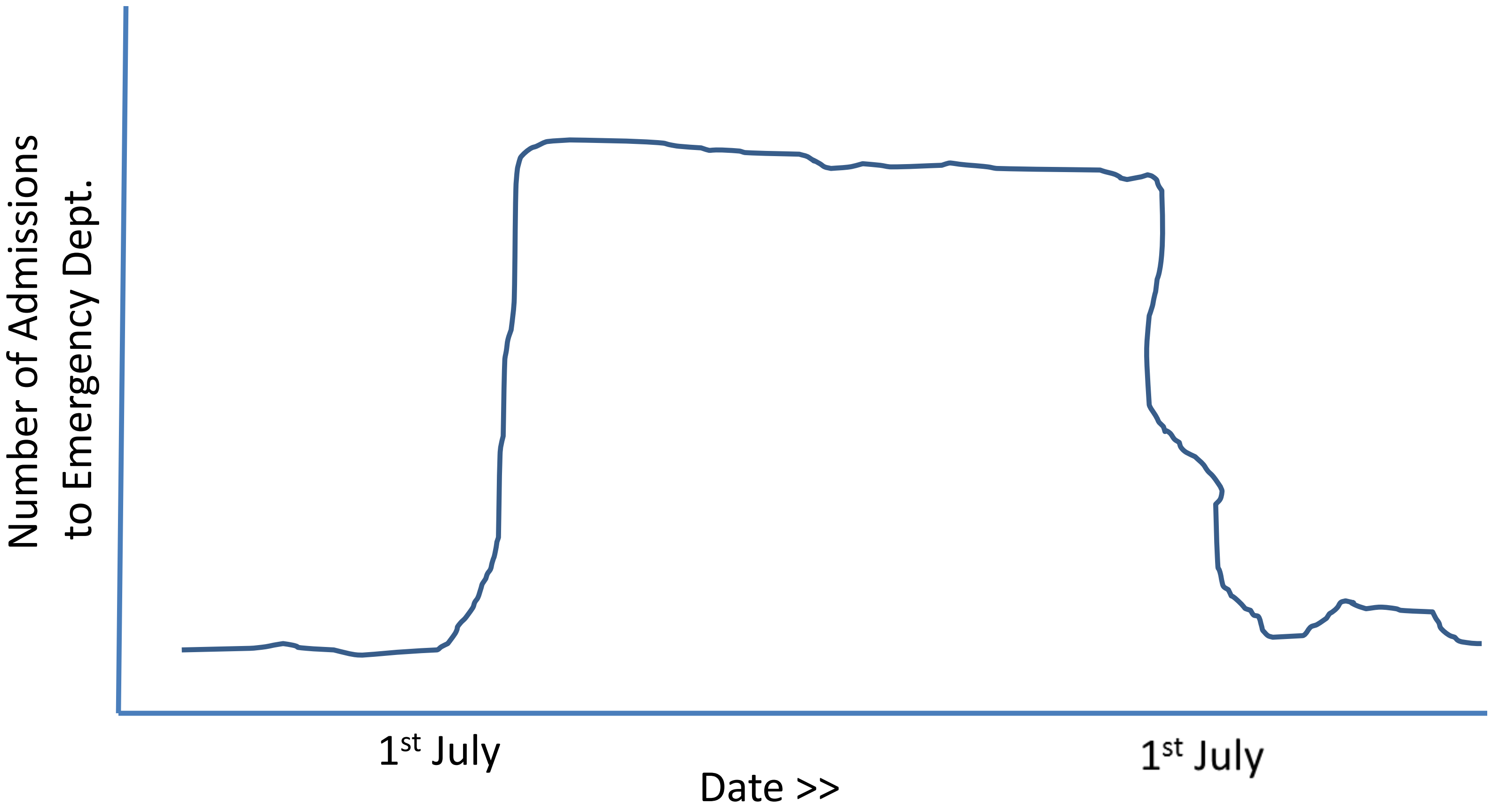
Admissions



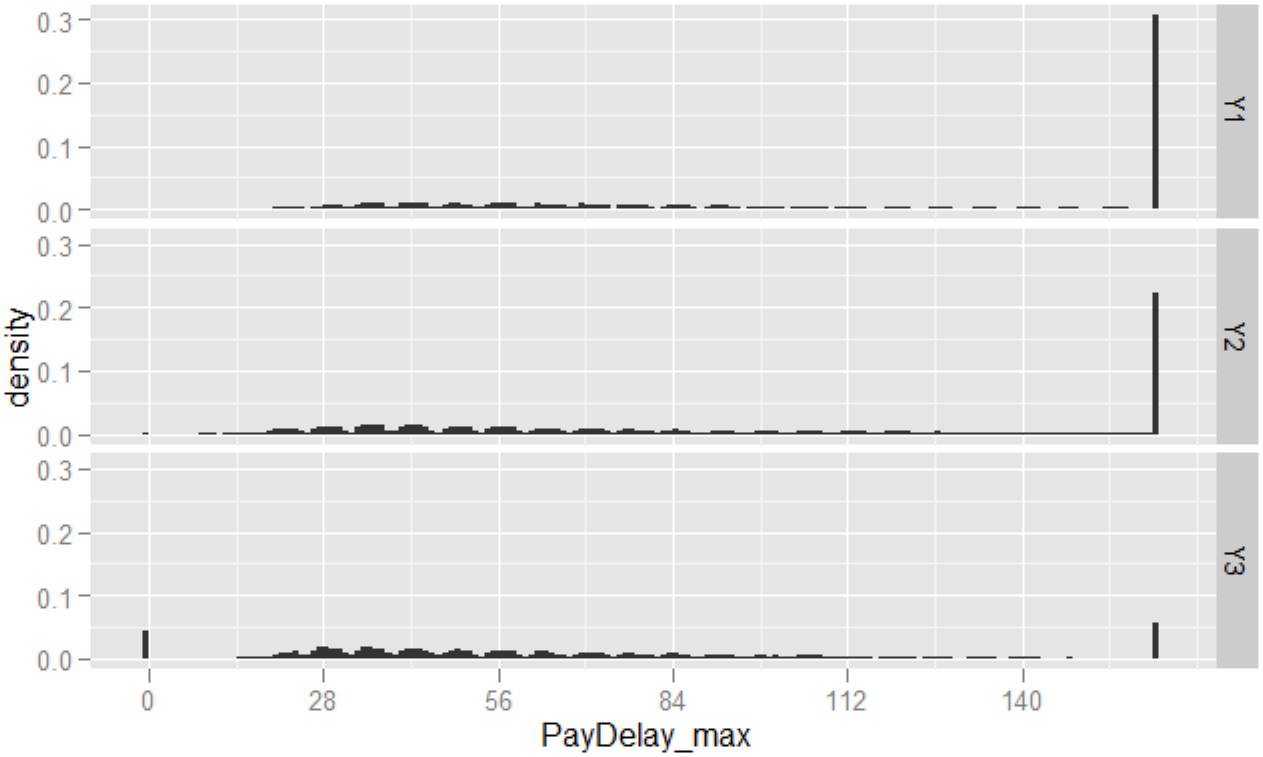
Has Congenital Heart Disease suddenly been cured?

Date >>

3. Moving Definitions



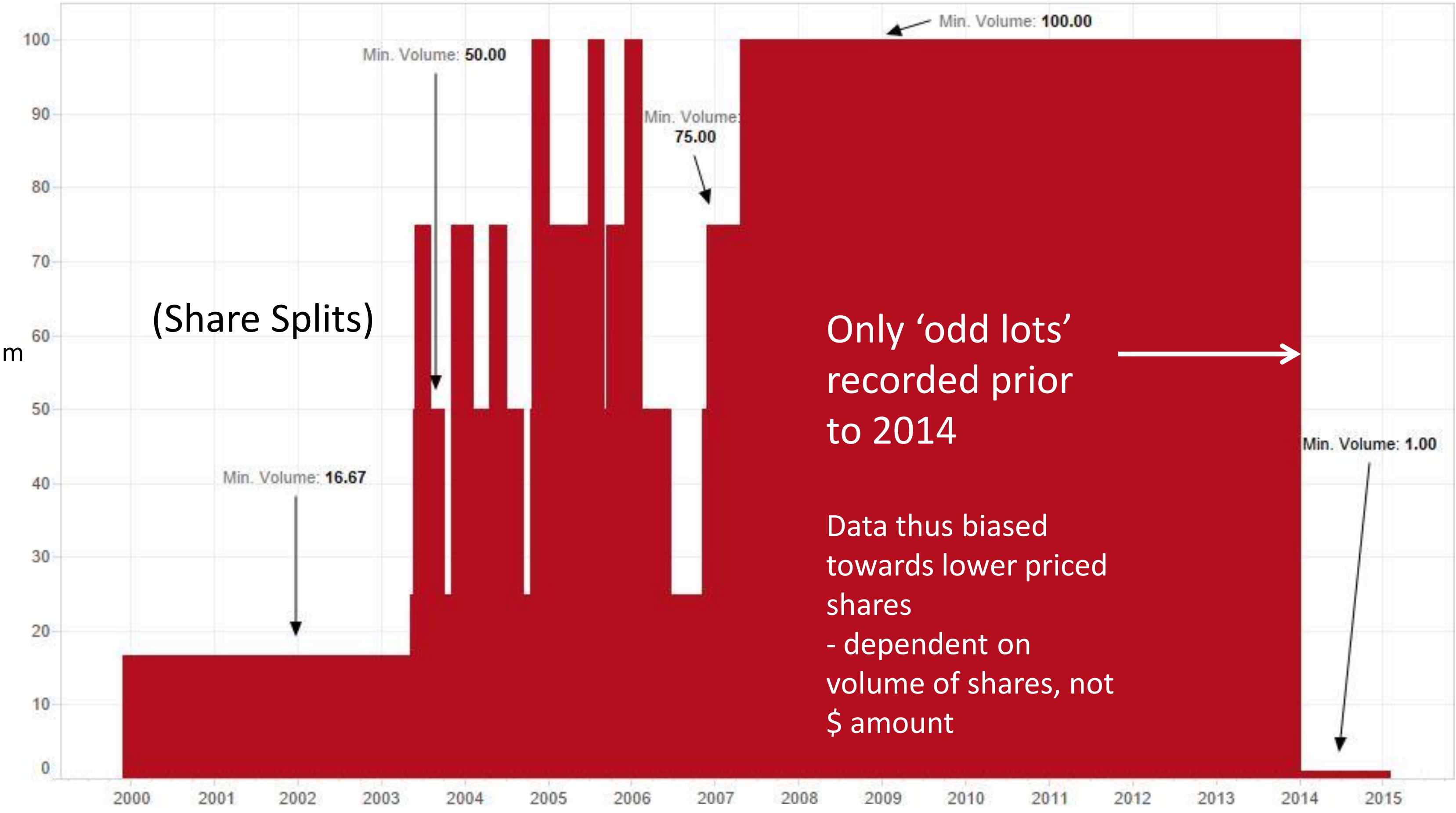
3. Moving Definitions



4. Data Capture Bias

Volume of Trades – Exchange reporting policy

Minimum
Volume



(Share Splits)

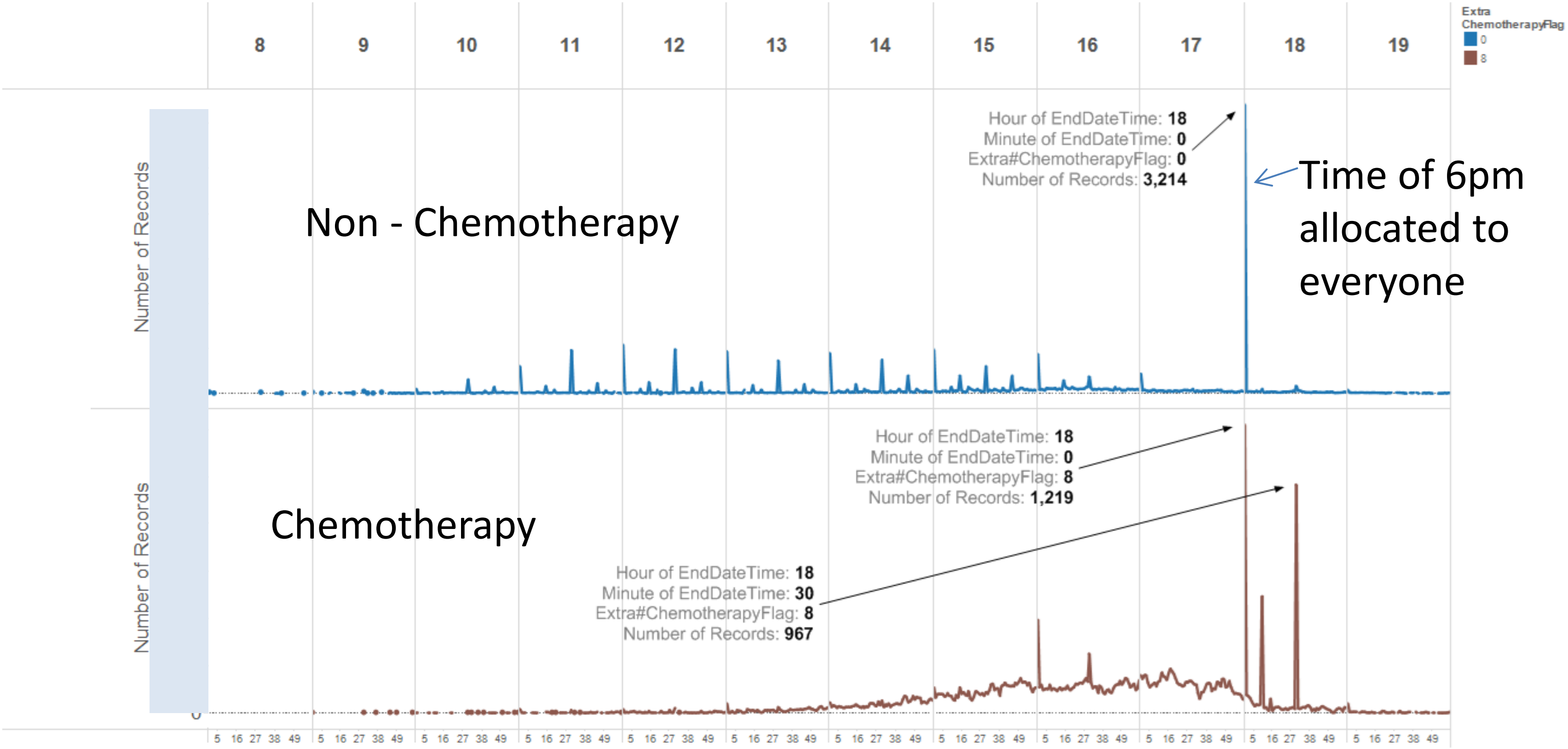
Only ‘odd lots’
recorded prior
to 2014

Data thus biased
towards lower priced
shares
- dependent on
volume of shares, not
$ amount

Date >>

5. Real or Systematic ?

Discharge Times from Hospital
(you don't actually clock out!)



Hour of EndDateTime: **18**
Minute of EndDateTime: **0**
Extra#ChemotherapyFlag: **0**
Number of Records: **3,214**

Time of 6pm allocated to everyone

Non - Chemotherapy

Hour of EndDateTime: **18**
Minute of EndDateTime: **0**
Extra#ChemotherapyFlag: **8**
Number of Records: **1,219**

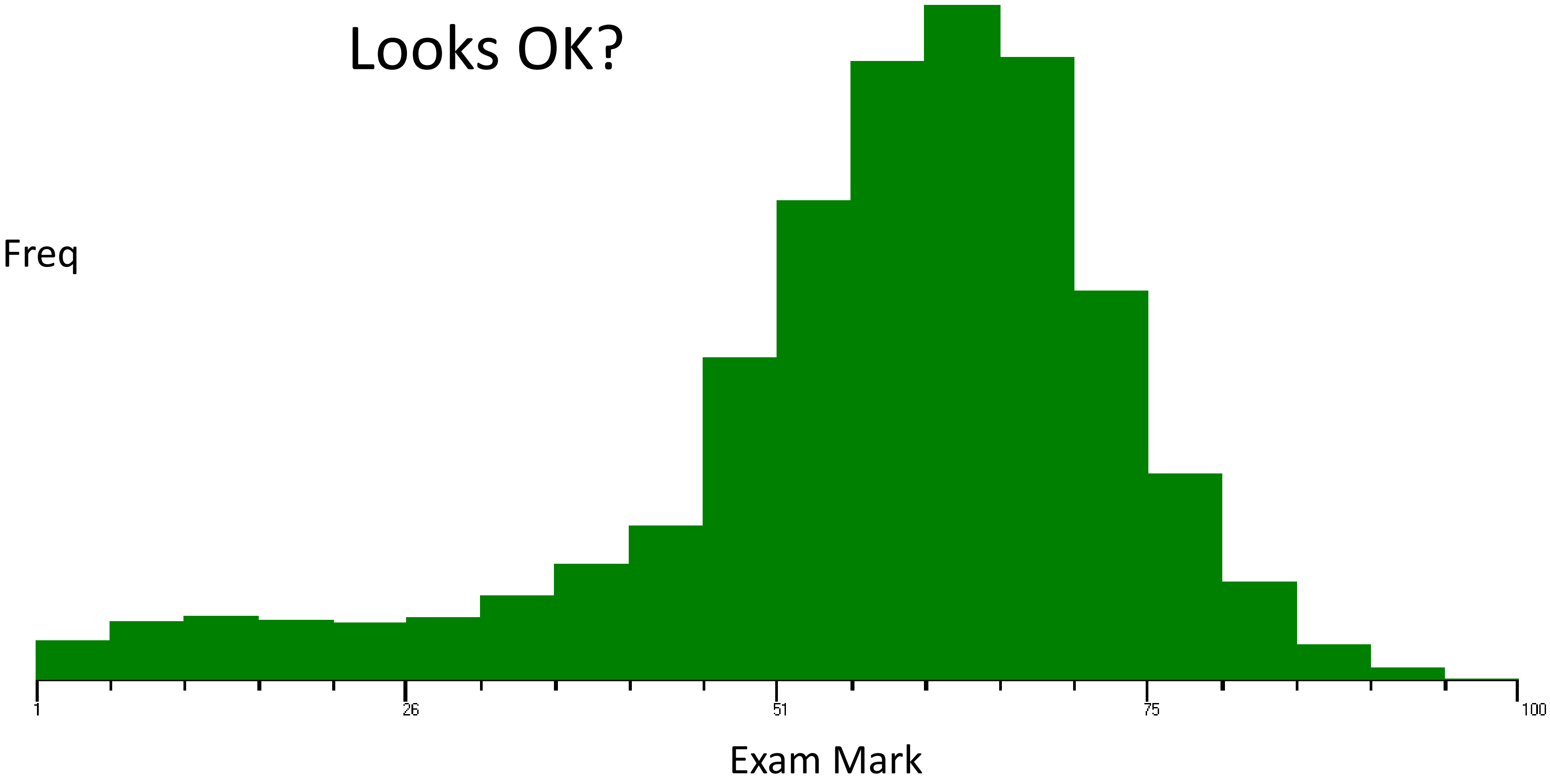
Chemotherapy

Hour of EndDateTime: **18**
Minute of EndDateTime: **30**
Extra#ChemotherapyFlag: **8**
Number of Records: **967**

Time of Day

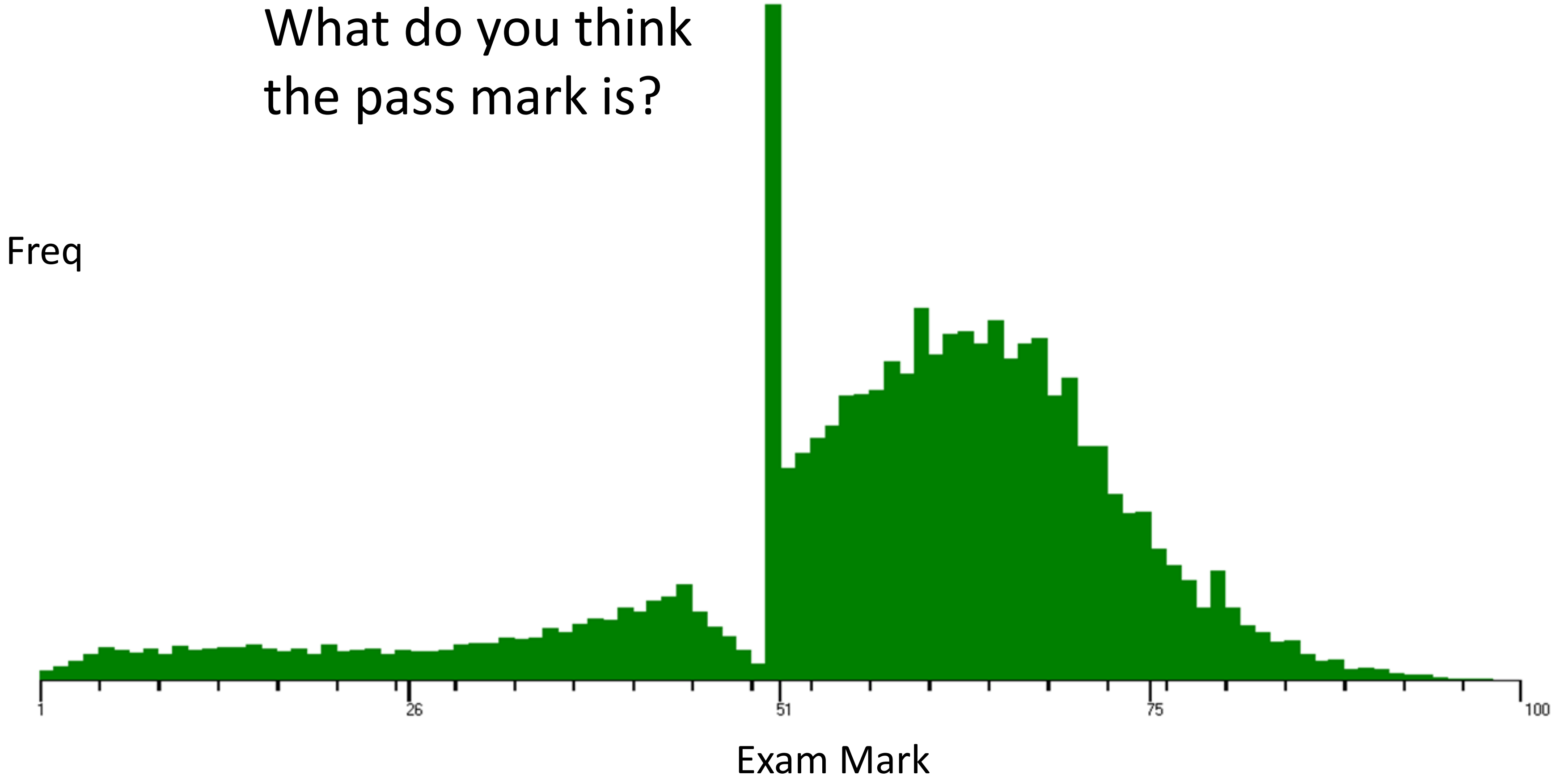
6. System Overrides

Looks OK?



6. System Overrides

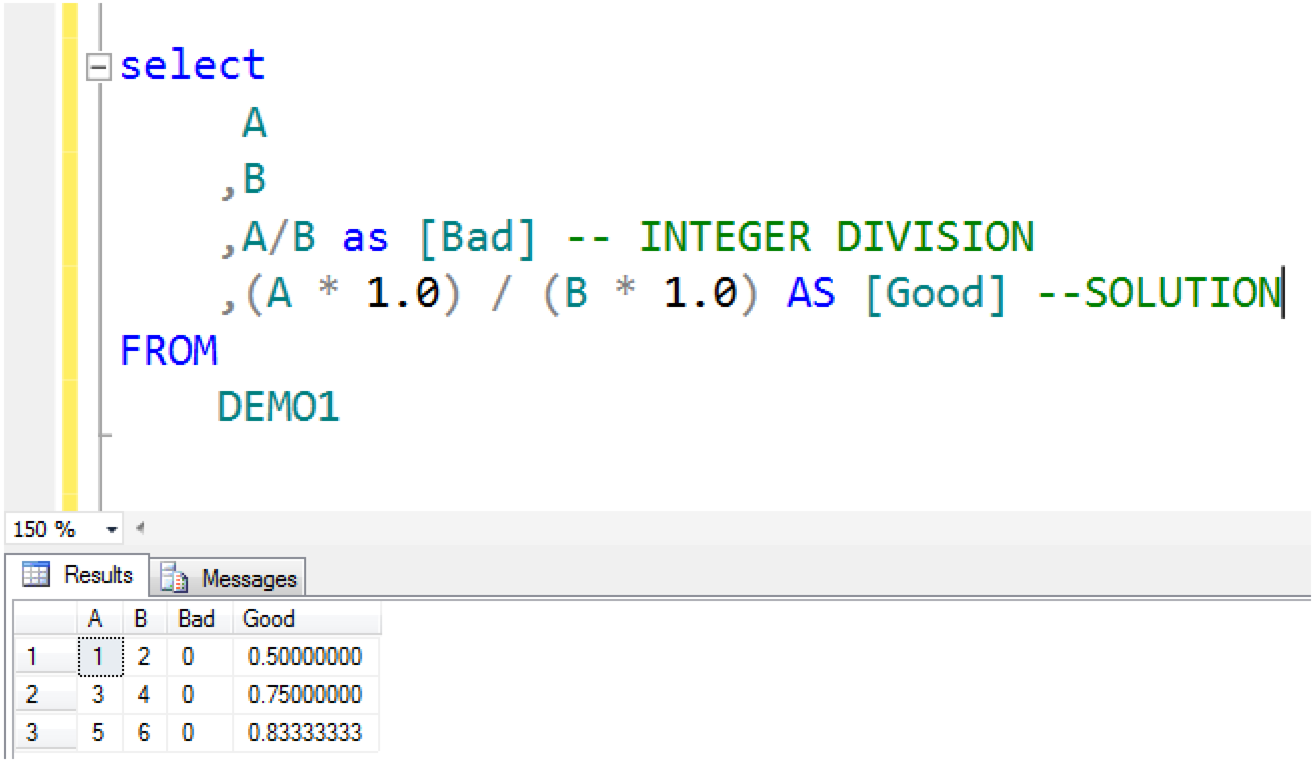
Hmmm!
What do you think
the pass mark is?



3. Pre-processing data

Avoid Self Imposed Coding Errors

1. Integer Division

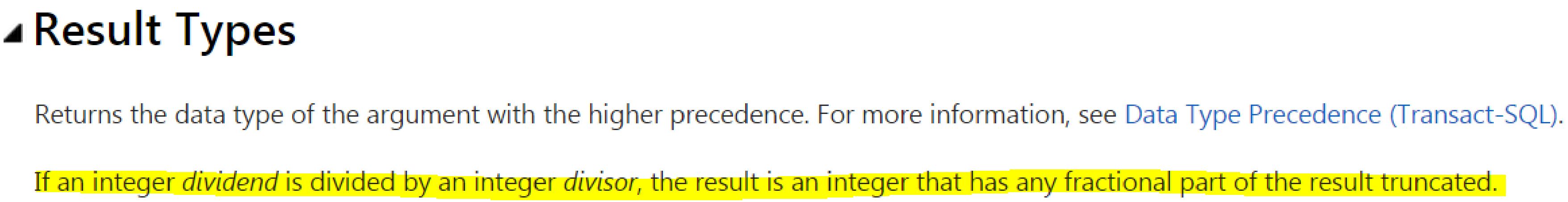


```sql
select
     A
    ,B
    ,A/B as [Bad] -- INTEGER DIVISION
    ,(A * 1.0) / (B * 1.0) AS [Good] --SOLUTION
FROM
    DEMO1
```

	A	B	Bad	Good
1	1	2	0	0.50000000
2	3	4	0	0.75000000
3	5	6	0	0.83333333

1. Integer Division

Documented – but who reads documentation!



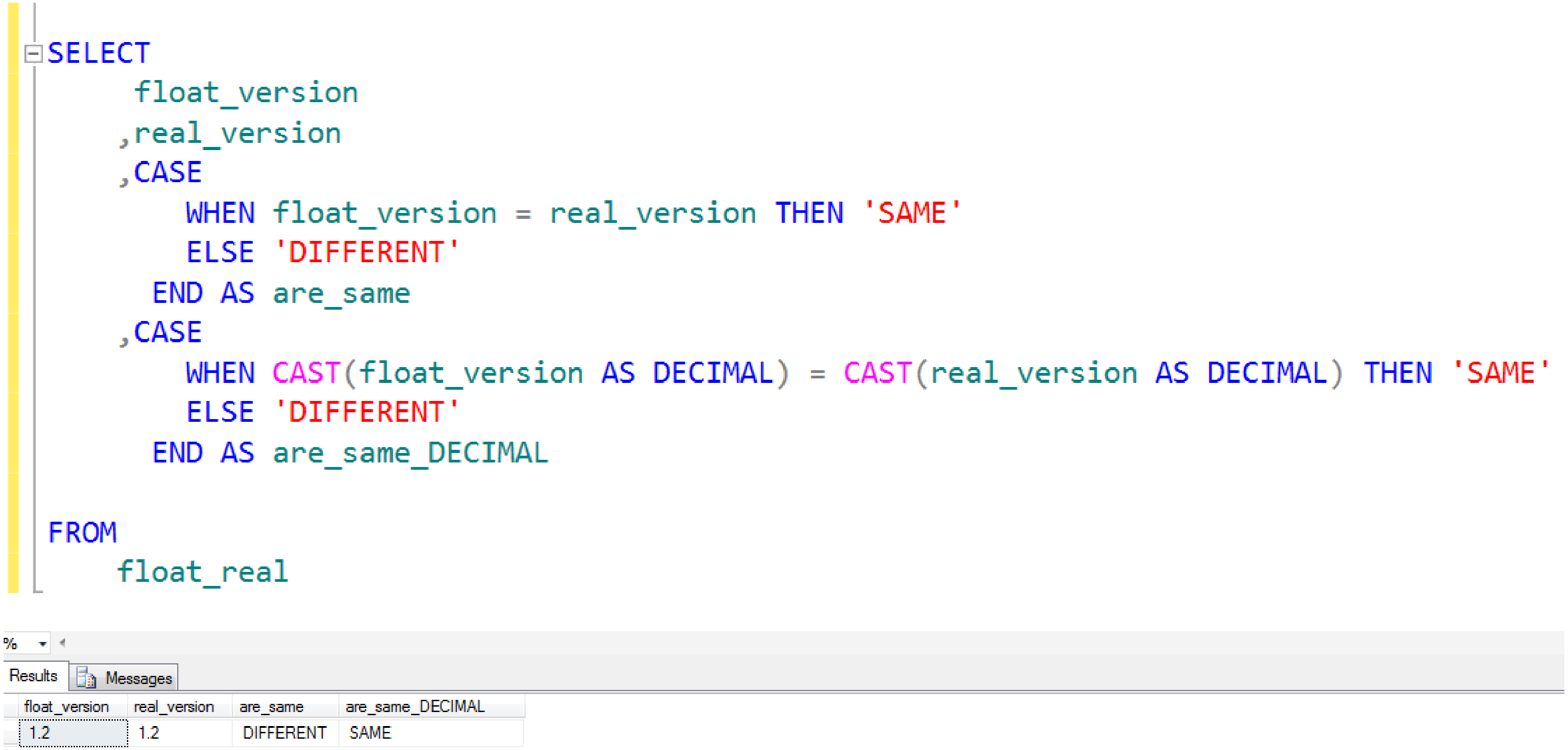
Result Types

Returns the data type of the argument with the higher precedence. For more information, see Data Type Precedence (Transact-SQL).

If an integer *dividend* is divided by an integer *divisor*, the result is an integer that has any fractional part of the result truncated.

2. Floats & Reals

Comparisons of numbers dependent on format



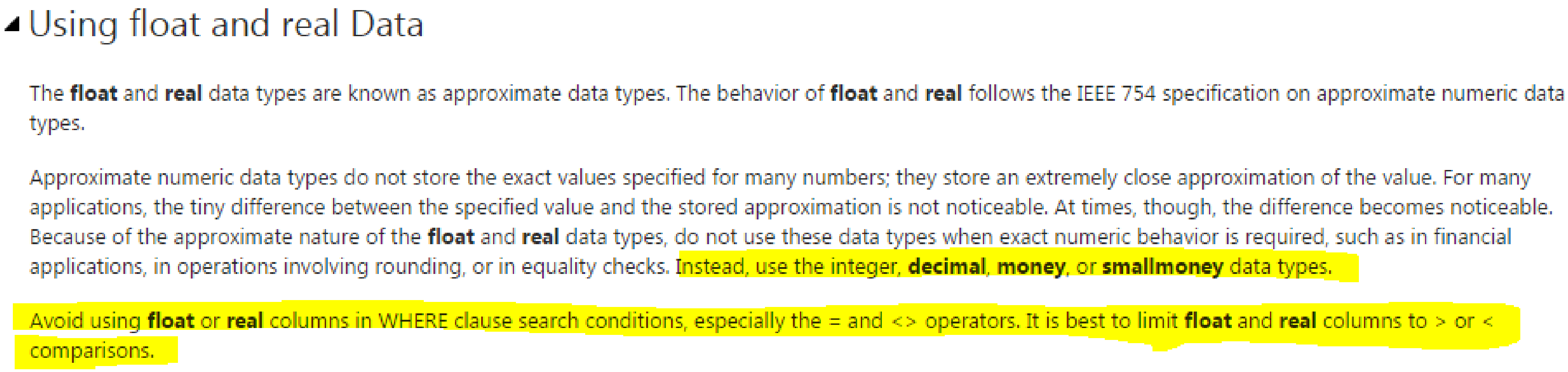
```
SELECT
     float_version
    ,real_version
    ,CASE
        WHEN float_version = real_version THEN 'SAME'
        ELSE 'DIFFERENT'
      END AS are_same
    ,CASE
        WHEN CAST(float_version AS DECIMAL) = CAST(real_version AS DECIMAL) THEN 'SAME'
        ELSE 'DIFFERENT'
      END AS are_same_DECIMAL

FROM
    float_real
```

float_version	real_version	are_same	are_same_DECIMAL
1.2	1.2	DIFFERENT	SAME

2. Floats & Reals

It is documented !!



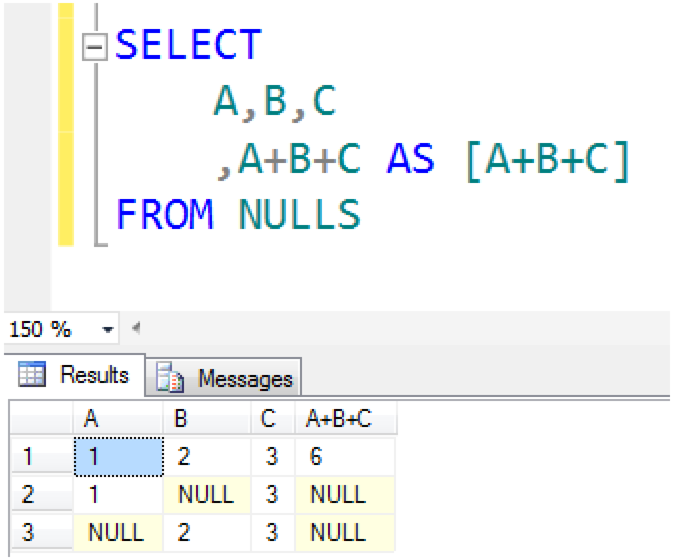
Using float and real Data

The **float** and **real** data types are known as approximate data types. The behavior of **float** and **real** follows the IEEE 754 specification on approximate numeric data types.

Approximate numeric data types do not store the exact values specified for many numbers; they store an extremely close approximation of the value. For many applications, the tiny difference between the specified value and the stored approximation is not noticeable. At times, though, the difference becomes noticeable. Because of the approximate nature of the **float** and **real** data types, do not use these data types when exact numeric behavior is required, such as in financial applications, in operations involving rounding, or in equality checks. Instead, use the integer, **decimal**, **money**, or **smallmoney** data types.

Avoid using **float** or **real** columns in WHERE clause search conditions, especially the = and <> operators. It is best to limit **float** and **real** columns to > or < comparisons.

3. Nulls

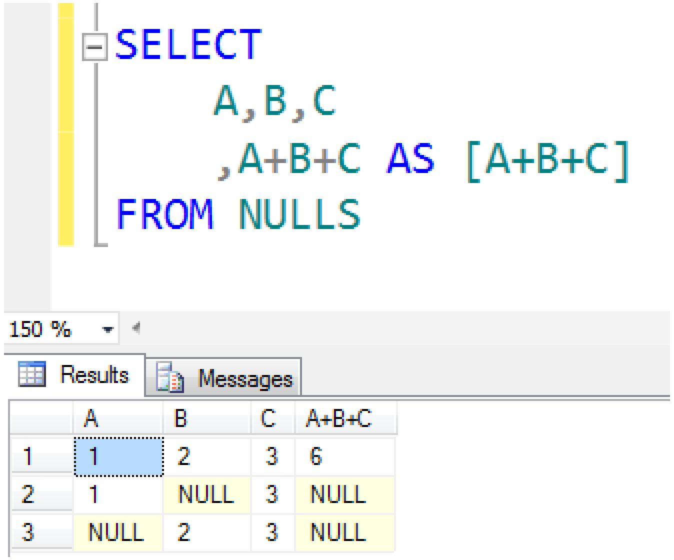


```
SELECT
        A,B,C
        ,A+B+C AS [A+B+C]
FROM NULLS
```

	A	B	C	A+B+C
1	1	2	3	6
2	1	NULL	3	NULL
3	NULL	2	3	NULL

Null is ‘unknown’ – so any calculation on records containing a Null correctly returns Null
(not necessarily intuitive)

3. Nulls



```
SELECT
     A,B,C
     ,A+B+C AS [A+B+C]
FROM NULLS
```

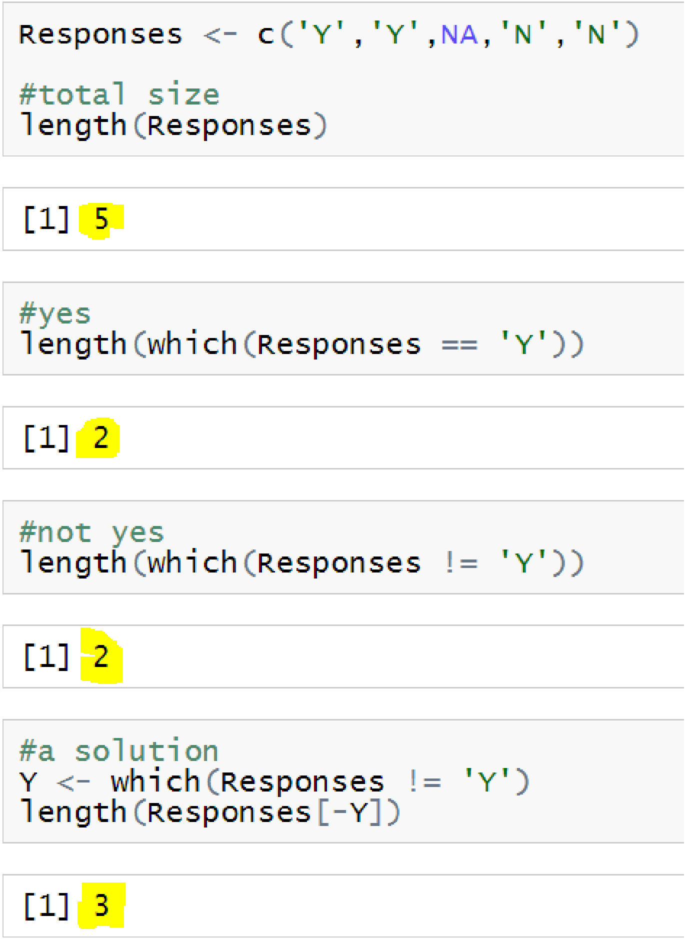
	A	B	C	A+B+C
1	1	2	3	6
2	1	NULL	3	NULL
3	NULL	2	3	NULL

Technically correct as NULL means ‘I don’t know’

…but none the less, not what you might be expecting

4. Nulls (again)

The disappearing record



```r
Responses <- c('Y','Y',NA,'N','N')

#total size
length(Responses)
```

Records = 5

```
[1] 5
```

```r
#yes
length(which(Responses == 'Y'))
```

Yes = 2

```
[1] 2
```

```r
#not yes
length(which(Responses != 'Y'))
```

Not Yes = 2

```
[1] 2
```

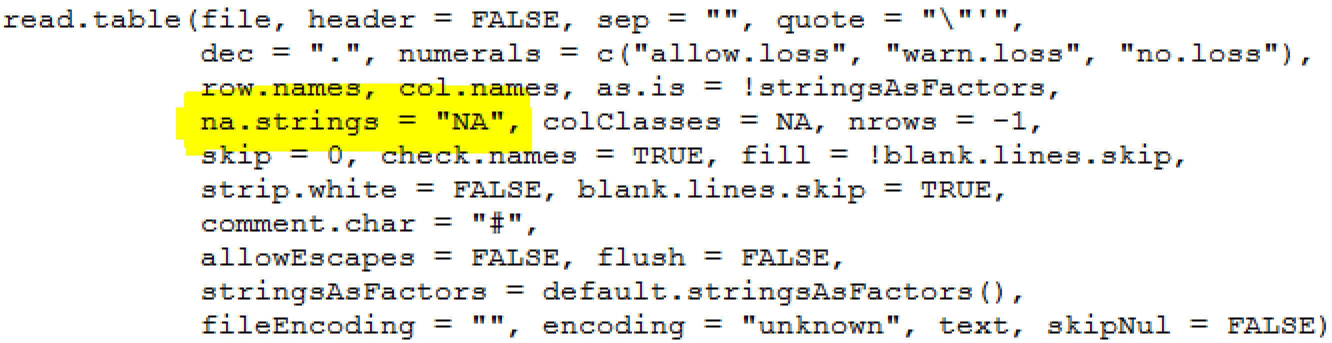
2 + 2 != 5
(not intuitive)

```r
#a solution
Y <- which(Responses != 'Y')
length(Responses[-Y])
```

```
[1] 3
```

5. Nulls (again, again)

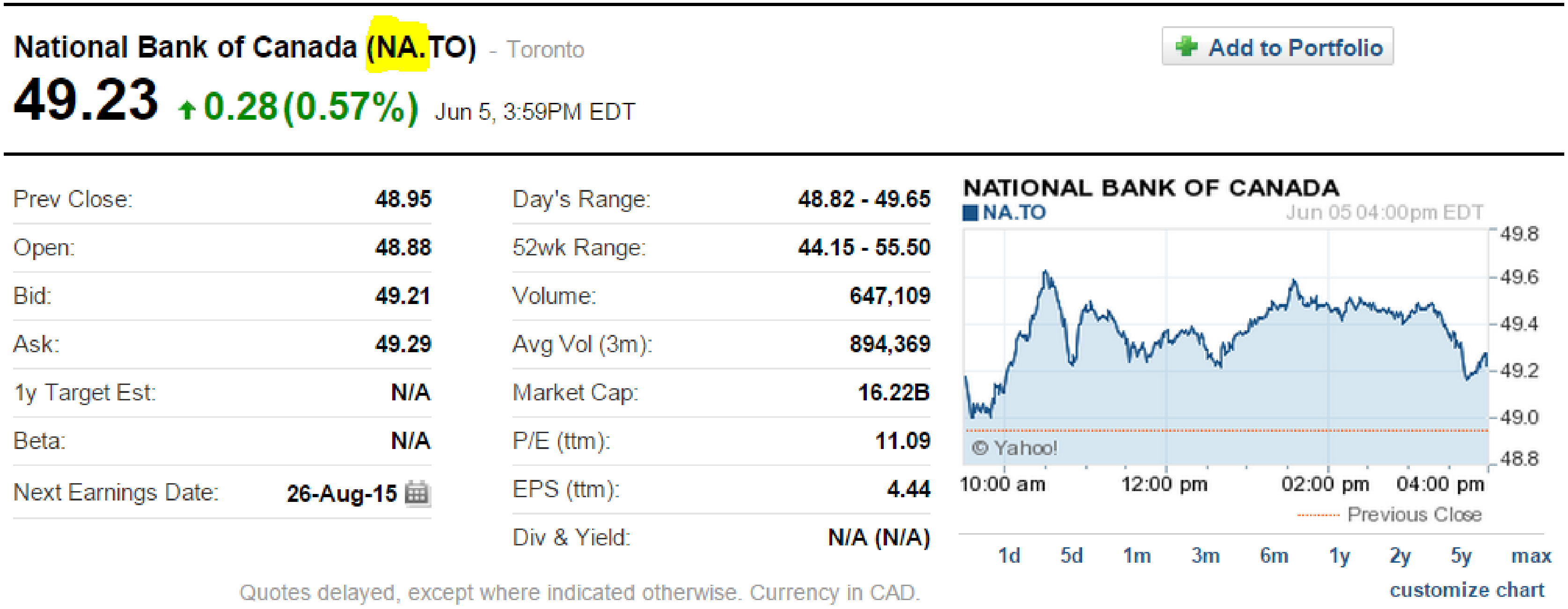
Beware Function Defaults



```
read.table(file, header = FALSE, sep = "", quote = "\"'",
           dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),
           row.names, col.names, as.is = !stringsAsFactors,
           na.strings = "NA", colClasses = NA, nrows = -1,
           skip = 0, check.names = TRUE, fill = !blank.lines.skip,
           strip.white = FALSE, blank.lines.skip = TRUE,
           comment.char = "#",
           allowEscapes = FALSE, flush = FALSE,
           stringsAsFactors = default.stringsAsFactors(),
           fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)
```

5. Nulls (again, again)

This genuinely caught me out!



National Bank of Canada (NA.TO) - Toronto

49.23 ↑0.28(0.57%) Jun 5, 3:59PM EDT

✚ Add to Portfolio

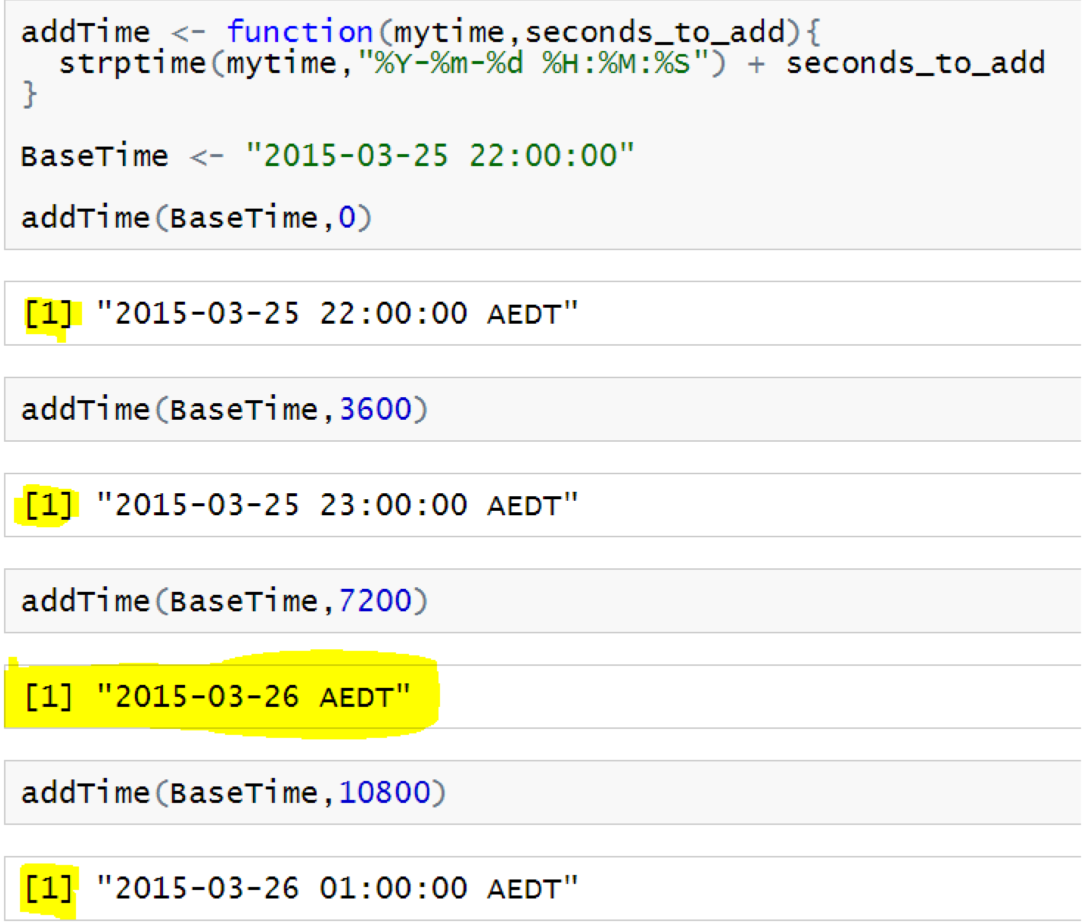
Prev Close:	48.95	Day's Range:	48.82 - 49.65
Open:	48.88	52wk Range:	44.15 - 55.50
Bid:	49.21	Volume:	647,109
Ask:	49.29	Avg Vol (3m):	894,369
1y Target Est:	N/A	Market Cap:	16.22B
Beta:	N/A	P/E (ttm):	11.09
Next Earnings Date:	26-Aug-15	EPS (ttm):	4.44
		Div & Yield:	N/A (N/A)

Quotes delayed, except where indicated otherwise. Currency in CAD.

6. Beware software quirks

- the midnight hour

(caught me out again!)



```
addTime <- function(mytime,seconds_to_add){
  strptime(mytime,"%Y-%m-%d %H:%M:%S") + seconds_to_add
}

BaseTime <- "2015-03-25 22:00:00"

addTime(BaseTime,0)
```

```
[1] "2015-03-25 22:00:00 AEDT"
```

```
addTime(BaseTime,3600)
```

```
[1] "2015-03-25 23:00:00 AEDT"
```

```
addTime(BaseTime,7200)
```

```
[1] "2015-03-26 AEDT"
```

```
addTime(BaseTime,10800)
```

```
[1] "2015-03-26 01:00:00 AEDT"
```

Bang on midnight the seconds disappear

7. Stay away from Excel !!

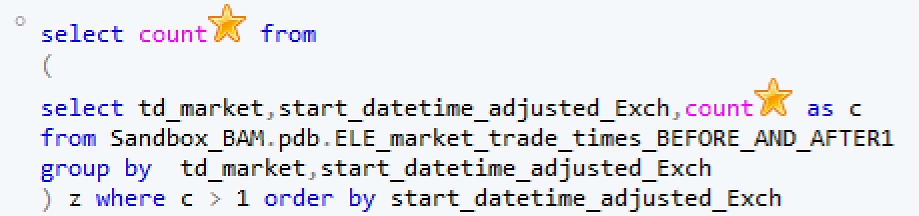
- 1-3
- Excel will convert it to 3-Jan
- Convert cell to text and it becomes 42007

- mm/dd/yy   or dd/mm/yy

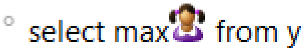
- Phil's Rules
  - avoid Excel as it has a mind of it’s own.
  - data used for modelling should go nowhere near Excel

8. Damn Smileys!

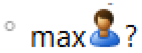
(damn Microsoft)



```
select count⭐ from
(
select td_market,start_datetime_adjusted_Exch,count⭐ as c
from Sandbox_BAM.pdb.ELE_market_trade_times_BEFORE_AND_AFTER1
group by  td_market,start_datetime_adjusted_Exch
) z where c > 1 order by start_datetime_adjusted_Exch
```



select max👧 from y



max👦?

4. Predictive Modelling

If it looks too good to be true
– it normally is

Predictive Modelling

- These days, all you need to know is:
  - Ensembling
  - Over fitting (or how to avoid it)
  - Calculating Variable Importance
    - Helps detect information leakage

If it’s too good to be true…

1. University Attrition
   (voluntary or involuntary)

2. Insurance Claims
   (level of cover)

3. ID is a proxy for the outcome
   (kdd Cup)

Thank you for listening