Monitoring Process Change with Bayesian Methods

Mick Cooney
michael.cooney@applied.ai

2 September 2015

Structure of Talk

- Discussion of Problem
- Bayesian Analysis and the Beta Distribution
- Adding Layers of Noise
- Distribution Distances and f-divergences

Monitoring Process Change

- NOT Change-point Analysis
- Time of change known - want to measure change effect
- Have measured metrics
- Need to determine change vs noise
- Generic technique for the problem

Sales-call Conversions

- Assume a binary outcome
- Conversion rate of sales calls to actual sales
- Amount irrelevant
- Data summarised monthly
- Change due to internal improvements leading to faster turnaround

Sales-call Conversions

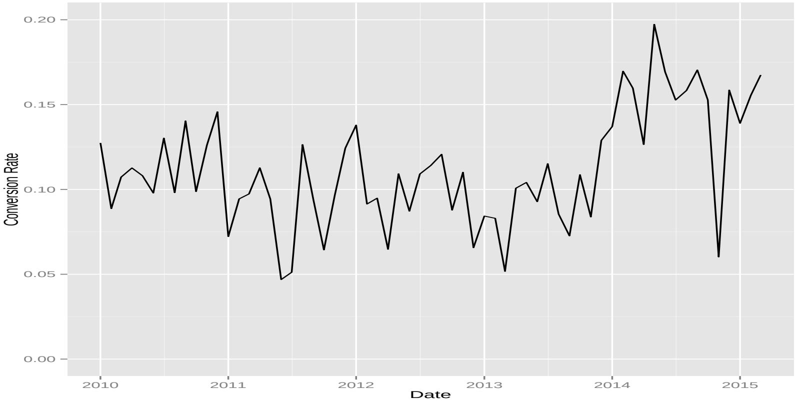
- Assume a binary outcome
- Conversion rate of sales calls to actual sales
- Amount irrelevant
- Data summarised monthly
- Change due to internal improvements leading to faster turnaround

Generating Data

Want to generate time-series for θ, use normal distribution:

```
generate_process_rates <- function(mu0 = 0.10, sd0 = 0.03, mu1 = 0.15, sd1 = 0.03,
                                   start_date  = as.Date("2010-01-01"),
                                   end_date    = as.Date("2015-03-01"),
                                   change_date = as.Date("2014-01-01")) {

    month_vector <- as.yearmon(seq(start_date, end_date, by = "month"));
    switch_month <- as.yearmon(change_date);

    switch_idx <- match(switch_month, month_vector);

    pre_rate  <- rnorm(switch_idx - 1, mu0, sd0);
    post_rate <- rnorm(length(month_vector) - switch_idx + 1, mu1, sd1);

    rate_dt <- data.table(rate_date = as.Date(month_vector), underlying_rate = c(pre_rate, post_rate));

    return(rate_dt)
}
```

```
plot_rate_dt <- generate_process_rates(mu0 = 0.10, sd0 = 0.02, mu1 = 0.15, sd1 = 0.03);

qplot(rate_date, underlying_rate, data = plot_rate_dt, geom = 'line', ylim = c(0, 0.2),
      xlab = 'Date', ylab = 'Conversion Rate');
```



```
generate_counts <- function(rate_dt, month_count) {
    rate_dt <- data.table(rate_dt, month_count = month_count);

    rate_dt[, conversion_count := mapply(rbinom, n = 1, month_count, underlying_rate)];
    rate_dt[, conversion_rate  := conversion_count / month_count];

    return(rate_dt);
}

generate_yearly_data <- function(rate_dt) {
    year_dt <- rate_dt[, list(a = sum(conversion_count), b = sum(month_count - conversion_count)),
                         by = list(data_year = format(rate_date, '%Y'))];
    year_dt[, c("cum_a", "cum_b") := list(cumsum(a) + 1, cumsum(b) + 1)];

    distrib_dt <- year_dt[, generate_beta_plot_data(cum_a, cum_b), by = data_year];

    return(distrib_dt);
}
```

Summary

R package: `mcmortswap`
`https://bitbucket.org/appliedai/mcmortswap`

michael.cooney@applied.ai
mickcooney@gmail.com

Slides available on github:
`https://github.com/kaybenleroll/dublin_r_workshops`