Using Time Series Models for Defect Prediction in Software Release Planning

James Tunnell
Central Washington University
Computational Science Program

May 21, 2015

Outline

Introduction

Release Planning Objectives

- Two primary objectives of software release planning are:
  - Improving functionality
  - Maintaining quality
- Both of these objectives are constrained by limits on development time and cost.

Quality Control

- Software defects (bugs) are inevitable
- Sufficient time should be available to ensure good quality (by testing and bug-fixing)
- Otherwise, there is a risk of
  - Low quality (failure to meet objective)
  - Schedule slip (failure to respect constraint)
- This quality control (QC) time can be allowed for by limiting the scope of work in the planned release

Quality Control (cont'd)

- To support release planning, QC time can be estimated
- Assumption: QC time depends (at least partly) on the number of software defects introduced
- Then, a basis for estimating QC time would be the predicted number of defects

Defect Prediction

- Approaches to defect prediction tend to focus on either
  - Code analysis
    - Lines of code
    - Number of decisions
    - Code churn
  - Historical information
    - Regression analysis
    - Time series modeling
- A multivariate time series model with exogenous inputs was chosen

Motivation

Explanatory Model

- Assumption: the number of defects in the future depends on more than just the number of defects in the past
- A defect prediction model that depends only on previous numbers of defects is not explanatory
- Such a non-explanatory model would always predict the same number of defects

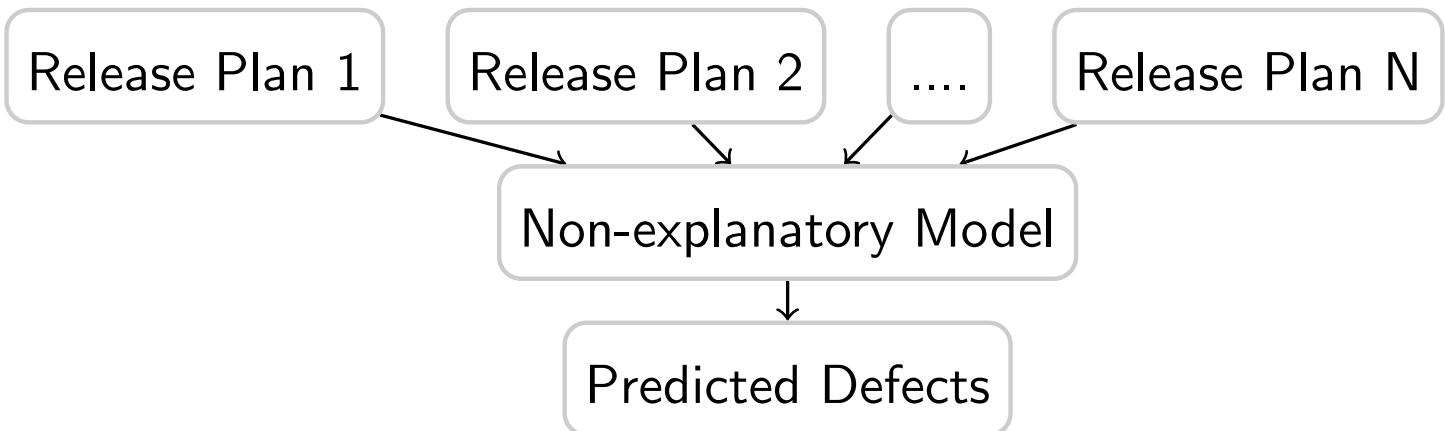

Figure : *A non-explanatory model.*

Explanatory Model (cont'd)

- A model could also depend on the key factors of a release plan
- This would be an explanatory model structure
- Such a model can potentially predict a different number of defects for every release plan

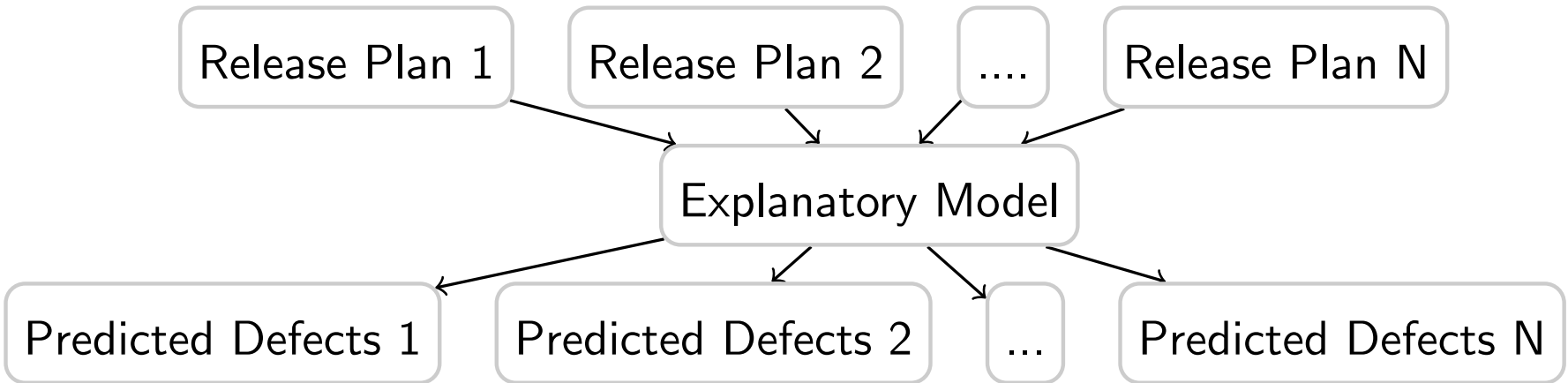

Figure : *An explanatory model.*

Time Series Modeling

Time Series

- A time series is a collection of observations that occur in order
- The process underlying a time series is assumed to be stochastic (non-deterministic)
- Each observation might depend on one or more previous observations
- This dependence is termed *autocorrelation*

Autoregressive Models

- A basic autoregressive (AR) model is a linear combination of previous values
- A white noise term accounts for stochastic fluctuation
- An $AR(p)$ model for predicting a value X at time t is

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \epsilon_t \tag{1}$$

where $\phi_1, \phi_2, ..., \phi_p$ are the p parameters, c is a constant, and ϵ_t is the white noise term

Autoregressive Models (cont'd)

- Extending the AR model to be multivariate results in a Vector AR (VAR) model
- This model can support time series for defect count, improvements, and new features

More on Modeling

Endogeneity vs Exogeneity

- Endogeneity vs Exogeneity
  - We only attempt to explain the number of defects, so all other model inputs are exogenous.
  - This makes a VAR model would become a VARX model
- Stationarity
  - The models discussed so far require stationary time series data
  - Statistical tests are applied to check for stationarity
  - Non-stationary time series are differenced

Modeling Methodology

Time Series Modeling Methodology

- Time series modeling methodology typically involves
  1. Specification
  2. Estimation
  3. Diagnostic Checking
  4. Selection

Specification & Estimation

- A $VARX(p)$ model is specified by choosing an order p
- Model order is the number of autoregressive terms
- This affects the number of parameters included in the model
- The model order is limited to avoid having too many parameters relative to the number of observations
- Maximum model order is p_max
- Models parameters are estimated for orders $1, 2, ..., p_{max}$

Diagnostic Checking

- Diagnostics can tell if a model should be rejected
- First diagnostic is for stability
  - AR model can have infinite impulse response
  - To be stable, the roots of the characteristic equation must lie outside the unit circle
  - Equivalently, the inverse of the roots must lie inside the unit circle
- Next diagnostic is residual autocorrelation
  - Model residuals should be indistinguishable from white noise
  - White noise is uncorrelated (no autocorrelation)
  - Ljung-Box test forms a statistic from the autocorrelation of the residuals

Model Selection

- Model selection criteria are used to compare models according to their fit
- Penalties for residual error and the number of parameters
- Some common selection criteria
  - Akaike Information Criterion (AIC)
  - AIC with correction (AICc)
  - Bayesian Information Criterion (BIC)
- Parameter penalty is more severe for BIC and AICC than for AIC
- AIC will be used, since the number of parameters is already limited in the specification step

Data Methodology

Data source

- Data for time series modeling will be derived from project historical data
- This historical data can be found in the project issue tracking system (ITS)
- The issues in an ITS can be bugs, features, improvements, etc.
- The *MongoDB* software project was selected to try out the modeling methodology
  - The project has been actively developed since 2009
  - Data from versions 0.9.3 through 3.0.0-rc6 are used
  - This dataset contained 7042 issues
- Only data on issues leading to software changes were kept.

Data Sampling

- Time series data was drawn for the issue data by
  - Sampling
  - Testing for stationarity
  - Windowing samples to limit exposure to underlying process changes
- A 78-week time window (approximately 18 months) was chosen
- This window was applied over all the data by a sliding window
- Modeling methodology is applied to each window

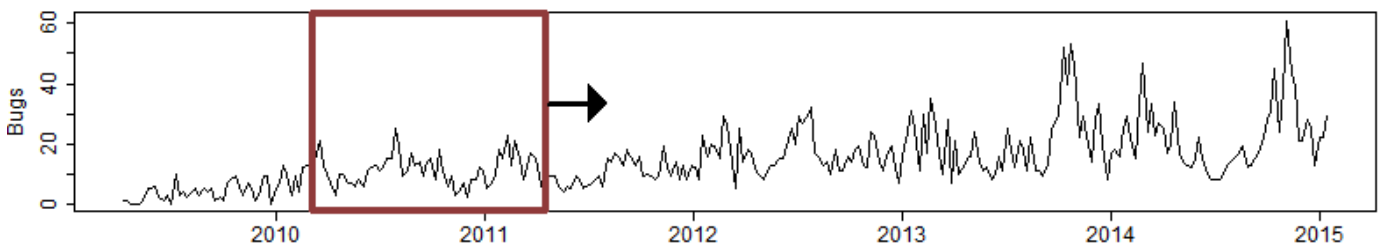

Figure : *Sliding time window moves over the entire time series.*

Results

MongoDB Time Series Data

Time series data was obtained by sampling the *MongoDB* dataset with a 7-day sample period

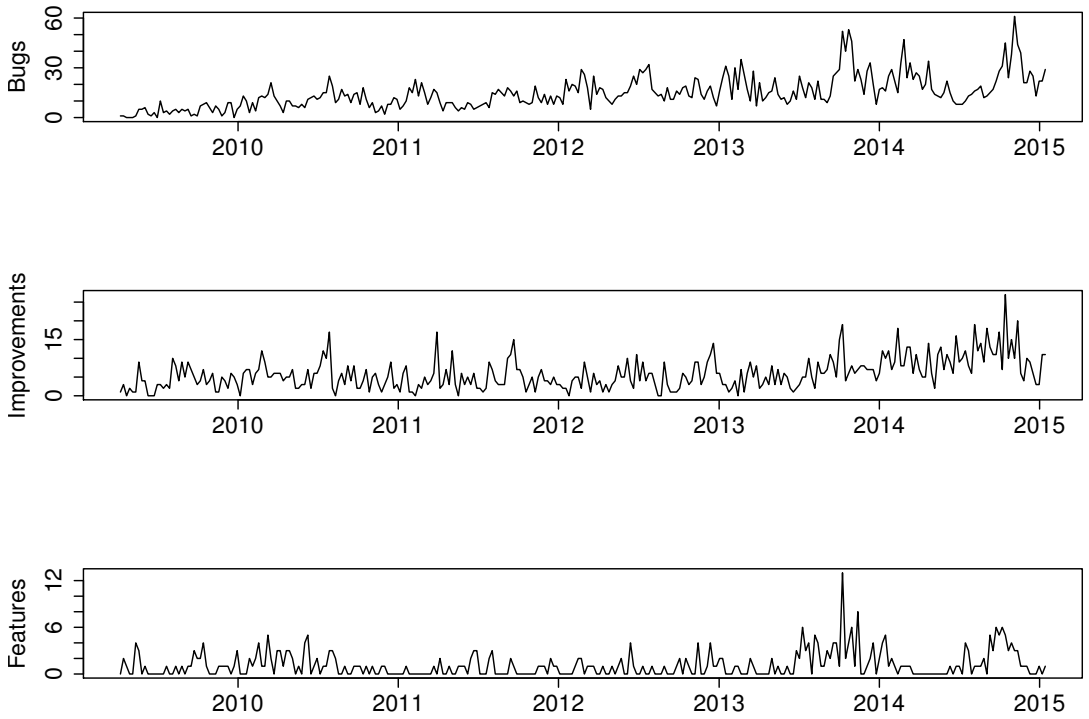

Figure : *MongoDB time series data*

Stationarity & Differencing

- At first, ADF and KPSS test results disagreed
- After differencing, test results agreed

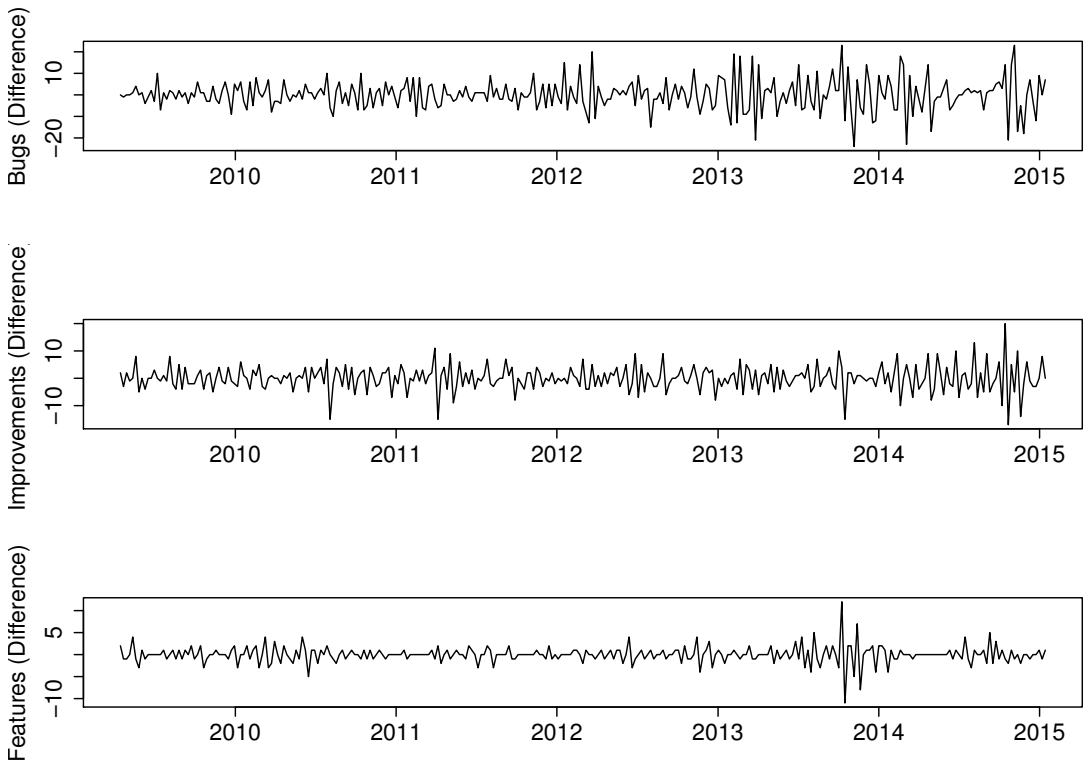

Figure : *MongoDB time series data after differencing*

One-step Predictions

One-step predictions are a visual indication of model fit. Here are the one-step predictions for three of the models.

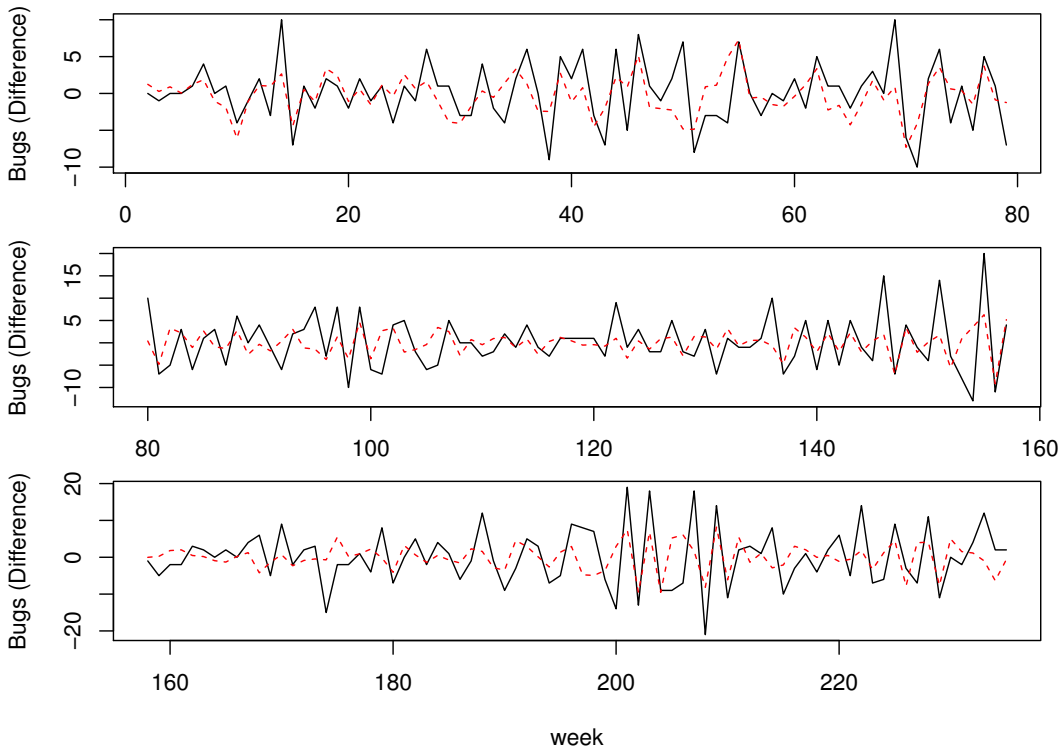

Figure : *Actual values (solid) vs. one-step predictions (dotted)*

Sliding Window Forecasts

- How useful is the VARX model in general, considering these conflicting results?
- To find out, a sliding 78-week window was used
- The sliding window started at the first sample period, and was shifted by one sample period after modeling
- Only the actual number of improvements and features were used in this forecasting

Sliding Window Forecasts (cont'd)

- Errors between the mean forecasted and actual number of bugs is shown as a histogram
- The histogram appears to be normally distributed (good)
- The variability is quite large (bad)
- The actual number of bugs was inside the 90% confidence interval for 23.87% of the sliding window ranges

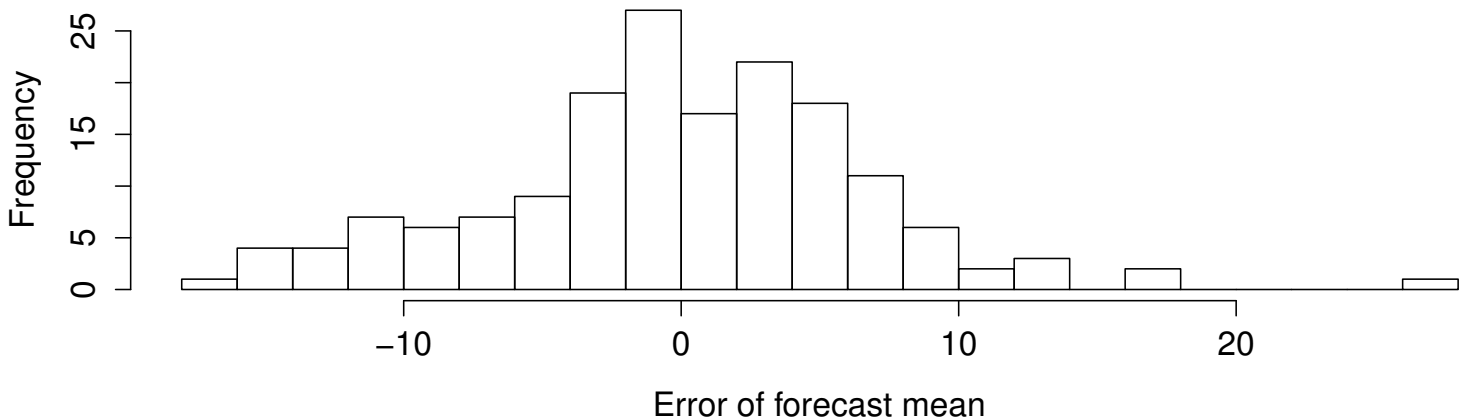

Figure : *Histogram of errors in forecast mean obtained using a 78-week sliding window.*

Conclusion & Future Work

Conclusions

- The VARX modeling methodology was successfully applied to the time series data collected from the *MongoDB* project
- Models were created for each of three time windows
- A model was selected for each window
- Forecast results using the models were inconclusive
- A better picture of the prediction performance was obtained using a sliding window
- This resulted in a normally distributed error in the mean forecasted values
- A low proportion (23.87%) of the sliding window ranges included the actual number of bugs in the 90% confidence interval
- These results may indicate that a VARX model will not be useful to make predictions for the the MongoDB dataset

Future Work

Having applied the VARX time series model to one project dataset, a next step is to apply the methodology to other software project data sets, such as *Eclipse* or *Mozilla*, to more conclusively determine the model's usefulness.