Using Time Series Models for Defect Prediction in Software Release Planning

James Tunnell
Central Washington University
Computational Science Program

June 5, 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

Release Plan Optimization

- A release plan is formed by selecting features and improvements to work on
- Release plans can be compared by the expected revenue they will generate
- This optimization problem is posed as The Next Release Problem (NRP)

Release Plan Optimization (cont'd)

- The NRP is an abstract optimization problem
- In practice, QC time should be considered to ensure constraints are respected
- With the help of a defect prediction model, QC time can be estimated
- In this context, release plans are being compared
- For a defect prediction model to be useful, it should depend in some way on the basic elements of the release plan (planned new features and improvements)

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

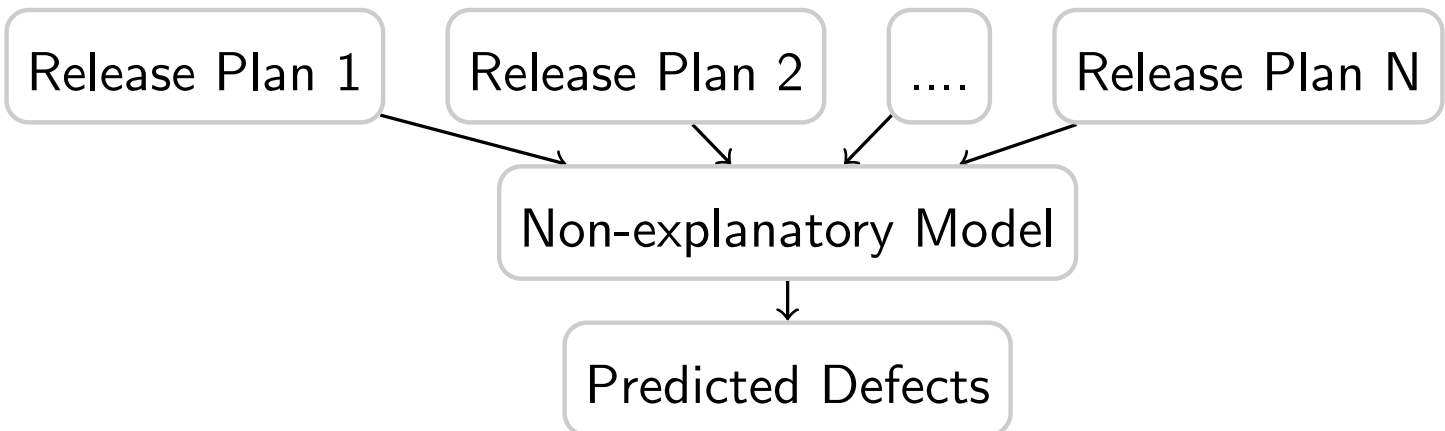


Figure : *A non-explanatory model.*

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

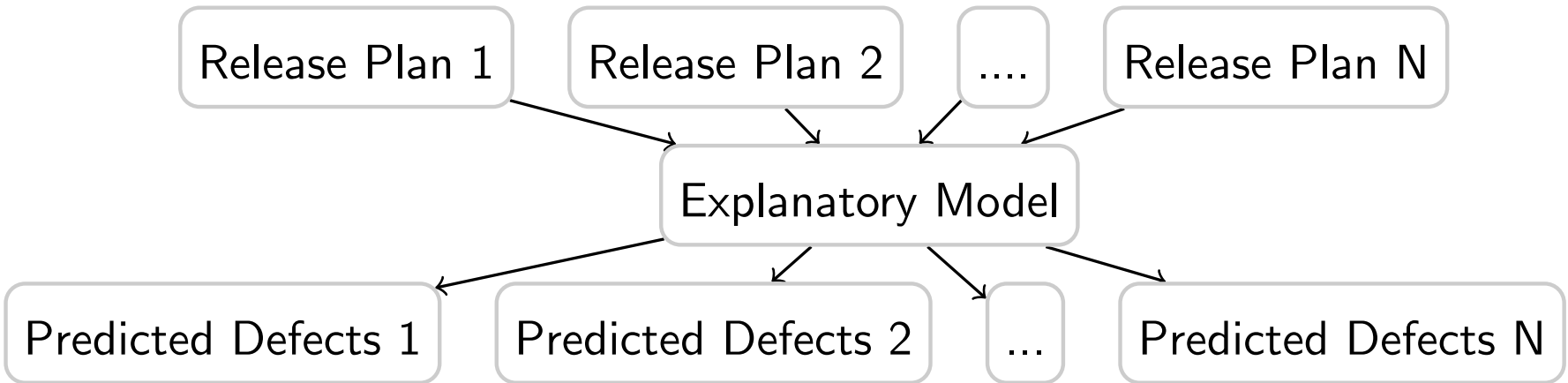


Figure : *An explanatory model.*

Time Series Modeling

Stationarity

- A stationary time series has time-invariant statistics
- The time series models so far require time series to be stationary
- Differencing a non-stationary series may produce a stationary series
- Stationary can be determined by testing for trends

Deterministic Trends

- A time series with a deterministic trend has a non-constant mean
- The time series movements will generally follow the deterministic function
- Fluctuations above or below this function are non-permanent
- Such a time series is said to be stationary around a deterministic trend

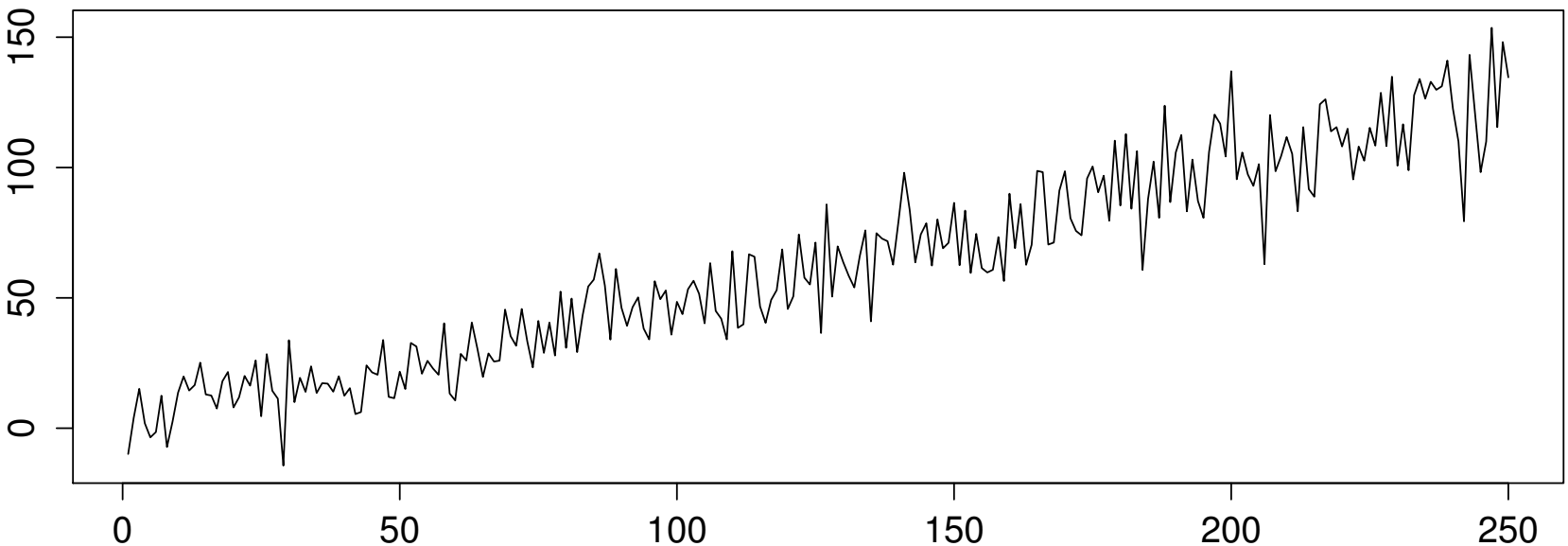


Figure : *Time series with a deterministic trend.*

Stochastic Trends

- A stochastic trend shows permanent effects due to random variations
- A series with stochastic trend will not necessarily fluctuate only close to the area of a deterministic function
- A time series with stochastic trend is non-stationary
- Differencing can be used to remove a stochastic trend

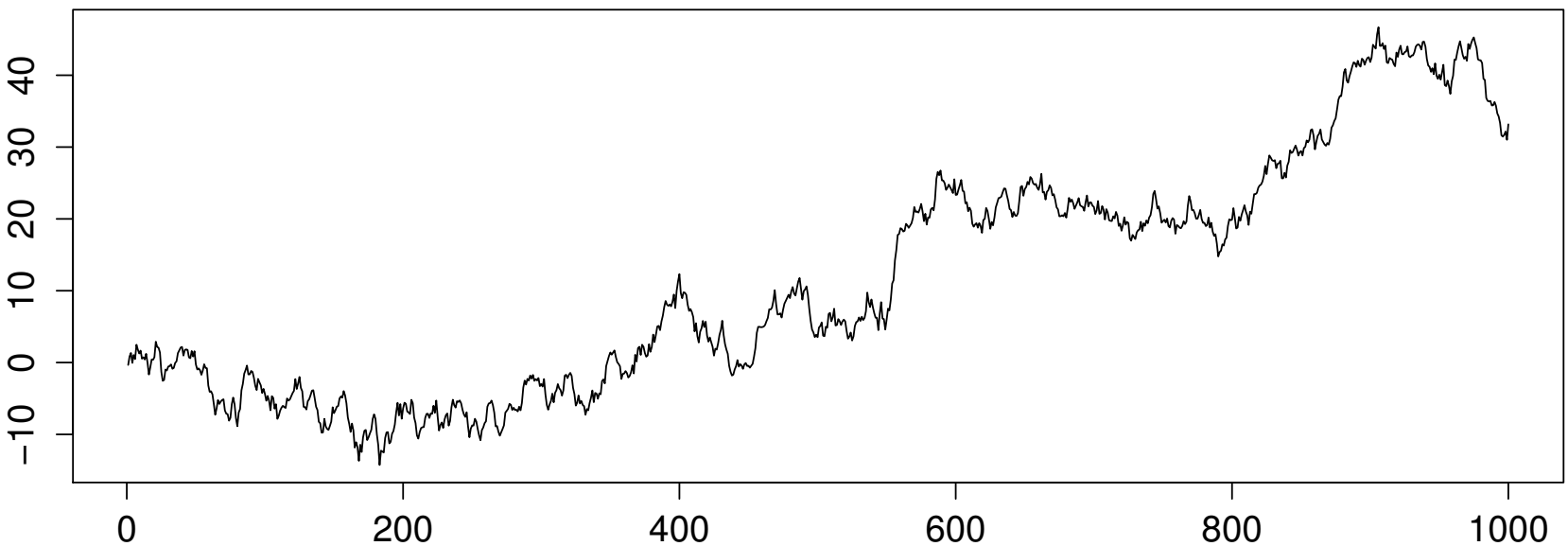


Figure : *Time series with a stochastic trend.*

Stationarity Testing

- A pure AR model of a time series with stochastic trend contains a unit root [3]
- Testing for the presence of a unit root can therefore be used to test for non-stationarity
- A unit-root test starts with the null hypothesis that an AR model has a unit root
- The alternative hypothesis is that an AR model of the time series does not have a unit root
- Next, a test statistic is measured
- If the p-value is below the chosen significance level, the null hypothesis is rejected
- Rejecting the null hypothesis provides reason to accept the alternative hypothesis
- The Augmented Dickey Fuller (ADF) test is commonly used for unit root testing

Stationarity Testing (cont'd)

- On the other hand is a stationarity test
- This test starts with the null hypothesis that a time series is stationary around a deterministic trend
- If the test statistic is above some significance level, this shows that the null hypothesis can be accepted
- Then the time series should be considered stationary
- The Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test can be applied for testing stationarity.

Modeling Methods

Time Series Modeling Methods

- Time series modeling methods 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
- To avoid having too many parameters relative to the number of observations, we use

$$p_{max} = \left\lfloor \frac{n}{mK_{min}} \right\rfloor \tag{1}$$

  - n is the number of time samples
  - m is the number of time series
  - K_{min} is the minimum acceptable ratio of observations to parameters

- 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 [2, p. 56]
  - 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 [1]
- AIC will be used, since the number of parameters is already limited in the specification step

Data Methods

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 basis for project selection is
  - Has been actively developed for at least several years
  - Has openly available issue tracking system data
  - Distinguishes between defects and other issue types

Selected Projects

MongoDB

- A NoSQL, document-oriented database
- The project has been actively developed since 2009
- The *core server* sub-project was chosen for modeling
- Uses JIRA for issue tracking
- Issue data was exported as XML

Selected Projects (cont'd)

Hibernate

- An object-relational mapping (ORM) framework
- The project has been actively developed since 2003
- The *orm* sub-project was chosen for modeling
- Uses JIRA for issue tracking
- Issue data was exported as XML

Selected Projects (cont'd)

NetBeans

- A software development platform, including an IDE
- The project has been actively developed as an open source project since 2000
- The *platform* and *java* sub-projects were chosen for modeling
- Uses Bugzilla for issue tracking
- Issue data was extracted from MySQL database dump

Data Cleansing

- Not all of the data was preserved for modeling
- No-change issues
  - Only issues with resolutions such as *fixed*, *complete*, or *done* were kept
  - Other issues did not result in any change and were not included
- Orphan sub-tasks
  - Issues that are sub-tasks are first converted to be the same type as the parent issue
  - Sub-tasks whose parent issue is not in the dataset are considered orphans, and discarded
  - Orphan sub-tasks can not be identified as improvement or new feature

Data Sampling

- The dataset was operated on to prepare it for time series modeling
- First, data was sampled at regular intervals, measuring
  - Number of bugs created
  - Number of improvements resolved
  - Number of new features resolved

Data Sampling (cont'd)

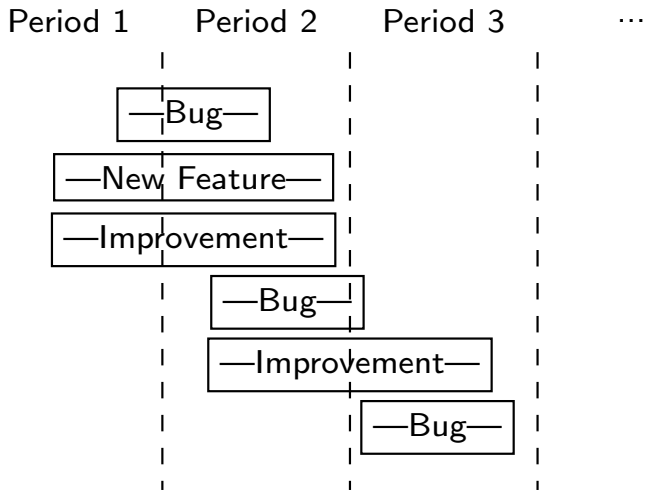


Figure : *Sampling example issue data.*

Period	Improvements Resolved	New Features Resolved	Bugs Created
1	0	0	1
2	1	1	1
3	1	0	1

Table : *Results of sampling example issues.*

Establishing Stationarity

- Establish stationarity by testing
  - ADF unit root test
  - KPSS stationariy test
- If test results agree, then no differencing is necessary
- Otherwise, difference data and retest

Time Windowing

- Assumption: the software development process underlying a given project might change over time
- The VARX model does not accommodate a changing process
- To account for a changing process, data will be time windowed
- Data will be used for modeling only if it occurs within the time window
- This should limit the amount of process change the model is exposed to

Time Windowing (cont'd)

To still cover the entire dataset, the modeling methods are applied as the window advances by one sample . This is termed a *sliding window*.

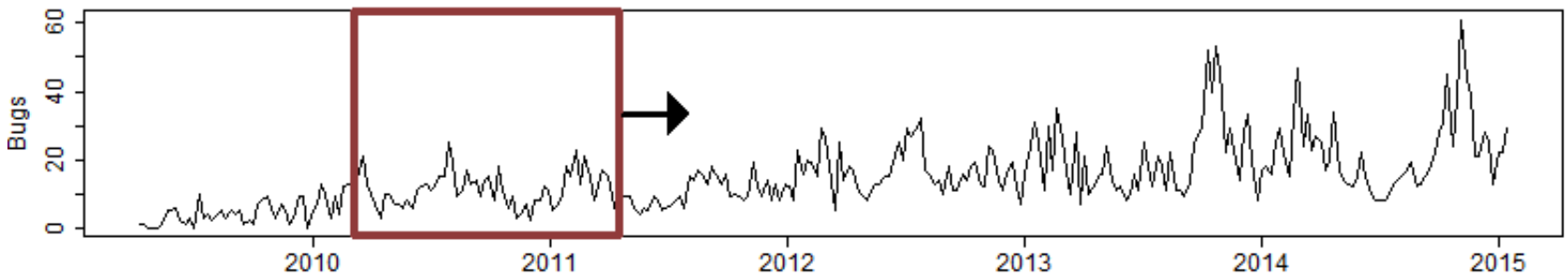


Figure : *Sliding time window moves over the entire time series.*

Results

Data collection results

Summary of the collected data

Project Product Name	Date Range	Initial Issue Count Count	Final Issue Count Count
MongoDB *core server*	Apr, 2009 - Jan, 2015	7,007	6,971
Hibernate *orm*	Apr, 2003 - Apr, 2015	14,262	8,278
NetBeans *platform*	Jan, 2001 - Jun, 2010	24,745	11,335
NetBeans *java*	Jan, 2001 - Jun, 2010	18,313	8,699

It is worth noting that none of the datasets contained many orphaned subtasks. The highest number found was 80 in the Hibernate *orm* dataset.

Sampling Results

Not knowing which sampling period would work best, sampling was performed for each of the following sampling periods: 7 days, 14 days, and 30 days.

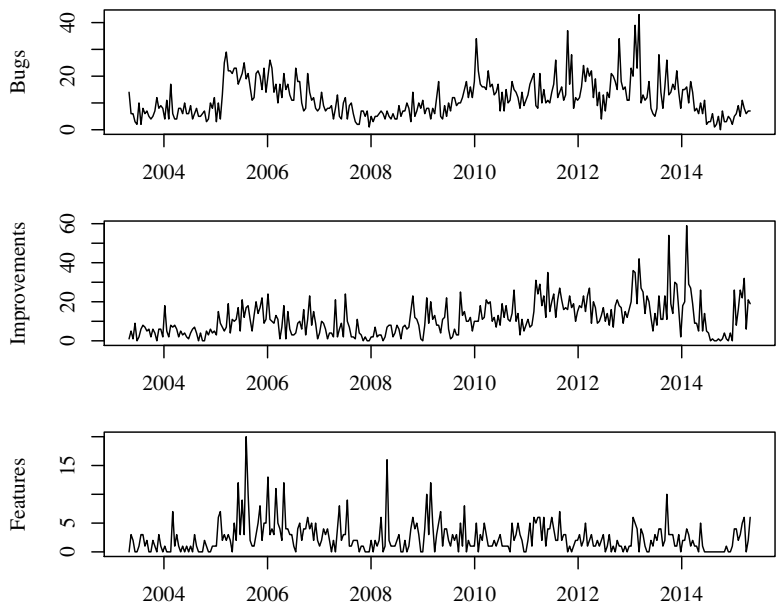


Figure : *Time series data from sampling the Hibernate orm dataset with a period of 14 days.*

Stationarity Testing & Differencing Results

- The time series were found almost without exception to be non-stationary.
- Differencing was found to remove nonstationarity
- Not knowing how differencing would affect model accuracy, data differencing of degrees of 0, 1, and 2 were made available for the modeling phase

Windowing Results

Not knowing which window size would work best for the sliding window, a range of window sizes were selected for each sampling period.

Table : *Sliding windows sizes used*

Sampling Period	Sliding Window Sizes
7 days	36, 39, 42, 45, 48, 51, 54, 57, 60, 63, 66, 69, 72, 75, 78
14 days	24, 27, 30, 33, 36, 39, 42, 45, 48, 51, 54
30 days	12, 15, 18, 21, 24, 27, 30, 33, 36

Exploratory Results

- Modeling methods were first applied using the sliding window, in an exploratory fashion
- Sampling period, window size, and degree of differencing are varied
- Results are compared by the following metrics
  - The none-valid proportion
  - The non-normal proportion
  - Root-mean-square error (RMSE), which is the standard deviation of the error distribution
  - The in-interval proportion
- First two metrics measure validity
- Last two metrics measure accuracy

Exploratory Results (cont'd)

- The only clear trend is that a higher degree of differencing results in lower model accuracy
- Other trends are not consistent for different sampling periods and across datasets.
- For a given dataset, the sampling period and window size can be chosen to maximize validity and accuracy.

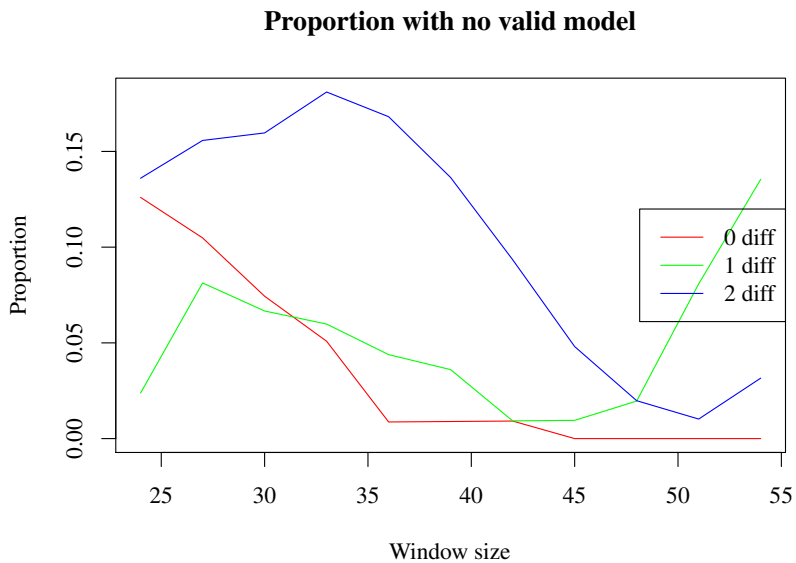


Figure : *An example of the inconsistent trends, from the MongoDB core server*

Exploratory Results (cont'd)

Sliding window parameter values were selected based on results from exploratory modeling.

Dataset	Degree of Differencing	Period	Window
MongoDB *core server*	1	14	24
Hibernate *orm*	1	30	24
NetBeans *platform*	1	14	27
NetBeans *java*	1	14	30

Final Results

- Final results were obtained by applying the sliding window approach to each dataset, using the parameter values from exploratory modeling.
- The results are compared by
  - The none-valid and non-normal proportions
  - The distribution of actual compared to the distribution of predicted number of bugs
  - The distribution of prediction errors, with scale and shape being described by RMSE and Q-Q plot, respectively
  - The in-interval proportion for a 75% or a 90% prediction interval

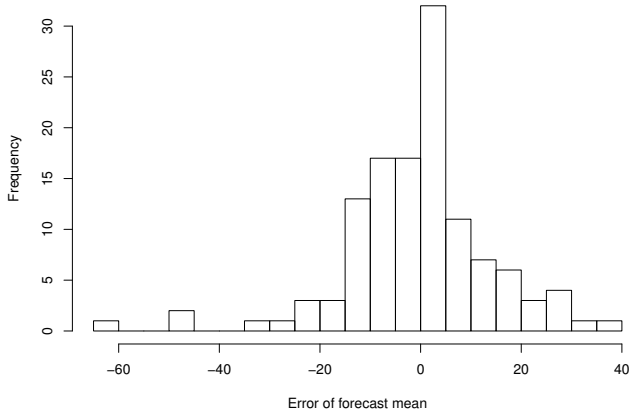
Final Results (cont'd)

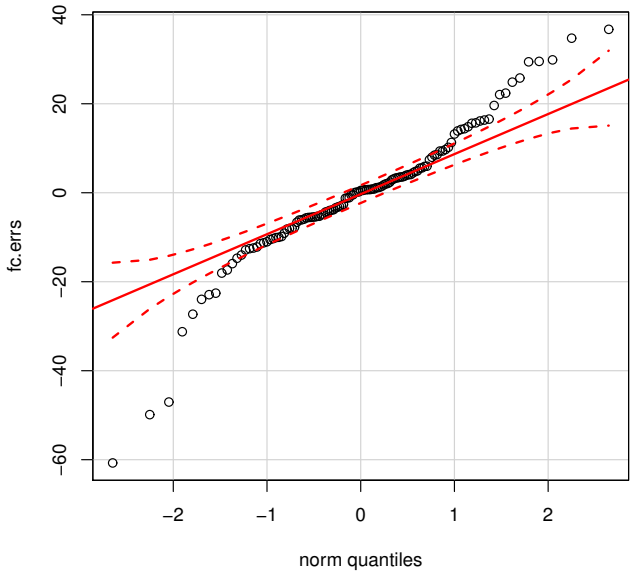
Summary of results

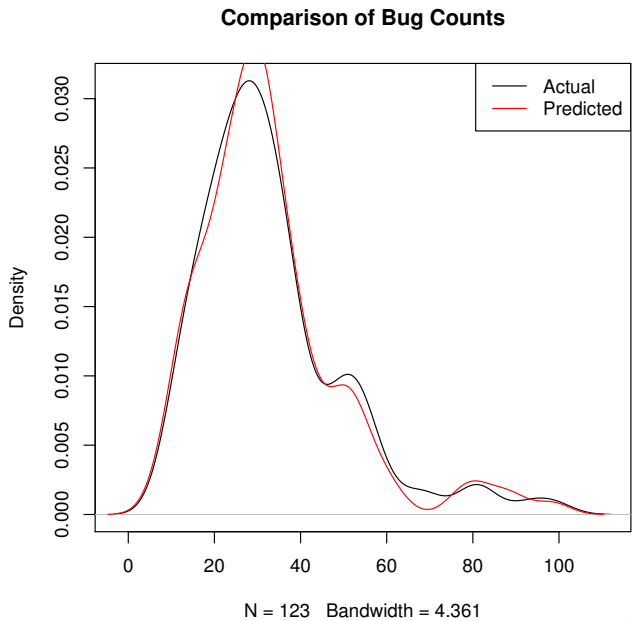
Dataset	Window Count	None-valid Proportion	Non-normal Proportion	RMSE	In 90% Interval	In 75% Interval
MongoDB *core server*	126	2.38%	0%	14.7230	36.59%	27.64%
Hibernate *orm*	121	4.13%	0.86%	10.2685	53.91%	45.22%
NetBeans *platform*	219	9.59%	2.53%	15.2702	46.11%	39.38%
NetBeans *java*	216	12.96%	14.89%	18.0469	43.13%	30.63%

Final Results (cont'd)

MongoDB *core server* dataset

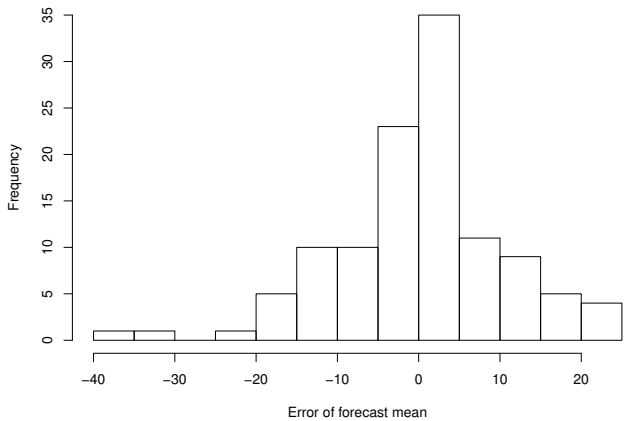


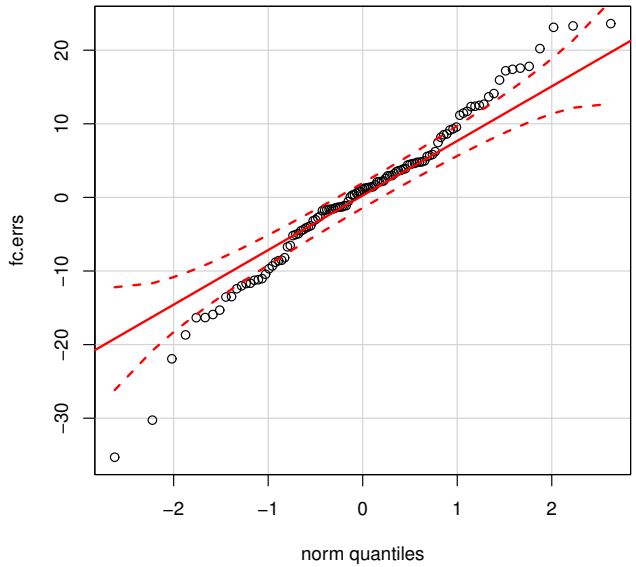


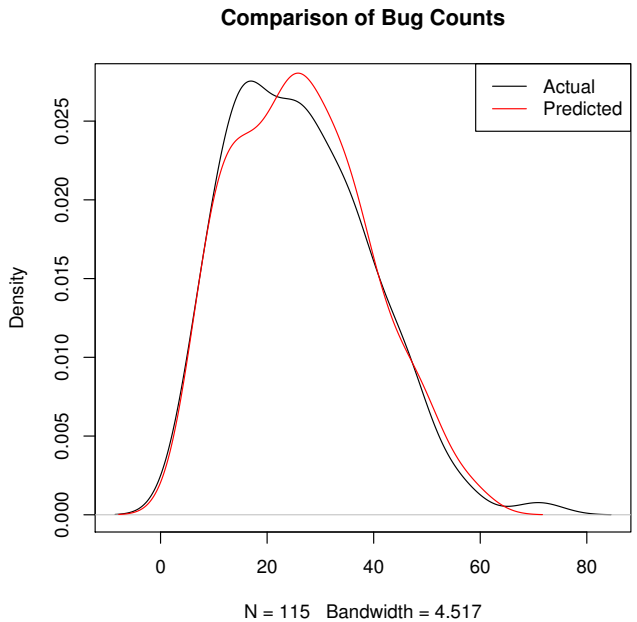


Final Results (cont'd)

Hibernate *orm* dataset

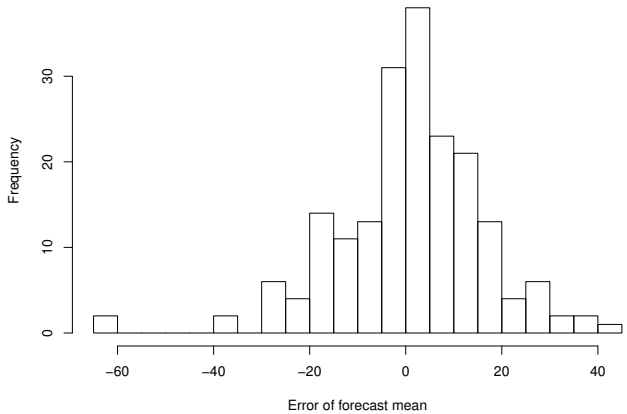


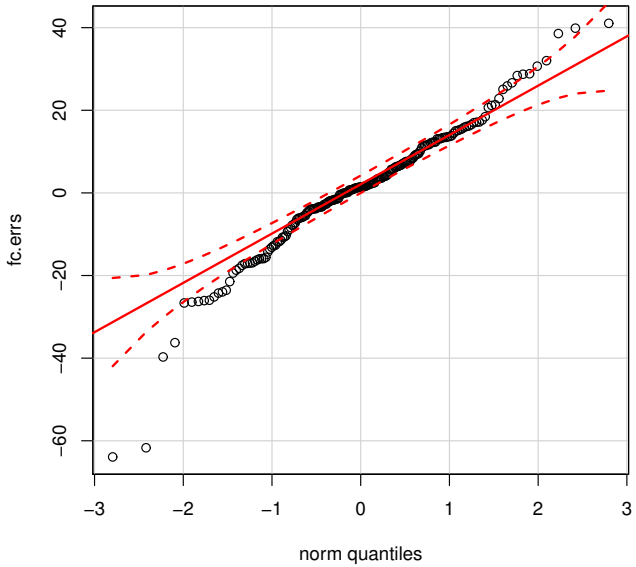


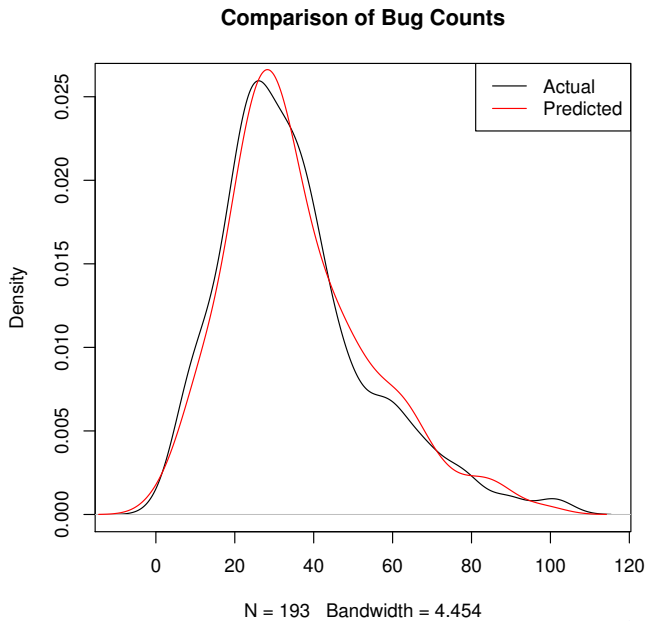


Final Results (cont'd)

NetBeans *platform* dataset

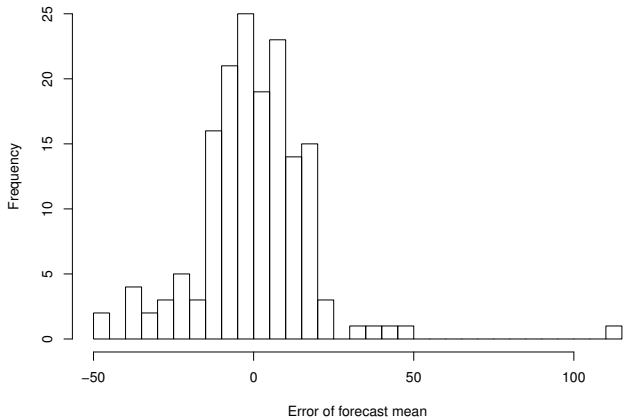


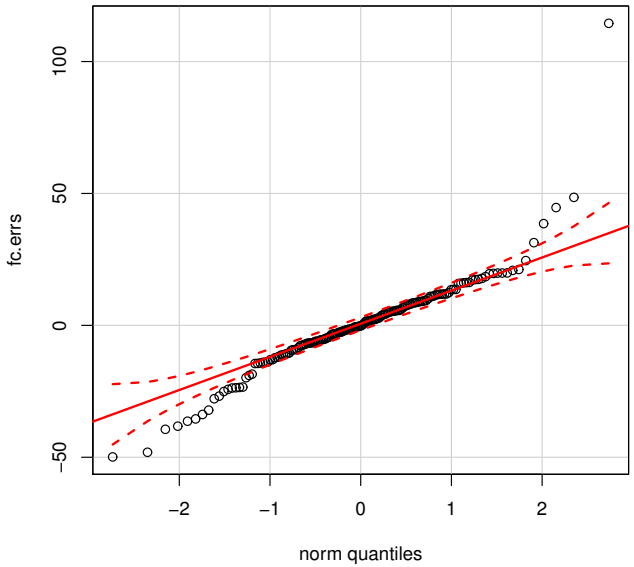


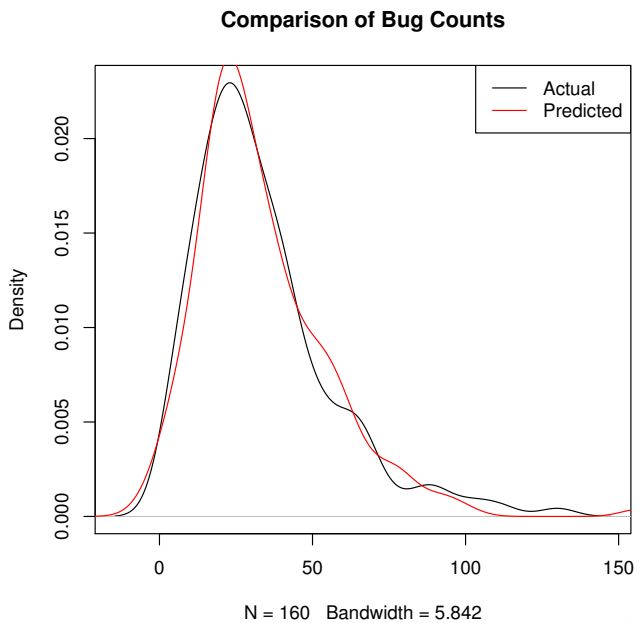


Final Results (cont'd)

NetBeans *java* dataset







Conclusion

Discussion

- Windowing was used to maximize validity and accuracy
- Validity
  - None-valid proportions were between 2% and 13%
  - Non-normal proportions were between 0% and 15%
  - Together, these proportions represent the risk that for any given sample window there will be no valid course for making a prediction
- Accuracy
  - in-interval proportions at a 90% prediction interval were between 36% and 54%
  - in-interval proportions at a 75% prediction interval were between 27% and 46%
  - RMSE was between 2.40 and 9.02 bugs per week
  - The best results were for the Hibernate *orm* dataset

Discussion (cont'd)

- The sliding window provides control over validity and accuracy
- It also conveys a picture of how a model can generally be expected to perform for any given window in the future

Modeling Results	Future Expectation
Low invalidity proportions	A valid model will likely be available
High in-interval proportion	Model prediction will likely not exceed prediction interval
Low RMSE	Model prediction error will likely be low

Future Work

Possible future research:

- Use change management data in a time series model
  - Problems with issue tracking system (ITS) data:
    - Subject to human error (unknown lag time before changes are entered)
    - Information is lost (change magnitude, location(s), author)
  - ITS data for changes made (new features, improvements) can be replaced with change management data
- Use birth-death process models
  - The issue tracking system data will always be non-negative, since it is count data
  - A birth-death process specifically models count data, being used for population size
  - Does not require sampling or differencing
  - Could directly model issue creation (birth) and resolution (death)

Thanks & Questions

- CS Faculty
  - Dr. Razvan Andonie
  - Dr. Filip Jagodzinski
  - Dr. John Anvik
- Math Faculty
  - Dr. Yvonne Chueh
  - Dr. Kathryn Temple

References I

S. Bisgaard and M. Kulahci.
Time series analysis and forecasting by example.
John Wiley & Sons, 2011.

G. E. P. Box, G. M. Jenkins, and G. C. Reinsel.
Time Series Analysis.
John Wiley, 2008.

P. H. Franses.
Time series models for business and economic forecasting.
Cambridge university press, 1998.