Fall 2014

Divide and recombine: Autoregressive models and STL+

Xiang Han
Purdue University

Graduate School Form 30
(Revised 08/14)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Xiang Han

Entitled
Divide and recombine: Autoregressive models and STL+

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

William S. Cleveland

Bowei Xi

Chuanhai Liu

Hao Zhang

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification/Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): William S. Cleveland
Bowei Xi

Approved by: Jun Xie 09/04/2014

Head of the Department Graduate Program Date

DIVIDE AND RECOMBINE: AUTOREGRESSIVE MODELS AND STL+

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Xiang Han

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2014

Purdue University

West Lafayette, Indiana

I dedicate my dissertation work to all of you, who support, help and advise me in all these years.

ACKNOWLEDGMENTS

I would never have been able to finish my dissertation without the guidance of my advisors and committee members, support from my family and help from friends.

First and foremost I offer my sincerest gratitude to my co-advisor, Dr. William S. Cleveland, who guided and suppported me with his wisdom and patience in all these years. His vision will always inspire me in my future career.

I would like to cordially thank my co-advisor, Dr. Bowei Xi. Her advice, especially on the dissertation writing, is indispensible. I will always remember her encouragment.

I would like to extend my appreciation to my Committee member, Dr. Chuanhai Liu and Dr. Hao Zhang. Dr. Liu offered new angles in the methodology for me. Dr. Zhang's advice to revise my thesis was greatly appreciated.

I am also indebted to Dr. Ryan Hafen, Dr. Jeff Li and Dr. Saptarshi Guha for their help in the discussions and lectures.

I am grateful to the love of my entire family. My father is my constant source of support and strength. My mother's spirit has always inspired me. My grandmother and late grandfather are my staunchest supporters. All my extended family has aided and encouraged me.

My dear friends assisted me in all the possible ways. I could not overcome all the setbacks without their support and care. I deeply value our friendship.

Finally I appreciate the financial support from Department of Statistics. I am thankful to the help of the graduate chair: Dr. Jun Xie.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

SYMBOLS

T_t	trend component at time t
S_t	seasonal component at time t
R_t	remainder component at time t
P_t	prediction at time t
E_t	prediction error at time t
R_j	remainder component in the Autoregressive model at time j
M_t	Media and Entertainment series
g_t	weight in STL+ at time t
w_t	trend window
w_s	seasonal window
d_t	trend degree
d_s	seasonal degree
i_i	number of inner iterations
i_o	number of outer iterations
α	coefficients of the autoregressive models
X_t	observations of the autoregressive models
ϵ_t	errors at t in the autoregressive models
π_i	coefficients of the long range dependent series
A	input time series
B	output time series
C	operator matrix
$\rho'(r_i)$	loss of the observation r_i in the Tukey's bisquare

ABBREVIATIONS

STL	Seasonal-Trend decomposition procedure based on Loess
BSM	Basic Structural Model
STM	Suitable Structural time series Models
ISP	Internet Service Provider
ICP	Internet Content Provider
DNS	Domain Name System
CIDR	Classless Inter-Domain Routing
IP	Internet Protocol
TCP	Transmission Control Protocol
FTP	File Transfer Protocol
LAN	local area network
WAN	wide area network
PDC	Primary Domain Controller
D&R	Divide-and-Recombine
RHIPE	R and Hadoop Integrated Programming Environment
HDFS	Hadoop Distributed File System
BSV	Betweeni-Subset Variables
WSV	Within-Subset Variables
fGn	fractional Gaussian noise
AR	AutoRegressive model
ARIMA	AutoRegressive Integrated Moving Average model
FARIMA	AutoRegressive Fractionally Integrated Moving Average model

ABSTRACT

Han, Xiang Ph.D., Purdue University, December 2014. Divide and Recombine: Autoregressive models and STL+ . Major Professor: William S Cleveland and Bowei Xi.

In this thesis multiple methods are proposed and applied to the Akamai CIDR time series data. The Akamai network is one of the world's largest distributed-computing platforms, with more than 250,000 servers in more than 80 countries. It is responsible for 15-20 percent of all web traffic. We obtained 110 GB raw CIDR data over a 18 month period, collected on the Akamai network from November 2011 to April 2013.

The Seasonal-Trend Decomposition procedure based on loess (STL+) is used to model the CIDR series. Motivated by the CIDR series analysis, we propose a general prediction based model selection procedure, where extensive visual diagnostics are part of the procedure for selecting the best performing model. Factorial experimental designs are used to explore the parameter space. We evaluate the performance of different models for the CIDR series using our proposed prediction based model selection procedure. Furthermore the analysis and modeling of the CIDR series is performed under the Divide and Recombine for large data framework. And we conduct a theoretical Divide and Recombine time series estimation study.

We also study the performance of Divide and Recombine estimates for Gaussian auto-regressive time series, Gaussian long range dependent series, and auto-regressive series with tails heavier than Gaussian.

1. INTRODUCTION

A time series [23, Hamilton, 1994] [35, Shumway et al, 2006] is a sequence of observations collected sequentially over time. It has intensive application in agriculture, business, economics, engineering, natural sciences and social sciences. The purpose of the time series analysis is to model the intrinsic mechanism behind the series and to predict the future values of the observations. The classic model building strategy introduced by Box and Jenkins [6, Box et al, 1970] can be executed in three steps: (1) model specification; (2) model fitting and (3) model diagnostics. Our stepwise model building approach also follows this strategy. We will illustrate the complete process in the modeling of Akamai CIDR time series data.

One of the general models used for a seasonally adjusted time series is decomposing the series into three comonents: seasonal component, trend component and the remainder/irregular component. [27, Macaulay, 1931] The seasonal component is defined as the cyclical variation that happens repeatedly from cycle to cycle. It can be either constant or in an evolving fashion. The trend component targets the variations in the series due to long term change of the behavior, i.e. trend. The irregular component is assumed to contain: (1) the white noise in the real world; (2) the sampling error and (3) the non-sampling error.

One official method for trend-seasonal decomposition: 'census X11' method [33, Shiskin et al, 1967] is adopted by international government agencies including US Bureau of the Census and Australian Bureau of Statistics. The procedure makes additive and multiplicative adjustments to the series and produces an output containing the aforementioned components. It also incorporates sliding spans analysis, which determines the suitability of the seasonal adjustment by its diagnostics.

Other methods adopted for the decomposition include: Seasonal-Trend decomposition procedure based on Loess(STL) [11, Cleveland et al, 1990], Basic Structural

Model(BSM) [28, Maravall et al, 1985] and Suitable Structural time series Models(STM) [9, Butter et al, 1990].

STL proposed by Cleveland [11, Cleveland et al, 1990] has desirable statistical and computational features compared to X11. Hafen [22, Hafen, 2010] contributed to its modeling and theoretical specifics in his STL+ work. However, there is still a lack of guideline in terms of model fitting of STL. The best parameter choice procedure is yet to formalize. In this dissertation, a data-driven prediction diagnostic based model fitting and selection method is proposed to regulate and normalize this long-awaited procedure.

In the Divide-and-Recombine framework(D&R) [20, Guha et al,2012], in order to estimate the coefficients in the comprehensive massive data(Big Data), the data are split into replicative subsets and estimates of the model coefficients are calculated in each subset then recombined into the final estimates: D&R estimates. Motivated by the different structure and characteristics of the data, various replicate division methods are put forward. It would be interesting to have a simulation study to test and compare the performances of these replicate division methods for the coefficient estimation in an Autoregressive time series model setup.

The roadmap to this dissertation The structure for the rest of this thesis is as following. Chapter 2 explains how STL modeling works. As for Chapter 3, the contributions of STL+ are discussed. In Chapter 4, the Akamai CIDR dataset is introduced. How we process the data and acquire the target time series is explained and general visual diagnostics are performed on the series. The clues from these plots allow us to understand the attributes of the dataset. A preliminary model is fitted and its model fitting is evaluated. In Chapter 5 we propose the guidelines to choose the best parameters set in the STL+ model fitting. The prediction series are constructed and the quantiles of their absolute prediction errors are used to illustrate the differences among the performances of multiple models in a factorial experimental design. Moreover, an Autoregressive order 2 model is built on the remainders of the candi-

date models to eliminate the dependence among the remainders. The final conclusion is based on the comprehensive consideration of the visual diagnostics and statistics. A simulation study for replicative division methods of D&R in the Autoregressive modeling is presented in Chapter 6. Their performances in various Autoregressive models are assessed in different sample sizes and subset sizes. Chapter 7 summarizes the examples and methodologies in Chapter 4, 5 and 6 and suggests possible future works.

2. STL FREQUENCY DECOMPOSITION

Seasonal-Trend decomposition procedure based on Loess (STL) [11, Cleveland et al, 1990] is a decomposition method for time series data analysis. Each observation in the time series is decomposed into three parts: the trend component, the seasonal component, and the remainder component. The long term change in the time series is captured by the trend component. A cyclical pattern is reflected in the seasonal component. The residuals, the remaining variation, are the remainder component.

Compared with a standard approach like X11, STL eliminates computational complication, provides robustness in the estimation, adds the missing value handling capacity and presents a scientific design as implanted in the inner loop of the algorithm.

The STL procedure is an iterative reweighted smoothing operation using loess ([10, Cleveland et al, 1988] [12, cleveland et al, 1991] [13, cleveland et al, 1992] [14, cleveland et al, 1996]) as the filter. Through iteration, the STL procedure separates the competing low frequency domain trend component and the higher frequency domain seasonal component.

Theoretically the iteration of the STL can be expressed like this: in each step of an iteration of STL procedure, the input time series A is transformed linearly into an output time series B. The filter, C, which is the matrix used in the transformation, is called operator matrix. So we can denote the transformation as: $B = A\% * \%C$. One iteration consists of several inner loops and one outer loop.

In each inner loop, the first step is detrending. For the very first iteration, the current trend component is initialized as 0. Then detrend the original time series using this current trend component by subtracting the trend 0 from the original series. The unchanged new series is now ready for the deseasonalization steps in the first inner loop.

The deseasonalization steps are designed to compute the seasonal component in the first inner loop and to be subtracted from the original series. To get the seasonal component in the first inner loop, firstly fit a loess curve in each of the cycle-subseries. Each cycle-subseries is a subseries made up of all the values in the same position of a seasonal cycle. For example, if a year is used as a seasonal cycle for a monthly data, all the observations for the same month would be in a subseries respectively. Thus there are 12 subseries in total. The loess smoothing values from all the cycle-subseries are recombined in one series in the natural order of the original series. That is the raw seasonal component in the first inner loop.

The raw seasonal component, however, has some trend that includes the remaining low intensity variation. So STL uses two moving average and one loess curve to capture the remaining trend in the raw seasonal component. Then after subtracting this remaining trend from the raw seasonal component, the official seasonal component in the first inner loop is obtained.

Then after subtracting this seasonal component from the original series there is the deseasonalized series. A loess procedure is fitted on this series using the trend window as the bandwidth for loess, which returns the official trend component in the first inner loop. This trend component will be used in the detrending step of the next inner loop.

This concludes a complete inner loop. In each iteration, multiple inner loops can be executed. Usually the trend and seasonal components converge very fast in a few inner loops. Then the remainder is calculated based on the trend and seasonal component acquired in the end of the inner loops.

Since the times series data could have outliers which drastically skewed the estimation of the model components, STL designs a robustness feature to eliminate the undue influence of these extreme values. The robustness feature is achieved in the form of robustness weights. The extremely large absolute values of the remainder of an observation indicates that this observation might be considered as an outlier. Thus it should be given smaller or zero weight.

The weight of each observation is proportional to the absolute value of the corresponding remainder and is calculated based on a bisquare function. This concludes the outer loop.

The robustness weights are multiplied by the weights of the loess procedure in the first iteration. The products are used as the operational weights of the loess procedure in the next iteration.

For most of the common time series, STL procedure usually takes a handful of iterations to converge.

The loess regression curve [10, Cleveland et al, 1988] [12, cleveland et al, 1991] is a non-parametric smoothing method for dependent variable y based on independent variable x. A locally fitted least squares polynomial with degree d=1(linear) or 2(quadratic) is applied to the fitting. The weight function is a tricube neighborhood weight function which assigns weights proportional or related to the distance between the x values of the points and what we want to estimate. A critical parameter q, which is the number of observations that would be involved in the weight funciton, is selected to decide the limits, in turn, essentially the smoothness of the local fit. A larger q would lead to a more smooth fit. The curve present in the data determines the degree d.

The STL algorithm applies two recursive cycles: the inner loops are nested inside an outer loop. Each pass of inner loop updates the trend and seasonal. Then after $n_{(i)}$ passes of inner loop, a new robustness weight will be calculated, which will be used for the inner loop(s) in the next outer loop. The combination of the inner loop(s) and weight caculation is a complete outer loop. The algorithm works in the following way:

There are six sub-steps in an inner loop. In each loop, given the trend and seasonal from last loop, the sub-steps are: (1). The series is detrended by deducting the trend from last loop. (2). In the output series from step (1), the subseries in the same position of the seasonal cycle are called cycle-subseries. In each cycle-subseries, a loess smoothed series is calculated then reordered as the original series. This is

the raw seasonal component. (3). The smoothed series is filtered by two moving averages and a loess smoothing. The results is a second catch of trend remained in the raw seasonal component. (4). The difference of the series in step (2) and (3) is computed as the official detrended smoothed seasonal component in this inner loop. (5). Subtract the official seasonal component in step (4) from the original series to get the new deseasonalized series. (6). The deseasonalized series in step 5 is smoothed by the loess trend filter. The output would be the trend component to be used in the next inner loop.

The robustness weight in each outer loop is based on how large the remainder is. A bisquare weight function is used to weed out the extreme values. The newly minted robustness weight will be multiplied to the neighborhood weight of loess in the next loop. A starting trend component value of 0 in the first loop is fine.

The cycle-subseries plot is graphed in which the midmeans of the values are the horizontal lines and the ranges of the values are the ends of the vertical lines. This is a standard visual diagnostic of the STL modeling.

A seasonal diagnostic plot is used in assistance of parameter selection especially for the seasonal window length. What is in the plot is the seasonal component plus the remainder component. Both are centered by the mean of the values in each cycle subseries.

Neither of these two plots uses a measure of fitness to select parameters.

To choose the suitable trend and seasonal window, the underlying objective is to avoid the trend and seasonal components competing each other for variations in the series. Through eigen value analysis, the condition that eigen values of seasonal and trend component should not both be nonnegligible is required to achieve the aforemetioned objective. The smallest value the seasonal window can take and still offers minimal data smoothing is 5. But the criterion by eigen value analysis gives the smallest possible value for the seasonal window to be 7. The lower bound of trend window is deduced by a function of seasonal cycle length and seasonal window. Since

seasonal window is at least 7, the rule of thumb for the lower bound of trend window is in the range of 1.5 to 2 times seasonal cycle length.

The STL procedure offers speedy computation, robust estimation and the capability of handling missing values. Compared with X-11 stardard time series decomposition method, STL enjoys faster speed and missing value solution. Meanwhile, the options to customize the parameters in STL modeling maximize the performance of acquiring the appropriate amount of smoothing in the trend and seasonal respectively.

3. STL+: STL WITH ADDED FEATURES

The STL+ [22, Hafen,2010] contributed in the following areas:

1) a lower bound proposed for the bandwidths in the local quadratic smoothing;

2) at end points blending with lower degree polynomials to tackle the end point issue;

3) software implementation with the capability to handle missing value and statistical inference, neither of which is available in the stl R library.

The original STL literatures discussed how to compute the lower bounds of the key parameters, i.e., trend window, seasonal window and low pass filter window for the local linear fitting. STL+ extended it into local quadractic fitting and established the theoritical lower bound for these windows using a similar approach. The shared goal of the general guidelines for parameters is to steer clear of the competition between the trend and seasonal components for variation in the data.

Trend and seasonal components are both computed utilizing loess method. Using the chosen degrees, which can be set to 0, 1, or 2 and set to different values for trend and seasonal components, the STL and STL+ literatures established the lower bounds for the trend window and seasonal window based on a theoretical and empirical study. STL+ focused on the lower bound for degree 2, local quadratic fitting. STL+ also increased the range of critical frequency to [0.05, 0.2] in increments of .03. In addition, STL+ suggested that a good rule to select high-pass filter window is to set it equal to the trend window.

STL+ also dabbed in the areas of endpoints problem. The points in the start or end position of the series do not get the same smoothing with symmetric weight as the points in the middle. That is why variance increases at the end points. The estimation of the end points is updated when new data points are available. Without new data points, STL+ proposed an approach: 'blending to lower degree polynomials'.

There are other solutions such as using reproducing kernel hilber space(RKHS) [5, Bianconcini, 2007] to address the end points estimation problem.

The RKHS method uses a second order tricube kernel for local linear loess fitting and a third order tricube kernel for local quadratic loess fitting. The weights for the tricube kernel are the same for the center points as the original loess fitting, and different for the end points. End points face the asymmetric smoothing scenario. Smaller and smaller neighborhood of the endpoints is used in loess smoothing when approaching the ends of the series. For the last point or the first point of a series, only $q/2$ observations are used in loess smoothing instead of q observations for the points in the middle. This method has difficulty in incorporating the robustness weights in the STL outer loop in the STL procedure.

The approach: 'blending to low degree polynomial at the end points' essentially results in a fractional degree by averaging the outputs of loess smoothing with different degrees for the end points.

The 'blending' means the final estimation of trend or seasoanl components at the end points is calculated as a weighted mean of the estimation based on loess smoothing given degree 1 or 2, and the estimation of loess smoothing given degree 0.

The degree 0 smoothing has nearly constant variance. So averaging degree 1 smoothing with degree 0 smoothing reduces the variance at the end points. This is the initial motivation of the method.

Both empirical (several case studies) and theoritical (based on power transfer function) support the idea to blend to 0, no matter whether the original degree is 2 or 1. Always use degree 0 smoothing to blend in the smoothing for the endpoints. If the original degree is 0, there is no need for blending.

The blending method can be considered as a shrinkage estimator which suffers from a slightly larger bias but is compensated by significantly reduced mean squared error for the endpoints estimation. It is a rewarding trade-off.

How much of the degree 0 polynomial is blended into the final estimation is controlled by the weight. The middle points in the series are able to fit symmetric loess

smoothing so there is no need to blend. A gradual increasing weight is given to degree 0 smoothing in blending for the points closer to the exact two ends. Since the loess fitting is more asymmetric towards the two ends, larger weight for degree 0 smoothing is required.

The values of the weight for blending have to be decided on a data-driven approach. Mean squared revision error is the proposed measure in STL+ literature. It is minimized for the final best set of blending weights. The revision error is the difference between the original symmetric estimation for each of the interior points and an asymmetric estimation using the blending when we pretend that they are end points.

Compared with the RKHS approach, the blending method prevails with smaller phase shift effect in terms of spectral properties. It also handily beats RKHS approach in mean squared revision error in real data assesement.

The STL+ is implemented in two packages 'stl2' and 'operator'. The 'operator' R library provides statistical inferences for the STL procedure. Furthermore the software package 'stl2' written in R language realizes the feature of handling missing values.

4. DATA

4.1 Internet Networking

A computer network [18, Gallo, 1999] is a telecommunication network that facilitates exchanging data among a collection of computers and other devices. The devices in the network pass data to each other over physical media through network connections by applying common networking protocols.

The computers and devices include any entities that linked to a network. They are either hosts such as computers, terminals and printers or networking related hardware such as routers and bridges. Here 'devices' is a generic term, the same as 'nodes', which is different from 'computers'. Computers are devices with their own operating system(e.g. windows or linux) while devices do not necessarily have the operating system.

The media is the physical environment that connects nodes. It has two broad types: cable and wireless. The examples of cable media are fiber-optic or twisted-pair. Wireless media includes radio waves(like Wi-Fi) and infrared radiation.

The communications among devices in the network obey certain rules called protocols, which specify how data are to be formatted, transmitted, routed and received between nodes. Internet, the largest computer network, is based on Internet Protocol suite. A lot of protocols at link, Internet, transport and application layers of the Internet are all parts of the suite. Transmission Control Protocol/Internet Protocol (TCP/IP), the most famous one, is the main protocol in the suite. [36, Taylor, 1998] Other protocols, like File Transfer Protocol(FTP) and Domain Name Service(DNS), all regulate specific applications.

The Internet as we know is a collection of networks. The Local Area Networks(LANs) and Wide Area Networks(WANs) built separately in the history are interconnected and combined globally into the form of Internet.

Most network applications used by end users are based on a client/server model. The end users are the people who actually use the the finished mature online applications. The client side provides end users an interface which is used to acquire certain services from the network. The server side is responsible for providing these services transparent to the end users. Both client and server can refer to either the web programs or the physical devices.

The devices or nodes are identified in the Internet using Internet addresses. An Internet address serves two major functions: host or network identification and location addressing. There are two versions of address in use: IPv4 and IPv6. IPv4 is the fourth version of the Internet Protocol and routes most traffic on the Internet. [4, BGP, 2013] It uses 32-bit address which has theoretical space of 2^{32} addresses. For example, the Purdue University website: www.purdue.edu has the IPv4 address 128.210.7.199.

To be transmitted among the devices in the Internet, data are partitioned into standard size (such as a few hundred bytes) datagrams called packets. A packet has two componenets: a header and a payload. The header contains routing information such as the source and destination IP addresses. The payload is the actual partitioned data to be sent. So the data encapsulation is the process of cutting a large file into pieces with standard size and putting each piece into a packet.

The packets are sent one at a time from the source node to the destination node at anytime. The network hardware delivers the packets via the virtual circuit to the destination which resembles the packets back into the original data. The packets do not necessarily follow the same traffic route to the destination. And they could arrive at the destination in any order. Packet is the smallest unit of data transformation in a network.

In a generic meaning, a ‘connection’ is just the linkage between two nodes: source and destination. The source node transfers data to the destination node through

a path or a route. The nodes directly connected to a node are its adjacent nodes. The source node passes information to its adjacent node and then the adjacent node passes on to its adjacent, so on so forth until the packets reach their destination.

Unlike connection-oriented link like telephone system, Internet links are connectionless. There are no physical exclusive connections between source and destination nodes. Nodes in a network share a communication channel through a virtual circuit. So all the packets with all kinds of destinations and sources can transmit through same link, share some of the routes and then part ways. All these traffic routing details are transparent to the end users.

To transfer datagrams among networks, routers are used to connect two networks to each other. When a packet jumps through a router en route to another network, it is called a hop. When too many packets are present at a router, the deteriorated performance and low speed we observe are called congestions.

Routing is the process of delivery of packets between nodes using best routes in the Internet. There are specific routing metrics to determine the best 'lowest cost' route. For example, the number of router 'hops', distance and bandwidth are considered in a metric. Various routing algorithms are utilized to serve corresponding metrics.

The routes and their metrics are saved in a routing table. When a router receive a packet, it reads the packet header to find the destination address and identify the destination network. Then it searches its routing table and applies the knowledge from the table to forward the packet.

Internet Service Providers(ISPs) are organizations offering Internet access service to both the general public and the business community. Their services include Internet access, Internet transit, domain name registration, web hosting and colocation. The typical ISPs are AT&T(Internet access), AOL.com(mail service) and Godaddy.com(domain registration).

Internet Content Providers(ICPs) are websites or organizations distributing content online. The typical ICPs are: Yahoo(news), Netflix(entertainment) and Google(search engine).

Domain Name System(DNS) is a hierarchical distributed naming system. Its most prominent service is translating domain names to IP addresses. It works as a phone book. The domain names typed in the web explorer are interpreted as IPv4 or IPv6 addresses, based on which the connection to the ICP and its service is constructed. The DNS assigns its functionality to different levels of servers. The root DNS are responsible for the main domain and the other name servers are delegated authority over subdomains. The domain information in the DNS servers is updated dynamically.

4.2 Data Provider Akamai

Akamai Technologies, Inc. is a Cambridge, MA based corporation specializes in Internet content delivery. [1, Akamai.com, 2014] Its network is one of the world's largest distributed-computing platforms which is responsible for 15-20 percent of all web traffic. [30, Nygren et al, 2010] Content delivery is a service offered by content delivery networks to serve content to Internet end users with high availability and high performance. The possible content includes online shopping, music downloading, web video and online gaming. Since the Internet is designed as a best effort structure, it does not guarantee the reliability or performance of the content delivery. Issues like latency, packet loss, web outages could all lead to unhealthy surfing experience for end users. The content delivery service offered by Akamai works like this: the content is offloaded directly from the content provider's original infrastructure to distributed Akamai servers and then delivered by them to geographically close Internet end users. Thus the end users actually acquire the content from the Akamai servers close to them not the servers of the ICP directly. This application vastly increases the service speed and decreases the entire load of the Internet since most of the copies of the content do not have to travel long range from the ICP servers to end users and hence do not take up too much bandwidth of the Internet for the same content.

The content delivery network also can be viewed as a virtual network which builts entirely on network provider's software and requires no enterprise client software and

makes no change to the Internet hardware. The main components of the Akamai content delivery network are: [1, Akamai.com, 2014]

(1). edge server assigned by DNS service to and close to the end user.

(2). transport system which is in charge of downloading the required data from the original enterprise client server.

(3). conmmunication, control and managment system that provides the enterprise clients to supervise and control the interaction between them and the end users.

(4). data collection and analysis system responsible for reporting and monitering the logs and server information of the content delivery.

The Akamai Network deploys more than 250,000 servers distributed in more than 80 countries. These servers are strategically located to mass population to shorten the routing of data transfer. The distributed cloud based system provides the relay service. Akamai rents out capacity on these servers to clients who want to expedite their websites service by serving content from locations closer to their end users. Their largest clients includes Facebook, Bing, Twitter and healthcare.gov. Through renting the content delivery capacity(servers) from Akamai, the clients provide superior online experience to their end users and potentially save infrastructure costs since they could share the capacity across the world with other clients.

The Akamai virtual network is treated by their enterprise clients as a natural extension of their own network. They keep the control and visibility of their content over the web. Management of content updates and completeness, multi-level control over various types and maintaining the grip of security issues are the all-in-one service in the package presented by Akamai content delivery system.

Its service relies on the self designed software platform implemented on the servers enhanced with state-of-the-art applied mathematical algorithms and computer networking technologies to improve performance. They boost end user experience, the efficiency of the data transfer and the utility of the servers and networks. At the meantime, they maximize availability, i.e., the speed of content loading and delivery to the end user. In addition, they minimize congestion, i.e., online traffic overload and

service outage induced, which can cost millions of dollars of lost revenue. They also clear security issues of the Internet traffic, i.e., the loss of data integrity, the breach of confidentiality of the content or the potential attacks on the service like steals of financial information.

Since these servers are distributed in more than 2000 networks all over the world monitoring the Internet in real time, the information they gather draws a vivid dynamic map of the on-time traffic status and trouble or hot spots of the entire Internet. The intelligence can be used to optimize the routing and data replication process to dynamically advance the Internet operation.

4.3 Data Content

CIDR stands for Classless Inter-Domain Routing, which is a standard for IP addresses allocation and IP packets routing. IP addresses are allocated to Internet service providers as well as end users in order to help indentify different host interfaces. The IP packets routing is performed by all hosts and routers to transport packets across networks.

Classless network was designed to replace the classful network. An IP address consists of two groups of bits in the address: the network address and host identifier. In the classful network design of the IPv4 address, the network address is one or more 8-bit groups of the 32-bit address, resulting in the net blocks of Class A,B and C address. So a Class A address has an 8-bit network address and a 24-bit host identifier. Class B has 16-bit and 16-bit in the first and second group respectively. Class C has 24-bit in the first group and 8-bit in the second. In consequence, there are 2^{24}, 2^{16} and 2^{8} addresses in each network of Class A, B and C respectively.

The Classless Inter-Domain Routing allocates address space on any address bit boundary rather than only on 8-bit segments. This means the network addresses of CIDR netblocks take first k-bit, where k is not necessarily 8, 16 or 24. Thus Internet

service providers and end users are no longer classified only as class A, B or C. This is the meaning of 'classless'.

On the Internet, physically a domain is a network of grouped devices, such as personal computers used to surf the Internet or servers used to host a website. Virtually a domain is also an identifier of the web service hoster. For example, in www.stat.purdue.edu/People, 'purdue' is a domain and 'People' is a directory.

Primary Domain Controller(PDC) are computers responsible for the management of the data flow and interaction of these domains. Inter-domain is the practice of the flow control by PDC.

So in a sense, CIDR is a bitwise standard for IP addresses notation. Blocks of IP addresses are grouped into entries in a simple routing table. These groups are CIDR netblocks sharing the same binary sequence as the subnet mask, i.e. the network address. For IPv4 addresses, the syntax looks like this: firstly a dotted-decimal address, followed by a slash, ended with a number ranging from 0 to 32. For example: 135.233.18.0/24. The first portion is the IPv4 address of a server in the subnet. The last portion is the length of the shared initial bits. The 24 in the example means they share 24 bits in address and this subnet has $2^{(32-24)} = 256$ hosts. The IPv4 addresses in this netblock are from 135.233.18.0 to 135.233.18.255.

Akamai CIDR data are the records of traffic metrics of different netblocks and are collected by Akamai over its own network. In the dataset, the number of hits and bytes of inquiries are aggregated within netblocks, service categories such as Media and Entertainment, Education etc., and localities, such as continent and countries. A hit is an online request sent to a server for a file such as a webpage, an image or a javascript. If a webpage has more than one file to download for the user, the user would generate more than one hit when he opens that webpage. So the number of hits is not a direct popularity measure of a web service. But it can be used to assess the traffic load. All the personal identifiable information was removed in this collection to protect privacy.

This Akamai data collection is not comprehensive and only concentrates in the activities from a selected group of /27 netblocks, such as:

"208.44.247.96/27", "219.143.240.64/27", "220.151.63.64/27".

The collection was conducted on an hourly basis over 18 months from November 2011 to April 2013. The first six raw CIDR data observations, in a R data frame, are showed below:

```
        cidr      start        hits        bytes                 cat
1 0.0.0.0/27 2012-02-01 11030518427 3.750790e+14 No content category
2 0.0.0.0/27 2012-02-01    28617441 9.945587e+11          Automotive
3 0.0.0.0/27 2012-02-01    14527286 4.571803e+10      Consumer Goods
4 0.0.0.0/27 2012-02-01    13805133 2.028242e+11   Consumer Services
5 0.0.0.0/27 2012-02-01      782446 1.171407e+10   Consumer Services
6 0.0.0.0/27 2012-02-01     3528478 2.366123e+11           Education
                   subcat country continent hour.of.day day.of.week
1 No content sub-category unknown   unknown           0         Wed
2      AU - Manufacturing unknown   unknown           0         Wed
3 CG - Perfume/ Cosmetics unknown   unknown           0         Wed
4    CS - Beauty & Health unknown   unknown           0         Wed
5       CS - Food Service unknown   unknown           0         Wed
6                         unknown   unknown           0         Wed
  week.of.year
1.           5
2.           5
3.           5
4.           5
5.           5
6.           5
```

The first field is the netblock. Not all the /27 netblock traffic was captured in this CIDR data. Packets were captured using tools such as tcpdump, which recorded only 6486 /27 CIDR netblocks. If a /27 netblock is captured, we have its specific address like '208.44.247.96/27'. The 0.0.0.0/27 records include all the /27 activities not in the captured netblocks, as displayed above. Hence 0.0.0.0/27 is used as a pseudo CIDR netblock. Total hits of captured CIDR are only about 1/1000 of the total 0.0.0.0/27 hits.

The second column is the date of that hourly record. The last three columns: hour of the day, day of the week and week of the year, if combined together, can show the exact time of that hourly record.

The third column is the total number of hits within the hour for a certain CIDR, in one category, one subcategory and country . Then the forth column is the total number of bytes of inquiries in that hour.

The fifth column is the service category. Categories are determined by the type of the content. For example, hits on purdue.edu belong to Education. If a record's service category can not be decided, it would be put into 'No Content Category'. The category configurations are static, which did not change in the record time period of this data set.

The sixth column is the service sub-category, which further describes the type of service and is nested in the corresponding category.

The seventh and eighth column are country and continent. When observations are from 0.0.0.0/27 netblock, no such information is given.

4.4 Data Structure

The raw data we obtained are about 110 GB. It is too big to load the entire dataset into the memory and analyze it in one process. The situation calls for a big data processing methodology and framework to preserve the entirety of the data and

extract useful information for specific tasks. Therefore we use the R software package RHIPE and apply MapReduce algorithm to process the CIDR data.

The MapReduce programming model [26, Lmmel, 2008] [15, Czajkowski et al, 2011] uses parallel and distributed computing algorithm to compute over large data sets saved on the cluster with a lot of nodes in an efficient and reliable manner, which is ideal for our dataset.

The general concept of Map and Reduce has a long history in functional programming within the computer science community. [16, Dean et al, 2004] Map refers to a higher-order function that applies a given function to the elements of a list and returns a list of results. Reduce, also known as fold, is a higher-order function that analyzes a data structure and combines the results through a given combining operation with a return value.

MapReduce comprises of two procedures: Map and Reduce. Map applies certain funcitons to the data in small chunks with intermediate values as output. Then the output of the map process is sorted then fed as the input of the reduce process, which is used to further process the data to some aggregated results.

Hadoop is the framework proposed by Yahoo and Google to implement the MapReduce scheme. [38, White, 2011] It comprises of Hadoop MapReduce(the programming model), Hadoop Common(the libraries collection), Hadoop Distributed File System(HDFS, the distributed file system) and Hadoop YARN(the resource management system). Hadoop open source code is written in JAVA.

HDFS is a fault-tolerant data storage system. Instead of being saved in one chunk, data are cut into small pieces with standard size and saved in different nodes of the cluster. Each piece of original data has three duplicate copies distributed over multiple servers and disks so when some nodes fail the entire cluster could still funciton. The computing nodes and storage nodes are usually the same. And the map process and reduce process are run individually on each working nodes.

In order to control the jobs on the nodes, the MapReduce framework has a single master JobTracker and one slave TaskTracker for each node in the cluster. The

master JobTracker makes the schedule for the tasks in the job, delegates the tasks to the TaskTracker slaves and combines the reported results from the slaves. The slaves are responsible for the calculation arranged by the master JobTracker and the reporting after that.

In map step, the master node gets the input and then separates it into smaller sub-problems. These problems are assigned and transfered to each worker nodes. Then the answers of the sub-problems generated on the worker nodes are sent back to the master node.

In the reduce step, the answers to the sub-problems are reorganized on the master node and processed in the worker nodes in a way to present the final answer to the original question. The combined answers are supposed to be(not necessarily) leaner in size than the answers to sub-problems hence the name 'reduce'.

The data are organized, saved and processed in the MapReduce framework in the form of key-value pairs. Input and output of the map and reduce process are always in a set of $< key, value >$ pairs. Key is an identifier for items to be processed. Value is what is identified by the key in the dataset.

We can further describe the MapReduce as a five-step algorithm:

(1). Map Input preparation: the input key(Key 1, the map key) value pairs and all the related input data are distributed to the worker nodes in the cluster by the master.

(2). Map code running: the user designated map tasks are run on each node and outputs are passed back to the master.

(3). Map output and Reduce input processing: the map output is shuffled in the system and distributed to each processor using the reduce key(Key 2) value pairs as the input of the reduce process.

(4). Reduce code running: the user reduce code is run on the nodes.

(5). Final results generation: the MapReduce system combines all the reduce output reported by each processor and produces the final results.

Use word count as an example. The job is to calculate the number of appearances of each word in a large document. In step 1, the document is cut into large number of pieces and sent to the nodes. Each node gets more than one piece. If some nodes finish faster they would get more. In the text file of this document on the HDFS, each line represents a $< key, value >$ pair. Each line is one map key(Key 1). Its corresponding map value is all the words in that line. In step 2, each node is responsible for running the map code on the key value pairs they get. In the map tasks, the words in the values are separated and each word becomes a new key and its value is set as 1 for easier summation later. In step 3, each reduce $< key, value >$ pair, e.g., $<$ '$Word'$, $1 >$ is put together and shuffled that the pairs with the same key are grouped together and redistributed in the nodes. In step 4, the nodes calculate the count for each word by simply getting a summation of the 1's(value) for each word(key) in their memory. In step 5, the $< word, count >$ pairs are reported to the master node and the counts from each nodes are aggregated for the same word again. The final results are a bunch of $< word, count >$ pairs exhausting each word in the document on the HDFS.

RHIPE is developed by Saptashi Guha. [34, RHIPE, 2011] It builts up a transparent interface between R and Hadoop. There are three components: the interface in R, the bridge from Java to R and the engine scripted in C.

The RHIPE users interact with HDFS, i.e. read and write data in HDFS through RHIPE. RHIPE takes care of the housekeeping issues with the MapReduce jobs. Its user just needs to write R code to perform the MapReduce algorithm and set up the related MapReduce parameters with RHIPE. The MapReduce jobs are submitted, monitored and controlled in R console while run in the cluster. Since it is implemented in R, all the results from RHIPE can be loaded as R objects thus are ready for further statistical functions performed directly on them.

In the backstage, RHIPE manages the tasks. It splits the user submitted job into small tasks and assigns a core in the cluster to do the computation for a task. Usually the number of tasks are far more than that of cores. Once a core finishes its current

task, RHIPE would assign one of the remaining tasks to it and so on and so forth until all the tasks are finished. Then the results are combined and saved in the HDFS and reported by RHIPE in the R console to the user. In a word, RHIPE and Hadoop brings the cores in the cluster to the data and the focus is on the data rather than the other way around.

The RHIPE-Hadoop environment in which we are working has two sets of servers. The first set runs R and RHIPE. The second runs RHIPE and Hadoop. We remotely log into the first set and operate in the R console with library RHIPE loaded. But the servers which actually carry the jobs we submit are the second set and the outputs are saved in the HDFS within it.

After some transformation the raw data provided by Akamai were loaded into HDFS as a list of lists showed in R. On the top level, they are just identifiers in RHIPE. Because there are storage limits in one RHIPE data frame (the maximum is 10,000 rows), the observations are separated using different keys. For each key value pair, it is a list of 2, which looks like this:

```
:List of 2
..$ : chr "97a5df377932e969ba5dde8463bc3f13"
..$ :'data.frame':    10000 obs. of  11 variables
```

The first list is an automatically produced RHIPE key. The second is a data frame containing the actual records, which is treated as the value for that key in RHIPE. It has 10,000, the maximum number of rows a dataframe can hold in RHIPE. The rows in the dataframe are similar to the example we gave in the data content section. Examples of the observations in the data frame:

```
                cidr      start hits    bytes                    cat
1000 12.148.18.192/27 2012-02-01  163  2453191 Media & Entertainment
1001 12.148.18.192/27 2012-02-01  693 13554528 Media & Entertainment
1002 12.148.18.192/27 2012-02-01   33   471569 Media & Entertainment
```

```
1003 12.148.18.192/27 2012-02-01   96   942772 Media & Entertainment
                           subcat       country     continent hod dow woy
1000 ME - Advertising Technology United States North America   0 Wed   5
1001   ME - Broadcast (TV Cable) United States North America   0 Wed   5
1002          ME - Portal/Search United States North America   0 Wed   5
1003             ME - Publishing United States North America   0 Wed   5
```

The observations in the dataframe are ordered by the categorical varaibles: CIDR, timing(hour of the day, day of the week, and week of the year), category and subcategory, each nested within the previous variables. Basically the data frame puts observations of the same CIDR together, inside of which is the same hour together, then inside of which is the same category and subcategory together. So the first changing categorical variable in the dataframe is the subcategory. Category is the second. So on and so forth.

4.5 Time Series Data

From this raw data, we can extract time series of number of hits or bytes of inquiries using the other variables like 'cidr' or 'content category' as the conditional variable.

Our series are all 78 weeks of time series data after further aggregation and each observation is the number of hourly hits. There are five types of time series data after data processing:

(1) Total hourly hits of Aggregated Individual CIDRs, i.e. $cidr! = 0.0.0.0/27$. It is aggregated over all the captured CIDRs on the /27 netblocks. It has 1 series;

(2) Hourly hits of Individual CIDRs aggregated by category. They are nested in the first type of series and one series for each of the 18 different content categories. It has 18 series;

(3) Hourly hits of Individual CIDRs aggregated by geographical locality. They are nested in the first type of series again but they are separated by the locality of the servers/recorders. It has hundreds of series.

(4) Total hourly hits of 'Other' CIDRs, i.e. $cidr == 0.0.0.0/27$, which represents all the other user activity not in captured CIDRs. It is aggregated over all the CIDRs other than the ones included in the first series. It has 1 series;

(5) Hourly hits of 'Other' CIDRs aggregated by category. They are similar to the second type except that they are nested in the forth type of series. It has 18 series;

Since there are no captured CIDRs for the forth type of series, we can not identify their locality. Thus no geographical separation is provided for them as the sixth type.

The first hour of the data is midnight to 1:00 a.m. on a Tuesday GMT. The length of the series is 13,104 observations.

In order to extract the series from the raw data, RHIPE MapReduce jobs were run. For example, to get the first series: Total hourly hits of Aggregated Individual CIDRs, in the map job, for observations with variable 'cidr' not equal to 0.0.0.0/27, the value of the variable 'hits' are collected as the value in the key value pair. The key is start time of that hour. The value is the hourly hits in that hour. In this way, all the hits with the same start time have the same key. In the reduce job, a simple summation of the values was generated for the same key. So all the hits in the same hour are aggregated for CIDR not equal to 0.0.0.0/27. After the key 'start time' is ordered, a simple univariate time series is produced as an object in R. The forth series are similar except we only catch the observations of variable 'cidr' with value 0.0.0.0/27.

For the second and fifth series, the procedures are similar except we only collect key value pairs in the map job for observations from that specific category in the raw data, i.e. only the series within that category are extracted and aggregated.

The data series of Total hourly hits of Aggregated Individual CIDRs and Total hourly hits of Other CIDRs are the two basic aggregate series in the dataset because they are captured /27 netblocks and not captured /27 netblocks respectively. If we add these two up we get all the hits in the dataset.

The first six observations of the 'Total hourly hits of Other CIDRs' series are:

```
[1] 44239407684 43968470998 41607957190
[4] 37885529154 32027381616 28101353937
```

When we do the analytics, we organize the series into a data frame:

```
                 time  hitrate week
1 2011-11-01 00:00:00 44.23941    1
2 2011-11-01 01:00:00 43.96847    1
```

```
3 2011-11-01 02:00:00 41.60796    1
4 2011-11-01 03:00:00 37.88553    1
5 2011-11-01 04:00:00 32.02738    1
6 2011-11-01 05:00:00 28.10135    1
```

The first column is the starting time of the observation. The second column is the hourly hits in billions. The third is the week that the observation is from, which is useful in analytical plots.

4.5.1 The Time Series Plot: Hourly Hits vs Hour

The general investigation of the series starts with the plots for two Total hourly hits series.

Aggregated series

left.pdf right.pdf

Figure 4.1.: Hits vs hour. Left: Aggregated Individual CIDR; Right: Other CIDR

In Figure 4.1 left, the hourly hits of Aggregated Individual CIDRs are plotted. The hits of all the captured /27 netblocks are aggregated and their value of variable 'cidr' can be anything other than 0.0.0.0/27. The unit is millions of hits. One panel shows one day's 24 observations. One row contains a week's data. There are 78 weeks and therefore there are 26 pages in one plot. The strip label shows which week the row is plotting.

For the Aggregated Individual CIDR series, we can identify a diurnal and a weekly pattern. For each day, the traffic decreases after midnight and increases with the start of working hours. The peak number of hourly hits of each day becomes smaller until the middle of the collection period and then increases again. So there might be a second degree trend. As for weekly cycle, Monday to Friday of each week have similar cycle of hits while the two weekend days have a significantly lower overall hits height in the cycle. That means the services experienced remarkably smaller visits during the weekends.

There are also some very small values close to the zero but none is equal to zero. We observe multiple periods of persistent anomalous behavior during the 78 weeks. We suspect these were recording equipment errors or traffic congestions in different measure points noticing that these are aggregated data.

In Figure 4.1 right, the Total hourly hits of Other CIDRs are plotted. They are the CIDRs other than the captured /27 netblocks and are labeled as 0.0.0.0/27. The unit is billions of hits instead of millions of hits. The setup of panels, rows and pages are similar to the first plot.

For the Other CIDR series, there is a clear diurnal pattern, but we do not observe a conspicuous weekly cycle. The diurnal cycle seems flatter compared with Aggregated Individual CIDR series. The peak number of hourly hits of each day gradually increases and becomes larger towards the end of the data collection period. Hence there is a strong sign for a first degree trend.

The small outliers in the series generally appear in the same period as the Aggregated Individual CIDR series.

Since these small outliers skew the scale of the display drastically in Figure 4.1, we remove the smaller outliers, i.e. replace the hourly hits less than 5 with 'NA', to zoom to the scale for better representation of the patterns in the series in the following two plots:

left.pdf right.pdf

Figure 4.2.: Cleaned hits vs hour. Left: Aggregated Individual CIDR; Right: Other CIDR

In Figure 4.2, less than 0.5% of the observations are removed as outliers so we do not lose too many observations. No space is wasted for them and the best aspect ratio for the ups and downs of the cycles is achieved.

Series by category The time series plots for the 18 series of Individual CIDRs aggregated by category are:

left.pdf center.pdf right.pdf

Figure 4.3.: Left: Automotive; Center: Business Services; Right: Consumer Goods

left.pdf center.pdf right.pdf

Figure 4.4.: Left: Consumer.Services; Center: Education; Right: Energy Utilities

left.pdf center.pdf right.pdf

Figure 4.5.: Left: Financial Services; Center: Foundation Not for Profit; Right: Gaming

left.pdf center.pdf right.pdf

Figure 4.6.: Left: High Technology; Center: Hotel Travel; Right: Manufacturing

left.pdf center.pdf right.pdf

Figure 4.7.: Left: Media&Entertainment; Center: No content category; Right: Pharma Health Care

left.pdf center.pdf right.pdf

Figure 4.8.: Left: Public Sector; Center: Retail; Right: Software as a Service

Most of these series by category have generally much smaller magnitude of hourly hits compared with the total aggregated series. Only two largest categories: 'Media & Entertainment' and 'No content category' have thousands of thousands hourly hits, which is in the neighborhood scale of the aggregated series. As for the other 16 series, most are in the magnitude of tens of thousands or hundreds of thousands hourly hits. Some series also have hour of the day and day of the week patterns as the aggregated series but the patterns are slightly different in terms of peak hours and seasonal shape. Some of them even have very irregular patterns due to their overall lack of hourly hits. This makes their sophisticated modeling almost meaningless. Simple statistics like mean and standard deviation might be enough to summarize them. Or an Autoregressive model could be helpful for these low hits series.

The time series plots for the 18 series of Other CIDRs aggregated by category are:

left.pdf center.pdf right.pdf

Figure 4.9.: Left: Automotive; Center: Business Services; Right: Consumer Goods

left.pdf center.pdf right.pdf

Figure 4.10.: Left: Consumer.Services; Center: Education; Right: Energy Utilities

left.pdf center.pdf right.pdf

Figure 4.11.: Left: Financial Services; Center: Foundation Not for Profit; Right: Gaming

left.pdf center.pdf right.pdf

Figure 4.12.: Left: High Technology; Center: Hotel Travel; Right: Manufacturing

left.pdf center.pdf right.pdf

Figure 4.13.: Left: Media&Entertainment; Center: No content category; Right: Pharma Health Care

left.pdf center.pdf right.pdf

Figure 4.14.: Left: Public Sector; Center: Retail; Right: Software as a Service

These Other CIDR by category plots also have similar patterns as the aggregated Other CIDRs. There are only clear hour of the day seasonal cycles but no obvious day of the week cycles. The largest two categories are still 'Media & Entertainment' and 'No Content Category' with tens of thousands of millions hourly hits. The remaining sixteen categories have several millions to several thousand millions hourly hits. These diurnal patterns are generally homogeneous, which are not ideal for STL modeling as exemplars since it is more of a waste of modeling capabilities.

4.5.2 The Time Series Plot: Hourly Hits vs Week

To see the change in the hourly hits for the same hour of a week, we produce these hourly hits vs week plots:

Aggregated series

left.pdf right.pdf

Figure 4.15.: Hits vs week. Left: Aggregated Individual CIDR; Right: Other CIDR

And for cleaned series where any observation less than 5 is marked as NA:

left.pdf right.pdf

Figure 4.16.: Clean hits vs week. Left: Aggregated Individual CIDR; Right: Other CIDR

In Figure 4.15 and Figure 4.16, the y axis is hourly hits and the x axis is the week time. The strip label is the hour number in a week. What is in the same panel are the 78 observations of the same hour in a week across 78 weeks. The pattern change among different season cycles is the focus of this plot.

The order of the panel is decided by the order of hour in a week. So the bottom first row of first page has all the first hour of each day from Tuesday to Monday. Then the second row of first page has all the second hour of each day, so on and so forth. There are 24 hours so there are $24 * 7 = 168$ panels in 6 pages in total.

In the Aggregated CIDR series, there is no obvious increase or decrease of hourly hits in each panel. The patterns are more bumpy especially for the hours in the afternoon and evening. It seems there is no precise parametric form to describe the patterns.

In the Other CIDR series, we see an upward trend for all the panels. Some have more linear patterns while the others show more curvy trends.

Series by category The by week plots for the 18 series of Individual CIDR by category are:

left.pdf center.pdf right.pdf

Figure 4.17.: Left: Automotive; Center: Business Services; Right: Consumer Goods

left.pdf center.pdf right.pdf

Figure 4.18.: Left: Consumer.Services; Center: Education; Right: Energy Utilities

left.pdf center.pdf right.pdf

Figure 4.19.: Left: Financial Services; Center: Foundation Not for Profit; Right: Gaming

left.pdf center.pdf right.pdf

Figure 4.20.: Left: High Technology; Center: Hotel Travel; Right: Manufacturing

left.pdf center.pdf right.pdf

Figure 4.21.: Left: Media&Entertainment; Center: No content category; Right: Pharma Health Care

left.pdf center.pdf right.pdf

Figure 4.22.: Left: Public Sector; Center: Retail; Right: Software as a Service

These plots can be assessed for the hints about seasonal window w_s and seasonal degree d_s. As a starting point, usually w_s can be a certain fraction of the total number of the seasonal cycles or points in each of the seasonal subseries panel. 1/4, 1/3 or 1/2 all can be the starting point to test if the fitted loess curve for the seasonal component gets enough but not too much smoothing. On the other hand, for d_s we are usually choosing from linear($d_s = 1$) or quadratic($d_s = 2$). From these plots, we see both choices are eligbile for a reason.

The by week plots for the 18 series of Other CIDR by category are:

left.pdf center.pdf right.pdf

Figure 4.23.: Left: Automotive; Center: Business Services; Right: Consumer Goods

left.pdf center.pdf right.pdf

Figure 4.24.: Left: Consumer.Services; Center: Education; Right: Energy Utilities

left.pdf center.pdf right.pdf

Figure 4.25.: Left: Financial Services; Center: Foundation Not for Profit; Right: Gaming

left.pdf center.pdf right.pdf

Figure 4.26.: Left: High Technology; Center: Hotel Travel; Right: Manufacturing

left.pdf center.pdf right.pdf

Figure 4.27.: Left: Media&Entertainment; Center: No content category; Right: Pharma Health Care

left.pdf center.pdf right.pdf

Figure 4.28.: Left: Public Sector; Center: Retail; Right: Software as a Service

The observation worth mentioning comparing these Other CIDR by category to the Individual CIDR by category is: some of the seasonal subseries present inconsistent behavior especailly in the first twenty or so weeks. In Figure 4.23 right, there is a huge jump in the frst 15 seasonal cycle then the rest run on a much lower and flatter level. All these different characteristics of the categories call for a systematic methodology to select the right w_s and d_s after the first test model.

4.5.3 Quantile Plots

In order to show the distribution of the hourly hits, we plot their quantiles:

left.pdf right.pdf

Figure 4.29.: Cleaned quantile plot. Left: Aggregated Individual CIDR; Right: Other CIDR

In Figure 4.29 left, if we kept the outliers, the aggregated Individual CIDR series would have had a heavy left tail. After cleaning-up, it has a distribution hard to quantify using

parametric methods. But we can imagine that the two more densed areas (8 to 10) and (20 to 25) on the y-axis match the hourly hits of weekends and early mornings of weekdays. In Figure 4.29 right, the other CIDR series also would have had a heavy left tail if we kept the outliers. After removing outliers, it displays an almost uniform distribution in the middle range.

4.6 Modeling Attempt for the Media and Entertainment Series

We use the largest category: Media and Entertainment in the Individual CIDR series as a showcase for data exploration and model building. Since it contains about 70% of the hits of the Aggregated Individual CIDR series, it has the similar magnitude of the aggregated series and preserves the characteristics in its pattern. As usual for the first step, we plot the series and try to get a first impression of the series.

4.6.1 General Informational Plots

pdf

Figure 4.30.: Media and Entertainment Series: the Time series plot: log hits vs hour

Remember in Figure 4.7: the time series plot for the Media and Entertainment Category of Individual CIDR, we identify a diurnal and weekly pattern just like in the Aggregated Individual CIDR series. So a natrual proposal for the parameter of length of the seasonal cycle is one week. That is, 24*7=168 observations in a cycle. Since we have 78 weeks of data, we have 78 seasonal cycles. Monday to Friday of each week have similar hits while the the two weekend days have generally less hits. That is a weekly pattern similar to the Aggregated Individual CIDR series as well.

We also observed some extremely small values which might be the results of recording equipment errors or server/traffic congestions. In Figure 4.30 we use a log base 2 transformation for the series, which is carefully decided after thorough consideration of the seasonal components in the later analysis. The unit is log base 2 millions of hits. The small values less than 1 after the log2 transformation are removed to keep the integrity of the visual scales.

In this section and chapter 4, all the plots and analysis are based on this cleaned log2 series with the same cut-off.

pdf

Figure 4.31.: Media and Entertainment Series: the Seasonal subseries plot: log hits vs week for each seasonal subseries

4.6.2 Diagnostics for the First STL+ Model

As we can see in the data displays, this series has many anomalies which complicate conventional analysis. Some of the anomalies can be readily explained. Others are not. So in order to categarize the patterns, our methods must be 'robust' to these anomalies, that is, able to describe general patterns without being distorted by the outliers. Bearing with robustness and fast computation in mind, the STL+ procedure is a natural fit to these challenges. On the other hand, the anomaly detection also requires the accuracy of the modeling. Only by quantifying what is normal can we determine what the anomalies are in an automated statistical way. The iterative re-weighting methods provide the robustness and loess filtering gives us the speed in the computation. In addition, non-parametric approaches are usually robust to model misspecification.

The STL+ model we fit here is: for $t = 1, 2, ..., 13104$,

$$M_t = T_t^M + S_t^M + R_t^M \tag{4.1}$$

Here M_t is the log2 cleaned Media Entertainment Individual CIDR series. T_t^M is the trend component of the STL+. S_t^M is the seasonal component and R_t^M is the remainder component.

There are six major parameters in model selection. They are:

(1). w_t: trend window; (2). w_s: seasonal window; (3). d_t: trend degree; (4). d_s: seasonal degree; (5). i_i: number of iterations in the inner loop and (6). i_o: number of iterations in the outer loop.

The initial selection of the values of these parameters are empirical and based on visual diagnostics.

Since there are 168 observations in a seasonal cycle, there are 168 seasonal subseries in the data. Figure 4.31 is a way to explore the mesh in which each seasonal subcycle varies. We put only one row in each page so that the pattern can be more closely scrutinized.

This plot helps us build a first impression with regards to the degree d_s and the window length w_s of the seasonal component. From the plot, we think $w_s = 35$, which is about 45% of the total number of cycles, is a good starting point to test the water in the parameter space. About the degree of the seasonal component d_s, it seems both first degree and second degree are applicable judged by the appearance of this plot. The fluctuations in some hour warrant the quadratic form while others are basically linear. We can start with a first degree seasonal $d_s = 1$.

pdf

Figure 4.32.: Media and Entertainment Series: the trend plot: the Weekly means of log hits vs week

In Figure 4.32, each point is a weekly mean. There are 78 weeks thus 78 points in total. It shows the general moving trend of the series. It dips at first then flats out from the mid field to the end. This might point to a quadratic form for trend degree d_t.

The components plots of STL modeling What we can see in the former plots is that the trend component T_t^M seems to have a second degree curve and the seasonal subseries can have a first degree order. So as a trial-and-error, we firstly fit a STL+ model with $d_t = 2, d_s = 1, w_t = 2601$ which is about 20% of the observations and $w_s = 35$. The number of inner loop i_i and outer loops i_o are both set to be 10 to ensure robustness.

pdf

Figure 4.33.: Media and Entertainment Series: The trend+seasonal and the real data

In Figure 4.33, log2 hits and the sum of trend and seasonal components $T_t^M + S_t^M$ are plotted against the hour t. Each panel is a day and each row is a week. The black points are the real observation. The red points are the summation of $S_t^M + T_t^M$.

This plot gives us a general idea how good the first STL+ model is in terms of characterizing the main pattern in the time series. Apparently $T_t^M + S_t^M$ are the parts in the observations explained by the model. With the remainder R_t^M added back in the real observations, the black points are supposed to be around the red points in a relatively consistent manner. And the red points should not be influenced too much by the extreme values of the black points. In this case we have a decent first model.

pdf

Figure 4.34.: Media and Entertainment Series: the trend plot: trend T_t^M and weekly means vs hour

In Figure 4.34, the trend component T_t^M are presented in a smoothed red curve while the weekly means are plotted in black points. The set-up is similar to Figure 4.32 except T_t^M is added. This plot is designed to illustrate if the trend component readily grasps the peaks and valleys of the time series. It shows that T_t^M fitting works well.

pdf

Figure 4.35.: Media and Entertainment Series: the seasonal series plot: the seasonal component vs hour

Figure 4.35 displays the detrended seasonal component S_t^M in the first STL model. The seasonal pattern verifies our original expectation. The weekdays have a larger number of hits than the weekends.

Since we have a diurnal pattern nested in the weekday pattern, two seasonal subseries plots are explored. In both plots seasonal component S_t^M is plotted against week. The conditional variable is either week day first and hour second or hour first and week day second.

The first plot is conditional on day of the week then on hour of the day:

pdf

Figure 4.36.: Media and Entertainment Series: the seasonal subseries plots: seasonal component S_t^M vs week, conditional on day of the week then on hour of the day

The second plot is conditional on hour of the day then on day of the week:

pdf

Figure 4.37.: Media and Entertainment Series: the seasonal subseries plots: seasonal component S_t^M vs week, conditional on hour of the day then on day of the week

In Figure 4.36, the panel factor day of the week is nested in the panel factor hour of the day. For the same hour, weekdays are compared with weekends. We can see for each hour of the same day, weekdays have a similar seasonal pattern and the weekends have a similar seasonal pattern.

In Figure 4.37, the panel factor hour of the day is nested in the panel factor day of the week. The seasonal patterns are compared within each day. There is a gradual change in the curve of each panel. How the hourly seasonal component evolves in different hours around clock is another hint for the selection of w_s and d_s. Notice the different shapes espcially between morning hours and afternoon/evening hours.

pdf

Figure 4.38.: Media and Entertainment Series: the detrended series plot: seasonal S_t^M and seasonal+remainder component $S_t^M + R_t^M$ vs week

In Figure 4.38, the remainder R_t^M is clipped, i.e. any $R_t^M < -1$ is labeled as -1. The clipping removes the extreme values which distort the aspect ratio and ensures that the display panels focus on the main data area. S_t^M is linked as a smoothed curve in black And the summation of seasonal and remainder components are plotted in red points. $S_t^M + R_t^M$ are generally randomly scattered around the seasonal curve, so the model appears generally healthy. the seasonal component S_t^M matches the inherent curves by and large in the detrended time series.

pdf

Figure 4.39.: Media and Entertainment Series: the remainder plot: the remainder component R_t^M vs hour

Again in Figure 4.39, the remainder R_t^M is clipped at -1. The remainder plot shows the remainders still have some inherent curves which means they are not yet completely random. We will use other methods to further treat this problem.

pdf

Figure 4.40.: Media and Entertainment Series: the remainder subseries plot: the remainder component R_t^M vs week

In Figure 4.40, the remainder R_t^M is clipped at -1. A Loess curve with span=1/3 is added for the remainders to check if there is some remaining pattern in the remainder subseries. This plot verifies what we found in the previous plot: there are some unfiltered patterns in the remainder. Further investigation is on the way.

pdf

Figure 4.41.: Media and Entertainment Series: the remainder R_t^M quantile plot

To further analyze the distribution of the remainder, in Figure 4.41 we produce a quantile plot of R_t^M. It is severely left skewed as we had anticipated.

The weight plots To check the final robustness weight g_t in the fitted STL+ model of each observation, three weight plots are exhibited: weight g_t vs hour, weight subseries plot and weight quantile plot.

left.pdf center.pdf right.pdf

Figure 4.42.: Media and Entertainment Series plots. Left: g_t vs hour; Center: the weight subseries plot: g_t vs week; Right: g_t quantile plot

In Figure 4.42, most of the observations are given reasonably large weight, which is close to 1. Meanwhile, the more extreme values are given smaller weights which is coherent with the idea of robust fitting.

All these plots show that the first choice of the parameters set passes the minimum requirement that the model fits the time series reasonably. The rule of thumb for the

parameter selection in STL+ as well as other models is: the model goodness of fit can not be simply verified by a set of statistics. The goal of thorough analytics is only achieved by checking these diagnostic plots in addition to the statistics. Only when these analytical plots give green light, we can be sure that a suitable model for the dataset is obtained.

5. TUNING PARAMETER SELECTION USING MINIMUM ABSOLUTE PREDICTION ERROR

5.1 Prediction Based Model Selection

According to the general guidelines in the STL and STL+ literature, the eligible parameters space is very large. In fact, for our example, the trend window w_t can be anything greater than, say, 1.91*168= 321. The seasonal window w_s can be anything greater than 7. The trend and seasonal degrees are also flexible. When the influence of each parameter on the model is evaluated independently, their effects are not linear. For example, a larger w_t is desired because we want to avoid over-fitting. But if it is too large, it would fail to capture the persistent trend in the series. When we consider w_s, a too large value of w_s would miss the high frequency variation in the series. If the effects of all the parameters are combined, it becomes much more complicated. Even if we only consider the four dimensional parameter space with w_t, w_s, d_t and d_s, the interactions among these four parameters are overwhelming. For one thing, the trend component and seasonal component can not be put into competition with each other in the frequency domain. And the choice of windows given different degrees is another step into the muddy water. Any change in one parameter could lead to butterfly effect in the model fitting and all the other parameters might endure adjustments. In order to find a best model in a systematic way, we propose a data-driven procedure to estimate the optimal parameter set based on the performance of prediction.

STL and STL+ model can predict one seasonal cycle away based on the model built on the historical data. For Akamai CIDR Media Entertainment series, suppose we are currently at time t_0, using the information till now, STL+ is capable of predicting values of $M_{t_0+1}, M_{t_0+2}, ..., M_{t_0+168}$, where $168 = 24 * 7$ is the number of observations in one seasonal cycle: a week. Using the standard time series terminolgy, for this case, the origin is t_0 and the lead time is from 1 to 168. The predictions for these lead times are calculated by setting

up all these 168 values as 'NA' in the dataset and run the STL+ function in R. The values of the predictions for $t = t_0 + 1, t_0 + 2, ..., t_0 + 168$ are:

$$P_t^M = T_t^M + S_t^M$$

Apparently based on different length or position of observations in the historical data, STL+ would give different values of the prediciton of the future one seasonal cycle. To fully utilize the information in this 78 weeks series, we construct a revolving system with multiple prediction series.

In our design, all prediction series are based on model built on historical data with a fixed predetermined length: in this example, 48 weeks, about 60% of the data.

We establish two sets of series, each set has 4873 series. (1). Modeling set of historical data series. Every series has 48 weeks of 48*168=8064 observations. (2). Prediction set of one cycle series. Every series has 1 week of 24*7=168 prediction values.

Each modeling series is used to fit the STL+ model with the parameter set to compute its prediction series. Each prediction series is computed to evaluate the prediction performance collectively.

In another word, for $k = 1, 2, ..., 4873$, the series k in the modeling set has observations $M_k, M_{k+1}, ..., M_{k+8063}$. The series k in prediction set has prediction value $P^M_{8064+k}, ..., P^M_{8231+k}$. So for the first prediction series, the origin is $t_0 = 8064$. It consists of $P^M_{8065}, P^M_{8066}, ..., P^M_{8232}$ and its modeling series consists of $M_1, M_2, ..., M_{8064}$.

After the first modeling series we step forward by adding one data point to the historical data at the end of the modeling series and removing the first data point in it. That becomes the second modeling series. Modeling series continue revolving in the same one step fashion until the historical data consist of the observations right before the last seasonal cycle in the time series, which is from week 30 to week 77. The last prediction series is the last seasonal cycle, week 78. That is why there are 29*168+1=4873 series in the modeling set and prediction set respectively.

5.2 Divide-and-Recombine Set-up for CIDR Prediction

Large, complex datasets, a.k.a, big data, are ubiquitous nowadays. The lower-level routine machine learning methodologies, unfortunately, often fail to grasp the crucial in-

formation from the datasets. For a cutting-edge data analyst and model builder, the goal is to keep the dynamic methodologies viable for the datasets elusive to the scale of the conventional statistical methods. That is the motivation of the Divide-and-Recombine framework. [21, Hafen et al, 2009] [39, Xi et al, 2010] [19, Guha et al, 2011] [2, Anderson et al, 2012] [20, Guha et al, 2012]

The finer granularity of the big data can not be simply reflected in summary statistics. Divide-and-Recombine sheds light on it by statistical sampling mechanism. Using Divide-and-Recombine, the original dataset is divided into representative subsets. This is called S computation [38, White, 2011]. In each subset, the detailed data are contained. These subsets are saved across the nodes of the cluster or cloud. Then both mathematical tools and visualization methods are applied to the sampled, if not all, subsets. This is the W computation. After the treatment the results of these analyses are 'recombined': we have a summary of the analyses or take various sophisticated forms of analysis. This is the B computation. In this sense, Divide-and-Recombine is a parallelization of the analysis.

There are two classes of division. [19, Guha et al, 2011] [40, Xia, 2011] The first is conditioning-variable division, where the subsets are defined by the Between-Subsets Variables(BSVs). The BSVs are the conditioning variables based on which the data are divided. Each value of the BSV defines and identifies one subset. The analysis is performed on the Within-Subset Variables(WSVs). The relationship of the WSVs are investigated. Recombination of the analysis also reports how these relationships of WSVs are different in terms of different BSVs. Conditioning-variable division is already widely used and it is implemented in the trellis display framework. [3, Becker et al, 1996] [31, Pinheiro et al, 2000] [17, fuentes et al, 2011] [32, Sarkar,2008] The second class is replicate division. The data are treated as replicates coming from the same experiment under the same conditions. Near-exact-replicate(NER) division, one of the replicate division methods, is capable of creating subsets that is representative of the characteristics of the whole dataset.

Three classes of recombination are proposed. [19, Guha et al, 2011] [40, Xia, 2011] In statistic recombination, the outputs of a statistical method applied to each subset are recombined. These outputs are usually in the form of statistics. The statistic could be an estimate of an estimand and the division and recombination procedure becomes a D&R estimator. The second class is analytic recombination. It is simply a continued analysis

of outputs in the W or B computation. Visualization recombination, the third class, is a display designed to combine all the subsets plots in the W computation. It rigorously reveals the granularity of the subsets data in a global perspective.

MapReduce, as an algorithm, is focused on the tactics of realizing local functions globally for the big data. Divide-and-Recombine, on the other hand, is a framework that puts thoughts on the strategy of attacking big data.

The dataset, which consists of 4873 prediction series, is massive if we use only one serial computing algorithm to calculate the results. But since Divide-and-Recombine assigns the tasks across the nodes, much faster and more efficient computation is expected. Using Divide-and-Recombine, the prediction set is treated as the BSV and the predictions, observations and errors are the WSVs.

To calculate the errors in our prediction sets, another map-reduce job was run. Since there are 4873 series of prediction, we use the set number as the key. Thus the key value is from 1 to 4873. Then for each key, the corresponding modeling series is extracted from the original series to calculate the STL+ predictions. Then the 168 absolute errors of a prediction series ordered by lag are saved as the value of the paring key. To expedite the calculation, since the bottle neck would be the I/O speed, the modeling series are extracted from the raw series and saved in HDFS first using a preliminary map-reduce job. Then the main job to actually calculate the predictions and errors is called to run on these seperately saved prediction series to streamline the I/O process. As output, the errors for each of the prediction series are read into an R object for further treatment.

5.3 Model Selection Criterion: Absolute Prediciton Error

The purposes of choosing an accuracy measure of the prediction are to quantify the extent of uncertainty in the forecasting and in turn help selecting the best model based on prediction performance. There are many measurements which all have their strength and weakness. Research indicates that the performances of different methods are related to the purpose of forecasting and the specific concerns of the situation using the forecasts. That is why from a theoretical point of view there is no single best method. [8, Brockwell et al, 2002]

The critiera to judge the different measurements refer to the reliability and discrimination of a method. Reliability is defined as the capability of producing consistent results when the methods are applied to different subsamples of the same series. Discrimination is the ability to tell apart better or worse models. Since it is not possible to optimize these criteria at the same time, trade-off should be considered.

Our proposed criteria to measure the performances of various parameter sets are based on the absolute prediction error: $E_t^M = |M_t - P_t^M|$. The ususal choice: Mean Squared Error is not suitable in this case because it is influeced by the outliers significantly. Our series are unique in that there is fair share of outliers. MSE gives too much sway to the outliers and the entire loss funciton is dominated by the extreme outliers in this dataset. The regular positions in the pattern contribute to the loss function only negligibly. What we want is a robust solution. The absolute prediction error does not discriminate against the regular values and limits the influence of the extreme outliers, which provides robustness in the assessment of the performance. On the other hand, all of the outliers are on the lower end of the data range. The absolute value of the error makes sure the influence from positive and negative errors are treated equally.

5.4 Experimental Design

An experiement can be designed to pay attention to one or more explanatory variables or factors. [29, Montgomery,2012] The effects of the factors are studied independently as well as how they are combined to influence the response variable. A factorial design contains the results obtained by combining each level of one explanatory variable with each level of another. It allows for simultaneous analysis of multiple factor effects in a process.

A set of single-factor experiments are inherently included in a factorial design. The analysis of interaction, which is a comparison among the simple effects of the experiment, as well as main effect, which is the average single-factor experiment, are the other two pieces of information in the factorial design. When interaction is found in the experiment, we need factorial experiment to isolate and single out the complexities of the explanations of the dataset. In the meantime, the factorial design allows us to control two or more

explanatory variables concurrently and to secure the deeper understanding of how these variables collectively produce the behavior in the dataset.

Because the levels of the factors are varied simultaneously rather than one at a time, the parameter space could be explored efficiently. In the full factorial design, responses are measured at all combinations of the experiment factor levels. The response measurement is an observation. Each experiment condition is a run and the entire set of runs is a design or an experiment.

In our design, we have four parameters in the parameter space: w_t, w_s, d_t and d_s. In every experiment, we fix the value of d_t and d_s and do a full factorial design for w_t and w_s to explore and cover the design space. Besides, the number of inner iterations i_i and the number of outer iterations i_o are also kept constant.

5.5 Best Parameter Set Selection Procedure

In order to find the best parameter set, we explore the parameter space and compare the results. The criteria are based on the minimization of the absolute prediction error. It is expected that the distribution of the absolute error would depend on the cycle position and the lag.

The prediction accuracy of one selected model is showed by the quantiles of the absolute errors by lag. Since the smallest and the largest quantiles would be unavoidably and disproportionately influenced by the extreme values or outliers, we consider five quantiles: .2,.35,.5,.65,.8. Based on these equally-spaced quantiles, holding trend degree d_t, seasonal degree d_s, number of inner iterations i_i and number of outer iterations i_o constant, we can compare the f the prediction series performances of the different combinations of trend windows w_t and seasonal windows w_s.

The factors are: w_t and w_s. Each factor has three levels in one experiment. So the proposed process is a 3^2 factorial design.

To illustrate the comparison process, suppose we have two runs: run 1 with w_{t1}, w_{s1}, d_t and d_s and run 2 with w_{t2}, w_{s2}, d_t and d_s(d_t and d_s are kept the same in these runs). Lag j goes from 1,2,... to 168.

At lag j, run 1 produces 4873 values and run 2 also produces 4873 values, each from one prediction series. The aforementioned five quantiles of these 4873 values are calculated for two runs separately. And for each quantile, the absolute prediction errors of these runs are plotted and appraised at each lag j. When there are more than two runs, similarly we are comparing their quantile curves in the plots.

We run multiple factorial designs, with the design space covering increasingly bigger w_t and w_s and we change both w_t and w_s ranges based on clues from previous experiments.

In all the following experimental series, inner and outer are fixed as $i_i = 10$ and $i_o = 10$. This ensures enough robustness of the fitting. In the experiment 1, we set $d_t = 2$ and $d_s = 2$ and vary the w_t and w_s in nine runs.

5.5.1 Experiment 1, 3^2 Full Factorial, $w_t = c(841, 1345, 1849), w_s = c(25, 30, 35), d_t = 2, d_s = 2$

left.pdf right.pdf

Figure 5.1.: Quantile plot of abs error vs lag conditional on $w_t * w_s$. Left: scale = 'sliced'; Right: scale = 'same'

In Figure 5.1, the quantiles of the absolute errors of the different predictions are plotted against the corresponding 168 lags. The second panel factor trend window w_t is nested in the first panel factor quantile. In each panel, three lines in different colors are plotted to show the differrences among the three seasonal windows w_s for the same trend window w_t. Black for $w_s = 25$, red for $w_s = 30$ and blue for $w_s = 35$.

The difference between these two plots is the different y scales used. The first uses 'sliced', which fixed the length of the y scale for each panel but the range is flexible. The second uses 'same', which fixed the range of the y scale so that the absolute discrepancy instead of relative one is revealed.

From the plots, we find the factor w_t has a larger influence on the response: absolute error than the factor w_s. And larger lags have larger absolute error, which is as expected. In the same panel, generally the larger seasonal window outperforms others especially for

the larger quantiles. In the same scale, we can see the difference in the smaller quantiles are negligible.

left.pdf right.pdf

Figure 5.2.: Quantile plot of abs error vs lag conditional on w_s * w_t. Left: scale = 'sliced'; Right: scale = 'same'

Figure 5.2 is similar to Figure 5.1 except the second factor seasonal window w_s is nested in the first panel factor quantile. In each panel, three lines in different colors are plotted to show the differences among the three trend windows w_t for the same seasonal window w_s. Black for $w_t = 841$, red for $w_t = 1345$ and blue for $w_t = 1949$.

From the plots, in the same panel, the larger trend window w_t outperforms others especially in the larger quantiles again. The hint that larger trend and seasonal window are better leads to a natural exploration into a new parameter space with larger trend and seasonal windows. So in the second experiment, the parameter sets use larger window values. The parameter values of $w_t = 1849$ and $w_s = 35$ are kept to compete with the new sets of parameters.

5.5.2 Experiment 2, $w_t = c(1849, 2353, 2857), w_s = c(35, 40, 45), d_t = 2, d_s = 2$

left.pdf right.pdf

Figure 5.3.: Quantile plot of abs error vs lag conditional on $w_t * w_s$. Left: scale = 'sliced'; Right: scale = 'same'

For Figure 5.3, in the same panel, the larger seasonal window outperforms others especially in the larger quantiles.

left.pdf right.pdf

Figure 5.4.: Quantile plot of abs error vs lag conditional on $w_s * w_t$. Left: scale = 'sliced'; Right: scale = 'same'

For Figure 5.4, in the same panel, it seems that larger trend window outperforms others especially in the larger quantiles. The results in the experiment 2 are similar to the ones in the experiment 1. So in the third experiment, the parameter sets again use larger both window values. The values $w_t = 2857$ and $w_s = 45$ are kept to compare with the new sets of parameters.

5.5.3 Experiment 3, $w_t = c(2857, 3361, 3865), w_s = c(45, 50, 55), d_t = 2, d_s = 2$

left.pdf right.pdf

Figure 5.5.: Quantile plot of abs error vs lag conditional on $w_t * w_s$. Left: scale = 'sliced'; Right: scale = 'same'

For Figure 5.5, the difference is very limited. If we focus on the larger quantile which has relatively larger differences among different seasonal windows, seasonal window 45 performs the best.

left.pdf right.pdf

Figure 5.6.: Quantile plot of abs error vs lag conditional on $w_s * w_t$. Left: scale = 'sliced'; Right: scale = 'same'

For Figure 5.6, in the same panel, it seems trend window 2857 and 3361 with seasonal window 45 are the top performers.

The results in the experiment 3 reveal that the benefits of taking larger trend window and seasonal window are reversed. To verify this judgment, in the forth experiment, the parameter sets use even larger window values to compare with the current benchmark: $w_t = 2857$ and $w_d = 45$. Note the steps are larger in the fourth experiment for trend window, which is 1858 between different trend window instead of 504 in the previous experiments. This makes sure that we exhaust the possible trend window space and that we are not trapped in some local minimum.

5.5.4 Experiment 4, $w_t = c(2857, 4705, 6553), w_s = c(45, 50, 55), d_t = 2, d_s = 2$

left.pdf right.pdf

Figure 5.7.: Quantile plot of abs error vs lag conditional on $w_t * w_s$. Left: scale = 'sliced'; Right: scale = 'same'

For Figure 5.7, the differences are apparant. Taking into consideration of all the quantiles especially the larger ones, it seems seasonal window 45 beats other seasonal windows hands down.

left.pdf right.pdf

Figure 5.8.: Quantile plot of abs error vs lag conditional on $w_s * w_t$. Left: scale = 'sliced'; Right: scale = 'same'

From Figure 5.8, in almost every panel, the trend window 2857 has smallest absolute error at almost all the lags. The results in the experiment 4 give us more confidence that the current benchmark: $w_t = 2857$ and $w_s = 45$ are the best in all the prameters sets in the third and fourth experiment.

In a word, the parameter space navigation process starts from reasonable smaller values in experiment 1 and heads to the larger value areas in subsequent experiments following the comparison results. Experiment 4 reaches large enough values that return much worse results. These four experiments investigate large range and out of them two parameter sets are the top performers.

5.6 Visual Model Validation

In the first two experiments, the best parameter set is clear: $w_t = 1849$ and $w_s = 35$. In the third and forth experiment, the best parameter set is: $w_t = 2857$ and $w_s = 45$. These two models are the best so far. In this section, we dig deeper to assess their goodness-of-fit with following plots and diagnostics.

5.6.1 Parameter Set 1: $w_t = 1849$ and $w_s = 35$

These plots are organized in the same rationale as the ones in 4.6.2. They check the model fitting in all aspects.

left.pdf center.pdf right.pdf

Figure 5.9.: Left: $T_t^M + S_t^M$ and M_t vs hour; Center: S_t^M vs hour; Right: T_t^M and weekly means vs hour.

left.pdf center.pdf right.pdf

Figure 5.10.: The subseries plots. Left: S_t^M vs week; Center: S_t^M vs week conditional on hour and weekday; Right: S_t^M and $S_t^M + R_t^M$ vs week.

In Figure 5.10 right, R_t is clipped at -1. All these plots reflect harmonious patterns, which bodes good fit for the model.

left.pdf center.pdf right.pdf

Figure 5.11.: Left: R_t^M vs hour; Center: R_t^M vs week; Right: R_t^M quantile plot.

In Figure 5.11 left and center, R_t is clipped at -1. For the left plot, the remainders present some correlation among them, the same issues raised as in model: $w_t = 2601, w_s = 35, d_t = 2$ and $d_s = 1$. In the center plot, the loess curves for the consecutive hours of the same day have similar curvature. That means the remainders in consecutive hours are not independent. Both plots signal that we need to explore more into this phenomenon.

left.pdf center.pdf right.pdf

Figure 5.12.: Left: g_t vs hour; Center: g_t vs week; Right: g_t quantile plot.

All the weight plots in Figure 5.12 have regular patterns. We conclude the robust process works reasonably well.

5.6.2 Parameter Set 2: $w_t = 2857$ and $w_s = 45$

These plots are similar to the ones in the subsection above.

left.pdf center.pdf right.pdf

Figure 5.13.: Left: $T_t^M + S_t^M$ and M_t vs hour; Center: S_t^M vs hour; Right: T_t^M and weekly means vs hour.

left.pdf center.pdf right.pdf

Figure 5.14.: The subseries plots. Left: S_t^M vs week; Center: S_t^M vs week conditional on hour and weekday; Right: S_t^M and $S_t^M + R_t^M$ vs week.

In Figure 5.14 right, R_t is clipped at -1. All these plots showed that the model operates well.

left.pdf center.pdf right.pdf

Figure 5.15.: Left: R_t^M vs hour; Center: R_t^M vs week; Right: R_t^M quantile plot.

In Figure 5.15 left and center, R_t is clipped at -1. The correlation of the remainders in both plots asks for solution of this issue.

left.pdf center.pdf right.pdf

Figure 5.16.: Left: g_t vs hour; Center: g_t vs week; Right: g_t quantile plot.

All the weight plots in Figure 5.16 have well-behaved patterns. The weights are assigned accordingly.

5.7 Autoregressive Modeling for the Remainder

To resolve the issue raised in the remainder plots, we proposed an autoregressive model to account for the correlation among the remainders. From the diagnostic plots, it seems a remainder is strongly correlated with the two remainders right before it.

So the proposed autoregressive model is:

$$R_j = \alpha_0 + \alpha_1 R_{j-1} + \alpha_2 R_{j-2};\ \text{for } j = 3, 4, ..., 13104. \tag{5.1}$$

The fitted regression coefficients for the remainders in the model with $w_t = 1849$ and $w_s = 35$ are: $\alpha_0 = -0.0010;\ \alpha_1 = 1.1490;\ \alpha_2 = -0.2038$.

The thorough evaluation of a model never ends at only presenting the parameters of the model. The model goodness-of-fit is assessed by two residual plots.

left.pdf right.pdf

Figure 5.17.: Left: RR_j vs hour; Right: RR_j normal quantile plot.

In Figure 5.17 left, these residuals are randomly scattered. The proposed autoregressive model successfully removes the correlation. Figure 5.17 right shows the residuals are close to normally distributed but with a little heavier tails.

The fitted regression coefficients for the remainders in the model with $w_t = 2857$ and $w_s = 45$ are: $\alpha_0 = -0.0011;\ \alpha_1 = 1.2039;\ \alpha_2 = -0.2506$.

left.pdf right.pdf

Figure 5.18.: Left: RR_j vs hour; Right: RR_j normal quantile plot.

The residual plots prove that the residuals are independent and close to normal.

5.8 Model Pair Comparison

The $3 * 3$ comparison among nine parameter sets sheds light on the global view of the parameter space covered in these nine sets. However, if we want to reveal the more subtle difference between the two candidate parameter sets, a local view of the difference is more effective and powerful. The following dianostic plots are designed to tell apart two sets.

For example, if we want to further compare the following two sets: mod (1): $w_t = 1849, w_s = 35, d_t = 2, d_s = 2, i_i = 10, i_o = 10$ and mod (2): $w_t = 2857, w_s = 45, d_t = 2, d_s = 2, i_1 = 10, i_o = 10$.

left.pdf center.pdf right.pdf

Figure 5.19.: Left: R_t^M quantile quantile plot; Center: R_t^M scatter plot; right: Scatter plot of mod(1) -mod(2) R_t^M vs mod 2 R_t^M

In Figure 5.19 left, these two model's remainders are almost identically distributed. On the left tail, the mod(1) has smaller remainders than the mod(2). In Figure 5.19 center, the remainders from the two models have very strong correlation. In Figure 5.19 right, the difference of the two model's remainders has a negative correlation with the mod(2) remainder. Most of the points are in the neighborhood of the origin.

left.pdf center.pdf right.pdf

Figure 5.20.: Left: RR_j quantile quantile plot; Center: RR_j scatter plot; right: Scatter plot of mod(1) -mod(2) RR_j vs mod 2 RR_j

In Figure 5.19, the residuals of the Autoregressive models of the remainders in the two models are compared. The correlation presented between the two model's remainders is mostly removed. And the differences are minimum. In a sense, the parameters of both models are located in the best performance area in the parameter space. The model selection process successfully identifies this sweet spot in the parameter space.

6. D&R TIME SERIES ESTIMATION

In Divide-and-Recombine, one of the important features is that to achieve efficiency, the data is parallelized instead of the algorithm. This expands the application of D&R to algorithms that can not be parallelized in nature on subsets of the big data. [19, Guha et al, 2011] [40, Xia, 2011] Replicate division is introduced to efficiently treat the more homogeneous large datasets. For example, the subsets after conditioning-variable division that are still deemed to be too large. The idea is this: the n observations in a dataset are treated as replicates in the same experiment. For each observation, there are $p + 1$ variables. The first is the response variable. The remaining p are explanatory variables. A statistical or machine learning model is built to describe the relationship between response and explanatory variables expressed as a function of these explanatory variables. The estimates of the coefficients in the model are calculated separately in each subset. Then the final estimates are obtained by combining these estimates from each subset. We call them D&R estimates.

Individual replicate division approaches are bound to have various influence on the output of the D&R estimates. However, there is still a lack of systemmatic study on the precise effects and performances of assorted replicate division methods.

We run a series of simulation to compare the coefficient estimation performance of four types of Divide-and-Recombine division and full series direct estimates for Autoregressive models including Autoregressive series with Gaussian errors, FARIMA series and heavier tailed Autoregressive series. Standard statistics and diagnostic displays are coupled to analyze the effects.

The notations used in the simulations are: n is the sample size of each sample. r is the number of the non-overlapping subsets of D&R in one sample. $m = n/r$ is the number of observations allocated to each subset. s is the number of samples in one run of the simulation.

For each subset, because there are m observations, there are m rows in the design matrix of the model fitting. In Autoregressive model, once the first column of the design matrix of a subset is decided, all the other elements in the design matrix and the vector of observations on the response variable are also determined. For example, in an AR(2) model, if X_3 is put into the first column first row of the design matrix of subset 1, then X_4 is the first element of the response vector and X_2 is the second explanatory variable and thus at the second column of the first row in the design matrix.

The four types of replicate division methods are as following:

(1) Local division with regard to t('Local t'): the first explanatory variable X_t's in each subset are grouped locally with regards to t. For subset $j, j = 1, ..., r$, the first column of the design matrix of subset j is:

$X_{1+m(j-1)}$,$X_{2+m(j-1)}$,...,$X_{m+m(j-1)}$.

(2) Near Exact Replicates with regard to t('ner t'): every subset has similar X_t's in terms of the order of t. In this spirit the first r observations are put into these r subsets in order, one for each. The second r observations are separated in the same way. So on and so forth. For subset $j, j = 1, ..., r$, the first column of the design matrix of subset j is: X_j, X_{r+j}, ... $X_{(m-1)r+j}$.

(3) Near Exact Replicates with regard to x('ner x'): every subset has similar X_t's in terms of the value of X_t. Firstly we sort X_t, then do the same division process as in Near Exact Replicates with regard to t. For first column of each subset, X_t's with similar values are put into the same subset.

(4) Random: randomly assign X_t's to the subsets without replacement.

In this simulation study all the D&R estimates are the mean of the subset estimates in the sample.

6.1 D&R for Autoregressive Gaussian Series

We start from the most basic autoregressive models with Gaussian error terms. The autoregressive process of order p is denoted AR(p) and defined by:

$$X_t = \sum_{r=1}^{p} \alpha_r X_{t-r} + \epsilon_t$$

where α_r's are fixed constants and $\epsilon_t \sim iidN(0,1)$. [25, Kendall, 1976] [7, Brockwell et al, 1986] Four types of Gaussian error term models with different coefficents and orders are tested.

6.1.1 $\alpha = 0.99$

We generated:

$$X_t = \alpha X_{t-1} + \epsilon_t, \text{ where } \epsilon_t \sim N(0,1), \alpha = .99, t = 1, 2, ..., n.$$

Since the series is stationary, $X_0 \sim N(0, \frac{1}{1-\alpha^2} = 50.2516)$. Split these n observations into r subsets of size m. Run a D&R Autoregressive order 1 model on these subsets. The examples of the four types of divisions are as following:

(1) Local division with regard to t: e.g. in the first subset, response variable: $X_1, X_2, ..., X_m$; explanatory variable: $X_0, X_1, ..., X_{m-1}$.

(2) Near Exact Replicates with regard to t: for each subset, pick one X_t in every $r = 20$ X_i's. e.g. in the first subset, response variable: $X_1, X_{21}, ..., X_{n-19}$; explanatory variable: $X_0, X_{20}, ..., X_{n-20}$.

(3) Near Exact Replicates with regard to x: e.g. in the first subset, the explanatory variable is the m smallest X_t's.

(4) Random: e.g. what is in the first subset could be any m X_t's.

We run the simulation in runs with varying sample size and subset size. All the series are generated with the same random seed.

Run 1: n = 1000, r = 20, m = 50, s=100

pdf

Figure 6.1.: Time series of the generated X_t

In Figure 6.1, on the vertical axis is generated X_t. On the horizontal axis is t.

pdf

Figure 6.2.: Normal Quantiles of the D&R estimates using 4 different division methods and direct estimate

In Figure 6.2, on the vertical axis is the D&R estimate. On the horizontal axis is the normal quantile. The horizontal line is the value of the real α. All the estimates are close to the real α. They are mostly normally distributed with slightly lighter right tails and heavier left tails. The lighter right tails are the sign of the boundary effect since estimates are influenced by the boundary value 1 of α. The 'local t' division has the worst performance while all others have similar results. It is expected since in 'local t' division the X_t's in the same subset are strongly correlated.

pdf

Figure 6.3.: Boxplots of the estimates using 4 different division methods and direct estimate

In Figure 6.3, on the vertical axis is the division method. On the horizontal axis is the estimates. The fact that all these estimates are left-skewed is expected since .99 is very close to the right boundary of the estimates support 1.

Table 6.1.: Statistics of the estimates of these 4 division methods and direct estimate

	mean	bias	sd	mse	var	var/mse
local.t	0.9652414	-0.024758621	0.012137558	7.588364e-04	1.473203e-04	0.1941397
ner.t	0.9866429	-0.003357125	0.005925503	4.603076e-05	3.511158e-05	0.7627852
ner.x	0.9861699	-0.003830058	0.006142530	5.202272e-05	3.773068e-05	0.7252731
random	0.9862890	-0.003711028	0.006362834	5.385252e-05	4.048565e-05	0.7517875
all	0.9861720	-0.003828035	0.006133159	5.189333e-05	3.761564e-05	0.7248646

From Table 6.1, the results are similar to what we found in the plots. 'Local t' is much worse than the others. 'Ner t' is the best since the division method largely eliminates the correlation in the subsets. Surprisingly it is even better than the full series direct estimate. Note the ratio Var/MSE of the 'local t' is extremely small because the bias is relatively huge compared to the variance.

Run 2: n = 500, r = 20, m = 25, s=100

left.pdf center.pdf right.pdf

Figure 6.4.: Left: Time series of the generated X_t; Center: Normal Quantiles of the estimates; Right: Boxplots of the estimates

Table 6.2.: Statistics of the estimates of these 4 division methods and direct estimate

	mean	bias	sd	mse	var	var/mse
local.t	0.9460891	-0.043910937	0.023107329	0.0024567796	5.339487e-04	0.2173368
ner.t	0.9819216	-0.008078390	0.009234891	0.0001496908	8.528321e-05	0.5697293
ner.x	0.9807226	-0.009277365	0.009583480	0.0001769942	9.184309e-05	0.5189046
random	0.9810784	-0.008921645	0.010176678	0.0001821249	1.035648e-04	0.5686470
all	0.9807325	-0.009267517	0.009579536	0.0001767367	9.176751e-05	0.5192329

The results are similar to the first run. Although the sample size n is much smaller, the estimates are still pretty good. Meanwhile, the ratios Var/MSE are all much smaller compared to the first run. Interestingly the biases of the estimates increase much faster than the variances when we decrease the sample size.

Run 3: n = 250, r = 10, m = 25, s=100

left.pdf center.pdf right.pdf

Figure 6.5.: Left: Time series of the generated X_t; Center: Normal Quantiles of the estimates; Right: Boxplots of the estimates

Table 6.3.: Statistics of the estimates of these 4 division methods and direct estimate

	mean	bias	sd	mse	var	var/mse
local.t	0.9328231	-0.05717693	0.02994933	0.0041571944	0.0008969625	0.2157615
ner.t	0.9703400	-0.01966002	0.01780387	0.0007003240	0.0003169776	0.4526156
ner.x	0.9732399	-0.01676006	0.01815501	0.0006072077	0.0003296042	0.5428196
random	0.9694896	-0.02051042	0.01886937	0.0007731698	0.0003560532	0.4605110
all	0.9692113	-0.02078866	0.01810258	0.0007565948	0.0003277034	0.4331293

In this run, it appears that the sample size n is too small that the performance of the estimators deteriorates quickly compared to the last run. In addition to the biases, the variances pick up substantially.

6.1.2 $\alpha = -0.99$

The only difference between this section and the first one is $\alpha = -.99$ instead of .99

Run 1: n = 1000, r = 20, m = 50, s=100

left.pdf center.pdf right.pdf

Figure 6.6.: Left: Time series of the generated X_t; Center: Normal Quantiles of the estimates; Right: Boxplots of the estimates

Table 6.4.: Statistics of the estimates of these 4 division methods and direct estimate

	mean	bias	sd	mse	var	var/mse
local.t	-0.9668324	0.023167588	0.010597729	6.479259e-04	1.123119e-04	0.1733406
ner.t	-0.9885026	0.001497407	0.004640097	2.355743e-05	2.153050e-05	0.9139582
ner.x	-0.9881226	0.001877436	0.004747850	2.584142e-05	2.254208e-05	0.8723234
random	-0.9883120	0.001687995	0.004797246	2.563276e-05	2.301356e-05	0.8978185
all	-0.9880959	0.001904058	0.004764404	2.609798e-05	2.269954e-05	0.8697814

Compared to the experiment $\alpha = .99$ with the same n, all the estimates have even smaller bias and standard deviation. The performances of the estimates are consistent with the previous trial.

Run 2: n = 500, r = 20, m = 25, s=100

left.pdf center.pdf right.pdf

Figure 6.7.: Left: Time series of the generated X_t; Center: Normal Quantiles of the estimates; Right: Boxplots of the estimates

Table 6.5.: Statistics of the estimates of these 4 division methods and direct estimate

	mean	bias	sd	mse	var	var/mse
local.t	-0.9522259	0.037774061	0.019440153	1.801020e-03	3.779196e-04	0.2098364
ner.t	-0.9877868	0.002213226	0.006360457	4.494922e-05	4.045541e-05	0.9000246
ner.x	-0.9869127	0.003087307	0.006712267	5.413544e-05	4.505452e-05	0.8322556
random	-0.9871743	0.002825673	0.006679857	5.215871e-05	4.462049e-05	0.8554752
all	-0.9868781	0.003121902	0.006748626	5.483478e-05	4.554395e-05	0.8305669

Run 3: n = 250, r = 10, m = 25, s=100

left.pdf center.pdf right.pdf

Figure 6.8.: Left: Time series of the generated X_t; Center: Normal Quantiles of the estimates; Right: Boxplots of the estimates

Table 6.6.: Statistics of the estimates of these 4 division methods and direct estimate

	mean	bias	sd	mse	var	var/mse
local.t	-0.9524674	0.037532564	0.03267825	0.0024658824	0.0010678677	0.4330570
ner.t	-0.9836877	0.006312312	0.01214192	0.0001857973	0.0001474263	0.7934791
ner.x	-0.9848365	0.005163538	0.01439514	0.0002318099	0.0002072200	0.8939220
random	-0.9831228	0.006877210	0.01372227	0.0002337136	0.0001883006	0.8056895
all	-0.9828323	0.007167690	0.01320078	0.0002238938	0.0001742606	0.7783182

Even with $n = 250$, the estimates in all the division models except 'local t' are still very accurate.

6.1.3 $\alpha_1 = 1.42, \alpha_2 = -.73$

We generated

$$X_t = \alpha_1 X_{t-1} + \alpha_2 X_{t-2} + \epsilon_t + 14.35, \text{ where } \epsilon_t \sim N(0, 5.8921);$$
$$\alpha_1 = 1.42; \alpha_2 = -.73; t = 2, 3, ..., n+1. \quad (6.1)$$

Here X_0 and $X_1 \sim N(0, 227.8)$ and the data are centered by the mean. Run a Divide-and-Recombine autoregressive order 2 model on the subsets.

Run 1: n = 1000, r = 20, m = 50, s=100

left.pdf center.pdf right.pdf

Figure 6.9.: Left: Time series of the generated X_t; Center: Normal Quantiles of the estimates; Right: Boxplots of the estimates

In this order 2 model, 'local t' is still the worst performer. The correlation within the subsets pulls the estimations of the two efficients closer to 0 than they should be. Meanwhile, there is little difference among other estimates.

Table 6.7.: Statistics of the estimates of these 4 division methods and direct estimate

α_1	mean	bias	sd	mse	var	var/mse
local.t	1.393027	-0.0269726187	0.02603446	0.0013985376	0.0006777933	0.4846444
ner.t	1.419048	-0.0009520206	0.02486737	0.0006131084	0.0006183860	1.0086078
ner.x	1.418870	-0.0011301615	0.02399693	0.0005713714	0.0005758527	1.0078430
random	1.418656	-0.0013440058	0.02267123	0.0005106510	0.0005139845	1.0065279
all	1.418161	-0.0018394942	0.02318171	0.0005354013	0.0005373915	1.0037172

α_2	mean	bias	sd	mse	var	var/mse
local.t	-0.7098341	2.016593e-02	0.02359710	0.0009579200	0.0005568233	0.5812837
ner.t	-0.7301049	-1.048663e-04	0.02271896	0.0005110004	0.0005161509	1.0100793
ner.x	-0.7304116	-4.116257e-04	0.02203389	0.0004808067	0.0004854921	1.0097451
random	-0.7300541	-5.410678e-05	0.02133426	0.0004506020	0.0004551506	1.0100944
all	-0.7295119	4.881148e-04	0.02151384	0.0004584551	0.0004628452	1.0095761

What is interesting in the table is: all the var/mse except 'local t' are larger than 1. This happens when the sample size in a simulation is small and the bias is much smaller than the standard deviation. Besides, all the estimates except the 'local t' are over-estimating the coefficients.

Run 2: n = 500, r = 20, m = 25, s=100

left.pdf center.pdf right.pdf

Figure 6.10.: Left: Time series of the generated X_t; Center: Normal Quantiles of the estimates; Right: Boxplots of the estimates

Table 6.8.: Statistics of the estimates of these 4 division methods and direct estimate

α_1	mean	bias	sd	mse	var	var/mse
local.t	1.376150	-0.043850376	0.03607273	0.003211085	0.001301242	0.4052344
ner.t	1.425616	0.005616379	0.03518894	0.001257423	0.001238261	0.9847616
ner.x	1.423566	0.003566462	0.03486795	0.001216336	0.001215774	0.9995380
random	1.423045	0.003044566	0.03484349	0.001211198	0.001214069	1.0023706
all	1.422433	0.002432963	0.03345710	0.001114103	0.001119377	1.0047343

α_2	mean	bias	sd	mse	var	var/mse
local.t	-0.6984240	0.031576035	0.03410350	0.002148464	0.001163048	0.5413395
ner.t	-0.7359314	-0.005931353	0.03497232	0.001246013	0.001223063	0.9815810
ner.x	-0.7344690	-0.004469048	0.03480069	0.001218950	0.001211088	0.9935506
random	-0.7330912	-0.003091222	0.03407877	0.001159305	0.001161363	1.0017752
all	-0.7334905	-0.003490541	0.03251785	0.001059021	0.001057411	0.9984799

Run 3: n = 250, r = 10, m = 25, s=100

left.pdf center.pdf right.pdf

Figure 6.11.: Left: Time series of the generated X_t; Center: Normal Quantiles of the estimates; Right: Boxplots of the estimates

Table 6.9.: Statistics of the estimates of these 4 division methods and direct estimate

α_1	mean	bias	sd	mse	var	var/mse
local.t	1.374526	-0.0454742058	0.05552843	0.005120476	0.003083406	0.6021718
ner.t	1.417335	-0.0026648316	0.05526951	0.003031272	0.003054718	1.0077347
ner.x	1.419819	-0.0001810521	0.05373507	0.002858615	0.002887457	1.0100894
random	1.416438	-0.0035622989	0.05435973	0.002938120	0.002954980	1.0057383
all	1.415923	-0.0040771002	0.05105298	0.002596966	0.002606407	1.0036355

α_2	mean	bias	sd	mse	var	var/mse
local.t	-0.6973293	0.0326706973	0.04874322	0.003419517	0.002375901	0.6948061
ner.t	-0.7273029	0.0026970994	0.04970107	0.002452769	0.002470196	1.0071053
ner.x	-0.7296489	0.0003511107	0.04854267	0.002332950	0.002356391	1.0100476
random	-0.7290335	0.0009664785	0.04809424	0.002290859	0.002313056	1.0096891
all	-0.7282051	0.0017948851	0.04739176	0.002226740	0.002245979	1.0086396

In this run, the estimates are underestimating the coefficients instead of overestimating in the two previous runs.

6.1.4 $\alpha_1 = 0.6, \alpha_2 = -.6^2, \alpha_3 = .6^3, ..., \alpha_{10} = -.6^{10}$

We generated

$$X_t = \alpha_1 X_{t-1} + \alpha_2 X_{t-2} + ... + \alpha_{10} X_{t-10} + \epsilon_t, \text{ where } \alpha_j = .6^j, j = 1, ..., 10, t = 10, 11, ..., 1009.$$

While it is analytically cumbersome to calculate the variance of X_0, we can do the generation by truncating the observations in the burn-in period.

Run 1: n = 1000, r = 20, m = 50, s=100

left.pdf center.pdf right.pdf

Figure 6.12.: Left: Time series of the generated X_t; Center: Normal Quantiles of the estimates; Right: Boxplots of the estimates

left.pdf center.pdf right.pdf

Figure 6.13.: Left: Means; Center: Bias; Right: Standard Deviation

In Figure 6.13, on the vertical axis is the statistics(mean,bias or sd) of the estimates. On the horizontal axis is the lag. Panel variable is the division method.

left.pdf center.pdf right.pdf

Figure 6.14.: Left: Mean Squared Error; Center: Variance; Right: Variance/MSE

In Figure 6.14, on the vertical axis is the statistics(MSE, variance or Var/MSE) of the estimates. On the horizontal axis is the lag. Panel variable is the division method.

For this model with 10 lags, the 'local t' has largest bias as usual but it beats other division methods in terms of standard deviation. As a result, all the four division methods have similar MSE's, which are larger than the MSE of direct estimate.

Run 2: n = 500, r = 20, m = 25, s=100

left.pdf center.pdf right.pdf

Figure 6.15.: Left: Time series of the generated X_t; Center: Normal Quantiles of the estimates; Right: Boxplots of the estimates

left.pdf center.pdf right.pdf

Figure 6.16.: Left: Means; Center: Bias; Right: Standard Deviation

left.pdf center.pdf right.pdf

Figure 6.17.: Left: Mean Squared Error; Center: Variance; Right: Variance/MSE

In Figure 6.16 and Figure 6.17, on the vertical axis is the statistics of the estimates. On the horizontal axis is the lag. Panel variable is the division method.

The run is also similar to the first. But with fewer observations in the sample, the MSE increases quickly for all the four division methods.

Run 3: n = 250, r = 10, m = 25, s=100

left.pdf center.pdf right.pdf

Figure 6.18.: Left: Time series of the generated X_t; Center: Normal Quantiles of the estimates; Right: Boxplots of the estimates

left.pdf center.pdf right.pdf

Figure 6.19.: Left: Means; Center: Bias; Right: Standard Deviation

left.pdf center.pdf right.pdf

Figure 6.20.: Left: Mean Squared Error; Center: Variance; Right: Variance/MSE

In Figure 6.19 and Figure 6.20, on the vertical axis is the statistics of the estimates. On the horizontal axis is the lag. Panel variable is the division method.

Even with $r = 10$, the performances of the estimates are very stable. The 'local t' has surprisingly smaller MSE than other division methods although it still has the largest bias.

6.2 D&R for Long Range Dependent Gaussian Series

The Hosking discrete analog of fractional Gaussian noise (fGn) is a fractionally differenced white noise. The fractional difference operator ∇^d is defined by a binomial series:

$$\nabla^d = (1-B)^d = \sum_{k=0}^{\infty} \binom{d}{k} (-B)^k = 1 - dB - \frac{1}{2}d(1-d)B^2 - ...,$$

where B is the backward-shift operator and d is in the range $-\frac{1}{2} \leq d \leq \frac{1}{2}$. The series is called a an ARIMA(0,d,0) process and is a long range dependent Gaussian series. [24, Hosking, 1981],

What we generate is an approximate series with only 128 lags:

$$X_t = \sum_{i=1}^{128} \pi_i X_{t-i} + \epsilon_t,$$

$$\text{where } \epsilon_t \sim N(0,1), \text{and } \pi_i = \frac{(i-d-1)!}{i!(-d-1)!}, d = .31, i = 1, 2, ..., 128, t = 128, 129, ...n + 127. \tag{6.2}$$

The generation starts with $x_0 \sim N(0,1)$ and use the coefficient π_1 only to generate x_1. Then x_2 is built on two coefficients π_1 and π_2 and x_0 and x_1. So on and so forth until we get the the first 128 x_i's. Then we use the complete 128 coefficients to generate all the other values. After that, the burn-in period is truncated.

In this experiment, n = 5000000, r = 5000, m = 1000, s=25. Since the n is too large, there are no direct estimates of the coefficients calculated. Fit a Divide-and-Recombine autoregressive regression model on these subsets and take the mean of the estimates to be the D&R estimates of the regression coefficients. Use three types of divisions and their examples are as following:

(1) Local division with regard to t: e.g. in the first subset, vector of response variable: $X_{128}, X_{129}, ..., X_{1127}$; first column of the design matrix: $X_{127}, X_{128}, ..., X_{1126}$.

(2) Near Exact Replicates with regard to t: for each subset, pick one X_t in every $r = 5000$ X_i's. e.g. in the first subset, vector of response variable: $X_{128}, X_{5128}, ..., X_{5000128}$; first column of the design matrix: $X_{127}, X_{5127}, ..., X_{5000127}$.

(3) Random: e.g. what is in the first subset could be any m X_t's.

pdf

Figure 6.21.: Normal Quantiles of the D&R estimates

On the vertical axis is the estimates. On the horizontal axis is the normal quantile. The horizontal line is the value of the actual alpha.

left.pdf center.pdf right.pdf

Figure 6.22.: Left: Mean; Center: Bias; Right: Standard Deviation

In Figure 6.22 left, on the vertical axis is the mean. On the horizontal axis is the lag. Panel variable is the division method. The actual alpha is superposd as red. In Figure 6.22 center and right, on the vertical axis is the bias or sd. On the horizontal axis is the lag. Panel variable is the division method. All the biases of the same division methods have similar magnitude for distinctive lags. The 'local t' method has the largest bias again. But all the standard deviations are similar among the three methods.

left.pdf center.pdf right.pdf

Figure 6.23.: Left: MSE; Center: Variance; Right: Var/MSE

left.pdf center.pdf right.pdf

Figure 6.24.: Left: Log10(Means) of the estimates; Center: (Mean of estimates) / (actual alpha) ; Right: (Mean of estimates) / SD

In Figure 6.24 left, through log transformation, we are able to stretch the scale of the estimates in the latter lags, which delivers discerning power to the visual tool. Although the relative bias becomes larger when the lag beccomes larger, all the division methods work reasonably well. The worst is still 'local t' and there is little difference between the other two methods. Figure 6.24 center is another way to standardize the estimates. It leads to similar conclusion as the previous one.

6.3 D&R for Heavier Tail Residual Series

In this section, we explore the time series with error terms with heavier tails. The reason that we choose t distribution with degree of freedom 3 is that it has the heaviest tail and largest variance in the t distribution family.

We generated:

$$X_t = \alpha X_{t-1} + \epsilon_t, \text{ where } \epsilon_t \sim t_3(0,1), \alpha = .99, t = 1, 2, ..., n.$$

Since the series is stationary, $X_0 \sim t_3(0, \frac{1}{1-\alpha^2} = 50.2516)$.

Linear least-squares estimates could behave badly when the distribution of the error terms are not normal. For this kind of heavier tail errors, it is even worse. One remedy is to remove the influential observations from the fitting process, which is not applicable here since we can not decide which observations should be removed or not in this case. Another solution is robust regression, which is not as vulnerable as least-squares regression to outliers. Our method of choice is Tukey's bisquare. [37, Tukey, 1977] Tukey's bisquare is

one of the most popular M-estimators in robust regression. The loss function of residual r_i is defined as:

$$\rho'(r_i) = \begin{cases} r_i\{1-(\frac{r_i}{c})^2\}^2 & \text{if } |r_i| \le c \\ 0 & \text{if } |r_i| > c \end{cases}$$

As usual, we split these n observations into r subsets of size m. Instead of a least-squares regression, we run a D&R bisquare autoregressive order 1 model on these subsets.

6.3.1 Run 1: n = 1000, r = 20, m = 50, s=100

left.pdf center.pdf right.pdf

Figure 6.25.: Left: Time series of the generated X_t; Center: Normal Quantiles of the estimates; Right: Boxplots of the estimates

Table 6.10.: Statistics of the estimates of these 4 division methods and direct estimate

	mean	bias	sd	mse	var	var/mse
local.t	0.9746307	-0.0153693170	0.009387054	3.234515e-04	8.811678e-05	0.2724265
ner.t	0.9891782	-0.0008218477	0.003524767	1.297518e-05	1.242398e-05	0.9575194
ner.x	0.9887990	-0.0012009527	0.003709538	1.506536e-05	1.376068e-05	0.9133986
random	0.9889635	-0.0010365000	0.003842386	1.569063e-05	1.476393e-05	0.9409397
all	0.9889350	-0.0010650301	0.003649019	1.431648e-05	1.331534e-05	0.9300711

The results are consistent with what we see in the Gaussian error term cases. 'Local t' is the worst performer while others are neck to neck. The biases are actually smaller across the board than the ones in the Gaussian error term series. Because this is a bisquare autoregressive model which is a robust procedure. And $n = 1000$ seems to be a sufficiently large sample size.

6.3.2 Run 2: n = 500, r = 20, m = 25, s=100

left.pdf center.pdf right.pdf

Figure 6.26.: Left: Time series of the generated X_t; Center: Normal Quantiles of the estimates; Right: Boxplots of the estimates

Table 6.11.: Statistics of the estimates of these 4 division methods and direct estimate

	mean	bias	sd	mse	var	var/mse
local.t	0.9552726	-0.034727447	0.018500754	1.544851e-03	3.422779e-04	0.2215605
ner.t	0.9846537	-0.005346334	0.007402735	8.283576e-05	5.480048e-05	0.6615558
ner.x	0.9840037	-0.005996262	0.007801825	9.621494e-05	6.086847e-05	0.6326301
random	0.9841908	-0.005809154	0.007694521	9.235987e-05	5.920565e-05	0.6410322
all	0.9842708	-0.005729165	0.007396605	8.698600e-05	5.470977e-05	0.6289491

Again smaller sample size leads to larger bias, standard deviation and MSE.

6.3.3 Run 3: n = 250, r = 10, m = 25, s=100

left.pdf center.pdf right.pdf

Figure 6.27.: Left: Time series of the generated X_t; Center: Normal Quantiles of the estimates; Right: Boxplots of the estimates

Table 6.12.: Statistics of the estimates of these 4 division methods and direct estimate

	mean	bias	sd	mse	var	var/mse
local.t	0.9471162	-0.042883795	0.02647230	0.0025327946	0.0007007825	0.2766835
ner.t	0.9794476	-0.010552382	0.01275925	0.0002725234	0.0001627986	0.5973748
ner.x	0.9825058	-0.007494193	0.01263840	0.0002142948	0.0001597291	0.7453711
random	0.9793783	-0.010621741	0.01259276	0.0002698132	0.0001585776	0.5877310
all	0.9800813	-0.009918742	0.01172790	0.0002345496	0.0001375436	0.5864158

$n = 250$ again proves to be an insufficient sample size. The estimates are universally inaccurate.

7. SUMMARY

STL modeling has vast application in seasonal adjustable time series modeling. To check model goodness-of-fit, a series of visual diagnostics is combined to make an informational judgment call. The plots for observations and trend, seasonal and remainder components of STL model are instrumental in providing the necessary clues to choose a first model. Among them, subseries plots demonstrate the properties of the seasonal cycles including the day of the week pattern(larger cycle) and hour of the day(smaller cycle). The weight plots also illustrate how the robustness feature of STL modeling is realized.

The proposed work on model fitting and parameter estimation based on quantiles of the absolute prediction error is the first attempt to formalize the model selection procedure for STL+. The D&R prediction series is designed to maximize the information extraction from the series while balancing the influence of all the observations. Fair share of outliers in the dataset supports the necessity of a robustness approach, which in term indicates using absolute prediction error as the criterion for model selection is a best decision. The complication in navigating the parameter space is the motivation of the factorial experiment in the design. The comparison plots illustrate the contrasts among combinations of various levels of the parameters, which are the factors in the experiment, as well as differing lags. Deeper diagnostics are performed to confirm how good the model is fitted.

The dependency remaining in the remainder component justifies an order 2 Autoregressive model for the remainders. After the treatment, it appears that the dependency is successfully removed. The residual plots from the AR(2) model are helpful in the pair-wise comparison in the model selection as well.

The replicate division methods in the D&R framework are still in their developing stage. The simulation runs to illuminate the differences among four replicate division methods are an effort to study the characteristics of the divisions, considering different sizes of the samples and subsets. Three different types of models: autoregressive models with Gaussian

terms, an autoregressive model with heavier tail terms and a persistent autoregressive model are selected to elucidate the comparison of results.

Future Works Future works can be devoted to the further development of the experimental designs in deciding the best STL+ model. For example, how to choose the best steps between levels of the factors given the data and parameter space. Besides, hundreds of series by geographical information in the Akamai CIDR data could be candidates for future STL+ modeling. Based on the STL+ model established by the model selection, the next step is to perform anomaly detection for the recorded Internet traffic. More models can be studied using the similar experiments in the replicate division methods comparison. New division methods, which might be customary to certain specific statistical or machine learning models, could be introduced.

LIST OF REFERENCES

LIST OF REFERENCES

[1] Akamai Technologies, Inc. World Wide Web, http://www.akamai.com, 2014.

[2] David Anderson, Bowei Xi and William S. Cleveland, *Multifractal and Gaussian Fractional Sum-difference Models for Internet Traffic*, Technical Report, Department of Statistics, Purdue University.

[3] RA. Becker, William S. Cleveland and MJ Shyu, *The Design and Control of Trellis Display*, Journal of Computational and Statistical Graphics, 1996, 5, 123-155.

[4] 'BGP Analysis Reports'. Retrieved 2013-01-09.

[5] S. Bianconcini *Loess Asymmetric Filters for Real Time Economic Analysis*, Second Italian Congress of Econometrics and Empirical Economics, 2007.

[6] G.E.P. Box and G. Jenkins *Time Series Analysis, Forcasting and Control*, Holden-Day, San Francisco.

[7] P.J. Brockwell and R.A. Davis, *Times Series: Theory and Methods*, Springer Series in Statistics, 1986.

[8] P.J. Brockwell and R.A. Davis, *Introduction to Time Series and Forecasting* Springer, 2002.

[9] F.A.G. Butter and T.J. Mourik *Seasonal Adjustment Using Structural Time Series Models: An Application and Comparison with Census X-11 Method*, Journal of Business and Economic Statistics 1990, 8(4), 385-393.

[10] William S Cleveland and Susan J Devlin, *Locally Weighted Regression: an Approach to Regression Analysis by Local Fitting*, Journal of the American Statistical Association 1988, 83, 596-610.

[11] Robert B. Cleveland and William S. Cleveland and Jean E. McRae and Irma Terpenning, *STL: A Seasonal-Trend Decomposition Procedure Based on Loess*, Journal of Official Statistics 1990, 6(1), 3-73.

[12] William S. Cleveland and E. Grosse, *Computational methods for local regression*, Statistics and Computing, 1991, 1(1), 47-62.

[13] William S. Cleveland and E. Grosse, *Local Regression Models*, Statistical Models in S, 1992, 309-376.

[14] William S. Cleveland and C. Loader, *Smoothing by Local Regression: Principles and Methods*, Statistical Theory and Computational Aspects of Smoothing, 1996, 10-49.

[15] Grzegorz Czajkowski, Marin Dvorsk, Jerry Zhao, and Michael Conley, *Sorting Petabytes with MapReduce - The Next Episode*, Google, 2011.

[16] Jeffrey Dean and Sanjay Ghemawat, *MapReduce: Simplified Data Processing on Large Clusters*, OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, 2004.

[17] M. Fuentes, Bowei Xi and William S. Cleveland, *Trellis Display for Modeling Data from Designed Experiments* Statistical Analysis and Data Mining, 2011,4, 133-145.

[18] Michael A. Gallo, *Networking explained*, Digital Press, 1999.

[19] Saptarshi Guha, Paul Kidwell, Ashrith Barthur, William S. Cleveland, J. Gerth and C. Bullard *A Streaming Statistical Algorithm for Detection of SSH Keystroke packets in TCP Connections* In Operations Research, Computing and Homeland Defense. Institute for Operations Research and Management Sciences, Hanover, Maryland, USA.

[20] S. Guha, R. Hafen, J. Rounds, J. Xia , J. Li, B. Xi, and W. S. Cleveland, *Large Complex Data: Divide and Recombine (D&R) with RHIPE*, Stat, 1: 5367.

[21] Ryan P. Hafen, DE Anderson, William S. Cleveland R. Maciejewski, DS. Ebert, A. Abusalah, M Yakout, SM. Ouzzani and S. Grannis, *Syndromic surveillance: STL for modeling, visualizing and monitoring disease counts*, BMC Medical Informatics and Decision Making, 2009, 9:21.

[22] Ryan P. Hafen, *Local Regression Models: Advancements, Applications and New Methods*, Ph. D. Dissertation 2010, Purdue University.

[23] James D. Hamilton, *Time Series Analysis*, Princeton University Press, 1994.

[24] J.R.M. Hosking, *Fractional Differencing* Biometrika, 1981, 68(1): 165-176.

[25] M. Kendall, *Time Series* Charles Griffin, 1976.

[26] R. Lmmel, *Google's Map Reduce programming model Revisited*, Science of Computer Programming,2008, 70: 1.

[27] Frederick R. Macaulay, *The Smoothing of Time Series*, National Bureau of Economic Research, 1933.

[28] S. Maravall and D.A. Pierce, *On Structural Time Series Models and the Characterisation of Components* Journal of Business and Economic Statistics 1985, 3, 350-355.

[29] Douglas C. Montgomery, *Design and Analysis of Experiments*, Wiley, 8th edition, 2012.

[30] Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun, *The Akamai Network: A Platform for High-Performance Internet Applications*, ACM SIGOPS Operating Systems Review, 2010, Vol. 44, No.3.

[31] JC. Pinheiro and DM. Bates, *Mixed-Effects Models in S and S-Plus* Springer, 2000, New York.

[32] D. Sarkar, *Lattice: Multivariate Data Visualization with R*, Springer, 2008, New York.

[33] Julius Shiskin, Allan H. Young, and John C. Musgrave, *The X-11 Variant of the Census Method II Seasonal Adjustment Program* Bureau of the Census, Technical Paper No.15.

[34] Core Development Group, World Wide Web. http://github.com/saptarshiguha/rhipe/, 2011.

[35] Robert H. Shumway, David S. Stoffer and David S. Stoffer *Time Series Analysis and Its Applications: With R Examples* Springer-Verlag New York, 2006.

[36] Ed Taylor, *TCP/IP Complete (Complete Series)* Computing Mcgraw-Hill, 1998.

[37] J.W. Tukey, *Exploratory Data Analysis* Addison-Wesley Publishers, 1977.

[38] T. White, *Hadoop: The Definitive Guide*, second edition, O'Reilly, 2011, Sebastopol, CA.

[39] Bowei Xi, Hui Chen, William S. Cleveland and T. Telkamp, *Statistical Analysis and Modeling of Internet VoIP Traffic for Network Engineering*, Electronic Journal of Statistics, 2010, 4, 58-116.

[40] Jin Xia, *Divide and Recombine (D&R) for the Analysis of Large and Complex Data Sets with Application to VOIP* Ph. D. Dissertation 2011, Purdue University.

APPENDICES

VITA

VITA

Xiang Han graduated from Nanjing University with Bachelor degree in Management Science in 2001. He earned his master degree in Mathematical Sciences at Clemson University in 2006. His research interests include statistical model building, machine learning, time series, data visualization and Bayesian statistics.