Stan 소개 및 응용

발표자: 문현지
발표일: 2018.12.03 (월) 13:30 – 14:45
발표장: 서울대학교

PRESENTATION CONTENTS

Please download package from: https://github.com/StanKorea/2018_Bayesian

1__ **Stan**

2__ **Stan Coding**

3__ **Stan Ecosystem**

4__ **Application: Time Series Forecasting**

5__ **Stan Korea**

Stan

What is Stan

Open source probabilistic programming language, inference algorithms
Statistical modeling and computation platform used for modeling, data analysis, and prediction

Program

Declares data and (constrained) parameter variables
Defines log posterior or (penalized) likelihood

Algorithm

MCMC sampling for full Bayes
Variational inference for approximate Bayes
Optimization for penalized MLE

Ecosystem

Language and math Library (C++)
Interfaces and tools (R, python, cmd, many more)
Documentation (example models, reference manual, case studies, R package vignettes)

Stan

Sampling through adaptive neighbor, Algorithms

MCMC sampling for full Bayes

NUTS : No-U-Turn Sampler
HMC : Hamiltonan Monte Carlo

Variational inference for approximate Bayes

ADVI : Automatic Differentiation Variational Inference

Optimization for penalized MLE

LBFGS : Limited-memory BFGS

Stan

Workflow for using Stan for Bayesian Inference

1. Represent a statistical model by writing its log posterior density using the Stan modeling language.
2. Translate the Stan program to C++ code using the stanc function.
3. Compile the C++ code to create a Dynamic Shared Object (DSO) that can be loaded by R.
4. Run the DSO to sample from the posterior distribution.
5. Diagnose non-convergence of the MCMC chains.
6. Conduct inference based on the posterior sample (MCMC draws from the posterior distribution).

-> stan( ) function

Stan

Where Stan Stands: Comparison to other Probabilistic Programming Languages

Stan	Fast Hamiltonian Monte-Carlo. Good for production use. Short History. There are Less books, journal articles, and tutorials on its use.
BUGS	Long History. There are many books, journal articles, and tutorials on its use. Written in Component Pascal, which very few people know, so development is very difficult
JAGS	Very easy to get running. Uses a variety of MCMC sampling methods. Doesn't have helper packages in R. Breaks down when sampling for complicated models.
PyMC	Smoothest integration with Python. Lacking in modeling features.
Edward	Very low level language built on top of TensorFlow supporting variational inference methods. Risky choice for production use.

Stan

Statistical Rethinking: A Bayesian Course with Examples in R and Stan

Bayesian Data Analysis

"essential characteristic is their **explicit use of probability for quantifying uncertainty**"
modeling uncertainty: uncertain parameters, models, decisions

Hierarchical Model (~ Multilevel Model)

model differenct and dependent parameters, pooling
- adjust estimates for repeat sampling
- adjust estimates for imbalance in sampling
- to study variation
- to avoid averaging

Model Comparison using Information Criteria

avoid overfitting & comparison of multiple non-null models to the same data

Stan Coding

Stan Block Language

data{

what data we have observed: outcome, variable, predictors

}

parameters{

relevant unobserved which we want to make inference

}

model{

how observed & unobserved, data & parameter relate to each other

}

generated quantities{

use the updated posterior to make prediction:
assuming the model is right, what other things you want to know?

}

Stan Coding

Example1: Eight School

School	Estimate (y_j)	Standard Error (σ_j)
A	28	15
B	8	10
C	-3	16
D	7	11
E	-1	9
F	1	11
G	18	10
H	12	18

$$y_j \sim \text{Normal}(\theta_j, \sigma_j), \quad j = 1, \ldots, 8$$
$$\theta_j \sim \text{Normal}(\mu, \tau), \quad j = 1, \ldots, 8$$

$$p(\mu, \tau) \propto 1$$

each σj assumed known

"We use the Eight Schools example here because it is simple but also represents a nontrivial Markov chain simulation problem in that there is dependence between the parameters of original interest in the study — the effects of coaching in each of the eight schools — and the hyperparameter representing the variation of these effects in the modeled population."

http://mc-stan.org/rstan/articles/rstan.html

Stan Coding

Example1: Eight School

```
data {
  int<lower=0> J;          // number of schools
  real y[J];               // estimated treatment effects
  real<lower=0> sigma[J]; // standard error of effect estimates
}
parameters {
  real mu;                 // population treatment effect
  real<lower=0> tau;       // standard deviation in treatment effects
  vector[J] eta;           // unscaled deviation from mu by school
}
transformed parameters {
  vector[J] theta = mu + tau * eta;        // school treatment effects
}
model {
  target += normal_lpdf(eta | 0, 1);       // prior log-density
  target += normal_lpdf(y | theta, sigma); // log-likelihood
}
```

Stan Coding

Example2: Baseball Hits

Complete Pooling

$$p(y_n \mid \phi) = \text{Binomial}(y_n \mid K_n, \phi).$$

$$p(y \mid \phi) = \prod_{n=1}^{N} \text{Binomial}(y_n \mid K_n, \phi).$$

$$p(\phi) = \text{Uniform}(\phi \mid 0, 1) = 1.$$

```
parameters {
  real<lower=0, upper=1> phi;
  // chance of success (pooled)
}
model {
  y ~ binomial(K, phi);
}
```

No Pooling

$$p(\theta_n) = \text{Uniform}(\theta_n \mid 0, 1),$$

$$p(\theta) = \prod_{n=1}^{N} \text{Uniform}(\theta_n \mid 0, 1).$$

$$p(y_n \mid \theta_n) = \text{Binomial}(y_n \mid K_n, \theta_n).$$

$$p(y \mid \theta) = \prod_{n=1}^{N} \text{Binomial}(y_n \mid K_n, \theta_n).$$

```
parameters {
  vector<lower=0, upper=1>[N] theta;
  // chance of success
}
model {
  y ~ binomial(K, theta);
  //likelihood
}
```

Partial Pooling

$$p(\theta_n \mid \alpha, \beta) = \text{Beta}(\theta_n \mid \alpha, \beta),$$

$$p(\theta \mid \alpha, \beta) = \prod_{n=1}^{N} \text{Beta}(\theta_n \mid \alpha, \beta).$$

$$\alpha = \kappa\phi \qquad \beta = \kappa(1 - \phi).$$

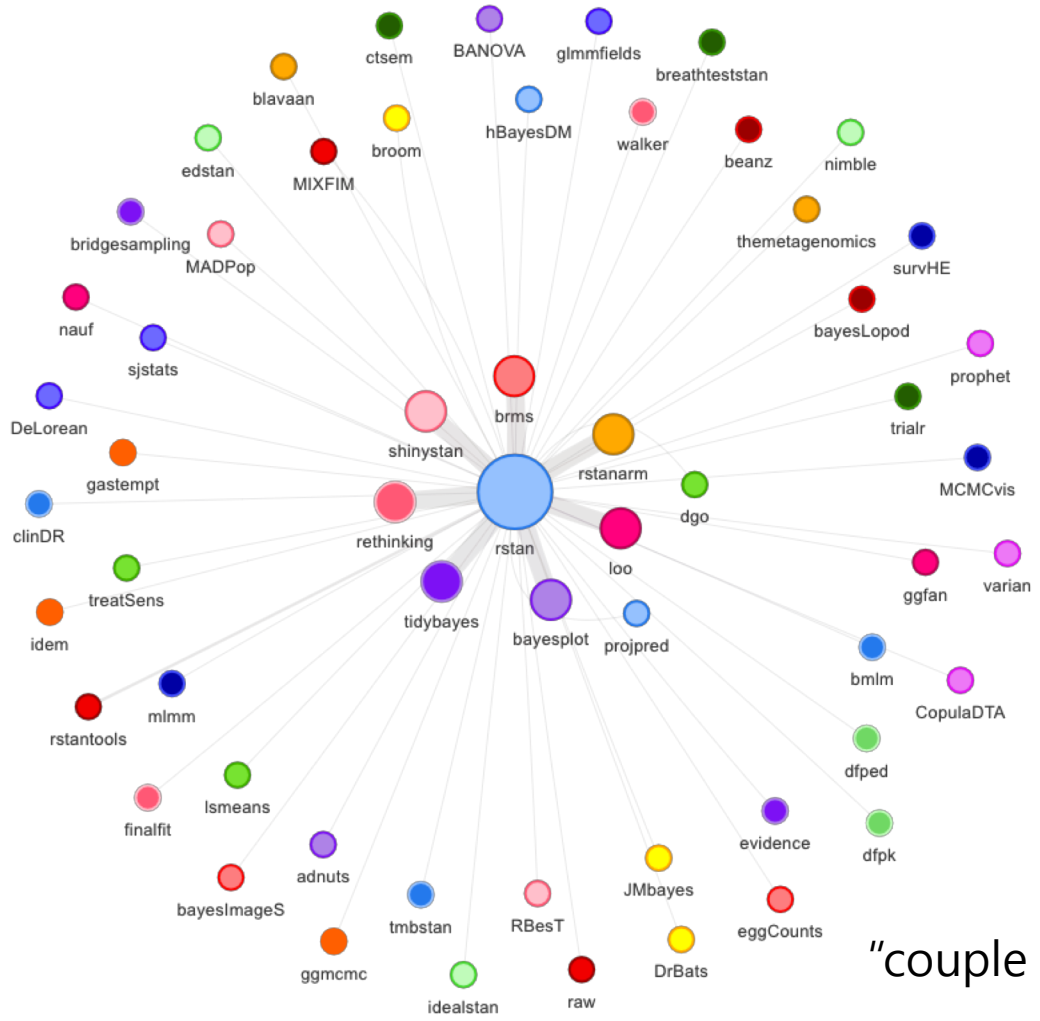
$$p(\phi) = \text{Uniform}(\phi \mid 0, 1),$$

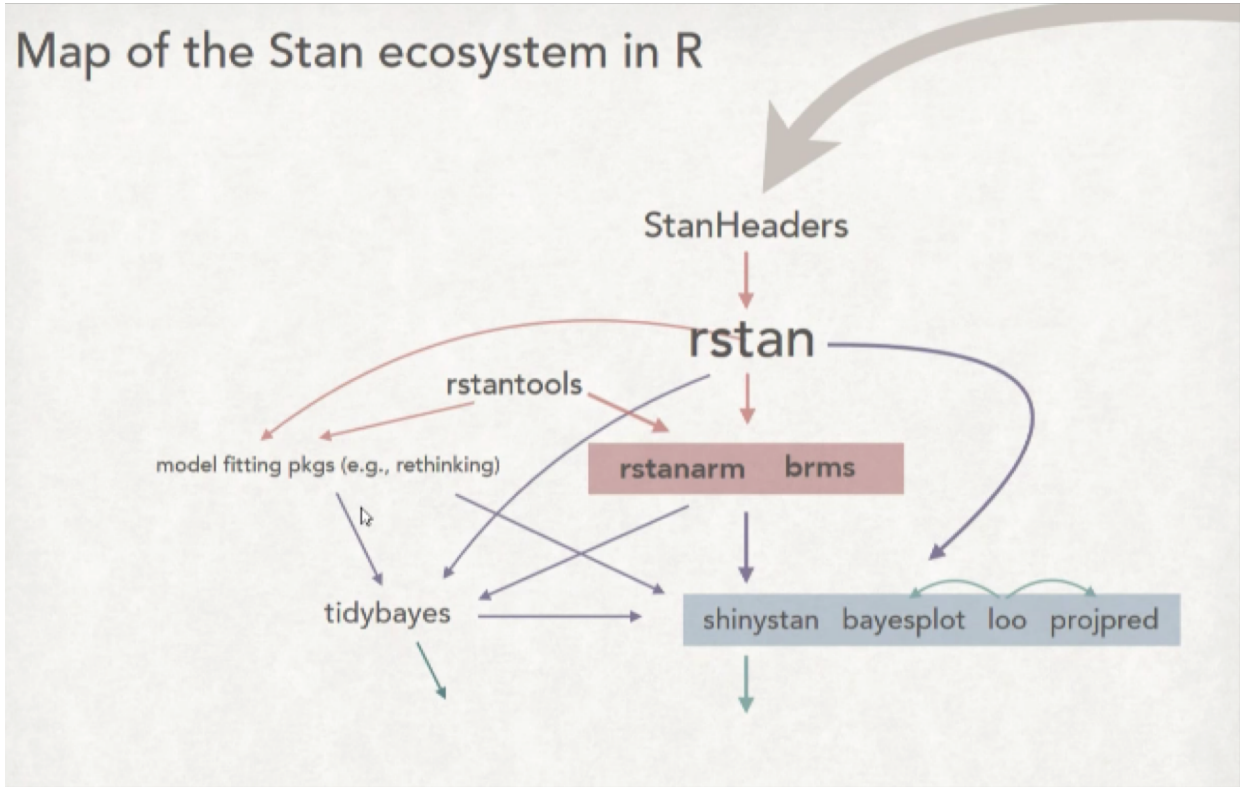
$$p(\kappa) = \text{Pareto}(\kappa \mid 1, 1.5) \propto \kappa^{-2.5}.$$

```
parameters {
  real<lower=0, upper=1> phi;
  // population chance of success
  real<lower=1> kappa;
  //population concentration
  vector<lower=0, upper=1> [N] theta;
  // chance of success
}
model {
  kappa ~ pareto(1, 1.5); // hyperprior
  theta ~ beta(phi * kappa, (1 - phi) * kappa); // prior
  y ~ binomial(K, theta); // likelihood
}
```

Stan Ecosystem

Rstan Ecosystem





"couple dozen packages working with Stan under the hood"

Stan Korea

Stan Ecosystem

Rstanarm & Brms

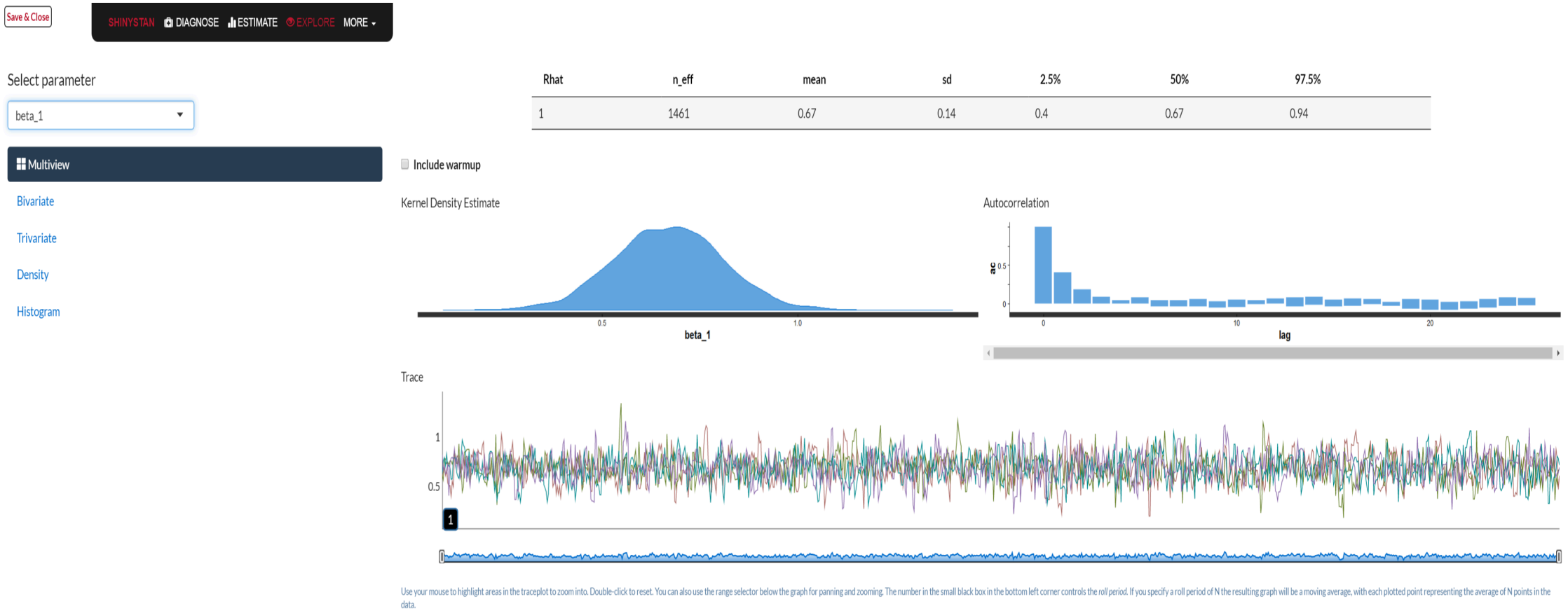
```
stan_glm(y ~ x + z + (1|g),
         data=d,
         family = binomial(link = "logit"),
         prior = normal(0, 1),
         prior_covariance = decov(regularization = 1, concentration = .5),
         QR = TRUE,
         chains = 2,
         cores = 2,
         iter = 2000)


brm(y ~ x + z + (1|g),
    data=d,
    family = bernoulli(link = 'logit'),
    prior = prior(normal(0, 1), class = b,
                  cauchy(0, 5), class = sd),
    chains = 2,
    cores = 2,
    iter = 2000)
```

Analysis	rstanarm	brms
lm	√	
glm	√	
multinomial		√
ordinal		√
mixed	√	√
additive	√	√
regularized	√	√
beyond		√

Source: https://m-clark.github.io/bayesian-basics/

Stan Ecosystem

ShinyStan: analysis and visualization GUI for MCMC



http://mc-stan.org/shinystan/

Stan Korea

Stan Ecosystem

CloudStan

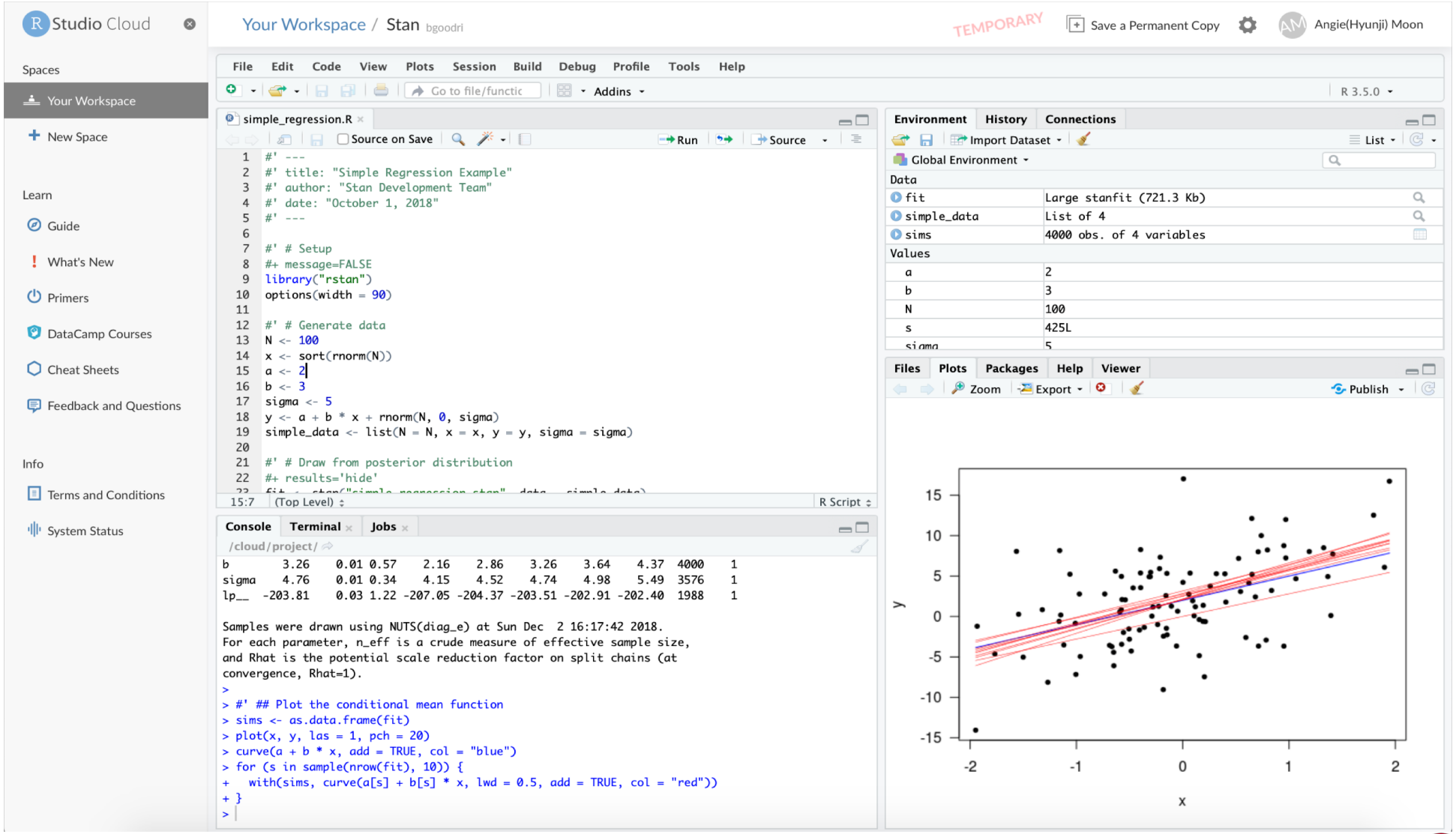
CloudStan is a client for Stan that runs from a browser. It will allow users to:

- connect to a remote server (where computation will be run)
- load, edit, and save Stan programs on that server
- instantiate a compiled Stan program with data that is either on the server or local
- run Stan
- track progress on the Stan program
- show output; plot
- ability to download results

https://github.com/stan-dev/stan/wiki/CloudStan-functional-specification

Stan Ecosystem

CloudStan



Stan Korea

Application: Time Series Forecasting

Prophet Model

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

growth	$g(t) = (k + \mathbf{a}(t)^\intercal \boldsymbol{\delta})t + (m + \mathbf{a}(t)^\intercal \boldsymbol{\gamma})$	$\delta_j = 0$ w.p. $\frac{T-S}{T}$, $\delta_j \sim \text{Laplace}(0, \lambda)$ w.p. $\frac{S}{T}$.
seasonality	$s(t) = X(t)\boldsymbol{\beta}$	$\boldsymbol{\beta} \sim \text{Normal}(0, \sigma^2)$
holiday	$h(t) = Z(t)\boldsymbol{\kappa}$	$\boldsymbol{\kappa} \sim \text{Normal}(0, \nu^2)$

Prophet is an additive regression model with four main components:

Growth: piecewise linear or logistic growth curve automatically detects changes in trends by selecting changepoints from the data

Seasonality: Fourier series + dummy variables

Holiday: user-provided list of important holidays

Application: Time Series Forecasting

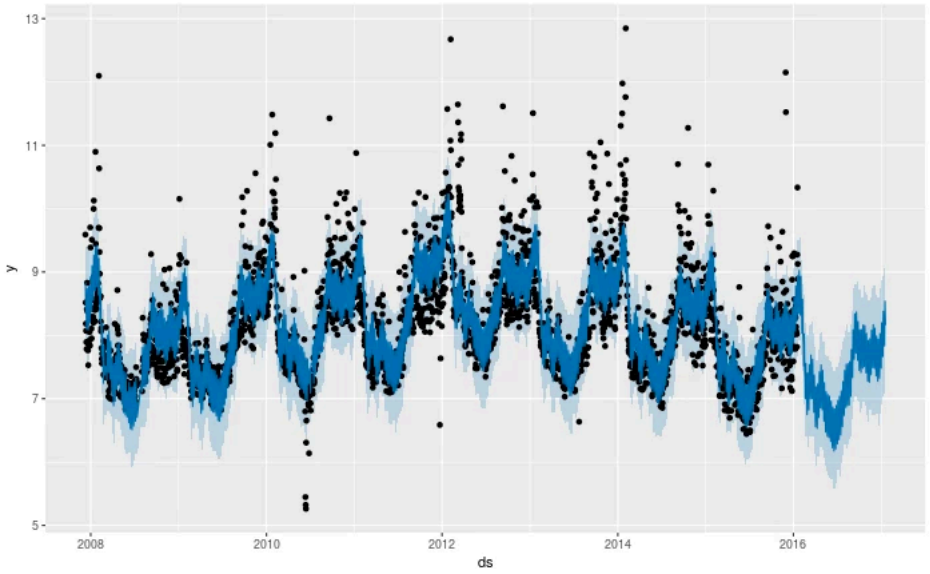
Prophet Model

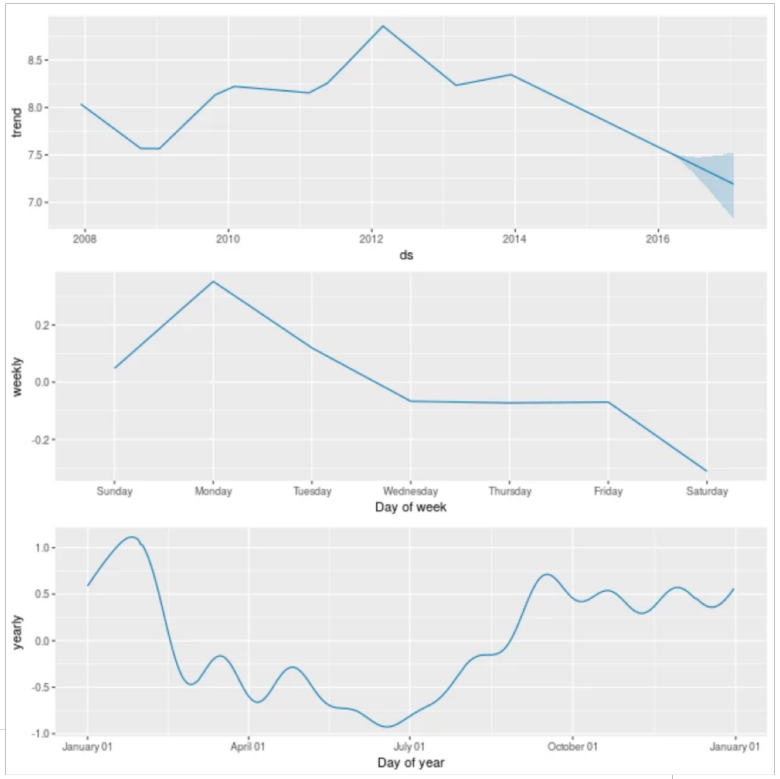
```
data {
  int T;                // Number of time periods
  int<lower=1> K;       // Number of regressors
  vector[T] t;          // Time
  vector[T] cap;        // Capacities for logistic trend
  vector[T] y;          // Time series
  int S;                // Number of changepoints
  vector[S] t_change;   // Times of trend changepoints
  matrix[T,K] X;        // Regressors
  vector[K] sigmas;     // Scale on seasonality prior
  real<lower=0> tau;    // Scale on changepoints prior
  int trend_indicator;  // 0 for linear, 1 for logistic
  vector[K] s_a;        // Indicator of additive features
  vector[K] s_m;        // Indicator of multiplicative features
}

transformed data {
  matrix[T, S] A;
  A = get_changepoint_matrix(t, t_change, T, S);
}

parameters {
  real k;                   // Base trend growth rate
  real m;                   // Trend offset
  vector[S] delta;          // Trend rate adjustments
  real<lower=0> sigma_obs;  // Observation noise
  vector[K] beta;           // Regressor coefficients
}
```

```
model {
  //priors
  k ~ normal(0, 5);
  m ~ normal(0, 5);
  delta ~ double_exponential(0, tau);
  sigma_obs ~ normal(0, 0.5);
  beta ~ normal(0, sigmas);

  // Likelihood
  if (trend_indicator == 0) {
    y ~ normal(
      linear_trend(k, m, delta, t, A, t_change)
      .* (1 + X * (beta .* s_m))
      + X * (beta .* s_a),
      sigma_obs
    );
  } else if (trend_indicator == 1) {
    y ~ normal(
      logistic_trend(k, m, delta, t, cap, A, t_change, S)
      .* (1 + X * (beta .* s_m))
      + X * (beta .* s_a),
      sigma_obs
    );
  }
}
```

Application: Time Series Forecasting

Prophet Model





Why we use Stan

Stan performs the MAP optimization for the Prophet model extremely quickly in practice (~1 second) and gives us the option to estimate parameter uncertainty using the Hamiltonian Monte Carlo algorithm. A prior version of Prophet we implemented with backfitting took much longer and did not provide uncertainty estimates. Since Stan has bindings for many languages, it allows us to re-use the model and fitting procedure to provide packages available in both R and Python based on the same Stan code.

Application: Time Series Forecasting

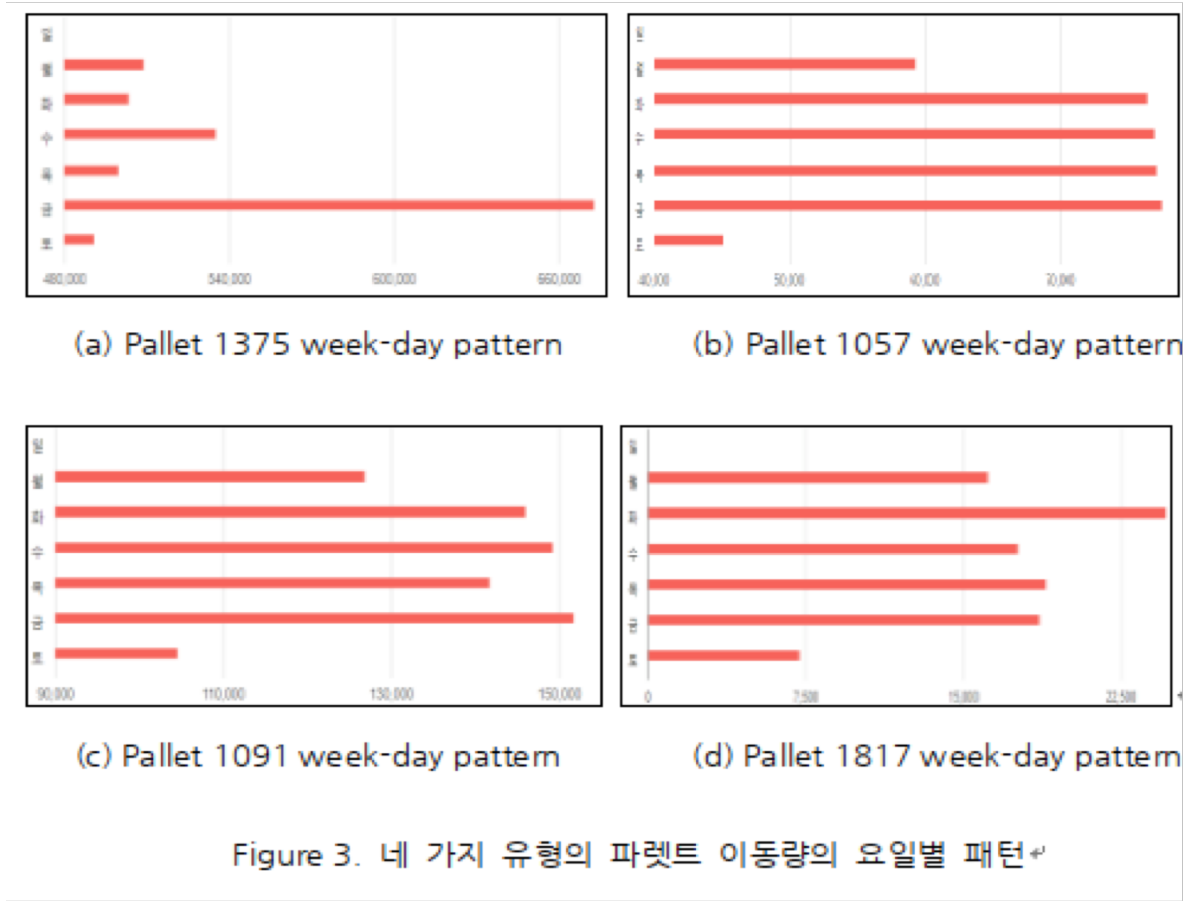
Time Unit Discretization

파렛트 유형에 따라 **요일별** 물동량 **패턴 상이**

이 group의 고유 패턴을 반영하는 예측모델이 필요

Time Unit Discretization 제안

1. Time Unit Clustering (TUC) Model
2. Time Unit Hierarchical (TUH) Model



(a) Pallet 1375 week-day pattern (b) Pallet 1057 week-day pattern

(c) Pallet 1091 week-day pattern (d) Pallet 1817 week-day pattern

Figure 3. 네 가지 유형의 파렛트 이동량의 요일별 패턴

Stan Korea

Application: Time Series Forecasting

TUC Model

Time Unit별 Clustering 진행 후, 각 Cluster마다 독립적으로 모수 학습 (cluster-wise prediction)

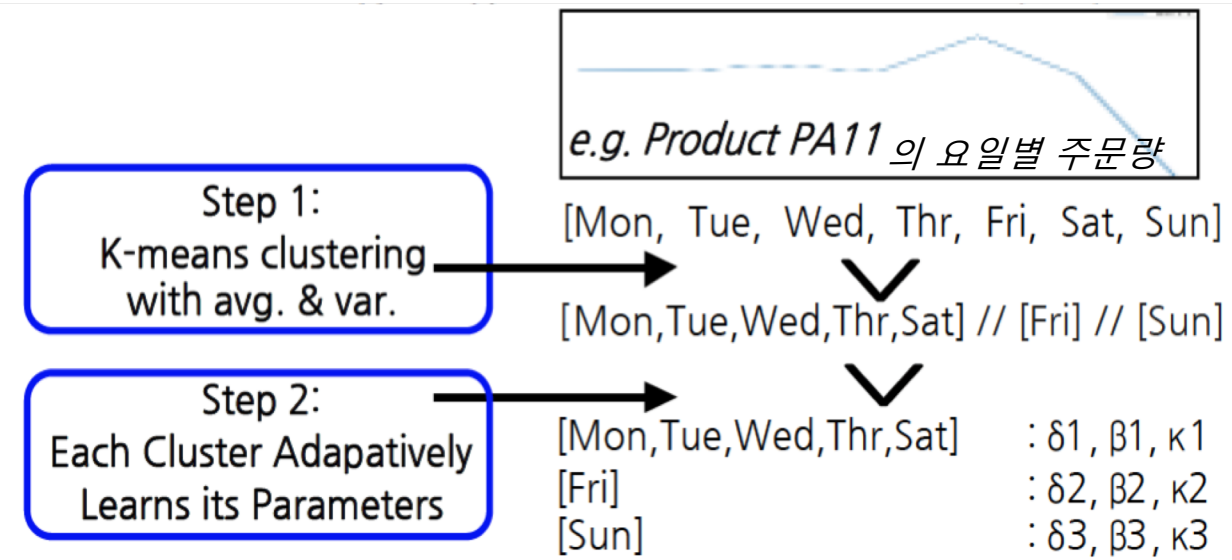


Table 4. 선택된 군집화 집합과 기존 프로펫 모델의 예측오차(MAE) 비교

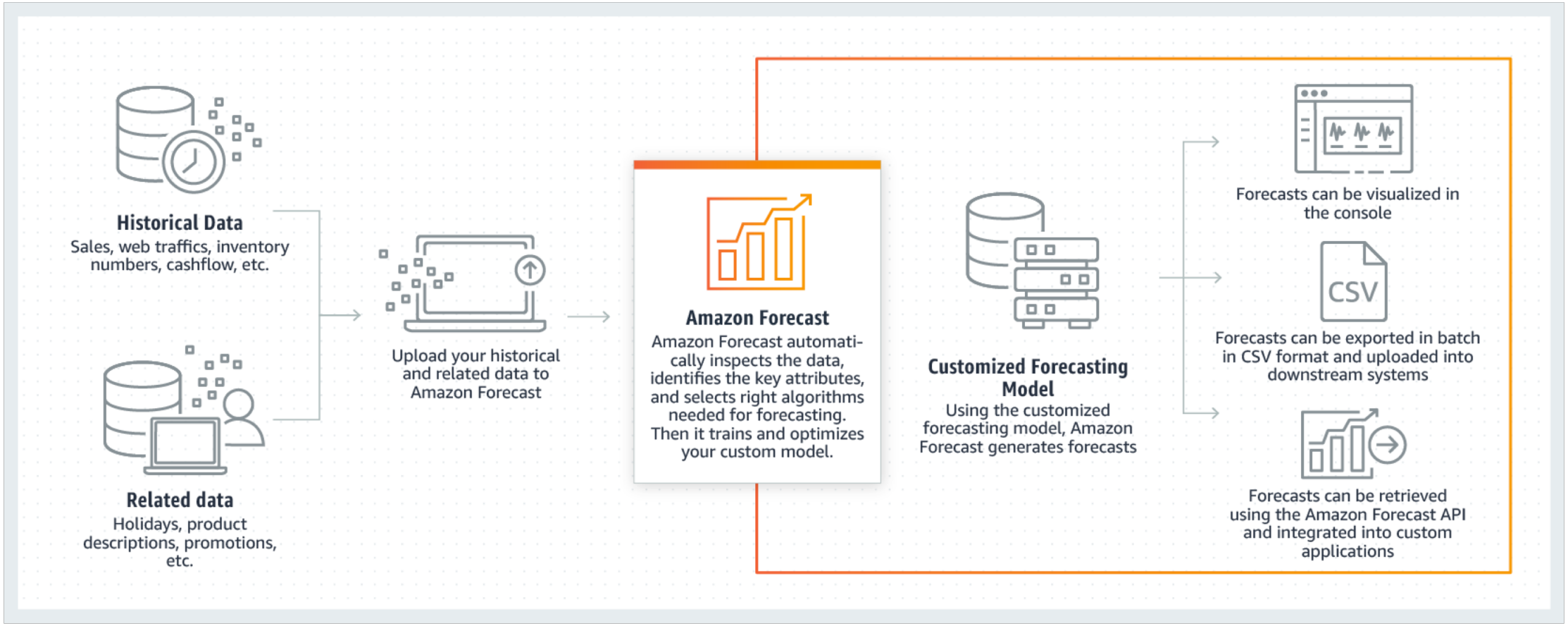
파렛트 종류	선택된 군집화 집합	프로펫 모델	시간단위 군집화 모델
1815	'0', '1234', '5', '6'	5,281	3,597
1041	'012345', '6'	1,056	1,054
1627	'05','134','2','6'	503	453

-> prophet model과 비교해 평균 10~20% 예측정확도 향상

Stan Korea

Application: Time Series Forecasting

Amazon Forecast



Use cases: Product Demand planning, Financial planning, Resource planning

Application: Time Series Forecasting

Amazon Forecast

Choosing an Amazon Forecast Recipe

Every Amazon Forecast predictor uses a recipe to train a model, then uses the model to make forecasts using an input dataset group. To help you get started, Amazon Forecast provides the following predefined recipes: ARIMA, DeepAR+, Exponential Smoothing (ETS), Mixture Density Network (MDN), Multiquantile Recurrent Neural Network (MQRNN), Non-Parametric Time Series (NPTS), Prophet, and Spline Quantile Forecaster (SQF).

Topics

- The ARIMA Recipe
- The DeepAR+ Recipe
- The Exponential Smoothing (ETS) Recipe
- The Mixture Density Networks (MDN) Recipe
- The Multi-Quantile Recurrent Neural Network (MQRNN) Recipe
- The Non-Parametric Time Series (NPTS) Recipe
- The Prophet Recipe
- The Spline Quantile Forecaster (SQF) Recipe

Predefined Dataset Domains and Dataset Types

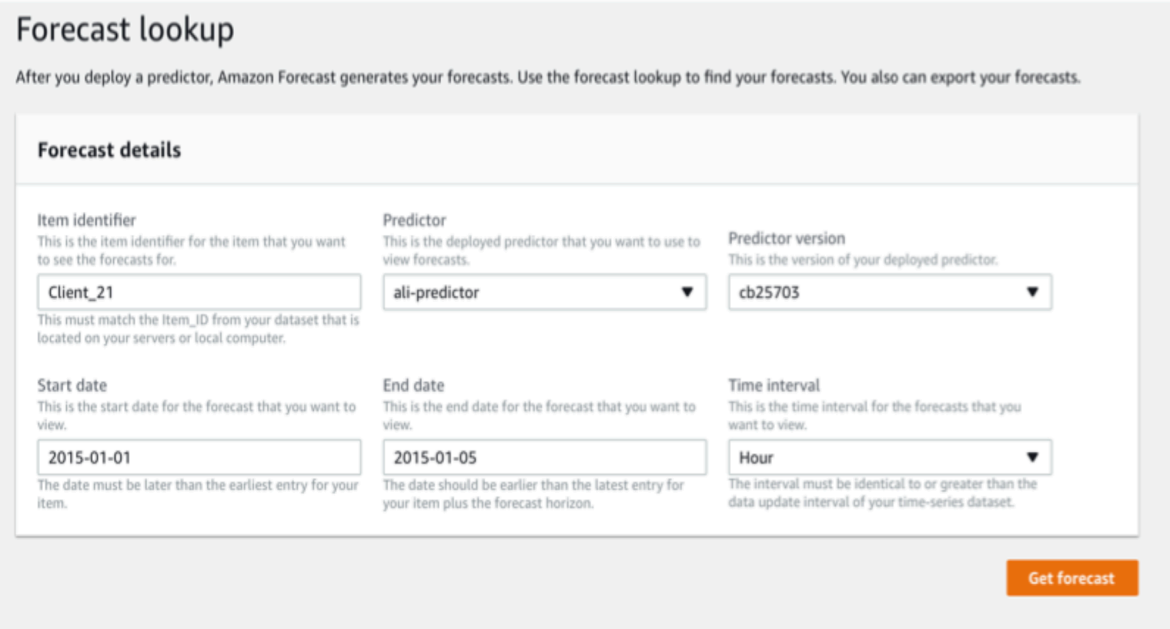
To train a predictor, you create one or more datasets, add them to a dataset group, and provide the dataset group for training. For each dataset that you create, you associate a dataset domain and one or more dataset types with it. A *domain* defines a forecasting use case. For each domain, there are one to three *dataset types*. The dataset type or types that you create for a domain is based on the type of data that you have and what you want to include in training.

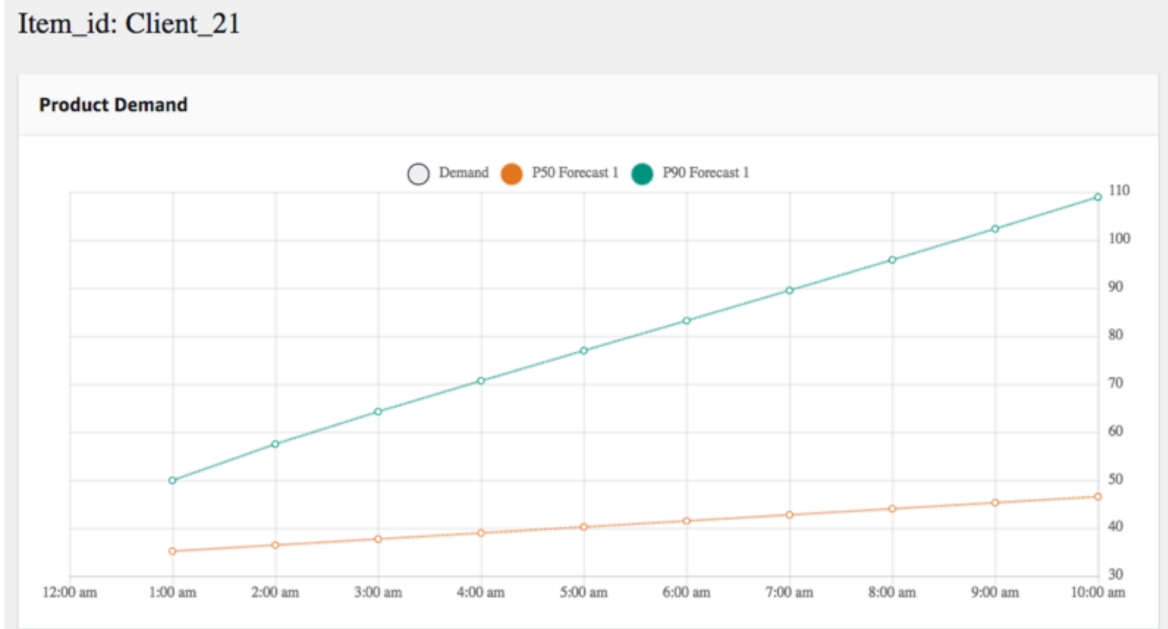
Amazon Forecast supports the following dataset domains:

- RETAIL Domain—For retail demand forecasting
- INVENTORY_PLANNING Domain—For supply chain and inventory planning
- EC2 CAPACITY Domain—For forecasting Amazon Elastic Compute Cloud (Amazon EC2) capacity
- WORK_FORCE Domain—For work force planning
- WEB_TRAFFIC Domain—For estimating future web traffic
- METRICS Domain—For forecasting metrics, such as revenue and cashflow
- CUSTOM Domain—For all other types of time-series forecasting

Application: Time Series Forecasting

Amazon Forecast





Interface example

Stan Korea

Stan Korea

Generative Statistical Modeler Program

Statistical Modeler

Generative Statistical Modeler

Generative Statistical Modeler who use engine

Generative Statistical Modeler who develop engine

Generative Statistical Modeler who develop language

Stan Korea

Generative Statistical Modeler Program

Generative Statistical Modeler를 위한 지식커뮤니티

사회과학 / 자연과학 / 공학 / 사업 등 전 분야에서 Stan 활용 중이거나 관심 있는 사람 누구나

활동 계획

- Stan Korea Meetup 정기적 개최 (2018.11.28 Stan Korea Meetup #1, 33명 참가)
- Stan 및 베이즈 통계 관련 논문 작성
- Stan 및 베이즈 통계 관련 교육자료 생산 (글, 영상, 번역 등)
- Stan 및 베이지안 스터디 운영

References

Bayesian

Gelman, A., Stern, H. S., Carlin, J. B., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). *Bayesian data analysis*. Chapman and Hall/CRC.

McElreath, R. (2016). *Statistical Rethinking: A Bayesian Course with Examples in R and Stan* (Vol. 122). CRC Press.

Stan

http://mc-stan.org/
https://andrewgelman.com/2017/05/31/compare-stan-pymc3-edward-hello-world/
http://mc-stan.org/rstan/articles/rstan.html
http://mc-stan.org/shinystan/
https://github.com/stan-dev/stan/wiki/CloudStan-functional-specification

Facebook Prophet

https://facebook.github.io/prophet/
https://research.fb.com/facebook-at-stancon-2018/

Amazon Forecast

https://aws.amazon.com/forecast/

Thank You.