Gaussian_Process_Code

Chiwan Kim

2/3/2020

##Part 1: Standard Gaussian Process

1-1: Fitting

```
library(rstan)
```

```
## Loading required package: StanHeaders

## Loading required package: ggplot2

## rstan (Version 2.19.2, GitRev: 2e1f913d3ca3)

## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)

## For improved execution time, we recommend calling
## Sys.setenv(LOCAL_CPPFLAGS = '-march=native')
## although this causes Stan to throw an error on a few processors.
```

```
source("gp.utility.R")

# Fitting GP model
stan_dat <- read_rdump('Financial_Data_Call_American.R')
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

## Loading required package: limSolve

##
## Attaching package: 'limSolve'

## The following object is masked from 'package:ggplot2':
##
##     resolution
```

```
## Loading required package: futile.logger

## Welcome to ragtop.  Logging can be enabled with commands such as
##   futile.logger::flog.threshold(futile.logger::INFO, name='ragtop.calibration')

## Parsed with column specification:
## cols(
##   .default = col_double(),
##   date = col_character(),
##   symbol = col_character(),
##   exdate = col_character(),
##   cp_flag = col_character(),
##   ticker = col_character(),
##   exercise_style = col_character()
## )

## See spec(...) for full column specifications.
```

```r
fit_gp_SGP_American <- stan(file="gp-fit-6dimension_withBS.stan", data=stan_dat,
               iter=100, chains=1);
```

```
## DIAGNOSTIC(S) FROM PARSER:
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line comments.
##
##
## SAMPLING FOR MODEL 'gp-fit-6dimension_withBS' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.017 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 170 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: WARNING: There aren't enough warmup iterations to fit the
## Chain 1:          three stages of adaptation as currently configured.
## Chain 1:          Reducing each adaptation stage to 15%/75%/10% of
## Chain 1:          the given number of warmup iterations:
## Chain 1:            init_buffer = 7
## Chain 1:            adapt_window = 38
## Chain 1:            term_buffer = 5
## Chain 1:
## Chain 1: Iteration:  1 / 100 [  1%]  (Warmup)
## Chain 1: Iteration: 10 / 100 [ 10%]  (Warmup)
## Chain 1: Iteration: 20 / 100 [ 20%]  (Warmup)
## Chain 1: Iteration: 30 / 100 [ 30%]  (Warmup)
## Chain 1: Iteration: 40 / 100 [ 40%]  (Warmup)
## Chain 1: Iteration: 50 / 100 [ 50%]  (Warmup)
## Chain 1: Iteration: 51 / 100 [ 51%]  (Sampling)
## Chain 1: Iteration: 60 / 100 [ 60%]  (Sampling)
## Chain 1: Iteration: 70 / 100 [ 70%]  (Sampling)
## Chain 1: Iteration: 80 / 100 [ 80%]  (Sampling)
## Chain 1: Iteration: 90 / 100 [ 90%]  (Sampling)
## Chain 1: Iteration: 100 / 100 [100%]  (Sampling)
## Chain 1:
```

```
## Chain 1:  Elapsed Time: 22.267 seconds (Warm-up)
## Chain 1:                18.185 seconds (Sampling)
## Chain 1:                40.452 seconds (Total)
## Chain 1:
```

```r
print(fit_gp_SGP_American, pars = c('theta','sigma2','gamma2'))
```

```
## Inference for Stan model: gp-fit-6dimension_withBS.
## 1 chains, each with iter=100; warmup=50; thin=1;
## post-warmup draws per chain=50, total post-warmup draws=50.
##
##           mean se_mean    sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## theta[1]  2.30    0.35  2.33  0.38  1.15  1.69  2.97  7.50    44 1.02
## theta[2]  6.72    0.23  2.11  3.39  5.11  6.38  8.00 10.73    85 1.02
## theta[3] 47.35    1.86 16.64 25.75 34.05 45.91 55.53 87.42    80 0.98
## theta[4]  0.56    0.02  0.18  0.27  0.46  0.54  0.64  0.94    56 0.99
## theta[5]  1.53    0.29  1.54  0.32  0.65  0.94  1.63  4.75    28 0.98
## theta[6]  0.28    0.01  0.11  0.13  0.21  0.27  0.32  0.54    51 0.98
## sigma2    0.02    0.00  0.00  0.01  0.02  0.02  0.02  0.02    44 1.01
## gamma2    6.75    0.29  2.46  3.34  4.81  6.52  8.20 12.14    72 0.99
##
## Samples were drawn using NUTS(diag_e) at Tue Mar 31 15:11:51 2020.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```r
sum_gp_SGP_American <- extract(fit_gp_SGP_American,permuted=FALSE)
```

```r
# Predicting from GP model
post_mean_theta_1_SGP <- mean(sum_gp_SGP_American[,1,1]) #theta
post_mean_theta_2_SGP <- mean(sum_gp_SGP_American[,1,2]) #theta
post_mean_theta_3_SGP <- mean(sum_gp_SGP_American[,1,3]) #theta
post_mean_theta_4_SGP <- mean(sum_gp_SGP_American[,1,4]) #theta
post_mean_theta_5_SGP <- mean(sum_gp_SGP_American[,1,5]) #theta
post_mean_theta_6_SGP <- mean(sum_gp_SGP_American[,1,6]) #theta
post_mean_sigma2_SGP <- mean(sum_gp_SGP_American[,1,7]) #sigma2
post_mean_gamma2_SGP <- mean(sum_gp_SGP_American[,1,8]) #gamma2
post_mean_mu_SGP <- stan_dat$blackscholes
```

```r
# test_start <- 323 #06/10 Puts
# test_end <- 559 #06/14 Puts

# test_start <- 560 #06/17 Puts
# test_end <- 852 #06/20 Puts

# test_start <- 269 #06/10 Calls
# test_end <- 432 #06/14 Calls

test_start <- 433 #06/17 Calls
test_end <- 700 #06/20 Calls

x2_bs <- cbind(as.numeric(stan_dat$total_calls_American$forward_price[test_start:test_end]),as.numeric(s
```

```r
library('qrmtools')
```

```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
library('ragtop')
blackscholes_test <- rep(NA,length(x2_bs[,1]))
for (row in 1:nrow(data.frame(x2_bs))){
  blackscholes_test[row] <- as.numeric(blackscholes(1,S0=as.numeric(stan_dat$total_calls_American$forwar
}

x.grid_1 <- as.numeric(stan_dat$total_calls_American$forward_price_scaled[test_start:test_end])
x.grid_2 <- as.numeric(stan_dat$total_calls_American$strike_price_scaled[test_start:test_end])
x.grid_3 <- as.numeric(stan_dat$total_calls_American$impl_volatility_scaled[test_start:test_end])
x.grid_4 <- as.numeric(stan_dat$total_calls_American$time_to_exp_scaled[test_start:test_end])
x.grid_5 <- as.numeric(stan_dat$total_calls_American$dividend_yield_scaled[test_start:test_end])
x.grid_6 <- as.numeric(stan_dat$total_calls_American$interest_rate_scaled[test_start:test_end])
x2 <- cbind(x.grid_1,x.grid_2,x.grid_3,x.grid_4,x.grid_5,x.grid_6)
```

1-2: Predictions

```r
post_data_SGP_American <- list(theta=c(post_mean_theta_1_SGP,post_mean_theta_2_SGP,post_mean_theta_3_SGP
# post_data

pred_gp_SGP <- stan(file="Predictive GP_6dimension_withBS.stan", data=post_data_SGP_American,iter=200, v
```

```
## DIAGNOSTIC(S) FROM PARSER:
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line comments.
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line comments.
##
##
## SAMPLING FOR MODEL 'Predictive GP_6dimension_withBS' NOW (CHAIN 1).
## Chain 1: Iteration:   1 / 200 [  0%]  (Sampling)
## Chain 1: Iteration: 100 / 200 [ 50%]  (Sampling)
## Chain 1: Iteration: 200 / 200 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                19.529 seconds (Sampling)
## Chain 1:                19.529 seconds (Total)
## Chain 1:
```

##Part2: Bdrycov Gaussian Process

2-1: Fitting

```r
# Fitting GP model for Bdrycov
fit_gp_Bdrycov_American <- stan(file="gp-fit-6dimension_withBS_Bdrycov.stan", data=stan_dat,
               iter=100, chains=1);
```

```
##
## SAMPLING FOR MODEL 'gp-fit-6dimension_withBS_Bdrycov' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.102 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1020 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: WARNING: There aren't enough warmup iterations to fit the
## Chain 1:          three stages of adaptation as currently configured.
## Chain 1:          Reducing each adaptation stage to 15%/75%/10% of
## Chain 1:          the given number of warmup iterations:
## Chain 1:            init_buffer = 7
## Chain 1:            adapt_window = 38
## Chain 1:            term_buffer = 5
## Chain 1:
## Chain 1: Iteration:  1 / 100 [  1%]  (Warmup)
## Chain 1: Iteration: 10 / 100 [ 10%]  (Warmup)
## Chain 1: Iteration: 20 / 100 [ 20%]  (Warmup)
## Chain 1: Iteration: 30 / 100 [ 30%]  (Warmup)
## Chain 1: Iteration: 40 / 100 [ 40%]  (Warmup)
## Chain 1: Iteration: 50 / 100 [ 50%]  (Warmup)
## Chain 1: Iteration: 51 / 100 [ 51%]  (Sampling)
## Chain 1: Iteration: 60 / 100 [ 60%]  (Sampling)
## Chain 1: Iteration: 70 / 100 [ 70%]  (Sampling)
## Chain 1: Iteration: 80 / 100 [ 80%]  (Sampling)
## Chain 1: Iteration: 90 / 100 [ 90%]  (Sampling)
## Chain 1: Iteration: 100 / 100 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 87.138 seconds (Warm-up)
## Chain 1:                87.126 seconds (Sampling)
## Chain 1:                174.264 seconds (Total)
## Chain 1:
```

```r
print(fit_gp_Bdrycov_American, pars = c('theta','sigma2','gamma2'))
```

```
## Inference for Stan model: gp-fit-6dimension_withBS_Bdrycov.
## 1 chains, each with iter=100; warmup=50; thin=1;
## post-warmup draws per chain=50, total post-warmup draws=50.
##
##           mean se_mean     sd 2.5%  25%   50%   75%  97.5% n_eff Rhat
## theta[1]  5.16    1.56   8.31 0.24 0.86  1.63  4.63  25.97    28 1.11
## theta[2] 10.66    7.46  42.58 0.24 0.74  1.59  3.04  54.06    33 1.01
## theta[3]  2.85    0.68   4.08 0.26 0.65  1.19  2.90  13.51    36 1.03
## theta[4] 28.25   16.44  74.86 0.17 1.13  2.84 11.52 171.66    21 0.99
## theta[5]  5.49    2.90  11.14 0.25 0.64  1.25  3.65  45.10    15 1.06
## theta[6] 31.18   26.01 168.86 0.34 0.49  1.50  3.10 144.54    42 1.01
## sigma2    8.37    1.57   4.84 0.51 2.90 10.17 12.43  14.47    10 1.09
## gamma2    4.68    1.59   4.83 0.38 0.99  1.69 10.54  13.30     9 1.10
```

```
##
## Samples were drawn using NUTS(diag_e) at Tue Mar 31 15:17:38 2020.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
sum_gp_Bdrycov_American <- extract(fit_gp_Bdrycov_American,permuted=FALSE)
# saveRDS(fit_gp,file ="fit_gp_vol50_within50spot_7to19days")
```

```
# Predicting from GP model - 2 dimensional case
post_mean_theta_1_Bdrycov <- mean(sum_gp_Bdrycov_American[,1,1]) #theta
post_mean_theta_2_Bdrycov <- mean(sum_gp_Bdrycov_American[,1,2]) #theta
post_mean_theta_3_Bdrycov <- mean(sum_gp_Bdrycov_American[,1,3]) #theta
post_mean_theta_4_Bdrycov <- mean(sum_gp_Bdrycov_American[,1,4]) #theta
post_mean_theta_5_Bdrycov <- mean(sum_gp_Bdrycov_American[,1,5]) #theta
post_mean_theta_6_Bdrycov <- mean(sum_gp_Bdrycov_American[,1,6]) #theta
post_mean_sigma2_Bdrycov <- mean(sum_gp_Bdrycov_American[,1,7]) #sigma2
post_mean_gamma2_Bdrycov <- mean(sum_gp_Bdrycov_American[,1,8]) #gamma2
post_mean_mu_Bdrycov <- stan_dat$blackscholes
```

2-2: Predictions

```
# X.grid <- expand.grid(x1 = x.grid_1, x2 = x.grid_2)

post_data_Bdrycov_American <- list(theta=c(post_mean_theta_1_Bdrycov,post_mean_theta_2_Bdrycov,post_mean
# post_data

pred_gp_Bdrycov <- stan(file="Predictive GP_6dimension_withBS_Bdrycov.stan", data=post_data_Bdrycov_Amer
```

```
## DIAGNOSTIC(S) FROM PARSER:
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line comments.
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line comments.
##
##
## SAMPLING FOR MODEL 'Predictive GP_6dimension_withBS_Bdrycov' NOW (CHAIN 1).
## Chain 1: Iteration:   1 / 200 [  0%]  (Sampling)
## Chain 1: Iteration: 100 / 200 [ 50%]  (Sampling)
## Chain 1: Iteration: 200 / 200 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                28.396 seconds (Sampling)
## Chain 1:                28.396 seconds (Total)
## Chain 1:
```

##Part 3 Predictions Versus Truth

3-1: Computing Means Standard GP

```
#Computing Mean
y_predict_values_SGP <- extract(pred_gp_SGP,permuted=FALSE)
y_mean_values_SGP <- c(colMeans(y_predict_values_SGP))
y_mean_values_SGP <- y_mean_values_SGP[1:(length(y_mean_values_SGP)-1)]
```

```
#Computing Standard Deviation
pred_gp_summary_SGP <- summary(pred_gp_SGP, sd=c("sd"))$summary
pred_gp_sd_SGP <- pred_gp_summary_SGP[, c("sd")]
y_sd_values_SGP <- pred_gp_sd_SGP[1:(length(pred_gp_sd_SGP)-1)]
```

3-2: Computing Means Bdrycov

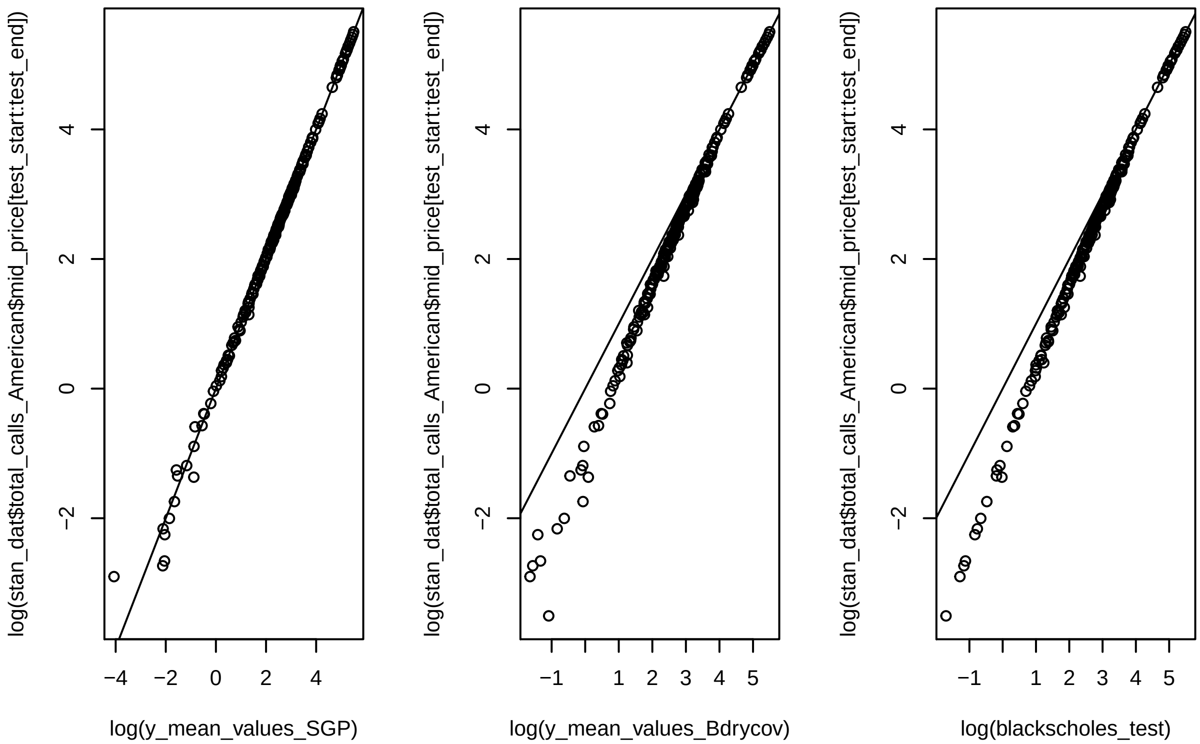
```
#Computing Mean
y_predict_values_Bdrycov <- extract(pred_gp_Bdrycov,permuted=FALSE)
y_mean_values_Bdrycov <- c(colMeans(y_predict_values_Bdrycov))
y_mean_values_Bdrycov <- y_mean_values_Bdrycov[1:(length(y_mean_values_Bdrycov)-1)]

#Computing Standard Deviation
pred_gp_summary_Bdrycov <- summary(pred_gp_Bdrycov, sd=c("sd"))$summary
pred_gp_sd_Bdrycov <- pred_gp_summary_Bdrycov[, c("sd")]
y_sd_values_Bdrycov <- pred_gp_sd_Bdrycov[1:(length(pred_gp_sd_Bdrycov)-1)]
```

3-3: Plotting Predicted Values against Truth

```
par(mfrow=c(1,3))
#Plotting Standard GP
plot(log(y_mean_values_SGP),log(stan_dat$total_calls_American$mid_price[test_start:test_end]),xlim = c(
```

```
## Warning in log(y_mean_values_SGP): NaNs produced

## Warning in log(y_mean_values_SGP): NaNs produced

## Warning in log(y_mean_values_SGP): NaNs produced
```

```
abline(0,1)

#Plotting BDrycov
plot(log(y_mean_values_Bdrycov),log(stan_dat$total_calls_American$mid_price[test_start:test_end]), xlim
abline(0,1)

#Plotting Blackscholes
plot(log(blackscholes_test),log(stan_dat$total_calls_American$mid_price[test_start:test_end]), xlim = c
abline(0,1)
```




```r
#MSE
library('MLmetrics')
```

```
##
## Attaching package: 'MLmetrics'

## The following object is masked from 'package:base':
##
##     Recall
```

```r
MSE(y_mean_values_SGP,stan_dat$total_calls_American$mid_price[test_start:test_end])
```

```
## [1] 0.2722204
```

```r
MSE(y_mean_values_Bdrycov,stan_dat$total_calls_American$mid_price[test_start:test_end])
```

```
## [1] 10.23453
```

```r
MSE(blackscholes_test,stan_dat$total_calls_American$mid_price[test_start:test_end])
```

```
## [1] 10.12211
```

##Part 4 Visualizations

4-1: Contour Plots of Forward Price & Strike Price

```
x.grid_1_cont <- as.numeric(stan_dat$total_calls_American$forward_price_scaled[test_start:test_end])
x.grid_2_cont <- as.numeric(stan_dat$total_calls_American$strike_price_scaled[test_start:test_end])

dim1 <- seq(min(x.grid_1_cont),max(x.grid_1_cont),length.out = 25)
dim2 <- seq(min(x.grid_2_cont),max(x.grid_2_cont),length.out = 25)
X.grid <- expand.grid(x1 = dim1, x2 = dim2)

x.grid_3_cont <- as.numeric(rep(mean(stan_dat$total_calls_American$impl_volatility_scaled[test_start:tes
x.grid_4_cont <- as.numeric(rep(mean(stan_dat$total_calls_American$time_to_exp_scaled[test_start:test_en
x.grid_5_cont <- as.numeric(rep(mean(stan_dat$total_calls_American$dividend_yield_scaled[test_start:test
x.grid_6_cont <- as.numeric(rep(mean(stan_dat$total_calls_American$interest_rate_scaled[test_start:test_

x2_cont <- cbind(X.grid,x.grid_3_cont,x.grid_4_cont,x.grid_5_cont,x.grid_6_cont)

x.grid_1_cont_bs <- as.numeric(stan_dat$total_calls_American$forward_price[test_start:test_end])
x.grid_2_cont_bs <- as.numeric(stan_dat$total_calls_American$strike_price[test_start:test_end])
x.grid_3_cont_bs <- as.numeric(rep(mean(stan_dat$total_calls_American$impl_volatility[test_start:test_en
x.grid_4_cont_bs <- as.numeric(rep(mean(stan_dat$total_calls_American$time_to_exp[test_start:test_end]))
x.grid_5_cont_bs <- as.numeric(rep(mean(stan_dat$total_calls_American$dividend_yield[test_start:test_end
x.grid_6_cont_bs <- as.numeric(rep(mean(stan_dat$total_calls_American$interest_rate[test_start:test_end]

dim1_bs <- seq(min(x.grid_1_cont_bs),max(x.grid_1_cont_bs),length.out = 25)
dim2_bs <- seq(min(x.grid_2_cont_bs),max(x.grid_2_cont_bs),length.out = 25)
X.grid_bs <- expand.grid(x1 = dim1_bs, x2 = dim2_bs)

x2_cont_bs <- cbind(X.grid_bs,x.grid_3_cont_bs,x.grid_4_cont_bs,x.grid_5_cont_bs,x.grid_6_cont_bs)

blackscholes_test_cont <- rep(NA,length(x2_cont_bs[,1]))
for (row in 1:nrow(data.frame(x2_cont_bs))){
  blackscholes_test_cont[row] <- as.numeric(blackscholes(1,S0=x2_cont_bs[row,1],K=x2_cont_bs[row,2],r=x2
}

post_data_cont_SGP <- list(theta=c(post_mean_theta_1_SGP,post_mean_theta_2_SGP,post_mean_theta_3_SGP,pos

post_data_cont_Bdrycov <- list(theta=c(post_mean_theta_1_Bdrycov,post_mean_theta_2_Bdrycov,post_mean_the


# post_data

pred_gp_cont_SGP <- stan(file="Predictive GP_6dimension_withBS.stan", data=post_data_cont_SGP,iter=200,
```

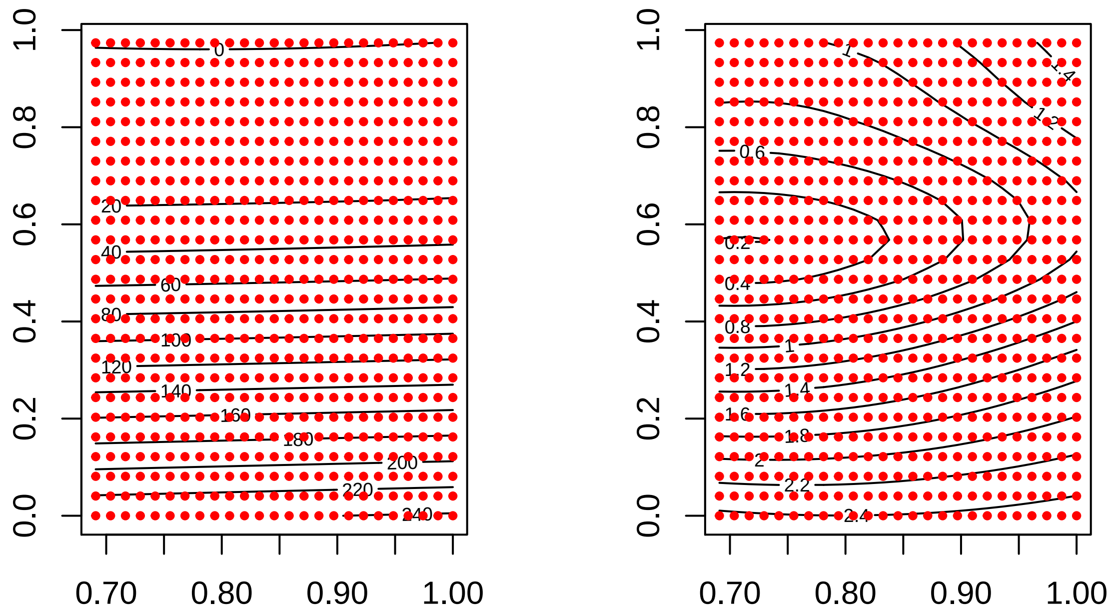
```
## DIAGNOSTIC(S) FROM PARSER:
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line comments.
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line comments.
##
##
## SAMPLING FOR MODEL 'Predictive GP_6dimension_withBS' NOW (CHAIN 1).
## Chain 1: Iteration:   1 / 200 [  0%]  (Sampling)
## Chain 1: Iteration: 100 / 200 [ 50%]  (Sampling)
## Chain 1: Iteration: 200 / 200 [100%]  (Sampling)
```

```
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                54.442 seconds (Sampling)
## Chain 1:                54.442 seconds (Total)
## Chain 1:
```

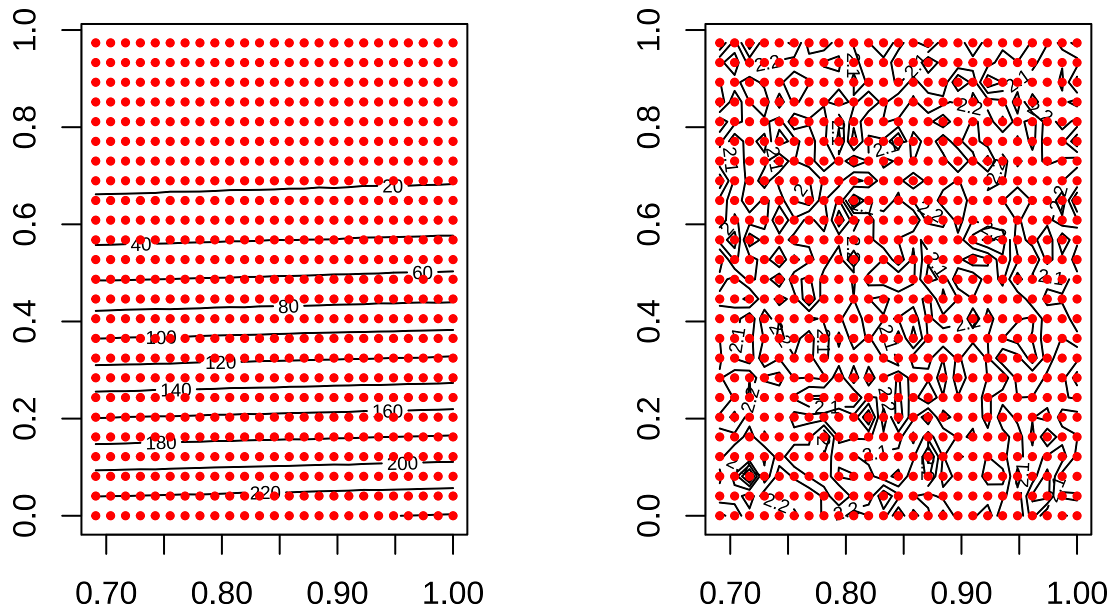
```r
pred_gp_cont_Bdrycov <- stan(file="Predictive GP_6dimension_withBS_Bdrycov.stan", data=post_data_cont_Bc
```

```
## DIAGNOSTIC(S) FROM PARSER:
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line comments.
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line comments.
##
##
## SAMPLING FOR MODEL 'Predictive GP_6dimension_withBS_Bdrycov' NOW (CHAIN 1).
## Chain 1: Iteration:   1 / 200 [  0%]  (Sampling)
## Chain 1: Iteration: 100 / 200 [ 50%]  (Sampling)
## Chain 1: Iteration: 200 / 200 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                102.114 seconds (Sampling)
## Chain 1:                102.114 seconds (Total)
## Chain 1:
```

```r
#Computing Mean
y_predict_values_cont_SGP <- extract(pred_gp_cont_SGP,permuted=FALSE)
y_mean_values_cont_SGP <- c(colMeans(y_predict_values_cont_SGP))
y_mean_values_cont_SGP <- y_mean_values_cont_SGP[1:(length(y_mean_values_cont_SGP)-1)]

#Computing Standard Deviation
pred_gp_summary_cont_SGP <- summary(pred_gp_cont_SGP, sd=c("sd"))$summary
pred_gp_sd_cont_SGP <- pred_gp_summary_cont_SGP[, c("sd")]
y_sd_values_cont_SGP <- pred_gp_sd_cont_SGP[1:(length(pred_gp_sd_cont_SGP)-1)]

par(mfrow = c(1, 2))
#Contour for Predictions aka mean values of predicitons
contour(dim1, dim2, matrix(y_mean_values_cont_SGP, length(dim1), length(dim2)))
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
#Contour of Variance
contour(dim1, dim2, matrix(y_sd_values_cont_SGP, length(dim1), length(dim2)))
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
```



```
#Computing Mean
y_predict_values_cont_Bdrycov <- extract(pred_gp_cont_Bdrycov,permuted=FALSE)
y_mean_values_cont_Bdrycov <- c(colMeans(y_predict_values_cont_Bdrycov))
y_mean_values_cont_Bdrycov <- y_mean_values_cont_Bdrycov[1:(length(y_mean_values_cont_Bdrycov)-1)]

#Computing Standard Deviation
pred_gp_summary_cont_Bdrycov <- summary(pred_gp_cont_Bdrycov, sd=c("sd"))$summary
pred_gp_sd_cont_Bdrycov <- pred_gp_summary_cont_Bdrycov[, c("sd")]
y_sd_values_cont_Bdrycov <- pred_gp_sd_cont_Bdrycov[1:(length(pred_gp_sd_cont_Bdrycov)-1)]

par(mfrow = c(1, 2))
#Contour for Predictions aka mean values of predicitons
contour(dim1, dim2, matrix(y_mean_values_cont_Bdrycov, length(dim1), length(dim2)))
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
#Contour of Variance
contour(dim1, dim2, matrix(y_sd_values_cont_Bdrycov, length(dim1), length(dim2)))
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
```



4-2: Contour Plots of Implied VOlatility & Time to Expiration

```
x.grid_1_cont <- as.numeric(stan_dat$total_calls_American$impl_volatility_scaled[test_start:test_end])
x.grid_2_cont <- as.numeric(stan_dat$total_calls_American$time_to_exp_scaled[test_start:test_end])

dim1 <- seq(min(x.grid_1_cont),max(x.grid_1_cont),length.out = 25)
dim2 <- seq(min(x.grid_2_cont),max(x.grid_2_cont),length.out = 25)
X.grid <- expand.grid(x1 = dim1, x2 = dim2)

x.grid_3_cont <- as.numeric(rep(mean(stan_dat$total_calls_American$forward_price_scaled[test_start:test_
x.grid_4_cont <- as.numeric(rep(mean(stan_dat$total_calls_American$strike_price_scaled[test_start:test_e
x.grid_5_cont <- as.numeric(rep(mean(stan_dat$total_calls_American$dividend_yield_scaled[test_start:tes
x.grid_6_cont <- as.numeric(rep(mean(stan_dat$total_calls_American$interest_rate_scaled[test_start:test_

x2_cont <- cbind(X.grid,x.grid_3_cont,x.grid_4_cont,x.grid_5_cont,x.grid_6_cont)

x.grid_1_cont_bs <- as.numeric(stan_dat$total_calls_American$impl_volatility[test_start:test_end])
x.grid_2_cont_bs <- as.numeric(stan_dat$total_calls_American$time_to_exp[test_start:test_end])
x.grid_3_cont_bs <- as.numeric(rep(mean(stan_dat$total_calls_American$forward_price[test_start:test_end]
x.grid_4_cont_bs <- as.numeric(rep(mean(stan_dat$total_calls_American$strike_price[test_start:test_end])
x.grid_5_cont_bs <- as.numeric(rep(mean(stan_dat$total_calls_American$dividend_yield[test_start:test_en
x.grid_6_cont_bs <- as.numeric(rep(mean(stan_dat$total_calls_American$interest_rate[test_start:test_end]

dim1_bs <- seq(min(x.grid_1_cont_bs),max(x.grid_1_cont_bs),length.out = 25)
dim2_bs <- seq(min(x.grid_2_cont_bs),max(x.grid_2_cont_bs),length.out = 25)
X.grid_bs <- expand.grid(x1 = dim1_bs, x2 = dim2_bs)
```

```r
x2_cont_bs <- cbind(X.grid_bs,x.grid_3_cont_bs,x.grid_4_cont_bs,x.grid_5_cont_bs,x.grid_6_cont_bs)

blackscholes_test_cont <- rep(NA,length(x2_cont_bs[,1]))
for (row in 1:nrow(data.frame(x2_cont_bs))){
  blackscholes_test_cont[row] <- as.numeric(blackscholes(1,S0=x2_cont_bs[row,3],K=x2_cont_bs[row,4],r=x2
}


post_data_cont_SGP <- list(theta=c(post_mean_theta_1_SGP,post_mean_theta_2_SGP,post_mean_theta_3_SGP,pos

post_data_cont_Bdrycov <- list(theta=c(post_mean_theta_1_Bdrycov,post_mean_theta_2_Bdrycov,post_mean_the


# post_data

pred_gp_cont_SGP <- stan(file="Predictive GP_6dimension_withBS.stan", data=post_data_cont_SGP,iter=200,
```

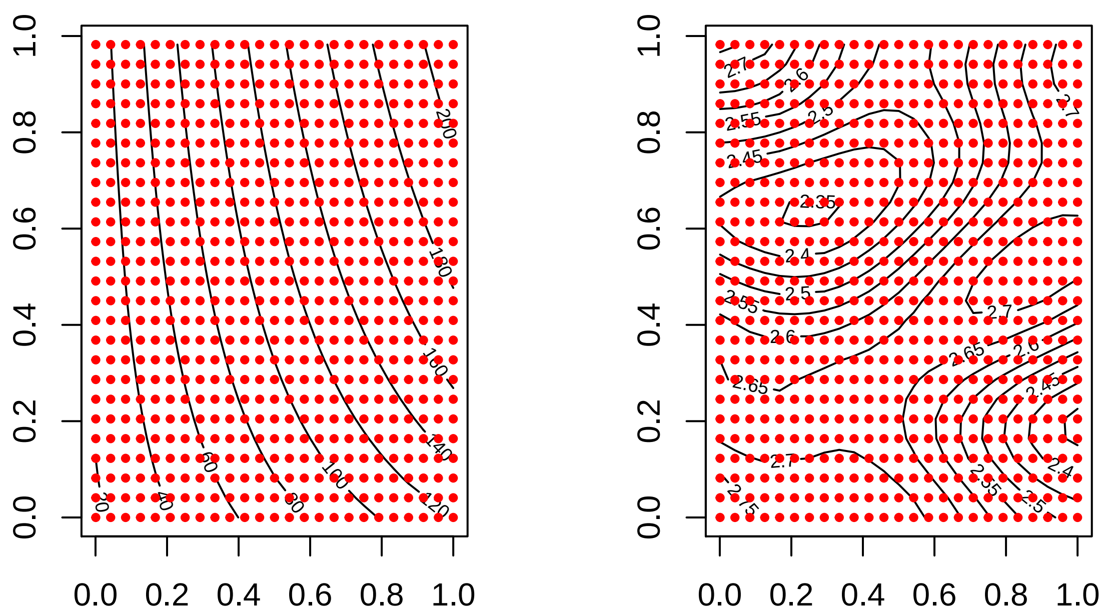
```
## DIAGNOSTIC(S) FROM PARSER:
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line comments.
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line comments.
##
##
## SAMPLING FOR MODEL 'Predictive GP_6dimension_withBS' NOW (CHAIN 1).
## Chain 1: Iteration:   1 / 200 [  0%]  (Sampling)
## Chain 1: Iteration: 100 / 200 [ 50%]  (Sampling)
## Chain 1: Iteration: 200 / 200 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                51.194 seconds (Sampling)
## Chain 1:                51.194 seconds (Total)
## Chain 1:
```

```r
pred_gp_cont_Bdrycov <- stan(file="Predictive GP_6dimension_withBS_Bdrycov.stan", data=post_data_cont_Bd
```

```
## DIAGNOSTIC(S) FROM PARSER:
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line comments.
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line comments.
##
##
## SAMPLING FOR MODEL 'Predictive GP_6dimension_withBS_Bdrycov' NOW (CHAIN 1).
## Chain 1: Iteration:   1 / 200 [  0%]  (Sampling)
## Chain 1: Iteration: 100 / 200 [ 50%]  (Sampling)
## Chain 1: Iteration: 200 / 200 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                87.024 seconds (Sampling)
## Chain 1:                87.024 seconds (Total)
## Chain 1:
```

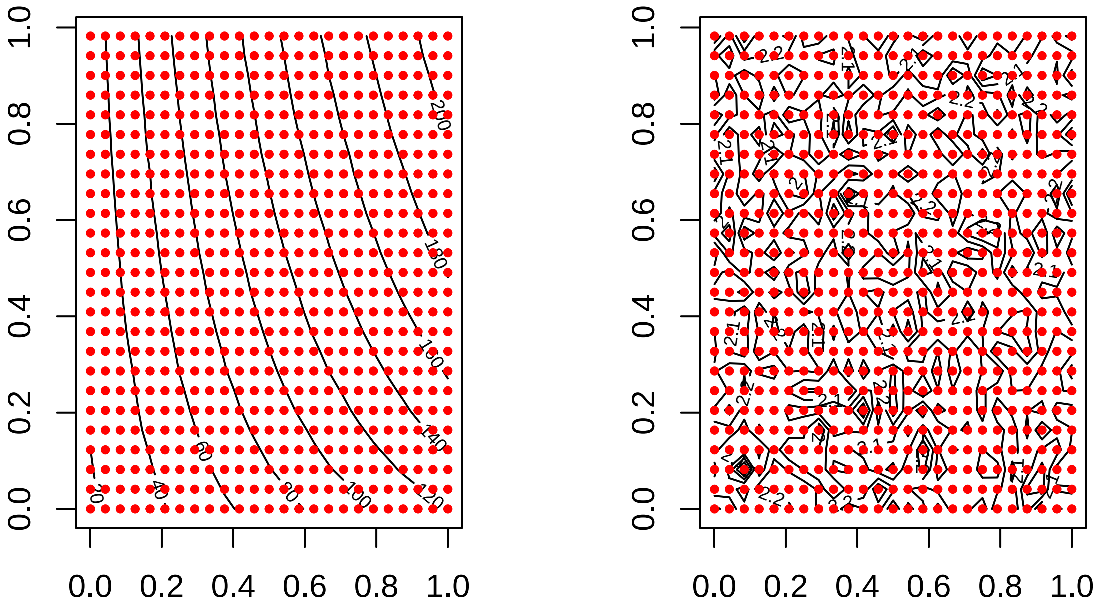
```r
#Computing Mean
y_predict_values_cont_SGP <- extract(pred_gp_cont_SGP,permuted=FALSE)
y_mean_values_cont_SGP <- c(colMeans(y_predict_values_cont_SGP))
```

```
y_mean_values_cont_SGP <- y_mean_values_cont_SGP[1:(length(y_mean_values_cont_SGP)-1)]

#Computing Standard Deviation
pred_gp_summary_cont_SGP <- summary(pred_gp_cont_SGP, sd=c("sd"))$summary
pred_gp_sd_cont_SGP <- pred_gp_summary_cont_SGP[, c("sd")]
y_sd_values_cont_SGP <- pred_gp_sd_cont_SGP[1:(length(pred_gp_sd_cont_SGP)-1)]

par(mfrow = c(1, 2))
#Contour for Predictions aka mean values of predicitons
contour(dim1, dim2, matrix(y_mean_values_cont_SGP, length(dim1), length(dim2)))
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
#Contour of Variance
contour(dim1, dim2, matrix(y_sd_values_cont_SGP, length(dim1), length(dim2)))
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
```




```
#Computing Mean
y_predict_values_cont_Bdrycov <- extract(pred_gp_cont_Bdrycov,permuted=FALSE)
y_mean_values_cont_Bdrycov <- c(colMeans(y_predict_values_cont_Bdrycov))
y_mean_values_cont_Bdrycov <- y_mean_values_cont_Bdrycov[1:(length(y_mean_values_cont_Bdrycov)-1)]

#Computing Standard Deviation
pred_gp_summary_cont_Bdrycov <- summary(pred_gp_cont_Bdrycov, sd=c("sd"))$summary
pred_gp_sd_cont_Bdrycov <- pred_gp_summary_cont_Bdrycov[, c("sd")]
y_sd_values_cont_Bdrycov <- pred_gp_sd_cont_Bdrycov[1:(length(pred_gp_sd_cont_Bdrycov)-1)]
```

```r
par(mfrow = c(1, 2))
#Contour for Predictions aka mean values of predicitons
contour(dim1, dim2, matrix(y_mean_values_cont_Bdrycov, length(dim1), length(dim2)))
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
#Contour of Variance
contour(dim1, dim2, matrix(y_sd_values_cont_Bdrycov, length(dim1), length(dim2)))
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
```



4-3: Contour Plots of Interest Rate & Dividend Yield

```r
x.grid_1_cont <- as.numeric(stan_dat$total_calls_American$dividend_yield_scaled[test_start:test_end])
x.grid_2_cont <- as.numeric(stan_dat$total_calls_American$interest_rate_scaled[test_start:test_end])

dim1 <- seq(min(x.grid_1_cont),max(x.grid_1_cont),length.out = 25)
dim2 <- seq(min(x.grid_2_cont),max(x.grid_2_cont),length.out = 25)
X.grid <- expand.grid(x1 = dim1, x2 = dim2)

x.grid_3_cont <- as.numeric(rep(mean(stan_dat$total_calls_American$impl_volatility_scaled[test_start:tes
x.grid_4_cont <- as.numeric(rep(mean(stan_dat$total_calls_American$time_to_exp_scaled[test_start:test_en
x.grid_5_cont <- as.numeric(rep(mean(stan_dat$total_calls_American$forward_price_scaled[test_start:test_
x.grid_6_cont <- as.numeric(rep(mean(stan_dat$total_calls_American$strike_price_scaled[test_start:test_e

x2_cont <- cbind(X.grid,x.grid_3_cont,x.grid_4_cont,x.grid_5_cont,x.grid_6_cont)

x.grid_1_cont_bs <- as.numeric(stan_dat$total_calls_American$dividend_yield[test_start:test_end])
x.grid_2_cont_bs <- as.numeric(stan_dat$total_calls_American$interest_rate[test_start:test_end])
```

```r
x.grid_3_cont_bs <- as.numeric(rep(mean(stan_dat$total_calls_American$impl_volatility[test_start:test_en
x.grid_4_cont_bs <- as.numeric(rep(mean(stan_dat$total_calls_American$time_to_exp[test_start:test_end])
x.grid_5_cont_bs <- as.numeric(rep(mean(stan_dat$total_calls_American$forward_price[test_start:test_end
x.grid_6_cont_bs <- as.numeric(rep(mean(stan_dat$total_calls_American$strike_price[test_start:test_end]

dim1_bs <- seq(min(x.grid_1_cont_bs),max(x.grid_1_cont_bs),length.out = 25)
dim2_bs <- seq(min(x.grid_2_cont_bs),max(x.grid_2_cont_bs),length.out = 25)
X.grid_bs <- expand.grid(x1 = dim1_bs, x2 = dim2_bs)

x2_cont_bs <- cbind(X.grid_bs,x.grid_3_cont_bs,x.grid_4_cont_bs,x.grid_5_cont_bs,x.grid_6_cont_bs)

blackscholes_test_cont <- rep(NA,length(x2_cont_bs[,1]))
for (row in 1:nrow(data.frame(x2_cont_bs))){
  blackscholes_test_cont[row] <- as.numeric(blackscholes(1,S0=x2_cont_bs[row,5],K=x2_cont_bs[row,6],r=x2
}


post_data_cont_SGP <- list(theta=c(post_mean_theta_1_SGP,post_mean_theta_2_SGP,post_mean_theta_3_SGP,pos

post_data_cont_Bdrycov <- list(theta=c(post_mean_theta_1_Bdrycov,post_mean_theta_2_Bdrycov,post_mean_the


# post_data

pred_gp_cont_SGP <- stan(file="Predictive GP_6dimension_withBS.stan", data=post_data_cont_SGP,iter=200,
```

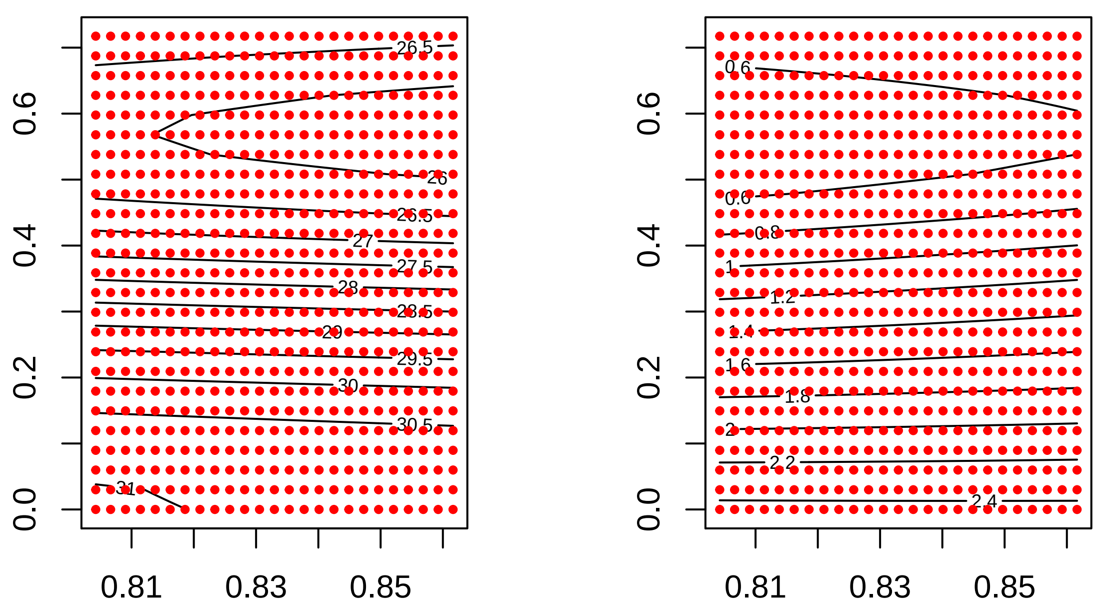
```
## DIAGNOSTIC(S) FROM PARSER:
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line comments.
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line comments.
##
##
## SAMPLING FOR MODEL 'Predictive GP_6dimension_withBS' NOW (CHAIN 1).
## Chain 1: Iteration:   1 / 200 [  0%]  (Sampling)
## Chain 1: Iteration: 100 / 200 [ 50%]  (Sampling)
## Chain 1: Iteration: 200 / 200 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                51.615 seconds (Sampling)
## Chain 1:                51.615 seconds (Total)
## Chain 1:
```

```r
pred_gp_cont_Bdrycov <- stan(file="Predictive GP_6dimension_withBS_Bdrycov.stan", data=post_data_cont_Bd
```

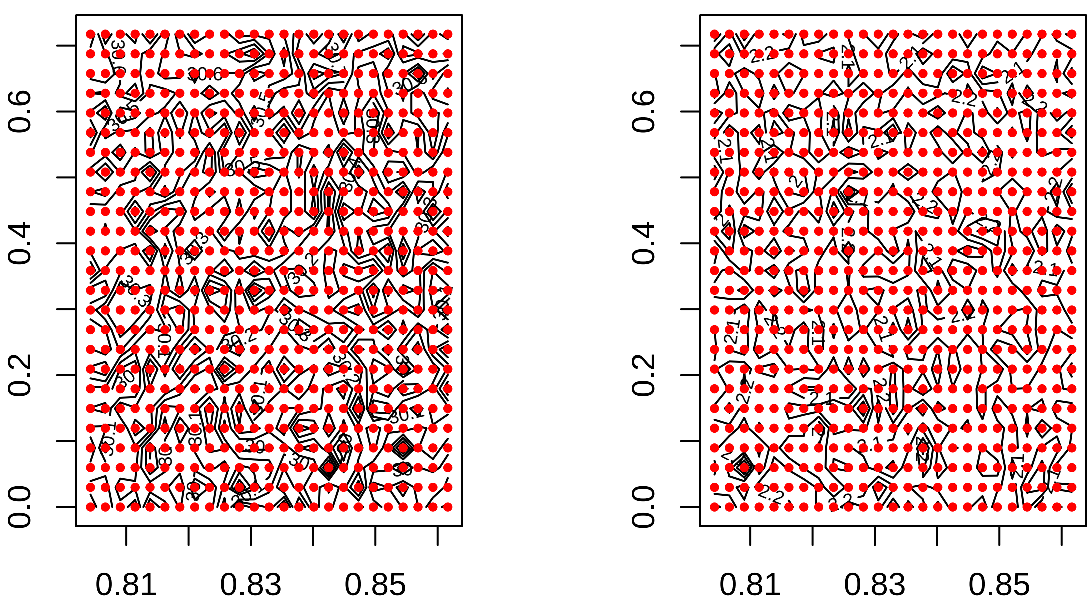
```
## DIAGNOSTIC(S) FROM PARSER:
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line comments.
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line comments.
##
##
## SAMPLING FOR MODEL 'Predictive GP_6dimension_withBS_Bdrycov' NOW (CHAIN 1).
## Chain 1: Iteration:   1 / 200 [  0%]  (Sampling)
## Chain 1: Iteration: 100 / 200 [ 50%]  (Sampling)
## Chain 1: Iteration: 200 / 200 [100%]  (Sampling)
```

```
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                85.286 seconds (Sampling)
## Chain 1:                85.286 seconds (Total)
## Chain 1:
```

```
#Computing Mean
y_predict_values_cont_SGP <- extract(pred_gp_cont_SGP,permuted=FALSE)
y_mean_values_cont_SGP <- c(colMeans(y_predict_values_cont_SGP))
y_mean_values_cont_SGP <- y_mean_values_cont_SGP[1:(length(y_mean_values_cont_SGP)-1)]

#Computing Standard Deviation
pred_gp_summary_cont_SGP <- summary(pred_gp_cont_SGP, sd=c("sd"))$summary
pred_gp_sd_cont_SGP <- pred_gp_summary_cont_SGP[, c("sd")]
y_sd_values_cont_SGP <- pred_gp_sd_cont_SGP[1:(length(pred_gp_sd_cont_SGP)-1)]

par(mfrow = c(1, 2))
#Contour for Predictions aka mean values of predicitons
contour(dim1, dim2, matrix(y_mean_values_cont_SGP, length(dim1), length(dim2)))
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
#Contour of Variance
contour(dim1, dim2, matrix(y_sd_values_cont_SGP, length(dim1), length(dim2)))
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
```



```r
#Computing Mean
y_predict_values_cont_Bdrycov <- extract(pred_gp_cont_Bdrycov,permuted=FALSE)
y_mean_values_cont_Bdrycov <- c(colMeans(y_predict_values_cont_Bdrycov))
y_mean_values_cont_Bdrycov <- y_mean_values_cont_Bdrycov[1:(length(y_mean_values_cont_Bdrycov)-1)]

#Computing Standard Deviation
pred_gp_summary_cont_Bdrycov <- summary(pred_gp_cont_Bdrycov, sd=c("sd"))$summary
pred_gp_sd_cont_Bdrycov <- pred_gp_summary_cont_Bdrycov[, c("sd")]
y_sd_values_cont_Bdrycov <- pred_gp_sd_cont_Bdrycov[1:(length(pred_gp_sd_cont_Bdrycov)-1)]

par(mfrow = c(1, 2))
#Contour for Predictions aka mean values of predicitons
contour(dim1, dim2, matrix(y_mean_values_cont_Bdrycov, length(dim1), length(dim2)))
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
#Contour of Variance
contour(dim1, dim2, matrix(y_sd_values_cont_Bdrycov, length(dim1), length(dim2)))
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
```



##Part 5: Improving the model by incorporating discrepancy

5-1: Computing Predicted European Option Prices

```r
# Fitting GP model
stan_dat_European <- read_rdump('Financial_Data_Call_European.R')
```

```
## Parsed with column specification:
```

```
## cols(
##   .default = col_double(),
##   date = col_character(),
##   symbol = col_character(),
##   exdate = col_character(),
##   cp_flag = col_character(),
##   ticker = col_character(),
##   exercise_style = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```r
fit_gp_SGP_European <- stan(file="gp-fit-6dimension_withBS.stan", data=stan_dat_European,
               iter=100, chains=1);
```

```
## DIAGNOSTIC(S) FROM PARSER:
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line comments.
##
##
## SAMPLING FOR MODEL 'gp-fit-6dimension_withBS' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.012 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 120 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: WARNING: There aren't enough warmup iterations to fit the
## Chain 1:          three stages of adaptation as currently configured.
## Chain 1:          Reducing each adaptation stage to 15%/75%/10% of
## Chain 1:          the given number of warmup iterations:
## Chain 1:            init_buffer = 7
## Chain 1:            adapt_window = 38
## Chain 1:            term_buffer = 5
## Chain 1:
## Chain 1: Iteration:  1 / 100 [  1%]  (Warmup)
## Chain 1: Iteration: 10 / 100 [ 10%]  (Warmup)
## Chain 1: Iteration: 20 / 100 [ 20%]  (Warmup)
## Chain 1: Iteration: 30 / 100 [ 30%]  (Warmup)
## Chain 1: Iteration: 40 / 100 [ 40%]  (Warmup)
## Chain 1: Iteration: 50 / 100 [ 50%]  (Warmup)
## Chain 1: Iteration: 51 / 100 [ 51%]  (Sampling)
## Chain 1: Iteration: 60 / 100 [ 60%]  (Sampling)
## Chain 1: Iteration: 70 / 100 [ 70%]  (Sampling)
## Chain 1: Iteration: 80 / 100 [ 80%]  (Sampling)
## Chain 1: Iteration: 90 / 100 [ 90%]  (Sampling)
## Chain 1: Iteration: 100 / 100 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 9.938 seconds (Warm-up)
## Chain 1:                8.756 seconds (Sampling)
## Chain 1:                18.694 seconds (Total)
## Chain 1:
```

```r
print(fit_gp_SGP_European, pars = c('theta','sigma2','gamma2'))
```

```
## Inference for Stan model: gp-fit-6dimension_withBS.
## 1 chains, each with iter=100; warmup=50; thin=1;
## post-warmup draws per chain=50, total post-warmup draws=50.
##
##              mean se_mean      sd    2.5%     25%     50%     75%    97.5%
## theta[1]     0.53    0.09    0.38    0.16    0.23    0.39    0.75     1.52
## theta[2]     6.17    0.28    1.91    2.96    4.83    5.97    7.18    10.84
## theta[3]    10.83    0.41    3.04    5.63    8.84   10.28   13.45    16.65
## theta[4]     0.45    0.01    0.11    0.31    0.37    0.42    0.51     0.66
## theta[5]     1.95    0.53    3.17    0.20    0.60    1.10    1.83     7.81
## theta[6]     0.16    0.01    0.05    0.10    0.13    0.15    0.19     0.29
## sigma2       0.03    0.00    0.00    0.02    0.02    0.03    0.03     0.03
## gamma2    5867.99  318.03 2399.01 3221.93 4136.17 5028.17 6871.16 12014.57
##           n_eff Rhat
## theta[1]     18 0.98
## theta[2]     46 1.05
## theta[3]     55 0.98
## theta[4]     61 1.03
## theta[5]     35 1.02
## theta[6]     63 1.02
## sigma2       71 0.98
## gamma2       57 1.07
##
## Samples were drawn using NUTS(diag_e) at Tue Mar 31 15:26:55 2020.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```r
sum_gp_SGP_European <- extract(fit_gp_SGP_European,permuted=FALSE)
```

```r
# Predicting from GP model
post_mean_theta_1_SGP <- mean(sum_gp_SGP_European[,1,1]) #theta
post_mean_theta_2_SGP <- mean(sum_gp_SGP_European[,1,2]) #theta
post_mean_theta_3_SGP <- mean(sum_gp_SGP_European[,1,3]) #theta
post_mean_theta_4_SGP <- mean(sum_gp_SGP_European[,1,4]) #theta
post_mean_theta_5_SGP <- mean(sum_gp_SGP_European[,1,5]) #theta
post_mean_theta_6_SGP <- mean(sum_gp_SGP_European[,1,6]) #theta
post_mean_sigma2_SGP <- mean(sum_gp_SGP_European[,1,7]) #sigma2
post_mean_gamma2_SGP <- mean(sum_gp_SGP_European[,1,8]) #gamma2
post_mean_mu_SGP <- stan_dat_European$blackscholes
```

```r
x2_bs <- cbind(as.numeric(stan_dat$total_calls_American$forward_price[test_start:test_end]),as.numeric(s

blackscholes_test <- rep(NA,length(x2_bs[,1]))
for (row in 1:nrow(data.frame(x2_bs))){
  blackscholes_test[row] <- as.numeric(blackscholes(1,S0=as.numeric(stan_dat$total_calls_American$forwar
}

x.grid_1 <- as.numeric(stan_dat$total_calls_American$forward_price_scaled[test_start:test_end])
x.grid_2 <- as.numeric(stan_dat$total_calls_American$strike_price_scaled[test_start:test_end])
```

```r
x.grid_3 <- as.numeric(stan_dat$total_calls_American$impl_volatility_scaled[test_start:test_end])
x.grid_4 <- as.numeric(stan_dat$total_calls_American$time_to_exp_scaled[test_start:test_end])
x.grid_5 <- as.numeric(stan_dat$total_calls_American$dividend_yield_scaled[test_start:test_end])
x.grid_6 <- as.numeric(stan_dat$total_calls_American$interest_rate_scaled[test_start:test_end])
x2 <- cbind(x.grid_1,x.grid_2,x.grid_3,x.grid_4,x.grid_5,x.grid_6)
```

```r
# X.grid <- expand.grid(x1 = x.grid_1, x2 = x.grid_2)

post_data_Bdrycov_American_disc <- list(theta=c(post_mean_theta_1_SGP,post_mean_theta_2_SGP,post_mean_t
# post_data

pred_gp_Bdrycov_disc <- stan(file="Predictive GP_6dimension_withBS_Bdrycov.stan", data=post_data_Bdryco
```

```
## DIAGNOSTIC(S) FROM PARSER:
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line comments.
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line comments.
##
##
## SAMPLING FOR MODEL 'Predictive GP_6dimension_withBS_Bdrycov' NOW (CHAIN 1).
## Chain 1: Iteration:   1 / 200 [  0%]  (Sampling)
## Chain 1: Iteration: 100 / 200 [ 50%]  (Sampling)
## Chain 1: Iteration: 200 / 200 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                26.212 seconds (Sampling)
## Chain 1:                26.212 seconds (Total)
## Chain 1:
```

```r
#Computing Mean
y_predict_values_Bdrycov_disc <- extract(pred_gp_Bdrycov_disc,permuted=FALSE)
y_mean_values_Bdrycov_disc <- c(colMeans(y_predict_values_Bdrycov_disc))
y_mean_values_Bdrycov_disc <- y_mean_values_Bdrycov_disc[1:(length(y_mean_values_Bdrycov_disc)-1)]

#Computing Standard Deviation
pred_gp_summary_Bdrycov_disc <- summary(pred_gp_Bdrycov_disc, sd=c("sd"))$summary
pred_gp_sd_Bdrycov_disc <- pred_gp_summary_Bdrycov_disc[, c("sd")]
y_sd_values_Bdrycov_disc <- pred_gp_sd_Bdrycov_disc[1:(length(pred_gp_sd_Bdrycov_disc)-1)]
```

3-3: Plotting Predicted Values against Truth

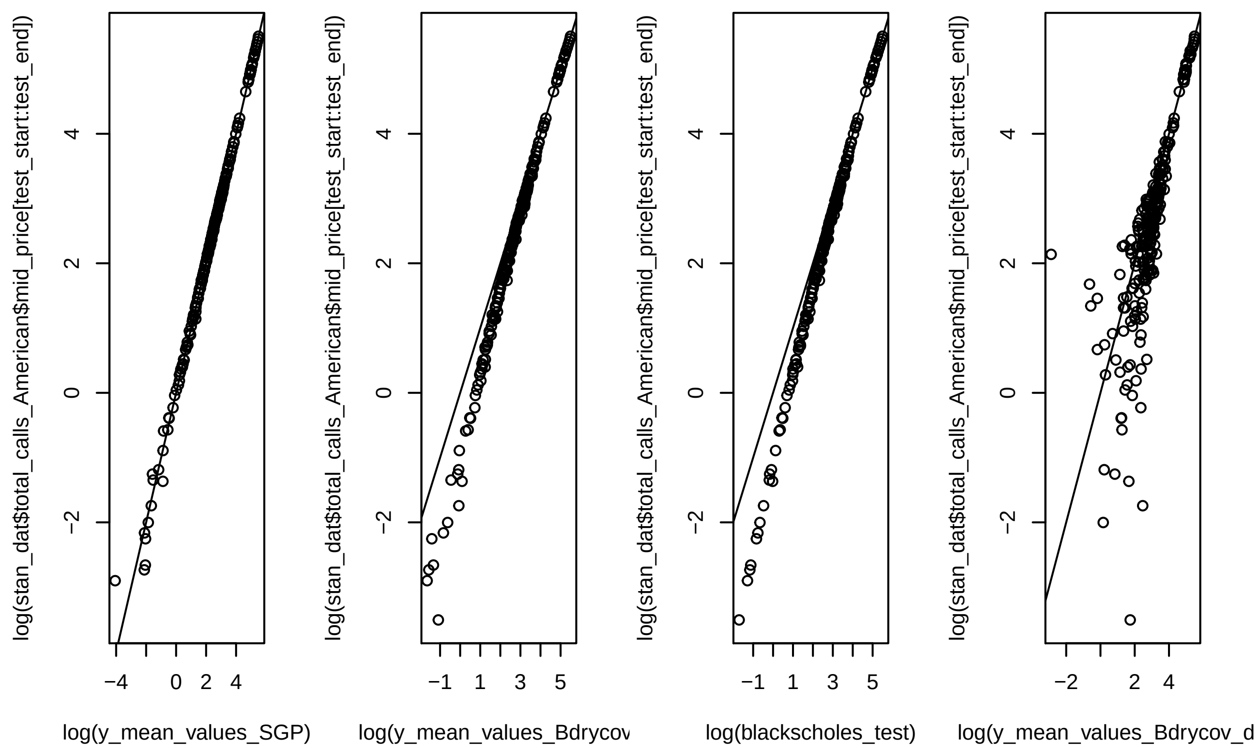
```r
par(mfrow=c(1,4))
#Plotting Standard GP
plot(log(y_mean_values_SGP),log(stan_dat$total_calls_American$mid_price[test_start:test_end]),xlim = c(
```

```
## Warning in log(y_mean_values_SGP): NaNs produced

## Warning in log(y_mean_values_SGP): NaNs produced

## Warning in log(y_mean_values_SGP): NaNs produced
```

```r
abline(0,1)

#Plotting BDrycov
plot(log(y_mean_values_Bdrycov),log(stan_dat$total_calls_American$mid_price[test_start:test_end]), xlim
abline(0,1)

#Plotting Blackscholes
plot(log(blackscholes_test),log(stan_dat$total_calls_American$mid_price[test_start:test_end]), xlim = c
abline(0,1)


#Plotting Discrepancy Model
plot(log(y_mean_values_Bdrycov_disc),log(stan_dat$total_calls_American$mid_price[test_start:test_end]),
```

```
## Warning in log(y_mean_values_Bdrycov_disc): NaNs produced

## Warning in log(y_mean_values_Bdrycov_disc): NaNs produced

## Warning in log(y_mean_values_Bdrycov_disc): NaNs produced
```

```r
abline(0,1)
```



```r
#MSE
library('MLmetrics')
MSE(y_mean_values_SGP,stan_dat$total_calls_American$mid_price[test_start:test_end])
```

```
## [1] 0.2722204
```

```r
MSE(y_mean_values_Bdrycov,stan_dat$total_calls_American$mid_price[test_start:test_end])
```

```
## [1] 10.23453
```

```r
MSE(blackscholes_test,stan_dat$total_calls_American$mid_price[test_start:test_end])
```

```
## [1] 10.12211
```

```r
MSE(y_mean_values_Bdrycov_disc,stan_dat$total_calls_American$mid_price[test_start:test_end])
```

```
## [1] 44.54636
```