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_Put_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.
```

```
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 0.091 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 910 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: 87.463 seconds (Warm-up)
## Chain 1:                77.141 seconds (Sampling)
## Chain 1:                164.604 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%
## theta[1]     0.21    0.00    0.04    0.15    0.19    0.21    0.24    0.30
## theta[2]     1.28    0.05    0.31    0.85    1.01    1.26    1.46    1.89
## theta[3]    10.93    0.29    1.59    7.33   10.22   10.94   12.09   13.24
## theta[4]     0.47    0.01    0.04    0.42    0.45    0.47    0.50    0.55
## theta[5]     1.35    0.23    0.97    0.64    0.83    1.07    1.44    4.82
## theta[6]    59.77    4.19   23.67   26.21   40.27   56.24   76.32  107.83
## sigma2       0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
## gamma2    4120.24  285.30 1392.81 2211.36 3035.92 4000.25 4920.80 7022.30
##           n_eff Rhat
## theta[1]     85 0.99
## theta[2]     39 0.99
## theta[3]     29 1.06
## theta[4]     49 1.00
## theta[5]     17 1.10
## theta[6]     32 1.04
## sigma2       38 0.98
## gamma2       24 1.07
##
## Samples were drawn using NUTS(diag_e) at Fri Mar 27 21:58:43 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
test_end <- 559 #06/14

# test_start <- 560 #06/17
```

test_end <- 852 #06/20

```
x.grid_1 <- as.numeric(stan_dat$total_puts_American$forward_price[test_start:test_end])
x.grid_2 <- as.numeric(stan_dat$total_puts_American$strike_price[test_start:test_end])
x.grid_3 <- as.numeric(stan_dat$total_puts_American$impl_volatility[test_start:test_end])
x.grid_4 <- as.numeric(stan_dat$total_puts_American$time_to_exp[test_start:test_end]*250)
x.grid_5 <- as.numeric(stan_dat$total_puts_American$dividend[test_start:test_end])
x.grid_6 <- as.numeric(stan_dat$total_puts_American$interest_rate[test_start:test_end])
x2 <- cbind(x.grid_1,x.grid_2,x.grid_3,x.grid_4,x.grid_5,x.grid_6)

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
```

```
library('ragtop')
blackscholes_2 <- rep(NA,length(x2[,1]))
for (row in 1:nrow(data.frame(x2))){
  blackscholes_2[row] <- as.numeric(blackscholes(-1,S0=x.grid_1[row],K=x.grid_2[row],r=x.grid_6[row],t=
  # blackscholes_2[row] <- Black_Scholes(0,x.grid_1[row],x.grid_6[row],x.grid_3[row],x.grid_2[row],x.gr
}

x.grid_1 <- as.numeric(stan_dat$total_puts_American$forward_price_scaled[test_start:test_end])
x.grid_2 <- as.numeric(stan_dat$total_puts_American$strike_price_scaled[test_start:test_end])
x.grid_3 <- as.numeric(stan_dat$total_puts_American$impl_volatility[test_start:test_end])
x.grid_4 <- as.numeric(stan_dat$total_puts_American$time_to_exp[test_start:test_end])
x.grid_5 <- as.numeric(stan_dat$total_puts_American$dividend[test_start:test_end])
x.grid_6 <- as.numeric(stan_dat$total_puts_American$interest_rate[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

```
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, w
```

```
## 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:                29.59 seconds (Sampling)
## Chain 1:                29.59 seconds (Total)
## Chain 1:
```

##Part2: Bdrycov Gaussian Process

2-1: Fitting

```r
# Fitting GP model for Bdrycov
stan_dat <- read_rdump('Financial_Data_Put_American.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_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 0.416 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 4160 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: 385.753 seconds (Warm-up)
## Chain 1:                428.585 seconds (Sampling)
## Chain 1:                814.338 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]  1.33    0.11  0.67 0.59 0.83  1.13   1.63   3.03    39 0.98
## theta[2]  1.25    0.13  0.73 0.55 0.80  1.05   1.33   3.28    29 0.99
## theta[3]  1.08    0.05  0.50 0.50 0.75  0.97   1.24   2.44    85 0.98
## theta[4]  1.24    0.08  0.59 0.55 0.84  1.16   1.55   2.39    56 0.98
## theta[5]  1.46    0.20  0.93 0.50 0.82  1.14   1.73   3.46    22 0.98
## theta[6]  1.39    0.16  0.80 0.58 0.75  1.15   1.78   3.41    25 1.00
## sigma2   63.59   19.23 57.43 0.01 2.59 46.53 122.11 152.56     9 1.00
## gamma2   75.14   20.37 59.32 0.30 6.14 81.88 129.62 158.79     8 1.02
##
## Samples were drawn using NUTS(diag_e) at Fri Mar 27 22:15:26 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_Bdrycov_American <- extract(fit_gp_Bdrycov_American,permuted=FALSE)
# saveRDS(fit_gp,file ="fit_gp_vol50_within50spot_7to19days")
```

```r
# 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


# x2 <- as.numeric(unlist(spx_spy_2019_06_30_put_2017_06_500rows_test['strike_price']))
# x2<- cbind(spy_2013_01_01_2013_01_31_put$strike_price[201:300],spy_2013_01_01_2013_01_31_put$impl_vol
# x2 <- seq(from=-2,to=2,by=0.01)

# x2 <- cbind(seq(from=0,to=1,by=0.01),seq(from=0,to=1,by=0.01))
```

2-2: Predictions

```r
# 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:                60.786 seconds (Sampling)
## Chain 1:                60.786 seconds (Total)
## Chain 1:
```

##Part 3 Predictions Versus Truth

3-1: Computing Means Standard GP

```r
#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

```r
#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

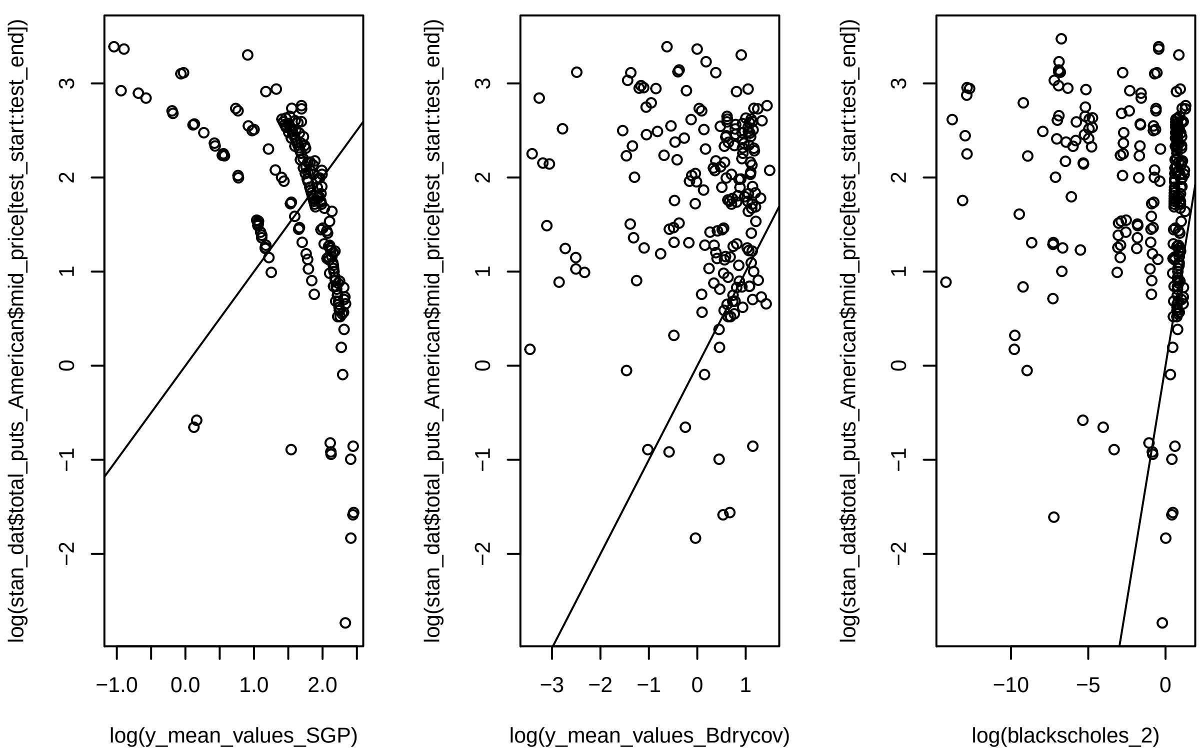
```r
par(mfrow=c(1,3))
#Plotting Standard GP
plot(log(y_mean_values_SGP),log(stan_dat$total_puts_American$mid_price[test_start:test_end]),xlim = c(mi
```

```
## 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$total_puts_American$mid_price[test_start:test_end]), xlim =
```

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

## Warning in log(y_mean_values_Bdrycov): NaNs produced

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

```
abline(0,1)

#Plotting Blackscholes
plot(log(blackscholes_2),log(stan_dat$total_puts_American$mid_price[test_start:test_end]), xlim = c(min
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$total_puts_American$mid_price[test_start:test_end])
```

```
## [1] 110.5443
```

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

```
## [1] 92.0102
```

```r
MSE(blackscholes_2,stan_dat$total_puts_American$mid_price[test_start:test_end])
```

```
## [1] 92.78114
```

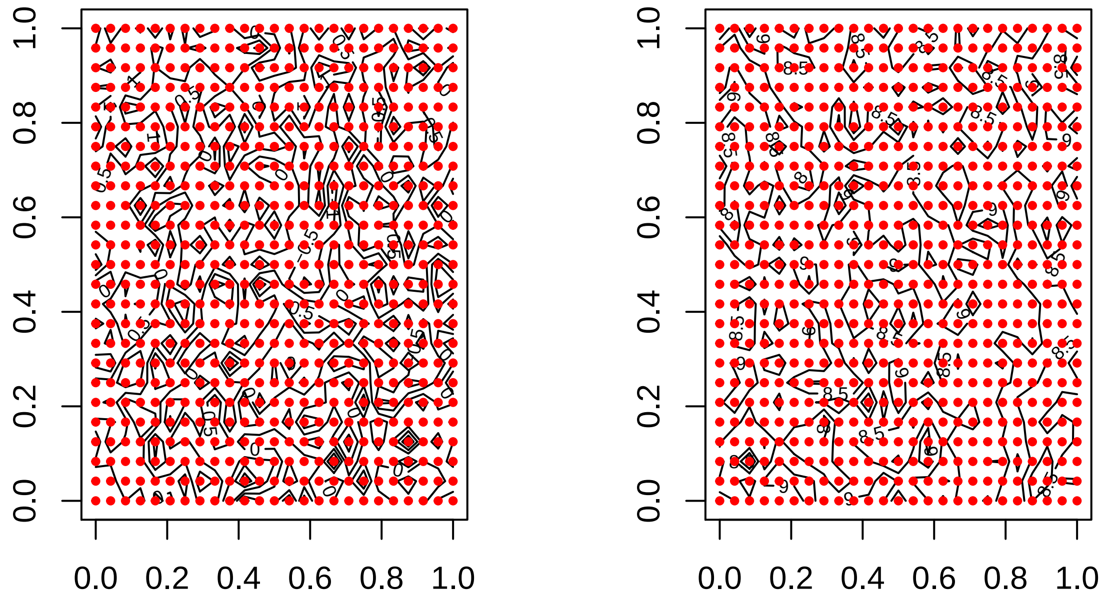
##Part 4 Visualizations

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

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

dim1 <- seq(0.000000001,1.000000001,length.out = 25)
dim2 <- seq(0.000000001,1.000000001,length.out = 25)
X.grid <- expand.grid(x1 = dim1, x2 = dim2)

x.grid_3_cont <- as.numeric(rep(mean(stan_dat$total_puts_American$impl_volatility[test_start:test_end]))
x.grid_4_cont <- as.numeric(rep(mean(stan_dat$total_puts_American$time_to_exp[test_start:test_end])),nro
x.grid_5_cont <- as.numeric(rep(mean(stan_dat$total_puts_American$dividend_yield[test_start:test_end]))
x.grid_6_cont <- as.numeric(rep(mean(stan_dat$total_puts_American$interest_rate[test_start:test_end])),r

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

blackscholes_2_cont <- rep(NA,length(x2_cont[,1]))
for (row in 1:nrow(data.frame(x2_cont))){
  blackscholes_2_cont[row] <- as.numeric(blackscholes(-1,S0=x2_cont[row,1],K=x2_cont[row,2],r=x2_cont[ro
}


post_data_cont <- list(theta=c(post_mean_theta_1_Bdrycov,post_mean_theta_2_Bdrycov,post_mean_theta_3_Bdr
# post_data

pred_gp_cont <- stan(file="Predictive GP_6dimension_withBS_Bdrycov.stan", data=post_data_cont,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_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:                203.437 seconds (Sampling)
## Chain 1:                203.437 seconds (Total)
## Chain 1:
```

```r
#Computing Mean
y_predict_values_cont <- extract(pred_gp_cont,permuted=FALSE)
y_mean_values_cont <- c(colMeans(y_predict_values_cont))
y_mean_values_cont <- y_mean_values_cont[1:(length(y_mean_values_cont)-1)]

#Computing Standard Deviation
pred_gp_summary_cont <- summary(pred_gp_cont, sd=c("sd"))$summary
pred_gp_sd_cont <- pred_gp_summary_cont[, c("sd")]
y_sd_values_cont <- pred_gp_sd_cont[1:(length(pred_gp_sd_cont)-1)]

par(mfrow = c(1, 2))
#Contour for Predictions aka mean values of predicitons
# x1_grid_cont <- seq(from=min(x.grid_1_cont), to=max(x.grid_1_cont), length.out=length(x.grid_1_cont))
# x2_grid_cont <- seq(from=min(x.grid_2_cont), to=max(x.grid_2_cont), length.out=length(x.grid_2_cont))

contour(dim1, dim2, matrix(y_mean_values_cont, 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, length(dim1), length(dim2)))
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
```

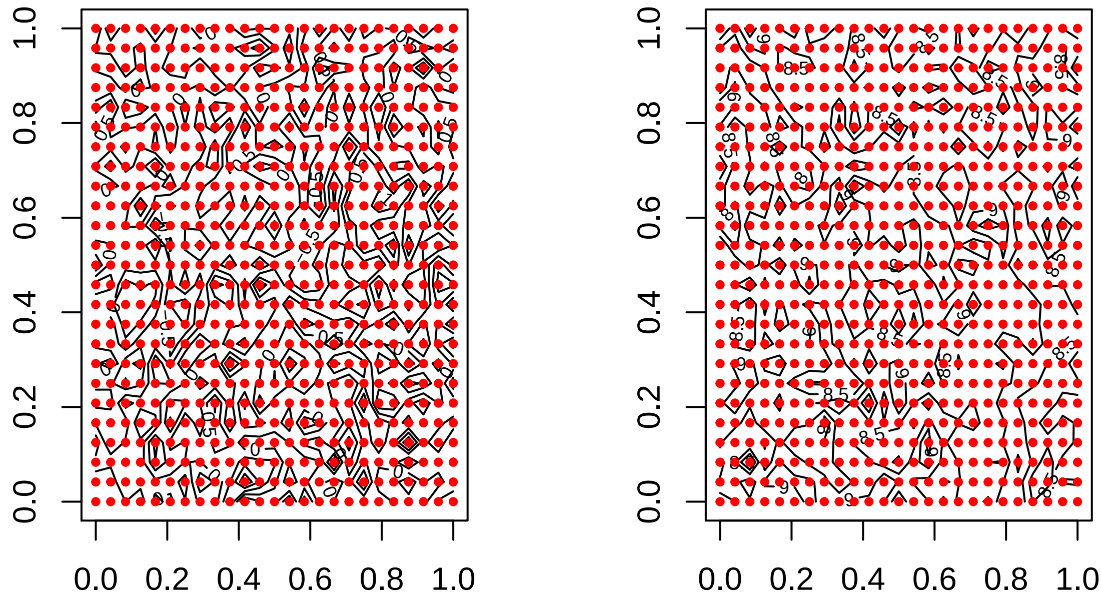


4-2: Contour Plots of Implied VOlatility & Time to Expiration

```
x.grid_1_cont <- as.numeric(stan_dat$total_puts_American$impl_volatility[test_start:test_end])
x.grid_2_cont <- as.numeric(stan_dat$total_puts_American$time_to_exp[test_start:test_end])

dim1 <- seq(0.000000001,1.000000001,length.out = 25)
dim2 <- seq(0.000000001,1.000000001,length.out = 25)
X.grid <- expand.grid(x1 = dim1, x2 = dim2)

x.grid_3_cont <- as.numeric(rep(mean(stan_dat$total_puts_American$forward_price_scaled[test_start:test_e
x.grid_4_cont <- as.numeric(rep(mean(stan_dat$total_puts_American$strike_price_scaled[test_start:test_en
x.grid_5_cont <- as.numeric(rep(mean(stan_dat$total_puts_American$dividend_yield[test_start:test_end]))
x.grid_6_cont <- as.numeric(rep(mean(stan_dat$total_puts_American$interest_rate[test_start:test_end])),r

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

blackscholes_2_cont <- rep(NA,length(x2_cont[,1]))
for (row in 1:nrow(data.frame(x2_cont))){
  blackscholes_2_cont[row] <- as.numeric(blackscholes(-1,S0=x2_cont[row,3],K=x2_cont[row,4],r=x2_cont[ro
}

post_data_cont <- list(theta=c(post_mean_theta_1_Bdrycov,post_mean_theta_2_Bdrycov,post_mean_theta_3_Bdr
# post_data

pred_gp_cont <- stan(file="Predictive GP_6dimension_withBS_Bdrycov.stan", data=post_data_cont,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_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:                173.783 seconds (Sampling)
## Chain 1:                173.783 seconds (Total)
## Chain 1:
```

```r
#Computing Mean
y_predict_values_cont <- extract(pred_gp_cont,permuted=FALSE)
y_mean_values_cont <- c(colMeans(y_predict_values_cont))
y_mean_values_cont <- y_mean_values_cont[1:(length(y_mean_values_cont)-1)]

#Computing Standard Deviation
pred_gp_summary_cont <- summary(pred_gp_cont, sd=c("sd"))$summary
pred_gp_sd_cont <- pred_gp_summary_cont[, c("sd")]
y_sd_values_cont <- pred_gp_sd_cont[1:(length(pred_gp_sd_cont)-1)]

par(mfrow = c(1, 2))
#Contour for Predictions aka mean values of predicitons
# x1_grid_cont <- seq(from=min(x.grid_1_cont), to=max(x.grid_1_cont), length.out=length(x.grid_1_cont))
# x2_grid_cont <- seq(from=min(x.grid_2_cont), to=max(x.grid_2_cont), length.out=length(x.grid_2_cont))

contour(dim1, dim2, matrix(y_mean_values_cont, 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, length(dim1), length(dim2)))
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
```

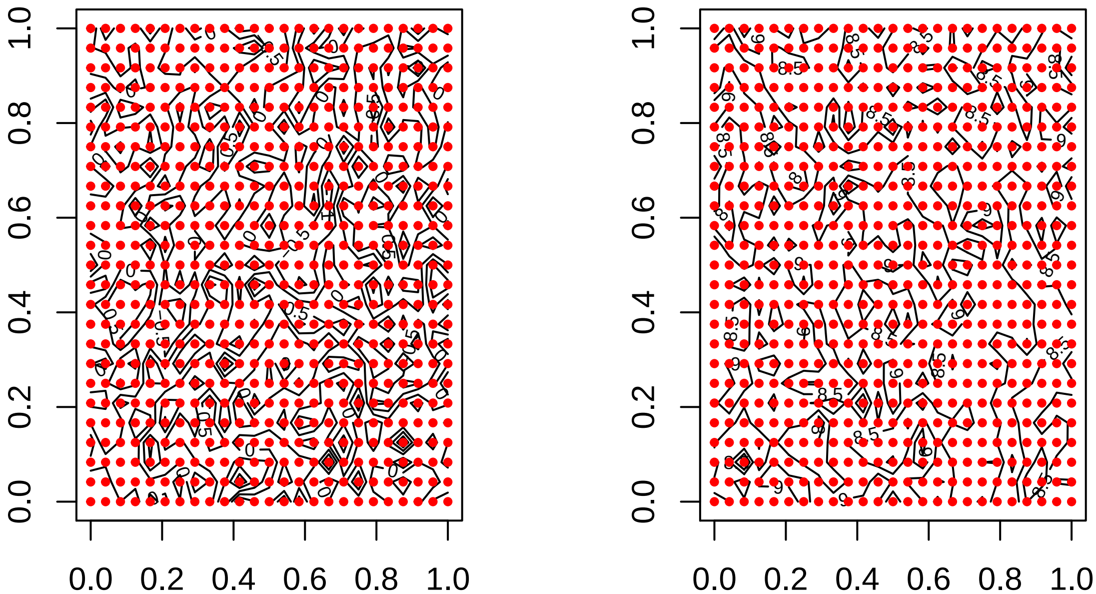


4-3: Contour Plots of Interest Rate & Time to Expiration

```
x.grid_1_cont <- as.numeric(stan_dat$total_puts_American$interest_rate[test_start:test_end])
x.grid_2_cont <- as.numeric(stan_dat$total_puts_American$time_to_exp[test_start:test_end])

dim1 <- seq(0.000000001,1.000000001,length.out = 25)
dim2 <- seq(0.000000001,1.000000001,length.out = 25)
X.grid <- expand.grid(x1 = dim1, x2 = dim2)

x.grid_3_cont <- as.numeric(rep(mean(stan_dat$total_puts_American$forward_price_scaled[test_start:test_e
x.grid_4_cont <- as.numeric(rep(mean(stan_dat$total_puts_American$strike_price_scaled[test_start:test_en
x.grid_5_cont <- as.numeric(rep(mean(stan_dat$total_puts_American$impl_volatility[test_start:test_end]))
x.grid_6_cont <- as.numeric(rep(mean(stan_dat$total_puts_American$dividend_yield[test_start:test_end]))

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

blackscholes_2_cont <- rep(NA,length(x2_cont[,1]))
for (row in 1:nrow(data.frame(x2_cont))){
  blackscholes_2_cont[row] <- as.numeric(blackscholes(-1,S0=x2_cont[row,3],K=x2_cont[row,4],r=x2_cont[ro
}

post_data_cont <- list(theta=c(post_mean_theta_1_Bdrycov,post_mean_theta_2_Bdrycov,post_mean_theta_3_Bdr
# post_data

pred_gp_cont <- stan(file="Predictive GP_6dimension_withBS_Bdrycov.stan", data=post_data_cont,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_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:                173.928 seconds (Sampling)
## Chain 1:                173.928 seconds (Total)
## Chain 1:
```

```r
#Computing Mean
y_predict_values_cont <- extract(pred_gp_cont,permuted=FALSE)
y_mean_values_cont <- c(colMeans(y_predict_values_cont))
y_mean_values_cont <- y_mean_values_cont[1:(length(y_mean_values_cont)-1)]

#Computing Standard Deviation
pred_gp_summary_cont <- summary(pred_gp_cont, sd=c("sd"))$summary
pred_gp_sd_cont <- pred_gp_summary_cont[, c("sd")]
y_sd_values_cont <- pred_gp_sd_cont[1:(length(pred_gp_sd_cont)-1)]

par(mfrow = c(1, 2))
#Contour for Predictions aka mean values of predicitons
# x1_grid_cont <- seq(from=min(x.grid_1_cont), to=max(x.grid_1_cont), length.out=length(x.grid_1_cont))
# x2_grid_cont <- seq(from=min(x.grid_2_cont), to=max(x.grid_2_cont), length.out=length(x.grid_2_cont))

contour(dim1, dim2, matrix(y_mean_values_cont, 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, 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
library(rstan)
source("gp.utility.R")

# Fitting GP model
stan_dat_European <- read_rdump('Financial_Data_Put_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 0.015 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 150 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: 15.119 seconds (Warm-up)
## Chain 1:                12.515 seconds (Sampling)
## Chain 1:                27.634 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%
## theta[1]       0.18     0.00      0.03      0.14      0.16      0.17
## theta[2]       2.21     0.06      0.48      1.53      1.87      2.10
## theta[3]       8.41     0.35      1.99      5.33      7.13      8.25
## theta[4]       0.22     0.00      0.03      0.18      0.20      0.22
## theta[5]       1.12     0.11      0.62      0.46      0.70      0.91
## theta[6]       1.76     0.14      0.71      0.83      1.14      1.65
## sigma2         0.00     0.00      0.00      0.00      0.00      0.00
## gamma2    266431.22 14712.74 104936.79 127741.62 205796.51 254846.75
##                 75%     97.5% n_eff Rhat
## theta[1]       0.19      0.23    85 0.98
```

```
## theta[2]      2.37      3.23    61 0.99
## theta[3]      9.48     12.71    31 1.02
## theta[4]      0.23      0.27    85 0.98
## theta[5]      1.44      2.15    33 1.00
## theta[6]      2.25      3.10    25 0.99
## sigma2        0.00      0.00    85 0.98
## gamma2   303771.02 461181.51    51 0.98
##
## Samples were drawn using NUTS(diag_e) at Fri Mar 27 22:27:44 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_SGP_European <- extract(fit_gp_SGP_European,permuted=FALSE)
```

```
# 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
```

```
x.grid_1 <- as.numeric(stan_dat$total_puts_American$forward_price[test_start:test_end])
x.grid_2 <- as.numeric(stan_dat$total_puts_American$strike_price[test_start:test_end])
x.grid_3 <- as.numeric(stan_dat$total_puts_American$impl_volatility[test_start:test_end])
x.grid_4 <- as.numeric(stan_dat$total_puts_American$time_to_exp[test_start:test_end]*250)
x.grid_5 <- as.numeric(stan_dat$total_puts_American$dividend[test_start:test_end])
x.grid_6 <- as.numeric(stan_dat$total_puts_American$interest_rate[test_start:test_end])
x2 <- cbind(x.grid_1,x.grid_2,x.grid_3,x.grid_4,x.grid_5,x.grid_6)

library('qrmtools')
library('ragtop')
blackscholes_2 <- rep(NA,length(x2[,1]))
for (row in 1:nrow(data.frame(x2))){
  blackscholes_2[row] <- as.numeric(blackscholes(-1,S0=x.grid_1[row],K=x.grid_2[row],r=x.grid_6[row],t=
  # blackscholes_2[row] <- Black_Scholes(0,x.grid_1[row],x.grid_6[row],x.grid_3[row],x.grid_2[row],x.gr
}

x.grid_1 <- as.numeric(stan_dat$total_puts_American$forward_price_scaled[test_start:test_end])
x.grid_2 <- as.numeric(stan_dat$total_puts_American$strike_price_scaled[test_start:test_end])
x.grid_3 <- as.numeric(stan_dat$total_puts_American$impl_volatility[test_start:test_end])
x.grid_4 <- as.numeric(stan_dat$total_puts_American$time_to_exp[test_start:test_end])
x.grid_5 <- as.numeric(stan_dat$total_puts_American$dividend[test_start:test_end])
x.grid_6 <- as.numeric(stan_dat$total_puts_American$interest_rate[test_start:test_end])
x2 <- cbind(x.grid_1,x.grid_2,x.grid_3,x.grid_4,x.grid_5,x.grid_6)
```

```
# 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
```

```r
# post_data

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

```
## 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:                25.555 seconds (Sampling)
## Chain 1:                25.555 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$total_puts_American$mid_price[test_start:test_end]),xlim = c(mi
```

```
## 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
```

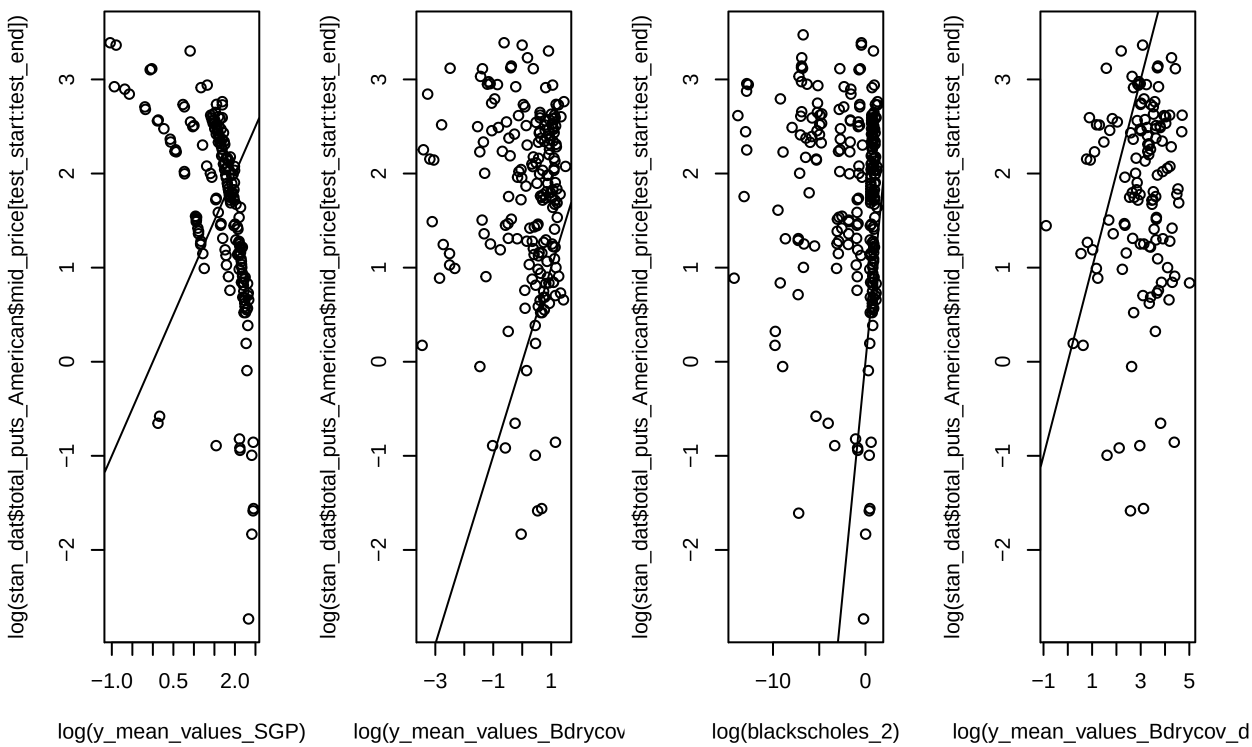
```r
abline(0,1)

#Plotting BDrycov
plot(log(y_mean_values_Bdrycov),log(stan_dat$total_puts_American$mid_price[test_start:test_end]), xlim =
```

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

## Warning in log(y_mean_values_Bdrycov): NaNs produced

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

```
abline(0,1)

#Plotting Blackscholes
plot(log(blackscholes_2),log(stan_dat$total_puts_American$mid_price[test_start:test_end]), xlim = c(min
abline(0,1)

#Plotting Discrepancy Model
plot(log(y_mean_values_Bdrycov_disc),log(stan_dat$total_puts_American$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
```

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




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

```
## [1] 110.5443
```

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

```
## [1] 92.0102
```

```r
MSE(blackscholes_2,stan_dat$total_puts_American$mid_price[test_start:test_end])
```

```
## [1] 92.78114
```

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

```
## [1] 1556.101
```