Gaussian_Process_Code

Chiwan Kim

2/3/2020

##Part 1: Standard Gaussian Process

1-1: Fitting

```r
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.
```

```r
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_SGP.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.
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line comments.
##
##
## SAMPLING FOR MODEL 'gp-fit-6dimension_SGP' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.097 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 970 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: 317.056 seconds (Warm-up)
## Chain 1:                357.773 seconds (Sampling)
## Chain 1:                674.829 seconds (Total)
## Chain 1:
```

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

```
## Inference for Stan model: gp-fit-6dimension_SGP.
## 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]     0.06    0.00    0.03   0.02    0.04    0.05    0.07    0.11
## theta[2]     1.17    0.09    0.57   0.41    0.81    1.01    1.34    2.43
## theta[3]     2.88    0.09    0.73   1.92    2.41    2.79    3.15    4.19
## theta[4]     0.23    0.02    0.12   0.07    0.14    0.20    0.28    0.50
## theta[5]     0.05    0.00    0.02   0.02    0.03    0.04    0.06    0.08
## theta[6]     0.02    0.00    0.01   0.01    0.01    0.01    0.02    0.03
## sigma2       0.00    0.00    0.00   0.00    0.00    0.00    0.00    0.00
## gamma2    2920.98  308.30 1651.33 814.94 1691.74 2734.89 3769.02 6166.51
##           n_eff Rhat
## theta[1]     75 0.98
## theta[2]     37 1.07
## theta[3]     62 0.98
## theta[4]     34 1.09
## theta[5]     40 0.99
## theta[6]     34 0.99
## sigma2       52 1.03
## gamma2       29 1.00
##
## Samples were drawn using NUTS(diag_e) at Tue Apr 28 05:28:02 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_American <- extract(fit_gp_SGP_American,permuted=FALSE)
```

```
# 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 <- mean(sum_gp_SGP_American[,1,9])
```

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

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

# test_start <- 609 #06/10 Calls
# test_end <- 999 #06/14 Calls

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

x_1 <- as.numeric(stan_dat$total_puts_American$forward_price_scaled[test_start:test_end])
x_2 <- as.numeric(stan_dat$total_puts_American$strike_price_scaled[test_start:test_end])
x_3 <- as.numeric(stan_dat$total_puts_American$impl_volatility_scaled[test_start:test_end])
x_4 <- as.numeric(stan_dat$total_puts_American$time_to_exp_scaled[test_start:test_end])
x_5 <- as.numeric(stan_dat$total_puts_American$dividend_yield_scaled[test_start:test_end])
x_6 <- as.numeric(stan_dat$total_puts_American$interest_rate_scaled[test_start:test_end])
x2 <- cbind(x_1,x_2,x_3,x_4,x_5,x_6)
```

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


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$total_puts_American$forwa
}
```

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_SGP.stan", data=post_data_SGP_American,iter=200, warn
```

```
## 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_SGP' 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:                22.022 seconds (Sampling)
## Chain 1:                22.022 seconds (Total)
## Chain 1:
```

##Part2: BS Integrated SGP

2-1: Fitting

```
# Fitting GP model for Bdrycov
fit_gp_bs_American <- stan(file="gp-fit-6dimension_withBS_SGP.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_SGP' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.085 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 850 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: 58.691 seconds (Warm-up)
## Chain 1:                68.379 seconds (Sampling)
## Chain 1:                127.07 seconds (Total)
## Chain 1:
```

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

```
## Inference for Stan model: gp-fit-6dimension_withBS_SGP.
## 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]   0.06    0.00  0.03 0.03  0.04  0.05  0.08  0.11    38 1.00
## theta[2]   1.89    0.13  0.72 0.92  1.31  1.78  2.43  3.60    31 1.05
## theta[3]   4.71    0.18  1.62 2.70  3.54  4.54  5.39  8.00    80 1.08
## theta[4]   0.47    0.03  0.18 0.22  0.35  0.40  0.55  0.93    43 1.09
## theta[5]   0.06    0.01  0.04 0.02  0.04  0.06  0.08  0.12    40 1.01
## theta[6]   0.05    0.00  0.02 0.03  0.04  0.05  0.06  0.10    53 1.05
## sigma2     0.00    0.00  0.00 0.00  0.00  0.00  0.00  0.00    56 0.98
## gamma2    27.80    3.33 18.17 9.89 17.83 22.36 34.07 64.84    30 1.07
##
## Samples were drawn using NUTS(diag_e) at Tue Apr 28 05:32:47 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_bs_American <- extract(fit_gp_bs_American,permuted=FALSE)
```

```r
# Predicting from GP model - 2 dimensional case
post_mean_theta_1_bs <- mean(sum_gp_bs_American[,1,1]) #theta
post_mean_theta_2_bs <- mean(sum_gp_bs_American[,1,2]) #theta
post_mean_theta_3_bs <- mean(sum_gp_bs_American[,1,3]) #theta
post_mean_theta_4_bs <- mean(sum_gp_bs_American[,1,4]) #theta
post_mean_theta_5_bs <- mean(sum_gp_bs_American[,1,5]) #theta
post_mean_theta_6_bs <- mean(sum_gp_bs_American[,1,6]) #theta
post_mean_sigma2_bs <- mean(sum_gp_bs_American[,1,7]) #sigma2
post_mean_gamma2_bs <- mean(sum_gp_bs_American[,1,8]) #gamma2
post_mean_mu_bs <- stan_dat$blackscholes
```

2-2: Predictions

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

post_data_bs_American <- list(theta=c(post_mean_theta_1_bs,post_mean_theta_2_bs,post_mean_theta_3_bs,pos
# post_data

pred_gp_bs <- stan(file="Predictive GP_6dimension_withBS_SGP.stan", data=post_data_bs_American,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_SGP' 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:                22.748 seconds (Sampling)
## Chain 1:                22.748 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)]
```

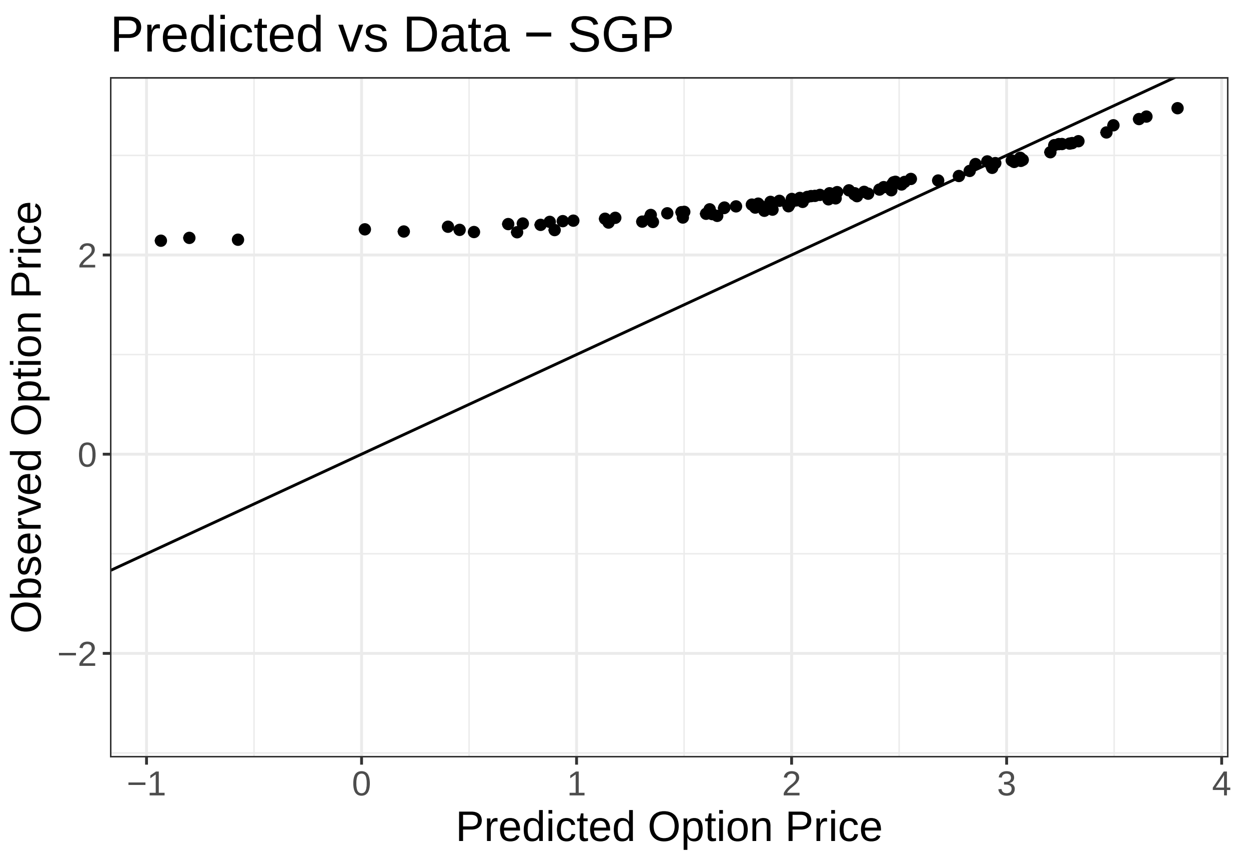
3-2: Computing Means bs

```
#Computing Mean
y_predict_values_bs <- extract(pred_gp_bs,permuted=FALSE)
y_mean_values_bs <- c(colMeans(y_predict_values_bs))
y_mean_values_bs <- y_mean_values_bs[1:(length(y_mean_values_bs)-1)]

#Computing Standard Deviation
pred_gp_summary_bs <- summary(pred_gp_bs, sd=c("sd"))$summary
pred_gp_sd_bs <- pred_gp_summary_bs[, c("sd")]
y_sd_values_bs <- pred_gp_sd_bs[1:(length(pred_gp_sd_bs)-1)]
```

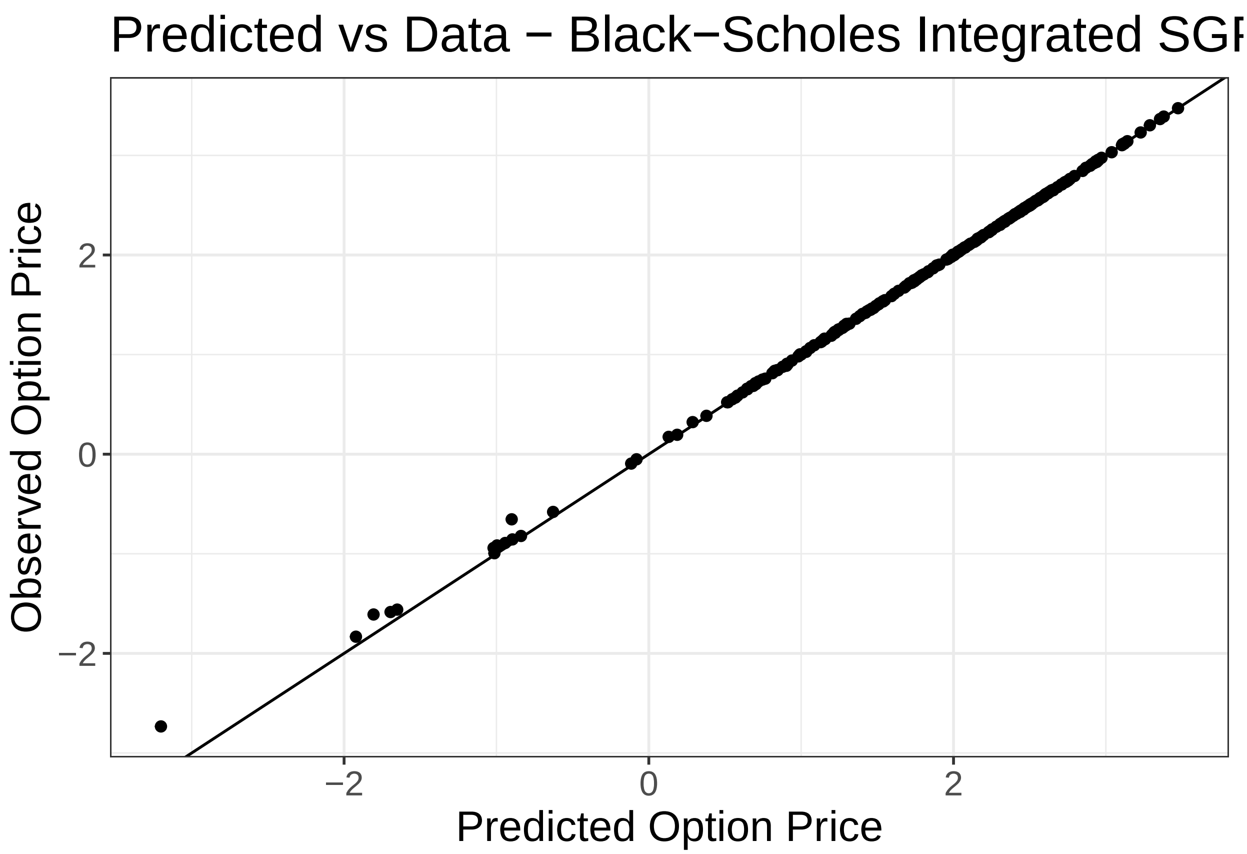
3-3: Plotting Predicted Values against Truth

```
par(mfrow=c(1,3))
#Plotting Standard GP
ggplot(mapping=aes(x=log(y_mean_values_SGP),y=log(stan_dat$total_puts_American$mid_price[test_start:tes
  geom_point() +
  labs(title = "Predicted vs Data - SGP",
       x = "Predicted Option Price",
       y = "Observed Option Price") +
  theme_bw() +
  theme(text=element_text(size=16)) +
  geom_abline(intercept = 0, slope = 1)
```

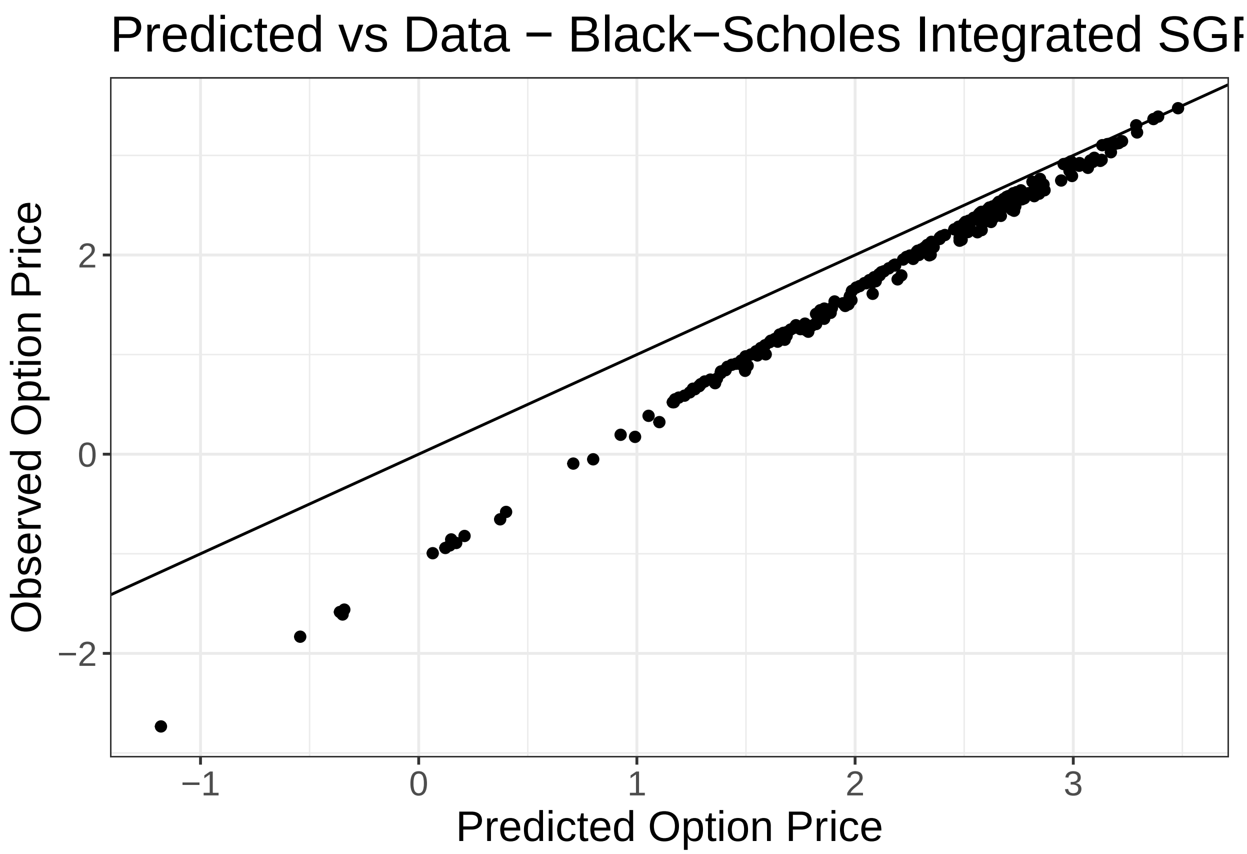
```
## Warning in log(y_mean_values_SGP): NaNs produced

## Warning in log(y_mean_values_SGP): NaNs produced

## Warning: Removed 140 rows containing missing values (geom_point).
```




```
#Plotting bs
ggplot(mapping=aes(x=log(y_mean_values_bs),y=log(stan_dat$total_puts_American$mid_price[test_start:test_
  geom_point() +
  labs(title = "Predicted vs Data - Black-Scholes Integrated SGP",
       x = "Predicted Option Price",
       y = "Observed Option Price") +
  theme_bw() +
  theme(text=element_text(size=16))+
  geom_abline(intercept = 0, slope = 1)
```




```r
#Plotting Blackscholes
ggplot(mapping=aes(x=log(blackscholes_test),y=log(stan_dat$total_puts_American$mid_price[test_start:tes
  geom_point() +
  labs(title = "Predicted vs Data - Black-Scholes Integrated SGP",
       x = "Predicted Option Price",
       y = "Observed Option Price") +
  theme_bw() +
  theme(text=element_text(size=16))+
  geom_abline(intercept = 0, slope = 1)
```




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

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

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

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

```
## [1] 136.0237
```

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

```
## [1] 0.002538695
```

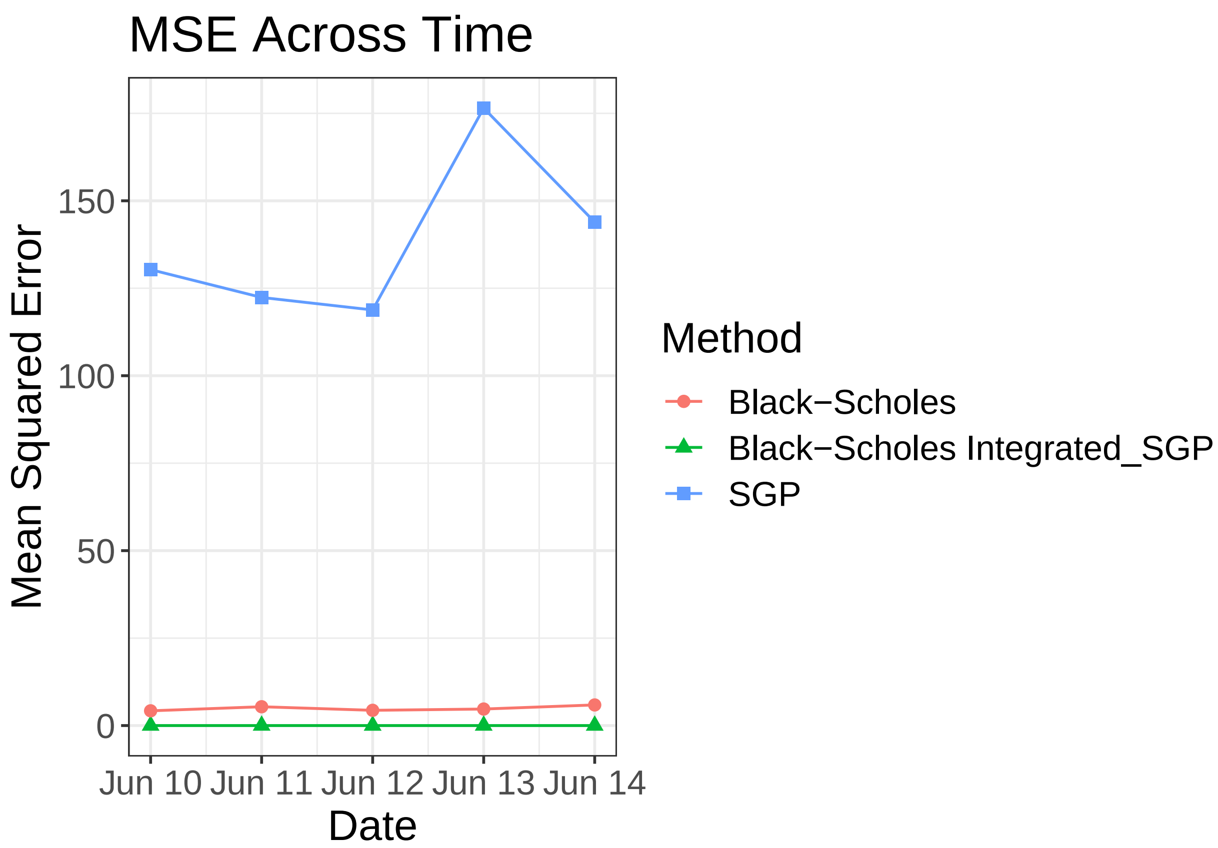
```
MSE(blackscholes_test,stan_dat$total_puts_American$mid_price[test_start:test_end])
```

```
## [1] 4.923348
```

##Part 4 MSE over Time

4-1: Plotting MSE over time

```
#MSE
library('MLmetrics')
SGP_MSE_0610 <- MSE(y_mean_values_SGP[1:71],stan_dat$total_puts_American$mid_price[323:393])
SGP_MSE_0611 <- MSE(y_mean_values_SGP[72:143],stan_dat$total_puts_American$mid_price[394:465])
SGP_MSE_0612 <- MSE(y_mean_values_SGP[144:164],stan_dat$total_puts_American$mid_price[466:486])
SGP_MSE_0613 <- MSE(y_mean_values_SGP[165:200],stan_dat$total_puts_American$mid_price[487:522])
SGP_MSE_0614 <- MSE(y_mean_values_SGP[201:237],stan_dat$total_puts_American$mid_price[523:559])
SGP_MSE <- rbind(SGP_MSE_0610,SGP_MSE_0611,SGP_MSE_0612,SGP_MSE_0613,SGP_MSE_0614)

BS_SGP_MSE_0610 <- MSE(y_mean_values_bs[1:71],stan_dat$total_puts_American$mid_price[323:393])
BS_SGP_MSE_0611 <- MSE(y_mean_values_bs[72:143],stan_dat$total_puts_American$mid_price[394:465])
BS_SGP_MSE_0612 <- MSE(y_mean_values_bs[144:164],stan_dat$total_puts_American$mid_price[466:486])
BS_SGP_MSE_0613 <- MSE(y_mean_values_bs[165:200],stan_dat$total_puts_American$mid_price[487:522])
BS_SGP_MSE_0614 <- MSE(y_mean_values_bs[201:237],stan_dat$total_puts_American$mid_price[523:559])
BS_SGP_MSE <- rbind(BS_SGP_MSE_0610,BS_SGP_MSE_0611,BS_SGP_MSE_0612,BS_SGP_MSE_0613,BS_SGP_MSE_0614)

BS_MSE_0610 <- MSE(blackscholes_test[1:71],stan_dat$total_puts_American$mid_price[323:393])
BS_MSE_0611 <- MSE(blackscholes_test[72:143],stan_dat$total_puts_American$mid_price[394:465])
BS_MSE_0612 <- MSE(blackscholes_test[144:164],stan_dat$total_puts_American$mid_price[466:486])
BS_MSE_0613 <- MSE(blackscholes_test[165:200],stan_dat$total_puts_American$mid_price[487:522])
BS_MSE_0614 <- MSE(blackscholes_test[201:237],stan_dat$total_puts_American$mid_price[523:559])
BS_MSE <- rbind(BS_MSE_0610,BS_MSE_0611,BS_MSE_0612,BS_MSE_0613,BS_MSE_0614)

MSE_data <- as.data.frame(rbind(SGP_MSE,BS_SGP_MSE,BS_MSE))

MSE_data$date <- c(as.Date('00/00/0000', format = '%m/%d/%Y'))
MSE_data$date[c(1,6,11)] <- as.Date('06/10/2019', format = '%m/%d/%Y')
MSE_data$date[c(2,7,12)] <- as.Date('06/11/2019', format = '%m/%d/%Y')
MSE_data$date[c(3,8,13)] <- as.Date('06/12/2019', format = '%m/%d/%Y')
MSE_data$date[c(4,9,14)] <- as.Date('06/13/2019', format = '%m/%d/%Y')
MSE_data$date[c(5,10,15)] <- as.Date('06/14/2019', format = '%m/%d/%Y')

MSE_data$Method <- "Method"
MSE_data$Method[1:5] <- "SGP"
MSE_data$Method[6:10] <- "Black-Scholes Integrated_SGP"
MSE_data$Method[11:15] <- "Black-Scholes"

MSE_data <- MSE_data %>%
  rename(MSE_values = V1)

#Plotting Blackscholes
ggplot(data=MSE_data, mapping=aes(x=date, y=MSE_values, colour = Method, shape = Method)) +
  geom_point(size = 2) +
  geom_line() +
  labs(title = "MSE Across Time",
       x = "Date",
       y = "Mean Squared Error") +
  theme_bw() +
  theme(text=element_text(size=16))
```



##Part 5 Interpretations

5-1: Contour Plots of Forward Price & Strike Price

```
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(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$total_puts_American$impl_volatility_scaled[test_start:tes
x.grid_4_cont <- as.numeric(rep(mean(stan_dat$total_puts_American$time_to_exp_scaled[test_start:test_en
x.grid_5_cont <- as.numeric(rep(mean(stan_dat$total_puts_American$dividend_yield_scaled[test_start:test_
x.grid_6_cont <- as.numeric(rep(mean(stan_dat$total_puts_American$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$total_puts_American$forward_price[test_start:test_end])
x.grid_2_cont_bs <- as.numeric(stan_dat$total_puts_American$strike_price[test_start:test_end])
x.grid_3_cont_bs <- as.numeric(rep(mean(stan_dat$total_puts_American$impl_volatility[test_start:test_en
x.grid_4_cont_bs <- as.numeric(rep(mean(stan_dat$total_puts_American$time_to_exp[test_start:test_end]))
x.grid_5_cont_bs <- as.numeric(rep(mean(stan_dat$total_puts_American$dividend_yield[test_start:test_end
x.grid_6_cont_bs <- as.numeric(rep(mean(stan_dat$total_puts_American$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)
```

```r
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=
}

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_bs <- list(theta=c(post_mean_theta_1_bs,post_mean_theta_2_bs,post_mean_theta_3_bs,post_me


# post_data

pred_gp_cont_SGP <- stan(file="Predictive GP_6dimension_SGP.stan", data=post_data_cont_SGP,iter=200, wa
```

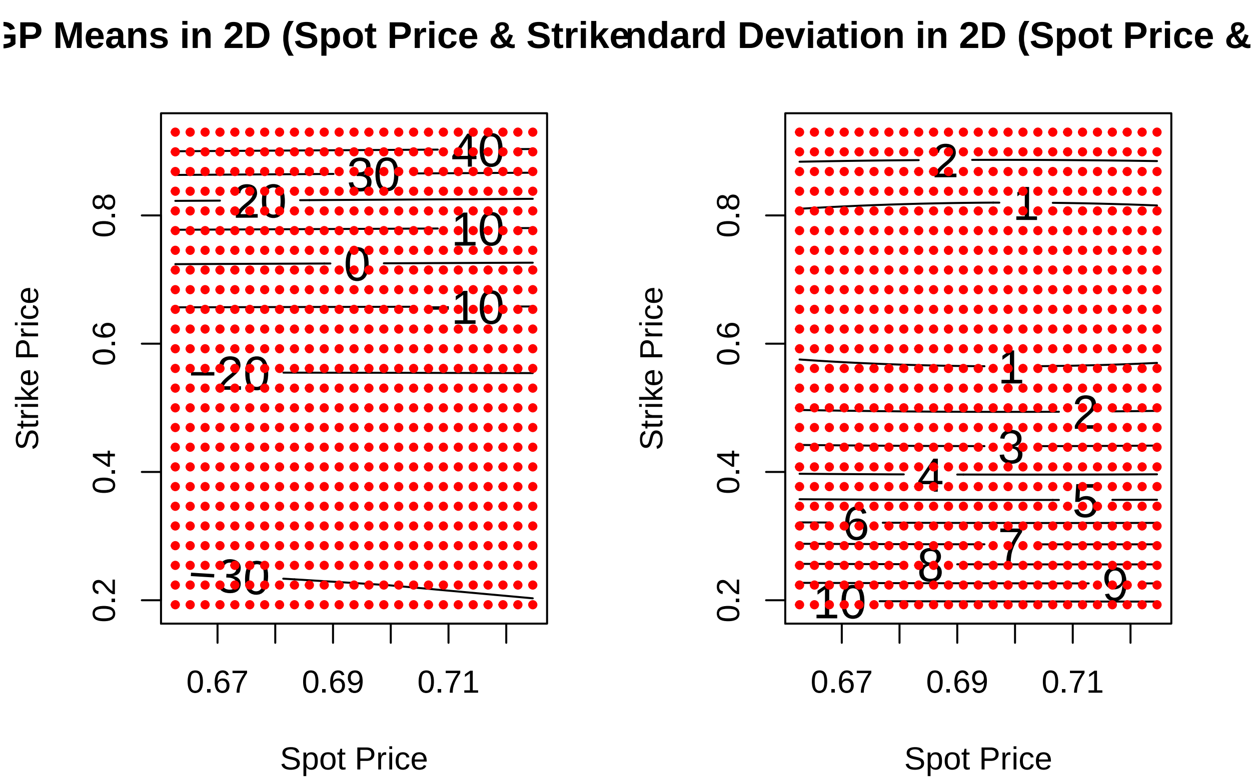
```
## DIAGNOSTIC(S) FROM PARSER:
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line comments.
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line comments.
##
##
## SAMPLING FOR MODEL 'Predictive GP_6dimension_SGP' NOW (CHAIN 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:                59.554 seconds (Sampling)
## Chain 1:                59.554 seconds (Total)
## Chain 1:
```

```r
pred_gp_cont_bs <- stan(file="Predictive GP_6dimension_withBS_SGP.stan", data=post_data_cont_bs,iter=20
```

```
## 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_SGP' 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:                58.381 seconds (Sampling)
## Chain 1:                58.381 seconds (Total)
## Chain 1:
```

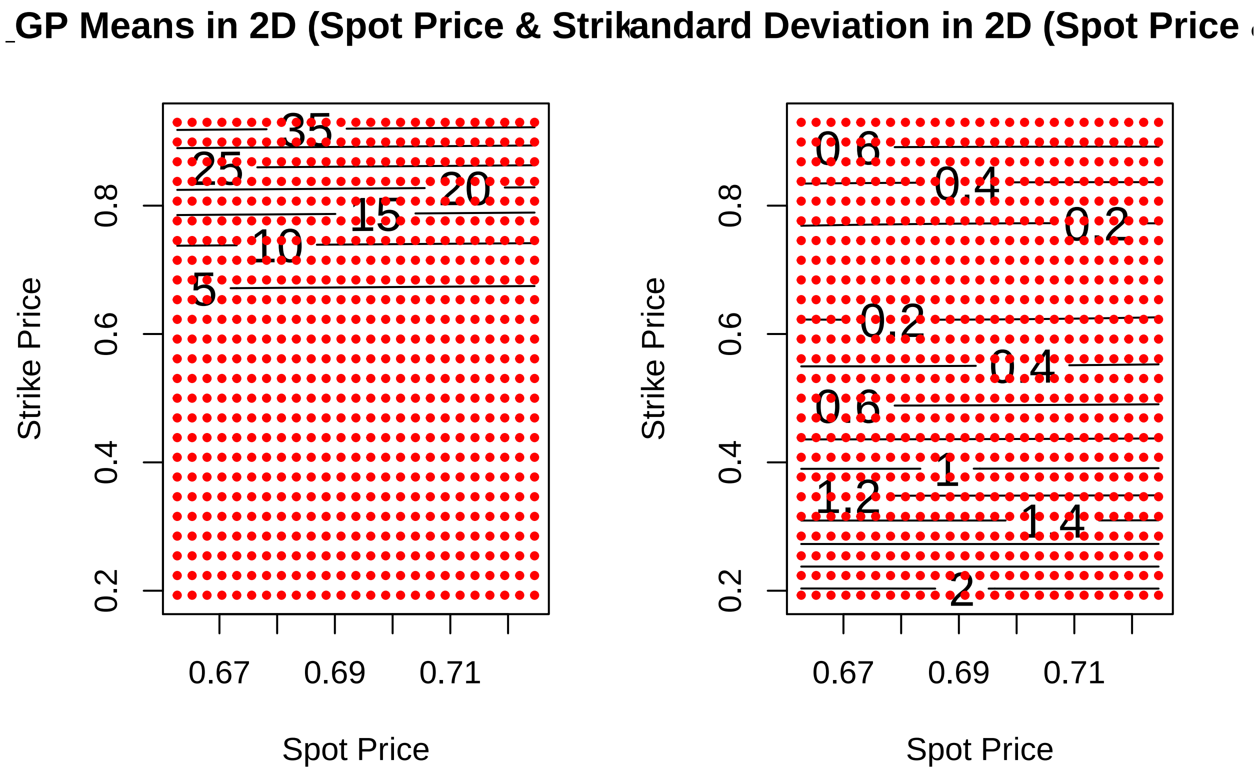
```r
#Computing Mean
y_predict_values_cont_SGP <- extract(pred_gp_cont_SGP,permuted=FALSE)
```

```
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)), labcex =1.5, main = "SGI
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)), labcex =1.5, main = "SGP :
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
```




```
#Computing Mean
y_predict_values_cont_bs <- extract(pred_gp_cont_bs,permuted=FALSE)
y_mean_values_cont_bs <- c(colMeans(y_predict_values_cont_bs))
y_mean_values_cont_bs <- y_mean_values_cont_bs[1:(length(y_mean_values_cont_bs)-1)]

#Computing Standard Deviation
pred_gp_summary_cont_bs <- summary(pred_gp_cont_bs, sd=c("sd"))$summary
pred_gp_sd_cont_bs <- pred_gp_summary_cont_bs[, c("sd")]
y_sd_values_cont_bs <- pred_gp_sd_cont_bs[1:(length(pred_gp_sd_cont_bs)-1)]
```

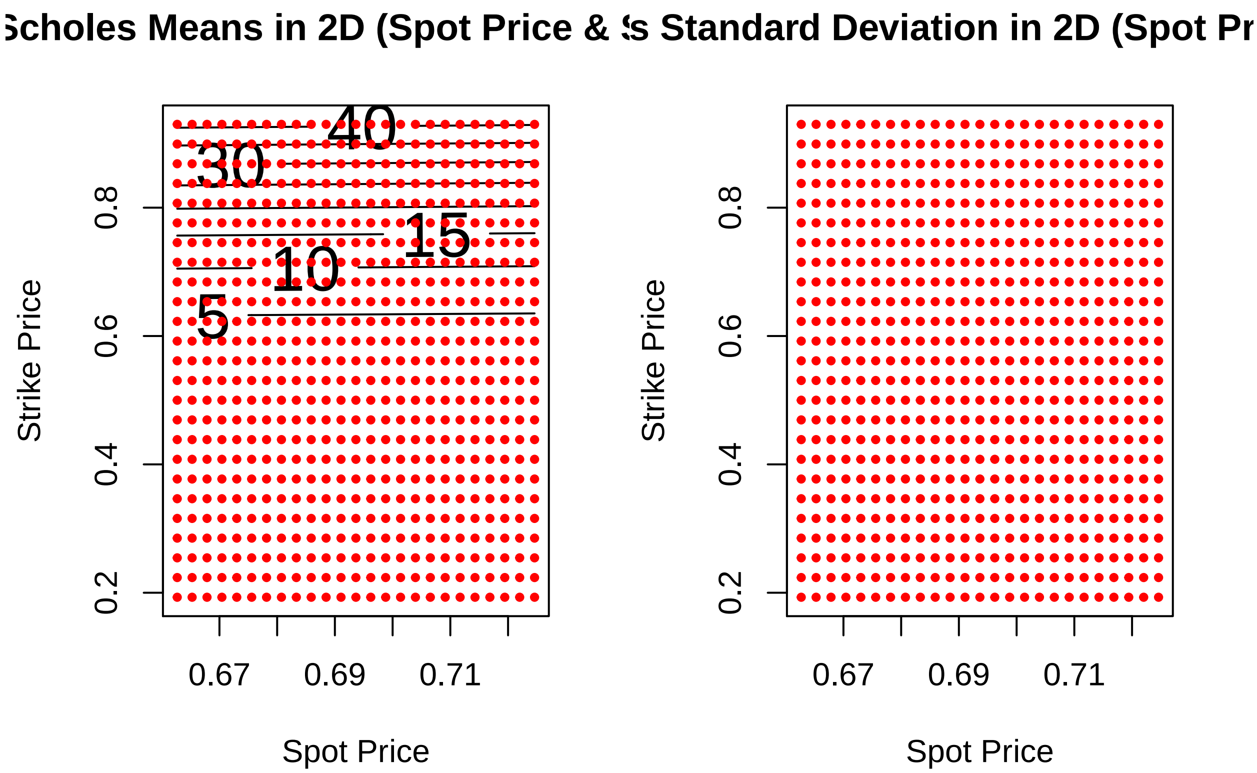
```r
par(mfrow = c(1, 2))
#Contour for Predictions aka mean values of predicitons
contour(dim1, dim2, matrix(y_mean_values_cont_bs, length(dim1), length(dim2)), labcex =1.5, main = "BS_(
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
#Contour of Variance
contour(dim1, dim2, matrix(y_sd_values_cont_bs, length(dim1), length(dim2)), labcex =1.5, main = "BS_GP
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
```




```r
par(mfrow = c(1, 2))
#Contour for Predictions aka mean values of predicitons
contour(dim1, dim2, matrix(blackscholes_test_cont, length(dim1), length(dim2)), labcex = 2, main = "Blac
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
#Contour of Variance
contour(dim1, dim2, matrix(sd(blackscholes_test_cont), length(dim1), length(dim2)), labcex =1.5, main =
```

```
## Warning in contour.default(dim1, dim2, matrix(sd(blackscholes_test_cont), :
## all z values are equal
```

```r
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
```



5-2: Contour Plots of Implied Volatility & Time to Expiration

```
x.grid_1_cont <- as.numeric(stan_dat$total_puts_American$impl_volatility_scaled[test_start:test_end])
x.grid_2_cont <- as.numeric(stan_dat$total_puts_American$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$total_puts_American$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_scaled[test_start:test_
x.grid_6_cont <- as.numeric(rep(mean(stan_dat$total_puts_American$interest_rate_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$total_puts_American$impl_volatility[test_start:test_end])
x.grid_2_cont_bs <- as.numeric(stan_dat$total_puts_American$time_to_exp[test_start:test_end])
x.grid_3_cont_bs <- as.numeric(rep(mean(stan_dat$total_puts_American$forward_price[test_start:test_end]
x.grid_4_cont_bs <- as.numeric(rep(mean(stan_dat$total_puts_American$strike_price[test_start:test_end])
x.grid_5_cont_bs <- as.numeric(rep(mean(stan_dat$total_puts_American$dividend_yield[test_start:test_end
x.grid_6_cont_bs <- as.numeric(rep(mean(stan_dat$total_puts_American$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,3],K=x2_cont_bs[row,4],r=
}


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_bs <- list(theta=c(post_mean_theta_1_bs,post_mean_theta_2_bs,post_mean_theta_3_bs,post_me


# post_data

pred_gp_cont_SGP <- stan(file="Predictive GP_6dimension_SGP.stan", data=post_data_cont_SGP,iter=200, wa
```

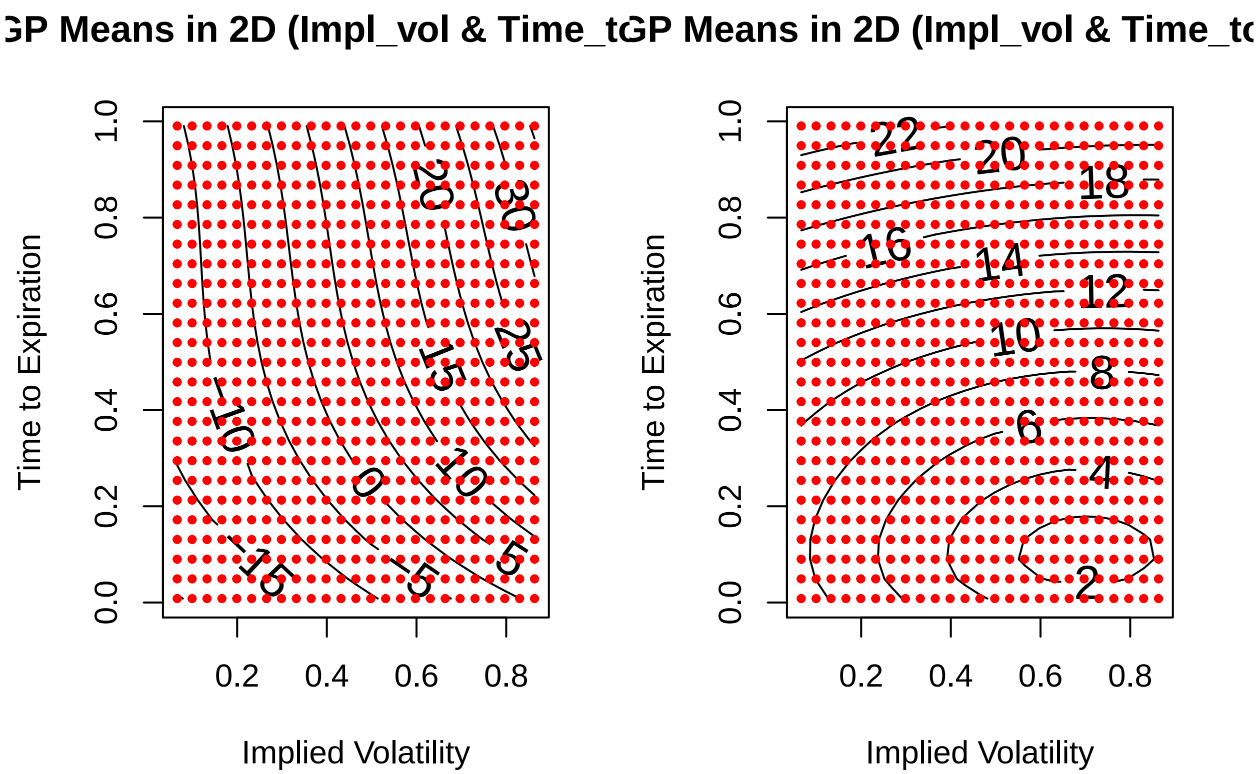
```
## DIAGNOSTIC(S) FROM PARSER:
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line comments.
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line comments.
##
##
## SAMPLING FOR MODEL 'Predictive GP_6dimension_SGP' NOW (CHAIN 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:                58.443 seconds (Sampling)
## Chain 1:                58.443 seconds (Total)
## Chain 1:
```

```
pred_gp_cont_bs <- stan(file="Predictive GP_6dimension_withBS_SGP.stan", data=post_data_cont_bs,iter=20
```

```
## 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_SGP' 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:                57.612 seconds (Sampling)
## Chain 1:                57.612 seconds (Total)
## Chain 1:
```

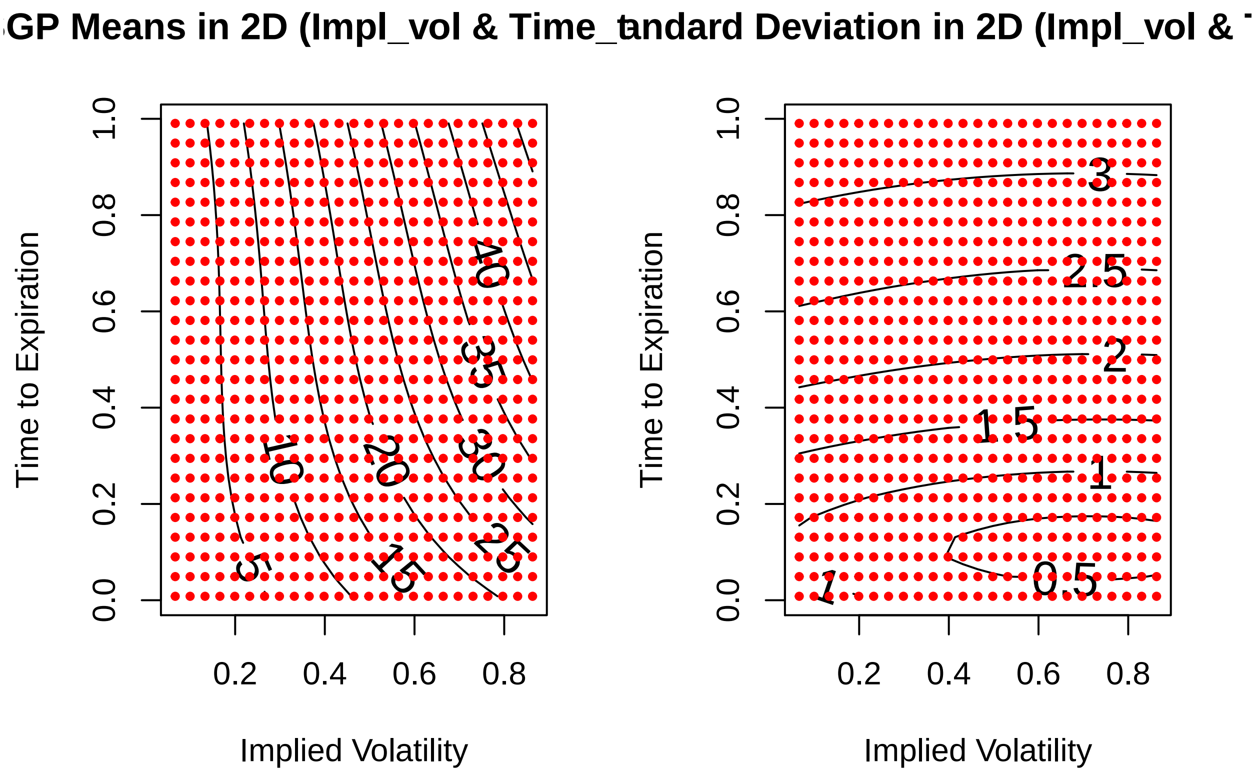
```
#Computing Mean
y_predict_values_cont_SGP <- extract(pred_gp_cont_SGP,permuted=FALSE)
y_mean_values_cont_SGP <- c(colMeans(y_predict_values_cont_SGP))
```

```r
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)), labcex =1.5, main = "SGI
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)), labcex =1.5, main = "SGP I
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
```




```r
#Computing Mean
y_predict_values_cont_bs <- extract(pred_gp_cont_bs,permuted=FALSE)
y_mean_values_cont_bs <- c(colMeans(y_predict_values_cont_bs))
y_mean_values_cont_bs <- y_mean_values_cont_bs[1:(length(y_mean_values_cont_bs)-1)]

#Computing Standard Deviation
pred_gp_summary_cont_bs <- summary(pred_gp_cont_bs, sd=c("sd"))$summary
pred_gp_sd_cont_bs <- pred_gp_summary_cont_bs[, c("sd")]
y_sd_values_cont_bs <- pred_gp_sd_cont_bs[1:(length(pred_gp_sd_cont_bs)-1)]
```

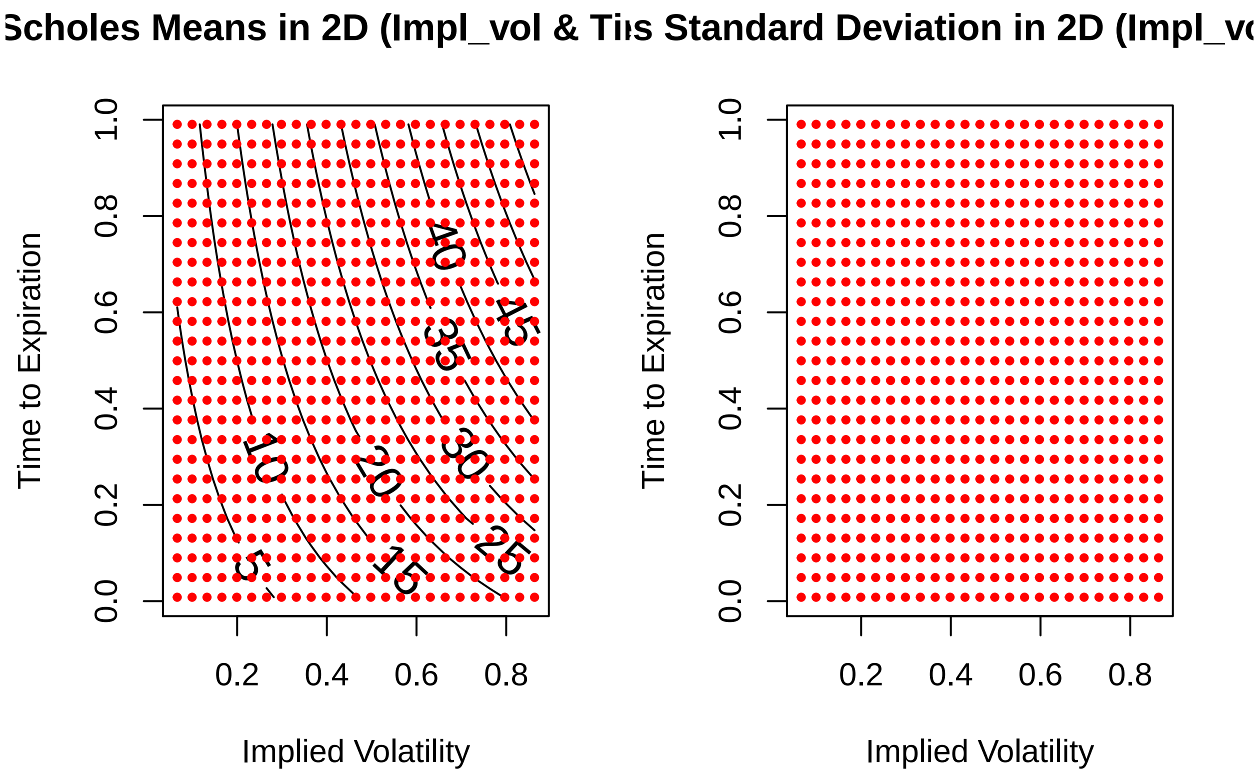
```r
par(mfrow = c(1, 2))
#Contour for Predictions aka mean values of predicitons
contour(dim1, dim2, matrix(y_mean_values_cont_bs, length(dim1), length(dim2)), labcex =1.5, main = "BSGP
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
#Contour of Variance
contour(dim1, dim2, matrix(y_sd_values_cont_bs, length(dim1), length(dim2)), labcex =1.5, main = "BSGP S
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
```




```r
par(mfrow = c(1, 2))
#Contour for Predictions aka mean values of predicitons
contour(dim1, dim2, matrix(blackscholes_test_cont, length(dim1), length(dim2)), labcex =1.5, main = "Bla
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
#Contour of Variance
contour(dim1, dim2, matrix(sd(blackscholes_test_cont), length(dim1), length(dim2)), labcex =1.5, main =
```

```
## Warning in contour.default(dim1, dim2, matrix(sd(blackscholes_test_cont), :
## all z values are equal
```

```r
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
```



5-3: Contour Plots of Strike Price and Implied Volatility

```
x.grid_1_cont <- as.numeric(stan_dat$total_puts_American$strike_price_scaled[test_start:test_end])
x.grid_2_cont <- as.numeric(stan_dat$total_puts_American$impl_volatility_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$total_puts_American$forward_price_scaled[test_start:test_e
x.grid_4_cont <- as.numeric(rep(mean(stan_dat$total_puts_American$time_to_exp_scaled[test_start:test_en
x.grid_5_cont <- as.numeric(rep(mean(stan_dat$total_puts_American$dividend_yield_scaled[test_start:test_
x.grid_6_cont <- as.numeric(rep(mean(stan_dat$total_puts_American$interest_rate_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$total_puts_American$strike_price[test_start:test_end])
x.grid_2_cont_bs <- as.numeric(stan_dat$total_puts_American$impl_volatility[test_start:test_end])
x.grid_3_cont_bs <- as.numeric(rep(mean(stan_dat$total_puts_American$forward_price[test_start:test_end]
x.grid_4_cont_bs <- as.numeric(rep(mean(stan_dat$total_puts_American$time_to_exp[test_start:test_end]))
x.grid_5_cont_bs <- as.numeric(rep(mean(stan_dat$total_puts_American$dividend_yield[test_start:test_end
x.grid_6_cont_bs <- as.numeric(rep(mean(stan_dat$total_puts_American$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,1],r=
}


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_bs <- list(theta=c(post_mean_theta_1_bs,post_mean_theta_2_bs,post_mean_theta_3_bs,post_me


# post_data

pred_gp_cont_SGP <- stan(file="Predictive GP_6dimension_SGP.stan", data=post_data_cont_SGP,iter=200, wa
```

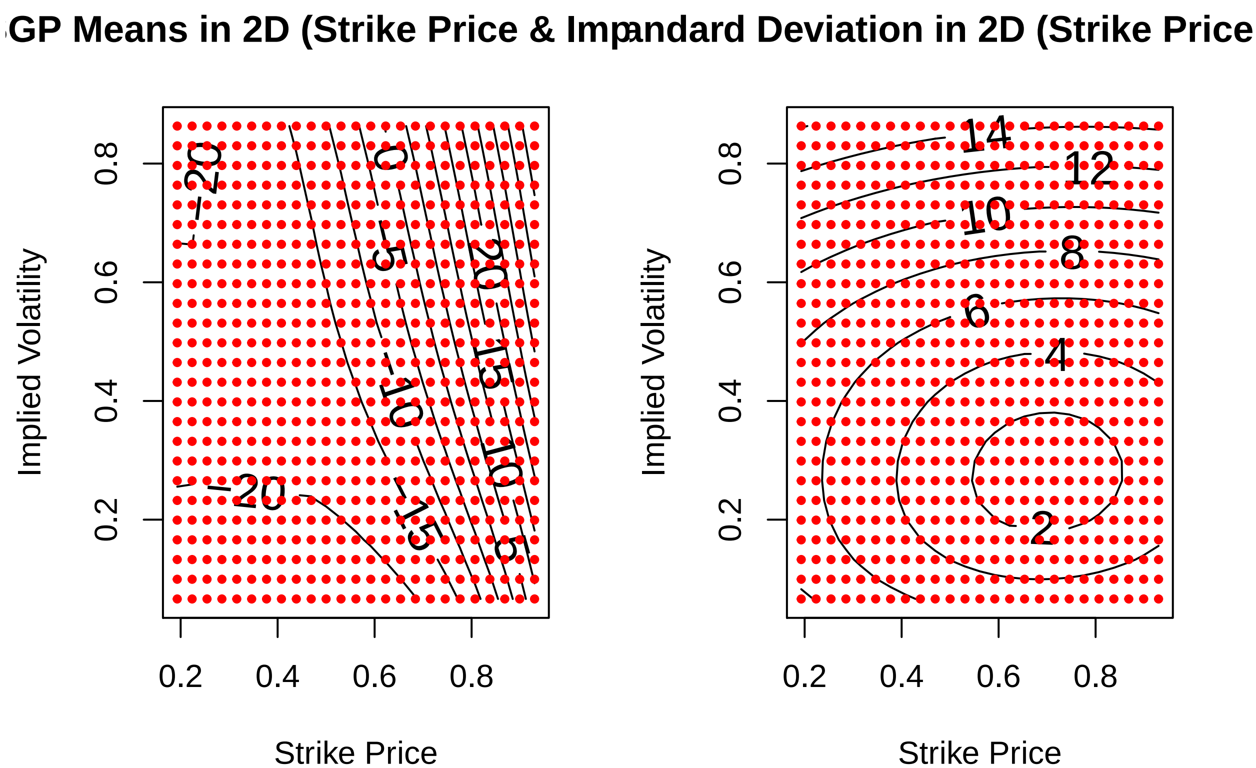
```
## DIAGNOSTIC(S) FROM PARSER:
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line comments.
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line comments.
##
##
## SAMPLING FOR MODEL 'Predictive GP_6dimension_SGP' NOW (CHAIN 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:                58.443 seconds (Sampling)
## Chain 1:                58.443 seconds (Total)
## Chain 1:
```

```r
pred_gp_cont_bs <- stan(file="Predictive GP_6dimension_withBS_SGP.stan", data=post_data_cont_bs,iter=20
```

```
## 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_SGP' 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:                57.944 seconds (Sampling)
## Chain 1:                57.944 seconds (Total)
## Chain 1:
```

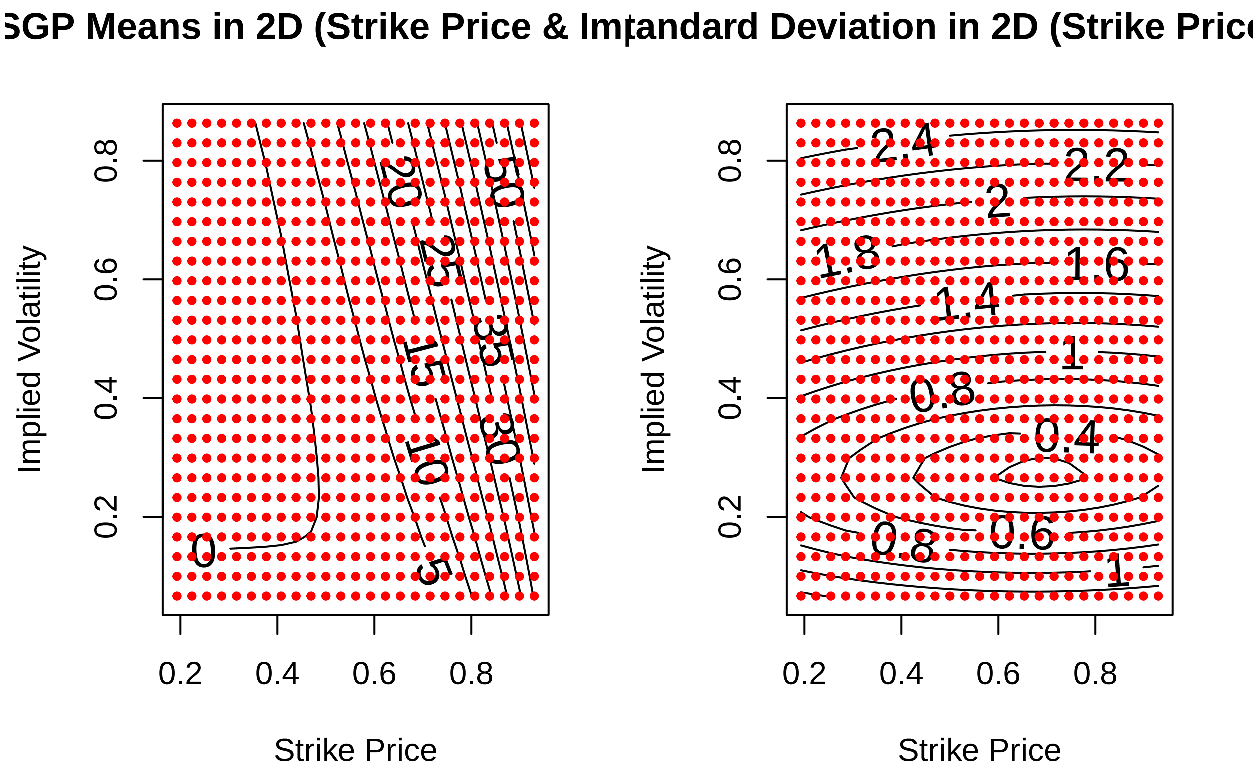
```r
#Computing Mean
y_predict_values_cont_SGP <- extract(pred_gp_cont_SGP,permuted=FALSE)
y_mean_values_cont_SGP <- c(colMeans(y_predict_values_cont_SGP))
```

```
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)), labcex =1.5, main = "SG
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)), labcex =1.5, main = "SGP
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
```




```
#Computing Mean
y_predict_values_cont_bs <- extract(pred_gp_cont_bs,permuted=FALSE)
y_mean_values_cont_bs <- c(colMeans(y_predict_values_cont_bs))
y_mean_values_cont_bs <- y_mean_values_cont_bs[1:(length(y_mean_values_cont_bs)-1)]

#Computing Standard Deviation
pred_gp_summary_cont_bs <- summary(pred_gp_cont_bs, sd=c("sd"))$summary
pred_gp_sd_cont_bs <- pred_gp_summary_cont_bs[, c("sd")]
y_sd_values_cont_bs <- pred_gp_sd_cont_bs[1:(length(pred_gp_sd_cont_bs)-1)]
```

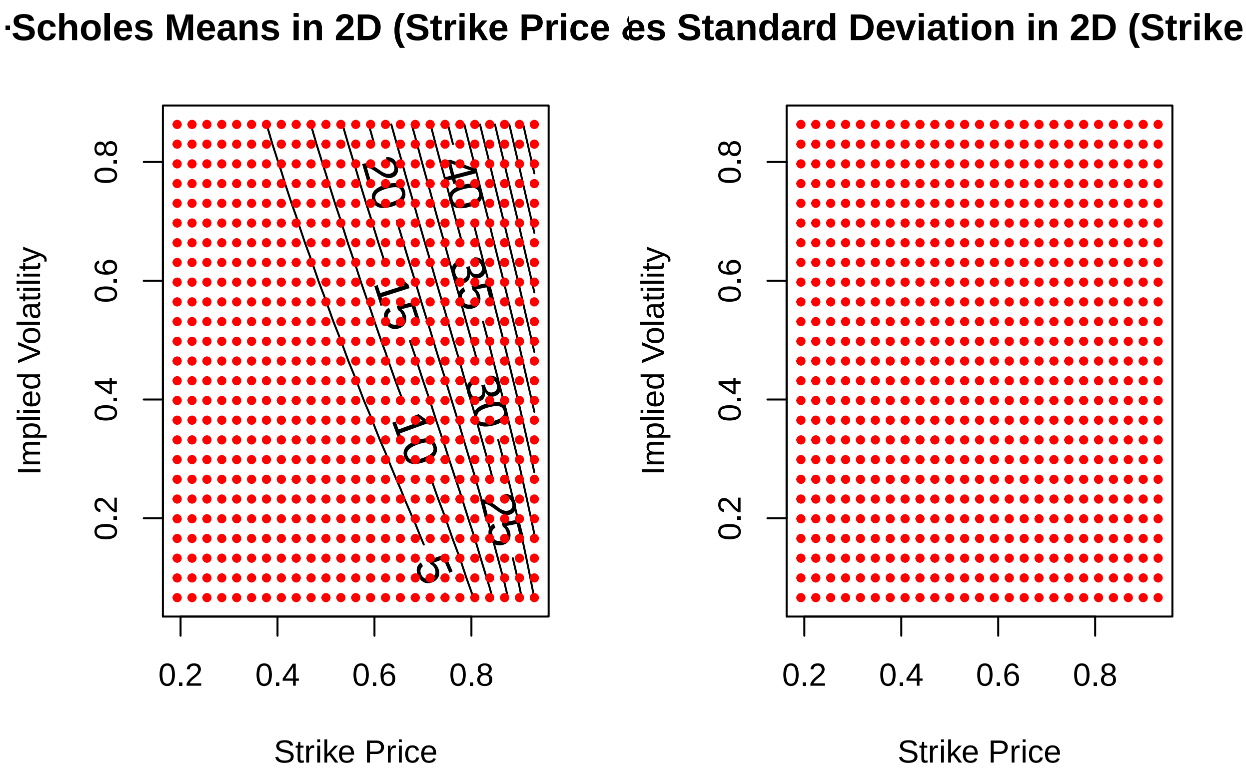
```r
par(mfrow = c(1, 2))
#Contour for Predictions aka mean values of predicitons
contour(dim1, dim2, matrix(y_mean_values_cont_bs, length(dim1), length(dim2)), labcex =1.5, main = "BSGP
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
#Contour of Variance
contour(dim1, dim2, matrix(y_sd_values_cont_bs, length(dim1), length(dim2)), labcex =1.5, main = "BSGP S
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
```




```r
par(mfrow = c(1, 2))
#Contour for Predictions aka mean values of predicitons
contour(dim1, dim2, matrix(blackscholes_test_cont, length(dim1), length(dim2)), labcex =1.5, main = "Bla
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
#Contour of Variance
contour(dim1, dim2, matrix(sd(blackscholes_test_cont), length(dim1), length(dim2)), labcex =1.5, main =
```

```
## Warning in contour.default(dim1, dim2, matrix(sd(blackscholes_test_cont), :
## all z values are equal
```

```r
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
```

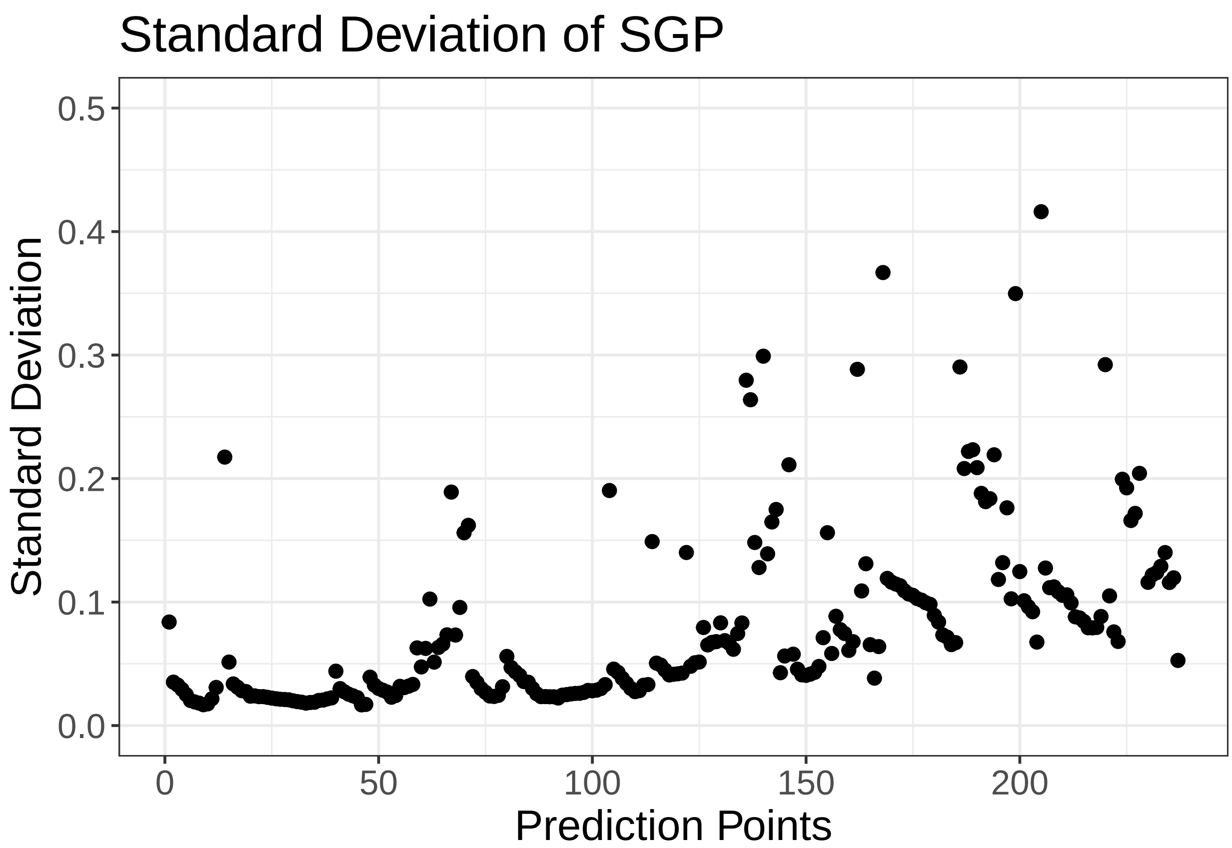


Part 6: Standard Deviation of Predicted Values

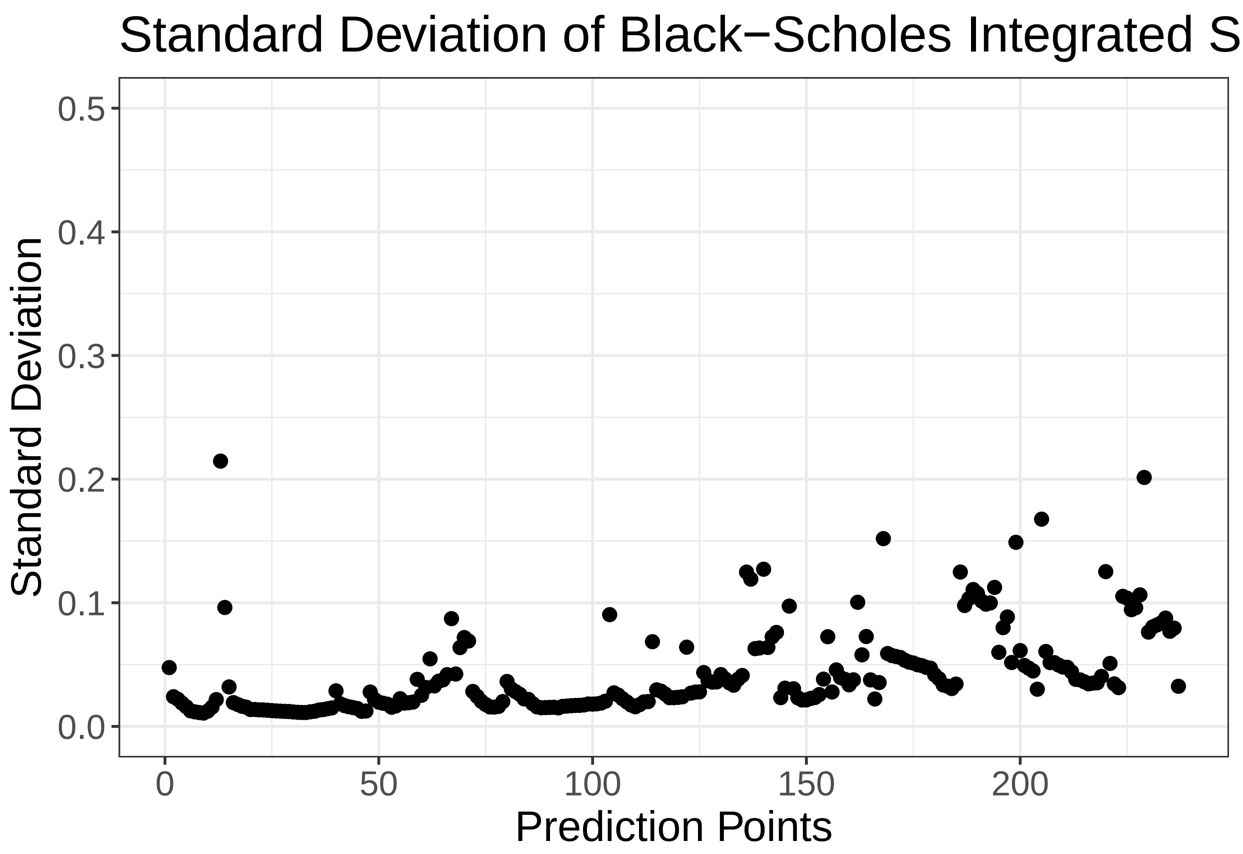
6-1: Plotting Standard Deviation of Predictions

```
par(mfrow=c(1,3))
ggplot(mapping = aes(x = seq(1:237),y=y_sd_values_SGP)) + geom_point(size = 2)+
  ylim(0,0.5)+ labs(title = "Standard Deviation of SGP",
       x = "Prediction Points",
       y = "Standard Deviation") +
  theme_bw() +
  theme(text=element_text(size=16))
```

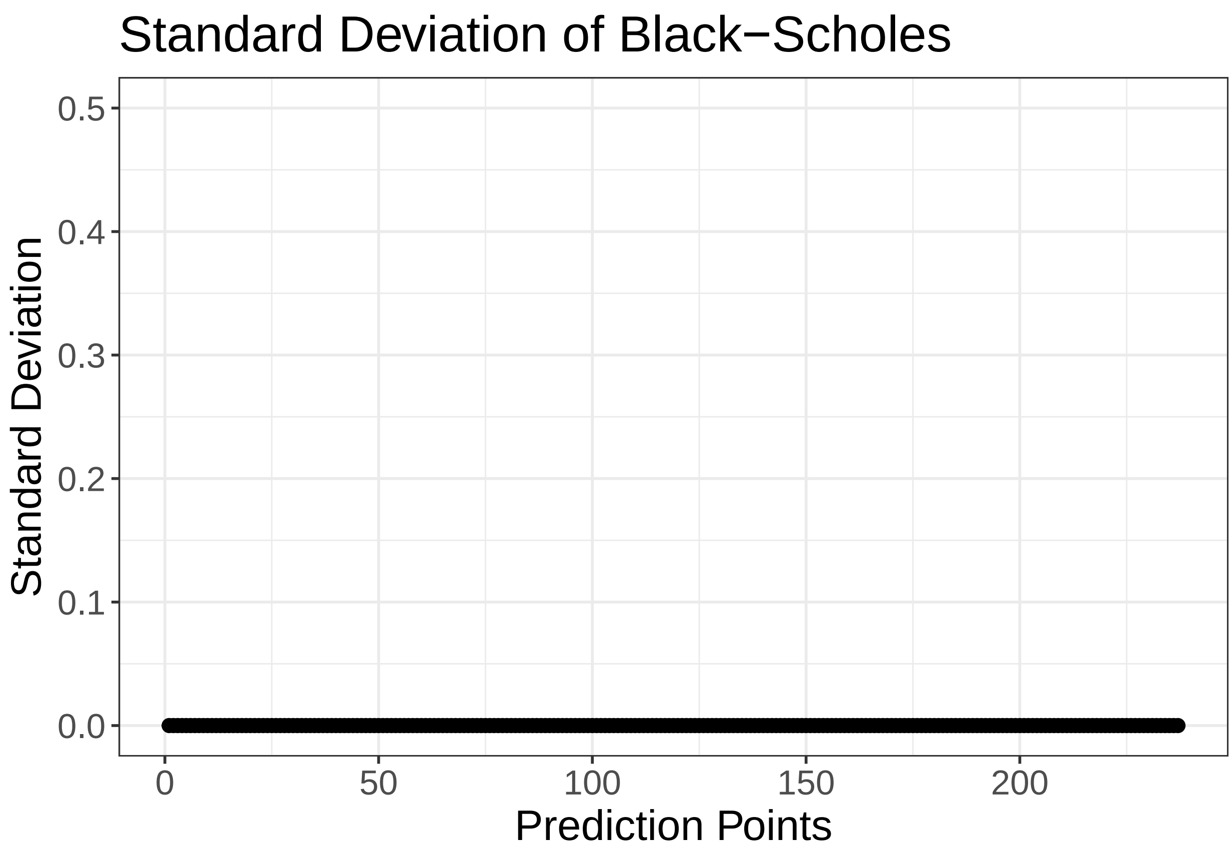
```
## Warning: Removed 2 rows containing missing values (geom_point).
```




```r
ggplot(mapping = aes(x = seq(1:237),y=y_sd_values_bs)) + geom_point(size = 2)+
  ylim(0,0.5)+
  labs(title = "Standard Deviation of Black-Scholes Integrated SGP",
       x = "Prediction Points",
       y = "Standard Deviation") +
  theme_bw() +
  theme(text=element_text(size=16))
```




```r
ggplot(mapping = aes(x = seq(1:237),y=rep(0,237))) + geom_point(size = 2)+
  ylim(0,0.5)+
  labs(title = "Standard Deviation of Black-Scholes",
       x = "Prediction Points",
       y = "Standard Deviation") +
  theme_bw() +
  theme(text=element_text(size=16))
```




```
mean(y_sd_values_SGP)
```

```
## [1] 0.08564898
```

```
mean(y_sd_values_bs)
```

```
## [1] 0.04273671
```

```
mean(0)
```

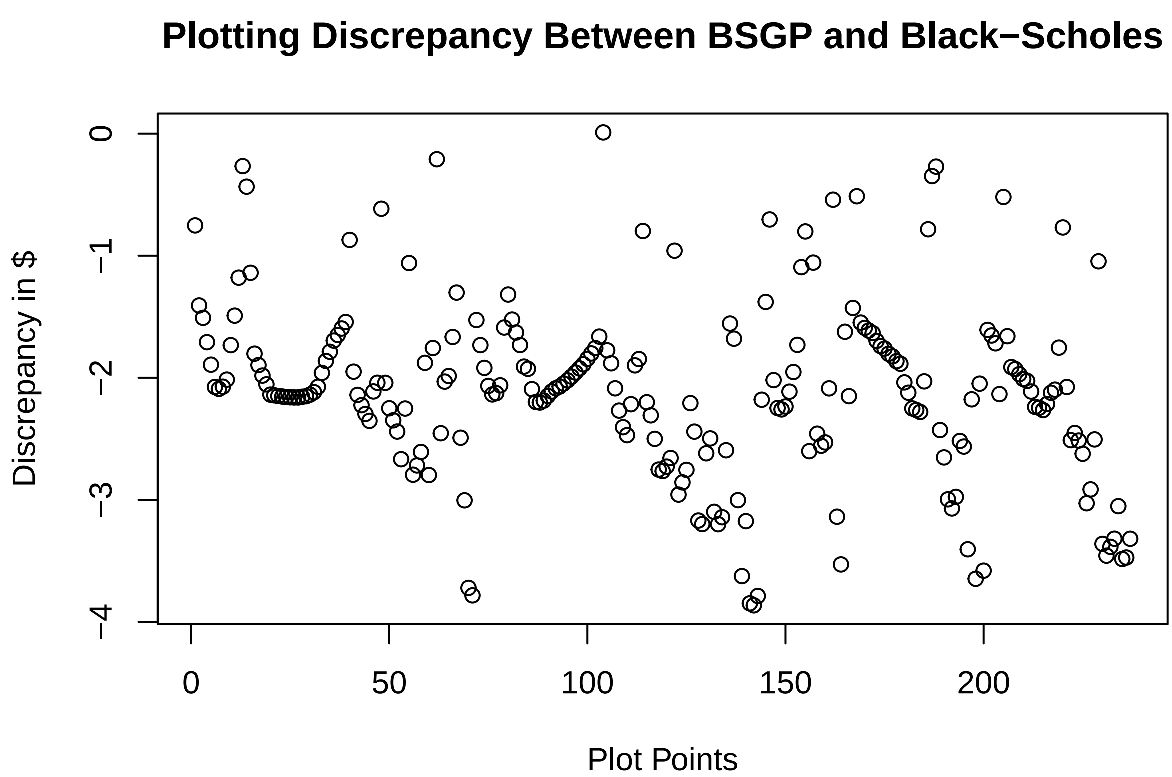
```
## [1] 0
```

##Part 7 Discrepancy Modeling

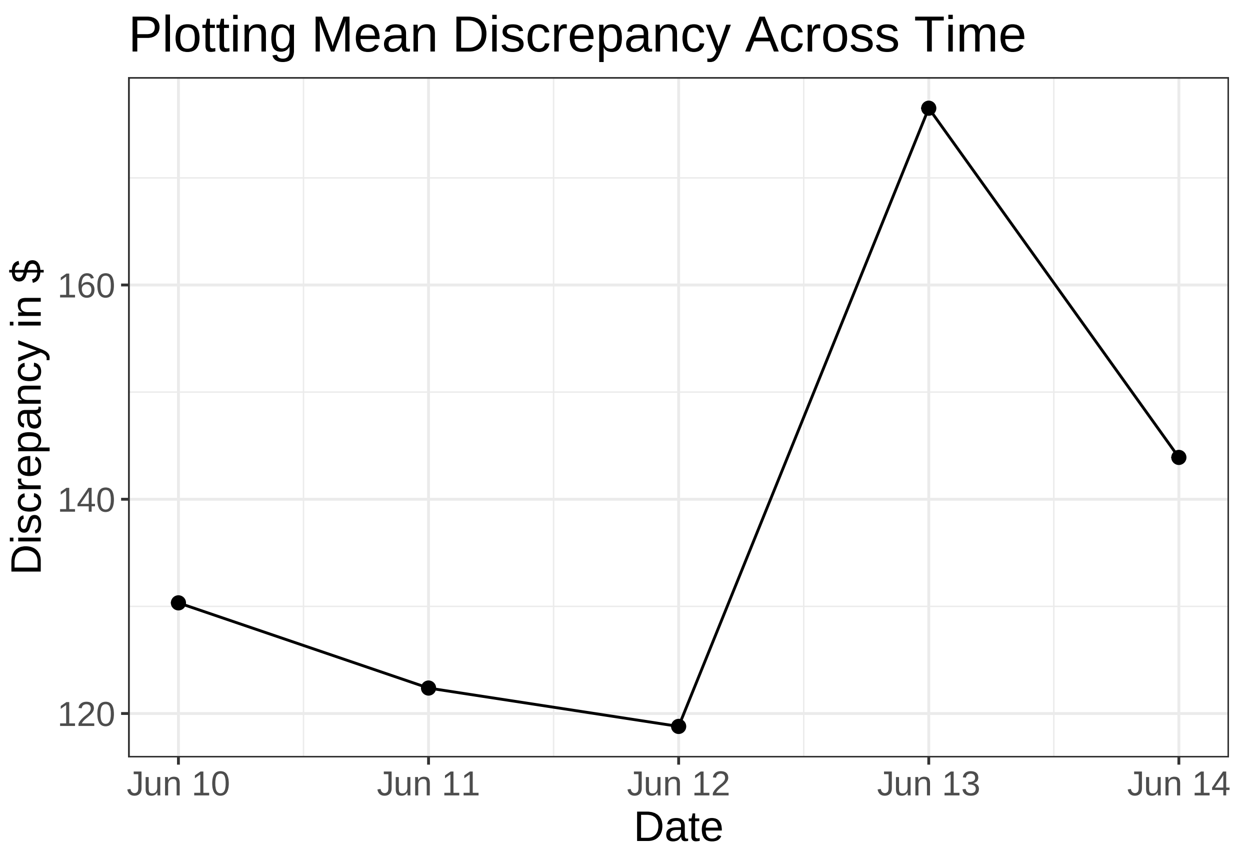
7-1: Computing Discrepancy

```
Discrepancy <- y_mean_values_bs - blackscholes_test

library('MLmetrics')
Discrepnacy_0610 <- mean(Discrepancy[1:71])
Discrepancy_0611 <- mean(Discrepancy[72:143])
Discrepancy_0612 <- mean(Discrepancy[144:164])
Discrepancy_0613 <- mean(Discrepancy[165:200])
Discrepancy_0614 <- mean(Discrepancy[201:237])
```

```
Discrepancy_data <- as.data.frame(rbind(SGP_MSE_0610,SGP_MSE_0611,SGP_MSE_0612,SGP_MSE_0613,SGP_MSE_0614

Discrepancy_data$date <- c(as.Date('00/00/0000', format = '%m/%d/%Y'))
Discrepancy_data$date[1] <- as.Date('06/10/2019', format = '%m/%d/%Y')
Discrepancy_data$date[2] <- as.Date('06/11/2019', format = '%m/%d/%Y')
Discrepancy_data$date[3] <- as.Date('06/12/2019', format = '%m/%d/%Y')
Discrepancy_data$date[4] <- as.Date('06/13/2019', format = '%m/%d/%Y')
Discrepancy_data$date[5] <- as.Date('06/14/2019', format = '%m/%d/%Y')

Discrepancy_data <- Discrepancy_data %>%
  rename(Discrepancy_values = V1)

plot(Discrepancy, main = "Plotting Discrepancy Between BSGP and Black-Scholes", xlab = "Plot Points", y
```




```
#Plotting Blackscholes
ggplot(data=Discrepancy_data, mapping=aes(x=date, y=Discrepancy_values)) +
  geom_point(size = 2) +
  geom_line() +
  labs(title = "Plotting Mean Discrepancy Across Time",
       x = "Date",
       y = "Discrepancy in $") +
  theme_bw() +
  theme(text=element_text(size=16))
```



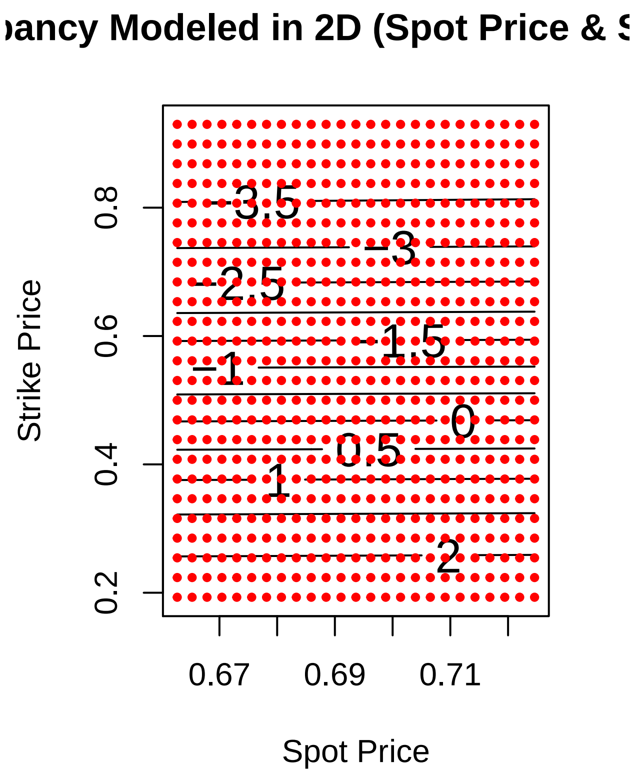
7-2: Discrepancy Contour Plots of Forward Price & Strike Price

```
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(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$total_puts_American$impl_volatility_scaled[test_start:tes
x.grid_4_cont <- as.numeric(rep(mean(stan_dat$total_puts_American$time_to_exp_scaled[test_start:test_en
x.grid_5_cont <- as.numeric(rep(mean(stan_dat$total_puts_American$dividend_yield_scaled[test_start:test_
x.grid_6_cont <- as.numeric(rep(mean(stan_dat$total_puts_American$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$total_puts_American$forward_price[test_start:test_end])
x.grid_2_cont_bs <- as.numeric(stan_dat$total_puts_American$strike_price[test_start:test_end])
x.grid_3_cont_bs <- as.numeric(rep(mean(stan_dat$total_puts_American$impl_volatility[test_start:test_en
x.grid_4_cont_bs <- as.numeric(rep(mean(stan_dat$total_puts_American$time_to_exp[test_start:test_end]))
x.grid_5_cont_bs <- as.numeric(rep(mean(stan_dat$total_puts_American$dividend_yield[test_start:test_end
x.grid_6_cont_bs <- as.numeric(rep(mean(stan_dat$total_puts_American$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,1],K=x2_cont_bs[row,2],r=
}

post_data_cont_bs <- list(theta=c(post_mean_theta_1_bs,post_mean_theta_2_bs,post_mean_theta_3_bs,post_m


# post_data


pred_gp_cont_bs <- stan(file="Predictive GP_6dimension_withBS_SGP.stan", data=post_data_cont_bs,iter=20
```

```
## 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_SGP' 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:                59.753 seconds (Sampling)
## Chain 1:                59.753 seconds (Total)
## Chain 1:
```

```r
#Computing Mean
y_predict_values_cont_bs <- extract(pred_gp_cont_bs,permuted=FALSE)
y_mean_values_cont_bs <- c(colMeans(y_predict_values_cont_bs))
y_mean_values_cont_bs <- y_mean_values_cont_bs[1:(length(y_mean_values_cont_bs)-1)]

#Computing Standard Deviation
pred_gp_summary_cont_bs <- summary(pred_gp_cont_bs, sd=c("sd"))$summary
pred_gp_sd_cont_bs <- pred_gp_summary_cont_bs[, c("sd")]
y_sd_values_cont_bs <- pred_gp_sd_cont_bs[1:(length(pred_gp_sd_cont_bs)-1)]

#Discrepancy
Discrepancy_function <- y_mean_values_cont_bs - blackscholes_test_cont

par(mfrow = c(1, 2))
#Contour for Predictions aka mean values of predicitons
contour(dim1, dim2, matrix(Discrepancy_function, length(dim1), length(dim2)), labcex =1.5, main = "Disc
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
```



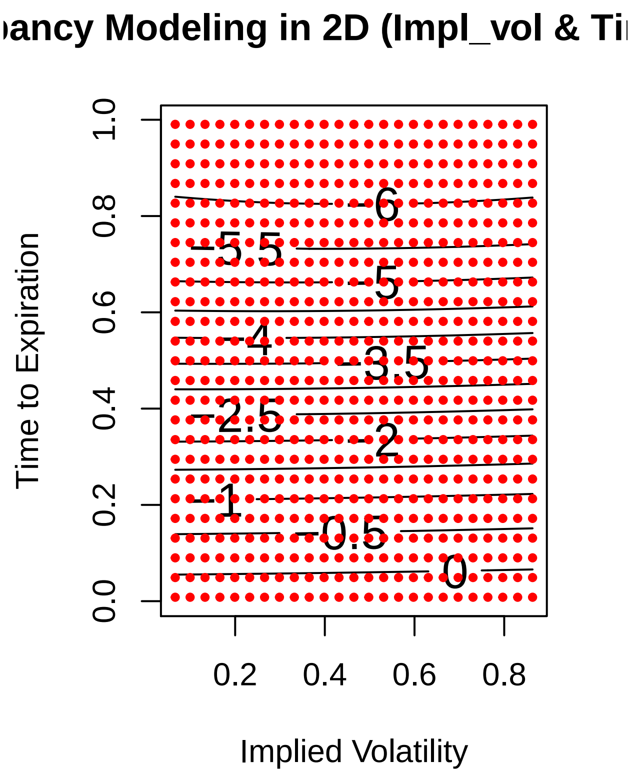
7-3: Discrepancy Contour Plots of Implied Volatility & Time to Expiration

```
x.grid_1_cont <- as.numeric(stan_dat$total_puts_American$impl_volatility_scaled[test_start:test_end])
x.grid_2_cont <- as.numeric(stan_dat$total_puts_American$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$total_puts_American$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_scaled[test_start:test_
x.grid_6_cont <- as.numeric(rep(mean(stan_dat$total_puts_American$interest_rate_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$total_puts_American$impl_volatility[test_start:test_end])
x.grid_2_cont_bs <- as.numeric(stan_dat$total_puts_American$time_to_exp[test_start:test_end])
x.grid_3_cont_bs <- as.numeric(rep(mean(stan_dat$total_puts_American$forward_price[test_start:test_end]
x.grid_4_cont_bs <- as.numeric(rep(mean(stan_dat$total_puts_American$strike_price[test_start:test_end])
x.grid_5_cont_bs <- as.numeric(rep(mean(stan_dat$total_puts_American$dividend_yield[test_start:test_end
x.grid_6_cont_bs <- as.numeric(rep(mean(stan_dat$total_puts_American$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,3],K=x2_cont_bs[row,4],r=
}


post_data_cont_bs <- list(theta=c(post_mean_theta_1_bs,post_mean_theta_2_bs,post_mean_theta_3_bs,post_me


# post_data

pred_gp_cont_bs <- stan(file="Predictive GP_6dimension_withBS_SGP.stan", data=post_data_cont_bs,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_SGP' 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:                57.827 seconds (Sampling)
## Chain 1:                57.827 seconds (Total)
## Chain 1:
```

```
#Computing Mean
y_predict_values_cont_bs <- extract(pred_gp_cont_bs,permuted=FALSE)
y_mean_values_cont_bs <- c(colMeans(y_predict_values_cont_bs))
y_mean_values_cont_bs <- y_mean_values_cont_bs[1:(length(y_mean_values_cont_bs)-1)]

#Computing Standard Deviation
pred_gp_summary_cont_bs <- summary(pred_gp_cont_bs, sd=c("sd"))$summary
pred_gp_sd_cont_bs <- pred_gp_summary_cont_bs[, c("sd")]
y_sd_values_cont_bs <- pred_gp_sd_cont_bs[1:(length(pred_gp_sd_cont_bs)-1)]

#Discrepancy
Discrepancy_function <- y_mean_values_cont_bs - blackscholes_test_cont

par(mfrow = c(1, 2))
#Contour for Predictions aka mean values of predicitons
contour(dim1, dim2, matrix(Discrepancy_function, length(dim1), length(dim2)), labcex =1.5, main = "Disc
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
```



7-4: Discrepancy Contour Plots of Strike Price & Implied Volatility

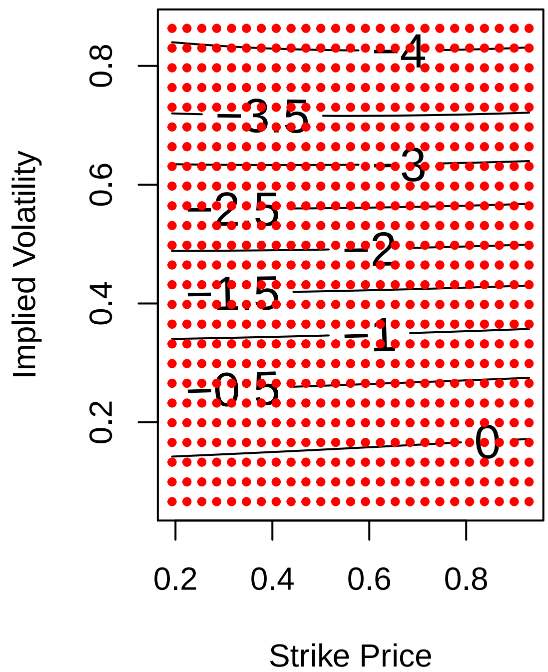
```
x.grid_1_cont <- as.numeric(stan_dat$total_puts_American$strike_price_scaled[test_start:test_end])
x.grid_2_cont <- as.numeric(stan_dat$total_puts_American$impl_volatility_scaled[test_start:test_end])

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$total_puts_American$forward_price_scaled[test_start:test_e
x.grid_4_cont <- as.numeric(rep(mean(stan_dat$total_puts_American$time_to_exp_scaled[test_start:test_en
x.grid_5_cont <- as.numeric(rep(mean(stan_dat$total_puts_American$dividend_yield_scaled[test_start:test_
x.grid_6_cont <- as.numeric(rep(mean(stan_dat$total_puts_American$interest_rate_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$total_puts_American$strike_price[test_start:test_end])
x.grid_2_cont_bs <- as.numeric(stan_dat$total_puts_American$impl_volatility[test_start:test_end])
x.grid_3_cont_bs <- as.numeric(rep(mean(stan_dat$total_puts_American$forward_price[test_start:test_end]
x.grid_4_cont_bs <- as.numeric(rep(mean(stan_dat$total_puts_American$time_to_exp[test_start:test_end]))
x.grid_5_cont_bs <- as.numeric(rep(mean(stan_dat$total_puts_American$dividend_yield[test_start:test_end
x.grid_6_cont_bs <- as.numeric(rep(mean(stan_dat$total_puts_American$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,1],r=
}


post_data_cont_bs <- list(theta=c(post_mean_theta_1_bs,post_mean_theta_2_bs,post_mean_theta_3_bs,post_me


# post_data

pred_gp_cont_bs <- stan(file="Predictive GP_6dimension_withBS_SGP.stan", data=post_data_cont_bs,iter=20
```

```
## 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_SGP' 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:                57.816 seconds (Sampling)
## Chain 1:                57.816 seconds (Total)
## Chain 1:
```

```r
#Computing Mean
y_predict_values_cont_bs <- extract(pred_gp_cont_bs,permuted=FALSE)
y_mean_values_cont_bs <- c(colMeans(y_predict_values_cont_bs))
y_mean_values_cont_bs <- y_mean_values_cont_bs[1:(length(y_mean_values_cont_bs)-1)]

#Computing Standard Deviation
pred_gp_summary_cont_bs <- summary(pred_gp_cont_bs, sd=c("sd"))$summary
pred_gp_sd_cont_bs <- pred_gp_summary_cont_bs[, c("sd")]
y_sd_values_cont_bs <- pred_gp_sd_cont_bs[1:(length(pred_gp_sd_cont_bs)-1)]

#Discrepancy
Discrepancy_function <- y_mean_values_cont_bs - blackscholes_test_cont

par(mfrow = c(1, 2))
#Contour for Predictions aka mean values of predicitons
contour(dim1, dim2, matrix(Discrepancy_function, length(dim1), length(dim2)), labcex =1.5, main = "Disc
points(x2_cont[,1], x2_cont[,2], pch = 19, cex = 0.5, col = "red")
```

pancy Modeling in 2D (Strike Price



##Part 8: Other Machine Learning Methods

8-1: Fitting ANN

```
library(rstan)
source("gp.utility.R")
library(neuralnet)
```

```
##
## Attaching package: 'neuralnet'

## The following object is masked from 'package:dplyr':
##
##     compute
```

```
# Fitting ANN model
stan_dat <- read_rdump('Financial_Data_Put_American.R')
```

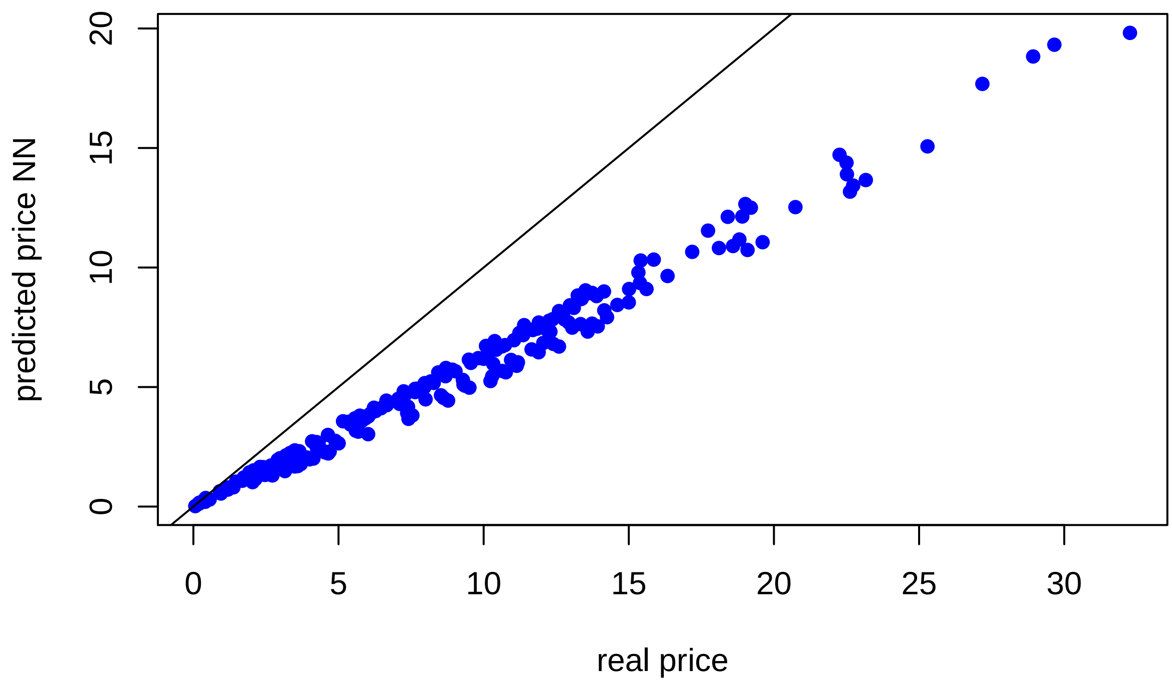
```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   date = col_character(),
##   symbol = col_character(),
##   exdate = col_character(),
##   cp_flag = col_character(),
```

```
##   ticker = col_character(),
##   exercise_style = col_character()
## )

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

## Warning: 98350 parsing failures.
##    row        col expected actual
## 142894 6/21/2019  a double  FALSE 'C:/Users/CK/Desktop/CK/Duke/Honors Thesis/Github/Independent_Stud
## 142894 9/20/2019  a double  FALSE 'C:/Users/CK/Desktop/CK/Duke/Honors Thesis/Github/Independent_Stud
## 142894 12/20/2019 a double  FALSE 'C:/Users/CK/Desktop/CK/Duke/Honors Thesis/Github/Independent_Stud
## 142895 6/21/2019  a double  FALSE 'C:/Users/CK/Desktop/CK/Duke/Honors Thesis/Github/Independent_Stud
## 142895 9/20/2019  a double  FALSE 'C:/Users/CK/Desktop/CK/Duke/Honors Thesis/Github/Independent_Stud
## ...... .......... ........ ...... ..................................................................
## See problems(...) for more details.
```

```
train <- data.frame(stan_dat$train)
n <- names(data.frame(stan_dat$train))
nn <- neuralnet(y_scaled ~ x_1 + x_2 + x_3 + x_4 + x_5 + x_6, data=train, hidden=c(2,1),linear.output=T
plot(nn)
```

8-2: Testing ANN

```
#Testing
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 <- 609 #06/10 Calls
# test_end <- 999 #06/14 Calls

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

x_1 <- as.numeric(stan_dat$total_puts_American$forward_price_scaled[test_start:test_end])
x_2 <- as.numeric(stan_dat$total_puts_American$strike_price_scaled[test_start:test_end])
x_3 <- as.numeric(stan_dat$total_puts_American$impl_volatility_scaled[test_start:test_end])
x_4 <- as.numeric(stan_dat$total_puts_American$time_to_exp_scaled[test_start:test_end])
x_5 <- as.numeric(stan_dat$total_puts_American$dividend_yield_scaled[test_start:test_end])
x_6 <- as.numeric(stan_dat$total_puts_American$interest_rate_scaled[test_start:test_end])
y_scaled <- as.numeric(stan_dat$total_puts_American$mid_price_scaled[test_start:test_end])
test <- data.frame(cbind(x_1,x_2,x_3,x_4,x_5,x_6,y_scaled))


predict_testNN <- neuralnet::compute(nn, test[,c(1:6)])
predict_testNN <- (predict_testNN$net.result * (max(stan_dat$total_puts_American$mid_price[test_start:t
plot(stan_dat$total_puts_American$mid_price[test_start:test_end], predict_testNN, col='blue', pch=16, y
abline(0,1)
```




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

```
## [1] 16.43276
```

8-3 Fitting & Testing Bagging

```
#Bagging
library(tree)
```

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

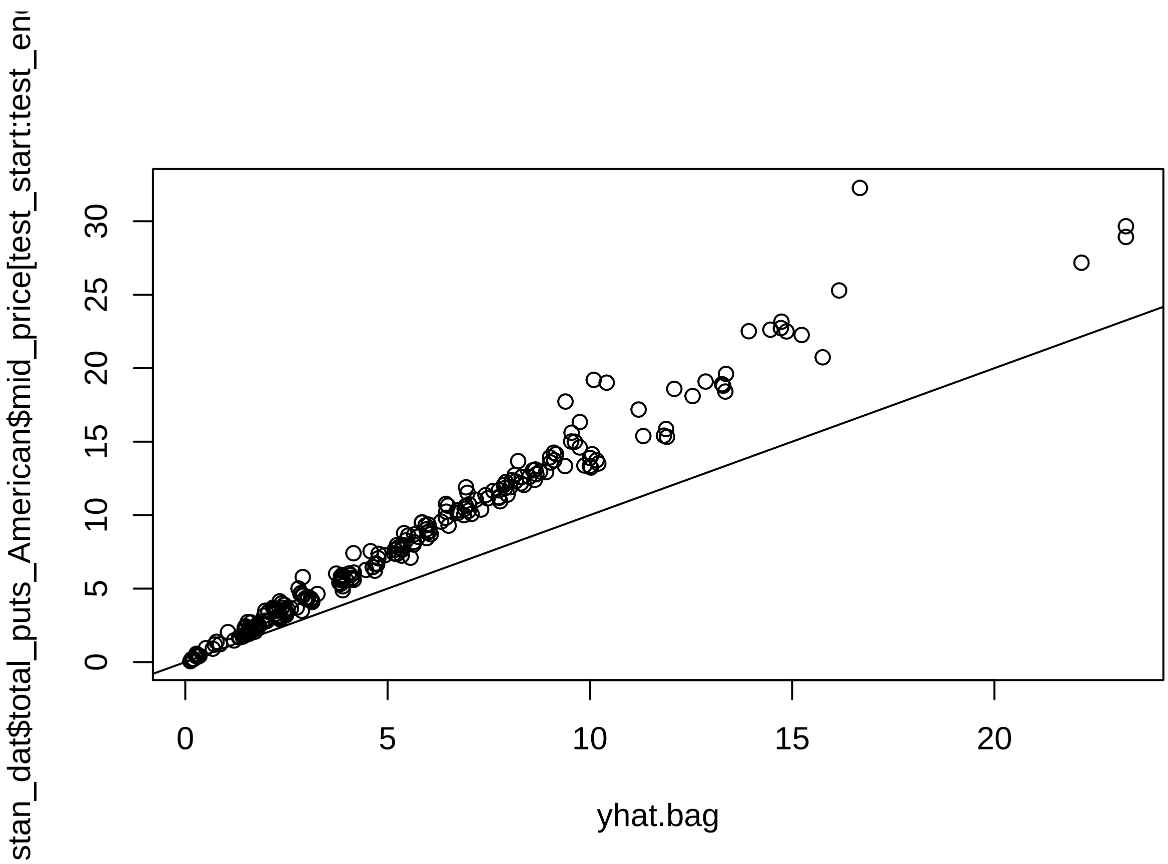
```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine

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

```
bagging <- randomForest(y_scaled~., data = train, mtry = 6, importance = TRUE)
bagging
```

```
##
## Call:
##  randomForest(formula = y_scaled ~ ., data = train, mtry = 6,      importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##           Mean of squared residuals: 0.0006930837
##                     % Var explained: 96.79
```

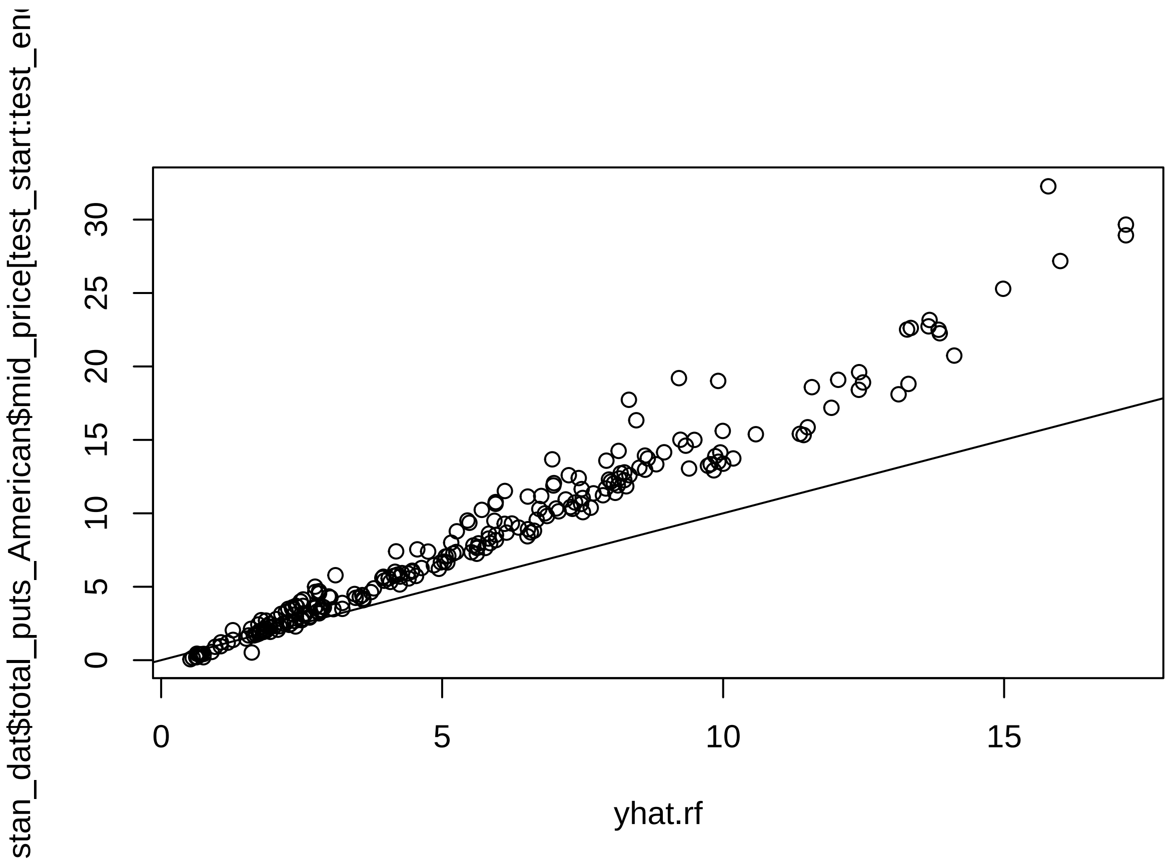
```
yhat.bag <- predict(bagging, newdata = test)
yhat.bag <- (yhat.bag * (max(stan_dat$total_puts_American$mid_price[test_start:test_end]) - min(stan_da
plot(yhat.bag, stan_dat$total_puts_American$mid_price[test_start:test_end])
abline(0,1)
```



```r
mean((yhat.bag - stan_dat$total_puts_American$mid_price[test_start:test_end])^2)
```

```
## [1] 12.04777
```

```r
#Random Forest
rf <- randomForest(y_scaled~., data = train, mtry = 2, importance = TRUE)
yhat.rf <- predict(rf, newdata = test)
yhat.rf <- (yhat.rf * (max(stan_dat$total_puts_American$mid_price[test_start:test_end]) - min(stan_dat$
plot(yhat.rf, stan_dat$total_puts_American$mid_price[test_start:test_end])
abline(0,1)
```



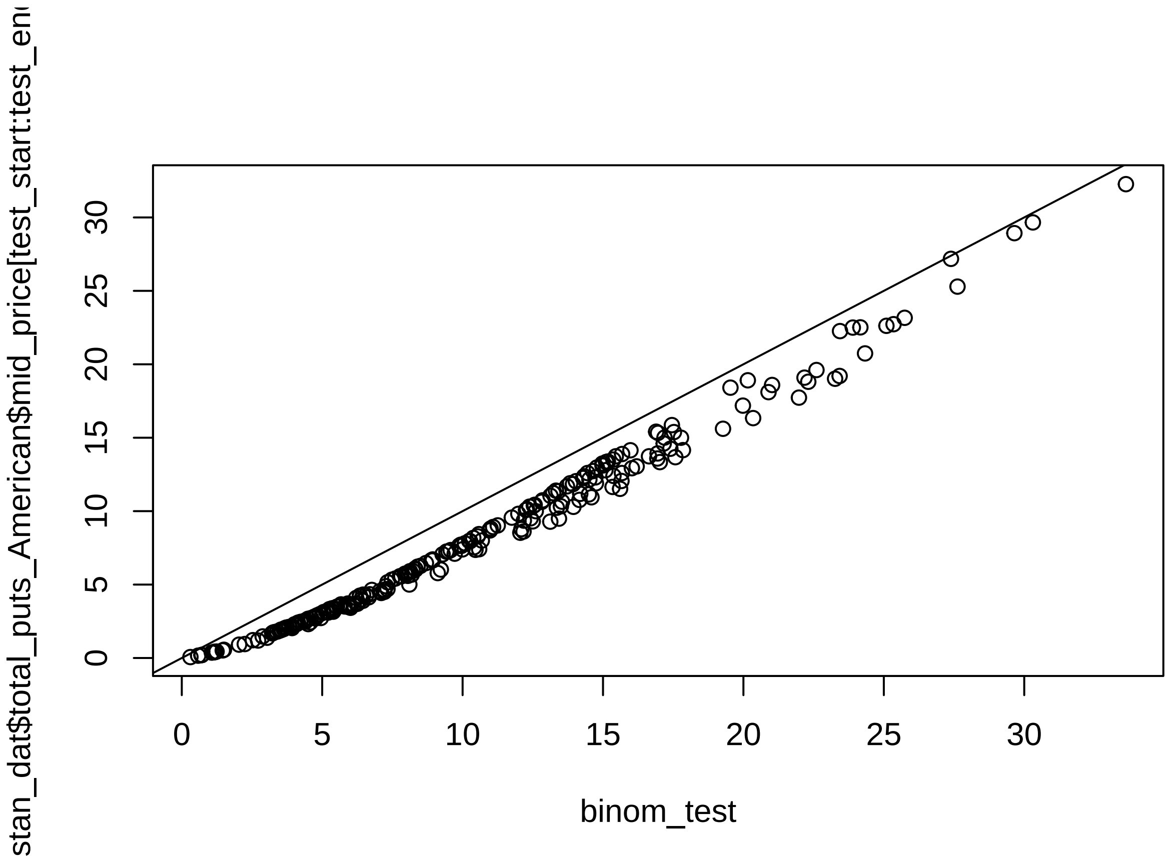

```r
mean((yhat.rf - stan_dat$total_puts_American$mid_price[test_start:test_end])^2)
```

```
## [1] 14.75094
```

8-4 Fitting & Testing Binomial Option Trees

```r
library(derivmkts)

x2_bs <- cbind(as.numeric(stan_dat$total_puts_American$forward_price[test_start:test_end]),as.numeric(st

binom_test <- rep(NA,length(x2_bs[,1]))
for (row in 1:nrow(data.frame(x2_bs))){
```

```r
  binom_test[row] <- binomopt(s=as.numeric(stan_dat$total_puts_American$forward_price[test_start:test_en
}

plot(binom_test, stan_dat$total_puts_American$mid_price[test_start:test_end])
abline(0,1)
```




```r
mean((binom_test - stan_dat$total_puts_American$mid_price[test_start:test_end])^2)
```

```
## [1] 5.549929
```

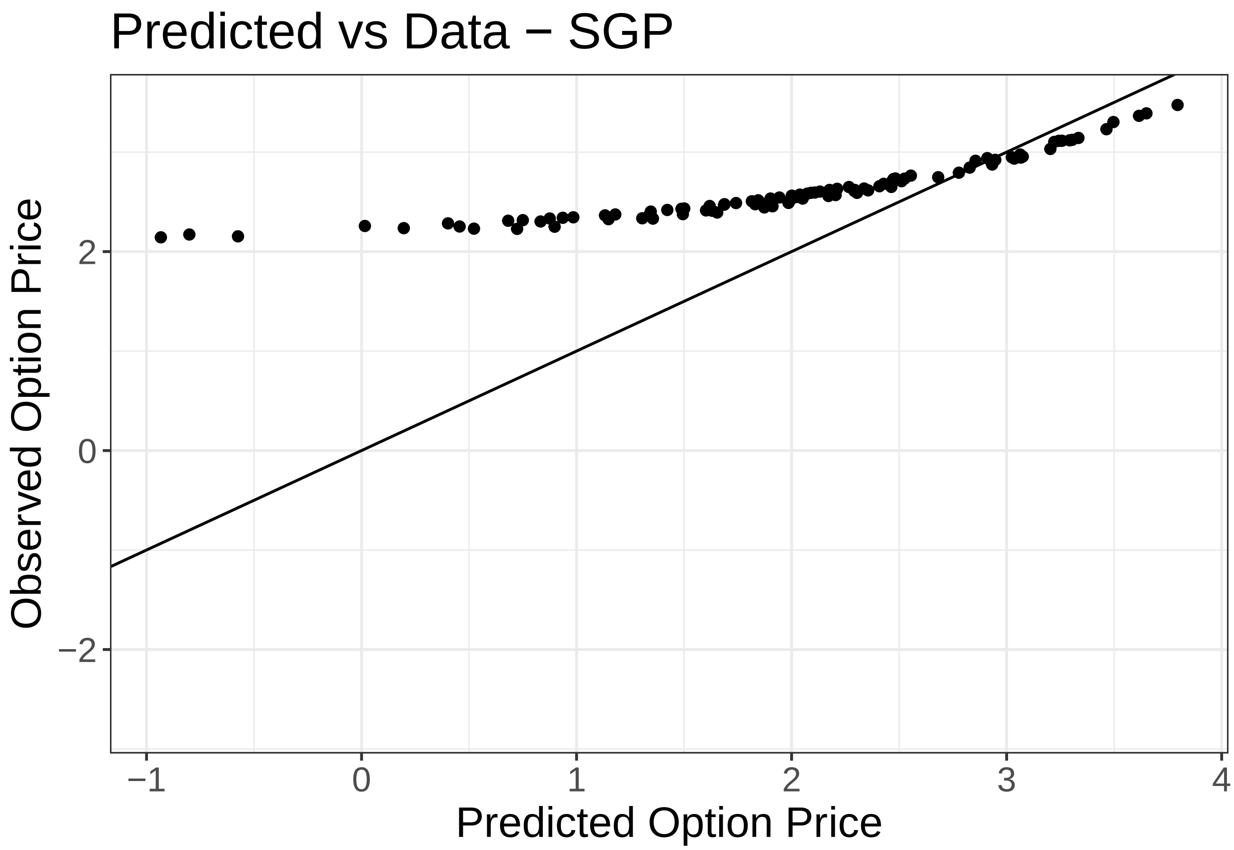
8-5 Summary of Other Machine Learning Methods

```r
par(mfrow=c(2,3))
#Plotting Standard GP
ggplot(mapping=aes(x=log(y_mean_values_SGP),y=log(stan_dat$total_puts_American$mid_price[test_start:tes
  geom_point() +
  labs(title = "Predicted vs Data - SGP",
       x = "Predicted Option Price",
       y = "Observed Option Price") +
  theme_bw() +
  theme(text=element_text(size=16)) +
  geom_abline(intercept = 0, slope = 1)
```

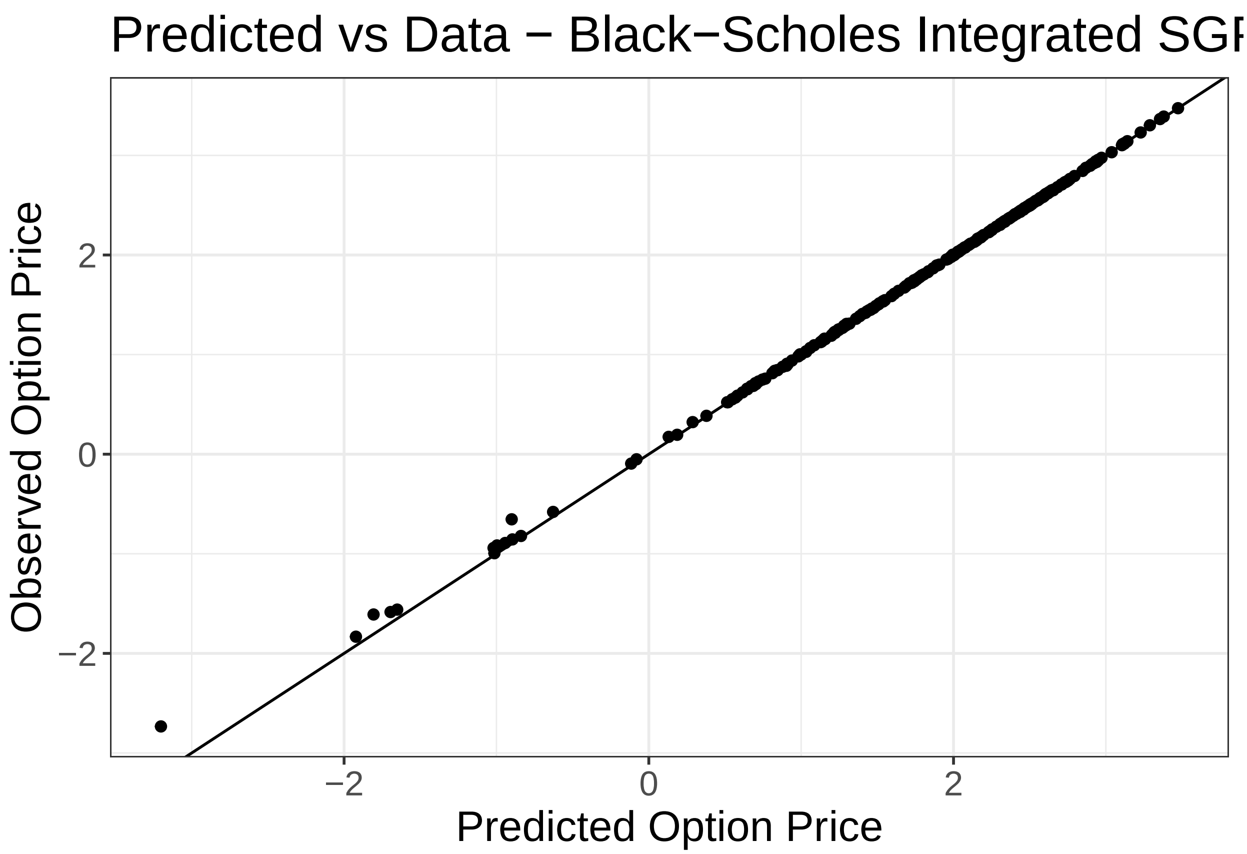
```
## Warning in log(y_mean_values_SGP): NaNs produced
```

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

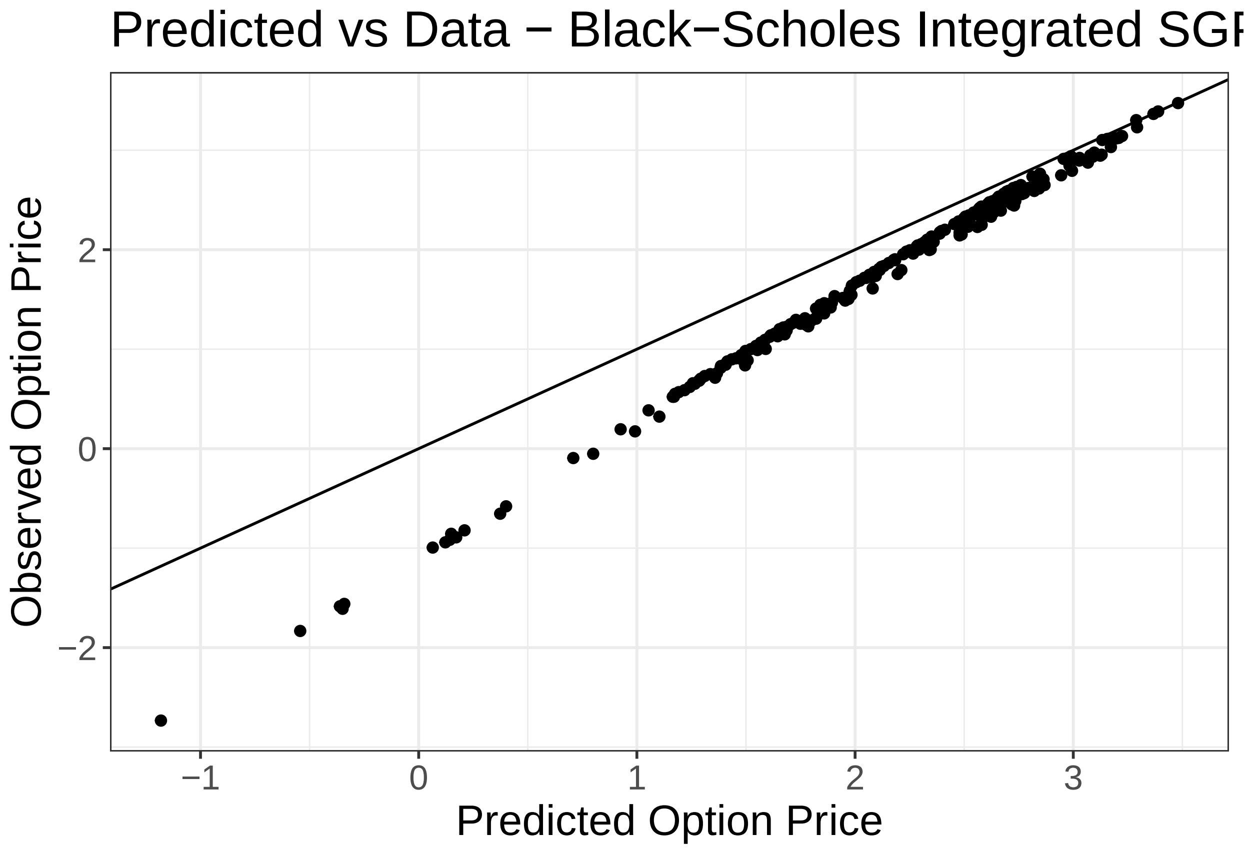
```
## Warning: Removed 140 rows containing missing values (geom_point).
```



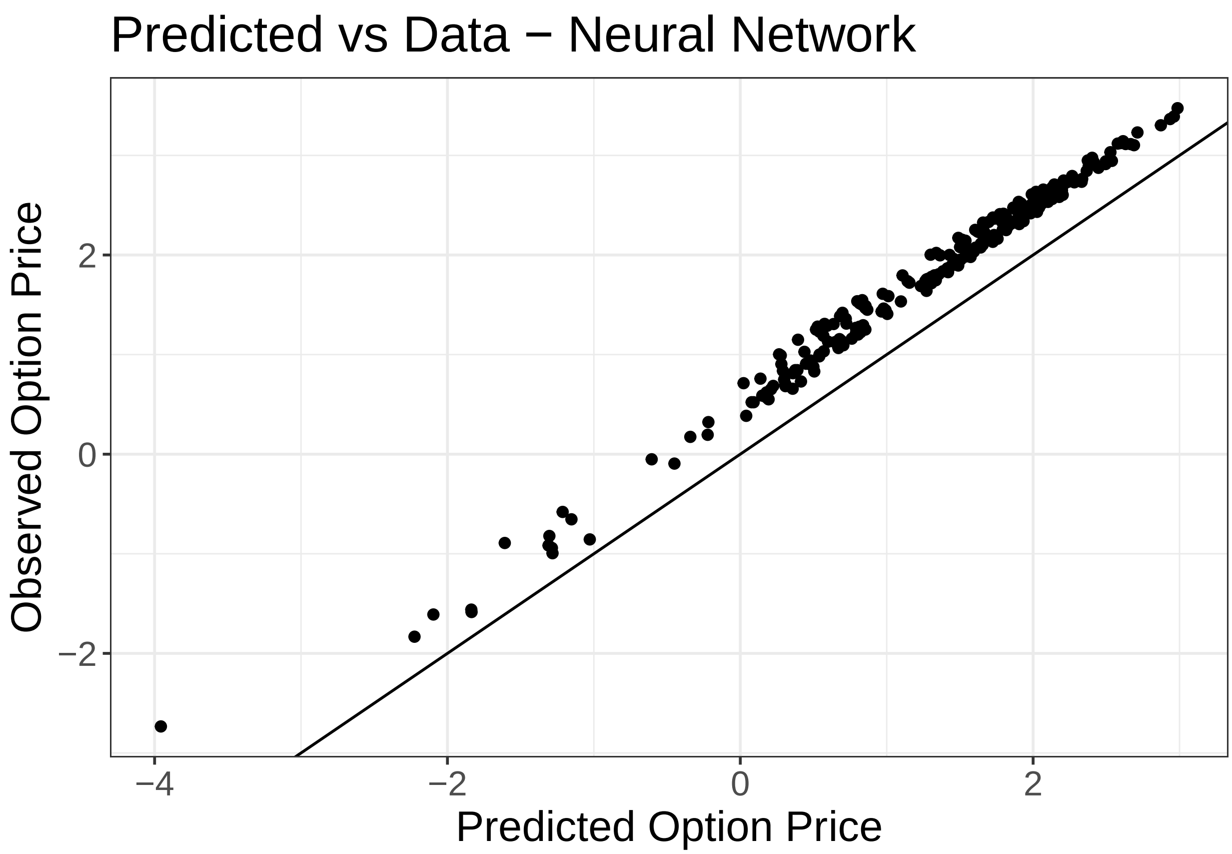

```
#Plotting bs
ggplot(mapping=aes(x=log(y_mean_values_bs),y=log(stan_dat$total_puts_American$mid_price[test_start:test_
  geom_point() +
  labs(title = "Predicted vs Data - Black-Scholes Integrated SGP",
       x = "Predicted Option Price",
       y = "Observed Option Price") +
  theme_bw() +
  theme(text=element_text(size=16))+
  geom_abline(intercept = 0, slope = 1)
```



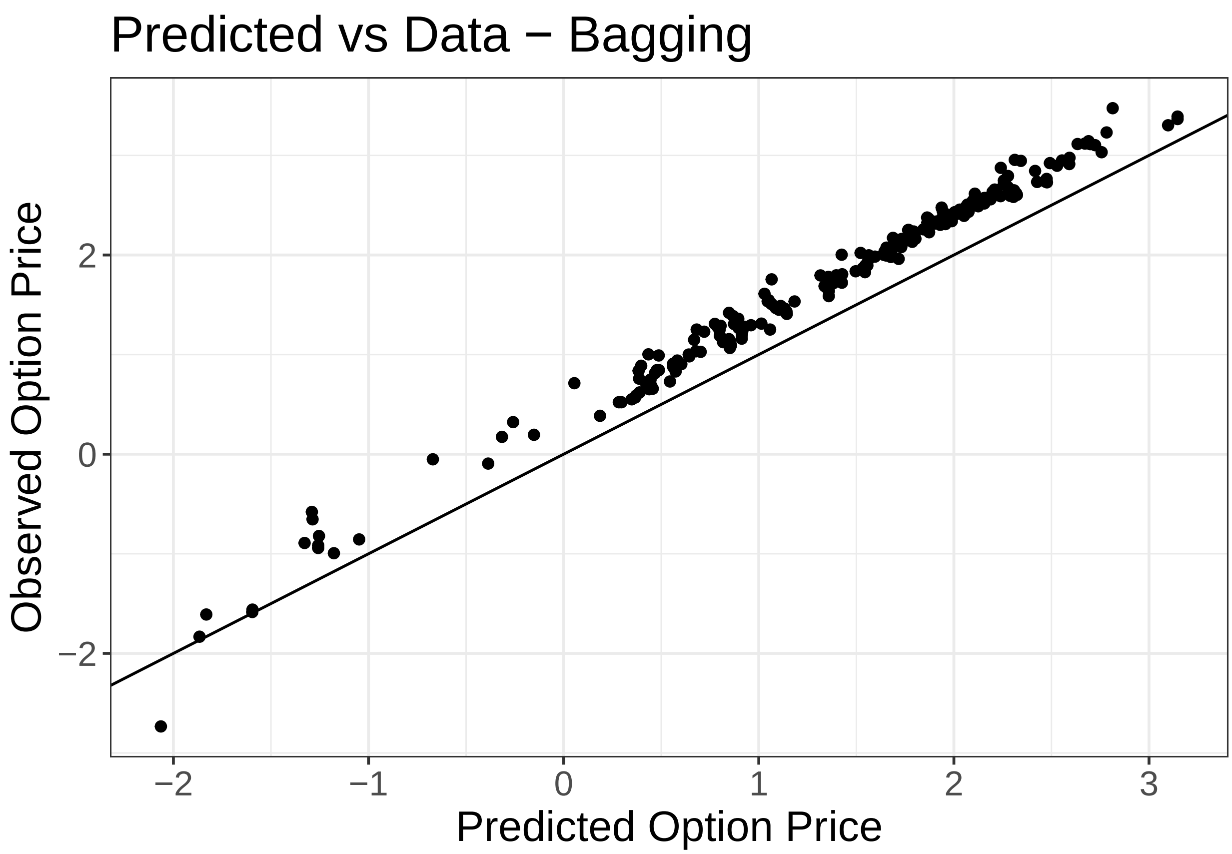

```
#Plotting Blackscholes
ggplot(mapping=aes(x=log(blackscholes_test),y=log(stan_dat$total_puts_American$mid_price[test_start:tes
  geom_point() +
  labs(title = "Predicted vs Data - Black-Scholes Integrated SGP",
       x = "Predicted Option Price",
       y = "Observed Option Price") +
  theme_bw() +
  theme(text=element_text(size=16))+
  geom_abline(intercept = 0, slope = 1)
```



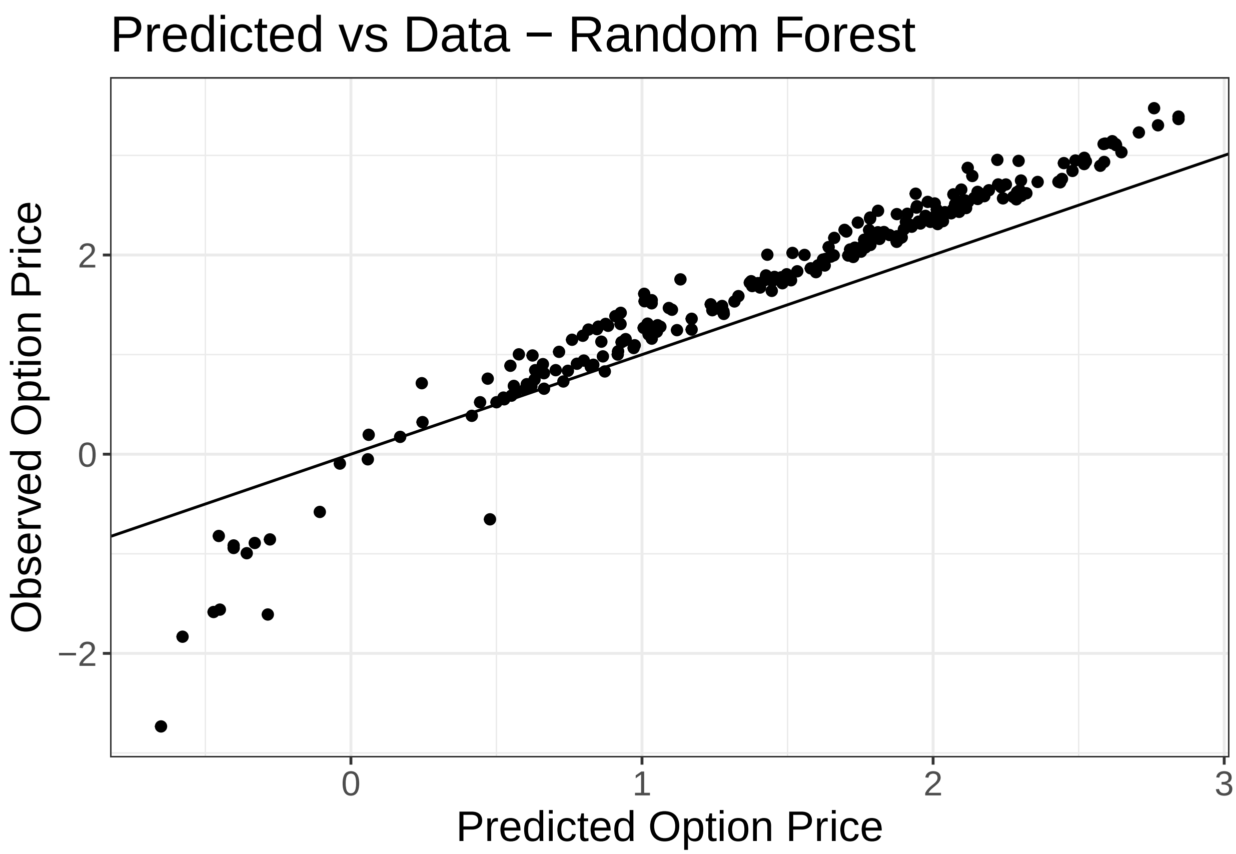

```r
#Plotting Artificial Neural Networks
ggplot(mapping=aes(x=log(predict_testNN),y=log(stan_dat$total_puts_American$mid_price[test_start:test_er
  geom_point() +
  labs(title = "Predicted vs Data - Neural Network",
       x = "Predicted Option Price",
       y = "Observed Option Price") +
  theme_bw() +
  theme(text=element_text(size=16)) +
  geom_abline(intercept = 0, slope = 1)
```



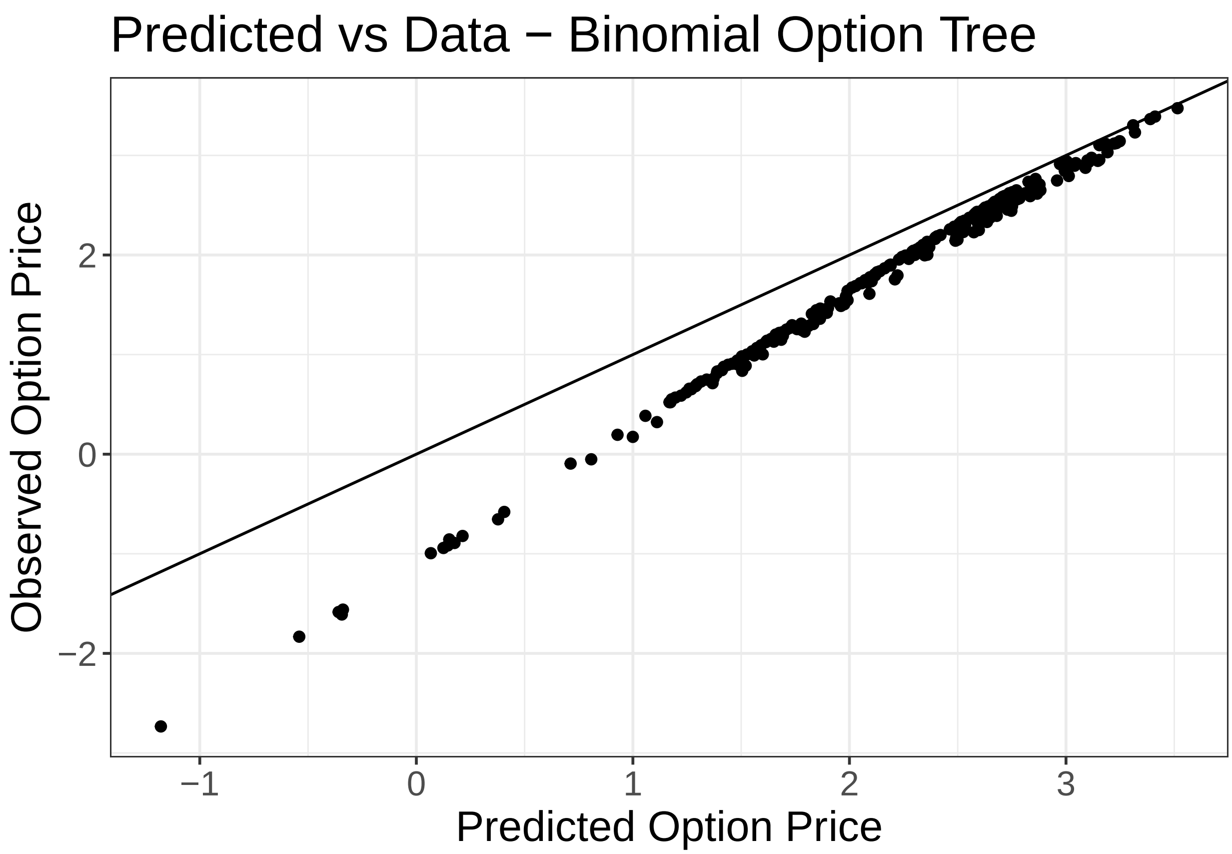

```
#Plotting Bagging
ggplot(mapping=aes(x=log(yhat.bag),y=log(stan_dat$total_puts_American$mid_price[test_start:test_end]))) +
  geom_point() +
  labs(title = "Predicted vs Data - Bagging",
       x = "Predicted Option Price",
       y = "Observed Option Price") +
  theme_bw() +
  theme(text=element_text(size=16)) +
  geom_abline(intercept = 0, slope = 1)
```




```r
#Plotting Random Forest
ggplot(mapping=aes(x=log(yhat.rf),y=log(stan_dat$total_puts_American$mid_price[test_start:test_end]))) +
  geom_point() +
  labs(title = "Predicted vs Data - Random Forest",
       x = "Predicted Option Price",
       y = "Observed Option Price") +
  theme_bw() +
  theme(text=element_text(size=16)) +
  geom_abline(intercept = 0, slope = 1)
```




```r
#Plotting Binomial Option Tree
ggplot(mapping=aes(x=log(binom_test),y=log(stan_dat$total_puts_American$mid_price[test_start:test_end])
  geom_point() +
  labs(title = "Predicted vs Data - Binomial Option Tree",
       x = "Predicted Option Price",
       y = "Observed Option Price") +
  theme_bw() +
  theme(text=element_text(size=16)) +
  geom_abline(intercept = 0, slope = 1)
```




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

```
## [1] 136.0237
```

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

```
## [1] 0.002538695
```

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

```
## [1] 4.923348
```

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

```
## [1] 16.43276
```

```
MSE(yhat.bag,stan_dat$total_puts_American$mid_price[test_start:test_end])
```

```
## [1] 12.04777
```

```
MSE(yhat.rf,stan_dat$total_puts_American$mid_price[test_start:test_end])
```

```
## [1] 14.75094
```

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

```
## [1] 5.549929
```