B\_FP\_R\_code

### DATA ###  
setwd('/Users/AM/Documents/\_CU Masters/2020 fall Bayesian\_7393/Final\_Project/data')  
  
SP\_500 = fread("^GSPC.csv")  
  
VIX\_9d\_CBOE = fread("vix9ddailyprices.csv")  
VIX\_30d\_CBOE = fread("vixcurrent.csv")  
VIX\_3m\_CBOE = fread("vix3mdailyprices.csv")  
VIX\_6m\_CBOE = fread("vix6mdailyprices.csv")  
  
realized = fread("oxfordmanrealizedvolatilityindices 2.csv")  
  
SP\_500 = SP\_500[,c("Date", "Close", "Volume")]  
colnames(SP\_500) = c("date", "close", "volume")  
SP\_500$date = as.Date(SP\_500$date)  
#remove(SP\_500)  
  
  
vix\_9d = VIX\_9d\_CBOE[-c(1,2,3,4), c("V1", "V5")]  
colnames(vix\_9d) = c("date", "vix\_9d")   
vix\_9d$date = as.Date(mdy(vix\_9d$date))  
vix\_9d$vix\_9d = as.numeric(vix\_9d$vix\_9d)  
remove(VIX\_9d\_CBOE)  
  
vix\_30d = VIX\_30d\_CBOE[-c(1,2), c("V1", "V5")]  
colnames(vix\_30d) = c("date", "vix\_30d")   
vix\_30d$date = as.Date(mdy(vix\_30d$date))  
vix\_30d$vix\_30d = as.numeric(vix\_30d$vix\_30d)  
remove(VIX\_30d\_CBOE)  
  
vix\_3m = VIX\_3m\_CBOE[-c(1,2,3), c("V1", "V5")]  
colnames(vix\_3m) = c("date", "vix\_3m")   
vix\_3m$date = as.Date(mdy(vix\_3m$date))  
vix\_3m$vix\_3m = as.numeric(vix\_3m$vix\_3m)  
remove(VIX\_3m\_CBOE)  
  
vix\_6m = VIX\_6m\_CBOE[-c(1,2,3), c("V1", "V5")]  
colnames(vix\_6m) = c("date", "vix\_6m")   
vix\_6m$date = as.Date(mdy(vix\_6m$date))  
vix\_6m$vix\_6m = as.numeric(vix\_6m$vix\_6m)  
remove(VIX\_6m\_CBOE)  
  
  
temp = left\_join(vix\_9d, vix\_30d, by = "date")  
temp = left\_join(temp, vix\_3m, by = "date")  
temp = left\_join(temp, vix\_6m, by = "date")  
vix\_df = temp  
rm("vix\_30d", "vix\_3m", "vix\_6m", "vix\_9d", "temp")  
  
realized$date = as\_datetime(realized$V1)  
realized = realized[,-c("V1")]  
realized$date=realized$date+hours(2) #needed it as RV$V1 has 23:00 because of the daylight time (which wrongly shifts date to the previous one)   
realized$date = as.Date(realized$date)  
  
temp = realized[realized$Symbol==".SPX",c("date", "close\_price")]  
  
#unique(realized$Symbol)  
rv = realized[realized$Symbol==".SPX",]

## lin return ##  
lin\_ret = function (time\_ser) { #the same as Delt(x, type = 'log') from Quantmod lib  
 l\_r = ROC(time\_ser, type = "discrete")  
 return(l\_r)  
}  
# function check  
lin\_ret(c(10,15,18,27))

## [1] NA 0.5 0.2 0.5

## log return ##  
  
log\_ret = function (time\_ser) { #the same as Delt(x, type = 'log') from Quantmod lib  
 log\_time\_ser = log(time\_ser)  
 l\_ret = momentum(log\_time\_ser)  
 return(l\_ret)  
}  
# function check  
log\_ret(c(exp(1), exp(3), exp(4), exp(2), exp(5)))

## [1] NA 2 1 -2 3

log\_ret(c(1,1.01,1.01^2))

## [1] NA 0.009950331 0.009950331

Delt(c(exp(1), exp(3), exp(4), exp(2), exp(5)), type = 'log')

## Delt.1.log  
## [1,] NA  
## [2,] 2  
## [3,] 1  
## [4,] -2  
## [5,] 3

Delt(c(1,1.01,1.01^2))

## Delt.1.arithmetic  
## [1,] NA  
## [2,] 0.01  
## [3,] 0.01

build\_vix9\_rv\_subset = function(start\_date, end\_date) { ### end\_date including 1 extra month for T\_out ###  
 rv\_vix = left\_join(rv[,c("date", "rv5")], vix\_df[,c("date", "vix\_9d")], by = "date")  
 rv\_vix\_subset = rv\_vix[((rv\_vix$date>=(as.Date(start\_date) - days(1))) & (rv\_vix$date<=end\_date)),]  
 rv\_vix\_subset$vix\_log\_ret = log\_ret(rv\_vix\_subset$vix\_9d)  
 rv\_vix\_subset$vix\_lin\_ret = lin\_ret(rv\_vix\_subset$vix\_9d)  
 return(rv\_vix\_subset[-c(1),])  
}

quality\_of\_fit = function(fit, T\_out, end\_date, subset, subset\_long, Rbsq\_presence = FALSE) {  
 qf\_df = data.frame(matrix(vector(), T\_out, 6,   
 dimnames = list(c(),   
 c("date" ,"r\_out", "r\_truth", "price\_r\_out",   
 "price\_r\_truth", "price\_truth"))),   
 stringsAsFactors=F)  
 for (i in 1:T\_out) {  
 temp\_name = gsub(" ", "", paste("r\_out[", as.character(i),"]", ""))  
 r\_out=extract(fit, pars=temp\_name)[[1]]  
 qf\_df$r\_out[i] = mean(r\_out)  
   
 }  
 d = max(which(subset\_long$date <= end\_date))  
 qf\_df$date = subset\_long$date[((d+1):(d+T\_out))]  
 qf\_df$r\_truth = subset\_long$vix\_lin\_ret[((d+1):(d+T\_out))]  
 qf\_df$price\_truth = subset\_long$vix\_9d[((d+1):(d+T\_out))]  
   
 qf\_df$price\_r\_truth[1] = (1 + qf\_df$r\_truth[1]) \* subset\_long$vix\_9d[d]  
 for (i in 2:T\_out) {  
 qf\_df$price\_r\_truth[i] = (1 + qf\_df$r\_truth[i]) \* qf\_df$price\_r\_truth[i-1]  
 }  
   
 qf\_df$price\_r\_out[1] = (1 + qf\_df$r\_out[1]) \* subset\_long$vix\_9d[d]  
 for (i in 2:T\_out) {  
 qf\_df$price\_r\_out[i] = (1 + qf\_df$r\_out[i]) \* qf\_df$price\_r\_out[i-1]  
 }  
   
 MAE\_ret = sum(abs(qf\_df$r\_truth - qf\_df$r\_out)) / T\_out  
 MAPE\_ret = sum((100\*abs(qf\_df$r\_truth - qf\_df$r\_out)/ qf\_df$r\_truth)) / T\_out  
 MSE\_ret = sum((qf\_df$r\_truth - qf\_df$r\_out)^2) / T\_out  
   
 MAE\_ind = sum(abs(qf\_df$price\_truth - qf\_df$price\_r\_out)) / T\_out  
 MAPE\_ind = sum((100\*abs(qf\_df$price\_truth - qf\_df$price\_r\_out)/ qf\_df$price\_truth)) / T\_out  
 MSE\_ind = sum((qf\_df$price\_truth - qf\_df$price\_r\_out)^2) / T\_out  
 ### to add MSE for actual and forecasted prices  
   
 # WAIC and LOOC and Rbsq  
 log\_lik = extract\_log\_lik(fit, merge\_chains = FALSE)  
 r\_eff = exp(relative\_eff(log\_lik))  
 if (Rbsq\_presence == TRUE) {  
 RBSQ=extract(fit, pars="Rbsq")[[1]]  
 Rbsq = mean(RBSQ)  
 } else {Rbsq = NA}  
   
 qf = list(qf\_df, summary(fit), waic(log\_lik), loo(log\_lik, r\_eff = r\_eff), Rbsq,  
 MAE\_ret, MAPE\_ret, MSE\_ret, MAE\_ind, MAPE\_ind, MSE\_ind) # add MSE  
 (names(qf) = c("returns", "summary", "waic", "looic", "Rbsq",  
 "MAE\_ret", "MAPE\_ret", "MSE\_ret", "MAE\_ind", "MAPE\_ind", "MSE\_ind")) #add MSE  
 return(qf)  
  
}

forecast\_vix = function (model\_code, start\_date, subset\_duration,   
 step\_size, n\_steps, T\_out,   
 sigma\_1, mu\_1, iter, chains,   
 Rbsq\_presence, max\_treedepth) {  
 qf\_list <- list()  
 for (i in (1:n\_steps)) {  
 t\_0 = as.Date(start\_date) + months((i-1) \* step\_size)  
 t\_1 = as.Date(t\_0) + months(subset\_duration + 1) #interim end\_date, for subset\_long  
 subset\_long = build\_vix9\_rv\_subset(t\_0, t\_1)  
 t\_1 = as.Date(t\_0) + months(subset\_duration) #stored end\_date, for subset  
 subset = build\_vix9\_rv\_subset(t\_0, t\_1)  
  
 T <- length(subset$vix\_lin\_ret)  
 r <- subset$vix\_lin\_ret  
 rv <- subset$rv5  
 v = var(r)  
 file\_name = paste(model\_name, as.character(Sys.Date()),   
 "for", as.character(t\_0), as.character(t\_1), ".rda", sep=" ")  
   
 stan\_dat = list(T = T, rv = rv, y = r, mu\_1 = mu\_1, sigma\_1 = sigma\_1, T\_out = T\_out, v = v)  
 r\_fit = stan(model\_code = model\_code, data = stan\_dat,   
 iter = iter, chains = chains,  
 control = list(max\_treedepth = max\_treedepth))  
   
 setwd('/Users/AM/Documents/\_CU Masters/2020 fall Bayesian\_7393/Final\_Project/output')  
 #temp\_1 = readRDS("AR1 2020-11-23 for 2017-01-22 2017-07-22 .rda")  
 #saveRDS(r\_fit, file = file\_name, compress = "xz") 1 # commented to print out the code  
   
 qf = quality\_of\_fit(fit = r\_fit, T\_out = T\_out, end\_date = t\_1,   
 subset = subset, subset\_long = subset\_long, Rbsq\_presence = Rbsq\_presence)  
 qf\_list[[i]] <- qf  
 names(qf\_list)[i] = as.character(t\_1) #end of sample and start of the forecast  
 }  
 return(qf\_list)  
}   
#sso <- launch\_shinystan(r\_fit)

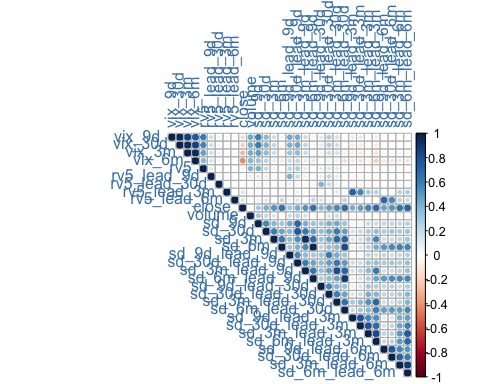
rv\_vix = left\_join(rv[,c("date", "rv5")], vix\_df[,c("date", "vix\_9d")], by = "date")  
rv\_vix\_subset = rv\_vix[((rv\_vix$date>"2015-01-01") & (rv\_vix$date<"2016-06-30")),]  
  
rv\_vix\_subset$vix\_log\_ret = log\_ret(rv\_vix\_subset$vix\_9d)  
rv\_vix\_subset$vix\_lin\_ret = lin\_ret(rv\_vix\_subset$vix\_9d)  
  
  
coeff = max(rv\_vix\_subset$vix\_9d, na.rm = TRUE)/ max(rv\_vix\_subset$rv5, na.rm = TRUE)  
ggplot(rv\_vix\_subset, aes(date)) +   
 geom\_line(aes(y = rv5, colour = "rv5")) +   
 geom\_line(aes(y = vix\_9d/ coeff, colour = "vix")) +  
 scale\_y\_continuous(  
 # Features of the first axis  
 name = "Realized variance (5-min)",  
 # Add a second axis and specify its features  
 sec.axis = sec\_axis(~.\*coeff, name="vix (same day)")) +   
 theme\_ipsum() +  
 theme(  
 axis.title.y = element\_text(color = "red", size=12),  
 axis.title.y.right = element\_text(color = "cyan3", size=12)) +  
 ggtitle("some inference")

SP\_500\_corr\_df = inner\_join(vix\_df, rv[,c("date","rv5")], by = "date")  
#colnames(SP\_500\_corr\_df)[6] = "rv5"  
# realized volatility rv5 today (i.e. as of (lead\_0d)=0 days in the future from "date")  
SP\_500\_corr\_df$rv5\_lead\_9d = lead(SP\_500\_corr\_df$rv5, 9)  
SP\_500\_corr\_df$rv5\_lead\_30d = lead(SP\_500\_corr\_df$rv5, 30)  
SP\_500\_corr\_df$rv5\_lead\_3m = lead(SP\_500\_corr\_df$rv5, 91)  
SP\_500\_corr\_df$rv5\_lead\_6m = lead(SP\_500\_corr\_df$rv5, 182)  
  
SP\_500\_corr\_df = left\_join(SP\_500\_corr\_df, SP\_500, by = "date")  
SP\_500\_corr\_df$volume = SP\_500\_corr\_df$volume / 10^9  
# historical volatility for the last (sd\_nd) days as of "date"   
SP\_500\_corr\_df$sd\_9d = runSD(SP\_500\_corr\_df$close, 9)   
SP\_500\_corr\_df$sd\_30d = runSD(SP\_500\_corr\_df$close, 30)  
SP\_500\_corr\_df$sd\_3m = runSD(SP\_500\_corr\_df$close, 91)  
SP\_500\_corr\_df$sd\_6m = runSD(SP\_500\_corr\_df$close, 182)  
  
# create temp columns with non leading na  
SP\_500\_corr\_df$temp\_lead\_9d = lead(SP\_500\_corr\_df$close, 9)   
SP\_500\_corr\_df$temp\_lead\_30d = lead(SP\_500\_corr\_df$close,30)  
SP\_500\_corr\_df$temp\_lead\_3m = lead(SP\_500\_corr\_df$close,91)  
SP\_500\_corr\_df$temp\_lead\_6m = lead(SP\_500\_corr\_df$close,182)  
  
SP\_500\_corr\_df = SP\_500\_corr\_df[SP\_500\_corr\_df$date < "2020-01-01", ]  
  
# historical volatility for the last (sd\_nd) days as of "date"+ lead\_9d (i.e. as of 9 days in the future from "date")   
SP\_500\_corr\_df$sd\_9d\_lead\_9d = runSD(SP\_500\_corr\_df$temp\_lead\_9d, 9)   
SP\_500\_corr\_df$sd\_30d\_lead\_9d = runSD(SP\_500\_corr\_df$temp\_lead\_9d, 30)   
SP\_500\_corr\_df$sd\_3m\_lead\_9d = runSD(SP\_500\_corr\_df$temp\_lead\_9d, 91)   
SP\_500\_corr\_df$sd\_6m\_lead\_9d = runSD(SP\_500\_corr\_df$temp\_lead\_9d, 182)   
  
# historical volatility for the last (sd\_nd) days as of "date"+ lead\_30d (i.e. as of 30 days in the future from "date")   
SP\_500\_corr\_df$sd\_9d\_lead\_30d = runSD(SP\_500\_corr\_df$temp\_lead\_30d, 9)   
SP\_500\_corr\_df$sd\_30d\_lead\_30d = runSD(SP\_500\_corr\_df$temp\_lead\_30d, 30)   
SP\_500\_corr\_df$sd\_3m\_lead\_30d = runSD(SP\_500\_corr\_df$temp\_lead\_30d, 91)   
SP\_500\_corr\_df$sd\_6m\_lead\_30d = runSD(SP\_500\_corr\_df$temp\_lead\_30d, 182)   
  
# historical volatility for the last (sd\_nd) days as of "date"+ lead\_3m (i.e. as of 91 days in the future from "date")   
SP\_500\_corr\_df$sd\_9d\_lead\_3m = runSD(SP\_500\_corr\_df$temp\_lead\_3m, 9)   
SP\_500\_corr\_df$sd\_30d\_lead\_3m = runSD(SP\_500\_corr\_df$temp\_lead\_3m, 30)   
SP\_500\_corr\_df$sd\_3m\_lead\_3m = runSD(SP\_500\_corr\_df$temp\_lead\_3m, 91)   
SP\_500\_corr\_df$sd\_6m\_lead\_3m = runSD(SP\_500\_corr\_df$temp\_lead\_3m, 182)   
  
# historical volatility for the last (sd\_nd) days as of "date"+ lead\_6m (i.e. as of 182 days in the future from "date")   
SP\_500\_corr\_df$sd\_9d\_lead\_6m = runSD(SP\_500\_corr\_df$temp\_lead\_6m, 9)   
SP\_500\_corr\_df$sd\_30d\_lead\_6m = runSD(SP\_500\_corr\_df$temp\_lead\_6m, 30)   
SP\_500\_corr\_df$sd\_3m\_lead\_6m = runSD(SP\_500\_corr\_df$temp\_lead\_6m, 91)   
SP\_500\_corr\_df$sd\_6m\_lead\_6m = runSD(SP\_500\_corr\_df$temp\_lead\_6m, 182)   
  
# remove temp columns   
SP\_500\_corr\_df = SP\_500\_corr\_df[,-c("temp\_lead\_9d", "temp\_lead\_30d", "temp\_lead\_3m", "temp\_lead\_6m")]  
  
sum(is.na(SP\_500\_corr\_df))

## [1] 1540

SP\_500\_corr\_df = SP\_500\_corr\_df[SP\_500\_corr\_df$date > "2012-01-01", ]  
  
  
### take-aways  
# vix does not forecast a point estimate of RV  
# vix has good positive correlation with interval volatility (especially 9 days)  
# there is some correlation vix~volume to be explored  
# there is some correlation vix\_6m~close to be explored  
# let's try to build the model around sd\_9/30d\_lead9/30d ~ vix9/30   
  
### code to double-check runSD  
#temp = SP\_500\_corr\_df$close[1:91]  
#temp1 = 0  
#for (i in (1:length(temp))){temp1 = temp1 + (temp[i]-mean(temp))^2}  
#sqrt(temp1/(length(temp)-1))

M<-cor(SP\_500\_corr\_df[,-c(1)])  
corrplot(M, type = "upper", tl.col = "steel blue")



## - ## pass mu[1] as data  
##---## play with different priors for GARCH(1,1)  
##---## shiny stan - what Yrep and PPcheck do??  
  
stan\_mod\_AR1 = "  
data {  
 int T;  
 vector[T] y;  
 int T\_out;  
 real<lower=0> v; // sample variance of y  
 real mu\_1; // the first mu  
  
}  
  
parameters {  
 real alpha;  
 real beta;  
 //real<lower=-1,upper=1> beta;  
 real<lower=0> sigma;  
}  
transformed parameters {  
 vector[T] mu;  
 for (i in 2:T) {  
 mu[i] = alpha + beta \* y[i-1];  
 }  
 mu[1] = mu\_1;  
}  
model {  
 sigma ~ inv\_gamma(1, 2);  
 alpha ~ normal(0, 1);  
 beta ~ normal(0, 1);  
 for (n in 2:T)  
 y[n] ~ normal(mu[n], sigma);  
}  
generated quantities {  
 vector[T\_out] r\_out;  
 vector[T] log\_lik;  
 //vector[T] lik;  
 real Rbsq; // goodness-of-fit  
 Rbsq = 1 - square(sigma)/v;  
 r\_out[1] = normal\_rng(alpha + beta \* y[T], sigma);  
   
 for (i in 2:T\_out) {  
 r\_out[i] = normal\_rng(alpha + beta \* y[i-1], sigma);  
 }  
 for (i in 1:T) {  
 log\_lik[i] = normal\_lpdf(y[i] | mu[i], sigma);  
 //lik[i] = exp(log\_lik[i]);  
 }  
  
}  
"

stan\_mod\_ARCH1 = "  
data {  
 int<lower=0> T; // number of time points  
 vector[T] y; // return at time t  
 real sigma\_1; // the first sigma  
 int T\_out; // forecast depth, days  
}  
parameters {  
 real mu; // average return  
 real<lower=0> alpha0; // noise intercept  
 real<lower=0,upper=1> alpha1; // noise slope  
   
}  
transformed parameters {  
 vector[T] sigma;  
 for (i in 2:T) {  
 sigma[i] = sqrt(alpha0 + alpha1 \* pow(y[i-1] - mu,2));  
 }  
 sigma[1] = sigma\_1;  
}  
  
model {  
 mu ~ normal(0.01, 0.2);  
 alpha0 ~ normal(0, 1);  
 alpha1 ~ beta(1.1, 1.1);  
 for (i in 2:T)  
 y[i] ~ normal(mu, sigma[i]);  
}  
  
generated quantities {  
 vector[T\_out] r\_out;  
 vector[T\_out] sigma\_out;  
 vector[T] log\_lik;  
   
 sigma\_out[1] = sqrt(alpha0 + alpha1 \* pow(y[T] - mu,2));  
 r\_out[1] = normal\_rng(mu, sigma\_out[1]);  
 for (i in 2:T\_out) {  
 sigma\_out[i] = sqrt(alpha0 + alpha1 \* pow(r\_out[i-1] - mu,2));  
 r\_out[i] = normal\_rng(mu, sigma\_out[i]);  
 }  
   
 for (i in 1:T) {  
 log\_lik[i] = normal\_lpdf(y[i] | mu, sigma[i]);  
 }  
  
}  
  
"

stan\_mod\_GARCH11 = "  
data {  
 int<lower=2> T;   
 vector[T] y;  
 real<lower=0> sigma\_1;  
 int<lower=0> T\_out;  
}  
parameters {  
 real mu;  
 real<lower=0> alpha0;  
 real<lower=0,upper=1> alpha1;  
 real<lower=0,upper=(1-alpha1)> beta1;  
}  
transformed parameters {  
 real<lower=0> sigma[T];  
 sigma[1] = sigma\_1;  
 for (i in 2:T)  
 sigma[i] = sqrt(alpha0  
 + alpha1 \* pow(y[i-1] - mu, 2)  
 + beta1 \* pow(sigma[i-1], 2));  
}  
model {  
 mu ~ normal(0.01, 0.2);  
 alpha0 ~ normal(0, 1);  
 alpha1 ~ beta(1.1, 1.1);  
 beta1 ~ beta(1.1, 1.1);  
 for (i in 2:T)  
 y[i] ~ normal(mu, sigma[i]);  
}  
generated quantities {  
 vector[T\_out] r\_out;  
 vector[T\_out] sigma\_out;  
 vector[T] log\_lik;  
 vector[T] y\_rep;  
   
 sigma\_out[1] = sqrt(alpha0  
 + alpha1 \* pow(y[T] - mu, 2)  
 + beta1 \* pow(sigma[T], 2));  
 r\_out[1] = normal\_rng(mu, sigma\_out[1]);  
 for (i in 2:T\_out) {  
 sigma\_out[i] = sqrt(alpha0  
 + alpha1 \* pow(r\_out[i-1] - mu, 2)  
 + beta1 \* pow(sigma\_out[i-1], 2));  
 r\_out[i] = normal\_rng(mu, sigma\_out[i]);  
 }  
   
 for (i in 1:T) {  
 log\_lik[i] = normal\_lpdf(y[i] | mu, sigma[i]);  
 y\_rep[i] = normal\_rng(mu, sigma[i]);  
 }  
   
}  
"  
  
#sso <- launch\_shinystan(r\_fit)

stan\_mod\_RealGARCH11 = "  
functions {  
 real normal\_lb\_rng(real muu, real siigma, real lb) {  
 real p = normal\_cdf(lb, muu, siigma); // cdf for bounds  
 real u = uniform\_rng(p, 1);  
 return (siigma \* inv\_Phi(u)) + muu; // inverse cdf for value  
 }   
}  
data {  
 int<lower=2> T;   
 vector[T] y; // return  
 vector<lower=0>[T] rv; // it is x in HH2012 (realized volatility estimate)   
 real<lower=0> sigma\_1;  
 int<lower=0> T\_out;  
}  
parameters {  
 real mu;  
 real<lower=0> alpha0; // it is omega in HH2012  
 real<lower=0, upper=1> beta1; // it is the same beta in HH2012  
 real<lower=0,upper=(1-beta1)> gamma;   
 real xi;  
 real phi;  
 real tau\_1;  
 real tau\_2;  
   
 //real<lower=0,upper=1-beta1-gamma> tau\_1;  
 //real<lower=0,upper=1-beta1-gamma-tau\_1> tau\_2;  
   
 real<lower=0> sigma\_rv;  
   
}  
transformed parameters {  
 vector<lower=0>[T] sigma; // it is h in HH2012  
 vector[T] z;  
 vector[T] mu\_rv;  
 sigma[1] = sigma\_1;  
 z[1] = (y[1]-mu)/sigma[1];  
 mu\_rv[1] = xi + phi \* pow(sigma[1], 2) + tau\_1 \* z[1] + tau\_2 \* (pow(z[1], 2) - 1);  
 for (i in 2:T) {  
 sigma[i] = sqrt(alpha0  
 + beta1 \* pow(sigma[i-1], 2)  
 + gamma \* rv[i-1]);  
 z[i] = (y[i]-mu)/sigma[i];  
 mu\_rv[i] = xi + phi \* pow(sigma[i], 2) + tau\_1 \* z[i] + tau\_2 \* (pow(z[i], 2) - 1);  
 }   
}  
model {  
 mu ~ normal(0.0084, 0.01);  
 alpha0 ~ normal(0, 1);  
 beta1 ~ beta(1.01, 1.01);  
 gamma ~ beta(1.01, 1.01);  
 xi ~ normal(0, 1.1);  
 phi ~ normal(0, 1.5);  
 tau\_1 ~ normal(0, 0.5);  
 tau\_2 ~ normal(0, 0.5);  
 sigma\_rv ~ inv\_gamma(10, 1);  
 for (i in 1:T) {  
 rv[i] ~ normal(mu\_rv[i], sigma\_rv);  
 }  
}  
  
generated quantities {  
 vector[T] log\_lik;  
 vector[T] y\_rep;  
  
   
 for (i in 1:T) {  
 log\_lik[i] = normal\_lpdf(y[i] | mu, sigma[i]);  
 y\_rep[i] = normal\_rng(mu, sigma[i]);  
 }  
}  
"  
#sso <- launch\_shinystan(r\_fit)

model\_code = stan\_mod\_AR1  
model\_name = "AR1"  
start\_date = "2017-01-22"  
subset\_duration = 6 # in months, incl 1 extra month for T\_out (validation)  
step\_size = 2 # in months  
n\_steps = 1 # changed from 10 for to print out the code  
T\_out <- 9 # forecast depth, days  
  
iter =10^4  
chains = 1  
max\_treedepth = 10  
  
sigma\_1 <- 0.1   
mu\_1 = 0  
  
f = forecast\_vix(model\_code = model\_code, start\_date = start\_date,   
 subset\_duration = subset\_duration,   
 step\_size = step\_size,n\_steps = n\_steps, T\_out = 9,   
 sigma\_1 = 0.1, mu\_1 = mu\_1, iter = iter, chains = chains,   
 Rbsq\_presence = TRUE, max\_treedepth = max\_treedepth)

## Trying to compile a simple C file

## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c  
## clang -mmacosx-version-min=10.13 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG -I"/Library/Frameworks/R.framework/Versions/4.0/Resources/library/Rcpp/include/" -I"/Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/" -I"/Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/unsupported" -I"/Library/Frameworks/R.framework/Versions/4.0/Resources/library/BH/include" -I"/Library/Frameworks/R.framework/Versions/4.0/Resources/library/StanHeaders/include/src/" -I"/Library/Frameworks/R.framework/Versions/4.0/Resources/library/StanHeaders/include/" -I"/Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppParallel/include/" -I"/Library/Frameworks/R.framework/Versions/4.0/Resources/library/rstan/include" -DEIGEN\_NO\_DEBUG -DBOOST\_DISABLE\_ASSERTS -DBOOST\_PENDING\_INTEGER\_LOG2\_HPP -DSTAN\_THREADS -DBOOST\_NO\_AUTO\_PTR -include '/Library/Frameworks/R.framework/Versions/4.0/Resources/library/StanHeaders/include/stan/math/prim/mat/fun/Eigen.hpp' -D\_REENTRANT -DRCPP\_PARALLEL\_USE\_TBB=1 -I/usr/local/include -fPIC -Wall -g -O2 -c foo.c -o foo.o  
## In file included from <built-in>:1:  
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/StanHeaders/include/stan/math/prim/mat/fun/Eigen.hpp:13:  
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/Dense:1:  
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/Core:88:  
## /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:1: error: unknown type name 'namespace'  
## namespace Eigen {  
## ^  
## /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:16: error: expected ';' after top level declarator  
## namespace Eigen {  
## ^  
## ;  
## In file included from <built-in>:1:  
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/StanHeaders/include/stan/math/prim/mat/fun/Eigen.hpp:13:  
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/Dense:1:  
## /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/Core:96:10: fatal error: 'complex' file not found  
## #include <complex>  
## ^~~~~~~~~  
## 3 errors generated.  
## make: \*\*\* [foo.o] Error 1  
##   
## SAMPLING FOR MODEL '21f2003f56d1fc1a106b303c7a54f66f' NOW (CHAIN 1).  
## Chain 1:   
## Chain 1: Gradient evaluation took 3.5e-05 seconds  
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.35 seconds.  
## Chain 1: Adjust your expectations accordingly!  
## Chain 1:   
## Chain 1:   
## Chain 1: Iteration: 1 / 10000 [ 0%] (Warmup)  
## Chain 1: Iteration: 1000 / 10000 [ 10%] (Warmup)  
## Chain 1: Iteration: 2000 / 10000 [ 20%] (Warmup)  
## Chain 1: Iteration: 3000 / 10000 [ 30%] (Warmup)  
## Chain 1: Iteration: 4000 / 10000 [ 40%] (Warmup)  
## Chain 1: Iteration: 5000 / 10000 [ 50%] (Warmup)  
## Chain 1: Iteration: 5001 / 10000 [ 50%] (Sampling)  
## Chain 1: Iteration: 6000 / 10000 [ 60%] (Sampling)  
## Chain 1: Iteration: 7000 / 10000 [ 70%] (Sampling)  
## Chain 1: Iteration: 8000 / 10000 [ 80%] (Sampling)  
## Chain 1: Iteration: 9000 / 10000 [ 90%] (Sampling)  
## Chain 1: Iteration: 10000 / 10000 [100%] (Sampling)  
## Chain 1:   
## Chain 1: Elapsed Time: 0.20872 seconds (Warm-up)  
## Chain 1: 0.249207 seconds (Sampling)  
## Chain 1: 0.457927 seconds (Total)  
## Chain 1:

## Warning:   
## 1 (0.8%) p\_waic estimates greater than 0.4. We recommend trying loo instead.

## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.

p = data.frame(matrix(vector(), n\_steps + 5, 20,   
 dimnames = list(c(),   
 c("window\_start", "window\_end",  
 "alpha", "alpha\_2.5", "alpha\_97.5",  
 "beta", "beta\_2.5", "beta\_97.5",  
 "sigma", "sigma\_2.5", "sigma\_97.5",  
 "WAIC", "LOOIC", "Rbsq",   
 "MAE\_ret", "MAPE\_ret", "MSE\_ret",   
 "MAE\_ind", "MAPE\_ind", "MSE\_ind"))),  
 stringsAsFactors=F)  
  
for (i in 1:n\_steps){  
 p[i, "window\_start"] = as.character(as.Date(names(f)[i]) - months(subset\_duration))  
 p[i, "window\_end"] = as.character(as.Date(names(f)[i]))  
 p[i, c("alpha", "alpha\_2.5", "alpha\_97.5")] = f[[i]]$summary$summary[,c("mean", "2.5%", "97.5%")][c("alpha"),]  
 p[i, c("beta", "beta\_2.5", "beta\_97.5")] = f[[i]]$summary$summary[,c("mean", "2.5%", "97.5%")][c("beta"),]  
 p[i, c("sigma", "sigma\_2.5", "sigma\_97.5")] = f[[i]]$summary$summary[,c("mean", "2.5%", "97.5%")][c("sigma"),]  
 p[i, c("WAIC")] = f[[i]]$waic$waic  
 p[i, c("LOOIC")] = f[[i]]$looic$looic  
 p[i, c("Rbsq")] = f[[i]]$Rbsq  
 p[i, c("MAE\_ret", "MAPE\_ret", "MSE\_ret", "MAE\_ind", "MAPE\_ind", "MSE\_ind")] = f[[i]][c("MAE\_ret", "MAPE\_ret", "MSE\_ret", "MAE\_ind", "MAPE\_ind", "MSE\_ind")]  
  
}

## Warning: Accessing waic using '$' is deprecated and will be removed in a  
## future release. Please extract the waic estimate from the 'estimates' component  
## instead.

## Warning: Accessing looic using '$' is deprecated and will be removed in a  
## future release. Please extract the looic estimate from the 'estimates' component  
## instead.

p[n\_steps + 1, "window\_end"] = "MEAN"  
p[n\_steps + 2, "window\_end"] = "SE"  
p[n\_steps + 3, "window\_end"] = "SE, %"  
  
standard\_error <- function(x, ...) sqrt(var(x, ...)/length(x))  
options(digits=3)  
for (i in 3:dim(p)[2]) {  
 p[(n\_steps + 1), i] = mean(p[1:n\_steps, i])  
 p[(n\_steps + 2), i] = standard\_error(p[1:n\_steps, i])  
 if (i>=3) {p[(n\_steps + 3), i] = round(100 \* p[(n\_steps + 2), i] / p[(n\_steps + 1), i], digits = 2)}  
}  
p[(n\_steps + 4), 1] = "iter"  
p[(n\_steps + 4), 2] = iter  
p[(n\_steps + 5), 1] = "chais"  
p[(n\_steps + 5), 2] = chains  
  
setwd('/Users/AM/Documents/\_CU Masters/2020 fall Bayesian\_7393/Final\_Project/output')  
write.xlsx(p,   
 paste(model\_name, as.character(Sys.time()), ".xlsx", sep="\_"),  
 sheetName=model\_name)

model\_code = stan\_mod\_ARCH1  
model\_name = "ARCH1"  
start\_date = "2017-01-22"  
subset\_duration = 6 # in months, incl 1 extra month for T\_out (validation)  
step\_size = 2 # in months  
n\_steps = 1 # number of different time windows (to build different models)  
T\_out <- 9 # forecast depth, days  
  
iter = 10^4  
chains = 4  
max\_treedepth = 10  
  
sigma\_1 <- 0.1   
mu\_1 = 0  
  
f = forecast\_vix(model\_code = model\_code, start\_date = start\_date,   
 subset\_duration = subset\_duration,   
 step\_size = step\_size,n\_steps = n\_steps, T\_out = T\_out,   
 sigma\_1 = sigma\_1, mu\_1 = mu\_1, iter = iter, chains = chains,   
 Rbsq\_presence = FALSE, max\_treedepth = max\_treedepth)

## Trying to compile a simple C file

## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c  
## clang -mmacosx-version-min=10.13 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG -I"/Library/Frameworks/R.framework/Versions/4.0/Resources/library/Rcpp/include/" -I"/Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/" -I"/Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/unsupported" -I"/Library/Frameworks/R.framework/Versions/4.0/Resources/library/BH/include" -I"/Library/Frameworks/R.framework/Versions/4.0/Resources/library/StanHeaders/include/src/" -I"/Library/Frameworks/R.framework/Versions/4.0/Resources/library/StanHeaders/include/" -I"/Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppParallel/include/" -I"/Library/Frameworks/R.framework/Versions/4.0/Resources/library/rstan/include" -DEIGEN\_NO\_DEBUG -DBOOST\_DISABLE\_ASSERTS -DBOOST\_PENDING\_INTEGER\_LOG2\_HPP -DSTAN\_THREADS -DBOOST\_NO\_AUTO\_PTR -include '/Library/Frameworks/R.framework/Versions/4.0/Resources/library/StanHeaders/include/stan/math/prim/mat/fun/Eigen.hpp' -D\_REENTRANT -DRCPP\_PARALLEL\_USE\_TBB=1 -I/usr/local/include -fPIC -Wall -g -O2 -c foo.c -o foo.o  
## In file included from <built-in>:1:  
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/StanHeaders/include/stan/math/prim/mat/fun/Eigen.hpp:13:  
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/Dense:1:  
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/Core:88:  
## /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:1: error: unknown type name 'namespace'  
## namespace Eigen {  
## ^  
## /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:16: error: expected ';' after top level declarator  
## namespace Eigen {  
## ^  
## ;  
## In file included from <built-in>:1:  
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/StanHeaders/include/stan/math/prim/mat/fun/Eigen.hpp:13:  
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/Dense:1:  
## /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/Core:96:10: fatal error: 'complex' file not found  
## #include <complex>  
## ^~~~~~~~~  
## 3 errors generated.  
## make: \*\*\* [foo.o] Error 1  
##   
## SAMPLING FOR MODEL 'f8cd0f33e0bccd8d2f0625b0c6227507' NOW (CHAIN 1).  
## Chain 1:   
## Chain 1: Gradient evaluation took 4.2e-05 seconds  
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.42 seconds.  
## Chain 1: Adjust your expectations accordingly!  
## Chain 1:   
## Chain 1:   
## Chain 1: Iteration: 1 / 10000 [ 0%] (Warmup)  
## Chain 1: Iteration: 1000 / 10000 [ 10%] (Warmup)  
## Chain 1: Iteration: 2000 / 10000 [ 20%] (Warmup)  
## Chain 1: Iteration: 3000 / 10000 [ 30%] (Warmup)  
## Chain 1: Iteration: 4000 / 10000 [ 40%] (Warmup)  
## Chain 1: Iteration: 5000 / 10000 [ 50%] (Warmup)  
## Chain 1: Iteration: 5001 / 10000 [ 50%] (Sampling)  
## Chain 1: Iteration: 6000 / 10000 [ 60%] (Sampling)  
## Chain 1: Iteration: 7000 / 10000 [ 70%] (Sampling)  
## Chain 1: Iteration: 8000 / 10000 [ 80%] (Sampling)  
## Chain 1: Iteration: 9000 / 10000 [ 90%] (Sampling)  
## Chain 1: Iteration: 10000 / 10000 [100%] (Sampling)  
## Chain 1:   
## Chain 1: Elapsed Time: 0.328314 seconds (Warm-up)  
## Chain 1: 0.325585 seconds (Sampling)  
## Chain 1: 0.653899 seconds (Total)  
## Chain 1:   
##   
## SAMPLING FOR MODEL 'f8cd0f33e0bccd8d2f0625b0c6227507' NOW (CHAIN 2).  
## Chain 2:   
## Chain 2: Gradient evaluation took 1.9e-05 seconds  
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.19 seconds.  
## Chain 2: Adjust your expectations accordingly!  
## Chain 2:   
## Chain 2:   
## Chain 2: Iteration: 1 / 10000 [ 0%] (Warmup)  
## Chain 2: Iteration: 1000 / 10000 [ 10%] (Warmup)  
## Chain 2: Iteration: 2000 / 10000 [ 20%] (Warmup)  
## Chain 2: Iteration: 3000 / 10000 [ 30%] (Warmup)  
## Chain 2: Iteration: 4000 / 10000 [ 40%] (Warmup)  
## Chain 2: Iteration: 5000 / 10000 [ 50%] (Warmup)  
## Chain 2: Iteration: 5001 / 10000 [ 50%] (Sampling)  
## Chain 2: Iteration: 6000 / 10000 [ 60%] (Sampling)  
## Chain 2: Iteration: 7000 / 10000 [ 70%] (Sampling)  
## Chain 2: Iteration: 8000 / 10000 [ 80%] (Sampling)  
## Chain 2: Iteration: 9000 / 10000 [ 90%] (Sampling)  
## Chain 2: Iteration: 10000 / 10000 [100%] (Sampling)  
## Chain 2:   
## Chain 2: Elapsed Time: 0.348664 seconds (Warm-up)  
## Chain 2: 0.321741 seconds (Sampling)  
## Chain 2: 0.670405 seconds (Total)  
## Chain 2:   
##   
## SAMPLING FOR MODEL 'f8cd0f33e0bccd8d2f0625b0c6227507' NOW (CHAIN 3).  
## Chain 3:   
## Chain 3: Gradient evaluation took 2.6e-05 seconds  
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.26 seconds.  
## Chain 3: Adjust your expectations accordingly!  
## Chain 3:   
## Chain 3:   
## Chain 3: Iteration: 1 / 10000 [ 0%] (Warmup)  
## Chain 3: Iteration: 1000 / 10000 [ 10%] (Warmup)  
## Chain 3: Iteration: 2000 / 10000 [ 20%] (Warmup)  
## Chain 3: Iteration: 3000 / 10000 [ 30%] (Warmup)  
## Chain 3: Iteration: 4000 / 10000 [ 40%] (Warmup)  
## Chain 3: Iteration: 5000 / 10000 [ 50%] (Warmup)  
## Chain 3: Iteration: 5001 / 10000 [ 50%] (Sampling)  
## Chain 3: Iteration: 6000 / 10000 [ 60%] (Sampling)  
## Chain 3: Iteration: 7000 / 10000 [ 70%] (Sampling)  
## Chain 3: Iteration: 8000 / 10000 [ 80%] (Sampling)  
## Chain 3: Iteration: 9000 / 10000 [ 90%] (Sampling)  
## Chain 3: Iteration: 10000 / 10000 [100%] (Sampling)  
## Chain 3:   
## Chain 3: Elapsed Time: 0.387747 seconds (Warm-up)  
## Chain 3: 0.355312 seconds (Sampling)  
## Chain 3: 0.743059 seconds (Total)  
## Chain 3:   
##   
## SAMPLING FOR MODEL 'f8cd0f33e0bccd8d2f0625b0c6227507' NOW (CHAIN 4).  
## Chain 4:   
## Chain 4: Gradient evaluation took 1.9e-05 seconds  
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.19 seconds.  
## Chain 4: Adjust your expectations accordingly!  
## Chain 4:   
## Chain 4:   
## Chain 4: Iteration: 1 / 10000 [ 0%] (Warmup)  
## Chain 4: Iteration: 1000 / 10000 [ 10%] (Warmup)  
## Chain 4: Iteration: 2000 / 10000 [ 20%] (Warmup)  
## Chain 4: Iteration: 3000 / 10000 [ 30%] (Warmup)  
## Chain 4: Iteration: 4000 / 10000 [ 40%] (Warmup)  
## Chain 4: Iteration: 5000 / 10000 [ 50%] (Warmup)  
## Chain 4: Iteration: 5001 / 10000 [ 50%] (Sampling)  
## Chain 4: Iteration: 6000 / 10000 [ 60%] (Sampling)  
## Chain 4: Iteration: 7000 / 10000 [ 70%] (Sampling)  
## Chain 4: Iteration: 8000 / 10000 [ 80%] (Sampling)  
## Chain 4: Iteration: 9000 / 10000 [ 90%] (Sampling)  
## Chain 4: Iteration: 10000 / 10000 [100%] (Sampling)  
## Chain 4:   
## Chain 4: Elapsed Time: 0.321748 seconds (Warm-up)  
## Chain 4: 0.291384 seconds (Sampling)  
## Chain 4: 0.613132 seconds (Total)  
## Chain 4:

## Warning:   
## 1 (0.8%) p\_waic estimates greater than 0.4. We recommend trying loo instead.

## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.

p\_col\_namses = c("window\_start", "window\_end",  
 "mu", "mu\_2.5", "mu\_97.5",  
 "alpha0", "alpha0\_2.5", "alpha0\_97.5",  
 "alpha1", "alpha1\_2.5", "alpha1\_97.5",  
 "WAIC", "LOOIC", "Rbsq",  
 "MAE\_ret", "MAPE\_ret", "MSE\_ret",  
 "MAE\_ind", "MAPE\_ind", "MSE\_ind")  
p = data.frame(matrix(vector(), n\_steps + 5, length(p\_col\_namses),   
 dimnames = list(c(),p\_col\_namses)),  
 stringsAsFactors=F)  
  
for (i in 1:n\_steps){  
 p[i, "window\_start"] = as.character(as.Date(names(f)[i]) - months(subset\_duration))  
 p[i, "window\_end"] = as.character(as.Date(names(f)[i]))  
 p[i, c("mu", "mu\_2.5", "mu\_97.5")] = f[[i]]$summary$summary[,c("mean", "2.5%", "97.5%")][c("mu"),]  
 p[i, c("alpha0", "alpha0\_2.5", "alpha0\_97.5")] = f[[i]]$summary$summary[,c("mean", "2.5%", "97.5%")][c("alpha0"),]  
 p[i, c("alpha1", "alpha1\_2.5", "alpha1\_97.5")] = f[[i]]$summary$summary[,c("mean", "2.5%", "97.5%")][c("alpha1"),]  
 p[i, c("WAIC")] = f[[i]]$waic$waic  
 p[i, c("LOOIC")] = f[[i]]$looic$looic  
 #p[i, c("Rbsq")] = f[[i]]$Rbsq  
 p[i, c("MAE\_ret", "MAPE\_ret", "MSE\_ret", "MAE\_ind", "MAPE\_ind", "MSE\_ind")] = f[[i]][c("MAE\_ret", "MAPE\_ret", "MSE\_ret", "MAE\_ind", "MAPE\_ind", "MSE\_ind")]  
  
}

## Warning: Accessing waic using '$' is deprecated and will be removed in a  
## future release. Please extract the waic estimate from the 'estimates' component  
## instead.

## Warning: Accessing looic using '$' is deprecated and will be removed in a  
## future release. Please extract the looic estimate from the 'estimates' component  
## instead.

p[n\_steps + 1, "window\_end"] = "MEAN"  
p[n\_steps + 2, "window\_end"] = "SE"  
p[n\_steps + 3, "window\_end"] = "SE, %"  
  
standard\_error <- function(x, ...) sqrt(var(x, ...)/length(x))  
options(digits=3)  
for (i in 3:dim(p)[2]) {  
 p[(n\_steps + 1), i] = mean(p[1:n\_steps, i])  
 p[(n\_steps + 2), i] = standard\_error(p[1:n\_steps, i])  
 if (i>=3) {p[(n\_steps + 3), i] = round(100 \* p[(n\_steps + 2), i] / p[(n\_steps + 1), i], digits = 2)}  
}  
p[(n\_steps + 4), 1] = "iter"  
p[(n\_steps + 4), 2] = iter  
p[(n\_steps + 5), 1] = "chais"  
p[(n\_steps + 5), 2] = chains  
setwd('/Users/AM/Documents/\_CU Masters/2020 fall Bayesian\_7393/Final\_Project/output')  
write.xlsx(p,   
 paste(model\_name, as.character(Sys.time()), ".xlsx", sep="\_"),  
 sheetName=model\_name)

model\_code = stan\_mod\_GARCH11  
model\_name = "GARCH11"  
start\_date = "2017-01-22"  
subset\_duration = 6 # in months, incl 1 extra month for T\_out (validation)  
step\_size = 2 # in months  
n\_steps = 1 # number of different time windows (to build different models) # changed from 10 for to print out the code  
T\_out <- 9 # forecast depth, days  
  
iter = 10^4  
chains = 4  
max\_treedepth = 12  
  
sigma\_1 <- 0.1   
mu\_1 = 0  
  
f = forecast\_vix(model\_code = model\_code, start\_date = start\_date,   
 subset\_duration = subset\_duration,   
 step\_size = step\_size,n\_steps = n\_steps, T\_out = T\_out,   
 sigma\_1 = sigma\_1, mu\_1 = mu\_1, iter = iter, chains = chains,   
 Rbsq\_presence = FALSE, max\_treedepth = max\_treedepth)

## Trying to compile a simple C file

## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c  
## clang -mmacosx-version-min=10.13 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG -I"/Library/Frameworks/R.framework/Versions/4.0/Resources/library/Rcpp/include/" -I"/Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/" -I"/Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/unsupported" -I"/Library/Frameworks/R.framework/Versions/4.0/Resources/library/BH/include" -I"/Library/Frameworks/R.framework/Versions/4.0/Resources/library/StanHeaders/include/src/" -I"/Library/Frameworks/R.framework/Versions/4.0/Resources/library/StanHeaders/include/" -I"/Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppParallel/include/" -I"/Library/Frameworks/R.framework/Versions/4.0/Resources/library/rstan/include" -DEIGEN\_NO\_DEBUG -DBOOST\_DISABLE\_ASSERTS -DBOOST\_PENDING\_INTEGER\_LOG2\_HPP -DSTAN\_THREADS -DBOOST\_NO\_AUTO\_PTR -include '/Library/Frameworks/R.framework/Versions/4.0/Resources/library/StanHeaders/include/stan/math/prim/mat/fun/Eigen.hpp' -D\_REENTRANT -DRCPP\_PARALLEL\_USE\_TBB=1 -I/usr/local/include -fPIC -Wall -g -O2 -c foo.c -o foo.o  
## In file included from <built-in>:1:  
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/StanHeaders/include/stan/math/prim/mat/fun/Eigen.hpp:13:  
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/Dense:1:  
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/Core:88:  
## /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:1: error: unknown type name 'namespace'  
## namespace Eigen {  
## ^  
## /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:16: error: expected ';' after top level declarator  
## namespace Eigen {  
## ^  
## ;  
## In file included from <built-in>:1:  
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/StanHeaders/include/stan/math/prim/mat/fun/Eigen.hpp:13:  
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/Dense:1:  
## /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/Core:96:10: fatal error: 'complex' file not found  
## #include <complex>  
## ^~~~~~~~~  
## 3 errors generated.  
## make: \*\*\* [foo.o] Error 1  
##   
## SAMPLING FOR MODEL 'cb6da87cadca17549a3b5f36385e4b22' NOW (CHAIN 1).  
## Chain 1:   
## Chain 1: Gradient evaluation took 6e-05 seconds  
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.6 seconds.  
## Chain 1: Adjust your expectations accordingly!  
## Chain 1:   
## Chain 1:   
## Chain 1: Iteration: 1 / 10000 [ 0%] (Warmup)  
## Chain 1: Iteration: 1000 / 10000 [ 10%] (Warmup)  
## Chain 1: Iteration: 2000 / 10000 [ 20%] (Warmup)  
## Chain 1: Iteration: 3000 / 10000 [ 30%] (Warmup)  
## Chain 1: Iteration: 4000 / 10000 [ 40%] (Warmup)  
## Chain 1: Iteration: 5000 / 10000 [ 50%] (Warmup)  
## Chain 1: Iteration: 5001 / 10000 [ 50%] (Sampling)  
## Chain 1: Iteration: 6000 / 10000 [ 60%] (Sampling)  
## Chain 1: Iteration: 7000 / 10000 [ 70%] (Sampling)  
## Chain 1: Iteration: 8000 / 10000 [ 80%] (Sampling)  
## Chain 1: Iteration: 9000 / 10000 [ 90%] (Sampling)  
## Chain 1: Iteration: 10000 / 10000 [100%] (Sampling)  
## Chain 1:   
## Chain 1: Elapsed Time: 1.17014 seconds (Warm-up)  
## Chain 1: 1.15701 seconds (Sampling)  
## Chain 1: 2.32716 seconds (Total)  
## Chain 1:   
##   
## SAMPLING FOR MODEL 'cb6da87cadca17549a3b5f36385e4b22' NOW (CHAIN 2).  
## Chain 2:   
## Chain 2: Gradient evaluation took 2.9e-05 seconds  
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.29 seconds.  
## Chain 2: Adjust your expectations accordingly!  
## Chain 2:   
## Chain 2:   
## Chain 2: Iteration: 1 / 10000 [ 0%] (Warmup)  
## Chain 2: Iteration: 1000 / 10000 [ 10%] (Warmup)  
## Chain 2: Iteration: 2000 / 10000 [ 20%] (Warmup)  
## Chain 2: Iteration: 3000 / 10000 [ 30%] (Warmup)  
## Chain 2: Iteration: 4000 / 10000 [ 40%] (Warmup)  
## Chain 2: Iteration: 5000 / 10000 [ 50%] (Warmup)  
## Chain 2: Iteration: 5001 / 10000 [ 50%] (Sampling)  
## Chain 2: Iteration: 6000 / 10000 [ 60%] (Sampling)  
## Chain 2: Iteration: 7000 / 10000 [ 70%] (Sampling)  
## Chain 2: Iteration: 8000 / 10000 [ 80%] (Sampling)  
## Chain 2: Iteration: 9000 / 10000 [ 90%] (Sampling)  
## Chain 2: Iteration: 10000 / 10000 [100%] (Sampling)  
## Chain 2:   
## Chain 2: Elapsed Time: 1.10258 seconds (Warm-up)  
## Chain 2: 1.16746 seconds (Sampling)  
## Chain 2: 2.27003 seconds (Total)  
## Chain 2:   
##   
## SAMPLING FOR MODEL 'cb6da87cadca17549a3b5f36385e4b22' NOW (CHAIN 3).  
## Chain 3:   
## Chain 3: Gradient evaluation took 2.5e-05 seconds  
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.25 seconds.  
## Chain 3: Adjust your expectations accordingly!  
## Chain 3:   
## Chain 3:   
## Chain 3: Iteration: 1 / 10000 [ 0%] (Warmup)  
## Chain 3: Iteration: 1000 / 10000 [ 10%] (Warmup)  
## Chain 3: Iteration: 2000 / 10000 [ 20%] (Warmup)  
## Chain 3: Iteration: 3000 / 10000 [ 30%] (Warmup)  
## Chain 3: Iteration: 4000 / 10000 [ 40%] (Warmup)  
## Chain 3: Iteration: 5000 / 10000 [ 50%] (Warmup)  
## Chain 3: Iteration: 5001 / 10000 [ 50%] (Sampling)  
## Chain 3: Iteration: 6000 / 10000 [ 60%] (Sampling)  
## Chain 3: Iteration: 7000 / 10000 [ 70%] (Sampling)  
## Chain 3: Iteration: 8000 / 10000 [ 80%] (Sampling)  
## Chain 3: Iteration: 9000 / 10000 [ 90%] (Sampling)  
## Chain 3: Iteration: 10000 / 10000 [100%] (Sampling)  
## Chain 3:   
## Chain 3: Elapsed Time: 1.17795 seconds (Warm-up)  
## Chain 3: 1.05283 seconds (Sampling)  
## Chain 3: 2.23078 seconds (Total)  
## Chain 3:   
##   
## SAMPLING FOR MODEL 'cb6da87cadca17549a3b5f36385e4b22' NOW (CHAIN 4).  
## Chain 4:   
## Chain 4: Gradient evaluation took 3.2e-05 seconds  
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.32 seconds.  
## Chain 4: Adjust your expectations accordingly!  
## Chain 4:   
## Chain 4:   
## Chain 4: Iteration: 1 / 10000 [ 0%] (Warmup)  
## Chain 4: Iteration: 1000 / 10000 [ 10%] (Warmup)  
## Chain 4: Iteration: 2000 / 10000 [ 20%] (Warmup)  
## Chain 4: Iteration: 3000 / 10000 [ 30%] (Warmup)  
## Chain 4: Iteration: 4000 / 10000 [ 40%] (Warmup)  
## Chain 4: Iteration: 5000 / 10000 [ 50%] (Warmup)  
## Chain 4: Iteration: 5001 / 10000 [ 50%] (Sampling)  
## Chain 4: Iteration: 6000 / 10000 [ 60%] (Sampling)  
## Chain 4: Iteration: 7000 / 10000 [ 70%] (Sampling)  
## Chain 4: Iteration: 8000 / 10000 [ 80%] (Sampling)  
## Chain 4: Iteration: 9000 / 10000 [ 90%] (Sampling)  
## Chain 4: Iteration: 10000 / 10000 [100%] (Sampling)  
## Chain 4:   
## Chain 4: Elapsed Time: 1.08866 seconds (Warm-up)  
## Chain 4: 0.998306 seconds (Sampling)  
## Chain 4: 2.08697 seconds (Total)  
## Chain 4:

## Warning: There were 1 divergent transitions after warmup. See  
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup  
## to find out why this is a problem and how to eliminate them.

## Warning: Examine the pairs() plot to diagnose sampling problems

## Warning:   
## 1 (0.8%) p\_waic estimates greater than 0.4. We recommend trying loo instead.

## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.

p\_col\_namses = c("window\_start", "window\_end",  
 "mu", "mu\_2.5", "mu\_97.5",  
 "alpha0", "alpha0\_2.5", "alpha0\_97.5",  
 "alpha1", "alpha1\_2.5", "alpha1\_97.5",  
 "beta1", "beta1\_2.5", "beta1\_97.5",  
 "WAIC", "LOOIC", "Rbsq",  
 "MAE\_ret", "MAPE\_ret", "MSE\_ret",  
 "MAE\_ind", "MAPE\_ind", "MSE\_ind")  
p = data.frame(matrix(vector(), n\_steps + 5, length(p\_col\_namses),   
 dimnames = list(c(),p\_col\_namses)),  
 stringsAsFactors=F)  
  
for (i in 1:n\_steps){  
 p[i, "window\_start"] = as.character(as.Date(names(f)[i]) - months(subset\_duration))  
 p[i, "window\_end"] = as.character(as.Date(names(f)[i]))  
 p[i, c("mu", "mu\_2.5", "mu\_97.5")] = f[[i]]$summary$summary[,c("mean", "2.5%", "97.5%")][c("mu"),]  
 p[i, c("alpha0", "alpha0\_2.5", "alpha0\_97.5")] = f[[i]]$summary$summary[,c("mean", "2.5%", "97.5%")][c("alpha0"),]  
 p[i, c("alpha1", "alpha1\_2.5", "alpha1\_97.5")] = f[[i]]$summary$summary[,c("mean", "2.5%", "97.5%")][c("alpha1"),]  
 p[i, c("beta1", "beta1\_2.5", "beta1\_97.5")] = f[[i]]$summary$summary[,c("mean", "2.5%", "97.5%")][c("beta1"),]  
 p[i, c("WAIC")] = f[[i]]$waic$waic  
 p[i, c("LOOIC")] = f[[i]]$looic$looic  
 #p[i, c("Rbsq")] = f[[i]]$Rbsq  
 p[i, c("MAE\_ret", "MAPE\_ret", "MSE\_ret", "MAE\_ind", "MAPE\_ind", "MSE\_ind")] = f[[i]][c("MAE\_ret", "MAPE\_ret", "MSE\_ret", "MAE\_ind", "MAPE\_ind", "MSE\_ind")]  
  
}

## Warning: Accessing waic using '$' is deprecated and will be removed in a  
## future release. Please extract the waic estimate from the 'estimates' component  
## instead.

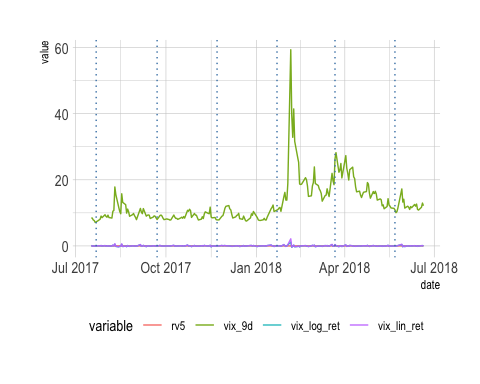
## Warning: Accessing looic using '$' is deprecated and will be removed in a  
## future release. Please extract the looic estimate from the 'estimates' component  
## instead.

p[n\_steps + 1, "window\_end"] = "MEAN"  
p[n\_steps + 2, "window\_end"] = "SE"  
p[n\_steps + 3, "window\_end"] = "SE, %"  
  
standard\_error <- function(x, ...) sqrt(var(x, ...)/length(x))  
options(digits=3)  
for (i in 3:dim(p)[2]) {  
 p[(n\_steps + 1), i] = mean(p[1:n\_steps, i])  
 p[(n\_steps + 2), i] = standard\_error(p[1:n\_steps, i])  
 if (i>=3) {p[(n\_steps + 3), i] = round(100 \* p[(n\_steps + 2), i] / p[(n\_steps + 1), i], digits = 2)}  
}  
p[(n\_steps + 4), 1] = "iter"  
p[(n\_steps + 4), 2] = iter  
p[(n\_steps + 5), 1] = "chais"  
p[(n\_steps + 5), 2] = chains  
  
setwd('/Users/AM/Documents/\_CU Masters/2020 fall Bayesian\_7393/Final\_Project/output')  
write.xlsx(p,   
 paste(model\_name, as.character(Sys.time()), ".xlsx", sep="\_"),  
 sheetName=model\_name)

model\_code = stan\_mod\_RealGARCH11  
model\_name = "RealGARCH11"  
start\_date = "2017-05-22"  
subset\_duration = 6 # in months, incl 1 extra month for T\_out (validation)  
step\_size = 2 # in months  
n\_steps = 1 # number of different time windows (to build different models)   
T\_out <- 9 # forecast depth, days  
  
iter = 10^3  
chains = 1  
max\_treedepth = 35  
  
sigma\_1 <- 0.1 # sqrt(var(lin\_vix\_ret)) > sqrt(var(rv\_vix\_subset$vix\_lin\_ret[-c(1)]))  
mu\_1 = 0  
  
#commented to provide output  
#f = forecast\_vix(model\_code = model\_code, start\_date = start\_date,   
# subset\_duration = subset\_duration,   
# step\_size = step\_size,n\_steps = n\_steps, T\_out = T\_out,   
# sigma\_1 = sigma\_1, mu\_1 = mu\_1, iter = iter, chains = chains,   
# Rbsq\_presence = FALSE, max\_treedepth = max\_treedepth)   
# p\_col\_namses = c("window\_start", "window\_end",  
# "mu", "mu\_2.5", "mu\_97.5",  
# "alpha0", "alpha0\_2.5", "alpha0\_97.5",  
# "beta1", "beta1\_2.5", "beta1\_97.5",  
# "gamma", "gamma\_2.5", "gamma\_97.5",  
# "xi", "xi\_2.5", "xi\_97.5",  
# "phi", "phi\_2.5", "phi\_97.5",  
# "tau\_1", "tau\_1\_2.5", "tau\_1\_97.5",  
# "tau\_2", "tau\_2\_2.5", "tau\_2\_97.5",  
# "WAIC", "LOOIC", "Rbsq",  
# "MAE\_ret", "MAPE\_ret", "MSE\_ret",  
# "MAE\_ind", "MAPE\_ind", "MSE\_ind")  
# p = data.frame(matrix(vector(), n\_steps + 5, length(p\_col\_namses),   
# dimnames = list(c(),p\_col\_namses)),  
# stringsAsFactors=F)  
#   
# for (i in 1:n\_steps){  
# p[i, "window\_start"] = as.character(as.Date(names(f)[i]) - months(subset\_duration))  
# p[i, "window\_end"] = as.character(as.Date(names(f)[i]))  
# p[i, c("mu", "mu\_2.5", "mu\_97.5")] = f[[i]]$summary$summary[,c("mean", "2.5%", "97.5%")][c("mu"),]  
# p[i, c("alpha0", "alpha0\_2.5", "alpha0\_97.5")] = f[[i]]$summary$summary[,c("mean", "2.5%", "97.5%")][c("alpha0"),]  
# p[i, c("beta1", "beta1\_2.5", "beta1\_97.5")] = f[[i]]$summary$summary[,c("mean", "2.5%", "97.5%")][c("beta1"),]  
# p[i, c("gamma", "gamma\_2.5", "gamma\_97.5")] = f[[i]]$summary$summary[,c("mean", "2.5%", "97.5%")][c("gamma"),]  
# p[i, c("xi", "xi\_2.5", "xi\_97.5")] = f[[i]]$summary$summary[,c("mean", "2.5%", "97.5%")][c("xi"),]  
# p[i, c("phi", "phi\_2.5", "phi\_97.5")] = f[[i]]$summary$summary[,c("mean", "2.5%", "97.5%")][c("phi"),]  
# p[i, c("tau\_1", "tau\_1\_2.5", "tau\_1\_97.5")] = f[[i]]$summary$summary[,c("mean", "2.5%", "97.5%")][c("tau\_1"),]  
# p[i, c("tau\_2", "tau\_2\_2.5", "tau\_2\_97.5")] = f[[i]]$summary$summary[,c("mean", "2.5%", "97.5%")][c("tau\_2"),]  
# p[i, c("WAIC")] = f[[i]]$waic$waic  
# p[i, c("LOOIC")] = f[[i]]$looic$looic  
# #p[i, c("Rbsq")] = f[[i]]$Rbsq  
# p[i, c("MAE\_ret", "MAPE\_ret", "MSE\_ret", "MAE\_ind", "MAPE\_ind", "MSE\_ind")] = f[[i]][c("MAE\_ret", "MAPE\_ret", "MSE\_ret", "MAE\_ind", "MAPE\_ind", "MSE\_ind")]  
#   
# }  
#   
# p[n\_steps + 1, "window\_end"] = "MEAN"  
# p[n\_steps + 2, "window\_end"] = "SE"  
# p[n\_steps + 3, "window\_end"] = "SE, %"  
#   
# standard\_error <- function(x, ...) sqrt(var(x, ...)/length(x))  
# options(digits=3)  
# for (i in 3:dim(p)[2]) {  
# p[(n\_steps + 1), i] = mean(p[1:n\_steps, i])  
# p[(n\_steps + 2), i] = standard\_error(p[1:n\_steps, i])  
# if (i>=3) {p[(n\_steps + 3), i] = round(100 \* p[(n\_steps + 2), i] / p[(n\_steps + 1), i], digits = 2)}  
# }  
# p[(n\_steps + 4), 1] = "iter"  
# p[(n\_steps + 4), 2] = iter  
# p[(n\_steps + 5), 1] = "chais"  
# p[(n\_steps + 5), 2] = chains  
#   
#   
# setwd('/Users/AM/Documents/\_CU Masters/2020 fall Bayesian\_7393/Final\_Project/output')  
# write.xlsx(p,   
# paste(model\_name, as.character(Sys.time()), ".xlsx", sep="\_"),  
# sheetName=model\_name)

start\_date = "2017-01-22"  
subset\_duration = 18 # in months, incl 1 extra month for T\_out (validation)  
step\_size = 2 # in months  
n\_steps = 1  
t\_0 = as.Date(start\_date)  
t\_1 = as.Date(t\_0) + months(subset\_duration) #stored end\_date, for subset  
subset = build\_vix9\_rv\_subset(t\_0, t\_1)  
  
#subset$vix\_9d\_out = NA  
  
# temp = data\_frame()  
# for (i in 1:9) {  
# temp = rbind(temp, f\_GARCH[[i]]$returns[,c("date", "price\_r\_out")])  
# }  
# subset = left\_join(subset, temp, by = "date")[,c(1,3,6)]  
# names(subset)[3] = "vix\_9d\_predicted"  
# names(subset)[2] = "vix\_9d\_true"  
  
  
rv\_long <- melt(subset, id="date")  
ggplot(data=rv\_long, aes(x=date, y=value, colour=variable)) +   
 geom\_line(alpha=.9) +  
 theme\_ipsum() +  
 geom\_vline(xintercept = as.Date(c("2017-07-22", "2017-09-22", "2017-11-22", "2018-01-22", "2018-03-22", "2018-05-22")),   
 show.legend = T, colour = "steel blue", linetype="dotted") +  
#ylim(0,0.0007) +  
 scale\_x\_date(limit=c(as.Date("2017-07-15"),as.Date("2018-06-20"))) +   
 theme(legend.position="bottom")

## Warning: Removed 564 row(s) containing missing values (geom\_path).



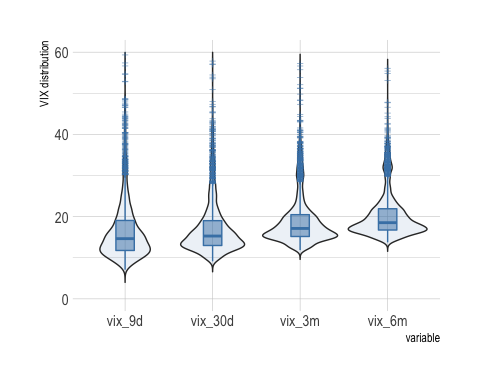
remove(rv\_long)

temp\_long = melt(vix\_df, id="date")  
ggplot(temp\_long, aes(y = value, x = variable)) +   
 geom\_violin(trim = F, show.legend = F, fill = "steel blue", alpha = .1) +   
 geom\_boxplot(width=0.21, fill="steel blue", outlier.color="steel blue", alpha = .5,  
 outlier.shape=3, color="steel blue", outlier.size=1) +   
 ylim(0,60) +   
 theme\_ipsum() +  
 ylab("VIX distribution")

## Warning: Removed 35 rows containing non-finite values (stat\_ydensity).

## Warning: Removed 35 rows containing non-finite values (stat\_boxplot).

## Warning: Removed 26 rows containing missing values (geom\_violin).



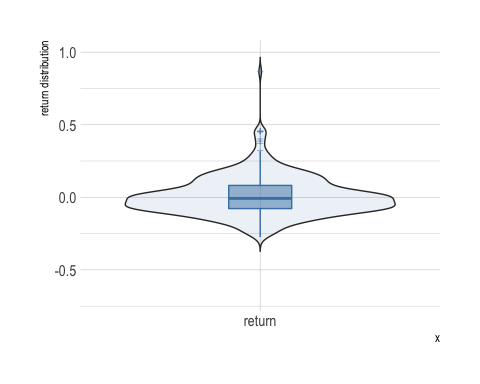
summary(temp\_long$value)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 7.1 14.3 16.8 18.6 20.6 106.7

rm(temp\_long)  
  
temp\_long = rv\_vix\_subset[, c("date", "vix\_lin\_ret")]  
ggplot(temp\_long, aes(y = vix\_lin\_ret, x = "return")) +   
 geom\_violin(trim = F, show.legend = F, fill = "steel blue", alpha = .1) +   
 geom\_boxplot(width=0.21, fill="steel blue", outlier.color="steel blue", alpha = .5,  
 outlier.shape=3, color="steel blue", outlier.size=1) +   
 ylim(-.7, 1) +   
 theme\_ipsum() +  
 ylab("return distribution")

## Warning: Removed 1 rows containing non-finite values (stat\_ydensity).

## Warning: Removed 1 rows containing non-finite values (stat\_boxplot).

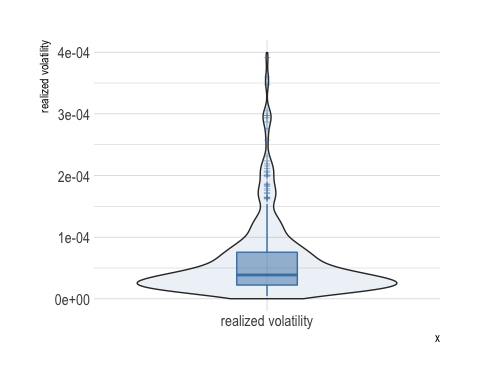


rm(temp\_long)  
  
temp\_long = rv\_vix\_subset[, c("date", "rv5")]  
ggplot(temp\_long, aes(y = rv5, x = "realized volatility")) +   
 geom\_violin(trim = F, show.legend = F, fill = "steel blue", alpha = .1) +   
 geom\_boxplot(width=0.21, fill="steel blue", outlier.color="steel blue", alpha = .5,  
 outlier.shape=3, color="steel blue", outlier.size=1) +   
 ylim(0, 0.0004) +   
 theme\_ipsum() +  
 ylab("realized volatility")

## Warning: Removed 7 rows containing non-finite values (stat\_ydensity).

## Warning: Removed 7 rows containing non-finite values (stat\_boxplot).

## Warning: Removed 60 rows containing missing values (geom\_violin).



summary(temp\_long$rv5)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.00000 0.00002 0.00004 0.00008 0.00008 0.00373

rm(temp\_long)