1 Empirical Findings

1.1 Density of the returns

1.1.1 MLE distribution parameters

In table 1.1 we can see the estimated parameters of the unconditional distribution functions. They are presented for the Skewed Generalized T-distribution (SGT) and limiting cases thereof previously discussed. Additionally, maximum likelihood score and the Aikake Information Criterion (AIC) is reported to compare goodness of fit of the different distributions. We find that the SGT-distribution has the highest maximum likelihood score of all. All other distributions have relatively similar likelihood scores, though slightly lower and are therefore not the optimal distributions. However, when considering AIC it is a tie between SGT and SGED. This provides some indication that we have a valid case to test the suitability of different SGED-GARCH VaR models as an alternative for the SGT-GARCH VaR models. While sacrificing some goodness of fit, the SGED distribution has the advantage of requiring one less parameter, which could possibly result in less errors due to misspecification and easier implementation. For the SGT parameters the standard deviation and skewness are both significant at the 1% level. For the SGED parameters, the standard deviation and the skewness are both significant

at respectively the 1% and 5% level. Both distributions are right-skewed. For both distributions the shape parameters are significant at the 1% level, though the q parameter was not estimated as it is by design set to infinity due to the SGED being a limiting case of SGT.[1]

Additionally, for every distribution fitted with MLE, plots are generated to compare the theoretical distribution with the observed returns. We see that except for the normal distribution which is quite off, the theoretical distributions are close to the actual data, except that they are too peaked. This problem is the least present for the SGT distribution.

Table 1.1: Maximum likelihood estimates of unconditional distribution functions

θ	α	β	ξ	κ	η	LLH	AIC
SGT	0.02	1.321	-0.04	1.381	3.314	-13973.01	27956.01
	(0.013)	(0.026)***	(0.013)***	(0.071)***	(0.538)***		
SGED	0.019	1.274	-0.018	0.916	Inf	-14008.63	27956.01
	(0.013)	(0.016)***	(0.01)***	(0.017)***			
GED	0.032	1.276	0	0.911	Inf	-14009.52	28025.04
	(0.009)***	(0.016)***		(0.017)***			
ST	0.019	1.481	-0.052	2	2.793	-13997.35	28002.71
	(0.014)	(0.054)***	(0.013)***		(0.098)***		
T	0.056	1.494	0	2	1.383	-14005.14	28016.29
	(0.01)***	(0.056)***			(0.097)***		
Normal	0.017	1.307	0	2	Inf	-15101.73	30207.46
	(0.014)	(0.01)***					

Table contains parameter estimates for SGT-distribution and some of its limiting cases. The underlying data is the daily return series of the Eurostoxx 50 for the period between December 31. 1986 and April 27. 2021. Standard errors are reported between brackets. L is the maximum log-likelihood value. *, ** and *** point out significance at 10

[1]To check whether the relative ranking of distributions still holds in different periods, we have calculated the maximum likelihood score and AIC for three smaller periods: The period up to the dotcom collapse (1987-2001), up to the GFC (2002-2009) and up to the present Covid-crash (2009-2021). There is no qualitative difference in relative ranking with these subsamples. Results are reported in the appendix.

1.2 Results of GARCH with constant higher moments

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(1,0,0)
## Distribution : sged
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.042288    0.009060   4.6677 0.000003
## ar1    -0.021117    0.010460  -2.0190 0.043492
## omega   0.019836    0.003115   6.3677 0.000000
## alpha1  0.096800    0.008038  12.0422 0.000000
## beta1   0.892028    0.008400 106.1906 0.000000
## skew    0.927350    0.012959  71.5597 0.000000
## shape   1.313056    0.025902  50.6923 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.042288    0.009106   4.6439 0.000003
## ar1    -0.021117    0.009974  -2.1173 0.034236
## omega   0.019836    0.003995   4.9650 0.000001
```

```
## alpha1  0.096800    0.010950   8.8398 0.000000
## beta1   0.892028    0.011277  79.1047 0.000000
## skew    0.927350    0.015679  59.1450 0.000000
## shape   1.313056    0.043162  30.4215 0.000000
##
## LogLikelihood : -13095.15
##
## Information Criteria
## ------------------------------------
##
## Akaike       2.9265
## Bayes        2.9321
## Shibata      2.9265
## Hannan-Quinn 2.9284
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic   p-value
## Lag[1]                      5.746 1.652e-02
## Lag[2*(p+q)+(p+q)-1][2]     5.768 6.987e-05
## Lag[4*(p+q)+(p+q)-1][5]     7.340 1.416e-02
## d.o.f=1
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.1424  0.7059
## Lag[2*(p+q)+(p+q)-1][5]    0.9639  0.8678
## Lag[4*(p+q)+(p+q)-1][9]    1.5151  0.9543
```

```
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]     1.338 0.500 2.000  0.2474
## ARCH Lag[5]     1.357 1.440 1.667  0.6307
## ARCH Lag[7]     1.744 2.315 1.543  0.7712
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  2.9961
## Individual Statistics:
## mu     0.1866
## ar1    1.0840
## omega  0.2334
## alpha1 0.5527
## beta1  0.6371
## skew   0.3147
## shape  0.5511
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.69 1.9 2.35
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value      prob sig
## Sign Bias           1.2820 1.999e-01
## Negative Sign Bias  0.9899 3.222e-01
```

```
## Positive Sign Bias  3.0691 2.154e-03 ***
## Joint Effect       29.9419 1.419e-06 ***
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     61.51    2.223e-06
## 2    30     86.73    1.151e-07
## 3    40    108.46    1.836e-08
## 4    50    120.25    6.373e-08
##
##
## Elapsed time : 2.228908
```

```
require(plyr)


Table.GARCH.function <- function(GARCHfit.object = garchfit.eGARCH){
#Making objects to fill
list.table.3.tmp <- vector(mode = "list", length = length(distributions)*2)
list.table.3 <- vector(mode = "list", length = length(distributions))
names(list.table.3) <- distributions
ref.distr <- seq(from = 1, to = length(distributions)*2, by = 2)
#Retriving all the data from the original lists
for(i in 1:length(distributions)){
    list.table.3.tmp[[ref.distr[i]]] <- GARCHfit.object[[i]]@fit$coef
    list.table.3.tmp[[ref.distr[i]+1]] <- GARCHfit.object[[i]]@fit$tval
}
#From list of vectors of different lengths to list of matrixes with empty spaces
for(i in 1:length(distributions)){
  list.table.3[[i]] <- cbind(list.table.3.tmp[[ref.distr[i]]], list.table.3.tmp[[re
```

```
}
#Function to rearrange the list from a list of matrices to list of vectors
list.restructure <- function(object = list.table.3){
  len.table <- length(object)
  len.inside.list <- rep(NA, len.table)
  for(i in 1:len.table){len.inside.list[i] <- nrow(object[[i]])}
  adj.list <- vector(mode = "list", length = len.table)
  ref.list <- vector(mode = "list", length = len.table)
  names.list <- vector(mode = "list", length = len.table)
  for(i in 1:len.table){ref.list[[i]] <- seq(from = 1, to = len.inside.list[i]*2, b
  for(i in 1:len.table){adj.list[[i]] <- names.list[[i]] <- rep(NA, len.inside.list
  for(i in 1:len.table){
      adj.list[[i]][ref.list[[i]]] <- round(object[[i]][,1],3)
      adj.list[[i]][ref.list[[i]]+1] <- paste0("(", round(object[[i]][,2],3),")")
  }
  for(i in 1:len.table){
    names.list[[i]][ref.list[[i]]] <- rownames(object[[i]])
    names.list[[i]][ref.list[[i]]+1] <- paste0("p-val ",rownames(object[[i]]))
    #names.list[[i]][ref.list[[i]]+1] <- ""
    }
  names(adj.list) <- distributions
  for(i in 1:len.table){names(adj.list[[i]]) <- names.list[[i]]}
  return(adj.list)
}
#Unlisting and removing NAs
adj.list <- list.restructure(object = list.table.3)
table.3.matrix <- matrix(unlist(lapply(adj.list, `length<-`, max(lengths(adj.list))
colnames(table.3.matrix) <- names(adj.list)
table.3.matrix[is.na(table.3.matrix)] <- ""
table.3.matrix[table.3.matrix=="(NA)"] <- ""
```

```
#Adjustments for std & ged distributions
table.3.matrix[c(length(table.3.matrix[,2])-1,length(table.3.matrix[,2])),2] <- tai
table.3.matrix[c(length(table.3.matrix[,2])-3,length(table.3.matrix[,2])-2),2] <- '
table.3.matrix[c(length(table.3.matrix[,4])-1,length(table.3.matrix[,4])),4] <- tai
table.3.matrix[c(length(table.3.matrix[,4])-3,length(table.3.matrix[,4])-2),4] <- '
#Log-Likelyhoods
LLH <- rep(NA, length(distributions))
for(i in 1:length(distributions)){LLH[i] <- GARCHfit.object[[i]]@fit$LLH}
names.table.3 <- revalue(names(adj.list[[3]]), c("mu"="$\\alpha_0$", "ar1"="$\\alph
kappa.object <- cbind(matrix(data = "", nrow = 2, ncol = 3) ,table.3.matrix[(nrow(t
eta <- cbind(table.3.matrix[(nrow(table.3.matrix)-1):(nrow(table.3.matrix)),1:3], m
table.3.matrix[(nrow(table.3.matrix)-1):(nrow(table.3.matrix)),] <- kappa.object
table.3.matrix <- rbind(table.3.matrix, eta, round(LLH,3))
table.3.matrix <- cbind(c(names.table.3,"$\\eta$","p-val eta","$LLH$"), table.3.mat
table.3.matrix <- as.data.frame(table.3.matrix, row.names = c(names.table.3,"$\\eta
return(table.3.matrix)
}

#PARTIAL RESULTS

Table.3.iGARCH <- Table.GARCH.function(GARCHfit.object = garchfit.iGARCH)
Table.3.eGARCH <- Table.GARCH.function(GARCHfit.object = garchfit.eGARCH)
Table.3.gjrGARCH <- Table.GARCH.function(GARCHfit.object = garchfit.gjrGARCH)
Table.3.sGARCH <- Table.GARCH.function(GARCHfit.object = garchfit.sGARCH)
Table.3.EWMA <- Table.GARCH.function(GARCHfit.object = garchfit.EWMA)
Table.3.fGARCH.AVGARCH <- Table.GARCH.function(GARCHfit.object = garchfit.fGARCH.AV
Table.3.fGARCH.NAGARCH <- Table.GARCH.function(GARCHfit.object = garchfit.fGARCH.NA
Table.3.fGARCH.TGARCH <- Table.GARCH.function(GARCHfit.object = garchfit.fGARCH.TGA

Table.3.function <- function(distribution = "sstd"){
```

```
ref <- which(names(Table.3.eGARCH)==distribution)#Reference column
#Taking all the relevant values
AVGARCH <- Table.3.fGARCH.AVGARCH[,c(1,ref)]
iGARCH <- Table.3.iGARCH[,c(1,ref)]
eGARCH <- Table.3.eGARCH[,c(1,ref)]
sGARCH <- Table.3.sGARCH[,c(1,ref)]
gjrGARCH <- Table.3.gjrGARCH[,c(1,ref)]
EWMA <- Table.3.EWMA[,c(1,ref)]
NAGARCH <- Table.3.fGARCH.NAGARCH[,c(1,ref)]
TGARCH <- Table.3.fGARCH.TGARCH[,c(1,ref)]
#Assigning the right names to columns
colnames(AVGARCH)[-1] <- rep("AVGARCH", length(colnames(AVGARCH)[-1]))
colnames(iGARCH)[-1] <- rep("iGARCH", length(colnames(iGARCH)[-1]))
colnames(eGARCH)[-1] <- rep("eGARCH", length(colnames(eGARCH)[-1]))
colnames(sGARCH)[-1] <- rep("sGARCH", length(colnames(sGARCH)[-1]))
colnames(gjrGARCH)[-1] <- rep("gjrGARCH", length(colnames(gjrGARCH)[-1]))
colnames(EWMA)[-1] <- rep("EWMA", length(colnames(EWMA)[-1]))
colnames(NAGARCH)[-1] <- rep("NAGARCH", length(colnames(NAGARCH)[-1]))
colnames(TGARCH)[-1] <- rep("TGARCH", length(colnames(TGARCH)[-1]))
#Binding all the columns & cleaning & ordering data
Table3 <- full_join(full_join(full_join(full_join(full_join(full_join(sGA
Table3[is.na(Table3)] <- ""
Table3 <- rbind(Table3[Table3[,1]!="$LLH$",], Table3[Table3[,1]=="$LLH$",])
Table3[substr(Table3[,1],1,5)=="p-val",1] <- ""
colnames(Table3) <- c("", colnames(Table3)[-1])
return(Table3)
}

Table.3 <- vector(mode = "list", length = length(distributions))
names(Table.3) <- c("Norm", "T", "ST", "GED", "SGED")
```

```r
for(i in 1:length(distributions)){
  Table.3[[i]] <- suppressMessages(Table.3.function(distribution = distributions[i]
}


#Testing the kabling
knitr::kable(Table.3$ST)
```

```r
print("iGARCH")
garchfit.iGARCH[[1]]@fit$coef
garchfit.iGARCH[[1]]@fit$se.coef
```

```r
print("EWMA")
garchfit.EWMA[[1]]@fit$coef
c(garchfit.EWMA[[1]]@fit$se.coef[1:2],NA,garchfit.EWMA[[1]]@fit$se.coef[3], NA)
```

```r
print("eGARCH")
garchfit.eGARCH[[1]]@fit$coef
garchfit.eGARCH[[1]]@fit$se.coef
```

```r
print("gjrGARCH")
garchfit.gjrGARCH[[1]]@fit$coef
garchfit.gjrGARCH[[1]]@fit$se.coef
```

```r
print("NAGARCH")
garchfit.fGARCH.NAGARCH[[1]]@fit$coef
garchfit.fGARCH.NAGARCH[[1]]@fit$se.coef
```

```r
print("TGARCH")
garchfit.fGARCH.TGARCH[[1]]@fit$coef
garchfit.fGARCH.TGARCH[[1]]@fit$se.coef
```

```r
garchfit.fGARCH.AVGARCH[[1]]@fit$coef
garchfit.fGARCH.AVGARCH[[1]]@fit$se.coef
```

Table 1.2: Model selection according to AIC

	SGARCH	IGARCH	EWMA	EGARCH	GJRGARCH	NAGARCH	TGARCH	AVGARCH
norm	2.995	2.998	3.034	2.962	2.967	2.955	2.957	2.954
std	2.924	2.924	2.935	2.900	2.904	2.897	2.896	2.896
sstd	2.920	2.920	2.930	2.895	2.900	2.891	2.891	2.890
ged	2.930	2.930	2.944	2.907	2.911	2.903	7.705	7.702
sged	2.927	2.927	2.940	2.902	2.906	2.898	7.675	7.672

Notes
[1] This table shows the AIC value for the respective model

```
# VaR table, unconditional coverage
# VaRTest(Egarch)
```

1.3 Results of GARCH with time-varying higher moments

```
require(racd)
require(rugarch)
require(parallel)
require(xts)
# ACD specification
sGARCH_ACDspec = acdspec(mean.model = list(armaOrder = c(1, 0)), variance.model = l
distribution.model = list(model = 'jsu', skewOrder = c(1, 1, 1), shapeOrder = c(1,1


# sGARCH
cl = makePSOCKcluster(10)
fit = acdfit(sGARCH_ACDspec, as.data.frame(R), solver = 'msoptim', solver.control =
```

```
# par(mfrow = c(2, 2), mai = c(0.75, 0.75, 0.3, 0.3))
# cm <- plot.zoo(xts(fit@model$modeldata$data, fit@model$modeldata$index), auto.g
# cm <- lines(fitted(fit), col = 2)
# cm
# cs <- plot(xts(abs(fit@model$modeldata$data),fit@model$modeldata$index), auto.g
```

```
# minor.ticks = FALSE, main = 'Conditional Sigma', yaxis.right = F,col = 'grey')
# cs <- lines(sigma(fit), col = 'steelblue')
# cs
# plot(racd::skewness(fit), col = 'steelblue',yaxis.right = F, main = 'Conditiona
# plot(racd::kurtosis(fit), col = 'steelblue', yaxis.right = F,main = 'Conditiona

# pnl <- function(fitted(fit),xts(fit@model$modeldata$data, fit@model$modeldata$i
#   panel.number <- parent.frame()$panel.number
#   if (panel.number == 1) lines(fitted(fit), xts(fit@model$modeldata$data, fit@m
#   lines(fitted(fit),xts(fit@model$modeldata$data, fit@model$modeldata$index), c
# }
# plot(xts(fit@model$modeldata$data, fit@model$modeldata$index), auto.grid = T,mi
# # lines(fitted(fit), col = 2) + grid()
#
# plot(xts(fit@model$modeldata$data, fit@model$modeldata$index), auto.grid = T,mi
```