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.

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.

```
## Warning in sqrt(diag(varcov)): NaNs produced
```

```
## Warning in sqrt(diag(varcov)): NaNs produced
```

```
## Warning in sqrt(diag(varcov)): NaNs produced
## Warning in matrix(value, n, p): data length [6] is not a sub-multiple or
## multiple of the number of columns [5]
```

```
## Warning in matrix(value, n, p): data length [6] is not a sub-multiple or
## multiple of the number of columns [5]
```

```
## Warning in matrix(value, n, p): data length [6] is not a sub-multiple or
## multiple of the number of columns [5]
```

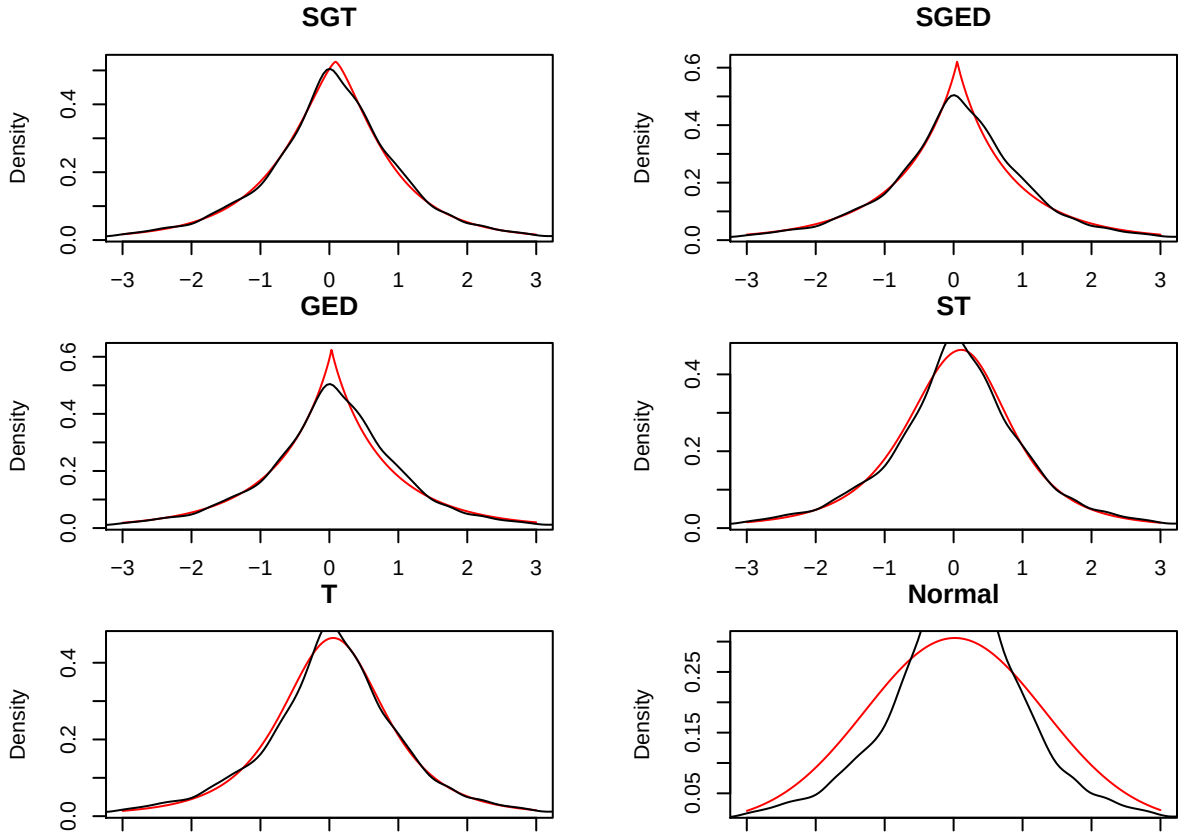

Table 1.1: Maximum likelihood estimates of unconditional distribution functions

	μ	σ	λ	p	q	ν	L	A
SGT	0.02	1.321	-0.04	1.381	3.317		-13973.01	27956
	(0.013)	(0.026)**	(0.012)**	(0.071)**	(0.534)**			
SGED	0.02	1.274	-0.018	0.918	Inf		-14008.18	27956
	(0.01)	(0.016)**	(0.008)*	(0.016)**				
GED	0.032	1.276	0	0.913	Inf		-14009.09	28028
	(0.005)**	(0.016)**		(0.016)**				
ST	0.019	1.487	0.949			2.785	-13997.35	28002
	(0.014)**	(0.056)**	(0.013)**			(0.1)**		
T	0.056	1.494				2.765	-14005.14	28016
	(0.01)**	(0.056)**				(0.097)**		
Normal	0.017	1.304	0	2	Inf		-15093.32	30196
	(0.014)	(0.015)**						

Table contains parameter estimates for SGT-distribution and some of its limiting cases. The underl
data is the daily return series of the Euro Stoxx 50 for the period between December 31. 1986
April 27. 2021. Standard errors are reported between brackets. L is the maximum log-likelihood v
*, ** point out significance at 5

1.2 Results of GARCH with constant higher moments

```
table3 <- matrix(nrow = 12, ncol = 5)
colnames(table3) <- distributions

#trying a loop, maybe you can solve that @filippo?
## column loop i = normal distribution, std, sstd, ged, sged
table3[1,1] <- garchfit.sGARCH[[1]]@fit$coef[1] #first parameter estimate
table3[2,1] <- garchfit.sGARCH[[1]]@fit$se.coef[1] #first standard error
table3[3,1] <- garchfit.sGARCH[[1]]@fit$coef[2] #second parameter estimate
table3[4,1] <- garchfit.sGARCH[[1]]@fit$se.coef[2]




#...
table3 <- round(table3, 3)

# for (i in length(distributions)) {
#   for (j in nrow(table3)) {
#     table3[j,i] <- garchfit.sGARCH[[i]]@fit$coef
#     table3[j+1,i] <-garchfit.sGARCH[[i]]@fit$se.coef
#     }
# }

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

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

```
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)
```

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

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

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

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

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

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 =

# plotxts comes from implementing https://stackoverflow.com/a/50051183/271616
# 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
```