# Appendix -- Rscripts

1. Heat sum calculation from temperature data, trend analysis of the heat sum
2. Precipitation sum, trend analysis of the precipitation sum
3. Maximum SWE, trend analysis of the maximum SWE
4. Calculation of length of continuous snow cover, onset/end date of the CSC from snow depth, trend analysis
5. Correlation analysis
6. Correlation analysis between simulation and observation data
7. Raster analysis (due to the capacity of the laptop, the first part of the analysis (gray scaled) is not done by the author)

## 1. Heat sum from temperature data, trend analysis of the heat sum

```
setwd("C://Users//Broker//Desktop//Filefjell//Temperature")      # set working directory

Sys.setenv(TZ="GMT")      # set time zone and random processes

set.seed(3)  # for reproducible random processes

#Load packages

library(dplyr)

library(zoo)

library(data.table)

#Load data

Temps <- read.table("Filefjell-tm.txt", # adjust the file name

                header = T,  # tells R that the data have a header

                sep = "\t",   # tells R that the data are tab-delimited

                dec = ".",     # tells R that the decimal sign is a dot

                row.names = 1)  # tells R that the first column contains row names

Temps[Temps==65535] <- NA   #Tell R about NA values in the data

# Get time stamp out of the row names # if year is given as yy use %y,   # if given as yyyy use %Y

TimeS <- as.POSIXct(rownames(Temps),  "%d.%m.%Y",   tz="UTC")

Temps.ts <- zoo(Temps, TimeS)

# Append various time indices to data:

# - Julian days

tmp1 <- as.POSIXlt(index(Temps.ts), format="%Y-%m-%d")

Temps.ts$julday <- tmp1$yday+1

# - month

tmp2 <- as.numeric(format(index(Temps.ts), "%m"))
```

```
Temps.ts$month <- tmp2

# - season (DJF, MAM, ...)

zw <- as.data.frame(coredata(Temps.ts))

for (i in 1:length(zw$month)) {

  if (zw$month[i] > 2 && zw$month[i] < 6){

    zw$seas[i] = 2

  } else if (zw$month[i] > 5 && zw$month[i] < 9){

    zw$seas[i] = 3

  } else if (zw$month[i] > 8 && zw$month[i] < 12){

    zw$seas[i] =4

  } else

    zw$seas[i] =1

}

Temps.ts <- zoo(zw, TimeS)

# - year

tmp4 <- as.numeric(format(index(Temps.ts), "%Y"))

Temps.ts$year <- tmp4

# Aggregate times series to monthly data, in this case: sums

sum0.na <- function(x){

  zw <- x[x>0]       # select just those temperatures above 0°C

  sum(zw, na.rm=TRUE)    # calculate the sum of all temperatures above 0°C

}

# adjust the sum function to  i) sum only those elements above zero, ii) remove NA values from calculation

# and apply it using aggregate()

Temps.monthly <- aggregate(Temps.ts,   # the time series to be aggregated

                           as.yearmon,    # how to aggregate: by month

                           sum0.na)    # how to aggregate

Heatsums.monthly <- Temps.monthly[,1:2]

# Recreate the different time indices, but adjusted to our monthly data:

# month-index

tmp3 <- as.numeric(format(index(Temps.monthly), "%m"))

Heatsums.monthly$month <- tmp3

# year index

tmp4 <- as.numeric(format(index(Temps.monthly), "%Y"))
```

```
Heatsums.monthly$year <- tmp4

# a fake year index: aggregate heat sums over a longer period, September to May

# adjust the year index in a sense that our September to May period falls into one year:

tmp5 <- as.numeric(format(index(Temps.monthly)+4/12, "%Y"))

Heatsums.monthly$fakeyear <- tmp5

# create a column stating whether we are inside the September to May

Heatsums.monthly$SepMay <- ifelse(Heatsums.monthly$month>5&Heatsums.monthly$month<9,0,1)

# remove the column which temporarily retained

Heatsums.monthly <- Heatsums.monthly[,-2]

# finally, set meaningful column names

names(Heatsums.monthly) <- c("heatsum_0", "month", "year", "fakeyear", "SepMay")


### Heat sum for the period September to May:

# Grab the table (i.e., the coredata) with the monthly heat sums from the time series object

HeatSum.SepMay <- data.frame(coredata(Heatsums.monthly))

SepMayHS <- data.frame (Winter  = seq(1957,2022,1),      # create a dataframe within which to store the results

                    HSSepMay = rep(0, length(seq(1957,2022,1))))

# Create an auxiliary index for the years from 1957 to 2023

ind <- seq(1957,2023,1)

# use a for loop to go (fake)year wise through the data, telling R  what to do for each fake year

for (i in 1:length(seq(1957,2022,1))){      # for each fake year from 1957 to 2022, we tell R to

  zw <- HeatSum.SepMay[HeatSum.SepMay$fakeyear==ind[i],]      # subset data by that fake year

  zw2 <- zw[zw$SepMay==1,]        # to subset the subset by the September to May period

  SepMayHS[i,2] <- sum(zw2$heatsum_0)      # to sum up the heat sums during this period and to write the result into the data frame

}

# winters with incomplete data, i.e. the winter 1956/57 and 2022/23

SepMayHS <- SepMayHS[-c(1, length(seq(1957,2023,1))),]


##### Trend analysis of the heat sum

#Load package

library(greenbrown)

# Transform the data to be analyzed to a ts object

HSSepMay.ts <- ts(data = SepMayHS$HSSepMay, start = 1957, end = 2022, frequency = 1)
```

```
HSSepMay.trd <- Trend(HSSepMay.ts, method="AAT")  # do trend analysis

HSSepMay.trd   # check results

plot(HSSepMay.trd, ylab="Heat sum")   # plot results
```

**2. Precipitation sum (Sep-May), trend analysis of the precipitation sum**

```
setwd("C://Users//Broker//Desktop//Filefjell//Precipitation")  # Set working directory

Sys.setenv(TZ="GMT")  # set time zone and random processes

set.seed(3)  # for reproducible random processes

#Load packages

library(dplyr)

library(zoo)

library(data.table)

#Load data

Precip <- read.table("Filefjell-pre.txt", header = T, sep = "\t", dec = ".",  row.names = 1)

Precip[Precip==65535] <- NA      #Tell R about NA values in the data

TimeS <- as.POSIXct(rownames(Precip),  "%d.%m.%Y",  tz="UTC")# Get time stamp out of the row names

Precip.ts <- zoo(Precip, TimeS)

# - Julian days

pre1 <- as.POSIXlt(index(Precip.ts), format="%Y-%m-%d")

Precip.ts$julday <- pre1$yday+1

# - month

pre2 <- as.numeric(format(index(Precip.ts), "%m"))

Precip.ts$month <- pre2

# - season (DJF, MAM, ...)

zw <- as.data.frame(coredata(Precip.ts))

for (i in 1:length(zw$month)) {

  if (zw$month[i] > 2 && zw$month[i] < 6){

    zw$seas[i] = 2

  } else if (zw$month[i] > 5 && zw$month[i] < 9){

    zw$seas[i] = 3

  } else if (zw$month[i] > 8 && zw$month[i] < 12){

    zw$seas[i] =4

  } else

    zw$seas[i] =1
```

```
}
Precip.ts <- zoo(zw, TimeS)
# - year
pre4 <- as.numeric(format(index(Precip.ts), "%Y"))
Precip.ts$year <- pre4
# Aggregate times series to monthly data, in this case: sums
# script the function to be used first
sum.na <- function(x){
  sum(x, na.rm=TRUE)
}
# remove NA values from calculation
# and apply it using aggregate()
Precip.monthly <- aggregate(Precip.ts,  as.yearmon,  sum.na)
Precipsums.monthly <- Precip.monthly[,1:2]
# month-index
pre3 <- as.numeric(format(index(Precip.monthly), "%m"))
Precipsums.monthly$month <- pre3
# year index
pre4 <- as.numeric(format(index(Precip.monthly), "%Y"))
Precipsums.monthly$year <- pre4
pre5 <- as.numeric(format(index(Precip.monthly)+4/12, "%Y"))
Precipsums.monthly$fakeyear <- pre5
# Create a column stating whether we are inside the September to May
# period (used for later aggregation):
Precipsums.monthly$SepMay <- ifelse(Precipsums.monthly$month>5&Precipsums.monthly$month<9,0,1)
# Remove the column which temporarily retained
Precipsums.monthly <- Precipsums.monthly[,-2]
# set meaningful column names
names(Precipsums.monthly) <- c("precipsum", "month", "year", "fakeyear", "SepMay")


## Total precipitation Sep-May################################
PrecipSum.SepMay <- data.frame(coredata(Precipsums.monthly))
# Create a dataframe within which to store the results
SepMayPR <- data.frame (Winter  = seq(1957,2022,1), PRSepMay = rep(0, length(seq(1957,2022,1))))
```

```
# Create an auxiliary index for the years from 1957 to 2023

ind <- seq(1957,2023,1)

for (i in 1:length(seq(1957,2023,1))){ # for each fake year from 1957 to 2023, we tell R to

  zw <- PrecipSum.SepMay[PrecipSum.SepMay$fakeyear==ind[i],] # subset data by that fake year

  zw2 <- zw[zw$SepMay==1,] # to subset the subset by the September to May period

  SepMayPR[i,2] <- sum(zw2$precipsum) # to sum up the precipitation during this period

  # and to write the result into the data frame

}

# winters with incomplete data, i.e. the winter 1956/57 and 2022/23

SepMayPR <- SepMayPR[-c(1, length(seq(1957,2023,1))),]


### Trend analysis of the precipitation sum

library(greenbrown)

# transform the data to be analyzed to a ts object

SepMayPR.ts <- ts(data = SepMayPR$PRSepMay,start = 1957, end = 2022, frequency = 1)

SepMayPR.trd <- Trend(SepMayPR.ts, method="AAT")

SepMayPR.trd

plot(SepMayPR.trd, ylab="Precipitation sums September-May")
```

**3. Maximum SWE, trend analysis of the maximum SWE**

```
setwd("C://Users//Broker//Desktop//Filefjell//SWE")    # set working directory

Sys.setenv(TZ="GMT")

set.seed(3)

#Load packages

library(dplyr)

library(zoo)

library(data.table)

# Load data

Swe.sim <- read.table("Filefjell-swe.txt",  header = T,  sep = "\t",  dec = ".",  row.names = 1)

Swe.sim[Swe.sim==65535] <- NA     #Tell R about NA values in the data

# Check data range

min(Swe.sim, na.rm = TRUE)

max(Swe.sim, na.rm = TRUE)

# Get time stamp out of the row names
```

```r
TimeS <- as.POSIXct(rownames(Swe.sim), "%d.%m.%Y", tz="UTC")
# define data as time series (zoo) object
Swe.sim.ts <- zoo(Swe.sim, TimeS)
# append various time indices to data
# - Julian days
swesim1 <- as.POSIXlt(index(Swe.sim.ts), format="%Y-%m-%d")
Swe.sim.ts$julday <- swesim1$yday+1
# - month
swesim2 <- as.numeric(format(index(Swe.sim.ts), "%m"))
Swe.sim.ts$month <- swesim2
# - season (DJF, MAM, ...)
zw <- as.data.frame(coredata(Swe.sim.ts))
for (i in 1:length(zw$month)) {
  if (zw$month[i] > 2 && zw$month[i] < 6){
    zw$seas[i] = 2
  } else if (zw$month[i] > 5 && zw$month[i] < 9){
    zw$seas[i] = 3
  } else if (zw$month[i] > 8 && zw$month[i] < 12){
    zw$seas[i] =4
  } else
    zw$seas[i] =1
}
Swe.sim.ts <- zoo(zw, TimeS)
# - year
swesim4 <- as.numeric(format(index(Swe.sim.ts), "%Y"))
Swe.sim.ts$year <- swesim4

### Aggregate to annual data
# define a function to aggregate over a year
max.na <- function(x) max(x, na.rm=TRUE)    # adjust the max function to exclude
toyear <- function(x) as.integer(as.yearmon(x))
Swesim.annual <- aggregate(Swe.sim.ts, toyear, max.na) # apply it

# Trend analysis
```

```
library(greenbrown)

# transform the data to be analyzed to a ts object

Swesim.annual.ts <- ts(data = Swesim.annual[-c(1,dim(Swesim.annual)[1]),1],start = 1957, end = 2022, frequency = 1)

Swesim.annual.trd <- Trend(Swesim.annual.ts, method="AAT")

Swesim.annual.trd

plot(Swesim.annual.trd,  ylab="Annual maximum SWE [mm]")
```

**4. Calculation of length of continuous snow cover, onset/end date of the CSC from snow depth**

```
setwd("C://Users//Broker//Desktop//Filefjell//Snowdepth")   #Set working directory

#Activate the packages

library(zoo)

library(dplyr)

library(lubridate)

#Load data

SD.Filefjell <- read.table("Filefjell-sd.txt", header=T)

SD.Filefjell[SD.Filefjell==65535] <- NA     #Remove NA values

x.date <- as.Date(SD.Filefjell$Date, format="%d.%m.%Y")      #Tell R that "Date" column has the time information

#Create a time series in R

SD.Filefjell.ts <- zoo(SD.Filefjell$sd, x.date)

#Continuous snow cover

startLoc=0 # create a variable to put the start date into

count=0 # create a variable to count days with snow cover

contSC=rep(0, length(SD.Filefjell$sd)) # create a storage vector

# loop through the snowdepth vector

for(i in 1:length(SD.Filefjell$sd)){

  p <- SD.Filefjell$sd[i] # get the current value

  if(count==0 & p>0){ # if the previous day was no snow cover and today is

    startLoc <- i # set the starting of the snow cover

    count <- count+1 # add one to the count

  }else if(count!=0 & p>0){ # if the snow cover is continuing

    count <- count+1 # add one to the count

  }else{ # no snow cover

    contSC[startLoc] <- count # stick the count into the starting day

    count <- 0 # reset the counter
```

```
    startLoc <- i # increment the startLoc to prevent overwriting
  }
}
# and add the resulting number of continuous days with snow cover to the original data frame
SD.Filefjell$SC <- contSC


### Extract just dates and length of continuous snow cover #####################
sc.dates <- SD.Filefjell$Date[which(SD.Filefjell$SC>0)]
sc.length <- SD.Filefjell$SC[which(SD.Filefjell$SC>0)]
sc.Filefjell <- data.frame(cbind(sc.dates,sc.length))
# and transform them into a time series object
x.date2 <- as.Date(sc.dates, format="%d.%m.%Y")
SC.ts <- zoo(sc.length, x.date2)
### Access maximum length of continuous snow cover per year (per winter)
contSC.Filefjell <- aggregate(SC.ts, cut(time(SC.ts),"y"), max)
# plot results as a barplot
barplot(contSC.Filefjell)


### Access number of days with snow cover per year
lengthSC.Filefjell <- aggregate(SC.ts, cut(time(SC.ts),"y"), sum)
# remove last winter, as data are incomplete
lengthSC.Filefjell <- lengthSC.Filefjell[-length(lengthSC.Filefjell)] # indexing (using []) the data, telling R to remove (-) the last entry.


## Trend analysis
library(greenbrown)
lengthSC.Filefjell.ts <- ts(coredata(lengthSC.Filefjell)[-1],start = 1957, end = 2023, frequency = 1)
lengthSC.Filefjell.trd <- Trend(lengthSC.Filefjell.ts)
lengthSC.Filefjell.trd
plot(lengthSC.Filefjell.trd, ylab="Number of days with CSC")


### Access onset date of continuous snow cover ###################################
sc.Filefjell$year<-as.factor(cut(x.date2,"y"))
On.ContSC <- sc.Filefjell %>%
```

```
  group_by(year) %>%
  slice(which.max(sc.length)) %>% # takes just the period of longest snow cover
  ungroup()
On.ContSC <- as.data.frame(On.ContSC)
# convert date to day of the year
Onset.CSC <- yday(as.POSIXlt(On.ContSC[,1], format="%d.%m.%Y"))
## Trend analysis of the onset of CSC
library(greenbrown)
Onset.CSC.ts <- ts(data = Onset.CSC, start=1957,end=2022,frequency=1)
Onset.CSC.trd <- Trend(Onset.CSC.ts, method="AAT")
Onset.CSC.trd
plot(Onset.CSC.trd, ylab="Onset of CSC")
### Testing for temporal autocorrelation (in other words, independence of the data)
acf(Onset.CSC)

### Getting the end date of CSC ################################################
End.ContSC <- as.Date(On.ContSC[,1], format="%d.%m.%Y") + as.numeric(On.ContSC[,2]) - 1
head(End.ContSC)
End.ContSC <- as.data.frame(End.ContSC)
# convert date to day of the year
End.CSC <- yday(as.POSIXlt(End.ContSC[,1], format="%d.%m.%Y"))
## Trend analysis of the end date of CSC
library(greenbrown)
End.CSC.ts <- ts(data = End.CSC, start=1957,end=2022,frequency=1)
End.CSC.trd <- Trend(End.CSC.ts, method="AAT")
End.CSC.trd
plot(End.CSC.trd, ylab="End date of CSC")
```

**5. Correlation analysis**

```
#Heat sum and precipitation
cor.test(SepMayHS$HSSepMay,SepMayPR$PRSepMay)
#Heat sum and maximum SWE
cor.test(Swesim.annual$swe.mm.[-c(1,dim(Swesim.annual)[1]),1],start = 1957, end = 2022, frequency = 1,
SepMayHS$HSSepMay)
```

```
#Maximum SWE and precipitation

cor.test(Swesim.annual$swe.mm.[-c(1,dim(Swesim.annual)[1]),1],start = 1957, end = 2022, frequency = 1,
SepMayPR$PRSepMay)

#Heat sum and the length of CSC

cor.test(SepMayHS$HSSepMay,lengthSC.Filefjell[-1])

#Precipitaiton and the length of CSC
cor.test(SepMayPR$PRSepMay,lengthSC.Filefjell[-1])

#Maximum SWE and the length of CSC

cor.test(Swesim.annual$swe.mm.[-c(1,dim(Swesim.annual)[1]),1],start = 1957, end = 2022, frequency = 1,
lengthSC.Filefjell[-1])
```

**6. Correlation analysis between simulation data and observation data**

```
### Simulation and observation data ##################################################

# Simulation data -maximum snow depth

setwd("C://Users//Broker//Desktop//Obs vs sim//Geilo")    # set working directory

Sys.setenv(TZ="GMT")

set.seed(3)

#Load packages

library(dplyr)

library(zoo)

library(data.table)

#Load data

Sd.sim <- read.table("sd-sim.txt", header = T,  sep = "\t",  dec = ".",  row.names = 1)

Sd.sim[Sd.sim==65535] <- NA       #Tell R about NA values in the data

# Check data range

min(Sd.sim, na.rm = TRUE)

max(Sd.sim, na.rm = TRUE)

# Get time stamp out of the row names

TimeS <- as.POSIXct(rownames(Sd.sim), "%d.%m.%Y", tz="UTC")

# define data as time series (zoo) object - this makes data aggregation into mean values more handy

Sd.sim.ts <- zoo(Sd.sim, TimeS)

# append various time indices to data, which might be useful later:

# - Julian days

sdsim1 <- as.POSIXlt(index(Sd.sim.ts), format="%Y-%m-%d")

Sd.sim.ts$julday <- sdsim1$yday+1
```

```
# - month
sdsim2 <- as.numeric(format(index(Sd.sim.ts), "%m"))
Sd.sim.ts$month <- sdsim2
# - season (DJF, MAM, ...)
zw <- as.data.frame(coredata(Sd.sim.ts))
for (i in 1:length(zw$month)) {
  if (zw$month[i] > 2 && zw$month[i] < 6){
    zw$seas[i] = 2
  } else if (zw$month[i] > 5 && zw$month[i] < 9){
    zw$seas[i] = 3
  } else if (zw$month[i] > 8 && zw$month[i] < 12){
    zw$seas[i] =4
  } else
    zw$seas[i] =1
}
Sd.sim.ts <- zoo(zw, TimeS)
# - year
sdsim4 <- as.numeric(format(index(Sd.sim.ts), "%Y"))
Sd.sim.ts$year <- sdsim4

### Aggregate to annual data
max.na <- function(x) max(x, na.rm=TRUE) # adjust the max function to exclude
toyear <- function(x) as.integer(as.yearmon(x)) # define a function to aggregate over a year,
Sdsim.annual <- aggregate(Sd.sim.ts, toyear, max.na) # apply it

## Station data (observation) -maximum snow depth
#Load data
Sd.obs <- read.table("sd-fixed.txt", header = T, sep = "\t", dec = ".", row.names = 1)
# Check data range
min(Sd.obs, na.rm = TRUE)
max(Sd.obs, na.rm = TRUE)
# Get time stamp out of the row names
TimeS <- as.POSIXct(rownames(Sd.obs),
                "%d.%m.%Y", # if year is given as yy use %y,
```

```
                # if given as yyyy use %Y)
                tz="UTC")
# define data as time series (zoo) object
Sd.obs.ts <- zoo(Sd.obs, TimeS)
# append various time indices to data
# - Julian days
sdobs1 <- as.POSIXlt(index(Sd.obs.ts), format="%Y-%m-%d")
Sd.obs.ts$julday <- sdobs1$yday+1
# - month
sdobs2 <- as.numeric(format(index(Sd.obs.ts), "%m"))
Sd.obs.ts$month <- sdobs2
# - season (DJF, MAM, ...)
zw <- as.data.frame(coredata(Sd.obs.ts))
for (i in 1:length(zw$month)) {
  if (zw$month[i] > 2 && zw$month[i] < 6){
    zw$seas[i] = 2
  } else if (zw$month[i] > 5 && zw$month[i] < 9){
    zw$seas[i] = 3
  } else if (zw$month[i] > 8 && zw$month[i] < 12){
    zw$seas[i] =4
  } else
    zw$seas[i] =1
}
Sd.obs.ts <- zoo(zw, TimeS)
# - year
sdobs4 <- as.numeric(format(index(Sd.obs.ts), "%Y"))
Sd.obs.ts$year <- sdobs4

### Aggregate to annual data
max.na <- function(x) max(x, na.rm=TRUE) # adjust the max function to exclude
toyear <- function(x) as.integer(as.yearmon(x))
Sdobs.annual <- aggregate(Sd.obs.ts, toyear, max.na)

### correlation between simulation and observation
```

```
cor.test(Sdsim.annual$sd, Sdobs.annual$sd)


## Maximum SWE data (simulation) – to compare to modeled snow depth data
# Load data
Swe.sim <- read.table("swe-sim.txt", header = T, sep = "\t", dec = ".", row.names = 1)
Swe.sim[Swe.sim==65535] <- NA)  #Tell R about NA values in the data
# Check data range
min(Swe.sim, na.rm = TRUE)
max(Swe.sim, na.rm = TRUE)
TimeS <- as.POSIXct(rownames(Swe.sim),  "%d.%m.%Y",  tz="UTC")  # Get time stamp out of the row names
Swe.sim.ts <- zoo(Swe.sim, TimeS)  # define data as time series (zoo) object
# append various time indices to data, which might be useful later:
# - Julian days
swesim1 <- as.POSIXlt(index(Swe.sim.ts), format="%Y-%m-%d")
Swe.sim.ts$julday <- swesim1$yday+1
# - month
swesim2 <- as.numeric(format(index(Swe.sim.ts), "%m"))
Swe.sim.ts$month <- swesim2
# - season (DJF, MAM, ...)
zw <- as.data.frame(coredata(Swe.sim.ts))
for (i in 1:length(zw$month)) {
  if (zw$month[i] > 2 && zw$month[i] < 6){
    zw$seas[i] = 2
  } else if (zw$month[i] > 5 && zw$month[i] < 9){
    zw$seas[i] = 3
  } else if (zw$month[i] > 8 && zw$month[i] < 12){
    zw$seas[i] =4
  } else
    zw$seas[i] =1
}
Swe.sim.ts <- zoo(zw, TimeS)
# - year
swesim4 <- as.numeric(format(index(Swe.sim.ts), "%Y"))
Swe.sim.ts$year <- swesim4
```

### Aggregate to annual data

max.na <- function(x) max(x, na.rm=TRUE)

toyear <- function(x) as.integer(as.yearmon(x))

Swesim.annual <- aggregate(Swe.sim.ts, toyear, max.na)

#### Correlation between SWE and SD (simulation)

cor.test(Sdsim.annual$sd,Swesim.annual$swe.mm.)

**7. Raster analysis**

## seNorge data: raster-based analysis of trends over time

install.packages("raster")

library(raster)

library(greenbrown)

### Trends in maximum SWE ##########################################################

setwd("C://Users//Broker//Desktop//master thesis//seNorge-SWE")

### Data are downloaded as *.nc files, one for each year. Within each nc-file, daily data are stored as layers.

The files cover entire Norway. The following lines of script deal with combining and clipping the data to a certain area of interest.

### List all *.nc files in the working directory

# nc <- list.files(pattern = ".nc", full.names = T, recursive = T)

### stack all files together into a huge raster stack

# SWE <- stack(nc)

### set the extent of the AOI (in UTM zone 33 coordinates), covering most of southern Norway from the western coast to Sweden in the east

# ext <- extent(matrix(c(-70000, 6550000,390000, 6960000), nrow=2))

### Crop the data to the extent of the AOI

# SWEcrop <- crop(SWE, ext)

### Export the cropped data to disk

# writeRaster(SWEcrop, "SWEcropped.tif")

### Read-in SWE data

SWEcrop2 <- brick("SWEcropped.tif")

### Calculate trend over time based on the annual maximum of the SWE

SWEmax.trd <- TrendRaster(SWEcrop2, start=1991, freq=365, breaks=0, h=10, funAnnual=max)

```
### and have a look at the trends

plot(SWEmax.trd)

### Extract trend and its significance for segment 1

### extract the raster showing the p values of the trend for each cell

p1 <- SWEmax.trd$PvalSEG1

### Create a mask of all values >0.05 to get a confidence level of 95%

m = c(0, 0.05, 1, 0.05, 1, 0)

rclmat = matrix(m, ncol=3, byrow=TRUE)

p.mask1 = reclassify(p1, rclmat)

fun=function(x) { x[x<1] <- NA; return(x)}

p.mask1.NA = calc(p.mask1, fun)

### mask all insignificant values in the trend map and only retain SWEmax changes significant at the 95% level.

SWEmax.trd1.sig <- mask(SWEmax.trd$SlopeSEG1, p.mask1.NA)

### Export this layer for use in GIS:

writeRaster(SWEmax.trd1.sig, "SWEmax1_sig_trends.tif")

SWETrends <- raster("SWEmax1_sig_trends.tif")

# Reclassify the values into 7 groups:

# >20        pronounced increase

# 10 – 20  moderate increase

# 2.5 – 10 slight increase

# -2.5 – 2.5 very slight change

# -10 - -2.5 slight decrease

# -20 - -10 moderate decrease

# <-20 pronounced decrease

# all values >= -100 and <= -20 become 1

m <- c(-100, -20, 1,  -20, -10, 2,  -10, -2.5, 3, -2.5, 2.5, 4, 2.5, 10, 5, 10, 20, 6, 20, 100, 7)

rclmat <- matrix(m, ncol=3, byrow=TRUE)

SWETrends.rc <- reclassify(SWETrends, rclmat)

# export the reclassified raster as a tif file for use in QGIS

writeRaster(SWETrends.rc, "SWEmax_sig_trends_rc.tif")
```