Example SmartSMEAR API calls, see https://avaa.tdata.fi/web/smart/smear/api

```
urlstring<-"https://avaa.tdata.fi/smear-services/smeardata.jsp?variables=Pamb0,UV_B&table=HYY_META&from=

urlstring2<-"https://avaa.tdata.fi/smear-services/smeardata.jsp?variables=Pamb0,UV_B&table=HYY_META&fro
```

The most simple way for retrieving data via SmartSMEAR API is using read.csv which converts the data stream into data frame. It works in base R without additional libraries:

```
data<-read.csv(urlstring)

data2<-read.csv(urlstring2)

class(data)
```

```
## [1] "data.frame"
```

```
data
```

```
##    Year Month Day Hour Minute Second HYY_META.Pamb0 HYY_META.UV_B
## 1  2016     4  11    0      0      0        1001.94        -4e-04
## 2  2016     4  11    0      1      0        1001.95        -4e-04
## 3  2016     4  11    0      2      0        1002.02        -4e-04
## 4  2016     4  11    0      3      0        1002.05        -4e-04
## 5  2016     4  11    0      4      0        1002.04         4e-04
## 6  2016     4  11    0      5      0        1002.15         4e-04
## 7  2016     4  11    0      6      0        1002.05         4e-04
## 8  2016     4  11    0      7      0        1002.07         4e-04
## 9  2016     4  11    0      8      0        1002.12         4e-04
## 10 2016     4  11    0      9      0        1002.18         4e-04
```

```
data2
```

```
##   Year Month Day Hour Minute Second HYY_META.Pamb0 HYY_META.UV_B
## 1 2016     4  11    0     11      0        1002.26        -4e-04
## 2 2016     4  11    0     12      0        1002.22        -8e-04
## 3 2016     4  11    0     13      0        1002.07         1e-04
## 4 2016     4  11    0     14      0        1002.15         1e-04
## 5 2016     4  11    0     15      0        1002.21         1e-04
## 6 2016     4  11    0     16      0        1002.13        -4e-04
## 7 2016     4  11    0     17      0        1002.12        -4e-04
## 8 2016     4  11    0     18      0        1002.18         4e-04
## 9 2016     4  11    0     19      0        1002.12         1e-04
```

```
data
```

```
##    Year Month Day Hour Minute Second HYY_META.Pamb0 HYY_META.UV_B
## 1  2016     4  11    0      0      0        1001.94        -4e-04
## 2  2016     4  11    0      1      0        1001.95        -4e-04
## 3  2016     4  11    0      2      0        1002.02        -4e-04
## 4  2016     4  11    0      3      0        1002.05        -4e-04
## 5  2016     4  11    0      4      0        1002.04         4e-04
## 6  2016     4  11    0      5      0        1002.15         4e-04
## 7  2016     4  11    0      6      0        1002.05         4e-04
## 8  2016     4  11    0      7      0        1002.07         4e-04
## 9  2016     4  11    0      8      0        1002.12         4e-04
## 10 2016     4  11    0      9      0        1002.18         4e-04
```

Convert datetime columns to more convenient data type:

```
data$datetim<-with(data,ISOdatetime(Year,Month,Day,Hour,Minute,Second))

data$datetim
```

```
##  [1] "2016-04-11 00:00:00 EEST" "2016-04-11 00:01:00 EEST"
##  [3] "2016-04-11 00:02:00 EEST" "2016-04-11 00:03:00 EEST"
##  [5] "2016-04-11 00:04:00 EEST" "2016-04-11 00:05:00 EEST"
##  [7] "2016-04-11 00:06:00 EEST" "2016-04-11 00:07:00 EEST"
##  [9] "2016-04-11 00:08:00 EEST" "2016-04-11 00:09:00 EEST"
```

```
data$datetim+86400
```

```
##  [1] "2016-04-12 00:00:00 EEST" "2016-04-12 00:01:00 EEST"
##  [3] "2016-04-12 00:02:00 EEST" "2016-04-12 00:03:00 EEST"
##  [5] "2016-04-12 00:04:00 EEST" "2016-04-12 00:05:00 EEST"
##  [7] "2016-04-12 00:06:00 EEST" "2016-04-12 00:07:00 EEST"
##  [9] "2016-04-12 00:08:00 EEST" "2016-04-12 00:09:00 EEST"
```

API makes your life easier when doing dynamic data retrievals within data processing/analysis scripts.

Construct times in POSIX time (seconds):

```
time2<-Sys.time()
format(time2,"%Y-%m-%d %H:%M:%S")
```

```
## [1] "2020-03-09 10:33:12"
```

```
time2<-as.POSIXct("2018-10-29 12:00:00")
format(time2,"%Y-%m-%d %H:%M:%S")
```

```
## [1] "2018-10-29 12:00:00"
```

```
time1<-time2-3600
format(time1,"%Y-%m-%d %H:%M:%S")
```

```
## [1] "2018-10-29 11:00:00"
```

Paste the times into API call for retrieving piece of data collected 24 h ago:

```
time2<-Sys.time()-86400
timestr2<-format(time2,"%Y-%m-%d%%20%H:%M:%S")
time1<-time2-3600
timestr1<-format(time1,"%Y-%m-%d%%20%H:%M:%S")

urlstring<-paste("https://avaa.tdata.fi/smear-services/smeardata.jsp?variables=T168,T672&table=HYY_META
    timestr1,"&to=",timestr2,
    "&quality=ANY&averaging=30MIN&type=ARITHMETIC",sep="")

urlstring
```

```
## [1] "https://avaa.tdata.fi/smear-services/smeardata.jsp?variables=T168,T672&table=HYY_META&from=2020
```

```
data<-read.csv(urlstring)

head(data,1)
```

```
##   Year Month Day Hour Minute Second HYY_META.T168 HYY_META.T672
## 1 2020     3   8    9     30      0     -1.631923     -2.500385
```

```
tail(data,1)
```

```
##   Year Month Day Hour Minute Second HYY_META.T168 HYY_META.T672
## 3 2020     3   8   10     30      0       -0.3575       -1.4575
```

Below simple function for constructing API call from given parameters and downloading data. Named parameters are used so the user can give table and variables separately or use table.variable notation, give parameters in any order and skip irrelevant parameters. The function employs read.csv which ignores any http return codes or error messages. Therefore, additional parsing of returned data frame is needed.

Different types of error affect the returned data in different ways. Be careful and take note of the column names of the returned data frame!

```
getSmearData<-function(time1,time2,...,dbtable="",dbvariables=list(),dbtablevariables=list(),
  quality="ANY",averaging="NONE",avgtype="NONE") {

# Simple function for retrieving data from SMEAR database
# No input check, error handling etc.
# time1 and time2 are start and end times as POSIX time.
# Downloaded variables are given as list of table.variable strings (parameter "dbtablevariables").
# or giving table (string "dbtable") and variables (list "dbvariables") separately.
# Results of the query are returned as data frame (also in case of error).

timestr1=as.character(time1,"%Y-%m-%d%%20%H:%M:%S")
timestr2=as.character(time2,"%Y-%m-%d%%20%H:%M:%S")

if(length(dbtablevariables)<1) {
  urlstring<-paste("https://avaa.tdata.fi/smear-services/smeardata.jsp?",
    "variables=",paste(dbvariables,collapse=","),
    "&table=",dbtable,
    "&from=",timestr1,
    "&to=",timestr2,
    "&quality=",quality,"&averaging=",averaging,"&type=",avgtype,sep="")
}
else {
  urlstring<-paste("https://avaa.tdata.fi/smear-services/smeardata.jsp?",
    "tablevariables=",paste(dbtablevariables,collapse=","),
    "&from=",timestr1,
    "&to=",timestr2,
    "&quality=",quality,"&averaging=",averaging,"&type=",avgtype,sep="")
}

writeLines(urlstring)

return(read.csv(urlstring))
}
```

Two examples of using the function:

```
time2<-as.POSIXct("2018-10-27 12:00:00")
time1<-time2-86400
tablename<-"HYY_META"
variables_list<-c("Pamb336","Tmm672")
avg_time="60MIN"
avg_type="Arithmetic"

data1<-getSmearData(time1,time2,dbtable=tablename,dbvariables=variables_list,
    averaging=avg_time,avgtype=avg_type)
```

```
## https://avaa.tdata.fi/smear-services/smeardata.jsp?variables=Pamb336,Tmm672&table=HYY_META&from=2018
```

```
head(data1)
```

```
##   Year Month Day Hour Minute Second HYY_META.Pamb336 HYY_META.Tmm672
## 1 2018    10  26    0      0      0              NaN      -0.4288333
## 2 2018    10  26    1      0      0              NaN      -0.5253333
## 3 2018    10  26    2      0      0              NaN      -0.6213334
## 4 2018    10  26    3      0      0              NaN      -0.7083333
## 5 2018    10  26    4      0      0              NaN      -0.8508334
## 6 2018    10  26    5      0      0              NaN      -0.9761667
```

```
time1<-Sys.Date()
time2<-time1+3600
tablevariables_list=c("HYY_META.Pamb336")

data2<-getSmearData(time1,time2,dbtablevariables=tablevariables_list)
```

```
## https://avaa.tdata.fi/smear-services/smeardata.jsp?tablevariables=HYY_META.Pamb336&from=2020-03-09%20
```

```
head(data2)
```

```
##   Year Month Day Hour Minute Second HYY_META.Pamb336
## 1 2020     3   9    0      0      0          980.640
## 2 2020     3   9    0      1      0          980.665
## 3 2020     3   9    0      2      0          980.670
## 4 2020     3   9    0      3      0          980.680
## 5 2020     3   9    0      4      0          980.660
## 6 2020     3   9    0      5      0          980.605
```

SmartSMEAR API gives http return codes and in most cases also meaningful error messages. Read.csv cannot handle the http codes and also tries to convert the error messages to data frame. Below some examples.

Some dedicated http libraries, for instance RCurl, can handle error messages better.

```
time2<-Sys.time()-86400
time1<-time2-180

writeLines("Error 1:")
```

```
## Error 1:
```

```
data<-getSmearData(time1,time2,dbtable="HYY_META",dbvariables=c("xxxx"))
```

```
## https://avaa.tdata.fi/smear-services/smeardata.jsp?variables=xxxx&table=HYY_META&from=2020-03-08%201
```

```
if(names(data)[1]!="Year" | dim(data)[1]<1) {
  print(data)
}
```

```
## [1] Year   Month  Day    Hour   Minute Second
## <0 rows> (or 0-length row.names)
```

```
writeLines("Error 2:")
```

```
## Error 2:
```

```
data<-getSmearData(time1,time2,dbtable="HYY_XXXX",dbvariables=c("Glob"))
```

```
## https://avaa.tdata.fi/smear-services/smeardata.jsp?variables=Glob&table=HYY_XXXX&from=2020-03-08%201
```

```
if(names(data)[1]!="Year" | dim(data)[1]<1) {
  print(data)
}
```

```
##    Invalid.table.parameter..It.should.be.one.of.
## 1                                        ERO_EDDY
## 2                                        HYY_META
## 3                                      HYY_EDDY233
## 4                                      HYY_EDDY330
## 5                                      HYY_EDDYTOW
## 6                                      HYY_EDDYSUB
## 7                                        HYY_DMPS
## 8                                        HYY_AERO
## 9                                     HYY_EDDYMAST
## 10                                    HYY_TREEFLOW
## 11                                     HYY_AUG_REA
## 12                                       KUM_META
## 13                                       KUM_EDDY
## 14                                       KUM_DMPS
## 15                                      HYY_DMPST
## 16                                       HYY_SLOW
## 17                                       HYY_TREE
## 18                                     VAR_HOURLY
## 19                                       VAR_META
## 20                                       VAR_TREE
## 21                                       VAR_EDDY
## 22                                       VAR_DMPS
## 23                                      SII1_META
## 24                                      SII1_EDDY
## 25                                      SII2_META
## 26                                      SII2_EDDY
## 27                                       TOR_EDDY
## 28                                       KPS_DMPS
## 29                                       KVJ_EDDY
## 30                                       KVJ_META
## 31                                        PUI_cdp
## 32                                   PUI_dmps_int
## 33                                   PUI_dmps_tot
## 34                                   PUI_maap_int
## 35                                   PUI_maap_tot
## 36                                   PUI_neph_int
## 37                                   PUI_neph_tot
## 38                                    PUI_weather
```

Specific notes for AVAA API:

When using tablevariables parameter, if any variable does not exist in given table, no data from that table are returned.

```
time2<-Sys.time()-86400
time1<-time2-180

# All variables exist
data<-getSmearData(time1,time2,dbtablevariables=c("HYY_META.Glob","HYY_META.Glob67","SII1_META.Glob"))
```

```
## https://avaa.tdata.fi/smear-services/smeardata.jsp?tablevariables=HYY_META.Glob,HYY_META.Glob67,SII1
```

```
head(data)
```

```
##   Year Month Day Hour Minute Second HYY_META.Glob HYY_META.Glob67
## 1 2020     3   8   10     31      0      373.1673         262.506
## 2 2020     3   8   10     32      0      375.0043         264.446
## 3 2020     3   8   10     33      0      375.1006         265.249
##   SII1_META.Glob
## 1       328.3710
## 2       330.0683
## 3       332.5690
```

```
# Glob127 does not exist in HYY_META, only data from SII1_META are returned
data<-getSmearData(time1,time2,dbtablevariables=c("HYY_META.Glob","HYY_META.Glob127","SII1_META.Glob"))
```

```
## https://avaa.tdata.fi/smear-services/smeardata.jsp?tablevariables=HYY_META.Glob,HYY_META.Glob127,SII
```

```
data
```

```
##   Year Month Day Hour Minute Second SII1_META.Glob
## 1 2020     3   8   10     31      0       328.3710
## 2 2020     3   8   10     32      0       330.0683
## 3 2020     3   8   10     33      0       332.5690
```

Specific notes for AVAA API:

Sometimes there are missing rows in the database, align the rows with merge.

Example: Hyytiälä and Siikaneva 1 meteo data in 2004/2005

```
urlstring<-"https://avaa.tdata.fi/smear-services/smeardata.jsp?variables=T168&table=HYY_META&from=2004-

urlstring2<-"https://avaa.tdata.fi/smear-services/smeardata.jsp?variables=T_a&table=SII1_META&from=2004

data<-read.csv(urlstring)
data
```

```
##   Year Month Day Hour Minute Second HYY_META.T168
## 1 2004    12  31   23      0      0     -2.077333
## 2 2004    12  31   23     30      0     -1.819000
## 3 2005     1   1    0      0      0     -1.658667
## 4 2005     1   1    0     30      0     -1.565667
```

```
data2<-read.csv(urlstring2)
data2
```

```
##   Year Month Day Hour Minute Second SII1_META.T_a
## 1 2005     1   1    0      0      0          -1.6
## 2 2005     1   1    0     30      0          -1.5
```

```
merge(data,data2,all=TRUE)
```

```
##   Year Month Day Hour Minute Second HYY_META.T168 SII1_META.T_a
## 1 2004    12  31   23      0      0     -2.077333            NA
## 2 2004    12  31   23     30      0     -1.819000            NA
## 3 2005     1   1    0      0      0     -1.658667          -1.6
## 4 2005     1   1    0     30      0     -1.565667          -1.5
```