An R Package for generating covariance matrices for maximum-entropy sampling from precipitation chemistry data

Hessa Al-Thani · Jon Lee

February 28, 2020

Abstract We present an open-source R package (**MESgenCov** v 0.1.0) for temporally fitting multivariate precipitation chemistry data and extracting a covariance matrix for use in the MESP (maximum-entropy sampling problem). We provide multiple functionalities for modeling and model assessment. The package is tightly coupled with data from the NADP/NTN (National Atmospheric Deposition Program / National Trends Network) on their set of 379 monitoring sites, 1978-present. The user specifies the sites, chemicals and time period desired, fits an appropriate user-specified univariate model for each site and chemical selected, and the package produces a covariance matrix for use by MESP algorithms.

Keywords maximum-entropy sampling · covariance matrix · environmental monitoring · environmetrics · NADP · NTN

Mathematics Subject Classification (2000) 90C27 · 62M30 · 62M10 · 94A17

Introduction

The MESP (maximum-entropy sampling) problem (see [SW87, SW00, FL00, Lee12]) has been applied to many domains where the objective is to determine a "most informative" subset Y_S, of pre-specified size $s = |S| > 0$, from a Gaussian random vecor Y_N, $|N| = n > s$. Information is typically measured

H. Al-Thani
Department of Industrial and Operations Engineering, Univ. of Michigan, Ann Arbor, MI 48105 USA
E-mail: hessakh@umich.edu

J. Lee
Department of Industrial and Operations Engineering, Univ. of Michigan, Ann Arbor, MI 48105 USA
E-mail: jonxlee@umich.edu

by (differential) entropy. Generally, we assume that Y_N has a joint Gaussian distribution with mean vector μ and covariance matrix C. Up to constants, the entropy of Y_S is the log of the determinant of the principle submatrix $C[S, S]$. So, the MESP seeks to maximize the (log) determinant of $C[S, S]$, for some $S \subseteq N$ with $|S| = s$.

The MESP is NP-hard (see [KLQ95]), and there has been considerable work on algorithms aimed at exact solutions for problems of moderate size; see [KLQ95,Lee98,AFLW99,LW03,HLW01,AL04,BL07,Ans18b,Ans18a,CFLL20]. All of this algorithmic work is based on a branch-and-bound framework introduced in [KLQ95]. The bulk of the contribution in all of these references is on different methods for upper bounding the optimal value. All of this work has been developed and validated in the context of a very small number of data sets, despite the fact that of course multivariate data is all around us. The reason for this shortcoming is that despite all of the raw multivariate data that is available, it is not at all simple to turn this data into meaningful covariance matrices for Gaussian random variables.

Our goal with the R package (**MESgenCov** v 0.1.0) that we have developed is to provide such a link — between readily available raw envrionmental-monitoring data and covariance matrices suitable for the MESP — in the context of environmental monitoring. Our work fits squarely into recent efforts to better exploit massive amounts of available data for operations research (in particular, mathematical programming) approaches to decision problems. Even if we have reliable raw data, we can only make good decisions if we have a means to prepare that data so that we can populate our optimization models to meet the assumptions of our models.

We note that another R package of interest is [LZWC19]: "EnviroStat provides functions for spatio-temporal modeling of environmental processes and designing monitoring networks for them based on an approach described in [LZ06]".

In §1, we discuss application of the MESP to environmental monitoring and the NADP/NTN (National Atmospheric Deposition Program / National Trends Network) data environment. In §2, we describe our methodology. In §3, we describe the R package (**MESgenCov** v 0.1.0). In §4, we make some concluding remarks.

1 Environmental monitoring and NADP/NTN data

A key area of application for the MESP has been in environmental monitoring (see [ZSL00, BLZ94, GLSZ93], for example). The idea is that precipitation is collected at many sites, and its chemistry is analyzed. This is costly, and it is a natural question as to whether a subset of the sites might yield data without much loss of information (as measured by entropy). But it is a challenge to process the raw data in such a way that multivariate normality is achieved, because only then are the model of the MESP and its related algorithms applicable.

The NADP maintains the NTN (see [NAD18]); this network has measured the chemistry (i.e., ammonium, calcium, chloride, hydrogen, magnesium, nitrate, pH, potassium, sodium, and sulfate) of precipitation at 379 monitoring sites across the US, with some data available as far back as 1978; at present, 255 sites are active.

Our R package is tightly coupled with this precipitation and chemistry data. We are interested in instances of the MESP where n *user-specified* site/chemical pairs comprise N. Precipitation data (measured in L) are available on a daily basis, and chemical concentrations (measured in mg/L) are available on a weekly basis. Both datasets are available in our R package and can be loaded respectively as

```
#load package's internal data
 >   data("weeklyConc")
 >   data("preDaily")
```

A full description of the daily and weekly precipitation data appears in Appendix B, derived from `http://NADP.slh.wisc.edu/data/ntn/meta/ntn-daily-Meta.pdf` and `http://NADP.slh.wisc.edu/data/ntn/meta/ntn-weekly-Meta.pdf`, courtesy of the NADP[1]

Small snapshots of the data can easily be viewed. For example, we can output the first 6 rows and first 5 columns of the weekly raw data.

```
#display part of the weeklyConc data frame
 weeklyConc[1:6,1:5]

   siteID              dateon             dateoff yrmonth    ph
 1   AB32 2016-09-13 18:40:00 2016-09-20 15:10:00  201609 -9.00
 2   AB32 2016-09-20 15:15:00 2016-09-28 16:00:00  201609 -9.00
 3   AB32 2016-09-28 16:00:00 2016-10-05 16:55:00  201610  6.56
 4   AB32 2016-10-05 16:55:00 2016-10-11 17:00:00  201610 -9.00
 5   AB32 2016-10-11 17:00:00 2016-10-18 20:00:00  201610 -9.00
 6   AB32 2016-10-18 20:00:00 2016-10-25 18:00:00  201610  4.73
```

2 Our methodology

2.1 NADP/NTN data processing

We process the raw NADP/NTN data in a similar way to earlier uses in the context of the MESP in the field of environmental statistics (see [GLSZ93]).

We calculate the level of a chemical's concentration by summing weekly quantities (mg) of the chemical, over a month, and dividing the monthly total by total precipitation (L), over dates in that month, to get monthly values

[1] National Atmospheric Deposition Program (NRSP-3). 2019. NADP Program Office, Wisconsin State Laboratory of Hygiene, 465 Henry Mall, Madison, WI 53706.

of sulfate concentration (mg/L). We use monthly concentrations instead of the given weekly concentrations because there is a large proportion of missing data of weeks in the year compared to full months. Furthermore the univariate models were better at predicting average monthly concentrations than they were are at predicting weekly concentrations.

For a given monitoring site, chemical, and month $t = 0, 1, \ldots, T-1$, let

$$\begin{aligned} W(t) &:= \text{set of weeks in month } t, \\ D(w) &:= \text{set of days in week } w, \\ c_w &:= \text{recorded chemical concentration (mg/L) for week } w \\ &\quad\ (c_w = * \text{ denotes an unrecorded value}), \\ p_d &:= \text{recorded precipitation quantity (L) for day } d, \\ p_w &:= \text{precipitation quantity (L) for week } w;\ p_w = \textstyle\sum_{d \in D(w)} p_d. \end{aligned}$$

Then the chemical concentration (mg/L) for month t is calculated as

$$y(t) := \frac{\sum_{w \in W(t): c_w \neq *} p_w c_w}{\sum_{w \in W(t): c_w \neq *} \sum_{d \in w} p_d}.$$

It should be noted that when there is no weekly value available for the chemical quantity, we do not use the precipitation values for any of the days in such a week (so as to not artificially dilute the chemical concentration level for the month).

Next, we fit a temporal model to $\log(y(t))$, which is a rather standard method for handling heavy-tailed distributions.

A quick look at some graphics indicates that there are clear long-term trends; see Figure 1[2], from which we can see that sulfate concentrations are generally trending downward over time. Again, looking at some data, we can easily see periodic trends; see Figure 2, where we can easily see a yearly periodicity.

The general model that we provide is

$$\widehat{\log(y(t))} = \sum_{i=0}^{r} \beta_i t^i + \sum_{j=1}^{k} \left[a_j \cos\left(\frac{2\pi j t}{S}\right) + b_j \sin\left(\frac{2\pi j t}{S}\right) \right], \tag{1}$$

with the parameters β_i, a_i, and b_i fit by ordinary linear regression. The user can specify the degree r for the polynomial part of the model which we think of as a truncated Taylor series, aimed at capturing aperiodic trends. Periodic trends are captured via a truncated Fourier series, truncated at level k.

We note that [GLSZ93] used the following model to de-seasonalize and de-trend the log-transformed monthly sulfate concentration values:

$$\widehat{\log(y(t))} = \beta_1 + \beta_2 t + a_1 \cos\left(\frac{2\pi t}{12}\right) + b_1 \sin\left(\frac{2\pi t}{12}\right). \tag{2}$$

[2] Reprinted with the kind permission of the National Atmospheric Deposition Program (NRSP-3). 2019. NADP Program Office, Wisconsin State Laboratory of Hygiene, 465 Henry Mall, Madison, WI 53706.

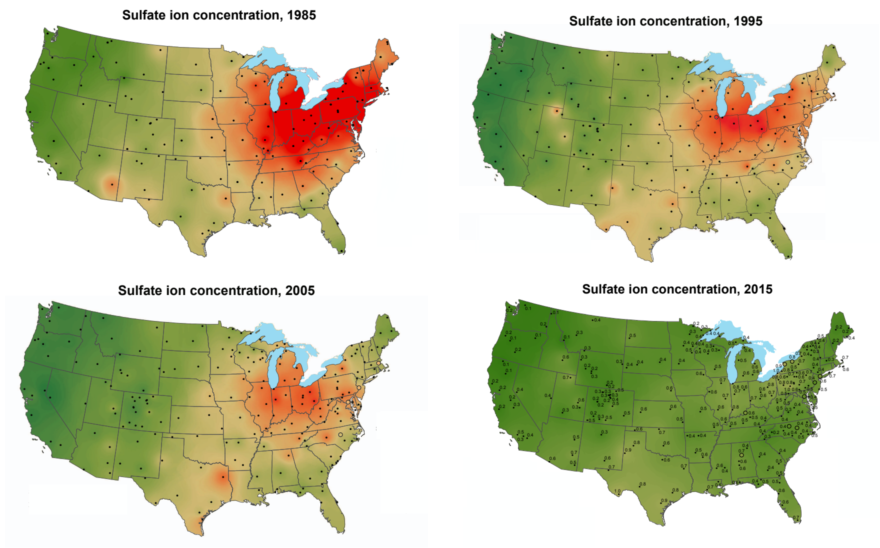


Fig. 1 Sulfate concentration over time

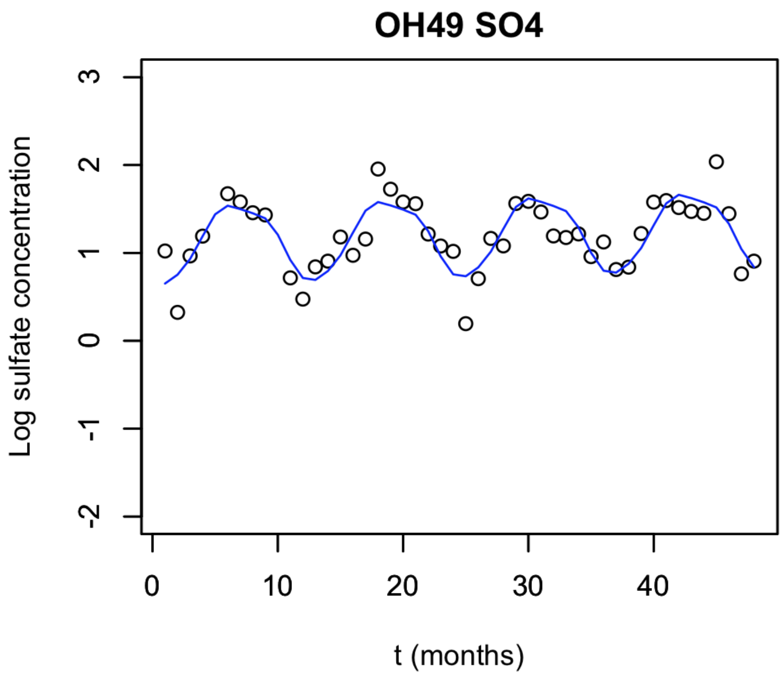


Fig. 2 Log sulfate concentration over a four-year period at a site

This is just an affine model $\beta_1 + \beta_2 t$ plus a sinusoidal model with monthly periodicity and intercept a_1. The simple model (2) is (1) with $r = 1$, $k = 1$

and $S = 12$. We found that (2) did well at normalizing the errors for certain sites, but some sites, such as "MD13" and "NC03", (2) did not do so well. So rather than fix (2) as the model for our R package, we provide the flexibility of (1).

To produce the covariance matrix we need error values for each time point. Missing values in the NADP/NTN data set means that each set of error values produced by the model may vary in size. So we have filled in missing values for each site and time point by sampling from a normal distribution with the mean being the predicted value by the univariate model and the standard deviation being the standard error of the univariate model.

3 MESgenCov

Our R package **MESgenCov** can be obtained from `https://github.com/hessakh/MESgenCov` and installed as follows:

```
#install MESgenCov
 >   install.packages("devtools")
 >   library(devtools)
 >   install_github("hessakh/MESgenCov")
 >   library(MESgenCov)
```

MESgenCov contains functions in the `S3` class to create a covariance matrix from the desired subset of NADP/NTN data. The function `getCov()` returns a covariance matrix, a list of univariate model summaries, and a table of normality tests produced by the MVN R package (see [KGZ14]). The multivariate analysis is used to asses the validity of the covariance matrix to be used as input for the MESP. The user can make adjustments to the input of `getCov()` so as to obtain a covariance matrix for a mulivariate Gaussian vector, and is thus valid for use in the MESP.

To avoid sites with a small sample size for the specified time-frame, the function `getSites()` outputs a vector of the sites with the largest sample of data for a given time-frame and measured chemical (see §3.2). To find sites that are spatially "spread out" but have at least some specified sample size, the function `maxDistSites()` can be used to obtain a list of geographically sparse sites (see §3.2). Finally, in the case where the residuals from a univariate model do not appear to be normally distributed, the function `lambertWtransform()` allows the user to transform the residuals (from a univariate model) using the R package LambertW: Probabilistic Models to Analyze and Gaussianize Heavy-Tailed, Skewed Data (see [Goe16]). This can be very effective in situations where the distributions seems to have heavy tails and some skewness (see [Goe11] and§3.3).

3.1 getCov

`getCov()` takes a 15 column data frame as input where each column corresponds to one of the user-specifications shown in Figure 3. The 15 specifications in the input allow the user to specify the subset of data to analyze and gives the user options in displaying different parts of the analysis.

3.1.1 Input

Arguments	Definition
startdateStr	Date and time of when to start analyzing the data, in the format = m/d/y H:M
enddateStr	Date and time of when to stop analyzing the data, in the format = m/d/y H:M
comp	String of pollutant or acidity level to be analyzed, the pollutants name should be used as it appears in weeklyConc
use36	TRUE if default 36 sites should be added, FALSE otherwise
siteAdd	List of strings of siteIDs that should be analyzed
outlierDatesbySite	List of sites where outliers should be analyzed
siteOutliers	List of sites where outliers should be removed
removeOutliers	Specify siteID string for outlier analysis
plotMulti	TRUE if multivariate analysis plots should be displayed, FALSE otherwise
sitePlot	Specify list of siteIDs to be plotted
plotAll	TRUE if plots for all sites should be displayed, FALSE otherwise
writeMat	TRUE if .mat file of the resulting covariance matrix should be written in the working directory
seas	Approximate periodicity of data, typically 12 for monthly data
r	Integer <=5, see univariate model
k	Integer <= 5, see univariate model

Fig. 3 Input parameters for `getCov`

A default set of inputs can be found in the stored data frame "defaultInput". Each column of "defaultInput" is an argument in the function `getCov()`. After storing "defaultInput" in a variable in the user's workspace, the input can be changed. For example, below we store the "defaultInput" data frame in a variable "df", and then change the end date:

```
#Load defaultInput data frame and store in df
 >   data("defaultInput")
 >   df <- defaultInput
```

```
>   df

       startdateStr   enddateStr      use36   siteAdd
    1  01/01/83 00:00 12/31/86 00:00  TRUE    NULL
      outlierDatesbySite siteOutliers comp plotMulti sitePlot
    1 NULL               NULL         SO4  FALSE     NULL
      plotAll writeMat seas r k
      FALSE   FALSE    12   1 1

#Change the end date to extend the sample of data taken from
 weeklyConc
df$enddateStr   <- "12/31/88 00:00"
```

3.1.2 Output

The function `getCov` produces a list with the the following elements:

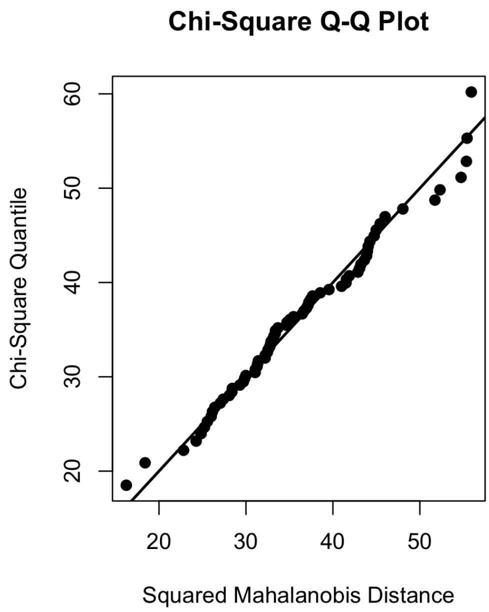
Output	Definition
cov	Covariance matrix produced by univariate model residuals
listMod	List of univariate model summaries produced by lm()
sites	List of sites that were analyzed
mvn	Output of the MVN package
univariateTest	Univariate test output, also by the MVN package
residualData	Data frame of residuals produced by the univariate model
residualDataNA	Data frame of residuals, where missing values are left as NA
rosnerTest	Output of the Rosner's test for outlier analysis produced by the EnvStats package; see [Mil13]
pred	List of predicted values produced by the univariate model for each site

Here we show how to access these elements, and certain plots after running the function:

1. Multivariate and univariate normality

```
#Change part of the input data frame df
>   df$plotMulti <- TRUE
#Change univariate model parameter from 1 to 3
>   df$k <- 3
#Store output in variable g so that the list of outputs given
```

```
 by getCov() can be called
>   g <- getCov(df)
```




```
>  g$univariateTest

           Test  Variable Statistic   p value Normality
1 Shapiro-Wilk  AL10S04      0.9347     0.001        NO
2 Shapiro-Wilk  IL11S04      0.9894    0.8121       YES
3 Shapiro-Wilk  IL18S04      0.9909    0.8854       YES
4 Shapiro-Wilk  IL19S04      0.9183     2e-04        NO
5 Shapiro-Wilk  IL35S04      0.8709    <0.001        NO
```

2. Output all MVN package analysis

   The following output is a call to the MVN package that produces multivariate analysis based on the Mardia method, univariate analysis based on the Shapiro-Wilson method and a multivariate outlier test that is presented as a plot and not as an output in the user's R console.

```
>  g <- getCov(df)
#Display full output of the MVN package
>  g$mvn

 $multivariateNormality
```

```
               Test Statistic p value Result
1 Mardia Skewness    14.019  0.1721    YES
2 Mardia Kurtosis   -0.1465  0.8835    YES
3             MVN      <NA>    <NA>    YES

$univariateNormality
          Test  Variable Statistic   p value Normality
1 Shapiro-Wilk  NY52S04     0.9821    0.3981    YES
2 Shapiro-Wilk  TN11S04     0.9830    0.4385    YES
3 Shapiro-Wilk  IL63S04     0.9873    0.6858    YES

$Descriptives
         n      Mean   Std.Dev  Median     Min    Max
NY52S04 72  1.1709e-17 0.2812  0.0333 -0.8815 0.6170
TN11S04 72 -1.4991e-02 0.4666 -0.0020 -0.9826 1.4243
IL63S04 72  4.8127e-18 0.3016 -0.0373 -0.7353 0.7899
              25th       75th        Skew   Kurtosis
NY52S04 -0.2211160 0.1840066 -0.3702252 0.08937981
TN11S04 -0.3258896 0.2130045  0.4513955 0.29813810
IL63S04 -0.1836875 0.1724338  0.3287155 0.06092834
```

The specific call of the MVN package is

```
>  mvn(dfRes[,-1], subset = NULL, mvnTest = "mardia",
 covariance = TRUE, tol = 1e-25, alpha = 0.5, scale = FALSE,
 desc = TRUE, transform = "none",univariateTest = "SW",
 univariatePlot = "none", multivariatePlot ="none",
 multivariateOutlierMethod = "none", bc = FALSE, bcType =
 "rounded", showOutliers = FALSE, showNewData = FALSE).
```

See [KGZ14] for details on the MVN package.

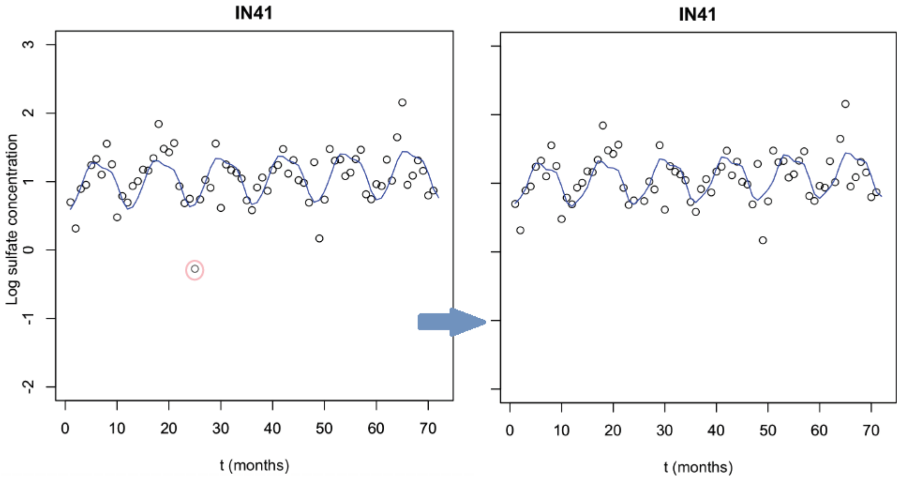
3. Outlier test for specific tests

```
 df$siteOutliers <- list(c("IN41"))
 df$sitePlot <- list(c("IN41"))
 g <- getCov(df)
 i <- match("IN41",g$sites)
 g$rosnerTest[[i]]$all.stats

  i  Mean.i   SD.i   Value Obs.Num  R.i+1 lambda.i+1 Outlier
1 0 -0.0069 0.2814 -0.9359      25 3.3013     3.2680    TRUE
2 1  0.0062 0.2604 -0.7194      30 2.7862     3.2628   FALSE
3 2  0.0165 0.2471  0.7215      66 2.8533     3.2576   FALSE
```

By changing the input, we can remove the outliers detected by the Rosner's test. Note that the plots are generated after running getCov(). Furthermore, getCov() does not need to be stored in a variable to generate the plots.

```
#Remove month 25 from site IN41's pollutant concentration data
>  df$outlierDatesbySite <- c("IN41",25)
>  getCov(df)
```

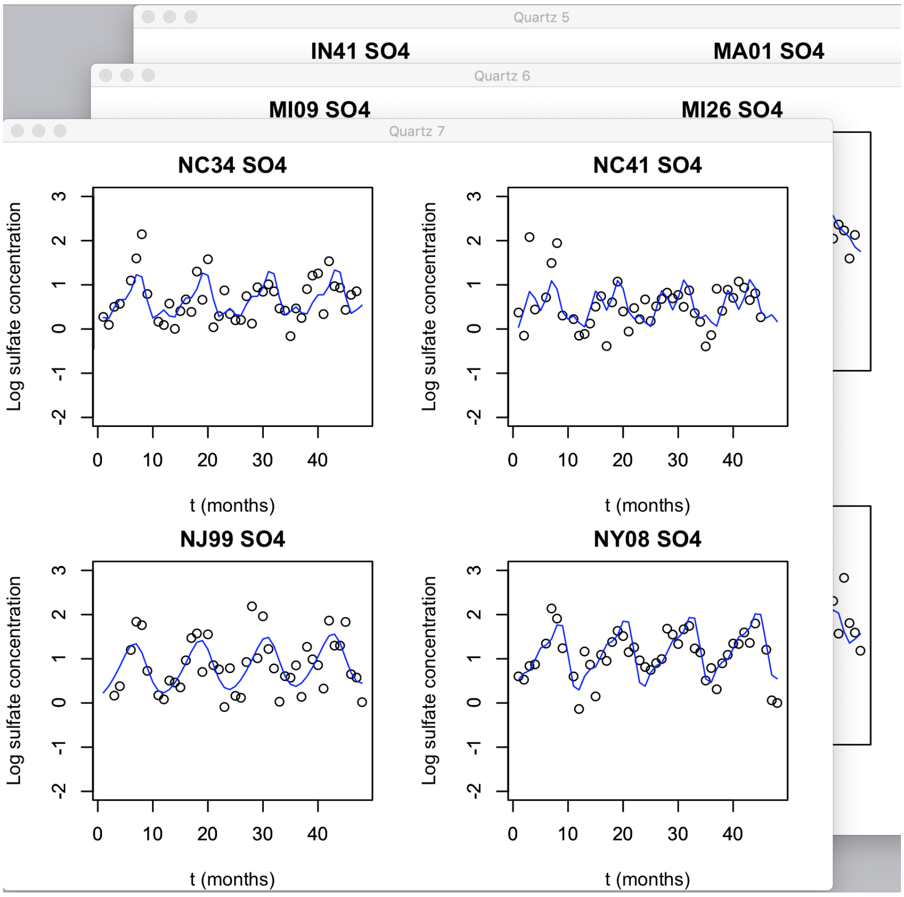


4. Outlier test for all sites

```
 #take sites used in analysis in g and run outlier test
  df$siteOutliers <- list(g$sites)
 #remove data points identified as outliers from these sites
  df$removeOutliers <- list(g$sites)
  g <- getCov(df)
```

5. Plot all sites

```
>  df$plotAll <- TRUE
>  getCov(df)
```



6. Covariance matrix

```
#Remove default list of sites so that their data is not
 analyzed
>  df$use36 <- FALSE
#Add new site list
>  df$siteAdd <- list(c("NY52", "TN11", "IL63"))
#Remove any set of sites and pollutant combinations
 that had been previously added
>  df$siteOutliers <- NULL
>  df$outlierDatesbySite <- NULL
>  df$removeOutliers <- NULL
>  g <- getCov(df)
#Print covariance matrix
>  round(g$cov,digits = 4)

         NY52SO4 TN11SO4 IL63SO4
```

```
NY52S04  0.0791  0.0047  0.0009
TN11S04  0.0047  0.2177  0.0185
IL63S04  0.0009  0.0185  0.0909
```

7. Save covariance matrix as a .mat file (to populate an instance of the MESP, for example).
   This is done by simply setting the input data frame attribute `writeMat` to TRUE. The .mat file will be saved to the user's current working directory as covSites.mat. For processing further with Matlab, use the (Matlab) 'load' command.

```
#Write cov into .mat file in current directory
>  df$writeMat <- TRUE
>  g           <- getCov(df)
```

   In the case that the user has already generated an output by the function `getCov()`, it is possible to also create the .mat file in the following manner.

```
>  library(rmatio)
>  write.mat(g$cov,filename = "covariance1.mat")
```

8. Univariate model summaries

```
>  result <- getCov(df)
#Store site list in sites variable
>  sites  <- result$sites
#Find site OH71 index in the list
>  i = match(c("NY52"),sites)
#Use site index to find model summary for NY52
>  result$listMod[i]


    [[1]]
    Call:
    lm(formula = y1 ~ I(cos(t*(2*pi/s))) + I(sin(t*(2 *
        pi/s))) + I(cos(t*(2*pi/s)*2)) + I(sin(t*
        (2*pi/s)*2)) + I(cos(t*(2*pi/s)*3)) +
        I(sin(t*(2*pi/s)*3)) + I(t), data = df)

    Residuals:
        Min      1Q  Median      3Q     Max
    -0.5236 -0.1677 -0.0189  0.1818  0.9087

    Coefficients:
                        Estimate Std. Error t value Pr(>|t|)
    (Intercept)           1.1110     0.0712   14.40  < 2e-16***
    I(cos(t*(2*pi/s)))   -0.3541     0.0494   -9.75  2.8e-14***
```

```
    I(sin(t*(2*pi/s)))   -0.1109    0.0498   -2.55    0.013*
    I(cos(t*(2*pi/s)*2))  0.0500    0.0494   -0.37    0.716
    I(sin(t*(2*pi/s)*2))  0.0797    0.0494    2.13    0.037*
    I(cos(t*(2*pi/s)*3))  0.0391    0.0494   -0.20    0.845
    I(sin(t*(2*pi/s)*3))  0.0536    0.0494   -0.52    0.604
    I(t)                 -0.0010    0.0017   -0.61    0.546
    ---
 Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

    Residual standard error: 0.2961 on 64 degrees of freedom
    Multiple R-squared:  0.6265,Adjusted R-squared:  0.5857
    F-statistic: 15.34 on 7 and 64 DF,  p-value: 1.327e-11
```

9. Output data frame of residuals

```
>   g <- getCov(df)
#Display dataframe containing the residuals
 from the fitted univariate model
>    g$residualData[1:5,]

      NY52S04     TN11S04    IL63S04
 1  0.1959813 -0.44377735  0.2824803
 2  0.6170340 -0.38791938 -0.3495080
 3 -0.4510128  1.05519620 -0.1158200
 4 -0.1580145 -0.01789642 -0.1297776
 5 -0.3218159 -0.47833685 -0.3415119
```

3.2 Functions for getting a vector of sites

`getCov()` takes site lists as input, the function `getSites()` produces a list of sites with available data for a specified time frame. The code below produces a list of 36 sites with the most weekly data between the years 1983–1986.

```
> result <- getSites("01/01/83 00:00","12/31/86 00:00",36,104,
          "S04","")
> result$finalList

  [1] "OH71" "NY08" "WV18" "MI53" "NH02" "OH49" "PA42" "ME09"
  [9] "IN34" "MA13" "NY52" "NY10" "WA14" "NY20" "OH17" "ME00"
 [17] "TN00" "IL63" "MI99" "WI28" "IN41" "PA29" "WI36" "ME02"
 [25] "MI09" "MO05" "NC03" "NJ99" "PA15" "CO19" "MN18" "WI37"
 [33] "AR27" "KS31" "ME98" "MO03"
```

The 4th input specifies the minimum sample of weekly data required to be included in the produced list and the last input tells the function to only look at sites in the Northern region of the US. Other options for region "W","S","N", see Appendix A for the precise geographic split.

```
> NSites <- getSites("01/01/83 00:00","12/31/86 00:00",36,104,
            "SO4","N")
> NSites$finalList

 [1] "OH71" "NY08" "MI53" "NH02" "OH49" "PA42" "ME09" "IN34"
 [9] "MA13" "NY52" "NY10" "NY20" "OH17" "ME00" "MI99" "WI28"
[17] "IN41" "PA29" "WI36" "ME02" "MI09" "NJ99" "PA15" "MN18"
[25] "WI37" "ME98" "IL11" "IL18" "MN16" "MI26" "NE15" "VT01"
[33] "NY99" "MA01" "MA08" "MN27"
```

The function `maxDistSites()` prioritizes sites that are farther away from each other. This function takes the same arguments as input as `getSites()` except for the last argument where instead of specifying a region, the user can specify which site should be included first. If the user let's the last argument be 1 then the site with the most data for the specified time period will be chosen, if the user let's the last argument be 2 then the site with the second most amount of data will be chosen, and so on.

```
>  maxdist <- maxDistSites("01/01/83 00:00","12/31/86 00:00",36,
             104,"SO4",1)
>  maxdist$finalList

 [1] "OH71" "WA14" "TX04" "FL11" "ME00" "WY06" "MN27" "LA12"
 [9] "CA45" "OK00" "NY99" "GA41" "MI99" "AZ03" "MT05" "NC35"
[17] "MO05" "CO00" "WY99" "IN34" "KY03" "MI09" "FL03" "MA01"
[25]  "OR10" "PA42" "AR27" "MN16" "TX21" "VT99" "NE15" "VA13"
[33]  "CO15" "CO22" "NY52" "AR02"
```

3.3 Lambert W transformation on univariate data

For a number of sites, the residuals produced by our univariate model have skewed distributions with heavy tails. In particular, this is the case for many sites when the sample of data is taken over a period longer than 4 years. To deal with this issue, we have incorporated functions from the LambertW package (see [Goe16]) in the function `lambertWtransform()` that will allow a user to transform the residuals produced by the deterministic univariate model. The LambertW package estimates the parameters that fit a Lambert W distribution on the given univariate data. Then the underlying Gaussian distribution implied by the Lambert W distribution is extracted and is used for the multivariate analysis in the function `lambertWtransform()`. The `lambertWtransform()` function takes the following as input: a data frame of residuals, and two logical inputs specifying whether to plot the multivariate qq plot and whether to produce the .mat file containing the covariance matrix with the Lambert W transformed residuals. Details on the algorithms that perform the transformation can be found in [Goe11]. Here we show an example where we transform the residuals of 50 sites stored in an internal dataset, named "dfRes50".

```
> data("dfRes50")
> loutput <- lambertWtransform(dfRes=dfRes50, plotMulti=FALSE,
             writeMat=FALSE)
> loutput$mvn$multivariateNormality

              Test  Statistic   p value   Result
 1 Mardia Skewness      22800    0.0004       NO
 2 Mardia Kurtosis      0.418    0.6763      YES
 3             MVN       <NA>      <NA>       NO
```

This function produces a list of four outputs:

1. `loutput$mvn` contains the results of applying the multivariate analysis by the MVN package
2. `loutput$cov` contains the covariance matrix produced by the transformed residuals
3. `loutput$newResiduals` contains the data frame of Lambert W transformed residuals
4. `loutput$univariateTest` contains the univariate tests produced by the MVN function for the transformed residuals

```
> data("dfRes50")
> dfRes50 <- dfRes50
> loutput <- lambertWtransform(dfRes=dfRes50, plotMulti=FALSE,
             writeMat=FALSE)
> loutput$mvn
> loutput$cov
> loutput$newResiduals
> loutput$univariateTest
```

Here we present an example where we use `maxDistSites()` to get a list of 50 sites that is geographically sparse and has at least 200 weeks of data between 1986 and 1994. From this list of sites, a covariance and its corresponding multivariate normality test is generated and compared to the Lambert W transformed output.

```
#get list of sites
> maxd  <- maxDistSites("01/01/86 00:00","12/31/94 00:00",50,
          200,"S04",1)
#create input data frame
```

```
>  df <- defaultInput
#use list of sites and specification in maxd
>  df$siteAdd <- list(maxd$finalList)
>  df$startdateStr <- maxd$startDate
>  df$use36 <- FALSE
>  df$comp <- maxd$comp
>  df$enddateStr <- maxd$endDate
>  df$writeMat <- TRUE
>  output <- getCov(df)
>  output$mvn$multivariateNormality

               Test   Statistic  p value    Result
  1 Mardia Skewness       22962   2.511e-05     NO
  2 Mardia Kurtosis      0.2408   0.8097       YES
  3             MVN        <NA>     <NA>        NO

loutput <- lambertWtransform(g$residualDataNA, TRUE,FALSE)
loutput$mvn$multivariateNormality

##              Test          Statistic           p value Result
## 1 Mardia Skewness  22266.3107494977 0.214106607512284    YES
## 2 Mardia Kurtosis -1.59221962100786 0.111335366028036    YES
## 3             MVN              <NA>              <NA>    YES
```

3.4 Internal datasets and their properties

We provide five internal datasets of covariance matrices produced by a geographically sparse list of sites over different time periods and for different pollutants. The exact specifications used to produce these sites can be found in Appendix C. We offer these covariance matrices for the convenience of the user. We note that although the list of sites are quite spread out geographically, they are not independent. We test the independence of the covariance matrix using a likelihood ratio test (see [RC12, p. 275]) The test statistic is

$$u := -\left(\nu - \frac{2m+5}{6}\right)\log(\det(R)),$$

where $m :=$ number of sites, $\nu := m(m+1)/2$, and R is the sample correlation matrix. If The null hypothesis H_0 is that the variates are independent, and we reject H_0 if $u > \chi^2_{m(m-1)/2,\alpha}$ where for our analysis $\alpha = 0.05$. The five covariance matrices are named "maxd1Cov", "maxd2Cov", "maxd3Cov", "maxd4Cov", and "maxd5Cov", and their corresponding test statistics u are 20182, 25851, 26133, 28331, and 24898, all comfortably giving evidence to reject H_0 at the $\alpha = 0.05$ level (we reject when $u > 1308$).

We have made a function available that performs this independence test. Here we show how it can be used on a data frame of residuals produced by `getCov()`.

```
> maxd1  <- maxDistSites("01/01/86 00:00","12/31/94 00:00",50,
            200,"SO4",1)
> df$comp <- maxd1$comp
> df$enddateStr <- maxd1$endDate
> df$startdateStr <- maxd1$startDate
> df$siteAdd <- list(maxd1$finalList)
> result  <- getCov(df)
> indp  <- independenceTest(result$residualData)
> indp$test

    chisq dist likelihood ratio  chisq    independent
  1                     20182  1307.54   FALSE
```

4 Concluding remarks

We are currently working on enhancements to **MESgenCov**. Ultimately, we would like to make it easy to use data sets from other application domains, and to make it easier for a user to use other models than the one we provide. Finally, we hope to eventually have a seamless integration with algorithms for the MESP.

Acknowledgments

The authors are very grateful to Dr. Martin Shafer and Robert Larson for helping us gain access to the NADP/NTN data in a convenient form.

Financial and Ethical disclosures

J. Lee was funded by the Air Force Office of Scientific Research (Complex Networks program), FA9550-19-1-0175. H. Al-Thani was funded by the Qatar National Research Fund (Graduate Sponsorship Research Award), GSRA4-2-0526-17114. The authors declare that they have no conflict of interest.

References

AFLW99. Kurt M. Anstreicher, Marcia Fampa, Jon Lee, and Joy Williams. Using continuous nonlinear relaxations to solve constrained maximum-entropy sampling problems. *Mathematical Programming*, 85(2, Ser. A):221–240, 1999.

AL04. Kurt M. Anstreicher and Jon Lee. A masked spectral bound for maximum-entropy sampling. In *mODa 7—Advances in model-oriented design and analysis*, Contrib. Statist., pages 1–12. Physica, Heidelberg, 2004.

Ans18a. Kurt M. Anstreicher. Efficient solution of maximum-entropy sampling problems. Preprint available at: `https://www.biz.uiowa.edu/faculty/anstreicher/papers/linx.pdf`, July 2018.

Ans18b. Kurt M. Anstreicher. Maximum-entropy sampling and the boolean quadric polytope. *Journal of Global Optimization*, 72(4):603–618, 2018.

BL07. Samuel Burer and Jon Lee. Solving maximum-entropy sampling problems using factored masks. *Mathematical Programming*, 109(2-3, Ser. B):263–281, 2007.

BLZ94. Philip J. Brown, Nhu D. Le, and James V. Zidek. Multivariate spatial interpolation and exposure to air pollutants. *Canad. J. Statist.*, 22(4):489–509, 1994.

CFLL20. Zhongzhu Chen, Marcia Fampa, Amélie Lambert, and Jon Lee. Mixing convex-optimization bounds for maximum-entropy sampling, 2020. `http://arxiv.org/abs/2001.11896`.

FL00. Valerii Fedorov and Jon Lee. Design of experiments in statistics. In *Handbook of semidefinite programming*, volume 27 of *Internat. Ser. Oper. Res. Management Sci.*, pages 511–532. Kluwer Acad. Publ., Boston, MA, 2000.

GLSZ93. Peter Guttorp, Nhu D. Le, Paul D. Sampson, and James V. Zidek. Using entropy in the redesign of an environmental monitoring network. In *Multivariate environmental statistics*, volume 6 of *North-Holland Ser. Statist. Probab.*, pages 175–202. North-Holland, Amsterdam, 1993.

Goe11. Georg M. Goerg. Lambert W random variables - a new family of generalized skewed distributions with applications to risk estimation. *Annals of Applied Statistics*, 5(3):2197–2230, 2011.

Goe16. Georg M. Goerg. *LambertW: Probabilistic Models to Analyze and Gaussianize Heavy-Tailed, Skewed Data. Version 0.6.4*, 2016. `https://cran.r-project.org/web/packages/LambertW/`.

HLW01. Alan J. Hoffman, Jon Lee, and Joy Williams. New upper bounds for maximum-entropy sampling. In *mODa 6—advances in model-oriented design and analysis (Puchberg/Schneeberg, 2001)*, Contrib. Statist., pages 143–153. Physica, Heidelberg, 2001.

KGZ14. Selcuk Korkmaz, Dincer Goksuluk, and Gokmen Zararsiz. MVN: An R Package for Assessing Multivariate Normality. *The R Journal*, 6(2):151–162, 2014.

KLQ95. Chun-Wa Ko, Jon Lee, and Maurice Queyranne. An exact algorithm for maximum entropy sampling. *Operations Research*, 43(4):684–691, 1995.

Lee98. Jon Lee. Constrained maximum-entropy sampling. *Operations Research*, 46(5):655–664, 1998.

Lee12. Jon Lee. *Encyclopedia of Environmetrics, A.H. El-Shaarawi and W.W. Piegorsch, eds.*, chapter Maximum entropy sampling, 2nd edition, pages 1570–1574. Wiley, 2012.

LW03. Jon Lee and Joy Williams. A linear integer programming bound for maximum-entropy sampling. *Mathematical Programming*, 94(2-3, Ser. B):247–256, 2003.

LZ06. Nhu D. Le and James V. Zidek. *Statistical analysis of environmental space-time processes.* Springer Series in Statistics. Springer, New York, 2006.

LZWC19. Nhu Le, Jim Zidek, Rick White, and Davor Cubranic. EnviroStat: Statistical Analysis of Environmental Space-Time Processes. Version 0.4-2. `https://rdrr.io/cran/EnviroStat/`, 2019.

Mil13. Steven P. Millard. *EnvStats: An R Package for Environmental Statistics.* Springer, New York, 2013.

NAD18. NADP. National Acidic Deposition Program, National Trends Network. `https://nadp.slh.wisc.edu/ntn/`, 2018.

RC12. A.C. Rencher and W.F. Christensen. *Methods of Multivariate Analysis.* Wiley Series in Probability and Statistics. Wiley, 2012.

SW87. Michael C. Shewry and Henry P. Wynn. Maximum entropy sampling. *Journal of Applied Statistics*, 14(2):165–170, 1987.

SW00. Paola Sebastiani and Henry P. Wynn. Maximum entropy sampling and optimal bayesian experimental design. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(1):145–157, 2000.

ZSL00. James V. Zidek, Weimin Sun, and Nhu D. Le. Designing and integrating composite networks for monitoring multivariate Gaussian pollution fields. *Journal of the Royal Statistical Society. Series C. Applied Statistics*, 49(1):63–79, 2000.

5 Appendices

Appendix A: Geographic Split

Appendix B: NADP/NTN Data Descriptions

NADP/NTN Daily Data

Column number	Field	Data type	Description
1	SiteID	Char(4)	Site Identifier
2	StartTime	Char(16)	Period start, reported in Greenwich Mean Time (GMT) YYYY-MM-DD hh:mm format
3	EndTime	Char(16)	Period end, reported in Greenwich Mean Time (GMT) YYYY-MM-DD hh:mm format
4	Amount	Integer	Precipitation depth, inches Missing = -9, Trace precipitation amount = -7

NADP/NTN Weekly Data

Column number	Field	Data type	Description
1	SiteID	Char(4)	Site Identifier
2	DateOn	Char(16)	Date on which the sample bucket was installed on the collector, reported in Greenwich Mean Time (GMT) YYYY-MM-DD hh:mm format
3	DateOff	Char(16)	Date on which the sample bucket was removed from the collector, reported in Greenwich Mean Time (GMT) YYYY-MM-DD hh:mm format
4	yrMonth	Integer	Year and Month of sample midpoint, in YYYYMM format
5	ph	Decimal	Negative log of the hydrogen ion concentration as measured at the CAL, in pH units
6	Ca	Decimal	Ca concentration, mg/L
7	Mg	Decimal	Mg concentration, mg/L
8	K	Decimal	K concentration, mg/L
9	Na	Decimal	Na concentration, mg/L
10	NH4	Decimal	NH4 concentration, mg/L
11	NO3	Decimal	NO3 concentration, mg/L
12	Cl	Decimal	Cl concentration, mg/L
13	SO4	Decimal	SO4 concentration, mg/L
14	Br	Decimal	Br concentration, mg/L

Appendix C: Internal Covariance Matrices Site Lists

```
#sites with maximum distance data sets, get 50 sites
> maxd1 <- maxDistSites("01/01/86 00:00","12/31/94 00:00",50,
          200,"SO4",1)
> maxd2 <- maxDistSites("01/01/07 00:00","12/31/14 00:00",50,
          230,"SO4",1)
> maxd3 <- maxDistSites("01/01/07 00:00","12/31/14 00:00",50,
          230,"NO3",1)
> maxd4 <- maxDistSites("01/01/07 00:00","12/31/14 00:00",50,
          230,"Na",1)
> maxd5 <- maxDistSites("01/01/07 00:00","12/31/14 00:00",50,
          230,"ph",1)
```