The R-Package 'surveillance'

Michael Höhle*, Andrea Riebler and Michaela Paul
Department of Statistics
University of Munich
Germany

September 5, 2006

Abstract

This document gives an introduction to the R-Package 'surveillance' containing tools for outbreak detection in routinely collected surveillance data. The package contains an implementation of the procedures described by Stroup et al. (1989), Farrington et al. (1996) and the system used at the Robert Koch Institute, Germany. For evaluation purposes, the package contains example data sets and functionality to generate surveillance data by simulation. To compare the algorithms, benchmark numbers like sensitivity, specificity, and detection delay can be computed for a set of time series. Being an open-source package it should be easy to integrate new algorithms; as an example of this process, a simple Bayesian surveillance algorithm is described, implemented and evaluated.

Keywords: infectious disease, monitoring, aberrations, outbreak, time series of counts.

1 Introduction

Public health authorities have in an attempt to meet the threats of infectious diseases to society created comprehensive mechanisms for the collection of disease data. As a consequence, the abundance of data has demanded the development of automated algorithms for the detection of abnormalities. Typically, such an algorithm monitors a univariate time series of counts using a combination of heuristic methods and statistical modelling. Prominent examples of surveillance algorithms are the work by Stroup et al. (1989) and Farrington et al. (1996). A comprehensive survey of outbreak detection methods can be found in (Farrington and Andrews, 2003).

*Author of correspondance: Department of Statistics, University of Munich, Ludwigstr. 33, 80539 München, Germany, Email: `hoehle@stat.uni-muenchen.de`

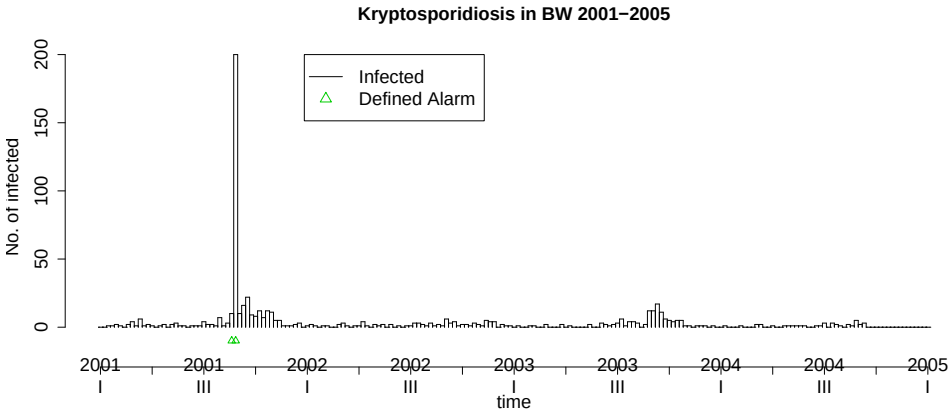
The R-package `surveillance` was written with the aim of providing a test-bench for surveillance algorithms. From the Comprehensive R Archive Network (CRAN) the package can be downloaded together with its source code. It allows users to test new algorithms and compare their results with those of standard surveillance methods. A few real world outbreak datasets are included together with mechanisms for simulating surveillance data. With the package at hand, comparisons like the one described by Hutwagner et al. (2005) should be easy to conduct.

The purpose of this document is to illustrate the basic functionality of the package with R-code examples. Section 2 contains a description of the data format used to store surveillance data, mentions the built-in datasets and illustrates how to create new datasets by simulation. Section 3 contains a short description of how to use the surveillance algorithms and illustrate the results. Further information on the individual functions can be found in the on-line documentation of the package, which is also provided in printed form as an Appendix of this document.

2 Surveillance Data

Denote by $\{y_t\,; t = 1, \ldots, n\}$ the time series of counts representing the surveillance data. Because such data typically are collected on a weekly basis, we shall also use the alternative notation $\{y_{i:j}\}$ with $j = \{1, \ldots, 52\}$ being the week number in year $i = \{-b, \ldots, -1, 0\}$. That way the years are indexed such that most current year has index zero. For evaluation of the outbreak detection algorithms it is also possible for each week to store – if known – whether there was an outbreak that week. The resulting multivariate series $\{(y_t, x_t)\,; t = 1, \ldots, n\}$ is in `surveillance` given by an object of class `disProg` (disease progress), which is basically a `list` containing two vectors: the observed number of counts and a boolean vector `state` indicating whether there was an outbreak that week. A number of time series are contained in the `data` directory, mainly originating from the SurvStat@RKI database at http://www3.rki.de/SurvStat/ maintained by the Robert Koch Institute, Germany (Robert Koch-Institut, 2004). For example the object `k1` describes Kryptosporidosis surveillance data for the German federal state Baden-Württemberg 2001-2005. The peak in 2001 is due to an outbreak of Kryptosporidosis among a group of army-soldiers in boot-camp (Robert Koch Institute, 2001). In `surveillance` the `readData` function is used to bring the time series on `disProg` form. The SurvStat@RKI database at http://www3.rki.de/SurvStat/ maintained by the Robert Koch Institute, Germany, uses a 53 weeks a year format; therefore a conversion with `correct53to52` is necessary.

```
> data(k1)
> plot(k1, main = "Kryptosporidiosis in BW 2001-2005")
```



For evaluation purposes it is also of interest to generate surveillance data using simulation. The package contains functionality to generate surveillance data containing point-source like outbreaks, for example with a Salmonella serovar. The model is a Hidden Markov Model (HMM) where a binary state $X_t, t = 1, \ldots, n$, denotes whether there was an outbreak and Y_t is the number of observed counts, see Fig. 1.

Figure 1: The Hidden Markov Model

The state X_t is a homogenous Markov chain with the following transition matrix

$X_t \backslash X_{t+1}$	0	1
0	p	$1-p$
1	$1-r$	r

Hence $1-p$ is the probability to switch to an outbreak state and $1-r$ is the probability that $X_t = 1$ is followed by $X_{t+1} = 1$. Furthermore, the observation Y_t is Poisson-distributed with log-link mean depending on a seasonal effect and time trend, i.e.

$$\log \mu_t = A \cdot \sin(\omega \cdot (t + \varphi)) + \alpha + \beta t.$$

In case of an outbreak ($X_t = 1$) the mean increases with a value of K, altogether

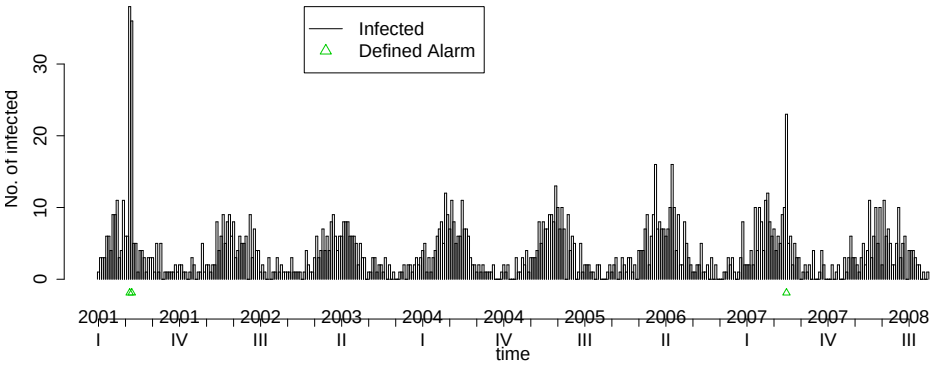
$$Y_t \sim \text{Po}(\mu_t + K \cdot X_t). \tag{1}$$

The model in (1) corresponds to a single-source, common-vehicle outbreak, where the length of an outbreak is controlled by the transition probability r. The daily numbers of outbreak-cases are simply independently Poisson distributed with mean K. A physiologically better motivated alternative could be to operate with a stochastic incubation time (e.g. log-normal or gamma distributed) for each individual exposed to the source, which results

in a temporal diffusion of the peak. The advantage of (1) is that estimation can be done by a generalized linear model (GLM) using X_t as covariate and that it allows for an easy definition of a correctly identified outbreak: each $X_t = 1$ has to be identified. More advanced setups would require more involved definitions of an outbreak, e.g. as a connected series of time instances, where the number of outbreak cases is greater than zero. Care is then required in defining what a correctly identified outbreak for time-wise overlapping outbreaks means.

In `surveillance` the function `sim.pointSource` is used to simulate such a point-source epidemic; the result is an object of class `disProg`.

```
> sts <- sim.pointSource(p = 0.99, r = 0.5, length = 400,
+     A = 1, alpha = 1, beta = 0, phi = 0, frequency = 1,
+     state = NULL, K = 1.7)
> plot(sts)
```



3 Surveillance Algorithms

Surveillance data often exhibit strong seasonality, therefore most surveillance algorithms only use a set of so called *reference values* as basis for drawing conclusions. Let $y_{0:t}$ be the number of cases of the current week (denoted week t in year 0), b the number of years to go back in time and w the number of weeks around t to include from those previous years. For the year zero we use w_0 as the number of previous weeks to include – typically $w_0 = w$. Altogether the set of reference values is thus defined to be

$$R(w, w_0, b) = \left(\bigcup_{i=1}^{b} \bigcup_{j=-w}^{w} y_{-i:t+j} \right) \cup \left(\bigcup_{k=-w_0}^{-1} y_{0:t+k} \right)$$

Note that the number of cases of the current week is not part of $R(w, w_0, b)$.

A surveillance algorithm is a procedure using the reference values to create a prediction $\hat{y}_{0:t}$ for the current week. This prediction is then compared with the observed $y_{0:t}$: if the observed number of cases is much higher than the predicted number, the current week is flagged for further investigations. In order to do surveillance for time $0:t$ an important concern is the choice of b and w. Values as far back as time $-b:t-w$ contribute to $R(w, w_0, b)$ and thus have to exist in the observed time series.

Currently, we have implemented four different type of algorithms in `surveillance`. The Centers for Disease Control and Prevention (CDC) method (Stroup et al., 1989), the Communicable Disease Surveillance Centre (CDSC) method (Farrington et al., 1996), the method used at the Robert Koch Institute (RKI), Germany (Altmann, 2003), and a Bayesian approach documented in Riebler (2004). A detailed description of each method is beyond the scope of this note, but to give an idea of the framework the Bayesian approach developed in Riebler (2004) is presented: Within a Bayesian framework, quantiles of the predictive posterior distribution are used as a measure for defining alarm thresholds.

The model assumes that the reference values are identically and independently Poisson distributed with parameter λ and a Gamma-distribution is used as Prior distribution for λ. The reference values are defined to be $R_{\text{Bayes}} = R(w, w_0, b) = \{y_1, \ldots, y_n\}$ and $y_{0:t}$ is the value we are trying to predict. Thus, $\lambda \sim \text{Ga}(\alpha, \beta)$ and $y_i|\lambda \sim \text{Po}(\lambda)$, $i = 1, \ldots, n$. Standard derivations show that the posterior distribution is

$$\lambda|y_1, \ldots, y_n \sim \text{Ga}(\alpha + \sum_{i=1}^{n} y_i, \beta + n).$$

Computing the predictive distribution

$$f(y_{0:t}|y_1, \ldots, y_n) = \int_0^\infty f(y_{0:t}|\lambda)\, f(\lambda|y_1, \ldots, y_n)\, d\lambda$$

we get the Poisson-Gamma-distribution

$$y_{0:t}|y_1, \ldots, y_n \sim \text{PoGa}(\alpha + \sum_{i=1}^{n} y_i, \beta + n),$$

which is a generalization of the negative Binomial distribution, i.e.

$$y_{0:t}|y_1, \ldots, y_n \sim \text{NegBin}(\alpha + \sum_{i=1}^{n} y_i, \tfrac{\beta+n}{\beta+n+1}).$$

Using the Jeffrey's Prior $\text{Ga}(\frac{1}{2}, 0)$ as non-informative Prior distribution for λ the parameters of the negative Binomial distribution are

$$\alpha + \sum_{i=1}^{n} y_i = \frac{1}{2} + \sum_{y_{i:j} \in R_{\text{Bayes}}} y_{i:j} \qquad \text{and} \quad \frac{\beta + n}{\beta + n + 1} = \frac{|R_{\text{Bayes}}|}{|R_{\text{Bayes}}| + 1}.$$

Using a quantile-parameter α, the smallest value y_α is computed, so that

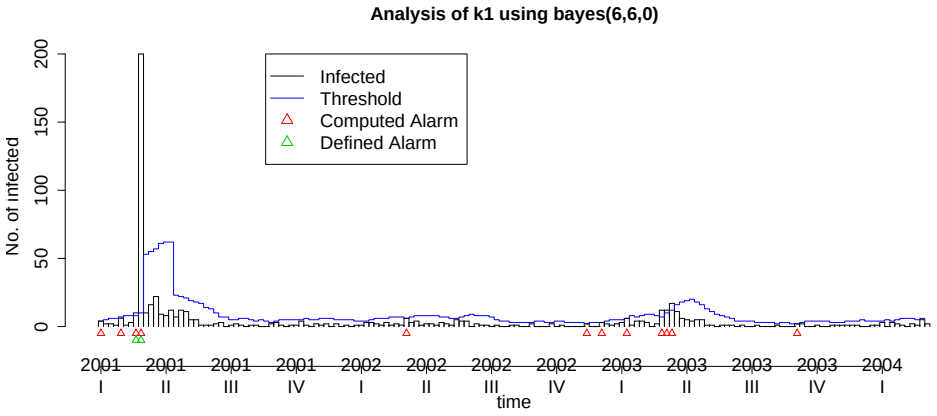
$$P(y \leq y_\alpha) \geq 1 - \alpha.$$

Now

$$A_{0:t} = I(y_{0:t} \geq y_\alpha),$$

i.e. if $y_{0:t} \geq y_\alpha$ the current week is flagged as an alarm. As an example, the `Bayes1` method uses the last six weeks as reference values, i.e. $R(w, w_0, b) = (6, 6, 0)$, and is applied to the `k1` dataset with $\alpha = 0.01$ as follows.

```
> k1.b660 <- algo.bayes(k1, control = list(range = 27:192,
+     b = 0, w = 6, alpha = 0.01))
> plot(k1.b660, disease = "k1", firstweek = 1, startyear = 2001)
```



Several extensions of this simple Bayesian approach are imaginable, for example the inane over-dispersion of the data could be modeled by using a negative-binomial distribution, time trends and mechanisms to correct for past outbreaks could be integrated, but all at the cost of non-standard inference for the predictive distribution. Here simulation based methods like Markov Chain Monte Carlo or heuristic approximations have to be used to obtain the required alarm thresholds.

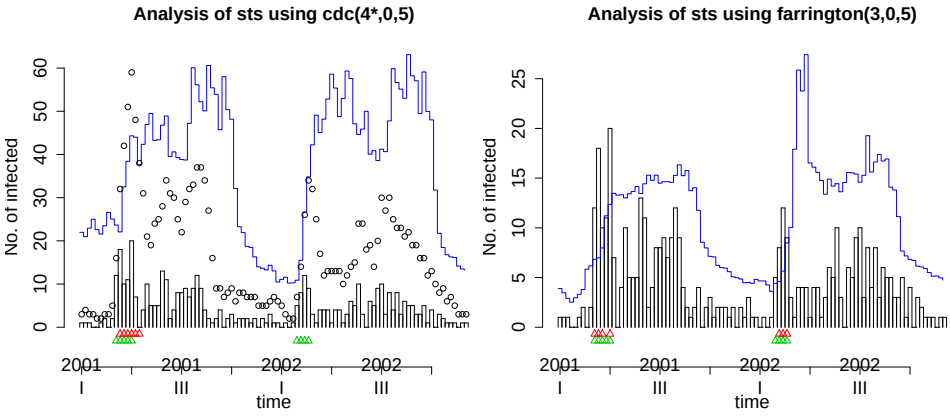
In general, the `surveillance` package makes it easy to add additional algorithms – also those not based on reference values – by using the existing implementations as starting point.

The following call uses the CDC and Farrington procedure on the simulated time series `sts` from page 4. Note that the CDC procedure operates with four-week aggregated data – to better compare the upper bound value, the aggregated number of counts for each week are shown as circles in the plot.

```
> cntrl <- list(range = 300:400, m = 1, w = 3, b = 5, alpha = 0.01)
> sts.cdc <- algo.cdc(sts, control = cntrl)
> sts.farrington <- algo.farrington(sts, control = cntrl)
```

■echo=false,results=hide■ if (compute) ■CDCFAR■

```
> par(mfcol = c(1, 2))
> plot(sts.cdc, legend = F)
> plot(sts.farrington, legend = F)
```



Typically, one is interested in evaluating the performance of the various surveillance algorithms. An easy way is to look at the sensitivity and specificity of the procedure – a correct identification of an outbreak is defined as follows: if the algorithm raises an alarm for time t, i.e. $A_t = 1$ and $X_t = 1$ we have a correct classification, if $A_t = 1$ and $X_t = 0$ we have a false-positive, etc. In case of more involved outbreak models, where an outbreak lasts for more than one week, a correct identification could be if at least one of the outbreak weeks is correctly identified, see e.g. Hutwagner et al. (2005).

To compute various performance scores the function `algo.quality` can be used on a `SurvRes` object.

```
> print(algo.quality(k1.b660))
```

```
     TP FP TN  FN Sens Spec      dist       mlag
[1,] 2  10 154 0  1    0.9390244 0.06097561 0
```

This computes the number of false positives, true negatives, false negatives, the sensitivity and the specificity. Furthermore, `dist` is defined as

$$\sqrt{(Spec-1)^2 + (Sens-1)^2},$$

that is the distance to the optimal point $(1, 1)$, which serves as a heuristic way of combining sensitivity and specificity into a single score. Of course, weighted versions are also imaginable. Finally, `lag` is the average number of weeks between the first of a consecutive number of $X_t = 1$'s (i.e. an outbreak) and the first alarm raised by the algorithm.

To compare the results of several algorithms on a single time series we declare a list of control objects – each containing the name and settings of the algorithm we want to apply to the data.

```
> control = list(list(funcName = "rki1"), list(funcName = "rki2"),
+     list(funcName = "rki3"), list(funcName = "bayes1"),
+     list(funcName = "bayes2"), list(funcName = "bayes3"),
+     list(funcName = "cdc", alpha = 0.05), list(funcName = "farrington",
+         alpha = 0.05))
> control <- lapply(control, function(ctrl) {
+     ctrl$range <- 300:400
+     return(ctrl)
+ })
```

In the above, `rki1`, `rki2` and `rki3` are three methods with reference values $R_{\text{rki1}}(6,6,0)$, $R_{\text{rki2}}(6,6,1)$ and $R_{\text{rki3}}(4,0,2)$ all called with $\alpha = 0.05$. The methods `bayes1-bayes3` is the Bayesian algorithm using the same setup of reference values. The CDC Method is special, since it operates on aggregated four-week blocks. To make everything comparable a common $\alpha = 0.05$ level is used for all algorithms. All algorithms in `control` are applied to `sts` using:

```
> algo.compare(algo.call(sts, control = control))
```

```
                  TP FP TN FN sens      spec      dist       mlag
rki(6,6,0)        6  2  90 3  0.6666667 0.9782609 0.3340415  0
rki(6,6,1)        7  1  91 2  0.7777778 0.9891304 0.2224879  0.5
rki(4,0,2)        8  2  90 1  0.8888889 0.9782609 0.1132178  0.5
bayes(6,6,0)      7  5  87 2  0.7777778 0.9456522 0.2287715  0
bayes(6,6,1)      8  4  88 1  0.8888889 0.9565217 0.1193149  0
bayes(4,0,2)      9  5  87 0  1         0.9456522 0.05434783 0
cdc(4*,0,5)       7  2  90 2  0.7777778 0.9782609 0.223283   1
farrington(3,0,5) 8  1  91 1  0.8888889 0.9891304 0.1116415  0.5
```

A test on a set of time series can be done as follows. Firstly, a list containing 10 simulated time series is created. Secondly, all the algorithms specified in the `control` object are applied to each series. Finally the results for the 10 series are combined in one result matrix.

```
> ten <- lapply(1:10, function(x) {
+     sim.pointSource(p = 0.975, r = 0.5, length = 400,
+         A = 1, alpha = 1, beta = 0, phi = 0, frequency = 1,
+         state = NULL, K = 1.7)
+ })

> ten.surv <- lapply(ten, function(ts) {
+     algo.compare(algo.call(ts, control = control))
+ })

> algo.summary(ten.surv)
```

```
                  TP FP  TN FN sens spec dist mlag
rki(6,6,0)        32 23 940 15 0.68 0.98 0.32 4.27
rki(6,6,1)        34  5 958 13 0.72 0.99 0.28 2.93
rki(4,0,2)        35  4 959 12 0.74 1.00 0.26 1.98
bayes(6,6,0)      40 90 873  7 0.85 0.91 0.18 1.32
bayes(6,6,1)      41 53 910  6 0.87 0.94 0.14 0.72
bayes(4,0,2)      45 42 921  2 0.96 0.96 0.06 0.00
cdc(4*,0,5)       21 34 929 26 0.45 0.96 0.55 8.05
farrington(3,0,5) 35 12 951 12 0.74 0.99 0.26 2.94
```

A similar procedure can be applied when evaluating the 14 surveillance series drawn from SurvStat@RKI (Robert Koch-Institut, 2004). A problem is however, that the series after conversion to 52 weeks/year are of length 209 weeks. This is insufficient to apply e.g. the CDC algorithm. To conduct the comparison on as large a dataset as possible the following trick is used: The function `enlargeData` replicates the requested `range` and inserts it before the original data, after which the evaluation can be done on all 209 values.

```
> range = (2 * 4 * 52) + 1:length(k1$observed)
> control <- lapply(control, function(cntrl) {
+     cntrl$range = range
+     return(cntrl)
+ })
> outbrks <- c("m1", "m2", "m3", "m4", "m5", "q1_nrwh",
+     "q2", "s1", "s2", "s3", "k1", "n1", "n2", "h1_nrwrp")
> outbrks <- lapply(outbrks, function(name) {
+     eval(substitute(data(name), list(name = name)))
+     enlargeData(get(name), range = 1:(4 * 52), times = 2)
+ })
> one.survstat.surv <- function(outbrk) {
+     algo.compare(algo.call(outbrk, control = control))
+ }

> algo.summary(lapply(outbrks, one.survstat.surv))

                  TP FP  TN FN sens spec dist mlag
rki(6,6,0)        32 23 940 15 0.68 0.98 0.32 4.27
rki(6,6,1)        34  5 958 13 0.72 0.99 0.28 2.93
rki(4,0,2)        35  4 959 12 0.74 1.00 0.26 1.98
bayes(6,6,0)      40 90 873  7 0.85 0.91 0.18 1.32
bayes(6,6,1)      41 53 910  6 0.87 0.94 0.14 0.72
bayes(4,0,2)      45 42 921  2 0.96 0.96 0.06 0.00
cdc(4*,0,5)       21 34 929 26 0.45 0.96 0.55 8.05
farrington(3,0,5) 35 12 951 12 0.74 0.99 0.26 2.94
```

In both this study and the earlier simulation study the Bayesian approach seems to do quite well. However, the extent of the comparisons do not make allowance for any more supported statements. Consult the work of Riebler (2004) for a more thorough comparision using simulation studies.

4 Multivariate surveillance

As of version 0.9-2 `surveillance` supports the visualization of multivariate time series of counts. An (multivariate) object of class `disProg` contains matrices with the observed number of counts and the respective state chains, where each column represents an individual time series. Additional elements of the `disProg`-object are a neighbourhood matrix and a matrix with population counts. However, only modelling of the time series as by Held et al. (2005) is currently available. In the near future the surveillance algorithms will also be extended to handle these multivariate data.

For example, consider the weekly counts of new measels cases for each "Kreis" (area) of the administrative district "Weser-Ems" in Lower Saxony, Germany, in 2001 and 2002 (Robert Koch-Institut, 2004). Figure 2 shows a map of the $m = 15$ areas. The corresponding $m \times m$ neighbourhood matrix has elements 1 if two areas share a common border and is 0 otherwise.

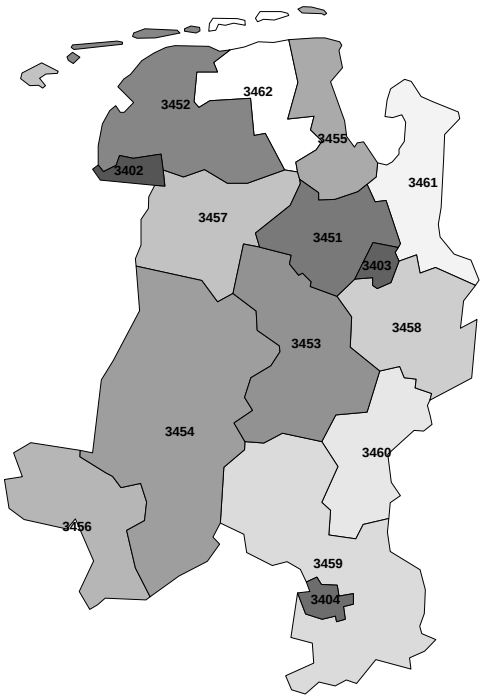
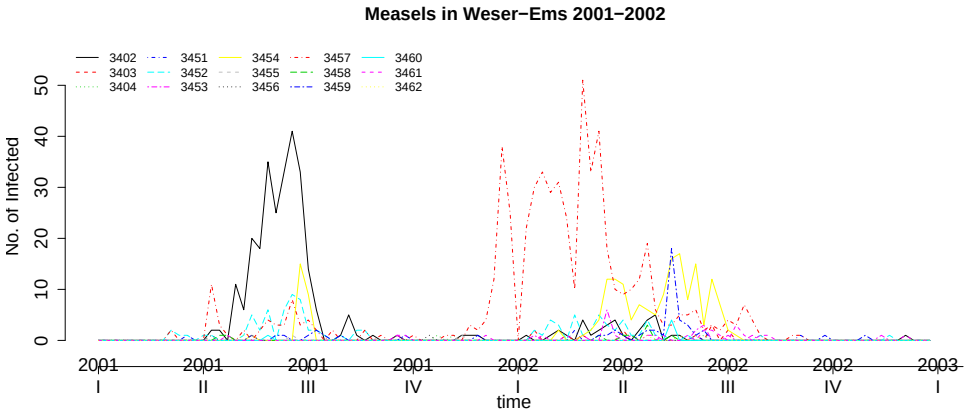


Figure 2: Map of the administrative district "Weser-Ems"

In the package `surveillance` the measels data are already available in

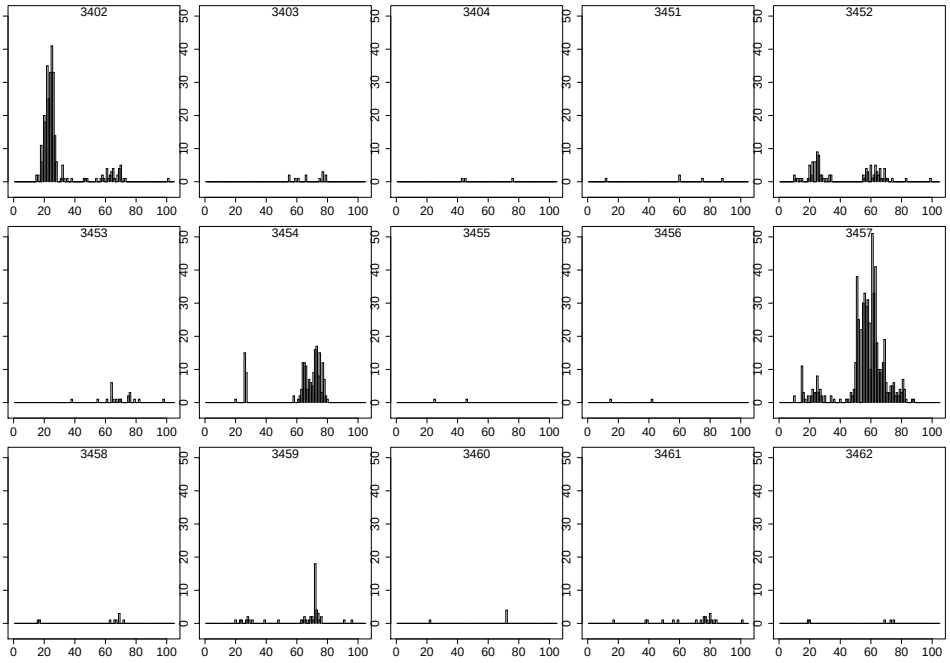
the form of a `disProg`-object.

```
> data(measels.weser)
> plot(measels.weser, title = "Measels in Weser-Ems 2001-2002",
+      xaxis.years = TRUE, startyear = 2001, firstweek = 1)
```



The number of counts for each area can also be looked at and plotted as individual time series. Here, the x-axis is the week number since 1st of January 2001 and the y-axis is the number of measels cases.

```
> plot(measels.weser, as.one = FALSE, xaxis.years = FALSE)
```



The data are analysed using the model proposed by Held et al. (2005). A call to the function `algo.hhh` fits a Poisson or negative binomial model with mean

$$\mu_{it} = \lambda y_{i,t-1} + \phi \sum_{j \sim i} y_{j,t-1} + n_{it}\nu_{it}\,, \quad i = 1, \ldots, m,\, t = 1, \ldots, n\,,$$

where $j \sim i$ denotes all neighbours of i, to a multivariate time series of counts. It is estimated by maximum likelihood using numerical optimization methods. The n_{it} are standardized population counts and $\log \nu_{it} = \alpha_i + \beta t + \sum_{s=1}^{S} \left(\gamma_s sin(\omega_s t) + \delta_s cos(\omega_s t)\right)$ with Fourier frequencies ω_s.

For the weekly measels data $\omega_s = 2s\pi/52$ (i.e. `period`=52). In the following, the model specified in `cntrl` is fitted to the data. The counts are assumed to be negative binomial distributed with mean μ_{it} and variance $\mu_{it} + \mu_{it}^2/\psi$. A linear time trend β, seasonal parameters γ_1 and β_1 (i.e. $S = 1$) as well as the autoregressive parameters λ and ϕ are included to specify the mean. All in all, there are $2S + m + 4$ parameters to be estimated for the negative binomial model. In case of a Poisson model, the number of parameters reduces by one as the overdispersion parameter ψ is omitted.

```
> cntrl <- list(linear = TRUE, nseason = 1, period = 52,
+     neighbours = TRUE, negbin = TRUE, lambda = TRUE)
> measels.hhh <- algo.hhh(measels.weser, control = cntrl)
```

Depending on the inital values for the parameters, the optimization algorithm might not converge or only find a local maximum as the parameter space is high-dimensional. It is therefore reasonable to try multiple starting values.

The function `create.grid` takes a `list` with elements in the form of `param = c(lower,upper,length)` to create a matrix of starting values. For each parameter a sequence of length `length` from `lower` to `upper` is built and the resulting grid contains all combinations of these parameter values. A call to `algo.hhh.grid` conducts a grid search until either all starting values are used or a time limit `maxTime` (in seconds) is exceeded. The result with the highest likelihood is returned.

```
> cntrl <- list(linear = TRUE, nseason = 1, period = 52,
+     neighbours = TRUE, negbin = TRUE, lambda = TRUE)
> grid <- create.grid(list(beta = c(-0.5, 0.5, 3), nseason = 1,
+     gammaDelta = c(-0.5, 0.5, 3), phi = c(0.1, 0.9, 5),
+     psi = c(0.3, 12, 5), lambda = c(0.1, 0.9, 5)))
> algo.hhh.grid(measels.weser, control = cntrl, thetastartMatrix = grid,
+     maxTime = 1800)

size of grid:  1125
grid search stopped after 145 iterations
```

```
Estimated parameters and standard errors:

          lambda     phi   beta gamma1 delta1    psi alpha1 alpha2
Estimates 0.4976 0.00125 0.0120  1.356 -0.508 0.5752  2.681  0.151
Std.Error 0.0738 0.00266 0.0036  0.163  0.134 0.0755  0.270  0.344
          alpha3 alpha4 alpha5 alpha6 alpha7 alpha8 alpha9 alpha10
Estimates -1.282 -0.584  1.375  0.436  0.704  -1.53 -1.469   2.886
Std.Error  0.613  0.682  0.284  0.397  0.305   1.26  0.733   0.220
          alpha11 alpha12 alpha13 alpha14 alpha15
Estimates  -0.180  0.0363  -0.631   0.947 -0.0301
Std.Error   0.424  0.3203   0.492   0.326  0.5724

loglikelihood: -910.3
```

5 Discussion and Future work

Many extensions and additions are imaginable to improve the package. For now, the package is intended as an academic tool providing a test-bench for integrating new surveillance algorithms. Because all algorithms are implemented in R, performance has not been an issue. Especially the current implementation of the Farrington Procedure is rather slow and would benefit from an optimization possible with fragments written in C.

One important improvement would be to provide more involved mechanisms for the simulation of epidemics. In particular it would be interesting to include multi-day outbreaks originating from single-source exposure, but with delay due to varying incubation time (Hutwagner et al., 2005) or SEIR-like epidemics (Andersson and Britton, 2000). However, defining what is meant by a correct outbreak identification, especially in the case of overlapping outbreaks, creates new challenges which have to be met.

6 Acknowledgements

We are grateful to K. Stark and D. Altmann, RKI, Germany, for discussions and information on the surveillance methods used by the RKI. Our thanks to C. Lang, University of Munich, for his work on the R–implementation and M. Kobl, T. Schuster and M. Rossman, University of Munich, for their initial work on gathering the outbreak data from SurvStat@RKI. The research was conducted with financial support from the Collaborative Research Centre SFB 386 funded by the German research foundation (DFG).

References

Altmann, D. (2003). The Surveillance System of the Robert Koch Institute, Germany. Personal Communication.

Andersson, H. and Britton, T. (2000). *Stochastic Epidemic Models and their Statistical Analysis*, volume 151 of *Springer Lectures Notes in Statistics*. Springer-Verlag.

Farrington, C. and Andrews, N. (2003). In Brookmeyer, R. and Stroup, D., editors, *Monitoring the Health of Populations*, chapter Outbreak Detection: Application to Infectious Disease Surveillance, pages 203–231. Oxford University Press.

Farrington, C. P., Andrews, N. J., Beale, A. D., and Catchpole, M. A. (1996). A statistical algorithm for the early detection of outbreaks of infectious disease. *Journal of the Royal Statistical Association. Series A*, 159:547–563.

Held, L., Höhle, M., and Hofmann, M. (2005). A statistical framework for the analysis of multivariate infectious disease surveillance counts. *Statistical Modelling*, 5:187–199.

Hutwagner, L., Browne, T., Seeman, G., and Fleischhauer, A. (2005). Comparing abberation detection methods with simulated data. *Emerging Infectious Diseases*, 11:314–316.

Riebler, A. (2004). Empirischer Vergleich von statistischen Methoden zur Ausbruchserkennung bei Surveillance Daten. Master's thesis, Department of Statistics, University of Munich. Bachelor's thesis.

Robert Koch-Institut (2004). SurvStat@RKI. http://www3.rki.de/SurvStat. Date of query: September 2004.

Robert Koch Institute (2001). Epidemiologisches Bulletin 39. Available from http://www.rki.de.

Stroup, D., Williamson, G., Herndon, J., and Karon, J. (1989). Detection of aberrations in the occurence of notifiable diseases surveillance data. *Statistics in Medicine*, 8:323–329.