The R package surveillance

Michael Höhle[1,2]

[1]Department of Statistics, Ludwig-Maximilians-Universität München, Germany [2]

[2]Munich Center of Health Sciences, University of Munich

Workshop on
Computer supported outbreak detection and signal management
RKI, Berlin, Germany
18 November 2008

Outline

1. Overview
2. Basic surveillance
3. Univariate surveillance
   - Farrington algorithm
   - Cumulative sum
4. Towards multivariate surveillance
5. Summing Up

Monitoring routine collected public health data

- Vast amount of data resulting from public health reporting demands the development of automated algorithms for the detection of abnormalities.
- Aim: statistical analysis of routinely collected surveillance data seen as multiple time series of counts
- Issues such as seasonality, low number of disease cases and presence of past outbreaks complicate the statistical analysis of the time series.

Overview of surveillance

Motivation

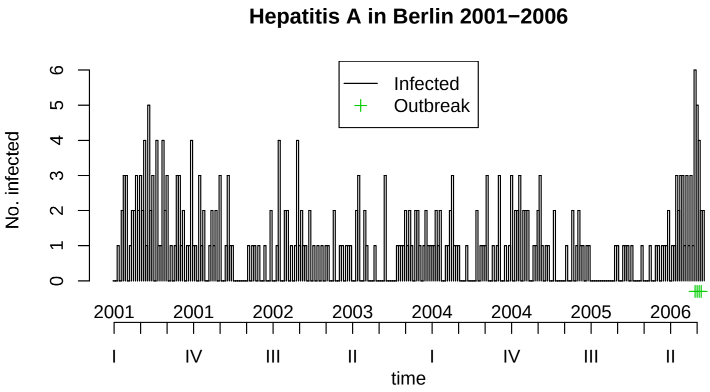
Free software for the *use* and *development* of surveillance algorithms

Features

- Visualization of surveillance data and algorithm output
- Outbreak data from SurvStat@RKI and through simulation from a hidden Markov model
- Implementation of well-known surveillance algorithms
- Functionality to compare classification performance
- Time series models for (multivariate) surveillance data

Example of surveillance data

- Weekly number of adult male hepatitis A cases in the federal state of Berlin during 2001-2006
- During summer 2006 health authorities noticed an increased amount of cases (Robert Koch Institute, 2006).



What is R?

- R is a free software environment for statistical computing and graphics available from `http://www.r-project.org`.
- R runs on a wide variety of UNIX platforms, Windows and Mac OS.
- R is an implementation of the S language (programming language oriented).
- R produces high-quality graphics in a variety of formats, including JPEG, PNG, EPS and PDF.
- R can be combined with Sweave/odfWeave for automatic report generation using LaTeX/OpenOffice.

What is surveillance? (1)

- An open source R package for the visualization and monitoring of count data time series in public health surveillance
- Surveillance algorithms for univariate time series:
  - `cdc` – Stroup et al. (1989)
  - `farrington` – Farrington et al. (1996)
  - `cusum` – Rossi et al. (1999)
  - `rogerson` – Rogerson and Yamada (2004)
  - `lrnb` and `glrnb` – H. and Paul (2008)
- Surveillance time series models:
  - `hhh` - Held et al. (2005); Paul et al. (2008)
  - `twins` - Held et al. (2006) (Experimental)

What is surveillance? (2)

- Comparison of surveillance algorithms using sensitivity, specificity and its variants in simulations
- History: Development started 2004 at the University of Munich as part of the DFG/SFB386 research project "Statistical methodology for infectious disease surveillance"
- Motivation: Provide data structure and framework for methodological developments
- Spinoff: Tool for epidemiologists and others working in applied infectious disease epidemiology
- Availability: CRAN, current development version from

  `http://surveillance.r-forge.r-project.org/`
- Package is available under the GNU General Public License (GPL) v. 2.0.

Data structure: The sts class (1)

- Possible multivariate surveillance time series $\{y_{it}\,; t = 1, \ldots, n, i = 1, \ldots, m\}$ is represented using objects of class `sts` (surveillance time series)
- The `sts` class has the following form

```
setClass( "sts", representation(week = "numeric",
                                freq = "numeric",
                                start = "numeric",
                                observed = "matrix",
                                state = "matrix",
                                alarm = "matrix",
                                upperbound  = "matrix",
                                neighbourhood= "matrix",
                                populationFrac= "matrix",
                                map = "SpatialPolygonsDataFrame",
                                control = "list"))
```

- Old S3 class `disProg` objects can be converted to `sts` using `disProg2sts`.

Data structure: The `sts` class (2)

`observed` A $n \times m$ matrix of counts representing y_{it}

`start` A vector of length two containing the origin of the time series as `c(year, week)`.

`freq` A numeric specifying the period of the time series, i.e. 52 for weekly data, 12 for monthly data, etc.

`state` A $n \times m$ matrix of Booleans, if any specific time points are known to contain outbreaks

`alarm` A $n \times m$ matrix of Booleans containing the result of applying a surveillance algorithm to the time series

`upperbound` A $n \times m$ matrix containing the number of cases which would result in an alarm (specific interpretation is algorithm dependent)

`control` List with control arguments used for the surveillance algorithm

Data I/O

- To import data into R one can use `read.table`/`read.csv`, package `foreign` (SAS, SPSS, Stata, Systat, dBase) or the `RODBC` database interface (Acess, Excel, SQL databases).
- An `sts` object is then created from the resulting matrix of counts.

```
R> ha.counts <- as.matrix(read.csv("ha.csv"))
R> ha <- new("sts", week = 1:nrow(ha.counts), start = c(2001,
+     1), freq = 52, observed = ha.counts, state = matrix(0,
+     nrow(ha.counts), ncol(ha.counts)))
```

- All plotting, accessing, aggregating and application of surveillance algorithms works on `sts` objects

Accessing sts objects (1)

- Printing provides basic information about the time series:

```
R> print(ha)
```

```
-- An object of class sts --
freq:                   52
start:                   2001 1
dim(observed):           290 12

Head of observed:
     chwi frkr lich mahe mitt neuk pank rein span zehl scho trko
[1,]    0    0    0    0    0    0    0    0    0    0    0    0

map:
 [1] chwi frkr lich mahe mitt neuk pank rein scho span trko zehl
12 Levels: chwi frkr lich mahe mitt neuk pank rein scho span ... zehl

head of neighbourhood:
     chwi frkr lich mahe mitt neuk pank rein span zehl scho trko
chwi   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA
```

Accessing sts objects (2)

- Matrix like accessing such as `ha[1:52,]` or `ha[,"mitt"]` results in `sts` objects containing the respective sub time series
- Functions such as `dim`, `nrow` and `ncol` are also defined:

```
R> dim(ha)
[1] 290  12
```

- The time series can be aggregated temporally and spatially:

```
R> dim(aggregate(ha, by = "unit"))
[1] 290   1
R> dim(aggregate(ha, by = "time"))
[1]   1 12
```

- Currently, the slots of `sts` objects are accessed directly

```
R> head(ha@observed, n = 1)
     chwi frkr lich mahe mitt neuk pank rein span zehl scho trko
[1,]    0    0    0    0    0    0    0    0    0    0    0    0
```

Accessing sts objects (3)

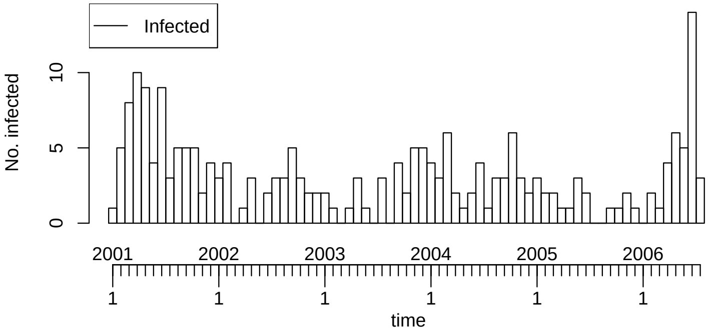
- Aggregation can also be of subsets.
- Example: Aggregate weekly data into 4 week blocks (corresponding to 13 observations per year)

```
R> ha4 <- aggregate(ha[, c("pank", "mitt", "frkr", "scho",
+     "chwi", "neuk")], nfreq = 13)
R> dim(ha4)

[1] 73  6
```

Visualizing sts objects (1)

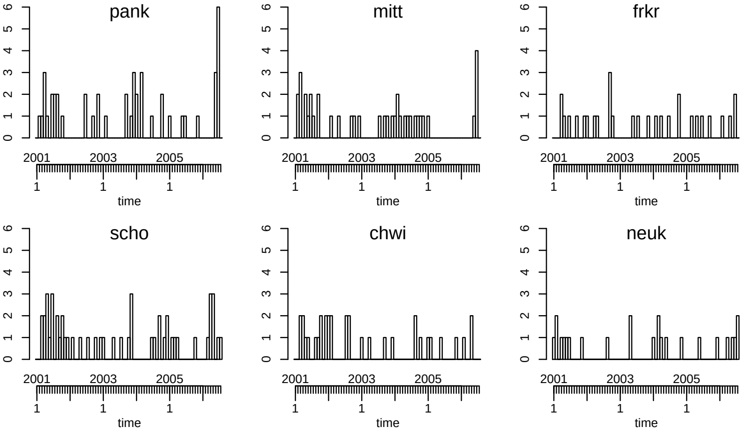
- The `plot` function provides an interface to several visual representations controlled by the `type` argument.

```
R> plot(ha4, type = observed ~ time)
```



Visualizing sts objects (2)

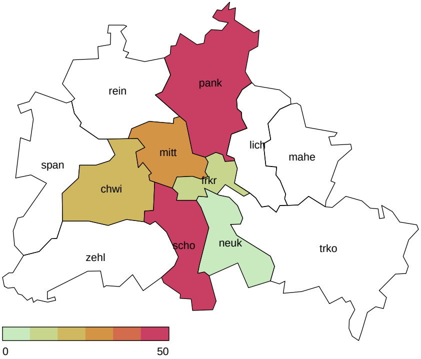
```
R> plot(ha4, type = observed ~ time | unit)
```



Visualizing sts objects (3)

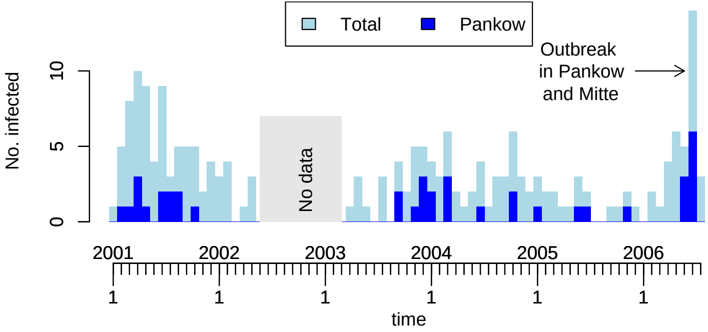
Using the `maptools` package shapefiles provide map visualizations

```
R> plot(ha4, type = observed ~ 1 | unit)
```



Visualizing sts objects (4)

- Using `type = observed~1|time*unit` one would have created an animation of pictures for each time index
- Plotting functionality is customizable as in R-graphics



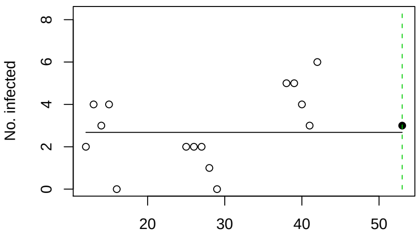
Farrington algorithm (1) – model

- Predict value y_{t_0} at time $t_0 = (t_0^m, t_0^y)$ using a set of reference values from window of size $2w+1$ up to b years back:

$$R(w,b) = \left(\bigcup_{i=1}^{b} \bigcup_{j=-w}^{w} y_{t_0^m + j : t_0^y - i} \right)$$

- Fit overdispersed Poisson GLM to the $b(2w+1)$ reference values where $\mathrm{E}(y_t) = \mu_t$, $\log \mu_t = \alpha + \beta t$ and $\mathrm{Var}(y_t) = \phi \mu_t$.

Prediction at time t=53 with b=3,w=2



Farrington algorithm (2) – outbreak detection

Predict and compare:

- An approximate $(1-\alpha)\%$ prediction interval for y_{t_0} based on the GLM has upper limit $U = \hat{\mu}_{t_0} + z_{1-\frac{\alpha}{2}} \cdot \sqrt{\text{Var}(y_{t_0} - \hat{\mu}_{t_0})}$
- If observed y_{t_0} is greater than U, then flag t_0 as outbreak

Remarks:

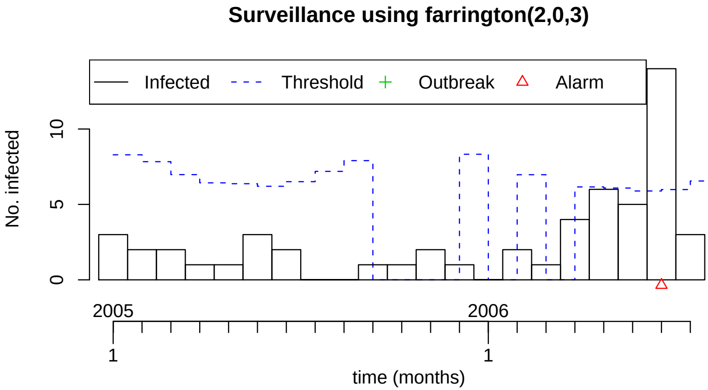
- Linear trend is only included if significant at 5% level, $b \geq 3$ and no over-extrapolation occurs
- Automatic correction for past outbreaks by computing Anscombe residuals for reference values and re-fit GLM assigning lower weights to values with large residuals
- Low count protection – the algorithm raises an alarm only if more than 5 cases in past 4 weeks

Farrington algorithm in surveillance (1)

- Function `farrington` takes an `sts` and a `control` object as arguments
- `control` is a list with the following components:
  - `range` Specifies the index of all timepoints in `sts` to monitor.
  - `b` Number of years to go back in time
  - `w` Window size
  - `reweight` Boolean stating whether to perform reweight step using Anscombe residuals
  - `trend` If `TRUE` a trend is included in first fit and kept in case the conditions are met. Otherwise no trend.
  - `alpha` An approximate two-sided $(1-\alpha)\%$ prediction interval is calculated

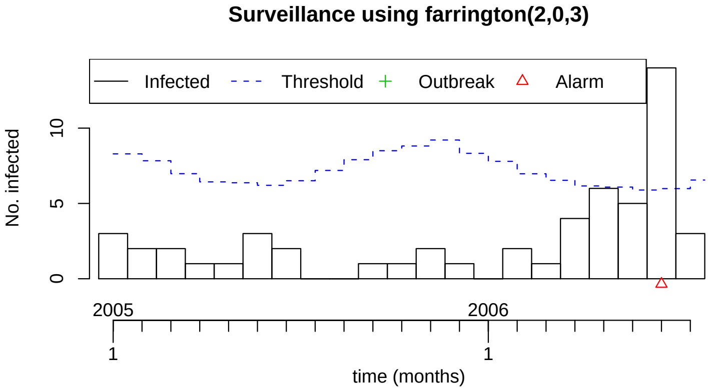
Farrington algorithm in surveillance (2)

```
R> cntrlFar <- list(range = 53:73, w = 2, b = 3, alpha = 0.01)
R> survha <- farrington(ha41, control = cntrlFar)
```



Farrington algorithm in surveillance (3)

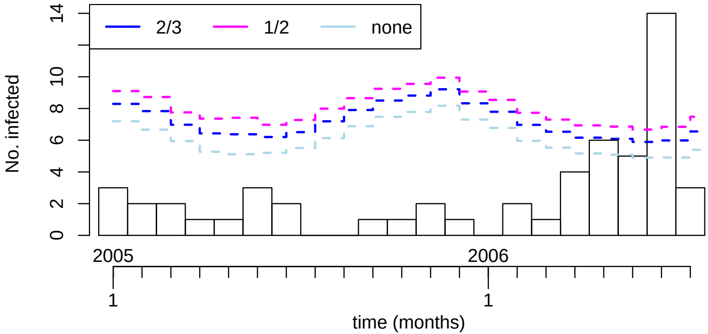
- Argument `limit54=c(cases,weeks)` specifies the low count protection
- Example using `control$limit54=c(0,4)`:



Farrington algorithm in surveillance (4)

- Argument `powertrans` in `control` indicates which power transformation to use:

`"2/3"`	skewness correction in low count scenario
`"1/2"`	variance stabilizing square-root transformation
`"none"`	no transformation



Correcting for past outbreaks (1)

- Problems arise when base-line counts contain outbreaks. A reweighting procedure is used to downweight such observation.
- Compute standardized Anscombe residuals for Poisson distribution:

$$s_t = \frac{r_t}{\hat{\phi}\sqrt{1 - h_{tt}}}, \quad \text{where } r_t = \frac{3(y_t^{\frac{2}{3}} - \hat{\mu}_t^{\frac{2}{3}})}{2\hat{\mu}_t^{\frac{1}{6}}}$$

- Define weights ω_t as

$$\omega_t = \begin{cases} \gamma \frac{1}{s_t^2} & \text{if } s_t > 1 \\ \gamma & \text{otherwise} \end{cases},$$

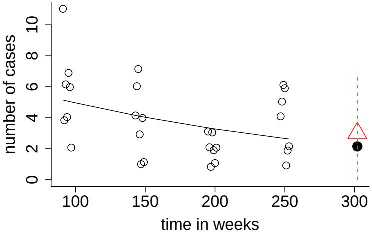
where γ ensures $\sum_{i=1}^{k} \omega_t = n$.

Correcting for past outbreaks (2)

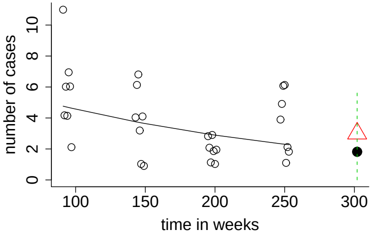
- Refit the GLM using the ω_t weights, i.e.

$$\mathrm{Var}(y_t) = \frac{\phi \mu_t}{\omega_t}$$

- Effect of weights is to downweight large positive outliers in the data:



fit



refit

CUSUM as Surveillance Algorithm (1)

- A control chart known from statistical process control

Cumulative Sum (CUSUM)

In control situation $X_1, \ldots, X_n \overset{\text{iid}}{\sim} N(0,1)$. Monitor shift to $N(\mu, 1)$ by

$$S_t = \max(0, S_{t-1} + X_t - k), \quad t = 1, \ldots, n$$

where $S_0 = 0$ and k is the *reference value*. Raise alarm if $S_t > h$, where h is called the *decision interval*.

- CUSUMs are better to detect sustained shifts
- Given h and k we can determine the *average run length* (ARL)

CUSUM as Surveillance Algorithm (2)

- CUSUM for count data $Y_1, \ldots, Y_n \overset{\text{iid}}{\sim} \text{Po}(m)$ by transforming data to normality (Rossi et al., 1999)

$$X_t = \frac{Y_t - 3m + 2\sqrt{m \cdot Y_t}}{2\sqrt{m}}$$

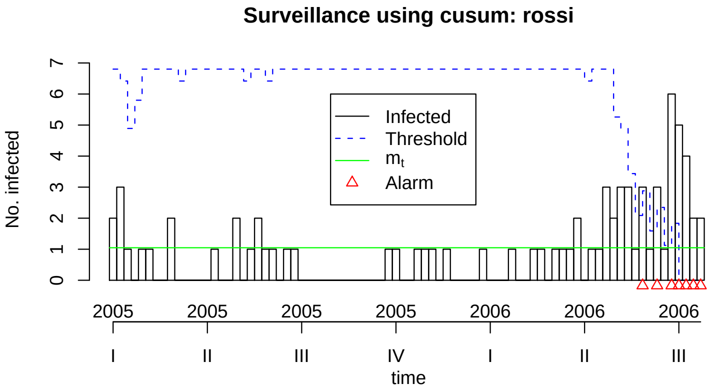
- Risk-adjust the chart by letting m be time varying, e.g. as output of a Poisson GLM model

$$\log(m_t) = \alpha + \beta t + \sum_{s=1}^{S} (\gamma_s \sin(\omega_s t) + \delta_s \cos(\omega_s t)),$$

where $\omega_s = \frac{2\pi}{52} s$ are the Fourier frequencies.

CUSUM as Surveillance Algorithm (3)

```
R> kh <- find.kh(ARLa = 500, ARLr = 7)
R> cntrlRossi <- list(range = 209:290, k = kh$k, h = kh$h,
+     trans = "rossi", m = NULL)
R> ha.cs <- cusum(aggregate(ha, by = "unit"), control = cntrlRossi)
```

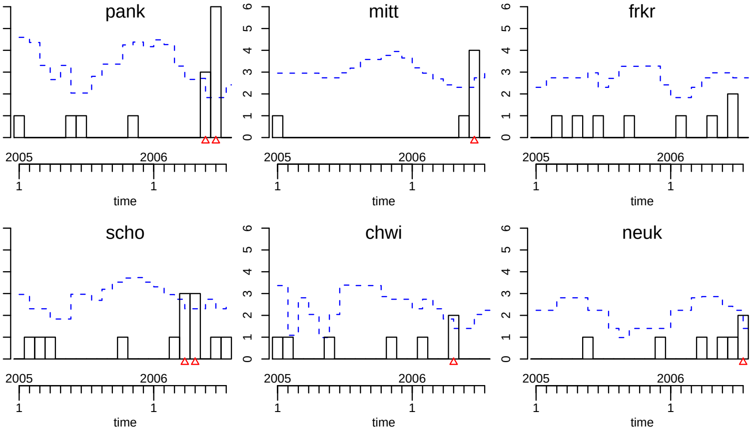


CUSUM as Surveillance Algorithm (3)

- Simulation studies show: For low counts it is better to use CUSUM directly on the counts instead of on transformed residuals
- Proposals for this setting implemented in `surveillance` are:
  - Function `rogerson`, which uses a reweighted Poisson CUSUM (Rogerson and Yamada, 2004)
  - Function `glrnb`, which uses a likelihood ratio and generalized likelihood ratio detector (H. and Paul, 2008)
- More flexibility to model the time series and to tune the detection algorithm → more work for each time series

Towards multivariate surveillance (1)

- A simple way to perform surveillance for a number of time series is to monitor each independently



Towards multivariate surveillance (2)

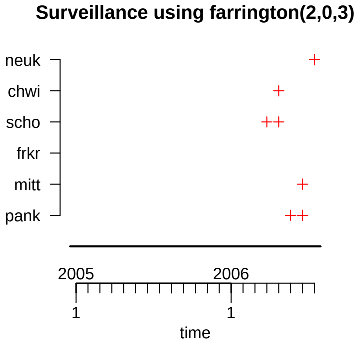
- Results for current month (say August 2006) are easily accessed for further report generation

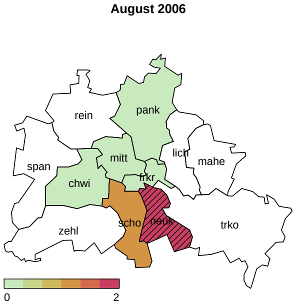
```
R> control <- list(b = 3, w = 2, range = 53:73, alpha = 0.01,
+     limit54 = c(0, 1))
R> ha4.surv <- farrington(ha4, control = control)

R> sapply(c("observed", "upperbound", "alarm"), function(str) {
+     slot(ha4.surv, str)[nrow(ha4.surv), ]
+ })
```

```
     observed upperbound alarm
pank        0       2.42     0
mitt        0       2.97     0
frkr        0       2.74     0
scho        1       2.42     0
chwi        0       2.23     0
neuk        2       1.40     1
```

Towards multivariate surveillance (3)

- An alarm plot gives an overview of alarms for the different time series
- Shaded regions indicate alarms for the current month





Summing Up

- `surveillance` offers an visualization and modeling of surveillance time series and an implementation of different detection algorithms
- A starting point to learn more about the package is H. (2007)
- Functionality for comparing algorithms exists, but was not shown in this talk
- Current work is e.g. an adaption of the algorithms to the binomial setting $y_t \sim \text{Bin}(n_t, \pi_t)$

Acknowledgements

Persons:

- Michaela Paul, Andrea Riebler and Leonhard Held, Institute of Social and Preventive Medicine, University of Zurich, Switzerland
- Valentin Wimmer, Ludwig-Maximilians-Universität München, Germany
- Christoph Staubach, Federal Research Institute for Animal Health, Germany
- Johannes Dreesman, Governmental Institute of Public Health of Lower Saxony, Germany
- Doris Altmann, Robert Koch Institute

Financial Support:

- German Science Foundation (DFG, 2003-2006)

Literature I

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

Held, L., Hofmann, M., Höhle, M., and Schmid, V. (2006). A two component model for counts of infectious diseases. *Biostatistics*, 7:422–437.

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

Höhle, M. (2007). surveillance: An R package for the monitoring of infectious diseases. *Computational Statistics*, 22(4):571–582.

Höhle, M. and Paul, M. (2008). Count data regression charts for the monitoring of surveillance time series. *Computational Statistics and Data Analysis*, 52(9):4357–4368.

Paul, M., Held, L., and Toschke, A. M. (2008). Multivariate modelling of infectious disease surveillance data. *Statistics in Medicine*, 27:6250–6267.

Robert Koch Institute (2006). Epidemiologisches Bulletin 33. Available from http://www.rki.de.

Rogerson, P. and Yamada, I. (2004). Approaches to syndromic surveillance when data consist of small regional counts. *Morbidity and Mortality Weekly Report*, 53:79–85.

Literature II

Rossi, G., Lampugnani, L., and Marchi, M. (1999). An approximate CUSUM procedure for surveillance of health events. *Statistics in Medicine*, 18:2111–2122.

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