R-Package robKalman — Kalman’s revenge or Robustness for Kalman Filtering Revisited

Fraunhofer ITWM
R-Package robKalman —
Kalman’s revenge or robustness for Kalman Filtering Revisited

Peter Ruckdeschel\textsuperscript{1} Bernhard Spangl\textsuperscript{2}

\textsuperscript{1} Fraunhofer ITWM, Kaiserslautern, Germany, Peter.Ruckdeschel@itwm.fraunhofer.de
\textsuperscript{2} Universität für Bodenkultur, Vienna, Austria, Bernhard.Spangl@boku.ac.at

Rennes, July 9, 2009
Euclidean State Space Models

Definitions and Assumptions:
— Time–Discrete, Euclidean Setup ideal model:

\[
\begin{align*}
    x_t &= F(x_{t-1}, t) + v_t, & v_t &\sim (0, Q_t), & [p\text{–dim}], \\
    y_t &= Z(x_t, t) + \varepsilon_t, & \varepsilon_t &\sim (0, V_t), & [q\text{–dim}], \\
    x_0 &\sim (a_0, Q_0), & & [p\text{–dim}], \\
\end{align*}
\]

\{v_t\}, \{\varepsilon_t\}, x_0 \text{ indep. as processes}

functions $F, Z$ smooth with known derivatives;
hyper–parameters $Q_t, V_t, a_0$ known

extensible to:
• continuous time (SDE’s)
• incorporate user–specified controls
Euclidean State Space Models

Definitions and Assumptions:
— Time–Discrete, Euclidean Setup ideal model:

\[
\begin{align*}
x_t &= F(x_{t-1}, t) + v_t, \\
y_t &= Z(x_t, t) + \varepsilon_t,
\end{align*}
\]

\( v_t \overset{\text{indep.}}{\sim} (0, Q_t), \) \quad \text{[p–dim]},

\( \varepsilon_t \overset{\text{indep.}}{\sim} (0, V_t), \) \quad \text{[q–dim]},

\( x_0 \overset{\text{indep.}}{\sim} (a_0, Q_0), \) \quad \text{[p–dim]},

\{v_t\}, \{\varepsilon_t\}, x_0 \text{ indep. as processes}

functions \( F, Z \) smooth with known derivatives;
hyper–parameters \( Q_t, V_t, a_0 \) known

extensible to:
• continuous time (SDE’s)
• incorporate user-specified controls
Euclidean State Space Models

Definitions and Assumptions:
— Time–Discrete, Euclidean Setup ideal model:

\[
\begin{align*}
x_t &= F(x_{t-1}, t) + v_t, & v_t \overset{\text{indep.}}{\sim} (0, Q_t), \quad [p\text{–dim}], \\
y_t &= Z(x_t, t) + \varepsilon_t, & \varepsilon_t \overset{\text{indep.}}{\sim} (0, V_t), \quad [q\text{–dim}], \\
x_0 &\sim (a_0, Q_0), \quad [p\text{–dim}],
\end{align*}
\]

\{v_t\}, \{\varepsilon_t\}, x_0 \text{ indep. as processes}

functions \( F, Z \) smooth with known derivatives;
hyper–parameters \( Q_t, V_t, a_0 \) known

extensible to:
• continuous time (SDE’s)
• incorporate user-specified controls
Types of Outliers

exogenous outliers affecting only singular observations

\[
\begin{align*}
AO & :: \quad \varepsilon_t^{re} \sim (1 - r_{AO}) \mathcal{L}(\varepsilon_t^{id}) + r_{AO} \mathcal{L}(\varepsilon_t^{di}) \\
SO & :: \quad y_t^{re} \sim (1 - r_{SO}) \mathcal{L}(y_t^{id}) + r_{SO} \mathcal{L}(y_t^{di})
\end{align*}
\]

endogenous outliers / structural changes

\[
\begin{align*}
IO & :: \quad v_t^{re} \sim (1 - r_{IO}) \mathcal{L}(v_t^{id}) + r_{IO} \mathcal{L}(v_t^{di})
\end{align*}
\]

Different and competing goals

A/SO attenuation of "false alarms"

IO tracking: detect structural changes as fast as possible;
recovering: clean data from structural changes

A/SO & IO identification problem:
simultaneous treatment only possible with delay
Types of Outliers
exogenous outliers affecting only singular observations

\[
\begin{align*}
\text{AO} & : \quad \varepsilon_t^\text{re} \sim (1 - r_{\text{AO}}) \mathcal{L}(\varepsilon_t^\text{id}) + r_{\text{AO}} \mathcal{L}(\varepsilon_t^\text{di}) \\
\text{SO} & : \quad y_t^\text{re} \sim (1 - r_{\text{SO}}) \mathcal{L}(y_t^\text{id}) + r_{\text{SO}} \mathcal{L}(y_t^\text{di})
\end{align*}
\]
endogenous outliers / structural changes

\[
\begin{align*}
\text{IO} & : \quad v_t^\text{re} \sim (1 - r_{\text{IO}}) \mathcal{L}(v_t^\text{id}) + r_{\text{IO}} \mathcal{L}(v_t^\text{di})
\end{align*}
\]

Different and competing goals

A/SO \quad \text{attenuation of “false alarms”}

IO \quad \text{tracking: detect structural changes as fast as possible;}
\quad \text{recovering: clean data from structural changes}

A/SO & IO \quad \text{identification problem:}
\quad \text{simultaneous treatment only possible with delay}
Classical Method: Kalman–Filter

Filter Problem

\[ E \left| x_t - f_t(y_{1:t}) \right|^2 = \min_{f_t} \],

with \( y_{1:t} = (y_1, \ldots, y_t) \), \( y_{1:0} := \emptyset \)

General solution: \( E[x_t | y_{1:t}] \) — difficult to compute

Kalman–Filter assuming \( F(x, t) = F_t x \), \( Z(x, t) = Z_t x \)
optimal solution among linear filters — Kalman[Bucy] [60/61]:

Initialization: \( x_{0|0} = a_0 \)

Prediction: \( x_{t|t-1} = F_t x_{t-1|t-1} \), \( [\Delta x_t = x_t - x_{t|t-1}] \)

Correction: \( x_{t|t} = x_{t|t-1} + M_{t}^0 \Delta y_t \), \( [\Delta y_t = y_t - Z_t x_{t|t-1}] \)

and corresponding recursions for the prediction/filtering error
covariances \( \Sigma_{t|t-1} \) and the Kalman gain \( M_{t}^0 \)
Classical Method: Kalman–Filter

Filter Problem

$$E \left| x_t - f_t(y_{1:t}) \right|^2 = \min_{f_t}$$,

with $$y_{1:t} = (y_1, \ldots, y_t)$$, $$y_{1:0} := \emptyset$$

General solution: $$E[x_t|y_{1:t}]$$ — difficult to compute

Kalman–Filter assuming $$F(x, t) = F_t x$$, $$Z(x, t) = Z_t x$$

optimal solution among linear filters — Kalman/Bucy [60/61]:

Initialization: $$x_0|0 = a_0$$

Prediction: $$x_{t|t-1} = F_t x_{t-1|t-1}$$, $$[\Delta x_t = x_t - x_{t|t-1}]$$

Correction: $$x_{t|t} = x_{t|t-1} + M^0_t \Delta y_t$$, $$[\Delta y_t = y_t - Z_t x_{t|t-1}]$$

and corresponding recursions for the prediction/filtering error covariances $$\Sigma_{t|t-1}$$ and the Kalman gain $$M^0_t$$
Features of the Kalman–Filter

+ an easy, understandable structure:
  initialization, prediction, correction step
+ correction step is easily evaluable and interpretable: it is linear!
+ strict recursivity / Markovian structure:
  all information from the past useful for the future is captured in
  the value of $x_{t|t-1}$.
  - the correction step is linear and thus not robust, as $y$ enters
    unbounded;

Aim of robustification: try to retain all “+”‘s, revise “−”
Features of the Kalman–Filter

+ an easy, understandable structure: initialization, prediction, correction step
+ correction step is easily evaluable and interpretable: it is linear!
+ strict recursivity / Markovian structure:
  all information from the past useful for the future is captured in the value of $x_{t|t-1}$.
  
  - the correction step is linear and thus not robust, as $y$ enters unbounded;

Aim of robustification: try to retain all “+”'s, revise “−”
**R-package robKalman — Contents**

- Kalman filter: filter, Kalman gain, covariances
- ACM-filter: filter, multivariate version, GM-estimator
- rLS-filter: filter, calibration of clipping height
  - AO/SO-robust version
  - IO-robust version
  - with a certain delay joint treatment of AO/SO’s & IO’s
- extensible to further recursive filters:
  - general interface `recursiveFilter`
  - with arguments:
    - data
    - state space model (hyper parameters)
      [will be: object of class SSM]
    - functions for the init./pred./corr.step
      [will be: object containing them]
    - [will be: control object]
R-package robKalman — Contents

- Kalman filter: filter, Kalman gain, covariances
- ACM-filter: filter, multivariate version, GM-estimator
- rLS-filter: filter, calibration of clipping height
  - AO/SO-robust version
  - IO-robust version
  - with a certain delay joint treatment of AO/SO’s & IO’s
- extensible to further recursive filters:
  - general interface recursiveFilter
    with arguments:
    - data
    - state space model (hyper parameters)
      [will be: object of class SSM]
    - functions for the init./pred./corr.step
      [will be: object containing them]
    - [will be: control object]
R-package robKalman — Contents

- Kalman filter: filter, Kalman gain, covariances
- ACM-filter: filter, multivariate version, GM-estimator
- rLS-filter: filter, calibration of clipping height
  - AO/SO-robust version
  - IO-robust version
  - with a certain delay joint treatment of AO/SO’s & IO’s
- extensible to further recursive filters:
  - general interface recursiveFilter
  with arguments:
  - data
  - state space model (hyper parameters)
    [will be: object of class SSM]
  - functions for the init./pred./corr.step
    [will be: object containing them]
  - [will be: control object]
Implementation concept

• Programming language
  – completely in S, perhaps some code in C later (→ FKF)
• Use existing infrastructure: zoo, timeSeries
  – for: graphics, diagnostics, management of date/time
• Code in different layers
  – internal functions: no S4-objects, no time stamps
    (helps bringing in code by “non-S4-people”)
  – user interface: S4-objects, time stamps
• Use generating functions for encapsulation
  – without using structured arguments:
    ★ too many arguments → user looses track
    ★ prone to name mis-matchings (positional, partial matching)
    ★ bad alternative: fix defaults...
  – have generating functions to produce control objects
  – control objects may be reused
Implementation concept

- Programming language
  - completely in \( S \), perhaps some code in \( C \) later (\( \rightsquigarrow \) FKF)
- Use existing infrastructure: zoo, timeSeries
  - for: graphics, diagnostics, management of date/time
- Code in different layers
  - internal functions: no S4-objects, no time stamps
    (helps bringing in code by “non-S4-people”)
  - user interface: S4-objects, time stamps
- Use generating functions for encapsulation
  - without using structured arguments:
    ★ too many arguments \( \rightsquigarrow \) user looses track
    ★ prone to name mis-matchings (positional, partial matching)
    ★ bad alternative: fix defaults...
  - have generating functions to produce control objects
  - control objects may be reused
Implementation concept

- **Programming language**
  - completely in S, perhaps some code in C later (⇒ FKF)
- **Use existing infrastructure:** zoo, timeSeries
  - for: graphics, diagnostics, management of date/time
- **Code in different layers**
  - internal functions: no S4-objects, no time stamps
    (helps bringing in code by “non-S4-people”)
  - user interface: S4-objects, time stamps
- **Use generating functions for encapsulation**
  - without using structured arguments:
    - too many arguments ⇒ user loses track
    - prone to name mis-matching (positional, partial matching)
    - bad alternative: fix defaults...
  - have generating functions to produce control objects
  - control objects may be reused
Implementation concept

- Programming language
  - completely in \texttt{S}, perhaps some code in \texttt{C} later (\(\rightarrow\) FKF)
- Use existing infrastructure: \texttt{zoo, timeSeries}
  - for: graphics, diagnostics, management of date/time
- Code in different layers
  - internal functions: no \texttt{S4}-objects, no time stamps
    (helps bringing in code by “non-S4-people”)
  - user interface: \texttt{S4}-objects, time stamps
- Use generating functions for encapsulation
  - without using structured arguments:
    \begin{itemize}
      \item too many arguments \(\rightarrow\) user looses track
      \item prone to name mis-matchings (positional, partial matching)
      \item bad alternative: fix defaults. . .
    \end{itemize}
  - have generating functions to produce control objects
  - control objects may be reused
Implementation so far

Interfaces so far

• preliminary, “S4-free” interfaces
  – Kalman filter (in our context) `KalmanFilter`
  – rLS: `rLSFilter`, `rLS.AO.Filter`, `rLS.IO.Filter`, `rLS.IOAO.Filter`
  – ACM: `ACMfilt`, `ACMfilter`, `mACMfilter`
  – all realized as wrappers to `recursiveFilter`

• availability: `robKalman` version 0.3 (incl. demos)
  [http://r-forge.r-project.org/projects/robkalman/](http://r-forge.r-project.org/projects/robkalman/)

Almost ready:

• S4 classes: for SSM’s; for output-classes; for method-classes; for control-classes (reuse code)
Implementation so far

Interfaces so far

- preliminary, “S4-free” interfaces
  - Kalman filter (in our context) KalmanFilter
  - rLS: rLSFilter (=rLS.AO.Filter), rLS.IO.Filter, rLS.IOAO.Filter
  - ACM: ACMfilt, ACMfilter, mACMfilter
  - all realized as wrappers to recursiveFilter

- availability: robKalman version 0.3 (incl. demos)

Almost ready:

- S4 classes: for SSM’s; for output-classes; for method-classes; for control-classes (reuse robustbase-code)
- interfaces between S4-layer and S4-free layer
to other SSM packages

(http://r-forge.r-project.org/projects/robkalman/)
Implementation so far

Interfaces so far

• preliminary, “S4-free” interfaces
  – Kalman filter (in our context) KalmanFilter
  – rLS: rLSFilter (=rLS.AO.Filter), rLS.IO.Filter, rLS.IOAO.Filter
  – ACM: ACMfilt, ACMfilter, mACMfilter
  – all realized as wrappers to recursiveFilter

• availability: robKalman version 0.3 (incl. demos)

http://r-forge.r-project.org/projects/robkalman/

Almost ready:

• S4 classes: for SSM’s; for output-classes; for method-classes; for control-classes (reuse robustbase-code)
• interfaces between S4-layer and S4-free layer to other SSM packages to robfilter (Roland Fried & K. Schettlinter)
Implementation so far

Interfaces so far

• preliminary, “S4-free” interfaces
  – Kalman filter (in our context) KalmanFilter
  – rLS: rLSFilter (=rLS.AO.Filter), rLS.IO.Filter, rLS.IOAO.Filter
  – ACM: ACMfilt, ACMfilter, mACMfilter
  – all realized as wrappers to recursiveFilter

• availability: robKalman version 0.3 (incl. demos)

http://r-forge.r-project.org/projects/robkalman/

Almost ready:

• S4 classes: for SSM’s; for output-classes; for method-classes; for control-classes (reuse robustbase-code)

• interfaces between S4-layer and S4-free layer
to other SSM packages
to robfilter (Roland Fried & K. Schettlinger)
Implementation so far

Interfaces so far

• preliminary, “S4-free” interfaces
  – Kalman filter (in our context) KalmanFilter
  – rLS: rLSFilter (≡rLS.AO.Filter), rLS.IO.Filter, rLS.IOAO.Filter
  – ACM: ACMfilt, ACMfilter, mACMfilter
  – all realized as wrappers to recursiveFilter

• availability: robKalman version 0.3 (incl. demos)
  http://r-forge.r-project.org/projects/robkalman/

Almost ready:

• S4 classes: for SSM’s; for output-classes; for method-classes;
  for control-classes (reuse robustbase-code)
• interfaces between S4-layer and S4-free layer
  to other SSM packages
to robfilter (Roland Fried & K. Schettlinger)
Work in process

Release Plans

• package robKalman should be on CRAN by UseR! 2009, but...
• at least: release on CRAN by end of August
• till then: refer to r-forge

Extensions

• robust smoothing (80% done)
• robust EM-Algorithm to estimate unknown hyper parameters (extending Shumway/Stoffer) (70% done)
• interpretation as random coefficient regression
  \[\rightsquigarrow\] robust regression-type approach (rIC, mIC) (30% done)
• connection to particle filters —
  theory and computer interface (10% done)
• speeding up things / bridging to fast Kalman filter of FKF by David Luethi, Philipp Erb (1% done)
Work in process

Release Plans

• package robKalman should be on CRAN by UseR! 2009, but...
• at least: release on CRAN by end of August
• till then: refer to r-forge

Extensions

• robust smoothing (80% done)
• robust EM-Algorithm to estimate unknown hyper parameters (extending Shumway/Stoffer) (70% done)
• interpretation as random coefficient regression → robust regression-type approach (rIC, mLIC) (30% done)
• connection to particle filters — theory and computer interface (10% done)
• speeding up things / bridging to fast Kalman filter of FKF by David Luethi, Philipp Erb (1% done)
Some experiences on collaborative programming on r-forge

• r-forge: very neat for collaborative R package development
  – version management (svn)
  – mail-forwarded log-files of committed code
    ~→ keep track of work of others
  – bug tracker, archived mailing lists, . . .
  – see slides by Stefan Theussl
• needs serious conceptional preparations
  – for separating/modularizing tasks
  – consistency: coding & documentation conventions
• helpful: scheduling, reminders/deadlines for collaborators . . .
• summarizing:

  Collaborative programming is enjoyable and very exciting!

  THANKS FOR YOUR ATTENTION!
Some experiences on collaborative programming on r-forge

- r-forge: **very neat** for collaborative R package development
  - version management (svn)
  - mail-forwarded log-files of committed code
    -> keep track of work of others
  - bug tracker, archived mailing lists, . . .
  - see slides by Stefan Theussl
- needs serious conceptional preparations
  - for separating/modularizing tasks
  - consistency: coding & documentation conventions
- helpful: scheduling, reminders/deadlines for collaborators . . .
- summarizing:
  
  Collaborative programming is enjoyable and very exciting!

THANKS FOR YOUR ATTENTION!
Some experiences on collaborative programming on r-forge

- **r-forge:**
  - very neat for collaborative R package development
  - version management (`svn`)
  - mail-forwarded log-files of committed code
    ~ keep track of work of others
  - bug tracker, archived mailing lists, . . .
  - see slides by Stefan Theussl
- needs serious conceptional preparations
  - for separating/modularizing tasks
  - consistency: coding & documentation conventions
- helpful: scheduling, reminders/deadlines for collaborators. . .
- summarizing:

  *Collaborative programming is enjoyable and very exciting!*

  **THANKS FOR YOUR ATTENTION!**
Some experiences on collaborative programming on r-forge

• **r-forge:**
  - very neat for collaborative R package development
  - version management (svn)
  - mail-forwarded log-files of committed code
    ~ keep track of work of others
  - bug tracker, archived mailing lists, ...
  - see slides by Stefan Theussl

• needs serious conceptional preparations
  - for separating/modularizing tasks
  - consistency: coding & documentation conventions

• helpful: scheduling, reminders/deadlines for collaborators...

• summarizing:

  *Collaborative programming is enjoyable and very exciting!*

  **THANKS FOR YOUR ATTENTION!**
References


References (cont.)

http://www.R-project.org


