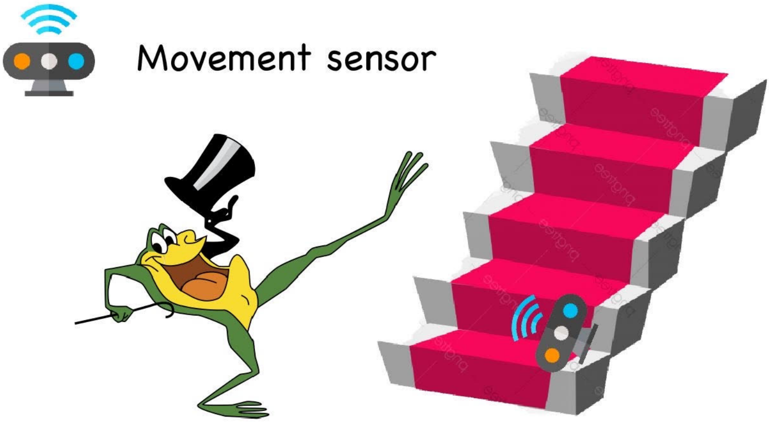
# Hidden Markov Chains

Marco, Nicolás, Sebastián and César

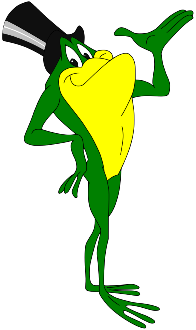
October 31, 2019

# Index

# Example: Hidden Mark-frog Chains



# Example: Hidden Mark-frog Chains



- Our frog is jumping on a ladder with $K = 6$ levels.
- From position $X_t$ level at which the frog is at time $t$ is **not observed**.
- Frog's detector at the lowest level of the ladder sends a signal $Y_t$

$$Y_t = 1 \; non - detection$$

$$Y_t = 2 \; detection$$

- Observations

$$y_{1:14} = (1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 2, 2, 1, 2)$$

# Example: Hidden Mark-frog Chains

**Transition matrix** $p = 0.4$

$$A_{1,2} = 1-p, A_{K,1} = \frac{1-p}{2}$$

$$A_{i,i+1} = \frac{1-p}{2} \text{ for } i = 1,\dots,K-1$$

$$A_{i,i} = p \text{ for } i = 1,\dots,K$$

$$A_{i,i-1} = \frac{1-p}{2} \text{ for } i = 2,\dots,K$$

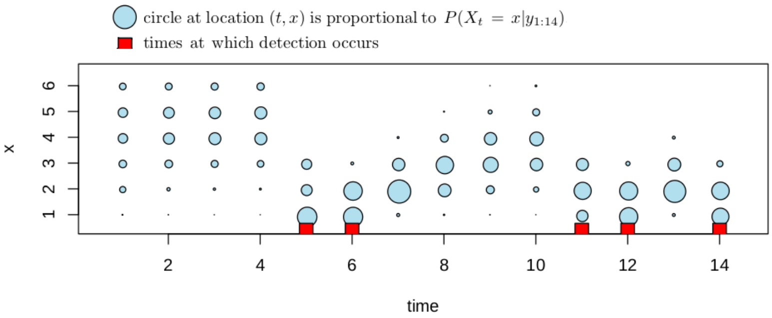
**Probability of detection**

$$P(Y_t = 2 | X_t = k)$$

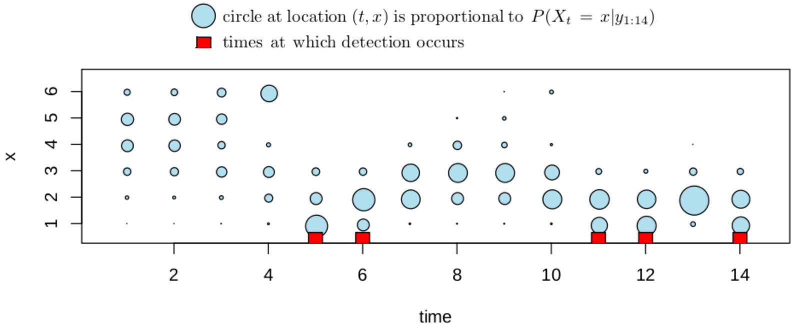
$$= \begin{cases} 0.9 & \text{if } k = 1 \\ 0.5 & \text{if } k = 1 \\ 0.1 & \text{if } k = 2 \\ 0 & \textit{otherwise} \end{cases}$$

**Problems:** From frog's observations of position at each time $t$, infer 1) filtering and 2) smoothing and 3) MAP estimate of pmf's.

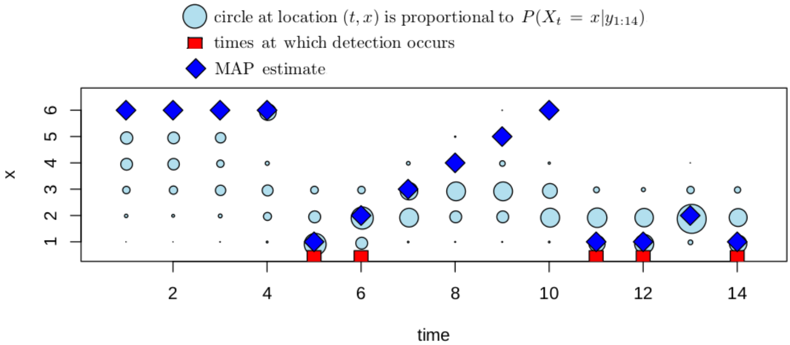
# Example: Hidden Mark-frog Chains - Filtering pmf



# Example: Hidden Mark-frog Chains - Smoothing pmf



# Example: Hidden Mark-frog Chains - MAP pmf




**Note:** Image shows Smoothing pmf over time $t$ and MAP estimate

# Smoothing

**General problem of smoothing:**
**Given:** observations up to time s (i.e. $y_{1:s}$ known),
**Problem:** compute $p(x_t|y_{1:s})$ distribution of $X_t$, for $t < s$.

A particular case is when we know all history of observations:

### Smoothing with all history of observations

**Given:** observations up to time T (i.e. $y_{1:T}$ known),
**Problem:** compute $p(x_t|y_{1:T})$ distribution of $X_t$, for $t < T$.

This case can be solved using a recursive algorithm **Forward-backward Smoothing**.

# Forward-backward Smoothing I

**Ideas:**

- $p(x_t|y_{1:T})$ can be split as:

$$p(x_t|y_{1:T}) = \frac{p(x_t, y_{1:t}) \cdot p(y_{t+1:T}|x_t)}{p(y_{1:T})}$$

- $p(y_{1:T})$ is a normalization constant (calculation explained in next section).
- Algorithm 1 Forward $\alpha$-recursion can estimate $\alpha_t(x_t) = p(x_t, y_{1:t})$.
- We can approach HHM structure to generate a recursion rule for

$$\beta_t(x_t) := p(y_{t+1:T}|x_t)$$

# Forward-backward Smoothing II

$$
\begin{aligned}
p(y_{t+1:T}|x_{t-1}) &= \sum_{x_t \in \mathcal{X}} p(y_{t+1:T}, x_t|x_{t-1}) \\
&= \sum_{x_t \in \mathcal{X}} p(y_t|y_{t+1:T}, x_t, x_{t-1}|x_{t-1}) p(y_{t+1:T}, x_t|x_{t-1}) \\
&= \sum_{x_t \in \mathcal{X}} p(y_t|x_{t-1}, x_t) p(y_{t+1:T}|x_t) \\
&= \sum_{x_t \in \mathcal{X}} p(y_t|x_t) p(y_{t+1:T}|x_t) p(y_t|x_{t-1})
\end{aligned}
$$

We have a backward recursion for $t = T, \ldots, 2$

$$
\beta_{t-1}(x_{t-1}) = \sum_{x_t \in \mathcal{X}} p(y_t|x_t) p(x_t|x_{t-1}) \beta_t(x_t), \;\; \beta_T(x_T) = 1
$$

# Forward-backward Smoothing III

If $\mathcal{X} = \{1, \ldots, K\}$:

**Algorithm 2** Backward $\beta$-recursion

- For $i = 1, \ldots, K$, set $\beta_T(i) = 1$
- For $t = 1, \ldots, T$
    - For $i = 1, \ldots, K$, set

$$\beta_{t-1}(i) = \sum_{j=1}^{K} g_j(y_t) A_{i,j} \beta_t(j)$$

Figure: Backward $\beta$-recursion Algorithm

# Forward-backward Smoothing III

**Backward and Forward Algorithm for smoothing:**

- Algorithm 1 Forward $\alpha$-recursion to get $\alpha_t(x_t) = p(x_t, y_{1:t})$.
- Algorithm 2 Backward $\beta$-recursion to get $\beta_t(x_t) = p(y_{t+1:T}|x_t)$.
- Smoothing probability mass function is estimated as:

$$p(x_t|y_{1:T}) = \frac{p(x_t, y_{1:T})}{p(y_{1:T})} = \frac{\alpha_t(x_t)\beta_t(x_t)}{\sum_{x_t \in \mathcal{X}} \alpha_t(x_t)\beta_t(x_t)}$$

**Notes:** 1) Algorithms 1 y 2 can be run independently, 2) Backward and Forward Algorithm is from order $\mathcal{O}(T \cdot |\mathcal{X}|^2)$

# Likelihood

**Given:** all observations (i.e. $y_{1:T}$ known),
**Problem:** compute $p(y_{1:T})$ likelihood function of observations.

**Ideas:**

- Notice
$$p(y_{1:T}) = \sum_{x_t \in \mathcal{X}} p(x_t, y_{1:t}) \cdot p(y_{t+1:T}|x_t)$$
- This implies
$$p(y_{1:T}) = \sum_{x_t \in \mathcal{X}} \alpha_t(x_t)\beta_t(x_t)$$
- $p(y_{1:T})$ can be estimated using algorithm 1 Forward $\alpha$-recursion and algorithm 2 Backward $\beta$-recursion.

# Most likely state path/Decoding

**Given:** history of observations (i.e. $y_{1:T}$ known),
**Problem:** Find the most likely state history $x_{0:T}$.

For fixed $y_{1:T}$, solve MAP problem with **conditional distribution**:

$$\hat{x}_0 = argmax_{x_{0:T}} \; p(x_{0:T} | y_{1:T})$$

Or equivalently with **joint distribution**:

$$\hat{x}_0 = argmax_{x_{0:T}} \; p(x_{0:T}, y_{1:T})$$

**Note:** unfeasible if number of different state paths $|\mathcal{X}|^{T+1}$ is large from optimization point of view!!.

# Most likely state path - Viterbi algorithm

However, we can use **Viterbi algorithm** to do MAP estimation.
**Ideas:**

- Explote structure of HMM do estimations based on backward-forward or forward-backward recursions.
- Compute messages from time $t$ to $t-1$ $(m_{t-1}(x_{t-1}) \leftarrow m_t(x_t))$
- Use messages to estimate feasible points $t-1$ to $t$ $(\hat{x}_{t-1} \rightarrow \hat{x}_t)$

$$
\begin{array}{lccccccccc}
\text{Backward} & m_0(x_0) & \longleftarrow & m_1(x_1) & \longleftarrow & \dots & \longleftarrow & m_{T-1}(x_{T-1}) & \longleftarrow & m_T(x_T) \\
 & \downarrow & & \downarrow & & & & \downarrow & & \downarrow \\
\text{Forward} & \widehat{x}_0 & \longrightarrow & \widehat{x}_1 & \longrightarrow & \dots & \longrightarrow & \widehat{x}_{T-1} & \longrightarrow & \widehat{x}_T
\end{array}
$$

Figure: Scheme of Vitterbi algorithm recursions

# What do you mean by messages $m_t(x_t)$? Part I

Factorizing $p(x_{0:T}, y_{1:T})$:

$$p(x_{0:T}, y_{1:T}) = p(x_0) \prod_{t=1}^{T} p(x_t|x_{t-1})p(y_t|x_t)$$

We can split our optimization problem in different optimization problems:

$$\Rightarrow \max_{x_0:x_T} p(x_{0:T}, y_{1:T}) =$$

$$\max_{x_0:x_{T-1}} \left\{ \left\{ p(x_0) \prod_{t=1}^{T-1} p(x_t|x_{t-1})p(y_t|x_t) \right\} \max_{x_T} p(x_T|x_{T-1})p(y_T|x_T) \right\}$$

$$= \max_{x_0:x_{T-1}} \left\{ \left\{ p(x_0) \prod_{t=1}^{T-1} p(x_t|x_{t-1})p(y_t|x_t) \right\} m_{T-1}(x_{T-1}) \right\}$$

# What do you mean by messages $m_t(x_t)$? Part II

Continuing this process we can define a set of messages based on iterative optimization problems:

$$m_{t-1}(x_{t-1}) = \max_{x_{t:T}} \left\{ \prod_{k=t}^{T} p(x_k|x_{k-1})p(y_k|x_k) \right\} \text{ for } t = T-1, \ldots, 1$$

$$m_T(x_T) = 1$$

This satisfies the following recursion:

$$m_{t-1}(x_{t-1}) = \max_{x_t} p(y_t|x_t)p(x_t|x_{t-1})m_t(x_t)$$

# What do you mean by messages $m_t(x_t)$? Part III

Definitions of messages is good for our problem because:

$$p(x_0)m_0(x_0) = \max_{x_{1:T}} p(x_{0:T}, y_{1:T})$$

$$\Rightarrow \hat{x}_0 = arg \max_{x_0} \left( \max_{x_{1:T}} p(x_{0:T}, y_{1:T}) \right) = arg \max_{x_0} m_0(x_0)p(x_0)$$

And also

$$\hat{x}_t = arg \max_{x_t} p(\hat{x}_{0:t-1}, x_t, x_{t+1:T}, y_{1:T})$$

$$= arg \max_{x_t} p(\hat{x}_{t-1}, x_t, x_{t+1:T}, y_{1:T})$$

$$= arg \max_{x_t} (m_t(x_t)p(y_t|x_t)p(x_t|\hat{x}_{t-1}))$$

# Most likely state path - Viterbi algorithm

**Algorithm 3** Viterbi algorithm for maximum a posteriori estimation

- For $i = 1, \ldots, K$, set $m_T(i) = 1$.
- For $t = T, \ldots, 1$
  - For $i = 1, \ldots, K$, let

$$m_{t-1}(i) = \max_{j=1,\ldots,K} g_j(y_t) A_{i,j} m_t(j)$$

- Set $\hat{x}_0 = \arg\max_{i=1\ldots,K} \; m_0(i)\mu(i)$
- For $t = 1, \ldots, T$
  - Set

$$\hat{x}_t = \arg\max_{i=1\ldots,K} \; m_t(i) g_i(y_t) A_{\hat{x}_{t-1},i}.$$

Figure: Vitterbi algorithm

**Notes:** 1) Viterbi algorithm has order $\mathcal{O}(T|\mathcal{X}|^2)$, 2) In practice logarithms are computed to assure numerical stability.

# Continuous-state Hidden Markov Models

In many problems often hidden parameters of interest are continuous.

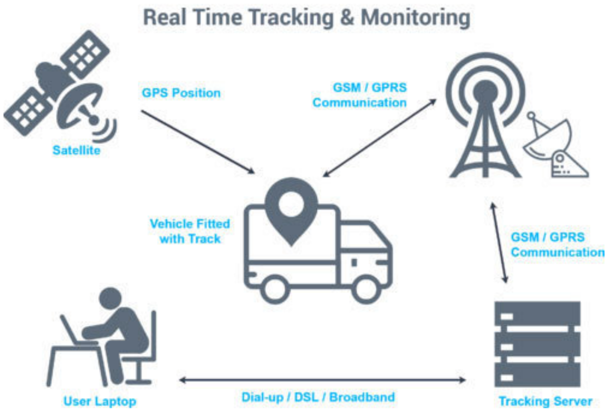



Figure: GPS object tracking; hidden parameters: position + velocity

# Continuous-state Hidden Markov Models

- Ideas developed for discrete case can be generalized to aboard problems with hidden continuous parameters of interest.
- We call these **Continuous-state Hidden Markov Models** (CS-HMM), also named as state-space models or dynamical systems.
- Let's focus on a particular subset of CS-HMM: **linear Gaussian state-space model** (LGSSM).

# Linear Gaussian state-space model (LGSSM)

**Hidden states:** $(X_0, \ldots, X_T)$ continuous r.v. taking values in $\mathbb{R}^{d_x}$
**Observations:** $(Y_1, \ldots, Y_T)$ continuous r.v. taking values in $\mathbb{R}^{d_y}$

$(X_0, \ldots, X_T, Y_1, \ldots, Y_T)$ is LGSSM if has two components such as:

- **state model:** $X_t$ is a linear transformation of $X_{t-1}$ plus a linear combination of Gaussian noise.

- **observation model:** $Y_t$ is a linear transformation of $X_t$ plus Gaussian noise.

# Linear Gaussian state-space model (LGSSM)

## Definition (LGSSM - State model)

$$X_t = F_t X_{t-1} + G_t V_t \textit{ for } t = 1, \ldots, T \textit{ State model} \tag{1}$$

- $X_t \in \mathbb{R}^{d_x}$ hidden state at time $t$,
- $F_t \in \mathbb{R}^{d_x \times d_x}$ transition state matrix at time $t$,
- $G_t \in \mathbb{R}^{d_x \times d_v}$ noise transfer matrix,
- $V_t \in \mathbb{R}^{d_v}$ state noise matrix, $V_t \sim \mathcal{N}(0, Q_t)$.

# Linear Gaussian state-space model (LGSSM)

## Definition (Observation model)

$$Y_t = H_t X_t + W_t \ \textit{for } t = 1, \dots, T \qquad \textit{Observation model} \qquad (2)$$

- $Y_t \in \mathbb{R}^{d_y}$ observation at time $t$,
- $H_t \in \mathbb{R}^{d_x \times d_x}$ observation matrix at time $t$,
- $W_t \in \mathbb{R}^{d_v}$ observation noise, $W_t \sim \mathcal{N}(0, R_t)$.

## Definition (Other conditions)

- $X_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$,
- $V_t \sim \mathcal{N}(0, Q_t), W_t \sim \mathcal{N}(0, R_t)$, *for* $t = 1, \dots T$,
- $X_0, V_1, \dots V_T, W_1, \dots W_T$ *are independent.*

# LGSSM and parametrization of joint pdf

As in discrete case, joint pdf of hidden variables and observation can be described as:

$$p(X_{0:T}, Y_{1:T}) = \prod_{t=1}^{T} p(y_t|x_t) \cdot p(x_{t-1}|x_t) \tag{3}$$

Using **LGSSM structure** and **properties of multivariate normal distributions** it can be shown that if $G_t Q_T G_T \in \mathbb{R}^{d_x \times d_x}$ has full rank then:

$$p(y_t|x_t) = \mathcal{N}(H_t x_t, R_t) \tag{4}$$

$$p(x_t|x_{t-1}) = \mathcal{N}(F_t x_{t-1}, G_t Q_T G_T) \tag{5}$$

# Inference in dynamic LGSSM

Let's consider the following inference problem:
**P1:** Given **observations up to time** $t$ find $Y_1 = y_1, \dots, Y_t = y_t$, find joint pdf $p(x_t|y_{1:t})$.

**P2:** Given **all observations** $Y_1 = y_1, \dots, Y_T = y_T$, find joint pdf $p(x_t|y_{1:T})$

Both can be solve by **sequentially computation of means and covariance matrices of conditionally distributions**: 1) P1 is called **Kallman filter**, and in 2) P2 is named **Kallman smoother**.

# Inference LGSSM - Kallman filter

**P1:** Determine $p(x_t|y_{1:t})$ of the hidden state $X_t$ given $t$ observations $y_{1:t}$.

$$\mu_{t|t-1} := E[X_t|Y_{1:t} = y_{1:t-1}] \tag{6}$$

$$\mu_{t|t} := E[X_t|Y_{1:t} = y_{1:t}] \tag{7}$$

$$\Sigma_{t|t-1} := E[(X_t - \mu_{t|t-1})(X_t - \mu_{t|t-1})^T|Y_{1:t} = y_{1:t-1}] \tag{8}$$

$$\Sigma_{t|t} := E[(X_t - \mu_{t|t})(X_t - \mu_{t|t})^T|Y_{1:t} = y_{1:t}] \tag{9}$$

# Kallman filter ideas - Prediction

Structure of LGSSM implies $p(x_t|y_{1:t-1}) = \mathcal{N}(\mu_{t|t-1}, \Sigma_{t|t-1})$ and the existence of recursive rules:

**Prediction** $(\mu_{t-1|t-1} \to \mu_{t|t-1})$ and $(\Sigma_{t-1|t-1} \to \Sigma_{t|t-1})$:

- $\mu_{t|t-1} = F_t \mu_{t-1|t-1}$,
- $\Sigma_{t|t-1} = F_t \Sigma_{t-1|t-1} F_t^T + G_t Q_t G_t^T$

# Kallman filter ideas - Update/correction

Again, structure of LGSSM implies $p(x_t|y_{1:t}) = \mathcal{N}(\mu_{t|t}, \Sigma_{t|t})$ and following recursion rule:

**Update/correction** $(\mu_{t|t-1} \to \mu_{t|t})$ and $(\Sigma_{t|t-1} \to \Sigma_{t|t})$:

- $\mu_{t|t} = \mu_{t|t-1} + K_t \nu_t$,
- $\Sigma_{t|t-1} = (I - K_t H_t)\Sigma_{t-1|t-1}$

Where $\nu_t = y_t - \hat{y}_t$, $\hat{y}_t = E[Y_t|Y_{1:t-1} = y_{1:t-1}] = H_t \mu_{t|t-1}$
$K_t = \Sigma_{t|t-1} H^T S_t^{-1}$,
$S_t = E[(Y_t - \hat{y}_t)(Y_t - \hat{y}_t)^T | Y_{1:t-1} = y_{1:t-1}] = H_t \Sigma_{t|t-1} H_t^T + R_t$
$K_t$ is called **Kallman gain**

# Kallman filter ideas

**Recursive strategy of Kallman filter**

$p(x_t|y_{1:t})$ can be determined estimating parameters for $s \in \mathbb{N}$ of:

- $p(x_s|y_{1:s-1}) = \mathcal{N}(\mu_{s|s-1}, \Sigma_{s|s-1})$
- $p(x_s|y_{1:s}) = \mathcal{N}(\mu_{s|s}, \Sigma_{s|s})$

as follows:

$$(\mu_0, \Sigma_0) \xrightarrow{\text{Predict.}} (\mu_{1|0}, \Sigma_{1|0}) \xrightarrow{\text{Update}} \dots$$

$$\dots \xrightarrow{\text{Predict.}} (\mu_{t-1|t-1}, \Sigma_{t-1|t-1}) \xrightarrow{\text{Update}}$$

$$(\mu_{t|t-1}, \Sigma_{t|t-1}) \xrightarrow{\text{Predict.}} (\mu_{t|t}, \Sigma_{t|t}) \rightarrow \dots$$

# Inference LGSSM - Kallman Smoother

**P2:** Determine $p(x_t|y_{1:T})$ of the hidden state $X_t$ given all the observations $y_{1:T}$.

$$\mu_{t|T} := E[X_t|Y_{1:T} = y_{1:T}] \tag{10}$$

$$\Sigma_{t|T} := E[(X_t - \mu_{t|T})(X_t - \mu_{t|T})^T|Y_{1:T} = t_{1:T}] \tag{11}$$

# Kallman Smoother ideas

Structure of LGSSM implies $p(x_t|y_{1:T}) = \mathcal{N}(\mu_{t|T}, \Sigma_{t|T})$ and the existence of recursive rules:

**Backward recursion**:

- $\mu_{t|T} = \mu_{t|t} + J_t(\mu_{t+1|T} - \mu_{t+1|t})$,
- $\Sigma_{t|T} = \Sigma_{t|t} + J_t(\Sigma_{t+1|T} - \Sigma_{t+1|t})J_t^T$

Where $J_t = \Sigma_{t|t} F_{t+1}^T \Sigma_{t+1|t}^{-1}$, is called **Backwards Kallman gain**.

**Recursive strategy of Kallman Smoother**

$p(x_t|y_{1:T})$ can be determined estimating parameters as follows:

- Compute $(\mu_{t|t}, \Sigma_{t|t})$ and $(\mu_{t+1|t}, \Sigma_{t+1|t})$ for $t, t+1 \leq T$ using Kallman filter.
- Use backward recursion until obtain $(\mu_{t|T}, \Sigma_{t|T})$.

# A example for LGSSM

Let's consider a toy example for a random walk given by:

$$X_t = X_{t-1} + V_t$$

$$Y_t = X_t + W_t$$

Where

- $X_t$: trigonometric function plus gaussian noise, for $t = 1, \dots, 50$.
- $X_0 = \mathcal{N}(0, 1)$, $V_t \sim \mathcal{N}(0, Q)$, $W_t \sim \mathcal{N}(0, R)$, with $Q = 0.02$ and $R = 0.2$

# Filtering

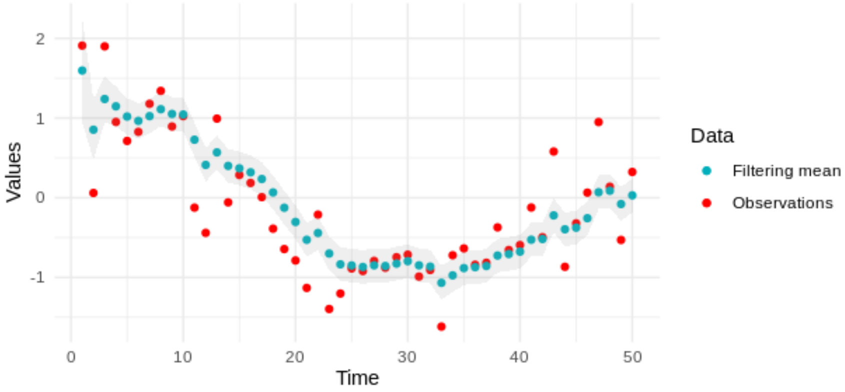



Figure: Observations and filtering mean and 99% credible intervals over time.

# Smoothing

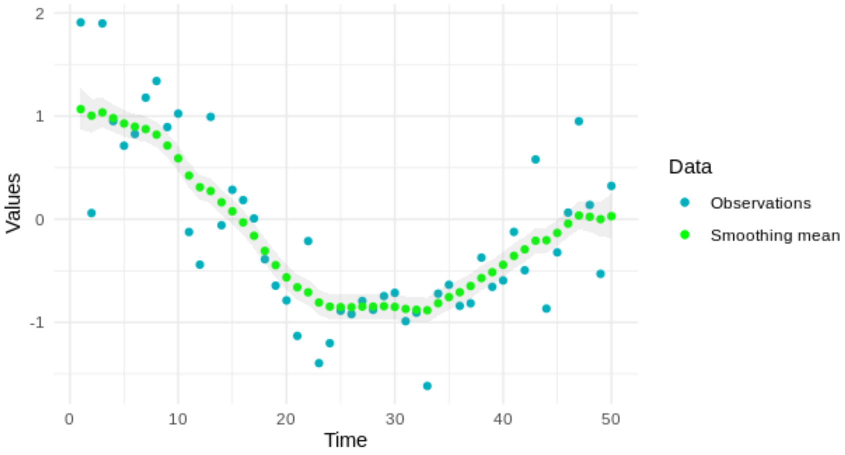



Figure: Observations and smoothing mean and 99% credible intervals over time