

# Package: phenoCDM (via r-universe)

September 10, 2024

**Title** Continuous Development Models for Incremental Time-Series Analysis

**Version** 0.1.3

**Date** 2018-05-01

**Author** Bijan Seyednasrollah, Jennifer J. Swenson, Jean-Christophe Domec, James S. Clark

**Maintainer** Bijan Seyednasrollah <[bijan.s.nasr@gmail.com](mailto:bijan.s.nasr@gmail.com)>

**Description** Using the Bayesian state-space approach, we developed a continuous development model to quantify dynamic incremental changes in the response variable. While the model was originally developed for daily changes in forest green-up, the model can be used to predict any similar process. The CDM can capture both timing and rate of nonlinear processes. Unlike static methods, which aggregate variations into a single metric, our dynamic model tracks the changing impacts over time. The CDM accommodates nonlinear responses to variation in predictors, which changes throughout development.

**Depends** R (>= 3.3.0)

**Imports** rjags

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.0.1.9000

**BugReports** <https://github.com/bnasr/phenoCDM/issues>

**Suggests** knitr, rmarkdown

**VignetteBuilder** knitr

**NeedsCompilation** no

**Date/Publication** 2018-05-02 03:45:15 UTC

**Repository** <https://bbcrown.r-universe.dev>

**RemoteUrl** <https://github.com/cran/phenoCDM>

**RemoteRef** HEAD

**RemoteSha** bb6ad3eba4afd33623b2091f3d256bdc98784ebf

## Contents

fitCDM . . . . .	2
getGibbsSummary . . . . .	3
phenoSim . . . . .	4
phenoSimPlot . . . . .	5
plotPOGibbs . . . . .	6
plotPost . . . . .	8
<b>Index</b>	<b>10</b>

---

fitCDM	<i>Fit a CDM Model</i>
--------	------------------------

---

## Description

This function fits a CDM model on the input data as it is described by the phenoSim function.

## Usage

```
fitCDM(x, z, connect = NULL, nGibbs = 1000, nBurnin = 1, n.adapt = 100,
       n.chains = 4, quiet = FALSE, calcLatentGibbs = FALSE, trend = +1)
```

## Arguments

x	Matrix of predictors [N x p].
z	Vector of response values [N x 1].
connect	The connectivity matrix for the z vector [n x 2]. Each row contains the last and next elements of the time-series. NA values indicate not connected.
nGibbs	Number of MCMC iterations
nBurnin	Number of burn-in iterations.
n.adapt	Number of iterations for adaptive sampling
n.chains	Number of MCMC chains
quiet	logical value indicating whether to report the progress
calcLatentGibbs	logical value indicating whether to calculate the latent states
trend	time-series expected trend as -1:decreasing, +1:increasing, 0: not constrained

**Examples**

```

#Summarize CDM Model Ouput

ssSim <- phenoSim(nSites = 2, #number of sites
                 nTSet = 30, #number of Time steps
                 beta = c(1, 2), #beta coefficients
                 sig = .01, #process error
                 tau = .1, #observation error
                 plotFlag = TRUE, #whether plot the data or not
                 miss = 0.05, #fraction of missing data
                 ymax = c(6, 3) #maximum of saturation trajectory
                 )

ssOut <- fitCDM(x = ssSim$x, #predictors
               nGibbs = 200,
               nBurnin = 100,
               z = ssSim$z, #response
               connect = ssSim$connect, #connectivity of time data
               quiet=TRUE)

summ <- getGibbsSummary(ssOut, burnin = 100, sigmaPerSeason = FALSE)

colMeans(summ$ymax)
colMeans(summ$betas)
colMeans(summ$tau)
colMeans(summ$sigma)

```

---

getGibbsSummary

*Summarize Output of the CDM Model*


---

**Description**

This function return a summary of the output from the Gibbs-Sampling of the CDM model.

**Usage**

```

getGibbsSummary(ssOut, burnin = NULL, colNames = NULL,
                sigmaPerSeason = TRUE)

```

**Arguments**

ssOut	CDM output list.
burnin	Number of burnin itterations .
colNames	vector of charachters includes names of each variable in the output.
sigmaPerSeason	logical value indicating whether each site/season has a separate process error

**Examples**

```
#Summarize CDM Model Output

ssSim <- phenoSim(nSites = 2, #number of sites
                 nTSet = 30, #number of Time steps
                 beta = c(1, 2), #beta coefficients
                 sig = .01, #process error
                 tau = .1, #observation error
                 plotFlag = TRUE, #whether plot the data or not
                 miss = 0.05, #fraction of missing data
                 ymax = c(6, 3) #maximum of saturation trajectory
                )

ssOut <- fitCDM(x = ssSim$x, #predictors
               nGibbs = 200,
               nBurnin = 100,
               z = ssSim$z, #response
               connect = ssSim$connect, #connectivity of time data
               quiet=TRUE)

summ <- getGibbsSummary(ssOut, burnin = 100, sigmaPerSeason = FALSE)

colMeans(summ$ymax)
colMeans(summ$betas)
colMeans(summ$tau)
colMeans(summ$sigma)
```

---

 phenoSim

---

*Simulate Green-up Phenology Data*


---

**Description**

This function return a set of simulated data for multiple green-up phenology time-series.

**Usage**

```
phenoSim(nSites = 1000, nTSet = c(3:6), p = 2, beta = NULL, sig = 0.1,
        tau = 0.01, miss = 0, plotFlag = FALSE, ymax = 1, trend = +1)
```

**Arguments**

nSites	Number of sites/seasons
nTSet	A vector of integer values. Length of each time-series will be randomly sampled from this vector.
p	Number of predictors in the model.
beta	Beta coefficients
sig	Process error.

tau	Observation error.
miss	Fraction of missing data.
plotFlag	logical value indicating whether to plot the resulted time-series.
ymax	Asymptotic maximum values.
trend	time-series expected trend as -1:decreasing, +1:increasing, 0: not constrained

### Examples

```
#Simulate Phenology Data
ssSim <- phenoSim(nSites = 2, #number of sites
                 nTSet = 30, #number of time steps
                 beta = c(1, 2), #beta coefficients
                 sig = .01, #process error
                 tau = .1, #observation error
                 plotFlag = TRUE, #whether plot the data or not
                 miss = 0.05, #fraction of missing data
                 ymax = c(6, 3) #maximum of saturation trajectory
)
```

---

phenoSimPlot                      *Plot Simulated Phenology Data*

---

### Description

This function plots the time-series data described with a connectivity matrix.

### Usage

```
phenoSimPlot(z, connect, add = FALSE, col = "blue", ylim = range(z, na.rm
= TRUE), pch = 1, lwd = 1)
```

### Arguments

z	A vector of time-series data [n x 1]
connect	The connectivity matrix for the z vector [n x 2]. Each row contains the last and next elements of the time-series. NA values means not connected.
add	logical value indicating whether the plot should be overlaid on the current panel.
col	The color variable as character
ylim	Range of the y axis
pch	pch value for the symbols
lwd	lwd value for line tickness

**Examples**

```

#Simulate Phenology Data
ssSim <- phenoSim(nSites = 2, #number of sites
                 nTSet = 30, #number of time steps
                 beta = c(1, 2), #beta coefficients
                 sig = .01, #process error
                 tau = .1, #observation error
                 plotFlag = TRUE, #whether plot the data or not
                 miss = 0.05, #fraction of missing data
                 ymax = c(6, 3) #maximum of saturation trajectory
)

#Plot Simulated Data
phenoSimPlot(ssSim$z, ssSim$connect)

```

---

plotPOGibbs

*Plot Observed vs Predicted*


---

**Description**

This function plot posterior distributions of the parameters.

**Usage**

```

plotPOGibbs(o, p, nburnin = NULL, xlim = range(o, na.rm = TRUE),
            ylim = range(p, na.rm = TRUE), xlab = "Observed", ylab = "Predicted",
            colSet = c("#fb8072", "#80b1d3", "black"), cex = 1, lwd = 2, pch = 19)

```

**Arguments**

o	Observed vector
p	Predicted Gibbs samples
nburnin	numbe of burn-in itterations
xlim	x-axis range
ylim	y-axis range
xlab	x-axis label
ylab	y-axis label
colSet	vector of colors for points, bars and the 1:1 line
cex	cex value for size
lwd	line width
pch	pch value for symbols

**Examples**

```

ssSim <- phenoSim(nSites = 2, #number of sites
                 nTSet = 30, #number of Time steps
                 beta = c(1, 2), #beta coefficients
                 sig = .01, #process error
                 tau = .1, #observation error
                 plotFlag = TRUE, #whether plot the data or not
                 miss = 0.05, #fraction of missing data
                 ymax = c(6, 3) #maximum of saturation trajectory
)

ssOut <- fitCDM(x = ssSim$x, #predictors
               nGibbs = 200,
               nBurnin = 100,
               z = ssSim$z, #response
               connect = ssSim$connect, #connectivity of time data
               quiet=TRUE)

summ <- getGibbsSummary(ssOut, burnin = 100, sigmaPerSeason = FALSE)

colMeans(summ$ymax)
colMeans(summ$betas)
colMeans(summ$tau)
colMeans(summ$sigma)

par(mfrow = c(1,3), oma = c(1,1,3,1), mar=c(2,2,0,1), font.axis=2)

plotPost(chains = ssOut$chains[,c("beta.1", "beta.2")], trueValues = ssSim$beta)
plotPost(chains = ssOut$chains[,c("ymax.1", "ymax.2")], trueValues = ssSim$ymax)
plotPost(chains = ssOut$chains[,c("sigma", "tau")], trueValues = c(ssSim$sig, ssSim$tau))

mtext('Posterior distributions of the parameters', side = 3, outer = TRUE, line = 1, font = 2)
legend('topleft', legend = c('posterior', 'true value'),
      col = c('black', 'red'), lty = 1, bty = 'n', cex=1.5, lwd =2)

yGibbs <- ssOut$latentGibbs
zGibbs <- ssOut$zpred
o <- ssOut$data$z
p <- apply(ssOut$rawsamples$y, 1, mean)
R2 <- cor(na.omit(cbind(o, p)))[1,2]^2
#Plot Observed vs Predicted
par( mar=c(4,4,1,1), font.axis=2)
plotPOGibbs(o = o , p = zGibbs,
            xlim = c(0,10), ylim=c(0,10),
            cex = .7, nburnin = 1000)
points(o, p, pch = 3)

mtext(paste0('R2 = ', signif(R2, 3)), line = -1, cex = 2, font = 2, side = 1, adj = .9)
legend('topleft', legend = c('mean', '95th percentile', '1:1 line', 'latent states'),
      col = c('#fb8072', '#80b1d3', 'black', 'black'),
      bty = 'n', cex=1.5,

```

```
lty = c(NA, 1, 2, NA), lwd = c(NA, 2, 2, 2), pch = c(16, NA, NA, 3))
```

---

plotPost

*Plot Posterior Distributions*

---

### Description

This function plot posterior distributions of the parameters.

### Usage

```
plotPost(chains, trueValues = NULL, outline = FALSE)
```

### Arguments

chains	Gibbs sampling chains
trueValues	numeric vector of true values
outline	logical value whether showing outliers

### Examples

```
ssSim <- phenoSim(nSites = 2, #number of sites
                 nTSet = 30, #number of Time steps
                 beta = c(1, 2), #beta coefficients
                 sig = .01, #process error
                 tau = .1, #observation error
                 plotFlag = TRUE, #whether plot the data or not
                 miss = 0.05, #fraction of missing data
                 ymax = c(6, 3) #maximum of saturation trajectory
)

ssOut <- fitCDM(x = ssSim$x, #predictors
               nGibbs = 200,
               nBurnin = 100,
               z = ssSim$z, #response
               connect = ssSim$connect, #connectivity of time data
               quiet=TRUE)

summ <- getGibbsSummary(ssOut, burnin = 100, sigmaPerSeason = FALSE)

colMeans(summ$ymax)
colMeans(summ$betas)
colMeans(summ$tau)
colMeans(summ$sigma)

par(mfrow = c(1,3), oma = c(1,1,3,1), mar=c(2,2,0,1), font.axis=2)

plotPost(chains = ssOut$chains[,c("beta.1", "beta.2")], trueValues = ssSim$beta)
```



```
plotPost(chains = ssOut$chains[,c("ymax.1", "ymax.2")], trueValues = ssSim$ymax)
plotPost(chains = ssOut$chains[,c("sigma", "tau")], trueValues = c(ssSim$sig, ssSim$tau))

mtext('Posterior distributions of the parameters', side = 3, outer = TRUE, line = 1, font = 2)
legend('topleft', legend = c('posterior', 'true value'), col = c('black', 'red'),
      lty = 1, bty = 'n', cex=1.5, lwd =2)
```

# Index

- \* **CDM**
    - fitCDM, 2
  - \* **Data**
    - phenoSim, 4
    - phenoSimPlot, 5
  - \* **Distributions**
    - plotPost, 8
  - \* **Fit**
    - fitCDM, 2
  - \* **Gibbs**
    - getGibbsSummary, 3
  - \* **Model**
    - fitCDM, 2
  - \* **Observed**
    - plotPOGibbs, 6
  - \* **Output**
    - getGibbsSummary, 3
  - \* **Phenology**
    - phenoSim, 4
    - phenoSimPlot, 5
  - \* **Plot**
    - phenoSimPlot, 5
    - plotPOGibbs, 6
    - plotPost, 8
  - \* **Posterior**
    - plotPost, 8
  - \* **Predicted**
    - plotPOGibbs, 6
  - \* **Summary**
    - getGibbsSummary, 3
  - \* **Sampling**
    - getGibbsSummary, 3
  - \* **Simulated**
    - phenoSimPlot, 5
  - \* **Simulate**
    - phenoSim, 4
  - \* **vs**
    - plotPOGibbs, 6
- fitCDM, 2
- getGibbsSummary, 3
- phenoSim, 4
- phenoSimPlot, 5
- plotPOGibbs, 6
- plotPost, 8