Package ‘ctmcmove’

July 22, 2016

Type Package

Title Modeling Animal Movement with Continuous-Time Discrete-Space Markov Chains

Version 1.2.6

Date 2016-07-21

Author Ephraim Hanks

Maintainer Ephraim Hanks <hanks@psu.edu>

Depends R (>= 2.10), raster, Matrix, fda

Suggests mgcv, dismo, crawl

Description Software to facilitates taking movement data in xyt format and pairing it with raster covariates within a continuous time Markov chain (CTMC) framework. As described in Hanks et al. (2015) <DOI:10.1214/14-AOAS803> , this allows flexible modeling of movement in response to covariates (or covariate gradients) with model fitting possible within a Poisson GLM framework.

NeedsCompilation no

Repository CRAN

Date/Publication 2016-07-22 00:57:34

R topics documented:

ctmcmove-package 2
ctmc.sim 6
ctmc2glm 7
get.rate.matrix 9
get.UD 10
mcmc.fmove 11
path2ctmc 13
Pctmc 14
rast.grad 15
seal 16

Index **18**

ctmcmove-package *Modeling Animal Movement with Continuous-Time Discrete-Space Markov Chains*

Description

Software to facilitates taking movement data in xyt format and pairing it with raster covariates within a continuous time Markov chain (CTMC) framework. As described in Hanks et al. (2015), this allows flexible modeling of movement in response to covariates (or covariate gradients) with model fitting possible within a Poisson GLM framework.

Details

The DESCRIPTION file:

Package:	ctmcmove
Type:	Package
Title:	Modeling Animal Movement with Continuous-Time Discrete-Space Markov Chains
Version:	1.0
Date:	2015-10-12
Author:	Ephraim Hanks
Maintainer:	Ephraim Hanks <hanks@psu.edu>
Depends:	crawl, raster, Matrix
Suggests:	mgcv, dismo
Description:	Software to facilitates taking movement data in xyt format and pairing it with raster covariates within a continu
License:	GPL-2

Index of help topics:

```
ctmc2glm                Convert a "ctmc" object into Poisson glm
                        format.
ctmcmove-package        Modeling Animal Movement with Continuous-Time
                        Discrete-Space Markov Chains
get.crawl.path          Simulate a CRAWL path within a raster object.
path2ctmc               Function to turn a discrete-time
                        continuous-space path into a CTMC path.
rast.grad               Creates gradient rasters from a raster object.
seal                    Data for one foraging trip by a male northern
                        fur seal (NFS).
```

Typical work flow for analysis of telemetry / GPS movement data:

1. Fit a CTCRW movement model to telemetry xyt data using the "crawl" package.

2. Create or import raster layers (from package "raster") for each covariate.

3. Impute a quasi-continuous CTCRW path using the "get.crawl.path" command.

4. Turn the quasi-continuous CRAWL path into a CTMC discrete-space path using the "path2ctmc" command.

5. Turn discrete-space path into Poisson GLM format using the "ctmc2glm" command.

6. Repeat #3 - #5 multiple times (M times). Stack together the response "z", model matrix "X", and offset "tau" elements from each imputed path.

7. Fit a Poisson GLM model to the stacked data with response "z", model matrix "X", offset "tau", and weights for each row equal to "1/M".

Author(s)

Ephraim Hanks

Maintainer: Ephraim Hanks <hanks@psu.edu>

References

Hanks, E. M.; Hooten, M. B. & Alldredge, M. W. Continuous-time Discrete-space Models for Animal Movement The Annals of Applied Statistics, 2015, 9, 145-165

Hanks, E.; Hooten, M.; Johnson, D. & Sterling, J. Velocity-Based Movement Modeling for Individual and Population Level Inference PLoS ONE, Public Library of Science, 2011, 6, e22795

Hooten, M. B.; Johnson, D. S.; Hanks, E. M. & Lowry, J. H. Agent-Based Inference for Animal Movement and Selection Journal of Agricultural, Biological, and Environmental Statistics, 2010, 15, 523-538

Examples

```
##
## Example of using a CTMC model for movement
##
## Steps:
##  1. Fit CRAWL to telemetry data (not done here)
##  2. Create covariate raster objects (the CTMC will be on the raster
##     grid cells)
##  3. Impute a CRAWL path
##  4. Turn quasi-continuous CRAWL path into a CTMC discrete-space path
##  5. Turn discrete-space path into latent Poisson GLM format
##  6. Fit a Poisson GLM model to the data
##

library(ctmcmove)

data(seal)
sim.obj=seal$sim.obj
cov.df=seal$cov.df
pred.times=seal$pred.times
xyt=seal$locs[,3:1]
head(xyt)
plot(xyt[,1:2],type="b")

##
```

```
## 2. Creating rasters
##

str(cov.df)

NN=sqrt(nrow(cov.df$X))
sst=matrix(seal$cov.df$X$sst,NN,byrow=TRUE)
sst=sst[NN:1,]
sst=raster(sst,xmn=min(seal$cov.df$X$x),xmx=max(seal$cov.df$X$x),
    ymn=min(seal$cov.df$X$y),ymx=max(seal$cov.df$X$y))


crs(sst)="+proj=longlat +datum=WGS84"
plot(sst)

chA=matrix(seal$cov.df$X$chA,NN,byrow=TRUE)
chA=chA[NN:1,]
chA=raster(chA,xmn=min(seal$cov.df$X$x),xmx=max(seal$cov.df$X$x),
    ymn=min(seal$cov.df$X$y),ymx=max(seal$cov.df$X$y))
crs(chA)="+proj=longlat +datum=WGS84"

pro=matrix(seal$cov.df$X$pro,NN,byrow=TRUE)
pro=pro[NN:1,]
npp=raster(pro,xmn=min(seal$cov.df$X$x),xmx=max(seal$cov.df$X$x),
    ymn=min(seal$cov.df$X$y),ymx=max(seal$cov.df$X$y))
crs(npp)="+proj=longlat +datum=WGS84"


int=sst
values(int) <- 1

d2r=int
rookery.cell=cellFromXY(int,xyt[1,1:2])
values(d2r)=NA
values(d2r)[rookery.cell]=0
d2r=distance(d2r)

grad.stack=stack(sst,chA,npp,d2r)
names(grad.stack) <- c("sst","cha","npp","d2r")

plot(sst)
points(xyt[,1:2],type="b")

plot(grad.stack)


##
## 3 Impute Quasi-Continuous Path Using Crawl
##

path=get.crawl.path(seal$sim.obj,int,
     mintime=min(sim.obj$datetime),maxtime=max(sim.obj$datetime))
```

```
str(path)
plot(sst)
points(path$xy,col=2,type="l")
points(xyt[,1:2],type="b",pch=2)


plot(sst)
for(i in 1:15){
    path=get.crawl.path(seal$sim.obj,int,
         mintime=min(sim.obj$datetime),maxtime=max(sim.obj$datetime))
    points(path$xy,col=i,type="l")
}
points(xyt[,1:2],type="b",pch=20,cex=2,lwd=2)

##
## 4. Turn continuous space path into a CTMC discrete space path
##

ctmc=path2ctmc(path$xy,path$t,int)

str(ctmc)

##
## 5. Turn CTMC discrete path into latent Poisson GLM data
##

loc.stack=stack(int,sst)
names(loc.stack) <- c("Intercept","sst.loc")

glm.data=ctmc2glm(ctmc,loc.stack,grad.stack)

str(glm.data)

## now repeat 3-5 for 10 imputations
for(i in 2:10){
    path=get.crawl.path(seal$sim.obj,int,
         mintime=min(sim.obj$datetime),maxtime=max(sim.obj$datetime))
    ctmc=path2ctmc(path$xy,path$t,int)
    glm.data=rbind(glm.data,ctmc2glm(ctmc,loc.stack,grad.stack))
}

str(glm.data)

## remove transitions that are nearly instantaneous

idx.0=which(glm.data$tau<10^-10)
idx.0
glm.data=glm.data[-idx.0,]

summary(glm.data)

##
## 6. Fit Poisson GLM and Poisson GAM with time-varying coefficients
```

```
##    (here we are fitting all "M" paths simultaneously,
##     giving each one a weight of "1/M")



fit=glm(z~cha+npp+sst+crw+d2r+sst.loc,
    weights=rep(1/10,nrow(glm.data)),family="poisson",offset=log(tau),data=glm.data)
summary(fit)

##
## 6. (Alternate) We can use any software which fits Poisson glm data.
##    The following uses "gam" in package "mgcv" to fit a time-varying
##    effect of "d2r" using penalized regression splines.  The result
##    is similar to that found in:
##
##    Hanks, E.; Hooten, M.; Johnson, D. & Sterling, J. Velocity-Based
##    Movement Modeling for Individual and Population Level Inference
##    PLoS ONE, Public Library of Science, 2011, 6, e22795
##

library(mgcv)

fit=gam(z~cha+npp+sst+crw++s(t,by=-d2r),
    weights=rep(1/10,nrow(glm.data)),family="poisson",offset=log(tau),data=glm.data)
summary(fit)
plot(fit)
```

`ctmc.sim` *Code to simulate a continuous-time Markov chain.*

Description

Simulates a CTMC with given rate matrix (Q) for a time (T), or until it reaches a final absorbing state.

Usage

```
ctmc.sim(Q,start.state=1,T=1,final.state=NA)
```

Arguments

`Q`	A square matrix. Either a rate matrix or the infinitessimal generator of the CTMC.
`start.state`	An integer - the starting state for the simulation.
`T`	A numeric value greater than zero. The time window for simulating the CTMC will be [0,T].

`final.state`	Either NA or an integer. If an integer, the chain will be simulated until it enters the "final.state", at which time the simulation will be terminated.

Details

This code uses the Gillespie algorithm to simulate a CTMC path in continuous time.

Value

`ec`	A vector of the sequential grid cells (the embedded chain) in the CTMC movement path
`rt`	A vector of residence times in each sequential grid cell in the CTMC movement path

Author(s)

Ephraim M. Hanks

References

None

Examples

```
## For example code, do
##
## > help(ctmcMove)
```

`ctmc2glm` *Convert a "ctmc" object into Poisson glm format.*

Description

Transforms a "ctmc" object and covariate rasters into data suitable for analysis using Poisson GLM software (like glm in R).

Usage

```
ctmc2glm(ctmc, stack.static, stack.grad, crw = TRUE,
    normalize.gradients = FALSE, grad.point.decreasing = TRUE,
    include.cell.locations=TRUE,directions=4,zero.idx=integer())
```

Arguments

- `ctmc` A "ctmc" object (output from "path2ctmc").
- `stack.static` A rasterStack object, where each layer in the stack is a location-based covariate.
- `stack.grad` A rasterStack object, where each layer in the stack is a directional gradient-based covariate
- `crw` Logical. If TRUE (default), an autocovariate is created for each cell visited in the CTMC movement path. The autocovariate is a unit-length directional vector pointing from the center of the most recent grid cell to the center of the current grid cell.
- `normalize.gradients` Logical. Default is FALSE. If TRUE, then all gradient covariates are normalized by dividing by the length of the gradient vector at each point.
- `grad.point.decreasing` Logical. If TRUE, then the gradient covariates are positive in the direction of decreasing values of the covariate. If FALSE, then the gradient covariates are positive in the direction of increasing values of the covariate (like a true gradient).
- `include.cell.locations` Logical. If TRUE, then the x and y locations of the centers of the (1) current and (2) neighboring raster cells will be returned for each row in the created data matrix.
- `directions` Integer. Either 4 (indicating a "Rook's neighborhood" of 4 neighboring grid cells) or 8 (indicating a "King's neighborhood" of 8 neighboring grid cells).
- `zero.idx` Integer vector of the indices of raster cells that are not passable and should be excluded. These are cells where movement should be impossible. Default is zero.idx=integer().

Details

This code creates one data row for each possible transition from each grid cell visited by the CTMC path.

Value

- `z` Response variable (either zero or 1) for analysis using GLM software.
- `X` Matrix of predictor variables for analysis using GLM software. Created from the location-based and gradient-based covariates.
- `tau` Offset for each row in a Poisson GLM with log link.
- `t` Vector of the time each raster grid cell was entered

Author(s)

Ephraim M. Hanks

References

Hanks, E. M.; Hooten, M. B. & Alldredge, M. W. Continuous-time Discrete-space Models for Animal Movement The Annals of Applied Statistics, 2015, 9, 145-165

Examples

```
## For example code, do
##
## > help(ctmcMove)
```

`get.rate.matrix` *Create a CTMC rate matrix from rasters and parameter estimates.*

Description

Creates a CTMC rate matrix from rasters and parameter estimates (perhaps from a GLM analysis).

Usage

```
get.rate.matrix(beta.static,beta.grad, stack.static, stack.grad,
    normalize.gradients = FALSE, grad.point.decreasing = TRUE,
    directions=4, zero.idx=integer())
```

Arguments

`beta.static`	A vector of regression parameters corresponding to "motility" covariates in stack.static. The length of beta.static should equal the number of layers in stack.static.
`beta.grad`	A vector of regression parameters corresponding to "directional" or "gradient" covariates in stack.grad. The length of beta.static should equal the number of layers in stack.grad.
`stack.static`	A rasterStack object, where each layer in the stack is a location-based covariate.
`stack.grad`	A rasterStack object, where each layer in the stack is a directional gradient-based covariate
`normalize.gradients`	Logical. Default is FALSE. If TRUE, then all gradient covariates are normalized by dividing by the length of the gradient vector at each point.
`grad.point.decreasing`	Logical. If TRUE, then the gradient covariates are positive in the direction of decreasing values of the covariate. If FALSE, then the gradient covariates are positive in the direction of increasing values of the covariate (like a true gradient).
`directions`	Integer. Either 4 (indicating a "Rook's neighborhood" of 4 neighboring grid cells) or 8 (indicating a "King's neighborhood" of 8 neighboring grid cells).
`zero.idx`	Integer vector of the indices of raster cells that are not passable and should be excluded. These are cells where movement should be impossible. Default is zero.idx=integer().

Details

This function takes the covariate rasters in stack.static (motility covariates) and stack.grad (gradient covariates) and creates a CTMC rate matrix defining movement between all neighboring raster grid cells. It is NOT possible to include an autocovariate here ("crw" in ctmc2glm).

Value

An n-by-n Matrix of CTMC rate values.

Author(s)

Ephraim M. Hanks

References

Hanks, E. M.; Hooten, M. B. & Alldredge, M. W. Continuous-time Discrete-space Models for Animal Movement. The Annals of Applied Statistics, 2015, 9, 145-165

Examples

```
## For example code, do
##
## > help(ctmcMove)
```

`get.UD` *Find the stationary distribution of the CTMC.*

Description

Finds the stationary distribution (proportional utilization distribution) implied by a CTMC movement model with a given rate matrix.

Usage

```
get.UD(R,zero.idx=integer())
```

Arguments

`R`	Rate matrix with R[i,j] equal to the CTMC rate of movement from raster cell i to neighboring raster cell j. R[i,j]=0 implies that cells i and j are not first order neighbors.
`zero.idx`	Integer vector of the indices of raster cells that are not passable and should be excluded. These are cells where movement should be impossible. Default is zero.idx=integer().

Details

This calculates the stationary distribution of the CTMC using the method on pg. 455 of Harrod and Plemmons (1984). If R is a sparse Matrix object, then sparse matrix methods are used, making this calculation extremely efficient.

Value

Vector of the stationary distribution at each raster grid cell

Author(s)

Ephraim M. Hanks

References

Harrod, W. J. & Plemmons, R. J. Comparison of some direct methods for computing stationary distributions of Markov chains. SIAM Journal on Scientific and Statistical Computing, 1984, 5, 453-469

Examples

```
## For example code, do
##
## > help(ctmcMove)
```

---

mcmc.fmove *Fit continuous-time functional movement model to telemetry data.*

Description

Fits a functional movement model to telemetry data following Buderman et al., 2015.

Usage

```
mcmc.fmove(xy,t,fdabasis,tpred=t,QQ="CAR2",a=1,b=1,r=1,q=1,
          n.mcmc=100,num.paths.save=10,sigma.fixed=NA)
```

Arguments

xy	A two-column matrix with each row corresponding to the x,y locations of a telemetry location.
t	A numeric vector of length = nrow(xy), with the i-th entry corresponding to the time of the i-th telemetry location in xy.
fdabasis	A "basisfd" object, typically resulting from a call to "create.bspline.basis" in the fda package. Other basis functions can be used.
tpred	Numeric vector of times to impute the quasi-continuous path.

`QQ`	The precision matrix of the fda basis coefficients. This can either be a string, taking on values of "CAR1" or "CAR2", or can be a user specified matrix (or sparse matrix using the Matrix package) of dimension equal to the number of basis functions in fdabasis. Defaults to "CAR2". "CAR1" will result in less-smooth paths.
`a`	The shape parameter of the inverse gamma prior on the observation variance.
`b`	The scale parameter of the inverse gamma prior on the observation variance.
`r`	The shape parameter of the inverse gamma prior on the partial sill parameter of the spline basis coefficients.
`q`	The scale parameter of the inverse gamma prior on the partial sill parameter of the spline basis coefficients.
`n.mcmc`	Number of mcmc iterations to run.
`num.paths.save`	Number of quasi-continuous path realizations to save. Defaults to 10.
`sigma.fixed`	Numeric value (or the default NA). If NA, then the observation variance sigma^2 is estimated using MCMC. If a numeric value, this is the fixed standard deviation of the observation error.

Details

Fits the functional movement model of Buderman et al., 2015, and outputs quasi-continuous paths that stochastically interpolate between telemetry locations. The model fit is as follows (written out for 1-D):

y_t = observed location at time t

z_t = Sum_k beta_k*phi_k(t) = true location at time t, expressed using a linear combination of spline basis functions phi_k(t).

y_t ~ N(z_t , sigma^2)

beta ~ N(0 , tau^2 * QQ^-1)

sigma^2 ~ IG(a,b)

tau^2 ~ IG(r,q)

Value

`s2.save`	Numeric vector of the values of sigma^2 at each mcmc iteration
`tau2.save`	Numeric vector of the values of tau^2 at each mcmc iteration
`pathlist`	A list of length num.paths.save, with each item itself being a list with two entries: xy = a matrix with rows corresponding to x,y locations of the quasi-continuous path imputation t = a vector with entries corresponding to the times at which the quasi-continuous path was imputed

Author(s)

Ephraim M. Hanks

References

Buderman, F.E.; Hooten, M. B.; Ivan, J. S. and Shenk, T. M. A functional model for characterizing long-distance movement behavior. Methods in Ecology and Evolution, 2016, 7, 264-273.

Examples

```
## For example code, do
##
## > help(ctmcMove)
```

---

path2ctmc — *Function to turn a discrete-time continuous-space path into a CTMC path.*

Description

This function takes a movement path defined by xyt values (not necessarily equally spaced in time), and converts it into a CTMC path (a continuous-time discrete-space path on grid cells in a raster).

Usage

```
path2ctmc(xy, t, rast,print.iter=FALSE)
```

Arguments

`xy`	A matrix of x,y locations at T time points.
`t`	A vector of T times associated with the T locations in "xy".
`rast`	A raster object or raster stack object that will define the discrete-space grid cells for the CTMC movement path.
`print.iter`	Logical. If true, then the progress stepping through each observed location in "xy" and "t" will be output in the terminal.

Details

This takes a xyt path and turns it into a list of the embedded chain and residence times of a continuous time Markov chain walk on the graph. "Diagonal" transitions are fixed with linear interpolation.

Value

`ec`	A vector of the sequential grid cells (the embedded chain) in the CTMC movement path
`rt`	A vector of residence times in each sequential grid cell in the CTMC movement path
`trans.times`	A vector of times in which the movement path enters the grid cell in "ec".

Author(s)

Ephraim M. Hanks

References

Hanks, E. M.; Hooten, M. B. & Alldredge, M. W. Continuous-time Discrete-space Models for Animal Movement The Annals of Applied Statistics, 2015, 9, 145-165

Examples

```
## For example code, do
##
## > help(ctmcMove)
```

`Pctmc` *Transition Matrix of a CTMC.*

Description

Computes the transition matrix P(t) of a CTMC with given rate matrix (Q) and time (t).

Usage

```
Pctmc(Q,t)
```

Arguments

`Q`	A square matrix. Either a rate matrix or the infinitessimal generator of the CTMC.
`t`	A numeric value - the time step.

Details

Uses the method of homogenization to compute the probability transition matrix given by exp(Q*t).

Value

A square matrix P with entries P[i,j]=Prob(X(t)=j|X(0)=i)

Author(s)

Ephraim M. Hanks

References

Hanks, E. M.; Hooten, M. B. & Alldredge, M. W. Continuous-time Discrete-space Models for Animal Movement The Annals of Applied Statistics, 2015, 9, 145-165

Examples

```
## For example code, do
##
## > help(ctmcMove)
```

`rast.grad` *Creates gradient rasters from a raster object.*

Description

This function takes a raster stack or raster object and creates two matrices for each raster layer, one which contains the x coordinates of the gradient of the raster layer and one which contains the y coordinates of the gradient of the raster layer.

Usage

```
rast.grad(rasterstack)
```

Arguments

`rasterstack`	A raster layer or raster stack from package "raster".

Details

The gradient is computed using the "terrain" function in raster.

Value

`xy`	A matrix of x and y coordinates of each cell in the raster stack or raster layer. The order is the order of the cells in the raster object.
`grad.x`	a matrix where each column is the x-coordinates of the gradient for one raster layer
`grad.y`	a matrix where each column is the y-coordinates of the gradient for one raster layer
`rast.grad.x`	A raster stack where each raster layer is the x-coordinates of the gradient for one covariate
`rast.grad.y`	A raster stack where each raster layer is the x-coordinates of the gradient for one covariate

Author(s)

Ephraim M. Hanks

References

Hanks, E. M.; Hooten, M. B. & Alldredge, M. W. Continuous-time Discrete-space Models for Animal Movement The Annals of Applied Statistics, 2015, 9, 145-165

Examples

```
## For example code, do
##
## > help(ctmcMove)
```

`seal` *Data for one foraging trip by a male northern fur seal (NFS).*

Description

seal$locs contains xyt locations for ARGOS fixes on the seal's location in the "datetime", "latitude", and "longitude" columns.

seal$cov.df contains a data.frame of spatial covariate values for sea surface temperature (sst), chlorophyll A levels (chA) and net primary production (npp).

Usage

```
data("seal")
```

Format

The format is:

$ locs :'data.frame': 163 obs. of 6 variables:

..$ datetime : num [1:163] 36741 36741 36741 36742 36742 ...

..$ latitude : num [1:163] 57.2 57.3 57.3 57.2 57.5 ...

..$ longitude : num [1:163] 190 190 190 190 190 ...

..$ landseamig: int [1:163] 0 1 1 1 1 1 1 1 1 1 ...

..$ lqadjust : int [1:163] 5 1 0 -2 -2 1 -2 -2 -2 -2 ...

..$ lq : Factor w/ 8 levels "0","1","2","3",..: 5 2 1 7 7 2 7 7 7 7 ...

$ sex : Factor w/ 2 levels "female","male": 2

$ cov.df :List of 4

..$ X :'data.frame': 10000 obs. of 5 variables:

.. ..$ x : num [1:10000] 184 184 184 184 184 ...

.. ..$ y : num [1:10000] 56.7 56.7 56.7 56.7 56.7 ...

.. ..$ chA: num [1:10000] 1.19 0.924 0.744 0.709 0.733 ...

.. ..$ sst: num [1:10000] 9.07 10.35 10.27 10.43 9.98 ...

.. ..$ pro: num [1:10000] 853 821 823 849 886 ...

Details

Covariate Rasters and ARGOS telemetry data for one NFS near the Pribilof islands.

Source

Hanks, E.; Hooten, M.; Johnson, D. & Sterling, J. Velocity-Based Movement Modeling for Individual and Population Level Inference PLoS ONE, Public Library of Science, 2011, 6, e22795

Examples

```
## For example code, do
##
## > help(ctmcmove)
```

Index

∗Topic **Animal Movement**
  ctmcmove-package, 2
∗Topic **\textasciitildekwd1**
  ctmc.sim, 6
  ctmc2glm, 7
  get.rate.matrix, 9
  get.UD, 10
  mcmc.fmove, 11
  path2ctmc, 13
  Pctmc, 14
  rast.grad, 15
∗Topic **\textasciitildekwd2**
  ctmc.sim, 6
  ctmc2glm, 7
  get.rate.matrix, 9
  get.UD, 10
  mcmc.fmove, 11
  path2ctmc, 13
  Pctmc, 14
  rast.grad, 15
∗Topic **datasets**
  seal, 16

ctmc.sim, 6
ctmc2glm, 7
ctmcmove (ctmcmove-package), 2
ctmcmove-package, 2

get.rate.matrix, 9
get.UD, 10

mcmc.fmove, 11

path2ctmc, 13
Pctmc, 14

rast.grad, 15

seal, 16