A brief introduction to IPMpack

Sean M. McMahon

August 09, 2015

The goal of IPMpack is to provide a suite of demographic tools based on Integral Projection Models (IPMs) to support biologists interested in making projections for populations where demography is strongly linked to a changing continuous variable, such as size. The package includes functions that can take data, such as size or age, as well as environmental covariates, and build models of growth, survival and fecundity. Functions are defined that then take these statistical models and construct IPMs. IPMpack has tools that compare different functional forms for the underlying statistical models, plotting them and returning model scores, as well as tools for diagnostic tests of the IPM models themselves. There are also methods to build population models for varying environments, estimate longevity and passage time, sensitivity and elasticity (of either parameters or matrix elements), and much more.

The basic ideas of IPMpack are published in Methods in Ecology and Evolution (Metcalf et al. 2013). Please cite that paper if you want to refer to IPMpack. This vignette is intended to introduce the concepts of IPMs as well as the implementation of IPMpack to biologists with a wide range of quantitative skills. This vignette is for IPMpack version 2.1, and so we encourage users to contact the IPMpack team at IPMpack@gmail.com with any feedback or mistakes they find. We also host a blog at R-forge (http://ipmpack.r-forge.r-project.org/) that contains news of updates, new features, and announcements of papers and meetings relevant to IPMs.

Introduction

IPMpack requires a several other packages, but by using the basic `install.packages()` function in R, everything should be set up for you. After installation, load the library.

```
library(IPMpack)
```

```
## Loading required package: Matrix
## Loading required package: nlme
```

We are ready to begin. But first ... data. One could go into the field for several years and monitor a population of choice. Or, we could use data fabriacated for quick use!

To use IPMpack's full capacities, the individual-level observations of demography should be organized in a {data frame} (a class of object in R [see help file for 'data.frame' in base]), where each row represents one observation of an organism in the population at one census time t with the following potential column names:

- size: size of individuals in census time t *
- sizeNext: size of individuals in census time $t+1$ *
- surv: survival of individuals from census time t to t+1 (contains: 0 for death or 1 for survival) *
- fec1, ...: as many columns as desired relating size to sexual reproduction. For example, this might be:
- fec1: probability of reproduction (output: 0 for no reproductive or 1 for reproductive)
- fec2: number of reproductive structures (output: 1, 2, 3, ...) when individual is reproductive, that is, when fec1 = 1
- fec3: number of propagules (output: 1, 2, 3, ...) per reproductive structure (e.g. seeds per flower in reproductive plant individual)
- ...
- stage: stage of individuals in census time t, used to distinguish discrete and continuous stages, etc. For rows in the data frame where {size} is not an NA, then

this must be the word continuous''. Where {\tt size} is NA, any variety of named discrete stages may be defined (e.g.seed bank'‘). If this column is missing, many procedures in IPMpack are designed to simply fill in this column assuming that only "continuous" state variables describe the life cycle of the species, i.e. there are no discrete stages. For running {makeFecObj}, the column must be a factor. If not supplied, the function will generate this column assuming all individuals are "continuous".

stageNext: stage of individuals in census time $t + 1$, in the simples case, continuous'' ordead'‘(which is redundant with 0''in the "‘surv"‘ column. As above, this column is not essential for many procedures in IPMpack. For running {\tt makeFecObj}, the column must be a factor. If not supplied, the function will generate this column assuming all individuals that are alive arecontinuous".

- number: number of individuals corresponding to each row in the data frame. For all rows corresponding to movement between continuous stages, this value will be 1, but for movement between discrete stages (e.g., from dormant seeds'' toseeds ready to germinate") then this number may be > 1, potentially directly reflecting observed individuals in the data. This information avoids having a data frame with a row for every individual in a discrete stage (e.g. seeds). As above, many procedures in IPMpack will simply assume that this value is always 1.
- covariate: value of a discrete covariate in census time t, such as light environment at time t, age at t, patch at t, etc.
- covariateNext: value of a discrete covariate in census time $t + 1$. \item ...any other covariates of interest, named as desired by the user are possible too (e.g., precipitation, habitat, temperature, etc). \item {offspringNext}: if the size contained in sizeNext corresponds to the size of an offspring, this column will contain either the value sexual" orclonal" (depending on whether sexual or clonal reproduction is being considered). If this column exists, rows that take these two values will be excluded from the growth analyses (functions {makeGrowthObj} and variants thereof, see below).

You can load data from our distributed file. Take a look at the structure using the common suite of commands.

```
dff <- read.csv(file = "Intro_to_IPMs_Exercises_Data.csv")
# str(dff)
# summary(dff)
# head(dff)
# tail(dff)
```

Making a growth object:

```
#  The growth and suvival objects require data and a formula.
#  Here we use a simple polynomial

gr1 <- makeGrowthObj(dff, Formula = incr ~ size + size2)
```

```
## [1] "building incr as sizeNext - size"
```

```
sv1 <- makeSurvObj(dff, Formula = surv ~ size + size2)
```

Looking at the structure of these objects, almost all of the information is related to the regression objects within them.

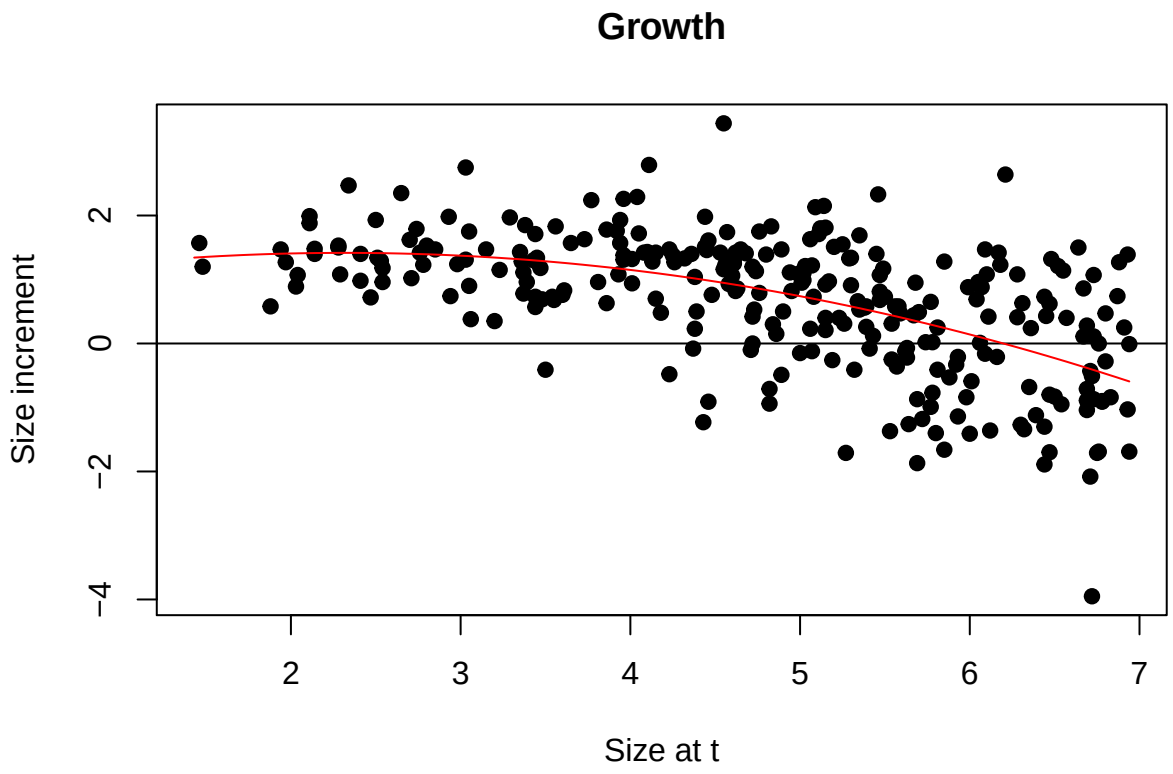
```
str(gr1)
```

```
## Formal class 'growthObjIncr' [package "IPMpack"] with 2 slots
##   ..@ fit:List of 12
##   .. ..$ coefficients : Named num [1:3] 0.912 0.435 -0.094
##   .. .. ..- attr(*, "names")= chr [1:3] "(Intercept)" "size" "size2"
##   .. ..$ residuals    : Named num [1:280] 0.626 -0.526 -0.567 -2.226 0.713 ...
##   .. .. ..- attr(*, "names")= chr [1:280] "3" "7" "8" "9" ...
##   .. ..$ effects      : Named num [1:280] -10.857 9.56 -3.085 -2.236 0.689 ...
##   .. .. ..- attr(*, "names")= chr [1:280] "(Intercept)" "size" "size2" "" ...
##   .. ..$ rank         : int 3
##   .. ..$ fitted.values: Named num [1:280] 0.9842 0.1956 1.2974 0.9962 -0.0828 ...
```

```
##   .. .. ..- attr(*, "names")= chr [1:280] "3" "7" "8" "9" ...
##   .. ..$ assign       : int [1:3] 0 1 2
##   .. ..$ qr           :List of 5
##   .. .. ..$ qr   : num [1:280, 1:3] -16.7332 0.0598 0.0598 0.0598 0.0598 ...
##   .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. ..$ : chr [1:280] "3" "7" "8" "9" ...
##   .. .. .. .. ..$ : chr [1:3] "(Intercept)" "size" "size2"
##   .. .. .. ..- attr(*, "assign")= int [1:3] 0 1 2
##   .. .. ..$ qraux: num [1:3] 1.06 1.05 1.02
##   .. .. ..$ pivot: int [1:3] 1 2 3
##   .. .. ..$ tol  : num 1e-07
##   .. .. ..$ rank : int 3
##   .. .. ..- attr(*, "class")= chr "qr"
##   .. ..$ df.residual  : int 277
##   .. ..$ xlevels      : Named list()
##   .. ..$ call         : language lm(formula = Formula, data = dataf)
##   .. ..$ terms        :Classes 'terms', 'formula' length 3 incr ~ size + size2
##   .. .. .. ..- attr(*, "variables")= language list(incr, size, size2)
##   .. .. .. ..- attr(*, "factors")= int [1:3, 1:2] 0 1 0 0 0 1
##   .. .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. .. ..$ : chr [1:3] "incr" "size" "size2"
##   .. .. .. .. .. ..$ : chr [1:2] "size" "size2"
##   .. .. .. ..- attr(*, "term.labels")= chr [1:2] "size" "size2"
##   .. .. .. ..- attr(*, "order")= int [1:2] 1 1
##   .. .. .. ..- attr(*, "intercept")= int 1
##   .. .. .. ..- attr(*, "response")= int 1
##   .. .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. .. .. ..- attr(*, "predvars")= language list(incr, size, size2)
##   .. .. .. ..- attr(*, "dataClasses")= Named chr [1:3] "numeric" "numeric" "numeric"
##   .. .. .. .. ..- attr(*, "names")= chr [1:3] "incr" "size" "size2"
##   .. ..$ model        :'data.frame': 280 obs. of  3 variables:
##   .. .. ..$ incr : num [1:280] 1.61 -0.33 0.73 -1.23 0.63 ...
##   .. .. ..$ size : num [1:280] 4.46 5.92 3.44 4.43 6.31 2.94 3.6 5.64 5.11 2.51 ...
```

```
##   .. .. ..$ size2: num [1:280] 19.9 35 11.8 19.6 39.8 ...
##   .. .. ..- attr(*, "terms")=Classes 'terms', 'formula' length 3 incr ~ size + size2
##   .. .. .. .. ..- attr(*, "variables")= language list(incr, size, size2)
##   .. .. .. .. ..- attr(*, "factors")= int [1:3, 1:2] 0 1 0 0 0 1
##   .. .. .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. .. .. ..$ : chr [1:3] "incr" "size" "size2"
##   .. .. .. .. .. .. ..$ : chr [1:2] "size" "size2"
##   .. .. .. .. ..- attr(*, "term.labels")= chr [1:2] "size" "size2"
##   .. .. .. .. ..- attr(*, "order")= int [1:2] 1 1
##   .. .. .. .. ..- attr(*, "intercept")= int 1
##   .. .. .. .. ..- attr(*, "response")= int 1
##   .. .. .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. .. .. .. ..- attr(*, "predvars")= language list(incr, size, size2)
##   .. .. .. .. ..- attr(*, "dataClasses")= Named chr [1:3] "numeric" "numeric" "numeric"
##   .. .. .. .. .. ..- attr(*, "names")= chr [1:3] "incr" "size" "size2"
##   .. ..- attr(*, "class")= chr "lm"
##   ..@ sd : num 0.869
```

```
picGrow(gr1, data = dff)
```




```
str(sv1)
```

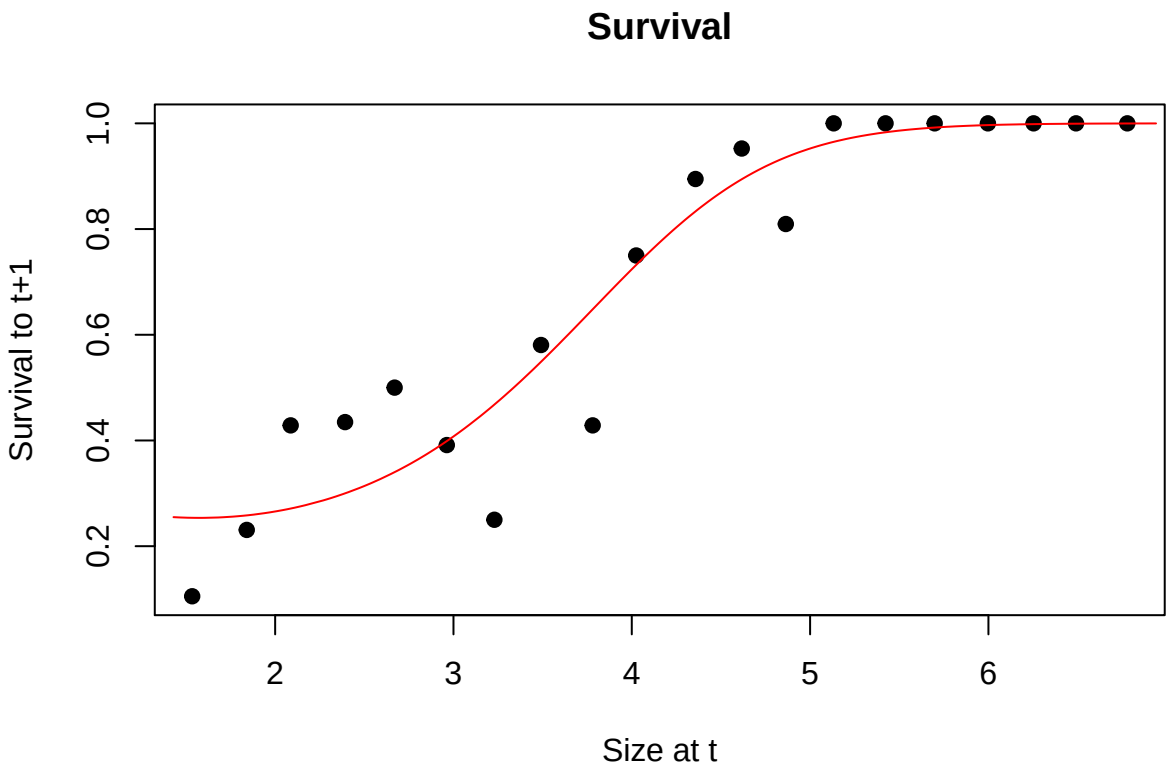
```
## Formal class 'survObj' [package "IPMpack"] with 1 slot
##   ..@ fit:List of 30
##   .. ..$ coefficients     : Named num [1:3] -0.218 -1.094 0.347
##   .. .. ..- attr(*, "names")= chr [1:3] "(Intercept)" "size" "size2"
##   .. ..$ residuals        : Named num [1:400] -1.75 -1.58 1.16 -1.34 -3.58 ...
##   .. .. ..- attr(*, "names")= chr [1:400] "1" "2" "3" "4" ...
##   .. ..$ fitted.values    : Named num [1:400] 0.43 0.366 0.86 0.254 0.72 ...
##   .. .. ..- attr(*, "names")= chr [1:400] "1" "2" "3" "4" ...
##   .. ..$ effects          : Named num [1:400] -0.929 7.821 -2.732 -0.504 -1.585 ...
##   .. .. ..- attr(*, "names")= chr [1:400] "(Intercept)" "size" "size2" "" ...
##   .. ..$ R                : num [1:3, 1:3] -7.15 0 0 -22.77 6.73 ...
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:3] "(Intercept)" "size" "size2"
##   .. .. .. ..$ : chr [1:3] "(Intercept)" "size" "size2"
##   .. ..$ rank             : int 3
##   .. ..$ qr               :List of 5
```

```
##   .. .. ..$ qr   : num [1:400, 1:3] -7.1476 0.0674 0.0486 0.0609 0.0628 ...
##   .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. ..$ : chr [1:400] "1" "2" "3" "4" ...
##   .. .. .. .. ..$ : chr [1:3] "(Intercept)" "size" "size2"
##   .. .. ..$ rank : int 3
##   .. .. ..$ qraux: num [1:3] 1.07 1.03 1.03
##   .. .. ..$ pivot: int [1:3] 1 2 3
##   .. .. ..$ tol  : num 1e-11
##   .. .. ..- attr(*, "class")= chr "qr"
##   .. ..$ family           :List of 12
##   .. .. ..$ family    : chr "binomial"
##   .. .. ..$ link      : chr "logit"
##   .. .. ..$ linkfun   :function (mu)
##   .. .. ..$ linkinv   :function (eta)
##   .. .. ..$ variance  :function (mu)
##   .. .. ..$ dev.resids:function (y, mu, wt)
##   .. .. ..$ aic       :function (y, n, mu, wt, dev)
##   .. .. ..$ mu.eta    :function (eta)
##   .. .. ..$ initialize:  expression({     if (NCOL(y) == 1) {         if (is.factor(y))
##   .. .. ..$ validmu   :function (mu)
##   .. .. ..$ valideta  :function (eta)
##   .. .. ..$ simulate  :function (object, nsim)
##   .. .. ..- attr(*, "class")= chr "family"
##   .. ..$ linear.predictors: Named num [1:400] -0.282 -0.55 1.812 -1.076 0.947 ...
##   .. .. ..- attr(*, "names")= chr [1:400] "1" "2" "3" "4" ...
##   .. ..$ deviance         : num 307
##   .. ..$ aic              : num 313
##   .. ..$ null.deviance    : num 489
##   .. ..$ iter             : int 7
##   .. ..$ weights          : Named num [1:400] 0.245 0.232 0.121 0.19 0.201 ...
##   .. .. ..- attr(*, "names")= chr [1:400] "1" "2" "3" "4" ...
##   .. ..$ prior.weights    : Named num [1:400] 1 1 1 1 1 1 1 1 1 1 ...
##   .. .. ..- attr(*, "names")= chr [1:400] "1" "2" "3" "4" ...
```

```
##   .. ..$ df.residual      : int 397
##   .. ..$ df.null          : int 399
##   .. ..$ y                : Named num [1:400] 0 0 1 0 0 0 1 1 1 0 ...
##   .. .. ..- attr(*, "names")= chr [1:400] "1" "2" "3" "4" ...
##   .. ..$ converged        : logi TRUE
##   .. ..$ boundary         : logi FALSE
##   .. ..$ model            :'data.frame': 400 obs. of  3 variables:
##   .. .. ..$ surv : int [1:400] 0 0 1 0 0 0 1 1 1 0 ...
##   .. .. ..$ size : num [1:400] 3.09 2.81 4.46 1.68 3.99 4.07 5.92 3.44 4.43 2.32 ...
##   .. .. ..$ size2: num [1:400] 9.55 7.9 19.89 2.82 15.92 ...
##   .. .. ..- attr(*, "terms")=Classes 'terms', 'formula' length 3 surv ~ size + size2
##   .. .. .. .. ..- attr(*, "variables")= language list(surv, size, size2)
##   .. .. .. .. ..- attr(*, "factors")= int [1:3, 1:2] 0 1 0 0 0 1
##   .. .. .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. .. .. ..$ : chr [1:3] "surv" "size" "size2"
##   .. .. .. .. .. .. ..$ : chr [1:2] "size" "size2"
##   .. .. .. .. ..- attr(*, "term.labels")= chr [1:2] "size" "size2"
##   .. .. .. .. ..- attr(*, "order")= int [1:2] 1 1
##   .. .. .. .. ..- attr(*, "intercept")= int 1
##   .. .. .. .. ..- attr(*, "response")= int 1
##   .. .. .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. .. .. .. ..- attr(*, "predvars")= language list(surv, size, size2)
##   .. .. .. .. ..- attr(*, "dataClasses")= Named chr [1:3] "numeric" "numeric" "numeric"
##   .. .. .. .. .. ..- attr(*, "names")= chr [1:3] "surv" "size" "size2"
##   .. ..$ call             : language glm(formula = Formula, family = binomial, data = dataf)
##   .. ..$ formula          :Class 'formula' length 3 surv ~ size + size2
##   .. .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. ..$ terms            :Classes 'terms', 'formula' length 3 surv ~ size + size2
##   .. .. .. ..- attr(*, "variables")= language list(surv, size, size2)
##   .. .. .. ..- attr(*, "factors")= int [1:3, 1:2] 0 1 0 0 0 1
##   .. .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. .. ..$ : chr [1:3] "surv" "size" "size2"
##   .. .. .. .. .. ..$ : chr [1:2] "size" "size2"
```

```
##   .. .. .. ..- attr(*, "term.labels")= chr [1:2] "size" "size2"
##   .. .. .. ..- attr(*, "order")= int [1:2] 1 1
##   .. .. .. ..- attr(*, "intercept")= int 1
##   .. .. .. ..- attr(*, "response")= int 1
##   .. .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. .. .. ..- attr(*, "predvars")= language list(surv, size, size2)
##   .. .. .. ..- attr(*, "dataClasses")= Named chr [1:3] "numeric" "numeric" "numeric"
##   .. .. .. .. ..- attr(*, "names")= chr [1:3] "surv" "size" "size2"
##   .. ..$ data             :'data.frame': 400 obs. of  7 variables:
##   .. .. ..$ size      : num [1:400] 3.09 2.81 4.46 1.68 3.99 4.07 5.92 3.44 4.43 2.32 ...
##   .. .. ..$ sizeNext  : num [1:400] NA NA 6.07 NA NA NA 5.59 4.17 3.2 NA ...
##   .. .. ..$ surv      : int [1:400] 0 0 1 0 0 0 1 1 1 0 ...
##   .. .. ..$ fec.seed  : int [1:400] NA NA 15 NA NA NA 17 31 24 NA ...
##   .. .. ..$ fec.flower: int [1:400] NA NA 1 NA NA NA 1 1 1 NA ...
##   .. .. ..$ size2     : num [1:400] 9.55 7.9 19.89 2.82 15.92 ...
##   .. .. ..$ size3     : num [1:400] 29.5 22.19 88.72 4.74 63.52 ...
##   .. ..$ offset           : NULL
##   .. ..$ control          :List of 3
##   .. .. ..$ epsilon: num 1e-08
##   .. .. ..$ maxit  : num 25
##   .. .. ..$ trace  : logi FALSE
##   .. ..$ method           : chr "glm.fit"
##   .. ..$ contrasts        : NULL
##   .. ..$ xlevels          : Named list()
##   .. ..- attr(*, "class")= chr [1:2] "glm" "lm"
```

```
picSurv(sv1, data = dff)
```



Let's make an IPM Pmatrix!

```
str(gr1) picGrow(gr1, data = dff)

str(sv1) picSurv(sv1, data = dff)
```

```
IPM.P <- makeIPMPmatrix(growObj = gr1, survObj = sv1, nBigMatrix = 100)


str(IPM.P)
```

```
## Formal class 'IPMmatrix' [package "IPMpack"] with 7 slots
##   ..@ .Data         : num [1:100, 1:100] 0.1333 0.161 0.1379 0.0836 0.0359 ...
##   ..@ nDiscrete     : num 0
##   ..@ nEnvClass     : num 1
##   ..@ nBigMatrix    : num 100
##   ..@ meshpoints    : num [1:100] -0.745 -0.235 0.275 0.785 1.295 ...
##   ..@ env.index     : int [1:100] 1 1 1 1 1 1 1 1 1 1 ...
##   ..@ names.discrete: chr ""
```

And now we can run some diagnostics, or find passage time, etc.

```
PT <- passageTime(5, IPM.P)


plot(IPM.P@meshpoints, PT)
```