Package ‘PoissonPCA’

October 12, 2022

Title Poisson-Noise Corrected PCA

Version 1.0.3

Description For a multivariate dataset with independent Poisson measurement error, calculates principal components of transformed latent Poisson means. T. Kenney, T. Huang, H. Gu (2019) <arXiv:1904.11745>.

Depends R (>= 3.2.3)

License GPL-3

Encoding UTF-8

Imports

NeedsCompilation yes

Author Toby Kenney [aut, cre],
Tianshu Huang [aut]

Maintainer Toby Kenney <tkenney@mathstat.dal.ca>

Repository CRAN

Date/Publication 2021-08-17 15:20:02 UTC

R topics documented:

get_scores	*Calculates principal scores for Poisson-noise corrected PCA*

Description

This function is based on principal component analysis of a transformation of latent Poisson means of a sample. Given the estimated principal components of the latent Poisson means, this function estimates scores using a combination of likelihood and mean squared error.

Usage

```
get_scores(X,V,d,k,transformation,mu)
get_scores_log(X,V,d,k,mu)
```

Arguments

`X`	The data matrix
`V`	Vector of all principal components of the transformed latent means
`d`	Eigenvalues corresponding to the principal components
`k`	Number of principal components to project onto
`transformation`	The transformation to be applied to the latent means
`mu`	The mean of the transformed latent means

Details

This function estimates the latent transformed Poisson means in order to minimise a combination of the log-likelihood plus the squared residuals of the projection of these latent means onto the first k principal components. Note that for transformed Poisson PCA, the scores are not nested, so the choice of k will have an impact on the projection. The `get_scores_log` function deals with the special case where the transformation is the log function.

Value

`scores`	The principal scores
`means`	The corresponding estimated latent Poisson means

Author(s)

Toby Kenney `<tkenney@mathstat.dal.ca>` and Tianshu Huang <>

Examples

```
n<-20  #20 observations
p<-5   #5 dimensions
r<-2   #rank 2

mean<-10*c(1,3,2,1,1)

set.seed(12345)

Z<-rnorm(n*r)
dim(Z)<-c(n,r)
U<-rnorm(p*r)
dim(U)<-c(r,p)

Latent<-Z%*%U+rep(1,n)%*%t(mean)

X<-rpois(n*p,as.vector(Latent))
dim(X)<-c(n,p)

Sigma<-LinearCorrectedVariance(X[-n,])

eig<-eigen(Sigma)


get_scores(X[n,],eig$vectors,eig$values,r,"linear",colMeans(X[-n,]))

Xlog<-rpois(n*p,exp(as.vector(Latent)+3))
dim(Xlog)<-c(n,p)

logtrans<-makelogtransformation(3,4)

Sigmalog<-TransformedVarianceECV(X[-n,],logtrans$g,logtrans$ECVar)

eiglog<-eigen(Sigmalog)

gX<-X[-n,]

if(!is.null(logtrans)){
    for(i in 1:(n-1)){
        for(j in 1:p){
            gX[i,j]<-logtrans$g(X[i,j])
        }
    }
}

mu<-colMeans(gX)


get_scores_log(X[n,],eiglog$vectors,eiglog$values,r,mu)
```

`LinearCorrectedVariance` *Estimates variance of a transformation of latent Poisson means*

Description

Given a data matrix X[i,j] which follows a Poisson distribution with mean Lambda[i,j], this function estimates the covariance matrix of a transformation f of the latent Lambda.

Usage

```
LinearCorrectedVariance(X)
LinearCorrectedVarianceSeqDepth(X)
TransformedVariance(X,g,CVar)
TransformedVarianceECV(X,g,ECVar)
```

Arguments

`X`	the data matrix
`g`	an estimator of the transformation function f(Lambda). That is, if X~Poisson(Lambda), g(X) should be an estimator of f(Lambda).
`CVar`	an estimator of the conditional variance of g(X) conditional on Lambda.
`ECVar`	an estimator of the conditional variance of g(X) conditional on Lambda.

Details

`LinearCorrectedVariance` merely estimates the covariance matrix of the latent Poisson means without transformation. `LinearCorrectedVarianceSeqDepth` deals with the common case in microbiome and other analysis, where the Poisson means are subject to large multiplicative noise not associated with the parameters of interest. In these cases, we would like to estimate the covariance of the compositional form of Lambda. That is, we want to scale the rows of Lambda to all have sum 1, and estimate the covariance matrix of the resultant matrix. This method uses the actual row sums of X as estimates of the scaling to be performed. `TransformedVariance` estimates the variance of a function of Lambda. It takes two additional parameters: `g` and `CVar` which are functions of X. `g` should be an estimator for the desired transformation f(Lambda) from an observation X. For example, if f(Lambda)=Lambda^2, then the unbiassed estimator is X*(X-1). `CVar` is an estimator for the conditional variance of g(X) given Lambda. For example, if f(Lambda)=Lambda^2, and we use the unbiassed g(X)=X*(X-1), then the variance of g(X) is 4*Lambda^3+3*Lambda^2, so an unbiassed estimator for this is `CVar(X)`=4*X*(X-1)*(X-2)+3*X*(X-1)=X*(X-1)*(4*X-5). The function `polynomial_transformation` will compute the unbiassed estimators for a given polynomial. The function `makelogtransformation` compute estimators for the log function. `TransformedVarianceECV` is the same as `TransformedVariance`, except that the third parameter estimates the average conditional variance from a sample of values of X, rather than a single value.

Value

An estimated covariance matrix for the transformed latent means.

Author(s)

Toby Kenney <tkenney@mathstat.dal.ca> and Tianshu Huang <>

Examples

```
n<-20  #20 observations
p<-5   #5 dimensions
r<-2   #rank 2

mean<-10*c(1,3,2,1,1)

Z<-rnorm(n*r)
dim(Z)<-c(n,r)
U<-rnorm(p*r)
dim(U)<-c(r,p)

Latent<-Z%*%U+rep(1,n)%*%t(mean)

X<-rpois(n*p,as.vector(Latent))
dim(X)<-c(n,p)

LinearCorrectedVariance(X)

seqdepth<-exp(rnorm(n)+2)
Xseqdep<-rpois(n*p,as.vector(diag(seqdepth)%*%Latent))
dim(Xseqdep)<-c(n,p)

LinearCorrectedVarianceSeqDepth(Xseqdep)


squaretransform<-polynomial_transformation(c(1,0))

Xsq<-rpois(n*p,as.vector(diag(seqdepth)%*%Latent)^2)

dim(Xsq)<-c(n,p)

TransformedVariance(Xsq,squaretransform$g,squaretransform$CVar)

Xexp<-rpois(n*p,as.vector(diag(seqdepth)%*%exp(Latent)))

logtrans<-makelogtransformation(3,4)

TransformedVarianceECV(X,logtrans$g,logtrans$ECVar)
```

`makelogtransformation` *constructs a log transformation for use with functions from the PoissoncorrectedPCA package.*

Description

When we are dealing with a transformation of the latent Poisson mean Lambda, we need various useful functions. This function computes the necessary functions for the log transformation, and returns a list of the required functions.

Usage

```
makelogtransformation(a,N,uselog=6,unbiassed=TRUE)
```

Arguments

`a`	The point about which to expand the Taylor series (see details)
`N`	The number of terms in the Taylor series to expand (see details)
`uselog`	Value above which we use the logarithm to approximate g(x). Typically should not be larger than 2a.
`unbiassed`	Indicates that the recommended unbiassed method should be used.

Details

The logarithmic transformation is fundamentally unestimable. There is no estimator which is an unbiassed estimator for log(Lambda). This is because the logarithm function has a singularity at zero, so has no globally convergent Taylor series expansion. Instead, we aim to use an approximately unbiassed estimator. For large enough X, g(X)=log(X) is a reasonable estimator. For smaller X, we need to compute a Taylor series for exp(-Lambda)log(Lambda). We do this from the equation log(x)=log(a)+log(x/a) and the Taylor expansion log(1+y)=y-y^2/2+y^3/3-... where y=x/a-1. This has radius of convergence 1, so will converge provided 0<x<2a. However, if we try to convert it to a polynomial in x, the coefficients will diverge. Instead, we truncate this Taylor series in y at a chosen number N terms. If the x is close to a, this truncated Taylor series should give an approximately unbiassed estimator for log(Lambda). Choice of N can have some effect. Larger values of N reduce the bias of g(X) but increase the variance. Experimentally, a=3 and N=6 seem to produce reasonable results, with g(X)=log(X) for X>6.

Value

`type`	="log"
`f`	function which evaluates the transformation
`g`	an estimator for the transformation of a latent Poisson mean
`solve`	function which computes the inverse transformation (often used for simulations)
`ECVar`	an estimator for the average conditional variance of g(X)

Author(s)

Toby Kenney <tkenney@mathstat.dal.ca> and Tianshu Huang

Examples

```
logtrans<-makelogtransformation(5,6)
X<-rpois(100,exp(1.4))
gX<-X
for(i in 1:100){
gX[i]<-logtrans$g(X[i])
}
mean(gX)
var(gX)
logtrans$ECVar(X)
```

`make_compositional_variance`
Converts a covariance matrix to compositional form

Description

Given a covariance matrix, removes multiplicative noise

Usage

```
make_compositional_variance(Sigma)
make_compositional_min_var(Sigma)
```

Arguments

`Sigma`	the uncorrected covariance matrix

Details

The two functions use different methods. `make_compositional_variance` calculates the variance of compositional data that agrees with Sigma (viewed as a bilinear form) on compositional vectors. That is, the return value Sigma_c is a symmetric matrix which satisfies `t(u)%*%Sigma_c%*%v=t(u)%*%Sigma%*%v` for any compositional vectors u and v, and also `rowSums(Sigma_c)=0`.

Value

The compositionally corrected covariance matrix.

Author(s)

Toby Kenney <tkenney@mathstat.dal.ca> and Tianshu Huang <>

Examples

```
n<-10
p<-5

X<-rnorm(n*p)

dim(X)<-c(n,p)
Sigma<-t(X)%*%X/(n-1)

SigmaComp<-make_compositional_variance(Sigma)

SigmaCompMin<-make_compositional_min_var(Sigma)
```

Poisson_Corrected_PCA *PCA with Poisson measurement error*

Description

Estimates the principal components of the latent Poisson means (possibly with transformation) of high-dimensional data with independent Poisson measurement error.

Usage

```
Poisson_Corrected_PCA(X,k=dim(X)[2]-1,transformation=NULL,seqdepth=FALSE)
```

Arguments

`X`	Matrix or data frame of count variables
`k`	Number of principal components to calculate.
`transformation`	For estimating the principal components of a transformation of the Poisson mean.
`seqdepth`	Indicates what sort of sequencing depth correction should be used (if any).

Details

The options for the transformation parameter are:

NULL or "linear" - these perform no transformation.

"log" - this performs a logarithmic transformation

a list of the following functions:

f(x) - evaluates the function deriv(x) - evaluates the derivative of the function solvefunction(target) - evaluates the inverse of the function g(x) - an estimator for f(lambda) from a Poisson observation x with mean lambda CVar(x) - an estimator for the conditional variance of g(x) conditional on lambda from the observed value x the function polynomial_transformation creates such a list in the case where f is a polynomial using unbiassed estimators for g and CVar. The function makelogtransformation creates an estimator for the logarithmic transformation. The "log" option uses this function with parameters a=3 and N=4, which from experiments appear to produce reasonable results in most situations.

The options for the seqdepth parameter are:

FALSE - indicating no sequencing depth correction

TRUE - indicating standard sequencing depth correction for linear PCA

"minvar" - uses the minimum covariance estimator for the corrected variance. This subtracts the largest constant from all entries of the matrix, such that the matrix is still non-negative definite.

"compositional" - uses the best compositional variance approximation to the estimated covariance matrix.

The package estimates latent principal components using the methods in http://arxiv.org/abs/1904.11745

Value

An object of type "princomp" and "transformedprincomp" that has the following components:

`sdev` The standard deviation associated to each principal component

`loadings` The principal component vectors

`center` The mean of the transformed data

`scale` A vector of ones of length n

`n.obs` The number of observations

`scores` The principal scores. For the linear transformation, these are just the projection of the data onto the principal component space. For transformed principal components, these use a combination of likelihood and mean squared error.

`means` The corresponding estimated untransformed Poisson means. This provides a useful diagnostic of the performance in simulation studies. These means should be closer to the true Lambda than the original X data.

`variance` The corrected covariance matrix for the transformed latent Sigma.

`non_compositional_variance`
The corrected covariance matrix without sequencing depth correction.

`call` The function call used

Author(s)

Toby Kenney <tkenney@mathstat.dal.ca> and Tianshu Huang <>

Examples

```
set.seed(12345)
n<-20  #20 observations
p<-5   #5 dimensions
r<-2   #rank 2
```

```
mean<-10*c(1,3,2,1,1)

Z<-rnorm(n*r)
dim(Z)<-c(n,r)
U<-rnorm(p*r)
dim(U)<-c(r,p)

Latent<-Z%*%U+rep(1,n)%*%t(mean)

X<-rpois(n*p,as.vector(Latent))
dim(X)<-c(n,p)

Poisson_Corrected_PCA(X,k=2,transformation=NULL,seqdepth=FALSE)

seqdepth<-exp(rnorm(n)+2)
Xseqdep<-rpois(n*p,as.vector(diag(seqdepth)%*%Latent))
dim(Xseqdep)<-c(n,p)

Poisson_Corrected_PCA(Xseqdep,k=2,transformation=NULL,seqdepth=TRUE)

squaretransform<-polynomial_transformation(c(1,0))

Xexp<-rpois(n*p,as.vector(diag(seqdepth)%*%exp(Latent)))

Poisson_Corrected_PCA(Xseqdep,k=2,transformation="log",seqdepth="minvar")
```

`polynomial_transformation` *constructs a polynomial transformation for use with functions from the PoissoncorrectedPCA package.*

Description

When we are dealing with a transformation of the latent Poisson mean Lambda, we need various useful functions. This function computes the necessary functions for a polynomial, and returns a list of the required functions.

Usage

```
polynomial_transformation(coeffs)
```

Arguments

`coeffs`	A vector of coefficents of the polynomial. The constant term should not be included.

Details

The coefficients of the polynomial should be given in order of decreasing degree, and should not include the constant term. For example `"coeffs"=c(1,2,3)` refers to the polynomial X^3+2*X^2+3*X. This function returns a list of functions for dealing with this transformation.

Value

`f`	evaluates the transformation
`g`	an estimator for the transformation of a latent Poisson mean
`solve`	computes the inverse transformation (often used for simulations)
`CVar`	an estimator for the conditional variance of g(X)

Author(s)

Toby Kenney <tkenney@mathstat.dal.ca> and Tianshu Huang <>

Examples

```
cubic<-polynomial_transformation(c(1,0,0))

X<-rpois(100,1.8^3)

gX<-X
varX<-X
for(i in 1:100){
gX[i]<-cubic$g(X[i])
varX[i]<-cubic$CVar(X)
}
mean(gX)
var(gX)
mean(varX)
```

Index