Dimension Reduction Regression in R

Sanford Weisberg*
School of Statistics, University of Minnesota, St. Paul, MN 55108-6042.
Supported by National Science Foundation Grant DUE 0109756.

December 19, 2002

Abstract

Regression is the study of the dependence of a response variable y on a collection p predictors collected in x. In *dimension reduction regression*, we seek to find a few linear combinations $\beta_1'x, \ldots, \beta_d'x$, such that all the information about the regression is contained in these d linear combinations. If d is very small, perhaps one or two, then the regression problem can be summarized using simple graphics; for example, for $d = 1$, the plot of y versus $\beta_1'x$ contains all the regression information. When $d = 2$, a 3D plot contains all the information.

Several methods for estimating d and relevant functions of $\beta_1 \ldots, \beta_d$ have been suggested in the literature. In this paper, we describe an R package for three important dimension reduction methods: sliced inverse regression or sir, sliced average variance estimates, or save, and principal Hessian directions, or phd. The package is very general and flexible, and can be easily extended to include other methods of dimension reduction. It includes tests and estimates of the dimension d, estimates of the relevant information including $\beta_1, \ldots, \beta_d$, and some useful graphical summaries as well.

1 Introduction

In the general regression problem, we have a response y of dimension k (usually, $k = 1$) and p-dimensional predictor x, and the goal is to learn about how the conditional distributions $F(y|x)$ as x varies through its sample space. In parametric regression, we specify a functional form for the conditional distributions that is known up to a few parameters. In nonparametric regression, no assumptions are made about F, but progress is really only possible if the dimensions p and k are small.

Dimension reduction regression is one intermediate possibility between the parametric and nonparametric extremes. In this setup, we assume without loss of information that the conditional distributions can be indexed by d linear combinations, or for some probably unknown $p \times d$ matrix B

$$F(y|x) = f(y|B'x) \tag{1}$$

or in words, all the information in x about y is contained in the d linear combinations $B'x$. This representation always holds trivially, by setting $B = I$, the $p \times p$ identity matrix, and so the usual goal is to find the B of lowest possible dimension for which this representation holds. If (1) holds for a particular B, then it also holds for $B^* = BA$, where A is any full rank matrix,

* *Journal of Statistical Software*, 2002, Volume 7, available from http://www.jstatsoft.org

and hence the unique part of the regression summary is the subspace that is spanned by B, which we denote $\mathcal{S}(B)$. Cook (1998) provides a more complete introduction to these ideas, including discussion of when this subspace, which we call the *central subspace*, exists, and when it is unique.

In this paper, we discuss software for estimating the subspace $\mathcal{S}(B)$ spanned by B, and tests concerning the dimension d based on dimension reduction methods. This software was written using R, but can also be used with Splus. Most, but not all, of the methods available here are also included in the *Xlisp-Stat* program *Arc*, Cook and Weisberg (1999). The R platform allows use of dimension reduction methods with existing statistical methods that are not readily available in *Xlisp-Stat*, and hence in *Arc*. For example, the R functions are more suitable for Monte Carlo experimentation than *Arc*. In addition, R includes a much wider array of options for smoothing, including multidimensional smoothers. On the other hand, *Arc* takes full advantage of the dynamic graphical capabilities of *Xlisp-Stat*, and at least for now the graphical summaries of dimension reduction regression are clearly superior in *Arc*. Thus, there appears to be good reason to have these methods available using both platforms.

Cook (1998) provides the most complete introduction to this area, and we leave the details of all the methods described here to that book. See also Cook and Weisberg (1994) for a more gentle introduction to dimension reduction. In this paper we give only the barest outline of dimension reduction methodology, concentrating on the software.

Suppose we have data (x_i, y_i), for $i = 1, \ldots, n$ that are independent and collected into a matrix X and a vector Y if $k = 1$ and a matrix Y if $k > 1$. In addition, suppose we have nonnegative weights $w_1, \ldots, w_n$ whose sum is n; if unspecified, we take all the $w_i = 1$. Generally following Yin (2000), a procedure for estimating $\mathcal{S}(B)$ and for obtaining tests concerning d is:

1. Scale and center X as
$$Z = W^{1/2}(X - 1\bar{x}')\hat{\Sigma}^{-1/2}$$
where $\bar{x} = \sum w_i x_i / \sum w_i$ is the vector of weighted column means, $W = \mathrm{diag}(w_i)$, and
$$\hat{\Sigma} = \frac{1}{n-1}(X - 1\bar{x}')'W(X - 1\bar{x}')$$
$\hat{\Sigma}^{-1/2}$ is any square root of the inverse of the sample covariance matrix for X (for example, using a singular value decomposition) and $\bar{x}$ is a vector of weighted sample means. In this scaling, the rows of Z have zero mean and identity sample covariance matrix.

2. Use the scaled and centered data Z to find a dimension $p \times p$ symmetric matrix $\hat{M}$ that is a consistent estimate of a population matrix M with the property that $\mathcal{S}(M) \subseteq \mathcal{S}(B)$. For most procedures, all we can guarantee is that M tells us about a part, but not necessarily all, of $\mathcal{S}(B)$. Each of the methods (for example, sir, save, and phd) have a different method for selecting $\hat{M}$.

3. Let $|\hat{\lambda}_1| \geq \ldots \geq |\hat{\lambda}_p|$ be the ordered absolute eigenvalues of $\hat{M}$, and $\hat{u}_1, \ldots, \hat{u}_p$ the corresponding eigenvectors of $\hat{M}$. In some applications (like phd) the eigenvalues may be negative.

4. A test that the dimension $d = d_0$ against the alternative that $d > d_0$ is based on a partial sum of eigenvalues of the form:
$$\Lambda_{d_0} = n\hat{c} \sum_{j=d_0+1}^{p} |\hat{\lambda}_j|^{\nu}$$

where $\hat{c}$ is a method-specific term, and ν is generally equal to 1, but it is equal to 2 for `phd`. The distribution of these partial sums depends on assumptions and on the method of obtaining $\hat{M}$.

5. Given d, the estimate of $\mathcal{S}(B)$ is the span of the first d eigenvectors. When viewed as a subspace of $\Re^n$, the basis for this estimated subspace is $Z\hat{u}_1, \ldots, Z\hat{u}_d$. These directions can then be back-transformed to the X-scale. Given the estimate of $\mathcal{S}(B)$, graphical methods can be used to recover information about F, or about particular aspects of the conditional distributions, such as the conditional mean function.

2 Usage

The R package for dimension reduction called `dr` can be obtained from the Comprehensive R Archive Network, CRAN, at `http://www.R-project.org`. The primary function in this package is called `dr`. Its usage is modeled after the R function `lm` that is used for fitting linear models, and so many of the arguments that can be used in `lm` can also be used in `dr`. The following statements will examine dimension reduction using sir applied to the Australian Institute of Sport data:

```
> library(dr)
> data(ais)
> attach(ais)
> i1 <- dr(LBM~ Ht + Wt + log(RCC) + WCC, method="sir", nslices=8)
```

First, provide a formula giving the response Y on the left side, and the predictors on the right side. The response can be a vector of n elements or for sir and save it can be a matrix with n rows and k columns. The model statement on the right side can include factors and interactions, or transformations, so the syntax is very general. The remaining arguments are all optional, and the values shown are all defaults. `method` can equal either `"sir"`, `"save"`, `"phd"`, `"phdres"`, `"phdq"` or `"phdy"`. Adding other methods is not difficult, and is described in Section 3.6. The number of slices is relevant only for the methods that use slicing, sir and save, and has default equal to the larger of 8 and the number of predictors plus 3. If the response has k columns, then the argument `nslices` should be a vector of k elements. If it is specified as a number rather than a vector, then that number will give the total number of cells, approximately. For example, if $k = 2$ and `nslices=8`, the program will slice $\sqrt{8} \approx 3$ slices along each of the two response variables for a total of $3 \times 3 = 9$ cells. The argument `numdir` specifies the number of directions to display in printed output and the number of tests to compute. The default is 4.

Keywords inherited from `lm` include `weights`, which if set should be a vector of the same length of the response of positive numbers; `contrasts`, which specifies how to turn factors into variables; `na.action`, which specifies how to "handle" missing values. The default for this argument is `na.omit`, which will reduce the dataset by eliminating all cases with one or more missing value. The argument `subset` can be set to a list of case indices to be used in the computations; all other cases are ignored.

Brief printed output from the `dr` method is obtained by simply typing the name of the object created, `i1` in the above example:

```
> i1

Call:
dr(formula = LBM ~ Ht + Wt + log(RCC) + WCC, method = "sir", nslices = 8)
```

```
Eigenvectors:
                Dir1          Dir2        Dir3         Dir4
Ht        0.01054752 -0.0001569418 -0.10750718  0.009197952
Wt        0.02374812  0.0040912479  0.06248766 -0.019286750
log(RCC)  0.99960915 -0.9999614779  0.99211547  0.757087590
WCC      -0.01031144  0.0077640009 -0.01563290  0.652963858
Eigenvalues:
[1] 0.87789585 0.15017504 0.03972711 0.01737281
```

This output repeats the call that created the object, and basic summary statistics, described more completely in Section 3.1. Using the `summary` method gives more complete output:

```
> summary(i1)

Call:
dr(formula = LBM ~ Ht + Wt + log(RCC) + WCC, method = "sir", nslices = 8)

Terms:
LBM ~ Ht + Wt + log(RCC) + WCC

Method:
sir with 8 slices, n = 202.

Slice Sizes:
26 26 25 25 25 27 30 18

Eigenvectors:
             Dir1       Dir2     Dir3      Dir4
Ht        0.01055  0.0001569 -0.10751  0.009198
Wt        0.02375 -0.0040912  0.06249 -0.019287
log(RCC)  0.99961  0.9999615  0.99212  0.757088
WCC      -0.01031 -0.0077640 -0.01563  0.652964

              Dir1   Dir2    Dir3    Dir4
Eigenvalues 0.8779 0.1502 0.03973 0.01737
R^2(OLS|dr) 0.9986 0.9987 0.99978 1.00000

Asymp. Chi-square tests for dimension:
               Stat df  p-value
0D vs >= 1D 219.205 28 0.000000
1D vs >= 2D  41.870 18 0.001153
2D vs >= 3D  11.534 10 0.317440
3D vs >= 4D   3.509  4 0.476465
```

This output repeats most of the basic output, plus it also gives information on the slices and on tests of dimension, if available.

The class of the object created by `dr` depends on the value of the argument `method`. For example, if `method="sir"`, the object is of class `sir` if the response is univariate and `msir` if the response is multivariate. If `method="save"`, the object is of class `save` or `msave`. All these objects inherit from the class `dr`.

Several additional quantities are computed and stored in the object. These include:

```
> names(i1)
 [1] "formula"          "contrasts"        "xlevels"          "call"
```

```
 [5] "ols.coef"         "ols.fit"          "weights"          "cols.used"
 [9] "offset"           "estimate.weights" "terms"            "method"
[13] "response.name"    "model"            "cases"            "evectors"
[17] "evalues"          "numdir"           "raw.evectors"     "decomp"
[21] "M"                "slice.info"
```

For example, to have access to the eigenvectors of $\hat{M}$, type

```
> i1$evectors
                Dir1          Dir2        Dir3         Dir4
Ht         0.01054752  0.0001569418 -0.10750718  0.009197952
Wt         0.02374812 -0.0040912479  0.06248766 -0.019286750
log(RCC)   0.99960915  0.9999614779  0.99211547  0.757087590
WCC       -0.01031144 -0.0077640009 -0.01563290  0.652963858
```

while `i1$M` returns the value of the matrix $\hat{M}$. `ols.fit` returns ols fitted values (or weighted least squares fitted values, if appropriate) which are computed and used by some of the methods and summaries.

3 Methods available

3.1 Sliced inverse regression

Sliced inverse regression, or sir, was proposed by Li (1991); see Cook (1998, Chapter 11). In sir, we make use of the fact that given certain assumptions on the marginal distribution of X[1], the inverse regression problem $F(z|y) \subseteq \mathcal{S}(B)$. The general computational outline for sir is as follows:

1. Examine $F(z|y)$ by dividing the range of Y into h slices, each with approximately the same number of observations. With a multivariate response (Y has k columns), divide the range of $Y_1 \times Y_2 \ldots \times Y_k$ into h cells. For example, when $k = 3$, and we slice Y_1 into 3 slices, Y_2 into 2 slices, and Y_3 into 4 slices, we will have $h = 2 \times 3 \times 4 = 24$ cells. The number of slices or cells h is a tuning parameter of the procedure.

2. Assume that within each slice or cell $F(z|y)$ is approximately constant. Then the expected value of the within-slice vector of sample means will be a vector in $\mathcal{S}(B)$.

3. Form the $h \times p$ matrix whose i-th row is the vector of weighted sample means in the i-th slice. The matrix $\hat{M}$ is the $p \times p$ sample covariance matrix of these sample mean vectors.

sir thus concentrates on the mean function $\mathrm{E}(Z|Y)$, and ignores any other dependence.

The output given in the last section is an example of typical output for sir. First is given the eigenvectors and eigenvalues of $\hat{M}$; the eigenvectors have been back-transformed to the original X-scale. Assuming that the dimension is d, the estimate of $\mathcal{S}(B)$ is given by the first d eigenvectors. Also given along with the eigenvectors is the square of the correlation between the ols fitted values and the first d principal directions. The first direction selected by sir is almost always about the same as the first direction selected by ols, as is the case in the example above.

[1] For the methods discussed here, we generally need the linearity condition, the $\mathrm{E}(A'x|B'x)$ is a linear function for $B'x$, where A is any matrix, and B is a matrix such that $F(y|x) = F(y|B'x)$. This condition holds, for example, if the marginal distribution of X is normal, although that assumption is much stronger than is needed. See Cook (1998), Cook and Weisberg (1994, 1999) for more discussion.

For sir, Li (1991) provided asymptotic tests of dimension based on partial sums of eigenvalues, and these tests are given in the summary. The tests have asymptotic Chi-square distributions, with the number of degrees of freedom shown in the output.

Examining the tests shown in the final output, we see that the test of $d = 0$ versus $d > 0$ has a very small p-value, so we would reject $d = 0$. The test for $d = 1$ versus $d > 1$ has p-value near 0.001, suggesting that d is at least 2. The test for $d = 2$ versus $d > 2$ has p-value of about 0.31, so we suspect that $d = 2$ for this problem. This suggests that further analysis of this regression problem can be done based on the 3D graph of the response versus the linear combinations of the predictors determined by the first two eigenvectors, and the dimension of the problem can be reduced from 4 to 2 without loss of information. See Cook (1998), and Cook and Weisberg (1994, 1999), for further examples and interpretation.

When the response is multivariate, the format of the call is:

```
m1 <- dr(cbind(LBM,RCC)~Ht+Wt+WCC))
```

The summary for a multivariate response is similar:

```
> summary(m1)

Call:
dr(formula = cbind(LBM, RCC) ~ Ht + Wt + WCC)

Terms:
cbind(LBM, RCC) ~ Ht + Wt + WCC

Method:
sir with 9 slices, n = 202, using weights.

Slice Sizes:
24 23 23 23 22 21 22 22 22

Eigenvectors:
      Dir1    Dir2    Dir3
Ht  0.4857  0.3879  0.1946
Wt  0.8171 -0.2238 -0.1449
WCC 0.3105 -0.8941  0.9701

              Dir1    Dir2    Dir3
Eigenvalues 0.7076 0.05105 0.02168
R^2(LBM|dr) 0.9911 0.99124 1.00000
R^2(RCC|dr) 0.9670 0.97957 1.00000

Asymp. Chi-square tests for dimension:
              Stat df p-value
0D vs >= 1D 157.63 24  0.0000
1D vs >= 2D  14.69 14  0.3995
2D vs >= 3D   4.38  6  0.6254
```

The test statistics are the same as in the univariate response case, as is the interpretation of the eigenvalues and vectors. The output gives the squared correlation of each of the responses with the eigenvectors.

3.2 Sliced average variance estimation

Sliced average variance estimation, or save, was proposed by Cook and Weisberg (1991). As with sir, we slice the range of Y into h slices, but rather than compute the within-slice mean we compute within-slice covariance matrices. If C_i is the weighted within slice sample covariance matrix in slice i, then the matrix $\hat{M}$ is given by

$$\hat{M} = \frac{1}{n}\sum g_i(I - C_i)^2$$

where g_i is the sum of the weights in the slice; if all weights are equal, then the g_i are just the number of observations in each slice. save looks at second moment information and may miss first-moment information, particularly it may miss linear trends.

Output for save is similar to sir, except that no asymptotic tests have been developed. However, tests of dimension based on a permutation test are available; see Section 4.

3.3 Principal Hessian direction

Li (1992) proposed the method called principal Hessian directions, or pHd. This method examines the matrix $\hat{M}$ given by

$$\hat{M} = \frac{1}{n}\sum_{i=1}^{n} w_i f_i z_i z_i'$$

where f_i is either equal to y_i (method phdy) or f_i is an ols residual (method phd or phdres), and w_i is the weight for the i-th observation (recall again that we assume that $\sum w_i = n$, and if this is not satisfied the program rescales the weights to meet this condition). While all methods produce $\hat{M}$ matrices whose eigenvectors are consistent estimates of vectors in $\mathcal{S}(B)$, the residual methods are more suitable for tests of dimension. See Cook (1998, Chapter 12) for details.

Output for phd is again similar to sir, except for the tests. Here is the output for the same setup as before, but for method phdres:

```
> i2 <- update(i1,method="phdres")
> summary(i2)

Call:
dr(formula = LBM ~ Ht + Wt + log(RCC) + WCC, method = "phdres")

Terms:
LBM ~ Ht + Wt + log(RCC) + WCC

Method:
phd, n = 202.

Eigenvectors:
             Dir1       Dir2      Dir3     Dir4
Ht        0.12764 -0.0003378  0.005550  0.02549
Wt       -0.02163  0.0326138 -0.007342 -0.01343
log(RCC) -0.74348  0.9816463  0.999930 -0.99909
WCC       0.65611 -0.1879008 -0.007408 -0.03157

               Dir1   Dir2   Dir3   Dir4
Eigenvalues 1.4303 1.1750 1.1244 0.3999
R^2(OLS|dr) 0.2781 0.9642 0.9642 1.0000
```

```
Asymp. Chi-square tests for dimension:
              Stat df Normal theory Indep. test General theory
0D vs >= 1D 35.015 10     0.0001241    0.005427        0.01811
1D vs >= 2D 20.248  6     0.0025012          NA        0.03200
2D vs >= 3D 10.281  3     0.0163211          NA        0.05530
3D vs >= 4D  1.155  1     0.2825955          NA        0.26625
```

The column of tests called "normal theory" were proposed by Li (1992) and require that the predictors are normally distributed. These statistics are asymptotically distributed as Chi-square, with the degrees of freedom shown.

When the method is `phdres` additional tests are provided. Since this method is based on residuals, it gives tests concerning the central subspace for the regression of the residuals on X rather than the response on X. The subspace for this residual regression may be, but need not be, smaller than the subspace for the original regression. For example, the column marked "Indep. test" is essentially a test of $d = 0$ versus $d > 0$ described by Cook (1998) for the residual regression. Should the significance level for this test be large, we might conclude that the residual regression subspace is of dimension zero. From this we have two possible conclusions: (1) the dimension of the response regression may be 1 if using the residuals removed a linear trend, or (2) the dimension may be 0 if the residuals did not remove a linear trend.

Similarly, if the significance level for the independence test is small, then we can conclude that the dimension is at least 1. It could be one if the method is picking up a nonlinear trend in the OLS direction, but it will be 2 if the nonlinearity is in some other direction.

The independence test and the final column, also from Cook (1998), use the same test statistic, but different distributions based on different assumptions. Significance levels are obtained by comparing the statistic to the distribution of a random linear combination of Chi-square statistics, each with one df. These statistics do not require normality of the predictors. The way the significance levels in this column are approximated using the method of Wood (1989).

3.4 Quadratic phd

Li (1992) proposed an alternative method of estimating the Hessian matrix based on quadratic regression, as follows: (1) fit the ols regression of the response on the full second-order model based on all p predictors; (2) set $\hat{M}$ to be the $p \times p$ matrix whose (i, j) element is the estimated coefficient of $x_i x_j$. Given this estimate of M, proceed as with other `phd` methods.

3.5 Multivariate phd

No multivariate extensions of `phd` have yet been proposed, and so the response must be univariate for any `phd` method.

3.6 Adding other methods to `dr`

You may skip this section unless you are interested in adding additional dimension reduction methods to `dr`. The `dr` function is designed to be flexible and to make adding additional methods to the package as easy as possible. Here are the steps you need to follow:

1. Select a name for your method. For example suppose you select the name "poly" to mean that you will be estimating M by fitting polynomials to the inverse plots each of the

Table 1: The `dr.fit.M` method for sir.

```
#########################################################################
#     Sliced Inverse Regression
#########################################################################

dr.fit.M.sir <-function(object,z,y,w=NULL,nslices=NULL,
                        slice.info=NULL,...) {
# get slice information
    h <- if (!is.null(nslices)) nslices else max(8, NCOL(z)+3)
    slices<- if(is.null(slice.info)) dr.slices(y,h) else slice.info
# initialize slice means matrix
    zmeans <- matrix(0,slices$nslices,NCOL(z))
    slice.weight <- slices$nslices
# make sure weights add to n
    wts <- if(is.null(w)) rep(1,NROW(z)) else NROW(z) * w /sum(w)
# compute weighted means within slice (weights always add to n)
    wmean <- function (x, wts) { sum(x * wts) / sum (wts) }
    for (j in 1:slices$nslices){
      sel <- slices$slice.indicator==j
      zmeans[j,]<- apply(z[sel,],2,wmean,wts[sel])
      slice.weight[j]<-sum(wts[sel])}
# get M matrix for sir
    M <- t(zmeans) %*% apply(zmeans,2,"*",slice.weight)/ sum(slice.weight)
    return (list (M=M,slice.info=slices))
}
```

predictors versus the response. When you call `dr` with `method="poly"`, an object of class `poly` will be created. If your response is a matrix, the object will be of type `mpoly`, which inherits from `poly`.

2. You will need to write a function called `dr.fit.M.poly` that estimates the M matrix for your method. You can model this function after the function `dr.fit.M.sir` shown in Table 1. The important arguments that are passed to this function include `z`, which is an $n \times p$ rotated and centered data matrix with no missing values; `y`, which is the response vector or matrix, and `w`, which is the vector of weights, if any. If your method requires other parameters, for example setting a degree of a polynomial, simply add the argument `degree=2` to the list of function arguments. This sets the default value of `degree` equal to 2. The "..." argument in the functions allow you to add the `degree` argument when you call the function `dr`. Your function must return a list, including the argument `M`, which is the matrix of interest. M can be either a square matrix leading to an analysis of eigenvalues and eigenvectors, as in the example for sir, or it can be a rectangular matrix, leading to use of singular values and vectors. All entries in the list will become attributes of the resulting object. For example, if your list is `list(z=z,degree=degree)`, when you create an object like

   ```
   > i1 <- dr(LBM~Ht+Wt+RCC+WCC,method="poly",degree=3)
   ```

   the value of `i1$degree` will be 3.

3. If your method works differently when `y` is a matrix, write a method called `dr.fit.M.mpoly` to do the computations for this case. For sir, the only difference between univariate and

Table 2: The `dr.test.sir` function.

```
dr.test.sir<-function(object,nd) {
#compute the sir test statistic for the first nd directions
    e<-sort(object$evalues)
    p<-length(object$evalues)
    n<-object$cases
    st<-df<-pv<-0
    nt <- min(p,nd)
    for (i in 0:nt-1)
      {st[i+1]<-n*(p-i)*mean(e[seq(1,p-i)])
       df[i+1]<-(p-i)*(object$slice.info$nslices-i-1)
       pv[i+1]<-1-pchisq(st[i+1],df[i+1])
      }
    z<-data.frame(cbind(st,df,pv))
    rr<-paste(0:(nt-1),"D vs >= ",1:nt,"D",sep="")
    dimnames(z)<-list(rr,c("Stat","df","p-value"))
    z
}
```

multivariate responses is in the way the slices are obtained, and the method `dr.slices` works for either case. As a result, a separate multivariate fit method is not required for sir.

4. If your method has tests, other than the permutation tests available for all methods, you will need to write a function called `dr.test.poly` (or `dr.test.mpoly` if a separate method is required for multivariate responses). The equivalent method for sir is shown in Table 2. The test method is called by the `summary.dr` function.

The function `dr.fit.y(object)` returns the response variable for use in computing M. For example, the function `dr.fit.y.phdy` returns the left-hand side variable from the formula specified in the call to `dr`, while `dr.fit.y.phdres` returns the residuals from the regression of the response on the predictors. You may also want to write a `summary.dr.poly` method if the default summary is not adequate for your needs.

4 Permutation tests

Cook (1998) and Yin (2000) discuss permutation tests of dimension that can be used with a dimension reduction method. These are implemented in the function `dr.permutation.test`. Typical use of this function is

```
> dr.permutation.test(i1,npermute=499)

Permutation tests
Number of permutations:
[1] 499

Test results:
               Stat p-value
0D vs >= 1D 219.205   0.000
```

```
1D vs >= 2D  41.870   0.002
2D vs >= 3D  11.534   0.284
3D vs >= 4D   3.509   0.354
```

The function requires the name of the object. The number of permutations defaults to 50 and the number of directions defaults to 3. Increasing either can increase the computation time required to obtain the solution. The permutation test results for the example are very similar to the asymptotic results given earlier.

5 Graphical methods

The function call `plot(i1)` returns a scatterplot matrix of the response and the principal directions. The call `plot(i1,mark.by.y=TRUE)` produces a scatterplot matrix of the principal directions, but with points marked (colored) according to the value of the response; the point marking does not work with Splus.

The function call `dr.coplot(i1)` returns a plot of the response versus the first principal direction, with a separate panel conditioning on the value of the second principal direction (a coplot). The call `dr.coplot(i1,mark.by.y=T)`, gives a coplot of the first direction versus the second direction conditioning on the third direction and using color to encode the information about the response.

The function `rotplot` is a generic function that allows looking at a number of static views of a 3D plot. The call

```
> rotplot(dr.directions(m1,1:2),dr.y(m1),number=16)
```

gives 16 views of the 3D plot of the response versus linear combinations of the first two principal directions.

6 Weights

Weights are generally used in dimension reduction methods to make the resulting weighted sample closer to a normal distribution than the original sample. Cook (1998, Section 8.4) discusses the method that is implemented here. When weights are present, they are used in centering the data and computing the covariance matrix, and they are used in computing the objective matrix M for `phd`. Weights may be provided by the user with the `weights` argument. If `weights=NULL`, the default, no weighting is used.

The function `dr.weights` is used to estimate weights using the algorithm described by Cook (1998, Sec. 8.4). There are several other arguments that control how the weights are computed, as described below, and on the help page for the function `dr.weights`. The algorithm works as follows:

1. For an $n \times p$ data matrix X, find estimates m and S of the mean and covariance matrix. For this purpose, in R the function `cov.rob` in the `lqs` package is used, while in Splus the function `covRob` in the `robust` package is used; in either case the needed package will be loaded automatically. If you do not want to use one of these routines, you must rewrite the function `robust.center.scale` to use your preferred code. In R, the method of computing m and S is determined by the argument `covmethod`. If `covmethod="classical"`, the usual estimator is used for S, but m is estimated by medians. If `method="mve"`, the default, or `method="mcd"`, the covariance matrix is estimated by the minimum volume ellipsoid

method and the minimum determinant method, respectively. These latter two also return a robust estimate of center. Any tuning parameters for the method to compute the robust estimate of m and S can be passed from the call to `dr`. See the documentation for `cov.rob` for a description of these additional parameters. All the defaults are sensible, so most users will not need to use these additional parameters.

2. Compute the matrix $Z = (X - 1m')S^{-1/2}$. If the data were normally distributed $\mathrm{N}(m, S)$, the rows of Z would be like a sample from $\mathrm{N}(0, I)$.

3. Obtain a random vector b from the $\mathrm{N}(0, \sigma^2 I)$ distribution. The parameter `sigma=1` is a tuning parameter that can be set in the call to `dr`, and values near 1 or slightly smaller seem appropriate. Find the row of Z that is closest to b (the code uses Euclidean distance), and increase a counter for that row by 1.

4. The argument `nsamples` determines the number of times this last step is repeated; the default is `nsamples=10*dim(x)[1]` where X is the $n \times p$ data matrix; this number may be too small.

5. Return a vector of weights given by the value of the counter divided by `nsamples` and multiplied by n, so the sum of the weights will be n.

An example of the use of weights is:

```
> wts <- dr.weights(LBM~Ht+Wt+RCC+WCC)
> i1 <- dr(LBM~Ht+Wt+RCC+WCC,weights=wts,method="phdres")
```

7 Miscellaneous functions included in `dr`

The function `dr.direction` takes two arguments, the object name, and which directions are wanted (for, example, `1:3` returns the first three directions). It returns the matrix XU, scaled to have unit column length unless the argument norm is false, where U gives the specified eigenvectors.

The function calls `dr.x(i1)` and `dr.y(i1)` return the model matrix and the response, respectively. `dr.z(i1,center=T,rotate=T)` returns the centered and rotated Z matrix from X. You can also use `dr.z` by explicity providing a matrix X in place of the first argument, and if necessary a vector of weights as a second argument.

The routine used for slicing is called `dr.slices(y,h)` to slice `y` into `h` slices. If `y` has p columns and `h` has p elements, then slicing is done recursively. The first column of `y` is sliced into `h[1]` slices. Within each of these slices, the second column of `y` is sliced into `h[2]` slices, giving `h[1]*h[2]` slices. This process is then repeated for any additional columns. If `h` is a scalar, then each dimension is sliced into the smallest integer larger than $\mathtt{h}^{1/p}$ slices. For example, if $p = 2$ and `h`$= 8$, then each dimension has 3 slices for a total of 9.

8 Bug Reports

Please send bug reports to sandy@stat.umn.edu.

9 Splus

`dr` has been tested with Splus versions 5 and 6 under Linux. If you want to use `dr` with Splus, you must change the value of `whichengine` on line 34 of the file R/dr to `whichengine <- "s6"` before you load the file. All options available in R seem to work in Splus, except that the argument `mark.by.y` on plots and coplots and the function `markby` do not work with Splus.

10 Acknowledgements

Jorge de la Vega did extensive testing of the code, and wrote some of the functions, and Cindy Yu wrote early versions of some of the functions. Referee's comments were very helpful in revising this paper and in documenting `dr`. Kurt Hornik was very helpful in getting the package to pass all the tests built-in to R.

11 References

Cook, R. D. (1998). *Regression Graphics.* New York: Wiley.

Cook, R. D. and Weisberg, S. (1991). Discussion of Li (1991). *Journal of the American Statistical Association*, 86, 328–332.

Cook, R. D. and Weisberg, S. (1994). *An Introduction to Regression Graphics.* New York: Wiley.

Cook, R. D. and Weisberg, S. (1999). *Applied Regression Including Computing and Graphics*, New York: Wiley.

Li, K C. (1991). Sliced inverse regression for dimension reduction (with discussion), *Journal of the American Statistical Association*, 86, 316-342.

Li, K C. (1992). On principal Hessian directions for data visualization and dimension reduction: Another application of Stein's lemma. *Journal of the American Statistical Association*, 87, 1025–1034.

Wood, A. (1989). An F-approximation to the distribution of a linear combination of chi-squared random variables. *Communication in Statistics, Part B – Simulation and Computation*, 18, 1439–1456.

Yin, Xiangrong (2000). Dimension reduction using inverse third moments and central k-th moment subspaces. Unpublished Ph. D. dissertation, University of Minnesota, School of Statistics.

12 Function documentation

`dr` *Dimension reduction regression*

Description

The function dr implements dimension reduction methods, including SIR, SAVE and pHd.

Usage

```
dr(formula, data=list(), subset, weights, na.action=na.omit, method="sir",
     contrasts=NULL,offset=NULL, ...)
dr.weights(formula, data=list(), subset, weights, na.action=na.omit, method="sir",
     contrasts=NULL,offset=NULL, ...)
```

Arguments

`formula`	a symbolic description of the model to be fit. The details of the model are the same as for lm.
`data`	an optional data frame containing the variables in the model. By default the variables are taken from the environment from which 'dr' is called.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`weights`	an optional vector of weights to be used where appropriate.
`na.action`	a function which indicates what should happen when the data contain 'NA's. The default is 'na.omit,' which will force calculations on a complete subset of cases.
`method`	This character string specifies the method of fitting. "sir" specifies sliced inverse regression and "save" specifies sliced average variance estimation. "phdy" uses principal hessian directions using the response as suggested by Li, and "phdres" uses the LS residuals as suggested by Cook. Other methods may be added
`contrasts`	an optional list. See the 'contrasts.arg' of 'model.matrix.default'.
`offset`	Set an offset or NULL
`...`	additional items that may be required or permitted by some methods. `nslices` is the number of slices used by sir and save. `numdir` is the maximum number of directions to compute, with default equal to 4.

Details

The general regression problem studies $F(y|x)$, the conditional distribution of a response y given a set of predictors x. This function provides methods for estimating the dimension and central subspace of a general regression problem. That is, we want to find a $p \times d$ matrix B such that

$$F(y|x) = F(y|B'x)$$

Both the dimension d and the subspace $R(B)$ are unknown. These methods make few assumptions. All the methods available in this function estimate the unknowns by study of the inverse problem, $F(x|y)$. In each, a kernel matrix M is estimated such that the column space of M should be close

to the central subspace. Eigenanalysis of M is then used to estimate the central subspace. Objects created using this function have appropriate print, summary and plot methods.

Weights can be used, essentially to specify the relative frequency of each case in the data. Empirical weights that make the contours of the weighted sample closer to elliptical can be computed using `dr.weights`. This will usually result in zero weight for some cases. The function will set zero estimated weights to missing.

Several functions are provided that require a dr object as input. `dr.permutation.tests` uses a permutation test to obtain significance levels for tests of dimension. `dr.coplot` allows visualizing the results using a coplot of either two selected directions conditioning on a third and using color to mark the response, or the resonse versus one direction, conditioning on a second direction. `plot.dr` provides the default plot method for dr objects, based on a scatterplot matrix.

Value

dr returns an object that inherits from dr (the name of the type is the value of the `method` argument), with attributes:

`M`	A matrix that depends on the method of computing. The column space of M should be close to the central subspace.
`evalues`	The eigenvalues of M (or squared singular values if M is not symmetric).
`evectors`	The eigenvectors of M (or of M'M if M is not square and symmetric) ordered according to the eigenvalues.
`numdir`	The maximum number of directions to be found. The output value of numdir may be smaller than the input value.
`ols.coef`	Estimated regression coefficients, excluding the intercept, for the (weighted) LS fit.
`ols.fit`	LS fitted values.
`slice.info`	output from sir.slice, used by sir and save.
`method`	the dimension reduction method used.

Other returned values repeat quantities from input.

`dr.weights` returns a vector of weights NA substituted for estimated zero weights.

Author(s)

Sanford Weisberg, sandy@stat.umn.edu

For weights, see R. D. Cook and C. Nachtsheim (1994), Reweighting to achieve elliptically contoured predictors in regression. Journal of the American Statistical Association, 89, 592–599.

References

The details of these methods are given by R. D. Cook (1998). Regression Graphics. New York: Wiley. Equivalent methods are also available in Arc, R. D. Cook and S. Weisberg (1999). Applied Regression Including Computing and Graphics, New York: Wiley, www.stat.umn.edu/arc.

See Also

`dr.permutation.test`,`dr.x`,`dr.y`, `dr.direction`,`dr.coplot`,`dr.weights`

Examples

```
library(dr)
data(ais)
attach(ais)  # the Australian athletes data
#fit dimension reduction using sir
m1 <- dr(LBM~Wt+Ht+RCC+WCC, method="sir", nslices = 8)
summary(m1)

# repeat, using save:

m2 <- update(m1,method="save")
summary(m2)

# repeat, using phd:

m3 <- update(m2, method="phdres")
summary(m3)
```

`dr.estimate.weights` *Compute estimated weighting toward normality*

Description

This function estimate weights to apply to the rows of a data matrix to make the resulting weighted matrix as close to multivariate normality as possible. This method is usually not called directly by the user.

Usage

```
dr.estimate.weights(object, sigma=1, covmethod="mve", nsamples=NULL, ...)
robust.center.scale(x, method,... )
```

Arguments

`object`	a dimension reduction regression object name, or an n by p matrix
`sigma`	A tuning parameter, with default 1, usually in the range .2 to 1.0
`covmethod`	covmethod is passed as the argument `method` to the function `cov.rob` in the required package `lqs`. The choices are `"classical"`, `"mve"` and `"mcd"`. This probably will not work with Splus. If classical is selected, the usual estimate of the covariance matrix is used, but the center is the medians, not the means.
`nsamples`	The weights are determined by random sampling from a data-determined normal distribution. This controls the number of samples
`x`	An $n \times p$ data matrix with no missing values
`method`	see covmethod above
`...`	Additional args are passed to `cov.rob`

Details

The basic outline is: (1) Estimate a mean m and covariance matrix S using a possibly robust method; (2) For each iteration, obtain a random vector from N(m,sigma*S). Add 1 to a counter for observation i if the i-th row of the data matrix is closest to the random vector; (3) return as weights the sample faction allocated to each observation. If you set the keyword `weights.only` to `T` on the call to `dr`, then only the list of weights will be returned.

Value

Returns a list of n weights, some of which may be zero.

Author(s)

Sanford Weisberg, sandy@stat.umn.edu

References

R. D. Cook and C. Nachtsheim (1994), Reweighting to achieve elliptically contoured predictors in regression. Journal of the American Statistical Association, 89, 592–599.

See Also

`dr.weights`, `lqs`

`dr.permutation.test` *Dimension Reduction Regression Functions*

Description

These functions require a dimension reduction regression object as input to produce output of various types.

Usage

```
dr.permutation.test(object, npermute=50,numdir=object$numdir,permute.weights=TRUE)
```

Arguments

`object`	a dimension reduction regression object created by dr
`npermute`	number of permutations to compute, default is 50
`numdir`	maximum permitted value of the dimension, with the default from the object
`permute.weights`	If TRUE, permute weights as well as data in the permutation test. If FALSE, do not permute the weights.

Value

Returns an object of type 'dr.permutation.test' that can be printed or summarized to give the summary of the test.

Author(s)

Sanford Weisberg, sandy@stat.umn.edu

References

See www.stat.umn.edu/arc/addons.html, and then select the article on dimension reduction regression or inverse regression.

See Also

`dr`

Examples

```
data(ais)
attach(ais)  # the Australian athletes data
#fit dimension reduction regression using sir
m1 <- dr(LBM~Wt+Ht+RCC+WCC, method="sir", nslices = 8)
summary(m1)
dr.permutation.test(m1,npermute=100)
plot(m1)
dr.coplot(m1)
```

`plot.dr` *Plotting methods for dimension reduction regression*

Description

These routines provide default plotting methods for dimension reduction regression.

Usage

```
plot.dr(object, which=1:object$numdir, mark.by.y=F,plot.method=pairs, ...)
dr.coplot(object, which=1:object$numdir, mark.by.y=F, ...)
```

Arguments

`object`	Any dimension reduction regression object
`which`	Which directions to plot
`mark.by.y`	if TRUE, use the response as a marking variable to color points; if FALSE, use response in the plot
`plot.method`	The default is to use the pairs or coplot method to draw the plots. If John Fox's car library is available, you can substitute scatterplot.matrix for pairs.
`...`	arguments passed to plot or coplot. In particular, if the car library is available, the argument panel=panel.car will add smoothers to a coplot.

Value

Produces a scatterplot matrix (plot) or coplot (dr.coplot) of the specified directions in an dimension reduction regression

Author(s)

Sanford Weisberg, sandy@stat.umn.edu

References

Cook, R. D. and Weisberg, S. (1999). Applied Regression Including Computing and Graphics. New York: Wiley.

Examples

```
data(ais)
attach(ais)
i1<-dr(LBM~Ht+Wt+RCC+WCC)
plot(i1)
dr.coplot(i1,mark.by.y=TRUE)
```

`rotplot` *draw many 2D projections of a 3D plot*

Description

This function draws several 2D views of a 3D plot, sort of like a spinning plot.

Usage

```
rotplot(x, y, number=9, theta=seq(0, pi/2, length = 9), ...)
```

Arguments

`x`	a matrix with 2 columns giving the horizontal axes of the full 3D plot.
`y`	the vertical axis of the 3D plot.
`number`	Number of 2D views of the 3D plot.
`theta`	a list of rotation angles
`...`	additional arguments passed to coplot

Details

For each value of theta, draw the plot of cos(theta)*x[,1]+sin(theta)*x[,2] versus y.

Value

returns a graph object.

Author(s)

Sanford Weisberg, sandy@stat.umn.edu

Examples

```
data(ais)
attach(ais)
m1 <- dr(LBM ~ Ht + Wt + WCC)
rotplot(dr.direction(m1,which=1:2),dr.y(m1),col=markby(Sex))
```

`dr.x` *Inverse regression term matrix*

Description

dr.x returns the matrix of data constructed from the formula for a dimension reduction regression. dr.y returns the response.

Usage

```
dr.x(object)
dr.y(object)
dr.z(x,weights=NULL,center=TRUE,rotate=TRUE,decomp="svd")
```

Arguments

`object`	An dimension reduction regression object
`x`	a matrix to be centered and scaled OR a dimension reduction object from which the model matrix will be extracted
`weights`	a vector of weights, or NULL if unweighted
`center`	if TRUE, center the output
`rotate`	if TRUE, multiply by the inverse square root of the sample covariance matrix.
`decomp`	Decomposition to be used in computing the rotation; the default is "svd".

Value

dr.x returns an $n \times p$ matrix of terms, excluding the intercept, constructed from the dimension reduction regression object. dr.y returns the response. dr.z returns a possibly centered and scaled version of x.

Author(s)

Sanford Weisberg, <sandy@stat.umn.edu>

See Also

dr

Examples

```
data(ais)
attach(ais)
m1 <- dr(LBM~Ht+Wt+RCC+WCC)
dr.x(m1)
dr.y(m1)
```

The following documentation is from the package lqs by Brian Ripley, and is included here for convenience:

Description

Compute a multivariate location and scale estimate with a high breakdown point – this can be thought of as estimating the mean and covariance of the `good` part of the data. `cov.mve` and `cov.mcd` are compatibility wrappers.

Usage

```
cov.rob(x, cor = FALSE, quantile.used = floor((n + p + 1)/2),
        method = c("mve", "mcd", "classical"), nsamp = "best", seed)
cov.mve(x, cor = FALSE, quantile.used = floor((n + p + 1)/2),
        nsamp = "best", seed)
cov.mcd(x, cor = FALSE, quantile.used = floor((n + p + 1)/2),
        nsamp = "best", seed)
```

Arguments

`x`	a matrix or data frame.
`cor`	should the returned result include a correlation matrix?
`quantile.used`	the minimum number of the data points regarded as `good` points.
`method`	the method to be used – minimum volume ellipsoid, minimum covariance determinant or classical product-moment. Using `cov.mve` or `cov.mcd` forces `mve` or `mcd` respectively.
`nsamp`	the number of samples or `"best"` or `"exact"` or `"sample"`. If `"sample"` the number chosen is `min(5*p, 3000)`, taken from Rousseeuw and Hubert (1997). If `"best"` exhaustive enumeration is done up to 5000 samples: if `"exact"` exhaustive enumeration will be attempted however many samples are needed.
`seed`	the seed to be used for random sampling: see `RNGkind`. The current value of `.Random.seed` will be preserved if it is set.

Details

For method `"mve"`, an approximate search is made of a subset of size `quantile.used` with an enclosing ellipsoid of smallest volume; in method `"mcd"` it is the volume of the Gaussian confidence ellipsoid, equivalently the determinant of the classical covariance matrix, that is minimized. The mean of the subset provides a first estimate of the location, and the rescaled covariance matrix a first estimate of scatter. The Mahalanobis distances of all the points from the location estimate for this covariance matrix are calculated, and those points within the 97.5% point under Gaussian assumptions are declared to be `good`. The final estimates are the mean and rescaled covariance of the `good` points.

The rescaling is by the appropriate percentile under Gaussian data; in addition the first covariance matrix has an *ad hoc* finite-sample correction given by Marazzi.

For method `"mve"` the search is made over ellipsoids determined by the covariance matrix of `p` of the data points. For method `"mcd"` an additional improvement step suggested by Rousseeuw and van Driessen (1997) is used, in which once a subset of size `quantile.used` is selected, an ellipsoid based on its covariance is tested (as this will have no larger a determinant, and may be smaller).

Value

A list with components

`center`	the final estimate of location.
`cov`	the final estimate of scatter.
`cor`	(only is `cor = TRUE`) the estimate of the correlation matrix.
`sing`	message giving number of singular samples out of total
`crit`	the value of the criterion on log scale. For MCD this is the determinant, and for MVE it is proportional to the volume.
`best`	the subset used. For MVE the best sample, for MCD the best set of size `quantile.used`.
`n.obs`	total number of observations.

Author(s)

B.D. Ripley

References

P. J. Rousseeuw and A. M. Leroy (1987) *Robust Regression and Outlier Detection.* Wiley.

A. Marazzi (1993) *Algorithms, Routines and S Functions for Robust Statistics.* Wadsworth and Brooks/Cole.

P. J. Rousseeuw and B. C. van Zomeren (1990) Unmasking multivariate outliers and leverage points, *Journal of the American Statistical Association*, **85**, 633–639.

P. J. Rousseeuw and K. van Driessen (1999) A fast algorithm for the minimum covariance determinant estimator. *Technometrics* **41**, 212–223.

P. Rousseeuw and M. Hubert (1997) Recent developments in PROGRESS. In *L1-Statistical Procedures and Related Topics* ed Y. Dodge, IMS Lecture Notes volume **31**, pp. 201–214.

See Also

`lqs`

Examples

```
data(stackloss)
set.seed(123)
cov.rob(stackloss)
cov.rob(stack.x, method = "mcd", nsamp = "exact")
```