rms Package Reference Card

Notation

`d` : a data frame with nice `label()`s, `level()`s, and `units()`s for variables (type `?HmiscOverview` to see an overview of the `Hmisc` package)

`y` : an uncensored response variable

`x1,x2,x3` : predictor variables (binary, factor, character, continuous)

`f` : a fit from an `rms` fitting function

`Help` : tells how to get detailed documentation on individual functions from the R command line. When there is no `Help` comment for a function below, type `?functionname` to obtain documentation.

Setting Up

Accessing the Package

```
install.packages('rms')    # one-time only
require(rms) # automatically attaches Hmisc
```

Data From a Fully Prepared Data Frame

```
dd ← datadist(d)       # compute data distribution summary
options(datadist='dd') # for plotting
f ← ols(y ~ x1 + x2*x3, data=d)
# Best not to attach d
. . .
```

Data from a Data Frame with Some Changes or Additions Needed

```
d  ← upData(d, rename=c(smoking='smoke'),
  drop=c('var1','var2'),
  sex =factor(sex, 0:1, c('non-current smoker',
                          'current smoker')),
  units =c(age='years', fev='L', height='inches'),
  labels=c(fev='Forced Expiratory Volume'))
# upData is in the Hmisc package
dd ← datadist(d)
options(datadist='dd')
...
```


Data from a Collection of Vectors

```
dd ← datadist(x1, x2, x3)
options(datadist='dd')
f ← lrm(y ~ rcs(x1,4)*x2)

Help: ?rmsOverview, ?datadist, ?rms
```

Special Model Fitting Functions

`ols` : ordinary and penalized least squares

`lrm` : binary and ordinal logistic regression with optional penalization[1]

`cph` : Cox proportional hazards model

`psm` : parametric survival models

`bj` : Buckey-James right-censored least squares model

`Glm` : Generalized linear model (version of `glm` that works with `rms`)

`Gls` : Generalized least squares (version of `gls` from `nlme` package)

`Rq` : Quantile regression (version of `rq` from `quantreg` package)

`Help` : `?ols`, `?lrm`, `?cph`, `?psm`, `?bj`, `?Glm`, `?Gls`, `?Rq`, `?rms`

Transformations of Predictors

`rcs(x1, 4)` : restricted cubic spline with 4 default knots

`rcs(x1, c(1,2,6,9))` : rcs with user-specified knot locations

`lsp(x1, c(1,2,6))` : linear spline (knot locations mandatory for `lsp`)

`pol(x1, p)` : ordinary polynomial of degree `p`

`scored(x1)` : expand categorical predictor having k numeric levels into linear term and $k-2$ dummy variables

[1] `lrm` fits the proportional odds model. In conjunction with the `cr.setup` function it fits the continuation ratio model.

`strat(x1)` : stratify on `x1` for `cph`

many R functions : e.g., `pmin(x1,4)`, `rcs(pmax(x1, 0), 4)`; plots will have innermost variables on axes

restricted interactions : `%ia%`

`Help` : `?rcs` etc., `?rms.trans`

Functions Operating on Fit Objects

Basic Generic Functions & Predictions

`print` : print model fit

`coef` : print coefficient vector

`fitted` : extract predicted values

`resid` : extract residuals and do goodness of fit tests

`formula` : print model formula

`specs` : print details about model specification (e.g., knots, categories, d.f.). Add `,long=TRUE` to see `datadist` info.

`predict` : predicted values and confidence limits[2]. For `ols` fits can get CLs for individuals and means.

`Predict` : predicted values and confidence limits easily varying a subset of predictors and leaving the rest set at default values

`Function` : build an R function that computes predicted values (the linear combination of predictors)

```
g ← Function(f)
g(x1=5:9, x2='drug')  # x3 defaults to median
```

`Help` : `residuals.lrm` etc., `?specs`, `?predict.rms`, `?Predict`, `?Function.rms`

[2] In `rms` all predictions are "safe" as knots and categories are remembered.

Modifying the Covariance Matrix with Optional Allowance for Intra-Cluster Correlation

`bootcov` : bootstrap "nonparametric" covariance matrix

`robcov` : Huber-White robust covariance matrix

```
# Add raw data to fit for resampling by using x=TRUE,
#  y=TRUE
f ← update(f, x=TRUE, y=TRUE)

f2 ← bootcov(f, subject.id, B=100)
anova(f2)
# all functions on f2 use new covariance matrix
```

Partial Wald χ^2 and F (for `ols`) Statistics

```
f ← lrm(y ~ x1 + x3*rcs(x2,4))

specs(f,T)     # shows knots chosen for x2
```

```
         Assumption                    Parameters d.f.
     x1 category      drug placebo                 1
     x3 asis                                       1
     x2 rcspline     0.0417 0.3570 0.6898 0.9563 3
x3 * x2 interaction linear x nonlinear - Ag(B)   3

                    x1 x3     x2
      Low:effect      NA  0 0.2566
       Adjust to    drug  0 0.5034
     High:effect      NA  1 0.7721
 Low:prediction    drug  0 0.0141
High:prediction placebo  1 0.9815
            Low    drug  0 0.0059
           High placebo  1 0.9989
```

```
print(anova(f,x2,x3),'names') # combined test of x2,x3
```

```
%
  Factor              Chi-Square d.f.      P
   x2                 20.95      6    0.0019
    All Interactions  16.81      3    0.0008
    Nonlinear          2.45      4    0.6543
   x3                 56.90      4    <.0001
    All Interactions  16.81      3    0.0008
 TOTAL                59.75      7    <.0001

                                  Tested
 x2,x2',x2'',x3 * x2,x3 * x2',x3 * x2''
 x3 * x2,x3 * x2',x3 * x2''
 x2',x2'',x3 * x2',x3 * x2''
 x3,x3 * x2,x3 * x2',x3 * x2''
 x3 * x2,x3 * x2',x3 * x2''
 x3,x2,x2',x2'',x3 * x2,x3 * x2',x3 * x2''
```

```
plot(anova(f))
lrtest(f, f2)     # likelihood ratio test for nested models

Help: ?anova.rms
```

Predictor Shape Plots

```
f ← lrm(y ∼ rcs(x1,4)*rcs(x2,4) + x3)
# Plot showing effect of x1 (x-axis) on log odds
# 3 curves for 3 values of x2; x3 set to mode or median
p ← Predict(f, x1=., x2=c(2,4,6))
plot(p)  # or plot(p, ∼ x1 | x2) or plot(p, ∼ x1, groups='x2')
# . causes plot to plot from 10th smallest to 10th
# largest value of x1 by default. Use x1=seq(...) otherwise.

# 3-D plot varying x1 and x2. Show prob. instead of logit.
p ← Predict(f, x1=., x2=., np=50, fun=plogis)
bplot(p, method='image', ylab='Prob.')
# plogis is equivalent to fun=function(x)1/(1+exp(-x))

# If x3 is discrete, make separate curve for each unique value
plot(Predict(f, x1=., x3=., conf.int=FALSE))

# Show shape and strength of all predictors, setting others
#  to reference values, by using common y-axis scale.
#  ref.zero shifts y to zero when x=reference value.
plot(Predict(f, ref.zero=TRUE))

# Show two kinds of CLs for ols fits
g  ← ols(y ∼ rcs(x1,5) + x2)
p1 ← Predict(f, x1=., conf.type='mean')
p2 ← Predict(f, x1=., conf.type='individual')
p  ← rbind(mean=p1, individual=p2)
# To get one graph using superposition:
plot(p, label.curve=FALSE)
plot(p, ∼ x1 | .set.)    # instead get 2 panels
```

Help: ?Predict, ?plot.Predict, ?bplot, ?lattice, ?rbind.Predict, ?labcurve

Survival Estimates and Curves

For fits from `psm` and `cph` (the latter working fastest if `surv=TRUE` was specified and you don't need accurate standard errors or confidence limits). `survplot` will also plot results from `survfit`. Here `etime` is a right-censored event time variable and `event` is an event/ censoring indicator.

```
f ← psm(Surv(etime, event) ∼ x1 + log(x2+2) + x3,
        dist='lognormal')
# Compute survival curve for x1=10, x2=3, x3='male'
survest(f, data.frame(x1=10, x2=3, x3='male'))
# Add ,times=c(2,4) to get survival only at 2 and 4 years

# Plot cumulative mortality for x1=2 and 8 for males

survplot(f, x1=c(2,8), x3='male', n.risk=TRUE,
```



```
         fun=function(y)1-y,
         ylab='Cumulative Probability')
# x2 defaults to median
```

Help: ?survest.cph, ?survest.psm, ?survplot

Charts Depicting Odds Ratios Hazard Ratios, Differences

```
summary(f)                # inter-quartile range differences
                          # and anti-logs
summary(f, x1=c(2,6))     # effect of increasing x1 from 2 to 6
summary(f, x1=c(2,4,6))# set x1 to 4 when examining x2,x3
                          # important if x1 interacts
plot(summary(f), log=TRUE)# odds ratio chart if f from lrm,
                          # log scale
```

Help: ?summary.rms

Nomogram

```
# Obtain predicted probabilities from logistic model
#  for any values of predictors in the observed range
# Override default axis for one of the variables
nom ← nomogram(f, x2=c(1,3,5,7,9),
               fun=plogis, funlabel='Prob[Y=1]')
plot(nom)    # print(nom) to show points tables
```

General Contrasts and Confidence Limits for Effects

Compare a subject with `x1=5` on drug to a subject with `x1=10` on placebo, accounting for nonlinearity and interaction.

```
f ← lrm(y ∼ rcs(x1,4)*x2 + x3)
contrast(f, list(x1=5,  x2='drug'   ),
            list(x1=10, x2='placebo'))
```

Compute drug effects separately for several values of `x1`. Also print the average effect over these levels of `x1`, with CLs.

```
for(type in c('individual','average'))
  print(contrast(f, list(x1=1:10, x2='drug'),
                    list(x1=1:10, x2='placebo'),
        type=type, conf.int=0.99))
```

Plot drug effects over values of `x1`, with error bars.


```
k ← contrast(f, list(x1=1:10, x2='drug'),
                list(x1=1:10, x2='placebo'))
xYplot(Cbind(Contrast, Lower, Upper) ∼ x1, data=k,
       ylab='Drug - Placebo')      # xYplot in Hmisc
# Use Cbind(exp(Contrast),exp(Lower),exp(Upper)) to get odds
  ratios
```

Use "no difference" contrasts to compute estimates of mean response for x1=5, x3=2 averaged over treatment (`x2`) groups, using observed frequencies of treatments as weights.

```
contrast(f, list(x1=5, x3=2, x2=levels(x2)),
         type='average', weights=table(x2))
```

If if `x2` has > 2 levels, and still allowing `x2` to interact nonlinearly with `x1`, test whether there is an association between `x1` and response for subjects on placebo (3 d.f.). Then test whether there is a difference between drug and placebo at any `x1` (4 d.f.).

```
x1s <- 1:10 # values must span the space of all x1 basis functions
contrast(f, list(x1=x1s, x2='placebo'),
            list(x1=1,   x2='placebo'),# pick one value for x1
            type='joint')
contrast(f, list(x1=x1s, x2='drug'),
            list(x1=x1s, x2='placebo'), type='joint')
```

Help: ?contrast.rms

Model Validation

`f` must contain the raw data to allow resampling. `validate` estimates the likely future performance of the model based on statistical indexes. `calibrate` does likewise for computing overfitting-corrected calibration (predicted vs. observed) curves. Below `f` must be the most full model examined. To validate a model derived from backward stepdown, specify the full model and `bw=TRUE` to `validate, calibrate`.

```
f ← update(f, x=TRUE, y=TRUE)
validate(f, B=140)
cal ← calibrate(f, B=150)
```

Help: ?validate, ?calibrate

Other Functions

Function	Purpose
`gendata`	Generate data for obtaining predictions
`fastbw`	Fast backward step-down var. selection
`sensuc`	Sensitivity analysis for an unmeasured confounder in `lrm` model
`which.influence`	Which obs. are overly influential
`latex`	LaTeX representation of fitted model
`Hazard`	R function analytic representation of fitted hazard function (for `psm`)
`Survival`	R function representation of fitted survival function (for `psm`, `cph`)
`Quantile`	R function representation of fitted function for quantiles of survival time (for `psm`, `cph`)
`Mean`	R function representation of fitted function for mean survival time or mean of an `lrm`
`val.prob`	External validation of a probability model
`val.surv`	External validation of a survival model
`vcov`	Compute/retrieve var-cov matrix for fit
`vif`	Variance inflation factors for fit

For More Information

The central web page for the `rms` package, for updates to this card, and for information on statistical methodology is `biostat.mc.vanderbilt.edu/rms`.

Please communicate corrections and improvements to Frank Harrell at `f.harrell@vanderbilt.edu`.

Version: September 13, 2009