

- (1) Quantile Regression in R.pdf
 - (2) Poisson Regression.pdf
 - (3) Beta Regression in R.pdf
 - (4) Beta Regression Models.pdf
 - (5) Gamma Regression.pdf
 - (6) Logit and Probit.pdf
 - (7) Gamma Regression 2.pdf
 - (8) Regression models with count outcomes.pdf
 - (9) VARs.pdf
 - (10) Unit Root Tests.pdf
 - (11) Panel Data Econometrics.pdf
- Bootstrapping in R.pdf
- Cross Validation for Predictive Analytics with R.pdf
- Diff in Diff.pdf
- Fixed Effects.pdf
- Impulse Response Functions.pdf
- Introduction to Bootstrapping in R.pdf
- Marginal Effects.pdf

QUANTILE REGRESSION IN R: A VIGNETTE

ROGER KOENKER

ABSTRACT. Quantile regression is an evolving body of statistical methods for estimating and drawing inferences about conditional quantile functions. An implementation of these methods in the R language is available in the package `quantreg`. This vignette offers a brief tutorial introduction to the package. R and the package `quantreg` are open-source software projects and can be freely downloaded from CRAN: <http://cran.r-project.org>.

1. INTRODUCTION

Beran's (2003) provocative definition of statistics as "the study of algorithms for data analysis" elevates computational considerations to the forefront of the field. It is apparent that the evolutionary success of statistical methods is to a significant degree determined by considerations of computational convenience. As a result, design and dissemination of statistical software has become an integral part of statistical research. Algorithms are no longer the exclusive purview of the numerical analyst, or the proto-industrial software firm; they are an essential part of the artisanal research process. Fortunately, modern computing has also transformed the software development process and greatly facilitated collaborative research; the massive collective international effort represented by the R project exceeds the most idealistic Marxist imagination.

Algorithms have been a crucial part of the research challenge of quantile regression methods from their inception in the 18th century. Stigler [1984] describes an amusing episode in 1760 in which the itinerant Croatian Jesuit Rudjer Boscovich sought computational advice in London regarding his nascent method for median regression. Ironically, a fully satisfactory answer to Boscovich's questions only emerged with dawn of modern computing. The discovery of the simplex method and subsequent developments in linear programming have made quantile regression methods competitive with traditional least squares methods in terms of their computational effort. These computational developments have also played a critical role in encouraging a deeper appreciation of the statistical advantages of these methods.

Since the early 1980's I have been developing software for quantile regression: initially for the S language of Chambers and Becker (1984), later for its commercial manifestation Splus, and since 1999 for the splendid open source dialect R , initiated by Ihaka and Gentleman [1996] and sustained by the R Development Core Team [2003]. Although there is now some functionality for quantile regression in most of the major commercial statistical packages, I have a natural predilection for the R environment and the software that I have developed for R . In what follows, I have tried to provide a brief tutorial introduction to this environment for quantile regression.

2. WHAT IS A VIGNETTE?

This document was written in the Sweave format of Leisch [2003]. Sweave is an implementation designed for R of the literate programming style advocated by Knuth [1992]. The format permits a natural interplay between code written in R, the output of that code, and commentary on the code. Sweave documents are preprocessed by R to produce a L^AT_EX document that may then be processed by conventional methods. Many R packages now have Sweave vignettes describing their basic functionality. Examples of vignettes can be found for many of the R packages including this one for the **quantreg** package in the source distribution directory `inst/doc`.

3. GETTING STARTED

I will not attempt to provide another introduction to R. There are already several excellent resources intended to accomplish this task. The books of Dalgaard [2002] and Venables and Ripley [2002] are particularly recommended. The CRAN website link to contributed documentation also offers excellent introductions in several languages.

R is an open source software project and can be freely downloaded from the CRAN website along with its associated documentation. For unix-based operating systems it is usual to download and build R from source, but binary versions are available for most computing platforms and can be easily installed. Once R is running the installation of additional packages is quite straightforward. To install the quantile regression package from R one simply types,

```
> install.packages("quantreg")
```

Provided that your machine has a proper internet connection and you have write permission in the appropriate system directories, the installation of the package should proceed automatically. Once the **quantreg** package is installed, it needs to be made accessible to the current R session by the command,

```
> library(quantreg)
```

These procedures provide access to an enormous variety of specialized **packages** for statistical analysis. As we proceed a variety of other packages will be called upon.

Online help facilities are available in two modalities. If you know precisely what you are looking for, and would simply like to check the details of a particular command you can, for example, try:

```
> help(package="quantreg")
> help(rq)
```

The former command gives a brief summary of the available commands in the package, and the latter requests more detailed information about a specific command. An convenient shorthand for the latter command is to type simply `?rq`. More generally one can initiate a web-browser help session with the command,

```
> help.start()
```

and navigate as desired. The browser approach is better adapted to exploratory inquiries, while the command line approach is better suited to confirmatory ones.

A valuable feature of R help files is that the examples used to illustrate commands are executable, so they can be pasted into an R session, or run as a group with a command like,

```
> example(rq)
```

The examples for the basic `rq` command include an analysis of the Brownlee stack-loss data: first the median regression, then the first quantile regression is computed, then the full quantile regression process. A curious feature of this often analysed data set, but one that is very difficult to find without quantile regression fitting, is the fact the 8 of the 21 points fall exactly on a hyperplane in 4-space.

The second example in the `rq` helpfile computes a weighted univariate median using randomly generated data. The original Engel [1857] data on the relationship between food expenditure and household income is considered in the third example. The data is plotted and then six fitted quantile regression lines are superimposed on the scatterplot. The final example illustrates the imposition of inequality constraints on the quantile regression coefficients using a simulated data set.

Let's consider the median regression results for the Engel example in somewhat more detail. Executing,

```
> data.engel
> fit1 <- rq(foodexp ~ income, tau = .5, data = engel)
```

assigns the output of the median regression computation to the object `fit1`. In the command `rq()` there are also many options. The first argument is a "formula" that specifies the model that is desired. In this case we wanted to fit a simple bivariate linear model so the formula is just $y \sim x$, if we had two covariates we could say, e.g. $y \sim x+z$. Factor variables, that is variables taking only a few discrete values, are treated specially by the formula processing and result in a group of indicator (dummy) variables.

If we would like to see a concise summary of the result we can simply type,

```
> fit1
Call:
rq(formula = foodexp ~ income, tau = 0.5, data = engel)
```

```
Coefficients:
(Intercept)      income
81.4822474    0.5601806
```

```
Degrees of freedom: 235 total; 233 residual
```

By convention for all the R linear model fitting routines, we see only the estimated coefficients and some information about the model being estimated. To obtain a more detailed evaluation of the fitted model, we can use,

```
> summary(fit1)
Call: rq(formula = foodexp ~ income, tau = 0.5, data = engel)

tau: [1] 0.5

Coefficients:
            coefficients lower bd   upper bd
```

(Intercept)	81.48225	53.25915	114.01156
income	0.56018	0.48702	0.60199

The resulting table gives the estimated intercept and slope in the first column and confidence intervals for these parameters in the second and third columns. By default, these confidence intervals are computed by the rank inversion method described in Koenker [2005], Section 3.4.5. To extract the residuals or the coefficients of the fitted relationship we can write,

```
> r1 <- resid(fit1)
> c1 <- coef(fit1)
```

They can then be easily used in subsequent calculations.

4. OBJECT ORIENTATION

A brief digression on the role of object orientation in R is perhaps worthwhile at this juncture. Expressions in R manipulate objects, objects may be data in the form of vectors, matrices or higher order arrays, but objects may also be functions, or more complex collections of objects. Objects have a class and this `clss` identifier helps to recognize their special features and enables functions to act on them appropriately. Thus, for example, the function `summary` when operating on an object of class `rq` as produced by the function `rq` can act quite differently on the object than it would if the object were of another class, say `lm` indicating that it was the product of least squares fitting. Summary of a data structure like a matrix or `data.frame` would have yet another intent and outcome. In the earlier dialects of S and R methods for various classes were distinguished by appending the class name to the method separated by a “.”. Thus, the function `summary.rq` would summarize an `rq` object, and `summary.lm` would summarize an `lm` object. In either case the main objective of the summary is to produce some inferential evidence to accompany the point estimates of parameters. Likewise, plotting of various classes of R objects can be carried out by the expression `plot(x)` with the expectation that the plot command will recognize the class of the object `x` and proceed accordingly. More recently, Chambers [1998] has introduced an elegant elaboration of the class, method-dispatch framework for S and R .

Assignment of objects is usually accomplished by the operator `<-`, and once assigned these new objects are available for the duration of the R session, or until they are explicitly removed from the session. R is an open source language so *all* of the source files describing the functionality of the language are ultimately accessible to the individual user, and users are free to modify and extend the functionality of the language in any way they see fit. To accomplish this one needs to be able to find functions and modify them. This takes us somewhat beyond the intended tutorial scope of this vignette, however suffice it to say that most of the functions of the `quantreg` package you will find used below, can be viewed by simple typing the name of the function perhaps concatenated with a class name.

5. FORMAL INFERENCE

There are several alternative methods of conducting inference about quantile regression coefficients. As an alternative to the rank-inversion confidence intervals, one can obtain a more conventional looking table of coefficients, standard errors, t-statistics, and p-values using the `summary` function:

```
> summary(fit1, se = "nid")
Call: rq(formula = foodexp ~ income, tau = 0.5, data = engel)

tau: [1] 0.5

Coefficients:
            Value    Std. Error t value Pr(>|t|)
(Intercept) 81.48225 19.25066   4.23270 0.00003
income       0.56018  0.02828  19.81032 0.00000
```

The standard errors reported in this table are computed as described in Section 3.2.3 for the quantile regression sandwich formula, and using the Hall-Sheather bandwidth rule. To obtain the Powell kernel version of the covariance matrix estimate, one specifies the option `se="ker"` in the `summary` command. It is also possible to control the bandwidths employed with the `bandwidth` option. Another option available in `summary.rq` is to compute bootstrapped standard errors. This is accomplished by specifying the option `se="boot"`. There are currently three flavors of the bootstrap available: the standard xy -pair bootstrap, the Parzen, Wei, and Ying [1994] version, and the Markov chain marginal bootstrap of He and Hu [2002] and Kocherginsky, He, and Mu [2004]. There is also the ability to specify m out n versions of the bootstrap in which the sample size of the bootstrap samples is different from (typically smaller than) the original sample size. This “subsampling” approach has a number of advantages, not the least of which is that it can be considerably faster than the full n out of n version. By default `summary` also produces components estimating the full covariance matrix of the estimated parameters and its constituent pieces. For further details, see the documentation for `summary.rq`. In the case of the bootstrap methods the full matrix of bootstrap replications is also available.

There are several options to the basic fitting routine `rq`. An important option that controls the choice of the algorithm used in the fitting is `method`. The default is `method = "br"` which invokes a variant of the Barrodale and Roberts [1974] simplex algorithm described in Koenker and d’Orey [1987]. For problems with more than a few thousand observations it is worthwhile considering `method = "fn"` which invokes the Frisch-Newton algorithm described in Portnoy and Koenker [1997]. Rather than traversing around the exterior of the constraint set like the simplex method, the interior point approach embodied in the Frisch-Newton algorithm burrows from within the constraint set toward the exterior. Instead of taking steepest descent steps at each intersection of exterior edges, it takes Newton steps based on a log-barrier Lagrangian form of the objective function. Special forms of Frisch-Newton are available for problems that include linear inequality constraints and for problems with sparse design matrices. For extremely large problems with plausibly exchangeable observations `method = "pfn"` implements a version of the Frisch-Newton algorithm with a preprocessing step that can further speed things up considerably.

In problems of moderate size where the default simplex option is quite practical, the parametric programming approach to finding the rank inversion confidence intervals can be rather slow. In such cases it may be advantageous to try one of the other inference methods based on estimation of the asymptotic covariance matrix, or to consider the bootstrap. Both approaches are described in more detail below.

To provide a somewhat more elaborate visualization of the Engel example consider an example that superimposes several estimated conditional quantile functions on the Engel data scatterplot. In the resulting figure the median regression line appears as a solid (blue) line, and the least squares line as a dashed (red) line. The other quantile regression lines appear in grey. Note that the plotting of the fitted lines is easily accomplished by the convention that the command **abline** looks for a pair of coefficients, which if found are treated as the slope and intercept of the plotted line. There are many options that can be used to further fine tune the plot. Looping over the quantiles is also conveniently handled by R's **for** syntax.

Often it is useful to compute quantile regressions on a discrete set of τ 's; this can be accomplished by specifying **tau** as a vector in **rq**:

```
> xx <- income - mean(income)
> fit1 <- summary(rq(foodexp~xx,tau=2:98/100))
> fit2 <- summary(rq(foodexp~xx,tau=c(.05, .25, .5, .75, .95)))
```

The results can be summarized as a plot.

```
> pdf("engelcoef.pdf",width=6.5,height=3.5)
> plot(fit1,mfrow = c(1,2))
> dev.off()
```

or by producing a latex-formatted table.

```
> latex(fit2, caption="Engel's Law", transpose=TRUE)
```

The **pdf** command preceding the plot tells R that instructions for the plotting should be written in encapsulated pdf format and placed in the file **engelcoef.pdf**. Such files are then conveniently included in L^AT_EX documents, for example. The **dev.off()** command closes the current pdf device and concludes the figure. The horizontal lines in the coefficient plots represent the least squares fit and its associated confidence interval.

In the one-sample setting we know that integrating the quantile function over the entire domain [0,1] yields the mean of the (sample) distribution,

$$\mu = \int_{-\infty}^{\infty} x dF(x) = \int_0^1 F^{-1}(t) dt.$$

Similarly, in the coefficient plots we may expect to see that integrating individual coefficients yields roughly mean effect as estimated by the associated least squares coefficient. One should be cautious, however, about this interpretation in very heterogeneous situations. For the Engel data, note that the least squares intercept is significantly above any of the fitted quantile regression curves in our initial scatter plot. The least squares fit is strongly affected by the two outlying observations with relatively low food expenditure; their attraction tilts the fitted line so its intercept drawn upward. In fact, the intercept for the Engel model is difficult to interpret since it asks us to consider food expenditure for households with zero income. Centering the covariate observations so they have mean zero, as we have done prior to computing **fit1** for the coefficient plot restores a reasonable interpretation of the intercept parameter. After centering the least squares estimate of the intercept is a prediction of mean food expenditure for a household with mean income, and the quantile regression intercept, $\hat{a}(\tau)$ is a prediction of the τ th quantile of food expenditure for households with mean income. In the terminology of Tukey, the "intercept" has become a "centercept."

```

> library(quantreg)
> data(engel)
> attach(engel)
> plot(income, foodexp, cex=.25, type="n", xlab="Household Income", ylab="Food Expenditure")
> points(income, foodexp, cex=.5, col="blue")
> abline(rq(foodexp~income, tau=.5), col="blue")
> abline(lm(foodexp~income), lty=2, col="red") #the dreaded ols line
> taus <- c(.05,.1,.25,.75,.90,.95)
> for( i in 1:length(taus)){
+   abline(rq(foodexp~income, tau=taus[i]), col="gray")
+ }

```

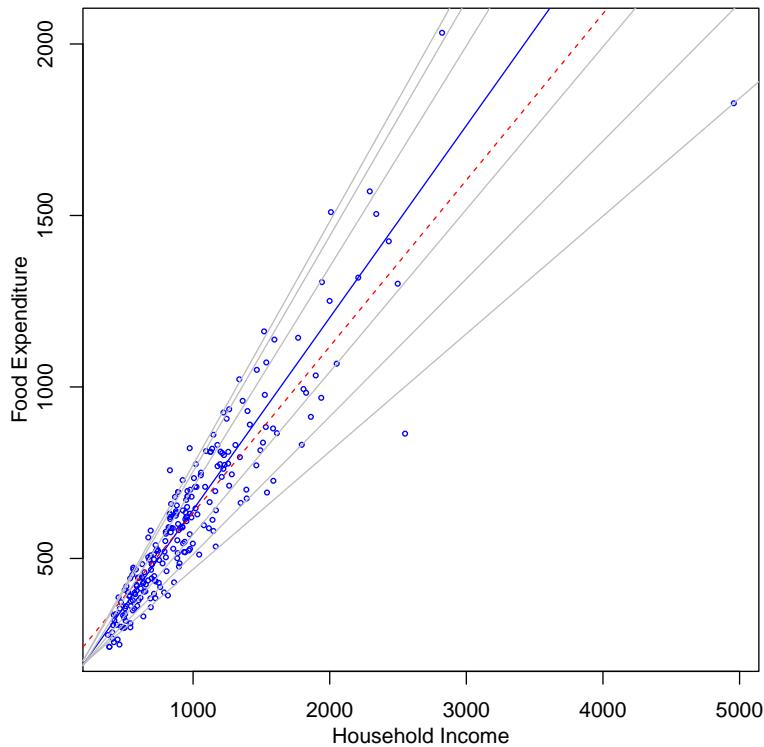


FIGURE 1. Scatterplot and Quantile Regression Fit of the Engel Food Expenditure Data: The plot shows a scatterplot of the Engel data on food expenditure vs household income for a sample of 235 19th century working class Belgian households. Superimposed on the plot are the $\{.05, .1, .25, .75, .90, .95\}$ quantile regression lines in gray, the median fit in solid black, and the least squares estimate of the conditional mean function as the dashed (red) line.

The `latex` command produces a L^AT_EX formatted table that can be easily included in documents. In many instances the plotted form of the results will provide a more economical and informative display. It should again be stressed that since the quantile regression functions and indeed all of R is open source, users can

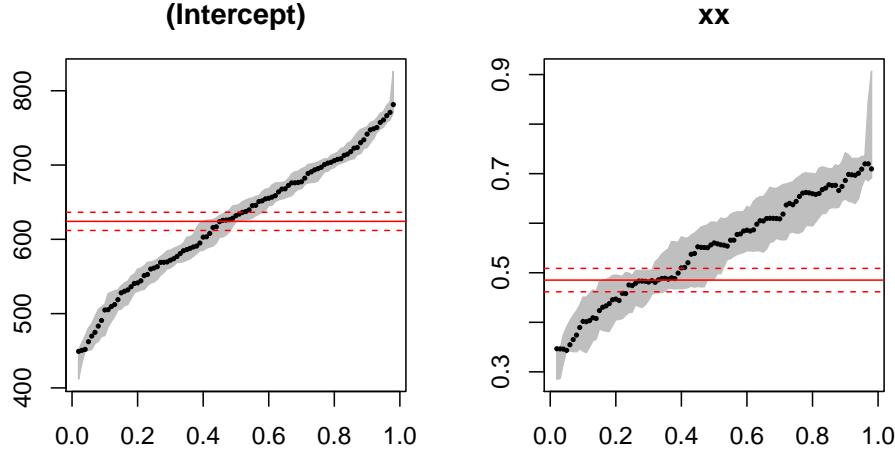


FIGURE 2. Engel Coefficient Plots: the slope and intercept of the estimated linear quantile regression line for the Engel food expenditure data are plotted as a function of τ . Note that the household income variable has been centered at its mean value for this plot, so the intercept is really a centercept and estimates the quantile function of food expenditure conditional on mean income.

TABLE 1. Engel's Law

Quantiles	(Intercept)	xx
0.05	462.223 (450.572, 480.503)	0.343 (0.343, 0.390)
0.25	561.277 (542.572, 570.726)	0.474 (0.420, 0.494)
0.50	631.845 (623.706, 645.461)	0.560 (0.487, 0.602)
0.75	695.123 (678.663, 704.105)	0.644 (0.580, 0.690)
0.95	760.745 (751.092, 771.916)	0.709 (0.674, 0.734)

always modify the available functions to achieve special effects required for a particular application. When such modifications appear to be of general applicability, it is desirable to communicate them to the package author, so they could be shared with the larger community.

If we want to see *all* the distinct quantile regression solutions for a particular model application we can specify a tau outside the range [0,1], e.g.

```
> z <- rq(foodexp~income,tau=-1)
```

This form of the function carries out the parametric programming steps required to find the entire sample path of the quantile regression process. The returned object is of class **rq.process** and has several components: the primal solution in **z\$sol**, and the dual solution in **z\$dso1**. In interactive mode typing the name of

an R object causes the program to print the object in some reasonably intelligible manner determined by the print method designated for the object's class. Again, plotting is often a more informative means of display and so there is a special **plot** method for objects of class **rq.process**.

Estimating the conditional quantile functions of y at a specific values of x is also quite easy. In the following code we plot the estimated empirical quantile functions of food expenditure for households that are at the 10th percentile of the sample income distribution, and the 90th percentile. In the right panel we plot corresponding density estimates for the two groups. The density estimates employ the adaptive kernel method proposed by Silverman [1986] and implemented in the **quantreg** function **akj**. This function is particularly convenient since it permits unequal mass to be associated with the observations such as those produced by the quantile regression process.

Thus far we have only considered Engel functions that are linear in form, and the scatterplot as well as the formal testing has revealed a strong tendency for the dispersion of food expenditure to increase with household income. This is a particularly common form of heteroscedasticity. If one looks more carefully at the fitting, one sees interesting departures from symmetry that would not be likely to be revealed by the typical textbook testing for heteroscedasticity. One common remedy for symptoms like these would be to reformulate the model in log linear terms. It is interesting to compare what happens after the log transformation with what we have already seen.

Note that the flag **log="xy"** produces a plot with log-log axes, and for convenience of axis labeling these logarithms are base 10, so the subsequent fitting is also specified as base 10 logs for plotting purposes, even though base 10 logarithms are *unnatural* and would never be used in reporting numerical results. This looks much more like a classical iid error regression model, although again some departure from symmetry is visible. An interesting exercise would be to conduct some formal testing for departures from the iid assumption of the type already considered above.

6. MORE ON TESTING

Now let's consider some other forms of formal testing. A natural first question is: do the estimated quantile regression relationships conform to the location shift hypothesis that assumes that all of the conditional quantile functions have the same slope parameters. To begin, suppose we just estimate the quartile fits for the Engel data and look at the default output:

```
> fit1 <- rq(foodexp~income,tau=.25)
> fit2 <- rq(foodexp~income,tau=.50)
> fit3 <- rq(foodexp~income,tau=.75)
```

Recall that **rq** just produces coefficient estimates and **summary** is needed to evaluate the precision of the estimates. This is fine for judging whether covariates are significant at particular quantiles but suppose that we wanted to test that the slopes were the same at the three quartiles? This is done with the **anova** command as follows:

```
> anova(fit1, fit2, fit3)
Quantile Regression Analysis of Deviance Table
```

```

> x.poor <- quantile(income,.1) #Poor is defined as at the .1 quantile of the sample distn
> x.rich <- quantile(income,.9) #Rich is defined as at the .9 quantile of the sample distn
> ps <- z$sol[1,]
> qs.poor <- c(c(1,x.poor)%*%z$sol[4:5,])
> qs.rich <- c(c(1,x.rich)%*%z$sol[4:5,])
> #now plot the two quantile functions to compare
> par(mfrow = c(1,2))
> plot(c(ps,ps),c(qs.poor,qs.rich), type="n",
+       xlab = expression(tau), ylab = "quantile")
> plot(stepfun(ps,c(qs.poor[1],qs.poor)), do.points=FALSE, add=TRUE)
> plot(stepfun(ps,c(qs.poor[1],qs.rich)), do.points=FALSE, add=TRUE,
+       col.hor = "gray", col.vert = "gray")
> ## now plot associated conditional density estimates
> ## weights from ps (process)
> ps.wts <- (c(0,diff(ps)) + c(diff(ps),0)) / 2
> ap <- akj(qs.poor, z=qs.poor, p = ps.wts)
> ar <- akj(qs.rich, z=qs.rich, p = ps.wts)
> plot(c(qs.poor,qs.rich),c(ap$dens,ar$dens),type="n",
+       xlab= "Food Expenditure", ylab= "Density")
> lines(qs.rich, ar$dens, col="gray")
> lines(qs.poor, ap$dens, col="black")
> legend("topright", c("poor","rich"), lty = c(1,1), col=c("black","gray"))

```

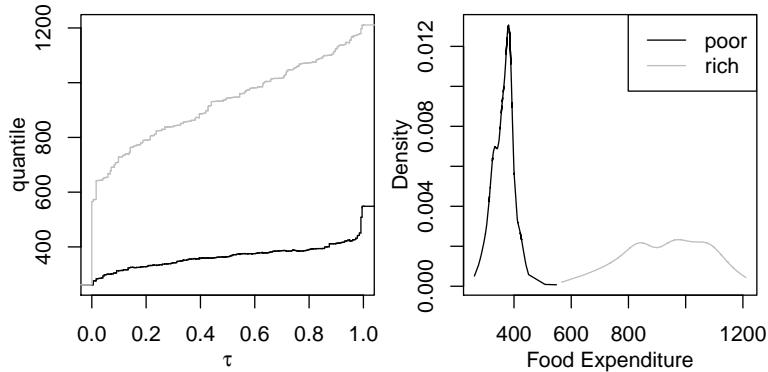


FIGURE 3. Estimated conditional quantile and density functions for food expenditure based on the Engel data: Two estimates are presented one for relatively poor households with income of 504.5 Belgian francs, and the other for relatively affluent households with 1538.99 Belgian francs.

```

Model: foodexp ~ income
Joint Test of Equality of Slopes: tau in { 0.25 0.5 0.75 }

Df Resid Df F value    Pr(>F)
1  2      703  15.557 2.449e-07 ***
---

```

```

> plot(income, foodexp, log="xy", xlab="Household Income", ylab="Food Expenditure")
> taus <- c(.05,.1,.25,.75,.90,.95)
> abline(rq(log10(foodexp)~log10(income),tau=.5),col="blue")
> abline(lm(log10(foodexp)~log10(income)),lty = 3,col="red")
> for( i in 1:length(taus)){
+   abline(rq(log10(foodexp)~log10(income),tau=taus[i]),col="gray")
+ }

```

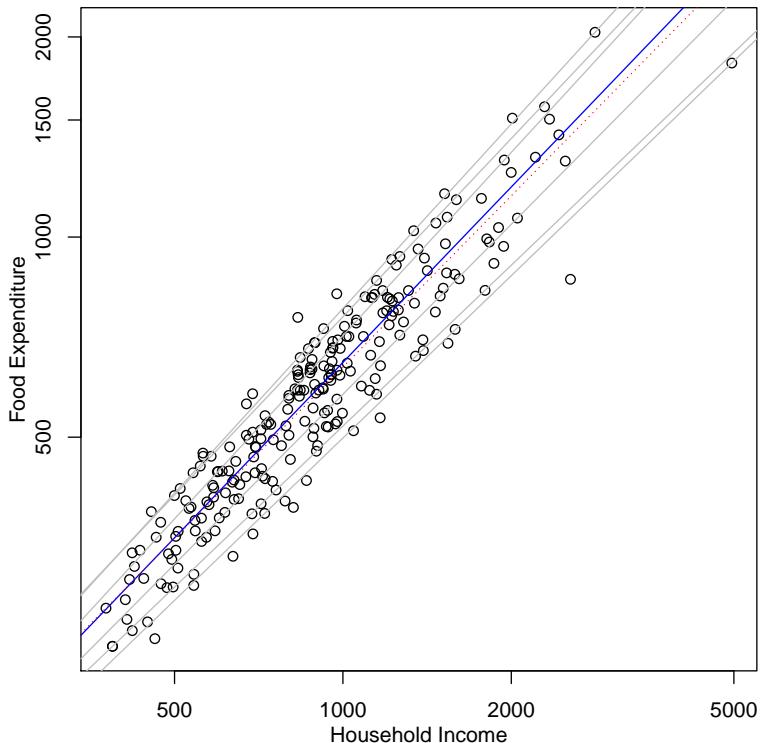


FIGURE 4. Quantile regression estimates for a log-linear version of the Engel food expenditure model.

Signif. codes:

0 *** 0.001 ** 0.01 * 0.05 . 0.1

This is an example of a general class of tests proposed in Koenker and Bassett [1982]. It may be instructive to look at the code for the command `anova.rq` to see how this test is carried out. The Wald approach is used and the asymptotic covariance matrix is estimated using the approach of Hendricks and Koenker [1991]. It also illustrates a general syntax for testing in R adapted to the present situation.

If you have estimated two models with different covariate specifications, but the same τ then `anova(f0,f1)` tests whether the more restricted model is preferred. Note that this requires that the models with fits say `f0` and `f1` are nested. The procedure `anova.rq` attempts to check whether the fitted models are nested, but this is not foolproof. These tests can be carried out either as Wald tests based on the estimated joint asymptotic covariance matrix of the coefficients, or using the

rank test approach described in Gutenbrunner, Jurečková, Koenker, and Portnoy [1993]. A variety of other options are described in the documentation of the function `anova.rq`.

7. INFERENCE ON THE QUANTILE REGRESSION PROCESS

In least squares estimation of linear models it is implicitly assumed that we are able to model the effects of the covariates as a pure location shift, or somewhat more generally as a location and scale shift of the response distribution. In the simplest case of a single binary treatment effect this amounts to assuming that the treatment and control distributions differ by a location shift, or a location-scale shift. Tests of these hypotheses in the two sample model can be conducted using the conventional two-sample Kolmogorov Smirnov statistic, but the appearance of unknown nuisance parameters greatly complicates the limiting distribution theory. Similar problems persist in the extension of these tests to the general quantile regression setting. Using an approach introduced by Khmaladze [1981], Koenker and Xiao [2002] consider general forms of such tests. The tests can be viewed as a generalization of the simple tests of equality of slopes across quantiles described in the previous section.

In this section we briefly describe how to implement these tests in R . The application considered is the quantile autoregression (QAR) model for weekly U.S. gasoline prices considered in ?. The data consists of 699 weekly observations running from August 1990 to February, 2004. The model considered is a QAR(4) model utilizing four lags. We are interested in testing whether the classical location-shift AR(4) is a plausible alternative to the QAR(4) specification, that is whether the four QAR lag coefficients are constant with respect to τ_{α} .

To carry out the test we can either compute the test using the full quantile regression process or on a moderately fine grid of τ_{α} s:

```
> source("gasprice.R")
> x <- gasprice
> n <- length(x)
> p <- 5 # lag length
> X <- cbind(x[(p-1):(n-1)], x[(p-2):(n-2)], x[(p-3):(n-3)], x[(p-4):(n-4)])
> y <- x[p:n]
> T1 <- KhmaladzeTest(y ~ X,taus = -1, nullH="location")
> T2 <- KhmaladzeTest(y ~ X,taus = 10:290/300, nullH="location",se="ker")
taus: 0.03333333 0.03666667 0.04 0.04333333 0.04666667 0.05 0.05333333 0.05666667 0.0
```

When `taus` contains elements outside of the the interval $(0,1)$ then the process is standardized by a simple form of the covariance matrix that assumes iid error. In the second version of the test Powell's kernel form of the sandwich formula estimate is used, see `summary.rq`. The function `KhmaladzeTest` computes both a joint test that *all* the covariate effects satisfy the null hypothesis, and a coefficient by coefficient version of the test. In this example the former component, `T1$Tn` is 4.8. This test has a 1 percent critical value of 5.56, so the test weakly rejects the null. For the Powell form the standardization the corresponding test statistic is more decisive, taking the value 11.03.

Tests of the location-scale shift form of the null hypothesis can be easily done by making the appropriate change in the `nullH` argument of the function.

8. NONLINEAR QUANTILE REGRESSION

Quantile regression models with response functions that are nonlinear in parameters can be estimated with the function **nlrq**. For such models the specification of the model formula is somewhat more esoteric than for ordinary linear models, but follows the conventions of the R command **nls** for nonlinear least squares estimation.

To illustrate the use of **nlrq** consider the problem of estimating the quantile functions of the Frank copula model introduced in Koenker [2005], Section 8.4. We begin by setting some parameters and generating data from the Frank model:

```
> n <- 200
> df <- 8
> delta <- 8
> set.seed(4003)
> x <- sort(rt(n,df))
> u <- runif(n)
> v <- -log(1-(1-exp(-delta))/(1+exp(-delta)*pt(x,df))*((1/u)-1))/delta
> y <- qt(v,df)
```

We plot the observations, superimpose three conditional quantile functions, and then estimate the same three quantile functions and plot their estimated curves as the dashed curves.

9. NONPARAMETRIC QUANTILE REGRESSION

Nonparametric quantile regression is initially most easily considered within a locally polynomial framework. Locally linear fitting is carried out by the following function:

```
> "lprq" <-
+ function(x, y, h, m=50, tau=.5)
+ {
+   xx <- seq(min(x), max(x), length=m)
+   fv <- xx
+   dv <- xx
+   for(i in 1:length(xx)) {
+     z <- x - xx[i]
+     wx <- dnorm(z/h)
+     r <- rq(y~z, weights=wx, tau=tau, ci=FALSE)
+     fv[i] <- r$coef[1.]
+     dv[i] <- r$coef[2.]
+   }
+   list(xx = xx, fv = fv, dv = dv)
+ }
```

If you study the function a bit you will see that it is simply a matter of computing a quantile regression fit at each of m equally spaced x -values distributed over the support of the observed x points. The function value estimates are returned as **fv** and the first derivative estimates at the m points are returned as **dv**. As usual you can specify τ , but now you also need to specify a bandwidth **h**.

Let's begin by exploring the effect of the **h** argument for fitting the motorcycle data.

```

> plot(x,y,col="blue",cex = .25)
> us <- c(.25,.5,.75)
> for(i in 1:length(us)){
+   u <- us[i]
+   v <- -log(1-(1-exp(-delta))/
+             (1+exp(-delta*pt(x,df))*((1/u)-1)))/delta
+   lines(x,qt(v,df))
+ }
> Dat <- NULL
> Dat$x <- x
> Dat$y <- y
> deltas <- matrix(0,3,length(us))
> FrankModel <- function(x,delta,mu,sigma,df,tau){
+   z <- qt(-log(1-(1-exp(-delta))/
+             (1+exp(-delta*pt(x,df))*((1/tau)-1)))/delta,df)
+   mu + sigma*z
+ }
> for(i in 1:length(us)){
+   tau = us[i]
+   fit <- nlrq(y~FrankModel(x,delta,mu,sigma,df=8,tau=tau),
+               data=Dat,tau= tau, start=list(delta=5,
+               mu = 0, sigma = 1),trace=TRUE)
+   lines(x, predict(fit, newdata=x), lty=2, col="green")
+   deltas[i,] <- coef(fit)
+ }

```

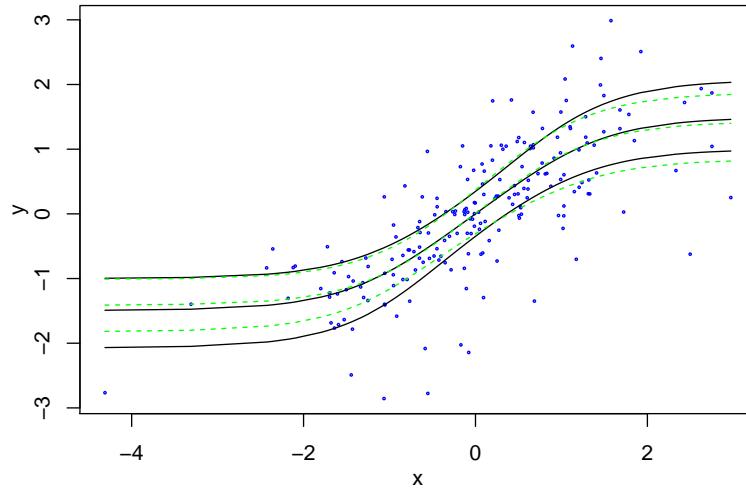


FIGURE 5. Nonlinear Conditional Quantile Estimation of the Frank Copula Model: The solid curves are the true conditional quantile functions and the corresponding estimated curves are indicated by the dashed curves.

```

> library(MASS)
> data(mtcycle)
> attach(mtcycle)
> plot(times,accel,xlab = "milliseconds", ylab = "acceleration")
> hs <- c(1,2,3,4)
> for(i in 1:length(hs)){
+     h = hs[i]
+     fit <- lprq(times,accel,h=h,tau=.5)
+     lines(fit$xx,fit$fv,lty=i)
+ }
> legend(45,-70,c("h=1","h=2","h=3","h=4"),lty=1:length(hs))

```

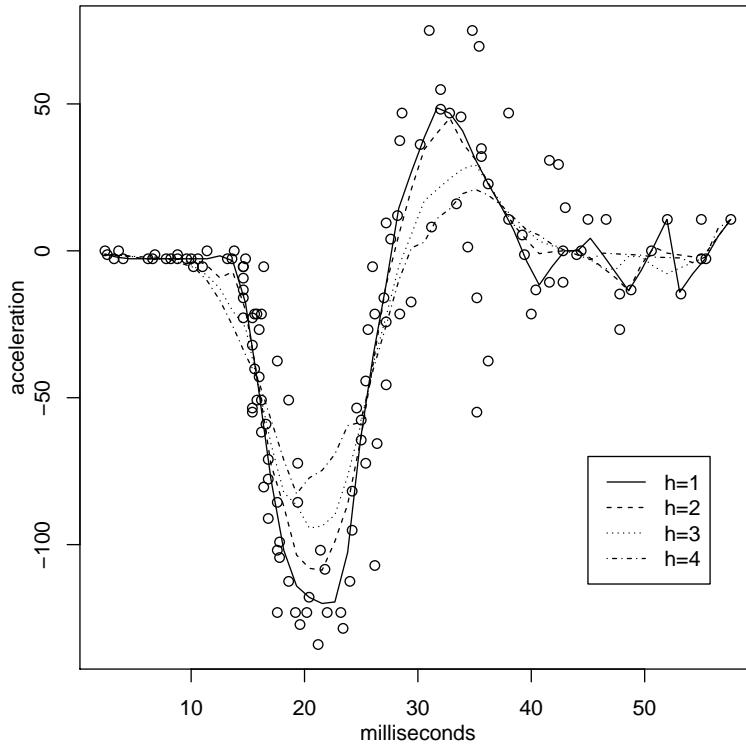


FIGURE 6. Estimation of a locally linear median regression model for the motorcycle data: four distinct bandwidths

Fitting derivatives, of course, requires somewhat larger bandwidth and larger sample size to achieve the same precision as function fitting. It is a straightforward exercise to adapt the function `lprq` so that it does locally quadratic rather than locally linear fitting.

Another simple, yet quite general, strategy for nonparametric quantile regression uses regression splines. The function `bs()` in the package `splines` gives a very flexible way to construct B-spline basis expansions. For example you can fit a new motorcycle model like this:

```

> library(splines)
> plot(times,accel,xlab = "milliseconds", ylab = "acceleration",type="n")
> points(times,accel,cex = .75)
> X <- model.matrix(accel ~ bs(times, df=15))
> for(tau in 1:3/4){
+     fit <- rq(accel ~ bs(times, df=15), tau=tau, data=mcycle)
+     accel.fit <- X %*% fit$coef
+     lines(times,accel.fit)
+ }

```

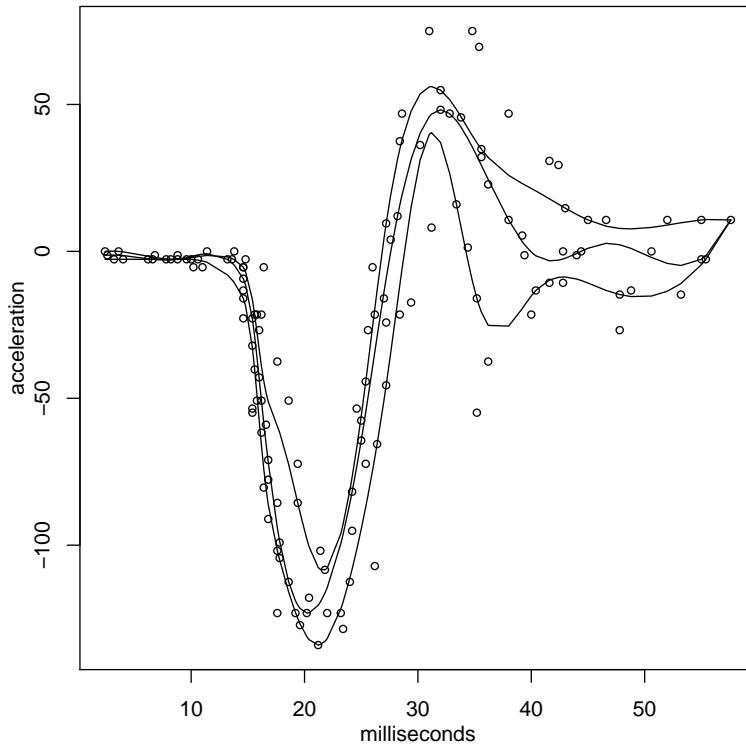


FIGURE 7. B-spline estimates of the three conditional quartile functions of the motorcycle data

The fitted conditional quantile functions do reasonably well except in the region beyond 50 milliseconds where the data is so sparse that all the quantile curve want to coalesce.

This procedure fits a piecewise cubic polynomial with 15 knots (breakpoints in the third derivative) arranged at quantiles of the x 's. (You can also explicitly specify the knot sequence and the order of the spline using the optional arguments to `bs`.) In this instance we have estimated three quartile curves for the B-spline model; a salient feature of this example is the quite dramatic variable in variability over the time scale. There is essentially no variability for the first few milliseconds, but quite substantial variability after the crash. One advantage of the B-spline approach is that it is very easy to add a partially linear model component. If there

were another covariate, say z , it could be added as a parametric component using the usual **formula** syntax,

```
> fit <- rq(y ~ bs(x,df=5)+z,tau=.33)
```

Another appealing approach to nonparametric smoothing involves penalty methods. Koenker, Ng, and Portnoy [1994] describe total variation penalty methods for fitting univariate functions; Koenker and Mizera [2004], extend to approach to bivariate function estimation. Again, partially linear models are easily adapted, and there are also easy ways to impose monotonicity and convexity on the fitted functions. In some applications it is desirable to consider models that involve nonparametric fitting with several covariates. A tractable approach that has been explored by many authors is to build additive models. This approach is available in R for least squares fitting of smoothing spline components in the **mgcv** and **gss** packages. A prototype package for quantile regression models of this type is available using the **rqss** function of the **quantreg** package.

The function **rqss** offers a formula interface to nonparametric quantile regression fitting with total variation roughness penalties. Consider the running speed of mammals example from Section 7. The objective is to estimate a model for the upper envelope of the scatterplot, a model that would reflect best evolutionary practice in mammalian ambulatory efficiency. In contrast to the least squares analysis of Chappell [1989] where they are omitted, no special allowance is made for the “specials” indicated by the plotting character **s** or “hoppers” indicated by **h**. The data is plotted on a (natural) log scale, the model is fit using $\lambda = 1$ as the penalty parameter, and the fitted curve is plotted using a special plotting function that understands the structure of the objects returned from **rqss**. The estimated turning point of the piecewise linear fitted function occurs at a weight of about 40 Kg.

```
> data(Mammals)
> attach(Mammals)
> require(MatrixModels)
```

Bivariate nonparametric fitting using the triogram methods described in Section 7 can be handled in a similar manner. If we consider the Cobar mining data from Green and Silverman [1994]

The **qss** term in this case case requires both x and y components. In addition one needs to specify a smoothing parameter λ , and the parameter **ndum** may be used to specify the number of artificial vertices introduced into the fitting procedure in addition to the actual observations. These artificial vertices contribute to the penalty term, but not to the fidelity.

By default the fit is rendered as a contour plot, but there are also two forms of perspective plots. A conventional R **persp** plot can be obtained by passing option **render** = "persp" to the plot command. More adventurous R gonauts are encouraged to explore the option **render** = "rgl", which produces a perspective plot in the dynamic graphics interface to the open GL library provided by the package **rgl**. Of course this package must be installed. A demonstration of how to create animations of **rqss** triogram output using **rgl** is available by running the command **demo(cobar)**.

Another advantage of the penalty approach embodied in **rqss** is that it is straightforward to impose additional qualitative constraints on the fitted functions.

```

> x <- log(weight)
> y <- log(speed)
> plot(x,y, xlab="Weight in log(Kg)", ylab="Speed in log(Km/hour)", type="n")
> points(x[hoppers],y[hoppers],pch = "h", col="red")
> points(x[specials],y[specials],pch = "s", col="blue")
> others <- (!hoppers & !specials)
> points(x[others],y[others], col="black",cex = .75)
> fit <- rqss(y ~ qss(x, lambda = 1),tau = .9)
> plot(fit)

```

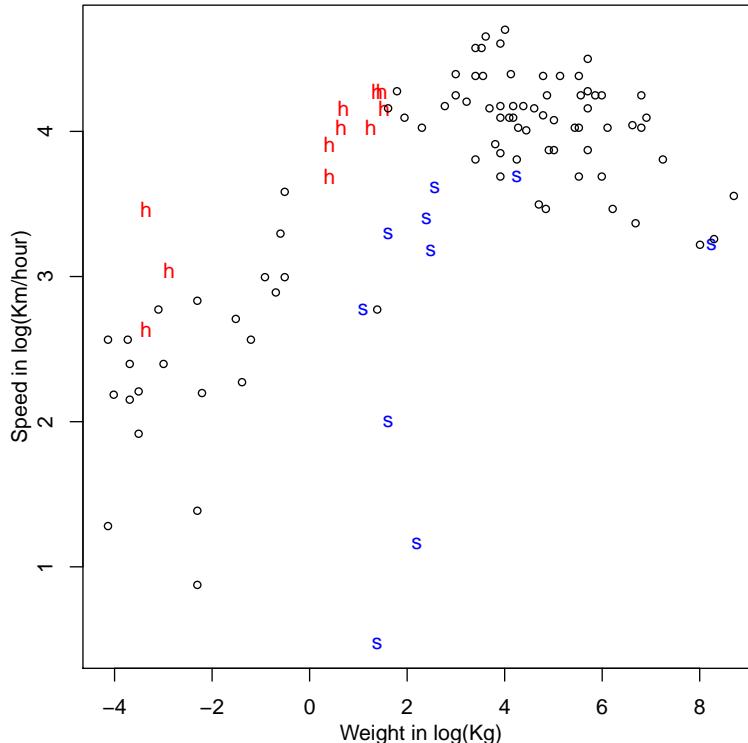


FIGURE 8. Maximal Running Speed of Terrestrial Mammals: The figure illustrates the relationship between adult body mass and maximal running speed for 107 species of terrestrial mammals. The piecewise linear curve is an estimate of the .90 conditional quantile function estimated subject to a constraint on the total variation of the function gradient.

Univariate functions can be constrained to be monotone and/or convex or concave. Bivariate functions can be constrained to be either convex or concave. This functionality is implemented by simply imposing nonnegativity constraints on certain linear combinations of model parameters and illustrates one of many possible applications for such constraints. An interesting open research problem involves formal inference on such constraints.

```
> data(CobarOre)
> fit <- rqss(z ~ qss(cbind(x,y), lambda = .01, ndum=100), data = CobarOre)
> plot(fit, axes = FALSE, xlab = "", ylab = "")
> rm(list=ls())
```

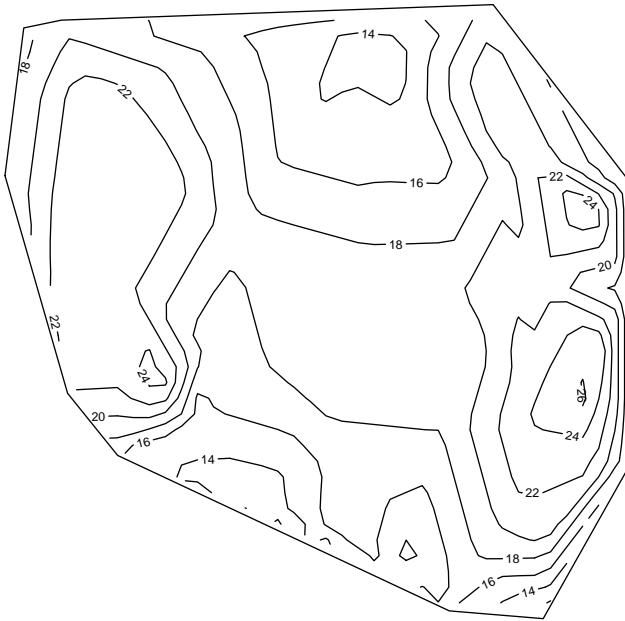


FIGURE 9. Contour plot of a triogram fit of the Cobar mining data.

10. CONCLUSION

A few of the capabilities of the R `quantreg` package have been described. Inevitably, new applications will demand new features and reveal unforeseen bugs. In either case I hope that users will send me their comments and suggestions. This document will be periodically updated and the current version will be made available in the R distribution of `quantreg`. See the R command `vignette()` for details on how to find and view vignettes from within R .

REFERENCES

- I. Barrodale and F. Roberts. Solution of an overdetermined system of equations in the ℓ_1 norm. *Communications of the ACM*, 17:319–320, 1974.
- R. Beran. Impact of the bootstrap on statistical algorithms and theory. *Statistical Science*, pages 175–184, 2003.
- J. M. Chambers. *Programming with Data: A Guide to the S Language*. Springer, 1998.

- Rick Chappell. Fitting best lines to data, with applications to allometry. *J. Theor. Biology*, 138:235–256, 1989.
- Peter Dalgaard. *Introductory Statistics with R*. Springer-Verlag, 2002.
- E. Engel. Die produktions- und konsumptionsverhältnisse des königreichs sachsen. *Zeitschrift des Statistischen Bureaus des Königlich Sächsischen Ministeriums des Innern*, 8:1–54, 1857.
- P. J. Green and B. W. Silverman. *Nonparametric Regression and Generalized Linear Models: A Roughness Penalty Approach*. ChapmnHall:Lnd, 1994.
- C. Gutenbrunner, J. Jurečková, R. Koenker, and S Portnoy. Tests of linear hypotheses based on regression rank scores. *J. of Nonparametric Statistics*, 2:307–33, 1993.
- X. He and F. Hu. Markov chain marginal bootstrap. *J. of Am. Stat. Assoc.*, 97: 783–795, 2002.
- W. Hendricks and R. Koenker. Hierarchical spline models for conditional quantiles and the demand for electricity. *J. of Am. Stat. Assoc.*, 87:58–68, 1991.
- R. Ihaka and R. Gentleman. R: A language for data analysis and graphics. *J. of Computation and Graphical Stat.*, 5:299–314, 1996.
- E. V. Khmaladze. Martingale approach in the theory of goodness-of-fit tests. *Theory of Prob. and its Apps*, 26:240–257, 1981.
- D. E. Knuth. *Literate Programming*. Center for the Study of Language and Information, 1992.
- M. Kocherginsky, X. He, and Y. Mu. Practical confidence intervals for regression quantiles. *J. of Comp. and Graphical Stat.*, 2004. forthcoming.
- R. Koenker. *Quantile Regression*. Cambridge U. Press, 2005.
- R. Koenker and G. Bassett. Robust tests for heteroscedasticity based on regression quantiles. *Econometrica*, 50:43–61, 1982.
- R. Koenker and V. d’Orey. Computing regression quantiles. *Applied Statistics*, 36: 383–393, 1987.
- R. Koenker and I. Mizera. Penalized triograms: Total variation regularization for bivariate smoothing. *J. Royal Stat. Soc. (B)*, 66:145–163, 2004.
- R. Koenker and Z. Xiao. Inference on the quantile regression process. *Econometrica*, 70:1583–1612, 2002.
- R. Koenker, P. Ng, and S. Portnoy. Quantile smoothing splines. *Biometrika*, 81: 673–80, 1994.
- Friedrich Leisch. Sweave, part II: Package vignettes. *R News*, 3(2):21–24, October 2003. URL <http://CRAN.R-project.org/doc/Rnews/>.
- M. I. Parzen, L.J. Wei, and Z. Ying. A resampling method based on pivotal estimating functions. *Biometrika*, 81:341–350, 1994.
- S. Portnoy and R. Koenker. The Gaussian hare and the Laplacian tortoise: Computability of squared-error versus absolute-error estimators, with discussion. *Stat. Science*, 12:279–300, 1997.
- R Development Core Team. *R: A language and environment for statistical computing*, 2003. <http://www.R-project.org>.
- B. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman-Hall: New York, 1986.
- S. Stigler. Boscovich, simpson and a 1760 manuscript note on fitting a linear relation. *Biometrika*, 71:615–620, 1984.

W.N. Venables and B. D. Ripley. *Modern applied statistics with S-PLUS*. Springer-Verlag, fourth edition, 2002.

Chapter 15 Section 2: Poisson Regression

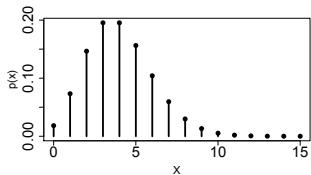
RECALL: The Poisson distribution

Let Y be distributed as a Poisson random variable with the single parameter λ .

$$P(Y = y) = \frac{e^{-\lambda} \lambda^y}{y!} \quad \text{for } y \in \{0, 1, 2, 3, 4, \dots\}$$

Y is a discrete random variable that has probability mass only on the whole numbers.

Suppose $\lambda = 4$, how is Y distributed?



1

2

If $Y \sim \text{Poisson}(\lambda)$,

$$\mathbf{E}(\mathbf{Y}) = \boldsymbol{\lambda} \quad \text{and} \quad \mathbf{V}(\mathbf{Y}) = \boldsymbol{\lambda}.$$

The mean and the variance are equal.

The variance is tied to the mean.

Suppose the mean of this Y depends on the covariate X ...

For example, let Y represent the number of accidents at an intersection, and X represent a characteristic of the intersection (like number of lanes).

If $E[Y|X = 2] < E[Y|X = 4]$, then

$$V[Y|X = 2] < V[Y|X = 4].$$

Or...

if the mean increases with X ,
so does the variance.

Poisson Regression

- When the response variable is a count following a Poisson distribution with a mean that depends on the covariates, we can fit a Poisson regression model.
- Poisson Regression model:

$$\ln(\lambda_i) = \beta_0 + \beta_1 x_{1i} + \dots + \beta_k x_{ki}$$

Or, $Y_i \sim \text{Poisson}(\lambda_i)$ where

$$\lambda_i = e^{\beta_0 + \beta_1 x_{1i} + \dots + \beta_k x_{ki}}$$

and the Poisson parameter λ_i depends on the covariates of each observation (so, each observation can have its own mean).

Thus, the mean depends on the covariates, and the variance depends on the covariates.

- The Poisson regression model is another **GENERALIZED LINEAR MODEL**.

- Instead of a logit function of the Bernoulli parameter π_i (logistic regression), we use a \log_e function of the Poisson parameter λ_i .

$$\lambda_i > 0 \Rightarrow -\infty < \ln(\lambda_i) < \infty$$

- The logit function in the logistic model and the \log_e function in the Poisson model are called the *link* functions for these generalized linear models.

- In this modeling, we assume the $\ln(\lambda_i)$ is linearly related to the independent variables. And that the mean and variance are equal for a given λ_i (as we're using Poisson).

- An iterative process is used to solve the likelihood equations and get maximum likelihood estimates.

3

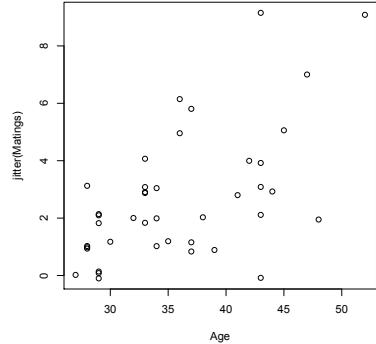
4

Example: Mating of elephants

There is competition for female mates between young and old male elephants.

Because male elephants continue to grow throughout their lives, older elephants are larger and tend to be more successful at mating.

J.H. Poole, Mate Guarding, Reproductive Success and Female Choice in African Elephants, Animal Behavior 37 (1989): 842-49.



Variables:

Response: Matings (number of mates)

Predictor: Age of male elephant (years)

It looks like the number of mates tends to be higher for older elephants, AND there seems to be more variability in the number of mates as age increases.

First, let's look at a scatterplot of the data.

```
> plot(Age,jitter(Matings))
```

5

6

If the dispersion increases with the mean for a count response, then Poisson regression may be a good modeling choice.

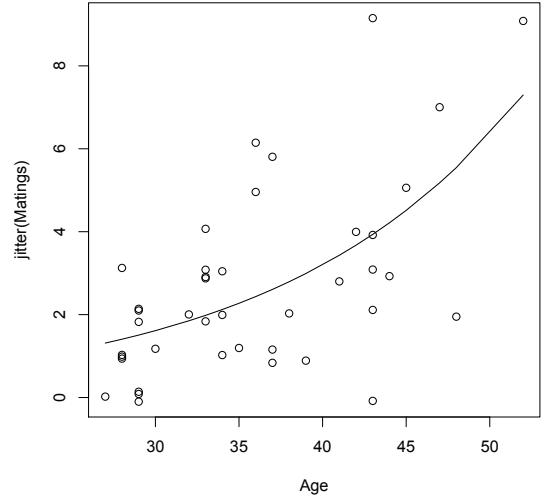
$$\log(\lambda_i) = \hat{\beta}_0 + \hat{\beta}_1 x$$

```
> glm.out=glm(Matings~Age,family=poisson)

## Get and plot the fitted mean structure:
> coeff=coef(glm.out)
> coeff
(Intercept)      Age
-1.57955278  0.06859472

> xvalues=sort(Age)
> log.means=coeff[1]+coeff[2]*xvalues

## Un-log the values to get to the lambdas:
> mean.values=exp(log.means)
> lines(xvalues,mean.values)
```



The Poisson-regression model is a nonlinear model for the expected response.

7

8

What is the fitted Poisson model for an elephant of 30 years?

```
> lambda=exp(coeff[1]+coeff[2]*30)
> lambda
  1.613311
```

mean number of mates = 1.6

variance in number of mates = 1.6

What is the fitted Poisson model for an elephant of 45 years?

```
> lambda=exp(coeff[1]+coeff[2]*45)
> lambda
  4.514117
```

mean number of mates = 4.5

variance in number of mates = 4.5

We can test for significant effects as before.

```
> summary(glm.out)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.57955   0.54595 -2.893  0.00381 **
Age          0.06859   0.01378  4.979  6.4e-07 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 75.372 on 40 degrees of freedom
Residual deviance: 51.176 on 39 degrees of freedom
AIC: 156.62
```

Number of Fisher Scoring iterations: 5

Age is a significant predictor of the number of mates for an elephant.

Since the coefficient is positive, the expected number of mates increases with *Age*.

9

10

Parameter interpretation

{one covariate, $\ln(\lambda_i) = \beta_0 + \beta_1 X_i$ }

Interpretation of β_0 :

e^{β_0} is the mean of the Poisson distribution when $X = 0$.

Interpretation of β_1 :

Increasing X by 1 unit, has multiplicative effect on the mean of the Poisson by e^{β_1} ,

$$\frac{\lambda_{(x+1)}}{\lambda_{(x)}} = \frac{e^{\beta_0+\beta_1(x+1)}}{e^{\beta_0+\beta_1x}} = \frac{e^{\beta_0} e^{\beta_1x} e^{\beta_1}}{e^{\beta_0} e^{\beta_1x}} = e^{\beta_1}$$

$$\Rightarrow \lambda_{(x+1)} = \lambda_{(x)} e^{\beta_1}$$

- If $\beta_1 > 0$, then the expected count increases as X increases
- If $\beta_1 < 0$, then the expected count decreases as X increases.

For the elephant data,

Interpretation of β_0 :

Not meaningful in the context of the data as $Age=0$ is not meaningful, and is out of the range of the data.

Interpretation of β_1 :

An increase of 1 year in age increases the expected number of elephant mates by a multiplicative factor of $e^{0.06859} \approx 1.07$

11

12

- **Example:** Days absent at high school

School administrators study the attendance behavior of high school juniors at two schools.

VARIABLES:

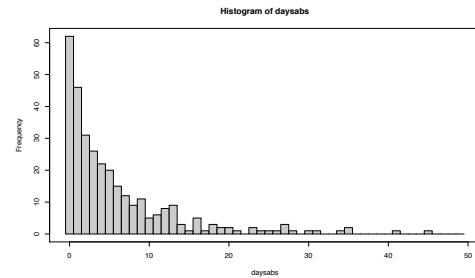
<i>daysabs</i>	days absent
<i>math</i>	standardized test scores
<i>langarts</i>	standardized test scores
<i>male</i>	gender (1=male, 0=female)

```
> attach(p)
> head(p)

  id school male      math langarts daysabs
1 1001     1   1 56.988830 42.45086     4
2 1002     1   1 37.094160 46.82059     4
3 1003     1   0 32.275460 43.56657     2
4 1004     1   0 29.056720 43.56657     3
5 1005     1   0  6.748048 27.24847     3
6 1006     1   0 61.654280 48.41482    13
```

13

A histogram of all counts:



A model with math, langarts, and male:

```
> glm.out.1=glm(daysabs~math+langarts+male,
                  family=poisson)
> summary(glm.out.1)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 2.687666  0.072651 36.994 < 2e-16 ***
math        -0.003523  0.001821 -1.934  0.0531 .
langarts    -0.012152  0.001835 -6.623 3.52e-11 ***
male        -0.400921  0.048412 -8.281 < 2e-16 ***
---
Signif. codes: 0 *** 0.001 ** 0.01 * 0.05 . 0.1 1
```

14

(Dispersion parameter for poisson family taken to be 1)

```
Null deviance: 2409.8 on 315 degrees of freedom
Residual deviance: 2234.5 on 312 degrees of freedom
AIC: 3103.9
```

Number of Fisher Scoring iterations: 6

```
## Performing the global null hypothesis:
> glm.out.null=glm(daysabs~1, family=poisson,data=p)
> chisq.st=glm.out.null$deviance-glm.out.1$deviance
> pchisq(chisq.st,3,lower.tail=FALSE)
[1] 9.245878e-38
```

Very significant. At least one of the predictors in the model helps explain days absent.

Since *math* was not significant, we will fit the simpler model:

```
> glm.out.2=glm(daysabs~langarts+male, family=poisson)
> summary(glm.out.2)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 2.646976  0.069776 37.935 < 2e-16 ***
langarts    -0.014670  0.001293 -11.342 < 2e-16 ***
male        -0.409353  0.048219 -8.489 < 2e-16 ***
```

For a student with an average language arts score, what affect does gender have?

```
> ave.lang=mean(langarts); ave.lang
[1] 50.06379

> lambda.female=exp(2.6470-0.0147*ave.lang + 0)
> lambda.female
[1] 6.760107

> lambda.male=exp(2.6470-0.0147*ave.lang + -0.4094)
> lambda.male
[1] 4.489039
```

For the average language arts student, a male is expected to miss ~ 2.3 fewer days.

15

16

As the coefficient on *langarts* is negative, there is a tendency for student with higher language arts scores to miss fewer days.

For a given sex...

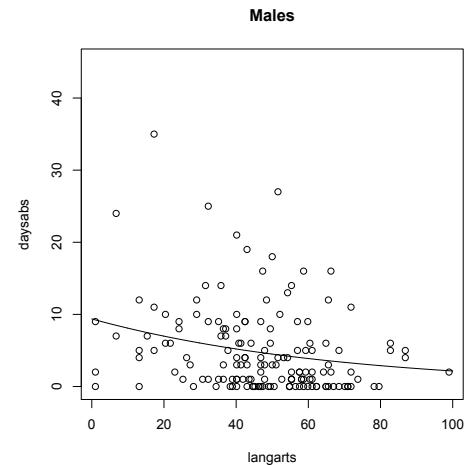
a 1-unit increase in *langarts* coincides with a decrease in the mean number of days missed by a multiplicative factor of $e^{-0.0147} = 0.9854$

a 10-unit increase in *langarts* coincides with a decrease in the mean number of days missed by a multiplicative factor of $e^{-0.0147*10} = 0.8633$

We can plot the fitted curve for each sex...

```
## Subset to only the male data points:
> male.data=p[male==1,]
> plot(male.data$daysabs~male.data$langarts,ylim=c(0,45)
      xlab="langarts",ylab="daysabs",main="Males")
```

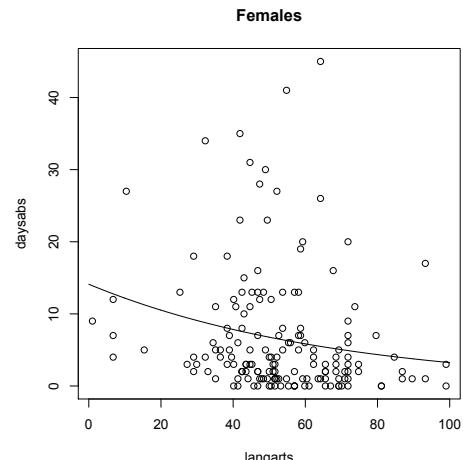
```
## Add the fitted curve:
> xvalues=seq(0,100,.1)
> mean.curve=exp(2.6469765-0.0147*xvalues-0.4094)
> lines(xvalues,mean.curve)
```



17

18

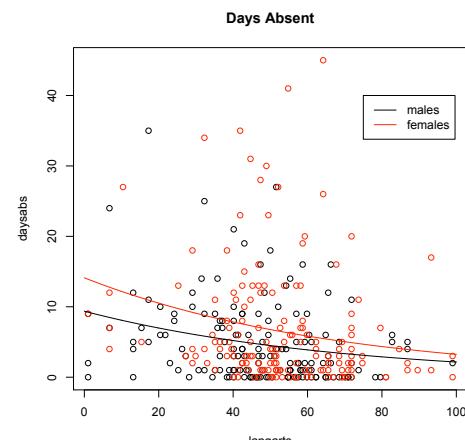
```
## Subset to only the female data points:
> female.data=p[male==0,]
> plot(female.data$daysabs~female.data$langarts,
       xlab="langarts",ylab="daysabs",main="Females")
## Add the fitted curve:
> xvalues=seq(0,100,.1)
> mean.curve=exp(2.6469765-0.0147*xvalues+0)
> lines(xvalues,mean.curve)
```



Females

Overlaid plots:

```
> plot(male.data$daysabs~male.data$langarts,
       xlab="langarts",ylab="daysabs",main="Days Absent",
       ylim=c(0,45))
> points(female.data$daysabs~female.data$langarts,col=2)
> mean.curve.m=exp(2.6469765-0.0147*xvalues-0.4094)
> lines(xvalues,mean.curve.m)
> mean.curve.f=exp(2.6469765-0.0147*xvalues+0)
> lines(xvalues,mean.curve.f,col=2)
> legend(75,40,c("males","females"),col=c(1,2),lty=c(1,1))
```



19

20

In some cases, the variance may not be quite equal to the mean, but they are still related.

There could be *overdispersion* (variance is greater than the mean) or *underdispersion* (variance is less than the mean). Shown as

$$V(Y_i|\mu_i) = \phi\mu_i$$

where ϕ is the *dispersion parameter*.

If $\phi > 1$, then the variance of Y increases more rapidly than the mean.

You can fit this model using a *quasi-likelihood*.

The estimated coefficients will actually be the same as in the Poisson model, but the standard errors (and tests) will more appropriately reflect the extra variability in the data.

21

Example: Days Absent at high school

The original fit of the data:

```
> glm.out.1=glm(daysabs~math+langarts+male,
                 family=poisson)
> summary(glm.out.1)
.
.
(Dispersion parameter for poisson family taken to be 1)

Null deviance: 2409.8 on 315 degrees of freedom
Residual deviance: 2234.5 on 312 degrees of freedom
AIC: 3103.9

Number of Fisher Scoring iterations: 6
```

If the mean and variance were equal, the residual deviance should be approximately equal to the *df* for error.

Resid. deviance is 2234.5 on 312 *df* for error.

Thus, it looks like we have *overdispersion*.

22

Re-fit the model allowing for overdispersion:

```
> glm.out.od=glm(daysabs~math+langarts+male,
                   family=quasipoisson)
> summary(glm.out.od)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 2.687666  0.216646 12.406 < 2e-16 ***
math        -0.003523  0.005431 -0.649  0.51701
langarts    -0.012152  0.005471 -2.221  0.02707 *
male        -0.400921  0.144365 -2.777  0.00582 **
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

(Dispersion parameter for quasipoisson family
 taken to be 8.892352)

Null deviance: 2409.8 on 315 degrees of freedom
Residual deviance: 2234.5 on 312 degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 6
```

The estimates are the same, but the standard errors are different (and the p-values are different).

Notice that the Dispersion parameter is 8.892352, and it was 1 before (i.e. mean equal to variance before).

References for the days absent data...

Long, J. S. 1997. Regression Models for Categorical and Limited Dependent Variables. Thousand Oaks, CA: Sage Publications.

Zeileis, A., Kleiber, C. and Jackman, S. Regression Models for Count Data in R.

Everitt, B. S. and Hothorn, T. A Handbook of Statistical Analyses Using R.

23

24

For the elephant data,

The original fit of the data:

```
> glm.out=glm(Matings~Age,family=poisson)
> summary(glm.out)

.
(Dispersion parameter for poisson family taken to be 1)

Null deviance: 75.372 on 40 degrees of freedom
Residual deviance: 51.176 on 39 degrees of freedom
AIC: 156.62

Number of Fisher Scoring iterations: 5
```

Resid. deviance is close to the df for error
(so, no overdispersion).

If we re-fit allowing for overdispersion (next page), it suggests that the original model with mean=variance fits reasonably well...

the standard errors don't change much,
the Dispersion parameter is 1.16

25

Model allowing for overdispersion:

```
> glm.out.od=glm(Matings~Age,family=quasipoisson)
> summary(glm.out.od)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.57955   0.58846 -2.684   0.0106 *
Age          0.06859   0.01485  4.619 4.13e-05 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

(Dispersion parameter for quasipoisson family
taken to be 1.161794)

Null deviance: 75.372 on 40 degrees of freedom
Residual deviance: 51.176 on 39 degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 5
```

26

• Example: Ear infection in swimmers

Count response: *Infections*
Number of ear infections

Categorical predictors:

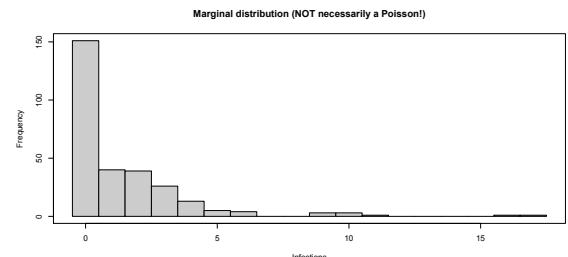
Swimmer: Frequent (0) or Occasional (1)
Location: Beach (0) or Non-beach (1)

```
> attach(ear.inf)
> head(ear.inf)
  Swimmer Location Infections
1    Occas NonBeach      0
2    Occas NonBeach      0
3    Occas NonBeach      0
4    Occas NonBeach      0
5    Occas NonBeach      0
6    Occas NonBeach      0

## Switch to sum-to-zero constraints (like 2-way ANOVA):
> contrasts(Swimmer)=contr.sum(levels(Swimmer))
> contrasts(Location)=contr.sum(levels(Location))

> hist(Infections,
       breaks=seq(0,18)-.5,col="grey80")
```

27



We will use the 'family=poisson' option:

```
> glm.out =glm(Infections~Swimmer + Location +
                  Swimmer:Location,family=poisson)
> summary(glm.out)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.25825   0.05355  4.822 1.42e-06 ***
Swimmer1    -0.29193   0.05355 -5.451 5.00e-08 ***
Location1   -0.23438   0.05355 -4.377 1.20e-05 ***
Swimmer1:Location1 0.06893   0.05355  1.287   0.198
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 824.51 on 286 degrees of freedom
Residual deviance: 763.00 on 283 degrees of freedom
AIC: 1143.4

Number of Fisher Scoring iterations: 6
```

28

The model (two-way ANOVA flavor):

$$\ln(\lambda_{ij}) = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij}$$

for $i = 1, 2$ and $j = 1, 2$

Overall null test:

$$H_0 : \alpha_i = \beta_j = (\alpha\beta)_{ij} = 0 \text{ for all } i, j$$

```
> chisq.stat=824.51-763.00
> pchisq(chisq.stat,3,lower.tail=FALSE)
[1] 2.796289e-13
```

We reject H_0 and conclude at least one of the terms in the model is significant.

From the previous summary output, we see that the interaction is not significant, I will refit the simpler model to continue.

```
> glm.out.2 =glm(Infections~Swimmer + Location,
  family=poisson)
> summary(glm.out.2)

Coefficients:
Estimate Std. Error z value Pr(>|z|)
(Intercept) 0.25505   0.05376  4.745 2.09e-06 ***
Swimmer1    -0.30652   0.05249 -5.839 5.24e-09 ***
Location1   -0.25437   0.05140 -4.948 7.49e-07 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

(Dispersion parameter for poisson family taken to be 1)
```

```
Null deviance: 824.51 on 286 degrees of freedom
Residual deviance: 764.65 on 284 degrees of freedom
AIC: 1143.0
```

Both factors are significant in the additive model.

Checking the way the factors are coded:

```
> contrasts(Swimmer)
 [,1]
Freq     1
Occas   -1
```

```
> contrasts(Location)
[,1]
Beach      1
NonBeach   -1
```

There are 4 distinct groups (or cells), each with its own expected number of ear infections.

```
> levels(Swimmer)
[1] "Freq"  "Occas"
> levels(Location)
[1] "Beach" "NonBeach"

> glm.out.2$coefficients
(Intercept)  Swimmer1  Location1
 0.2550506 -0.3065178 -0.2543653
```

The **Swimmer1** ('Freq') coefficient or $\hat{\alpha}_1$ is negative, so frequent swimmers tend to have fewer ear infections (and occasional swimmers tend to have more infections).

The **Location1** ('Beach') coefficient or $\hat{\beta}_1$ is negative, so beach swimmers tend to have fewer ear infections (and non-beach swimmers tend to have more infections).

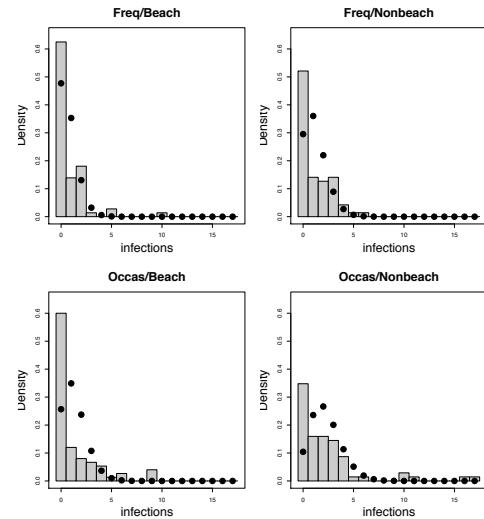
In other words, occasional non-beach swimmers tend to have the most ear infections.

(Note that this is an observational study and people with more ear infections may just choose to swim less).

Hand D.J., Daly F., Lunn A.D., McConway K.J., Ostrowski E. (1994). A Handbook of Small Data Sets. London: Chapman & Hall. Data set 328

Group	Poisson mean
Frequent Swimmers at the Beach	$\lambda = e^{(\hat{\mu} + \hat{\alpha}_1 + \hat{\beta}_1)} = 0.74$
Frequent Swimmers not at the Beach	$\lambda = e^{(\hat{\mu} + \hat{\alpha}_1 + \hat{\beta}_2)} = 1.22$
Occasional Swimmers at the beach	$\lambda = e^{(\hat{\mu} + \hat{\alpha}_2 + \hat{\beta}_1)} = 1.36$
Occasional Swimmers not at the beach	$\lambda = e^{(\hat{\mu} + \hat{\alpha}_2 + \hat{\beta}_2)} = 2.26$

When your predictors are categorical, you can look at the observed distribution of counts compared to the predicted distribution of counts based on the fitted Poisson distribution (or, similarly, the observed relative frequencies compared to the fitted probabilities) for each group separately...



In this case, though the predictors do seem to impact the distribution of ear infections, the Poisson may not be a good fit due to the large number of zeros.

In this type of situation...

33

34

... we can fit a zero-inflated Poisson (ZIP) regression.

This type of model accounts for the increased number of zeros.

MORE ON THIS IN THE NEXT SECTION OF NOTES.

35

Beta Regression in R

Francisco Cribari-Neto

Universidade Federal de Pernambuco

Achim Zeileis

Universität Innsbruck

Abstract

This introduction to the R package **betareg** is a (slightly) modified version of [Cribari-Neto and Zeileis \(2010\)](#), published in the *Journal of Statistical Software*. A follow-up paper with various extensions is [Grün, Kosmidis, and Zeileis \(2012\)](#) – a slightly modified version of which is also provided within the package as `vignette("betareg-ext", package = "betareg")`

The class of beta regression models is commonly used by practitioners to model variables that assume values in the standard unit interval $(0, 1)$. It is based on the assumption that the dependent variable is beta-distributed and that its mean is related to a set of regressors through a linear predictor with unknown coefficients and a link function. The model also includes a precision parameter which may be constant or depend on a (potentially different) set of regressors through a link function as well. This approach naturally incorporates features such as heteroskedasticity or skewness which are commonly observed in data taking values in the standard unit interval, such as rates or proportions. This paper describes the **betareg** package which provides the class of beta regressions in the R system for statistical computing. The underlying theory is briefly outlined, the implementation discussed and illustrated in various replication exercises.

Keywords: beta regression, rates, proportions, R.

1. Introduction

How should one perform a regression analysis in which the dependent variable (or response variable), y , assumes values in the standard unit interval $(0, 1)$? The usual practice used to be to transform the data so that the transformed response, say \tilde{y} , assumes values in the real line and then apply a standard linear regression analysis. A commonly used transformation is the logit, $\tilde{y} = \log(y/(1 - y))$. This approach, nonetheless, has shortcomings. First, the regression parameters are interpretable in terms of the mean of \tilde{y} , and not in terms of the mean of y (given Jensen's inequality). Second, regressions involving data from the unit interval such as rates and proportions are typically heteroskedastic: they display more variation around the mean and less variation as we approach the lower and upper limits of the standard unit interval. Finally, the distributions of rates and proportions are typically asymmetric, and thus Gaussian-based approximations for interval estimation and hypothesis testing can be quite inaccurate in small samples. [Ferrari and Cribari-Neto \(2004\)](#) proposed a regression model for continuous variates that assume values in the standard unit interval, e.g., rates, proportions, or concentration indices. Since the model is based on the assumption that the response is beta-distributed, they called their model *the beta regression model*. In their model, the regression parameters are interpretable in terms of the mean of y (the variable of interest)

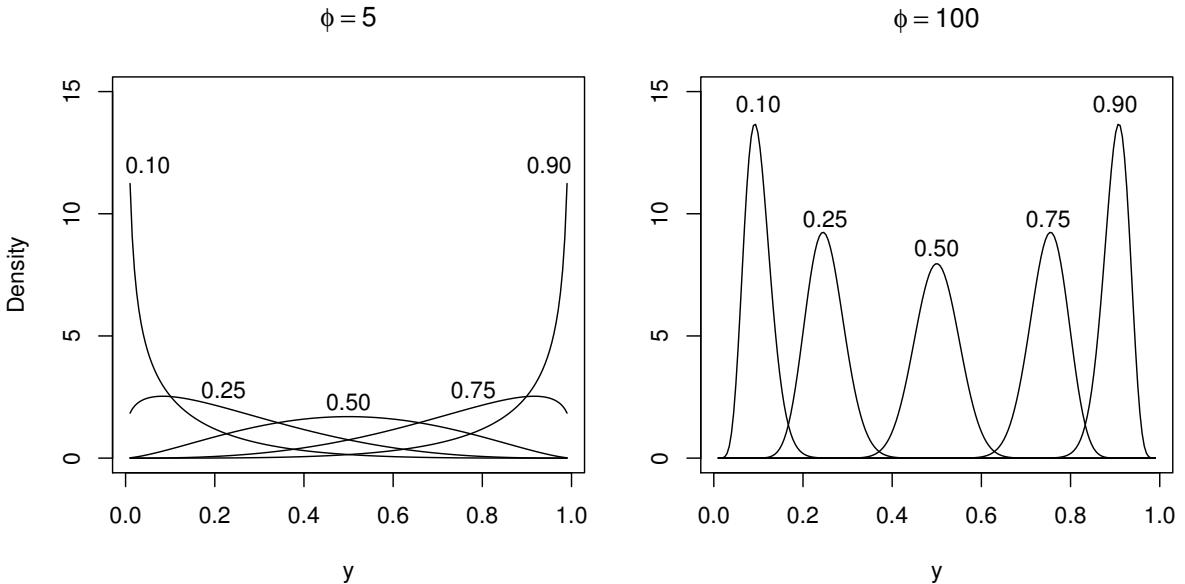


Figure 1: Probability density functions for beta distributions with varying parameters $\mu = 0.10, 0.25, 0.50, 0.75, 0.90$ and $\phi = 5$ (left) and $\phi = 100$ (right).

and the model is naturally heteroskedastic and easily accommodates asymmetries. A variant of the beta regression model that allows for nonlinearities and variable dispersion was proposed by [Simas, Barreto-Souza, and Rocha \(2010\)](#). In particular, in this more general model, the parameter accounting for the precision of the data is not assumed to be constant across observations but it is allowed to vary, leading to the *variable dispersion beta regression model*.

The chief motivation for the beta regression model lies in the flexibility delivered by the assumed beta law. The beta density can assume a number of different shapes depending on the combination of parameter values, including left- and right-skewed or the flat shape of the uniform density (which is a special case of the more general beta density). This is illustrated in Figure 1 which depicts several different beta densities. Following [Ferrari and Cribari-Neto \(2004\)](#), the densities are parameterized in terms of the mean μ and the precision parameter ϕ ; all details are explained in the next section. The evident flexibility makes the beta distribution an attractive candidate for data-driven statistical modeling.

The idea underlying beta regression models dates back to earlier approaches such as [Williams \(1982\)](#) or [Prentice \(1986\)](#). The initial motivation was to model binomial random variables with extra variation. The model postulated for the (discrete) variate of interest included a more flexible variation structure determined by independent beta-distributed variables which are related to a set of independent variables through a regression structure. However, unlike the more recent literature, the main focus was to model binomial random variables. Our interest in what follows will be more closely related to the recent literature, i.e., modeling continuous random variables that assume values in $(0, 1)$, such as rates, proportions, and concentration or inequality indices (e.g., Gini).

In this paper, we describe the **betareg** package which can be used to perform inference in both

fixed and variable dispersion beta regressions. The package is implemented in the R system for statistical computing ([R Development Core Team 2009](#)) and available from the Comprehensive R Archive Network (CRAN) at <http://CRAN.R-project.org/package=betareg>. The initial version of the package was written by [Simas and Rocha \(2006\)](#) up to version 1.2 which was orphaned and archived on CRAN in mid-2009. Starting from version 2.0-0, Achim Zeileis took over maintenance after rewriting/extending the package's functionality.

The paper unfolds as follows: Section 2 outlines the theory underlying the beta regression model before Section 3 describes its implementation in R. Sections 4 and 5 provide various empirical applications: The former focuses on illustrating various aspects of beta regressions in practice while the latter provides further replications of previously published empirical research. Finally, Section 6 contains concluding remarks and directions for future research and implementation.

2. Beta regression

The class of beta regression models, as introduced by [Ferrari and Cribari-Neto \(2004\)](#), is useful for modeling continuous variables y that assume values in the open standard unit interval $(0, 1)$. Note that if the variable takes on values in (a, b) (with $a < b$ known) one can model $(y - a)/(b - a)$. Furthermore, if y also assumes the extremes 0 and 1, a useful transformation in practice is $(y \cdot (n - 1) + 0.5)/n$ where n is the sample size ([Smithson and Verkuilen 2006](#)).

The beta regression model is based on an alternative parameterization of the beta density in terms of the variate mean and a precision parameter. The beta density is usually expressed as

$$f(y; p, q) = \frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q)} y^{p-1} (1-y)^{q-1}, \quad 0 < y < 1,$$

where $p, q > 0$ and $\Gamma(\cdot)$ is the gamma function.¹ [Ferrari and Cribari-Neto \(2004\)](#) proposed a different parameterization by setting $\mu = p/(p+q)$ and $\phi = p+q$:

$$f(y; \mu, \phi) = \frac{\Gamma(\phi)}{\Gamma(\mu\phi)\Gamma((1-\mu)\phi)} y^{\mu\phi-1} (1-y)^{(1-\mu)\phi-1}, \quad 0 < y < 1,$$

with $0 < \mu < 1$ and $\phi > 0$. We write $y \sim \mathcal{B}(\mu, \phi)$. Here, $\mathbb{E}(y) = \mu$ and $\text{VAR}(y) = \mu(1-\mu)/(1+\phi)$. The parameter ϕ is known as the precision parameter since, for fixed μ , the larger ϕ the smaller the variance of y ; ϕ^{-1} is a dispersion parameter.

Let y_1, \dots, y_n be a random sample such that $y_i \sim \mathcal{B}(\mu_i, \phi)$, $i = 1, \dots, n$. The beta regression model is defined as

$$g(\mu_i) = x_i^\top \beta = \eta_i,$$

where $\beta = (\beta_1, \dots, \beta_k)^\top$ is a $k \times 1$ vector of unknown regression parameters ($k < n$), $x_i = (x_{i1}, \dots, x_{ik})^\top$ is the vector of k regressors (or independent variables or covariates) and η_i is a linear predictor (i.e., $\eta_i = \beta_1 x_{i1} + \dots + \beta_k x_{ik}$; usually $x_{i1} = 1$ for all i so that the model has an intercept). Here, $g(\cdot) : (0, 1) \mapsto \mathbb{R}$ is a link function, which is strictly increasing and twice differentiable. The main motivation for using a link function in the regression structure

¹A beta regression model based on this parameterization was proposed by [Vasconcellos and Cribari-Neto \(2005\)](#). We shall, however, focus on the parameterization indexed by the mean and a precision parameter.

is twofold. First, both sides of the regression equation assume values in the real line when a link function is applied to μ_i . Second, there is an added flexibility since the practitioner can choose the function that yields the best fit. Some useful link functions are: logit $g(\mu) = \log(\mu/(1-\mu))$; probit $g(\mu) = \Phi^{-1}(\mu)$, where $\Phi(\cdot)$ is the standard normal distribution function; complementary log-log $g(\mu) = \log\{-\log(1-\mu)\}$; log-log $g(\mu) = -\log\{-\log(\mu)\}$; and Cauchy $g(\mu) = \tan\{\pi(\mu - 0.5)\}$. Note that the variance of y is a function of μ which renders the regression model based on this parameterization naturally heteroskedastic. In particular,

$$\text{VAR}(y_i) = \frac{\mu_i(1-\mu_i)}{1+\phi} = \frac{g^{-1}(x_i^\top \beta)[1-g^{-1}(x_i^\top \beta)]}{1+\phi}. \quad (1)$$

The log-likelihood function is $\ell(\beta, \phi) = \sum_{i=1}^n \ell_i(\mu_i, \phi)$, where

$$\begin{aligned} \ell_i(\mu_i, \phi) &= \log \Gamma(\phi) - \log \Gamma(\mu_i \phi) - \log \Gamma((1-\mu_i)\phi) + (\mu_i \phi - 1) \log y_i \\ &\quad + \{(1-\mu_i)\phi - 1\} \log(1-y_i). \end{aligned} \quad (2)$$

Notice that $\mu_i = g^{-1}(x_i^\top \beta)$ is a function of β , the vector of regression parameters. Parameter estimation is performed by maximum likelihood (ML).

An extension of the beta regression model above which was employed by [Smithson and Verkuilen \(2006\)](#) and formally introduced (along with further extensions) by [Simas et al. \(2010\)](#) is the variable dispersion beta regression model. In this model the precision parameter is not constant for all observations but instead modeled in a similar fashion as the mean parameter. More specifically, $y_i \sim \mathcal{B}(\mu_i, \phi_i)$ independently, $i = 1, \dots, n$, and

$$g_1(\mu_i) = \eta_{1i} = x_i^\top \beta, \quad (3)$$

$$g_2(\phi_i) = \eta_{2i} = z_i^\top \gamma, \quad (4)$$

where $\beta = (\beta_1, \dots, \beta_k)^\top$, $\gamma = (\gamma_1, \dots, \gamma_h)^\top$, $k+h < n$, are the sets of regression coefficients in the two equations, η_{1i} and η_{2i} are the linear predictors, and x_i and z_i are regressor vectors. As before, both coefficient vectors are estimated by ML, simply replacing ϕ by ϕ_i in Equation 2.

[Simas et al. \(2010\)](#) further extend the model above by allowing nonlinear predictors in Equations 3 and 4. Also, they have obtained analytical bias corrections for the ML estimators of the parameters, thus generalizing the results of [Ospina, Cribari-Neto, and Vasconcellos \(2006\)](#), who derived bias corrections for fixed dispersion beta regressions. However, as these extensions are not (yet) part of the **betareg** package, we confine ourselves to these short references and do not provide detailed formulas.

Various types of residuals are available for beta regression models. The raw response residuals $y_i - \hat{\mu}_i$ are typically not used due to the heteroskedasticity inherent in the model (see Equation 1). Hence, a natural alternative are *Pearson residuals* which [Ferrari and Cribari-Neto \(2004\)](#) call *standardized ordinary residuals* and define as

$$r_{P,i} = \frac{y_i - \hat{\mu}_i}{\sqrt{\widehat{\text{VAR}}(y_i)}}, \quad (5)$$

where $\widehat{\text{VAR}}(y_i) = \hat{\mu}_i(1-\hat{\mu}_i)/(1+\hat{\phi}_i)$, $\hat{\mu}_i = g_1^{-1}(x_i^\top \hat{\beta})$, and $\hat{\phi}_i = g_2^{-1}(z_i^\top \hat{\gamma})$. Similarly, deviance residuals can be defined in the standard way via signed contributions to the excess likelihood.

Further residuals were proposed by Espinheira, Ferrari, and Cribari-Neto (2008b), in particular one residual with better properties that they named *standardized weighted residual 2*:

$$r_{\text{sw2},i} = \frac{y_i^* - \hat{\mu}_i^*}{\sqrt{\hat{v}_i(1 - h_{ii})}}, \quad (6)$$

where $y_i^* = \log\{y_i/(1 - y_i)\}$ and $\mu_i^* = \psi(\mu_i\phi) - \psi((1 - \mu_i)\phi)$, $\psi(\cdot)$ denoting the digamma function. Standardization is then by $v_i = \{\psi'(\mu_i\phi) + \psi'((1 - \mu_i)\phi)\}$ and h_{ii} , the i th diagonal element of the hat matrix (for details see Ferrari and Cribari-Neto 2004; Espinheira *et al.* 2008b). Hats denote evaluation at the ML estimates.

It is noteworthy that the beta regression model described above was developed to allow practitioners to model continuous variates that assume values in the unit interval such as rates, proportions, and concentration or inequality indices (e.g., Gini). However, the data types that can be modeled using beta regressions also encompass proportions of “successes” from a number of trials, if the number of trials is large enough to justify a continuous model. In this case, beta regression is similar to a binomial generalized linear model (GLM) but provides some more flexibility – in particular when the trials are not independent and the standard binomial model might be too strict. In such a situation, the fixed dispersion beta regression is similar to the quasi-binomial model (McCullagh and Nelder 1989) but fully parametric. Furthermore, it can be naturally extended to variable dispersions.

3. Implementation in R

To turn the conceptual model from the previous section into computational tools in R, it helps to emphasize some properties of the model: It is a standard maximum likelihood (ML) task for which there is no closed-form solution but numerical optimization is required. Furthermore, the model shares some properties (such as linear predictor, link function, dispersion parameter) with generalized linear models (GLMs; McCullagh and Nelder 1989), but it is not a special case of this framework (not even for fixed dispersion). There are various models with implementations in R that have similar features – here, we specifically reuse some of the ideas employed for generalized count data regression by Zeileis, Kleiber, and Jackman (2008).

The main model-fitting function in **betareg** is **betareg()** which takes a fairly standard approach for implementing ML regression models in R: **formula** plus **data** is used for model and data specification, then the likelihood and corresponding gradient (or estimating function) is set up, **optim()** is called for maximizing the likelihood, and finally an object of S3 class “**betareg**” is returned for which a large set of methods to standard generics is available. The workhorse function is **betareg.fit()** which provides the core computations without **formula**-related data pre- and post-processing. **Update:** Recently, **betareg()** has been extended to optionally include an additional Fisher scoring iteration after the **optim()** optimization in order to improve the ML result (or apply a bias correction or reduction).

The model-fitting function **betareg()** and its associated class are designed to be as similar as possible to the standard **glm()** function (R Development Core Team 2009) for fitting GLMs. An important difference is that there are potentially two equations for mean and precision (Equations 3 and 4, respectively), and consequently two regressor matrices, two linear predictors, two sets of coefficients, etc. In this respect, the design of **betareg()** is similar to the functions described by Zeileis *et al.* (2008) for fitting zero-inflation and hurdle

models which also have two model components. The arguments of `betareg()` are

```
betareg(formula, data, subset, na.action, weights, offset,
  link = "logit", link.phi = NULL, control = betareg.control(...),
  model = TRUE, y = TRUE, x = FALSE, ...)
```

where the first line contains the standard model-frame specifications (see Chambers and Hastie 1992), the second line has the arguments specific to beta regression models and the arguments in the last line control some components of the return value.

If a `formula` of type $y \sim x_1 + x_2$ is supplied, it describes y_i and x_i for the mean equation of the beta regression (3). In this case a constant ϕ_i is assumed, i.e., $z_i = 1$ and g_2 is the identity link, corresponding to the basic beta regression model as introduced in Ferrari and Cribari-Neto (2004). However, a second set of regressors can be specified by a two-part formula of type $y \sim x_1 + x_2 | z_1 + z_2 + z_3$ as provided in the **Formula** package (Zeileis and Croissant 2010). This model has the same mean equation as above but the regressors z_i in the precision equation (4) are taken from the $| z_1 + z_2 + z_3$ part. The default link function in this case is the log link $g_2(\cdot) = \log(\cdot)$. Consequently, $y \sim x_1 + x_2$ and $y \sim x_1 + x_2 | 1$ correspond to equivalent beta likelihoods but use different parametrizations for ϕ_i : simply $\phi_i = \gamma_1$ in the former case and $\log(\phi_i) = \gamma_1$ in the latter case. The link for the ϕ_i precision equation can be changed by `link.phi` in both cases where "identity", "log", and "sqrt" are allowed as admissible values. The default for the μ_i mean equation is always the logit link but all link functions for the `binomial` family in `glm()` are allowed as well as the log-log link: "logit", "probit", "cloglog", "cauchit", "log", and "loglog".

ML estimation of all parameters employing analytical gradients is carried out using R's `optim()` with control options set in `betareg.control()`. All of `optim()`'s methods are available but the default is "BFGS", which is typically regarded to be the best-performing method (Mittelhammer, Judge, and Miller 2000, Section 8.13) with the most effective updating formula of all quasi-Newton methods (Nocedal and Wright 1999, p. 197). Starting values can be user-supplied, otherwise the β starting values are estimated by a regression of $g_1(y_i)$ on x_i . The starting values for the γ intercept are chosen as described in (Ferrari and Cribari-Neto 2004, p. 805), corresponding to a constant ϕ_i (plus a link transformation, if any). All further γ coefficients (if any) are initially set to zero. The covariance matrix estimate is derived analytically as in Simas *et al.* (2010). However, by setting `hessian = TRUE` the numerical Hessian matrix returned by `optim()` can also be obtained. **Update:** In recent versions of `betareg`, the `optim()` is still performed but optionally it may be complemented by a subsequent additional Fisher scoring iteration to improve the result.

The returned fitted-model object of class "betareg" is a list similar to "glm" objects. Some of its elements – such as `coefficients` or `terms` – are lists with a mean and precision component, respectively. A set of standard extractor functions for fitted model objects is available for objects of class "betareg", including the usual `summary()` method that includes partial Wald tests for all coefficients. No `anova()` method is provided, but the general `coeftest()` and `waldtest()` from **lmtest** (Zeileis and Hothorn 2002), and `linear.hypothesis()` from **car** (?) can be used for Wald tests while `lrtest()` from **lmtest** provides for likelihood-ratio tests of nested models. See Table 1 for a list of all available methods. Most of these are standard in base R, however, methods to a few less standard generics are also provided. Specifically, there are tools related to specification testing and computation of sandwich covariance matrices

Function	Description
<code>print()</code> <code>summary()</code>	simple printed display with coefficient estimates standard regression output (coefficient estimates, standard errors, partial Wald tests); returns an object of class “ <code>summary.betareg</code> ” containing the relevant summary statistics (which has a <code>print()</code> method)
<code>coef()</code> <code>vcov()</code> <code>predict()</code> <code>fitted()</code> <code>residuals()</code> <code>estfun()</code>	extract coefficients of model (full, mean, or precision components), a single vector of all coefficients by default associated covariance matrix (with matching names) predictions (of means μ_i , linear predictors η_{1i} , precision parameter ϕ_i , or variances $\mu_i(1 - \mu_i)/(1 + \phi_i)$) for new data fitted means for observed data
 <code>terms()</code> <code>model.matrix()</code> <code>model.frame()</code> <code>logLik()</code>	extract residuals (deviance, Pearson, response, or different weighted residuals, see Espinheira et al. 2008b), defaulting to standardized weighted residuals 2 from Equation 6 compute empirical estimating functions (or score functions), evaluated at observed data and estimated parameters (see Zeileis 2006b)
 <code>bread()</code>	extract “bread” matrix for sandwich estimators (see Zeileis 2006b)
 <code>plot()</code> <code>hatvalues()</code> <code>cooks.distance()</code> <code>gleverage()</code>	diagnostic plots of residuals, predictions, leverages etc. hat values (diagonal of hat matrix) (approximation of) Cook’s distance compute generalized leverage (Wei, Hu, and Fung 1998 ; Rocha and Simas 2010)
 <code>coeftest()</code> <code>waldtest()</code> <code>linear.hypothesis()</code> <code>lrtest()</code> <code>AIC()</code>	partial Wald tests of coefficients Wald tests of nested models Wald tests of linear hypotheses likelihood ratio tests of nested models compute information criteria (AIC, BIC, ...)

Table 1: Functions and methods for objects of class “`betareg`”. The first four blocks refer to methods, the last block contains generic functions whose default methods work because of the information supplied by the methods above.

as discussed by [Zeileis \(2004, 2006b\)](#) as well as a method to a new generic for computing generalized leverages ([Wei et al. 1998](#)).

4. Beta regression in practice

To illustrate the usage of `betareg` in practice we replicate and slightly extend some of the

analyses from the original papers that suggested the methodology. More specifically, we estimate and compare various flavors of beta regression models for the gasoline yield data of Prater (1956), see Figure 2, and for the household food expenditure data taken from Griffiths, Hill, and Judge (1993), see Figure 4. Further pure replication exercises are provided in Section 5.

4.1. The basic model: Estimation, inference, diagnostics

Prater's gasoline yield data

The basic beta regression model as suggested by Ferrari and Cribari-Neto (2004) is illustrated in Section 4 of their paper using two empirical examples. The first example employs the well-known gasoline yield data taken from Prater (1956). The variable of interest is `yield`, the proportion of crude oil converted to gasoline after distillation and fractionation, for which a beta regression model is rather natural. Ferrari and Cribari-Neto (2004) employ two explanatory variables: `temp`, the temperature (in degrees Fahrenheit) at which all gasoline has vaporized, and `batch`, a factor indicating ten unique batches of conditions in the experiments (depending on further variables). The data, encompassing 32 observations, is visualized in Figure 2.

Ferrari and Cribari-Neto (2004) start out with a model where `yield` depends on `batch` and `temp`, employing the standard logit link. In `betareg`, this can be fitted via

```
R> data("GasolineYield", package = "betareg")
R> gy_logit <- betareg(yield ~ batch + temp, data = GasolineYield)
R> summary(gy_logit)

Call:
betareg(formula = yield ~ batch + temp, data = GasolineYield)

Standardized weighted residuals 2:
    Min      1Q   Median      3Q      Max 
-2.8750 -0.8149  0.1601  0.8384  2.0483 

Coefficients (mean model with logit link):
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) -6.1595710  0.1823247 -33.784 < 2e-16 ***
batch1       1.7277289  0.1012294  17.067 < 2e-16 ***
batch2       1.3225969  0.1179020  11.218 < 2e-16 ***
batch3       1.5723099  0.1161045  13.542 < 2e-16 ***
batch4       1.0597141  0.1023598  10.353 < 2e-16 ***
batch5       1.1337518  0.1035232  10.952 < 2e-16 ***
batch6       1.0401618  0.1060365   9.809 < 2e-16 ***
batch7       0.5436922  0.1091275   4.982 6.29e-07 ***
batch8       0.4959007  0.1089257   4.553 5.30e-06 ***
batch9       0.3857930  0.1185933   3.253  0.00114 ** 
temp         0.0109669  0.0004126  26.577 < 2e-16 ***
```

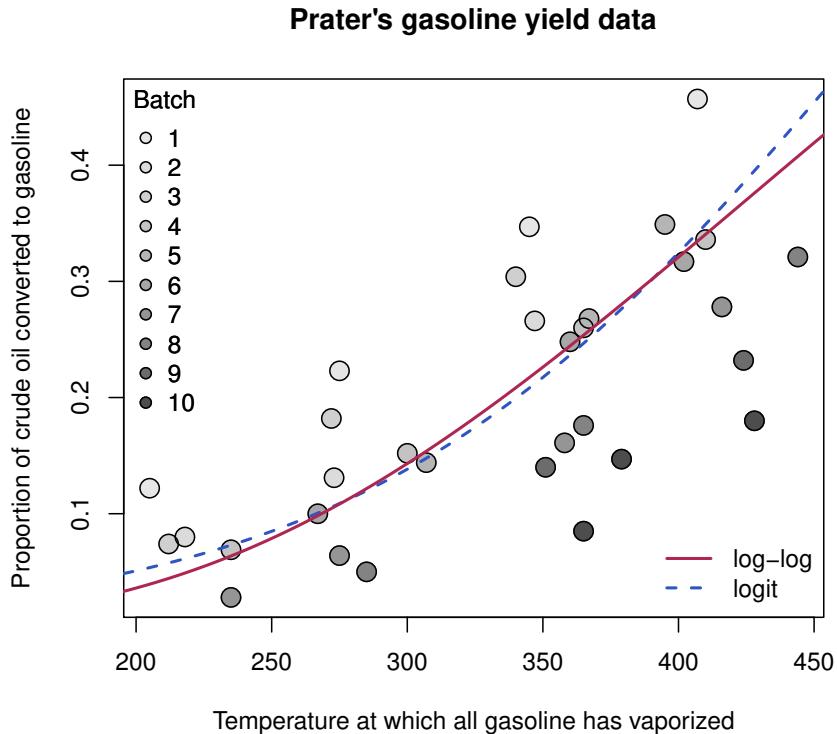


Figure 2: Gasoline yield data from Prater (1956): Proportion of crude oil converted to gasoline explained by temperature (in degrees Fahrenheit) at which all gasoline has vaporized and given batch (indicated by gray level). Fitted curves correspond to beta regressions `gy_loglog` with log-log link (solid, red) and `gy_logit` with logit link (dashed, blue). Both curves were evaluated at varying temperature with the intercept for batch 6 (i.e., roughly the average intercept).

```

Phi coefficients (precision model with identity link):
  Estimate Std. Error z value Pr(>|z|)
(phi)    440.3     110.0   4.002 6.29e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Type of estimator: ML (maximum likelihood)
Log-likelihood:  84.8 on 12 Df
Pseudo R-squared: 0.9617
Number of iterations: 51 (BFGS) + 3 (Fisher scoring)

```

which replicates their Table 1. The goodness of fit is assessed using different types of diagnostic displays shown in their Figure 2. This graphic can be reproduced (in a slightly different order) using the `plot()` method for “`betareg`” objects, see Figure 3.

```
R> set.seed(123)
```

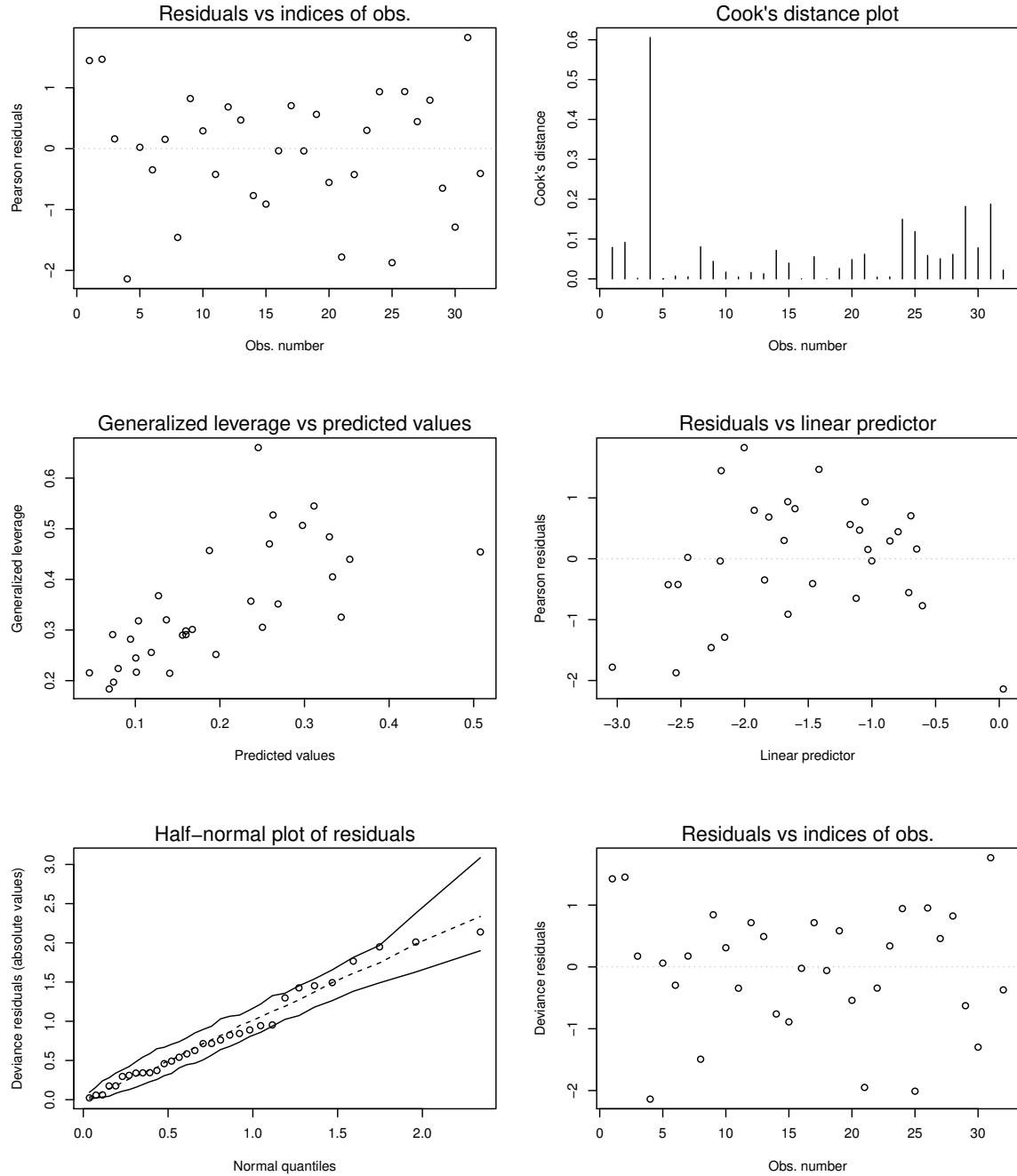


Figure 3: Diagnostic plots for beta regression model `gy_logit`.

```
R> plot(gy_logit, which = 1:4, type = "pearson")
R> plot(gy_logit, which = 5, type = "deviance", sub.caption = "")
R> plot(gy_logit, which = 1, type = "deviance", sub.caption = "")
```

As observation 4 corresponds to a large Cook's distance and large residual, [Ferrari and Cribari-](#)

Neto (2004) decided to refit the model excluding this observation. While this does not change the coefficients in the mean model very much, the precision parameter ϕ increases clearly.

```
R> gy_logit4 <- update(gy_logit, subset = -4)
R> coef(gy_logit, model = "precision")

(phi)
440.2784

R> coef(gy_logit4, model = "precision")

(phi)
577.7907
```

Household food expenditures

Ferrari and Cribari-Neto (2004) also consider a second example: household food expenditure data for 38 households taken from Griffiths *et al.* (1993, Table 15.4). The dependent variable is **food/income**, the proportion of household **income** spent on **food**. Two explanatory variables are available: the previously mentioned household **income** and the number of **persons** living in the household. All three variables are visualized in Figure 4.

To start their analysis, Ferrari and Cribari-Neto (2004) consider a simple linear regression model fitted by ordinary least squares (OLS):

```
R> data("FoodExpenditure", package = "betareg")
R> fe_lm <- lm(I(food/income) ~ income + persons, data = FoodExpenditure)
```

To show that this model exhibits heteroskedasticity, they employ the studentized Breusch and Pagan (1979) test of Koenker (1981) which is available in R in the **lmtest** package (Zeileis and Hothorn 2002).

```
R> library("lmtest")
R> bptest(fe_lm)

studentized Breusch-Pagan test

data: fe_lm
BP = 5.9348, df = 2, p-value = 0.05144
```

One alternative would be to consider a logit-transformed response in a traditional OLS regression but this would make the residuals asymmetric. However, both issues – heteroskedasticity and skewness – can be alleviated when a beta regression model with a logit link for the mean is used.

```
R> fe_beta <- betareg(I(food/income) ~ income + persons,
+   data = FoodExpenditure)
R> summary(fe_beta)
```

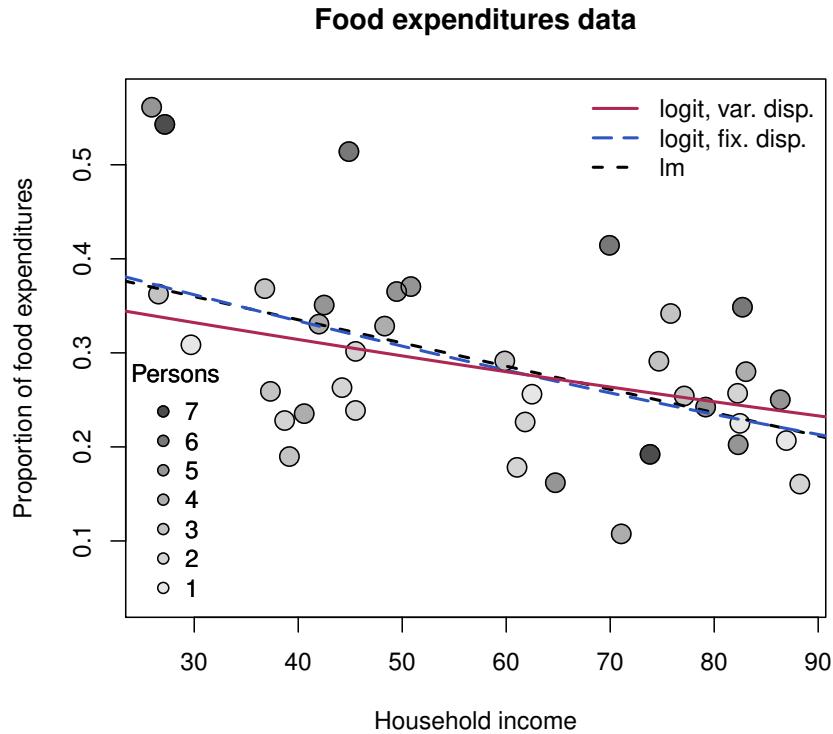


Figure 4: Household food expenditure data from Griffiths *et al.* (1993): Proportion of household income spent on food explained by household income and number of persons in household (indicated by gray level). Fitted curves correspond to beta regressions `fe_beta` with fixed dispersion (long-dashed, blue), `fe_beta2` with variable dispersion (solid, red), and the linear regression `fe_lin` (dashed, black). All curves were evaluated at varying income with the intercept for mean number of persons (= 3.58).

Call:

```
betareg(formula = I(food/income) ~ income + persons, data = FoodExpenditure)
```

Standardized weighted residuals 2:

Min	1Q	Median	3Q	Max
-2.7818	-0.4445	0.2024	0.6852	1.8755

Coefficients (mean model with logit link):

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	-0.622548	0.223854	-2.781	0.005418 **
income	-0.012299	0.003036	-4.052	5.09e-05 ***
persons	0.118462	0.035341	3.352	0.000802 ***

Phi coefficients (precision model with identity link):

Estimate	Std. Error	z value	Pr(> z)
----------	------------	---------	----------

```
(phi)    35.61      8.08   4.407 1.05e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Type of estimator: ML (maximum likelihood)
Log-likelihood: 45.33 on 4 Df
Pseudo R-squared: 0.3878
Number of iterations: 28 (BFGS) + 4 (Fisher scoring)
```

This replicates Table 2 from [Ferrari and Cribari-Neto \(2004\)](#). The predicted means of the linear and the beta regression model, respectively, are very similar: the proportion of household income spent on food decreases with the overall income level but increases in the number of persons in the household (see also Figure 4).

Below, further extended models will be considered for these data sets and hence all model comparisons are deferred.

4.2. Variable dispersion model

Prater's gasoline yield data

Although the beta model already incorporates naturally a certain pattern in the variances of the response (see Equation 1), it might be necessary to incorporate further regressors to account for heteroskedasticity as in Equation 4 ([Simas et al. 2010](#)). For illustration of this approach, the example from Section 3 of the online supplements to [Simas et al. \(2010\)](#) is considered. This investigates Prater's gasoline yield data based on the same mean equation as above, but now with temperature `temp` as an additional regressor for the precision parameter ϕ_i :

```
R> gy_logit2 <- betareg(yield ~ batch + temp | temp, data = GasolineYield)
```

for which `summary(gy_logit2)` yields the MLE column in Table 19 of [Simas et al. \(2010\)](#). To save space, only the parameters pertaining to ϕ_i are reported here

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	1.3640888	1.2257812	1.1128	0.2658
temp	0.0145703	0.0036183	4.0269	5.653e-05 ***

Signif. codes:	0 '***'	0.001 '**'	0.01 '*'	0.05 '.'
	0.1 ' '	1		

which signal a significant improvement by including the `temp` regressor. Instead of using this Wald test, the models can also be compared by means of a likelihood-ratio test (see their Table 18) that confirms the results:

```
R> lrtest(gy_logit, gy_logit2)
```

```
Likelihood ratio test
```

```

Model 1: yield ~ batch + temp
Model 2: yield ~ batch + temp | temp
  #Df LogLik Df Chisq Pr(>Chisq)
1 12 84.798
2 13 86.977  1 4.359    0.03681 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Note that this can also be interpreted as testing the null hypothesis of equidispersion against a specific alternative of variable dispersion.

Household food expenditures

For the household food expenditure data, the Breusch-Pagan test carried out above illustrated that there is heteroskedasticity that can be captured by the regressors `income` and `persons`. Closer investigation reveals that this is mostly due to the number of persons in the household, also brought out graphically by some of the outliers with high values in this variable in Figure 4. Hence, it seems natural to consider the model employed above with `persons` as an additional regressor in the precision equation.

```
R> fe_beta2 <- betareg(I(food/income) ~ income + persons | persons,
+   data = FoodExpenditure)
```

This leads to significant improvements in terms of the likelihood and the associated BIC.²

```
R> lrtest(fe_beta, fe_beta2)
```

Likelihood ratio test

```

Model 1: I(food/income) ~ income + persons
Model 2: I(food/income) ~ income + persons | persons
  #Df LogLik Df Chisq Pr(>Chisq)
1  4 45.334
2  5 49.185  1 7.7029   0.005513 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

```
R> AIC(fe_beta, fe_beta2, k = log(nrow(FoodExpenditure)))
```

	df	AIC
fe_beta	4	-76.11667
fe_beta2	5	-80.18198

Thus, there is evidence for variable dispersion and model `fe_beta2` seems to be preferable. As visualized in Figure 4, it describes a similar relationship between response and explanatory variables although with a somewhat shrunken `income` slope.

²In R, the BIC can be computed by means of `AIC()` when `log(n)` is supplied as the penalty term `k`

4.3. Selection of different link functions

Prater's gasoline yield data

As in binomial GLMs, selection of an appropriate link function can greatly improve the model fit (McCullagh and Nelder 1989), especially if extreme proportions (close to 0 or 1) have been observed in the data. To illustrate this problem in beta regressions, we replicate parts of the analysis in Section 5 of Cribari-Neto and Lima (2007). This reconsiders Prater's gasoline yield data but employs a log-log link instead of the previously used (default) logit link

```
R> gy_loglog <- betareg(yield ~ batch + temp, data = GasolineYield,
+   link = "loglog")
```

which clearly improves the pseudo R^2 of the model:

```
R> summary(gy_logit)$pseudo.r.squared
```

```
[1] 0.9617312
```

```
R> summary(gy_loglog)$pseudo.r.squared
```

```
[1] 0.9852334
```

Similarly, the AIC³ (and BIC) of the fitted model is not only superior to the logit model with fixed dispersion `gy_logit` but also to the logit model with variable dispersion `gy_logit2` considered in the previous section.

```
R> AIC(gy_logit, gy_logit2, gy_loglog)
```

df	AIC
gy_logit	12 -145.5951
gy_logit2	13 -147.9541
gy_loglog	12 -168.3101

Moreover, if `temp` were included as a regressor in the precision equation of `gy_loglog`, it would no longer yield significant improvements. Thus, improvement of the model fit in the mean equation by adoption of the log-log link has waived the need for a variable precision equation.

To underline the appropriateness of the log-log specification, Cribari-Neto and Lima (2007) consider a sequence of diagnostic tests inspired by the RESET (regression specification error test; Ramsey 1969) in linear regression models. To check for misspecifications, they consider powers of fitted means or linear predictors to be included as auxiliary regressors in the mean equation. In well-specified models, these should not yield significant improvements. For the gasoline yield model, this can only be obtained for the log-log link while all other link functions result in significant results indicating misspecification. Below, this is exemplified for a likelihood-ratio test of squared linear predictors. Analogous results can be obtained for `type = "response"` or higher powers.

³Note that Cribari-Neto and Lima (2007) did not account for estimation of ϕ in their degrees of freedom. Hence, their reported AICs differ by 2.

```
R> lrtest(gy_logit, . ~ . + I(predict(gy_logit, type = "link")^2))

Likelihood ratio test

Model 1: yield ~ batch + temp
Model 2: yield ~ batch + temp + I(predict(gy_logit, type = "link")^2)
#Df LogLik Df Chisq Pr(>Chisq)
1 12 84.798
2 13 96.001 1 22.407 2.205e-06 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R> lrtest(gy_loglog, . ~ . + I(predict(gy_loglog, type = "link")^2))

Likelihood ratio test

Model 1: yield ~ batch + temp
Model 2: yield ~ batch + temp + I(predict(gy_loglog, type = "link")^2)
#Df LogLik Df Chisq Pr(>Chisq)
1 12 96.155
2 13 96.989 1 1.6671 0.1966
```

The improvement of the model fit can also be brought out graphically by comparing absolute raw residuals (i.e., $y_i - \hat{\mu}_i$) from both models as in Figure 5.

```
R> plot(abs(residuals(gy_loglog, type = "response")),
+      abs(residuals(gy_logit, type = "response")))
R> abline(0, 1, lty = 2)
```

This shows that there are a few observations clearly above the diagonal (where the log-log-link fits better than the logit link) whereas there are fewer such observations below the diagonal. A different diagnostic display that is useful in this situation (and is employed by [Cribari-Neto and Lima 2007](#)) is a plot of predicted values ($\hat{\mu}_i$) vs. observed values (y_i) for each model. This can be created by `plot(gy_logit, which = 6)` and `plot(gy_loglog, which = 6)`, respectively.

In principle, the link function g_2 in the precision equation could also influence the model fit. However, as the best-fitting model `gy_loglog` has a constant ϕ , all links g_2 lead to equivalent estimates of ϕ and thus to equivalent fitted log-likelihoods. However, the link function can have consequences in terms of the inference about ϕ and in terms of convergence of the optimization. Typically, a log-link leads to somewhat improved quadratic approximations of the likelihood and less iterations in the optimization. For example, refitting `gy_loglog` with $g_2(\cdot) = \log(\cdot)$ converges more quickly:

```
R> gy_loglog2 <- update(gy_loglog, link.phi = "log")
R> summary(gy_loglog2)$iterations

optim scoring
      21        2
```

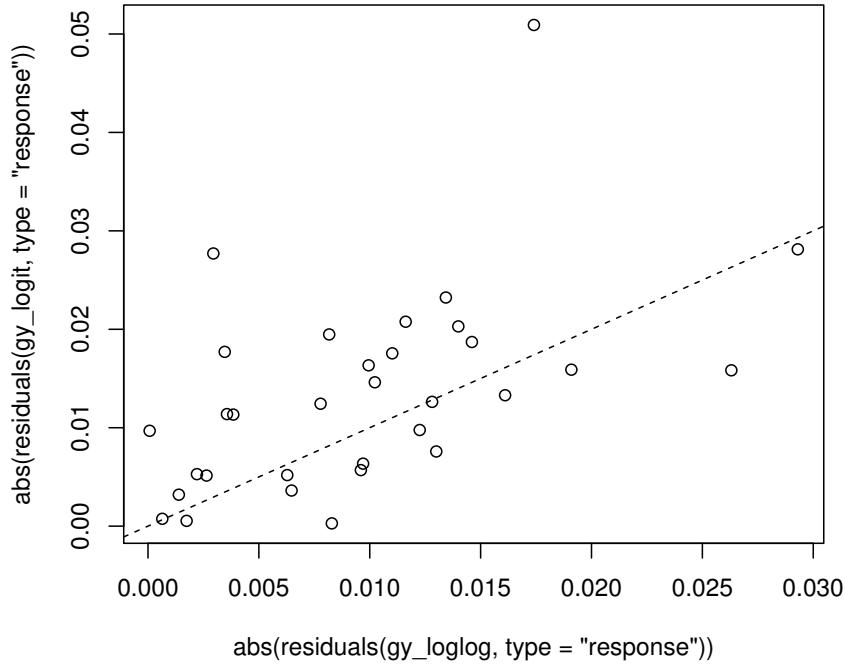


Figure 5: Scatterplot comparing the absolute raw residuals from beta regression modes with log-log link (x-axis) and logit link (y-axis).

with a lower number of iterations than for `gy_loglog` which had 51 iterations.

Household food expenditures

One could conduct a similar analysis as above for the household food expenditure data. However, as the response takes less extreme observations than for the gasoline yield data, the choice of link function is less important. In fact, refitting the model with various link functions shows no large differences in the resulting log-likelihoods. These can be easily extracted from fitted models using the `logLik()` function, e.g., `logLik(fe_beta2)`. Below we use a compact `sapply()` call to obtain this for updated versions of `fe_beta2` with all available link functions.

```
R> sapply(c("logit", "probit", "cloglog", "cauchit", "loglog"),
+   function(x) logLik(update(fe_beta2, link = x)))
logit  probit  cloglog  cauchit  loglog
49.18495 49.08044 49.35888 50.01105 48.86718
```

Only the Cauchy link performs somewhat better than the logit link and might hence deserve further investigation.

5. Further replication exercises

In this section, further empirical illustrations of beta regressions are provided. While the emphasis in the previous section was to present how the various features of **betareg** can be used in practice, we focus more narrowly on replication of previously published research articles below.

5.1. Dyslexia and IQ predicting reading accuracy

We consider an application that analyzes reading accuracy data for nondyslexic and dyslexic Australian children ([Smithson and Verkuilen 2006](#)). The variable of interest is **accuracy** providing the scores on a test of reading accuracy taken by 44 children, which is predicted by the two regressors **dyslexia** (a factor with sum contrasts separating a dyslexic and a control group) and nonverbal intelligent quotient (**iq**, converted to z scores), see Figure 6 for a visualization. The sample includes 19 dyslexics and 25 controls who were recruited from primary schools in the Australian Capital Territory. The children's ages ranged from eight years five months to twelve years three months; mean reading accuracy was 0.606 for dyslexic readers and 0.900 for controls.

[Smithson and Verkuilen \(2006\)](#) set out to investigate whether **dyslexia** contributes to the explanation of **accuracy** even when corrected for **iq** score (which is on average lower for dyslexics). Hence, they consider separate regressions for the two groups fitted by the interaction of both regressors. To show that OLS regression is no suitable tool in this situation, they first fit a linear regression of the logit-transformed response:

```
R> data("ReadingSkills", package = "betareg")
R> rs_ols <- lm(qlogis(accuracy) ~ dyslexia * iq, data = ReadingSkills)
R> coeftest(rs_ols)

t test of coefficients:

            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1.60107   0.22586  7.0888 1.411e-08 ***
dyslexia    -1.20563   0.22586 -5.3380 4.011e-06 ***
iq           0.35945   0.22548  1.5941  0.11878    
dyslexia:iq -0.42286   0.22548 -1.8754  0.06805 .  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The interaction effect does not appear to be significant, however this is a result of the poor fit of the linear regression as will be shown below. Figure 6 clearly shows that the data are asymmetric and heteroskedastic (especially in the control group). Hence, [Smithson and Verkuilen \(2006\)](#) fit a beta regression model, again with separate means for both groups, but they also allow the dispersion to depend on the main effects of both variables.

```
R> rs_beta <- betareg(accuracy ~ dyslexia * iq / dyslexia + iq,
+   data = ReadingSkills, hessian = TRUE)
R> coeftest(rs_beta)
```

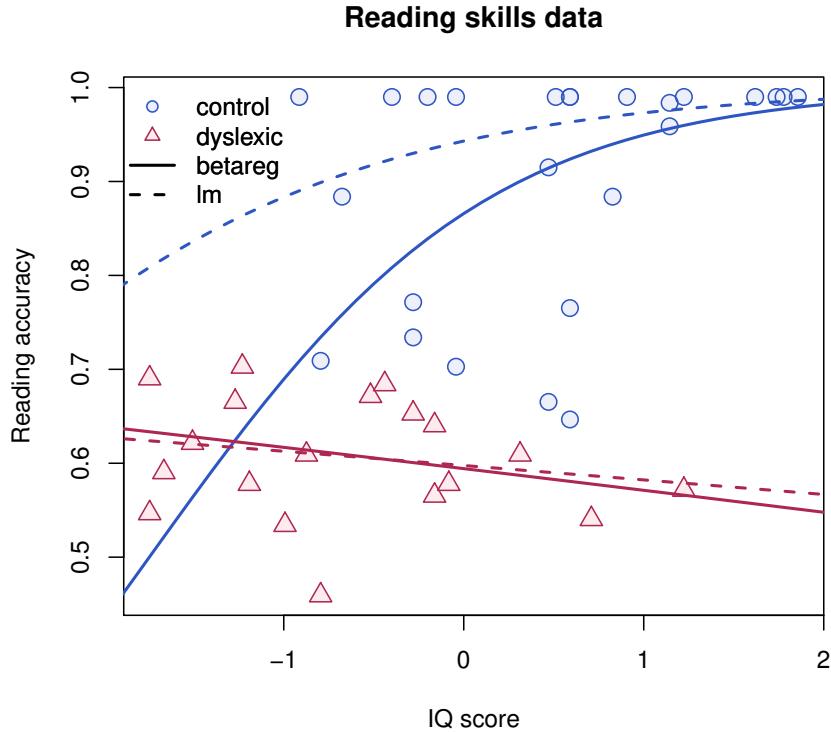


Figure 6: Reading skills data from Smithson and Verkuilen (2006): Linearly transformed reading accuracy by IQ score and dyslexia status (control, blue vs. dyslexic, red). Fitted curves correspond to beta regression `rs_beta` (solid) and OLS regression with logit-transformed dependent variable `rs_ols` (dashed).

```
z test of coefficients:
```

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	1.12323	0.15089	7.4441	9.758e-14 ***
dyslexia	-0.74165	0.15145	-4.8969	9.736e-07 ***
iq	0.48637	0.16708	2.9109	0.0036034 **
dyslexia:iq	-0.58126	0.17258	-3.3681	0.0007568 ***
(phi)_-(Intercept)	3.30443	0.22650	14.5890	< 2.2e-16 ***
(phi)_-dyslexia	1.74656	0.29398	5.9410	2.832e-09 ***
(phi)_-iq	1.22907	0.45957	2.6744	0.0074862 **
<hr/>				
Signif. codes:	0	'***'	0.001	'**'
	0.01	'*'	0.05	'. '
	0.1	' '	1	

This shows that precision increases with `iq` and is lower for controls while in the mean equation there is a significant interaction between `iq` and `dyslexia`. As Figure 6 illustrates, the beta regression fit does not differ much from the OLS fit for the dyslexics group (with responses close to 0.5) but fits much better in the control group (with responses close to 1).

The estimates above replicate those in Table 5 of [Smithson and Verkuilen \(2006\)](#), except for the signs of the coefficients of the dispersion submodel which they defined in the opposite way. Note that their results have been obtained with numeric rather than analytic standard errors hence `hessian = TRUE` is set above for replication. The results are also confirmed by [Espinheira, Ferrari, and Cribari-Neto \(2008a\)](#), who have also concluded that the dispersion is variable. As pointed out in Section 4.2, to formally test equidispersion against variable dispersion `lrtest(rs_beta, . ~ . | 1)` (or the analogous `waldtest()`) can be used.

[Smithson and Verkuilen \(2006\)](#) also consider two other psychometric applications of beta regressions the data for which are also provided in the `betareg` package: see `?MockJurors` and `?StressAnxiety`. Furthermore, `demo("SmithsonVerkuilen2006", package = "betareg")` is a complete replication script with comments.

5.2. Structural change testing in beta regressions

As already illustrated in Section 4, “`betareg`” objects can be plugged into various inference functions from other packages because they provide suitable methods to standard generic functions (see Table 1). Hence `lrtest()` could be used for performing likelihood-ratio testing inference and similarly `coeftest()`, `waldtest()` from `lmtest` ([Zeileis and Hothorn 2002](#)) and `linear.hypothesis()` from `car` (?) can be employed for carrying out different flavors of Wald tests.

In this section, we illustrate yet another generic inference approach implemented in the `strucchange` package for structural change testing. This is concerned with a different testing problem compared to the functions above: It assesses whether the model parameters are stable throughout the entire sample or whether they change over the observations $i = 1, \dots, n$. This is of particular interest in time series applications where the regression coefficients β and γ change at some unknown time in the sample period (see [Zeileis 2006a](#), for more details and references to the literature).

While originally written for linear regression models ([Zeileis, Leisch, Hornik, and Kleiber 2002](#)), `strucchange` was extended by [Zeileis \(2006a\)](#) to compute generalized fluctuation tests for structural change in models that are based on suitable estimating functions. The idea is to capture systematic deviations from parameter stability by cumulative sums of the empirical estimating functions: If the parameters are stable, the cumulative sum process should fluctuate randomly around zero. However, if there is an abrupt shift in the parameters, the cumulative sums will deviate clearly from zero and have a peak at around the time of the shift. If the estimating functions can be extracted by an `estfun()` method (as for “`betareg`” objects), models can simply be plugged into the `gefpl()` function for computing these cumulative sums (also known as generalized empirical fluctuation processes). To illustrate this, we replicate the example from Section 5.3 in [Zeileis \(2006a\)](#).

Two artificial data sets are considered: a series y_1 with a change in the mean μ , and a series y_2 with a change in the precision ϕ . Both simulated series start with the parameters $\mu = 0.3$ and $\phi = 4$ and for the first series μ changes to 0.5 after 75% of the observations while ϕ remains constant whereas for the second series ϕ changes to 8 after 50% of the observations and μ remains constant.

```
R> set.seed(123)
R> y1 <- c(rbeta(150, 0.3 * 4, 0.7 * 4), rbeta(50, 0.5 * 4, 0.5 * 4))
```

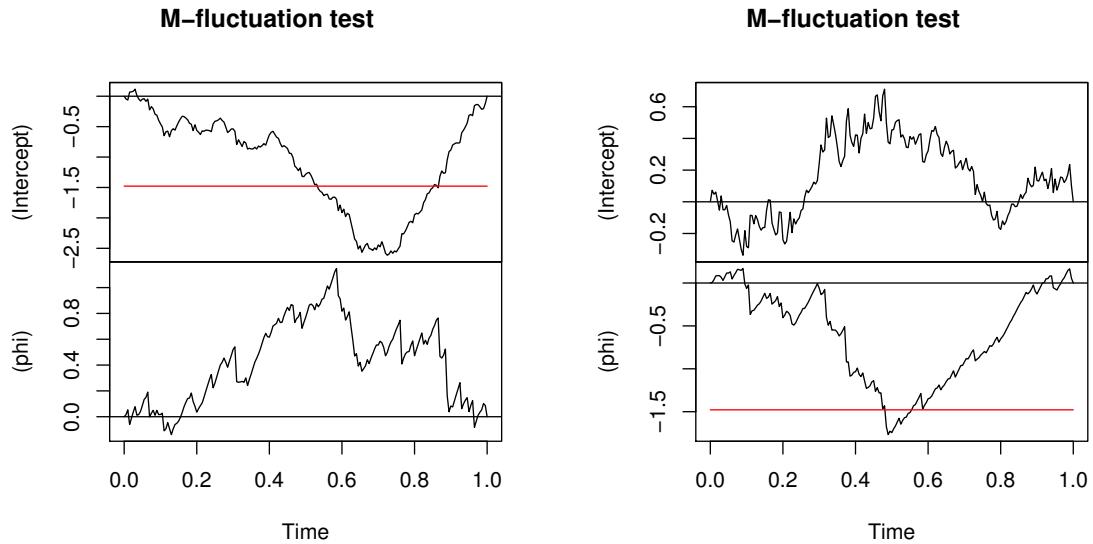


Figure 7: Structural change tests for artificial data y_1 with change in μ (left) and y_2 with change in ϕ (right).

```
R> y2 <- c(rbeta(100, 0.3 * 4, 0.7 * 4), rbeta(100, 0.3 * 8, 0.7 * 8))
```

To capture instabilities in the parameters over “time” (i.e., the ordering of the observations), the generalized empirical fluctuation processes can be derived via

```
R> library("strucchange")
R> y1_gefp <- gefp(y1 ~ 1, fit = betareg)
R> y2_gefp <- gefp(y2 ~ 1, fit = betareg)
```

and visualized by

```
R> plot(y1_gefp, aggregate = FALSE)
R> plot(y2_gefp, aggregate = FALSE)
```

The resulting Figure 7 (replicating Figure 4 from Zeileis 2006a) shows two 2-dimensional fluctuation processes: one for y_1 (left) and one for y_2 (right). Both fluctuation processes behave as expected: There is no excessive fluctuation of the process pertaining to the parameter that remained constant while there is a clear peak at about the time of the change in the parameter with the shift. In both series the structural change is significant due to the crossing of the red boundary that corresponds to the 5% critical value. For further details see Zeileis (2006a).

6. Summary

This paper addressed the R implementation of the class of beta regression models available in the **betareg** package. We have presented the fixed and variable dispersion beta regression

models, described how one can model rates and proportions using **betareg** and presented several empirical examples reproducing previously published results. Future research and implementation shall focus on the situation where the data contain zeros and/or ones (see e.g., Kieschnick and McCullough 2003). An additional line of research and implementation is that of dynamic beta regression models, such as the class of β ARMA models proposed by Rocha and Cribari-Neto (2009).

Acknowledgments

FCN gratefully acknowledges financial support from CNPq/Brazil. Both authors are grateful to A.B. Simas and A.V. Rocha for their work on the previous versions of the **betareg** package (up to version 1.2). Furthermore, detailed and constructive feedback from two anonymous reviewers, the associated editor, as well as from B. Grün was very helpful for enhancing both software and manuscript.

References

- Breusch TS, Pagan AR (1979). “A Simple Test for Heteroscedasticity and Random Coefficient Variation.” *Econometrica*, **47**, 1287–1294.
- Chambers JM, Hastie TJ (eds.) (1992). *Statistical Models in S*. Chapman & Hall, London.
- Cribari-Neto F, Lima LB (2007). “A Misspecification Test for Beta Regressions.” *Technical report*.
- Cribari-Neto F, Zeileis A (2010). “Beta Regression in R.” *Journal of Statistical Software*, **34**(2), 1–24. URL <http://www.jstatsoft.org/v34/i02/>.
- Espinheira PL, Ferrari SLP, Cribari-Neto F (2008a). “Influence Diagnostics in Beta Regression.” *Computational Statistics & Data Analysis*, **52**(9), 4417–4431.
- Espinheira PL, Ferrari SLP, Cribari-Neto F (2008b). “On Beta Regression Residuals.” *Journal of Applied Statistics*, **35**(4), 407–419.
- Ferrari SLP, Cribari-Neto F (2004). “Beta Regression for Modelling Rates and Proportions.” *Journal of Applied Statistics*, **31**(7), 799–815.
- Griffiths WE, Hill RC, Judge GG (1993). *Learning and Practicing Econometrics*. John Wiley & Sons, New York.
- Grün B, Kosmidis I, Zeileis A (2012). “Extended Beta Regression in R: Shaken, Stirred, Mixed, and Partitioned.” *Journal of Statistical Software*, **48**(11), 1–25. URL <http://www.jstatsoft.org/v48/i11/>.
- Kieschnick R, McCullough BD (2003). “Regression Analysis of Variates Observed on (0, 1): Percentages, Proportions and Fractions.” *Statistical Modelling*, **3**(3), 193–213.
- Koenker R (1981). “A Note on Studentizing a Test for Heteroscedasticity.” *Journal of Econometrics*, **17**, 107–112.

- McCullagh P, Nelder JA (1989). *Generalized Linear Models*. 2nd edition. Chapman & Hall, London.
- Mittelhammer RC, Judge GG, Miller DJ (2000). *Econometric Foundations*. Cambridge University Press, New York.
- Nocedal J, Wright SJ (1999). *Numerical Optimization*. Springer-Verlag, New York.
- Ospina R, Cribari-Neto F, Vasconcellos KLP (2006). “Improved Point and Interval Estimation for a Beta Regression Model.” *Computational Statistics & Data Analysis*, **51**(2), 960–981.
- Prater NH (1956). “Estimate Gasoline Yields from Crudes.” *Petroleum Refiner*, **35**(5), 236–238.
- Prentice RL (1986). “Binary Regression Using an Extended Beta-Binomial Distribution, With Discussion of Correlation Induced by Covariate Measurement.” *Journal of the American Statistical Association*, **81**(394), 321–327.
- Ramsey JB (1969). “Tests for Specification Error in Classical Linear Least Squares Regression Analysis.” *Journal of the Royal Statistical Society B*, **31**, 350–371.
- R Development Core Team (2009). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL <http://www.R-project.org/>.
- Rocha AV, Cribari-Neto F (2009). “Beta Autoregressive Moving Average Models.” *Test*, **18**, 529–545.
- Rocha AV, Simas AB (2010). “Influence Diagnostics in a General Class of Beta Regression Models.” *Test*, **20**, 95–119.
- Simas AB, Barreto-Souza W, Rocha AV (2010). “Improved Estimators for a General Class of Beta Regression Models.” *Computational Statistics & Data Analysis*, **54**(2), 348–366.
- Simas AB, Rocha AV (2006). ***betareg***: Beta Regression. R package version 1.2, URL <http://CRAN.R-project.org/src/contrib/Archive/betareg/>.
- Smithson M, Verkuilen J (2006). “A Better Lemon Squeezer? Maximum-Likelihood Regression with Beta-Distributed Dependent Variables.” *Psychological Methods*, **11**(1), 54–71.
- Vasconcellos KLP, Cribari-Neto F (2005). “Improved Maximum Likelihood Estimation in a New Class of Beta Regression Models.” *Brazilian Journal of Probability and Statistics*, **19**(1), 13–31.
- Wei BC, Hu YQ, Fung WK (1998). “Generalized Leverage and Its Applications.” *Scandinavian Journal of Statistics*, **25**(1), 25–37.
- Williams DA (1982). “Extra Binomial Variation in Logistic Linear Models.” *Applied Statistics*, **31**(2), 144–148.
- Zeileis A (2004). “Econometric Computing with HC and HAC Covariance Matrix Estimators.” *Journal of Statistical Software*, **11**(10), 1–17. URL <http://www.jstatsoft.org/v11/i10/>.

- Zeileis A (2006a). “Implementing a Class of Structural Change Tests: An Econometric Computing Approach.” *Computational Statistics & Data Analysis*, **50**(11), 2987–3008.
- Zeileis A (2006b). “Object-Oriented Computation of Sandwich Estimators.” *Journal of Statistical Software*, **16**(9), 1–16. URL <http://www.jstatsoft.org/v16/i09/>.
- Zeileis A, Croissant Y (2010). “Extended Model Formulas in R: Multiple Parts and Multiple Responses.” *Journal of Statistical Software*, **34**(1), 1–13. URL <http://www.jstatsoft.org/v34/i01/>.
- Zeileis A, Hothorn T (2002). “Diagnostic Checking in Regression Relationships.” *R News*, **2**(3), 7–10. URL <http://CRAN.R-project.org/doc/Rnews/>.
- Zeileis A, Kleiber C, Jackman S (2008). “Regression Models for Count Data in R.” *Journal of Statistical Software*, **27**(8), 1–25. URL <http://www.jstatsoft.org/v27/i08/>.
- Zeileis A, Leisch F, Hornik K, Kleiber C (2002). “**strucchange**: An R Package for Testing for Structural Change in Linear Regression Models.” *Journal of Statistical Software*, **7**(2), 1–38. URL <http://www.jstatsoft.org/v07/i02/>.

Affiliation:

Francisco Cribari-Neto
 Departamento de Estatística, CCEN
 Universidade Federal de Pernambuco
 Cidade Universitária
 Recife/PE 50740-540, Brazil
 E-mail: cribari@ufpe.br
 URL: <http://www.de.ufpe.br/~cribari/>

Achim Zeileis
 Department of Statistics
 Universität Innsbruck
 Universitätsstr. 15
 6020 Innsbruck, Austria
 E-mail: Achim.Zeileis@R-project.org
 URL: <http://statmath.wu.ac.at/~zeileis/>

A Short Course in Beta Regression Models

Raydonal Ospina Martínez

Department of Statistics
Federal University of Pernambuco
Cidade Universitária
Recife/PE 50740–540, Brazil

E-mail: raydonal@ufpe.br
URL: <http://www.de.ufpe.br/~raydonal/>

XXI SIMPOSIO DE ESTADÍSTICA “MODELOS DE REGRESIÓN”
19 al 23 de julio de 2011

This material is based mainly on the papers [Cribari-Neto, F., and Zeileis, A. (2010). Beta Regression in R. *Journal of Statistical Software*, 34(2), 1-24.] and [Simas, A.B., and Barreto-Souza, W., and Rocha, A.V. (2010). Improved Estimators for a General Class of Beta Regression Models. *Computational Statistics & Data Analysis*, 54(2), 348-366.]. Some snippets of text in the document are literal copies of the papers.

Outline of the Course

- ▶ Two lectures, each 1.5 hours.
 - ▶ Lecture 1. Basics in Beta Regression Models.
 - ▶ Motivation.
 - ▶ Model.
 - ▶ Estimation.
 - ▶ Testing.
 - ▶ Diagnostics.
 - ▶ Lecture 2. Beta regression models in practice.
 - ▶ Applications.

Lecture 1. Basics in Beta Regression Models.

Motivation

How should we perform a regression analysis in which the dependent variable is restricted to the standard unit interval such as rates and proportions?

Possible solution

- ▶ Transform the dependent variable y so that it assumes values on the real line. (Example: $\tilde{y} = \log(y/(1 - y))$.)
- ▶ This approach, however, has drawbacks, one of them being the fact that the model parameters cannot be easily interpreted in terms of the original response.
- ▶ Regressions involving data from the unit interval are typically heteroskedastic: they display more variation around the mean and less variation as we approach the lower and upper limits of the standard unit interval.
- ▶ Another shortcoming is that measures of proportions typically display asymmetry, and hence inference based on the normality assumption can be misleading.

An intelligent approach

- ▶ [Ferrari and Cribari-Neto, 2004] proposed a regression model for continuous variates that assume values in the standard unit interval, e.g., rates, proportions, or concentrations indices.
- ▶ The model is based on the assumption that the response is beta-distributed, they called their model *the beta regression model*.
- ▶ The regression parameters are interpretable in terms of the mean of y (the variable of interest) and the model is naturally heteroskedastic and easily accommodates asymmetries.
- ▶ A variant of the beta regression model that allows for nonlinearities and variable dispersion was proposed by [Simas et al., 2010].

- ▶ The chief motivation for the beta regression model lies in the flexibility delivered by the assumed beta law.
- ▶ The beta density can assume a number of different shapes depending on the combination of parameter values, including left- and right-skewed or the flat shape of the uniform density (which is a special case of the more general beta density).
- ▶ The beta density is usually expressed as

$$f(y; p, q) = \frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q)} y^{p-1} (1-y)^{q-1}, \quad 0 < y < 1,$$

where $p, q > 0$ and $\Gamma(\cdot)$ is the gamma function.

- ▶ A beta regression model based on this parameterization was proposed by [Vasconcellos and Cribari-Neto, 2005].

Density

- ▶ [Ferrari and Cribari-Neto, 2004] proposed a different parameterization by setting $\mu = p/(p + q)$ and $\phi = p + q$:

$$f(y; \mu, \phi) = \frac{\Gamma(\phi)}{\Gamma(\mu\phi)\Gamma((1-\mu)\phi)} y^{\mu\phi-1} (1-y)^{(1-\mu)\phi-1}, \quad 0 < y < 1,$$

with $0 < \mu < 1$ and $\phi > 0$.

- ▶ We write $y \sim \mathcal{B}(\mu, \phi)$. Here, $E(y) = \mu$ and $\text{VAR}(y) = \mu(1 - \mu)/(1 + \phi)$.
- ▶ The parameter ϕ is known as the precision parameter since, for fixed μ , the larger ϕ the smaller the variance of y ; ϕ^{-1} is a dispersion parameter.

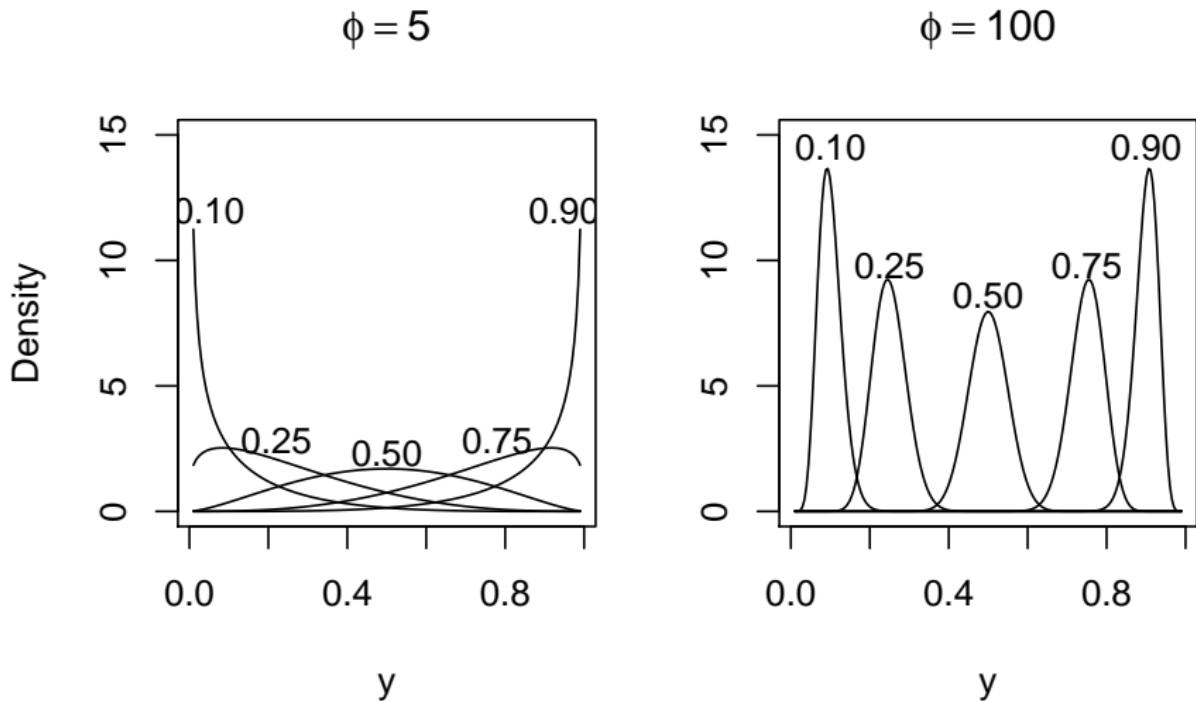


Figure 1: Probability density functions for beta distributions with varying parameters $\mu = 0.10, 0.25, 0.50, 0.75, 0.90$ and $\phi = 5$ (left) and $\phi = 100$ (right).

The model - General version -

- ▶ Let $y = (y_1, \dots, y_n)^T$ be a random sample, where $y_i \sim \mathcal{B}(\mu_i, \phi_i)$, $i = 1, \dots, n$.
- ▶ Suppose the mean and the precision parameter of y_i satisfies the following functional relations:

$$\begin{aligned} g_1(\mu_i) &= \eta_{1i} = f_1(x_i^T; \beta) \\ g_2(\phi_i) &= \eta_{2i} = f_2(z_i^T; \theta). \end{aligned} \tag{1}$$

The model

- ▶ Here, $\beta = (\beta_1, \dots, \beta_k)^T$ and $\theta = (\theta_1, \dots, \theta_h)^T$ are vectors of unknown regression parameters which are assumed to be functionally independent, $\beta \in \mathbb{R}^k$ and $\theta \in \mathbb{R}^h$, $k + h < n$.
- ▶ η_{1i} and η_{2i} are predictors.
- ▶ $X_{i1}, \dots, X_{iq_1}, Z_{i1}, \dots, Z_{iq_2}$ are observations on q_1 and q_2 known covariates, which need not to be exclusive.
- ▶ We assume that the derivative matrices $\tilde{X} = \partial\eta_1/\partial\beta$ and $\tilde{Z} = \partial\eta_2/\partial\theta$ have rank k and h , respectively.
- ▶ ϕ^{-1} is a dispersion parameter.

The model

- ▶ The link functions $g_1 : (0, 1) \rightarrow \mathbb{R}$ and $g_2 : (0, \infty) \rightarrow \mathbb{R}$ are strictly monotonic and twice differentiable.
- ▶ A number of different link functions can be used, such as the logit specification $g_1(\mu) = \log\{\mu/(1 - \mu)\}$, the probit function $g_1(\mu) = \Phi^{-1}(\mu)$, where $\Phi(\cdot)$ denotes the standard normal distribution function, the complementary log-log function $g_1(\mu) = \log\{-\log(1 - \mu)\}$, among others, and for g_2 , $g_2(\phi) = \log \phi$, the logarithmic function, $g_2(\phi) = \sqrt{\phi}$, the square root function, $g_2(\phi) = \phi$ (with special attention on the positivity of the estimates), among others.
- ▶ A rich discussion of link functions can be found in McCullagh and Nelder (1989); see also Atkinson (1985, Chapter 7).

Especial cases

- ▶ The linear beta regression model

We have, in (1), $g_1(\mu_i) = g(\mu_i) = \eta_{1i} = x_i^T \beta$, where $g(\cdot)$ is some link function and $g_2(\phi) = \phi_i = \phi$. We have that in this case $\tilde{X} = X$ and $\tilde{Z} = \mathbf{1}$ where X is the matrix of covariates with rows given by x_i^T .

- ▶ The linear beta regression model with dispersion covariates

The precision parameter ϕ vary through a linear regression structure. More precisely, in (1), $g_1(\mu_i) = g(\mu_i) = \eta_{1i} = x_i^T \beta$, and $g_2(\phi_i) = \eta_{2i} = z_i^T \theta$. In this case $\tilde{X} = X$ and $\tilde{Z} = Z$ where X and Z are covariates matrix with rows given by x_i^T and z_i^T , respectively.

Especial cases

- ▶ The nonlinear beta regression model with linear dispersion covariates

The precision parameter ϕ vary through a linear regression structure. For this model the equation (1) becomes

$$g_1(\mu_i) = \eta_{1i} = f(x_i^T; \beta) \text{ and } g_2(\phi_i) = \eta_{2i} = z_i^T \theta,$$

where $\beta \in \mathbb{R}^k$ and $\theta \in \mathbb{R}^h$. Then, we have that for this model \tilde{X} remaining the same, and $\tilde{Z} = Z$, where Z is the matrix of covariates with rows given by z_i^T .

The log-likelihood function

- ▶ For this class of beta regression models has

$$\ell(\beta, \theta) = \sum_{i=1}^n \ell_i(\mu_i, \phi_i), \quad (2)$$

where

$$\begin{aligned}\ell_i(\mu_i, \phi_i) &= \log \Gamma(\phi_i) - \log \Gamma((1 - \mu_i)\phi_i) + (\mu_i\phi_i - 1) \log y_i \\ &\quad + \{(1 - \mu_i)\phi_i - 1\} \log(1 - y_i);\end{aligned}$$

$\mu_i = g_1^{-1}(\eta_{1i})$, $\phi_i = g_2^{-1}(\eta_{2i})$, as defined in (1), are functions of β and θ , respectively.

- ▶ It is possible to show that this beta regression model is regular.

Score function for β 's

- ▶ The components of the score vector, obtained by differentiation of the log-likelihood function with respect to the parameters.
- ▶ For $r = 1, \dots, k$, as

$$U_r(\beta, \theta) = \frac{\partial \ell(\beta, \theta)}{\partial \beta_r} = \sum_{i=1}^n \phi_i (y_i^* - \mu_i^*) \frac{d\mu_i}{d\eta_{1i}} \frac{\partial \eta_{1i}}{\partial \beta_k},$$

where $d\mu_i/d\eta_{1i} = 1/g'_1(\mu_i)$, $y_i^* = \log(y_i/(1-y_i))$, $\mu_i^* = \psi(\mu_i \phi_i) - \psi((1-\mu_i)\phi_i)$, and $\psi(\cdot)$ is the digamma¹ function.

¹We denote generally the polygamma function by $\psi^{(m)}(\cdot)$, $m = 0, 1, \dots$, where $\psi^{(m)}(x) = (d^{m+1}/dx^{m+1}) \log \Gamma(x)$, $x > 0$.

Score function for θ 's

- ▶ For

$$U_R(\beta, \theta) = \frac{\partial \ell(\beta, \theta)}{\partial \theta_R}$$
$$= \sum_{i=1}^n \{ \mu_i(y_i^* - \mu_i^*) + \psi(\phi_i) - \psi((1 - \mu_i)\phi_i) + \log(1 - y_i) \} \frac{d\phi_i}{d\eta_{2i}} \frac{\partial \eta_{2i}}{\partial \theta_R},$$

where $d\phi_i/d\eta_{2i} = 1/g'_2(\phi_i)$, and $R = 1, \dots, h$.

- ▶ Further, the regularity conditions implies that

$$E \left(\log \frac{y_i}{1 - y_i} \right) = \psi(\mu_i \phi_i) - \psi((1 - \mu_i)\phi_i),$$

and

$$E \{ \log(1 - y_i) \} = \psi((1 - \mu_i)\phi_i) - \psi(\phi_i).$$

Score vector

- ▶ Consider the complete parameter vector $\zeta = (\beta^T, \theta^T)^T$.
- ▶ Define the vectors $y^* = (y_1^*, \dots, y_n^*)^T$, $\mu^* = (\mu_1^*, \dots, \mu_n^*)^T$, $v = (v_1, \dots, v_n)^T$, where
 $v_i = \mu_i(y_i^* - \mu_i^*) + \psi(\phi_i) - \psi((1 - \mu_i)\phi_i) + \log(1 - y_i)$.
- ▶ The matrix $T_1 = \text{diag}(d\mu_i/d\eta_{1i})$, $T_2 = \text{diag}(d\phi_i/d\eta_{2i})$, $\Phi = \text{diag}(\phi_i)$.
- ▶ The $(k + h) \times 1$ dimensional score vector $U(\zeta)$ in the form $(U_\beta(\beta, \theta)^T, U_\theta(\beta, \theta)^T)^T$, with

$$\begin{aligned} U_\beta(\beta, \theta) &= \tilde{X}^T \Phi T_1 (y^* - \mu^*), \\ U_\theta(\beta, \theta) &= \tilde{Z}^T T_2 v. \end{aligned} \tag{3}$$

Fisher's information matrix

- ▶ It is possible to obtain Fisher's information matrix for the parameter vector $\zeta = (\beta^T, \theta^T)^T$ as

$$K(\zeta) = P^T W P.$$

- ▶ Define P as the $2n \times (k + h)$ dimensional matrix

$$P = \begin{pmatrix} \tilde{X} & 0 \\ 0 & \tilde{Z} \end{pmatrix}. \quad (4)$$

- ▶ let W be the $2n \times 2n$ matrix

$$W = \begin{pmatrix} W_{\beta\beta} & W_{\beta\theta} \\ W_{\beta\theta} & W_{\theta\theta} \end{pmatrix}, \quad (5)$$

► Here

$$W_{\beta\beta} = \text{diag} \left(\phi_i^2 a_i \left(\frac{d\mu_i}{d\eta_{1i}} \right)^2 \right),$$

$$W_{\beta\theta} = \text{diag} \left(\phi_i \{ \mu_i a_i - \psi'((1 - \mu_i)\phi_i) \} \left(\frac{d\mu_i}{d\eta_{1i}} \right) \left(\frac{d\phi_i}{d\eta_{2i}} \right) \right),$$

$$W_{\theta\theta} = \text{diag} \left(b_i \left(\frac{d\phi_i}{d\eta_{2i}} \right)^2 \right).$$

- where, $a_i = \psi'((1 - \mu_i)\phi_i) + \psi'(\mu_i\phi_i)$ and
 $b_i = \psi'((1 - \mu_i)\phi_i)(1 - \mu_i)^2 + \psi'(\mu_i\phi_i)\mu_i^2 - \psi'(\phi_i)$.

Estimation

- ▶ Note that $W_{\beta\theta} \neq 0$, thus indicating that the parameters β and θ are not orthogonal, in contrast to the class of generalized linear models (McCullagh and Nelder, 1989), where such orthogonality holds.
- ▶ Nevertheless, the MLEs $\hat{\zeta}$ and $K(\hat{\zeta})$ are consistent estimators of ζ and $K(\zeta)$, respectively, where $K(\hat{\zeta})$ is the Fisher's information matrix evaluated at $\hat{\zeta}$.
- ▶ The MLEs of β and θ are obtained as the solution of the nonlinear system $U(\zeta) = 0$. In practice, the MLEs can be obtained through a numerical maximization of the log-likelihood function using a nonlinear optimization algorithm, e.g., BFGS. For details, see Press et al. (1992).

Estimation

- ▶ As initial guesses for β and θ , we suggest one to obtain the estimates from the following normal nonlinear regression model with dispersion covariates: $g_1(\mu_i) = f_1(x_i; \beta)$ and $g_2(\sigma_i^2) = f_1(z_i; \theta)$.
- ▶ This will produce $\hat{\beta}^{(0)}$ and $\hat{\theta}^{(0)}$, which will be our initial guesses, where for this initial guess we assume that $Y_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$.
- ▶ Note that we are viewing the normal nonlinear model above in the generalized nonlinear model sense, since we are using the link functions.
- ▶ The R package `n1me2` fits such a model.

- ▶ The beta distribution has been incorporated in the GAMLSS framework (Rigby & Stasinopoulos, 2005).
- ▶ GAMLSS allows the flexible modeling of each of the three parameters that index the distribution using parametric terms involving linear or nonlinear predictors, smooth nonparametric terms (e.g., cubic splines or loess), and random effects.
- ▶ Maximum (penalized) likelihood estimation is approached through a Newton-Raphson or Fisher scoring algorithm with the backfitting algorithm for the additive components.

- ▶ The GAMLSS approach consists of an application of the `gamlss` functions, which are fully documented in the `gamlss` package (Stasinopoulos & Rigby, 2007; see also <http://www.gamlss.org>).
- ▶ The structure of the `gamlss` functions is familiar to readers who are used to the R (or S-Plus) syntax (the `glm` function, in particular).
- ▶ The set of `gamlss` packages can be freely downloaded from the R library at <http://www.r-project.org/>.

In large samples

- If $J(\zeta) = \lim_{n \rightarrow \infty} K(\zeta)/n$ exists and is nonsingular, we have that

$$\sqrt{n} (\hat{\zeta} - \zeta) \xrightarrow{d} N_{k+h}(0, J(\zeta)^{-1}).$$

- If ζ_r denotes the r th component of ζ , it follows that

$$(\hat{\zeta} - \zeta) \{K(\hat{\zeta})^{rr}\}^{-1/2} \xrightarrow{d} N(0, 1),$$

where $K(\hat{\zeta})^{rr}$ is the r th diagonal element of $K(\hat{\zeta})^{-1}$.

- For $0 < \alpha < 1/2$, and q_γ representing the γ quantile of the $N(0, 1)$ distribution, we have, for $r = 1, \dots, k$,

$$\hat{\beta}_r \pm q_{1-\alpha/2} (K(\hat{\zeta})^{rr})^{1/2}$$

as the limits of asymptotic confidence intervals for β_r with asymptotic coverage of $100(1 - \alpha)\%$.

Testing

- ▶ The likelihood ratio, Rao's score, and Wald's (W) statistics to test hypotheses on the parameters can be calculated from the log-likelihood function, the score vector, the Fisher information matrix, and its inverse given above.
- ▶ Their null distributions are usually unknown and the tests rely on asymptotic approximations.
- ▶ In large samples, a chi-squared distribution can be used as an approximation to the true null distributions.

- ▶ For testing the significance of the i th regression parameter that models μ , one can use the signed square root of Wald's statistic, $\hat{\beta}_i / \text{s.e.}(\hat{\beta}_i)$, where $\text{s.e.}(\hat{\beta}_i)$ is the asymptotic standard error of the MLE of β_i obtained from the inverse of Fisher's information matrix evaluated at the maximum likelihood estimates.
- ▶ The limiting null distribution of the test statistic is standard normal.
- ▶ Significance tests on the θ 's can be performed in a similar fashion.

- ▶ A RESET-type misspecification test for fixed dispersion beta regression, similar to that of [Ramsey, 1969] for linear regressions, was proposed by [Cribari-Neto and Lima, 2007].
- ▶ The authors considered different variants of the test and concluded that the best performing testing strategy in finite samples is to include $\hat{\eta}_i^2 = (x_i^\top \hat{\beta})^2$, where $\hat{\beta}$ denotes the ML estimator of β , as an additional regressors in an augmented (artificial) regression, and then test its exclusion using a score test.
- ▶ Rejection of the null hypothesis suggests that the model is misspecified. Misspecification can follow, e.g., from incorrectly specifying the link function, from neglecting nonlinearities or from omitting important regressors.

Model selection

- ▶ Nested beta regression models can be compared via the likelihood ratio test, using twice the difference between the maximized log-likelihoods of a full model and a restricted model whose covariates are a subset of the full model.
- ▶ Information criteria, such as the generalized Akaike information criterion (GAIC), $GAIC = \hat{D} + d\varphi$, can be used for comparing non-nested models.
- ▶ $\hat{D} = -2\hat{\ell}$ is the global fitted deviance (Rigby & Stasinopoulos, 2005), $\hat{\ell}$ is the maximized log-likelihood and d is the dimension of ζ .
- ▶ The model with the smallest GAIC is then selected.

Diagnostics - Generalized leverage -

- The generalized leverage (GL) proposed by Wei et al. (1998) is defined by

$$GL(\tilde{\zeta}) = \frac{\partial \tilde{y}}{\partial y^T}$$

where ζ is an s -vector such that $E(y) = \mu(\zeta)$ and $\tilde{\zeta}$ is an estimator of ζ , with $\tilde{y} = \mu(\tilde{\zeta})$.

- The (i, l) element of $GL(\tilde{\zeta})$ i.e. the GL of the estimator $\tilde{\zeta}$ at (i, l) , is the instantaneous rate of change in the i th predicted value with respect to the l th response value.
- The GL is obtained as

$$D_{\zeta} \left(-\frac{\partial^2 \ell}{\partial \zeta \partial \zeta^T} \right)^{-1} \frac{\partial^2 \ell}{\partial \zeta \partial y^T} \quad (6)$$

evaluated at $\hat{\zeta}$, where $D_{\zeta} = \partial \mu / \partial \zeta^T$.

Diagnostics - Residuals -

- ▶ The raw response residuals $y_i - \hat{\mu}_i$ are typically not used due to the heteroskedasticity inherent in the model.
- ▶ A natural alternative are *Pearson residuals* which [Ferrari and Cribari-Neto, 2004] call *standardized ordinary residuals* and define as

$$r_{P,i} = \frac{y_i - \hat{\mu}_i}{\sqrt{\widehat{\text{VAR}}(y_i)}}, \quad (7)$$

where $\widehat{\text{VAR}}(y_i) = \hat{\mu}_i(1 - \hat{\mu}_i)/(1 + \hat{\phi}_i)$, $\hat{\mu}_i = g_1^{-1}(f_1(x_i^\top; \hat{\beta}))$, and $\hat{\phi}_i = g_2^{-1}(f_2(z_i^\top; \hat{\gamma}))$.

- ▶ Deviance residuals can be defined in the standard way via signed contributions to the excess likelihood.

- ▶ Further residuals were proposed by [Espinheira et al., 2008b], in particular one residual with better properties that they named *standardized weighted residual 2 our score residual*:

$$r_{\text{sw2},i} = \frac{y_i^* - \hat{\mu}_i^*}{\sqrt{\hat{v}_i(1 - h_{ii})}}, \quad (8)$$

where $y_i^* = \log\{y_i/(1 - y_i)\}$ and $\mu_i^* = \psi(\mu_i\phi) - \psi((1 - \mu_i)\phi)$, $\psi(\cdot)$ denoting the digamma function. Standardization is then by $v_i = \{\psi'(\mu_i\phi) + \psi'((1 - \mu_i)\phi)\}$ and h_{ii} , the i th diagonal element of the hat matrix.

- ▶ The problem with the above residual is that it does not take into account the discrepant values in the covariate matrix Z .

- ▶ [Rocha and Simas, 2010] proposed a modification of the score residual that takes into account all the discrepancy of the model.

$$r_{\text{MS},i} = \frac{y_i^* - \hat{\mu}_i^*}{\sqrt{\hat{\nu}_i(1 - \hat{G}L_{ii})}}, \quad (9)$$

where GL_{ii} is the i th element of the diagonal of the generalized leverage matrix given in (6).

- ▶ Another alternative. Use the randomized quantile residual (Dunn & Smyth, 1996). It is a randomized version of the Cox & Snell (1968) residual and given by

$$r_t^q = \Phi^{-1}(u_t), \quad t = 1, \dots, n, \quad (10)$$

where $\Phi(\cdot)$ denotes the standard normal distribution function.

Diagnostics -Envelope plot -

- ▶ A plot of these residuals against the index of the observations (i) should show no detectable pattern.
- ▶ A detectable trend in the plot of some residual against the predictors may be suggestive of link function misspecification.
- ▶ Normal probability plots with simulated envelopes are a helpful diagnostic tool (Atkinson, 1985).
- ▶ Simulation results not presented here indicated that the residuals perform well in detecting whether the distribution assumption is incorrect.

Diagnostics - Global goodness-of-fit measure -

- ▶ A simple global goodness-of-fit measure is a pseudo R^2 , say R_p^2 defined by the square of the sample correlation coefficient between the outcomes, y_1, \dots, y_n , and their corresponding predicted values, $\hat{\mu}_1, \dots, \hat{\mu}_n$.
- ▶ A perfect agreement between the y 's and $\hat{\mu}$'s yields $R_p^2 = 1$.
- ▶ Other pseudo R^2 's are defined as $R_p^{2*} = 1 - \log \hat{L} / \log \hat{L}_0$ (McFadden, 1974) and $R_{LR}^2 = 1 - (\hat{L}_0 / \hat{L})^{2/n}$ (Cox and Snell, 1989, p. 208-209), where \hat{L}_0 and \hat{L} are the maximized likelihood functions of the null model and the fitted model, respectively.
- ▶ The ratio of the likelihoods or log-likelihoods may be regarded as measures of the improvement over the model with only three parameters μ and ϕ , achieved by the model under investigation.

Diagnostics - Influence measures -

- ▶ A well-known measure of the influence of each observation on the regression parameter estimates is the likelihood displacement (Cook & Weisberg, 1982, Ch. 3).
- ▶ The likelihood displacement that results from removing the t th observation from the data is defined by

$$LD_t = 2\{\ell(\hat{\zeta}) - \ell(\hat{\zeta}_{(t)})\},$$

where d is the dimension of ζ and $\hat{\zeta}_{(t)}$ is the MLE of ζ obtained after removing the t th observation from the data.

- ▶ LD_t combines leverage and residuals.
- ▶ It is common practice to plot LD_t against t .
- ▶ Other diagnostic measures can be considered, such as local influence measures (Cook, 1986).

Lecture 2. Beta regression models in practice.

betareg: An implementation of beta regression models using R

- ▶ Beta regression as suggested by Ferrari and Cribari-Neto (2004) and extended by Simas, Barreto-Souza, and Rocha (2010) is implemented in the **betareg** package.
- ▶ It is modeled to be beta-distributed with parametrization using mean and precision parameter.
- ▶ The mean is linked, as in generalized linear models (GLMs), to the responses through a link function and a linear predictor.
- ▶ The precision parameter phi can be linked to another (potentially overlapping) set of regressors through a second link function, resulting in a model with variable dispersion.

- ▶ Estimation is performed by maximum likelihood (ML) via `optim()` using analytical gradients and (by default) starting values from an auxiliary linear regression of the transformed response.
- ▶ The main model-fitting function in **betareg** is `betareg()` which takes a fairly standard approach for implementing ML regression models in R: `formula` plus `data` is used for model and data specification, then the likelihood and corresponding gradient (or estimating function) is set up, `optim()` is called for maximizing the likelihood, and finally an object of S3 class “`betareg`” is returned for which a large set of methods to standard generics is available.
- ▶ The workhorse function is `betareg.fit()` which provides the core computations without `formula`-related data pre-and post-processing.

- ▶ The arguments of `betareg()` are

```
betareg(formula, data, subset, na.action, weights,  
        offset, link = "logit", link.phi = NULL,  
        control = betareg.control(...), model =  
        TRUE, y = TRUE, x = FALSE, ...)
```

Some methods for **betareg**

Function	Description
<code>print()</code> <code>summary()</code>	simple printed display with coefficient estimates standard regression output (coefficient estimates, standard errors, partial Wald tests); returns an object of class "summary.betareg" containing the relevant summary statistics (which has a <code>print()</code> method)
<code>coef()</code> <code>vcov()</code> <code>predict()</code>	extract coefficients of model (full, mean, or precision components), a single vector of all coefficients by default associated covariance matrix (with matching names) predictions (of means μ_i , linear predictors η_{1i} , precision parameter ϕ_i , or variances $\mu_i(1 - \mu_i)/(1 + \phi_i)$) for new data
 <code>fitted()</code> <code>residuals()</code>	fitted means for observed data extract residuals [Espinheira et al., 2008b, deviance, Pearson, response, or different weighted residuals, see], defaulting to standardized weighted residuals 2 from Equation 8
 <code>estfun()</code>	compute empirical estimating functions (or score functions), evaluated at observed data and estimated parameters [Zeileis, 2006, see]
 <code>bread()</code>	extract "bread" matrix for sandwich estimators [Zeileis, 2006, see]

Function	Description
terms() model.matrix() model.frame() logLik()	extract terms of model components extract model matrix of model components extract full original model frame extract fitted log-likelihood
plot() hatvalues() cooks.distance() gleverage()	diagnostic plots of residuals, predictions, leverages etc. hat values (diagonal of hat matrix) (approximation of) Cook's distance compute generalized leverage [Wei et al., 1998, Rocha and Simas, 2010]
coeftest() waldtest() linear.hypothesis() lrtest() AIC()	partial Wald tests of coefficients Wald tests of nested models Wald tests of linear hypotheses likelihood ratio tests of nested models compute information criteria (AIC, BIC, ...)

Table 1: Functions and methods for objects of class “betareg”. The first four blocks refer to methods, the last block contains generic functions whose default methods work because of the information supplied by the methods above.

Beta regression in practice: An illustration

Prater's gasoline yield data

- ▶ We estimate and compare various flavors of beta regression models for the gasoline yield data of [Prater, 1956]
- ▶ The variable of interest is `yield`, the proportion of crude oil converted to gasoline after distillation and fractionation, for which a beta regression model is rather natural.
- ▶ [Ferrari and Cribari-Neto, 2004] employ two explanatory variables: `temp`, the temperature (in degrees Fahrenheit) at which all gasoline has vaporized, and `batch`, a factor indicating ten unique batches of conditions in the experiments (depending on further variables).

The basic model: Estimation, inference, diagnostics

[Ferrari and Cribari-Neto, 2004] start out with a model where yield depends on batch and temp, employing the standard logit or log-log link. In **betareg**, this can be fitted via

```
R> data("GasolineYield", package = "betareg")
R> gy_logit <- betareg(yield ~ batch + temp,
+   data = GasolineYield)

R> gy_loglog <- betareg(yield ~ batch + temp,
+   data = GasolineYield, link = "loglog")
```

```

R> summary(gy_logit)

Call:
betareg(formula = yield ~ batch + temp, data = GasolineYield)

Standardized weighted residuals 2:
      Min     1Q   Median     3Q    Max 
-2.8750 -0.8149  0.1601  0.8384  2.0483 

Coefficients (mean model with logit link):
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) -6.1595710  0.1823247 -33.784 < 2e-16 *** 
batch1       1.7277289  0.1012294  17.067 < 2e-16 *** 
batch2       1.3225969  0.1179021  11.218 < 2e-16 *** 
batch3       1.5723099  0.1161045  13.542 < 2e-16 *** 
batch4       1.0597141  0.1023598  10.353 < 2e-16 *** 
batch5       1.1337518  0.1035232  10.952 < 2e-16 *** 
batch6       1.0401618  0.1060365  9.809 < 2e-16 *** 
batch7       0.5436922  0.1091275  4.982 6.29e-07 *** 
batch8       0.4959007  0.1089257  4.553 5.30e-06 *** 
batch9       0.3857929  0.1185933  3.253  0.00114 **  
temp        0.0109669  0.0004126  26.577 < 2e-16 *** 

Phi coefficients (precision model with identity link):
            Estimate Std. Error z value Pr(>|z|)    
(phi)      440.3      110.0   4.002 6.29e-05 *** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 

Log-likelihood:  84.8 on 12 Df
Pseudo R-squared: 0.9617
Number of iterations in BFGS optimization: 51

```

Prater's gasoline yield data

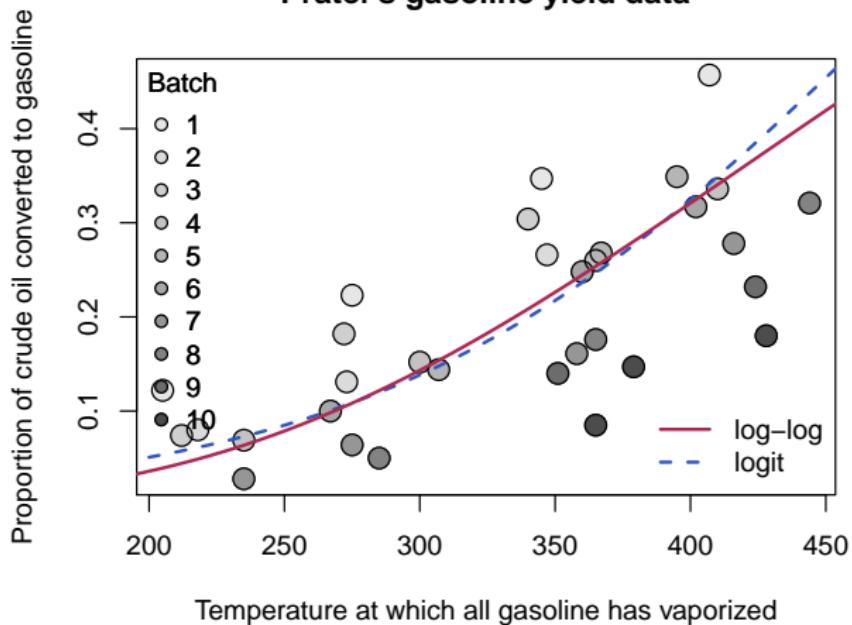


Figure 2: Gasoline yield data from [Prater, 1956]: Proportion of crude oil converted to gasoline explained by temperature (in degrees Fahrenheit) at which all gasoline has vaporized and given batch (indicated by gray level). Fitted curves correspond to beta regressions `gy_loglog` with log-log link (solid, red) and `gy_logit` with logit link (dashed, blue). Both curves were evaluated at varying temperature with the intercept for batch 6 (i.e., roughly the average intercept).

Goodness of fit is assessed using different types of diagnostic displays shown in their Figure 2. This graphic can be reproduced (in a slightly different order) using the `plot()` method for “`betareg`” objects, see Figure 3.

```
R> set.seed(123)
R> plot(gy_logit, which = 1:4, type = "pearson")
R> plot(gy_logit, which = 5, type = "deviance",
+       sub.caption = "")
R> plot(gy_logit, which = 1, type = "deviance",
+       sub.caption = "")
```

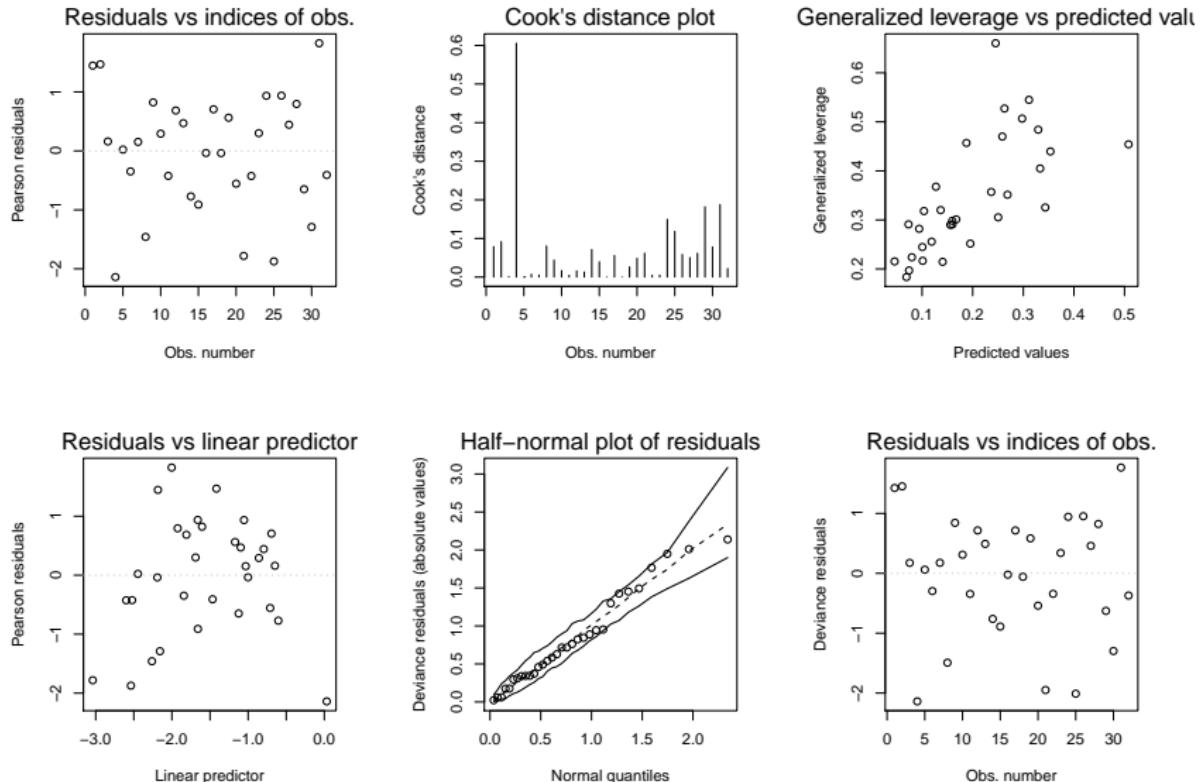


Figure 3: Diagnostic plots for beta regression model `gy_logit`.

As observation 4 corresponds to a large Cook's distance and large residual, [Ferrari and Cribari-Neto, 2004] decided to refit the model excluding this observation. While this does not change the coefficients in the mean model very much, the precision parameter ϕ increases clearly.

```
R> gy_logit4 <- update(gy_logit, subset = -4)  
R> coef(gy_logit, model = "precision")
```

```
    (phi)  
440.2783
```

```
R> coef(gy_logit4, model = "precision")  
    (phi)  
577.7907
```

Variable dispersion model

- ▶ The beta model already incorporates naturally a certain pattern in the variances of the response.
- ▶ It might be necessary to incorporate further regressors to account for heteroskedasticity [Simas et al., 2010].
- ▶ The Prater's gasoline yield data based on the same mean equation as above, but now with temperature `temp` as an additional regressor for the precision parameter ϕ_i :

```
R> gy_logit2 <- betareg(yield ~ batch  
+ + temp | temp, data = GasolineYield)
```

for which `summary(gy_logit2)` yields the MLE column in Table 19 of [Simas et al., 2010].

Here, only the parameters pertaining to ϕ_i are reported

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	1.3641103	1.2257813	1.1128	0.2658
temp	0.0145703	0.0036183	4.0268	5.653e-05 ***

Signif. codes:	0 '***'	0.001 '**'	0.01 '*'	0.05 '.'
	0.1 ' '	1		

which signal a significant improvement by including the `temp` regressor.

Instead of using this Wald test, the models can also be compared by means of a likelihood-ratio test (see their Table 18) that confirms the results:

```
R> library("lmtest")
R> lrtest(gy_logit, gy_logit2)

Likelihood ratio test

Model 1: yield ~ batch + temp
Model 2: yield ~ batch + temp | temp
#Df LogLik Df Chisq Pr(>Chisq)
1   12 84.798
2   13 86.977  1 4.359     0.03681 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that this can also be interpreted as testing the null hypothesis of equidispersion against a specific alternative of variable dispersion.

Selection of different link functions

- ▶ Selection of an appropriate link function can greatly improve the model fit [McCullagh and Nelder, 1989],
- ▶ We replicate parts of the analysis in Section 5 of [Cribari-Neto and Lima, 2007].
- ▶ This reconsiders Prater's gasoline yield data but employs a log-log link instead of the previously used (default) logit link

```
R> gy_loglog <- betareg(yield ~ batch + temp,  
+   data = GasolineYield, link = "loglog")
```

- ▶ Clearly the pseudo R^2 of the model is improved:

```
R> summary(gy_logit)$pseudo.r.squared
```

```
[1] 0.9617312
```

```
R> summary(gy_loglog)$pseudo.r.squared
```

```
[1] 0.9852334
```

- ▶ Similarly, the AIC² (and BIC) of the fitted model is not only superior to the logit model with fixed dispersion `gy_logit` but also to the logit model with variable dispersion `gy_logit2` considered in the previous section.

```
R> AIC(gy_logit, gy_logit2, gy_loglog)
```

	df	AIC
gy_logit	12	-145.5951
gy_logit2	13	-147.9541
gy_loglog	12	-168.3101

²Note that [Cribari-Neto and Lima, 2007] did not account for estimation of ϕ in their degrees of freedom. Hence, their reported AICs differ by 2.

- ▶ If `temp` were included as a regressor in the precision equation of `gy_loglog`, it would no longer yield significant improvements.
- ▶ Improvement of the model fit in the mean equation by adoption of the log-log link has waived the need for a variable precision equation.
- ▶ [Cribari-Neto and Lima, 2007] consider a sequence of diagnostic tests inspired by the RESET [Ramsey, 1969, regression specification error test] in linear regression models.
- ▶ To check for misspecifications, they consider powers of fitted means or linear predictors to be included as auxiliary regressors in the mean equation.
- ▶ In well-specified models, these should not yield significant improvements.

Analogous results can be obtained for type = "response" or higher powers.

```
R> lrtest(gy_logit, . ~ . + I(predict(gy_logit, type = "link")^2))  
Likelihood ratio test  
  
Model 1: yield ~ batch + temp  
Model 2: yield ~ batch + temp + I(predict(gy_logit, type = "link")^2)  
#Df LogLik Df  Chisq Pr(>Chisq)  
1  12 84.798  
2  13 96.001  1 22.407  2.205e-06 ***  
---  
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
  
R> lrtest(gy_loglog, . ~ . + I(predict(gy_loglog, type = "link")^2))  
Likelihood ratio test  
  
Model 1: yield ~ batch + temp  
Model 2: yield ~ batch + temp + I(predict(gy_loglog, type = "link")^2)  
#Df LogLik Df  Chisq Pr(>Chisq)  
1  12 96.155  
2  13 96.989  1 1.6671      0.1966
```

- ▶ The improvement of the model fit can also be brought out graphically by comparing absolute raw residuals (i.e., $y_i - \hat{\mu}_i$) from both models.
- ▶ A different diagnostic display that is useful in this situation [Cribari-Neto and Lima, 2007, and is employed by] is a plot of predicted values ($\hat{\mu}_i$) vs. observed values (y_i) for each model. This can be created by `plot(gy_logit, which = 6)` and `plot(gy_loglog, which = 6)`, respectively.

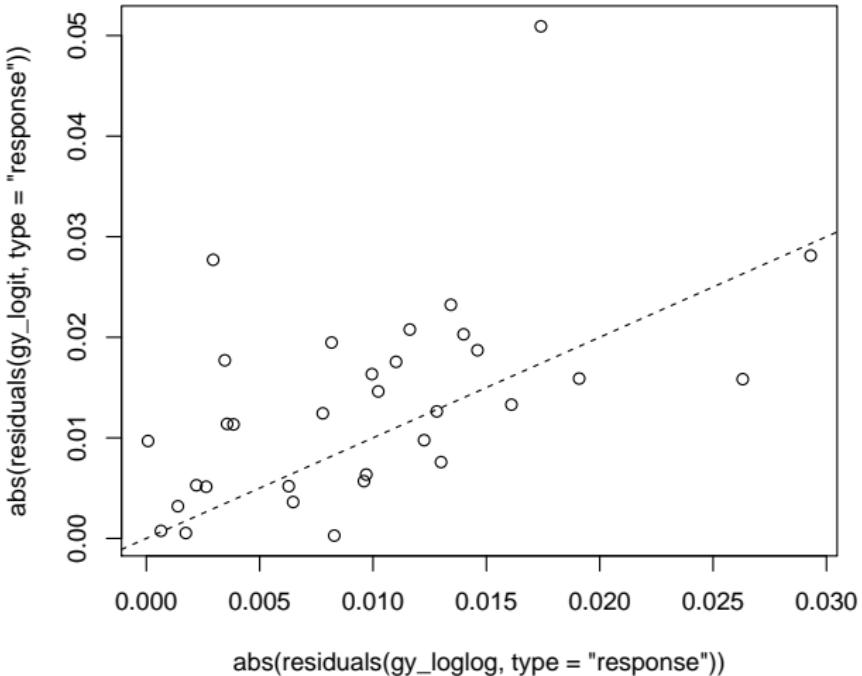


Figure 4: Scatterplot comparing the absolute raw residuals from beta regression modes with log-log link (x-axis) and logit link (y-axis).

Especial topics

- ▶ [Simas et al., 2010] further extend the beta regression model by allowing nonlinear predictors.
- ▶ Analytical bias corrections for the ML estimators of the parameters and generalizing is discussed by [Ospina et al., 2006] and [Simas et al., 2010].
- ▶ Diagnostics for the beta regression models is discussed by [Espinheira et al., 2008b], [Espinheira et al., 2008a], [Ferrari. et al., 2011], and [Rocha and Simas, 2010] among others.

- ▶ A general class of regression models for continuous proportions when the data contain zeros or ones is proposed by [Ospina and Ferrari, 2011].
- ▶ The proposed class of models assumes that the response variable has a mixed continuous-discrete distribution with probability mass at zero or one. The beta distribution is used to describe the continuous component of the model.
- ▶ A suitable parameterization of the beta law in terms of its mean and a precision parameter is used [Ospina and Ferrari, 2010].
- ▶ The parameters of the mixture distribution are modeled as functions of regression parameters.
- ▶ The authors provide inference, diagnostic, and model selection tools for this class of models.

Bibliography I

-  Cribari-Neto, F. and Lima, L. B. (2007).
A misspecification test for beta regressions.
Technical report.
-  Espinheira, P. L., Ferrari, S. L. P., and Cribari-Neto, F. (2008a).
Influence diagnostics in beta regression.
Computational Statistics & Data Analysis, 52(9):4417–4431.
-  Espinheira, P. L., Ferrari, S. L. P., and Cribari-Neto, F. (2008b).
On beta regression residuals.
Journal of Applied Statistics, 35(4):407–419.
-  Ferrari, S. L. P. and Cribari-Neto, F. (2004).
Beta regression for modelling rates and proportions.
Journal of Applied Statistics, 31(7):799–815.
-  Ferrari., S. L. P., Espinheira, P. L., and Cribari-Neto, F. (2011).
Diagnostic tools in beta regression with varying dispersion.
Statistica Neerlandica, pages no–no.
-  McCullagh, P. and Nelder, J. A. (1989).
Generalized Linear Models.
Chapman & Hall, London, 2nd edition.

Bibliography II

-  Ospina, R., Cribari-Neto, F., and Vasconcellos, K. L. P. (2006). Improved point and interval estimation for a beta regression model. *Computational Statistics & Data Analysis*, 51(2):960–981.
-  Ospina, R. and Ferrari, S. (2010). Inflated beta distributions. *Statistical Papers*, 51:111–126.
10.1007/s00362-008-0125-4.
-  Ospina, R. and Ferrari, S. L. P. (2011). A general class of zero-or-one inflated beta regression models. submitted.
-  Prater, N. H. (1956). Estimate gasoline yields from crudes. *Petroleum Refiner*, 35(5):236–238.
-  Ramsey, J. B. (1969). Tests for specification error in classical linear least squares regression analysis. *Journal of the Royal Statistical Society B*, 31:350–371.
-  Rocha, A. V. and Simas, A. B. (2010). Influence diagnostics in a general class of beta regression models. *Test*.

Bibliography III

-  Simas, A. B., Barreto-Souza, W., and Rocha, A. V. (2010). Improved estimators for a general class of beta regression models. *Computational Statistics & Data Analysis*, 54(2):348–366.
-  Vasconcellos, K. L. P. and Cribari-Neto, F. (2005). Improved maximum likelihood estimation in a new class of beta regression models. *Brazilian Journal of Probability and Statistics*, 19(1):13–31.
-  Wei, B.-C., Hu, Y.-Q., and Fung, W.-K. (1998). Generalized leverage and its applications. *Scandinavian Journal of Statistics*, 25(1):25–37.
-  Zeileis, A. (2006). Object-oriented computation of sandwich estimators. *Journal of Statistical Software*, 16(9):1–16.

Thanks !!!

Lecture 8: Gamma regression

Claudia Czado

TU München



Overview

- Models with constant coefficient of variation
- Gamma regression: estimation and testing
- Gamma regression with weights

Motivation

Linear models: $Var(Y_i) = \sigma^2$ constant

Poisson models: $Var(Y_i) = E(Y_i) = \mu_i$ non constant, linear

**Constant coefficient
of variation:**

$$\frac{[Var(Y_i)]^{1/2}}{E(Y_i)} = \sigma$$

$$\Rightarrow Var(Y_i) = \sigma^2 \mu_i^2 \quad \text{non constant, quadratic}$$

Which transformation stabilizes the variance if $Var(Y_i) = \sigma^2 \mu_i^2$ holds?

Answer: $Z_i := \log(Y_i)$

Proof: According to a 2nd order Taylor expansion we have

$$Z_i = \log(Y_i) \approx \log(\mu_i) + \frac{1}{\mu_i}(Y_i - \mu_i) - \frac{1}{2\mu_i^2}(Y_i - \mu_i)^2$$

for σ^2 small

$$\Rightarrow E(Z_i) \approx \log(\mu_i) - \frac{1}{2\mu_i^2} \underbrace{Var(Y_i)}_{=\sigma^2 \mu_i^2} = \log(\mu_i) - \frac{1}{2}\sigma^2$$

According to a 1st order Taylor expansion we have

$$E(Z_i) \approx \log(\mu_i)$$

$$E(Z_i^2) \approx E \left([\log(\mu_i) + \frac{1}{\mu_i}(Y_i - \mu_i)]^2 \right)$$

$$= (\log \mu_i)^2 + 2E \left(\log(\mu_i) \cdot \frac{1}{\mu_i}(Y_i - \mu_i) \right) + E \left(\frac{1}{\mu_i^2}(Y_i - \mu_i)^2 \right)$$

$$= (\log \mu_i)^2 + 0 + \frac{1}{\mu_i^2}\sigma^2 \mu_i^2 = (\log \mu_i)^2 + \sigma^2$$

$$\Rightarrow Var(Z_i) \approx (\log \mu_i)^2 + \sigma^2 - (\log \mu_i)^2 = \sigma^2$$

For small σ^2 we have $E(Z_i) \approx \log(\mu_i) - \sigma^2/2$ and $Var(Z_i) \approx \sigma^2$

Multiplicative error structure: Log normal model

Assume $Y_i = \underbrace{e^{\beta_0} x_{i1}^{\beta_1} \cdots x_{ip}^{\beta_p}}_{E(Y_i) =: \mu_i} (1 + \epsilon_i)$

where $\epsilon_i := \frac{Y_i - E(Y_i)}{E(Y_i)} \Rightarrow E(\epsilon_i) = 0, \text{Var}(\epsilon_i) = \frac{\text{Var}(Y_i)}{E(Y_i)^2} =: \sigma^2$

$$\Rightarrow \ln(Y_i) = \beta_0 + \beta_1 \ln x_{i1} + \dots + \beta_p \ln x_{ip} + \ln(1 + \epsilon_i)$$

where $E(\ln(1 + \epsilon_i)) = E\left(\ln\left(1 + \frac{Y_i - E(Y_i)}{E(Y_i)}\right)\right) = E\left(\ln\left(\frac{Y_i}{E(Y_i)}\right)\right) = \underbrace{E(\ln(Y_i))}_{\approx \ln \mu_i} - \ln E(Y_i) = -\sigma^2/2$

and $\text{Var}(\ln(1 + \epsilon_i)) = \text{Var}(\ln(Y_i) - \ln E(Y_i)) = \text{Var}(\ln(Y_i)) \approx \sigma^2$

Consider

$$\ln Y_i = \beta_0 + \beta_1 \ln x_{i1} + \dots + \beta_p \ln x_{ip} - \frac{\sigma^2}{2} + \epsilon'_i,$$

where $\epsilon'_i := \ln(1 + \epsilon_i) + \frac{\sigma^2}{2} \Rightarrow E(\epsilon'_i) \approx 0, \text{Var}(\epsilon'_i) \approx \sigma^2$

If one assumes $\epsilon'_i \sim N(0, \sigma^2)$ i.i.d. and fits the log normal model

$$\ln Y_i = \alpha_0 + \alpha_1 \ln x_{i1} + \dots + \alpha_p \ln x_{ip} + \epsilon'_i$$

then $\hat{\alpha}_1, \dots, \hat{\alpha}_p$ are consistent estimates for β_1, \dots, β_p , but $\hat{\alpha}_0$ is biased for β_0 with approx. bias $-\sigma^2/2$.

Original scale: Gamma regression

If one wants to work on original scale

$$\mu_i = E(Y_i) = e^{\beta_0} x_{i1}^{\beta_1} \cdots x_{ip}^{\beta_p} = e^{\underbrace{(\ln \mathbf{x}_i)^t \boldsymbol{\beta}}_{\eta_i}}$$

$$\ln \mathbf{x}_i := (1, \ln(x_{i1}, \dots, \ln(x_{ip})))^t$$

$$\Rightarrow g(\mu_i) = \ln(\mu_i).$$

We need distribution for $Y_i \geq 0$ with $E(Y_i) = \mu_i$ and $Var(Y_i) = \sigma^2 \mu_i^2$. Such a distribution is the **Gamma distribution**, i.e.

$$Y \sim \text{Gamma}(\nu, \lambda) \quad \text{with} \quad f_Y(y) = \frac{\lambda}{\Gamma(\nu)} (\lambda y)^{\nu-1} e^{-\lambda y} \quad y \geq 0$$

$$\Rightarrow E(Y) = \frac{\nu}{\lambda} = \mu, \quad Var(Y) = \frac{\nu}{\lambda^2} = \left(\frac{\nu}{\lambda}\right)^2 \frac{1}{\nu} = \mu^2 \underbrace{\frac{1}{\nu}}_{=\sigma^2}.$$

Summary

$$Var(Y_i) = \sigma^2(E(Y_i))^2$$

$$\mu_i = E(Y_i) = e^{\beta_0} x_{i1}^{\beta_1} \cdots x_{ip}^{\beta_p}$$

Log normal model
(normal error on log scale)
Linear model on log scale

GLM with $Y_i \sim \Gamma(\nu, \lambda_i)$
 $\mu_i = \frac{\nu}{\lambda_i}$ $\sigma^2 = \frac{1}{\nu}$
Gamma regression

Properties of the Gamma distribution

1) Mean parametrization

$$\mu = \frac{\nu}{\lambda} \Rightarrow \lambda = \frac{\nu}{\mu}$$

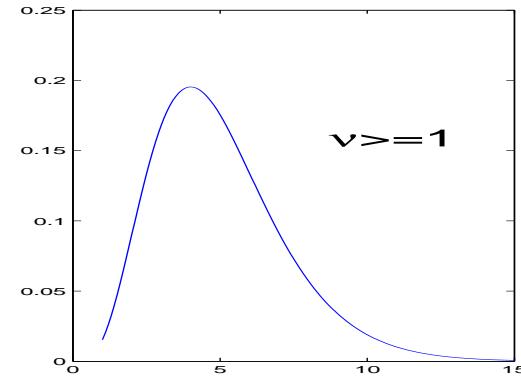
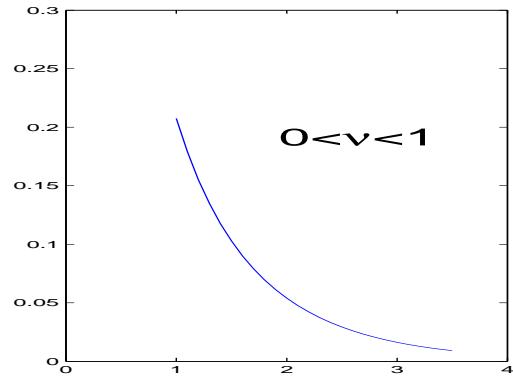
$$\begin{aligned}\Rightarrow f_Y(y) &= \frac{\lambda}{\Gamma(\nu)} (\lambda y)^{\nu-1} e^{-\lambda y} \\ &= \frac{1}{\Gamma(\nu)} \left(\frac{\nu}{\mu}\right) \left(\frac{\nu}{\mu}y\right)^{\nu-1} e^{-\frac{\nu}{\mu}y} \\ &= \frac{1}{\Gamma(\nu)} \left(\frac{\nu y}{\mu}\right)^\nu e^{-\frac{\nu}{\mu}y} \frac{1}{y}\end{aligned}$$

$$\Rightarrow f_Y(y)dy = \frac{1}{\Gamma(\nu)} \left(\frac{\nu y}{\mu}\right)^\nu e^{-\frac{\nu}{\mu}y} \underbrace{\frac{1}{y} dy}_{d(\ln y)}$$

Mean reparametrization

Gamma(μ, ν)

2) The form of the density is determined by ν



Skewed to the right

$$\begin{aligned}\text{Skewness} &= \frac{E(Y-\mu)^3}{\sigma^3} \\ &= 2\nu^{-1/2} \rightarrow 0 \text{ for } \nu \rightarrow \infty\end{aligned}$$

3) Special cases :

- $\nu = 1$ exponential distribution
- $\nu \rightarrow \infty$ normal distribution

Gamma regression as GLM

For known ν in $Y \sim Gamma(\mu, \nu)$ the log likelihood is given by

$$\begin{aligned} l(\mu, \nu, y) &= (\nu - 1) \log(y) - \frac{\nu}{\mu}y + \nu \log \nu - \nu \log \mu - \log \Gamma(\nu) \\ &= \nu \left(-\frac{y}{\mu} - \log \mu \right) + c(y, \nu) \\ \Rightarrow \theta &= -1/\mu, \quad b(\theta) = \log \mu = \log(-1/\theta) = -\log(-\theta) \\ \Rightarrow b'(\theta) &= -\frac{(-1)}{-\theta} = -1/\theta = \mu \quad \text{expectation} \\ b''(\theta) &= \frac{1}{\theta^2} = \mu^2 \quad \text{variance function} \\ a(\phi) &= \phi, \quad \phi = 1/\nu \quad \text{dispersion parameter} \end{aligned}$$

EDA for Gamma regression

$$\mu_i = e^{\mathbf{x}_i^t \boldsymbol{\beta}} \Rightarrow \log(\mu_i) = \mathbf{x}_i^t \boldsymbol{\beta}$$

\Rightarrow a plot of x_{ij} versus $\log(Y_i)$ should be linear

$$\mu_i = \frac{1}{\mathbf{x}_i^t \boldsymbol{\beta}} \Rightarrow \frac{1}{\mu_i} = \mathbf{x}_i^t \boldsymbol{\beta}$$

\Rightarrow a plot of x_{ij} versus $1/Y_i$ should be linear

Example: Canadian Automobile Insurance Claims

Source: Bailey, R.A. and Simon, LeRoy J. (1960). Two studies in automobile insurance. ASTIN Bulletin, 192-217.

Description: The data give the **Canadian automobile insurance experience** for policy years **1956 and 1957** as of June 30, 1959. The data includes virtually every insurance company operating in Canada and was collated by the Statistical Agency (Canadian Underwriters' Association - Statistical Department) acting under instructions from the Superintendent of Insurance. The data given here is for **private passenger automobile liability for non-farmers for all of Canada excluding Saskatchewan**.

The variable **Merit** measures the number of years since the last claim on the policy. The variable **Class** is a **collation of age, sex, use and marital status**. The variables **Insured** and **Premium** are two measures of the **risk exposure** of the insurance companies.

Variable Description:

Merit

- 3 licensed and accident free ≥ 3 years
- 2 licensed and accident free 2 years
- 1 licensed and accident free 1 year
- 0 all others

Class

- 1 pleasure, no male operator < 25
- 2 pleasure, non-principal male
operator < 25
- 3 business use
- 4 unmarried owner or principal
operator < 25
- 5 married owner or principal
operator < 25

Variable Description(continued):

Insured	Earned car years
Premium	Earned premium in 1000's (adjusted to what the premium would have been had all cars been written at 01 rates)
Claims	Number of claims
Cost	Total cost of the claim in 1000's of dollars

Variable of Interest is **Cost**.

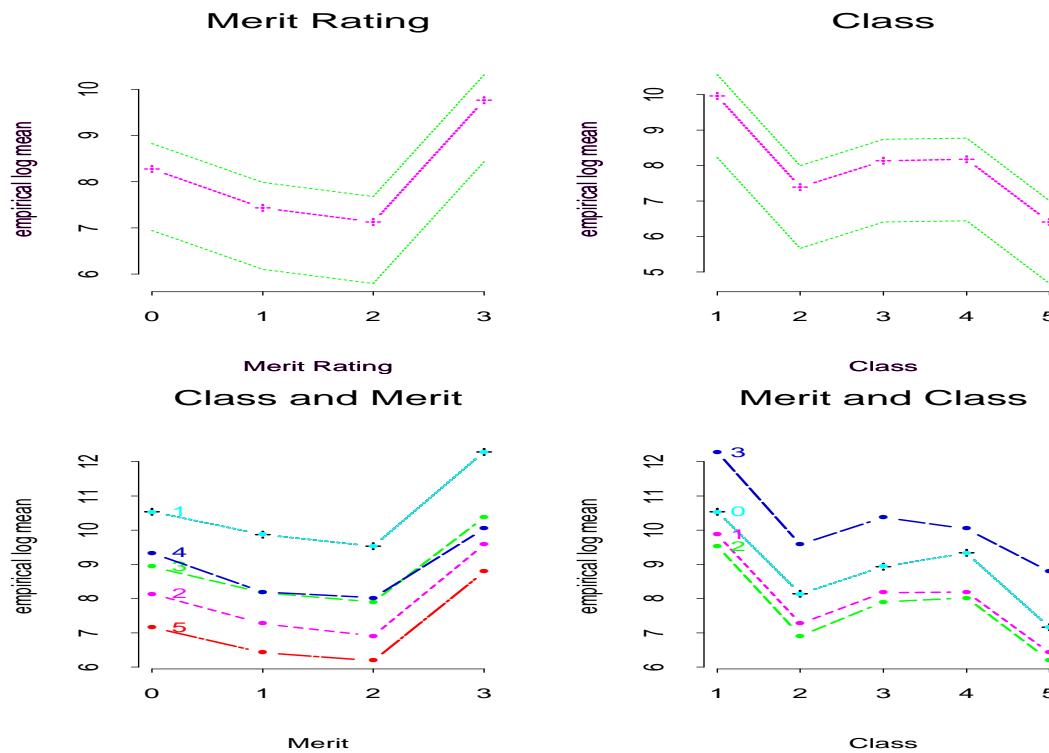
Data:

```
> cancar.data
```

	Merit	Class	Insured	Premium	Claims	Cost
1	3	1	2757520	159108	217151	63191
2	3	2	130535	7175	14506	4598
3	3	3	247424	15663	31964	9589
4	3	4	156871	7694	22884	7964
5	3	5	64130	3241	6560	1752
6	2	1	130706	7910	13792	4055
7	2	2	7233	431	1001	380
8	2	3	15868	1080	2695	701
9	2	4	17707	888	3054	983
10	2	5	4039	209	487	114
11	1	1	163544	9862	19346	5552
12	1	2	9726	572	1430	439
13	1	3	20369	1382	3546	1011
14	1	4	21089	1052	3618	1281
15	1	5	4869	250	613	178
16	0	1	273944	17226	37730	11809
17	0	2	21504	1207	3421	1088
18	0	3	37666	2502	7565	2383
19	0	4	56730	2756	11345	3971
20	0	5	8601	461	1291	382

Exploratory Data Analysis (unweighted case):

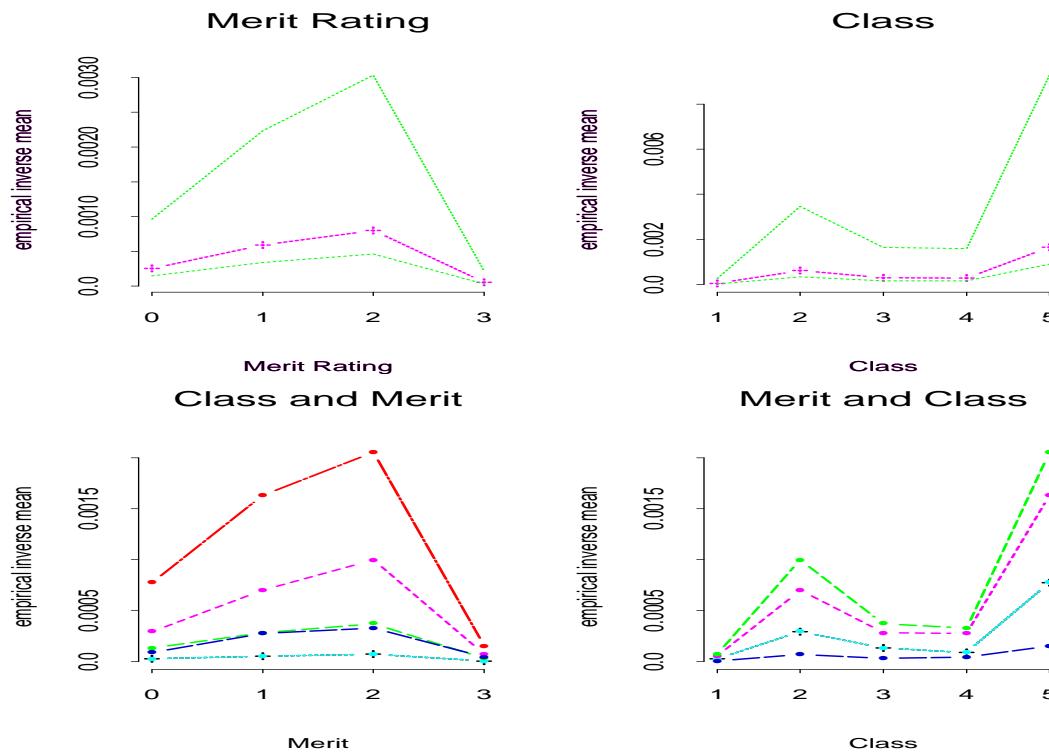
Log Link:



No obvious functional relationships, treatment of covariates as factors might be appropriate. No strong interactions expected.

Exploratory Data Analysis (continued):

Inverse Link:



No obvious functional relationships, treatment of covariates as factors might be appropriate. Some interactions expected.

Residual deviance in Gamma regression

$$Y_i \sim \text{Gamma}(\mu_i, \nu) \quad \text{independent} \quad \mu_i = e^{x_i^t \beta}$$

$$\begin{aligned}
 l(\boldsymbol{\mu}, \nu, \mathbf{y}) &= \sum_{i=1}^n \left\{ \nu \left[-\frac{y_i}{\mu_i} - \ln(\mu_i) \right] - \ln(\Gamma(\nu)) + \nu \ln(\nu y_i) - \ln(y_i) \right\} \\
 \Rightarrow l(\mathbf{y}, \nu, \mathbf{y}) &= \sum_{i=1}^n \left\{ \nu [-1 - \ln(y_i)] - \ln(\Gamma(\nu)) + \nu \ln(\nu y_i) - \ln(y_i) \right\} \\
 &\qquad\qquad\qquad - \text{saturated log likelihood} \\
 \Rightarrow -2(l(\hat{\boldsymbol{\mu}}, \nu, \mathbf{y}) - l(\mathbf{y}, \nu, \mathbf{y})) &= 2 \left[\sum_{i=1}^n \nu \left(-1 - \ln(y_i) + \frac{y_i}{\hat{\mu}_i} + \ln(\hat{\mu}_i) \right) \right] \\
 &= -2 \sum_{i=1}^n \nu \left[\ln \left(\frac{y_i}{\hat{\mu}_i} \right) - \frac{y_i - \hat{\mu}_i}{\hat{\mu}_i} \right] \\
 \Rightarrow D(\mathbf{y}, \hat{\boldsymbol{\mu}}) &= -2 \sum_{i=1}^n \left[\ln \left(\frac{y_i}{\hat{\mu}_i} \right) - \frac{y_i - \hat{\mu}_i}{\hat{\mu}_i} \right] \quad \text{deviance}
 \end{aligned}$$

We have $D(\mathbf{y}, \hat{\boldsymbol{\mu}}) \stackrel{a}{\sim} \phi \chi_{n-p}^2$ where $\phi = 1/\nu$.

Residual deviance test

$$Y_i \sim \text{Gamma}(\mu_i, \nu) \quad \text{independent} \quad \mu_i = e^{\mathbf{x}_i^t \boldsymbol{\beta}} \quad (\text{Model } (*))$$

Reject model (*) at level α if $\frac{D(\mathbf{y}, \hat{\boldsymbol{\mu}})}{\hat{\phi}} > \chi_{n-p, 1-\alpha}^2$

Here $\hat{\phi}$ is an estimate of ϕ .

Partial deviance test

Consider $\beta = (\beta_1^t, \beta_2^t)^t$ $\beta_1 \in \mathbb{R}^{p_1}$, $\beta_2 \in \mathbb{R}^{p_2}$

- $\hat{\mu}_1$ = fitted mean in reduced model with design X_2
- $\hat{\mu}_2$ = fitted mean in full model with design X_1 and X_2
- $\hat{\phi}_1$ = estimated dispersion parameter in reduced model
- $\hat{\phi}_2$ = estimated dispersion parameter in full model

The partial deviance test for $H_0 : \beta_1 = \mathbf{0}$ versus $H_1 : \beta_1 \neq \mathbf{0}$ is given by:

Reject H_0 at level $\alpha \Leftrightarrow$

$$\frac{D(\mathbf{y}, \hat{\mu}_1)}{\hat{\phi}_1} - \frac{D(\mathbf{y}, \hat{\mu}_2)}{\hat{\phi}_2} > \chi_{p_1, 1-\alpha}^2$$

Canonical link

$$\eta_i = \theta_i = -\frac{1}{\mu_i}$$

Since we need $\mu_i > 0$ we need $\eta_i < 0$, which gives restrictions on β . Therefore the canonical link is not often used. Most often the **log link** is used.

Estimation of the dispersion parameter

For the estimation of the dispersion parameter $\phi = \sigma^2 = 1/\nu$ **method of moments** is used

$$E(Y_i) = \mu_i \quad Var(Y_i) = \phi\mu_i^2 \quad \forall i$$

$\Rightarrow Z_i = \frac{Y_i}{\mu_i}$ has expectation 1, variance ϕ and is i.i.d.

$$\Rightarrow \hat{\phi} = \frac{1}{n-p} \sum_{i=1}^n \left(\frac{Y_i}{\hat{\mu}_i} - 1 \right)^2 = \frac{1}{n-p} \sum_{i=1}^n \left(\frac{Y_i - \hat{\mu}_i}{\hat{\mu}_i} \right)^2$$

since p parameters need to be estimated.

Remark: Pearson χ^2 statistic for the gamma regression is given by

$$\begin{aligned} \chi_P^2 &= \sum_{i=1}^n \frac{(Y_i - \hat{\mu}_i)^2}{V(\hat{\mu}_i)} \quad V(\hat{\mu}_i) = \hat{\mu}_i^2 \\ &= \sum_{i=1}^n \left(\frac{Y_i - \hat{\mu}_i}{\hat{\mu}_i} \right)^2 \end{aligned}$$

$$\Rightarrow \hat{\phi} = \chi_P^2 / (n - p).$$

Relationship between gamma regression with log link and a linear model on the log scala

For $\sigma^2 = \phi$ small we have $Var(\log(Y_i)) \approx \sigma^2$, therefore we can use

$$\log Y_i = x_i^t \alpha + \epsilon_i \quad \epsilon_i \sim N(0, \sigma^2) \text{ i.i.d.} \quad \text{linear model on log scala}$$

$$\Rightarrow Cov(\hat{\alpha}) = \sigma^2(X^t X)^{-1}.$$

For the gamma regression with $Y_i \sim \Gamma(\mu_i, 1/\sigma^2)$ and $\mu_i = \exp\{x_i^t \beta\}$ we have

$$\hat{\beta} \stackrel{a}{\sim} N(\beta, I(\hat{\beta})^{-1}), \quad \text{where}$$

$$I(\hat{\beta}) = \text{Fisher information} = \left(-E \left(\frac{\partial^2 l(\beta)}{\partial \beta \partial \beta^t} \right) \right) \Big|_{\beta=\hat{\beta}}.$$

Here

$$l(\beta) = \frac{1}{\sigma^2} \sum_{i=1}^n \left(-\frac{y_i}{\mu_i} - \log \mu_i + const \right)$$

$$\Rightarrow \quad \frac{\partial l(\beta)}{\partial \beta_j} = \frac{1}{\sigma^2} \sum_{i=1}^n \left(\frac{y_i x_{ij}}{e^{x_i^t \beta}} - x_{ij} \right) \quad j = 1, \dots, p$$

$$\Rightarrow \quad \frac{\partial^2 l(\beta)}{\partial \beta_s \partial \beta_j} = \frac{1}{\sigma^2} \sum_{i=1}^n \left(-\frac{y_i x_{is} x_{ij}}{e^{x_i^t \beta}} \right)$$

$$\Rightarrow \quad -E \left(\frac{\partial^2 l(\beta)}{\partial \beta_s \partial \beta_j} \right) = \frac{1}{\sigma^2} \sum_{i=1}^n x_{is} x_{ij} (\text{independent of } \beta)$$

$$\Rightarrow \quad I(\hat{\beta}) = \frac{1}{\sigma^2} X^t X$$

$$\Rightarrow \quad Cov(\hat{\beta}) \approx \sigma^2 (X^t X)^{-1} = Cov(\hat{\alpha})$$

Distinction between both models is difficult. If X corresponds to an orthogonal design, i.e. $(X^t X)^{-1} = I_p$, we have that the components of $\hat{\alpha}$ ($\hat{\beta}$) are (asymptotically) independent.

Gamma regression with weights

Example: Y_i = total claim size arising from n_i claims in group i with covariates \mathbf{x}_i .

$$\Rightarrow Y_i = \sum_{j=1}^{n_i} Y_{ij}; \quad Y_{ij} = j^{\text{th}} \text{ claim in group } i$$

$Y_i^s := Y_i/n_i =$ average claim size in group i

If $Y_{ij} \sim \text{Gamma}(\mu_i, \nu)$, independent, we have

$$E(Y_i^s) = \frac{1}{n_i} \sum_{j=1}^{n_i} \underbrace{E(Y_{ij})}_{=\mu_i} = \mu_i$$

$$\text{Var}(Y_i^s) = \frac{1}{n_i} \text{Var}(Y_{i1}) = \frac{1}{n_i} \mu_i^2 / \nu = \mu_i^2 / n_i \nu.$$

Since $n_i Y_i^s = \sum_{j=1}^{n_i} Y_{ij} \sim \text{Gamma}(n_i \mu_i, n_i \nu)$

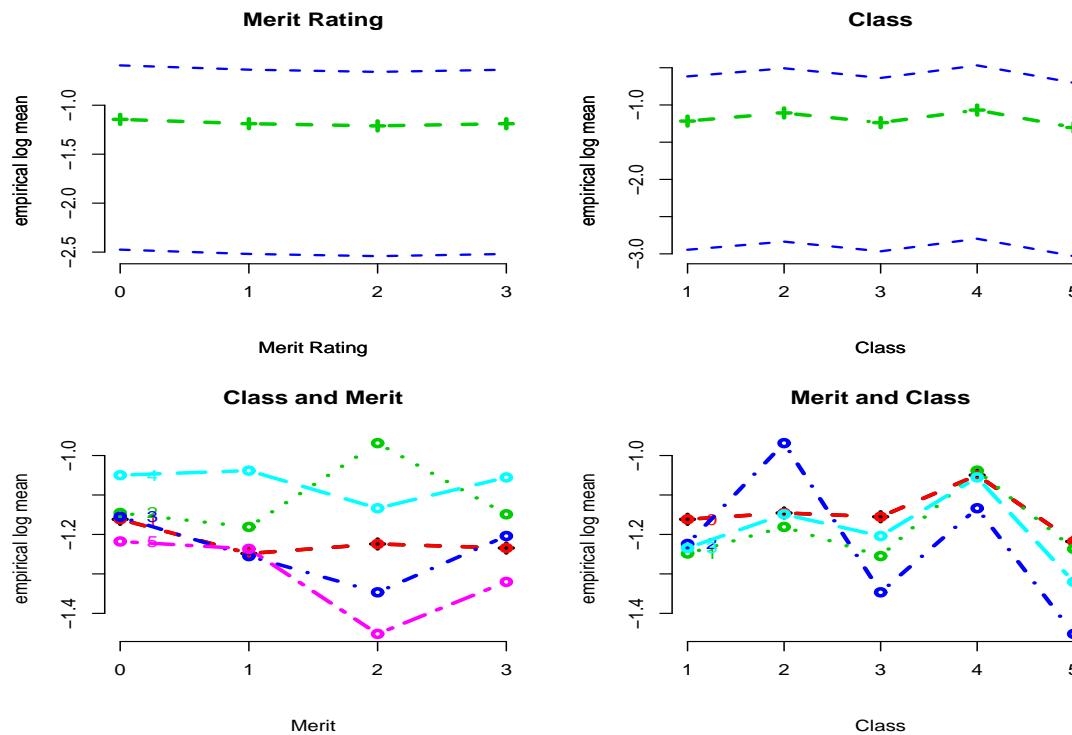
we have $Y_i^s \sim \text{Gamma}(\mu_i, n_i \nu)$. This follows since

$$\begin{aligned} m_{Y_i^s}(t) &:= E(e^{tY_i^s}) = E(e^{\frac{tn_i Y_i^s}{n_i}}) = \frac{1}{(1 - \frac{n_i \mu_i t}{n_i \nu})^{n_i \nu}} \\ &= \frac{1}{(1 - \frac{\mu_i t}{n_i \nu})^{n_i \nu}} = m_X(t) \quad \forall t, \quad \text{where } X \sim \text{Gamma}(\mu_i, n_i \nu). \end{aligned}$$

Therefore we need to introduce weights for a gamma regression for Y_i^s . Since $E(Y_i^s) = \mu_i$, the EDA for a weighted gamma regression does not change compared to an unweighted one.

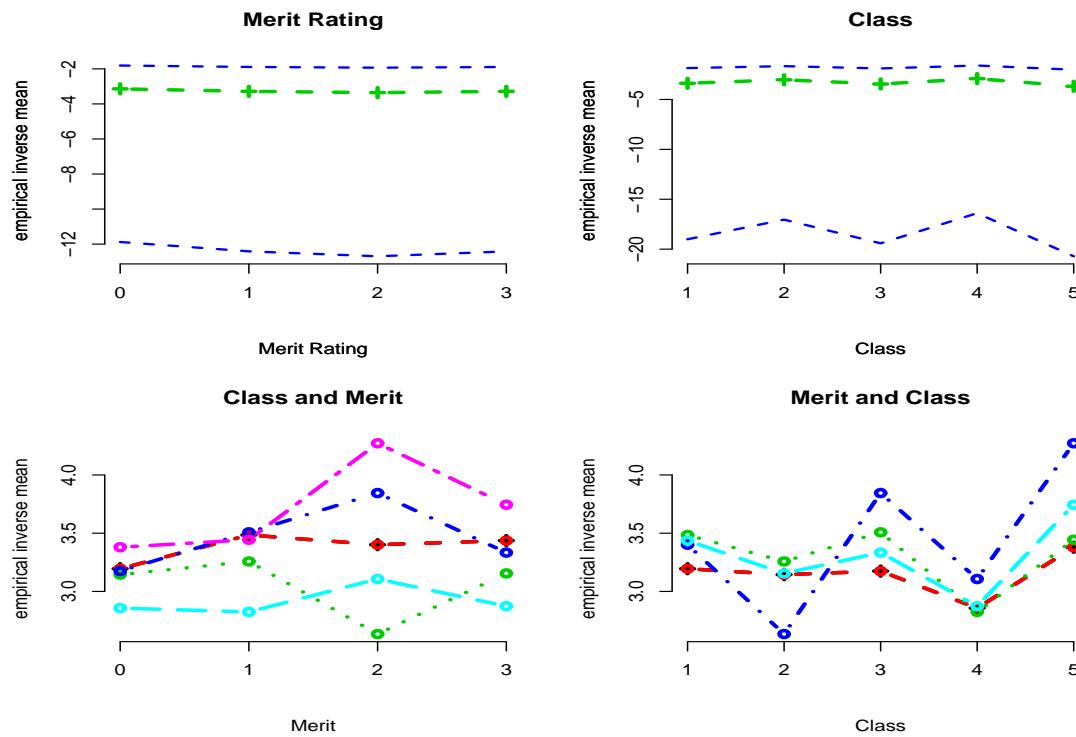
Exploratory Data Analysis (weighted case):

Log Link:



Exploratory Data Analysis (weighted case):

Inverse Link:



For estimation of the dispersion parameter $\sigma^2 = \phi = 1/\nu$ in a weighted model we use

$$\hat{\sigma}^2 = \hat{\phi} = \frac{1}{n-p} \sum_{i=1}^n n_i \left(\frac{Y_i - \hat{\mu}_i}{\hat{\mu}_i} \right)^2$$

Residuals: $E(Y_i^s) = \mu_i$ $Var(Y_i^s) = \frac{\mu_i^2}{n_i \nu} = \mu_i^2 \frac{\sigma^2}{n_i}$

$$\Rightarrow \frac{Y_i^s - \mu_i}{\frac{\sigma}{\sqrt{n_i}} \mu_i} = \frac{\sqrt{n_i}}{\sigma} \left(\frac{Y_i^s - \mu_i}{\mu_i} \right) \quad \text{has zero expectation and unit variance}$$

i.e. **standardized residuals** can be defined by

$$r_i^s := \frac{\sqrt{n_i}}{\sigma} \left(\frac{Y_i^s - \hat{\mu}_i}{\hat{\mu}_i} \right).$$

Example: Canadian Automobile Insurance Claims: Gamma Regression:

Main Effects with Log Link

```
> f.gamma.main_Cost/Claims ~ Merit + Class  
> r.gamma.log.main_glm(f.gamma.main, family =  
Gamma(link = "log"), weights = Claims)
```

Example: Gamma Regression:

```
> summary(r.gamma.log.main,cor=F)
Call: glm(formula = Cost/Claims ~ Merit + Class,
family= Gamma(link = "log"), weights = Claims)
```

Coefficients:

	Value	Std. Error	t value
(Intercept)	-1.175	0.016	-75.58
Merit1	-0.069	0.026	-2.63
Merit2	-0.070	0.029	-2.41
Merit3	-0.057	0.016	-3.48
Class2	0.083	0.026	3.13
Class3	0.016	0.018	0.86
Class4	0.160	0.019	8.23
Class5	-0.081	0.039	-2.08

(Dispersion Parameter for Gamma family
taken to be 13)

Null Deviance:1556 on 19 degrees of freedom
Residual Deviance:157 on 12 degrees of freedom

Example: Gamma Regression (continued):

To check the dispersion estimate, use the Pearson Chi Square Statistics.

```
> sum(resid(r.gamma.log.main,type="pearson"))^2  
[1] 159  
> 159/12  
[1] 13
```

For the residual deviance test, we need to scale the deviance

```
> 1-pchisq(156/13,12)  
[1] 0.45
```

A p-value of .45 shows no lack of fit.

Main Effects with Inverse Link

```
> f.gamma.main_Cost/Claims ~ Merit + Class  
> r.gamma.inverse.main_glm(f.gamma.main,  
    family = Gamma, weights = Claims)  
> summary(r.gamma.inverse.main,cor=F)
```

Coefficients:

	Value	Std. Error	t value
(Intercept)	3.247	0.051	63.72
Merit1	0.215	0.089	2.42
Merit2	0.224	0.099	2.25
Merit3	0.177	0.053	3.32
Class2	-0.268	0.086	-3.12
Class3	-0.054	0.063	-0.85
Class4	-0.498	0.059	-8.39
Class5	0.287	0.149	1.93

(Dispersion Parameter for Gamma family taken to be 14)

Null Deviance:1556 on 19 degrees of freedom

Residual Deviance:167 on 12 degrees of freedom

Example: Gamma Regression (continued):

For the residual deviance test, we need to scale the deviance

```
> 1-pchisq(167/14,12)
[1] 0.45
```

A p-value of .45 shows no lack of fit.

Log Link With Interaction

```
> f.gamma.inter_Cost/Claims ~ Merit * Class  
> r.gamma.inter_glm(f.gamma.inter, family  
= Gamma, weights = Claims))
```

Example: Gamma Regression (continued):

```
> summary(r.gamma.log.inter,cor=F)
```

Coefficients:

	Value	Std. Error	t value
(Intercept)	3.195	NA	NA
Merit1	0.289	NA	NA
Merit2	0.206	NA	NA
Merit3	0.241	NA	NA
Class2	-0.051	NA	NA
Class3	-0.020	NA	NA
Class4	-0.338	NA	NA
Class5	0.185	NA	NA
Merit1Class2	-0.176	NA	NA
Merit2Class2	-0.716	NA	NA
Merit3Class2	-0.231	NA	NA
Merit1Class3	0.043	NA	NA
Merit2Class3	0.464	NA	NA
Merit3Class3	-0.083	NA	NA
Merit1Class4	-0.322	NA	NA
Merit2Class4	0.044	NA	NA
Merit3Class4	-0.225	NA	NA
Merit1Class5	-0.225	NA	NA
Merit2Class5	0.686	NA	NA
Merit3Class5	0.123	NA	NA

(Dispersion Parameter for Gamma family taken
to be NA)

Example: Gamma Regression (continued):

Null Deviance: 1556 on 19 degrees of freedom

Residual Deviance: 0 on 0 degrees of freedom

Number of Fisher Scoring Iterations: 1

This model is the **saturated** model since only one observation per cell. Therefore the **dispersion estimate can no longer be estimated**, since the Pearson Chi Square Statistics is 0. To fit a model with **fixed dispersion**, use

```
> summary(r.gamma.log.inter,dispersion=1,cor=F)
```

Example: Gamma Regression (continued):

Coefficients:

	Value	Std. Error	t value
(Intercept)	3.195	0.016	194.24
Merit1	0.289	0.030	9.66
Merit2	0.206	0.033	6.19
Merit3	0.241	0.018	13.39
Class2	-0.051	0.056	-0.90
Class3	-0.020	0.040	-0.51
Class4	-0.338	0.031	-10.74
Class5	0.185	0.095	1.93
Merit1Class2	-0.176	0.106	-1.67
Merit2Class2	-0.716	0.105	-6.85
Merit3Class2	-0.231	0.062	-3.70
Merit1Class3	0.043	0.075	0.57
Merit2Class3	0.464	0.089	5.21
Merit3Class3	-0.083	0.045	-1.84
Merit1Class4	-0.322	0.062	-5.21
Merit2Class4	0.044	0.071	0.62
Merit3Class4	-0.225	0.037	-6.00
Merit1Class5	-0.225	0.171	-1.32
Merit2Class5	0.686	0.218	3.15
Merit3Class5	0.123	0.106	1.16

(Dispersion Parameter for Gamma family taken to be 1)

Null Deviance: 1556 on 19 degrees of freedom

Residual Deviance: 0 on 0 degrees of freedom

From this we see that certain interaction effects are strongly significant.

Log Link With Some Interaction Terms

```
> f.gamma.inter3_Cost/Claims ~ Merit + Class +
  I((Merit == 2) * (
  Class == 2)) + I((Merit == 2) * (Class ==
  3)) + I((Merit == 1) * (Class == 4)) + I(((
  Merit == 3) * (Class == 4)) + I((Merit ==
  3) * (Class == 2)) + I((Merit == 2) * (
  Class == 5))
> r.gamma.log.inter3_glm(f.gamma.inter3,family
=Gamma(link="log"),weights=Claims)
```

Example: Gamma Regression (continued):

```
> summary(r.gamma.log.inter3,cor=F)
Call:glm(formula=Cost/Claims ~ Merit+Class+I(
  Merit == 2) * (Class == 2)) + I((Merit ==
  2) * (Class == 3)) + I((Merit == 1) * (
  Class == 4)) + I((Merit == 3) * (Class ==
  4)) + I((Merit == 3) * (Class == 2)) + I((
  Merit == 2) * (Class == 5)), family =
  Gamma(link = "log"), weights = Claims)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-1.8	-0.43	-2.4e-07	0.44	2.2

Example: Gamma Regression (continued):

Coefficients:

	Value	Std. Error
(Intercept)	-1.164	0.0076
Merit1	-0.084	0.0127
Merit2	-0.064	0.0145
Merit3	-0.070	0.0081
Class2	0.033	0.0245
Class3	0.023	0.0086
Class4	0.110	0.0153
Class5	-0.074	0.0182
I((Merit==2)*(Class==2))	0.226	0.0590
I((Merit==2)*(Class==3))	-0.142	0.0354
I((Merit==1)*(Class==4))	0.100	0.0331
I((Merit==3)*(Class==4))	0.068	0.0191
I((Merit==3)*(Class==2))	0.052	0.0283
I((Merit==2)*(Class==5))	-0.150	0.0781

Example: Gamma Regression (continued):

	t value
(Intercept)	-154.1
Merit1	-6.6
Merit2	-4.4
Merit3	-8.6
Class2	1.3
Class3	2.7
Class4	7.2
Class5	-4.1
I((Merit == 2) * (Class == 2))	3.8
I((Merit == 2) * (Class == 3))	-4.0
I((Merit == 1) * (Class == 4))	3.0
I((Merit == 3) * (Class == 4))	3.6
I((Merit == 3) * (Class == 2))	1.8
I((Merit == 2) * (Class == 5))	-1.9

(Dispersion Parameter for Gamma family taken to be 2.7)

Null Deviance: 1556 on 19 degrees of freedom

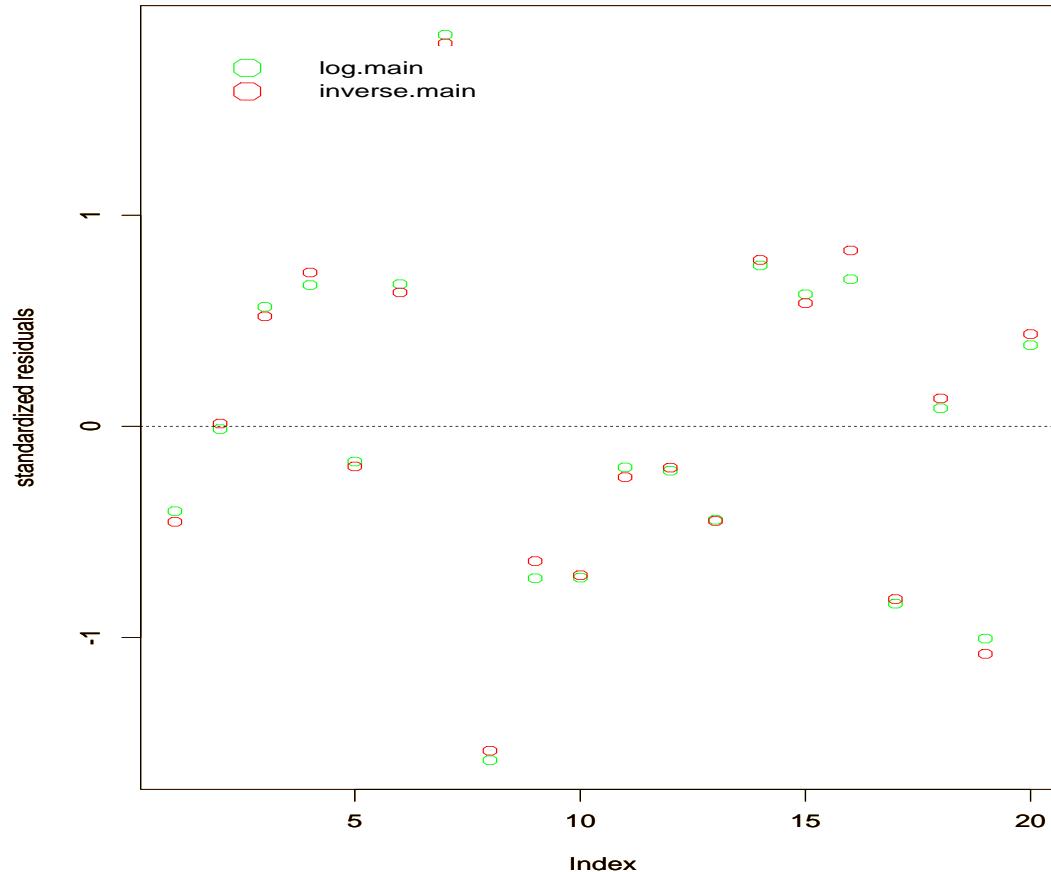
Residual Deviance: 16 on 6 degrees of freedom

Example: Gamma Regression (continued):

```
> 1-pchisq(16/2.7,6)
[1] 0.43 # Residual Deviance Test
> 1-pchisq((156/13)-(16/2.7),6)
[1] 0.41 # Partial Deviance Test
```

We see that the joint effect of the interaction terms are not significant, since the partial deviance test yields a p-value of .41. A partial deviance test when only $I(Merit == 2) * (Class == 2)$ and $I(Merit == 2) * (Class == 3)$ are included gives a p-value of .39, thus showing that the interaction effects are not present in this data set. The same results are true when an inverse link is used instead.

Standardized Residual Plots::



Difference between log link and inverse link is **minimal**.

What needs to be done if one wants to use the linear model on the log scale?

$$Var(Y_i^s) = \frac{\sigma^2}{n_i} \mu_i^2$$

$$\begin{aligned}\Rightarrow E(\log(Y_i^s)) &\approx E \left(\log(\mu_i) + \frac{1}{\mu_i}(Y_i^s - \mu_i) - \frac{1}{\mu_i^2}(Y_i^s - \mu_i)^2 \right) \\ &= \log(\mu_i) - \frac{1}{\mu_i^2} \underbrace{Var(Y_i^s)}_{\mu_i^2 \frac{\sigma^2}{n_i}} = \log(\mu_i) - \frac{\sigma^2}{2n_i}\end{aligned}$$

$$\begin{aligned}E [(\log(Y_i^s))^2] &\approx E \left([\log(\mu_i) + \frac{1}{\mu_i}(Y_i^s - \mu_i)]^2 \right) \\ &= [\log(\mu_i)]^2 + \frac{Var(Y_i^s)}{\mu_i^2} \\ &= (\log \mu_i)^2 + \frac{\sigma^2}{n_i}\end{aligned}$$

$$\Rightarrow Var(\log(Y_i^s)) \approx (\log \mu_i)^2 + \frac{\sigma^2}{n_i} - (\log \mu_i)^2 = \frac{\sigma^2}{n_i} \quad \text{for } \sigma^2 \text{ small}$$

\Rightarrow linear model on the log scale needs also to be weighted

Log Normal Models:

```
> f.lognormal.main_log(Cost/Claims)~Merit + Class  
> r.lognormal.main_glm(f.lognormal.main,  
weights=Claims)  
> summary(r.lognormal.main,cor=F)
```

Call: `glm(formula = log(Cost/Claims) ~ Merit +
Class, weights = Claims)`

Deviance Residuals:

Min	1Q	Median	3Q	Max
-6	-1.8	-0.24	2.3	6.2

Log Normal Models (continued):

Coefficients:

	Value	Std. Error	t value
(Intercept)	-1.175	0.015	-76.33
Merit1	-0.069	0.026	-2.66
Merit2	-0.072	0.029	-2.51
Merit3	-0.057	0.016	-3.50
Class2	0.082	0.026	3.12
Class3	0.015	0.018	0.85
Class4	0.160	0.019	8.29
Class5	-0.082	0.039	-2.12

(Dispersion Parameter for Gaussian family taken to be 13)

Null Deviance:1515 on 19 degrees of freedom

Residual Deviance:156 on 12 degrees of freedom

Log Normal Models (continued):

```
> 1-pchisq(156/13,12)
[1] 0.45 # Residual Deviance Test
# Results when Splus function lm() is used:
> summary(lm(f.lognormal.main,weights=Claims),cor=F)
Coefficients:
              Value Std. Error t value
(Intercept) -1.175    0.015   -76.327
Merit1       -0.069    0.026    -2.658
Merit2       -0.072    0.029    -2.508
Merit3       -0.057    0.016    -3.501
Class2        0.082    0.026     3.117
Class3        0.015    0.018     0.849
Class4        0.160    0.019     8.292
Class5       -0.082    0.039    -2.124
                  Pr(>|t|)
(Intercept) 0.000
Merit1      0.021
Merit2      0.028
Merit3      0.004
Class2      0.009
Class3      0.412
Class4      0.000
Class5      0.055
```

Log Normal Models (continued):

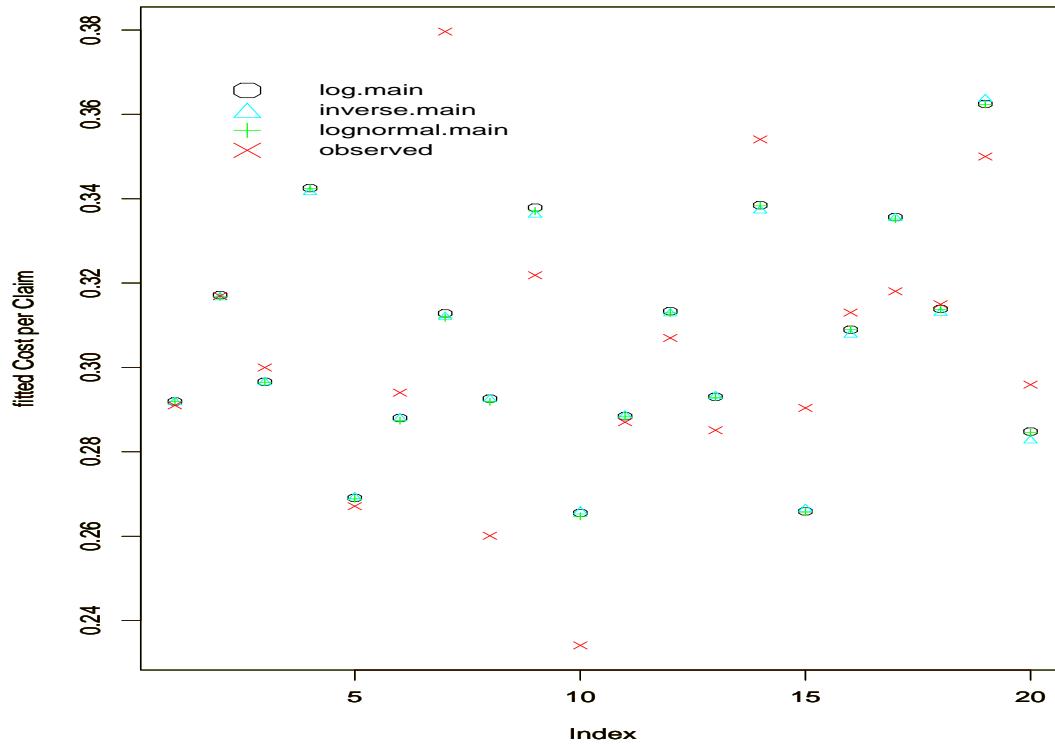
Residual standard error: 3.6 on 12
degrees of freedom

Multiple R-Squared: 0.9

F-statistic: 15 on 7 and 12 degrees of freedom,
the p-value is 4.7e-05

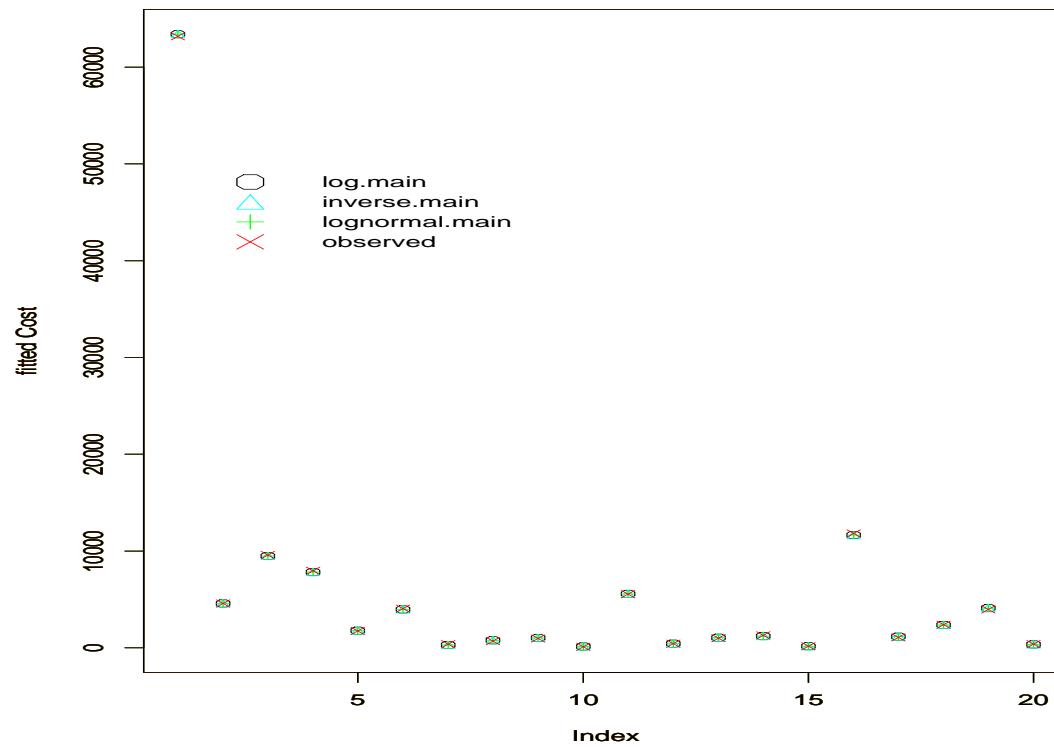
Both approaches (R-squared=.9, residual deviance p-value=.45) show **no lack of fit.**

Fitted Costs per Claims:



Difference between Models are minimal.

Fitted Cost:



Difference between Models are minimal.



Logit/Probit models in R

(work in progress, 2.0)

Oscar Torres-Reyna

otorres@princeton.edu



December 2014

http://dss.princeton.edu/training/
R

If outcome or dependent variable is binary and in the form 0/1, then use logit or probit models.

Some examples are:

Did you vote in the last election?

- 0 'No'
- 1 'Yes'

Do you prefer to use public transportation or to drive a car?

- 0 'Prefer to drive'
- 1 'Prefer public transport'

If outcome or dependent variable is categorical but are ordered (i.e. low to high), then use ordered logit or ordered probit models. Some examples are:

Do you agree or disagree with the President?

- 1 'Disagree'
- 2 'Neutral'
- 3 'Agree'

What is your socioeconomic status?

- 1 'Low'
- 2 'Middle'
- 3 'High'

If outcome or dependent variable is categorical without any particular order, then use multinomial logit. Some examples are:

If elections were held today, for which party would you vote?

- 1 'Democrats'
- 2 'Independent'
- 3 'Republicans'

What do you like to do on the weekends?

- 1 'Rest'
- 2 'Go to movies'
- 3 'Exercise'

Logit model

```
# Getting sample data
```

```
library(foreign)
```

```
mydata <- read.dta("http://dss.princeton.edu/training/Panel101.dta")
```

```
# Running a logit model
```

```
logit <- glm(y_bin ~ x1 + x2 + x3, family=binomial(link="logit"), data=mydata)
```



```
summary(logit)
```

```
Call:  
glm(formula = y_bin ~ x1 + x2 + x3, family = binomial(link = "logit"),  
     data = mydata)
```

```
Deviance Residuals:
```

Min	1Q	Median	3Q	Max
-2.0277	0.2347	0.5542	0.7016	1.0839

```
Coefficients:
```

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	0.4262	0.6390	0.667	0.5048
x1	0.8618	0.7840	1.099	0.2717
x2	0.3665	0.3082	1.189	0.2343
x3	0.7512	0.4548	1.652	0.0986 .

```
Signif. codes:  0 '****' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
(Dispersion parameter for binomial family taken to be 1)
```

```
Null deviance: 70.056  on 69  degrees of freedom  
Residual deviance: 65.512  on 66  degrees of freedom  
AIC: 73.512
```

```
Number of Fisher Scoring iterations: 5
```

The **Pr(>|z|)** column shows the two-tailed p-values testing the null hypothesis that the coefficient is equal to zero (i.e. no significant effect). The usual value is 0.05, by this measure none of the coefficients have a significant effect on the log-odds ratio of the dependent variable. The coefficient for **x3** is significant at 10% (<0.10).

The **z value** also tests the null that the coefficient is equal to zero. For a 5% significance, the z-value should fall outside the ± 1.96 .

The **Estimate** column shows the coefficients in log-odds form. When **x3** increase by one unit, the expected change in the log odds is 0.7512. What you get from this column is whether the effect of the predictors is positive or negative. See next page for an extended explanation.

Logit model: odds ratio

Odds ratio interpretation (OR): Based on the output below, when x_3 increases by one unit, the odds of $y = 1$ increase by 112% $-(2.12-1)*100\%$. Or, the odds of $y = 1$ are 2.12 times higher when x_3 increases by one unit (keeping all other predictors constant). To get the odds ratio, you need exponentiate the logit coefficient.

Estimating the odds ratio by hand

```
cbind(Estimate=round(coef(logit),4),  
      OR=round(exp(coef(logit)),4))
```

	Estimate	OR
(Intercept)	0.4262	1.5314
x1	0.8618	2.3674
x2	0.3665	1.4427
x3	0.7512	2.1196

Using package --mfx--

```
library(mfx)  
logitor(y_bin ~ x1 + x2 + x3, data=mydata)  
Call:  
logitor(formula = y_bin ~ x1 + x2 + x3, data = mydata)
```

Odds Ratio:

	OddsRatio	Std. Err.	z	P> z
x1	2.36735	1.85600	1.0992	0.27168
x2	1.44273	0.44459	1.1894	0.23427
x3	2.11957	0.96405	1.6516	0.09861 .

Signif. codes: 0 '****' 0.001 '***' 0.01 '**' 0.05 '*' 0.1 '.' 1

The **Estimate** column shows the coefficients in log-odds form. When x_3 increase by one unit, the expected change in the log odds is 0.7512. Lets hold x_1 and x_2 constant at their means, and vary x_3 with values 1, 2, and 3, to get the predicted log-odds given each of the three values of x_3 :

```
r1 <- logit$coeff[1] + logit$coeff[2]*mean(mydata$x1) +  
      logit$coeff[3]*mean(mydata$x2) +  
      logit$coeff[4]*1  
> r1  
1.784902
```

```
r2 <- logit$coeff[1] + logit$coeff[2]*mean(mydata$x1) +  
      logit$coeff[3]*mean(mydata$x2) +  
      logit$coeff[4]*2
```

```
> r2  
2.536113
```

```
r3 <- logit$coeff[1] + logit$coeff[2]*mean(mydata$x1) +  
      logit$coeff[3]*mean(mydata$x2) +  
      logit$coeff[4]*3
```

```
> r3  
3.287325
```

When x_3 increases from 1 to 2, the log-odds increases:

```
r2-r1  
0.7512115
```

When x_3 increases from 2 to 3, the log-odds increases:

```
r3-r2  
0.7512115
```

Which corresponds to the estimate for x_3 above.

The odds ratio, is the exponentiation of the difference of the log-odds

```
> exp(r2-r1)  
2.119566
```

Or, the ratio of the exponentiation of each of the log-odds.

```
> exp(r2)/exp(r1)  
2.119566
```

OTR **Which corresponds to the OR value for x_3 above.** 4

Logit model: predicted probabilities (case 1)

To estimate the predicted probabilities, we need to set the initial conditions.

CASE 1: Getting predicted probabilities holding all predictors or independent variables to their means.

```
allmean <- data.frame(x1=mean(mydata$x1),  
                      x2=mean(mydata$x2),  
                      x3=mean(mydata$x3))
```

Creating a new dataset with
the mean values of the
predictors

```
allmean  
x1           x2           x3  
1 0.6480006 0.1338694 0.761851
```

After estimating the logit model and creating the dataset with the mean values of the predictors, you can use the `predict()` function to estimate the predicted probabilities (for help/details type `?predict.glm`), and add them to the `allmean` dataset.

```
allmean$pred.prob <- predict(logit, newdata=allmean, type="response")
```

The object with the
logit coefficients

Dataset with the
conditions

Requesting predicted
probabilities

```
allmean  
x1           x2           x3 pred.prob  
1 0.6480006 0.1338694 0.761851 0.8328555
```

When all predictor values are held to their means, the probability of $y = 1$ is 83%.

Logit model: marginal effects

```
# Using package -mfx-
# See http://cran.r-project.org/web/packages/mfx/mfx.pdf
install.packages("mfx") #Do this only once
library(mfx)
logitmfx(y_bin ~ x1+x2+x3, data=mydata)
Call:
logitmfx(formula = y_bin ~ x1 + x2 + x3, data = mydata)
```

Marginal Effects:

	dF/dx	Std. Err.	z	P> z
x1	0.119965	0.104836	1.1443	0.25249
x2	0.051024	0.041155	1.2398	0.21504
x3	0.104574	0.053890	1.9405	0.05232 .

Signif. codes: 0 '****' 0.001 '***' 0.01 '**' 0.05 '*' 0.1 '.' 1

Marginal effects show the change in probability when the predictor or independent variable increases by one unit. For continuous variables this represents the instantaneous change given that the 'unit' may be very small. For binary variables, the change is from 0 to 1, so one 'unit' as it is usually thought.

Ordinal logit model

```
# Getting sample data
library(foreign)
mydata <- read.dta("http://dss.princeton.edu/training/Panel101.dta")
```

```
# Loading library -MASS-
library(MASS)
```

```
# Running the ordered logit model
```

```
m1 <- polr(opinion ~ x1 + x2 + x3, data=mydata, Hess=TRUE)
```



```
summary(m1)
```

```
Call:
```

```
polr(formula = opinion ~ x1 + x2 + x3, data = mydata, Hess = TRUE)
```

Coefficients:

	Value	Std. Error	t value
x1	0.98140	0.5641	1.7397
x2	0.24936	0.2086	1.1954
x3	0.09089	0.1549	0.5867

Intercepts:

	Value	Std. Error	t value
Str agree Agree	-0.2054	0.4682	-0.4388
Agree Disag	0.7370	0.4697	1.5690
Disag Str disag	1.9951	0.5204	3.8335

Residual Deviance: 189.6382

AIC: 201.6382

Ordinal logit model: p-values

```
# Getting coefficients and p-values
```

```
m1.coef <- data.frame(coef(summary(m1)))
```

```
m1.coef$pval = round((pnorm(abs(m1.coef$t.value), lower.tail = FALSE) * 2), 2)
```

```
m1.coef
```

	Value	Std..Error	t.value	pval
x1	0.98139603	0.5641136	1.7397134	0.08
x2	0.24935530	0.2086027	1.1953599	0.23
x3	0.09089175	0.1549254	0.5866807	0.56
Str agree Agree	-0.20542664	0.4682027	-0.4387558	0.66
Agree Disag	0.73696754	0.4696907	1.5690486	0.12
Disag Str disag	1.99507902	0.5204282	3.8335334	0.00

Ordinal logit model: predicted probabilities

```
# Use "probs" for predicted probabilities
```

```
m1.pred <- predict(m1, type="probs")  
summary(m1.pred)
```

Str agree	Agree	Disag	Str disag
Min. :0.1040	Min. :0.1255	Min. :0.1458	Min. :0.07418
1st Qu.:0.2307	1st Qu.:0.2038	1st Qu.:0.2511	1st Qu.:0.17350
Median :0.2628	Median :0.2144	Median :0.2851	Median :0.23705
Mean :0.2869	Mean :0.2124	Mean :0.2715	Mean :0.22923
3rd Qu.:0.3458	3rd Qu.:0.2271	3rd Qu.:0.2949	3rd Qu.:0.26968
Max. :0.5802	Max. :0.2313	Max. :0.3045	Max. :0.48832

The bold numbers are the predicted probabilities of each category when all predictors are at their mean value

Ordinal logit model: predicted probabilities

```
# At specific values, example x1 and x2 at their means, and x3 = 1 and x3 = 2.  
# Use "probs" for predicted probabilities given specific predictors  
setup1 <- data.frame(x1=rep(mean(mydata$x1),2),  
                      x2=rep(mean(mydata$x2),2),  
                      x3=c(1,2))  
  
setup1  
      x1      x2 x3  
1 0.6480006 0.1338694 1  
2 0.6480006 0.1338694 2
```

Setup for new predicted probabilities

```
setup1[, c("pred.prob")] <- predict(m1, newdata=setup1, type="probs")  
setup1  
      x1      x2 x3 pred.prob.Str agree pred.prob.Agree pred.prob.Disag pred.prob.Str disag  
1 0.6480006 0.1338694 1          0.2757495     0.2184382     0.2804806     0.2253318  
2 0.6480006 0.1338694 2          0.2579719     0.2135235     0.2869123     0.2415923
```

Use "class" for the predicted category

```
setup1[, c("pred.prob")] <- predict(m1, newdata=setup1, type="class")  
setup1  
      x1      x2 x3 pred.prob  
1 0.6480006 0.1338694 1      Disag  
2 0.6480006 0.1338694 2      Disag
```

Ordinal logit model: marginal effects

```
# Load package "erer", use function ocMe() for marginal effects

library(erer)

x <- ocME(m1, x.mean=TRUE)

x

      effect.Str agree effect.Agree effect.Disag effect.Str disag
x1        -0.198      -0.047       0.076       0.169
x2        -0.050      -0.012       0.019       0.043
x3        -0.018      -0.004       0.007       0.016

# Type the following if you want t and p-values

x$out
```

Sources

Greene, *Econometric Analysis*, 7th. ed.

UCLA, <http://www.ats.ucla.edu/stat/r/dae/>

StatsExchange, <http://stats.stackexchange.com/>

R packages:

-mfx- <http://cran.r-project.org/web/packages/mfx/mfx.pdf>

-erer- <http://cran.r-project.org/web/packages/erer/erer.pdf>

GLM with a Gamma-distributed Dependent Variable.

1 Introduction

I started out to write about why the Gamma distribution in a GLM is useful.

I've found it difficult to find an example which proves that is true. If you fit a GLM with the correct link and right-hand side functional form, then using the Normal (or Gaussian) distributed dependent variable instead of a Gamma will probably not result in disaster.

However, I did find out something really important, something which is mentioned in Myers, Montgomery, and Vining at several points, but I did not appreciate it until now:

The GLM really is different than OLS, even with a Normally distributed dependent variable, when the link function g is not the identity.

Using OLS with “manually transformed” data leads to horribly wrong parameter estimates.

Let y_i be the dependent variable with mean μ_i . OLS estimates:

$$E(g(y_i)) = b_0 + b_1 x_i$$

but the GLM estimates

$$g\{E(y_i)\} = b_0 + b_1 x_i$$

Suppose your link is the “natural log” $g(\mu_i) = \ln(\mu_i)$ or the “inverse” $g(\mu_i) = 1/\mu_i$. The OLS and GLM estimates will differ for any nonlinear link function.:

$$\ln(\mu_i) = f(X_i, b) = b_0 + b_1/x_i$$

or

$$1/\mu_i = f(X_i, b) = b_0 + b_1/x_i$$

then you estimate a Generalized Linear model with a Gamma distribution with

`glm(y ~ I(1/x), family=Gamma(link="log"))`

or

`glm(y ~ I(1/x), family=Gamma(link="inverse"))`.

If you mistakenly use a Normal, as in

`glm(y ~ I(1/x), family=gaussian(link="log"))`

or

`glm(y ~ I(1/x), family=gaussian(link="inverse"))`

then the estimated b's from the Gamma and Normal models will probably be similar. If your dependent variable is truly Gamma, the Gaussian is “wrong” on a variety of levels, but the predicted values are “about right.”

However, if you think you can just transform the variables yourself and run this through an OLS program, as in

`lm(ln(y) ~ I(1/x))`

or

`lm(I(1/y) ~ I(1/x))`

then the parameter estimates will be far from the mark. That happens because you have not transformed the expected values, but rather the observed values.

The only time that the GLM and OLS estimates line up is when the link function is the “identity” function

```
glm(y~I(1/x),family=Gamma(link="identity"))
```

will be similar to the OLS estimates from

```
lm(y~I(1/x))
```

What do I conclude from this? You can't stay with your old friend OLS. You really must learn and understand the GLM. My feeling then is similar to the comment attributed to NBA great Bill Russell: “They scheduled the game. We have to play. We might as well win.” If you have to learn the GLM anyway, and you use it, you might as well use the correct distribution while you are doing it.

2 Review the Gamma Handout

The Gamma handout is available in the Distributions folder of my web site.

To review briefly, let the shape parameter be α_i and scale be β_i . For the i 'th case being considered, we are acting as though there are individualized parameters for each case. It is annoying to keep all of this indexed by i , but sometimes it pays off. The probability density of observing a particular value y_i given parameters α_i and β_i is

$$f(y_i) = \frac{1}{\beta_i^{\alpha_i} \Gamma(\alpha_i)} y_i^{(\alpha_i - 1)} e^{-(y_i/\beta_i)} \quad y_i, \alpha_i, \beta_i > 0$$

and

$$E(y_i) = \alpha_i * \beta_i$$

$$Var(y_i) = \alpha_i * \beta_i^2$$

Regression with the gamma model is going to use input variables X_i and coefficients to make a prediction about the mean of y_i , but in actuality we are really focused on the scale parameter β_i . Generally, we assume $\alpha_i = \alpha$, α_i is the same for all observations. Variation from case to case in $\mu_i = \beta_i \alpha$ is due simply to variation in β_i . The shape parameter is just a multiplier (which is equal to the inverse of the “dispersion parameter” ϕ that is defined for all distributions that are members of the exponential family).

3 Note the linkage between mean and variance

The ratio of the mean to the variance is a constant—the same no matter how large or small the mean is. As a result, when the expected value is small—near zero—the variance is small as well. Conversely, when the expected value is larger, the observed scores are less predictable in absolute terms.

$$\frac{Var(y_i)}{E(y_i)} = \frac{\alpha_i \beta_i^2}{\alpha_i \beta_i} = \beta_i$$

If your Gamma variable has an expected value of 100, the variance has to be $100 \cdot \beta_i$. Strange, but true.

The so-called coefficient of variation, which is used in introductory statistics as a summary of variability, is the ratio of standard deviation to mean. It is also a constant

$$CV = \frac{\sqrt{Var(y_i)}}{E(y_i)} = \frac{\sqrt{\alpha_i \cdot \beta_i^2}}{\alpha_i \beta_i} = \frac{\sqrt{\alpha_i} \beta_i}{\alpha_i \beta_i} = \frac{1}{\sqrt{\alpha_i}}$$

If your Gamma variable's expected value is 100, the standard deviation is $100/\sqrt{\alpha_i}$.

It seems odd (surprising, interesting, possibly mistaken) to me that the ratio Var/E depends on β_i but the ratio of $StdDev/E$ depends on α_i .

The relationship between mean and variance here is different than some other distributions because it is “adjustable”. In contrast, the Poisson or Negative Binomial distributions have no such tuning parameter.

4 Gamma as a member of the Exponential Family

In order to treat this as the basis for a Generalized Linear Model, you act as though α is a known feature, the same for all observations. So we don't need to write subscripts on α . Then we treat β_i —the scale parameter—as the parameter of interest. That is what we are trying to predict.

Recall the exponential family has this form, $\exp[(y_i \cdot \theta_i - c(\theta_i))/\phi + h(y_i, \phi)]$. Rearrange the density for the Gamma as follows:

$$\exp\{-y_i/\beta_i + (\alpha - 1)\ln(y_i) - \ln[\beta_i^{alp ha}] - \ln[\Gamma(\alpha)]\}$$

$$\exp\{-y_i/\beta_i + (\alpha - 1)\ln(y_i) - \alpha\ln[\beta_i] - \ln[\Gamma(\alpha)]\}$$

$$\exp\{-y_i/\beta_i - \alpha\ln[\beta_i] + (\alpha - 1)\ln(y_i) - \ln[\Gamma(\alpha)]\}$$

Now a sneaky math guy trick appears. “Guess” that the natural parameter is

$$\theta_i = -\frac{1}{\alpha\beta_i}$$

Consequently,

$$\frac{-1}{\beta_i} = \theta_i\alpha$$

and

$$\beta_i = -\frac{1}{\theta_i\alpha}$$

Using those findings in the previous expression,

$$\exp\{\alpha y_i \theta_i - \alpha \ln(-\frac{1}{\theta_i\alpha}) - \alpha \ln(\alpha) + (\alpha - 1)\ln(y_i) - \ln[\Gamma(\alpha)]\}$$

$$\exp\{\alpha y_i \theta_i - \alpha \ln(-\frac{\alpha}{\theta_i\alpha}) + (\alpha - 1)\ln(y_i) - \ln[\Gamma(\alpha)]\}$$

$$\exp\{\alpha y_i \theta_i - \alpha \ln(-\frac{1}{\theta_i}) + (\alpha - 1)\ln(y_i) - \ln[\Gamma(\alpha)]\}$$

$$\exp\{\alpha(y_i \theta_i - \ln(-\frac{1}{\theta_i})) + (\alpha - 1)\ln(y_i) - \ln[\Gamma(\alpha)]\}$$

That was quite a lot of work to find out that $\alpha = 1/\phi$ and that $c(\theta_i) = \ln(-1/\theta_i)$. But if we rearrange just one more time, we find the Gamma in the form of the exponential density.

$$\exp\{\frac{y_i \theta_i - \ln(-1/\theta_i)}{\phi} + (\frac{1-\phi}{\phi})\ln(y_i) - \ln[\Gamma(\phi^{-1})]\}$$

Then you can use the GLM Facts described on my GLM handout #1.

GLM Fact #1 states that $\mu_i = dc(\theta_i)/d\theta_i$, and so that implies the Gamma's μ_i is

$$\frac{dc(\theta_i)}{d\theta_i} = \frac{d\ln(-1/\theta_i)}{d\theta_i} = -\frac{d\ln(\theta_i)}{d\theta_i} = -\frac{1}{\theta_i} = \alpha_i\beta_i$$

GLM Fact #2 states that $V(\mu_i) = d^2c(\theta_i)/d\theta_i^2$, and so, in this case,

$$V(\mu_i) = \frac{d}{d\theta_i^2}(-1/\theta) = \frac{1}{\theta_i^2} = \mu^2 = (\alpha\beta_i)^2$$

These findings are internally consistent with what we know already (or can check in textbooks). Recall from the GLM notes that the observed variance of y_i has two components: .

$$Var(y_i) = \phi_i V(\mu_i)$$

For the Gamma, we already know that $E(y_i) = \mu_i = \alpha * \beta$ and $Var(y_i) = \alpha * \beta^2$. The variance function is $V(\mu_i) = \mu_i^2 = \alpha^2 * \beta^2$, and the dispersion parameter ϕ_i must be equal to the reciprocal of the shape parameter $1/\alpha$. You can easily verify that all of these separate pieces work together in a logical way:

$$\begin{aligned} Var(y_i) &= \phi_i V(\mu) = \phi_i \cdot a^2 \beta^2 \\ &= \alpha \beta^2 \text{ where } \phi_i = \frac{1}{\alpha} \end{aligned}$$

Keep in mind, then, that when the GLM routine estimates dispersion- ϕ -it is estimating the reciprocal of the shape parameter.

5 The Reciprocal is the Canonical Link

The canonical link for the GLM with a Gamma-distributed dependent variable is the reciprocal, $1/\mu_i$. That means that the expected value of your observed y_i , ($E(y_i) = \mu_i$), is related to your input variables as, for example,

$$\frac{1}{\mu_i} = b_0 + b_1 x_{1i}$$

Which obviously implies

$$\mu_i = \frac{1}{b_0 + b_1 x_{1i}}$$

Plot that! In Figure 1, you see there is some serious potential for funny business with this function. There is funny business because:

- It is not always positive
- It has vertical asymptotes

6 Why would you want a Gamma-distributed dependent variable?

This is a difficult question. Theoretically, the Gamma should be the right choice when the dependent variable is real-valued on a range from 0 to ∞ . And the Gamma is suitable when you suspect the linkage between mean and variance is “fixed”. If you expect a small value of y_i , you should also expect only a small amount of variability in observed values. Conversely, if you expect a huge value of y_i , you should expect a lot of variability.

However, after some testing, I have developed some doubts about the need to change from a model based on the Normal distribution to a model based on the Gamma. The Gamma may be “theoretically right” but there are several cases in which the old “theoretically wrong” Normal OLS model seems to do about as well.

This is especially true if the Gamma parameters are tuned so that the distribution is symmetrical, but even when it is pretty badly skewed, I find the OLS predictions are as good.

However, I find some cases where using the GLM with a Gamma distribution has a dramatic impact. The differences hinge on the functional form being investigated.

So I’ve prepared some vignettes.

Figure 1: Reciprocal Link

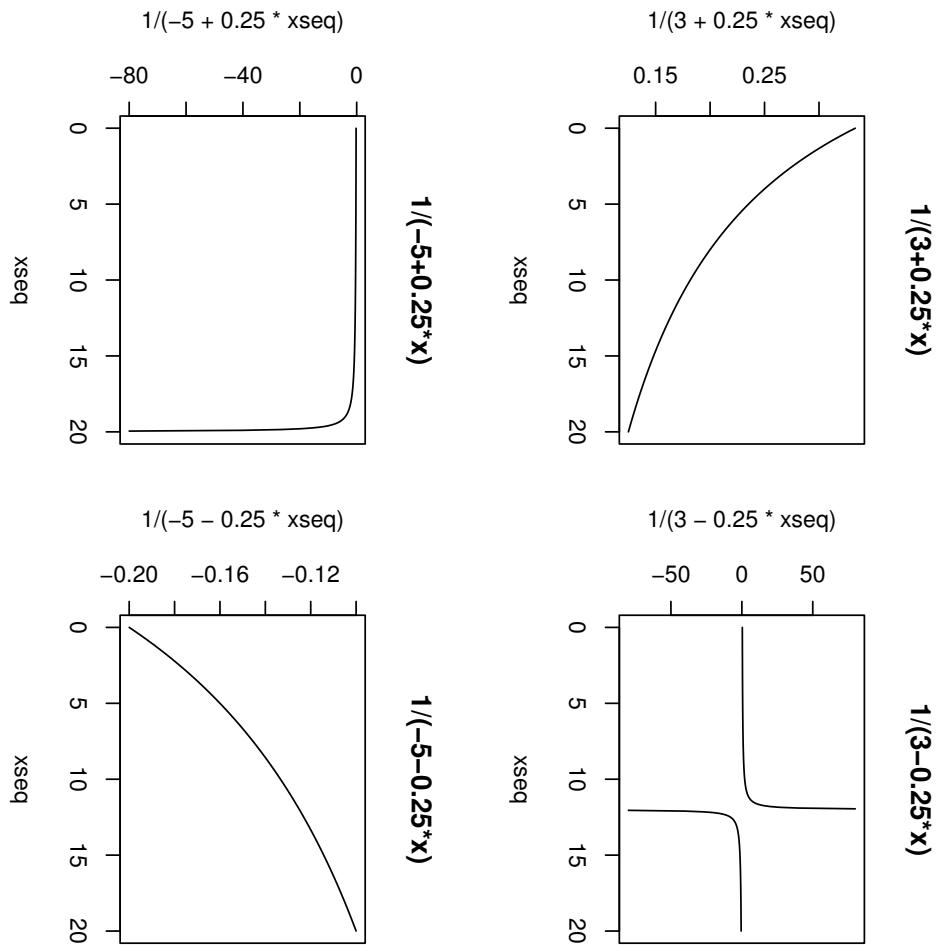
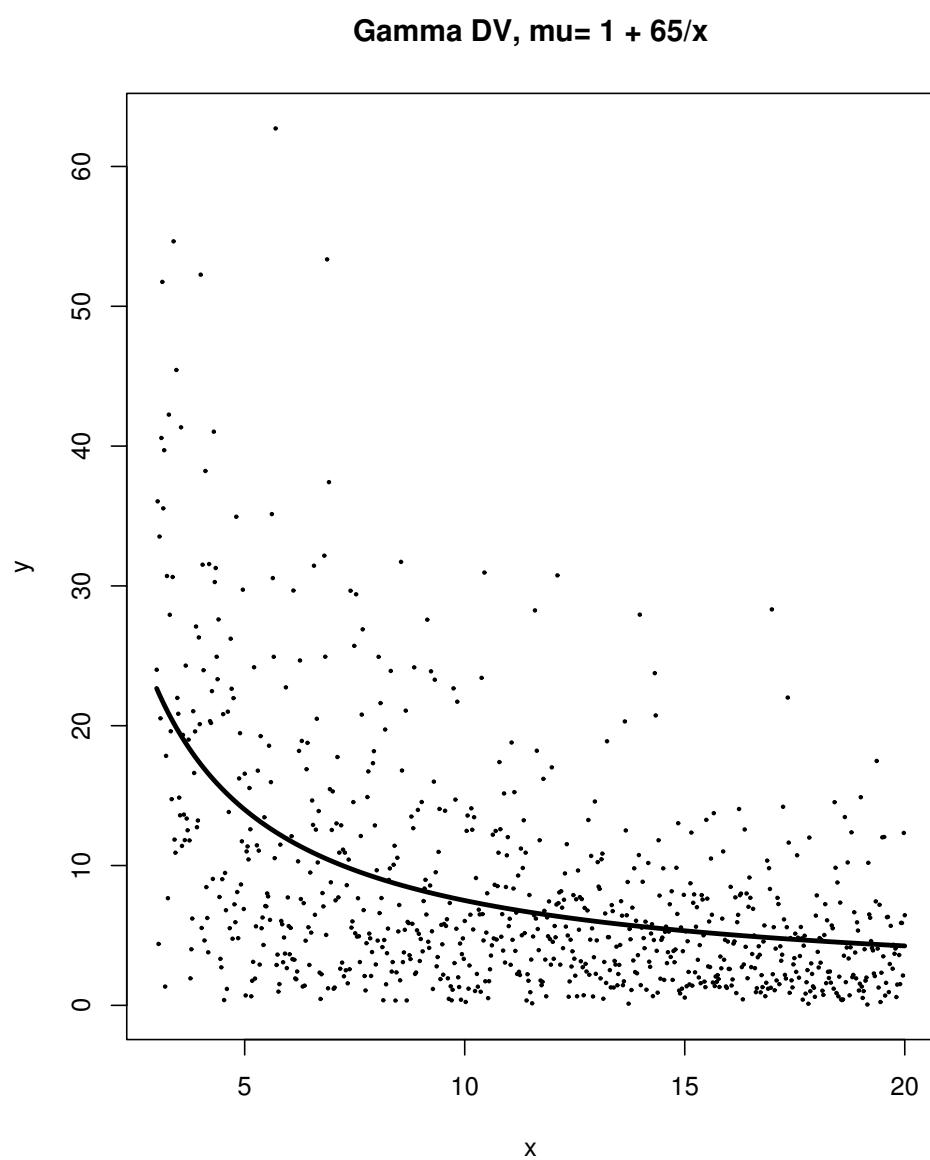


Figure 2: Gamma Dependent Variable $\mu_i = 1 + 65/x_i$, shape=1.5



7 Identity link, reciprocal on the right hand side: $\mu_i = b_o + b_1/x_i$

Some simulated data is presented in Figure2. The line represents the “true” value of μ_i , the expected value of the dependent variable.

7.1 GLM fit for the Reciprocal Model

The Generalized Linear Model can be fit with the “identity” link with these commands.

```
> agam <- glm(yobs ~ I(1/xseq), family = Gamma(link = "identity"),
+   control = glm.control(maxit = 100), start = c(1, 65))
> library(MASS)
> myshape <- gamma.shape(agam)
> gampred <- predict(agam, type = "response", se = T, dispersion = 1/myshape$alpha)
```

(Side note about estimating dispersion: This uses the MASS library’s function gamma.shape to calculate a more precise estimate of the gamma distribution’s shape parameter, which is equal to the reciprocal of the GLM’s dispersion $\alpha = 1/\phi$. This is useful because the estimate of the dispersion offered by the default GLM summary command does not take into account the special information about the dispersion that can be calculated by using the Gamma distribution. Not all GLMs have a model-specific, enhanced way to estimate dispersion.)

```
> summary(agam, dispersion = 1/myshape$alpha)

Call:
glm(formula = yobs ~ I(1/xseq), family = Gamma(link = "identity"),
     start = c(1, 65), control = glm.control(maxit = 100))

Deviance Residuals:
    Min      1Q  Median      3Q      Max 
-2.5702 -0.8164 -0.2555  0.3364  2.5154 

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept)  0.7738    0.4524   1.71   0.0872 .  
I(1/xseq)   67.7209   5.4534  12.42  <2e-16 *** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for Gamma family taken to be 0.6757266)

Null deviance: 786.66 on 799 degrees of freedom
Residual deviance: 599.11 on 798 degrees of freedom
AIC: 4756

Number of Fisher Scoring iterations: 3
```

7.2 Linear Model Fit with the Normal Distribution

Suppose you make the mistaken assumption that this data is Normally distributed. The default settings of the glm estimator in R lead to estimates for a Normally distributed dependent variable with the identity link.

```
> lmod <- glm(yobs ~ I(1/xseq))
> impred <- predict(lmod, se = T)
```

We have asked predict for the standard errors because they are useful for plots shown below.

```
> summary(lmmod)

Call:
glm(formula = yobs ~ I(1/xseq))

Deviance Residuals:
    Min      1Q  Median      3Q      Max 
-21.467   -3.929   -1.500    2.537   49.790 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  0.3723     0.5212   0.714   0.475    
I(1/xseq)    71.6044    4.0309  17.764  <2e-16 ***  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 55.12723)

Null deviance: 61387 on 799 degrees of freedom
Residual deviance: 43992 on 798 degrees of freedom
AIC: 5482
```

Number of Fisher Scoring iterations: 2

Please note that you get the same parameter estimate if you put the same relationship through the ordinarily least squares regression procedure, lm.

```
> lmmod1a <- lm(yobs ~ I(1/xseq))
> summary(lmmod1a)

Call:
lm(formula = yobs ~ I(1/xseq))

Residuals:
    Min      1Q  Median      3Q      Max 
-21.467   -3.929   -1.500    2.537   49.790 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  0.3723     0.5212   0.714   0.475    
I(1/xseq)    71.6044    4.0309  17.764  <2e-16 ***  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.425 on 798 degrees of freedom
Multiple R-Squared:  0.2834,    Adjusted R-squared:  0.2825 
F-statistic: 315.6 on 1 and 798 DF,  p-value: < 2.2e-16
```

7.3 Is the Normal Model “Just as Good”?

No, of course it isn’t. It is the wrong model. But if you mistakenly used OLS, would you make a major mistake? In this test case, the answer is no.

1. The parameter estimates from the two models are “about the same.”

$$\text{Gamma GLM} : \hat{\mu}_i = 0.77 + 67.72/x_i$$

$$\text{Normal GLM} : \hat{\mu}_i = 0.37 + 71.60/x_i$$

2. Consequently, the predicted values from the two models are “about the same.”

Consider the plotted lines in Figure 3. It is difficult to distinguish the two lines representing the predicted values. I had a hard time believing that the two lines could actually be so close to one another, so I printed out the first 10 observations of the two models:

```
> cbind(glmGamma = gampred$fit[1:10], glmNormal = lmpred$fit[1:10])

  glmGamma glmNormal
1 23.34747 24.24043
2 23.18850 24.07235
3 23.03175 23.90661
4 22.87719 23.74318
5 22.72475 23.58200
6 22.57440 23.42303
7 22.42610 23.26622
8 22.27980 23.11154
9 22.13546 22.95892
10 21.99305 22.80835
```

The plotted estimates of the means, along with the “confidence intervals”, are illustrated in Figure 4.

3. If you (mistakenly) choose models by T statistics, you will be wrong.

It upsets me when students say one model is “more significant” to mean that a model has coefficients with bigger t values. In this case, the t value for the coefficient of $1/x_i$ in the Normal model is 17.76, while the comparable value from the Gamma fit is 12.42. That does not mean the Normal model is better, for many reasons. The fact is that these tests assume you have chosen the correct model and then estimate on the variability of the \hat{b} based on your specification. They do not constitute a way to choose between two models.

4. The Deviance is different: Gamma looks significantly better.

The residual deviance of the Gamma fit is 599.11 on 798 degrees of freedom, and the Akaike Information Criterion is 4765. The residual deviance of the Normal model is 43992 on 798 degrees of freedom, and the AIC is 5482. The model with the smaller AIC is preferred.

8 Reciprocal link, reciprocal on the right hand side: $\mu_i = x_i/(b_1 + b_0 x_i)$

R documents refer to this as a Michaelson-Morley model, but it has other names because it has been found in many fields. You theorize that:

$$\mu_i = \frac{x_i}{b_1 + b_0 x_i}$$

Note: the numbering of the coefficients is not mistaken.

Rewrite and you should see why the reciprocal link (the canonical link) makes sense:

$$\frac{1}{\mu_i} = \frac{b_1 + b_0 x_i}{x_i}$$

Figure 3: Fitted Models for Gamma Dependent Variable

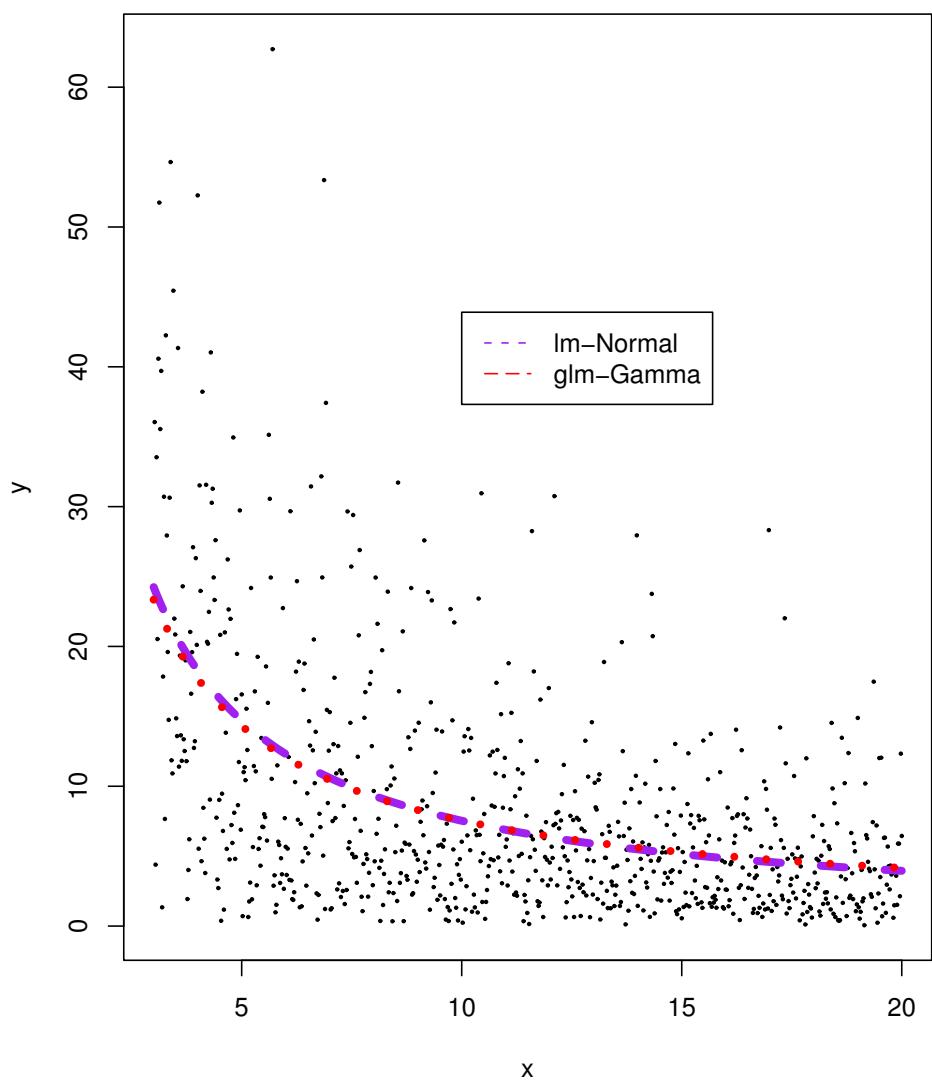
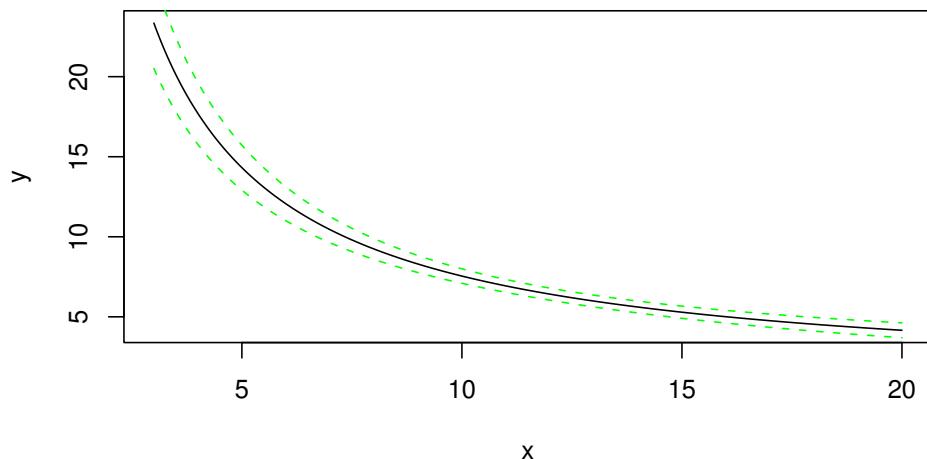


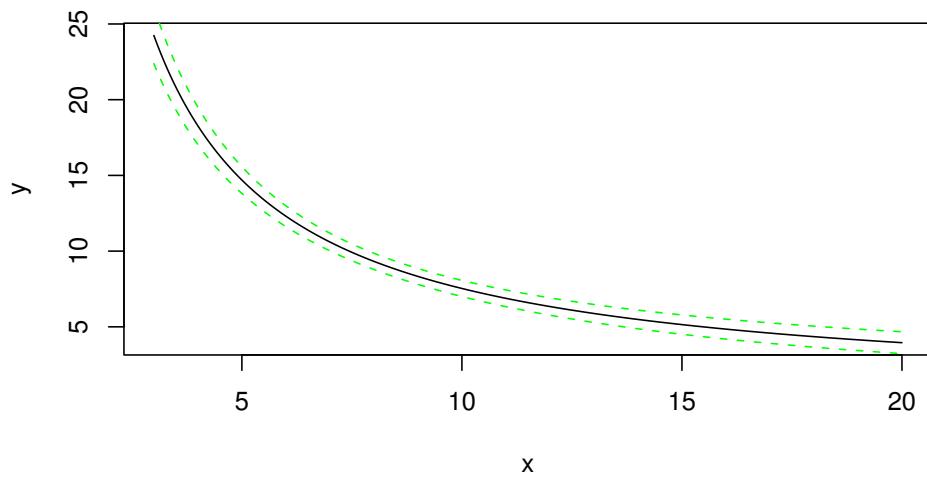
Figure 4: Predicted Values from glm with Gamma and Gaussian Distributions

```
> par(mfcol = c(2, 1))
> plot(xseq, gampred$fit, type = "l", xlab = "x", ylab = "y", main = "Gamma, link=identity")
> lines(xseq, gampred$fit + 2 * gampred$se, lty = 2, col = "green")
> lines(xseq, gampred$fit - 2 * gampred$se, lty = 2, col = "green")
> plot(xseq, lmpred$fit, type = "l", xlab = "x", ylab = "y", main = "Gaussian, link=identity")
> lines(xseq, lmpred$fit + 2 * lmpred$se, lty = 2, col = "green")
> lines(xseq, lmpred$fit - 2 * lmpred$se, lty = 2, col = "green")
```

Gamma, link=identity



Gaussian, link=identity



$$\frac{1}{\mu_i} = b_0 + b_1 \frac{1}{x_i}$$

Write it like this and it should remind you of a logistic regression.

$$\mu_i = \frac{1}{b_0 + b_1(1/x_i)}$$

One should graph that. For $x_i = 0$, this is undefined (just as Gamma density is undefined). For very small values of x_i , say 0.001, you can see the expected value is a very small number. As x_i gets bigger and bigger, the expected value tends to $1/b_0$.

Variants of this are known in ecology, biochemistry, and physics. In R, one finds it discussed as the Michaelson-Morley model.

A hypothetical example of Gamma distributed data with $\mu_i = x_i/(3 + 0.25x_i)$ with the Gamma shape parameter equal to 1.5 is presented in Figure 5.

8.1 GLM fit with a Gamma Variable and Log Link

If one algebraically re-arranged the model as

$$\frac{1}{\mu_i} = b_0 + b_1 \frac{1}{x_i}$$

then one would have to transform the input variable in the regression model, but the `glm` procedure will handle the transformation of the left hand side. One should write the `glm` formula as

`y ~ I(1/x)`

but specify the link as the “inverse”, so that the mean of y_i , μ_i , is transformed.

The `glm` procedure to fit a Gamma distributed dependent variable of this sort is:

```
> agam2 <- glm(yobs2 ~ I(1/xseq), family = Gamma(link = "inverse"),
+   control = glm.control(maxit = 100), start = c(2, 4))
> library(MASS)
> myshape2 <- gamma.shape(agam2)
> gampred2 <- predict(agam2, type = "response", se = T, dispersion = 1/myshape2$alpha)
```

This uses the MASS library’s `gamma.shape` method to get a better estimate of the dispersion parameter, which is then used in making predictions and also in preparing the summary output. The estimate of the dispersion coefficient affects the standard errors, but not the estimates of the b ’s.

`> summary(agam2, dispersion = 1/myshape2$alpha)`

Call:

```
glm(formula = yobs2 ~ I(1/xseq), family = Gamma(link = "inverse"),
  start = c(2, 4), control = glm.control(maxit = 100))
```

Deviance Residuals:

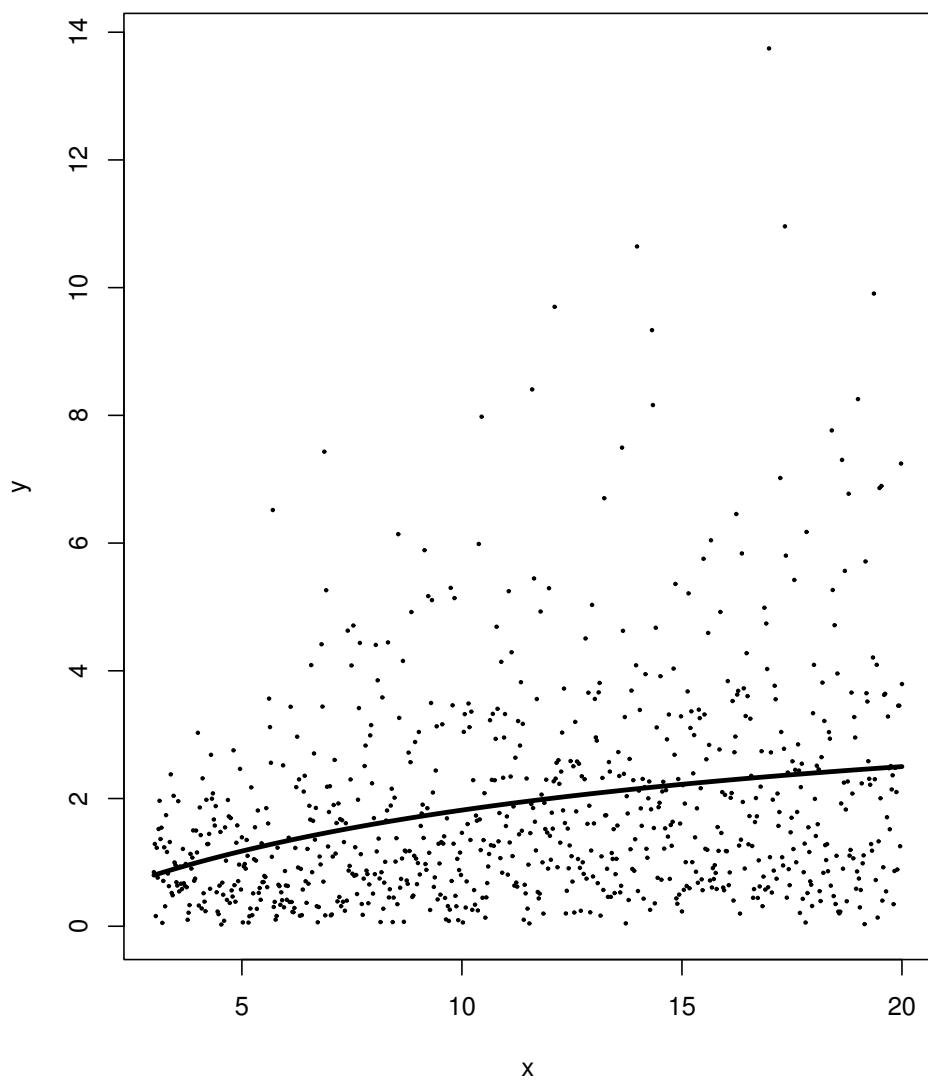
Min	1Q	Median	3Q	Max
-2.5708	-0.8150	-0.2532	0.3325	2.5156

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	0.26736	0.03352	7.976	1.51e-15 ***
I(1/xseq)	2.80621	0.34735	8.079	6.54e-16 ***

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Figure 5: Gamma Dependent Variable $\mu_i = x_i/(3 + 0.25x_i)$



(Dispersion parameter for Gamma family taken to be 0.6756965)

Null deviance: 654.43 on 799 degrees of freedom
Residual deviance: 599.09 on 798 degrees of freedom
AIC: 2479.4

Number of Fisher Scoring iterations: 7

8.2 What if you used OLS?

You can translate this into a form that looks like an ordinary regression model: just “tack on an error term” (recall OLS: expected value of 0, constant variance):

$$\frac{1}{y_i} = b_0 + b_1 \frac{1}{x_i} + e_i$$

and create transformed variables $1/y_i$ and $1/x_i$ and estimate this with OLS. How gauche.

```
> lmmod2 <- lm(I(1/yobs2) ~ I(1/xseq))
> lmpred2 <- predict(lmmod2, se = T)
```

There are a number of reasons why you should not do that. It violates the usual OLS assumptions. It assumes the mismatch between the expected and observed is of a very peculiar sort, $E(e_i) = 0$ and constant variance.

The most important reason why you should not fit these parameters with OLS is that the resulting parameter estimates are grossly wrong.

```
> summary(lmmod2)

Call:
lm(formula = I(1/yobs2) ~ I(1/xseq))

Residuals:
    Min      1Q  Median      3Q     Max 
-2.24314 -1.00187 -0.70688 -0.04022 34.96570 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  0.8402     0.2020   4.159 3.55e-05 ***
I(1/xseq)    5.9798     1.5627   3.827  0.00014 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.878 on 798 degrees of freedom
Multiple R-Squared:  0.01802,    Adjusted R-squared:  0.01679 
F-statistic: 14.64 on 1 and 798 DF,  p-value: 0.0001401
```

Note, the predicted values from this model are presented on a reciprocal scale, so the predicted values must be transformed as $1/predicted$ in order to be plotted on the scale of the original, untransformed data.

8.3 But did you really need the Gamma in the glm?

Here we are back to the main question: is it the functional form that is the source of the trouble, or is it the assumed statistical distribution.

The reason that the OLS estimation fails so dramatically is that all of the y_i values are transformed. We really only wanted to represent the transformation $1/\mu_i$, but in the lm framework, doing os requires us

Figure 6: Fitted Models for $\mu = x_i/(3 + .25x_i)$ with a Gamma Distributed Variable

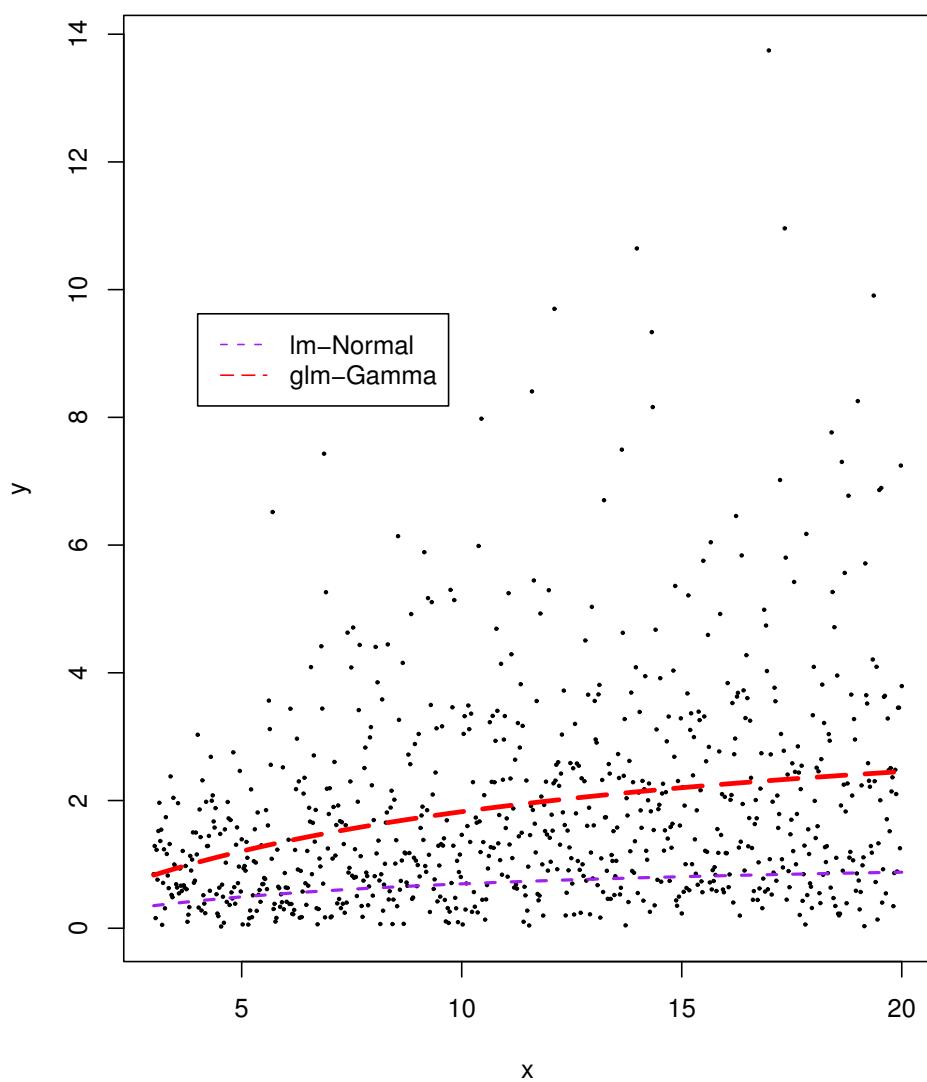
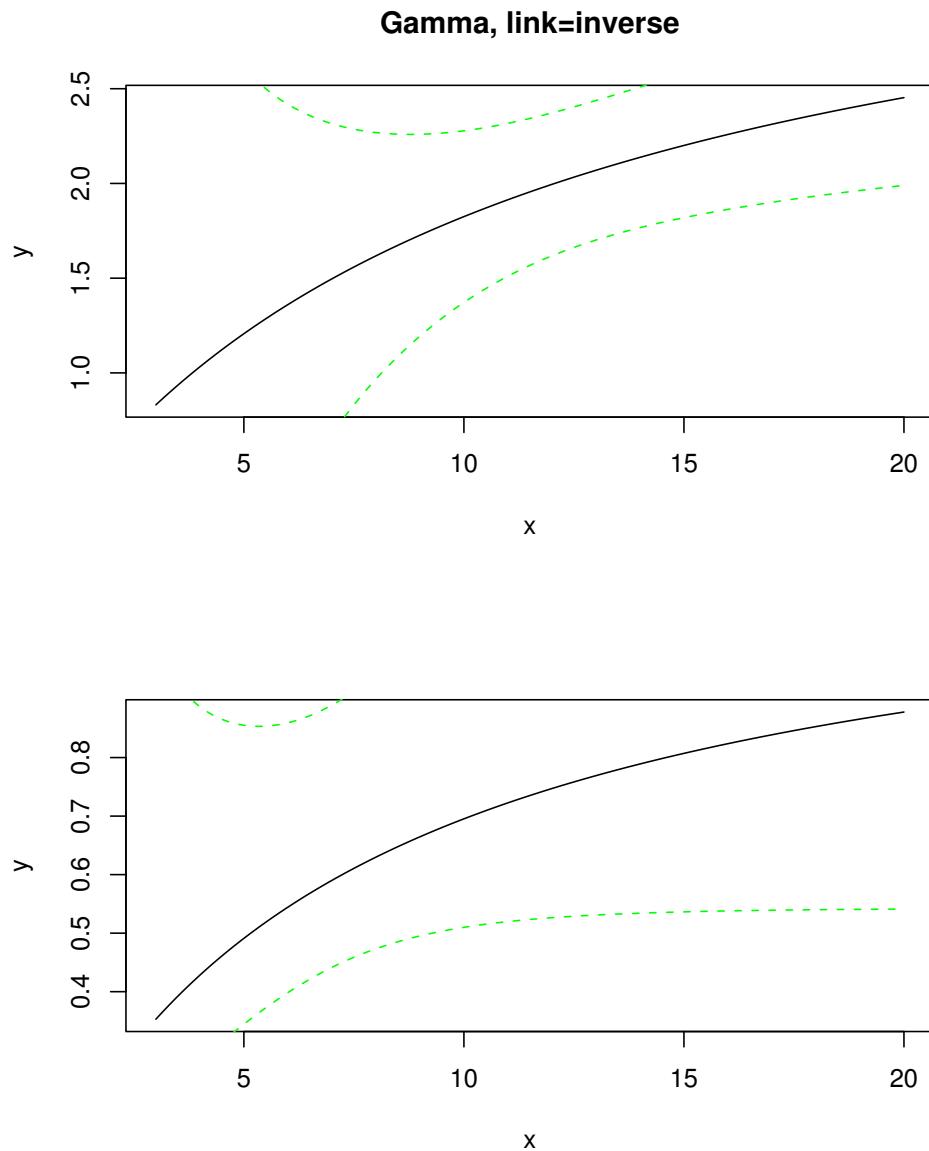


Figure 7: Predicted Values from glm-Gamma and lm-Normal

```
> par(mfcol = c(2, 1))
> plot(xseq, gampred2$fit, type = "l", xlab = "x", ylab = "y",
+       main = "Gamma, link=inverse")
> lines(xseq, gampred2$fit + 2 * gampred$se, lty = 2, col = "green")
> lines(xseq, gampred2$fit - 2 * gampred$se, lty = 2, col = "green")
> plot(xseq, 1/lmpred2$fit, type = "l", xlab = "x", ylab = "y",
+       main = "")
> ypsem <- lmpred2$fit - 1.96 * lmpred$se
> ypsep <- lmpred2$fit + 1.96 * lmpred$se
> lines(xseq, 1/ypsem, lty = 2, col = "green")
> lines(xseq, 1/ypsep, lty = 2, col = "green")
```



to transform the observe values $1/y_i$. We really want only to model the transformation of the mean and GLM does that.

Now we can use `glm` with a Gaussian distribution but with an inverse link. And, unlike the model discussed in the previous section, there IS a difference between using `lm` and `glm` on the same data.

To estimate the model with GLM,

$$\frac{1}{\mu_i} = b_0 + b_1 \frac{1}{x_i}$$

the following R commands are used:

```
> lmmod3 <- glm(yobs2 ~ I(1/xseq), family = gaussian(link = "inverse"))
> summary(lmmod3)

Call:
glm(formula = yobs2 ~ I(1/xseq), family = gaussian(link = "inverse"))

Deviance Residuals:
    Min      1Q  Median      3Q      Max 
-2.3914 -1.0866 -0.3662  0.5736 11.4282 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  0.26355   0.03979   6.623 6.46e-11 ***
I(1/xseq)   2.85141   0.50427   5.655 2.18e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 2.702359)

Null deviance: 2320.6 on 799 degrees of freedom
Residual deviance: 2156.5 on 798 degrees of freedom
AIC: 3069.6

Number of Fisher Scoring iterations: 7
```

Note that the parameter estimates coincide with the GLM-Gamma estimates of the earlier analysis. Figure 8 plots the predicted values for the `glm` models fit with the Normal and Gamma distributions. The predicted values once-again coincide in these two models fit with `glm`.

8.4 In the end, its all about the deviance

The residual deviance for the GLM-Gamma model in section 8.1 was 599.09 on 798 degrees of freedom with an AIC of 2479.4. The GLM-Normal model has deviance 2156.6 and the AIC was 3069.6.

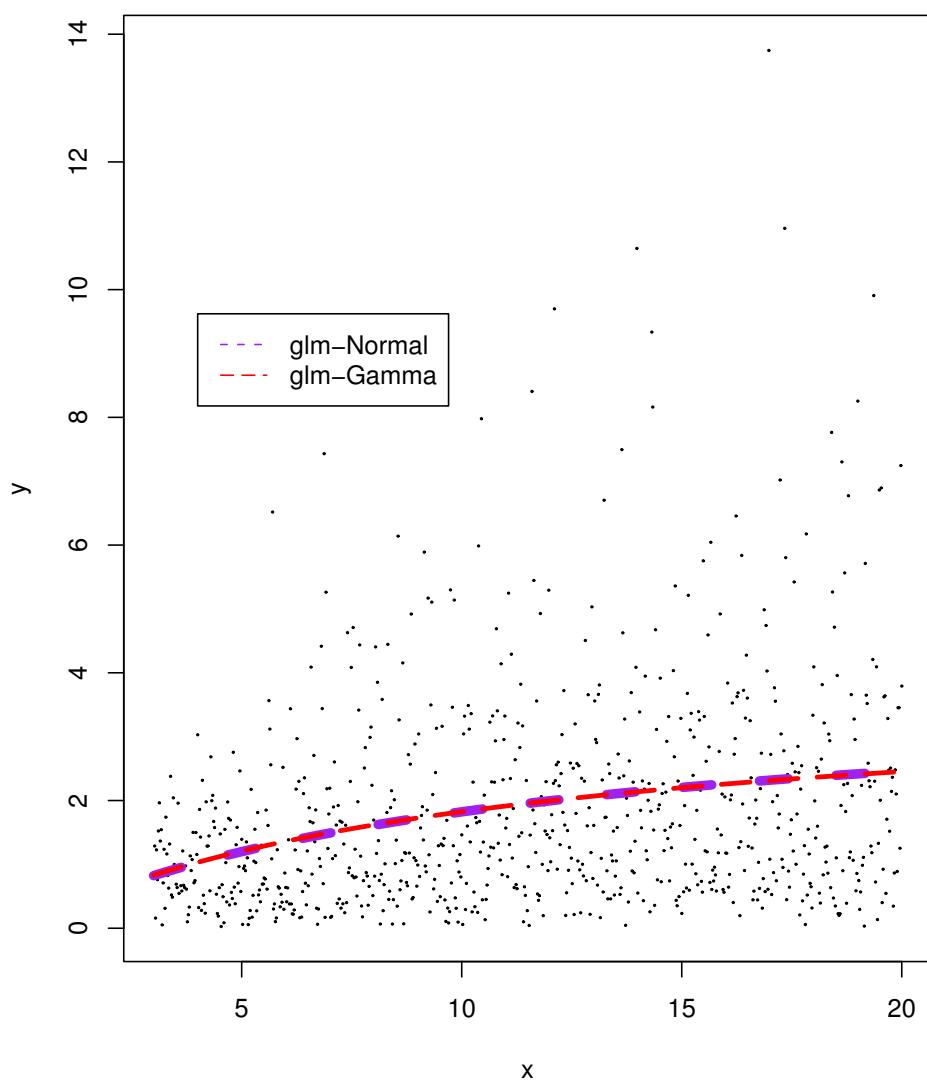
9 Final note about the shape parameter

You might have noticed that the Gamma is a two-parameter distribution. However, it is not necessary to know α before doing the estimates of the slope coefficients. After fitting the slope coefficients, then the dispersion can be estimated (similar in the way OLS estimates the slopes and then the variance of the error term).

In MASS 4ed, Venables and Ripley justify the separate estimate of the dispersion coefficient from the other coefficients by observing that they are orthogonal (p. 185). They observe (revising terminology to fit this note)

$$E \left(\frac{\partial l(\theta, \phi; y)}{\partial \theta \partial \phi} \right) = 0$$

Figure 8: Fitted Models for $\mu = x_i/(3 + .25x_i)$ with a Gamma Distributed Variable



In words, the likelihood's rate of change in θ is not affected by a change in ϕ , and conversely, the rate of change in ϕ is not affected by θ . So we can estimate the b coefficients which impact θ separately from ϕ .

Practical Assessment, Research & Evaluation

A peer-reviewed electronic journal.

Copyright is retained by the first or sole author, who grants right of first publication to *Practical Assessment, Research & Evaluation*. Permission is granted to distribute this article for nonprofit, educational purposes if it is copied in its entirety and the journal is credited. PARE has the right to authorize third party reproduction of this article in print, electronic and database forms.

Volume 21, Number 2, February 2016

ISSN 1531-7714

Tutorial on Using Regression Models with Count Outcomes using R

A. Alexander Beaujean, Grant B. Morgan, *Baylor University*

Education researchers often study count variables, such as times a student reached a goal, discipline referrals, and absences. Most researchers that study these variables use typical regression methods (i.e., ordinary least-squares) either with or without transforming the count variables. In either case, using typical regression for count data can produce parameter estimates that are biased, thus diminishing any inferences made from such data. As count-variable regression models are seldom taught in training programs, we present a tutorial to help educational researchers use such methods in their own research. We demonstrate analyzing and interpreting count data using Poisson, negative binomial, zero-inflated Poisson, and zero-inflated negative binomial regression models. The count regression methods are introduced through an example using the number of times students skipped class. The data for this example are freely available and the R syntax used run the example analyses are included in the Appendix.

Count variables such as number of times a student reached a goal, discipline referrals, and absences are ubiquitous in school settings. After a review of published single-case design studies Shadish and Sullivan (2011) recently concluded that nearly all outcome variables were some form of a count. Yet, most analyses they reviewed used traditional data analysis methods designed for normally-distributed continuous data.

It is not surprising that most educational research uses errant analysis techniques for count data. It is seldom taught in education coursework, and methods surveys (Aiken, West, & Millsap, 2008; Little, Akin-Little, & Lee, 2003; Perham, 2010) do not even ask about the use of count data. Nonetheless, using inappropriate regression methods can produce biased coefficients and standard errors, which can lead to errant conclusions. Consequently, for educational researchers to make the appropriate data-based decisions about positive and problem behaviors, as well as the effectiveness of interventions that target these areas, they must recognize and acknowledge the nature

of the collected variables and use the appropriate data analytic tools.

The purpose of this article is to assist educational researchers in understanding appropriate methods for count data, as well as being able to conduct independent analyses of such data. To that end, we discuss the nature of count data and present an example using freely available data. We provide the R (R Development Core Team, 2015) syntax to replicate and extend our analyses in the supplemental material. For those interested in using other software such (e.g., Stata, SAS) or extensions of the count models we discuss, Hilbe (2014) provides a book-length treatment on the topic as well as some worked examples.

Count Variables

Count variables share certain properties: (a) their values are always integers/whole numbers; (b) their lowest possible value is zero, so they can never be negative; and (c) they frequently appear to be positively skewed, with most values being low and relatively few values are high (Cameron & Trivedi, 1998). Figure 1a

shows a histogram of a typical count variable. Histograms of normally distributed and dichotomous (binomial) variables are shown in Figures 1b and 1c, respectively.

To add to their complexity, some count variables have a considerable number of zero values (see Figures 2b and 2c). This typically occurs when the variable is

residuals follow the distribution of the outcome variable, which for count variables is often neither normally distributed nor even not even symmetrical (see Figure 2). Moreover, the residual variance often increases as the predictor variables increases, which produces heteroscedasticity. Thus, using typical regression methods with count outcome variables can

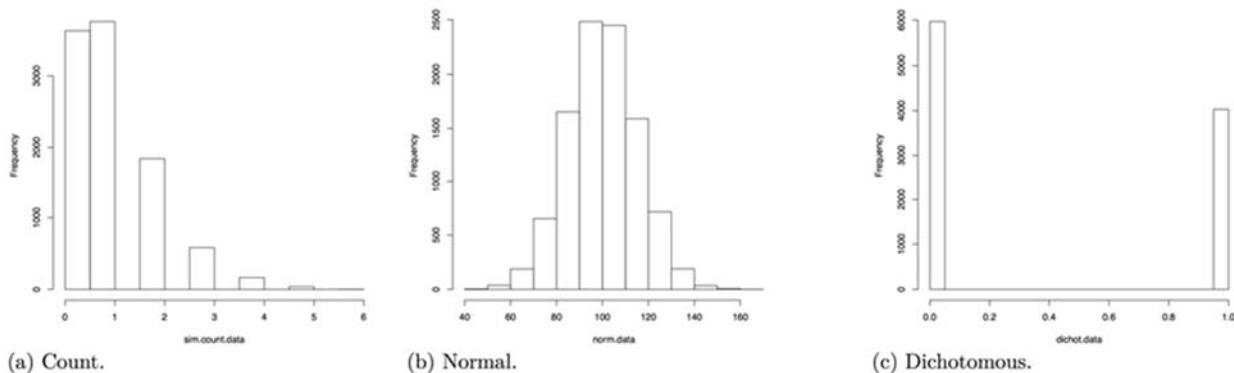


Figure 1. Histograms of variable distributions.

thought to be particularly deleterious (e.g., drug use, school expulsions), and very few observations exhibit the behavior. No matter how many zeros a count

lead to parameter bias as well as standard error and confidence interval estimates of the wrong size. This can ultimately lead an educational researcher to make

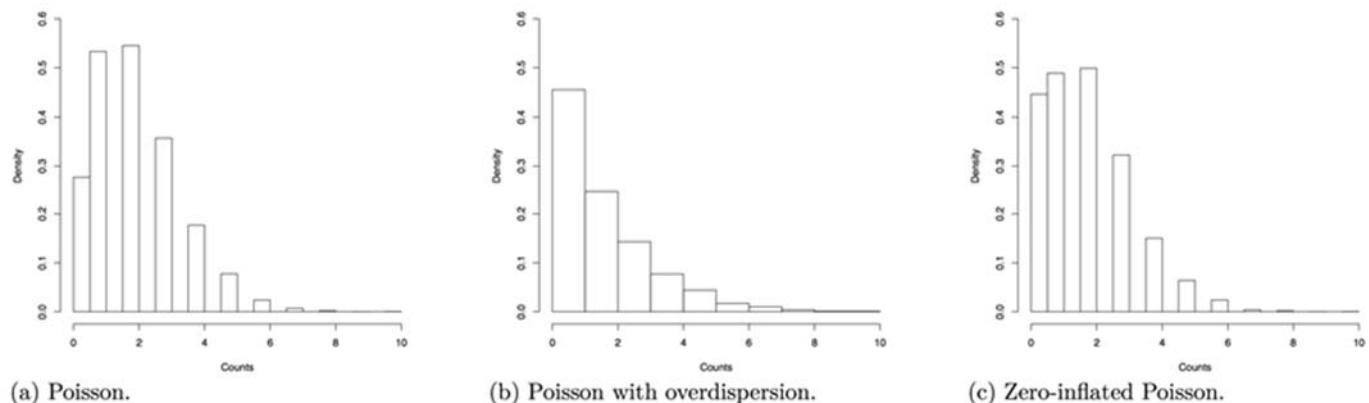


Figure 2. Plots of count variable distributions.

variable has, the plots in Figure 2 make it obvious that such data are not even symmetrical, much less normally distributed.

Regression Models for Count Data

A major assumption of typical multiple regression models is that the residuals follow a normal distribution (Cohen, Cohen, West, & Aiken, 2003). Typically, the

invalid inferences and poor decisions. Instead, regression models that account for the count nature of the outcome variable (and the subsequent nature of the model's residuals) are more appropriate.

Prior to the development of regression models for count data and their availability in common statistical programs, count variables were typically dealt with in two ways. First, people used typical linear models,

surmising that the models were robust enough to handle any assumption violations caused by the count variables. Second, they transformed the count variables to make them fit more traditional models. Both approaches are problematic.

There is a lengthy literature devoted to the robustness of traditional linear regression (e.g., multiple regression, ANOVA) to departures from normality. Although previous research has shown that the traditional regression model can produce unbiased regression coefficients with some non-normal distributions (e.g., Box, 1953; Cochran, 1947; Lix, Keselman, & Keselman, 1996), it tends to produce inflated standard errors. Moreover, the traditional regression model makes continuous predictions even though count outcomes are discrete. Therefore, the residuals tend to be heteroscedastic, which violates a major assumption of the traditional regression model.

There are a variety of transformations available for count data. One transformation involves dichotomizing the responses (e.g., yes-no/present-absent), which is then used in a logistic regression. Problems with dichotomizing variables are well known, however, and are seldom appropriate (MacCallum, Zhang, Preacher, & Rucker, 2002). Another option is a nonlinear transformation (e.g., square root, logarithm) to make the variable more closely approximate a normal distribution. Unfortunately, such transformations often have little effect when the range of values is very narrow, do not handle having an excessive amount of small values well, and do not completely eliminate heteroscedasticity (Coxe, West & Aiken, 2009). The more general Box-Cox power transformations tend to work better, but they do not always fix the problems with normality and heteroscedasticity (Sakia, 1992). Moreover, any nonlinear transformation of the outcome comes at the cost of having a more difficult model to interpret (e.g., predicting the square root of times using a drug). This cost may be acceptable when there are no known models to handle the outcome variable's native distribution. When models exist that can directly handle the variable's distribution, it is better to use them. With the development of the generalized linear model, models now exist that can directly handle the distribution of the count variables.

The generalized linear model (GLM; McCullagh & Nelder, 1989) is a framework designed to handle regression models for a variety of outcome variable types. All GLMs require two components: proper

specification of residuals' distribution and a function to link the outcome and the linear combination of the predictor variables. In a typical regression, the residuals follow a normal distribution and the link is the identity function (i.e., multiply the regression by one). For a logistic regression, the residuals follow a binomial distribution and the link is the logit function.

Count variables need to be modeled differently than either continuous or dichotomous variables (Cameron & Trivedi, 1998). Because of the different ways count variables can be distributed, there are multiple forms of the GLM for count data. In what follows, we discuss four common types of GLMs for count data, each of which is designed for a different type of count variable distribution.

Poisson Model

The most common type of distribution for count variables is a Poisson distribution, an example is shown in Figure 2a. The Poisson distribution is used because it is a probability distribution designed for non-negative

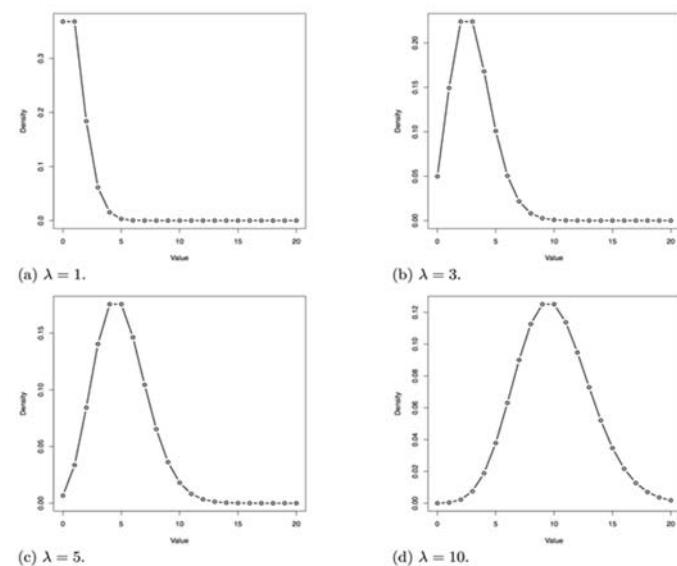


Figure 3. Plots of Poisson variable distributions with different values of λ .

integers. It is defined by a single parameter, λ , which estimates both the mean and variance of the distribution, thereby completely controlling the distribution's shape. When λ is close to zero the distribution is very positively skewed, but as λ increases the distribution becomes less skewed and appears closer to a normal distribution (see Figure 3).

The major differences between a Poisson regression and its typical regression counterpart are twofold. First, the Poisson regression model assumes the residuals follow a Poisson distribution rather than a normal distribution. Second, predictor variables are linked to the outcome via a natural log transformation (Cameron & Trivedi, 1998), similar to what is done with logistic regression (Hosmer & Lemeshow, 2000). The log transformation guarantees that the regression model's predicted values are never negative.

For a simple Poisson regression, the model is

$$\ln(\lambda_i) = \mu_i = a + bX_i$$

Predicted count (Transformed by Link) Structural (Original)

(1)

where X is a predictor variable, i represents a group of observations with the same value on X , a and b are the intercept and slope, respectively, and μ_i is the expected value of the outcome variable for all respondents whose value on X is X_i . As the mean of a Poisson distribution is λ and the link function for a Poisson regression is the natural log, Equation (1) shows that the mean of the regression equation, μ_i , equals $\ln(\lambda_i)$.

To return the outcome variable to its original count scale requires transforming the structural part of Equation (1) by the inverse of the link function. The inverse of the natural log function is the exponent function, giving

$$\lambda_i = \exp(\mu_i) = \exp(a + bX_i)$$

Predicted count (Original) Structural (Transformed by Inverse Link)

(2)

Negative Binomial Model

The Poisson distribution assumes that the mean and variance of the variable are equal. Sometimes count variables do not meet this assumption, especially when there are more zeros or more high values than expected. This is called overdispersion and results in a variable's variance (v) being much larger than its mean (λ). Overdispersion can be incorporated into the GLM regression by estimating the amount of extra variation. One way of doing this is by using a negative binomial (NB) distribution for the residuals. The NB distribution models variance as

$$v = \lambda + \frac{\lambda^2}{\theta}$$

(3)

where θ is an overdispersion parameter (Jay & Peter, 2007).

Zero-Inflated Models

When describing count variables, we stated that it is common for many of the respondents to have never have exhibited the behavior for outcomes that are particularly negative. The resulting variable's distribution has many zeros and just a few other values (Atkins, Baldwin, Zheng, Gallop, & Neighbors, 2013; Atkins & Gallop, 2007). For such cases, there is a class of regression models that can account for the excess zeros, called *zero-inflated models*.

Zero-inflated Poisson (ZIP) and zero-inflated negative binomial (ZINB) regression directly model the excessive number of zeros in the outcome variable. They do this by fitting a mixture model, which combines multiple distributions (Muthén & Shedden, 1999). For ZIP and ZINB models, the outcome variable's distribution is approximated by mixing two models and two distributions. This first model examines if the behavior ever occurred by using a logistic regression. Logistic regression is commonly used to predict a behavior's occurrence, but with ZIP/ZINB models the logistic regression part of the model predicts non-occurrence (i.e., it predicts the zeros). The second model examines how frequently the behavior occurred, using either a Poisson or NB regression. The resulting zero-inflated models produce two sets of coefficients, one predicting if the behavior never occurred (logistic) and the other predicting how frequently the behavior occurred (Poisson or NB). Because mixture models are flexible, the predictors for the two parts of the model can be different. Thus, ZIP and ZINB models are particularly well suited for variables thought to be determined by two different processes—one that influences occurrence and the other that influences the frequency of occurrence.

Parameter Estimation

Typical regression models often use ordinary least squares to estimate the parameters. For count data, the regression model uses maximum likelihood (ML) to estimate the parameters. ML seeks to find values for the regression coefficients that have the highest probability (i.e., maximum likelihood) to have produced the observed data. Enders (2005) provides a particularly readable introduction to ML estimation.

ML usually requires an iterative set of procedures to find the parameter estimates, which can be problematic with models that use many predictor variables or have small sample sizes. If the ML estimation procedure converges, it means it found a unique set of values for each parameter, the combination of which returned the highest likelihood value of all parameter values examined. Thus, it will return parameter estimates, standard errors, and the maximum likelihood value. This likelihood value, or transformations of it, is used to compare the fit of competing models. For more information about the parameter estimation process for count regression, see Cameron and Trivedi (1998).

Parameter Interpretation

In typical regression, the two major parameters to interpret are the intercept and slope/regression coefficients. The intercept is the expected value of the outcome variable when all the predictor variables have a value of zero. Each regression coefficient represents the expected change in the outcome variable for a one-unit change in the predictor variable, holding all the other predictor variables constant.

For Poisson and NB regressions, the two major parameters to interpret are still the intercept and slope/regression coefficients. The interpretation is trickier than with typical regression models because of the log link function, which places the regression coefficients on the natural log scale. Exponentiating the regression coefficients places the predicted values for the outcome on its original scale (cf. Equation 2), but this does not completely solve the interpretation problem. A result of using the log link function is that it forces the continuous predictor variables to have a non-linear relationship with the outcome. Specifically, the Poisson and NB models really specify multiplicative regression models instead of additive ones.

To aid in interpreting Poisson and NB models' coefficients, Atkins and Gallop (2007) recommend different strategies. One strategy is to use the regression equation to generate predictions over a specified range of values of the predictor variables. Specifically, they recommend setting the predictor variables at multiple values of interest and examine how the expected values of the outcome variable change. For the predictor

variables of major interest, they suggest using values across the middle 95% of their values. For predictor variables that are not of major interest (i.e., control variables, covariates) either set their values to the mean (if continuous) or the reference value (if categorical).

A second interpretive strategy is to use the inherent multiplicative nature of the variable's relationships to examine the percentage change in the expected counts, defined as

$$\text{Percentage Change in the Expected Counts} = 100 \times [\exp(b \times \Delta) - 1] \quad (4)$$

where b is the regression coefficient from the Poisson or NB regression, and Δ is the amount of change in the predictor (e.g., for one unit change, $\Delta = 1$).

Because ZIP and ZINB regressions model two separate processes, they produce two sets of coefficients: one for the count part of the model and the other for the logistic part of the model. The coefficients for the count part of the model can be interpreted the same as for a typical Poisson or NB model. The coefficients for the logistic part of zero-inflated models are on the logit scale

$$\text{logit} = \ln\left(\frac{\pi}{1-\pi}\right) \quad (5)$$

where π is the proportion of zeros. A common way of interpreting logistic regression models is to exponentiate the coefficients, which places the coefficients in an odds-ratio scale. An alternative approach is to use the inverse logit function to transform the resulting regression model, which places the outcome on the probability scale:

$$\text{inverse logit} = \frac{\exp\left(\frac{\pi}{1-\pi}\right)}{\exp\left(\frac{\pi}{1-\pi}\right) + 1} \quad (6)$$

Model Comparison

An important aspect of all regression models is to determine how well the model fits the data, either by comparing the actual values with the model-predicted values or by comparing a model to competing models. We demonstrate both approaches in the following example.

In typical regression, R^2 is usually used as the measure of how close the actual values are to the predicted values. While pseudo- R^2 values for count regression models exist, they have the same issues as

Another model-fit measure that penalizes models for complexity is Schwartz's Bayesian information criterion (BIC). While it is not technically related to information theory, it can still be useful in model

Table 1. Variables Used in Count Regression Models (n = 889)

Variable (Name in NELS Dataset)	Description	Values
Skips (<i>F1S10B</i>)	Number of times student cut/skipped class (Outcome variable)	0 = 0 times; 1 = 1-2 times; 2 = 3-6 times; 3 = 7-9 times; 4 = ≥ 10 times
College (<i>F1S51</i>)	Plan on going to college	0 = No; 1 = Yes
Male (<i>BYS12</i>)	Sex	0 = Female; 1 = Male
Race (<i>BYS31A</i>)	Self-described race	0 = White; 1 = Asian; 2 = Hispanic; 3 = Black; 4 = Native American
Achievement (<i>BYTEXCOMP</i>)	Standardized reading and math achievement test composite	Continuous
Self Concept (<i>BYCNCPT1</i>)	Positive self concept, which is a composite of four items	Continuous
SES (<i>BYES</i>)	Socioeconomic status composite	Continuous

Note. Continuous variables were mean centered for all analyses.

pseudo- R^2 for logistic regression, such as not really measuring variance and different formulae producing disparate values. Consequently, since there are a finite number of possible outcome values for count regression models, we examine model-data fit by examining the raw difference between the predicted counts and actual counts at each value of the outcome.

When comparing competing models, information-criterion based fit indices are useful (Burnham & Anderson, 2002). The basic principle of such fit measures is to select the simplest models that can describe the data well (Sherman & Funder, 2009). A commonly used measure from the information-theoretic tradition is Akaike's information criterion (AIC). AIC balances the model's goodness-of-fit to the data and a penalty for model complexity. The general method for using the AIC is to choose the model that has the smallest AIC value. Individual AIC values are not directly interpretable because they contain arbitrary constants and are greatly affected by sample size. These artificial increases in AIC values can sometimes make it appear that multiple models ostensibly appear to have very similar AIC values, even though some models fit the data substantially better than others. The AIC values for a set of models can be transformed so that they sum to the value one, so can be treated like probabilities. These values are called Akaike weights and are typically interpreted as the probability that model a given model is the best model for the data out of all the compared models.

selection. The general method for using the BIC is to choose the model that has the smallest BIC value. With small sample sizes the BIC tends to be overly-conservative (i.e., prefer models with too few variables), but when the sample size is large it tends to select the correct model if a set of competing models includes the true model.

Model Diagnostics. An important part of all regression analyses is to examine residual diagnostics, influential data points, and nonlinearity in the predictors (Andersen, 2012; Belsley, Kuh, & Welsch, 2005). Many of the common tools to assess typical regression models have been extended to count regression, including standardized residuals, influence diagnostics (e.g., Cook's D), and predictor nonlinearity.

In our example, we graphically examined the residuals' distribution and their relation to the predicted values. We used deviance residuals for all the models except zero-inflated, where we used Pearson residuals.

To examine influential observations, we calculated Cook's (1977) D values for each case based on each model. Cook's D is an index that reflects the amount of influence each case has on the model parameter estimates. A common criterion used for identifying cases that could be influential is whether an observation's D value is greater than $4/n$ (Cohen et al., 2003). In addition, to assist with the identification of influential cases we plotted the D values for each observation and inserted a horizontal line at $y = 4/n$.

To examine linearity, we created scatterplots of each predictor variable and the residuals. We looked for a similar distribution of residuals at each level of the predictor variables. Since this requires multiple plots for each model, we only show the results (and R syntax) for the negative binomial model.

Count Regression Example

Data

Data for this example were taken from a subset of the National Education Longitudinal Study of 1988 (NELS), provided by Keith (2006)¹. The variables used for this analysis are given in Table 1. We only used the observations with values for each of the variables.

The outcome is the number of times a student cut/skipped class (skips), placed into one of five categories. A histogram of the skips variable is shown in Figure 7a, and indicates that the variable is not symmetrical, so cannot follow a normal distribution. Thus, because it is a count variable that is distinctly not normally distributed, it is a prime candidate for a count regression model.

For this particular example, we were interested in predicting the number of skips by race, sex, positive self-concept, academic achievement, socioeconomic status (SES), and whether the student plans on going to college. We chose these variables as they represent a mixture of continuous and categorical predictor variables. To make interpretation easier, we mean centered the continuous variables (SES, self-concept, and academic achievement) and dummy-coded the categorical variables.

Typical Regression

As a baseline, we fit a typical regression model to the data, i.e., a model that assumes the residuals follow a normal distribution. Often, these regression parameters are estimated through ordinary least squares (OLS). With normally-distributed residuals, OLS and maximum likelihood (ML) parameter estimates are the same (Kutner, Neter, Nachtsheim, & Li, 2004). For consistency with the other models we fit, we used ML estimation for this model.

The results are shown in the second column of Table 2. The intercept is 0.90. Because of our predictor variable coding mechanisms and centering, the intercept in the model is interpreted as the predicted number of skips for a white female who does not plan to go to college and who had average levels of self-concept, academic achievement, and SES.

The regression coefficients are interpreted as any other unstandardized coefficients from a typical regression. For example, the coefficient associated with going to college is -0.32, indicating that the average number of skips for those who plan on attending college is lower than the average for students not wanting to attend college, after controlling for all the other predictor variables. As another example, the regression coefficient associated with academic achievement is -0.01, meaning that for each one-unit increase in academic achievement, the average skip category decreases by 0.01 units.

The typical regression model assumes that the residuals follow a normal distribution. A plot of the residuals for typical regression model is shown in Figure 4 and clearly shows they do not follow a normal distribution. Another plot that is useful to examine is to compare the residuals to the predicted values. There should be no relationship between these two values, so the LOWESS line should be horizontal and close to zero (for more about LOWESS lines, see Trexler & Travis, 1993). Figure 5 shows plots of the residuals vs. the predicted values. The typical regression shows a

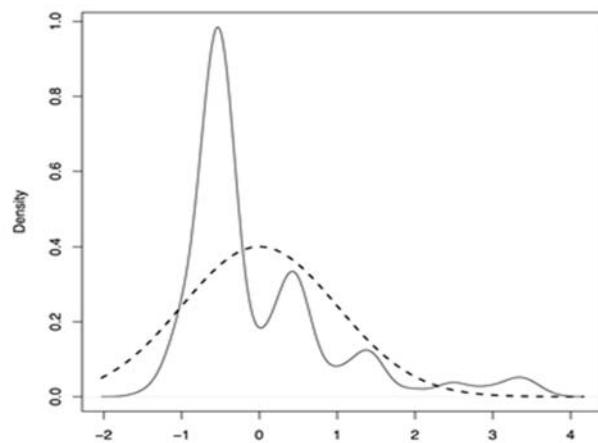


Figure 4. Residuals for typical regression model. A normal distribution is overlaid (dashed line) with the same mean and standard deviation of the residuals

¹ Data can be downloaded from <https://baylor.box.com/countdata>. The datafile's name is *count.dat*. It is a comma-delimited file with a period (.) and 999.98 used as missing value indicators.

Table 2. Comparison of the Regression Models for the School Skip Data (n = 889).

Variable	Typical	Poisson	NB	ZIP		ZINB	
				Count	Logistic	Count	Logistic
Coefficients							
Intercept	0.90	-0.19	-0.20	0.54	0.08	0.39	-0.21
Male	-0.03	-0.04	-0.03	-0.30	-0.65	-0.32	-1.02
Asian ^a	0.05	0.08	0.05	0.42	0.74	0.43	1.00
Hispanic ^a	0.36	0.42	0.41	-0.07	-2.16	0.02	-15.36
Black ^a	-0.06	-0.08	-0.10	-0.28	-0.54	-0.24	-0.65
Native							
American ^a	0.03	0.06	0.04	0.08	0.04	0.08	0.08
College ^b	-0.32	-0.37	-0.37	-0.38	0.01	-0.40	-0.05
Self Concept	-0.03	-0.05	-0.05	0.06	0.29	0.05	0.38
SES	-0.07	-0.11	-0.13	-0.11	-0.01	-0.07	0.13
Achievement	-0.01	-0.01	-0.02	-0.01	0.01	-0.01	0.00
Standard Errors							
Intercept	0.11	0.12	0.16	0.15	0.35	0.20	0.52
Male	0.07	0.09	0.11	0.12	0.30	0.14	0.52
Asian ^a	0.14	0.19	0.23	0.25	0.43	0.30	0.59
Hispanic ^a	0.11	0.12	0.16	0.16	1.35	0.18	939.35
Black ^a	0.12	0.15	0.19	0.23	0.64	0.25	0.98
Native							
American ^a	0.17	0.20	0.26	0.27	0.55	0.31	0.74
College ^b	0.11	0.11	0.15	0.14	0.36	0.16	0.49
Self Concept	0.05	0.06	0.07	0.07	0.18	0.08	0.25
SES	0.05	0.07	0.08	0.10	0.23	0.10	0.29
Achievement	0.00	0.01	0.01	0.01	0.02	0.01	0.03
Likelihood							
Log Likelihood	-1258	-1000	-958	-951		-948	
Model df	11	10	11	20		21	
Fit Measures							
AIC	2537	2021	1939	1942		1938	
AIC Weight	0.00	0.00	0.40	0.06		0.54	
BIC	2590	2069	1992	2038		2039	

Note. Typical: Model assuming normally-distributed residuals, fitted with maximum likelihood estimation.

NB: Negative binomial; ZI: Zero-inflated. AIC: Akaike information criterion; BIC: Bayesian information criterion.

^a Reference category is White.

^b Reference category is not planning on going to college.

slight pattern in the results as the LOWESS line is slanted downward. The other models all have horizontal LOWESS lines, with the negative-binomial model having the lowest range of residual values.

Plots of the Cook's D values for each observation are shown in Figure 6. Overall, the NB model resulted in fewest influential cases (i.e., cases with $D \leq 4/n$), indicating that each observation contributed equitably to the parameter estimates. As measures of model fit, AIC and BIC values are shown in the bottom of Table 2. Figure 7a graphically shows how well the model

predicts the count values by overlaying the predicted probabilities for each skip category on the frequency histogram of the actual skip data. It appears that the typical regression model under-predicts the 0 and 4 skip categories, but over-predicts all the other categories. Figure 7b shows the actual and predicted values numerically, and echoes the over- and under-predictions shown in Figure 7a. Thus, it appears that both model fit and model diagnostics converge in indicating that the typical regression model does not account for the count nature of the skip data very well.

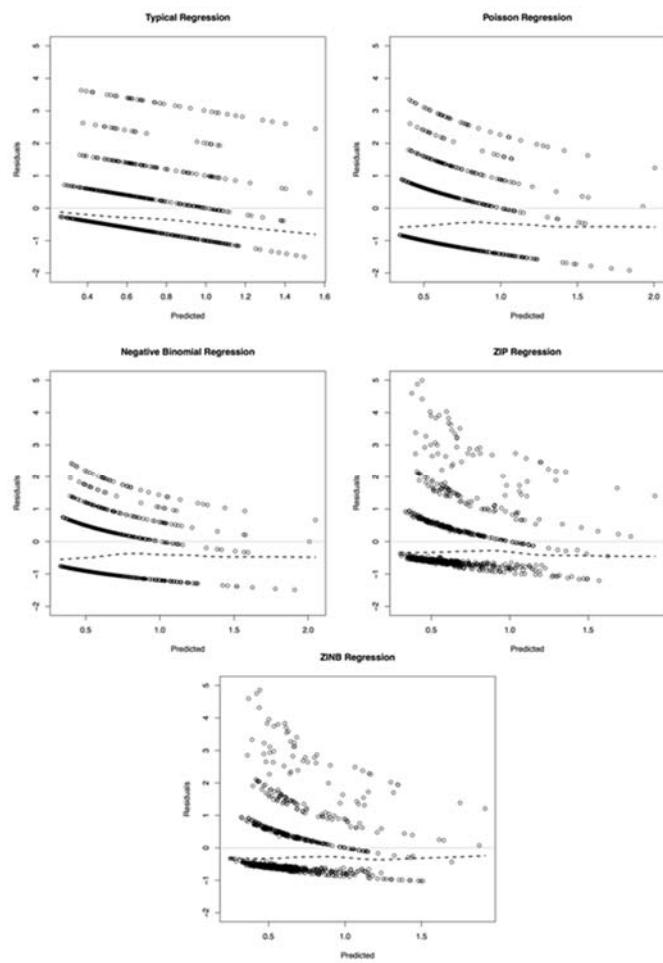


Figure 5. Predicted values vs. residual plots. LOWESS lines are dashed

Poisson Regression

We fit the Poisson regression model using the same predictors of skips we used with the typical regression model. The results from the Poisson regression are shown in the third column of Table 2.

As with the typical regression, the intercept represents the predicted number of skips for a white female who does not plan to go to college and who had average levels of self-concept, academic achievement, and SES. The intercept is -0.19 , but the log link makes this value hard to interpret. This can be remedied by exponentiating the value. For this example, $\exp(-0.19) = 0.82$, indicating that the average skip category for a white female who does not plan to go to college and who had average levels of self-concept, academic

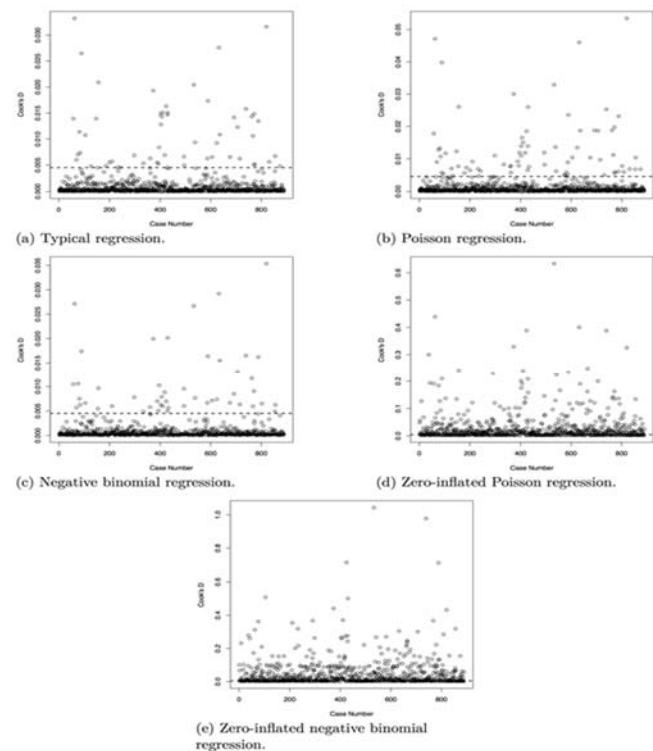
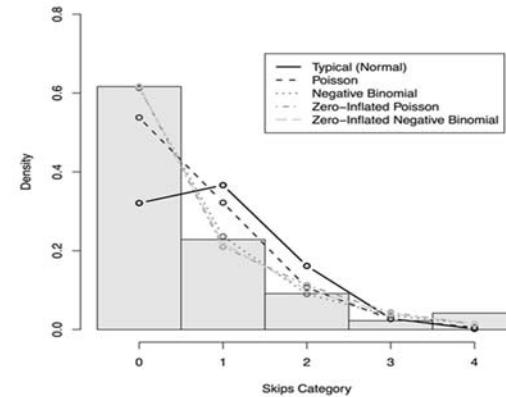


Figure 6. Plots of Cook's D values for each regression model. Threshold is a dashed line



(a) Histogram of school skip categories with overlaid predicted probabilities from each regression model.

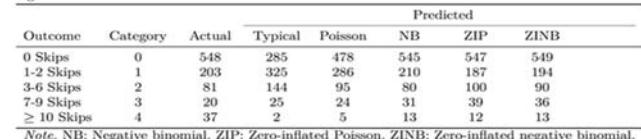


Figure 7. Comparison of actual and predicted category counts

achievement, and SES is 0.82. This is between the 0 and 1-2 skips categories, closer to the latter than the former.

The coefficient associated with wanting to go to college is -0.37 . As wanting to go to college is dummy-coded, the negative sign indicates that the average number of skips for those who want to go to college is smaller than for those who do not want to go to college. Since $\exp(-0.37) = 0.69$, the difference between the two groups is over two-thirds of a skip category.

The method of interpreting dummy-coded categorical variables does not directly extend to continuous predictors. Previously, we discussed Atkins and Gallop's (2007) recommendations for interpreting these variables. As the first is a common way to help interpret any regression model (Cohen et al., 2003), we only focus on the second: percentage change in the expected counts.

The percentage change in the expected counts method of interpretation requires two values: the regression coefficient and the desired amount of change in the variable. For the academic achievement variable, the regression coefficient is -0.01 . For the amount of change we use one SD, which is 9.98 . Plugging those values into Equation (4) produces

$$100 \times [\exp(-0.01 \times 9.98) - 1] = -13.4$$

meaning there is a 13.4% decrease in the expected skip category value for a one SD increase in academic achievement.

To examine model fit, we first compared the AIC and BIC values for the Poisson model to those from the typical regression model (see Table 2). As the values are much smaller for the Poisson regression, this indicates the Poisson model provides an improvement over the typical regression model. Second, we compared the predicted skip category values against the actual values in Figure 7a and Figure 7b. The Poisson model appears to do a much better job capturing the skip data than the typical regression model across all skip value categories.

Negative Binomial Regression

The results from the negative binomial (NB) regression are shown in the fourth column of Table 2. The NB model is very similar to the Poisson model, thus the NB model's coefficients are interpreted in the same way as the Poisson regression. The main difference between the NB and Poisson models is that the NB model allows for more variability (dispersion) in the outcome by not assuming the mean and variance

of the residuals are the same. Consequently, the NB model estimates one extra parameter than the Poisson model: a overdispersion parameter (see Equation 3). The value for overdispersion parameter for the skip data is 1.11. Since the NB and Poisson models are so similar, it is not surprising that the regression coefficients for the two models are very close. The standard errors, however, are larger for the NB model reflecting its larger residual variance.

The plots of the predictor variables against the standardized residuals are shown in Figure 8. Based on visual inspection, we determined that the residual distributions were approximately the same across levels of the predictor variables. We noticed that there were fewer observations in the lower tail of the self-concept distribution, which produces a slightly dissimilar pattern of residuals for that predictor variable. On the whole, the residual patterns across all predictor variables from the NB model were acceptable.

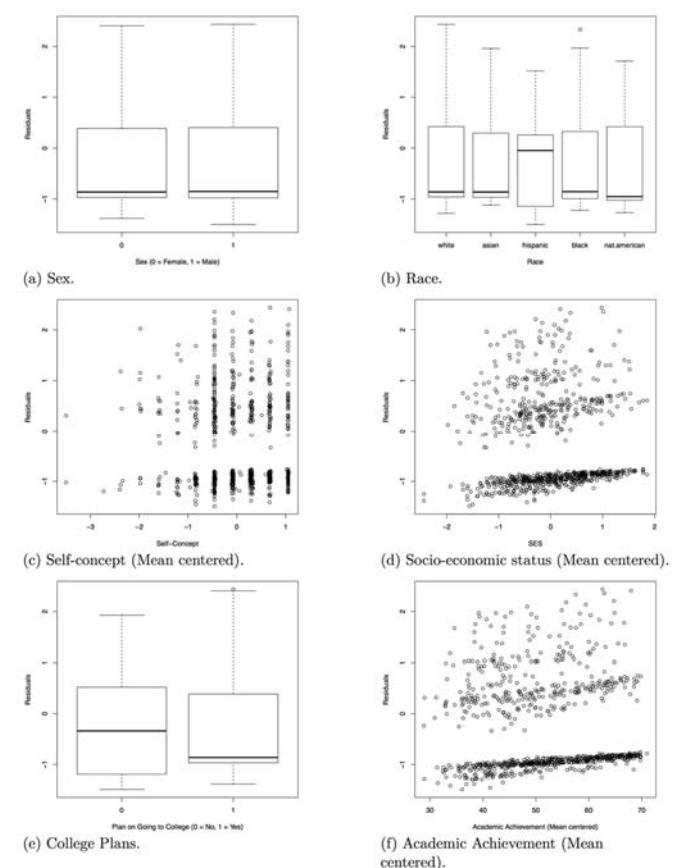


Figure 8. Plots of negative binomial model predictors by residuals.

The AIC and BIC values for NB model are smaller than those for the Poisson model, indicating that the NB model fits the data somewhat better than the Poisson model. Although the amount is not that large, there appears to be enough overdispersion in the skip variable that the Poisson model was not able to capture the variance as well as the NB model. This result is echoed in Figure 7a and Figure 7b, which shows that the NB model provides more accurate predictions than

other variables constant, students planning on going to college typically skip fewer classes than those not planning on going to college, and the difference is approximately the size of two-thirds of a skip category.

Exponentiating the logistic coefficient gives 0.95, indicating that after holding all the other variables constant, students planning on going to college have slightly decreased odds of never having skipped a class

Table 3. Results for Final Count Regression Model (Negative Binomial)

Variable	<i>B</i>	SE	exp(<i>B</i>)	95% Confidence Interval for exp(<i>B</i>)	
				Lower Bound	Upper Bound
Intercept	-0.16	0.14	0.85	0.65	1.13
SES	-0.18	0.08	0.84	0.71	0.98
Achievement	-0.02	0.01	0.99	0.97	1.00
College ^a	-0.37	0.15	0.69	0.51	0.93

Note. B: Unstandardized coefficient; SE: Standard error; exp(B): Exponentiated regression coefficient. Log Likelihood: -963 (df = 5); AIC: 1935; BIC: 1959.

^a. Reference category is not planning on going to college.

the Poisson regression model for all skip categories except category three (7-9 skips).

Zero-Inflated Regression

The results from the zero-inflated models are shown in Table 2. Each zero-inflated model has two sets of regression coefficients. Thus, the zero-inflated Poisson (ZIP) regression values are shown in columns five and six, while the values from the zero-inflated negative binomial (ZINB) regression models are shown in columns seven and eight.

For each zero-inflated model's set of coefficients in Table 2, the first column shows the count regression part, while the second column is the logistic regression portion of the model. The coefficients for the count part of the model can be interpreted the same as was done previously for the Poisson model. The coefficients for the logistic regression are on the logit scale, so exponentiating them transforms the values to odds ratios. Remember, with zero-inflated models the logistic part of the model predicts non-occurrence of the outcome.

As an example interpretation, in the ZINB model the coefficients for planning on going to college are -0.40 and -0.05 for the count and logistic parts of the model, respectively. Exponentiating the count coefficient gives 0.67. Consequently, holding all the

than students not planning on going to college. Alternatively, by using Equation (6) the results indicate that the probability of not having skipped a class for students planning on going to college is 0.01 units lower than for students not planning on going to college. The interpretation of the logistic part of the model must be tempered, however, as an odds ratio of 0.95 represent a small effect and the 95% confidence interval (0.76 - 1.13) contains 1.0 (i.e., no difference). Thus, it is likely that planning on going to college only has an influence on the number of skips, not whether a student has ever skipped a class.

Final M When examining fit across all five of the models, the AIC values and AIC weights favor the ZINB and NB over the other three models, while the BIC favors the NB model. As the ZINB model requires twice as many parameters as the NB model, the NB model is the more parsimonious model. Thus, it appears that accounting for the overdispersion is sufficient to capture the excess number of zero values.

Support for the NB model over the ZINB model is bolstered by Figure 7a and Figure 7b. The NB model does as good of a job as the ZINB model in capturing the first two and last two skip categories, and does a slightly better for the third category (3-6 skips).

All the models fit thus far assume that all the predictor variables are needed. An examination of the values in Table 2 indicates this is likely not the case. Specifically, it appears that only the academic achievement and planning on going to college variables might be needed. Consequently, we pruned the model removing each variable singly and in sets.

The BIC indicted that the model with only the achievement and college plans variables fit the best, while the AIC indicated that achievement, college plans, and SES should be kept. We opted to keep achievement, college plans, and SES in the model. The results for the final, pruned NB binomial model are shown in Table 3.

Discussion

Count outcome variables are very common in many areas of education research. Older traditions of dichotomizing or transforming the outcome variable can produce estimation and interpretive problems. This tutorial introduced methods to analyze count data using the general linear model framework, which is a robust way to handle count outcomes in regression. We discussed four ways to model count data in regression, and then demonstrated the analysis of the data. R syntax for all the analyses is given in the Appendix.

Extensions of Count Regression Models

We ignored two related areas that are growing in this field. The first is including count data in a multilevel framework. Often, data that interests education researchers is multilevel in nature, as when using data from students coming from the same classroom or school (Graves & Frohwerk, 2009). Analyzing such nested data can be tricky when the variables are continuous, much less when they are counts. Nonetheless, recent work in this area has shown how to include count outcomes when the data is nested (Atkins et al., 2013; Gelman & Hill, 2006).

The second is including count data in a repeated measures framework. While this situation could be considered a type of nested data, it is more frequently used with single case designs. We stated in the beginning of this article that Shadish and Sullivan's (2011) single-case design (SCD) review showed that nearly all outcome variables used with SCDs are count variables. As with multilevel data, extending count regression models to SCDs is a difficult endeavor. Nonetheless, there are some promising signs that this,

too, can be included in the educational researcher's data analysis tools (e.g. Shadish, Zuur, & Sullivan, 2014).

References

- Aiken, L. S., West, S. G., & Millsap, R. E. (2008). Doctoral training in statistics, measurement, and methodology in psychology: Replication and extension of Aiken, West, Sechrest, and Reno's (1990) survey of PhD programs in North America. *American Psychologist*, 63, 32-50. doi: 10.1037/0003-066X.63.1.32
- Andersen, R. (2012). Methods for detecting badly behaved data: Distributions, linear models, and beyond. In H. Cooper, P. M. Camic, D. L. Long, A. T. Panter, D. Rindskopf, & K. J. Sher (Eds.), APA handbook of research methods in psychology, Vol 3: Data analysis and research publication (pp. 5-26). Washington, DC: American Psychological Association.
- Atkins, D. C., Baldwin, S., Zheng, C., Gallop, R. J., & Neighbors, C. (2013). A tutorial on count regression and zero-altered count models for longitudinal addictions data. *Psychology of Addictive Behaviors*, 27, 166-177. doi: 10.1037/a0029508
- Atkins, D. C., & Gallop, R. J. (2007). Rethinking how family researchers model infrequent outcomes: A tutorial on count regression and zero-inflated models. *Journal of Family Psychology*, 21, 726-735. doi: 10.1037/0893-3200.21.4.726
- Belsley, D. A., Kuh, E., & Welsch, R. E. (2005). Regression diagnostics: Identifying influential data and sources of collinearity. New York, NY: John Wiley & Sons.
- Box, G. E. P. (1953). Non-normality and tests on variances. *Biometrika*, 40, 318-335. doi: 10.2307/2333350
- Burnham, K. P., & Anderson, D. (2002). Model selection and multimodel inference: A practical information-theoretic approach (2nd ed.). New York: Springer-Verlag.
- Cameron, A. C., & Trivedi, P. K. (1998). Regression analysis of count data. New York, NY: Cambridge University Press.
- Cochran W. G. (1947). Some consequences when the assumptions for the analysis of variance are not satisfied. *Biometrics*, 3, 22-38. doi: 10.2307/3001535
- Cohen, J., Cohen, P., West, S. G., & Aiken, L. S. (2003). Applied multiple regression/correlation analysis for the behavioral sciences (3rd ed.). Mahwah, NJ: Lawrence Erlbaum.
- Cook, R. D. (1977). Detection of influential observation in linear regression. *Technometrics*, 19, 15-18.

- Coxe, S., West, S. G., & Aiken, L. S. (2009). The analysis of count data: A gentle introduction to poisson regression and its alternatives. *Journal of Personality Assessment*, 91, 121-136. doi: 10.1080/00223890802634175
- Enders, C. K. (2005). Maximum likelihood estimation. In B. Everitt & D. C. Howell (Eds.), *Encyclopedia of behavioral statistics* (pp. 1164-1170). West Sussex, England: Wiley.
- Gelman, A., & Hill, J. (2006). Data analysis using regression and multilevel/hierarchical models. New York, NY: Cambridge.
- Graves, J., Scott L., & Frohwert, A. (2009). Multilevel modeling and school psychology: A review and practical example. *School Psychology Quarterly*, 24, 84-94. doi: 10.1037/a0016160
- Hilbe, J. M. (2014). Modeling count data. New York, NY: Cambridge University Press.
- Hosmer, D. W., & Lemeshow, S. (2000). Applied logistic regression (2nd ed.). Hoboken, NJ: Wiley.
- Jay, M. V. H., & Peter, L. B. (2007). Quasi-Poisson vs. negative binomial regression: How should we model overdispersed count data? *Ecology*, 88, 2766-2772.
- Keith, T. Z. (2006). Multiple regression and beyond. Boston: Pearson.
- Kutner, M. H., Neter, J., Nachtsheim, C. J., & Li, W. (2004). Applied linear regression models (4th ed.). New York, NY: McGraw-Hill.
- Little, S. G., Akin-Little, A., & Lee, H. B. (2003). Education in statistics and research design in school psychology. *School Psychology International*, 24, 437-448. doi: 10.1177/01430343030244006
- Lix, L. M., Keselman, J. C., & Keselman, H. J. (1996). Consequences of assumption violation revisited: A quantitative review of alternatives to the one-way analysis of variance F test. *Review of Educational Research*, 66, 579-619.
- MacCallum, R. C., Zhang, S., Preacher, K. J., & Rucker, D. D. (2002). On the practice of dichotomization of quantitative variables. *Psychological Methods*, 7, 19-40. doi: 10.1037/1082-989X.7.1.19
- McCullagh, P., & Nelder, J. A. (1989). Generalized linear models (2nd ed.). London, England: Chapman and Hall.
- Muthén, B. O., & Shedden, K. (1999). Finite mixture modeling with mixture outcomes using the EM algorithm. *Biometrics*, 55, 463-469. doi: 10.1111/j.0006-341X.1999.00463.x
- Perham, H. (2010). Quantitative training of doctoral school psychologists: Statistics, measurement, and methodology curriculum. Unpublished master's thesis, Arizona State University, Tempe, AZ.
- R Development Core Team. (2015). R: A language and environment for statistical computing [Computer software]. Vienna, Austria: R Foundation for Statistical Computing.
- Sakia, R. M. (1992). The Box-Cox transformation technique: A review. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 41, 169-178. doi: 10.2307/2348250
- Shadish, W. R., & Sullivan, K. J. (2011). Characteristics of single-case designs used to assess intervention effects in 2008. *Behavior Research Methods*, 43, 971-980. doi: 10.3758/s13428-011-0111-y
- Shadish, W. R., Zuur, A. F., & Sullivan, K. J. (2014). Using generalized additive (mixed) models to analyze single case designs. *Journal of School Psychology*, 52, 149-178. doi: 10.1016/j.jsp.2013.11.004
- Sherman, R. A., & Funder, D. C. (2009). Evaluating correlations in studies of personality and behavior: Beyond the number of significant findings to be expected by chance. *Journal of Research in Personality*, 43, 1053-1063. doi: 10.1016/j.jrp.2009.05.010
- Trexler, J. C., & Travis, J. (1993). Nontraditional regression analyses. *Ecology*, 74, 1629-1637. doi: 10.2307/1939921

Appendix

R Syntax

Import Data

```
# import data
nels.data <- read.table("count.dat", sep=",", na.strings = c(".", "999.98"), header=TRUE)
# subset variables of interest from NELS data
```

```
nels.var <- c("F1S10B", "BYS12", "BYS31A", "BYCNCPT1", "BYES", "BYTXCOMP", "F1S51")
# create new dataset with only variables of interest
count.data <- nels.data[nels.var]
# change names of variables
names(count.data) <- c("skipped", "sex", "race", "self.con1", "ses", "achievement",
                       "F1S51")
# change sex so that female = 0
count.data$male <- ifelse(count.data$sex==2, 0, count.data$sex)
# recode college plans variable to 0 = no, 1= yes,
count.data$college <- count.data$F1S51
count.data$college[count.data$F1S51==1] <- 0
count.data$college[count.data$F1S51==2 | count.data$F1S51==3 | count.data$F1S51==4 |
                   count.data$F1S51==5] <- 1
# make race variable a factor and name the levels
count.data$race <- factor(count.data$race, levels=1:5, labels=c("asian", "hispanic",
                     "black", "white", "nat.american"))
# make white the comparison race group
count.data$race <- relevel(count.data$race, ref = 4)
# create new dataset without missing data
count.data <- na.omit(count.data)
# mean center continuous variables
count.data$self.con1.m <- scale(count.data$self.con1, center = TRUE, scale = FALSE)
count.data$ses.m <- scale(count.data$ses, center = TRUE, scale = FALSE)
count.data$achievement.m <- scale(count.data$achievement, center = TRUE, scale = FALSE)
```

Fit Regression Models

```
# normal theory regression using maximum likelihood
skip.normal <- glm(skipped ~ male + race + college + self.con1.m + ses.m + achievement.m,
                    data = count.data, family = gaussian)
# summary of results
summary(skip.normal)
# poisson regression
skip.pois <- glm(skipped ~ male + race + college + self.con1.m + ses.m + achievement.m,
                  data = count.data, family = poisson)
# summary of results
summary(skip.pois)

# load MASS package
library(MASS)
# negative binomial regression
skip.nb <- glm.nb(skipped ~ male + race + college + self.con1.m + ses.m + achievement.m,
                   data = count.data)
# summary of results
summary(skip.nb)
# overdispersion
summary(skip.nb)$theta

# load pscl package
library(pscl)
# zero-inflated Poisson regression
# the | separates the count model from the logistic model
skip.zip <- zeroinfl(skipped ~ male + race + college + self.con1.m + ses.m +
                      achievement.m | male + race + college + self.con1.m + ses.m + achievement.m, data =
                      count.data, link = "logit", dist = "poisson", trace = TRUE)
# summary of results
summary(skip.zip)

# load pscl package
library(pscl)
```

```
# zero-inflated negative binomial regression
# the | separates the count model from the logistic model
skip.zinb <- zeroinfl(skipped ~ male + race + college + self.con1.m + ses.m +
    achievement.m | male + race + college + self.con1.m + ses.m + achievement.m, data =
    count.data, link = "logit", dist = "negbin", trace = TRUE, EM = FALSE)
# summary of results
summary(skip.zinb)
```

Model Fit

```
# AIC values
AIC(skip.normal)
AIC(skip.pois)
AIC(skip.nb)
AIC(skip.zip)
AIC(skip.zinb)

# AIC weights
compare.models <- list( )
compare.models[[1]] <- skip.normal
compare.models[[2]] <- skip.pois
compare.models[[3]] <- skip.nb
compare.models[[4]] <- skip.zip
compare.models[[5]] <- skip.zinb
compare.names <- c("Typical", "Poisson", "NB", "ZIP", "ZINB")
names(compare.models) <- compare.names
compare.results <- data.frame(models = compare.names)
compare.results$aic.val <- unlist(lapply(compare.models, AIC))
compare.results$aic.delta <- compare.results$aic.val-min(compare.results$aic.val)
compare.results$aic.likelihood <- exp(-0.5* compare.results$aic.delta)
compare.results$aic.weight <-
    compare.results$aic.likelihood/sum(compare.results$aic.likelihood)
# BIC values
AIC(skip.normal, k = log(nrow(count.data)))
AIC(skip.pois, k = log(nrow(count.data)))
AIC(skip.nb, k = log(nrow(count.data)))
AIC(skip.zip, k = log(nrow(count.data)))
AIC(skip.zinb, k = log(nrow(count.data)))

# observed zero counts
# actual
sum(count.data$skip < 1)
# typical
sum(dnorm(0, fitted(skip.normal)))
# poisson
sum(dpois(0, fitted(skip.pois)))
# nb
sum(dnbinom(0, mu = fitted(skip.nb), size = skip.nb$theta))
# zip
sum(predict(skip.zip, type = "prob")[,1])
# zinb
sum(predict(skip.zinb, type = "prob")[,1])

Diagnostics
# normal residuals density plot
plot(density(residuals(skip.normal)))
# histogram plot with fitted probabilities
# predicted values for typical regression
```

```
normal.y.hat <- predict(skip.normal, type = "response")
normal.y <- skip.normal$y
normal.yUnique <- 0:max(normal.y)
normal.nUnique <- length(normal.yUnique)
phat.normal <- matrix(NA, length(normal.y.hat), normal.nUnique)
dimnames(phat.normal) <- list(NULL, normal.yUnique)
for (i in 1:normal.nUnique) {
    phat.normal[, i] <- dnorm(mean = normal.y.hat, sd = sd(residuals(skip.normal)), x =
    normal.yUnique[i])
}
# mean of the normal predicted probabilities for each value of the outcome
phat.normal.mn <- apply(phat.normal, 2, mean)
# probability of observing each value and mean predicted probabilities for
#count regression models
phat.pois <- predprob(skip.pois)
phat.pois.mn <- apply(phat.pois, 2, mean)
phat.nb <- predprob(skip.nb)
phat.nb.mn <- apply(phat.nb, 2, mean)
phat.zip <- predprob(skip.zip)
phat.zip.mn <- apply(phat.zip, 2, mean)
phat.zinb <- predprob(skip.zinb)
phat.zinb.mn <- apply(phat.zinb, 2, mean)
# histogram
hist(count.data$skip, prob = TRUE, col = "gray90", breaks=seq(min(count.data$skip)-0.5,
    max(count.data$skip)+.5, 1), xlab = "Skips Category", ylim=c(0,.8))
# overlay predicted values
lines(x = seq(0, 4, 1), y = phat.normal.mn, type = "b", lwd=2, lty=1, col="black")
lines(x = seq(0, 4, 1), y = phat.pois.mn, type = "b", lwd=2, lty=2, col="gray20")
lines(x = seq(0, 4, 1), y = phat.nb.mn, type = "b", lwd=2, lty=3, col="gray40")
lines(x = seq(0, 4, 1), y = phat.zip.mn, type = "b", lwd=2, lty=4, col="gray60")
lines(x = seq(0, 4, 1), y = phat.zinb.mn, type = "b", lwd=2, lty=5, col="gray80")
# legend
legend(1, 0.7, c("Typical (Normal)", "Poisson", "Negative Binomial", "Zero-Inflated
    Poisson", "Zero-Inflated Negative Binomial"), lty=seq(1:5), col =
    c("black", "gray20", "gray40", "gray60", "gray80"), lwd=2)
# predicted vs. residual plots
# typical
plot(predict(skip.normal, type="response"), residuals(skip.normal), main="Typical
    Regression", ylab="Residuals", xlab="Predicted", ylim=c(-2,5))
abline(h=0,lty=1,col="gray")
lines(lowess(predict(skip.normal,type="response"),residuals(skip.normal)), lwd=2, lty=2)
# poisson
plot(predict(skip.pois,type="response"),residuals(skip.pois), main="Poisson Regression",
    ylab="Residuals", xlab="Predicted", ylim=c(-2,5))
abline(h=0,lty=1,col="gray")
lines(lowess(predict(skip.pois,type="response"),residuals(skip.pois)),lwd=2, lty=2)
# negative binomial
plot(predict(skip.nb,type="response"),residuals(skip.nb), main="Negative Binomial
    Regression", ylab="Residuals", xlab="Predicted", ylim=c(-2,5))
abline(h=0,lty=1,col="gray")
lines(lowess(predict(skip.nb,type="response"),residuals(skip.nb)), lwd=2, lty=2)
# ZIP
plot(predict(skip.zip,type="response"),residuals(skip.zip), main="ZIP Regression",
    ylab="Residuals", xlab="Predicted", ylim=c(-2,5))
abline(h=0,lty=1,col="gray")
lines(lowess(predict(skip.zip,type="response"),residuals(skip.zip)),lwd=2, lty=2)
# ZINB
plot(predict(skip.zinb,type="response"),residuals(skip.zinb), main="ZINB Regression",
    ylab="Residuals", xlab="Predicted", ylim=c(-2,5))
abline(h=0,lty=1,col="gray")
```

```
lines(lowess(predict(skip.zinb,type="response"),residuals(skip.zinb)),lwd=2, lty=2)
```

A Cook's D computation function is built into **R** for the typical, Poisson, and negative binomial regression models, but not zero-inflated models. Consequently, we wrote an iterative function to compute the D values for each case in the zero-inflated models.

```
# plot Cook's D for the typical regression
plot(cooks.distance(skip.normal), main="Cook's D Estimates", ylab="Cook's D",
      xlab="Observation")
abline(h=(4/nrow(count.data)), col="red", lwd=2)

# plot Cook's D for the Poisson model
plot(cooks.distance(skip.pois), main="Cook's D Estimates", ylab="Cook's D",
      xlab="Observation")

# plot Cook's D for the negative binomial model
plot(cooks.distance(skip.nb), main="Cook's D Estimates", ylab="Cook's D",
      xlab="Observation")
abline(h=(4/nrow(count.data)), col="red", lwd=2)

# compute generalized Cook's D for zero-inflated models
g.cooks.zi<-function(model){
  n <- nrow(count.data)
  cooks <- as.matrix(rep(0,nrow(count.data)))
  for (i in 1:n){
    if(model=="ZIP"){
      skip.zip.red <- zeroinfl(skipped ~ male + race + self.conl.m + ses.m +
        achievement.m + college | male + race + self.conl.m + ses.m + achievement.m +
        college, data = count.data[-i,],
      link = "logit", dist = "poisson", trace = TRUE)
      cooks[i]<-t(rbind(as.matrix(skip.zip.red$coefficients$count),
        as.matrix(skip.zip.red$coefficients$zero))-
        rbind(as.matrix(skip.zip$coefficients$count),
        as.matrix(skip.zip$coefficients$zero)))%*%
        (-skip.zip$optim$hessian))%*%(rbind(
        as.matrix(skip.zip.red$coefficients$count),
        as.matrix(skip.zip.red$coefficients$zero))-
        rbind(as.matrix(skip.zip$coefficients$count),
        as.matrix(skip.zip$coefficients$zero)))
    }
    if(model=="NB"){
      skip.zinb.red <- zeroinfl(skipped ~ male + race + self.conl.m + ses.m +
        achievement.m + college | male + race + self.conl.m + ses.m + achievement.m +
        college, data = count.data[-i,], link = "logit", dist = "negbin", trace = TRUE, EM =
        FALSE)
      cooks[i]<-t(rbind(as.matrix(skip.zinb.red$coefficients$count),
        as.matrix(skip.zinb.red$coefficients$zero),
        as.matrix(skip.zinb.red$theta))-
        rbind(as.matrix(skip.zinb$coefficients$count),
        as.matrix(skip.zinb$coefficients$zero),
        as.matrix(skip.zinb$theta)))%*%
        (-skip.zinb$optim$hessian))%*%(rbind(
        as.matrix(skip.zinb.red$coefficients$count),
        as.matrix(skip.zinb.red$coefficients$zero))-
        as.matrix(skip.zinb.red$theta))-
        rbind(as.matrix(skip.zinb$coefficients$count),
        as.matrix(skip.zinb$coefficients$zero),
        as.matrix(skip.zinb$theta)))
    }
  }
}
```

```
    return(cooks)
}

# generate and plot Cook's D for the zero-inflated Poisson model
cooks.out<-g.cooks.zi(model="ZIP")
plot(cooks.out ,xlab="Case Number", ylab="Cook's D")
abline(h=(4/nrow(count.data)), col="red")

# generate and plot Cook's D for the zero-inflated negative binomial model
cooks.out <- g.cooks.zi(model="ZINB")
plot(cooks.out ,xlab="Case Number", ylab="Cook's D")
abline(h=(4/nrow(count.data)), col="red")

# linearity plots for negative binomial model
plot(as.factor(count.data$male),resid(skip.nb),xlab="Sex (0 = Female, 1 = Male)" ,
     ylab="Residuals")
plot(count.data$race,resid(skip.nb),xlab="Race",ylab="Residuals")
plot(count.data$self.con1.m,resid(skip.nb),xlab="Self-concept", ylab="Residuals")
plot(count.data$ses.m,resid(skip.nb),xlab="SES", ylab="Residuals")
plot(as.factor(count.data$college),resid(skip.nb),xlab="Plan on Going to College (0 = No,
      1 = Yes)", ylab="Residuals")
plot(count.data$achievement.m,resid(skip.nb),xlab="Academic Achievement",
      ylab="Residuals")
```

Select Final Model

```
# define the NB models to compare
cand.models <- list( )
cand.models[[1]] <- glm.nb(skipped ~ male + race + college + self.con1.m + ses.m +
    achievement.m, data = count.data)
cand.models[[2]] <- glm.nb(skipped ~ male + race + college + self.con1.m +
    achievement.m, data = count.data)
cand.models[[3]] <- glm.nb(skipped ~ male + race + college + ses.m + achievement.m, data =
    count.data)
cand.models[[4]] <- glm.nb(skipped ~ male + race + self.con1.m + ses.m + achievement.m,
    data = count.data)
cand.models[[5]] <- glm.nb(skipped ~ male + college + self.con1.m + ses.m +
    achievement.m, data = count.data)
cand.models[[6]] <- glm.nb(skipped ~ race + college + self.con1.m + ses.m +
    achievement.m, data = count.data)
cand.models[[7]] <- glm.nb(skipped ~ male + race + college + self.con1.m + ses.m, data =
    count.data)
cand.models[[8]] <- glm.nb(skipped ~ race + college + ses.m + achievement.m, data =
    count.data)
cand.models[[9]] <- glm.nb(skipped ~ college + ses.m + achievement.m, data =
    count.data)
cand.models[[10]] <- glm.nb(skipped ~ college + achievement.m, data = count.data)
cand.models[[11]] <- glm.nb(skipped ~ college , data = count.data)

# name the models
model.names <- c("Full", "SES", "SlfCon", "College", "Race", "Sex", "Ach",
    "Sex.SlfCon", "Sex.SlfCon.Race", "Sex.SlfCon.Race.SES", "Sex.SlfCon.Race.SES.Ach")
names(cand.models) <- model.names

# calculate and combine AIC, AIC weights, and BIC
results <- data.frame(models = model.names)
results$bic.val <- unlist(lapply(cand.models, BIC))
results$bic.rank <- rank(results$bic.val)
results$aic.val <- unlist(lapply(cand.models, AIC))
results$aic.delta <- results$aic.val-min(results$aic.val)
```

```
results$aic.likelihood <- exp(-0.5 * results$aic.delta)
results$aic.weight <- results$aic.likelihood/sum(results$aic.likelihood)
# sort models by AIC weight
results <- results[rev(order(results[, "aic.weight"])),]
results$cum.aic.weight <- cumsum(results[, "aic.weight"])

# final model
skip.final.nb <- glm.nb(skipped ~ college + ses.m + achievement.m, data = count.data)
```

Citation:

Beaujean, A. Alexander, Grant, Morgan B. (2016). Tutorial on Using Regression Models with Count Outcomes using R. *Practical Assessment, Research & Evaluation*, 21(2). Available online:
<http://pareonline.net/getvn.asp?v=21&n=2>

Corresponding Author:

A. Alexander Beaujean
Department of Educational Psychology
One Bear Place #97301
Waco TX 76798-7301

email: Alex_Beaujean [at] baylor.edu

VAR, SVAR and SVEC Models: Implementation Within R Package vars

Bernhard Pfaff

Kronberg im Taunus

Abstract

The structure of the package **vars** and its implementation of vector autoregressive-, structural vector autoregressive- and structural vector error correction models are explained in this paper. In addition to the three cornerstone functions **VAR()**, **SVAR()** and **SVEC()** for estimating such models, functions for diagnostic testing, estimation of a restricted models, prediction, causality analysis, impulse response analysis and forecast error variance decomposition are provided too. It is further possible to convert vector error correction models into their level VAR representation. The different methods and functions are elucidated by employing a macroeconomic data set for Canada. However, the focus in this writing is on the implementation part rather than the usage of the tools at hand.

Keywords: vector autoregressive models, structural vector autoregressive models, structural vector error correction models, R, **vars**.

1. Introduction

Since the critique of [Sims \(1980\)](#) in the early eighties of the last century, multivariate data analysis in the context of vector autoregressive models (henceforth: VAR) has evolved as a standard instrument in econometrics.¹ Because statistical tests are frequently used in determining inter-dependencies and dynamic relationships between variables, this methodology was soon enriched by incorporating non-statistical a priori information. VAR models explain the endogenous variables solely by their own history, apart from deterministic regressors. In contrast, structural vector autoregressive models (henceforth: SVAR) allow the explicit modeling of contemporaneous interdependence between the left-hand side variables. Hence, these types of models try to bypass the shortcomings of VAR models. At the same time as Sims jeopardized the paradigm of multiple structural equation models laid out by the Cowles Foundation in the 1940s and 1950s, [Granger \(1981\)](#) and [Engle and Granger \(1987\)](#) endowed econometricians with a powerful tool for modeling and testing economic relationships, namely, the concept of cointegration. Nowadays these branches of research are unified in the form of vector error correction (henceforth: VECM) and structural vector error correction models (henceforth: SVEC). A thorough theoretical exposition of all these models is provided in the monographs of [Lütkepohl \(2006\)](#), [Hendry \(1995\)](#), [Johansen \(1995\)](#), [Hamilton \(1994\)](#), [Banerjee, Dolado, Galbraith, and Hendry \(1993\)](#).

To the author's knowledge, currently only functions in the base distribution of R and in

¹This vignette has been publicized as an article in the Journal of Statistical Software (see [Pfaff 2008](#)).

the CRAN (Comprehensive R Archive Network) packages **dse** (Gilbert 2000, 1995, 1993) and **fArma** (Würtz 2007) are made available for estimating ARIMA and VARIMA time series models. Although the CRAN package **MSBVAR** (Brandt and Appleby 2007) provides methods for estimating frequentist and Bayesian vector autoregression (BVAR) models, the methods and functions provided in the package **vars** try to fill a gap in the econometrics' methods landscape of R by providing the "standard" tools in the context of VAR, SVAR and SVEC analysis.

This article is structured as follows: in the next section the considered models, i.e., VAR, SVAR, VECM and SVEC, are presented. The structure of the package as well as the implemented methods and functions are explained in Section 3. In the last part, examples of applying the tools contained in **vars** are exhibited. Finally, a summary and a computational details section conclude this article.

2. The considered models

2.1. Vector autoregressive models

In its basic form, a VAR consists of a set of K endogenous variables $\mathbf{y}_t = (y_{1t}, \dots, y_{kt}, \dots, y_{Kt})$ for $k = 1, \dots, K$. The VAR(p)-process is then defined as:²

$$\mathbf{y}_t = A_1 \mathbf{y}_{t-1} + \dots + A_p \mathbf{y}_{t-p} + \mathbf{u}_t , \quad (1)$$

with A_i are $(K \times K)$ coefficient matrices for $i = 1, \dots, p$ and \mathbf{u}_t is a K -dimensional process with $E(\mathbf{u}_t) = \mathbf{0}$ and time invariant positive definite covariance matrix $E(\mathbf{u}_t \mathbf{u}_t^\top) = \Sigma_u$ (white noise).

One important characteristic of a VAR(p)-process is its stability. This means that it generates stationary time series with time invariant means, variances and covariance structure, given sufficient starting values. One can check this by evaluating the characteristic polynomial:

$$\det(I_K - A_1 z - \dots - A_p z^p) \neq 0 \quad \text{for } |z| \leq 1 . \quad (2)$$

If the solution of the above equation has a root for $z = 1$, then either some or all variables in the VAR(p)-process are integrated of order one, i.e., $I(1)$. It might be the case, that cointegration between the variables does exist. This instance can then be better analyzed in the context of a VECM.

In practice, the stability of an empirical VAR(p)-process can be analyzed by considering the companion form and calculating the eigenvalues of the coefficient matrix. A VAR(p)-process can be written as a VAR(1)-process:

$$\boldsymbol{\xi}_t = A \boldsymbol{\xi}_{t-1} + \mathbf{v}_t , \quad (3)$$

²Without loss of generality, deterministic regressors are suppressed in the following notation. Furthermore, vectors are assigned by small bold letters and matrices by capital letters. Scalars are written out as small letters, which are possibly sub-scripted.

with:

$$\boldsymbol{\xi}_t = \begin{bmatrix} \mathbf{y}_t \\ \vdots \\ \mathbf{y}_{t-p+1} \end{bmatrix}, A = \begin{bmatrix} A_1 & A_2 & \cdots & A_{p-1} & A_p \\ I & 0 & \cdots & 0 & 0 \\ 0 & I & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & I & 0 \end{bmatrix}, \mathbf{v}_t = \begin{bmatrix} \mathbf{u}_t \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}, \quad (4)$$

whereby the dimensions of the stacked vectors $\boldsymbol{\xi}_t$ and \mathbf{v}_t is $(KP \times 1)$ and the dimension of the matrix A is $(Kp \times Kp)$. If the moduli of the eigenvalues of A are less than one, then the VAR(p)-process is stable.

For a given sample of the endogenous variables $\mathbf{y}_1, \dots, \mathbf{y}_T$ and sufficient presample values $\mathbf{y}_{-p+1}, \dots, \mathbf{y}_0$, the coefficients of a VAR(p)-process can be estimated efficiently by least-squares applied separately to each of the equations.

Once a VAR(p) model has been estimated, the avenue is wide open for further analysis. A researcher might/should be interested in diagnostic tests, such as testing for the absence of autocorrelation, heteroscedasticity or non-normality in the error process. He might be interested further in causal inference, forecasting and/or diagnosing the empirical model's dynamic behavior, i.e., impulse response functions (henceforth: IRF) and forecast error variance decomposition (henceforth: FEVD). The latter two are based upon the Wold moving average decomposition for stable VAR(p)-processes which is defined as:

$$\mathbf{y}_t = \Phi_0 \mathbf{u}_t + \Phi_1 \mathbf{u}_{t-1} + \Phi_2 \mathbf{u}_{t-2} + \dots, \quad (5)$$

with $\Phi_0 = I_K$ and Φ_s can be computed recursively according to:

$$\Phi_s = \sum_{j=1}^s \Phi_{s-j} A_j \text{ for } s = 1, 2, \dots, , \quad (6)$$

whereby $A_j = 0$ for $j > p$.

Finally, forecasts for horizons $h \geq 1$ of an empirical VAR(p)-process can be generated recursively according to:

$$\mathbf{y}_{T+h|T} = A_1 \mathbf{y}_{T+h-1|T} + \dots + A_p \mathbf{y}_{T+h-p|T}, \quad (7)$$

where $\mathbf{y}_{T+j|T} = \mathbf{y}_{T+j}$ for $j \leq 0$. The forecast error covariance matrix is given as:

$$Cov \left(\begin{bmatrix} \mathbf{y}_{T+1} - \mathbf{y}_{T+1|T} \\ \vdots \\ \mathbf{y}_{T+h} - \mathbf{y}_{T+h|T} \end{bmatrix} \right) = \begin{bmatrix} I & 0 & \cdots & 0 \\ \Phi_1 & I & \cdots & 0 \\ \vdots & & \ddots & 0 \\ \Phi_{h-1} & \Phi_{h-2} & \cdots & I \end{bmatrix} (\Sigma_{\mathbf{u}} \otimes I_h) \begin{bmatrix} I & 0 & \cdots & 0 \\ \Phi_1 & I & \cdots & 0 \\ \vdots & & \ddots & 0 \\ \Phi_{h-1} & \Phi_{h-2} & \cdots & I \end{bmatrix}^{\top}$$

and the matrices Φ_i are the empirical coefficient matrices of the Wold moving average representation of a stable VAR(p)-process as shown above. The operator \otimes is the Kronecker product.

2.2. Structural vector autoregressive models

Recall from Section 2.1 the definition of a VAR(p)-process, in particular Equation 1. A VAR(p) can be interpreted as a reduced form model. A SVAR model is its structural form and is defined as:

$$A\mathbf{y}_t = A_1^*\mathbf{y}_{t-1} + \dots + A_p^*\mathbf{y}_{t-p} + B\boldsymbol{\varepsilon}_t . \quad (8)$$

It is assumed that the structural errors, $\boldsymbol{\varepsilon}_t$, are white noise and the coefficient matrices A_i^* for $i = 1, \dots, p$, are structural coefficients that differ in general from their reduced form counterparts. To see this, consider the resulting equation by left-multiplying Equation 8 with the inverse of A :

$$\begin{aligned} \mathbf{y}_t &= A^{-1}A_1^*\mathbf{y}_{t-1} + \dots + A^{-1}A_p^*\mathbf{y}_{t-p} + A^{-1}B\boldsymbol{\varepsilon}_t \\ \mathbf{y}_t &= A_1\mathbf{y}_{t-1} + \dots + A_p\mathbf{y}_{t-p} + \mathbf{u}_t . \end{aligned} \quad (9)$$

A SVAR model can be used to identify shocks and trace these out by employing IRA and/or FEVD through imposing restrictions on the matrices A and/or B . Incidentally, though a SVAR model is a structural model, it departs from a reduced form VAR(p) model and only restrictions for A and B can be added. It should be noted that the reduced form residuals can be retrieved from a SVAR model by $\mathbf{u}_t = A^{-1}B\boldsymbol{\varepsilon}_t$ and its variance-covariance matrix by $\Sigma_{\mathbf{u}} = A^{-1}BB^\top A^{-1}$.

Depending on the imposed restrictions, three types of SVAR models can be distinguished:

- **A model:** B is set to I_K
(minimum number of restrictions for identification is $K(K - 1)/2$).
- **B model:** A is set to I_K
(minimum number of restrictions to be imposed for identification is the same as for A model).
- **AB model:** restrictions can be placed on both matrices
(minimum number of restrictions for identification is $K^2 + K(K - 1)/2$).

The parameters are estimated by minimizing the negative of the concentrated log-likelihood function:

$$\begin{aligned} \ln L_c(A, B) &= -\frac{KT}{2} \ln(2\pi) + \frac{T}{2} \ln |A|^2 - \frac{T}{2} \ln |B|^2 \\ &\quad - \frac{T}{2} \text{tr}(A^\top B^{-1} B^{-1} A \tilde{\Sigma}_u) , \end{aligned} \quad (10)$$

whereby $\tilde{\Sigma}_u$ signifies an estimate of the reduced form variance/covariance matrix for the error process.

2.3. Vector error correction models

Reconsider the VAR from Equation 1:

$$\mathbf{y}_t = A_1\mathbf{y}_{t-1} + \dots + A_p\mathbf{y}_{t-p} + \mathbf{u}_t , \quad (11)$$

The following vector error correction specifications do exist, which can be estimated with function `ca.jo()` contained in `urca` for more details (Pfaff 2006):

$$\Delta\mathbf{y}_t = \boldsymbol{\alpha}\boldsymbol{\beta}^\top \mathbf{y}_{t-p} + \Gamma_1\Delta\mathbf{y}_{t-1} + \dots + \Gamma_{p-1}\Delta\mathbf{y}_{t-p+1} + \mathbf{u}_t , \quad (12)$$

with

$$\Gamma_i = -(I - A_1 - \dots - A_i), \quad i = 1, \dots, p-1. \quad (13)$$

and

$$\Pi = \boldsymbol{\alpha} \boldsymbol{\beta}^\top = -(I - A_1 - \dots - A_p) \quad . \quad (14)$$

The Γ_i matrices contain the cumulative long-run impacts, hence this VECM specification is signified by “long-run” form. The other specification is given as follows and is commonly employed:

$$\Delta \mathbf{y}_t = \boldsymbol{\alpha} \boldsymbol{\beta}^\top \mathbf{y}_{t-1} + \Gamma_1 \Delta \mathbf{y}_{t-1} + \dots + \Gamma_{p-1} \mathbf{y}_{t-p+1} + \mathbf{u}_t \quad , \quad (15)$$

with

$$\Gamma_i = -(A_{i+1} + \dots + A_p) \quad i = 1, \dots, p-1. \quad (16)$$

Equation 14 applies to this specification too. Hence, the Π matrix is the same as in the first specification. However, the Γ_i matrices now differ, in the sense that they measure transitory effects. Therefore this specification is signified as “transitory” form. In case of cointegration the matrix $\Pi = \boldsymbol{\alpha} \boldsymbol{\beta}^\top$ is of reduced rank. The dimensions of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ is $K \times r$ and r is the cointegration rank, i.e. how many long-run relationships between the variables \mathbf{y}_t do exist. The matrix $\boldsymbol{\alpha}$ is the loading matrix and the coefficients of the long-run relationships are contained in $\boldsymbol{\beta}$.

2.4. Structural vector error correction models

Reconsider the VECM from Equation 15. It is possible to apply the same reasoning of SVAR models to SVEC models, in particular when the equivalent level-VAR representation of the VECM is used. However, the information contained in the cointegration properties of the variables are thereby not used for identifying restrictions on the structural shocks. Hence, typically a B model is assumed whence a SVEC model is specified and estimated.

$$\Delta \mathbf{y}_t = \boldsymbol{\alpha} \boldsymbol{\beta}^\top \mathbf{y}_{t-1} + \Gamma_1 \Delta \mathbf{y}_{t-1} + \dots + \Gamma_{p-1} \mathbf{y}_{t-p+1} + B \boldsymbol{\varepsilon}_t \quad , \quad (17)$$

whereby $\mathbf{u}_t = B \boldsymbol{\varepsilon}_t$ and $\boldsymbol{\varepsilon}_t \sim N(\mathbf{0}, I_K)$. In order to exploit this information, one considers the Beveridge-Nelson moving average representation of the variables \mathbf{y}_t if they adhere to the VECM process as in Equation 15:

$$\mathbf{y}_t = \Xi \sum_{i=1}^t \mathbf{u}_i + \sum_{j=0}^{\infty} \Xi_j^* \mathbf{u}_{t-j} + \mathbf{y}_0^* \quad . \quad (18)$$

The variables contained in \mathbf{y}_t can be decomposed into a part that is integrated of order one and a part that is integrated of order zero. The first term on the right-hand-side of Equation 18 is referred to the “common trends” of the system and this term drives the system \mathbf{y}_t . The middle term is integrated of order zero and it is assumed that the infinite sum is bounded, i.e. Ξ_j^* converge to zero as $j \rightarrow \infty$. The initial values are captured by \mathbf{y}_0^* . For the modeling of SVEC the interest centers on the common trends in which the long-run effects of shocks are captured. The matrix Ξ is of reduced rank $K - r$, whereby r is the count of stationary cointegration relationships. The matrix is defined as:

$$\Xi = \boldsymbol{\beta}_\perp \left[\boldsymbol{\alpha}_\perp^\top \left(I_K - \sum_{i=1}^{p-1} \Gamma_i \right) \boldsymbol{\beta}_\perp \right]^{-1} \boldsymbol{\alpha}_\perp^\top . \quad (19)$$

Function or method	Class	Methods for class	Functions for class
VAR	varest	coef, fevd, fitted, irf, logLik, Phi, plot, predict, print, Psi, resid, summary	Acoef, arch.test, Bcoef, BQ, causality, normality.test, restrict, roots, serial.test, stability
SVAR	svarest	fevd, irf, logLik, Phi, print, summary	
SVEC	sveccest	fevd, irf, logLik, Phi, print, summary	
vec2var	vec2var	fevd, fitted, irf, logLik, Phi, predict, print, Psi, resid	arch.test, normality.test, serial.test
fevd	varfevd	plot, print	
irf	varirf	plot, print	
predict	varprd	plot, print	fanchart
summary	varsum, svarsum, svecsum	print	
arch.test	varcheck	plot, print	
normality.test	varcheck	plot, print	
serial.test	varcheck	plot, print	
stability	varstabil	plot, print	

Table 1: Structure of package **vars**.

Because of its reduced rank only $K - r$ common trends drive the system. Therefore, by knowing the rank of Π one can then conclude that at most r of the structural errors can have a transitory effect. This implies that at most r columns of Ξ can be set to zero. One can combine the Beveridge-Nelson decomposition with the relationship between the VECM error terms and the structural innovations. The common trends term is then $\Xi B \sum_{t=1}^{\infty} \varepsilon_t$ and the long-run effects of the structural innovations are captured by the matrix ΞB . The contemporaneous effects of the structural errors are contained in the matrix B . As in the case of SVAR models of type B one needs for local just-identified SVEC models $\frac{1}{2}K(K - 1)$ restrictions. The cointegration structure of the model provides $r(K - r)$ restrictions on the long-run matrix. The remaining restrictions can be placed on either matrix, whereby at least $r(r - 1)/2$ of them must be imposed directly on the contemporaneous matrix B .

3. Classes, methods, and functions

3.1. Overview

In Table 1 the structure of the package **vars** is outlined. The functions and methods will be addressed briefly here. A more detailed discussion is provided in the following subsections.

When a $\text{VAR}(p)$ has been fitted with `VAR()` one obtains a list object with class attribute `varest`.

```
VAR(y, p = 1, type = c("const", "trend", "both", "none"),
    season = NULL, exogen = NULL, lag.max = NULL,
    ic = c("AIC", "HQ", "SC", "FPE"))
```

As one can see from Table 1, basically all relevant investigations can be conducted with the methods and functions made available for this kind of object. Plotting, prediction, forecast error variance decomposition, impulse response analysis, log-likelihood, MA representations and summary are implemented as methods. Diagnostic testing, restricted VAR estimation, stability analysis in the sense of root-stability and empirical fluctuation processes as well as causality analysis are implemented as functions. Some of the methods have their own `print` and `plot` methods. Furthermore, extractor methods for obtaining the residuals, the fitted values and the estimated coefficients do exist.

In Section 2.2 it has been argued that a VAR can be viewed as particular SVAR model. The function `SVAR()` requires therefore an object of class `varest`. The default estimation method of a SVAR model is a scoring algorithm, as proposed by [Amisano and Giannini \(1997\)](#). The restrictions for the A , B or A and B matrices have to be provided in explicit form as arguments `Amat` and/or `Bmat`. Alternatively, the different SVAR types can be estimated by directly minimizing the negative log-likelihood. The latter is used if the estimation method has then to be set to `estmethod = "direct"`.

```
SVAR(x, estmethod = c("scoring", "direct"), Amat = NULL, Bmat = NULL,
      start = NULL, max.iter = 100, conv.crit = 1e-07, maxls = 1,
      lrtest = TRUE, ...)
```

For objects with class attribute `svarest` forecast error variance decomposition, impulse response analysis, retrieval of its MA representation, the value of the log-likelihood as well as a summary are available as methods.

Structural vector error correction models can be estimated with the function `SVEC()`.

```
SVEC(x, LR = NULL, SR = NULL, r = 1, start = NULL, max.iter = 100,
      conv.crit = 1e-07, maxls = 1, lrtest = TRUE, boot = FALSE, runs = 100)
```

The returned object has class attribute `svecest`. The same methods that are available for objects of class `svarest` are at hand for these kind of objects.

Finally, objects of formal class `ca.jo` generated with function `ca.jo()` contained in the package `urca` can be converted to their level VAR representation with the function `vec2var()`. The resultant object has class attribute `vec2var` and the analytical tools are likewise applicable as in the case of objects with class attribute `varest`.

```
vec2var(z, r = 1)
```

3.2. Cornerstone functions

The function for estimating a $\text{VAR}(p)$ model is **VAR()**. It consists of seven arguments. A data matrix or an object that can be coerced to it has to be provided for **y**. The lag-order has to be submitted as integer for **p**. In general, this lag-order is unknown and **VAR()** offers the possibility to automatically determine an appropriate lag inclusion. This is achieved by setting **lag.max** to an upper bound integer value and **ic** to a desired information criteria. Possible options are Akaike (**ic = "AIC"**), Hannan-Quinn (**ic = "HQ"**), Schwarz (**ic = "SC"**) or the forecast prediction error (**ic = "FPE"**). The calculations are based upon the same sample size. That is **lag.max** values are used for each of the estimated models as starting values.³ The type of deterministic regressors to include into the $\text{VAR}(p)$ is set by the argument **type**. Possible values are to include a constant, a trend, both or none deterministic regressors. In addition, the inclusion of centered seasonal dummy variables can be achieved by setting **season** to the seasonal frequency of the data (e.g., for quarterly data: **season = 4**). Finally, exogenous variables, like intervention dummies, can be included by providing a matrix object for **exogen**.

The **summary** method returns – aside of descriptive information about the estimated VAR – the estimated equations as well as the variance/covariance and the correlation matrix of the residuals. It is further possible to report summary results for selected equations only. This is achieved by using the function's argument **equation** which expects a character vector with the names of the desired endogenous variables. The **plot** method displays for each equation in a VAR a diagram of fit, a residual plot, the auto-correlation and partial auto-correlation function of the residuals in a single layout. If the **plot** method is called interactively, the user is requested to enter <RETURN> for commencing to the next plot. It is further possible to plot the results for a subset of endogenous variables only. This is achieved by using the argument **name** in the **plot** method. The appearance of the plot can be adjusted to ones liking by setting the relevant arguments of the **plot** method.

A SVAR model is estimated with the function **SVAR()**. An object with class attribute **varest** has to be provided as argument for **x**. The structural parameters are estimated either by a scoring algorithm (the default) or by direct minimization of the negative log-likelihood function. Whether an *A*, *B* or *AB* model will be estimated, is dependent on the setting for **Amat** and **Bmat**. If a restriction matrix for **Amat** with dimension $(K \times K)$ is provided and the argument **Bmat** is left **NULL**, an *A* model will be estimated. In this case **Bmat** is set internally to an identity matrix I_K . Alternatively, if only a matrix object for **Bmat** is provided and **Amat** is left unchanged, then a *B* model will be estimated and internally **Amat** is set to an identity matrix I_K . Finally, if matrix objects for both arguments are provided, then an *AB* model will be estimated. In all cases, the matrix elements to be estimated are marked by **NA** entries at the relevant positions. The user has the option to provide starting values for the unknown coefficients by providing a vector object for the argument **start**. If **start** is left **NULL**, then **0.1** will be used as the starting value for all coefficients. The arguments **max.iter**, **conv.crit** and **maxls** can be used for tuning the scoring algorithm. The maximum number of iterations is controlled by **max.iter**, the convergence criterion is set by **conv.crit** and the maximum step length is set by **maxls**. A likelihood ratio test is computed per default for over-identified systems. This default setting can be offset by **lrtest = FALSE**. If a just-identified

³As an alternative one can use the function **VARselect()**. The result of this function is a list object with elements **selection** and **criteria**. The element **selection** is a vector of optimal lag length according to the above mentioned information criteria. The element **criteria** is a matrix containing the particular values for each of these criteria up to the maximal chosen lag order.

has been set-up, a warning is issued that an over-identification test cannot be computed in case of `lrtest = TRUE`. The ellipsis argument (...) is passed to `optim()` in case of direct optimization.

The returned object of function `SVAR()` is a list with class attribute `svarest`. Dependent on the chosen model and if the argument `hessian = TRUE` has been set in case of `estmethod = "direct"`, the list elements `A`, `Ase`, `B`, `Bse` contain the estimated coefficient matrices with the numerical standard errors. The element `LRIM` does contain the long-run impact matrix in case a SVAR of type Blanchard & Quah is estimated with function `BQ()`, otherwise this element is `NULL` (see [Blanchard and Quah 1989](#)). The list element `Sigma.U` is the variance-covariance matrix of the reduced form residuals times 100, i.e., $\Sigma_U = A^{-1}BB^\top A^{-1^\top} \times 100$. The list element `LR` is an object with class attribute `htest`, holding the likelihood ratio over-identification test. The element `opt` is the returned object from function `optim()` in case `estmethod = "direct"` has been used. The remaining five list items are the vector of starting values, the SVAR model type, the `varest` object, the number of iterations and the `call` to `SVAR()`.

A SVEC model is estimated with the function `SVEC()`. The supplied object for the argument `x` must be of formal class `ca.jo`. The restrictions on the long-run and short-run structural coefficient matrices must be provided as arguments `LR` and `SR`, respectively. These matrices have either zero or `NA` entries as their elements. It is further necessary to specify the cointegration rank of the estimated VECM via the argument `r`. Likewise to `SVAR()`, the arguments `start`, `max.iter`, `conv.crit` and `maxls` can be used for tuning the scoring algorithm. The argument `lrtest` applies likewise as in `SVAR()`. Finally, the logical flag `boot` can be used for calculating standard errors by applying the bootstrap method to the SVEC. The count of repetition is set by the argument `runs`. The returned list object from `SVEC()` and its associated methods will be bespoken in Section 4, where a SVEC model is specified for the Canadian labor market.

Finally, with function `vec2var()` a VECM (i.e., an object of formal class `ca.jo`, generated by the function `ca.jo()` contained in the package `urca`) is transformed into its level-VAR representation. Aside of this argument the function requires the cointegrating rank as this information is needed for the transformation (see Equations 12–16). The `print` method does return the coefficient values, first for the lagged endogenous variables, next for the deterministic regressors.

3.3. Diagnostic testing

In the package `vars` functions for diagnostic testing are `arch.test()`, `normality.test()`, `serial.test()` and `stability()`. The former three functions return a list object with class attribute `varcheck` for which `plot` and `print` method exist. The plots – one for each equation – include a residual plot, an empirical distribution plot and the ACF and PACF of the residuals and their squares. The `plot` method offers additional arguments for adjusting its appearance.

The implemented tests for heteroscedasticity are the univariate and multivariate ARCH test (see [Engle 1982](#); [Hamilton 1994](#); [Lütkepohl 2006](#)). The multivariate ARCH-LM test is based on the following regression (the univariate test can be considered as special case of the exhibition below and is skipped):

$$\text{vech}(\hat{\mathbf{u}}_t \hat{\mathbf{u}}_t^\top) = \boldsymbol{\beta}_0 + B_1 \text{vech}(\hat{\mathbf{u}}_{t-1} \hat{\mathbf{u}}_{t-1}^\top) + \dots + B_q \text{vech}(\hat{\mathbf{u}}_{t-q} \hat{\mathbf{u}}_{t-q}^\top) + \mathbf{v}_t , \quad (20)$$

whereby \mathbf{v}_t assigns a spherical error process and vech is the column-stacking operator for symmetric matrices that stacks the columns from the main diagonal on downward. The dimension of β_0 is $\frac{1}{2}K(K+1)$ and for the coefficient matrices B_i with $i = 1, \dots, q$, $\frac{1}{2}K(K+1) \times \frac{1}{2}K(K+1)$. The null hypothesis is: $H_0 := B_1 = B_2 = \dots = B_q = 0$ and the alternative is: $H_1 : B_1 \neq 0 \cap B_2 \neq 0 \cap \dots \cap B_q \neq 0$. The test statistic is defined as:

$$\text{VARCH}_{LM}(q) = \frac{1}{2}TK(K+1)R_m^2 , \quad (21)$$

with

$$R_m^2 = 1 - \frac{2}{K(K+1)} \text{tr}(\hat{\Omega}\hat{\Omega}_0^{-1}) , \quad (22)$$

and $\hat{\Omega}$ assigns the covariance matrix of the above defined regression model. This test statistic is distributed as $\chi^2(qK^2(K+1)^2/4)$.

The default is to compute the multivariate test only. If `multivariate.only = FALSE`, the univariate tests are computed too. In this case, the returned list object from `arch.test()` has three elements. The first element is the matrix of residuals. The second, signified by `arch.uni`, is a list object itself and holds the univariate test results for each of the series. The multivariate test result is contained in the third list element, signified by `arch.mul`.

```
arch.test(x, lags.single = 16, lags.multi = 5, multivariate.only = TRUE)
```

The returned tests have class attribute `htest`, hence the `print` method for these kind of objects is implicitly used as `print` method for objects of class `varcheck`. This applies likewise to the functions `normality.test()` and `serial.test()`, which will be bespoken next.

The Jarque-Bera normality tests for univariate and multivariate series are implemented and applied to the residuals of a $\text{VAR}(p)$ as well as separate tests for multivariate skewness and kurtosis (see [Bera and Jarque 1980, 1981](#); [Jarque and Bera 1987](#); [Lütkepohl 2006](#)). The univariate versions of the Jarque-Bera test are applied to the residuals of each equation. A multivariate version of this test can be computed by using the residuals that are standardized by a Choleski decomposition of the variance-covariance matrix for the centered residuals. Please note, that in this case the test result is dependent upon the ordering of the variables. The test statistics for the multivariate case are defined as:

$$JB_{mv} = s_3^2 + s_4^2 , \quad (23)$$

whereby s_3^2 and s_4^2 are computed according to:

$$s_3^2 = T\mathbf{b}_1^\top \mathbf{b}_1 / 6 \quad (24a)$$

$$s_4^2 = T(\mathbf{b}_2 - \mathbf{3}_K)^\top (\mathbf{b}_2 - \mathbf{3}_k) / 24 , \quad (24b)$$

with \mathbf{b}_1 and \mathbf{b}_2 are the third and fourth non-central moment vectors of the standardized residuals $\hat{\mathbf{u}}_t^s = \tilde{P}^-(\hat{\mathbf{u}}_t - \bar{\hat{\mathbf{u}}}_t)$ and \tilde{P} is a lower triangular matrix with positive diagonal such that $\tilde{P}\tilde{P}^\top = \tilde{\Sigma}_{\mathbf{u}}$, i.e., the Choleski decomposition of the residual covariance matrix. The test statistic JB_{mv} is distributed as $\chi^2(2K)$ and the multivariate skewness, s_3^2 , and kurtosis test, s_4^2 are distributed as $\chi^2(K)$.

```
normality.test(x, multivariate.only = TRUE)
```

The matrix of residuals is the first element in the returned list object. The function's default is to compute the multivariate test statistics only. The univariate tests are returned if `multivariate.only = FALSE` is set. Similar to the returned list object of function `arch.test()` for this case, the univariate versions of the Jarque-Bera test are applied to the residuals of each equation and are contained in the second list element signified by `jb.uni`. The multivariate version of the test as well as multivariate tests for skewness and kurtosis are contained in the third list element signified by `jb.mul`.

For testing the lack of serial correlation in the residuals of a VAR(p), a Portmanteau test and the Breusch-Godfrey LM test are implemented in the function `serial.test()`. For both tests small sample modifications can be calculated too, whereby the modification for the LM test has been introduced by [Edgerton and Shukur \(1999\)](#).

The Portmanteau statistic is defined as:

$$Q_h = T \sum_{j=1}^h \text{tr}(\hat{C}_j^\top \hat{C}_0^{-1} \hat{C}_j \hat{C}_0^{-1}) , \quad (25)$$

with $\hat{C}_i = \frac{1}{T} \Sigma_{t=i+1}^T \hat{\mathbf{u}}_t \hat{\mathbf{u}}_{t-i}^\top$. The test statistic has an approximate $\chi^2(K^2 h - n^*)$ distribution, and n^* is the number of coefficients excluding deterministic terms of a VAR(p). The limiting distribution is only valid for h tending to infinity at a suitable rate with growing sample size. Hence, the trade-off is between a decent approximation to the χ^2 distribution and a loss in power of the test, when h is chosen too large. The small sample adjustment of the test statistic is given as:

$$Q_h^* = T^2 \sum_{j=1}^h \frac{1}{T-j} \text{tr}(\hat{C}_j^\top \hat{C}_0^{-1} \hat{C}_j \hat{C}_0^{-1}) , \quad (26)$$

and is computed if `type = "PT.adjusted"` is set.

The Breusch-Godfrey LM statistic is based upon the following auxiliary regressions (see [Breusch 1978; Godfrey 1978](#)):

$$\hat{\mathbf{u}}_t = A_1 \mathbf{y}_{t-1} + \dots + A_p \mathbf{y}_{t-p} + C D_t + B_1 \hat{\mathbf{u}}_{t-1} + \dots + B_h \hat{\mathbf{u}}_{t-h} + \varepsilon_t . \quad (27)$$

The null hypothesis is: $H_0 : B_1 = \dots = B_h = 0$ and correspondingly the alternative hypothesis is of the form $H_1 : \exists B_i \neq 0$ for $i = 1, 2, \dots, h$. The test statistic is defined as:

$$LM_h = T(K - \text{tr}(\tilde{\Sigma}_R^{-1} \tilde{\Sigma}_e)) , \quad (28)$$

where $\tilde{\Sigma}_R$ and $\tilde{\Sigma}_e$ assign the residual covariance matrix of the restricted and unrestricted model, respectively. The test statistic LM_h is distributed as $\chi^2(hK^2)$. [Edgerton and Shukur \(1999\)](#) proposed a small sample correction, which is defined as:

$$LMF_h = \frac{1 - (1 - R_r^2)^{1/r}}{(1 - R_r^2)^{1/r}} \frac{Nr - q}{Km} , \quad (29)$$

with $R_r^2 = 1 - |\tilde{\Sigma}_e|/|\tilde{\Sigma}_R|$, $r = ((K^2 m^2 - 4)/(K^2 + m^2 - 5))^{1/2}$, $q = 1/2 Km - 1$ and $N = T - K - m - 1/2(K - m + 1)$, whereby n is the number of regressors in the original system and $m = Kh$. The modified test statistic is distributed as $F(hK^2, \text{int}(Nr - q))$. This test statistic is returned if `type = "ES"` has been used.

```
serial.test(x, lags.pt = 16, lags.bg = 5,
            type = c("PT.asymptotic", "PT.adjusted", "BG", "ES"))
```

The test statistics are returned in the list element `serial` and have class attribute `htest`. Per default the asymptotic Portmanteau test is returned. The residuals are contained in the first list element.

The function `stability()` returns a list object with class attribute `varstabil`. The function itself is just a wrapper for the function `efp()` contained in the package **strucchange** (see [Zeileis, Leisch, Hornik, and Kleiber 2002](#), for a detailed exposition of the package's capabilities). The first element of the returned list object is itself a list of objects with class attribute `efp`. Hence, the `plot` method for objects of class `varstabil` just calls the `plot` method for objects of class `efp`.

```
stability(x, type = c("OLS-CUSUM", "Rec-CUSUM", "Rec-MOSUM",
                      "OLS-MOSUM", "RE", "ME", "Score-CUSUM", "Score-MOSUM", "fluctuation"),
           h = 0.15, dynamic = FALSE, rescale = TRUE)
```

3.4. Prediction, impulse responses and forecast error decomposition

A `predict` method for objects with class attribute `varest` or `vec2var` is available. The `n.ahead` forecasts are computed recursively for the estimated $\text{VAR}(p)$ -process and a value for the forecast confidence interval can be provided too. Its default value is 0.95. The confidence interval is inferred from the empirical forecast error covariance matrix.

```
predict(object, ..., n.ahead = 10, ci = 0.95, dumvar = NULL)
```

The `predict` method returns a list with class attribute `varprd`. The forecasts are contained as a list in its first element signified as `fcst`. The second entry are the endogenous variables themselves. The last element is the submitted model object to `predict`. The `print` method returns the forecasts with upper and lower confidence levels, if applicable. The `plot` method draws the time series plots, whereby the start of the out-of-sample period is marked by a dashed vertical line. If this method is called interactively, the user is requested to browse through the graphs for each variable by hitting the <RETURN> key. For visualizing forecasts (i.e., objects with class attribute `varprd`) fan charts can be generated by the function `fanchart()` (see [Britton, Fisher, and Whitley 1998](#)). If the functional argument `color` is not set, the colors are taken from the gray color scheme. Likewise, if no confidence levels are supplied, then the fan charts are produced for confidence levels ranging from 0.1 to 0.9 with step size 0.1.

```
fanchart(x, colors = NULL, cis = NULL, names = NULL, main = NULL,
          ylab = NULL, xlab = NULL, col.y = NULL, nc, plot.type = c("multiple",
          "single"), mar = par("mar"), oma = par("oma"), ...)
```

The impulse response analysis is based upon the Wold moving average representation of a $\text{VAR}(p)$ -process (see Equations 5 and 6 above). It is used to investigate the dynamic interactions between the endogenous variables. The (i, j) th coefficients of the matrices Φ_s are thereby interpreted as the expected response of variable $y_{i,t+s}$ to a unit change in variable

y_{jt} . These effects can be accumulated through time, $s = 1, 2, \dots$, and hence one would obtain the simulated impact of a unit change in variable j to the variable i at time s . Aside of these impulse response coefficients, it is often conceivable to use orthogonal impulse responses as an alternative. This is the case, if the underlying shocks are less likely to occur in isolation, but when contemporaneous correlations between the components of the error process \mathbf{u}_t exist, i.e., the off-diagonal elements of $\Sigma_{\mathbf{u}}$ are non-zero. The orthogonal impulse responses are derived from a Choleski decomposition of the error variance-covariance matrix: $\Sigma_{\mathbf{u}} = PP^{\top}$ with P being a lower triangular. The moving average representation can then be transformed to:

$$\mathbf{y}_t = \Psi_0 \boldsymbol{\varepsilon}_t + \Psi_1 \boldsymbol{\varepsilon}_{t-1} + \dots, \quad (30)$$

with $\boldsymbol{\varepsilon}_t = P^{-1} \mathbf{u}_t$ and $\Psi_i = \Phi_i P$ for $i = 0, 1, 2, \dots$ and $\Psi_0 = P$. Incidentally, because the matrix P is lower triangular, it follows that only a shock in the first variable of a VAR(p)-process does exert an influence on all the remaining ones and that the second and following variables cannot have a direct impact on y_{1t} . Please note, that a different ordering of the variables might produce different outcomes with respect to the impulse responses. The non-uniqueness of the impulse responses can be circumvented by analyzing a set of endogenous variables in the SVAR framework.

Impulse response analysis has been implemented as a method for objects with class attribute of either `varest`, `svarest`, `svecest` or `vec2var`. These methods are utilizing the methods `Phi` and `Psi`, where applicable.

```
irf(x, impulse = NULL, response = NULL, n.ahead = 10, ortho = TRUE,
cumulative = FALSE, boot = TRUE, ci = 0.95, runs = 100, seed = NULL, ...)
```

The impulse variables are set as a character vector `impulse` and the responses are provided likewise in the argument `response`. If either one is unset, then all variables are considered as impulses or responses, respectively. The default length of the impulse responses is set to 10 via argument `n.ahead`. The computation of orthogonal and/or cumulative impulse responses is controlled by the logical switches `ortho` and `cumulative`, respectively. Finally, confidence bands can be returned by setting `boot = TRUE` (default). The preset values are to run 100 replications and return 95% confidence bands. It is at the user's leisure to specify a `seed` for the random number generator. The standard percentile interval is calculated as $CI_s = [s_{\gamma/2}^*, s_{(1-\gamma)/2}^*]$, where $s_{\gamma/2}^*$ and $s_{(1-\gamma)/2}^*$ are the $\gamma/2$ and $(1 - \gamma)/2$ quantiles of the estimated bootstrapped impulse response coefficients $\hat{\Phi}^*$ or $\hat{\Psi}^*$ (see Efron and Tibshirani 1993). The `irf` method returns a list object with class attribute `varirf` for which `print` and `plot` methods do exist. Likewise to the `plot` method of objects with class attribute `varprd`, the user is requested to browse through the graphs for each variable by hitting the <RETURN> key, whence the method is called interactively. The appearance of the plots can be adjusted.

The forecast error variance decomposition is based upon the orthogonal impulse response coefficient matrices Ψ_n . The FEVD allows the user to analyze the contribution of variable j to the h -step forecast error variance of variable k . If the element-wise squared orthogonal impulse responses are divided by the variance of the forecast error variance, $\sigma_k^2(h)$, the resultant is a percentage figure. The `fevd` method is available for conducting FEVD. Methods for objects of classes `varest`, `svarest`, `svecest` and `vec2var` do exist. Aside of the object itself, the argument `n.ahead` can be specified; its default value is 10.

```
fevd(x, n.ahead = 10, ...)
```

The method returns a list object with class attribute **varfevd** for which **print** and **plot** methods do exist. The list elements are the forecast error variances organized on a per-variable basis. The **plot** method for these objects is similar to the ones for objects with class attribute **varirf** and/or **varprd** and the appearance of the plots can be adjusted too.

4. Example

Functions and methods from the last section are now illustrated with a macro economic data set for Canada. It is shown how the results presented in Breitung, Brüggemann, and Lütkepohl (2004) can be replicated. However, the R code snippets should illustrate the ease of application rather than commenting and interpreting the results in depth.

The authors investigated the Canadian labor market. They utilized the following series: labor productivity defined as the log difference between GDP and employment, the log of employment, the unemployment rate and real wages, defined as the log of the real wage index. These series are signified by “**prod**”, “**e**”, “**U**” and “**rw**”, respectively. The data is taken from the OECD data base and spans from the first quarter 1980 until the fourth quarter 2004.

In a first step the package **vars** is loaded into the workspace. The Canadian data set which is included in the package **vars** is brought into memory.

```
R> library("vars")
R> data("Canada")
R> summary(Canada)

      e          prod          rw          U
Min. :929  Min. :401  Min. :386  Min. : 6.70
1st Qu.:935  1st Qu.:405  1st Qu.:424  1st Qu.: 7.78
Median :946  Median :406  Median :444  Median : 9.45
Mean   :944  Mean   :408  Mean   :441  Mean   : 9.32
3rd Qu.:950  3rd Qu.:411  3rd Qu.:461  3rd Qu.:10.61
Max.   :962  Max.   :418  Max.   :470  Max.   :12.77

R> plot(Canada, nc = 2, xlab = "")
```

A preliminary data analysis is conducted by displaying the summary statistics of the series involved as well as the corresponding time series plots (see Figure 1). In a next step, the authors conducted unit root tests by applying the Augmented Dickey-Fuller test regressions to the series (henceforth: ADF test). The ADF test has been implemented in the package **urca** as function **ur.df()**, for instance. The result of the ADF tests are summarized in Table 2.⁴

```
R> adf1 <- summary(ur.df(Canada[, "prod"], type = "trend", lags = 2))
R> adf1

#####
# Augmented Dickey-Fuller Test Unit Root Test #
```

⁴In the following only R code excerpts are shown. The R code for producing the Tables and Figures below are provided in a separate file that accompanies this text.

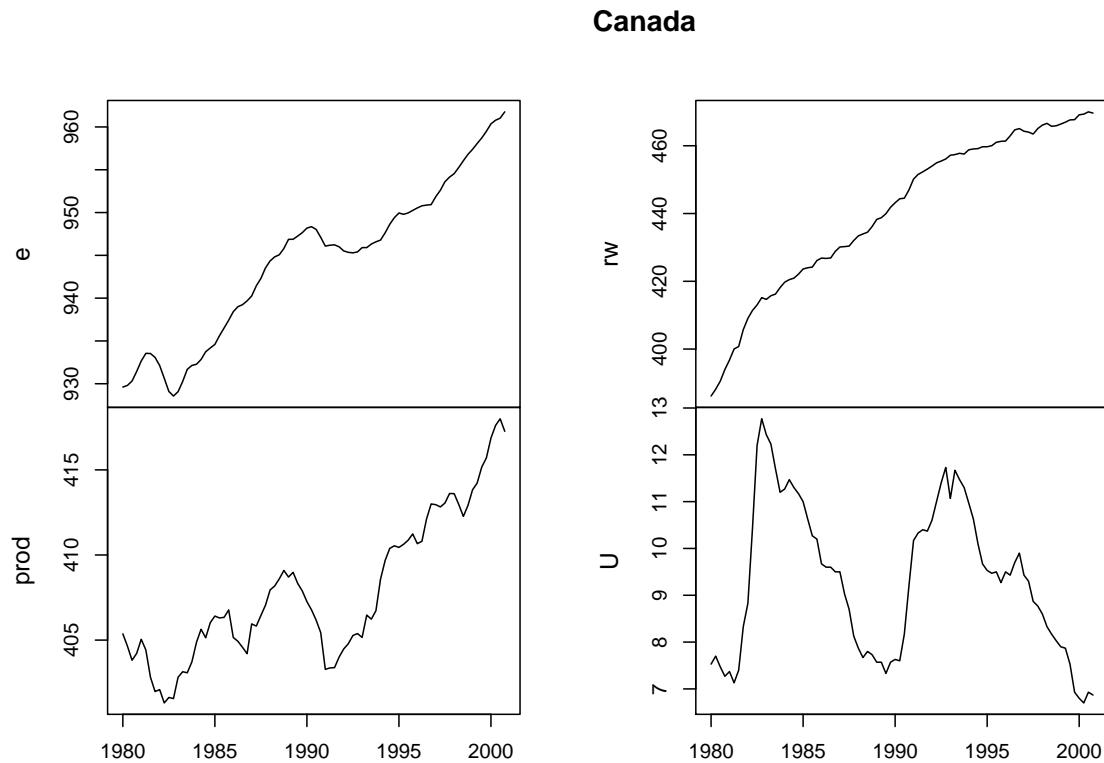


Figure 1: Canadian labor market time series.

```
#####
#
```

```
Test regression trend
```

```
Call:
```

```
lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
```

```
Residuals:
```

Min	1Q	Median	3Q	Max
-2.1992	-0.3899	0.0429	0.4191	1.7166

```
Coefficients:
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	30.41523	15.30940	1.99	0.051 .
z.lag.1	-0.07579	0.03813	-1.99	0.050 .
tt	0.01390	0.00642	2.16	0.034 *
z.diff.lag1	0.28487	0.11436	2.49	0.015 *
z.diff.lag2	0.08002	0.11609	0.69	0.493

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.685 on 76 degrees of freedom

Multiple R-squared: 0.135, Adjusted R-squared: 0.0899

F-statistic: 2.98 on 4 and 76 DF, p-value: 0.0244

Value of test-statistic is: -1.9875 2.3 2.3817

Critical values for test statistics:

1pct 5pct 10pct

tau3 -4.04 -3.45 -3.15

phi2 6.50 4.88 4.16

phi3 8.73 6.49 5.47

```
R> adf2 <- summary(ur.df(diff(Canada[, "prod"])), type = "drift",
+   lags = 1))
```

```
R> adf2
```

```
#####
# Augmented Dickey-Fuller Test Unit Root Test #
#####
```

Test regression drift

Call:

```
lm(formula = z.diff ~ z.lag.1 + 1 + z.diff.lag)
```

Residuals:

Min	1Q	Median	3Q	Max
-2.0512	-0.3953	0.0782	0.4111	1.7513

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	0.1153	0.0803	1.44	0.15
z.lag.1	-0.6889	0.1335	-5.16	1.8e-06 ***
z.diff.lag	-0.0427	0.1127	-0.38	0.71

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.697 on 78 degrees of freedom

Multiple R-squared: 0.361, Adjusted R-squared: 0.345

F-statistic: 22.1 on 2 and 78 DF, p-value: 2.53e-08

Variable	Deterministic terms	Lags	Test value	Critical values		
				1%	5%	10%
prod	constant, trend	2	-1.99	-4.04	-3.45	-3.15
Δ prod	constant	1	-5.16	-3.51	-2.89	-2.58
e	constant, trend	2	-1.91	-4.04	-3.45	-3.15
Δ e	constant	1	-4.51	-3.51	-2.89	-2.58
U	constant	1	-2.22	-3.51	-2.89	-2.58
Δ U		0	-4.75	-2.60	-1.95	-1.61
rw	constant, trend	4	-2.06	-4.04	-3.45	-3.15
Δ rw	constant	3	-2.62	-3.51	-2.89	-2.58
Δ rw	constant	0	-5.60	-3.51	-2.89	-2.58

Table 2: ADF tests for Canadian data.

```
Value of test-statistic is: -5.1604 13.318
```

Critical values for test statistics:

	1pct	5pct	10pct
tau2	-3.51	-2.89	-2.58
phi1	6.70	4.71	3.86

It can be concluded that all time series are integrated of order one. Please note, that the reported critical values differ slightly from the ones that are reported in Breitung *et al.* (2004). The authors utilized the software **JMulTi** (Lütkepohl and Krätsig 2004) in which the critical values of Davidson and MacKinnon (1993) are used, whereas in the function `ur.df()` the critical values are taken from Dickey and Fuller (1981) and Hamilton (1994).

In an ensuing step, the authors determined an optimal lag length for an unrestricted VAR for a maximal lag length of eight.

```
R> VARselect(Canada, lag.max = 8, type = "both")

$selection
AIC(n)  HQ(n)  SC(n)  FPE(n)
      3       2       1       3

$criteria
      1       2       3       4       5       6
AIC(n) -6.2725791 -6.6366697 -6.7711769 -6.6346092 -6.3981322 -6.3077048
HQ(n)   -5.9784294 -6.1464203 -6.0848278 -5.7521604 -5.3195837 -5.0330565
SC(n)   -5.5365580 -5.4099679 -5.0537944 -4.4265460 -3.6993884 -3.1182803
FPE(n)   0.0018898  0.0013195  0.0011660  0.0013632  0.0017821  0.0020442
      7       8
AIC(n) -6.0707273 -6.0615969
HQ(n)   -4.5999792 -4.3947490
SC(n)   -2.3906220 -1.8908109
FPE(n)   0.0027686  0.0030601
```

According to the AIC and FPE the optimal lag number is $p = 3$, whereas the HQ criterion indicates $p = 2$ and the SC criterion indicates an optimal lag length of $p = 1$. They estimated for all three lag orders a VAR including a constant and a trend as deterministic regressors and conducted diagnostic tests with respect to the residuals. In the R code example below, the relevant commands are exhibited for the VAR(1) model. First, the variables have to be reordered in the same sequence as in Breitung *et al.* (2004). This step is necessary, because otherwise the results of the multivariate Jarque-Bera test, in which a Choleski decomposition is employed, would differ slightly from the reported ones in Breitung *et al.* (2004). In the R code lines below the estimation of the VAR(1) as well as the summary output and the diagram of fit for equation “e” is shown.

```
R> Canada <- Canada[, c("prod", "e", "U", "rw")]
R> p1ct <- VAR(Canada, p = 1, type = "both")
R> p1ct

VAR Estimation Results:
=====
Estimated coefficients for equation prod:
=====
Call:
prod = prod.l1 + e.l1 + U.l1 + rw.l1 + const + trend

prod.l1      e.l1      U.l1      rw.l1      const      trend
0.963137  0.012912  0.211089 -0.039094 16.243407  0.046131

Estimated coefficients for equation e:
=====
Call:
e = prod.l1 + e.l1 + U.l1 + rw.l1 + const + trend

prod.l1      e.l1      U.l1      rw.l1      const      trend
0.194650  1.238923  0.623015 -0.067763 -278.761211 -0.040660

Estimated coefficients for equation U:
=====
Call:
U = prod.l1 + e.l1 + U.l1 + rw.l1 + const + trend

prod.l1      e.l1      U.l1      rw.l1      const      trend
-0.123192 -0.248442  0.391580  0.065808 259.982010  0.034517

Estimated coefficients for equation rw:
=====
```

```

Call:
rw = prod.l1 + e.l1 + U.l1 + rw.l1 + const + trend

      prod.l1      e.l1      U.l1      rw.l1      const      trend
-0.223087 -0.051044 -0.368640  0.948909 163.024531  0.071422

R> summary(p1ct, equation = "e")

VAR Estimation Results:
=====
Endogenous variables: prod, e, U, rw
Deterministic variables: both
Sample size: 83
Log Likelihood: -207.525
Roots of the characteristic polynomial:
0.95 0.95 0.904 0.751
Call:
VAR(y = Canada, p = 1, type = "both")

Estimation results for equation e:
=====
e = prod.l1 + e.l1 + U.l1 + rw.l1 + const + trend

      Estimate Std. Error t value Pr(>|t|)
prod.l1    0.1947    0.0361     5.39  7.5e-07 ***
e.l1       1.2389    0.0863    14.35 < 2e-16 ***
U.l1       0.6230    0.1693     3.68  0.00043 ***
rw.l1      -0.0678    0.0283    -2.40  0.01899 *
const     -278.7612   75.1830    -3.71  0.00039 ***
trend      -0.0407    0.0197    -2.06  0.04238 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.47 on 77 degrees of freedom
Multiple R-Squared:    1,          Adjusted R-squared:    1
F-statistic: 5.58e+07 on 6 and 77 DF,  p-value: <2e-16

Covariance matrix of residuals:
      prod      e      U      rw
prod  0.46952  0.0677 -0.0413  0.00214
e     0.06767  0.2210 -0.1320 -0.08279
U     -0.04128 -0.1320  0.1216  0.06374
rw    0.00214 -0.0828  0.0637  0.59317

```

```
Correlation matrix of residuals:
      prod      e       U      rw
prod  1.00000  0.210 -0.173  0.00406
e     0.21008  1.000 -0.805 -0.22869
U    -0.17275 -0.805  1.000  0.23731
rw   0.00406 -0.229  0.237  1.00000
```

```
R> plot(p1ct, names = "e")
```

The resulting graphic is displayed in Figure 2. Next, it is shown how the diagnostic tests are conducted for the VAR(1) model. The results of all diagnostic tests, i.e., for the VAR(1), VAR(2) and VAR(3) model, are provided in Table 3 and the graphics of the `varcheck` object `arch1` for the employment equation and the OLS-CUSUM tests for the VAR(1) model are shown in Figures 3 and 4, respectively.

```
R> ser11 <- serial.test(p1ct, lags.pt = 16, type = "PT.asymptotic")
R> ser11$serial
```

```
Portmanteau Test (asymptotic)
```

```
data: Residuals of VAR object p1ct
Chi-squared = 233.5, df = 240, p-value = 0.606
```

```
R> norm1 <- normality.test(p1ct)
R> norm1$jb.mul
```

```
$JB
```

```
JB-Test (multivariate)
```

```
data: Residuals of VAR object p1ct
Chi-squared = 9.9189, df = 8, p-value = 0.2708
```

```
$Skewness
```

```
Skewness only (multivariate)
```

```
data: Residuals of VAR object p1ct
Chi-squared = 6.356, df = 4, p-value = 0.1741
```

```
$Kurtosis
```

```
Kurtosis only (multivariate)
```

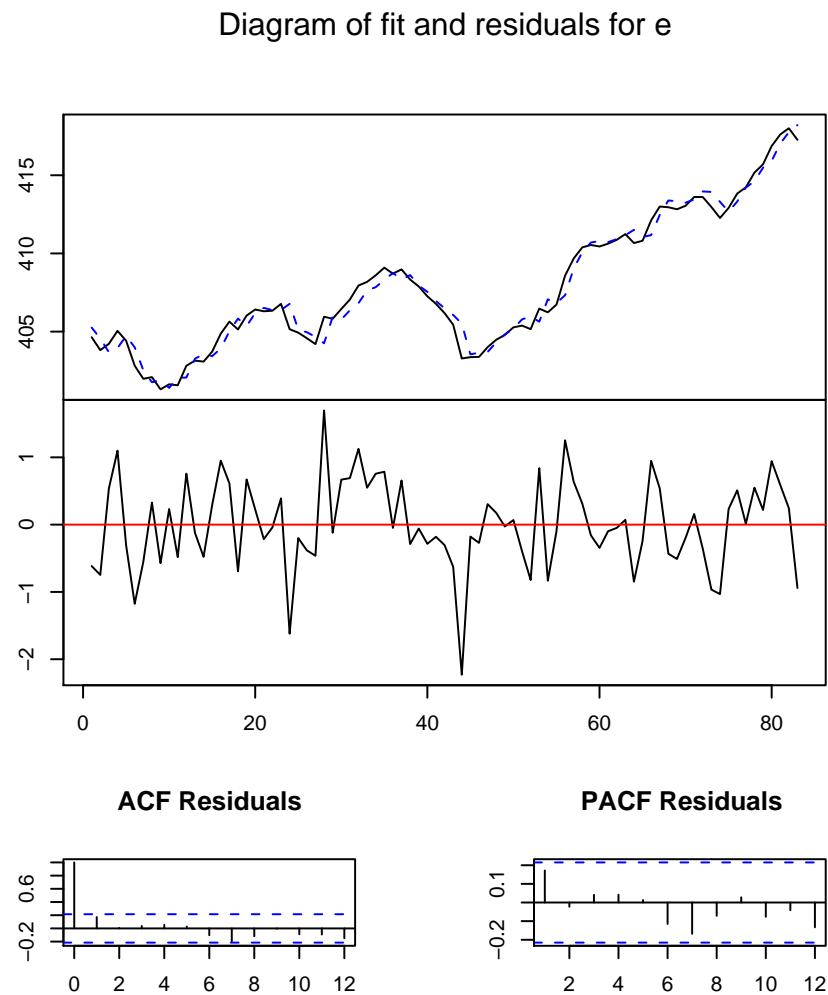


Figure 2: Plot of VAR(1) for equation “e”.

Model	Q_{16}	p value	Q_{16}^*	p value	JB_4	p value	$MARCH_5$	p value
$p = 3$	174.0	0.96	198.0	0.68	9.66	0.29	512.0	0.35
$p = 2$	209.7	0.74	236.1	0.28	2.29	0.97	528.1	0.19
$p = 1$	233.5	0.61	256.9	0.22	9.92	0.27	570.1	0.02

Table 3: Diagnostic tests of VAR(p) specifications for Canadian data.

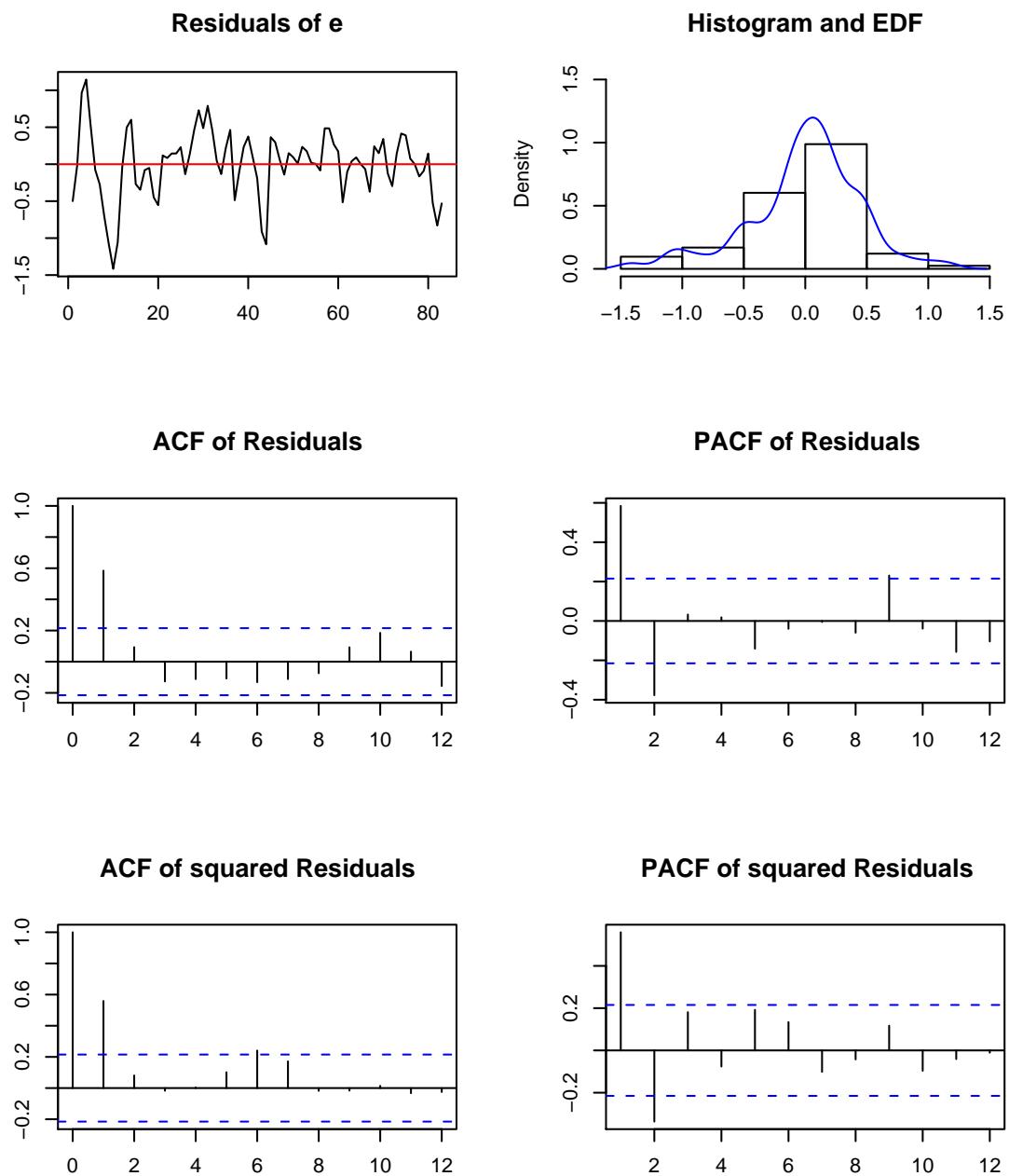


Figure 3: Diagnostic plot of VAR(1) for equation “e”.

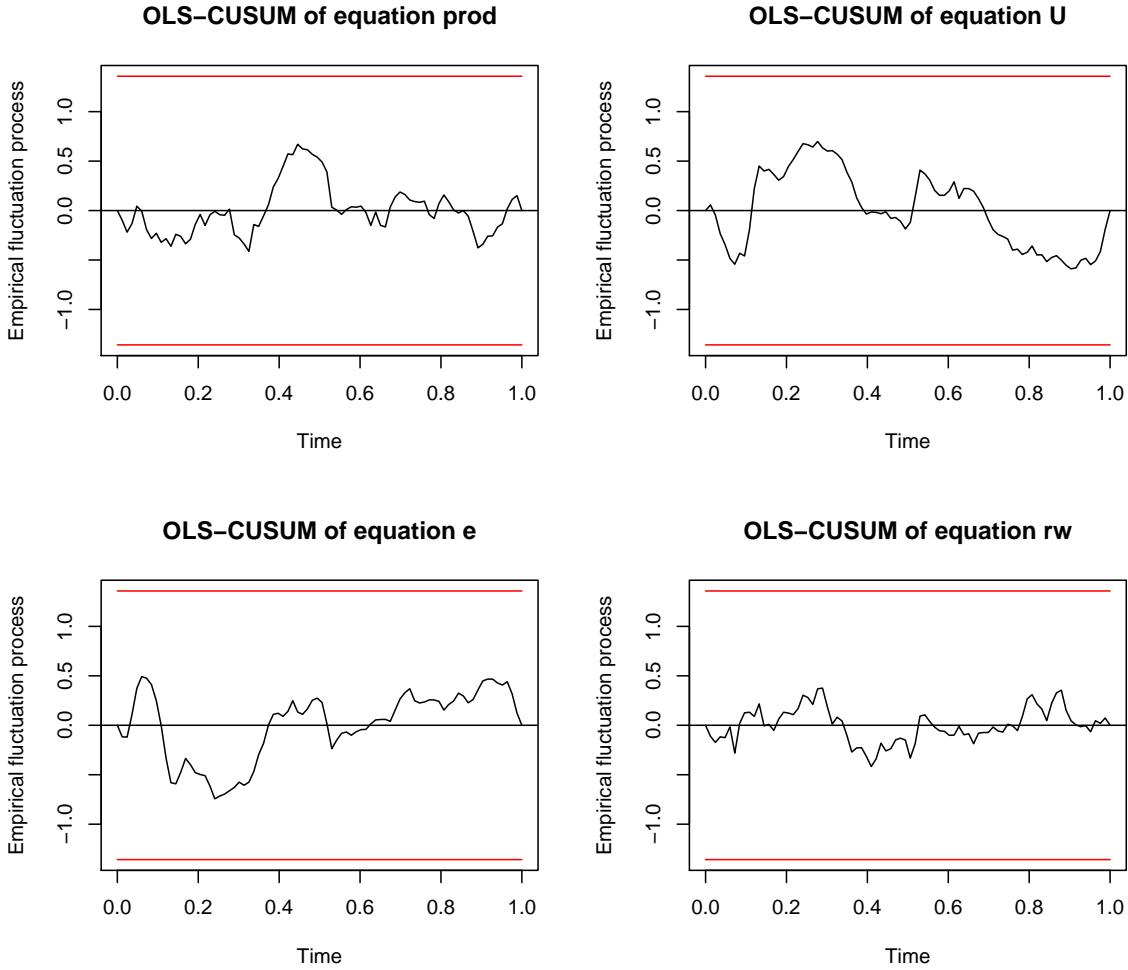


Figure 4: OLS-CUSUM test of VAR(1).

```

data: Residuals of VAR object pict
Chi-squared = 3.5629, df = 4, p-value = 0.4684

R> arch1 <- arch.test(pict, lags.multi = 5)
R> arch1$arch.mul

      ARCH (multivariate)

data: Residuals of VAR object pict
Chi-squared = 570.14, df = 500, p-value = 0.01606

R> plot(arch1, names = "e")
R> plot(stability(pict), nc = 2)

```

Given the diagnostic test results the authors concluded that a VAR(1)-specification might be too restrictive. They argued further, that although some of the stability tests do indicate

\mathcal{H}_0	Test Statistics		Critical Values		
	$p = 3$	$p = 2$	90%	95%	99%
$r = 0$	84.92	86.12	59.14	62.99	70.05
$r = 1$	36.42	37.33	39.06	42.44	48.45
$r = 2$	18.72	15.65	22.76	25.32	30.45
$r = 3$	3.85	4.10	10.49	12.25	16.26

Table 4: Johansen cointegration tests for Canadian system.

deviations from parameter constancy, the time-invariant specification of the VAR(2) and VAR(3) model will be maintained as tentative candidates for the following cointegration analysis.

The authors estimated a VECM whereby a deterministic trend has been included in the cointegration relation. The estimation of these models as well as the statistical inference with respect to the cointegration rank can be swiftly accomplished with the function `ca.jo()`. Although the following R code examples are using functions contained in the package `urca`, it is however beneficial to reproduce these results for two reasons: the interplay between the functions contained in package `urca` and `vars` is exhibited and it provides an understanding of the then following SVEC specification.

```
R> summary(ca.jo(Canada, type = "trace", ecdet = "trend", K = 3,
+   spec = "transitory"))
R> summary(ca.jo(Canada, type = "trace", ecdet = "trend", K = 2,
+   spec = "transitory"))
```

The outcome of the trace tests is provided in Table 4. These results do indicate one cointegration relationship. The reported critical values differ slightly from the ones that are reported in Table 4.3 of Breitung *et al.* (2004). The authors used the values that are contained in Johansen (1995), whereas the values from Osterwald-Lenum (1992) are used in the function `ca.jo()`. In the R code snippet below the VECM is re-estimated with this restriction and a normalization of the long-run relationship with respect to real wages. The results are shown in Table 5.

```
R> vecm <- ca.jo(Canada[, c("rw", "prod", "e", "U")], type = "trace",
+   ecdet = "trend", K = 3, spec = "transitory")
R> vecm.r1 <- cajorls(vecm, r = 1)
```

For a just identified SVEC model of type B one needs $\frac{1}{2}K(K - 1) = 6$ linear independent restrictions. It is further reasoned from the Beveridge-Nelson decomposition that there are $k^* = r(K - r) = 3$ shocks with permanent effects and only one shock that exerts a temporary effect, due to $r = 1$. Because the cointegration relation is interpreted as a stationary wage-setting relation, the temporary shock is associated with the wage shock variable. Hence, the four entries in the last column of the long-run impact matrix ΞB are set to zero. Because this matrix is of reduced rank, only $k^*r = 3$ linear independent restrictions are imposed thereby.

	Vector	prod	e	U	rw	trend
$\hat{\beta}^\top$		0.545 (0.90)	-0.013 (-0.02)	1.727 (1.19)	1.000	-0.709 (-2.57)
$\hat{\alpha}^\top$		-0.012 (-0.92)	-0.016 (-2.16)	-0.009 (-1.49)	-0.085 (-5.71)	

Table 5: Cointegration vector and loading parameters (with t statistics in parentheses).

It is therefore necessary to set $\frac{1}{2}k^*(k^* - 1) = 3$ additional elements to zero. The authors assumed constant returns of scale and therefore productivity is only driven by output shocks. This reasoning implies zero coefficients in the first row of the long-run matrix for the variables employment, unemployment and real wages, hence the elements $\Xi B_{1,j}$ for $j = 2, 3, 4$ are set to zero. Because $\Xi B_{1,4}$ has already been set to zero, only two additional restrictions have been added. The last restriction is imposed on the element $B_{4,2}$. Here, it is assumed that labor demand shocks do not exert an immediate effect on real wages.

In the R code example below the matrix objects `LR` and `SR` are set up accordingly and the just-identified SVEC is estimated with function `SVEC()`. In the call to the function `SVEC()` the argument `boot = TRUE` has been employed such that bootstrapped standard errors and hence t statistics can be computed for the structural long-run and contemporaneous coefficients.

```
R> vecm <- ca.jo(Canada[, c("prod", "e", "U", "rw")], type = "trace",
+   ecdet = "trend", K = 3, spec = "transitory")
R> SR <- matrix(NA, nrow = 4, ncol = 4)
R> SR[4, 2] <- 0
R> LR <- matrix(NA, nrow = 4, ncol = 4)
R> LR[1, 2:4] <- 0
R> LR[2:4, 4] <- 0
R> svec <- SVEC(vecm, LR = LR, SR = SR, r = 1, lrtest = FALSE, boot = TRUE,
+   runs = 100)
R> summary(svec)

SVEC Estimation Results:
=====

Call:
SVEC(x = vecm, LR = LR, SR = SR, r = 1, lrtest = FALSE, boot = TRUE,
      runs = 100)

Type: B-model
Sample size: 81
Log Likelihood: -161.838
Number of iterations: 10

Estimated contemporaneous impact matrix:
    prod        e        U        rw
prod  0.545
e     -0.013
U     1.727
rw    1.000
```

```
prod  0.5840  0.0743 -0.15258  0.0690
e     -0.1203  0.2614 -0.15510  0.0898
U     0.0253 -0.2672  0.00549  0.0498
rw    0.1117  0.0000  0.48377  0.4879
```

Estimated standard errors for impact matrix:

	prod	e	U	rw
prod	0.0803	0.1047	0.2125	0.0660
e	0.0705	0.0596	0.1665	0.0402
U	0.0527	0.0433	0.0568	0.0309
rw	0.1462	0.0000	0.6142	0.0888

Estimated long run impact matrix:

	prod	e	U	rw
prod	0.791	0.000	0.000	0
e	0.202	0.577	-0.492	0
U	-0.159	-0.341	0.141	0
rw	-0.153	0.596	-0.250	0

Estimated standard errors for long-run matrix:

	prod	e	U	rw
prod	0.152	0.0000	0.000	0
e	0.230	0.1801	0.550	0
U	0.115	0.0884	0.149	0
rw	0.184	0.1549	0.264	0

Covariance matrix of reduced form residuals (*100):

	prod	e	U	rw
prod	37.464	-2.10	-0.251	2.51
e	-2.096	11.49	-6.927	-4.47
U	-0.251	-6.93	7.454	2.98
rw	2.509	-4.47	2.978	48.46

The results are summarized in the Tables 6 and 7. The values of the t statistics differ slightly from the reported ones in Breitung *et al.* (2004) which can be attributed to sampling. In the R code example above only 100 runs have been executed, whereas in Breitung *et al.* (2004) 2000 repetitions have been used.

The authors investigated further if labor supply shocks do have no long-run impact on unemployment. This hypothesis is mirrored by setting $\Xi B_{3,3} = 0$. Because one more zero restriction has been added to the long-run impact matrix, the SVEC model is now over-identified. The validity of this over-identification restriction can be tested with a LR test. In the R code example below first the additional restriction has been set and then the SVEC is re-estimated by using the `update` method. The result of the LR test is contained in the returned list as named element `L Rover`.

```
R> LR[3, 3] <- 0
R> svec.oi <- update(svec, LR = LR, lrtest = TRUE, boot = FALSE)
```

Equation	$\varepsilon_t^{\text{gdp}}$	$\varepsilon_t^{\text{Labor}^d}$	$\varepsilon_t^{\text{Labor}^s}$	$\varepsilon_t^{\text{wage}}$
Output	0.58 (7.28)	0.07 (0.71)	-0.15 (-0.72)	0.07 (1.05)
Labor demand	-0.12 (-1.71)	0.26 (4.39)	-0.16 (-0.93)	0.09 (2.23)
Unemployment	0.03 (0.48)	-0.27 (-6.16)	0.01 (0.10)	0.05 (1.61)
Real wages	0.11 (0.76)	0.00 (0.79)	0.48 (5.50)	0.49

Table 6: Estimated coefficients of the contemporaneous impact matrix (with t statistics in parentheses).

Equation	$\varepsilon_t^{\text{gdp}}$	$\varepsilon_t^{\text{Labor}^d}$	$\varepsilon_t^{\text{Labor}^s}$	$\varepsilon_t^{\text{wage}}$
Output	0.79 (5.21)	0.00	0.00	0.00
Labor demand	0.20 (0.88)	0.58 (3.20)	-0.49 (-0.89)	0.00
Unemployment	-0.16 (-1.38)	-0.34 (-3.86)	0.14 (0.95)	0.00
Real wages	-0.15 (-0.83)	0.60 (3.85)	-0.25 (-0.95)	0.00

Table 7: Estimated coefficients of the long-run impact matrix (with t statistics in parentheses).

```
R> svec.oi$LRover
LR overidentification
data: vecm
Chi^2 = 6.0745, df = 1, p-value = 0.01371
```

The value of the test statistic is 6.07 and the p value of this $\chi^2(1)$ -distributed variable is 0.014. Therefore, the null hypothesis that shocks to the labor supply do not exert a long-run effect on unemployment has to be rejected for a significance level of 5%.

In order to investigate the dynamic effects on unemployment, the authors applied an impulse response analysis. The impulse response analysis shows the effects of the different shocks, i.e., output, labor demand, labor supply and wage, to unemployment. In the R code example below the `irf` method for objects with class attribute `svecest` is employed and the argument `boot = TRUE` has been set such that confidence bands around the impulse response trajectories can be calculated.

```
R> svec.irf <- irf(svec, response = "U", n.ahead = 48, boot = TRUE)
R> plot(svec.irf)
```

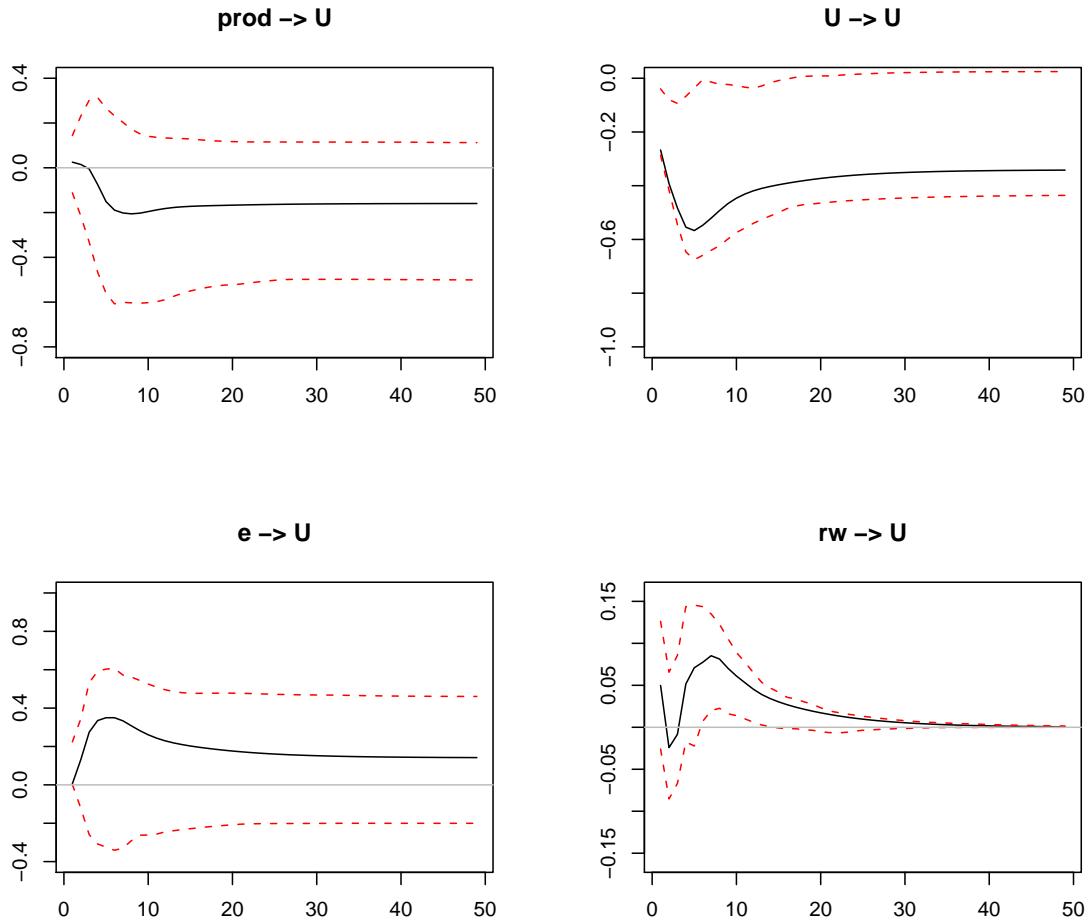


Figure 5: Responses of unemployment to economic shocks with a 95% bootstrap confidence interval.

The outcome of the IRA is exhibited in Figure 5.

In a final step, a forecast error variance decomposition is conducted with respect to unemployment. This is achieved by applying the `fevd` method to the object with class attribute `svecest`.

```
R> fevd.U <- fevd(svec, n.ahead = 48)$U
```

The authors report only the values for selected quarters. These results are displayed in Table 8.

Period	$\varepsilon_t^{\text{gdp}}$	$\varepsilon_t^{\text{Labor}^d}$	$\varepsilon_t^{\text{Labor}^s}$	$\varepsilon_t^{\text{wage}}$
1	0.01	0.96	0.00	0.03
4	0.01	0.78	0.21	0.01
8	0.05	0.69	0.24	0.01
12	0.08	0.68	0.23	0.01
24	0.10	0.69	0.21	0.01
48	0.12	0.70	0.18	0.00

Table 8: Forecast error variance decomposition of Canadian unemployment.

5. Summary

In this paper it has been described how the different functions and methods contained in the package **vars** are designed and offer the researcher a fairly easy to use environment for conducting VAR, SVAR and SVEC analysis. This is primarily achieved through methods for impulse response functions, forecast error variance decomposition, prediction as well as tools for diagnostic testing, determination of a suitable lag length for the model and stability / causality analysis.

The package **vars** complements the package **urca** in the sense that most of the above described tools are available for VECM that can be swiftly transformed to their level-VAR representation whence the cointegrating rank has been determined.

6. Computational details

The package's code is purely written in R (R Development Core Team 2008) and S3-classes with methods have been utilized. It is shipped with a NAMESPACE and a ChangeLog file. It has dependencies to **MASS** (Venables and Ripley 2002), **strucchange** (Zeileis *et al.* 2002) and **urca** (Pfaff 2006). R itself as well as these packages can be obtained from CRAN at <http://CRAN.R-project.org/>. Furthermore, daily builds of package **vars** are provided on R-Forge (see <http://R-Forge.R-project.org/projects/vars/>). It has been published under GPL version 2 or newer. The results used in this paper were obtained using R 2.7.0 with packages **vars** 1.3-8, **strucchange** 1.3-3, **urca** 1.1-6 and **MASS** 7.2-42.

Acknowledgments

I would like to thank the anonymous reviewers and Achim Zeileis for valuable feedback on this article as well as the suggested improvements for package **vars**.

References

- Amisano G, Giannini C (1997). *Topics in Structural VAR Econometrics*. Springer-Verlag, Berlin, 2nd edition.
- Banerjee A, Dolado J, Galbraith J, Hendry D (1993). *Co-Integration, Error-Correction, and the Econometric Analysis of Non-Stationary Data*. Oxford University Press, New York.
- Bera AK, Jarque CM (1980). “Efficient Tests for Normality, Homoscedasticity and Serial Independence of Regression Residuals.” *Economics Letters*, **6**(3), 255–259.
- Bera AK, Jarque CM (1981). “Efficient Tests for Normality, Homoscedasticity and Serial Independence of Regression Residuals: Monte Carlo Evidence.” *Economics Letters*, **7**(4), 313–318.
- Blanchard O, Quah D (1989). “The Dynamic Effects of Aggregate Demand and Supply Disturbances.” *The American Economic Review*, **79**(4), 655–673.
- Brandt PT, Appleby J (2007). *MSBVAR: Bayesian Vector Autoregression Models, Impulse Responses and Forecasting*. R package version 0.3.1, URL <http://CRAN.R-project.org/package=MSBVAR>.
- Breitung J, Brüggemann R, Lütkepohl H (2004). “Structural Vector Autoregressive Modeling and Impulse Responses.” In H Lütkepohl, M Krätsig (eds.), “Applied Time Series Econometrics,” chapter 4, pp. 159–196. Cambridge University Press, Cambridge.
- Breusch TS (1978). “Testing for Autocorrelation in Dynamic Linear Models.” *Australien Economic Papers*, **17**, 334–355.
- Britton E, Fisher PG, Whitley JD (1998). “The Inflation Report Projections: Understanding the Fan Chart.” *Bank of England Quarterly Bulletin*, **38**, 30–37.
- Davidson R, MacKinnon J (1993). *Estimation and Inference in Econometrics*. Oxford University Press, London.
- Dickey DA, Fuller WA (1981). “Likelihood Ratio Statistics for Autoregressive Time Series with a Unit Root.” *Econometrica*, **49**, 1057–1072.
- Edgerton D, Shukur G (1999). “Testing Autocorrelation in a System Perspective.” *Econometric Reviews*, **18**, 343–386.
- Efron B, Tibshirani RJ (1993). *An Introduction to the Bootstrap*. Chapman & Hall, New York.
- Engle RF (1982). “Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation.” *Econometrica*, **50**, 987–1007.
- Engle RF, Granger CWJ (1987). “Co-Integration and Error Correction: Representation, Estimation and Testing.” *Econometrica*, **55**, 251–276.

- Gilbert PD (1993). “State Space and ARMA Models: An Overview of the Equivalence.” *Working Paper 93-4*, Bank of Canada, Ottawa, Canada. URL <http://www.bank-banque-canada.ca/pgilbert/>.
- Gilbert PD (1995). “Combining VAR Estimation and State Space Model Reduction for Simple Good Predictions.” *Journal of Forecasting: Special Issue on VAR Modelling*, **14**, 229–250.
- Gilbert PD (2000). “A Note on the Computation of Time Series Model Roots.” *Applied Economics Letters*, **7**, 423–424.
- Godfrey LG (1978). “Testing for Higher Order Serial Correlation in Regression Equations when the Regressors Include Lagged Dependent Variables.” *Econometrica*, **46**, 1303–1310.
- Granger CWJ (1981). “Some Properties of Time Series Data and Their Use in Econometric Model Specification.” *Journal of Econometrics*, **16**, 121–130.
- Hamilton JD (1994). *Time Series Analysis*. Princeton University Press, Princeton.
- Hendry DF (1995). *Dynamic Econometrics*. Oxford University Press, Oxford.
- Jarque CM, Bera AK (1987). “A Test for Normality of Observations and Regression Residuals.” *International Statistical Review*, **55**, 163–172.
- Johansen S (1995). *Likelihood Based Inference in Cointegrated Vector Autoregressive Models*. Oxford University Press, Oxford.
- Lütkepohl H (2006). *New Introduction to Multiple Time Series Analysis*. Springer-Verlag, New York.
- Lütkepohl H, Krätzig M (2004). *Applied Time Series Econometrics*. Cambridge University Press, Cambridge.
- Osterwald-Lenum M (1992). “A Note with Quantiles of the Asymptotic Distribution of the Maximum Likelihood Cointegration Rank Test Statistics.” *Oxford Bulletin of Economics and Statistics*, **55**(3), 461–472.
- Pfaff B (2006). *Analysis of Integrated and Cointegrated Time Series with R*. Springer-Verlag, New York. URL <http://CRAN.R-project.org/package=urca>.
- Pfaff B (2008). “VAR, SVAR and SVEC Models: Implementation Within R Package vars.” *Journal of Statistical Software*, **27**(4). URL <http://www.jstatsoft.org/v27/i04/>.
- R Development Core Team (2008). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL <http://www.R-project.org/>.
- Sims CA (1980). “Macroeconomics and Reality.” *Econometrica*, **48**, 1–48.
- Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. Springer-Verlag, New York, 4th edition.
- Würtz D (2007). *fArma: Rmetrics – ARMA Time Series Modelling*. R package version 260.72, URL <http://CRAN.R-project.org/package=fArma>.

Zeileis A, Leisch F, Hornik K, Kleiber C (2002). “**strucchange**: An R Package for Testing for Structural Change in Linear Regression Models.” *Journal of Statistical Software*, **7**(2), 1–38. URL <http://www.jstatsoft.org/v07/i02/>.

Affiliation:

Bernhard Pfaff
61476 Kronberg im Taunus, Germany
E-mail: bernhard@pfaffikus.de
URL: <http://www.pfaffikus.de>

4

Unit Root Tests

4.1 Introduction

Many economic and financial time series exhibit trending behavior or non-stationarity in the mean. Leading examples are asset prices, exchange rates and the levels of macroeconomic aggregates like real GDP. An important econometric task is determining the most appropriate form of the trend in the data. For example, in ARMA modeling the data must be transformed to stationary form prior to analysis. If the data are trending, then some form of trend removal is required.

Two common trend removal or de-trending procedures are first differencing and time-trend regression. First differencing is appropriate for $I(1)$ time series and time-trend regression is appropriate for trend stationary $I(0)$ time series. Unit root tests can be used to determine if trending data should be first differenced or regressed on deterministic functions of time to render the data stationary. Moreover, economic and finance theory often suggests the existence of long-run equilibrium relationships among nonstationary time series variables. If these variables are $I(1)$, then cointegration techniques can be used to model these long-run relations. Hence, pre-testing for unit roots is often a first step in the cointegration modeling discussed in Chapter 12. Finally, a common trading strategy in finance involves exploiting mean-reverting behavior among the prices of pairs of assets. Unit root tests can be used to determine which pairs of assets appear to exhibit mean-reverting behavior.

This chapter is organized as follows. Section 4.2 reviews $I(1)$ and trend stationary $I(0)$ time series and motivates the unit root and stationary tests described in the chapter. Section 4.3 describes the class of autoregressive unit root tests made popular by David Dickey, Wayne Fuller, Pierre Perron and Peter Phillips. Section 4.4 describes the stationarity tests of Kwiatkowski, Phillips, Schmidt and Shin (1992). Section 4.5 discusses some problems associated with traditional unit root and stationarity tests, and Section 4.6 presents some recently developed so-called “efficient unit root tests” that overcome some of the deficiencies of traditional unit root tests.

In this chapter, the technical details of unit root and stationarity tests are kept to a minimum. Excellent technical treatments of nonstationary time series may be found in Hamilton (1994), Hatanaka (1995), Fuller (1996) and the many papers by Peter Phillips. Useful surveys on issues associated with unit root testing are given in Stock (1994), Maddala and Kim (1998) and Phillips and Xiao (1998).

4.2 Testing for Nonstationarity and Stationarity

To understand the econometric issues associated with unit root and stationarity tests, consider the stylized trend-cycle decomposition of a time series y_t :

$$\begin{aligned} y_t &= TD_t + z_t \\ TD_t &= \kappa + \delta t \\ z_t &= \phi z_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim WN(0, \sigma^2) \end{aligned}$$

where TD_t is a deterministic linear trend and z_t is an AR(1) process. If $|\phi| < 1$ then y_t is $I(0)$ about the deterministic trend TD_t . If $\phi = 1$, then $z_t = z_{t-1} + \varepsilon_t = z_0 + \sum_{j=1}^t \varepsilon_j$, a stochastic trend and y_t is $I(1)$ with drift. Simulated $I(1)$ and $I(0)$ data with $\kappa = 5$ and $\delta = 0.1$ are illustrated in Figure 4.1. The $I(0)$ data with trend follows the trend $TD_t = 5 + 0.1t$ very closely and exhibits trend reversion. In contrast, the $I(1)$ data follows an upward drift but does not necessarily revert to TD_t .

Autoregressive *unit root tests* are based on testing the null hypothesis that $\phi = 1$ (difference stationary) against the alternative hypothesis that $\phi < 1$ (trend stationary). They are called unit root tests because under the null hypothesis the autoregressive polynomial of z_t , $\phi(z) = (1 - \phi z) = 0$, has a root equal to unity.

Stationarity tests take the null hypothesis that y_t is trend stationary. If y_t is then first differenced it becomes

$$\begin{aligned} \Delta y_t &= \delta + \Delta z_t \\ \Delta z_t &= \phi \Delta z_{t-1} + \varepsilon_t - \varepsilon_{t-1} \end{aligned}$$

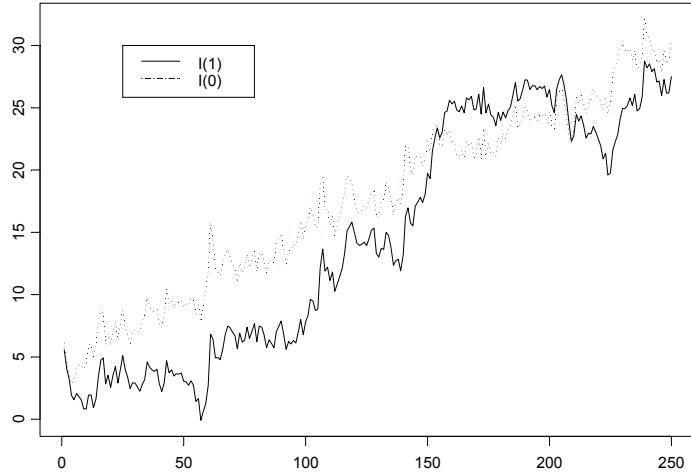


FIGURE 4.1. Simulated trend stationary ($I(0)$) and difference stationary ($I(1)$) processes.

Notice that first differencing y_t , when it is trend stationary, produces a unit moving average root in the ARMA representation of Δz_t . That is, the ARMA representation for Δz_t is the non-invertible ARMA(1,1) model

$$\Delta z_t = \phi \Delta z_{t-1} + \varepsilon_t + \theta \varepsilon_{t-1}$$

with $\theta = -1$. This result is known as *overdifferencing*. Formally, stationarity tests are based on testing for a unit moving average root in Δz_t .

Unit root and stationarity test statistics have nonstandard and nonnormal asymptotic distributions under their respective null hypotheses. To complicate matters further, the limiting distributions of the test statistics are affected by the inclusion of deterministic terms in the test regressions. These distributions are functions of standard Brownian motion (Wiener process), and critical values must be tabulated by simulation techniques. MacKinnon (1996) provides response surface algorithms for determining these critical values, and various **S+FinMetrics** functions use these algorithms for computing critical values and p-values.

4.3 Autoregressive Unit Root Tests

To illustrate the important statistical issues associated with autoregressive unit root tests, consider the simple AR(1) model

$$y_t = \phi y_{t-1} + \varepsilon_t, \text{ where } \varepsilon_t \sim WN(0, \sigma^2)$$

The hypotheses of interest are

$$\begin{aligned} H_0 &: \phi = 1 \text{ (unit root in } \phi(z) = 0) \Rightarrow y_t \sim I(1) \\ H_1 &: |\phi| < 1 \Rightarrow y_t \sim I(0) \end{aligned}$$

The test statistic is

$$t_{\phi=1} = \frac{\hat{\phi} - 1}{SE(\hat{\phi})}$$

where $\hat{\phi}$ is the least squares estimate and $SE(\hat{\phi})$ is the usual standard error estimate¹. The test is a one-sided left tail test. If $\{y_t\}$ is stationary (i.e., $|\phi| < 1$) then it can be shown (c.f. Hamilton (1994) pg. 216)

$$\sqrt{T}(\hat{\phi} - \phi) \xrightarrow{d} N(0, (1 - \phi^2))$$

or

$$\hat{\phi} \stackrel{A}{\sim} N\left(\phi, \frac{1}{T}(1 - \phi^2)\right)$$

and it follows that $t_{\phi=1} \stackrel{A}{\sim} N(0, 1)$. However, under the null hypothesis of nonstationarity the above result gives

$$\hat{\phi} \stackrel{A}{\sim} N(1, 0)$$

which clearly does not make any sense. The problem is that under the unit root null, $\{y_t\}$ is not stationary and ergodic, and the usual sample moments do not converge to fixed constants. Instead, Phillips (1987) showed that the sample moments of $\{y_t\}$ converge to random functions of Brownian

¹The AR(1) model may be re-written as $\Delta y_t = \pi y_{t-1} + u_t$ where $\pi = \phi - 1$. Testing $\phi = 1$ is then equivalent to testing $\pi = 0$. Unit root tests are often computed using this alternative regression and the **S+FinMetrics** function **unitroot** follows this convention.

motion²:

$$\begin{aligned} T^{-3/2} \sum_{t=1}^T y_{t-1} &\xrightarrow{d} \sigma \int_0^1 W(r) dr \\ T^{-2} \sum_{t=1}^T y_{t-1}^2 &\xrightarrow{d} \sigma^2 \int_0^1 W(r)^2 dr \\ T^{-1} \sum_{t=1}^T y_{t-1} \varepsilon_t &\xrightarrow{d} \sigma^2 \int_0^1 W(r) dW(r) \end{aligned}$$

where $W(r)$ denotes a standard Brownian motion (Wiener process) defined on the unit interval. Using the above results Phillips showed that under the unit root null $H_0 : \phi = 1$

$$T(\hat{\phi} - 1) \xrightarrow{d} \frac{\int_0^1 W(r) dW(r)}{\int_0^1 W(r)^2 dr} \quad (4.1)$$

$$t_{\phi=1} \xrightarrow{d} \frac{\int_0^1 W(r) dW(r)}{\left(\int_0^1 W(r)^2 dr \right)^{1/2}} \quad (4.2)$$

The above yield some surprising results:

- $\hat{\phi}$ is *super-consistent*; that is, $\hat{\phi} \xrightarrow{P} \phi$ at rate T instead of the usual rate $T^{1/2}$.
- $\hat{\phi}$ is not asymptotically normally distributed and $t_{\phi=1}$ is not asymptotically standard normal.
- The limiting distribution of $t_{\phi=1}$ is called the Dickey-Fuller (DF) distribution and does not have a closed form representation. Consequently, quantiles of the distribution must be computed by numerical approximation or by simulation³.
- Since the *normalized bias* $T(\hat{\phi} - 1)$ has a well defined limiting distribution that does not depend on nuisance parameters it can also be used as a test statistic for the null hypothesis $H_0 : \phi = 1$.

²A Wiener process $W(\cdot)$ is a continuous-time stochastic process, associating each date $r \in [0, 1]$ a scalar random variable $W(r)$ that satisfies: (1) $W(0) = 0$; (2) for any dates $0 \leq t_1 \leq \dots \leq t_k \leq 1$ the changes $W(t_2) - W(t_1), W(t_3) - W(t_2), \dots, W(t_k) - W(t_{k-1})$ are independent normal with $W(s) - W(t) \sim N(0, (s - t))$; (3) $W(s)$ is continuous in s .

³Dickey and Fuller (1979) first considered the unit root tests and derived the asymptotic distribution of $t_{\phi=1}$. However, their representation did not utilize functions of Wiener processes.

4.3.1 Simulating the DF and Normalized Bias Distributions

As mentioned above, the DF and normalized bias distributions must be obtained by simulation methods. To illustrate, the following S-PLUS function **wiener** produces one random draw from the functions of Brownian motion that appear in the limiting distributions of $t_{\phi=1}$ and $T(\hat{\phi} - 1)$:

```
wiener = function(nobs) {
  e = rnorm(nobs)
  y = cumsum(e)
  ym1 = y[1:(nobs-1)]
  intW2 = nobs^(-2) * sum(ym1^2)
  intWdW = nobs^(-1) * sum(ym1*e[2:nobs])
  ans = list(intW2=intW2,
             intWdW=intWdW)
  ans
}
```

A simple loop then produces the simulated distributions:

```
> nobs = 1000
> nsim = 1000
> NB = rep(0,nsim)
> DF = rep(0,nsim)
> for (i in 1:nsim) {
+   BN.moments = wiener(nobs)
+   NB[i] = BN.moments$intWdW/BN.moments$intW2
+   DF[i] = BN.moments$intWdW/sqrt(BN.moments$intW2)
}
```

Figure 4.2 shows the histograms and density estimates of the simulated distributions. The DF density is slightly left-skewed relative to the standard normal, and the normalized bias density is highly left skewed and non-normal. Since the alternative is one-sided, the test statistics reject if they are sufficiently negative. For the DF and normalized bias densities the empirical 1%, 5% and 10% quantiles are

```
> quantile(DF,probs=c(0.01,0.05,0.1))
    1%      5%     10%
-2.451 -1.992 -1.603
> quantile(NB,probs=c(0.01,0.05,0.1))
    1%      5%     10%
-11.94 -8.56 -5.641
```

For comparison purposes, note that the 5% quantile from the standard normal distribution is -1.645.

The simulation of critical values and p-values from (4.1) and (4.2) is straightforward but time consuming. The **punitroot** and **qunitroot** func-

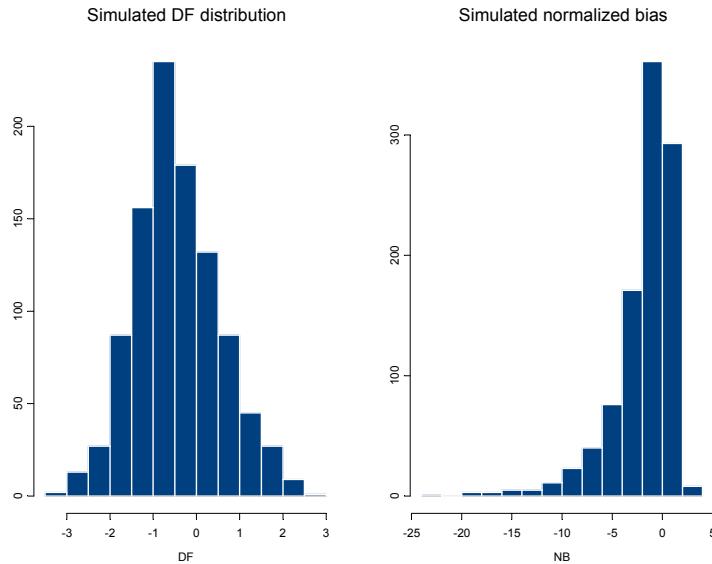


FIGURE 4.2. Histograms of simulated DF and normalized bias distributions.

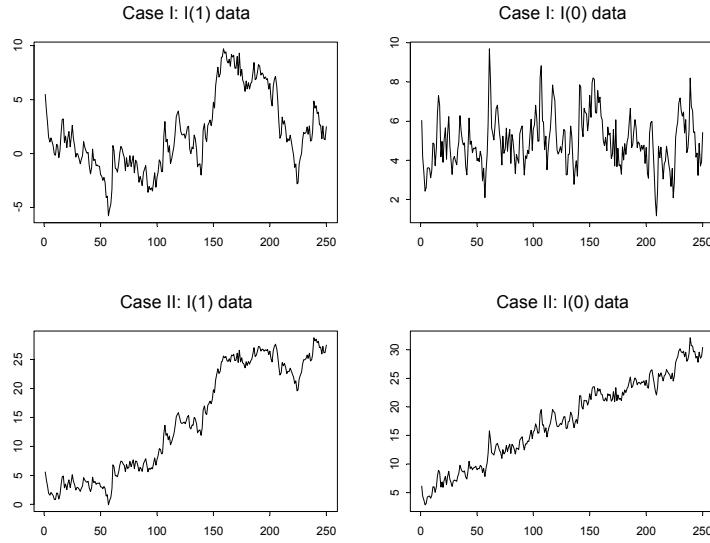
tions in **S+FinMetrics** produce p-values and quantiles of the DF and normalized bias distributions based on MacKinnon's (1996) response surface methodology. The advantage of the response surface methodology is that accurate p-values and quantiles appropriate for a given sample size can be produced. For example, the 1%, 5% and 10% quantiles for (4.2) and (4.1) based on a sample size of 100 are

```
> qunitroot(c(0.01,0.05,0.10), trend="nc", statistic="t",
+ n.sample=100)
[1] -2.588 -1.944 -1.615
> qunitroot(c(0.01,0.05,0.10), trend="nc", statistic="n",
+ n.sample=100)
[1] -13.086 -7.787 -5.565
```

The argument **trend="nc"** specifies that no constant is included in the test regression. Other valid options are **trend="c"** for constant only and **trend="ct"** for constant and trend. These trend cases are explained below. To specify the normalized bias distribution, set **statistic="n"**. For asymptotic quantiles set **n.sample=0**.

Similarly, the p-value of -1.645 based on the DF distribution for a sample size of 100 is computed as

```
> punitroot(-1.645, trend="nc", statistic="t")
[1] 0.0945
```

FIGURE 4.3. Simulated $I(1)$ and $I(0)$ data under trend cases I and II.

4.3.2 Trend Cases

When testing for unit roots, it is crucial to specify the null and alternative hypotheses appropriately to characterize the trend properties of the data at hand. For example, if the observed data does not exhibit an increasing or decreasing trend, then the appropriate null and alternative hypotheses should reflect this. The trend properties of the data *under the alternative hypothesis* will determine the form of the test regression used. Furthermore, the type of deterministic terms in the test regression will influence the asymptotic distributions of the unit root test statistics. The two most common trend cases are summarized below and illustrated in Figure 4.3.

Case I: Constant Only

The test regression is

$$y_t = c + \phi y_{t-1} + \varepsilon_t$$

and includes a constant to capture the nonzero mean under the alternative. The hypotheses to be tested are

$$\begin{aligned} H_0 &: \phi = 1 \Rightarrow y_t \sim I(1) \text{ without drift} \\ H_1 &: |\phi| < 1 \Rightarrow y_t \sim I(0) \text{ with nonzero mean} \end{aligned}$$

This formulation is appropriate for non-trending financial series like interest rates, exchange rates, and spreads. The test statistics $t_{\phi=1}$ and $T(\hat{\phi} - 1)$

are computed from the above regression. Under $H_0 : \phi = 1$ the asymptotic distributions of these test statistics are different from (4.2) and (4.1) and are influenced by the presence but not the coefficient value of the constant in the test regression. Quantiles and p-values for these distributions can be computed using the **S+FinMetrics** functions **punitroot** and **qunitroot** with the **trend="c"** option:

```
> qunitroot(c(0.01,0.05,0.10), trend="c", statistic="t",
+ n.sample=100)
[1] -3.497 -2.891 -2.582
> qunitroot(c(0.01,0.05,0.10), trend="c", statistic="n",
+ n.sample=100)
[1] -19.49 -13.53 -10.88
> punitroot(-1.645, trend="c", statistic="t", n.sample=100)
[1] 0.456
> punitroot(-1.645, trend="c", statistic="n", n.sample=100)
[1] 0.8172
```

For a sample size of 100, the 5% left tail critical values for $t_{\phi=1}$ and $T(\hat{\phi} - 1)$ are -2.891 and -13.53, respectively, and are quite a bit smaller than the 5% critical values computed when **trend="nc"**. Hence, inclusion of a constant pushes the distributions of $t_{\phi=1}$ and $T(\hat{\phi} - 1)$ to the left.

Case II: Constant and Time Trend

The test regression is

$$y_t = c + \delta t + \phi y_{t-1} + \varepsilon_t$$

and includes a constant and deterministic time trend to capture the deterministic trend under the alternative. The hypotheses to be tested are

$$\begin{aligned} H_0 &: \phi = 1 \Rightarrow y_t \sim I(1) \text{ with drift} \\ H_1 &: |\phi| < 1 \Rightarrow y_t \sim I(0) \text{ with deterministic time trend} \end{aligned}$$

This formulation is appropriate for trending time series like asset prices or the levels of macroeconomic aggregates like real GDP. The test statistics $t_{\phi=1}$ and $T(\hat{\phi} - 1)$ are computed from the above regression. Under $H_0 : \phi = 1$ the asymptotic distributions of these test statistics are different from (4.2) and (4.1) and are influenced by the presence but not the coefficient values of the constant and time trend in the test regression. Quantiles and p-values for these distributions can be computed using the **S+FinMetrics** functions **punitroot** and **qunitroot** with the **trend="ct"** option:

```
> qunitroot(c(0.01,0.05,0.10), trend="ct", statistic="t",
+ n.sample=100)
[1] -4.052 -3.455 -3.153
```

```

> qunitroot(c(0.01,0.05,0.10), trend="ct", statistic="n",
+ n.sample=100)
[1] -27.17 -20.47 -17.35
> punitroot(-1.645, trend="ct", statistic="t", n.sample=100)
[1] 0.7679
> punitroot(-1.645, trend="ct", statistic="n", n.sample=100)
[1] 0.9769

```

Notice that the inclusion of a constant and trend in the test regression further shifts the distributions of $t_{\phi=1}$ and $T(\hat{\phi}-1)$ to the left. For a sample size of 100, the 5% left tail critical values for $t_{\phi=1}$ and $T(\hat{\phi}-1)$ are now -3.455 and -20.471.

4.3.3 Dickey-Fuller Unit Root Tests

The unit root tests described above are valid if the time series y_t is well characterized by an AR(1) with white noise errors. Many financial time series, however, have a more complicated dynamic structure than is captured by a simple AR(1) model. Said and Dickey (1984) augment the basic autoregressive unit root test to accommodate general ARMA(p, q) models with unknown orders and their test is referred to as the *augmented Dickey-Fuller* (ADF) test. The ADF test tests the null hypothesis that a time series y_t is $I(1)$ against the alternative that it is $I(0)$, assuming that the dynamics in the data have an ARMA structure. The ADF test is based on estimating the test regression

$$y_t = \beta' \mathbf{D}_t + \phi y_{t-1} + \sum_{j=1}^p \psi_j \Delta y_{t-j} + \varepsilon_t \quad (4.3)$$

where \mathbf{D}_t is a vector of deterministic terms (constant, trend etc.). The p lagged difference terms, Δy_{t-j} , are used to approximate the ARMA structure of the errors, and the value of p is set so that the error ε_t is serially uncorrelated. The error term is also assumed to be homoskedastic. The specification of the deterministic terms depends on the assumed behavior of y_t under the alternative hypothesis of trend stationarity as described in the previous section. Under the null hypothesis, y_t is $I(1)$ which implies that $\phi = 1$. The ADF t-statistic and normalized bias statistic are based on the least squares estimates of (4.3) and are given by

$$\begin{aligned} ADF_t &= t_{\phi=1} = \frac{\hat{\phi} - 1}{SE(\phi)} \\ ADF_n &= \frac{T(\hat{\phi} - 1)}{1 - \hat{\psi}_1 - \dots - \hat{\psi}_p} \end{aligned}$$

An alternative formulation of the ADF test regression is

$$\Delta y_t = \beta' \mathbf{D}_t + \pi y_{t-1} + \sum_{j=1}^p \psi_j \Delta y_{t-j} + \varepsilon_t \quad (4.4)$$

where $\pi = \phi - 1$. Under the null hypothesis, Δy_t is $I(0)$ which implies that $\pi = 0$. The ADF t-statistic is then the usual t-statistic for testing $\pi = 0$ and the ADF normalized bias statistic is $T\hat{\pi}/(1 - \hat{\psi}_1 - \dots - \hat{\psi}_p)$. The test regression (4.4) is often used in practice because the ADF t-statistic is the usual t-statistic reported for testing the significance of the coefficient y_{t-1} . The **S+FinMetrics** function **unitroot** follows this convention.

Choosing the Lag Length for the ADF Test

An important practical issue for the implementation of the ADF test is the specification of the lag length p . If p is too small then the remaining serial correlation in the errors will bias the test. If p is too large then the power of the test will suffer. Ng and Perron (1995) suggest the following data dependent lag length selection procedure that results in stable size of the test and minimal power loss. First, set an upper bound p_{\max} for p . Next, estimate the ADF test regression with $p = p_{\max}$. If the absolute value of the t-statistic for testing the significance of the last lagged difference is greater than 1.6 then set $p = p_{\max}$ and perform the unit root test. Otherwise, reduce the lag length by one and repeat the process.

A useful rule of thumb for determining p_{\max} , suggested by Schwert (1989), is

$$p_{\max} = \left[12 \cdot \left(\frac{T}{100} \right)^{1/4} \right] \quad (4.5)$$

where $[x]$ denotes the integer part of x . This choice allows p_{\max} to grow with the sample so that the ADF test regressions (4.3) and (4.4) are valid if the errors follow an ARMA process with unknown order.

Example 19 Testing for a unit root in exchange rate data using ADF tests

To illustrate the ADF test procedure, consider testing for a unit root in the logarithm of the US/CA monthly spot exchange rate, denoted s_t , over the 30 year period 1976 - 1996. Figure 4.4 shows s_t , Δs_t as well as the sample autocorrelations for these series. The data and plots are created with the **S-PLUS** commands

```
> uscn.spot = lexrates.dat[, "USCNS"]
> uscn.spot@title = "Log US/CN spot exchange rate"
> par(mfrow=c(2,2))
> plot.timeSeries(uscn.spot, reference.grid=F,
+ main="Log of US/CN spot exchange rate")
```

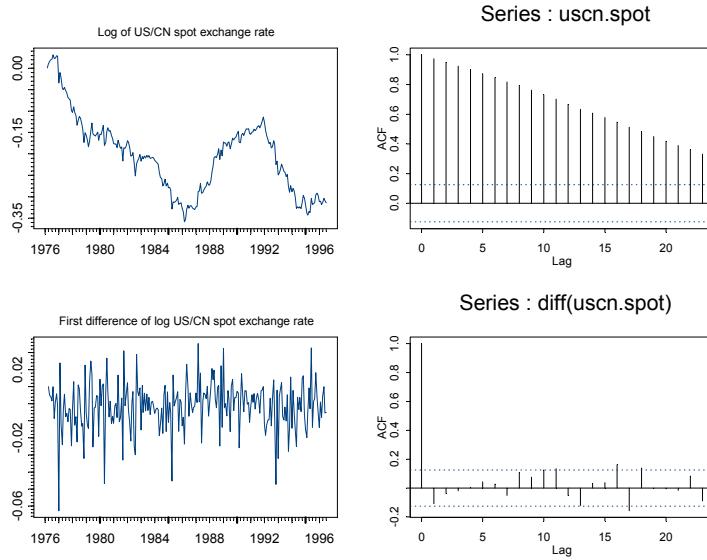


FIGURE 4.4. US/CN spot rate, first difference and SACF.

```
> xx = acf(uscn.spot)
> plot.timeSeries(diff(uscn.spot), reference.grid=F,
+ main="First difference of log US/CN spot exchange rate")
> xx = acf(diff(uscn.spot))
```

Clearly, s_t exhibits random walk like behavior with no apparent positive or negative drift. However, Δs_t behaves very much like a white noise process. The appropriate trend specification is to include a constant in the test regression. Regarding the maximum lag length for the Ng-Perron procedure, given the lack of serial correlation in Δs_t a conservative choice is $p_{\max} = 6$. The ADF t-statistic computed from the test regression with a constant and $p = 6$ lags can be computed using the **S+FinMetrics** function **unitroot** as follows

```
> adft.out = unitroot(uscn.spot, trend="c", statistic="t",
+ method="adf", lags=6)
> class(adft.out)
[1] "unitroot"
```

The output of **unitroot** is an object of class “**unitroot**” for which there are **print** and **summary** methods. Typing the name of the object invokes the **print** method and displays the basic test result

```
> adft.out
Test for Unit Root: Augmented DF Test
```

Null Hypothesis: there is a unit root

Type of Test: t-test

Test Statistic: -2.6

P-value: 0.09427

Coefficients:

lag1	lag2	lag3	lag4	lag5	lag6	constant
-0.0280	-0.1188	-0.0584	-0.0327	-0.0019	0.0430	-0.0075

Degrees of freedom: 239 total; 232 residual

Time period: from Aug 1976 to Jun 1996

Residual standard error: 0.01386

With $p = 6$ the ADF t-statistic is -2.6 and has a p-value (computed using `punitroot`) of 0.094. Hence we do not reject the unit root null at the 9.4% level. The small p-value here may be due to the inclusion of superfluous lags. To see the significance of the lags in the test regression, use the `summary` method

```
> summary(adft.out)
Test for Unit Root: Augmented DF Test
```

Null Hypothesis: there is a unit root

Type of Test: t test

Test Statistic: -2.6

P-value: 0.09427

Coefficients:

	Value	Std. Error	t value	Pr(> t)
lag1	-0.0280	0.0108	-2.6004	0.0099
lag2	-0.1188	0.0646	-1.8407	0.0669
lag3	-0.0584	0.0650	-0.8983	0.3700
lag4	-0.0327	0.0651	-0.5018	0.6163
lag5	-0.0019	0.0651	-0.0293	0.9766
lag6	0.0430	0.0645	0.6662	0.5060
constant	-0.0075	0.0024	-3.0982	0.0022

Regression Diagnostics:

R-Squared 0.0462

Adjusted R-Squared 0.0215

Durbin-Watson Stat 2.0033

Residual standard error: 0.01386 on 235 degrees of freedom

F-statistic: 1.874 on 6 and 232 degrees of freedom, the

```
p-value is 0.08619
Time period: from Aug 1976 to Jun 1996
```

The results indicate that too many lags have been included in the test regression. Following the Ng-Perron backward selection procedure $p = 2$ lags are selected. The results are

```
> adft.out = unitroot(uscn.spot, trend="c", lags=2)
> summary(adft.out)
```

Test for Unit Root: Augmented DF Test

```
Null Hypothesis: there is a unit root
Type of Test: t test
Test Statistic: -2.115
P-value: 0.2392
```

Coefficients:

	Value	Std. Error	t value	Pr(> t)
lag1	-0.0214	0.0101	-2.1146	0.0355
lag2	-0.1047	0.0635	-1.6476	0.1007
constant	-0.0058	0.0022	-2.6001	0.0099

Regression Diagnostics:

```
R-Squared 0.0299
Adjusted R-Squared 0.0218
Durbin-Watson Stat 2.0145
```

```
Residual standard error: 0.01378 on 239 degrees of freedom
F-statistic: 3.694 on 2 and 240 degrees of freedom, the
p-value is 0.02629
```

Time period: from Apr 1976 to Jun 1996

With 2 lags the ADF t-statistic is -2.115, the p-value 0.239 and we have greater evidence for a unit root in s_t . A similar result is found with the ADF normalized bias statistic

```
> adfn.out = unitroot(uscn.spot, trend="c", lags=2,
+ statistic="n")
> adfn.out
```

Test for Unit Root: Augmented DF Test

```
Null Hypothesis: there is a unit root
Type of Test: normalized test
Test Statistic: -5.193
P-value: 0.4129
```

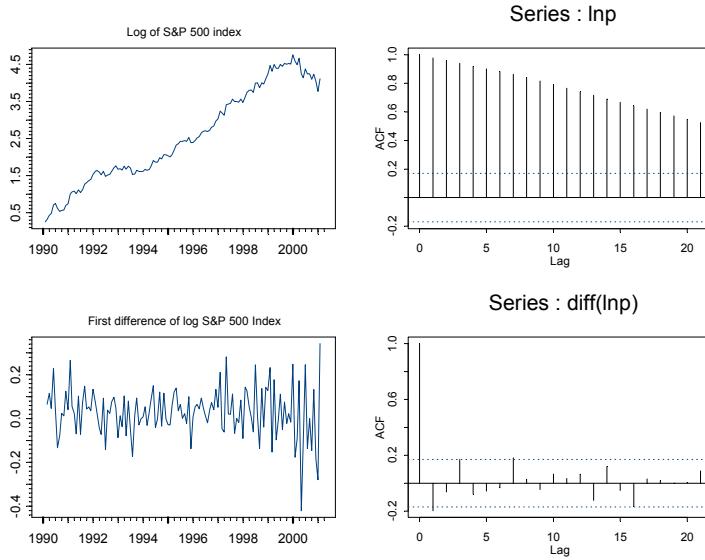


FIGURE 4.5. Log prices on the S&P 500 index, first difference and SACF.

Coefficients:

```
lag1    lag2 constant
-0.0214 -0.1047 -0.0058
```

Degrees of freedom: 243 total; 240 residual

Time period: from Apr 1976 to Jun 1996

Residual standard error: 0.01378

Example 20 Testing for a unit root in log stock prices

The log levels of asset prices are usually treated as $I(1)$ with drift. Indeed, the random walk model of stock prices is a special case of an $I(1)$ process. Consider testing for a unit root in the log of the monthly S&P 500 index, p_t , over the period January 1990 through January 2001. The data is taken from the **S+FinMetrics** “timeSeries” **singleIndex.dat**. The data and various plots are created with the **S-PLUS** commands

```
> lnp = log(singleIndex.dat[,1])
> lnp@title = "Log of S&P 500 Index"
> par(mfrow=c(2,2))
> plot.timeSeries(lnp, reference.grid=F,
+ main="Log of S&P 500 index")
> acf.plot(acf(lnp,plot=F))
> plot.timeSeries(diff(lnp), reference.grid=F,
```

```
+ main="First difference of log S&P 500 Index")
> acf.plot(acf(diff(lnp),plot=F))
```

and are illustrated in Figure 4.5. Clearly, the p_t is nonstationary due to the positive trend. Also, there appears to be some negative autocorrelation at lag one in Δp_t . The null hypothesis to be tested is that p_t is $I(1)$ with drift, and the alternative is that the p_t is $I(0)$ about a deterministic time trend. The ADF t-statistic to test these hypotheses is computed with a constant and time trend in the test regression and four lags of Δp_t (selecting using the Ng-Perron backward selection method)

```
> adft.out = unitroot(lnp, trend="ct", lags=4)
> summary(adft.out)
```

Test for Unit Root: Augmented DF Test

Null Hypothesis: there is a unit root
 Type of Test: t test
 Test Statistic: -1.315
 P-value: 0.8798

Coefficients:

	Value	Std. Error	t value	Pr(> t)
lag1	-0.0540	0.0410	-1.3150	0.1910
lag2	-0.1869	0.0978	-1.9111	0.0583
lag3	-0.0460	0.0995	-0.4627	0.6444
lag4	0.1939	0.0971	1.9964	0.0481
constant	0.1678	0.1040	1.6128	0.1094
time	0.0015	0.0014	1.0743	0.2848

Regression Diagnostics:

R-Squared 0.1016
 Adjusted R-Squared 0.0651
 Durbin-Watson Stat 1.9544

Residual standard error: 0.1087 on 125 degrees of freedom
 F-statistic: 2.783 on 5 and 123 degrees of freedom, the
 p-value is 0.0204
 Time period: from May 1990 to Jan 2001

$ADF_t = -1.315$ and has a p-value of 0.8798, so one clearly does not reject the null that p_t is $I(1)$ with drift.

4.3.4 Phillips-Perron Unit Root Tests

Phillips and Perron (1988) developed a number of unit root tests that have become popular in the analysis of financial time series. The Phillips-Perron (PP) unit root tests differ from the ADF tests mainly in how they deal with serial correlation and heteroskedasticity in the errors. In particular, where the ADF tests use a parametric autoregression to approximate the ARMA structure of the errors in the test regression, the PP tests ignore any serial correlation in the test regression. The test regression for the PP tests is

$$\Delta y_t = \beta' \mathbf{D}_t + \pi y_{t-1} + u_t$$

where u_t is $I(0)$ and may be heteroskedastic. The PP tests correct for any serial correlation and heteroskedasticity in the errors u_t of the test regression by directly modifying the test statistics $t_{\pi=0}$ and $T\hat{\pi}$. These modified statistics, denoted Z_t and Z_π , are given by

$$\begin{aligned} Z_t &= \left(\frac{\hat{\sigma}^2}{\hat{\lambda}^2} \right)^{1/2} \cdot t_{\pi=0} - \frac{1}{2} \left(\frac{\hat{\lambda}^2 - \hat{\sigma}^2}{\hat{\lambda}^2} \right) \cdot \left(\frac{T \cdot SE(\hat{\pi})}{\hat{\sigma}^2} \right) \\ Z_\pi &= T\hat{\pi} - \frac{1}{2} \frac{T^2 \cdot SE(\hat{\pi})}{\hat{\sigma}^2} (\hat{\lambda}^2 - \hat{\sigma}^2) \end{aligned}$$

The terms $\hat{\sigma}^2$ and $\hat{\lambda}^2$ are consistent estimates of the variance parameters

$$\begin{aligned} \sigma^2 &= \lim_{T \rightarrow \infty} T^{-1} \sum_{t=1}^T E[u_t^2] \\ \lambda^2 &= \lim_{T \rightarrow \infty} \sum_{t=1}^T E[T^{-1} S_T^2] \end{aligned}$$

where $S_T = \sum_{t=1}^T u_t$. The sample variance of the least squares residual \hat{u}_t is a consistent estimate of σ^2 , and the Newey-West long-run variance estimate of u_t using \hat{u}_t is a consistent estimate of λ^2 .

Under the null hypothesis that $\pi = 0$, the PP Z_t and Z_π statistics have the same asymptotic distributions as the ADF t-statistic and normalized bias statistics. One advantage of the PP tests over the ADF tests is that the PP tests are robust to general forms of heteroskedasticity in the error term u_t . Another advantage is that the user does not have to specify a lag length for the test regression.

Example 21 Testing for a unit root in exchange rates using the PP tests

Recall the arguments for the **S+FinMetrics** **unitroot** function are

```
> args(unitroot)
function(x, trend = "c", method = "adf",
```

```
statistic = "t", lags = 1, bandwidth = NULL,
window = "bartlett", asymptotic = F, na.rm = F)
```

The PP statistics may be computed by specifying the optional argument `method="pp"`. When `method="pp"` is chosen, the argument `window` specifies the weight function and the argument `bandwidth` determines the lag truncation parameter used in computing the long-run variance parameter λ^2 . The default bandwidth is the integer part of $(4 \cdot (T/100))^{2/9}$ where T is the sample size.

Now, consider testing for a unit root in the log of the US/CN spot exchange rate using the PP Z_t and Z_π statistics:

```
> unitroot(uscn.spot, trend="c", method="pp")
Test for Unit Root: Phillips-Perron Test

Null Hypothesis: there is a unit root
Type of Test: t-test
Test Statistic: -1.97
P-value: 0.2999

Coefficients:
lag1 constant
-0.0202 -0.0054

Degrees of freedom: 244 total; 242 residual
Time period: from Mar 1976 to Jun 1996
Residual standard error: 0.01383

> unitroot(uscn.spot, trend="c", method="pp", statistic="n")
Test for Unit Root: Phillips-Perron Test

Null Hypothesis: there is a unit root
Type of Test: normalized test
Test Statistic: -4.245
P-value: 0.5087

Coefficients:
lag1 constant
-0.0202 -0.0054

Degrees of freedom: 244 total; 242 residual
Time period: from Mar 1976 to Jun 1996
Residual standard error: 0.01383
```

As with the ADF tests, the PP tests do not reject the null that the log of the US/CN spot rate is $I(1)$ at any reasonable significance level.

4.4 Stationarity Tests

The ADF and PP unit root tests are for the null hypothesis that a time series y_t is $I(1)$. Stationarity tests, on the other hand, are for the null that y_t is $I(0)$. The most commonly used stationarity test, the KPSS test, is due to Kwiatkowski, Phillips, Schmidt and Shin (1992). They derive their test by starting with the model

$$\begin{aligned} y_t &= \beta' \mathbf{D}_t + \mu_t + u_t \\ \mu_t &= \mu_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim WN(0, \sigma_\varepsilon^2) \end{aligned} \quad (4.6)$$

where \mathbf{D}_t contains deterministic components (constant or constant plus time trend), u_t is $I(0)$ and may be heteroskedastic. Notice that μ_t is a pure random walk with innovation variance σ_ε^2 . The null hypothesis that y_t is $I(0)$ is formulated as $H_0 : \sigma_\varepsilon^2 = 0$, which implies that μ_t is a constant. Although not directly apparent, this null hypothesis also implies a unit moving average root in the ARMA representation of Δy_t . The KPSS test statistic is the Lagrange multiplier (LM) or score statistic for testing $\sigma_\varepsilon^2 = 0$ against the alternative that $\sigma_\varepsilon^2 > 0$ and is given by

$$KPSS = \left(T^{-2} \sum_{t=1}^T \hat{S}_t^2 \right) / \hat{\lambda}^2 \quad (4.7)$$

where $\hat{S}_t = \sum_{j=1}^t \hat{u}_j$, \hat{u}_t is the residual of a regression of y_t on \mathbf{D}_t and $\hat{\lambda}$ is a consistent estimate of the long-run variance of u_t using \hat{u}_t . Under the null that y_t is $I(0)$, Kwiatkowski, Phillips, Schmidt and Shin show that KPSS converges to a function of standard Brownian motion that depends on the form of the deterministic terms \mathbf{D}_t but not their coefficient values β . In particular, if $\mathbf{D}_t = 1$ then

$$KPSS \xrightarrow{d} \int_0^1 V_1(r) dr \quad (4.8)$$

where $V_1(r) = W(r) - rW(1)$ and $W(r)$ is a standard Brownian motion for $r \in [0, 1]$. If $\mathbf{D}_t = (1, t)'$ then

$$KPSS \xrightarrow{d} \int_0^1 V_2(r) dr \quad (4.9)$$

where $V_2(r) = W(r) + r(2 - 3r)W(1) + 6r(r^2 - 1) \int_0^1 W(s) ds$. Critical values from the asymptotic distributions (4.8) and (4.9) must be obtained by simulation methods, and these are summarized in Table 4.1.

The stationary test is a one-sided right-tailed test so that one rejects the null of stationarity at the $100 \cdot \alpha\%$ level if the KPSS test statistic (4.7) is greater than the $100 \cdot (1 - \alpha)\%$ quantile from the appropriate asymptotic distribution (4.8) or (4.9).

Distribution	Right tail quantiles				
	0.90	0.925	0.950	0.975	0.99
$\int_0^1 V_1(r)dr$	0.349	0.396	0.446	0.592	0.762
$\int_0^1 V_2(r)dr$	0.120	0.133	0.149	0.184	0.229

TABLE 4.1. Quantiles of the distribution of the KPSS statistic

4.4.1 Simulating the KPSS Distributions

The distributions in (4.8) and (4.9) may be simulated using methods similar to those used to simulate the DF distribution. The following S-PLUS code is used to create the quantiles in Table 4.1:

```

wiener2 = function(nobs) {
  e = rnorm(nobs)
  # create detrended errors
  e1 = e - mean(e)
  e2 = residuals(OLS(e~seq(1,nobs)))
  # compute simulated Brownian Bridges
  y1 = cumsum(e1)
  y2 = cumsum(e2)
  intW2.1 = nobs^(-2) * sum(y1^2)
  intW2.2 = nobs^(-2) * sum(y2^2)
  ans = list(intW2.1=intW2.1,
             intW2.2=intW2.2)
  ans
}
#
# simulate KPSS distributions
#
> nobs = 1000
> nsim = 10000
> KPSS1 = rep(0,nsim)
> KPSS2 = rep(0,nsim)
> for (i in 1:nsim) {
  BN.moments = wiener2(nobs)
  KPSS1[i] = BN.moments$intW2.1
  KPSS2[i] = BN.moments$intW2.2
}
#
# compute quantiles of distribution
#
> quantile(KPSS1, probs=c(0.90,0.925,0.95,0.975,0.99))
  90.0%   92.5%   95.0%   97.5%   99.0%
  0.34914 0.39634 0.46643 0.59155 0.76174
> quantile(KPSS2, probs=c(0.90,0.925,0.95,0.975,0.99))

```

```
90.0% 92.5% 95.0% 97.5% 99.0%
0.12003 0.1325 0.14907 0.1841 0.22923
```

Currently, only asymptotic critical values are available for the KPSS test.

4.4.2 Testing for Stationarity Using the *S+FinMetrics* Function **stationaryTest**

The **S+FinMetrics** function **stationaryTest** may be used to test the null hypothesis that a time series y_t is $I(0)$ based on the KPSS statistic (4.7). The function **stationaryTest** has arguments

```
> args(stationaryTest)
function(x, trend = "c", bandwidth = NULL, na.rm = F)
```

where **x** represents a univariate vector or “**timeSeries**”. The argument **trend** specifies the deterministic trend component in (4.6) and valid choices are “**c**” for a constant and “**ct**” for a constant and time trend. The argument **bandwidth** determines the lag truncation parameter used in computing the long-run variance parameter λ^2 . The default bandwidth is the integer part of $(4 \cdot (T/100))^{2/9}$ where T is the sample size. The output of **stationaryTest** is an object of class “**stationaryTest**” for which there is only a **print** method. The use of **stationaryTest** is illustrated with the following example.

Example 22 Testing for stationarity in exchange rates

Consider the US/CN spot exchange data used in the previous examples. To test the null that s_t is $I(0)$, the KPSS statistic is computed using a constant in (4.6):

```
> kpss.out = stationaryTest(uscn.spot, trend="c")
> class(kpss.out)
[1] "stationaryTest"
> kpss.out
```

Test for Stationarity: KPSS Test

Null Hypothesis: stationary around a constant

Test Statistics:

USCNS

1.6411**

* : significant at 5% level

** : significant at 1% level

```
Total Observ.: 245
Bandwidth : 5
```

The KPSS statistic is 1.641 and is greater than the 99% quantile, 0.762, from Table 4.1. Therefore, the null that s_t is $I(0)$ is rejected at the 1% level.

4.5 Some Problems with Unit Root Tests

The ADF and PP tests are asymptotically equivalent but may differ substantially in finite samples due to the different ways in which they correct for serial correlation in the test regression. In particular, Schwert (1989) finds that if Δy_t has an ARMA representation with a large and negative MA component, then the ADF and PP tests are severely size distorted (reject $I(1)$ null much too often when it is true) and that the PP tests are more size distorted than the ADF tests. Recently, Perron and Ng (1996) have suggested useful modifications to the PP tests to mitigate this size distortion. Caner and Killion (2001) have found similar problems with the KPSS test.

In general, the ADF and PP tests have very low power against $I(0)$ alternatives that are close to being $I(1)$. That is, unit root tests cannot distinguish highly persistent stationary processes from nonstationary processes very well. Also, the power of unit root tests diminish as deterministic terms are added to the test regressions. That is, tests that include a constant and trend in the test regression have less power than tests that only include a constant in the test regression. For maximum power against very persistent alternatives the recent tests proposed by Elliot, Rothenberg and Stock (1996) and Ng and Perron (2001) should be used. These tests are described in the next section.

4.6 Efficient Unit Root Tests

Assume that T observations are generated by

$$y_t = \beta' \mathbf{D}_t + u_t, \quad u_t = \phi u_{t-1} + v_t$$

where \mathbf{D}_t represents a vector of deterministic terms, $E[u_0] < \infty$, and v_t is a 1-summable linear process with long-run variance λ^2 . Typically $D_t = 1$ or $\mathbf{D}_t = [1, t]$. Consider testing the null hypothesis $\phi = 1$ versus $|\phi| < 1$. If the distribution of the data were known then the Neyman-Pearson Lemma gives the test with best power against any point alternative $\bar{\phi}$. The power of this optimal test plotted as a function of $\bar{\phi}$ gives an upper bound (envelope) for the power of any test based on the same distribution of the

data. An undesirable feature of this power envelope is that it depends on the specific value of $\bar{\phi}$, so that there is no uniformly most power full test that can be used against all alternatives $|\phi| < 1$. However, using asymptotic approximations based on the local-to-unity alternative $\phi = 1 + c/T$, for $c < 0$, Elliot, Rothenberg and Stock (2001) (hereafter ERS) derive a class of test statistics that come very close to the power envelope for a wide range of alternatives. These tests are referred to as *efficient unit root tests*, and they can have substantially higher power than the ADF or PP unit root tests especially when ϕ is close to unity.

4.6.1 Point Optimal Tests

The starting point for the class of efficient tests is the feasible test statistic that is optimal (asymptotically) for the point alternative $\bar{\phi} = 1 - \bar{c}/T$, $\bar{c} < 0$. This test is constructed as follows. Define the T -dimensional column vector \mathbf{y}_ϕ and $T \times q$ dimensional matrix \mathbf{D}_ϕ by

$$\begin{aligned}\mathbf{y}_\phi &= (y_1, y_2 - \phi y_1, \dots, y_T - \phi y_{T-1})' \\ \mathbf{D}_\phi &= (\mathbf{D}'_1, \mathbf{D}'_2 - \phi \mathbf{D}'_1, \dots, \mathbf{D}'_T - \phi \mathbf{D}'_{T-1})'\end{aligned}$$

All of the elements of \mathbf{y}_ϕ and \mathbf{D}_ϕ , except the first, are quasi-differenced using the operator $1 - \phi L$. Next, for any value of ϕ , define $S(\phi)$ as the sum of squared residuals from a least squares regression of \mathbf{y}_ϕ on \mathbf{D}_ϕ . That is,

$$S(\phi) = \tilde{\mathbf{y}}'_\phi \tilde{\mathbf{y}}_\phi$$

where $\tilde{\mathbf{y}}_\phi = \mathbf{y}_\phi - \mathbf{D}_\phi \hat{\beta}_\phi$ and $\hat{\beta}_\phi = (\mathbf{D}'_\phi \mathbf{D}_\phi)^{-1} \mathbf{D}'_\phi \mathbf{y}_\phi$. ERS show that the feasible *point optimal unit root test* against the alternative $\bar{\phi} = 1 - \bar{c}/T$ has the form

$$P_T = [S(\bar{\phi}) - \bar{\phi} S(1)] / \hat{\lambda}^2 \quad (4.10)$$

where $\hat{\lambda}^2$ is a consistent estimate of λ^2 . ERS derive the asymptotic distribution of P_T for $D_t = 1$ and $D_t = (1, t)$ and provide asymptotic and finite sample critical values for tests of size 1%, 2.5%, 5% and 10%⁴.

Through a series of simulation experiments, ERS discover that if $\bar{\phi} = 1 + \bar{c}/T$ is chosen such that the power of P_T is tangent to the power envelope at 50% power then the overall power of P_T , for a wide range of $\bar{\phi}$ values less than unity, is close to the power envelope. For a given sample size T , the value of $\bar{\phi}$ that results in P_T having 50% power depends on \bar{c} and the form of the deterministic terms in \mathbf{D}_t . ERS show that if $D_t = 1$ then $\bar{c} = -7$, and if $\mathbf{D}_t = (1, t)$ then $\bar{c} = -13.5$.

The ERS P_T statistic may be computed using the function `unitroot` with `method="ers"`.

⁴These critical values are given in ERS Table I panels A and B.

4.6.2 DF-GLS Tests

In the construction of the ERS feasible point optimal test (4.10), the unknown parameters β of the trend function are efficiently estimated under the alternative model with $\bar{\phi} = 1 + \bar{c}/T$. That is, $\hat{\beta}_{\bar{\phi}} = (\mathbf{D}'_{\bar{\phi}} \mathbf{D}_{\bar{\phi}})^{-1} \mathbf{D}'_{\bar{\phi}} \mathbf{y}_{\bar{\phi}}$. ERS use this insight to derive an efficient version of the ADF t-statistic, which they call the *DF-GLS test*. They construct this t-statistic as follows. First, using the trend parameters $\hat{\beta}_{\bar{\phi}}$ estimated under the alternative, define the detrended data

$$y_t^d = y_t - \hat{\beta}'_{\bar{\phi}} \mathbf{D}_t$$

ERS call this detrending procedure *GLS detrending*⁵. Next, using the GLS detrended data, estimate by least squares the ADF test regression which omits the deterministic terms

$$\Delta y_t^d = \pi y_{t-1}^d + \sum_{j=1}^p \psi_j \Delta y_{t-j}^d + \varepsilon_t \quad (4.11)$$

and compute the t-statistic for testing $\pi = 0$. When $D_t = 1$, ERS show that the asymptotic distribution of the DF-GLS test is the same as the ADF t-test, but has higher asymptotic power (against local alternatives) than the DF t-test. Furthermore, ERS show that the DF-GLS test has essentially the same asymptotic power as the ERS point optimal test when $\bar{c} = -7$. When $\mathbf{D}_t = (1, t)$ the asymptotic distribution of the DF-GLS test, however, is different from the ADF t-test. ERS and Ng and Perron (2001) provide critical values for the DF-GLS test in this case. ERS show that the DF-GLS test has the same asymptotic power as the ERS point optimal test with $c = -13.5$, and has higher power than the DF t-test against local alternatives.

The DF-GLS t-test may be computed using the function `unitroot` with `method="dfgls"`.

4.6.3 Modified Efficient PP Tests

Ng and Perron (2001) use the GLS detrending procedure of ERS to create efficient versions of the modified PP tests of Perron and Ng (1996). These *efficient modified PP tests* do not exhibit the severe size distortions of the PP tests for errors with large negative MA or AR roots, and they can have substantially higher power than the PP tests especially when ϕ is close to unity.

⁵For deterministically trending trend data with ergodic-stationary deviations from trend, Grenander's Theorem gives the result that least squares estimates of the trend parameters ignoring serial correlation are asymptotically equivalent to the generalized least squares estimates that incorporate serial correlation.

Using the GLS detrended data y_t^d , the efficient modified PP tests are defined as

$$\begin{aligned}\overline{MZ}_\alpha &= (T^{-1}y_T^d - \hat{\lambda}^2) \left(2T^{-2} \sum_{t=1}^T y_{t-1}^d \right)^{-1} \\ \overline{MSB} &= \left(T^{-2} \sum_{t=1}^T y_{t-1}^d / \hat{\lambda}^2 \right)^{1/2} \\ \overline{MZ}_t &= \overline{MZ}_\alpha \times \overline{MSB}\end{aligned}$$

The statistics \overline{MZ}_α and \overline{MZ}_t are efficient versions of the PP Z_α and Z_t tests that have much smaller size distortions in the presence of negative moving average errors. Ng and Perron derive the asymptotic distributions of these statistics under the local alternative $\phi = 1 - c/T$ for $D_t = 1$ and $\mathbf{D}_t = (1, t)$. In particular, they show that the asymptotic distribution of \overline{MZ}_t is the same as the DF-GLS t-test.

The statistic \overline{MZ}_t may be computed using the function `unitroot` with `method="mpp"`.

4.6.4 Estimating λ^2

Ng and Perron (2001) emphasize that the estimation of the long-run variance λ^2 has important implications for the finite sample behavior of the ERS point optimal test and the efficient modified PP tests. They stress that an autoregressive estimate of λ^2 should be used to achieve stable finite sample size. They recommend estimating λ^2 from the ADF test regression (4.11) based on the GLS detrended data:

$$\hat{\lambda}_{AR} = \frac{\hat{\sigma}_p^2}{\left(1 - \hat{\psi}(1)\right)^2}$$

where $\hat{\psi}(1) = \sum_{j=1}^p \hat{\psi}_j$ and $\hat{\sigma}_p^2 = (T - p)^{-1} \sum_{t=p+1}^T \hat{\varepsilon}_t^2$ are obtained from (4.11) by least squares estimation.

4.6.5 Choosing Lag Lengths to Achieve Good Size and Power

Ng and Perron also stress that good size and power properties of the all the efficient unit root tests rely on the proper choice of the lag length p used for specifying the test regression (4.11). They argue, however, that traditional model selection criteria such as AIC and BIC are not well suited for determining p with integrated data. Ng and Perron suggest the *modified information criteria* (MIC) that selects p as $p_{mic} = \arg \min_{p \leq p_{max}} MIC(p)$

where

$$\begin{aligned} MIC(p) &= \ln(\hat{\sigma}_p^2) + \frac{C_T(\tau_T(p) + p)}{T - p_{\max}} \\ \tau_T(p) &= \frac{\hat{\pi}^2 \sum_{t=p_{\max}+1}^T y_{t-1}^d}{\hat{\sigma}_p^2} \\ \hat{\sigma}_p^2 &= \frac{1}{T - p_{\max}} \sum_{t=p_{\max}+1}^T \hat{\varepsilon}_t^2 \end{aligned}$$

with $C_T > 0$ and $C_T/T \rightarrow 0$ as $T \rightarrow \infty$. The maximum lag, p_{\max} , may be set using (4.5). The modified AIC (MAIC) results when $C_T = 2$, and the modified BIC (MBIC) results when $C_T = \ln(T - p_{\max})$. Through a series of simulation experiments, Ng and Perron recommend selecting the lag length p by minimizing the MAIC.

Example 23 Efficient unit root tests

To illustrate the efficient unit root tests, consider testing for a unit root in the 30-day interest rate differential formed from the difference between monthly US and UK spot exchange rates:

```
> fd = lexrates.dat[, "USUKS"] - lexrates.dat[, "USUKF"]
> colIds(fd) = "USUKFD"
> fd$title = "US/UK 30-day interest rate differential"
```

The interest rate differential, its SACF, and the SACF of its first difference are depicted in Figure 4.6. The graphs clearly show that the interest rate differential has a high degree of persistence, and that there is little persistence in the first difference.

The ERS P_T test, DF-GLS t-test and Ng-Perron $\overline{M}\overline{Z}_t$ test all with $D_t = 1$ may be computed using the function `unitroot` as follows:

```
> ers = unitroot(fd, trend="c", method="ers", max.lags=12)
> dfgls = unitroot(fd, trend="c", method="dfgls", max.lags=12)
> mpp = unitroot(fd, trend="c", method="mpp", max.lags=12)
```

Since the optional argument `lags` is omitted, the lag length for the test regression (4.11) is determined by minimizing the MAIC with $p_{\max} = 12$ set by the optional argument `max.lags=12`. The results of the efficient unit root tests are:

```
> ers.test

Test for Unit Root: Elliott-Rothenberg-Stock Test

Null Hypothesis: there is a unit root
Test Statistic: 1.772**
```

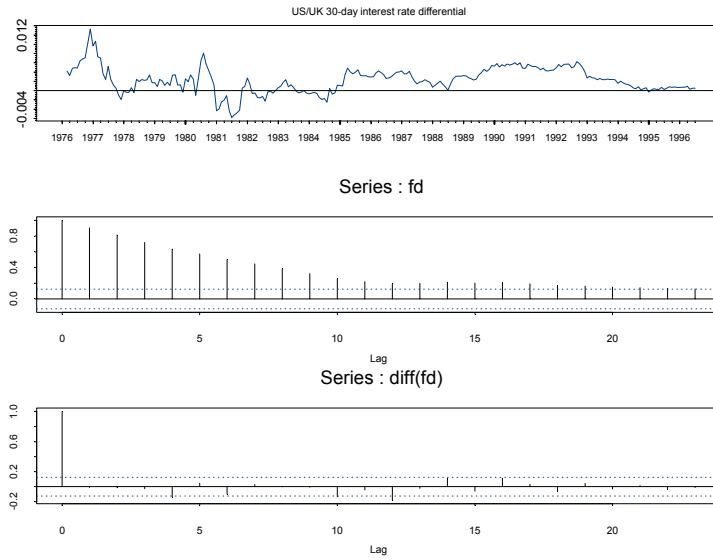


FIGURE 4.6. 30-day US/UK interest rate differential.

* : significant at 5% level
 ** : significant at 1% level

Coefficients:

lag1
 -0.07

Degrees of freedom: 244 total; 243 residual

Time period: from Mar 1976 to Jun 1996

Residual standard error: 0.00116

> dfgls.test

Test for Unit Root: DF Test with GLS detrending

Null Hypothesis: there is a unit root

Type of Test: t-test
 Test Statistic: -2.9205**
 * : significant at 5% level
 ** : significant at 1% level

Coefficients:

lag1

```
-0.07
```

Degrees of freedom: 244 total; 243 residual
 Time period: from Mar 1976 to Jun 1996
 Residual standard error: 0.00116

```
> mpp.test
```

Test for Unit Root: Modified Phillips-Perron Test

Null Hypothesis: there is a unit root
 Type of Test: t-test
 Test Statistic: -2.8226**
 * : significant at 5% level
 ** : significant at 1% level

Coefficients:

```
lag1  
-0.07
```

Degrees of freedom: 244 total; 243 residual
 Time period: from Mar 1976 to Jun 1996
 Residual standard error: 0.00116

Minimizing the MAIC gives $p = 0$, and with this lag length all tests reject the null hypothesis of a unit root at the 1% level.

4.7 References

- [1] CANER, M. AND L. KILIAN (2001). "Size Distortions of Tests of the Null Hypothesis of Stationarity: Evidence and Implications for the PPP Debate," *Journal of International Money and Finance*, 20, 639-657.
- [2] DICKEY, D. AND W. FULLER (1979). "Distribution of the Estimators for Autoregressive Time Series with a Unit Root," *Journal of the American Statistical Association*, 74, 427-431.
- [3] DICKEY, D. AND W. FULLER (1981). "Likelihood Ratio Statistics for Autoregressive Time Series with a Unit Root," *Econometrica*, 49, 1057-1072.
- [4] ELLIOT, G., T.J. ROTHENBERG, AND J.H. STOCK (1996). "Efficient Tests for an Autoregressive Unit Root," *Econometrica*, 64, 813-836.

- [5] FULLER, W. (1996). *Introduction to Statistical Time Series, Second Edition.* John Wiley, New York.
- [6] HAMILTON, J. (1994). *Time Series Analysis.* Princeton University Press, New Jersey.
- [7] HATANAKA, T. (1995). *Time-Series-Based Econometrics: Unit Roots and Co-Integration.* Oxford University Press, Oxford.
- [8] KWIATKOWSKI, D., P.C.B. PHILLIPS, P. SCHMIDT AND Y. SHIN (1992). “Testing the Null Hypothesis of Stationarity Against the Alternative of a Unit Root,” *Journal of Econometrics*, 54, 159-178.
- [9] MACKINNON, J. (1996). “Numerical Distribution Functions for Unit Root and Cointegration Tests,” *Journal of Applied Econometrics*, 11, 601-618.
- [10] MADDALA, G.S. AND I.-M. KIM (1998). *Unit Roots, Cointegration and Structural Change.* Oxford University Press, Oxford.
- [11] NG, S., AND P. PERRON (1995). “Unit Root Tests in ARMA Models with Data-Dependent Methods for the Selection of the Truncation Lag,” *Journal of the American Statistical Association*, 90, 268-281.
- [12] NG, S., AND P. PERRON (2001). “Lag Length Selection and the Construction of Unit Root Tests with Good Size and Power,” *Econometrica*, 69, 1519-1554.
- [13] PERRON, P. AND S. NG. (1996). “Useful Modifications to Some Unit Root Tests with Dependent Errors and their Local Asymptotic Properties,” *Review of Economic Studies*, 63, 435-463.
- [14] PHILLIPS, P.C.B. (1987). “Time Series Regression with a Unit Root,” *Econometrica*, 55, 227-301.
- [15] PHILLIPS, P.C.B. AND P. PERRON (1988). “Testing for Unit Roots in Time Series Regression,” *Biometrika*, 75, 335-346.
- [16] PHILLIPS, P.C.B. AND Z. XIAO (1998). “A Primer on Unit Root Testing,” *Journal of Economic Surveys*, 12, 423-470.
- [17] SCHWERT, W. (1989). “Test for Unit Roots: A Monte Carlo Investigation,” *Journal of Business and Economic Statistics*, 7, 147-159.
- [18] SAID, S.E. AND D. DICKEY (1984). “Testing for Unit Roots in Autoregressive Moving-Average Models with Unknown Order,” *Biometrika*, 71, 599-607.
- [19] STOCK, J.H. (1994). “Units Roots, Structural Breaks and Trends,” in R.F. Engle and D.L. McFadden (eds.), *Handbook of Econometrics, Volume IV.* North Holland, New York.

Panel Data Econometrics in R: The **plm** Package

Yves Croissant

Université de La Réunion

Giovanni Millo

University of Trieste and Generali SpA

Abstract

This introduction to the **plm** package is a slightly modified version of [Croissant and Millo \(2008\)](#), published in the Journal of Statistical Software.

Panel data econometrics is obviously one of the main fields in the profession, but most of the models used are difficult to estimate with R. **plm** is a package for R which intends to make the estimation of linear panel models straightforward. **plm** provides functions to estimate a wide variety of models and to make (robust) inference.

Keywords: panel data, covariance matrix estimators, generalized method of moments, R.

1. Introduction

Panel data econometrics is a continuously developing field. The increasing availability of data observed on cross-sections of units (like households, firms, countries etc.) *and* over time has given rise to a number of estimation approaches exploiting this double dimensionality to cope with some of the typical problems associated with economic data, first of all that of unobserved heterogeneity.

Timewise observation of data from different observational units has long been common in other fields of statistics (where they are often termed *longitudinal* data). In the panel data field as well as in others, the econometric approach is nevertheless peculiar with respect to experimental contexts, as it is emphasizing model specification and testing and tackling a number of issues arising from the particular statistical problems associated with economic data.

Thus, while a very comprehensive software framework for (among many other features) maximum likelihood estimation of linear regression models for longitudinal data, packages **nlme** ([Pinheiro, Bates, DebRoy, and the R Core team 2007](#)) and **lme4** ([Bates 2007](#)), is available in the R ([R Development Core Team 2008](#)) environment and can be used, e.g., for estimation of random effects panel models, its use is not intuitive for a practicing econometrician, and maximum likelihood estimation is only one of the possible approaches to panel data econometrics. Moreover, economic panel datasets often happen to be *unbalanced* (i.e., they have a different number of observations between groups), which case needs some adaptation to the methods and is not compatible with those in **nlme**. Hence the need for a package doing panel data “from the econometrician’s viewpoint” and featuring at a minimum the basic techniques econometricians are used to: random and fixed effects estimation of static linear panel data models, variable coefficients models, generalized method of moments estimation of dynamic models; and the basic toolbox of specification and misspecification diagnostics.

Furthermore, we felt there was the need for automation of some basic data management tasks such as lagging, summing and, more in general, applying (in the R sense) functions to the data, which, although conceptually simple, become cumbersome and error-prone on two-dimensional data, especially in the case of unbalanced panels.

This paper is organized as follows: Section 2 presents a very short overview of the typical model taxonomy¹. Section 3 discusses the software approach used in the package. The next three sections present the functionalities of the package in more detail: data management (Section 4), estimation (Section 5) and testing (Section 6), giving a short description and illustrating them with examples. Section 7 compares the approach in **plm** to that of **nlme** and **lme4**, highlighting the features of the latter two that an econometrician might find most useful. Section 8 concludes the paper.

2. The linear panel model

The basic linear panel models used in econometrics can be described through suitable restrictions of the following general model:

$$y_{it} = \alpha_{it} + \beta_{it}^\top x_{it} + u_{it} \quad (1)$$

where $i = 1, \dots, n$ is the individual (group, country ...) index, $t = 1, \dots, T$ is the time index and u_{it} a random disturbance term of mean 0.

Of course u_{it} is not estimable with $N = n \times T$ data points. A number of assumptions are usually made about the parameters, the errors and the exogeneity of the regressors, giving rise to a taxonomy of feasible models for panel data.

The most common one is parameter homogeneity, which means that $\alpha_{it} = \alpha$ for all i, t and $\beta_{it} = \beta$ for all i, t . The resulting model

$$y_{it} = \alpha + \beta^\top x_{it} + u_{it} \quad (2)$$

is a standard linear model pooling all the data across i and t .

To model individual heterogeneity, one often assumes that the error term has two separate components, one of which is specific to the individual and doesn't change over time². This is called the unobserved effects model:

$$y_{it} = \alpha + \beta^\top x_{it} + \mu_i + \epsilon_{it} \quad (3)$$

The appropriate estimation method for this model depends on the properties of the two error components. The idiosyncratic error ϵ_{it} is usually assumed well-behaved and independent of both the regressors x_{it} and the individual error component μ_i . The individual component may be in turn either independent of the regressors or correlated.

If it is correlated, the ordinary least squares (OLS) estimator of β would be inconsistent, so it is customary to treat the μ_i as a further set of n parameters to be estimated, as if in the

¹Comprehensive treatments are to be found in many econometrics textbooks, e.g. Baltagi (2001) or Wooldridge (2002): the reader is referred to these, especially to the first 9 chapters of Baltagi (2001).

²For the sake of exposition we are considering only the individual effects case here. There may also be time effects, which is a symmetric case, or both of them, so that the error has three components: $u_{it} = \mu_i + \lambda_t + \epsilon_{it}$.

general model $\alpha_{it} = \alpha_i$ for all t . This is called the fixed effects (a.k.a. *within* or *least squares dummy variables*) model, usually estimated by OLS on transformed data, and gives consistent estimates for β .

If the individual-specific component μ_i is uncorrelated with the regressors, a situation which is usually termed *random effects*, the overall error u_{it} also is, so the OLS estimator is consistent. Nevertheless, the common error component over individuals induces correlation across the composite error terms, making OLS estimation inefficient, so one has to resort to some form of feasible generalized least squares (GLS) estimators. This is based on the estimation of the variance of the two error components, for which there are a number of different procedures available.

If the individual component is missing altogether, pooled OLS is the most efficient estimator for β . This set of assumptions is usually labelled *pooling* model, although this actually refers to the errors' properties and the appropriate estimation method rather than the model itself. If one relaxes the usual hypotheses of well-behaved, white noise errors and allows for the idiosyncratic error ϵ_{it} to be arbitrarily heteroskedastic and serially correlated over time, a more general kind of feasible GLS is needed, called the *unrestricted* or *general* GLS. This specification can also be augmented with individual-specific error components possibly correlated with the regressors, in which case it is termed *fixed effects* GLS.

Another way of estimating unobserved effects models through removing time-invariant individual components is by first-differencing the data: lagging the model and subtracting, the time-invariant components (the intercept and the individual error component) are eliminated, and the model

$$\Delta y_{it} = \beta^\top \Delta x_{it} + \Delta u_{it} \quad (4)$$

(where $\Delta y_{it} = y_{it} - y_{i,t-1}$, $\Delta x_{it} = x_{it} - x_{i,t-1}$ and, from (3), $\Delta u_{it} = u_{it} - u_{i,t-1} = \Delta \epsilon_{it}$ for $t = 2, \dots, T$) can be consistently estimated by pooled OLS. This is called the *first-difference*, or FD estimator. Its relative efficiency, and so reasons for choosing it against other consistent alternatives, depends on the properties of the error term. The FD estimator is usually preferred if the errors u_{it} are strongly persistent in time, because then the Δu_{it} will tend to be serially uncorrelated.

Lastly, the *between* model, which is computed on time (group) averages of the data, discards all the information due to intragroup variability but is consistent in some settings (e.g., non-stationarity) where the others are not, and is often preferred to estimate long-run relationships. Variable coefficients models relax the assumption that $\beta_{it} = \beta$ for all i, t . Fixed coefficients models allow the coefficients to vary along one dimension, like $\beta_{it} = \beta_i$ for all t . Random coefficients models instead assume that coefficients vary randomly around a common average, as $\beta_{it} = \beta + \eta_i$ for all t , where η_i is a group- (time-) specific effect with mean zero.

The hypotheses on parameters and error terms (and hence the choice of the most appropriate estimator) are usually tested by means of:

- *pooling* tests to check poolability, i.e. the hypothesis that the same coefficients apply across all individuals,
- if the homogeneity assumption over the coefficients is established, the next step is to establish the presence of unobserved effects, comparing the null of spherical residuals with the alternative of group (time) specific effects in the error term,

- the choice between fixed and random effects specifications is based on Hausman-type tests, comparing the two estimators under the null of no significant difference: if this is not rejected, the more efficient random effects estimator is chosen,
- even after this step, departures of the error structure from sphericity can further affect inference, so that either screening tests or robust diagnostics are needed.

Dynamic models and in general lack of strict exogeneity of the regressors, pose further problems to estimation which are usually dealt with in the generalized method of moments (GMM) framework.

These were, in our opinion, the basic requirements of a panel data econometrics package for the R language and environment. Some, as often happens with R, were already fulfilled by packages developed for other branches of computational statistics, while others (like the fixed effects or the between estimators) were straightforward to compute after transforming the data, but in every case there were either language inconsistencies w.r.t. the standard econometric toolbox or subtleties to be dealt with (like, for example, appropriate computation of standard errors for the demeaned model, a common pitfall), so we felt there was need for an “all in one” econometrics-oriented package allowing to make specification searches, estimation and inference in a natural way.

3. Software approach

3.1. Data structure

Panel data have a special structure: each row of the data corresponds to a specific individual and time period. In **plm** the **data** argument may be an ordinary **data.frame** but, in this case, an argument called **index** has to be added to indicate the structure of the data. This can be:

- **NULL** (the default value), it is then assumed that the first two columns contain the individual and the time index and that observations are ordered by individual and by time period,
- a character string, which should be the name of the individual index,
- a character vector of length two containing the names of the individual and the time index,
- an integer which is the number of individuals (only in case of a balanced panel with observations ordered by individual).

The **pdata.frame** function is then called internally, which returns a **pdata.frame** which is a **data.frame** with an attribute called **index**. This attribute is a **DATA.FRAME** that contains the individual and the time indexes.

It is also possible to use directly the **pdata.frame** function and then to use the **pdata.frame** in the estimation functions.

3.2. Interface

Estimation interface

plm provides four functions for estimation:

- **plm**: estimation of the basic panel models, *i.e.* within, between and random effect models. Models are estimated using the **lm** function to transformed data,
- **pvcm**: estimation of models with variable coefficients,
- **pgmm**: estimation of generalized method of moments models,
- **pggls**: estimation of general feasible generalized least squares models.

The interface of these functions is consistent with the **lm()** function. Namely, their first two arguments are **formula** and **data** (which should be a **data.frame** and is mandatory). Three additional arguments are common to these functions:

- **index**: this argument enables the estimation functions to identify the structure of the data, *i.e.* the individual and the time period for each observation,
- **effect**: the kind of effects to include in the model, *i.e.* individual effects, time effects or both³,
- **model**: the kind of model to be estimated, most of the time a model with fixed effects or a model with random effects.

The results of these four functions are stored in an object which class has the same name of the function. They all inherit from class **panelmodel**. A **panelmodel** object contains: **coefficients**, **residuals**, **fitted.values**, **vcov**, **df.residual** and **call** and functions that extract these elements are provided.

Testing interface

The diagnostic testing interface provides both **formula** and **panelmodel** methods for most functions, with some exceptions. The user may thus choose whether to employ results stored in a previously estimated **panelmodel** object or to re-estimate it for the sake of testing.

Although the first strategy is the most efficient one, diagnostic testing on panel models mostly employs OLS residuals from pooling model objects, whose estimation is computationally inexpensive. Therefore most examples in the following are based on **formula** methods, which are perhaps the cleanest for illustrative purposes.

³Although in most models the individual and time effects cases are symmetric, there are exceptions: estimating the **fd** model on time effects is meaningless because cross-sections do not generally have a natural ordering, so trying **effect="time"** stops with an error message as does **effect="twoways"** which is not defined for **fd** models.

3.3. Computational approach to estimation

The feasible GLS methods needed for efficient estimation of unobserved effects models have a simple closed-form solution: once the variance components have been estimated and hence the covariance matrix of errors \hat{V} , model parameters can be estimated as

$$\hat{\beta} = (X^\top \hat{V}^{-1} X)^{-1} (X^\top \hat{V}^{-1} y) \quad (5)$$

Nevertheless, in practice plain computation of $\hat{\beta}$ has long been an intractable problem even for moderate-sized datasets because of the need to invert the $N \times N \hat{V}$ matrix. With the advances in computer power, this is no more so, and it is possible to program the “naive” estimator (5) in R with standard matrix algebra operators and have it working seamlessly for the standard “guinea pigs”, e.g. the Grunfeld data. Estimation with a couple of thousands of data points also becomes feasible on a modern machine, although excruciatingly slow and definitely not suitable for everyday econometric practice. Memory limits would also be very near because of the storage needs related to the huge \hat{V} matrix. An established solution exists for the random effects model which reduces the problem to an ordinary least squares computation.

The (quasi-)demeaning framework

The estimation methods for the basic models in panel data econometrics, the pooled OLS, random effects and fixed effects (or within) models, can all be described inside the OLS estimation framework. In fact, while pooled OLS simply pools data, the standard way of estimating fixed effects models with, say, group (time) effects entails transforming the data by subtracting the average over time (group) to every variable, which is usually termed *time-demeaning*. In the random effects case, the various feasible GLS estimators which have been put forth to tackle the issue of serial correlation induced by the group-invariant random effect have been proven to be equivalent (as far as estimation of β s is concerned) to OLS on *partially demeaned* data, where partial demeaning is defined as:

$$y_{it} - \theta \bar{y}_i = (X_{it} - \theta \bar{X}_i) \beta + (u_{it} - \theta \bar{u}_i) \quad (6)$$

where $\theta = 1 - [\sigma_u^2 / (\sigma_u^2 + T\sigma_e^2)]^{1/2}$, \bar{y} and \bar{X} denote time means of y and X , and the disturbance $v_{it} - \theta \bar{v}_i$ is homoskedastic and serially uncorrelated. Thus the feasible RE estimate for β may be obtained estimating $\hat{\theta}$ and running an OLS regression on the transformed data with `lm()`. The other estimators can be computed as special cases: for $\theta = 1$ one gets the fixed effects estimator, for $\theta = 0$ the pooled OLS one.

Moreover, instrumental variable estimators of all these models may also be obtained using several calls to `lm()`.

For this reason the three above estimators have been grouped inside the same function.

On the output side, a number of diagnostics and a very general coefficients’ covariance matrix estimator also benefits from this framework, as they can be readily calculated applying the standard OLS formulas to the demeaned data, which are contained inside `plm` objects. This will be the subject of Subsection 3.4.

The object oriented approach to general GLS computations

The covariance matrix of errors in general GLS models is too generic to fit the quasi-demeaning framework, so this method calls for a full-blown application of GLS as in (5). On the other hand, this estimator relies heavily on n -asymptotics, making it theoretically most suitable for situations which forbid it computationally: e.g., “short” micropanels with thousands of individuals observed over few time periods.

R has general facilities for fast matrix computation based on object orientation: particular types of matrices (symmetric, sparse, dense etc.) are assigned the relevant class and the additional information on structure is used in the computations, sometimes with dramatic effects on performance (see Bates 2004) and packages **Matrix** (see Bates and Maechler 2016) and **SparseM** (see Koenker and Ng 2016). Some optimized linear algebra routines are available in the R package **bdsmatrix** (see Therneau 2014) which exploit the particular block-diagonal and symmetric structure of \hat{V} making it possible to implement a fast and reliable full-matrix solution to problems of any practically relevant size.

The \hat{V} matrix is constructed as an object of class **bdsmatrix**. The peculiar properties of this matrix class are used for efficiently storing the object in memory and then by ad-hoc versions of the **solve** and **crossprod** methods, dramatically reducing computing times and memory usage. The resulting matrix is then used “the naive way” as in (5) to compute $\hat{\beta}$, resulting in speed comparable to that of the demeaning solution.

3.4. Inference in the panel model

General frameworks for restrictions and linear hypotheses testing are available in the R environment⁴. These are based on the Wald test, constructed as $\hat{\beta}^\top \hat{V}^{-1} \hat{\beta}$, where $\hat{\beta}$ and \hat{V} are consistent estimates of β and $V(\beta)$. The Wald test may be used for zero-restriction (i.e., significance) testing and, more generally, for linear hypotheses in the form $(R\hat{\beta} - r)^\top [R\hat{V}R^\top]^{-1} (R\hat{\beta} - r)$ ⁵. To be applicable, the test functions require extractor methods for coefficients’ and covariance matrix estimates to be defined for the model object to be tested. Model objects in **plm** all have **coef()** and **vcov()** methods and are therefore compatible with the above functions.

In the same framework, robust inference is accomplished substituting (“plugging in”) a robust estimate of the coefficient covariance matrix into the Wald statistic formula. In the panel context, the estimator of choice is the White system estimator. This called for a flexible method for computing robust coefficient covariance matrices *à la White* for **plm** objects.

A general White system estimator for panel data is:

$$\hat{V}_R(\beta) = (X^\top X)^{-1} \sum_{i=1}^n X_i^\top E_i X_i (X^\top X)^{-1} \quad (7)$$

where E_i is a function of the residuals \hat{e}_{it} , $t = 1, \dots, T$ chosen according to the relevant heteroskedasticity and correlation structure. Moreover, it turns out that the White covariance matrix calculated on the demeaned model’s regressors and residuals (both part of **plm** objects) is a consistent estimator of the relevant model’s parameters’ covariance matrix, thus the

⁴See packages **lmtest** (Hothorn, Zeileis, Farebrother, Cummins, Millo, and Mitchell 2015) and **car** (Fox 2016).

⁵Moreover, **coeftest()** provides a compact way of looking at coefficient estimates and significance diagnostics.

method is readily applicable to models estimated by random or fixed effects, first difference or pooled OLS methods. Different pre-weighting schemes taken from package **sandwich** (see [Zeileis 2004](#); [Lumley and Zeileis 2015](#)) are also implemented to improve small-sample performance. Robust estimators with any combination of covariance structures and weighting schemes can be passed on to the testing functions.

4. Managing data and formulae

The package is now illustrated by application to some well-known examples. It is loaded using

```
R> library("plm")
```

The four datasets used are **EmplUK** which was used by [Arellano and Bond \(1991\)](#), the **Grunfeld** data ([Kleiber and Zeileis 2008](#)) which is used in several econometric books, the **Produc** data used by [Munnell \(1990\)](#) and the **Wages** used by [Cornwell and Rupert \(1988\)](#).

```
R> data("EmplUK", package="plm")
R> data("Produc", package="plm")
R> data("Grunfeld", package="plm")
R> data("Wages", package="plm")
```

4.1. Data structure

As observed above, the current version of **plm** is capable of working with a regular **data.frame** without any further transformation, provided that the individual and time indexes are in the first two columns, as in all the example datasets but **Wages**. If this weren't the case, an **index** optional argument would have to be passed on to the estimating and testing functions.

```
R> head(Grunfeld)
```

	firm	year	inv	value	capital
1	1	1935	317.6	3078.5	2.8
2	1	1936	391.8	4661.7	52.6
3	1	1937	410.6	5387.1	156.9
4	1	1938	257.7	2792.2	209.2
5	1	1939	330.8	4313.2	203.4
6	1	1940	461.2	4643.9	207.2

```
R> E <- pdata.frame(EmplUK, index=c("firm", "year"), drop.index=TRUE, row.names=TRUE)
R> head(E)
```

	sector	emp	wage	capital	output
1-1977	7	5.041	13.1516	0.5894	95.7072
1-1978	7	5.600	12.3018	0.6318	97.3569
1-1979	7	5.015	12.8395	0.6771	99.6083
1-1980	7	4.715	13.8039	0.6171	100.5501
1-1981	7	4.093	14.2897	0.5076	99.5581
1-1982	7	3.166	14.8681	0.4229	98.6151

```
R> head(attr(E, "index"))
```

	firm	year
1	1	1977
2	1	1978
3	1	1979
4	1	1980
5	1	1981
6	1	1982

Two further arguments are logical : `drop.index` drop the indexes from the `data.frame` and `row.names` computes “fancy” row names by pasting the individual and the time indexes. While extracting a series from a `pdata.frame`, a `pseries` is created, which is the original series with the index attribute. This object has specific methods, like `summary` and `as.matrix` are provided. The former indicates the total variation of the variable and the share of this variation that is due to the individual and the time dimensions. The latter gives the matrix representation of the series, with, by default, individual as rows and time as columns.

```
R> summary(E$emp)
```

total sum of squares : 261539.4	
id	time
0.980765381	0.009108488

```
R> head(as.matrix(E$emp))
```

	1976	1977	1978	1979	1980	1981	1982	1983	1984
1	NA	5.041	5.600	5.015	4.715	4.093	3.166	2.936	NA
2	NA	71.319	70.643	70.918	72.031	73.689	72.419	68.518	NA
3	NA	19.156	19.440	19.900	20.240	19.570	18.125	16.850	NA
4	NA	26.160	26.740	27.280	27.830	27.169	24.504	22.562	NA
5	86.677	87.100	87.000	90.400	89.200	82.700	73.700	NA	NA
6	0.748	0.766	0.762	0.729	0.731	0.779	0.782	NA	NA

4.2. Data transformation

Panel data estimation requires to apply different transformations to raw series. If x is a series of length nT (where n is the number of individuals and T is the number of time periods), the transformed series \tilde{x} is obtained as $\tilde{x} = Mx$ where M is a transformation matrix. Denoting j a vector of one of length T and I_n the identity matrix of dimension n , we get:

- the between transformation: $P = \frac{1}{T}I_n \otimes jj'$ returns a vector containing the individual means. The `Between` and `between` functions performs this operation, the first one returning a vector of length nT , the second one a vector of length n ,
- the within transformation: $Q = I_{nT} - P$ returns a vector containing the values in deviation from the individual means. The `Within` function performs this operation.

- the first difference transformation $D = I_n \otimes d$ where

$$d = \begin{pmatrix} 1 & -1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & -1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & -1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 & -1 \end{pmatrix}$$

is of dimension $(T - 1, T)$.

Note that R's `diff()` and `lag()` functions don't compute correctly these transformations for panel data because they are unable to identify when there is a change in individual in the data. Specific methods for `pseries` objects are therefore have been rewritten in order to handle correctly panel data. Note that compares to the `lag` method for `ts` objects, the order of lags are indicated by positive integer. Moreover, 0 is a relevant value and a vector argument may be provided:

```
R> head(lag(E$emp, 0:2))
```

	0	1	2
1-1977	5.041	NA	NA
1-1978	5.600	5.041	NA
1-1979	5.015	5.600	5.041
1-1980	4.715	5.015	5.600
1-1981	4.093	4.715	5.015
1-1982	3.166	4.093	4.715

Further functions called `Between`, `between` and `Within` are also provided to compute the between and the within transformation. The `between` returns unique values, whereas `Between` duplicates the values and returns a vector which length is the number of observations.

```
R> head(diff(E$emp), 10)
```

1-1977	1-1978	1-1979	1-1980	1-1981	1-1982	1-1983
NA	0.5590000	-0.5850000	-0.2999997	-0.6220003	-0.9270000	-0.2299998
2-1977	2-1978	2-1979				
NA	-0.6760020	0.2750010				

```
R> head(lag(E$emp, 2), 10)
```

1-1977	1-1978	1-1979	1-1980	1-1981	1-1982	1-1983	2-1977	2-1978	2-1979
NA	NA	5.041	5.600	5.015	4.715	4.093	NA	NA	71.319

```
R> head(Within(E$emp))
```

1-1977	1-1978	1-1979	1-1980	1-1981	1-1982
0.6744285	1.2334285	0.6484285	0.3484288	-0.2735715	-1.2005715

```
R> head(between(E$emp), 4)
```

```

      1          2          3          4
4.366571 71.362428 19.040143 26.035000

R> head(Between(E$emp), 10)

      1          1          1          1          1          1          1          1          1          2
4.366571 4.366571 4.366571 4.366571 4.366571 4.366571 4.366571 4.366571 71.362428
      2          2
71.362428 71.362428

```

4.3. Formulas

There are circumstances where standard `formula` are not very useful to describe a model, notably while using instrumental variable like estimators: to deal with these situations, we use the `Formula` package.

The `Formula` package provides a class which enables to construct multi-part formula, each part being separated by a pipe sign. `plm` provides a `pFormula` object which is a `Formula` with specific methods.

The two formulas below are identical:

```

R> emp~wage+capital/lag(wage,1)+capital

emp ~ wage + capital | lag(wage, 1) + capital

R> emp~wage+capital/.-wage+lag(wage,1)

emp ~ wage + capital | . - wage + lag(wage, 1)

```

In the second case, the `.` means the previous parts which describes the covariates and this part is “updated”. This is particularly interesting when there are a few external instruments.

5. Model estimation

5.1. Estimation of the basic models with `plm`

Several models can be estimated with `plm` by filling the `model` argument:

- the fixed effects model (`within`),
- the pooling model (`pooling`),
- the first-difference model (`fd`),
- the between model (`between`),
- the error components model (`random`).

The basic use of `plm` is to indicate the model formula, the data and the model to be estimated. For example, the fixed effects model and the random effects model are estimated using:

```
R> grun.fe <- plm(inv~value+capital, data = Grunfeld, model = "within")
R> grun.re <- plm(inv~value+capital, data = Grunfeld, model = "random")

R> summary(grun.re)

Oneway (individual) effect Random Effect Model
(Swamy-Arora's transformation)

Call:
plm(formula = inv ~ value + capital, data = Grunfeld, model = "random")

Balanced Panel: n=10, T=20, N=200

Effects:
      var std.dev share
idiosyncratic 2784.46   52.77 0.282
individual     7089.80   84.20 0.718
theta:    0.8612

Residuals :
    Min. 1st Qu. Median 3rd Qu. Max.
-178.00 -19.70    4.69   19.50 253.00

Coefficients :
            Estimate Std. Error t-value Pr(>|t|)
(Intercept) -57.834415  28.898935 -2.0013  0.04674 *
value         0.109781   0.010493 10.4627 < 2e-16 ***
capital       0.308113   0.017180 17.9339 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Total Sum of Squares:  2381400
Residual Sum of Squares: 548900
R-Squared: 0.7695
Adj. R-Squared: 0.76716
F-statistic: 328.837 on 2 and 197 DF, p-value: < 2.22e-16
```

For a random model, the `summary` method gives information about the variance of the components of the errors. Fixed effects may be extracted easily using `fixef`. An argument `type` indicates how fixed effects should be computed : in level `type = 'level'` (the default), in deviation from the overall mean `type = 'dmean'` or in deviation from the first individual `type = 'dfirst'`.

```
R> fixef(grun.fe, type = 'dmean')
```

1	2	3	4	5	6
-11.552778	160.649753	-176.827902	30.934645	-55.872873	35.582644
7	8	9	10		
-7.809534	1.198282	-28.478333	52.176096		

The `fixef` function returns an object of class `fixef`. A summary method is provided, which prints the effects (in deviation from the overall intercept), their standard errors and the test of equality to the overall intercept.

```
R> summary(fixef(grun.fe, type = 'dmean'))
```

	Estimate	Std. Error	t-value	Pr(> t)	
1	-11.5528	49.7080	-0.2324	0.8164700	
2	160.6498	24.9383	6.4419	9.627e-10 ***	
3	-176.8279	24.4316	-7.2377	1.130e-11 ***	
4	30.9346	14.0778	2.1974	0.0292129 *	
5	-55.8729	14.1654	-3.9443	0.0001129 ***	
6	35.5826	12.6687	2.8087	0.0054998 **	
7	-7.8095	12.8430	-0.6081	0.5438694	
8	1.1983	13.9931	0.0856	0.9318489	
9	-28.4783	12.8919	-2.2090	0.0283821 *	
10	52.1761	11.8269	4.4116	1.725e-05 ***	
<hr/>					
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1					

In case of a two-ways effect model, an additional argument `effect` is required to extract fixed effects:

```
R> grun.twfe <- plm(inv~value+capital,data=Grunfeld,model="within",effect="twoways")
R> fixef(grun.twfe, effect="time")
```

1935	1936	1937	1938	1939	1940	1941
-32.83632	-52.03372	-73.52633	-72.06272	-102.30660	-77.07140	-51.64078
1942	1943	1944	1945	1946	1947	1948
-53.97611	-75.81394	-75.93509	-88.51936	-64.00560	-72.22856	-76.55283
1949	1950	1951	1952	1953	1954	
-106.33142	-108.73243	-95.31723	-97.46866	-100.55428	-126.36254	

5.2. More advanced use of plm

Random effects estimators

As observed above, the random effect model is obtained as a linear estimation on quasi-demeaned data. The parameter of this transformation is obtained using preliminary estimations.

Four estimators of this parameter are available, depending on the value of the argument `random.method`:

- **swar**: from Swamy and Arora (1972), the default value,
- **walhus**: from Wallace and Hussain (1969),
- **amemiya**: from Amemiya (1971),
- **nerlove**: from Nerlove (1971).

For example, to use the **amemiya** estimator:

```
R> grun.amem <- plm(inv~value+capital, data=Grunfeld,
+                      model="random", random.method="amemiya")
```

The estimation of the variance of the error components are performed using the **ercomp** function, which has a **method** and an **effect** argument, and can be used by itself:

```
R> ercomp(inv~value+capital, data=Grunfeld, method = "amemiya", effect = "twoways")

      var std.dev share
idiosyncratic 2644.13   51.42 0.236
individual     8294.72   91.08 0.740
time          270.53    16.45 0.024
theta : 0.8747 (id) 0.2969 (time) 0.296 (total)
```

Introducing time or two-ways effects

The default behavior of **plm** is to introduce individual effects. Using the **effect** argument, one may also introduce:

- time effects (**effect="time"**),
- individual and time effects (**effect="twoways"**).

For example, to estimate a two-ways effect model for the Grunfeld data:

```
R> grun.tways <- plm(inv~value+capital, data = Grunfeld, effect = "twoways",
+                      model = "random", random.method = "amemiya")
R> summary(grun.tways)
```

```
Twoways effects Random Effect Model
(Amemiya's transformation)
```

```
Call:
plm(formula = inv ~ value + capital, data = Grunfeld, effect = "twoways",
model = "random", random.method = "amemiya")
```

Balanced Panel: n=10, T=20, N=200

Effects:

```
var std.dev share
idiosyncratic 2644.13 51.42 0.236
individual     8294.72 91.08 0.740
time          270.53 16.45 0.024
theta : 0.8747 (id) 0.2969 (time) 0.296 (total)
```

Residuals :

Min.	1st Qu.	Median	3rd Qu.	Max.
-176.00	-18.00	3.02	18.00	233.00

Coefficients :

	Estimate	Std. Error	t-value	Pr(> t)							
(Intercept)	-64.351811	31.183651	-2.0636	0.04036 *							
value	0.111593	0.011028	10.1192	< 2e-16 ***							
capital	0.324625	0.018850	17.2214	< 2e-16 ***							

Signif. codes:	0	'***'	0.001	'**'	0.01	'*'	0.05	'..'	0.1	' '	1

Total Sum of Squares: 2038000
 Residual Sum of Squares: 514120
 R-Squared: 0.74774
 Adj. R-Squared: 0.74518
 F-statistic: 291.965 on 2 and 197 DF, p-value: < 2.22e-16

In the “effects” section of the result, the variance of the three elements of the error term and the three parameters used in the transformation are now printed. The two-ways effect model is for the moment only available for balanced panels.

Unbalanced panels

Most of the features of `plm` are implemented for panel models with some limitations:

- the random two-ways effect model is not implemented,
- the only estimator of the variance of the error components is the one proposed by [Swamy and Arora \(1972\)](#)

The following example is using data used by ([Harrison and Rubinfeld 1978](#)) to estimate an hedonic housing prices function. It is reproduced in ([Baltagi 2001](#)), p.174.

```
R> data("Hedonic", package = "plm")
R> Hed <- plm(mv~crim+zn+indus+chas+nox+rm+age+dis+rad+tax+ptratio+blacks+lstat,
+               data = Hedonic, model = "random", index = "townid")
R> summary(Hed)
```

Oneway (individual) effect Random Effect Model
 (Swamy-Arora's transformation)

Call:

```
plm(formula = mv ~ crim + zn + indus + chas + nox + rm + age +
  dis + rad + tax + ptratio + blacks + lstat, data = Hedonic,
  model = "random", index = "townid")
```

Unbalanced Panel: n=92, T=1-30, N=506

Effects:

	var	std.dev	share			
idiosyncratic	0.01718	0.13106	0.572			
individual	0.01287	0.11342	0.428			
theta :						
	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
	0.2439	0.5409	0.6218	0.6078	0.7093	0.7936

Residuals :

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
	-0.62700	-0.06680	-0.00127	-0.00219	0.06860	0.55400

Coefficients :

	Estimate	Std. Error	t-value	Pr(> t)
(Intercept)	9.6871e+00	1.9603e-01	49.4159	< 2.2e-16 ***
crim	-7.4447e-03	1.0502e-03	-7.0891	4.713e-12 ***
zn	8.4524e-05	6.4423e-04	0.1312	0.8956688
indus	1.4759e-03	3.9871e-03	0.3702	0.7114187
chasyes	-3.3181e-03	2.9256e-02	-0.1134	0.9097487
nox	-5.8387e-03	1.2451e-03	-4.6895	3.552e-06 ***
rm	9.0316e-03	1.1903e-03	7.5877	1.643e-13 ***
age	-8.4567e-04	4.6851e-04	-1.8050	0.0716850 .
dis	-1.4620e-01	4.3834e-02	-3.3352	0.0009166 ***
rad	9.5854e-02	2.6343e-02	3.6387	0.0003030 ***
tax	-3.7787e-04	1.7506e-04	-2.1585	0.0313730 *
ptratio	-2.9445e-02	8.9631e-03	-3.2851	0.0010921 **
blacks	5.6059e-01	1.0214e-01	5.4886	6.497e-08 ***
lstat	-2.9213e-01	2.3940e-02	-12.2025	< 2.2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Total Sum of Squares: 1003.2

Residual Sum of Squares: 9.056

R-Squared: 0.99099

Adj. R-Squared: 0.99076

F-statistic: 4154.45 on 13 and 492 DF, p-value: < 2.22e-16

Instrumental variable estimators

All of the models presented above may be estimated using instrumental variables. The instruments are specified at the end of the formula after a | sign.

The instrumental variables estimator used is indicated with the `inst.method` argument:

- `bvk`, from [Balestra and Varadharajan-Krishnakumar \(1987\)](#), the default value,
- `baltagi`, from [Baltagi \(1981\)](#).

as illustrated on the following example from [Baltagi \(2001\)](#), p.120.

```
R> data("Crime", package = "plm")
R> cr <- plm(log(crmrte) ~ log(prbarr) + log(polpc) + log(prbconv) +
+           log(prbpris) + log(avgsen) + log(density) + log(wcon) +
+           log(wtuc) + log(wtrd) + log(wfir) + log(wser) + log(wmfg) +
+           log(wfed) + log(wsta) + log(wloc) + log(pctymle) + log(pctmin) +
+           region + smsa + factor(year) | . - log(prbarr) -log(polpc) +
+           log(taxpc) + log(mix), data = Crime,
+           model = "random")
R> summary(cr)
```

```
Oneway (individual) effect Random Effect Model
(Swamy-Arora's transformation)
Instrumental variable estimation
(Balestra-Varadharajan-Krishnakumar's transformation)
```

Call:

```
plm(formula = log(crmrte) ~ log(prbarr) + log(polpc) + log(prbconv) +
    log(prbpris) + log(avgsen) + log(density) + log(wcon) + log(wtuc) +
    log(wtrd) + log(wfir) + log(wser) + log(wmfg) + log(wfed) +
    log(wsta) + log(wloc) + log(pctymle) + log(pctmin) + region +
    smsa + factor(year) | . - log(prbarr) - log(polpc) + log(taxpc) +
    log(mix), data = Crime, model = "random")
```

Balanced Panel: n=90, T=7, N=630

Effects:

```
var std.dev share
idiosyncratic 0.02244 0.14981 0.327
individual     0.04629 0.21515 0.673
theta:   0.7455
```

Residuals :

Min.	1st Qu.	Median	3rd Qu.	Max.
-0.74900	-0.07100	0.00401	0.07840	0.47600

Coefficients :

	Estimate	Std. Error	t-value	Pr(> t)		
(Intercept)	-0.4555112	1.7025145	-0.2676	0.789136		
log(prbarr)	-0.4140239	0.2208454	-1.8747	0.061314 .		
log(polpc)	0.5048062	0.2275663	2.2183	0.026907 *		
log(prbconv)	-0.3432062	0.1323357	-2.5935	0.009732 **		
log(prbpris)	-0.1900033	0.0732730	-2.5931	0.009743 **		
log(avgsen)	-0.0064595	0.0289406	-0.2232	0.823457		
log(density)	0.4343352	0.0710812	6.1104	1.784e-09 ***		
log(wcon)	-0.0042528	0.0414258	-0.1027	0.918267		
log(wtuc)	0.0444633	0.0215454	2.0637	0.039473 *		
log(wtrd)	-0.0085496	0.0419878	-0.2036	0.838718		
log(wfir)	-0.0040260	0.0294595	-0.1367	0.891344		
log(wser)	0.0105490	0.0215817	0.4888	0.625165		
log(wmfg)	-0.2017303	0.0838911	-2.4047	0.016488 *		
log(wfed)	-0.2132122	0.2150104	-0.9916	0.321773		
log(wsta)	-0.0601899	0.1202840	-0.5004	0.616977		
log(wloc)	0.1834259	0.1396384	1.3136	0.189488		
log(pctymle)	-0.1458336	0.2266046	-0.6436	0.520106		
log(pctmin)	0.1948622	0.0459017	4.2452	2.528e-05 ***		
regionwest	-0.2281843	0.1009432	-2.2605	0.024145 *		
regioncentral	-0.1987812	0.0606986	-3.2749	0.001118 **		
smsayes	-0.2594699	0.1498521	-1.7315	0.083873 .		
factor(year)82	0.0132028	0.0299883	0.4403	0.659904		
factor(year)83	-0.0847856	0.0320025	-2.6493	0.008276 **		
factor(year)84	-0.1062250	0.0387884	-2.7386	0.006353 **		
factor(year)85	-0.0977658	0.0511663	-1.9107	0.056511 .		
factor(year)86	-0.0719561	0.0605729	-1.1879	0.235331		
factor(year)87	-0.0396637	0.0758245	-0.5231	0.601098		

Signif. codes:	0 '***'	0.001 '**'	0.01 '*'	0.05 '.'	0.1 ' '	1
Total Sum of Squares:	30.193					
Residual Sum of Squares:	12.422					
R-Squared:	0.59247					
Adj. R-Squared:	0.5749					
F-statistic:	33.1779	on 26 and 603 DF,	p-value:	< 2.22e-16		

The Hausman-Taylor model (see [Hausman and Taylor 1981](#)) may be estimated with the **pht** function. The following example is from [Baltagi \(2001\)](#) p.130.

```
R> ht <- pht(lwage~wks+south+smsa+married+exp+I(exp^2)+  
+           bluecol+ind+union+sex+black+ed |  
+           sex+black+bluecol+south+smsa+ind,  
+           data=Wages,index=595)  
R> summary(ht)
```

```
Oneway (individual) effect Hausman-Taylor Model  
Call:
```

```
pht(formula = lwage ~ wks + south + smsa + married + exp + I(exp^2) +
  bluecol + ind + union + sex + black + ed | sex + black +
  bluecol + south + smsa + ind, data = Wages, index = 595)
```

```
T.V. exo : bluecol, south, smsa, ind
T.V. endo : wks, married, exp, I(exp^2), union
T.I. exo : sex, black
T.I. endo : ed
```

Balanced Panel: n=595, T=7, N=4165

Effects:

var	std.dev	share	
idiosyncratic	0.02304	0.15180	0.025
individual	0.88699	0.94180	0.975
theta:	0.9392		

Residuals :

Min.	1st Qu.	Median	3rd Qu.	Max.
-1.92000	-0.07070	0.00657	0.07970	2.03000

Coefficients :

	Estimate	Std. Error	t-value	Pr(> t)							
(Intercept)	2.9127e+00	2.8365e-01	10.2687	< 2.2e-16 ***							
wks	8.3740e-04	5.9973e-04	1.3963	0.16263							
southyes	7.4398e-03	3.1955e-02	0.2328	0.81590							
smsayes	-4.1833e-02	1.8958e-02	-2.2066	0.02734 *							
marriedyes	-2.9851e-02	1.8980e-02	-1.5728	0.11578							
exp	1.1313e-01	2.4710e-03	45.7851	< 2.2e-16 ***							
I(exp^2)	-4.1886e-04	5.4598e-05	-7.6718	1.696e-14 ***							
bluecolyes	-2.0705e-02	1.3781e-02	-1.5024	0.13299							
ind	1.3604e-02	1.5237e-02	0.8928	0.37196							
unionyes	3.2771e-02	1.4908e-02	2.1982	0.02794 *							
sexfemale	-1.3092e-01	1.2666e-01	-1.0337	0.30129							
blackyes	-2.8575e-01	1.5570e-01	-1.8352	0.06647 .							
ed	1.3794e-01	2.1248e-02	6.4919	8.474e-11 ***							

Signif. codes:	0	'***'	0.001	'**'	0.01	'*'	0.05	'..'	0.1	' '	1

Total Sum of Squares: 886.9

Residual Sum of Squares: 95.947

F-statistic: 2852.33 on 12 and 4152 DF, p-value: < 2.22e-16

5.3. Variable coefficients model

The pvcm function enables the estimation of variable coefficients models. Time or individual

effects are introduced if `effect` is fixed to "time" or "individual" (the default value).

Coefficients are assumed to be fixed if `model="within"` or random if `model="random"`. In the first case, a different model is estimated for each individual (or time period). In the second case, the Swamy model (see [Swamy 1970](#)) model is estimated. It is a generalized least squares model which uses the results of the previous model. Denoting $\hat{\beta}_i$ the vectors of coefficients obtained for each individual, we get:

$$\hat{\beta} = \left(\sum_{i=1}^n \left(\hat{\Delta} + \hat{\sigma}_i^2 (X_i^\top X_i)^{-1} \right)^{-1} \right) \left(\hat{\Delta} + \hat{\sigma}_i^2 (X_i^\top X_i)^{-1} \right)^{-1} \hat{\beta}_i \quad (8)$$

where $\hat{\sigma}_i^2$ is the unbiased estimator of the variance of the errors for individual i obtained from the preliminary estimation and:

$$\hat{\Delta} = \frac{1}{n-1} \sum_{i=1}^n \left(\hat{\beta}_i - \frac{1}{n} \sum_{i=1}^n \hat{\beta}_i \right) \left(\hat{\beta}_i - \frac{1}{n} \sum_{i=1}^n \hat{\beta}_i \right)^\top - \frac{1}{n} \sum_{i=1}^n \hat{\sigma}_i^2 (X_i^\top X_i)^{-1} \quad (9)$$

If this matrix is not positive-definite, the second term is dropped.

With the `Grunfeld` data, we get:

```
R> grun.varw <- pvcm(inv~value+capital, data=Grunfeld, model="within")
R> grun.varr <- pvcm(inv~value+capital, data=Grunfeld, model="random")

[1] 3.339740e-02 1.633363e-03 -1.120478e+03
attention

R> summary(grun.varr)

Oneway (individual) effect Random coefficients model

Call:
pvcm(formula = inv ~ value + capital, data = Grunfeld, model = "random")

Balanced Panel: n=10, T=20, N=200

Residuals:
total sum of squares : 2177914
      id      time
0.67677732 0.02974195

Estimated mean of the coefficients:
Estimate Std. Error z-value Pr(>|z|)
(Intercept) -9.629285 17.035040 -0.5653 0.5718946
value        0.084587  0.019956  4.2387 2.248e-05 ***
capital      0.199418  0.052653  3.7874 0.0001522 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Estimated variance of the coefficients:

	(Intercept)	value	capital
(Intercept)	2344.24402	-0.6852340	-4.0276612
value	-0.68523	0.0031182	-0.0011847
capital	-4.02766	-0.0011847	0.0244824

Total Sum of Squares: 474010000

Residual Sum of Squares: 2194300

Multiple R-Squared: 0.99537

5.4. Generalized method of moments estimator

The generalized method of moments is mainly used in panel data econometrics to estimate dynamic models (Arellano and Bond 1991; Holtz-Eakin, Newey, and Rosen 1988).

$$y_{it} = \rho y_{it-1} + \beta^\top x_{it} + \mu_i + \epsilon_{it} \quad (10)$$

The model is first differenced to get rid of the individual effect:

$$\Delta y_{it} = \rho \Delta y_{it-1} + \beta^\top \Delta x_{it} + \Delta \epsilon_{it} \quad (11)$$

Least squares are inconsistent because $\Delta \epsilon_{it}$ is correlated with Δy_{it-1} . y_{it-2} is a valid, but weak instrument (see Anderson and Hsiao 1981). The GMM estimator uses the fact that the number of valid instruments is growing with t :

- $t = 3$: y_1 ,
- $t = 4$: y_1, y_2 ,
- $t = 5$: y_1, y_2, y_3

For individual i , the matrix of instruments is then:

$$W_i = \begin{pmatrix} y_1 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & x_{i3} \\ 0 & y_1 & y_2 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & x_{i4} \\ 0 & 0 & 0 & y_1 & y_2 & y_3 & \dots & 0 & 0 & 0 & 0 & x_{i5} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & \dots & \dots & y_1 & y_2 & \dots & y_{t-2} & x_{iT-2} \end{pmatrix} \quad (12)$$

The moment conditions are: $\sum_{i=1}^n W_i^\top e_i(\beta)$ where $e_i(\beta)$ is the vector of residuals for individual i . The GMM estimator minimize:

$$\left(\sum_{i=1}^n e_i(\beta)^\top W_i \right) A \left(\sum_{i=1}^n W_i^\top e_i(\beta) \right) \quad (13)$$

where A is the weighting matrix of the moments.

One-step estimators are computed using a known weighting matrix. For the model in first differences, one uses:

$$A^{(1)} = \left(\sum_{i=1}^n W_i^\top H^{(1)} W_i \right)^{-1} \quad (14)$$

with:

$$H^{(1)} = d^\top d = \begin{pmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \dots & 0 \\ 0 & -1 & 2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & -1 & 2 \end{pmatrix} \quad (15)$$

Two-steps estimators are obtained using $H_i^{(2)} = \sum_{i=1}^n e_i^{(1)} e_i^{(1)\top}$ where $e_i^{(1)}$ are the residuals of the one step estimate.

[Blundell and Bond \(1998\)](#) show that with weak hypothesis on the data generating process, supplementary moment conditions exist for the equation in level:

$$y_{it} = \gamma y_{it-1} + \mu_i + \eta_{it}$$

More precisely, they show that $\Delta y_{it-2} = y_{it-2} - y_{it-3}$ is a valid instrument. The estimator is obtained using the residual vector in difference and in level:

$$e_i^+ = (\Delta e_i, e_i)$$

and the matrix of augmented moments:

$$Z_i^+ = \begin{pmatrix} Z_i & 0 & 0 & \dots & 0 \\ 0 & \Delta y_{i2} & 0 & \dots & 0 \\ 0 & 0 & \Delta y_{i3} & \dots & 0 \\ 0 & 0 & 0 & \dots & \Delta y_{iT-1} \end{pmatrix}$$

The moment conditions are then

$$\begin{aligned} \left(\sum_{i=1}^n Z_i^{+\top} \begin{pmatrix} \bar{e}_i(\beta) \\ e_i(\beta) \end{pmatrix} \right)^\top = & \left(\sum_{i=1}^n y_{i1} \bar{e}_{i3}, \sum_{i=1}^n y_{i1} \bar{e}_{i4}, \sum_{i=1}^n y_{i2} \bar{e}_{i4}, \dots, \right. \\ & \sum_{i=1}^n y_{i1} \bar{e}_{iT}, \sum_{i=1}^n y_{i2} \bar{e}_{iT}, \dots, \sum_{i=1}^n y_{iT-2} \bar{e}_{iT}, \sum_{i=1}^n \sum_{t=3}^T x_{it} \bar{e}_{it} \\ & \left. \sum_{i=1}^n e_{i3} \Delta y_{i2}, \sum_{i=1}^n e_{i4} \Delta y_{i3}, \dots, \sum_{i=1}^n e_{iT} \Delta y_{iT-1} \right)^\top \end{aligned}$$

The GMM estimator is provided by the **pgmm** function. It's main argument is a **dynformula** which describes the variables of the model and the lag structure.

In a GMM estimation, there are “normal” instruments and “GMM” instruments. GMM instruments are indicated in the second part of the formula. By default, all the variables of the model that are not used as GMM instruments are used as normal instruments, with the same lag structure; “normal” instruments may also be indicated in the third part of the formula.

The effect argument is either `NULL`, `"individual"` (the default), or `"twoways"`. In the first case, the model is estimated in levels. In the second case, the model is estimated in first differences to get rid of the individuals effects. In the last case, the model is estimated in first differences and time dummies are included.

The `model` argument specifies whether a one-step or a two-steps model is required (`"onestep"` or `"twosteps"`).

The following example is from [Arellano and Bond \(1991\)](#). Employment is explained by past values of employment (two lags), current and first lag of wages and output and current value of capital.

```
R> emp.gmm <- pgmm(log(emp)~lag(log(emp), 1:2)+lag(log(wage), 0:1)+log(capital)+  
+           lag(log(output), 0:1)|lag(log(emp), 2:99),  
+           data = EmplUK, effect = "twoways", model = "twosteps")  
R> summary(emp.gmm)  
  
Twoways effects Two steps model  
  
Call:  
pgmm(formula = log(emp) ~ lag(log(emp), 1:2) + lag(log(wage),  
 0:1) + log(capital) + lag(log(output), 0:1) | lag(log(emp),  
 2:99), data = EmplUK, effect = "twoways", model = "twosteps")  
  
Unbalanced Panel: n=140, T=7-9, N=1031  
  
Number of Observations Used: 611  
  
Residuals  
    Min.   1st Qu.   Median   Mean   3rd Qu.   Max.     
-0.6191000 -0.0255700  0.0000000 -0.0001339  0.0332000  0.6410000  
  
Coefficients  
              Estimate Std. Error z-value Pr(>|z|)  
lag(log(emp), 1:2)1  0.474151  0.185398  2.5575 0.0105437 *  
lag(log(emp), 1:2)2 -0.052967  0.051749 -1.0235 0.3060506  
lag(log(wage), 0:1)0 -0.513205  0.145565 -3.5256 0.0004225 ***  
lag(log(wage), 0:1)1  0.224640  0.141950  1.5825 0.1135279  
log(capital)        0.292723  0.062627  4.6741 2.953e-06 ***  
lag(log(output), 0:1)0 0.609775  0.156263  3.9022 9.530e-05 ***  
lag(log(output), 0:1)1 -0.446373  0.217302 -2.0542 0.0399605 *  
---  
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Sargan Test: chisq(25) = 30.11247 (p.value=0.22011)
Autocorrelation test (1): normal = -1.53845 (p.value=0.12394)
Autocorrelation test (2): normal = -0.2796829 (p.value=0.77972)
Wald test for coefficients: chisq(7) = 142.0353 (p.value=< 2.22e-16)
Wald test for time dummies: chisq(6) = 16.97046 (p.value=0.0093924)
```

The following example is from [Blundell and Bond \(1998\)](#). The “sys” estimator is obtained using `transformation = "ld"` for level and difference. The `robust` argument of the `summary` method enables to use the robust covariance matrix proposed by [Windmeijer \(2005\)](#).

```
R> z2 <- pgmm(log(emp) ~ lag(log(emp), 1) + lag(log(wage), 0:1) +
+                 lag(log(capital), 0:1) | lag(log(emp), 2:99) +
+                 lag(log(wage), 2:99) + lag(log(capital), 2:99),
+                 data = EmplUK, effect = "twoways", model = "onestep",
+                 transformation = "ld")
R> summary(z2, robust = TRUE)
```

Twoways effects One step model

Call:

```
pgmm(formula = log(emp) ~ lag(log(emp), 1) + lag(log(wage), 0:1) +
      lag(log(capital), 0:1) | lag(log(emp), 2:99) + lag(log(wage),
      2:99) + lag(log(capital), 2:99), data = EmplUK, effect = "twoways",
      model = "onestep", transformation = "ld")
```

Unbalanced Panel: n=140, T=7-9, N=1031

Number of Observations Used: 1642

Residuals

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
-0.7530000	-0.0369000	0.0000000	0.0002882	0.0466100	0.6002000

Coefficients

	Estimate	Std. Error	z-value	Pr(> z)
lag(log(emp), 1)	0.935605	0.026295	35.5810 < 2.2e-16	***
lag(log(wage), 0:1)0	-0.630976	0.118054	-5.3448 9.050e-08	***
lag(log(wage), 0:1)1	0.482620	0.136887	3.5257 0.0004224	***
lag(log(capital), 0:1)0	0.483930	0.053867	8.9838 < 2.2e-16	***
lag(log(capital), 0:1)1	-0.424393	0.058479	-7.2572 3.952e-13	***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```
Sargan Test: chisq(100) = 118.763 (p.value=0.097096)
Autocorrelation test (1): normal = -4.808434 (p.value=1.5212e-06)
Autocorrelation test (2): normal = -0.2800133 (p.value=0.77947)
```

```
Wald test for coefficients: chisq(5) = 11174.82 (p.value=< 2.22e-16)
Wald test for time dummies: chisq(7) = 14.71138 (p.value=0.039882)
```

5.5. General FGLS models

General FGLS estimators are based on a two-step estimation process: first an OLS model is estimated, then its residuals \hat{u}_{it} are used to estimate an error covariance matrix more general than the random effects one for use in a feasible-GLS analysis. Formally, the estimated error covariance matrix is $\hat{V} = I_n \otimes \hat{\Omega}$, with

$$\hat{\Omega} = \sum_{i=1}^n \frac{\hat{u}_{it}\hat{u}_{it}^\top}{n}$$

(see [Wooldridge 2002](#), 10.4.3 and 10.5.5).

This framework allows the error covariance structure inside every group (if `effect="individual"`) of observations to be fully unrestricted and is therefore robust against any type of intragroup heteroskedasticity and serial correlation. This structure, by converse, is assumed identical across groups and thus general FGLS is inefficient under groupwise heteroskedasticity. Cross-sectional correlation is excluded a priori.

Moreover, the number of variance parameters to be estimated with $N = n \times T$ data points is $T(T + 1)/2$, which makes these estimators particularly suited for situations where $n \gg T$, as e.g. in labour or household income surveys, while problematic for “long” panels, where \hat{V} tends to become singular and standard errors therefore become biased downwards.

In a pooled time series context (`effect="time"`), symmetrically, this estimator is able to account for arbitrary cross-sectional correlation, provided that the latter is time-invariant (see [Greene 2003](#), 13.9.1–2, pp.321–2). In this case serial correlation has to be assumed away and the estimator is consistent with respect to the time dimension, keeping n fixed.

The function `pggls` estimates general FGLS models, with either fixed or “random” effects⁶.

The “random effect” general FGLS is estimated by:

```
R> zz <- pggls(log(emp)~log(wage)+log(capital), data=EmplUK, model="pooling")
R> summary(zz)
```

NA

```
Call:
pggls(formula = log(emp) ~ log(wage) + log(capital), data = EmplUK,
      model = "pooling")
```

Unbalanced Panel: n=140, T=7-9, N=1031

Residuals

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
------	---------	--------	------	---------	------

⁶The “random effect” is better termed “general FGLS” model, as in fact it does not have a proper random effects structure, but we keep this terminology for general language consistency.

```

-1.80700 -0.36550  0.06181  0.03230  0.44280  1.58700

Coefficients:
            Estimate Std. Error z-value Pr(>|z|)
(Intercept) 2.023480   0.158468 12.7690 < 2.2e-16 ***
log(wage)   -0.232329   0.048001 -4.8401 1.298e-06 ***
log(capital) 0.610484   0.017434 35.0174 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Total Sum of Squares: 1853.6
Residual Sum of Squares: 402.55
Multiple R-squared: 0.78283

```

The fixed effects **pggls** (see Wooldridge 2002, p.276) is based on the estimation of a within model in the first step; the rest follows as above. It is estimated by:

```
R> zz <- pggls(log(emp)~log(wage)+log(capital), data=EmplUK, model="within")
```

The **pggls** function is similar to **plm** in many respects. An exception is that the estimate of the group covariance matrix of errors (**zz\$sigma**, a matrix, not shown) is reported in the model objects instead of the usual estimated variances of the two error components.

6. Tests

As sketched in Section 2, specification testing in panel models involves essentially testing for poolability, for individual or time unobserved effects and for correlation between these latter and the regressors (Hausman-type tests). As for the other usual diagnostic checks, we provide a suite of serial correlation tests, while not touching on the issue of heteroskedasticity testing. Instead, we provide heteroskedasticity-robust covariance estimators, to be described in Subsection 6.7.

6.1. Tests of poolability

pooltest tests the hypothesis that the same coefficients apply to each individual. It is a standard F test, based on the comparison of a model obtained for the full sample and a model based on the estimation of an equation for each individual. The first argument of **pooltest** is a **plm** object. The second argument is a **pvcm** object obtained with **model=within**. If the first argument is a pooling model, the test applies to all the coefficients (including the intercepts), if it is a **within** model, different intercepts are assumed.

To test the hypothesis that all the coefficients in the **Grunfeld** example, excluding the intercepts, are equal, we use :

```
R> znp <- pvcm(inv~value+capital, data=Grunfeld, model="within")
R> zplm <- plm(inv~value+capital, data=Grunfeld, model="within")
R> pooltest(zplm,znp)
```

```
F statistic
```

```
data: inv ~ value + capital
F = 5.7805, df1 = 18, df2 = 170, p-value = 1.219e-10
alternative hypothesis: unstability
```

The same test can be computed using a formula as first argument of the `pooltest` function:

```
R> pooltest(inv~value+capital, data=Grunfeld, model="within")
```

6.2. Tests for individual and time effects

`plmtest` implements Lagrange multiplier tests of individual or/and time effects based on the results of the pooling model. Its main argument is a `plm` object (the result of a pooling model) or a formula.

Two additional arguments can be added to indicate the kind of test to be computed. The argument `type` is one of:

- `bp`: Breusch and Pagan (1980),
- `honda`: Honda (1985), the default value,
- `kw`: King and Wu (1997),
- `ghm`: Gourieroux, Holly, and Monfort (1982).

The effects tested are indicated with the `effect` argument (one of `individual`, `time` or `twoways`).

To test the presence of individual and time effects in the `Grunfeld` example, using the Gourieroux *et al.* (1982) test, we use:

```
R> g <- plm(inv ~ value + capital, data=Grunfeld, model="pooling")
R> plmtest(g, effect="twoways", type="ghm")
```

```
Lagrange Multiplier Test - two-ways effects (Gourieroux, Holly and
Monfort) for balanced panels
```

```
data: inv ~ value + capital
chibarsq = 798.16, df0 = 0.00, df1 = 1.00, df2 = 2.00, w0 = 0.25, w1 =
0.50, w2 = 0.25, p-value < 2.2e-16
alternative hypothesis: significant effects
```

or

```
R> plmtest(inv~value+capital, data=Grunfeld, effect="twoways", type="ghm")
```

pFtest computes F tests of effects based on the comparison of the **within** and the **pooling** models. Its main arguments are either two **plm** objects (the results of a **pooling** and a **within** model) or a formula.

```
R> gw <- plm(inv ~ value + capital, data=Grunfeld, effect="twoways", model="within")
R> gp <- plm(inv ~ value + capital, data=Grunfeld, model="pooling")
R> pFtest(gw, gp)

F test for twoways effects

data: inv ~ value + capital
F = 17.403, df1 = 28, df2 = 169, p-value < 2.2e-16
alternative hypothesis: significant effects

R> pFtest(inv~value+capital, data=Grunfeld, effect="twoways")
```

6.3. Hausman test

phtest computes the Hausman test which is based on the comparison of two sets of estimates (see [Hausman 1978](#)). Its main arguments are two **panelmodel** objects or a formula. A classical application of the Hausman test for panel data is to compare the fixed and the random effects models:

```
R> gw <- plm(inv~value+capital, data=Grunfeld, model="within")
R> gr <- plm(inv~value+capital, data=Grunfeld, model="random")
R> phtest(gw, gr)

Hausman Test

data: inv ~ value + capital
chisq = 2.3304, df = 2, p-value = 0.3119
alternative hypothesis: one model is inconsistent
```

6.4. Tests of serial correlation

A model with individual effects has composite errors that are serially correlated by definition. The presence of the time-invariant error component⁷ gives rise to serial correlation which does not die out over time, thus standard tests applied on pooled data always end up rejecting the null of spherical residuals⁸. There may also be serial correlation of the “usual” kind in the idiosyncratic error terms, e.g. as an AR(1) process. By “testing for serial correlation” we mean testing for this latter kind of dependence.

⁷Here we treat fixed and random effects alike, as components of the error term, according with the modern approach in econometrics (see [Wooldridge 2002](#)).

⁸Neglecting time effects may also lead to serial correlation in residuals (as observed in [Wooldridge 2002](#), 10.4.1).

For these reasons, the subjects of testing for individual error components and for serially correlated idiosyncratic errors are closely related. In particular, simple (*marginal*) tests for one direction of departure from the hypothesis of spherical errors usually have power against the other one: in case it is present, they are substantially biased towards rejection. *Joint* tests are correctly sized and have power against both directions, but usually do not give any information about which one actually caused rejection. *Conditional* tests for serial correlation that take into account the error components are correctly sized under presence of both departures from sphericity and have power only against the alternative of interest. While most powerful if correctly specified, the latter, based on the likelihood framework, are crucially dependent on normality and homoskedasticity of the errors.

In **plm** we provide a number of joint, marginal and conditional ML-based tests, plus some semiparametric alternatives which are robust vs. heteroskedasticity and free from distributional assumptions.

Unobserved effects test

The unobserved effects test à la Wooldridge (see [Wooldridge 2002](#), 10.4.4), is a semiparametric test for the null hypothesis that $\sigma_\mu^2 = 0$, i.e. that there are no unobserved effects in the residuals. Given that under the null the covariance matrix of the residuals for each individual is diagonal, the test statistic is based on the average of elements in the upper (or lower) triangle of its estimate, diagonal excluded: $n^{-1/2} \sum_{i=1}^n \sum_{t=1}^{T-1} \sum_{s=t+1}^T \hat{u}_{it} \hat{u}_{is}$ (where \hat{u} are the pooled OLS residuals), which must be “statistically close” to zero under the null, scaled by its standard deviation:

$$W = \frac{\sum_{i=1}^n \sum_{t=1}^{T-1} \sum_{s=t+1}^T \hat{u}_{it} \hat{u}_{is}}{[\sum_{i=1}^n (\sum_{t=1}^{T-1} \sum_{s=t+1}^T \hat{u}_{it} \hat{u}_{is})^2]^{1/2}}$$

This test is (n -) asymptotically distributed as a standard Normal regardless of the distribution of the errors. It does also not rely on homoskedasticity.

It has power both against the standard random effects specification, where the unobserved effects are constant within every group, as well as against any kind of serial correlation. As such, it “nests” both random effects and serial correlation tests, trading some power against more specific alternatives in exchange for robustness.

While not rejecting the null favours the use of pooled OLS, rejection may follow from serial correlation of different kinds, and in particular, quoting [Wooldridge \(2002\)](#), “should not be interpreted as implying that the random effects error structure *must* be true”.

Below, the test is applied to the data and model in [Munnell \(1990\)](#):

```
R> pwtest(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp, data=Produc)
      Wooldridge's test for unobserved individual effects

data: formula
z = 3.9383, p-value = 8.207e-05
alternative hypothesis: unobserved effect
```

Locally robust tests for serial correlation or random effects

The presence of random effects may affect tests for residual serial correlation, and the opposite.

One solution is to use a joint test, which has power against both alternatives. A joint LM test for random effects *and* serial correlation under normality and homoskedasticity of the idiosyncratic errors has been derived by [Baltagi and Li \(1991\)](#) and [Baltagi and Li \(1995\)](#) and is implemented as an option in **pbsytest**:

```
R> pbsytest(log(gsp)~log(pcap)+log(pc)+log(emp)+unemp, data=Produc, test="j")

Baltagi and Li AR-RE joint test

data: formula
chisq = 4187.6, df = 2, p-value < 2.2e-16
alternative hypothesis: AR(1) errors or random effects
```

Rejection of the joint test, though, gives no information on the direction of the departure from the null hypothesis, i.e.: is rejection due to the presence of serial correlation, of random effects or of both?

[Bera, Sosa-Escudero, and Yoon \(2001\)](#) derive locally robust tests both for individual random effects and for first-order serial correlation in residuals as “corrected” versions of the standard LM test (see **plmttest**). While still dependent on normality and homoskedasticity, these are robust to *local* departures from the hypotheses of, respectively, no serial correlation or no random effects. The authors observe that, although suboptimal, these tests may help detecting the right direction of the departure from the null, thus complementing the use of joint tests. Moreover, being based on pooled OLS residuals, the BSY tests are computationally far less demanding than likelihood-based conditional tests.

On the other hand, the statistical properties of these “locally corrected” tests are inferior to those of the non-corrected counterparts when the latter are correctly specified. If there is no serial correlation, then the optimal test for random effects is the likelihood-based LM test of Breusch and Godfrey (with refinements by Honda, see **plmttest**), while if there are no random effects the optimal test for serial correlation is, again, Breusch-Godfrey’s test⁹. If the presence of a random effect is taken for granted, then the optimal test for serial correlation is the likelihood-based conditional LM test of [Baltagi and Li \(1995\)](#) (see **pbltest**).

The serial correlation version is the default:

```
R> pbsytest(log(gsp)~log(pcap)+log(pc)+log(emp)+unemp, data=Produc)

Bera, Sosa-Escudero and Yoon locally robust test

data: formula
chisq = 52.636, df = 1, p-value = 4.015e-13
alternative hypothesis: AR(1) errors sub random effects
```

The BSY test for random effects is implemented in the one-sided version¹⁰, which takes heed that the variance of the random effect must be non-negative:

⁹LM₃ in [Baltagi and Li \(1995\)](#).

¹⁰Corresponding to RSO_{μ}^* in the original paper.

```
R> pbsytest(log(gsp)~log(pcap)+log(pc)+log(emp)+unemp, data=Produc, test="re")
Bera, Sosa-Escudero and Yoon locally robust test

data: formula
z = 57.914, p-value < 2.2e-16
alternative hypothesis: random effects sub AR(1) errors
```

Conditional LM test for AR(1) or MA(1) errors under random effects

Baltagi and Li (1991) and Baltagi and Li (1995) derive a Lagrange multiplier test for serial correlation in the idiosyncratic component of the errors under (normal, heteroskedastic) random effects. Under the null of serially uncorrelated errors, the test turns out to be identical for both the alternative of AR(1) and MA(1) processes. One- and two-sided versions are provided, the one-sided having power against positive serial correlation only. The two-sided is the default, while for the other one must specify the `alternative` option to `onesided`:

```
R> pbltest(log(gsp)~log(pcap)+log(pc)+log(emp)+unemp,
+           data=Produc, alternative="onesided")

Baltagi and Li one-sided LM test

data: log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp
z = 21.69, p-value < 2.2e-16
alternative hypothesis: AR(1)/MA(1) errors in RE panel model
```

As usual, the LM test statistic is based on residuals from the maximum likelihood estimate of the restricted model (random effects with serially uncorrelated errors). In this case, though, the restricted model cannot be estimated by OLS any more, therefore the testing function depends on `lme()` in the `nlme` package for estimation of a random effects model by maximum likelihood. For this reason, the test is applicable only to balanced panels.

No test has been implemented to date for the symmetric hypothesis of no random effects in a model with errors following an AR(1) process, but an asymptotically equivalent likelihood ratio test is available in the `nlme` package (see Section 7)..

General serial correlation tests

A general testing procedure for serial correlation in fixed effects (FE), random effects (RE) and pooled-OLS panel models alike can be based on considerations in (Wooldridge 2002, 10.7.2).

Recall that `plm` model objects are the result of OLS estimation performed on “demeaned” data, where, in the case of individual effects (else symmetric), this means time-demeaning for the FE (`within`) model, quasi-time-demeaning for the RE (`random`) model and original data, with no demeaning at all, for the pooled OLS (`pooling`) model (see Section 3).

For the random effects model, Wooldridge (2002) observes that under the null of homoskedasticity and no serial correlation in the idiosyncratic errors, the residuals from the quasi-demeaned regression must be spherical as well. Else, as the individual effects are wiped

out in the demeaning, any remaining serial correlation must be due to the idiosyncratic component. Hence, a simple way of testing for serial correlation is to apply a standard serial correlation test to the quasi-demeaned model. The same applies in a pooled model, w.r.t. the original data.

The FE case needs some qualification. It is well-known that if the original model's errors are uncorrelated then FE residuals are negatively serially correlated, with $\text{cor}(\hat{u}_{it}, \hat{u}_{is}) = -1/(T-1)$ for each t, s (see Wooldridge 2002, 10.5.4). This correlation clearly dies out as T increases, so this kind of AR test is applicable to **within** model objects only for T "sufficiently large"¹¹. On the converse, in short panels the test gets severely biased towards rejection (or, as the induced correlation is negative, towards acceptance in the case of the one-sided DW test with `alternative="greater"`). See below for a serial correlation test applicable to "short" FE panel models.

plm objects retain the "demeaned" data, so the procedure is straightforward for them. The wrapper functions `pbgtest` and `pdwtest` re-estimate the relevant quasi-demeaned model by `OLS` and apply, respectively, standard Breusch-Godfrey and Durbin-Watson tests from package `lmtest`:

```
R> ## this can be taken away as soon as attached to plm.rnw
R> grun.fe <- plm(inv ~ value + capital, data = Grunfeld, model = "within")
R> pbgtest(grun.fe, order = 2)

Breusch-Godfrey/Wooldridge test for serial correlation in panel models

data: inv ~ value + capital
chisq = 42.587, df = 2, p-value = 5.655e-10
alternative hypothesis: serial correlation in idiosyncratic errors
```

The tests share the features of their `OLS` counterparts, in particular the `pbgtest` allows testing for higher-order serial correlation, which might turn useful, e.g., on quarterly data. Analogously, from the point of view of software, as the functions are simple wrappers towards `bgtest` and `dwtest`, all arguments from the latter two apply and may be passed on through the '...' operator.

Wooldridge's test for serial correlation in "short" FE panels

For the reasons reported above, under the null of no serial correlation in the errors, the residuals of a FE model must be negatively serially correlated, with $\text{cor}(\hat{\epsilon}_{it}, \hat{\epsilon}_{is}) = -1/(T-1)$ for each t, s . Wooldridge suggests basing a test for this null hypothesis on a pooled regression of FE residuals on themselves, lagged one period:

$$\hat{\epsilon}_{i,t} = \alpha + \delta \hat{\epsilon}_{i,t-1} + \eta_{i,t}$$

¹¹Baltagi and Li derive a basically analogous T-asymptotic test for first-order serial correlation in a FE panel model as a Breusch-Godfrey LM test on within residuals (see Baltagi and Li 1995, par. 2.3 and formula 12). They also observe that the test on within residuals can be used for testing on the RE model, as "the within transformation [time-demeaning, in our terminology] wipes out the individual effects, whether fixed or random". Generalizing the Durbin-Watson test to FE models by applying it to fixed effects residuals is documented in Bhargava, Franzini, and Narendranathan (1982).

Rejecting the restriction $\delta = -1/(T - 1)$ makes us conclude against the original null of no serial correlation.

The building blocks available in **plm**, together with the function **linearHypothesis()** in package **car**, make it easy to construct a function carrying out this procedure: first the FE model is estimated and the residuals retrieved, then they are lagged and a pooling AR(1) model is estimated. The test statistic is obtained by applying **linearHypothesis()** to the latter model to test the above restriction on δ , supplying a heteroskedasticity- and autocorrelation-consistent covariance matrix (**vcovHC** with the appropriate options, in particular **method="arellano"**)¹².

```
R> pwartest(log(emp) ~ log(wage) + log(capital), data=EmplUK)

Wooldridge's test for serial correlation in FE panels

data: plm.model
chisq = 312.3, p-value < 2.2e-16
alternative hypothesis: serial correlation
```

The test is applicable to any FE panel model, and in particular to “short” panels with small T and large n .

Wooldridge’s first-difference-based test

In the context of the first difference model, Wooldridge (2002, 10.6.3) proposes a serial correlation test that can also be seen as a specification test to choose the most efficient estimator between fixed effects (**within**) and first difference (**fd**).

The starting point is the observation that if the idiosyncratic errors of the original model u_{it} are uncorrelated, the errors of the (first) differenced model¹³ $e_{it} \equiv u_{it} - u_{i,t-1}$ will be correlated, with $\text{cor}(e_{it}, e_{i,t-1}) = -0.5$, while any time-invariant effect, “fixed” or “random”, is wiped out in the differencing. So a serial correlation test for models with individual effects of any kind can be based on estimating the model

$$\hat{u}_{i,t} = \delta \hat{u}_{i,t-1} + \eta_{i,t}$$

and testing the restriction $\delta = -0.5$, corresponding to the null of no serial correlation. Drukker (2003) provides Monte Carlo evidence of the good empirical properties of the test.

On the other extreme (see Wooldridge 2002, 10.6.1), if the differenced errors e_{it} are uncorrelated, as by definition $u_{it} = u_{i,t-1} + e_{it}$, then u_{it} is a random walk. In this latter case, the most efficient estimator is the first difference (**fd**) one; in the former case, it is the fixed effects one (**within**).

The function **pwfptest** allows testing either hypothesis: the default behaviour **h0="fd"** is to test for serial correlation in *first-differenced* errors:

```
R> pwfdtest(log(emp) ~ log(wage) + log(capital), data=EmplUK)
```

¹²see Subsection 6.7.

¹³Here, e_{it} for notational simplicity (and as in Wooldridge): equivalent to $\Delta \epsilon_{it}$ in the general notation of the paper.

Wooldridge's first-difference test for serial correlation in panels

```
data: plm.model
chisq = 1.5251, p-value = 0.2169
alternative hypothesis: serial correlation in differenced errors
```

while specifying `h0="fe"` the null hypothesis becomes no serial correlation in *original* errors, which is similar to the `pwttest`.

```
R> pwttest(log(emp) ~ log(wage) + log(capital), data=EmplUK, h0="fe")
```

Wooldridge's first-difference test for serial correlation in panels

```
data: plm.model
chisq = 131.55, p-value < 2.2e-16
alternative hypothesis: serial correlation in original errors
```

Not rejecting one of the two is evidence in favour of using the estimator corresponding to `h0`. Should the truth lie in the middle (both rejected), whichever estimator is chosen will have serially correlated errors: therefore it will be advisable to use the autocorrelation-robust covariance estimators from the Subsection 6.7 in inference.

6.5. Tests for cross-sectional dependence

Next to the more familiar issue of serial correlation, over the last years a growing body of literature has been dealing with cross-sectional dependence (henceforth: XSD) in panels, which can arise, e.g., if individuals respond to common shocks (as in the literature on *factor models*) or if spatial diffusion processes are present, relating individuals in a way depending on a measure of distance (*spatial models*).

The subject is huge, and here we touch only some general aspects of misspecification testing and valid inference. If XSD is present, the consequence is, at a minimum, inefficiency of the usual estimators and invalid inference when using the standard covariance matrix¹⁴. The plan is to have in **plm** both misspecification tests to detect XSD and robust covariance matrices to perform valid inference in its presence, like in the serial dependence case. For now, though, only misspecification tests are included.

CD and LM-type tests for global cross-sectional dependence

The function `pcdtest` implements a family of XSD tests which can be applied in different settings, ranging from those where T grows large with n fixed to “short” panels with a big n dimension and a few time periods. All are based on (transformations of-) the product-moment correlation coefficient of a model’s residuals, defined as

$$\hat{\rho}_{ij} = \frac{\sum_{t=1}^T \hat{u}_{it} \hat{u}_{jt}}{(\sum_{t=1}^T \hat{u}_{it}^2)^{1/2} (\sum_{t=1}^T \hat{u}_{jt}^2)^{1/2}}$$

¹⁴This is the case, e.g., if in an unobserved effects model when XSD is due to an unobservable factor structure, with factors that are uncorrelated with the regressors. In this case the within or random estimators are still consistent, although inefficient (see De Hoyos and Sarafidis 2006).

i.e., as averages over the time dimension of pairwise correlation coefficients for each pair of cross-sectional units.

The Breusch-Pagan (Breusch and Pagan 1980) LM test, based on the squares of ρ_{ij} , is valid for $T \rightarrow \infty$ with n fixed; defined as

$$LM = \sum_{i=1}^{n-1} \sum_{j=i+1}^n T_{ij} \hat{\rho}_{ij}^2$$

where in the case of an unbalanced panel only pairwise complete observations are considered, and $T_{ij} = \min(T_i, T_j)$ with T_i being the number of observations for individual i ; else, if the panel is balanced, $T_{ij} = T$ for each i, j . The test is distributed as $\chi^2_{n(n-1)/2}$. It is inappropriate whenever the n dimension is “large”. A scaled version, applicable also if $T \rightarrow \infty$ and then $n \rightarrow \infty$ (as in some pooled time series contexts), is defined as

$$SCLM = \sqrt{\frac{1}{n(n-1)}} \left(\sum_{i=1}^{n-1} \sum_{j=i+1}^n \sqrt{T_{ij}} \hat{\rho}_{ij}^2 \right)$$

and distributed as a standard Normal.

Pesaran's (Pesaran 2004, 2015) CD test

$$CD = \sqrt{\frac{2}{n(n-1)}} \left(\sum_{i=1}^{n-1} \sum_{j=i+1}^n \sqrt{T_{ij}} \hat{\rho}_{ij} \right)$$

based on ρ_{ij} without squaring (also distributed as a standard Normal) is appropriate both in n - and in T -asymptotic settings. It has remarkable properties in samples of any practically relevant size and is robust to a variety of settings. The only big drawback is that the test loses power against the alternative of cross-sectional dependence if the latter is due to a factor structure with factor loadings averaging zero, that is, some units react positively to common shocks, others negatively.

The default version of the test is "cd". These tests are originally meant to use the residuals of separate estimation of one time-series regression for each cross-sectional unit, so this is the default behaviour of `pcdtest`.

```
R> pcdtest(inv~value+capital, data=Grunfeld)
```

```
Pesaran CD test for cross-sectional dependence in panels
```

```
data: inv ~ value + capital
z = 5.3401, p-value = 9.292e-08
alternative hypothesis: cross-sectional dependence
```

If a different model specification (`within`, `random`, ...) is assumed consistent, one can resort to its residuals for testing¹⁵ by specifying the relevant `model` type. The main argument of

¹⁵This is also the only solution when the time dimension's length is insufficient for estimating the heterogeneous model.

this function may be either a model of class **panelmodel** or a **formula** and a **data.frame**; in the second case, unless **model** is set to **NULL**, all usual parameters relative to the estimation of a **plm** model may be passed on. The test is compatible with any consistent **panelmodel** for the data at hand, with any specification of **effect**. E.g., specifying **effect="time"** or **effect="twoways"** allows to test for residual cross-sectional dependence after the introduction of time fixed effects to account for common shocks.

```
R> pcdtest(inv ~ value + capital, data=Grunfeld, model="within")
```

```
Pesaran CD test for cross-sectional dependence in panels
```

```
data: inv ~ value + capital
z = 4.6612, p-value = 3.144e-06
alternative hypothesis: cross-sectional dependence
```

If the time dimension is insufficient and **model=NULL**, the function defaults to estimation of a **within** model and issues a warning.

CD(p) test for local cross-sectional dependence

A *local* variant of the *CD* test, called *CD(p)* test (Pesaran 2004), takes into account an appropriate subset of *neighbouring* cross-sectional units to check the null of no XSD against the alternative of *local* XSD, i.e. dependence between neighbours only. To do so, the pairs of neighbouring units are selected by means of a binary proximity matrix like those used in spatial models. In the original paper, a regular ordering of observations is assumed, so that the m -th cross-sectional observation is a neighbour to the $(m - 1)$ -th and to the $(m + 1)$ -th. Extending the *CD(p)* test to irregular lattices, we employ the binary proximity matrix as a selector for discarding the correlation coefficients relative to pairs of observations that are not neighbours in computing the *CD* statistic. The test is then defined as

$$CD = \sqrt{\frac{1}{\sum_{i=1}^{n-1} \sum_{j=i+1}^n w(p)_{ij}}} \left(\sum_{i=1}^{n-1} \sum_{j=i+1}^n [w(p)]_{ij} \sqrt{T_{ij}} \hat{\rho}_{ij} \right)$$

where $[w(p)]_{ij}$ is the (i, j) -th element of the p -th order proximity matrix, so that if h, k are not neighbours, $[w(p)]_{hk} = 0$ and $\hat{\rho}_{hk}$ gets “killed”; this is easily seen to reduce to formula (14) in Pesaran (2004) for the special case considered in that paper. The same can be applied to the *LM* and *SCLM* tests.

Therefore, the *local* version of either test can be computed supplying an $n \times n$ matrix (of any kind coercible to **logical**), providing information on whether any pair of observations are neighbours or not, to the **w** argument. If **w** is supplied, only neighbouring pairs will be used in computing the test; else, **w** will default to **NULL** and all observations will be used. The matrix needs not really be binary, so commonly used “row-standardized” matrices can be employed as well: it is enough that neighbouring pairs correspond to nonzero elements in **w**¹⁶.

¹⁶The very comprehensive package **spdep** for spatial dependence analysis (see Bivand 2008) contains features for creating, lagging and manipulating *neighbour list* objects of class **nb**, that can be readily converted to and from proximity matrices by means of the **nb2mat** function. Higher orders of the *CD(p)* test can be obtained lagging the corresponding **nbs** through **nb1lag**.

6.6. Unit root tests

Preliminary results

We consider the following model:

$$y_{it} = \delta y_{it-1} + \sum_{L=1}^{p_i} \theta_i \Delta y_{it-L} + \alpha_m d_{mt} + \epsilon_{it}$$

The unit root hypothesis is $\rho = 1$. The model can be rewritten in difference:

$$\Delta y_{it} = \rho y_{it-1} + \sum_{L=1}^{p_i} \theta_i \Delta y_{it-L} + \alpha_m d_{mt} + \epsilon_{it}$$

So that the unit-root hypothesis is now $\rho = 0$.

Some of the unit-root tests for panel data are based on preliminary results obtained by running the above Augmented Dickey-Fuller regression.

First, we have to determine the optimal number of lags p_i for each time-series. Several possibilities are available. They all have in common that the maximum number of lags have to be chosen first. Then, p_i can be chosen using:

- the Swartz information criteria (SIC),
- the Akaike information criteria (AIC),
- the Hall method, which consist on removing the higher lags while it is not significant.

The ADF regression is run on $T - p_i - 1$ observations for each individual, so that the total number of observations is $n \times \tilde{T}$ where $\tilde{T} = T - p_i - 1$

\bar{p} is the average number of lags. Call e_i the vector of residuals.

Estimate the variance of the ϵ_i as:

$$\hat{\sigma}_{\epsilon_i}^2 = \frac{\sum_{t=p_i+1}^T e_{it}^2}{df_i}$$

Levin-Lin-Chu model

Then, compute artificial regressions of Δy_{it} and y_{it-1} on Δy_{it-L} and d_{mt} and get the two vectors of residuals z_{it} and v_{it} .

Standardize these two residuals and run the pooled regression of $z_{it}/\hat{\sigma}_i$ on $v_{it}/\hat{\sigma}_i$ to get $\hat{\rho}$, its standard deviation $\hat{\sigma}(\hat{\rho})$ and the t-statistic $t_{\hat{\rho}} = \hat{\rho}/\hat{\sigma}(\hat{\rho})$.

Compute the long run variance of y_i :

$$\hat{\sigma}_{y^i}^2 = \frac{1}{T-1} \sum_{t=2}^T \Delta y_{it}^2 + 2 \sum_{L=1}^{\bar{K}} w_{KL} \left[\frac{1}{T-1} \sum_{t=2+L}^T \Delta y_{it} \Delta y_{it-L} \right]$$

Define \bar{s}_i as the ratio of the long and short term variance and \bar{s} the mean for all the individuals of the sample

$$s_i = \frac{\hat{\sigma}_{yi}}{\hat{\sigma}_{\epsilon_i}}$$

$$\bar{s} = \frac{\sum_{i=1}^n s_i}{n}$$

$$t_\rho^* = \frac{t_\rho - n\bar{T}\bar{s}\hat{\sigma}_\epsilon^{-2}\hat{\sigma}(\hat{\rho})\mu_{m\tilde{T}}^*}{\sigma_{m\tilde{T}}^*}$$

follows a normal distribution under the null hypothesis of stationarity. $\mu_{m\tilde{T}}^*$ and $\sigma_{m\tilde{T}}^*$ are given in table 2 of the original paper and are also available in the package.

Im, Pesaran and Shin test

This test does not require that ρ is the same for all the individuals. The null hypothesis is still that all the series have an unit root, but the alternative is that some may have a unit root and others have different values of $\rho_i < 0$.

The test is based on the average of the student statistic of the ρ obtained for each individual:

$$\bar{t} = \frac{1}{n} \sum_{i=1}^n t_{\rho i}$$

The statistic is then:

$$z = \frac{\sqrt{n}(\bar{t} - E(\bar{t}))}{\sqrt{V(\bar{t})}}$$

$\mu_{m\tilde{T}}^*$ and $\sigma_{m\tilde{T}}^*$ are given in table 2 of the original paper and are also available in the package.

6.7. Robust covariance matrix estimation

Robust estimators of the covariance matrix of coefficients are provided, mostly for use in Wald-type tests. `vcovHC` estimates three “flavours” of White’s heteroskedasticity-consistent covariance matrix¹⁷ (known as the *sandwich* estimator). Interestingly, in the context of panel data the most general version also proves consistent vs. serial correlation.

All types assume no correlation between errors of different groups while allowing for heteroskedasticity across groups, so that the full covariance matrix of errors is $V = I_n \otimes \Omega_i; i = 1, \dots, n$. As for the *intragroup* error covariance matrix of every single group of observations, “`white1`” allows for general heteroskedasticity but no serial correlation, *i.e.*

$$\Omega_i = \begin{bmatrix} \sigma_{i1}^2 & \dots & \dots & 0 \\ 0 & \sigma_{i2}^2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & \dots & \sigma_{iT}^2 \end{bmatrix} \quad (16)$$

¹⁷See White (1980) and White (1984).

while "white2" is "white1" restricted to a common variance inside every group, estimated as $\sigma_i^2 = \sum_{t=1}^T \hat{u}_{it}^2/T$, so that $\Omega_i = I_T \otimes \sigma_i^2$ (see [Greene \(2003, 13.7.1–2\)](#) and [Wooldridge \(2002, 10.7.2\)](#)); "arellano" (see *ibid.* and the original ref. [Arellano 1987](#)) allows a fully general structure w.r.t. heteroskedasticity and serial correlation:

$$\Omega_i = \begin{bmatrix} \sigma_{i1}^2 & \sigma_{i1,i2} & \dots & \dots & \sigma_{i1,iT} \\ \sigma_{i2,i1} & \sigma_{i2}^2 & & & \vdots \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \sigma_{iT-1}^2 & \sigma_{iT-1,iT} \\ \sigma_{iT,i1} & \dots & \dots & \sigma_{iT,iT-1} & \sigma_{iT}^2 \end{bmatrix} \quad (17)$$

The latter is, as already observed, consistent w.r.t. timewise correlation of the errors, but on the converse, unlike the White 1 and 2 methods, it relies on large n asymptotics with small T .

The fixed effects case, as already observed in Section 6.4 on serial correlation, is complicated by the fact that the demeaning induces serial correlation in the errors. The original White estimator (`white1`) turns out to be inconsistent for fixed T as n grows, so in this case it is advisable to use the `arellano` version (see [Stock and Watson 2008](#)).

The errors may be weighted according to the schemes proposed by [MacKinnon and White \(1985\)](#) and [Cribari-Neto \(2004\)](#) to improve small-sample performance¹⁸.

The main use of `vcovHC` is together with testing functions from the `lmtest` and `car` packages. These typically allow passing the `vcov` parameter either as a matrix or as a function (see [Zeileis 2004](#)). If one is happy with the defaults, it is easiest to pass the function itself:

```
R> library("lmtest")
R> re <- plm(inv~value+capital, data=Grunfeld, model="random")
R> coeftest(re, vcovHC)

t test of coefficients:

            Estimate Std. Error t value Pr(>|t|)
(Intercept) -57.834415  23.449626 -2.4663  0.01451 *
value         0.109781   0.012984  8.4551 6.186e-15 ***
capital       0.308113   0.051889  5.9379 1.284e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

else one may do the covariance computation inside the call to `coeftest`, thus passing on a matrix:

```
R> coeftest(re, vcovHC(re, method="white2", type="HC3"))
```

¹⁸The HC3 and HC4 weighting schemes are computationally expensive and may hit memory limits for nT in the thousands, where on the other hand it makes little sense to apply small sample corrections.

For some tests, e.g. for multiple model comparisons by **waldtest**, one should always provide a function¹⁹. In this case, optional parameters are provided as shown below (see also [Zeileis 2004](#), p.12):

```
R> waldtest(re, update(re,.~.-capital),
+           vcov=function(x) vcovHC(x, method="white2", type="HC3"))

Wald test

Model 1: inv ~ value + capital
Model 2: inv ~ value
  Res.Df Df  Chisq Pr(>Chisq)
1     197
2     198 -1 87.828 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Moreover, **linearHypothesis** from package **car** may be used to test for linear restrictions:

```
R> library("car")
R> linearHypothesis(re, "2*value=capital", vcov.=vcovHC)
```

Linear hypothesis test

Hypothesis:
2 value - capital = 0

Model 1: restricted model
Model 2: inv ~ value + capital

Note: Coefficient covariance matrix supplied.

```
  Res.Df Df  Chisq Pr(>Chisq)
1     198
2     197   1 3.4783    0.06218 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

A specific **vcovHC** method for **pgmm** objects is also provided which implements the robust covariance matrix proposed by [Windmeijer \(2005\)](#) for generalized method of moments estimators.

7. plm versus nlme/lme4

¹⁹Joint zero-restriction testing still allows providing the **vcov** of the unrestricted model as a matrix, see the documentation of package **lmtest**.

The models termed *panel* by the econometricians have counterparts in the statistics literature on *mixed* models (or *hierarchical models*, or *models for longitudinal data*), although there are both differences in jargon and more substantial distinctions. This language inconsistency between the two communities, together with the more complicated general structure of statistical models for longitudinal data and the associated notation in the software, is likely to scare some practicing econometricians away from some potentially useful features of the R environment, so it may be useful to provide here a brief reconciliation between the typical panel data specifications used in econometrics and the general framework used in statistics for mixed models²⁰.

R is particularly strong on mixed models' estimation, thanks to the long-standing **nlme** package (see Pinheiro *et al.* 2007) and the more recent **lme4** package, based on S4 classes (see Bates 2007)²¹. In the following we will refer to the more established **nlme** to give some examples of "econometric" panel models that can be estimated in a likelihood framework, also including some likelihood ratio tests. Some of them are not feasible in **plm** and make a useful complement to the econometric "toolbox" available in R.

7.1. Fundamental differences between the two approaches

Econometrics deal mostly with non-experimental data. Great emphasis is put on specification procedures and misspecification testing. Model specifications tend therefore to be very simple, while great attention is put on the issues of endogeneity of the regressors, dependence structures in the errors and robustness of the estimators under deviations from normality. The preferred approach is often semi- or non-parametric, and heteroskedasticity-consistent techniques are becoming standard practice both in estimation and testing.

For all these reasons, although the maximum likelihood framework is important in testing²² and sometimes used in estimation as well, panel model estimation in econometrics is mostly accomplished in the generalized least squares framework based on Aitken's Theorem and, when possible, in its special case OLS, which are free from distributional assumptions (although these kick in at the diagnostic testing stage). On the contrary, longitudinal data models in **nlme** and **lme4** are estimated by (restricted or unrestricted) maximum likelihood. While under normality, homoskedasticity and no serial correlation of the errors OLS are also the maximum likelihood estimator, in all the other cases there are important differences.

The econometric GLS approach has closed-form analytical solutions computable by standard linear algebra and, although the latter can sometimes get computationally heavy on the machine, the expressions for the estimators are usually rather simple. ML estimation of longitudinal models, on the contrary, is based on numerical optimization of nonlinear functions without closed-form solutions and is thus dependent on approximations and convergence criteria. For example, the "GLS" functionality in **nlme** is rather different from its "econometric" counterpart. "Feasible GLS" estimation in **plm** is based on a single two-step procedure, in which an

²⁰This discussion does not consider GMM models. One of the basic reasons for econometricians not to choose maximum likelihood methods in estimation is that the strict exogeneity of regressors assumption required for consistency of the ML models reported in the following is often inappropriate in economic settings.

²¹The standard reference on the subject of mixed models in S/R is Pinheiro and Bates (2000).

²²Lagrange Multiplier tests based on the likelihood principle are suitable for testing against more general alternatives on the basis of a maintained model with spherical residuals and find therefore application in testing for departures from the classical hypotheses on the error term. The seminal reference is Breusch and Pagan (1980).

inefficient but consistent estimation method (typically OLS) is employed first in order to get a consistent estimate of the errors' covariance matrix, to be used in GLS at the second step; on the converse, "GLS" estimators in **nlme** are based on iteration until convergence of two-step optimization of the relevant likelihood.

7.2. Some false friends

The *fixed/random effects* terminology in econometrics is often recognized to be misleading, as both are treated as random variates in modern econometrics (see e.g. Wooldridge 2002, 10.2.1). It has been recognized since Mundlak's classic paper (Mundlak 1978) that the fundamental issue is whether the unobserved effects are correlated with the regressors or not. In this last case, they can safely be left in the error term, and the serial correlation they induce is cared for by means of appropriate GLS transformations. On the contrary, in the case of correlation, "fixed effects" methods such as least squares dummy variables or time-demeaning are needed, which explicitly, although inconsistently²³, estimate a group- (or time-) invariant additional parameter for each group (or time period).

Thus, from the point of view of model specification, having *fixed effects* in an econometric model has the meaning of allowing the intercept to vary with group, or time, or both, while the other parameters are generally still assumed to be homogeneous. Having *random effects* means having a group- (or time-, or both) specific component in the error term.

In the mixed models literature, on the contrary, *fixed effect* indicates a parameter that is assumed constant, while *random effects* are parameters that vary randomly around zero according to a joint multivariate Normal distribution.

So, the FE model in econometrics has no counterpart in the mixed models framework, unless reducing it to OLS on a specification with one dummy for each group (often termed *least squares dummy variables*, or LSDV model) which can trivially be estimated by OLS. The RE model is instead a special case of a mixed model where only the intercept is specified as a random effect, while the "random" type variable coefficients model can be seen as one that has the same regressors in the fixed and random sets. The unrestricted generalized least squares can in turn be seen, in the **nlme** framework, as a standard linear model with a general error covariance structure within the groups and errors uncorrelated across groups.

7.3. A common taxonomy

To reconcile the two terminologies, in the following we report the specification of the panel models in **plm** according to the general expression of a mixed model in Laird-Ware form (see the web appendix to Fox 2002) and the **nlme** estimation commands for maximum likelihood estimation of an equivalent specification²⁴.

²³For fixed effects estimation, as the sample grows (on the dimension on which the fixed effects are specified) so does the number of parameters to be estimated. Estimation of individual fixed effects is $T-$ (but not $n-$) consistent, and the opposite.

²⁴In doing so, we stress that "equivalence" concerns only the specification of the model, and neither the appropriateness nor the relative efficiency of the relevant estimation techniques, which will of course be dependent on the context. Unlike their mixed model counterparts, the specifications in **plm** are, strictly speaking, distribution-free. Nevertheless, for the sake of exposition, in the following we present them in the setting which ensures consistency and efficiency (e.g., we consider the hypothesis of spherical errors part of the specification of pooled OLS and so forth).

The Laird-Ware representation for mixed models

A general representation for the linear mixed effects model is given in [Laird and Ware \(1982\)](#).

$$\begin{aligned} y_{it} &= \beta_1 x_{1ij} + \dots + \beta_p x_{pij} \\ &\quad b_1 z_{1ij} + \dots + b_p z_{pij} + \epsilon_{ij} \\ b_{ik} &\sim N(0, \psi_k^2), \text{ Cov}(b_k, b_{k'}) = \psi_{kk'} \\ \epsilon_{ij} &\sim N(0, \sigma^2 \lambda_{ijj}), \text{ Cov}(\epsilon_{ij}, \epsilon_{ij'}) = \sigma^2 \lambda_{ijj'} \end{aligned}$$

where the x_1, \dots, x_p are the fixed effects regressors and the z_1, \dots, z_p are the random effects regressors, assumed to be normally distributed across groups. The covariance of the random effects coefficients $\psi_{kk'}$ is assumed constant across groups and the covariances between the errors in group i , $\sigma^2 \lambda_{ijj'}$, are described by the term $\lambda_{ijj'}$ representing the correlation structure of the errors within each group (e.g., serial correlation over time) scaled by the common error variance σ^2 .

Pooling and Within

The *pooling* specification in **plm** is equivalent to a classical linear model (i.e., no random effects regressor and spherical errors: $b_{iq} = 0 \forall i, q$, $\lambda_{ijj} = \sigma^2$ for $j = j'$, 0 else). The *within* one is the same with the regressors' set augmented by $n - 1$ group dummies. There is no point in using **nlme** as parameters can be estimated by OLS which is also ML.

Random effects

In the Laird and Ware notation, the RE specification is a model with only one random effects regressor: the intercept. Formally, $z_{1ij} = 1 \forall i, j$, $z_{qij} = 0 \forall i, \forall j, \forall q \neq 1$ $\lambda_{ij} = 1$ for $i = j$, 0 else). The composite error is therefore $u_{ij} = 1b_{i1} + \epsilon_{ij}$. Below we report coefficients of Grunfeld's model estimated by GLS and then by ML

```
R> library(nlme)
R> reGLS <- plm(inv~value+capital, data=Grunfeld, model="random")
R> reML <- lme(inv~value+capital, data=Grunfeld, random=~1/firm)
R> coef(reGLS)

(Intercept)      value      capital
-57.8344149   0.1097812   0.3081130

R> summary(reML)$coefficients$fixed

(Intercept)      value      capital
-57.8644245   0.1097897   0.3081881
```

Variable coefficients, “random”

Swamy's variable coefficients model ([Swamy 1970](#)) has coefficients varying randomly (and independently of each other) around a set of fixed values, so the equivalent specification

is $z_q = x_q \forall q$, i.e. the fixed effects and the random effects regressors are the same, and $\psi_{kk'} = \sigma_\mu^2 I_N$, and $\lambda_{ijj} = 1$, $\lambda_{ijj'} = 0$ for $j \neq j'$, that's to say they are not correlated.

Estimation of a mixed model with random coefficients on all regressors is rather demanding from the computational side. Some models from our examples fail to converge. The below example is estimated on the Grunfeld data and model with time effects.

```
R> vcm <- pvcv(inv~value+capital, data=Grunfeld, model="random", effect="time")

[1] 6.318535e-04 -2.453520e-02 -1.410394e+03
attention

R> vcmML <- lme(inv~value+capital, data=Grunfeld, random=~value+capital/year)
R> coef(vcm)

y
(Intercept) -18.5538638
value         0.1239595
capital       0.1114579

R> summary(vcmML)$coefficients$fixed

(Intercept)      value      capital
-26.3558395   0.1241982   0.1381782
```

Variable coefficients, “within”

This specification actually entails separate estimation of T different standard linear models, one for each group in the data, so the estimation approach is the same: OLS. In **nlme** this is done by creating an **lmList** object, so that the two models below are equivalent (output suppressed):

```
R> vcmf <- pvcv(inv~value+capital, data=Grunfeld, model="within", effect="time")
R> vcmfML <- lmList(inv~value+capital/year, data=Grunfeld)
```

Unrestricted fgls

The general, or unrestricted, feasible GLS, **pggls** in the **plm** nomenclature, is equivalent to a model with no random effects regressors ($b_{iq} = 0 \forall i, q$) and an error covariance structure which is unrestricted within groups apart from the usual requirements. The function for estimating such models with correlation in the errors but no random effects is **gls()**.

This very general serial correlation and heteroskedasticity structure is not estimable for the original Grunfeld data, which have more time periods than firms, therefore we restrict them to firms 4 to 6.

```
R> sGrunfeld <- Grunfeld[Grunfeld$firm%in%4:6,]
R> ggls <- pggls(inv~value+capital, data=sGrunfeld, model="pooling")
R> gglsML <- gls(inv~value+capital, data=sGrunfeld,
+                   correlation=corSymm(form=~1/year))
R> coef(ggls)

(Intercept)      value      capital
1.19679342   0.10555908   0.06600166

R> summary(gglsML)$coefficients

(Intercept)      value      capital
-2.4156266    0.1163550   0.0735837
```

The *within* case is analogous, with the regressors' set augmented by $n - 1$ group dummies.

7.4. Some useful “econometric” models in nlme

Finally, amongst the many possible specifications estimable with **nlme**, we report a couple cases that might be especially interesting to applied econometricians.

AR(1) pooling or random effects panel

Linear models with groupwise structures of time-dependence²⁵ may be fitted by **gls()**, specifying the correlation structure in the **correlation** option²⁶:

```
R> Grunfeld$year <- as.numeric(as.character(Grunfeld$year))
R> lmAR1ML <- gls(inv~value+capital,data=Grunfeld,
+                     correlation=corAR1(0,form=~year/firm))
```

and analogously the random effects panel with, e.g., AR(1) errors (see Baltagi 2001, chap 5), which is a very common specification in econometrics, may be fit by **lme** specifying an additional random intercept:

```
R> reAR1ML <- lme(inv~value+capital, data=Grunfeld,random=~1/firm,
+                     correlation=corAR1(0,form=~year/firm))
```

The regressors' coefficients and the error's serial correlation coefficient may be retrieved this way:

```
R> summary(reAR1ML)$coefficients$fixed

(Intercept)      value      capital
-40.27650822   0.09336672   0.31323330
```

²⁵Take heed that here, in contrast to the usual meaning of serial correlation in time series, we always speak of serial correlation *between the errors of each group*.

²⁶note that the time index is coerced to numeric before the estimation.

```
R> coef(reAR1ML$modelStruct$corStruct, unconstrained=FALSE)
```

```
Phi
0.823845
```

Significance statistics for the regressors' coefficients are to be found in the usual **summary** object, while to get the significance test of the serial correlation coefficient one can do a likelihood ratio test as shown in the following.

An LR test for serial correlation and one for random effects

A likelihood ratio test for serial correlation in the idiosyncratic residuals can be done as a nested models test, by **anova()**, comparing the model with spherical idiosyncratic residuals with the more general alternative featuring AR(1) residuals. The test takes the form of a zero restriction test on the autoregressive parameter.

This can be done on pooled or random effects models alike. First we report the simpler case. We already estimated the pooling AR(1) model above. The GLS model without correlation in the residuals is the same as OLS, and one could well use **lm()** for the restricted model. Here we estimate it by **gls()**.

```
R> lmML <- gls(inv~value+capital, data=Grunfeld)
```

```
R> anova(lmML, lmAR1ML)
```

	Model	df	AIC	BIC	logLik	Test	L.Ratio	p-value
lmML		1	4	2400.217	2413.350	-1196.109		
lmAR1ML		2	5	2094.936	2111.352	-1042.468	1 vs 2	307.2813 <.0001

The AR(1) test on the random effects model is to be done in much the same way, using the random effects model objects estimated above:

```
R> anova(reML, reAR1ML)
```

	Model	df	AIC	BIC	logLik	Test	L.Ratio	p-value
reML		1	5	2205.851	2222.267	-1097.926		
reAR1ML		2	6	2094.802	2114.501	-1041.401	1 vs 2	113.0496 <.0001

A likelihood ratio test for random effects compares the specifications with and without random effects and spherical idiosyncratic errors:

```
R> anova(lmML, reML)
```

	Model	df	AIC	BIC	logLik	Test	L.Ratio	p-value
lmML		1	4	2400.217	2413.350	-1196.109		
reML		2	5	2205.851	2222.267	-1097.926	1 vs 2	196.366 <.0001

The random effects, AR(1) errors model in turn nests the AR(1) pooling model, therefore a likelihood ratio test for random effects sub AR(1) errors may be carried out, again, by comparing the two autoregressive specifications:

```
R> anova(1mAR1ML, reAR1ML)
```

	Model	df	AIC	BIC	logLik	Test	L.Ratio	p-value
1mAR1ML		1	5	2094.936	2111.352	-1042.468		
reAR1ML		2	6	2094.802	2114.501	-1041.401	1 vs 2	2.134349 0.144

whence we see that the Grunfeld model specification doesn't seem to need any random effects once we control for serial correlation in the data.

8. Conclusions

With **plm** we aim at providing a comprehensive package containing the standard functionalities that are needed for the management and the econometric analysis of panel data. In particular, we provide: functions for data transformation; estimators for pooled, random and fixed effects static panel models and variable coefficients models, general GLS for general covariance structures, and generalized method of moments estimators for dynamic panels; specification and diagnostic tests. Instrumental variables estimation is supported. Most estimators allow working with unbalanced panels. While among the different approaches to longitudinal data analysis we take the perspective of the econometrician, the syntax is consistent with the basic linear modeling tools, like the **lm** function.

On the input side, **formula** and **data** arguments are used to specify the model to be estimated. Special functions are provided to make writing formulas easier, and the structure of the data is indicated with an **index** argument.

On the output side, the model objects (of the new class **panelmodel**) are compatible with the general restriction testing frameworks of packages **lmttest** and **car**. Specialized methods are also provided for the calculation of robust covariance matrices; heteroskedasticity- and correlation-consistent testing is accomplished by passing these on to testing functions, together with a **panelmodel** object.

The main functionalities of the package have been illustrated here by applying them on some well-known datasets from the econometric literature. The similarities and differences with the maximum likelihood approach to longitudinal data have also been briefly discussed.

We plan to expand the methods in this paper to systems of equations and to the estimation of models with autoregressive errors. Addition of covariance estimators robust vs. cross-sectional correlation are also in the offing. Lastly, conditional visualization features in the R environment seem to offer a promising toolbox for visual diagnostics, which is another subject for future work.

Acknowledgments

While retaining responsibility for any error, we thank Jeffrey Wooldridge, Achim Zeileis and three anonymous referees for useful comments. We also acknowledge kind editing assistance by Lisa Benedetti.

References

- Amemiya T (1971). “The Estimation of the Variances in a Variance–Components Model.” *International Economic Review*, **12**(1), 1–13.
- Anderson T, Hsiao C (1981). “Estimation of Dynamic Models With Error Components.” *Journal of the American Statistical Association*, **76**(375), 598–606.
- Arellano M (1987). “Computing Robust Standard Errors for Within–Group Estimators.” *Oxford Bulletin of Economics and Statistics*, **49**(4), 431–434.
- Arellano M, Bond S (1991). “Some Tests of Specification for Panel Data : Monte Carlo Evidence and an Application to Employment Equations.” *Review of Economic Studies*, **58**(2), 277–297.
- Balestra P, Varadharajan-Krishnakumar J (1987). “Full Information Estimations of a System of Simultaneous Equations With Error Components.” *Econometric Theory*, **3**(2), 223–246.
- Baltagi B (1981). “Simultaneous Equations With Error Components.” *Journal of Econometrics*, **17**(2), 189–200.
- Baltagi B (2001). *Econometric Analysis of Panel Data*. 3rd edition. John Wiley and Sons ltd.
- Baltagi B, Li Q (1991). “A Joint Test for Serial Correlation and Random Individual Effects.” *Statistics and Probability Letters*, **11**(3), 277–280.
- Baltagi B, Li Q (1995). “Testing AR(1) Against MA(1) Disturbances in an Error Component Model.” *Journal of Econometrics*, **68**(1), 133–151.
- Bates D (2004). “Least Squares Calculations in R.” *R-news*, **4**(1), 17–20.
- Bates D (2007). *lme4: Linear Mixed-Effects Models Using S4 Classes*. R package version 0.99875-9, URL <https://cran.r-project.org/package=lme4>.
- Bates D, Maechler M (2016). *Matrix: Sparse and Dense Matrix Classes and Methods*. R package version 1.2-7.1, URL <https://cran.r-project.org/package=Matrix>.
- Bera A, Sosa-Escudero W, Yoon M (2001). “Tests for the Error Component Model in the Presence of Local Misspecification.” *Journal of Econometrics*, **101**(1), 1–23.
- Bhargava A, Franzini L, Narendranathan W (1982). “Serial Correlation and the Fixed Effects Model.” *Review of Economic Studies*, **49**(4), 533–554.
- Bivand R (2008). *spdep: Spatial Dependence: Weighting Schemes, Statistics and Models*. R package version 0.4-17, URL <https://cran.r-project.org/package=spdep>.
- Blundell R, Bond S (1998). “Initial Conditions and Moment Restrictions in Dynamic Panel Data Models.” *Journal of Econometrics*, **87**(1), 115–143.
- Breusch T, Pagan A (1980). “The Lagrange Multiplier Test and Its Applications to Model Specification in Econometrics.” *Review of Economic Studies*, **47**(1), 239–253.

- Cornwell C, Rupert P (1988). “Efficient Estimation With Panel Data: An Empirical Comparison of Instrumental Variables Estimators.” *Journal of Applied Econometrics*, **3**(2), 149–155.
- Cribari-Neto F (2004). “Asymptotic Inference Under Heteroskedasticity of Unknown Form.” *Computational Statistics & Data Analysis*, **45**(2), 215–233.
- Croissant Y, Millo G (2008). “Panel Data Econometrics in R: The **plm** Package.” *Journal of Statistical Software*, **27**(2). URL <http://www.jstatsoft.org/v27/i02/>.
- De Hoyos R, Sarafidis V (2006). “Testing for Cross–Sectional Dependence in Panel–Data Models.” *The Stata Journal*, **6**(4), 482–496.
- Drukker D (2003). “Testing for Serial Correlation in Linear Panel–Data Models.” *The Stata Journal*, **3**(2), 168–177.
- Fox J (2002). *An R and S-plus Companion to Applied Regression*. Sage.
- Fox J (2016). *car: Companion to Applied Regression*. R package version 2.1-3, URL <https://cran.r-project.org/package=car>, <http://socserv.socsci.mcmaster.ca/jfox/>.
- Gourieroux C, Holly A, Monfort A (1982). “Likelihood Ratio Test, Wald Test, and Kuhn–Tucker Test in Linear Models With Inequality Constraints on the Regression Parameters.” *Econometrica*, **50**(1), 63–80.
- Greene W (2003). *Econometric Analysis*. 5th edition. Prentice Hall.
- Harrison D, Rubinfeld D (1978). “Hedonic housing prices and the demand for clean air.” *Journal of Environmental Economics and Management*, **5**(1), 81–102.
- Hausman J (1978). “Specification Tests in Econometrics.” *Econometrica*, **46**(6), 1251–1271.
- Hausman J, Taylor W (1981). “Panel Data and Unobservable Individual Effects.” *Econometrica*, **49**(6), 1377–1398.
- Holtz-Eakin D, Newey W, Rosen H (1988). “Estimating Vector Autoregressions With Panel Data.” *Econometrica*, **56**(6), 1371–1395.
- Honda Y (1985). “Testing the Error Components Model With Non–Normal Disturbances.” *Review of Economic Studies*, **52**(4), 681–690.
- Hothorn T, Zeileis A, Farebrother RW, Cummins C, Millo G, Mitchell D (2015). *lmtest: Testing Linear Regression Models*. R package version 0.9-34, URL <https://cran.r-project.org/package=lmtest>.
- King M, Wu P (1997). “Locally Optimal One–Sided Tests for Multiparameter Hypotheses.” *Econometric Reviews*, **16**(2), 131–156.
- Kleiber C, Zeileis A (2008). *Applied Econometrics with R*. Springer-Verlag, New York. ISBN 978-0-387-77316-2, URL <https://cran.r-project.org/package=AER>.
- Koenker R, Ng P (2016). *SparseM: Sparse Linear Algebra*. R package version 1.72, URL <https://cran.r-project.org/package=SparseM>.

- Laird N, Ware J (1982). “Random–Effects Models for Longitudinal Data.” *Biometrics*, **38**(4), 963–974.
- Lumley T, Zeileis A (2015). *sandwich: Robust Covariance Matrix Estimators*. R package version 2.3-4, URL <https://cran.r-project.org/package=sandwich>.
- MacKinnon J, White H (1985). “Some Heteroskedasticity–Consistent Covariance Matrix Estimators With Improved Finite Sample Properties.” *Journal of Econometrics*, **29**(3), 305–325.
- Mundlak Y (1978). “On the Pooling of Time Series and Cross Section Data.” *Econometrica*, **46**(1), 69–85.
- Munnell A (1990). “Why Has Productivity Growth Declined? Productivity and Public Investment.” *New England Economic Review*, pp. 3–22.
- Nerlove M (1971). “Further Evidence on the Estimation of Dynamic Economic Relations from a Time–Series of Cross–Sections.” *Econometrica*, **39**(2), 359–382.
- Pesaran M (2004). “General Diagnostic Tests for Cross Section Dependence in Panels.” CESifo Working Paper Series, 1229.
- Pesaran M (2015). “Testing Weak Cross–Sectional Dependence in Large Panels.” *Econometric Reviews*, **34**(6-10), 1089–1117.
- Pinheiro J, Bates D (2000). *Mixed–Effects Models in S and S-plus*. Springer-Verlag.
- Pinheiro J, Bates D, DebRoy S, the R Core team DS (2007). *nlme: Linear and Nonlinear Mixed Effects Models*. R package version 3.1-86, URL <https://cran.r-project.org/package=nlme>.
- R Development Core Team (2008). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL <http://www.r-project.org/>.
- Stock JH, Watson MW (2008). “Heteroskedasticity–Robust Standard Errors for Fixed Effects Panel Data Regression.” *Econometrica*, **76**(1), 155–174.
- Swamy P (1970). “Efficient Inference in a Random Coefficient Regression Model.” *Econometrica*, **38**(2), 311–323.
- Swamy P, Arora S (1972). “The Exact Finite Sample Properties of the Estimators of Coefficients in the Error Components Regression Models.” *Econometrica*, **40**(2), 261–275.
- Therneau T (2014). *bdsmatrix: Routines for Block Diagonal Symmetric matrices*. R package version 1.3-2, URL <https://cran.r-project.org/package=bdsmatrix>.
- Wallace T, Hussain A (1969). “The Use of Error Components Models in Combining Cross Section With Time Series Data.” *Econometrica*, **37**(1), 55–72.
- White H (1980). *Asymptotic Theory for Econometricians*. Academic Press, Orlando.
- White H (1984). “A Heteroskedasticity–Consistent Covariance Matrix and a Direct Test for Heteroskedasticity.” *Econometrica*, **48**(4), 817–838.

- Windmeijer F (2005). “A Finite Sample Correction for the Variance of Linear Efficient Two-Step GMM Estimators.” *Journal of Econometrics*, **126**(1), 25–51.
- Wooldridge J (2002). *Econometric Analysis of Cross-Section and Panel Data*. MIT press.
- Zeileis A (2004). “Econometric Computing With HC and HAC Covariance Matrix Estimators.” *Journal of Statistical Software*, **11**(10), 1–17. URL <http://www.jstatsoft.org/v11/i10/>.

Affiliation:

Yves Croissant
CEMOI
Faculté de Droit et d’Economie
Université de La Réunion
15 avenue René Cassin
CS 92003
F-97744 Saint-Denis Cedex 9
Telephone: +262262938446
E-mail: yves.croissant@univ-reunion.fr

Giovanni Millo
DiSES, Un. of Trieste and R&D Dept., Generali SpA
Via Machiavelli 4
34131 Trieste (Italy)
Telephone: +39/040/671184
Fax: +39/040/671160
E-mail: Giovanni_Millo@Generali.com

Sutton said, “that’s where the money is”. Working for a startup is really hard. The best part is that there are no rules. The worst part, so far, is that we have no revenues, although we’re moving nicely to change this. Life is fun and it’s a great time to be doing statistical graphics.

I’m looking forward to seeing everyone in NYC this August. If you don’t see me, it means that I may have failed to solve our revenue problem. Please consider getting involved with section activities. The health of

our section, one of the largest in ASA, depends on volunteers.

Stephen G. Eick, Ph.D.
Co-founder and CTO Visintuit
eick@visintuit.com
630-778-0050



TOPICS IN STATISTICAL COMPUTING

An Introduction to the Bootstrap with Applications in R

A. C. Davison and Diego Kuonen

kuonen@statoo.com

Introduction

Bootstrap methods are resampling techniques for assessing uncertainty. They are useful when inference is to be based on a complex procedure for which theoretical results are unavailable or not useful for the sample sizes met in practice, where a standard model is suspect but it is unclear with what to replace it, or where a ‘quick and dirty’ answer is required. They can also be used to verify the usefulness of standard approximations for parametric models, and to improve them if they seem to give inadequate inferences. This article, a brief introduction on their use, is based closely on parts of Davison and Hinkley (1997), where further details and many examples and practicals can be found. A different point of view is given by Efron and Tibshirani (1993) and a more mathematical survey by Shao and Tu (1995), while Hall (1992) describes the underlying theory.

Basic Ideas

The simplest setting is when the observed data y_1, \dots, y_n are treated as a realisation of a random sample Y_1, \dots, Y_n from an unknown underlying distribution F . Interest is focused on a parameter θ , the outcome of applying the statistical functional $t(\cdot)$ to F , so $\theta = t(F)$. The simplest example of such a functional is the average, $t(F) = \int y dF(y)$; in general we think of $t(\cdot)$ as an algorithm to be applied to F .

The estimate of θ is $t = t(\hat{F})$, where \hat{F} is an estimate of F based on the data y_1, \dots, y_n . This might be a parametric model such as the normal, with parameters estimated by maximum likelihood or a more robust method,

or the empirical distribution function (EDF) \hat{F} , which puts mass n^{-1} on each of the y_j . If partial information is available about F , it may be injected into \hat{F} . However \hat{F} is obtained, our estimate t is simply the result of applying the algorithm $t(\cdot)$ to \hat{F} .

Typical issues now to be addressed are: what are bias and variance estimates for t ? What is a reliable confidence interval for θ ? Is a certain hypothesis consistent with the data? Hypothesis tests raise the issue of how the null hypothesis should be imposed, and are discussed in detail in Chapter 4 of Davison and Hinkley (1997). Here we focus on confidence intervals, which are reviewed in DiCiccio and Efron (1996), Davison and Hinkley (1997, Chapter 5) and Carpenter and Bithell (2000).

Confidence Intervals

The simplest approach to confidence interval construction uses normal approximation to the distribution of T , the random variable of which t is the observed value. If the true bias and variance of T are

$$\begin{aligned} b(F) &= E(T | F) - \theta = E(T | F) - t(F), \\ v(F) &= \text{var}(T | F), \end{aligned} \quad (1)$$

then we might hope that in large samples

$$Z = \frac{T - \theta - b(F)}{v(F)^{1/2}} \sim N(0, 1);$$

the conditioning in (1) indicates that T is based on a random sample Y_1, \dots, Y_n from F . In this case an approximate $(1 - 2\alpha)$ confidence interval for θ is

$$t - b(F) - z_{1-\alpha} v(F)^{1/2}, \quad t - b(F) - z_\alpha v(F)^{1/2}, \quad (2)$$

where z_α is the α quantile of the standard normal distribution. The adequacy of (2) depends on F , n , and T and cannot be taken for granted.

As it stands (2) is useless, because it depends on the unknown F . A key idea, sometimes called the *bootstrap* or *plug-in principle*, is to replace the unknown F with its known estimate \hat{F} , giving bias and variance estimates $b(\hat{F})$ and $v(\hat{F})$. For all but the simplest estimators T these cannot be obtained analytically and so

simulation is used. We generate R independent bootstrap samples Y_1^*, \dots, Y_n^* by sampling independently from \hat{F} , compute the corresponding estimator random variables T_1^*, \dots, T_R^* , and then hope that

$$b(F) \doteq b(\hat{F}) = E(T | \hat{F}) - t(\hat{F}) \quad (3)$$

$$\doteq R^{-1} \sum_{r=1}^R T_r^* - t = \bar{T}^* - t, \quad (4)$$

$$v(F) \doteq v(\hat{F}) = \text{var}(T | \hat{F}) \quad (5)$$

$$\doteq \frac{1}{R-1} \sum_{r=1}^R (T_r^* - \bar{T}^*)^2. \quad (6)$$

There are two errors here: statistical error due to replacement of F by \hat{F} , and simulation error from replacement of expectation and variance by averages. Evidently we must choose R large enough to make the second of these errors small relative to the first, and if possible use $b(\hat{F})$ and $v(\hat{F})$ in such a way that the statistical error, unavoidable in most situations, is minimized. This means using approximate pivots where possible.

If the normal approximation leading to (2) fails because the distribution of $T - \theta$ is not close to normal, an alternative approach to setting confidence intervals may be based on $T - \theta$. The idea is that if $T^* - t$ and $T - \theta$ have roughly the same distribution, then quantiles of the second may be estimated by simulating those of the first, giving $(1 - 2\alpha)$ basic bootstrap confidence limits

$$t - (T_{((R+1)(1-\alpha))}^* - t), \quad t - (T_{((R+1)\alpha)}^* - t),$$

where $T_{(1)}^* < \dots < T_{(R)}^*$ are the sorted T_r^* 's. When an approximate variance V for T is available and can be calculated from Y_1, \dots, Y_n , studentized bootstrap confidence intervals may be based on $Z = (T - \theta)/V^{1/2}$, whose quantiles are estimated from simulated values of the corresponding bootstrap quantity $Z^* = (T^* - t)/V^{1/2}$. This is justified by Edgeworth expansion arguments valid for many but not all statistics (Hall, 1992).

Unlike the intervals mentioned above, *percentile* and *bias-corrected adjusted* (BCa) intervals have the attractive property of invariance to transformations of the parameters. The percentile intervals with level $(1 - 2\alpha)$ is $(T_{((R+1)\alpha)}^*, T_{((R+1)(1-\alpha))}^*)$, while the BCa interval has form $(T_{((R+1)\alpha')}^*, T_{((R+1)(1-\alpha''))}^*)$, with α' and α'' cleverly chosen to improve the properties of the interval. DiCiccio and Efron (1996) describe the reasoning underlying these intervals and their developments.

The BCa and studentized intervals are second-order accurate. Numerical comparisons suggest that both tend to undercover, so the true probability that a 0.95 interval contains the true parameter is smaller than 0.95, and

that BCa intervals are shorter than studentized ones, so they undercover by slightly more.

Bootstrapping in R

R (Ihaka and Gentleman, 1996) is a language and environment for statistical computing and graphics. Additional details can be found at www.r-project.org. The two main packages for bootstrapping in R are `boot` and `bootstrap`. Both are available on the ‘Comprehensive R Archive Network’ (CRAN, cran.r-project.org) and accompany Davison and Hinkley (1997) and Efron and Tibshirani (1993) respectively. The package `boot`, written by Angelo Canty for use within S-Plus, was ported to R by Brian Ripley and is much more comprehensive than any of the current alternatives, including methods that the others do not include. After downloading the package from CRAN and installing the package, one simply has to type

```
require(boot)
```

at the R prompt. Note that the installation could also be performed within R by means of

```
install.packages("boot")
```

A good starting point is to carefully read the documentations of the R functions `boot` and `boot.ci`

```
?boot
```

```
?boot.ci
```

and to try out one of the examples given in the ‘Examples’ section of the corresponding help file. In what follows we illustrate their use.

Example

Figure 1 shows data from an experiment in which two laser treatments were randomized to eyes on patients. The response is visual acuity, measured by the number of letters correctly identified in a standard eye test. Some patients had only one suitable eye, and they received one treatment allocated at random. There are 20 patients with paired data and 20 patients for whom just one observation is available, so we have a mixture of paired comparison and two-sample data.

```
blue <- c(4, 69, 87, 35, 39, 79, 31, 79, 65, 95, 68,
       62, 70, 80, 84, 79, 66, 75, 59, 77, 36, 86,
       39, 85, 74, 72, 69, 85, 85, 72)
red <- c(62, 80, 82, 83, 0, 81, 28, 69, 48, 90, 63,
       77, 0, 55, 83, 85, 54, 72, 58, 68, 88, 83, 78,
       30, 58, 45, 78, 64, 87, 65)
acui<-data.frame(str=c(rep(0,20),
                       rep(1,10)), red, blue)
```

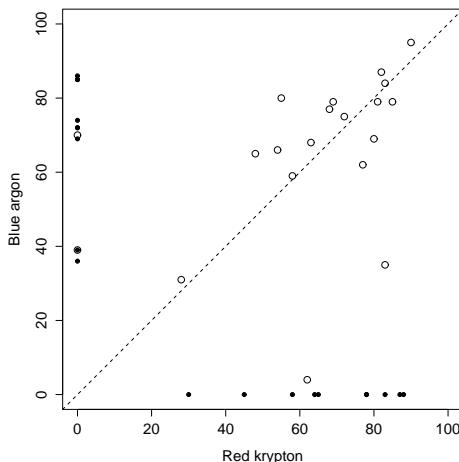


Figure 1: Paired (circles) and unpaired data (small blobs).

We denote the fully observed pairs $y_j = (r_j, b_j)$, the responses for the eyes treated with red and blue treatments, and for these n_d patients we let $d_j = b_j - r_j$. Individuals with just one observation give data $y_j = (?, b_j)$ or $y_j = (r_j, ?)$; there are n_b and n_r of these. The unknown variances of the d 's, r 's and b 's are σ_d^2 , σ_r^2 and σ_b^2 .

For illustration purposes, we will perform a standard analysis for each. First, we could only consider the paired data and construct the classical Student- t 0.95 confidence interval for the mean of the differences, of form $\bar{d} \pm t_{n-1}(0.025)s_d/n_d^{1/2}$, where $\bar{d} = 3.25$, s_d is the standard deviation of the d 's and $t_{n-1}(0.025)$ is the quantile of the appropriate t distribution. This can be done in R by means of

```
> acu.pd <- acui[acui$str==0,]
> dif <- acu.pd$blue-acu.pd$red
> n <- nrow(acu.pd)
> tmp<-qt(0.025,n-1)*sd(dif)/sqrt(n)
> c(mean(dif)+tmp, mean(dif)-tmp)
[1] -9.270335 15.770335
```

But a Q-Q plot of the differences looks more Cauchy than normal, so the usual model might be thought unreliable. The bootstrap can help to check this. To perform a nonparametric bootstrap in this case we first need to define the *bootstrap function*, corresponding to the algorithm $t(\cdot)$:

```
acu.pd.fun <- function(data, i){
  d <- data[i,]
  dif <- d$blue-d$red
  c(mean(dif), var(dif)/nrow(d)) }
```

A set of $R = 999$ bootstrap replicates can then be easily obtained with $acu.pd.b<-boot(acu.pd, acu.pd.fun, R=999)$. The result-

ing nonparametric 0.95 bootstrap confidence intervals can be calculated as shown previously or using directly

```
> boot.ci(acu.pd.b,
  type=c("norm", "basic", "stud"))
...
Normal          Basic          Studentized
(-8.20,14.95) (-8.10,15.05) (-8.66,15.77)
```

The normal Q-Q plot of the $R = 999$ replicates in the left panel of Figure 2 underlines the fact that the Student- t and the bootstrap intervals are essentially equal.

An alternative is to consider only the two-sample data and compare the means of the two populations issuing from the patients for whom just one observation is available, namely

```
acu.ts<- acui[acui$str==1,]
```

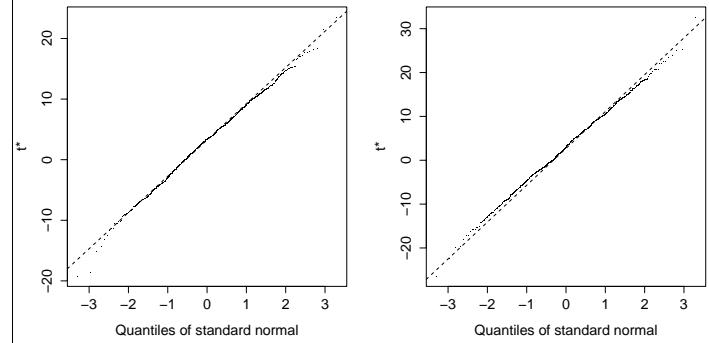


Figure 2: Normal Q-Q plots of bootstrap estimate t^* .

Left: for the paired analysis.
Right: for the two-sample analysis.

The classical normal 0.95 confidence interval for the difference of the means is $(\bar{b} - \bar{r}) \pm z_{0.025}(s_b^2/n_b + s_r^2/n_r)^{1/2}$, where s_b and s_r are the standard deviations of the b 's and r 's, and $z_{0.025}$ is the 0.025 quantile of the standard normal distribution.

```
> acu.ts <- acui[acui$str==1,]
> dif <- mean(acu.ts$blue)-mean(acu.ts$red)
> tmp <- qnorm(0.025)*
  sqrt(var(acu.ts$blue)/nrow(acu.ts) +
  var(acu.ts$red)/nrow(acu.ts))
> c(dif+tmp, dif-tmp)
[1] -13.76901 19.16901
```

The obvious estimator and its estimated variance are

$$t = \bar{b} - \bar{r}, \quad v = s_b^2/n_b + s_r^2/n_r,$$

whose values for these data are 2.7 and 70.6. To construct bootstrap confidence intervals we generate

$R = 999$ replicates of t and v , with each simulated dataset containing n_b values sampled with replacement from the bs and n_r values sampled with replacement from the rs . In R :

```
y<-c(acui$blue[21:30],acui$red[21:30])
acu<-data.frame(col=rep(c(1,2),c(10,10)),y)
acu.ts.f <- function(data, i){
d <- data[i,]
m <- mean(d$y[1:10])-mean(d$y[11:20])
v <- var(d$y[1:10])/10+var(d$y[11:20])/10
c(m, v) }
acu.ts.boot<-boot(acu,acu.ts.f,R=999,
strata=acu$col)
```

Here `strata=acu$col` ensures stratified simulation. The Q–Q plot of these 999 values in the right panel of Figure 2 is close to normal, and the bootstrap intervals computed using `boot.ci` differ little from the classical normal interval.

We now combine the analyses, hoping that the resulting confidence interval will be shorter. If the variances σ_d^2 , σ_r^2 and σ_b^2 of the ds , rs and bs were known, a minimum variance unbiased estimate of the difference between responses for blue and red treatments would be

$$\frac{n_d \bar{d} / \sigma_d^2 + (\bar{b} - \bar{r}) / (\sigma_b^2 / n_b + \sigma_r^2 / n_r)}{n_d / \sigma_d^2 + 1 / (\sigma_b^2 / n_b + \sigma_r^2 / n_r)}.$$

As σ_d^2 , σ_r^2 and σ_b^2 are unknown, we replace them by estimates, giving estimated treatment difference and its variance

$$t = \frac{n_d \bar{d} / \hat{\sigma}_d^2 + (\bar{b} - \bar{r}) / (\hat{\sigma}_b^2 / n_b + \hat{\sigma}_r^2 / n_r)}{n_d / \hat{\sigma}_d^2 + 1 / (\hat{\sigma}_b^2 / n_b + \hat{\sigma}_r^2 / n_r)},$$

$$v = \left\{ n_d / \hat{\sigma}_d^2 + 1 / (\hat{\sigma}_b^2 / n_b + \hat{\sigma}_r^2 / n_r) \right\}^{-1}.$$

Here $t = 3.07$ and $v = 4.873^2$, so a naive 0.95 confidence interval for the treatment difference is $(-6.48, 12.62)$.

One way to apply the bootstrap here is to generate a bootstrap dataset by taking n_d pairs randomly with replacement from \hat{F}_y , n_b values with replacement from \hat{F}_b and n_r values with replacement from \hat{F}_r , each resample being taken with equal probability:

```
acu.f <- function(data, i) {
d <- data[i,]
m <- sum(data$str)
if(length(unique((i)==(1:nrow(data))))!=1) {
d$blue[d$str==1]<-sample(d$blue,size=m,T)
d$red[d$str==1]<-sample(d$red,size=m,T) }
dif<- d$blue[d$str==0]-d$red[d$str==0]
d2 <- d$blue[d$str==1]
d3 <- d$red[d$str==1]
```

```
v1 <- var(dif)/length(dif)
v2 <- var(d2)/length(d2)+var(d3)/length(d3)
v <- 1/(1/v1+1/v2)
c((mean(dif)/v1+(mean(d2)-mean(d3))/v2)*v,v)
acu.b<-boot(acui,acu.f,R=999,strata=acui$str)
boot.ci(acu.b,type=c("norm","basic","stud",
"perc","bca"))
```

giving all five sets of confidence limits. The interested reader can continue the analysis.

Regression

A linear regression model has form $y_j = x_j^\top \beta + \varepsilon_j$, where the (y_j, x_j) are the response and the $p \times 1$ vector of covariates for the j th response y_j . We are usually interested in confidence intervals for the parameters, the choice of covariates, or prediction of the future response y_+ at a new covariate x_+ . The two basic resampling schemes for regression models are

- *resampling cases* $(y_1, x_1), \dots, (y_n, x_n)$, under which the bootstrap data are

$$(y_1, x_1)^*, \dots, (y_n, x_n)^*,$$

taken independently with equal probabilities n^{-1} from the (y_j, x_j) , and

- *resampling residuals*. Having obtained fitted values $x_j^\top \hat{\beta}$, we take ε_j^* randomly from centred standardized residuals e_1, \dots, e_n and set

$$y_j^* = x_j^\top \hat{\beta} + \varepsilon_j^*, \quad j = 1, \dots, n.$$

Under case resampling the resampled design matrix does not equal the original one. For moderately large data sets this doesn't matter, but it can be worth bearing in mind if n is small or if a few observations have a strong influence on some aspect of the design. If the wrong model is fitted and this scheme is used we get an appropriate measure of uncertainty, so case resampling is in this sense robust. The second scheme is more efficient than resampling pairs if the model is correct, but is not robust to getting the wrong model, so careful model-checking is needed before it can be used. Either scheme can be stratified if the data are inhomogeneous. In the most extreme form of stratification the strata consist of just one residual; this is the *wild bootstrap*, used in non-parametric regressions.

Variants of residual resampling needed for generalized linear models, survival data and so forth are all constructed essentially by looking for the exchangeable aspects of the model, estimating them, and then resampling them. Similar ideas also apply to time series models such as ARMA processes. Additional examples and further details can be found in Davison and Hinkley

(1997, Chapters 6–8). We now illustrate case and residual resampling.

The survival data (Efron, 1988) are survival percentages for rats at a succession of doses of radiation, with two or three replicates at each dose; see Figure 3. The data come with the package `boot` and can be loaded using

```
> data(survival)
```

To have a look at the data, simply type `survival` at the R prompt. The theoretical relationship between survival rate (`surv`) and dose (`dose`) is exponential, so linear regression applies to

$$x = \text{dose}, \quad y = \log(\text{surv}).$$

There is a clear outlier, case 13, at $x = 1410$. The least squares estimate of slope is -59×10^{-4} using all the data, changing to -78×10^{-4} with standard error 5.4×10^{-4} when case 13 is omitted.

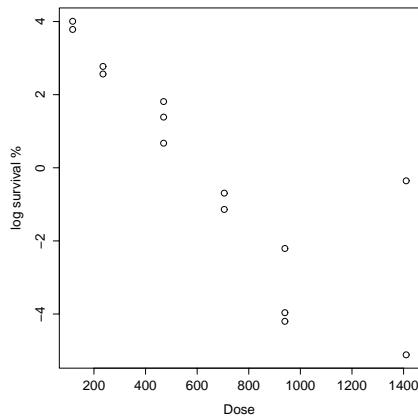


Figure 3: Scatter plot of survival data.

To illustrate the potential effect of an outlier in regression we resample cases, using

```
surv.fun <- function(data, i) {
  d <- data[i,]
  d.reg <- lm(log(d$surv) ~ d$dose)
  c(coef(d.reg)) }
surv.boot<-boot(survival,surv.fun,R=999)
```

The effect of the outlier on the resampled estimates is shown in Figure 4, a histogram of the $R = 999$ bootstrap least squares slopes $\hat{\beta}_1^*$. The two groups of bootstrapped slopes correspond to resamples in which case 13 does not occur and to samples where it occurs once or more. The resampling standard error of $\hat{\beta}_1^*$ is 15.6×10^{-4} , but only 7.8×10^{-4} for samples without case 13.

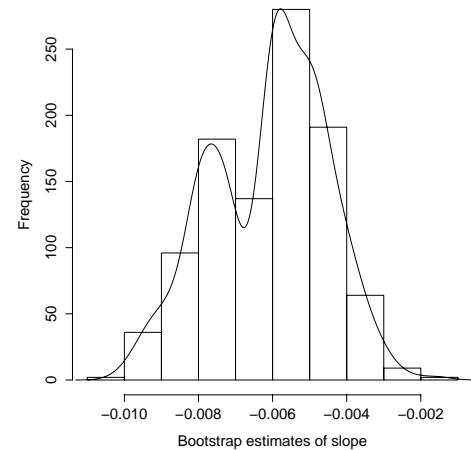


Figure 4: Histogram of bootstrap estimates of slope $\hat{\beta}_1^*$ with superposed kernel density estimate.

A jackknife-after-bootstrap plot (Efron, 1992; Davison and Hinkley, 1997, Section 3.10.1) shows the effect on $T^* - t$ of resampling from datasets from which each of the observations has been removed. Here we expect deletion of case 13 to have a strong effect, and Figure 5 obtained through

```
> jack.after.boot(surv.boot, index=2)
```

shows clearly that this case has an appreciable effect on the resampling distribution, and that its omission would give much tighter confidence limits on the slope.

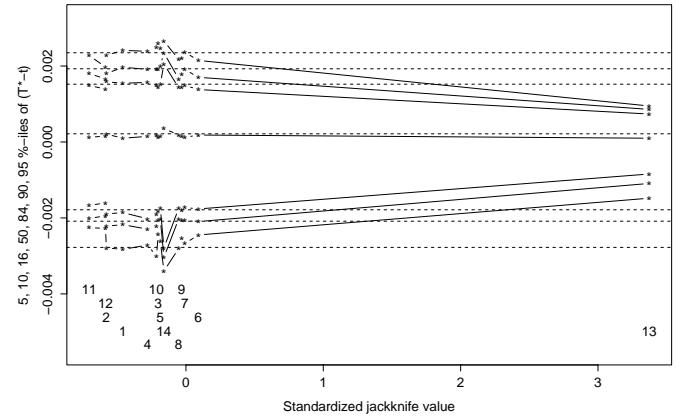


Figure 5: Jackknife-after-bootstrap plot for the slope. The vertical axis shows quantiles of $T^* - t$ for the full sample (horizontal dotted lines) and without each observation in turn, plotted against the influence value for that observation.

The effect of this outlier on the intercept and slope when resampling residuals can be assessed using

`sim=parametric` in the `boot` call. The required R code is:

```
fit <- lm(log(survival$surv) ~ survival$dose)
res <- resid(fit)
f <- fitted(fit)
surv.r.mle<- data.frame(f,res)
surv.r.fun<-function(data)
  coef(lm(log(data$surv) ~ data$dose))
surv.r.sim <- function(data, mle) {
  data$surv<-exp(mle$f+sample(mle$res,T))
  data
}
surv.r.boot<- boot(survival,surv.r.fun,
  R=999,sim="parametric",
  ran.gen=surv.r.sim,mle=surv.r.mle)
```

Having understood what this code does, the interested reader may use it to continue the analysis.

Discussion

Bootstrap resampling allows empirical assessment of standard approximations, and may indicate ways to fix them when they fail. The computer time involved is typically negligible — the resampling for this article took far less than the time needed to examine the data, devise plots and summary statistics, and to code (and check) the simulations.

Bootstrap methods offer considerable potential for modelling in complex problems, not least because they enable the choice of estimator to be separated from the assumptions under which its properties are to be assessed. In principle the estimator chosen should be appropriate to the model used, or there is a loss of efficiency. In practice, however, there is often some doubt about the exact error structure, and a well-chosen resampling scheme can give inferences robust to precise assumptions about the data.

Although the bootstrap is sometimes touted as a replacement for ‘traditional statistics’, we believe this to be misguided. It is unwise to use a powerful tool without understanding why it works, and the bootstrap rests on ‘traditional’ ideas, even if their implementation via simulation is not ‘traditional’. Populations, parameters, samples, sampling variation, pivots and confidence lim-

its are fundamental statistical notions, and it does no-one a service to brush them under the carpet. Indeed, it is harmful to pretend that mere computation can replace thought about central issues such as the structure of a problem, the type of answer required, the sampling design and data quality. Moreover, as with any simulation experiment, it is essential to monitor the output to ensure that no unanticipated complications have arisen and to check that the results make sense, and this entails understanding how the output will be used. Never forget: *the aim of computing is insight, not numbers; garbage in, garbage out.*

References

- Carpenter, J. and Bithell, J.** (2000). Bootstrap confidence intervals: when, which, what? A practical guide for medical statisticians. *Statistics in Medicine*, **19**, 1141–1164.
- Davison, A. C. and Hinkley, D. V.** (1997). *Bootstrap Methods and their Application*. Cambridge University Press.
(statwww.epfl.ch/davison/BMA/)
- DiCiccio, T. J. and Efron, B.** (1996). Bootstrap confidence intervals (with Discussion). *Statistical Science*, **11**, 189–228.
- Efron, B.** (1988). Computer-intensive methods in statistical regression. *SIAM Review*, **30**, 421–449.
- Efron, B.** (1992). Jackknife-after-bootstrap standard errors and influence functions (with Discussion). *Journal of the Royal Statistical Society series B*, **54**, 83–127.
- Efron, B. and Tibshirani, R. J.** (1993). *An Introduction to the Bootstrap*. New York: Chapman & Hall.
- Hall, P.** (1992). *The Bootstrap and Edgeworth Expansion*. New York: Springer-Verlag.
- Ihaka, R. and Gentleman, R.** (1996). R: a language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, **5**, 299–314.
- Shao, J. and Tu, D.** (1995). *The Jackknife and Bootstrap*. New York: Springer-Verlag.

SOFTWARE PACKAGES

GGobi

Deborah F. Swayne, AT&T Labs – Research

dfs@research.att.com

GGobi is a new interactive and dynamic software sys-

tem for data visualization, the result of a significant redesign of the older XGobi system (Swayne, Cook and Buja, 1992; Swayne, Cook and Buja, 1998), whose development spanned roughly the past decade. GGobi differs from XGobi in many ways, and it is those differences that explain best why we have undertaken this redesign.



Cross-Validation for Predictive Analytics Using R

[R blog](#) By Sergio Venturini May 3, 2016 Tags: cross-validation, data mining, models, predictive analytics No Comments

Introduction

Since ancient times, humankind has always avidly sought a way to predict the future. One of the most widely known examples of this kind of activity in the past is the [Oracle of Delphi](#), who dispensed previews of the future to her petitioners in the form of divine inspired prophecies¹. In the modern days, the desire to know the future is still of interest to many of us, even if my feeling is that the increasing rapidity of technology innovations we observe everyday has somewhat lessened this instinct: things that few years ago seemed futuristic are now available to the great mass (e.g. the World Wide Web).

Among the many areas of the human being where predictions are highly needed there is *business decision making*. The tools for formulating predictions about quantities of interest are commonly known as **predictive analytics**, which is itself an essential part of [data science](#). At the heart of any prediction there is always a model, which typically depends on some unknown *structural parameters* (e.g. the coefficients of a regression model) as well as on one or more *tuning parameters* (e.g. the number of basis functions in a smoothing spline or the degree of a polynomial). The former are commonly estimated using a sample of data, while the latter have to be chosen to guarantee that the model itself provides predictions which are accurate enough. Tuning parameters usually regulate the model complexity and hence are a key ingredient for any predictive task. In this blog entry we focus on the most common strategy for eliciting reasonable values for the tuning parameters, the cross-validation approach.

The Bias-Variance Dilemma

The reason why one should care about the choice of the tuning parameter values is because these are intimately linked with the accuracy of the predictions returned by the model. What an analyst typically wants is a model that is able to predict well samples that have not been used for estimating the structural parameters (the so called *training sample*). In other words, a predictive model is considered good when it is capable of predicting previously unseen samples with high accuracy. The accuracy of a model's predictions is usually gauged using a **loss function**. Popular choices for the loss functions are the mean-squared error for continuous outcomes, or the 0-1 loss for a categorical outcome².

At this point, it is important to distinguish between different prediction error concepts:

- the **training error**, which is the average loss over the **training sample**,
- the **test error**, the prediction error over an independent **test sample**.

The training error gets smaller as long as the predicted responses are close to the observed responses, and will get larger if for some of the observations, the predicted and observed responses differ substantially. The training error is calculated using the training sample used to fit the model. Clearly, we shouldn't care too much about the model's predictive accuracy on the training data. On the contrary, we would like to assess the model's ability to predict observations never seen during estimation. The test error provides a measure of this ability. In general, one should select the model corresponding to the lowest test error.

The R code below implements these idea via simulated data. In particular, I simulate 100 training sets each of size 50 from a polynomial regression model, and for each I fit a sequence of cubic spline models with degrees of freedom from 1 to 30.

```

1 # Generate the training and test samples
2 seed <- 1809
3 set.seed(seed)
4
5 gen_data <- function(n, beta, sigma_eps) {
6   eps <- rnorm(n, 0, sigma_eps)
7   x <- sort(runif(n, 0, 100))
8   X <- cbind(1, poly(x, degree = (length(beta) - 1), raw = TRUE))
9   y <- as.numeric(X %*% beta + eps)
10
11  return(data.frame(x = x, y = y))
12 }
13
14
15 # Fit the models
16 require(splines)
17
18 n_rep <- 100
19 n_df <- 30
20 df <- 1:n_df
21 beta <- c(5, -0.1, 0.004, -3e-05)
22 n_train <- 50
23 n_test <- 10000
24 sigma_eps <- 0.5
25
26 xy <- res <- list()
27 xy_test <- gen_data(n_test, beta, sigma_eps)
28 for (i in 1:n_rep) {
29   xy[[i]] <- gen_data(n_train, beta, sigma_eps)
30   x <- xy[[i]][, "x"]
31   y <- xy[[i]][, "y"]
32   res[[i]] <- apply(t(df), 2, function(degf) lm(y ~ ns(x, df = degf)))
33 }
```

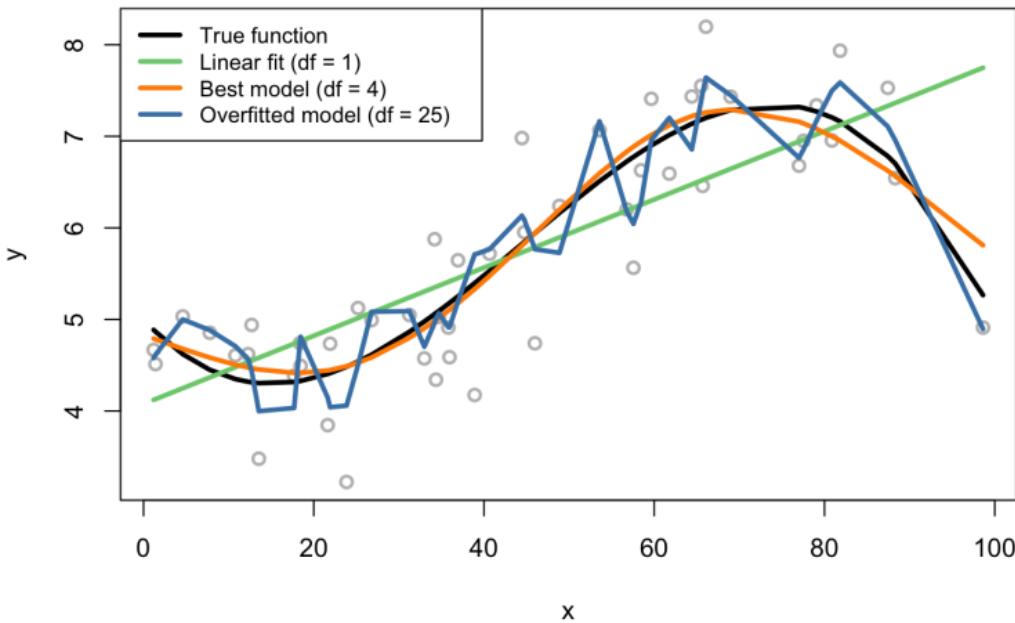
The next plot shows the first simulated training sample together with three fitted models corresponding to cubic splines with 1 (green line), 4 (orange line) and 25 (blue line) degrees of freedom respectively. These numbers have been chosen to show the full set of possibilities one may encounter in practice, i.e., either a

model with low variability but high bias (degrees of freedom = 1), or a model with high variability but low bias (degrees of freedom = 25), or a model which tries to find a compromise between bias and variance (degrees of freedom = 4).

```

1 # Plot the data
2 x <- xy[[1]]$x
3 X <- cbind(1, poly(x, degree = (length(beta) - 1), raw = TRUE))
4 y <- xy[[1]]$y
5 plot(y ~ x, col = "gray", lwd = 2)
6 lines(x, X %*% beta, lwd = 3, col = "black")
7 lines(x, fitted(res[[1]][[1]]), lwd = 3, col = "palegreen3")
8 lines(x, fitted(res[[1]][[4]]), lwd = 3, col = "darkorange")
9 lines(x, fitted(res[[1]][[25]]), lwd = 3, col = "steelblue")
10 legend(x = "topleft", legend = c("True function", "Linear fit (df = 1)", "Best model (df = 4)",
11         "Overfitted model (df = 25)"), lwd = rep(3, 4), col = c("black", "palegreen3",
12         "darkorange", "steelblue"), text.width = 32, cex = 0.85)
13

```



Then, for each training sample and fitted model, I compute the corresponding test error using a large test sample generated from the same (known!) population. These are represented in the following plot together with their *averages*, which are shown using thicker lines³. The solid points represent the three models illustrated in the previous diagram.

```

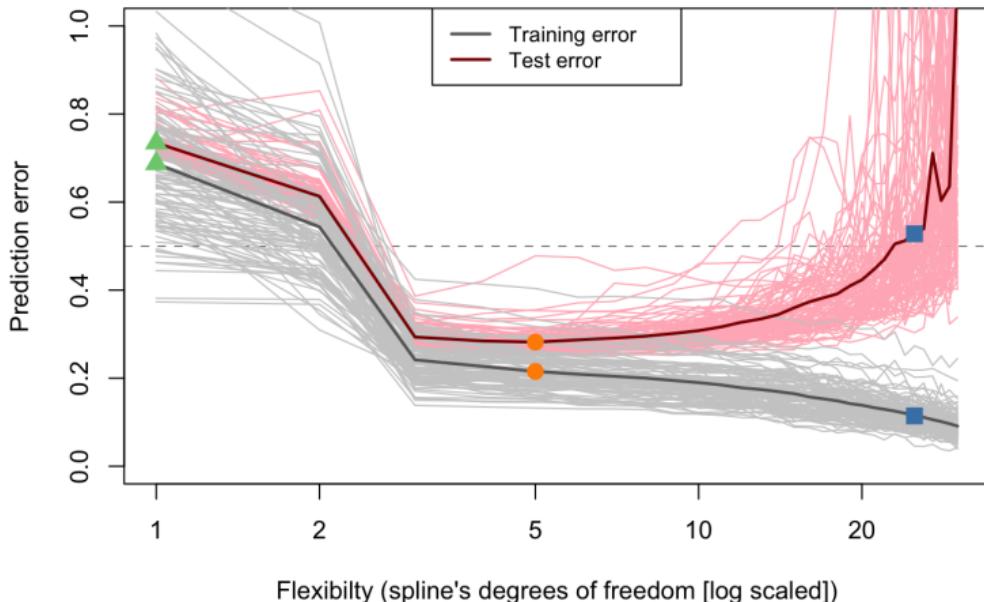
1 # Compute the training and test errors for each model
2 pred <- list()
3 mse <- te <- matrix(NA, nrow = n_df, ncol = n_rep)
4 for (i in 1:n_rep) {
5   msel[, i] <- sapply(res[[i]], function(obj) deviance(obj)/nobs(obj))
6   pred[[i]] <- mapply(function(obj, degf) predict(obj, data.frame(x = xy_test$x)),
7     res[[i]], df)
8   te[, i] <- sapply(as.list(data.frame(pred[[i]])), function(y_hat) mean((xy_test$y -
9     y_hat)^2))
10

```

```

11 }
12
13 # Compute the average training and test errors
14 av_mse <- rowMeans(mse)
15 av_te <- rowMeans(te)
16
17 # Plot the errors
18 plot(df, av_mse, type = "l", lwd = 2, col = gray(0.4), ylab = "Prediction error",
19       xlab = "Flexibilty (spline's degrees of freedom [log scaled])", ylim = c(0,
20       1), log = "x")
21 abline(h = sigma_eps, lty = 2, lwd = 0.5)
22 for (i in 1:n_rep) {
23   lines(df, te[, i], col = "lightpink")
24 }
25 for (i in 1:n_rep) {
26   lines(df, mse[, i], col = gray(0.8))
27 }
28 lines(df, av_mse, lwd = 2, col = gray(0.4))
29 lines(df, av_te, lwd = 2, col = "darkred")
30 points(df[1], av_mse[1], col = "palegreen3", pch = 17, cex = 1.5)
31 points(df[1], av_te[1], col = "palegreen3", pch = 17, cex = 1.5)
32 points(df[which.min(av_te)], av_mse[which.min(av_te)], col = "darkorange", pch = 16,
33       cex = 1.5)
34 points(df[which.min(av_te)], av_te[which.min(av_te)], col = "darkorange", pch = 16,
35       cex = 1.5)
36 points(df[25], av_mse[25], col = "steelblue", pch = 15, cex = 1.5)
37 points(df[25], av_te[25], col = "steelblue", pch = 15, cex = 1.5)
38 legend(x = "top", legend = c("Training error", "Test error"), lwd = rep(2, 2),
39       col = c(gray(0.4), "darkred"), text.width = 0.3, cex = 0.85)

```



One can see that the training errors decrease monotonically as the model gets more complicated (and less smooth). On the other side, even if the test error initially decreases, from a certain flexibility level on it starts increasing again. The change point occurs in correspondence of the orange model, that is, the model that provides a good compromise between bias and variance. The reason why the test error starts increasing for degrees of freedom larger than 3 or 4 is the so called **overfitting** problem. Overfitting is the tendency of a model to adapt too well to the training data, at the expense of generalization to previously unseen data.

points. In other words, an overfitted model fits the noise in the data rather than the actual underlying relationships among the variables. Overfitting usually occurs when a model is unnecessarily complex.

It is possible to show that the (expected) test error for a given observation in the test set can be decomposed into the sum of three components, namely

$$\text{Expected Test Error} = \text{Irreducible Noise} + (\text{Model Bias})^2 + \text{Model Variance}$$

which is known as the **bias-variance decomposition**. The first term is the data generating process variance. This term is unavoidable because we live in a noisy stochastic world, where even the best ideal model has non-zero error. The second term originates from the difficulty to catch the correct functional form of the relationship that links the dependent and independent variables (sometimes it is also called the *approximation bias*). The last term is due to the fact that we estimate our models using only a limited amount of data. Fortunately, this term gets closer and closer to zero as long as we collect more and more training data. Typically, the more complex (i.e., flexible) we make the model, the lower the bias but the higher the variance. This general phenomenon is known as the **bias-variance trade-off**, and the challenge is to find a model which provides a good compromise between these two issues.

Clearly, the situation illustrated above is only ideal, because in practice:

- We do not know the true model that generates the data. Indeed, our models are typically more or less mis-specified.
- We do only have a limited amount of data.

One way to overcome these hurdles and approximate the search for the optimal model is to use the cross-validation approach.

A Solution: Cross-Validation

In essence, all these ideas bring us to the conclusion that it is not advisable to compare the predictive accuracy of a set of models using the same observations used for estimating the models. Therefore, for assessing the models' predictive performance we should use an independent set of data (the test sample). Then, the model showing the lowest error on the test sample (i.e., the lowest test error) is identified as the best.

Unfortunately, in many cases it is not possible to draw a (possibly large) independent set of observations for testing the models' performance, because collecting data is typically an expensive activity. The immediate reaction to this statement is that we can solve this issue by *splitting* the available data in two sets, one of which will be used for training while the other is used for testing. The split is usually performed randomly to guarantee that the two parts have the same distribution⁴.

Even if data splitting provides an unbiased estimate of the test error, it is often quite noisy. A possible solution⁵ is to use **cross-validation** (CV). In its basic version, the so called k-fold cross-validation, the

samples are randomly partitioned into k sets (called *folds*) of roughly equal size. A model is fit using all the samples except the first subset. Then, the prediction error of the fitted model is calculated using the first held-out samples. The same operation is repeated for each fold and the model's performance is calculated by averaging the errors across the different test sets. k is usually fixed at 5 or 10. Cross-validation provides an estimate of the test error for each model⁶. Cross-validation is one of the most widely-used method for model selection, and for choosing tuning parameter values.

The code below illustrates k-fold cross-validation using the same simulated data as above but not pretending to know the data generating process. In particular, I generate 100 observations and choose k=10. Together with the training error curve, in the plot I report both the CV and test error curves. Additionally, I provide also the standard error bars, which are the standard errors of the individual prediction error for each of the k=10 parts.

```

1 set.seed(seed)
2
3 n_train <- 100
4 xy <- gen_data(n_train, beta, sigma_eps)
5 x <- xy$x
6 y <- xy$y
7
8 fitted_models <- apply(t(df), 2, function(degf) lm(y ~ ns(x, df = degf)))
9 mse <- sapply(fitted_models, function(obj) deviance(obj)/nobs(obj))
10
11 n_test <- 10000
12 xy_test <- gen_data(n_test, beta, sigma_eps)
13 pred <- mapply(function(obj, degf) predict(obj, data.frame(x = xy_test$x)),
14   fitted_models, df)
15 te <- sapply(as.list(data.frame(pred)), function(y_hat) mean((xy_test$y - y_hat)^2))
16
17 n_folds <- 10
18 folds_i <- sample(rep(1:n_folds, length.out = n_train))
19 cv_tmp <- matrix(NA, nrow = n_folds, ncol = length(df))
20 for (k in 1:n_folds) {
21   test_i <- which(folds_i == k)
22   train_xy <- xy[-test_i, ]
23   test_xy <- xy[test_i, ]
24   x <- train_xy$x
25   y <- train_xy$y
26   fitted_models <- apply(t(df), 2, function(degf) lm(y ~ ns(x, df = degf)))
27   x <- test_xy$x
28   y <- test_xy$y
29   pred <- mapply(function(obj, degf) predict(obj, data.frame(ns(x, df = degf))),
30     fitted_models, df)
31   cv_tmp[k, ] <- sapply(as.list(data.frame(pred)), function(y_hat) mean((y -
32     y_hat)^2))
33 }
34 cv <- colMeans(cv_tmp)
35
36 require(Hmisc)
37
38 plot(df, mse, type = "l", lwd = 2, col = gray(0.4), ylab = "Prediction error",
39   xlab = "Flexibilit (spline's degrees of freedom [log scaled])", main = paste0(n_folds,
40     "-fold Cross-Validation"), ylim = c(0.1, 0.8), log = "x")
41 lines(df, te, lwd = 2, col = "darkred", lty = 2)
42 cv_sd <- apply(cv_tmp, 2, sd)/sqrt(n_folds)
43 errbar(df, cv, cv + cv_sd, cv - cv_sd, add = TRUE, col = "steelblue2", pch = 19,
44   lwd = 0.5)
45 lines(df, cv, lwd = 2, col = "steelblue2")
46 points(df, cv, col = "steelblue2", pch = 19)
47

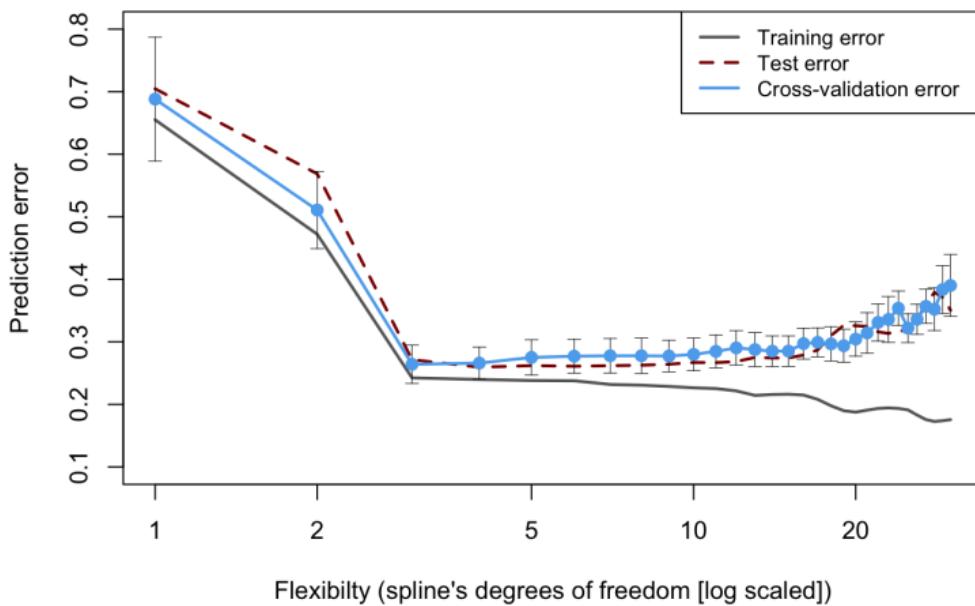
```

```

48 | legend(x = "topright", legend = c("Training error", "Test error", "Cross-validation error"),
49 |   lty = c(1, 2, 1), lwd = rep(2, 3), col = c(gray(0.4), "darkred", "steelblue2"),
50 |   text.width = 0.4, cex = 0.85)

```

10-fold Cross-Validation



Often a “one-standard error” rule is used with cross-validation, according to which one should choose the most parsimonious model whose error is no more than one standard error above the error of the best model. In the example above, the best model (that for which the CV error is minimized) uses 3 degrees of freedom, which also satisfies the requirement of the one-standard error rule.

The case where $k=n$ corresponds to the so called **leave-one-out cross-validation** (LOOCV) method. In this case the test set contains a single observation. The advantages of LOOCV are: 1) it doesn't require random numbers to select the observations to test, meaning that it doesn't produce different results when applied repeatedly, and 2) it has far less bias than k -fold CV because it employs larger training sets containing $n-1$ observations each. On the other side, LOOCV presents also some drawbacks: 1) it is potentially quite intense computationally, and 2) due to the fact that any two training sets share $n-2$ points, the models fit to those training sets tend to be strongly correlated with each other.

The code below implements LOOCV using the same example I discussed so far. The next plot shows that most of the times LOOCV does not provide dramatically different results with respect to CV.

```

1 | require(splines)
2 |
3 |
4 | loocv_tmp <- matrix(NA, nrow = n_train, ncol = length(df))
5 | for (k in 1:n_train) {
6 |   train_xy <- xy[-k, ]
7 |   test_xy <- xy[k, ]
8 |   x <- train_xy$x

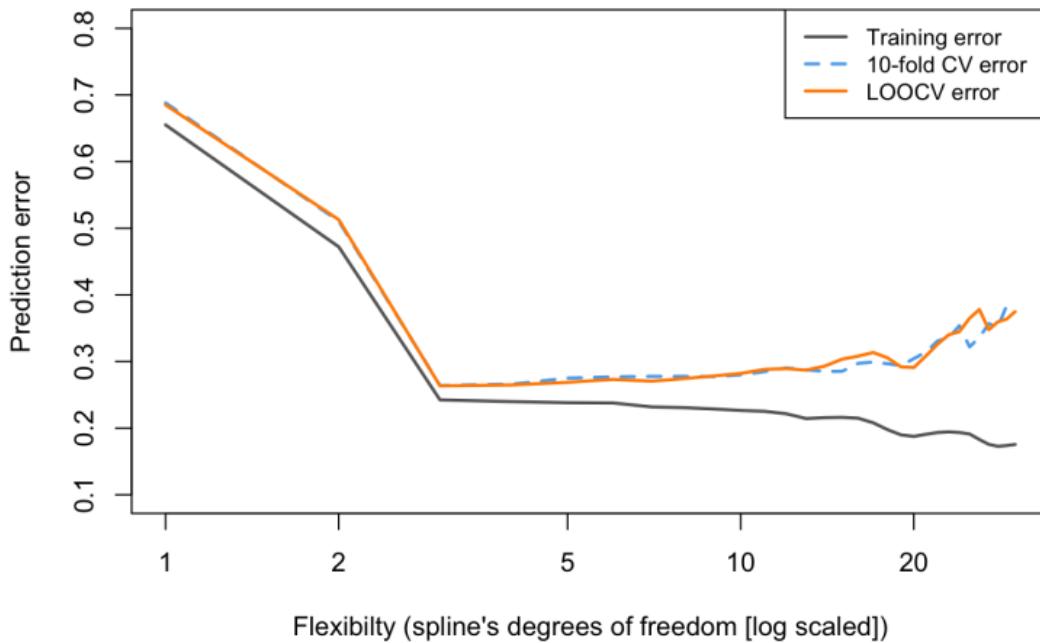
```

```

9  y <- train_xy$y
10 fitted_models <- apply(t(df), 2, function(degf) lm(y ~ ns(x, df = degf)))
11 pred <- mapply(function(obj, degf) predict(obj, data.frame(x = test_xy$x)),
12                  fitted_models, df)
13 loocv_tmp[k, ] <- (test_xy$y - pred)^2
14 }
15 loocv <- colMeans(loocv_tmp)
16
17 plot(df, mse, type = "l", lwd = 2, col = gray(.4), ylab = "Prediction error",
18       xlab = "Flexibility (spline's degrees of freedom [log scaled])",
19       main = "Leave-One-Out Cross-Validation", ylim = c(.1, .8), log = "x")
20 lines(df, cv, lwd = 2, col = "steelblue2", lty = 2)
21 lines(df, loocv, lwd = 2, col = "darkorange")
22 legend(x = "topright", legend = c("Training error", "10-fold CV error", "LOOCV error"),
23         lty = c(1, 2, 1), lwd = rep(2, 3), col = c(gray(.4), "steelblue2", "darkorange"),
24         text.width = .3, cex = .85)

```

Leave-One-Out Cross-Validation



Doing Cross-Validation With R: the caret Package

There are many R packages that provide functions for performing different flavors of CV. In my opinion, one of the best implementation of these ideas is available in the **caret** package by Max Kuhn (see Kuhn and Johnson 2013)⁷. The aim of the **caret** package (acronym of **c**lassification **a**nd **r**egression **t**raining) is to provide a very general and efficient suite of commands for building and assessing predictive models. It allows to compare the predictive accuracy of a multitude of models (currently more than 200), including the most recent ones from machine learning. The comparison of different models can be done using cross-validation as well as with other approaches. The package also provides many options for data pre-processing. It is not my aim to provide here a thorough presentation of all the package features. Rather, I will focus only on a handful of its functions, those that allow to perform CV. For more details on the other

package functions, you can inspect the package documentation and its [website](#). To illustrate these feature I will use some data for a credit scoring application whose data can be found [here](#).

Since credit scoring is a classification problem, I will use the number of misclassified observations as the loss measure. The data set contains information about 4,455 individuals for the following variables:

Variable	Description
<i>Status</i>	credit status
<i>Seniority</i>	job seniority (years)
<i>Home</i>	type of home ownership
<i>Time</i>	time of requested loan
<i>Age</i>	client's age
<i>Marital</i>	marital status
<i>Records</i>	existence of records
<i>Job</i>	type of job
<i>Expenses</i>	amount of expenses
<i>Income</i>	amount of income
<i>Assets</i>	amount of assets
<i>Debt</i>	amount of debt
<i>Amount</i>	amount requested of loan
<i>Price</i>	price of good

Here I use the “cleaned” version of the data set, where some pre-processing has already been performed (i.e., removal of few observations, imputation of missing values and categorization of continuous predictors). The tidy data are contained in the file CleanCreditScoring.csv.

```

1 require(RCurl)
2 require(prettyR)
3
4 url <- "https://raw.githubusercontent.com/gastonstat/CreditScoring/master/CleanCreditScoring.csv"
5

```

```
6 | cs_data <- getURL(url)
7 | cs_data <- read.csv(textConnection(cs_data))
8 | describe(cs_data)
```

```
1 |
2 | ## Description of cs_data
```

```
1 |
2 |
3 | ## Numeric
4 | ##      mean median     var      sd valid.n
5 | ## Seniority   7.99    5.00    66.85    8.18    4446
6 | ## Time        46.45   48.00   214.56   14.65    4446
7 | ## Age         37.08   36.00   120.70   10.99    4446
8 | ## Expenses    55.60   51.00   381.06   19.52    4446
9 | ## Income      140.63  124.00  6428.50  80.18    4446
10 | ## Assets      5354.95 3000.00 133040726.62 11534.33  4446
11 | ## Debt        342.26   0.00   1549264.52 1244.69  4446
12 | ## Amount      1038.76 1000.00 225385.62 474.75  4446
13 | ## Price        1462.48 1400.00 395081.60 628.56  4446
14 | ## Finrat       72.62   77.10   415.78   20.39  4446
15 | ## Savings      3.86    3.12    13.89    3.73  4446
16 |
17 | ## Factor
18 |
19 | ## Status      good    bad
20 | ## Count      3197.00 1249.00
21 | ## Percent     71.91  28.09
22 | ## Mode good
23 |
24 | ## Home        owner   rent parents other  priv ignore
25 | ## Count      2106.00 973.00 782.00 319.00 246.00 20.00
26 | ## Percent     47.37 21.88 17.59  7.17  5.53  0.45
27 | ## Mode owner
28 |
29 | ## Marital     married single separated widow divorced
30 | ## Count      3238.00 973.00 130.00 67.00  38.00
31 | ## Percent     72.83 21.88  2.92  1.51  0.85
32 | ## Mode married
33 |
34 | ## Records    no_rec yes_rec
35 | ## Count      3677.0  769.0
36 | ## Percent     82.7  17.3
37 | ## Mode no_rec
38 |
39 | ## Job         fixed freelance partime others
40 | ## Count      2803.00 1021.00 451.00 171.00
41 | ## Percent     63.05 22.96 10.14  3.85
42 | ## Mode fixed
43 |
44 | ## seniorityR sen (-1,1] sen (3,8] sen (14,99] sen (1,3] sen (8,14]
45 | ## Count      1042.00   978    880.00   789.00   757.00
46 | ## Percent     23.44    22     19.79    17.75    17.03
47 | ## Mode sen (-1,1]
48 |
49 | ## timeR      time (48,99] time (24,36] time (36,48] time (12,24] time (0,12]
50 | ## Count      1949.00   991.00  885.00   441.00   180.00
51 | ## Percent     43.84    22.29   19.91    9.92     4.05
52 | ## Mode time (48,99]
53 |
54 | ## ageR       age (30,40] age (40,50] age (25,30] age (0,25] age (50,99]
55 | ## Count      1415.00   900.00  781.00   699.00   651.00
56 | ## Percent     31.83    20.24   17.57    15.72    14.64
57 | ## Mode age (30,40]
58 |
59 | ## expensesR exp (0,40] exp (40,50] exp (50,60] exp (60,80] exp (80,1e+04]
60 | ## Count      1219.00  999.00  979.00   798.00   451.00
```

```

61 ## Percent      27.42      22.47      22.02      17.95      10.14
62 ## Mode exp (0,40]
63 ##
64 ## incomeR    inc (80,110] inc (140,190] inc (0,80] inc (110,140]
65 ## Count       954.00      915.00      886.00      866.00
66 ## Percent     21.46      20.58      19.93      19.48
67 ##
68 ## incomeR    inc (190,1e+04]
69 ## Count       825.00
70 ## Percent     18.56
71 ## Mode inc (80,110]
72 ##
73 ## assetsR    asset (-1,0] asset (3e+03,5e+03] asset (8e+03,1e+06]
74 ## Count       1626.00      937.00      719.00
75 ## Percent     36.57      21.08      16.17
76 ##
77 ## assetsR    asset (0,3e+03] asset (5e+03,8e+03]
78 ## Count       626.00       538.0
79 ## Percent     14.08      12.1
80 ## Mode asset (-1,0]
81 ##
82 ## debtR      debt (-1,0] debt (500,1.5e+03] debt (2.5e+03,1e+06] debt (0,500]
83 ## Count       3667.00      230.00      197.00      193.00
84 ## Percent     82.48       5.17       4.43       4.34
85 ##
86 ## debtR      debt (1.5e+03,2.5e+03]
87 ## Count       159.00
88 ## Percent     3.58
89 ## Mode debt (-1,0]
90 ##
91 ## amountR    am (900,1.1e+03] am (1.1e+03,1.4e+03] am (600,900] am (0,600]
92 ## Count       945.00       925.00      911.00      895.00
93 ## Percent     21.26      20.81      20.49      20.13
94 ##
95 ## amountR    am (1.4e+03,1e+05]
96 ## Count       770.00
97 ## Percent     17.32
98 ## Mode am (900,1.1e+03]
99 ##
100 ## priceR     priz (1.5e+03,1.8e+03] priz (1e+03,1.3e+03] priz (0,1e+03]
101 ## Count       1028.00      985.00      821.00
102 ## Percent     23.12      22.15      18.47
103 ##
104 ## priceR     priz (1.8e+03,1e+05] priz (1.3e+03,1.5e+03]
105 ## Count       811.00       801.00
106 ## Percent     18.24      18.02
107 ## Mode priz (1.5e+03,1.8e+03]
108 ##
109 ## finratR    finr (80,90] finr (90,100] finr (50,70] finr (70,80] finr (0,50]
110 ## Count       995.00       960.00      954.00      821.00      716.0
111 ## Percent     22.38       21.59      21.46      18.47      16.1
112 ## Mode finr (80,90]
113 ##
114 ## savingsR   sav (2,4] sav (0,2] sav (6,99] sav (4,6] sav (-99,0]
115 ## Count       1396.0      1111.00      827.0       814.00      298.0
116 ## Percent     31.4        24.99      18.6        18.31       6.7
117 ## Mode sav (2,4]

```

The caret package provides functions for splitting the data as well as functions that automatically do all the job for us, namely functions that create the resampled data sets, fit the models, and evaluate performance.

Among the functions for data splitting I just mention `createDataPartition()` and `createFolds()`. The former allows to create one or more test/training random partitions of the data, while the latter randomly splits the data into k subsets. In both functions the random sampling is done within the levels of y (when y is

categorical) to balance the class distributions within the splits. These functions return vectors of indexes that can then be used to subset the original sample into training and test sets.

```

1 require(caret)
2
3 classes <- cs_data[, "Status"]
4 predictors <- cs_data[, -match(c("Status", "Seniority", "Time", "Age", "Expenses",
5   "Income", "Assets", "Debt", "Amount", "Price", "Finrat", "Savings"), colnames(cs_data))]
6
7 train_set <- createDataPartition(classes, p = 0.8, list = FALSE)
8 str(train_set)
9

```

```

1
2 ##  int [1:3558, 1] 1 2 3 4 5 6 7 8 9 11 ...
3 ##  - attr(*, "dimnames")=List of 2
4 ##    ..$ : NULL
5 ##    ..$ : chr "Resample1"

```

```

1
2 train_predictors <- predictors[train_set, ]
3 train_classes <- classes[train_set]
4 test_predictors <- predictors[-train_set, ]
5 test_classes <- classes[-train_set]
6
7 set.seed(seed)
8 cv_splits <- createFolds(classes, k = 10, returnTrain = TRUE)
9 str(cv_splits)

```

```

1
2 ## List of 10
3 ## $ Fold01: int [1:4002] 1 2 3 4 5 6 7 8 9 10 ...
4 ## $ Fold02: int [1:4002] 2 3 4 5 6 7 8 9 10 11 ...
5 ## $ Fold03: int [1:4001] 1 2 3 4 5 6 7 8 9 10 ...
6 ## $ Fold04: int [1:4002] 1 2 3 4 5 6 7 8 9 10 ...
7 ## $ Fold05: int [1:4001] 1 2 3 4 7 8 10 11 12 13 ...
8 ## $ Fold06: int [1:4001] 1 2 3 4 5 6 7 8 9 10 ...
9 ## $ Fold07: int [1:4001] 1 2 4 5 6 7 9 10 11 12 ...
10 ## $ Fold08: int [1:4002] 1 2 3 4 5 6 7 8 9 10 ...
11 ## $ Fold09: int [1:4001] 1 2 3 5 6 8 9 11 12 13 ...
12 ## $ Fold10: int [1:4001] 1 3 4 5 6 7 8 9 10 11 ...

```

To automatically split the data, fit the models and assess the performance, one can use the `train()` function in the `caret` package. The code below shows an example of the `train()` function on the credit scoring data by modeling the outcome using all the predictors available with a penalized logistic regression. More specifically, I use the `glmnet` package (Friedman, Hastie, and Tibshirani 2008), that fits a generalized linear model via penalized maximum likelihood. The algorithm implemented in the package computes the regularization path for the *elastic-net* penalty over a grid of values for the *regularization parameter* λ . The tuning parameter λ controls the overall strength of the penalty. A second tuning parameter, called the *mixing percentage* and denoted with α , represents the elastic-net penalty (Zou and Hastie 2005). This parameter takes value in $[0,1]$ and bridges the gap between the *lasso* ($\alpha=1$) and the *ridge* ($\alpha=0$) approaches.

The `train()` function requires the model formula together with the indication of the model to fit and the grid of tuning parameter values to use. In the code below this grid is specified through the `tuneGrid` argument,

while `trControl` provides the method to use for choosing the optimal values of the tuning parameters (in our case, 10-fold cross-validation). Finally, the `preProcess` argument allows to apply a series of pre-processing operations on the predictors (in our case, centering and scaling the predictor values).

```
1 | require(glmnet)
```

```
1 | ## Warning: package 'Matrix' was built under R version 3.2.5
```

```
1 | set.seed(seed)
2 |
3 |
4 | cs_data_train <- cs_data[train_set, ]
5 | cs_data_test <- cs_data[-train_set, ]
6 |
7 | glmnet_grid <- expand.grid(alpha = c(0, .1, .2, .4, .6, .8, 1),
8 |                             lambda = seq(.01, .2, length = 20))
9 | glmnet_ctrl <- trainControl(method = "cv", number = 10)
10 | glmnet_fit <- train(Status ~ ., data = cs_data_train,
11 |                      method = "glmnet",
12 |                      preProcess = c("center", "scale"),
13 |                      tuneGrid = glmnet_grid,
14 |                      trControl = glmnet_ctrl)
15 | glmnet_fit
```

```
1 | ## glmnet
2 | ##
3 | ## 3558 samples
4 | ## 26 predictor
5 | ## 2 classes: 'bad', 'good'
6 | ##
7 | ## Pre-processing: centered (68), scaled (68)
8 | ## Resampling: Cross-Validated (10 fold)
9 | ## Summary of sample sizes: 3202, 3203, 3202, 3203, 3202, 3202, ...
10 | ## Resampling results across tuning parameters:
11 | ##
12 | ##
13 | ##   alpha    lambda  Accuracy   Kappa
14 | ##   0.0      0.01    0.8021427  0.4613907413
15 | ##   0.0      0.02    0.7998916  0.4520486081
16 | ##   0.0      0.03    0.7976412  0.4402614685
17 | ##   0.0      0.04    0.7987633  0.4407093800
18 | ##   0.0      0.05    0.7982015  0.4355350784
19 | ##   0.0      0.06    0.7979182  0.4313111542
20 | ##   0.0      0.07    0.7953893  0.4205306747
21 | ##   0.0      0.08    0.7931413  0.4105376360
22 | ##   0.0      0.09    0.7922978  0.4050557210
23 | ##   0.0      0.10    0.7892072  0.3920192662
24 | ##   0.0      0.11    0.7841454  0.3722554927
25 | ##   0.0      0.12    0.7824600  0.3640031420
26 | ##   0.0      0.13    0.7807739  0.3557226473
27 | ##   0.0      0.14    0.7793694  0.3482341774
28 | ##   0.0      0.15    0.7807746  0.3491274474
29 | ##   0.0      0.16    0.7810571  0.3472621824
30 | ##   0.0      0.17    0.7796511  0.3411817028
31 | ##   0.0      0.18    0.7796526  0.3373484610
32 | ##   0.0      0.19    0.7807746  0.3374411775
33 | ##   0.0      0.20    0.7785267  0.3288713594
34 | ##   0.1      0.01    0.8015794  0.4596697325
35 | ##   0.1      0.02    0.8010160  0.4506840219
36 | ##   0.1      0.03    0.7987672  0.4392696923
```

```

37 ## 0.1 0.04 0.7962367 0.4270137579
38 ## 0.1 0.05 0.7953909 0.4179208158
39 ## 0.1 0.06 0.7922986 0.4039787586
40 ## 0.1 0.07 0.7892056 0.3879423318
41 ## 0.1 0.08 0.7880804 0.3782808312
42 ## 0.1 0.09 0.7841454 0.3596294655
43 ## 0.1 0.10 0.7807723 0.3432057135
44 ## 0.1 0.11 0.7748718 0.3176779656
45 ## 0.1 0.12 0.7726223 0.3058039618
46 ## 0.1 0.13 0.7706544 0.2940236563
47 ## 0.1 0.14 0.7672804 0.2743480275
48 ## 0.1 0.15 0.7641905 0.2598076390
49 ## 0.1 0.16 0.7613815 0.2443603691
50 ## 0.1 0.17 0.7585718 0.2288614463
51 ## 0.1 0.18 0.7549153 0.2107358484
52 ## 0.1 0.19 0.7518231 0.1940038500
53 ## 0.1 0.20 0.7495751 0.1809109004
54 ## 0.2 0.01 0.8021396 0.4602812440
55 ## 0.2 0.02 0.7982070 0.4415693991
56 ## 0.2 0.03 0.7951147 0.4239150748
57 ## 0.2 0.04 0.7917424 0.4066236599
58 ## 0.2 0.05 0.7911742 0.3954568138
59 ## 0.2 0.06 0.7875218 0.3766126319
60 ## 0.2 0.07 0.7813388 0.3485492065
61 ## 0.2 0.08 0.7765588 0.3236774842
62 ## 0.2 0.09 0.7723398 0.3024505090
63 ## 0.2 0.10 0.7706583 0.2883765872
64 ## 0.2 0.11 0.7664425 0.2668721852
65 ## 0.2 0.12 0.7627868 0.2474032680
66 ## 0.2 0.13 0.7554779 0.2144108884
67 ## 0.2 0.14 0.7509796 0.1882593847
68 ## 0.2 0.15 0.7445165 0.1570413702
69 ## 0.2 0.16 0.7419869 0.1374198064
70 ## 0.2 0.17 0.7372108 0.1088243186
71 ## 0.2 0.18 0.7346811 0.0884942046
72 ## 0.2 0.19 0.7304645 0.0666304068
73 ## 0.2 0.20 0.7293401 0.0587304848
74 ## 0.4 0.01 0.7998916 0.4500521540
75 ## 0.4 0.02 0.7951147 0.4262079087
76 ## 0.4 0.03 0.7894920 0.3986179726
77 ## 0.4 0.04 0.7833091 0.3669902064
78 ## 0.4 0.05 0.7793694 0.3406658720
79 ## 0.4 0.06 0.7765596 0.3202875649
80 ## 0.4 0.07 0.7726246 0.2966774741
81 ## 0.4 0.08 0.7639128 0.2564191601
82 ## 0.4 0.09 0.7546352 0.2079469868
83 ## 0.4 0.10 0.7456401 0.1554338724
84 ## 0.4 0.11 0.7369299 0.1037171598
85 ## 0.4 0.12 0.7324316 0.0745712422
86 ## 0.4 0.13 0.7296218 0.0585154515
87 ## 0.4 0.14 0.7282173 0.0515741506
88 ## 0.4 0.15 0.7270937 0.0460444276
89 ## 0.4 0.16 0.7237213 0.0275193624
90 ## 0.4 0.17 0.7203474 0.0079998432
91 ## 0.4 0.18 0.7189429 0.0000000000
92 ## 0.4 0.19 0.7189429 0.0000000000
93 ## 0.4 0.20 0.7189429 0.0000000000
94 ## 0.6 0.01 0.7998940 0.4479646834
95 ## 0.6 0.02 0.7900538 0.4056859000
96 ## 0.6 0.03 0.7830274 0.3693670539
97 ## 0.6 0.04 0.7765572 0.3347009487
98 ## 0.6 0.05 0.7757161 0.3177186986
99 ## 0.6 0.06 0.7655990 0.2676239096
100 ## 0.6 0.07 0.7549153 0.2082278135
101 ## 0.6 0.08 0.7422693 0.1330653578
102 ## 0.6 0.09 0.7332758 0.0786721840
103 ## 0.6 0.10 0.7296218 0.0585273347
104 ## 0.6 0.11 0.7279364 0.0501825990
105 ## 0.6 0.12 0.7223176 0.0195861846
106 ## 0.6 0.13 0.7189429 0.0000000000

```

```

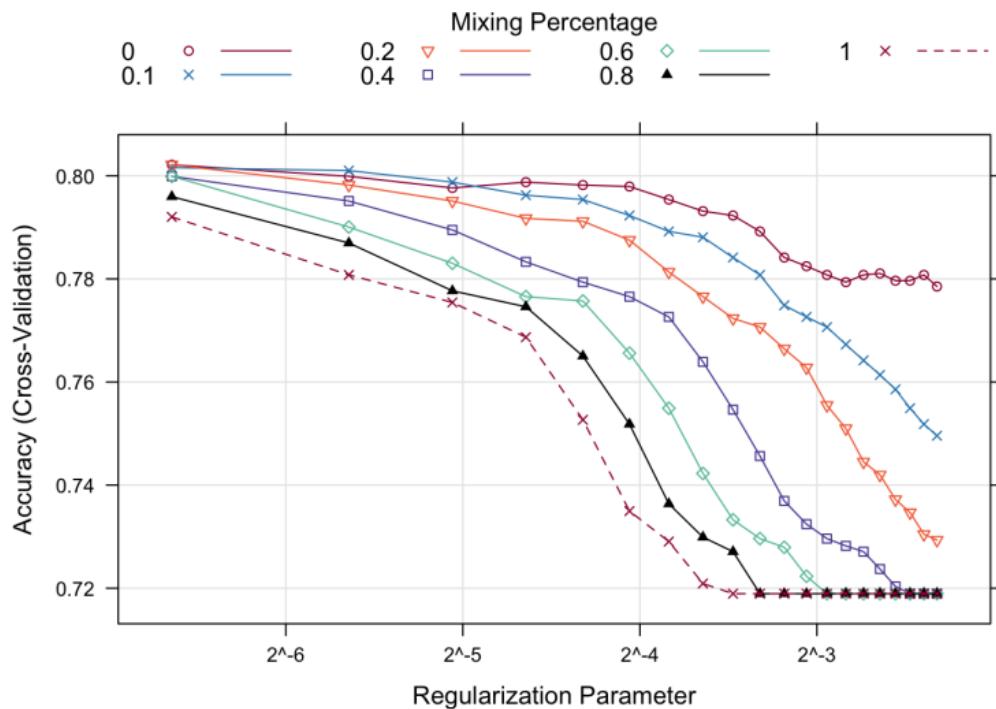
107 ## 0.6 0.14 0.7189429 0.0000000000
108 ## 0.6 0.15 0.7189429 0.0000000000
109 ## 0.6 0.16 0.7189429 0.0000000000
110 ## 0.6 0.17 0.7189429 0.0000000000
111 ## 0.6 0.18 0.7189429 0.0000000000
112 ## 0.6 0.19 0.7189429 0.0000000000
113 ## 0.6 0.20 0.7189429 0.0000000000
114 ## 0.8 0.01 0.7959582 0.4342802453
115 ## 0.8 0.02 0.7869623 0.3901784671
116 ## 0.8 0.03 0.7776832 0.3436570802
117 ## 0.8 0.04 0.7745925 0.3177549651
118 ## 0.8 0.05 0.7650348 0.2651758625
119 ## 0.8 0.06 0.7518270 0.1887158977
120 ## 0.8 0.07 0.7363681 0.0981410803
121 ## 0.8 0.08 0.7299027 0.0607058454
122 ## 0.8 0.09 0.7270937 0.0452185157
123 ## 0.8 0.10 0.7189429 0.0008725808
124 ## 0.8 0.11 0.7189429 0.0000000000
125 ## 0.8 0.12 0.7189429 0.0000000000
126 ## 0.8 0.13 0.7189429 0.0000000000
127 ## 0.8 0.14 0.7189429 0.0000000000
128 ## 0.8 0.15 0.7189429 0.0000000000
129 ## 0.8 0.16 0.7189429 0.0000000000
130 ## 0.8 0.17 0.7189429 0.0000000000
131 ## 0.8 0.18 0.7189429 0.0000000000
132 ## 0.8 0.19 0.7189429 0.0000000000
133 ## 0.8 0.20 0.7189429 0.0000000000
134 ## 1.0 0.01 0.7920241 0.4209892004
135 ## 1.0 0.02 0.7807794 0.3655088353
136 ## 1.0 0.03 0.7754360 0.3289251638
137 ## 1.0 0.04 0.7686897 0.2864096482
138 ## 1.0 0.05 0.7526713 0.1943914570
139 ## 1.0 0.06 0.7349620 0.0898775655
140 ## 1.0 0.07 0.7290600 0.0557404582
141 ## 1.0 0.08 0.7209100 0.0108497047
142 ## 1.0 0.09 0.7189429 0.0000000000
143 ## 1.0 0.10 0.7189429 0.0000000000
144 ## 1.0 0.11 0.7189429 0.0000000000
145 ## 1.0 0.12 0.7189429 0.0000000000
146 ## 1.0 0.13 0.7189429 0.0000000000
147 ## 1.0 0.14 0.7189429 0.0000000000
148 ## 1.0 0.15 0.7189429 0.0000000000
149 ## 1.0 0.16 0.7189429 0.0000000000
150 ## 1.0 0.17 0.7189429 0.0000000000
151 ## 1.0 0.18 0.7189429 0.0000000000
152 ## 1.0 0.19 0.7189429 0.0000000000
153 ## 1.0 0.20 0.7189429 0.0000000000
154 ##
155 ## Accuracy was used to select the optimal model using the largest value.
156 ## The final values used for the model were alpha = 0 and lambda = 0.01.

```

```

1 trellis.par.set(caretTheme())
2 plot(glmnet_fit, scales = list(x = list(log = 2)))

```



The previous plot shows the “accuracy”, that is the percentage of correctly classified observations, for the penalized logistic regression model with each combination of the two tuning parameters α and λ . The optimal tuning parameter values are $\alpha=0$ and $\lambda=0.01$.

Then, it is possible to predict new samples with the identified optimal model using the predict method:

```
1 pred_classes <- predict(glmnet_fit, newdata = cs_data_test)
2 table(pred_classes)
```

```
1
2 ## pred_classes
3 ## bad good
4 ## 172 716
```

```
1
2 pred_probs <- predict(glmnet_fit, newdata = cs_data_test, type = "prob")
3 head(pred_probs)
```

```
1
2 ##      bad      good
3 ## 1 0.07142151 0.9285785
4 ## 2 0.04231067 0.9576893
5 ## 3 0.03736701 0.9626330
6 ## 4 0.14796622 0.8520338
7 ## 5 0.12416939 0.8758306
8 ## 6 0.42359516 0.5764048
```

If you need to deepen your knowledge of predictive analytics, you may find something interesting in the R Course [Data Mining with R](#).

Stay tuned for the next article on the [MilanoR blog!](#)

References

- Efron, B., and R. Tibshirani. 1993. *An Introduction to the Bootstrap*. CRC Press.
- Friedman, J., T. Hastie, and R. Tibshirani. 2008. "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software* 33 (1): 1–22.
- Hastie, T., R. Tibshirani, and J. Friedman. 2009. *The Elements of Statistical Learning*. 2nd ed. Springer.
- James, G., D. Witten, T. Hastie, and R. Tibshirani. 2013. *An Introduction to Statistical Learning*. Springer.
- Kuhn, M., and K. Johnson. 2013. *Applied Predictive Modeling*. Springer.
- Zou, H., and T. Hastie. 2005. "Regularization and Variable Selection via the Elastic Net." *Journal of the Royal Statistical Association B* 67 (2): 301–20.

-
1. By the way, it seems that the oracular powers appeared to be associated with hallucinogenic gases that puffed out from the temple floor. [🔗](#)
 2. You can find a thorough formal illustration of all these concepts in Hastie, Tibshirani, and Friedman (2009), Chapter 7. A somewhat simpler presentation can be found in James et al. (2013). [🔗](#)
 3. More precisely, the light red curves correspond to what is called **conditional test error**, which means that each curve is conditional on the corresponding training sample. The heavier red curve correspond to the **expected test error**. In general, we would like to focus on the conditional test error for the particular training sample we have. However, this curve is very difficult to be estimated and in practice the expected test error is typically targeted. As we will see, cross-validation is a method for estimating the expected test error. For more details see Hastie, Tibshirani, and Friedman (2009). [🔗](#)
 4. A variant of the purely random split is to use stratified random sampling in order to create subsets that are balanced with respect to the outcome. This is useful in particular in classification problems when one class has a disproportionately small frequency compared to the others. [🔗](#)
 5. An alternative approach for the same objective is the **bootstrap**, that won't be illustrated here (see Efron and Tibshirani (1993)). [🔗](#)
 6. More precisely, cross-validation provides an estimate of the *expected* test error. [🔗](#)
 7. The boot package contains also a nice function called cv.glm, which implements k-fold cross-validation for generalized linear models. [🔗](#)

Related Post

[Performing Principal Components Regression \(PCR\) i...](#)

This article was originally posted on Quantide blog - see here. Principal components regression (PCR) is a regression technique based on principal ...

[Divide or Mix. Flexible Approaches to Data Analysi...](#)

A very interesting paradigm in data analysis comes from the necessity to model data where it is difficult to think of a single global function to be c...

[My first date with R - Free live class in Milano](#)

R Users!! Are you ready for an awesome news?! Microsoft, in collaboration with Quantide, is offering a one-day live course in Milano....

Share: [f](#) [t](#) [p](#) [g+](#)



Sergio Venturini

Lecturer of Statistics at Bocconi University. SDA Professor of Quantitative Methods at Bocconi School of Management. R enthusiast.

Leave a Reply

You must be logged in to post a comment.

Sponsored by



Upcoming Events

May 2016						
S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Powered by Eventbrite

Our free web books



Rabbit - Introduction to R

Ramarro - R for Developers

We are part of



Join the community

New user? → [Join now](#)

Close friend? → [Login](#)

R courses



Data Manipulation with R

Statistical Models with R

R for Beginners

Data Mining with R**R for Developer****Our meetings**

1st MilanoR Meeting

Top Authors

MilanoR (55 Posts)



Quantide srl (16 Posts)



Nicola Sturaro Sommacal

(13 Posts)



Enrico Tonini (7 Posts)



Michele Usuelli (6 Posts)



Andrea Spanò (5 Posts)



Michy Alice (4 Posts)



Andrea Pedretti (4 Posts)



Mec Analytics (3 Posts)



Max Marchi (3 Posts)

Tags

Agenda Algorithm

Announcements Big

Data bioR book COURSES

data.table data import data

management data

manipulation data mining

dplyr GenABEL ggmap ggplot

ggplot2 graphics Hadoop

MapReduce maps methods MilanoR

MilanoR Meeting MilanoR

meetings models My first... Next

Courses Past Courses Past

Meetings plot programming

qplot quantide R blog

Revolution Analytics r for beginners

rnr2 rworldmap s4 Sales dashboard shiny

tidyR tutorial useR2015

[R-project](#)[R-project: R manuals](#)[The R Journal](#)

[Task Views](#)[Quantide: R Web Books](#)[R Bloggers](#)

[Crantastic](#)

Sponsored by [Quantide](#) | Powered by [WordPress](#)



Differences-in-Differences (using R)

(work in progress)

Oscar Torres-Reyna

otorres@princeton.edu



August 2015

<http://dss.princeton.edu/training/> ®

Difference in differences (DID)

Estimation step-by-step

```
# Getting sample data.

library(foreign)
mydata = read.dta("http://dss.princeton.edu/training/Panel101.dta")

# Create a dummy variable to indicate the time when the treatment started. Lets
# assume that treatment started in 1994. In this case, years before 1994 will have a
# value of 0 and 1994+ a 1. If you already have this skip this step.

mydata$time = ifelse(mydata$year >= 1994, 1, 0)

# Create a dummy variable to identify the group exposed to the treatment. In this
# example lets assumed that countries with code 5,6, and 7 were treated (=1).
# Countries 1-4 were not treated (=0). If you already have this skip this step.

mydata$treated = ifelse(mydata$country == "E" |
                        mydata$country == "F" |
                        mydata$country == "G", 1, 0)

# Create an interaction between time and treated. We will call this interaction
# 'did'.

mydata$did = mydata$time * mydata$treated
```

Difference in differences (DID)

Estimation step-by-step

```
# Estimating the DID estimator
```

```
didreg = lm(y ~ treated + time + did, data = mydata)
summary(didreg)

Call:
lm(formula = y ~ treated + time + did, data = mydata)

Residuals:
    Min          1Q      Median          3Q          Max  
-9.768e+09 -1.623e+09  1.167e+08  1.393e+09  6.807e+09 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 3.581e+08  7.382e+08   0.485   0.6292    
treated     1.776e+09  1.128e+09   1.575   0.1200    
time        2.289e+09  9.530e+08   2.402   0.0191 *    
did         -2.520e+09  1.456e+09  -1.731   0.0882 .    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.953e+09 on 66 degrees of freedom
Multiple R-squared:  0.08273, Adjusted R-squared:  0.04104 
F-statistic: 1.984 on 3 and 66 DF,  p-value: 0.1249
```

```
# The coefficient for 'did' is the differences-in-differences
estimator. The effect is significant at 10% with the treatment having
a negative effect.
```

Difference in differences (DID)

Estimation step-by-step

```
# Estimating the DID estimator (using the multiplication method, no  
need to generate the interaction)
```

```
didreg1 = lm(y ~ treated*time, data = mydata)  
summary(didreg1)
```

Call:

```
lm(formula = y ~ treated * time, data = mydata)
```

Residuals:

Min	1Q	Median	3Q	Max
-9.768e+09	-1.623e+09	1.167e+08	1.393e+09	6.807e+09

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	3.581e+08	7.382e+08	0.485	0.6292
treated	1.776e+09	1.128e+09	1.575	0.1200
time	2.289e+09	9.530e+08	2.402	0.0191 *
treated:time	-2.520e+09	1.456e+09	-1.731	0.0882 .

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.953e+09 on 66 degrees of freedom

Multiple R-squared: 0.08273, Adjusted R-squared: 0.04104

F-statistic: 1.984 on 3 and 66 DF, p-value: 0.1249

```
# The coefficient for 'treated#time' is the differences-in-  
differences estimator ('did' in the previous example). The effect is  
significant at 10% with the treatment having a negative effect.
```

References

Introduction to econometrics, James H. Stock, Mark W. Watson. 2nd ed., Boston: Pearson Addison Wesley, 2007.

“Difference-in-Differences Estimation”, Imbens/Wooldridge, Lecture Notes 10, summer 2007.

http://www.nber.org/WNE/lect_10_diffindiffs.pdf

“Lecture 3: Differences-in-Differences”, Fabian Waldinger

http://www2.warwick.ac.uk/fac/soc/economics/staff/ffwaldinger/teaching/ec9a8/slides/lecture_3_-_did.pdf



Getting Started in Fixed/Random Effects Models using R

(ver. 0.1-*Draft*)

Oscar Torres-Reyna

Data Consultant
otorres@princeton.edu



Fall 2010



Panel data (also known as longitudinal or cross-sectional time-series data) is a dataset in which the behavior of entities are observed across time.

These entities could be states, companies, individuals, countries, etc.

Panel data looks like this



country	year	Y	X1	X2	X3
1	2000	6.0	7.8	5.8	1.3
1	2001	4.6	0.6	7.9	7.8
1	2002	9.4	2.1	5.4	1.1
2	2000	9.1	1.3	6.7	4.1
2	2001	8.3	0.9	6.6	5.0
2	2002	0.6	9.8	0.4	7.2
3	2000	9.1	0.2	2.6	6.4
3	2001	4.8	5.9	3.2	6.4
3	2002	9.1	5.2	6.9	2.1

For a brief introduction on the theory behind panel data analysis please see the following document:
<http://dss.princeton.edu/training/Panel101.pdf>

The contents of this document rely heavily on the document:
“Panel Data Econometrics in R: the plm package” <http://cran.r-project.org/web/packages/plm/vignettes/plm.pdf> and notes from the ICPSR’s Summer Program in Quantitative Methods of Social Research (summer 2010)

Exploring panel data

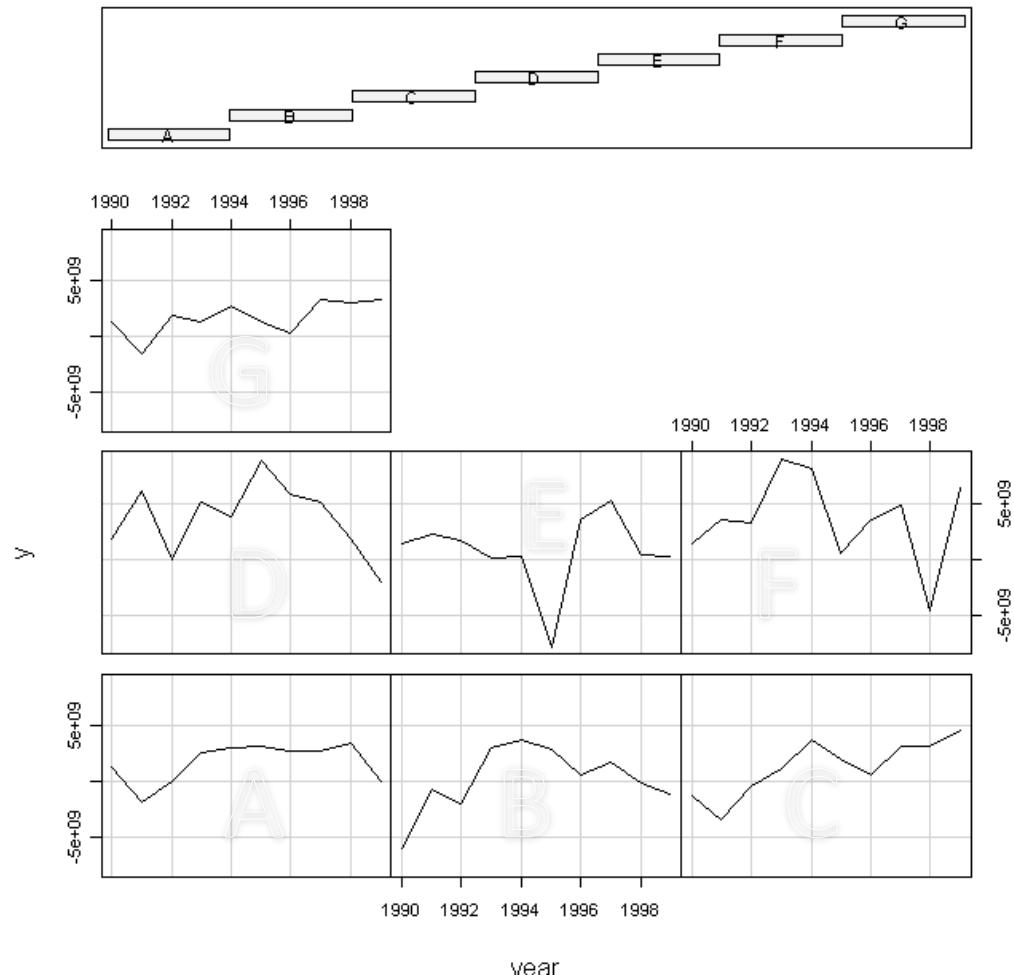
```
library(foreign)
Panel <- read.dta("http://dss.princeton.edu/training/Panel101.dta")
coplot(y ~ year|country, type="l", data=Panel)      # Lines
coplot(y ~ year|country, type="b", data=Panel)      # Points and lines
```

Bars at top indicates corresponding graph (i.e. countries)

from left to right starting on the bottom row

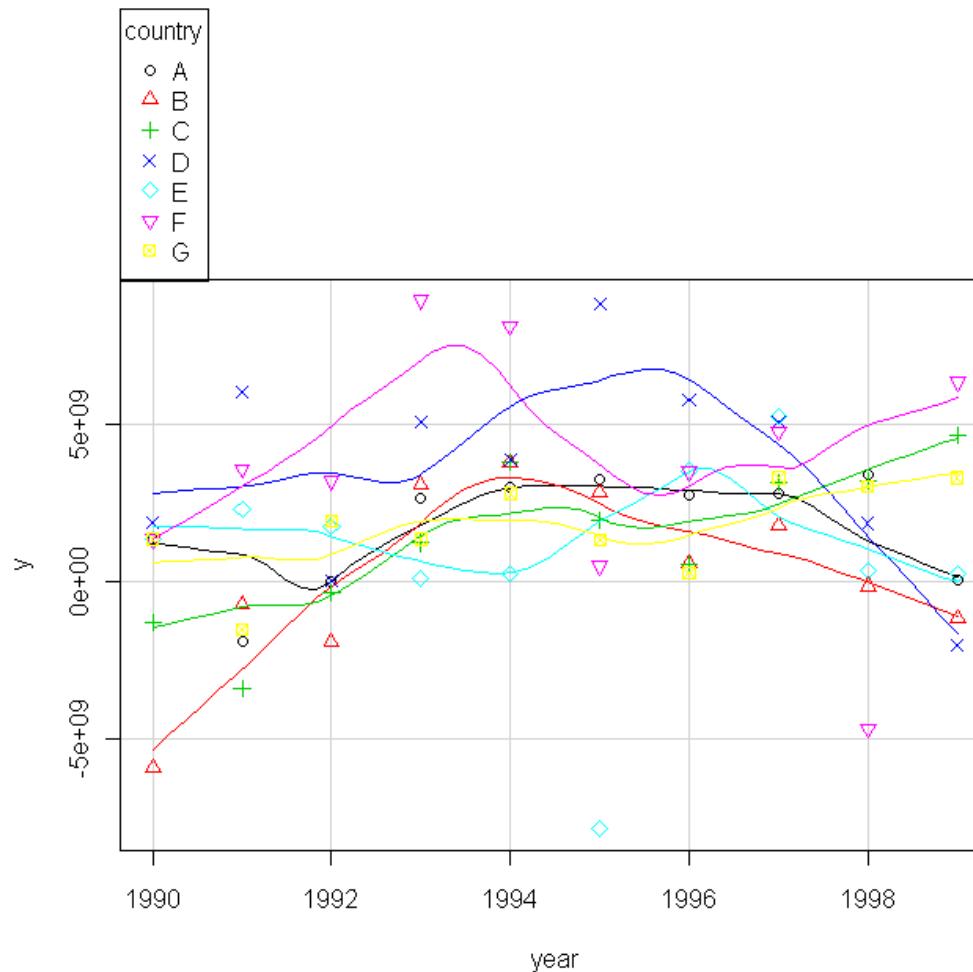
(Muenchen/Hilbe:355)

Given: country



Exploring panel data

```
library(foreign)
Panel <- read.dta("http://dss.princeton.edu/training/Panel101.dta")
library(car)
scatterplot(y~year|country, boxplots=FALSE, smooth=TRUE, reg.line=FALSE, data=Panel)
```



FIXED-EFFECTS MODEL

*(Covariance Model, Within Estimator,
Individual Dummy Variable Model, Least
Squares Dummy Variable Model)*

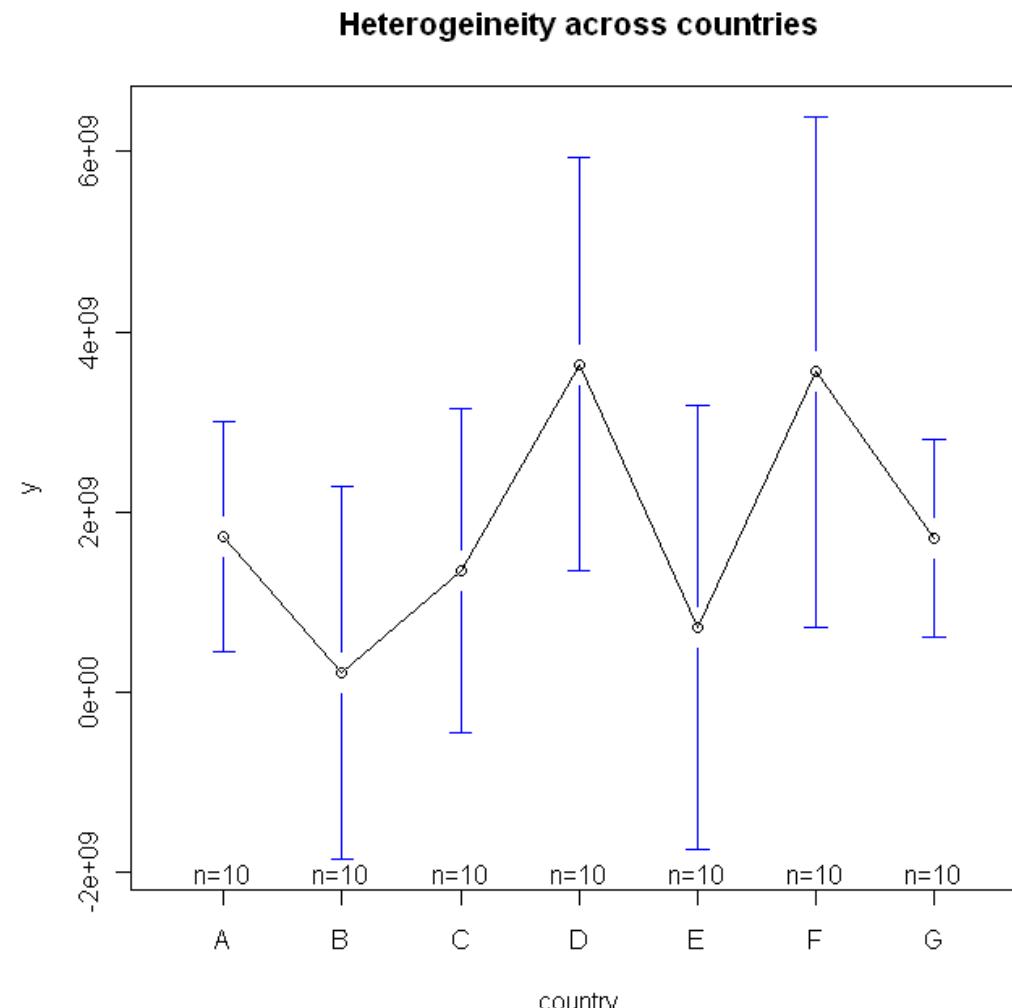
Fixed effects: Heterogeneity across countries (or entities)

```
library(foreign)
Panel <- read.dta("http://dss.princeton.edu/training/Panel101.dta")
library(gplots)
plotmeans(y ~ country, main="Heterogeneity across countries", data=Panel)

# plotmeans draw a 95%
confidence interval
around the means

detach("package:gplots")

# Remove package 'gplots'
from the workspace
```



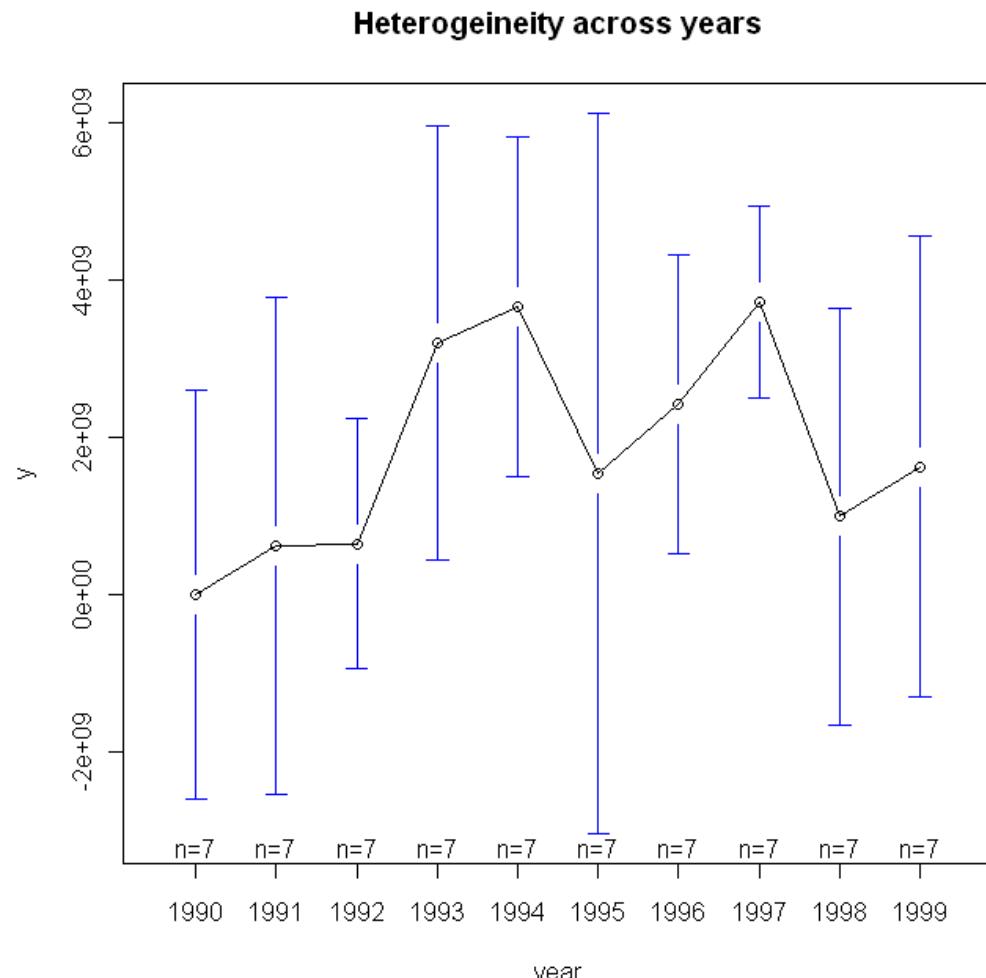
Fixed effects: Heterogeneity across years

```
library(foreign)
Panel <- read.dta("http://dss.princeton.edu/training/Panell01.dta")
library(gplots)
plotmeans(y ~ year, main="Heterogeneity across years", data=Panel)

# plotmeans draw a 95%
confidence interval
around the means

detach("package:gplots")

# Remove package 'gplots'
from the workspace
```



OLS regression

```
> library(foreign)
> Panel <- read.dta("http://dss.princeton.edu/training/Panel101.dta")
> ols <- lm(y ~ x1, data=Panel)
> summary(ols)
```

```
Call:
lm(formula = y ~ x1, data = Panel)

Residuals:
    Min      1Q  Median      3Q     Max 
-9.546e+09 -1.578e+09  1.554e+08  1.422e+09  7.183e+09 

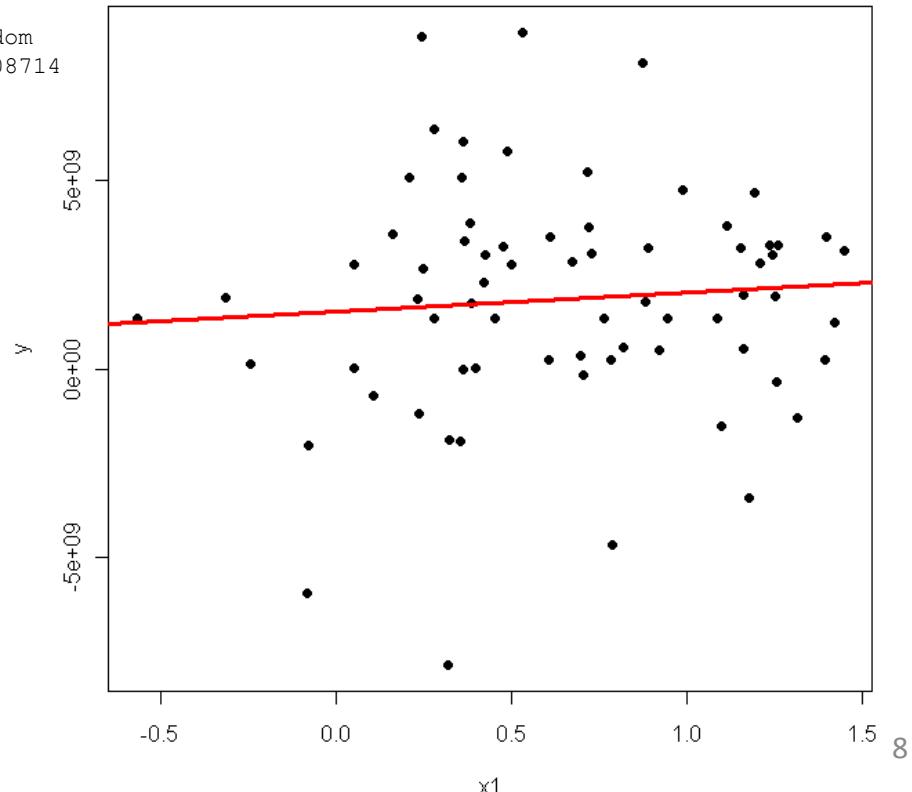
Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 1.524e+09  6.211e+08   2.454   0.0167 *  
x1          4.950e+08  7.789e+08   0.636   0.5272    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 3.028e+09 on 68 degrees of freedom
Multiple R-squared: 0.005905, Adjusted R-squared: -0.008714 
F-statistic: 0.4039 on 1 and 68 DF, p-value: 0.5272
```

```
> yhat <- ols$fitted
```

```
> plot(Panel$x1, Panel$y, pch=19, xlab="x1", ylab="y")
> abline(lm(Panel$y~Panel$x1), lwd=3, col="red")
```

Regular OLS regression does
not consider heterogeneity
across groups or time



Fixed effects using Least squares dummy variable model

```
> library(foreign)
>Panel <- read.dta("http://dss.princeton.edu/training/Panel101.dta")

> fixed.dum <- lm(y ~ x1 + factor(country) - 1, data=Panel)
> summary(fixed.dum)

Call:
lm(formula = y ~ x1 + factor(country) - 1, data = Panel)

Residuals:
    Min      1Q  Median      3Q     Max 
-8.634e+09 -9.697e+08  5.405e+08  1.386e+09  5.612e+09 

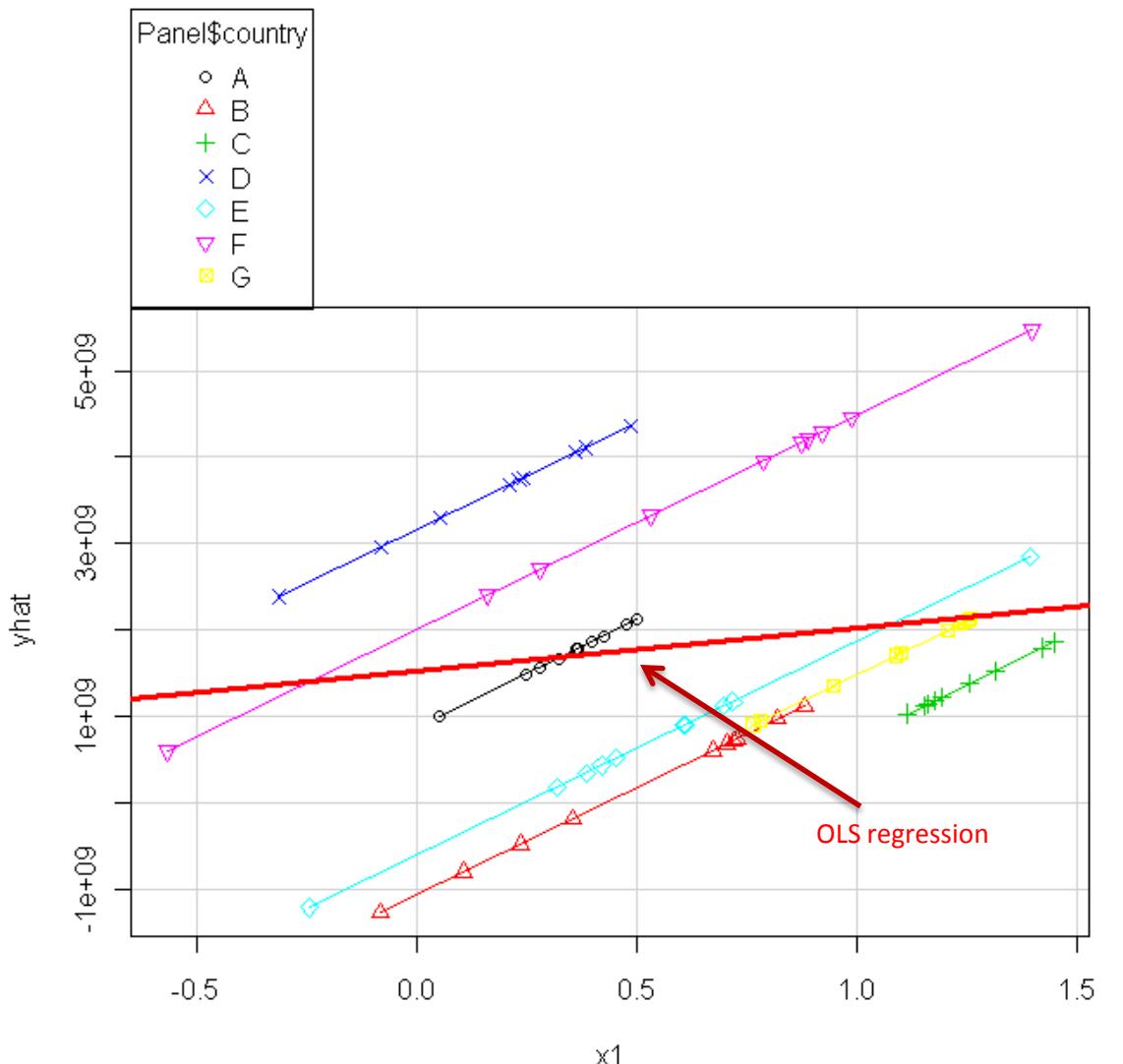
Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
x1          2.476e+09  1.107e+09  2.237 0.02889 *  
factor(country)A 8.805e+08  9.618e+08  0.916 0.36347  
factor(country)B -1.058e+09  1.051e+09 -1.006 0.31811  
factor(country)C -1.723e+09  1.632e+09 -1.056 0.29508  
factor(country)D  3.163e+09  9.095e+08  3.478 0.00093 *** 
factor(country)E -6.026e+08  1.064e+09 -0.566 0.57329  
factor(country)F  2.011e+09  1.123e+09  1.791 0.07821 .  
factor(country)G -9.847e+08  1.493e+09 -0.660 0.51190  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.796e+09 on 62 degrees of freedom
Multiple R-squared:  0.4402,    Adjusted R-squared:  0.368 
F-statistic: 6.095 on 8 and 62 DF,  p-value: 8.892e-06
```

For the theory behind fixed effects, please see <http://dss.princeton.edu/training/Panel101.pdf>

Least squares dummy variable model

```
> yhat <- fixed.dum$fitted  
> library(car)  
> scatterplot(yhat~Panel$x1|Panel$country, boxplots=FALSE, xlab="x1", ylab="yhat", smooth=FALSE)  
> abline(lm(Panel$y~Panel$x1), lwd=3, col="red")
```



Comparing OLS vs LSDV model

Each component of the factor variable (country) is absorbing the effects particular to each country. Predictor x_1 was not significant in the OLS model, once controlling for differences across countries, x_1 became significant in the OLS_DUM (i.e. LSDV model).

```
> library(apsrtable)
> apsrtable(ols,fixed.dum, model.names = c("OLS", "OLS_DUM")) # Displays a table in Latex form
```

```
\begin{table}[!ht]
\caption{}
\label{}
\begin{tabular}{ l D{.}{.}{2}D{.}{.}{2} }
\hline
& \multicolumn{1}{c}{OLS} & \multicolumn{1}{c}{OLS\_DUM} \\
\hline
% & OLS & OLS\_DUM \\
(Intercept) & 1524319070.05 ** & \\
& (621072623.86) & \\
x1 & 494988913.90 & 2475617827.10 ** \\
& (778861260.95) & (1106675593.60) \\
factor(country)A & 880542403.99 & \\
& (961807052.24) & \\
factor(country)B & -1057858363.16 & \\
& (1051067684.19) & \\
factor(country)C & -1722810754.55 & \\
& (1631513751.40) & \\
factor(country)D & 3162826897.32 ** \\
& (909459149.66) & \\
factor(country)E & -602622000.33 & \\
& (1064291684.41) & \\
factor(country)F & 2010731793.24 & \\
& (1122809097.35) & \\
factor(country)G & -984717493.45 & \\
& (1492723118.24) & \\
\$N\$ & 70 & 70 \\
\$R^2\$ & 0.01 & 0.44 \\
adj. \$R^2\$ & -0.01 & 0.37 \\
Resid. sd & 3028276248.26 & 2795552570.60 \\
\multicolumn{3}{l}{\footnotesize Standard errors in parentheses} \\
\multicolumn{3}{l}{\footnotesize ** indicates significance at $p< 0.05$} \\
\end{tabular}
\end{table}
```

The coefficient of x_1 indicates how much Y changes when X increases by one unit. Notice x_1 is not significant in the OLS model

The coefficient of x_1 indicates how much Y changes overtime, controlling by differences in countries, when X increases by one unit. Notice x_1 is significant in the LSDV model

```
> cat(apsrtable(ols, fixed.dum, model.names = c("OLS", "OLS_DUM"), Sweave=F), file="ols_fixed1.txt")
# Exports the table to a text file (in Latex code).
```

Fixed effects: n entity-specific intercepts (using plm)

```
> library(plm)
> fixed <- plm(y ~ x1, data=Panel, index=c("country", "year"), model="within")
> summary(fixed)
```

Oneway (individual) effect Within Model

Predictor variable(s)

Call:

```
plm(formula = y ~ x1, data = Panel, model = "within", index = c("country",
  "year"))
```

Balanced Panel: n=7, T=10, N=70

n = # of groups/panels, T = # years, N = total # of observations

Residuals :

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
-8.63e+09	-9.70e+08	5.40e+08	1.81e-10	1.39e+09	5.61e+09

Coefficients :

	Estimate	Std. Error	t-value	Pr(> t)
x1	2475617827	1106675594	2.237	0.02889 *

Signif. codes: 0 '****' 0.001 '***' 0.01 '**' 0.05 '*' 0.1 '.' 1

Total Sum of Squares: 5.2364e+20

Residual Sum of Squares: 4.8454e+20

R-Squared : 0.074684

Adj. R-Squared : 0.066148

F-statistic: 5.00411 on 1 and 62 DF, p-value: 0.028892

$\text{Pr}(>|t|)$ = Two-tail p-values test the hypothesis that each coefficient is different from 0. To reject this, the p-value has to be lower than 0.05 (95%, you could choose also an alpha of 0.10), if this is the case then you can say that the variable has a significant influence on your dependent variable (y)

The coeff of $x1$ indicates how much Y changes overtime, on average per country, when X increases by one unit.

```
> fixef(fixed) # Display the fixed effects (constants for each country)
```

A	B	C	D	E	F
880542404	-1057858363	-1722810755	3162826897	-602622000	2010731793
G					
-984717493					

```
> pFtest(fixed, ols) # Testing for fixed effects, null: OLS better than fixed
```

F test for individual effects

```
data: y ~ x1
F = 2.9655, df1 = 6, df2 = 62, p-value = 0.01307
alternative hypothesis: significant effects
```

If this number is < 0.05 then your model is ok. This is a test (F) to see whether all the coefficients in the model are different than zero.

If the p-value is < 0.05 then the fixed effects model is a better choice

RANDOM-EFFECTS MODEL

(Random Intercept, Partial Pooling Model)

Random effects (using plm)

```
> random <- plm(y ~ x1, data=Panel, index=c("country", "year"), model="random")
> summary(random)
```

Oneway (individual) effect Random Effect Model
(Swamy-Arora's transformation)

Call:

```
plm(formula = y ~ x1, data = Panel, model = "random", index = c("country",
"year"))
```

Balanced Panel: n=7, T=10, N=70

Effects:

	var	std.dev	share
idiosyncratic	7.815e+18	2.796e+09	0.873
individual	1.133e+18	1.065e+09	0.127
theta:	0.3611		

Interpretation of the coefficients is tricky since they include both the within-entity and between-entity effects. In the case of TSCS data represents the average effect of X over Y when X changes across time and between countries by one unit.

Outcome variable

Random effects (using plm)

Predictor variable(s)

Panel setting

Random effects option

n = # of groups/panels, T = # years, N = total # of observations

Residuals :

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
-8.94e+09	-1.51e+09	2.82e+08	5.29e-08	1.56e+09	6.63e+09

Coefficients :

	Estimate	Std. Error	t-value	Pr(> t)
(Intercept)	1037014284	790626206	1.3116	0.1941
x1	1247001782	902145601	1.3823	0.1714

$Pr(>|t|)$ =Two-tail p-values test the hypothesis that each coefficient is different from 0. To reject this, the p-value has to be lower than 0.05 (95%, you could choose also an alpha of 0.10), if this is the case then you can say that the variable has a significant influence on your dependent variable (y)

Total Sum of Squares: 5.6595e+20

Residual Sum of Squares: 5.5048e+20

R-Squared : 0.02733

Adj. R-Squared : 0.026549

F-statistic: 1.91065 on 1 and 68 DF, p-value: 0.17141

If this number is < 0.05 then your model is ok. This is a test (F) to see whether all the coefficients in the model are different than zero.

```
# Setting as panel data (an alternative way to run the above model
```

```
Panel.set <- plm.data(Panel, index = c("country", "year"))
```

```
# Random effects using panel setting (same output as above)
```

```
random.set <- plm(y ~ x1, data = Panel.set, model="random")
summary(random.set)
```

FIXED OR RANDOM?

To decide between fixed or random effects you can run a Hausman test where the null hypothesis is that the preferred model is random effects vs. the alternative the fixed effects (see Green, 2008, chapter 9). It basically tests whether the unique errors (u_i) are correlated with the regressors, the null hypothesis is they are not.

Run a fixed effects model and save the estimates, then run a random model and save the estimates, then perform the test. If the p-value is significant (for example <0.05) then use fixed effects, if not use random effects.

```
> phtest(fixed, random)
```

Hausman Test

```
data: y ~ x1
chisq = 3.674, df = 1, p-value = 0.05527
alternative hypothesis: one model is inconsistent
```

If this number is < 0.05 then use fixed effects



OTHER TESTS/ DIAGNOSTICS

Testing for time-fixed effects

```
> library(plm)
> fixed <- plm(y ~ x1, data=Panel, index=c("country", "year"),
model="within")
> fixed.time <- plm(y ~ x1 + factor(year), data=Panel, index=c("country",
"year"), model="within")
> summary(fixed.time)
Oneway (individual) effect Within Model

Call:
plm(formula = y ~ x1 + factor(year), data = Panel, model = "within",
index = c("country", "year"))

Balanced Panel: n=7, T=10, N=70

Residuals :
    Min.   1st Qu.    Median     Mean   3rd Qu.    Max. 
-7.92e+09 -1.05e+09 -1.40e+08  1.48e-07  1.63e+09  5.49e+09 

Coefficients :
            Estimate Std. Error t-value Pr(>|t|)    
x1          1389050354 1319849567  1.0524  0.29738  
factor(year)1991 296381559 1503368528  0.1971  0.84447  
factor(year)1992 145369666 1547226548  0.0940  0.92550  
factor(year)1993 2874386795 1503862554  1.9113  0.06138 .  
factor(year)1994 2848156288 1661498927  1.7142  0.09233 .  
factor(year)1995 973941306 1567245748  0.6214  0.53698  
factor(year)1996 1672812557 1631539254  1.0253  0.30988  
factor(year)1997 2991770063 1627062032  1.8388  0.07156 .  
factor(year)1998 367463593 1587924445  0.2314  0.81789  
factor(year)1999 1258751933 1512397632  0.8323  0.40898  
---
Signif. codes:  0 `****' 0.001 `***' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Total Sum of Squares:  5.2364e+20
Residual Sum of Squares: 4.0201e+20
R-Squared : 0.23229
Adj. R-Squared : 0.17588
F-statistic: 1.60365 on 10 and 53 DF, p-value: 0.13113
```

```
> # Testing time-fixed effects. The null is that no time-fixed effects needed
```

```
> pftest(fixed.time, fixed)
```

F test for individual effects

```
data: y ~ x1 + factor(year)
F = 1.209, df1 = 9, df2 = 53, p-value = 0.3094
alternative hypothesis: significant effects
```

```
> plmtest(fixed, c("time"), type="bp")
```

Lagrange Multiplier Test - time effects (Breusch-Pagan)

```
data: y ~ x1
chisq = 0.1653, df = 1, p-value = 0.6843
alternative hypothesis: significant effects
```

If this number is < 0.05 then use time-fixed effects. In this example, no need to use time-fixed effects.

Testing for random effects: Breusch-Pagan Lagrange multiplier (LM)

```
> # Regular OLS (pooling model) using plm
>
> pool <- plm(y ~ x1, data=Panel, index=c("country", "year"), model="pooling")
> summary(pool)
  Oneway (individual) effect Pooling Model

Call:
plm(formula = y ~ x1, data = Panel, model = "pooling", index = c("country",
"year"))

Balanced Panel: n=7, T=10, N=70

Residuals :
    Min.   1st Qu.    Median      Mean   3rd Qu.      Max. 
-9.55e+09 -1.58e+09  1.55e+08  1.77e-08  1.42e+09  7.18e+09 

Coefficients :
            Estimate Std. Error t-value Pr(>|t|)    
(Intercept) 1524319070  621072624  2.4543  0.01668 *  
x1          494988914  778861261  0.6355  0.52722    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Total Sum of Squares:  6.2729e+20
Residual Sum of Squares: 6.2359e+20
R-Squared : 0.0059046
Adj. R-Squared : 0.0057359
F-statistic: 0.403897 on 1 and 68 DF, p-value: 0.52722

> # Breusch-Pagan Lagrange Multiplier for random effects. Null is no panel effect (i.e. OLS better).

> plmtest(pool, type=c("bp"))

Lagrange Multiplier Test - (Breusch-Pagan)

data: y ~ x1
chisq = 2.6692, df = 1, p-value = 0.1023
alternative hypothesis: significant effects
```

The LM test helps you decide between a random effects regression and a simple OLS regression.

The null hypothesis in the LM test is that variances across entities is zero. This is, no significant difference across units (i.e. no panel effect). (<http://dss.princeton.edu/training/Panel101.pdf>)

Here we failed to reject the null and conclude that random effects is not appropriate. This is, no evidence of significant differences across countries, therefore you can run a simple OLS regression.

Testing for cross-sectional dependence/contemporaneous correlation: using Breusch-Pagan LM test of independence and Pasaran CD test

According to Baltagi, cross-sectional dependence is a problem in macro panels with long time series. This is not much of a problem in micro panels (few years and large number of cases).

The null hypothesis in the B-P/LM and Pasaran CD tests of independence is that residuals across entities are not correlated. B-P/LM and Pasaran CD (cross-sectional dependence) tests are used to test whether the residuals are correlated across entities*. Cross-sectional dependence can lead to bias in tests results (also called contemporaneous correlation).

```
> fixed <- plm(y ~ x1, data=Panel, index=c("country", "year"), model="within")  
  
> pcdtest(fixed, test = c("lm"))
```

Breusch-Pagan LM test for cross-sectional dependence in panels

```
data: formula  
chisq = 28.9143, df = 21, p-value = 0.1161  
alternative hypothesis: cross-sectional dependence
```

No cross-sectional dependence

```
> pcdtest(fixed, test = c("cd"))
```

Pesaran CD test for cross-sectional dependence in panels

```
data: formula  
z = 1.1554, p-value = 0.2479  
alternative hypothesis: cross-sectional dependence
```

*Source: Hoechle, Daniel, "Robust Standard Errors for Panel Regressions with Cross-Sectional Dependence",
http://fmwww.bc.edu/repec/bocode/x/xtscce_paper.pdf

Testing for serial correlation

Serial correlation tests apply to macro panels with long time series. Not a problem in micro panels (with very few years). The null is that there is not serial correlation.

```
> pbgttest(fixed)
Loading required package: lmtest

Breusch-Godfrey/Wooldridge test for serial correlation in panel models

data: y ~ x1
chisq = 14.1367, df = 10, p-value = 0.1668
alternative hypothesis: serial correlation in idiosyncratic errors
```

No serial correlation

The Dickey-Fuller test to check for stochastic trends. The null hypothesis is that the series has a unit root (i.e. non-stationary). If unit root is present you can take the first difference of the variable.

```
> Panel.set <- plm.data(Panel, index = c("country", "year"))

> library(tseries)

> adf.test(Panel.set$y, k=2)

Augmented Dickey-Fuller Test

data: Panel.set$y
Dickey-Fuller = -3.9051, Lag order = 2, p-value = 0.01910
alternative hypothesis: stationary
```

If p-value < 0.05 then no unit roots present.



Testing for heteroskedasticity

The null hypothesis for the Breusch-Pagan test is homoskedasticity.

```
> library(lmtest)
> bptest(y ~ x1 + factor(country), data = Panel, studentize=F)

Breusch-Pagan test

data: y ~ x1 + factor(country)
BP = 14.6064, df = 7, p-value = 0.04139
```

Presence of heteroskedasticity

If heteroskedasticity is detected you can use robust covariance matrix to account for it. See the following pages.

Controlling for heteroskedasticity: Robust covariance matrix estimation (Sandwich estimator)

The `--vcovHC`- function estimates three heteroskedasticity-consistent covariance estimators:

- "white1" - for general heteroskedasticity but no serial correlation. Recommended for random effects.
- "white2" - is "white1" restricted to a common variance within groups. Recommended for random effects.
- "arellano" - both heteroskedasticity and serial correlation. Recommended for fixed effects.

The following options apply*:

- HC0 - heteroskedasticity consistent. The default.
- HC1, HC2, HC3 - Recommended for small samples. HC3 gives less weight to influential observations.
- HC4 - small samples with influential observations
- HAC - heteroskedasticity and autocorrelation consistent (type `?vcovHAC` for more details)

See the following pages for examples

For more details see:

- <http://cran.r-project.org/web/packages/plm/vignettes/plm.pdf>
- <http://cran.r-project.org/web/packages/sandwich/vignettes/sandwich.pdf> (see page 4)
- Stock and Watson 2006.
- *Kleiber and Zeileis, 2008.

Controlling for heteroskedasticity: Random effects

```
> random <- plm(y ~ x1, data=Panel, index=c("country", "year"), model="random")

> coeftest(random)      # Original coefficients

t test of coefficients:

            Estimate Std. Error t value Pr(>|t|) 
(Intercept) 1037014284  790626206  1.3116  0.1941 
x1          1247001782  902145601  1.3823  0.1714 

> coeftest(random, vcovHC)      # Heteroskedasticity consistent coefficients

t test of coefficients:

            Estimate Std. Error t value Pr(>|t|) 
(Intercept) 1037014284  907983029  1.1421  0.2574 
x1          1247001782  828970247  1.5043  0.1371 

> coeftest(random, vcovHC(random, type = "HC3")) # Heteroskedasticity consistent coefficients, type 3

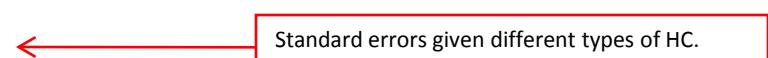
t test of coefficients:

            Estimate Std. Error t value Pr(>|t|) 
(Intercept) 1037014284  943438284  1.0992  0.2756 
x1          1247001782  867137585  1.4381  0.1550 

> # The following shows the HC standard errors of the coefficients

> t(sapply(c("HC0", "HC1", "HC2", "HC3", "HC4"), function(x) sqrt(diag(vcovHC(random, type = x)))))

            (Intercept)           x1
HC0      907983029 828970247
HC1      921238957 841072643
HC2      925403820 847733474
HC3      943438284 867137584
HC4      941376033 866024033



Standard errors given different types of HC.


```

```

> fixed <- plm(y ~ x1, data=Panel, index=c("country", "year"), model="within")

> coeftest(fixed)      # Original coefficients

t test of coefficients:

    Estimate Std. Error t value Pr(>|t|)
x1 2475617827 1106675594   2.237  0.02889 *
---
Signif. codes:  0 '****' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> coeftest(fixed, vcovHC) # Heteroskedasticity consistent coefficients

t test of coefficients:

    Estimate Std. Error t value Pr(>|t|)
x1 2475617827 1358388942   1.8225  0.07321 .
---
Signif. codes:  0 '****' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> coeftest(fixed, vcovHC(fixed, method = "arellano")) # Heteroskedasticity consistent coefficients (Arellano)

t test of coefficients:

    Estimate Std. Error t value Pr(>|t|)
x1 2475617827 1358388942   1.8225  0.07321 .
---
Signif. codes:  0 '****' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> coeftest(fixed, vcovHC(fixed, type = "HC3")) # Heteroskedasticity consistent coefficients, type 3

t test of coefficients:

    Estimate Std. Error t value Pr(>|t|)
x1 2475617827 1439083523   1.7203  0.09037 .
---
Signif. codes:  0 '****' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> # The following shows the HC standard errors of the coefficients
> t(sapply(c("HC0", "HC1", "HC2", "HC3", "HC4"), function(x) sqrt(diag(vcovHC(fixed, type = x)))))

  HC0.x1     HC1.x1     HC2.x1     HC3.x1     HC4.x1
[1,] 1358388942 1368196931 1397037369 1439083523 1522166034

```

Controlling for heteroskedasticity: Fixed effects

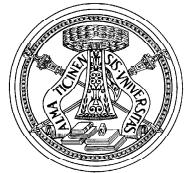
Standard errors given different types of HC.

References/Useful links

- DSS Online Training Section <http://dss.princeton.edu/training/>
- Princeton DSS Libguides <http://libguides.princeton.edu/dss>
- John Fox's site <http://socserv.mcmaster.ca/jfox/>
- Quick-R <http://www.statmethods.net/>
- UCLA Resources to learn and use R <http://www.ats.ucla.edu/stat/R/>
- UCLA Resources to learn and use Stata <http://www.ats.ucla.edu/stat/stata/>
- DSS - Stata http://dss/online_help/stats_packages/stata/
- DSS - R http://dss.princeton.edu/online_help/stats_packages/r
- Panel Data Econometrics in R: the plm package <http://cran.r-project.org/web/packages/plm/vignettes/plm.pdf>
- Econometric Computing with HC and HAC Covariance Matrix Estimators
<http://cran.r-project.org/web/packages/sandwich/vignettes/sandwich.pdf>

References/Recommended books

- *An R Companion to Applied Regression*, Second Edition / John Fox , Sanford Weisberg, Sage Publications, 2011
- *Data Manipulation with R* / Phil Spector, Springer, 2008
- *Applied Econometrics with R* / Christian Kleiber, Achim Zeileis, Springer, 2008
- *Introductory Statistics with R* / Peter Dalgaard, Springer, 2008
- *Complex Surveys. A guide to Analysis Using R* / Thomas Lumley, Wiley, 2010
- *Applied Regression Analysis and Generalized Linear Models* / John Fox, Sage, 2008
- *R for Stata Users* / Robert A. Muenchen, Joseph Hilbe, Springer, 2010
- *Introduction to econometrics* / James H. Stock, Mark W. Watson. 2nd ed., Boston: Pearson Addison Wesley, 2007.
- *Data analysis using regression and multilevel/hierarchical models* / Andrew Gelman, Jennifer Hill. Cambridge ; New York : Cambridge University Press, 2007.
- *Econometric analysis* / William H. Greene. 6th ed., Upper Saddle River, N.J. : Prentice Hall, 2008.
- *Designing Social Inquiry: Scientific Inference in Qualitative Research* / Gary King, Robert O. Keohane, Sidney Verba, Princeton University Press, 1994.
- *Unifying Political Methodology: The Likelihood Theory of Statistical Inference* / Gary King, Cambridge University Press, 1989
- *Statistical Analysis: an interdisciplinary introduction to univariate & multivariate methods* / Sam Kachigan, New York : Radius Press, c1986
- *Statistics with Stata (updated for version 9)* / Lawrence Hamilton, Thomson Books/Cole, 2006



Università di Pavia

Impulse Response Functions

Eduardo Rossi



VAR

VAR(p):

$$\mathbf{y}_t = (y_{1t}, \dots, y_{Nt})'$$

$$\Phi(L)\mathbf{y}_t = \boldsymbol{\epsilon}_t$$

$$\Phi(L) = \mathbf{I}_N - \Phi_1 L - \dots - \Phi_p L^p$$

$$\boldsymbol{\epsilon}_t = (\epsilon_{1t}, \dots, \epsilon_{Nt})'$$

$$\boldsymbol{\epsilon}_t \sim \text{independent } VWN(\mathbf{0}, \boldsymbol{\Omega})$$

The process is stable if

$$\det(\mathbf{I}_N - \Phi_1 z - \dots - \Phi_p z^p) \neq 0 \text{ for } |z| \leq 1$$

On the assumption that the process has been initiated in the infinite past ($t = 0, \pm 1, \pm 2, \dots$) it generates stationary time series that have time-invariant means, variances, and covariances.



VAR

Because VAR models represent the correlations among a set of variables, they are often used to analyze certain aspects of the relationships between the variables of interest.



GRANGER-CAUSALITY

Granger (1969) has defined a concept of *causality* which, under suitable conditions, is fairly easy to deal with in the context of VAR models. Therefore it has become quite popular in recent years. The idea is that *a cause cannot come after the effect*.

If a variable x affects a variable z , the former should help improving the predictions of the latter variable.

Ω_t is the information set containing all the relevant information in the universe available up to and including period t .

$z_t(h|\Omega_t)$ be the optimal (minimum MSE) h -step predictor of the process z_t at origin t , based on the information in Ω_t .

The corresponding forecast MSE: $\Sigma_t(h|\Omega_t)$.



GRANGER-CAUSALITY

The process x_t is said to cause z_t in Granger's sense if

$$\Sigma_t(h|\Omega_t) < \Sigma_t(h|\Omega_t \{x_s : s \leq t\}) \quad \text{for at least one} \quad h = 1, 2, \dots$$

$\Omega_t \{x_s : s \leq t\}$ is the set containing all the relevant information in the universe except for the information in the past and present of the x_t process.

If z_t can be predicted more efficiently if the information in the x_t process is taken into account in addition to all other information in the universe, then x_t is *Granger-causal* for z_t .



INSTANTANEOUS CAUSALITY

If $\mathbf{x}_t : (N \times 1)$ causes $\mathbf{z}_t : (M \times 1)$ and \mathbf{z}_t also causes \mathbf{x}_t the process $(\mathbf{z}'_t, \mathbf{x}'_t)'$ is called a *feedback system*.

We say that there is *instantaneous causality* between z_t and x_t if

$$\Sigma_z(1|\Omega_t \cup \mathbf{x}_{t+1}) \neq \Sigma_z(1|\Omega_t)$$

In other words, in period t , adding \mathbf{x}_{t+1} to the information set helps to improve the forecast of \mathbf{z}_{t+1} .

This concept of causality is really symmetric, that is, if there is instantaneous causality between \mathbf{z}_t and \mathbf{x}_t , then there is also instantaneous causality between \mathbf{x}_t and \mathbf{z}_t .



INSTANTANEOUS CAUSALITY

A possible criticism of the foregoing definitions could relate to the choice of the MSE as a measure of the forecast precision. Of course, the choice of another measure could lead to a different definition of causality.

Equality of the MSEs will imply equality of the corresponding predictors.

In that case a process \mathbf{z}_t is not Granger-caused by \mathbf{x}_t if the optimal predictor of \mathbf{z}_t does not use information from the \mathbf{x}_t process. This result is intuitively appealing.



INSTANTANEOUS CAUSALITY

A more serious practical problem is the choice of the information set Ω_t . Usually all the relevant information in the universe is not available to a forecaster and, thus, the optimal predictor given Ω_t cannot be determined. Therefore a less demanding definition of causality is often used in practice.



INSTANTANEOUS CAUSALITY

Instead of all the information in the universe, only the information in the past and present of the process under study is considered relevant and Ω_t is replaced by $\{\mathbf{z}_s, \mathbf{x}_s | s \leq t\}$. Furthermore, instead of optimal predictors, *optimal linear predictors* are compared.



IMPULSE RESPONSES FUNCTIONS

Granger-causality may not tell us the complete story about the interactions between the variables of a system.

In applied work, it is often of interest to know the response of one variable to an impulse in another variable in a system that involves a number of further variables as well.

One would like to investigate the impulse response relationship between two variables in a higher dimensional system. Of course, if there is a reaction of one variable to an impulse in another variable we may call the latter causal for the former.

We will study this type of causality by tracing out the effect of an exogenous shock or innovation in one of the variables on some or all of the other variables.

This kind of impulse response analysis is called *multiplier analysis*.



IMPULSE RESPONSES FUNCTIONS

For instance, in a system consisting of an inflation rate and an interest rate, the effect of an increase in the inflation rate may be of interest.

In the real world, such an increase may be induced exogenously from outside the system by events like the increase of the oil price in 1973/74 when the OPEC agreed on a joint action to raise prices.

Alternatively, an increase or reduction in the interest rate may be administered by the central bank for reasons outside the simple two variable system under study.



IMPULSE RESPONSES FUNCTIONS

Three-variables system:

$$\mathbf{y}_t = \begin{bmatrix} y_{1,t} \\ y_{2,t} \\ y_{3,t} \end{bmatrix} \quad \begin{array}{l} \text{investment} \\ \text{income} \\ \text{consumption} \end{array}$$

the effect of an innovation in investment. To isolate such an effect, suppose that all three variables assume their mean value prior to time $t = 0$, $\mathbf{y}_t = \boldsymbol{\mu}, t < 0$, and investment increases by one unit in period $t = 0$, i.e. $\epsilon_{10} = 1$.



IMPULSE RESPONSES FUNCTIONS

Now we can trace out what happens to the system during periods $t = 1, 2, \dots$ if no further shocks occur, that is,

$$\epsilon_{20} = \epsilon_{30} = 0$$

and

$$\epsilon_2 = 0, \epsilon_3 = 0, \dots$$

We assume that all three variables have mean zero and set $\mathbf{c} = \mathbf{0}$, then

$$\mathbf{y}_t = \Phi_1 \mathbf{y}_{t-1} + \epsilon_t$$



IMPULSE RESPONSES FUNCTIONS

$$\mathbf{y}_t = \begin{bmatrix} y_{1,t} \\ y_{2,t} \\ y_{3,t} \end{bmatrix} = \begin{bmatrix} .5 & 0 & 0 \\ .1 & .1 & .3 \\ 0 & .2 & .3 \end{bmatrix} \begin{bmatrix} y_{1,t-1} \\ y_{2,t-1} \\ y_{3,t-1} \end{bmatrix} + \begin{bmatrix} \epsilon_{1,t} \\ \epsilon_{2,t} \\ \epsilon_{3,t} \end{bmatrix}$$

Tracing a unit shock in the first variable in period $t = 0$ in this system we get

$$\mathbf{y}_0 = \begin{bmatrix} y_{1,0} \\ y_{2,0} \\ y_{3,0} \end{bmatrix} = \begin{bmatrix} \epsilon_{1,0} \\ \epsilon_{2,0} \\ \epsilon_{3,0} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{y}_1 = \begin{bmatrix} y_{1,1} \\ y_{2,1} \\ y_{3,1} \end{bmatrix} = \Phi_1 \mathbf{y}_0 = \begin{bmatrix} 0.5 \\ 0.1 \\ 0 \end{bmatrix}$$



IMPULSE RESPONSES FUNCTIONS

$$\mathbf{y}_2 = \begin{bmatrix} y_{1,2} \\ y_{2,2} \\ y_{3,2} \end{bmatrix} = \Phi_1 \mathbf{y}_1 = \Phi_1^2 \mathbf{y}_0 = \begin{bmatrix} 0.5 \\ 0.06 \\ 0.02 \end{bmatrix}$$

it turns out that $\mathbf{y}_i = (y_{1,i}, y_{2,i}, y_{3,i})'$ is just the first column of Φ_1^i

$$\mathbf{y}_i = \Phi_1^i \mathbf{u}_i$$

where

$$\mathbf{u}'_i = (0, 0, \dots, 1, \dots, 0)$$

the elements of Φ_1^i represent the effects of unit shocks in the variables of the system after i periods. They are called *impulse responses* or *dynamic multipliers*.



IMPULSE RESPONSES FUNCTIONS

Recall that $\Phi_1^i = \Psi_i$ just the i -th coefficient matrix of the MA representation of a VAR(1) process.

The MA coefficient matrices contain the impulse responses of the system. This result holds more generally for higher order VAR(p) processes as well.

$VMA(\infty)$ representation:

$$\mathbf{y}_t = \sum_{i=0}^{\infty} \boldsymbol{\Psi}_i \boldsymbol{\epsilon}_{t-i} \quad \boldsymbol{\Psi}_0 = \mathbf{I}_n$$



IMPULSE RESPONSES FUNCTIONS

Impulse-response function

$$\mathbf{y}_{t+n} = \sum_{i=0}^{\infty} \boldsymbol{\Psi}_i \boldsymbol{\epsilon}_{t+n-i}$$

$$\{\boldsymbol{\Psi}_n\}_{i,j} = \frac{\partial y_{it+n}}{\partial \epsilon_{jt}}$$

the response of $y_{i,t+n}$ to a one-time impulse in $y_{j,t}$ with all other variables dated t or earlier held constant.

The response of variable i to a unit shock (forecast error) in variable j is sometimes depicted graphically to get a visual impression of the dynamic interrelationships within the system.



IMPULSE RESPONSES FUNCTIONS

If the variables have different scales, it is sometimes useful to consider innovations of one standard deviation rather than unit shocks.

For instance, instead of tracing an unexpected unit increase in investment in the investment/income/consumption system with $Var(\epsilon_{1,t}) = 2.25$, one may follow up on a shock of $\sqrt{2.25} = 1.5$ units because the standard deviation of $\epsilon_{1,t}$ is 1.5.

Of course, this is just a matter of rescaling the impulse responses.

The impulse responses are zero if one of the variables does not Granger-cause the other variables taken as a group.

An innovation in variable k has no effect on the other variables if the former variable does not Granger-cause the set of the remaining variables.



THE ORTHOGONALIZED IMPULSE-RESPONSE FUNCTION

A problematic assumption in this type of impulse response analysis is that a shock occurs only in one variable at a time. Such an assumption may be reasonable if the shocks in different variables are independent.

If they are not independent one may argue that the error terms consist of all the influences and variables that are not directly included in the set of y variables.

Thus, in addition to forces that affect all the variables, there may be forces that affect variable 1, say, only. If a shock in the first variable is due to such forces it may again be reasonable to interpret the Ψ_i coefficients as dynamic responses.



THE ORTHOGONALIZED IMPULSE-RESPONSE FUNCTION

On the other hand, correlation of the error terms may indicate that a shock in one variable is likely to be accompanied by a shock in another variable.

In that case, setting all other errors to zero may provide a misleading picture of the actual dynamic relationships between the variables.



THE ORTHOGONALIZED IMPULSE-RESPONSE FUNCTION

This is the reason why impulse response analysis is often performed in terms of the MA representation:

$$\Omega = \mathbf{P}\mathbf{P}' \quad \text{Cholesky decomposition}$$

\mathbf{P} is a lower triangular matrix. From

$$\mathbf{y}_t = \boldsymbol{\epsilon}_t + \boldsymbol{\Psi}_1\boldsymbol{\epsilon}_{t-1} + \boldsymbol{\Psi}_2\boldsymbol{\epsilon}_{t-2} + \dots \quad \boldsymbol{\Psi}_0 = \mathbf{I}_n$$

to

$$\mathbf{y}_t = \boldsymbol{\Theta}_0\mathbf{w}_t + \boldsymbol{\Theta}_1\mathbf{w}_{t-1} + \boldsymbol{\Theta}_2\mathbf{w}_{t-2} + \dots$$

with

$$\boldsymbol{\Theta}_i = \boldsymbol{\Psi}_i\mathbf{P}$$

$$\mathbf{w}_t = \mathbf{P}^{-1}\boldsymbol{\epsilon}_t$$

$$E[\mathbf{w}_t\mathbf{w}_t'] = \mathbf{I}_N$$



THE ORTHOGONALIZED IMPULSE-RESPONSE FUNCTION

It is reasonable to assume that a change in one component of \mathbf{w}_t has no effect on the other components because the components are orthogonal (uncorrelated).

Moreover, the variances of the components are one. Thus, a unit innovation is just an innovation of size one standard deviation.

The elements of the Θ_i are interpreted as responses of the system to such innovations.

$$\{\Theta_i\}_{jk}$$

is assumed to represent the effect on variable j of a unit innovation in the k -th variable that has occurred i periods ago.



THE ORTHOGONALIZED IMPULSE-RESPONSE FUNCTION

To relate these impulse responses to a VAR model, we consider the zero mean VAR(p)process:

$$\Phi(L)\mathbf{y}_t = \boldsymbol{\epsilon}_t$$

This process can be rewritten in such a way that the disturbances of different equations are uncorrelated. For this purpose, we choose a decomposition of the white noise covariance matrix

$$\Omega = \mathbf{W}\Sigma\mathbf{W}'$$

Σ is a diagonal matrix with positive diagonal elements and \mathbf{W} is a lower triangular matrix with unit diagonal.



THE ORTHOGONALIZED IMPULSE-RESPONSE FUNCTION

This decomposition is obtained from the Choleski decomposition $\Omega = \mathbf{P}\mathbf{P}'$ by defining a diagonal matrix \mathbf{D} which has the same main diagonal as \mathbf{P} and by specifying

$$\mathbf{W} = \mathbf{P}\mathbf{D}^{-1}$$

$$\Sigma = \mathbf{D}\mathbf{D}'$$

then from

$$\Omega = \mathbf{P}\mathbf{P}'$$

and

$$\mathbf{P} = \mathbf{W}\mathbf{D}$$

$$\Omega = \mathbf{P}\mathbf{P}' = \mathbf{W}\mathbf{D}\mathbf{D}'\mathbf{W}' = \mathbf{W}\Sigma\mathbf{W}'$$



THE ORTHOGONALIZED IMPULSE-RESPONSE FUNCTION

Premultiplying the VAR(p) by $\mathbf{A} \equiv \mathbf{W}^{-1}$

$$\mathbf{A}\mathbf{y}_t = \mathbf{A}_1^*\mathbf{y}_{t-1} + \dots + \mathbf{A}_p^*\mathbf{y}_{t-p} + \mathbf{e}_t$$

where

$$\mathbf{A}_i^* = \mathbf{A}\Phi_i$$

$$\mathbf{e}_t = \mathbf{A}\epsilon_t$$



THE ORTHOGONALIZED IMPULSE-RESPONSE FUNCTION

\mathbf{e}_t has a diagonal var-cov matrix:

$$E[\mathbf{e}_t \mathbf{e}'_t] = \mathbf{A} E[\boldsymbol{\epsilon}_t \boldsymbol{\epsilon}'_t] \mathbf{A}' = \mathbf{A} \boldsymbol{\Omega} \mathbf{A}'$$

Adding $(\mathbf{I}_N - \mathbf{A})\mathbf{y}_t$ to both sides gives

$$(\mathbf{I}_N - \mathbf{A})\mathbf{y}_t + \mathbf{A}\mathbf{y}_t = (\mathbf{I}_N - \mathbf{A})\mathbf{y}_t + \mathbf{A}_1^*\mathbf{y}_{t-1} + \dots + \mathbf{A}_p^*\mathbf{y}_{t-p} + \mathbf{e}_t$$

$$\mathbf{y}_t = \mathbf{A}_0^*\mathbf{y}_t + \mathbf{A}_1^*\mathbf{y}_{t-1} + \dots + \mathbf{A}_p^*\mathbf{y}_{t-p} + \mathbf{e}_t$$

Because \mathbf{W} is lower triangular with unit diagonal, the same is true for \mathbf{A} .



THE ORTHOGONALIZED IMPULSE-RESPONSE FUNCTION

Hence,

$$\mathbf{A}_0^* = (\mathbf{I}_N - \mathbf{A})$$

$$= \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ \beta_{12} & 0 & \dots & 0 & 0 \\ \vdots & \ddots & & \ddots & \vdots \\ \vdots & & & & \vdots \\ \beta_{N,1} & \beta_{N,2} & \dots & \beta_{N,N-1} & 0 \end{bmatrix}$$

is a lower triangular matrix with zero diagonal and, thus, in the representation of the VAR(p) process, the first equation contains no instantaneous y 's on the right-hand side. The second equation may contain $y_{1,t}$ and otherwise lagged y 's on the right-hand side. More generally, the k -th equation may contain $y_{1,t}, \dots, y_{k-1,t}$ and not $y_{k,t}, \dots, y_{N,t}$ on the right-hand side.



THE ORTHOGONALIZED IMPULSE-RESPONSE FUNCTION

Thus, if the VAR(p) with instantaneous effects reflects the actual goings on in the system, y_{st} cannot have an instantaneous impact on y_{kt} for $k < s$.

In the econometrics literature such a system is called a *recursive model* (Theil (1971)). Wold has advocated these models where the researcher has to specify the instantaneous "causal" ordering of the variables. This type of causality is therefore sometimes referred to as *Wold-causality*.



THE ORTHOGONALIZED IMPULSE-RESPONSE FUNCTION

If we trace e_{it} innovations of size one standard error through the system, we just get the Θ impulse responses. This can be seen by solving the system for \mathbf{y}_t :

$$(\mathbf{I}_N - \mathbf{A}_0^*)\mathbf{y}_t = \mathbf{A}_1^*\mathbf{y}_{t-1} + \dots + \mathbf{A}_p^*\mathbf{y}_{t-p} + \mathbf{e}_t$$

$$\mathbf{y}_t = (\mathbf{I}_N - \mathbf{A}_0^*)^{-1}\mathbf{A}_1^*\mathbf{y}_{t-1} + \dots + (\mathbf{I}_N - \mathbf{A}_0^*)^{-1}\mathbf{A}_p^*\mathbf{y}_{t-p} + (\mathbf{I}_N - \mathbf{A}_0^*)^{-1}\mathbf{e}_t$$

Noting that $(\mathbf{I}_N - \mathbf{A}_0^*)^{-1} = \mathbf{W} = \mathbf{P}\mathbf{D}^{-1}$ shows that the instantaneous effects of one-standard deviation shocks (e_{it} 's of size one standard deviation) to the system are represented by the elements of

$$\mathbf{WD} = \mathbf{P} = \Theta_0$$

because the diagonal elements of \mathbf{D} are just standard deviations of the components of \mathbf{e}_t .



THE ORTHOGONALIZED IMPULSE-RESPONSE FUNCTION

The Θ_i may provide response functions that are quite different from the Ψ_i responses.

Note that $\Theta_0 = \mathbf{P}$ is lower triangular and some elements below the diagonal will be nonzero if Ω has nonzero off-diagonal elements.

For the investment/income/consumption example

$$\Theta_0 = \mathbf{P} = \begin{bmatrix} 1.5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0.5 & 0.7 \end{bmatrix}$$

indicates that an income (y_2) innovation has an immediate impact on consumption (y_3).



THE ORTHOGONALIZED IMPULSE-RESPONSE FUNCTION

If the white noise covariance matrix Ω contains zeros, some components of ϵ_t are contemporaneously uncorrelated. Suppose, for instance, that ϵ_{1t} is uncorrelated with $\epsilon_{it}, i = 2, \dots, N$.

In this case, $\mathbf{A} = \mathbf{W}^{-1}$ and thus \mathbf{A}_0^* has a block of zeros so that y_1 has no instantaneous effect on $y_i, i = 2, \dots, N$. In the example, investment has no instantaneous impact on income and consumption because

$$\Omega = \begin{bmatrix} 2.25 & 0 & 0 \\ 0 & 1 & 0.5 \\ 0 & 0.5 & 0.74 \end{bmatrix}$$

ϵ_{1t} is uncorrelated with ϵ_{2t} and ϵ_{3t} . This, of course, is reflected in the matrix of instantaneous effects Θ_0 (*impact multipliers*).



THE ORTHOGONALIZED IMPULSE-RESPONSE FUNCTION

The fact that Θ_0 is lower triangular shows that the ordering of the variables is of importance, that is, it is important which of the variables is called y_1 and which one is called y_2 and so on.

One problem with this type of impulse response analysis is that the ordering of the variables cannot be determined with statistical methods but has to be specified by the analyst. The ordering has to be such that the first variable is the only one with a potential immediate impact on all other variables.

The second variable may have an immediate impact on the last $N - 2$ components of \mathbf{y}_t but not on y_{1t} and so on. To establish such an ordering may be a quite difficult exercise in practice.



THE ORTHOGONALIZED IMPULSE-RESPONSE FUNCTION

The choice of the ordering, the Wold causal ordering, may, to a large extent, determine the impulse responses and is therefore critical for the interpretation of the system.

For the investment/income/consumption example it may be reasonable to assume that an increase in income has an immediate effect on consumption while increased consumption stimulates the economy and, hence, income with some time lag.



THE ORTHOGONALIZED IMPULSE-RESPONSE FUNCTION

Besides specifying the relevant impulses to a system, there are a number of further problems that render the interpretation of impulse responses difficult.

A major limitation of our systems is their potential incompleteness. Although in real economic systems almost everything depends on everything else, we will usually work with low-dimensional VAR systems.

All effects of omitted variables are assumed to be in the innovations. If important variables are omitted from the system, this may lead to major distortions in the impulse responses and makes them worthless for structural interpretations.



THE ORTHOGONALIZED IMPULSE-RESPONSE FUNCTION

Consider a system \mathbf{y}_t which is partitioned in vectors \mathbf{z}_t and \mathbf{x}_t , with VMA representation

$$\mathbf{y}_t = \begin{bmatrix} \mathbf{z}_t \\ \mathbf{x}_t \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix} + \begin{bmatrix} \boldsymbol{\Psi}_{11}(L) & \boldsymbol{\Psi}_{12}(L) \\ \boldsymbol{\Psi}_{21}(L) & \boldsymbol{\Psi}_{22}(L) \end{bmatrix} \begin{bmatrix} \boldsymbol{\epsilon}_{1t} \\ \boldsymbol{\epsilon}_{2t} \end{bmatrix}$$

If the \mathbf{z}_t variables are considered only and the \mathbf{x}_t variables are omitted from the analysis, we get a prediction error MA representation:

$$\mathbf{z}_t = \boldsymbol{\mu}_1 + \sum_{i=0}^{\infty} \boldsymbol{\Psi}_{11,i} \boldsymbol{\epsilon}_{1t-i} + \sum_{i=1}^{\infty} \boldsymbol{\Psi}_{12,i} \boldsymbol{\epsilon}_{2t-i}$$

$$\mathbf{z}_t = \boldsymbol{\mu}_1 + \sum_{i=0}^{\infty} \mathbf{F}_i \mathbf{v}_{t-i}$$

The actual reactions of the \mathbf{z}_t components to innovations $\boldsymbol{\epsilon}_{1t}$ may be given by the $\boldsymbol{\Psi}_{11,i}$ matrices.



THE ORTHOGONALIZED IMPULSE-RESPONSE FUNCTION

On the other hand, the \mathbf{F}_i or corresponding orthogonalized impulse response are likely to be interpreted as impulse responses if the analyst does not realize that important variables have been omitted. The \mathbf{F}_i will be equal to the $\Psi_{11,i}$, if and only if \mathbf{x}_t does not Granger-cause \mathbf{z}_t .



The stable VAR(p) has a Wold MA representation:

$$\mathbf{y}_t = \boldsymbol{\epsilon}_t + \boldsymbol{\Psi}_1 \boldsymbol{\epsilon}_{t-1} + \boldsymbol{\Psi}_2 \boldsymbol{\epsilon}_{t-2} + \dots$$

where

$$\boldsymbol{\Psi}_s = \sum_{j=1}^s \boldsymbol{\Psi}_{s-j} \boldsymbol{\Phi}_j \quad s = 1, 2, \dots$$

with $\boldsymbol{\Psi}_0 = \mathbf{I}_K$. The elements of the $\boldsymbol{\Theta}_j$ are *the forecast error impulse responses*.

Different ways to orthogonalize the impulses. Cholesky decomposition of $\boldsymbol{\Omega}$. Such an approach is arbitrary, unless there are special reasons for a recursive structure.

Different ways to use nonsample information in specifying unique innovations and hence unique i-r.



Structural form model

$$\mathbf{A}\mathbf{y}_t = \Phi_1\mathbf{y}_{t-1} + \Phi_2\mathbf{y}_{t-2} + \dots + \Phi_p\mathbf{y}_{t-p} + \mathbf{B}\mathbf{e}_t$$

The matrix \mathbf{A} contains the instantaneous relations between the left-hand-side variables. It has to be invertible.



SVAR

The reduced forms are obtained by premultiplying with \mathbf{A}^{-1}

$$\mathbf{y}_t = \mathbf{A}^{-1}\Phi_1\mathbf{y}_{t-1} + \mathbf{A}^{-1}\Phi_2\mathbf{y}_{t-2} + \dots + \mathbf{A}^{-1}\Phi_p\mathbf{y}_{t-p} + \mathbf{A}^{-1}\mathbf{B}\mathbf{e}_t$$

$$\boldsymbol{\epsilon}_t = \mathbf{A}^{-1}\mathbf{B}\mathbf{e}_t$$

which relates the reduced-form disturbances $\boldsymbol{\epsilon}_t$ to the underlying structural shocks \mathbf{e}_t .



To identify the structural form parameters, we must place restrictions on the parameter matrices.

Even if $\mathbf{A} = \mathbf{I}_N$ the assumption of orthogonal shocks

$$E [\mathbf{e}_t \mathbf{e}'_t] = \mathbf{I}_N$$

is not sufficient to achieve identification.

For a N -dimensional system, $\frac{N(N-1)}{2}$ restrictions are necessary for orthogonalizing the shocks because there are $\frac{N(N-1)}{2}$ potentially different instantaneous covariances.

An example of such an identification scheme is the triangular (or recursive) identification suggested by Sims (1980). Such a scheme is also called *Wold causal chain system* and is often associated with a causal chain from the first to the last variable in the system.



IDENTIFICATION

Because the i-r functions computed from these models depend on the ordering of the variables, nonrecursive identification schemes that also allow for *instantaneous effects* of the variables

$$\mathbf{A} \neq \mathbf{I}_N$$

have been suggested in the literature (Sims (1986), Bernanke (1986)). Restrictions on the long-run effects of some shocks are also used to identify SVAR models (Blanchard and Quah (1989), Galí (1999), King, Plosser, Stock, and Watson (1991)).



IDENTIFICATION

$$\mathbf{A}\boldsymbol{\epsilon}_t = \mathbf{B}\mathbf{e}_t$$

The most popular kind of restrictions:

- $\mathbf{B} = \mathbf{I}_N$, the A-model.
- $\mathbf{A} = \mathbf{I}_N$, the innovation is $\boldsymbol{\epsilon}_t = \mathbf{B}\mathbf{e}_t$, the B-model.
- the AB-model
- Prior information.

Sutton said, “that’s where the money is”. Working for a startup is really hard. The best part is that there are no rules. The worst part, so far, is that we have no revenues, although we’re moving nicely to change this. Life is fun and it’s a great time to be doing statistical graphics.

I’m looking forward to seeing everyone in NYC this August. If you don’t see me, it means that I may have failed to solve our revenue problem. Please consider getting involved with section activities. The health of

our section, one of the largest in ASA, depends on volunteers.

Stephen G. Eick, Ph.D.
Co-founder and CTO Visintuit
eick@visintuit.com
630-778-0050



TOPICS IN STATISTICAL COMPUTING

An Introduction to the Bootstrap with Applications in R

A. C. Davison and Diego Kuonen

kuonen@statoo.com

Introduction

Bootstrap methods are resampling techniques for assessing uncertainty. They are useful when inference is to be based on a complex procedure for which theoretical results are unavailable or not useful for the sample sizes met in practice, where a standard model is suspect but it is unclear with what to replace it, or where a ‘quick and dirty’ answer is required. They can also be used to verify the usefulness of standard approximations for parametric models, and to improve them if they seem to give inadequate inferences. This article, a brief introduction on their use, is based closely on parts of Davison and Hinkley (1997), where further details and many examples and practicals can be found. A different point of view is given by Efron and Tibshirani (1993) and a more mathematical survey by Shao and Tu (1995), while Hall (1992) describes the underlying theory.

Basic Ideas

The simplest setting is when the observed data y_1, \dots, y_n are treated as a realisation of a random sample Y_1, \dots, Y_n from an unknown underlying distribution F . Interest is focused on a parameter θ , the outcome of applying the statistical functional $t(\cdot)$ to F , so $\theta = t(F)$. The simplest example of such a functional is the average, $t(F) = \int y dF(y)$; in general we think of $t(\cdot)$ as an algorithm to be applied to F .

The estimate of θ is $t = t(\hat{F})$, where \hat{F} is an estimate of F based on the data y_1, \dots, y_n . This might be a parametric model such as the normal, with parameters estimated by maximum likelihood or a more robust method,

or the empirical distribution function (EDF) \hat{F} , which puts mass n^{-1} on each of the y_j . If partial information is available about F , it may be injected into \hat{F} . However \hat{F} is obtained, our estimate t is simply the result of applying the algorithm $t(\cdot)$ to \hat{F} .

Typical issues now to be addressed are: what are bias and variance estimates for t ? What is a reliable confidence interval for θ ? Is a certain hypothesis consistent with the data? Hypothesis tests raise the issue of how the null hypothesis should be imposed, and are discussed in detail in Chapter 4 of Davison and Hinkley (1997). Here we focus on confidence intervals, which are reviewed in DiCiccio and Efron (1996), Davison and Hinkley (1997, Chapter 5) and Carpenter and Bithell (2000).

Confidence Intervals

The simplest approach to confidence interval construction uses normal approximation to the distribution of T , the random variable of which t is the observed value. If the true bias and variance of T are

$$\begin{aligned} b(F) &= E(T | F) - \theta = E(T | F) - t(F), \\ v(F) &= \text{var}(T | F), \end{aligned} \quad (1)$$

then we might hope that in large samples

$$Z = \frac{T - \theta - b(F)}{v(F)^{1/2}} \sim N(0, 1);$$

the conditioning in (1) indicates that T is based on a random sample Y_1, \dots, Y_n from F . In this case an approximate $(1 - 2\alpha)$ confidence interval for θ is

$$t - b(F) - z_{1-\alpha} v(F)^{1/2}, \quad t - b(F) - z_\alpha v(F)^{1/2}, \quad (2)$$

where z_α is the α quantile of the standard normal distribution. The adequacy of (2) depends on F , n , and T and cannot be taken for granted.

As it stands (2) is useless, because it depends on the unknown F . A key idea, sometimes called the *bootstrap* or *plug-in principle*, is to replace the unknown F with its known estimate \hat{F} , giving bias and variance estimates $b(\hat{F})$ and $v(\hat{F})$. For all but the simplest estimators T these cannot be obtained analytically and so

simulation is used. We generate R independent bootstrap samples Y_1^*, \dots, Y_n^* by sampling independently from \hat{F} , compute the corresponding estimator random variables T_1^*, \dots, T_R^* , and then hope that

$$b(F) \doteq b(\hat{F}) = E(T | \hat{F}) - t(\hat{F}) \quad (3)$$

$$\doteq R^{-1} \sum_{r=1}^R T_r^* - t = \bar{T}^* - t, \quad (4)$$

$$v(F) \doteq v(\hat{F}) = \text{var}(T | \hat{F}) \quad (5)$$

$$\doteq \frac{1}{R-1} \sum_{r=1}^R (T_r^* - \bar{T}^*)^2. \quad (6)$$

There are two errors here: statistical error due to replacement of F by \hat{F} , and simulation error from replacement of expectation and variance by averages. Evidently we must choose R large enough to make the second of these errors small relative to the first, and if possible use $b(\hat{F})$ and $v(\hat{F})$ in such a way that the statistical error, unavoidable in most situations, is minimized. This means using approximate pivots where possible.

If the normal approximation leading to (2) fails because the distribution of $T - \theta$ is not close to normal, an alternative approach to setting confidence intervals may be based on $T - \theta$. The idea is that if $T^* - t$ and $T - \theta$ have roughly the same distribution, then quantiles of the second may be estimated by simulating those of the first, giving $(1 - 2\alpha)$ basic bootstrap confidence limits

$$t - (T_{((R+1)(1-\alpha))}^* - t), \quad t - (T_{((R+1)\alpha)}^* - t),$$

where $T_{(1)}^* < \dots < T_{(R)}^*$ are the sorted T_r^* 's. When an approximate variance V for T is available and can be calculated from Y_1, \dots, Y_n , studentized bootstrap confidence intervals may be based on $Z = (T - \theta)/V^{1/2}$, whose quantiles are estimated from simulated values of the corresponding bootstrap quantity $Z^* = (T^* - t)/V^{1/2}$. This is justified by Edgeworth expansion arguments valid for many but not all statistics (Hall, 1992).

Unlike the intervals mentioned above, *percentile* and *bias-corrected adjusted* (BCa) intervals have the attractive property of invariance to transformations of the parameters. The percentile intervals with level $(1 - 2\alpha)$ is $(T_{((R+1)\alpha)}^*, T_{((R+1)(1-\alpha))}^*)$, while the BCa interval has form $(T_{((R+1)\alpha')}^*, T_{((R+1)(1-\alpha''))}^*)$, with α' and α'' cleverly chosen to improve the properties of the interval. DiCiccio and Efron (1996) describe the reasoning underlying these intervals and their developments.

The BCa and studentized intervals are second-order accurate. Numerical comparisons suggest that both tend to undercover, so the true probability that a 0.95 interval contains the true parameter is smaller than 0.95, and

that BCa intervals are shorter than studentized ones, so they undercover by slightly more.

Bootstrapping in R

R (Ihaka and Gentleman, 1996) is a language and environment for statistical computing and graphics. Additional details can be found at www.r-project.org. The two main packages for bootstrapping in R are `boot` and `bootstrap`. Both are available on the ‘Comprehensive R Archive Network’ (CRAN, cran.r-project.org) and accompany Davison and Hinkley (1997) and Efron and Tibshirani (1993) respectively. The package `boot`, written by Angelo Canty for use within S-Plus, was ported to R by Brian Ripley and is much more comprehensive than any of the current alternatives, including methods that the others do not include. After downloading the package from CRAN and installing the package, one simply has to type

```
require(boot)
```

at the R prompt. Note that the installation could also be performed within R by means of

```
install.packages("boot")
```

A good starting point is to carefully read the documentations of the R functions `boot` and `boot.ci`

```
?boot
```

```
?boot.ci
```

and to try out one of the examples given in the ‘Examples’ section of the corresponding help file. In what follows we illustrate their use.

Example

Figure 1 shows data from an experiment in which two laser treatments were randomized to eyes on patients. The response is visual acuity, measured by the number of letters correctly identified in a standard eye test. Some patients had only one suitable eye, and they received one treatment allocated at random. There are 20 patients with paired data and 20 patients for whom just one observation is available, so we have a mixture of paired comparison and two-sample data.

```
blue <- c(4, 69, 87, 35, 39, 79, 31, 79, 65, 95, 68,
       62, 70, 80, 84, 79, 66, 75, 59, 77, 36, 86,
       39, 85, 74, 72, 69, 85, 85, 72)
red <- c(62, 80, 82, 83, 0, 81, 28, 69, 48, 90, 63,
       77, 0, 55, 83, 85, 54, 72, 58, 68, 88, 83, 78,
       30, 58, 45, 78, 64, 87, 65)
acui<-data.frame(str=c(rep(0,20),
                      rep(1,10)), red, blue)
```

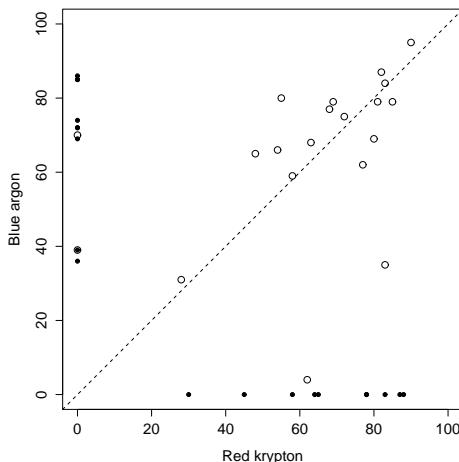


Figure 1: Paired (circles) and unpaired data (small blobs).

We denote the fully observed pairs $y_j = (r_j, b_j)$, the responses for the eyes treated with red and blue treatments, and for these n_d patients we let $d_j = b_j - r_j$. Individuals with just one observation give data $y_j = (?, b_j)$ or $y_j = (r_j, ?)$; there are n_b and n_r of these. The unknown variances of the d 's, r 's and b 's are σ_d^2 , σ_r^2 and σ_b^2 .

For illustration purposes, we will perform a standard analysis for each. First, we could only consider the paired data and construct the classical Student- t 0.95 confidence interval for the mean of the differences, of form $\bar{d} \pm t_{n-1}(0.025)s_d/n_d^{1/2}$, where $\bar{d} = 3.25$, s_d is the standard deviation of the d 's and $t_{n-1}(0.025)$ is the quantile of the appropriate t distribution. This can be done in R by means of

```
> acu.pd <- acui[acui$str==0,]
> dif <- acu.pd$blue-acu.pd$red
> n <- nrow(acu.pd)
> tmp<-qt(0.025,n-1)*sd(dif)/sqrt(n)
> c(mean(dif)+tmp, mean(dif)-tmp)
[1] -9.270335 15.770335
```

But a Q-Q plot of the differences looks more Cauchy than normal, so the usual model might be thought unreliable. The bootstrap can help to check this. To perform a nonparametric bootstrap in this case we first need to define the *bootstrap function*, corresponding to the algorithm $t(\cdot)$:

```
acu.pd.fun <- function(data, i){
  d <- data[i,]
  dif <- d$blue-d$red
  c(mean(dif), var(dif)/nrow(d)) }
```

A set of $R = 999$ bootstrap replicates can then be easily obtained with $acu.pd.b<-boot(acu.pd, acu.pd.fun, R=999)$. The result-

ing nonparametric 0.95 bootstrap confidence intervals can be calculated as shown previously or using directly

```
> boot.ci(acu.pd.b,
  type=c("norm", "basic", "stud"))
...
Normal          Basic          Studentized
(-8.20,14.95) (-8.10,15.05) (-8.66,15.77)
```

The normal Q-Q plot of the $R = 999$ replicates in the left panel of Figure 2 underlines the fact that the Student- t and the bootstrap intervals are essentially equal.

An alternative is to consider only the two-sample data and compare the means of the two populations issuing from the patients for whom just one observation is available, namely

```
acu.ts<- acui[acui$str==1,]
```

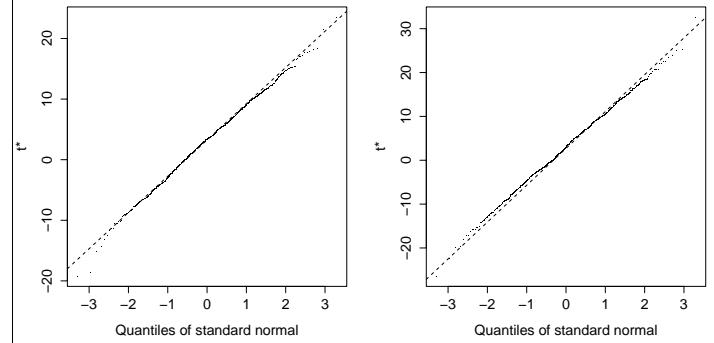


Figure 2: Normal Q-Q plots of bootstrap estimate t^* .

Left: for the paired analysis.
Right: for the two-sample analysis.

The classical normal 0.95 confidence interval for the difference of the means is $(\bar{b} - \bar{r}) \pm z_{0.025}(s_b^2/n_b + s_r^2/n_r)^{1/2}$, where s_b and s_r are the standard deviations of the b 's and r 's, and $z_{0.025}$ is the 0.025 quantile of the standard normal distribution.

```
> acu.ts <- acui[acui$str==1,]
> dif <- mean(acu.ts$blue)-mean(acu.ts$red)
> tmp <- qnorm(0.025)*
  sqrt(var(acu.ts$blue)/nrow(acu.ts) +
  var(acu.ts$red)/nrow(acu.ts))
> c(dif+tmp, dif-tmp)
[1] -13.76901 19.16901
```

The obvious estimator and its estimated variance are

$$t = \bar{b} - \bar{r}, \quad v = s_b^2/n_b + s_r^2/n_r,$$

whose values for these data are 2.7 and 70.6. To construct bootstrap confidence intervals we generate

$R = 999$ replicates of t and v , with each simulated dataset containing n_b values sampled with replacement from the bs and n_r values sampled with replacement from the rs . In R :

```
y<-c(acui$blue[21:30],acui$red[21:30])
acu<-data.frame(col=rep(c(1,2),c(10,10)),y)
acu.ts.f <- function(data, i){
d <- data[i,]
m <- mean(d$y[1:10])-mean(d$y[11:20])
v <- var(d$y[1:10])/10+var(d$y[11:20])/10
c(m, v) }
acu.ts.boot<-boot(acu,acu.ts.f,R=999,
strata=acu$col)
```

Here `strata=acu$col` ensures stratified simulation. The Q–Q plot of these 999 values in the right panel of Figure 2 is close to normal, and the bootstrap intervals computed using `boot.ci` differ little from the classical normal interval.

We now combine the analyses, hoping that the resulting confidence interval will be shorter. If the variances σ_d^2 , σ_r^2 and σ_b^2 of the ds , rs and bs were known, a minimum variance unbiased estimate of the difference between responses for blue and red treatments would be

$$\frac{n_d \bar{d} / \sigma_d^2 + (\bar{b} - \bar{r}) / (\sigma_b^2 / n_b + \sigma_r^2 / n_r)}{n_d / \sigma_d^2 + 1 / (\sigma_b^2 / n_b + \sigma_r^2 / n_r)}.$$

As σ_d^2 , σ_r^2 and σ_b^2 are unknown, we replace them by estimates, giving estimated treatment difference and its variance

$$t = \frac{n_d \bar{d} / \hat{\sigma}_d^2 + (\bar{b} - \bar{r}) / (\hat{\sigma}_b^2 / n_b + \hat{\sigma}_r^2 / n_r)}{n_d / \hat{\sigma}_d^2 + 1 / (\hat{\sigma}_b^2 / n_b + \hat{\sigma}_r^2 / n_r)},$$

$$v = \left\{ n_d / \hat{\sigma}_d^2 + 1 / (\hat{\sigma}_b^2 / n_b + \hat{\sigma}_r^2 / n_r) \right\}^{-1}.$$

Here $t = 3.07$ and $v = 4.873^2$, so a naive 0.95 confidence interval for the treatment difference is $(-6.48, 12.62)$.

One way to apply the bootstrap here is to generate a bootstrap dataset by taking n_d pairs randomly with replacement from \hat{F}_y , n_b values with replacement from \hat{F}_b and n_r values with replacement from \hat{F}_r , each resample being taken with equal probability:

```
acu.f <- function(data, i){
d <- data[i,]
m <- sum(data$str)
if(length(unique((i)==(1:nrow(data))))!=1) {
d$blue[d$str==1]<-sample(d$blue,size=m,T)
d$red[d$str==1]<-sample(d$red,size=m,T) }
dif<- d$blue[d$str==0]-d$red[d$str==0]
d2 <- d$blue[d$str==1]
d3 <- d$red[d$str==1]
```

```
v1 <- var(dif)/length(dif)
v2 <- var(d2)/length(d2)+var(d3)/length(d3)
v <- 1/(1/v1+1/v2)
c((mean(dif)/v1+(mean(d2)-mean(d3))/v2)*v,v)
acu.b<-boot(acui,acu.f,R=999,strata=acui$str)
boot.ci(acu.b,type=c("norm","basic","stud",
"perc","bca"))
```

giving all five sets of confidence limits. The interested reader can continue the analysis.

Regression

A linear regression model has form $y_j = x_j^\top \beta + \varepsilon_j$, where the (y_j, x_j) are the response and the $p \times 1$ vector of covariates for the j th response y_j . We are usually interested in confidence intervals for the parameters, the choice of covariates, or prediction of the future response y_+ at a new covariate x_+ . The two basic resampling schemes for regression models are

- *resampling cases* $(y_1, x_1), \dots, (y_n, x_n)$, under which the bootstrap data are

$$(y_1, x_1)^*, \dots, (y_n, x_n)^*,$$

taken independently with equal probabilities n^{-1} from the (y_j, x_j) , and

- *resampling residuals*. Having obtained fitted values $x_j^\top \hat{\beta}$, we take ε_j^* randomly from centred standardized residuals e_1, \dots, e_n and set

$$y_j^* = x_j^\top \hat{\beta} + \varepsilon_j^*, \quad j = 1, \dots, n.$$

Under case resampling the resampled design matrix does not equal the original one. For moderately large data sets this doesn't matter, but it can be worth bearing in mind if n is small or if a few observations have a strong influence on some aspect of the design. If the wrong model is fitted and this scheme is used we get an appropriate measure of uncertainty, so case resampling is in this sense robust. The second scheme is more efficient than resampling pairs if the model is correct, but is not robust to getting the wrong model, so careful model-checking is needed before it can be used. Either scheme can be stratified if the data are inhomogeneous. In the most extreme form of stratification the strata consist of just one residual; this is the *wild bootstrap*, used in non-parametric regressions.

Variants of residual resampling needed for generalized linear models, survival data and so forth are all constructed essentially by looking for the exchangeable aspects of the model, estimating them, and then resampling them. Similar ideas also apply to time series models such as ARMA processes. Additional examples and further details can be found in Davison and Hinkley

(1997, Chapters 6–8). We now illustrate case and residual resampling.

The survival data (Efron, 1988) are survival percentages for rats at a succession of doses of radiation, with two or three replicates at each dose; see Figure 3. The data come with the package `boot` and can be loaded using

```
> data(survival)
```

To have a look at the data, simply type `survival` at the R prompt. The theoretical relationship between survival rate (`surv`) and dose (`dose`) is exponential, so linear regression applies to

$$x = \text{dose}, \quad y = \log(\text{surv}).$$

There is a clear outlier, case 13, at $x = 1410$. The least squares estimate of slope is -59×10^{-4} using all the data, changing to -78×10^{-4} with standard error 5.4×10^{-4} when case 13 is omitted.

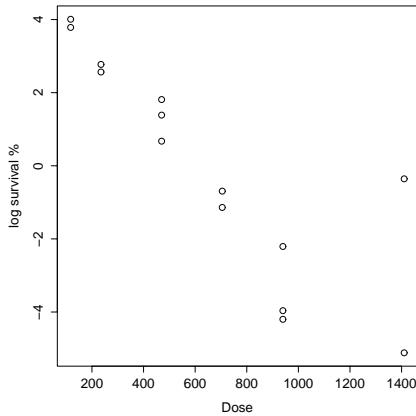


Figure 3: Scatter plot of survival data.

To illustrate the potential effect of an outlier in regression we resample cases, using

```
surv.fun <- function(data, i) {
  d <- data[i,]
  d.reg <- lm(log(d$surv) ~ d$dose)
  c(coef(d.reg)) }
surv.boot<-boot(survival,surv.fun,R=999)
```

The effect of the outlier on the resampled estimates is shown in Figure 4, a histogram of the $R = 999$ bootstrap least squares slopes $\hat{\beta}_1^*$. The two groups of bootstrapped slopes correspond to resamples in which case 13 does not occur and to samples where it occurs once or more. The resampling standard error of $\hat{\beta}_1^*$ is 15.6×10^{-4} , but only 7.8×10^{-4} for samples without case 13.

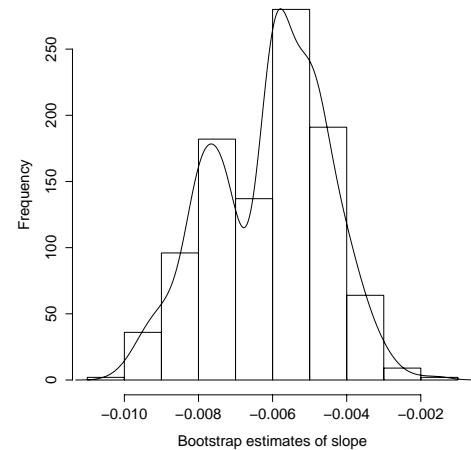


Figure 4: Histogram of bootstrap estimates of slope $\hat{\beta}_1^*$ with superposed kernel density estimate.

A jackknife-after-bootstrap plot (Efron, 1992; Davison and Hinkley, 1997, Section 3.10.1) shows the effect on $T^* - t$ of resampling from datasets from which each of the observations has been removed. Here we expect deletion of case 13 to have a strong effect, and Figure 5 obtained through

```
> jack.after.boot(surv.boot, index=2)
```

shows clearly that this case has an appreciable effect on the resampling distribution, and that its omission would give much tighter confidence limits on the slope.

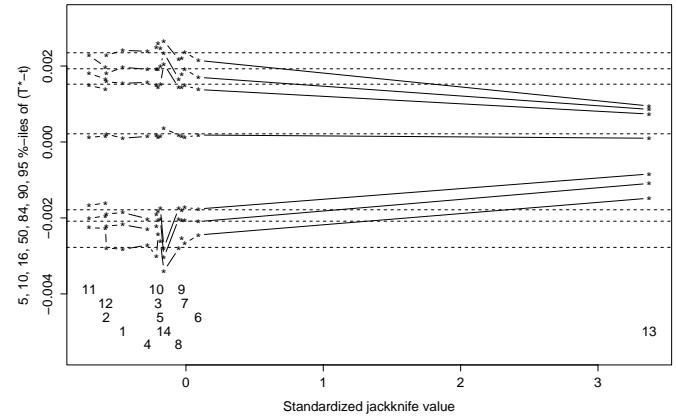


Figure 5: Jackknife-after-bootstrap plot for the slope. The vertical axis shows quantiles of $T^* - t$ for the full sample (horizontal dotted lines) and without each observation in turn, plotted against the influence value for that observation.

The effect of this outlier on the intercept and slope when resampling residuals can be assessed using

`sim=parametric` in the `boot` call. The required R code is:

```
fit <- lm(log(survival$surv) ~ survival$dose)
res <- resid(fit)
f <- fitted(fit)
surv.r.mle<- data.frame(f,res)
surv.r.fun<-function(data)
  coef(lm(log(data$surv) ~ data$dose))
surv.r.sim <- function(data, mle) {
  data$surv<-exp(mle$f+sample(mle$res,T))
  data
}
surv.r.boot<- boot(survival,surv.r.fun,
  R=999,sim="parametric",
  ran.gen=surv.r.sim,mle=surv.r.mle)
```

Having understood what this code does, the interested reader may use it to continue the analysis.

Discussion

Bootstrap resampling allows empirical assessment of standard approximations, and may indicate ways to fix them when they fail. The computer time involved is typically negligible — the resampling for this article took far less than the time needed to examine the data, devise plots and summary statistics, and to code (and check) the simulations.

Bootstrap methods offer considerable potential for modelling in complex problems, not least because they enable the choice of estimator to be separated from the assumptions under which its properties are to be assessed. In principle the estimator chosen should be appropriate to the model used, or there is a loss of efficiency. In practice, however, there is often some doubt about the exact error structure, and a well-chosen resampling scheme can give inferences robust to precise assumptions about the data.

Although the bootstrap is sometimes touted as a replacement for ‘traditional statistics’, we believe this to be misguided. It is unwise to use a powerful tool without understanding why it works, and the bootstrap rests on ‘traditional’ ideas, even if their implementation via simulation is not ‘traditional’. Populations, parameters, samples, sampling variation, pivots and confidence lim-

its are fundamental statistical notions, and it does no-one a service to brush them under the carpet. Indeed, it is harmful to pretend that mere computation can replace thought about central issues such as the structure of a problem, the type of answer required, the sampling design and data quality. Moreover, as with any simulation experiment, it is essential to monitor the output to ensure that no unanticipated complications have arisen and to check that the results make sense, and this entails understanding how the output will be used. Never forget: *the aim of computing is insight, not numbers; garbage in, garbage out.*

References

- Carpenter, J. and Bithell, J.** (2000). Bootstrap confidence intervals: when, which, what? A practical guide for medical statisticians. *Statistics in Medicine*, **19**, 1141–1164.
- Davison, A. C. and Hinkley, D. V.** (1997). *Bootstrap Methods and their Application*. Cambridge University Press.
(statwww.epfl.ch/davison/BMA/)
- DiCiccio, T. J. and Efron, B.** (1996). Bootstrap confidence intervals (with Discussion). *Statistical Science*, **11**, 189–228.
- Efron, B.** (1988). Computer-intensive methods in statistical regression. *SIAM Review*, **30**, 421–449.
- Efron, B.** (1992). Jackknife-after-bootstrap standard errors and influence functions (with Discussion). *Journal of the Royal Statistical Society series B*, **54**, 83–127.
- Efron, B. and Tibshirani, R. J.** (1993). *An Introduction to the Bootstrap*. New York: Chapman & Hall.
- Hall, P.** (1992). *The Bootstrap and Edgeworth Expansion*. New York: Springer-Verlag.
- Ihaka, R. and Gentleman, R.** (1996). R: a language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, **5**, 299–314.
- Shao, J. and Tu, D.** (1995). *The Jackknife and Bootstrap*. New York: Springer-Verlag.

SOFTWARE PACKAGES

GGobi

Deborah F. Swayne, AT&T Labs – Research

dfs@research.att.com

GGobi is a new interactive and dynamic software sys-

tem for data visualization, the result of a significant redesign of the older XGobi system (Swayne, Cook and Buja, 1992; Swayne, Cook and Buja, 1998), whose development spanned roughly the past decade. GGobi differs from XGobi in many ways, and it is those differences that explain best why we have undertaken this redesign.

Package ‘mfx’

February 20, 2015

Type Package

Title Marginal Effects, Odds Ratios and Incidence Rate Ratios for GLMs

Version 1.1

Date 2013-12-12

Author Alan Fernihough

Maintainer Alan Fernihough <alan.fernighough@gmail.com>

Description Estimates probit, logit, Poisson, negative binomial, and beta regression models, returning their marginal effects, odds ratios, or incidence rate ratios as an output.

License GPL-2 | GPL-3

Depends stats, sandwich, lmtest, MASS, betareg

NeedsCompilation no

Repository CRAN

Date/Publication 2014-01-19 20:45:05

R topics documented:

betamfx	2
betaor	3
logitmfx	4
logitor	6
negbinirr	7
negbinmfx	8
poissonirr	9
poissonmfx	11
probitmfx	12

Index

14

betamfx*Marginal effects for a beta regression.*

Description

This function estimates a beta regression model and calculates the corresponding marginal effects.

Usage

```
betamfx(formula, data, atmean = TRUE, robust = FALSE,
        clustervar1 = NULL, clustervar2 = NULL,
        control = betareg.control(), link.phi = NULL, type = "ML")
```

Arguments

<code>formula</code>	an object of class “formula” (or one that can be coerced to that class).
<code>data</code>	the data frame containing these data. This argument must be used.
<code>atmean</code>	default marginal effects represent the partial effects for the average observation. If <code>atmean = FALSE</code> the function calculates average partial effects.
<code>robust</code>	if <code>TRUE</code> the function reports White/robust standard errors.
<code>clustervar1</code>	a character value naming the first cluster on which to adjust the standard errors.
<code>clustervar2</code>	a character value naming the second cluster on which to adjust the standard errors for two-way clustering.
<code>control</code>	a list of control arguments specified via betareg.control .
<code>link.phi</code>	as in the betareg function.
<code>type</code>	as in the betareg function.

Details

The underlying link function in the mean model (`mu`) is “logit”. If both `robust=TRUE` and `!is.null(clustervar1)` the function overrides the `robust` command and computes clustered standard errors.

Value

<code>mfxest</code>	a coefficient matrix with columns containing the estimates, associated standard errors, test statistics and p-values.
<code>fit</code>	the fitted betareg object.
<code>dcvar</code>	a character vector containing the variable names where the marginal effect refers to the impact of a discrete change on the outcome. For example, a factor variable.
<code>call</code>	the matched call.

References

Francisco Cribari-Neto, Achim Zeileis (2010). Beta Regression in R. *Journal of Statistical Software* 34(2), 1-24.

Bettina Gruen, Ioannis Kosmidis, Achim Zeileis (2012). Extended Beta Regression in R: Shaken, Stirred, Mixed, and Partitioned. *Journal of Statistical Software*, 48(11), 1-25.

See Also

[betaor](#), [betareg](#)

Examples

```
# simulate some data
set.seed(12345)
n = 1000
x = rnorm(n)

# beta outcome
y = rbeta(n, shape1 = plogis(1 + 0.5 * x), shape2 = (abs(0.2*x)))
# use Smithson and Verkuilen correction
y = (y*(n-1)+0.5)/n

data = data.frame(y,x)
betamfx(y~x|x, data=data)
```

betaor

Odds ratios for a beta regression.

Description

This function estimates a beta regression model and calculates the corresponding odds ratios.

Usage

```
betaor(formula, data, robust = FALSE, clustervar1 = NULL, clustervar2 = NULL,
       control = betareg.control(), link.phi = NULL, type = "ML")
```

Arguments

formula	an object of class “formula” (or one that can be coerced to that class).
data	the data frame containing these data. This argument must be used.
robust	if TRUE the function reports White/robust standard errors.
clustervar1	a character value naming the first cluster on which to adjust the standard errors.
clustervar2	a character value naming the second cluster on which to adjust the standard errors for two-way clustering.
control	a list of control arguments specified via betareg.control .
link.phi	as in the betareg function.
type	as in the betareg function.

Details

The underlying link function in the mean model (`mu`) is "logit". If both `robust=TRUE` and `!is.null(clustervar1)` the function overrides the `robust` command and computes clustered standard errors.

Value

<code>oddsratio</code>	a coefficient matrix with columns containing the estimates, associated standard errors, test statistics and p-values.
<code>fit</code>	the fitted betareg object.
<code>call</code>	the matched call.

References

Francisco Cribari-Neto, Achim Zeileis (2010). Beta Regression in R. *Journal of Statistical Software* 34(2), 1-24.

Bettina Gruen, Ioannis Kosmidis, Achim Zeileis (2012). Extended Beta Regression in R: Shaken, Stirred, Mixed, and Partitioned. *Journal of Statistical Software*, 48(11), 1-25.

See Also

[betamfx](#), [betareg](#)

Examples

```
# simulate some data
set.seed(12345)
n = 1000
x = rnorm(n)

# beta outcome
y = rbeta(n, shape1 = plogis(1 + 0.5 * x), shape2 = (abs(0.2*x)))
# use Smithson and Verkuilen correction
y = (y*(n-1)+0.5)/n

data = data.frame(y,x)
betaor(y~x|x, data=data)
```

logitmfx

Marginal effects for a logit regression.

Description

This function estimates a binary logistic regression model and calculates the corresponding marginal effects.

Usage

```
logitmfx(formula, data, atmean = TRUE, robust = FALSE, clustervar1 = NULL,
          clustervar2 = NULL, start = NULL, control = list())
```

Arguments

formula	an object of class “formula” (or one that can be coerced to that class).
data	the data frame containing these data. This argument must be used.
atmean	default marginal effects represent the partial effects for the average observation. If atmean = FALSE the function calculates average partial effects.
robust	if TRUE the function reports White/robust standard errors.
clustervar1	a character value naming the first cluster on which to adjust the standard errors.
clustervar2	a character value naming the second cluster on which to adjust the standard errors for two-way clustering.
start	starting values for the parameters in the glm model.
control	see glm.control .

Details

If both robust=TRUE and !is.null(clustervar1) the function overrides the robust command and computes clustered standard errors.

Value

mfest	a coefficient matrix with columns containing the estimates, associated standard errors, test statistics and p-values.
fit	the fitted glm object.
dcvar	a character vector containing the variable names where the marginal effect refers to the impact of a discrete change on the outcome. For example, a factor variable.
call	the matched call.

References

William H. Greene (2008). Econometric Analysis (6th ed.). Prentice Hall, N.Y. pp 770-787.

See Also

[logitor](#), [glm](#)

Examples

```
# simulate some data
set.seed(12345)
n = 1000
x = rnorm(n)

# binary outcome
y = ifelse(pnorm(1 + 0.5*x + rnorm(n))>0.5, 1, 0)

data = data.frame(y,x)
logitmfx(formula=y~x, data=data)
```

logitor

Odds ratios for a logit regression.

Description

This function estimates a binary logistic regression model and calculates the corresponding odds ratios.

Usage

```
logitor(formula, data, robust = FALSE, clustervar1 = NULL, clustervar2 = NULL,
        start = NULL, control = list())
```

Arguments

formula	an object of class “formula” (or one that can be coerced to that class).
data	the data frame containing these data. This argument must be used.
robust	if TRUE the function reports White/robust standard errors.
clustervar1	a character value naming the first cluster on which to adjust the standard errors.
clustervar2	a character value naming the second cluster on which to adjust the standard errors for two-way clustering.
start	starting values for the parameters in the glm model.
control	see glm.control .

Details

If both `robust=TRUE` and `!is.null(clustervar1)` the function overrides the `robust` command and computes clustered standard errors.

Value

oddsratio	a coefficient matrix with columns containing the estimates, associated standard errors, test statistics and p-values.
fit	the fitted glm object.
call	the matched call.

See Also

[logitmfx](#), [glm](#)

Examples

```
# simulate some data
set.seed(12345)
n = 1000
x = rnorm(n)

# binary outcome
y = ifelse(rnorm(1 + 0.5*x + rnorm(n))>0.5, 1, 0)

data = data.frame(y,x)
logitor(formula=y~x, data=data)
```

negbinirr

Incidence rate ratios for a negative binomial regression.

Description

This function estimates a negative binomial regression model and calculates the corresponding incidence rate ratios.

Usage

```
negbinirr(formula, data, robust = FALSE, clustervar1 = NULL,
          clustervar2 = NULL, start = NULL, control = glm.control())
```

Arguments

formula	an object of class “formula” (or one that can be coerced to that class).
data	the data frame containing these data. This argument must be used.
robust	if TRUE the function reports White/robust standard errors.
clustervar1	a character value naming the first cluster on which to adjust the standard errors.
clustervar2	a character value naming the second cluster on which to adjust the standard errors for two-way clustering.
start	starting values for the parameters in the glm.nb model.
control	see glm.control .

Details

If both `robust=TRUE` and `!is.null(clustervar1)` the function overrides the `robust` command and computes clustered standard errors.

Value

- irr** a coefficient matrix with columns containing the estimates, associated standard errors, test statistics and p-values.
fit the fitted [glm.nb](#) object.
call the matched call.

See Also

[negbinmfx](#), [glm.nb](#)

Examples

```
# simulate some data
set.seed(12345)
n = 1000
x = rnorm(n)
y = rnbinom(n, mu = exp(1 + 0.5 * x), theta = 0.5)

data = data.frame(y,x)

negbinirr(formula=y~x,data=data)
```

negbinmfx

Marginal effects for a negative binomial regression.

Description

This function estimates a negative binomial regression model and calculates the corresponding marginal effects.

Usage

```
negbinmfx(formula, data, atmean = TRUE, robust = FALSE, clustervar1 = NULL,
          clustervar2 = NULL, start = NULL, control = glm.control())
```

Arguments

- formula** an object of class “formula” (or one that can be coerced to that class).
data the data frame containing these data. This argument must be used.
atmean default marginal effects represent the partial effects for the average observation.
If **atmean** = FALSE the function calculates average partial effects.
robust if TRUE the function reports White/robust standard errors.
clustervar1 a character value naming the first cluster on which to adjust the standard errors.
clustervar2 a character value naming the second cluster on which to adjust the standard errors for two-way clustering.
start starting values for the parameters in the [glm.nb](#) model.
control see [glm.control](#).

Details

If both `robust=TRUE` and `!is.null(clustervar1)` the function overrides the `robust` command and computes clustered standard errors.

Value

<code>mfest</code>	a coefficient matrix with columns containing the estimates, associated standard errors, test statistics and p-values.
<code>fit</code>	the fitted <code>glm.nb</code> object.
<code>dcvar</code>	a character vector containing the variable names where the marginal effect refers to the impact of a discrete change on the outcome. For example, a factor variable.
<code>call</code>	the matched call.

See Also

`negbinirr, glm.nb`

Examples

```
# simulate some data
set.seed(12345)
n = 1000
x = rnorm(n)
y = rnbinom(n, mu = exp(1 + 0.5 * x), theta = 0.5)

data = data.frame(y,x)

negbinmfx(formula=y~x,data=data)
```

`poissonirr`

Incidence rate ratios for a Poisson regression.

Description

This function estimates a negative binomial regression model and calculates the corresponding incidence rate ratios.

Usage

```
poissonirr(formula, data, robust = FALSE, clustervar1 = NULL,
           clustervar2 = NULL, start = NULL, control = list())
```

Arguments

<code>formula</code>	an object of class “formula” (or one that can be coerced to that class).
<code>data</code>	the data frame containing these data. This argument must be used.
<code>robust</code>	if TRUE the function reports White/robust standard errors.
<code>clustervar1</code>	a character value naming the first cluster on which to adjust the standard errors.
<code>clustervar2</code>	a character value naming the second cluster on which to adjust the standard errors for two-way clustering.
<code>start</code>	starting values for the parameters in the <code>glm</code> model.
<code>control</code>	see <code>glm.control</code> .

Details

If both `robust=TRUE` and `!is.null(clustervar1)` the function overrides the `robust` command and computes clustered standard errors.

Value

<code>irr</code>	a coefficient matrix with columns containing the estimates, associated standard errors, test statistics and p-values.
<code>fit</code>	the fitted <code>glm</code> object.
<code>call</code>	the matched call.

See Also

`poissonmfx`, `glm`

Examples

```
# simulate some data
set.seed(12345)
n = 1000
x = rnorm(n)
y = rnbinom(n, mu = exp(1 + 0.5 * x), theta = 0.5)

data = data.frame(y,x)

poissonirr(formula=y~x,data=data)
```

`poissonmfx`

Marginal effects for a Poisson regression.

Description

This function estimates a Poisson regression model and calculates the corresponding marginal effects.

Usage

```
poissonmfx(formula, data, atmean = TRUE, robust = FALSE, clustervar1 = NULL,
           clustervar2 = NULL, start = NULL, control = list())
```

Arguments

<code>formula</code>	an object of class “formula” (or one that can be coerced to that class).
<code>data</code>	the data frame containing these data. This argument must be used.
<code>atmean</code>	default marginal effects represent the partial effects for the average observation. If <code>atmean</code> = FALSE the function calculates average partial effects.
<code>robust</code>	if TRUE the function reports White/robust standard errors.
<code>clustervar1</code>	a character value naming the first cluster on which to adjust the standard errors.
<code>clustervar2</code>	a character value naming the second cluster on which to adjust the standard errors for two-way clustering.
<code>start</code>	starting values for the parameters in the <code>glm</code> model.
<code>control</code>	see <code>glm.control</code> .

Details

If both `robust=TRUE` and `!is.null(clustervar1)` the function overrides the `robust` command and computes clustered standard errors.

Value

<code>mfest</code>	a coefficient matrix with columns containing the estimates, associated standard errors, test statistics and p-values.
<code>fit</code>	the fitted <code>glm</code> object.
<code>dcvar</code>	a character vector containing the variable names where the marginal effect refers to the impact of a discrete change on the outcome. For example, a factor variable.
<code>call</code>	the matched call.

See Also

`poissonirr`, `glm`

Examples

```
# simulate some data
set.seed(12345)
n = 1000
x = rnorm(n)
y = rnbinom(n, mu = exp(1 + 0.5 * x), theta = 0.5)

data = data.frame(y,x)

poissonmfx(formula=y~x,data=data)
```

probitmfx

Marginal effects for a probit regression.

Description

This function estimates a probit regression model and calculates the corresponding marginal effects.

Usage

```
probitmfx(formula, data, atmean = TRUE, robust = FALSE, clustervar1 = NULL,
           clustervar2 = NULL, start = NULL, control = list())
```

Arguments

- | | |
|--------------------|--|
| formula | an object of class “formula” (or one that can be coerced to that class). |
| data | the data frame containing these data. This argument must be used. |
| atmean | default marginal effects represent the partial effects for the average observation.
If atmean = FALSE the function calculates average partial effects. |
| robust | if TRUE the function reports White/robust standard errors. |
| clustervar1 | a character value naming the first cluster on which to adjust the standard errors. |
| clustervar2 | a character value naming the second cluster on which to adjust the standard errors for two-way clustering. |
| start | starting values for the parameters in the glm model. |
| control | see glm.control . |

Details

If both **robust**=TRUE and !is.null(**clustervar1**) the function overrides the **robust** command and computes clustered standard errors.

Value

mfest	a coefficient matrix with columns containing the estimates, associated standard errors, test statistics and p-values.
fit	the fitted glm object.
dcvar	a character vector containing the variable names where the marginal effect refers to the impact of a discrete change on the outcome. For example, a factor variable.
call	the matched call.

References

William H. Greene (2008). Econometric Analysis (6th ed.). Prentice Hall, N.Y. pp 770-787.

See Also

[glm](#)

Examples

```
# simulate some data
set.seed(12345)
n = 1000
x = rnorm(n)

# binary outcome
y = ifelse(pnorm(1 + 0.5*x + rnorm(n))>0.5, 1, 0)

data = data.frame(y,x)
probitmfx(formula=y~x, data=data)
```

Index

betamfx, 2, 4
betaor, 3, 3
betareg, 2–4
betareg.control, 2, 3

glm, 5–7, 10–13
glm.control, 5–8, 10–12
glm.nb, 7–9

logitmfx, 4, 7
logitor, 5, 6

negbinirr, 7, 9
negbinmfx, 8, 8

poissonirr, 9, 11
poissonmfx, 10, 11
print.betamfx (betamfx), 2
print.betaor (betaor), 3
print.logitmfx (logitmfx), 4
print.logitor (logitor), 6
print.negbinirr (negbinirr), 7
print.negbinmfx (negbinmfx), 8
print.poissonirr (poissonirr), 9
print.poissonmfx (poissonmfx), 11
print.probitmfx (probitmfx), 12
probitmfx, 12