6 GETS modelling in logit regressions

Logit models can be estimated with the `glm()` function (part of base R). However, the `glm()` function does not offer dynamic logit specifications or robust standard errors. Also, its numerical optimisation routine is neither flexible nor numerically robust. The package **gets** thus comes with its own logit function, `logitx()`, which offers:

- The estimation of static logit models
- The estimation of dynamic logit models with covariates ("X")
- Robust coefficient standard errors of the Newey and West (1987) type (optional), in addition to ordinary standard errors (default)
- Flexible and numerically robust estimation based on `nlminb()` (part of base R)

6.1 The Autoregressive (AR) logit-X model

At its most general, the specifications admitted by `logitx()` are contained in a modified version of the dynamic binary model considered by Kauppi and Saikkonen (2008):

$$Pr(y_t = 1|\boldsymbol{z}_t) = \frac{1}{1+\exp(-h_t)}, \qquad y_t \in \{0,1\}, \qquad t = 1,\ldots,n, \tag{6.1}$$

$$h_t = \alpha_0 + \underbrace{\sum_{p=1}^{P} \alpha_p y_{t-p}}_{\texttt{ar}} + \underbrace{\sum_{q\in Q} \beta_q \text{EqWMA}_{q,t-1}}_{\texttt{ewma}} + \underbrace{\sum_{d=1}^{D} \delta_d x_{d,t}}_{\texttt{xreg}}, \tag{6.2}$$

The model is a modification of Kauppi and Saikkonen (2008) in two ways. First, it does not allow for lagged h_t's. Second, it explicitly allows for lagged Equally Weighted Moving Average (EqWMA) terms, which can be interpreted as observable proxies for the omitted lagged h_t's. The $\boldsymbol{z}_t$ in $Pr(y_t = 1|\boldsymbol{z}_t)$ is shorthand notation for all the conditioning variables.

The standard logit model is obtained by removing the `ar` and `ewma` terms:

$$h_t = \alpha_0 + \delta_1 x_{1t} + \cdots + \delta_k x_{kt}.$$

The term labelled `ar` is analogous to AR-terms in the AR-X model of Chapter 3. The AR-terms enable the study of whether past values of y_t's help predict the probability that $y_t = 1$. The EqWMA terms, where $\text{EqWMA}_{q,t-1} = (y_{t-1} + \cdots + y_{t-q})/q$, offer great flexibility in modelling the persistence of the conditional probability $Pr(y_t = 1|\boldsymbol{z}_t)$. For example, if y_t is observed daily, then $\text{EqWMA}_{7,t-1}$, $\text{EqWMA}_{28,t-1}$ and $\text{EqWMA}_{84,t-1}$ can be interpreted as slowly changing "weekly", "monthly" and "quarterly" estimates, respectively, of the conditional probability $Pr(y_t = 1|\boldsymbol{z}_t)$. When $q = 1$, then $\text{EqWMA}_{q,t-1} = y_{t-1}$.

6.2 `logitx()`: Estimation

The function `logitxSim()` can be used to simulate from the model given by (6.1) – (6.2). For example, the following code simulates from a standard specification with $h_t = 0.2 + 0.5x_t$, where x_t is an iid normal variable:

```
set.seed(123) #for reproducibility
x <- rnorm(100)
y <- logitxSim(100, intercept=0.2, xreg=0.5*x)
```

Next, the following code estimates the model, stores the result in `"mymodel"` and prints the estimation result:

```
mymodel <- logitx(y, xreg=x)
mymodel

Date: Thu Dec 23 19:42:28 2021
Dependent var.: y
Method: Maximum Likelihood (logit)
Variance-Covariance: Ordinary
No. of observations: 100
Sample: 1 to 100

Estimation results:

              coef std.error t-stat p-value
intercept 0.084731  0.204923 0.4135 0.34008
mxreg     0.448885  0.233464 1.9227 0.02871 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log-lik.(n=100) -67.1797
```

To plot the fitted probabilities, type `plot(mymodel)`. To instead plot the 0-probabilities, set `zero.prob = TRUE`.

If we would like to fit an AR(2)-logit-X specification using the same data, the following code can be used:

```
logitx(y, ar=c(1,2), xreg=x)

Date: Fri Dec 24 11:07:55 2021
Dependent var.: y
Method: Maximum Likelihood (logit)
Variance-Covariance: Ordinary
No. of observations: 98
Sample: 3 to 100

Estimation results:

               coef std.error  t-stat p-value
intercept -0.137570  0.377862 -0.3641 0.35831
ar1        0.323684  0.418421  0.7736 0.22056
ar2        0.089038  0.428414  0.2078 0.41790
mxreg      0.474419  0.244072  1.9438 0.02746 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Log-lik.(n=98) -65.5343
```

Next, to add the lagged 7-period moving average and the lagged 14-period moving average as predictors, then this is achieved by the following code:

```
logitx(y, ar=c(1,2), ewma=list(length=c(7,14)), xreg=x)

Date: Fri Dec 24 11:08:21 2021
Dependent var.: y
Method: Maximum Likelihood (logit)
Variance-Covariance: Ordinary
No. of observations: 86
Sample: 15 to 100

Estimation results:
```

```
              coef std.error  t-stat p-value
intercept -0.14870   0.80665 -0.1843 0.42710
ar1        0.17701   0.50290  0.3520 0.36289
ar2       -0.18615   0.53410 -0.3485 0.36418
EqWMA(7)   1.30772   1.84856  0.7074 0.24068
EqWMA(14) -0.71316   2.13436 -0.3341 0.36958
mxreg      0.35613   0.25688  1.3864 0.08475 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Log-lik.(n=86) -57.7102
```

To extract specific items of from a stored estimation results (e.g. `mymodel` above), there is a variety of methods (S3) available for models of the `"logitx"` class, including

`coef(), fitted(), gets(), logLik(), plot(), print(), summary(), toLatex(), vcov()`

Maximum Likelihood (ML) estimation of (6.1) – (6.2) is undertaken with the `nlminb()` optimisation routine (part of base R). For the exact details, see the code of the "intermediate" function `logit()`. The `nlminb()` routine is numerically robust. It is also very flexible, since the parameter values can be constrained during estimation. When using the `logitx()` function, the call to `nlminb()` can be controlled and tuned via the arguments `initial.values`, `lower`, `upper` and `control`. In `nlminb()`, the first argument is named `start`, whereas the other three are named the same. Suitable initial parameter values are important for numerical robustness. By default, `logitx()` sets these automatically internally. However, if the user wishes to do so, they can instead be specified via the `initial.values` argument. In the estimation result (a list) returned by `logitx()`, the item named `initial.values` contains the initial parameter values used. During estimation, `nlminb()` calls an objective function – the logit log-likelihood – in each iteration. For additional numerical robustness, checks of the parameters are conducted at each iteration within the objective function, see the code of `logit()` for the details. Another potential source of numerical problems are fitted zero-probabilities too close to zero (due to the application of the logarithm on the fitted zero probability in the objective function). To avoid values that are too close to zero, the value in the `eps.tol` is used as the lower bound of the fitted zero probability. Finally, the inverse of the estimated Hessian – which is needed to compute the coefficient covariance – is computed with `solve()`, whose tolerance for detecting linear dependencies in the columns is given by the `solve.tol` argument in the `logitx()` function.

6.3 Example: Are positive stock returns predictable?

It is well-known that daily stock market returns are difficult to predict. Here, we explore whether the *probability* of a positive return can be predicted. For this illustration, we use daily data of the Standard and Poor's 500 (SP500) index:

```
data("sp500data", package = "gets")
sp500data <- zoo(sp500data[, -1],
  order.by = as.Date(sp500data[, "Date"]))
sp500data <- window(sp500data, start = as.Date("1983-07-01"))
```

The dataset contains the daily value of the SP500 index, its highs and lows, and the daily volume. The code `[, -1]` removes an unneeded column, and the last line shortens the sample, since not all variables are available from the start. Next, we construct a return variable, a binary variable y_t equal to 1 when return is positive and 0 otherwise, and estimate a model with an intercept only:

```
sp500ret <- diff(log(sp500data[, "Adj.Close"])) * 100
y <- sp500ret > 0

mymodel1 <- logitx(y)
mymodel1

Date: Sat Dec 25 11:24:37 2021
Dependent var.: y
Method: Maximum Likelihood (logit)
Variance-Covariance: Ordinary
No. of observations: 8240
Sample: 1983-07-05 to 2016-03-08

Estimation results:
```

```
              coef std.error t-stat  p-value
intercept 0.135157  0.022083 6.1204 4.88e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log-lik.(n=8240) -5692.76
```

Plotting the fitted probabilities (`plot(mymodel)`), or printing them (`fitted(mymodel)`), reveals that about 53.4% of the returns were positive over the sample. Can we increase the predictive probability using additional information?

The additional information we will use are past values of the y_t's, the maximum and minimum values of the SP500 index, daily volume and day-of-the-week dummies. Specifically, the following lines of code create a lagged range-based volatility proxy, the lagged log-difference of volume and day-of-the-week dummies:

```
relrange <- (log(sp500data[, "High"]) - log(sp500data[, "Low"]) ) * 100
volproxy <- log(relrange^2)
volproxylag <- lag(volproxy, k = -1)
volume <- log(sp500data[, "Volume"])
volumediff <- diff(volume) * 100
volumedifflag <- lag(volumediff, k = -1)

##day-of-the-week dummies:
sp500Index <- index(sp500Ret)
days <- weekdays(sp500Index)
days <- union(days, days)
dTue <- zoo(as.numeric(weekdays(sp500Index) == days[1]),
  order.by = sp500Index)
dWed <- zoo(as.numeric(weekdays(sp500Index) == days[2]),
  order.by = sp500Index)
dThu <- zoo(as.numeric(weekdays(sp500Index) == days[3]),
  order.by = sp500Index)
dFri <- zoo(as.numeric(weekdays(sp500Index) == days[4]),
  order.by = sp500Index)
```

We can now estimate the starting model or GUM:

```
mymodel2 <- logitx(y, ar = 1:5, ewma=list(length=c(20,60,120)),
  xreg = cbind(volproxylag, volumedifflag, dTue, dWed, dThu, dFri))
mymodel2

Date: Sat Dec 25 14:14:59 2021
Dependent var.: y
Method: Maximum Likelihood (logit)
Variance-Covariance: Ordinary
No. of observations: 8120
Sample: 1983-12-22 to 2016-03-08

Estimation results:

coef  std.error  t-stat p-value
intercept     -0.3783725  0.2678696 -1.4125 0.07892 .
ar1           -0.0824550  0.0464951 -1.7734 0.03810 *
ar2           -0.1000969  0.0466468 -2.1458 0.01596 *
ar3           -0.1199218  0.0464319 -2.5827 0.00491 **
ar4           -0.0319881  0.0462894 -0.6910 0.24478
ar5           -0.0040336  0.0461810 -0.0873 0.46520
EqWMA(20)     -0.4135721  0.2997231 -1.3798 0.08384 .
EqWMA(60)     -0.4807224  0.6490426 -0.7407 0.22946
EqWMA(120)     2.1610609  0.7566467  2.8561 0.00215 **
volproxylag   -0.0043708  0.0209305 -0.2088 0.41730
volumedifflag  0.0010638  0.0011760  0.9046 0.18284
dTue          -0.0133449  0.0711182 -0.1876 0.42558
dWed           0.0472649  0.0727936  0.6493 0.25808
dThu           0.0097764  0.0719075  0.1360 0.44593
dFri           0.0582887  0.0716584  0.8134 0.20800
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Log-lik.(n=8120) -5593.63
```

Next, simplification via GETS modelling is undertaken with the `gets()` method (S3) for models of class `"logitx"`:

```
finalmodel <- gets(mymodel2)
finalmodel
```

```
....

Final model:

              coef std.error  t-stat   p-value
ar2       -0.112386  0.044943 -2.5006  0.006208 **
ar3       -0.132671  0.044928 -2.9530  0.001578 **
EqWMA(120)  0.508064  0.077029  6.5958 2.248e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Log-lik.(n=8120) -5601.03
```

On a standard computer under default settings, the simplification takes about 1 minute. By setting `max.paths = 1`, the time can be reduced to about 1/10 without changing the final model. The likely reason for this is that the number of observations is large (more than 8 000). To obtain an idea of whether the final model increase the predictive probability in notable ways, a summery of the fitted probabilities can be generated:

```
summary(fitted(finalmodel))

    Index              fitted(finalmodel)
Min.   :1983-12-22   Min.   :0.4917
1st Qu.:1992-01-05   1st Qu.:0.5149
Median :2000-01-16   Median :0.5364
Mean   :2000-01-23   Mean   :0.5350
3rd Qu.:2008-02-13   3rd Qu.:0.5459
Max.   :2016-03-08   Max.   :0.5839
```

As is clear, the maximum predictive probability is about 58.4% when the additional information is used. For most purposes, the increase from 53.4% to 58.4% is negligible.