melt package review

My review is based on Version 1.7.0 from CRAN.

Package Review

- [X] As the reviewer I confirm that there are no conflicts of interest for me to review this work (If you are unsure whether you are in conflict, please speak to your editor *before* starting your review).

Compliance with Standards

- [X] This package complies with a sufficient number of standards for a silver badge
- [X] This grade of badge is the same as what the authors wanted to achieve

The following standards currently deemed non-applicable (through tags of `@srrstatsNA`) could potentially be applied to future versions of this software:

G1.6 Software should include code necessary to compare performance claims with alternative implementations in other R packages.: Other packages compute empirical likelihood tests and confidence intervals. We would like to see how the `melt` package compares with the others. For example, my *momentfit* (or *gmm*) package produces the same type of confidence intervals or confidence areas for linear models and the mean. The package emplik now includes regression models.

- [X] This package extends beyond minimal compliance with standards in the following ways: (please describe)

This is clearly not a package suited for a unique application. the algorithm to solve restricted empirical likelihood models is efficient and can be applied to many more models not yet implemented by the package. For example, The author seems to discard the idea of applying the method to over-identified models (from the NOT TO DO section of CONTRIBUTING.md), but it could easily be implemented since constrained empirical likelihood is in fact over-identified. I could also see other packages being based on `melt` for specific applications.

Comments

"@srrstats {RE1.4} Some of the core distributional assumptions are documented.": The list of regularity conditions is not that long. We do see some distribution assumptions in the `EL` documentation where the EL method is presented for general estimating equations $E[g(X,\theta)] = 0$, but it would be important to add at least the assumption on the variance of $g(X,\theta)$. Users using only `el_mean` or `el_sd` should know what distribution assumptions are required. For these two functions, the detail section is empty. At least, we should see what the estimating equations are and then be given a link to the EL documentation for more details.

"@srrstats {RE4.11} `logLik()` method extracts the empirical log-likelihood value which can be interpreted as a measure of goodness-of-fit": It is really not clear how to interpret the value. Can we have a more detailed explanation? First, it is the empirical likelihood value of what? For example, if we estimate a linear model:

```r
set.seed(112233)
x <- rnorm(40); y <- rnorm(40)
fit <- el_lm(y~x)
print(logLik(fit), digits=10)
```

```
## 'Empirical log Lik.' -147.5551782 (df=2)
```

This is the log-likelihood of what? If I understand correctly (not clear from the documentation), It is not the one from the model that restrict the slope coefficient to be 0, which is what we have in the slot `logl` or if we sum the log of the implied probabilities:

```r
print(fit@logl, digits=10)
```

```
## [1] -147.5556237
```

```r
print(sum(fit@logp), digits=10)
```

```
## [1] -147.5556237
```

After more testing, it seems to be the one from the unrestricted model for which the implied probabilities are all equal to $1/n$:

```r
print(40*log(1/40), digits=10)
```

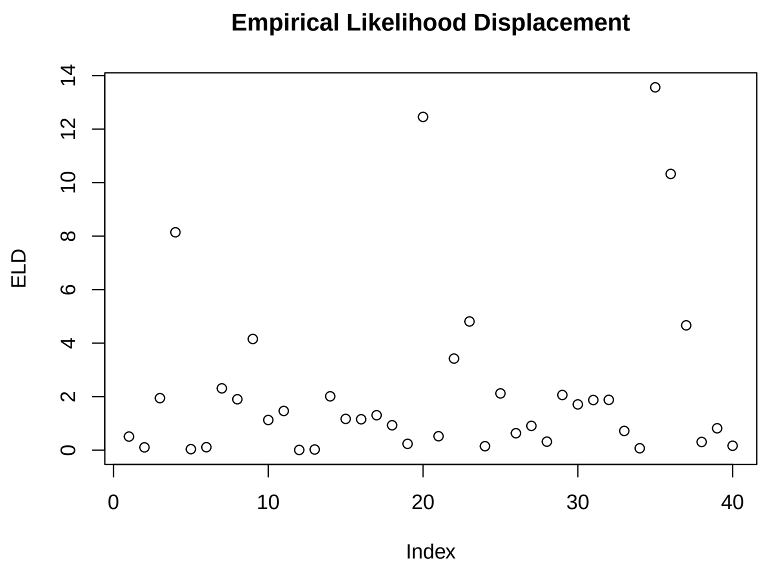
```
## [1] -147.5551782
```

I think this needs to be explained in more details.

"@srrstats {RE6.0} `plot` method is available for an `ELD` object that is returned by the `eld()` method.": The authors may consider adding some diagnostic-type plot methods for `EL` (or any other) class as we have for `lm` objects. It is great to have a plot for `eld()` to detect outliers, but it would be even better I think if the plot was applied directly to `EL` class. For example, a Cook Distance exists separately for `lm`, but we can plot it directly from `lm` objects by selecting `which=4`:

```r
## clearly not the best method, but you see the idea
setMethod("plot", "EL", function(x, which = 1, ...) {
    if (which==1)
        plot(eld(x))
    else
        cat("other diagnostic plots if any\n")
})
```

```r
plot(fit, which=1)
```



General Review

This is a very well designed package. The outputs of the different estimating or testing methods are easy to read and the algorithms seem to be quite efficient. The following are mostly suggestions that in my view could improve the package.

Documentation

The package includes all the following forms of documentation:

- [ ] **A statement of need** clearly stating problems the software is designed to solve and its target audience in README

I think we should see a more detailed presentation of why one would need to use the `melt` package to perform hypothesis tests or to construct confidence intervals. For example, when the empirical likelihood starts to produce confidence intervals with better coverage than `lm` does for linear regression? In the example section of `el_lm`, we see an application with iid and normally distributed variables. In this case, even when the same size is very small, a least square confidence interval has an exact coverage while EL does not. So why should we use the `melt` package?

- [X] **Installation instructions:** for the development version of package and any non-standard dependencies in README
- [X] **Community guidelines** including contribution guidelines in the README or CONTRIBUTING
- [ ] The documentation is sufficient to enable general use of the package beyond one specific use case

There is room for improvement here.

The `optim` slot

We see that a slot `optim` exists in the LM object created by `el_lm` through the CEL-class. But it is not clear from the documentation what these values are.

```r
set.seed(5649)
df <- data.frame(y = rnorm(50), x = rnorm(50))
fit <- el_lm(y ~ x, df)
slotNames(fit)
```

```
##  [1] "sigTests"     "call"         "terms"        "misc"         "optim"
##  [6] "logp"         "logl"         "loglr"        "statistic"    "df"
## [11] "pval"         "nobs"         "npar"         "weights"      "coefficients"
## [16] "method"       "data"
```

From the CEL-class documentation, it says that `par` is $\hat{\theta}$ (subject to the constraints which we do not have here) and `lambda` is $\hat{\lambda}$. But it is not:

```
coef(fit)
```

```
## (Intercept)           x
##  -0.1131575  -0.1089830
```

```
fit@optim$par
```

```
## (Intercept)           x
## -0.09474126  0.00000000
```

```
fit@optim$lambda
```

```
## [1] -0.04315758 -0.24222693
```

What is `par`? What is `lambda`? Since the model is just-identified, the latter should be a vector of zeroes when there are no constraints. Are they the estimates when we restrict all coefficients to be zero? (except for the intercept). Can that be added to the documentation? If we explore further, we learn that it seems to be the case:

```
set.seed(5649)
df2 <- data.frame(y = rnorm(50), x = rnorm(50), z = rnorm(50))
fit2 <- el_lm(y ~ x+z, df2)
fit2@optim$par
```

```
## (Intercept)           x           z
## -0.08886676  0.00000000  0.00000000
```

I think it is important information that may be useful for some users.

In the `el_lm` and `el_glm` (maybe others?) it is written in the details that the test for the null $H_0:\ \theta_1 = \theta_2 = \cdots = \theta_{p-1} = 0$ is in `optim` and the test parameter by parameter is in `sigTests`, but there is no test result in `optim`. It only includes the parameters, the lambdas, the number of iterations and the convergence code. Also, why is it important to return the implied probabilities (the `logp` slot) and the lambdas for the joint test and not for the individual ones?

The other EL slots

The slots are documented in EL, but it is not easy to get it for other objects. For example, in the `el_glm` documentation, the Value is "An object of class GLM". If we click in it, we only see that it inherits from LM. If we click on it, we see some slots, but not all. It inherits from CEL, so we click on it and only the `optim` slot is defined. We finally get the description of all slots by clicking on EL. It is not an easy process from someone like me who only work in terminals. It requires to call `help()` several times. It would be helpful if the slots were defined for all objects that inherits from EL.

Also, it is not always clear what the slots are. In EL, we see "A numeric vector of the log probabilities obtained from empirical likelihood" for the `logp` slot. But which probabilities? I figured out for `el_lm` that it is for the restricted model when all parameters are 0 except the intercept. It should be specified. It is the same for `logl`, `loglr`, etc. This is something we see across the different help files. It is another reason the author should define all slots for all functions that return an EL object.

The Empirical Likelihood method

I would expect a little more details about the EL method. It is well presented in the EL documentation for general estimating equations $g(X, \theta)$, but I think it would be helpful for the method to also be explained in `el_lm`, `el_glm`, `el_sd`, `el_mean` and even `el_eval`. The EL method does not need to be repeated, but at least we should see the specific estimating equation used for each model and a link to EL for the details. For example, it is not clear to me which estimating equation is used for the standard deviation or for generalized linear models.

It is not clear what is the purpose of the `el_eval`. It says that it is for general estimating equation, but the input is a matrix, so it is an estimating equation without parameters. The exact same result can be obtained with 'el_mean': the following two produced identical `optim`, `logp`, `logl` and `loglr`:

```
x <- matrix(rnorm(200), 50, 4)
fit <- el_eval(x)
fit2 <- el_mean(x,rep(0,4))
```

So, why `el_eval` does not simply call `el_mean`? If the output has to be a list, it can returned as a such.

Other comments

As a suggestion, the package can be greatly improved by adding more general nonlinear regression models. In `el_glm` is a nonlinear model, so it should bot be to difficult to include nonlinear regression like in `nls` or even more general cases. With a method to estimate general nonlinear models, it becomes straightforward to test nonlinear restrictions, if they are limited to restrictions that can be written as $H_0: \ \theta_i = f(\theta_{-i})$. if not too difficult to implement because it only implies replacing θ_i by $f(\theta_i)$ in the estimating equation function $g(X, \theta)$.

Many recent applications rely on nonlinear estimating equations that are often over-identified. It is the case of all methods recently developed in the missing value literature (or the causal effect literature, which is just a special case).

Algorithms

The `confint` method

It seems that all methods for the `EL`, `QGLM` and `SD` classes are almost identical and can be reduced to a single method applied to a union class. I can see some benefits from avoiding repeating the same codes in several methods, especially if changes need to be applied to them.

Why the `getMethodEL(object)` is used to get the type of method for the classes `EL` and `QGLM` and not for the class `SD`? Is it because the `getMethodEL` method does not exist for this class? For the latter, the type is explicitly added (see line 214 of confint-methods.R)

The `elt` method

Would be nice to see the hypothesis being tested when the result is printed (see for example the `linearHypothesis` function from the *car* package).

Testing

Don't see any issue with the different tests.

Visualisation (where appropriate)

No issue here other than the above suggestion about the `plot` method.

Package Design

- As I mentioned above, it is a well designed package. If we are looking for EL confidence intervals for traditional models, which is exactly the purpose as described by the author, it does a great job. It is also one of the most efficient algorithm I have tried to estimate restricted empirical likelihood model.
- **External Design:**
- In relation to **External Design:** The objects created by most estimating functions have methods that produce output comparable to `lm` `glm` and `nls`. We can then easily compare the results.

- [X] **Packaging guidelines**: The package conforms to the rOpenSci packaging guidelines

Estimated hours spent reviewing: 16

- [X] Should the author(s) deem it appropriate, I agree to be acknowledged as a package reviewer ("rev" role) in the package DESCRIPTION file.