An introduction to the R package \texttt{sn}

Adelchi Azzalini
Università di Padova, Italia

Skewed world of data:
Workshop in honor of Reinaldo B. Arellano-Valle’s 65th birthday
October 2017
Pontificia Universidad Católica de Chile
Pacakge sn: a one-slide panorama

**probability**

- classical-type functions for distributions:
  - univariate dist'n: \{d,p,q,r\}\{sn,st,sc\}
  - multivariate dist'n: \{d,p,r\}\{msn,mst,msc\}
  - utilities: dp2cp, cp2dp, \{sn,st\}.cumulants,…
  - build your own dist'n: \{d,r\}\{SymmModulated\},
    \{d,r\}\{mSymmModulated\} and plot (if \(d = 2\))

- SEC distribution ‘objects’
  - two classes: SECdistrUv, SECdistrMv; protocol: S4
  - create object: makeSECdistr (also extractSECdistr)
  - manipulate it: marginalSECdistr, affineTransSECdistr, conditionalSECdistr (only for SN)
  - methods: summary, plot, show, mean, sd, var,…

**statistics**

- core general function: selm (two object types: selm, mselm)
- methods: show, plot, summary, coef, residuals, fitted, predict, logLik, profile, confint,…
Prob-std
Simple functions: `dsn` and plotting SN density

```r
# plotting SN densities
library(sn)
x <- seq(-2, 4, length=301)
y1 <- dsn(x, xi=0, omega=1.2, alpha=10)
plot(x, y1, type="l", ylab="density")
y2 <- dsn(x, 0, 1.2, 5)
lines(x, y2, col=2)
y3 <- dsn(x, dp=c(0, 1.2, 2))
lines(x, y3, col=4, lty=2)
```

![Graph showing SN densities plotted with different parameters.](image)
Simple functions: \texttt{cp2dp}, \texttt{dp2dp}, \texttt{rst}, \texttt{dst}

# switch between parameterizations
\begin{verbatim}
cpST <- c(1, 1.5, 1.5, 5.1)
# CP = (mean, st.dev, gamma1, gamma2)
dpST <- cp2dp(cpST, family="ST")
\end{verbatim}

\begin{verbatim}
print(dpST)
# xi omega alpha nu
# -0.6202 1.8754 3.6733 7.1797
print(dp2cp(dpST, family="ST"))
# back to cpST
\end{verbatim}

# sampling from ST and plotting density
\begin{verbatim}
y <- rst(1000, dp=dpST)
hist(y, prob=TRUE, col="gray90")
x <- seq(min(y), max(y), length=301)
pdfST <- dst(x, dp=dpST)
lines(x, pdfST, col=4)
\end{verbatim}

print(sd(y)) # about cpST[2]
Unleash your creativity: case $d = 1$

# SGN of Arellano-Valle et al. (2004)

```r
wSGN <- function(z, lambda) z * lambda[1]/sqrt(1 + lambda[2]*z^2)
x  <- seq(-1, 14, length=301)
pdf <- dSymmModulated(x, 5, 2, f0="normal", G0="normal", w=wSGN,  
  lambda=c(3,5))
plot(x, pdf, type="l")
```

now change f0, G0, w,... and

make your density
Unlash your creativity: case $d > 1$

```r
x <- matrix((1:12)/3, 4, 3)
S <- diag(1:3) + outer(1:3, 1:3)/2
wMvTrigs <- function(z, p, q) sin(z %% p)/(1 + cos(z %% q))
pdf <- dmSymmModulated(x, xi=1:3, Omega=S, f0="t", G0="logistic",
                       w=wMvTrigs, par.f0=5, par.G0=NULL, p=c(2,3,-2), q=c(1,1,0))

# plotting when d=2
range <- cbind(c(-4,4), c(-4,4))
plot2D.SymmModulated(range, xi=c(0,0), Omega=S[1:2,1:2],
                       f0="normal", G0="normal", w=wMvTrigs,
                       par.f0=NULL, par.G0=NULL, p=c(1,-3), q=c(1,1), col=4)
y <- rmSymmModulated(2500, xi=c(0,0), Omega=S[1:2,1:2],
                       f0="normal", G0="normal", w=wMvTrigs,
                       par.f0=NULL, par.G0=NULL, p=c(1,-3), q=c(1,1))
points(y, cex=0.3, col="gray60")
```
Unlash your creativity: plotting if $d = 2$
<table>
<thead>
<tr>
<th>Scheme</th>
<th>Prob-std</th>
<th>Prob-obj</th>
<th>Stats</th>
<th>q()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Prob-obj
Working with distribution ‘objects’

- Idea is to **make** a SEC distribution and work with this **object**
- Once the distribution object is created, we can:
  - extract/compute various characteristics
  - plot it
  - manipulate it to create a new distribution (if \( d > 1 \))
- Technical note: distributions are S4-type objects
How to create a distribution

- Chief procedure is `makeSECDistr` (another route is `extractSECDistr` from `selm` or `mselm` object)
- we must to specify DP parameters
  - if univariate case, DP is assigned as a vector
  - if multivariate case, DP is assigned as a list
- we must to specify the family,
  options are: "SN", "ESN", "ST", "SC"
- optional: name of the distribution, names of components
- `?makeSECDistr` tells you everything
f1 <- makeSECdistr(dp=c(3,2,5), family="SN", name="First-SN")
#
show(f1) # of just type ’f1’
# Probability distribution of variable ’First-SN’
# Skew-elliptically contoured distribution of univariate family SN
# Direct parameters:
# xi omega alpha
# 3 2 5

summary(f1) # longer output
#
c(mean(f1), sd(f1)) # lends 4.565 1.246

plot(f1)

# many possible variants, such as:
plot(f1, probs=c(0.1, 0.9))

?plot.SECdistr # says more, see method for signature ’SECdistrUv’
A simple illustration – plot 'First-SN'

**Probability density of First-SN**

SN distribution, dp = (3, 2, 5)
An illustration with $d = 3$

dp3 <- list(xi = c(3, -2, 0), Omega = diag(1:3) + outer(1:3,1:3)/2, 
          alpha = c(5, -2, 6), nu = 5)

st3 <- makeSECdistr(dp=dp3, family="ST", name="Multiv.ST", 
                      compNames=c("X", "Y", "Z"))

show(st3) # of just type 'st3'

mean(st3)
# X   Y   Z
# 3.944 -1.253 2.194

vcov(st3) # 3x3 variance matrix

summary(st3) # longer output

plot(st3, col="blue", landmarks="", main=NULL) # note p=0.xx labels

?plot.SECdistr # look at method for signature 'SECdistrMv'
An illustration with $d = 3$: matrix plot
Manipulation of a distribution

- Refers to multivariate distributions only
- `affineTransSECDistr(object, a, A, <etc>)`
  applies affine transformation \( a + A'Y \),
  where \( a \) is \( m \)-vector and \( A \) is \( d \times m \) matrix
- `marginalSECDistr(object, comp, <etc>)`
  get marginal distribution of \( comp \) components from \( object \)
- `conditionalSECDistr(object, fixed.comp, fixed.values, <etc>)`
  applies conditioning on \( fixed.comp \) components
  (only for "SN" and "ESN" distributions)
Function `selm` for model fitting

Naming: `lm + <Skew-Elliptical error> = selm`
Also similar logic of `lm`: fit a linear model to location parameter

```
fit <- selm( response ~ formula, family = "SN", <plus>)
```

S4 obj vector, matrix like in `lm` also "ST", "SC"

Optional `<plus>` terms include:
data, subset the same as in `lm`
start starting values
method estimation methods are "MLE" and "MPLE";
the latter can be used to set a prior distribution of \( \alpha \)
penalty only relevant for method="MPLE"
fixed.param allows only limited specifications, such as nu=<value>
(if family="ST") and alpha=0
... additional options
Methods for a \texttt{selm|mselm} object

classes \texttt{selm} returns a S4 object of class \texttt{selm} (for univariate response) or \texttt{mselm} (for multivariate response)

methods for each class, a bunch of ‘methods’ are available

- methods like for \texttt{lm} S3-objects: \texttt{summary, plot, residuals, fitted, coef, predict, confint}
- additional methods: \texttt{logLik, profile, vcov}

in addition \texttt{extractSECDistr} supplies a link to the probability section

Note: for simpler interpretability, the default parameterization is CP. This contrasts with probability section which uses DP.
A simple example: Barolo phenols

```r
library(sn)
data(wines)
olo.ph <- wines[wines$wine=="Barolo", "phenols"]
fit <- selm(olo.ph ~ 1, family="SN")
plot(fit, which=2:3)
#
# try
summary(fit)  # works with CP
summary(fit, param.type="DP")
```
A simple example: Barolo phenols – plots
library(sn)
data(wines)

fit2 <- selm(cbind(tartaric, malic, uronic) ~ colour + hue,
             family="ST", data=wines, subset=(wine != "Grignolino"))

plot(fit2, which=2:3)

summary(fit2)  # works with CP
summary(fit2, param.type="DP")

# constraint on degrees of freedom:
fit3 <- selm(cbind(tartaric, malic, uronic) ~ colour + hue,
             family="ST", fixed.param=list(nu=8),
             data=wines, subset=(wine != "Grignolino"))
A multivariate example: más vino, plots
For the more adventurous: profile log $L$ (LRT in fact)

# re-use earlier model for Barolo phenols
show(fit)
# Object class: selm
# Call: selm(formula = ba.ph ~ 1, family = "SN")
# [...omissis...]

summary(fit)
# [...omissis...]
# Parameters of the SEC random component
#   estimate std.err
# s.d.   0.337   0.04
# gamma1 0.703   0.26

pll <- profile(fit, "cp", param.name="gamma1", param.val=c(0, 0.97))

profile(fit, "dp", param.name=c("omega", "alpha"),
        param.val=list(c(0.25, 1), c(-1, 9)), npt=c(51,51) )

Note: works for selm-class objects, not mselm-class
For the more adventurous: profile log $L$, plots

Plots of (profile) Deviance $\equiv$ LRT statistic
Only for the more technically oriented people

- `selm` is the user interface function
- `selm` prepares work for the lower-level function `selm.fit`
- however, not even `selm.fit` performs the actual fitting
- depending on the fitted model, specific functions are called: `sn.mple, st.mple, msn.mle, msm.mple, mst.mple`
- To improve efficiency, one can call `selm.fit` directly, at the cost of some more programming effort
- One can even call the bottom-level functions, below `selm.fit`
- For more details, see `?selm.fit`
<table>
<thead>
<tr>
<th>Scheme</th>
<th>Prob-std</th>
<th>Prob-obj</th>
<th>Stats</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\(q()\)