The **SpecsVerification** package user guide

Stefan Siegert

Version: January 2014

Contents

1 Install and load

Specify the directory of the package source (i.e. the directory that contains the subdirectories `R`, `man` and the files `DESCRIPTION` and `NAMESPACE` etc) and the directory where the package should be installed. For example:

```
pkgdir <- "/home/stefan/folders/specs/r-package/specs-verification-git"
destdir <- "/tmp"
```

Then install and load the package as follows:

```
install.packages(pkgdir, repo=NULL, lib=destdir, type="source")
library("SpecsVerification", lib.loc=destdir)
```

2 Load some example data

We will use seasonal ensemble forecast of surface temperature averaged over the `Atl3` region between 1993 and 2009, initialized twice a year (1 May and 1 November) and run 1 up to 4 months into the future. "Observations" are generated from the `ERAint` data set. Ensemble forecasts are generated by `ECMWF System4` (15 members) and an anomaly initialized ensemble generated at IC3. The data can be downloaded from the SPECS wiki: Atl3.ERAint.Rdata

```
file <- "Atl3.ERAint.Rdata"
if (!file.exists(file)) {
  download.file(url="http://www.specs-fp7.eu/wiki/images/a/a8/Atl3.ERAint.Rdata",
                destfile=file)
}
load(file)

# the dimensions of obs are 1:initdate, 2:leadtime
dimnames(obs)
```

```
R output >>  [[1]]
R output >>   [1] "1993-05-01" "1993-11-01" "1994-05-01" "1994-11-01" "1995-05-01"
R output >>   [6] "1995-11-01" "1996-05-01" "1996-11-01" "1997-05-01" "1997-11-01"
R output >>  [11] "1998-05-01" "1998-11-01" "1999-05-01" "1999-11-01" "2000-05-01"
R output >>  [16] "2000-11-01" "2001-05-01" "2001-11-01" "2002-05-01" "2002-11-01"
R output >>  [21] "2003-05-01" "2003-11-01" "2004-05-01" "2004-11-01" "2005-05-01"
R output >>  [26] "2005-11-01" "2006-05-01" "2006-11-01" "2007-05-01" "2007-11-01"
R output >>  [31] "2008-05-01" "2008-11-01" "2009-05-01" "2009-11-01"
R output >>
R output >>  [[2]]
R output >>  [1] "1" "2" "3" "4"
```

```
# the dimensions of ens are 1:initdate, 2:leadtime, 3:ensemble, 4:member
dimnames(ens)
```

```
R output >>  [[1]]
R output >>   [1] "1993-05-01" "1993-11-01" "1994-05-01" "1994-11-01" "1995-05-01"
R output >>   [6] "1995-11-01" "1996-05-01" "1996-11-01" "1997-05-01" "1997-11-01"
R output >>  [11] "1998-05-01" "1998-11-01" "1999-05-01" "1999-11-01" "2000-05-01"
R output >>  [16] "2000-11-01" "2001-05-01" "2001-11-01" "2002-05-01" "2002-11-01"
```

```
R output >>  [21] "2003-05-01" "2003-11-01" "2004-05-01" "2004-11-01" "2005-05-01"
R output >>  [26] "2005-11-01" "2006-05-01" "2006-11-01" "2007-05-01" "2007-11-01"
R output >>  [31] "2008-05-01" "2008-11-01" "2009-05-01" "2009-11-01"
R output >>
R output >>  [[2]]
R output >>  [1] "1" "2" "3" "4"
R output >>
R output >>  [[3]]
R output >>  [1] "ecmwf" "l00w"
R output >>
R output >>  [[4]]
R output >>   [1] "member.1"  "member.2"  "member.3"  "member.4"  "member.5"
R output >>   [6] "member.6"  "member.7"  "member.8"  "member.9"  "member.10"
R output >>  [11] "member.11" "member.12" "member.13" "member.14" "member.15"
```

```
t <- paste(seq.Date(from=as.Date("1993-05-01"),
           to=as.Date("2009-05-01"), by="1 year"))
ecmwf <- ens[t,"1","ecmwf",]
l00w <- ens[t,"1","l00w",1:5]
ver <- obs[t,"1"]
plot(NULL, xlim=c(1, length(ver)), ylim=range(c(ver,l00w,ecmwf)),
     xlab="time", ylab="Atl3 temp. [C]")
points(ver, pch=15, cex=1.5, col=gray(.5))
for (i in 1:nrow(ecmwf)) points(rep(i-.2,15), ecmwf[i,], pch=15, cex=0.5)
for (i in 1:nrow(ecmwf)) lines(rep(i-.2,2), range(ecmwf[i,]))
for (i in 1:nrow(l00w)) points(rep(i+.2,5), l00w[i,], pch=16, cex=0.5)
for (i in 1:nrow(l00w)) lines(rep(i+.2,2), range(l00w[i,]))
legend(x="bottomright", ncol=3, legend=c("l00w","ecmwf","obs"),
       lty=c(1,1,NA), pch=c(16,15,15), pt.cex=c(0.7,0.7,1.5),
       col=c("black","black",gray(.5)))
```

3 Fair Brier Score analysis

The function `FairBrier` returns individual values of fair Brier scores of ensembles and their verifications. The argument `tau` is the threshold to whose exceedance defines the binary event:

```
fbr.ecmwf <- FairBrier(ens=ecmwf, obs=ver, tau=28.5)
fbr.l00w <- FairBrier(ens=l00w, obs=ver, tau=28.5)
plot(fbr.ecmwf, type="b", pch=15, col=gray(.5))
lines(fbr.l00w, type="b", pch=16, col="black")
```

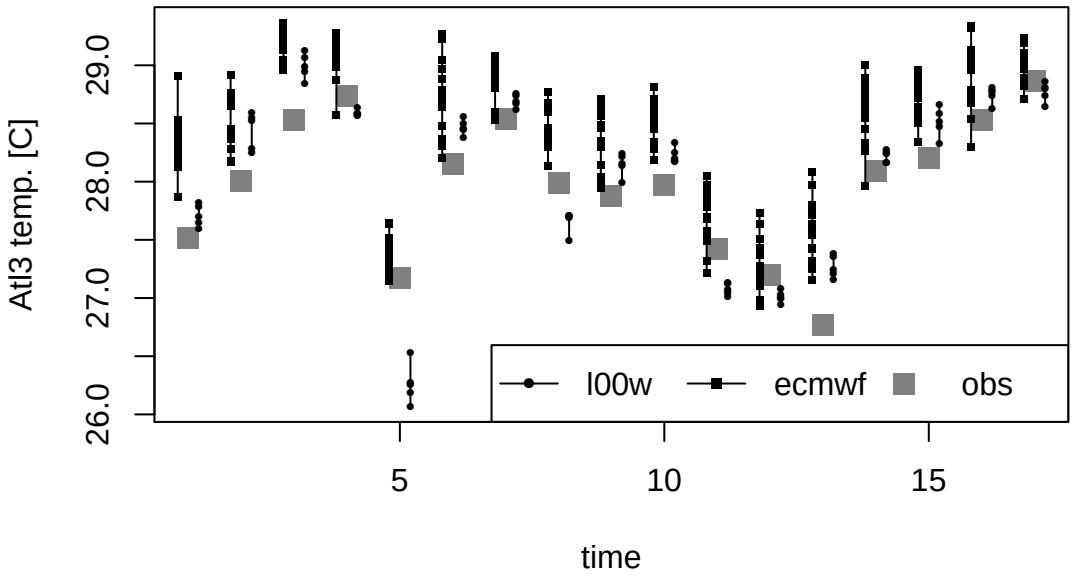

Figure 1: Time series of the ensemble data and observations

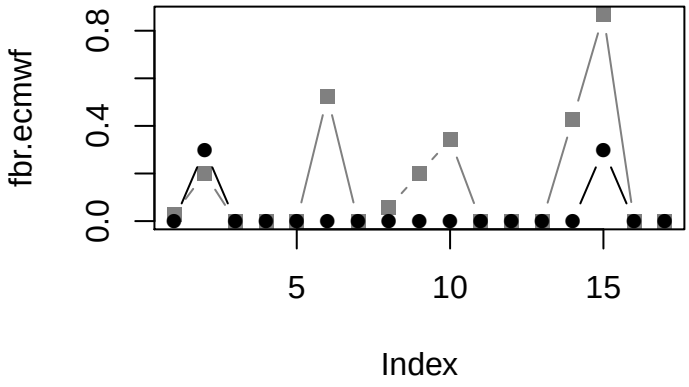

Figure 2: Plot of fair Brier Scores of ECMWF (gray) and l00w (black).

```
print(c(mean(fbr.ecmwf), mean(fbr.l00w)))
```

```
R output >>  [1] 0.15574 0.03529
```

The function `AnalyzeFairBrier` returns the mean fair Brier score difference and optional estimated quantiles of the sampling distribution of the mean difference:

```
FairBrierDiff(ens=l00w, ens.ref=ecmwf, obs=ver,
                 tau=28.5, probs=c(0.05, 0.95))
```

```
R output >>  $br.diff
R output >>  [1] 0.1204
R output >>
R output >>  $sampling.quantiles
R output >>     0.05    0.95
R output >>  0.03162 0.20928
```

4 Fair CRPS analysis

The fair continuously ranked probability score for ensemble forecasts has similar routines as the fair Brier score:

```
fcrps.ecmwf <- FairCrps(ens=ecmwf, obs=ver)
fcrps.l00w <- FairCrps(ens=l00w, obs=ver)
plot(fcrps.ecmwf, type="b", pch=15, col=gray(.5))
lines(fcrps.l00w, type="b", pch=16, col="black")
```

```
print(c(mean(fcrps.ecmwf), mean(fcrps.l00w)))
```

```
R output >>  [1] 0.3485 0.2566
```

```
print(FairCrpsDiff(ens=l00w, ens.ref=ecmwf, obs=ver, probs=c(0.05, 0.95)))
```

```
R output >>  $crps.diff
R output >>  [1] 0.0919
R output >>
R output >>  $sampling.quantiles
R output >>      0.05     0.95
R output >>  -0.01167  0.19547
```

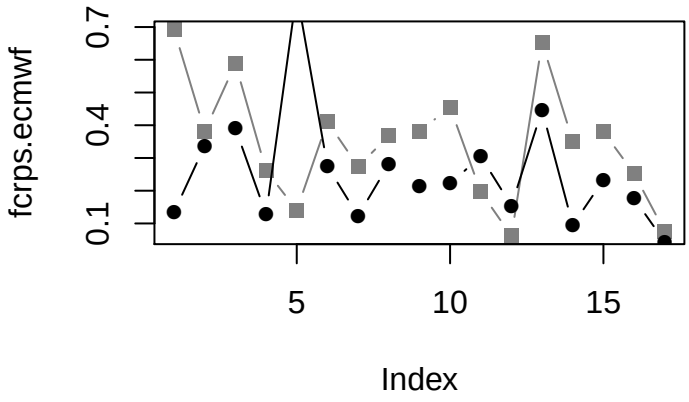

Figure 3: Plot of fair CRPS of ECMWF (gray) and l00w (black).

5 Rank histogram analysis

The ECMWF ensemble and the observations are transformed to anomalies by centering them around zero. Then the rank histogram is drawn in the "raw" version and on probability paper:

```
ecmwf <- ens[,1,"ecmwf",]
ecmwf <- ecmwf - mean(ecmwf)
ver <- obs[,1]
ver <- ver - mean(ver)
rh <- Rankhist(ens=ecmwf, obs=ver)
PlotRankhist(rh, mode="raw")
```

```
PlotRankhist(rh, mode="prob.paper")
```

6 Reliability diagram analysis

The experimental l00w ensemble and the observations are transformed to anomalies. The event of interest is the exceedance of a value of 1 and probabilities are generated for this event by counting ensemble members:

```
l00w <- ens[,4,"l00w",]
l00w <- l00w - mean(l00w, na.rm=TRUE)
p <- rowMeans(l00w > 1, na.rm=TRUE)
```

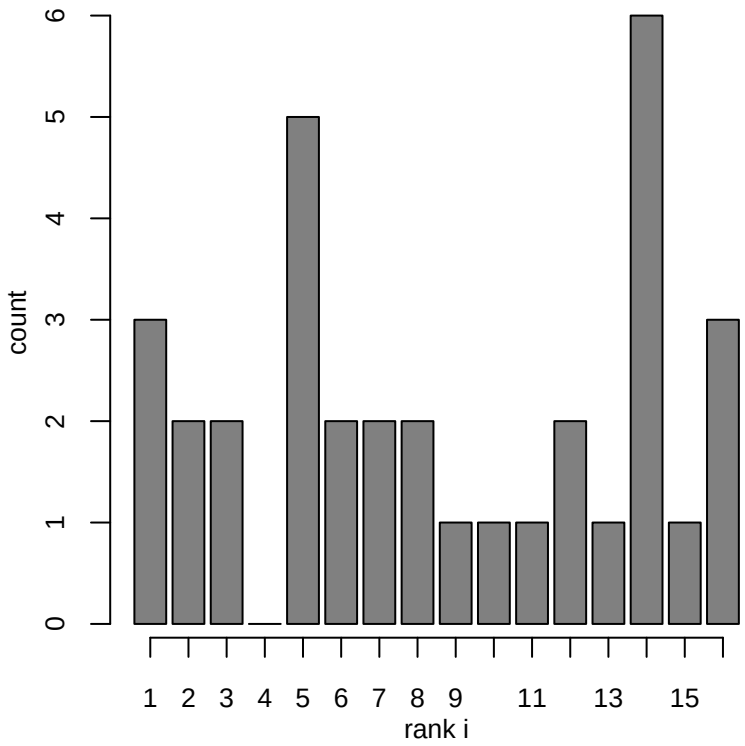

Figure 4: plot of chunk rankhist

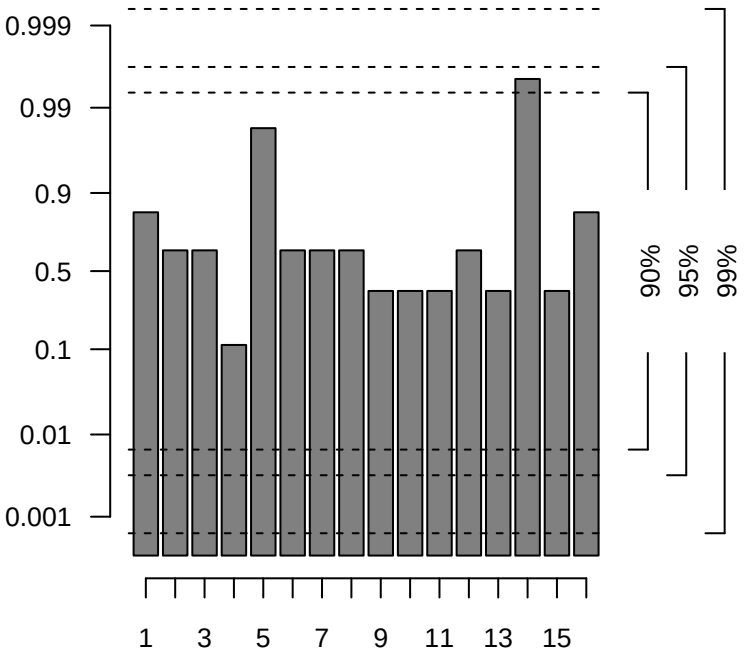

Figure 5: plot of chunk rankhist

```
ver <- obs[,4]
ver <- ver - mean(ver)
y <- 1 * (ver > 1)
rd <- ReliabilityDiagram(probs=p, obs=y, bins=c(0,1/3,2/3,1),
                         plot=TRUE, nboot=500, mc.cores=8)
```

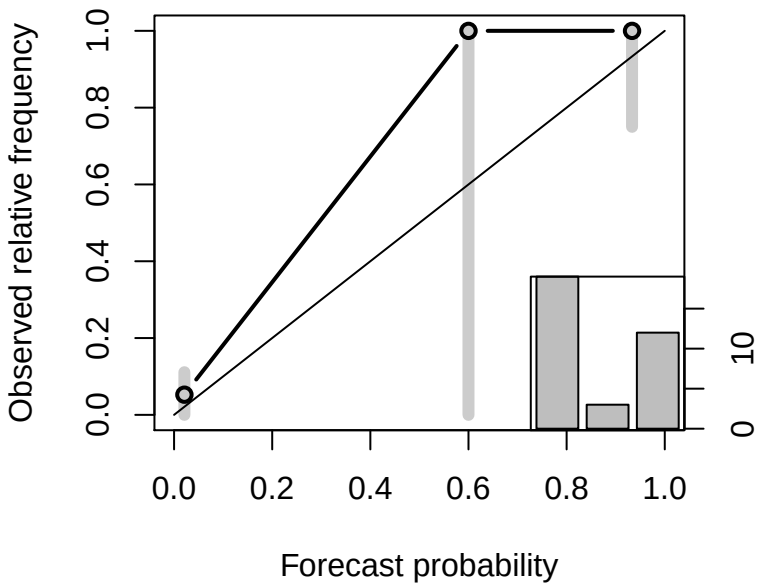

Figure 6: plot of chunk reldiag

The object `rd` contains all the numerical information to create the diagram from scratch:

```
print(rd)
```

```
R output >>     p.avgs cond.probs cbar.lo cbar.hi
R output >>  1 0.02105    0.05263    0.00  0.1111
R output >>  2 0.60000    1.00000    0.00  1.0000
R output >>  3 0.93333    1.00000    0.75  1.0000
```

7 Brier Score decomposition

The decomposition of the Brier score for probability forecasts quanitifies the violation of realibility and the resolution of the forecast, and the uncertainty of the observations:

```
BrierScoreDecomposition(p,y,calibration=list(method="bin", bins=c(0,1/3,2/3,1)))
```

```
R output >>       REL     RES     UNC
R output >>   0.02149 0.22127 0.24913
```

A new (experimental) method to estimate the calibration function by a logistic regression model is implemented:

```
BrierScoreDecomposition(p,y,calibration=list(method="logistic"))
```

```
R output >>       REL     RES     UNC
R output >>   0.02412 0.23443 0.24913
```

8 Ensemble Dressing

Ensemble dressing is a method to transform a discrete ensemble into a continuous probability distribution function by dressing each ensemble member with a kernel function (for example a Gaussian distribution function) and averaging over the individual kernels.

A few ensemble dressing methods have been implemented. The `DressEnsemble` method transforms a $N \times K$ matrix of K ensemble members into a `dressed.ensemble` object. The function `PlotDressedEns` is useful for examining the result.

```
t <- paste(seq.Date(from=as.Date("1993-05-01"),
                    to=as.Date("2009-05-01"), by="1 year"))
ecmwf <- ens[t,4,'ecmwf',]
ver <- obs[t,4]
d.ens <- DressEnsemble(ecmwf)
PlotDressedEns(d.ens, plot.ens=TRUE, obs=ver)
```

By default, the function `DressEnsemble` uses Gaussian kernels, centered on the ensemble members, and the kernel width is calculated by Silverman's rule of thumb ($\sigma_{ker} = \left(\frac{4}{3K}\right)^{1/5} \sigma_{ens}$).

A more clever version of ensemble dressing is Affine Kernel Dressing (AKD), where the centers of the kernels are transformations of the ensemble members and the kernel variance is a function of the ensemble variance:

$$p(x|e) = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{s_k(e)} \varphi \left(\frac{x - m_k(e)}{s_k(e)} \right)$$

where

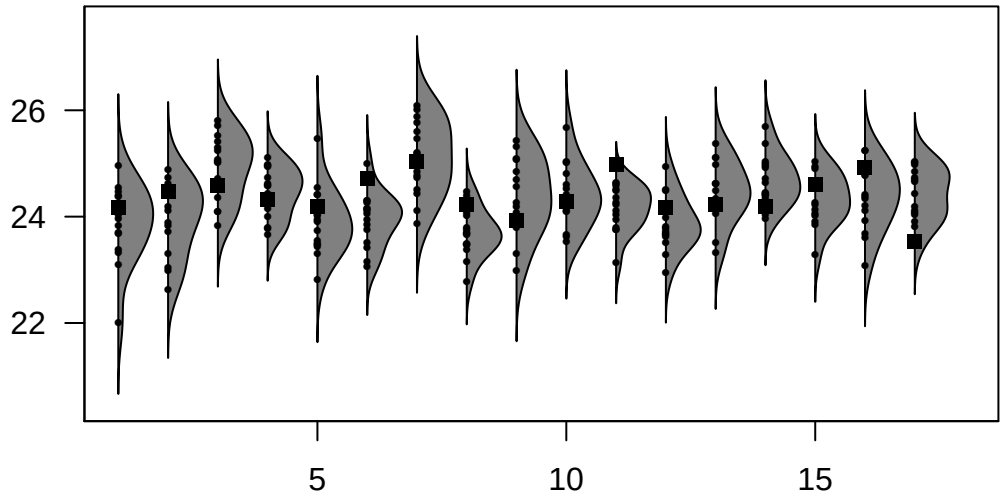

Figure 7: Plot of the dressed ECMWF ensemble

$$m_k(e) = r_1 + r_2\bar{e} + ae_k$$
$$s_k^2(e) = \left(\frac{4}{3K}\right)^{2/5}\left(s_1 + s_2 a^2 \sigma_e^2\right)$$

The parameters of the two transformations can be specified by the user or can be estimated by minimum CRPS estimation.

A few examples illustrate the capabilities of the method. The following behaves like the standard Silverman method, but the ensemble means are shifted by three degrees:

```
shift <- 3
d.ens <- DressEnsemble(ecmwf, dressing.method="akd",
                       parameters=list(r1=shift, r2=0, a=1, s1=0, s2=1))
PlotDressedEns(d.ens, plot.ens=TRUE, obs=ver)
```

The following uses very narrow kernels:

```
d.ens <- DressEnsemble(ecmwf, dressing.method="akd",
                       parameters=list(r1=0, r2=0, a=1, s1=0, s2=0.01))
PlotDressedEns(d.ens, plot.ens=TRUE, obs=ver)
```

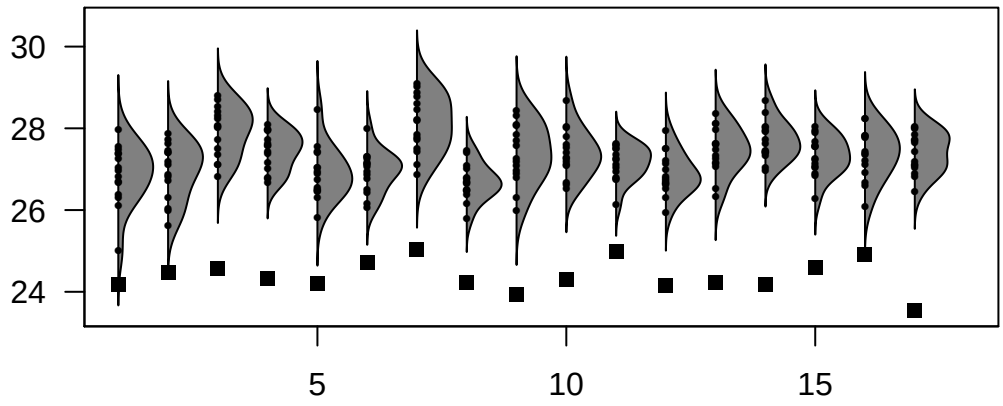

Figure 8: Shifted ensemble

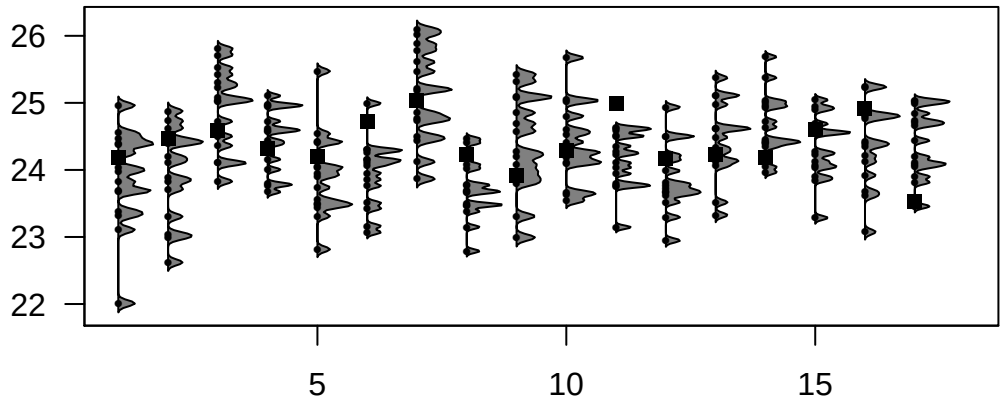

Figure 9: Narrow kernels

The AKD parameters can be calculated by minimum CRPS estimation, for which a set of observations must be provided:

```
d.ens <- DressEnsemble(ecmwf, dressing.method="akd.fit",
                       parameters=list(obs=ver))
PlotDressedEns(d.ens, plot.ens=TRUE, obs=ver)
```

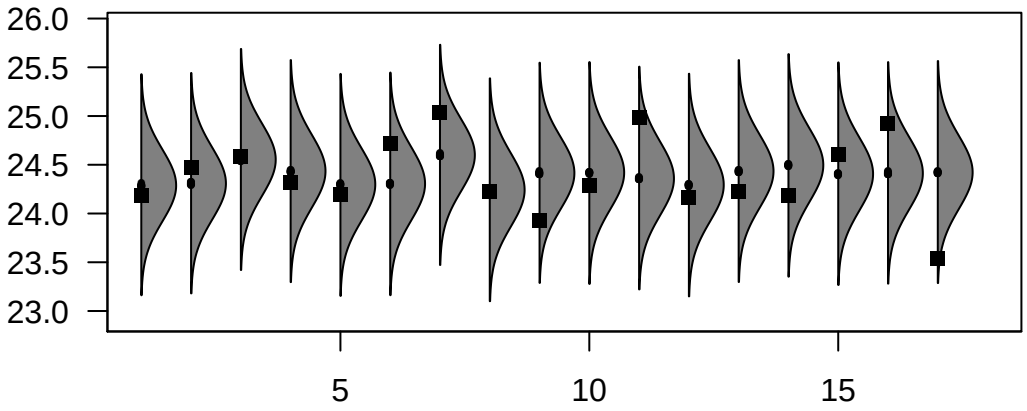

Figure 10: Minimum CRPS parameters

The function `FitAkdParameters` returns the actual parameter values:

```
FitAkdParameters(ecmwf, ver)
```

```
R output >>          a        r1        r2        s1        s2
R output >>    -0.0183   18.2725    0.2703    0.3799 -123.1538
```

which basically shows that the influence of the individual ensemble members is small ($a \approx 0$) and that the influence of the ensemble mean and the ensemble standard deviation is small ($r_2 \ll r_1$ and $s_2 a^2 \ll s1$).

The functions `DressCrps` and `DressIgn` calculate the continuously ranked probability score and Ignorance score of a collection of dressed ensembles and their verifications:

```
DressCrps(d.ens, obs)
```

```
R output >>   [1] 3.020 2.440 3.244 1.986 4.026 2.125 3.926 1.422 2.550 2.231 3.580
R output >>  [12] 2.304 3.894 1.725 3.369 1.295 3.244
```

```
DressIgn(d.ens, obs)
```

```
R output >>   [1] 55.08 36.85 62.07 24.65 93.62 27.94 90.09 13.55 40.14 30.82 73.17
R output >>  [12] 32.44 87.19 19.12 65.63 11.70 61.22
```

The CRPS and Ignorance differences between two dressed ensemble forecasts can be analyzed:

```
l00w <- ens[t,4,'l00w',1:5]
d.ens.new <- DressEnsemble(l00w, dressing.method="akd.fit",
                           parameters=list(obs=ver))
DressCrpsDiff(d.ens.new, d.ens, ver, probs=c(0.05, 0.95))
```

```
R output >>  $crps.diff
R output >>  [1] 0.005817
R output >>
R output >>  $sampling.quantiles
R output >>      0.05     0.95
R output >>  -0.01107  0.02270
```

```
DressIgnDiff(d.ens.new, d.ens, ver, probs=c(0.05, 0.95))
```

```
R output >>  $ign.diff
R output >>  [1] 0.02231
R output >>
R output >>  $sampling.quantiles
R output >>     0.05    0.95
R output >>  -0.1024  0.1470
```

The average score differences are slightly larger than zero but the sampling fluctuation of the score is probably too large to make a definite statement about which ensemble is better. It is important to realize that the above analysis is not an evaluation of the ensemble forecasts themselves, but an evaluation of the ensemble forecast *and* the statistical machinery that transformed the ensemble into a smooth pdf.