Software Tutorial: Verification and post-processing

Alexander Jordan, Sebastian Lerch

ScienceFore Summer School, October 2017

Some introduction?

absolute forecast evaluation, scoringRules introduction, some simulations?

Data

Set path to directory containing the data

```
data_dir <- "/path/to/data/"
```

and load the data set

```
load(paste0(data_dir, "HDwind.Rdata"))
```

Contents of the data set

The data set contains objects `ensfc`, `obs` and `dates`

The vector `dates` contains dates in 2015 and 2016, at which the forecasts and corresponding observations are valid (at 12 UTC).

```
str(dates)
```

```
##  Date[1:731], format: "2015-01-01" "2015-01-02" "2015-01
```

```
range(dates)
```

```
## [1] "2015-01-01" "2016-12-31"
```

Contents of the data set: Forecasts

The matrix `ensfc` contains ECMWF ensemble forecasts of wind speed for a grid point close to Heidelberg with a lead time of 60 hours (2.5 days).

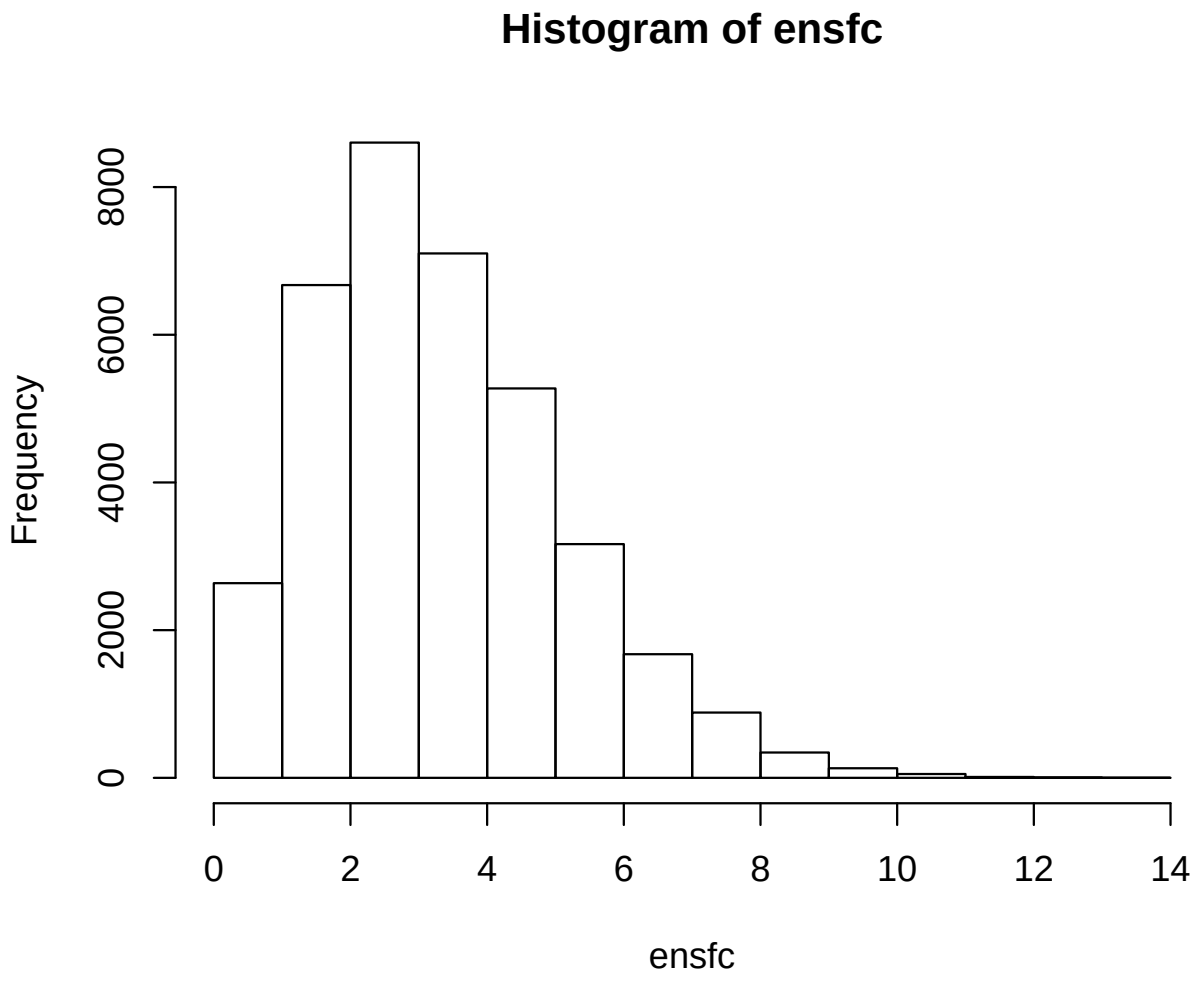
```
str(ensfc)
```

```
##  num [1:731, 1:50] 1.43 4.66 3.34 3.49 1.82 ...
```

For each of the 731 dates, an ensemble with 50 member forecasts is available.

Contents of the data set: Forecasts

```
hist(ensfc)
```



Contents of the data set: Observations

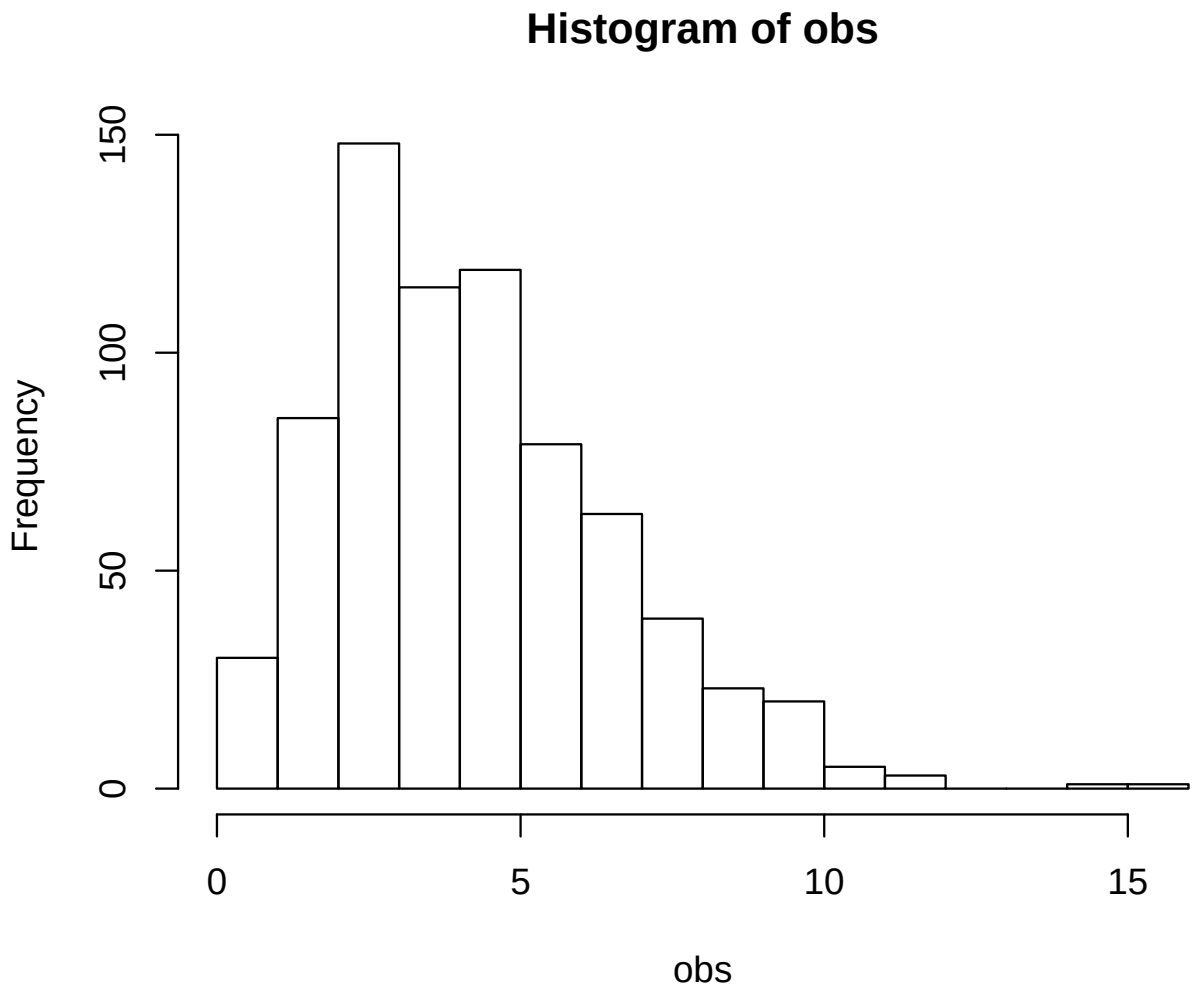
The vector `obs` contains reanalysis values corresponding to the valid times of the ensemble forecasts.

```
str(obs)
```

```
##  num [1:731] 2.21 7.93 4.92 5.02 3.37 ...
```

Contents of the data set: Observations, continued

```
hist(obs, breaks = seq(0, max(ceiling(obs)), 1))
```



The scoringRules package

To install the `scoringRules` package

```
install.packages("scoringRules")
```

To load the `scoringRules` package

```
library(scoringRules)
```

Check if version is $\geq$ 0.9.3

```
packageVersion("scoringRules") >= "0.9.3"
```

```
## [1] TRUE
```

If this is not the case, re-install the package from CRAN.

Documentation of the scoringRules package

The documentation of individual functions can be accessed via e.g.

```
?crps_sample
```

To browse the documentation of the functions available in the package use

```
help.start()
```

and navigate to 'packages' - 'scoringRules'

Vignettes with introductions and background information provided with the package can be accessed via

```
browseVignettes("scoringRules")
```

More information is available in our working paper 'Evaluating probabilistic forecasts with the R package scoringRules' available at https://arxiv.org/abs/1709.04743.

Compute the CRPS of the ensemble forecast

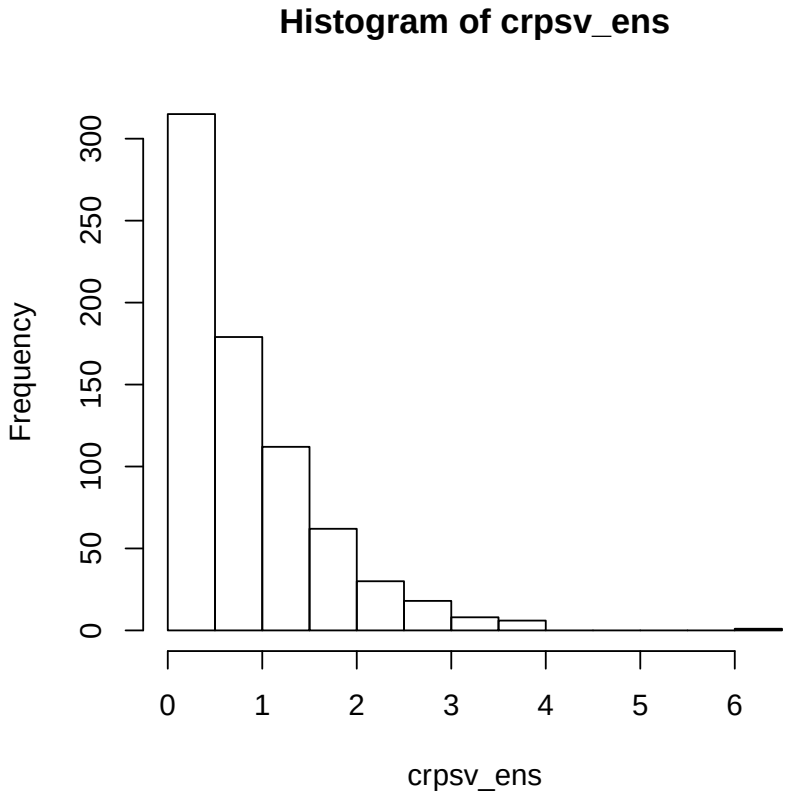
.. or: show simulation examples and use this as exercise?

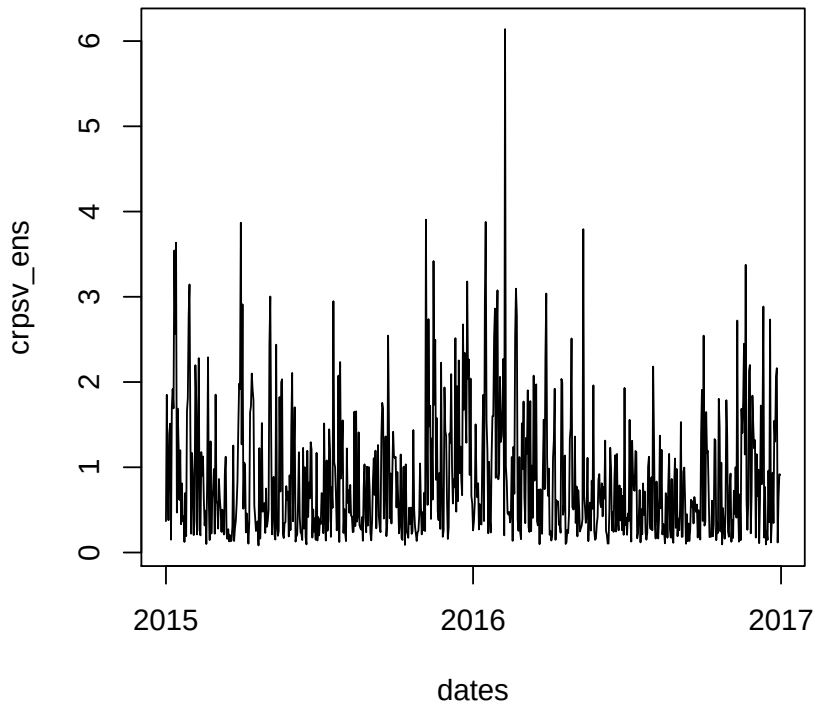
```
crpsv_ens <- crps_sample(y = obs, dat = ensfc)
summary(crpsv_ens)
```

```
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.0832  0.2746  0.5988  0.8583  1.1984  6.1404
```

Plot CRPS values

```
par(mfrow = c(1,2))
hist(crpsv_ens)
plot(dates, crpsv_ens, type = "l")
```





Post-processing example

Aim: Fit a simple EMOS model for wind speed to data of 2016.

As a basic example, we will start by fitting a truncated normal model of the form

$$Y|X_1, \ldots, X_m \sim \mathcal{N}_{[0,\infty)}(\mu, \sigma^2),$$

where the location parameter μ is a linear function of the ensemble mean,

$$\mu = a + b\bar{X},$$

and the scale parameter σ is assumed to be constant,

$$\sigma^2 = c$$

Estimation - Training set

We estimate the model parameters a, b, c based on training data from pairs of forecasts and observations in 2015, and evaluate the out of sample forecasts in year 2016.

The parameters are estimated by numerically minimizing the mean CRPS over the training set.

Let's start by extracting the training data:

```
ind_training <- which(dates <= "2015-12-29")
ensfc_mean_training <- apply(ensfc[ind_training,],
                             1, mean)
obs_training <- obs[ind_training]
```

Estimation - Objective function

Next, we define an objective function we aim to minimize. The objective function computes the (aggregated) CRPS over the training period, as a function of a, b, c.

```
objective_fun_minCRPS <- function(par, ens_mean_train,
                                  obs_train){
  m <- cbind(1, ens_mean_train) %*% par[1:2]
  s <- sqrt(par[3])
  return(sum(crps_tnorm(y = obs_train,
                        location = m, scale = s,
                        lower = 0, upper = Inf)))
}
```

In the objective function, we use a function from the `scoringRules` package to compute the CRPS of the truncated normal distribution, to see details, try

```
?crps_tnorm
```

Estimation - Numerical optimization

Now, we can estimate optimal values of a, b, c using the numerical optimization function `optim()`

```
optim_out <- optim(par = c(1,1,1), # starting values
                   fn = objective_fun_minCRPS,
                   ens_mean_train = ensfc_mean_training,
                   obs_train = obs_training)
```

To view details on the outcome, take a look at `optim_out`. The optimal parameters can be accessed via

```
optim_out$par
```

```
## [1] -0.09082381  1.29780652  0.97667948
```

Out of sample evaluation - Preparations

First, let' save the optimal parameter values we just determined:

```
opt_par <- optim_out$par
```

Now, we extract ensemble forecasts and observations during the evaluation period (2016):

```
ind_2016 <- which(dates >= "2016-01-01")
obs_2016 <- obs[ind_2016]
ensfc_2016 <- ensfc[ind_2016, ]
```

Out of sample evaluation - Forecast distribution parameters

Recall that in our truncated normal model,

$$\mu = a + b\bar{X} \qquad \sigma^2 = c.$$

Based on the estimated parameters a, b, c in `opt_par`, we can now compute the values of μ and σ for 2016.

```
ens_mean_2016 <- apply(ensfc[ind_2016,], 1, mean)
tn_mu <- c(cbind(1, ens_mean_2016) %*% opt_par[1:2])
tn_sigma <- sqrt(opt_par[3])
```

Based on `tn_mu` and `tn_sigma`, we can now compute the CRPS for our post-processing model.

```
crps_emos <- crps_tnorm(y = obs_2016,
                        location = tn_mu, scale = tn_sigma,
                        lower = 0, upper = Inf)
```

Out of sample evaluation - Scores

For comparison, we also compute the CRPS of the raw ensemble in 2016.

```
crps_ens <- crps_sample(y = obs_2016, dat = ensfc_2016)
```

Now, we can compare the raw and post-processed ensemble forecasts

```
mean(crps_ens)
```

```
## [1] 0.8312655
```

```
mean(crps_emos)
```

```
## [1] 0.54303
```

Out of sample evaluation - Results

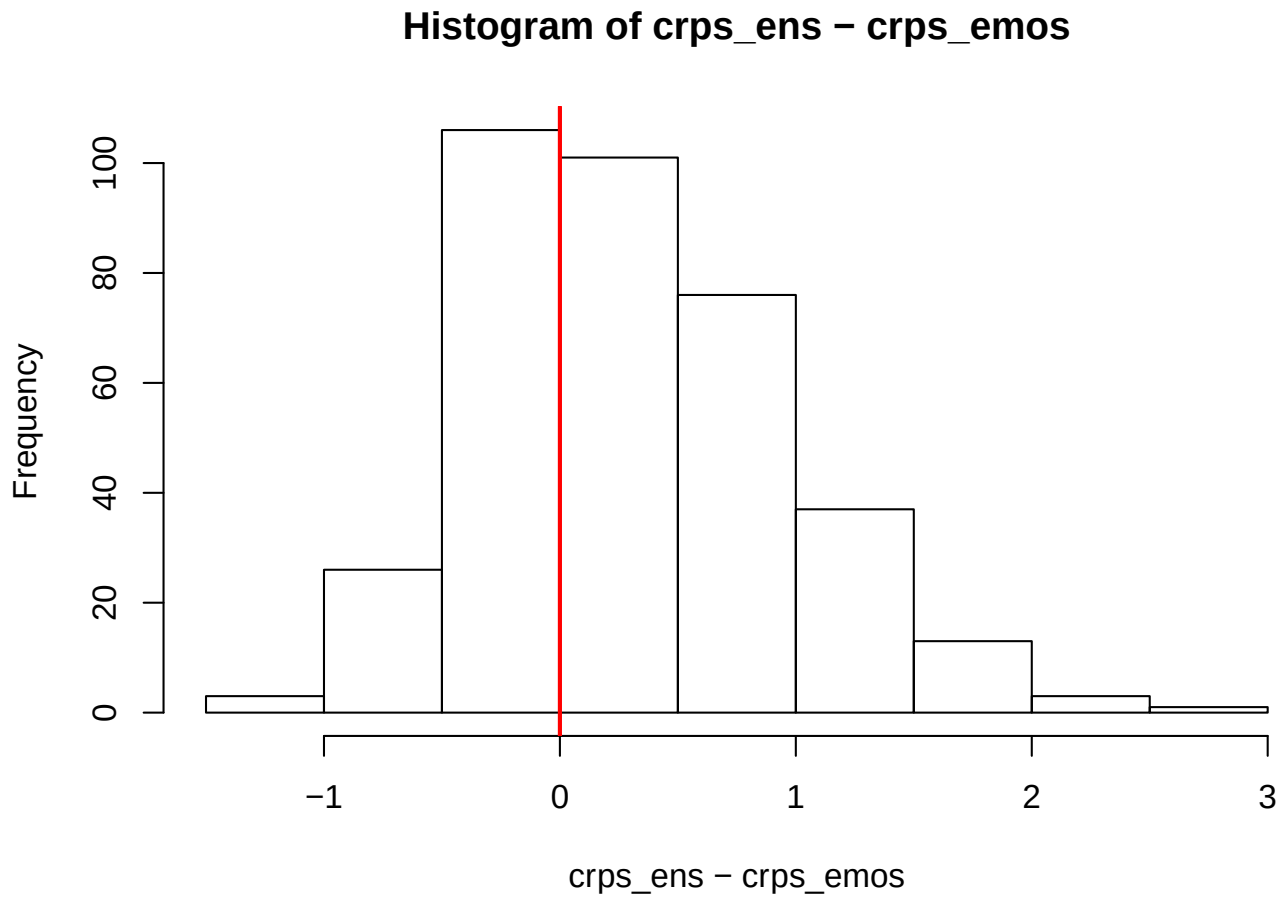
Post-processing substantially improves the raw ensemble forecasts:

```
summary(crps_ens - crps_emos)
```

```
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.3920 -0.1881  0.2626  0.2882  0.6544  2.5712
```

Out of sample evaluation - Results, continued

```
hist(crps_ens - crps_emos)
abline(v = 0, col = "red", lwd = 2)
```



Out of sample evaluation - Calibration

To compute PIT values, we need a function to compute the CDF of a truncated normal distribution

```
ptnorm0 <- function(q, mu, sig){
  t1 <- pnorm((q-mu)/sig) - pnorm(-mu/sig)
  t2 <- 1 - pnorm(-mu/sig)
  return(t1/t2)
}
```

Now, we can compute the PIT values

```
pit_emos <- ptnorm0(obs_2016, tn_mu, tn_sigma)
```

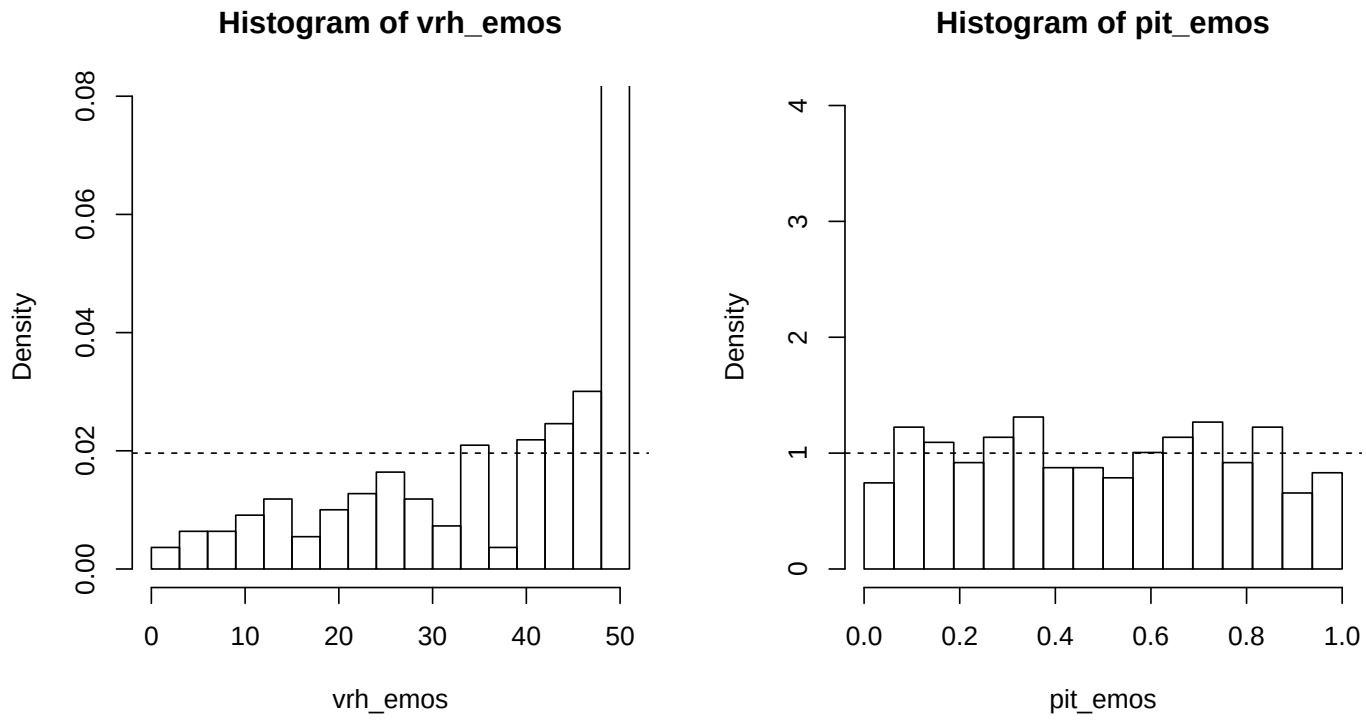
Out of sample evaluation - Calibration, continued

Similarly, we need to compute the verification ranks of the observation when pooled within the raw ensemble forecasts.

```
vrh_emos <- numeric(length = length(obs_2016))
for(i in 1:length(obs_2016)){
  vrh_emos[i] <- rank(c(obs_2016[i], ensfc_2016[i,]))[1]
}
```

Out of sample evaluation - Calibration, continued

```
par(mfrow=c(1,2))
hist(vrh_emos, freq = FALSE, breaks = seq(0, 51, 3),
     ylim = 1/51*c(0,4)); abline(h = 1/51, lty = 2)
hist(pit_emos, breaks = seq(0, 1, length.out = 17),
     freq = FALSE, ylim = c(0,4)); abline(h = 1, lty = 2)
```



Exercise

Build your own post-processing model. Try to outperform our simple benchmark model for data from 2016.

Exercise

Build your own post-processing model. Try to outperform our simple benchmark model for data from 2016.

Some hints for potential improvements:

- what other information from the ensemble be used?
- are there alternative ways to select the training period?
- are there other parametric families that might be useful for wind speed?