Notes on prediction assessment

Adapted from MATH10093: Statistical Computing

Finn Lindgren, The University of Edinburgh, July 2020

- Reminders of some probability theory
- Probabilistic prediction
- Proper scoring rules
- Cross validation, bootstrap, and formal score comparisons

Some essential and useful probability theory

Some essential and useful probability theory

Expectation and variance

$$\mathsf{E}_{Y\sim F}[h(Y)] = \sum_{k\in K} h(k)\, f_Y(k), \quad \text{prob. mass fcn } f_Y(\cdot) = \mathsf{P}_F(Y=k), \text{ discrete outcomes } K$$

$$\mathsf{E}_{Y\sim F}[h(Y)] = \int_D h(y) f_Y(y)\,\mathrm{d}y, \quad \text{prob. density fcn } f_Y(\cdot), \text{ continuous outcomes } D$$

$$\mathsf{Var}_{Y\sim F}(Y) = \mathsf{E}_F\left\{[Y - \mathsf{E}_F(Y)]^2\right\} = \mathsf{E}_F(Y^2) - \mathsf{E}_F(Y)^2$$

- When it's clear which variable the expectation/variance is taken for, the $Y \sim F$ notation is often simplified to just F, as in $\mathsf{E}_F(Y) = \mathsf{E}_{Y\sim F}(Y)$.
- When it's clear which distribution the random variables have but the expectations may be taken over different variables, the F may be dropped instead.

Some essential and useful probability theory

Law of total expectation, or "The tower property"

$$\mathsf{E}_Y(Y) = \mathsf{E}_X\left[\mathsf{E}_{Y|X}(Y \mid X)\right]$$
$$\mathsf{Var}_Y(Y) = \mathsf{E}_X\left[\mathsf{Var}_{Y|X}(Y \mid X)\right] + \mathsf{Var}_X\left[\mathsf{E}_{Y|X}(Y \mid X)\right]$$

Bayesian credible intervals

A Bayesian $(1-\alpha)\cdot 100$ *percent credible interval* $\mathrm{CI}_\theta = (a, b)$ for a parameter θ is an interval computed from the posterior distribution, such that $\mathrm{P}(a < \theta < b \mid \boldsymbol{y}) \geq 1-\alpha$.

- Often, a and b are chosen so that the tail probabilities outside the interval are both $= \alpha/2$.
- Another option is to choose the smallest set or interval with the required probaiblity. This is achieved by finding a value c such that the set $\mathrm{CI}_\theta = \{\theta; p(\theta|\boldsymbol{y}) > c\}$ fulfils $\mathrm{P}(\theta \in \mathrm{CI}_\theta|\boldsymbol{y}) \geq 1-\alpha$. The resulting CI_θ is called a *highest posterior density* (HPD) region, and is well defined also for multi-dimensional parameter vectors.
- When c is chosen as large as possible, the HPD is the smallest credible region possible for the given model parameterisation. However, due to how probability densities are changed under transformation of the random variable, HPD region constructions are not invariant under reparameterisation.

Credible region vs. Confidence region

Posterior credible region (Bayesian concept)

A *level* $1-\alpha$ *posterior credible region* for θ, $C_\theta(\boldsymbol{y})$, is a set (usually an interval) such that

$$\mathrm{P}_{\theta\sim p(\theta|\boldsymbol{y})}\left(\theta \in C_\theta(\boldsymbol{y})\right) \geq 1-\alpha$$

for the fixed set of observations $\boldsymbol{y}$. The probability is a direct statement about our posterior beliefs about the random variable θ.

Confidence region (Frequentist concept)

A *level* $1-\alpha$ *confidence region* procedure for θ, $C_\theta(\boldsymbol{y})$, generates sets (usually intervals) in such a way that

$$\mathrm{P}_{\boldsymbol{y}\sim p(\boldsymbol{y};\theta)}\left(\theta \in C_\theta(\boldsymbol{y})\right) \geq 1-\alpha$$

for every possibe θ (or at least for the true value). The probability statement concerns the confidence region construction *procedure* under repeated experiments; the observations $\boldsymbol{y}$ are random.

Prediction and scoring rules

Prediction and proper scoring rules

- When using numerical estimation methods subject to numerical precision errors, random data collection variation, and methodological approximations, it's useful to be able to assess the end result in a way that's not tied to a particular method or model.
- One approach: *split* the data into *observations for estimation* and *test* data: $\mathcal{Y}_{\text{obs}}$ and $\mathcal{Y}_{\text{test}}$.
- Estimate the model parameters using the *estimation* data $\mathcal{Y}_{\text{obs}}$.
- Assess how good the estimated model is at predicting the values of the *test* data $\mathcal{Y}_{\text{test}}$.
- The most common assessment is to compare *point predictions* with their corresponding actual value in the test data:
  Squared error $= (y_{\text{test}} - \widehat{y})^2$, where $\widehat{y} = \mathsf{E}_F(y)$, the expectation under the prediction distribution F.
- To assess *methodology*, it's often useful to use *simulated* data, so that we know the true model. If the method is able to come close to the true values, we are more confident that it will also work on data where we don't know the true model.
- When the goal is to compare different *models*, care must be taken if the models are estimated in different inference frameworks or with different estimation methods. This is to avoid the comparison actually comparing the different software implementations instead of the models themselves.

Simple estimation/test assessment

Consider the model $y_i \sim \mathsf{N}(\theta_1, \theta_2^2)$, independent y_i, $i = 1, \ldots, n$.

```
## Simulate data:
N <- 100
Y <- rnorm(N, mean = 10, sd = 2)

## Split the data, 75 for estimation, 25 for testing:
n <- c(75, 25)
obs <- sample(rep(c(TRUE, FALSE), c(75, 25)), size = N, replace = FALSE)  # Split randomly
y_obs <- Y[obs] # Extract the observations to be used for estimation
y_test <- Y[!obs] # Extract the observations to be used for testing

## Estimate the paramters:
theta1_hat <- mean(y_obs)
theta2_hat <- sd(y_obs)

## Test:
mean( (y_test - theta1_hat)^2 ) ## Average squared error

## [1] 5.306928

mean( (y_test - 0)^2 ) ## Average squared error for a bad model estimate

## [1] 110.9462
```

Forecasting and prediction

- The term *forecasting* typically means doing a *prediction* of future events or values, e.g. for weather forecasts
- Based on some statistical model and observed data, we typically construct a *point estimate* that is our best guess of the future value. Ideally, we also compute some measure of *uncertainty* about how large we expect the error to be, i.e. the difference between the point estimate and the true future value.
- In statistical terminology, this process is called *prediction*, and we seek useful *predictive distributions* that encode our knowledge from a statistical model and previously observed data into a representative distribution of possible future data values.
- Note: In Bayesian statistics, *prediction* (distributions and prediction intervals) can apply to *any* quantity that has not yet been observed. In frequentist statistics, fixed but unknown parameter values are instead associated with *confidence intervals*, and *prediction intervals* are reserved for observable random quantities.
- We will gloss over the differences between frequentist and Bayesian approaches, and focus on prediction of observable data values.

Predictive distributions

- From asymptotic frequentist likelihood theory it is known that, approximately, $\widehat{\boldsymbol{\theta}} - \boldsymbol{\theta}_{\text{true}} \sim \mathsf{N}(\mathbf{0}, \boldsymbol{\Sigma}_\theta)$, where $\boldsymbol{\Sigma}_\theta^{-1} \approx H(\widehat{\boldsymbol{\theta}})$, the negated log-likelihood Hessian.
- In Bayesian settings, we represent the uncertainty by a distribution of *potential* parameter values, e.g. if the posterior distribution is Gaussian, $\boldsymbol{\theta} \sim \mathsf{N}(\widehat{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_\theta)$, with density written as $f_{\boldsymbol{\theta}}(\boldsymbol{\theta})$
- The conditional density for the observable values y is written as $f_{y|\boldsymbol{\theta}}(y)$
- The *predictive density* $f_y(\cdot)$ is then obtained by integrating over $\boldsymbol{\theta}$:

$$f_y(y) = \int_{\mathbb{R}} f_{y|\boldsymbol{\theta}}(y) f_{\boldsymbol{\theta}}(\boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{\theta}$$

- We will identify the predictive distribution with the CDF F, $F(x) = \mathsf{P}_{y\sim F}(y \leq x) = \int_{-\infty}^{x} f_y(y) \, \mathrm{d}y$.
- Often, we only consider the predictive expectation and variance:

$$\mu_F = \mathsf{E}_{y\sim F}(y), \qquad \sigma_F^2 = \mathsf{Var}_{y\sim F}(y)$$

Example: Linear expectation and log-linear variance model

- Let $y \sim \mathrm{N}\left[\boldsymbol{z}_E^\top \boldsymbol{\theta}, \exp(\boldsymbol{z}_V^\top \boldsymbol{\theta})\right]$, i.e. the expectation has a linear model, and the variance has a log-linear model, where the same parameters could potentially influence both the expectation and the variance.
- With the $\boldsymbol{z}_{\cdot}$ vectors for each observation stored as rows in two matrices $\boldsymbol{Z}_E$ and $\boldsymbol{Z}_V$, the vector of observation expectations can be written as $\mathrm{E}_{y|\theta}(\boldsymbol{y}) = \boldsymbol{Z}_E\boldsymbol{\theta}$, and similarly for the log-variances.

Example: Take the special case $\boldsymbol{z}_{E_i}^\top = \begin{bmatrix}1 & x_i & 0 & 0\end{bmatrix}$ and $\boldsymbol{z}_{V_i}^\top = \begin{bmatrix}0 & 0 & 1 & x_i\end{bmatrix}$, i.e. the parameters for expectation and log-variance are not directly coupled, and we have an intercept and a single covariate for both of the submodels.

```
model_Z <- function(x) {
  Z0 <- model.matrix(~ 1 + x)
  list(ZE = cbind(Z0, Z0 * 0), ZV = cbind(Z0 * 0, Z0))
}
```

This takes a vector of covariate values x_i, one for each observation, as input, and return a list, with $\boldsymbol{Z}_E$ and $\boldsymbol{Z}_V$ as named elements.

Example: Predictive distribution

- Using numerical optimisation on the negative loglikelihood,

```
neg_log_lik <- function(theta, Z, y) {
  -sum(dnorm(y, mean = Z$ZE %*% theta, sd = exp(Z$ZV %*% theta)^0.5, log = TRUE))
}
```

  provides $\widehat{\boldsymbol{\theta}}$ and $\boldsymbol{\Sigma}_\theta$.

- The *tower property* ($\mathsf{E}_A(A) = \mathsf{E}_B[\mathsf{E}_{A|B}(A)]$) gives

$$\mathsf{E}_F(y) = \boldsymbol{z}_E^\top \widehat{\boldsymbol{\theta}}, \qquad \mathsf{Var}_F(y) = \mathsf{E}_\theta\left[\exp\left(\boldsymbol{Z}_V\boldsymbol{\theta}\right)\right] + \mathsf{Var}_\theta\left(\boldsymbol{z}_E^\top\boldsymbol{\theta}\right).$$

  The second term of the variance is

$$\mathsf{Var}_\theta\left(\boldsymbol{z}_E^\top\boldsymbol{\theta}\right) = \mathsf{Cov}_\theta\left(\boldsymbol{z}_E^\top\boldsymbol{\theta}, \boldsymbol{z}_E^\top\boldsymbol{\theta}\right) = \boldsymbol{z}_E^\top \mathsf{Cov}_\theta\left(\boldsymbol{\theta},\boldsymbol{\theta}\right)\boldsymbol{z}_E = \boldsymbol{z}_E^\top\boldsymbol{\Sigma}_\theta\boldsymbol{z}_E.$$

  For the first term, we use a result that says that if $x \sim \mathsf{N}(\mu, \sigma^2)$, then $\mathsf{E}(\mathrm{e}^x) = \mathrm{e}^{\mu+\sigma^2/2}$:

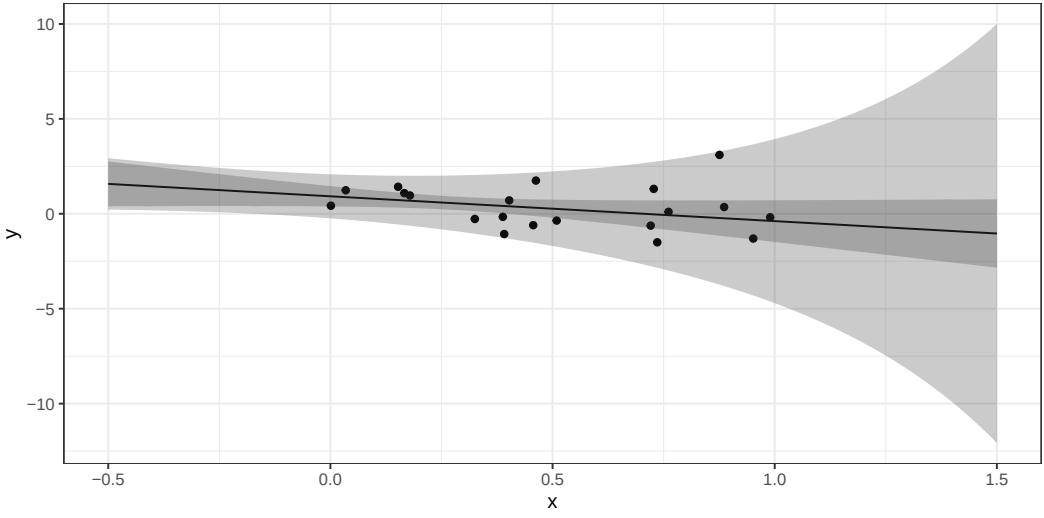
$$\mathsf{E}_\theta[\exp(\boldsymbol{z}_V^\top\boldsymbol{\theta})] = \exp\left(\boldsymbol{z}_V^\top\widehat{\boldsymbol{\theta}} + \boldsymbol{z}_V^\top\boldsymbol{\Sigma}_\theta\boldsymbol{z}_V/2\right).$$

  Combining the results, we get the predictive variance as

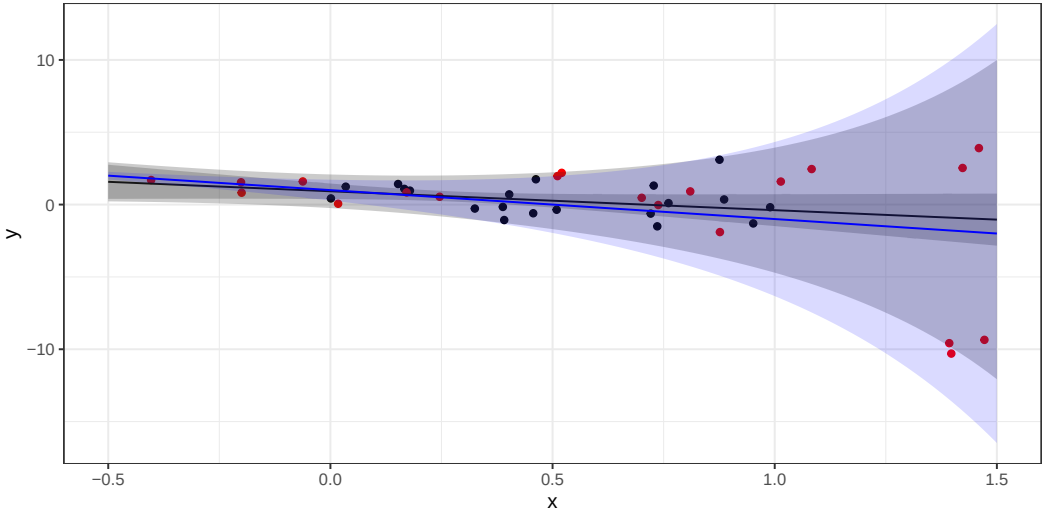
$$\sigma_V^2 = \mathsf{Var}_F(y) = \exp\left(\boldsymbol{z}_V^\top\widehat{\boldsymbol{\theta}} + \boldsymbol{z}_V^\top\boldsymbol{\Sigma}_\theta\boldsymbol{z}_V/2\right) + \boldsymbol{z}_E^\top\boldsymbol{\Sigma}_\theta\boldsymbol{z}_E.$$

```r
# Value: data.frame with columns (mu, sigma, lwr, upr)
model_predict <- function(theta, data, Sigma_theta = NULL,
                          type = c("expectation", "log-variance", "observation"),
                          alpha = 0.05, df = Inf,
                          nonlinear.correction = TRUE) {
  type <- match.arg(type)
  Z <- model_Z(data) ## Note: Will use model_Z() defined in the global workspace!
  fit_E <- Z$ZE %*% theta
  fit_V <- Z$ZV %*% theta
  if (is.null(Sigma_theta)) {
    ZE_var <- ZV_var <- 0
  } else {
    ZE_var <- rowSums(Z$ZE * (Z$ZE %*% Sigma_theta))
    ZV_var <- rowSums(Z$ZV * (Z$ZV %*% Sigma_theta))
  }
  if (type == "expectation") { ## confidence interval
    ...
  } else if (type == "observation") { ## observation predictions
    fit <- fit_E
    sigma <- (exp(fit_V + ZV_var / 2 * nonlinear.correction) + ZE_var)^0.5
  }
  q <- qt(1 - alpha / 2, df = df) ## If the user wants a t-quantile instead of Normal.
  lwr <- fit - q * sigma
  upr <- fit + q * sigma
  data.frame(mu = fit, sigma, lwr, upr)
}
```

Estimation data, point estimates, confidence and prediction intervals



Add the true model&predictions, and some test data



Scores

- We want to quantify how well our predictions represent the test data.
- We define *scores* $S(F, y)$ that in some way measure how well the prediction F matched the actual value, y.
- The scores defined here are *negatively oriented*, meaning that the *lower the score, the better*.
- We first define some common scores that are based on expectations, variances, and log-likelihoods.

Squared errors and log-likelihood scores

- Squared Error (SE): $S_{\text{SE}}(F, y) = (y - \widehat{y}_F)^2$,
  where $\widehat{y}_F$ is a point estimate under F, e.g. the expectation μ_F.
- Logarithmic/Ignorance score (LOG/IGN): $S_{\text{LOG}}(F, y) = -\log f(y)$,
  where $f(\cdot)$ is the predictive density.
- Dawid-Sebastiani (DS): $S_{\text{DS}}(F, y) = \frac{(y-\mu_F)^2}{\sigma_F^2} + \log(\sigma_F^2)$.

The SE score is extremely popular, in particular as root mean squared error (RMSE), which is the square root of the average SE score for a set of predictions. However, it has the major limitation that it does not take the prediction uncertainty into account.

Example

We evaluate the SE and DS scores for the true model, a naive simplistic reference model, and the estimated full model.

```
## For the estimation data:
obs_pred_true <- model_predict(theta_true, x_obs, type = "observation")
obs_pred_ref <- data.frame(mu = mean(y_obs), sigma = sd(y_obs))
obs_pred_hat <- model_predict(theta_hat, x_obs, Sigma_theta = Sigma_theta, type = "observation")

## For the test data:
pred_true <- model_predict(theta_true, x_test, type = "observation")
pred_ref <- data.frame(mu = mean(y_obs), sigma = sd(y_obs))
pred_hat <- model_predict(theta_hat, x_test, Sigma_theta = Sigma_theta, type = "observation")
```

Example

```r
# Input:
#   pred : data.frame with (at least) a column "mu"
#   y : data vector
score_se <- function(pred, y) {
  (y - pred$mu)^2
}

## SE for observed and test data
rbind(
  obs = c(true = mean(score_se(obs_pred_true, y_obs)),
          ref = mean(score_se(obs_pred_ref, y_obs)),
          hat = mean(score_se(obs_pred_hat, y_obs))),
  test = c(true = mean(score_se(pred_true, y_test)),
           ref = mean(score_se(pred_ref, y_test)),
           hat = mean(score_se(pred_hat, y_test)))
)
```

```
##           true       ref       hat
## obs   1.429616  1.247714  1.215824
## test 13.800697 17.275713 14.711117
```

Example

```r
# Input:
#   pred : data.frame with (at least) columns "mu" and "sigma"
#   y : data vector
score_ds <- function(pred, y) {
  ((y - pred$mu) / pred$sigma)^2 + 2 * log(pred$sigma)
}

## DS for observed and test data
rbind(
  obs = c(true = mean(score_ds(obs_pred_true, y_obs)),
          ref = mean(score_ds(obs_pred_ref, y_obs)),
          hat = mean(score_ds(obs_pred_hat, y_obs))),
  test = c(true = mean(score_ds(pred_true, y_test)),
           ref = mean(score_ds(pred_ref, y_test)),
           hat = mean(score_ds(pred_hat, y_test)))
)
```

```
##          true       ref       hat
## obs  1.002367  1.222606 0.9429584
## test 2.202256 13.426209 2.0991537
```

Score expectations and proper scoring rules

- What functions of the predictive distributions are useful scores?
- We want to reward accurate (unbiased) and precise (small variance) predictions, but not at the expense of understating true uncertainty.
- First, we define the expectation of a score under a true distribution G s

$$S(F, G) = \mathsf{E}_{y \sim G}[S(F, y)]$$

Proper scores/scoring rules

A negatively oriented score is *proper* if it fulfils

$$S(F, G) \geq S(G, G).$$

A proper score that has equality of the expectations *only* when F and G are the same, $F(\cdot) \equiv G(\cdot)$, is said to be *strictly proper*.

The practical interpretation of this is that a proper score does not reward cheating; stating a lower (or higher) forecast/prediction uncertainty will not, on average, give a better score than stating the truth.

Proper scores: examples

$$\begin{aligned} S_{\mathsf{SE}}(F,G) &= \mathsf{E}_{y\sim G}[S_{\mathsf{SE}}(F,y)] = \mathsf{E}_{y\sim G}[(y-\mu_F)^2] = \mathsf{E}_{y\sim G}[(y-\mu_G+\mu_G-\mu_F)^2] \\ &= \mathsf{E}_{y\sim G}[(y-\mu_G)^2 + 2(y-\mu_G)(\mu_G-\mu_F) + (\mu_G-\mu_F)^2] \\ &= \mathsf{E}_{y\sim G}[(y-\mu_G)^2] + 2(\mu_G-\mu_F)\mathsf{E}_{y\sim G}[y-\mu_G] + (\mu_G-\mu_F)^2 \\ &= \sigma_G^2 + (\mu_G-\mu_F)^2 \end{aligned}$$

This is minimised when $\mu_F = \mu_G$. Therefore $S_{\mathsf{SE}}(F,G) \geq S_{\mathsf{SE}}(G,G) = \sigma_G^2$, so the score is proper. Is it strictly proper? No, since any distribution F with the same expectation as G minimises the expected score.

$$\begin{aligned} S_{\mathsf{DS}}(F,G) &= \mathsf{E}_{y\sim G}[S_{\mathsf{DS}}(F,y)] = \frac{\mathsf{E}_{y\sim G}[(y-\mu_F)^2]}{\sigma_F^2} + \log(\sigma_F^2) \\ &= \frac{\sigma_G^2 + (\mu_G-\mu_F)^2}{\sigma_F^2} + \log(\sigma_F^2) \end{aligned}$$

This is minimised when $\mu_F = \mu_G$ and $\sigma_F = \sigma_G$. Therefore $S_{\mathsf{DS}}(F,G) \geq S_{\mathsf{DS}}(G,G) = 1 + \log(\sigma_G^2)$, so the score is proper. Is it strictly proper? No, since any distribution F with the same expectation and variance as G minimises the expected score.

Absolute error and CRPS

In order to avoid large outliers dominating the prediction scores, sometimes scores based on absolute deviations are preferred over scores related to squared deviations.

Absolute error and Continuous Ranked Probability Score

- Absolute Error (AE): $S_{\text{AE}}(F, y) = |y - \widehat{y}_F|$, where $\widehat{y}_F$ is a point estimate under F, e.g. the *median* $F^{-1}(1/2)$.
- CRPS: $S_{\text{CRPS}}(F, y) = \int_{-\infty}^{\infty} \left[\mathbb{I}(y \leq x) - F(x)\right]^2 \, \mathrm{d}x$

Proper and strictly proper score summary:

- SE is proper with respect to the predictive expectation
- AE is proper with respect to the predictive median
- DS is proper with respect to the predictive expectation and variance
- The LOG and CRPS scores are strictly proper

Next, we look at how to assess prediction intervals, and then the special case of applying SE to binary event outcomes, called the Brier score.

How good are prediction interval procedures?

Tradeoffs for prediction intervals

Desired properties for methods generating prediction intervals (PIs) for a quantity Y:

1. Appropriate *coverage* under the true distribution, G: $\mathsf{P}_{Y\sim G}(Y \in \mathsf{PI}_F) \geq 1-\alpha$
2. Narrow intervals

Note: For confidence intervals similar properties are desired, but then the F are random.

- A wide prediction F helps with 1 but makes 2 difficult
- A narrow prediction F helps with 2 but makes 1 difficult

A proper score for interval predictions

The *Interval Score* For a PI (L_F, U_F) is defined by

$$S_{\mathsf{INT}}(F,y) = U_F - L_F + \frac{2}{\alpha}(L_F - y)\mathbb{I}(y < L_F) + \frac{2}{\alpha}(y - U_F)\mathbb{I}(y > U_F)$$

It is a proper scoring rule, consistent for equal-tail error probability intervals:
$S(F,G)$ is minimised for the narrowest PI that has expected coverage $1-\alpha$.

Assessing event prediction probabilities with LOG and Brier scores

Often, specific events are of particular interest. We can use an estimated general model to compute predictive probabilities for several events, or construct a model that is only aimed at predicting a specific event. The indicator variable $Z = \text{The event occurred}$ has a Binomial prediction distribution, $\text{Bin}(1, p_F)$. We can therefore use the Log-score, $S_{\text{LOG}}(F, z) = -z \log(p_F) - (1 - z) \log(1 - p_F)$, to assess predictive performance.

Brier score

The *Brier Score* is a another popular event prediction assessment score that for single events can be defined as the Squared Error score with respect to the Binomial expectation:

$$S_{\text{Brier}}(F, z) = (z - p_F)^2$$

For multi-option outcomes, where an observation is classified into one of several categories (instead of just "no/yes"), the score can be generalised to

$$S_{\text{Brier}}(F, z) = \sum_{k=1}^{K} [\mathbb{I}(z = k) - \mathsf{P}_F(Z = k)]^2$$

The Brier score is proper, but only strictly proper w.r.t. a specific set of event categories.

Mean/average scores

When we predict a whole set of data points, e.g. from a training/test split, we often summarise the scores by taking the average:

Mean/average score

Given a collection of prediction/truth pairs, $\{(F_i, y_i), i = 1, \ldots, n\}$, define the *mean* score:

$$\overline{S}(\{(F_i, y_i), i = 1, \ldots, n\}) = \frac{1}{n} \sum_{i=1}^{n} S(F_i, y_i)$$

- When comparing prediction quality, we often look at the difference in average scores across the test data set.
- When comparing two different models or prediction methods the *pairwise* differences are useful: For two prediction methods, F and F', define

$$S_i^{\Delta}(F_i, F_i', y_i) = S(F_i, y_i) - S(F_i', y_i)$$

We can have $\overline{S}^{\Delta} \approx 0$ at the same time as all $|S_i^{\Delta}| \gg 0$, if the two models/methods are both good, but in different situations.

Cross validation, Bootstrap, and Tests

Proper scores and data splits

- Recall the the expectation of a score under a true distribution G,

$$S(F, G) = \mathsf{E}_{y \sim G}[S(F, y)].$$

- A (negatively oriented) score is *proper* if $S(F, G) \geq S(G, G)$ for all predictions F.

Before, we split the data in *observation* and *test*. Generalised split:

- Observation/Estimation/Training — Data used to estimate a model
- Validation — Data for assessing estimates and taking modelling decisions
- Test — Data used in a final step to assess the resulting model

Decisions based on scores evaluated on Training or Validation data might lead to overestimation of the predictive ability.
Holding out a separate Test set provides a safer way of assessing predictive ability.

Basic score uncertainty

- We're interested in the expected average prediction test score $\overline{S}(F^{\text{test}}, G^{\text{test}})$
- Before looking at the Test data, we only have access to an *estimate* of $\overline{S}(F^{\text{test}}, G^{\text{test}})$, based on a training/validation data split:

$$\widehat{S}^{\text{valid}} = \frac{1}{N} \sum_{i=1}^{N} S(F_i^{\text{valid}}, y_i^{\text{valid}})$$

- Note: To investigate the *difference* in expected score between two models or methods, replace $S(F_i, y_i)$ by the pairwise differences $S_i^{\Delta} = S^{\Delta}(F_i, F_i', y_i) = S(F_i, y_i) - S(F_i', y_i)$ everywhere. This helps handle the inherent *dependence* between the prediction scores, since they involve the same observation y_i.
- The empirical variance estimate for $\widehat{S}^{\text{valid}}$ is

$$\widehat{\text{Var}}[\widehat{S}^{\text{valid}}] = \frac{1}{N(N-1)} \sum_{i=1}^{N} \left[S(F_i^{\text{valid}}, y_i^{\text{valid}}) - \widehat{S}^{\text{valid}} \right]^2$$

- The variance estimate may be biased due to dependence between the scores.

Cross validation

- We're interested in the expected average prediction test score $\overline{S}(F^{\text{test}}, G^{\text{test}})$ when using all the Training and Validation data to estimate the parameters of the final model.
- The Training set is a subset; may lead to overestimation of the expected score
- The Validation set is a small subset; high variability in the score estimator
- Different splits might give different score estimates and hence different modelling decisions
- Partial solution: Do multiple splits

K-fold Cross Validation: CV(K)

- Split the N data points $\mathcal{D}$ into K subsets $\mathcal{D}_k^{(\text{K})}$, each of size N/K.
- Iterate over the K subsets, treating each as a Validation set, $\mathcal{D}_k^{\text{valid}} = \mathcal{D}_k^{(\text{K})}$, and the remaining $K-1$ subsets as Training data $\mathcal{D}_k^{\text{train}} = \cup_{j \neq k} \mathcal{D}_j^{(\text{K})}$.
- Average over the resulting K score estimates.

Cross-validation scores

- For each of the K *folds*, the estimator of the expected score is

$$\widehat{S}_k^{(\mathsf{K})} = \frac{K}{N} \sum_{i=1}^{N/K} S(F_{ki}^{\mathsf{valid}}, y_{ki}^{\mathsf{valid}})$$

- The combined cross-validation score is

$$\widehat{S}^{\mathsf{CV}(\mathsf{K})} = \frac{1}{K} \sum_{k=1}^{K} \widehat{S}_k^{(\mathsf{K})}$$

- There are many options for estimating the variance of the combined CV score. Simple:

$$\widehat{\mathsf{Var}}[\widehat{S}^{\mathsf{CV}(\mathsf{K})}] = \frac{1}{K(K-1)} \sum_{k=1}^{K} [\widehat{S}_k^{(\mathsf{K})} - \widehat{S}^{\mathsf{CV}(\mathsf{K})}]^2$$

- No universal rule for what K and splitting choices will minimise the bias and variance of the estimators. Common choice is $K = 10$ and random splitting.

Common problem-dependent splitting options

- Leave-one-out CV; LOOCV=CV(N)
  In general very expensive, but for some model classes fast approximations are possible; Notably in Gaussian time series and spatial models
- Structured, only partially random, cross-validation examples:
  - Leave-station-out (to assess spatial predictive ability)
  - Leave-country-out (to assess macro scale generalisability, including potentially different measurement systems)
  - Leave-timepoint-out (to assess temporal interpolation ability)
  - Related non-cross-validation example: Leave-future-out (to assess forecasting ability)
  - Leave-subject-out (to assess generalisability of medical treatment between patients)
- Instead of complete splitting, do multiple Validation subset selections as random subsamples with replacement (related to Bootstrap)

Cross validation and Bootstrap

- Cross validation splits the data in K parts and performs model estimation on $(K-1)$ parts and validation assessment on the Kth part, all for each of the parts.
- Bootstrap resamples *with replacement* to obtain a random sample of the same size as the original sample.

Basic Bootstrap resampling

Let $Y = \{(y_i, x_i), i = 1, \ldots, N\}$ be a data collection with response values y_i and predictors/covariates x_i.

- Define a *Bootstrap sample* $Y^{(j)}$ by drawing N pairs (y_i, x_i) from Y with equal probability, and with replacement.
- Repeat this procedure for $j = 1, \ldots, J$, with $J \gg 1$.

The resampling procedure draws a random sample from the *empirical distribution* for the data collection.

The Bootstrap principle

- Each boostrap sample $Y^{(j)}$ can be used to apply some model estimation procedure, each generating a parameter estimate $\widehat{\theta}^{(j)}$.
- We want to use these bootstrapped estimates to say something about the properties of the estimator $\widehat{\theta}$ which is based on the original data Y.
- Idea: The parameter estimate as a deterministic function of the data, the *empirical parameter value* for the observed sample Y: $\widehat{\theta} = \theta(Y)$ and $\widehat{\theta}^{(j)} = \theta(Y^{(j)})$.

The Bootstrap principle

According to the *Bootstrap principle*, the errors of the bootstrapped estimates have the same distribution as the error of $\widehat{\theta}$. In particular, if the true parameter is θ_{true}, then

$$\mathrm{E}(\widehat{\theta} - \theta_{\text{true}}) = \mathrm{E}(\widehat{\theta}^{(j)} - \widehat{\theta}),$$
$$\mathrm{Var}(\widehat{\theta} - \theta_{\text{true}}) = \mathrm{Var}(\widehat{\theta}^{(j)} - \widehat{\theta}).$$

Bootstrap estimation

- The usual expectation and variance estimators can be used:

$$\widehat{\mathsf{E}}(\widehat{\theta} - \theta_{\mathsf{true}}) = \frac{1}{J}\sum_{j=1}^{J}(\widehat{\theta}^{(j)} - \widehat{\theta}) = \overline{\widehat{\theta}^{(\cdot)}} - \widehat{\theta}, \quad \widehat{\mathsf{Var}}(\widehat{\theta} - \theta_{\mathsf{true}}) = \frac{1}{J-1}\sum_{j=1}^{J}\left(\widehat{\theta}^{(j)} - \overline{\widehat{\theta}^{(\cdot)}}\right)^2.$$

- Bias adjusted estimator $\widehat{\theta} - \left(\overline{\widehat{\theta}^{(\cdot)}} - \widehat{\theta}\right)$. Properties? Need *double bootstrap*!
- Confidence intervals for θ_{true}: Consider the quantiles of the error distribution:
  Find a and b such that $\mathsf{P}(a < \widehat{\theta}^{(j)} - \widehat{\theta} < b) = 1 - \alpha$ from the empirical quantiles of the Bootstrap sample residuals $\widehat{\theta}^{(j)} - \widehat{\theta}$.
  According to the Bootstrap principle,

$$\mathsf{P}(a < \widehat{\theta}^{(j)} - \widehat{\theta} < b) = \mathsf{P}(a < \widehat{\theta} - \theta_{\mathsf{true}} < b),$$

  so that

$$\mathsf{P}(\widehat{\theta} - b < \theta_{\mathsf{true}} < \widehat{\theta} - a) = 1 - \alpha,$$

  and a confidence interval is given by

$$\mathsf{CI}_{\mathsf{boot}}(\theta) = \left(\widehat{\theta} - b, \widehat{\theta} - a\right)$$

Alternative confidence interval derivation

Instead of using the empirical quantiles of the boostrap *residuals*, we can use the samples themselves:
Find A and B such that $\mathsf{P}(A < \widehat{\theta}^{(j)} < B) = 1 - \alpha$ from the empirical quantiles of the Bootstrap samples $\widehat{\theta}^{(j)}$.
According to the Bootstrap principle,

$$\mathsf{P}(A < \widehat{\theta}^{(j)} - \widehat{\theta} < b) = \mathsf{P}(A - \widehat{\theta} < \widehat{\theta}^{(j)} - \widehat{\theta} < B - \widehat{\theta}) = \mathsf{P}(A - \widehat{\theta} < \widehat{\theta} - \theta_{\mathsf{true}} < B - \widehat{\theta}),$$

so that

$$\mathsf{P}(2\widehat{\theta} - B < \theta_{\mathsf{true}} < 2\widehat{\theta} - A) = 1 - \alpha,$$

and a confidence interval is given by

$$\mathsf{CI}_{\mathsf{boot}}(\theta) = \left(2\widehat{\theta} - B, 2\widehat{\theta} - A\right).$$

Remark:
In Bayesian statistics, *Credible Intervals* can be obtained from the empirical quantiles of samples from the *posterior distribution*. It could therefore be tempting to just use the empirical Bootstrap sample quantiles (A, B) as the interval, but this would not work in the presence of Bias or skewness of the Bootstrap distributions.

Parametric bootstrap

- If the data size is small, basic Boostrap that resamples with replacement from the raw data has very little information.
- An alternative is to sample entirely new data from the model that was estimated on the whole data set.

Parametric Bootstrap sampling

Let $Y = \{(y_i, x_i), i = 1, \ldots, N\}$ be a data collection with response values y_i and predictors/covariates x_i, such that the conditional data density function is $p(y_i|x_i, \theta)$. Let $\widehat{\theta}$ be the maximum likelihood estimate of θ.

- Define the *Bootstrap sample* $Y^{(j)}$ by drawing N pairs $(y_i^{(j)}, x_i)$, where for each x_i, $y_i^{(j)}$ is drawn from $p(y_i|x_i, \widehat{\theta})$.
- Repeat this procedure for $j = 1, \ldots, J$, with $J \gg 1$.

Depending on the problem, the x_i values can either be kept fixed to their values in the original data set, resampled with replacement, or simulated from some generative model.

Residual resampling

- We can relax model assumptions, e.g. if we don't trust a Gaussian assumption for regression residuals.
- Instead of resampling the raw data, we resample the model *residuals*.

Residual resampling in regression models

Let $Y = \{(y_i, x_i), i = 1, \ldots, N\}$ be a data collection with response values y_i and predictors/covariates x_i, such that $\mu_i = \mathsf{E}(y_i|x_i, \theta)$ is the conditional expectation in a regression model. Let $\widehat{\theta}$ be the maximum likelihood estimate of θ, and $\widehat{\mu}_i = \mathsf{E}(y_i|x_i, \widehat{\theta})$.

- Define the residuals $r_i = y_i - \widehat{\mu}_i$, and construct a *residual Bootstrap sample* $Y^{(j)}$ by drawing N pairs $(y_i^{(j)}, x_i)$, where for each x_i, $y_i^{(j)} = \widehat{\mu}_i + r_i^{(j)}$, where $r_i^{(j)}$ is drawn with replacement from the empirical distributon of $\{r_1, r_2, \ldots, r_N\}$.
- Repeat this procedure for $j = 1, \ldots, J$, with $J \gg 1$.

For x_i, the same options as for fully parametric Bootstrap are available.

- This is a simple example of an *exchangeability* assumption; we no longer assume Gaussianity, but we do assume that all the residuals come from some common, but unknown, distribution; the individual residuals are (probabilistically) indistinguishable.

Randomisation/permutation tests

When assessing differences between two statistical populations, the hypotheses often take the form of some kind of *exchangeability structure*.

Exchangeability test example

- Let $Y_A = \{y_1^A, \ldots, y_{N_A}^A\}$ and $Y_B = \{y_1^B, \ldots, y_{N_B}^B\}$, and we want to test the hypotheses

  H_0 : The A and B come from the same distribution

  H_1 : The A and B do not come from the same distribution

- Given a test statistic $T(Y_A, Y_B)$, such as $\overline{y_\cdot^A} - \overline{y_\cdot^B}$, we need the distribution of T under H_0.
- Under H_0, the joint sample $Y_{A\cup B} = \{y_1^A, \ldots, y_{N_A}^A, y_1^B, \ldots, y_{N_B}^B\}$ is a collection of exchangeable variables.
- Each random permutation of $Y_{A\cup B}$ has the same distribution as $Y_{A\cup B}$, under H_0 (but not under H_1).

Randomisation/permutation tests

Assume that large test statistics T indicate deviations from H_0.

Permutation tests

- For $j = 1, \ldots, J$, draw $Y_{A\cup B}^{(j)}$ as a random permutation of $Y_{A\cup B}$, and split the result into subsets $Y_A^{(j)}$ and $Y_B^{(j)}$ of size N_A and N_B, respectively.
- Compute the test statistics $T^{(j)} = T(Y_A^{(j)}, Y_B^{(j)})$.
- The average $\frac{1}{J}\sum_{j=1}^{J} \mathbb{I}\{T^{(j)} \geq T(Y_A, Y_B)\}$ is an unbiased estimator of the p-value w.r.t. T for the hypothesis H_0 : the elements of Y_A and Y_B are mutually exchangeable.
- If $N_A + N_B$ is small, the limit $J \to \infty$ can be obtained by using all possible permutations instead of independent random permutations.

- In other exchangeability situations, similar tests can be designed.

Randomisation/permutation test for proper scores

- In a collection of prediction scores for two models A and B, each data point has its own predictive distribution, so we don't have full exchangability.
- We do have *pairwise exchangeability* if the two model predictions are equivalent.
- To construct a formal test, we randomise the scores within each pair. If we investigate the difference between scores, $S_i^\Delta = S(F_i^A, y_i) - S(F_i^B, y_i)$, $i = 1, \ldots, N$, that means swapping the sign of the difference. For testing the average difference, we can use the test statistic

$$T(\{S_i^\Delta\}) = \frac{1}{N} \sum_{i=1}^{N} S_i^\Delta$$

- For $j = 1, \ldots, J$ and each $i = 1, \ldots, N$, draw $S_i^{\Delta(j)} = S_i^\Delta$ with probability 0.5, and $-S_i^\Delta$ with probability 0.5 (equivalent to swapping the A and B scores within each pair i).
- Compute the test statistics, $T^{(j)} = T(\{S_i^{\Delta(j)}, i = 1, \ldots, N\})$.
- The average $\frac{1}{J} \sum_{j=1}^{J} \mathbb{I}\{T^{(j)} \geq T(\{S_i^\Delta\})\}$ is an unbiased estimator of the one-sided p-value w.r.t. T for the hypothesis H_0 : the scores of the two models are pairwise exchangeable vs. H_1 : Model B is better than A (according to the score used for the test)

The Monte Carlo error of randomisation tests

- Randomisation tests have p-value estimators of the form $\widehat{p} = \frac{1}{J}\sum_{j=1}^{J} \mathbb{I}\{T^{(j)} \geq T(Y_A, Y_B)\}$ (which includes the pairwise exchangeability test).
- Each term is a $0/1$ indicator variable, drawn from a Binomial distribution $\mathsf{Bin}(1, p_0)$, where p_0 is the true p-value for the test.
- This means that $J\widehat{p} \sim \mathsf{Bin}(J, p_0)$, so that $\mathsf{E}(\widehat{p}) = p_0$, and $\mathsf{Var}(\widehat{p}) = \frac{p_0(1-p_0)}{J}$.
- The relative Monte Carlo error, defined as $\frac{\sqrt{\mathsf{E}[(\widehat{p}-p_0)^2]}}{p_0}$, is

$$\sqrt{\frac{1-p_0}{Jp_0}}$$

  which shows that for accurate (relative) estimation of the p-value, J needs to be larger for smaller p_0.
- Fortunately, the pairwise exchangeability test for score comparisons does not involve any re-estimation of model parameters, so implementation can be very fast even for large J.

Example (with suboptimal code!)

A simulation model with known expectation structure, but using estimated variance structure

```
set.seed(12345L)
N_train <- 100
N_test <- 30
phi <- 0.1
data_train <- tibble(
  x = runif(N_train),
  y = x + rnorm(N_train, sd = 2*phi*x)
)
data_test <- tibble(
  x = runif(N_test),
  y = x + rnorm(N_test, sd = 2 * phi * x)
)
# Simple parameter estimates
beta_est <- mean(data_train$y * data_train$x) / mean(data_train$x^2)
phi_est <- sd(data_train$y - data_train$x)
# Three predictions of the test data
pred <- rbind(
  data_test %>% mutate(method = "A", mu = x * beta_est, sigma = phi_est),
  data_test %>% mutate(method = "B", mu = x * beta_est, sigma = 2 * phi_est * x),
  data_test %>% mutate(method = "C", mu = x, sigma = 2 * phi * x)
)
```

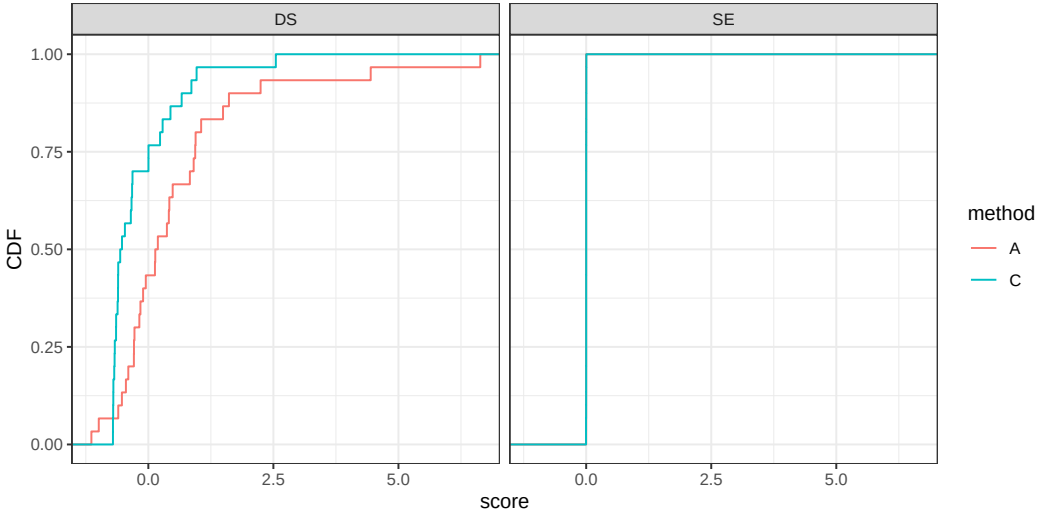
```r
# Prediction scores
pred <- rbind(
  pred %>% group_by(method) %>%
    mutate(score_type = "SE",
           score = score_se(pred = data.frame(mu, sigma), y = y)) %>%
    ungroup(),
  pred %>% group_by(method) %>%
    mutate(score_type = "DS",
           score = score_ds(pred = data.frame(mu, sigma), y = y)) %>%
    ungroup())
# Average/mean scores
pred %>% group_by(method, score_type) %>%
  summarise(score = mean(score)) %>%
  ungroup()
```

```
## # A tibble: 6 x 3
##   method score_type    score
##   <chr>  <chr>         <dbl>
## 1 A      DS          -3.43
## 2 A      SE           0.00949
## 3 B      DS          -4.03
## 4 B      SE           0.00949
## 5 C      DS          -4.23
## 6 C      SE           0.00978
```

```r
# Score differences relative to method B
pred_diff <- pred %>%
  filter(method != "B") %>%
  group_by(method) %>%
  mutate(score = score - (pred %>% filter(method == "B"))[["score"]]) %>%
  ungroup()
# Average/mean score differences relative to method B
pred_diff %>%
  group_by(method, score_type) %>%
  summarise(score = mean(score)) %>%
  ungroup()
```

```
## # A tibble: 4 x 3
##   method score_type     score
##   <chr>  <chr>          <dbl>
## 1 A      DS          0.593
## 2 A      SE          0
## 3 C      DS         -0.204
## 4 C      SE          0.000292
```

Empirical CDFs for the pairwise score differences



Empirical CDFs for the pairwise score differences under exchangeability