Assessing and Accounting for the Dependence of Age in Xe-MRI Using GAMLSS

Joseph Plummer

5/31/2022

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents.

When you click the **Knit** button in **R-studio**, a document will be generated that includes both content as well as the output of any embedded R code chunks within the document.

Goals of this demo

1. Share GAMLSS code examples via this interactive notebook.
2. Assess the dependence of age.
3. Account for age dependence in gas-exchange binned analysis.

Load libraries

You can install a library if necessary by using `install.packages()`.

```r
library(gamlss) # For GAMLSS
library(ggplot2) # For plotting
library(mgcv) # For GAMs
library(visreg) # For plotting GAMs
```

A brief introduction

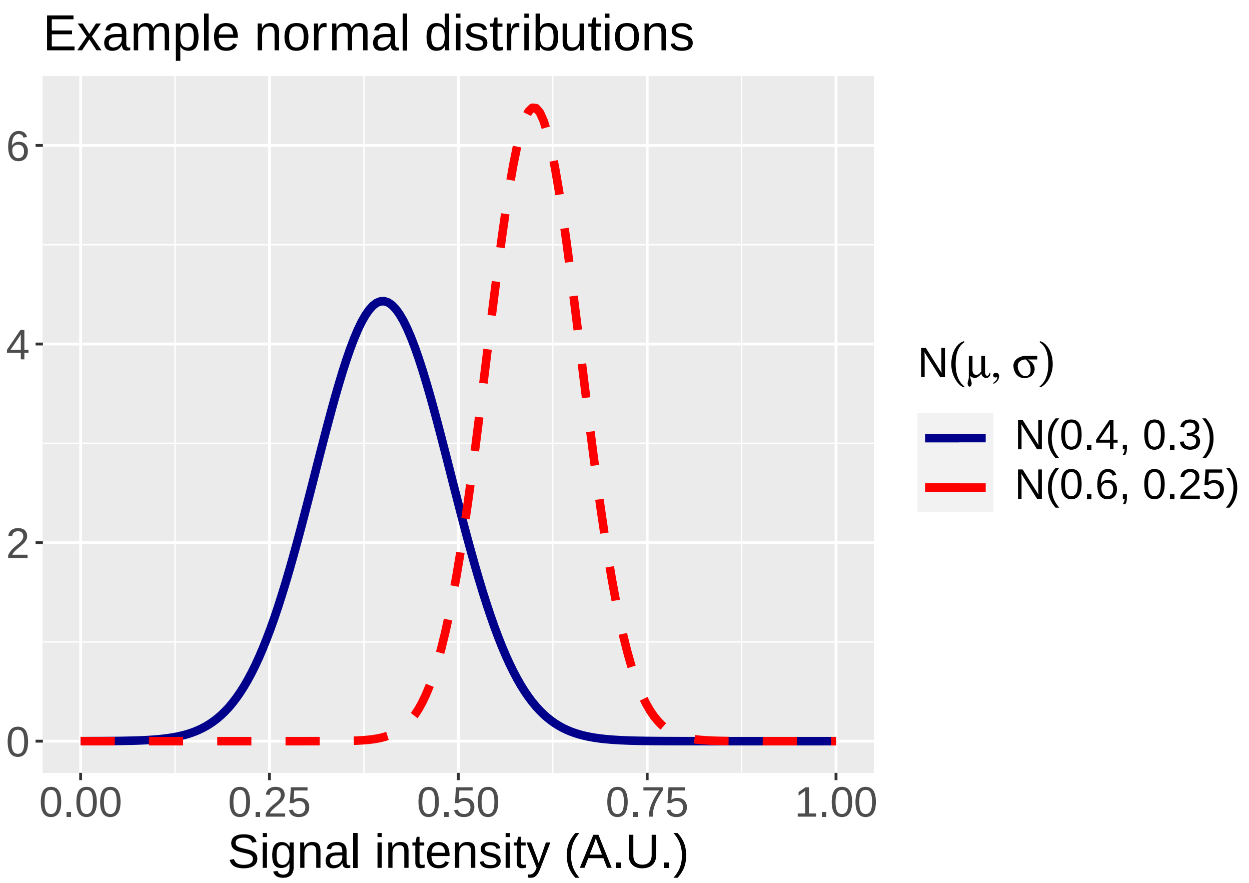
The Xe-MRI community uses several methods to perform image analysis. The most trivial method is to simply calculate the median signal intensity, μ, of the voxels of a disease subject. However, this method only provides a global overview of a subject's voxels, and is likely to be insensitive to small changes caused by disease (which is what we are fundamentally interested in).

Thus, many studies implement a linear binning (LB) technique to make statistical comparisons {Z. Wang, *et. al.* 2017}. The LB method assumes that, for an aggregated healthy-reference cohort, the voxel-wise signal intensities of a gas-exchange metric, $\mathbf{y}$, fits a normal distribution, described by:

$$\mathbf{y} \sim N(\mu, \sigma) \tag{1}$$

where σ is the variation. Note, bold typeface represents a vector (lowercase) or matrix (uppercase), and non-bold typeface represents scalar values. Subsequently, statistical inferences are made by calculating the fractions of the lung that fall into each z-score (percentile) bin (-2, -1, 0, +1, +2) of the healthy normal distribution.

```
# Choose a range of signal intensities to simulate distribution over:
simulated_locations <- seq(0, 1, length.out = 200)

# Simulate a normal distribution:
simulated_distributions <- data.frame(indx = simulated_locations)
mu_base <- 0.4
sigma_base <- 0.3
simulated_distributions$normal_base <- dNO(
  x = simulated_locations,
  mu = mu_base,
  sigma = (sigma_base)^2
)

# Simulate another normal distribution:
mu_perturbed <- 0.6
sigma_perturbed <- 0.25
simulated_distributions$normal_perturbed <- dNO(
  x = simulated_locations,
  mu = mu_perturbed,
  sigma = (sigma_perturbed)^2
)

# Legend:
leg = c(substitute(paste("N(",mb, ", ", sb,")"), list(mb=mu_base[1], sb=sigma_base[1])),
        substitute(paste("N(",mp, ", ", sp,")"), list(mp=mu_perturbed[1], sp=sigma_perturbed[1])))

# Plot the distributions:
ggplot() +
  geom_path(data = simulated_distributions,
            aes(x = indx,
                y = normal_base,
                color = "Normal"),
            linetype = "solid",
            size = 1.5,
            na.rm = TRUE) +
  geom_path(data = simulated_distributions,
            aes(x = indx,
                y = normal_perturbed,
                color = "Perturbed Normal"),
            linetype = "dashed",
            size = 1.5,
            na.rm = TRUE) +
  scale_color_manual(name = expression(N(mu,sigma)),
                     values = c("Normal" = "darkblue", "Perturbed Normal" = "red"),
                     labels = c(leg)) +
  ggtitle("Example normal distributions") +
  theme(legend.position = "right") +
  xlab('Signal intensity (A.U.)') +
  theme(axis.title.x = element_text(color = "black", size = 18),
        axis.title.y = element_blank(),
        legend.text = element_text(size = 16),
        legend.key.width = unit(1, 'cm'),
        title = element_text(size = 16),
        axis.text = element_text(size = 16),
        plot.margin = unit(c(0, 0, 0, 0), "mm"))
```

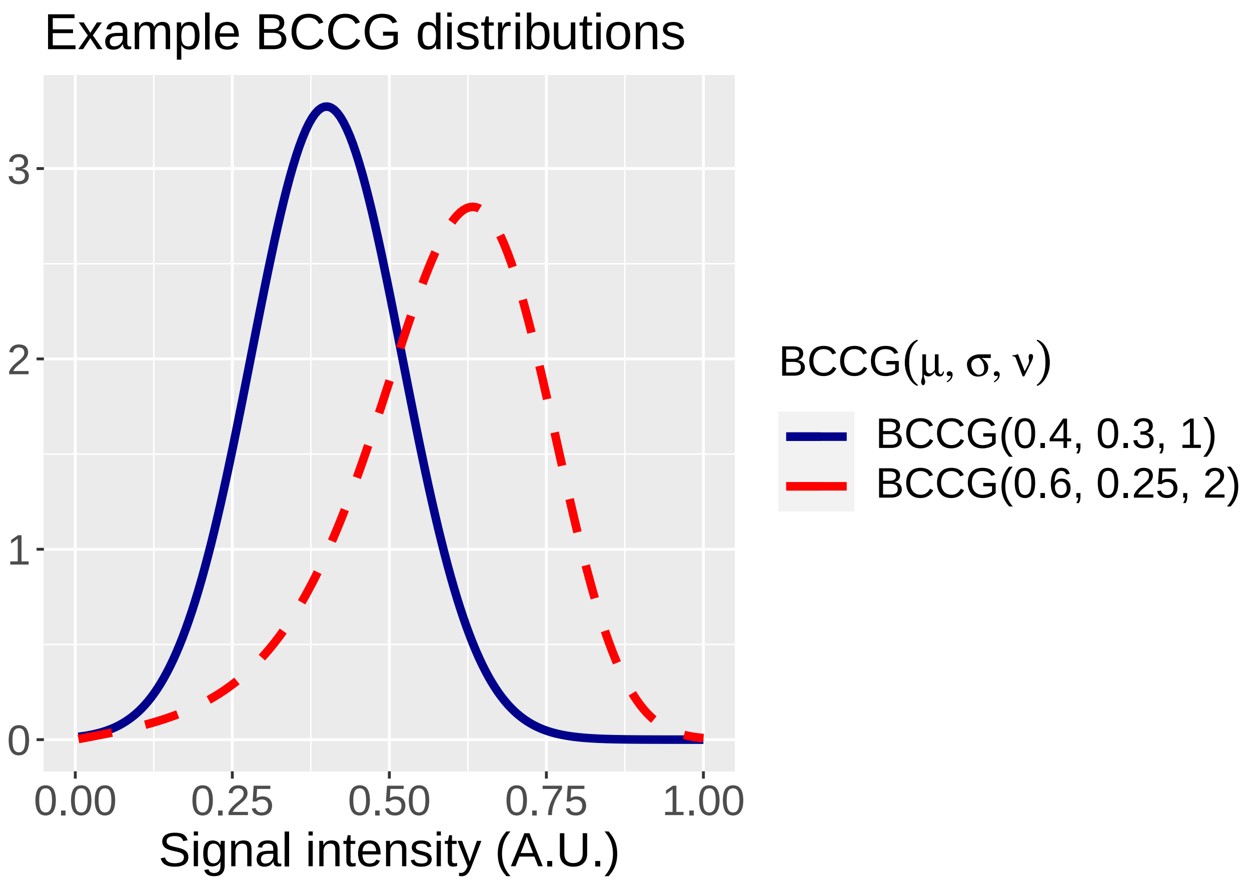


A generalized linear-binning (GLB) method was later proposed to account for skew, by using a Box-Cox transformation to transform a skewed distribution into a normal distribution {M. He, *et. al.* 2017}. Once transformed, the same z-score approach can then be implemented and then transformed back into its original distribution. An equivalent way to represent this transformation is to fit y to a Box-Cox Cole-Green distribution:

$$\mathbf{y} \sim BCCG(\mu, \sigma, \nu) \tag{2}$$

where ν is the distribution skew, equal to the Box-Cox transform power used in the transformation.

```r
# Choose a range of signal intensities to simulate distribution over:
simulated_locations <- seq(0, 1, length.out = 200)

# Simulate a BCCG distribution:
mu_base <- 0.4
sigma_base <- 0.3
nu_base <- 1
simulated_distributions$BCCG_base <- dBCCG(
  x = simulated_locations,
  mu = mu_base,
  sigma = sigma_base,
  nu = nu_base
)

# Simulate another BCCG distribution:
mu_perturbed <- 0.6
sigma_perturbed <- 0.25
nu_perturbed <- 2
simulated_distributions$BCCG_perturbed <- dBCCG(
  x = simulated_locations,
  mu = mu_perturbed,
  sigma = sigma_perturbed,
  nu = nu_perturbed
)

# Legend:
leg = c(substitute(paste("BCCG(",mb, ", ", sb,", ", nb,")"), list(mb=mu_base[1], sb=sigma_base[1], nb=nu
        substitute(paste("BCCG(",mp, ", ", sp,", ", np,")"), list(mp=mu_perturbed[1], sp=sigma_perturbed

# Plot the distributions:
ggplot() +
  geom_path(data = simulated_distributions,
            aes(x = indx,
                y = BCCG_base,
                color = "BCCG"),
            linetype = "solid",
            size = 1.5,
            na.rm = TRUE) +
  geom_path(data = simulated_distributions,
            aes(x = indx,
                y = BCCG_perturbed,
                color = "Perturbed BCCG"),
            linetype = "dashed",
            size = 1.5,
            na.rm = TRUE) +
  scale_color_manual(name = expression(BCCG(mu,sigma,nu)),
                     values = c("BCCG" = "darkblue", "Perturbed BCCG" = "red"),
                     labels = c(leg)) +
  ggtitle("Example BCCG distributions") +
  theme(legend.position = "right") +
  xlab('Signal intensity (A.U.)') +
  theme(axis.title.x = element_text(color = "black", size = 18),
        axis.title.y = element_blank(),
        legend.text = element_text(size = 16),
        legend.key.width = unit(1, 'cm'),
        title = element_text(size = 16),
        axis.text = element_text(size = 16),
        plot.margin = unit(c(0, 0, 0, 0), "mm"))
```



To date, the above methods have assumed constant distribution parameters (median, variance, and skew), and thus a constant healthy-reference distribution. However, pulmonary structure and function are known to vary substantially with demographic variables, like *age*.

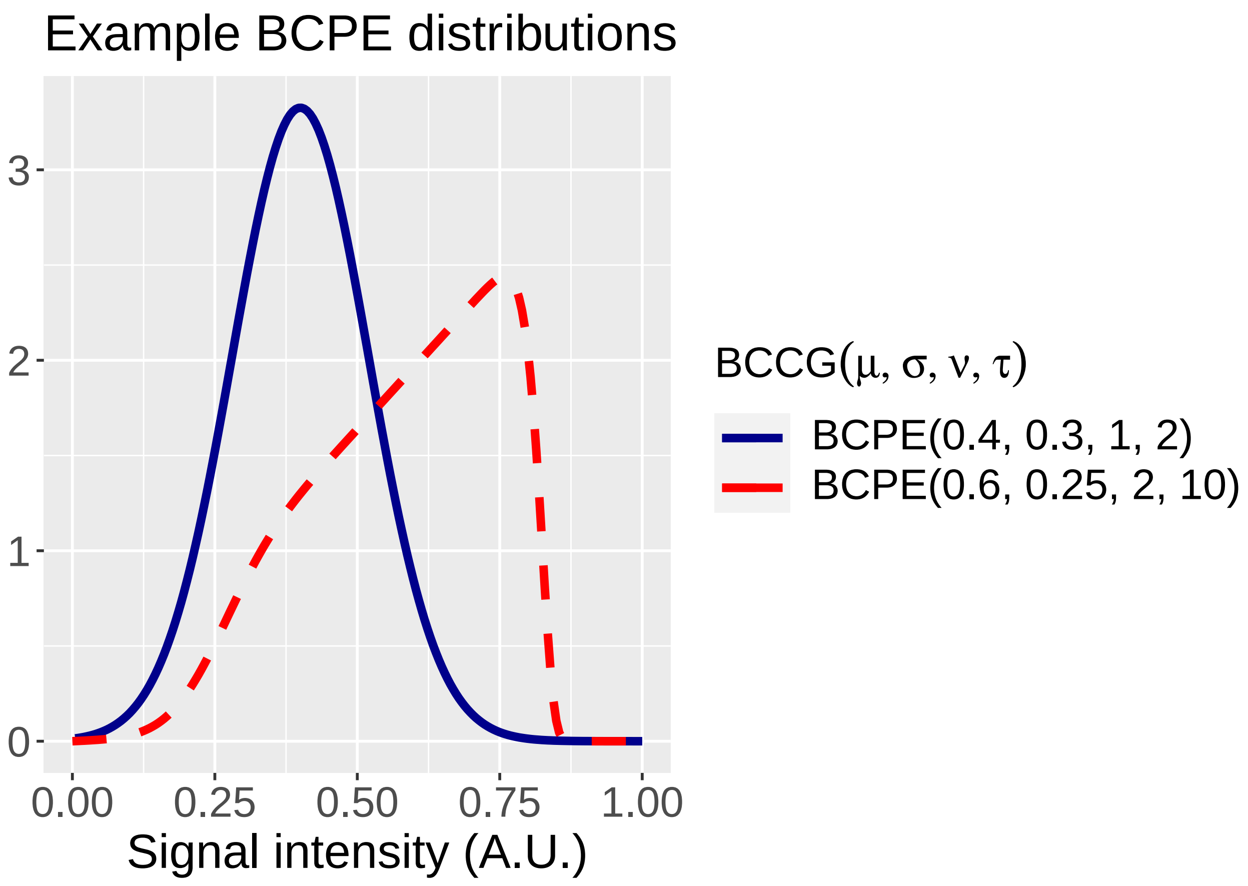
Our proposal

To expand prior methods, we propose a method that fits **y** to a Box-Cox power-exponential distribution, that is dependent on all possible distribution parameters (median, variance, skew, and kurtosis). Furthermore, we use GAMLSS methods to make our fitted distribution flexible to some explanatory variable, **x**, which in our case is age. This can be described by:

$$\mathbf{y} \sim BCPE(\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\nu}, \boldsymbol{\tau}) \tag{3}$$

The final distribution parameter, τ, represents kurtosis. The bold typeface is used to show the non-scalar (vector) type of the parameter. Prior methods asssumed a constant distribution, so a scalar distribution parameter set could be used. The proposed method fits a flexible distribution (depends on age) to **y**, therefore the distribution parameters inside the model cannot be represented by scalars. The mathematics behind this is discussed in some detail in our paper, with a more rigorous discussion in the GAMLSS book: A flexible regression approach using GAMLSS in R.

```r
# Choose a range of signal intensities to simulate distribution over:
simulated_locations <- seq(0, 1, length.out = 200)

# Simulate a BCPE distribution:
simulated_distributions <- data.frame(indx = simulated_locations)
mu_base <- 0.4
sigma_base <- 0.3
nu_base <- 1
tau_base <- 2
simulated_distributions$BCPE_base <- dBCPE(
  x = simulated_locations,
  mu = mu_base,
  sigma = sigma_base,
  nu = nu_base,
  tau = tau_base
)

# Simulate another BCPE distribution:
mu_perturbed <- 0.6
sigma_perturbed <- 0.25
nu_perturbed <- 2
tau_perturbed <- 10
simulated_distributions$BCPE_perturbed <- dBCPE(
  x = simulated_locations,
  mu = mu_perturbed,
  sigma = sigma_perturbed,
  nu = nu_perturbed,
  tau = tau_perturbed
)

# Legend:
leg = c(substitute(paste("BCPE(",mb, ", ", sb,", ", nb,", ", tb,")"),
                   list(mb=mu_base[1], sb=sigma_base[1], nb=nu_base[1], tb=tau_base[1])),
        substitute(paste("BCPE(",mp, ", ", sp,", ", np,", ", tp,")"),
                   list(mp=mu_perturbed[1], sp=sigma_perturbed[1],np=nu_perturbed[1],tp=tau_perturbed[1

# Plot the distributions:
ggplot() +
  geom_path(data = simulated_distributions,
            aes(x = indx,
                y = BCPE_base,
                color = "BCPE"),
            linetype = "solid",
            size = 1.5,
            na.rm = TRUE) +
  geom_path(data = simulated_distributions,
            aes(x = indx,
                y = BCPE_perturbed,
                color = "Perturbed BCPE"),
            linetype = "dashed",
            size = 1.5,
            na.rm = TRUE) +
    scale_color_manual(name = expression(BCCG(mu,sigma,nu,tau)),
                      values = c("BCPE" = "darkblue", "Perturbed BCPE" = "red"),
                      labels = c(leg)) +
  ggtitle("Example BCPE distributions") +
  theme(legend.position = "right") +
  xlab('Signal intensity (A.U.)') +
  theme(axis.title.x = element_text(color = "black", size = 18),
        axis.title.y = element_blank(),
        legend.text = element_text(size = 16),
        legend.key.width = unit(1, 'cm'),
        title = element_text(size = 16),
        axis.text = element_text(size = 16),
        plot.margin = unit(c(0, 0, 0, 0), "mm"))
```



Assess age dependence in gas-exchange MRI

Import data

We will be using randomly sampled RBC:membrane (ratio) data from a range of subjects at Cincinnati Children's Hospital. All personal information has been removed, and only 500 voxels were sampled per subject.

```
# Import data:
load("data/RBC_Membrane_Voxels.rdata")
```

Split the dataframe into respective ages, and study variables.

```
y = Voxels
y_subj = split(Voxels, Voxels$Age)
x = unique(y$Age) # A 'cheat' way of selecting the unique ages of each subject.
num_subj = length(x)
num_voxels = dim(Voxels)[1]/num_subj
```

Use GAMLSS to fit distributions

Fit each subject's voxels, y_{subj} to a BCPE distribution. We only want a single distribution per subject, so we are setting the link functions for each distribution per subject to `.formula = ~ 1`. This means a single distribution parameter will be extracted for all **x**.

```
# Declare distribution family:
fit_fam = BCPE(
  mu.link = "identity",
  sigma.link = "identity",
  nu.link = "identity",
  tau.link = "identity"
)

# Initialize our starting parameter values:
mu = 0
sigma = 0
nu = 0
tau = 0
Age = 0

# Iterate a GAMLSS model (BCPE) for each subject:
for (ii in 1:num_subj) {
  mdl_per_subj <- gamlss(
    Intensity ~ 1,
    sigma.formula = ~ 1,
    nu.formula = ~ 1,
    tau.formula = ~ 1,
    data = y_subj[[ii]],
    family = fit_fam
  )

  # Extract the fitting parameters we are interested in:
```

```r
  mu[ii] = mdl_per_subj$mu.fv[1] # all values are the same, so we only need first one
  sigma[ii] = mdl_per_subj$sigma.fv[1]
  nu[ii] = mdl_per_subj$nu.fv[1]
  tau[ii] = mdl_per_subj$tau.fv[1]
  Age[ii] = x[ii]
}

# Store results in final data frame:
fitted_params <- data.frame(Age, mu, sigma, nu, tau)
```

View the fitted distribution parameters.

```r
head(fitted_params, 10)
```

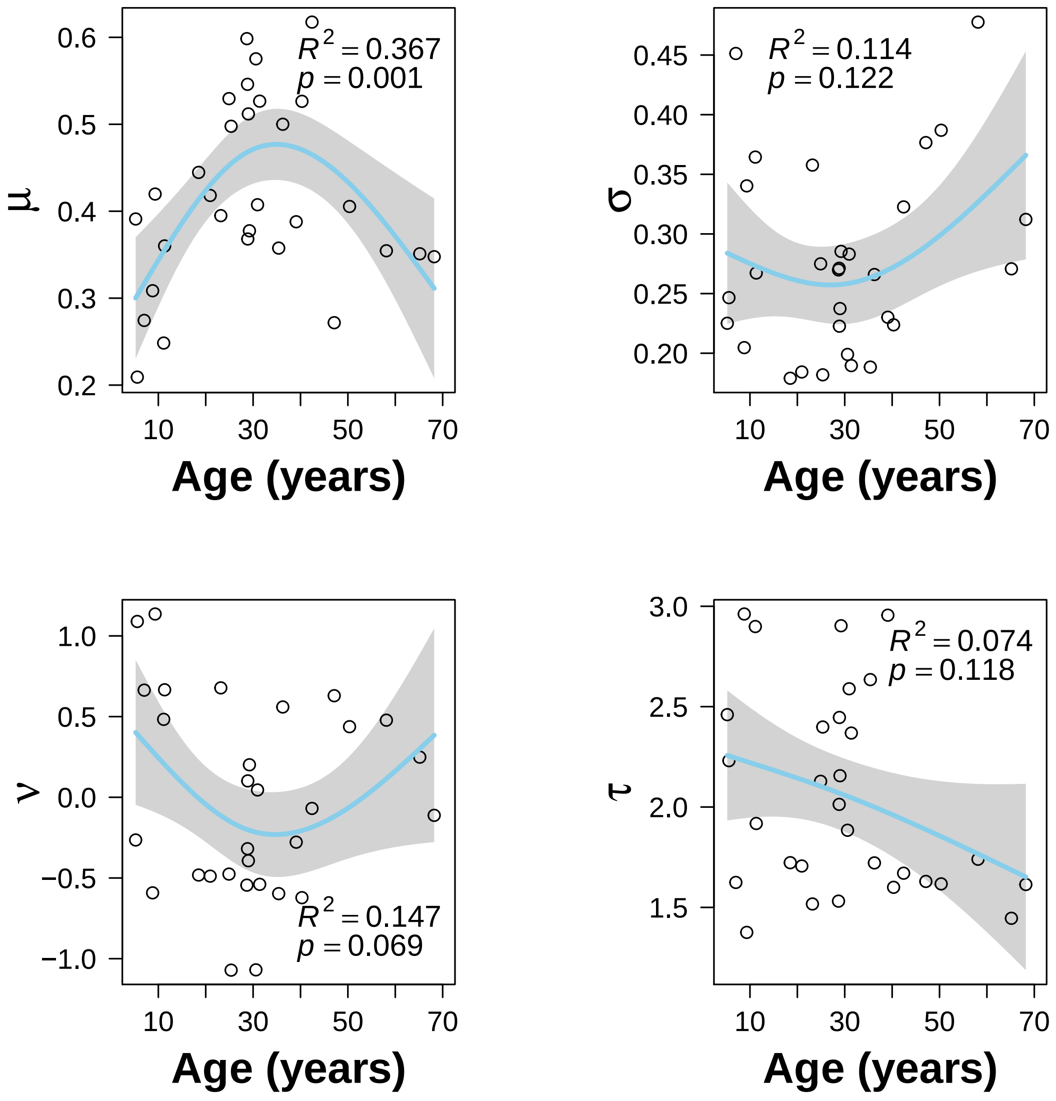
```
##          Age        mu     sigma         nu      tau
## 1   5.213889 0.3910771 0.2251267 -0.2646020 2.460078
## 2   5.561111 0.2092115 0.2465894  1.0895199 2.231012
## 3   7.020000 0.2744127 0.4513040  0.6638549 1.624401
## 4   8.794444 0.3085493 0.2046852 -0.5920511 2.961385
## 5   9.319444 0.4197593 0.3403064  1.1356584 1.375501
## 6  11.130556 0.2483926 0.3644364  0.4829486 2.898980
## 7  11.325000 0.3600696 0.2672548  0.6661701 1.917998
## 8  18.502778 0.4446260 0.1790021 -0.4817928 1.722743
## 9  20.938889 0.4181363 0.1842775 -0.4885926 1.706180
## 10 23.183333 0.3949078 0.3577311  0.6780160 1.517090
```

Model distribution parameters against age

Now that we have a fitted distribution parameter for each subject, we can assess the parameters against age and see if there is any sort of correlation. There are many ways to do this, but one of the most robust methods is to use generalized additive models (GAM) from the mgcv library. We can fit each parameter to a GAM against age, and then use the R^2 values on the converged fits to assess whether any dependence on age exists. GAMs are powerful in this aspect because they can converge on both linear and non-linear (smooth) fits.

```r
# Fit generalized additive models:
mod_gam_mu <- gam(mu ~ s(Age, k = 1, bs = "cr"), data = fitted_params)
mod_gam_sigma <- gam(sigma ~ s(Age, k = 1, bs = "cr"), data = fitted_params)
mod_gam_nu <- gam(nu ~ s(Age, k = 1, bs = "cr"), data = fitted_params)
mod_gam_tau <- gam(tau ~ s(Age, k = 1, bs = "cr"), data = fitted_params)

# Plot model fits:
par(mar=c(2, 4, 1, 1) + 2.5, mfrow=c(2,2),
    oma = c(0, 0, 2.5, 2.5))
visreg(mod_gam_mu, "Age",
       ylab = expression(mu),
       xlab = "Age (years)",
       cex.lab = 2,
       cex.axis = 1.3,
       font.lab = 2, # bold
       xlim = c(5,70),
       points = list(col = "black", pch = 21, cex = 1.2),
       line = list(col = "skyblue"),
       fill = list(col = "lightgray"),
       alpha = 0.05)
rp = vector('expression',2)
rp[1] = substitute(expression(italic(R)^2 == MYVALUE),
                   list(MYVALUE = round(summary(mod_gam_mu)$r.sq,3)))[2]
if (round(summary(mod_gam_mu)$s.table[,4],3) >= 0.001) {
  rp[2] = substitute(expression(italic(p) == MYOTHERVALUE),
                     list(MYOTHERVALUE = round(summary(mod_gam_mu)$s.table[,4],3)))[2]
} else {
  rp[2] = (expression(italic(p) < 0.001))
}
legend("topright",
       legend = rp,
       bty = "n",
       pch = NA,
       cex = 1.4,
       text.font = 2)

visreg(mod_gam_sigma, "Age",
       ylab = expression(sigma),
       xlab = "Age (years)",
       cex.lab = 2,
       cex.axis = 1.3,
       font.lab = 2, # bold
       xlim = c(5,70),
       points = list(col = "black", pch = 21, cex = 1.2),
       line = list(col = "skyblue"),
       fill = list(col = "lightgray"),
       alpha = 0.05)
rp = vector('expression',2)
rp[1] = substitute(expression(italic(R)^2 == MYVALUE),
                   list(MYVALUE = round(summary(mod_gam_sigma)$r.sq,3)))[2]
if (round(summary(mod_gam_sigma)$s.table[,4],3) >= 0.001)
{
  rp[2] = substitute(expression(italic(p) == MYOTHERVALUE),
                     list(MYOTHERVALUE = round(summary(mod_gam_sigma)$s.table[,4],3)))[2]
} else {
  rp[2] = (expression(italic(p) < 0.001))
}
legend("topleft",
       legend = rp,
       bty = "n",
       pch = NA,
       cex = 1.4,
       text.font = 2)

visreg(mod_gam_nu, "Age",
       ylab = expression(nu),
       xlab = "Age (years)",
       cex.lab = 2,
       cex.axis = 1.3,
       font.lab = 2, # bold
       xlim = c(5,70),
       points = list(col = "black", pch = 21, cex = 1.2),
       line = list(col = "skyblue"),
       fill = list(col = "lightgray"),
       alpha = 0.05)
rp = vector('expression',2)
rp[1] = substitute(expression(italic(R)^2 == MYVALUE),
                   list(MYVALUE = round(summary(mod_gam_nu)$r.sq,3)))[2]
if (round(summary(mod_gam_nu)$s.table[,4],3) >= 0.001)
{
  rp[2] = substitute(expression(italic(p) == MYOTHERVALUE),
                     list(MYOTHERVALUE = round(summary(mod_gam_nu)$s.table[,4],3)))[2]
} else {
  rp[2] = (expression(italic(p) < 0.001))
}
legend("bottomright",
       legend = rp,
       bty = "n",
       pch = NA,
       cex = 1.4,
       text.font = 2)

visreg(mod_gam_tau, "Age",
       ylab = expression(tau),
       xlab = "Age (years)",
       cex.lab = 2,
       cex.axis = 1.3,
       font.lab = 2, # bold
       xlim = c(5,70),
       points = list(col = "black", pch = 21, cex = 1.2),
       line = list(col = "skyblue"),
       fill = list(col = "lightgray"),
       alpha = 0.05)
rp = vector('expression',2)
rp[1] = substitute(expression(italic(R)^2 == MYVALUE),
                   list(MYVALUE = round(summary(mod_gam_tau)$r.sq,3)))[2]
if (round(summary(mod_gam_tau)$s.table[,4],3) >= 0.001)
{
  rp[2] = substitute(expression(italic(p) == MYOTHERVALUE),
                     list(MYOTHERVALUE = round(summary(mod_gam_tau)$s.table[,4],3)))[2]
} else {
  rp[2] = (expression(italic(p) < 0.001))
}
legend("topright",
       legend = rp,
       bty = "n",
       pch = NA,
       cex = 1.4,
       text.font = 2)

mtext(paste("Generalized Additive Models for RBC:membrane"), outer = TRUE, cex = 2)
```

Generalized Additive Models for RBC:membrane



Interpretation

As described in the paper, the R^2 values can be used to determine the effect size of age on a given distribution parameter where effect sizes of 4%, 25%, and 64% are termed small, medium, and large effects, respectively. (This is found by multiplying R^2 by 100). The p-value is not a measure of effect size; rather, it is a measure of the likelihood that the fit is correct for the given data. As recommended by the initial GAM papers, a formal cut-off of p<0.05 should not be used to define significance.

For a quick interpretation, one can see that age has: medium effect size on the μ distribution parameter (median), and small effect size on the σ, ν, and τ parameters (coefficient of variation, skew, and kurtosis). This information is useful, because we can now use it to build our full GAMLSS model. For example, if we wanted to reduce the possible degrees of freedom in the model, we could say that the effect size of age on τ is very small - therefore we will restrict this parameter link function in the model to having no dependence on age. However, this is up to the user depending on how many data points they have, and how many explanatory variables.

Account for age in Xe gas-exchange MRI analysis

Now that we understand that age is a significant variable in our distribution parameters, we can now incorporate it into the full flexible distribution model, across all subjects. We incorporate it in a very similar way to the prior section, except now we set our response variable to **y**, a vector of all the voxel intensities.

Before we run the model, we should account for some extra steps. First, we can apply a weighting to each of the data points, for each subject. The weighting should account for the number of data points at a given age. This is important if the number of voxels per subject changes, as you don't want to weight your model towards a largely sampled subject. However, in our case where we only have 500 voxels per subject, this won't actually be important.

Next, we can consider the type of link function (essentially a function that describes the modeled relationship between age and a distribution parameter) to use in our model. We chose to use a `pb-spline` link function - a penalized b-spline function that can be easily penalized to converge on smooth solutions with few degrees of freedom. One could also set the link functions to converge on linear or logarithmic models, but we reasoned that a pb-spline function would be most robust for whatever data we throw into the model.

Finally, we must consider the penalty we apply to the link function. Penalties in GAMLSS work the same way as they do for cubic-spline regressions in more commonly used GAMs, for which there are many useful youtube videos to help explain them. The default penalty value for a pb-spline that is penalized by `GAIc` (Generlazed Akaike Criterion) methods is 2. We opted for a larger penalty of to ensure we do not overfit, which could be highly possible given our relatively few number of subjects (although we counter this by using 500 voxels per subject). We chose a penalty value of 3.84, corresponding to the chi-square value for a single degree of freedom with a P-value of 0.05.

```
# Declare distribution family:
fit_fam = BCPE(
  mu.link = "identity",
  sigma.link = "identity",
  nu.link = "identity",
  tau.link = "identity"
)

# Declare weighting for each subject:
weighting = rep(1 / num_voxels, nrow(Voxels))

# Declare a penalty, k: current agreed value is 3.84, corresponding to chi-square test. Default is 2.
chi = 3.84

# Run GAMLSS model:
voxel_mdl <- gamlss(Intensity ~ pb(Age, method = "GAIC", k = chi),
                    # Use a pb-spline link function if age has at least a small effect size on distribu
                    if (summary(mod_gam_mu)$r.sq < 0.04) {
                      mu.formula = ~ 1} else {
                        mu.formula = ~ pb(Age, method = "GAIC", k = chi)},
                    if (summary(mod_gam_sigma)$r.sq < 0.04) {
                      sigma.formula = ~ 1} else {
                        sigma.formula = ~ pb(Age, method = "GAIC", k = chi)},
                    if (summary(mod_gam_nu)$r.sq < 0.04) {
                      nu.formula = ~ 1} else {
                        nu.formula = ~ pb(Age, method = "GAIC", k = chi)},
                    if (summary(mod_gam_tau)$r.sq < 0.04) {
                      tau.formula = ~ 1} else {
                        tau.formula = ~ pb(Age, method = "GAIC", k = chi)},
                    data = y,
                    family = fit_fam,
                    weights = weighting,
                    method = mixed(20,100)) # Convergence method for the GAMLSS model
```

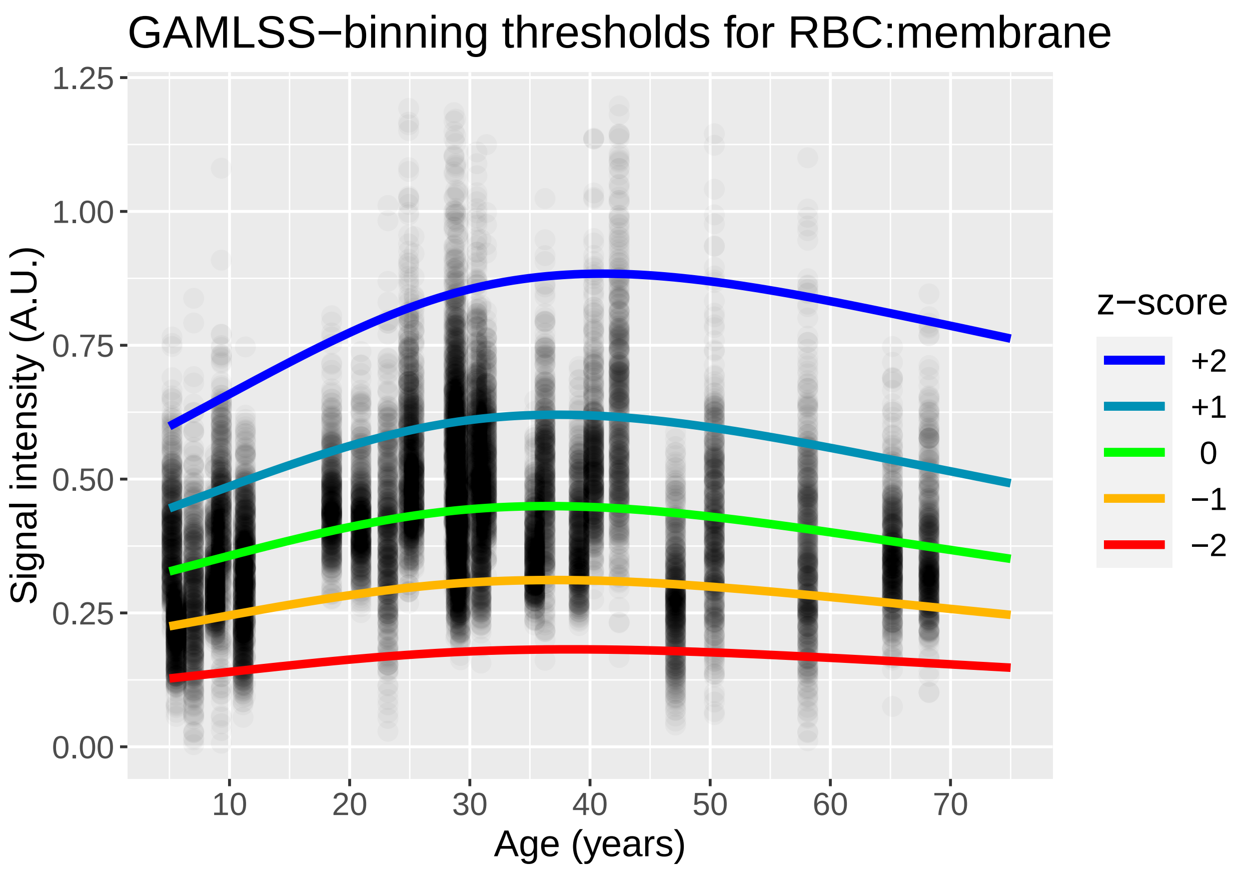
Predict the age dependent thresholds

Now that we have run the model (beware, this can take a while for large dimensions of **y**), we can use the model to predict the age dependent z-score bins - just like with prior linear binning and generalized linear binning methods.

```
# Simulate an age range:
sim_age <- data.frame(Age = seq(5, 75, 0.25))

# Use the centiles.pred function to predict percentiles (can be correlated to a distribution z-score).
thresholds <- centiles.pred(
  voxel_mdl,
  xname = "Age",
  xvalues = sim_age$Age,
  cent = c(2.275013, 15.865525, 50, 84.134475, 97.724987)
)

# Store the predictions:
Pred <- data.frame(Age = sim_age$Age)
Pred$Age <- sim_age$Age
Pred$Thresh1 <- thresholds$`2.275013`
Pred$Thresh2 <- thresholds$`15.865525`
Pred$Thresh3 <- thresholds$`50`
Pred$Thresh4 <- thresholds$`84.134475`
Pred$Thresh5 <- thresholds$`97.724987`

# View the top 10 rows:
head(Pred, 10)
```

Plot the thresholds using ggplot commands:

```
par(mfrow = c(1,1))

ggplot(data = y,
       aes(x = Age, y = Intensity)) +
  geom_point(alpha = 0.03,
             size = 3,
             color = "black") +
  geom_path(data = Pred,
            aes(x = Age, y = Thresh1, color = "1"),
            size = 1.5) +
  geom_path(data = Pred,
            aes(x = Age, y = Thresh2, color = "2"),
            size = 1.5) +
  geom_path(data = Pred,
            aes(x = Age, y = Thresh3, color = "3"),
            size = 1.5) +
  geom_path(data = Pred,
            aes(x = Age, y = Thresh4, color = "4"),
            size = 1.5) +
  geom_path(data = Pred,
            aes(x = Age, y = Thresh5, color = "5"),
            size = 1.5) +
  scale_colour_manual(values = c(rgb(1,0,0),
                                 rgb(1, 0.7143, 0),
                                 rgb(0, 1, 0),
                                 rgb(0, 0.57, 0.71),
                                 rgb(0, 0, 1)),
                      labels = c("-2",
                                 "-1",
                                 " 0",
                                 "+1",
                                 "+2")) +
  guides(color = guide_legend(reverse = TRUE)) +
  scale_x_continuous(limits = c(min(sim_age), max(sim_age)),
                     breaks = seq(10, 70,10)) +
  theme(legend.text = element_text(size = 12),
        legend.key.width = unit(1, 'cm'),
        title = element_text(size = 14),
        axis.text = element_text(size = 12),
        plot.margin = unit(c(1, 1, 1, 1), "mm")) +
  labs(x = "Age (years)",
       y = "Signal intensity (A.U.)",
       color = "z-score") +
  ylim(c(0, 1.2)) +
  ggtitle("GAMLSS-binning thresholds for RBC:membrane")
```



Interpretation

The GAMLSS model has converged to a solution that shows a clear age dependence in the z-score thresholds. The median location of the flexible distribution appears to change the most, as expected from the prior distribution parameter GAMs. The bin widths also change with age, with uneven widths between z-scores. This is reflective of variation, skew, and kurtosis parameters being affected by age in the model. By not accounting for these age dependent changes in distribution, overall sensitivity to disease will be lost.

Summary

In this markdown document, we have shown the basic elements of implementing GAMLSS into Xe gas-exchange MRI analysis. The code from this notebook can be copied and implemented into your own studies, all using open-source software in R-studio. It is possible to make an R executable for MATLAB or Python, so your analysis package can be easily implemented to your current studies.

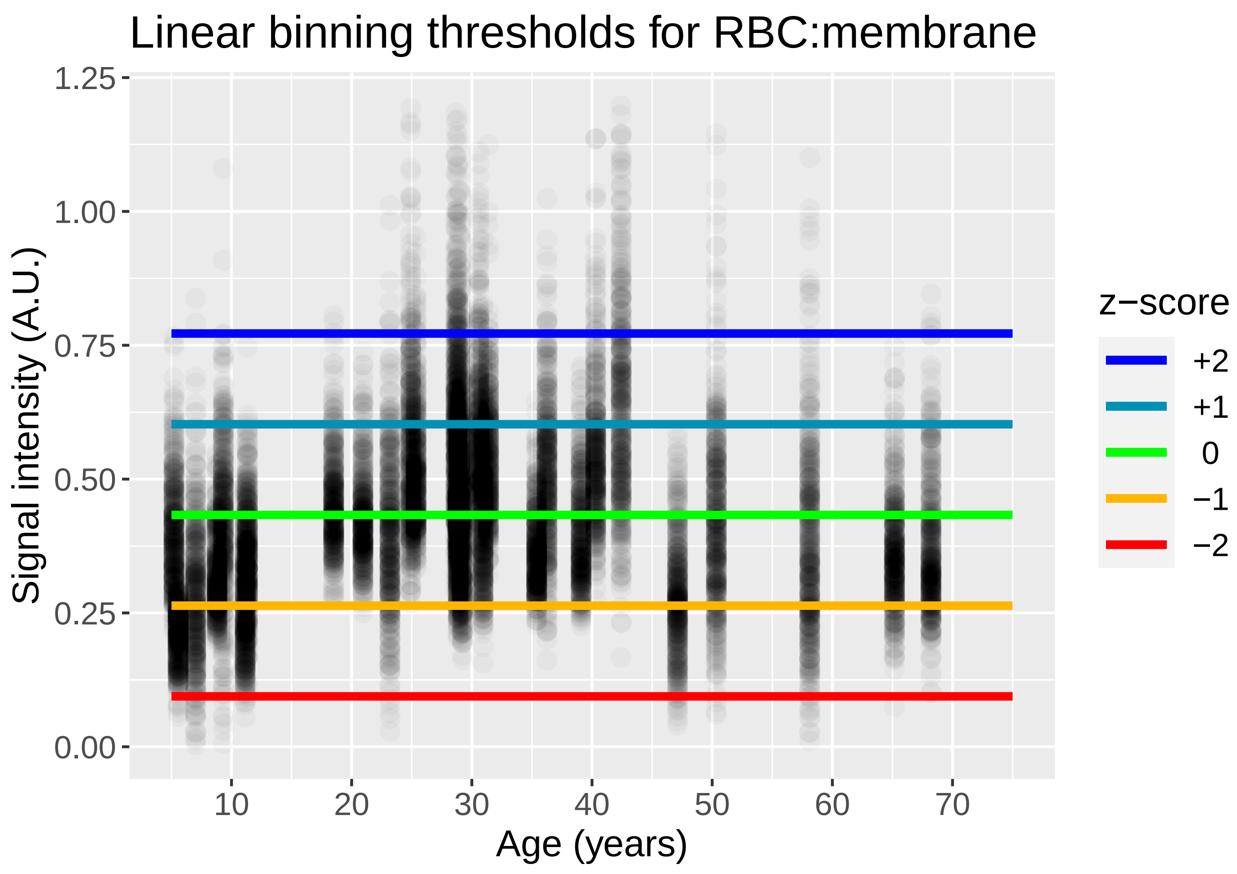
Of course, we have only shown the most basic elements of GAMLSS functionality. One of the most exciting features is that you can add more explanatory variables, such as sex. This is possible by simply changing your model arguments to `Intensity ~ pb(Age + Sex, method = "GAIC", k = chi)`, and then ensuring your data is set up with the right type of variable (`factor` for sex), and ensuring you have an adequate amount of data to avoid over-fitting.

Appendix

You can also implement current thresholding methods, like linear binning and generalized linear binning. This is achieved as below:

Linear binning thresholds

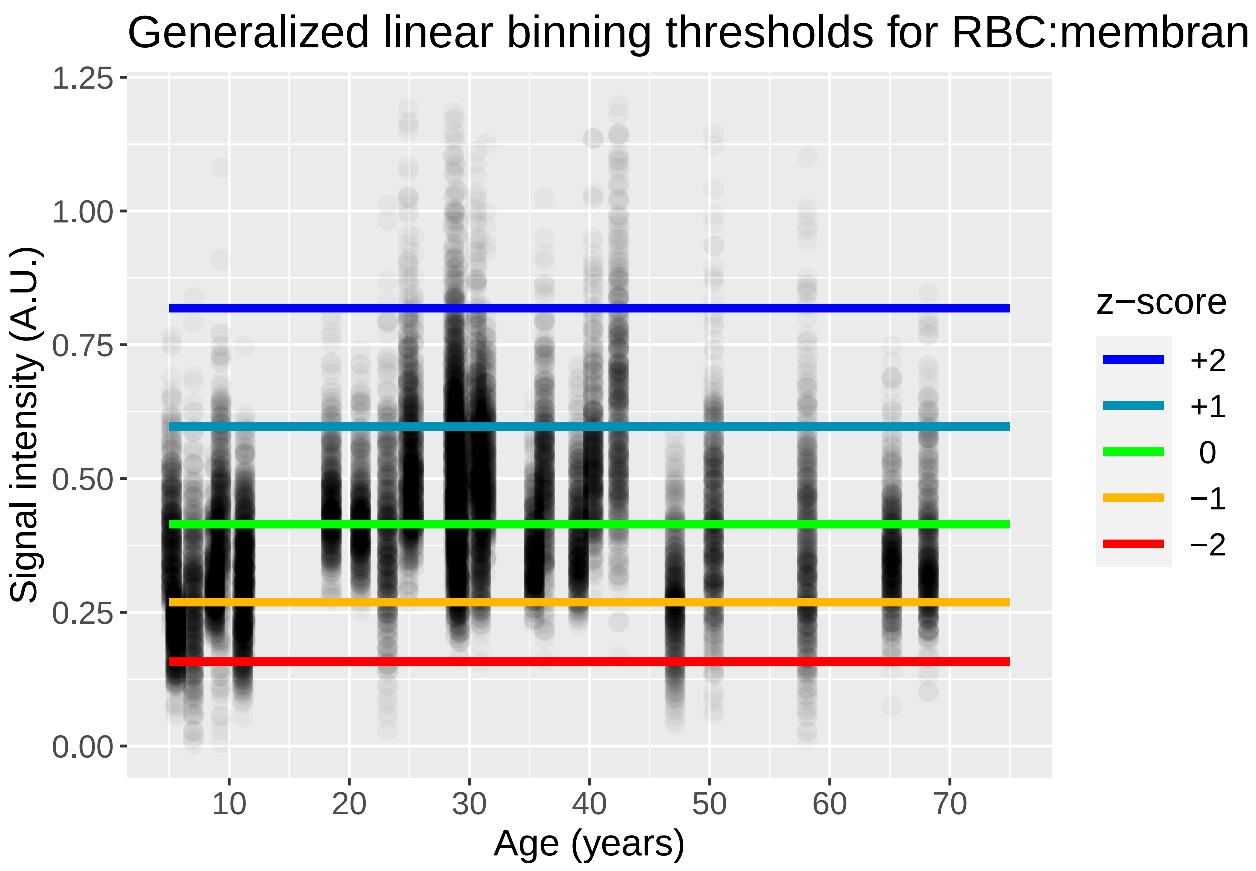
```r
# Declare distribution family:
fit_fam = NO(
  mu.link = "identity",
  sigma.link = "identity"
)

# Declare weighting for each subject:
weighting = rep(1 / num_voxels, nrow(Voxels))

# Declare a penalty, k: current agreed value is 3.84, corresponding to chi-square test. Default is 2.
chi = 3.84

# Run GAMLSS model:
voxel_mdl <- gamlss(Intensity ~ 1,
                    sigma.formula = ~ 1,
                    data = y,
                    family = fit_fam)

# Simulate an age range:
sim_age <- data.frame(Age = seq(5, 75, 0.25))

# Use the centiles.pred function to predict percentiles (can be correlated to a distribution z-score).
thresholds <- centiles.pred(
  voxel_mdl,
  xname = "Age",
  xvalues = sim_age$Age,
  cent = c(2.275013, 15.865525, 50, 84.134475, 97.724987)
)

# Store the predictions:
Pred <- data.frame(Age = sim_age$Age)
Pred$Age <- sim_age$Age
Pred$Thresh1 <- thresholds$`2.275013`
Pred$Thresh2 <- thresholds$`15.865525`
Pred$Thresh3 <- thresholds$`50`
Pred$Thresh4 <- thresholds$`84.134475`
Pred$Thresh5 <- thresholds$`97.724987`

# Plot the thresholds:
ggplot(data = y,
       aes(x = Age, y = Intensity)) +
  geom_point(alpha = 0.03,
             size = 3,
             color = "black") +
  geom_path(data = Pred,
            aes(x = Age, y = Thresh1, color = "1"),
            size = 1.5) +
  geom_path(data = Pred,
            aes(x = Age, y = Thresh2, color = "2"),
            size = 1.5) +
  geom_path(data = Pred,
            aes(x = Age, y = Thresh3, color = "3"),
            size = 1.5) +
  geom_path(data = Pred,
            aes(x = Age, y = Thresh4, color = "4"),
            size = 1.5) +
  geom_path(data = Pred,
            aes(x = Age, y = Thresh5, color = "5"),
            size = 1.5) +
  scale_colour_manual(values = c(rgb(1,0,0),
                                 rgb(1, 0.7143, 0),
                                 rgb(0, 1, 0),
                                 rgb(0, 0.57, 0.71),
                                 rgb(0, 0, 1)),
                      labels = c("-2",
                                 "-1",
                                 " 0",
                                 "+1",
                                 "+2")) +
  guides(color = guide_legend(reverse = TRUE)) +
  scale_x_continuous(limits = c(min(sim_age), max(sim_age)),
                     breaks = seq(10, 70,10)) +
  theme(legend.text = element_text(size = 12),
        legend.key.width = unit(1, 'cm'),
        title = element_text(size = 14),
        axis.text = element_text(size = 12),
        plot.margin = unit(c(1, 1, 1, 1), "mm")) +
  labs(x = "Age (years)",
       y = "Signal intensity (A.U.)",
       color = "z-score") +
  ylim(c(0, 1.2)) +
  ggtitle("Linear binning thresholds for RBC:membrane")
```



Generalized linear binning thresholds

```
# Declare distribution family:
fit_fam = BCCG(
  mu.link = "identity",
  sigma.link = "identity",
  nu.link = "identity"
)

# Declare weighting for each subject:
weighting = rep(1 / num_voxels, nrow(Voxels))

# Declare a penalty, k: current agreed value is 3.84, corresponding to chi-square test. Default is 2.
chi = 3.84

# Run GAMLSS model:
voxel_mdl <- gamlss(Intensity ~ 1,
                    sigma.formula = ~ 1,
                    data = y,
                    family = fit_fam)

# Simulate an age range:
sim_age <- data.frame(Age = seq(5, 75, 0.25))
```

```r
# Use the centiles.pred function to predict percentiles (can be correlated to a distribution z-score).
thresholds <- centiles.pred(
  voxel_mdl,
  xname = "Age",
  xvalues = sim_age$Age,
  cent = c(2.275013, 15.865525, 50, 84.134475, 97.724987)
)

# Store the predictions:
Pred <- data.frame(Age = sim_age$Age)
Pred$Age <- sim_age$Age
Pred$Thresh1 <- thresholds$`2.275013`
Pred$Thresh2 <- thresholds$`15.865525`
Pred$Thresh3 <- thresholds$`50`
Pred$Thresh4 <- thresholds$`84.134475`
Pred$Thresh5 <- thresholds$`97.724987`

# Plot the thresholds:
ggplot(data = y,
       aes(x = Age, y = Intensity)) +
  geom_point(alpha = 0.03,
             size = 3,
             color = "black") +
  geom_path(data = Pred,
            aes(x = Age, y = Thresh1, color = "1"),
            size = 1.5) +
  geom_path(data = Pred,
            aes(x = Age, y = Thresh2, color = "2"),
            size = 1.5) +
  geom_path(data = Pred,
            aes(x = Age, y = Thresh3, color = "3"),
            size = 1.5) +
  geom_path(data = Pred,
            aes(x = Age, y = Thresh4, color = "4"),
            size = 1.5) +
  geom_path(data = Pred,
            aes(x = Age, y = Thresh5, color = "5"),
            size = 1.5) +
  scale_colour_manual(values = c(rgb(1,0,0),
                                 rgb(1, 0.7143, 0),
                                 rgb(0, 1, 0),
                                 rgb(0, 0.57, 0.71),
                                 rgb(0, 0, 1)),
                      labels = c("-2",
                                 "-1",
                                 " 0",
                                 "+1",
                                 "+2")) +
  guides(color = guide_legend(reverse = TRUE)) +
  scale_x_continuous(limits = c(min(sim_age), max(sim_age)),
                     breaks = seq(10, 70,10)) +
  theme(legend.text = element_text(size = 12),
        legend.key.width = unit(1, 'cm'),
        title = element_text(size = 14),
        axis.text = element_text(size = 12),
        plot.margin = unit(c(1, 1, 1, 1), "mm")) +
  labs(x = "Age (years)",
       y = "Signal intensity (A.U.)",
       color = "z-score") +
  ylim(c(0, 1.2)) +
  ggtitle("Generalized linear binning thresholds for RBC:membrane")
```


Generalized linear binning thresholds for RBC:membran
Signal intensity (A.U.)
Age (years)
z−score