Oh the roGFPs you will see

Document setup options

```
knitr::opts_chunk$set(echo = TRUE)
library(knitr)
opts_chunk$set(tidy.opts=list(width.cutoff=55),tidy=TRUE)
shh <- suppressPackageStartupMessages
shh(require(sensorOverlord))
shh(require(ggplot2))
shh(require(cowplot))
shh(require(stringr))
shh(require(ggalt))
shh(require(tidyverse))
library(RColorBrewer)
```

Initalize Sensors

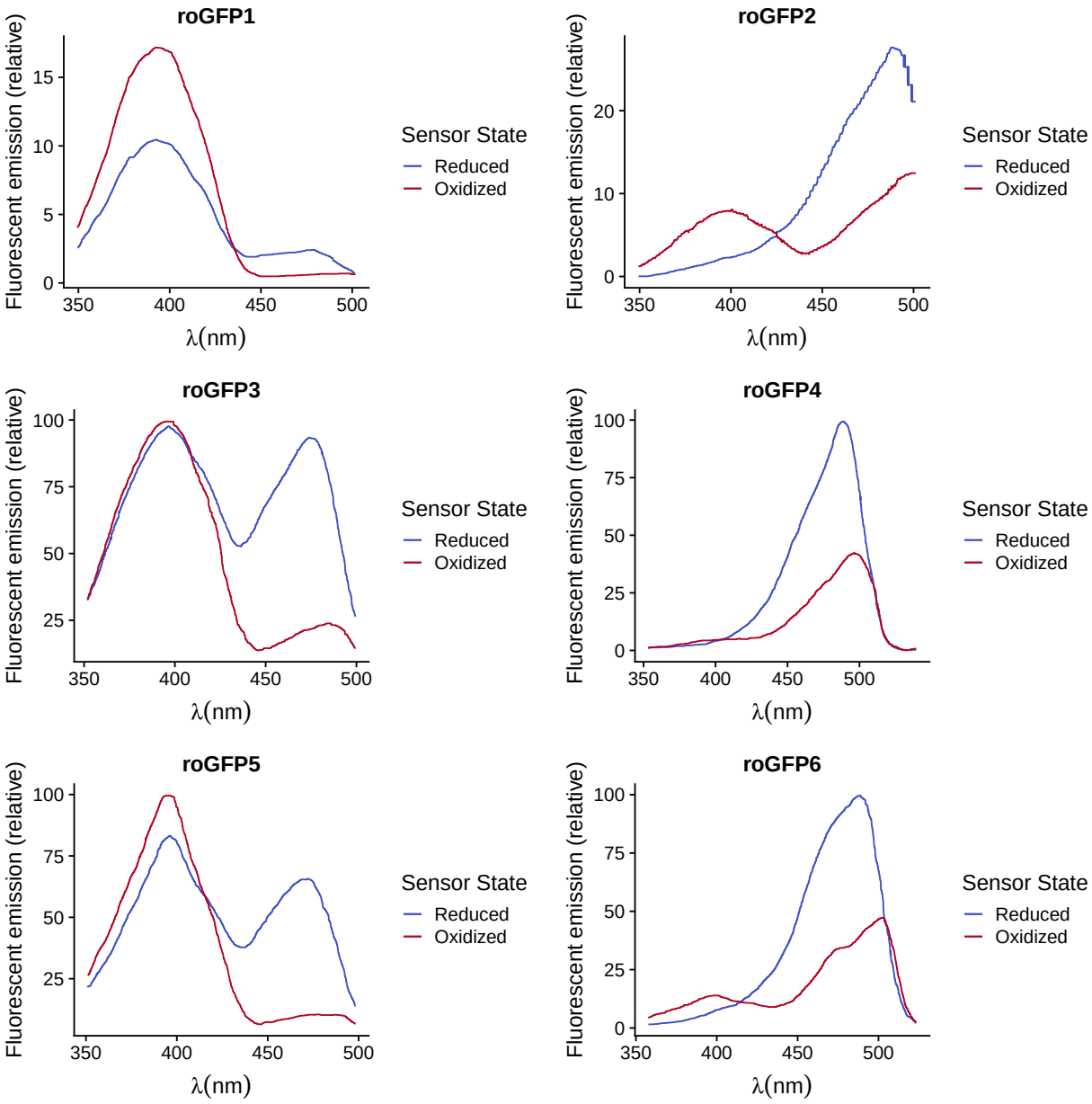
```
sensor_repo <- "../Raw_Spectra/"

# roGFP1
roGFP1_data <- read.csv(paste(sensor_repo, "rogfp1.csv",
    sep = ""), header = TRUE)
roGFP1_spectra <- spectraMatrixFromValues(lambdas_minimum = roGFP1_data$Lambda_Reduced,
    values_minimum = roGFP1_data$Values_Reduced, lambdas_maximum = roGFP1_data$Lambda_Oxidized,
    values_maximum = roGFP1_data$Values_Oxidized)
roGFP1_sensor <- new("redoxSensor", Rmin = 4.3, Rmax = 30.6,
    delta = 0.2, e0 = -281)

# roGFP1-R9
roGFP1_R9_data <- read.csv(paste(sensor_repo, "rogfp1_R9.csv",
    sep = ""), header = TRUE)
roGFP1_R9_spectra <- spectraMatrixFromValues(lambdas_minimum = roGFP1_R9_data$Lambda_reduced,
    values_minimum = roGFP1_R9_data$reduced, lambdas_maximum = roGFP1_R9_data$Lambda_oxidized,
    values_maximum = roGFP1_R9_data$oxidized)
roGFP1_R9_sensor <- new("redoxSensor", newSensorFromSpectra(roGFP1_R9_spectra,
    lambda_1 = c(380, 400), lambda_2 = c(460, 480)), e0 = -278)

# roGFP1-R12 empirical
roGFP1_R12_empirical_sensor <- new("redoxSensor", Rmin = 0.667,
    Rmax = 5.207, delta = 0.171, e0 = -265)

# roGFP1-R12 from spectra
roGFP1_R12_data <- read.csv(paste(sensor_repo, "rogfp1_R12.csv",
    sep = ""), header = FALSE)
roGFP1_R12_spectra <- spectraMatrixFromValues(lambdas_minimum = roGFP1_R12_data$V3,
    values_minimum = roGFP1_R12_data$V4, lambdas_maximum = roGFP1_R12_data$V1,
    values_maximum = roGFP1_R12_data$V2)
roGFP1_R12_sensor <- new("redoxSensor", newSensorFromSpectra(roGFP1_R12_spectra,
    lambda_1 = c(390, 410), lambda_2 = c(460, 480)), e0 = -265)
```

```r
# roGFP1_iE
roGFP1_iE_data <- read.csv(paste(sensor_repo, "rogfp1_iE.csv",
    sep = ""), header = FALSE, fileEncoding = "UTF-8-BOM")
roGFP1_iE_spectra <- spectraMatrixFromValues(lambdas_minimum = roGFP1_iE_data$V3,
    values_minimum = roGFP1_iE_data$V4, lambdas_maximum = roGFP1_iE_data$V1,
    values_maximum = roGFP1_iE_data$V2)
roGFP1_iE_sensor <- new("redoxSensor", Rmin = 0.856, Rmax = 3.875,
    delta = 0.5, e0 = -236)

# roGFP2
roGFP2_data <- read.csv(paste(sensor_repo, "rogfp2.csv",
    sep = ""), header = FALSE, fileEncoding = "UTF-8-BOM")
roGFP2_spectra <- spectraMatrixFromValues(lambdas_minimum = roGFP2_data$V3,
    values_minimum = roGFP2_data$V4, lambdas_maximum = roGFP2_data$V1,
    values_maximum = roGFP2_data$V2)
roGFP2_sensor <- new("redoxSensor", Rmin = 0.09, Rmax = 1.7,
    delta = 0.3, e0 = -272)

# grx1_roGFP2
grx1_roGFP2_sensor <- new("redoxSensor", Rmin = 0.3, Rmax = 2,
    delta = 0.5, e0 = -272)

# roGFP2_iL
roGFP2_iL_data <- read.csv(paste(sensor_repo, "rogfp2_iL.csv",
    sep = ""), header = FALSE, fileEncoding = "UTF-8-BOM")
roGFP2_iL_spectra <- spectraMatrixFromValues(lambdas_minimum = roGFP2_iL_data$V3,
    values_minimum = roGFP2_iL_data$V4, lambdas_maximum = roGFP2_iL_data$V1,
    values_maximum = roGFP2_iL_data$V2)
roGFP2_iL_sensor <- new("redoxSensor", Rmin = 0.19, Rmax = 0.45,
    delta = 0.65, e0 = -229)

# roGFP3
roGFP3_data <- read.csv(paste(sensor_repo, "rogfp3.csv",
    sep = ""), header = TRUE)
roGFP3_spectra <- spectraMatrixFromValues(lambdas_minimum = roGFP3_data$Lambda_330,
    values_minimum = roGFP3_data$X.330.mv, lambdas_maximum = roGFP3_data$Lambda_240,
    values_maximum = roGFP3_data$X.240.mv)
roGFP3_sensor <- new("redoxSensor", newSensorFromSpectra(roGFP3_spectra,
    lambda_1 = c(380, 400), lambda_2 = c(460, 480)), e0 = -299)


# roGFP4
roGFP4_data <- read.csv(paste(sensor_repo, "rogfp4.csv",
    sep = ""), header = TRUE)
roGFP4_spectra <- spectraMatrixFromValues(lambdas_minimum = roGFP4_data$Lambda_320,
    values_minimum = roGFP4_data$X.320.mv, lambdas_maximum = roGFP4_data$Lambda_230,
    values_maximum = roGFP4_data$X.230.mv)
roGFP4_sensor <- new("redoxSensor", newSensorFromSpectra(roGFP4_spectra,
    lambda_1 = c(380, 400), lambda_2 = c(460, 480)), e0 = -286)

roGFP4_sensor <- new("redoxSensor", newSensorFromSpectra(roGFP4_spectra,
    lambda_1 = c(505, 515), lambda_2 = c(465, 475)), e0 = -286)
```

```
# roGFP5
roGFP5_data <- read.csv(paste(sensor_repo, "rogfp5.csv",
    sep = ""), header = TRUE)
roGFP5_spectra <- spectraMatrixFromValues(lambdas_minimum = roGFP5_data$Lambda_330,
    values_minimum = roGFP5_data$X.330.mv, lambdas_maximum = roGFP5_data$Lambda_250,
    values_maximum = roGFP5_data$X.250.mv)
roGFP5_sensor <- new("redoxSensor", newSensorFromSpectra(roGFP5_spectra,
    lambda_1 = c(380, 400), lambda_2 = c(460, 480)), e0 = -296)
# roGFP6
roGFP6_data <- read.csv(paste(sensor_repo, "rogfp6.csv",
    sep = ""), header = TRUE)
roGFP6_spectra <- spectraMatrixFromValues(lambdas_minimum = roGFP6_data$Lambda_310,
    values_minimum = roGFP6_data$X.310.mv, lambdas_maximum = roGFP6_data$Lambda_230,
    values_maximum = roGFP6_data$X.230.mv)
roGFP6_sensor <- new("redoxSensor", newSensorFromSpectra(roGFP6_spectra,
    lambda_1 = c(380, 400), lambda_2 = c(460, 480)), e0 = -280)
```

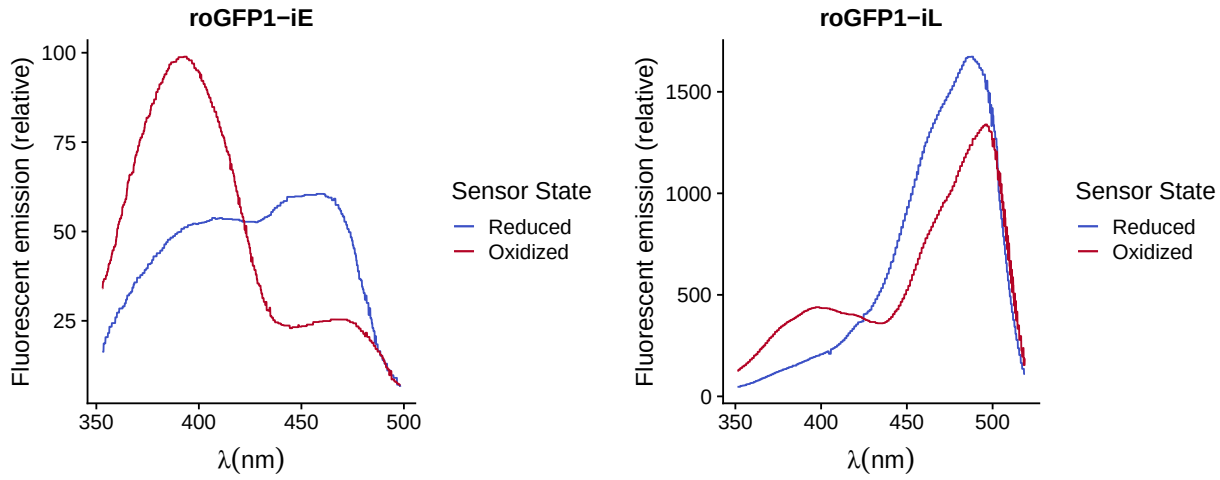
Original roGFP spectra

```
gfp1_spectraPlot <- plotSpectra(roGFP1_spectra, "Reduced",
    "Oxidized") + ggtitle("roGFP1")
gfp2_spectraPlot <- plotSpectra(roGFP2_spectra, "Reduced",
    "Oxidized") + ggtitle("roGFP2")
gfp3_spectraPlot <- plotSpectra(roGFP3_spectra, "Reduced",
    "Oxidized") + ggtitle("roGFP3")
gfp4_spectraPlot <- plotSpectra(roGFP4_spectra, "Reduced",
    "Oxidized") + ggtitle("roGFP4")
gfp5_spectraPlot <- plotSpectra(roGFP5_spectra, "Reduced",
    "Oxidized") + ggtitle("roGFP5")
gfp6_spectraPlot <- plotSpectra(roGFP6_spectra, "Reduced",
    "Oxidized") + ggtitle("roGFP6")

plot_grid(gfp1_spectraPlot, gfp2_spectraPlot, gfp3_spectraPlot,
    gfp4_spectraPlot, gfp5_spectraPlot, gfp6_spectraPlot,
    ncol = 2)
```



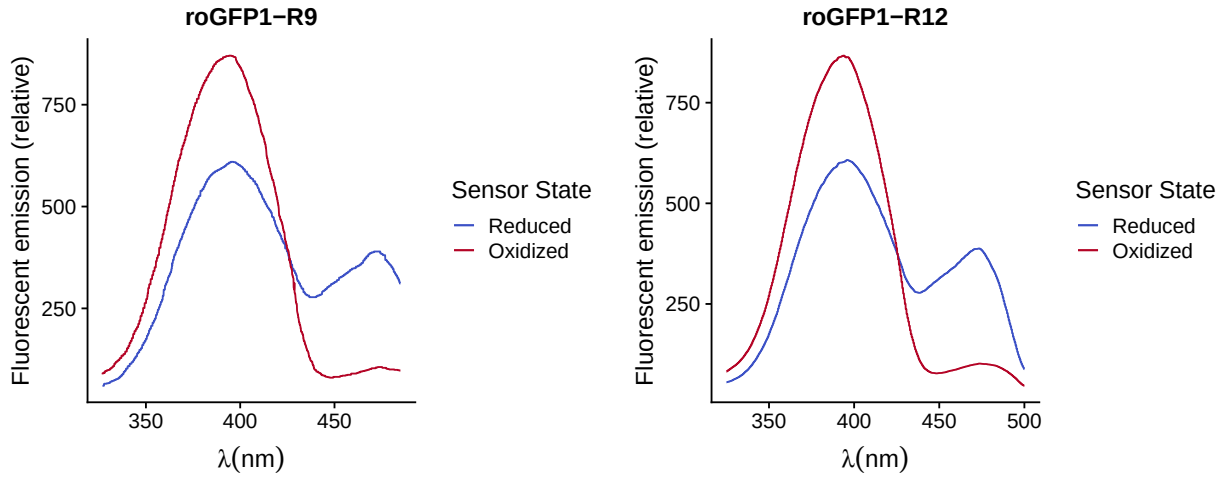
iL and iE spectra

```
gfp1_iE_spectraPlot <- plotSpectra(roGFP1_iE_spectra, "Reduced",
    "Oxidized") + ggtitle("roGFP1-iE")
gfp2_iL_spectraPlot <- plotSpectra(roGFP2_iL_spectra, "Reduced",
    "Oxidized") + ggtitle("roGFP1-iL")

plot_grid(gfp1_iE_spectraPlot, gfp2_iL_spectraPlot)
```



R12 and R9

```
gfp1_R12_spectraPlot <- plotSpectra(roGFP1_R12_spectra,
    "Reduced", "Oxidized") + ggtitle("roGFP1-R12")
gfp1_R9_spectraPlot <- plotSpectra(roGFP1_R9_spectra, "Reduced",
    "Oxidized") + ggtitle("roGFP1-R9")

plot_grid(gfp1_R9_spectraPlot, gfp1_R12_spectraPlot)
```



12 Total sensors created

- roGFP1 - roGFP6 (6)
- roGFP1-iE and roGFP2-iL (2)
- roGFP1-R9 and roGFP1-R12 and empirical roGFP1-R12 (3)
- grx1_roGFP2 (No spectra, 1)

```
q <- function(...) {
    sapply(match.call()[-1], deparse)
}

sensorList <- q(roGFP1_sensor, roGFP2_sensor, roGFP3_sensor,
```

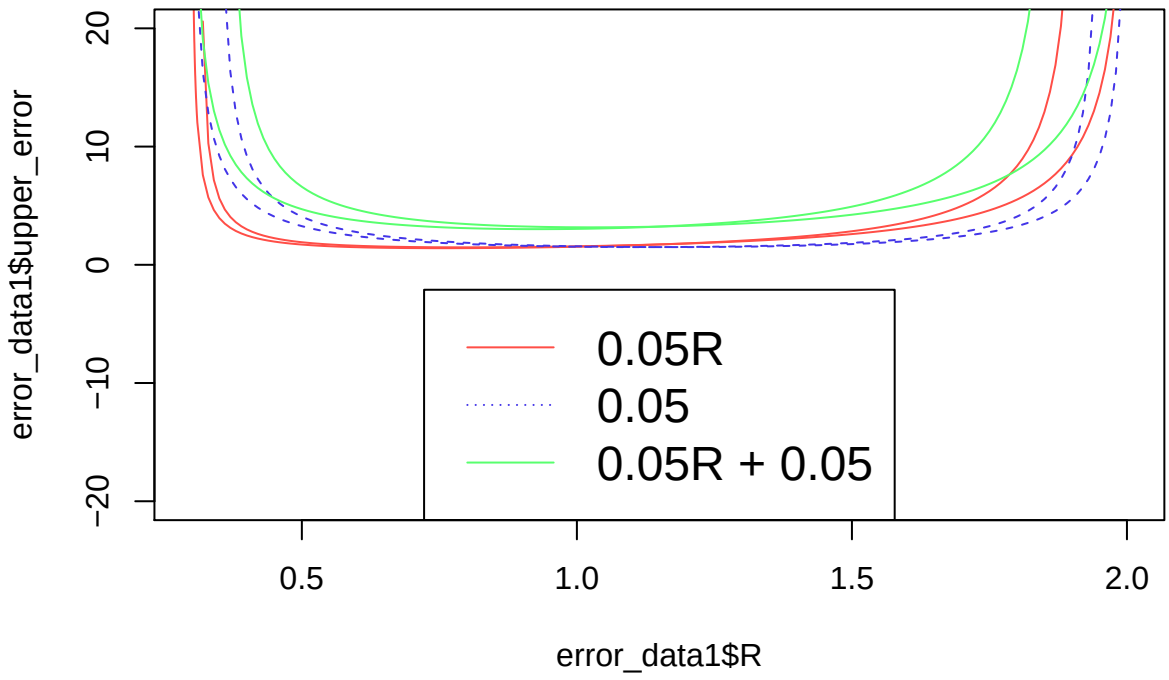
```
    roGFP4_sensor, roGFP5_sensor, roGFP6_sensor, roGFP1_iE_sensor,
    roGFP2_iL_sensor, roGFP1_R9_sensor, roGFP1_R12_sensor,
    roGFP1_R12_empirical_sensor, grx1_roGFP2_sensor)

charsMatrix <- c()

for (sensorName in sensorList) {
    sensor <- get(sensorName)
    charsMatrix <- rbind(charsMatrix, c(sensorName, round(sensor@Rmin,
        2), round(sensor@Rmax, 2), round(sensor@delta, 2),
        round(sensor@e0, 2)))
}

knitr::kable(data.frame(charsMatrix), col.names = c("Sensor",
    "Rmin", "Rmax", "Delta", "e0"), digits = 2)
```

Sensor	Rmin	Rmax	Delta	e0
roGFP1_sensor	4.3	30.6	0.2	-281
roGFP2_sensor	0.09	1.7	0.3	-272
roGFP3_sensor	1.04	4.69	0.23	-299
roGFP4_sensor	0.39	1.03	0.36	-286
roGFP5_sensor	1.19	9.34	0.16	-296
roGFP6_sensor	0.06	0.41	0.36	-280
roGFP1_iE_sensor	0.86	3.88	0.5	-236
roGFP2_iL_sensor	0.19	0.45	0.65	-229
roGFP1_R9_sensor	1.58	8.53	0.27	-278
roGFP1_R12_sensor	1.57	8.41	0.26	-265
roGFP1_R12_empirical_sensor	0.67	5.21	0.17	-265
grx1_roGFP2_sensor	0.3	2	0.5	-272

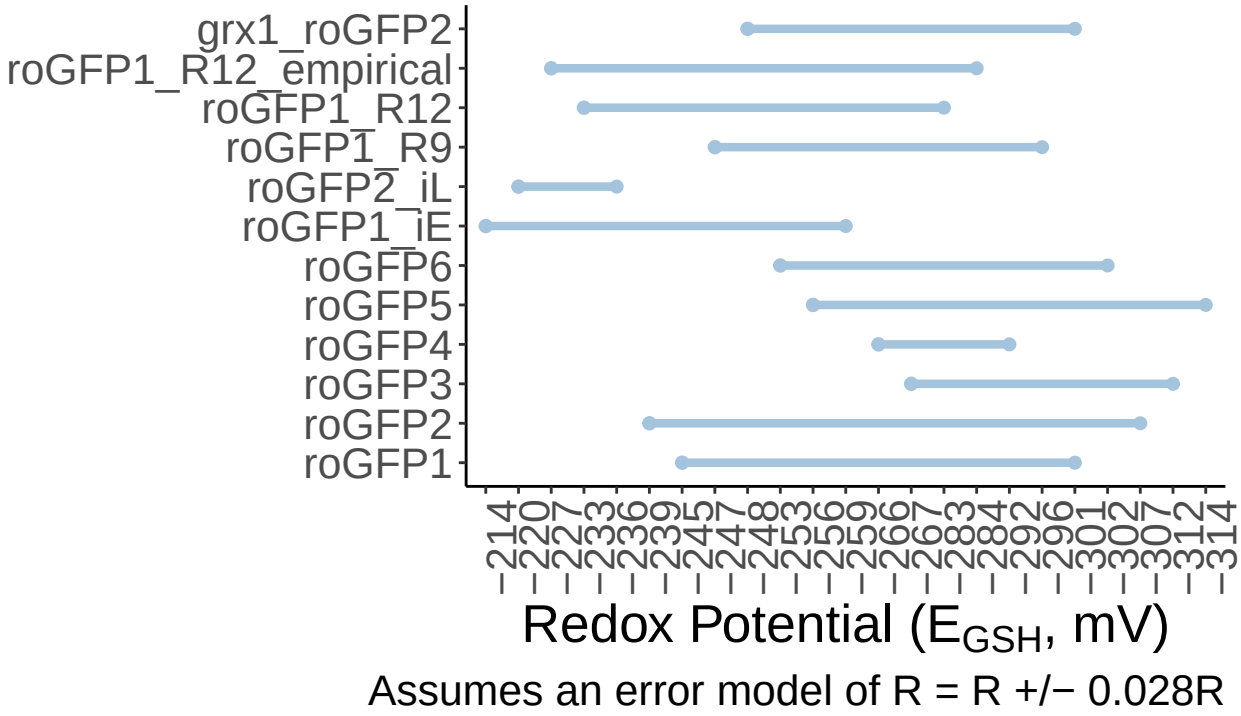
```
acceptable_error <- 2
error_model <- function(x) {
    return(0.028 * x)
}

minMaxMatrix <- c()

for (sensorName in sensorList) {
    sensor <- get(sensorName)
    sensorName <- str_replace(sensorName, "_sensor", "")
    error_data <- getErrorTable(sensor, R = getR(sensor),
        FUN = getE, Error_Model = error_model)

    error_filter <- subset(error_data, error_data$max_abs_error <
        acceptable_error)

    minimum <- ifelse(test = length(error_filter$FUN_true) ==
        0, yes = NaN, no = min(error_filter$FUN_true))

    maximum <- ifelse(test = length(error_filter$FUN_true) ==
        0, yes = NaN, no = max(error_filter$FUN_true))
```

```
    minMaxMatrix <- rbind(minMaxMatrix, c(sensorName, round(minimum,
        0), round(maximum, 0)))
}

ranges <- data.frame(minMaxMatrix)
colnames(ranges) <- c("Sensor_Name", "Minimum", "Maximum")
```

```
error_model <- function(x) {
    return(0.05 * x)
}
error_data1 <- getErrorTable(sensor, R = getR(sensor), FUN = getE,
    Error_Model = error_model)

error_model <- function(x) {
    return(0.05)
}
error_data2 <- getErrorTable(sensor, R = getR(sensor), FUN = getE,
    Error_Model = error_model)

error_model <- function(x) {
    return(0.05 + 0.05 * x)
}
error_data3 <- getErrorTable(sensor, R = getR(sensor), FUN = getE,
    Error_Model = error_model)


# plot(error_data1$FUN_true ~ error_data1$R, type = 'l',
# ylim = c(-280, -220))

redCol <- "#FF4E47"
blueCol <- "#4335E8"
greenCol <- "#58FF6E"

plot(error_data1$upper_error ~ error_data1$R, type = "l",
    col = redCol, ylim = c(-20, 20))
points(error_data1$lower_error ~ error_data1$R, type = "l",
    col = redCol)
points(error_data2$upper_error ~ error_data2$R, type = "l",
    lty = "dashed", col = blueCol)
points(error_data2$lower_error ~ error_data2$R, type = "l",
    lty = "dashed", col = blueCol)
points(error_data3$upper_error ~ error_data2$R, type = "l",
    col = greenCol)
points(error_data3$lower_error ~ error_data2$R, type = "l",
    col = greenCol)

legend("bottom", legend = c("0.05R", "0.05", "0.05R + 0.05"),
    col = c(redCol, blueCol, greenCol), lty = c(1, 3, 1),
    cex = 1.5)
```

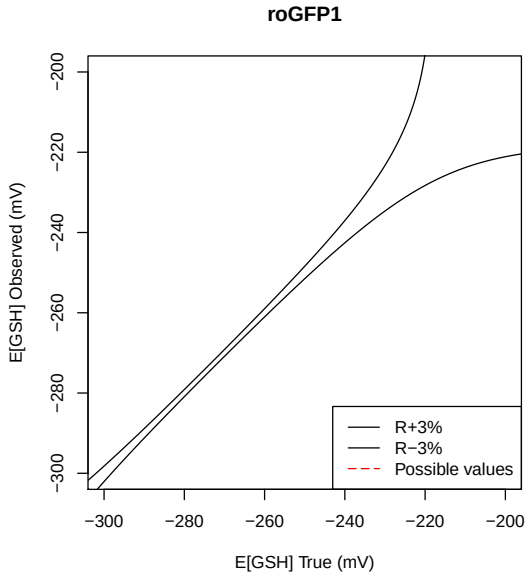


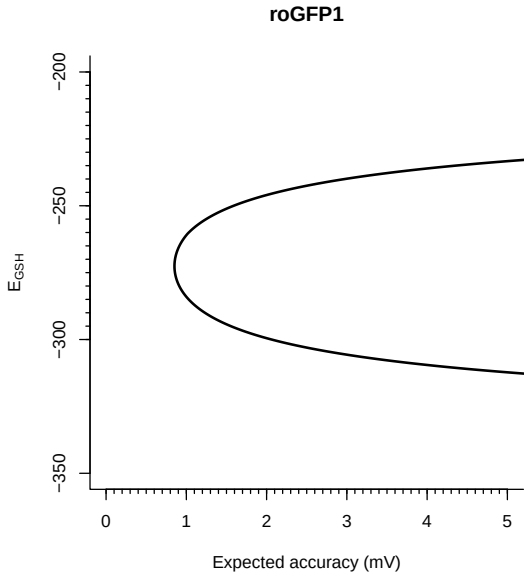

```
theme_set(theme_classic())

ranges$Sensor_Name <- factor(ranges$Sensor_Name,
                             levels=as.character(ranges$Sensor_Name))

gg <- ggplot(ranges, aes(x=Minimum, xend=Maximum,
                         y = Sensor_Name, group=Sensor_Name)) +
        geom_dumbbell(color="#a3c4dc",
                      size=1.5) +
           labs(x = expression("Redox Potential (" * E[GSH] * ", mV" * ")"),
             y=NULL,
             title="",
             caption="Assumes an error model of R = R +/- 0.028R",
             subtitle="") +
    theme(plot.title = element_text(hjust=0.5, face="bold"),
              #plot.background=element_rect(fill="#f7f7f7"),
              #panel.background=element_rect(fill="#f7f7f7"),
              panel.grid.minor=element_blank(),
              panel.grid.major.y=element_blank(),
              legend.position="top",
              panel.border=element_blank(),
          text = element_text(size = 20),
          axis.text.x = element_text(angle = 90, hjust = 1))
plot(gg)
```

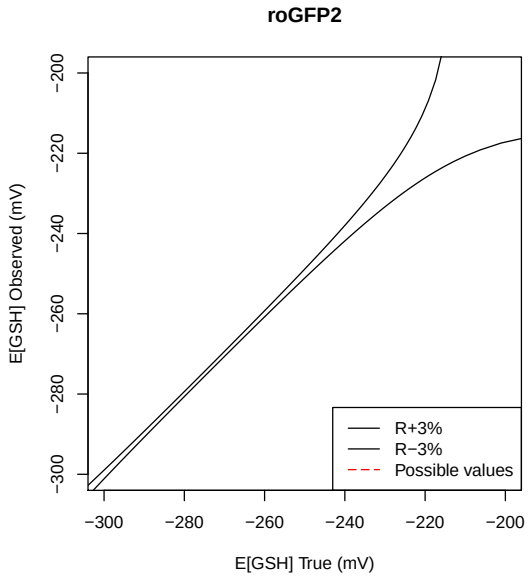


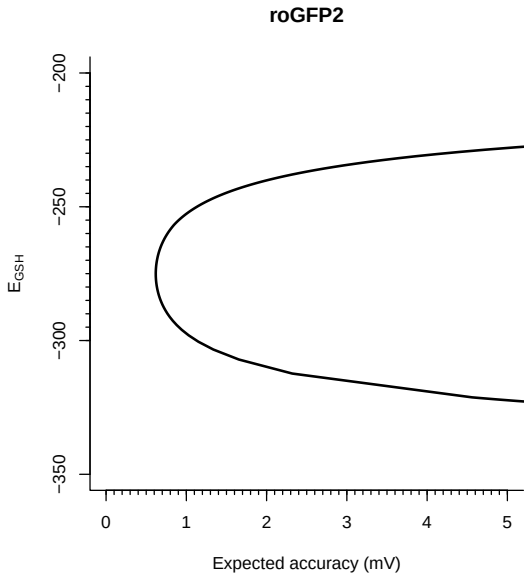

```
## [1] "roGFP1"
```

```
## [1] "roGFP2"
```



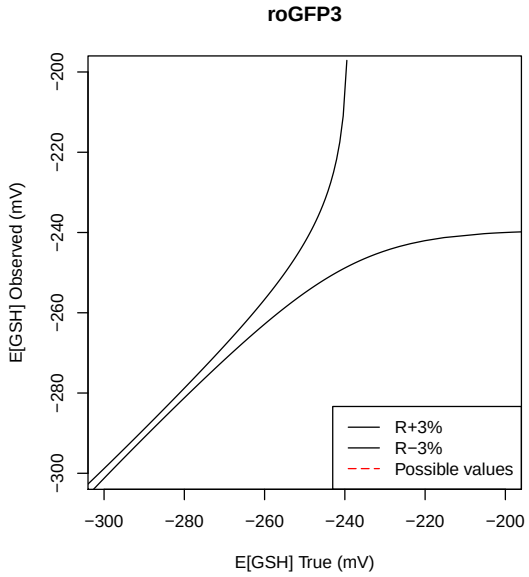


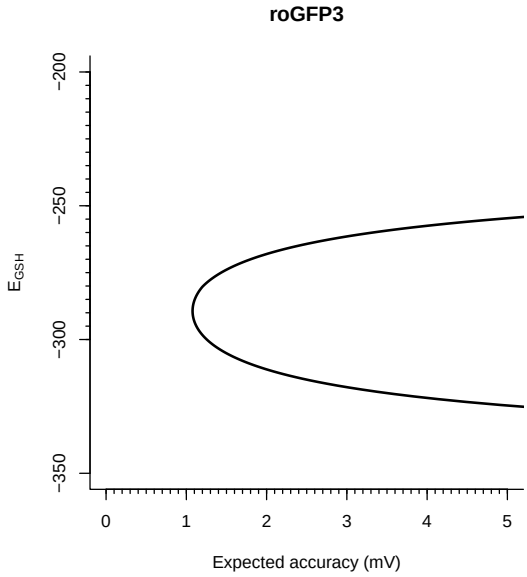


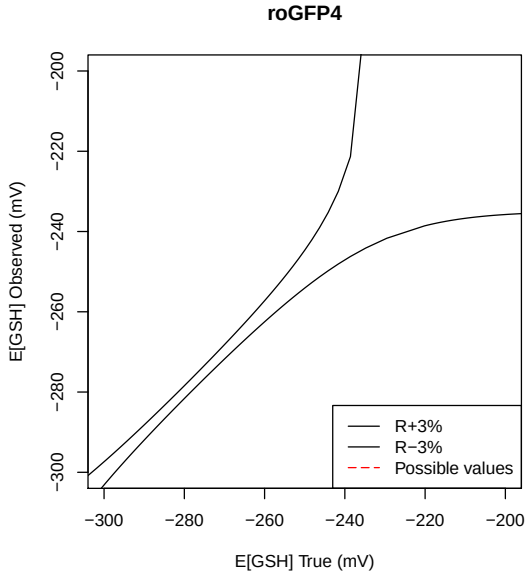


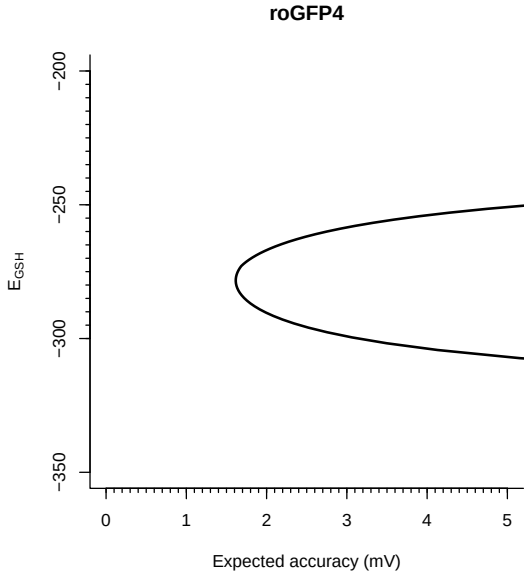

```
## [1] "roGFP3"
```

```
## [1] "roGFP4"
```



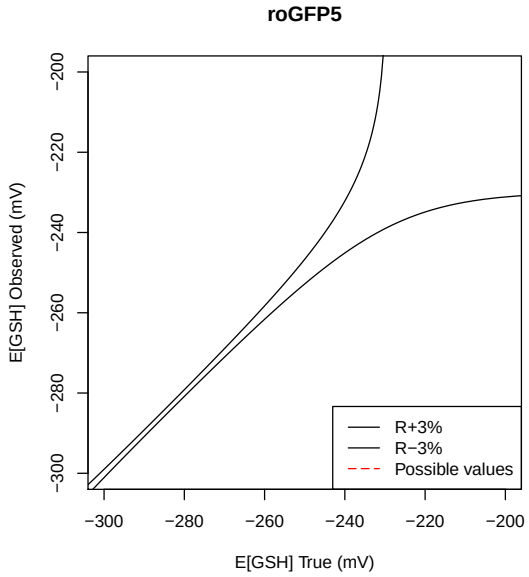


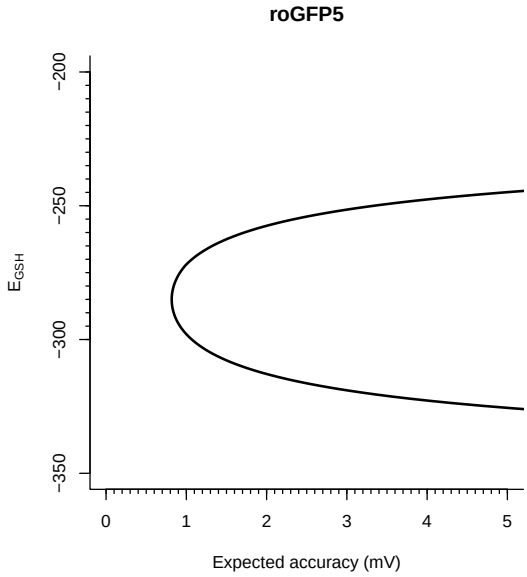


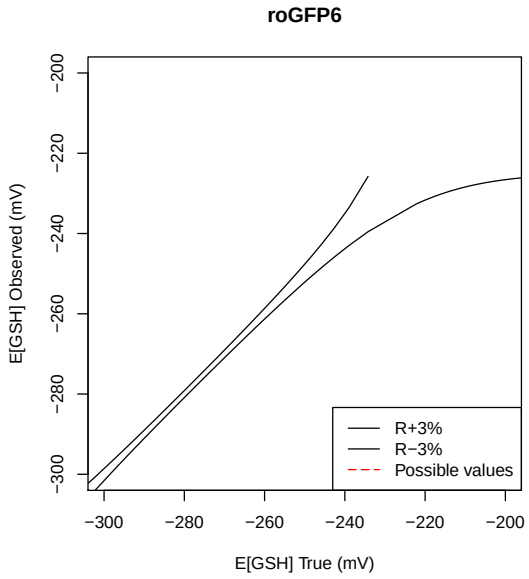


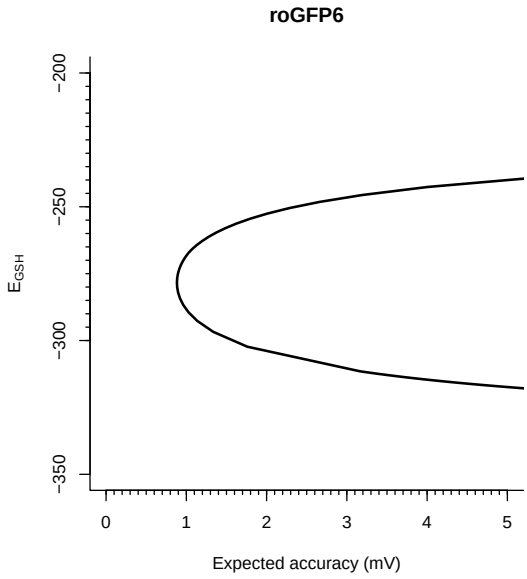

```
## [1] "roGFP5"
```

```
## [1] "roGFP6"
```



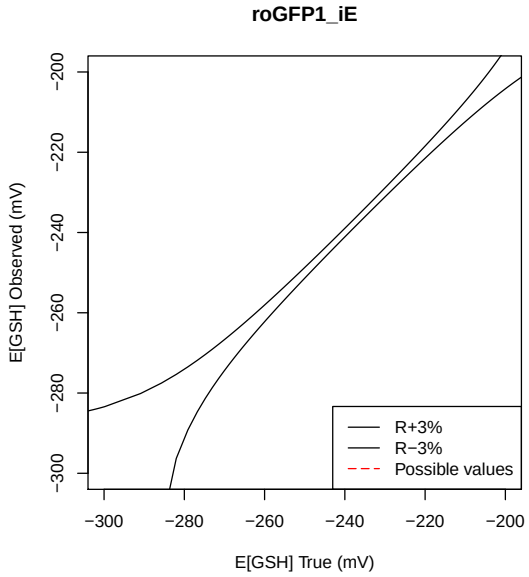


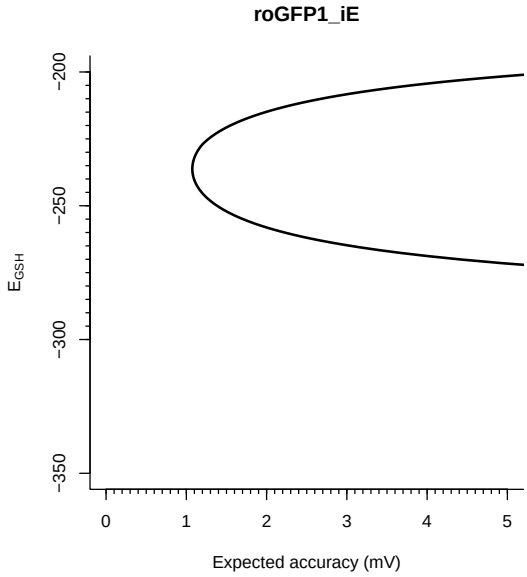


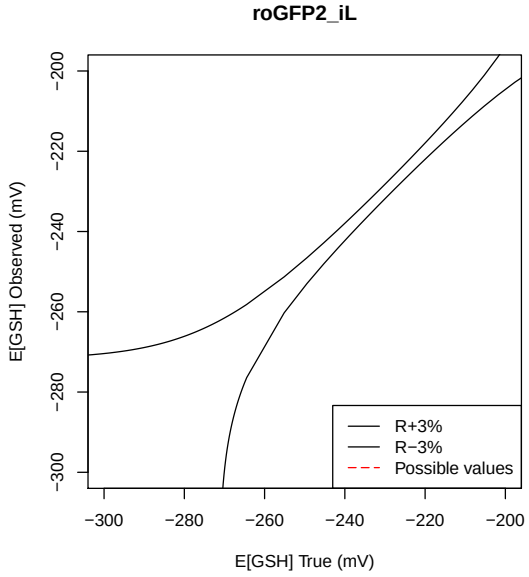


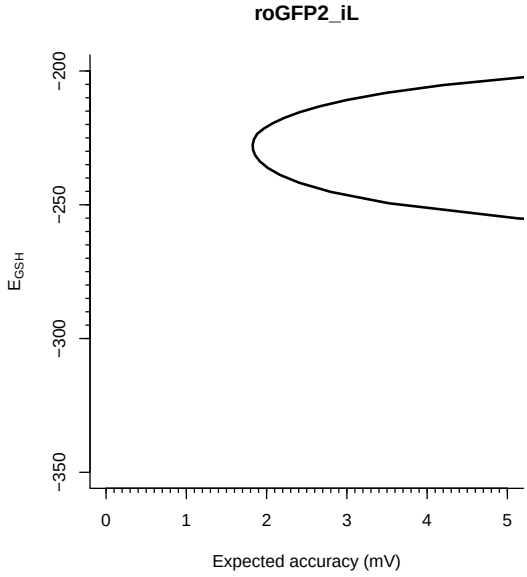

```
## [1] "roGFP1_iE"
```

```
## [1] "roGFP2_iL"
```



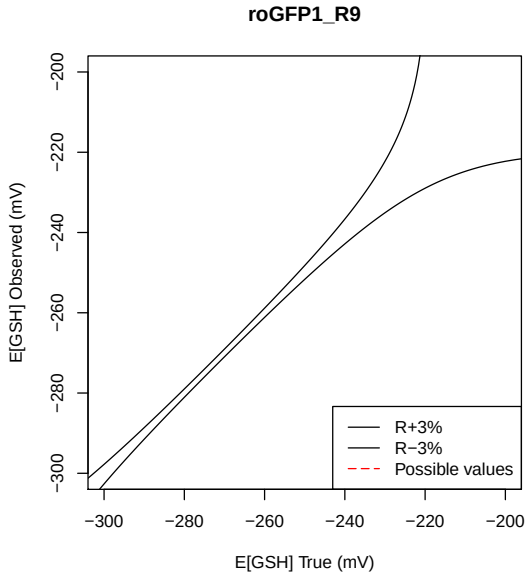


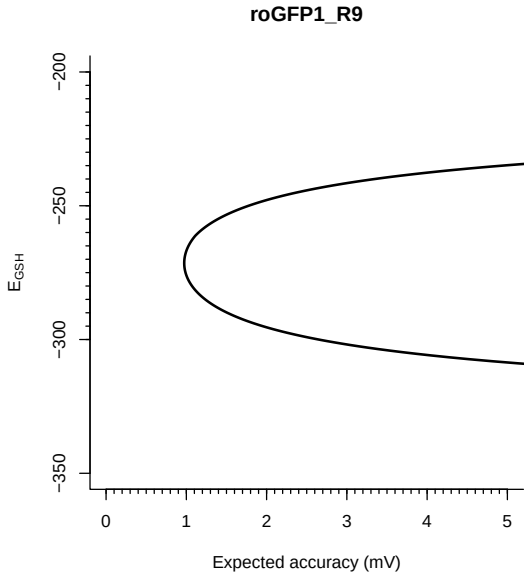


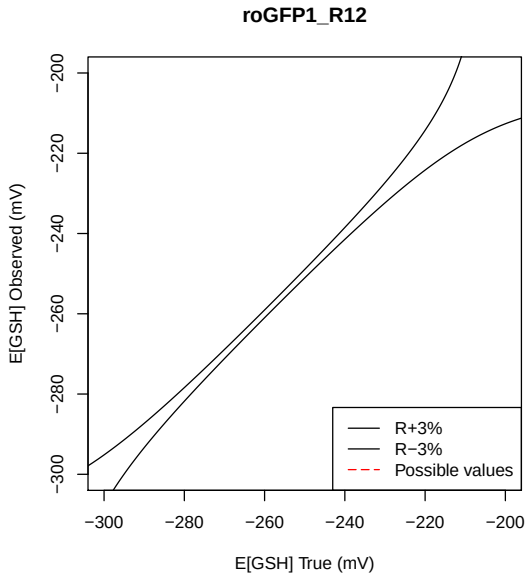


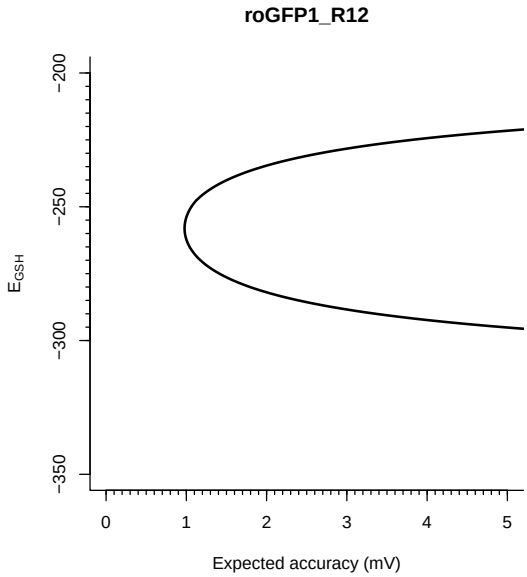

```
## [1] "roGFP1_R9"
```

```
## [1] "roGFP1_R12"
```

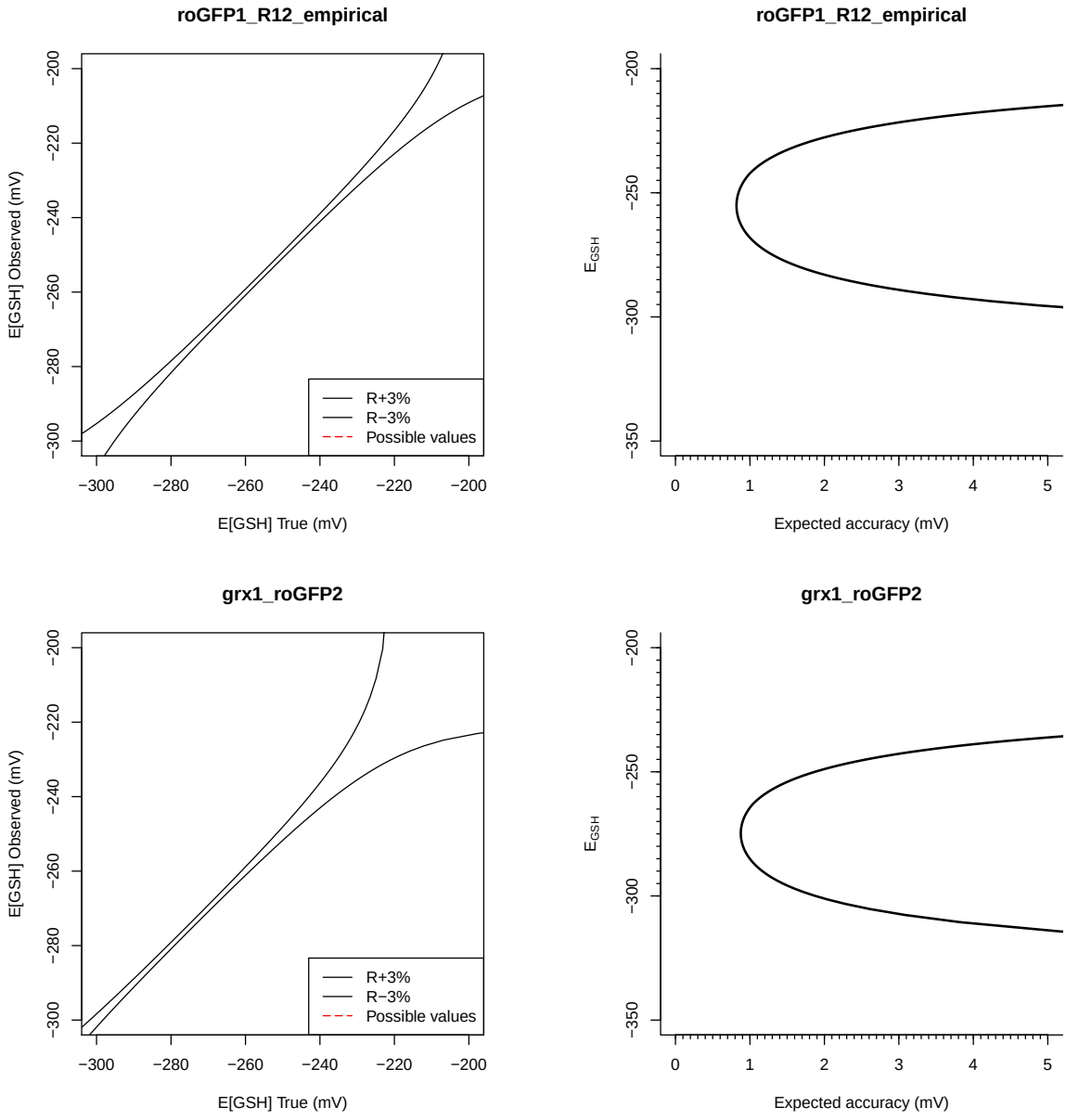









```
## [1] "roGFP1_R12_empirical"
## [1] "grx1_roGFP2"
```



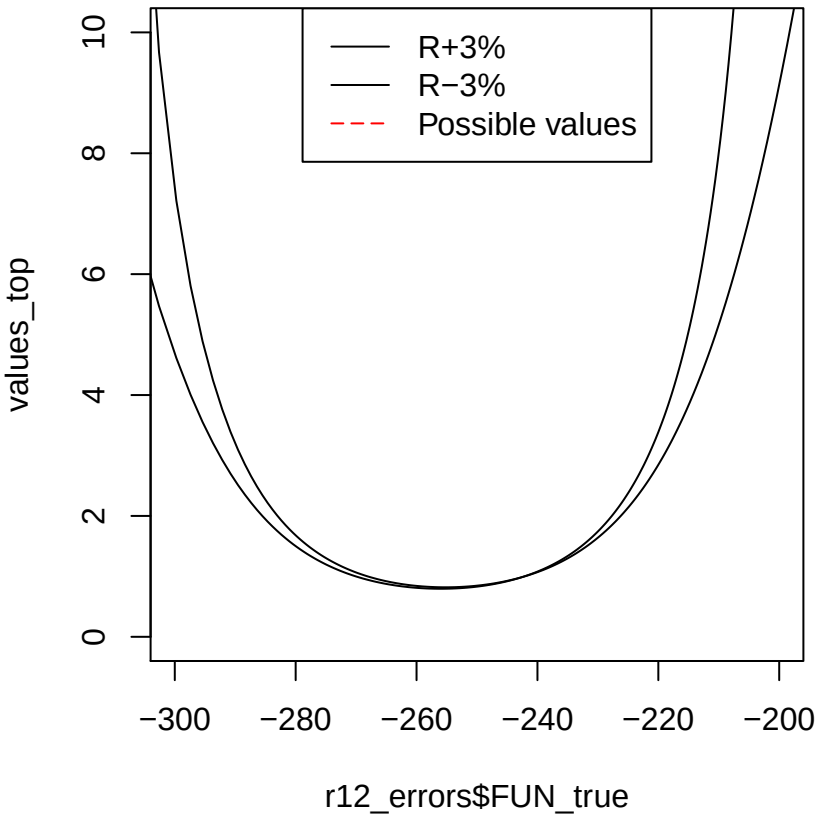
Extra

```
r12_errors <- getErrorTable(roGFP1_R12_empirical_sensor,
    R = getR(roGFP1_R12_empirical_sensor), Error_Model = function(x) {
        return(x * 0.03)
    }, FUN = getE)

r12_errors$lower_error_neg <- (r12_errors$lower_error *
    -1)

r12_errors$lower_value <- (r12_errors$FUN_true + r12_errors$lower_error_neg)
r12_errors$upper_value <- (r12_errors$FUN_true + r12_errors$upper_error)
```

```
r12_trunc <- data.frame(true = rep(r12_errors$FUN_true,
    2), observed = c(r12_errors$lower_value, r12_errors$upper_value),
    error = c(r12_errors$lower_error_neg, r12_errors$upper_error),
    absError = c(r12_errors$lower_error, r12_errors$upper_error),
    maxAbsError = rep(r12_errors$max_abs_error, 2), type = c(rep("lower",
        length(r12_errors$FUN_true)), rep("upper", length(r12_errors$FUN_true))))
```

```
# par(pty = 's', mfrow = c(1,1), mai = c(0.8, 0.8, 0.4,
# 0.8)) values_actual <- r12_errors$FUN_true values_top
# <- r12_errors$upper_error values_bottom <-
# r12_errors$lower_error_neg plot(values_top ~
# r12_errors$FUN_true , type = 'l', ylim = c(-20, 20),
# xlim = c(-300, -200), lwd = 1) points(values_bottom
# ~r12_errors$FUN_true, type = 'l', lwd = 1) limits <-
# c(5, 435)
# polygon(c(values_actual[limits[1]:limits[2]],
# rev(values_actual[limits[1]:limits[2]])), y =
# c(values_top[limits[1]:limits[2]],
# rev(values_bottom[limits[1]:limits[2]])), col = rgb(1,
# 0, 0, alpha = 1), density = 50, lty = 'dotted')
# legend('top', legend = c('R+3%', 'R-3%', 'Possible
# values'), col = c('black', 'black', rgb(1,0,0,alpha =
# 1)), lty = c(1, 1, 5))
```

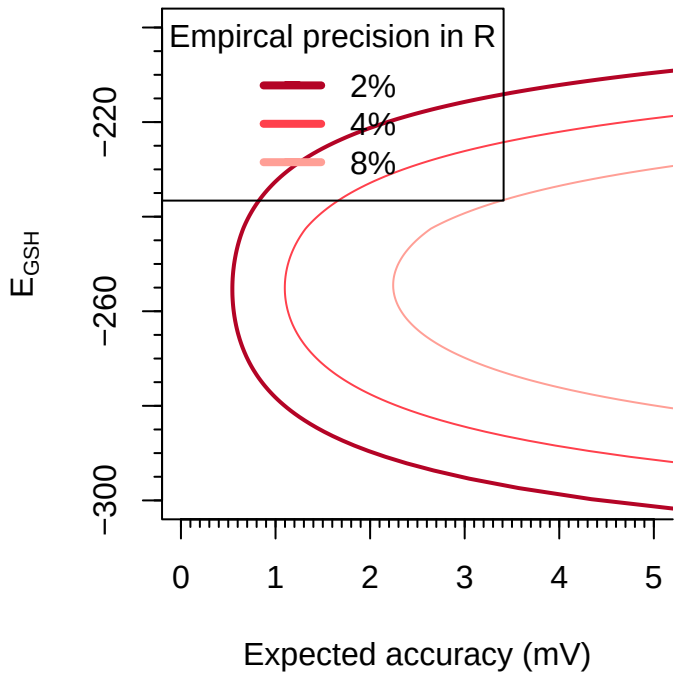
```
par(pty = "s", mfrow = c(1, 1), mai = c(0.8, 0.8, 0.4, 0.8))
values_actual <- r12_errors$FUN_true
values_top <- r12_errors$upper_error
values_bottom <- r12_errors$lower_error
values_max = r12_errors$max_abs_error

plot(values_top ~ r12_errors$FUN_true, type = "l", ylim = c(0,
    10), xlim = c(-300, -200), lwd = 1)
points(values_bottom ~ r12_errors$FUN_true, type = "l",
    lwd = 1)
limits <- c(5, 439)
# polygon(c(values_actual[limits[1]:limits[2]],
# rev(values_actual[limits[1]:limits[2]])), y =
# c(values_max[limits[1]:limits[2]], rev(rep(0,
# length(limits[1]:limits[2])))), col = rgb(1, 0, 0,
# alpha = 1), density = 50, lty = 'dotted')
legend("top", legend = c("R+3%", "R-3%", "Possible values"),
    col = c("black", "black", rgb(1, 0, 0, alpha = 1)),
    lty = c(1, 1, 5))
```




```
# Define coolwarm color gradient
coolwarm <- colorRampPalette(c(rgb(60, 81, 198, maxColorValue = 255),
    rgb(61, 86, 203, maxColorValue = 255), rgb(63, 91, 207,
        maxColorValue = 255), rgb(65, 96, 212, maxColorValue = 255),
    rgb(67, 101, 216, maxColorValue = 255), rgb(69, 106,
        220, maxColorValue = 255), rgb(71, 111, 224, maxColorValue = 255),
    rgb(74, 116, 227, maxColorValue = 255), rgb(76, 121,
        231, maxColorValue = 255), rgb(79, 127, 233, maxColorValue = 255),
    rgb(83, 132, 236, maxColorValue = 255), rgb(86, 137,
        238, maxColorValue = 255), rgb(90, 143, 240, maxColorValue = 255),
    rgb(94, 148, 242, maxColorValue = 255), rgb(99, 153,
        243, maxColorValue = 255), rgb(103, 159, 244, maxColorValue = 255),
    rgb(109, 164, 244, maxColorValue = 255), rgb(114, 169,
        245, maxColorValue = 255), rgb(120, 174, 245, maxColorValue = 255),
    rgb(126, 179, 245, maxColorValue = 255), rgb(132, 184,
        244, maxColorValue = 255), rgb(139, 188, 243, maxColorValue = 255),
    rgb(146, 193, 242, maxColorValue = 255), rgb(153, 197,
        241, maxColorValue = 255), rgb(161, 201, 239, maxColorValue = 255),
    rgb(169, 205, 238, maxColorValue = 255), rgb(177, 209,
        236, maxColorValue = 255), rgb(186, 212, 233, maxColorValue = 255),
    rgb(195, 215, 231, maxColorValue = 255), rgb(204, 218,
        229, maxColorValue = 255), rgb(214, 221, 226, maxColorValue = 255),
    rgb(223, 223, 223, maxColorValue = 255), rgb(235, 218,
```

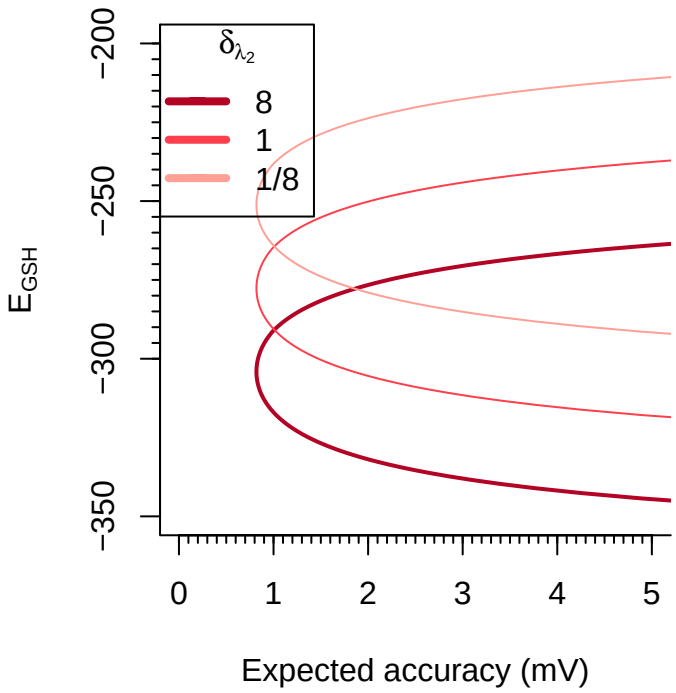
```
        215, maxColorValue = 255), rgb(245, 213, 207, maxColorValue = 255),
    rgb(255, 206, 198, maxColorValue = 255), rgb(255, 192,
        184, maxColorValue = 255), rgb(255, 180, 170, maxColorValue = 255),
    rgb(255, 168, 159, maxColorValue = 255), rgb(255, 157,
        148, maxColorValue = 255), rgb(255, 147, 139, maxColorValue = 255),
    rgb(255, 138, 130, maxColorValue = 255), rgb(255, 129,
        122, maxColorValue = 255), rgb(255, 121, 115, maxColorValue = 255),
    rgb(255, 113, 109, maxColorValue = 255), rgb(255, 105,
        103, maxColorValue = 255), rgb(255, 98, 98, maxColorValue = 255),
    rgb(255, 91, 93, maxColorValue = 255), rgb(255, 85,
        89, maxColorValue = 255), rgb(255, 78, 85, maxColorValue = 255),
    rgb(255, 72, 81, maxColorValue = 255), rgb(255, 67,
        78, maxColorValue = 255), rgb(255, 61, 75, maxColorValue = 255),
    rgb(255, 56, 72, maxColorValue = 255), rgb(255, 50,
        70, maxColorValue = 255), rgb(255, 45, 67, maxColorValue = 255),
    rgb(255, 41, 65, maxColorValue = 255), rgb(252, 35,
        62, maxColorValue = 255), rgb(242, 30, 58, maxColorValue = 255),
    rgb(233, 24, 55, maxColorValue = 255), rgb(223, 20,
        51, maxColorValue = 255), rgb(212, 15, 48, maxColorValue = 255),
    rgb(202, 11, 44, maxColorValue = 255), rgb(191, 7, 41,
        maxColorValue = 255), rgb(180, 4, 38, maxColorValue = 255)))
```

```
par(pty = "s", mfrow = c(1, 1))
colors <- rev(coolwarm(6))

r12_errors <- getErrorTable(roGFP1_R12_empirical_sensor,
    R = getR(roGFP1_R12_empirical_sensor), Error_Model = function(x) {
        return(x * 0.02)
    }, FUN = getE)
plot(r12_errors$FUN_true ~ r12_errors$max_abs_error, type = "l",
    xlim = c(0, 5), ylim = c(-300, -200), col = colors[1],
    lwd = 2, bty = "l", xlab = "Expected accuracy (mV)",
    ylab = expression(E[GSH]), main = "")
r12_errors <- getErrorTable(roGFP1_R12_empirical_sensor,
    R = getR(roGFP1_R12_empirical_sensor), Error_Model = function(x) {
        return(x * 0.04)
    }, FUN = getE)
points(r12_errors$FUN_true ~ r12_errors$max_abs_error, type = "l",
    col = colors[2])
r12_errors <- getErrorTable(roGFP1_R12_empirical_sensor,
    R = getR(roGFP1_R12_empirical_sensor), Error_Model = function(x) {
        return(x * 0.08)
    }, FUN = getE)
points(r12_errors$FUN_true ~ r12_errors$max_abs_error, type = "l",
    col = colors[3])

axis(side = 1, at = seq(0, 5, by = 0.1), labels = FALSE,
    tcl = -0.2)
axis(side = 2, at = seq(-300, -200, by = 5), labels = FALSE,
    tcl = -0.2)

legend("topleft", title = "Empircal precision in R", xpd = TRUE,
    c("2%", "4%", "8%"), pch = "-", lwd = 4, col = c(colors[1:3]),
    cex = 1)
```



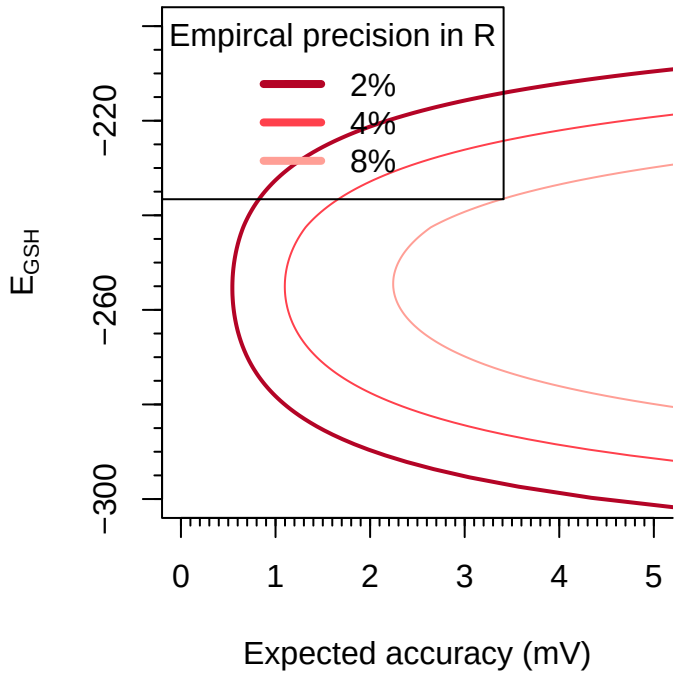

```
par(pty = "s", mfrow = c(1, 1))
colors <- rev(coolwarm(6))

roGFP1_R12_empirical_sensor@delta = 8
r12_errors <- getErrorTable(roGFP1_R12_empirical_sensor,
    R = getR(roGFP1_R12_empirical_sensor), Error_Model = function(x) {
        return(x * 0.03)
    }, FUN = getE)
plot(r12_errors$FUN_true ~ r12_errors$max_abs_error, type = "l",
    xlim = c(0, 5), ylim = c(-350, -200), col = colors[1],
    lwd = 2, bty = "l", xlab = "Expected accuracy (mV)",
    ylab = expression(E[GSH]), main = "")
roGFP1_R12_empirical_sensor@delta = 1
r12_errors <- getErrorTable(roGFP1_R12_empirical_sensor,
    R = getR(roGFP1_R12_empirical_sensor), Error_Model = function(x) {
        return(x * 0.03)
    }, FUN = getE)
points(r12_errors$FUN_true ~ r12_errors$max_abs_error, type = "l",
    col = colors[2])
roGFP1_R12_empirical_sensor@delta = 0.125
r12_errors <- getErrorTable(roGFP1_R12_empirical_sensor,
    R = getR(roGFP1_R12_empirical_sensor), Error_Model = function(x) {
        return(x * 0.03)
    }, FUN = getE)
points(r12_errors$FUN_true ~ r12_errors$max_abs_error, type = "l",
```

```
    col = colors[3])
roGFP1_R12_empirical_sensor@delta = 0.171

axis(side = 1, at = seq(0, 5, by = 0.1), labels = FALSE,
    tcl = -0.2)
axis(side = 2, at = seq(-300, -200, by = 5), labels = FALSE,
    tcl = -0.2)

legend("topleft", title = expression(delta[lambda[2]]),
    xpd = TRUE, c("8", "1", "1/8"), pch = "-", lwd = 4,
    col = c(colors[1:3]), cex = 1)
```



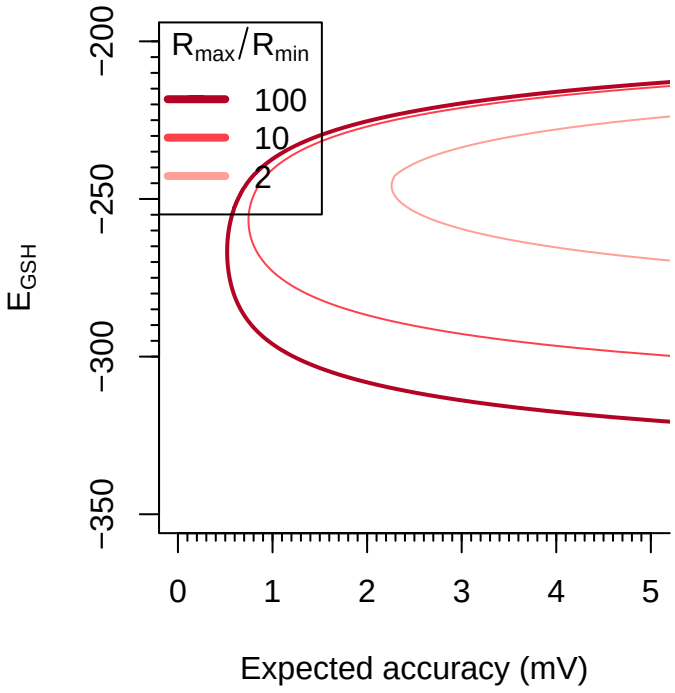

```
par(pty = "s", mfrow = c(1, 1))
colors <- rev(coolwarm(6))

r12_errors <- getErrorTable(roGFP1_R12_empirical_sensor,
    R = getR(roGFP1_R12_empirical_sensor), Error_Model = function(x) {
        return(x * 0.02)
    }, FUN = getE)
plot(r12_errors$FUN_true ~ r12_errors$max_abs_error, type = "l",
    xlim = c(0, 5), ylim = c(-300, -200), col = colors[1],
    lwd = 2, bty = "l", xlab = "Expected accuracy (mV)",
    ylab = expression(E[GSH]), main = "")
r12_errors <- getErrorTable(roGFP1_R12_empirical_sensor,
    R = getR(roGFP1_R12_empirical_sensor), Error_Model = function(x) {
        return(x * 0.04)
```

```
    }, FUN = getE)
points(r12_errors$FUN_true ~ r12_errors$max_abs_error, type = "l",
    col = colors[2])
r12_errors <- getErrorTable(roGFP1_R12_empirical_sensor,
    R = getR(roGFP1_R12_empirical_sensor), Error_Model = function(x) {
        return(x * 0.08)
    }, FUN = getE)
points(r12_errors$FUN_true ~ r12_errors$max_abs_error, type = "l",
    col = colors[3])

axis(side = 1, at = seq(0, 5, by = 0.1), labels = FALSE,
    tcl = -0.2)
axis(side = 2, at = seq(-300, -200, by = 5), labels = FALSE,
    tcl = -0.2)

legend("topleft", title = "Empircal precision in R", xpd = TRUE,
    c("2%", "4%", "8%"), pch = "-", lwd = 4, col = c(colors[1:3]),
    cex = 1)
```




```
par(pty = "s", mfrow = c(1, 1))
colors <- rev(coolwarm(6))

# Delta 1 of 8
roGFP1_R12_empirical_sensor@Rmax = 46.78
roGFP1_R12_empirical_sensor@Rmin = 1
r12_errors <- getErrorTable(roGFP1_R12_empirical_sensor,
```

```
    R = getR(roGFP1_R12_empirical_sensor), Error_Model = function(x) {
        return(x * 0.03)
    }, FUN = getE)
plot(r12_errors$FUN_true ~ r12_errors$max_abs_error, type = "l",
    xlim = c(0, 5), ylim = c(-350, -200), col = colors[1],
    lwd = 2, bty = "l", xlab = "Expected accuracy (mV)",
    ylab = expression(E[GSH]), main = "")

roGFP1_R12_empirical_sensor@Rmax = 10
roGFP1_R12_empirical_sensor@Rmin = 1
r12_errors <- getErrorTable(roGFP1_R12_empirical_sensor,
    R = getR(roGFP1_R12_empirical_sensor), Error_Model = function(x) {
        return(x * 0.03)
    }, FUN = getE)
points(r12_errors$FUN_true ~ r12_errors$max_abs_error, type = "l",
    col = colors[2])
roGFP1_R12_empirical_sensor@Rmax = 2
roGFP1_R12_empirical_sensor@Rmin = 1
r12_errors <- getErrorTable(roGFP1_R12_empirical_sensor,
    R = getR(roGFP1_R12_empirical_sensor), Error_Model = function(x) {
        return(x * 0.03)
    }, FUN = getE)
points(r12_errors$FUN_true ~ r12_errors$max_abs_error, type = "l",
    col = colors[3])
roGFP1_R12_empirical_sensor@Rmax = 5.207
roGFP1_R12_empirical_sensor@Rmin = 0.667

axis(side = 1, at = seq(0, 5, by = 0.1), labels = FALSE,
    tcl = -0.2)
axis(side = 2, at = seq(-300, -200, by = 5), labels = FALSE,
    tcl = -0.2)

legend("topleft", title = expression(R[max]/R[min]), xpd = TRUE,
    c("100", "10", "2"), pch = "-", lwd = 4, col = c(colors[1:3]),
    cex = 1)
```

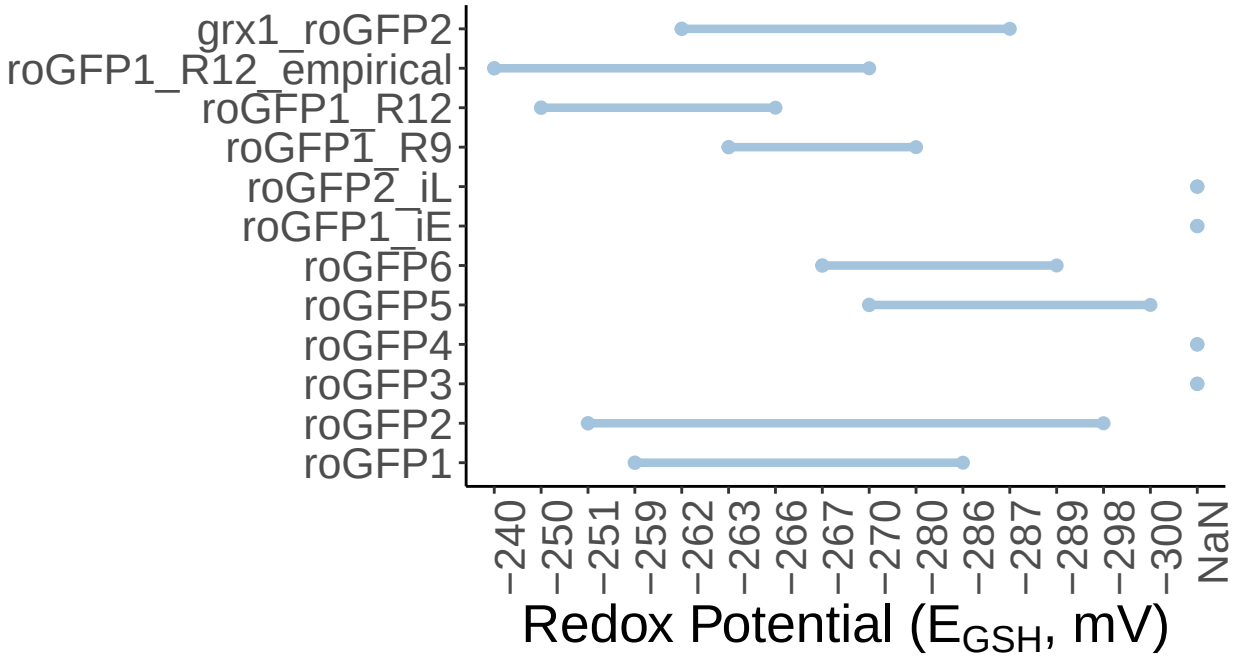



```
# error_model <- function(x) {return(0.025*x)} value <-
# -270 plot(R12_error$FUN_true ~
# R12_error$max_abs_error, type = 'l', ylim = c(value-1,
# value+1), xlim = c(0, 10)) for (sensorName in
# sensorList) { sensor <- get(sensorName) sensorName <-
# str_replace(sensorName, '_sensor', '') error_data <-
# getErrorTable(sensor, R = getR(sensor), FUN = getE,
# Error_Model = error_model) points(error_data$FUN_true
# ~ error_data$max_abs_error, type = 'l')
# print(sensorName) print(subset(error_data,
# abs(error_data$FUN_true - value) <
# 0.1)$max_abs_error[1]) }
```

For figures

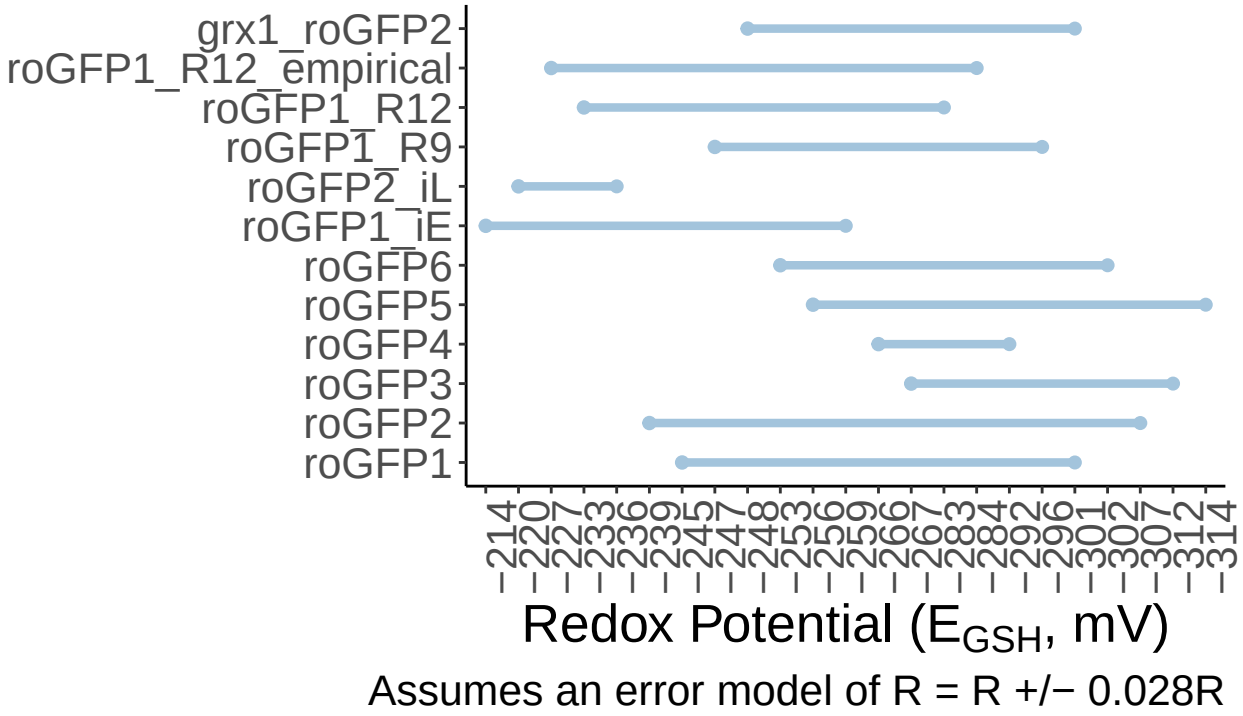
```
# For figures!
acceptable_error <- 1
error_model <- function(x) {return(0.028*x)}

minMaxMatrix <- c()

for (sensorName in sensorList) {
    sensor <- get(sensorName)
    sensorName <- str_replace(sensorName, "_sensor", "")
    error_data <- getErrorTable(sensor, R = getR(sensor),
```

```
                                FUN = getE, Error_Model = error_model)

    error_filter <- subset(error_data,
                           error_data$max_abs_error < acceptable_error)

    minimum <- ifelse(test = length(error_filter$FUN_true) == 0,
           yes = NaN, no = min(error_filter$FUN_true))

    maximum <- ifelse(test = length(error_filter$FUN_true) == 0,
           yes = NaN, no = max(error_filter$FUN_true))

    minMaxMatrix <- rbind(minMaxMatrix, c(sensorName, round(minimum, 0), round(maximum,0)))
}

ranges <- data.frame(minMaxMatrix)
colnames(ranges) <- c("Sensor_Name", "Minimum", "Maximum")

theme_set(theme_classic())

ranges$Sensor_Name <- factor(ranges$Sensor_Name,
                             levels=as.character(ranges$Sensor_Name))

gg <- ggplot(ranges, aes(x=Minimum, xend=Maximum,
                         y = Sensor_Name, group=Sensor_Name)) +
        geom_dumbbell(color="#a3c4dc",
                      size=1.5) +
           labs(x = expression("Redox Potential (" * E[GSH] * ", mV" * ")"),
             y=NULL,
             title="",
             caption="Assumes an error model of R = R +/- 0.028R",
             subtitle="") +
    theme(plot.title = element_text(hjust=0.5, face="bold"),
              #plot.background=element_rect(fill="#f7f7f7"),
              #panel.background=element_rect(fill="#f7f7f7"),
              panel.grid.minor=element_blank(),
              panel.grid.major.y=element_blank(),
              legend.position="top",
              panel.border=element_blank(),
          text = element_text(size = 20),
          axis.text.x = element_text(angle = 90, hjust = 1))
plot(gg)
```

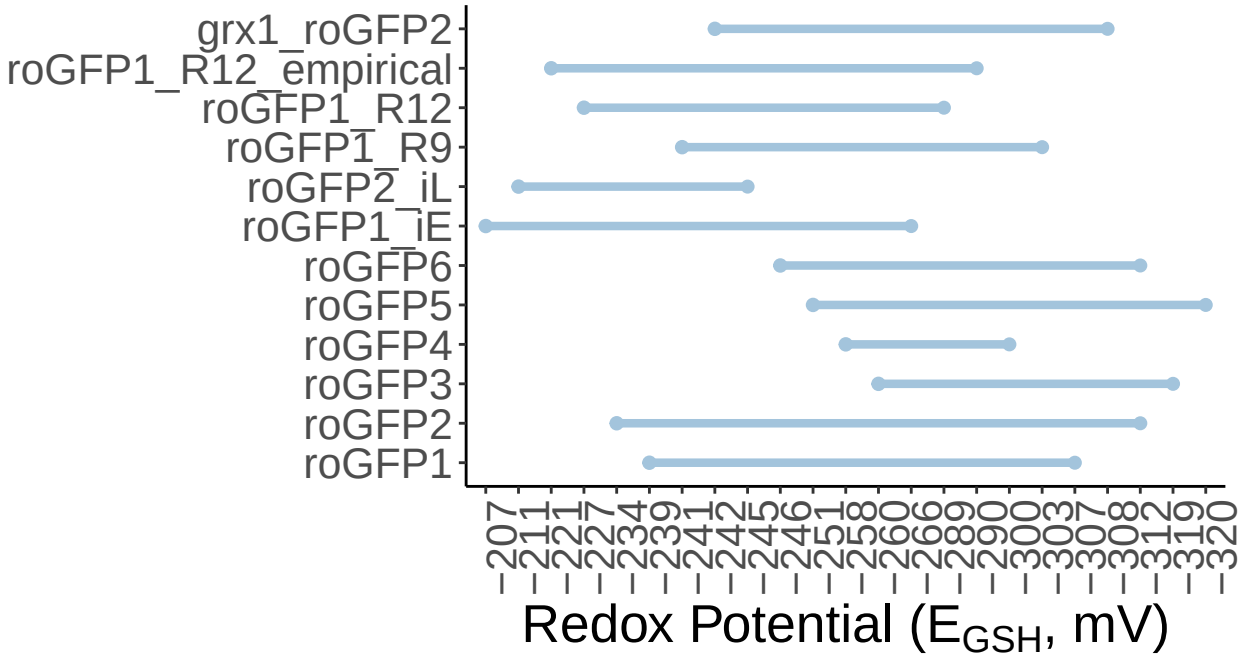



```
acceptable_error <- 2
error_model <- function(x) {return(0.028*x)}

minMaxMatrix <- c()

for (sensorName in sensorList) {
    sensor <- get(sensorName)
    sensorName <- str_replace(sensorName, "_sensor", "")
    error_data <- getErrorTable(sensor, R = getR(sensor),
                                FUN = getE, Error_Model = error_model)

    error_filter <- subset(error_data,
                           error_data$max_abs_error < acceptable_error)

    minimum <- ifelse(test = length(error_filter$FUN_true) == 0,
           yes = NaN, no = min(error_filter$FUN_true))

    maximum <- ifelse(test = length(error_filter$FUN_true) == 0,
           yes = NaN, no = max(error_filter$FUN_true))

    minMaxMatrix <- rbind(minMaxMatrix, c(sensorName, round(minimum, 0), round(maximum,0)))
}

ranges <- data.frame(minMaxMatrix)
colnames(ranges) <- c("Sensor_Name", "Minimum", "Maximum")
```

```r
theme_set(theme_classic())

ranges$Sensor_Name <- factor(ranges$Sensor_Name,
                             levels=as.character(ranges$Sensor_Name))

gg <- ggplot(ranges, aes(x=Minimum, xend=Maximum,
                         y = Sensor_Name, group=Sensor_Name)) +
        geom_dumbbell(color="#a3c4dc",
                      size=1.5) +
           labs(x = expression("Redox Potential (" * E[GSH] * ", mV" * ")"),
             y=NULL,
             title="",
             caption="Assumes an error model of R = R +/- 0.028R",
             subtitle="") +
    theme(plot.title = element_text(hjust=0.5, face="bold"),
              #plot.background=element_rect(fill="#f7f7f7"),
              #panel.background=element_rect(fill="#f7f7f7"),
              panel.grid.minor=element_blank(),
              panel.grid.major.y=element_blank(),
              legend.position="top",
              panel.border=element_blank(),
          text = element_text(size = 20),
          axis.text.x = element_text(angle = 90, hjust = 1))
plot(gg)
```



```r
acceptable_error <- 3
error_model <- function(x) {return(0.028*x)}

minMaxMatrix <- c()

for (sensorName in sensorList) {
    sensor <- get(sensorName)
    sensorName <- str_replace(sensorName, "_sensor", "")
    error_data <- getErrorTable(sensor, R = getR(sensor),
                                FUN = getE, Error_Model = error_model)

    error_filter <- subset(error_data,
                           error_data$max_abs_error < acceptable_error)

    minimum <- ifelse(test = length(error_filter$FUN_true) == 0,
           yes = NaN, no = min(error_filter$FUN_true))

    maximum <- ifelse(test = length(error_filter$FUN_true) == 0,
           yes = NaN, no = max(error_filter$FUN_true))

    minMaxMatrix <- rbind(minMaxMatrix, c(sensorName, round(minimum, 0), round(maximum,0)))
}

ranges <- data.frame(minMaxMatrix)
colnames(ranges) <- c("Sensor_Name", "Minimum", "Maximum")

theme_set(theme_classic())

ranges$Sensor_Name <- factor(ranges$Sensor_Name,
                             levels=as.character(ranges$Sensor_Name))

gg <- ggplot(ranges, aes(x=Minimum, xend=Maximum,
                         y = Sensor_Name, group=Sensor_Name)) +
        geom_dumbbell(color="#a3c4dc",
                      size=1.5) +
           labs(x = expression("Redox Potential (" * E[GSH] * ", mV" * ")"),
             y=NULL,
             title="",
             caption="Assumes an error model of R = R +/- 0.028R",
             subtitle="") +
    theme(plot.title = element_text(hjust=0.5, face="bold"),
              #plot.background=element_rect(fill="#f7f7f7"),
              #panel.background=element_rect(fill="#f7f7f7"),
              panel.grid.minor=element_blank(),
              panel.grid.major.y=element_blank(),
              legend.position="top",
              panel.border=element_blank(),
          text = element_text(size = 20),
          axis.text.x = element_text(angle = 90, hjust = 1))
plot(gg)
```




```
theme_pub <- function(base_size = 14, base_family = "helvetica") {
    library(grid)
    library(ggthemes)
    (theme_foundation(base_size = base_size, base_family = base_family) +
        theme(plot.title = element_text(face = "bold", size = rel(1.2),
            hjust = 0.5), text = element_text(), panel.background = element_rect(colour = NA),
            plot.background = element_rect(colour = NA),
            panel.border = element_rect(colour = NA), axis.title = element_text(face = "bold",
                size = rel(1)), axis.title.y = element_text(angle = 90,
                vjust = 2), axis.title.x = element_text(vjust = -0.2),
            axis.text = element_text(), axis.line = element_line(colour = "black"),
            axis.ticks = element_line(), panel.grid.major = element_line(colour = "#f0f0f0"),
            panel.grid.minor = element_blank(), legend.key = element_rect(colour = NA),
            legend.position = "bottom", legend.direction = "horizontal",
            legend.key.size = unit(0.2, "cm"), legend.margin = unit(0,
                "cm"), legend.title = element_text(face = "italic"),
            plot.margin = unit(c(10, 5, 5, 5), "mm"), strip.background = element_rect(colour = "#f0f0f0
                fill = "#f0f0f0"), strip.text = element_text(face = "bold")))

}

scale_fill_Publication <- function(...) {
    library(scales)
    discrete_scale("fill", "Publication", manual_pal(values = c("#386cb0",
        "#fdb462", "#7fc97f", "#ef3b2c", "#662506", "#a6cee3",
        "#fb9a99", "#984ea3", "#ffff33")), ...)
```

```
}

scale_colour_Publication <- function(...) {
    library(scales)
    discrete_scale("colour", "Publication", manual_pal(values = c("#386cb0",
        "#fdb462", "#7fc97f", "#ef3b2c", "#662506", "#a6cee3",
        "#fb9a99", "#984ea3", "#ffff33")), ...)

}
```

```
# generateMinMax <- function(error_model, error_cutoffs,
# sensor_list) { minMaxMatrix <- data.frame(Sensor_Name
# = c(), Minimum = c(), Maximum = c(), error_thresh =
# c()) for(acceptable_error in error_cutoffs) { for
# (sensor_name in sensor_list) { sensor <-
# get(sensor_name) sensor_name <-
# str_replace(sensor_name, '_sensor', '') error_data <-
# getErrorTable(sensor, R = getR(sensor), FUN = getE,
# Error_Model = error_model) error_filter <-
# subset(error_data, error_data$max_abs_error <
# acceptable_error) minimum <- ifelse(test =
# length(error_filter$FUN_true) == 0, yes = NaN, no =
# min(error_filter$FUN_true)) maximum <- ifelse(test =
# length(error_filter$FUN_true) == 0, yes = NaN, no =
# max(error_filter$FUN_true)) minMaxMatrix <-
# rbind(minMaxMatrix, data.frame(Sensor_Name =
# sensor_name, Minimum = round(minimum, 0), Maximum =
# round(maximum,0), acceptable_error =
# as.numeric(acceptable_error))) } } ranges <-
# minMaxMatrix colnames(ranges) <- c('Sensor_Name',
# 'Minimum', 'Maximum', 'error_thresh') return(ranges) }
# error_model <- function(x) {return(0.028*x)} ranges <-
# generateMinMax(error_model, c(1, 1.5, 2, 2.5, 3.0),
# sensorList) insert_minor <- function(major_labs,
# n_minor) {labs <- c( sapply( major_labs, function(x)
# c(x, rep('', 4) ) ) ) labs[1:(length(labs)-n_minor)]}
# mc_tribble <- function(indf, indents = 4, mdformat =
# TRUE) { name <- as.character(substitute(indf)) name <-
# name[length(name)] meat <-
# capture.output(write.csv(indf, quote = TRUE, row.names
# = FALSE)) meat <- paste0( paste(rep(' ', indents),
# collapse = ''), c(paste(sprintf('~%s', names(indf)),
# collapse = ', '), meat[-1])) if (mdformat) meat <-
# paste0(' ', meat) obj <- paste(name, ' <-
# tribble(\n', paste(meat, collapse = ',\n'), ')', sep
# = '') if (mdformat) cat(paste0(' ', obj)) else
# cat(obj) } mc_tribble(ranges) ranges <- tribble(
# ~Sensor_Name, ~Minimum, ~Maximum, ~error_thresh,
# 'roGFP1',-286,-259,1, 'roGFP2',-298,-251,1,
# 'roGFP3',NA,NA,1, 'roGFP4',NA,NA,1,
# 'roGFP5',-300,-270,1, 'roGFP6',-289,-267,1,
# 'roGFP1_iE',NA,NA,1, 'roGFP2_iL',NA,NA,1,
# 'roGFP1_R9',-280,-263,1, 'roGFP1_R12',-266,-250,1,
```

```
# 'roGFP1_R12_empirical',-270,-240,1,
# 'grx1_roGFP2',-287,-262,1, 'roGFP1',-296,-250,1.5,
# 'roGFP2',-303,-244,1.5, 'roGFP3',-307,-272,1.5,
# 'roGFP4',NA,NA,1.5, 'roGFP5',-309,-261,1.5,
# 'roGFP6',-297,-258,1.5, 'roGFP1_iE',-253,-219,1.5,
# 'roGFP2_iL',NA,NA,1.5, 'roGFP1_R9',-291,-252,1.5,
# 'roGFP1_R12',-278,-239,1.5,
# 'roGFP1_R12_empirical',-279,-231,1.5,
# 'grx1_roGFP2',-297,-253,1.5, 'roGFP1',-301,-245,2,
# 'roGFP2',-307,-239,2, 'roGFP3',-312,-267,2,
# 'roGFP4',-292,-266,2, 'roGFP5',-314,-256,2,
# 'roGFP6',-302,-253,2, 'roGFP1_iE',-259,-214,2,
# 'roGFP2_iL',-236,-220,2, 'roGFP1_R9',-296,-247,2,
# 'roGFP1_R12',-283,-233,2,
# 'roGFP1_R12_empirical',-284,-227,2,
# 'grx1_roGFP2',-301,-248,2, 'roGFP1',-304,-241,2.5,
# 'roGFP2',-312,-236,2.5, 'roGFP3',-316,-263,2.5,
# 'roGFP4',-296,-261,2.5, 'roGFP5',-317,-253,2.5,
# 'roGFP6',-302,-248,2.5, 'roGFP1_iE',-263,-210,2.5,
# 'roGFP2_iL',-242,-213,2.5, 'roGFP1_R9',-300,-243,2.5,
# 'roGFP1_R12',-287,-230,2.5,
# 'roGFP1_R12_empirical',-287,-223,2.5,
# 'grx1_roGFP2',-305,-245,2.5, 'roGFP1',-307,-239,3,
# 'roGFP2',-312,-234,3, 'roGFP3',-319,-260,3,
# 'roGFP4',-300,-258,3, 'roGFP5',-320,-251,3,
# 'roGFP6',-312,-246,3, 'roGFP1_iE',-266,-207,3,
# 'roGFP2_iL',-245,-211,3, 'roGFP1_R9',-303,-241,3,
# 'roGFP1_R12',-289,-227,3,
# 'roGFP1_R12_empirical',-290,-221,3,
# 'grx1_roGFP2',-308,-242,3) ggplot() +
# geom_linerange(data = ranges %>%
# arrange(-error_thresh), mapping=aes(x = Sensor_Name,
# ymin = Minimum, ymax = Maximum, lwd = 1, color =
# error_thresh), size = 10) + coord_cartesian(ylim =
# c(-240, -300)) + scale_color_continuous(high =
# 'lightgreen', low = 'forestgreen') + xlab('')+
# ylab('Glutathione Redox Potential (mV)') +
# coord_flip() + theme_classic() + theme(aspect.ratio =
# 1) ggplot() + geom_linerange(data = ranges %>%
# arrange(-as.numeric(error_thresh)), mapping=aes(x =
# Sensor_Name, ymin = Minimum, ymax = Maximum, lwd = 1,
# color = error_thresh)) + scale_color_manual(values =
# rev(brewer.pal(5,'Greens'))) + ylab('Redox Potential')
# + xlab('') + scale_y_continuous(breaks=
# seq(-400,-200,by = 10), labels = insert_minor(
# seq(-400, -200, by = 50), 4 ), limits = c(-350,-200),
# expand = c(0,0)) + coord_flip()
```