On probabilities plots and (R) polygons

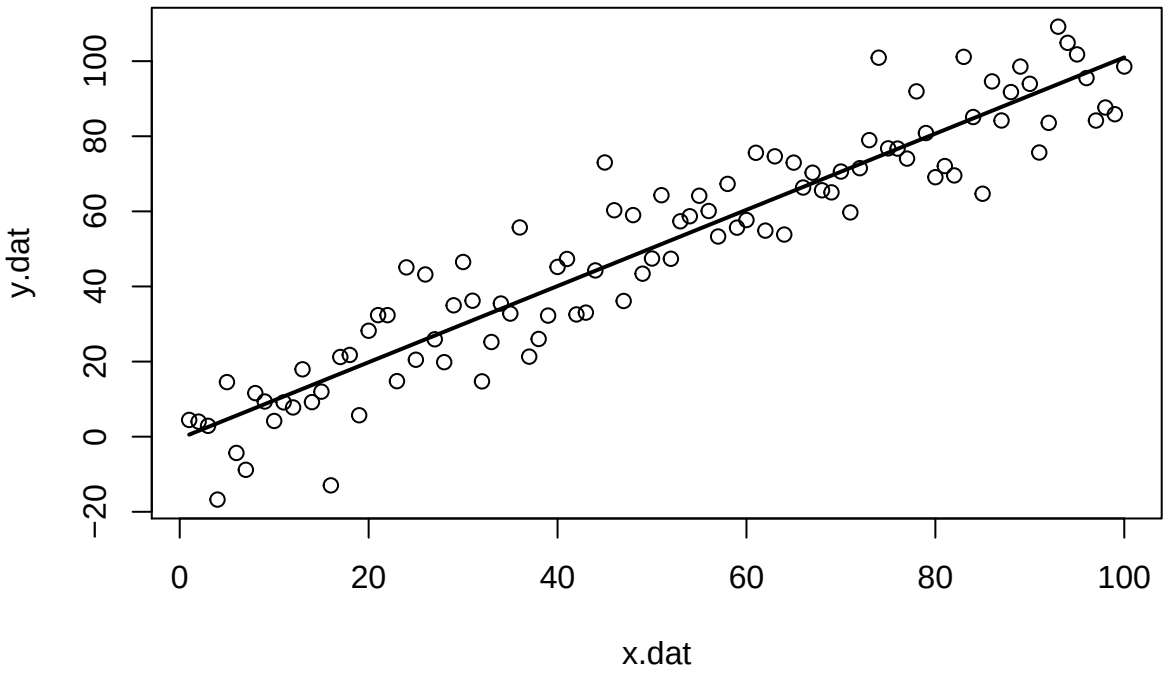
Silas Tittes

June 15, 2015

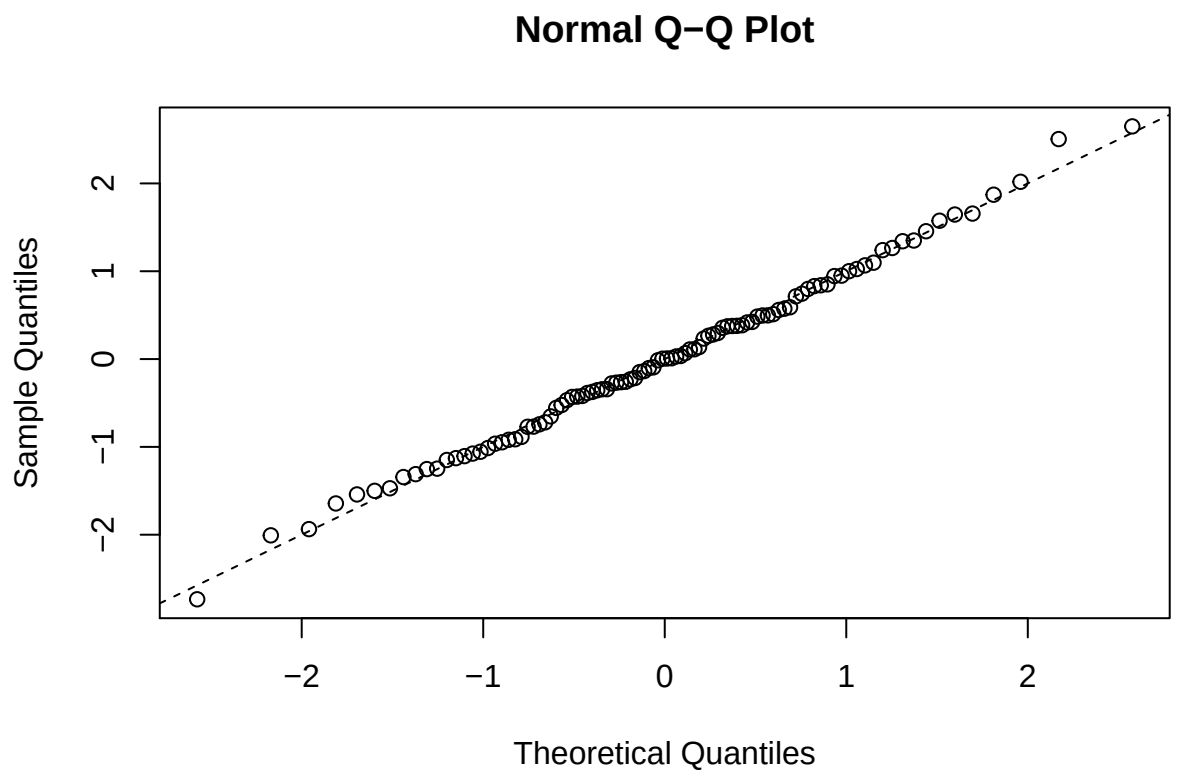
What's a probability plot?

I was introduced to quantile-quantile plots (aka q-q plots, which are very similar or even the same as probability plots) as an undergrad in a course called, Biometry. Famous to CU Boulder's EBIO students, Biometry has been taught for many years (decades even?) by the most excellent, Doktor Michael Grant (aka, Hey You). The course introduced me to the use of quantile-quantile plots for testing the assumption of normally distributed residuals when fitting your standard, everyday pair of blue jeans, linear model. R has a nice function called qqnorm that does all the work for you. Here's a brief demo:

```
################## set up a model #
n.size <- 100
x.dat <- 1:n.size  # a fake explanatory variable
err <- rnorm(n = n.size, mean = 0,
    sd = 10)  #normally distributed error
y.dat <- x.dat + err  # a fake predictor variable with error
model <- lm(y.dat ~ x.dat)  # fit linear model
plot(x.dat, y.dat)  #plot
mod.coef <- coef(model)  #store parameter estimates
curve(mod.coef[1] + mod.coef[2] *
    x, lwd = 2, add = T)  #leave your abline at home
```




```
################## the q-q plot ##
qqnorm(scale(resid(model)))  #scale = values / standard deviation (values)
abline(0, 1, lty = 2)  #there's abline, are you happy now?
```



The basic idea is that if have an assumption about how your values of interest are distributed, you can visualize (and even quantify) how well that assumption is met by comparing the

ranked (and sometimes scaled) values to the quantiles of the distribution you believe the values were drawn from. If there is a near one-to-one correspondence between the two, you're in the clear. It's also a good way to see how well a transform corrects for skew, heteroscedasticity, and the likes. By the way, quantile is a fancy word that more or less means a set a values that break a cumulative distribution into regular intervals (i.e. quantiles are the values that break off the first 10%, 20%, 30%, ..., 90% of the density of a distribution). Naturally, distributions have their densities distributed differently over the set of values for which they are defined, so they will have different quantiles that regularly break their densities.

While a cool idea, I've always been a little dissatisfied with the q-q plot. How can an assumption check of a statistical test be so, un-statistical? Perhaps one argument is, if you can test the validity of a test, why not test the validity of the test you used to validate the original test of interest? Like an infinite mirror illusion, we could keep doing tests until the resolution of the reflection became too faint or too far into the margins of our vantage point. Sounds like a waste of time. Nonetheless, the assumption check could be a little more satisfying, and what better way to satisfy than using simulations?!

To add to the probability plot, lets compare the quantiles of a distribution to many random draws from that distribution, and then see if and how our values of interest depart from the overall patterns we observe in the plot. This is a quick way to better understand the expected degree of random departures in a way that can't be captured using the R's standard qqnorm function.

Working with probability distributions is made ridiculously easy in R. For example, if you want 100 draws from a normal distribution with a mean of 10 and a standard deviation of 5, simple type, "rnorm(100, 10, 5)". Bam. For every distribution native to R, there are 4 utilities: d, r, p, q. - density, random, probability, and quantile. We will concern ourselves with the random and quantile utilities.

Enough English, lets take a look at the function:

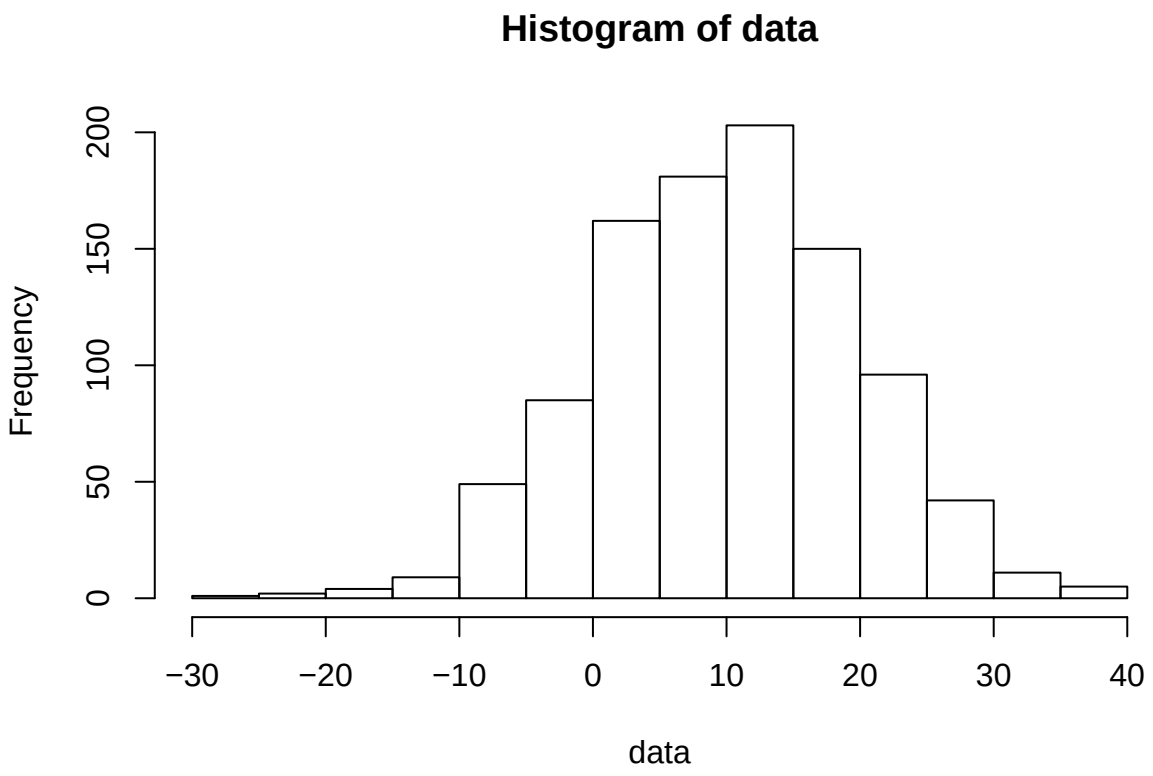
```
probPlotNorm <- function(values,
    reps, ...) {

    # get mean and standard
    # deviation from values
    mt <- mean(values)
    msd <- sd(values)

    # get quantiles of normal dist
    # with mean = mt and sd = msd
    probs <- ppoints(n = length(values))
    quantNorm <- qnorm(p = probs,
        mean = mt, sd = msd)
```

```
# draw reps*length(values)
# random values from an normal
# with mt and msd
rvalues <- rnorm(n = reps *
    length(values), mean = mt,
    sd = msd)
# store in a matrix with reps
# number of columns and
# length(values) number of rows
normMat <- matrix(rvalues,
    nrow = reps, ncol = length(values),
    byrow = T)
# sort within each row
sortedMat <- apply(normMat,
    1, sort)
# get the min and max for each
# row
rangeMat <- apply(sortedMat,
    1, range)

# make an empty plot with the
# correct dimensions
xxlab <- paste("Quanatiles of Gaussian, with mean of ",
    round(mt, 1), "\nand standard deviation of ",
    round(msd, 1), sep = "")
yylab <- paste(reps, " random draws from Gaussian,\nand some data",
    sep = "")

# adjust plot margins
par(mar = c(5, 5, 0, 2))

plot(quantNorm, sort(normMat[1,
    ]), type = "n", ylim = c(range(normMat)[1],
    range(normMat)[2]), xlab = xxlab,
    ylab = yylab)
# make the polygoon. notice the
# concatenation of the x axis
# twice, second time reversed.
# notice this too for the y
# axis.
polygon(x = c(quantNorm, rev(quantNorm)),
    y = c(rangeMat[1, ], rev(rangeMat[2,
        ])), lwd = 2, col = "grey80")
```

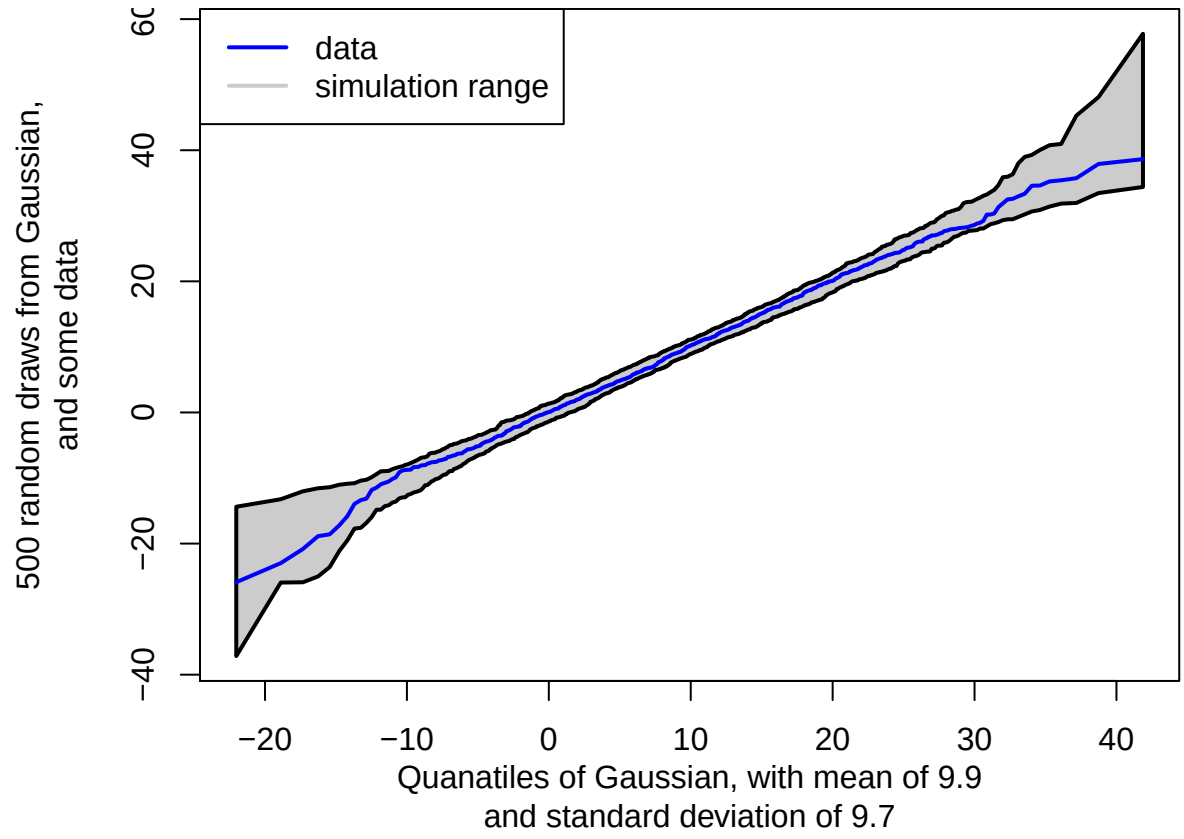
```
    # plot the ranked data over the
    # polygon
    lines(quantNorm, sort(values),
        lwd = 2, col = "blue")

    legend("topleft", c("data",
        "simulation range"), lty = 1,
        lwd = c(2, 2), col = c("blue",
            "grey80"))

    # reset plot margins
    par(mar = c(5, 4, 4, 2) * 1.1)

    # return quantiles of normal
    # dist with estimated
    # parameters from data
    return(list(quantNorm = quantNorm))

}
```

Example of calling the function

```
sim.n <- 1000
data <- rnorm(n = sim.n, mean = 10,
    sd = 10)
# data <- rexp(sim.n, rate =
# 1/10)
hist(data)
```

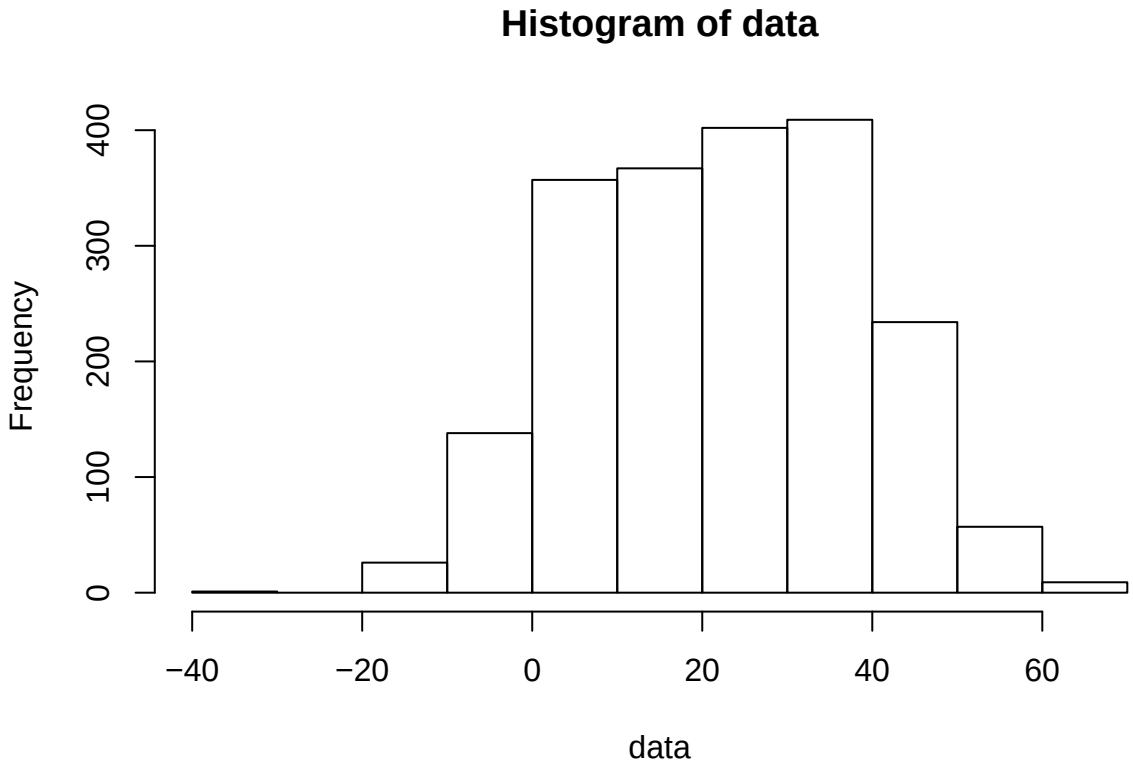



```
example <- probPlotNorm(data, reps = 500)
```

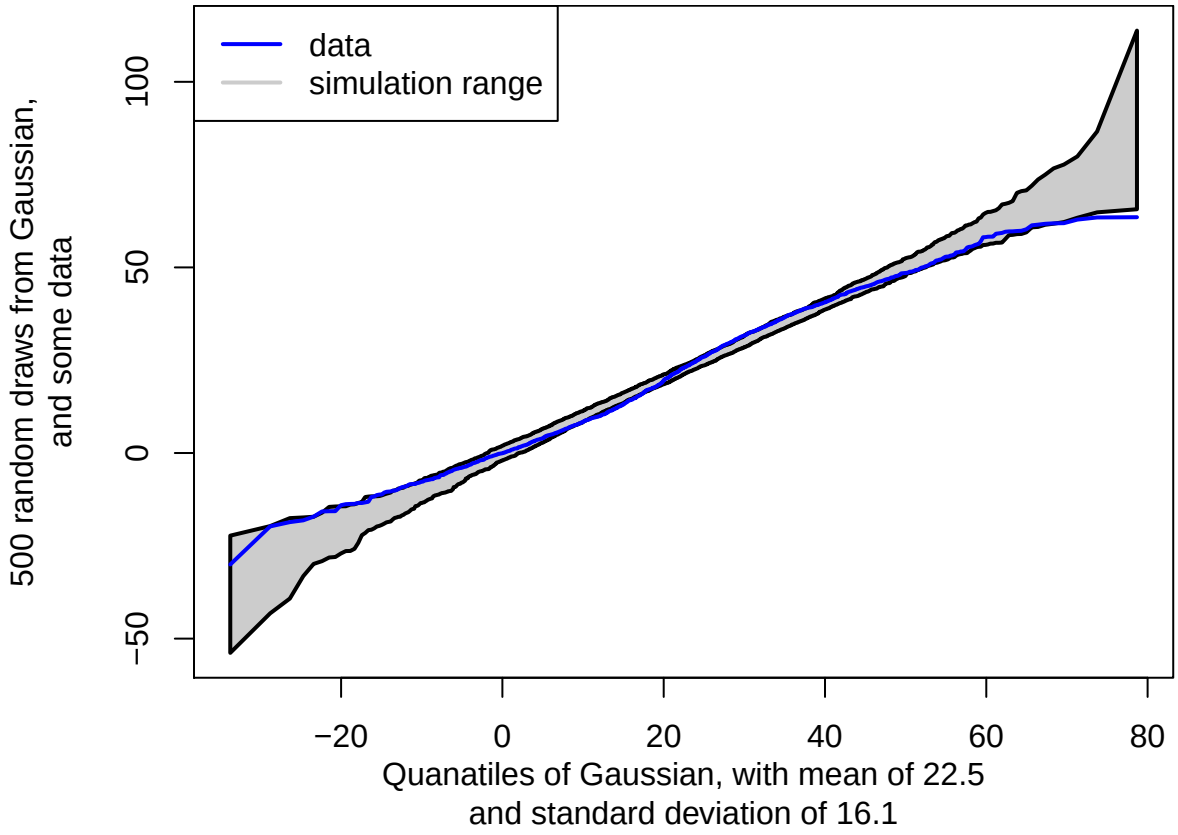


As mentioned above, this plot shows many random draws from a normal distribution compared to the quantiles from the normal distribution with the mean and standard deviation estimated from the data. The relationship between the quantiles and the data (data is the input, which could be residuals, or any set of values you expect to be drawn from a normal distribution) is shown as the blue line. The grey shape in the background is a summary of the random draws. I'm realizing this blog post has had little to do with the polygon function so far – the grey shape was made using polygon. For each quantile value, I saved the minimum and maximum ranked values from the random draws. This is a quick and dirty summary that doesn't account for the density of points at each quantile value – perhaps another blog post in the future. So, like the basic qqnorm function, we see the general pattern of the ranked data points compared to the quantiles of a normal distribution, but we also we the size of the departures we might expect to see given finite sample sizes. From this plot, we don't get a great sense of how much non linearity we should be comfortable with, but using the ranges as boundaries is still quite informative about how much departure from the one-to-one line we can expect – in general. This is best demonstrated by intentionally providing data that is not normally distributed, and is instead, lets say, normal but bi-modal:

```
sim.n <- 1000
data <- c(rnorm(n = sim.n, mean = 10,
    sd = 10), rnorm(n = sim.n,
    mean = 35, sd = 10))
hist(data)
```



```
example <- probPlotNorm(data, reps = 500)
```



To see how much this observed bi-modal distribution departs from the normal distribution we expect, it's more effective to see the differences between the minimum and maximum expected values at each quantile (the range), versus where the data lands on each quantile. This in a sense flattens the plot and emphasizes where the data is not between the min and max, and where there is curvature in the departures:

The supplemental plot function

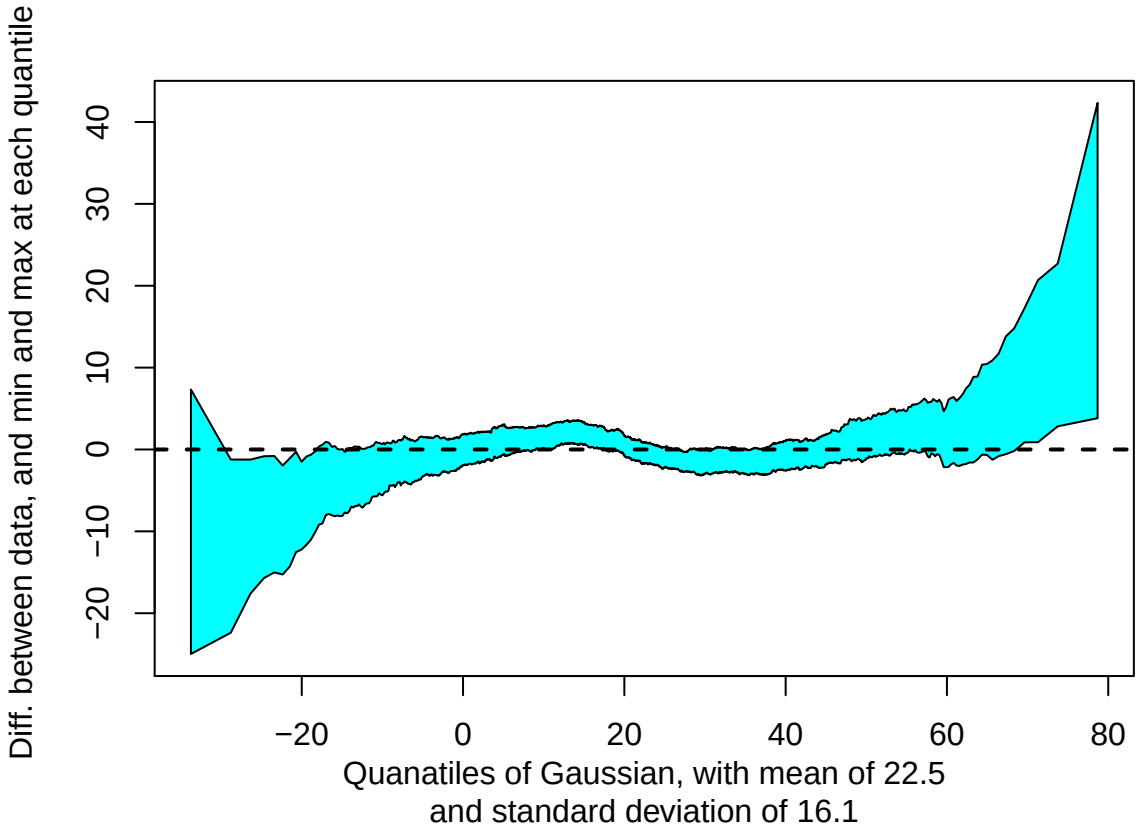
```
probPlotNorm_supp <- function(values,
    reps, ...) {

    # get mean and standard
    # deviation from values
    mt <- mean(values)
    msd <- sd(values)

    # get quantiles of normal dist
    # with mean = mt and sd = msd
    probs <- ppoints(n = length(values))
    quantNorm <- qnorm(p = probs,
```

```
    mean = mt, sd = msd)

# draw reps*length(values)
# random values from an normal
# with mt and msd
rvalues <- rnorm(n = reps *
    length(values), mean = mt,
    sd = msd)
normMat <- matrix(rvalues,
    nrow = reps, ncol = length(values),
    byrow = T)
sortedMat <- apply(normMat,
    1, sort)
rangeMat <- apply(sortedMat,
    1, range)


svals <- sort(values)
# difference between minimum
# simulated values at each
# quantile and the data values
# at each quantile
lowrange <- rangeMat[1, ] -
    svals
# difference between maxiumum
# simulated values at each
# quantile and the data values
# at each quantile
highrange <- rangeMat[2, ] -
    svals

# plot size range for y axis
ptrng <- c(lowrange, highrange)

# x and y labels
xxlab <- paste("Quanatiles of Gaussian, with mean of ",
    round(mt, 1), "\nand standard deviation of ",
    round(msd, 1), sep = "")
yylab <- "Diff. between data, and min and max at each quantile."


# change plot margins
par(mar = c(5, 5, 2, 2))
```

```
    # plot
    plot(1:length(lowrange), lowrange,
        type = "n", ylim = range(ptrng),
        xlim = range(quantNorm),
        ylab = yylab, xlab = xxlab)

    polygon(x = c(quantNorm, rev(quantNorm)),
        y = c(lowrange, rev(highrange)),
        lwd = 1, col = "cyan")
    abline(h = 0, lty = 2, lwd = 2,
        col = "black")

    # reset plot margins
    par(mar = c(5, 4, 4, 2) * 1.1)
}
```

Calling the supplemental plot (same data as before)

```
probPlotNorm_supp(data, reps = 500)
```



We see some clear non-linear patterns in the data that extend well beyond the expected range of the random draws. This is still not a quantitative test, but it gives (me at least) a much

better sense of when and by how much a set a values that are truly drawn from a normal distribution (or any distribution for which the parameters are readily estimated) can depart from the expected pattern by chance alone. Sounds a little more statistical at least eh?