Datenanalyse

Abgabe 1

Lukas Mahler (Matr. Nr. 11908553)

01.04.2021

Setup

Loading the library and the data from the loaded library:

```r
library("StatDA")
```

```
## Warning: package 'StatDA' was built under R version 4.0.4
## Loading required package: sgeostat
## Warning: package 'sgeostat' was built under R version 4.0.3
## Registered S3 method overwritten by 'geoR':
##   method         from
##   plot.variogram sgeostat
```

```r
data(chorizon)
```

Examples:

1st Example:

Histogram:

Histograms: a histogram is a graphical representation of numerical data. The data is divided into several bins that get drawn as rectangles according to the amount of data values contained in the bin. The interval length can get calculated by several distinct formulas, e.g. Sturges or Freedman and Diaconis. Histograms are used to get a broad overview of the given data to further analyze it with other tools.

Histogram for the given data (interval length from Sturges):

```r
pt <- chorizon$Pt
length(pt) # amount of values
```

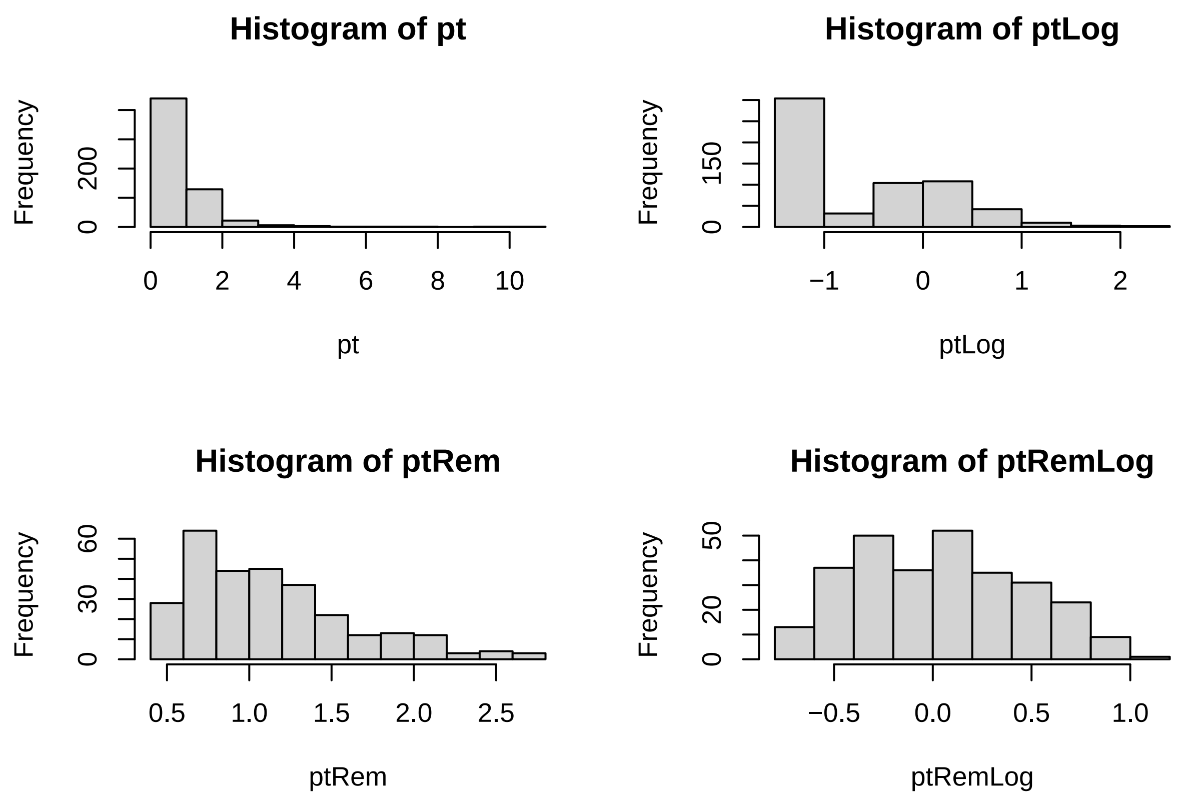
```
## [1] 605
```

```r
length(pt[pt %in% 0.25]) # amount of values = 0.25
```

```
## [1] 304
```

```r
ptRem <- pt[!pt %in% 0.25] # Removes the 0.25 values
ptRem <- ptRem[ptRem < 3] # Removes all values above 5
ptLog <- log(pt) # transforms to log()
ptRemLog <- log(ptRem) # transforms to log()
```

```r
par(mfrow = c (2,2))
hist(pt)
hist(ptLog)
hist(ptRem)
hist(ptRemLog)
```

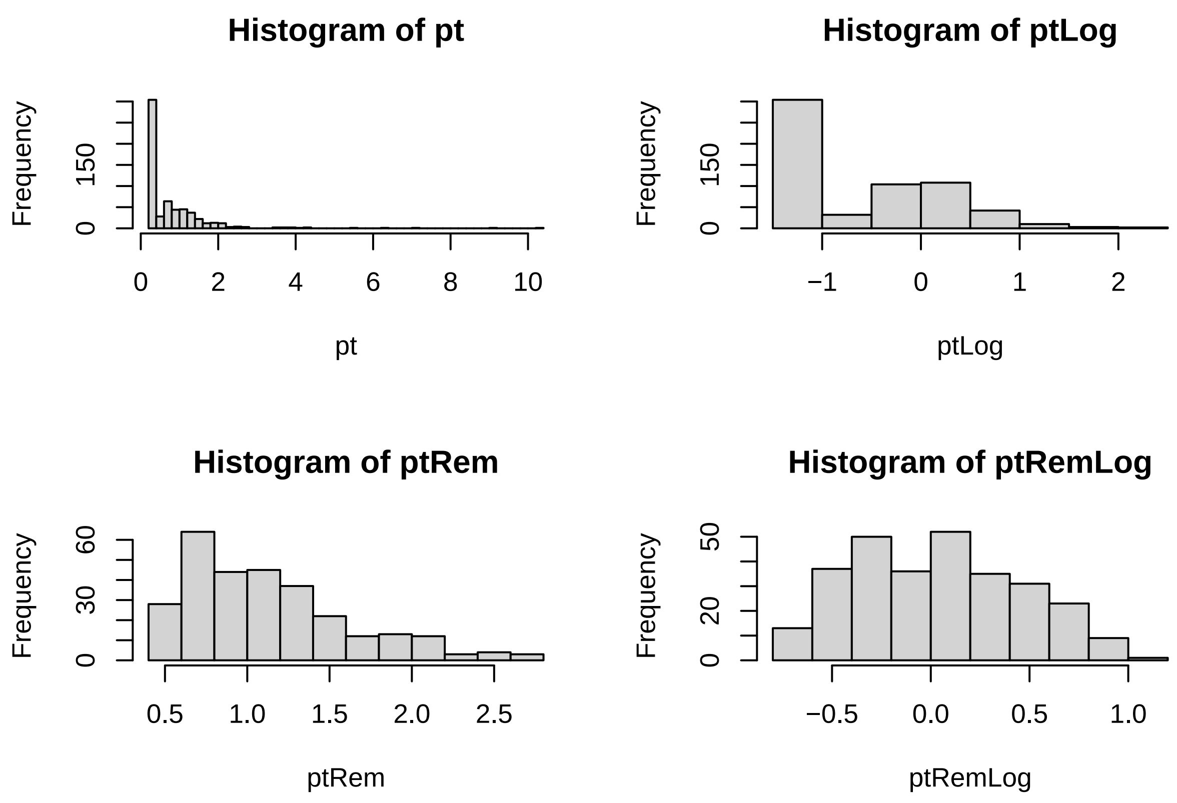


Here one cannot see how the values are distributed because the outliers increase the interval length rapidly. One can only see that most of the values lie between 0 and 2. Outliers range up to over 10.

To better accomodate a normal distribution we can do a log-transformation which gives the dataset more of a normal distribution. The Problems are the heavy outliers on the right side and the many duplicates of the value 0.25. Those get stripped in the variable ptRem. Depending on if the values of 0.25 are legit the transformation to a normalized variable could be helpful. On the other side, the amount of values = 0.25 make it impossible to get a normal distribution.

Histogram for the given data (interval length from Freedman and Diaconis):

```r
par(mfrow = c (2,2))
hist(pt, breaks="FD")
hist(ptLog, breaks="FD")
hist(ptRem, breaks="FD")
hist(ptRemLog, breaks="FD")
```

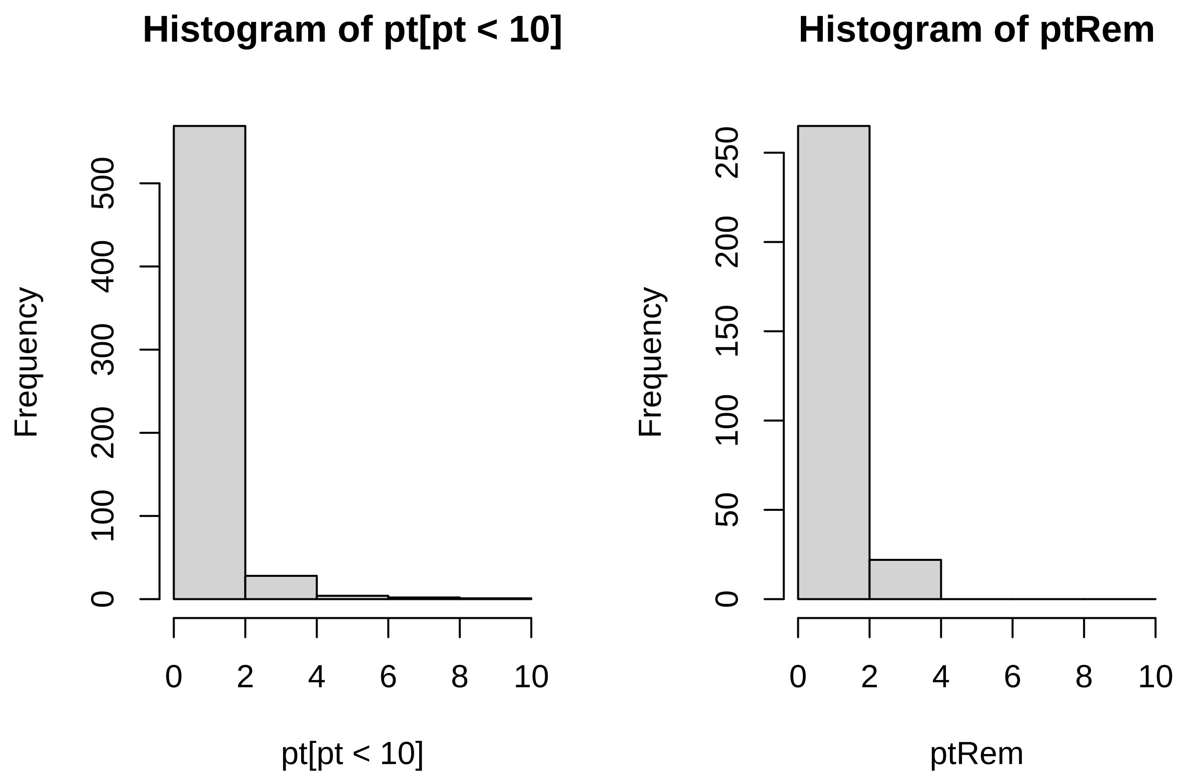


Here we can see the distribution more in depth. The interval length is small enough to see that most of the values not only lie between 0 and 2 but rather that over 300 of the values lie directly in the first interval (almost all having the value of 0.25 as we can see in the of "length(pt[pt %in% 0.25]".

The histogram of ptRemLog lets us see how this data could be transformed almost into a normal distribution (as we will be able to see with qpplot later) but at the cost of deleting several values (of 0.25 and above 3). As it is unlikely that the set has so many errors (more than half the data would be wrong) we will continue using the pt data (the original data without a log transformation).

Histogram for the given data (equidistant interval of 5 classes à length = 2)

```
br <- seq(0,10,2)
par(mfrow = c (1,2))
hist(pt[pt < 10], breaks=br)
hist(ptRem, breaks=br)
```



The histogram with the Friedman-Diaconis method works best for my data because the outliers increase the width of the intervals which reduces the detail to see any possible distributions. The fixed bin width of 2 is not ideal because important details get lost. Even more details get lost than in the standard interval method from Sturges.

When transforming to log, a potential normal distribution of the values from -0.5 to 0.7 can be spotted but is rather unlikely for the whole set because approximately half of the values is 0.25.

Density approximation:

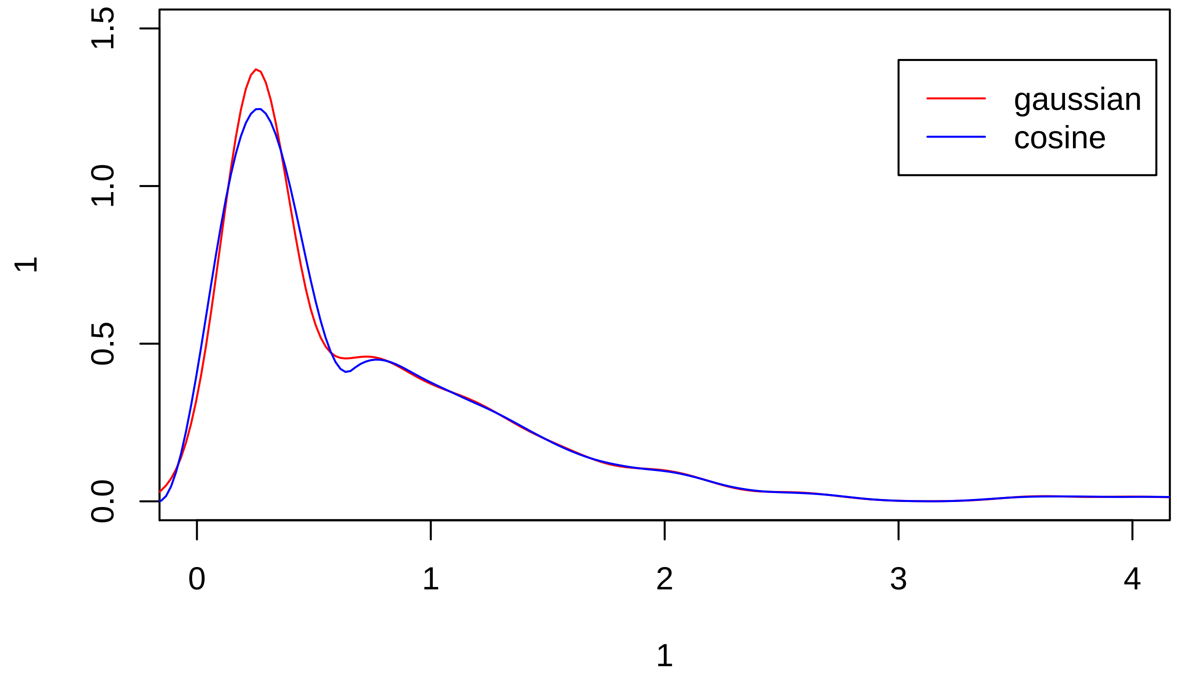
```
gaussian <- density(pt, kernel="gaussian")
cosine <- density(pt, kernel="cosine")

gaussian
```

```
##
## Call:
##  density.default(x = pt, kernel = "gaussian")
##
## Data: pt (605 obs.); Bandwidth 'bw' = 0.1486
##
##        x                 y
##  Min.   :-0.1958   Min.   :0.0000000
##  1st Qu.: 2.5271   1st Qu.:0.0003976
##  Median : 5.2500   Median :0.0033811
##  Mean   : 5.2500   Mean   :0.0916735
##  3rd Qu.: 7.9729   3rd Qu.:0.0281704
##  Max.   :10.6958   Max.   :1.3703846
```

```
cosine
```

```
##
## Call:
##  density.default(x = pt, kernel = "cosine")
##
## Data: pt (605 obs.); Bandwidth 'bw' = 0.1486
##
##        x                 y
##  Min.   :-0.1958   Min.   :0.0000000
##  1st Qu.: 2.5271   1st Qu.:0.0002764
##  Median : 5.2500   Median :0.0033182
##  Mean   : 5.2500   Mean   :0.0917264
##  3rd Qu.: 7.9729   3rd Qu.:0.0251358
##  Max.   :10.6958   Max.   :1.2442632
```

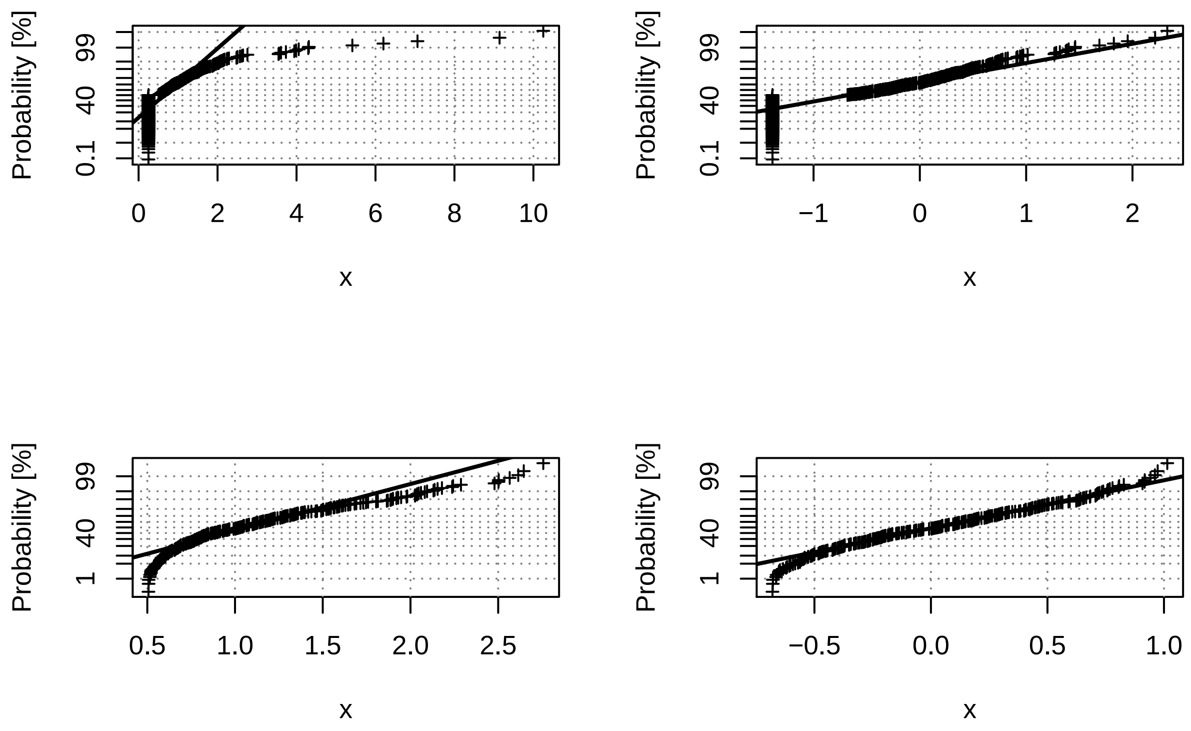
```
plot(1,1, type="n", xlim=c(0,4), ylim=c(0,1.5)); lines(gaussian, col="red"); lines(cosine, col="blue");
```



The cosine kernel would be more appropriate because there we can see the 'dip' between the many 0.25 values and the rest of the distribution that follows more of a normal distribution if transformed with log. On the gaussian kernel this fact is not included and would generate more generic datasets where this bump would be not as strong as in the cosine kernel.

Below there are some extra qpplots to show the possibility of a normal distribution of the values in the interval (0.25,3].

```r
par(mfrow= c(2,2))
qpplot.das(pt)
qpplot.das(ptLog)
qpplot.das(ptRem)
qpplot.das(ptRemLog)
```



2nd Example: