A3_tests

Azoacha Forcheh, 20558994

1. Have a look at the video https://youtu.be/HEheh1BH34Q which compares the sizes of various astronomical bodies. The video tries to give some sense of the comparative size of these bodies.

   In the `R` code folder there is a file called `stars.R` that contains measurements of the radius in kilometres of several astronomical bodies in our solar system (in a vector called `solarSystemRadii`) and of the radius of many stars as measured in numbers of solar radii (in a vector called `starRadii`). Load this file into `R` (use `source("directory/stars.R")` with the `directory` changed to wherever you put the file `stars.R`). For example:

```
source("./stars.R")
# Have a look at the contents of each
head(solarSystemRadii)
```

```
##         radius
## Sun     696000
## Jupiter  69911
## Saturn   58232
## Uranus   25362
## Neptune  24622
## Earth     6371
```

```
head(starRadii)
```

```
##                     radius
## binarystarvvcephei   1900
## v354cephei           1520
## mucephei             1420
## kycygni              1420
## v509cassiopeiae       900
## v838monocerotis      1570
```

   (a) In the video, the relative size of the planets is encoded in at least three ways.

      i. **(3 marks)** Name them.

         Volume, position along a common scale, length

      ii. **(3 marks)** According to Stevens's law, which encoding is most reliably encoded? Which least? (Justify your answers by appeal to this law.)

         According to Steven's law, length is most reliable encoding as it creates the smallest bias when encoding the relative size of the planets, which is due to its higher empirically observed range of values $0.9 \leq \beta \leq 1.1$, making $p(x) \approx cx$. The least reliable encoding is volume, which has the largest bias of the 3 when encoding the relative size of the planets, due to its lower empirically observed range of values $0.5 \leq \beta \leq 0.8$.

      iii. **(1 marks)** How might position along a common scale have been used instead? As the bodies being considered were all spheres, the position along a common scale could have been used to display and compare the radii (a measurement of length) of the bodies instead of the volumes, which is only dependent on the radii and is a less accurate encoding according to Steven's law.

   (b) For the `solarSystemRadii` data:

      Here you are going to draw the various astronomical bodies in our solar system in much the same way as they appear in that video. You will be making use of the `grid` package (as seen in previous assignments).

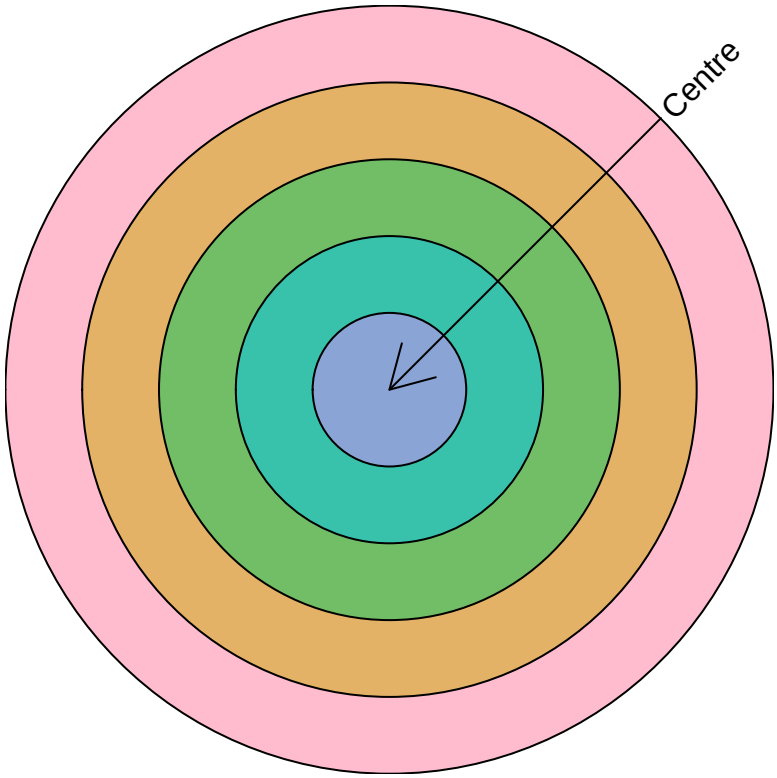
      You will need to recall a few things about `grid`. As before, you will just be drawing circles to represent the various bodies so the function `gridcircle(..)` will be used. Remember that (so far anyway) that you are always drawing within a [0,1] rectangle do that everything you draw will need to be translated to

this range (e.g. dividing all radii by the maximum diameter will allow all circles will fit within the unit square when centred at (0.5, 0.5).

You will also need a palette of colours, one for each body. How to construct a palette of was described during an earlier lecture on numbers and made available on the course web page.

The code below should give you some idea of what is possible. (See help(...) on any function to explore more parameter choices.)

```
library(grid)
library(colorspace)
cols <- rainbow_hcl(n=5, c=100)  # 5 different colours having chroma = 100

grid.newpage()
for (i in 1:5) {
  grid.circle(x=0.5,
              y=0.5,
              r = (6-i)/10,
              gp=gpar(fill=adjustcolor(cols[i], alpha.f = 0.5))
  )
}
# Now add a label and an arrow
xcentre <- 0.5
ycentre <- 0.5
degrees <- 45
radians <- pi * degrees / 180
arrowLength <- 0.5
xarrowFrom <- xcentre + arrowLength * cos(radians)
yarrowFrom <- ycentre + arrowLength * sin(radians)
xarrowTo <- xcentre
yarrowTo <- ycentre
# draw arrow
grid.lines(x=c(xarrowFrom, xarrowTo), y=c(yarrowFrom, yarrowTo),
           arrow=arrow(),
           gp = gpar(col="black", lwd=1, lty = 1))

delta <- 0.01
xText <- xarrowFrom + delta * cos(radians)
yText <- yarrowFrom + delta * sin(radians)
grid.text("Centre", x= xText, y= yText,
          just="left", rot = degrees,
          gp = gpar(col="black"))
```



These functions (and the `grid` package) are to be used to address the following questions.

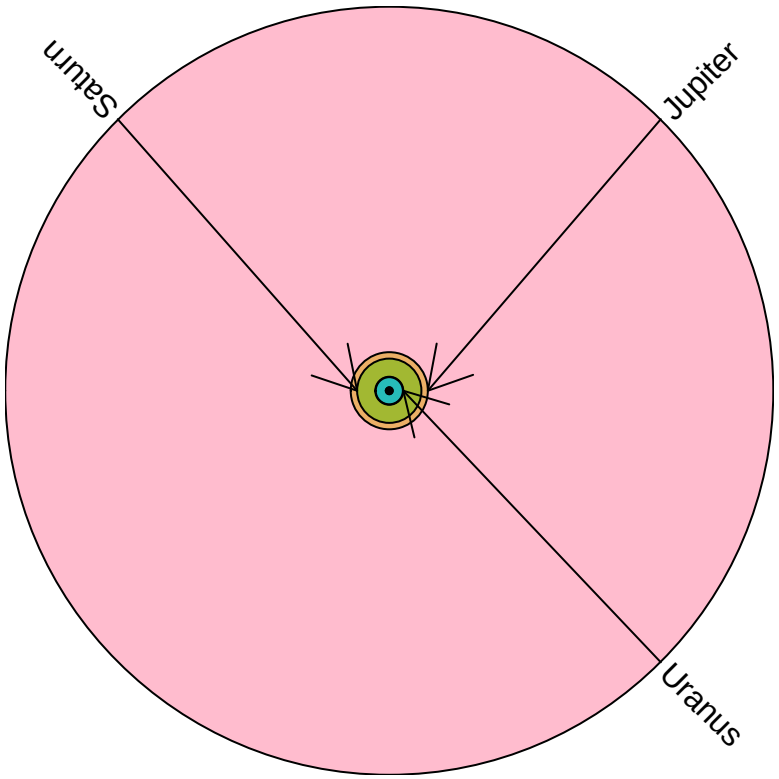
i. **(5 marks)** Represent each body by a circle whose radius is proportional to the radius in kilometres of that body (use the radius of the sun as the maximum possible radius in the display).

Align the circles so that they are all centred on (0.5, 0.5). Label each of the three largest bodies (excluding the sun) on the plot. Show your code and the picture produced.

IMPORTANT: you need to maintain an aspect ratio of 1 so begin your `r` code chunks in `Rmarkdown` with something like `{r, fig.align="center", fig.width=4, fig.height=4}`

```
library(grid)
library(colorspace)

xcentre = 0.5
ycentre = 0.5
cols = rainbow_hcl(6, c = 100)
n = dim(solarSystemRadii)[1]
max_diam = solarSystemRadii['Sun',]

grid.newpage()
for (i in 1:n) {
  grid.circle(x=xcentre,
              y=ycentre,
              r = (solarSystemRadii[i,]/max_diam)/2,
              gp=gpar(fill=adjustcolor(cols[i], alpha.f = 0.5))
  )
}

degs = c(45, 135, 315)
for (i in 2:4) {
  degrees = degs[i-1]
  radians = pi * degrees / 180
  arrowLength = 0.5
```

```
  xarrowFrom = xcentre + arrowLength * cos(radians)
  yarrowFrom = ycentre + arrowLength * sin(radians)

  if (degrees == 135) {
    xarrowTo = xcentre - (solarSystemRadii[i,]/max_diam)/2
  } else {
    xarrowTo = xcentre + (solarSystemRadii[i,]/max_diam)/2
  }

  yarrowTo = ycentre
  # draw arrow
  grid.lines(x=c(xarrowFrom, xarrowTo), y=c(yarrowFrom, yarrowTo),
             arrow=arrow(),
             gp = gpar(col="black", lwd=1, lty = 1))

  delta = 0.01
  xText = xarrowFrom + delta * cos(radians)
  yText = yarrowFrom + delta * sin(radians)

  body_name = rownames(solarSystemRadii)[i]
  grid.text(body_name, x= xText, y= yText,
            just="left", rot = degrees,
            gp = gpar(col="black"))
}
```



ii. **(2 marks)** Briefly comment on how easily it is to compare the relative sizes of Uranus to Saturn? Of Saturn to Jupiter? Of Jupiter to the Sun? How about comparing Uranus to the Earth or the Moon?

Because the two circles representing the planets are both relatively large and are overlaid immediately (i.e. with no other circle between the two) over each other, and because the colors of the circles are diverging, it is easy to see that Uranus is much smaller than Saturn as we can still see much of the area of the circle representing Saturn. However, it is difficult to determine by what proportion this

difference is as we comparing the two via their areas.

For Saturn and Jupiter, it is once again easy to compare the two planets for the exact reasons mentioned above, with it being easy to determine that Saturn is slightly larger than Jupiter. The same applies for comparing Jupiter to the Sun, though it is harder to make this comparison and most of Jupiter is not visible under the overlaying circles.

Due to the Moon and Earth being significantly smaller than the Sun, they are very difficult to see and compare with any of the other bodies. One can tell that the Moon and Earth are smaller than Uranus, but it is extremely difficult to determine the ratio of difference.

iii. **(4 marks)** According to Stevens's law, what is the range of values we might expect for the ratio of the areas (smaller to larger) of each of the above comparisons? How do these compare to the ratio of actual areas?

$$\begin{aligned}
\text{Uranus vs. Saturn} &: [(\frac{25362^2\pi}{58232^2\pi})^{0.9}, (\frac{25362^2\pi}{58232^2\pi})^{0.6}] = [0.2239955, 0.3688299] \\
\text{Saturn vs. Jupiter} &: [(\frac{58232^2\pi}{69911^2\pi})^{0.9}, (\frac{58232^2\pi}{69911^2\pi})^{0.6}] = [0.7196298, 0.8030442] \\
\text{Jupiter vs. the Sun} &: [(\frac{69911^2\pi}{696000^2\pi})^{0.9}, (\frac{69911^2\pi}{696000^2\pi})^{0.6}] = [0.01597663, 0.06343421] \\
\text{Earth vs. Uranus} &: [(\frac{6371^2\pi}{25362^2\pi})^{0.9}, (\frac{6371^2\pi}{25362^2\pi})^{0.6}] = [0.08318469, 0.1905588] \\
\text{The Moon vs. Uranus} &: [(\frac{1737.1^2\pi}{25362^2\pi})^{0.9}, (\frac{1737.1^2\pi}{25362^2\pi})^{0.6}] = [0.00796984, 0.0398994]
\end{aligned}$$

In comparison, the actual ratios are:

$$\begin{aligned}
\text{Uranus vs. Saturn} &: 0.1896896 \\
\text{Saturn vs. Jupiter} &: 0.6937969 \\
\text{Jupiter vs. the Sun} &: 0.01008957 \\
\text{Earth vs. Uranus} &: 0.004691186 \\
\text{The Moon vs. Uranus} &: 0.06310274
\end{aligned}$$

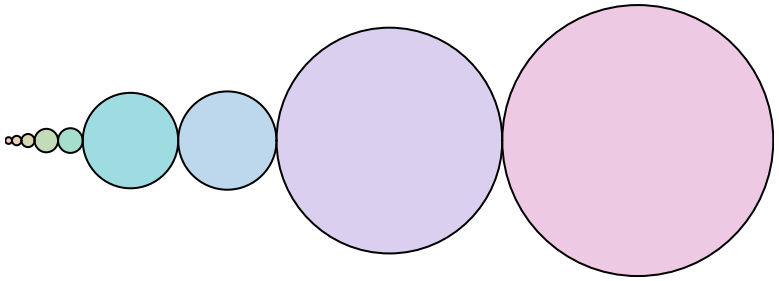
Every one of the ratio of actual areas falls outside and to the left of the calculated ranges for the ratios. The demonstrates the bias and reduced accuracy of using area as a visual encoding of magnititude.

iv. **(5 marks)** Consider now all of the bodies in the solar system **excluding the sun**. Using the `grid` package and `grid.circle(...)` etc. lay out all of the remaining bodies from the smallest at the left to the largest at the right with their centres all at $y = 0.5$ but locate them so that they do not overlap. Have the radius of each circles be proportional to the true radius of that body. Mark the Earth, Uranus, Saturn, Jupiter on the plot.

Show your code and your output. Some functions you might find useful are `order(...)` to determine the order of values and possibly `%in%` as in `"foo" %in% c("foo", "bar")` will return `TRUE` if the string `"foo"` can be found in the vector `c("foo", "bar")`; the latter may be helpful in deciding whether to label or not.

```
cols = rainbow_hcl(n-1)

asc_order = order(solarSystemRadii)
planets = solarSystemRadii[asc_order[! asc_order %in% c(1)],]
planet_names = rownames(solarSystemRadii)[asc_order[! asc_order %in% c(1)]]
total_length = 2*sum(planets)+0.08
planetsr = planets/total_length

xcenters = replicate(n-1, 0)
for (i in 1:(n-1)) {
  xcenters[i] = (2*sum(planetsr[1:i-1])) + planetsr[i]
}
```

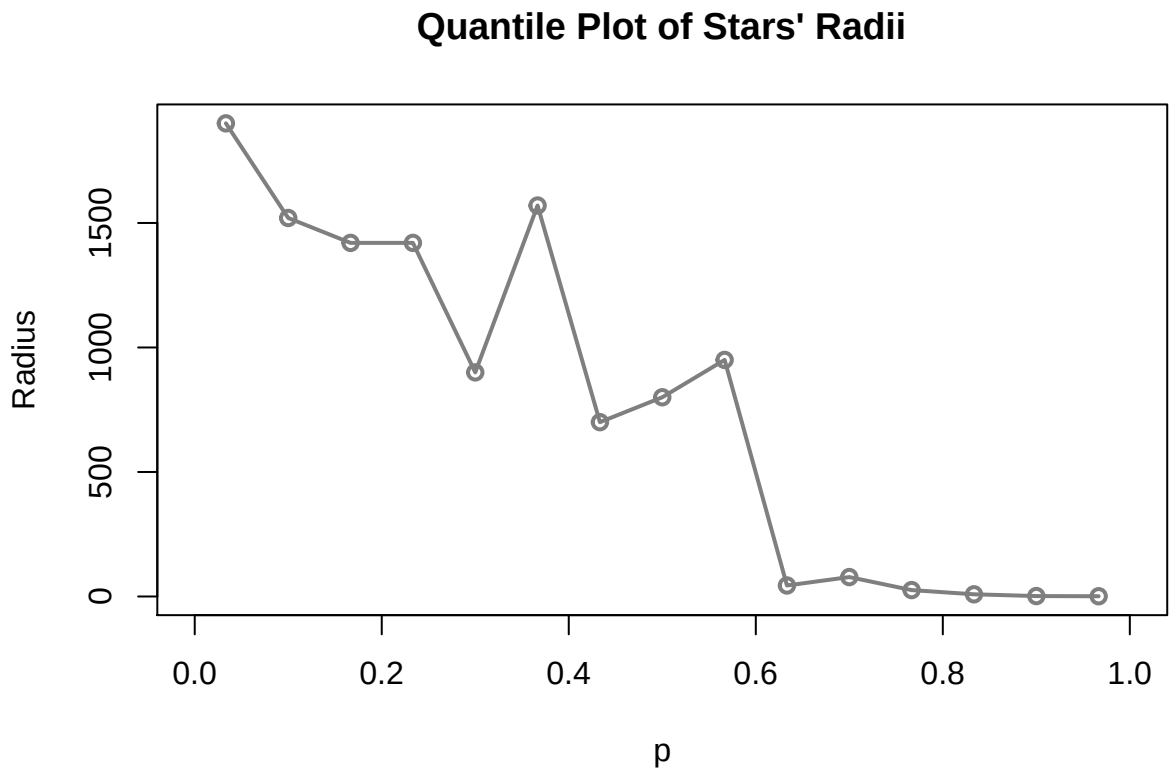
```
grid.newpage()
xcenter = 0
ycenter = 0.5

for (i in 1:(n-1)) {
  grid.circle(x=xcenters[i],
              y=ycenter,
              r = planetsr[i],
              gp=gpar(fill=adjustcolor(cols[i], alpha.f = 0.5)))

  body_name = planet_names[i]
  if (body_name %in% c('Earth', 'Uranus', 'Saturn', 'Jupiter')) {
    grid.text(body_name, x=xcenters[i], y=0.09,
              just="left",
              gp = gpar(col="black"))
  }
}
```



(c) For the `stars` data,

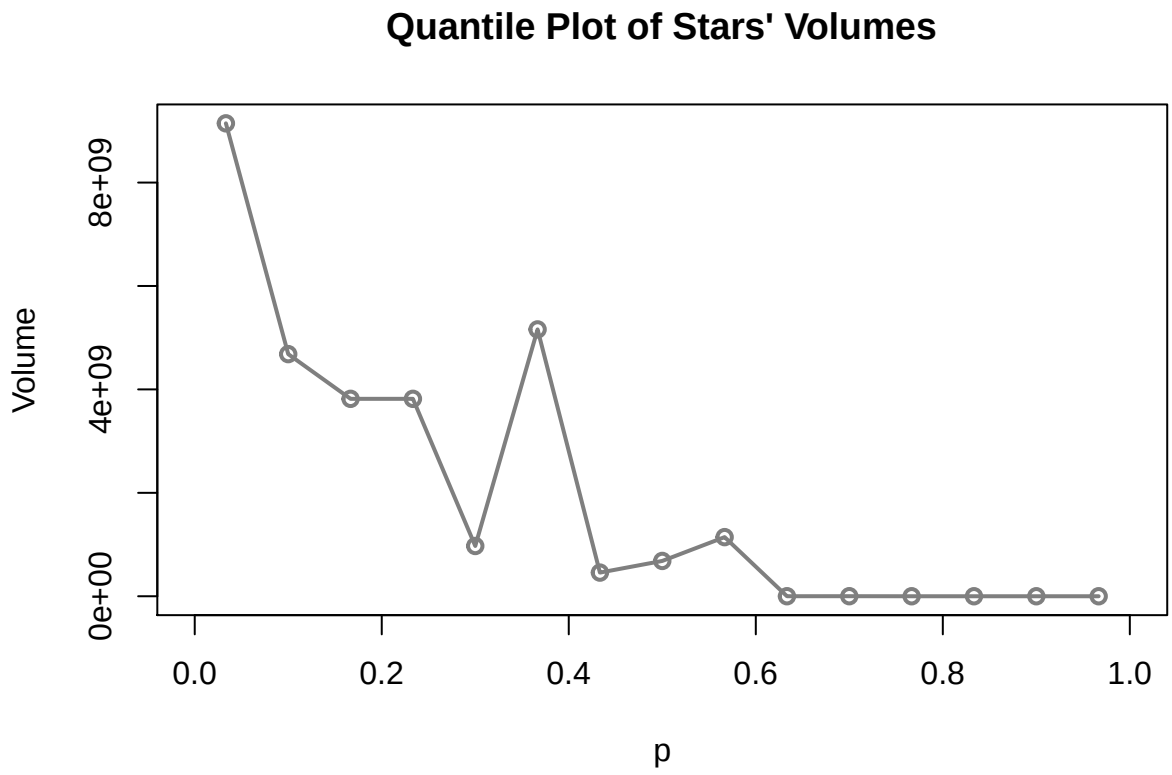
i. **(3 marks)** Construct a quantile plot of the radii of the `stars`. Describe whatever patterns you see in the data.

```
n = dim(starRadii)[1]
p = ppoints(n)
plot(x = p, y=starRadii[,],
      type="o", lwd=2, col="grey50",
      xlab="p", ylab="Radius",
      main="Quantile Plot of Stars' Radii",
      xlim =c(0,1))
```



ii. **(3 marks)** Construct a quantile plot of the volume of the `stars`. (Recall: the volume of a sphere is $\frac{4}{3}\pi r^3$.) Describe whatever patterns you see in the data.

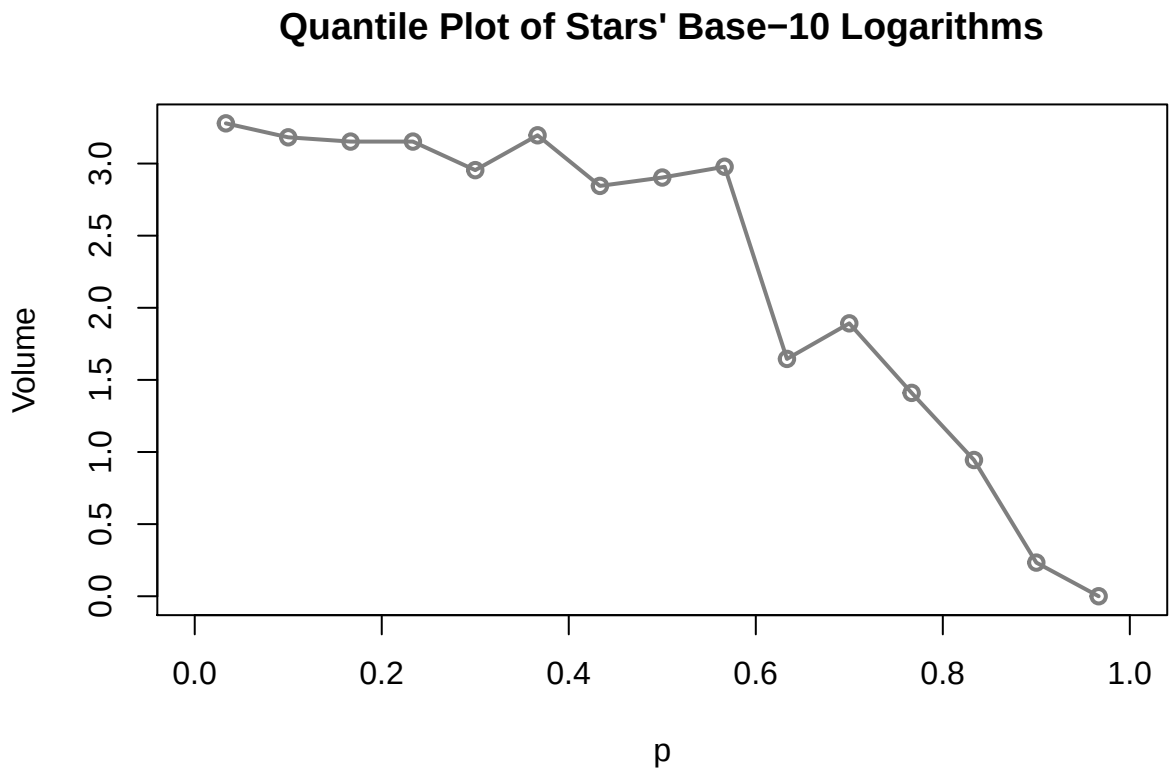
```
starVolumes = 4*(starRadii[,]^3)/3
plot(x = p, y=starVolumes,
      type="o", lwd=2, col="grey50",
      xlab="p", ylab="Volume",
      main="Quantile Plot of Stars' Volumes",
      xlim =c(0,1))
```



iii. **(2 marks)** How do these two summaries differ?

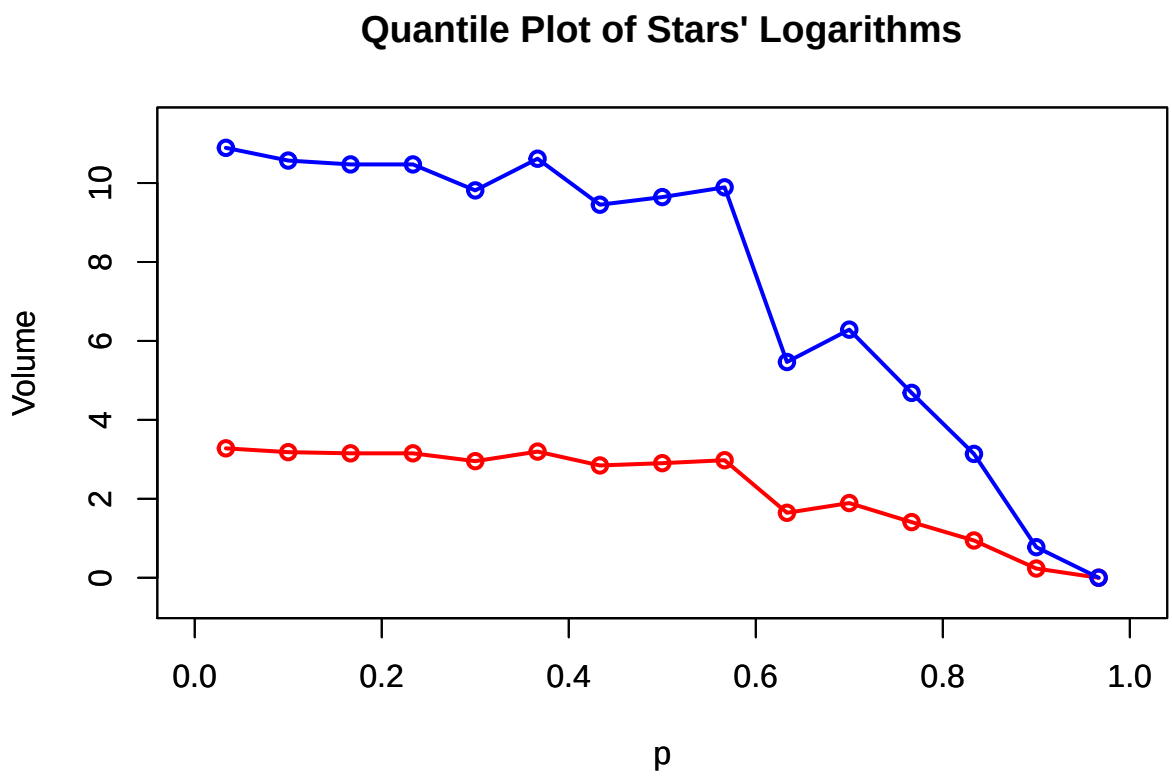
iv. **(3 marks)** Construct a quantile plot of the base 10 logarithms of the stellar radii. Describe whatever patterns you see in the data.

```
starLogs = log(starRadii[,], base=10)
plot(x = p, y=starLogs,
      type="o", lwd=2, col="grey50",
      xlab="p", ylab="Volume",
      main="Quantile Plot of Stars' Base-10 Logarithms",
      xlim =c(0,1))
```



v. **(4 marks)** Again construct a quantile plot of the logarithms of the stellar radii **but** this time use base 2 logarithms. In the same plot, overlay the quantiles for the base 10 logarithms. Explain how and why the two sets of quantiles differ.

```
starLog2s = log(starRadii[,], base=2)
ylims = extendrange(c(starLogs, starLog2s))

plot(x = p, y=starLogs,
      type="o", lwd=2, col="red",
      xlab="p", ylab="Volume",
      main="Quantile Plot of Stars' Logarithms",
      xlim =c(0,1), ylim=ylims)

par(new = TRUE)

plot(x = p, y=starLog2s,
      type="o", lwd=2, col="blue",
      ylab="Volume",
      xlim =c(0,1), ylim=ylims)
```


Quantile Plot of Stars' Logarithms
Volume
0
2
4
6
8
10
0.0
0.2
0.4
0.6
0.8
1.0
p