

# Practice Questions Computer Test

## STAT5003 Semester 1, 2022

### Question 1

Consider a simple linear regression

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i.$$

However instead of assuming the error variables  $\varepsilon \sim \mathcal{N}(0, \sigma^2)$  where  $\sigma > 0$ , assume  $\varepsilon_i \sim \text{Laplace}(\sigma)$  where *Laplace* denotes the Laplace distribution which has the density,

$$f(x|\sigma) = \frac{1}{2\sigma} \exp(-|x|/\sigma)$$

This can lead to the likelihood model (you do not need to show this) where the log likelihood of the parameters  $\theta = (\beta_0, \beta_1, \sigma)$  is given by

$$\mathcal{L}(\theta|\mathcal{X}) = \begin{cases} -\infty, & \text{if } \sigma < 0; \\ -n \log(2\sigma) - \frac{1}{\sigma} \sum_{i=1}^n |Y_i - \beta_0 - \beta_1 X_i|, & \text{otherwise.} \end{cases}$$

where  $\mathcal{X} = (Y_1, \dots, Y_n, X_{11}, \dots, X_{1n}, X_{21}, \dots, X_{2n})$

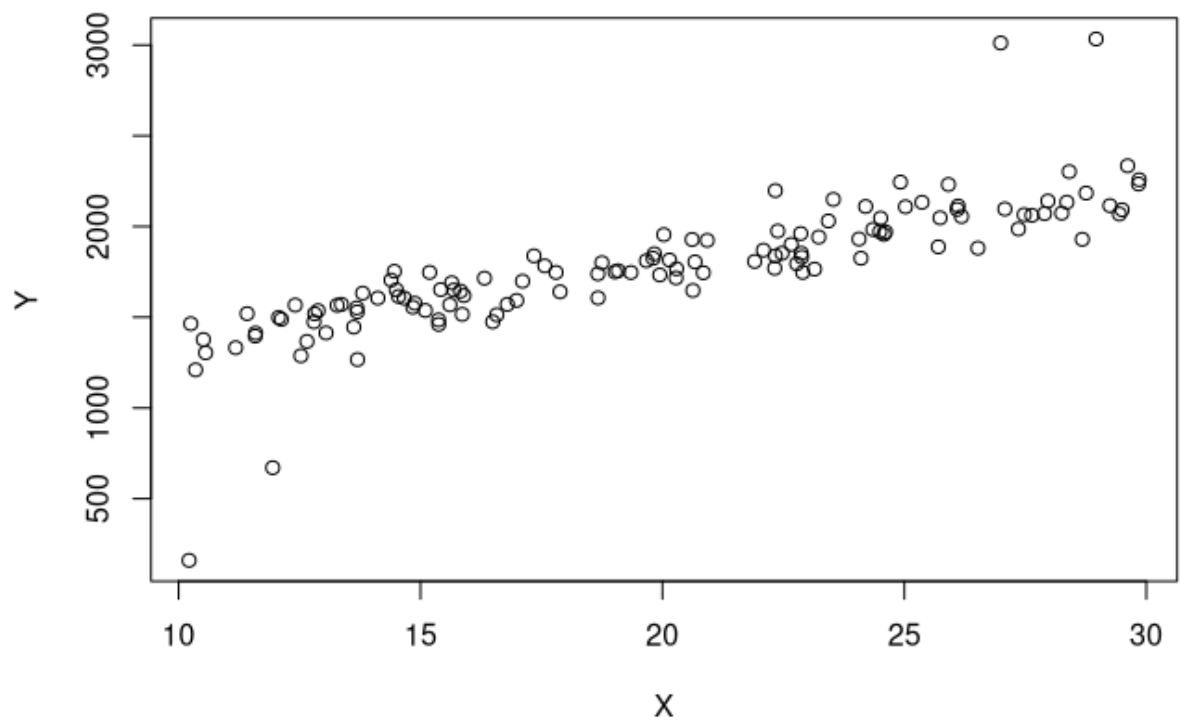
For this question consider estimating the simple linear regression coefficients using the method of Maximum Likelihood Estimation (MLE) assuming Laplace errors instead of Gaussian errors. A dataset is provided to explore this estimation method. It is given in the rds file `q1dat.rds` available on canvas.

- a. [3 marks] Load and inspect the `q1.dat` data, verify it has 2 numeric variables. Then create a scatterplot of the data.

```
q1.dat <- readRDS("q1dat.rds")
str(q1.dat)
```

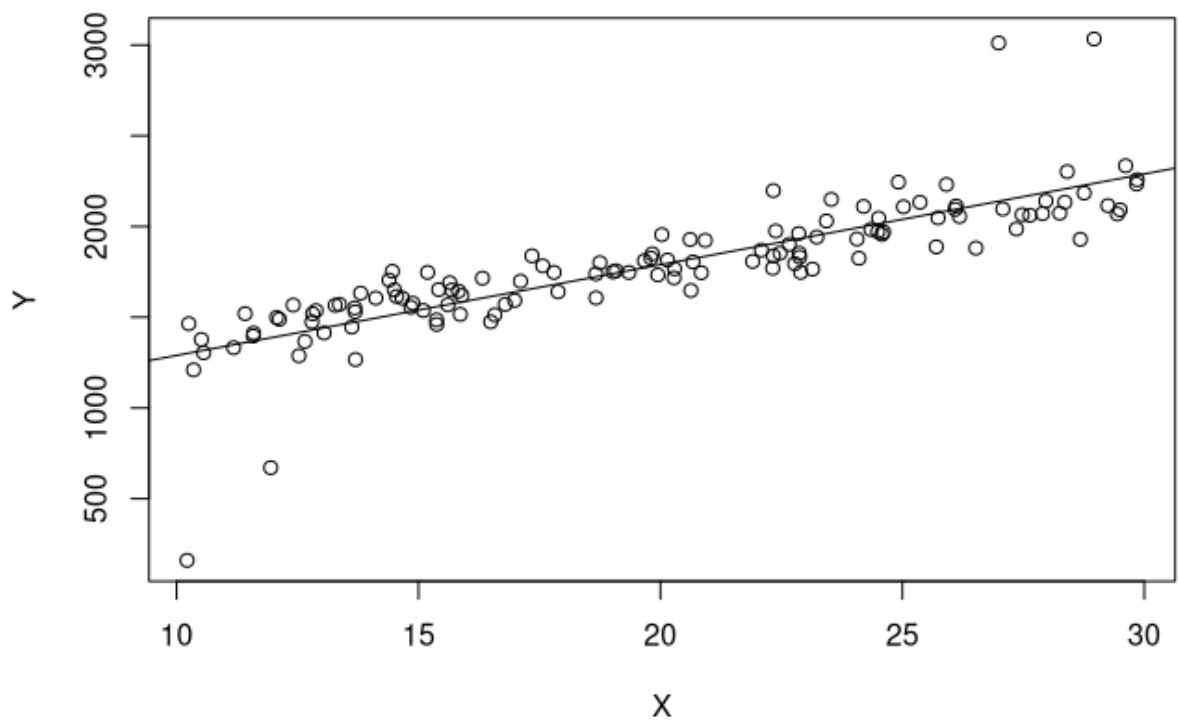
```
## 'data.frame':    128 obs. of  2 variables:
##  $ X: num  22.3 25.9 24.2 25.7 27 ...
##  $ Y: num  1770 2232 2110 2048 3012 ...
```

```
# Input your answer here
plot(Y ~ X, data = q1.dat)
```



- b. [4 marks] Fit the linear regression using the `lm` function in `R` and extract the regression coefficients (the  $\beta$  values) and the estimate of  $\sigma$ . Add the least squares regression line to your scatterplot.

```
# Input your answer here
q1.lm <- lm(Y ~ X, data = q3.dat)
q1.lm.coefs <- coef(q3.lm)
q1.lm.sigma <- sigma(q3.lm)
plot(Y ~ X, data = q1.dat)
abline(q1.lm)
```



- c. [3 marks] Define a function in `R` that computes the *negative* log-likelihood as a function of  $\theta$  defined above. (Note  $\theta = (\beta_0, \beta_1, \sigma)$  has 3 elements and will require a function with three input arguments).

```
# Input your answer here
negLaplaceLikelihood <- function(beta0, beta1, sigma)
{
  Y <- q1.dat[["Y"]]
  X <- q1.dat[["X"]]
  if (sigma < 0) return(Inf)
  length(X) * log(2 * sigma) + sum(abs(Y - beta0 - beta1 * X))/sigma
}
```

- d. [2 marks] As an alternative algorithm to estimate the regression coefficients and  $\sigma$ , use the `q1.dat` data to compute the MLE of  $\theta$  using the `stats4::mle` function (i.e. the function `mle` in the `stats4` package). Use the least squares estimates computed in the previous part as the starting point of the algorithm and compare the results to the least squares estimates.

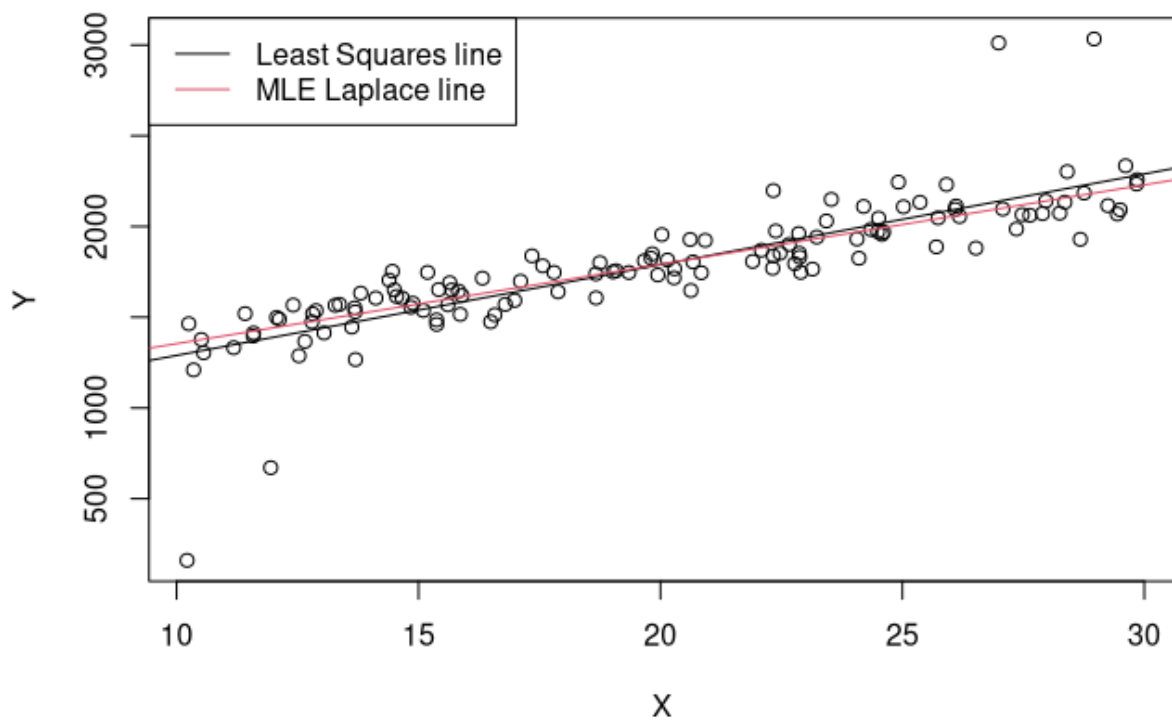
```
# Input your answer here
starting.point <- list(beta0 = q1.lm.coefs[1], beta1 = q1.lm.coefs[2], sigma = q1.lm.sigma)
mle.estimates <- stats4::mle(negLaplaceLikelihood, start = starting.point)
mle.coef <- mle.estimates@coef
matrix(c(q1.lm.coefs, q3.lm.sigma, mle.coef), nrow = 2, byrow = TRUE,
       dimnames = list(c("Least Squares", "MLE"), c("beta0", "beta1", "sigma")))
```

```
##                beta0    beta1    sigma
## Least Squares 789.9687 49.99192 193.0717
## MLE           917.1096 43.75590 108.7159
```

This is seen with a smaller slope but larger intercept for the MLE estimates compared to the Least Squares estimates.

- e. **[4 marks]** Create a scatterplot which has all the data and both the least squares line and MLE Laplace line estimate (use a legend to label the lines). Comment on the the Laplace line vs least squares line compared to the observed data and refer to the scatterplot in your answer.

```
# Input your answer here
plot(Y ~ X, data = q1.dat)
abline(q1.lm.coefs)
abline(mle.coef[1:2], col = 2)
legend("topleft", legend = c("Least Squares line", "MLE Laplace line"),
      col = 1:2, lty = rep(1, 2))
```



The MLE Laplace line is closer to the center cloud of points while the least squares line is tilted closer to the outliers on the top right and bottom left. It is less affected by the outliers in the Y values at the edge of the domain.

## Question 2

In some datasets confidentiality is an issue and to prevent disclosure, datasets are modified and anonymised to prevent sensitive information from being leaked. An anonymised output has been provided and is available on canvas in the file `encoded.data.rds`. This data file has numeric variables/features along the columns except the last column which is a class label variable denoting the assigned `type` for that case. The command to load the data is given below.

```
encoded.data <- readRDS("encoded.data.rds")
```

Using this data, answer the following questions. In the questions that involve Principal Components Analysis (PCA), use the `prcomp` command in R with the default options.

- [2 marks]** Create a `data.frame` or `matrix` with the `type` class labels removed.

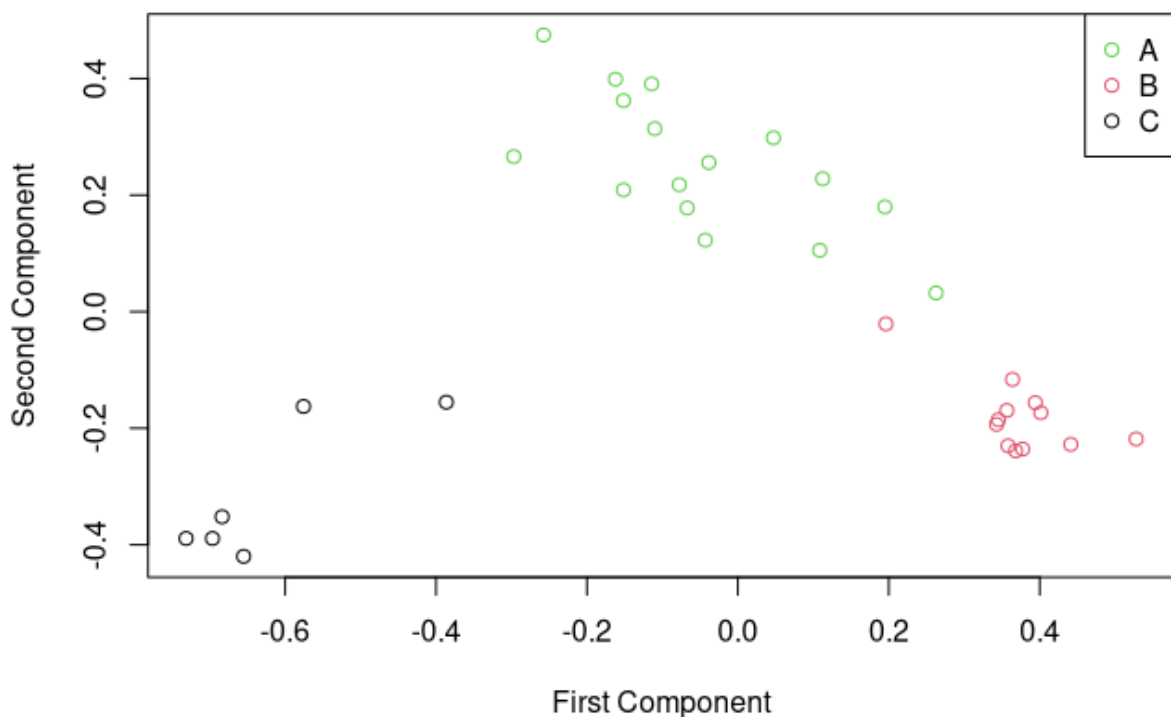
Determine the dimension of the new `data.frame` or `matrix` **Solution:**

```
# The following definitions are equivalent
encoded.data.without.labels <- encoded.data[-ncol(encoded.data)]
encoded.data.without.labels <- encoded.data[-11]
encoded.data.without.labels <- encoded.data[, -11]
encoded.data.without.labels <- encoded.data[, -which(names(encoded.data)
== "type")]
dim(encoded.data.without.labels)
```

```
## [1] 34 10
```

- b. [5 marks] Perform principal components analysis on the unlabelled dataset. Make a scatterplot using the first two principal components and colour the scatterplot by the `type` label. Comment on whether the dimension reduction successfully allows to discriminate between the different text `type`.

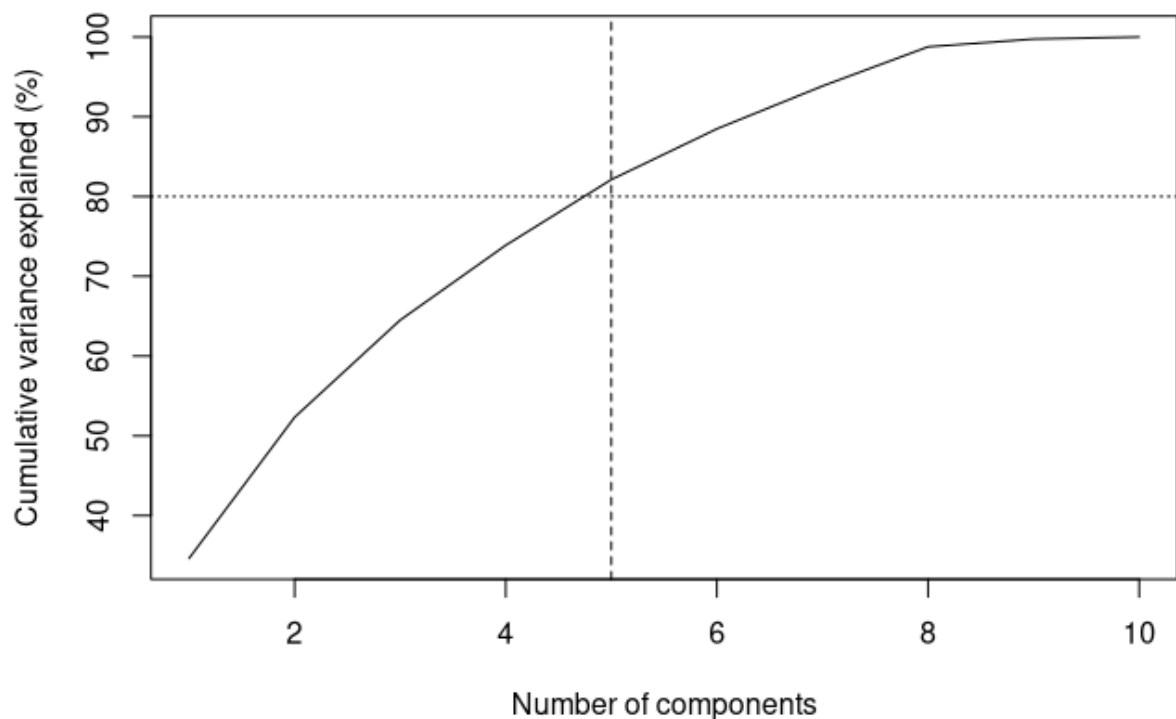
```
pca <- prcomp(encoded.data.without.labels)
plot(pca$x[,1], pca$x[,2], col = encoded.data[[ncol(encoded.data)]],
     xlab = "First Component", ylab = "Second Component")
legend("topright", legend = levels(encoded.data[[ncol(encoded.data)]]),
     col = unclass(unique(encoded.data[[ncol(encoded.data)]])),
     pch = 1)
```



Based off the plot we can see that it seems to adequately discriminate between the different data types. If the Second component is positive, it seems to be type A, if the second component is negative it is either type B or C where it is type B if it also has a first component positive or type C if the first component is negative.

- c. [4 marks] Compute the amount of variance in the unlabelled encoded.data that is explained by the PCA for each principal component. Using this, produce a plot of the *cumulative* variance explained against the number of components. Determine the minimum number of components required to explain at least 80% of the variance in the encoding.

```
variance.explained <- pca$sdev^2
cumulative.variance <- cumsum(prop.table(pca$sdev^2) * 100)
plot(1:10, cumulative.variance, xlab = "Number of components", ylab = "Cumulative variance explained (%)", type = 'l')
abline(h = 80, lty = 'dotted')
abline(v = 5, lty = 'dashed')
```



```
which.max(cumulative.variance >= 80)
```

```
## [1] 5
```

From the plot and calculation we can see that 5 components are needed to explain more than 80% of the variation in the data.

- d. [2 marks] Construct  $B = 34$  bootstrap samples of the data loaded in part a. by resampling the rows *with replacement*.

```
B <- nrow(encoded.data.without.labels)
set.seed(21042021) # Both commands below equivalent
bootstrap.samples <- lapply(1:B, function(x) encoded.data.without.labels
[sample(1:B, size = B, replace = TRUE), ])
alternative.bootstrap.indexing <- lapply(1:B, function(x) sample(1:B, si
ze = B, replace = TRUE))
alternative.bootstrap.samples <- lapply(alternative.bootstrap.indexing,
function(x) encoded.data.without.labels[x, ])
```

- e. [2 marks] Recalculate the PCA on each bootstrapped sample. For each bootstrapped PCA, compute the cumulative variance explained by each principle component within the PCA. Create a `data.frame` with all the the results, it should have either 306 or 340 rows (the last cumulative variance calculation is always 100% for the last principal component and is optional to compute) and two columns. The first column should be the cumulative variance explained and the second should be the count of number of components used.

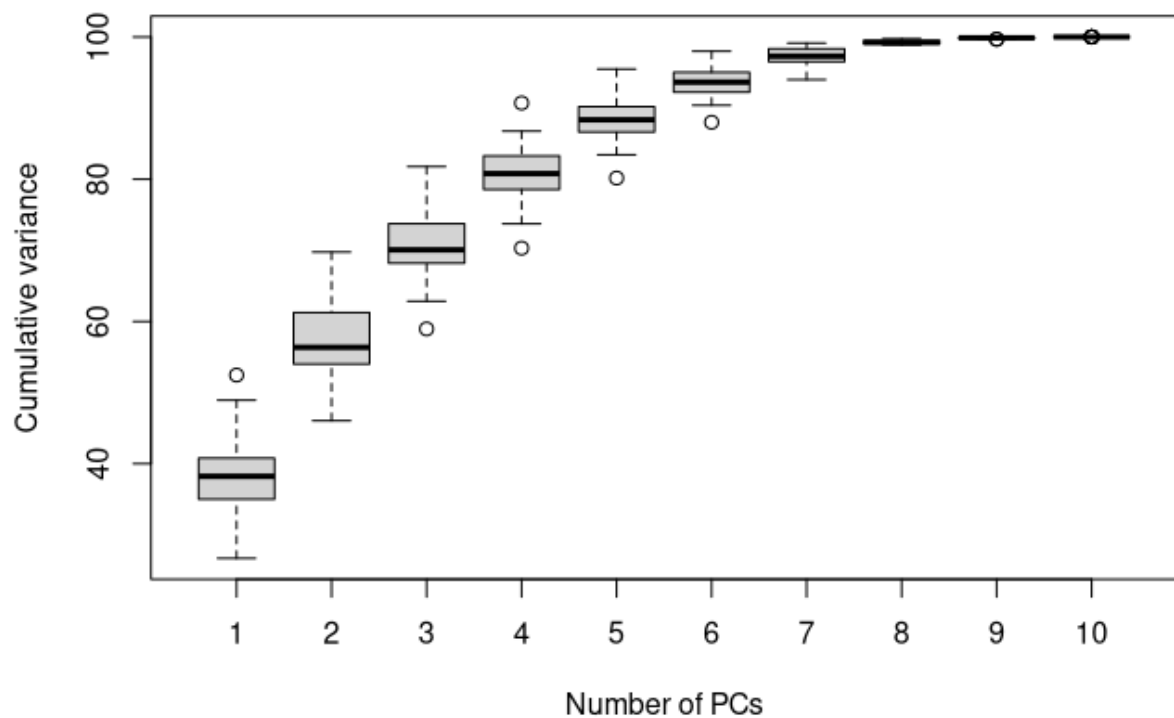
```
pcas<- lapply(bootstrap.samples, prcomp)
cumulative.variances <- vapply(pcas,
                             function(x) cumsum(prop.table(x[["sdev"]])
^2) * 100),
                             numeric(10L))
cumulative.variances <- as.vector(cumulative.variances)
all.variances <- data.frame(`Cumulative variance` = cumulative.variances
,
                             `Number of PCs` = 1:10, check.names = FALSE)
str(all.variances)
```

```
## 'data.frame':   340 obs. of  2 variables:
## $ Cumulative variance: num  32.9 54.3 69 79.5 86.3 ...
## $ Number of PCs      : int  1 2 3 4 5 6 7 8 9 10 ...
```

- f. [4 marks] Visualize your results showing how the cumulative variance explained increases with each additional component and indicate how the variability of the cumulative variance changes as well.

```
boxplot(`Cumulative variance` ~ `Number of PCs`, data = all.variances)
```





The variance explained with 1 component is quite low at 40% and has high variability as seen with the long first boxplot. The extra variance explained steadily increases at what appears to be a linear rate with similar variance (box size) with each additional component until around the 4th components. After which there are diminishing returns and only small increases (and small variability on the additional variance explained). Mainly since it has a hard boundary at 100%.