Exercise I: Principal Component Analysis

Recall the `mtcars` dataset we work with before, which compirses fuel consumption and other aspects of design and performance for 32 cars from 1974. The dataset has 11 dimensions, that is more than it is possible to visualize at the same.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.1
## v tidyr   1.1.1     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

a. Use `prcomp()` to compute a PCA for `mtcars`. Remember to set the scale parameter, as the variables are in different units and have different ranges

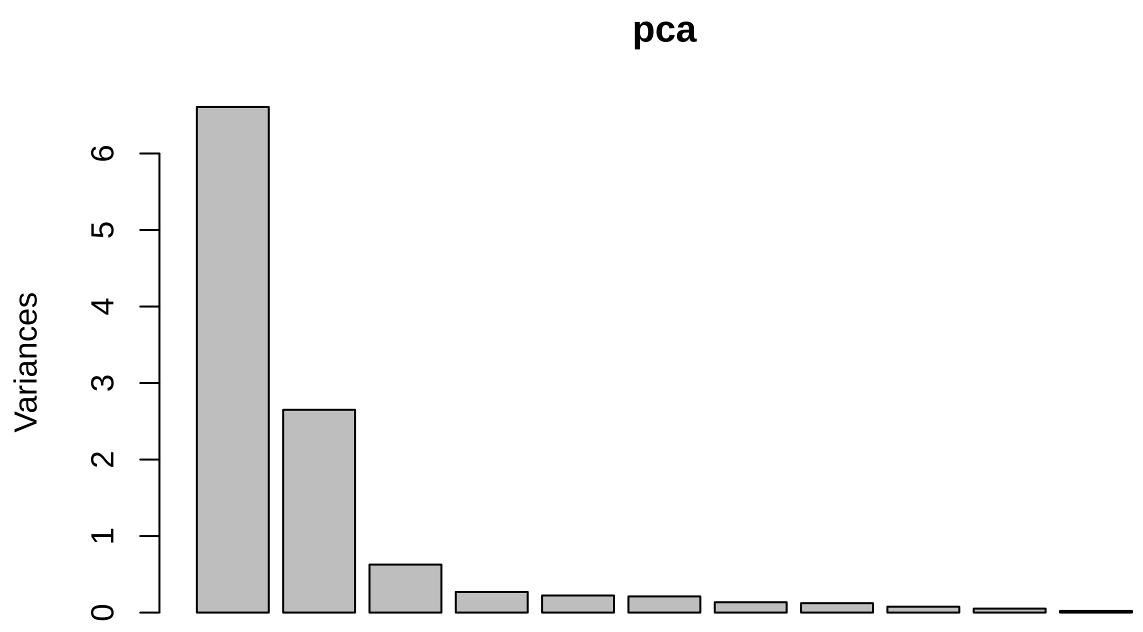
```
length(mtcars)
```

```
## [1] 11
```

```
pca <- prcomp(mtcars, scale. = TRUE)
```

b. Generate a scree plot and note how many dimensions should you retain.

```
screeplot(pca, npcs = 11)
```



```r
#2 dimensions
```

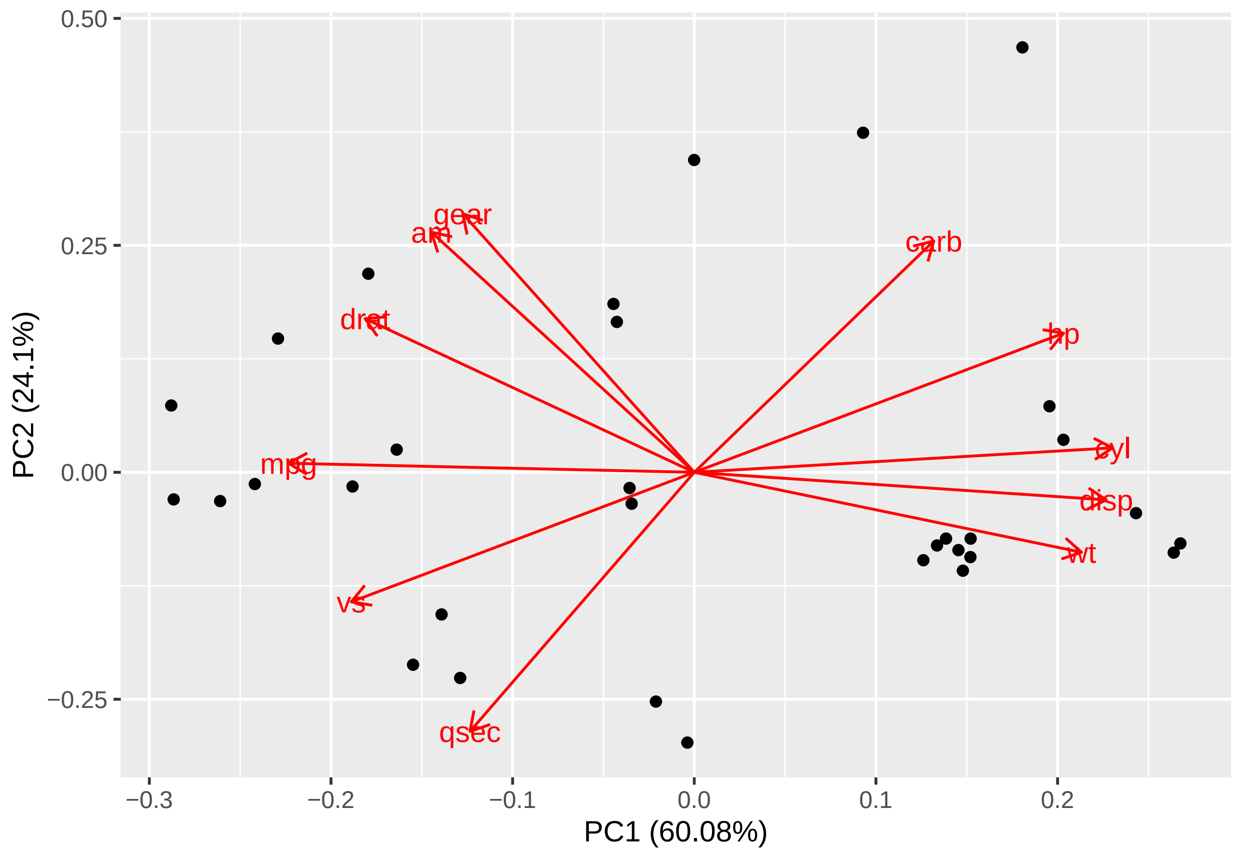
c. Compute the percentage of variance explained by each of the principal components.

```r
summary(pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6    PC7
## Standard deviation     2.5707 1.6280 0.79196 0.51923 0.47271 0.46000 0.3678
## Proportion of Variance 0.6008 0.2409 0.05702 0.02451 0.02031 0.01924 0.0123
## Cumulative Proportion  0.6008 0.8417 0.89873 0.92324 0.94356 0.96279 0.9751
##                            PC8    PC9    PC10   PC11
## Standard deviation     0.35057 0.2776 0.22811 0.1485
## Proportion of Variance 0.01117 0.0070 0.00473 0.0020
## Cumulative Proportion  0.98626 0.9933 0.99800 1.0000
```

d. Generate a biplot for the PCA projection. Use the loadings matrix to inspect which variables contributes most to PC1 and which to PC2. What do the PC1 and PC2 correspond to? How are the cars distributed on this representation? Does the "car map" make sense?

```r
library(ggfortify)

autoplot(pca, loadings = TRUE, loadings.label = TRUE)
```



```
pca$rotation
```

```
##              PC1         PC2         PC3          PC4         PC5         PC6
## mpg  -0.3625305  0.01612440 -0.22574419 -0.022540255  0.10284468 -0.10879743
## cyl   0.3739160  0.04374371 -0.17531118 -0.002591838  0.05848381  0.16855369
## disp  0.3681852 -0.04932413 -0.06148414  0.256607885  0.39399530 -0.33616451
## hp    0.3300569  0.24878402  0.14001476 -0.067676157  0.54004744  0.07143563
## drat -0.2941514  0.27469408  0.16118879  0.854828743  0.07732727  0.24449705
## wt    0.3461033 -0.14303825  0.34181851  0.245899314 -0.07502912 -0.46493964
## qsec -0.2004563 -0.46337482  0.40316904  0.068076532 -0.16466591 -0.33048032
## vs   -0.3065113 -0.23164699  0.42881517 -0.214848616  0.59953955  0.19401702
## am   -0.2349429  0.42941765 -0.20576657 -0.030462908  0.08978128 -0.57081745
## gear -0.2069162  0.46234863  0.28977993 -0.264690521  0.04832960 -0.24356284
## carb  0.2140177  0.41357106  0.52854459 -0.126789179 -0.36131875  0.18352168
##               PC7          PC8          PC9        PC10         PC11
## mpg   0.367723810 -0.754091423  0.235701617  0.13928524 -0.124895628
## cyl   0.057277736 -0.230824925  0.054035270 -0.84641949 -0.140695441
## disp  0.214303077  0.001142134  0.198427848  0.04937979  0.660606481
## hp   -0.001495989 -0.222358441 -0.575830072  0.24782351 -0.256492062
## drat  0.021119857  0.032193501 -0.046901228 -0.10149369 -0.039530246
## wt   -0.020668302 -0.008571929  0.359498251  0.09439426 -0.567448697
## qsec  0.050010522 -0.231840021 -0.528377185 -0.27067295  0.181361780
## vs   -0.265780836  0.025935128  0.358582624 -0.15903909  0.008414634
## am   -0.587305101 -0.059746952 -0.047403982 -0.17778541  0.029823537
## gear  0.605097617  0.336150240 -0.001735039 -0.21382515 -0.053507085
## carb -0.174603192 -0.395629107  0.170640677  0.07225950  0.319594676
```

```
#cyl is biggest for PC1
#qsec is biggest for PC2
```

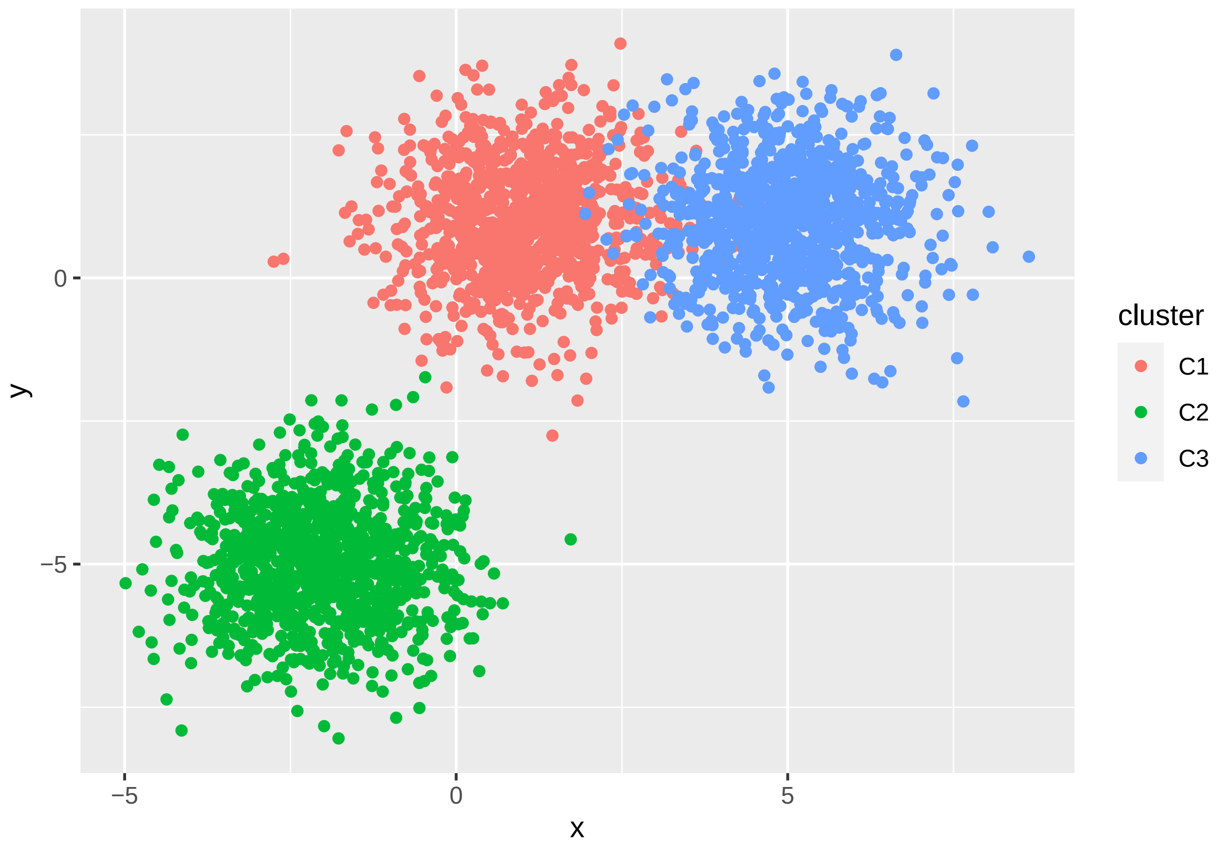
Exercise 2: Cluster Analysis

Part 1: k-means clustering

We will generate synthetic clustered data to use for k-means clustering.

```
set.seed(489576)
N <- 1000
C1 <- data.frame(cluster = "C1", x = rnorm(n = N, mean = 1), y = rnorm(n = N, mean = 1))
C2 <- data.frame(cluster = "C2", x = rnorm(n = N, mean = -2), y = rnorm(n = N, mean = -5))
C3 <- data.frame(cluster = "C3", x = rnorm(n = N, mean = 5), y = rnorm(n = N, mean = 1))
DF <- rbind(C1, C2, C3)
```

```
ggplot(DF, aes(x, y, color = cluster)) +
  geom_point()
```



a. Apply k-means with k = 3 (as you know the true number of clusters). Print the cluster centers.

```r
library(SingleCellExperiment)
```

```
## Loading required package: SummarizedExperiment
```

```
## Loading required package: GenomicRanges
```

```
## Loading required package: stats4
```

```
## Loading required package: BiocGenerics
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
```

```
## The following objects are masked from 'package:dplyr':
##
##     combine, intersect, setdiff, union
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which, which.max, which.min
```

```
## Loading required package: S4Vectors
```

```
##
## Attaching package: 'S4Vectors'
```

```
## The following objects are masked from 'package:dplyr':
##
##     first, rename
```

```
## The following object is masked from 'package:tidyr':
##
##     expand
## The following object is masked from 'package:base':
##
##     expand.grid

## Loading required package: IRanges

##
## Attaching package: 'IRanges'

## The following objects are masked from 'package:dplyr':
##
##     collapse, desc, slice

## The following object is masked from 'package:purrr':
##
##     reduce

## Loading required package: GenomeInfoDb

## Loading required package: Biobase

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

## Loading required package: DelayedArray

## Loading required package: matrixStats

##
## Attaching package: 'matrixStats'

## The following objects are masked from 'package:Biobase':
##
##     anyMissing, rowMedians

## The following object is masked from 'package:dplyr':
##
##     count

##
## Attaching package: 'DelayedArray'

## The following objects are masked from 'package:matrixStats':
##
##     colMaxs, colMins, colRanges, rowMaxs, rowMins, rowRanges
## The following object is masked from 'package:purrr':
##
##     simplify

## The following objects are masked from 'package:base':
##
##     aperm, apply, rowsum
```

```
library(dplyr)
library(tidyverse)

clust.kmeans <- kmeans(DF %>%select(x,y), centers=3)
```

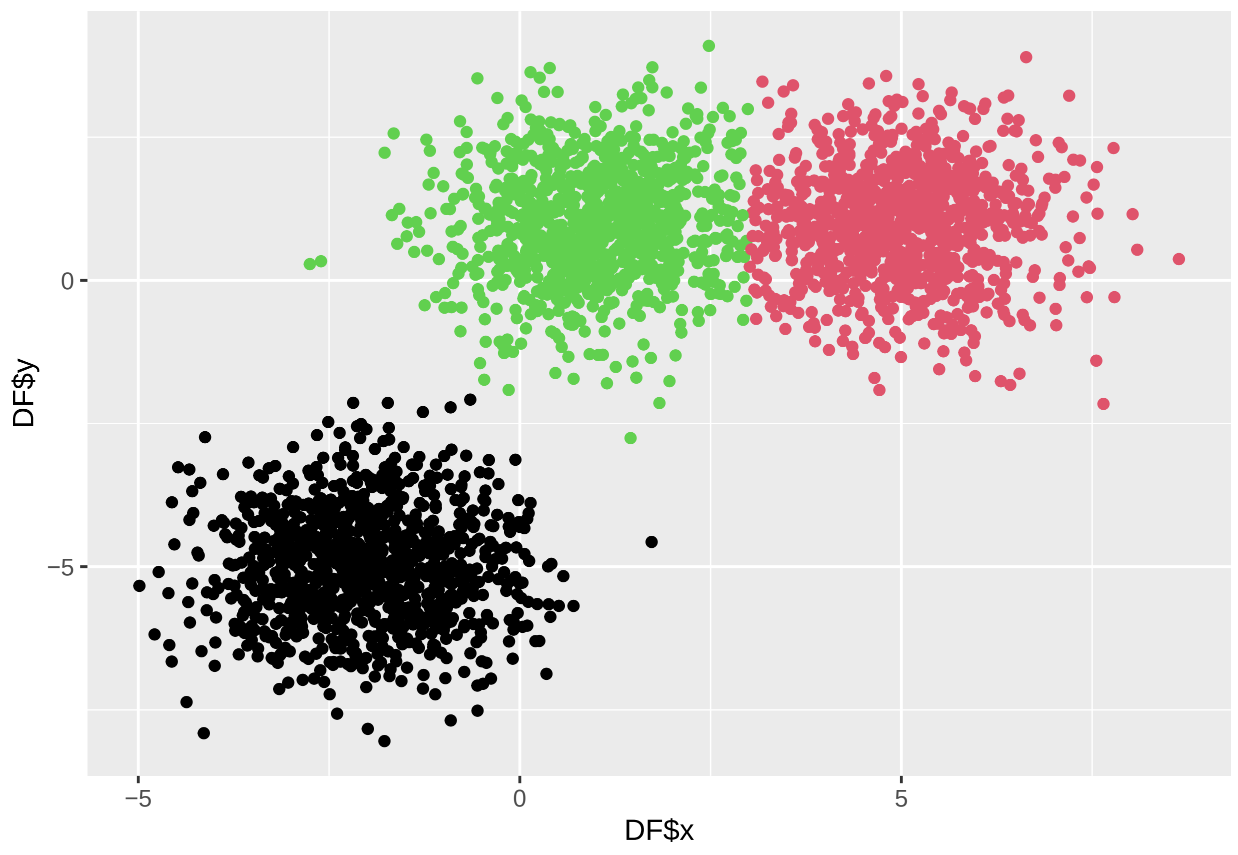
b. Print a confusion map to compare k-means cluster assignment with the true cluster labels.

```
table(DF[,1],clust.kmeans$cluster)
```

```
##
##         1   2   3
##   C1    0  32 968
##   C2  999   0   1
##   C3    0 977  23
```

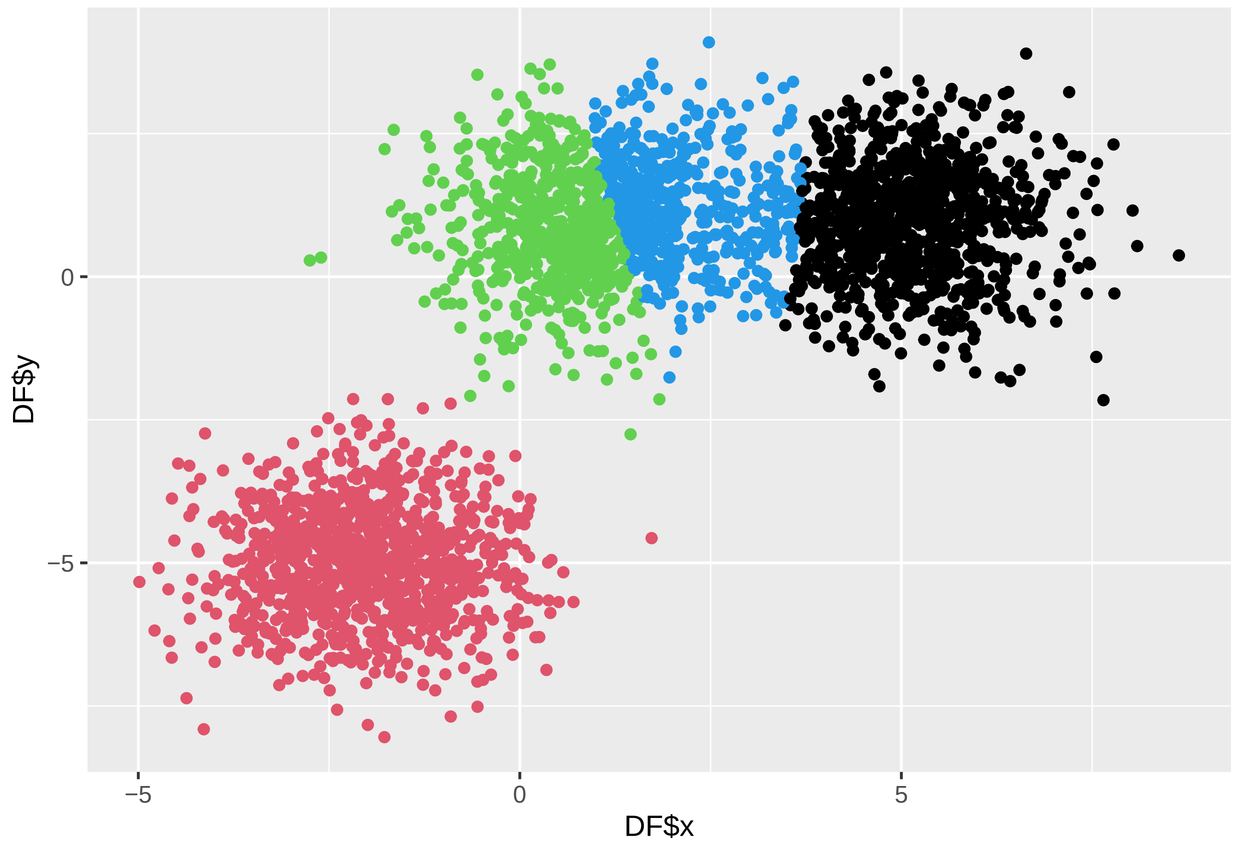
c. Generate a scatter plot of points, now colored by the cluster assignment.

```
ggplot() + geom_point(aes(x=DF$x,y=DF$y), color=clust.kmeans$cluster)
```



d. Now pretend that you don't know the real number of clusters. Use k = 4 and recompute kmeans. Plot the results and see what happened.

```
clust.kmeans2 <- kmeans(DF %>%select(x,y), centers=4)
ggplot() + geom_point(aes(x=DF$x,y=DF$y), color=clust.kmeans2$cluster)
```



```
#every time you run it, the groups are different. but with k=3, answers were consistent
```

e. Still Pretending that you do not know the real number of clusters, how would you select it? USe your favorite method (elbow, gap statistic or silhouette) to find a reasonable estimate of the number of clusters

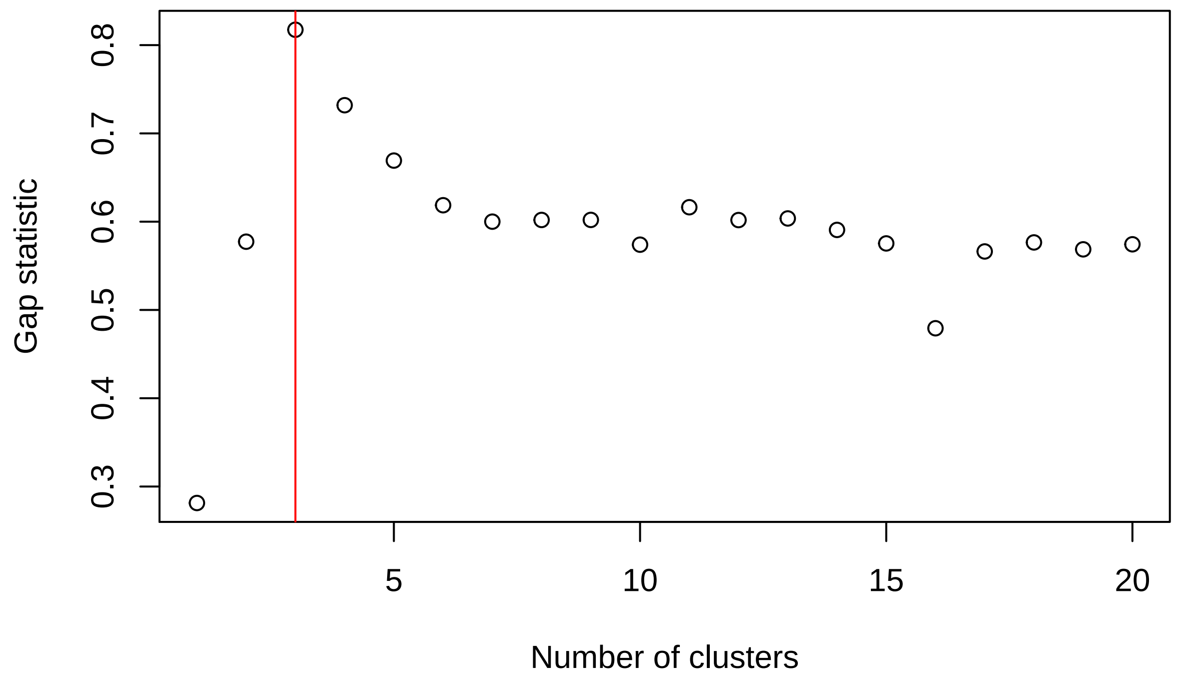
```
library(cluster)
set.seed(110010101)
gaps <- clusGap(DF %>%select(x,y), kmeans, K.max=20)
```

```
## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations
## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations
```

```
best.k <- maxSE(gaps$Tab[,"gap"], gaps$Tab[,"SE.sim"])
best.k
```

```
## [1] 3
```

```
plot(gaps$Tab[,"gap"], xlab="Number of clusters", ylab="Gap statistic")
abline(v=best.k, col="red")
```



Part 2: Hierarchical Clustering

In this exercise you will you use a dataset published in a study by Khan et al. 2001 to perform a hierarchical clustering of the patients in the study based on their overall gene expression data.

This data set consists of expression levels for 2,308 genes. The training and test sets consist of 63 and 20 observations (tissue samples) respectively.

Here, we will use the train set, as we now are only interested in learning how `hclust()` works. First, load the `ISLR` where the data is available. The gene expression data is available in an object `Khan$xtrain`; you can learn more about the data set by typing in `?Khan` after loading `ISLR` package.

```r
library(ISLR)
gene.expression <- Khan$xtrain
dim(gene.expression)
```

```
## [1]   63 2308
```

a. Compute a (Euclidean) distance matrix between each pair of samples.

```r
dist(gene.expression, method = "euclidean", diag=FALSE,upper=FALSE,p=2)
```

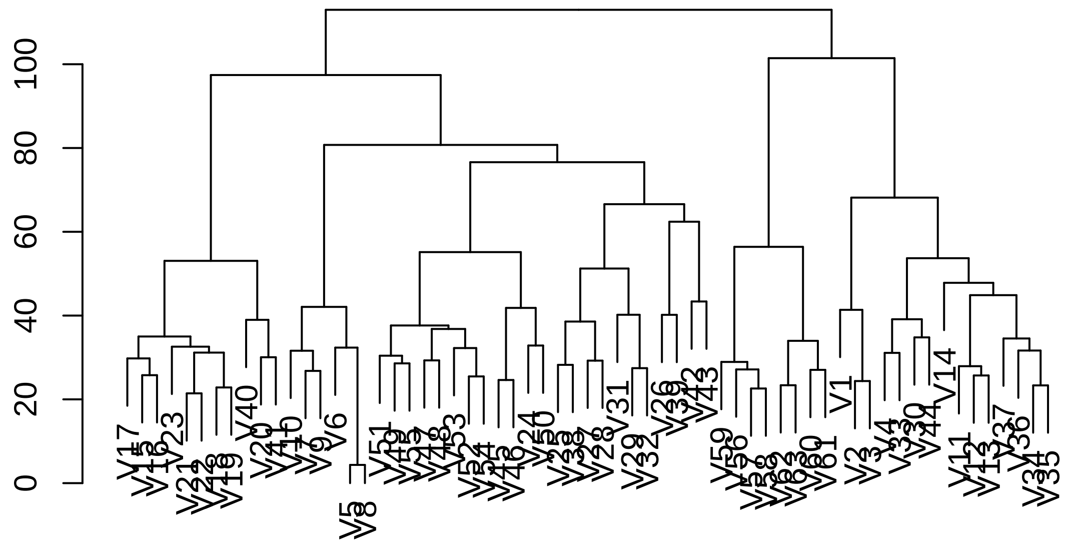
```
##            V1        V2        V3        V4        V5        V6        V7
## V2  36.689456
## V3  38.959648 24.367679
## V4  43.363067 42.798538 42.406602
## V5  49.730456 42.463557 40.216370 39.843152
## V6  51.770434 44.111827 41.207970 41.152773 28.373861
## V7  58.539300 50.954328 49.017453 53.200114 33.090408 33.748673
## V8  49.023305 42.266384 39.922326 40.677311  4.351971 27.832308 32.016982
## V9  55.008019 50.710981 47.924663 52.970791 32.386369 36.552399 26.790130
## V10 49.795459 45.073964 42.926095 46.411239 28.096500 36.488628 32.889637
## V11 44.777871 36.633088 33.906156 34.793082 37.958136 37.644680 47.422956
## V12 43.638369 38.495903 35.958327 31.210814 40.288934 41.501152 50.699603
## V13 49.243974 44.447572 41.620859 33.339654 41.896748 43.224402 51.509854
## V14 56.566989 46.866088 47.832903 47.174983 52.133516 48.234530 56.666786
## V15 58.868908 49.895560 46.475676 38.962431 41.220299 40.883814 50.950384
## V16 55.804382 48.741230 45.399980 35.877881 35.901667 37.340074 47.962724
## V17 57.989554 47.887709 44.762077 37.858301 34.872659 36.284529 43.858773
## V18 54.483855 46.306355 43.130562 37.471074 35.543318 37.452505 45.737775
## V19 50.386367 43.609257 41.750843 41.027841 37.497829 37.752354 44.566585
## V20 66.267283 58.942762 55.022281 48.642281 47.134469 45.957339 51.414036
## V21 57.148917 50.403648 47.251083 42.173953 37.308444 34.882114 44.578871
## V22 55.021539 47.406506 44.902917 41.853648 35.850142 34.391814 41.901381
## V23 55.877577 51.800223 48.356835 37.779777 37.963100 39.170097 49.246605
## V24 57.195904 52.138128 49.773416 46.697463 36.105802 37.912053 41.779539
## V25 58.995714 50.255691 47.110870 39.235891 36.114082 36.791076 47.902361
## V26 60.120024 59.076929 56.907436 49.690066 50.014071 48.177843 58.125343
## V27 55.463219 48.097071 45.980971 42.811288 35.884221 32.610694 45.525874
## V28 55.330193 46.991504 45.634404 41.038028 37.659004 37.007062 47.310773
## V29 54.194376 46.539160 44.428039 44.737408 37.285990 33.841817 42.675409
## V30 51.547183 41.203722 40.448663 39.852080 45.902779 44.865476 54.029802
## V31 63.718309 57.765146 55.175015 54.983404 40.407724 39.382787 39.129900
## V32 56.640713 53.031977 51.705993 53.332825 45.293397 40.428880 48.573759
## V33 50.244856 42.633250 42.003323 31.088011 44.349034 43.181258 54.601315
## V34 51.624017 40.852193 38.493361 42.668913 46.341218 46.986700 54.450322
## V35 50.329043 43.965047 41.397553 38.154447 45.600989 47.621743 54.704193
## V36 47.002668 46.024696 45.181681 33.403079 45.258366 46.736696 54.823979
## V37 55.059177 48.078731 44.736426 39.137782 46.345137 44.727430 53.149854
## V38 61.763124 51.819660 48.536336 42.993378 40.510611 38.693698 47.973817
## V39 65.127987 64.727898 61.211994 51.134702 54.698419 53.202193 64.100300
## V40 63.601132 59.744112 57.046226 49.544970 50.350958 49.109921 57.236206
## V41 60.040481 53.098939 48.970959 43.776534 43.414927 42.575830 51.052235
## V42 59.351946 55.195277 51.239715 48.079444 45.373819 43.273729 51.057918
## V43 63.264938 56.511337 52.886997 56.355967 52.525854 49.443590 53.844711
## V44 43.669607 39.678146 38.553551 36.262615 41.901514 40.906206 50.564362
## V45 50.413321 38.594328 37.840620 39.100075 40.772667 39.165377 48.311887
## V46 48.375277 37.873557 36.407036 39.336717 37.462608 36.412483 45.002400
## V47 48.478890 43.322592 41.910831 37.727413 35.881812 35.353088 48.094209
## V48 55.603425 48.821332 47.413420 44.387745 37.538356 30.218599 42.815279
## V49 50.237047 47.786377 45.984865 41.868344 37.844823 34.470117 48.900594
## V50 54.829003 48.203875 46.622486 42.072648 41.024971 38.162572 49.391321
## V51 51.722812 48.024395 47.031722 46.364375 40.569915 35.729831 47.017918
## V52 51.091557 43.644944 43.968385 42.822918 37.964190 34.182735 45.541698
## V53 53.005474 44.418488 42.506673 41.779956 37.955845 34.233116 45.902536
## V54 51.347000 46.753976 45.635757 45.840833 38.972884 35.800751 46.337326
## V55 53.298302 48.295912 46.509598 46.099237 39.757437 31.695638 45.615547
## V56 45.084616 40.592224 40.226342 44.860272 46.805269 46.905388 54.323656
## V57 46.651860 40.791863 39.173126 44.064371 43.086952 44.329100 49.725725
## V58 46.224664 39.325788 38.511430 42.198523 42.432877 44.894320 50.426664
## V59 47.537950 36.699781 36.050564 40.814267 42.143023 41.825639 50.713293
## V60 58.621205 51.657186 49.494826 44.675760 40.690922 38.087803 48.313534
## V61 54.754949 47.120188 45.666872 44.846951 38.945147 36.364780 45.866059
## V62 55.897737 48.420070 47.532304 48.790414 40.993527 40.350732 46.498252
## V63 56.176669 47.529654 47.887113 50.295684 41.245431 39.284104 43.935595
##            V8        V9       V10       V11       V12       V13       V14
## V2
## V3
## V4
## V5
## V6
## V7
## V8
## V9  31.076685
## V10 27.433144 27.854263
## V11 38.055006 47.496988 43.635057
## V12 40.718347 51.298958 44.989143 27.709740
## V13 42.326248 51.828868 46.315717 27.044984 25.700382
## V14 52.338387 59.756534 56.116175 41.476331 41.507255 39.404976
## V15 41.922011 53.085819 46.947267 38.057906 40.479836 35.327537 44.348734
## V16 36.571143 49.133153 43.681580 33.221276 37.262037 32.247996 45.344613
## V17 35.854442 48.086274 42.606771 35.809594 35.959029 32.913809 44.982024
## V18 36.037714 47.362566 41.926713 32.197076 34.752311 30.258804 42.872628
## V19 37.316541 45.542192 40.911597 33.164948 37.717286 35.079748 46.227449
## V20 47.299176 54.293453 51.842785 43.985447 47.523324 39.744908 49.654477
## V21 37.204932 46.307721 44.330183 35.884876 41.955628 36.357226 48.405282
## V22 35.826887 44.603806 41.946600 35.028943 39.364001 35.685379 44.277714
## V23 38.340410 50.293369 45.075560 38.145264 36.603259 34.996543 49.350385
## V24 35.663869 40.223625 38.624151 44.235088 45.799971 44.590795 51.489049
## V25 36.964581 48.886428 43.459670 41.735451 41.388506 40.525182 50.280853
## V26 49.850849 56.766554 54.300994 50.962567 52.174826 51.842994 62.104889
## V27 36.077108 44.886181 42.150945 42.095545 44.386722 45.517089 51.272329
## V28 38.338082 46.530694 42.494199 41.584146 42.564719 43.481703 50.273430
## V29 37.193620 41.354101 40.528607 43.147796 44.666151 46.058235 50.343478
## V30 46.397510 55.416865 50.872641 41.997738 40.887223 42.933757 47.461417
## V31 39.606341 37.667780 39.762885 51.469601 53.884847 52.028468 57.496217
## V32 44.717812 46.029238 47.379832 50.204875 52.420743 54.218612 53.817769
## V33 44.946129 55.355464 50.182801 35.286238 36.312270 34.297406 44.689665
## V34 46.676848 55.655441 49.284586 34.258567 36.117577 36.414121 40.014463
## V35 46.086538 55.612326 48.758271 35.954054 34.232704 32.001361 43.083465
## V36 45.524003 54.686104 48.085537 37.764271 33.162947 31.766412 41.898894
## V37 46.755421 54.822964 51.766798 32.916372 33.558125 31.572918 45.088353
## V38 41.213166 51.074137 47.199435 41.574553 41.089056 39.229659 45.331732
## V39 54.613985 63.282994 60.101634 51.838003 51.623688 50.218425 60.330169
## V40 50.230034 57.500960 55.350666 46.804850 50.406439 45.185878 51.634186
## V41 43.587023 51.880039 48.597123 40.929787 43.410595 38.325195 49.413921
## V42 45.141880 52.695063 50.886247 43.971759 45.544325 42.807581 54.084452
## V43 52.018347 56.604378 56.577471 50.543449 51.826999 50.907664 56.375483
## V44 42.009719 50.335678 45.717989 39.018489 37.050544 40.558641 49.697572
## V45 41.211444 49.782979 45.564263 36.452843 36.377012 39.695144 42.102983
## V46 37.714046 45.795209 41.642281 34.676104 35.139710 38.276632 41.150885
## V47 36.240003 46.714083 39.126864 40.135696 40.229762 42.501072 51.219936
## V48 37.567984 43.906070 43.131152 44.252678 46.496169 47.951776 51.382479
## V49 37.661937 45.735253 41.497961 41.868114 43.657651 46.170912 51.296076
## V50 41.046119 49.345144 45.577920 41.918427 43.311832 44.293911 48.240015
## V51 40.046851 45.680992 44.687384 45.700766 47.877132 50.581180 51.041389
## V52 37.949365 44.585045 42.644008 40.563011 42.412205 45.586264 45.489985
## V53 38.298183 45.978067 41.895614 40.956926 43.236342 44.967846 50.193313
## V54 38.593642 44.046632 41.136177 42.467116 45.551832 47.270197 52.106786
## V55 39.387646 44.683272 45.514959 44.304030 47.761451 50.826670 52.786641
## V56 46.405552 51.821284 48.808673 39.992677 41.499720 47.140122 53.125182
## V57 42.722004 47.950092 45.612235 38.404661 40.813065 43.706593 53.890669
## V58 42.411702 49.438276 45.503934 36.719671 39.466312 43.032084 51.992535
## V59 42.306933 50.983921 45.663557 37.330784 39.604336 43.324197 47.641733
## V60 40.825197 49.454477 47.122449 44.733121 47.373449 48.867915 54.206147
## V61 38.970807 46.196948 43.659401 43.421767 46.252057 48.807499 52.915683
## V62 40.982363 47.102312 44.668312 47.325718 49.292723 52.319826 54.434908
## V63 41.000749 46.507672 46.151492 47.511672 50.382164 53.608358 53.311487
##           V15       V16       V17       V18       V19       V20       V21
## V2
## V3
## V4
## V5
## V6
## V7
## V8
## V9
## V10
## V11
## V12
## V13
## V14
## V15
## V16 25.770379
## V17 29.992525 27.601434
## V18 29.349768 24.271763 25.635777
## V19 33.588603 29.033420 32.969572 22.836965
## V20 33.446945 34.375324 34.460143 34.839088 39.056299
## V21 32.059855 25.821113 30.593126 27.252765 27.741804 33.125839
## V22 32.010751 27.011062 28.370987 26.545358 26.568201 34.649645 21.430855
## V23 34.790817 29.031445 29.790680 25.314971 32.983595 36.508158 29.580527
## V24 44.438165 41.382025 42.164539 39.362457 42.501941 43.382717 42.487070
## V25 33.588141 32.658189 29.642873 32.921641 38.942882 39.405680 35.750598
## V26 53.070130 47.284259 49.180715 48.056059 50.835396 53.050991 48.699769
## V27 42.070171 39.462628 41.495414 39.119438 41.955406 47.503769 40.594799
## V28 41.541679 38.503170 39.010898 39.187063 42.083843 47.340781 42.156369
## V29 44.069746 42.666830 41.944517 40.630359 41.334650 48.725895 41.891326
## V30 46.394097 47.470633 41.978229 46.557850 48.549274 52.971789 49.636161
## V31 50.388224 48.322304 47.267361 45.814937 47.783790 49.435676 44.699020
## V32 52.634614 50.867740 52.388928 48.272734 47.837067 55.542805 48.432913
## V33 38.388411 38.643507 38.263791 39.587014 43.035122 44.658527 42.946296
## V34 40.339311 42.519236 39.781387 38.907810 42.196119 45.429153 46.166618
## V35 40.113420 41.819547 37.629531 38.043385 41.894415 44.304733 45.811870
## V36 42.279287 41.621816 39.492127 38.838103 43.057042 46.201884 45.816575
## V37 42.631492 41.331487 37.693731 38.912142 44.336778 41.552376 44.215418
## V38 35.509468 35.924743 29.323865 32.533543 39.341464 35.289031 36.799679
## V39 50.713723 45.560110 49.205285 46.768085 52.008378 48.088020 48.270099
## V40 43.667132 40.065711 44.832248 40.500870 44.872180 37.543752 41.904510
## V41 34.333898 33.693099 32.821141 33.855378 38.072673 30.054791 34.617254
## V42 45.573124 42.017431 41.950865 41.290338 44.864862 42.796717 41.774864
## V43 53.510333 52.168061 48.298741 50.111132 51.111335 49.062816 49.643911
## V44 47.356933 45.105468 42.203122 44.722233 46.011913 53.592724 46.785701
## V45 42.261010 42.466174 39.978801 40.294073 43.220841 48.263847 45.592402
## V46 41.924187 40.736814 37.883593 37.376365 40.674620 46.550141 43.479234
## V47 40.527724 38.184470 38.719390 38.438313 40.542778 49.358423 42.798548
## V48 43.776392 41.711258 41.592620 41.326586 42.957303 49.545218 40.351454
## V49 45.705786 42.041262 45.167097 41.232731 42.987028 50.423714 44.072568
## V50 43.461275 40.727814 43.571182 40.405153 42.826818 46.332467 44.780377
## V51 49.810207 47.277447 47.951166 45.541675 46.319846 54.958085 46.806667
## V52 45.134956 42.178668 43.240301 40.269832 41.878780 50.542662 44.177620
## V53 42.900919 41.256219 40.589377 40.921496 42.515327 49.069624 43.444450
## V54 46.393590 43.566012 45.716172 42.261285 41.825854 50.302238 43.478407
## V55 49.412094 46.726750 48.335758 46.919924 46.825906 54.466454 47.533062
## V56 54.503205 50.976833 52.546717 49.243581 47.968729 59.706483 52.160740
## V57 51.973491 47.819383 47.478287 47.339211 45.885692 54.453853 47.989599
## V58 49.565176 45.538063 46.177923 44.916098 44.125551 55.066278 46.790234
## V59 45.958013 44.068481 42.769303 44.037931 44.271135 53.270496 45.809513
## V60 46.902947 42.115127 44.948249 44.185483 46.099898 50.879674 43.026137
## V61 45.750980 42.359079 44.400153 43.413317 43.371017 52.018900 42.591460
## V62 50.174611 46.870931 46.759978 46.348616 45.980907 56.111901 46.332643
## V63 51.636614 48.224174 47.560276 47.001160 46.335138 56.710053 46.569409
##           V22       V23       V24       V25       V26       V27       V28
## V2
## V3
## V4
## V5
## V6
## V7
## V8
## V9
## V10
## V11
## V12
## V13
## V14
## V15
## V16
## V17
## V18
## V19
## V20
## V21
## V22
## V23 32.009852
## V24 42.267540 38.778381
## V25 35.941871 32.212952 37.404629
## V26 49.699754 43.559688 45.169657 43.779954
## V27 40.127485 40.910986 38.484412 29.690828 44.677623
## V28 40.794106 42.421208 39.979219 32.842420 38.143697 29.242648
## V29 40.911266 43.137821 36.278359 32.624179 46.554871 29.726321 33.678299
## V30 46.895867 50.109805 49.538391 41.742483 56.301896 44.055407 42.651351
## V31 44.278855 45.459882 35.863239 37.305850 52.414994 38.846143 45.017576
## V32 47.349209 49.087228 38.064455 43.113152 48.583448 37.066237 41.980497
## V33 42.311150 41.892201 43.039762 37.166418 50.823932 41.881459 39.904660
## V34 42.863585 43.839514 46.804312 38.961896 51.638988 41.176989 40.731146
## V35 43.064217 42.391794 47.230462 38.363192 51.831699 43.436507 41.318499
## V36 43.853573 40.024468 43.134841 39.643199 50.229790 43.997872 44.014038
## V37 43.430177 41.103137 43.553789 39.807933 50.990962 43.460160 43.300770
## V38 34.635124 35.107332 40.788521 28.237330 49.041121 35.684956 37.344342
## V39 49.944725 40.617757 44.762771 46.731771 40.175387 50.786420 51.805492
## V40 42.868948 40.019386 41.863000 43.359253 46.707057 46.591812 46.830982
## V41 34.608894 35.298968 39.220682 33.686347 46.661377 41.192911 41.080966
## V42 42.397111 41.458391 43.836072 40.361212 50.723504 41.969277 45.348694
## V43 49.632641 52.024715 51.820015 49.510353 57.744533 50.800090 52.470441
## V44 44.318621 45.785637 49.650567 42.670697 51.020932 41.516917 40.007046
## V45 42.614786 43.446710 40.593200 39.730096 52.275538 40.874501 39.210172
## V46 39.564806 40.206441 38.024563 37.026741 48.877576 36.787117 37.038220
## V47 40.391853 39.582937 41.453479 36.770155 45.684014 35.577576 34.490291
## V48 38.755324 41.909927 42.812690 38.142619 51.201547 34.365440 38.472747
## V49 43.402065 40.120271 34.879501 40.393317 44.259368 35.298193 38.050880
## V50 44.145047 41.216532 32.847438 41.616085 47.490535 41.036583 40.537552
## V51 45.433639 45.059267 40.859873 43.712831 49.144391 37.769704 42.157952
## V52 41.199065 42.991502 37.311862 41.989862 48.904265 37.163846 36.587570
## V53 42.225582 44.540149 41.360395 39.457418 49.496158 36.920705 35.325197
## V54 42.136768 43.422893 36.225635 43.208768 49.904670 39.379373 40.091483
## V55 45.837592 48.755878 41.273785 45.047359 48.887052 35.242152 38.845295
## V56 51.706137 51.391570 50.536791 50.663558 53.905453 46.420408 47.009317
## V57 47.969828 49.826076 49.404598 48.186902 52.872667 45.915557 44.247700
## V58 46.310270 47.899117 49.574474 46.518748 53.157311 44.557704 43.315851
## V59 44.247764 46.942563 49.166787 43.307818 53.479116 42.954782 42.398378
## V60 44.293423 44.416607 44.069816 42.971095 49.085153 39.435823 40.759548
## V61 42.685230 45.468341 45.194289 43.181842 51.956717 39.266062 40.281761
## V62 45.130673 48.710184 47.436374 45.678680 53.798429 43.043718 43.346787
## V63 44.556117 49.746699 48.072748 47.042266 55.049240 43.720682 44.833659
##           V29       V30       V31       V32       V33       V34       V35
## V2
## V3
## V4
## V5
## V6
## V7
## V8
## V9
## V10
## V11
## V12
## V13
## V14
## V15
## V16
## V17
## V18
## V19
## V20
## V21
## V22
## V23
## V24
## V25
## V26
## V27
## V28
## V29
## V30 43.226424
## V31 36.209149 54.463941
## V32 27.448326 53.393700 38.594316
## V33 42.189568 31.132098 52.421391 51.705562
## V34 42.173465 37.585953 51.525311 50.254457 36.118550
## V35 44.583691 38.242579 52.025271 54.196001 34.269264 23.329759
## V36 44.476449 40.981898 51.392259 52.185466 33.679791 32.868133 26.357647
## V37 44.664002 39.546419 51.771887 53.126162 32.417036 32.787420 31.490680
## V38 37.800742 42.455177 43.943508 47.645201 38.818842 37.161877 37.084835
## V39 53.476631 59.094691 57.729665 55.073841 50.622521 53.250057 54.566508
## V40 49.505229 56.847132 51.639536 52.845714 46.623945 47.905741 47.624988
## V41 41.618210 44.439446 47.249358 50.440484 36.948824 39.281941 38.939161
## V42 44.157210 46.817385 47.667332 51.687367 42.766060 45.081999 45.082944
## V43 50.391506 50.750623 53.821859 56.130516 50.024785 49.808372 51.045674
## V44 43.111693 34.793947 53.402209 51.298171 36.947079 40.830064 39.909911
## V45 39.518314 38.541983 49.484151 46.018792 36.029467 38.244139 39.930111
## V46 36.819019 39.491191 44.804970 42.672898 36.112890 35.701315 37.912834
## V47 37.318402 42.791558 48.493488 45.845717 39.689117 42.859151 42.985821
## V48 35.138878 47.381786 41.975224 41.640049 44.577854 48.596479 49.690510
## V49 37.838374 49.969291 45.900596 39.040474 44.985044 48.131772 49.536786
## V50 39.414298 48.536032 48.710485 43.012800 42.433869 47.178206 47.801986
## V51 38.197227 50.823855 45.470357 38.430340 48.080538 49.395865 52.105070
## V52 36.524093 47.139181 47.428651 37.596307 43.968527 45.932284 48.414362
## V53 36.882814 42.457432 49.000901 44.552776 40.276200 44.565504 45.650199
## V54 38.130281 49.580548 46.510719 39.426095 45.971318 49.244900 50.875810
## V55 37.882587 49.156249 48.268261 40.750064 47.239740 50.828644 52.962303
## V56 46.791975 49.555472 56.799438 50.986370 47.607283 46.080464 47.476634
## V57 45.906538 45.240071 54.024181 52.781238 44.664511 45.111479 44.029729
## V58 44.866876 44.153329 54.454554 51.627683 43.419803 42.557313 43.375859
## V59 43.173614 39.564733 52.936256 50.483943 40.192254 38.754439 41.632872
## V60 43.038331 51.213040 50.725919 48.388837 47.153676 50.612873 51.465600
## V61 39.186278 49.325767 48.697153 45.683355 46.966826 49.005614 50.646505
## V62 41.422635 51.653979 50.053589 46.813304 50.974084 50.303206 53.049762
## V63 41.703925 52.189274 49.647294 45.358135 52.475653 50.983205 54.269820
##           V36       V37       V38       V39       V40       V41       V42
## V2
## V3
## V4
## V5
## V6
## V7
## V8
## V9
## V10
## V11
## V12
## V13
## V14
## V15
## V16
## V17
## V18
## V19
## V20
## V21
## V22
## V23
## V24
## V25
## V26
## V27
## V28
## V29
## V30
## V31
## V32
## V33
## V34
## V35
## V36
## V37 33.031193
## V38 40.132010 36.064305
## V39 48.825391 48.030196 48.342404
## V40 45.211776 43.326944 42.774546 43.602753
## V41 39.762158 34.557622 31.480517 41.055705 36.345011
## V42 44.525707 39.985427 39.750671 46.035490 46.261199 36.182078
## V43 51.156860 45.914328 46.554231 57.067290 52.922602 44.824904 43.364699
## V44 39.980966 41.424044 43.715707 56.431967 54.904975 45.795147 46.643072
## V45 40.091914 39.306676 38.974143 54.702694 49.864537 44.096815 47.877451
## V46 37.676573 37.257365 36.775834 51.614636 46.474924 41.909797 44.576467
## V47 42.624681 45.077584 40.991667 50.162160 49.790363 42.091627 45.048001
## V48 48.404398 48.218923 40.736829 56.180771 51.514444 44.772121 46.875235
## V49 46.039865 48.680202 44.783828 47.390214 47.134558 45.940295 48.159820
## V50 44.989236 45.377762 42.642884 47.656859 45.114767 42.935852 48.342374
## V51 48.842603 51.198796 47.117808 54.133029 52.007664 50.314990 51.725794
## V52 46.340835 46.133696 41.790616 52.008796 49.292604 45.622845 48.918811
## V53 46.317848 44.452051 40.772984 55.190922 51.073310 42.582260 46.754578
## V54 49.499624 49.630068 45.850817 52.491401 50.333799 45.396091 48.531380
## V55 51.449401 49.573332 46.433991 54.930728 52.005667 49.366994 49.137080
## V56 46.455323 48.088117 52.574606 59.774152 56.116230 54.722426 55.942630
## V57 45.319835 44.506662 49.175037 60.390721 54.322286 49.978734 51.084458
## V58 44.224729 44.474407 48.456401 59.510273 54.520001 50.242193 51.911801
## V59 43.508541 43.779713 44.340062 59.084367 53.770940 48.195421 51.856388
## V60 50.335829 49.705546 46.058780 52.753414 49.520739 48.073823 51.181168
## V61 50.148148 50.432938 45.324182 57.939326 52.791619 48.132524 51.762484
## V62 52.103949 54.007531 48.678972 60.584728 55.947295 52.785530 55.998246
## V63 53.647908 54.612588 49.566633 62.041748 56.767311 53.654713 56.310752
##           V43       V44       V45       V46       V47       V48       V49
## V2
## V3
## V4
## V5
## V6
## V7
## V8
## V9
## V10
## V11
## V12
## V13
## V14
## V15
## V16
## V17
## V18
## V19
## V20
## V21
## V22
## V23
## V24
## V25
## V26
## V27
## V28
## V29
## V30
## V31
## V32
## V33
## V34
## V35
## V36
## V37
## V38
## V39
## V40
## V41
## V42
## V43
## V44 51.276546
## V45 51.990727 37.717353
## V46 49.830141 34.110783 24.614659
## V47 54.250465 30.342261 38.144542 34.308928
## V48 53.976463 37.149897 40.913469 35.918118 29.302378
## V49 56.588294 40.531911 40.085509 36.638014 29.558010 33.505790
## V50 54.609103 46.033365 28.045657 36.110346 37.728102 41.533907 31.847050
## V51 57.002527 40.229999 42.218832 36.937505 35.104684 32.301662 28.590021
## V52 54.719066 38.361612 35.573806 31.807275 32.246978 32.469965 27.219470
## V53 52.820182 35.923222 38.167620 35.067931 30.505741 32.937769 33.240775
## V54 55.290937 40.544741 39.809758 36.605021 33.044056 34.071424 28.989815
## V55 55.697795 41.353654 42.309033 39.022736 35.008199 34.097593 28.586799
## V56 57.756663 42.355099 42.261769 40.172746 45.462568 49.342165 43.854780
## V57 53.293260 38.951621 42.414567 40.542602 44.183876 47.765980 46.430779
## V58 54.849245 39.007269 41.839498 39.738807 42.893188 47.404158 45.342627
## V59 53.516184 35.759363 37.910494 36.156703 39.150316 42.808945 43.808212
## V60 56.628455 47.372399 45.126701 43.753842 40.770530 40.760986 39.453473
## V61 55.814188 45.126140 43.359285 42.080566 38.856172 36.695519 39.394562
## V62 59.037166 48.026037 45.809854 43.534081 42.280431 41.131933 42.933607
## V63 57.933891 48.771017 45.972155 43.568516 44.531068 41.200559 43.852082
##           V50       V51       V52       V53       V54       V55       V56
## V2
## V3
## V4
## V5
## V6
## V7
## V8
## V9
## V10
## V11
## V12
## V13
## V14
## V15
## V16
## V17
## V18
## V19
## V20
## V21
## V22
## V23
## V24
## V25
## V26
## V27
## V28
## V29
## V30
## V31
## V32
## V33
## V34
## V35
## V36
## V37
## V38
## V39
## V40
## V41
## V42
## V43
## V44
## V45
## V46
## V47
## V48
## V49
## V50
## V51 38.122899
## V52 32.120418 30.107098
## V53 37.611998 35.506062 30.171735
## V54 34.261585 34.542577 25.473438 31.199151
## V55 37.730709 31.299823 29.911359 33.325990 34.084098
## V56 46.885374 46.406838 44.137078 46.694726 46.548240 46.947780
## V57 48.000714 49.384896 45.934953 45.074569 46.758066 46.733689 27.695399
## V58 48.007073 47.560345 44.505569 44.060429 46.275011 47.236979 24.357130
## V59 45.949576 45.227740 42.378525 41.398577 44.932347 45.886220 28.866317
## V60 41.920336 43.668356 41.868678 41.355481 42.719809 41.231391 39.997577
## V61 42.292880 41.848014 39.562453 39.804843 41.827150 40.117045 36.545316
## V62 46.882836 43.235445 42.467122 43.749831 44.454764 44.282543 37.212559
## V63 47.440332 42.569581 40.811878 44.291972 44.634031 44.122159 39.166193
##           V57       V58       V59       V60       V61       V62
## V2
## V3
## V4
## V5
## V6
## V7
## V8
## V9
## V10
## V11
## V12
## V13
## V14
## V15
## V16
## V17
## V18
## V19
## V20
## V21
## V22
## V23
## V24
## V25
## V26
## V27
## V28
## V29
## V30
## V31
## V32
## V33
## V34
## V35
## V36
## V37
## V38
## V39
## V40
## V41
## V42
## V43
## V44
## V45
## V46
## V47
## V48
## V49
## V50
## V51
## V52
## V53
## V54
## V55
## V56
## V57
## V58 22.592502
## V59 29.313990 24.603732
## V60 40.079186 37.387391 35.176779
## V61 37.273523 34.436291 31.419907 27.035524
## V62 39.032624 34.753541 32.575050 30.286965 27.577101
## V63 41.768154 36.664386 34.882215 32.471611 29.198692 23.368415
```

b. Perform hierarchical clustering using average linkage.

```r
dist.416b <- dist(gene.expression)
tree.416b <- hclust(dist.416b, "ward.D2")

dend <- as.dendrogram(tree.416b, hang=0.1)
```

c. Plot a dendrogram associated with the hierarchical clustering you just computed.

```r
plot(dend)
```



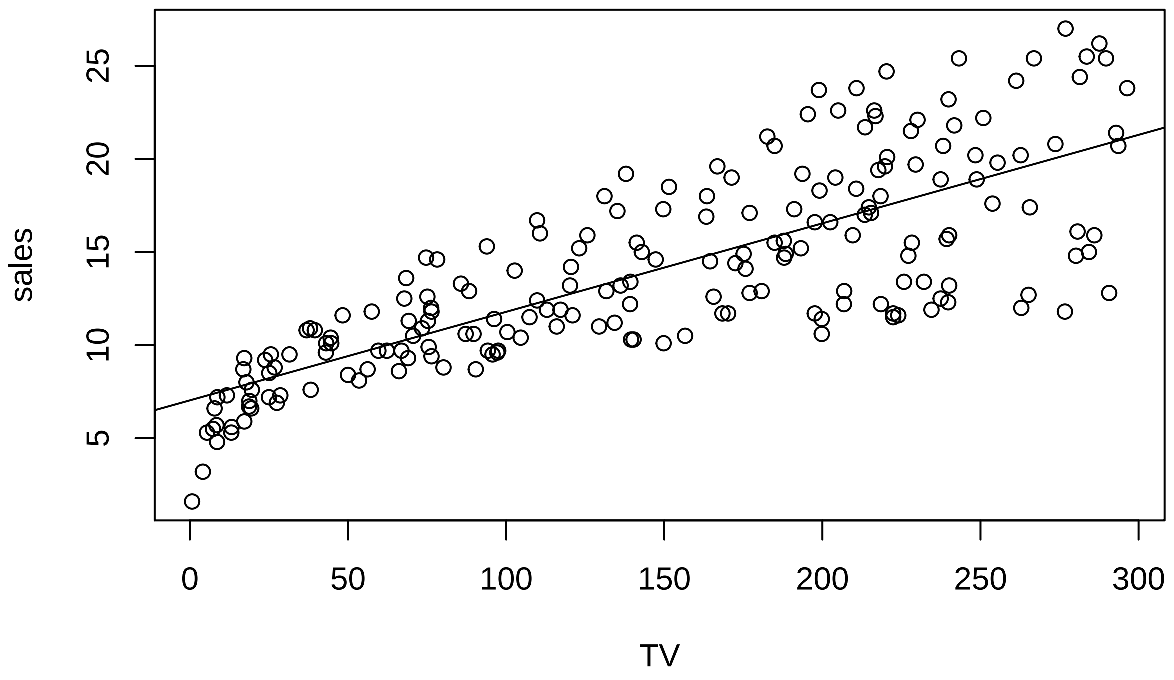
Exercise 3: Ads

a. Read the data from “http://www-bcf.usc.edu/~gareth/ISL/Advertising.csv” containing information on sales of a product and the amount spent on advertising using different media channels.

```r
ads <- read.csv("Advertising.csv")
```

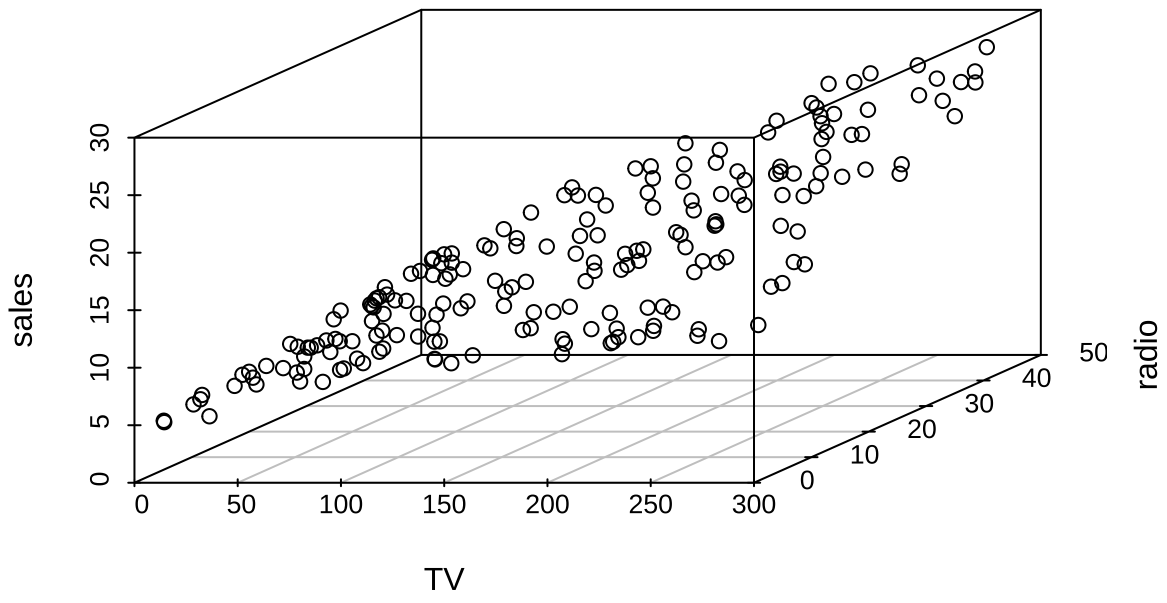
b. Generate a scatterplot of sales against the amount of TV advertising and add a linear fit line.

```r
plot(sales ~ TV, data = ads)
abline(lm(sales ~ TV, data = ads))
```



c. Now make a 3D scatterplot with axes corresponding to 'sales', 'TV' and 'radio'.

```
library("scatterplot3d") # load


scatterplot3d(ads[,c(2,3,5)], angle = 40)
```



d. The dataset has 200 rows. Divide it into a train set with 150 observations and a test set with 50 observations, i.e. use `sample(1:200, n = 150)` to randomly choose row indices of the advertising dataset to include in the train set. The remaining indices should be used for the test set. Remember to choose and set the seed for randomization!

```
set.seed(100)

x <- sample(1:200, size = 200)

train <- x[1:150]
test <- x[151:200]
```

e. Fit a linear model to the training set, where the sales values are predicted by the amount of TV advertising. Print the summary of the fitted model. Then, predict the sales values for the test set and evaluate the test model accuracy in terms of root mean squared error (MSE), which measures the average level of error between the prediction and the true response.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$$

```
l <- lm(sales ~ TV, ads[train,])
summary(l)
```

```
##
```

```
## Call:
## lm(formula = sales ~ TV, data = ads[train, ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.2501 -1.8569 -0.0714  1.8906  7.3073
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7.042490   0.530316   13.28   <2e-16 ***
## TV          0.047010   0.003085   15.24   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.228 on 148 degrees of freedom
## Multiple R-squared:  0.6107, Adjusted R-squared:  0.6081
## F-statistic: 232.2 on 1 and 148 DF,  p-value: < 2.2e-16
```

```
testSet <- ads[test,]
b0 = 7.042490
b1 = .047010

se = 0
for (row in 1:nrow(testSet)) {
  tv <- testSet[row, "TV"]
  sales <- testSet[row, "sales"]
  predict_value = b0 + b1 * tv

  se = se + (sales-predict_value) ^ 2
}
rmse1 = sqrt(se/50)
```

f. Fit a multiple linerar regression model including all the variables 'TV', 'radio', 'newspaper' to model the 'sales' in the training set. Then, compute the predicted sales for the test set with the new model and evalued the RMSE.
Did the error decrease from the one correspodning to the previous model?

```
l2 <- lm(sales ~ TV + radio + newspaper, ads[train,])
summary(l2)
```

```
##
## Call:
## lm(formula = sales ~ TV + radio + newspaper, data = ads[train,
##     ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.7057 -0.7780  0.2515  1.1511  2.8430
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.867562   0.362800   7.904 6.04e-13 ***
## TV          0.045332   0.001620  27.984  < 2e-16 ***
## radio       0.185882   0.010419  17.841  < 2e-16 ***
## newspaper   0.005226   0.007158   0.730    0.466
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.692 on 146 degrees of freedom
## Multiple R-squared:  0.8946, Adjusted R-squared:  0.8924
## F-statistic: 412.9 on 3 and 146 DF,  p-value: < 2.2e-16
```

```
testSet <- ads[test,]
b0 = 2.867562
b1 = .045332
b2 = .185882
b3 = .005226


se2 = 0
for (row in 1:nrow(testSet)) {
  tv <- testSet[row, "TV"]
  radio <- testSet[row, "radio"]
  newp <- testSet[row, "newspaper"]

  sales <- testSet[row, "sales"]
  predict_value = b0 + b1 * tv + b2 * radio + b3 * newp

  se2 = se2 + (sales-predict_value) ^ 2
}
rmse2 = sqrt(se2/50)

rmse2 - rmse1
```

```
## [1] -1.662255
```

```
#the error decreased
```

g. Look at the summary output for the multiple regression model and note which of the coefficient in the model is significant. Are all of them significant? If not refit the model including only the features found significant. Which of the models should you choose?

```
summary(l2)
```

```
##
## Call:
## lm(formula = sales ~ TV + radio + newspaper, data = ads[train,
##     ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.7057 -0.7780  0.2515  1.1511  2.8430
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.867562   0.362800   7.904 6.04e-13 ***
## TV          0.045332   0.001620  27.984  < 2e-16 ***
## radio       0.185882   0.010419  17.841  < 2e-16 ***
## newspaper   0.005226   0.007158   0.730    0.466
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.692 on 146 degrees of freedom
## Multiple R-squared:  0.8946, Adjusted R-squared:  0.8924
## F-statistic: 412.9 on 3 and 146 DF,  p-value: < 2.2e-16
```

```
#newspaper is insignificant

l3 <- lm(sales ~ TV + radio, ads[train,])
summary(l3)
```

```
##
## Call:
## lm(formula = sales ~ TV + radio, data = ads[train, ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.8604 -0.8568  0.2472  1.1225  2.8355
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.946431   0.345795   8.521 1.74e-14 ***
## TV          0.045379   0.001616  28.079  < 2e-16 ***
## radio       0.188946   0.009522  19.844  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.689 on 147 degrees of freedom
## Multiple R-squared:  0.8942, Adjusted R-squared:  0.8927
## F-statistic: 621.1 on 2 and 147 DF,  p-value: < 2.2e-16
```

```
b0 = 2.946431
b1 = .045379
b2 = .188946


se3 = 0
for (row in 1:nrow(testSet)) {
  tv <- testSet[row, "TV"]
  radio <- testSet[row, "radio"]

  sales <- testSet[row, "sales"]
  predict_value = b0 + b1 * tv + b2 * radio

  se3 = se3 + (sales-predict_value) ^ 2
}
rmse3 = sqrt(se3/50)
rmse3
```

```
## [1] 1.660314
```

```r
rmse2
```

```
## [1] 1.688029
```

```r
#the RMSE value is smaller for this model, so we should chose this model
```

Exercise 4: Movies

Recall the movies data-frame we used ealier in the bootcamp. It contains information on movies from the last three decates, which was scrapped from the IMDB database.

```r
library(dplyr)
url <- "https://raw.githubusercontent.com/Juanets/movie-stats/master/movies.csv"
movies <- tbl_df(read.csv(url))
```

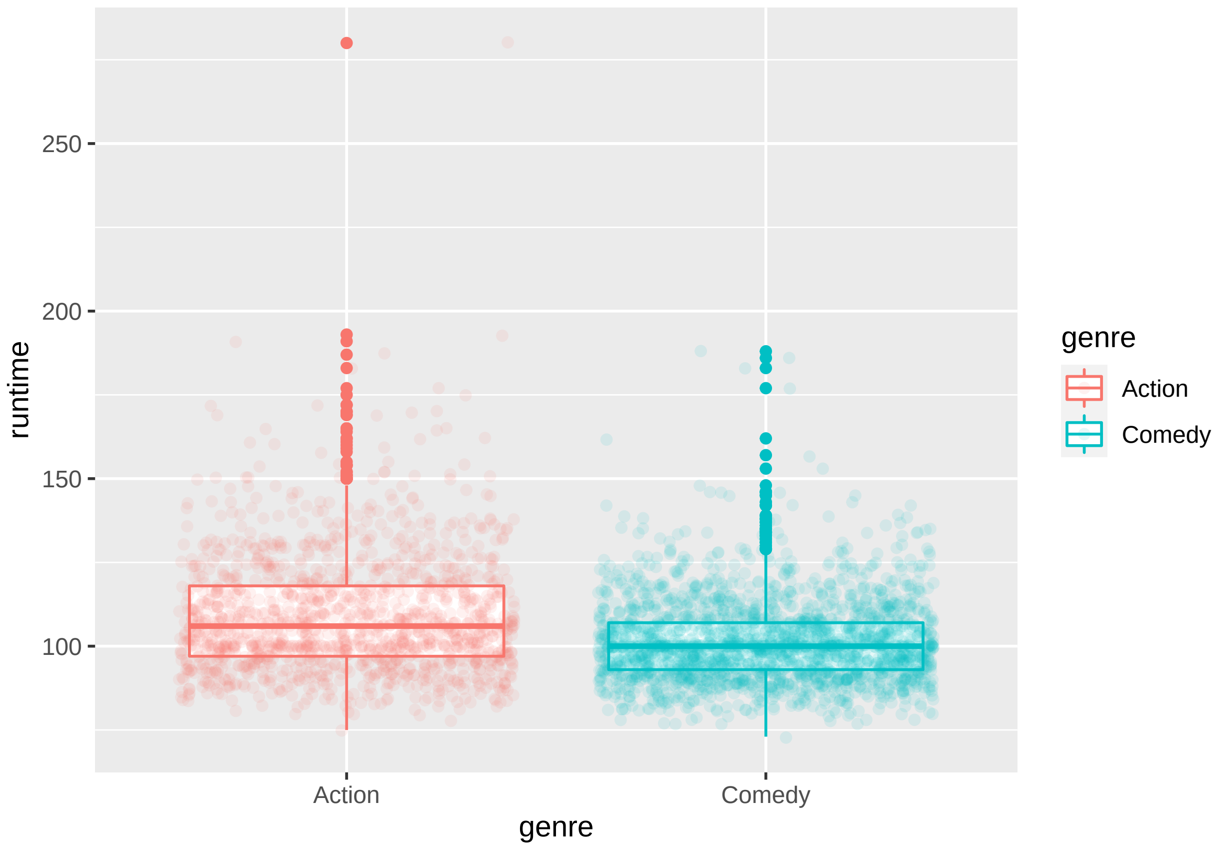
```
## Warning: `tbl_df()` is deprecated as of dplyr 1.0.0.
## Please use `tibble::as_tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

a. Generate a boxplot of runtimes for action movies and comedies with jittered points overlaid on top. You might consider setting collor, fill and alpha arguments to modify clarity and transparency of the plot.

```r
colnames(movies)
```

```
##  [1] "budget"   "company"  "country"  "director" "genre"    "gross"
##  [7] "name"     "rating"   "released" "runtime"  "score"    "star"
## [13] "votes"    "writer"   "year"
```

```r
movies2 <- movies %>% filter(genre == "Action" | genre == "Comedy")

ggplot(movies2,aes(x=genre, y=runtime, color = genre)) +
  geom_boxplot() +
  geom_jitter(alpha = .1)
```



b. Test a hypothesis that the action movies have higher mean runtime (length) than the comedies. Is the difference statistically greater than zero at significance level $\alpha = 0.05$?

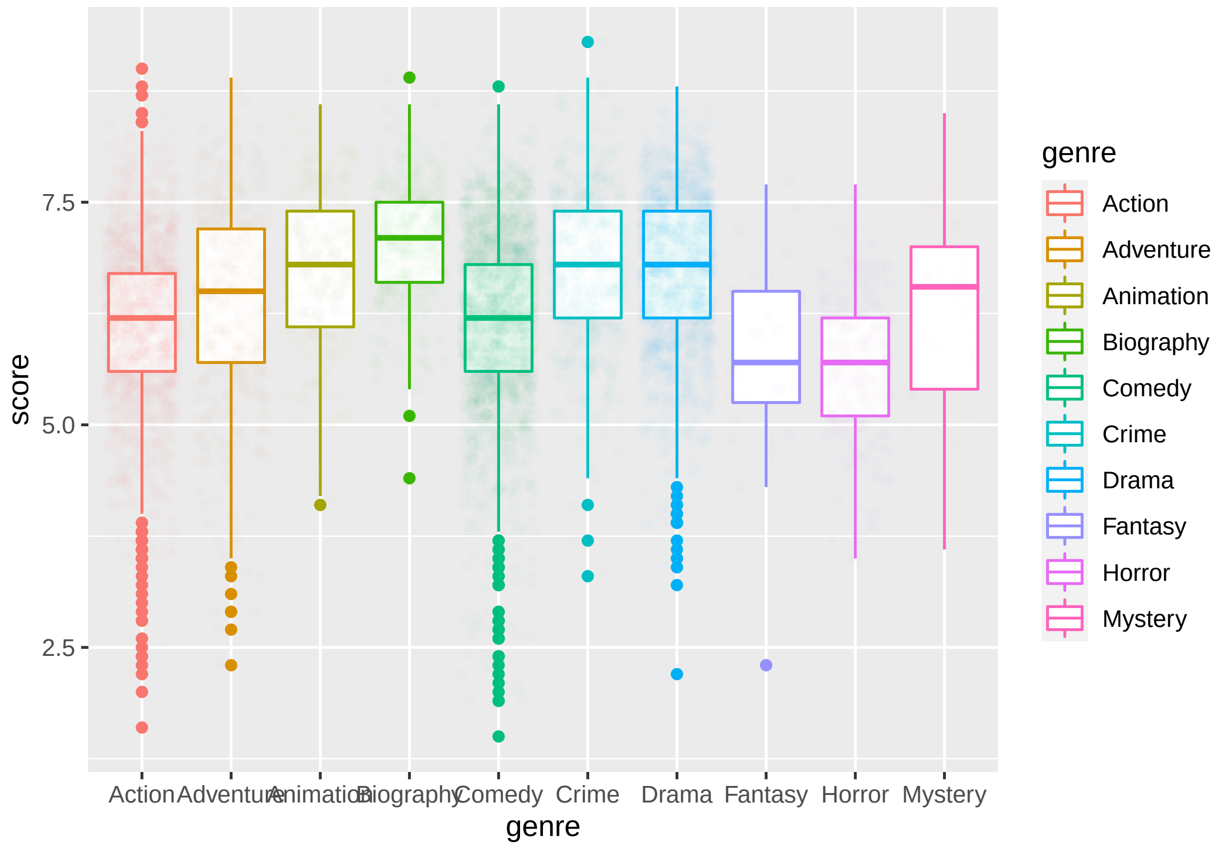
```r
library(tidyverse)
library(dplyr)

action <- movies %>% filter(genre == "Action")
comedy <- movies %>% filter(genre == "Comedy")

t.test(action$runtime, comedy$runtime, alternative = "greater")
```

```
##
##  Welch Two Sample t-test
##
## data:  action$runtime and comedy$runtime
## t = 14.094, df = 2120.7, p-value < 2.2e-16
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  6.881018      Inf
## sample estimates:
## mean of x mean of y
##  109.0008  101.2101
```

```
#yes, the difference is greater than 0 at significance level p < 2.2e-16
```

c. Test the hypothesis that the scores are the same across movie types (keep the movie genre which have at least 20 movies). Plot the data before making a test of your choice. State all the assumptions that you are making when devising your test.

```
movies3 <- movies %>% group_by(genre) %>% mutate(genre_count = n())


ggplot(movies3 %>% filter(genre_count>20),aes(x=genre, y=score, color = genre)) +
  geom_boxplot() +
  geom_jitter(alpha = .01)
```



```
anova_movies <- aov(score ~ genre, movies3 %>% filter(genre_count>20))
summary(anova_movies)
```

```
##                 Df Sum Sq Mean Sq F value Pr(>F)
## genre            9    771   85.65   95.69 <2e-16 ***
## Residuals     6742   6035    0.90
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
#assumption -> normal distribution
```

d. Is there a reason to believe that the scores might differ according to genre? How would you test which one is different (do the test if you have reason to believe that this is the case).

```r
#anova is significant, so lm to look at each individual categorical option (genre)

lm_movies <- lm(score ~ genre, movies3 %>% filter(genre_count>20))
summary(lm_movies)
```

```
##
## Call:
## lm(formula = score ~ genre, data = movies3 %>% filter(genre_count >
##     20))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6658 -0.5489  0.0596  0.6342  2.8939
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      6.10609    0.02593 235.463  < 2e-16 ***
## genreAdventure   0.24876    0.05437   4.576 4.83e-06 ***
## genreAnimation   0.64085    0.06248  10.257  < 2e-16 ***
## genreBiography   0.93430    0.05626  16.606  < 2e-16 ***
## genreComedy      0.05968    0.03321   1.797   0.0723 .
## genreCrime       0.65024    0.04886  13.308  < 2e-16 ***
## genreDrama       0.60963    0.03595  16.958  < 2e-16 ***
## genreFantasy    -0.33109    0.16924  -1.956   0.0505 .
## genreHorror     -0.41908    0.06248  -6.707 2.14e-11 ***
## genreMystery     0.23602    0.15565   1.516   0.1295
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9461 on 6742 degrees of freedom
## Multiple R-squared:  0.1133, Adjusted R-squared:  0.1121
## F-statistic: 95.69 on 9 and 6742 DF,  p-value: < 2.2e-16
```