Solution: Statistics, dimensionality reduction, and clustering

Author name goes here

2023-08-31

## Demo

### The standard normal distribution

#### pnorm()

The quantiles represent a set of possible values in the distribution.

data\_quantiles <- seq(-5, 5, 0.1)  
head(data\_quantiles)

## [1] -5.0 -4.9 -4.8 -4.7 -4.6 -4.5

The function pnorm() returns the probability of a observing a value less or equal to each quantile given.

data\_pnorm <- pnorm(q = data\_quantiles)   
head(data\_pnorm)

## [1] 2.866516e-07 4.791833e-07 7.933282e-07 1.300807e-06 2.112455e-06  
## [6] 3.397673e-06

The base R plot() function can be used to quickly plot each quantile and its probability.

plot(data\_quantiles, data\_pnorm)

#### qnorm()

The function qnorm() takes as input a vector of probabilities between 0 and 1.

data\_probabilities <- seq(from = 0, to = 1, by = 0.02)  
head(data\_probabilities)

## [1] 0.00 0.02 0.04 0.06 0.08 0.10

The function qnorm() the quantiles (i.e., values) that correspond to those probabilities.

data\_qnorm <- qnorm(p = data\_probabilities)   
head(data\_qnorm)

## [1] -Inf -2.053749 -1.750686 -1.554774 -1.405072 -1.281552

The base R plot() function can be used to quickly plot each quantile and its probability.

plot(data\_probabilities, data\_qnorm)

#### dnorm

The quantiles represent a set of possible values in the distribution.

data\_quantiles <- seq(-5, 5, 0.2)  
head(data\_quantiles)

## [1] -5.0 -4.8 -4.6 -4.4 -4.2 -4.0

The function pnorm() returns the probability of a observing a value less or equal to each quantile given.

data\_dnorm <- dnorm(x = data\_quantiles)   
head(data\_dnorm)

## [1] 1.486720e-06 3.961299e-06 1.014085e-05 2.494247e-05 5.894307e-05  
## [6] 1.338302e-04

The base R plot() function can be used to quickly plot each quantile and its probability.

plot(data\_quantiles, data\_dnorm)

#### rnorm

data\_rnorm <- rnorm(n = 100)  
head(data\_rnorm)

## [1] 0.61897357 0.04457844 -2.48340181 0.18583057 1.48973777 -1.13369039

hist(data\_rnorm, breaks = 20)

## Demo

### A parameterised normal distribution

#### pnorm()

The quantiles represent a set of possible values in the distribution.

data\_quantiles <- seq(-500, 500, 50)  
head(data\_quantiles)

## [1] -500 -450 -400 -350 -300 -250

The function pnorm() returns the probability of a observing a value less or equal to each quantile given.

data\_pnorm <- pnorm(q = data\_quantiles, mean = 50, sd = 100)  
head(data\_pnorm)

## [1] 1.898956e-08 2.866516e-07 3.397673e-06 3.167124e-05 2.326291e-04  
## [6] 1.349898e-03

The base R plot() function can be used to quickly plot each quantile and its probability.

plot(data\_quantiles, data\_pnorm)

#### qnorm()

The function qnorm() takes as input a vector of probabilities between 0 and 1.

data\_probabilities <- seq(from = 0, to = 1, by = 0.02)  
head(data\_probabilities)

## [1] 0.00 0.02 0.04 0.06 0.08 0.10

The function qnorm() the quantiles (i.e., values) that correspond to those probabilities.

data\_qnorm <- qnorm(p = data\_probabilities, mean = 50, sd = 100)   
head(data\_qnorm)

## [1] -Inf -155.37489 -125.06861 -105.47736 -90.50716 -78.15516

The base R plot() function can be used to quickly plot each quantile and its probability.

plot(data\_probabilities, data\_qnorm)

#### dnorm

The quantiles represent a set of possible values in the distribution.

data\_quantiles <- seq(-500, 500, 50)  
head(data\_quantiles)

## [1] -500 -450 -400 -350 -300 -250

The function pnorm() returns the probability of a observing a value less or equal to each quantile given.

data\_dnorm <- dnorm(x = data\_quantiles, mean = 50, sd = 100)   
head(data\_dnorm)

## [1] 1.076976e-09 1.486720e-08 1.598374e-07 1.338302e-06 8.726827e-06  
## [6] 4.431848e-05

The base R plot() function can be used to quickly plot each quantile and its probability.

plot(data\_quantiles, data\_dnorm)

#### rnorm

data\_rnorm <- rnorm(n = 100, mean = 50, sd = 100)  
head(data\_rnorm)

## [1] 113.843076 174.330022 2.217055 22.221379 158.342632 -11.240325

hist(data\_rnorm, breaks = 20)

## Demo

### A parameterised binomial distribution

#### pnorm()

The quantiles represent a set of possible values in the distribution.

data\_quantiles <- seq(0, 50, 1)  
head(data\_quantiles)

## [1] 0 1 2 3 4 5

The function pnorm() returns the probability of a observing a value less or equal to each quantile given.

data\_pbinom <- pbinom(q = data\_quantiles, size = 50, prob = 0.1)  
head(data\_pbinom)

## [1] 0.005153775 0.033785860 0.111728756 0.250293906 0.431198407 0.616123008

The base R plot() function can be used to quickly plot each quantile and its probability.

plot(data\_quantiles, data\_pbinom)

#### qbinom()

The function qbinom() takes as input a vector of probabilities between 0 and 1.

data\_probabilities <- seq(from = 0, to = 1, by = 0.02)  
head(data\_probabilities)

## [1] 0.00 0.02 0.04 0.06 0.08 0.10

The function qbinom() the quantiles (i.e., values) that correspond to those probabilities.

data\_qbinom <- qbinom(p = data\_probabilities, size = 50, prob = 0.1)  
head(data\_qbinom)

## [1] 0 1 2 2 2 2

The base R plot() function can be used to quickly plot each quantile and its probability.

plot(data\_probabilities, data\_qbinom)

#### dbinom

The quantiles represent a set of possible values in the distribution.

data\_quantiles <- seq(0, 50, 1)  
head(data\_quantiles)

## [1] 0 1 2 3 4 5

The function pbinom() returns the probability of a observing a value less or equal to each quantile given.

data\_dbinom <- dbinom(x = data\_quantiles, size = 50, prob = 0.1)  
head(data\_dbinom)

## [1] 0.005153775 0.028632084 0.077942897 0.138565150 0.180904501 0.184924601

The base R plot() function can be used to quickly plot each quantile and its probability.

plot(data\_quantiles, data\_dbinom)

#### rbinom

data\_rbinom <- rbinom(n = 100, size = 50, prob = 0.1)  
head(data\_rbinom)

## [1] 4 5 5 8 3 4

hist(data\_rbinom, breaks = 20)

## Demo

### Quantiles

data\_rnorm <- rnorm(n = 1000, mean = 50, sd = 100)  
quantile(data\_rnorm, probs = seq(0, 1, 0.1))

## 0% 10% 20% 30% 40% 50% 60%   
## -323.02247 -76.67681 -31.37632 -1.63237 32.21421 55.52598 79.53564   
## 70% 80% 90% 100%   
## 105.96925 136.10909 173.78082 349.13385

plot(  
 seq(0, 1, 0.1),  
 quantile(data\_rnorm, probs = seq(0, 1, 0.1))  
)

## Demo

### Parametric t-test

set.seed(1)  
x <- rnorm(n = 1000, mean = 0, sd = 1)  
y <- rnorm(n = 2000, mean = 1, sd = 1)

In base R, par(mfrow=c(i, j))) can be used to display plots in a grid of i rows and j columns.

par(mfrow = c(2, 1))  
hist(x, breaks = 30)  
hist(y, breaks = 30)

par(mfrow = c(1, 1 ))

t.test(x, y)

##   
## Welch Two Sample t-test  
##   
## data: x and y  
## t = -25.223, df = 1999, p-value < 2.2e-16  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## -1.089791 -0.932552  
## sample estimates:  
## mean of x mean of y   
## -0.01164814 0.99952356

t.test(y, x)

##   
## Welch Two Sample t-test  
##   
## data: y and x  
## t = 25.223, df = 1999, p-value < 2.2e-16  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## 0.932552 1.089791  
## sample estimates:  
## mean of x mean of y   
## 0.99952356 -0.01164814

## Demo

### Paired test

Simulate a vector of values x. Then simulate values y that are systematically 2 units greater than their x counterpart, with a bit of noise (normally distributed).

set.seed(1)  
n\_sample <- 10  
x <- runif(n = n\_sample, min = 10, max = 20)  
y <- x + 2 + rnorm(n = n\_sample, mean = 0, sd = 1)

The average difference between y and x values is approximately 2, as intended.

mean(y - x)

## [1] 2.086629

t.test(x, y, paired = TRUE)

##   
## Paired t-test  
##   
## data: x and y  
## t = -6.0238, df = 9, p-value = 0.0001967  
## alternative hypothesis: true mean difference is not equal to 0  
## 95 percent confidence interval:  
## -2.870241 -1.303017  
## sample estimates:  
## mean difference   
## -2.086629

## Demo

### Non-parametric tests

set.seed(1)  
x <- runif(n = 10, min = 1, max = 11)  
y <- runif(n = 5, min = 3, max = 13)

wilcox.test(x, y)

##   
## Wilcoxon rank sum exact test  
##   
## data: x and y  
## W = 20, p-value = 0.5941  
## alternative hypothesis: true location shift is not equal to 0

wilcox.test(x, y, alternative = "less")

##   
## Wilcoxon rank sum exact test  
##   
## data: x and y  
## W = 20, p-value = 0.297  
## alternative hypothesis: true location shift is less than 0

## Demo

### Analysis of Variance (ANOVA)

set.seed(1)  
n\_sample <- 1000  
x1 <- rnorm(n = n\_sample, mean = 10, sd = 2)  
x2 <- x1 + 5 + rnorm(n = n\_sample, mean = 0, sd = 1)  
x3 <- x2 + 0 + rnorm(n = n\_sample, mean = 0, sd = 0.5)  
data\_aov <- data.frame(  
 value = c(x1, x2, x3),  
 group = c(  
 rep("x1", length(x1)),  
 rep("x2", length(x2)),  
 rep("x3", length(x3))  
 )  
)  
head(data\_aov)

## value group  
## 1 8.747092 x1  
## 2 10.367287 x1  
## 3 8.328743 x1  
## 4 13.190562 x1  
## 5 10.659016 x1  
## 6 8.359063 x1

aov\_result <- aov(value ~ group, data\_aov)  
aov\_result

## Call:  
## aov(formula = value ~ group, data = data\_aov)  
##   
## Terms:  
## group Residuals  
## Sum of Squares 16583.9 15450.7  
## Deg. of Freedom 2 2997  
##   
## Residual standard error: 2.270548  
## Estimated effects may be unbalanced

summary(aov\_result)

## Df Sum Sq Mean Sq F value Pr(>F)   
## group 2 16584 8292 1608 <2e-16 \*\*\*  
## Residuals 2997 15451 5   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

## Demo

### Fisher’s Exact Test

data\_fisher <- matrix(  
 data = c(12, 4, 3, 23),  
 nrow = 2, ncol = 2,  
 dimnames = list(  
 c("DE", "Not DE"),  
 c("In pathway", "Not in pathway")  
 )  
)  
data\_fisher

## In pathway Not in pathway  
## DE 12 3  
## Not DE 4 23

fisher.test(data\_fisher)

##   
## Fisher's Exact Test for Count Data  
##   
## data: data\_fisher  
## p-value = 4.983e-05  
## alternative hypothesis: true odds ratio is not equal to 1  
## 95 percent confidence interval:  
## 3.592731 170.706615  
## sample estimates:  
## odds ratio   
## 20.56889

## Demo

### Multiple-testing correction

data\_p\_values <- runif(1E3, min = 0, max = 1)  
data\_p\_adjusted <- p.adjust(data\_p\_values, method = "BH")  
head(sort(data\_p\_adjusted))

## [1] 0.8215773 0.8215773 0.8215773 0.8215773 0.8215773 0.8215773

par(mfrow = c(2, 1))  
hist(data\_p\_values, xlim = c(0, 1), breaks = seq(0, 1, 0.05))  
hist(data\_p\_adjusted, xlim = c(0, 1), breaks = seq(0, 1, 0.05))

par(mfrow = c(1, 1))

## Exercise

### Setup

* Import the iris data set.

data(iris)  
head(iris)

## Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
## 1 5.1 3.5 1.4 0.2 setosa  
## 2 4.9 3.0 1.4 0.2 setosa  
## 3 4.7 3.2 1.3 0.2 setosa  
## 4 4.6 3.1 1.5 0.2 setosa  
## 5 5.0 3.6 1.4 0.2 setosa  
## 6 5.4 3.9 1.7 0.4 setosa

* Separate the matrix of measurements in a new object named iris\_features.

iris\_features <- iris[, 1:4]  
head(iris\_features)

## Sepal.Length Sepal.Width Petal.Length Petal.Width  
## 1 5.1 3.5 1.4 0.2  
## 2 4.9 3.0 1.4 0.2  
## 3 4.7 3.2 1.3 0.2  
## 4 4.6 3.1 1.5 0.2  
## 5 5.0 3.6 1.4 0.2  
## 6 5.4 3.9 1.7 0.4

## Exercise

### Apply Principal Components Analysis (PCA)

The prcomp() function allows you to standardise the data as part of the principal components analysis itself.

* Apply PCA, centering and scaling the matrix of features. Assign the result to an object called pca\_iris.

pca\_iris <- prcomp(iris\_features, center = TRUE, scale. = TRUE )  
pca\_iris

## Standard deviations (1, .., p=4):  
## [1] 1.7083611 0.9560494 0.3830886 0.1439265  
##   
## Rotation (n x k) = (4 x 4):  
## PC1 PC2 PC3 PC4  
## Sepal.Length 0.5210659 -0.37741762 0.7195664 0.2612863  
## Sepal.Width -0.2693474 -0.92329566 -0.2443818 -0.1235096  
## Petal.Length 0.5804131 -0.02449161 -0.1421264 -0.8014492  
## Petal.Width 0.5648565 -0.06694199 -0.6342727 0.5235971

* Examine the object pca\_iris. Display the loading of each feature on each principal component.

pca\_iris

## Standard deviations (1, .., p=4):  
## [1] 1.7083611 0.9560494 0.3830886 0.1439265  
##   
## Rotation (n x k) = (4 x 4):  
## PC1 PC2 PC3 PC4  
## Sepal.Length 0.5210659 -0.37741762 0.7195664 0.2612863  
## Sepal.Width -0.2693474 -0.92329566 -0.2443818 -0.1235096  
## Petal.Length 0.5804131 -0.02449161 -0.1421264 -0.8014492  
## Petal.Width 0.5648565 -0.06694199 -0.6342727 0.5235971

summary(pca\_iris)

## Importance of components:  
## PC1 PC2 PC3 PC4  
## Standard deviation 1.7084 0.9560 0.38309 0.14393  
## Proportion of Variance 0.7296 0.2285 0.03669 0.00518  
## Cumulative Proportion 0.7296 0.9581 0.99482 1.00000

pca\_iris$rotation[1:4, ]

## PC1 PC2 PC3 PC4  
## Sepal.Length 0.5210659 -0.37741762 0.7195664 0.2612863  
## Sepal.Width -0.2693474 -0.92329566 -0.2443818 -0.1235096  
## Petal.Length 0.5804131 -0.02449161 -0.1421264 -0.8014492  
## Petal.Width 0.5648565 -0.06694199 -0.6342727 0.5235971

pca\_iris$x

## PC1 PC2 PC3 PC4  
## [1,] -2.25714118 -0.478423832 0.127279624 0.024087508  
## [2,] -2.07401302 0.671882687 0.233825517 0.102662845  
## [3,] -2.35633511 0.340766425 -0.044053900 0.028282305  
## [4,] -2.29170679 0.595399863 -0.090985297 -0.065735340  
## [5,] -2.38186270 -0.644675659 -0.015685647 -0.035802870  
## [6,] -2.06870061 -1.484205297 -0.026878250 0.006586116  
## [7,] -2.43586845 -0.047485118 -0.334350297 -0.036652767  
## [8,] -2.22539189 -0.222403002 0.088399352 -0.024529919  
## [9,] -2.32684533 1.111603700 -0.144592465 -0.026769540  
## [10,] -2.17703491 0.467447569 0.252918268 -0.039766068  
## [11,] -2.15907699 -1.040205867 0.267784001 0.016675503  
## [12,] -2.31836413 -0.132633999 -0.093446191 -0.133037725  
## [13,] -2.21104370 0.726243183 0.230140246 0.002416941  
## [14,] -2.62430902 0.958296347 -0.180192423 -0.019151375  
## [15,] -2.19139921 -1.853846555 0.471322025 0.194081578  
## [16,] -2.25466121 -2.677315230 -0.030424684 0.050365010  
## [17,] -2.20021676 -1.478655729 0.005326251 0.188186988  
## [18,] -2.18303613 -0.487206131 0.044067686 0.092779618  
## [19,] -1.89223284 -1.400327567 0.373093377 0.060891973  
## [20,] -2.33554476 -1.124083597 -0.132187626 -0.037630354  
## [21,] -1.90793125 -0.407490576 0.419885937 0.010884821  
## [22,] -2.19964383 -0.921035871 -0.159331502 0.059398340  
## [23,] -2.76508142 -0.456813301 -0.331069982 0.019582826  
## [24,] -1.81259716 -0.085272854 -0.034373442 0.150636353  
## [25,] -2.21972701 -0.136796175 -0.117599566 -0.269238379  
## [26,] -1.94532930 0.623529705 0.304620475 0.043416203  
## [27,] -2.04430277 -0.241354991 -0.086075649 0.067454082  
## [28,] -2.16133650 -0.525389422 0.206125707 0.010241084  
## [29,] -2.13241965 -0.312172005 0.270244895 0.083977887  
## [30,] -2.25769799 0.336604248 -0.068207276 -0.107918349  
## [31,] -2.13297647 0.502856075 0.074757996 -0.048027970  
## [32,] -1.82547925 -0.422280389 0.269564311 0.239069476  
## [33,] -2.60621687 -1.787587272 -0.047070727 -0.228470534  
## [34,] -2.43800983 -2.143546796 0.082392024 -0.048053409  
## [35,] -2.10292986 0.458665270 0.169706329 0.028926042  
## [36,] -2.20043723 0.205419224 0.224688852 0.168343905  
## [37,] -2.03831765 -0.659349230 0.482919584 0.195702902  
## [38,] -2.51889339 -0.590315163 -0.019370918 -0.136048774  
## [39,] -2.42152026 0.901161067 -0.192609402 -0.009705907  
## [40,] -2.16246625 -0.267981199 0.175296561 0.007023875  
## [41,] -2.27884081 -0.440240541 -0.034778398 0.106626042  
## [42,] -1.85191836 2.329610745 0.203552303 0.288896090  
## [43,] -2.54511203 0.477501017 -0.304745527 -0.066379077  
## [44,] -1.95788857 -0.470749613 -0.308567588 0.176501717  
## [45,] -2.12992356 -1.138415464 -0.247604064 -0.150539117  
## [46,] -2.06283361 0.708678586 0.063716370 0.139801160  
## [47,] -2.37677076 -1.116688691 -0.057026813 -0.151722682  
## [48,] -2.38638171 0.384957230 -0.139002234 -0.048671707  
## [49,] -2.22200263 -0.994627669 0.180886792 -0.014878291  
## [50,] -2.19647504 -0.009185585 0.152518539 0.049206884  
## [51,] 1.09810244 -0.860091033 0.682300393 0.034717469  
## [52,] 0.72889556 -0.592629362 0.093807452 0.004887251  
## [53,] 1.23683580 -0.614239894 0.552157058 0.009391933  
## [54,] 0.40612251 1.748546197 0.023024633 0.065549239  
## [55,] 1.07188379 0.207725147 0.396925784 0.104387166  
## [56,] 0.38738955 0.591302717 -0.123776885 -0.240027187  
## [57,] 0.74403715 -0.770438272 -0.148472007 -0.077111455  
## [58,] -0.48569562 1.846243998 -0.248432992 -0.040384912  
## [59,] 0.92480346 -0.032118478 0.594178807 -0.029779844  
## [60,] 0.01138804 1.030565784 -0.537100055 -0.028366154  
## [61,] -0.10982834 2.645211115 0.046634215 0.013714785  
## [62,] 0.43922201 0.063083852 -0.204389093 0.039992104  
## [63,] 0.56023148 1.758832129 0.763214554 0.045578465  
## [64,] 0.71715934 0.185602819 0.068429700 -0.164256922  
## [65,] -0.03324333 0.437537419 -0.194282030 0.108684396  
## [66,] 0.87248429 -0.507364239 0.501830204 0.104593326  
## [67,] 0.34908221 0.195656268 -0.489234095 -0.190869932  
## [68,] 0.15827980 0.789451008 0.301028700 -0.204612265  
## [69,] 1.22100316 1.616827281 0.480693656 0.225145511  
## [70,] 0.16436725 1.298259939 0.172260719 -0.051554138  
## [71,] 0.73521959 -0.395247446 -0.614467782 -0.083006045  
## [72,] 0.47469691 0.415926887 0.264067576 0.113189079  
## [73,] 1.23005729 0.930209441 0.367182178 -0.009911322  
## [74,] 0.63074514 0.414997441 0.290921638 -0.273304557  
## [75,] 0.70031506 0.063200094 0.444537765 0.043313222  
## [76,] 0.87135454 -0.249956017 0.471001057 0.101376117  
## [77,] 1.25231375 0.076998069 0.724727099 0.039556002  
## [78,] 1.35386953 -0.330205463 0.259955701 0.066604931  
## [79,] 0.66258066 0.225173502 -0.085577197 -0.036318171  
## [80,] -0.04012419 1.055183583 0.318506304 0.064571834  
## [81,] 0.13035846 1.557055553 0.149482697 -0.009371129  
## [82,] 0.02337438 1.567225244 0.240745761 -0.032663020  
## [83,] 0.24073180 0.774661195 0.150707074 0.023572390  
## [84,] 1.05755171 0.631726901 -0.104959762 -0.183354200  
## [85,] 0.22323093 0.286812663 -0.663028512 -0.253977520  
## [86,] 0.42770626 -0.842758920 -0.449129446 -0.109308985  
## [87,] 1.04522645 -0.520308714 0.394464890 0.037084781  
## [88,] 1.04104379 1.378371048 0.685997804 0.136378719  
## [89,] 0.06935597 0.218770433 -0.290605718 -0.146653279  
## [90,] 0.28253073 1.324886147 -0.089111491 0.008876070  
## [91,] 0.27814596 1.116288852 -0.094172116 -0.269753497  
## [92,] 0.62248441 -0.024839814 0.020412763 -0.147193289  
## [93,] 0.33540673 0.985103828 0.198724011 0.006508757  
## [94,] -0.36097409 2.012495825 -0.105467721 0.019505467  
## [95,] 0.28762268 0.852873116 -0.130452657 -0.107043742  
## [96,] 0.09105561 0.180587142 -0.128547696 -0.229191812  
## [97,] 0.22695654 0.383634868 -0.155691572 -0.132163118  
## [98,] 0.57446378 0.154356489 0.270743347 -0.019794366  
## [99,] -0.44617230 1.538637456 -0.189765199 0.199278855  
## [100,] 0.25587339 0.596852285 -0.091572385 -0.058426315  
## [101,] 1.83841002 -0.867515056 -1.002044077 -0.049085303  
## [102,] 1.15401555 0.696536401 -0.528389994 -0.040385459  
## [103,] 2.19790361 -0.560133976 0.202236658 0.058986583  
## [104,] 1.43534213 0.046830701 -0.163083761 -0.234982858  
## [105,] 1.86157577 -0.294059697 -0.394307408 -0.016243853  
## [106,] 2.74268509 -0.797736709 0.580364827 -0.101045973  
## [107,] 0.36579225 1.556289178 -0.983598122 -0.132679346  
## [108,] 2.29475181 -0.418663020 0.649530452 -0.237246445  
## [109,] 1.99998633 0.709063226 0.392675073 -0.086221779  
## [110,] 2.25223216 -1.914596301 -0.396224508 0.104488870  
## [111,] 1.35962064 -0.690443405 -0.283661780 0.107500284  
## [112,] 1.59732747 0.420292431 -0.023108991 0.058136869  
## [113,] 1.87761053 -0.417849815 -0.026250468 0.145926073  
## [114,] 1.25590769 1.158379741 -0.578311891 0.098826244  
## [115,] 1.46274487 0.440794883 -1.000517746 0.274738504  
## [116,] 1.58476820 -0.673986887 -0.636297054 0.191222383  
## [117,] 1.46651849 -0.254768327 -0.037306280 -0.154811637  
## [118,] 2.41822770 -2.548124795 0.127454475 -0.272892966  
## [119,] 3.29964148 -0.017721580 0.700957033 0.045037725  
## [120,] 1.25954707 1.701046715 0.266643612 -0.064963167  
## [121,] 2.03091256 -0.907427443 -0.234015510 0.167390481  
## [122,] 0.97471535 0.569855257 -0.825362161 0.027662914  
## [123,] 2.88797650 -0.412259950 0.854558973 -0.126911337  
## [124,] 1.32878064 0.480202496 0.005410239 0.139491837  
## [125,] 1.69505530 -1.010536476 -0.297454114 -0.061437911  
## [126,] 1.94780139 -1.004412720 0.418582432 -0.217609339  
## [127,] 1.17118007 0.315338060 -0.129503907 0.125001677  
## [128,] 1.01754169 -0.064131184 -0.336588365 -0.008625505  
## [129,] 1.78237879 0.186735633 -0.269754304 0.030983849  
## [130,] 1.85742501 -0.560413289 0.713244682 -0.207519953  
## [131,] 2.42782030 -0.258418706 0.725386035 -0.017863520  
## [132,] 2.29723178 -2.617554417 0.491826144 -0.210968943  
## [133,] 1.85648383 0.177953334 -0.352966242 0.099675959  
## [134,] 1.11042770 0.291944582 0.182875741 -0.185721512  
## [135,] 1.19845835 0.808606364 0.164173760 -0.487849130  
## [136,] 2.78942561 -0.853942542 0.541093785 0.294893130  
## [137,] 1.57099294 -1.065013214 -0.942695700 0.035486875  
## [138,] 1.34179696 -0.421020154 -0.180271551 -0.214702016  
## [139,] 0.92173701 -0.017165594 -0.415434449 0.005220919  
## [140,] 1.84586124 -0.673870645 0.012629804 0.194543500  
## [141,] 2.00808316 -0.611835930 -0.426902678 0.246711805  
## [142,] 1.89543421 -0.687273065 -0.129640697 0.468128374  
## [143,] 1.15401555 0.696536401 -0.528389994 -0.040385459  
## [144,] 2.03374499 -0.864624030 -0.337014969 0.045036251  
## [145,] 1.99147547 -1.045665670 -0.630301866 0.213330527  
## [146,] 1.86425786 -0.385674038 -0.255418178 0.387957152  
## [147,] 1.55935649 0.893692855 0.026283300 0.219456899  
## [148,] 1.51609145 -0.268170747 -0.179576781 0.118773236  
## [149,] 1.36820418 -1.007877934 -0.930278721 0.026041407  
## [150,] 0.95744849 0.024250427 -0.526485033 -0.162533529

* Visualise the PCA projection using plot().

plot(x = pca\_iris$x[,"PC2"], y = pca\_iris$x[,"PC1"] )

### Bonus point

* Edit the plot above, coloring data points according to their class label.

plot(x = pca\_iris$x[,"PC1"], y = pca\_iris$x[,"PC2"] , col = iris$Species )

## Exercise

### Variance explained

* Compute the variance explained by principal components, using information present in the return value of the prcomp() function.

pca\_iris$sdev

## [1] 1.7083611 0.9560494 0.3830886 0.1439265

explained\_variance\_ratio <- (pca\_iris$sdev^2)/ sum(pca\_iris$sdev^2)  
explained\_variance\_ratio

## [1] 0.729624454 0.228507618 0.036689219 0.005178709

* Visualise the variance explained by each principal component using barplot().

barplot(explained\_variance\_ratio )

# Exercise

## Hierarchical clustering

* Perform hierarchical clustering on the iris\_features data set, using the euclidean distance and method ward.D2. Use the functions dist() and hclust().

dist\_iris <- dist(iris\_features, method = "euclidean", diag = FALSE, upper=FALSE, p=2)  
hclust\_iris\_ward <- hclust(dist\_iris, "ward.D2", members = NULL )  
hclust\_iris\_ward

##   
## Call:  
## hclust(d = dist\_iris, method = "ward.D2", members = NULL)  
##   
## Cluster method : ward.D2   
## Distance : euclidean   
## Number of objects: 150

* Plot the clustering tree using plot().

plot(hclust\_iris\_ward, labels = NULL, hang =0.1, check = TRUE, axes = TRUE, frame.plot=FALSE, ann=TRUE, main ="Cluster Dendrogram", sub =NULL, xlab=NULL, ylab = "Height" )

How many clusters would you call from a visual inspection of the tree?

Answer:

* Cut the tree in 3 clusters and extract the cluster label for each flower. Use the function cutree().

iris\_hclust\_ward\_labels <- cutree(hclust\_iris\_ward, k = 3 )  
iris\_hclust\_ward\_labels

## [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [75] 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3 3 2 3 3 3 3  
## [112] 3 3 2 2 3 3 3 3 2 3 2 3 2 3 3 2 2 3 3 3 3 3 2 2 3 3 3 2 3 3 3 2 3 3 3 2 3  
## [149] 3 2

* Repeat clustering using 3 other agglomeration methods:
  + complete
  + average
  + single

# complete  
hclust\_iris\_complete <- hclust(dist\_iris, "complete", members = NULL )  
iris\_hclust\_complete\_labels <- cutree(hclust\_iris\_complete, k =3 )  
iris\_hclust\_complete\_labels

## [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 3 2 3 2 3 2 3 3 3 3 2 3 2 3 3 2 3 2 3 2 2  
## [75] 2 2 2 2 2 3 3 3 3 2 3 2 2 2 3 3 3 2 3 3 3 3 3 2 3 3 2 2 2 2 2 2 3 2 2 2 2  
## [112] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [149] 2 2

# average  
hclust\_iris\_average <- hclust(dist\_iris, "average", members = NULL )  
iris\_hclust\_average\_labels <- cutree(hclust\_iris\_average, k =3 )  
iris\_hclust\_average\_labels

## [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3 3 2 3 3 3 3  
## [112] 3 3 2 2 3 3 3 3 2 3 2 3 2 3 3 2 2 3 3 3 3 3 2 3 3 3 3 2 3 3 3 2 3 3 3 2 3  
## [149] 3 2

# single  
hclust\_iris\_single <- hclust(dist\_iris, "single", members = NULL )  
iris\_hclust\_single\_labels <- cutree(hclust\_iris\_single, k=3 )  
iris\_hclust\_single\_labels

## [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [112] 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [149] 2 2

* Compare clustering results on scatter plots of the data using plot().

par(mfrow = c(3, 2))  
palette(c("red", "blue", "green"))  
plot(x = pca\_iris$x[,"PC1"], y = pca\_iris$x[,"PC2"] , col =iris$Species)  
plot(x = pca\_iris$x[,"PC1"], y = pca\_iris$x[,"PC2"] , col =iris\_hclust\_ward\_labels )  
plot(x = pca\_iris$x[,"PC1"], y = pca\_iris$x[,"PC2"] , col =iris\_hclust\_complete\_labels)  
plot(x = pca\_iris$x[,"PC1"], y = pca\_iris$x[,"PC2"] , col =iris\_hclust\_average\_labels )  
plot(x = pca\_iris$x[,"PC1"], y = pca\_iris$x[,"PC2"] , col =iris\_hclust\_single\_labels )  
par(mfrow = c(1, 1))

table(iris\_hclust\_ward\_labels, iris$Species)

##   
## iris\_hclust\_ward\_labels setosa versicolor virginica  
## 1 50 0 0  
## 2 0 49 15  
## 3 0 1 35