UQ Winter School - The Basics, session 3

Kathryn Kemper

15/06/2022

Objective

This practical session will review some of the key concepts in the lectures today. Namely, start with a review of variance, covariance and correlation from lecture 2 and highlight some useful properties of variance that will be used later on the in practical. The the remainder of the session will focus on regression using matrix notation and finally using `lm()` and `predict.lm()` in R. By the end of the session you should be able to interpret the results from `lm()` and be able to reproduce many of the outputs by hand.

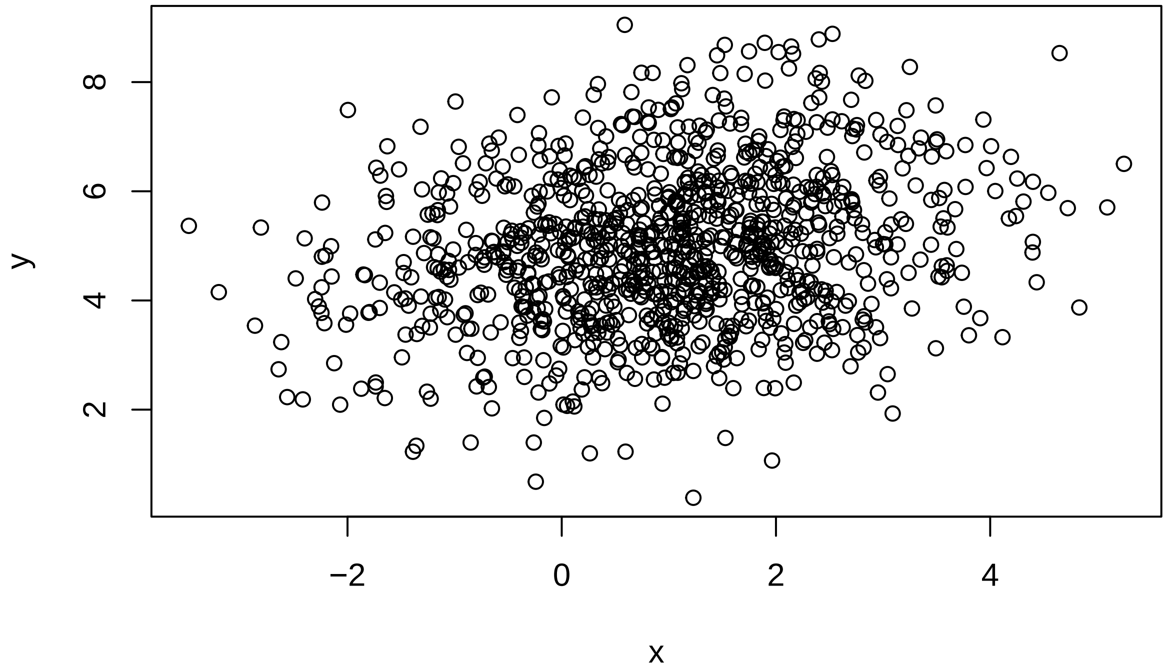
Q1: variance, covariance and correlation

Two variables, x and y, can show dependence on each other. That is, if we know the value of x we have some understanding of the value of y. The dependence between the two variables can be quantified using covariance (average sum of products) and correlation (a dimensionless value between 0 and 1). Covariance is typically hard to interpret on its own but can be viewed as an intermediary to calculating more interesting quantities, such as the correlation.

Q1. With the data provided, rename the data columns as x and y and calculate the variance, covariance and correlation for them. Try doing this by-hand and comparing the result to var() and corr() function in R. Create a new variable (z) which is the sum of x and y. What is the variance of z and how does this relate to the variance of x and y?

Note that you will need to modify the code to read the following directory:
`dir=/data/module1/basicsSession3/`

```
# read in the data, and name columns 1 and 2 as x and y
a = read.table(paste0(dir,"2_Q4.txt"))
x = a[,1]
y = a[,2]
plot(y~x)
```



```
# variance, covariance and correlation
n=length(x)
sum((x-mean(x))^2)/(n-1)
```

```
## [1] 1.991952
```

```
var(x)
```

```
## [1] 1.991952
```

```
sum((y-mean(y))^2)/(n-1)
```

```
## [1] 2.027531
```

```
var(y)
```

```
## [1] 2.027531
```

```
sum((y-mean(y))*(x-mean(x)))/(n-1)
```

```
## [1] 0.4704969
```

```r
var(x,y)
```

```
## [1] 0.4704969
```

```r
var(x,y)/sqrt(var(x)*var(y))
```

```
## [1] 0.2341174
```

```r
cor(x,y)
```

```
## [1] 0.2341174
```

We will now make the new variable $z = x + y$ and calculate its variance. A useful property is that the variance of the sum of variables is the sum of all possible variances and covariances. i.e.

$$var(x + y) = var(x) + var(y) + 2cov(x, y)$$

```r
#make a new variable z
z = x + y

#predict its variance, i.e. var(x+y) = var(x)+var(y)+2cov(x+y)
var(x) + var(y) + 2*cov(x,y)
```

```
## [1] 4.960477
```

```r
# compared to the direct calculation of its variance
var(z)
```

```
## [1] 4.960477
```

Can you write your own function to calculate the variance between two variables?

```r
covariance<-function(x,y) {
  mu1=mean(x)
  mu2=mean(y)
  n=length(x)
  sum((x-mu1)*(y-mu2))/(n-1)
}
covariance(x,y)
```

```
## [1] 0.4704969
```

Other useful properties of variances, where X and Y are random variables and a and b are constants, include:
(1) $Var(x + a) = Var(x)$
(2) $Var(ax) = a^2Var(x)$
(3) $Var(aX + bY) = a^2Var(X) + b^2Var(Y) + 2abCov(X, Y)$
(4) $Var(aX - bY) = a^2Var(X) + b^2Var(Y) - 2abCov(X, Y)$

Q2: simple linear regression (SLR)

SLR means fitting a simple model regressing y on x, fitting an intercept and slope. Note that the independent variable x is usually the ascertained or predictor variable. It is the variable influenced by experimental design. y is the dependent or response variable.

Q2. Construct and find the regression co-efficients (intercept and slope) for the data provided in Q1. Calculate the variance explained by the regression and plot the results. Compare your results to using the `lm()` function in R.

As a reminder, the least squares equation for the regression co-efficient is:

$$\hat{\beta} = [X'X]^{-1} X'Y$$

```
n=length(y)
Y = matrix(y,ncol=1)

 #set up  X, our model only has the intercept (mean) plus 'x' variable
X1 =  matrix(rep(1,n),ncol=1)
X2 = matrix(x,ncol=1)
X = cbind(X1,X2)

# set up the matrices
XpX = t(X)%*%X ; XpX
```

```
##           [,1]      [,2]
## [1,] 1000.0000  987.5228
## [2,]  987.5228 2965.1611
```

```
XpXinv = solve(XpX)
XpY = t(X)%*%Y ; XpY
```

```
##          [,1]
## [1,] 4979.309
## [2,] 5387.208
```

Here its useful to look at the matrices we have constructed and what they are doing. XpX (or $X'X$) is a symmetrical square 2x2 matrix with the following elements,
(1) $X'X_{1,1} = n$
(2) $X'X_{1,2} = sum(x)$,
(3) $X'X_{2,2} = sum(x^2)$

XpY (or $X'Y$) is a 2x1 column vector with the following elements,
(1) $X'Y_{1,1} = sum(y)$
(2) $X'Y_{2,1} = sum(y * x)$

For a full derivation of the equivalence between the scalar solution for $\hat{\beta}$, i.e. $\hat{\beta} = \frac{SSQ_{xy}}{SSQ_x}$ and the least-squares estimators in matrix form see page 5 of the following:
https://www.stat.purdue.edu/~boli/stat512/lectures/topic3.pdf

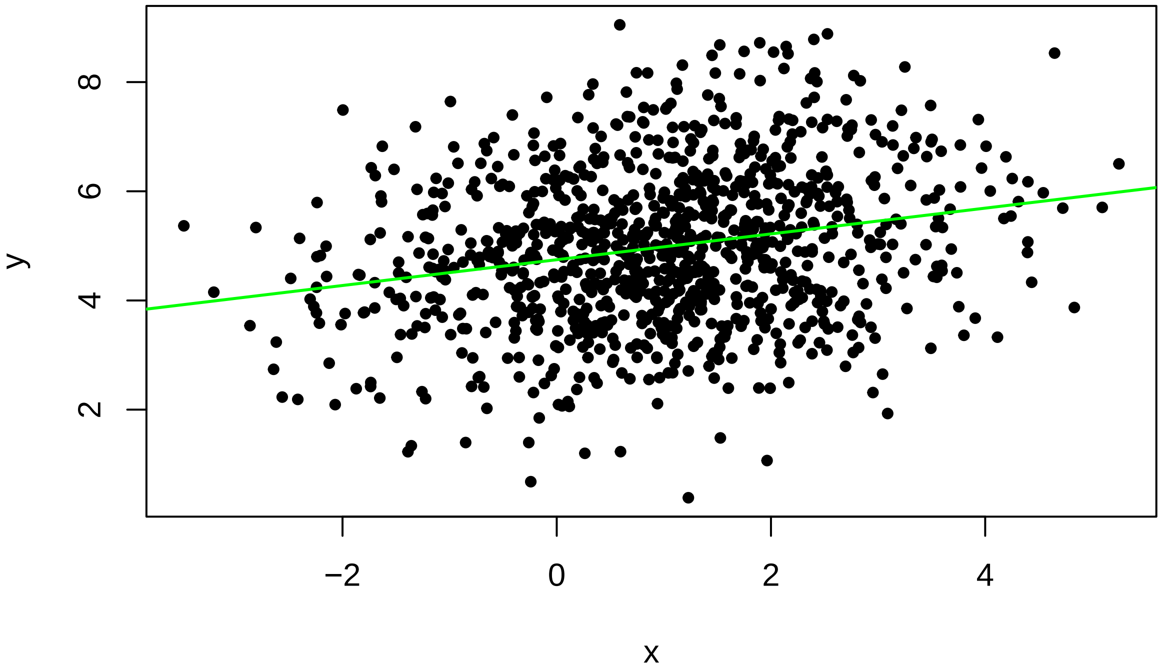
```
# solve
beta = XpXinv %*% XpY
beta
```

```
##          [,1]
## [1,] 4.746057
## [2,] 0.236199
```

```
#lets compare our results to  using lm()
lm1=lm(y~x)
summary(lm1)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6501 -0.9534  0.0087  0.8941  4.1634
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.74606    0.05346  88.771  < 2e-16 ***
## x            0.23620    0.03105   7.607 6.45e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.385 on 998 degrees of freedom
## Multiple R-squared:  0.05481,    Adjusted R-squared:  0.05386
## F-statistic: 57.87 on 1 and 998 DF,  p-value: 6.449e-14
```

```
# plot our points and the regression line
plot(y~x, pch=20)
abline(a=beta[1],b=beta[2],col="green",lwd=1.5)
```



Try `plot(linearModelObject)` at the command line to see R's diagnostic plots of the residuals. In Q5 below we will make the QQ-plot by hand.

Q3: model fit and s.e. of coefficients

Recall that model fit can be assessed by R^2 as:

$$R^2 = \frac{SSQ_{reg}}{SSQ_{total}}$$

where the SSQ_{total} and SSQ_{reg} are calculated in a very similar manner to the ANOVA test. Thus $SSQ_{total} = \sum(y - \bar{y})^2$, $SSQ_{resid} = \sum(y - \hat{y})^2$ and $SSQ_{reg} = SSQ_{tot} - SSQ_{resid}$.

Q3. In your regression, what proportion of the variance in y is accounted for by x? Is this relationship significant? What are the s.e. for your regression co-efficients.

```
#predicted values
H = X%*%XpXinv%*%t(X)
yHat = H%*%Y
# yHat = beta[1]+beta[2]*x # alternative
e = y - yHat

SSQreg = sum((yHat-mean(y))^2)
```

```r
SSQtot = sum((y-mean(y))^2)
R2 = SSQreg/SSQtot ; R2
```

```
## [1] 0.05481095
```

```r
#compare to the R output
summary(lm1)$r.squared
```

```
## [1] 0.05481095
```

Note that the structure of an object in R can be accessed using `str(object)`. Useful quantities associated with the `summary(linearModelObject)` function of the linear model include the residuals, and the coefficients table.

We will now conduct the F-test for the regression.

```r
SSQe = SSQtot - SSQreg
F = SSQreg/(2-1) * (n-2)/SSQe ; F
```

```
## [1] 57.87342
```

```r
1-pf(F,1,n-2)
```

```
## [1] 6.450396e-14
```

We saw in the lectures that the variance of the regression coefficients is given by:

$$var(\hat{\beta}) = [X'X]^{-1}\,\sigma_e^2$$

The standard errors of the regression co-efficients are square-root of the diagonal elements of this variance-covariance matrix.

```r
sigmaE=sum(e^2)/(n-2)
Vcov = XpXinv*sigmaE
se = sqrt(diag(Vcov))
```

Which is in agreement with the coefficients table in `summary()`. From here we already know how to conduct a t-test, and produce p-values. Hence,

```r
tValue = beta/se
2*pt(tValue, df=998, lower.tail =FALSE)
```

```
##              [,1]
## [1,] 0.000000e+00
## [2,] 6.448574e-14
```

Q4: s.e. for the predicted values

Q4. What is the s.e. around the predicted values. Plot 95% confidence intervals on your regression plot.

This problem requires determining $var(\hat{Y})$, remembering that $\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 x$ and that each of our coefficients has a standard error surrounding it.

Recall the property that $Var(aX + bY) = a^2Var(X) + b^2Var(Y) + 2abCov(X, Y)$. We can use this property to determine the variance of $\hat{Y}$ as it is a simple linear combination of terms. Then

$$Var(\hat{Y}) = Var(\hat{\beta}_0) + x^2Var(\hat{\beta}_1) + 2xcov(\hat{\beta}_0, \hat{\beta}_1)$$

95% confidence intervals are calculated as $\hat{y} \pm 1.96s.e.$ (can you recall why it is 1.96?). We show above the calculation of the $var(\hat{Y})$ and have already stored the variance-covariance matrix ('Vcov') from the previous question. The final step is to define an x value range for the predictions and the CI. We can compare our answer to the `predict.lm()` function in R.

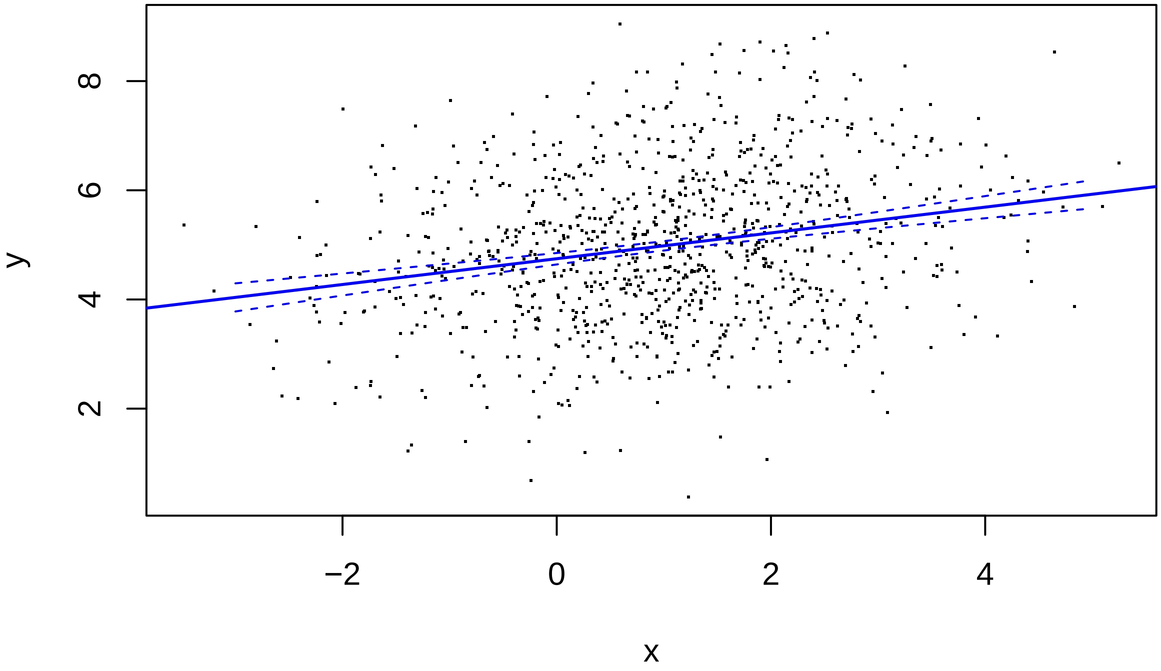
```
#range over which we want to make confidence intervals
new = data.frame(x=seq(-3,5,0.1))

# se of a predicted value
seYhat <- function(x) {
  sqrt(Vcov[1,1] + x^2*Vcov[2,2] + 2*x*Vcov[1,2])
}

pred = beta[1] + beta[2]*new$x
seFit = seYhat(new$x)
upper = pred + qt(0.975,df=998)*seFit
lower = pred - qt(0.975,df=998)*seFit

#our s.e. of the predicted values
head(seFit)
```

```
## [1] 0.1313249 0.1284020 0.1254879 0.1225831 0.1196884 0.1168044
```

```
#using predict.lm() function
head(predict.lm(lm1,new,se.fit = TRUE)$se.fit)
```

```
##         1         2         3         4         5         6
## 0.1313249 0.1284020 0.1254879 0.1225831 0.1196884 0.1168044
```

```
# plot
plot(y~x, pch=".")
abline(a=beta[1],b=beta[2],col="blue",lwd=1.5)
points(upper~new$x,col="blue",lwd=1,lty=2,type="l")
points(lower~new$x,col="blue",lwd=1,lty=2,type="l")
```



Q5: constructing a QQ plot

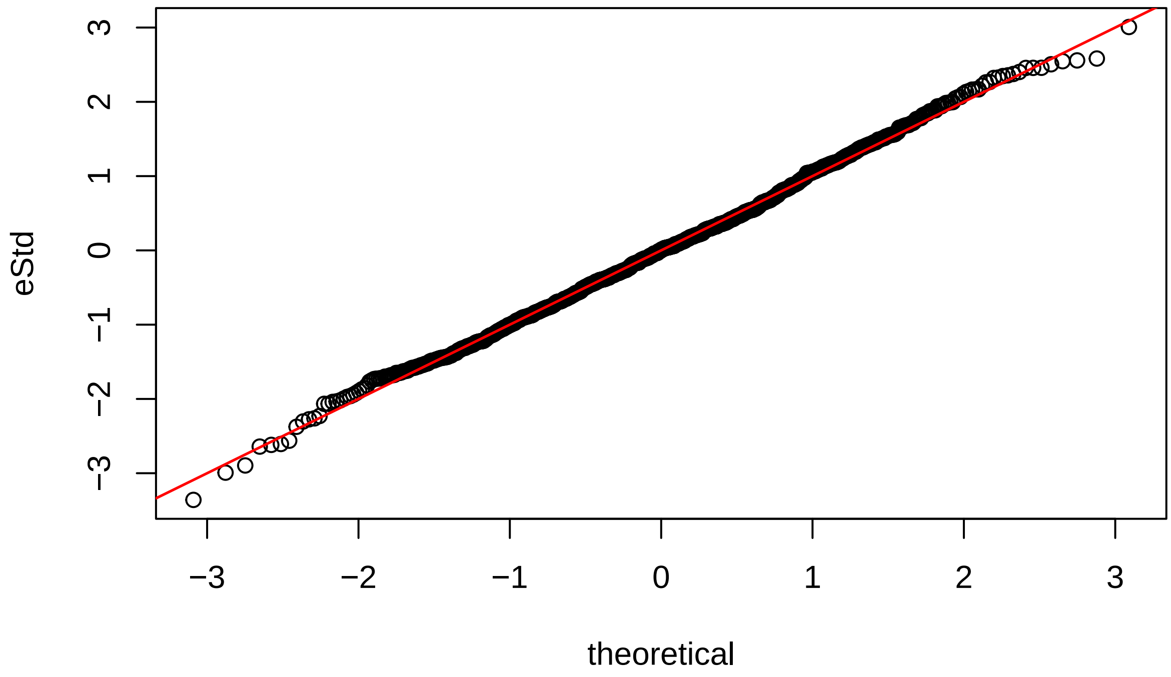
QQ-plots are often used as diagnostic tools in linear models to check the assumption of normality for the residuals. They can also be easily applied using distributions, and are often examined in GWAS to check for departures from the expected distribution of the test statistic. QQ-plots are subjective but give a quick visual impression of the distribution.

QQ-plots involved plotting standardized residuals ($y - axis$) on a standard normal curve or theoretical quartiles for the data points. The expectation (if the residuals are distributed according to the distribution tested) is that the data points will lie on a straight y = x line.

Q5. Make a QQ-plot for the residuals from our model. Do they meet the assumptions of the model?

```
 # first we need to sort and standardize our residuals to z-scores
eStd = sort(e)/sd(e)

# next we calculate the theoretical distribution of z-scores, for the given sample size
# note here we use n+1 to prevent returning -Inf value
quartiles = 1:n/(n+1)
theoretical = qnorm(quartiles)

# plot the result, with the expected y = x line
plot(eStd ~ theoretical)
```

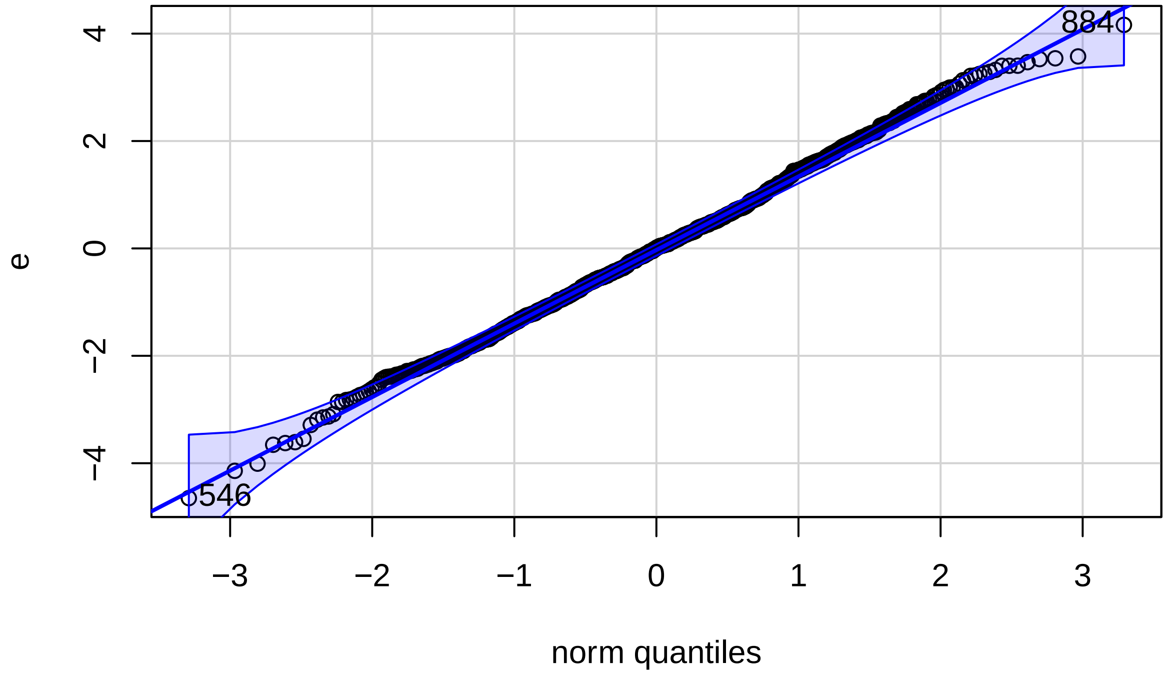
```r
abline(0,1, lwd=1.3, col="red",
       xlab="Theoretical Quartiles", ylab = "Sample Quartiles")
```



```r
# or use qqPlot in the cars package
library(car)
```

```
## Loading required package: carData
```

```r
qqPlot(e)
```



```
## [1] 546 884
```