

# EPIB-613 - Basic Analysis and Graphics

Today we will cover:

1. Basic Statistics: Covers t-tests with confidence intervals for means, chi-square tests with confidence intervals for proportions, Fisher's Exact Test, non-parametrics, area under probability density curves (normal, t, F), quantiles, probability functions (binomial), and linear, logistic and Poisson regression.
2. Graphics: Scatter plots, histograms, boxplots, 3-D plots, dotplots, and barplots.

# 1 Basic Statistical Routines

You can use basic R (without add-on packages) to do any of the standard statistical routines you will typically encounter. In this section we will see various functions that will be needed in epidemiology.

First, we will create some data so we can apply our tests.

```
> x<-rnorm(10, mean=3, sd=2)
> x
 [1] 2.135011 7.573118 3.840814 5.544555 3.393277 2.731562
 [6] 1.780588 4.576343 3.250077 3.908082
> y<-rbinom(10, size=1, prob=0.4)
> y
 [1] 0 1 1 1 0 0 1 0 1 0
> z<- 5 + 2*x + 3*y +rnorm(10, mean=0, sd=0.5)
> z
 [1] 9.44732 23.30763 15.34688 19.62479 11.30383 10.89258
 [6] 11.22488 13.33018 15.11767 12.81894
```

```
> my.data.frame<-data.frame(x,y,z)
> my.data.frame
      x y      z
1 2.135011 0 9.44732
2 7.573118 1 23.30763
3 3.840814 1 15.34688
4 5.544555 1 19.62479
5 3.393277 0 11.30383
6 2.731562 0 10.89258
7 1.780588 1 11.22488
8 4.576343 0 13.33018
9 3.250077 1 15.11767
10 3.908082 0 12.81894
```

Now to see some commands for basic statistical routines, to calculate confidence intervals (and much less useful  $p$ -values):

## T-Tests and Nonparametric Wilcoxon Test with Confidence Intervals

```
> group1 <- x[y==1]
```

```

> group0 <- x[y==0]
> group1
[1] 7.573118 3.840814 5.544555 1.780588 3.250077
> group0
[1] 2.135011 3.393277 2.731562 4.576343 3.908082
> t.test(group1, group0)

```

Welch Two Sample t-test

```

data: group1 and group0 t = 0.9667, df = 5.431, p-value = 0.3748
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.675244  3.773195
sample estimates: mean of x mean of y
 4.397831  3.348855

```

The same data can be analyzed using nonparametric methods:

```

> wilcox.test(group1, group0, conf.int=T)

```

Wilcoxon rank sum test

```

data: group1 and group0
W = 15, p-value = 0.6905
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
 -1.612689  4.179841
sample estimates:
difference in location
          0.968212

```

Note similar but not identical CI as that given by *t*-test.

### Chi-square tests

Suppose you have a two-by-two table of data, such as:

	Patient Independent	Patient Dependent
Stroke Unit	67	34
Medical Unit	46	45

You can simply enter:

```
> prop.test(c(67,46), c(67+34, 46+45) )
```

2-sample test for equality of proportions with continuity correction

```
data: c(67, 46) out of c(67 + 34, 46 + 45) X-squared = 4.2965, df
= 1, p-value = 0.03819 alternative hypothesis: two.sided 95
percent confidence interval:
 0.009420794 0.306322868
sample estimates:
  prop 1    prop 2
0.6633663 0.5054945
```

**More generally:**

```
> smokers <- rbinom(100, 1, 0.2)
> non.smokers<-rbinom(200, 1, 0.12)
> prop.test(c(sum(smokers), sum(non.smokers)), c(100,200), correct=F)
```

2-sample test for equality of proportions without continuity correction

```
data: c(sum(smokers), sum(non.smokers)) out of c(100, 200)
X-squared = 2.1319, df = 1, p-value = 0.1443 alternative
hypothesis: two.sided 95 percent confidence interval:
-0.02659294 0.15659294
sample estimates: prop 1 prop 2
 0.200 0.135
```

**Fisher's Exact Test**

Again consider the two-by-two table of data

	Patient Independent	Patient Dependent
Stroke Unit	67	34
Medical Unit	46	45

Unlike the Chi-Square test, here you need to have the data in Matrix format.

```
data.fisher <- matrix(c(67,34,46,45), nrow=2, byrow=T)
> data.fisher
  [,1] [,2]
```

```
[1,] 67 34
[2,] 46 45
```

Now that you have the data in the right format, you can simply enter:

```
> fisher.test(data.fisher )
```

```
Fisher's Exact Test for Count Data
```

```
data: data.fisher
p-value = 0.02865
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 1.034074 3.599766
sample estimates:
odds ratio
 1.921062
```

Note that the CI is for the Odds Ratio.

## Area Under Probability Curves, Probability Functions, and Quantiles

R can provide areas under the curve for any continuous density, and also provide any quantiles for any density. We will now see some examples for normal,  $t$  and  $F$  densities.

Suppose we have a  $N(\mu = 2, \sigma = 5)$  density, and we want the area under this normal curve from, say,  $-\infty$  to 0, or from 0 to 3, or from -7 to 1. We can type

```
> pnorm(0, mean=2, sd=5)
[1] 0.3445783
> pnorm(3, mean=2, sd=5) - pnorm(0, mean=2, sd=5)
[1] 0.2346815
> pnorm(1, mean=2, sd=5) - pnorm(-7, mean=2, sd=5)
[1] 0.38481
```

Note that each gives the area from  $-\infty$  to the first number plugged into the function, so subtracting gives areas between two numbers.

Very similarly for the  $t$  distribution, except instead of mean and variance, we need to provide the degrees of freedom to define which  $t$  distribution we are using. Suppose we want to know

the area under the  $t$  distribution from  $-\infty$  to 1 with 4 degrees, or between 2 and 3 with 100 degrees of freedom. We can type

```
> pt(1, df=4)
[1] 0.8130495
> pt(3, df=100) - pt(2, df=100)
[1] 0.02240213
```

Similarly for the  $F$  density, but now we need to supply two degrees of freedom (one for the numerator, one for the denominator). Suppose we want to area above 1, in a  $F$  density with 100 and 200 degrees of freedom. We can type

```
> 1-pf(1, df1 = 100, df2=200)
[1] 0.4923146
```

Note the subtraction from 1, because we want the upper tail, while default in all of these functions is to give the lower tail probability.

Quantiles use the same function name, but changing the "p" each time for a "q". Some examples:

```
> qnorm(0.975, mean = 0, sd=1)
[1] 1.959964
> qnorm(0.01, mean = 1, sd=5)
[1] -10.63174
> qt(0.975, df=4)
[1] 2.776445
> qf(0.95, df1=10, df2=10)
[1] 2.978237
```

The same applies to discrete probabilities, such as the binomial:

```
> args(pbinom)
function (q, size, prob, lower.tail = TRUE, log.p = FALSE)
> pbinom(0:10, 10, 0.6)
[1] 0.0001048576 0.0016777216 0.0122945536 0.0547618816 0.1662386176
[6] 0.3668967424 0.6177193984 0.8327102464 0.9536425984 0.9939533824 1.0000000000
> dbinom(0:10, 10, 0.6)
```

```

[1] 0.0001048576 0.0015728640 0.0106168320 0.0424673280 0.1114767360
[6] 0.2006581248 0.2508226560 0.2149908480 0.1209323520 0.0403107840 0.0060466176
args(qbinom)
function (p, size, prob, lower.tail = TRUE, log.p = FALSE)
> qbinom(seq(0, 1, by = 0.2), 10, 0.6)
[1] 0 5 6 6 7 10

```

## Simple Linear Regression

Recall our data frame:

```

> my.data.frame
      x y      z
1  2.135011 0  9.44732
2  7.573118 1 23.30763
3  3.840814 1 15.34688
4  5.544555 1 19.62479
5  3.393277 0 11.30383
6  2.731562 0 10.89258
7  1.780588 1 11.22488
8  4.576343 0 13.33018
9  3.250077 1 15.11767
10 3.908082 0 12.81894

```

Suppose we want simple linear regression model of  $z$  on  $x$  and  $y$ .

```

> lin.reg<-lm(z ~ x + y)
> summary(lin.reg)

```

Call:

```
lm(formula = z ~ x + y)
```

Residuals:

Min	1Q	Median	3Q	Max
-0.6821	-0.4339	0.0893	0.3849	0.5679

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	4.8644	0.4245	11.460	8.66e-06 ***
x	1.9989	0.1064	18.792	3.00e-07 ***
y	3.2690	0.3450	9.475	3.05e-05 ***

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5162 on 7 degrees of freedom

Multiple R-Squared: 0.9889, Adjusted R-squared: 0.9857

F-statistic: 311.6 on 2 and 7 DF, p-value: 1.444e-07

Recall that z was generated by

```
> z<- 5 + 2*x + 3*y +rnorm(10, mean=0, sd=0.5)
```

so our estimates are close to expected (including sd estimate).

Other types of regression equations are available through generalized linear models:

For logistic regression, type (assuming z is dichotomous)

```
> glm(z ~ x + y, family=binomial(link="logit"))
```

For poisson regression, type (assuming z is a count variable)

```
> glm(z ~ x + y, family=poisson(link = "log"))
```

and so on.

## 2 Graphics

R is capable of producing very high quality graphics in a variety of formats (wmf, ps, jpg, pdf, png, bmp and tiff).

For a scatter plot, simply type:

```
> plot(x,z)
```



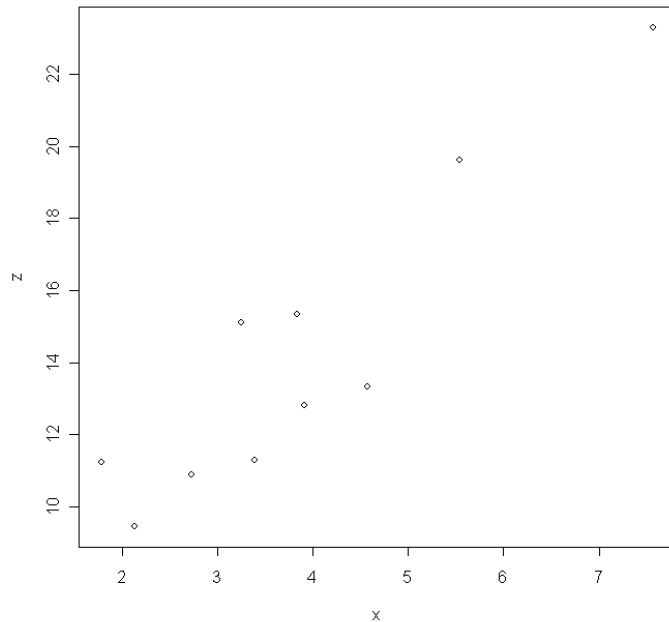


Figure 1: A simple scatter plot

Adding titles and labels for each axis is easy:

```
> plot(x,z, main="Scatter plot of age versus outcome", xlab="age", ylab="outcome")
```

Adding the best fitting regression line to the plot as well:

```
> plot(x,z, main="Scatter plot of age versus outcome", xlab="age", ylab="outcome")
> lsfit(x,z)$coef
Intercept      X
 5.236471  2.324865
> age.range <- seq(2,7,by=0.01)
> outcome.fit <- 5.236471 + 2.324865*age.range
> points(age.range, outcome.fit, type="l")
```

You can plot more than one curve on a single plot, and label them via a legend:

```
> range <- seq(-10, 10, by = 0.001)
> norm1 <- dnorm(range, mean=0, sd=1)
```

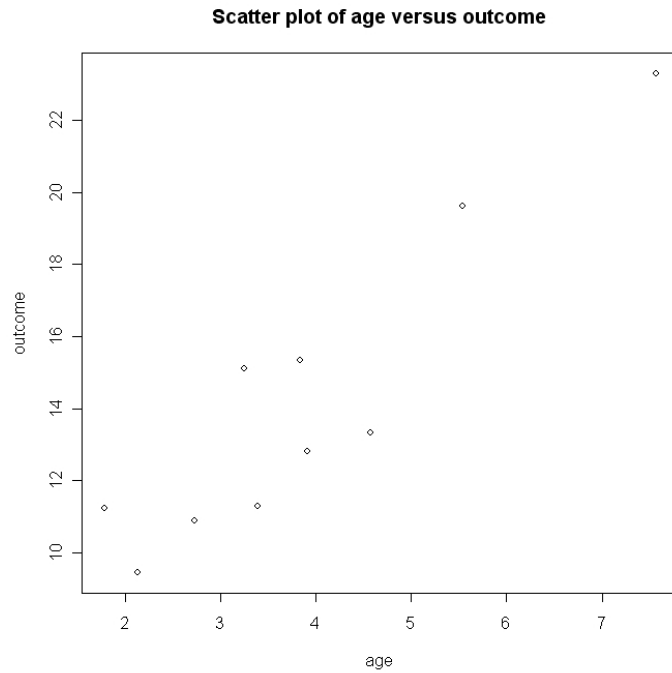


Figure 2: A simple scatter plot with labels and title

```
> norm2 <- dnorm(range, mean=1, sd=2)
> plot(range, norm1, type="l", lty=1, main="Two Normal Distributions",
       xlab="Range", ylab="Probability Density")
> points(range, norm2, type="l", lty=2)
> legend(x=-10, y=0.4, legend= c("N(0,1)", "N(1,2)"), lty=c(1,2))
```

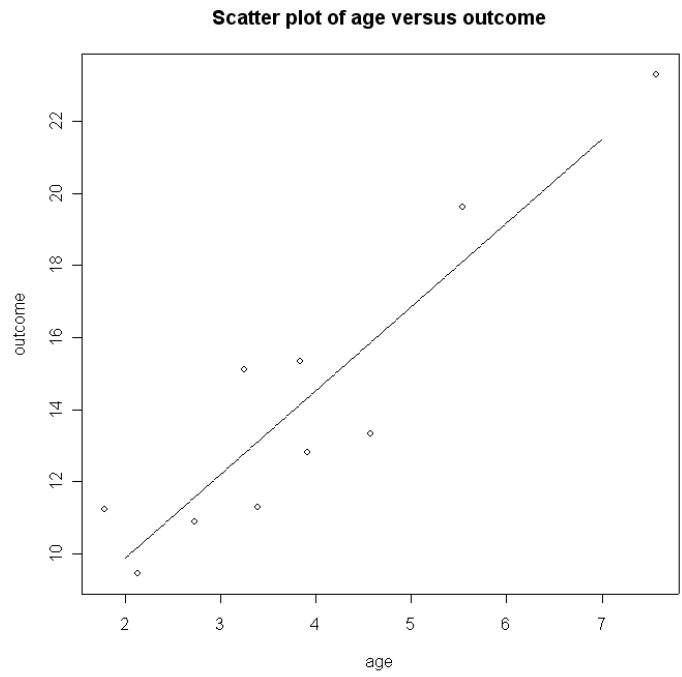


Figure 3: Scatter plot with best fitting line added

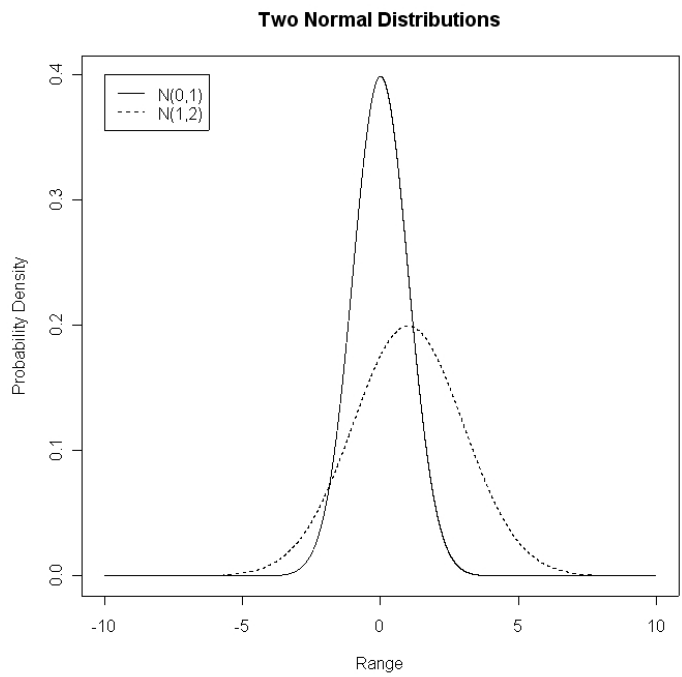


Figure 4: Distributions with legend

You also do histograms ...

```
hist(rnorm(1000,mean=0,  
sd=1), main = "Sample From N(0,1)  
Distribution", xlab = "Range", ylab="Frequency")
```

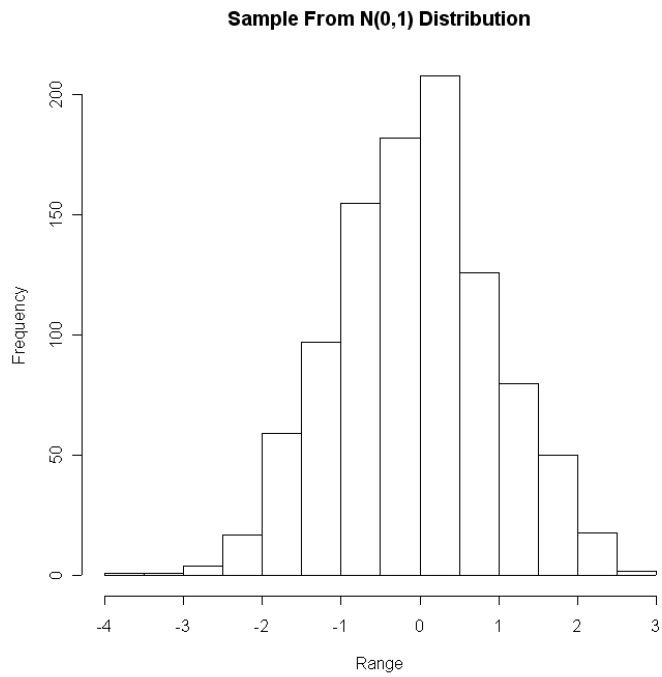


Figure 5: Histogram

...and boxplots.

```
> boxplot(rnorm(1000, mean=0, sd=1), rnorm(1000, mean=1, sd=2),  
          names=c("N(0,1)", "N(1,2)"))
```

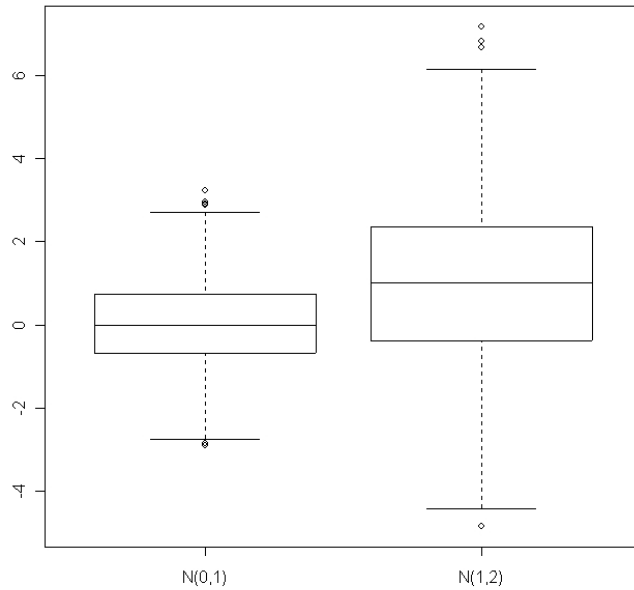


Figure 6: Boxplot

To save any graph, make sure the focus window is the graph, and use the “file → save as” menu item, and save to whatever format you prefer. Available formats include wmf, ps, jpg, pdf, png, bmp and tiff.

Plots perhaps less used in epidemiology are also available, including 3-D plots, dot plots and barplots. Here are some examples:

## A perspective plot in 3-D

```
x <- seq(-10, 10, length= 30)
y <- x
f <- function(x,y) { r <- sqrt(x^2+y^2); 10 * sin(r)/r }
z <- outer(x, y, f)
z[is.na(z)] <- 1
persp(x, y, z, theta = 30, phi = 30, expand = 0.5, col = "lightblue",
      ltheta = 120, shade = 0.75, ticktype = "detailed",
      xlab = "X", ylab = "Y", zlab = "Sinc( r )"
    )
```

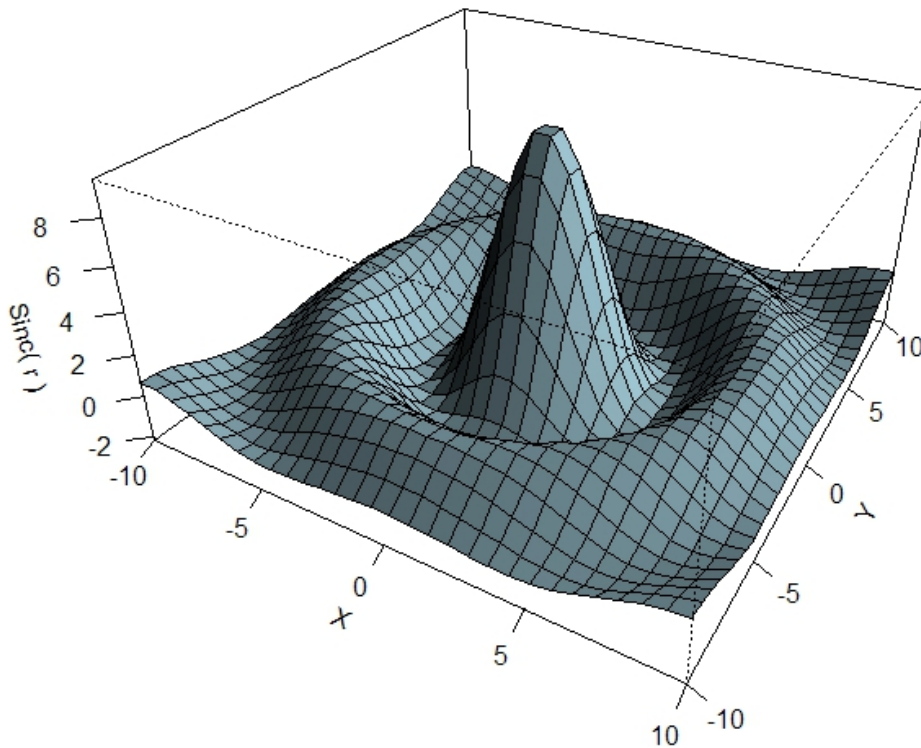


Figure 7: A 3-D plot using the persp command

## Dot plots

There used to be a dotplot command, but it has recently been replaced by the dotchart command. Here is an example, using an already existing (built-in) data set:

```
dotchart(t(VADeaths), xlim = c(0,100),  
        main = "Death Rates in Virginia - 1940")
```

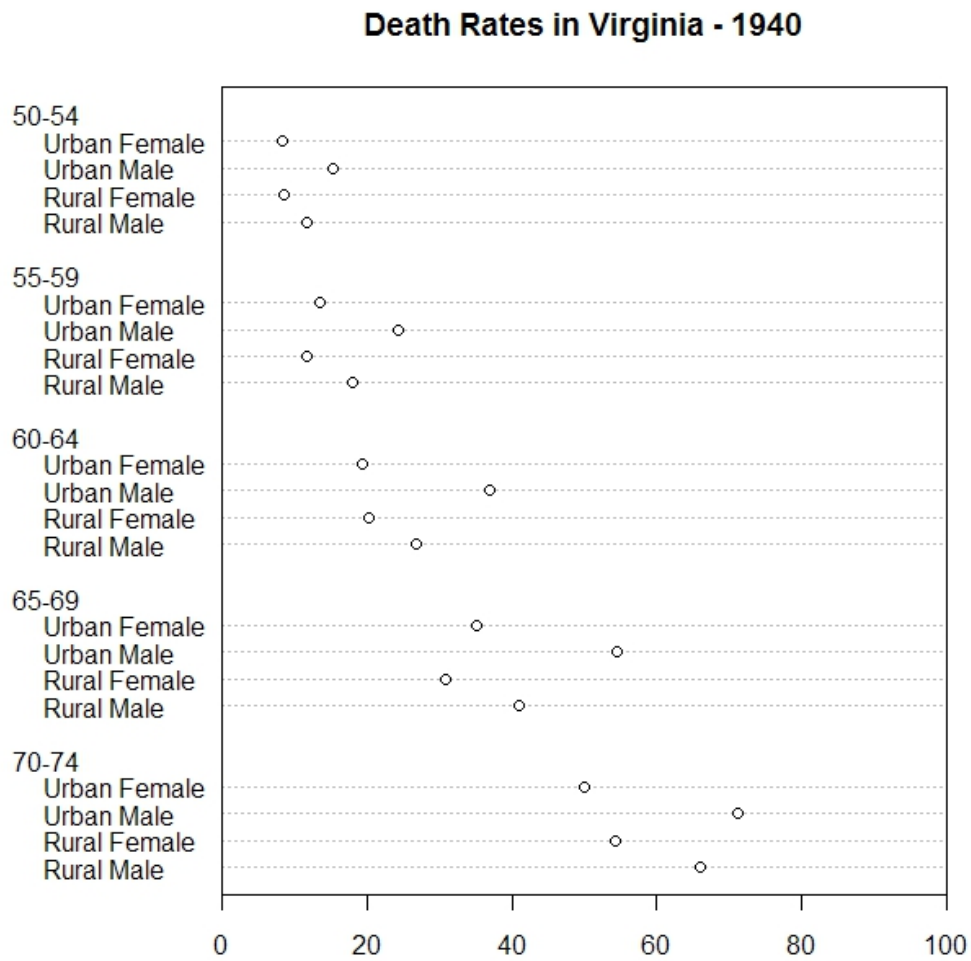


Figure 8: Example of a dot plot

## Bar plots

Here is an example of a barplot:

```
barplot(VADeaths)
```

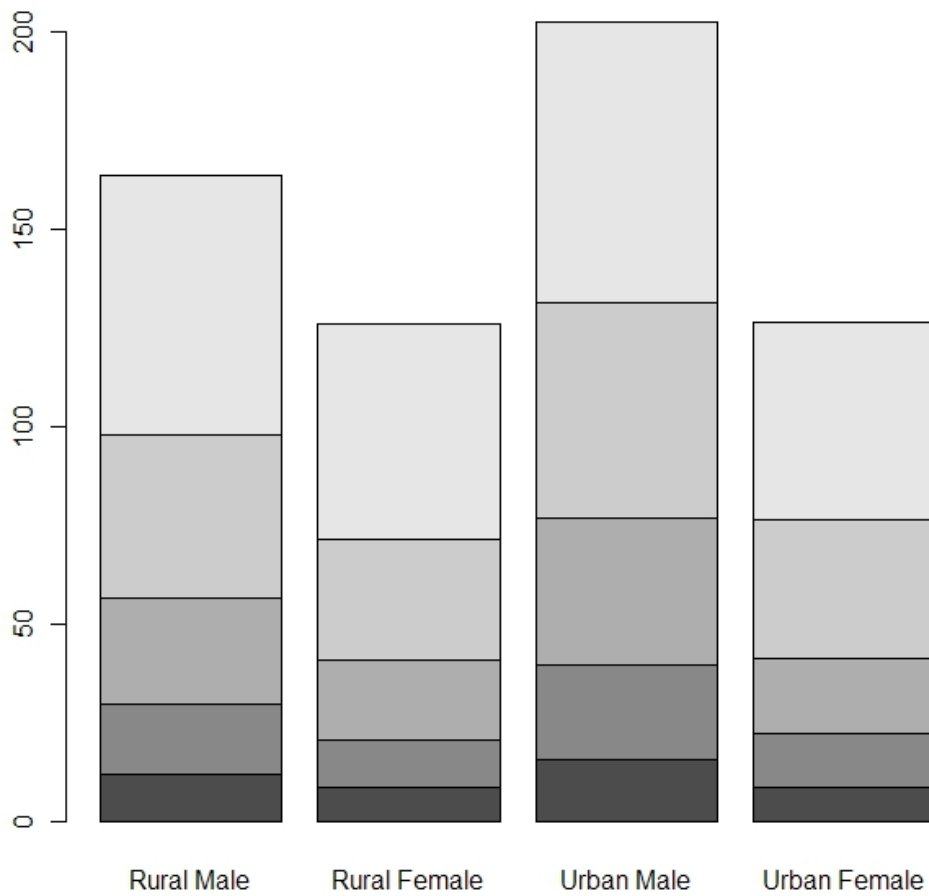


Figure 9: Example of a barplotd