

Probability Plots

Source

Reprinted with permission from Quick-R (<http://www.statmethods.net>).
© Copyright 2008 Robert I. Kabacoff, Ph.D. All rights reserved.

Background

This section describes creating probability plots in R for both didactic purposes and for data analyses. R makes it easy to draw probability distributions and demonstrate statistical concepts. Some of the more common probability distributions available in R are given below.

distribution	R name
Beta	beta
Binomial	binom
Cauchy	cauchy
Chisquare	chisq
Exponential	exp
F	f
Gamma	gamma
Geometric	geom
Hypergeometric	hyper
Logistic	logis

distribution	R name
Lognormal	lnorm
Negative Binomial	nbinom
Normal	norm
Poisson	pois
Student t	t
Uniform	unif
Tukey	tukey
Weibull	weib
Wilcoxon	wilcox

For a comprehensive list, see Statistical Distributions on the R wiki. The functions available for each distribution follow this format:

name	description
dname()	density or probability function
pname()	cumulative density function
qname()	quantile function
rname()	random deviates

For example, $\text{pnorm}(0) = 0.05$ (the area under the standard normal curve to the left of zero). $\text{qnorm}(0.9) = 1.28$ (1.28 is the 90th percentile of the standard normal distribution). $\text{rnorm}(100)$ generates 100 random deviates from a standard normal distribution.

Each function has parameters specific to that distribution. For example, $\text{rnorm}(100, m=50, sd=10)$ generates 100 random deviates from a normal distribution with mean 50 and standard deviation 10.

You can use these functions to demonstrate various aspects of probability distributions. Two common examples are given below.

Student's t Distribution

Display the Student's t distributions with various degrees of freedom and compare to the normal distribution

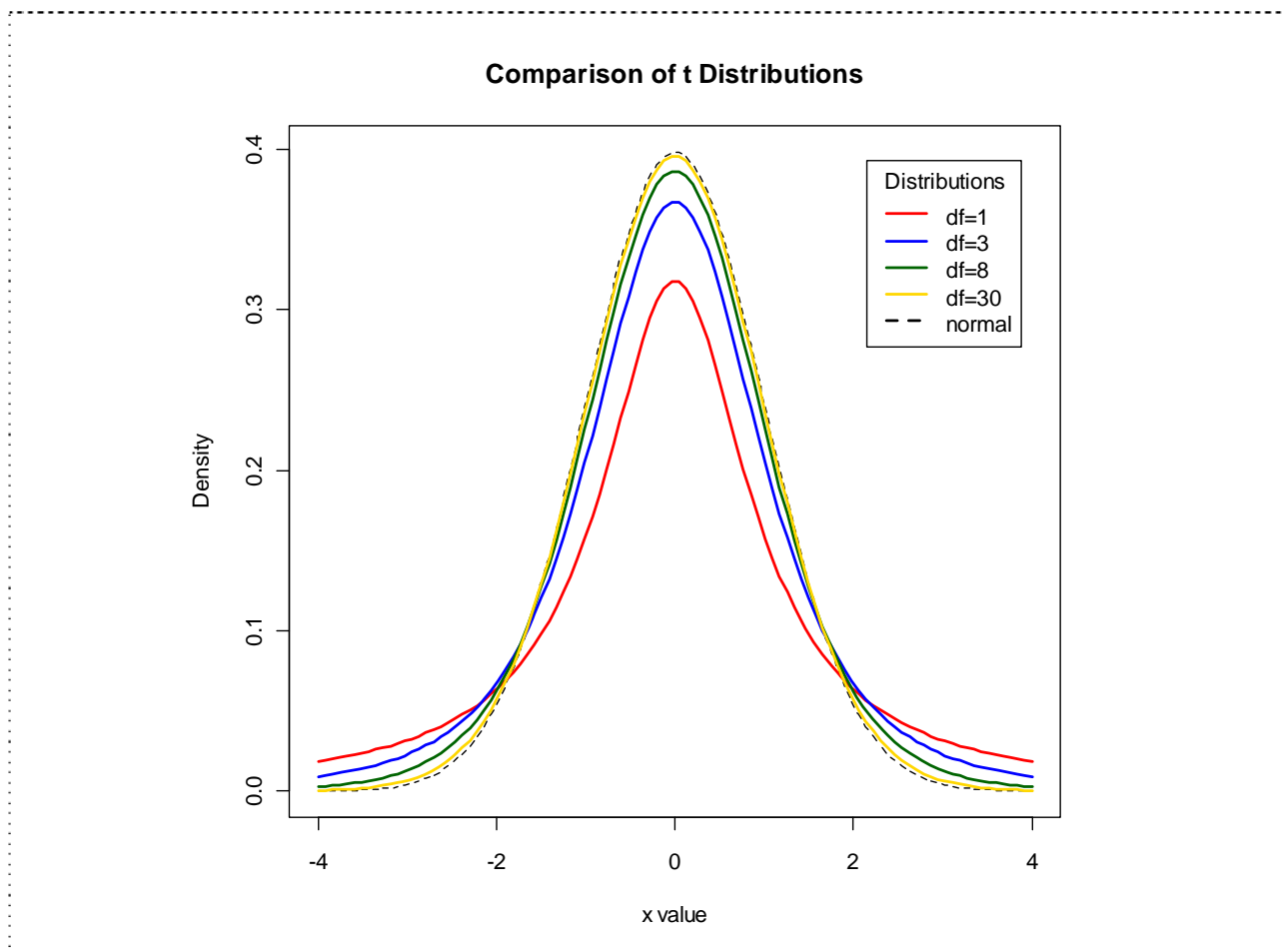
```
x <- seq(-4, 4, length=100)
hx <- dnorm(x)

degf <- c(1, 3, 8, 30)
colors <- c("red", "blue", "darkgreen", "gold", "black")
labels <- c("df=1", "df=3", "df=8", "df=30", "normal")

show <- function()
{
  plot(
    x,
    hx,
    type="l",
    lty=2,
    xlab="x value",
    ylab="Density",
    main="Comparison of t Distributions"
  )

  for (i in 1:4)
  {
    lines(
      x,
      dt(x,degf[i]),
      lwd=2,
      col=colors[i]
    )
  }
  legend(
    "topright",
    inset=.05,
    title="Distributions",
    labels,
    lwd=2,
    lty=c(1, 1, 1, 1, 2),
    col=colors
  )
}

show()
```



Integrate Area Under a Distribution

Children's IQ scores are normally distributed with a mean of 100 and a standard deviation of 15. What proportion of children are expected to have an IQ between 80 and 120?

```
mean=100
```

```
sd=15
```

```
lb=80
```

```
ub=120
```

```
x <- seq(-4,4,length=100)*sd + mean
```

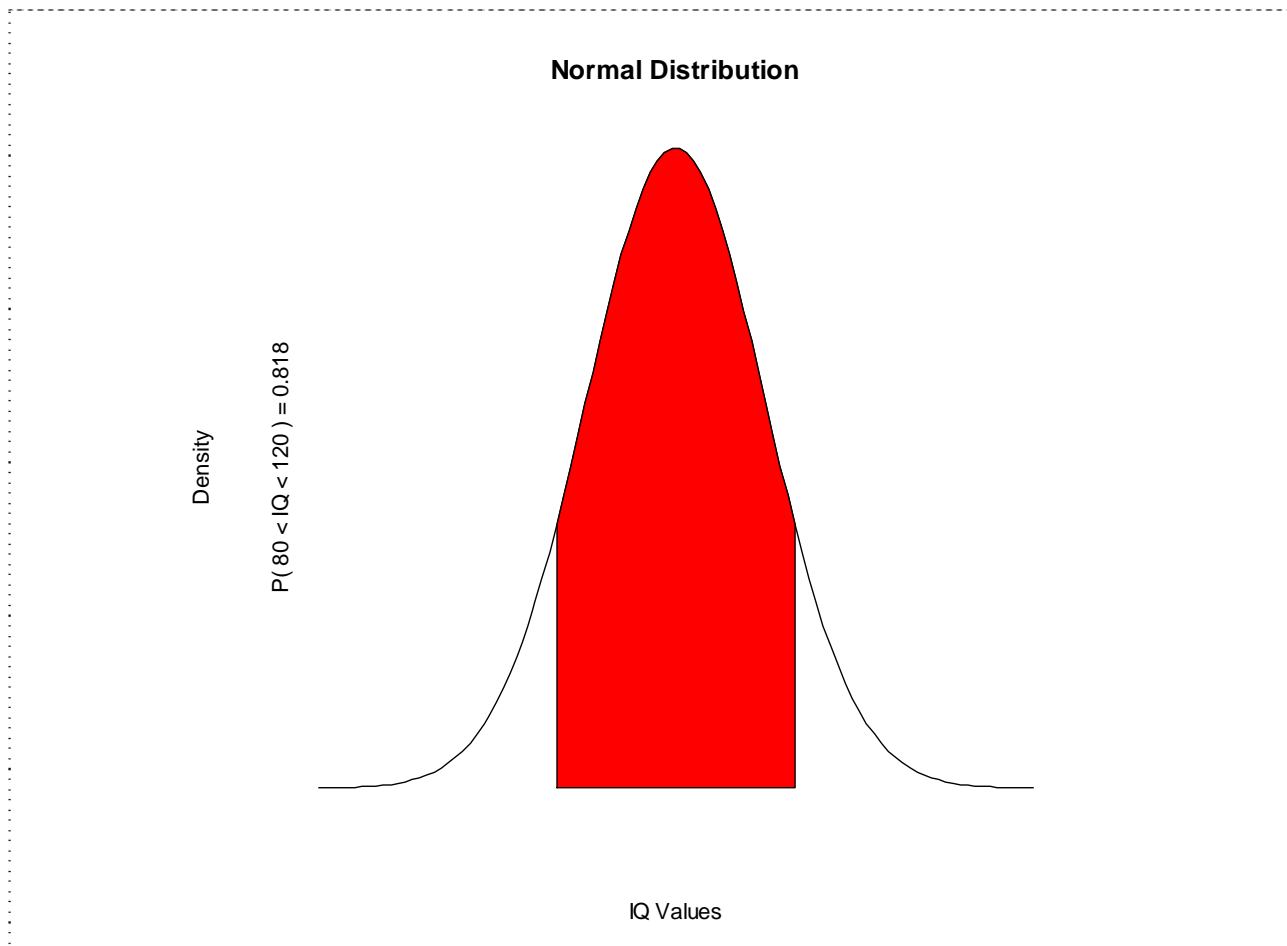
```
hx <- dnorm(x,mean,sd)
```

```
show <- function()
```

```
{
  plot(
    x,
    hx,
    type="n",
    xlab="IQ Values",
    ylab="Density",
    main="Normal Distribution",
    axes=FALSE
  )
}
```

```
)  
i <- x >= lb & x <= ub  
lines(x, hx)  
polygon(  
  c(lb,x[i],ub),  
  c(0,hx[i],0),  
  col="red"  
)  
area <- pnorm(ub, mean, sd) - pnorm(lb, mean, sd)  
result <- paste(  
  "P(",lb,"< IQ <",ub,") =",  
  signif(area, digits=3)  
)  
mtext(result,2)  
}
```

show()



For a comprehensive view of probability plotting in R, see Vincent Zonekynd's Probability Distributions.

Fitting Distributions

There are several methods of fitting distributions in R. Here are some options. You can use the `qqnorm()` function to create a Quantile-Quantile plot evaluating the fit of sample data to the normal distribution. More generally, the `qqplot()` function creates a Quantile-Quantile plot for any theoretical distribution.

Q-Q plots

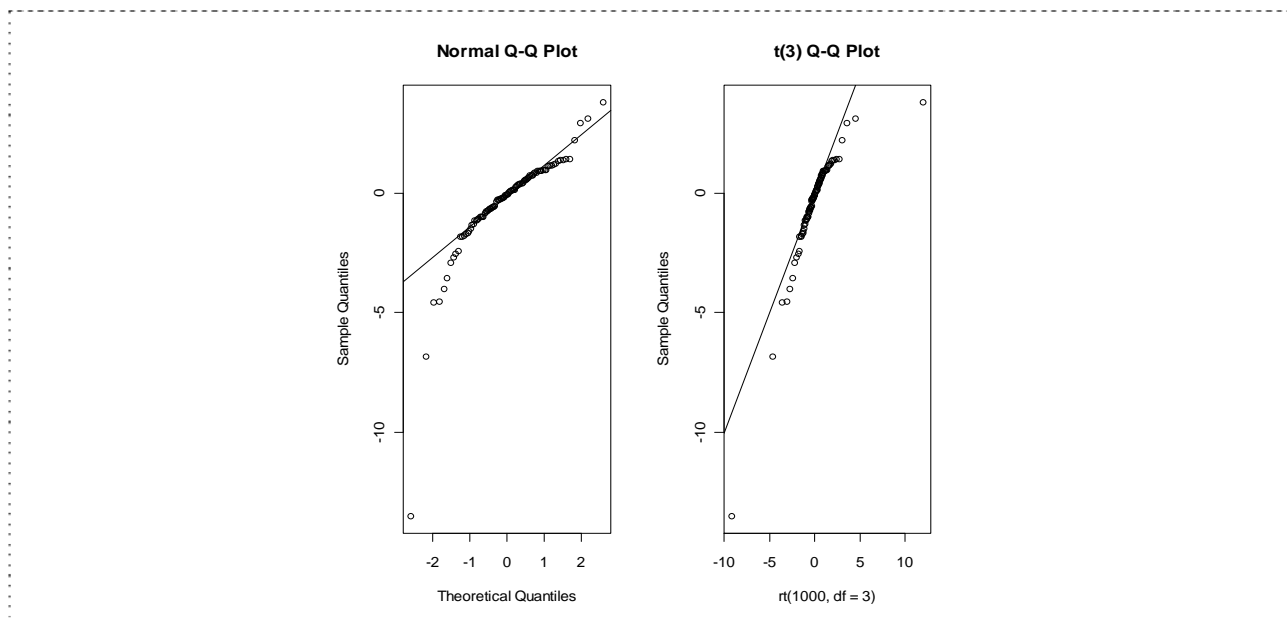
```
par(mfrow=c(1,2))

# create sample data
x <- rt(100, df=3)

# normal fit
qqnorm(x)
qqline(x)

show <- function()
{
  # t(3Df) fit
  qqplot(
    rt(1000,df=3),
    x,
    main="t(3) Q-Q Plot",
    ylab="Sample Quantiles"
  )
  abline(0,1)
}

show()
```



The `fitdistr()` function in the MASS package provides maximum-likelihood fitting of univariate distributions. The format is `fitdistr(x, densityfunction)` where `x` is the sample data and `densityfunction` is one of the following: "beta", "cauchy", "chi-squared", "exponential", "f", "gamma", "geometric", "log-normal", "lognormal", "logistic", "negative binomial", "normal", "Poisson", "t" or "weibull".

Estimate parameters assuming log-Normal distribution

```
# create some sample data
x <- rlnorm(100)

# estimate paramters
fitdistr(x, "lognormal")
```

Finally R has a wide range of goodness of fit tests for evaluating if it is reasonable to assume that a random sample comes from a specified theoretical distribution. These include chi-square, Kolmogorov-Smirnov, and Anderson-Darling.

For more details on fitting distributions, see Vito Ricci's *Fitting Distributions with R*. For general (non R) advice, see Bill Huber's *Fitting Distributions to Data*.

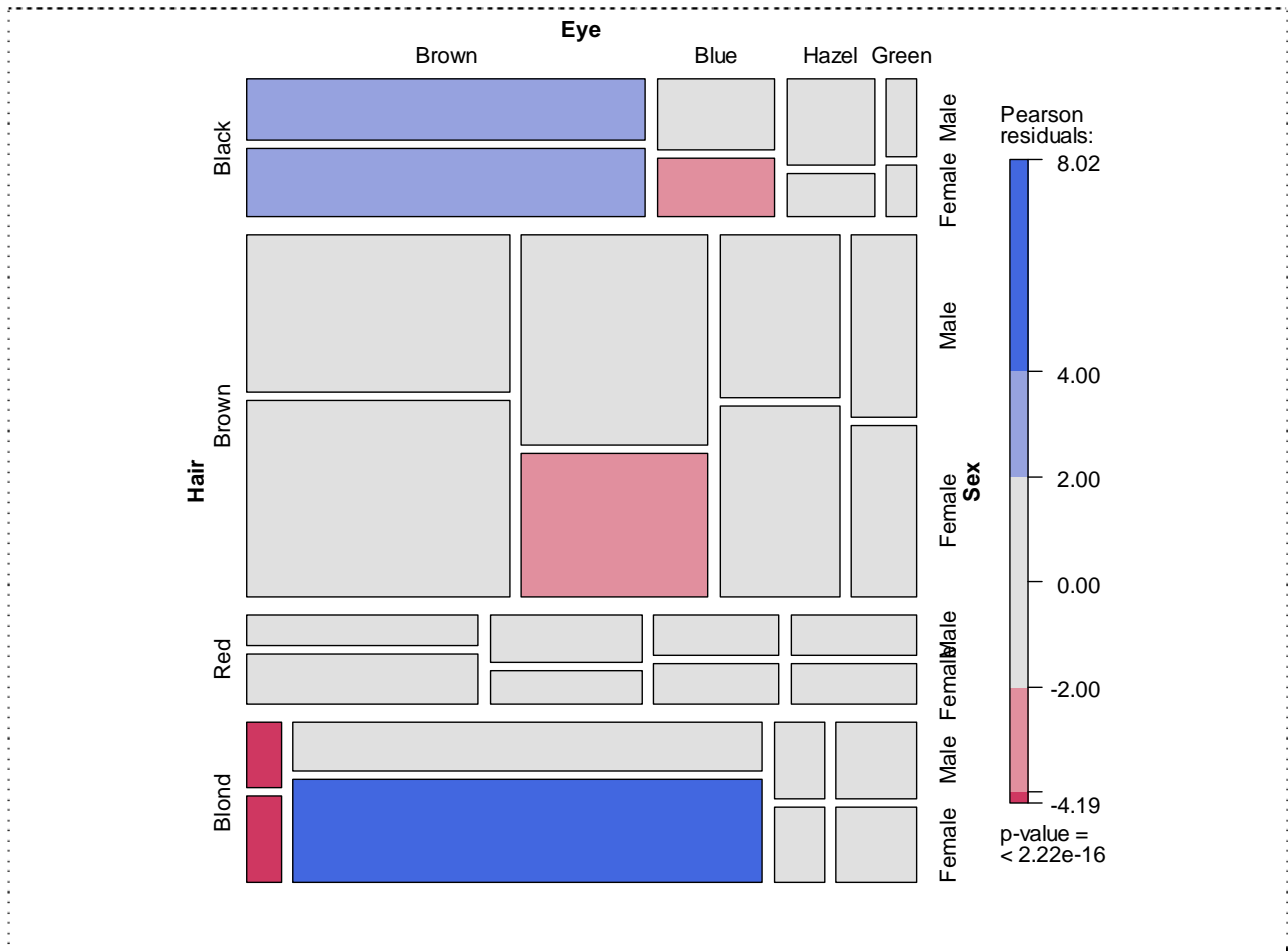
Visualizing Categorical Data

The `vcd` package provides a variety of methods for visualizing multivariate categorical data, inspired by Michael Friendly's wonderful "Visualizing Categorical Data". Extended mosaic and association plots are described here. Each provides a method of visualizing complex data and evaluating deviations from a specified independence model. For more details, see The *Strucplot Framework*.

Mosaic Plots

For extended mosaic plots, use `mosaic(x, condvar=, data=)` where `x` is a table or formula, `condvar=` is an optional conditioning variable, and `data=` specifies a dataframe or a table. Include `shade=TRUE` to color the figure, and `legend=TRUE` to display a legend for the Pearson residuals.

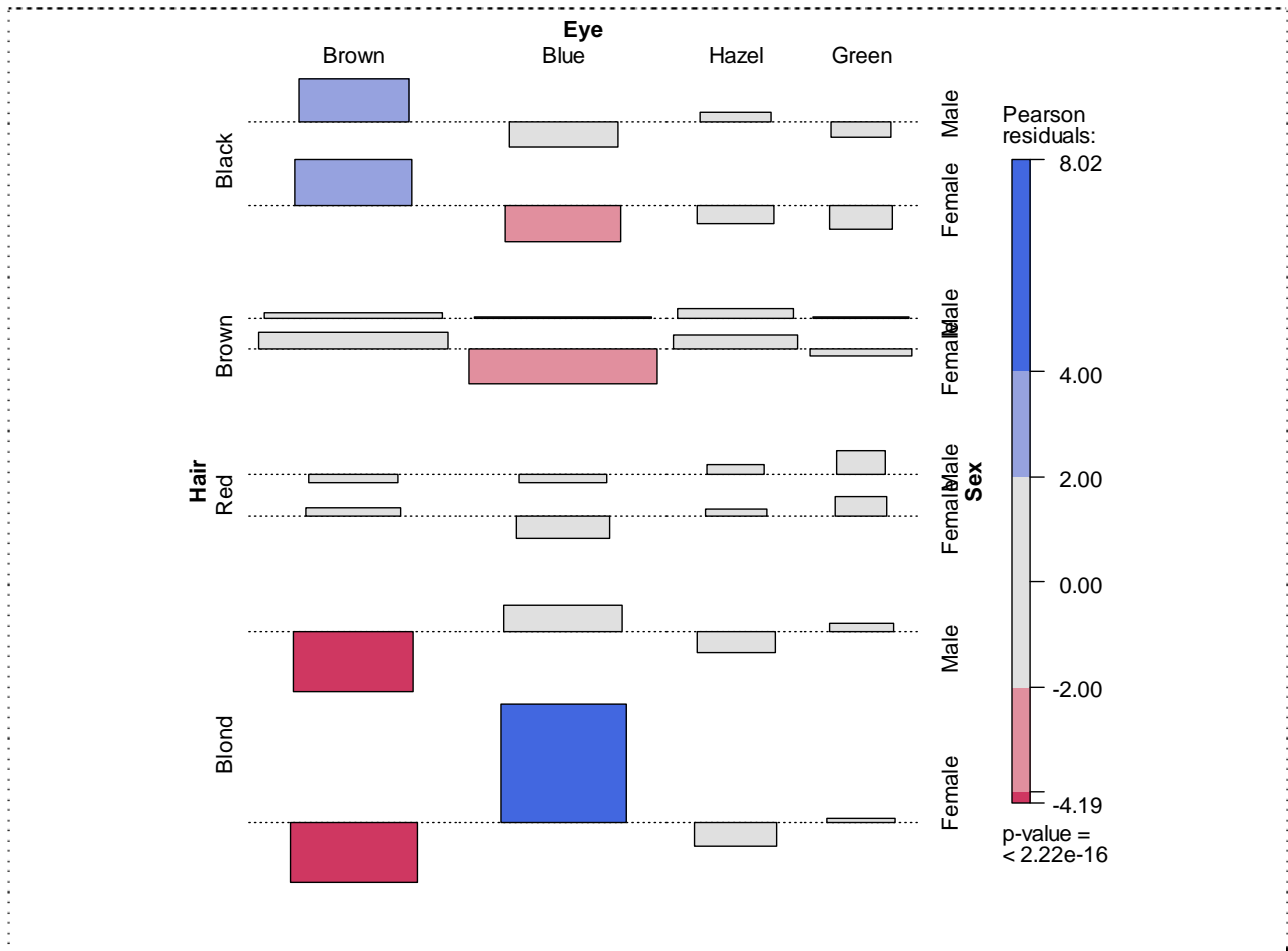
```
mosaic(HairEyeColor, shade=TRUE, legend=TRUE)
```



Association Plots

To produce an extended association plot use `assoc(x, row_vars, col_vars)` where `x` is a contingency table, `row_vars` is a vector of integers giving the indices of the variables to be used for the rows, and `col_vars` is a vector of integers giving the indices of the variables to be used for the columns of the association plot.

```
assoc(HairEyeColor, shade=TRUE)
```



Going Further

Both functions are complex and offer multiple input and output options. See `help(mosaic)` and `help(assoc)` for more details.