Using R for statistical analyses

Robert Bauer

Warnemünde, 04/24/2012

Leibniz Institute for
Baltic Sea Research
Warnemünde

Day 3 - Agenda:

- Last Exercises
- Random Samples and Probability Density Functions (PDF)
- Adding Lines, Text, Legend
- Saving Plots
- Kolmogorov-Smirnov tests (One and Two Samples)
- Shapiro-Wilk test
- Exercises

Exercises

CTD.xls

1. plot the salinity profile
2. plot the temperature and salinity profiles above each other
3. plot the profiles of temperature, salinity, conductivity and oxygen in a 2x2 plotting window

Parasite.xls

1. Import Parasite.csv
2. plot the age distribution of both sexes as 2 histograms but in one window
3. redraw the age distribution as colored lines using the density and plot function
4. add a legend
5. create boxplots of the weight of infected and non infected organisms and both sexes

Results - CTD

```
setwd("~/Dropbox/R_course/day2") # set working directory
CTD <- read.table("CTD.csv", header=T, sep=',', dec=".") #
    load dataframe
head(CTD)
colnames(CTD) <- c(colnames(CTD)[1:5], "Pressure", "
    Conductivity", "Temp",  "Sal",  "OxySat")
head(CTD)
attach(CTD)

Depth <- Pressure -1
```

Results - CTD

```
ylabel <- "depth [m]"
ylimits <- c(max(Depth),0)

par(mfrow=c(2,2))

# plot temperature profile
plot(Temp, Depth, ylim=ylimits, xlim=c(5,20),  type="l",
    ylab=ylabel, xlab="", axes=F)
axis(2, pos=5) # plot y-axis to the left
axis(3, pos=0) # plot x-axis above the plot
mtext("temperature [degrees C]", side=3, line=2, cex=0.8)
```

Results - CTD

```
# plot salinity profile
plot(Sal, Depth, ylim=ylimits, xlim=c(10,30), type="l", ylab
    =ylabel, xlab="", axes=F)
axis(2, pos=10) # plot y-axis to the left
axis(3, pos=0) # plot x-axis above the plot
mtext("salinity", side=3, line=2, cex=0.8)

# plot conductivity
plot(Conductivity, Depth, ylim=ylimits, xlim=c(15,30) , type
    ="l", ylab=ylabel, xlab="", axes=F)
axis(2, pos=15) # plot y-axis to the left
axis(3, pos=0) # plot x-axis above the plot
mtext("conductivity", side=3, line=2, cex=0.8)
```

Random Samples

```
n <- 10                                # sample size
x <- rnorm(n, mean=24, sd=4)           # normal distribution
```

Random Samples

```
n <- 10                                # sample size
x <- rnorm(n, mean=24, sd=4)           # normal distribution
```

```
Range <- seq(10,40,by=1)               # define groups of values
f <- dnorm(Range, mean=24, sd=4)       # prob. density function
sum(f)                                 # sum of f = 1
```

Random Samples

```
n <- 10                                  # sample size
x <- rnorm(n, mean=24, sd=4)             # normal distribution

Range <- seq(10,40,by=1)                 # define groups of values
f <- dnorm(Range, mean=24, sd=4)         # prob. density function
sum(f)                                   # sum of f = 1
```

```
# prepare data for plotting
f <- f*n  # multiply densities by number of oberservations

hist(x, breaks = Range, col="grey")
# add red line showing the probability density function
lines(10:40, f, col="red", lwd = 2)
```

Random Samples

```
Y <- seq(0,12,0.05)     # define range/groups of values

# genreate different probability density functions (PDF)
A <- dnorm(Y,mean=4,sd=1)
B <- dnorm(Y,mean=8,sd=1)
C <- dnorm(Y,mean=8,sd=0.5)
```

Random Samples

```r
plot(Y, A,                                 # plot first PDF
     type="l",                             # plotting line
     lwd = 2,                              # line width (default=1)
     ylim=c(0,1),                          # Range for the y-axis
     ylab = "relative frequency",          # label for the y-axis
     xlab = "Y",                           # label for the x-axis
     font.lab = 3)                         # font type
# font type: 1=plain text, 2=bold, 3=italic, 4=bold italic
```

Random Samples

```
plot(Y, A,                          # plot first PDF
     type="l",                      # plotting line
     lwd = 2,                       # line width (default=1)
     ylim=c(0,1),                   # Range for the y-axis
     ylab = "relative frequency",   # label for the y-axis
     xlab = "Y",                    # label for the x-axis
     font.lab = 3)                  # font type
# font type: 1=plain text, 2=bold, 3=italic, 4=bold italic
```

```
lines(Y, B, lwd = 2, lty = 2)       # add second PDF
lines(Y, C, lwd = 2, lty = 3)       # add third PDF
# Line type: 0=blank, 1=solid, 2=dashed, 3=dotted, 4=dotdash
    , 5=longdash, 6=twodash
```

Random Samples

```
# add PHD label
text(3,0.38, "A", font = 2)
text(10,0.2, "B", font = 2)
text(9,0.8,  "C", font = 2)
# font type: 1=plain text, 2=bold, 3=italic, 4=bold italic
```

```
# add legend
legend("topleft",
       legend = c("A", "B", "C"),# Legend labels
       text.width = 1,           # legend width (default=1)
       lwd = 2,                  # line width (default=1)
       bty="n",) # box type: "n" no box; "o" show box
       lty=c(1,2,3))             # Line types
# Line types: 0=blank, 1=solid, 2=dashed, 3=dotted, 4=
    dotdash, 5=longdash, 6=twodash
```

Random Samples

```
# add PHD information
fontsize = 0.9
text(4, 0.5,                       # text coordinates (x,y)
     "mean = 4 ; sigma = 1",  # text to be written
     col="red",                    # text color
     cex=fontsize,                 # text size (default=1)
     font= 2)                      # font type
# font types: 1=plain text, 2=bold, 3=italic, 4=bold italic
```

```
text(11, 0.35, "mean = 8 ; sigma = 1", col="red", cex=
    fontsize, font= 2)
text(10.5, 0.65, "mean = 8 ; sigma = 0.5", col="red", cex=
    fontsize, font= 2)
```

Saving Plots - Pixel Graphics

```
setwd("path/where/to/save/the plots")

# save files as pixel graphic
bmp('plot.bmp',
    res = 300,      # plot resolution in dpi
    width = 2220,   # plot width according to defined units
    height = 1220, # plot height according to defined units
    units = "px")  # units of width and height;
# default units="px" (pixels);
# other options: "in" (inches), "cm" or "mm"

plot(1:10, 51:60)  # dummy plot
dev.off()          # closes graphic device
```

```
# jpeg('plot.jpeg', ..)
# png('plot.png', ..)
# tiff('plot.tiff', ..)
```

Saving Plots - Vector Graphics

```
setEPS()  #  set defaults appropriate for publication
postscript("plot.eps",
           width = 7,  # plot width in inches (default=7)
           height = 7) # plot height in inches (default=7)

plot(1:10, 51:60)      # dummy plot
dev.off()              # closes graphic device
```

Kolmogorov-Smirnov test - Normal Distribution

```
n <- 1000       # sample size
x <- rnorm(n)   # normally distributed numbers

# Kolmogorov-Smirnov test for normal distribution
ks.test(x, "pnorm", mean = mean(x), sd = sqrt(var(x)))
```

```
One-sample Kolmogorov-Smirnov test
data:  x
D = 0.0186, p-value = 0.8808
alternative hypothesis: two-sided
```

Shapiro-Wilk test

```r
n <- 1000       # sample size
x <- rnorm(n)   # normally distributed numbers
```

```r
shapiro.test(x)     # better for small sampling size (<50)
```

```
Shapiro-Wilk normality test
data:  x
W = 0.9977, p-value = 0.1795
```

Random Numbers

Function	Description
`rnorm(n, mean = 0, sd = 1)`	Normal (Gaussian)
`rlnorm(n, meanlog = 0, sdlog = 1)`	Lognormal
`runif(n, min=0, max=1)`	Uniform
`rbinom(n, size, prob)`	Binominal
`rnbinom(n, size, prob)`	Negative Binominal
`rpois(n, lambda)`	Poisson
`rgamma(n, shape, scale=1)`	Gamma
`rexp(n, rate=1)`	Exponential

Kolmogorov-Smirnov test - Uniform Distribution

```r
n <- 1000       # sample size
y <- runif(n, min = -3, max = 3) # uniformly distributed
    numbers

# Kolmogorov-Smirnov test for uniform distribution
ks.test(y, "punif", min = -3, max = 3)
```

```
One-sample Kolmogorov-Smirnov test
data:  y
D = 0.0159, p-value = 0.9613
alternative hypothesis: two-sided
```

Kolmogorov-Smirnov test - 2 Samples 1 Distribution?

```
# two-samples of the same distribution?
par(mfrow=c(2,1))
hist(x, main="normally distributed values", xlim=c(-3,3))
hist(y, main="uniformly distributed values", xlim=c(-3,3))

ks.test(x, y)
```

```
Two-sample Kolmogorov-Smirnov test
data:  x and y
D = 0.196, p-value < 2.2e-16
alternative hypothesis: two-sided
```

Exercises

1. generate random Poisson-distributed numbers
   `lambda <- c(0.1, 1, 2, 3, 10)`
2. plot associated probability density functions (PDF)
   (one plot with multiple lines of different type)
   ```
   dpois(range, lambda)
   ```
3. add a legend
4. perform Kolmogorov-Smirnov tests for Normal Distribution