STATS 406 F15: Lab 03

1 Plotting: making your plots prefessional

1.1 Recap: basic plot commands

- **plot()**. Here we only recap list plots. In practice you rarely work with function plots in R.

```
# Recap: plot()
a_vec = seq(from=0, to=3, by=0.1);
b_vec = a_vec + rnorm(length(a_vec));
plot(b_vec~a_vec);
```

  Looks awkward. Today's lab is dedicated to improve it.

- **hist()**.

```
# Recap: hist()
# continued from the previous example...
hist(b_vec);
```

- **boxplot()**.

```
# Recap: boxplot();
# 3 groups of data, each group contains 20 random numbers
groupsize = 20; ngroup = 3;
a_vec = rnorm(ngroup*groupsize); a_vec = a_vec + seq(from=0, to=5, length.out=length(a_vec));
b_vec = as.factor( rep( letters [1:ngroup], rep(groupsize, ngroup)) );
# side-by-side group-wise boxplots
test_data_frame = data.frame(a_vec, b_vec);
boxplot( with(test_data_frame, a_vec ~ b_vec) );
```

1.2 Manage screen layout

- Method 1: **par(mfrow=c(n1, n2))**(by row) or **par(mfcol=c(n1, n2))**(by column).

```
# par(mfrow=...)
# Open a new screen.
x11(); # Not applicable under commandline
# Split screen
par(mfrow = c(2, 1));
# Plots
set.seed(2015);
plot(rnorm(1000));
qqnorm(runif(1000));
#
# When we're done:
dev.off(); # or click to close the window
```

  Drawback: always evenly split the screen, cannot control screen block sizes.

- Method 2: **layout(splitmatrix)**. More flexible.
  Example: If we want a big square plot at the top-left corner and split the rest accordingly, first set the split matrix that looks like:

$$\text{split matrix} = \begin{pmatrix} 1 & 1 & 1 & 2 & 2 \\ 1 & 1 & 1 & 2 & 2 \\ 1 & 1 & 1 & 2 & 2 \\ 3 & 3 & 3 & 4 & 4 \\ 3 & 3 & 3 & 4 & 4 \end{pmatrix}$$

```
# layout(splitmatrix)
x11();
# Prepare the split matrix
SplitMat = array(0, c(5, 5));
SplitMat[1:3, 1:3] = 1;
SplitMat[4:5, 1:3] = 2; SplitMat[1:3, 4:5] = 3;
SplitMat[4:5, 4:5] = 4;
```

  and then split the screen accordingly and plot:

```
# Split the screen
layout(SplitMat, 2, 2);
layout.show(4); # just to confirm, not required
# Plots
set.seed(2015);
plot(rnorm(1000));
qqnorm(runif(1000));
qqnorm(rt(1000, df=3));
hist(rnorm(1000));
#
# When we're done:
dev.off(); # or click to close the window
```

- Method 3: **split.screen()**. Will not elaborate. Check documentation if interested.

1.3 More plotting commands: contour plots

Apart from line plots, another important family of plots in R are contour plots.

- **contour()**. Shows only contour lines.

```
# A naive contour plot
# Prepare an integer multiply chart within 100
x_mat = (1:10) %*% t(rep(1, 10));
y_mat = t(x_mat);
z_mat = x_mat*y_mat;
# Contour plot
contour(z_mat);
```

- **levelplot(). Requires package "lattice"**. If we want to use colors to indicate entry values...

```
# Continuing the last example
require(lattice);
print(levelplot(z_mat)); # Always use print with levelplot.
```

- **image()**. Similar to **levelplot()**.

```
# Continuing the last example
image(z_mat);
```

1.4 Adjusting plot parameters

Plots almost never appear "raw"(or "naked") in formal write-ups. Now we learn how to tune the appearance of our plots. To embed our discussion in a specific context, let's recall the light bulb example we went through last week:

> Example: a simple on-off Markov chain
> Each light bulb has two status: 1=lighted up or 0=burnt out. In each round, a light bulb goes from status 1 to 0 with probability $q = 0.05$ or from status 0 to 1 with probability $p = 0.10$.

This time we focus the update track of a light bulb. First, prepare data for plotting:

```
# Illustrate the update track of one light bulb
# Prepare data
set.seed(2015);
p = 0.10; q = 0.05;
nround = 2000;
UpdateTrack = integer(nround);
UpdateTrack[1] = 1;
for(i in 2:nround){
  if(UpdateTrack[i-1]==0){
    UpdateTrack[i] = rbinom(1, 1, p);
  } else{
    UpdateTrack[i] = rbinom(1, 1, 1-q);
  }
}
```

Then we plot it:

```
# Plot
plot(UpdateTrack,
  type='b', lty=3, pch=25, col='blue',
  xlim=c(-1, 52), ylim=c(-0.2, 1.2),
  main=paste('Light bulb status update within ', nround, ' rounds', sep=''), sub='An on-off process illustration', xlab
   ='Index', ylab='Status',
  cex.lab=1.5, cex.main=1.25, cex.sub=1.25, # col.lab='darkblue',
  axes=FALSE, # to be specified later
)
axis(1, at=c(1, 25, 50), cex.axis=1.5);
axis(2, at=c(0, 1), cex.axis=1.5);
box();
```

This example contains a few frequently used plotting parameters:

- Line/Point styles: **type**, **lty**, **pch**, **col**, ...
  - **type**: consult **?plot**, it refers to the type of the line. 'b' for showing both the points and the line, 'o' for both but over-positioning each other, 'l' for line-only and 'p' for points-only.
  - **lty**: line type, take integer values.
  - **pch**: the point marker used, take integer or character values.
  - **col**: color, take integer or character values.
- Plot range: **xlim**, **ylim**. Here we expanded the plot ranges to have more comfortable margins.
- Texts: **main**, **sub**, **xlab**, **ylab**, ...
  - **main**: main title of the plot.
  - **sub**: subtitle.
  - **xlim**, **ylim**: ranges of axes.
- Font sizes: **cex**(default), **cex.main**, **cex.sub**, **cex.lab**, **cex.axis**, ...

Quiz: How to assign different colors to different points?

1.5 Low-level plots

We can add additional plotting objects like points or lines, as well as texts to an existing plot. Such additions are called low-level plots.

- Adding points: **points(x, y)**.

```
# Continuing the last example
Points_x = c(13,15,17,30,31,39);
Points_y = c(0.3, 0.4, 0.5, 0.6, 0.7, 0.8);
points(x, y);
```

- Adding lines: **lines(x, y)**, similar to **points**, omitted here. Check documentation.
- Adding a reference line: **abline(a=intercept, b=slope)**.

```
# Continuing the last example
abline(0.2, 0.5/nround, type=3, col='lightblue');
```

- Adding a legend: **legend()**
  - Usage: **legend(location, legendnames, optimalstyle)**.
  - The style options should be consistent with curves.

```
# Continuing the last example
legend('topright', c('LightBulb_1'), lty=3, pch=25, col=c('blue'));
```

1.6 Exercises

See Lab_3.R.

2 Law of large numbers and central limit theorem

2.1 Illustrations of strong- and weak- laws of large numbers

```
### Illustrations of SLLNo's and WLLNo's
nsamples = 1000; samplesize = 1000;
BigRVMatrix = matrix(runif(nsamples*samplesize), c(nsamples, samplesize));
```

2.2 Central limit theorem

In a random sample $S = \{X_1, X_2, \ldots, X_n\}$, ...

- **Whose distribution is asymptotically normal?**
- **Does the sample distribution go to normal?**
- **Compare CLT under different parent population distributions**

```
# See Lab_3.R
```