compgen2021: Week 1 exercises

Irem YUCEL

Exercises for Week1

Statistics for genomics

How to summarize collection of data points: The idea behind statistical distributions

1. Calculate the means and variances of the rows of the following simulated data set, and plot the distributions of means and variances using `hist()` and `boxplot()` functions. [Difficulty: **Beginner/Intermediate**]

```
set.seed(100)

# sample data matrix from normal distribution
gset = rnorm(600, mean = 200, sd = 70)
data = matrix(gset, ncol = 6)

row_means = rowMeans(data)
row_vars = apply(data, 1, var)

par(mfrow = c(2, 2))
hist(row_means)
boxplot(row_means)
hist(row_vars)
boxplot(row_vars)
```

2. Using the data generated above, calculate the standard deviation of the distribution of the means using the `sd()` function. Compare that to the expected standard error obtained from the central limit theorem keeping in mind the population parameters were $\sigma = 70$ and $n = 6$. How does the estimate from the random samples change if we simulate more data with `data=matrix(rnorm(6000,mean=200,sd=70),ncol=6)`? [Difficulty: **Beginner/Intermediate**]

```
gset = rnorm(600, mean = 200, sd = 70)
data = matrix(gset, ncol = 6)
row_means = rowMeans(data)

means_sd = sd(row_means)
ctl_sd = 70/sqrt(6)

# The standard deviation of the sample means is similar to the standard
# deviation from CLT.

data2 = matrix(rnorm(6000, mean = 200, sd = 70), ncol = 6)
```

```
row_means2 = rowMeans(data2)
means_sd2 = sd(row_means2)

# Changing to 6000 did not make a difference, the estimate from the random
# sampling is still very similar to that of from CLT.
```

3. Simulate 30 random variables using the `rpois()` function. Do this 1000 times and calculate the mean of each sample. Plot the sampling distributions of the means using a histogram. Get the 2.5th and 97.5th percentiles of the distribution. [Difficulty: **Beginner/Intermediate**]

4. Use the `t.test()` function to calculate confidence intervals of the mean on the first random sample `pois1` simulated from the `rpois()` function below. [Difficulty: **Intermediate**]

```
# HINT
set.seed(100)

# sample 30 values from poisson dist with lambda parameter = 30 (5?)
pois1 = do(1000) * mean(rpois(30, lambda = 5))
hist(pois1[, 1], col = "cornflowerblue", border = "white", xlab = "Sample Means")
q = quantile(pois1[, 1], p = c(0.025, 0.975))
abline(v = c(q[1], q[2]), col = "red")
text(x = q[1], y = 200, round(q[1], 2))
text(x = q[2], y = 200, round(q[2], 2))

t.test(pois1)
```

5. Use the bootstrap confidence interval for the mean on `pois1`. [Difficulty: **Intermediate/Advanced**]

```
# Bootstrapping is when we don't know the population parameter (Lambda in this
# example).So we take samples from the observations (pois1) until we have
# sample size of the original sample (n=30) and compute the 95% CI for these
# new samples.

set.seed(100)
pois1 = do(1000) * mean(rpois(30, lambda = 5))
boot_samples = do(1000) * sample(pois1, 30, replace = T)
quantile(boot_samples[, 1], p = c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 4.166667 5.866667
```

How to test for differences in samples

1. Test the difference of means of the following simulated genes using the randomization, `t-test()`, and `wilcox.test()` functions. Plot the distributions using histograms and boxplots. [Difficulty: **Intermediate/Advanced**]

```
set.seed(101)
gene1 = rnorm(30, mean = 4, sd = 3)
gene2 = rnorm(30, mean = 3, sd = 3)
```

```r
t.test(gene1, gene2)
wilcox.test(gene1, gene2)

# Randomization Save the real difference between the gene1 and geen2 groups
# Generate a new data frame using gene1 and gene2 values (in esp column) with
# 30 test and 30 control labels (group column).

org.diff = mean(gene1) - mean(gene2)
gene.df = data.frame(exp = c(gene1, gene2), group = c(rep("test", 30), rep("control",
    30)))

# Shuffle by the group labels and get the means for control and test, get the
# difference, repeat 1000 times
shuffled_means = do(1000) * diff(mosaic::mean(exp ~ shuffle(group), data = gene.df))

hist(shuffled_means[, 1], xlab = "null distribution | no difference in samples",
    main = expression(paste(H[0], " :no difference in means")), xlim = c(-2, 2),
    col = "cornflowerblue", border = "white")
abline(v = quantile(exp.null[, 1], 0.95), col = "red")
abline(v = org.diff, col = "blue")
text(x = quantile(exp.null[, 1], 0.95), y = 200, "0.05", adj = c(1, 0), col = "red")
text(x = org.diff, y = 200, "org. diff.", adj = c(1, 0), col = "blue")
```

2. Test the difference of the means of the following simulated genes using the randomization, `t-test()` and `wilcox.test()` functions. Plot the distributions using histograms and boxplots. [Difficulty: **Intermediate/Advanced**]

```r
set.seed(100)
gene1 = rnorm(30, mean = 4, sd = 2)
gene2 = rnorm(30, mean = 2, sd = 2)

## Same as above

t.test(gene1, gene2)
wilcox.test(gene1, gene2)

# Randomization Save the real difference between the gene1 and geen2 groups
# Generate a new data frame using gene1 and gene2 values (in esp column) with
# 30 test and 30 control labels (group column).

org.diff = mean(gene1) - mean(gene2)
gene.df = data.frame(exp = c(gene1, gene2), group = c(rep("test", 30), rep("control",
    30)))

# Shuffle by the group labels and get the means for control and test, get the
# difference, repeat 1000 times
shuffled_means = do(1000) * diff(mosaic::mean(exp ~ shuffle(group), data = gene.df))

hist(shuffled_means[, 1], xlab = "null distribution | no difference in samples",
    main = expression(paste(H[0], " :no difference in means")), xlim = c(-2, 2),
    col = "cornflowerblue", border = "white")
abline(v = quantile(exp.null[, 1], 0.95), col = "red")
abline(v = org.diff, col = "blue")
```

```
text(x = quantile(exp.null[, 1], 0.95), y = 200, "0.05", adj = c(1, 0), col = "red")
text(x = org.diff, y = 200, "org. diff.", adj = c(1, 0), col = "blue")
```

3. We need an extra data set for this exercise. Read the gene expression data set as follows: `gexpFile=system.file("extdata","geneExpMat.rds",package="compGenomRData")` `data=readRDS(gexpFile)`. The data has 100 differentially expressed genes. The first 3 columns are the test samples, and the last 3 are the control samples. Do a t-test for each gene (each row is a gene), and record the p-values. Then, do a moderated t-test, as shown in section "Moderated t-tests" in this chapter, and record the p-values. Make a p-value histogram and compare two approaches in terms of the number of significant tests with the 0.05 threshold. On the p-values use FDR (BH), Bonferroni and q-value adjustment methods. Calculate how many adjusted p-values are below 0.05 for each approach. [Difficulty: **Intermediate/Advanced**]

```
# Read the data
gexpFile = system.file("extdata", "geneExpMat.rds", package = "compGenomRData")
data = readRDS(gexpFile)

# For the whole data Check if the variances are similar for the t.test:
col_vars = apply(data, 2, var)

# Use Welch Two Sample t-test since variances are different:
test = data[, 1:3]
control = data[, 4:6]
n1 = 3
n2 = 3
stats::t.test(test, control)
```

```
##
##  Welch Two Sample t-test
##
## data:  test and control
## t = -14.74, df = 4115, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.2222141 -0.1700416
## sample estimates:
##     mean of x     mean of y
## -0.0004465826  0.1956812552
```

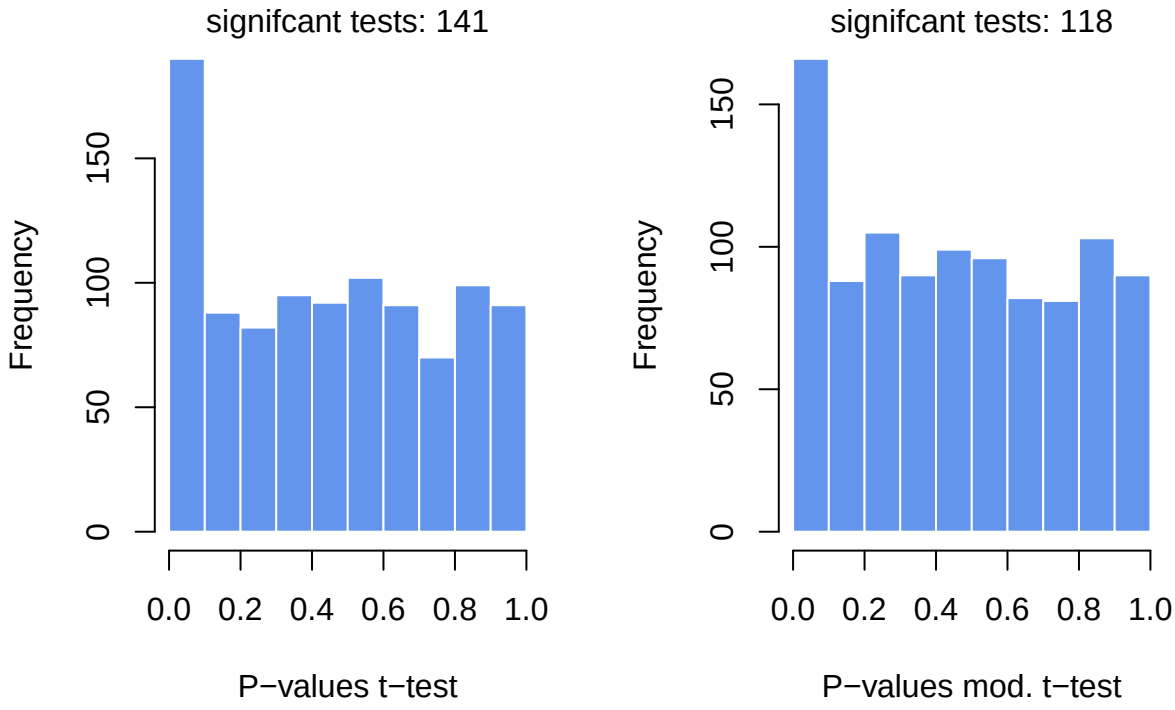
```
# Testing each gene: The formula for the Welch's t-test is: Differences of
# means/ variance Calculate the differences of the means of 2 groups:
diff_means = rowMeans(test) - rowMeans(control)

# Get the estimate of pooled variance
pooled_var = sqrt((rowVars(test) * (n1 - 1) + rowVars(control) * (n2 - 1))/(n1 +
    n2 - 2) * (1/n1 + 1/n2))

# Estimate t statistic without moderated variance and calculate P-value of
# rejecting null
t = diff_means/pooled_var
p = 2 * pt(-abs(t), n1 + n2 - 2)
```

```r
# Same with the moderated variance
mod.pooled_var = (pooled_var + median(pooled_var))/2  # moderation in variation
t.mod <- diff_means/mod.pooled_var
p.mod = 2 * pt(-abs(t.mod), n1 + n2 - 2)

par(mfrow = c(1, 2))
hist(p, col = "cornflowerblue", border = "white", main = "", xlab = "P-values t-test")
mtext(paste("signifcant tests:", sum(p < 0.05)))
hist(p.mod, col = "cornflowerblue", border = "white", main = "", xlab = "P-values mod. t-test")
mtext(paste("signifcant tests:", sum(p.mod < 0.05)))
```




```r
# Moderated t-test reduced the number of significant tests to 118 from 141


# Pvalue correction for the t-test and the moderated t-test:
qvalues <- qvalue(p)
qvaluesmod <- qvalue(p.mod)
bonf.pval = p.adjust(p, method = "bonferroni")
bonf.pvalmod = p.adjust(p.mod, method = "bonferroni")
fdr.adj.pval = p.adjust(p, method = "fdr")
fdr.adj.pvalmod = p.adjust(p.mod, method = "fdr")

# For t-test

num_sig_q <- sum(qvalues$qvalues < 0.05)
num_sig_bonf <- sum(bonf.pval < 0.05)
num_sig_fdr <- sum(fdr.adj.pval < 0.05)

paste("Qvals:", num_sig_q)
```

```
## [1] "Qvals: 100"
```

```r
paste("Bonferroni:", num_sig_bonf)
```

```
## [1] "Bonferroni: 25"
```

```r
paste("FDR:", num_sig_fdr)
```

```
## [1] "FDR: 99"
```

```r
# For the moderated t-test

num_sig_q <- sum(qvaluesmod$qvalues < 0.05)
num_sig_bonf <- sum(bonf.pvalmod < 0.05)
num_sig_fdr <- sum(fdr.adj.pvalmod < 0.05)

paste("Qvals:", num_sig_q)
```

```
## [1] "Qvals: 100"
```

```r
paste("Bonferroni:", num_sig_bonf)
```

```
## [1] "Bonferroni: 2"
```

```r
paste("FDR:", num_sig_fdr)
```

```
## [1] "FDR: 100"
```

Relationship between variables: Linear models and correlation

Below we are going to simulate X and Y values that are needed for the rest of the exercise.

```r
# set random number seed, so that the random numbers from the text is the same
# when you run the code.
set.seed(32)

# get 50 X values between 1 and 100
x = runif(50, 1, 100)

# set b0,b1 and variance (sigma)
b0 = 10
b1 = 2
sigma = 20
# simulate error terms from normal distribution
eps = rnorm(50, 0, sigma)
# get y values from the linear equation and addition of error terms
y = b0 + b1 * x + eps



# 1. Run the code then fit a line to predict Y based on X.
```

```r
# [Difficulty:**Intermediate**]

model = lm(y ~ x)
summary(model)



# 2. Plot the scatter plot and the fitted line. [Difficulty:**Intermediate**]

plot(x, y)
abline(model, col = "red")


# 3. Calculate correlation and R^2. [Difficulty:**Intermediate**]

corr = cor(x, y)
r_sqr = corr^2


# 4. Run the `summary()` function and try to extract P-values for the model
# from the object returned by `summary`. See `?summary.lm`.
# [Difficulty:**Intermediate/Advanced**]

model_summary = summary(model)
model_summary


# 5. Plot the residuals vs. the fitted values plot, by calling the `plot()`
# function with `which=1` as the second argument. First argument is the model
# returned by `lm()`. [Difficulty:**Advanced**]

plot(lm(y ~ x), which = 1)


# 6. For the next exercises, read the data set histone modification data set.
# Use the following to get the path to the file:

hmodFile = system.file("extdata", "HistoneModeVSgeneExp.rds", package = "compGenomRData")

# There are 3 columns in the dataset. These are measured levels of H3K4me3,
# H3K27me3 and gene expression per gene. Once you read in the data, plot the
# scatter plot for H3K4me3 vs. expression. [Difficulty:**Beginner**]

histone_data = readRDS(hmodFile)
plot(histone_data$H3k4me3, histone_data$measured_log2, xlab = "Measured levels of H3K4me3",
    ylab = " Gene Expression")



# 7. Plot the scatter plot for H3K27me3 vs. expression.
# [Difficulty:**Beginner**]

plot(histone_data$H3k27me3, histone_data$measured_log2, xlab = "Measured levels of H3k27me3",
```

```
    ylab = " Gene Expression")



# 8. Fit the model for prediction of expression data using: 1) Only H3K4me3 as
# explanatory variable, 2) Only H3K27me3 as explanatory variable, and 3) Using
# both H3K4me3 and H3K27me3 as explanatory variables. Inspect the `summary()`
# function output in each case, which terms are significant.
# [Difficulty:**Beginner/Intermediate**]

fit1 = lm(histone_data$measured_log2 ~ histone_data$H3k4me3)
fit2 = lm(histone_data$measured_log2 ~ histone_data$H3k27me3)
fit3 = lm(histone_data$measured_log2 ~ histone_data$H3k4me3 + histone_data$H3k27me3)

summary(fit1)
summary(fit2)
summary(fit3)

# All terms are significant



# 10. Is using H3K4me3 and H3K27me3 better than the model with only H3K4me3?
# [Difficulty:**Intermediate**]

# Comparing R-squared values, which is: R-squared = Explained variation / Total
# variation. So higher R-squared value is better, as it means that the model
# can explain most of the variance in the data. Using both H3K4me3 and H3K27me3
# together slightly increased the R-squared ( from .6511 to .6723). Anova could
# be used here as we have 2 categorical variables to compare the models.
```