Appendix

Robin Boudry, Maarten Rahier, Tom Schipper, Laurens Van Paemel

Contents

Preparation

```
# Used libraries
library(plyr)
```

Importing and cleaning the data

```
# Read in data
data <- read.table("bptrial.txt", header=TRUE, sep=",", dec=".")
original <- data
# Set column names
colnames(data) <- c("treatment", "sex", "weight", "age",
                    "comp", "dbpdif", "dbp3", "dbp2", "dbp1")
# Add column dbp_end for the 5th measurement at the end of the experiment
data$dbp_end <- data$dbp3 + data$dbpdif
# Mean dbp during the run-in period
data$dbp_mean <- round((data$dbp1 + data$dbp2 + data$dbp3) / 3, 1)
# Raw data subset
data_raw <- data
# Treatment group as factor
data$treatment <- as.factor(mapvalues(
  data$treatment,
  from=c(1, 2, 3),
  to=c("Treatment 1", "Placebo", "Treatment 3")
))
# Gender as factor
data$sex <- as.factor(mapvalues(data$sex, c(0, 1), c("Female", "Male")))
# 80% adherence rule
data <- data[data$comp >= 0.8,]
# outlier removal
data=data[data$dbp3 != 66,]
# Subset of treatment 1 and the placebo
data12 <- data[data$treatment != "Treatment 3",]
levels(data12$treatment) <- c("Treatment 1", "Placebo", NA)
# Seperate subsets of treatment 1 and placebo
data1 <- data[data$treatment == "Treatment 1",]
data2 <- data[data$treatment == "Placebo",]
```

1. Power comparison

In the code, trailing 1's are a reference to "Treatment 1" and 2's to "Placebo".

Exploration and preparation

```
# Amount of observations
n <- nrow(data12)
n1 <- summary(data12$treatment)[["Treatment 1"]]
n2 <- summary(data12$treatment)[["Placebo"]]
```

Simulating the dbpdif data

```
# Checking normality
qqnorm(data1$dbpdif); qqline(data1$dbpdif)

qqnorm(data2$dbpdif); qqline(data2$dbpdif)

# Testing the homogeneity of variances
var.test(data1$dbpdif, data2$dbpdif)
## Although just not significant, the ratio of the variances
## seems to deviate from 1. It would be safer to assume that
## the variances are not equal.

# Observed means
mu_dif1 <- mean(data1$dbpdif)
mu_dif2 <- mean(data2$dbpdif)
# Observed SD
sd_dif1 <- sd(data1$dbpdif)
sd_dif2 <- sd(data2$dbpdif)
# Resample dbpdif
rdif1 <- rnorm(n1, mu_dif1, sd_dif1)
rdif2 <- rnorm(n2, mu_dif2, sd_dif2)
# Checking for normality
qqnorm(rdif1)
qqline(rdif1)

qqnorm(rdif2)
qqline(rdif2)

## QQplots from data generated randomly from normal distributions
## show similar deviations as the QQplots of the data
```

Simulating the dbp_end data

```
# Checking normality
qqnorm(data1$dbp_end); qqline(data1$dbp_end)

qqnorm(data2$dbp_end); qqline(data2$dbp_end)
```

```r
#The dbp_end seems to show fewer deviations from normality than dbpdif

# Observed means
mu_end1 <- mean(data1$dbp_end)
mu_end2 <- mean(data2$dbp_end)
# SD
sd_end1 <- sd(data1$dbp_end)
sd_end2 <- sd(data2$dbp_end)

# Resample dbp_end
rend1 <- rnorm(n1, mu_end1, sd_end1)
rend2 <- rnorm(n2, mu_end2, sd_end2)
# Check normality
qqnorm(rend1); qqline(rend1)

qqnorm(rend2); qqline(rend2)

## QQplots from data generated randomly from normal distributions
## show similar deviations as the QQplots of the data
```

Simulation with relative differences

```r
## 1000 simulations for each of 41 differences in means, for both
## dbpdif and dbp_end. These are plotted together for comparison

# Amount of simulations
n.sim <- 1000
# Collect results
pwr_end <- vector(,41)
pwr_dif <- vector(, 41)
# Set seed for reproducibility
set.seed(2018)
# Test 41 differences in means
for(y in 0:40){
  x <- 0.025 * y
  # Array of p-values
  pdif <- vector(, n.sim)
  pend <- vector(, n.sim)
  # Permutate
  for(i in 1:n.sim) {
    # dbp_dif
    mu_dif1.2 <- mu_dif1 + (abs(mu_dif1) - abs(mu_dif2)) * x
    rdif1 <- rnorm(n1, mu_dif1.2, sd_dif1)
    rdif2 <- rnorm(n2, mu_dif2, sd_dif2)
    tdif <- t.test(rdif1, rdif2, "less", mu=0, paired=F, var.equal=F)
    pdif[i] <- tdif$p.value
    # dbp_end
    mu_end1.2 <- mu_end1 + (mu_end2 - mu_end1) * x
    rend1 <- rnorm(n1, mu_end1.2, sd_end1)
    rend2 <- rnorm(n2, mu_end2, sd_end2)
    tend <- t.test(rend1, rend2, "less", mu=0, paired=F, var.equal=F)
```

```
    pend[i] <- tend$p.value
  }
  pwr_dif[y+1] <- mean(pdif < 0.05)
  pwr_end[y+1] <-mean(pend < 0.05)
}

# Plotting both power simulations in the same graph
plot(pwr_dif, type="n", xaxt="n", main="Comparative power", yaxt="n",
     ylab="Power (%)", xlab="Difference in means (% observed difference)")
lines(rev(pwr_dif), lty=1, col="blue")
lines(rev(pwr_end), pch=1, lty=2, col="red")
axis(side=1, at=seq(from=0, to=40, by=4), labels=seq(from=0, to=100, by=10))
axis(side=2, at=seq(from=0, to=1, by=0.2), labels=seq(from=0, to=100, by=20))
legend(33.77, 1.036, legend=c("dbp_dif", "dbp_end"),
       col=c("blue", "red"), lty=1:2, cex=1.2)
```

dbp_end has more power when adjusting means by a relative difference

Simulation with absolute differences

```
# Power simulations
power_f_delta <- function(N.sim, variable1, variable2){
  pdif <- vector()
  power <- vector()
  meandif <- vector()
  mean1 <- vector()
  mean2 <- vector()
  for(y in 1:60){
    ## By observation, the means differ no more than |13|. Therefore the
    ## means are first equalized and then increased step by step
    x <- c(y-60)*0.2
    for(i in 1:N.sim){
      # Treatment
      X1 <- rnorm(n=length(variable1),
                  mean=mean(variable2) + x, # Adjusted placebo mean
                  sd=sd(variable1))
      # Placebo
      X2 <- rnorm(n=length(variable2),
                  mean=mean(variable2),
                  sd=sd(variable2))
      # Welch T-test
      test <- t.test(X1, X2, alternative='less', var.equal=FALSE)
      # Store p-value
      pdif[i] <- test$p.value
      # Store means
      mean1[i] <- mean(X1)
      mean2[i] <- mean(X2)
    }
    power[y] <- mean(pdif < 0.05)
    meandif[y] <- mean(mean1) - mean(mean2)
  }
```

```
  return(cbind(meandif,power))
}

# Amount of simulations
n.sim <- 1000

# Simulate dbp_end
dbpend_power_sim <- power_f_delta(n.sim, data1$dbp_end, data2$dbp_end)
# Simulate dbp_dif
dbpdif_power_sim <- power_f_delta(n.sim, data1$dbpdif, data2$dbpdif)

# Plot the result
plot(dbpend_power_sim, type="l",col='red',
     xlab="Difference in means (observed mean difference of simulations)",
     main="Comparative power, fixed Delta",
     ylab="Power")
lines(dbpdif_power_sim, col='blue')
```

dbp_dif has more power when adjusting means by an absolute difference.

2. Analytical calculations on the observed Power

Relative difference in means

```
alpha <- 0.05
n <- n1 + n2
# Alpha is not divided by 2, as we test one-sided
t.alpha <- qt(1-alpha, df=n - 2)

# dbpdif: with a difference between groups of 42.5% of observed difference
delta_dif <- -(mu_dif1 - mu_dif2) * 0.425
lambda_dif <- delta_dif / sqrt(var(data1$dbpdif)/n1 + var(data2$dbpdif)/n2)
1 - pt(t.alpha, df=n - 2, ncp=lambda_dif)

# dbp_end: with a difference between groups of 42.5% of observed difference
delta_end <- -(mu_end1 - mu_end2) * 0.425
lambda_end <- delta_end / sqrt(var(data1$dbp_end)/n1 + var(data2$dbp_end)/n2)
1 - pt(t.alpha, df=n - 2, ncp=lambda_end)

# Non-parametric alternative: l (lambda) is too large, this doesn't work
#l_dif.2 <- sqrt((12 * n1 * n2) / (n + 1)) * delta_dif.2
#1 - pt(t.alpha, df=n - 2, ncp=l_dif.2)
```

Absolute difference in means

```
# power function to compare hard differences
powertest <- function(alpha=0.05, variable1, variable2, delta_dif){
  n1 <-length(variable1)
  n2 <- length(variable2)
  n <- n1+n2
```

```r
  # Alpha is not divided by 2, as we test one-sided
  t.alpha <- qt(1-alpha, df=n - 2)
  lambda_dif <- delta_dif /  sqrt(var(variable1)/n1 + var(variable2)/n2)
  print(1 - pt(t.alpha, df=n - 2, ncp=lambda_dif))
}

# Absolute difference of 3
powertest(alpha=0.05, data1$dbpdif, data2$dbpdif, delta_dif=3)
powertest(alpha=0.05, data1$dbp_end, data2$dbp_end, delta_dif=3)

# Absolute difference of 5
powertest(alpha=0.05, data1$dbpdif, data2$dbpdif, delta_dif=5)
powertest(alpha=0.05, data1$dbp_end, data2$dbp_end, delta_dif=5)
```

3. Multiple testing

dbp_end and dbp_dif are simulated simultaneously, but dependent on each other, under H0 and both are tested for a significant effect. For each simulation, the lowest p-value of the two is taken.

```r
# Means and SD
mu_end <- mean(data12$dbp_end)
mu_base <- mean(data12$dbp3)
sd_base1 <- sd(data1$dbp3)
sd_base2 <- sd(data2$dbp3)

# Array to store p-values
p_min <- vector(, 1000)
p_end <- vector(, 1000)
p_dif <- vector(, 1000)
# Set seed for reproducibility
set.seed(2242)

# Permutation
for(i in 1 : 1000) {
  # Sample with equal means to simulate H0
  # dbp_end
  sim_end1 <- rnorm(n1, mu_end, sd_end1)
  sim_end2 <- rnorm(n2, mu_end, sd_end2)
  # dbp3
  sim_base1 <- rnorm(n1, mu_base, sd_base1)
  sim_base2 <- rnorm(n2, mu_base, sd_base2)
  # sim_dif is calculated as a function of sim_end to simulate dependence
  sim_dif1 <- sim_end1 - sim_base1
  sim_dif2 <- sim_end2 - sim_base2
  # Test dbp_end and dbp_dif under H0
  t_test_end <- t.test(sim_end1, sim_end2, "less", paired=F, var.equal=F)
  t_test_dif <- t.test(sim_dif1, sim_dif2, "less", paired=F, var.equal=F)
  # Store P-values
```

```
  p_end[i] <- t_test_end$p.value
  p_dif[i] <- t_test_dif$p.value
  p_min[i] <- min(c(t_test_end$p.value, t_test_dif$p.value))
}

# Type 1 error rate of individual variables
mean(p_end < 0.05)
mean(p_dif < 0.05)
## Type I error rate is well-controlled for the individual variables

# Type 1 error rate of the testing method
mean(p_min < 0.05)
## However, the Type I error rate is too high when the lowest
## p-value of the two is used.
```

4. Bonferroni Correction

Bonferroni correction

Applying Bonferroni correction based on the previous result

```
mean((p_min * 2) < 0.05)
```

Calculate power

```
# Preparation
data1 <- data[data$treatment == "Treatment 1",]
data2 <- data[data$treatment == "Placebo",]
# Amount of observations
n1<- length(data1)
n2<- length(data2)
# Means
mean_end1 <- mean(data1$dbp_end)
mean_end2 <- mean(data2$dbp_end)
mean_dif1 <- mean(data1$dbpdif)
mean_dif2 <- mean(data2$dbpdif)
# SD
sd_end1 <- sd(data1$dbp_end)
sd_end2 <- sd(data2$dbp_end)
sd_dif1 <- sd(data1$dbpdif)
sd_dif2 <- sd(data2$dbpdif)
# Arrays to store the result
p_min <- vector()
p_end <- vector()
p_dif <- vector()
p_and <- vector()
# Set seed for reproducibility
set.seed(2242)
```

```r
# Simulate
for(i in 1 : 1000) {
  # The means in both groups are equal to simulate H0
  # dbp_end
  sim_end1 <- rnorm(n1, mean_end1, sd_end1)
  sim_end2 <- rnorm(n2, mean_end2, sd_end2)
  # dbp3
  sim_base1 <- rnorm(n1, mean_dif1, sd_dif1)
  sim_base2 <- rnorm(n2, mean_dif2, sd_dif2)
  # sim_dif is calculated as a function of sim_end to simulate dependence
  sim_dif1 <- sim_end1 - sim_base1
  sim_dif2 <- sim_end2 - sim_base2
  # Test dbp_end and dbp_dif under H0
  t_test_end <- t.test(sim_end1, sim_end2, "less", paired=F, var.equal=F)
  t_test_dif <- t.test(sim_dif1, sim_dif2, "less", paired=F, var.equal=F)
  # Apply Bonferroni correction
  p_end[i] <- p.adjust(t_test_end$p.value, method="bonferroni", n=2)
  p_dif[i] <- p.adjust(t_test_dif$p.value, method="bonferroni", n=2)
  # Store smallest corrected P-value
  p_min[i] <- min(c(t_test_end$p.value, method="bonferroni", n=2,
                    t_test_dif$p.value, method="bonferroni", n=2))
}
# Power
mean(p_min < 0.05)
```