Solution to the exercises day 8

Basic Statistics for health researchers 2020

23 November 2020

Warming up

The following code load the data and reshape it to the long format (same as in the R-demo file):

```
dW.pp <- read.table("prepost.txt")
dL.pp <- reshape2::melt(dW.pp,
                        id.var = c("subject","treatment","lesion"),
                        measure.vars = c("visual0","visual4"),
                        variable.name = "week",
                        value.name = "visual")
dL.pp <- dL.pp[order(dL.pp$subject),]
## convert week as factor with appropriate values
dL.pp$week <- factor(dL.pp$week,
                     level = c("visual0","visual4"),
                     labels = c(0,4))
## remove row names
rownames(dL.pp) <- NULL
```

2. There are several ways to compute summary statistics on subsets of the data in R. One can use basic R code:

```
vec.mean <- c(mean(dW.pp[dW.pp$treatment=="Active","visual0"]),
              mean(dW.pp[dW.pp$treatment=="Active","visual4"]),
              mean(dW.pp[dW.pp$treatment=="Placebo","visual0"]),
              mean(dW.pp[dW.pp$treatment=="Placebo","visual4"]))
vec.mean
```

```
[1] 54.42105 50.91228 55.26496 53.96581
```

and then re-arrange the result in a `data.frame`:

```
vec.treatment <- c("Active","Active","Placebo","Placebo")
vec.week <- c("0","4","0","4")
dftable <- data.frame(treatment = vec.treatment,
                      week = vec.week,
                      mu = vec.mean)
dftable
```

```
  treatment week       mu
1    Active    0 54.42105
2    Active    4 50.91228
3   Placebo    0 55.26496
4   Placebo    4 53.96581
```

Alternatively one could use the `aggregate` function, e.g.:

```
aggregate(visual ~ week + treatment,
          data = dL.pp, FUN = "mean")
```

```
  week treatment   visual
1    0    Active 54.42105
2    4    Active 50.91228
3    0   Placebo 55.26496
4    4   Placebo 53.96581
```

or `data.table`:

```
library(data.table)
dtL.pp <- as.data.table(dL.pp)
dtL.pp[,list(mu=mean(visual), sigma=sd(visual)),
                 by = c("treatment","week")]
```

```
   treatment week       mu    sigma
1:    Active    0 54.42105 14.59718
2:    Active    4 50.91228 15.81114
3:   Placebo    0 55.26496 15.11872
4:   Placebo    4 53.96581 15.90973
```

3. We can compute the average change in each treatment group:

4. directly from the observations, by first computing the change in vision between baseline and week 4 and then taking the average within each group:

```
dW.pp$visualDiff04 <- dW.pp$visual4 - dW.pp$visual0
c(mean(dW.pp[dW.pp$treatment=="Active","visualDiff04"]),
  mean(dW.pp[dW.pp$treatment=="Placebo","visualDiff04"]))
```

```
[1] -3.508772 -1.299145
```

- based on the already computed mean at each timepoint, by first extracting the estimated mean in each group:

```
mu.active <- dftable[dftable$treatment=="Active","mu"]
mu.placebo <- dftable[dftable$treatment=="Placebo","mu"]
```

and the computing the difference using the `diff` function:

```
change <- c(Placebo = diff(mu.active),
            Active = diff(mu.placebo))
change
```

```
  Placebo    Active
-3.508772 -1.299145
```

The estimate treatment effect is then:

```
as.double(change["Active"]-change["Placebo"])
```

```
[1] 2.209627
```

4. For the t-test, split the dataset according to treatmnet groups:

```
dW.ppA <- dW.pp[dW.pp$treatment=="Active",]
dW.ppP <- dW.pp[dW.pp$treatment=="Placebo",]
```

For the linear regression, define Placebo as the reference group:

```
dW.pp$treatment <- relevel(factor(dW.pp$treatment),"Placebo")
```

Run each approach:

```
library(nlme)
e.t.test <- t.test(x=dW.ppP$visualDiff04, y=dW.ppA$visualDiff04)
e.lm <- lm(visualDiff04 ~ treatment, data = dW.pp)
e.gls <- gls(visualDiff04 ~ treatment, data = dW.pp,
             weights = varIdent(form=~1|treatment))
```

output the estimated effect:

```
c("t-test" = as.double(diff(e.t.test$estimate)),
  "lm" = as.double(coef(e.lm)["treatmentActive"]),
  "gls" = as.double(coef(e.gls)["treatmentActive"]))
```

```
   t-test        lm       gls
-2.209627 -2.209627 -2.209627
```

and the associated uncertainty:

```
c("t-test" = as.double(e.t.test$stderr),
  "lm" = sqrt(vcov(e.lm)["treatmentActive","treatmentActive"]),
  "gls" = sqrt(vcov(e.gls)["treatmentActive","treatmentActive"]))
```

```
  t-test       lm      gls
1.094427 1.092472 1.094427
```

The estimates are the same in all approaches but the uncertainty differs (very slightly) between `lm` and `gls=/=t-tests`.

8. The p-value output after using `glht` are adjusted for multiple comparisons (here 2) while the ones of `summary(e.lm)` are unadjusted.

Exercise A: Longitudinal study

1. The `str` function reveals the presence of missing values in the dataset. We can also visualize see them when looking at the first rows of the dataset:

```
head(armdW)
```

```
  subject lesion line0 visual0 visual4 visual12 visual24 visual52 treat.f miss.pat
1       1      3    12      59      55       45       NA       NA  Active     --XX
2       2      1    13      65      70       65       65       55  Active     ----
3       3      4     8      40      40       37       17       NA Placebo     ---X
4       4      2    13      67      64       64       64       68 Placebo     ----
5       5      1    14      70      NA       NA       NA       NA  Active     XXXX
6       6      3    12      59      53       52       53       42  Active     ----
```

We can use `is.na` to test the presence of a missing value:

```
head(is.na(armdW))
```

```
     subject lesion line0 visual0 visual4 visual12 visual24 visual52 treat.f miss.pat
[1,]   FALSE  FALSE FALSE   FALSE   FALSE    FALSE     TRUE     TRUE   FALSE    FALSE
[2,]   FALSE  FALSE FALSE   FALSE   FALSE    FALSE    FALSE    FALSE   FALSE    FALSE
[3,]   FALSE  FALSE FALSE   FALSE   FALSE    FALSE    FALSE     TRUE   FALSE    FALSE
[4,]   FALSE  FALSE FALSE   FALSE   FALSE    FALSE    FALSE    FALSE   FALSE    FALSE
[5,]   FALSE  FALSE FALSE   FALSE    TRUE     TRUE     TRUE     TRUE   FALSE    FALSE
[6,]   FALSE  FALSE FALSE   FALSE   FALSE    FALSE    FALSE    FALSE   FALSE    FALSE
```

If we sum the number of missing value per column, we obtain the number of missing values for each variable:

```
colSums(is.na(armdW))
```

```
subject   lesion    line0  visual0  visual4 visual12 visual24 visual52  treat.f miss.pat
      0        1        0        0        9       13       26       45        0        0
```

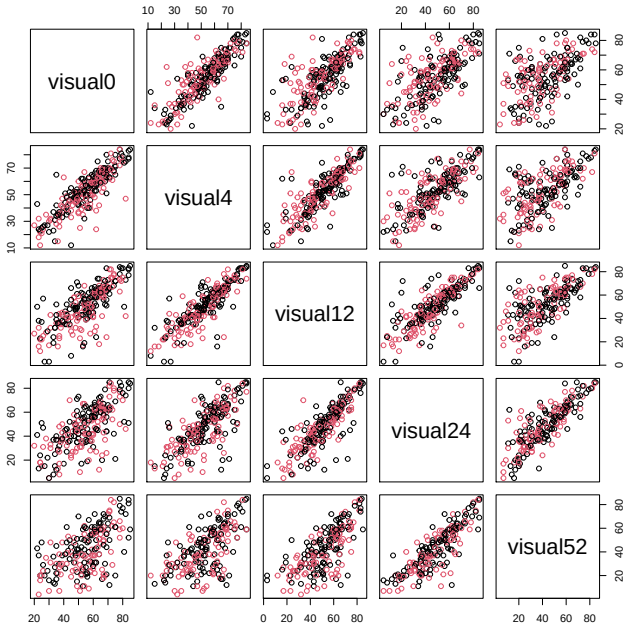
If we sum the number of missing value per row, we obtain the number of missing values per subject:

```
rowSums(is.na(armdW))[1:5]
```

```
[1] 2 0 1 0 4
```

2. We can use the wide format to make scatterplots:

```
vec.visual <- c("visual0", "visual4", "visual12",
                "visual24", "visual52")
plot(armdW[,vec.visual], col = armdW$treat.f)
```

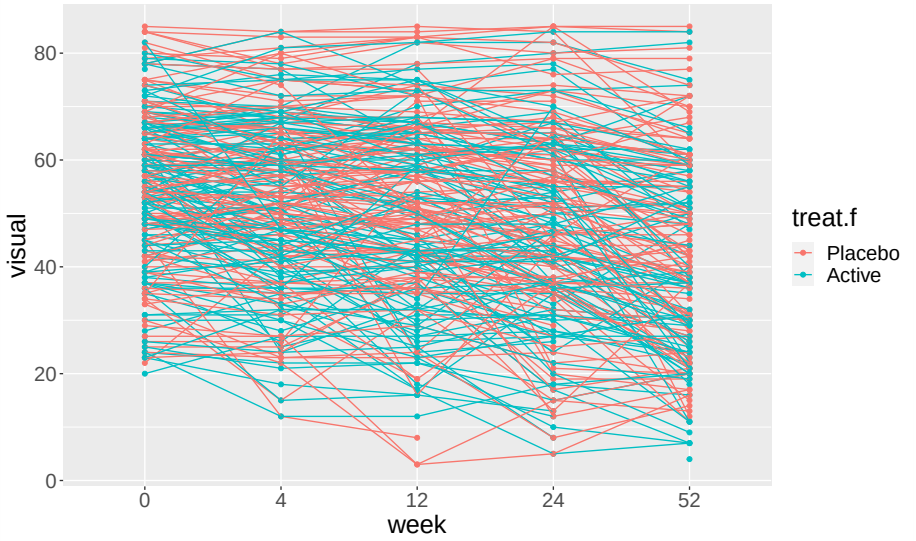


We can see the all pairs of measurement are correlated but the correlation seems to decrease with the time interval between the measurements (i.e. the further away we go from the diagonal). There does not seems to be any clear outlier but it is not straightforward to conclude about the time trend or the treatment effect.

Because there are many observations, a single spaghetti is hard too read:

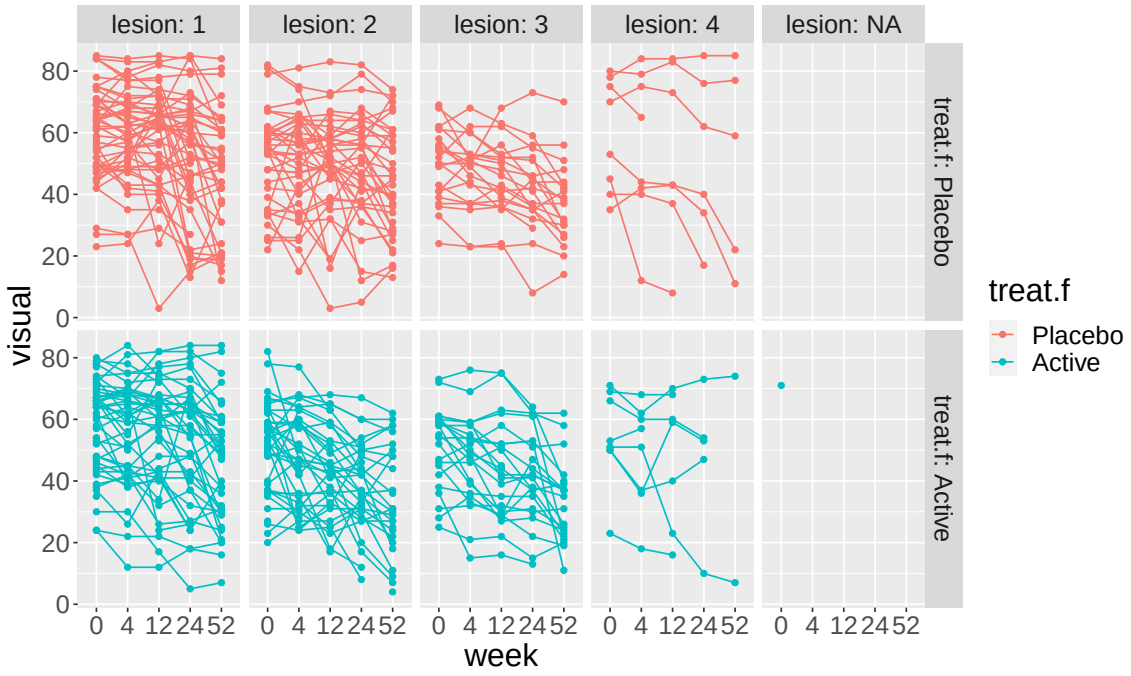
```
library(ggplot2)
gg <- ggplot(armdL, aes(x = week, y = visual,
                        group = subject, color = treat.f))
gg <- gg + geom_point() + geom_line()
gg
```

```
Warning messages:
1: Removed 93 rows containing missing values (geom_point).
2: Removed 84 row(s) containing missing values (geom_path).
```



It nevertheless seems to indicate that the values decrease over time. The variance seems also to slightly increase over time as the vision of some patients remains stable while the vision of other patients get worse. To have a better view we could subsample the dataset or we can do several spaghetti plots, one for each treatment group and type of lesion:

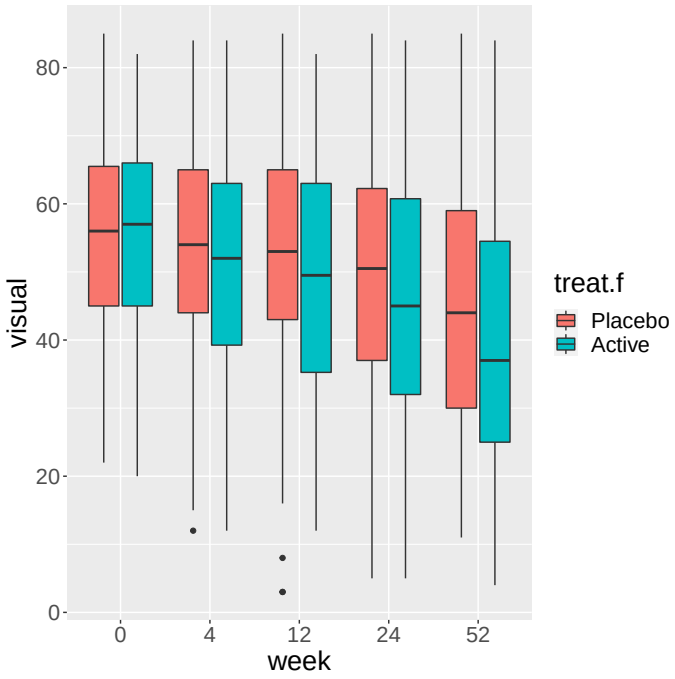
```
gg2 <- gg + facet_grid(treat.f~lesion, labeller = label_both)
gg2
```



Finally, we can also display a boxplot of the vision over time by treatment group:

```
gg <- ggplot(armdL, aes(x = week, y = visual,
                        fill = treat.f))
gg <- gg + geom_boxplot()
gg
```

```
Warning message:
Removed 93 rows containing non-finite values (stat_boxplot).
```



It confirms our first impression of decreasing values over time and increasing variance over time.

3. To compute summary statistics, we can subset the dataset in the wide format by treatment group:

```
armdW.p <- armdW[armdW$treat.f=="Placebo",]
armdW.a <- armdW[armdW$treat.f=="Active",]
```

and compute for instance the mean at a given timepoint:

```
mean(armdW.a$visual0)
```

```
[1] 54.57851
```

We can also compute the summary statistics in a more systematic way using `data.table`:

```
library(data.table)
dtL <- as.data.table(armdL)
dtL[,.(n = length(visual),
       missing = sum(is.na(visual)),
       observed = sum(!is.na(visual)),
       mean = mean(visual, na.rm=TRUE),
       std = sd(visual, na.rm=TRUE),
       min = min(visual, na.rm=TRUE),
       max = max(visual, na.rm=TRUE)),
    by = c("week","treat.f")]
```

```
    week treat.f   n missing observed     mean      std min max
 1:    0  Active 121       0      121 54.57851 14.82270  20  82
 2:    0 Placebo 119       0      119 55.33613 15.00129  22  85
 3:    4  Active 121       7      114 50.91228 15.81114  12  84
 4:    4 Placebo 119       2      117 53.96581 15.90973  12  84
 5:   12  Active 121      11      110 48.67273 17.47665  12  82
 6:   12 Placebo 119       2      117 52.87179 17.20091   3  85
 7:   24  Active 121      19      102 45.46078 18.08050   5  84
 8:   24 Placebo 119       7      112 49.33036 18.51242   5  85
 9:   52  Active 121      31       90 39.10000 18.40069   4  84
10:   52 Placebo 119      14      105 44.43810 18.53683  11  85
```

The number of missing values seems to increase over time, especially in the active group. In a real analysis, this would be concerning. Indeed, if patients with bad vision are more likely to drop out the mean computed at the later timepoints will be biased and not in a conservative way. It could also reflect side effects of the treatment that are so serious that the patients choose/have to leave study. However, for simplicity, we will ignore this problem in the latter questions and act as if censoring was independent of the outcome and of the treatment.

Finally, we can compute the correlation between measurements over time:

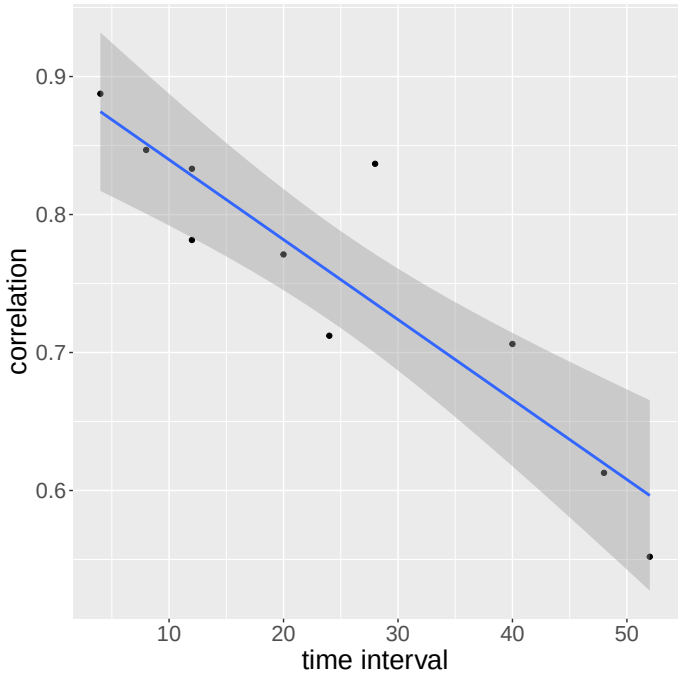
```
rho <- cor(armdW[,c("visual0","visual4","visual12",
                    "visual24","visual52")], use = "complete")
rho
```

```
          visual0   visual4  visual12  visual24  visual52
visual0  1.0000000 0.8875360 0.7815256 0.7121620 0.5519911
visual4  0.8875360 1.0000000 0.8468003 0.7710692 0.6127882
visual12 0.7815256 0.8468003 1.0000000 0.8331742 0.7061752
visual24 0.7121620 0.7710692 0.8331742 1.0000000 0.8367625
visual52 0.5519911 0.6127882 0.7061752 0.8367625 1.0000000
```

As one would expect the measurements are strongly correlated. The correlation however typically decreases when the time interval between two measurements increases:

```
position <- which(lower.tri(rho), arr.ind =TRUE)
position[] <- c(0,4,12,24,52)[as.double(position)]
df.rho <- data.frame(interval = position[,1] - position[,2],
                     correlation = rho[lower.tri(rho)])
gg <- ggplot(df.rho, aes(x = interval, y = correlation))
gg <- gg + geom_point() + geom_smooth(method = "lm")
gg <- gg + xlab("time interval")
gg
```

```
`geom_smooth()` using formula 'y ~ x'
```



4. We use the wide format to make multiple t-tests.

   First we compute for each follow-up time the change from baseline:

```
armdW.p$change4 <- armdW.p$visual4 - armdW.p$visual0
armdW.p$change12 <- armdW.p$visual12 - armdW.p$visual0
armdW.p$change24 <- armdW.p$visual24 - armdW.p$visual0
armdW.p$change52 <- armdW.p$visual52 - armdW.p$visual0

armdW.a$change4 <- armdW.a$visual4 - armdW.a$visual0
armdW.a$change12 <- armdW.a$visual12 - armdW.a$visual0
armdW.a$change24 <- armdW.a$visual24 - armdW.a$visual0
armdW.a$change52 <- armdW.a$visual52 - armdW.a$visual0
```

   Finally we can perform one t-test at each follow-up time:

```
ttest4 <- t.test(armdW.a$change4,armdW.p$change4)
ttest12 <- t.test(armdW.a$change12,armdW.p$change12)
ttest24 <- t.test(armdW.a$change24,armdW.p$change24)
ttest52 <- t.test(armdW.a$change52,armdW.p$change52)
```

   For instance at the last follow-up time we find that the estimated decrease in vision is 15.48 in the active group and 11.18 in the control group:

```
ttest52
```

```
        Welch Two Sample t-test

data:  armdW.a$change52 and armdW.p$change52
t = -1.8842, df = 191.47, p-value = 0.06106
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -8.7949525  0.2013017
sample estimates:
mean of x mean of y
-15.47778 -11.18095
```

   The corresponding treatmnent effects are:

```
c(week4 = as.double(diff(ttest4$estimate)),
  week12 = as.double(diff(ttest12$estimate)),
  week24 = as.double(diff(ttest24$estimate)),
  week52 = as.double(diff(ttest52$estimate)))
```

```
   week4    week12    week24    week52
2.209627 3.608314 3.363270 4.296825
```

If we know before hand that we want to look at the last timepoint, we can also report the confidence interval and the p-value. However if we decided to select the last timepoint because it had the largest effect, we need to adjust for the fact that we are no more looking at the effect at week 52 but at the largest effect over week 4, 12, 24 and 52. Indeed, applying the "max" impacts the distribution (and thus the uncertainty) of the estimate. A Bonferroni adjustment is an option, even though it is (too?) conservative:

```
p.adjust(c(ttest4$p.value, ttest12$p.value,
           ttest24$p.value, ttest52$p.value),
         method = "bonferroni")
```

```
[1] 0.1787437 0.0835091 0.3195546 0.2442350
```

There are three limitations of the t-test approach:

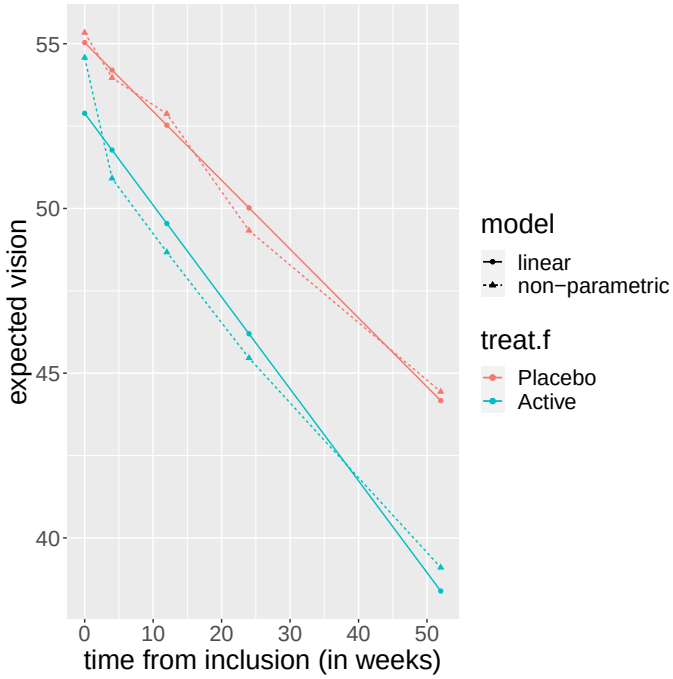
- it makes it more complex (but not impossible) to summarize the treatment effect over time or to efficiently adjust for multiple comparisons. This is because the estimation of the treatment effect is performed separately at each timepoint.
- it is sometimes too flexible, e.g., one may want to model that the treatment effect should be proportional/a decreasing function of the number of weeks under treatment.
- we don't make full use of the data, i.e. exploit the correlation between the measurement: a patient who has had no change until week 24 is likely to not change at week 52. So have incomplete follow-up should still be somehow informative when estimating the treatment effect. This can be especially relevant here: instead of assuming that censoring is independent of the outcome, we could assume that censoring at time t is independent of the outcome at time t but may depend on the previous outcome(s).

5. The difference between the two models is how the treatment effect is modeled. Let's compute the fitted values by each model:

```
X$fit.linear <- predict(e.linear, newdata = X)
X$fit.np <- predict(e.np, newdata = X)
```

and display them:

```
gg <- ggplot(X, aes(x = week.num, group = treat.f, color = treat.f))
gg <- gg + geom_line(aes(y = fit.linear, linetype = "linear"))
gg <- gg + geom_line(aes(y = fit.np, linetype = "non-parametric"))
gg <- gg + geom_point(aes(y = fit.linear, shape = "linear"))
gg <- gg + geom_point(aes(y = fit.np, shape = "non-parametric"))
gg <- gg + xlab("time from inclusion (in weeks)")
gg <- gg + ylab("expected vision")
gg <- gg + labs(shape = "model", linetype = "model")
gg
```



We can see the model `e.linear` model a linear decrease in vision over time, while the model `e.np` do not put such a constraint. However in this study a linear decrease appears to fit well-enough the data:

```
anova(e.linear,e.np)
```

```
Analysis of Variance Table

Model 1: visual ~ treat.f * week.num
Model 2: visual ~ treat.f * week
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1   1103 315778
2   1097 315072  6    705.87 0.4096  0.873
```

and makes it much simpler to test and communicate about a possible treatment effect:

```
summary(e.linear)$coef
```

```
                           Estimate Std. Error   t value     Pr(>|t|)
(Intercept)            55.03540122 0.98256155 56.012167 0.000000e+00
treat.fActive          -2.14681455 1.39850004 -1.535084 1.250501e-01
week.num               -0.20899987 0.03871455 -5.398483 8.225443e-08
treat.fActive:week.num -0.06988277 0.05623126 -1.242775 2.142151e-01
```

Here the expected vision in the placebo group is 55 at baseline and decreases by 0.21 point per week. The expected vision in the active group is 53.86 (=55-2.14) at baseline and decreases by 0.28 (0.21+0.07) point per week. While the decrease over time is statistically significant, the treatment effect (at baseline or over time) are not.

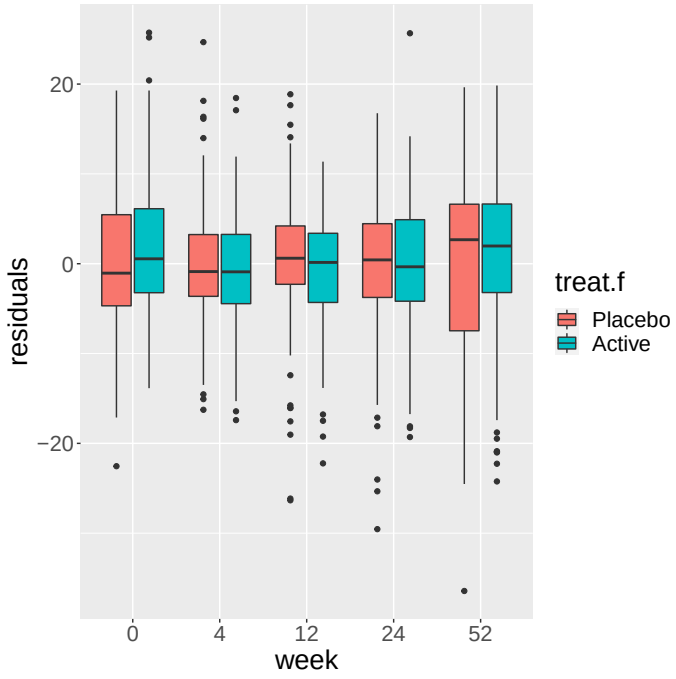
The linear regression `e.np` is the closest to the multiple t-test approach. However, both linear regressions (`e.np` and `e.linear`) incorrectly assume that all the observations are independent and that the variance is the same between groups and over time.

6. We are assuming:

- that the variability of the residuals is the same in both treatment groups and is constant over time.
- that the within-subject correlation between the residuals is constant, i.e. independent of the time interval between measurements.

We can first make a boxplot of the residuals by group over time:

```
gg <- ggplot(armdL.cc, aes(x = week, y = residuals, fill = treat.f))
gg <- gg + geom_boxplot()
gg
```



The assumption of same residual variance between treatment groups seems reasonnable. However the residual variance seems to vary over time, being for instance lower at week 4 and 12. We can also compute the within-subject correlation between residuals:

```
cor(armaW2.cc[,c("0","4","12","24","52")], use = "complete")
```

```
            0          4          12         24         52
0   1.0000000  0.47486844 -0.14871074 -0.5436564 -0.6494628
4   0.4748684  1.00000000  0.02607296 -0.4680554 -0.6740261
12 -0.1487107  0.02607296  1.00000000 -0.1267222 -0.3294881
24 -0.5436564 -0.46805541 -0.12672224  1.0000000  0.2949542
52 -0.6494628 -0.67402615 -0.32948812  0.2949542  1.0000000
```

and see that it is clearly not constant.

We can display the new fitted value using the same approach as before:

```
X$fit.lme <- predict(e.lme, newdata = X, level = 0)

gg <- ggplot(X, aes(x = week.num, group = treat.f, color = treat.f))
gg <- gg + geom_line(aes(y = fit.lme))
gg <- gg + geom_point(aes(y = fit.lme))
gg <- gg + xlab("time from inclusion (in weeks)") + ylab("expected
    vision")
gg
```

and a test for the treatment effect can be obtained from the summary:

```
summary(e.lme)$tTable
```

```
                            Value  Std.Error  DF    t-value       p-value
(Intercept)            55.14359253 1.42309210 865  38.749138 3.094222e-191
treat.fActive          -2.06324228 2.00860639 238  -1.027201  3.053687e-01
week.num               -0.21531457 0.02034942 865 -10.580871  1.093168e-24
treat.fActive:week.num -0.07525012 0.02978514 865  -2.526431  1.170017e-02
```

We can see that the estimation of the treatment effect over time and the corresponding p-value have changed compared to the (INCORRECT) linear model.

7. We can re-use once more the same approach:

```
X$fit.gls <- predict(e.gls, newdata = X)

gg <- ggplot(X, aes(x = week.num, group = treat.f, color = treat.f))
gg <- gg + geom_line(aes(y = fit.lme, linetype = "random intercept"))
gg <- gg + geom_line(aes(y = fit.gls, linetype = "unstructured"))
gg <- gg + geom_point(aes(y = fit.lme, shape = "random intercept"))
gg <- gg + geom_point(aes(y = fit.gls, shape = "unstructured"))
gg <- gg + xlab("time from inclusion (in weeks)") + ylab("expected
    vision")
gg <- gg + labs(shape = "model", linetype = "model")
gg
```

and see that the estimate of the treatment effect over time and its p-value have changed a little bit compared to the random intercept model:

```
summary(e.gls)$tTable
```

```
                            Value  Std.Error    t-value       p-value
(Intercept)            55.35275376 1.35645912 40.8067984 1.257776e-222
treat.fActive          -1.22259440 1.91075886 -0.6398476  5.224046e-01
week.num               -0.21115437 0.02903629 -7.2720839  6.696667e-13
treat.fActive:week.num -0.07676322 0.04235127 -1.8125361  7.017518e-02
```

Because this model do not rely on unrealistic assumptions, it is to be prefered over the random intercept model.

Exercise B: Data visualisation

1. The diagonal elements (`sd1^2`, `sd2^2`, `sd3^3`) represent the variance of the endpoint at the various measurement times. The off-diagonal element are the covariance between two measurements performed in the same individual at different timepoints. Here since the variance is 1 at all timepoints the covariance is the same as the correlation. The first off diagonal (element `cov`) represent the correlation between two measurements performed at two consecutive timepoints (e.g. baseline and week 4) while `cov2` is the correlation between two measurements performed at two non-consecutive timepoints (baseline and week 8). One would expect that the longer the time interval between two measurements, the lower the correlation and this is why `cov2` < `cov`.

2. We use the `mean` function to compute the mean and the `cov` function to compute the covariance:

```
c(mean(dW[,"X1"]),mean(dW[,"X2"]),mean(dW[,"X3"]))
```

```
[1] 0.01280056 1.06239273 1.94829877
```

```
cov(dW[,c("X1","X2","X3")])
```

```
          X1        X2        X3
X1 1.1694945 0.8077581 0.6218703
X2 0.8077581 1.1405203 0.8881733
X3 0.6218703 0.8881733 1.1212154
```

The true values are:

```
mu
```

```
[1] 0 1 2
```

and

```
Sigma
```

```
     X1  X2   X3
X1 1.00 0.7 0.49
X2 0.70 1.0 0.70
X3 0.49 0.7 1.00
```

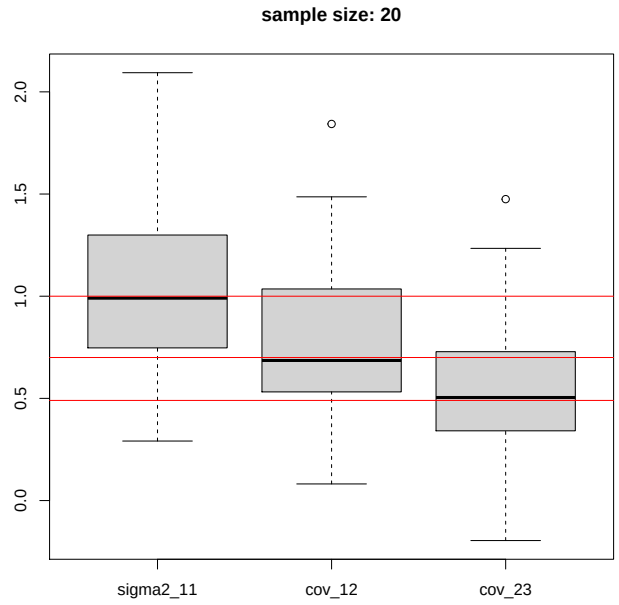
We can see that the estimate values are rather close to the true values. If we were to decrease the sample size, our estimates of the mean and variance would be less precise meaning that they are more likely to be far away from the true one, e.g.:

```
n2 <- 20
set.seed(10)
dW2 <- data.frame(id = 1:n2, rmvnorm(n2, mean = 0:2, sigma = Sigma))
cov(dW2[,c("X1","X2","X3")])
```

```
          X1       X2        X3
X1 0.7560959 0.651642 0.6348051
X2 0.6516420 1.021912 0.9873450
X3 0.6348051 0.987345 1.2500082
```

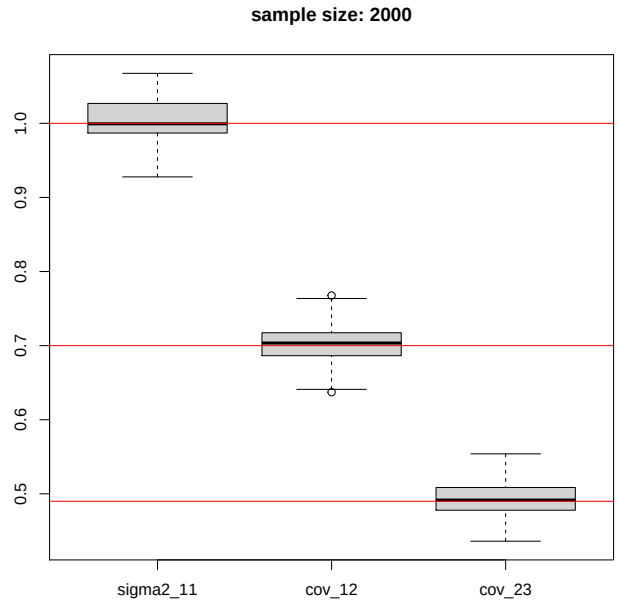
We can in fact assess how precise our estimates are by generating several datasets (here 20), computing the covariance matrix, and displaying the distribution (over datasets) of the estimated coefficients:

```
set.seed(10)
df.res <- do.call(rbind,lapply(1:100, function(i){
  dW2 <- data.frame(id = 1:n2, rmvnorm(n2, mean = 0:2, sigma = Sigma))
  Sigma.hat <- cov(dW2[,c("X1","X2","X3")])
  return(setNames(Sigma.hat[,1], c("sigma2_11","cov_12","cov_23")))
}))
boxplot(df.res, main = "sample size: 20")
abline(h = c(sd1^2,cov,cov2), col = "red")
```



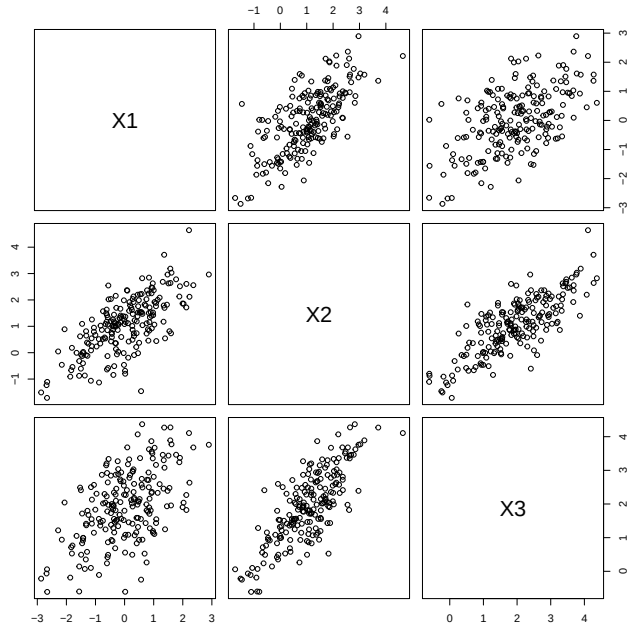
On the other hand, having 10 times more observations (here 2000) would lead to much more precise estimates:

```
n3 <- 2000
set.seed(10)
df.res <- do.call(rbind,lapply(1:100, function(i){
  dW2 <- data.frame(id = 1:n3, rmvnorm(n3, mean = 0:2, sigma = Sigma))
  Sigma.hat <- cov(dW2[,c("X1","X2","X3")])
  return(setNames(Sigma.hat[,1], c("sigma2_11","cov_12","cov_23")))
}))
boxplot(df.res, main = "sample size: 2000")
abline(h = c(sd1^2,cov,cov2), col = "red")
```



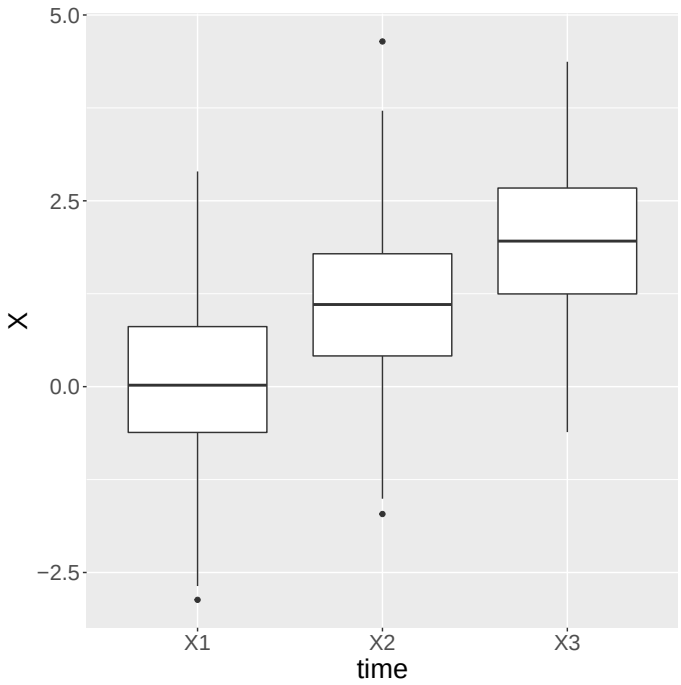
3. A scatterplot of the data can be obtain using the `plot` function:

```
plot(dW[,c("X1","X2","X3")])
```



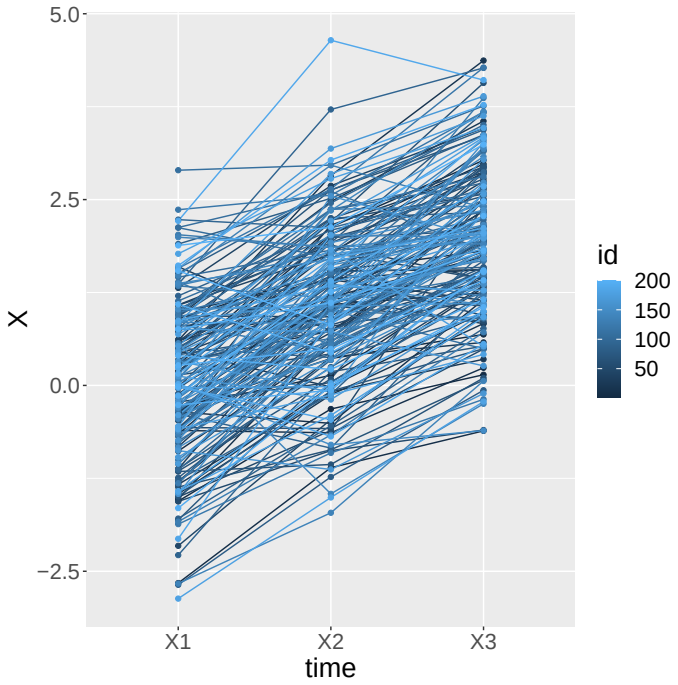
A boxplot of the data can be obtained using the `boxplot` function or `geom_boxplot`:

```
boxplot(X ~ time, data = dL)
ggplot(dL, aes(x = time, y = X)) + geom_boxplot()
```



Finally a spaghetti plot can be obtained using the `matplot` function or `geom_line`, specifying to group the observation by `id`:

```
matplot(t(dW[,c("X1","X2","X3")]), type = "l")
ggplot(dL, aes(x = time, y = X, group = id, color = id)) + geom_line()
    + geom_point()
```



A boxplot gives a very concise and readable representation of the data, even when with many timepoints and with a large number of observations. One can quickly identify trends in mean and variance over repetitions. One may also be able to identify certain deviation to normality (e.g. asymetry). It, however, does give any information about the correlation between measurements. So in some sense it may exagerate the variability. It is also not well suited to identify subgroups (e.g. half of the people respond to the treatment and the other half do not).
Spaghetti plots are well suited when there is an ordering of the repetitions (e.g. over time, this is not the case when looking at several brain regions though). They can be used to visually assess the correlation over time and detect groups of observations (e.g. some go up and some go down). However when the sample increase, they become hard to read and one should consider displaying subsets of the observations. Scatterplots give the full picture of the data when having only 2 measurements but becomes increasingly hard to read as the number of timepoints increases (as the number of plots increases).

4. The correlation can be varied via the argument `rho` between -1 and 1. This will not affect the boxplot but it will affect the scatterplot (values more or less aligned along the identity lign) and the spaghetti-plot (slope of the line connecting the dots).
   Varying the variance will affect all graph by essentially increasing the dispersion of the points.
   Varying the sample size have a limited affect on the boxplot. Large sample size will make the spaghetti plot and, to a lesser extend, the scatter plot more difficult to read.

5. We can identify the mean and standard deviation for the endpoint at two times, as well as their correlation. See slide 32 in the lecture.
   The color scale represent the amount / frequence of observations we find in a specific region. For instance most observations are around $X1 = 0$ and $X2 = 1$. The further way from this point the less likely it is to find an observation, though when the two endpoints are correlated, the decrease is not the same in all directions.
   This type of graph involves a lot of processing of the data to give a smooth representation and can sometimes be misleading, especially when there are only few observations. So it is mostly useful in large dataset when one wants to investigate the precise relationship between two variables (go beyond "they are correlated").