Supplementary

Complete analysis for clinical pharmacokinetic studies

# Basics and housekeeping

To easily find and understand all your hard work when you want to look at it a year from now, it is a good idea to keep your data work nice and tidy.

In RStudio you can create projects for the different tasks you are working on, and we recommend you start by creating a new project for the work you will be doing in this tutorial. Click File and then New Project. R creates a new folder in the designated directory with a .RProj file. In this new folder, you might also create a folder called “Data”, where you save the raw data (**Supplementary Table S1**), and a folder called “R”, where you save the R code.

Another option is to simply set the working directory. The working directory is the path on your computer that sets the default location of any files you read into R or save out of R.

# Set my working directory to the following path.  
setwd(dir = "/Users/acdunvald/OneDrive - Syddansk Universitet/[3] Projects/Tutorial")

## Getting ready

When starting a new project, you first have to call the packages needed to perform this project. Here is the list of packages we used for this specific project. All of them are installed in advance. If any of the packages are missing and needs to be installed in your library, R will warn you when you run the code. Then you simply install the package by calling install.packages().

For this project, you will need the following packages:

* **readxl**: For importing excel data and returning a tibble/data frame.
* **reshape2**: For reshaping data from long to wide format.
* **tidyverse**: Is a set of packages that work in harmony. When installing or loading the tidyverse package you get multiple core packages at the same time, including dplyr, ggplot2, readxl, and many more. The packages included are described here, but are not independently loaded (in the code below) as they are imbedded in the tidyverse package.
* **dplyr**: Needed for some function e.g. %>%.
* **ncappc**: A package for (i) traditional non-compartmental analysis (NCA) and (ii) simulation-based posterior predictive checks for population pharmacokinetic (PK) and/or pharmacodynamic (PKPD) models using NCA metrics. [See reference manual](https://cran.r-project.org/web/packages/ncappc/index.html).
* **Publish**: Functions needed to calculate 95% CI for AUC, etc.
* **ggpubr**: To check for normal distribution before performing the t-test.
* **ggplot2**: For plotting using ggplot.
* **ggthemes**: [Themes beyond the basic ggplot themes.](https://yutannihilation.github.io/allYourFigureAreBelongToUs/ggthemes/)
* **RColorBrewer**: [Color palette for R.](https://www.r-graph-gallery.com/38-rcolorbrewers-palettes.html)
* **Tmaptools**: The function palette\_explorer let you explore Color Brewer Palettes.

# Load the required packages.  
library(readxl)  
library(reshape2)  
library(tidyverse)   
library(ncappc)  
library(Publish)   
library(ggpubr)   
library(ggthemes)   
library(RColorBrewer)   
library(tmaptools)

Next, you need to import the dataset you are going to work on. Store it as a dataframe object by adding your file path in the read\_excel/read\_xlsx/read\_xls function. In this case the **Supplementary Table S1** Excel spreadsheet file (called RawData) is imported and stored as a data frame/tibble in the object named “RawPK”:

# Import data.  
RawPK <- read\_excel("~/OneDrive - Syddansk Universitet/[3] Projects/Tutorial/Data/RawData.xlsx")

Column abbreviations are:

* **ID**: Trial subject ID
* **TAD**: Time After Dose (hours). In this mock dataset we only created one time variable. In a real setting, you may work with both planned\_TAD (for plotting) and observed\_TAD (for statistical analysis)
* **DOSE**: Dose (ng) - please note the conversion from mg to ng.
* **DV**: Dependent variable (midazolam plasma concentration in ng/ml)
* **IND**: Induction. 0 = no induction, 1 = induction

Now you have made all the preparations and are ready to work with your data.

# Section I

## Plotting a plasma concentration-time curve for Figure 2

Generate a new dataset (CTdata) that contains mean and standard deviation of the plasma concentration (DV) at each time point (TAD) for the two groups, no induction (IND=0) and induction (IND=1). Use Summarize() to compute mean plasma concentration from all the plasma concentration measurements for a given time point (TAD) with and without induction. This is done by grouping on TAD and IND; summarizing then yields mean plasma concentration of the 12 study subjects at each of the 25 time points for each group, with and without induction. New columns with mean concentration and standard deviation are created in the CTdata dataset. The %>% operator (called pipe) allows an efficient and easily readable code.

# Generate a new dataset with columns for mean concentration and standard deviation (SD).  
CTdata <- RawPK %>%  
 group\_by(TAD, IND) %>%  
 summarise(mean\_conc = mean(DV), SD\_conc = sd(DV))

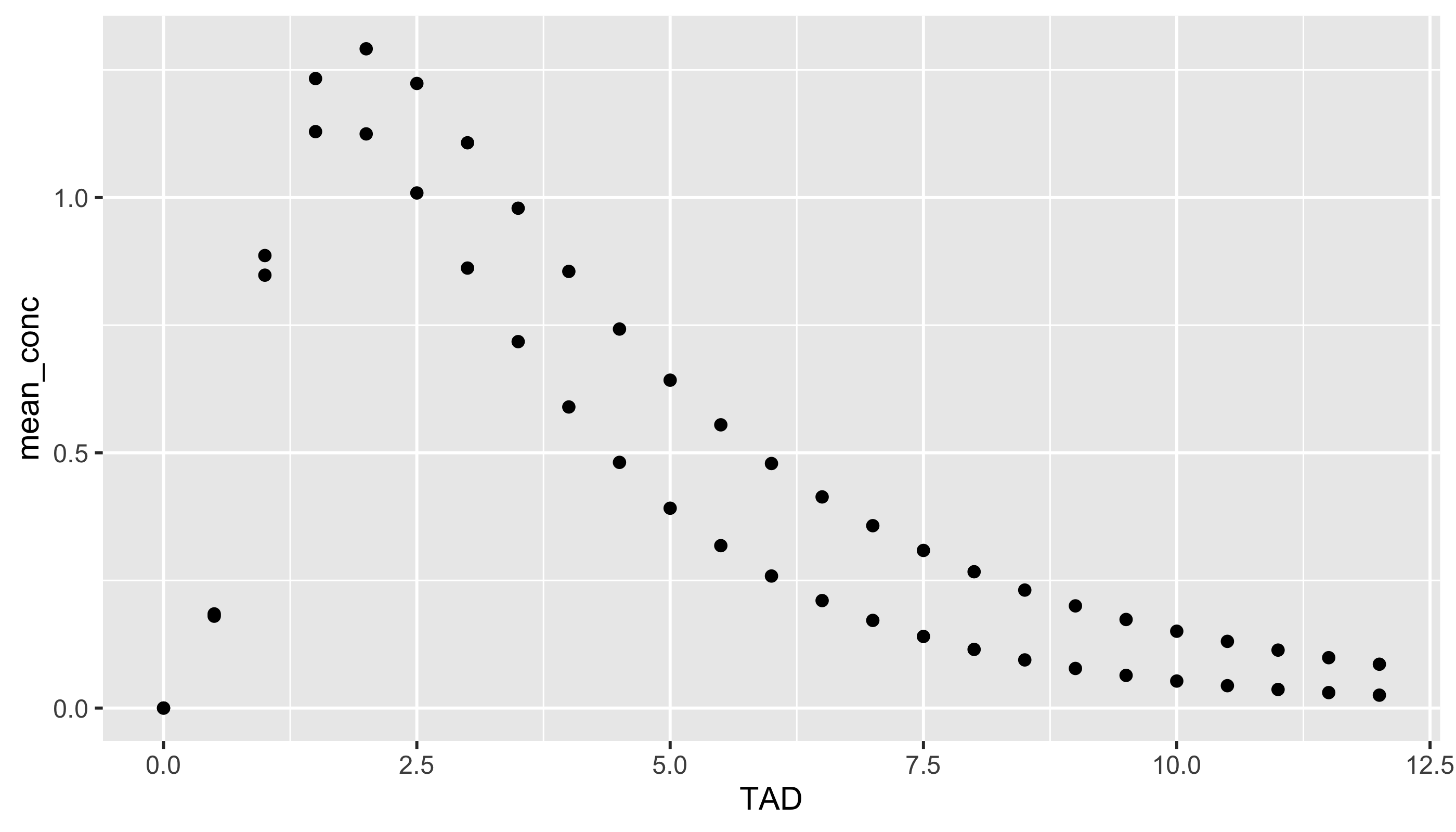
## `summarise()` regrouping output by 'TAD' (override with `.groups` argument)

Relabel the entries in the column IND from a numeric object of 0 or 1 to the corresponding character object/string of “No inducer” or “Inducer”.

# Recode IND from numeric to character.  
CTdata$IND <- recode(CTdata$IND, "0" = "No inducer", "1" = "Inducer")

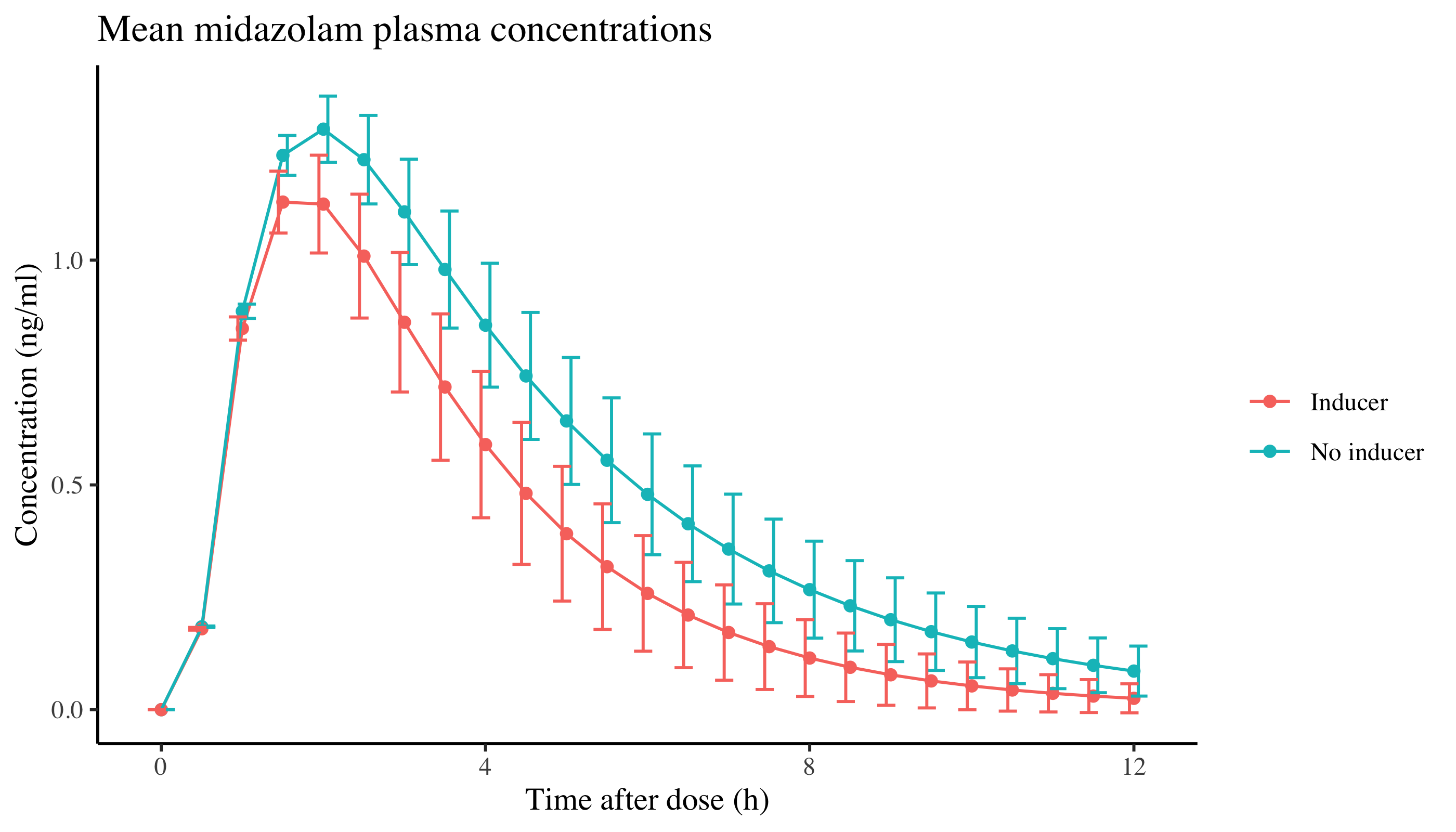
Plot the data in a simple concentration-time curve with ggplot by calling the ggplot function. Plotting can be done in steps by continuously adding layers/functions (and hence complexity) to the plot. In a simple and basic plot, as illustrated below, we have plotted the individual data points (geom\_point) grouped by induction in a graph where the x-axis is time and the y-axis is mean midazolam plasma concentrations.

# Plotting a simple graph of mean midazolam plasma concentration and time.  
ggplot() +   
 geom\_point(data = CTdata, aes(x=TAD, y=mean\_conc, group=IND))



To increase the amount of information presented in the graph and to improve readability, we suggest to apply another two layers to the graph: geom\_line for connecting the mean concentrations and geom\_errorbar for the standard deviations (SD) of concentrations. Colours, labels, and themes are also defined. A lot of possibilities exist to work around with colors and other graphics e.g. RColorBrewer, ggthemes, and palette\_explorer(). Colors can be set by adding a layer to ggplot e.g. + scale\_colour\_manual(values =c("blue", "red")). Also, hex values can be listed instead of names using e.g. palette\_explorer() or the colourpicker package.

# Plotting mean midazolam plasma concentration (and standard deviation (SD)) and time.  
ggplot() +  
geom\_point(data = CTdata, aes(x=TAD, y=mean\_conc, group=IND, color=IND)) +  
geom\_line(data = CTdata, aes(x=TAD, y=mean\_conc, group=IND, color=IND)) +  
geom\_errorbar(data=CTdata, aes(x=TAD, y = mean\_conc, ymax = mean\_conc + SD\_conc, ymin = mean\_conc - SD\_conc, color=IND),  
 position = position\_dodge(width=0.22)) +  
 labs(x ="Time after dose (h)", y = "Concentration (ng/ml)", title ="Mean midazolam plasma concentrations", color = "") +  
 theme\_classic() +  
 theme(text=element\_text(family="Times"))

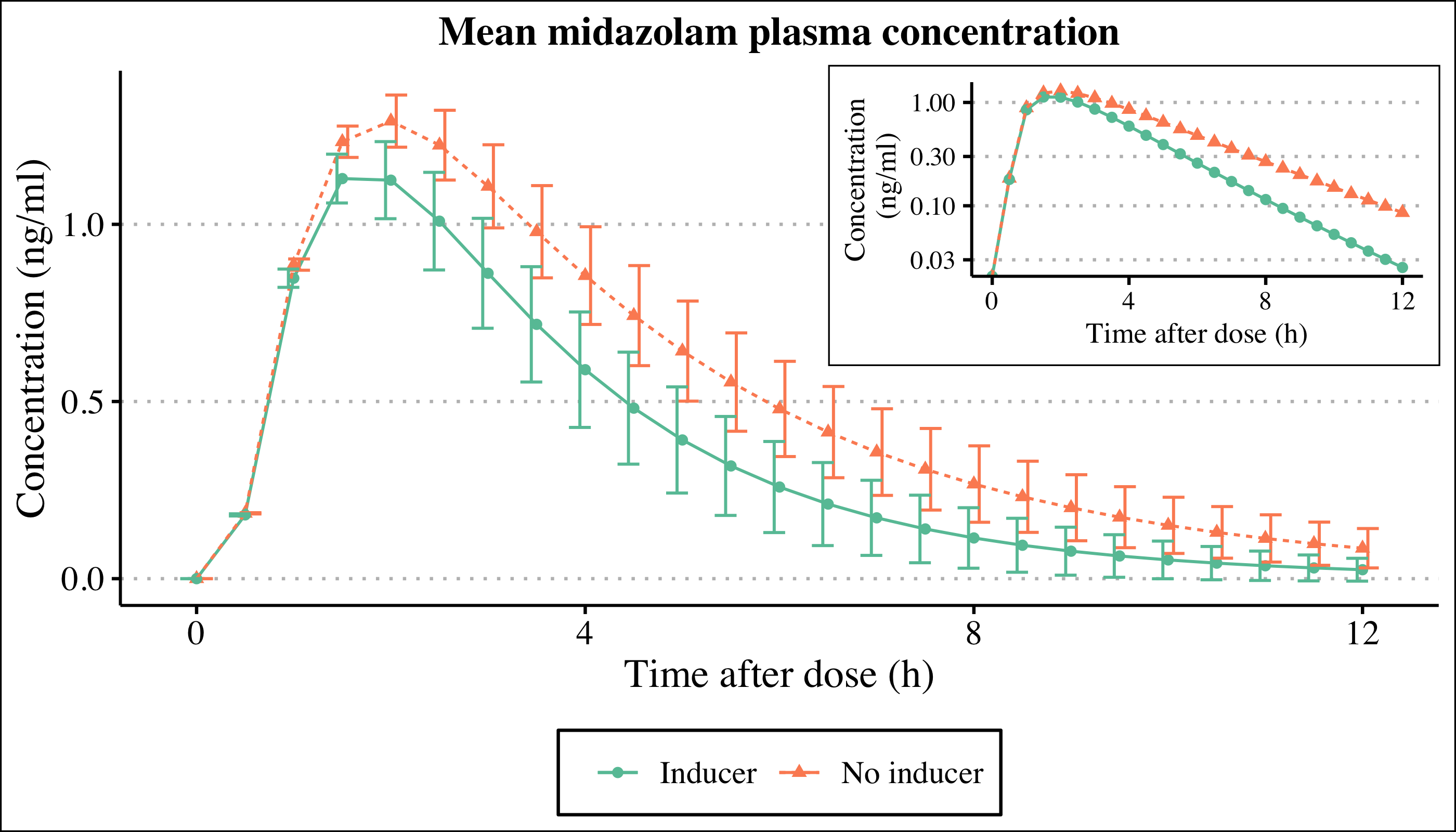


Create a plot with logarithmic y-axis and save it as an object “log\_plot” that will be used as an inset in the main graph.

# Create y-log scale graph for inset in the main graph.  
log\_plot <- ggplot(data = CTdata, aes(x = TAD, y = mean\_conc, group = IND)) +  
 geom\_point(aes(color = IND, shape = IND)) +  
 geom\_line(aes(linetype = IND, color = IND)) +  
 labs(x = "Time after dose (h)", y = "Concentration\n (ng/ml)") +  
 scale\_color\_brewer(palette = "Set2") +  
 theme\_clean() +  
 theme(plot.title = element\_text(hjust = 2), legend.position = "none",   
 text=element\_text(size=12)) +  
 scale\_x\_continuous(limits=c(0, 12), breaks=c(0, 4, 8, 12)) +  
 scale\_y\_log10() +  
 theme(text=element\_text(family="Times"))

# Combine the main graph and the y-log scale graph in one graph.   
ggplot(data = CTdata, aes(x = TAD, y = mean\_conc, group = IND)) +  
 geom\_point(aes(color = IND, shape = IND)) +  
 geom\_line(aes(linetype = IND, color = IND)) +  
 geom\_errorbar(aes(ymax = mean\_conc + SD\_conc, ymin = mean\_conc - SD\_conc, color = IND),   
 position=position\_dodge(width = .22)) +  
 labs(x = "Time after dose (h)", y = "Concentration (ng/ml)", title = "Mean midazolam plasma concentration", shape = "Induction", color = "Induction", linetype = "Induction") +  
 scale\_color\_brewer(palette = "Set2") +  
 theme\_clean() +  
 theme(plot.title = element\_text(hjust = 0.5), legend.position = "bottom", legend.title = element\_blank(), legend.text=element\_text(family="Times"), axis.text=element\_text(size=12), axis.title=element\_text(size=14),text=element\_text(family="Times")) +  
 annotation\_custom(ggplotGrob(log\_plot), xmin = 6.5, xmax = 12.8,   
 ymin = 0.6, ymax = 1.45)

## Warning: Transformation introduced infinite values in continuous y-axis  
  
## Warning: Transformation introduced infinite values in continuous y-axis

  
The warning message "Transformation introduced infinite values in continuous y-axis" is due to the concentration of 0 ng/ml at 0 hours. Log(0) is not defined, but is approaching negative infinity.

# Section II

## Non-compartmental pharmacokinetic analysis

Non-compartmental analysis can be done using the ncappc package. See the ncappc vignette and manual via this [link.](https://cran.r-project.org/web/packages/ncappc/index.html)

# Compute the non-compartmental pharmacokinetic analysis.  
ncappc(  
 obsFile = RawPK,  
 str1Nm = "IND",  
 str1 = c(0,1),  
 concUnit = "ng/ml",  
 timeUnit = "hours",  
 doseUnit = "ng",  
 idNmObs = "ID",  
 timeNmObs = "TAD",  
 concNmObs = "DV",  
 doseAmtNm = "DOSE",  
 onlyNCA = TRUE,  
 LambdaTimeRange = "NULL",  
 method = "linearup-logdown",  
 studyName = "PK Tutorial",  
 outFileNm = "WOW\_PK\_TUTORIAL",  
 extrapolate = TRUE,  
 gg\_theme = theme\_classic()  
)

## [1] "Note: Incorrect time range for Lambda calculation. LambdaTimeRange will not be used."  
## [1] "Note: EVID column is not present. EVID will not be used to process the observed data."

## Warning: The `add` argument of `group\_by()` is deprecated as of dplyr 1.0.0.  
## Please use the `.add` argument instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_warnings()` to see where this warning was generated.

##   
## processing file:ncappc\_report-WOW\_PK\_TUTORIAL.Rmd

## Output created:  
## /Users/acdunvald/Library/CloudStorage/OneDrive-SyddanskUniversitet/[3] Projects/Tutorial/ncappc\_report-WOW\_PK\_TUTORIAL.html

When creating this tutorial, several group members were unable to run the non-compartmental analyses and received this error message: Error in melt\_dataframe(data, as.integer(id.ind - 1), as.integer(measure.ind - : function 'Rcpp\_precious\_remove' not provided by package 'Rcpp' This might be solved by install.packages('Rcpp') or update.packages(). In one case, additional updates of packages (including a vctrs package) was needed in order to run the non-compartmental analyses.

The analyses above provides you with multiple new files that are saved in R Global Environment and in the folder assigned as your Project or working directory. You will need two of the files for the following analyses; The first file contains the statistical parameters calculated for the entire population or the stratified population (named ´ObsStat´ ), and the second file contains the estimated values of the NCA metrics for each individual (named ´ncaOutput´). The latter will be used for the paired t-testing in the end of this tutorial. Also, a set of population histograms are created to present the population distribution of four parameters (AUClast, AUCINF\_obs, Cmax, and Tmax) for the estimated population means compared to the observed population. These histograms might be used to determine the general performance of the population pharmacokinetic model that is also embedded in the ncappc package. See the original paper by [Acharya et al (2016)](https://pdf.sciencedirectassets.com/271322/1-s2.0-S0169260716X0003X/1-s2.0-S0169260715300262/main.pdf?X-Amz-Security-Token=IQoJb3JpZ2luX2VjEIf%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaCXVzLWVhc3QtMSJIMEYCIQDtLQH0KkbjfzJjQIO1mneXuLdoHVQQXfy15SWkRvA8cQIhAKuZU56fjqx%2Bc3n7rejCLRgXiF2O4htKTL9fOXJuGkNYKvoDCHAQBBoMMDU5MDAzNTQ2ODY1IgyJs3QQ1kQWI9vOu6Uq1wNdDQoidnXU0WduIjYkp14GD4s%2BmKJ41ysrRQ6BXhMStQtU0YdbNTEIttkjx3OpsKeN8%2BV0PCaf0WYV%2B5u3iRaE8xpdcyiNQD3yabDRPmegZ2Or190hpZuKUI%2FDgll4OMT4zESJ5sboENsXKNCFQ5GTzSSco6tGOg%2BqyKsNa%2BAxNa2N5KfS%2BGzRkplBmvM7gNupuv7Qj5x8Pql%2F0I%2Br4Kvl3745%2FxywBDffnK%2F8MRex%2FdNwUY6iHZ0pCqavZFMar0gEROSaL%2BxjuokZ8p63VyqytxZkxLz574chJEGh4TmUAv9UTPPTTyiSZb%2BOhQzSc%2FQhL%2FnZo7We4ClhDHiiCpxShW7bXs1ycqO8%2BC5Tqsv10h6WEBH3UWIfJEKCoIw5lBG2HdVt0IdQUhVymxDK%2FPUvTzr%2BGdic9Vzjgnhf6K4g1gXk0ZtT3i6mdsiaLKPtxVUPYYgy%2FqmMLgWppyafPa2rspvEY5Hb2qKS%2F1daXBGIQ22CSSz7cZeJ6UfL4To4CLkiomID1uMJyvTZHK0etrUzgVuwUIvDlftrffSBeuVJ%2FFXNmtxqw93ram4KB3aGQYIKNYY2TKPl0tGLRfqM5JEaasHOR3HAnqBceGSKW7pfCPbZZC0KF68wjJjmjQY6pAGsb3rikmxvEGk%2FqZtyKQ%2F%2FsfPuP4xvDhjMevUBLBFLwIcst%2BGh6lUDXrjIMThJapjHuLLir%2FH23R29qwuTsusuA9o5u9inXpzhO4%2Bo0QPi6kxzkgW6dODoCI%2BL7uR5qNzPKK3Wvq2hKObAPqbfXrxv9yp%2B3pyRzeY%2BckeSo3TWzzrYlbLaPhegfv8r43ynBcYRf1tE0KFAFvZcPSbFppOKomPiug%3D%3D&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20211215T071631Z&X-Amz-SignedHeaders=host&X-Amz-Expires=300&X-Amz-Credential=ASIAQ3PHCVTYUD7NCK55%2F20211215%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Signature=fb707e3292b203a604b5f51b4e0ef186fe67be53d8658481f4742da1598ecbd2&hash=2bd1729c462f48f25cde3c9a515ea822bad736a59020d683fe2b89def6a09d33&host=68042c943591013ac2b2430a89b270f6af2c76d8dfd086a07176afe7c76c2c61&pii=S0169260715300262&tid=spdf-a69ee59f-66d4-455f-ab1d-be3d10a2eff8&sid=e4aaabb825f6e2468c6808e68937261aeb78gxrqb&type=client) for further details.

Following NCA it is important to assess the performance og the analysis. Rsq in the ncaOutput file is the regression coefficient and describes the performance of our regression line used to estimate the elimination rate constant. The closer to one, the better fit. An unacceptable regression coefficient may be due to outliers or missing data in the individuals data, and requires an evaluation of the data before moving on.

The created files include much more data than needed. To ease the following analyses and to keep the overview, we create a new dataframe, PK\_analysis, only containing the parameters needed.

# Create a new dataframe containing only the data needed (Cmax, AUClast, HL\_lambda\_z (terminal half-life), and Cl\_obs).  
PK\_analysis <- ncaOutput %>%  
 select(ID, IND, Tmax, Cmax, Cl\_obs, HL\_Lambda\_z, AUClast)  
  
# Change the unit of Cl\_obs from ml/h to l/h, to increase readability of the coming Table 2.   
PK\_analysis$Cl\_obs <- (PK\_analysis$Cl\_obs)/1000

# Section III

## Descriptive statistics for Table 2 (Non-compartmental PK analysis stratified by DDI)

In this section, we will work with the parameters calculated in the non-compartmental analyses. These results will make up a Table 2. In Table 2, we aim to present medians with interquartile range (IQR) for the five variables of interest: Cmax, AUC (AUClast), elimination half-life (HL\_lambda\_z), clearance (CL\_obs), and Tmax. Further, we calculate geometric mean ratio (GMR) with 95% CI. GMR is not relevant for Tmax,as this is a categorical variable. To ease the work in the following, we reshape data from long to wide format. This means, that each individual will have only one row and the parameters with and without induction (e.g. Cmax\_0, Cmax\_1, Cl\_obs\_0, Cl\_obs\_1, etc.) will occupy individual columns.

# Reshape data from long format to wide format.  
PK\_analysis\_wide <- dcast(melt(PK\_analysis, id.vars=c("ID", "IND")), ID ~ variable+IND)

First, we are calculating the ratio between induction and no induction for the individual variables.

# Add a new column with ratio.  
PK\_analysis\_wide$Cmax\_ratio <- with(PK\_analysis\_wide, Cmax\_1/Cmax\_0)  
PK\_analysis\_wide$Cl\_obs\_ratio <- with(PK\_analysis\_wide, Cl\_obs\_1/Cl\_obs\_0)  
PK\_analysis\_wide$HL\_Lambda\_z\_ratio <- with(PK\_analysis\_wide, HL\_Lambda\_z\_1/HL\_Lambda\_z\_0)  
PK\_analysis\_wide$AUClast\_ratio <- with(PK\_analysis\_wide, AUClast\_1/AUClast\_0)

Second, we are calculating the geometric mean including 95% CI for the ratios of the individual variables. For example, this can be calculated by ci.mean(Cmax\_ratio, statistic="geometric"). To ease the work, we apply a function to calculate the results continuously and at the same time order the confidence intervals properly for table layout. One of the main reasons for creating the entire table in R is to avoid multiple copying/pasting and hence reducing the risk of errors.

# Define the function for calculating geometric mean ratio and 95% CI.  
geo <- function(x) {  
 aggr <- ci.mean(x, statistic = "geometric")  
 paste0(round(aggr$geomean, 2), " (", round(aggr$lower, 2), "-", round(aggr$upper, 2), ")")  
}  
# Calculate geometric mean and 95% CI for the individual parameters.   
Cmax\_GMR <- geo(PK\_analysis\_wide$Cmax\_ratio)  
Cl\_GMR <- geo(PK\_analysis\_wide$Cl\_obs\_ratio)  
HL\_GMR <- geo(PK\_analysis\_wide$HL\_Lambda\_z\_ratio)  
AUC\_GMR <- geo(PK\_analysis\_wide$AUClast\_ratio)  
Tmax\_GMR <- "NA"  
# Assemble results in one column for Table 2.  
results\_GMR <- rbind(AUC\_GMR, Cmax\_GMR, HL\_GMR, Cl\_GMR, Tmax\_GMR)

# Define the function for calculating median.  
medians <- function(x) {  
 med <- median(x)  
 iqr\_low <- quantile(x, 0.25)  
 iqr\_up <- quantile(x, 0.75)  
 paste0(round(med, 2), " (", round(iqr\_low, 2), "-", round(iqr\_up, 2), ")")  
}  
# Calculate median for the individual parameters with and without inducer.  
Cmax\_0\_M <- medians(PK\_analysis\_wide$Cmax\_0)  
Cmax\_1\_M <- medians(PK\_analysis\_wide$Cmax\_1)  
Cl\_0\_M <- medians(PK\_analysis\_wide$Cl\_obs\_0)  
Cl\_1\_M <- medians(PK\_analysis\_wide$Cl\_obs\_1)  
HL\_0\_M <- medians(PK\_analysis\_wide$HL\_Lambda\_z\_0)  
HL\_1\_M <- medians(PK\_analysis\_wide$HL\_Lambda\_z\_1)  
AUC\_0\_M <- medians(PK\_analysis\_wide$AUClast\_0)  
AUC\_1\_M <- medians(PK\_analysis\_wide$AUClast\_1)  
Tmax\_0\_M <- medians(PK\_analysis\_wide$Tmax\_0)  
Tmax\_1\_M <- medians(PK\_analysis\_wide$Tmax\_1)  
  
# Assemble results in two columns (with and without induction) for Table 2.  
results\_IND <- rbind(AUC\_1\_M, Cmax\_1\_M, HL\_1\_M, Cl\_1\_M, Tmax\_1\_M)  
results\_noIND <- rbind(AUC\_0\_M, Cmax\_0\_M, HL\_0\_M, Cl\_0\_M, Tmax\_0\_M)

By using the results created above, we can generate an entire Table 2.

# Assemble Table 2  
Table2 <- matrix(nrow=5, ncol=0)  
Table2 <- cbind(results\_noIND, results\_IND, results\_GMR, Table2)  
colnames1 <- c("WITHOUT INDUCER (Median (IQR))", "WITH INDUCER (Median (IQR))", "GMR (95% CI)")  
Table2 <- rbind(colnames1, Table2)  
  
rownames1 <- c("DRUG", "Midazolam", "", "", "", "")  
rownames2 <- c("PARAMETER", "AUC 0-last (ng\*h ml^-1)", "Cmax (ng ml^-1)", "T1/2 (h)", "CL/F (L h^-1)", "Tmax (h)")  
Table2 <- cbind(rownames1, rownames2, Table2)

Tmax is a categorical variable and we can not apply GMR. Instead, we can use a paired Wilcoxon rank-sum test to compare data with and without induction. This equivalent to the students t-test used in Section VI, though on a different type of data.

# Test Tmax with paired Wilcoxon rank-sum test.   
Test\_tmax <- wilcox.test(Tmax ~ IND, data = ncaOutput, paired = TRUE)

## Warning in wilcox.test.default(x = c(2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2), :  
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(x = c(2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2), :  
## cannot compute exact p-value with zeroes

Test\_tmax

##   
## Wilcoxon signed rank test with continuity correction  
##   
## data: Tmax by IND  
## V = 36, p-value = 0.005962  
## alternative hypothesis: true location shift is not equal to 0

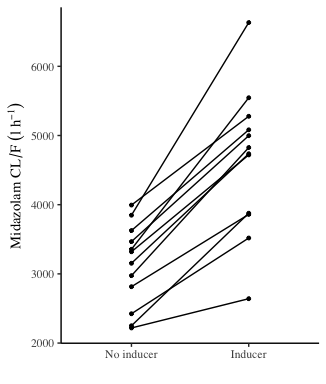
# Section IV

## Plotting individual spaghettiplots for Figure 2

A spaghetti plot is used to visualize and compare induction and no induction for each individual. Also, here is an opportunity to label subjects with outliers such as genetic variants.

# Recode the induction variable from numeric to categorical.  
Data\_fig2 <- ncaOutput  
Data\_fig2$IND <- factor(ncaOutput$IND, levels= c("0", "1"), labels=c("No inducer", "Inducer"))  
  
# Change the unit of Cl\_obs from ml/h to l/h, to increase readability of the coming Table 2.   
Data\_fig2$Cl\_obs <- (Data\_fig2$Cl\_obs)/1000

# Create a spaghettiplot for the individual clearances.   
ggplot(data = Data\_fig2, aes(x = IND, y = Cl\_obs)) +  
 geom\_point(size=1) +  
 geom\_line(aes(group = ID), size=0.5) +   
 theme\_classic() +  
 labs(x = "", y = expression(Midazolam~CL/F~(l~h^-1))) +  
 scale\_y\_continuous(position = "left", breaks = scales ::pretty\_breaks(n=4)) +  
 guides(color=none) +  
 theme(text=element\_text(family="Times"))



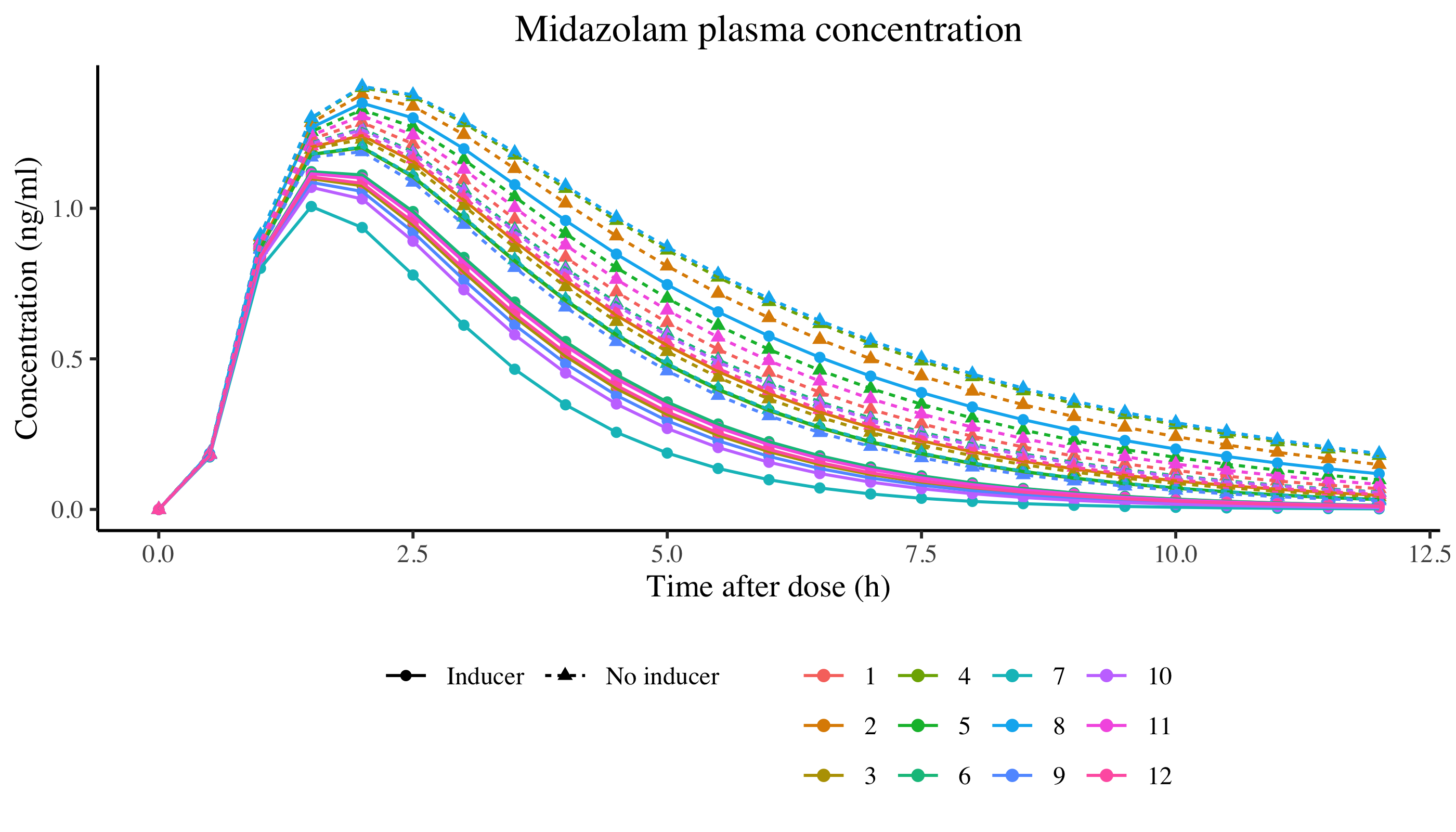
# Section V

## Concentration-time curve for individual data

Below is a concentration-time curve of each individual study subject. First, the variable ID is changed from a continuous numerical scale to a categorical factor. Otherwise, the legend in ggplot will be continuous instead of categorical:

# Recode ID from numeric to categorical.  
RawPK$ID <- factor(RawPK$ID)  
  
# Recode IND from numeric to character.  
RawPK$IND <- recode(RawPK$IND, "0" = "No inducer", "1" = "Inducer")

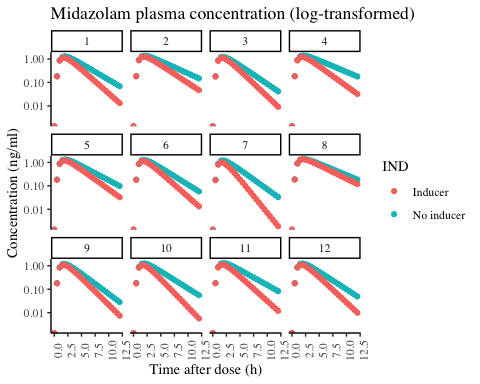
# Plot midazolam plasma concentrations and time for the individual subjects.   
ggplot() +  
 geom\_point(data = RawPK, aes(x = TAD, y = DV, color = ID, shape = IND)) +  
 geom\_line(data = RawPK, aes(x = TAD, y = DV, group=interaction(ID, IND), color = ID, linetype = IND)) +  
 labs(x ="Time after dose (h)", y = "Concentration (ng/ml)", title ="Midazolam plasma concentration", color = "") +  
 theme\_classic() +  
 theme(plot.title = element\_text(hjust = 0.5), legend.position = "bottom", legend.title = element\_blank(), text=element\_text(family="Times"))



The function facetwrap is used to make a concentration-time plot for each individual study subject for a better visibility. The warning message “Transformation introduced infinite values in continuous y-axis” is due to the concentration of 0 ng/ml at 0 hours. Log(0) is not defined, but is approaching negative infinity.

# Individual plots of midazolam plasma concentration and time.  
ggplot(data = RawPK, aes(x = TAD, y = DV, color =IND)) +  
 geom\_point() +  
 facet\_wrap(~ID) +  
 scale\_y\_log10() +  
 labs(x ="Time after dose (h)", y = "Concentration (ng/ml)", title ="Midazolam plasma concentration (log-transformed)") +   
 theme(axis.text.x = element\_text(angle = 90), text=element\_text(family="Times"))

## Warning: Transformation introduced infinite values in continuous y-axis



# Section VI

## Paired t-testing

For the supplementary, we calculate mean +/- standard deviation (SD) and perform paired t-testing. Here, we only show examples for AUC\_last. Similar calculations should be done for the other variables.

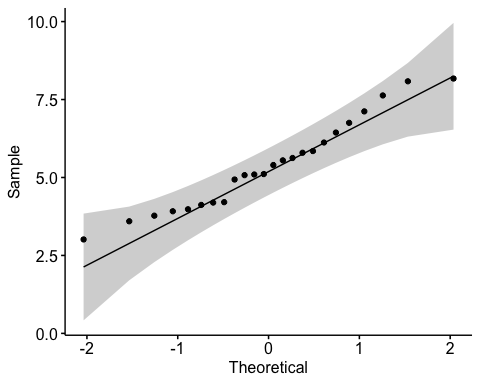
# Calculate mean AUC\_last +/- standard deviation (SD) for the two groups (no inducer/inducer).  
PK\_analysis %>%   
 group\_by(IND) %>%   
 summarise(mean(AUClast), SD = sd(AUClast))

## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 2 x 3  
## IND `mean(AUClast)` SD  
## <dbl> <dbl> <dbl>  
## 1 0 6.33 1.12  
## 2 1 4.47 1.09

The paired t-test assumes that data are normally distributed. A number of methods can be applied to check for normality, including Q-Q plot.

# Q-Q plot to check for normality.   
ggqqplot(PK\_analysis$AUClast)

 Normality is supported by the Q-Q plot where the data points approximately follow the reference line.

If data are not normally distributed, an alternative method is to log-transform data before testing. This should only be done if necessary, as it complicates the interpretation of data. The log-transformed data are then used in the paired t-test.

PK\_analysis$logAUClast = log(PK\_analysis$AUClast)

In the following the paired t-test is performed.

# Define the function for paired t.test to compare groups of induction and no induction.  
Ttest <- function(x) {  
 t.test(x ~ IND, data = PK\_analysis, paired = TRUE)  
}  
# Compute the paired t-test for AUC\_last.  
p\_AUClast <- Ttest(PK\_analysis$AUClast)  
p\_AUClast

##   
## Paired t-test  
##   
## data: x by IND  
## t = 11.955, df = 11, p-value = 0.0000001209  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## 1.514181 2.197555  
## sample estimates:  
## mean of the differences   
## 1.855868

The computed output gives a p-value (<0.05, meaning that we can reject the null-hypothesis), a mean difference between pairs (sample estimates), and the adjacent 95% CI. Even though we might interpret this p-value as a statistically significant result, we should aim to determine whether the result is “clinically significant” based on a clinical interpretation of the size of the difference. The summary statistics and paired t-test may not be relevant to include in the main manuscript of your article, however, the results should be included in the supplementary as they may be highly valuable for future researchers (for example when doing power calculations).

# Section VII

## Sample size calculation

Sample size calculation is an important aspect of experiments and should be conducted before the initiation of the study. Sample size is often referred to as power analysis. Three parameters are of interest when calculating sample size:  
- **Effect size (d)**: The strength of a relationship between two variables in a population. In this case, where we use an independent samples t-test, Cohen’s d is an appropriate measure of effect size. It is calculated where .  
- **Significance level (sig.level)**: Probability of finding an effect that is not there (type I error).  
- **Power (power)**: Probability of finding an effect that is there (1 - type II error)

**Sample size (n)** is the unknown variable and is the number of subjects to be included in the study to fulfill the abovementioned conditions.

The pwr package in R implements power analysis for multiple different tests. The power analysis should be based on the primary outcome, and hence the statistical analysis planned for this. In this pharmacokinetic trial, the primary outcome is a difference in AUC of midazolam with and without induction. We use a paired t-test to assess this. Therefore, the power calculation is based on a paired t-test, as demonstrated here below. For further functions please check out the [link](https://cran.r-project.org/web/packages/pwr/pwr.pdf).

The following example is based on mean and standard deviation (SD) calculated in **Supplementary, Section VI**.  
- No induction (here named mean\_1 ± SD\_1) = 6.33 ± 1.12  
- Induction (here named mean\_2 ± SD\_2) = 4.47 ± 1.09  
In the previously calculated mean there is a 40% difference between no induction and induction. If the study aims to detect another difference (e.g. 20%), mean\_2 is calculated as 20% smaller than mean\_1.

# install.packages("pwr")  
library(pwr)

# SD\_pooled = sqrt((SD\_1^2 + SD\_2^2)/2) = sqrt((1.12^2 + 1.09^2)⁄2)) = 1.11  
# Cohen's d = (mean\_2-mean\_1)/SD\_pooled = (4.47 - 6.33)/1.11 = -1.68  
  
pwr.t.test(n= NULL, d=-1.68, sig.level = 0.05, power =0.80,   
 type = c("paired"),  
 alternative = c("two.sided") )

##   
## Paired t test power calculation   
##   
## n = 5.006306  
## d = 1.68  
## sig.level = 0.05  
## power = 0.8  
## alternative = two.sided  
##   
## NOTE: n is number of \*pairs\*

This means that a total number of 5 pairs is needed for a study assessing induction of midazolam with a 80% power, significance level of 0.05, and an expected difference of 40% in mean AUC between no induction and induction. Remember to add additional subjects to account for drop out (often 20%).