Session 1 - R Basics

Ya-Feng Wen Fan Wang

2021-06-18

Contents

Objectives

- Project Management
  - Git and version control
  - Command Line
- RStudio and R markdown
- lubridate
- Loops/Functions
- ggplot2
- dplyr

Git and version control

- Install Git
  - Mac: use terminal, `git --version`. If Git is not found, you will be asked to install it.
  - Windows: search *Git for Windows*. Download the most recent build.
    - Select the *nano* as default editor. If familiar with *vi* or *vim*, you can use these.
    - Select *Git and optional Unix tools from the Windows Command Prompt* so you can use Git from within RStudio
    - Once Git/Git Bash is installed, in Tools -> Global Options -> Terminal -> select *Git Bash*.
- Connect RStudio with GitHub

```
# Bash, run in Terminal
# Not run, modify accordingly
git config --global user.name "Your Name"
git config --global user.mail "your@email.com"
```

- Tools -> Global Options -> Git/SVN, enter a path for the Git executable. (Default on the Windows: *C:/Program File/Git/bin/git.exe*)
- Create *SSH RSA Key* by clicking the *Create RSA Key* button. This enable you to avoid entering password each time trying to access GitHub repository.
- Github setup
  - Create a repository on GitHub or GitHub Enterprise
- Initialize a Git directory
  - Initialize a Git directory and connect it to the upsteam repository
  - Set up a R project

```
# Bash, run in Terminal
# Not run, modify accordingly
pwd
mkdir directory
git init https://github.com/yafengwen/project.git
cd project
```

- Overview of Git

Main actions:

1. **clone** an existing GitHub Upstream Repository, including the entire Git structure: *Working Directory*, *Staging Area*, and *Local Repository*.
2. **pull** changes from the GitHub repo
3. **stage** (add) files
4. **commit** changes to the local repo
5. **push** changes to the GitHub repo
6. **branch** and **merge** to facilitate collaboration. See more details here

Some usefual functions:

```
# Bash, run in Terminal

# Compare files in the Working Directory with GitHub repo
git status

# Add a file to the Staging Area
```

Figure 1: Git Overview

```
git add new-file.txt
git status

# Commit the changes to the local repo
git commit -m "add a message"
git status

# keep track of all the changes
git log new-file.txt

# Push the changes to the upstream repo
git push

# Pull changes from the upstream repo to working direcotry
git pull
```

- Use Git and GitHub in RStudio
- Other software to facilitate the version control
  - GitHub Desktop
  - GitKraken

Command Line

- `pwd`: show full path of the working directory
- `ls`: list directory content
  - Arugment: -a (all), -l (long), -t (chronological order), -r (reverse order), -lart (combine all the agruments)
- `mkdir`, `rmdir`: make and remove a directory
- `cd`: change directory
  - `cd ~`, `cd ..`, `cd ../..`
- `mv`: move files or rename files
- `cp`: copy files
- `rm`: remove files
  - Argument: -r (recursive), -f (force), -rf (force to remove files recrusively)
- `less`, `more`: view files

Practice

1. Create the following folders using terminal: data, rds, figs
2. Create relevant `.R` and `.Rmd` files: download-data.R, wrangle-data.R, analysis.R, report.Rmd

RStudio and R markdown

We won't be covering this material this year, but you can see more content from last year, UMN Pharmacometrics Workshop 2020 Basics in R - I or refer to Reference 4-6.

Get things ready for the exercise!

```
# load the packages will be used in this session
library(tidyverse)
library(lubridate)
```

Practice

1. Read in data from data folder (dataSession1.csv)
2. Check the type of each column. Does the data type make sense? If not, convert to the correct type of object.

```
# load the data set will be used in the session

# if you know the column type ahead of time,
# you can convert column to the right type
data <- read_csv("../data/dataSession1.csv",
                 col_types = cols(DATE = col_date(format = "%m/%d/%Y"),
                                  TIME = col_time(format = "%H:%M:%S"),
                                  STUDY = col_factor(levels = c("1", "2", "3")),
                                  SEXF = col_factor(levels = c("0", "1"))))

# check the data type to see if this is what you expected
str(data)
```

```
# if you don't know, if can convert to the correct type later
dataRaw <- read_csv("../data/dataSession1.csv")

# convert data to the right type
dataRightType <- dataRaw %>%
  mutate(STUDY = factor(STUDY),
         SEXF = factor(SEXF))

str(dataRightType)
```

dplyr

- `filter()`: select rows
- `select()`: select columns
- `arrange()`: reorder rows
- `mutate()`: create new variables based on existing variables
- `summarise()`: summary values within a columns
  - `group_by()`: use with the main 5 functions

Syntax

1. The first argument: a data frame
2. The subsequent arugments: what to do with the data frame, using the variable names without quotes

3. The result: a new data frame

lubridate

- Parse date-times: `ymd()`, `ymd_hms()`, `dmy()`, `dmy_hms()`, `mdy()`
- Get and set date-times: `year()`, `month()`, `mday()`, `hour()`, `minute()`, `second()`
- `today()`
- `now()`

Practice

1. Convert DATE in dataRaw to date object

```
dataRaw %>%
  mutate(DATE = mdy(DATE))
```

2. Calculate the actual time since first dose (ATSFD)

Hint:

- Combine date and time into a new column using `ymd_hms()` and `paste()`
- Create a new column for the first datetime per subject
- Calculate the datetime difference using `difftime()`, then turn into duration using `duration()`, then calculate the hours from duration using `dhours()`

```
# combine date and time into a new column
data <- data %>%
  mutate(DATETIME = ymd_hms(paste(data$DATE, data$TIME))) %>%
  relocate(DATETIME, .after = TIME)

# calculate the time difference
data <- data %>%
  group_by(ID) %>%
  mutate(firstDateTime = min(DATETIME)) %>%
  mutate(ATSFD = as.duration(difftime(DATETIME,firstDateTime))/dhours(1)) %>%
  select(-firstDateTime) %>%
  ungroup() %>%
  relocate(ATSFD, .after = DATETIME)
```

ggplot2

The ggplot2 official website contains a great deal of information and resources worth exploring! (Many content presented here is based on previous material prepared by Ashwin!)

The Grammar of graphics

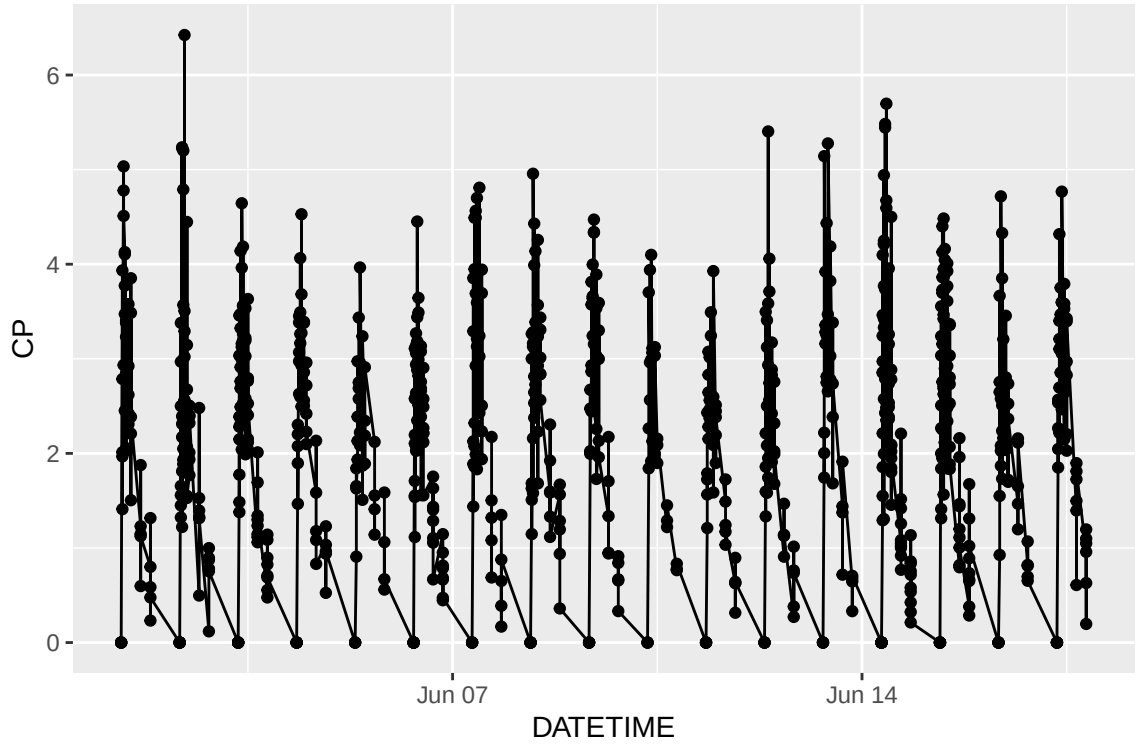
1. Data: very important!
2. Mapping: aesthetic and facet mapping, allow dataset to be understood by the graphic system
3. Statistics: transform input variables to displayed values
4. Scales: consider type of variable (categorical->color, numeric->position)
5. Geometries: plot type, use cheatsheet as the guide
6. Facets: small multiples
7. Coordinates
8. Theme

Practice

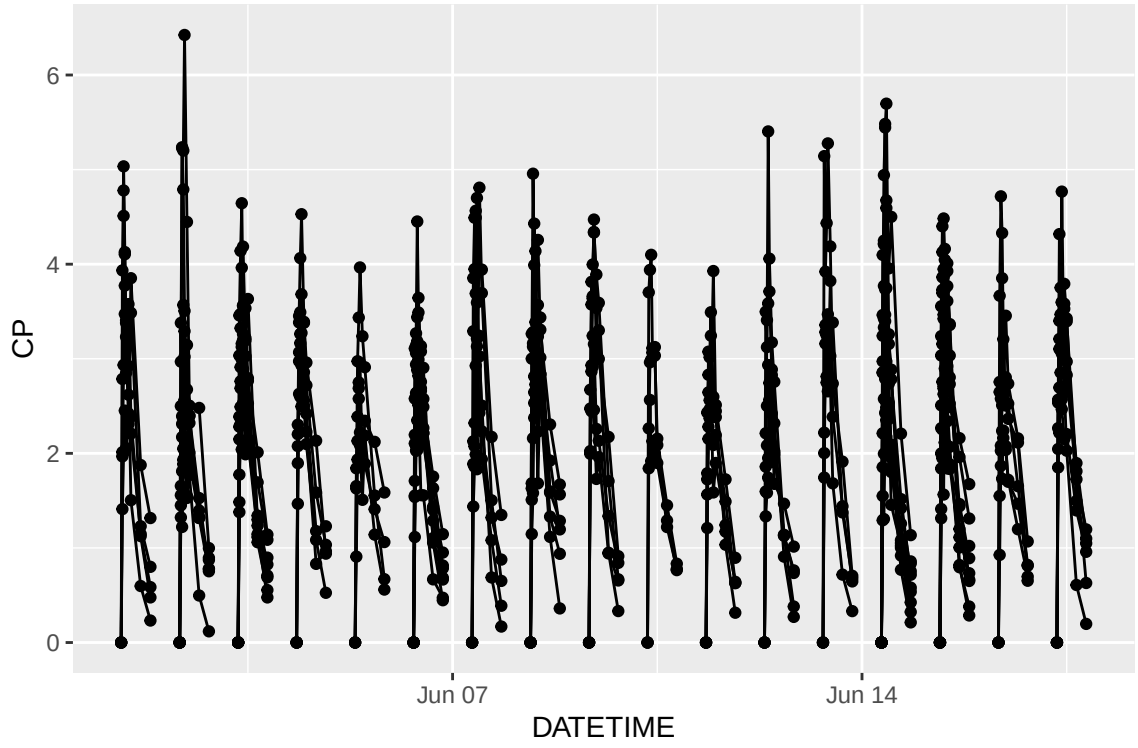
1. Plot the individual concentration-time curve
    - Every subject on the same graph
    - Each subject on a different graph
    - Plot subjects whose weight is above and below the mean weight with 2 different graphs
    - add a trend line using `geom_smooth()` with the default (loess method) and with standard deviation (cab also use `geom_errorbar()`)
    - Plot mean concentration by gender

- Every subject on the same graph

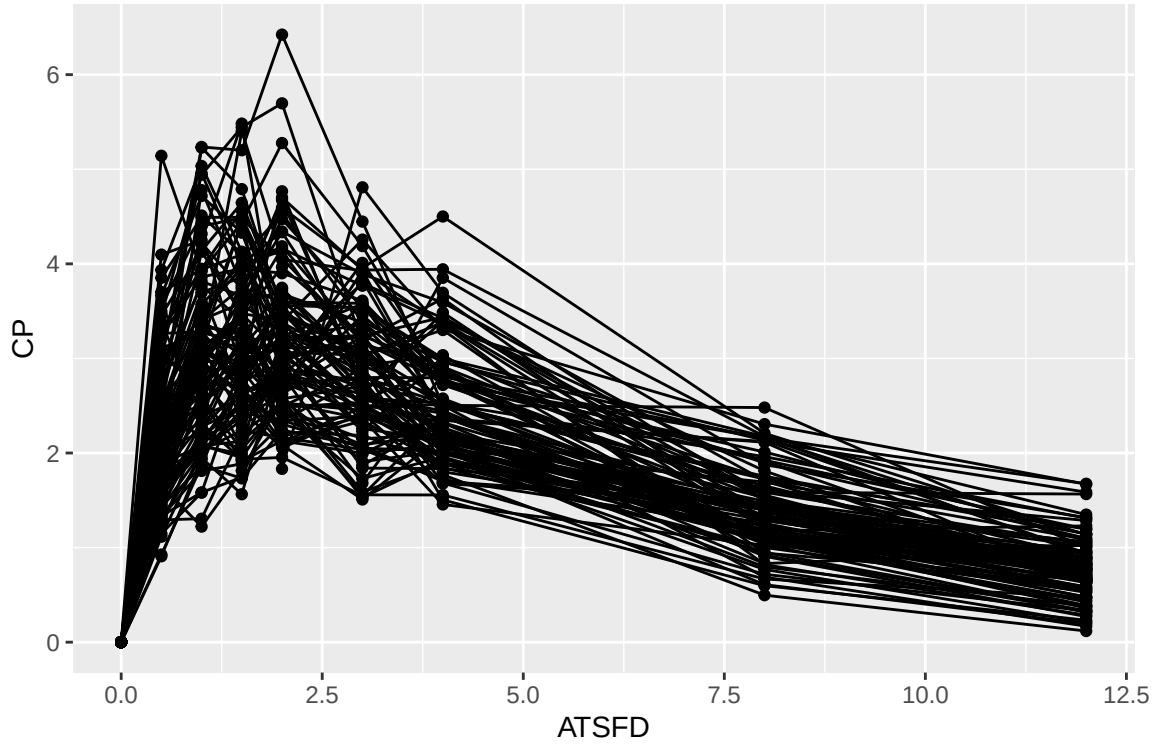
```
# doesn't look right...
ggplot(data, aes(DATETIME, CP))+
  geom_point()+
  geom_line()
```



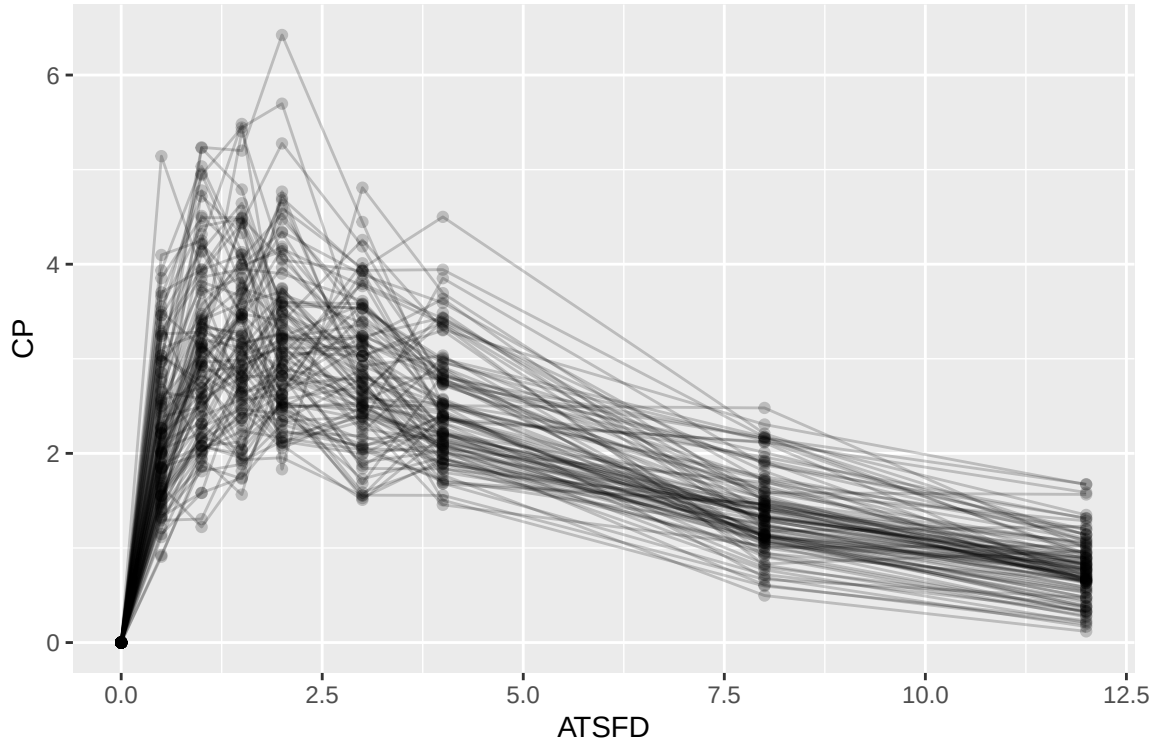

```
# better, but subjects start at different time
ggplot(data, aes(DATETIME, CP, group = ID))+
  geom_point()+
  geom_line()
```



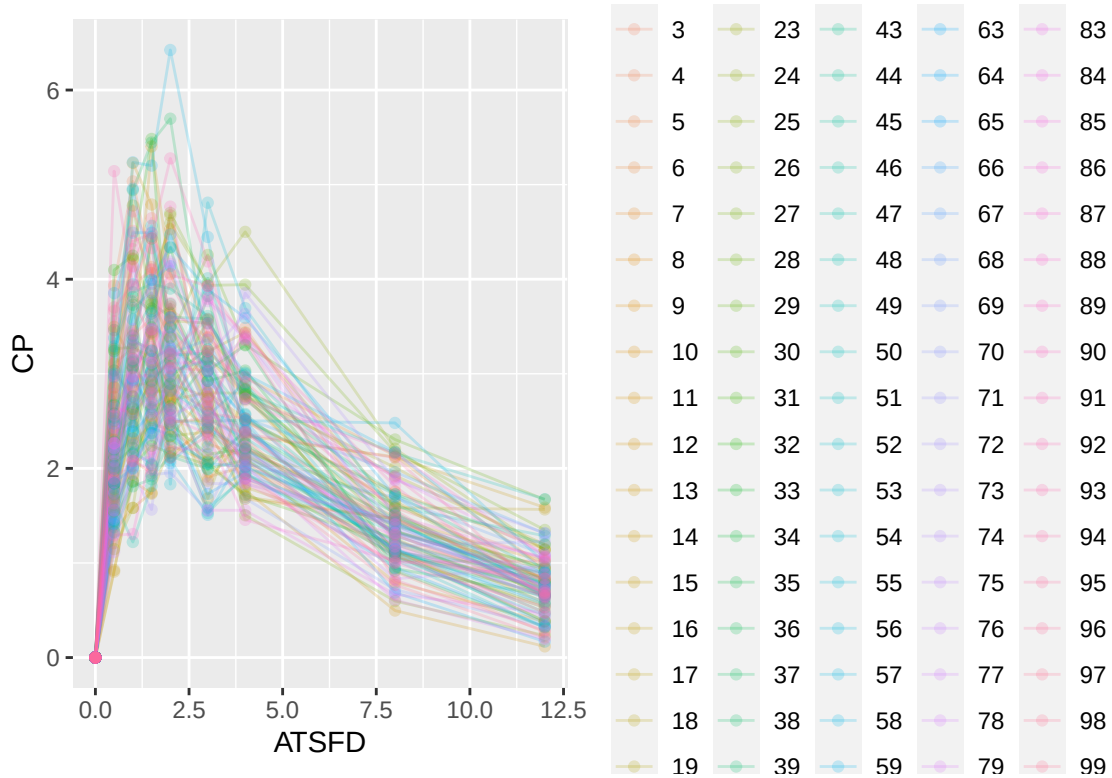
```
# looks something acceptable
ggplot(data, aes(ATSFD, CP, group = ID)) +
  geom_line() +
  geom_point()
```



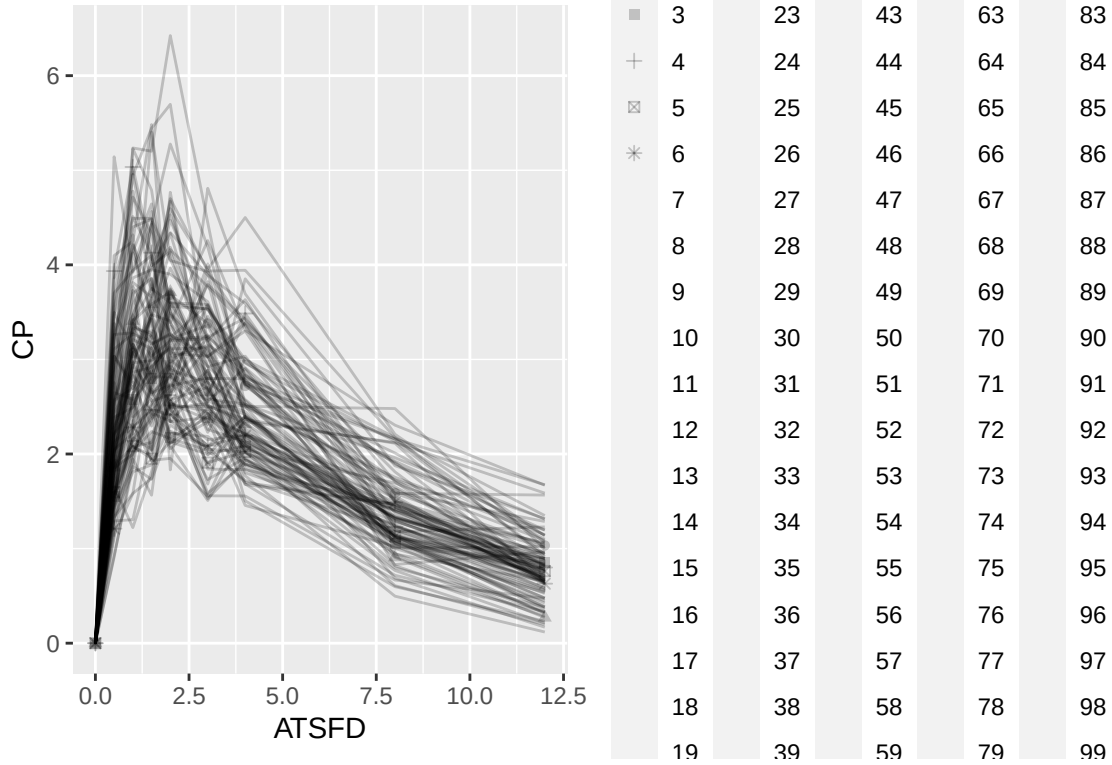

```
# make it prettier
ggplot(data, aes(ATSFD, CP, group = ID)) +
  geom_line(alpha=0.2) + # alpha for  transparency
  geom_point(alpha=0.2)
```



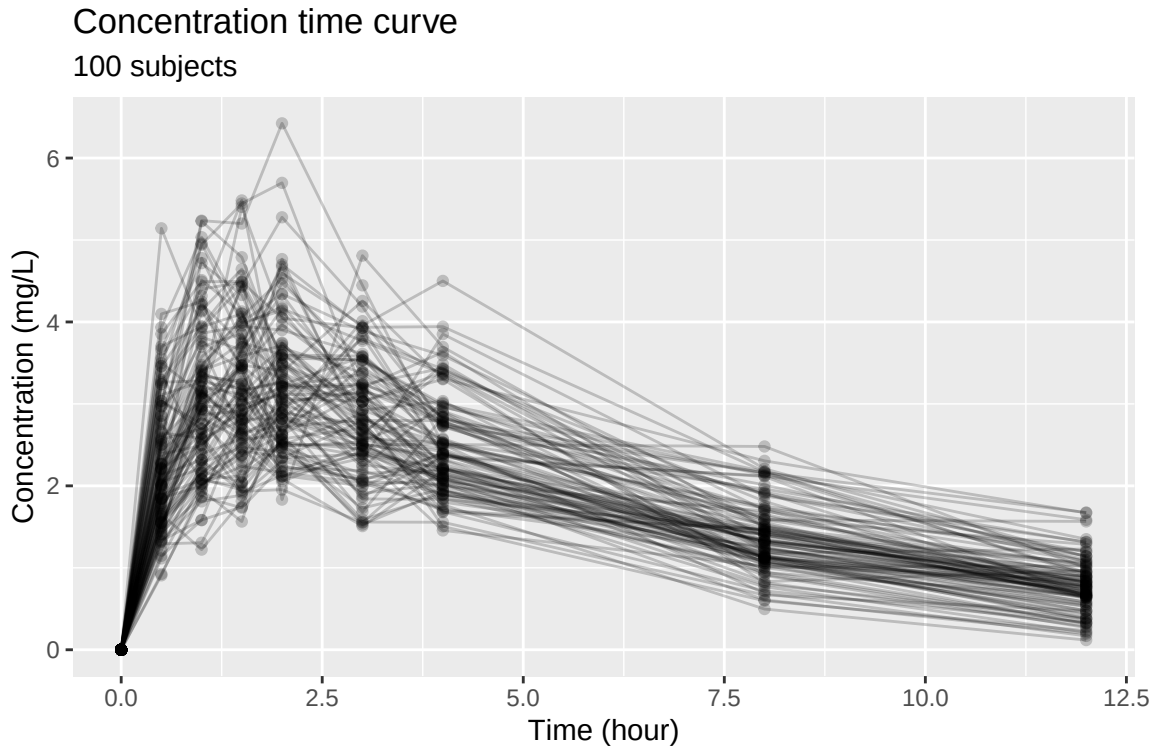

```
# maybe try some color or shape?
# probably not a good idea
ggplot(data, aes(ATSFD, CP, color = factor(ID))) +
  geom_line(alpha=0.2) +
  geom_point(alpha=0.2)
```




```
ggplot(data, aes(ATSFD, CP, shape = factor(ID))) +
  geom_line(alpha=0.2) +
  geom_point(alpha=0.2)
```

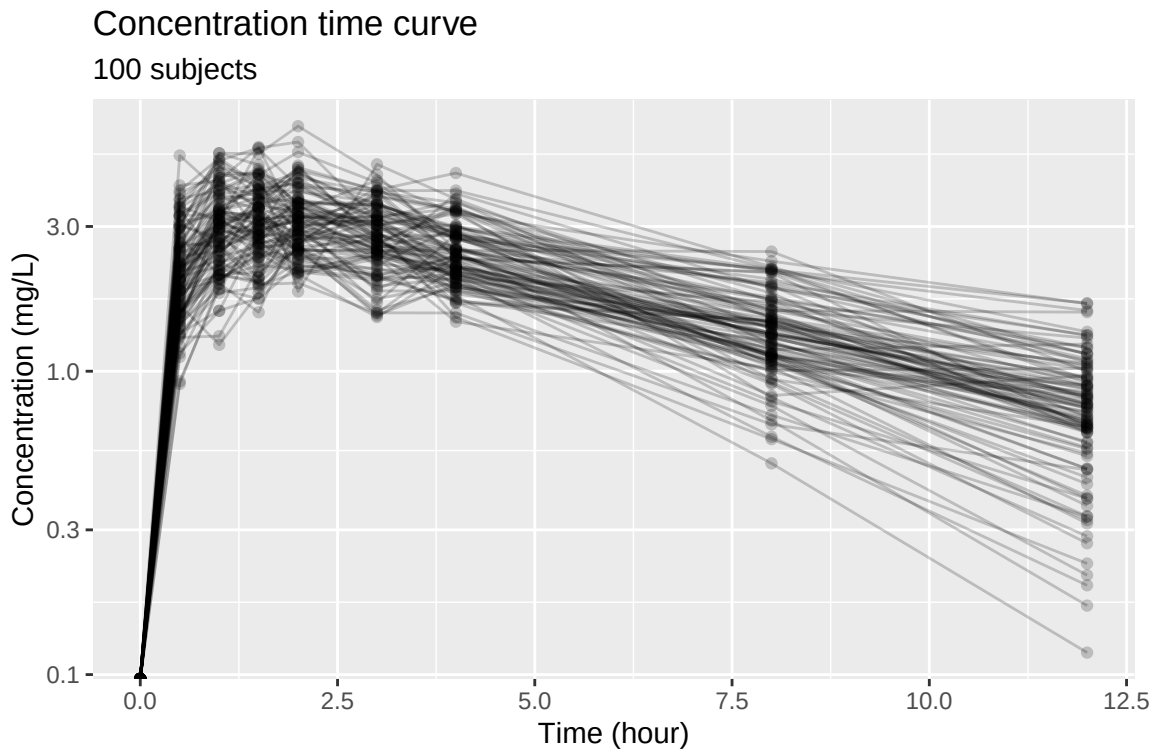


```
# the axis is not informative to many people
ggplot(data, aes(ATSFD, CP, group = ID)) +
  geom_line(alpha=0.2) +
  geom_point(alpha=0.2) +
  labs(title = "Concentration time curve",
       subtitle = "100 subjects",
       x = "Time (hour)",
       y = "Concentration (mg/L)")
```

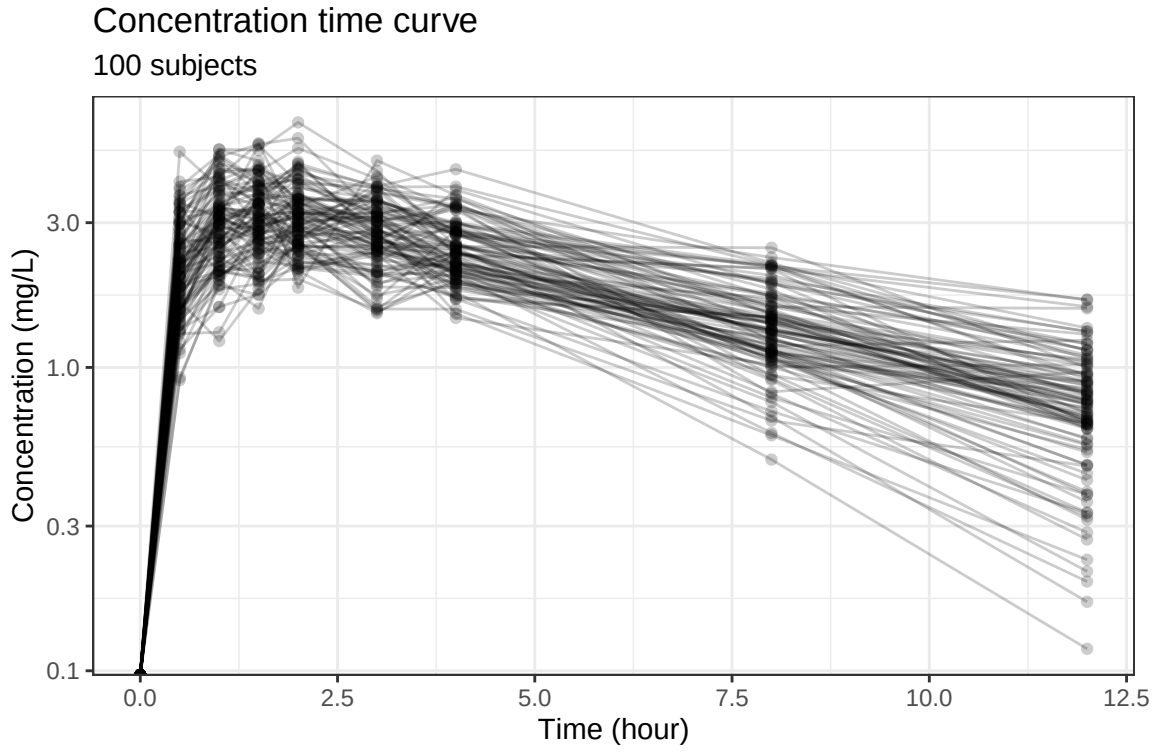



```
# we can save this as a base plot then do further manipulation
concTime <- ggplot(data, aes(ATSFD, CP, group = ID)) +
  geom_line(alpha=0.2) +
  geom_point(alpha=0.2) +
  labs(title = "Concentration time curve", # format axis labels with `labs()`
       subtitle = "100 subjects",
       x = "Time (hour)",
       y = "Concentration (mg/L)")
```

```
# log scale
concTime + scale_y_log10()
```

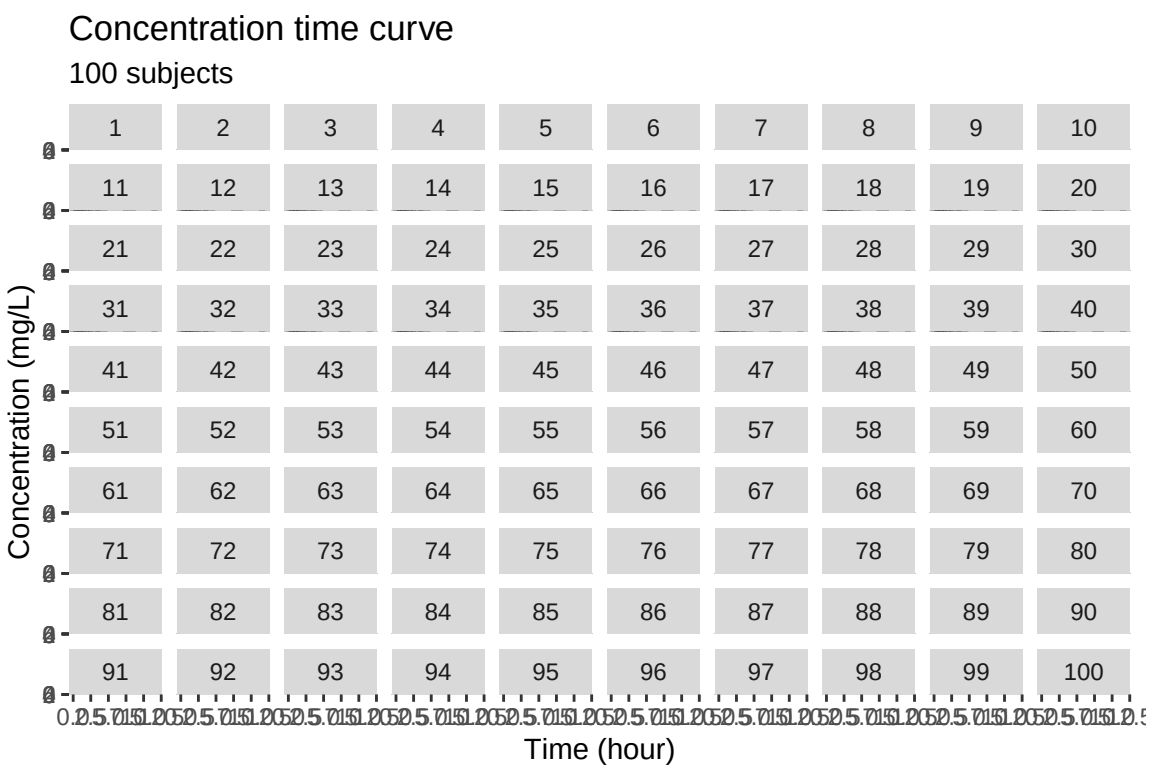



```
# many journal requires white background
concTime + scale_y_log10() + theme_bw()
```



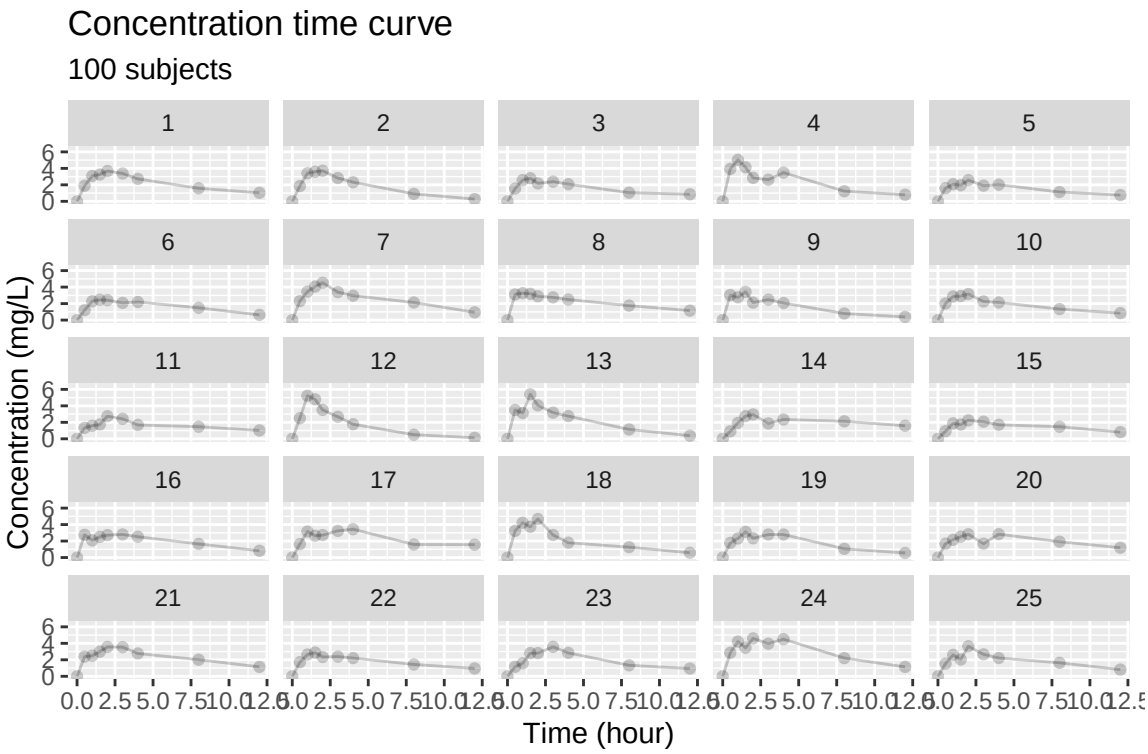
- Each subject on a different graph

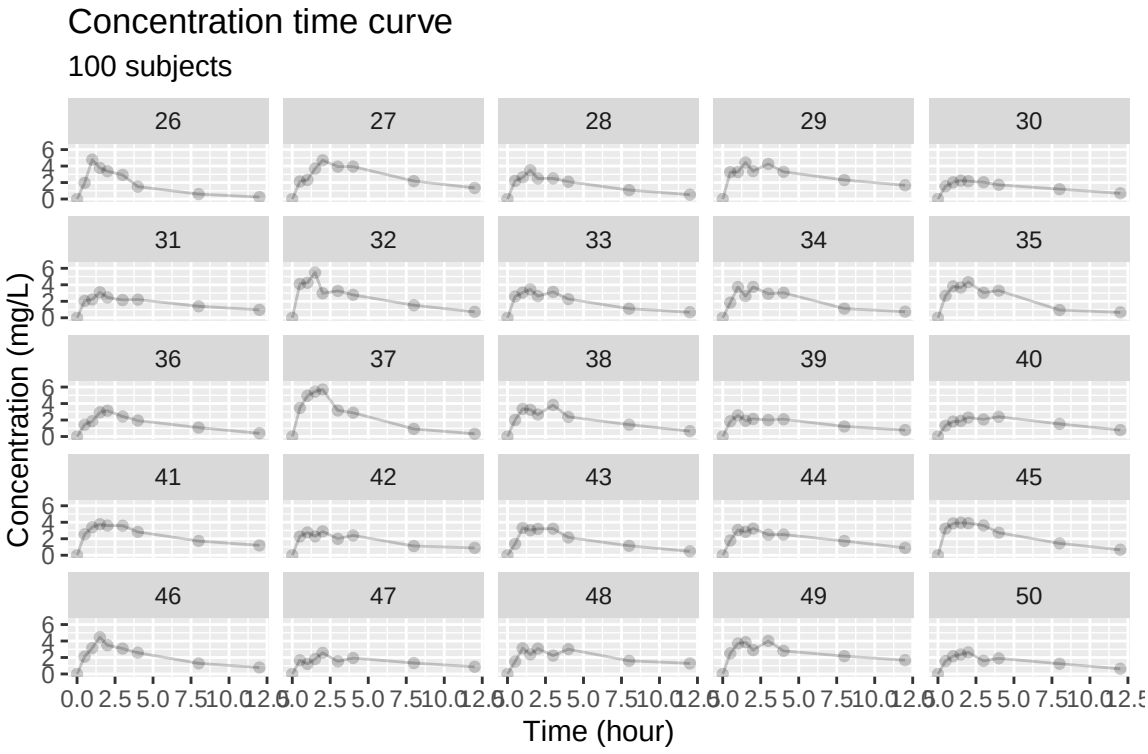
```
# plot each individuals
concTime + facet_wrap(~ID) # Cannot see anything
```

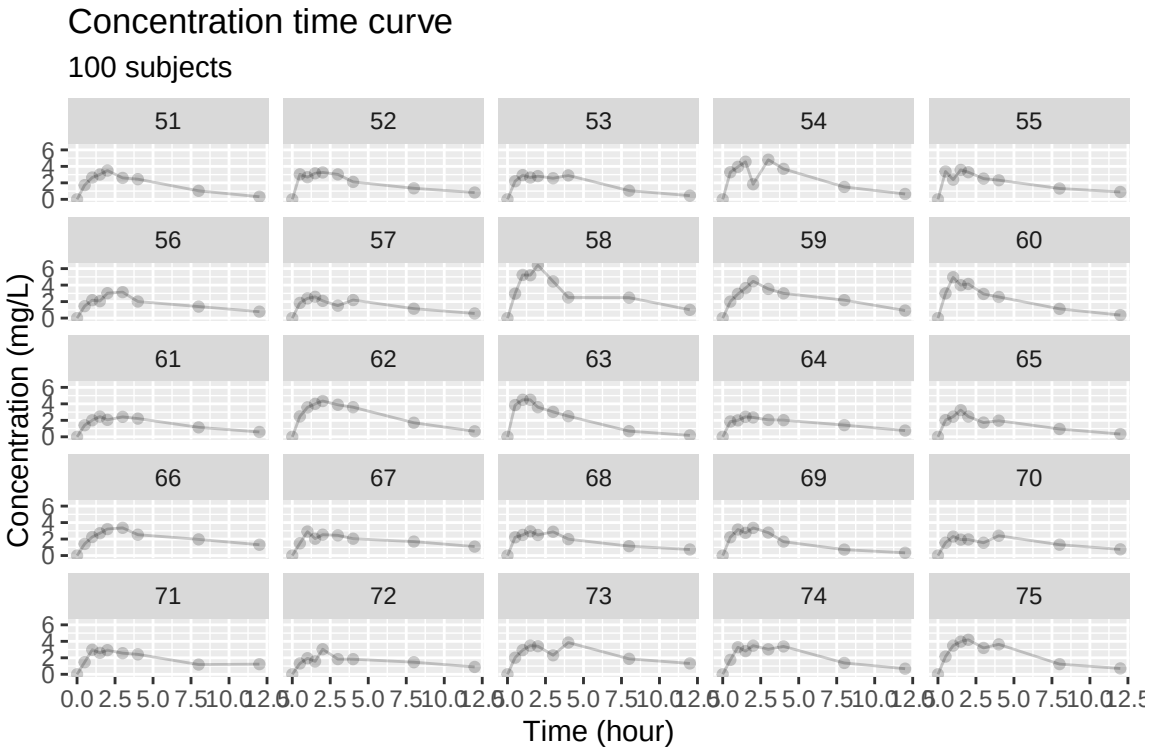


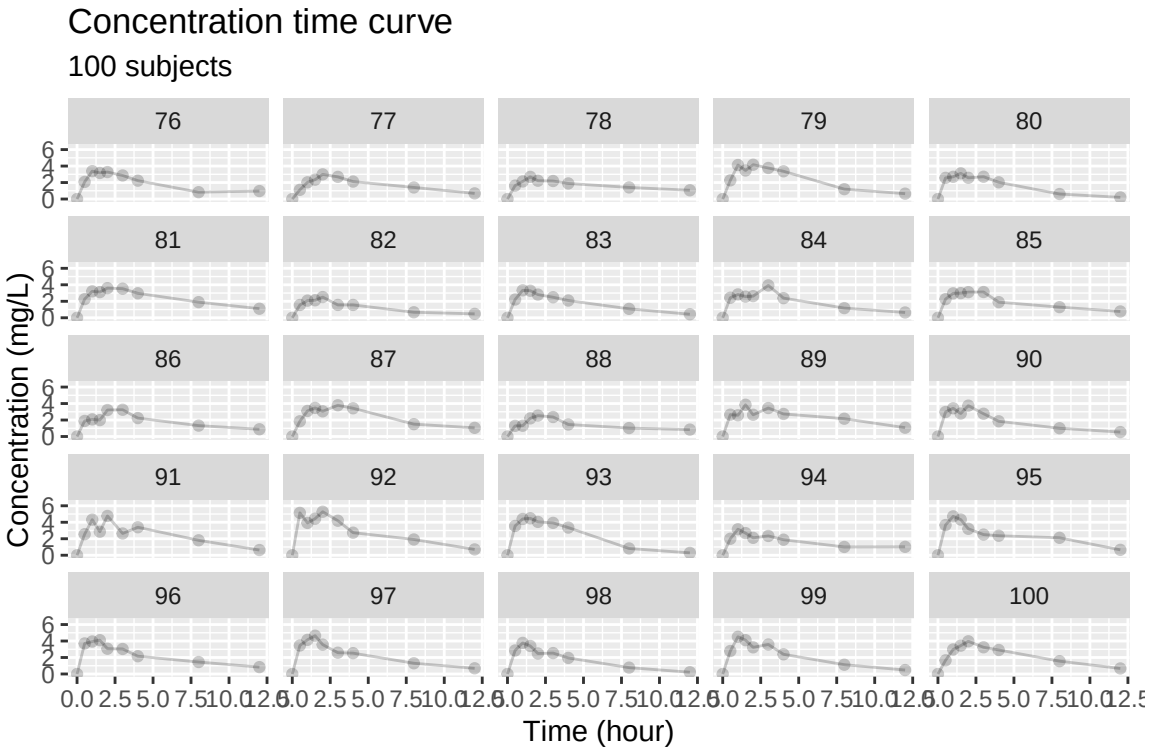

```
# potential solution: ggforce package
numPages = 4

for (i in seq_len(numPages)) {
  print(concTime +
          ggforce::facet_wrap_paginate(~ ID, ncol = 5, nrow = 5, page = i))
}
```


Concentration time curve
100 subjects
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
Concentration (mg/L)
Time (hour)

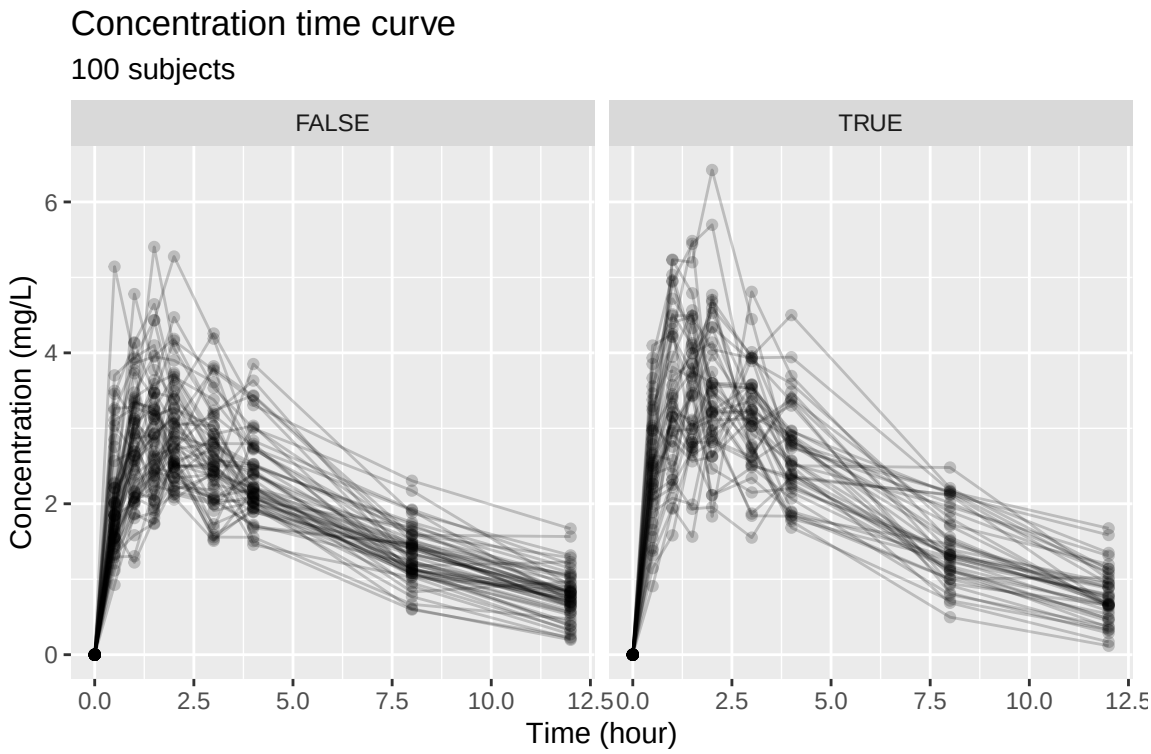

Concentration time curve
100 subjects
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
Concentration (mg/L)
Time (hour)





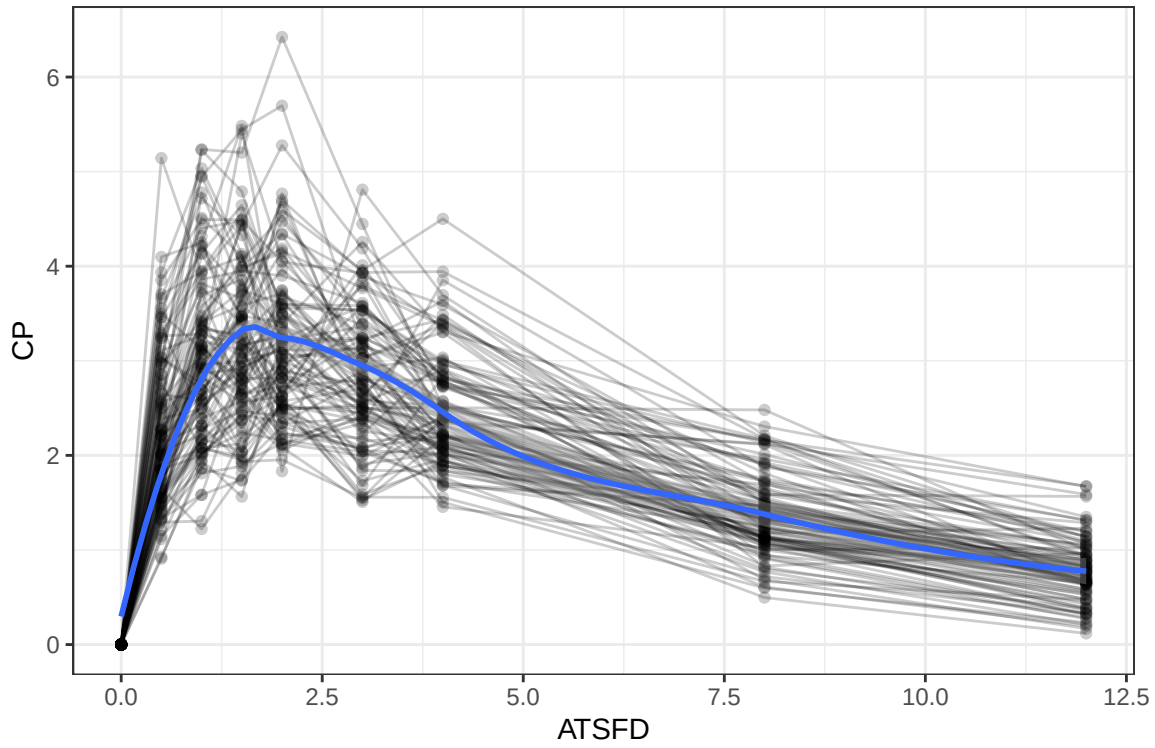
- Plot subjects whose weight is above and below the mean weight with 2 different graphs

```
concTime + facet_grid(~WT < mean(WT))
```



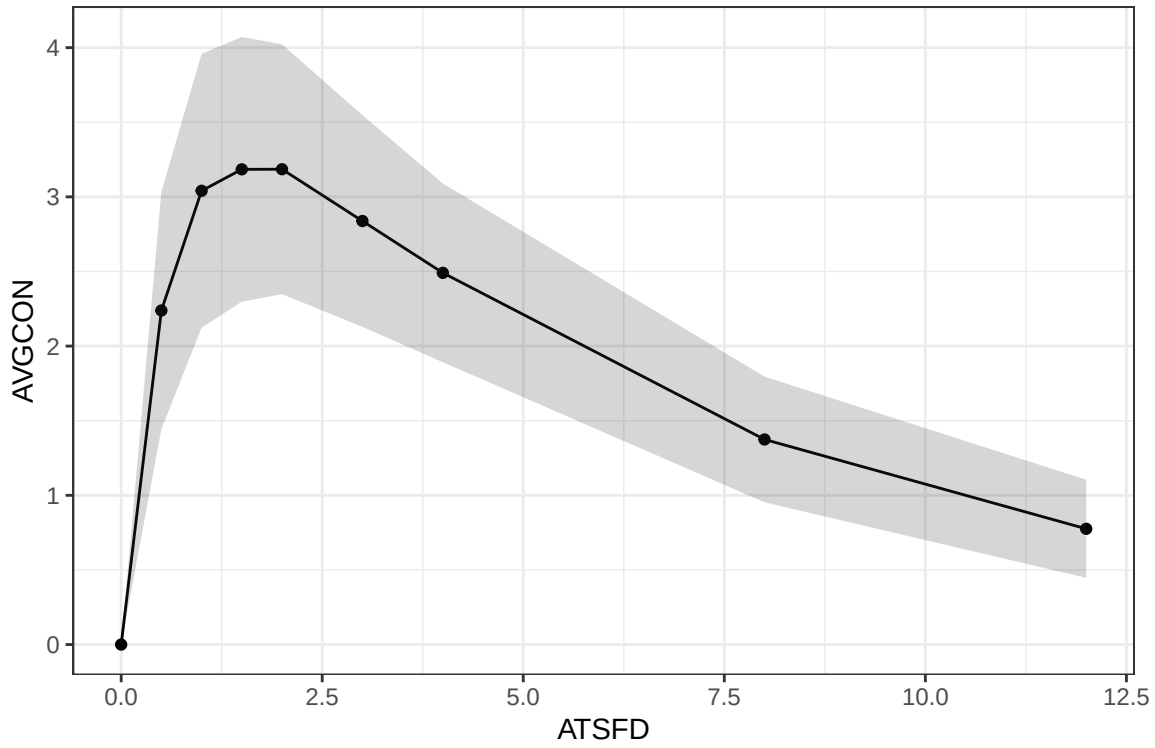
- Add a trend line using `geom_smooth()` with the default (loess method)

```
ggplot(data, aes(ATSFD, CP)) +
  geom_line(aes(group = ID),alpha = 0.2) +
  # group cannot be set within ggplot() since geom_smooth does not
  # need the grouping aesthetic
  geom_point(alpha = 0.2)+
  geom_smooth(se = TRUE, method = "loess")+ # other methods check ?geom_smooth
  theme_bw()
```

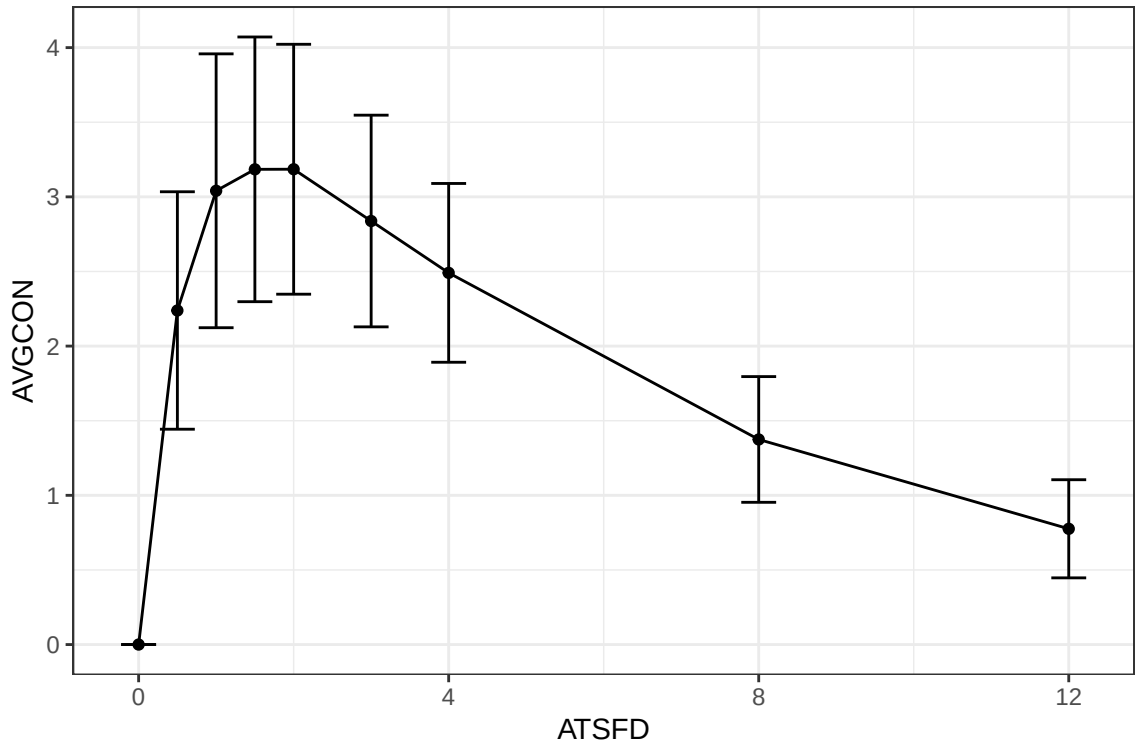


- Add a trend line using `geom_ribbon` and `geom_errorbar()`

```
data %>%
  group_by(ATSFD) %>%
  summarise(AVGCON = mean(CP),
            SDCON = sd(CP)) %>%
  ggplot(aes(ATSFD, AVGCON))+
  geom_point()+
  geom_line()+
  geom_ribbon(aes(ymin = AVGCON - SDCON, ymax = AVGCON + SDCON),
              alpha = 0.2)+
  theme_bw()
```

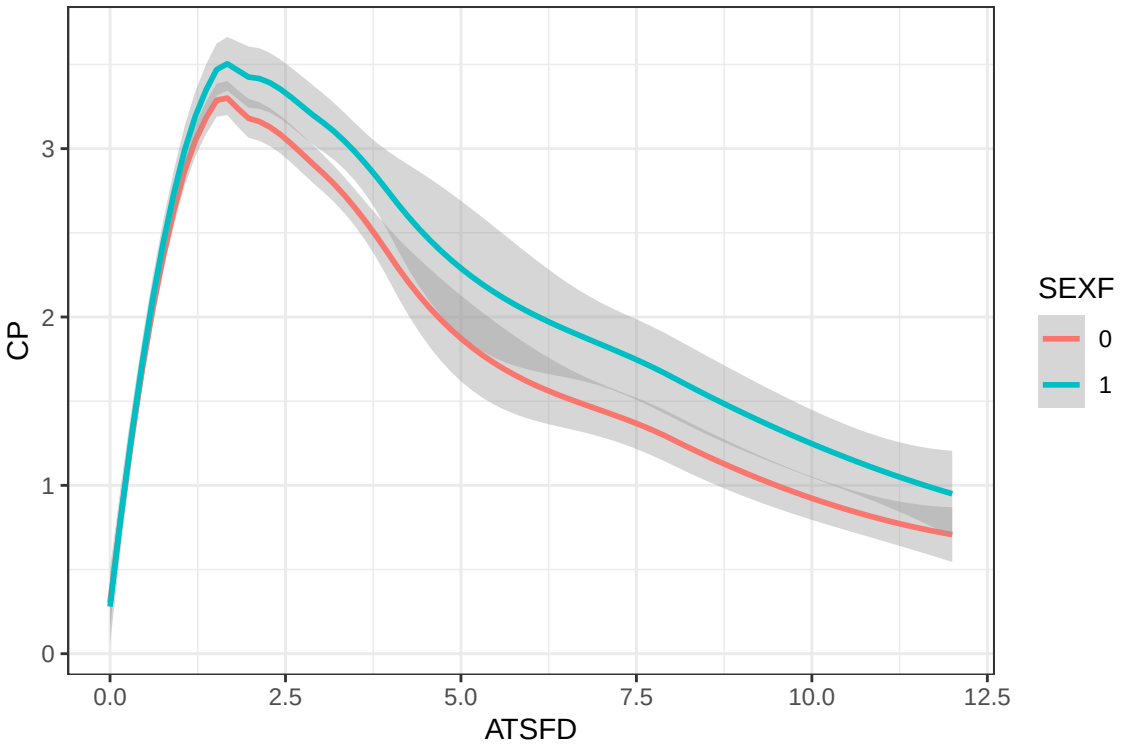



```
data %>%
  group_by(ATSFD) %>%
  summarise(AVGCON = mean(CP),
            SDCON = sd(CP)) %>%
  ggplot(aes(ATSFD, AVGCON))+
  geom_point()+
  geom_line()+
  geom_errorbar(aes(ymin = AVGCON - SDCON, ymax = AVGCON + SDCON),
                linetype = 1,
                size = 0.5)+
  theme_bw()
```



- Plot mean concentration by gender

```
# multiple trend lines
ggplot(data, aes(ATSFD, CP))+
  geom_smooth(aes(color=SEXF), se=TRUE)+
  theme_bw()
```


CP
3
2
1
0
0.0
2.5
5.0
7.5
10.0
12.5
ATSFD
SEXF
0
1

Functions

Practice

1. Create a function to calculate creatinine clearance using the Cockcroft-Gault Equation
2. Calculate creatinine clearance for all the subject in the dataset and save as a new column called CRCL

```
# calculate different body weights: IBW and adjust BW
# return in kilograms
weight_cal <- function(weight_lb, height_in, sex, type=c("IBW","AjBW")) {
  if (!sex %in% c(0,1) ) {
    stop("Specify sex with only 0 (male) or 1 (female)")
  }

   if (!type %in% c("IBW","AjBW") ) {
    stop("Specify type with only 'IBW' or 'AjBW'")
  }

  if (sex == 0 & height_in >= 60) {
    IBW = 50 + 2.3*(height_in-60)
    AjBW = IBW + 0.4*(weight_lb/2.2-IBW)
    AjBW = ifelse(AjBW < IBW, weight_lb/2.2, AjBW)
    # if ABW is less than IBW (underweight), ABW must set to Actual BW
  } else if (sex == 1 & height_in >= 60) {
    IBW = 45.5 + 2.3*(height_in-60)
    AjBW = IBW + 0.4*(weight_lb/2.2-IBW)
    AjBW = ifelse(AjBW < IBW, weight_lb/2.2, AjBW)
  } else if (sex == 0 & height_in < 60) {
    IBW = 50 - (50/60*(60-height_in))
    AjBW = IBW + 0.4*(weight_lb/2.2-IBW)
    AjBW = ifelse(AjBW < IBW, weight_lb/2.2, AjBW)
    # Using baseline method for those shorter than 60 inches
  } else if (sex == 1 & height_in < 60) {
    IBW = 45.5 + (45.5/60*(60-height_in))
    AjBW = IBW + 0.4*(weight_lb/2.2-IBW)
    AjBW = ifelse(AjBW < IBW, weight_lb/2.2, AjBW)
  }

  if (type == "IBW") {
    return(round(IBW,1))
  } else if (type == "AjBW") {
    return(round(AjBW,1))
  } else {
    stop("Incorrect type selected. Use ony 'IBW' or 'AjBW' for type.")
  }
}
```

```
# test the function thoroughly
weight_cal(weight_lb = 80*2.2, height_in = 180/2.54, sex = 0, type = "IBW")
weight_cal(weight_lb = 80*2.2, height_in = 180/2.54, sex = 0, type = "AjBW")
weight_cal(weight_lb = 80*2.2, height_in = 180/2.54, sex = 2, type = "IBW")
weight_cal(weight_lb = 80*2.2, height_in = 180/2.54, sex = 2, type = "IBw")
weight_cal(weight_lb = 80*2.2, height_in = 180/2.54, sex = 1, type = "IBw")
```

```
# calculate creatinine clearance and eGFR
# Cockcroft and Gault formula
```

```r
crcl_cg <- function(scr, age, weight_kg, sex, type=c("CG", "CKD-EPI", "MDRD")) {
  if (sex == 0) {
    crcl_cg = (140-age)*(weight_kg)/(72*scr)
  } else if (sex == 1) {
    crcl_cg = (140-age)*(weight_kg)/(72*scr)*0.85
  }
  return(round(crcl_cg,1))
}

crcl_cg <- function(scr, age, weight_kg, sex) {
  if (sex == 0) {
    crcl_cg = (140-age)*(weight_kg)/(72*scr)
  } else if (sex == 1) {
    crcl_cg = (140-age)*(weight_kg)/(72*scr)*0.85
  }
  return(round(crcl_cg,1))
}
```

```r
# calculate IBW, ABW, and CrCAL
data %>%
  mutate(IBW = weight_cal(weight_lb = WT*2.2, height_in = HT/2.54, sex = SEXF, type = "IBW"),
         ABW = weight_cal(weight_lb = WT*2.2, height_in = HT/2.54, sex = SEXF, type = "AjBW"),
         CRCLCG = crcl_cg(scr=SCR, age=AGE, weight_kg = WT, sex=SEXF))
```

Resources

1. R for Data Science by Hadley Wickham and Garrett Grolemund, online at https://r4ds.had.co.nz/
2. Introduction to Data Science by Rafael A. Irizarry, online at https://rafalab.github.io/dsbook/
3. R cheetsheets https://rstudio.com/resources/cheatsheets/
    - Data Transformation Cheatsheet
    - Dates and Times Cheatsheet
    - R Markdown Cheatsheet
    - Data Visualization Cheatsheet
4. R Markdown: The Definitive Guide by Yihui Xie, J. J. Allaire, Garrett Grolemund, online at https://bookdown.org/yihui/rmarkdown/
5. RMarkdown for Scientists by Nicholas Tierney, online at https://rmd4sci.njtierney.com/
6. bookdown: Authoring Books and Technical Documents with R Markdown by Yihui Xie, online at https://bookdown.org/yihui/bookdown/
7. R Packages by Hadley Wickham and Jenny Bryan, online at https://r-pkgs.org/

Session Info

```
sessionInfo()
```

```
## R version 4.1.0 (2021-05-18)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19042)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] lubridate_1.7.10 forcats_0.5.1    stringr_1.4.0    dplyr_1.0.6
##  [5] purrr_0.3.4      readr_1.4.0      tidyr_1.1.3      tibble_3.1.2
##  [9] ggplot2_3.3.3    tidyverse_1.3.1
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_1.1.1  xfun_0.23         lattice_0.20-44   splines_4.1.0
##  [5] haven_2.4.1       colorspace_2.0-1  vctrs_0.3.8       generics_0.1.0
##  [9] htmltools_0.5.1.1 mgcv_1.8-35       yaml_2.2.1        utf8_1.2.1
## [13] rlang_0.4.11      pillar_1.6.1      glue_1.4.2        withr_2.4.2
## [17] DBI_1.1.1         dbplyr_2.1.1      modelr_0.1.8      readxl_1.3.1
## [21] lifecycle_1.0.0   munsell_0.5.0     gtable_0.3.0      cellranger_1.1.0
## [25] rvest_1.0.0       evaluate_0.14     labeling_0.4.2    knitr_1.33
## [29] fansi_0.4.2       highr_0.9         broom_0.7.6       Rcpp_1.0.6
## [33] scales_1.1.1      backports_1.2.1   jsonlite_1.7.2    farver_2.1.0
## [37] fs_1.5.0          hms_1.1.0         digest_0.6.27     stringi_1.6.1
## [41] grid_4.1.0        cli_2.5.0         tools_4.1.0       magrittr_2.0.1
```

```
## [45] crayon_1.4.1      pkgconfig_2.0.3   Matrix_1.3-3      ellipsis_0.3.2
## [49] xml2_1.3.2        reprex_2.0.0      assertthat_0.2.1  rmarkdown_2.8
## [53] httr_1.4.2        rstudioapi_0.13   R6_2.5.0          nlme_3.1-152
## [57] compiler_4.1.0
```