PHM5002: Metabolomics Data Preprocessing and QC

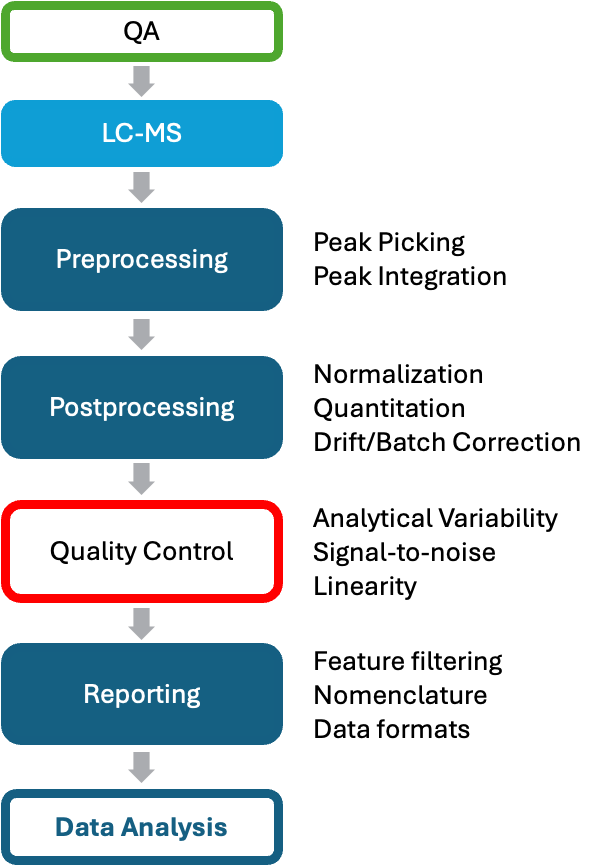
From Raw Data to Reliable Analytical Results

Bo Burla / Singapore Lipidomics Incubator @ NUS

2025-09-15

## 1. Introduction

Welcome to this R-based tutorial on data processing of a targeted plasma lipidomics dataset. We will use the dataset published by [Tan et al., *Arterioscler Thromb Vasc Biol*, 2022](https://www.ahajournals.org/doi/suppl/10.1161/ATVBAHA.121.316847) as an example. In that study, a panel of 413 plasma lipids was quantified by liquid chromatography–tandem mass spectrometry (LC–MS/MS) using multiple reaction monitoring (MRM). We begin with preprocessed LC-MS raw data, i.e., peak-area data that were generated using the MRMkit software ([Teo et al., *Anal. Chem.,* 2020](https://dx.doi.org/10.1021/acs.analchem.0c03060)). See Figure 1 for an overview of the typical data processing flow, of which we will cover aspects of the postprocessing, quality control and reporting.



**Figure 1**: Data Processing Overview

## 2. Loading R packages

We will use several packages from the [tidyverse](https://www.tidyverse.org) for this tutorial. The [here](https://here.r-lib.org) package will be used to tracking data file paths. Furthermore, we define the point colors and shapes for the QC sample types, which will be used later in the plots.

|  |
| --- |
| Important |
| NOTE: It is a good practice to restart R before you run or rerun code to clear all loaded data and variables. This can be done via *Session -> Restart R* or via SHIFT-ALT-0 (Windows) or SHIFT-⌘-0 (MacOS). |

library(tidyverse)  
#> ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
#> ✔ dplyr 1.1.4 ✔ readr 2.1.5  
#> ✔ forcats 1.0.0 ✔ stringr 1.5.1  
#> ✔ ggplot2 3.5.2 ✔ tibble 3.3.0  
#> ✔ lubridate 1.9.4 ✔ tidyr 1.3.1  
#> ✔ purrr 1.1.0   
#> ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
#> ✖ dplyr::filter() masks stats::filter()  
#> ✖ dplyr::lag() masks stats::lag()  
#> ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors  
library(here)  
#> here() starts at /Users/lsibjb/Documents/Code/PHM5002  
here::i\_am("PHM5002-Tutorial.qmd")  
#> here() starts at /Users/lsibjb/Documents/Code/PHM5002  
  
# Define point colors and shapes for each qc\_type in the plots  
qc\_colors <- c(SAMPLE = "grey50", QC = "red", BLANK = "#00db7c", RESP = "blue")  
qc\_shapes <- c(SAMPLE = 1, QC = 21, BLANK = 23, RESP = 22)

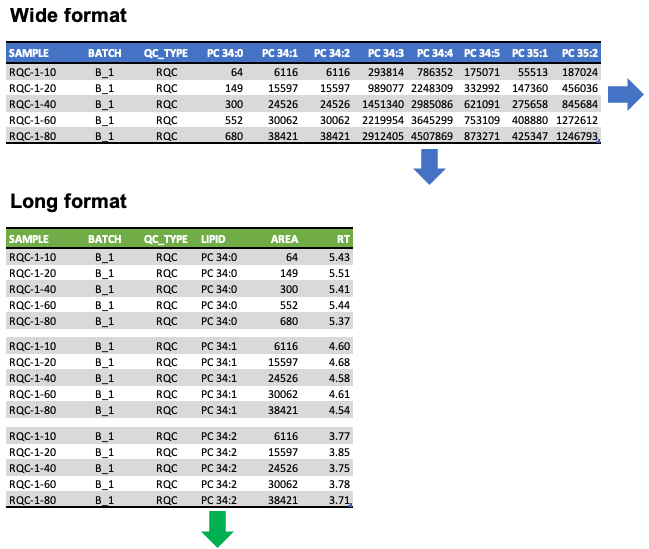
## 3. Importing raw data

We begin by loading the table containing peak area data. After import, it is important to verify that the data were loaded correctly, completely, and with appropriate column types. Missing values or undefined text entries can cause columns to be interpreted as character rather than numeric data.

d\_orig <- readr::read\_csv(  
 file = here("data/SPERFECT\_RawAreas\_Tutorial.csv"),  
 col\_names = TRUE,   
 trim\_ws = TRUE,   
 na = c("NA", "ND", "n.d."))  
#> Rows: 445 Columns: 416  
#> ── Column specification ────────────────────────────────────────────────────────  
#> Delimiter: ","  
#> chr (3): filename, batch\_id, qc\_type  
#> dbl (413): CE 14:0, CE 15:0, CE 16:0, CE 16:1, CE 16:2, CE 17:0, CE 17:1, CE...  
#>   
#> ℹ Use `spec()` to retrieve the full column specification for this data.  
#> ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.  
  
head(d\_orig)  
#> # A tibble: 6 × 416  
#> filename batch\_id qc\_type `CE 14:0` `CE 15:0` `CE 16:0` `CE 16:1` `CE 16:2`  
#> <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
#> 1 PBLK-1.mzML Longit\_1 BLANK 47.3 54.6 440. 143. 0.568  
#> 2 RQC-1-10.m… Longit\_1 RESP 87.3 262. 23404. 3271. 248.   
#> 3 RQC-1-20.m… Longit\_1 RESP 210. 530. 37327. 4811. 226.   
#> 4 RQC-1-40.m… Longit\_1 RESP 335. 186. 52478. 4923. 307.   
#> 5 RQC-1-60.m… Longit\_1 RESP 189. 239. 66109. 5774. 417.   
#> 6 RQC-1-80.m… Longit\_1 RESP 592. 230. 75214. 6100. 256.   
#> # ℹ 408 more variables: `CE 17:0` <dbl>, `CE 17:1` <dbl>, `CE 18:0` <dbl>,  
#> # `CE 18:1` <dbl>, `CE 18:1 d7 (ISTD)` <dbl>, `CE 18:2` <dbl>,  
#> # `CE 18:3` <dbl>, `CE 20:1` <dbl>, `CE 20:2` <dbl>, `CE 20:3` <dbl>,  
#> # `CE 20:4` <dbl>, `CE 20:5` <dbl>, `CE 22:0` <dbl>, `CE 22:1` <dbl>,  
#> # `CE 22:4` <dbl>, `CE 22:5` <dbl>, `CE 22:6` <dbl>, `CE 24:0` <dbl>,  
#> # `CE 24:1` <dbl>, `CE 24:4` <dbl>, `CE 24:5` <dbl>, `CE 24:6` <dbl>,  
#> # `Cer d18:0/16:0` <dbl>, `Cer d18:0/18:0` <dbl>, `Cer d18:0/20:0` <dbl>, …

## 4. Preparing and reshaping the data (Data Wrangling)

First, we will rearrange the imported dataset for the subsequent steps. We remove the .mzML file extension from the entries in the sample\_id column, which will then be renamed to sample\_id. Furthermore, we will add a column containing the analysis order number (run\_no), assuming the imported data were in the actual analysis sequence. In the second step, we will convert the data into a *long format*. In this format, each observation (a unique sample/lipid pair) uses a single row. Columns then represent the measured variables for that observation, such as peak area or retention time.



**Figure 2**: Comparison of Wide vs. Long-Format Tables. The column shown in red is an additional feature variable not available in the wide table.

d\_orig <- d\_orig |>  
 mutate(filename = stringr::str\_replace(filename, ".mzML", "")) |>  
 mutate(run\_no = row\_number(), .before = 1)  
  
d\_full <- d\_orig |>  
 pivot\_longer(  
 names\_to = "lipid",   
 values\_to = "area",  
 cols = -run\_no:-qc\_type  
 ) |>  
 arrange(lipid) |>  
 rename(sample\_id = filename)  
  
head(d\_full)  
#> # A tibble: 6 × 6  
#> run\_no sample\_id batch\_id qc\_type lipid area  
#> <int> <chr> <chr> <chr> <chr> <dbl>  
#> 1 1 PBLK-1 Longit\_1 BLANK CE 14:0 47.3  
#> 2 2 RQC-1-10 Longit\_1 RESP CE 14:0 87.3  
#> 3 3 RQC-1-20 Longit\_1 RESP CE 14:0 210.   
#> 4 4 RQC-1-40 Longit\_1 RESP CE 14:0 335.   
#> 5 5 RQC-1-60 Longit\_1 RESP CE 14:0 189.   
#> 6 6 RQC-1-80 Longit\_1 RESP CE 14:0 592.

# Example Workflow

## 5. Subset the data

We will focus on a selected subset of the measured analytes in this first part of the tutorial for illustrative purposes. For this, few measured triacylglycerol (TG) species and the corresponding internal standard will be selected. The code to process the full dataset can be found at the end of this tutorial (Section 15).

We subset the measured lipid species using *regular expressions* and include the TG internal standard as well. You can modify the code below to select other lipid species.

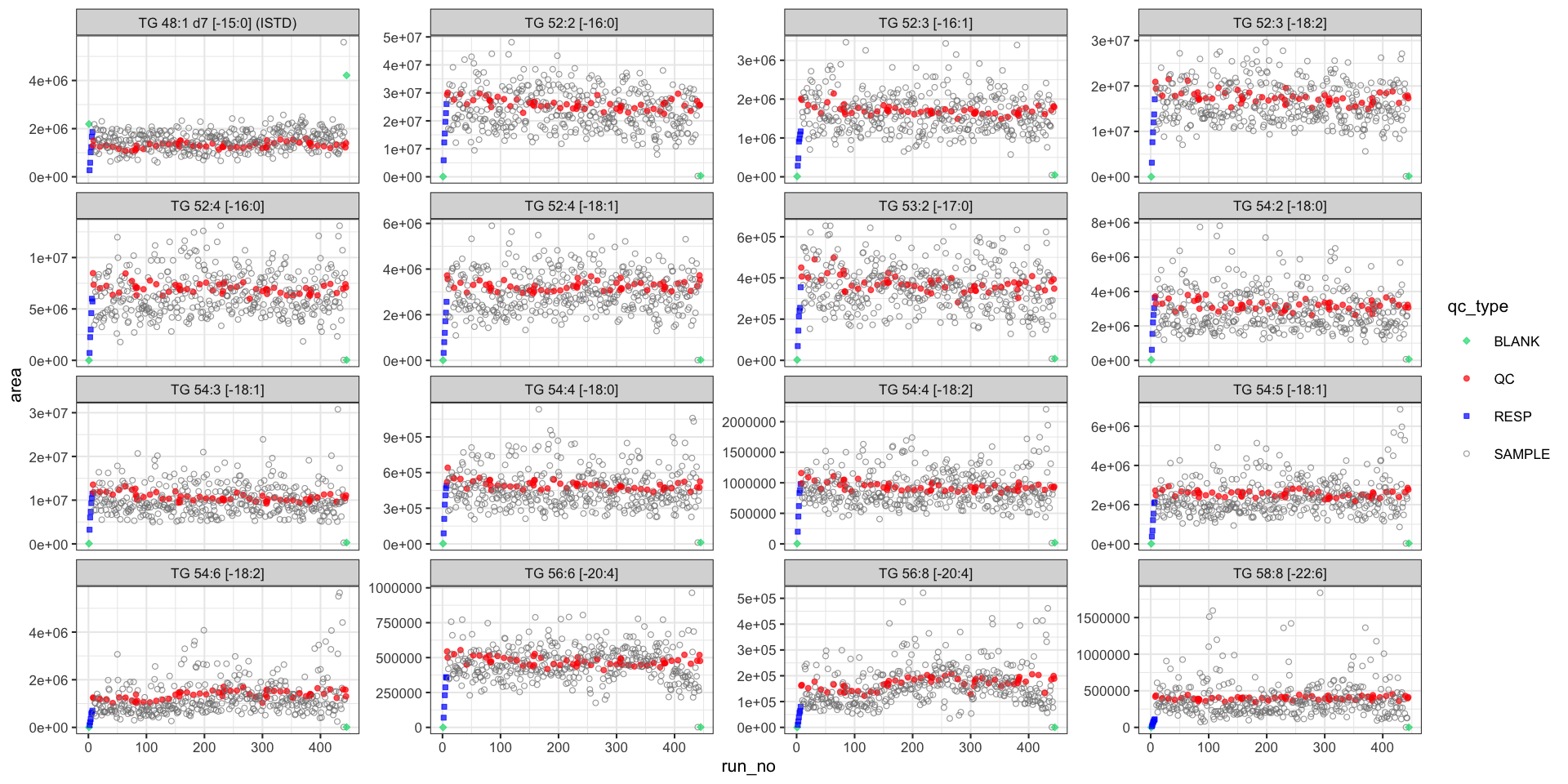
d\_subset <- d\_full |>   
 filter(str\_detect(lipid, "^TG 5[2-9]:[2-8] \\[\\-") | lipid == "TG 48:1 d7 [-15:0] (ISTD)")  
head(d\_subset)  
#> # A tibble: 6 × 6  
#> run\_no sample\_id batch\_id qc\_type lipid area  
#> <int> <chr> <chr> <chr> <chr> <dbl>  
#> 1 1 PBLK-1 Longit\_1 BLANK TG 48:1 d7 [-15:0] (ISTD) 2190563.  
#> 2 2 RQC-1-10 Longit\_1 RESP TG 48:1 d7 [-15:0] (ISTD) 274380.  
#> 3 3 RQC-1-20 Longit\_1 RESP TG 48:1 d7 [-15:0] (ISTD) 585267.  
#> 4 4 RQC-1-40 Longit\_1 RESP TG 48:1 d7 [-15:0] (ISTD) 1018345.  
#> 5 5 RQC-1-60 Longit\_1 RESP TG 48:1 d7 [-15:0] (ISTD) 1236618.  
#> 6 6 RQC-1-80 Longit\_1 RESP TG 48:1 d7 [-15:0] (ISTD) 1663751.

## 6. A first look into the data

Now we take a first look on how the analysis went. To do this, we inspect the peak areas of over the analysis sequence. Different QC samples were included in this analysis:

* QC (Quality Controls): Pooled plasma samples, co-extracted with study samples
* BLANK (Blanks): Samples without matrix (plasma) to determine background signals
* RESP (Response QCs): To check for linear response

ggplot(d\_subset, aes(x = run\_no, y = area)) +   
 geom\_point(aes(colour = qc\_type,   
 fill = qc\_type,   
 shape = qc\_type),  
 size = 1,   
 alpha =0.7,   
 stroke = 0.3) +  
 facet\_wrap(vars(lipid),   
 ncol = 4,   
 nrow = 4,   
 scales="free\_y") +  
 scale\_shape\_manual(na.value = NA, values = qc\_shapes) +  
 scale\_fill\_manual(values = qc\_colors, na.value = NA) +  
 scale\_colour\_manual(values = qc\_colors, na.value = NA) +  
 scale\_x\_continuous(breaks = seq(0, max(d\_subset$run\_no), by = 100 )) +   
 scale\_y\_continuous(limits = c(0, NA)) +   
 theme\_bw(base\_size = 8)



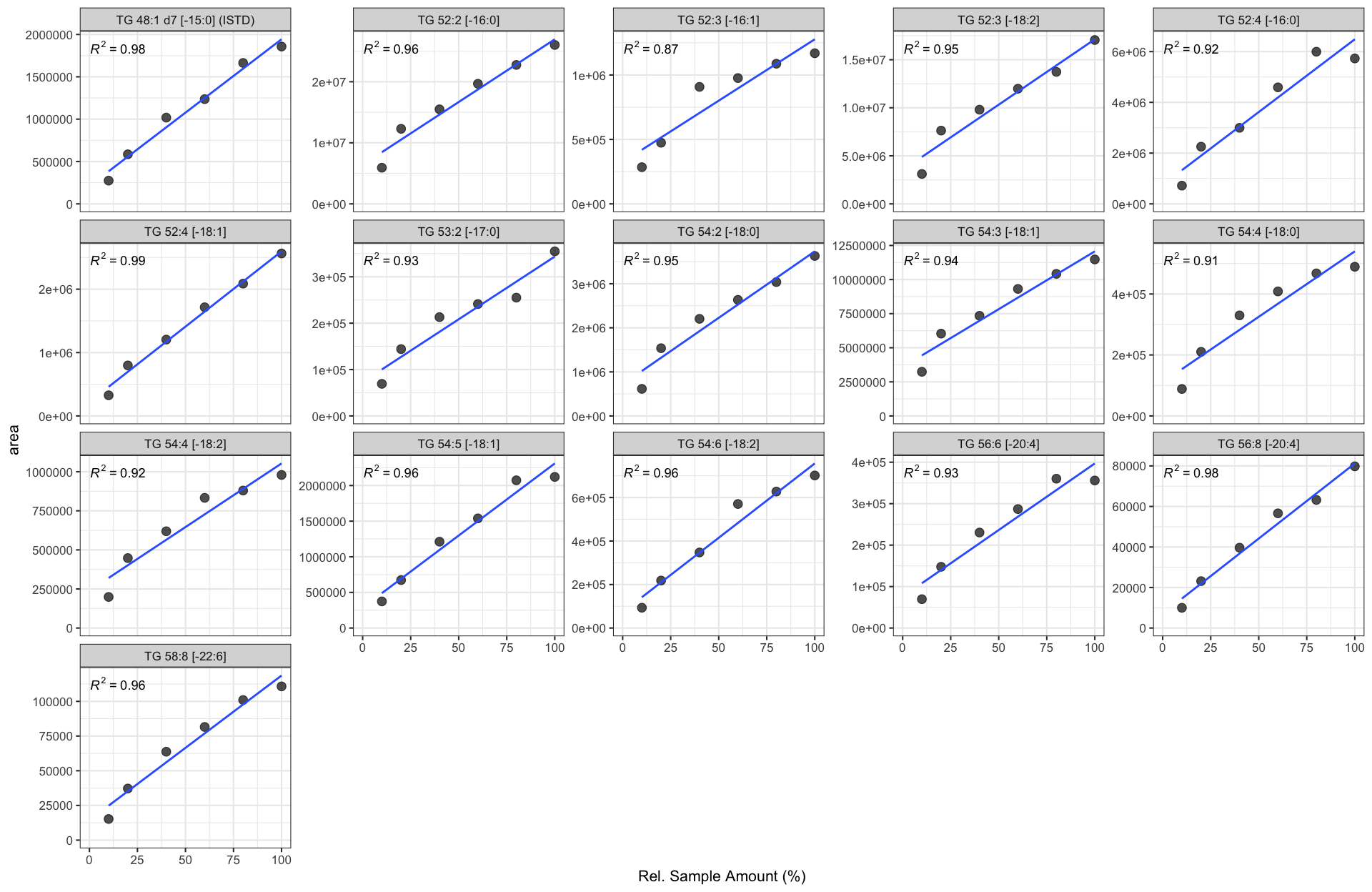
The quality-control (QC) samples (red points) are relatively stable and exhibit low variability throughout the analysis (4 days). This suggests a low analytical variability in the measurement of the shown analytes. In contrast, the study samples (grey) show greater variability, reflecting biological variability in addition to analytical variability.

What do you observe for the spiked-in internal standard (top left)? What about the blanks?

## 7. Linearity

Analyzed sample amounts need to be carefully chosen when measuring analytes covering a large concentration range. It is a trade-off between sensitivity and not exceeding the linear range of the measurement. Below we plot the response curves and fit linear regression lines to the data. We observe an overall good linearity for most species.

# Extract sample details from the sample name  
d\_rqc <- d\_subset |>   
 filter(qc\_type == "RESP") |>   
 separate(col = sample\_id,   
 into = c("type","curve\_no","amount"),   
 sep = "-",   
 remove = FALSE, convert = TRUE)  
d\_rqc$amount <- as.numeric(d\_rqc$amount)  
  
# plot   
ggplot(d\_rqc, aes(x = amount, y = area)) +  
 geom\_point(size = 2, alpha =0.7, stroke = 0.3) +  
 facet\_wrap(vars(lipid), ncol = 5, nrow = 4, scales="free\_y") +  
 ggpmisc::stat\_poly\_line(linewidth = 0.5, se = FALSE) +  
 ggpmisc::stat\_poly\_eq(aes(label = after\_stat(rr.label)),  
 size = 2.4,  
 lineheight = 1) +  
 scale\_x\_continuous(limits = c(0, NA)) +   
 scale\_y\_continuous(limits = c(0, NA)) +   
 labs(x = "Rel. Sample Amount (%)") +  
 theme\_bw(base\_size = 8)  
#> Registered S3 methods overwritten by 'ggpp':  
#> method from   
#> heightDetails.titleGrob ggplot2  
#> widthDetails.titleGrob ggplot2



## 8. Normalization and quantification

To reduce technical variability and convert peak areas to concentrations, we will use internal-standard (ISTD) normalization. Internal standards are compounds with similar chemical/physicochemical properties to the analytes, but are distinguishable in the used assay. Suitable ISTDs include isotope‑labelled versions of the analytes or structurally similar compounds not present in the sample matrix.

Below we iterate over each sample and divide the peak area of each lipid species by the peak area of the corresponding TG class‑specific ISTD in the sample. The normalized areas are stored in a new column norm\_area.

d\_subset <- d\_subset |>  
 group\_by(sample\_id) |>  
 mutate(  
 norm\_area = area / area[lipid == "TG 48:1 d7 [-15:0] (ISTD)"]  
 )

Since we know how much ISTD was spiked into each sample, we can calculate the concentration of each lipid species using:

is the peak area of a lipid species, is the peak area of the class-specific ISTD used for normalization, is the volume of ISTD solution spiked into the sample, is the concentration of ISTD in the mix, and is the amount (volume) of sample extracted.

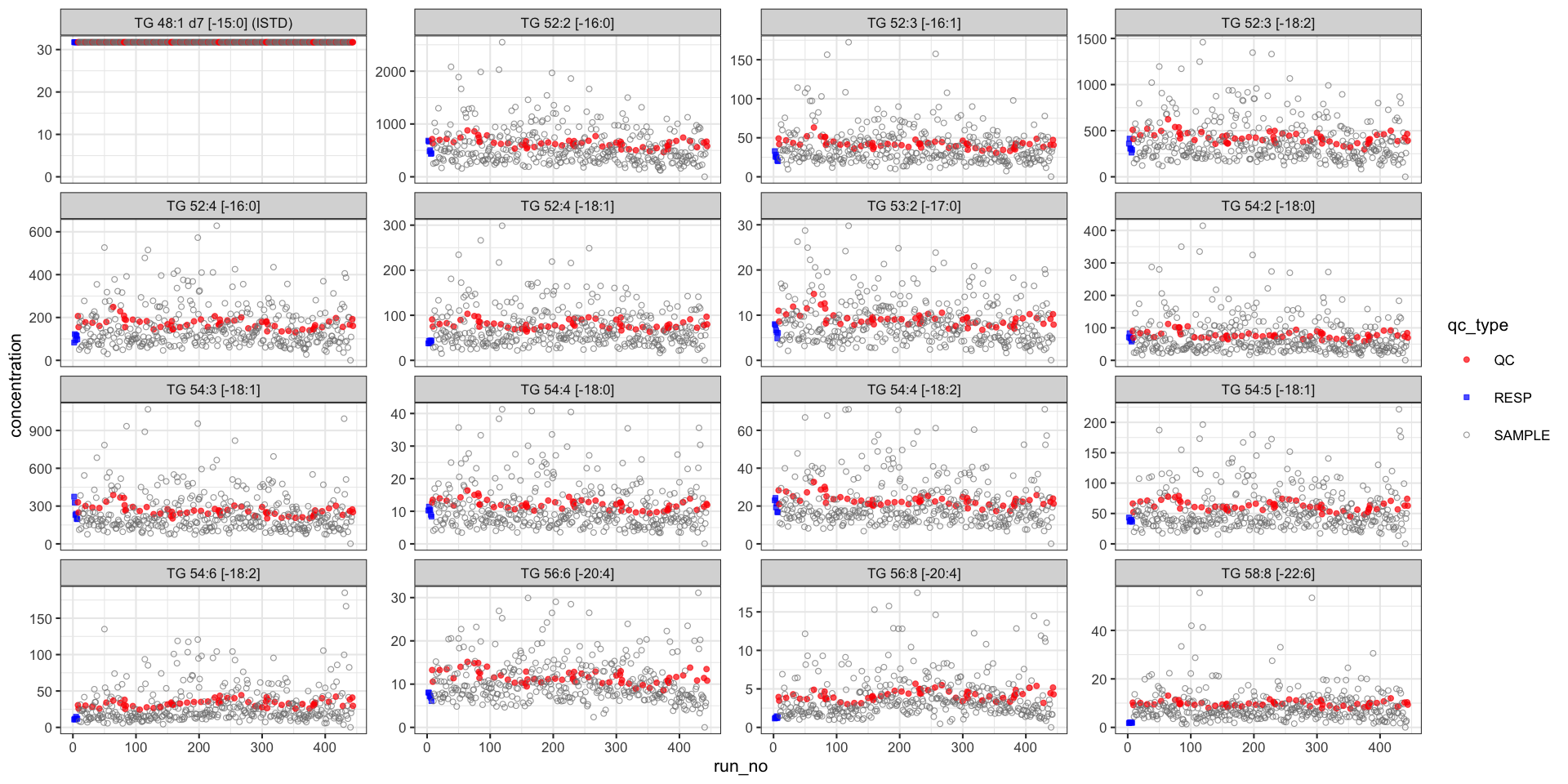
In our case, 4.5 µL of ISTD solution was spiked into 10 µL plasma, and the concentration of the TG ISTD in the spike-in solution is 70.53 µmol/L. We hence calculate the concentrations as shown below, with the calculated concentrations being stored in the new column concentration. The concentration unit is µmol/L.

d\_subset <- d\_subset |>  
 group\_by(sample\_id) |>  
 mutate(  
 concentration = norm\_area \* (4.5 \* 70.53) / 10   
 )

## 9. Inspecting the normalized data

Normalization with a class-specific ISTD can reduce variability that resulted from sample processing and the LC–MS analysis, but it may also introduce noise or artifacts. Let’s examine how the data look after normalization.

plot\_runscatter <- function(data, variable){  
 data <- data |> filter(!str\_detect(qc\_type, "BLANK|RQC"))  
 ggplot(data, aes(x = run\_no, y = {{variable}})) +   
 geom\_point(aes(colour = qc\_type,   
 fill = qc\_type,   
 shape = qc\_type),  
 size = 1, alpha =0.7, stroke = 0.3) +  
 facet\_wrap(vars(lipid),   
 ncol = 4,   
 nrow = 4,   
 scales="free\_y") +  
 scale\_shape\_manual(na.value = NA, values = qc\_shapes) +  
 scale\_fill\_manual(values = qc\_colors, na.value = NA) +  
 scale\_colour\_manual(values = qc\_colors, na.value = NA) +  
 scale\_x\_continuous(breaks = seq(0, max(d\_subset$run\_no), by = 100 )) +   
 scale\_y\_continuous(limits = c(0, NA)) +   
 theme\_bw(base\_size = 8)   
 }  
  
plot\_runscatter(d\_subset, concentration)



## 10. Drift and batch correction

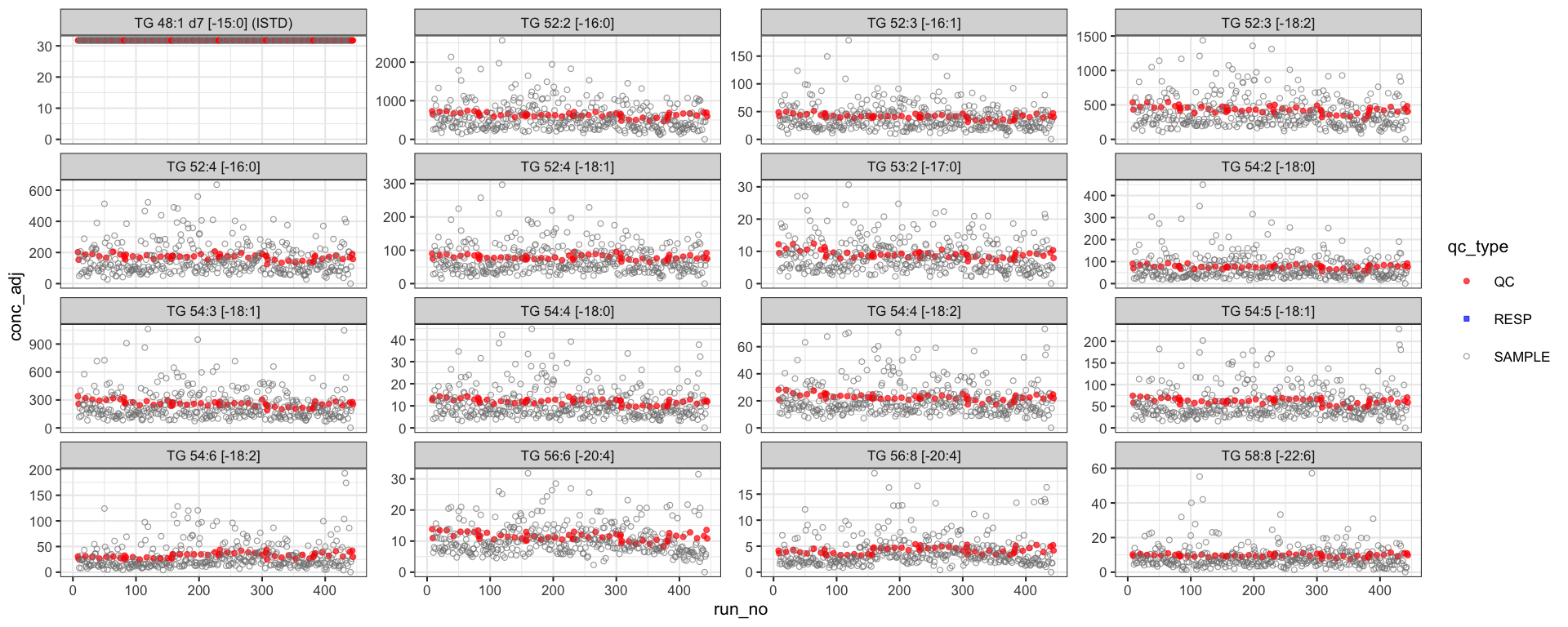
Some drifts are still present in the data. We will attempt to correct these drifts and batch effects, assuming that the QCs reflect the same drift and batch effects affecting the study samples. First, we apply within-batch loess smoothing.

**Note**: In the publication another algorithm was used, which was based on the all data points rather than just QCs.

d\_subset$y\_predicted <- NULL # clean up in case chunk is rerun  
  
  
# Define a robust loess-based smoothing function  
get\_loess <- function(d) {  
 tryCatch({  
 dt <- tibble::tibble(run\_no = seq(min(d$run\_no), max(d$run\_no), 1))  
 res <- stats::loess(  
 data = subset(d, qc\_type == "QC"), formula = conc\_log ~ run\_no, span = 0.8) |>   
 stats::predict(dt) |> as.numeric()  
 res},  
 error = function(e) {return(rep(NA\_real\_, length(d$run\_no)))})  
}  
  
# Log-transform concentration values to make loess more robust  
d\_subset$conc\_log <- log2(d\_subset$concentration ) # log-transform concentration  
  
# Calculate loess fit for each lipid in each batch  
d\_subset <- d\_subset |>   
 group\_by(lipid, batch\_id) |>   
 nest() |>  
 mutate(y\_predicted = purrr::map(data, \(x) get\_loess(x))) |>   
 unnest(cols = c(data, y\_predicted))  
  
# Smooth study samples based on the loess fit of the QCs and   
# and transform values back to concentration scale  
d\_subset <- d\_subset |>  
 group\_by(lipid, batch\_id) |>   
 mutate(  
 y\_predicted = y\_predicted - median(y\_predicted, na.rm = TRUE),  
 conc\_adj = 2^(conc\_log - y\_predicted)) |>   
 ungroup()

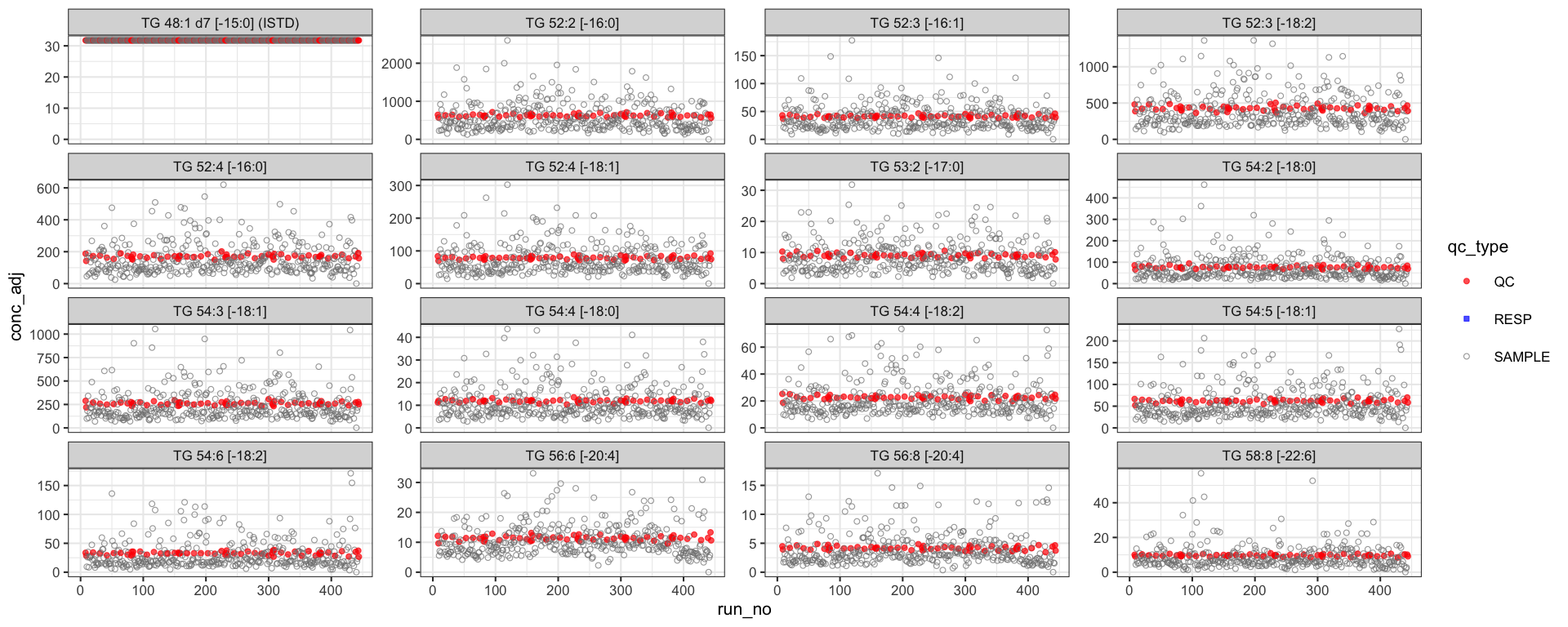
Let’s have a look how the data looks after drift correction.

plot\_runscatter(d\_subset, conc\_adj)



Next, we apply QC-based median-centering to correct batch effects and plot the data again.

# Between-batch correction via QC-based median centering  
d\_subset <- d\_subset |>   
 group\_by(lipid, batch\_id) |>   
 mutate(  
 conc\_median = median(conc\_adj[qc\_type == "QC"], na.rm = TRUE),  
 conc\_adj = conc\_adj/conc\_median  
 )   
  
# Rescaling to grand mean  
d\_subset <- d\_subset |>   
 group\_by(lipid) |>  
 mutate(  
 conc\_adj = conc\_adj \* mean(conc\_median[qc\_type == "QC"], na.rm = TRUE)  
 ) |>   
 ungroup()  
  
plot\_runscatter(d\_subset, conc\_adj)



What do you observe? Do you have any concerns with this approach?

## 11. Calculate quality-control (QC) metrics

To evaluate the analysis quality and filter the data, we calculate QC metrics for each lipid species. These include the analytical coefficient of variation (%CV) based on the QCs and the signal-to-blank ratio.

cv <- function(x) sd(x, na.rm = TRUE)/mean(x, na.rm = TRUE) \* 100  
  
d\_qc <- d\_subset |>   
 group\_by(lipid) |>   
 summarise(  
 signal\_blank\_ratio = median(area[qc\_type == "SAMPLE"], na.rm = TRUE) /   
 median(area[qc\_type == "BLANK"], na.rm = TRUE),  
 cv\_qc\_area = cv(area[qc\_type == "QC"]),  
 cv\_qc\_concadj = cv(conc\_adj[qc\_type == "QC"]),   
 mean\_conc = mean(conc\_adj[qc\_type == "SAMPLE"], na.rm = TRUE)   
 )  
  
 d\_qc  
#> # A tibble: 16 × 5  
#> lipid signal\_blank\_ratio cv\_qc\_area cv\_qc\_concadj mean\_conc  
#> <chr> <dbl> <dbl> <dbl> <dbl>  
#> 1 TG 48:1 d7 [-15:0] (IS… 0.469 9.21 1.76e-13 31.7   
#> 2 TG 52:2 [-16:0] 103. 7.23 5.94e+ 0 586.   
#> 3 TG 52:3 [-16:1] 54.1 6.91 5.70e+ 0 38.9   
#> 4 TG 52:3 [-18:2] 184. 7.98 7.66e+ 0 395.   
#> 5 TG 52:4 [-16:0] 302. 7.20 6.81e+ 0 158.   
#> 6 TG 52:4 [-18:1] 285. 6.29 6.10e+ 0 73.2   
#> 7 TG 53:2 [-17:0] 65.5 10.6 7.34e+ 0 8.61  
#> 8 TG 54:2 [-18:0] 55.1 8.94 8.09e+ 0 72.5   
#> 9 TG 54:3 [-18:1] 51.7 8.91 6.60e+ 0 242.   
#> 10 TG 54:4 [-18:0] 62.8 7.20 5.19e+ 0 11.3   
#> 11 TG 54:4 [-18:2] 73.3 7.31 6.22e+ 0 21.4   
#> 12 TG 54:5 [-18:1] 123. 6.39 6.10e+ 0 56.7   
#> 13 TG 54:6 [-18:2] 115. 13.2 8.76e+ 0 30.1   
#> 14 TG 56:6 [-20:4] 364. 6.61 6.91e+ 0 10.6   
#> 15 TG 56:8 [-20:4] 1011. 13.0 7.86e+ 0 3.94  
#> 16 TG 58:8 [-22:6] 488. 6.18 5.82e+ 0 8.79

| What do you observe? How do the %CVs compare with clinical diagnostics–grade TG assays?

## 12. Feature Filtering

Next, we exclude features (lipid species) that were not reliably measured, i.e., features that do not meet predefined QC criteria such as acceptable analytical variability and sufficient signal-to-blank ratio. From the QC metrics above, all processed TG species have %CV < 25% and a signal-to-blank ratio > 10. Thus, all measured species meet our QC criteria and will be reported.

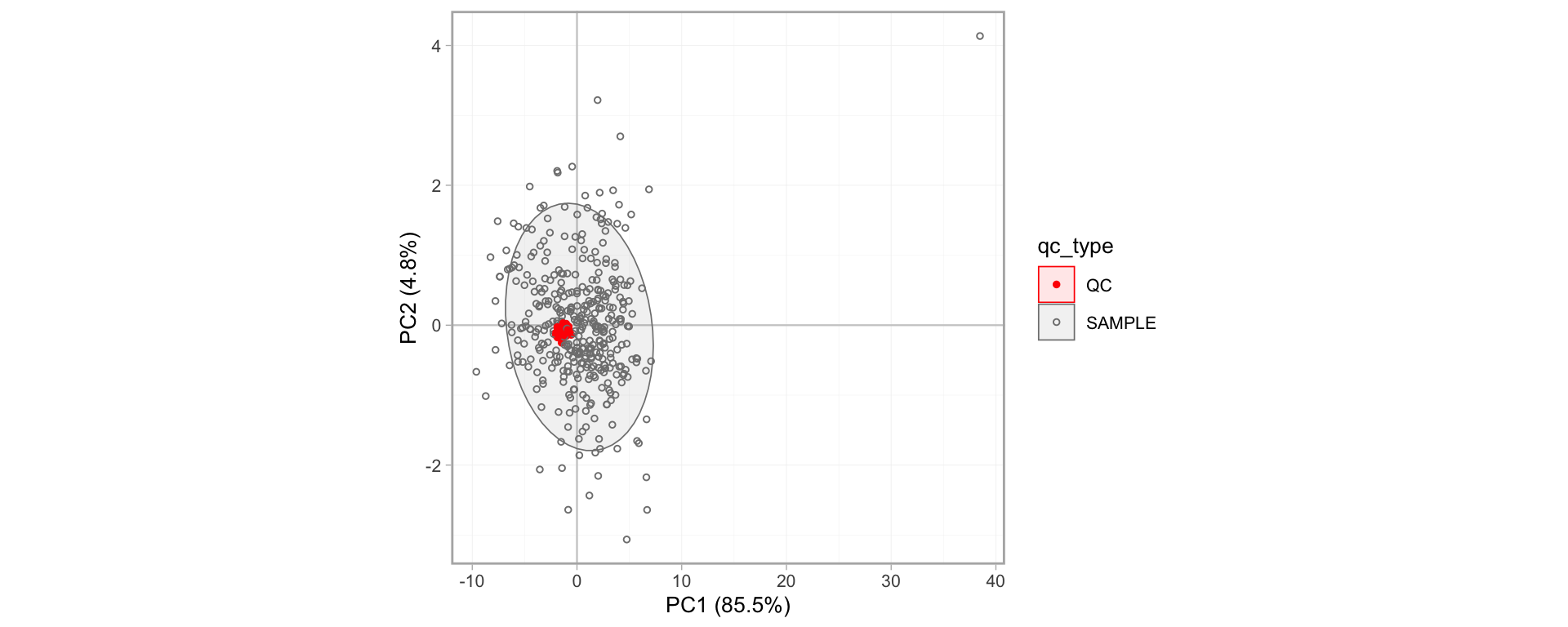
See Section 18 below for the results of the feature filtering when applied to the full dataset.

## 13. Principal Component Analysis (PCA)

As a final step, we generate a PCA plot from the concentration data to assess overall structure and detect any clear outliers that may need to be investigated.

| What do you observe? What patterns would you expect to see?

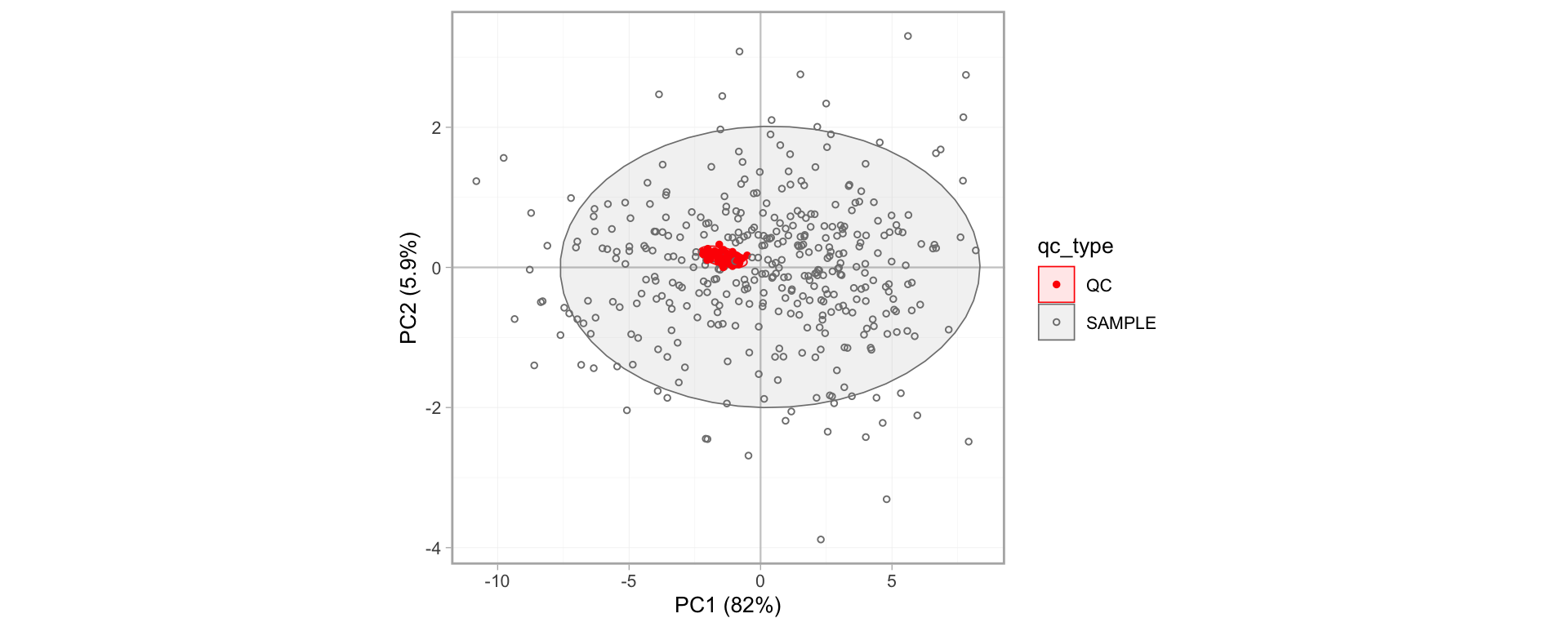
plot\_pca <- function(data, variable) {  
 d\_pca <- data |>  
 filter(qc\_type %in% c("QC", "SAMPLE"), !str\_detect(lipid, "ISTD")) |>  
 select(sample\_id, qc\_type, batch\_id, lipid, {{variable}}) |>  
 pivot\_wider(names\_from = "lipid", values\_from = {{variable}}) |>  
 drop\_na() # NA rows of rows that could not be smoothed before  
   
 d\_metadata <- d\_pca |> select(sample\_id, qc\_type, batch\_id) |> distinct()  
   
 m\_raw <- d\_pca |> select(-qc\_type, -batch\_id) |>  
 select(where( ~ !any(is.na(.)))) |>  
 column\_to\_rownames("sample\_id") |>  
 as.matrix()  
   
 m\_raw <- log2(m\_raw)  
   
   
 # get pca result with annotation  
 pca\_res <- prcomp(m\_raw, scale = TRUE, center = TRUE)  
 pca\_annot <- pca\_res |> broom::augment(d\_metadata)  
 pca\_contrib <- pca\_res |> broom::tidy(matrix = "eigenvalues")  
   
 p <- ggplot(  
 data = pca\_annot,  
 aes(  
 x = .fittedPC1,  
 y = .fittedPC2,  
 color = qc\_type,  
 fill = qc\_type,  
 shape = qc\_type,  
 label = "sample\_id"  
 )  
 ) +  
 ggplot2::geom\_hline(yintercept = 0,  
 linewidth = 0.4,  
 color = "grey80") +  
 ggplot2::geom\_vline(xintercept = 0,  
 linewidth = 0.4,  
 color = "grey80") +  
 ggplot2::stat\_ellipse(  
 geom = "polygon",  
 level = 0.95,  
 alpha = 0.1,  
 linewidth = 0.3  
 ) +  
 ggplot2::geom\_point(size = 1) +  
 ggplot2::scale\_color\_manual(values = qc\_colors, drop = TRUE) +  
 ggplot2::scale\_fill\_manual(values = qc\_colors, drop = TRUE) +  
 ggplot2::scale\_shape\_manual(values = qc\_shapes, drop = TRUE) +  
 ggplot2::theme\_light(base\_size = 10) +  
 ggplot2::xlab(glue::glue("PC1 ({round(pca\_contrib[[1,'percent']]\*100,1)}%)")) +  
 ggplot2::ylab(glue::glue("PC2 ({round(pca\_contrib[[2,'percent']]\*100,1)}%)")) +  
 ggplot2::theme(  
 panel.grid = ggplot2::element\_line(linewidth = 0.3, color = "grey95"),  
 panel.border = ggplot2::element\_rect(linewidth = 1, color = "grey70"),  
 aspect.ratio = 1  
 )  
 p  
}  
  
plot\_pca(d\_subset, conc\_adj)



## 14. Exclusion of Outlier Sample

We observed a potential outlier in the PCA. After a detailed review, it became apparent that this outlier was caused by a technical error during sample extraction. We therefore exclude this sample from the dataset and re-plot the PCA to confirm its removal and inspect the updated data structure.

d\_subset\_filt <- d\_subset |> filter(sample\_id != "LT\_batch6\_51")  
plot\_pca(d\_subset\_filt, conc\_adj)



## 15. Final dataset

We now export the final, drift/batch-corrected concentration data as a wide CSV table, ready for downstream analysis.

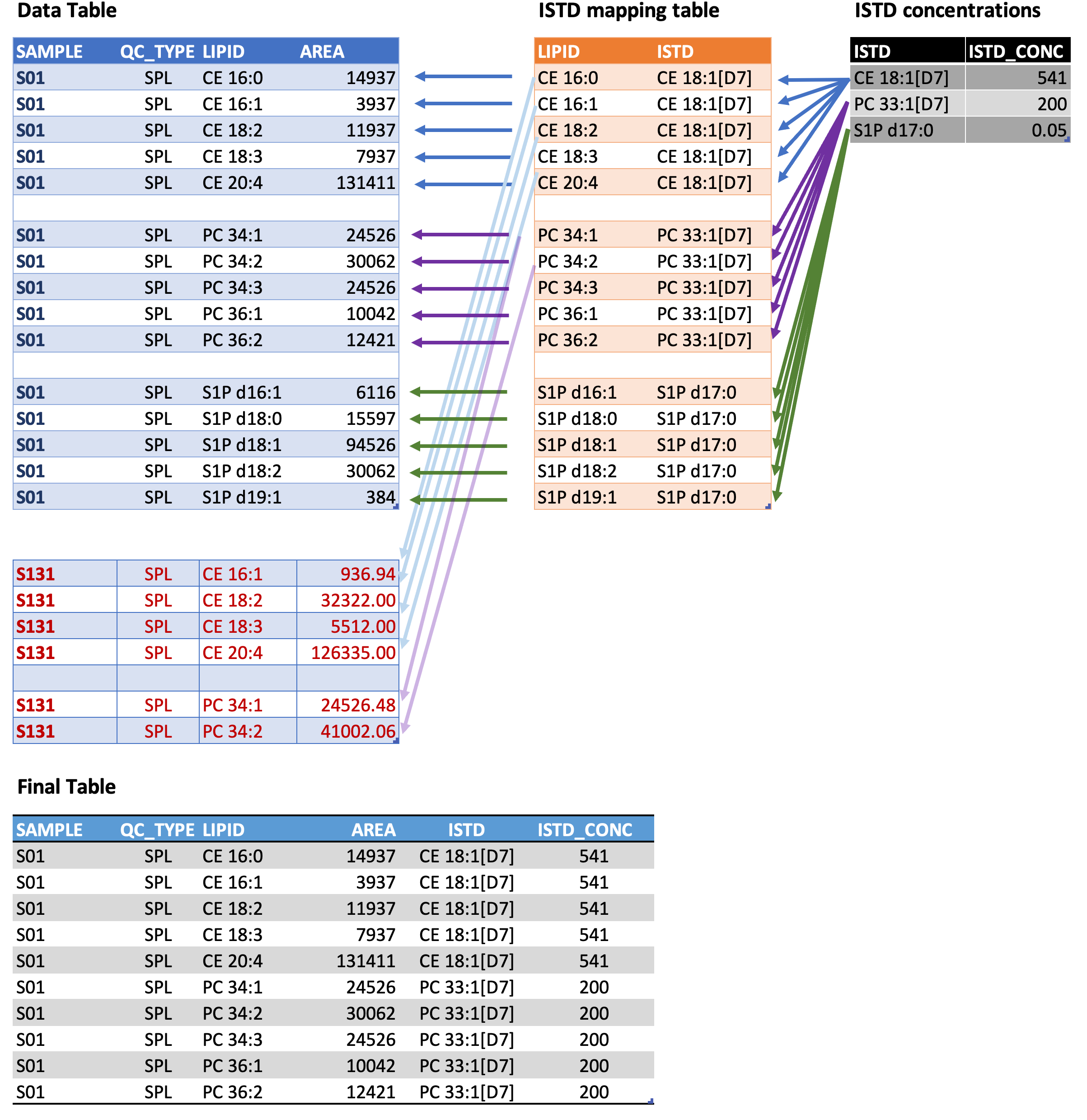
# QC filter data  
d\_final <- d\_subset\_filt |>   
 filter(qc\_type == "SAMPLE", !str\_detect(lipid, "ISTD")) |>   
 pivot\_wider(id\_cols = c(sample\_id, qc\_type), names\_from = "lipid", values\_from = "conc\_adj")   
  
write\_csv(d\_final, here("output/results-tg-subset.csv"))

# Processing the Full Dataset

In the following sections, we process the full dataset, rather than the illustrative subset used above. We will apply the same workflow to all measured lipid species.

## 16. Normalization, quantification and drif/batch correction (full dataset)

We first import a table that maps each measured lipid species with the ISTD it should be normalized with. Additionally, we import a table that defines the concentrations of the ISTD in the spiked-in ISTD mix solution. Then, we join the tables together as shown in Figure 3, then group the table into ISTD groups and proceed with normalization and quantification as described in section 8. Drift- and batch correction is performed as in section 10.



**Figure 3:** Joining of mapped ISTD and ISTD concentrations

d\_istd\_map <- read\_csv(  
 file = here("data/ISTD\_mapping.csv"),  
 col\_names = TRUE, trim\_ws = TRUE, col\_types = "c")  
  
d\_istd\_conc <- read\_csv(  
 file = here("data/ISTD\_conc.csv"),  
 col\_names = TRUE, trim\_ws = TRUE, col\_types = "c")  
  
d\_full <- d\_full |>   
 left\_join(d\_istd\_map, by = c("lipid")) |>   
 left\_join(d\_istd\_conc, by = c("istd")) |>   
 mutate(is\_istd = (lipid == istd)) |>   
 group\_by(istd, sample\_id) |>   
 mutate(  
 norm\_area = area/area[is\_istd],  
 concentration = norm\_area \* istd\_conc\_nmolar / 1000 \* 4.5 / 10) |>   
 ungroup()  
  
  
# Log-transform concentration values to make loess more robust  
d\_full$conc\_log <- log2(d\_full$concentration ) # log-transform concentration  
  
# Calculate loess fit for each lipid in each batch  
d\_full <- d\_full |>   
 group\_by(lipid, batch\_id) |>   
 nest() |>  
 mutate(y\_predicted = purrr::map(data, \(x) get\_loess(x))) |>   
 unnest(cols = c(data, y\_predicted))  
  
# Smooth study samples based on the loess fit of the QCs and   
# and transform values back to concentration scale  
d\_full <- d\_full |>  
 group\_by(lipid, batch\_id) |>   
 mutate(  
 y\_predicted = y\_predicted - median(y\_predicted, na.rm = TRUE),  
 conc\_adj = 2^(conc\_log - y\_predicted)) |>   
 ungroup()  
  
# Between-batch median-centering  
  
d\_full <- d\_full |>   
 dplyr::group\_by(lipid, batch\_id) |>   
 dplyr::mutate(  
 conc\_adj = conc\_adj/median(conc\_adj[qc\_type == "QC"], na.rm = TRUE)  
 ) |>   
 dplyr::ungroup()

## 17. Inspecting normalized data (full dataset)

As in Section 9, we plot the data as scatter plots. Because the full dataset includes many lipid species, we export a multi-page PDF containing plots for all lipids. We first define a reusable plotting function, then generate PDFs for both raw peak areas and final concentrations.

# We define the function  
  
runscatter <- function(data, var){  
 data <- data |> filter(!str\_detect(qc\_type, "BLANK|RQC"))  
 plot\_page <- function(data, nrows, ncols){  
 ggplot(data, aes(x=run\_no, y={{var}})) +   
 geom\_point(aes(colour = qc\_type, fill = qc\_type, shape = qc\_type),  
 size = 1, alpha =0.7, stroke = 0.3) +  
 facet\_wrap(vars(lipid), ncol = ncols, nrow = nrows, scales="free\_y") +  
 geom\_smooth(data= subset(data, qc\_type=="BQC"), aes(x=run\_no, y={{var}}),   
 method = "loess", span = 0.75, formula = y ~ x, se = FALSE,  
 na.rm = TRUE, color = "brown3", linewidth = .7)+  
 scale\_shape\_manual(na.value = NA, values = qc\_shapes) +  
 scale\_fill\_manual(values = qc\_colors, na.value = NA) +  
 scale\_colour\_manual(values = qc\_colors, na.value = NA) +  
 scale\_x\_continuous(breaks = seq(0, max(d\_full$run\_no), by = 100 )) +   
 scale\_y\_continuous(limits = c(0, NA)) +   
 theme\_bw(base\_size = 8)   
 }  
 rows\_page = 5  
 columns\_page = 5  
 #get a table with page numbers for each lipid species  
 d\_pages <- data |> select(lipid) |> distinct() |>   
 mutate(page\_no = ceiling(row\_number() / (rows\_page \* columns\_page)))  
   
 #plot each page from a nested table  
 d\_plots <- data |>  
 filter(!str\_detect(qc\_type, "BLK|RQC"), !str\_detect(lipid, "ISTD")) |>   
 left\_join(d\_pages, by = "lipid") |>  
 group\_by(page\_no) |>   
 nest() |>   
 mutate(plt = map(data, ~ plot\_page(., rows\_page, columns\_page)))  
   
 # Save pages to a PDF.   
 pdf(file = here(paste0("output/runscatter\_", quo\_name(enquo(var)), ".pdf")),  
 onefile = TRUE, width = 280/25.4, height = 180/25.4)  
 #d\_plots$plt   
 invisible(purrr::walk(d\_plots$plt, print)) # use this to prevent printing of index  
 dev.off()  
}  
# and run it twice, plotting raw areas and concentrations  
  
runscatter(d\_full, area)  
#> quartz\_off\_screen   
#> 2  
runscatter(d\_full, conc\_adj)  
#> quartz\_off\_screen   
#> 2

## 18. QC-filtering of dataset

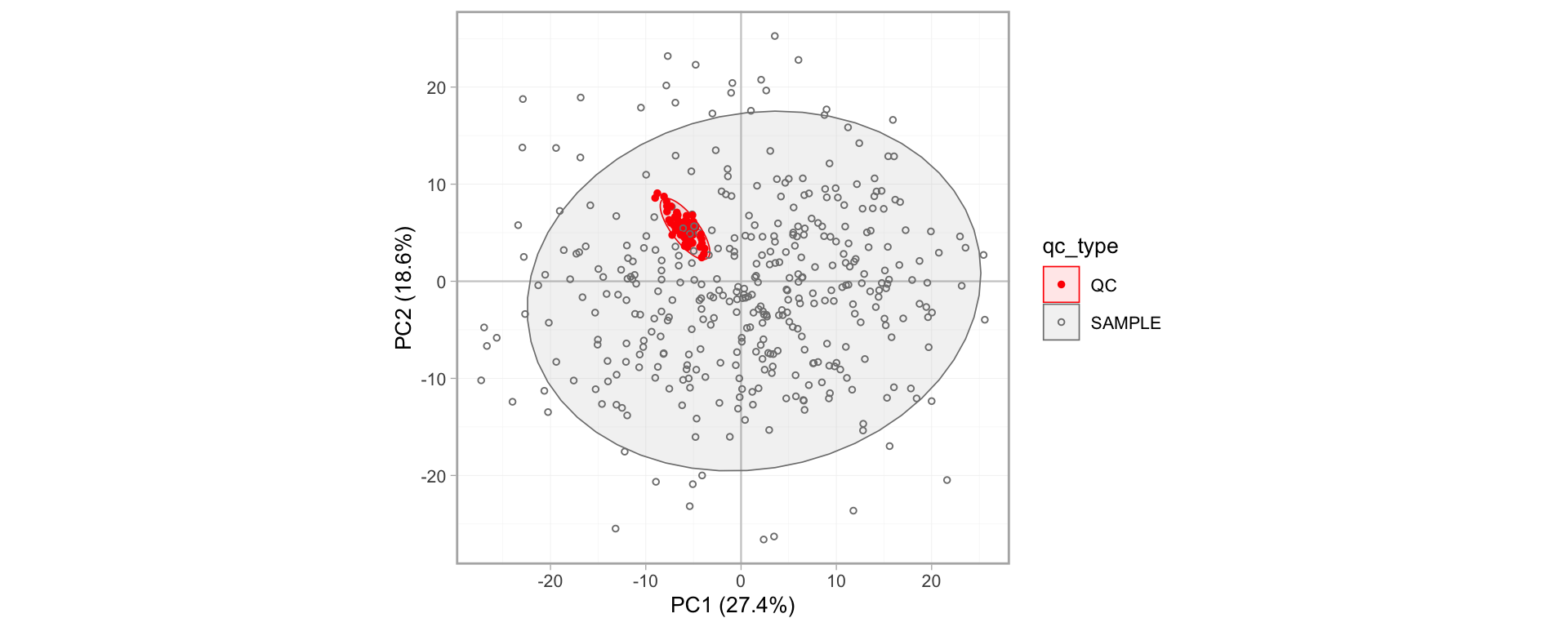
To evaluate analysis quality and filter the data, we calculate a QC metric for each lipid species and apply QC filtering using %CV < 25% and a signal-to-blank ratio > 10 as criteria. You can modify these parameters and see how it affects the number of species that pass/fail the filtering step. Before we calculate the QC matrics, we remove that outlier sample identified in Section 14.

cv <- function(x) sd(x, na.rm = TRUE)/mean(x, na.rm = TRUE) \* 100  
  
d\_full <- d\_full |> filter(sample\_id != "LT\_batch6\_51")  
  
# Get QC metrics  
d\_qc <- d\_full|>   
 group\_by(lipid) |>   
 summarise(  
 signal\_blank\_ratio = median(area[qc\_type == "SAMPLE"], na.rm = TRUE) /   
 median(area[qc\_type == "BLANK"], na.rm = TRUE),  
 cv\_qc\_area = cv(area[qc\_type == "QC"]),  
 cv\_qc\_concadj = cv(conc\_adj[qc\_type == "QC"]),   
 cv\_sample\_area = cv(area[qc\_type == "SAMPLE"]),  
 cv\_sample\_concadj = cv(conc\_adj[qc\_type == "SAMPLE"]),   
 mean\_conc = mean(conc\_adj[qc\_type == "SAMPLE"], na.rm = TRUE)   
 )  
  
# Filter the QC metrics  
d\_qc <- d\_qc |>   
 mutate(   
 qc\_pass =   
 cv\_qc\_concadj < 25 &   
 signal\_blank\_ratio > 10 &  
 !str\_detect(lipid, "ISTD"))  
  
cat(paste0("QC filtering: ", nrow(d\_qc[d\_qc$qc\_pass, ]), " of ", nrow(d\_qc), " species passed QC"))  
#> QC filtering: 319 of 413 species passed QC  
  
# QC-filter dataset  
d\_filtered <- d\_full |>   
 semi\_join(d\_qc, by = "lipid")

## 19. Principal Component Analysis (PCA)

As in Section 13, we check the final data using PCA plots

plot\_pca(d\_filtered, conc\_adj)



## 20. Final Dataset

We now save the final, drift-corrected and QC-filtered concentration data as a wide CSV table, which then will be used for Part 2. Have fun! :)

# QC filter data  
d\_final <- d\_filtered |>   
 filter(qc\_type == "SAMPLE", !str\_detect(lipid, "ISTD")) |>   
 pivot\_wider(id\_cols = c(sample\_id, qc\_type), names\_from = "lipid", values\_from = "conc\_adj")   
  
write\_csv(d\_final, here("output/results-full.csv"))