INTENT_Appendix

Supplementary tables and figures for data analysis

```
suppressPackageStartupMessages({

library(dplyr)
library(readr)
library(purrr)
library(lubridate)
library(knitr)

})

# Define 'safe' versions of the conversion functions
safe_date_conversion <- function(date_string) {
  # 'possibly' will create a wrapper around 'ymd_hms' that returns NA_real_ in case of error
  date_conversion_function <- possibly(ymd_hms, otherwise = NA_real_)
  date_conversion_function(date_string)
}

safe_integer_conversion <- function(int_string) {
  # 'possibly' will create a wrapper around 'as.integer' that returns NA_integer_ in case of error
  integer_conversion_function <- possibly(as.integer, otherwise = NA_integer_)
  integer_conversion_function(int_string)
}


td <- read_csv("combined_test.csv", col_types = cols())

td <- td %>%
  mutate(row_number = row_number()) %>%
  select(row_number, everything())


original_data <- td

# Read the data from CSV
td <- td  %>%
  mutate(
    # Attempt to convert the 'PreHbDate' column
    PreHbDate = safe_date_conversion(PreHbDate),
    # Attempt to convert the 'PreHb' column
    PreHb = safe_integer_conversion(PreHb)
  ) %>%
  # Removing rows with NA (which were failed conversions)
```

```r
  filter(!is.na(PreHb), !is.na(PreHbDate))


# Define the vector of values you want to exclude
#excluded_groupings <- c("Apheresis", "OR", "OP/MDU", "PMH OP", "HD", "PM OP", "PMCC", "TWH Arth", "Oth


# Mutate the data to add the new columns
td <- td %>%
  mutate(
    PreHb_Time = hour(PreHbDate), # Extracts the hour
    PreHb_Date = as.Date(PreHbDate) # Extracts the date
  )

cleaned_data <- td


library(dplyr)

td <- td %>%
  mutate(Grouping_Mapping = case_when(
    Groupings == "PMH OP" ~ "Outpatients",
    Groupings == "PMH IP" ~ "Inpatient Oncology",
    Groupings == "TGH MIP" ~ "General Medicine",
    Groupings == "OP/MDU" ~ "Red Cell Disorders",
    Groupings == "TWH MIP" ~ "General Medicine",
    Groupings == "HD" ~ "Dialysis",
    Groupings == "Apheresis" ~ "Apheresis",
    Groupings == "ER" ~ "Emergency",
    Groupings == "PMCC" ~ "Cardiology",
    Groupings == "MOTP" ~ "Transplant",
    Groupings %in% c("MedImaging", "Medimaging") ~ "Radiology",
    Groupings == "OR" ~ "Operating Room",
    Groupings %in% c("TGH SW", "TWH SW", "TWH Arth") ~ "Surgery",
    Groupings %in% c("TGH ICU", "TWH ICU") ~ "Critical Care",
    Groupings == "KNSP" ~ "Neurology",
    TRUE ~ "Other"  # This will be the default for all other values not matched above
  ))



# Add 'Time_of_Day' based on 'PreHbDate'
td <- td %>%
  mutate(
    Time_of_Day = if_else(hour(PreHbDate) >= 8 & hour(PreHbDate) < 18, "Day", "Night"),
    Month_of_Year = format(PreHbDate, "%B") # Extracting the month name from 'PreHbDate'
  )

td <- td %>%
  mutate(
    PreHb_Date = as.Date(PreHb_Date),  # if not already in Date format
  )
```

```r
# Print the first few rows of the modified dataframe to check new columns
print(head(td))
```

```
## # A tibble: 6 x 13
##   row_number Site  Product IssueDate     PreHb PreHbDate           IssueLocation
##        <int> <chr> <chr>   <chr>         <int> <dttm>              <chr>
## 1          1 PMH   RBC     2016-03-22 0~    80 2016-03-24 12:16:00 Urgent Care ~
## 2          2 PMH   RBC     2016-01-01 0~    70 2016-01-03 12:58:00 PMH 14C Auto~
## 3          3 PMH   RBC     2016-01-01 0~    69 2016-01-03 10:50:00 PMH 15A Leuk~
## 4          4 PMH   RBC     2016-01-01 0~    74 2016-01-03 16:42:00 PMH 17A Brea~
## 5          5 PMH   RBC     2016-01-01 0~    74 2016-01-03 16:42:00 PMH 17A Brea~
## 6          6 PMH   RBC     2016-01-01 0~    36 2016-01-03 01:07:00 PMH 15A Leuk~
## # i 6 more variables: Groupings <chr>, PreHb_Time <int>, PreHb_Date <date>,
## #   Grouping_Mapping <chr>, Time_of_Day <chr>, Month_of_Year <chr>
```

```r
# excluded_groupings <- c("Outpatients")
#
# # Filter the data
# td <- td %>%
#   filter(!Grouping_Mapping %in% excluded_groupings)  # The '!' negates the statement, so it filters o


# Add the 'Weekend' column
td <- td %>%
  mutate(Weekend = if_else(wday(PreHb_Date) %in% c(1, 7), 1, 0)) # wday returns the day of week (Sunday

# Add the 'July' column
td <- td %>%
  mutate(July = if_else(month(PreHb_Date) == 7, 1, 0)) # month() returns the month

# Add the 'Night' column
td <- td %>%
  mutate(Night = if_else(((PreHb_Time) >= 17) | ((PreHb_Time) < 8), 1, 0)) # checking if it's night

# View the first few rows of the updated dataframe to verify the changes
print(head(td))
```

```
## # A tibble: 6 x 16
##   row_number Site  Product IssueDate     PreHb PreHbDate           IssueLocation
##        <int> <chr> <chr>   <chr>         <int> <dttm>              <chr>
## 1          1 PMH   RBC     2016-03-22 0~    80 2016-03-24 12:16:00 Urgent Care ~
## 2          2 PMH   RBC     2016-01-01 0~    70 2016-01-03 12:58:00 PMH 14C Auto~
## 3          3 PMH   RBC     2016-01-01 0~    69 2016-01-03 10:50:00 PMH 15A Leuk~
## 4          4 PMH   RBC     2016-01-01 0~    74 2016-01-03 16:42:00 PMH 17A Brea~
## 5          5 PMH   RBC     2016-01-01 0~    74 2016-01-03 16:42:00 PMH 17A Brea~
## 6          6 PMH   RBC     2016-01-01 0~    36 2016-01-03 01:07:00 PMH 15A Leuk~
## # i 9 more variables: Groupings <chr>, PreHb_Time <int>, PreHb_Date <date>,
## #   Grouping_Mapping <chr>, Time_of_Day <chr>, Month_of_Year <chr>,
## #   Weekend <dbl>, July <dbl>, Night <dbl>
```

```r
# Filter td as per your criteria
specific_groupings <- c("Outpatients", "Apheresis", "Red Cell Disorders", "Radiology", "Operating Room"
```

```r
temp_td2 <- td %>%
  filter((Grouping_Mapping %in% specific_groupings))



# Calculating the numbers based on the previous explanation
N1 <- nrow(original_data)
N2 <- nrow(cleaned_data)
Excluded_NA <- N1 - N2
N3 <- nrow(temp_td2)
Excluded_Groupings <- N2 - N3

# Creating a summary table
summary_table <- tibble(
  Step = c("Initial Data Import", "Conversion and NA Exclusion", "Feature Extraction", "Exclusion of Sp
  Description = c("Imported dataset and added row numbers.",
                  "Processed 'PreHbDate' and 'PreHb' for data type conversions and excluded rows with N
                  "Added new features and mapped groupings to more descriptive names.",
                  "Excluded rows with specific groupings like 'Outpatients', 'Apheresis', etc.",
                  "Added 'Weekend', 'July', and 'Night' features for categorization."),
  Rows_Start = c(NA, N1, N2, N2, N3),
  Rows_Excluded = c(NA, Excluded_NA, NA, Excluded_Groupings, NA),
  Rows_End = c(N1, N2, N2, N3, N3)
)

# Print the table
kable(summary_table, format = "markdown", caption = "Data Cleaning Process Summary")
```

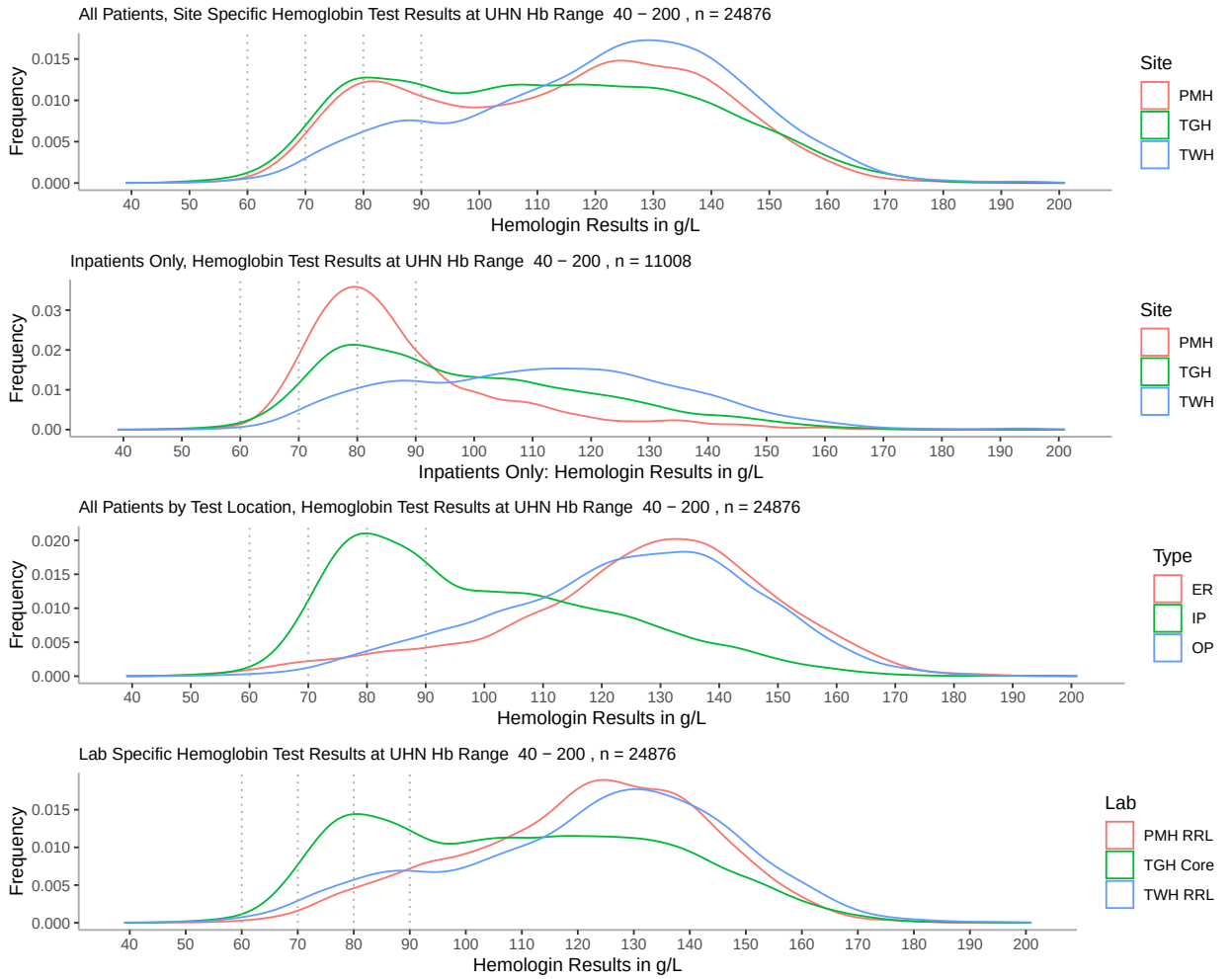
Table 1: Data Cleaning Process Summary

Step	Description	Rows_Start	Rows_Excluded	Rows_End
Initial Data Import	Imported dataset and added row numbers.	NA	NA	257435
Conversion and NA Exclusion	Processed ‘PreHbDate’ and ‘PreHb’ for data type conversions and excluded rows with NAs.	257435	45563	211872
Feature Extraction	Added new features and mapped groupings to more descriptive names.	211872	NA	211872
Exclusion of Specific Groupings	Excluded rows with specific groupings like ‘Outpatients’, ‘Apheresis’, etc.	211872	77566	134306
Final Feature Additions	Added ‘Weekend’, ‘July’, and ‘Night’ features for categorization.	134306	NA	134306

```r
suppressPackageStartupMessages({


library(dplyr)
library(tidyverse)
library(rvest)
library(ggplot2)
library(data.table)
library(gridExtra)

})
```

```r
# Specify the path to the actual file, not just the directory
file_path <- "PureHb_Analysis/HPM_hb_copy.csv"

# Check if the file exists
if (!file.exists(file_path)) {
  stop("File does not exist: ", file_path)
}

# Read the .csv file
hb_dat <- read_csv(file_path)

# Convert to data frame, if necessary (read_csv returns a tibble, which is a type of data frame)
hb_df <- as.data.frame(hb_dat)

# Now hb_df is a data frame with your .csv data



colnames(hb_df) <- c("RequestNum", "Barcode", "Lab", "MRN", "CollectionT", "ReceiptT", "ResultT", "Sampl

hbde <- hb_df %>% filter(!(is.na(Hb)))  #remove rows with pre-hb = null


hbdf <-hbde #for beta testing, subset of data

hbdf <- hbdf %>% separate(CollectionT, c("CollectionDate", "CollectionTime"), " ", fill="right") #separ
hbdf <- hbdf %>% separate(ReceiptT, c("ReceiptDate", "ReceiptTime"), " ", fill="right") #separate date
hbdf <- hbdf %>% separate(ResultT, c("ResultDate", "ResultTime"), " ", fill="right") #separate date and

hbdf <- hbdf[c("RequestNum", "Lab", "MRN", "CollectionDate", "CollectionTime", "SampleLocation", "Site"

#hbdf <- hbdf %>% filter(Type=="IP") #filter only inpatient

hbdf <- hbdf %>% filter(!(Site %in% c("TRI", "REF"))) #filter only inpatient

hbdf <- hbdf %>% filter(!(Type %in% c("GRA","BD"))) #filter only inpatient


hb_upper= 200
hb_lower= 40

p1 <- ggplot(hbdf, aes(x=Hb)) +
  geom_density(aes(colour=Site), bins = hb_upper, alpha=1, position="dodge") + ggtitle (paste("All Pati


p2 <- ggplot(hbdf, aes(x=Hb)) +
  geom_density(aes(colour=Type), bins = hb_upper, alpha=1, position="dodge") + ggtitle (paste("All Pati


p3 <- ggplot(hbdf, aes(x=Hb)) +
  geom_density(aes(colour=Lab), bins = hb_upper, alpha=1, position="dodge") + ggtitle (paste("Lab Speci
```

```
hbdf <- hbdf %>% filter(Type=="IP") #filter only inpatient

p4 <- ggplot(hbdf, aes(x=Hb)) +
  geom_density(aes(colour=Site), bins = hb_upper, alpha=1, position="dodge") + ggtitle (paste("Inpatien


grid.arrange(p1, p4, p2, p3, nrow = 4)
```



Appendix Figure: Density plots depicting all hemoglobin test results. Vertical lines highlighting thresholds of 70, 80, and 90 g/L. Plots depicting hospital sites (PMH, TGH, TWH), patient populations (All Patients, Inpatients), Test Location, and Laboratory sites (PMH RRL, TGH Core, TWH RRL)

```
suppressPackageStartupMessages({

library(dplyr)
})

# Assuming hbdf is your dataframe and Hb is your running variable

# Create a dataframe with the count of transfusions for each Hb level
hb_counts <- hbdf %>%
```

```r
  group_by(Hb) %>%
  summarise(Count = n()) %>%
  ungroup()

# Define your thresholds
thresholds <- c(69.5, 79.5, 89.5)

# Define the window
window <- 5

# Initialize a data frame to store results
results <- data.frame(
  Threshold = numeric(),
  Coefficients = numeric(),  # Coefficients of the treatment effect
  P_Value = numeric(),       # P-values of the treatment effect
  R_Squared = numeric()      # R-squared values of the models
)

for (threshold in thresholds) {

  # Filter the data to only include observations within the window around the threshold
  hb_counts_window <- hb_counts %>%
    filter(Hb >= threshold - window & Hb <= threshold + window)

  # Create a binary treatment variable based on the threshold
  hb_counts_window$Treatment <- ifelse(hb_counts_window$Hb > threshold, 1, 0)

  # Create the distance to the threshold variable
  hb_counts_window$Distance <- hb_counts_window$Hb - threshold

  # Fit the quadratic regression model
  model <- lm(Count ~ Treatment + Distance + I(Distance^2), data = hb_counts_window)

  # Get the summary of the model
  summary_model <- summary(model)

  # Extract the treatment effect and its p-value
  treatment_effect <- coef(summary_model)["Treatment", "Estimate"]
  p_value <- coef(summary_model)["Treatment", "Pr(>|t|)"]

  # Store the results
  results <- rbind(results, data.frame(
    Threshold = threshold,
    Coefficients = treatment_effect,
    P_Value = p_value,
    R_Squared = summary_model$r.squared
  ))
}


p <- ggplot() +
  theme_void() +
  theme(panel.background = element_blank())
```

```r
# Convert the 'result' data frame to a grid table and add it to the plot
# The tableGrob function creates a table-like element from the data frame
table <- tableGrob(results, theme = ttheme_minimal())

# Combine the empty plot and table
final_plot <- p + annotation_custom(grob = table)

# Display the plot with the table
print(final_plot)
```

	Threshold	Coefficients	P_Value	R_Squared
1	69.5	3.05000000000003	0.874237843734911	0.958065661894491
2	79.5	−20.7999999999999	0.384740855598452	0.357255604581135
3	89.5	−28.2	0.167236772512247	0.823651469314539

Appendix Table: Regression discontinuity analysis testing the effect of leading digit thresholds on all hemoglobin results at University Health Network

```r
suppressPackageStartupMessages({

library(dplyr)
library(lubridate)
library(gridExtra)
library(ggplot2)
})

# Assuming 'td' is your data frame

# 1. Summary for all rows
overall_summary <- td %>%
  summarise(
    Count = n(),
    Mean_PreHb = mean(PreHb, na.rm = TRUE),
    StDev_PreHb = sd(PreHb, na.rm = TRUE)
  )

# 2. Summary by 'Site'
summary_by_site <- td %>%
  group_by(Site) %>%
  summarise(
    Count = n(),
    Mean_PreHb = mean(PreHb, na.rm = TRUE),
    StDev_PreHb = sd(PreHb, na.rm = TRUE)
  )

# 3. Summary by 'Groupings'
summary_by_groupings <- td %>%
  group_by(Grouping_Mapping) %>%
  summarise(
```

```
    Count = n(),
    Mean_PreHb = mean(PreHb, na.rm = TRUE),
    StDev_PreHb = sd(PreHb, na.rm = TRUE)
  )

# 4. Summary by year within 'IssueDate'
summary_by_year <- td %>%
  mutate(Year = year(IssueDate)) %>% # Extract year from 'IssueDate'
  group_by(Year) %>%
  summarise(
    Count = n(),
    Mean_PreHb = round(mean(PreHb, na.rm = TRUE),0),
    StDev_PreHb = round(sd(PreHb, na.rm = TRUE),1)
  )

# Add an identifier for each summary
overall_summary$Source <- "Overall"
summary_by_site$Source <- "Hospital Site"
summary_by_groupings$Source <- "Clinical Grouping"
summary_by_year$Source <- "Year"

# Combine all summaries into one data frame
combined_summary <- bind_rows(
  overall_summary,
  summary_by_site,
  summary_by_groupings,
  summary_by_year
)

# If your summaries have different columns, you might want to select only common ones
combined_summary <- combined_summary %>%
  select(Source, everything())  # This reorders the 'Source' column as the first one

#print(combined_summary)

# Assuming your data frame is named 'data_frame'
result <- combined_summary %>%

  # Combine 'Site', 'Grouping_Mapping', and 'Year' into 'Subgroup'
  mutate(Subgroup = ifelse(is.na(Site) | Site == "<NA>", "", as.character(Site)),
         Subgroup = ifelse(is.na(Grouping_Mapping) | Grouping_Mapping == "<NA>", Subgroup, paste(Subgrou
         Subgroup = ifelse(is.na(Year) | Year == "<NA>", Subgroup, paste(Subgroup, as.character(Year), 

  # Select and rename the columns, excluding the original 'Site', 'Grouping_Mapping', and 'Year' column
  select(Category = Source, Subgroup, Count, Mean_PreHb, StDev_PreHb) %>%

  # Optional: Remove any leading/trailing whitespace that may have resulted from the concatenation proc
  mutate(Subgroup = trimws(Subgroup))

# View the resulting tibble/data frame
#print(result)
```

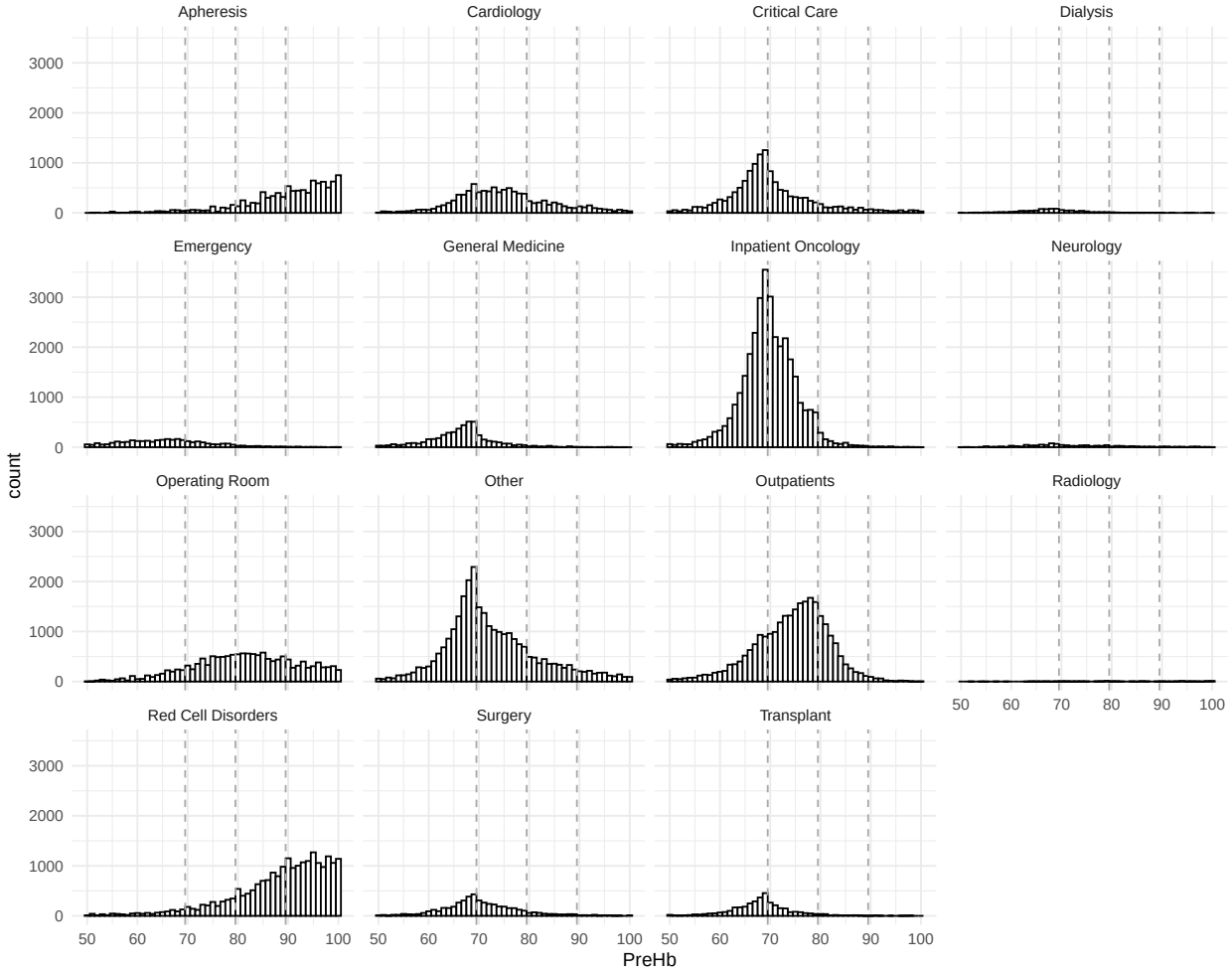
```
p <- ggplot() +
  theme_void() +
  theme(panel.background = element_blank())

# Convert the 'result' data frame to a grid table and add it to the plot
# The tableGrob function creates a table-like element from the data frame
table <- tableGrob(result, theme = ttheme_minimal())

# Combine the empty plot and table
final_plot <- p + annotation_custom(grob = table)

# Display the plot with the table
print(final_plot)
```

	Category	Subgroup	Count	Mean_PreHb	StDev_PreHb
1	Overall		211872	81.12245	17.922002
2	Hospital Site	PMH	60875	71.55326	7.191791
3	Hospital Site	TGH	138666	85.71468	19.090473
4	Hospital Site	TWH	12331	76.72200	21.369956
5	Clinical Grouping	Apheresis	22304	102.09850	13.721342
6	Clinical Grouping	Cardiology	10542	76.57304	12.589146
7	Clinical Grouping	Critical Care	13819	70.58796	11.390139
8	Clinical Grouping	Dialysis	981	67.96738	6.901268
9	Clinical Grouping	Emergency	3775	63.94543	13.202720
10	Clinical Grouping	General Medicine	5125	66.09054	8.265831
11	Clinical Grouping	Inpatient Oncology	33462	69.62862	5.888933
12	Clinical Grouping	Neurology	1103	73.40163	12.322831
13	Clinical Grouping	Operating Room	21641	93.96410	22.054969
14	Clinical Grouping	Other	30224	74.74381	14.227812
15	Clinical Grouping	Outpatients	26091	74.03771	7.933498
16	Clinical Grouping	Radiology	261	85.98467	17.165477
17	Clinical Grouping	Red Cell Disorders	33785	95.15317	13.991991
18	Clinical Grouping	Surgery	5005	70.57862	8.651346
19	Clinical Grouping	Transplant	3754	68.01652	7.234472
20	Year	2016	32900	81.00000	17.700000
21	Year	2017	32385	82.00000	18.000000
22	Year	2018	35015	79.00000	17.000000
23	Year	2019	11618	80.00000	16.900000
24	Year	2020	54307	82.00000	18.200000
25	Year	2021	45647	82.00000	18.500000

Appendix Table: Pre-transfusion hemoglobin results for all patients and by subgroups (hospital site, clinical grouping, and year of test)

```
suppressPackageStartupMessages({

library(ggplot2)
library(dplyr)
```

```
})

# Step 1: Filter the data
filtered_td <- td %>%
  filter(PreHb >= 50 & PreHb <= 100)

# Step 2: Plotting with ggplot2
p <- ggplot(filtered_td, aes(x = PreHb)) +
  geom_histogram(binwidth = 1, color = "black", fill = "white") +  # you can adjust binwidth as needed
  facet_wrap(~Grouping_Mapping) +  # creates a grid of plots by 'Groupings'
  theme_minimal()  # optional theme

# Step 3: Adding vertical lines
p <- p + geom_vline(aes(xintercept = 69.5), linetype = "dashed", color = "darkgray", size = 0.5) +
  geom_vline(aes(xintercept = 79.5), linetype = "dashed", color = "darkgray", size = 0.5) +
  geom_vline(aes(xintercept = 89.5), linetype = "dashed", color = "darkgray", size = 0.5)

# Show the plot
print(p)
```



Appendix Figure: Bar graphs depicting pre-transfusion hemoglobin distributions and vertical lines at thresholds of 70, 80, and 90 g/L across clinical groupings

```r
suppressPackageStartupMessages({

library(ggplot2)
library(dplyr)
library(gridExtra)
library(readr)
library(rdd)
library(purrr)
library(magrittr)  # for the %$% operator, although it's not used if you follow the correction below
library(grid)  # Needed for the grid.draw function

})

# Define your threshold

thresholds <- c(69.5, 79.5, 89.5, 70.5, 80.5, 90.5)

# Initialize a data frame to store results
results <- data.frame(
  Threshold = numeric(),
  Effect_Size = numeric(),
  P_Value = numeric(),
  Sample_Size = integer()
)

for (t in thresholds) {

  thold <- t

# Step 1: Filter the data
temp_td <- td %>%
  filter(PreHb >= thold-5 & PreHb <= thold+5)

# Preparing for RDD: Creating a smaller data frame and threshold variable
hb_t_df <- temp_td %>%
  group_by(PreHb) %>%
  summarise(Count = n()) %>%
  mutate(threshold = ifelse(PreHb >= thold, 1, 0))

# Perform the RDD analysis
lm_quadratic <- hb_t_df %>%
  mutate(distance = PreHb - thold) %$%  # Distance from the threshold
  lm(Count ~ threshold + distance + I(distance^2) + threshold:distance + threshold:I(distance^2))

# Extract results
pval <- round(summary(lm_quadratic)$coefficients["threshold",4], 4)
effsize <- round(lm_quadratic$coefficients["threshold"], 2)
analytic_sample_size <- nrow(temp_td)

# Store results
results <- rbind(results, data.frame(
  Threshold = thold,
  Effect_Size = effsize,
```

```
  P_Value = pval,
  Sample_Size = analytic_sample_size
))
}

# View the resulting tibble/data frame
#print(result)

q <- ggplot() +
  theme_void() +
  theme(panel.background = element_blank())

# Convert the 'result' data frame to a grid table and add it to the plot
# The tableGrob function creates a table-like element from the data frame
table <- tableGrob(results, theme = ttheme_minimal())

# Combine the empty plot and table
final_plot <- q + annotation_custom(grob = table)

# Display the plot with the table
print(final_plot)
```

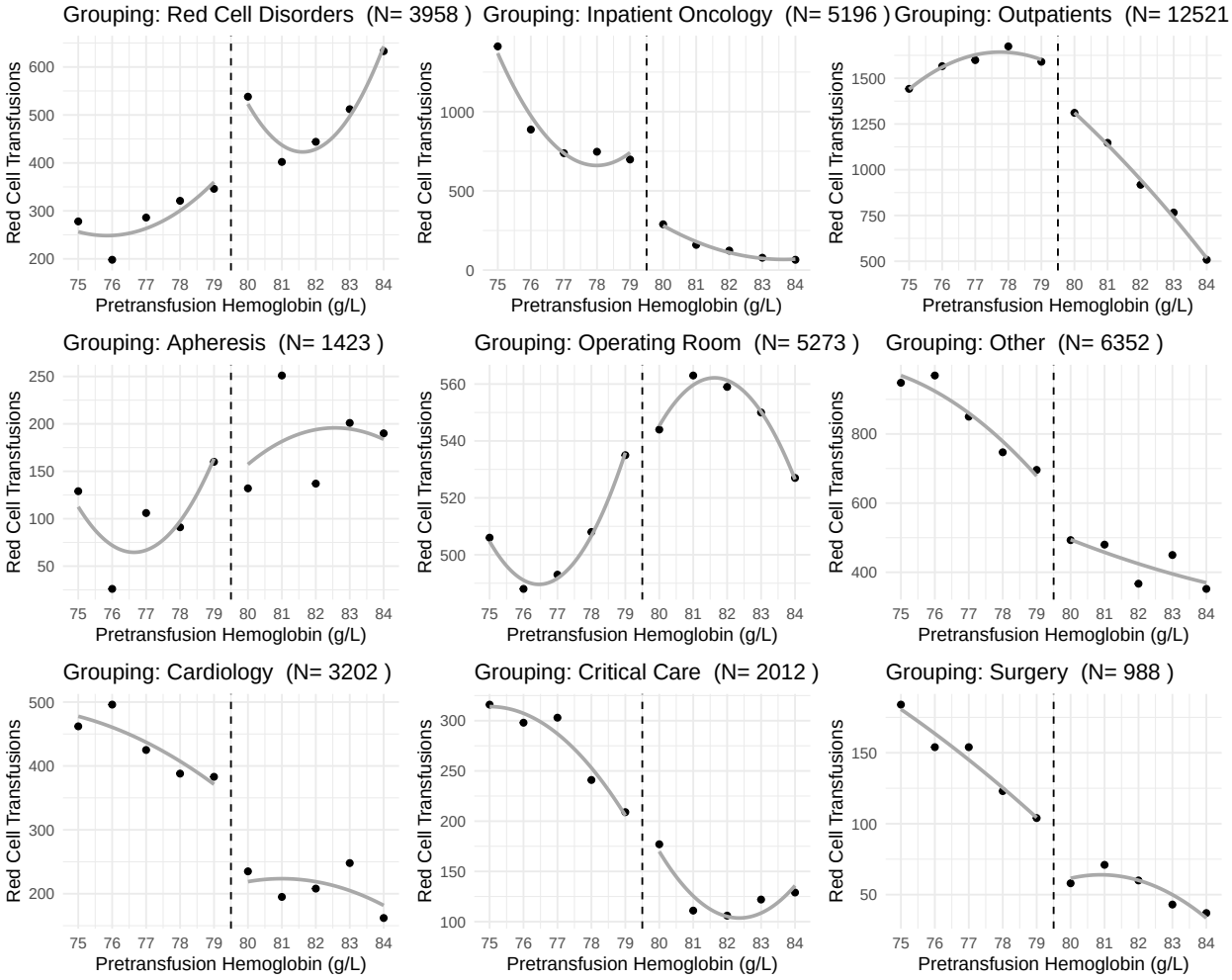
	Threshold	Effect_Size	P_Value	Sample_Size
threshold	69.5	−2801.05	0.0283	73878
threshold1	79.5	−882.45	0.0356	42560
threshold2	89.5	189.43	0.5848	24174
threshold3	70.5	−870.99	0.5781	74794
threshold4	80.5	173.78	0.68	39474
threshold5	90.5	−537.27	0.1231	23848

Appendix Table: Results of regression discontinuity analyses using the entire dataset, leading digit numbers (69, 79, 89) as thresholds and Benford Numbers (71, 81, 91) as controls")

```
suppressPackageStartupMessages({

library(ggplot2)
library(dplyr)
library(gridExtra)
library(readr)
library(rdd)
library(purrr)
library(magrittr)  # for the %$% operator, although it's not used if you follow the correction below
library(grid)  # Needed for the grid.draw function
```

```
})



library(dplyr)
library(ggplot2)
library(gridExtra)


thold <- 79.5


# Filter td as per your criteria
filtered_td <- td %>%
  filter(PreHb >= thold-5 & PreHb <= thold+5) %>%
  group_by(Grouping_Mapping) %>%
  add_count() %>%
  filter(n > 500)  # Ensuring groups have more than 500 observations

# This will store the ggplot objects
plots <- list()

# This variable is used to keep track of the plot list index
plot_index <- 1

# Loop through each unique Grouping_Mapping after filtering
for (group in unique(filtered_td$Grouping_Mapping)) {

  # Filter data for the specific group
  group_data <- filtered_td %>%
    filter(Grouping_Mapping == group) %>%
    group_by(PreHb) %>%
    summarise(Count = n()) %>%
    mutate(threshold = ifelse(PreHb >= thold, 1, 0),
           distance = PreHb - thold)  # Distance from the threshold

  # Perform the RDD analysis similar to above, per group
  lm_quadratic <- group_data %$%
    lm(Count ~ threshold + distance + I(distance^2) + threshold:distance + threshold:I(distance^2))

  # You could extract and print/store results here similar to the single analysis above

  # Create individual ggplot objects
  p <- group_data %>%
    ggplot(aes(x = PreHb, y = Count)) +
    geom_point() +
    geom_smooth(data = subset(group_data, threshold == 0),
                method = "lm",
                formula = y ~ poly(x, 2),
                se = FALSE,
                aes(group = threshold),
                color = "darkgray",
                size = 1) +
```

```
    geom_smooth(data = subset(group_data, threshold == 1),
                method = "lm",
                formula = y ~ poly(x, 2),
                se = FALSE,
                aes(group = threshold),
                color = "darkgray",
                size = 1) +
    geom_vline(xintercept = thold, linetype = "dashed", color = "black", size = 0.5) +
    labs(title = paste("Grouping:", group, " (N=", nrow(filtered_td %>%filter(Grouping_Mapping == group)
         x = "Pretransfusion Hemoglobin (g/L)",
         y = "Red Cell Transfusions") +
    scale_x_continuous(breaks = seq(floor(min(group_data$PreHb)), ceiling(max(group_data$PreHb)), by =
    theme_minimal()

  # Store the ggplot object in the list
  plots[[plot_index]] <- p

  # Increment the plot index
  plot_index <- plot_index + 1
}

# Arrange the plots in a grid (you can customize the layout as needed)
grid.arrange(grobs = plots, ncol=3)  # change the number of columns as per your preference
```



Appendix Figure: Regression discontinuity analysis graphs across clinical groupings. Subgroups with fewer than 500 transfusions have been suppressed