

# DDC Data Challenge: Data Prep

Yulia Zamriy

January 15, 2018

## Overview

This is a support file for my response to DDC Data Challenge. It contains the code for creating the main analytical dataset. Please note that I kept only those pieces of code that were explicitly used (with a few exceptions) to build final dataset. A lot of “exploratory” code has been left out for simplicity and clarity reasons. I can provide it upon request.

### Step 1. Read in the data

I decided to work with the provided sample data in response to the challenge. The data was provided in an Excel file that contained four tabs. The main tabs of interest were *Projects*, *Schedule* and *Budget*. Before processing them in R, I split them into three different csv files:

```
library(tidyverse)
library(gridExtra)

Projects <- read_csv("DDC_sampledata Projects.csv")
Schedule <- read_csv("DDC_sampledata Schedule.csv")
Budget <- read_csv("DDC_sampledata Budget.csv")

# Removing rows with all NAs
Projects <- Projects[!apply(is.na(Projects), 1, all),]
Schedule <- Schedule[!apply(is.na(Schedule[, 3:50]), 1, all),]
Budget <- Budget[!apply(is.na(Budget), 1, all),]
```

The next step would be merging three datasets. After investigating the data, I identified the key field: *Project Id*

```
str(Projects$ProjectId)

## chr [1:1000] "ACEDOS501" "ACEDOS503" "ACEDOT601" "ACEDOT602" ...

str(Schedule$ProjectId)

## chr [1:946] "ACEDOS501" "ACEDOS503" "ACSLSPCAR" "AGCOVELLO" "BED776" ...

str(Budget$project_id)

## chr [1:1000] "ACEDOS501" "ACEDOS501" "ACSBELL" "ACSBELL" "ACSBELL" ...
```

However, it appeared that Project ID in the *Budget* dataset had a slightly different format than in the other two. Hence, I made an explicit assumption that by removing “-” in Project IDs I achieve more consistency across keys without changing underlying relationships.

```
Projects$ProjectId2 <- sub("-", "", Projects$ProjectId)
Schedule$ProjectId2 <- sub("-", "", Schedule$ProjectId)
Budget$ProjectId2 <- sub("-", "", Budget$project_id)
```

Then it appeared that the three datasets are not unique on the Project ID level.

```
nrow(Projects)

## [1] 1000
length(unique(Projects$ProjectId2))

## [1] 943
nrow(Schedule)

## [1] 946
length(unique(Schedule$ProjectId2))

## [1] 944
nrow(Budget)

## [1] 1000
length(unique(Budget$ProjectId2))

## [1] 214
```

Hence, more processing was required to understand the nature of duplicates.

## Step 2. Processing *Schedule* data

Schedule data contains dates for different stages of projects. Therefore, this data is key to answering the main challenge question.

However, all the dates have been read in as character variables. I converted them to date format before doing anything else.

```
datefor <- function(x) as.Date(x, format = '%Y-%m-%d')
Schedule[,3:51] <- lapply(Schedule[,3:51], datefor)
```

As stated in the analysis overview document, I defined my key metric as number of days between project start and closeout dates. The schedule contained many different types of dates: projected, original, actual etc. After some data exploration I chose to use *original project dates* to calculate project durations. At first, I wanted to use *actual project dates*, however, there were very few projects with non-missing actual project closeout dates. It might be due to the way this sample file was generated. Original closeout date had much better coverage in the file.

The code below creates a subset of Schedule data as following:

- Remove all rows that are missing Original Project Closeout Dates
- Keep only Project Id and other Original dates related to project duration
- Calculate total project duration as well as duration of other project stages (in days)
- Create some other variables that might be helpful in the analysis (project duration split by stage, month of project start etc.)

```
ScheduleSub <- Schedule %>%
  filter(!is.na(OrgProjectCloseout)) %>%
  select(ProjectId, ProjectId2,
         OrgProjectStart,
         OrgDesignStart,
         OrgDesignCompletion,
```

```

    OrgConstructionStart,
    OrgConstructionCompletion,
    OrgProjectCloseout) %>%
mutate(
  OrgInitiationDur = as.numeric(OrgDesignStart - OrgProjectStart),
  OrgDesignDur = as.numeric(OrgConstructionStart - OrgDesignStart),
  OrgConstructionDur = as.numeric(OrgConstructionCompletion - OrgConstructionStart),
  OrgCloseoutDur = as.numeric(OrgProjectCloseout - OrgConstructionCompletion),
  OrgProjectDur = as.numeric(OrgProjectCloseout - OrgProjectStart),
  OrgInitiationDurPct = round(OrgInitiationDur / OrgProjectDur, 2),
  OrgDesignDurPct = round(OrgDesignDur / OrgProjectDur, 2),
  OrgConstructionDurPct = round(OrgConstructionDur / OrgProjectDur, 2),
  OrgCloseoutDurPct = round(OrgCloseoutDur / OrgProjectDur, 2),
  OrgProjectStartYear = format(OrgProjectStart, "%Y"),
  OrgProjectStartMonth = format(OrgProjectStart, "%m"),
  OrgDesignStartMonth = format(OrgDesignStart, "%m"),
  OrgConstructionStartMonth = format(OrgConstructionStart, "%m"),
  OrgProjectCloseoutYear = format(OrgProjectCloseout, "%Y"),
  OrgProjectCloseoutMonth = format(OrgProjectCloseout, "%m"))

nrow(ScheduleSub)

## [1] 673

```

### Step 3. Processing *Projects* data

Projects data contains descriptive information for Project IDs. There are a lot of columns containing useful information. But for this analysis I decided to focus on a selected few that seemed most relevant to me. If I had more time, I would have performed some text analysis of long text fields to identify any patterns that could be used for this assignment.

```

# Createing a subds of Projects data
ProjectsSub <- Projects %>%
  select(ProjectId, ProjectId2,
    DivisionName,
    UnitName,
    Borough,
    Sponsor,
    PhaseName,
    ProjectType,
    Priority,
    DesignContractType,
    ConstructionContractType) %>%
  mutate(UnitName = tolower(UnitName))

sapply(ProjectsSub, function(x) sum(is.na(x)))

```

##	ProjectId	ProjectId2	DivisionName
##	0	0	6
##	UnitName	Borough	Sponsor
##	4	317	73
##	PhaseName	ProjectType	Priority
##	99	86	113
##	DesignContractType	ConstructionContractType	

```
##
```

```
186
```

```
217
```

A lot of selected fields had missing data. I decided to recode missing values into “Unknown” category.

```
ProjectsSub$DivisionName[is.na(ProjectsSub$DivisionName)] <- "Unknown"
ProjectsSub$UnitName[is.na(ProjectsSub$UnitName)] <- "Unknown"
ProjectsSub$Borough[is.na(ProjectsSub$Borough)] <- "Unknown"
ProjectsSub$Sponsor[is.na(ProjectsSub$Sponsor)] <- "Unknown"
ProjectsSub$PhaseName[is.na(ProjectsSub$PhaseName)] <- "Unknown"
ProjectsSub$ProjectType[is.na(ProjectsSub$ProjectType)] <- "Unknown"
ProjectsSub$Priority[is.na(ProjectsSub$Priority)] <- "Unknown"
ProjectsSub$Priority[ProjectsSub$Priority == "Mayoral In"] <- "Mayoral Initiative"
ProjectsSub$DesignContractType[is.na(ProjectsSub$DesignContractType)] <- "Unknown"
ProjectsSub$ConstructionContractType[is.na(ProjectsSub$ConstructionContractType)] <- "Unknown"

nrow(ProjectsSub)
```

```
## [1] 1000
```

```
length(unique(ProjectsSub$ProjectId))
```

```
## [1] 945
```

Based on the above numbers, it appeared that there were till duplicate rows. Data exploration showed that most of them were complete duplicates. Hence, it was safe to remove them.

```
ProjectsSub <- ProjectsSub[!duplicated(ProjectsSub), ]
nrow(ProjectsSub)
```

```
## [1] 945
```

```
length(unique(ProjectsSub$ProjectId))
```

```
## [1] 945
```

#### Step 4. Processing *Budgets* data

This data contains budgets for various projects. However, as shown at the beginning of this document, there were only 214 unique Project IDs covered by this data. On the other hand, Projects and Schedule files contain almost a thousand of unique Project IDs. Moreover, the overlap of projects across all three files was even smaller. At the end I decided not to use the Budgets data for the analysis because I did not know the reasons behind low match rates. Was it because of how sample data was pulled? Or was it because of how budget data for projects was generated? In any case, I don't think it would be valid to draw any conclusions based on the small pool of projects with budget data.

However, since I had to process the data to make this decision, I am providing the code to prepare Budget data for the analysis. If I were to do this analysis again, I would probably requested to get a larger file with budget numbers. Another important note about Budget dataset. Unlike to others, it was not unique at Project ID, but rather on Project ID + pobject. Hence, I had to do a different type of processing for it.

The following code does the following:

- Aggregates different types of budgets by *Project ID* and *pobject* variables because each project in this file is broken out by pobject (with values corresponding to Construction, Contingency, Design, etc.)
- Aggregates different types of budgets by Project ID

```
BudgetTemp <- Budget %>%
  group_by(ProjectId2, pobject) %>%
  summarize(PlannedAmtTtl = sum(PlannedAmt),
```

```

        CommittedAmtTtl = sum(CommittedAmt),
        LiquidatedAmtTtl = sum(LiquidatedAmt),
        AvailableAmt = sum(AvailableAmt))

BudgetTemp1 <- BudgetTemp %>%
  select(ProjectId2, pobject, PlannedAmtTtl) %>%
  spread(pobject, PlannedAmtTtl, sep = ".PlanAmt.")

BudgetTemp2 <- BudgetTemp %>%
  select(ProjectId2, pobject, CommittedAmtTtl) %>%
  spread(pobject, CommittedAmtTtl, sep = ".ComAmt.")

BudgetTemp3 <- BudgetTemp %>%
  select(ProjectId2, pobject, LiquidatedAmtTtl) %>%
  spread(pobject, LiquidatedAmtTtl, sep = ".LiqAmt.")

BudgetTemp4 <- BudgetTemp %>%
  select(ProjectId2, pobject, AvailableAmt) %>%
  spread(pobject, AvailableAmt, sep = ".AvailAmt.")

BudgetAggr <- Budget %>%
  group_by(ProjectId2) %>%
  summarize(PlannedAmtTtl = sum(PlannedAmt),
            CommittedAmtTtl = sum(CommittedAmt),
            LiquidatedAmtTtl = sum(LiquidatedAmt),
            AvailableAmtTtl = sum(AvailableAmt),
            AgcyName = first(AgcyName)) %>%
  left_join(BudgetTemp1, by = "ProjectId2") %>%
  left_join(BudgetTemp2, by = "ProjectId2") %>%
  left_join(BudgetTemp3, by = "ProjectId2") %>%
  left_join(BudgetTemp4, by = "ProjectId2")

rm(list = c("BudgetTemp1", "BudgetTemp2", "BudgetTemp3", "BudgetTemp4", "BudgetTemp"))

BudgetAggr <- BudgetAggr[, !apply(is.na(BudgetAggr), 2, all)]
BudgetAggr[is.na(BudgetAggr)] <- 0

nrow(BudgetAggr)

```

```
## [1] 214
```

```
length(unique(BudgetAggr$ProjectId2))
```

```
## [1] 214
```

Here's the evidence that after merging Schedule, Projects and Budget data by Project ID, number of projects with Budget data is rather low for the analysis.

```

FullDataWBudget <- ScheduleSub %>%
  inner_join(ProjectsSub, by = "ProjectId") %>%
  left_join(BudgetAggr, by = c("ProjectId2.x" = "ProjectId2")) %>%
  filter(!is.na(OrgProjectDur))

summary(FullDataWBudget$PlannedAmtTtl)

```

```
##      Min.   1st Qu.   Median     Mean   3rd Qu.     Max.    NA's
```

```
##      7000    206250    930500    7462893    3264750    92714000      570
```

PlannedAmtTtl has 570 missing values. Missing values in this case correspond to un-matched Project IDs.

### Step 5. Creating Analytical Dataset

Since I decided not to use Budget data, I merged Schedule and Projects data by the original Project ID. I also filtered out all the rows that had missing data for Original Project Duration variable because I selected it as the main variable of interest.

```
FullData <- ScheduleSub %>%
  inner_join(ProjectsSub, by = "ProjectId") %>%
  filter(!is.na(OrgProjectDur))

FullData$ProjectId2.y <- NULL
FullData$ProjectId2 <- FullData$ProjectId2.x
FullData$ProjectId2.x <- NULL

dim(FullData)

## [1] 598  32

save(FullData, file = "DDCChallengeAnalysisData.RData")
```

My final analytical dataset contains **598** unique Project IDs.