# Correcting bias in operational expense planning

HarvardX PH125.9x Data Science: Capstone Project

edxLux

31st January 2022

## Contents

## Disclaimer

This report has been developed as part of the HarvardX PH125.9x Data Science: Capstone Project. I hereby declare that the report is my own original work and has not been submitted before to any institution for assessment purposes. The copyright and ownership of the original work remains with the author of this report.

# 1 Introduction

## 1.1 Project background and objective

Companies throughout the world require liquidity (i.e. cash) to perform operations and it is usually the task of the Finance department to ensure that sufficient liquidity is available to the company. However, providing liquidity also comes at a cost, as investors will be expecting profits and creditors will ask for interests. Too much liquidity therefore generates unnecessary cost while to little liquidity will inhibit the company's operations.

In order to arrange adequate liquidity instruments, Finance departments therefore require a sufficiently accurate prediction of a company's upcoming spending. Such a prediction usually relies on the resource consumption plan of the company's operational departments.

> "It is difficult to make predictions, especially about the future" (Steincke 1948)
>
> "No plan of operations extends with any certainty beyond the first encounter with the main enemy forces" (Moltke and Generalstab 1892)

The spending plan of an operational department in turn relies on a great number of assumptions, especially if dealing with topics where there is little previous experience. These uncertainties are then further amplified by disturbances from external influences that are increasing significantly, the farther a spending plan reaches into the future. (Cohn 2012) explains for example that at the beginning of a software development project, the *cone of uncertainty* can reach 1.6 times of the original estimate (see Figure 1).

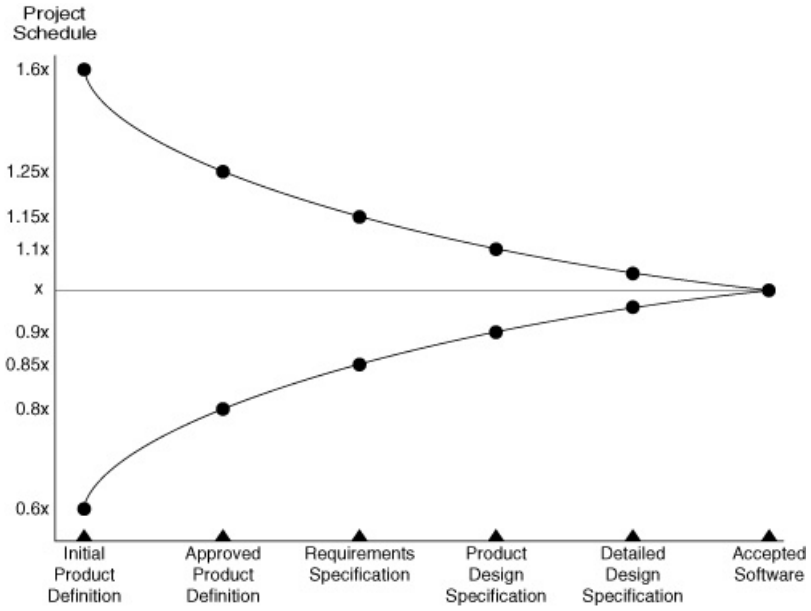



Figure 1: Cone of Uncertatinty

The author's own experience from controlling software projects also shows that operational managers therefore often chose to err on the *side of caution* and overstate expense plans. On company level, these deviations between operational project planning and actual spending in each project pile up over a fiscal year and create a significant challenge for a Finance department.

However, when looking at operational spending on a company level, it is also the authors experience that there are reappearing patterns:

- Spendings for external suppliers are often rushed before the end of a fiscal year (when yearly budgets expire)
- Internal personal cost tend to decrease after a bonus month, when employees chose to change jobs
- Planned spendings in the beginning phases of a project are regularly delayed to later points in time
- Project cost seldom ramp down as planned in the final phase of a project

Experienced Finance departments are therefore using judgment and approximations to manually correct the operational expense plans of a company on an aggregated company level.

These **objective of this project** is therefore to develop a machine learning application to **correct, on company level**, **a planning biases in operational expense planning**

## 1.2 Used Data Set

Obtaining operational expense planning data for research purposes is difficult as such data is often confidential. One notable exception is (Thiele, Ryan, and Abbasi 2021) who provide a real world data set of **eight**[1] projects **from the Australian construction sector**, which were conducted in the early 2010s.

The data set includes a wide range of **data, raging from budgeting, cost and productivity data, scheduling to revenue information**.[2]

While the data set is very extensive, it is unfortunately provided in the form of manually prepared Excel files for each project. This means that there are inconsistencies in data formating (e.g. changing variable names) and multiple errors from incorrect formula or cell linkages, which require intensive manual adjustments.

Nevertheless, this is one of the very few - if not the only - publicly available, *real world* data set with an unparalleled depth of operational and financial project management information. The authors of this data set therefore cannot be credited enough!

While the original data is extremely deep, only a small subset will be used in this project:

- The **estimate data** consists of a bottom-up estimate of resource needs for work packages ("portfolio WBS") in a project. This data forms the overall "performance measurement baseline" of a project. It includes information about the resources name, resource type (e.g. labor or material), resource quantity and measuring unit, cost, etc.
- The **progress data** consists of monthly project observations on five data points for the above work packages:
  - planned and actual quantity,
  - planned and actual cost,
  - as well as earned value (i.e. the amount of overall planned work that was finished during a certain month)

And even from this subset only an excerpt will be used in this project.

## 1.3 Project Approach and Key Steps

Based on the above data set, the whole data analysis chain (as taught by HarvardX's Data Science Professional Certificate Program) was performed through the following steps:

[1] **at the time of writing this paper**

[2] The data can be accessed via: <https://figshare.com/articles/dataset/Project_Portfolio_Dataset/12998822>.

1. The data is imported from the individual Excel files and extensive pre-processing and wrangling is applied to create a homogeneous data set (chapter 2).
2. Exploratory analysis is then performed to understand the data's structure and to identify potential features that can be used for a machine learning application (chapter 3).
3. The data set is then transformed from a *monthly* format to a *standard period* format, in order to make projects with different run times comparable to each other (chapter 4.1.2).
4. In a a first run of analysis, ten different regression models are then applied to the data (chapter 4.2).
5. In a second run of analysis, the most promising model from the first run is expanded to an regression tree analysis[3] (chapter 4.4).
6. Eventually, the results from both analysis runs are summarised and suggestions for future research is given (chapter **??**).

This project itself therefore consists of this project *report* and an accompanying R source code *script*. The *report* will outline the key operations of the *script* and explain the underlying rational.

[3] At least two methods were required by the course instructions

# 2 Data Import and Wrangling

## 2.1 Raw Data

The data set has been saved as a ZIP archive into the author's github repository[4] to ensure a reproducible *script* behavior during the grading process.

As a first step, the project's *script* therefore creates a new directory *DataPreperation* within the user's current work directory and then downloads and extracts the raw data files from this archive into this directory.[5]

This raw data consists of the multiple Excel files for each project that has been analyzed by (Thiele, Ryan, and Abbasi 2021). However, for this analysis, only those files are relevant that contain the *estimation* and *progress* data. Therefore the *script* creates an overview of the available projects and the respective Excel files within the raw data (see Table 1):

Table 1: List of Projects

| projectNumber | projectName | fileName | fileType |
|---|---|---|---|
| 1 | Airport Carpark | P018 - P01 Airport Carpark.xlsx | xlsx |
| 2 | Landfill Cell | P019 - P02 Landfill Cell.xlsx | xlsx |
| 3 | Regional Bypass Road | P020 - P03 Regional Bypass Road.xlsx | xlsx |
| 4 | Regional Arterial Pavement Repairs | P021 - P04 Regional Arterial Pavement Repairs.xlsx | xlsx |
| 5 | Semi-Urban Road Reconstruction | P022 - P05 Semi-Urban Road Reconstruction.xlsx | xlsx |
| 6 | Urban Road Reconstruction | P024 - P06 Urban Road Reconstruction.xlsx | xlsx |
| 17 | Marina Sub-division | P038 - P17 Marina Sub-division.xlsx | xlsx |
| 18 | Rural Road Repairs | P039 - P18 Rural Road Repairs.xlsx | xlsx |

## 2.2 Estimation Data Import

Each Excel file contains a sheet called "Estimate" which holds the effort estimation of the project. These estimates are broken down into work packages, called $portfolioWbs$. Table 2 shows an excerpt of some of the estimation data that is stored per $portfolioWbs$:

Table 2: Excerpt of estimations for project "Airport Carpark"

| portfolioWbs | resourceName | resourceUnit | resourceType | resourceQuantity | cost |
|---|---|---|---|---|---|
| 11 | Surveyor | hr | L | 12.00 | 2,220.00 |
| 11 | Surveyor | hr | L | 18.00 | 3,330.00 |
| 11 | Surveyor | hr | L | 60.00 | 11,100.00 |
| 11 | Surveyor | hr | L | 12.00 | 2,220.00 |
| 11 | Surveyor | hr | L | 26.00 | 4,810.00 |
| 41 | CAT297 | hr | P | 7,813.00 | 2,043.59 |
| 41 | Labour | hr | L | 7,813.00 | 1,670.03 |
| 41 | Water Cart - Hire | hr | P | 7,813.00 | 4,175.07 |
| 41 | Tip Fees | tonne | S | 129.21 | 5,814.46 |

[4] ProjectData.zip in the repoistory https://github.com/rraebild/2020-R-FinalProject

[5] The data set is also provided as a project submission, so it the download fails, the ZIP file can also be manually copied to the *DataPreperation* directory.

| portfolioWbs | resourceName | resourceUnit | resourceType | resourceQuantity | cost |
|---|---|---|---|---|---|
| 41 | Excavator - 30T | hr | P | 445.00 | 2,162.68 |
| 41 | Tipper | hr | P | 445.00 | 4,085.07 |
| 41 | Labour | hr | L | 445.00 | 608.76 |
| 41 | Excavator - 25T | hr | P | 9.00 | 1,215.00 |
| 41 | Tipper | hr | P | 18.00 | 1,530.00 |
| 41 | Labour | hr | L | 18.00 | 684.00 |
| 41 | Stabilised Sand | $m^3$ | M | 10.00 | 2,585.00 |
| 41 | Sand | tonne | M | 20.00 | 359.00 |
| 41 | Labour | hr | L | 6.00 | 228.00 |
| 41 | Excavator - 25T | hr | P | 83.00 | 1,204.84 |
| 41 | Tipper | hr | P | 83.00 | 168.58 |
| 41 | Labour | hr | L | 83.00 | 75.36 |
| 41 | Tip Fees | tonne | S | 10.00 | 450.00 |
| 907 | Float | LS | P | 750.00 | 750.00 |
| 907 | Float | LS | P | 1,800.00 | 1,800.00 |
| 907 | Float | LS | P | 750.00 | 750.00 |
| 907 | Float | LS | P | 500.00 | 500.00 |
| 907 | Float | LS | P | 900.00 | 900.00 |
| 907 | Float | LS | P | 4,600.00 | 4,600.00 |
| 907 | Float | LS | P | 800.00 | 800.00 |
| 907 | Labour | hr | L | 100.00 | 3,800.00 |

Due to (Thiele, Ryan, and Abbasi 2021)'s manual processing of the data in Excel, there are unfortunately numerous inconsistencies within the data. For example:

- In project number one's estimation sheet, there is no $portfolioWbs$ number 15. However, in other parts of the same Excel, there is data referencing to such a work package.
- The estimation sheet also tries to summarize the the $portfolioWbs$ packages into higher level $wbs$ packages, which are apparently used for commercial negotiation and financial settlement with the customer. However, the summaries are within the same columns than the original data and the structuring is achieved by using **color coding** of cells and other ambiguous elements, which has made it quite challenging to identify whether a specific row is summary data or original data.
- In Project Number 18 there is also an additional level used to sum up the cost estimates for certain streets, e.g. for "Fabris Road," "Harvey Road" or "Ann Street." These summary levels also had to be distinguished.

The *script* therefore imports and cleans up a wide variety of data points from the Excel files and also applies a consistent variable naming scheme. However, from these data points, only the following variables are relevant for the current analysis:

- $portfolioWbs$ is a string and identifies a specific work package within a project
- $resourceType$ is a character value and classifies the used resources within in a work packages. It can be of the following types:
  - $L$ = Labour
  - $P$ = Plant
  - $M$ = Material
  - $S$ = Subcontract

- *cost* is floating point value and refers to the resource cost in Australian Dollar

Table 3 gives an summary of the available estimation data after cleansing[6]:

Table 3: Summary of estimation data

| projectNumber | projectName | numberOfWorkPackages | numberOfItemEstimates |
|---|---|---|---|
| 1 | Airport Carpark | 60 | 1017 |
| 2 | Landfill Cell | 28 | 352 |
| 3 | Regional Bypass Road | 44 | 1155 |
| 4 | Regional Arterial Pavement Repairs | 9 | 55 |
| 5 | Semi-Urban Road Reconstruction | 42 | 839 |
| 6 | Urban Road Reconstruction | 49 | 2033 |
| 17 | Marina Sub-division | 56 | 564 |
| 18 | Rural Road Repairs | 33 | 668 |

## 2.3 Progress Data Import

Another sheet within each Excel file is called "Portfolio WBS" and contains progress data throughout the projects duration. After cleansing[7], it provides the following information for each project month:

Table 4: Excerpt of Progress Data for Project Number 4

| portfolioWbs | description | pq.1 | pv.1 | aq.1 | ac.1 | ev.1 |
|---|---|---|---|---|---|---|
| 13 | QA Testing | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 62 | Purchase Water - IRC | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 64 | Sweep & Prime / Primer seal or Seal | 4,250.00 | 14,560.07 | 0.00 | 0.00 | 0.00 |
| 68 | Insitu Stabilisation | 4,250.00 | 70,964.16 | 1,875.00 | 22,621.73 | 31,307.72 |
| 191 | Dayworks - Cleaning Drains | 4.00 | 6,494.55 | 1.20 | 4,751.28 | 1,948.36 |
| 902 | Project Supervision | 8.00 | 6,342.93 | 18.00 | 11,964.15 | 5,441.69 |
| 905 | Small Tools | 0.00 | 0.00 | 0.05 | 359.35 | 0.00 |
| 907 | Site Relocations (Along Road) | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 911 | Accommodation | 41.00 | 4,259.24 | 67.00 | 9,078.12 | 8,090.03 |
| CSA | Cement Powder (Stabilising) | 54.18 | 19,168.27 | 41.00 | 13,825.00 | 14,952.71 |
| QM2.3 | Road Base | 25.00 | 754.92 | 0.00 | 0.00 | 0.00 |
| QMR | Sealing Aggregate (10/7) | 208.62 | 1,083.52 | 0.00 | 0.00 | 0.00 |

- $portfolioWbs$ is a string and refers to the identifier of specific work package (see chapter 2.2)
- *description* is a string and refers to the name of a specific work package
- *pq* is a floating point value and refers to the *planned quantity*
- *pv* is a floating point value and refers to the *planned value* (i.e. "planned cost")
- *aq* is a floating point value and refers to the *actual* (consumed) *quantity*
- *ac* is a floating point value and refers to the *actual* (consumed) *cost*

[6]The specific cleansing activities for effort data are omitted for brevity. Please refer to the *script* for annotated details
[7]The specific cleansing activities for progress data are omitted for brevity. Please refer to the *script* for annotated details.

- *ev* is a floating point value and refers to the *earned value*, which is the part of the planned work that was actually finished during a project month.

The values *pq*, *pv*, *aq*, *ac* and *ev* are maintained cumulatively for each month, so the value for each month also contains a summary of the values from previous month, as table 5 illustrates:

Table 5: Illustration of cummulative data structure

| portfolioWbs | type | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 68 | pv | 70,964.16 | 141,928.32 | 212,892.47 | 283,856.63 | 355,321.71 | 438,809 | 438,809 | 438,809 |
| 68 | %_pv | 0.16 | 0.32 | 0.49 | 0.65 | 0.81 | 1 | 1 | 1 |

A challenge again is that the raw data is not consistently structured, e.g. the various Excel files differ in the number, naming and ordering of their columns. Details for amending these inconsistencies can be found in the *script*.

As a last step, the progress data is then transformed from wide data to a tidy long data structure and added to the data set under the new variable *periodProgress*. The first 10 rows of this variable are shown below:

Table 6: First 10 rows of the variable "periodProgress"

| portfolioWbs | description | valueType | period | periodValue |
|---|---|---|---|---|
| 11 | Survey - Construction | pq | 1 | 10.00 |
| 11 | Survey - Construction | pv | 1 | 2,519.15 |
| 11 | Survey - Construction | aq | 1 | 7.60 |
| 11 | Survey - Construction | ac | 1 | 1,400.00 |
| 11 | Survey - Construction | ev | 1 | 1,232.66 |
| 11 | Survey - Construction | pq | 2 | 10.00 |
| 11 | Survey - Construction | pv | 2 | 2,519.15 |
| 11 | Survey - Construction | aq | 2 | 7.60 |
| 11 | Survey - Construction | ac | 2 | 1,400.00 |
| 11 | Survey - Construction | ev | 2 | 1,232.66 |

## 2.4 Cleansed Dataset

The final, cleansed data is stored in the tibble *projects* and has the following structure:

```
## # A tibble: 8 x 7
##   projectNumber projectName   periodProgress fileName fileType estimate progress
##           <int> <chr>         <list>         <chr>    <chr>    <list>   <list>
## 1             1 Airport Carp~ <tibble>       P018 - ~ xlsx     <tibble> <tibble>
## 2             2 Landfill Cell <tibble>       P019 - ~ xlsx     <tibble> <tibble>
## 3             3 Regional Byp~ <tibble>       P020 - ~ xlsx     <tibble> <tibble>
## 4             4 Regional Art~ <tibble>       P021 - ~ xlsx     <tibble> <tibble>
## 5             5 Semi-Urban R~ <tibble>       P022 - ~ xlsx     <tibble> <tibble>
## 6             6 Urban Road R~ <tibble>       P024 - ~ xlsx     <tibble> <tibble>
## 7            17 Marina Sub-d~ <tibble>       P038 - ~ xlsx     <tibble> <tibble>
## 8            18 Rural Road R~ <tibble>       P039 - ~ xlsx     <tibble> <tibble>
```

- *projectNumber*, *projectName*, *fileName* and *fileType* are normal columns that contain integer or string values

- *periodProgress*, *estimate* and *progress* are *list-columns* where into each row a tibble with the respective project information is nested. The structure of these nested tibbles is homogeneous within each column

The cleansed, raw data set is now ready for various analyses.

## 2.5 Base Data Set

After the raw data has been cleansed, the *base data set* for the analysis is selected, which is an excerpt of the raw data.

This is archived via the new function *baseSet*, which extracts the *actual cost* (*ac*) and the *planned values* (*pv*) for each project from *periodProgress*. Other raw data is omitted. The function also removes rows which contain $NA$ values to ease the analysis. If required, the function can optionally also add a row index before $NA$ values are omitted (to allow a reconstruction of the full base data set).

```
## function to define the base data set
baseSet <- function( .removeNAs = TRUE , .baseIndex = FALSE ,
                     .includeValueTypes = c( "ac", "pv" ) ) {

  # start with the full project data set
  projects %>%

    # omit project data that is not relevant for this analysis
    select( projectNumber, projectName, periodProgress ) %>%

    # unnest the progress data to have a tidy, long table with all observations
    unnest( periodProgress ) %>%

    # filter for the periodValues that are part of this analysis
    filter( valueType %in% .includeValueTypes ) %>%

    # if an index shall be added, e.g. in order to restore to original values
    { if (.baseIndex) rownames_to_column( . , var = "baseIndex" ) else . } %>%

    # remove NA entries if necessary
    { if (.removeNAs) na.omit(.) else . }

}
```

A random sample of 20 observations from the *baseDataSet* is shown below:

Table 7: Sample from Base Data Set

| projectNumber | projectName | portfolioWbs | description | valueType | period | periodValue |
|---|---|---|---|---|---|---|
| 1 | Airport Carpark | 113 | Electrical Sub-contract works | pv | 6 | 69,827.00 |
| 1 | Airport Carpark | RSM | Mesh | ac | 5 | 136.65 |
| 3 | Regional Bypass Road | 151 | Supply & Place Silt fences & protective devices | ac | 4 | 2,466.42 |
| 3 | Regional Bypass Road | 67 | Install Road Furniture | ac | 1 | 0.00 |

| projectNumber | projectName | portfolioWbs | description | valueType | period | periodValue |
|---|---|---|---|---|---|---|
| 3 | Regional Bypass Road | 910 | De-Establishment & Cleanup | pv | 2 | 0.00 |
| 3 | Regional Bypass Road | 111 | Install Conduits / pits / rag bolts | ac | 5 | 7,286.30 |
| 3 | Regional Bypass Road | 901 | Project Management | ac | 5 | 5,950.00 |
| 6 | Urban Road Reconstruction | 67 | Install Road Furniture | pv | 9 | 4,105.00 |
| 6 | Urban Road Reconstruction | 33 | Re-spread topsoil | ac | 3 | 2,940.25 |
| 6 | Urban Road Reconstruction | 907 | Site Establishment / disestablishment | ac | 2 | 1,281.52 |
| 6 | Urban Road Reconstruction | 62 | Place & Compact Gravel | pv | 10 | 269,788.04 |
| 6 | Urban Road Reconstruction | 905 | Small Tools | ac | 4 | 1,850.07 |
| 6 | Urban Road Reconstruction | 57 | Remove & Replace unsuitable | pv | 10 | 129,486.51 |
| 17 | Marina Sub-division | 81 | > 450 | ac | 14 | 16,600.83 |
| 17 | Marina Sub-division | QM2.3 | 2.2999999999999998 | pv | 12 | 18,984.00 |
| 17 | Marina Sub-division | 916 | Site Administration | pv | 4 | 16,800.00 |
| 17 | Marina Sub-division | 86 | SW Sub-contractors | pv | 5 | 6,500.00 |
| 18 | Rural Road Repairs | 901 | Project Management | ac | 2 | 37,971.88 |
| 18 | Rural Road Repairs | SWP | Supply Stormwater Pipe | ac | 6 | 0.00 |
| 18 | Rural Road Repairs | 64 | Sweep & Prime / Primer seal or Seal | pv | 5 | 350,474.00 |

# 3 Data exploration

## 3.1 Company Level Expenses

Figure 2 plots a comparison of the aggregated planned versus the aggregated actual project expanses of the company over time[8].

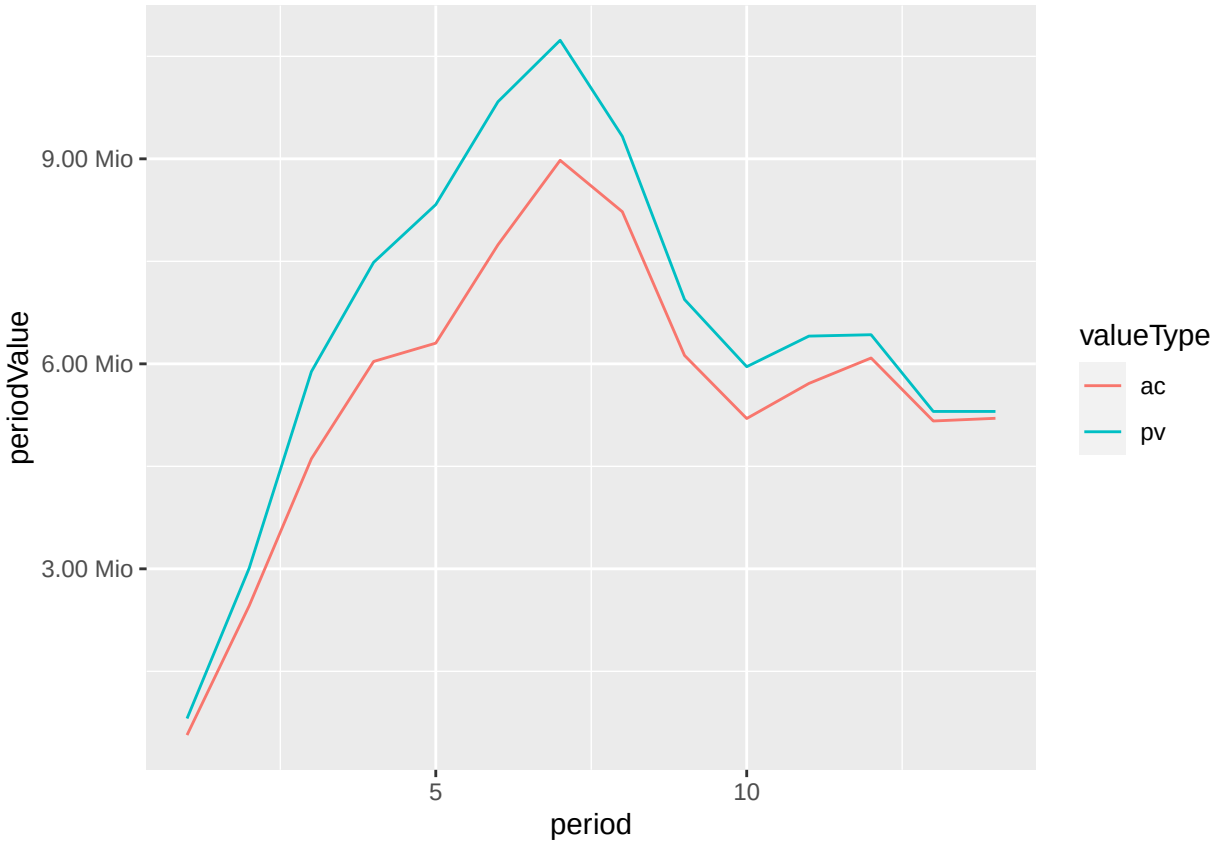



Figure 2: Planned and Actual Company Level Expanses

This reveals that there is a significant mismatch between what the company **plans** to pay out (e.g. to employees, suppliers and for material) and what it **actually** pays out. As explained in chapter 1 this leads to unnecessary additional expanses, as the unused capital needs to be financed.

Assuming a cost of capital of 4% per year, this would for example create additional cost of 6,976.34 in period six:

```
baseSet() %>%
  pivot_wider( names_from = "valueType", values_from = "periodValue") %>%
  mutate ( unusedCapital = pv - ac) %>%
  group_by( period ) %>%
  summarise( unusedCaptial = sum( unusedCapital, na.rm = TRUE) ) %>%
  filter( period == 6) %$%
  unusedCaptial * (0.04 /12)
```

[8]For simplification it is assumed that each project starts at the same period (i.e. month).

```
## [1] 6976.343
```

## 3.2 Project Level Expanses

Breaking down the planned and actual expenses to project level reveals a similar deviation between planned and actual expenses across most projects. However, it appears that the deviations seem be larger during quarter two and three of a project.

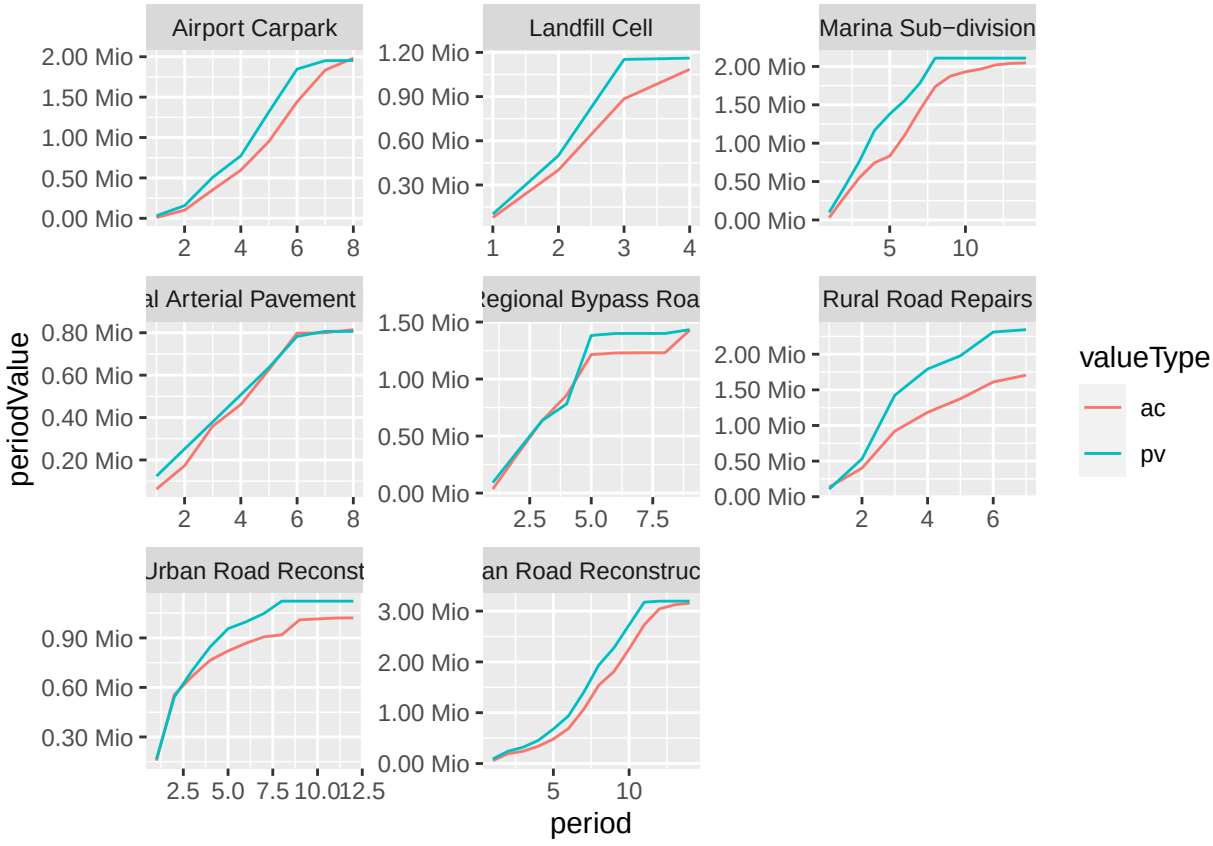



Figure 3: Project Level Expenses

The idea of this project is therefore to use machine learning approaches to reveal and formalize these patterns, which would allow the finance department to introduce a bias correction between planned and actual expenses in order to reduce the financing cost of the company.

## 3.3 Work Package Level Expenses

At the time of writing this paper, (Thiele, Ryan, and Abbasi 2021) provide data for only eight projects. As a rule of thumb (for example in regression analysis) there should be at least 10 observations per potential variable. Therefore, the data set is too small on project level to create stable estimates.

However, the data in the base set is provided for each project on work package level ($portfolioWbs$). An illustrative sample from project number 1 shows in figure 4 that there seem to be similar patterns present on this level.

In total, there are 351 work packages in the base data set, so the expectation is that this is a sufficient number of observations to create a stable estimate.

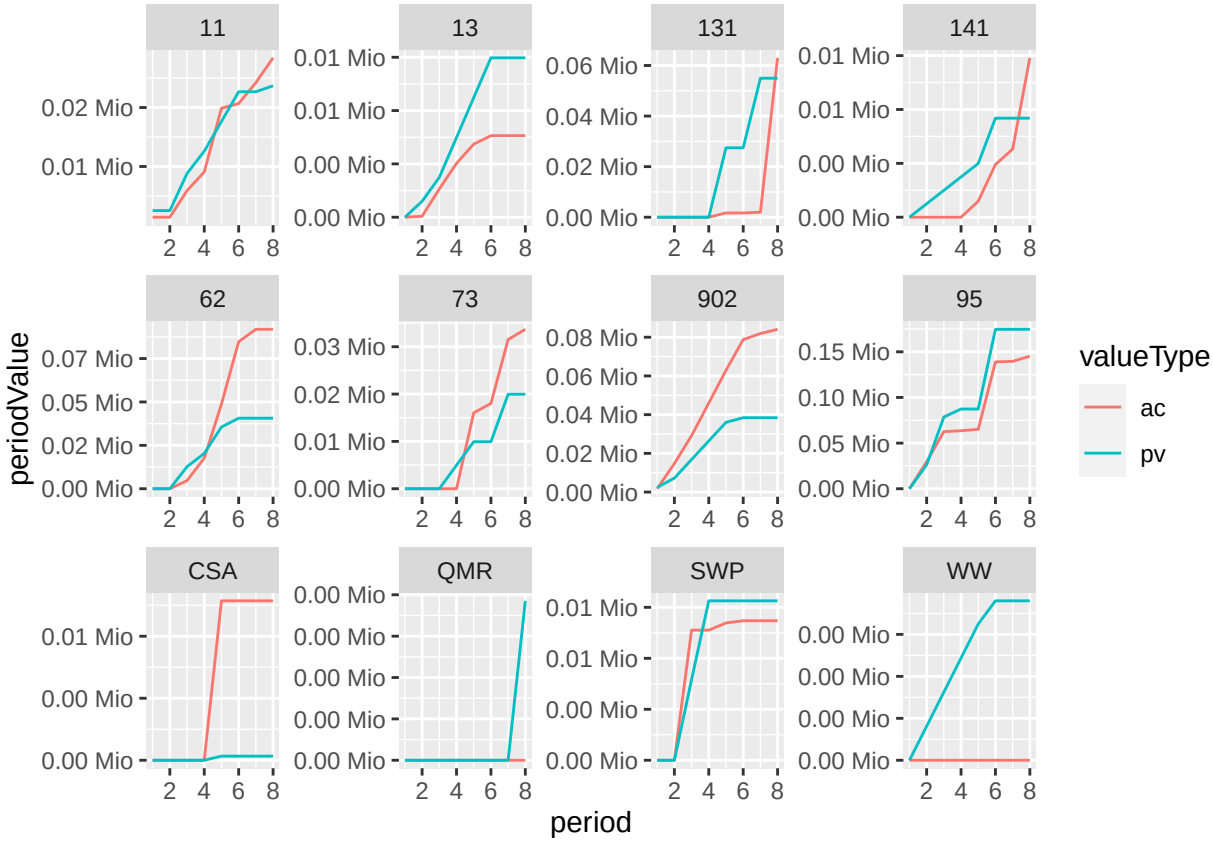



Figure 4: Expenses for a sample of Work Packages from Project Number 1

# 4 Modeling of the Bias Correction

## 4.1 Preparation

### 4.1.1 Training and Test Set

According to the machine learning principles taught by (Irizarry 2022), the base data set is first to be divided into a *training set* and a *test set* and the suggested split of 80/20 is used here.

```
# set the seed to create reproducible partitions
set.seed(20)

# central definition of the base data set
baseSet() %T>%

  # create training index (within the global environment)
  { trainIndex <<- createDataPartition( y = .$periodValue,
                                        times = 1,
                                        p = 0.8,
                                        list = FALSE)} %>%

  # create a list with training and testing data sets (in global environment)
  { dataSet <<- list ( training = .[ trainIndex , ],
                       testing = .[ -trainIndex , ] )}


# nest the progress data to project level
for ( i in c(1,2) ) {
  dataSet[[i]] %<>%
    nest( progress = -c(projectNumber, projectName))
}
```

This creates the new list *dataSet* which contains the two similar structured tibbles *training* and *testing*[9]. Each of these subsets is structured in a similar way

### 4.1.2 Transforming into Standard Periods

Each project has a different duration (e.g. project number one lasts 8 periods, project number two lasts 4 periods, while project number six lasts for 14 periods)[10]. Since the assumption is that the estimation bias somehow varies within the project duration, the approach of this analysis is to standardize the duration of all projects to the same length of 100 *standardPeriods*.

This is supposed to allow for the detection of time-based patterns, e.g. if a bias is present for example only during the later phases of a project.

The transformation is archived by first calculating a *stepvalue* ( number of project *periods* divided by number of *standardPeriods* ) and then adding a cumulative sum of step values to create a *conversion break value* for each *standardPeriod*. The respectice planned or actual value of a *period* value is then assigned to the corresponding *standardPeriod*, if the *period* number is smaller or equal to the *conversion break value* but bigger than the previous *period* number. Figure 5 shows a simplified example for a project with 4 *periods* and a setting with 10 *standardPeriods*.

[9] Structuring the data like this allows for developing general functions that can later be applied to both data sets.
[10] A *project period* resembles one *calendar month*

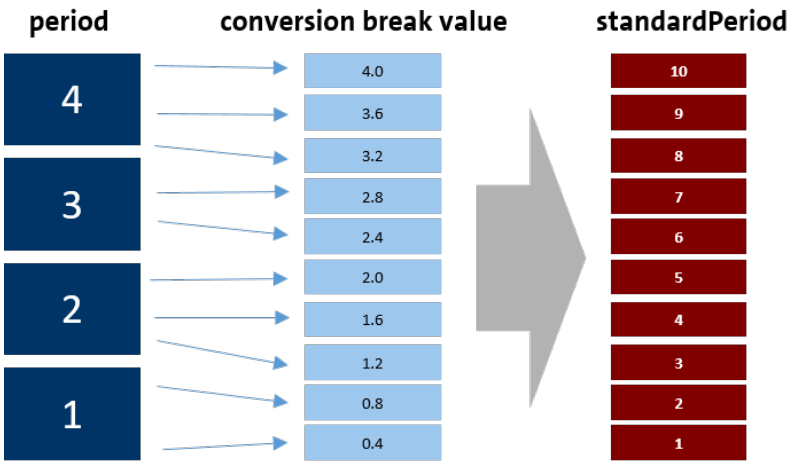



Figure 5: Conversion Break Values for standardPeriods

The mapping of *period*s to *standardPeriod*s is captured in a mapping table and added to the *data set*.

```
# maximum number of periods that exist in the dataset
maxPeriods <- fullDataSet() %>%
  unnest( progress ) %$%
  max( period )


# Number of standardized periods (shall always be bigger than
# normal periods, usually 100)
stdPeriodNumber = max( 100,  maxPeriods)


# stepvalue
conversionBreak = maxPeriods / stdPeriodNumber

# Mapping table between periods and stdPeriods
periodMappings <-

  # create a tibble with a row for each period
  tibble ( period = 1:maxPeriods) %>%

  # find the fraction until which the stdPeriod covers
  rowwise () %>%
  mutate( toBreak = max( seq( from = conversionBreak,
                              to = period,
                              by = conversionBreak ))) %>%

  # find the fraction from which the stdPeriod covers
  ungroup () %>%
  mutate( fromBreak = lag(toBreak, default = 0) + conversionBreak ) %>%

  # nest the list of stdPeriod that match do a specific period into a new column
  rowwise() %>%
  mutate( stdPeriod = list( seq( from = fromBreak,
                                 to = toBreak,
```

```
                                    by = conversionBreak )
                                / conversionBreak )) %>%

    # remove the temporary columns
    select( -fromBreak, -toBreak)

  # add the mapping table to the dataSet
  dataSet %<>%
    c( periodMapping = list( periodMappings ) )
```

Afterwards, the mapping table is used to transform the progress data to *standardPeriods*.

```
  ## transform both training and testing project progress data (list element 1 and 2)
  ## to standardized periods
  for (i in 1:2 ) {

    # dataSet is changed
    dataSet[[i]] %<>%
      unnest( progress ) %>%

      # add the list of stdPeriods to progress data
      left_join( dataSet$periodMapping, by = "period") %>%

      # expand the new rows for stdPeriods (by definition: stdPeriods are
      # always more than normal Periods)
      unnest(stdPeriod) %>%

      # nest back the progress data
      nest( progress = -any_of( c( "projectNumber", "projectName", "total" )))
  }
```

### 4.1.3 Adding Resource Information

Besides the elapsed project time, an additional source of explanation for the bias might be the *Mix of Resources* in each work package. In order to utilize this information, first a function $portfolioWbs$ is defined that can pull the resource information for a specific work package and then calculate the respective share of resources types. E.g. for a work package number 41 of project 1 the resource mix is like this (L=Labor, M=Material, P=Production, S=Supplier).

```
portfolioWbs( .projectNumber = 1, .portfolioWbs = 41) %>%
  unnest( resources )
```

```
## # A tibble: 4 x 4
##   projectNumber portfolioWbs resourceType resourceWeight
##           <int> <chr>        <chr>                 <dbl>
## 1             1 41           L                     0.112
## 2             1 41           M                     0.101
## 3             1 41           P                     0.571
## 4             1 41           S                     0.216
```

This information is then added to the *dataSet*.

An example for a complete set observations therefore looks like this:

Table 8: Observation set from Training Data in Project Number 1 in Standard Period 10

| description | valueType | periodValue | weightM | weightS | weightL | weightP | weightNoResource |
|---|---|---|---|---|---|---|---|
| Demolition works | ac | 9,050.68 | 0.1 | 0.22 | 0.11 | 0.57 | 0 |
| Demolition works | pv | 21,794.57 | 0.1 | 0.22 | 0.11 | 0.57 | 0 |

## 4.2 First Round of Modeling

### 4.2.1 Models

In the first round of modeling, there are 10 models to be trained. Model number 1 is the most simple one, as it just uses the planned value to forecast the actual value. Model number 2 to 10 are using *Generalized Linear Regression* (glm) with different predictor variables. E.g. model 3 uses two variables: the *planned value* and the *share of the resource type material* within a work package.

Table 9: Models to be trained

| modelId | modelName |
|---|---|
| 1 | just use plan value |
| 2 | glm with value.ac ~ value.pv |
| 3 | glm with value.ac ~ value.pv + weightM |
| 4 | glm with value.ac ~ value.pv + weightS |
| 5 | glm with value.ac ~ value.pv + weightL |
| 6 | glm with value.ac ~ value.pv + weightP |
| 7 | glm with value.ac ~ value.pv + stdPeriod |
| 8 | glm with value.ac ~ value.pv + weightNoResource |
| 9 | glm with value.ac ~ value.pv + weightM + weightS + weightL + weightP |
| 10 | glm with value.ac ~ value.pv + weightM + weightS + weightL + weightP + weightNoResource |

### 4.2.2 Training and Prediction

There are two different functions implemented to start the modeling:

- The function *training* accepts a *data set* (i.e. either the *training* or the *testing* data) along with the *model number*. Based on this, it returns a *trained model*:

```
model <- training( .dataset = dataSet$training,
                   .modelNo = 9 )
```

```
## $model
## Generalized Linear Model
##
## 16587 samples
##     5 predictor
##
## No pre-processing
## Resampling: None
```

- The function *prediction* accepts a (new) *data set* along with a *trained model* and returns a *prediction*, i.e. a *set of forecasted cost values* that correspond to the supplied input data.

```
forecast <- prediction( .dataSet = dataSet$testing,
                        .model = model )
```

Table 10: Samples rows and columns from forecasted values

| projectNumber | portfolioWbs | stdPeriod | weightM | weightS | weightL | weightP | value.pv | value.fc |
|---|---|---|---|---|---|---|---|---|
| 1 | 103 | 10 | 0.15 | 0.85 | 0 | 0 | 0 | NA |
| 1 | ETC | 52 | 1.00 | 0.00 | 0 | 0 | NA | NA |
| 6 | CSS | 77 | 0.00 | 0.00 | 0 | 0 | NA | NA |

## 4.3 Loss and Evaluation

### 4.3.1 Loss

(Irizarry 2022) describes the *Loss function* as a metric to determine the "best" approach among several alternatives in Machine Learning. Very often the concept of *Root Mean Squared Error* (RMSE) is used for this.

However, in this analysis, we are choosing a variant of the Mean Absolute Error (MAE) as decisive metric. The reason for this is that it fits very elegantly with the problem description of minimizing the impact from deviations between the planned and actual company expenses:

- If the delta between the forecasted and the actual cost in a certain *Period* is smaller or equal to zero, then we are experiencing a project cost *overrun*. In this case the delta is multiplied with an *OverrunPenalty* (e.g. the interest rate for a short-term loan)
- If the delta between the forecasted and the actual cost in a certain *Period* is bigger than zero, then we are experiencing a project cost *underrun*. In this case the delta is multiplied with an $UnderRunPenalty$ (e.g. the interest rate for the now unused long-term loan)
- However, since this penalty only happens on company level, we want to sum up all the deltas from all work package within a certain period and we specifically want positive and negative deltas within a period to cancel each other out (i.e. if the overrun from one work package is compensated by underrun of another work package, then there are no losses for the company)

Figure 6 shows this principle as a mathematical formula. To keep things simple, both the $OverRunPenalty$ and the $UnderRunPenalty$ are set to 0.05 for the first round of analysis.

$$LOSS = \sum^{Periods} \begin{cases} \sum^{WorkPackages} \left(\hat{Y} - Y\right) \cdot OverRunPenalty, & \text{if } \left(\hat{Y} - Y\right) \leqq 0 \\ \sum^{WorkPackages} \left(\hat{Y} - Y\right) \cdot UnderRunPenalty, & \text{if } \left(\hat{Y} - Y\right) > 0 \end{cases}$$

Figure 6: Loss function

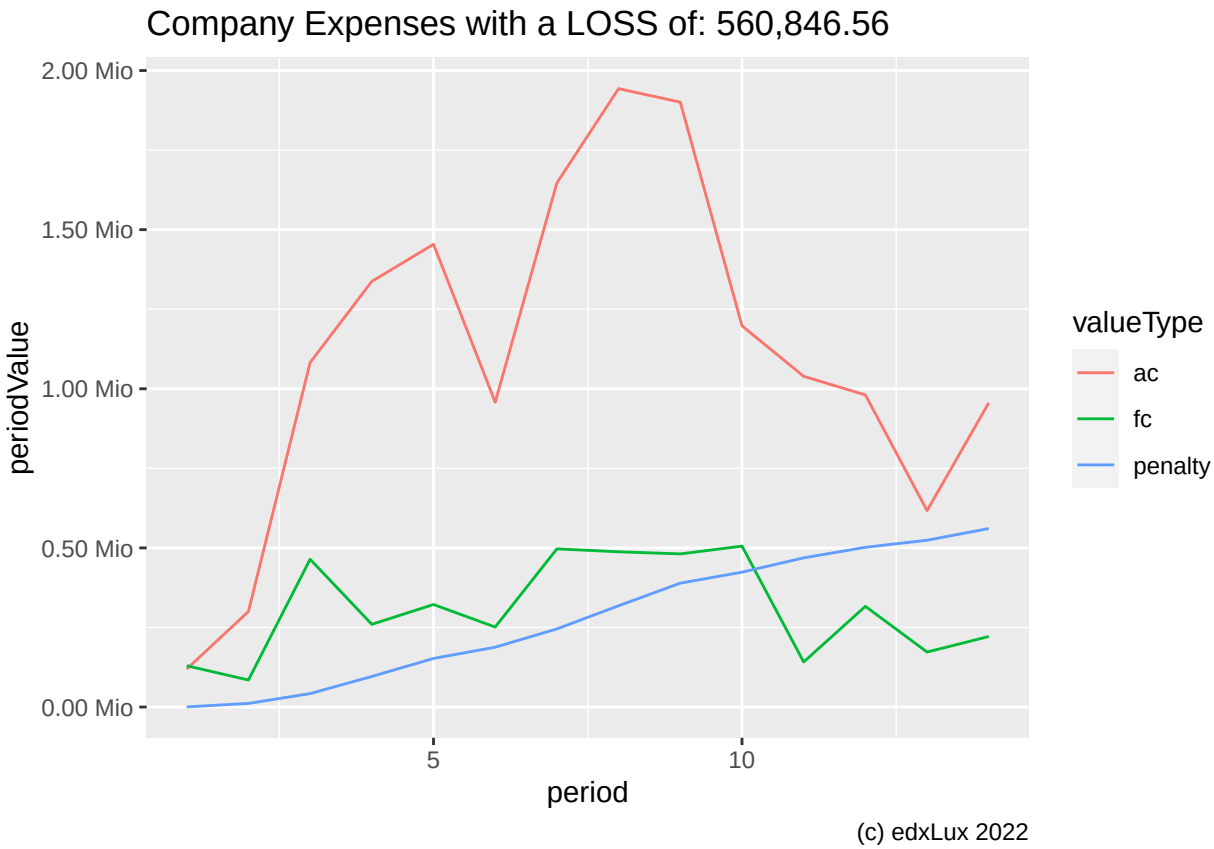
### 4.3.2 Evaluation

The function *evaluation* implements the *loss* calculation. It accepts a *prediction* object (i.e. a set of forecasted and actual values for a range of periods) along with a settings for the $OverRunPenalty$ and the

*UnderRunPenalty* (the default value for both is 0.05). The function returns a list with three objects: (a) a tibble with the predictions, (b) a *ggplot object* that plots the forecasted values against the actual values, and (c) the calcualted *loss*.

```
  evaluation( .prediction = forecast,
              .overrunPenatly = 0.05,
              .underrunePenalty = 0.05 )
```

```
## $predictions
## # A tibble: 1,288 x 14
##    projectNumber projectName     portfolioWbs description period weightM weightS
##            <int> <chr>           <chr>        <chr>        <int>   <dbl>   <dbl>
##  1             1 Airport Carpark 103          Water Sub-~      1 0.154     0.846
##  2             1 Airport Carpark 103          Water Sub-~      2 0.154     0.846
##  3             1 Airport Carpark 103          Water Sub-~      7 0.154     0.846
##  4             1 Airport Carpark 11           Survey - C~      1 0         0
##  5             1 Airport Carpark 11           Survey - C~      4 0         0
##  6             1 Airport Carpark 11           Survey - C~      5 0         0
##  7             1 Airport Carpark 11           Survey - C~      6 0         0
##  8             1 Airport Carpark 111          Install Co~      1 0.00765   0.148
##  9             1 Airport Carpark 111          Install Co~      4 0.00765   0.148
## 10             1 Airport Carpark 111          Install Co~      5 0.00765   0.148
## # ... with 1,278 more rows, and 7 more variables: weightL <dbl>, weightP <dbl>,
## #   weightNoResource <dbl>, stdPeriod <list>, value.ac <dbl>, value.pv <dbl>,
## #   value.fc <dbl>
##
## $companyPlot
```




```
##
## $loss
## [1] 560846.6
```

### 4.3.3 Results from First Run of Modeling

Based on the preparations mentioned above, a first run of modeling is applied:

```
# Function that goes through all the modeling steps for a specific model and returns the loss
modelingLoss <- function ( currentModel ) {

  model <- training( .dataset = dataSet$training,
                     .modelNo = currentModel)

  forecast <- prediction( .model = model,
                          .dataSet = dataSet$testing )

  evaluation( forecast ) %>%
    pluck(3) %>%
    return()
}


# add a column with the first run results to the modelComparison
modelComparison %<>%
```

```
      mutate( firstRunLOSS = map_dbl( modelComparison$modelId,
                                      ~ modelingLoss( .x )))
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

The results are however disappointing. None of the new models (2-10) yields any improvement over the benchmark approach (model 1) of just using the "planned" values as forecast. Out of all the new models, model number 7 gives the best result, but its *loss* of 558,460.40 is still very far away from the benchmark of 301,673.90 from "just using the plan value."

Table 11: Results from First Modeling Run

| modelId | modelName | firstRunLOSS |
|---|---|---|
| 1 | just use plan value | 301,673.9 |
| 2 | glm with value.ac ~ value.pv | 562,386.8 |
| 3 | glm with value.ac ~ value.pv + weightM | 562,650.0 |
| 4 | glm with value.ac ~ value.pv + weightS | 560,642.0 |
| 5 | glm with value.ac ~ value.pv + weightL | 562,752.5 |
| 6 | glm with value.ac ~ value.pv + weightP | 561,976.4 |
| 7 | glm with value.ac ~ value.pv + stdPeriod | 558,460.4 |
| 8 | glm with value.ac ~ value.pv + weightNoResource | 562,401.1 |
| 9 | glm with value.ac ~ value.pv + weightM + weightS + weightL + weightP | 560,846.6 |
| 10 | glm with value.ac ~ value.pv + weightM + weightS + weightL + weightP + weightNoResource | 560,846.6 |

## 4.4 Second Run

Due to the limited success of the First Run, the model is to be expanded. During Chapter 3.2 it was observed that there seems to be a difference in the planning bias for earlier and for later phases of a project, i.e. the planned values for earlier periods seem to be closer to the actual values than the planned values for later periods.

### 4.4.1 Regression Tree

As an advanced step, a *Regression Tree analysis* is now applied. Hereby the the data is separated into two or more sections and the regression analysis is then applied to each section separately.

To implement this approach, two new functions are introduced:

- *splitAtStdPeriod* takes a *data set* and a *breakpoint* as inputs. It then splits all the observations into a "head" part and a "tail" part, according to the breakpoint. An example for 10 randomly selected observations for a *breakpoint* at 25 is given below:

Table 12: Examples from Head and Tail split with stdPeriod 25 as Breakpoint

| branch | projectNumber | portfolioWbs | valueType | period | periodValue | stdPeriod |
|---|---|---|---|---|---|---|
| tail | 4 | 64 | pv | 6 | 90,032.61 | 42 |
| tail | 1 | 57 | ac | 8 | 41,783.66 | 53 |

| branch | projectNumber | portfolioWbs | valueType | period | periodValue | stdPeriod |
|---|---|---|---|---|---|---|
| tail | 3 | 62 | ac | 6 | 102,754.37 | 38 |
| tail | 6 | 67 | ac | 8 | 415.45 | 53 |
| tail | 17 | 905 | pv | 7 | 5,000.00 | 46 |
| tail | 17 | QM2.3 | pv | 7 | 18,984.00 | 44 |
| tail | 6 | 66 | pv | 9 | 0.00 | 58 |
| tail | 3 | 141 | pv | 8 | 115,916.75 | 55 |
| tail | 18 | 57 | pv | 7 | 162,992.63 | 49 |
| tail | 5 | 87 | pv | 5 | 63,795.66 | 30 |

- Then there are two other new functions *trainingOnSplitData* and *preditionOnSplitData*, which perform the same operations than their counterparts from the First Run (see chapter 4.2.2 ), just separately for the *head* and *tail* section.

### 4.4.2 Finding the optimal Splitting Point and Results from Second Run

Based on the above functions the optimal *breakpoint* can be found via a trial-and-error procedure. The Analysis starts with model 7, since this has previously delivered the best results.

Figure 7 plots the results from the trial-and-error run and reveals that the best result is archived with a breakpoint at *stdPeriod* 56 which results in a LOSS of 554,788.

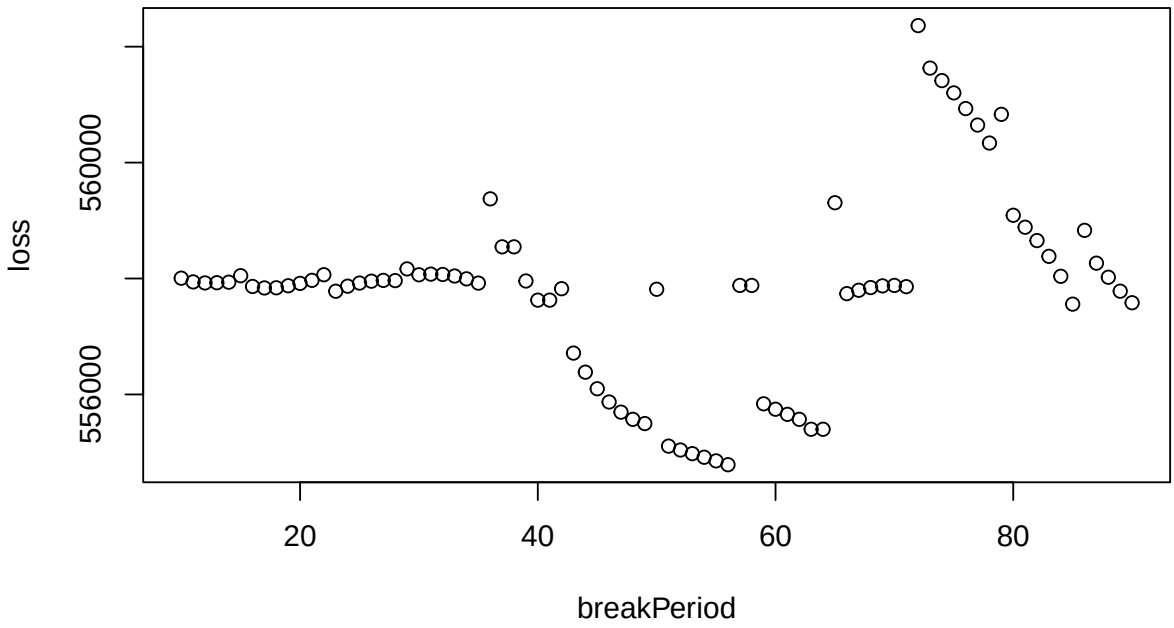



Figure 7: LOSS from regression tree analysis with model 7 according to different Breakpoints

However, this is also a very disappointing result, as it is only a marginal improvement over the first run (558,460.4) and still far away from the benchmark of simply using the planned values (301,673.9).

### 4.5 Analysis of shortcomings

Figure 8 reveals that model in the Second Run also suffers from a significant underestimation in nearly all periods.

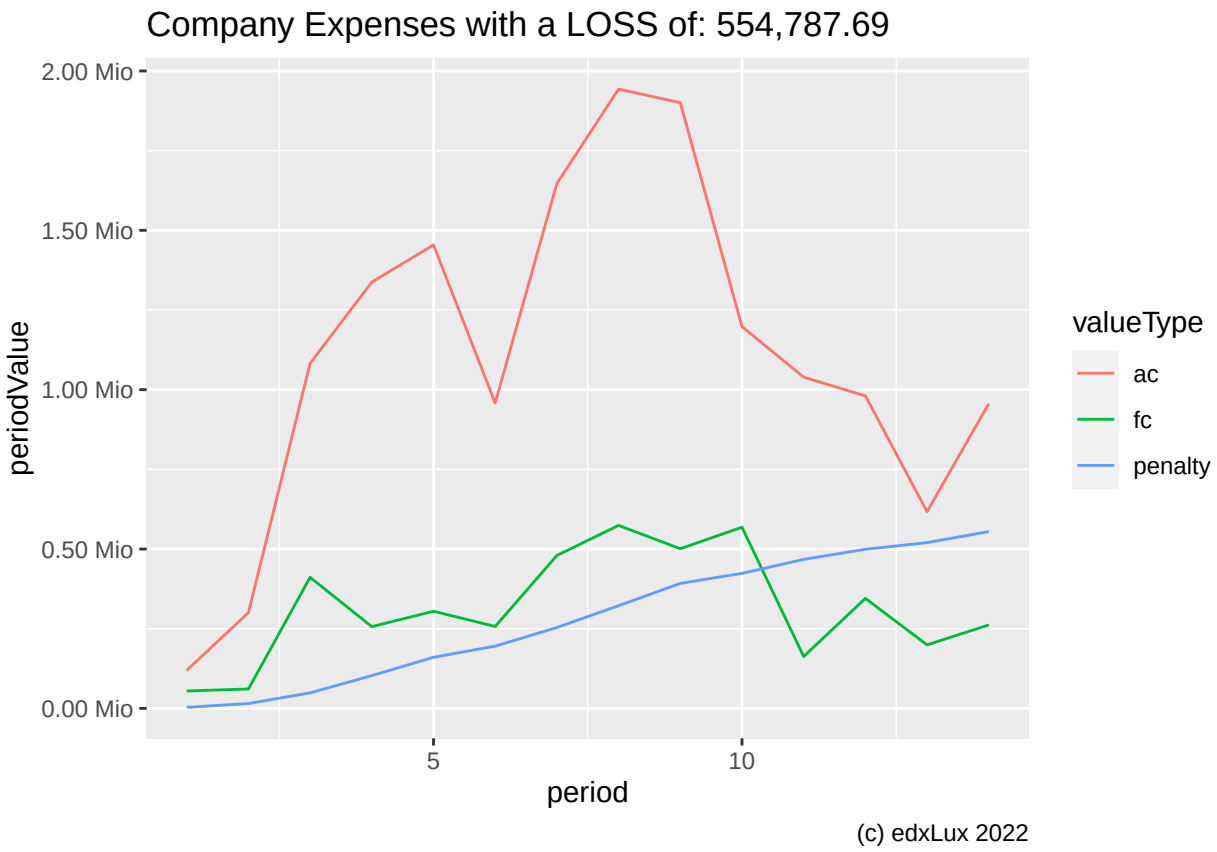



Figure 8: Companylevel plot of of Model 7 during Second Run (Regression Tree with Head and Tail)

Trying to find the reason for this underestimation is revealed by looking at the first 10 observations from the prediction. It shows that 9 out of 10 observations contain $NA$s instead of an planned or forecasted value.

Table 13: First 10 observations from Second Run

| projectNumber | portfolioWbs | period | stdPeriod | value.pv | value.ac | value.fc |
|---|---|---|---|---|---|---|
| 1 | 111 | 8 | 57 | 65,406.52 | NA | NA |
| 1 | 112 | 8 | 57 | NA | 150.00 | NA |
| 1 | 121 | 8 | 57 | NA | 7,800.00 | NA |
| 1 | 134 | 8 | 57 | NA | 708.30 | NA |
| 1 | 142 | 8 | 57 | 16,500.00 | 4,339.41 | 14,812.65 |
| 1 | 21 | 8 | 57 | NA | 12,095.59 | NA |
| 1 | 221 | 8 | 57 | NA | 491.47 | NA |
| 1 | 23 | 8 | 57 | 7,000.00 | NA | NA |
| 1 | 33 | 8 | 57 | NA | 14,500.11 | NA |
| 1 | 52 | 8 | 57 | NA | 164,870.94 | NA |

This is most likely due to a bug somewhere in the source code. However, due to time limits, the author was not able to fix this bug before the submission deadline.

# 5 Conclusion and Suggestion for Future Research

This project has set itself the objective of developing a machine learning approach that helps Finance departments to correct a financial planning bias from a company's operational planning.

To achieve this aim, significant efforts were invested to cleanse and wrangle the extensive data set that was provided from (Thiele, Ryan, and Abbasi 2021).

Afterwards, a data exploration was performed, which showed that there seem to be a reappearing "pattern" on the levels of both *project* and individual *work packages*.

Based on this hypothesis, first a regression analysis with ten different models was performed and since this did not provide convincing results, the model was extended to the advanced model of an regression tree.

However, this also did not provide convincing results and an analysis of the predictions revealed that most likely there are some errors within the code, which unfortunately could not be resolved before the submission date.

Other researches can therefore directly only benefit from the intensive data cleansing work that was applied to the data set, in order to facilitate their own research. Additional benefits might come from the concept of standard Periods and the general outline of the intended machine learning approach.

Future research should therefore first be invested to amend the shortcomings in the *script*, so that the full potential of this research can be revealed.

After that, a deeper analysis into the prediction power of different variables could be worthwhile, e.g. to evaluate whether certain resource types have a higher likelihood of creating a bias than others.

After that, different machine learning algorithms could also be applied. E.g. trying to categorize the work packages according to their resource composition might be a worthwhile approach, since it is the authors experience, that for example supplier cost estimates are much less volatile than labor cost estimates.

Other analysis approaches (e.g. by using different machine learning algorithms) might also be easily possible, due to the very flexible implementation of the script's machine learning functions on (Thiele, Ryan, and Abbasi 2021)'s data set.

# References


Cohn, Mike. 2012. *Agile Estimating and Planning.* Prentice Hall PTR.

Irizarry, Rafael A. 2022. "Introduction to Data Science." 2022. https://rafalab.github.io/dsbook/.

Moltke, Helmuth von, and Der Grosse Generalstab. 1892. *Moltkes Militärische Werke, Abtheilung II: Die Thätigkeit Als Chef Des Generalstabes Der Armee Im Frieden, Bd. 1-3 (i 6 Vol.).* Ernst Siegfried Mittler & Sohn.

Steincke, Karl Kristian. 1948. *Arvel Og Tak: Minder Og Meninger.* Fremad.

Thiele, Brett, Michael Ryan, and Alireza Abbasi. 2021. "Developing a Dataset of Real Projects for Portfolio, Program and Project Control Management Research." *Data in Brief* 34: 106659. https://doi.org/https://doi.org/10.1016/j.dib.2020.106659.