How to Speed Up the Container Unloading Process

A Case Study in Warehouse Logistics of a Medium Sized Online Retailer

by Matthias Langenhan

03.10.2021

Introduction & Motivation

My name is Matthias Langenhan and I am currently enrolled in an university program to get my Master's degree in Economics. During both, my undergraduate and graduate programs, a lot of the curriculum was focused on statistics and econometrics (basically applied statistics with focus on economics-specific contexts). Those courses were the most engaging and challenging in the entire programs. Soon I realized, that if I wanted to apply what I learned about statistics in the real world, and not only during exams, then I would have to take it upon myself.

So, I did a little brainstorming into what idea or problem I wanted to tackle. Pretty soon I came to the conclusion that whatever I wanted to look into, it had to be within the context of my job as a working student in warehouse logistics. There, I managed a team of six people in the inbound part of the logistics process. Specifically, I was responsible for receiving and unloading of container freight. This process involved tasks such as: validating the shipment, checking whether the correct items in the ordered quantities are delivered, passing this data to the warehouse inventory management system, delegate individual tasks to the team members, keep motivation up, meeting deadlines, reporting to management and head of logistics.

At the same time, my interest in learning a programming language with focus on statistics was sparked. I felt like my little statistic project would be the perfect opportunity to learn and apply the features of that programming language. Back then, my choice was to use *Python* as the main tool for my analysis. Tinkered around a bit with the data and statistics packages, but ultimately I was disappointed. It felt dishonest to use functions that someone else had built and just plug in my values for the arguments. A big black box that produces output without me knowing what magic happened on the inside. Instead of focusing on my little statistics project, I wanted to know the in and outs of Python. A journey that I am still on. That project I started? Adjourned for the moment and all that data I had already collected was sitting on my hard drive.

Fast forward two years: I am about to finish the Google Data Analytics Certificate. The only thing left to do is the capstone project. A perfect opportunity for me, to continue the side-tracked project I started out on in 2019. Although this time, I opted for *R* as the primary tool of my data analysis instead of *Python*, just because I feel so comfortable after having learned about it in the Google Data Analytics course.

Business Problem & Data

The company we are looking at is a mid level online retailer for computer parts and accessories. Specifically, we are looking into a process that takes part within the inbound part of the warehouse: the unloading of 40 feet containers. Items are stacked loosely in the container and need to be put on palettes and these need to be pulled out by pallete trucks. The weekly frequency of container shipments ranges from zero up to fifteen, largely depending on the season.

Since there is no specially dedicated team that handles the unloading process, workforce has to be reallocated from the usual tasks in warehouse logistics (such as packing, picking, storing, refilling, etc.) whenever a container arrives. This leads to a decreased output in the outbound part of the logistics process. In turn, customer will receive their orders at a reduced speed. Focus on customer satisfaction is key in the highly competitive online marketplace, so it is advised to fulfill customer orders as soon as possible. Also, after 3 hours of standing time, the container forwarding company will charge fees.

In order to keep the time and labor force bound to the unloading process at a minimum, it is necessary to look into some data, then draw conclusions from it and make suggestions to the stakeholders (logistics team leads, head of logistics).

First, I am looking for a measure that shows how much time it takes to unload a container, because that is the outcome variable that the company wants to influence. It would be easy to just stop the time it takes for the entire container to be emptied. But that leaves little room for a closer look at the details. I identified the following measure to be more useful:

- **Time per Palette (seconds)**
  The items in a container are loose and have to be stacked onto a palette and pulled out of the container on a palette truck. I measured the time in seconds it took for an empty palette to be fully stacked and pulled out of the container onto the loading ramp.

Further, there are some important choice variables. These are variables, which we have some control over:

- **Team Size**
  *The amount of workers that are present at the unloading bay for a given palette. This is a key variable, since it should not only be the main driver of the variable **Time per Palette (seconds)**, but from a management perspective, the variable that is the easiest and fastest to adapt to the circumstances.*

- **Item Volume (cubic centimeter)**
  *Here, we are looking at the geometrical dimension in centimeter of any item in a container(length*height*depth). In almost all cases, a given item has its' "own" palette. Meaning there won't be multiple different items stacked on a palette. It can be changed in the mid-term by negotiating with the suppliers to make changes with regard to the card box sizes and/or quantity of items contained within the card boxes.*

Here are some more variables, that serve as controls. These are "given", meaning it is prohibitively difficult for us to change those. We will need them later on for a multiple regression:

- **Item Weight (kilograms)**
  *This control variable contains the weight of an item in kilograms. Contrary to the variable **Item Volume (cubic centimeters)**, the weight of an item can hardly be changed by the company. If all, this change can only occur in the long-run.*

- **Item Quantity per Palette**
  How many of the same items should be stacked onto a palette is determined by the logistics team lead. Usually, the quantity of items per palette is maximized such that it doesn't exceed the height and weight limits of the storage shelves in the warehouse.

- **Palette Number**
  It can be argued, that the more palettes left the container, the more fatigued workers get. Also, it controls for the increasing distance a palette has to cover as the container is emptied in order to be counted as a completed palette (a palette counts as "completed" as soon as it passes the loading ramp). This variable is simply a running index: the first palette of a given container is 1 and each following palette adds 1 to this variable. Such that the last palette is assigned the number 50 when fifty total palettes were in a given container.

Data Generation

As I briefly mentioned, I gathered this data on my own. All the information I aggregated is not confidential. I didn't name company, suppliers, items, brands, peoples' names and any other identifying details.

The first step in the data generation process was it to prepare and print a sheet of paper with *Microsoft Excel* that included rows for observations and columns for the relevant variables and characteristics of each observation. Date and time were included on each of the sheets.

Next, it was simple data entry work with pen and paper. I used a stopwatch to note the exact time for the **Time per Palette (seconds)** variable. The data is kept in a continuous *mm:ss* format. **Team Size** was tracked on a per palette basis and noted in the data sheet. **Item Volume (cubic centimeters)**, **Item Weight (kilograms)** and **Item Quantity**

per Palette were extracted from the inventory management system and noted in the data sheet. **Palette Number** was built into to the data entry sheet template as an ascending whole number via *Excel*'s auto-fill feature.

Once the data was put to paper, all that is left for the data generation process was to manually transfer the data from paper to *Excel*. In total, 235 observations from September to October of 2019 were made. If you are interested in looking at the raw data, you'll find an *Excel* file named *rawDataContainerUnloading2019v1.xlsx* attached.

Data Cleaning

Excel

For the process of data cleaning in *Excel*, let me refer to the changelog. It documents all changes made in chronological order in a clear and concise manner. It is also accessible as *changelog_rawDataContainerUnloading2019.txt*.

Changes made to the raw data *Excel* sheet:

```
# Changelog
This txt-file contains notable changes to the raw data file

Version 2 (04.10.2021) "rawDataContainerUnloading2019v2.xlsx"
## New
    - Added column TotalPaletteCompletionTimeSeconds, which contains(
        PaletteCompletionTimeMinutes*60+PaletteCompletionTimeSeconds). This outputs the
         time for completion of a palette in seconds only. This value accumulates with
        each row untile PaletteNumber bounces back to 1 whenever the observations for a
         new container begin
    - Added column DiffTotalPaletteCompletionTimeSeconds, which contains (IF(
        PaletteNumber=1;DiffTotalPaletteCompletionTimeSeconds_CURRENTROW;
        DiffTotalPaletteCompletionTimeSeconds_CURRENTROW-
        DiffTotalPaletteCompletionTimeSeconds_ROWABOVE)). This removes the cumulation
        in column TotalPaletteCompletionTimeSeconds and outputs the time it takes to
        complete a palette in seconds
    - Added column ItemHeightCentimeter, ItemLengthCentimeter, ItemDepthCentimeter and
         used Excel's Text-to-Columns-feature to split the text in
        ItemDimensionsCentimeter at the "*" delimiter into each of the three added
        columns
    - Added filter in header column
    - Added column ItemVolumeCubiccentimeter, which contains (ItemHeightCentimeter*
        ItemLengthCentimeter*ItemDepthCentimeter). Skipped this calculation for
        observations with Date 09.09.2019, since no values for ItemDimensionsCentimeter
         where recorded, only ItemVolumeCubiccentimeter

## Changes
    - Changed column name ItemDimensions to ItemDimensionsCentimeter
    - Changed header font to bold font
    - Changed data format in ItemHeightCentimeter, ItemLengthCentimeter,
        ItemDepthCentimeter from text to number

## Fixes
    - Fixed wrong PaletteNumber data entries (originally started at 4 instead of 1)

Version 1 (03.10.2021) "rawDataContainerUnloading2019v1.xlsx"
## New
    - Added columns Date, PaletteCompletionTimeMinutes, PaletteCompletionTimeSeconds,
        TeamSize, ItemWeight, ItemQuantityPerPalette, PaletteNumber
    - Added 239 observations manually
```

```
## Changes
    - Changed column headers from general format to text format and aligned text
        orientation to center
    - Changed data format in column Date to date format
    - Changed data format in columns PaletteCompletionTimeMinutes,
        PaletteCompletionTimeSeconds, TeamSize, ItemWeight, ItemQuantityPerPalette,
        PaletteNumber to number format
```

After this quick and basic data cleaning and transforming in *Excel*, the file is saved separately as a csv-file and we explore further data cleaning and transformation opportunities with the programming language *R*.

R

In *R* we did the following steps to make the data ready for analysis:

Changes made to the raw data *Excel* sheet:

```
# Installing packages
install.packages("tidyverse")

# Loading packages
library(tidyverse)

# Check and set current working directory
getwd()
setwd("C:/Users/Matthias/Desktop/Google_Data_Analytics_Capstone_Project_Container")

# Importing and brief inspection of the data
ContainerData <- read_csv2("rawDataContainerUnloading2019v2.csv")
glimpse(ContainerData) # Number of rows and columns is correct. Data types are correct
    .

# Narrowing the data down to the columns of interest
core_data <- select(ContainerData, DiffTotalPaletteCompletionTimeSeconds, TeamSize,
    ItemWeight, ItemVolumeCubiccentimeter, ItemQuantityPerPalette, PaletteNumber)
core_data <- rename(core_data,
                    time_seconds = DiffTotalPaletteCompletionTimeSeconds,
                    team_size = TeamSize,
                    item_weight = ItemWeight,
                    item_volume = ItemVolumeCubiccentimeter,
                    palette_quantity = ItemQuantityPerPalette,
                    palette_number = PaletteNumber)
glimpse(core_data)

## Checking if values in columns "make sense"
# Are the unique values reasonable?
unique(core_data$team_size)
unique(core_data$palette_quantity)
unique(core_data$palette_number)
# Is the data range from lowest to highest value reasonable?
summary(core_data)

## Column time_seconds shows some questionable values (lowest value of five seconds to
     fully stack a palette is humanly impossible). Further investigation is needed.
write.csv2(core_data, "core_data_v1.csv") # Save copy of the original data file
sorted <- arrange(core_data, time_seconds) # Sort data by time_seconds, ascending
print(sorted$time_seconds) # Output shows that there are multiple observations in the
    time_seconds column that seem nonsensical
```

```
# To gage which nonsensical observations to drop, we introduce a new column that shows
    the time it takes to stack only one item onto a palette
core_data_v2 <- core_data %>%
  mutate(item_time = time_seconds / palette_quantity)

# Show properties of new column
summary(core_data_v2)
sorted_2 <- arrange(core_data_v2, item_time)
print(sorted_2$item_time)

# We make the ad-hoc decision to drop all observations with item_time < 3 seconds
write.csv2(core_data_v2, "core_data_v2.csv") # Save copy of the original data file
core_data_v3 <- filter(core_data_v2, item_time >= 3)
core_data_v3 <- rename(core_data_v3,
                       item_weight_kg = item_weight,
                       item_volume_cc = item_volume,
                       item_time_sec = item_time,
                       time_sec = time_seconds)
summary(core_data_v3)
glimpse(core_data_v3) # Now we have 199 observations left
write.csv2(core_data_v3, "core_data_v3.csv")
```

Analysis

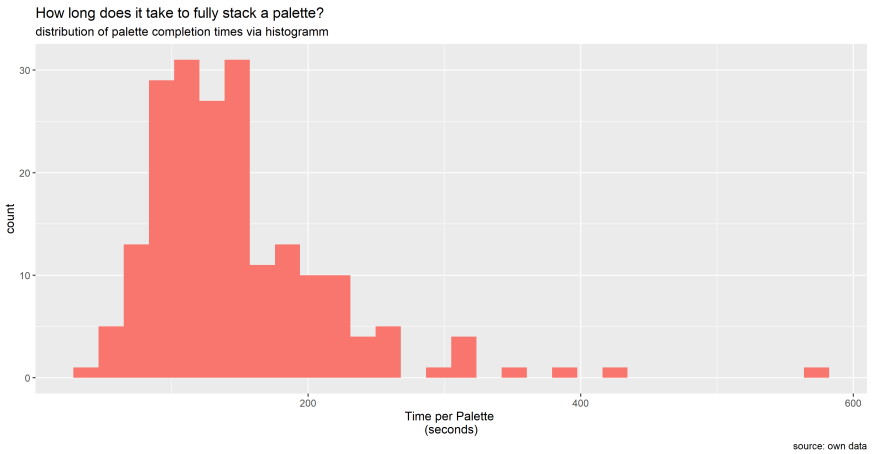
Now the data is cleaned and we can make the next step and look closer into what insights can be extracted from it. A first glimpse at some descriptive statistics reveals the following:

Table 1: Descriptive Statistics

Statistic	N	Mean	St. Dev.	Min	Pctl(25)	Pctl(75)	Max
Time per Palette (seconds)	199	147.930	69.676	44	102	177.5	580
Team Size	199	5.156	0.587	3	5	5	6
Item Volume (cubic centimeters)	199	59,942.750	18,573.130	30,723	41,538	85,800	85,800
Item Weight (kilograms)	199	6.633	2.129	3.040	4.350	8.840	8.840
Item Quantity per Palette	199	23.668	8.184	14	14	24	50
Palette Number	199	23.503	14.302	1	11.5	34	59
Time per Item (seconds)	199	6.658	3.465	3.000	4.310	7.857	30.500

Our outcome variable **Time per Palette (seconds)** has an average of around 148 seconds. Given that for any container there are about 40 to 60 palettes until it is empty, you could expect the process to take anywhere from 148 x 40= 5920 seconds (1 hour 39 minutes) and 148 x 60 = 8800 seconds (2 hours 28 minutes). So on average, we should also avoid the standing fee charged for containers that take more than 3 hours to empty. On bad days however, when **Time per Palette (seconds)** is above the 75% percentile of 178 seconds, we exceed the 3 hour limit if there is a total of 60 palettes. A scenario in which the total amount of palettes (not to be confused with the variable **Number of Palettes**, as that variable is just a running index for the individual palettes in a given container) is at the lower bound of 40 would force us to keep **Time per Palette (seconds)** below 270 seconds or below 1.7 standard deviations away from the mean (assuming a normal distribution, this would happen in 96% of the cases).

To complete our first impression of **Time per Palette (seconds)** we take a look at its distribution via histogram. We note that it is somewhat approaching a normal distribution with a slight skew to the left. Also, there are a few outliers to the right. Of course, this is something management would want to eliminate completely. From my experience I can tell that these scenarios occur when either break times conflict with the container schedule or one shift transitions to the other which leads to a lot of friction, ultimately resulting in overall time loss.



For our two choice variables, **Item Volume (cubic centimeters)** and **Team Size**. the situation is as follows: on average, there are 5 workers designated to work on the container, with a minimum of 3 and a maximum of 6. The vast majority of the time we have 5 workers present. On average, an item has the size of around 60,000 cubic centimeters, which is about two to three shoe boxes.

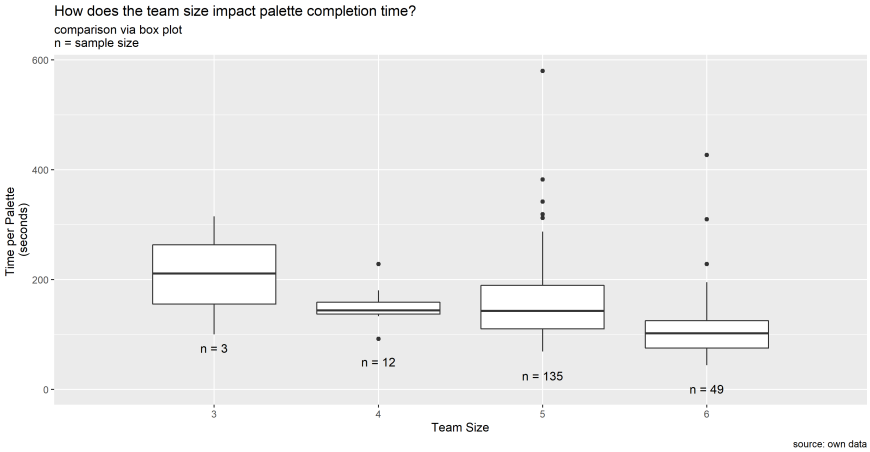
The remaining variables are controls, which will be used at a later stage of the analysis. They are not really changeable, so we won't spend any time discussing them any further.

Moving on, let's take closer look at how our main choice variable **Team Size** relates to **Time per Palette (seconds)**. Here, we compare all different observed values for **Team Size** and their respective mean values for **Mean Time per Palette (seconds)**:

Team Size	Mean Time per Palette (seconds)
3	208.6667
4	150.2500
5	158.3111
6	115.0408

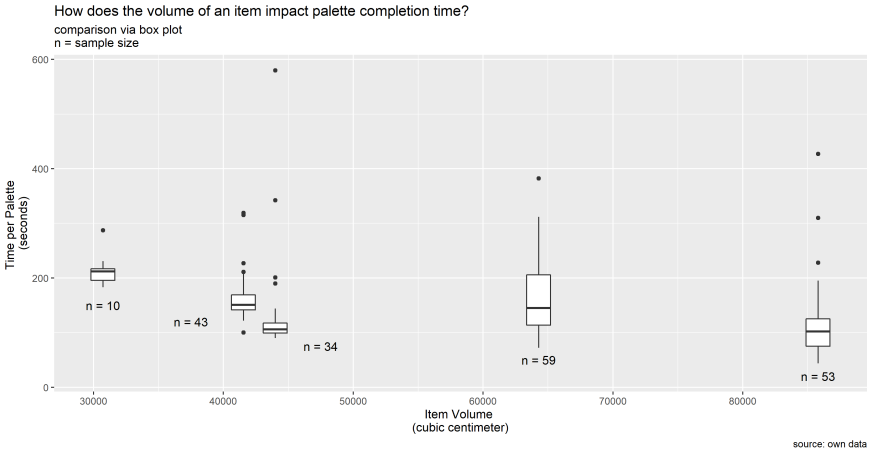
We can make out a trend that as we increase the amount of workers, we reduce the time it takes to empty a container. For teams of 4 and 5 there seems a reverse in effect, that is probably due to limitations in the data (sample sizes were too low for some values).

The following box plot gives a more detailed comparison of the relationship of both variables:



The plot shows the median as the horizontal bar within the box. The boxes contain the middle 50% of the observations at the given value for **Team Size**. The parts of the vertical line outside the box contain the upper and lower 25% of all values. The outlier points are well outside 99.65% of all made observations.
Here we see the data base is much stronger for observations at **Team Size == 5** and **Team Size == 6**. The difference between both sizes is around 53 seconds. Assuming 40[60] palettes per container, this would net 4[60] x 53 = 2120[3180] seconds or 35[53] minutes on average.
While more data is certainly needed to make robust claims, the whole picture aligns with first intuition that more workers lead to a lower duration for emptying containers.

We use a box plot as well to look at our secondary choice variable **Item Volume** in relation to **Time per Palette (seconds)**:



Sample sizes are higher than with **Team Size** values, but could still see improvement in future analysis. The general trend is obvious: the higher the item volume, the lower on average is the time it takes to finish one palette. We can make out that roughly 100 seconds are between the biggest item and the smallest item in our observations.

Our final analysis adds all the control variables mentioned earlier into the equation. We will conduct a multiple linear regression in two ways: first, with the nominal values, and second, with z-standardized values of the nominal values. This normalizes values such as **Item Volume (cubic centimeters)** down to their standard deviations, which allows for a better comparison between the marginal effects of variables.
Here are the results for the regular multiple linear regression:

Table 2: Multiple Linear Regression

	Dependent variable:
	time_sec
Team Size	−18.726
	(14.276)
Item Weight (kilograms)	12.571
	(8.002)
Item Volume (cubic centimeters)	−0.001
	(0.001)
Item Quantity per Palette	1.662
	(1.082)
Number of Palette	0.375
	(0.382)
Constant	191.219***
	(63.680)
Observations	199
R^2	0.111
Adjusted R^2	0.088
Residual Std. Error	66.529 (df = 193)
F Statistic	4.836*** (df = 5; 193)
Note:	*p<0.1; **p<0.05; ***p<0.01

The rows list all the explanatory variables, while the variable to be explained is displayed in the column header **Time per Palette (seconds**). The lower potion of the table highlights measures of how well the model explains our data. R^2 and Adjusted R^2 can range between 0 and 1. This measures how much of the variation of (**Time per Palette (seconds**) is explained by the variation in the explanatory variables. In our model that value is at around 10% which is quite low.
The next important measure is the F-Statistic, that is a measure for the joined significance of the estimators. Our value here is statistically significant at above 99.9% confidence.
Moving on to the values for our estimators, we find that beside the constant, none of our estimators are significantly different from zero. This doesn't necessary mean that there isn't a relationship between them, but more data is needed to verify or dismiss that hypothesis in this case.
Our choice variables show the minus signs that we already saw in the descriptive analysis. For **Team Size** a marginal change will lead to a decrease of around 19 seconds in **Time per Palette (seconds)**. The estimator of **Item Volume (cubic centimeters)** is a bit odd to interpret, that is why we turn to the z-standardized multiple regression now:

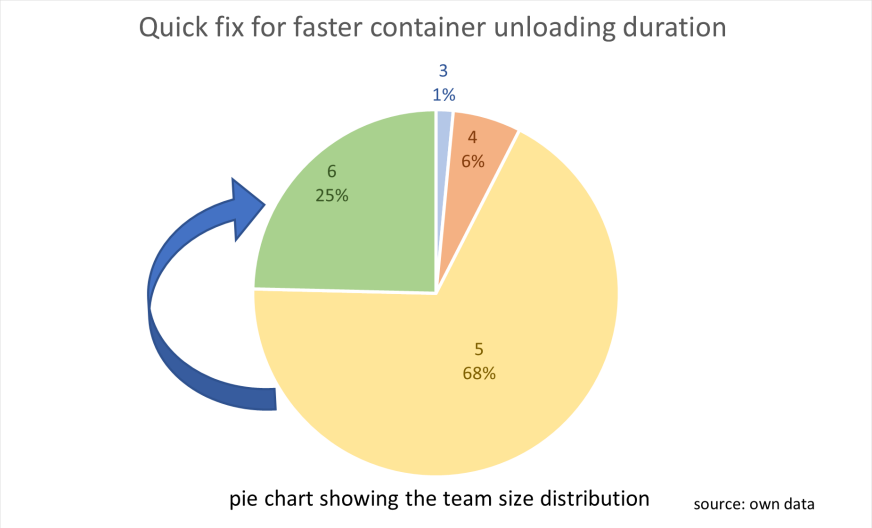
Table 3: Z-Standardized Multiple Linear Regression

	Dependent variable:
	z_time_sec
Team Size	−0.158
	(0.120)
Item Weight (kilograms)	0.384
	(0.244)
Item Volume (cubic centimeters)	−0.348
	(0.334)
Item Quantity per Palette	0.195
	(0.127)
Number of Palette	0.077
	(0.079)
Constant	0.000
	(0.068)
Observations	199
R^2	0.111
Adjusted R^2	0.088
Residual Std. Error	0.955 (df = 193)
F Statistic	4.836^{***} (df = 5; 193)
Note:	*p<0.1; **p<0.05; ***p<0.01

The measures for the model's accuracy remain unchanged. Key difference here is that the estimator have been transformed in such a way that their values now show how much a change of one standard deviation in any of the explanatory variables affects the outcome variable **Time per Palette (seconds)**, also denoted in standard deviations. Example: an increase of one standard deviation in the choice variable **Item Volume (cubic centimeter)** decreases **Time per Palette (seconds)** by -0.348 standard deviations. For the relationship **Team Size** and **Time per Palette (seconds)** an increase of one standard deviation leads to a decrease of -0.158 standard deviations. Reminder: the standard deviations of the individual variables can be found in the descriptive statistics table at the beginning of this section.

Recommendations for Decreasing the Unloading Time

As discussed earlier, we have two variables that we can adjust to speed up the unloading process. Increasing the **Team Size** is fastest way make that adjustment and empty the container quicker. Team leads simply have to assign one additional worker to that task. Ideally, we'd aim for six workers. Let's see how team sizes were distributed in the data presented in this report:



Here we see that only in one quarter of the cases six workers were allocated to the container. It is good to note that in more than two thirds of the cases five workers were present and rarely less than that, but there is still a lot of room for improvement.
It has to be noted, however, that just blindly sending in more workers is not a no-brainer. The time of any additional worker must be "worth" it. There is a rule that should be established to estimate whether that shift in resources is needed: total time saved for present workers > time investment for an additional worker. Example: Team leads are contemplating whether they should assign an additional worker to the container. Past data shows that going from 5 workers to 6 workers could save 30 minutes. For the individual worker the task at the container unloading bay costs 2 hours of working time. Total time saved for all workers would be $5x30 = 150$ minutes or 2.5 hours. 2.5 hours is greater than the two hours of the time the single additional worker is tied up in the unloading task. According to this rule, the worker would help unloading the container.

Item Volume is not as easy to change as the **Team Size**. This is more of a mid-term solution, but as we saw, changing its standard deviation does much more to overall duration. Making adjustments in that regard would need communication with suppliers to either persuade them to store items in large boxes so that more items can be stacked on a palette in one motion or ask them to pre-stack the items on palettes. Why would suppliers do this? First, we have some power on the market as a reasonably large retailer. Second, for the pre-stacking argument: this would speed up their loading process as well.

To summarize our recommendations:

- Increase **Team Size** to 6 for most of the time. Maybe look into collecting data for even more workers to see if diminishing returns kick in at some point.
- Use rule *total time saved for present workers > time investment for an additional worker* based on past data with the correct parameters
- Leverage market power to make suppliers adjust their shipping methods

What's Next?

For future analysis on the same topic we'd certainly want a larger sample size overall. On the same note, a greater variation in the choice variables would give us broader data to draw conclusions from.
Once our recommended changes are implemented, we should gather the data and see if we really saved more time in the entire process and then decide if these changes are worth it.
For **Team Size** it would be interesting to know if there are diminishing returns in terms of **Time per Palette (seconds)**

when more and more workers are assigned to the task.