Description of Imputation Methodology

Section 1 - Overview

This R Markdown document will demonstrate the calculation of imputation for each of the below measures:

- *Available_Stay_Unit_Nights*
- *Occupied_Nights*
- *Guest_Nights*
- *Guest_Arrivals*
- *Domestic_Nights* (and by extension, International Nights, which is $Guest\_Nights - Domestic\_Nights$)

We will first inspect the first 6 rows of the data at the start of the imputation process. Where no data for a particular measure is provided, this is indicated by `NA`. For these properties, there is a mix of responses and non responses for each the data fields shown (non imputed values).

The imputation process will compute an imputation to fill each `NA` with an imputed value, in the order of columns as per the table below (left to right). The monthly capacity of each property is always known. This is done by computing an operational ratio, and multiplying the ratio with the value of the field preceding the field for which an imputation is being computed.

For example, to calculate an imputation for `available_stay_unit_nights` for property 8440, we first calculate an operational ratio given by non-imputed data from donor properties, and multiply that ratio with the value of the field `monthly_capacity` for property 8440.

This hotel is also missing data for `occupied_nights` and so this field will need to be computed. As per the previous imputation, an operational ratio is calculated and is multiplied with the imputed value of the preceding field, `available_stay_unit_nights`

```
table_properties_head
```

order id	fresh id	name	territorial authority	rto	property type	property tracking score (>=0 open, <0 closed)	data source	imputation cell for occupancy and ratio of available to capacity	imputation cell for guest nights and arrivals	imputation cell for origin	new and improved formula for z score
1	1706	Queenstown Park Boutique Hotel	Queenstown-Lakes District	Queenstown RTO	Hotels (over 20 units)	8	2.3	Queenstown RTOHotels (over 20 units)	Queenstown RTOHotels (over 20 units)	Queenstown RTOHotels (over 20 units)	0.0954427
2	8440	Executive Residence Boutique Hotel	Dunedin City	Dunedin RTO	Hotels (over 20 units)	8	DB	Dunedin RTOHotels (over 20 units)	Dunedin RTOHotels (over 20 units)	Dunedin RTOHotels (over 20 units)	0.0000000
3	434	Greenlane Suites	Auckland	Auckland RTO	Hotels (over 20 units)	8	2.3	Auckland RTOHotels (over 20 units)	Auckland RTOHotels (over 20 units)	Auckland RTOHotels (over 20 units)	1.3931924
5	7054	Awaroa Lodge	Tasman District	Nelson Tasman RTO	Hotels (over 20 units)	-3	2.3	Nelson Tasman RTOHotels (over 20 units)	Nelson Tasman RTOHotels (over 20 units)	Nelson Tasman RTOHotels (over 20 units)	0.0000000
6	5806	Peppers Parehua Martinborough	South Wairarapa District	Wairarapa RTO	Hotels (over 20 units)	8	2.3	Hotels (over 20 units)	Hotels (over 20 units)	Hotels (over 20 units)	1.4224072
7	731	Wai Ora Lakeside Spa Resort	Rotorua District	Rotorua RTO	Hotels (over 20 units)	8	2.3	Rotorua RTOHotels (over 20 units)	Rotorua RTOHotels (over 20 units)	Rotorua RTOHotels (over 20 units)	0.6985171

order id	available stay unit nights	occupied nights	guest nights	guest arrivals	domestic	international	monthly capacity	imputed available stay unit nights	imputed occupied nights	imputed guest nights	imputed guest arrivals	imputed domestic	imputed international	margin of error calculations	exclude from imputation (manual)	flagged as a response
1	660	369	727	NA	727	0	660	660	369	727	264	727	0	0.55909090909090908	NA	1
2	NA	NA	NA	NA	NA	NA	720	720	349	569	346	567	2	NA	NA	0
3	750	632	NA	NA	NA	NA	750	750	632	992	592	904	88	0.84266666666666667	NA	1
5	0	0	0	0	0	0	780	0	0	0	0	0	0	NA	NA	1
6	840	685	1384	810	1350	34	840	840	685	1384	810	1350	34	0.81547619047619047	NA	1
7	900	366	920	240	920	0	900	900	366	920	240	920	0	0.40666666666666668	NA	1

Section 2 - Outlier Processing

We define outliers as unusual values that have the potential to materially influence regional and national results. Values that we considered to be plausible but highly influential were left in the data but are not included in the imputation process for other properties. We currently recognise two types of outliers:

- Managed isolation and quarantine facilities: Because of the short-term nature of the isolation stay, these hotels are included in the counts of available stay-units, occupied nights, guest nights, and guest arrivals.
- Properties with extreme (high/low) occupancy rates: We calculate the z-score for occupancy observations within each property type and multiply this number by 1000 * share of capacity for that property within its type. This provided a measure of the relative influence of a property's performance on the results. Those with a value greater than 10 are flagged as outliers.

2.1 - Calculate Standard Deviation of Occupancy Rate for Each Property Group

```r
properties = properties %>%
  mutate(propert_occupancy_rate = ifelse(
    flagged_as_a_response == 1,
    non_imputed_occupied_nights/non_imputed_available_stay_unit_nights,
    NA))

group_sd_occupancy_rates = properties %>%
  group_by(property_type) %>%
  summarise(sd_occupancy_rate = sd(propert_occupancy_rate, na.rm = T))
```

The table below shows the standard deviations of occupancy rates for each property group.

```r
kbl_df
```

property_type	sd_occupancy_rate
Backpackers (over 20 units)	0.20
Holiday parks and campgrounds	0.18
Hotels (over 20 units)	0.20
Motels and serviced apartments (6-20 units)	0.22
Motels and serviced apartments (over 20 units)	0.20
Other accommodation (over 5 units)	0.24

Merge properties table with the above standard deviation table.

```r
properties = left_join(
  x = properties,
  y = group_sd_occupancy_rates,
  by = c('property_type' = 'property_type')
)
```

2.2 - Calculate Aggregate Measures for Each Property Group

Calculate the sum of `available_stay_unit_nights` and `occupied_nights` for each property group.

```
# Statistics for Target Population
target_pop_stats = properties_all %>%
  group_by(property_type) %>%
  summarise(
    # Total number of available stay unit nights across each property type
    stats_sum_available_stay_unit_nights =  sum(imputed_available_stay_unit_nights, na.rm = T),

    # Total number of occupied nights across each property type
    stats_sum_occupied_stay_unit_nights =   sum(imputed_occupied_nights, na.rm = T)
    ) %>%

  # Calculate occupancy rate across each property type
  mutate(stats_occupancy_rate = stats_sum_occupied_stay_unit_nights / stats_sum_available_stay_unit_nights)
```

```
kbl_df
```

property type	stats sum available stay unit nights	stats sum occupied stay unit nights	stats occupancy rate
Backpackers (over 20 units)	410179	127089	0.31
Holiday parks and campgrounds	1145807	243264	0.21
Hotels (over 20 units)	827962	443085	0.54
Motels and serviced apartments (6-20 units)	375192	215845	0.58
Motels and serviced apartments (over 20 units)	323278	187178	0.58
Other accommodation (over 5 units)	89857	27952	0.31

```
# Merge with target pop stats to get the group measures
properties = left_join(
  x = properties,
  y = target_pop_stats,
  by = c("property_type" = "property_type")
)
```

2.3 - Calculate Z-Score

```
# Calculate Z-score
properties = properties %>%
  mutate(
    z_score = abs(
      (non_imputed_occupied_nights / non_imputed_available_stay_unit_nights) - stats_occupancy_rate
    )
    /
      sd_occupancy_rate
    *
      (imputed_available_stay_unit_nights / stats_sum_available_stay_unit_nights)
    * 1000
  )
```

2.4 - Remove Properties based on z-score

```
properties  = properties %>%
  mutate(outlier_excluded = ifelse(z_score > 10, 'excluded', 'not_excluded'))
```

The below table shows two hotel properties in which one has been determined to be an outlier based on its occupancy rate.

```
kbl_df
```

property type	property occupancy rate	stats occupancy rate	z score	outlier excluded	non imputed occupied nights	non imputed available stay unit nights
Hotels (over 20 units)	0.14	0.54	10.11	excluded	610	4320
Hotels (over 20 units)	0.60	0.54	3.22	not_excluded	5037	8400

Section 3 - Imputation Process

3.1 Derive Donor Property Set

This step involves finding the set of other comparable properties for each property. The set of comparable properties will consist of either

- properties sharing the same property type within the same RTO, or
- properties sharing the same property type nationwide.

The former is used when there are sufficient respondents who have submitted data for a given attribute. If there are insufficient respondents, data from all respondents across the country sharing the same property type is used for imputation. This is done to protect the confidentiality of respondents, and also to increase the diversity of responses and to reduce the risk of one or two properties exerting undue influence on the imputation calculation.

We define property A as a property for which an imputation calculation is required. We also define the donor property set as the set of properties which will 'donate' or provide values for the calculation of the imputation for property A.

When calculating the imputation for any given field, the donor property set is determined by :

- Property Type
- RTO - If there are three or more active and responding properties of the same type within the same RTO (including property A), then the donor set will consist of these properties (excluding property A); otherwise the donor set will consist of all properties of the same property type across New Zealand.
- Whether a property has submitted data for a specified field. This field depends on the field which is being imputed. For example, when calculating the imputation for the field *Available_Stay_Unit_Nights* for hotels in the Auckland RTO, the donor property set consists of all hotels in Auckland RTO which have submitted data for *Occupied_Nights*. If there are less than three properties which fulfill this criteria, then the donor property set would instead consist of hotels across New Zealand.

Field to be imputed	Field to determine donor
Available_Stay_Unit_Nights	*Occupied_Nights*
Occupied_Nights	*Occupied_Nights*
Guest_Nights	*Guest_Arrivals*
Guest_Arrivals	*Guest_Arrivals*
Domestic_Nights	*Domestic_Nights*

The following code examples will demonstrate the process for calculating the ratio which is used to impute the field *Occupied_Nights*; This code is then generalised to apply for all of the imputation calculations.

We first need to count the number of properties which fulfill the three matching criteria as defined above.

```r
df_grouped = properties%>%
  group_by_at(c("rto", "property_type")) %>%
  summarise(

    # Count properties with at least and at least one occupied night - This is used to determine 'donor_set_type'
    count = sum( (non_imputed_occupied_nights > 0), na.rm = T),
     )%>%
  ungroup()

df_grouped %>%
  filter(rto == "Bay of Plenty RTO") %>%
  kbl() %>%
  kable_classic(html_font = "Cambria") %>%
  kable_styling(
    bootstrap_options = c("striped", "hover", "condensed", "responsive")) %>%
  scroll_box(width = "100%")
```

rto	property_type	count
Bay of Plenty RTO	Backpackers (over 20 units)	3
Bay of Plenty RTO	Holiday parks and campgrounds	7
Bay of Plenty RTO	Hotels (over 20 units)	0
Bay of Plenty RTO	Motels and serviced apartments (6-20 units)	17
Bay of Plenty RTO	Motels and serviced apartments (over 20 units)	5
Bay of Plenty RTO	Other accommodation (over 5 units)	1

We then proceed to name the donor property set. Based on the counts shown above, we must confidentialise the rto + property groups which have less than three properties. These groups will have group type `donor_set_type = 'property_type'` as they consist of properties from across the country sharing the same property type. Groups which have counts of three or greater will consist of properties only within the same rto, and so have `donor_set_type = rto_and_property_type`

```
# Determine comp set type and name
df_grouped = df_grouped %>%
  mutate(
    donor_set_type = ifelse(
      count >= 3,
      "rto_and_property_type",
      "property_type"),

    donor_set_name = ifelse(
      count >= 3,
      paste(rto, property_type, sep = ""),
      property_type)
  )
```

```
table_df_grouped
```

rto	property_type	count	donor_set_type	donor_set_name
Bay of Plenty RTO	Backpackers (over 20 units)	3	rto_and_property_type	Bay of Plenty RTOBackpackers (over 20 units)
Bay of Plenty RTO	Holiday parks and campgrounds	7	rto_and_property_type	Bay of Plenty RTOHoliday parks and campgrounds
Bay of Plenty RTO	Hotels (over 20 units)	0	property_type	Hotels (over 20 units)
Bay of Plenty RTO	Motels and serviced apartments (6-20 units)	17	rto_and_property_type	Bay of Plenty RTOMotels and serviced apartments (6-20 units)
Bay of Plenty RTO	Motels and serviced apartments (over 20 units)	5	rto_and_property_type	Bay of Plenty RTOMotels and serviced apartments (over 20 units)
Bay of Plenty RTO	Other accommodation (over 5 units)	1	property_type	Other accommodation (over 5 units)

3.2 - Compute Operational Ratios for each possible donor group

For each `donor_set_name`, we will compute an operational ratio which will be used as an imputation ratio for all accomodation providers within this group (where an imputation is required).

Example: Available Stay Unit Nights: In this example, the occupancy rate is being calculated. This operational ratio will be used to impute the `available stay unit nights` for any properties in this group where data has not been provided for this field.

The numerator represents the total number of occupied nights and the denominator represents the total number of available stay unit nights across all the properties belonging to `donor_set_name`.

$$r = \frac{\sum_{i \in \{i || available\_stay\_unit\_nights_i > 0\}} available\_stay\_unit\_nights_i}{\sum_{i \in \{i || available\_stay\_unit\_nights_i > 0\}} monthly\_capacity_i}$$

where i represents the $i - th$ donor property, and N is the total number of properties within the donor property set.

```r
df_grouped = properties%>%

  # Filter out properties with outlying occupancy rates (z-score > 10)

  filter(outlier_excluded != 'excluded') %>%
  group_by_at(c("rto", "property_type")) %>%
  summarise(

    # Count properties with at least one occupied night - This is used to determine `donor_set_type`
    count = sum(non_imputed_available_stay_unit_nights > 0, na.rm = T),

    # Numerator
      sum_non_imputed_available_stay_unit_nights = sum(non_imputed_available_stay_unit_nights[
        (non_imputed_available_stay_unit_nights > 0) & (non_imputed_occupied_nights > 0)
        ], na.rm = T),

    # Denominator
    sum_monthly_capacity = sum(monthly_capacity[
      (non_imputed_available_stay_unit_nights > 0) & (non_imputed_occupied_nights > 0)
      ], na.rm = T)
     )%>%
  ungroup()
```

```r
# Determine comp set type and name
df_grouped = df_grouped %>%
  mutate(
    donor_set_type = ifelse(
      count >= 3,
      "rto_and_property_type",
      "property_type"),

    donor_set_name = ifelse(
      count >= 3,
      paste(rto, property_type, sep = ""),
      property_type)
  )


# For donor_set_type == 'property_type'
aggregations_comp_set_property.df <- df_grouped %>%
  group_by(donor_set_name = property_type) %>%
  summarise(
    sum_non_imputed_available_stay_unit_nights = sum(sum_non_imputed_available_stay_unit_nights),
    sum_monthly_capacity = sum(sum_monthly_capacity)
  )

aggregations_comp_set_property.df
```

```
## # A tibble: 6 x 3
##   donor_set_name                  sum_non_imputed_available_st~ sum_monthly_capa~
##   <chr>                                                   <dbl>             <dbl>
## 1 Backpackers (over 20 units)                            133735            145350
## 2 Holiday parks and campgrounds                          592883            712740
## 3 Hotels (over 20 units)                                 453360            479790
## 4 Motels and serviced apartment~                         148101            157380
## 5 Motels and serviced apartment~                         133531            144330
## 6 Other accommodation (over 5 u~                          27428             35250
```

```r
imputation_per_property_type.df = inner_join(
  x = df_grouped %>% select(donor_set_name, donor_set_type, rto, property_type, count),
  y = aggregations_comp_set_property.df,
  by = c("donor_set_name" = "donor_set_name")
)

# For donor_set_type == 'rto_and_property_type'
imputation_per_rto_and_property_type.df = df_grouped %>%
  filter(donor_set_type == "rto_and_property_type")

imputation_schedule_available_stay_unit_nights = rbind(imputation_per_property_type.df, imputation_per_rto_and_property_type.df)

# Compute Ratio
num = imputation_schedule_available_stay_unit_nights$sum_non_imputed_available_stay_unit_nights
den = imputation_schedule_available_stay_unit_nights$sum_monthly_capacity
imputation_schedule_available_stay_unit_nights$ratio_available_stay_unit_nights = num / den

imputation_schedule_available_stay_unit_nights %>%
  filter(count <3)
```

```
## # A tibble: 51 x 8
##    donor_set_name donor_set_type rto   property_type count sum_non_imputed~
##    <chr>          <chr>          <chr> <chr>         <int>            <dbl>
##  1 Other accommo~ property_type  Bay ~ Other accomm~     1            27428
##  2 Backpackers (~ property_type  Cent~ Backpackers ~     1           133735
##  3 Hotels (over ~ property_type  Cent~ Hotels (over~     1           453360
##  4 Backpackers (~ property_type  Clut~ Backpackers ~     1           133735
##  5 Holiday parks~ property_type  Clut~ Holiday park~     1           592883
##  6 Backpackers (~ property_type  Coro~ Backpackers ~     2           133735
##  7 Motels and se~ property_type  Coro~ Motels and s~     2           133531
##  8 Holiday parks~ property_type  Dune~ Holiday park~     2           592883
##  9 Other accommo~ property_type  Dune~ Other accomm~     2            27428
## 10 Backpackers (~ property_type  East~ Backpackers ~     1           133735
## # ... with 41 more rows, and 2 more variables: sum_monthly_capacity <dbl>,
## #   ratio_available_stay_unit_nights <dbl>
```

```
table_imputation_schedule_available_stay_unit_nights_bop
```

donor set name	donor set type	rto	property__type	count of properties	sum available stay	donor set name	ratio occupancy
Other accommodation (over 5 units)	property__type	Bay of Plenty RTO	Other accommodation (over 5 units)	1	27428	35250	0.7780993
Bay of Plenty RTOBackpackers (over 20 units)	rto_and__property__type	Bay of Plenty RTO	Backpackers (over 20 units)	3	4688	5220	0.8980843
Bay of Plenty RTOHoliday parks and campgrounds	rto_and__property__type	Bay of Plenty RTO	Holiday parks and campgrounds	7	36053	40860	0.8823544
Bay of Plenty RTOMotels and serviced apartments (6-20 units)	rto_and__property__type	Bay of Plenty RTO	Motels and serviced apartments (6-20 units)	17	5977	6720	0.8894345
Bay of Plenty RTOMotels and serviced apartments (over 20 units)	rto_and__property__type	Bay of Plenty RTO	Motels and serviced apartments (over 20 units)	5	4077	4920	0.8286585

The formula to derive the operational ratio for each imputation calculation is given in the table below.

Field	Description	Formula
Available Stay Unit Nights	Stay units made available for short-term guests multiplied by the number of days of availability. This includes occupied and unoccupied stay units, and those located for managed isolation.	$r = \frac{\sum_{i \in \{i \| available\_stay\_unit\_nights_i > 0\}} available\_stay\_unit\_nights_i}{\sum_{i \in \{i \| available\_stay\_unit\_nights_i > 0\}} monthly\_capacity_i}$
Occupied Nights	Stay units occupied by short-term guests during the month.	$r = \frac{\sum_{i \in \{i \| occupied\_nights_i > 0\}} occupied\_nights_i}{\sum_{i \in \{i \| occupied\_nights_i > 0\}} available\_stay\_unit\_nights_i}$
Guest Nights	Cumulative number of nights spent short-term guests. Equivalent to one guest spending one night in a property. This includes managed isolation guests.	$r = \frac{\sum_{i \in \{i \| guest\_nights_i > 0\}} guest\_nights_i}{\sum_{i \in \{i \| guest\_nights_i > 0\}} occupied\_nights_i}$
Guest Arrivals	Number of individual short-term guests checking in during the reference month. This includes managed isolation guests.	$r = \frac{\sum_{i \in \{i \| guest\_arrivals_i > 0\}} guest\_arrivals_i}{\sum_{i \in \{i \| guest\_arrivals_i > 0\}} guest\_nights_i}$
Domestic Nights	Nights spent by guests that were identified as being from New Zealand.	$r = \frac{\sum_{i \in \{i \| domestic\_nights_i \geq 0\}} domestic\_nights_i}{\sum_{i \in \{i \| domestic\_nights_i \geq 0\}} guest\_nights_i}$

We parameterise the The below function computes an imputation schedule for a particular imputation measure.

```
compute_imputation_schedule <- function(
  df,
  numerator_field,
  denominator_field,
  imputation_name,
  imputation_comp_set_field
  ){

  df_grouped = df %>%
    # filter_at(vars(numerator_field), ~ .>0) %>%
    group_by_at(c("rto", "property_type")) %>%
    summarise_(
      count = interp(
        ~sum((numerator_var > 0) & (imputation_comp_set_var > 0), na.rm = T),
        numerator_var = as.name(numerator_field),
        imputation_comp_set_var = as.name(imputation_comp_set_field)
        ),

      numerator = interp(~sum(numerator_var[(numerator_var > 0) & (imputation_comp_set_var > 0)], na.rm = T),
                         imputation_comp_set_var = as.name(imputation_comp_set_field),
                         numerator_var = as.name(numerator_field)
                         ),

      denominator = interp(~sum(denominator_var[(numerator_var > 0) & (imputation_comp_set_var >0)], na.rm = T),
                           imputation_comp_set_var = as.name(imputation_comp_set_field),
                           denominator_var = as.name(denominator_field),
                           numerator_var = as.name(numerator_field))
      ) %>% ungroup()

  # Determine comp set type and name
  df_grouped = df_grouped %>%
    mutate(
      donor_set_type = ifelse(
        count >= 3,
        "rto_and_property_type",
        "property_type"),
      donor_set_name = ifelse(
        count >= 3,
        paste(rto, property_type, sep = ""),
        property_type)
    )

  aggregations_comp_set_property.df <- df_grouped %>%
  group_by(donor_set_name = property_type) %>%
  summarise(
    numerator = sum(numerator),
    denominator = sum(denominator)
  )

  imputation_per_property_type.df = inner_join(
    x = df_grouped %>%
      select(donor_set_name, donor_set_type, rto, property_type, count),
    y = aggregations_comp_set_property.df,
    by = c("donor_set_name" = "donor_set_name")
  )

  imputation_per_rto_and_property_type.df = df_grouped %>%
    filter(donor_set_type == "rto_and_property_type")

  # Row bind the imputation schedules for both `rto_and_property_type` and either `rtop` or `property_type`
  imputation_schedule = rbind(imputation_per_property_type.df, imputation_per_rto_and_property_type.df)

  imputation_schedule = imputation_schedule %>%
    rename_at("donor_set_type", ~paste(imputation_name, "donor_set_type", sep = "_")) %>%
    rename_at("donor_set_name", ~paste(imputation_name, "donor_set_name", sep = "_")) %>%
    rename_at("count", ~ paste(imputation_name, "imputation_donors_count", "count", sep = "_")) %>%

  # Calculate Ratio
  mutate(ratio = numerator/denominator) %>%

  # Rename Columns
  rename_at("numerator", ~ paste(imputation_name, "sum", numerator_field,  sep = "_")) %>%
  rename_at("denominator", ~ paste(imputation_name,"sum", denominator_field,  sep = "_")) %>%
  rename_at("ratio", ~paste(imputation_name, "ratio", sep = "_"))
```

```r
  return(imputation_schedule)
}
```

Our five imputation schedules are as follows. Note that in the imputation for available stay unit nights and occupied nights, properties with outlying occupancy rates are excluded.

```r
imputation_available_stay_unit_nights = compute_imputation_schedule(
  df = properties %>% filter(outlier_excluded != 'excluded'), # Exclude properties with z-score > 10
  numerator_field = "non_imputed_available_stay_unit_nights",
  denominator_field = "monthly_capacity",
  imputation_name = "available_stay_unit_nights",
  imputation_comp_set_field = "non_imputed_occupied_nights"
)

imputation_occupied_nights = compute_imputation_schedule(
  df = properties %>% filter(outlier_excluded != 'excluded'),
  numerator_field = "non_imputed_occupied_nights",
  denominator_field = "non_imputed_available_stay_unit_nights",
  imputation_name = "occupied_nights",
  imputation_comp_set_field = "non_imputed_occupied_nights"
)

imputation_guest_nights = compute_imputation_schedule(
  df = properties,
  numerator_field = "non_imputed_guest_nights",
  denominator_field = "non_imputed_occupied_nights",
  imputation_name = "guest_nights",
  imputation_comp_set_field = "non_imputed_guest_arrivals"
)

imputation_guest_arrivals = compute_imputation_schedule(
  df = properties,
  numerator_field = "non_imputed_guest_arrivals",
  denominator_field = "non_imputed_guest_nights",
  imputation_name = "guest_arrivals",
  imputation_comp_set_field = "non_imputed_guest_arrivals"
)
```

```
imputation_domestic_nights = compute_imputation_schedule(
  df = properties,
  numerator_field = "non_imputed_domestic",
  denominator_field = "non_imputed_guest_nights",
  imputation_name = "domestic_nights",
  imputation_comp_set_field = "non_imputed_domestic"
)
```

3.3 - Calculating the Imputation

The below code demonstrates how the operational ratio is used to compute an imputation for `guest_nights`. In this scenario, property 1318 is a property in the Bay of Plenty RTO with property type `Holiday parks and campgrounds.` We look up the property and the rto in the respective imputation schedule for `occupied_nights`. Since there are 6 other properties with the same property type within the Bay of Plenty RTO, the set of donors will consist of those 6 properties.

```
table_1318
```

fresh_id	rto	property type	occupied nights	guest nights
1318	Bay of Plenty RTO	Holiday parks and campgrounds	3150	NA

To compute the imputed occupied nights, we simply multiply the operational ratio with with the `occupied_nights` of property 1318:

$$guest\_nights_{1318} = r \times occupied\_nights_{1318}$$

where

$$r = \frac{\sum_{i \in \{i | guest\_nights_i > 0\}} guest\_nights_i}{\sum_{i \in \{i | guest\_nights_i > 0\}} occupied\_nights_i}$$

where i represents the $i-th$ donor property, and N is the total number of properties within the donor property set which comprises of (Holiday parks and camp-grounds in Bay of Plenty.

```r
filtered.df = properties%>%
  filter(fresh_id == 1318) %>%
  select(fresh_id, rto, property_type, non_imputed_occupied_nights, non_imputed_guest_nights)

# Lookup the rto and property type in the imputation comp set table
filtered.df = left_join(
  x = filtered.df,
  y = imputation_guest_nights,
  by = c("rto", "property_type")
)
```

```r
# Multiply the ratio by the respective field
filtered.df$imputed_guest_nights = filtered.df$guest_nights_ratio * filtered.df$non_imputed_occupied_nights

# Final Imputed Value
round(filtered.df$imputed_guest_nights, 0)
```

```
## [1] 8231
```

```r
table_1318_imputed
```

fresh id	rto	property type	occupied nights	guest nights	guest nights donor set name	guest nights donor set type	guest nights imputation donors count count	guest nights sum guest nights	guest nights sum occupied nights	guest nights ratio	imputed guest nights
1318	Bay of Plenty RTO	Holiday parks and camp-grounds	3150	NA	Bay of Plenty RTOHoliday parks and camp-grounds	rto_and_property_type	5	12414	4751	2.612924	8230.709