```r
library(tidyverse)
library(knitr)
```

Sampling Frame

Download the data

```r
# file path to csv with addresses
aru_file_path <- "https://opendata.arcgis.com/datasets/c3c0ae91dca54c5d9ce56962fa0dd645_

ap_file_path <- "https://opendata.arcgis.com/datasets/aa514416aaf74fdc94748f1e56e7cc8a_0

# create a directory for downloading the data
if (!dir.exists("data/")) {
  dir.create("data")
}

# if the data doesn't already exist, download the data
if (!file.exists("data/aru.csv")) {
  download.file(aru_file_path, "data/aru.csv")
}

if (!file.exists("data/ap.csv")) {
  download.file(ap_file_path, "data/ap.csv")
}
```

Address Residential Units

The first dataset is Address Residential Units

The dataset does not contain a variable for quadrant, so we extract quadrant from the full address.

```r
aru <- read_csv("data/aru.csv") %>%
  rename_all(tolower) %>%
  select(unit_id, address_id, fulladdress, status, unitnum, unittype)

# extract quadrant
aru <- aru %>%
  mutate(quadrant = str_sub(fulladdress, start = -2, end = -1))
```

Address Residential Units contains residential units with status set to "RETIRED". We drop these cases as well.

```r
count(aru, status) %>%
  kable()
```

status	n
ACTIVE	244046
ASSIGNED	47
RETIRE	7087

```
aru <- aru %>%
  filter(status != "RETIRE")
```

Adress Points

```
# load the data and convert the variable names to lower case
ap <- read_csv("data/ap.csv", guess_max = 10000) %>%
  rename_all(tolower) %>%
  select(address_id, status, type_, entrancetype, quadrant, fulladdress, objectid_1, ass
```

Address Points contains residential units, non-residential units, and mixed-use units. Residential units and mixed-use units contain residences that belong to our sampling frame. We drop non-residential units.

```
count(ap, res_type) %>%
  kable()
```

res_type	n
MIXED USE	473
NON RESIDENTIAL	15807
RESIDENTIAL	131370

```
ap <- ap %>%
  filter(res_type != "NON RESIDENTIAL")
```

Address points contains residential units with status set to “RETIRED”. We drop these cases as well.

```
count(ap, status) %>%
  kable()
```

status	n
ACTIVE	128490
ASSIGNED	668
RETIRE	2675
TEMPORARY	10

```r
ap <- ap %>%
  filter(status != "RETIRE")
```

After the above filtering, there are 98 observations from Address Points and 3,706 observations in Address Residential Units that have missing addresses. We investigated joining the two datasets on `address_id` to fill in the address but all records missing an address in one dataset were missing an address in the other dataset.

We dropped the missing values which represented about 1.5 percent of observations in Address Residential Units and 0.07 percent of observations in Address Points.

```r
ap <- ap %>%
  filter(!is.na(fulladdress))

aru <- aru %>%
  filter(!is.na(fulladdress))
```

```r
missing_aru <- filter(aru, is.na(fulladdress))

# join ap to aru missing

missing_aru <- left_join(missing_aru, ap, by = "address_id")

anti_join(missing_aru, ap, by = "address_id")
```

```
## # A tibble: 0 x 27
## # ... with 27 variables: unit_id <dbl>, address_id <dbl>,
## #   fulladdress.x <chr>, status.x <chr>, unitnum <chr>, unittype <chr>,
## #   quadrant.x <chr>, status.y <chr>, type_ <chr>, entrancetype <chr>,
## #   quadrant.y <chr>, fulladdress.y <chr>, objectid_1 <dbl>,
## #   assessment_nbhd <chr>, cfsa_name <chr>, census_tract <chr>,
## #   vote_prcnct <chr>, ward <chr>, zipcode <dbl>, anc <chr>,
## #   census_block <chr>, census_blockgroup <chr>, latitude <dbl>,
## #   longitude <dbl>, active_res_unit_count <dbl>, res_type <chr>,
## #   active_res_occupancy_count <dbl>
```

```r
count(missing_aru, fulladdress.y)
```

```
## # A tibble: 0 x 2
## # ... with 2 variables: fulladdress.y <chr>, n <int>
```

```r
missing_ap <- filter(ap, is.na(fulladdress))

missing_ap <- left_join(missing_ap, aru, by = "address_id")

anti_join(missing_ap, aru, by = "address_id")
```

```
## # A tibble: 0 x 27
## # ... with 27 variables: address_id <dbl>, status.x <chr>, type_ <chr>,
## #   entrancetype <chr>, quadrant.x <chr>, fulladdress.x <chr>,
## #   objectid_1 <dbl>, assessment_nbhd <chr>, cfsa_name <chr>,
## #   census_tract <chr>, vote_prcnct <chr>, ward <chr>, zipcode <dbl>,
## #   anc <chr>, census_block <chr>, census_blockgroup <chr>,
## #   latitude <dbl>, longitude <dbl>, active_res_unit_count <dbl>,
## #   res_type <chr>, active_res_occupancy_count <dbl>, unit_id <dbl>,
## #   fulladdress.y <chr>, status.y <chr>, unitnum <chr>, unittype <chr>,
## #   quadrant.y <chr>
```

```
count(missing_ap, fulladdress.y)
```

```
## # A tibble: 0 x 2
## # ... with 2 variables: fulladdress.y <chr>, n <int>
```

```
# merge using address id
```

Merge variables

Address Points has interesting variables not present in Address Residential Units. So we merge the Address Points dataset with the Address Residential Units dataset. The join works for all but 572 cases, most of which are in a new building at the Wharf.

```
aru_expanded <- aru %>%
  select(-status) %>%
  left_join(ap, by = c("fulladdress", "address_id")) %>%
  select(quadrant = quadrant.x, everything(), -quadrant.y)

anti_join(aru, ap, by = c("fulladdress", "address_id"))
```

```
## # A tibble: 572 x 7
##    unit_id address_id fulladdress         status unitnum unittype quadrant
##      <dbl>      <dbl> <chr>               <chr>  <chr>   <chr>    <chr>
##  1  223379     276680 600 WATER STREET SW ACTIVE 6-12    RENTAL   SW
##  2  223380     276680 600 WATER STREET SW ACTIVE 6-13    RENTAL   SW
##  3  223381     276680 600 WATER STREET SW ACTIVE 6-14    RENTAL   SW
##  4  223384     276680 600 WATER STREET SW ACTIVE 1-1     RENTAL   SW
##  5  223389     276680 600 WATER STREET SW ACTIVE 1-6     RENTAL   SW
##  6  223392     276680 600 WATER STREET SW ACTIVE 1-9     RENTAL   SW
##  7  223494     276680 600 WATER STREET SW ACTIVE 8-16    RENTAL   SW
##  8  223497     276680 600 WATER STREET SW ACTIVE 9-3     RENTAL   SW
##  9  223503     276680 600 WATER STREET SW ACTIVE 9-9     RENTAL   SW
## 10  223508     276680 600 WATER STREET SW ACTIVE 9-14    RENTAL   SW
## # ... with 562 more rows
```

Combination

Next, we need to drop addresses in the Address Points dataset that exist in the Address Residential Units dataset so we don't overcount addresses in multi-dwelling units.

```r
ap <- ap %>%
  filter(!address_id %in% unique(aru_expanded$address_id))
```

Finally, we can combine the two datasets to create a sampling frame that contains approximately every residential address in Washington D.C.

```r
sampling_frame <- bind_rows(ap, aru_expanded)

#summarize_all(addresses, list(~sum(is.na(.))))

write_csv(sampling_frame, "sampling_frame.csv")
```

```r
filter(aru, str_detect(fulladdress, "1930 NEW HAMPSHIRE"))
```

```
## # A tibble: 49 x 7
##    unit_id address_id fulladdress          status unitnum unittype quadrant
##      <dbl>      <dbl> <chr>                <chr>  <chr>   <chr>    <chr>
##  1  160596     226097 1930 NEW HAMPSHIRE ~ ACTIVE 1       CONDO    NW
##  2  160597     226097 1930 NEW HAMPSHIRE ~ ACTIVE 2       CONDO    NW
##  3  160598     226097 1930 NEW HAMPSHIRE ~ ACTIVE 3       CONDO    NW
##  4  160599     226097 1930 NEW HAMPSHIRE ~ ACTIVE 4       CONDO    NW
##  5  160600     226097 1930 NEW HAMPSHIRE ~ ACTIVE 5       CONDO    NW
##  6  160601     226097 1930 NEW HAMPSHIRE ~ ACTIVE 6       CONDO    NW
##  7  160602     226097 1930 NEW HAMPSHIRE ~ ACTIVE 7       CONDO    NW
##  8  160606     226097 1930 NEW HAMPSHIRE ~ ACTIVE 11      CONDO    NW
##  9  160607     226097 1930 NEW HAMPSHIRE ~ ACTIVE 12      CONDO    NW
## 10  160608     226097 1930 NEW HAMPSHIRE ~ ACTIVE 13      CONDO    NW
## # ... with 39 more rows
```

```r
filter(ap, str_detect(fulladdress, "1930 NEW HAMPSHIRE"))
```

```
## # A tibble: 0 x 21
## # ... with 21 variables: address_id <dbl>, status <chr>, type_ <chr>,
## #   entrancetype <chr>, quadrant <chr>, fulladdress <chr>,
## #   objectid_1 <dbl>, assessment_nbhd <chr>, cfsa_name <chr>,
## #   census_tract <chr>, vote_prcnct <chr>, ward <chr>, zipcode <dbl>,
## #   anc <chr>, census_block <chr>, census_blockgroup <chr>,
## #   latitude <dbl>, longitude <dbl>, active_res_unit_count <dbl>,
## #   res_type <chr>, active_res_occupancy_count <dbl>
```

Pilot survey

```
set.seed(20190714)

pilot_sample <- sampling_frame %>%
  group_by(quadrant) %>%
  sample_n(25)

write_csv(pilot_sample, "data/pilot_sample.csv")
```

Picking stratum sizes

For a desired bound V_0 on the sampling variance $V(\bar{y}_{str})$, we may find an optimal allocation using the followng algorithm.

1) Assign, for each stratum, 1 unit to be selected for the sample.
2) Fil in the following table and number these values starting from 1, inc decreasing order.

$\frac{N_1^2 S_1^2}{1 \cdot 2}$	$\frac{N_1^2 S_1^2}{2 \cdot 3}$	$\frac{N_1^2 S_1^2}{3 \cdot 4}$	$\dots$
$\frac{N_1^2 S_1^2}{1 \cdot 2}$	$\frac{N_1^2 S_1^2}{2 \cdot 3}$	$\frac{N_1^2 S_1^2}{3 \cdot 4}$	$\dots$
.	.	.	$\dots$
.	.	.	$\dots$
.	.	.	$\dots$
$\frac{N_H^2 S_H^2}{1 \cdot 2}$	$\frac{N_H^2 S_H^2}{2 \cdot 3}$	$\frac{N_H^2 S_H^2}{3 \cdot 4}$	$\dots$

3) SInce the initial allocation is $(n_{11}, n_{21}, ..., n_{H1}) = (1, 1, ..., 1)$, compute $V(\bar{y}_{str} | n_{11} = 1, n_{21} = 1, ..., n_{H1} = 1) = \frac{1}{N^2} \sum_{h=1}^{H} ((N_h^2 - N_h) S_h^2)$