dplyr/purr library in R programming

Kesav Adithya Venkidusamy

Bellevue university - Master of Science in Data Science

Course Name: DSC520-T301 Statistics for Data Science (2221-1)

Assignment: Week 5.2 Assignment

Instructor: Dr Richard Bushart

Due Date: 10/03/2021

Assignment 5.2

Using either the same dataset(s) you used in the previous weeks' exercise or a brand-new dataset of your choosing, perform the following transformations (Remember, anything you learn about the Housing dataset in these two weeks can be used for a later exercise!)

Using the dplyr package, use the 6 different operations to analyze/transform the data - GroupBy, Summarize, Mutate, Filter, Select, and Arrange – Remember this isn't just modifying data, you are learning about your data also – so play around and start to understand your dataset in more detail

```
> #Housing dataset

> library("readxl")

> # Set the working directory to the root of your DSC 520 directory

> setwd("E:/Personal/Bellevue University/Course/github/dsc520")

> #Load the `data/scores.csv` to df

> housing_df <- read_excel("data/week-7-housing.xlsx")

> attributes(housing_df)

$class

[1] "tbl_df"     "tbl"        "data.frame"


$row.names

  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15

 [16]  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30

 [31]  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45

 [46]  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60

 [61]  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75

 [76]  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90

 [91]  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105
```

```
[106] 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
[121] 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135
[136] 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150
[151] 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165
[166] 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
[181] 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195
[196] 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210
[211] 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225
[226] 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240
[241] 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255
[256] 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270
[271] 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285
[286] 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300
[301] 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315
[316] 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330
[331] 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345
[346] 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
[361] 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375
[376] 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390
[391] 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405
[406] 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420
[421] 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435
[436] 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450
[451] 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465
[466] 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480
[481] 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495
[496] 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510
```

```
[511] 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525
[526] 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540
[541] 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555
[556] 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570
[571] 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585
[586] 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600
[601] 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615
[616] 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630
[631] 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645
[646] 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660
[661] 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675
[676] 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690
[691] 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705
[706] 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720
[721] 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735
[736] 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750
[751] 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765
[766] 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780
[781] 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795
[796] 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810
[811] 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825
[826] 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840
[841] 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855
[856] 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870
[871] 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885
[886] 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900
[901] 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915
```

```
 [916]  916  917  918  919  920  921  922  923  924  925  926  927  928  929  930
 [931]  931  932  933  934  935  936  937  938  939  940  941  942  943  944  945
 [946]  946  947  948  949  950  951  952  953  954  955  956  957  958  959  960
 [961]  961  962  963  964  965  966  967  968  969  970  971  972  973  974  975
 [976]  976  977  978  979  980  981  982  983  984  985  986  987  988  989  990
 [991]  991  992  993  994  995  996  997  998  999 1000
 [ reached getOption("max.print") -- omitted 11865 entries ]

$names
 [1] "Sale Date"                "Sale Price"
 [3] "sale_reason"              "sale_instrument"
 [5] "sale_warning"             "sitetype"
 [7] "addr_full"                "zip5"
 [9] "ctyname"                  "postalctyn"
[11] "lon"                      "lat"
[13] "building_grade"           "square_feet_total_living"
[15] "bedrooms"                 "bath_full_count"
[17] "bath_half_count"          "bath_3qtr_count"
[19] "year_built"               "year_renovated"
[21] "current_zoning"           "sq_ft_lot"
[23] "prop_type"                "present_use"

> str(housing_df)
tibble [12,865 x 24] (S3: tbl_df/tbl/data.frame)
 $ Sale Date               : POSIXct[1:12865], format: "2006-01-03" "2006-01-03" ...
 $ Sale Price              : num [1:12865] 698000 649990 572500 420000 369900 ...
 $ sale_reason             : num [1:12865] 1 1 1 1 1 1 1 1 1 1 ...
 $ sale_instrument         : num [1:12865] 3 3 3 3 3 15 3 3 3 3 ...
 $ sale_warning            : chr [1:12865] NA NA NA NA ...
 $ sitetype                : chr [1:12865] "R1" "R1" "R1" "R1" ...
 $ addr_full               : chr [1:12865] "17021 NE 113TH CT" "11927 178TH PL NE" "13315
174TH AVE NE" "3303 178TH AVE NE" ...
 $ zip5                    : num [1:12865] 98052 98052 98052 98052 98052 ...
 $ ctyname                 : chr [1:12865] "REDMOND" "REDMOND" NA "REDMOND" ...
 $ postalctyn              : chr [1:12865] "REDMOND" "REDMOND" "REDMOND"
"REDMOND" ...
 $ lon                     : num [1:12865] -122 -122 -122 -122 -122 ...
 $ lat                     : num [1:12865] 47.7 47.7 47.7 47.6 47.7 ...
 $ building_grade          : num [1:12865] 9 9 8 8 7 7 10 10 9 8 ...
 $ square_feet_total_living: num [1:12865] 2810 2880 2770 1620 1440 4160 3960 3720 4160
2760 ...
 $ bedrooms                : num [1:12865] 4 4 4 3 3 4 5 4 4 4 ...
 $ bath_full_count         : num [1:12865] 2 2 1 1 1 2 3 2 2 1 ...
 $ bath_half_count         : num [1:12865] 1 0 1 0 0 1 0 1 1 0 ...
 $ bath_3qtr_count         : num [1:12865] 0 1 1 1 1 1 1 0 1 1 ...
 $ year_built              : num [1:12865] 2003 2006 1987 1968 1980 ...
 $ year_renovated          : num [1:12865] 0 0 0 0 0 0 0 0 0 0 ...
 $ current_zoning          : chr [1:12865] "R4" "R4" "R6" "R4" ...
 $ sq_ft_lot               : num [1:12865] 6635 5570 8444 9600 7526 ...
 $ prop_type               : chr [1:12865] "R" "R" "R" "R" ...
 $ present_use             : num [1:12865] 2 2 2 2 2 2 2 2 2 2 ...
> colnames(housing_df)
 [1] "Sale Date"                "Sale Price"
 [3] "sale_reason"              "sale_instrument"
 [5] "sale_warning"             "sitetype"
 [7] "addr_full"            "zip5"
 [9] "ctyname"              "postalctyn"
[11] "lon"                  "lat"
[13] "building_grade"       "square_feet_total_living"
[15] "bedrooms"             "bath_full_count"
[17] "bath_half_count"      "bath_3qtr_count"
[19] "year_built"           "year_renovated"
[21] "current_zoning"       "sq_ft_lot"
[23] "prop_type"            "present_use"
> library("dplyr")

Attaching package: ‘dplyr’


The following objects are masked from ‘package:stats’:


    filter, lag


The following objects are masked from ‘package:base’:


    intersect, setdiff, setequal, union


> #Number of rows and columns using dim function
> housing_df %>% head(4) %>% dim
[1]  4 24
> ##Count of records
> nrow(housing_df)
[1] 12865
```

```
> ncol(housing_df)
[1] 24
> ## Select zip, bedrooms and sales price from dataset
> housing_df %>% select(zip5, bedrooms, "Sale Price")
# A tibble: 12,865 x 3
    zip5 bedrooms `Sale Price`
   <dbl>    <dbl>        <dbl>
 1 98052        4       698000
 2 98052        4       649990
 3 98052        4       572500
 4 98052        3       420000
 5 98052        3       369900
 6 98053        4       184667
 7 98053        5      1050000
 8 98053        4       875000
 9 98053        4       660000
10 98052        4       650000
# ... with 12,855 more rows
> ## Filter the dataset only for bedrooms 3,4,5
> housing_df %>% select(zip5, bedrooms, "Sale Price") %>% filter(bedrooms > 2 & bedrooms
< 6)
# A tibble: 11,055 x 3
    zip5 bedrooms `Sale Price`
   <dbl>    <dbl>        <dbl>
 1 98052        4       698000
 2 98052        4       649990
 3 98052        4       572500
 4 98052        3       420000
 5 98052        3       369900
 6 98053        4       184667
 7 98053        5      1050000
 8 98053        4       875000
 9 98053        4       660000
10 98052        4       650000
# ... with 11,045 more rows
> housing_df %>% select(zip5, bedrooms, "Sale Price") %>% filter(bedrooms %in% c(3,4,5))
# A tibble: 11,055 x 3
    zip5 bedrooms `Sale Price`
   <dbl>    <dbl>        <dbl>
 1 98052        4       698000
 2 98052        4       649990
 3 98052        4       572500
 4 98052        3       420000
 5 98052        3       369900
 6 98053        4       184667
 7 98053        5      1050000
 8 98053        4       875000
 9 98053        4       660000
10 98052        4       650000
# ... with 11,045 more rows
> housing_df %>% select(zip5, bedrooms, "Sale Price") %>% filter(bedrooms == 3 | bedrooms
== 4 | bedrooms == 5)
# A tibble: 11,055 x 3
    zip5 bedrooms `Sale Price`
   <dbl>    <dbl>       <dbl>
 1 98052        4      698000
 2 98052        4      649990
 3 98052        4      572500
 4 98052        3      420000
 5 98052        3      369900
 6 98053        4      184667
 7 98053        5     1050000
 8 98053        4      875000
 9 98053        4      660000
10 98052        4      650000
# ... with 11,045 more rows
> ## Apply Slice to get sample rows
> housing_df %>% select(zip5, bedrooms, "Sale Price") %>% filter(bedrooms > 2 & bedrooms
< 6) %>% slice(1:5)
# A tibble: 5 x 3
   zip5 bedrooms `Sale Price`
  <dbl>    <dbl>       <dbl>
1 98052        4      698000
2 98052        4      649990
3 98052        4      572500
4 98052        3      420000
5 98052        3      369900
> ## Calculate price per sq feet using mutate function
> housing_df %>% select(zip5, bedrooms, square_feet_total_living, "Sale Price")
# A tibble: 12,865 x 4
    zip5 bedrooms square_feet_total_living `Sale Price`
   <dbl>    <dbl>                  <dbl>      <dbl>
 1 98052        4                   2810     698000
 2 98052        4                   2880     649990
 3 98052        4                   2770     572500
 4 98052        3                   1620     420000
 5 98052        3                   1440     369900
 6 98053        4                   4160     184667
 7 98053        5                   3960    1050000
 8 98053        4                   3720     875000
 9 98053        4                   4160     660000
10 98052        4                   2760     650000
# ... with 12,855 more rows
> housing_df %>% mutate(price_per_sq_ft = as.double(round(`Sale
Price`/square_feet_total_living,2))) %>% select(zip5, bedrooms, square_feet_total_living, "Sale
Price", price_per_sq_ft)
# A tibble: 12,865 x 5
    zip5 bedrooms square_feet_total_living `Sale Price` price_per_sq_ft
   <dbl>    <dbl>                  <dbl>      <dbl>           <dbl>
 1 98052        4                   2810     698000            248.
 2 98052        4                   2880     649990            226.
 3 98052        4                   2770     572500            207.
 4 98052        3                   1620     420000            259.
 5 98052        3                   1440     369900            257.
 6 98053        4                   4160     184667             44.4
 7 98053        5                   3960    1050000            265.
 8 98053        4                   3720     875000            235.
 9 98053        4                   4160     660000            159.
10 98052        4                   2760     650000            236.
```

```
# ... with 12,855 more rows

> ##Calculate the age of house

> housing_df %>% mutate(no_of_year = as.integer(format(Sys.Date(), "%Y")) - year_built)
%>% select(no_of_year)

# A tibble: 12,865 x 1

   no_of_year

        <dbl>

 1         18

 2         15

 3         34

 4         53

 5         41

 6         16

 7         28

 8         33

 9         43

10         45

# ... with 12,855 more rows

> #Mean, max and min price of house by zip code and bedrooms for 3,4 and 5 bedrooms

> housing_df %>% filter(bedrooms %in% c(3,4,5)) %>% group_by(zip5, bedrooms) %>%
summarize(AvgPrice=mean(`Sale Price`), MaxPrice = max(`Sale Price`), MinPrice = min(`Sale
Price`))

`summarise()` has grouped output by 'zip5'. You can override using the `.groups` argument.

# A tibble: 10 x 5

# Groups:   zip5 [4]

    zip5 bedrooms AvgPrice MaxPrice MinPrice

   <dbl>    <dbl>    <dbl>    <dbl>    <dbl>

 1 98052        3  528491.  4400000     2031
 2 98052        4  714620.  4380542     4059
 3 98052        5  826658.  4380542     2500
 4 98053        3  616553.  3850000      698
 5 98053        4  770975.  3462000      698
 6 98053        5  851388.  3175000    14000
 7 98059        4  645000    645000   645000
 8 98074        3  701527.  1300000   520000
 9 98074        4  931692.  1895000   475000
10 98074        5 1311727.  2160200   773300
> ##Arrange the above result by bedrooms and AvgPrice
> housing_df %>% filter(bedrooms %in% c(3,4,5)) %>% group_by(zip5, bedrooms) %>% summarize(AvgPrice=mean(`Sale Price`), MaxPrice = max(`Sale Price`), MinPrice = min(`Sale Price`)) %>% arrange(bedrooms, zip5, desc(AvgPrice))
`summarise()` has grouped output by 'zip5'. You can override using the `.groups` argument.
# A tibble: 10 x 5
# Groups:   zip5 [4]
    zip5 bedrooms AvgPrice MaxPrice MinPrice
   <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
 1 98052        3  528491.  4400000     2031
 2 98053        3  616553.  3850000      698
 3 98074        3  701527.  1300000   520000
 4 98052        4  714620.  4380542     4059
 5 98053        4  770975.  3462000      698
 6 98059        4  645000    645000   645000
 7 98074        4  931692.  1895000   475000
 8 98052        5  826658.  4380542     2500
 9 98053        5  851388.  3175000    14000
10 98074        5 1311727.  2160200   773300
```

b. Using the purrr package – perform 2 functions on your dataset. You could use zip_n, keep, discard, compact, etc.

```
> #Purr package and functions
> library(purrr)
> #Calculate mean price
> housing_df %>% select(`Sale Price`) %>% map_dbl(mean)
Sale Price
  660737.7
> housing_df %>% map(class)
$`Sale Date`
[1] "POSIXct" "POSIXt"


$`Sale Price`
[1] "numeric"


$sale_reason
[1] "numeric"


$sale_instrument
[1] "numeric"


$sale_warning
[1] "character"


$sitetype
[1] "character"
```

```
$addr_full
[1] "character"

$zip5
[1] "numeric"

$ctyname
[1] "character"

$postalctyn
[1] "character"

$lon
[1] "numeric"

$lat
[1] "numeric"

$building_grade
[1] "numeric"

$square_feet_total_living
[1] "numeric"

$bedrooms
[1] "numeric"
```

```
$bath_full_count
[1] "numeric"

$bath_half_count
[1] "numeric"

$bath_3qtr_count
[1] "numeric"

$year_built
[1] "numeric"

$year_renovated
[1] "numeric"

$current_zoning
[1] "character"

$sq_ft_lot
[1] "numeric"

$prop_type
[1] "character"

$present_use
[1] "numeric"
```

```
> map(housing_df, mean)
$`Sale Date`
[1] "2011-07-28 15:07:32 UTC"

$`Sale Price`
[1] 660737.7

$sale_reason
[1] 1.550019

$sale_instrument
[1] 3.67773

$sale_warning
[1] NA

$sitetype
[1] NA

$addr_full
[1] NA

$zip5
[1] 98052.54

$ctyname
[1] NA
```

```
$postalctyn
[1] NA

$lon
[1] -122.0792

$lat
[1] 47.68358

$building_grade
[1] 8.24042

$square_feet_total_living
[1] 2539.506

$bedrooms
[1] 3.478663

$bath_full_count
[1] 1.798445

$bath_half_count
[1] 0.6133696

$bath_3qtr_count
[1] 0.4939759
```

```
$year_built
[1] 1993.003

$year_renovated
[1] 26.24431

$current_zoning
[1] NA

$sq_ft_lot
[1] 22228.57

$prop_type
[1] NA

$present_use
[1] 6.597746

Warning messages:
1: In mean.default(.x[[i]], ...) :
  argument is not numeric or logical: returning NA
2: In mean.default(.x[[i]], ...) :
  argument is not numeric or logical: returning NA
3: In mean.default(.x[[i]], ...) :
  argument is not numeric or logical: returning NA
4: In mean.default(.x[[i]], ...) :
  argument is not numeric or logical: returning NA
5: In mean.default(.x[[i]], ...) :
  argument is not numeric or logical: returning NA
6: In mean.default(.x[[i]], ...) :
  argument is not numeric or logical: returning NA
7: In mean.default(.x[[i]], ...) :
  argument is not numeric or logical: returning NA
> #Houses over 1 million using keep function
> high_cost <- housing_df$`Sale Price` |> keep(~ (.x) > 1000000)
> high_cost[1:10]
 [1] 1050000 1392000 1445000 1053649 1900000 1080135 1075000 1520000 1390000 1390000
> length(high_cost)
[1] 934
> #Low cost houses under 350k using discard function
> low_cost <- housing_df$`Sale Price` |> discard(~ (.x) > 350000)
> low_cost[1:10]
 [1] 184667 165000 265000 335105 270000 350000 345000 148000 275000 229000
> length(low_cost)
[1] 1168
```

c. Use the cbind and rbind function on your dataset

```
> #House filters based on number of bedrooms
> house_cols <- housing_df %>% select(`Sale Date`,`Sale Price`, zip5, ctyname, postalctyn, square_feet_total_living, bedrooms, sq_ft_lot)
> nrow(house_cols)
[1] 12865
> house_0 <- house_cols %>% filter(bedrooms == 0)
> house_1 <- house_cols %>% filter(bedrooms == 1)

> house_2 <- house_cols %>% filter(bedrooms == 2)

> house_3 <- house_cols %>% filter(bedrooms == 3)

> house_big <- house_cols %>% filter(bedrooms > 3)

> #Combine all the data frames back to  single data frame using rbind

> house_combine <- rbind(house_0, house_1, house_2, house_3, house_big)

> nrow(house_combine)

[1] 12865

> #Combine all the columns back to single data frame using cbind

> house_cols <- housing_df %>% select(`Sale Date`,`Sale Price`, zip5, ctyname, postalctyn,
square_feet_total_living, bedrooms, sq_ft_lot)

> house_cols_exclude <- housing_df %>% select(-c(`Sale Date`,`Sale Price`, zip5, ctyname,
postalctyn, square_feet_total_living, bedrooms, sq_ft_lot))

> colnames(house_cols)

[1] "Sale Date"                "Sale Price"              "zip5"

[4] "ctyname"                  "postalctyn"              "square_feet_total_living"

[7] "bedrooms"                 "sq_ft_lot"

> colnames(house_cols_exclude)

 [1] "sale_reason"     "sale_instrument" "sale_warning"    "sitetype"

 [5] "addr_full"       "lon"             "lat"             "building_grade"

 [9] "bath_full_count" "bath_half_count" "bath_3qtr_count" "year_built"

[13] "year_renovated"  "current_zoning"  "prop_type"       "present_use"

> house_all_cols <- cbind(house_cols_exclude, house_cols)

> colnames(house_all_cols)

 [1] "sale_reason"              "sale_instrument"

 [3] "sale_warning"             "sitetype"

 [5] "addr_full"                "lon"

 [7] "lat"                      "building_grade"
 [9] "bath_full_count"          "bath_half_count"
[11] "bath_3qtr_count"          "year_built"
[13] "year_renovated"           "current_zoning"
[15] "prop_type"                "present_use"
[17] "Sale Date"                "Sale Price"
[19] "zip5"                     "ctyname"
[21] "postalctyn"               "square_feet_total_living"
[23] "bedrooms"                 "sq_ft_lot"
> #Validating the number of columns
> ncol(house_cols_exclude)
[1] 16
> ncol(house_cols)
[1] 8
> ncol(house_all_cols)
[1] 24
```

d. Split a string, then concatenate the results back together

```
> require(stringr)
> #Get month and year from Sale date
> class(housing_df$`Sale Date`)
[1] "POSIXct" "POSIXt"
> year <- housing_df$`Sale Date` %>% str_sub(start=1, end=4)
> year[1:10]
 [1] "2006" "2006" "2006" "2006" "2006" "2006" "2006" "2006" "2006" "2006"
> length(year)
[1] 12865
> month <-  housing_df$`Sale Date` %>% str_sub(start=6, end=7)
> month[1:10]
 [1] "01" "01" "01" "01" "01" "01" "01" "01" "01" "01"
> length(month)
[1] 12865
> housing_df$sales_month <- paste(month, year, sep='-')
> head(housing_df$sales_month)
[1] "01-2006" "01-2006" "01-2006" "01-2006" "01-2006" "01-2006"
> length(housing_df$sales_month)
[1] 12865
```