PSTAT 100 Lab 5

Lab 5: Principle Components

Principal components analysis (PCA) is a widely-used multivariate analysis technique. Depending on the application, PCA is variously described as:

- a dimension reduction method
- a an approximation method
- a latent factor model
- a filtering or compression method

The core technique of PCA is *finding linear data transformations that preserve variance.*

What does it mean to say that '*principal components are linear data transformations*'? Suppose you have a dataset with n observations and p variables. We can represent the values as a data matrix $\mathbf{X}$ with n rows and p columns:

$$\mathbf{X} = \underbrace{\begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_p \end{bmatrix}}_{\text{column vectors}} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix}$$

To say that the principal components are linear data transformations means that each principal component is of the form:

$$\text{PC} = \mathbf{X}\mathbf{v} = v_1\mathbf{x}_1 + v_2\mathbf{x}_2 + \cdots + v_p\mathbf{x}_p$$

for some vector $\mathbf{v}$. In PCA, the following terminology is used:

- linear combination coefficients v_j are known as *loadings*
- values of the linear combinations are known as *scores*
- the vector of loadings $\mathbf{v}$ is known as a *principal axis*

As discussed in lecture, the values of the loadings are found by decomposing the correlation structure.

```
# Load libraries
library(dplyr)
library(ggplot2)
```

Objectives

In this lab, you'll focus on computing and interpreting principal components:

- finding the loadings (linear combination coefficients) for each PC;
- quantifying the variation captured by each PC;
- visualization-based techniques for selecting a number of PC's to analyze;
- plotting and interpreting loadings.

You'll work with a selection of county summaries from the 2010 U.S. census. The first few rows of the dataset are shown below:

```
# Import tidy county-level 2010 census data
census <- read.csv('data/census2010.csv', fileEncoding = 'latin1')
head(census)
```

```
    State  County    Women    White  Citizen IncomePerCap  Poverty ChildPoverty
1 Alabama Autauga 51.56734 75.78823 73.74912     24974.50 12.91231     18.70758
2 Alabama Baldwin 51.15134 83.10262 75.69406     27316.84 13.42423     19.48431
3 Alabama Barbour 46.17184 46.23159 76.91222     16824.22 26.50563     43.55962
4 Alabama    Bibb 46.58910 74.49989 77.39781     18430.99 16.60375     27.19708
5 Alabama  Blount 50.59435 87.85385 73.37550     20532.27 16.72152     26.85738
6 Alabama Bullock 46.99382 22.19918 75.45420     17579.57 24.50260     37.29116
  Professional  Service   Office Production    Drive   Carpool    Transit
1     32.79097 17.17044 24.28243   17.15713 87.50624  8.781235 0.09525905
2     32.72994 17.95092 27.10439   11.32186 84.59861  8.959078 0.12662092
3     26.12404 16.46343 23.27878   23.31741 83.33021 11.056609 0.49540324
4     21.59010 17.95545 17.46731   23.74415 83.43488 13.153641 0.50313661
5     28.52930 13.94252 23.83692   20.10413 84.85031 11.279222 0.36263213
6     19.55253 14.92420 20.17051   25.73547 74.77277 14.839127 0.77321596
  OtherTransp WorkAtHome MeanCommute Employed PrivateWork SelfEmployed
1   1.3059687  1.8356531    26.50016 43.43637    73.73649     5.433254
2   1.4438000  3.8504774    26.32218 44.05113    81.28266     5.909353
3   1.6217251  1.5019456    24.51828 31.92113    71.59426     7.149837
4   1.5620952  0.7314679    28.71439 36.69262    76.74385     6.637936
```

```
5   0.4199411  2.2654133    34.84489 38.44914    81.82671     4.228716
6   1.8238247  3.0998783    28.63106 36.19592    79.09065     5.273684
  FamilyWork Unemployment Minority
1 0.00000000     7.733726 22.53687
2 0.36332686     7.589820 15.21426
3 0.08977425    17.525557 51.94382
4 0.39415148     8.163104 24.16597
5 0.35649281     7.699640 10.59474
6 0.00000000    17.890026 76.53587
```

The observational units are U.S. counties, and each row is an observation on one county. The values are, for the most part, percentages of the county population. You can find variable descriptions in the metadata file `census2010metadata.csv` in the data directory.

Correlations

PCA identifies variable combinations that capture covariation by decomposing the correlation matrix. So, to start with, let's examine the correlation matrix for the 2010 county-level census data to get a sense of which variables tend to vary together.

The correlation matrix is a matrix of all pairwise correlations between variables. If x_ij denotes the value for the i^{th} observation of variable j, then the entry at row j and column k of the correlation matrix $\mathbf{R}$ is:

$$r_{jk} = \frac{\sum_i (x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k)}{S_j S_k}$$

In the census data, the `State` and `County` columns indicate the geographic region for each observation; essentially, they are a row index. So we'll drop them before computing the matrix $\mathbf{R}$:

```r
# Drop 'State' and 'County' columns
x_mx <- census[, !(names(census) %in% c('State', 'County'))]
```

From here, the matrix is simple to compute using `cor()`

```r
# Compute the correlation matrix
corr_mx <- cor(x_mx, use = "pairwise.complete.obs")
```

The matrix can be inspected directly to determine which variables vary together. For example, we could look at the correlations between employment rate and every other variable in the dataset by extracting the `Employed` column from the correlation matrix and sorting the correlations:

```
# Correlation between employment rate and other variables, sorted
sorted_corr <- sort(corr_mx[, 'Employed'])
sorted_corr
```

```
ChildPoverty      Poverty Unemployment     Minority      Service  MeanCommute
 -0.74451012  -0.73556942  -0.69798490  -0.43905287  -0.40326105  -0.25211090
       Drive      Carpool   Production      Citizen       Office  OtherTransp
 -0.21503784  -0.14433612  -0.13627668  -0.08734254  -0.01483840  -0.01004103
  FamilyWork        Women      Transit SelfEmployed  PrivateWork   WorkAtHome
  0.05565414   0.13118134   0.15169999   0.15410719   0.26482563   0.30383880
       White Professional  IncomePerCap     Employed
  0.43285551   0.47341297   0.76700097   1.00000000
```

Recall that correlation is a number in the interval [-1, 1] whose magnitude indicates the strength of the linear relationship between variables:

- correlations near -1 are *strongly negative*, and mean that the variables *tend to vary in opposition*
- correlations near 1 are *strongly positive*, and mean that the variables *tend to vary together*

From examining the output above, it can be seen that the percentage of the county population that is employed is:

- strongly *negatively* correlated with child poverty, poverty, and unemployment, meaning it *tends to vary in opposition* with these variables
- strongly *positively* correlated with income per capita, meaning it *tends to vary together* with this variable

If instead we wanted to look up the correlation between just two variables, we could retrieve the relevant entry directly using `corr_mx['...','...']` with the variable names:

```
# Correlation between employment and income per capita
corr_mx['Employed', 'IncomePerCap']
```

```
[1] 0.767001
```

So across U.S. counties employment is, perhaps unsurprisingly, strongly and positively correlated with income per capita, meaning that higher employment rates tend to coincide with higher incomes per capita.

Question 1

Find the correlation between the poverty rate and demographic minority rate and store the value as `pov_dem_rate`. Interpret the value in context.

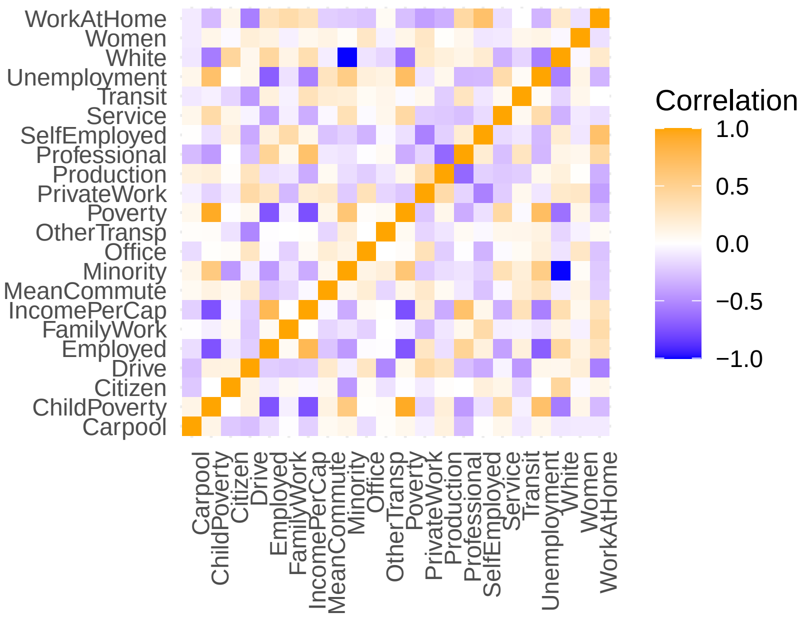
YOUR ANSWER:

(Type your answer here, replacing this text.)

```
# correlation between poverty and percent minority
# pov_dem_rate <- ...

# print
# pov_dem_rate
```

While direct inspection is useful, it can be cumbersome to check correlations for a large number of variables this way. A heatmap – a colored image of the matrix – provides a (sometimes) convenient way to see what's going on without having to examine the numerical values directly. The cell below shows one way of constructing this plot. Notice the diverging color scale; this should always be used.

```
# Melt (reshape) the correlation matrix into long format
corr_mx_long <- as.data.frame(corr_mx) %>%
  tibble::rownames_to_column(var = "row") %>%
  pivot_longer(cols = -row, names_to = "col", values_to = "Correlation")

# Construct heatmap using ggplot2
ggplot(corr_mx_long, aes(x = col, y = row, fill = Correlation)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", mid = "white", high = "orange", midpoint = 0,
                       limits = c(-1, 1)) +
  labs(x = "", y = "", fill = "Correlation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  coord_fixed() # Ensures square aspect ratio
```



Question 2

Which variable is self employment rate most *positively* correlated with? Refer to the heatmap.

YOUR ANSWER:

(*Type your answer here, replacing this text.*)

Computing principal components

Each principal component is of the form:

$$\text{PC}_i = \sum_j v_j x_{ij} \quad \text{(PC score for observation } i\text{)}$$

The loading v_j for each component indicate which variables are most influential (heavily weighted) on that principal axis, and thus offer an indirect picture of which variables are driving variation and covariation in the original data.

Loadings and scores

In **R**, the `prcomp()` function from the **stats** package provides an easy-to-use implementation for Principal Component Analysis (PCA). Alternatively, the **FactoMineR** or **psych** packages offer more detailed PCA methods.

```
# Perform PCA with standardization
pca <- prcomp(x_mx, center = TRUE, scale. = TRUE)
```

In **R**, most quantities you might need for PCA can be accessed as attributes of the `pca` object. Specifically:

- `pca$rotation` contains the **loadings** (principal component coefficients).
- `pca$x` contains the **scores** (principal component projections of the data).
- `pca$sdev^2` contains the **eigenvalues** (variances along each principal axis, see lecture notes).

Examine the loadings below. Each column represents the loadings for one principal component, with components ordered from largest to smallest variance

```
# Inspect loadings
pca$rotation
```

```
                     PC1         PC2         PC3         PC4          PC5
Women        -0.02005470  0.13995796  0.18760029 -0.17661354 -0.310715946
White         0.28961352  0.19654947 -0.28890170 -0.07805879  0.242440854
Citizen       0.05069832  0.06499401 -0.28190402 -0.46798560  0.404243822
IncomePerCap  0.33486321  0.02043167  0.28407388 -0.02219716  0.040679873
Poverty      -0.36521186 -0.12017151 -0.04016959 -0.12823071 -0.076355051
ChildPoverty -0.36483606 -0.08108640 -0.07743320 -0.09858529 -0.074419968
Professional  0.24013945 -0.17561108  0.28763560 -0.25878851 -0.094541294
Service      -0.20325420 -0.13971356  0.00595703 -0.12214487  0.377802407
Office       -0.05216762  0.18980300  0.28139826 -0.26719484 -0.006059406
Production   -0.09430653  0.28232940 -0.28550004  0.35510641 -0.051805149
Drive        -0.10219740  0.40613046 -0.09922901 -0.26107726 -0.288984427
Carpool      -0.07912940 -0.06374361 -0.09569624  0.45796222  0.110105055
Transit       0.03023258 -0.10114242  0.39086936  0.05224460  0.281640988
OtherTransp  -0.02187119 -0.20940286  0.13931485  0.22109776  0.318696859
WorkAtHome    0.21835338 -0.33163575 -0.11606828 -0.11316556 -0.067242014
MeanCommute  -0.09700319  0.17673925  0.13532223 -0.14440820  0.232196198
Employed      0.34558790  0.05465314  0.15772553  0.12870934 -0.115728882
PrivateWork   0.03553872  0.44192238  0.15870947  0.14672543  0.033461203
```

```
SelfEmployed  0.15530037 -0.31617380 -0.26679800 -0.10445280 -0.200414827
FamilyWork    0.08507737 -0.22113731 -0.20330137 -0.06481704 -0.213610661
Unemployment -0.33342029 -0.04304656  0.06993779 -0.12523474  0.125138110
Minority     -0.29246071 -0.19162849  0.28223050  0.07490075 -0.250837543
                      PC6         PC7          PC8         PC9         PC10
Women        -0.274825757 -0.53755055  0.235285611 -0.40839901 -0.229221367
White        -0.061416460 -0.11717366  0.053328609 -0.05465901 -0.022883565
Citizen      -0.025596995 -0.15592068 -0.014819254 -0.02798139 -0.098203407
IncomePerCap -0.030926403  0.06845242 -0.016533393 -0.01623905 -0.028274977
Poverty      -0.073253156 -0.13691843 -0.090574367 -0.06095347 -0.094903738
ChildPoverty -0.101474889 -0.12828189 -0.087316544 -0.06750947 -0.098937013
Professional -0.004750115  0.08267846 -0.071357991  0.08710249 -0.223003108
Service       0.257542861  0.15772282  0.090143534 -0.58598089  0.153836113
Office        0.155655223 -0.15072310  0.492650630  0.22396098  0.270637341
Production   -0.168182382 -0.23138607 -0.299673205  0.06073182 -0.021054952
Drive         0.168809154  0.21690962 -0.042703094  0.02560902  0.011982227
Carpool      -0.235709523  0.18282648  0.641950030 -0.14521142 -0.136890653
Transit      -0.377725118 -0.01947904 -0.374445157 -0.24825279  0.163688603
OtherTransp   0.279204081 -0.54976126  0.034250736  0.37232903 -0.034314534
WorkAtHome   -0.168719788 -0.03164712  0.055562778  0.07140385 -0.055435758
MeanCommute  -0.590952917  0.24992589  0.093081343  0.34686237  0.003542312
Employed      0.060929750 -0.10215176 -0.008287909 -0.19044829 -0.015788694
PrivateWork  -0.060590098 -0.12730437 -0.037899522 -0.10135594  0.151123347
SelfEmployed -0.181008157 -0.02623184  0.068297202  0.01318501 -0.177111984
FamilyWork   -0.203810877 -0.14968988  0.048505043 -0.01109510  0.815791472
Unemployment -0.132275718 -0.12313061  0.026469385  0.13708990 -0.013801428
Minority      0.054749876  0.11698843 -0.058901686  0.05521860  0.022567761
                     PC11        PC12         PC13         PC14         PC15
Women        -0.21270905 -0.17006589  0.157758146 -0.042220596 -0.113942650
White        -0.04195689  0.12030555 -0.027053519 -0.393618253 -0.035013737
Citizen       0.01900749  0.18601449  0.021192595  0.619243271  0.116601178
IncomePerCap -0.13346351 -0.05143849 -0.050939997  0.266482439 -0.070709854
Poverty       0.05092970  0.20020103 -0.131663650 -0.095262803  0.319450230
ChildPoverty  0.07517783  0.14593872 -0.105866642 -0.098661626  0.393329173
Professional -0.32233310  0.33615991 -0.204391865 -0.056315952  0.102993796
Service      -0.05987629 -0.40202423 -0.026179803 -0.029447165  0.005201637
Office        0.52794063  0.13748397  0.050947888 -0.013481096 -0.052530709
Production    0.12044142 -0.05796958 -0.029882645  0.264431809 -0.163273217
Drive        -0.16662155 -0.08367892  0.158781651 -0.048910720 -0.029596127
Carpool      -0.10601844  0.29873231 -0.037121348  0.141038422  0.054541069
Transit       0.28714186  0.25313260  0.243861977 -0.124948907 -0.119416204
```

```
OtherTransp  -0.22307706 -0.19854840  0.181742247 -0.068611355  0.168988266
WorkAtHome    0.26228170 -0.35146464 -0.556380972 -0.076534114  0.002744154
MeanCommute  -0.16333093 -0.40526423  0.147735244 -0.020224527  0.240347484
Employed      0.06281395 -0.08090196 -0.055604904  0.285760884  0.169774676
PrivateWork   0.11711388 -0.10097894 -0.450447713 -0.006197744  0.291232279
SelfEmployed  0.34574275 -0.17794571  0.329360369  0.016309837 -0.052208307
FamilyWork   -0.31026375  0.09039875  0.006316347  0.080816018  0.069118506
Unemployment -0.15264181  0.02982072 -0.362180139  0.046871754 -0.665369650
Minority      0.04915624 -0.12694106  0.026108851  0.395336656  0.045659759
                     PC16         PC17         PC18         PC19         PC20
Women         0.190960035  0.148377223  0.005946459  0.050017263 -0.04726318
White        -0.049062054 -0.148104966  0.053616941  0.007430223 -0.02016588
Citizen       0.067242647  0.179710069 -0.100342758  0.082991447  0.09668911
IncomePerCap -0.294452459 -0.047195002  0.713774870  0.165983752 -0.18518978
Poverty      -0.047636631 -0.227490780 -0.018015824 -0.011492236 -0.27642535
ChildPoverty -0.163165499 -0.202834657  0.405313030  0.099722252  0.24952591
Professional -0.045454382  0.053740716 -0.107539966 -0.608185236 -0.05195490
Service      -0.064617046 -0.082027187  0.069712735 -0.353849395 -0.07991925
Office        0.079231650 -0.117818643  0.127661260 -0.229184876 -0.06807285
Production    0.142329941 -0.094739103  0.232051063 -0.561800729 -0.10567920
Drive        -0.165461729  0.040402488  0.103963910 -0.156394480  0.60335560
Carpool      -0.065394875  0.071457299  0.044920498 -0.106102833  0.23507969
Transit       0.008535726  0.104085054 -0.005321210 -0.045498995  0.34158076
OtherTransp  -0.132962328  0.067492993  0.025104833 -0.098007122  0.24870686
WorkAtHome    0.289911470  0.127683887  0.140831120 -0.037032742  0.33654334
MeanCommute   0.064461875 -0.182458878 -0.072583270 -0.061522334 -0.08445479
Employed     -0.031260858 -0.706297890 -0.305263235  0.041238731  0.23068060
PrivateWork  -0.439396528  0.371222932 -0.231748936  0.013571569 -0.06156636
SelfEmployed -0.608598485  0.057153704 -0.122195163 -0.161609416 -0.07963655
FamilyWork   -0.052283096 -0.002424154  0.012158950 -0.045391321  0.00532158
Unemployment -0.319148076 -0.244551551 -0.142296513  0.084087798  0.08885622
Minority      0.039957574  0.157149134 -0.074605279  0.001124201  0.02561754
                     PC21          PC22
Women        -0.0287139068  0.0002575236
White         0.0299975160 -0.7096441869
Citizen       0.0093718452  0.0043633017
IncomePerCap  0.1955416290 -0.0085011873
Poverty       0.6936146391  0.0114286599
ChildPoverty -0.5323898315 -0.0072821878
Professional -0.1482155984 -0.0070492083
Service      -0.0303280095 -0.0063509981
```

```
Office       -0.0277657760 -0.0061606803
Production   -0.0103255426 -0.0048554238
Drive         0.3054413854  0.0083638940
Carpool       0.1229043696 -0.0018836635
Transit       0.1295565869  0.0016973982
OtherTransp   0.1217941849 -0.0010907891
WorkAtHome    0.1680371570  0.0038147964
MeanCommute  -0.0321942056  0.0027393009
Employed     -0.0019983929  0.0025251045
PrivateWork  -0.0283245215  0.0122170296
SelfEmployed -0.0142844158  0.0085174421
FamilyWork   -0.0004372989  0.0005252417
Unemployment -0.0352507958 -0.0023502609
Minority     -0.0100576853 -0.7040205566
```

Similarly, inspect the scores below and check your understanding; each row is an observation and the columns give the scores on each principal axis.

```r
# Compute variance of PCA scores
apply(pca$x, 2, var)
```

```
        PC1         PC2         PC3         PC4         PC5         PC6 
5.782838803 3.334635888 2.510826804 1.686634079 1.195581036 1.133893501 
        PC7         PC8         PC9        PC10        PC11        PC12 
1.040986545 0.884572938 0.807115690 0.740102943 0.579171402 0.484406271 
       PC13        PC14        PC15        PC16        PC17        PC18 
0.387114639 0.370498343 0.314212825 0.225260086 0.172343653 0.138746458 
       PC19        PC20        PC21        PC22 
0.103811535 0.058562962 0.046216149 0.002467451 
```

Importantly, `statsmodels` rescales the scores so that they have unit inner product; in other words, so that the variances are all $\frac{1}{n-1}$.

```r
# Variance of PCA scores
scores_var <- apply(pca$x, 2, var)
scores_var
```

```
        PC1         PC2         PC3         PC4         PC5         PC6 
5.782838803 3.334635888 2.510826804 1.686634079 1.195581036 1.133893501 
        PC7         PC8         PC9        PC10        PC11        PC12 
```

```
1.040986545 0.884572938 0.807115690 0.740102943 0.579171402 0.484406271
       PC13        PC14        PC15        PC16        PC17        PC18
0.387114639 0.370498343 0.314212825 0.225260086 0.172343653 0.138746458
       PC19        PC20        PC21        PC22
0.103811535 0.058562962 0.046216149 0.002467451
```

```r
# For comparison
1 / (nrow(x_mx) - 1)
```

```
[1] 0.0003108486
```

In **R**, to change this behavior and disable normalization (i.e., prevent standardization of variables), set `scale. = FALSE` when computing the principal components using `prcomp()`

Question 3

Check your understanding. Which variable contributes most to the sixth principal component? Store the variable name exactly as it appears among the original column names as `pc6_most_influential_variable`, and store the corresponding loading as `pc6_most_influential_variable_loading`. Print the variable name.

YOUR ANSWER:

```r
# find most influential variable
# pc6_most_influential_variable <- ...

# find loading
# pc6_most_influential_variable_loading <- ...

# print
```

Variance ratios

The *variance ratios* indicate the proportions of total variance in the data captured by each principal axis. You may recall from lecture that the variance ratios are computed from the eigenvalues of the correlation (or covariance, if data are not standardized) matrix.

When using `statsmodels`, these need to be computed manually.

```r
# Compute variance ratios
var_ratios <- (pca$sdev^2) / sum(pca$sdev^2)

# Print variance ratios
print(var_ratios)
```

```
 [1] 0.2628563092 0.1515743585 0.1141284911 0.0766651854 0.0543445926
 [6] 0.0515406137 0.0473175702 0.0402078608 0.0366870768 0.0336410429
[11] 0.0263259728 0.0220184669 0.0175961200 0.0168408338 0.0142824011
[16] 0.0102390948 0.0078338024 0.0063066572 0.0047187061 0.0026619528
[21] 0.0021007340 0.0001121568
```

Note again that the principal components have been computed in order of *decreasing* variance.

Question 4

Check your understanding. What proportion of variance is captured *jointly* by the first three components taken together? Provide a calculation to justify your answer.

YOUR ANSWER:

(*Type your answer here, replacing this text.)*

Selecting a subset of PCs

PCA generally consists of choosing a small subset of components. The basic strategy for selecting this subset is to determine how many are needed to capture some analyst-chosen minimum portion of total variance in the original data.

Most often this assessment is made graphically by inspecting the variance ratios and their cumulative sum, *i.e.*, the amount of total variation captured jointly by subsets of successive components. We'll store these quantities in a data frame.

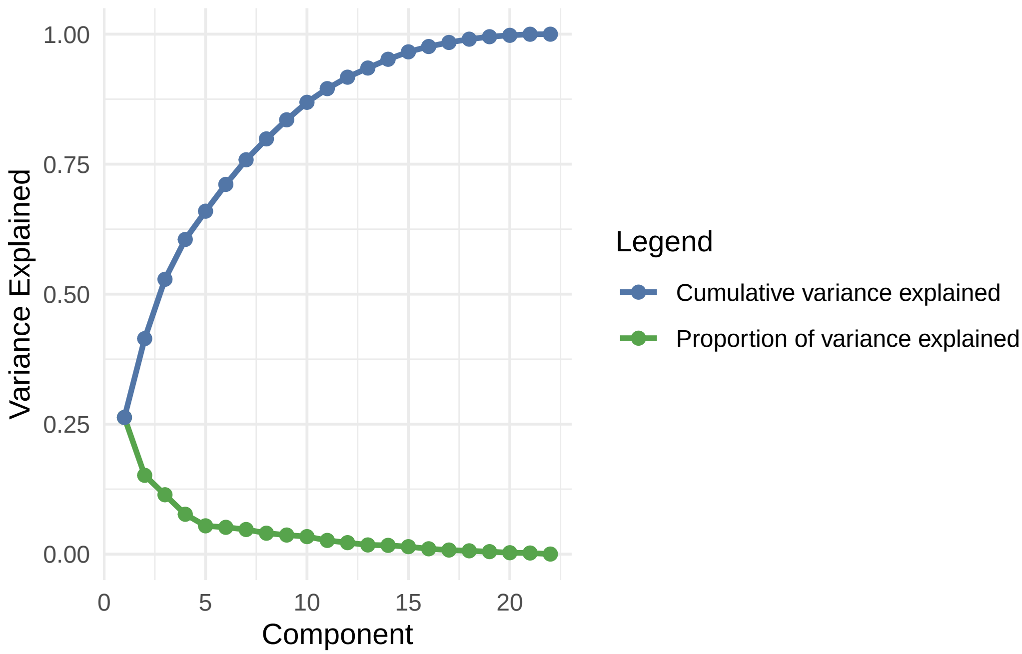
```r
# Load necessary library
library(dplyr)

# Create the data frame first
pca_var_explained <- data.frame(
  Component = seq(1, length(var_ratios)),
  Proportion_of_variance_explained = var_ratios
)

# Add cumulative variance explained
pca_var_explained <- pca_var_explained %>%
  mutate(Cumulative_variance_explained = cumsum(Proportion_of_variance_explained))

# Print first few rows
head(pca_var_explained)
```

```
  Component Proportion_of_variance_explained Cumulative_variance_explained
1         1                       0.26285631                     0.2628563
2         2                       0.15157436                     0.4144307
3         3                       0.11412849                     0.5285592
4         4                       0.07666519                     0.6052243
5         5                       0.05434459                     0.6595689
6         6                       0.05154061                     0.7111096
```

Now we'll make a dual-axis plot showing, on one side, the proportion of variance explained (y) as a function of component (x), and on the other side, the cumulative variance explained (y) also as a function of component (x). Make sure that you've completed Q1(a) before running the next cell.

```r
# Create the base plot
var_explained_plot <- ggplot(pca_var_explained, aes(x = Component)) +

  # Proportion of variance explained (green line & points)
  geom_line(aes(y = Proportion_of_variance_explained, color = "Proportion of variance explai
  geom_point(aes(y = Proportion_of_variance_explained, color = "Proportion of variance expla

  # Cumulative variance explained (blue line & points)
  geom_line(aes(y = Cumulative_variance_explained, color = "Cumulative variance explained"),
  geom_point(aes(y = Cumulative_variance_explained, color = "Cumulative variance explained")

  # Custom colors for lines
  scale_color_manual(values = c("Proportion of variance explained" = "#57A44C",
```

```
                                     "Cumulative variance explained" = "#5276A7")) +

  # Axis labels and theme adjustments
  labs(x = "Component", y = "Variance Explained", color = "Legend") +
  theme_minimal()

# Display the plot
print(var_explained_plot)
```



The purpose of making this plot is to quickly determine the fewest number of principal components that capture a considerable portion of variation and covariation. 'Considerable' here is a bit subjective.

Question 5

How many principal components explain more than 6% of total variation individually? Store this number as `num_pc`, and store the proportion of variation that they capture jointly as `var_explained`.

YOUR ANSWER:

```
# number of selected components
# num_pc <- ...

# variance explained
# var_explained <- ...

#print
# print('number selected: ', num_pc)
# print('proportion of variance captured: ', var_explained)
```

Interpreting loadings

Now that you've chosen the number of components to work with, the next step is to examine loadings to understand just *which* variables the components combine with significant weight.

We'll store the scores for the components you selected as a dataframe.

```
# Define number of principal components to keep
num_pc <- 5  # Example value, replace with actual num_pc

# Subset loadings (select first `num_pc` principal components)
loading_df <- pca$rotation[, 1:num_pc]

# Rename columns to "PC1", "PC2", ..., "PC{num_pc}"
colnames(loading_df) <- paste0("PC", seq(1, num_pc))

# Print first few rows
head(loading_df)
```

```
                     PC1         PC2         PC3         PC4         PC5
Women        -0.02005470  0.13995796  0.18760029 -0.17661354 -0.31071595
White         0.28961352  0.19654947 -0.28890170 -0.07805879  0.24244085
Citizen       0.05069832  0.06499401 -0.28190402 -0.46798560  0.40424382
IncomePerCap  0.33486321  0.02043167  0.28407388 -0.02219716  0.04067987
Poverty      -0.36521186 -0.12017151 -0.04016959 -0.12823071 -0.07635505
ChildPoverty -0.36483606 -0.08108640 -0.07743320 -0.09858529 -0.07441997
```

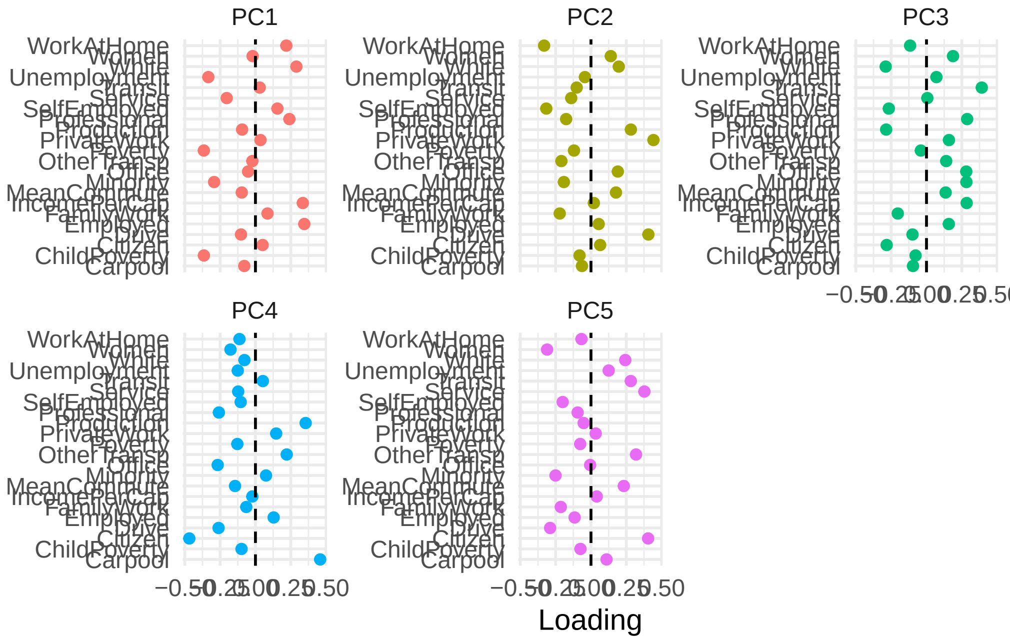
Again, the loadings are the *weights* with which the variables are combined to form the principal components. For example, the `PC1` column tells us that this component is equal to:

$$(-0.020055 \times \text{women}) + (0.289614 \times \text{white}) + (0.050698 \times \text{citizen}) + \ldots$$

Since the components together capture over half the total variation, the heavily weighted variables in the selected components are the ones that drive variation in the original data.

By visualizing the loadings, we can see which variables are most influential for each component, and thereby also which variables seem to drive total variation in the data.

```r
# Load necessary libraries
library(tidyr)
library(dplyr)
library(ggplot2)

# Melt from wide to long format
loading_plot_df <- loading_df %>%
  as.data.frame() %>%
  tibble::rownames_to_column(var = "Variable") %>%
  pivot_longer(cols = -Variable, names_to = "Principal_Component", values_to = "Loading")

# Add a column of zeros for x = 0 reference line
loading_plot_df <- loading_plot_df %>%
  mutate(zero = 0)

# Create base plot
loadings_plot <- ggplot(loading_plot_df, aes(x = Loading, y = Variable, color = Principal_Com

  # Add lines + points for loadings
  geom_line(aes(group = Variable)) +
  geom_point() +

  # Add vertical reference line at x = 0
  geom_vline(aes(xintercept = zero), color = "black", linetype = "dashed", size = 0.5) +

  # Facet by Principal Component
  facet_wrap(~ Principal_Component, scales = "free_y") +

  # Adjust labels and theme
  labs(x = "Loading", y = "", color = "Principal Component") +
  theme_minimal() +
  theme(legend.position = "none") # Hide legend since facets already distinguish components

# Display the plot
print(loadings_plot)
```



Look first at PC1: the variables with the largest loadings (points farthest in either direction from the zero line) are Child Poverty (positive), Employed (negative), Income per capita (negative), Poverty (positive), and Unemployment (positive). We know from exploring the correlation matrix that employment rate, unemployment rate, and income per capita are all related, and similarly child poverty rate and poverty rate are related. Therefore, the positively-loaded variables are all measuring more or less the same thing, and likewise for the negatively-loaded variables.

Essentially, then, PC1 is predominantly (but not entirely) a representation of income and poverty. In particular, counties have a higher value for PC1 if they have lower-than-average income per capita and higher-than-average poverty rates, and a smaller value for PC1 if they have higher-than-average income per capita and lower-than-average poverty rates.

A system for loading interpretation

Often interpreting principal components can be difficult, and sometimes there's no clear interpretation available! That said, it helps to have a system instead of staring at the plot and scratching our heads. Here is a semi-systematic approach to interpreting loadings:

1. Divert your attention away from the zero line.
2. Find the largest positive loading, and list all variables with similar loadings.
3. Find the largest negative loading, and list all variables with similar loadings.
4. The principal component represents the difference between the average of the first set and the average of the second set.
5. Try to come up with a description of less than 4 words.

This system is based on the following ideas:

- a high loading value (negative or positive) indicates that a variable strongly influences the principal component;
- a negative loading value indicates that
  - increases in the value of a variable *decrease* the value of the principal component
  - and decreases in the value of a variable *increase* the value of the principal component;
- a positive loading value indicates that
  - increases in the value of a variable *increase* the value of the principal component
  - and decreases in the value of a variable *decrease* the value of the principal component;
- similar loadings between two or more variables indicate that the principal component reflects their *average*;
- divergent loadings between two sets of variables indicates that the principal component reflects their *difference*.

Question 6

Work with your neighbor to interpret PC2. Come up with a name for the component and explain which variables are most influential.

YOUR ANSWER:

(Type your answer here, replacing this text.)

Standardization

Data are typically standardized because otherwise the variables on the largest scales tend to dominate the principal components, and most of the time PC1 will capture the majority of the variation. However, that is artificial. In the census data, income per capita has the largest magnitudes, and thus, the highest variance.

```r
# Compute column-wise variances
var_values <- apply(x_mx, 2, var)

# Sort in descending order and get top 3 variances
top_3_vars <- sort(var_values, decreasing = TRUE)[1:3]

# Print results
top_3_vars
```

```
IncomePerCap     Minority        White
3.804072e+07 5.265263e+02 5.264985e+02
```

When PCs are computed without normalization, the total variation is mostly just the variance of income per capita because it is orders of magnitude larger than the variance of any other variable. But that's just because of the *scale* of the variable – incomes per capita are large numbers – not a reflection that it varies more or less than the other variables.

Run the cell below to see what happens to the variance ratios if the data are not normalized.

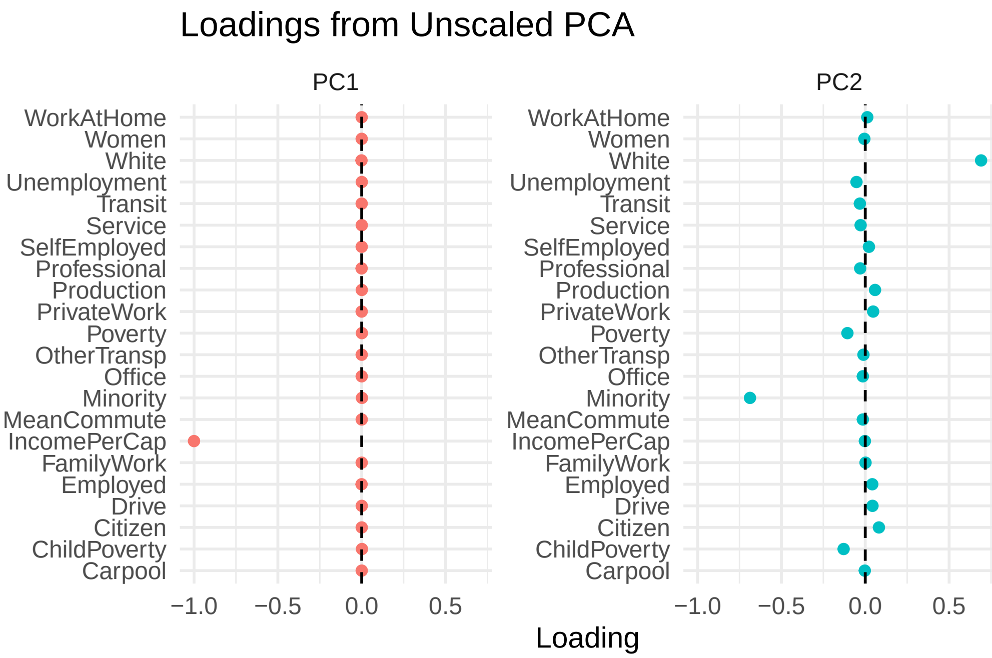
```r
# Recompute PCA without standardization
pca_unscaled <- prcomp(x_mx, center = TRUE, scale. = FALSE)

# Compute variance ratios for the first three principal components
var_ratios_unscaled <- (pca_unscaled$sdev^2) / sum(pca_unscaled$sdev^2)

# Show variance ratios for the first three PCs
var_ratios_unscaled[1:3]
```

```
[1] 9.999649e-01 2.535162e-05 2.831369e-06
```

Further, let's look at the loadings when data are not standardized:

```r
# Subset loadings (first two principal components)
unscaled_loading_df <- pca_unscaled$rotation[, 1:2]

# Rename columns to "PC1", "PC2"
colnames(unscaled_loading_df) <- c("PC1", "PC2")

# Melt from wide to long format
unscaled_loading_plot_df <- unscaled_loading_df %>%
  as.data.frame() %>%
```

```r
  tibble::rownames_to_column(var = "Variable") %>%
  pivot_longer(cols = -Variable, names_to = "Principal_Component", values_to = "Loading") %>%
  mutate(zero = 0)  # Add column for x = 0 reference line

# Create base plot
unscaled_loading_plot <- ggplot(unscaled_loading_plot_df, aes(x = Loading, y = Variable, colo

  # Add lines + points for loadings
  geom_line(aes(group = Variable)) +
  geom_point() +

  # Add vertical reference line at x = 0
  geom_vline(aes(xintercept = zero), color = "black", linetype = "dashed", size = 0.5) +

  # Facet by Principal Component
  facet_wrap(~ Principal_Component, scales = "free_y") +

  # Adjust labels, theme, and title
  labs(x = "Loading", y = "", color = "Principal Component", title = "Loadings from Unscaled
  theme_minimal() +
  theme(legend.position = "none")  # Hide legend since facets already distinguish components

# Display the plot
print(unscaled_loading_plot)
```



Notice that the variables with nonzero loadings in unscaled PCA are simply the three variables with the largest variances.

```
# Compute column-wise variances
var_values <- apply(x_mx, 2, var)

# Sort in descending order and get top 3 variances
top_3_vars <- sort(var_values, decreasing = TRUE)[1:3]

# Print results
top_3_vars
```

```
IncomePerCap     Minority        White
3.804072e+07 5.265263e+02 5.264985e+02
```

Exploratory analysis based on PCA

Now that we have the principal components, we can use them for exploratory data visualizations. To this end, let's retrieve the scores from the components you selected:

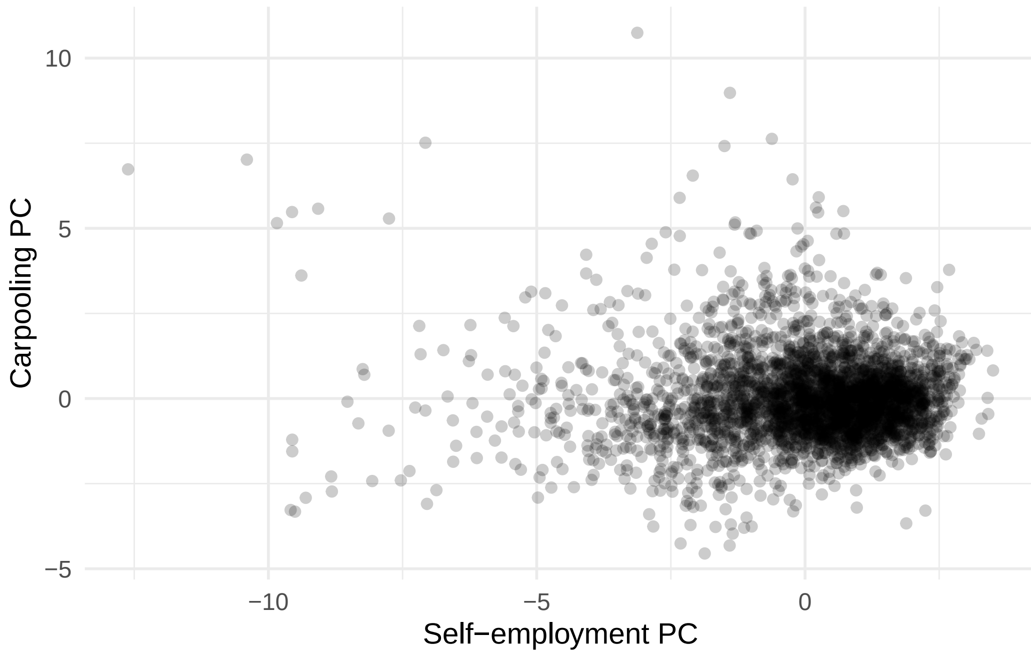
```
# Subset scores (first `num_pc` principal components)
score_df <- as.data.frame(pca$x[, 1:num_pc])
```

```r
# Rename columns to "PC1", "PC2", ..., "PC{num_pc}"
colnames(score_df) <- paste0("PC", seq(1, num_pc))

# Add State and County columns from census data
score_df <- cbind(score_df, census[, c("State", "County")])

# Print first few rows
head(score_df)
```

```
        PC1         PC2         PC3         PC4         PC5   State  County
1  0.0688066  1.64753870  0.74983510 -0.50051714 -0.44913788 Alabama Autauga
2  0.7028141  1.42878047  0.99971360 -1.16512454 -0.03940984 Alabama Baldwin
3 -4.0133953  0.07130920 -0.70435000  0.19534093  0.12839719 Alabama Barbour
4 -1.5564781  1.08025742 -1.89286275  1.54379328  0.94226487 Alabama    Bibb
5 -0.6367626  2.47324097 -0.20196611  0.09517785 -0.09961437 Alabama  Blount
6 -4.2976081 -0.02837128 -0.06098282  2.26121481  0.15232736 Alabama Bullock
```

The PC's can be used to construct scatterplots of the data and search for patterns. We'll illustrate that by identifying some outliers. The cell below plots PC2 (employment type) against PC4 (carpooling?):

```r
# Create scatter plot of PC2 vs. PC4
scatter_plot <- ggplot(score_df, aes(x = PC2, y = PC4)) +

  # Add scatter points with transparency (opacity = 0.2)
  geom_point(alpha = 0.2) +

  # Set axis labels
  labs(x = "Self-employment PC", y = "Carpooling PC") +

  # Use a minimal theme
  theme_minimal()

# Display the plot
print(scatter_plot)
```

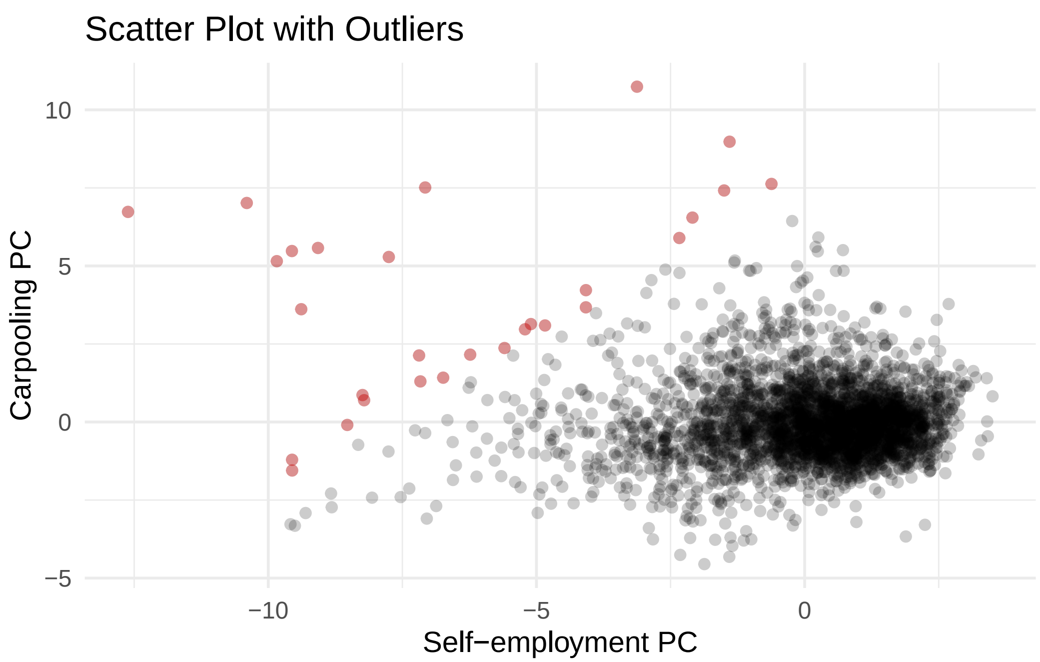


Notice that there are a handful of outlying points in the upper right region away from the dense scatter. What are those?

In order to inspect the outlying counties, we first need to figure out how to identify them. The outlying values have a large *sum* of PC2 and PC4. We can distinguish them by finding a cutoff value for the sum; a simple quantile will do.

```
# Compute cutoff value (99.999th percentile) for PC2 + PC4
pc2_pc4_sum <- score_df$PC2 + score_df$PC4
cutoff <- quantile(pc2_pc4_sum, probs = 0.99999)

# Store outlying rows using cutoff
outliers <- score_df %>% filter((-PC2 + PC4) > cutoff)

# Create scatter plot of all data points
scatter_plot <- ggplot(score_df, aes(x = PC2, y = PC4)) +

  # Base scatter points (opacity = 0.2)
  geom_point(alpha = 0.2, color = "black") +

  # Overlay outliers in red (opacity = 0.3)
  geom_point(data = outliers, aes(x = PC2, y = PC4), color = "red", alpha = 0.3) +

  # Set axis labels
```

```
  labs(x = "Self-employment PC", y = "Carpooling PC", title = "Scatter Plot with Outliers") +

  # Use a minimal theme
  theme_minimal()

# Display the plot
print(scatter_plot)
```



Notice that almost all the outlying counties are remote regions of Alaska:

```
# Print the outliers
print(outliers)
```

```
         PC1        PC2         PC3         PC4           PC5      State
1  1.3476088  -3.125398 -1.24839628 10.74231861 -0.5120004696     Alaska
2  1.8672816  -1.501056  0.55956956  7.41718614 -0.6855812250     Alaska
3 -3.4070436  -9.073309  4.39492052  5.57572033  4.8280672864     Alaska
4 -1.5131398  -7.752609  3.70539581  5.28694928  3.7009642300     Alaska
5  1.6650159  -7.187764  0.55919696  2.13247897  2.5642150455     Alaska
6 -5.8276148 -12.614982  5.39465231  6.73153945  7.8139571740     Alaska
7 -0.9434993 -10.400522  3.81175940  7.01675963  5.5331076564     Alaska
8 -2.8829379  -9.840638  3.77394489  5.15268190  5.1681718969     Alaska
9 -2.3978951  -7.074290  3.02860920  7.51355381  2.6257324223     Alaska
```

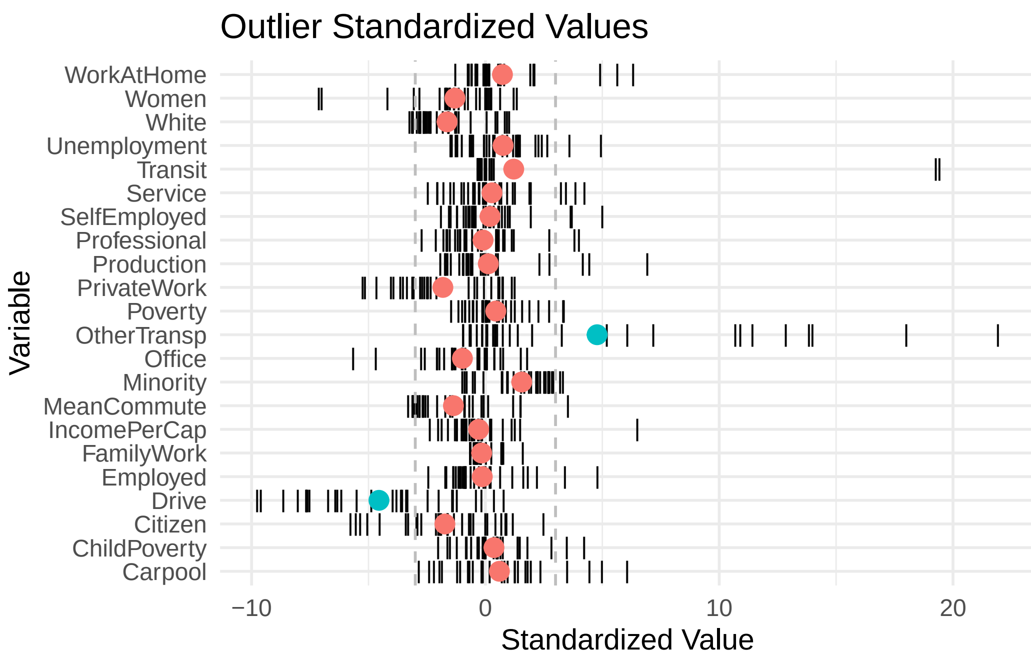
```
10 -3.1284344  -9.559099  4.16475406  5.47747285  4.8936403761       Alaska
11 -0.4394263  -4.840019 -0.22626769  3.09313414  1.8736093200       Alaska
12  2.4602279  -4.078486 -0.02658797  3.67412325  0.5617258106       Alaska
13 -2.2712389  -9.385133  2.50391841  3.61337657  4.1876770309       Alaska
14  1.2392797  -5.103495  2.02335153  3.13732156  6.4538134016     Colorado
15  0.2874233  -1.398938 -2.06736266  8.97870868  2.3748786852      Indiana
16 -2.1022126  -0.617782  0.56130706  7.62713085 -2.5346146495       Kansas
17  7.2155797  -9.555783 -4.24211717 -1.20923239 -2.3602821661      Montana
18  5.8296555  -7.163431 -3.27114114  1.30122986 -1.9696770406     Nebraska
19 -4.0037504  -5.595001 11.94810679  2.37077570  7.1319785834     New York
20  5.1316287  -6.738516 14.28412618  1.42073588  7.2312917468     New York
21  1.2491869  -2.091100 -1.73332988  6.54858915  1.8856527391         Ohio
22 -4.4791512  -8.242810 -3.00613174  0.86430665  0.1976501361  Puerto Rico
23  6.4631635  -8.526588 -4.19644643 -0.09358657 -2.6220995002 South Dakota
24 -1.5724113  -9.554963 -1.61230130 -1.55217987 -3.2293765691 South Dakota
25 -6.1407935  -8.212792  0.82330802  0.69811535 -0.9650185979 South Dakota
26 -1.5955357  -2.336449 -0.05921648  5.89501561 -0.0005786166        Texas
27 -5.0553722  -5.213103  0.77966169  2.97038003 -1.8969311248        Texas
28 -2.2727363  -6.235229  1.71395490  2.15837949 -2.2726158400        Texas
29 -6.2750543  -4.077441  1.47813602  4.22427281 -2.2610794822        Texas
                             County
1            Aleutians East Borough
2        Aleutians West Census Area
3                Bethel Census Area
4            Dillingham Census Area
5         Hoonah-Angoon Census Area
6              Kusilvak Census Area
7        Lake and Peninsula Borough
8                  Nome Census Area
9               North Slope Borough
10          Northwest Arctic Borough
11 Prince of Wales-Hyder Census Area
12          Yakutat City and Borough
13        Yukon-Koyukuk Census Area
14                         San Juan
15                         LaGrange
16                           Seward
17                           Carter
18                           Arthur
19                            Bronx
20                         New York
21                           Holmes
22                          Culebra
```

```
23                              Harding
24                             Mellette
25                                 Todd
26                                Garza
27                             Hudspeth
28                             Presidio
29                                Starr
```

What sets them apart? The cell below retrieves the normalized data and county name for the outlying rows, and then plots the Standardized values of each variable for all 9 counties as vertical ticks, along with a point indicating the mean for the outlying counties. This plot can be used to determine which variables are over- or under-average for the outlying counties relative to the nation by simply locating means that are far from zero in either direction.

```r
# Standardize x_mx (mean = 0, std = 1)
x_ctr <- as.data.frame(scale(x_mx))  # Center and scale data

# Convert row names of outliers to numeric indices
outlier_indices <- match(outliers$County, census$County)

# Retrieve normalized data for outlying rows & join with County info
outlier_data <- x_ctr %>%
  slice(outlier_indices) %>%  # Select only outlier rows
  mutate(County = census$County[outlier_indices])  # Add County column

# Melt (reshape) data from wide to long format
outlier_plot_df <- outlier_data %>%
  pivot_longer(cols = -County, names_to = "Variable",
               values_to = "Standardized_value")

# Compute means of each variable across counties
means_df <- outlier_plot_df %>%
  group_by(Variable) %>%
  summarize(group_mean = mean(Standardized_value)) %>%
  mutate(large = abs(group_mean) > 3)  # Flag large deviations

# Create base plot
ticks_plot <- ggplot(outlier_plot_df,
                     aes(x = Standardized_value, y = Variable)) +

  # Add tick marks for outlier values
  geom_point(shape = "|", size = 3) +
```

```r
  # Add vertical lines for ±3 SD range
  geom_vline(xintercept = c(-3, 3), linetype = "dashed", color = "gray") +

  # Add mean markers
  geom_point(data = means_df,
             aes(x = group_mean, y = Variable, color = large),
             size = 3) +

  # Axis labels and minimal theme
  labs(x = "Standardized Value", y = "Variable",
       title = "Outlier Standardized Values") +
  theme_minimal() +
  theme(legend.position = "none")

# Display plot
print(ticks_plot)
```



Question 7

The two variables that clearly set the outlying counties apart from the nation are the percentage of the population using alternative transportation (extremely above average) and the percentage that drive to work (extremely below average). What about those counties explains this?

(*Hint:* take a peek at the Wikipedia page on transportation in Alaska.)

YOUR ANSWER:

(Type your answer here, replacing this text.)

Submission

1. Save the notebook.

2. Restart the kernel and run all cells. (**CAUTION**: if your notebook is not saved, you will lose your work.)

3. Carefully look through your notebook and verify that all computations execute correctly. You should see **no errors**; if there are any errors, make sure to correct them before you submit the notebook.

4. Download the notebook as an `.qmd` file. This is your backup copy.

5. Export the notebook as PDF and upload to Canvas.