Principal Component Analysis Demo

Vishesh Saharan, Jonathan Ramos

2023-11-04

Wisconsin Diagnostic Breast Cancer (WDBC)

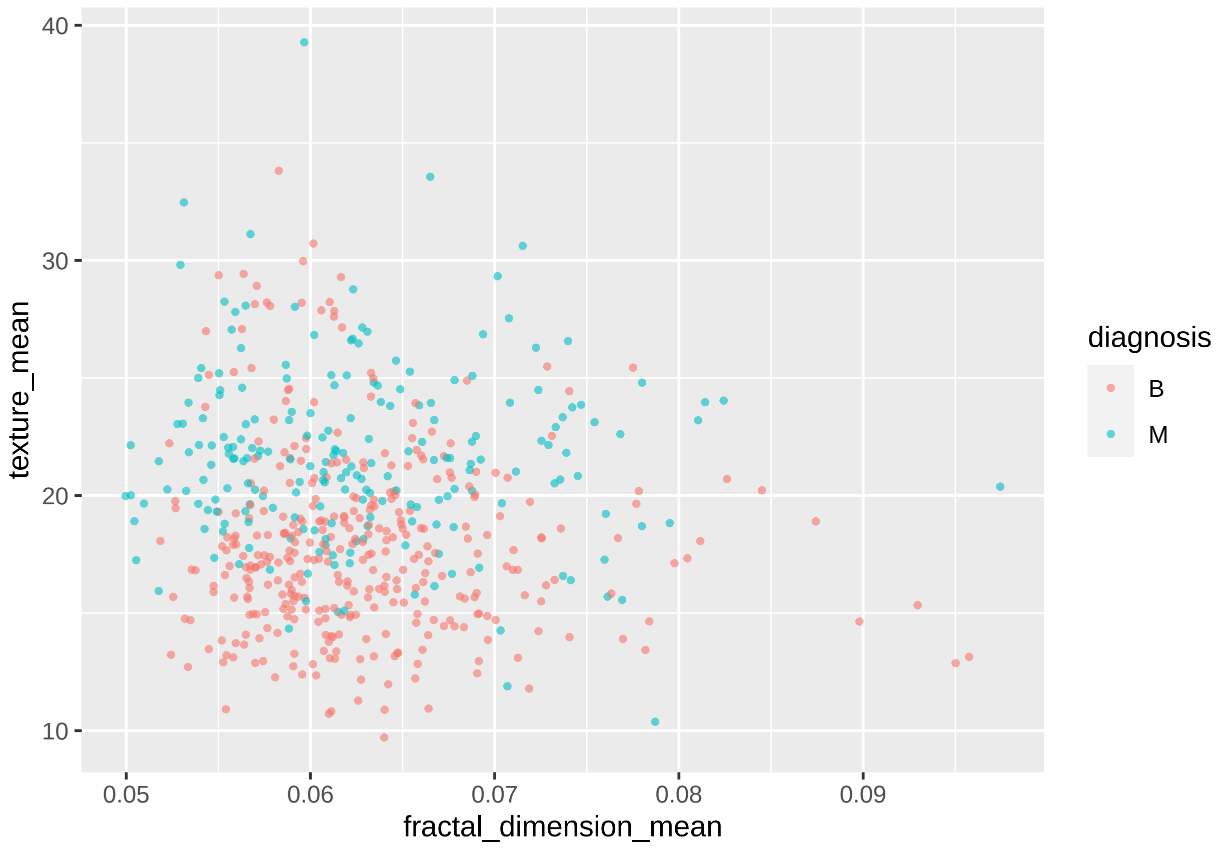
Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. A few of the images can be found at http://www.cs.wisc.edu/~street/images/ The labels 'B' and 'M' represent denote whether a sample was benign or malignant respectively.

One potential question of interest would be "which feature or subset of features contributes most to the variability in this set?" To address this question, we can use principal component analysis to more carefully inspect how each variable contributes to the variance of the data. Answering this question can be a useful step in building models or guiding further research questions to investigate specific properties of benign vs malignant samples.

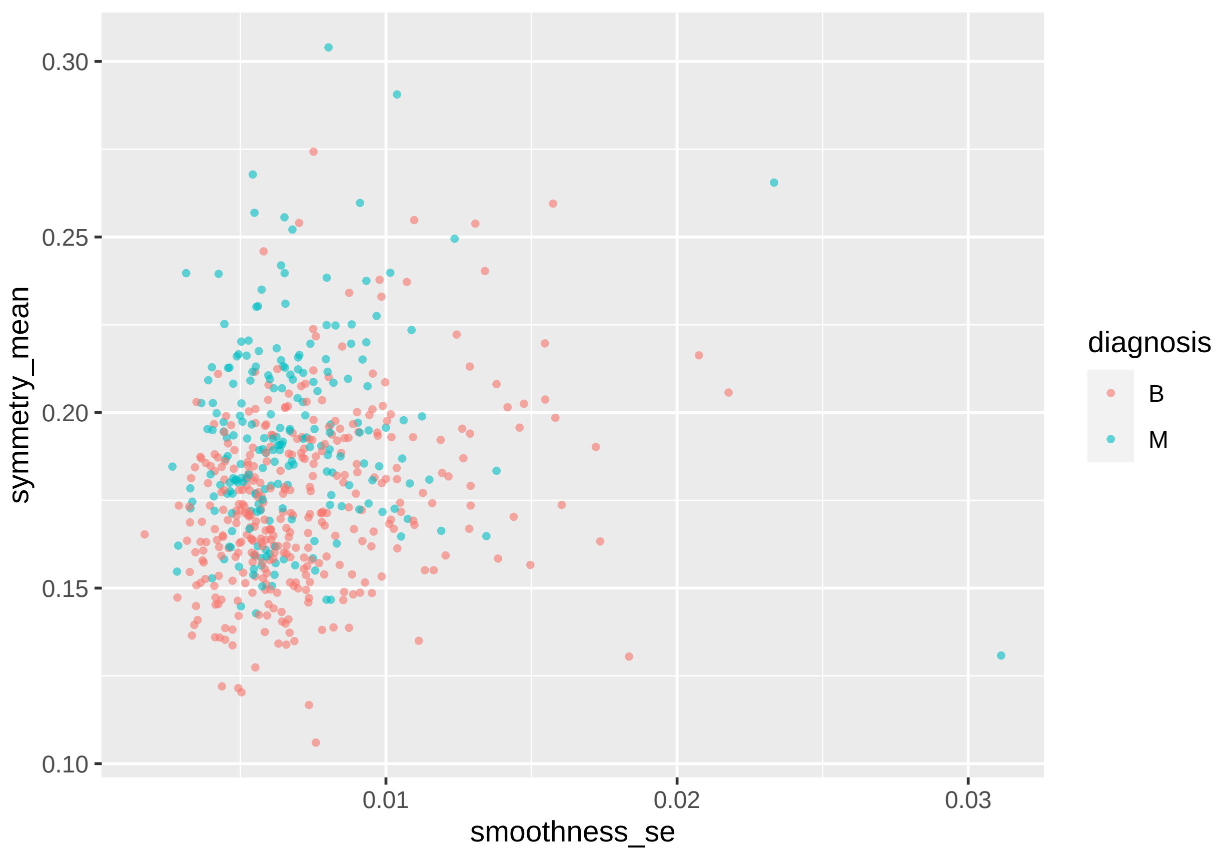
```r
# load in data
wdbc <- read.csv('wdbc_data.csv')

# check cols; these data are well suited for PCA because we have many columns of numerical data
str(wdbc)
```

```
## 'data.frame':    569 obs. of  32 variables:
##  $ id                     : int  842302 842517 84300903 84348301 84358402 843786 844359 84458202 8449
##  $ diagnosis              : chr  "M" "M" "M" "M" ...
##  $ radius_mean            : num  18 20.6 19.7 11.4 20.3 ...
##  $ texture_mean           : num  10.4 17.8 21.2 20.4 14.3 ...
##  $ perimeter_mean         : num  122.8 132.9 130 77.6 135.1 ...
##  $ area_mean              : num  1001 1326 1203 386 1297 ...
##  $ smoothness_mean        : num  0.1184 0.0847 0.1096 0.1425 0.1003 ...
##  $ compactness_mean       : num  0.2776 0.0786 0.1599 0.2839 0.1328 ...
##  $ concavity_mean         : num  0.3001 0.0869 0.1974 0.2414 0.198 ...
##  $ concave_points_mean    : num  0.1471 0.0702 0.1279 0.1052 0.1043 ...
##  $ symmetry_mean          : num  0.242 0.181 0.207 0.26 0.181 ...
##  $ fractal_dimension_mean : num  0.0787 0.0567 0.06 0.0974 0.0588 ...
##  $ radius_se              : num  1.095 0.543 0.746 0.496 0.757 ...
##  $ texture_se             : num  0.905 0.734 0.787 1.156 0.781 ...
##  $ perimeter_se           : num  8.59 3.4 4.58 3.44 5.44 ...
##  $ area_se                : num  153.4 74.1 94 27.2 94.4 ...
##  $ smoothness_se          : num  0.0064 0.00522 0.00615 0.00911 0.01149 ...
##  $ compactness_se         : num  0.049 0.0131 0.0401 0.0746 0.0246 ...
##  $ concavity_se           : num  0.0537 0.0186 0.0383 0.0566 0.0569 ...
##  $ concave_points_se      : num  0.0159 0.0134 0.0206 0.0187 0.0188 ...
##  $ symmetry_se            : num  0.03 0.0139 0.0225 0.0596 0.0176 ...
##  $ fractal_dimension_se   : num  0.00619 0.00353 0.00457 0.00921 0.00511 ...
##  $ radius_worst           : num  25.4 25 23.6 14.9 22.5 ...
##  $ texture_worst          : num  17.3 23.4 25.5 26.5 16.7 ...
##  $ perimeter_worst        : num  184.6 158.8 152.5 98.9 152.2 ...
##  $ area_worst             : num  2019 1956 1709 568 1575 ...
##  $ smoothness_worst       : num  0.162 0.124 0.144 0.21 0.137 ...
##  $ compactness_worst      : num  0.666 0.187 0.424 0.866 0.205 ...
##  $ concavity_worst        : num  0.712 0.242 0.45 0.687 0.4 ...
##  $ concave_points_worst   : num  0.265 0.186 0.243 0.258 0.163 ...
##  $ symmetry_worst         : num  0.46 0.275 0.361 0.664 0.236 ...
##  $ fractal_dimension_worst: num  0.1189 0.089 0.0876 0.173 0.0768 ...
```

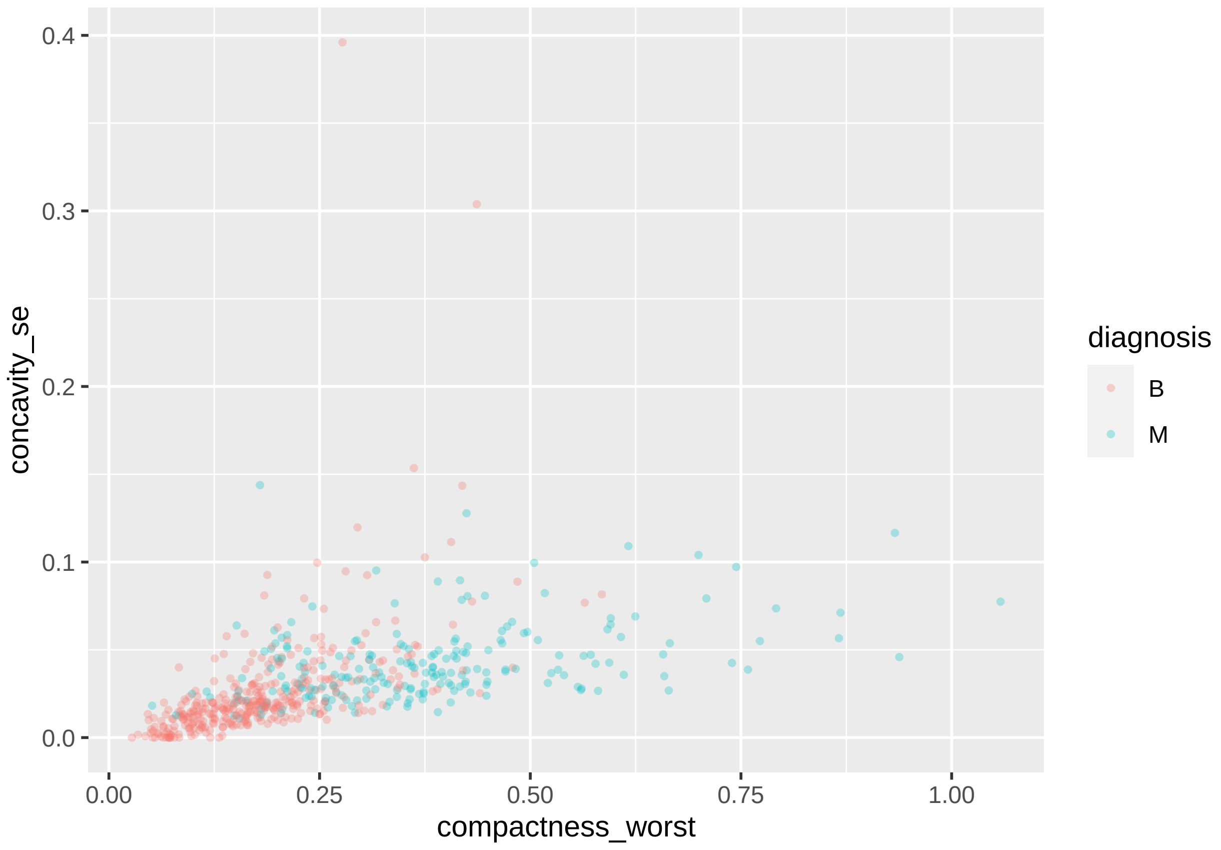
```
# let's make some 2D plots with random pairs of features
# some definitely look better than others, which of these best represents the patterns in our data?
ggplot(wdbc, aes(x=fractal_dimension_mean, y=texture_mean, color=diagnosis)) + geom_point(alpha=0.6, si
```



```
ggplot(wdbc, aes(x=smoothness_se, y=symmetry_mean, color=diagnosis)) + geom_point(alpha=0.6, size=0.8)
```



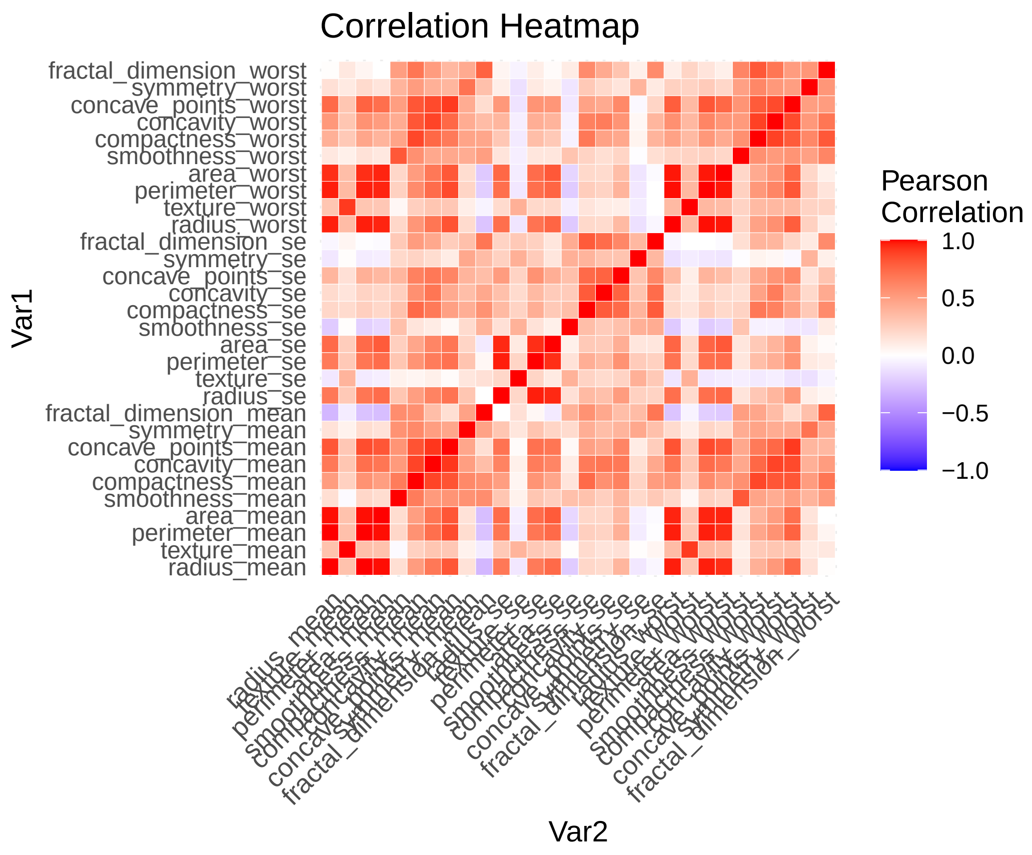
```
ggplot(wdbc, aes(x=compactness_worst, y=concavity_se, color=diagnosis)) + geom_point(alpha=0.3, size=0.8
```



```r
# correlation matrix
cormat <- round(cor(wdbc[,-2:-1]),2)

melted_cormat <- melt(cormat)
head(melted_cormat)
```

```
##               Var1        Var2 value
## 1      radius_mean radius_mean  1.00
## 2     texture_mean radius_mean  0.32
## 3   perimeter_mean radius_mean  1.00
## 4        area_mean radius_mean  0.99
## 5  smoothness_mean radius_mean  0.17
## 6 compactness_mean radius_mean  0.51
```

```r
ggplot(data = melted_cormat, aes(Var2, Var1, fill = value))+
 geom_tile(color = "white")+
 scale_fill_gradient2(low = "blue", high = "red", mid = "white",
   midpoint = 0, limit = c(-1,1), space = "Lab",
   name="Pearson\nCorrelation") +
  theme_minimal()+
  theme(axis.text.x = element_text(angle = 45, vjust = 1.01,
    size = 9.5, hjust = 1))+
  coord_fixed() +
  ggtitle('Correlation Heatmap')
```



The exploratory 2D scatter plots are only one way of looking at some of the relationships in these data. Since we have 30 features (and we can't make a 30-dimensional plot), it would be extremely tedious to evaluate every possible pairwise combination of features. If we reduce the number of dimensions with PCA we can better visualize patterns of variance in the data. The correlation heat map indicates multicolinearity in this dataset which causes problems in regression. This means that we can apply PCA to generate new orthogonal, uncorrelated features. By using principal components as new features on which to build models, benefits of PCA also include improving model performance, interpretability, and generalizeability.

Eigenvalue Decomposition of the Covariance Matrix

The assumption of linearity allows us to frame PCA as a change of basis. In short, PCA boils down to a change of basis from our original feature space into a new orthonormal basis defined by the eigenvectors of the covariance matrix. These eigenvectors that we consider as principal components or PCs, are derived from linear combinations of our original features which point in the direction of maximal variance. We may select a subset of PCs based on their respective eigenvalues to project our original data onto.

Recall that covariance is a measure which represents the degree to which 2 random variables change together and is given by:

$cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{n-1}$

If we consider our data to be an m x n matrix with m measurements and n variables, we may build an n x n matrix where each position i,j represents the covariance between the ith and jth variables from our data. Note that down main diagonal, we have the covariance between a variable and itself, which is equal its variance. For example if we had three variables, or dimensions $x, yandz$, its covariance matrix looks like

$$C = \begin{bmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{bmatrix}$$

Because the covariance matrix is always square, we can compute its eigenvalues and eigenvectors. Eigenvectors have the special property of retaining their direction when a linear transformation is applied to it. Since we are applying this process onto the centered (and often standardized) covariance matrix, we end up with a set of orthogonal vectors in the direction of maximal variance which we can use to form a new basis for our data.

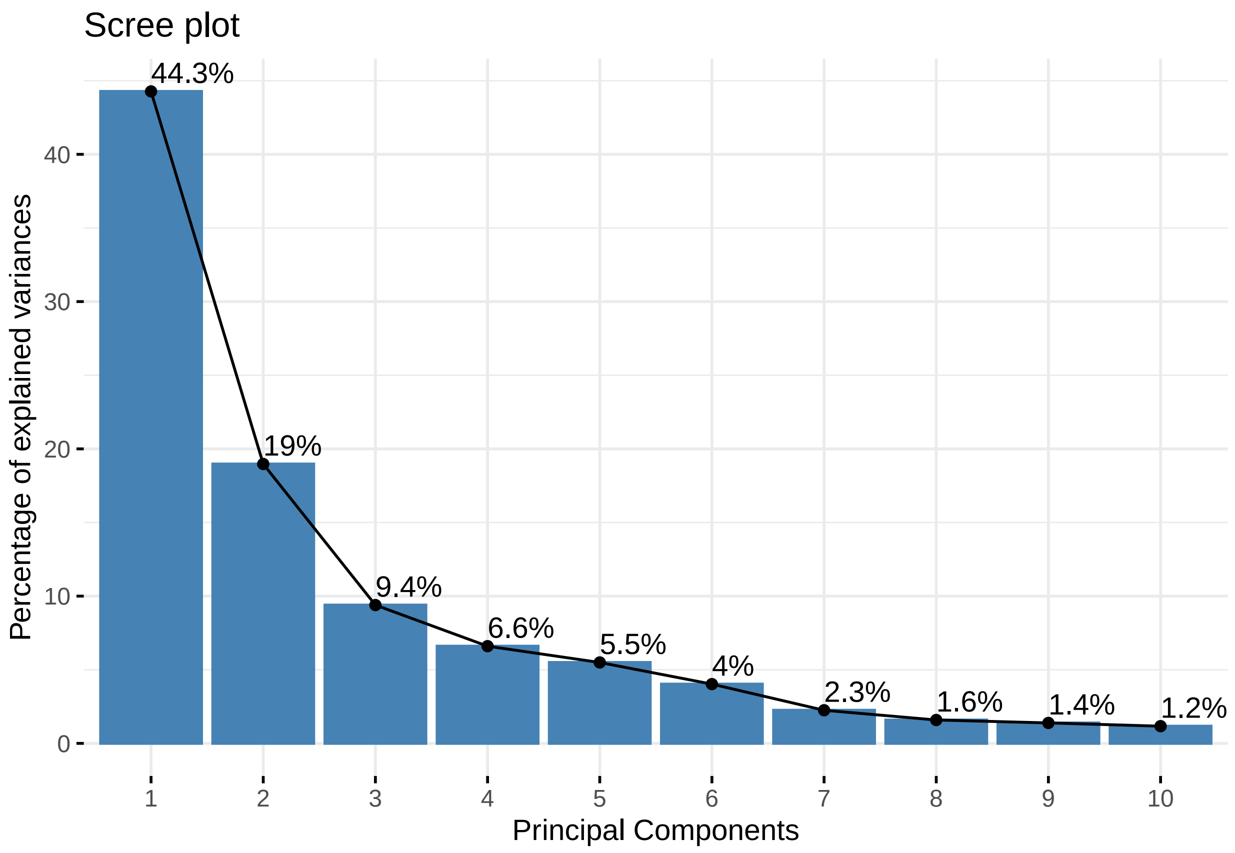
R function: princomp

Our PCs are linear combinations of our original features. We can look at the scalar values in each of these eigenvectors and determine how much each original feature contributes to the PC. These scalars are called loadings and the projected data points are called scores. We can carry out PCA with the built in function princomp(). This function offers an argument called cor to center and standardize the data. Additionally, the argument called scores denotes whether or not to compute coordinates on each PC (the scores).

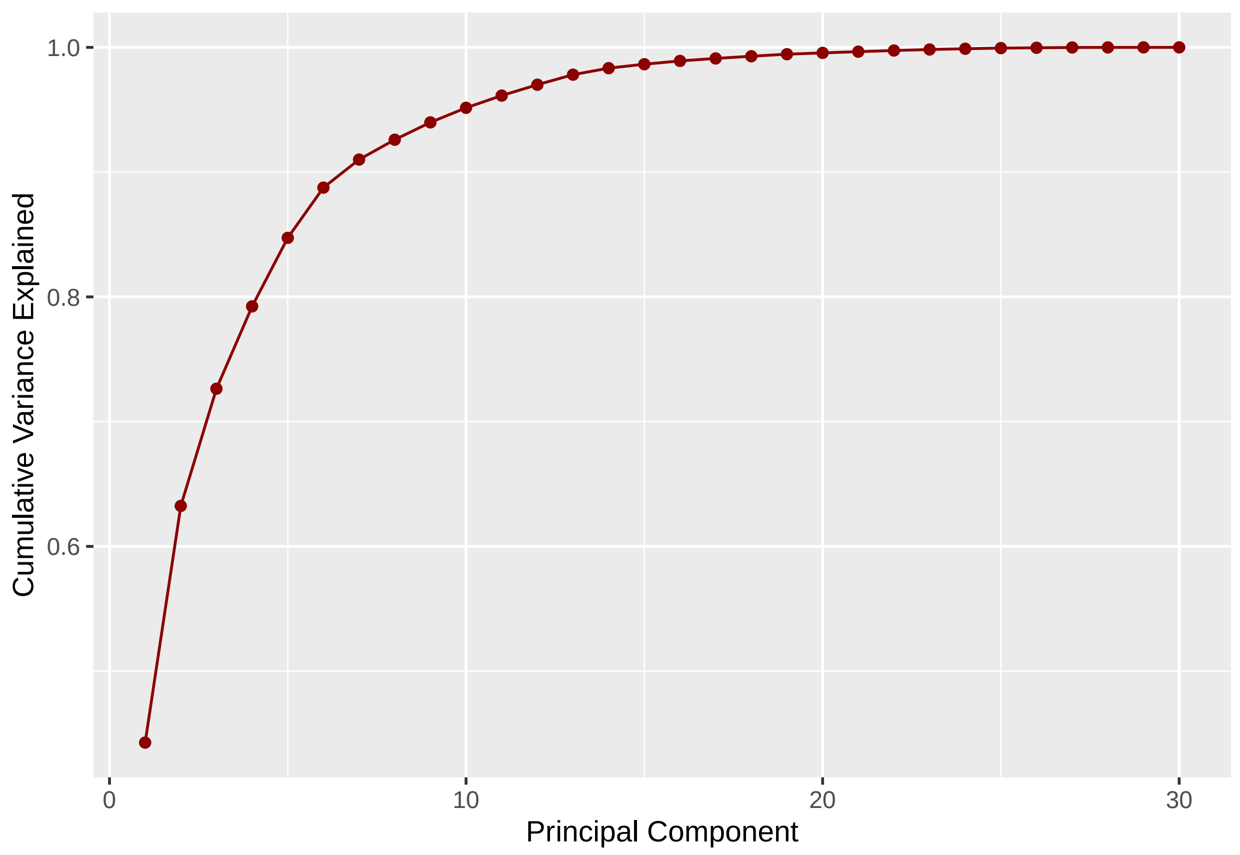
```r
# removing labels
wdbc.features <- wdbc[,-2:-1]

# Apply PCA with princomp.
# args: cor=TRUE to center and standardize data, score=TRUE to compute scores
pca_result <- princomp(wdbc.features, cor=TRUE, scores=TRUE)

# Summary of the PCA results
summary <- summary(pca_result)
(summary)
```

```
## Importance of components:
##                           Comp.1    Comp.2     Comp.3     Comp.4     Comp.5
## Standard deviation     3.6443940 2.3856560 1.67867477 1.40735229 1.28402903
## Proportion of Variance 0.4427203 0.1897118 0.09393163 0.06602135 0.05495768
## Cumulative Proportion  0.4427203 0.6324321 0.72636371 0.79238506 0.84734274
##                            Comp.6     Comp.7     Comp.8     Comp.9    Comp.10
## Standard deviation     1.09879780 0.82171778 0.69037464 0.64567392 0.59219377
## Proportion of Variance 0.04024522 0.02250734 0.01588724 0.01389649 0.01168978
## Cumulative Proportion  0.88758796 0.91009530 0.92598254 0.93987903 0.95156881
##                           Comp.11     Comp.12    Comp.13     Comp.14
## Standard deviation     0.54213992 0.511039500 0.49128148 0.396244525
## Proportion of Variance 0.00979719 0.008705379 0.00804525 0.005233657
## Cumulative Proportion  0.96136600 0.970071383 0.97811663 0.983350291
##                            Comp.15     Comp.16     Comp.17     Comp.18
## Standard deviation     0.306814219 0.282600072 0.243719178 0.229387845
## Proportion of Variance 0.003137832 0.002662093 0.001979968 0.001753959
## Cumulative Proportion  0.986488123 0.989150216 0.991130184 0.992884143
##                            Comp.19     Comp.20      Comp.21      Comp.22
## Standard deviation     0.222435590 0.176520261 0.1731268145 0.1656484305
## Proportion of Variance 0.001649253 0.001038647 0.0009990965 0.0009146468
## Cumulative Proportion  0.994533397 0.995572043 0.9965711397 0.9974857865
##                             Comp.23      Comp.24      Comp.25     Comp.26
## Standard deviation     0.1560155049 0.1343689213 0.1244237573 0.090430304
## Proportion of Variance 0.0008113613 0.0006018336 0.0005160424 0.000272588
## Cumulative Proportion  0.9982971477 0.9988989813 0.9994150237 0.999687612
##                             Comp.27      Comp.28      Comp.29      Comp.30
## Standard deviation     0.0830690308 3.986650e-02 0.0273642668 1.153451e-02
## Proportion of Variance 0.0002300155 5.297793e-05 0.0000249601 4.434827e-06
## Cumulative Proportion  0.9999176271 9.999706e-01 0.9999955652 1.000000e+00
```

```
# visualization of variance explained by PCs
fviz_eig(pca_result, addlabels = TRUE, xlab = 'Principal Components')
```



```
# conversely, we may also plot the cumulative sum of variance explained
var_explained <- pca_result$sdev ** 2 / sum(pca_result$sdev**2)
cum_var_explained <- cumsum(var_explained)
var_explained_df <- data.frame(PC=1:length(var_explained), cumulative_var_explained=cum_var_explained)
ggplot(data=var_explained_df, aes(x=PC, y=cumulative_var_explained)) +
  geom_line(color='darkred') +
  geom_point(color='darkred') +
  xlab('Principal Component') +
  ylab('Cumulative Variance Explained')
```



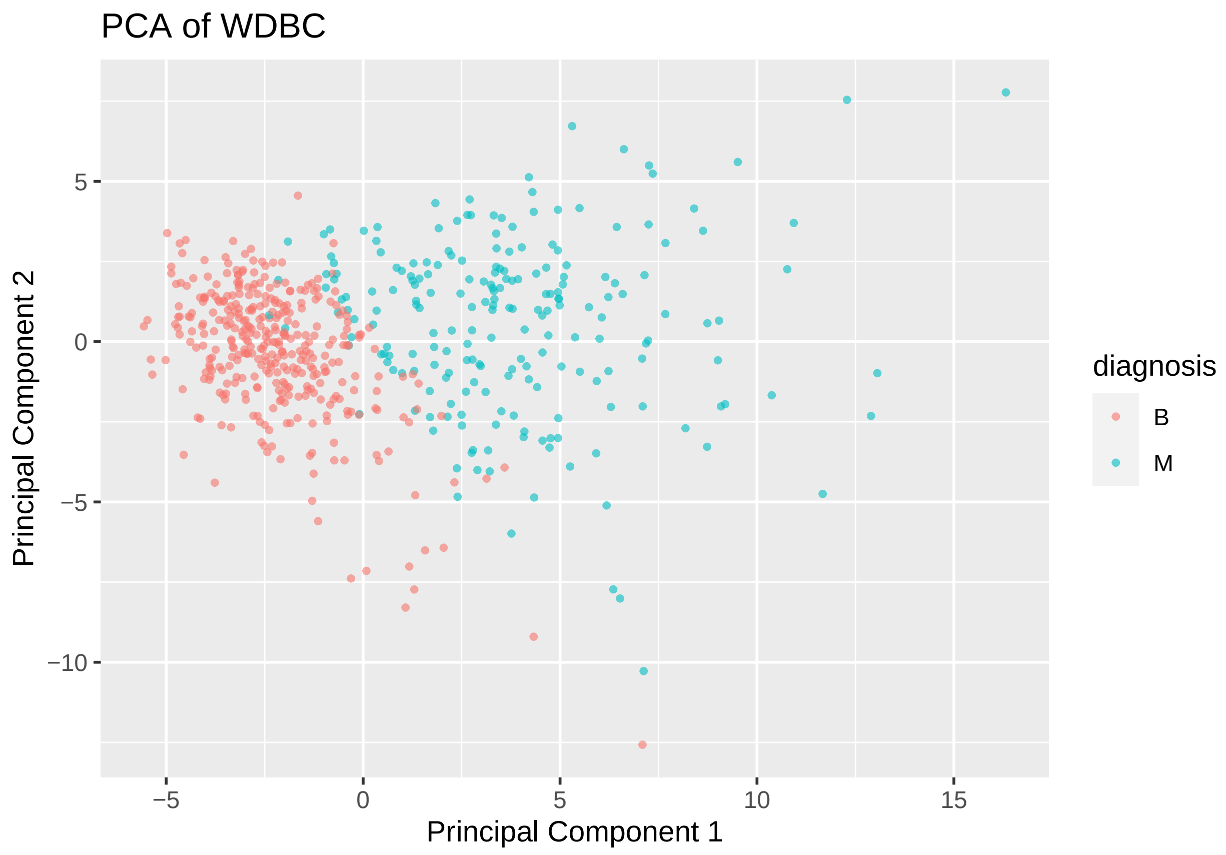
```
# both indicate we can explain over 90% of the variance with 7 PCs.
```

The scree plot above plots the percent variance explained of each component in descending order. The percent variance explained is computed by taking the sum of all the eigenvalues and dividing any given eigenvalue by the sum. The key take away is that the first few components explain the majority of the variance in the data while the last components contribute much less. This means that if we set a cut off at 90% variance explained we can select only the first seven PCs and still retain about 91.1% of the variance.

The cumulative variance explained plot is an alternative visualization of the same data as a scree plot, where each point, n, on the x axis represents the sum of the variance explained by the first n components. Note that this will always max out at 100% by the end of the plot where we consider all 30 PCs.

Now that we've got our PCs, we can select a subset of them to construct a new basis.

```
# Extract the PCA scores (coordinates of the data in the PCA space)
pca_scores <- pca_result$scores[,1:2]

# Create a data frame that includes the PCA scores and the species information
pca_data <- data.frame(diagnosis = wdbc$diagnosis, pca_scores)

# Use ggplot2 to create a scatter plot of the first two principal components
# note reflection about y axis because underlying implementation of PCA in prcomp() is SVD based
ggplot(data = pca_data, aes(x = Comp.1, y = Comp.2, color = diagnosis)) +
  geom_point(alpha = 0.6, size = 0.8) +
  ggtitle("PCA of WDBC") +
  xlab("Principal Component 1") +
  ylab("Principal Component 2")
```



This 2D plot is definitely an improvement from our initial exploratory plots; While PCA does not maximize class separability, PCA allows us to see new patterns in the data the we couldn't visualize very well during the exploratory analyses.

There are a few important interpretations here: The eigenvalues represent the variance explained by its respective eigenvector. If we consider that the orthonormal eigenvector represents the direction of most variance, its eigenvalue represents its scale. Since we know that our PCs are ordered, the first few PCs likely describe the majority of the variance in these data. In this case, the proportion of variance explained by the first two PCs is 0.6324.

Since eigenvectors are orthogonal, they are uncorrelated. This can be useful when we need to eliminate mulitcolinearity between features or engineer new uncorrelated features from the original ones to improve model performance.

Inspecting scores

We mentioned previously that each PC is a linear combination of the original features. By inspecting the loadings of each PC more closely we can see what this means in terms of our original features.

```r
# cor=TRUE to center and standardize data, score=TRUE to compute scores
pca_result <- princomp(wdbc.features, cor=TRUE, scores=TRUE)

# inspect loading matrix for first 2 PCs
pca_result$loadings[, 1:2]
```

```
##                                 Comp.1      Comp.2
## radius_mean             0.21890244  0.233857132
## texture_mean            0.10372458  0.059706088
## perimeter_mean          0.22753729  0.215181361
## area_mean               0.22099499  0.231076711
## smoothness_mean         0.14258969 -0.186113023
## compactness_mean        0.23928535 -0.151891610
## concavity_mean          0.25840048 -0.060165363
## concave_points_mean     0.26085376  0.034767500
## symmetry_mean           0.13816696 -0.190348770
## fractal_dimension_mean  0.06436335 -0.366575471
## radius_se               0.20597878  0.105552152
## texture_se              0.01742803 -0.089979682
## perimeter_se            0.21132592  0.089457234
## area_se                 0.20286964  0.152292628
## smoothness_se           0.01453145 -0.204430453
## compactness_se          0.17039345 -0.232715896
## concavity_se            0.15358979 -0.197207283
## concave_points_se       0.18341740 -0.130321560
## symmetry_se             0.04249842 -0.183848000
## fractal_dimension_se    0.10256832 -0.280092027
## radius_worst            0.22799663  0.219866379
## texture_worst           0.10446933  0.045467298
## perimeter_worst         0.23663968  0.199878428
## area_worst              0.22487053  0.219351858
## smoothness_worst        0.12795256 -0.172304352
## compactness_worst       0.21009588 -0.143593173
## concavity_worst         0.22876753 -0.097964114
## concave_points_worst    0.25088597  0.008257235
## symmetry_worst          0.12290456 -0.141883349
## fractal_dimension_worst 0.13178394 -0.275339469
```

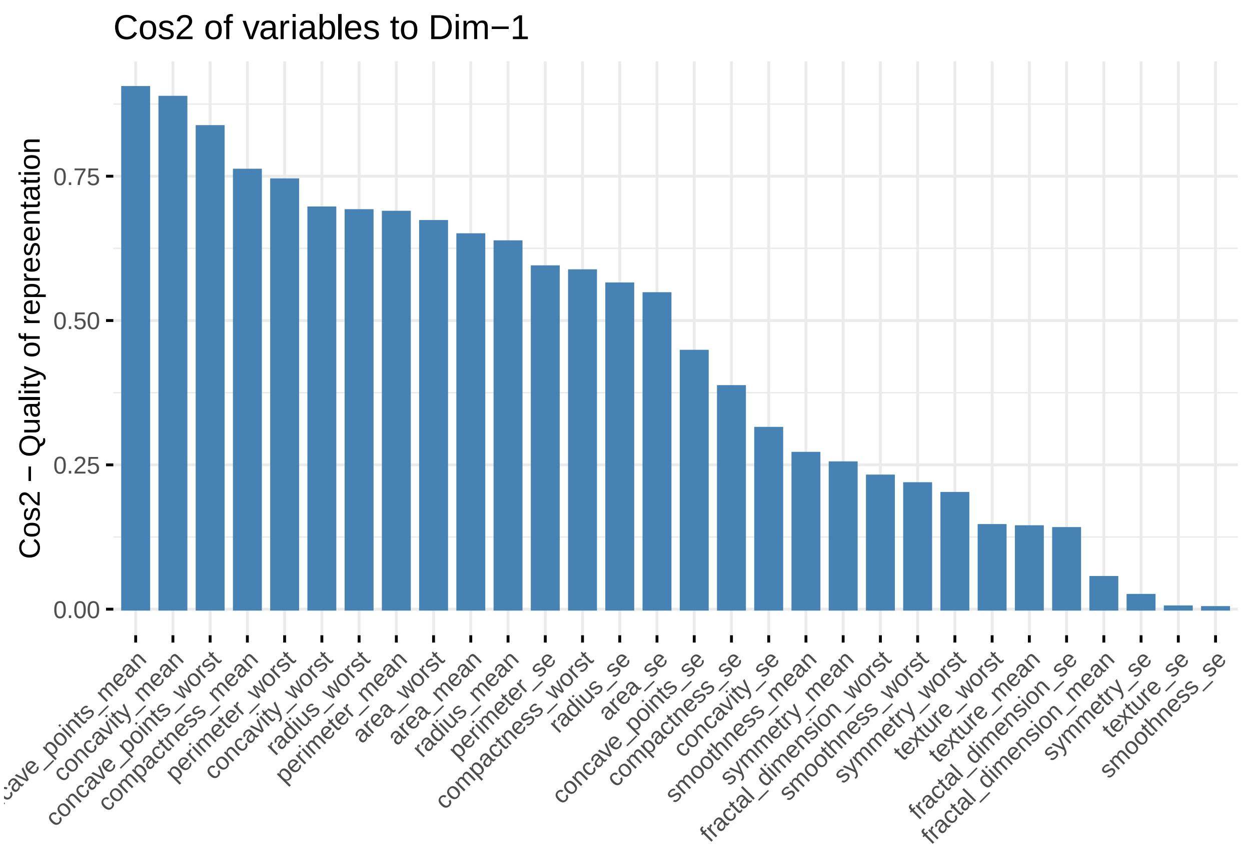
Loadings range from (-1,1) and loading with a high absolute value indicates that the variable has a strong influence in that particular component. We can think of the loadings essentially as the weight or contribution of a feature to a given PC. From the loading matrix above, we can see that concavity_mean and concave_points_mean have the highest loadings for PC1 which indicates that the concavity of cell bodies can be an important morphological trait that contributes to the variance of the data. Notably, perimeter_worst and radius_worst have relatively high loadings in both PC1 and PC2.

Quality of Representation

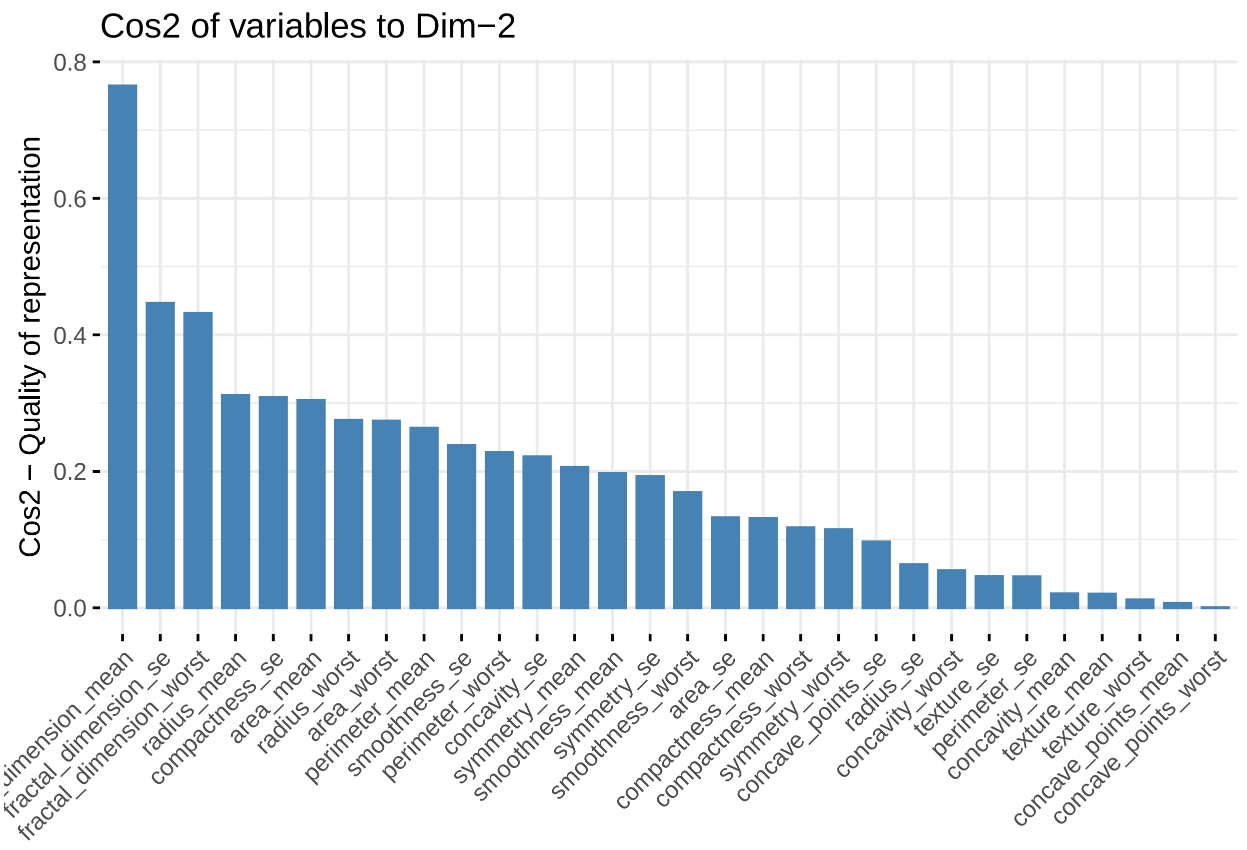
From the loadings we can see that not all features influence a given PC equally. Features that have a lot of influence in the PCs have a higher quality of representation. Features that do not influence PCs very much are not well represented by the PCs. The quality of the representation is computed by the squared cosine of the loadings. The squared cosine provides insight into how much of the variance in the feature is explained by that principal component and highlights which features are more influential and which are less relevant. A high squared cosine value indicates that a variable contributes significantly to the variance explained by the principal component, while a low value suggests that the variable has less influence on that component. This information can guide feature selection or further data analysis, as it allows us to focus on the most informative variables or components.

Plotting note: Biplots are useful for showing which features are correlated and which are not: features that are well correlated will cluster together on the biplot whereas features that re not correlated will be farther away (opposite) from each other. Further, features that contribute more to the variation in a PC will be represented by longer vectors that approach the edge of the circle while features that do not contribute very much will lie closer to the center of the circle.

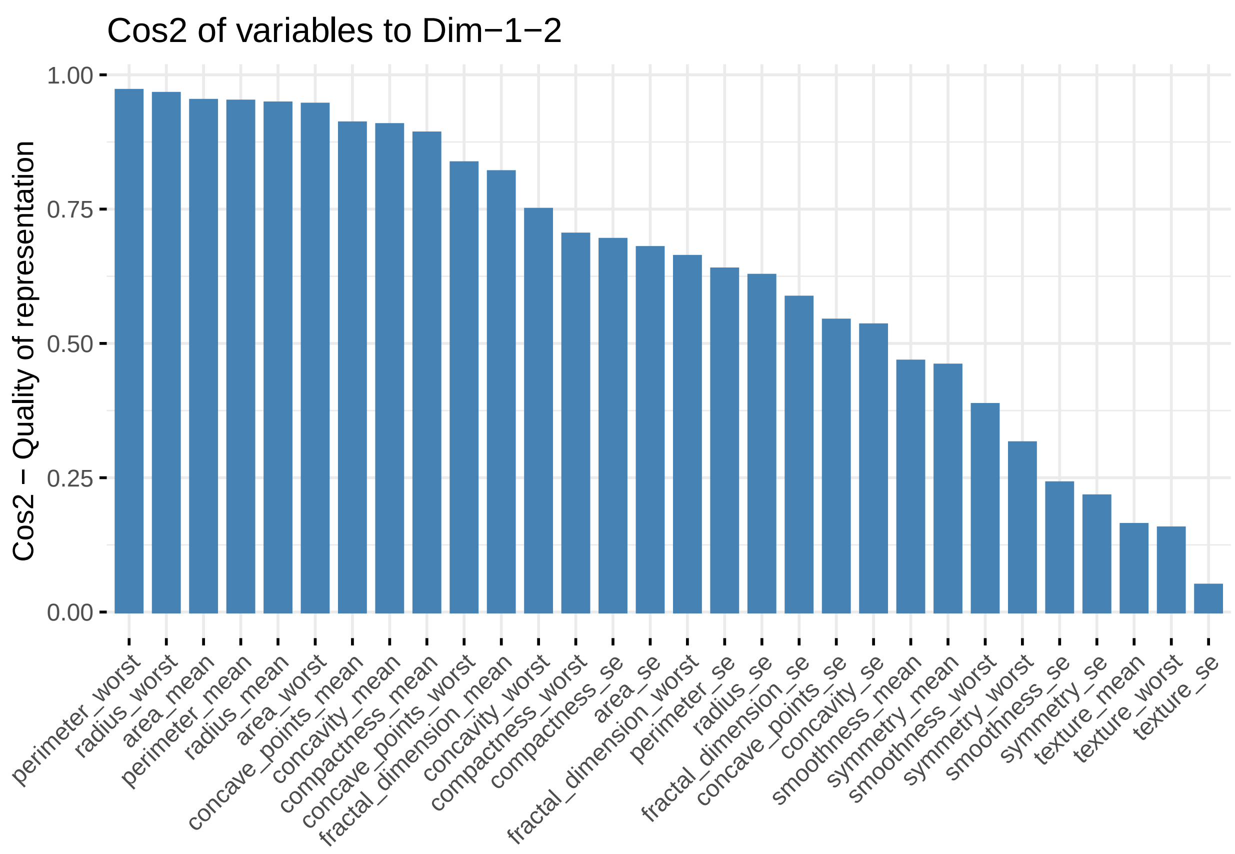
```r
# plot squared cosine across loadings for first PC
fviz_cos2(pca_result, choice = "var", axes = 1)
```



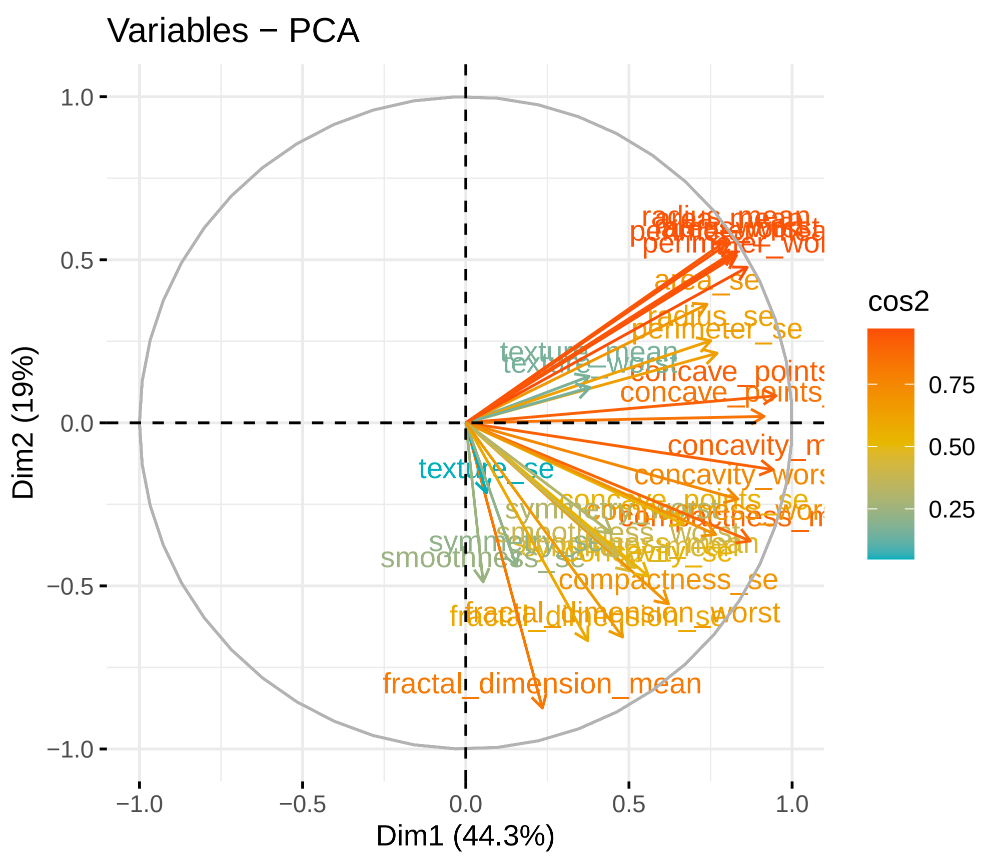
```r
# plot squared cosine across loadings for second PC
fviz_cos2(pca_result, choice = "var", axes = 2)
```



```r
# plot squared cosine across loadings for both PCs
fviz_cos2(pca_result, choice = "var", axes = 1:2)
```



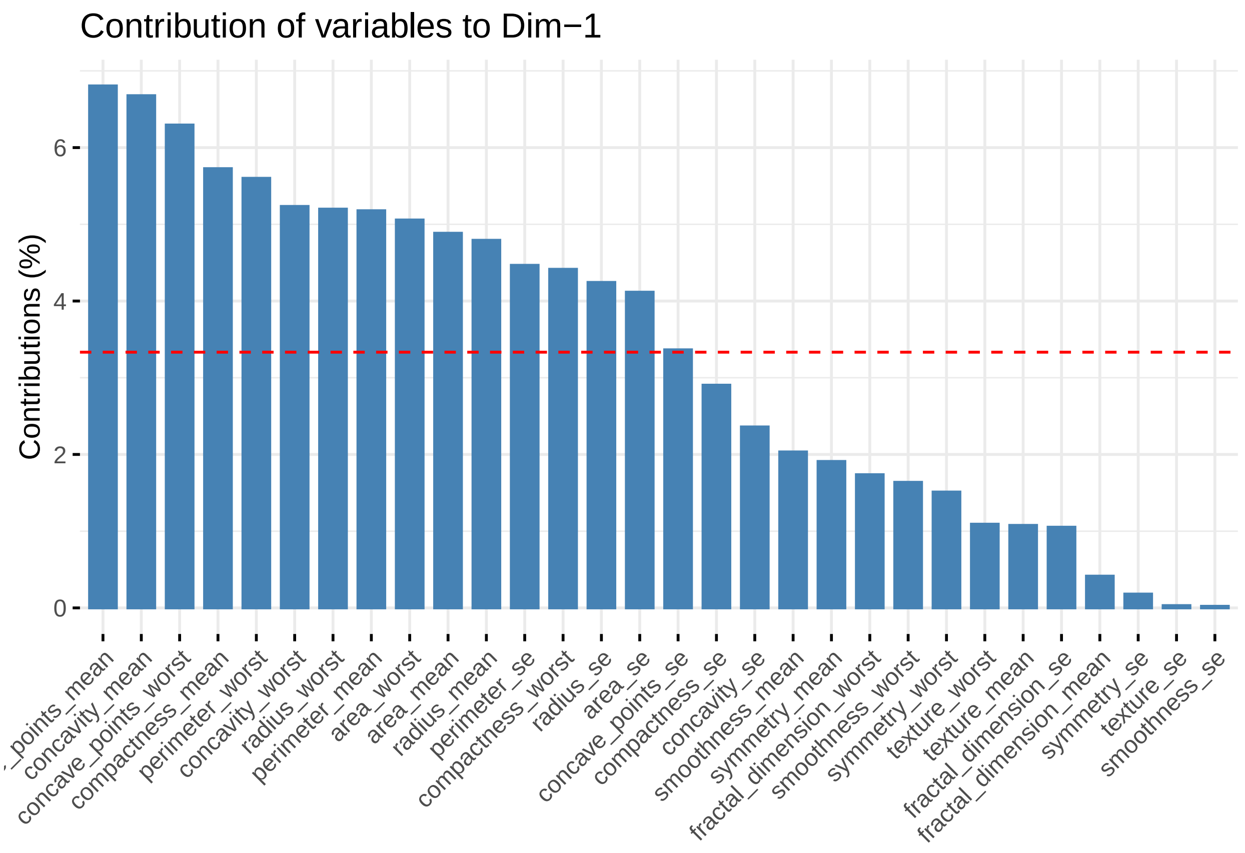
```
# biplot color coded by squared cosine
fviz_pca_var(pca_result, col.var = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07")
             )
```



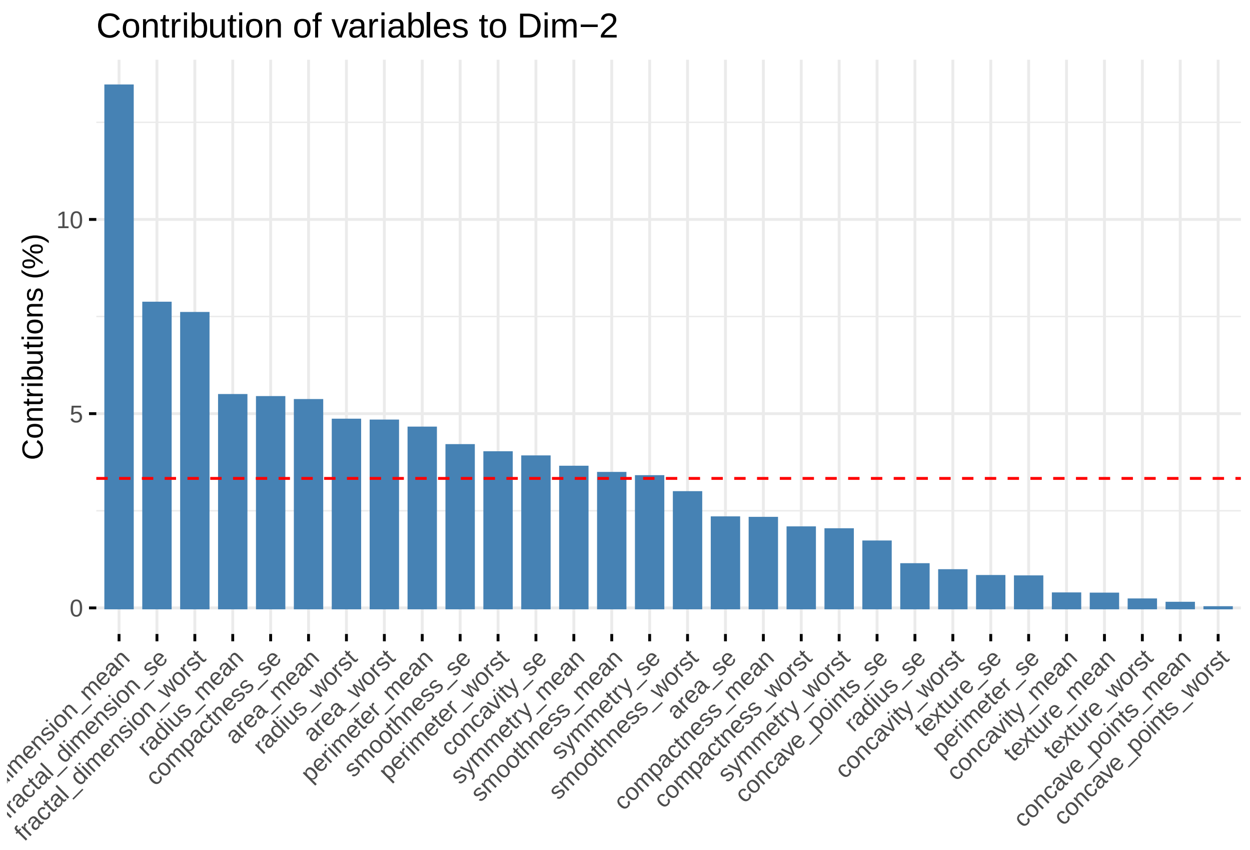
Contribution

We can also express the squared cosine as a percent of the overall variance explained by that particular PC. In this way, contribution is given by (var.cos2 * 100) / (total cos2 of the component) and essentially represents a re-scaling of cosine squared as a percent. We can reproduce the squared cosine plots above in terms of contribution.

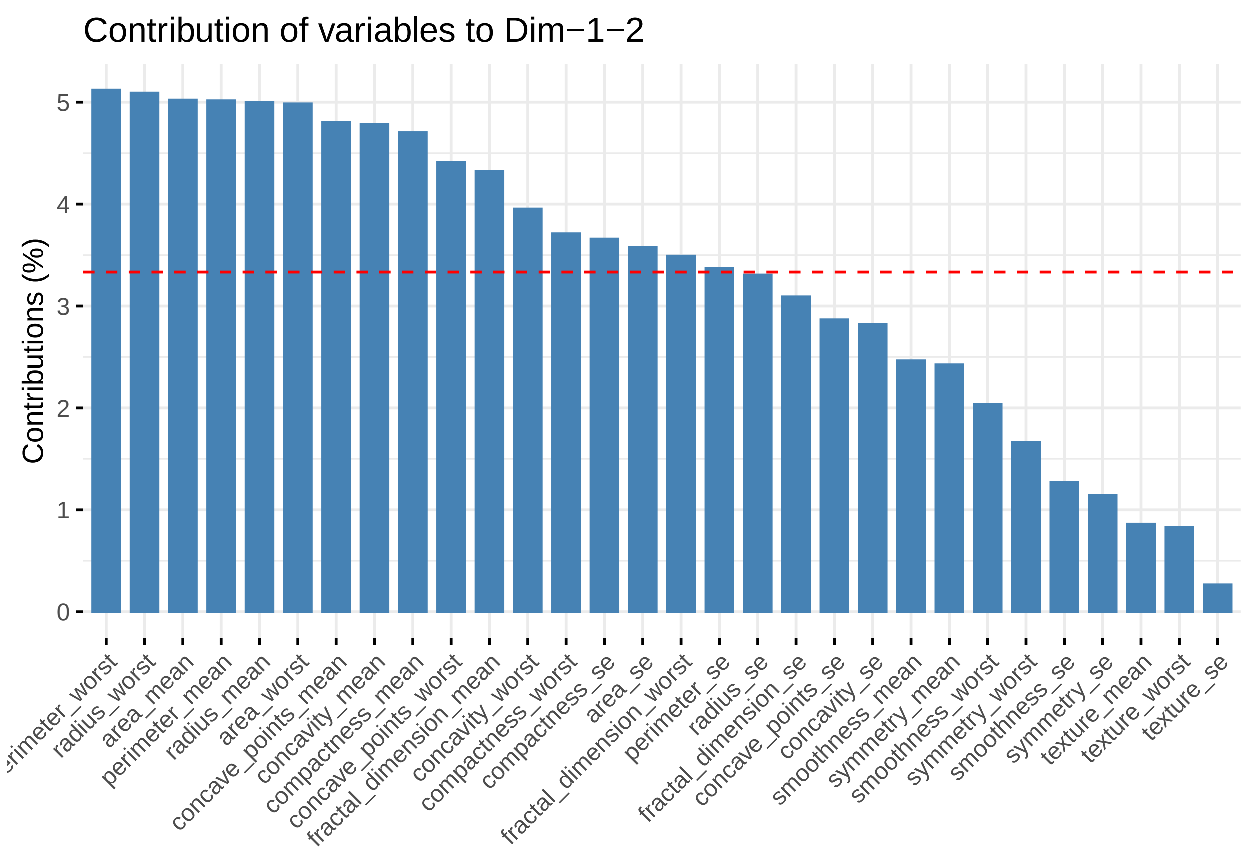
```r
# plot contribution across loadings for first PC
fviz_contrib(pca_result, choice = "var", axes = 1)
```



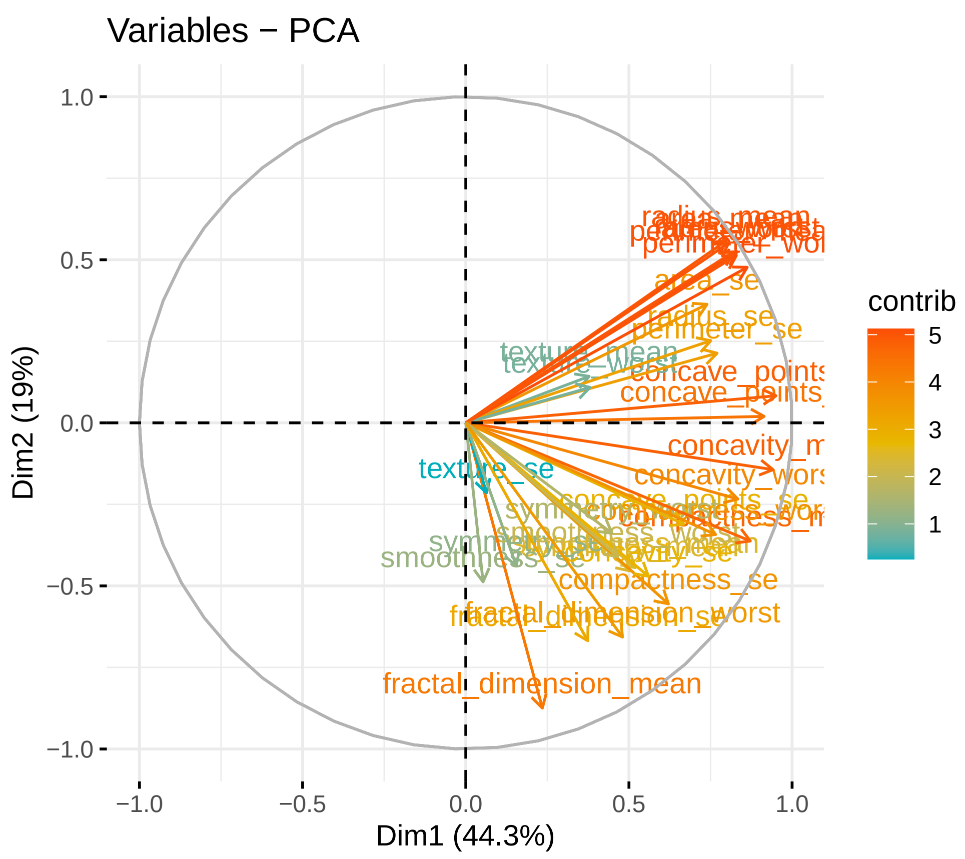
```r
# plot contribution across loadings for second PC
fviz_contrib(pca_result, choice = "var", axes = 2)
```



```
# plot contribution across loadings for both PCs
fviz_contrib(pca_result, choice = "var", axes = 1:2)
```



```r
# biplot color coded by contribution
fviz_pca_var(pca_result, col.var = "contrib",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07")
             )
```



Conclusion

Overall variations of perimeter, radius and area all contribute heavily to the the variation of data over the first 2 PCs. Concavity metrics also ranked highly in this regard. From the data description, the label 'worst' just refers to the largest measurement in a given image, so the columns called worst perimeter or the worst radius contain the maximum measurement for those images. This means that we may conclude that most of the variance in this data is captured by measurements of perimeter, which implies malignant cells probably appear larger and more irregular in shape than benign cells. Further statistical analyses would be needed to confirm whether or not benign cells differ from malignant cells in this way, but PCA has given us some insight into what features play a larger role in the variance in this dataset.

In this set, the issue of mulitcolinearity also arises because many of the columns in this set describe the same thing and are well correlated (perimeter, area, and radius are all a measure of size in terms of radius). If we were training models on the original features of this data it would be best to just pick one of these. This application of PCA also helped us build new uncorrelated features which we could use to train multiple regression models. From the cumulative variance explained, the first 7 PCs describe over 90% of the variance explained and would be a fair cut off.