HarvardX Data Science program capstone project

Human Activity and Postural Transisions (HAPT) recognition

Vladimir Pedchenko

11/24/2021

Contents

1 Introduction **2**

2 Dataset description **2**

2.1 Features description 5

3 Exploratory Data Analysis **7**

3.1 Features analysis 9

3.2 Principal Component Analisys 11

4 Literature **13**

1 Introduction

Human activities recognition is the necessary basis for the development of many applications such as:

- Health monitoring;
- Sport trackers;
- Context notifications/reminders, based on activity;
- Personal exercise advisers;
- VR/AR applications.

In the past, to track human activities was not possible without expensive hardware which had to be mounted on the body and connected by wires to data processing computer. Fortunately, it is not a problem anymore: most of the people always have smartphone or/and smartwatch with them. These devises have set of sensors, such as accelerometer and gyroscope which able to provide an information about device movements and position. These signals can be preprocessed and be used as input for machine learning algorithms to recognize an activity of the person who has the device in hands or pocket.

Goal of the project is to build a model, which can recognize human activities based on preprocessed smartphone sensors data.

2 Dataset description

In order to build, train and test the model HAPT dataset was used. It can be found at: http://archive.ics.uci.edu/ml/datasets/Smartphone-Based+Recognition+of+Human+Activities+and+Postural+Transitions

This dataset was build by the experiments were carried out with a group of 30 volunteers within an age bracket of 19-48 years. They performed a protocol of activities composed of six basic activities: three static postures (standing, sitting, lying) and three dynamic activities (walking, walking downstairs and walking upstairs). The experiment also included postural transitions that occurred between the static postures. These are: stand-to-sit, sit-to-stand, sit-to-lie, lie-to-sit, stand-to-lie, and lie-to-stand. All the participants were wearing a smartphone (Samsung Galaxy S II) on the waist during the experiment execution. Experimenters captured 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz using the embedded accelerometer and gyroscope of the device. The experiments were video-recorded to label the data manually.

The obtained dataset was randomly partitioned into two sets, where 70% of the volunteers was selected for generating the training data and 30% the test data.

The sensor signals (accelerometer and gyroscope) were pre-processed by applying noise filters and then sampled in fixed-width sliding windows of 2.56 sec and 50% overlap (128 readings/window). The sensor acceleration signal, which has gravitational and body motion components, was separated using a Butterworth low-pass filter into body acceleration and gravity. The gravitational force is assumed to have only low frequency components, therefore a filter with 0.3 Hz cutoff frequency was used. From each window, a vector of 561 features was obtained by calculating variables from the time and frequency domain.

The dataset archive includes the following files:

- 'README.txt'
- 'RawData/acc_expXX_userYY.txt': The raw triaxial acceleration signal for the experiment number XX and associated to the user number YY. Every row is one acceleration sample (three axis) captured at a frequency of 50Hz.
- 'RawData/gyro_expXX_userYY.txt': The raw triaxial angular speed signal for the experiment number XX and associated to the user number YY. Every row is one angular velocity sample (three axis) captured at a frequency of 50Hz.
- 'RawData/labels.txt': include all the activity labels available for the dataset (1 per row).
  - Column 1: experiment number ID,
  - Column 2: user number ID,
  - Column 3: activity number ID
  - Column 4: Label start point (in number of signal log samples (recorded at 50Hz))
  - Column 5: Label end point (in number of signal log samples)
- 'features_info.txt': Shows information about the variables used on the feature vector.
- 'features.txt': List of all features.
- 'activity_labels.txt': Links the activity ID with their activity name.
- 'Train/X_train.txt': Training set.
- 'Train/y_train.txt': Training labels.
- 'Test/X_test.txt': Test set.
- 'Test/y_test.txt': Test labels.
- 'Train/subject_id_train.txt': Each row identifies the subject who performed the activity for each window sample. Its range is from 1 to 30.
- 'Test/subject_id_test.txt': Each row identifies the subject who performed the activity for each window sample. Its range is from 1 to 30.

Because there are already preprocessed data in Training and Testing sets, raw data will not be used in this project.

Archive is downloaded and read by the following code:

```r
# If 'Data' folder is not exist, create it
if (!dir.exists("./data")) {
    dir.create("./data")
}

# Download file if it is not downloaded yet
if (!file.exists("./data//HAPT Data Set.zip")) {
    download.file("http://archive.ics.uci.edu/ml/machine-learning-databases/00341/HAPT%20Data%20Set.zip
        "./data//HAPT Data Set.zip")
}

# Read data from all files in the archive and assign to variables

# First read features labels, to be able to name columns in the features
# dataframe
features <- read.csv(unzip("./data//HAPT Data Set.zip", "features.txt"), header = FALSE)
features <- as.vector(features[, 1])

# Replace '-' to '_' from the features names, because it replaces by '.' when
# apply to column names
features <- str_replace_all(features, "-", "_")
# And remove spaces
features <- str_replace_all(features, " ", "")

# Read activity labels to replace activity number by activity label in the
# outcome vector
activity_labels <- read.csv(unzip("./data//HAPT Data Set.zip", "activity_labels.txt"),
    sep = " ", header = FALSE)
colnames(activity_labels) <- c("Activity_number", "Activity")
activity_labels <- activity_labels %>%
    select("Activity_number", "Activity")

# Unzip and read training data
x_train <- read.csv(unzip("./data//HAPT Data Set.zip", "Train/X_train.txt"), sep = " ",
    header = FALSE, col.names = features)
y_train <- read.csv(unzip("./data//HAPT Data Set.zip", "Train/y_train.txt"), sep = " ",
    header = FALSE, col.names = "Activity_number")

# Merge outcome vector and activity labels to have labels instead of numbers
y_train <- y_train %>%
    left_join(activity_labels, by = "Activity_number") %>%
    select(-"Activity_number")


# Unzip and read testing data (feature names vector is the same that for
# training data)
x_test <- read.csv(unzip("./data//HAPT Data Set.zip", "Test/X_test.txt"), sep = " ",
    header = FALSE, col.names = features)
y_test <- read.csv(unzip("./data//HAPT Data Set.zip", "Test/y_test.txt"), sep = " ",
    header = FALSE, col.names = "Activity_number")

# Merge outcome vector and activity labels to have labels instead of numbers
y_test <- y_test %>%
```

```
    left_join(activity_labels, by = "Activity_number") %>%
    select(-"Activity_number")

# Delete unzipped files and folders
unlink(c("Test", "Train", "activity_labels.txt", "features.txt"), recursive = T)

# Change Activity column type to factor
y_train <- y_train %>%
    mutate(Activity = factor(Activity))
y_test <- y_test %>%
    mutate(Activity = factor(Activity))


# df_validation = x_test + y_test will be considered as unknown data and will
# not be used till the end of the project as final validation.  Data analysis,
# model training and selection will be performed entirely on the train dataset
# (df = x_train + y_train)

# combine x_train and y_train to one dataframe for EDA

df <- cbind(x_train, y_train)
df_validation <- cbind(x_test, y_test)

rm(x_train, y_train, x_test, y_test)
```

It reads Training and Testing sets from files, apply feature names from features.txt to columns names. Then, activities encoded as numbers are converted to meaningful labels. Finally, train features and outcome data are combined to *df* dataset and test features and outcome data are combined to *df_validation* dataset.

df is the dataset which will be used for analysis and machine learning. *df_validation* is the validation hold-out dataset, which will not be used for any purposes until final model validation at the very end of the project.

Additional notes from Readme.txt:

- Features are normalized and bounded within [-1,1].
- Each feature vector is a row on the 'X' and 'y' files.
- The units used for the accelerations (total and body) are 'g's (gravity of earth -> 9.80665 m/seg2).
- The gyroscope units are rad/seg.
- A video of the experiment including an example of the 6 recorded activities with one of the participants can be seen in the following link: http://www.youtube.com/watch?v=XOEN9W05_4A

2.1 Features description

Dataset archive contains features_info.txt file, which provides important information about features, how raw data from sensors was preprocessed. In case of using different raw data, they can be preprocessed the same way.

The features selected for this database come from the accelerometer and gyroscope 3-axial raw signals *tAcc-XYZ* and *tGyro-XYZ*. These time domain signals (prefix '*t*' to denote time) were captured at a constant rate of 50 Hz. Then they were filtered using a median filter and a 3rd order low pass Butterworth filter with a

corner frequency of 20 Hz to remove noise. Similarly, the acceleration signal was then separated into body and gravity acceleration signals (*tBodyAcc-XYZ* and *tGravityAcc-XYZ*) using another low pass Butterworth filter with a corner frequency of 0.3 Hz.

Subsequently, the body linear acceleration and angular velocity were derived in time to obtain Jerk signals (*tBodyAccJerk-XYZ* and *tBodyGyroJerk-XYZ*). Also the magnitude of these three-dimensional signals were calculated using the Euclidean norm (*tBodyAccMag*, *tGravityAccMag*, *tBodyAccJerkMag*, *tBodyGyroMag*, *tBodyGyroJerkMag*).

Finally a Fast Fourier Transform (FFT) was applied to some of these signals producing *fBodyAcc-XYZ*, *fBodyAccJerk-XYZ*, *fBodyGyro-XYZ*, *fBodyAccJerkMag*, *fBodyGyroMag*, *fBodyGyroJerkMag*. (Note the '*f*' to indicate frequency domain signals).

These signals were used to estimate variables of the feature vector for each pattern:

'*-XYZ*' is used to denote 3-axial signals in the X, Y and Z directions.

- *tBodyAcc-XYZ*
- *tGravityAcc-XYZ*
- *tBodyAccJerk-XYZ*
- *tBodyGyro-XYZ*
- *tBodyGyroJerk-XYZ*
- *tBodyAccMag*
- *tGravityAccMag*
- *tBodyAccJerkMag*
- *tBodyGyroMag*
- *tBodyGyroJerkMag*
- *fBodyAcc-XYZ*
- *fBodyAccJerk-XYZ*
- *fBodyGyro-XYZ*
- *fBodyAccMag*
- *fBodyAccJerkMag*
- *fBodyGyroMag*
- *fBodyGyroJerkMag*

The set of variables that were estimated from these signals are:

- mean(): Mean value
- std(): Standard deviation
- mad(): Median absolute deviation
- max(): Largest value in array
- min(): Smallest value in array

- sma(): Signal magnitude area
- energy(): Energy measure. Sum of the squares divided by the number of values.
- iqr(): Interquartile range
- entropy(): Signal entropy
- arCoeff(): Autorregresion coefficients with Burg order equal to 4
- correlation(): correlation coefficient between two signals
- maxInds(): index of the frequency component with largest magnitude
- meanFreq(): Weighted average of the frequency components to obtain a mean frequency
- skewness(): skewness of the frequency domain signal
- kurtosis(): kurtosis of the frequency domain signal
- bandsEnergy(): Energy of a frequency interval within the 64 bins of the FFT of each window.
- angle(): Angle between to vectors.

Additional vectors obtained by averaging the signals in a signal window sample. These are used on the angle() variable:

- *gravityMean*
- *tBodyAccMean*
- *tBodyAccJerkMean*
- *tBodyGyroMean*
- *tBodyGyroJerkMean*

3 Exploratory Data Analysis

First, let's get some general information about dataset.

```
# dataset dimensions
dim(df)
```

```
## [1] 7767  562
```

```
# NAs in dataset
df %>%
    is.na() %>%
    sum()
```

```
## [1] 0
```

The dataset contains 7767 rows and 562 columns. There are 0 missing values in it. List of features names is too long to print it out completely, therefore we will look on head and tail of the dataset with just first three features and outcome column:

Table 1: First 10 rows of the dataset with three features and outcome

	tBodyAcc_Mean_1	tBodyAcc_Mean_2	tBodyAcc_Mean_3	Activity
1	0.0435797	-0.0059702	-0.0350543	STANDING
2	0.0394800	-0.0021313	-0.0290674	STANDING
3	0.0399778	-0.0051527	-0.0226507	STANDING
4	0.0397846	-0.0118088	-0.0289158	STANDING
5	0.0387581	-0.0022885	-0.0238629	STANDING
6	0.0389880	0.0041089	-0.0173403	STANDING
7	0.0398975	-0.0053243	-0.0204565	STANDING
8	0.0390823	-0.0160471	-0.0302413	STANDING
9	0.0390262	-0.0074100	-0.0273007	STANDING
10	0.0403539	0.0042448	-0.0179322	STANDING

Table 2: Last 10 rows of the dataset with three features and outcome

	tBodyAcc_Mean_1	tBodyAcc_Mean_2	tBodyAcc_Mean_3	Activity
7758	0.0385597	-0.0927131	0.0135742	WALKING_UPSTAIRS
7759	0.0458565	-0.0254181	-0.0417472	WALKING_UPSTAIRS
7760	0.0164071	-0.0218810	-0.0572198	WALKING_UPSTAIRS
7761	0.0110252	0.0767841	-0.0909770	WALKING_UPSTAIRS
7762	0.0231663	0.0130153	-0.0448918	WALKING_UPSTAIRS
7763	0.0480484	-0.0424452	-0.0658843	WALKING_UPSTAIRS
7764	0.0376386	0.0064304	-0.0443447	WALKING_UPSTAIRS
7765	0.0374509	-0.0027244	0.0210094	WALKING_UPSTAIRS
7766	0.0440110	-0.0045358	-0.0512422	WALKING_UPSTAIRS
7767	0.0689538	0.0018103	-0.0803234	WALKING_UPSTAIRS

Distribution of outcomes in the dataset:

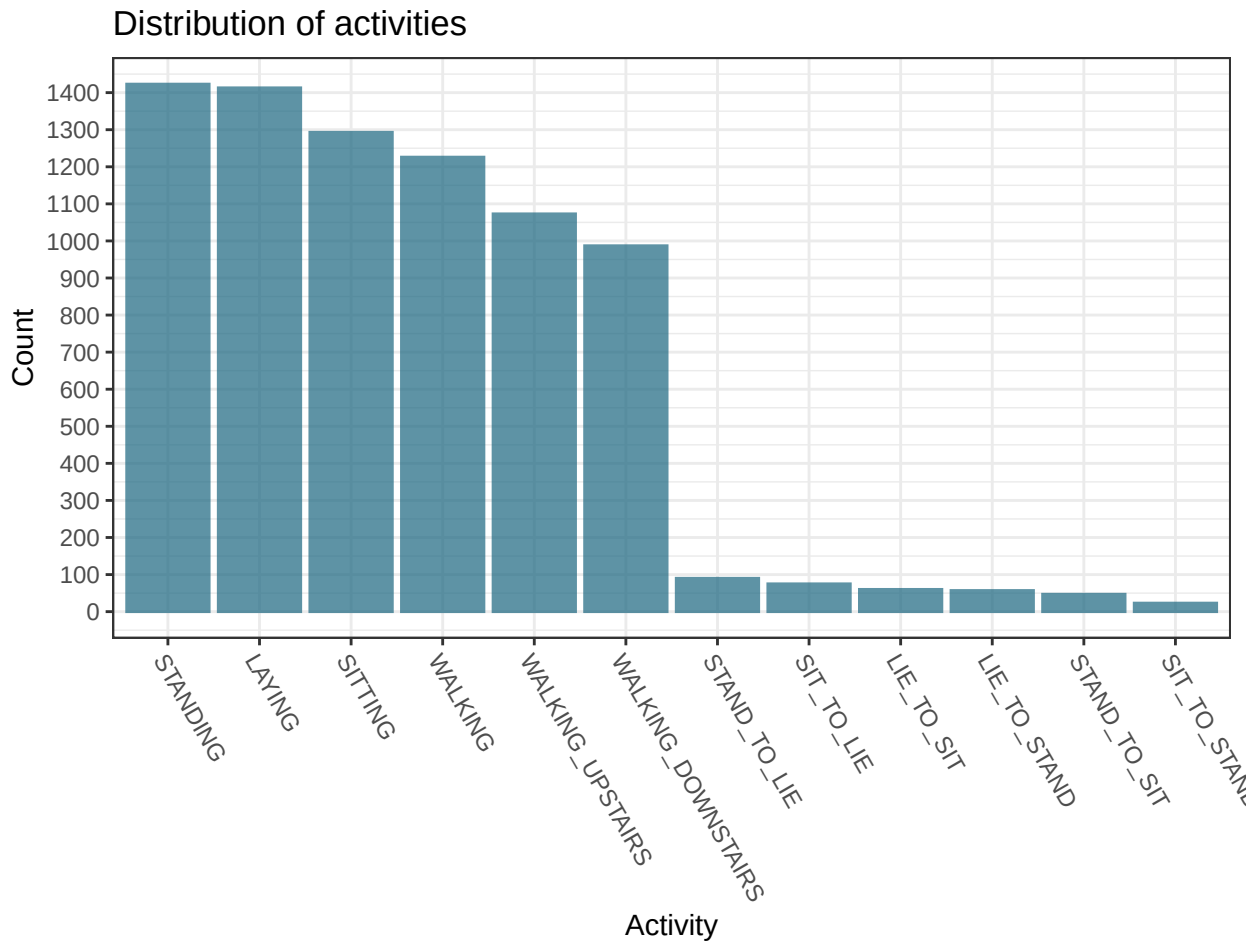

Figure 1: Distribution of activities

As we can see, outcomes are not equally represented in the dataset. The biggest amount of measures are classified as STANDING and there are 1423 of them. From other side, the least appearing activity SIT_TO_STAND have only 23 representations in the dataset.

The dataset is highly unbalanced, which must be taken in account during model building and evaluation. All postural transitions are minority classes, compare to other activities. It is expected that real distribution is the same: static activities takes more time compare to posture changes, which are short, momentary events.

3.1 Features analysis

The dataset contains 561 features. They are are normalized and bounded within [-1,1]. Visualization of features statistics:

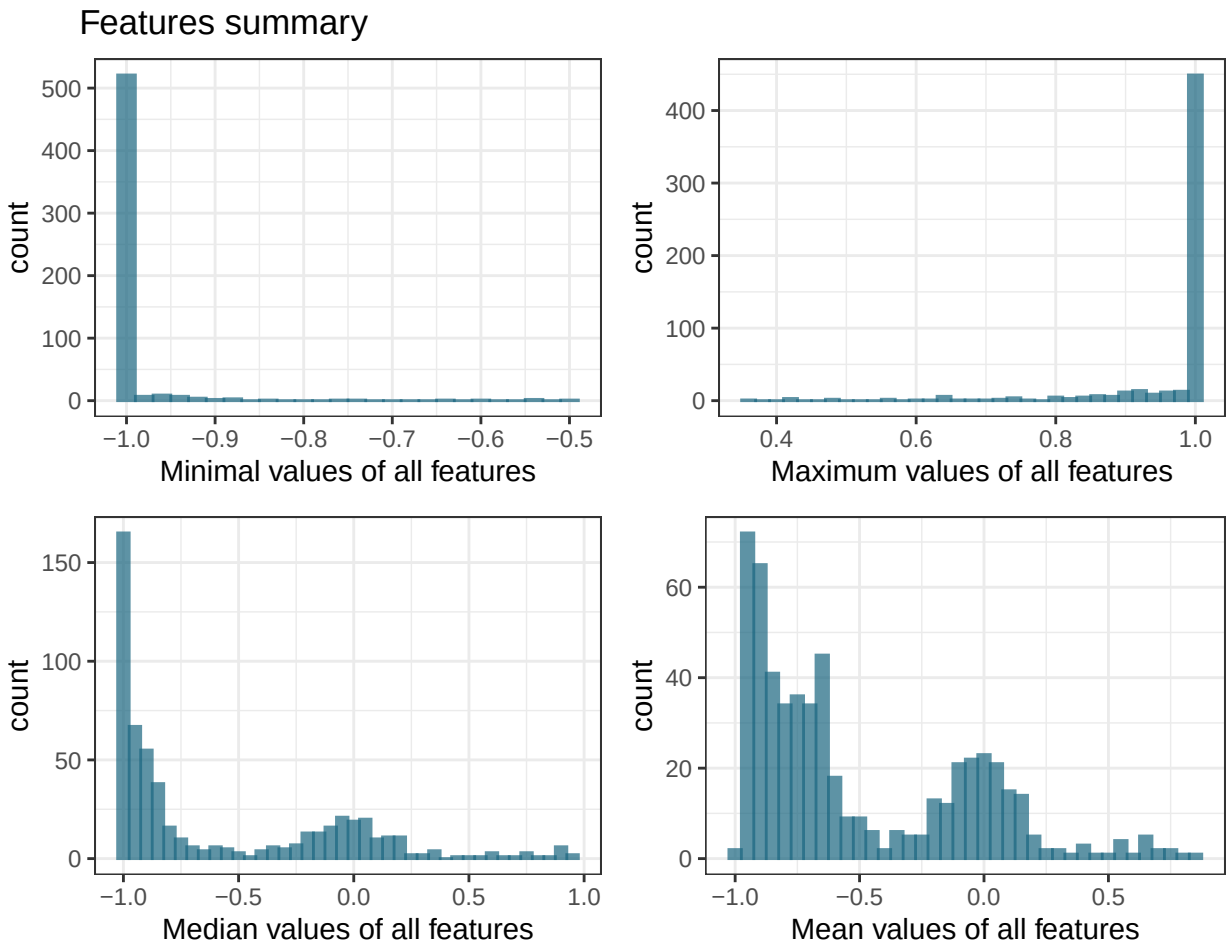

Figure 2: Features summary

Figure 2 confirms that vast majority of the features have minimal value as -1 and maximum value as 1. Also, we can see, that many features have very low median value (-1 - -0.9), which means that they have low variances. But looking to mean values trend, we see, that peak is shifted towards center a bit, which means that they have some variance. Remember Figure 1, which tells us, that some classes are vast minority, we can't call these features "not important" and drop them. It can be, that these features are important for minority classes, such as posture changes, detection. Let us look on five features with minimum mean value:

Table 3: Five features with lowest mean value

fBodyAccJerk_BandsEnergyOld_8
fBodyGyro_BandsEnergyOld_19
fBodyGyro_BandsEnergyOld_22
fBodyGyro_BandsEnergyOld_8
fBodyGyro_BandsEnergyOld_25

These features are energy measure of frequency domain signals (see chapter 2.1). Visualization of their distributions for different outcome classes are shown on figure 3:

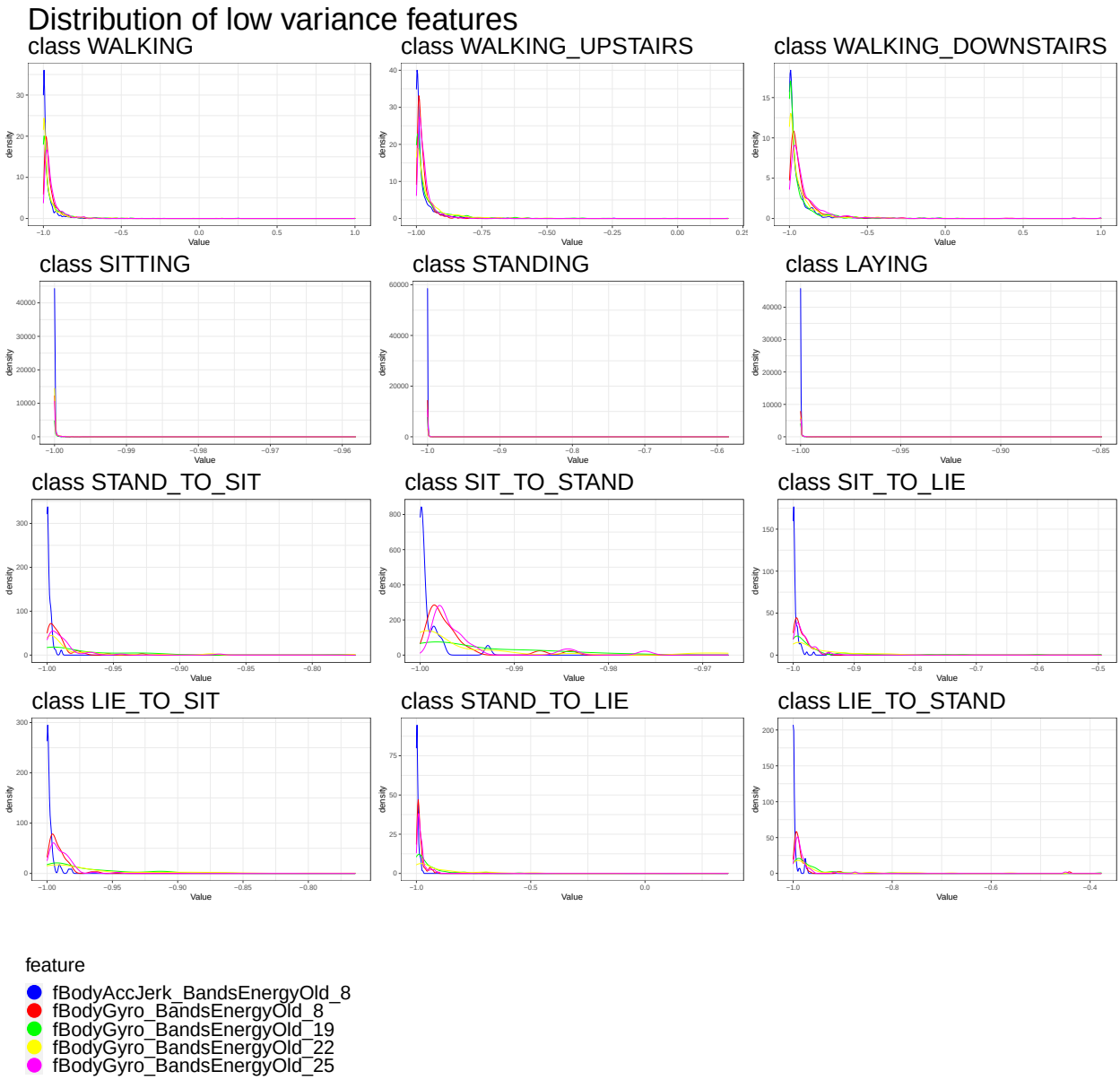

Figure 3: Distribution of low variance features

It is clearly seen that some of these features define posture transitions and specifically the minor class SIT_TO_STAND.

Unfortunately, it is not possible to effectively visualize all 561 features, therefore Principal Component Analysis will be performed.

3.2 Principal Component Analisys

Principal component analysis (PCA) is the process of computing the principal components and using them to perform a change of basis on the data, sometimes using only the first few principal components and ignoring the rest. PCA is defined as an orthogonal linear transformation that transforms the data to a new

coordinate system such that the greatest variance by some scalar projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on.

We can calculate principal components using function *prcomp* with argument scale = TRUE, which scales the variables to have unit variance before the analysis takes place:

```
pca <- prcomp(df[-ncol(df)], scale. = TRUE)
```

Visualizing the variance explained by each component help understand more about the data. It helps us identifying visually, how many principal components are needed to explain the data variation:

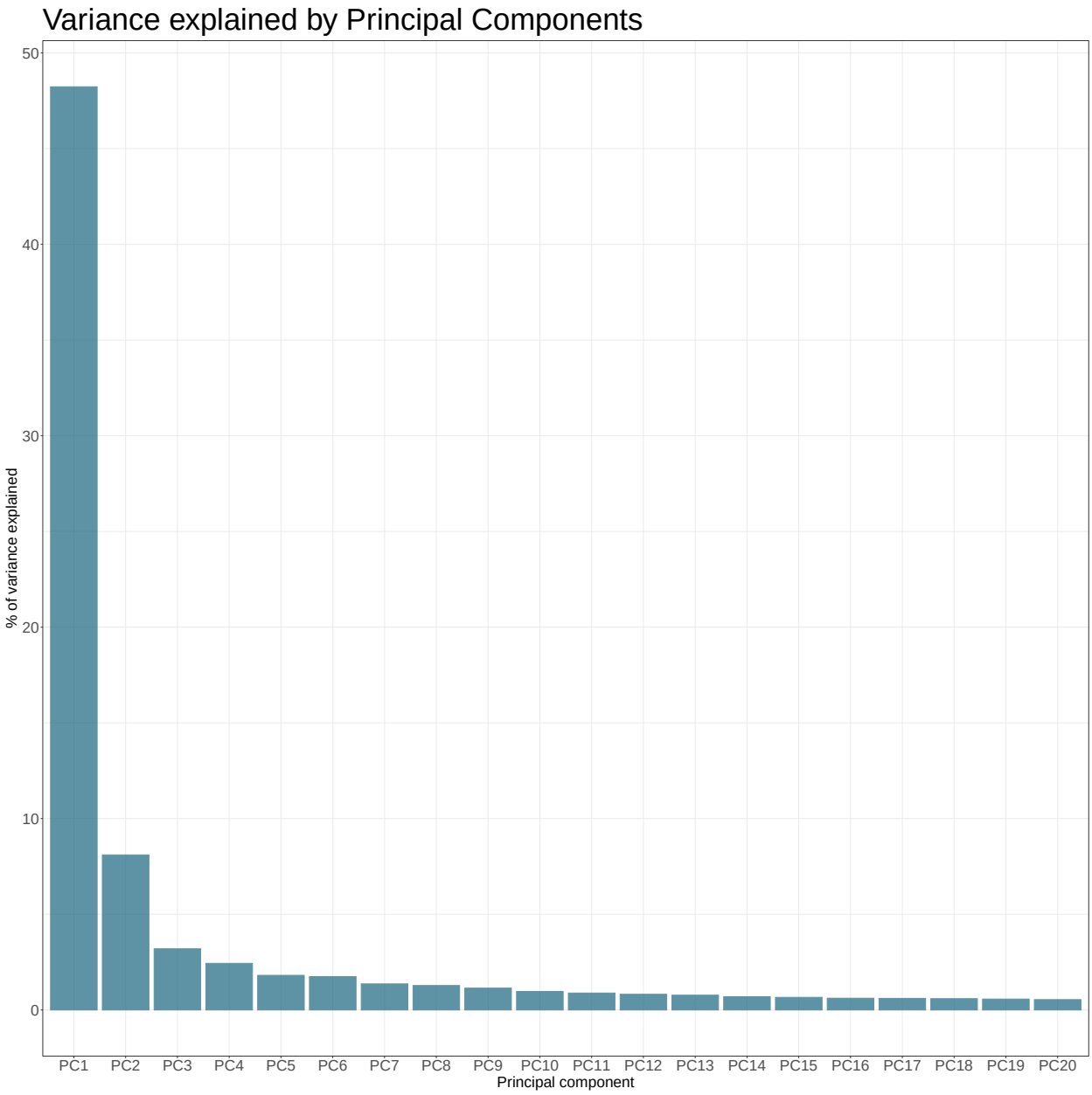

Figure 4: Variance explained by each of principal components

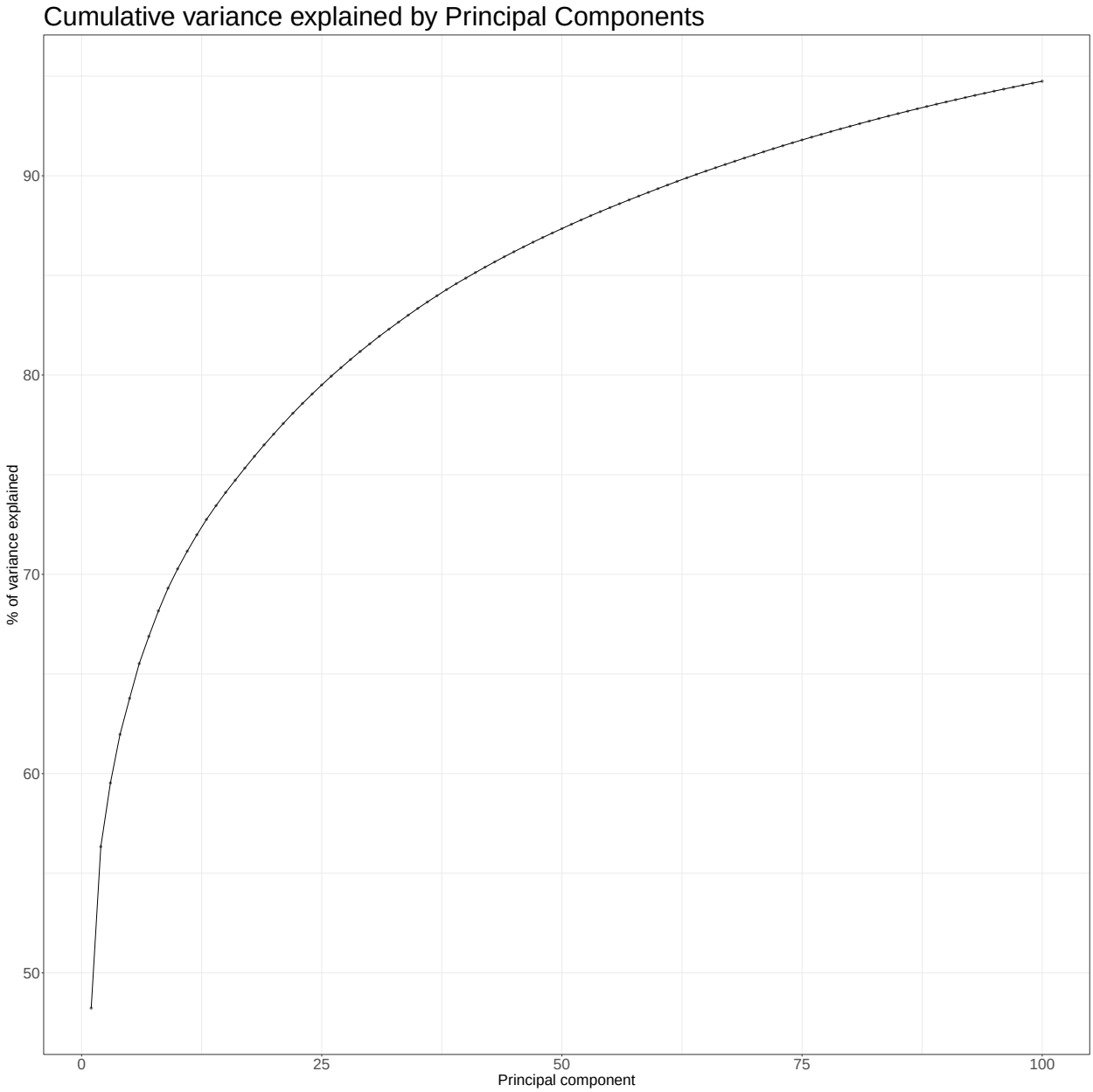

Figure 5: Cumulative variance explained by each of principal components

As we can see, almost half of variances can be explained by one principal component. But looking on the total variance explained by principal components, we can see, that to explain at least 95% of variance, we need more than 100 principal components. Conclusion is that our data space is indeed very multidimensional.

Let's which classes are easily distinguished by few first principal components:

4 Literature

1. Jorge-L. Reyes-Ortiz, Luca Oneto, Albert Samà, Xavier Parra, Davide Anguita. Transition-Aware Human Activity Recognition Using Smartphones. Neurocomputing. Springer 2015.

2. Rafael A. Irizarry, Introduction to Data Science
3. Bex T., Comprehensive Guide to Multiclass Classification Metrics
4. Max Kuhn, The caret Package
5. Jolliffe, I. T. (2002). Principal Component Analysis