Unsupervised Learning

The inferential and predictive models covered thus far can be categorized as **supervised learning** models. For each observation x_i in the data, there is an associated response y_i in a supervised learning setting, and the goal is to fit a model that relates y to one or more predictors to understand relationships or predict future values on the basis of the identified associations. However, in an **unsupervised learning** setting, no response y_i is associated with x_i. As a result, we cannot *supervise* the analysis and are limited to understanding how observations cluster or group together based on patterns across the available p attributes.

People analytics often involves the unique challenge of analyzing high-dimensional data with a large number of p attributes but relatively few n observations—a phenomenon often referred to as the *curse of dimensionality*. Given the sample size requirements covered in previous chapters, we ideally want n to be an order of magnitude larger than p to support statistical power and increase our chances of detecting meaningful patterns and population effects in sample data. Since people data sets are often wide and short, **dimension reduction** is important for reducing the dimensions to a limited subset that captures the majority of the information and optimizes the $n : p$ ratio.

Consider a case in which a colleague uses verbose rhetoric to convey a simple message that could be effectively communicated with fewer words. The superfluous language is unnecessary and does not provide additional information or value. This is analogous to dimension reduction in that we are interested in identifying a limited set of meaningful attributes and discarding redundant and unimportant information that does not contribute to the analysis objectives.

Dimensionality reduction techniques project data onto a lower dimensional subspace that retains the majority of the variance in the data points. If we take a picture of a group of colleagues during a team outing, for example, we would lose some 3D information by encoding the information into a 2D image. This 2D representation is a *subspace* of the 3D coordinates. While we would not know how far one person is from another in the 2D representation, we could see that people

C. Starbuck, *The Fundamentals of People Analytics*,
https://doi.org/10.1007/978-3-031-28674-2_14

in the back may appear smaller than people in the front. Therefore, the perspective in the 2D image would still capture *some* information about distance. The limited information loss in moving from three to two dimensions is likely acceptable, assuming the objective is to capture the memory of the team in a photograph.

Dimension reduction is particularly important in survey research because longer surveys are costly and may result in lower response rates due to the increased completion time requirements. Survey instrumentation with strong psychometric properties features highly correlated survey items for multidimensional constructs that are relatively uncorrelated with survey items used to measure other independent constructs. Intuitively, we know that highly correlated variables do not capture unique information, as one is often a sufficient proxy to capture most of the available signal in the larger number of features. As we have covered, models with highly correlated variables can create problems due to multicollinearity, and dimension reduction is an alternative approach to variable selection techniques such as the backward stepwise procedure covered in chapter "Linear Regression".

This chapter will cover dimension reduction fundamentals as well as technical implementations.

Factor Analysis

The development of survey instrumentation, whether a single item or a larger multidimensional scale, begins with a good theory. The theory provides conceptual support for the construct—the particular dimensions that characterize the construct, the antecedent variables which theoretically influence it, and the outcomes it will likely drive. With a strong theoretical framework, the researcher can begin proposing ways of *operationalizing* the conceptual scheme into a measurement approach.

There are clear measurement approaches for business metrics such as leads generated, new business growth, cNPS, and net revenue but in the social sciences, we often need indicators of **latent constructs** that are difficult—or impossible—to directly measure. If we want to understand the extent to which employees are engaged in their work, we need a comprehensive measure that captures facets of the theoretical frame. For example, vigor, absorption, and dedication are dimensions of Schaufeli et al.'s (2006) conception of work engagement which were operationalized in the Utrecht Work Engagement Scale (UWES).

Quantifying the energy levels one brings to work (vigor), the extent to which one feels time passes quickly while working (absorption), and the level of one's commitment to seeing tasks through to completion (dedication) is challenging since we cannot leverage transactional data or digital exhaust to directly quantify this as we can with operational business metrics. We need a comprehensive—yet parsimonious—set of survey items that function as indicators of the dimensions of the latent work engagement construct. Constructing a larger aggregate measure from the individual indicators, such as the average or sum of all survey items, enables us to reduce the number of variables and optimize the $n : p$ ratio in supervised learning settings.

Exploratory Factor Analysis (EFA)

Exploratory factor analysis (EFA) is a variable reduction technique by which factors are extracted from data—usually as part of the development process for new survey instruments.

A researcher may work with a panel of experts in a particular domain to develop an inventory of items that tap various aspects of the construct per the theoretical framework that underpins it. Based on how the items cluster together, the empirical data will be reconciled against the theoretical conception to define dimensions of the measure. Within a cluster of highly correlated items for a particular dimension, the researcher needs to decide which items are essential and which are redundant and eligible for removal. Aside from the principal clusters (or factors), remaining items also need to be evaluated for their relevance and support for the underlying theory. If items are believed to be members of the theoretical dimensions but do not cluster together with other similar items, it may be indicative of poorly written items that have different interpretations among survey takers. EFA is the empirical process that supports these objectives.

To illustrate the steps for EFA, we will leverage the `survey_responses` data.

```r
# Load library
library(peopleanalytics)

# Load data
data("survey_responses")

# Store data in df with curtailed name
survey_dat <- survey_responses

# Show dimensions of survey data
dim(survey_dat)
```

```
## [1] 400  12
```

EFA is implemented via a three-step procedure:

1. Assess the factorability of the data.
2. Extract the factors.
3. Rotate and interpret the factors.

Step 1: Factorability Assessment
With respect to factorability, there needs to be some correlation among variables in order for a dimension reduction technique to collapse variables into linear combinations that capture a large portion of the variance in the data. The data feature sufficient factorability if we achieve a **Kaiser–Meyer–Olkin (KMO)** statistic of at least 0.60 (Kaiser, 1974) and **Bartlett's Test of Sphericity** reaches statistical significance (Bartlett, 1954). The KMO statistic estimates the proportion of variance that may be common variance; the lower the proportion, the greater the factorability. Bartlett's test essentially measures the degree of redundancy in the data, where the null hypothesis states that the variables are orthogonal (uncorrelated); rejecting this null hypothesis indicates that there is sufficient correlation for dimension reduction.

The `KMO()` and `cortest.bartlett()` functions from the `psych` library can be used for the KMO statistic and Bartlett's test, respectively:

```
# Load library
library(psych)

# Kaiser-Meyer-Olkin (KMO) statistic
psych::KMO(survey_dat)
```

```
## Kaiser-Meyer-Olkin factor adequacy
## Call: psych::KMO(r = survey_dat)
## Overall MSA =  0.9
## MSA for each item =
##  belong  effort incl  eng_1  eng_2  eng_3  happ psafety  ret_1  ret_2
##    0.94    0.86 0.86  0.86   0.89   0.89   0.92    0.90   0.91   0.89
##   ret_3  ldrshp
##    0.90    0.93
```

```
# Bartlett's Test of Sphericity
psych::cortest.bartlett(cor(survey_dat), nrow(survey_dat))
```

```
## $chisq
## [1] 2933.161
##
## $p.value
## [1] 0
##
## $df
## [1] 66
```

Data satisfy the factorability requirements since $KMO = 0.90$ (`Overall MSA`) and Bartlett's test is significant at the $p < 0.001$ level.

Step 2: Factor Extraction
For the second step, we will visually inspect a **scree plot** and determine how many factors are necessary to explain most of the variance in the data. A scree plot is

a line plot that helps visualize the portion of the total variance explained by each factor using **eigenvalues**. While the linear algebraic underpinnings are out of scope for this book, it is important to understand that **eigenvectors** are vectors of a linear transformation which have corresponding eigenvalues λ that represent factors by which the vectors are scaled. As a general rule, factors with $\lambda \geq 1$ are extracted when running a factor analysis.

The `scree()` function from the `psych` library can be used to generate a scree plot:

```
# Produce scree plot
psych::scree(survey_dat, pc = FALSE)
```

Based on Fig. 1, factors 1 and 2 appear to provide relatively outsized information gain as $\lambda > 1$ for both.

You may notice the `pc = FALSE` argument in the `scree()` function call. This relates to **principal components analysis (PCA)**, which is an alternative method of dimension reduction. Principal components are new independent variables that represent linear transformations of scaled $((x - \bar{x})/s)$ versions of the observed variables. While we will focus on factor analysis and PCA in this section, which are most common in the social sciences, there are additional dimension reduction techniques one could explore (e.g., **parallel analysis**).

Though there are similarities between factor analysis and PCA, the mathematics are fundamentally different. PCA approaches dimension reduction by creating one

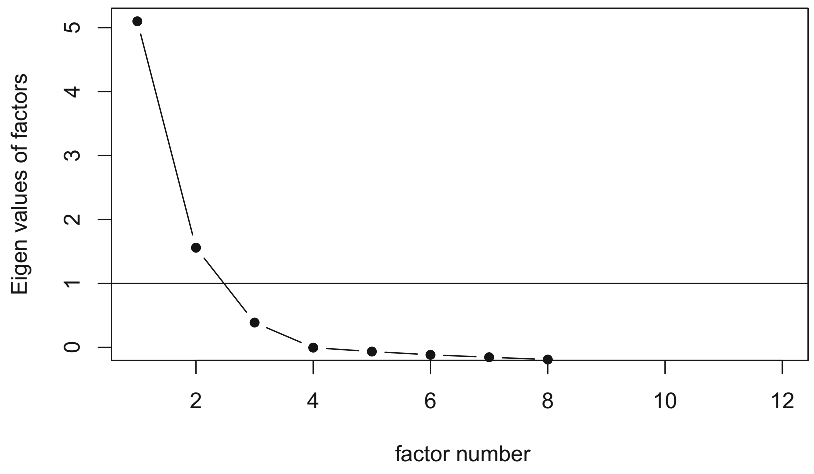

Fig. 1 Scree plot showing eigenvalues by factor relative to the extraction threshold (horizontal line)

or more index variables (linear combinations of original variables) from a larger set of measured variables; these new index variables are referred to as components. On the other hand, factor analysis can be viewed as a set of regression equations with weighted relationships that represent the measurement of a latent variable. Most variables are latent in social psychology contexts since we cannot directly measure constructs like psychological safety or belonging.

To illustrate how to extract principal components, we can use base R's `prcomp()` function:

```r
# Load library
library(ggplot2)

# Perform PCA
pca <- prcomp(survey_dat, scale = TRUE)

# Calculate explained variance for each principal component
pca_var = (pca$sdev^2 / sum(pca$sdev^2)) * 100

# Create scree plot
ggplot2::qplot(1:length(pca_var), pca_var) +
ggplot2::geom_line() +
ggplot2::scale_x_continuous(breaks = 1:length(pca_var)) +
ggplot2::labs(x = "Principal Component", y = "Variance
↪  Explained (%)") +
ggplot2::theme_bw()
```

Note that while we can theoretically have as many factors as we have variables ($p = 12$), this defeats the purpose of dimension reduction—whether PCA or factor analysis. The objective of dimension reduction is to *reduce* the number of factors (or components) to a subset that captures the majority of the information in the data.

As shown in Fig. 2, principal components are sorted in descending order according to the percent of total variance they explain. The first principal component alone explains nearly half of the total variance in the data. We are looking for the *elbow* to ascertain the inflection point at which explained variance plateaus. It is clear that the slope of the line begins to flatten beyond the third principal component, indicating that components 4–12 provide relatively little information. Put differently, we could extract only the first three components without sacrificing much information and gain the benefit of fewer, more meaningful variables.

As an aside, in a supervised learning context, we could insert these principal components as predictors in a regression model in lieu of a larger number of original variables. This is known as **principal components regression (PCR)**. However, given the importance of explaining models in a people analytics setting, PCR will not be covered since inserting index variables as predictors in the model compromises interpretability.

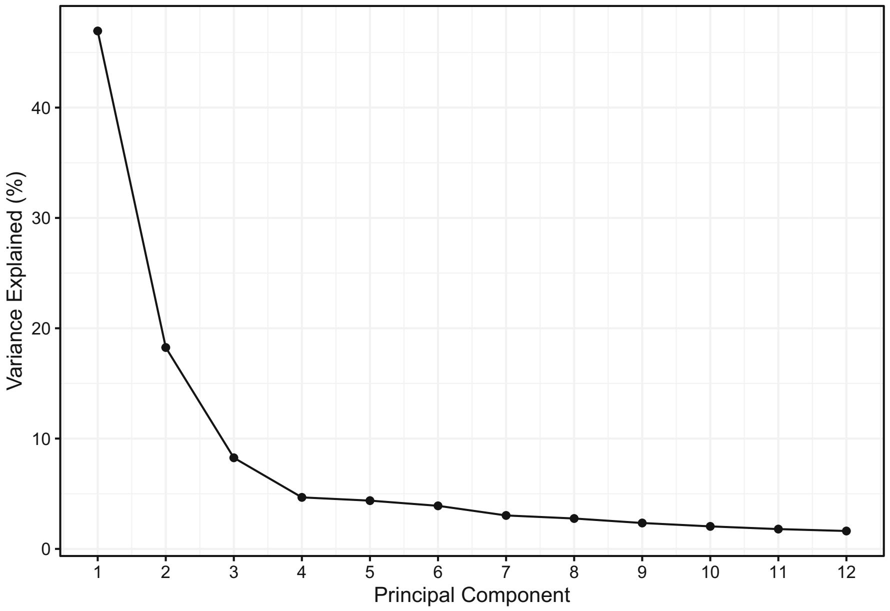

Fig. 2 Plot showing the percent of total variance explained by 12 principal components

Step 3: Factor Rotation and Interpretation

For the third step, we will use an Oblimin method to rotate the factor matrix. The Oblimin rotation is an oblique—rather than orthogonal—rotation and is selected here since it is best suited when underlying dimensions are assumed to be correlated (Hair et al., 2006).

The `fa()` (factor analysis) function from the `psych` package can be used for the implementation in R. Based on the scree plot, we will specify three factors for this analysis. Note that the Oblimin rotation is the default for factor analysis, while a varimax (orthogonal) rotation is the default for PCA. Many other rotations can be implemented based on the nature of data and *n*-count.

```
# Principal axis factoring using 3 factors and oblimin
↪  rotation
efa.fit <- psych::fa(survey_dat, nfactors = 3, rotate =
↪  'oblimin')

# Display factor loadings
efa.fit$loadings
```

```
##
## Loadings:
##          MR1    MR2    MR3
## belong   0.283          0.456
## effort  -0.114  0.869
## incl                    0.747
## eng_1           0.886
## eng_2    0.172  0.782
## eng_3           0.799
## happ     0.558          0.355
## psafety                 0.609
## ret_1    0.791
## ret_2    0.922         -0.111
## ret_3    0.822
## ldrshp   0.556          0.276
##
##                  MR1   MR2   MR3
## SS loadings    2.906 2.817 1.363
## Proportion Var 0.242 0.235 0.114
## Cumulative Var 0.242 0.477 0.590
```

The sum of squared loadings (`SS loadings`) represents the eigenvalues for each factor. It is also helpful to review the percent of total variance explained by each factor (`Proportion Var`) along with the cumulative percent of total variance (`Cumulative Var`). We can see that the three factors have $\lambda \geq 1$, which together explain 59% of the total variance in the data.

By reviewing the factor loadings, we gain an understanding of which variables are part of each factor (i.e., highly correlated variables which cluster together). Factor loadings represent the correlation of each item with the respective factor. While there is not a consensus on thresholds, a general rule of thumb is that *absolute* factor loadings should be at least 0.5. Items with lower factor loadings should be removed from the measurement model.

For the first factor `MR1`, the three retention items cluster together with happiness and leadership. This indicates that happier employees who have more favorable perceptions of leadership are less likely to leave the organization.

Loadings for the second factor `MR2` indicate that the three engagement items cluster together with discretionary effort. This makes intuitive sense, as we would expect highly engaged employees to contribute higher levels of effort toward their work.

Loadings for the third factor `MR3` show that belonging, inclusion, and psychological safety cluster together. In other words, employees who feel a stronger sense of belonging and perceive the environment to be more inclusive tend to experience a more favorable climate with respect to psychological safety.

We can visualize this information using the `fa.diagram()` function from the `psych` library (Fig. 3):

```
psych::fa.diagram(efa.fit)
```

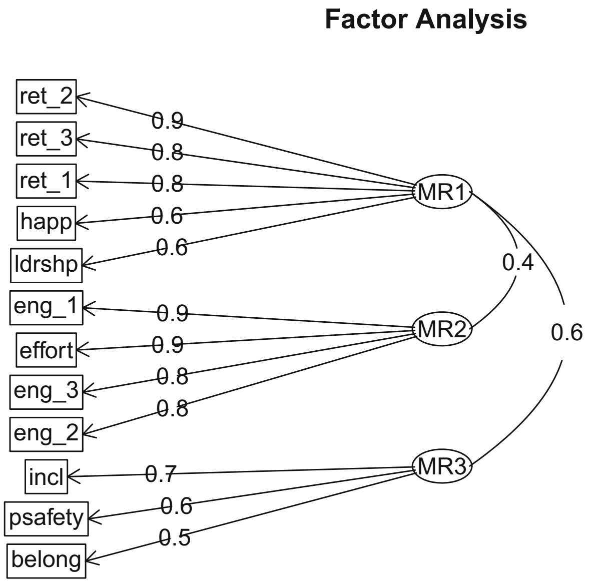

Fig. 3 Diagram showing factor loadings (correlations) for each item with the respective factor

Confirmatory Factor Analysis (CFA)

Confirmatory factor analysis (CFA) is used to test how well data align with a theoretical factor structure.

We expect items associated with a given construct to be highly correlated with one another but relatively uncorrelated with items associated with independent constructs. Consider engagement and retention, which are two independent—yet likely correlated—constructs. If multiple items are needed to measure the theoretical dimensions of both engagement and retention, we would expect the engagement items to be more highly correlated with one another than with the retention items. Theory may suggest that retention likelihood increases as engagement increases, but there are many other factors which also influence one's decision to leave an organization beyond engagement, so we would not expect changes in engagement levels to *always* be associated with a commensurate change in retention.

We can illustrate using our `survey_responses` data, which contains three items for both engagement and retention. Figure 4 shows pairwise relationships between the items. As expected, `eng_1`, `eng_2`, and `eng_3` have stronger correlations with one another ($r \geq 0.70$) than with `ret_1`, `ret_2`, or `ret_3` ($r \leq 0.52$).

CFA enables us to move beyond inter-item correlations to quantify the extent to which latent variables in our data fit an expected theoretical model. We can leverage the `lavaan` package in R to perform CFA, which is implemented via a three-step procedure:

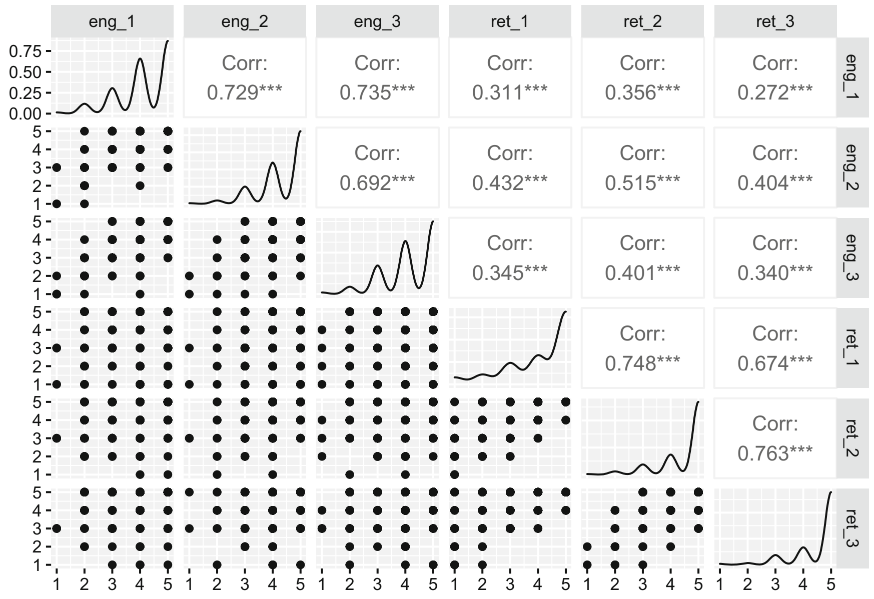

Fig. 4 Bivariate correlations and relationship visualizations for engagement and retention survey items

1. Define the model.
2. Fit the model.
3. Interpret the output.

Step 1: Define the Model
Step 1 is defining the model within a string per the syntax required by `lavaan`:

```
# Load library
library(lavaan)

# Model specification; each line represents a separate latent
↪  factor
model <- paste('engagement =~ eng_1 + eng_2 + eng_3
                retention =~ ret_1 + ret_2 + ret_3')
```

Step 2: Fit the Model
Step 2 is fitting the model to the data using the `cfa()` function from the `lavaan` package:

```
# Fit the model
cfa.fit <- lavaan::cfa(model, data = survey_dat)
```

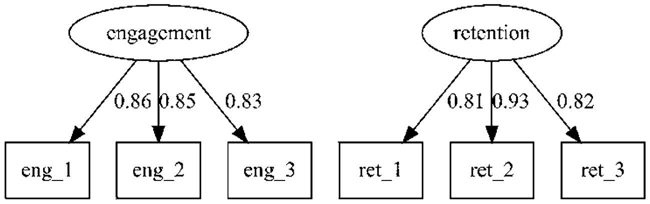

Fig. 5 Path diagram showing survey items as indicators of latent engagement and retention factors

We can also create what is known as a **path diagram** to assist with understanding the CFA model. A path diagram is a symbolic visualization of the measurement model, with circles depicting latent variables (factors), rectangles representing observed indicators (survey items), and arrows indicating paths (relationships) between variables. The measurement model (CFA) together with the structural (path) model is known as **structural equation modeling (SEM)**; CFA is a subset of the SEM umbrella.

The `lavaanPlot()` package can be used to create and visualize path diagrams in R:

```
# Load library
library(lavaanPlot)

# Visualize path diagram
lavaanPlot::lavaanPlot(model = cfa.fit, coefs = TRUE, stand =
↪  TRUE)
```

Factor loadings are shown for each indicator of the latent variable in Fig. 5. All are well above the *absolute* threshold of 0.5.

Step 3: Interpret the Model

Step 3 is interpreting the output of the fitted model:

```
cfa.fit <- lavaan::cfa(model, data = survey_dat)

# Summarize the model
summary(cfa.fit, fit.measures = TRUE)
```

```
## lavaan 0.6-12 ended normally after 25 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                        13
```

```
##
##   Number of observations                           400
##
## Model Test User Model:
##
##   Test statistic                                35.477
##   Degrees of freedom                                 8
##   P-value (Chi-square)                           0.000
##
## Model Test Baseline Model:
##
##   Test statistic                              1495.174
##   Degrees of freedom                                15
##   P-value                                        0.000
##
## User Model versus Baseline Model:
##
##   Comparative Fit Index (CFI)                    0.981
##   Tucker-Lewis Index (TLI)                       0.965
##
## Loglikelihood and Information Criteria:
##
##   Loglikelihood user model (H0)              -2456.223
##   Loglikelihood unrestricted model (H1)      -2438.484
##
##   Akaike (AIC)                                4938.446
##   Bayesian (BIC)                              4990.335
##   Sample-size adjusted Bayesian (BIC)         4949.085
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.093
##   90 Percent confidence interval - lower         0.063
##   90 Percent confidence interval - upper         0.125
##   P-value RMSEA <= 0.05                          0.011
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.040
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
```

```
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   engagement =~
##     eng_1             1.000
##     eng_2             0.880    0.044   19.827    0.000
##     eng_3             0.963    0.050   19.355    0.000
##   retention =~
##     ret_1             1.000
##     ret_2             0.799    0.039   20.384    0.000
##     ret_3             0.699    0.038   18.562    0.000
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   engagement ~~
##     retention         0.406    0.051    7.932    0.000
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .eng_1             0.235    0.027    8.839    0.000
##    .eng_2             0.188    0.021    9.013    0.000
##    .eng_3             0.265    0.027    9.820    0.000
##    .ret_1             0.482    0.044   10.952    0.000
##    .ret_2             0.095    0.018    5.145    0.000
##    .ret_3             0.215    0.020   10.571    0.000
##     engagement        0.649    0.064   10.188    0.000
##     retention         0.917    0.097    9.458    0.000
```

The `lavaan` package provides many fit measures, but we will focus only on the most common for evaluating how well the data fit the measurement model.

- Model Chi-Square (χ^2): Tests whether the covariance matrix derived from the model represents the population covariance (Test Statistic under the Model Test User Model section of the `lavaan` output)
- Comparative Fit Index (CFI): Values range from 0 to 1, with $CFI > 0.95$ indicating good fit
- Tucker Lewis Index (TLI): Values range from 0 to 1, with $TLI > 0.95$ indicating good fit
- Root Mean Square Error of Approximation (RMSEA): Values of 0.01, 0.05, and 0.08 indicate excellent, good, and mediocre fit, respectively (though some texts suggest 0.10 is an adequate threshold for mediocre fit)
- Standardized Root Mean Square Residual (SRMR): Square root of the difference between residuals of the sample covariance matrix and the hypothesized model, with $SRMR < 0.08$ indicating good fit

Given data sets are often small in people analytics, it is important to note that RMSEA often exceeds thresholds with small df and n—even when the model is correctly specified (Kenny et al., 2015). Therefore, it is important to index more on fit indices such as CFI and LTI in determining how well the data fit the measurement model.

The χ^2 statistic is sometimes referred to as a "badness of fit" measure since rejecting the null hypothesis ($p < 0.05$) indicates a lack of fit. Though χ^2 is significant ($p < 0.001$), both CFI (0.98) and TLI (0.97) are above the 0.95 threshold for good fit. In addition, $SRMR = 0.04$ is beneath the threshold of 0.08 and $RMSEA = 0.09$ is between the mediocre fit threshold range of 0.08 and 0.10. Therefore, the indicators (survey items) in these data adequately fit the two latent constructs defined by this measurement model.

For more extensive coverage of SEM, Kline (2005) is an excellent resource.

Clustering

Clustering is an ML technique that groups observations into clusters which have similar characteristics but different characteristics relative to the observations in other clusters. Clustering is similar to factor analysis in that it is also unsupervised since there is not a response variable, but it differs in that it does not seek to find a low-dimensional representation of observations that capture a large portion of variance in the data; clustering aims to find homogeneous subgroups among observations.

Clustering is common in marketing in which it is implemented to create customer segments with shared characteristics. By grouping customers based on attributes such as income, household size, occupation, and geography, companies can tailor marketing campaigns to each segment based on what is most likely to appeal to the unique needs of each.

In people analytics, clustering has important applications as well. For example, clustering can be implemented to define personas based on unique talent development needs (e.g., early tech career, newly promoted people leaders) or attrition risk (e.g., rising stars with hot skills, low performers in high churn roles, high performers in specialized roles). Grouping employees based on relative attrition risk levels is often a more viable path for action planning, as there are important legal and privacy considerations when applying predictive model scores at the individual level of analysis.

This section will cover two popular clustering techniques: *k-means* and *hierarchical* clustering.

K-Means Clustering

K-means clustering is a simple approach to grouping observations into K distinct clusters. K-means clustering is implemented via a four-step process:

1. Define K.
2. Randomly assign observations to one of K clusters.
3. For each of the K clusters, compute the cluster centroid.
4. Assign each observation to the cluster with the closet centroid (middle).

To assign observations to the cluster with the nearest centroid, a distance metric needs to be selected in order to measure the distance between each observation and cluster centroids. While calculating the distance between observations in two dimensions is simple, distance in higher dimensional space is more challenging. We will focus on the most common distance metric, **Euclidean distance**, though there are many others (e.g., Manhattan, Jaccard, Minkowski, Cosine). The Euclidean distance between two data points is the straight line distance based on the observations' coordinates using the **Pythagorean theorem**:

$$a^2 + b^2 = c^2,$$

where a and b are sides of a triangle that intersect to form a right angle, and c is the hypotenuse (the side opposite the right angle).

Let us implement K-means clustering using numeric variables in the `employees` data. Since the scale of variables matters when comparing distances between observations and cluster centers, we will first scale the variables to have $\bar{x} = 0$ and $s = 1$ in support of a consistent, apples-to-apples comparison:

```r
# Filter employee data to numeric variables
idx <- which(sapply(employees, is.numeric)) # store indices of
↪  numeric variables
employees <- employees[, idx] # filter df using indices

# Drop unimportant and sparsely populated sales variables
employees <- subset(employees, select = -c(employee_id,
↪  standard_hrs, ytd_leads, ytd_sales))

# Center and scale data
employees_trans <- scale(employees, center = TRUE, scale =
↪  TRUE)
```

Next, we need to define K. One way to determine the optimal number of clusters is to leverage the `fviz_nbclust()` function from the `factoextra` library to visualize the sum of squared differences between observations and cluster centers against the range of clusters:

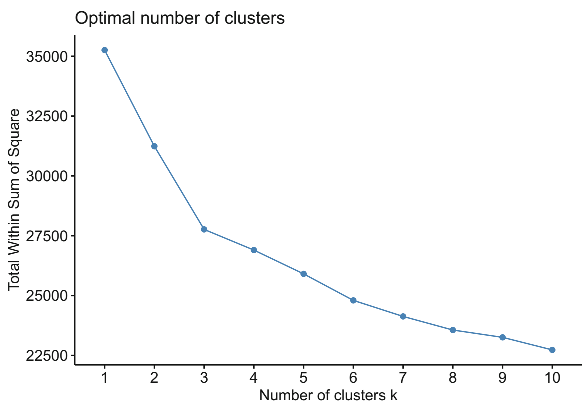

Fig. 6 Total within sum of squares (WSS) across cluster count

```
# Load library
library(factoextra)

# Determine optimal number of clusters
factoextra::fviz_nbclust(employees_trans, kmeans, method =
↪   "wss")
```

When interpreting Fig. 6, we are looking for the *elbow* which marks the inflection point at which the sum of squares begins to level off. The goal is to achieve the fewest number of clusters, optimizing for subgroups that are distinctly different *between* and highly similar *within*. The elbow indicates the optimal number of clusters, as additional clusters beyond the elbow do not offer a meaningful improvement in achieving homogeneous subgroups of the observations.

There is a discernible elbow at three clusters in Fig. 6. Intuitively, fewer clusters promote action taking in people analytics since clusters need to be defined, and this becomes increasingly challenging as the number of clusters increases. With a large number of clusters, it may be difficult to meaningfully tailor career development or retention strategies, for example, to the unique needs of employees assigned to each cluster as the distinction between each subgroup becomes more opaque.

We can now implement K-means clustering with $K = 3$ using the `kmeans()` function in base R:

```
# Perform K-means clustering
km <- kmeans(employees_trans, centers = 3)
```

```
# Return n-count of clusters
km$size
```

```
## [1] 592 603 275
```

Of the 1470 employees in our `employees` data, the n distribution across the $K = 3$ clusters is 592, 603, and 275.

We can calculate the mean (and other descriptives) for each variable by cluster to better understand cluster distinctions:

```
# Calculate mean of each cluster using original data
aggregate(employees, by = list(cluster = km$cluster), mean)
```

```
##   cluster stock_opt_lvl trainings      age commute_dist   ed_lvl engagement
## 1       1     0.7787162  2.805743 35.31757     8.907095 2.880068   2.685811
## 2       2     0.7860697  2.781095 35.38474     9.646766 2.854063   2.792703
## 3       3     0.8436364  2.825455 43.75636     8.810909 3.112727   2.687273
##    job_lvl hourly_rate daily_comp monthly_comp annual_comp salary_hike_pct
## 1 1.834459    47.09291   376.7432      8162.77    97953.24        15.26520
## 2 1.724710    83.51078   668.0862     14475.20   173702.42        15.24876
## 3 3.301818    67.72364   541.7891     11738.76   140865.16        15.00364
##   perf_rating prior_emplr_cnt  env_sat  job_sat  rel_sat wl_balance  work_exp
## 1    3.148649        2.668919 2.746622 2.859797 2.673986   1.824324  9.315878
## 2    3.159204        2.645108 2.666667 2.665008 2.706468   1.852405  8.684909
## 3    3.152727        2.850909 2.789091 2.585455 2.807273   1.854545 21.196364
##   org_tenure job_tenure last_promo mgr_tenure interview_rating
## 1   5.146959   3.452703   1.467905   3.300676         3.979730
## 2   4.552239   2.827529   1.077944   2.878939         3.952570
## 3  16.527273   8.974545   6.170909   8.621818         4.090545
```

We can see that relative to the first two clusters, the third cluster has—on average—an older demographic with more education and a higher job level. In addition, employees in the first cluster earn significantly lower compensation, on average, which may be correlated with categorical variables that were initially dropped such as `dept` or `job_title`.

We can also add a new column in the `employees` data frame with the cluster assignment from K-means to facilitate further analysis:

```
# Add cluster assignment to df
employees <- cbind(employees, km_cluster = km$cluster)
```

While K-means clustering is a simple and efficient algorithm (even for large data sets), an a priori specification of K is not always ideal. K-means clustering will create K clusters—even if they are nonsensical—so caution must be exercised. Plotting WSS against cluster count as shown in Fig. 6 can be helpful in defining K, but alternative clustering algorithms exist that do not require K to be predefined.

Hierarchical Clustering

Like K-means clustering, **hierarchical clustering** seeks to group observations into clusters which have similar characteristics but different characteristics relative to the observations in other clusters. However, unlike K-means, the number of clusters is not specified prior to implementing the algorithm with hierarchical clustering. The optimal number of clusters is determined using a **dendrogram**, which is a tree diagram visualizing the hierarchical relationships in data.

One key difference between K-means and hierarchical clustering is that hierarchical clustering involves linkage methods to measure cluster similarity. There is not a one-size-fits-all option for linkage, as the performance of a given linkage technique can vary based on the structure of the data. Outlined below are the five most common types of linkage in hierarchical clustering:

1. **Complete Linkage:** the distance between two clusters is defined as the maximum distance between any individual data point in cluster A and any individual data point in cluster B
2. **Single Linkage:** the distance between two clusters is defined as the minimum distance between any individual data point in cluster A and any individual data point in cluster B
3. **Average Linkage:** the distance between two clusters is defined as the average distance between data points in cluster A and data points in cluster B
4. **Centroid Method:** the distance between two clusters is defined as the distance between the centroid of cluster A and the centroid of cluster B
5. **Ward's Method:** ANOVA-based approach in which the distance between clusters A and B is based on how the sum of squared distances increases when the clusters are merged

To implement hierarchical clustering, we will leverage the same centered and scaled data used for K-means clustering in the prior section. Note that the `km_cluster` column was only added to the original `employees` data; if this column was present in `employees_trans`, we would need to drop it so that the hierarchical clustering algorithm is not influenced by results of another clustering technique (K-means).

Since we do not know what linkage method will work best for these data, we will also develop a function that enables us to try a range of techniques and select the one that performs best. The `agnes()` function from the `cluster` library is used to implement hierarchical clustering:

```r
# Load library
library(cluster)

# Define linkage methods
# Note: centroid is not available for agnes() function
methods <- c("complete", "single", "average", "ward")
names(methods) <- c("complete", "single", "average", "ward")

# Create function to compute agglomerative coefficient
agg_coeff <- function(x) {
  cluster::agnes(employees_trans, method = x)$ac
}

# Compute agglomerative coefficient for each linkage method
sapply(methods, agg_coeff)
```

```
##  complete    single   average      ward
## 0.7990373 0.6248688 0.7582732 0.9571736
```

Agglomerative coefficients closer to 1 indicate stronger clustering performance. Therefore, Ward's distance measure performs best on these data, and we will implement hierarchical clustering using this linkage option.

```
# Perform hierarchical clustering using Ward's linkage method
hclust <- cluster::agnes(employees_trans, method = "ward")
```

To produce a dendrogram, the `pltree()` function from the `cluster` library can be used in conjunction with the `hclust` object holding the clustering results:

```
cluster::pltree(hclust, main = "Dendrogram")
```

At the bottom of the dendrogram shown in Fig. 7, each leaf of the tree represents an individual observation. Since $n = 1470$, the bottom of the tree is too congested to

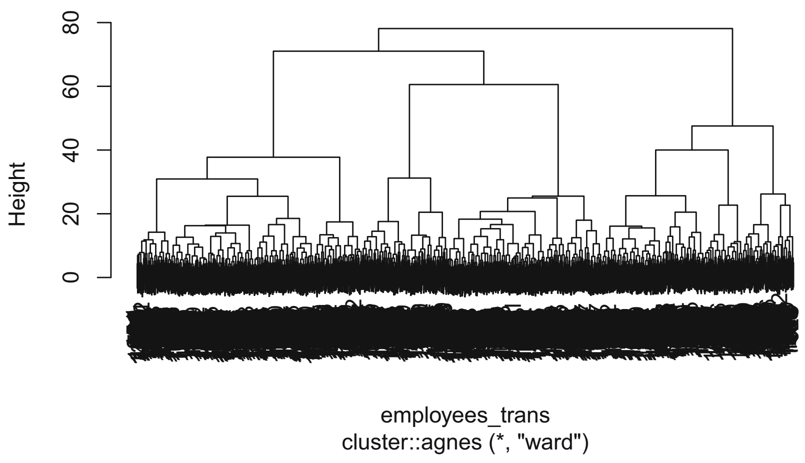

Fig. 7 Dendrogram for hierarchical clustering of employees using Ward linkage

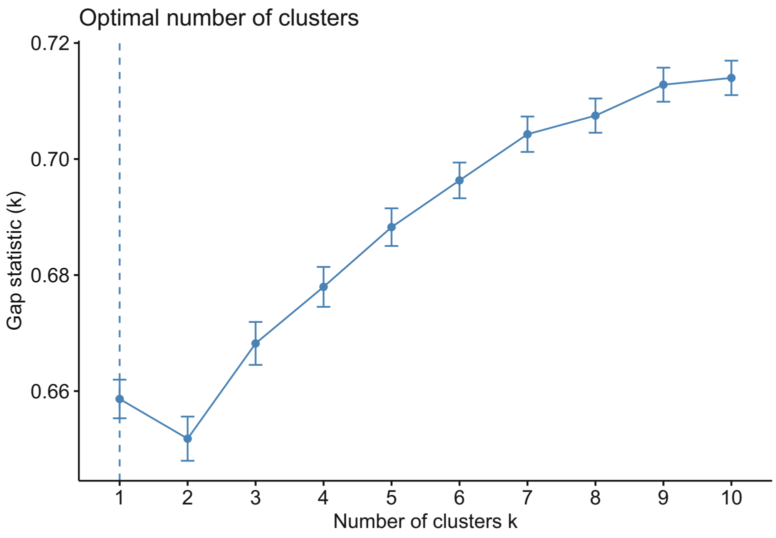

Fig. 8 Plot of gap statistic against cluster count for hierarchical clustering

interpret. As we move up the tree, individual observations are fused together based on the degree of similarity as defined by Ward's linkage method.

To aid in determining the optimal number of clusters, a **gap statistic** can be calculated, which compares the within-cluster variation for different K values to reference values for a random uniform distribution with no clustering. We will use the `clusGap()` function from the `cluster` library to calculate the gap statistic and then visualize using the `fviz_gap_stat()` function from the `factoextra` library:

```
# Calculate gap statistic across 1-10 clusters
gap_stat <- cluster::clusGap(employees_trans, FUN = hcut,
↪  nstart = 25, K.max = 10, B = 50)
# Generate plot of gap statistic against cluster count
factoextra::fviz_gap_stat(gap_stat)
```

We ideally want to select the value of K that maximizes the gap statistic. In practice, however, balancing cluster parsimony with maximization of the gap statistic is not always straightforward. Figure 8 indicates that the gap statistic increase is fairly constant across the range of $K = 2$ to $K = 10$ clusters. In this case, we may look to select a value of K based on an inflection point at which the trajectory of increase in the gap statistic begins to slow. Based on this approach, we may select $K = 7$.

We can now cut the dendrogram into 7 clusters using the `cutree()` function and then append the cluster to each observation in our original `employees` data:

```
# Compute distance matrix
d_matrix <- dist(employees_trans, method = "euclidean")

# Perform hierarchical clustering using Ward's method
hclust_final <- hclust(d_matrix, method = "ward.D2" )

# Cut the dendrogram into 7 clusters
groups <- cutree(hclust_final, k = 7)

# Append cluster labels to original data
employees <- cbind(employees, hier_cluster = groups)
```

Review Questions

1. How can high-dimensional data create problems in analytics, and how do dimension reduction techniques remediate these issues?
2. What is the difference between Exploratory Factor Analysis (EFA) and Confirmatory Factor Analysis (CFA)?
3. What is the difference between Exploratory Factor Analysis (EFA) and Principal Components Analysis (PCA)?
4. What is Structural Equation Modeling (SEM), and what are some use cases for it in people analytics?
5. How can we test whether data satisfy the eligibility criteria for factor analysis?
6. How are factor loadings interpreted to ascertain which variables are members of each factor?
7. What is a data-informed approach to selecting the optimal value of K in K-means clustering?
8. What is Euclidean distance, and what is its function in clustering?
9. How is a dendrogram interpreted in the context of hierarchical clustering?
10. When optimizing for both cluster parsimony and gap statistic maximization is not feasible, how can the optimal value of K be determined in hierarchical clustering?