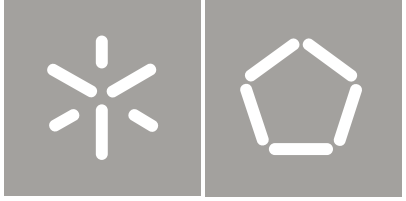
Universidade do Minho
Escola de Engenharia

André Filipe Gonçalves Magalhães

Data Mining methods to detect discrimination patterns along temporal databases

Outubro de 2012



Universidade do Minho
Escola de Engenharia

André Filipe Gonçalves Magalhães

Data mining methods to detect discrimination patterns along temporal databases

Mestrado em Engenharia Informática

Trabalho orientado por:

Professor Doutor Paulo Jorge Azevedo

Outubro de 2012

DECLARAÇÃO

Nome: André Filipe Gonçalves Magalhães

Endereço electrónico: afgmagalhaes@gmail.com Telefone: 966573862

Número do Bilhete de Identidade: 13599571

Título dissertação: Data mining methods to detect discrimination patterns along temporal databases

Orientador(es): Professor Doutor Paulo Jorge de Sousa Azevedo

Ano de conclusão: 2012

Designação do Mestrado: Mestrado Engenharia Informática

Universidade do Minho, __/ __ /____

Assinatura: _____

Acknowledgements

First of all, I am truly grateful for professor Paulo Azevedo guidance and assistance throughout the whole time dedicated to writing this dissertation. His suggestions proved to be of the utmost importance, and I have to give my sincere gratitude to him.

Assistance provided by all involved in the research scholarship reference "012/TT/12" was greatly appreciated. The research done was closely related and ended up being very valuable in this work.

I wish to express my sincere thanks to professor Miguel Portela for providing some of the data used and tested in this thesis, the time spent with me and the valuable feedback given.

I also place on record, my sense of gratitude to one and all who, directly or indirectly, have lent their helping hand in this venture, especially my friends and family for the unending encouragement and support.

Data Mining methods to detect discrimination patterns along temporal databases

In certain Data Analysis tasks, understanding the underlying differences between groups or classes is of the utmost importance. Contrast Set Mining relies on discovering significant patterns by contrasting two or more groups, each pattern being a Contrast Set which is a conjunction of attribute-value pairs that differ meaningfully in its distribution across groups.

One technique proposed is Rules for Contrast Sets (RCS) which seeks to express each Contrast Set found in terms of rules. The main purpose of this work is to extend this approach to a Temporal Data Mining task, developing a set of patterns in order to capture the significant changes in the contrasts discovered along the entire timeline considered. To ascertain the proposal accuracy and ability to discover relevant information, it will be applied in two different real-life datasets.

Métodos de Mineração de Dados para detecção de padrões discriminatórios em bases de dados temporais

Em determinadas tarefas de Mineração de Dados, perceber as diferenças fundamentais entre grupos ou classes é de extrema importância. Constrat Set Mining baseia-se na descoberta de padrões significativos contrastando dois ou mais grupos, onde cada padrão é um Contrast Set, que é uma conjunção de pares atributo-valor que diferem substancialmente na sua distribuição entre os grupos.

Uma técnica proposta é o Rules for Contrast Sets (RCS) que procura expressar cada contraste encontrado em termos de regras. O principal propósito deste trabalho é estender esta abordagem a uma tarefa de Temporal Data Mining, desenvolvendo um conjunto de padrões específicos para capturar as modificações significativas ao longo da linha temporal estabelecida. Para averiguar a precisão da proposta e a sua capacidade de encontrar informação relevante, ela será aplicada em dois *datasets* distintos, com dados de situações reais.

Contents

List of Figures

List of Tables

Listings

Acronyms list

3P%	Three Point Percentage
ABA	American Basketball Association
API	Application Programming Interface
ARM	Association Rule Mining
ARMA	Autoregressive Moving Average
AS	Advanced Scout
C	Center
CAREN	Class project Association Rule Engine
CIGAR	Contrasting, Grouped Association Rules
CS	Contrast Set
CSM	Contrast Set Mining
CSV	Comma-separated values
EP	Emerging Patterns
FG%	Field Goal Percentage
FT%	Field Throw Percentage

GSP	Generalized Sequential Pattern
GUI	Graphical User Interface
JCF	Java Collection Framework
LHS	Left-Hand Side
minconf	Minimum confidence
minsup	Minimum support
NBA	National Basketball Association
NUTS	Nomenclature of Territorial Units for Statistics
OLAP	On-line Analytical Processing
PF	Power Forward
PG	Point Guard
PPCS	Post-Processing Contrast Sets
RCS	Rules for Contrast Sets
REs	Regular Expressions
RHS	Right-Hand Side
SF	Small Forward
SG	Shooting Guard
sigdif	Significant difference
SM	Subgroup Mining
SPADE	Sequential Pattern Discovery using Equivalence Classes

SPIRIT Sequential Pattern Mining with Regular Expression Constraints

STUCCO Search and Testing for Understandable Consistent Contrasts

supdif Support difference between contrasted groups

TDM Temporal Data Mining

TID transaction unique identifier

TS Time Series

Chapter 1

Introduction

Data Mining is an area which seeks to discover hidden patterns from large datasets by using methods and techniques from several fields like Database Systems, Artificial Intelligence, Machine Learning and Statistics. Governments and organizations resort to these methods to improve their knowledge of the operational field they belong. One of the most popular, researched and used technique is Association Rule Mining (ARM) (Agrawal & Srikant, 1994; Agrawal, Imieliński, & Swami, 1993) that finds all the gripping associations in a dataset. These patterns are presented in the form of rules of the type $X \Rightarrow Y$, where X and Y are sets of atomic units occurring in the data (*itemsets*).

Understanding the difference between contrasting groups is also a fundamental Data Mining task (Bay & Pazzani, 1999; Webb, Butler, & Newlands, 2003) and it is a problem present in many domains. For example, census data collected this year can be compared to the data collected in a previous census activity, like contrasting the data collected this year against the one collected thirty years ago. This comparison involves two groups (e.g. 2011 *vs* 1981), and in this scenario it is fairly easy to infer some differences among these two groups in contrast that can be expected to be retrieved: the mean number of

children per couple should be lower nowadays, but the income and education likely follow the reverse trend. This notion can easily be extrapolated to other domains.

Although ARM captures the relations between the items present in the data, it does not discriminate in regard to difference towards those same items. Even so, one proposal has shown that a commercial Association Rule Learner (Magnum_OPUS) with some tweaks could achieve this task fairly well (Webb et al., 2003). Due to this latent inability, some techniques derived from ARM have been proposed to tackle this problem. Contrast Set Mining (CSM) (Bay & Pazzani, 1999; Hilderman & Peckham, 2005; P. J. Azevedo, 2010) has emerged as a Data Mining task whose objective is to effectively collect every Contrast Set (CS), a formalism used to represent the patterns that we are looking for, i.e. the group differences. Emerging Patterns (EP) (Dong & Li, 1999) and Subgroup Mining (SM) (Klosgen, 1996) are other techniques that achieve different yet similar objectives. The formulation and notation is divergent due to the fact that they have been proposed by different communities for various purposes (Novak, Lavrač, & Webb, 2009).

Rules for Contrast Sets (RCS) is one algorithm that redesigns an association rule engine to derive rules to describe CSs (P. J. Azevedo, 2010). and the work developed in this thesis is mainly centred around this proposal specifically by extending its usage into a Temporal Data Mining (TDM) task. Temporal Data Mining deals with the discovering of potential useful information from temporal data (Theophano, 2010). This requires the inclusion of *temporality* into databases as well as its data representation in order to discover the *temporal patterns* of interest (Laxman & Sastry, 2006). The proposal present in this thesis has its foundations in both CSM and TDM and pretends to extend the discrimination provided by contrasting groups to a temporal scenario.

1.1 Motivation

In contrasting two or more elements, the main objective relies on obtaining the attributes that distinguish those elements from each other. This comparison task can also be valuable as a form to deepen even further the knowledge about the problem in question. Some proposals have been made in order to perform this task, but none took the notion further in order to contrast and differentiate in a timely manner, verifying the contrast modification at different points of time.

Two certain groups being contrasted in a certain point of time could have just a few distinguishable features. Nothing guarantees that this relation between them has suffered a meaningful modification somewhere in another period either on the past or the future for a certain attribute or set of attributes. This proposal pretends, essentially, to look up on this matter, understanding and identifying the contrasts evolution along a defined timeline bridging together the areas of CSM and TDM, each one proven valuable in obtaining patterns of interest.

1.2 Objectives

The main goal of this thesis is set on proposing and developing a solution for understanding group difference in a temporal-based dataset. However, in order to accomplish the objective of contrasting in a timely manner, the development of a new set of temporal patterns to define trends and phenomenons that have occurred in the time window considered affecting the set of CSs found is a crucial requirement. This set of patterns will allow to detect and point possible situations of interest that mark a significant change in the contrast behaviour, potentially comprising highly valuable information for the end user.

Since these contrasts can suffer constant modification at each period or even remain unaltered, a form to evaluate its stability surge as a relevant mean to globally define each contrast behaviour in the entire time window. This can assume several trends: a legit accentuating or discarding factor depending on what is sought, contrasts that take on a stable behaviour for most periods or the opposite.

Not only what has been stated before is a fundamental goal of this proposal, but the visualization of the results obtained is of the utmost significance as well. The results should be clear to someone who is going to analyse the results obtained, and the form how they are presented assume a pivotal role in that regard.

Efficiency and accuracy are relevant aims likewise. Any Data Mining method performance either in terms of computational effort, precision and faithfulness of the results obtained, need to be confronted against datasets, preferably with data from real contexts. This will allow to perceive the technique computational burden in regard to the dataset size and in terms of the results obtained, facing them up against common knowledge of the context at hand.

1.3 Document organization

In the next chapter, some of the most-known proposals from both CSM and TDM are presented that, directly or indirectly, contributed with some elements for the development of the solution presented.

The proposal developed is described with detail in the third chapter. The entire set of patterns and measures developed are exposed, always revealing what typical situations they tend to identify and alert. The used and developed applications are exhibited as well as its functional objectives and how

they are coupled in order to accomplish the task proposed.

In the next two chapters, two distinct case studies are presented, both encompassing data from real-life scenarios. The results obtained in each case will be the object of detailed analysis and faced against the context-related knowledge, trying to understand some relations (or lack of) between contrasted groups, as well as interesting or surprising events.

In the last chapter, the conclusions come up where a global appreciation of the proposal is done, debating its own limitations and possible improvements as a form of future work.

Chapter 2

State of the Art

2.1 Association Rule Mining

Some authors refer this area as closely related to CSM (Bay & Pazzani, 2001; Hilderman & Peckham, 2005). This problem is normally associated with customers' buying patterns from retail data (known as Market-Basket data) and it is formally stated as follows. Let $A = \{A_1, A_2, ..., A_n\}$ be the set of attributes called *items*. D is a set of *transactions*, where each transaction T contains a vector of attribute-value pairs of the type $A_1 = V_1, A_2 = V_2, ..., A_m = V_m$, and V_i can only take a value from the domain {0,1}, where $V_i = 1$ represent that the item i was purchased and $V_i = 0$ otherwise. The set of items associated with each transaction, $A \subseteq T$, is called a *itemset*. To able to distinguish each transaction, a transaction unique identifier (TID) is given.

An example of Market-Basket data is present in Table 2.1, each transaction is identified by its unique TID (5 in total), $|A| = 3$ and for example, transactions with TID 2 & 4 represent a customer that purchased meat but not eggs and milk.

Rules that are discovered take the form of $X \Rightarrow Y$, where $X \subset A$, $Y \subset A$ and $X \cap Y = 0$. X is the Left-Hand Side (LHS) of the rule, also known as the

TID	Eggs	Milk	Meat
1	1	1	0
2	0	0	1
3	1	1	1
4	0	0	1
5	1	1	1

Table 2.1: Small example of Market-Basket Data

antecedent while Y corresponds to the Right-Hand Side (RHS) or *consequent*.

The rules found have to satisfy some constraints, the most usual ones are Confidence and Support (Agrawal et al., 1993). *Confidence* of a rule $X \Rightarrow Y$ is the percentage of transactions in D containing X that also contain Y, which is the conditional probability of Y given X, i.e $P(Y|X)$. *Support* represents the coverage of items in D, the percentage of transactions that contain X and Y. Others constraints have been proposed like *lift* (Berry & Linoff, 1997), *conviction* (Brin, Motwani, Ullman, & Tsur, 1997), *interest* (Brin et al., 1997) or *improvement* (Bayardo,Jr., Agrawal, & Gunopulos, 2000). Original association rules discovery (Agrawal et al., 1993) in this context outputs all rules $X \Rightarrow Y$ such that $support \geq minsup$ and $confidence \geq minconf$ where *minsup* and *minconf* are user-defined variables.

Regarding the data present in Table 2.1, one rule that could be derived (assuming the *Minimum support (minsup)* and *Minimum confidence (minconf)* constraints are met) as an example is:

```
(eggs = 1) & (milk = 1) => (meat = 1)
[sup = 0.4; conf = 0.66]
```

This can be read as: 40% of the whole population of consumers buy eggs, milk and meat. Also, 66% of the transactions that include the purchase of eggs and milk include meat. Alternatively this can be read as: 66% of the sub-population who buys eggs and milk also buy meat. The numbers between

brackets can easily be calculated because of the small size of the dataset used as example. Support is 40% because two transactions out of five include purchase of eggs, milk and meat and confidence is 66% because just two consumers who bought milk and eggs have also bought meat out of a universe of three customers.

Some implementations have been proposed for finding Association Rules, *Apriori* (Agrawal & Srikant, 1994; Agrawal et al., 1993) was the first proposal and consists of a two-step algorithm, first finding all *large itemsets* (set of items that occur frequently, i.e. meet *minsup* constraint) then builds rules that contain the *large itemsets* found in the previous step. A later proposal, *FP-Tree* (Han, Pei, & Yin, 2000) is an order of magnitude faster than *Apriori* because it compresses the dataset in a condensed and compact data structure that avoids some of the costly database scans. It uses a divide-and-conquer strategy and a pattern fragment growth method that permits to be more efficient by avoiding the costly process of candidate generation and testing used by *Apriori*. Other algorithms have also been proposed like *Eclat* (Zaki, 2000) and *OPUS_Search* (Webb, 1995).

2.2 Contrast Set Mining

Proposed for the first time by Bay and Pazzani (1999), this problem was defined as having a goal of finding all contrast sets whose support differ meaningfully across groups. As referred in the last section, ARM usually deals with market-basket data but for this problem the data model evolves to grouped categorical data. The *itemset* concept present in Association rules can be extended to contrast sets as defined by Bay and Pazzani (1999):

Definition 1 *Let $A_1, A_2, ..., A_k$ be a set of k variables called attributes. Each A_i can take on values from the set $\{V_{i1}, V_{i2}, ..., V_{im}\}$. Then a* ***contrast-set*** *is*

a conjunction of attribute-value pairs defined on groups $G_1, G_2, ..., G_n$.

Example: $(sex = male) \wedge (occupation = manager)$

In this context, the support is considered in regard to the group and not to the whole dataset, meaning that the support of a contrast set cs is the percentage of examples in group G where the contrast set is true.

Formally, the objective is to find all the contrast sets (cs) that meet the following criteria:

$$\exists ij \; P(cs|G_i) \neq P(cs|G_j) \tag{2.1}$$

$$max_{i,j}|sup(cs, G_i) - sup(cs, G_j)| \geq \delta \tag{2.2}$$

where δ is a user-defined threshold named *minimum support difference*. These two equations albeit different represent the same goal, finding contrast sets whose *support* differ meaningfully across groups. Equation 2.1 guarantees that the contrast set represent a true difference between at least a pair of groups (i.e. the basis of a statistical test of meaningful) and equation 2.2 ensures that only contrast sets whose difference is big enough to be considered relevant are obtained. The contrast sets that Eq. 2.1 is statistically valid are called *significant* and those that met Eq. 2.2 are referred as *large*. If both criteria is met, they are considered as *deviations*.

The problem of mining contrast sets is a tree search problem. The root node is an empty contrast set and for each children one item is added to the contrast set, so at level k of the tree we have all possible contrast sets that contain k items. A small example is show in figure 2.1 with only two attributes (A_1 and A_2) with two possible values for each one. A canonical ordering for attributes is used (Riddle, Segal, & Etzioni, 1994) and that ensures that each node is not visited twice. For that reason, nodes in level 1 that include A_2

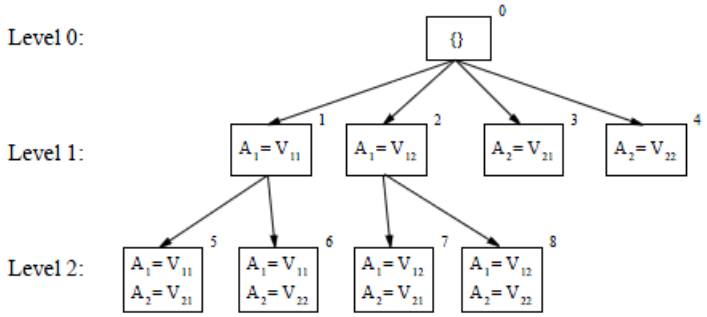


Figure 2.1: Example of a search tree with two Attributes $A_1 = \{V_{11}, V_{12}\}$ and $A_2 = \{V_{21}, V_{22}\}$ [Bay and Pazzani 1999]

don't have any children node in figure 2.1 (A_1 is visited first). It is easy to imagine that for a real case there will be a lot of attributes and a considerable set of possible values for each one contributing for the tree to grow exponential in the number of Attribute-Value pairs making the traversal of the *search space* very costly. Fortunately, *pruning* methods allow to reduce significantly the number of nodes needed to visit because the discarded (pruned) nodes will not contain interesting discoveries.

These are the basis CSM concepts but there are specificities that vary from each implementation and those will be dealt in the following subsections while discussing specific algorithms.

2.2.1 STUCCO

Presented in the first paper that introduced CSM (Bay & Pazzani, 1999), Search and Testing for Understandable Consistent Contrasts (STUCCO) is still widely used for mining CS. It is based on Max-Miner (Bayardo,Jr., 1998) rule-discovery algorithm and uses a breadth-first search framework.

In order to check for *significant* contrast sets (Eq. 2.1) a statistical test is required. The null hypothesis to be tested is: *contrast set support is equal across all groups*. The *support* counts needed for this are organized in a $2 \times G$

contingency table where the row variable represent the truth of the contrast set and the columns represent each group considered. Table 2.2 serves as an example.

STUCCO uses a standard test for testing independence of variables in a contingency table, the *chi-square* test. The statistic χ^2 is calculated as follows:

$$\chi^2 = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \tag{2.3}$$

where O_{ij} is the observed frequency count for the cell in row i and column j . E_{ij} is the expected frequency count in cell ij given independence of the row and column variables and is calculated as follows: $E_{ij} = \sum_j O_{ij} \sum_i O_{ij}/N$ with N being the total number of observations. The result is then compared to the distribution of χ^2 when the null hypothesis is true.

	Location=urban	Location=rural	$\sum row$
$Stress = high$	194	355	549
$\neg(Stress = high)$	360	511	871
$\sum column$	554	866	1420

Table 2.2: Contingency table for example $Stress \times Location$

A test α level has to be selected in order to check if the differences are significant. This choice sets the maximum probability of rejecting the null hypothesis when it was in fact true and shouldn't have been rejected. Considering table 2.2, $\chi^2 = 5.09$ with 1 degree of freedom and a p-value of 0.024. Since the p-value is less than the $\alpha = 0.05$ (standard value for statistical tests), it can be inferred that the null hypothesis is likely false and the conclusion that arises is that *contrast set support is not equal across all groups.*

The α level sets the maximum probability of falsely rejecting the null hypothesis for each test. In a case where multiple tests have to be applied,

this probability quickly rises. Imagine, for example, a scenario where 2000 tests have to be performed, with α set as 0.05 we are expected to get 100 false discoveries (2000×0.05) and that is clearly unacceptable. Incorrectly rejecting the null hypothesis (concluding that a difference exists when it doesn't) is known as a *Type I error* or *false positive*.

Type I Error probability can be controlled by using the Bonferroni inequality, that is defined as follows: given any set of events $e_1, e_2, ..., e_n$, the probability of their union $(e_i \vee e_2 \vee ... \vee e_n)$ is less than or equal to the sum of the individual probabilities. Applied to hypothesis testing, we let e_i be the rejection of the *ith* hypothesis h_i. Consequently, h_i is rejected if $p_i \leq \alpha_i$ where $\sum \alpha_i \leq \alpha$. Usually $\alpha_i = \alpha/n$, where n is the total number of tests.

STUCCO uses a specific Bonferroni adjustment given by the following equation:

$$\alpha_l = min(\frac{\alpha}{2^l}/|C_l|, \alpha_{l-1}) \tag{2.4}$$

where α_l is the cutoff at level l and $|C_l|$ is the number of candidates (number of hypothesis evaluated) at level l. The minimum requirement ensures that the α become smaller at each level, reducing the probability of false discoveries at lower levels but it also decreases the number of contrast sets at these levels (those with a significant number of items).

Regarding pruning, STUCCO prunes away all nodes that are not *deviations*. Nodes of the search tree are pruned based on some criteria (Bay & Pazzani, 1999, 2001) when there's a guarantee that a node and its own subtree won't contribute for finding deviations, for this reason they don't need to be visited further.

Still, not all *deviations* are shown to the user as some of them are pruned away. This is **Interest Based Pruning** (Bay & Pazzani, 2001) and involves

the notion of what is considered *interesting*. CSs are not interesting when they add no new information and this may happen when specializations of the contrast set have the same *support* than its generalization. If the *support* remains the same on both generalization and specialization, they will cover the same instances in the database and having the same *support* in many cases, represent trivial findings (Bay & Pazzani, 2001).

Algorithm 1 represent how STUCCO is implemented as shown by Bay and Pazzani (2001).

input : data D
output: $D_{surprising}$
Set of Candidates $C \leftarrow \{\}$
Set of Deviations $D \leftarrow \{\}$
Set of Pruned Candidates $P \leftarrow \{\}$
Let prune(c) return true if c should be pruned
while C *is not empty* **do**
    scan data and count support $\forall c \in C$
    foreach $c \in C$ **do**
        if *significant(c)* $\wedge$ *large(c)* **then** $D \leftarrow D \cup c$
        if *prune(c) is true* **then** $P \leftarrow P \cup c$
        else
            $C_{new} \leftarrow C_{new} \cup$ GenChildren(c,P)
        end
        $C \leftarrow C_{new}$
    end
    $D_{surprising} \leftarrow$ FindSurprising(D)
end

Algorithm 1: STUCCO algorithm

Last but not least, after determining which CS are interesting, it is presented to the user in the following form:

```
hours_per_week = ]20.6:40.2]
2880 857 161 | 0.537815 0.497388 0.389831
============================
d.f.    chi^2    pvalue
2       38.37    4.65e-09
============================
```

This is a CS with just one item (hours per week) in a domain with 3 groups. In the second line there is the absolute and relative values of *support* within each group and below the statistical values like degrees of freedom, χ^2 statistic value and its p-value. This representation has a flaw because it does not show in which combination of groups there is a significant difference in *support*.

2.2.2 CIGAR

Contrasting, Grouped Association Rules (CIGAR) was proposed by Hilderman and Peckham (2005) and is closely related to STUCCO approach to mine CS from categorical data. Like STUCCO, CIGAR also wants to find all contrast set that meet Eq. 2.1 and 2.2 but adds three new constraints.

$$support(X, G_i) \geq \beta \tag{2.5}$$

$$correlation(X, G_i, G_j) \geq \lambda \tag{2.6}$$

$$|correlation(X, G_i, G_j) - correlation(child(X, G_i, G_j))| \geq \gamma \tag{2.7}$$

β is the user-defined *minimum support threshold*, λ is the user-defined *minimum correlation threshold* and γ is the user-defined *minimum correlation*

difference threshold. The CSs that meet Eq.2.5 are called *frequent* and those who meet Eq.2.6 are called *strong*. The *deviation* concept is redefined in CIGAR because a *deviation* here is a CS that meets the first four conditions. The *deviations* that do not fulfil the last equation are *spurious* and pruned away.

The basic strategy remains the same, finding all interesting *deviations*. The search space is traversed and candidates generated in the same breadth-first, level wise manner. *Significant* and *large* candidate sets are verified the same way as STUCCO. Because of Eq.2.6 and 2.7, CIGAR needs to calculate the correlation factor and that is obtained using the *Phi correlation coefficient* ϕ (Hilderman & Peckham, 2005) from the values present in the corresponding contingency table. This coefficient measures the relation between two variables and takes a value between -1 and 1. A value close to zero represents that the two variables have no association and should be pruned away.

While STUCCO organized frequency counts in a $2 \times G$ contingency tables, CIGAR breaks each one of these contingency tables into a series of 2×2 contingency tables (the exact number being $\begin{pmatrix} G \\ 2 \end{pmatrix}$ where G is the number of groups). This allows to distinguish between each exact groups the differences in *support* occur as this is a much fine-grained approach compared to STUCCO.

Despite the Type I Error control present in STUCCO, the authors of CIGAR decided not to control this. The reason behind that lies in the fact that decreasing the probability of a Type I Error increases the probability of Type II Error (accepting the null hypothesis when it was false) and this kind of control decrease the chance of obtaining more specific contrasts sets because of a smaller α at each level.

Pruning is obtained much like STUCCO strategy pruning away all non-deviations candidate sets. One minor difference lies in Statistical Significance

pruning. Pruning every node where the expected frequencies are too small is the strategy followed by STUCCO but that can mean a lost opportunity of obtaining a interesting contrast set. To avoid this, CIGAR uses Yates' correction for continuity (Everitt, 1992) that provides a more conservative estimate of the χ^2 statistic and more accurate when the frequencies are considered small.

Unfortunately, pseudo code and algorithm output were not revealed in the paper that introduced CIGAR (Hilderman & Peckham, 2005).

2.2.3 Rules for Contrast Sets

RCS (P. J. Azevedo, 2010) is a proposal that makes uses of an existing association rule engine (P. Azevedo, 2012) redesigned to mine contrast sets that are expressed in form of rules. Rules are known by their ease of interpretation and expressive power making this representation easier to read than the one STUCCO adopted. Like a frequent *itemset* algorithm, search space traversal is performed in a depth-first manner contrasting to other proposals like STUCCO and CIGAR that do it in breadth-first manner. This type of traversal does not fully exploit the downward closure property of *support* (Agrawal et al., 1993) but still leads to an efficient rule-based algorithm (P. J. Azevedo, 2010).

CSs mined by this algorithm have to meet Eq.2.1 and 2.2. Although not specifically introduced as a equation, the *minsup* criteria is also used here.

	G_i	G_j	$\sum row$
cs	a	b	a + b
$\neg cs$	c	d	c + d
$\sum column$	a + c	b + d	n

Table 2.3: A generic Contingency table for two groups and a contrast set.

This implementation, like CIGAR, uses 2×2 contingency tables like the

one present in table 2.3. As seen before, this allows to perceive between which exactly groups the differences are significant but unlike the previous two proposals, a χ^2 test is replaced by a Fisher-exact test that is directional (one-sided) to determine if the frequencies observed are significant. The p-value is computed as follows:

$$p = \sum_{i=0}^{min(b,c)} \frac{(a+b)!(c+d)!(a+c)!(b+d)!}{(a+b+c+d)!(a+i)!(b-i)!(c-i)!(d+i)!} \tag{2.8}$$

Fisher exact-test has some advantages compared to χ^2 test. Fisher is an exact test and suitable for small samples, this way no Yates' correction is needed. Being a directional test instead of a two-sided test that is the case of χ^2, a smaller p-value is generally obtained for datasets with 2 groups allowing more patterns to be found within the same cut-off level(P. J. Azevedo, 2010). The directionality of the test implies a slightly different null hypothesis:

$$H_0 : P(cs|G_i) \leq P(cs|G_j) \tag{2.9}$$

Because of this, instead of Eq. (2.1), $\exists i, j \; P(cs|G_i) > P(cs|G_j)$ is used. Even being slightly different, the same principle applies as both derivations still capture the same *significant* patterns.

False discoveries are controlled differently than STUCCO. The layered critical values (Webb, 2008) proposal is adapted for this context. The adjustment used in STUCCO (Bay & Pazzani, 2001) considers the number of hypothesis evaluated rather than the size of the search space. That is, only accounts for candidate patterns that met a set of constraints. However, candidates are the patterns most likely to pass the statistical test. Thus, the critical value should be adjusted by the number of patterns from which those to be tested are selected instead of the number of times the statistical test

is applied. Work done by Webb (2007) introduces a form to calculate the search space size before deriving any rule either with Market-Basket data or Attribute-value data that is easily adapted to this scenario. This is shown with more detail in these works (P. J. Azevedo, 2010; Webb, 2008).

RCS also differ from STUCCO in pruning non-relevant rules. Relevance pruning used here is more elaborate than the interest based pruning used in STUCCO and relies on the premise that a specialization of a rule should only be revealed to the user if it yields a significant improvement. Fisher-exact test is used here for this effect and each candidate rule has to pass n statistical tests where n is the number of generalizations of that rule including the contrast set $\emptyset$.

input : dataset D, list of groups G, minsup ms, cutoff α
output: ResultSet of contrast rules RS
1 Compute $\alpha'_1, \alpha'_2, ..., \alpha'_n$, for a supplied n or using number of frequent_items/attributes;
2 $RS := \emptyset$;
3 **foreach** *node* $i \in depth\_first\_search(D, ms) : sup(i) \geq ms$ **do**
4 **foreach** *pair* $g_a, g_b \in G$ *where* $sup(i, g_a) >> sup(i, g_b)$ **do**
5 **if** $Fisherpvalue(g_a, g_b, i, \emptyset) \leq \alpha'_{length(i)}$ **then**
6 **if** $\forall g_a >> g_b \leftarrow cs \in RS$:
 $cs \subset i$ & $Fisherpvalue(g_a, g_b, i, cs) \leq \alpha'_{length(i)}$ **then**
7 $RS := RS \cup \{g_a >> g_b \leftarrow i\}$;
8 **end**
9 **end**
10 **end**
11 **end**

Algorithm 2: (RCS) Rules for Contrast Sets.

Algorithm 2 shows how RCS mines the rules that describe contrast sets. Line 1 represent the calculation of the layered critical values that will be used for statistical tests each level of the search space. Line 3 shows the depth-first traversal of the search tree while checking only nodes whose *support* is higher than *minsup*. Line 5 & 6 show the application of the Fisher-exact tests either

for statistical significance (Line 5) and relevance pruning (Line 6).

Rules that describe the CSs whose *support* differ across groups are formally organized as follows: $G_1 >> G_2, ..., G_i >> G_j \leftarrow cs$, where cs represents the contrast set and G_i each group. The pairs $G_i >> G_j$ indicate the direction to where the *support* differs (G_i has bigger *support* than G_j). Consider the following example:

```
Gsup = 0.17191 | 0.04121  p = 1.1110878451E-017 education=Doctorate >> education=Masters
Gsup = 0.17191 | 0.01681  p = 3.0718399575E-040 education=Doctorate >> education=Bachelors
Sup(CS) =  0.03097                                          <---   workclass=State-gov  &
                                                                   class > 50K.
```

The rule is to be read as: *the occurrence of the contrast set "working for the state government and making an income of more than 50K" is significantly larger within people holding an PhD than a MSc. The same occurs between PhD and BSc holders. Gsup* refers to the *support* of the CS in the group (for example, 17,91% of the PhD holders are State-Governers and have a salary superior to 50,000) and *Sup(CS)* is the *support* of the CS in the entire database. The *p-value* of the Fisher-exact test is also shown. This approach is much more readable than STUCCO output because of the rule format and because it is explicit which groups are involved and which direction occurs the difference between those groups.

2.2.4 Related Work

Other relevant contributions to CSM that were not closely described before include a technique presented by Wong and Tseng (2005) for describing contrast sets that can include the negation of each attribute-value pair and used it for designing custom insurance programs. Webb et al. (2003) investigated the use of CSM in a real case of retail sales. Other proposal COSINE (Simeon & Hilderman, 2011), is a CSM algorithm that mines maximal contrast sets using

a vertical representation of the database and deals with the discretization of numeric values evolving from the concept of categorical data.

Emerging patterns (Dong & Li, 1999) is an association rules based approach to report differences among datasets. As it also seeks to find differences, it is a related topic of finding discrimination patterns. The pattern ep is an Emerging Pattern if the $growth\_rate(ep) = \frac{sup_1(ep)}{sup_2(ep)} \geq \rho$, where ρ is a user provided threshold. The Growth Rate serve as a measure to the quality of each emerging pattern. This technique has been widely used in bioinformatics, especially in microarray data analysis (Novak et al., 2009).

Subgroup mining can also be identified as a discrimination process and was defined as following (Klosgen, 1996): Given a population of individuals and a property of those individuals that we are interested in, find population subgroups that are statistically most interesting, for example, are as large as possible and have the most unusual statistical (distributional) characteristics with respect to the property of interest. One example is (Jorge, Azevedo, & 0002, 2006) where numeric properties of interest are analysed instead of the usual categorical data.

A survey of CSM, EP and SM (Novak et al., 2009) introduces a unifying framework that with new terminology, task definitions and heuristics interrelates the three areas which can be seen as a single task of pattern discovery.

2.3 Temporal Data Mining

TDM is concerned with data mining of large *sequential* datasets (Laxman & Sastry, 2006). Temporal data may be categorized in two main types: sequential data, a sequence composed by a series of nominal symbols from a particular alphabet (Antunes & Oliveira, 2001) and time series data, also a sequence but composed of continuous and real-valued element values where

each event has uniform distance in the time window (Shahnawaz, Ranjan, & Danish, 2011).

Time Series (TS) analysis date back longer than TDM. Stock market, medical care or weather forecasting are examples of the most common problems studied by TS analysis (Laxman & Sastry, 2006) (Antunes & Oliveira, 2001). TDM, however, has a different approach as the goals are somewhat distinct.

The typical datasets used size differ quite significantly. TDM methods need to deal efficiently with datasets that hold a massive number of records. This size is insupportable for typical TS modelling techniques like Autoregressive Moving Average (ARMA) that employ much more computational effort, taking hours or even days to fully execute. Besides this, sequential data, which is commonly present in these datasets, renders this type of TS techniques inapplicable.

Another element that distinguishes these two approaches is the type of information that is expected to be obtained or extracted from their usage. TS analysis is usually related to forecasting and regression analysis while in TDM, there are different techniques used for prediction, but also for pattern mining, classification, clustering among others which composes a broader analysis, suited for distinct situations with different objectives.

Since the focus of this dissertation relies on the detection of trends and patterns, the techniques and notions presented in the next sections are solely related to pattern discovery.

2.3.1 Sequence mining

To put it bluntly, sequence mining seeks to unearth all patterns of interest (Shahnawaz et al., 2011) from sequential data. To discover such patterns in a sequence of events, three steps are usually associated with this approach

(Antunes & Oliveira, 2001):

1. Representation and modelling: Temporal data is transformed into a suitable form;
2. Similarity measure: Definition of similarity measures between sequences;
3. Mining Operation: Application of models and representations to the actual problem;

The general problem of sequence mining was stated by Pujari (2001) as:

Definition 2 *Let* $\sum = \{i_1, i_2, ..., i_m\}$ *be a set of distinct items comprising the alphabet. An event is a non-empty, disordered collection of items denoted as* $(i_1, i_2, ..., i_k)$ *where* i_j *is an item in* $\sum$*. A sequence* $s = \{t_1, t_2, ..., t_n\}$ *is an ordered set of events.*

GSP algorithm

Work developed by Agrawal et al. (Agrawal & Srikant, 1995) is arguably seen as the birth of TDM field and is basically an extension of the original ARM framework proposed for a database containing a set of unordered transactions known as the *Apriori* algorithm (Agrawal et al., 1993), briefly seen in section 2.1.

This sequential pattern mining framework keeps the notion of frequent *itemsets* like in *Apriori* except they now have a temporal order associated to them. The unordered set of transactions that composed the database used in *Apriori* gives place to a new database with a timestamp associated to each single transaction. The transactions of each single costumer are a sequence of *itemsets* ordered by time.

A sequence $a = \{a_1, a_2, ..., a_n\}$ is contained in another sequence $b = \{b_1, b_2, ..., b_n\}$ if there exists integers $i_1 < i_2 < ... < i_n$ such that $a_1 \subseteq$

$b_1, a_2 \subseteq b_2, ..., a_n \subseteq b_n$. For example, the sequence $\{(1), (2\ 3)\}$ is contained in $\{(9), (1\ 4), (2\ 3\ 5), (8)\}$ since $(1) \subseteq (1\ 4)$ and $(2\ 3) \subseteq (2\ 3\ 5)$. But, the sequence $\{(1), (2)\}$ is not contained in $\{(1\ 2)\}$. The former represents a situation where products 1 and 2 were bought on separate occasions, while the latter represents both products being bought at the same time. A given sequence s is maximal if s is not contained in any other sequence.

The *support* for any sequence, s, is the fraction of total customer transactions which contain s (Agrawal & Srikant, 1995). The problem at hand, for mining sequential patterns, is to find the maximal sequences from the set of sequence that meet the *minsup* constraint (large sequence). Each such maximal sequence represents a *sequential pattern*.

Considering the database represented in table 2.4 as a set of customer sequences, if *minsup* is set to 20% (i.e. minimum *support* of two customers), the sequences $\{(3), (9)\}$ (supported by customers 1 and 4) and $\{(3), (4\ 7)\}$ (customers 2 and 4) are maximal and meet the *support* constraint. Thus, they are the *sequential patterns* sought. Some sequences like $\{(3)\}$ and $\{(4\ 7)\}$, although having minimum *support*, are not *sequential patterns* as they are not maximal.

Customer ID	Customer Sequence
1	$\{(3), (9)\}$
2	$\{(1\ 2), (3), (4\ 6\ 7)\}$
3	$\{(3\ 5\ 7)\}$
4	$\{(3), (4\ 7), (9)\}$
5	$\{(9)\}$

Table 2.4: Example of a Customer-Sequence Database

The Generalized Sequential Pattern (GSP) algorithm makes multiple data passes (Srikant & Agrawal, 1996). First, all sequences of length 1 (1-sequences) are counted. From the frequent items, the candidate set of 2-sequences is formed with another pass done in order to count their frequency. The fre-

quent 2-sequences generate the candidate sequences with length 3 and so on until there are no more frequent sequences found. The algorithm is essentially comprised of two main steps (Srikant & Agrawal, 1996):

- Candidate Generation: From the set of $(k-1)$-frequent sequences $F(k-1)$, the candidate set for the next pass is generated by the union of $F(k-1)$ with itself.
- Support Counting: An usual hash tree-based search is employed for efficiently counting the frequency of the sequences with the goal of removing the non-maximal ones.

A pseudo-code representation of the algorithm is visible in Algorithm 3.

input : Database D
output: ResultSet of all frequent sequences RS

```
1  k = 2;
2  RS = ∅;
3  F_1 = set of frequent sequences of length 1;
4  while F(k − 1) != NULL do
5      Generate candidate sets C_k;
6      foreach sequence s : database D do
7          Increment count of all a in C_k if s supports a;
8          F_k = {a ∈ C_k such that its frequency exceeds the threshold};
9          k = k + 1;
10     end
11     RS = ∪_k F_k;
12 end
```

Algorithm 3: GSP algorithm

SPADE

Sequential Pattern Discovery using Equivalence Classes (SPADE) was proposed by Zaki (2001) to efficiently discover *sequential patterns* especially by avoiding the repeated database scans used by GSP. The databases passes are reduced to only three outperforming GSP by a factor of two (Zaki, 2001).

It uses a vertical database representation where each list is associated to each item. This representations allows for enumerating all frequent sequences through temporal joins. SPADE uses a *bottom-up* approach generating all frequent sequences while moving up the lattice by using some efficient search techniques and taking advantage of some properties that are guaranteed by the lattice hierarchy.

SPIRIT

Sequential Pattern Mining with Regular Expression Constraints (SPIRIT) is a sequence mining algorithm (Garofalakis, Rastogi, & Shim, 1999) that gives the user a chance to specify constraints in the form of Regular Expressions (REs). These REs provide a simple a natural syntax and they have plenty expressive power to specify a wide range of interesting, non-trivial patterns constraints (Garofalakis et al., 1999).

Not all frequent sequences are mined from a specific database D, just those who beside being frequent, all of its subsequences satisfy the set of constraints specified C. This set of constraints will allow to reduce the volume of potentially useless results and by implication the computational burden that is eased by discarding a considerable number of candidate sequences which do not match the REs provided.

2.3.2 Frequent episodes

Another approach for unearthing temporal patterns is the frequent episode discovery framework (Mannila, Toivonen, & Inkeri Verkamo, 1997). In this framework, the objective is to find temporal patterns (designated here as *episodes*) that appear a sufficient number of times from the *event sequences* given.

Manilla et al. (1997) applied the framework in a telecommunication alarm

setting. The main objective was to find relationships between alarms from the discovered *episodes* in order to better explain the problems that cause alarms to fire and to predict severe faults.

The sequence of events composes the input provided. Each event has an associated time of occurrence. Given a set E of *event types*, an *event* is a pair (A, t) where $A \in E$ and t is the time of the event. An *event sequence* $\mathbf{s}$ on E is 3-tuple (s, T_s, T_e), where $s = \{(A_1, t_1), (A_2, t_2), ..., (A_n, t_n)\}$ is an ordered sequence of events such that $A_i \in E$ for all $i = 1, ..., n$, and $t_i \leq t_{i+i}$ for all $i = 1, ..., n-1$. T_s represents the starting time and T_e the ending time with $T_S \leq t_i < T_e$ for all $i = 1, ..., n$.

The concept of *interest* for a *frequent episode* is given by how close in time the events of an *episode* must arise. The user is able to define the width of the *time window* within the *episode* must occur. Formally stated, a *window* on an *event sequence* $\mathbf{s} = (s, T_s, T_e)$ is an event sequence $\mathbf{w} = (w, t_s, t_e)$, where $t_s < T_e$ and $t_e > T_s$, and w consists of those pairs (A, t) from s where $t_s \leq t < t_e$. The time elapsed $t_e - t_s$ is designated as the width of the window w.

An *episode* is a partially ordered collection of events occurring together (Mannila et al., 1997) and can be described by *directed acyclic graphs*. Depending on the partial order of the set of nodes, an *episode* can be either parallel, serial or injective. When specifying *sequential patterns*, the notion of containment was defined. In here, there is a similar idea but designated as subepisodes.

β is a subepisode of α if all the *event types* in β appear in α as well, and if the partial order on the *event types* are the same for each corresponding *event type* in both episodes. For example, (A $\rightarrow$ C) is a subepisode of the serial episode (A $\rightarrow$ B $\rightarrow$ C).

The notion of *frequency* of an *episode* assume a similar meaning as *support*.

It is defined as the fraction of all fixed-width sliding windows over the data in which the episodes occurs at least once (Mannila et al., 1997).

Algorithm 4 shows an high-level representation of the proposed algorithm named WINEPI. The $findFreqEp$ performs a breadth-first search in a level manner in the set of episodes ϵ following the subepisode relation in order to obtain all frequent episodes.

input : Set of event types E, an event sequence s, a set of episodes ϵ, window width w, a frequency threshold $minfr$, confidence threshold $minconf$
output: Episode rules that hold in s with respect to win, $minfr$ and $minconf$

```
1 /* Find frequent episodes */
2 F = findFreqEp(s, win, minfr);
3 foreach episode α : F do
4     foreach subepisode β : α do
5         if fr(α)/fr(β) ≥ minconf then
6             Output the rule β → α with a confidence of fr(α)/fr(β);
7         end
8     end
9 end
```

Algorithm 4: WINEPI algorithm

2.4 Conclusion

This chapter presented some of the research that has already been performed on either field of CSM and TDM. Some Data Mining algorithms that are able to detect discrimination patterns and to extract temporal patterns have been introduced. RCS is going to be the starting point of the proposal and it is intended to bring to the table the readability already given by the rules' syntax beside the detection of all significant CSs present in the data.

Next chapter, the whole proposal is presented step by step stating each application mission with the set of temporal patterns developed, how they are

extracted and exactly which kind of information it is pretended to be revealed by each one of them.

Chapter 3

Proposal

The proposal can be summed up in a 3-step process that occur in a serialized manner. Figure 3.1 represents the whole process and how each individual step is related, showing the output that is produced at each stage which serves as input for the next step.

RCS is the first operator in the chain. It is the algorithm included in CAREN (P. Azevedo, 2012) for discovering CSs in any given dataset. This will be used in order to obtain the contrasts at each observation (single period of time considered). Having individual datasets for each period, it will be executed as many times as the number of periods. For each execution, an Comma-separated values (CSV) output file that contains all the contrasts found and related information like group support values, *p-values*, among others will be produced.

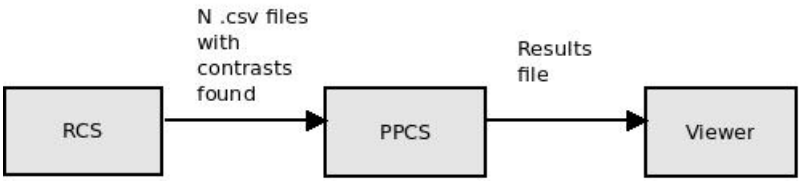


Figure 3.1: Process overview

PPCS was developed in order to process the set of output files produced by RCS at each period and to yield the temporal patterns that occur in them. The results are then printed out to the screen or written to a text file. An additional file is produced in order to be used by the *PPCS Viewer*.

The Viewer emerges as an optional but recommended manner to interpret the output given in the last step. Due to fact that there is some inability to interpret the results given in a textual format (at least not in an easy and intuitive manner), a graphical tool (*Viewer*) was developed in order to surpass those difficulties. It makes use of graphical representations like Histograms, filtering and searching features that significantly improve readability and increase the user interactivity.

3.1 Rules for Contrast Sets

This algorithm has already been introduced in section 2.2.3 and it has been explained in detail how contrasts are derived, which techniques are applied and how to interpret the results given. In here, there is a shift to another perspective and RCS is seen as a part of the process.

It is assumed that each period of time to be considered consists of one single dataset. That implies that some pre-processing might be necessary in order to break down a single dataset in parts according to the temporal unit that is selected (day, month, year, decade, etc.). For example, if there is a dataset with the transactions of a company in one specific year and a monthly approach is decided as the time unit of interest, then the dataset is required to be split in twelve parts, one for each month.

Refer to section A.1 for specific information on CAREN usage and options provided for the task at hand. It is crucial to guarantee that each CAREN execution is done with the same parameters for all periods and the results are

exported to the required *.csv* format.

Using CAREN for discovering CS is of major benefit since it is easy to use, portable and allows knowledge extraction in a friendly format with one contrast per line.

3.2 Post-processing Contrast Sets

Let's proceed by describing how to expand the CSM into a Temporal Data Mining task. First, the temporal patterns introduced by this proposal will be addressed, their intents and how valuable can they be in a real-life scenario. Other parts that compose this extension and the underlying decisions that were taken will be exposed as well.

Details regarding the implementation like how is data collected and maintained in memory, how patterns are discovered and organized are object of some scrutiny.

To close up the section, we check up on how results are shown and discuss its pros and cons, setting up the motivation for the development of another piece of software and last part of the process - the Viewer.

3.2.1 Comparing Contrast Sets from different periods

Despite being a self-sufficient form of rule, CSs require that a measure of interest is defined. This will enable the contrast comparison from different periods and will be able to contribute for patterns finding. That measure would ideally capture the contrast own strength at each period. Its evolution along the timeline would reveal if that specific contrast got "stronger" or instead got "weaker".

The support of a CS was the first approach in trying to obtain this measure. However, it was deemed insufficient since it only provides the antecedent

representation in the dataset and no description about the groups being contrasted. There can be a contrast in which there is a significant difference in the groups being contrasted even with a low support and the opposite can happen too (i.e. an high support could yield only a small difference between the contrasting groups).

The support of a CS was not the answer for the problem but it gave a nice hint to solve the issue. By justifying how the support of a Contrast Set was not a good solution, it contributed to focus in what it seemed the right direction: difference between groups in contrast.

Other measures could be considered like the number of observations of each group and their own support. The number of observations is not ideal since it is always dependent on dataset size. A difference of a hundred observations in the groups is significant if the dataset being used has a small size but that kind of difference might be insignificant in a large dataset with millions of entries. Support values for each group are relative values (proportions) and do not suffer from that issue.

Supdif remains as the only and obvious choice and it indeed answers successfully the questions posed before. It can act as a measure of interest to gauge the strength of a contrast (bigger the difference, stronger the contrast). Observations regarding how the contrast evolved along the time can be done with ease and some patterns involving this notion will be presented next.

3.2.2 Patterns of interest

From the contrasts present at each single point of time, the goal is to find patterns that can somehow express how some contrast has evolved along time. Those results are presented in the form of temporal patterns.

Growth and Shrink

The first patterns relate to the widening and narrowing of the support difference of the involved groups in consecutive periods. The issue here revolves around the quantification of how much does the contrast needs to grow or shrink to consider it a significant change. It is clearly dependent on the context involved. A 1% growth might be very significant in one scenario but absolutely meaningless in another one. This definitely imposes the requirement of some input from the user which should be able to discern the adequate value due to its context knowledge.

This threshold is called the Significant difference (sigdif) and operates much like *support* and *confidence*. If the difference from one period to the next is greater (in module) than the *sigdif* value then the variation is deemed as significant and should be reported.

With this threshold, the first two patterns appear and they are called *Growth* and *Shrink*. They are the dual of each other with the first referring to the situation when some contrast has its Support difference between contrasted groups (supdif) grow bigger than the sigdif value from the period N to the period $N+1$ whereas the second is the exact opposite.

Consider this example with one contrast appearing in two consecutive periods represented in temporal order:

```
1) Obs =  000657 | 000541  Gsup  = 0.78308 | 0.36019
p = 1.3948317823E-089  phi =   0.40569    pos=C >> pos=G
Sup(CS) =  0.53962           <---     3p%=[0 : 0.10]

2) Obs =  000687 | 000305  Gsup  = 0.76080 | 0.16433
p = 9.4731979393E-209  phi =  0.58324    pos=C >> pos=G
Sup(CS) =  0.39741           <---     3p%=[0 : 0.10]
```

Let's assume we are dealing with a 0.05 *sigdif* threshold. Next, we need to calculate the *supdif* for each period:

- Supdif (p1): $0.78308 - 0.36019 = 0.42289$
- Supdif (p2): $0.76080 - 0.16433 = 0.59647$

The *supdif* value has increased and in this case, we are facing a *Growth* pattern because it has grown 0,17358 $[Supdif(p2) - Supdif(p1)]$ which is over the *sigdif* threshold defined. If these two contrasts swapped their temporal order, we would be seeing a *Shrink* pattern instead.

These two patterns assume a important role, because they alert the end-user to a relevant spike in a contrast by moving to the next period. This change highlight that the groups being contrasted suffered some kind of modification for that specific antecedent and that change might be potentially useful for the end-user. This might help to locate some specific contextual phenomenon that occurred at that time and thus enable him to establish some possible relation of cause-effect.

Spring Up and Fade Out

After *Growth* and Shrink patterns, two other patterns came up one opposite to another, much like the two listed above. This time, the *sigdif* value and concept is not present but instead the goal here solely involves the appearance and disappearance of a contrast along the time line.

For example, let's imagine that a contrast is found for period N and $N+2$ but not for period $N+1$. This "hole" should immediately make the analyst query what happened at that moment and getting to know exactly why there is no contrast might entail strategical and valuable information.

From the period N to period $N+1$ we face the disappearance of the contrast and to that kind of pattern the designation *Fade Out* was selected.

That same contrast arises again from period $N+1$ to period $N+2$, an example of an occurrence of a *Spring Up*.

Flip

The last pattern is the *Flip*, and the name selected is well representative of its nature because of the "180º turn" notion that this pattern entails. Let's consider that for some antecedent there are two groups being contrasted, A and B. At some point in time, the contrast $A >> B$ exists but a few periods later this contrast disappears and gives place to $B >> A$. Hence the name Flip because the contrast was turned around.

Due to its specific nature, the *Flip* was the less frequent temporal pattern. Nonetheless, it is as important as the other patterns since it identifies specific situations that even being less frequent might bring information as significant as the information revealed by the other patterns.

Table 3.1 summarizes all the patterns introduced in this section.

Table 3.1: Summary of Temporal patterns introduced

Pattern	Description
Growth	From one period to the next the *sigdif* of a contrast grows more than the threshold defined
Shrink	From one period to the next the *sigdif* of a contrast shrinks more than the threshold defined
Spring Up	A contrast appears in a period when it was not present in the preceding one
Fade Out	A contrast disappears in a period after being present in the preceding period
Flip	Contrast that swapped its directionality along the time line considered

3.2.3 Stability measure

Apart from the patterns developed and presented in section 3.2.2, the lack of a global mean to evaluate a contrast motivated the development of this measure. The patterns introduced with exception of *Flip* operate in consecutive periods (i.e. locally) and do not allow to categorize or obtain the general behaviour of a specific contrast in its whole lifetime.

The existence of a numerical value that could gauge the variability of a contrast would provide an easy and intuitive manner to understand without getting into details how the contrast evolved, if it was object to frequent abrupt changes or instead if it has remained relatively stable in all considered periods.

This measure can also hold value for the user as a factor in discarding some contrasts. That is obviously dependent on his intents. let's imagine stable contrasts are not what he is currently seeking. Hence, contrasts patterns that do not yield a significant variation are discarded.

To achieve its purpose, this stability measure will be based on the following two premisses:

- The maximum score or value will be given to a contrast that appeared in all the periods considered and did not suffer any significant variations (no *Growth* or *Shrink* patterns);
- Any pattern found will contribute to lower the score since they translate significant variations that affect what we consider contrast stability;

The proposed formula for this measure that abides by the remarks stated above is the following:

$$Stability = \frac{T - \frac{P}{2}}{N - 1} \tag{3.1}$$

N represents the number of periods considered. T stands for the number of consecutive periods with contrasts found. P is the number of *Growth* and *Shrink* patterns found in the whole time line.

In the denominator, $N-1$ simply represents the number of transitions present in the periods considered. Best case scenario, $T = N-1$, which means a contrasts has been found for every single period. Thus, it becomes evident that Stability varies from 0 to 1. If there are never two contrasts found in consecutive periods then $T = 0$. Consequently Stability $= 0$, which seems adequate since there is absolutely no consistency as contrasts that appear in one period immediately disappear in the next one.

Let's now check the premises stated. For the first one, we know that maximum score possible is 1. If there is a contrast for each single period, then $T = N-1$, if there is no *Shrink* or *Growth* patterns found then $P = 0$ so the formula simplifies to $T/(N-1)$. However, since the numerator and denominator represent exactly the same value, one is obtained as expected.

The second premise is also fulfilled as *Growth* and *Shrink* patterns are subtracted from the T variable by a factor of 0.5, punishing the variations in *supdif*. *Fade Outs* and *Spring Ups* are not explicitly considered in this P variable. Nevertheless, they are affecting the score as the contrast is appearing and disappearing, thus affecting T.

Let's verify this concept in the following example, with a *sigdif* of 2:

```
Antecedent:  dwage = 2
Contrast: sex="female" >> sex="male"

Period 1 (2002) : Not found!
Period 2 (2003) : Supdif -> 0,70
Period 3 (2004) : Supdif -> 0,78
Period 4 (2005) : Supdif -> 1,13
```

```
Period 5 (2006) : Supdif -> 2,26
Period 6 (2007) : Supdif -> 2,60
Period 7 (2008) : Supdif -> 3,22
Period 8 (2009) : Supdif -> 5,76

Patterns:

- Spring Up from period 1 (2002) to period 2 (2003).
- Growth from period 7 (2008) to period 8 (2009) with a
difference of 2,54
```

In this example $N-1=7$, $H=6$ because there is no contrast at the first period and since there is one *Growth* pattern, $P=1$. Applying the formula we get:

$$Stability = \frac{6-\frac{1}{2}}{7} \Longrightarrow Stability = 0,79\ (2\ decimal\ places) \tag{3.2}$$

3.2.4 Implementation

In this section, the principal decisions taken for the development of PPCS, the middle block in figure 3.1, will be described from the reading of CAREN output files until the results file produced by this application.

Java was chosen due to its portability, multi-threading capabilities, easy of use and clear Application Programming Interface (API) for *Collections* and file I/O. Furthermore, this option provides a smooth coupling with CAREN, also a *Java* based software.

The application can be summarized in a high-level, simplified pseudo-code listed in algorithm 5:

```
   input  : files F, sigdif S
   output: results R
 1 R := ∅;
 2 D := ∅;
 3 validateUserInput();
 4 foreach file ∈ F do
 5     D += insertIntoDataStructure(file);
 6 end
 7 foreach antecedent ∈ D do
 8     R += findFlip(antecedent);
 9     foreach contrast ∈ antecedent do
10         R += findPatterns(antecedent, contrast, S);
11         R += calculateStability(antecedent, contrast);
12     end
13 end
14 Return R;
```

Algorithm 5: PPCS pseudo-code

The set of files (F) produced by CAREN will be read and its contents inserted into the data structure. Then, the list of antecedents (A) and contrasts (C) will be traversed in order to find Patterns and calculate Stability. This algorithm has a time complexity of $\Theta(|F| + |A| \times |C|)$.

Multi-threaded environment and data structure

Multiple files, each one with a great number of contrasts constitute a problem with considerable size. This requires an adequate approach to ensure the best response time is achieved.

Multi-threaded programs that are properly designed make better use of resources available, reducing computation time and improving performance as long as the program logic can be split in threads that execute concurrently (Goetz & Peierls, 2006).

By having a thread triggered to process each file, performance can be increased several times by making use of all available resources which could not be achieved by a serial application. The only drawback besides a more complex

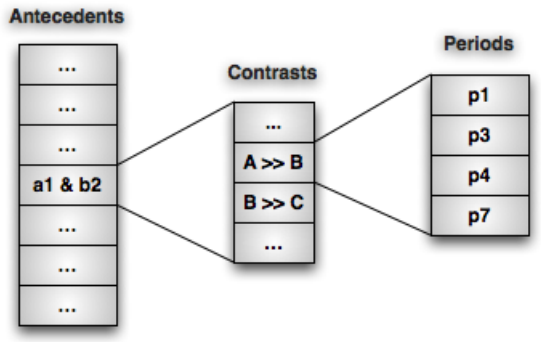


Figure 3.2: Representation of the data structure used by PPCS

programming model is that the data structure selected to hold all information has to be properly synchronized to avoid the well-known concurrency problems like *deadlock* or *starvation*.

Data structure selection assumes a crucial importance as it will encompass a great number of contrasts. An important requirement is to obtain a fast access, retrieval and insertion of information. The atomic unit that will compose the data structure is the Contrast class, which has a specific contrast information plus the period it originates from.

With help from the Java Collection Framework (JCF), a data structure was obtained and and it is represented by figure 3.2. Each atomic unit represents a period and is indexed by antecedent and contrast, much like two layers of indirection which organized them in their natural hierarchy.

One major advantage of this data structure relies in the organization of the contrasts. For certain tasks it is required to access all contrasts that share the same antecedent (for example, for finding Flips). For others it is only needed the data present for one specific antecedent and contrast. this situation will be made clear when we approach the pattern finding stage in a section below.

This hierarchy organises the data collected and enables a fast retrieval since we can get to any contrast by having the antecedent and contrast strings combination and that is achieved in constant time $\Theta(1)$. The insertion is also

done in constant time as well for the *Collection* used (*HashMap (Java Platform SE 6)*, 2011).

The only issue is that neither of the *Collections* used are synchronized and due to the multi-threading scenario envisioned we need to guarantee that concurrency pitfalls are avoided and no data is lost despite many threads accessing and inserting in this shared resource. Locking is one synchronization solution but requires some careful thought on exactly what needs to be locked and in which situations.

Locking the whole data structure would make insertion serialized and only one thread would be inserting at a time but due to the data structure organization and hierarchy we can have a finer grain. Threads inserting data at different places of the data structure (contrasts with different antecedents for example) do not affect each other and should be allowed to insert at the same time. The code ran by each thread for insertion is listed in listing 3.1 where the variable *list* is our data structure.

Listing 3.1: Code for inserting into the data structure

```
public void add(String ant, String cons, Contrast r)
 {
  HashMap<String, ArrayList<Contrast>> lr = null;
  ArrayList<Contrast> ar = null;

  // Insertion of new antecedent
  synchronized(list)
  {
    if(!list.containsKey(ant))
    {
      lr = new HashMap<String, ArrayList<Contrast>>();
      list.put(ant, lr);
    }
```

```
    }


    if(lr == null) lr = list.get(ant);

    //Insertion of new contrast
    synchronized(lr)
    {
      if(!lr.containsKey(cons))
      {
        ar = new ArrayList<Contrast>();
        lr.put(cons,ar);
      }
    }

    if(ar == null) ar = lr.get(cons);

    // Insertion of new period for existing contrast and
       antecedent
    synchronized(ar)
    {
      ar.add(r);
    }
  }
```

Three *synchronized* blocks appear in the code, and no more than one thread is allowed to execute inside one of this blocks at once. To check if an antecedent is already present we need to lock the whole structure and that is the more restrictive lock used. That has to be done in order to avoid a race condition when two (or more threads) want to insert the same antecedent (the one that executes later would overwrite the first one and data would be lost).

Similar scenario occurs when inserting the same contrast by different threads but in here it is only required to lock the inner *HashMap* and threads that

insert in other spaces of the data structure can execute at will and are not blocked. The insertion of a new period in an already existent antecedent and contrast, requires the *ArrayList* to be locked for insertion. This method of locking is thread-safe (can be executed by multiple threads) and no data is lost.

Processing and pattern finding

After loading the all the data into the data structure. Mining for temporal patterns can occur by multiple processing threads.

Each thread will be responsible for the set of periods that share the same antecedent and contrast and will discover all patterns that emerge from that contrast (except *Flip* which is done at antecedent level). Stability is calculated along this process. The main thread will wait for all its child threads to stop execution and collect all the results from them, starting to construct the output according to user directives.

Flip finding, which differs from other patterns, is performed at each antecedent in the main thread of execution and is achieved using some string comparison. The listing 3.2 contains the code responsible for that task.

Listing 3.2: Code for finding *Flips*

```
// Searches for flips in the antecedent given by parameter
// type -> 0 (-flips mode)  ||   type -> 1 (-full mode)
public int checkFlip(String a, int type)
{
  Set<String> aux1;
  // Set of contrast strings that share this antecedent
  aux1 = rules.getLista().get(a).keySet();
  String [] ret = new String[aux1.size()];
  aux1.toArray(ret);

  // If there is only one contrast, no flips can happen
```

```
    if(ret.length < 2) return 0;

    // Will compare all the strings combinations for flips
    for (int i = 0; i < ret.length - 1; i++)
    {
      for(int j = i + 1; j < ret.length; j++)
      {
        // Checks if one string is the opposite of another
        if(isFlip(ret[i],ret[j]))
        {
          printFlip(type,ret[i],ret[j],a);
          return 1;
        }
      }
    }

    return 0;
  }
```

The contrasts from the specific antecedent are extracted and then checked whether there is one opposite pair. In the positive case, the *printFlip* function is called which is responsible for building the output with regard to the *-mode* flag.

It is important to highlight that one opposite pair of contrasts might contribute with more than one Flip pattern. The $A >> B$ contrast might become $B >> A$ which is one *Flip* but that $B >> A$ contrast might revert back again to $A >> B$ in a future period and that is already another distinct Flip. Consequently, the *printFlip* recursively finds all the *Flips* and arranges the output accordingly.

For the remaining four patterns, the thread will iterate over the contrast periods discovering them along the way. The listing 3.3 shows how *Growths* and *Shrinks* patterns are found.

Listing 3.3: Code for finding *Growths* and *Shrinks*

```
int ord_sd = 0; double supdifant = 0.00;
double supdifcur = 0.00;

for(Contrast r : al)
{
  // It's the immediate next period
  if(ord_sd == r.getOrd()-1)
  {
    // If it's the first period
    if(ord_sd == 0)
    {
      ord_sd = 1;
      supdifant = r.getGsup1()-r.getGsup2();
    }
    // If not
    else
    {
      // There was a contrast in the previous period
      if(supdifant != 0)
      {
        supdifcur = r.getGsup1()-r.getGsup2();
        // Growth
        if(supdifcur - supdifant >= sigdif)
        {
          pat = buildPatternGS("Growth", ord_sd, supdifcur-
              supdifant);
          patterns.add(pat);
          sdifs++;
        }
        else
        // Shrink
        if(Math.abs(supdifcur - supdifant) >= sigdif)
        {
```

```
            pat = buildPatternGS("Shrink", ord_sd, supdifcur-
                supdifant);
            patterns.add(pat);
            sdifs++;
          }

          supdifant = supdifcur;
          ord_sd++;
        }
      }
    }
    // It was not the immediate period after
    else
    {
      supdifant = r.getGsup1()-r.getGsup2();
      ord_sd = r.getOrd();
    }

  } // end of for cycle
```

This code cycles the whole periods found for a specific antecedent and contrast and finds the *Growth* and *Shrink* patterns with help from the variables: *ord_sd* (order of last found period), *supdifant* (*supdif* from the last period found) and *supdifcur* (*supdif* from the current period). These patterns only occur when two consecutive periods are present. In that case, *supdif* is checked against the threshold and in the positive case a pattern is found an added to the pattern list.

To discover *Spring Ups* and *Fade Outs*, all atomic instances that share the same antecedent and contrast are visited. A few tests are done in order to find the patterns. *Spring Ups* are the easier case, since we find one every time there is a contrast in a period which did not occurred in the previous one. *ord_su* variable permits to track that.

For *Fade Outs* we need to find gaps in between the periods. After finding one a contrast in a period, there is a *Fade Out* whenever a period or more with no contrasts occur. This is done using *found_fo* flag and *ord_fo* to keep track of the period expected to come next. Comparing this to the next period allow us to discover if there is in fact a Fade Out at that moment. Listing 3.4 shows how the code to extract these patterns.

Listing 3.4: Code for finding *Spring Ups* and *Fade Outs*

```
int ord_su = 0; int ord_fo = 1;
int found_fo = 0;

for(Contrast r : al)
{
  // Springs Ups
  // It's the next period (or first) -> No Spring Up!
  if(ord_su == r.getOrd()-1)
  {
    ord_su++;
  }
  // No contrast in previous period -> Spring Up!
  else
  {
    ord_su = r.getOrd();
    pat = buildPatternSUFO("Spring Up", ord_su);
    patterns.add(pat);
  }

  // Fade Outs
  // It's the next period (or first) -> No Fade Out!
  if(ord_fo == r.getOrd())
  {
    ord_fo++; found_fo=1;
  }
```

```
    else
    {
      // If a contrast has been found before but has now
         disappeared -> Fade Out!
      if(found_fo!=0)
      {
        pat = buildPatternSUFO("Fade Out", ord_fo-1);
        patterns.add(pat);
        ord_fo = r.getOrd()+1;
      }
      // First contrast to be found
      else
      {
        found_fo = 1;
        ord_fo = r.getOrd()+1;
      }
    }
  }

  // If there has been no contrasts in the last periods -> Fade
     Out!
  if(found_fo == 1 && ord_fo <= num_max_rules)
  {
    pat = buildPatternSUFO("Fade Out", ord_fo-1);
    patterns.add(pat);
  }
```

Note that described patterns are found in a single for loop (thread execution is $O(N)$ in the number of periods). The code has been split for each pattern for the sake of simplicity. Some controlling code has been omitted as well like checking the mode flag to avoid finding unnecessary patterns.

The last task performed by each thread is to calculate each contrast stability. Method *calcStab()* implements this calculation. Since this calculation

requires the number of patterns found, it is only computed at the end of the process.

Listing 3.5: Code for calculating Stability

```
// Stability calculation
public double calcStab()
{
  double stab = 0.0;
  // sdifs contains the number of patterns found by this thread
  double sdifs_r = sdifs / 2;

  // in_a_row -> number of consecutive periods with contrasts
     found
  stab = in_a_row - sdifs_r;
  stab = stab / (num_max_rules - 1);
  return stab;
}
```

3.2.5 Output and results

After all threads finish processing, all patterns found are extracted from their respective threads and the output is constructed depending on the parameters set by the user.

The *-out* flag specifies where the results are written (text file instead of *stdout*) and does not change any of the content or its disposition. Due to the fact that the probability of working with a problem with considerable size is pretty high, outputting these results to a text file seems a more reasonable decision most of the time.

The default mode (*-full* or *-f*) provides the most complete output while other flags just include some subset of the information provided by this mode. Listing 3.6 shows all the information given for one antecedent and all its

contrasts, including patterns, stability and *supdifs*. For other output options, please refer to appendix A.2

Listing 3.6: Output provided by *-full* flag for one specific antecedent

```
==================================================

Antecedent:  fgapg=]5.00 : 8.00]  &  dreb=[0 : 1.50]

==================================================

--------------------------------------------------

Contrast:  pos=G >> pos=C

Period 1 (40s_results) : Not found!
Period 2 (50s_results) : Not found!
Period 3 (60s_results) : Not found!
Period 4 (70s_results) : Not found!
Period 5 (80s_results) : Supdif -> 0.157
Period 6 (90s_results) : Supdif -> 0.092
Period 7 (2000s_results) : Supdif -> 0.061


Stability: 0,17


Patterns:

- Spring Up from period 4 (70s_results) to period 5 (80
    s_results);
- Shrink from period 5 (80s_results) to period 6 (90s_results)
    with a difference of -0.065;
- Shrink from period 6 (90s_results) to period 7 (2000s_results
    ) with a difference of -0.031;
```

```
--------------------------------------------------

Contrast:  pos=G >> pos=F

Period 1 (40s_results) : Not found!
Period 2 (50s_results) : Not found!
Period 3 (60s_results) : Not found!
Period 4 (70s_results) : Not found!
Period 5 (80s_results) : Supdif -> 0.127
Period 6 (90s_results) : Not found!
Period 7 (2000s_results) : Not found!


Stability: 0


Patterns:

- Spring Up from period 4 (70s_results) to period 5 (80
   s_results);
- Fade Out from period 5 (80s_results) to period 6 (90s_results
   );


--------------------------------------------------

Flips : Not found!

--------------------------------------------------
```

This approach has its merits but has some flaws as well. It feels a little outdated to modern standards and has little visual impact. To compare different contrasts, some scrolling activity in the file might be needed and that does

not facilitates the analysis because locating a specific contrast we are seeking might not be as immediate as desired. These cons motivated taking a step further for developing a visual application in order to provide a more simple, attractive and intuitive manner to navigate through the results obtained.

3.3 PPCS Results Viewer

The last component of the process is the *PPCS Results Viewer*. This application intends to be an alternative to the results expressed in a textual form. It makes use of visual representations and some features to perform different tasks that can enrich the analysis that was very hard to perform in the previous format.

Starting from the motivation that led us to develop the *Viewer* as a compliment to PPCS, implementation details and features developed will be addressed regarding their main goals that generally were not possible (or not as optimal) in the textual output obtained before.

3.3.1 Motivation for another application

The Viewer was ever seen as a complement, an alternative for helping the user to grasp the information from another perspective with different options provided to accomplish that goal. Nonetheless, it provides a significant increase in terms of readability and understanding of the contrasts previously found through the graphical features used.

In On-line Analytical Processing (OLAP) scenarios, managers and stakeholders from an enterprise are just involved in the data analysis process by reading the reports created and operating in a dashboard querying the information present in the Data Warehouse (Golfarelli & Rizzi, 2009). This example can be extrapolated to our process and shows a specific user pro-

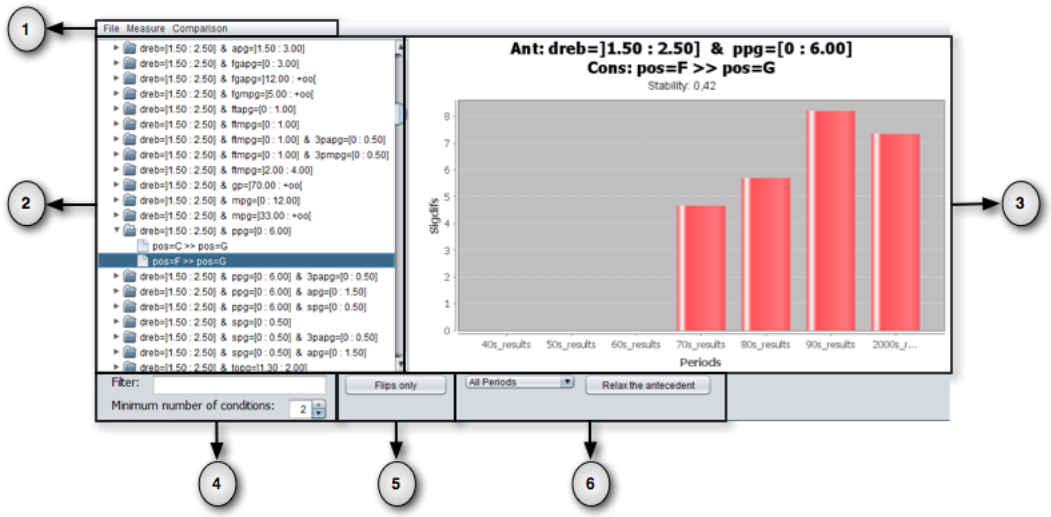


Figure 3.3: *Viewer* main frame decomposed by areas

file which would only operate with the *Viewer* and not with *PPCS*. Thus, it justifies the application independence.

For developing the *Viewer*, there is no reason to stand away from the *Java* universe. Swing, provides an API for building GUI for *Java* programs. The general benefits of developing in *Java* stated in section 3.2.4 fit in here as well and *Swing* provides all the visual components needed to implement the envisioned features.

3.3.2 Data import

The first consideration has to be how to bridge the gap from PPCS to this point. The *Viewer* needs to be fed with information to be able to display it and let the user interact with it. The *Viewer* can specifically select data from the results produced by PPCS.

When this file is appropriately selected (see appendix A.3), the main frame containing all the features available is constructed and it is represented in figure 3.3.

3.3.3 Features

Table 3.2 relates each decomposed area from figure 3.3 to its description in order to facilitate referring certain features provided by the interface developed.

Table 3.2: Designation of each area composing the GUI

Area	Designation
1	Menu Bar
2	Tree Model with CS
3	Graphics panel
4	Filtering features
5	Patterns filters
6	Relaxation feature

Tree Model

In this area, every CS found is contained in this tree-like representation. It follows a 2-level approach much similar to the one taken by the data structure used 3.2. This allows for a better organization of the CS and makes navigation more intuitive.

Immediately after importing the *resultsImp.txt* file, this tree is built with all the information collected. All the antecedents with discovered contrasts are shown. The number of antecedents will probably be such that it will be impossible to see them all in one go and the inclusion of a vertical *scroll bar* was required. This could indicate a possible burden in locating a specific antecedent and for that matter, some features have been developed to facilitate this action and are detailed in section 3.3.3.

This model provides two distinct interactions: Expanding an antecedent to see the contrasts that exist within it and clicking each tree item. In figure 3.3 there is one expanded antecedent ($dreb =]1.50; 2.50]$ & $ppg = [0; 6.0]$) and that specific antecedent has two contrasts ($pos = C >> pos = G$ and $pos = F >> pos = G$). The second interaction involves clicking in either a contrast or an antecedent. That action is reflected in the *Graphical Panel* where information of the item clicked is represented. More on this is detailed in the section below.

Graphical Panel

This is the area in the interface where information stated before is displayed. What is represented is dependent on what is selected in the *Tree model*. This entails the close relation between both components and how they depend on each other.

One major requirement that required a clear answer was how to represent a contrast in a graphical manner that would contribute to easily identify the following situations:

- Periods with contrasts and periods without;
- The patterns found;

A chart like an *histogram* can accomplish these objectives by having all periods represented in the *X-axis* and the contrast *supdifs* in the *Y-axis*. That required choosing a *Java* library which could display charts. *JFreeChart* (*JFreeChart*, 2012) came up as a solid solution due to the well-documented API, being open-source and mainly due to the simplicity for displaying a chart in a *Swing* application. Beside these benefits, it allowed us to incorporate another features in the charts like exporting them in different format, printing, zooming in and out with little to no effort.

Figure 3.3 reveals the histogram for the selected contrast. In the *X-axis*, all the periods are represented from the first to the last in their temporal order. In this specific case, contrasts are found from the *70s_ results* period onwards. *Supdif* comes represented in the *Y-axis*. The title contains the antecedent and the contrast involved with the indication of the stability calculated. The first objective proposed is accomplished as periods with contrasts have a bar that represents the *supdif* for that same period. The periods without contrasts do not have any representation.

Regarding patterns, *Spring Ups* and *Fade Outs* are intrinsically related with this graphical representation, thus identifying these patterns become trivial by implication as well. *Growths* and *Shrinks* are now less stringent and more prone to the user own interpretation of the contrast at hand. The user can consider that an increase in *supdif* can be significant and the same growth might not be for a different contrast. This fits on a real case where not everything weighs the same from the user perspective. The inclusion of the *Growths & Shrinks* button allows for quickly filtering the contrasts that grow or shrink above a threshold introduced by the user (see section 3.3.3).

These differences are visually identified due to the bar sizes being proportional to their *supdif*. *Flips* are, like seen before, a special case seen at antecedent level and are addressed in section 3.3.3 as well.

The example shown in figure 3.3 is for a specific contrast but like stated before, clicking in the superior tree level (the antecedent) also affects the *Graphical Panel*. An example of this is shown in figure 3.4.

The histogram remains similar except for the fact that the whole CS from the antecedent selected is represented. This representation is nothing more than the agglomeration of each individual contrast in a single *histogram* which gives a better overview and permits an easier comparison between them. A legend matching the bar colors to each contrast to distinguish them is provided

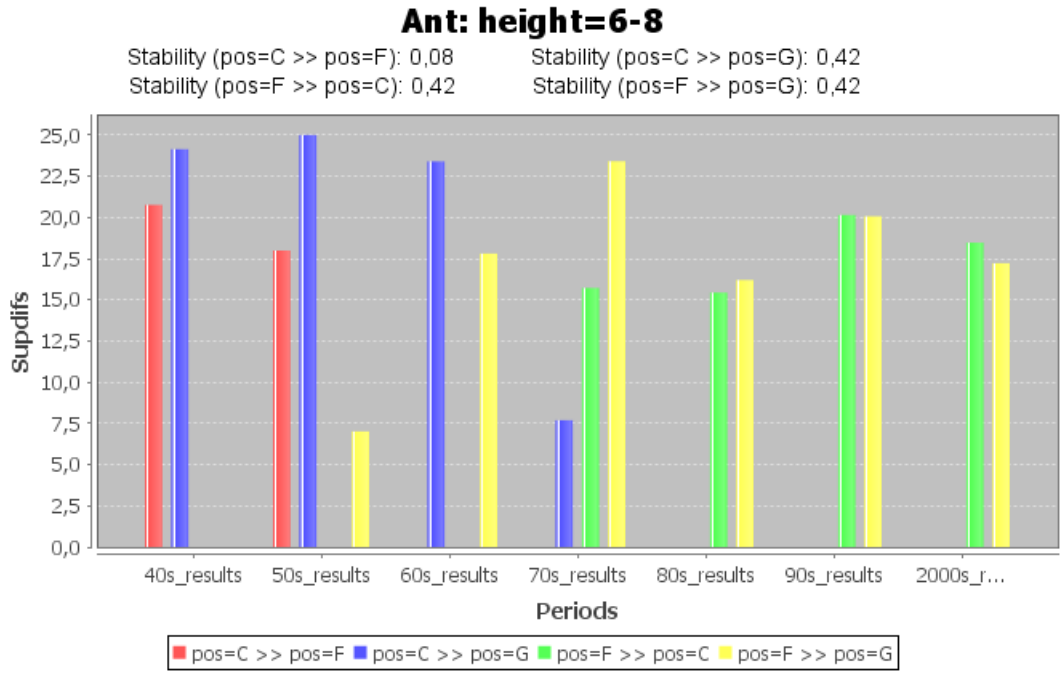


Figure 3.4: Histogram for a specific antecedent showing all contrasts contained

as well as the *stability* value for every single contrast.

There is still one more modification allowed to the *Graphical Panel*. That is changing the *Y-Axis* of each *histogram* to *p-values* instead of *supdifs* by clicking in the Measure item in the *Menu Bar*. This contributes for a different and more technical view to verify the "strength" of the statistical test used to reject the null hypothesis and thus finding the contrast (Fisher's exact test used in RCS). Figure 3.5 represents the *histogram* of one contrast using *p-values* instead of the default *supdif*.

The scale used for the *histogram* became a issue because some of the *p-values* used diverged from each other in multiple orders of magnitude. Some *p-values* like 1×10^{-200} were not even visible next to others with only a few decimal places. To surpass this limitation, it was decided to represent not the exact *p-value* in the *Y-axis* but its order of magnitude instead. This way, the scale problem disappears. However, in case the user needs to know the exact *p-value*, hovering over on one of the *histogram* bars yields a tooltip with

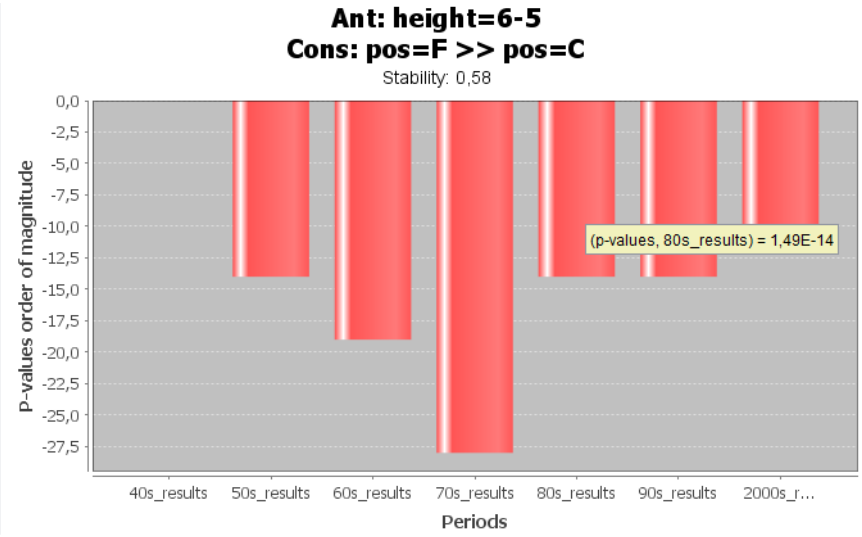


Figure 3.5: Histogram for a contrast with *p-values* order of magnitude in the *Y-axis*

the exact value. Example is figure 3.5 where the *p-value* of $80s\_results$ is 1.49×10^{-14} (the order of magnitude is -14 and that defines the bar height).

Filtering features & patterns finding

Tree Model usually has a considerable number of items present and locating some specific antecedent might involve some scrolling effort which is not desirable. Two features have been implemented in order to improve that situation.

The first one relates to a typical filter as indicated by the label and a *text box* in which the user can type. This works as a filter over antecedents if the introduced text matches. The items are filtered as the user is typing, avoiding the need for an extra click and implementing a button.

The other one discards antecedents which number of items are inferior to the number present in the *spinner*. This is useful for finding the complex antecedents which can reveal interesting relations.

Figure 3.6 has a filtering example. It is relevant to notice that both filters can be used in conjunction and in this concrete example, only antecedents

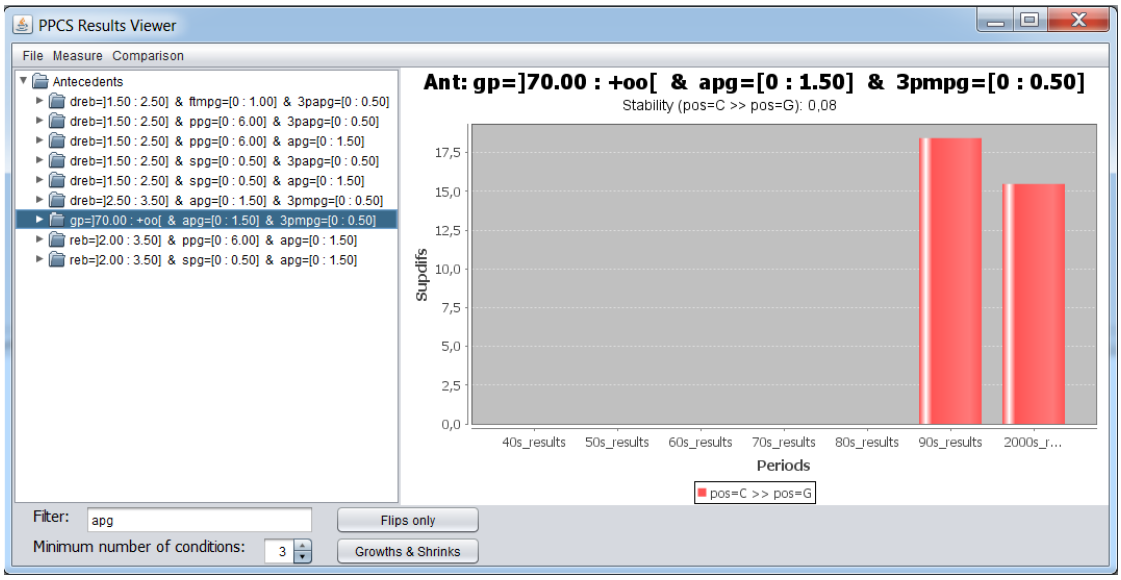


Figure 3.6: Filtering example

with the text *"apg"* with at least three items are shown.

Like stated in section 3.3.3, *Flips* are a distinct case. They usually appear just a few times and a mechanism to find them was developed in the form of a toggle button which, when pressed on, the *Tree Model* shows only the CS that contain at least one *Flip* pattern making them easily detected.

Figure 3.4 happens to reveal a *Flip* pattern when contrast $pos = C >> pos = F$ present in the first two periods (red bars) turns into $pos = F >> pos = C$ from the $70s\_results$ period onwards (green bars).

A button for filtering *Growths & Shrinks* was also introduced. This provides a chance to instantly detect the contrasts that suffer from abrupt alterations from one period to the following. To grant some extra flexibility to this filter, it is not based on the *sigdif*, the parameter provided to *PPCS*. Instead, a new dialog shows up requiring a value between zero and a hundred which will serve as the threshold. For example, if the value X is introduced, only the contrasts that the *supdif* has increased or decreased over that value in at least one period to the next will appear in the *Tree Model* area.

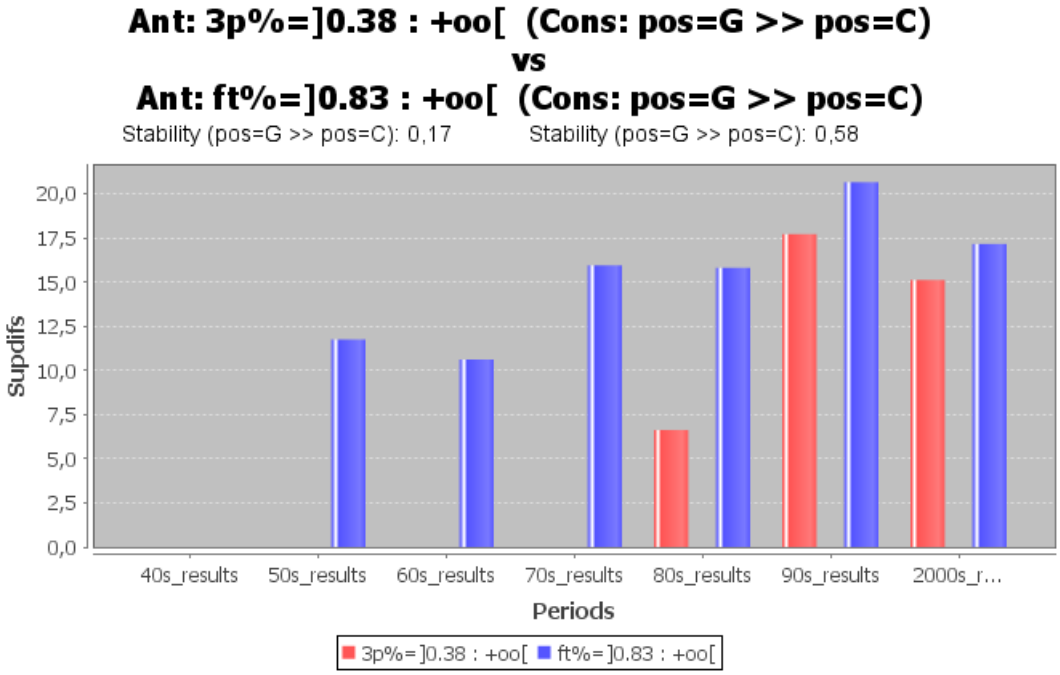


Figure 3.7: Histogram state after using Head to Head feature

Head to Head

The lack of a possibility to compare contrasts from different antecedents motivated the creation of this *Head to Head* feature. The objective here is as simple as letting the user to compare any pair of contrasts.

By pressing the *Comparison* item in the *Menu Bar*, a new dialog appears. This dialog contains a total of four *combo boxes*, two for the antecedents and two for the contrasts that are going to be part of the comparison.

When the selections have been made in the four *combo boxes*, the button becomes enabled (as long as they do not hold the exact same contrast). The user can press it to make the dialog disappear and return to the main interface with an updated *Graphics Panel* for the selection made. Figure 3.7 shows how the *histogram* is updated to reflect the selected comparison.

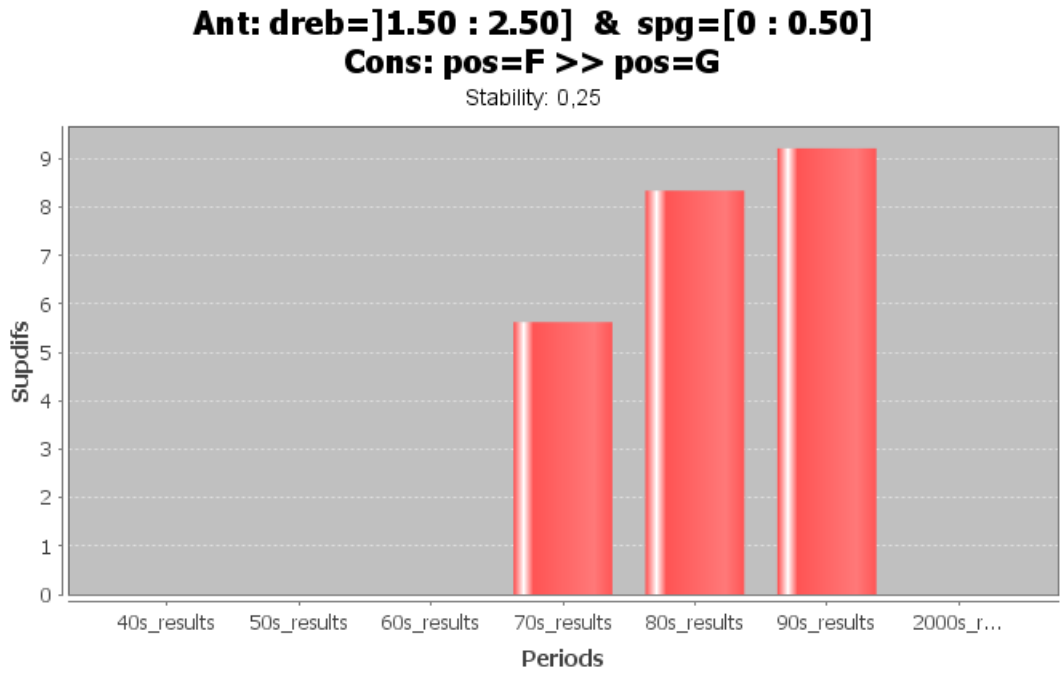


Figure 3.8: Example of a contrast with no occurrence in the last period

Antecedent relaxing

This last feature attempts to lead the user in finding a possible explanation to why some specific contrast was not discovered in a specific period. If an antecedent with minus one item than the antecedent being analysed, has a contrast in that specific period, an individual might conclude that the item removed may be the main reason for that event (or at least contribute to that). In figure 3.8, after three periods, the difference between the groups keeps increasing. Surprisingly or not, the contrast is not present in the last period. This is an example of a potential situation where this feature could be valuable for the user in order to discover why this phenomenon has occurred.

To perform this, the user can select from the respective *check box*, one period with no occurrence from the contrast selected or even select the option *"All periods"* as seen in figure 3.3. It should be noted that this button and *check box* are only visible when a single contrast (not an antecedent) is selected

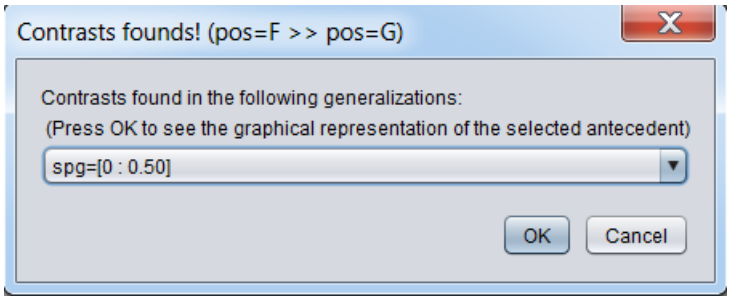


Figure 3.9: Dialog shown when the relaxation is successful

and the considered antecedent has at least size two in terms of items contained.

For a given antecedent, all its one step above generalizations are considered. For the selected period and contrast, a list of antecedent generalizations is constructed. The option *"All periods"* widens the test not only to one period but to all periods without contrasts. If the antecedent being tested has at least one contrast in a period where the more specific one did not, the more general antecedent is added to the list. Figure 3.9 shows the dialog after executing the relaxation for period *"2000s_results"* (there is only the item shown in the *combo box*).

By clicking the *OK* button, both antecedents are shown in the *Graphical Panel* much like the *Head to Head histogram* seen in figure 3.7 before. This yields an enhancement for user understanding. In this specific case, he might be led to conclude that the absence of a contrast in the last period for the antecedent of "$dreb =]1.50 : 2.50]$" is the main culprit for the unusual behaviour that is being analysed.

Still, there is the possibility of another outcome. This happens when no antecedent with one less item has a contrast for the selected period (or periods). In that a case, a dialog shows up and questions the user if he wants to find a generalization of that antecedent (of any size) that has a contrast for that specific period.

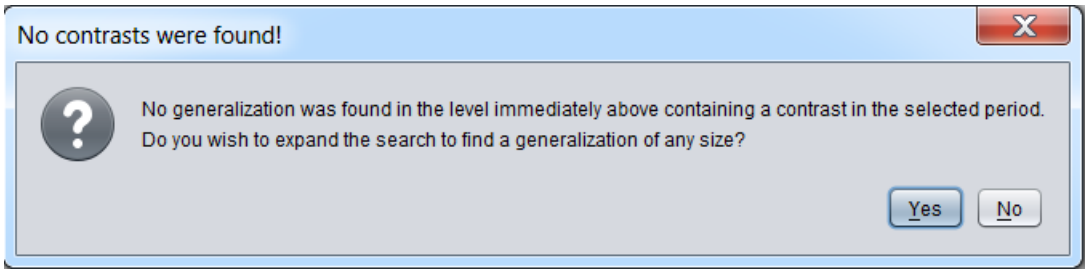


Figure 3.10: Dialog shown when the relaxation has no contrast in the level above

3.4 Conclusion

The whole process was exposed either in terms of applications and patterns developed. What has been proposed can not be validated until it is applied and tested with appropriate data. For that same reason, in the next two chapters, the technique will be applied to two different case studies in order to check its accuracy and usability in practice.

First scenario, involves data from working individuals from several enterprises and industries in Portugal. The main goal is to check how the gender of an individual affects elements like salary, education, etc.

The other case study is related to sports, more specifically basketball. The application of the developed technique intends to understand how each player position affects the defined attributes and if that has significantly changed over the years.

Chapter 4

Case Study #1 - Labour data

This first case study involves the study of the data collected from the Portuguese Ministry of Labour and Social Security for all employed individuals in the private sector ranging from 1986 until 2009. This data was provided to the economics and management department (EEG) of the University of Minho, and it was granted the right for use in this project after the signing of a contract that safeguards the data confidentiality and requires the clarification that the entity which collected the data is not responsible for the results presented in here and its own interpretation.

The chapter will start by stating the main goal of this analysis, what it is intended to be accomplished and how it relates to other work also done for this data. Next, will come the description of the attributes present in the data and which treatment they received until they were finally ready to be processed by the set of applications developed.

Finally, the results are inspected with particular attention to situations considered relevant and somehow surprising, focusing on the temporal patterns developed for this effect.

4.1 Objective of the analysis

Earlier studies were performed with this data, using econometric techniques like regression models. The aim was to derive relations between variables and perform forecasting. One interesting scenario that was being studied, was the discrimination of male and female workers in the labour world. The discrimination aspect blends well with the proposal developed that involves CSM, and since the data at hand involves a considerable time window, the proposal displayed in chapter 3 can be used for this purpose.

The defined group objective is the worker gender and contrasts are discovered in regard to this property. It will be intriguing to check how the Portuguese labour world has changed in the last few years in regard to sexual discrimination. The main goals for analysis are the following:

- Discover a set of situations that have considerably changed over the years;
- Understand the effects of the Portuguese population characteristics in sexual discrimination;

4.2 Data obtained

The available data contains a record for every individual employed in every single year they were bounded contractually to some Portuguese enterprises in the private sector. The set of attributes which seemed relevant and could contribute in finding interesting contrasts is presented in the table 4.1.

The presence of attributes with the year and the gender of each employee guarantee the application of this proposal as they will serve as the temporal period and the group to be contrasted.

The ownership of each firm is given by the *owner* attribute that can have

Table 4.1: Labour dataset obtained

Attribute	Description
idtrab	Worker ID
ano	Year (ranges from 1986 to 2009)
sexo	Gender
educ	Years of education
idade	Years of age
antiguidade	Years of tenure
tamanho	Number of workers in the firm
owner	Type of ownership (public, private or foreign)
loca1	Location of the firm (according to Nomenclature of Territorial Units for Statistics (NUTS) level 2)
inda1	Industry classification
salario	Log hourly wage real (ln €)
vendas	Log real sales (ln 1000 €)

one of three possible values: private, public or foreign. Regarding each company, comes the *loca1* attribute which contains the NUTS developed by Eurostat that divides the territory into statistical subregions for all the member states of the European Union. For this attribute, it is used the NUTS level two that divides the Portuguese territory into seven zones: North Coast, Center Coast, Lisbon and Tagus Valley, Inland, Algarve, Azores and Madeira.

Last, come the sales and salary attributes (*vendas* & *salario* respectively). They are represented in terms of the natural logarithm (logarithm to the base e) which will need to be discretized (see section 4.2.2).

The number of attributes selected is not impressive, but the dataset size is extremely considerable as there are an average of two million records per year which comprises a remarkably consistent context for analysis.

4.2.1 NULL values

One problem that is frequent when dealing with data is the absence of some values (NULL values). This dataset has some NULL values in a few records.

Since the percentage of these records is pretty insignificant, they are simply ignored.

With more influence than the negligible missing values in the attributes taken into consideration, is the total absence of records for these two years: 1990 and 2001. For some unknown reason, data for these years is inexistent, and this should be taken into consideration when defining the temporal periods and interpreting results from them.

4.2.2 Data transformation and discretization

To improve the contrasts readability, the only transformation done was in regard to swap some attributes codes for their respective values. For example, the *sexo* attribute which contains the gender, instead of the 1 and 2 coding, they were changed for their correspondent value, *male* and *female*.

CAREN algorithm has a limitation of not dealing with numerical, continuous attributes directly (it provides a module for discretization) for finding CS as they are treated as categorical by default. This affects some of the attributes in this dataset since they fall in the numerical category mentioned. These attributes are discretized into a finite number of categorical intervals to bypass this situation. Two different strategies were selected to deal with this situation: user-defined and equal-frequency discretization.

For age and education, the discretization was user-defined because the intervals selected hold specific meaning that are going to be more valuable than any ones obtained by automatic discretizations methods. For example, the education attribute present in the dataset holds the number of years of education of a certain individual. The intervals defined intended to represent the typical educational attainment in the Portuguese educational system. For example, $educ < 4$ corresponds to an individual that just had primary education and $educ = 10 - 12$ to classify someone who had secondary education

or dropped without fully completing it.

The remaining numerical attributes were categorized based on the notion of quantiles dividing the data into k equal-sized intervals. The difference from the age and education attributes relies on the fact that it was not as immediate to settle for intervals with proper meaning. Thus, it was decided to use intervals with equal frequency, automatically defined with no human intervention.

4.2.3 Temporal division

The current data holds a year attribute, and since there are no other temporal units, a smaller period can not be defined. This holds a total of 24 periods (2009 - 1986 + 1), or more specifically 22 since there are two years with no data (see section 4.2.1).

Since every year has a representative amount of dataset records and is the natural data microunit, there is no reason to group years together. Besides, observing the proposal performance with a considerable number of periods is of interest and differs from the approach which will be used in section 5.5.3.

For the application of the RCS algorithm, the dataset is split into multiple data files, one for each period (i.e. each year).

4.3 Results analysis

Having the data ready, the execution of CAREN for discovering CS is done for each period with a *support* of 0.05 followed by the PPCS with a *sigdif* of 0.01:

```
java -jar PostProcessingCS.jar  86.csv,87.csv,88.csv,...,
07.csv,08.csv,09.csv 0.01 -f -out
```

Regarding the objectives stated in section 4.2, in the following subsections different attributes or set of attributes present in the data will be inspected while detecting interesting and surprising events that clearly separate both genders. The effect of time in situations where the contrasts have significantly changed for each passing period will also be studied.

4.3.1 Education

The effects of education are the first item of analysis as it is such an evident element that reliably distinguishes the male and female groups in this labour setting.

Starting from individuals who have higher education, the contrasts found are present in figure 4.1. In the first eleven periods, no contrasts are discovered which could be indicative of a fair division in terms of gender at that time. However, since 1998, the contrast $sexo = "female" >> sexo = "male"$ appears for all following periods, and the gap between both sexes is clearly increasing for each passing year.

This confirms some reports which state that women pursue higher education more often than their male counterparts and the tendency keeps increasing at a steady pace.

As for workers that concluded secondary education or at least completed part of it, the contrast remains the same, $sexo = "female" >> sexo = "male"$. The difference here relies on the fact that this same contrasts appears for all years except 1986 and 1987. An increasing tendency is also displayed until 2002 (although not as accentuated) but stabilized from there.

Figure 4.2 reveals the contrasts found for individuals who concluded or dropped during basic education (excluding primary education). A first glance at it reveals a *Flip* pattern like stated in the results file:

```
[sexo="female" >> sexo="male"] (99) becomes
```

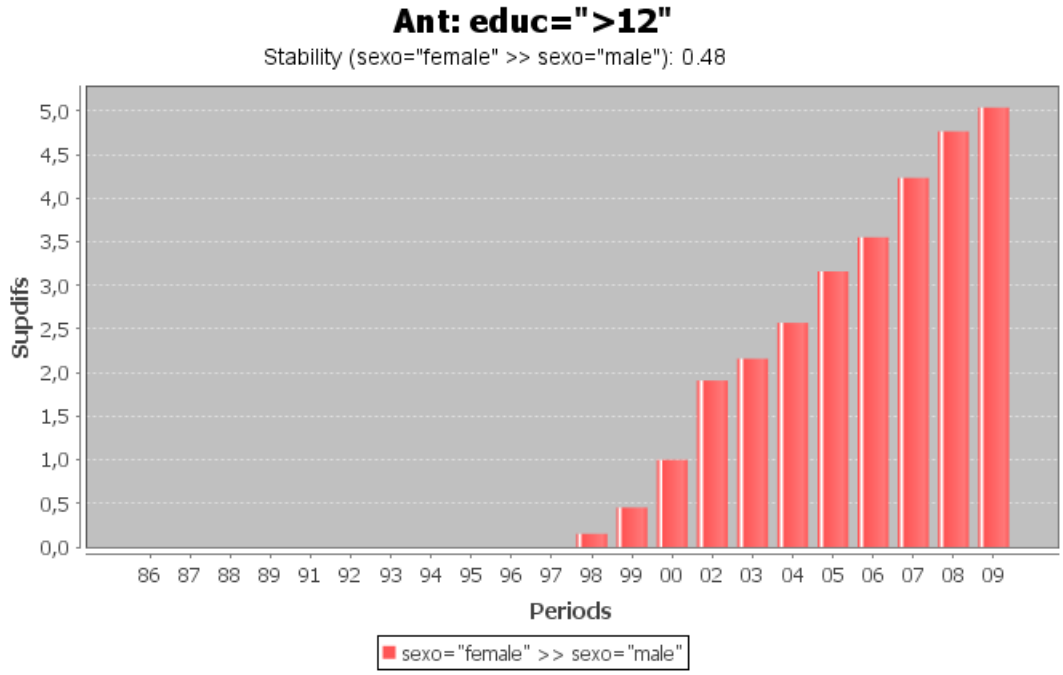


Figure 4.1: Contrasts found for individuals with higher education

```
[sexo="male" >> sexo="female"] (02)
```

This pattern indicates a significant change as women stopped being the dominant gender for this specific education and the throne has been assumed by men since 2002. Not only the masculine gender become the dominant one in this situation but the difference towards women is widening uniformly as seen by the crescent blue bars in the referenced figure.

Since the *supdif* between some of those periods is greater than the threshold defined as 1% some *Growth* patterns are derived in three of them:

```
- Growth from period 15 (02) to the period 16 (03) with
a difference of 0.012;
- Growth from period 17 (04) to the period 18 (05) with
a difference of 0.011;
- Growth from period 18 (05) to the period 19 (06) with
a difference of 0.011;
```

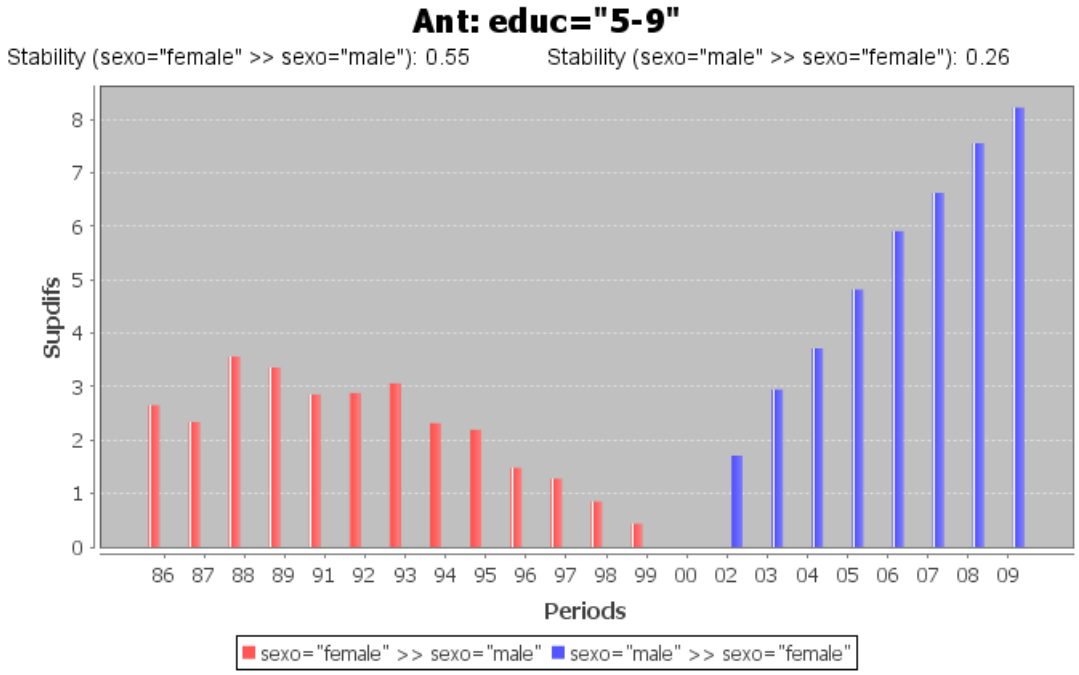


Figure 4.2: Contrasts found for individuals with between five and nine years of education

For the last interval, $educ < 5$, the contrast $sexo =$ "$male$" $>>$ $sexo =$ "$female$" was extracted for all periods and is pretty stable as the *stability* measure amounts to 0.98 (only a *Growth* pattern was derived once).

4.3.2 Location

The location of the employees firm (*inda1*), is one attribute which revealed interesting situations. The *Lisbon and Tagus Valley* region comprise the biggest entrepreneurial zone of the entire country and the contrasts discovered are shown in figure 4.3.

A *Flip* pattern is derived as the initial contrast swaps its directionality in 2003. The male supremacy in this area is truly accentuated in the late eighties but since the early nineties it drops considerably as confirmed by the three *Shrink* patterns discovered:

```
- Shrink from period 6 (92) to the period 7 (93) with
```

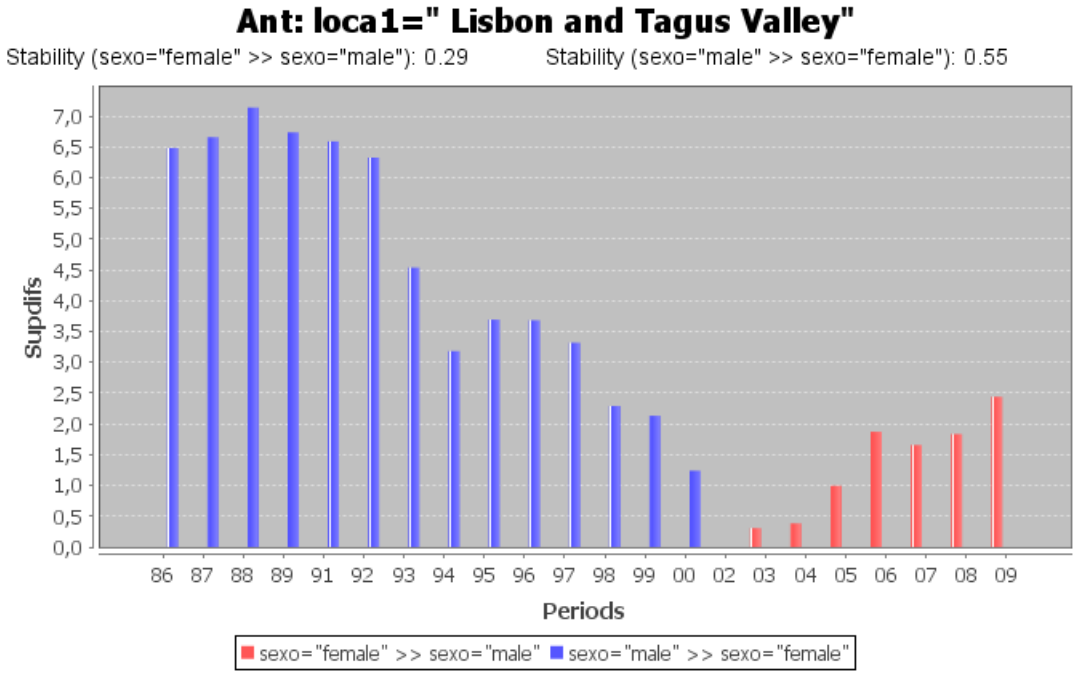


Figure 4.3: Contrasts found for employees of companies located in Lisbon and Tagus Valley area

```
a difference of -0.018;
- Shrink from period 7 (93) to the period 8 (94) with
a difference of -0.014;
- Shrink from period 11 (97) to the period 12 (98) with
a difference of -0.010;
```

By the last periods, females overthrow the former male dominance for this attribute and the tendency seems to indicate the increment of this difference.

Interesting enough is the contrasts obtained for the North Coast area like shown in figure 4.4. The results appear to be very similar with the exception that the *flipped* contrasts are in the opposite order (i.e. female superiority into male one).

Many reasons could have contributed for the contrasts found. Some relations with the predominant industries at each area can be one of these reasons. For example, the contrast $sexo = "male" >> sexo = "female"$ was found

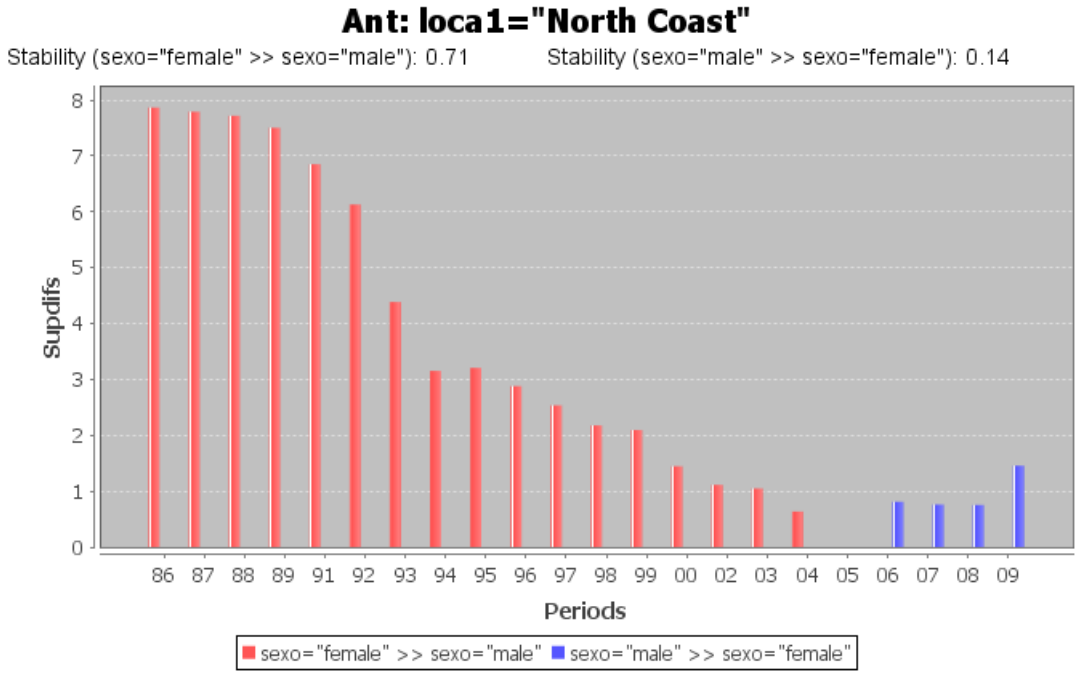


Figure 4.4: Contrasts found for employees of companies located in the North Coast area

with antecedent $inda1 = "construction"$ & $loca1 = "NorthCoast"$ and has increased significantly along the years. No contrasts were found for the construction industry in Lisbon area. A similar example involving the wholesale industry, with women having the edge, was found for the Lisbon area but not for the North Coast.

There were more industries that defended this position. A relation could be established with the education observations done in the previous section as the industries which typically require more qualified workers are associated more often with the Lisbon area than the North Coast. Therefore, justifying the results obtained.

4.3.3 Salary

Some studies made earlier with this data suggested that the average salary received by female employees is considerably inferior to a male employee.

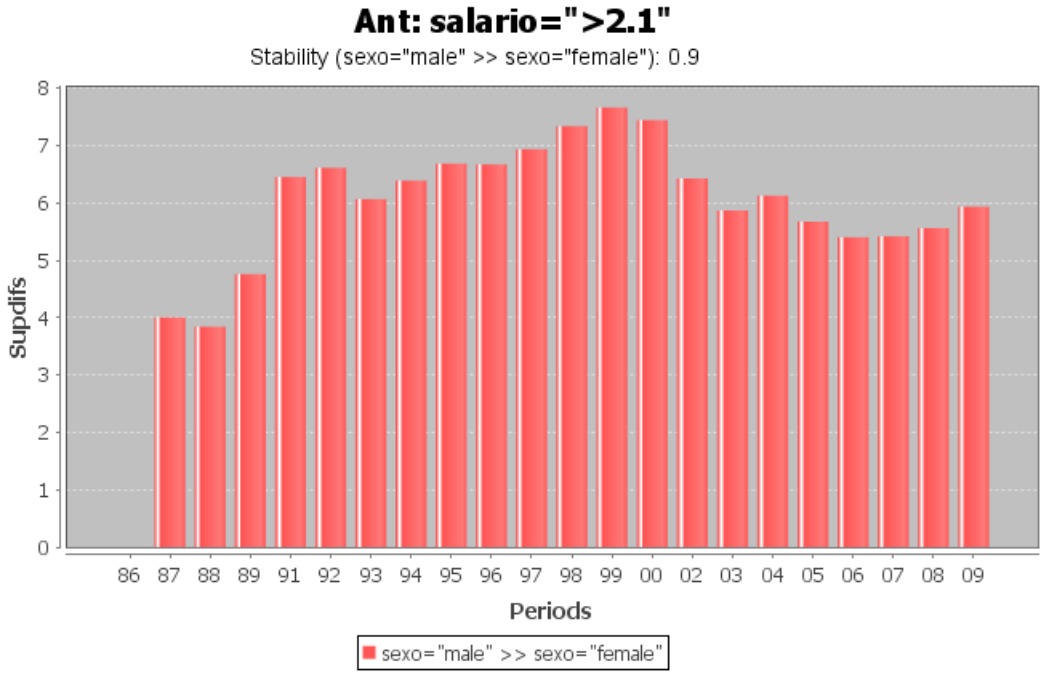


Figure 4.5: Contrasts found for employees whose income is over 2.1 (in natural logarithm)

This notion is well represented by the contrasts found for antecedents that contained items from the *salario* attribute.

This attribute was discretized into eight disjoint intervals. For the four intervals which contain the higher values of salary, the only contrast found was $sexo = "male" >> sexo = "female"$ in almost all periods as stated by the value of *stability* calculated for each one of them (the smaller being 0.9). This is strong evidence on the fact that women are clearly discriminated in their income.

Figure 4.5 discloses the contrasts found for the attribute $salario = " > 2.1"$ which is pretty representative of the top four intervals listed before.

Only the contrast for the first period was not found, probably due to this interval cut point being a value of income which was incompatible with the salaries at that time (the inflation effect was not applied to this attribute). Thus, not meeting the *minsup* constraint. Since a contrast appears in the

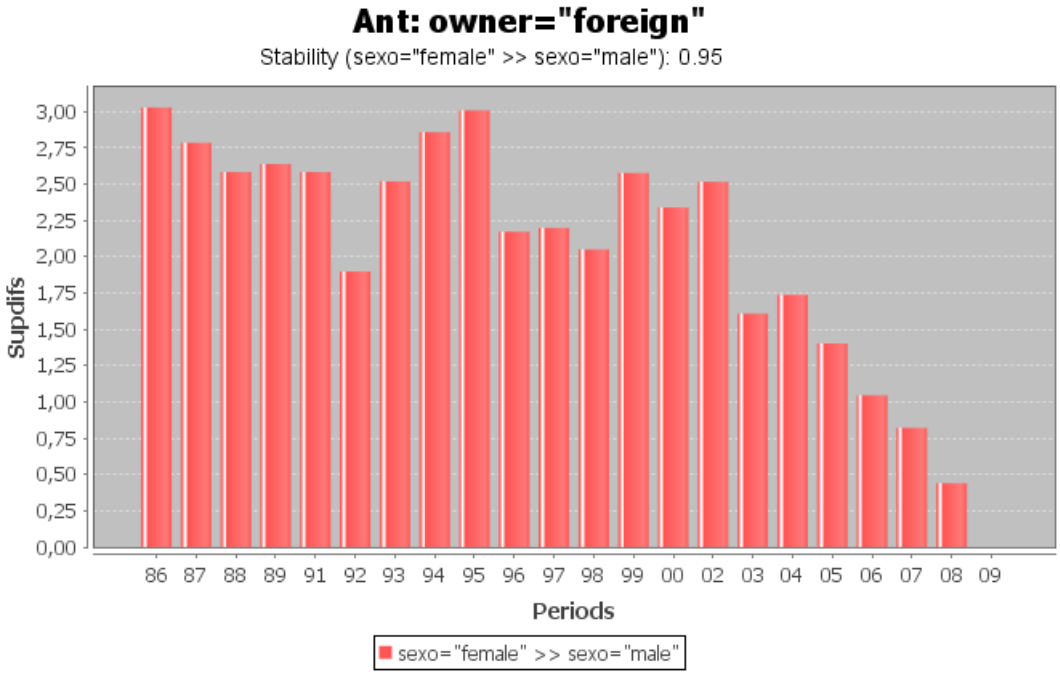


Figure 4.6: Contrasts found for employees of companies with foreign capital

following period, a *Spring Up* pattern is discovered.

```
- Spring Up from period 1 (86) to period 2 (87);
```

4.3.4 Ownership

The dataset analysed consists of all companies of the private sector. This does not consider that some of these companies could have foreign and public capital as well. The *owner* attribute intends to represent this situation. Each value of this attribute comprises a decidedly distinct situation, which led us to view it as an element of sheer discriminative power.

Companies with foreign capital (see figure 4.6) show that the number of female employees is significantly superior in relation to males for most periods. This relation has some stability until 2002 when the *supdif* of the contrast $sexo = "female" >> sexo = "male"$ started dropping and is absent for the last period as proved by the *Fade Out* pattern extracted:

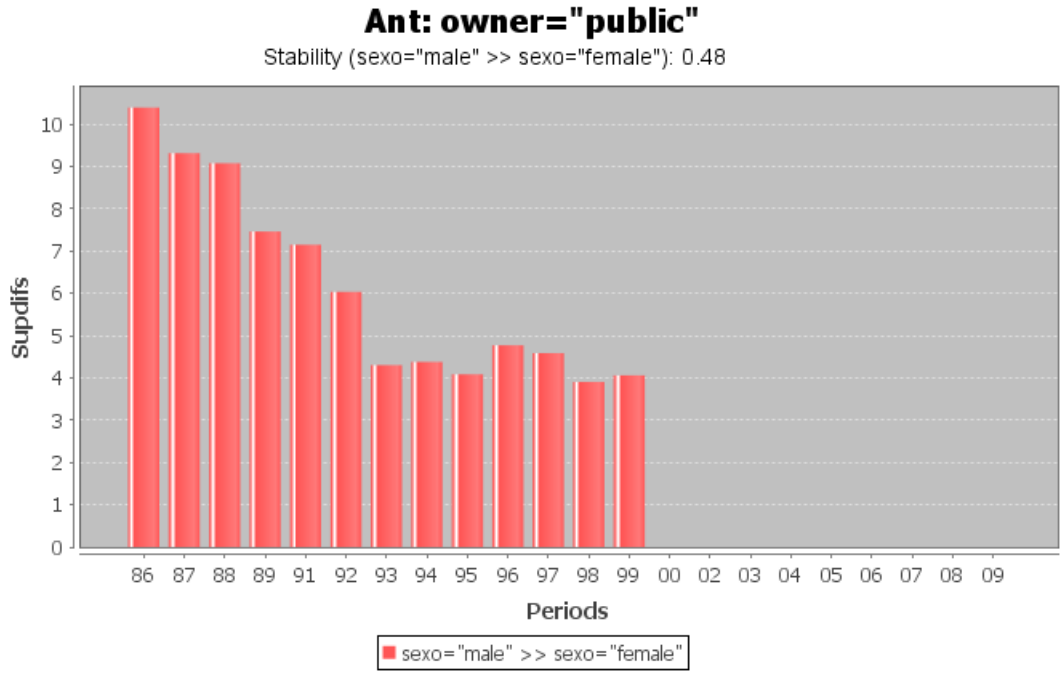


Figure 4.7: Contrasts found for employees of companies with public capital

```
- Fade Out from period 21 (08) to period 22 (09);
```

What is curious about it is how distinct the relation is in regard to enterprises with public capital shown in figure 4.7. For these firms, the opposite contrast is present: $sexo = "male" >> sexo = "female"$. The difference is particularly substantial in early years as shown by the considerable *supdifs* of the contrast. The difference keeps decreasing until 1993 and then kept constant until 1999 where it disappeared abruptly and never showed up again.

Since there was no data loss, the lack of contrasts for the last decade can be read as a relation of equality in the employment of both men and women in firms with public capitals. All values of the *owner* attribute seem to appoint for this conclusion in the latter years as the contrasts are not present or have tenuous *sigdif*.

4.4 Conclusions

The results presented reveal how some factors are highly discriminative in regard to gender in the Portuguese private business sector. The objectives proposed in section 4.2 were fulfilled as some situations with a relevant change along the years considered were found and some attributes were identified as having a key role in discriminating along gender.

The first two attributes seen (education & location) present two situations which assume distinct realities in the present in comparison to previous periods. Females are now investing more in their education leaving men behind even more. This relation was not true in the first years seen, at least for higher education.

For the four situations studied, each attribute revealed some form of discrimination in regard to gender. These elements were not the only drawing contrasts, which proves the dissimilarity between sexes in this setting.

In terms of accuracy, results went accordingly to some common knowledge like women pursing higher education more often than men and receiving a lower salary on average. The next case study, shown in chapter 5, will allow to check the accuracy of the proposal as well. This also serves as evidence that contrasting can be a form of learning if the dataset used is of unknown context.

Chapter 5

Case Study #2 - Basketball data

This second case study revolves around the sport of basketball. It is a very popular sport and with a tradition of a long recording of match statistics. For this reason, it ends up being a interesting case study for a Data Mining task.

An introductory part involves the history of the sport and how it has evolved. Some of the terminology used and a positional overview will be described.

Next, comes the goals of applying our proposal to a basketball context, what would be interesting to find and how this distances itself from the other case study and other Data Mining proposals done in this context.

The data used will be detailed as well, from the form it was collected, all data preprocessing performed in order to improve data quality and to meet all the requirements.

Results will come last and they are going to be thoroughly discussed and analysed towards what is considered relevant and conclusions will be drawn in regard to the goals set before.

5.1 History of the sport

Back in 1891, Dr. James Naismith, a Canadian physical education teacher was asked to come up with an indoor game to occupy a group of undisciplined students and to keep them away from the harsh cold of winter (Naismith & Baker, 1996). The game, which evolved and is now known as basketball, involved two peach baskets, one for each team and was played with a soccer ball where the goal was to get the ball inside the opponent team basket which was hanged in a balcony.

Naismith put together the original thirteen rules (*Naismith's Original 13 Rules*, n.d.) which differ quite a bit from current rules as running with the ball was not allowed (dribbling was not mentioned) and there was not any kind of shot differentiation as they were awarded one point regardless of where the ball was shot.

Due to the nature of the basket, every time a team scored a point the game had to be halted so someone could climb a ladder to retrieve the ball from up there. Later, the bottoms of these baskets were removed in order for the ball to fall down but in 1906 the metal rim with net and board much like the ones present today were introduced. The soccer ball was replaced to a dedicated basket ball around that time as well.

The sport received some recognition worldwide and made it in the Olympics for the first time in 1936 in Berlin where 21 national teams participated. In United States of America, the first professional league appeared in 1898 but was disbanded a few years later. For the next fifty years, many leagues were formed but none became prominent until there was an effort in 1949 to establish a real nationwide league which remains active, the National Basketball Association (NBA).

The introduction of the shot clock in 1954 revolutionized the game (*History of Basketball - SportsKnowHow.com*, n.d.). Before, teams had no time limit in

order to shoot the ball. This led sometimes to very long possessions and teams would take advantage of this when winning by stalling as much as possible. The 24-seconds shot clock improved the game pace and speed making the sport much more appealing.

Another league, the American Basketball Association (ABA) was founded in 1967. It would eventually end up merging with the NBA in 1976 but introduced a few concepts like the 3-point field goal that was adopted by the NBA in 1979, the Slam Dunk contest and faster pace of play present. These ABA innovations carried over to the NBA right after the merger making the game much more exciting and interesting for the spectator. That translated in a bigger audience and brought more attention to the sport (Pluto, 2007).

The sport rules did not received significant modifications from that point until now, but the game evolved especially in athleticism and from a tactical standpoint. It continued to grew attention from people worldwide due to its exciting fast pace and unpredictable game outcomes being one of the most followed sports in the world today.

5.2 Relevant Basketball terminology

Before proceeding into the data collected, the main basketball terms required for our task need to be introduced. Firstly, it is important to understand the scoring system because a shot may be awarded one, two or three points.

Certain infractions may award the player who got fouled a chance to go to the free throw line and shoot from there without opposition. This shot is called the *free throw* and each successfully converted free throw is worth one point. Figure 5.1 shows where this line is located. To gauge the efficiency of a player in this situation, the Field Throw Percentage (FT%) is the statistic used. For example, a player with 0.5 FT% converts half the free throws he

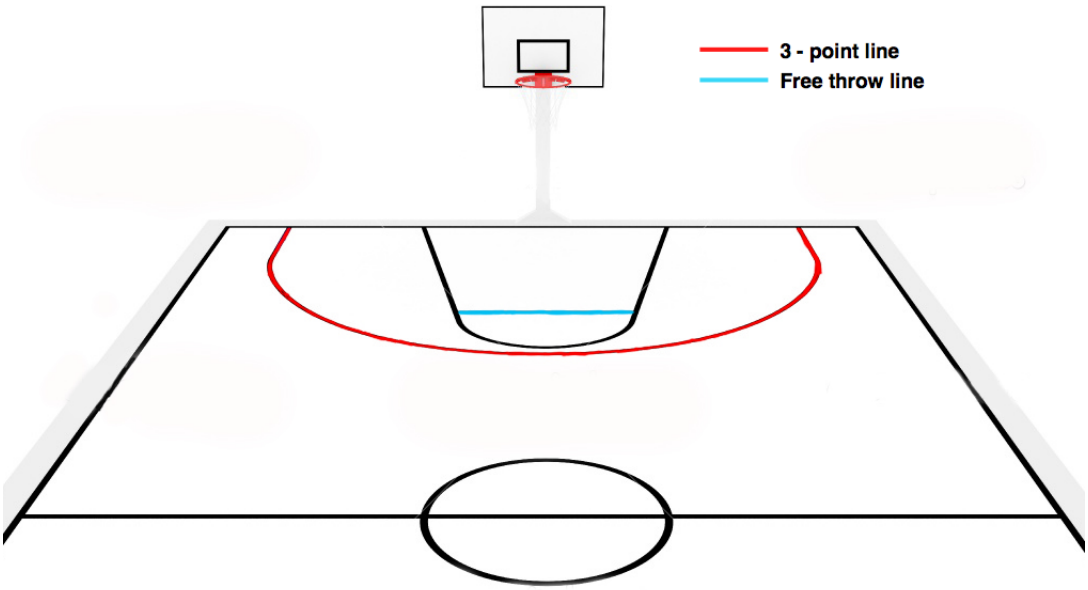


Figure 5.1: Basketball court representation with free-throw line and 3-point line

gets.

Except from this special situation, any shot is awarded two or three points, most commonly known as *field goal*, depending from the distance of the attempt. Figure 5.1 reveals the three-point line which separates a two-point shot from a three-point shot. A converted shot inside that area or when stepping on that line is worth two points and outside is worth three. The Field Goal Percentage (FG%) measures a player shooting efficiency and Three Point Percentage (3P%) measures the efficiency from the three-point range.

Other than the shots and their own associated statistics, there are also other actions in the game that are also important and have impact in the course of a game. A **rebound** is the act of recovering the ball after a missed *free throw* or *field goal*. They can be divided in defensive and offensive *rebounds*. The defensive ones are gained by the team defending at that moment and mark a change in ball possession and in the offensives ones, the attacking team recovers the ball after a missed shot keeping the ball possession.

A **block** (abbreviation from blocked shot) is a legal deflection (i.e. not

fouling) of an opponent field goal attempt thereby preventing it. The hands of the opponent cannot be touched and the ball cannot be intercepted in its downward course (goaltending violation).

An **assist** is a pass that leads to a converted shot and a **steal** is the act of regaining the ball possession from an opponent dribble or pass, striping him of the ball causing a **turnover**.

The *turnover* is the act of losing ball possession without shooting at the basket, a *steal* causes it but other violations like travelling and throwing the ball out of bounds are other examples of *turnovers*.

5.3 Positional overview

Another important part of the analysis will be in regard to player positions so this requires a previous clarification as well. There are not specials positions with different permissions in basketball like the goalkeeper in football or the libero in volleyball. Every player on the field is under the same set of rules and can perform exactly the same set of actions.

Each team has five players on the court simultaneously with each one at each position. There are three main positions in basketball: the *Guard*, the *Forward* and the *Center*. A correlation with football can be made as there are the defenders, midfielders and forwards but specifically there are more positions like the left-back, right-back and centre-back just mentioning the defenders. In basketball, the same logic prevails with just a small reservation. Defenders are most focused on the defensive end and the forwards in the offensive end while in basketball all positions have responsibilities at both ends of the floor.

Figure 5.2 shows the different positions and their usual distribution on the court. The *guards* make the backcourt and the *forwards* and *center* the

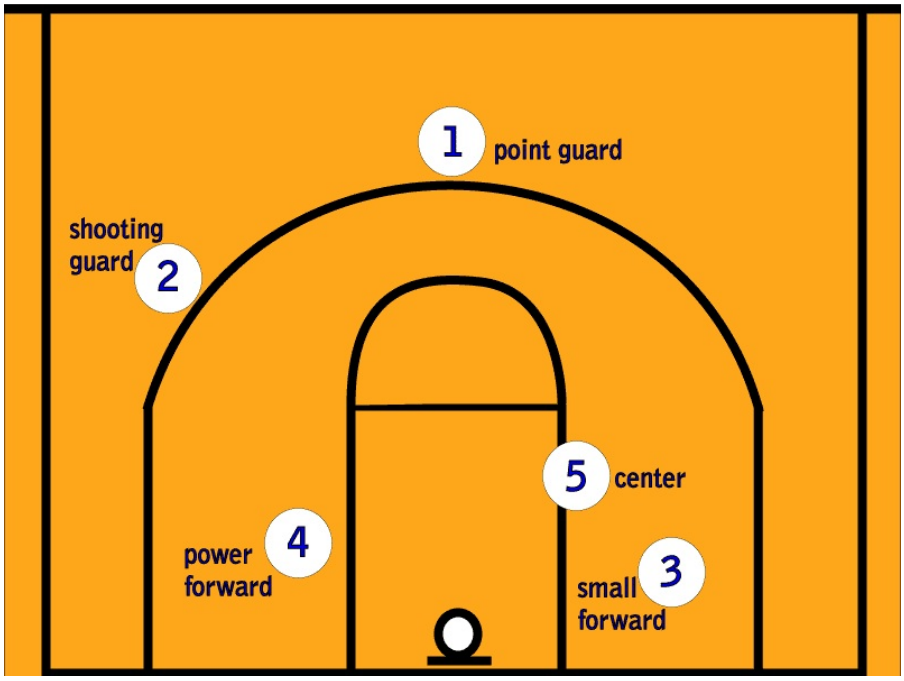


Figure 5.2: Positional distribution on the court

frontcourt. Let's specify each one task and common profile.

Starting with the backcourt, the *guard* position can be divided in two positions, the Point Guard (PG) and the Shooting Guard (SG). The PG also known as the one, is the team ball handler responsible for passing and making plays in order to facilitate the shot for his own team mate, requiring great vision to perform his job. It is usually the smallest player on the team. The SG or the two, is responsible for scoring and should have great shooting range, being able to consistently hit shots from the three-point range but closer to the basket as well.

Regarding the frontcourt and starting with the *forwards*, the Small Forward (SF) or the three, is usually the most versatile player in game because it executes plays typically associated with the SG as well as those done by a Power Forward (PF). To be able to do that, quickness and strength are requirements for them. The PF or the four has a style of play much more physical and usually plays with their backs to the basket offensively, reason why some

PFS also play as *centers* as both positions share some similarities.

The Center (C) position is associated to the taller and stronger players because they make use of that size in order to score close to the basket and to protect their own basket in the defensive end. They assume such a crucial role on defense that they are usually referred as "anchors".

These are the more traditional roles associated for each position, which does not invalidate that some players may perform a bit differently regarding their own abilities and also for strategical reasons. Sometimes a player besides the PG with good ball handling skills may lead and organize the offensive process which happens with relative frequency. Other examples could be given instead.

5.4 Objective of the analysis

Basketball has already been used as the context for some Data Mining tasks. Neural networks have been used for scouting purposes (Ivankovic, Rackovic, Markoski, Radosav, & Ivkovic, 2010) and for predicting game outcomes (Beckler, Wang, & Papamichael, 2008). Advanced Scout (AS) was a Data Mining application used by most of NBA teams back in 1996 that made use of the Attribute Focusing technique (Bhandari, 1995) to find the interesting patterns and reveal them in a user-friendly format (Bhandari, Colet, & Parker, 1997). Another work used different measures of similarity to find the most similar player to Michael Jordan, the consensual best player of all time (Jiang, 2011).

What makes basketball a good area for using Data Mining or statistical methods is the abundance of metrics and statistics that the game provides. Figure 5.3 shows a typical *Boxscore* from a basketball game. There are individual records for each player from one both teams that participated in the

USC TROJANS												
STARTERS	**MIN**	**FGM-A**	**3PM-A**	**FTM-A**	**OREB**	**REB**	**AST**	**STL**	**BLK**	**TO**	**PF**	**PTS**
DeMar DeRozan, F	27	5-11	0-1	3-4	4	6	2	0	0	3	1	13
Taj Gibson, F	29	5-11	0-0	7-10	6	15	0	1	5	1	3	17
Daniel Hackett, G	29	2-7	2-5	4-6	0	3	9	1	1	1	2	10
Dwight Lewis, G	29	7-13	1-3	1-2	2	6	3	1	0	2	0	16
Leonard Washington, F	23	4-10	0-1	1-1	6	12	1	0	1	1	1	9
BENCH	**MIN**	**FGM-A**	**3PM-A**	**FTM-A**	**OREB**	**REB**	**AST**	**STL**	**BLK**	**TO**	**PF**	**PTS**
Mamadou Diarra, F	8	1-1	0-0	0-0	0	1	0	0	1	0	2	2
Percy Miller, G	3	0-1	0-1	0-0	0	0	1	0	0	0	0	0
Marcus Simmons, F	15	1-2	0-0	2-2	1	2	0	0	1	2	1	4
Keith Wilkinson, F	5	1-2	0-0	0-0	1	2	0	0	0	0	1	2
Donte Smith, G	16	0-5	0-4	0-0	0	2	3	0	0	2	4	0
Kasey Cunningham, F	16	0-1	0-0	0-0	1	4	0	0	1	2	4	0
TOTALS		**FGM-A**	**3PM-A**	**FTM-A**	**OREB**	**REB**	**AST**	**STL**	**BLK**	**TO**	**PF**	**PTS**
		26-64	**3-15**	**18-25**	**21**	**53**	**19**	**3**	**10**	**14**	**19**	**73**
		40.6%	20.0%	72.0%								

Figure 5.3: *Boxscore* from a team performance in a basketball game

game. From such a table a lot of information can be extracted and each player performance can be evaluated. Most of the categories present there have already been explained in section 5.2.

Apart from the other works already presented, our goals here are quite different and the technique used will be different as well. It is not intended to predict future events like some of them but to assume an analytical role of learning and understanding from historical data. The main goal of this task is to understand the following:

- How each position influences the stat sheet, what is a typical contribution from a guard, forward or center player in terms of the categories involved and how they differ from each other;
- The evolution of the game and how the play style of each position has changed along the years;
- Verify if the results obtained match the expected common knowledge to ascertain the method accuracy;

These three queries summarize the goals of our analysis. From this, it can be concluded that the task will revolve around the players positions, being

this attribute the one containing the group values to be contrasted. Historical data is also a requirement for our method and it will serve the purpose of understanding the game evolution, if it has changed over years significantly and in which specific areas.

For someone with good insight and is knowledgeable about basketball, the results obtained probably will not constitute a breakthrough discovery and might not carry great value as they can be considered somehow trivial. That does not invalidate our task which intends to validate our method accuracy and provide the information of how each category observed is related to each position, serving at least as an alternative to deepen the game knowledge for those who do not follow it closely and wish to do so.

5.5 Data obtained

Considering the last section, some hints about the data have already been disclosed. Data from at least a significant period of time is required to compose the periods for our task. The aim is to check how some of these attributes have changed along the considered time window. An attribute with the players position is needed as well, since it will serve as the group attribute from which contrasts are going to be derived.

From section 5.2 terminology and figure 5.3 data, the type of attributes that are sought closely match our requirements.

There are plenty of websites that provide this kind of information by querying. The user might navigate through the website in order to see some information about a certain player career, his statistics in a certain season and even go down to a single game performance. The problem relies that these dedicated websites do not allow data to be used by a third-party or do not provide an option to extract the intended data in bulk.

Fortunately, *databasebasketball.com* (2011) provides a downloadable dataset with data from the early days of NBA (1946) until more recent years (2009). It contains a lot of data in CSV files regarding players, teams, drafts and coaches. However, the one that is going to be used is the *"player_regular_season.csv"* since it contains every player yearly contribution per line with the attributes usually present in a *Boxscore*. The playoff data was not considered because the sample is small (less games) and not every player make it to that point of the season, reason why it was decided to not merge them. Refer to table B.2 in appendix B to check the attributes contained in the dataset with a small description of each one.

This dataset contains a lot of interesting attributes but still lacks a position attribute which was deemed as imperative. That was easily fixed because there was a *"players.csv"* file containing that attribute and joining both datasets by the player ID solved the issue. That also allowed to include the height attribute (expressed in feet and inches) which will be a target of interesting conclusions (see section 5.6.1).

5.5.1 NULL values

The presence of NULL values has a significant impact in this dataset since there are a few attributes with no values especially in the earlier years. This happened because data from those times is hardly attainable compared to recent years in which data is easily spread and accessible.

Table 5.1 refers which attributes have NULL values and where they are located. They are very frequent at the first years until a certain point where the missing values do not appear any more. The attributes referred in table B.2 that are omitted in this table are absent of any NULL value.

Table 5.1: NULL values present in the dataset

Attribute	NULL Values
Minutes	Absent before 1951
Rebounds total	Absent before 1950
Defensive & Offensive Rebounds	Absent before 1973
Steals & Blocks	Absent before 1973 (just a few sparse values before)
Turnovers	Absent before 1977 (just a few sparse values before)
3-points	Present between 1967 & 1976 for ABA players and from 1979 onwards

After some investigation, it became evident why some attributes just have values from a certain year onwards. Let's take *blocks* and *steals* attributes for example. They just start appearing in the dataset in the year 1973. This happens because in this specific year, both the NBA (*Season review: 1973-74 | NBA.com*, 2012) and the ABA started to record these statistics which did not happened the years before. The reason still applies for offensive and defensive rebounds at the same year or turnovers in 1977.

This requires proceeding with caution and will have some implications in the results found. The periods constitution will be addressed in section 5.5.3 but the periods formed that contain data from these early stages will be seriously affected from this data absence and not much can be inferred from them.

5.5.2 Data transformation and discretization

By analysing the collected data, three situations that required some preprocessing were immediately spotted.

First one, was simple and involved the height obtained. That happened because it was split in two attributes, *h_feet* and *h_inches*. There was no reason for them to be split as there is no interest in analysing them separately. They are just parts of a single measure - the height. Both were put together in a single attribute separated by an hyphen.

Other situation, involved the totals obtained in a season. This has some major disadvantages. Players with good contributions and few minutes could be on par with players with poorer contributions but had benefited from more playing time. Also, players that were sidelined many minutes due to injury or suspension would be severely penalized.

In reality, the most common way to see this statistics done in a season is usually in a per game basis which gives the average contribution of a player for a single match. Another possibility also used but less frequent is the per-36 or per-48 minutes adjustment in which the contributions are averaged for each 36 or 48 minutes played.

This motivated to transform all the attributes that contained totals into a per game average. The presence of a games played attribute in the data allows this transformation to be done.

In this situation, a user-defined discretization will be used, meaning the interval cut points are defined by using background knowledge. The greatest challenge of this approach relies on the fact that each attribute is a single case and requires different input on what would make an adequate set of intervals. The first step involved studying each single attribute and understand its amplitude, average value and standard deviation. This allowed to perceive each specific attribute behaviour and to assist the discretization process.

Two types of intervals were always aimed and those were the first and the last ones. The first ones always tried to represent the concept of insignificance. If a player falls on that interval in that particular category, it means he is not proficient in that department ranking in the lower ends. With the last interval it is intended to express the exact opposite; a player who is exceptional and a great contributor in that specific category. The intervals located in the middle are more attribute-dependent and are more prone to individual options and decisions.

From the initial attributes, a small remark on the fact that the attributes related to players names (first and last one) are removed as they hold no value for our analysis.

5.5.3 Temporal division

The last step before having the data ready to apply our method is defining the periods of analysis. Our data is grouped by year, so that would be the minimal period size possible. Using yearly periods, that would total 64 periods (2009 - 1946 + 1) which is a very considerable amount. Not only that, but having this many periods would make each one with only a few hundred records possibly comprising a problem with such a small sample.

An extended period was then deemed a better choice with the final selection falling in the decade. The case study about labour data (chapter 4) already made use of a significant number of periods and already proved the technique worth in that situation. By reducing the periods to seven (from the $40s$ to the $2000s$), our task is simplified and the decade was always the time measure used when remembering and discussing past accomplishments and players in sports history.

From the last dataset iteration in which numerical attributes where transformed in categorical ones, this block of data is split in multiple files, one for

each period defined accordingly to requirements stated in section 3.1.

5.6 Results analysis

With the adequate data, the execution of CAREN for discovering CS is done for each period with a *support* of 0.05 followed by the PPCS with a *sigdif* of 0.02:

```
java -jar PostProcessingCS.jar  40s.csv,50s.csv,60s.csv,
70s.csv,80s.csv,90s.csv,2000s.csv 0.02 -f -out
```

Regarding the objectives stated in section 5.4, in the following subsections different attributes or set of attributes present in the data will be scrutinized while checking the evolution along the years and how each position is reflected and differs from each other on that own group of attributes.

5.6.1 Height

Height is the only attribute present in the results that derived any *Flip* patterns. That immediately reveals that, along the years the average height of the players playing in a certain position has changed.

First, contrasts were found for every height ranging from 5'10" (1,78m) to 7'2" (2,18m) although smaller players (like Muggsy Bogues - 5'3" tall) and bigger players (like Manute Bol - 7'7" tall) have played the game, it is just they form a small sample and the minsup constraint is not met.

Starting from the *Guard* position, contrasts of the type $pos = G >> X$, where X is any other position are found from height=5'10" to height=6'7" but not for the higher values of height which is indicative that *Guard* players are the ones with less stature compared to *Forwards* or *Centers*.

Players playing the C position, like stated in section 5.3, play near the basket and take advantage of their size to score. This implies that they are

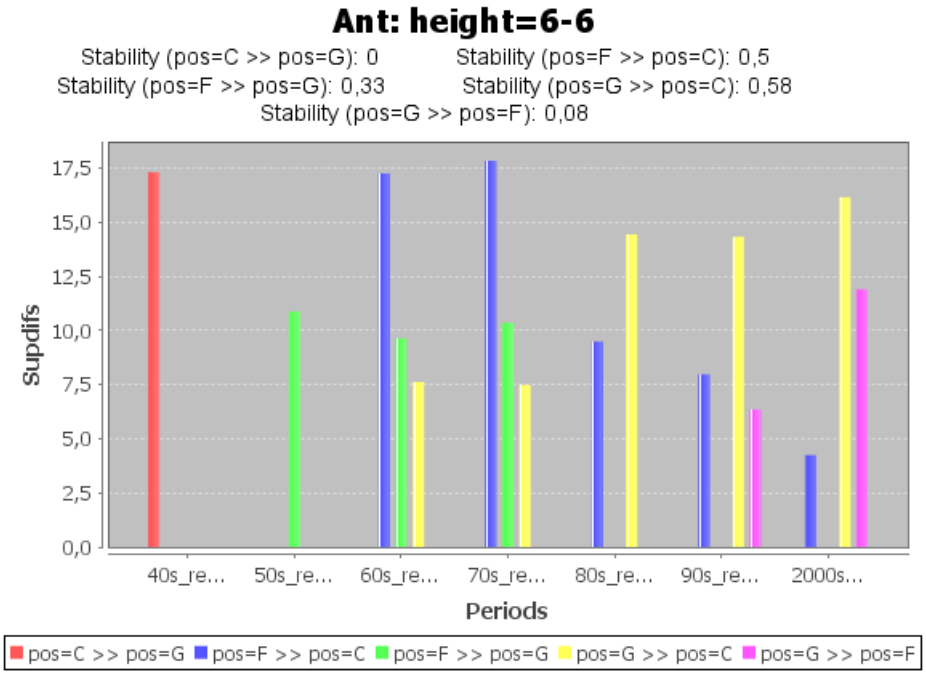


Figure 5.4: Contrasts found for height = 6'6"

the bigger players in the team. That is proved by the contrasts found that range from height=6'6" to height=7'2" staying away from the smaller statures.

The *Forwards* fall somewhere in between, bigger than *Guards* but smaller than *Centers*, explained by the absence of the contrast $pos = G >> pos = F$ and the contrast $pos = F >> pos = C$ in any period for any height.

Now, let's go back to what has been stated in the section beginning about the *Flips* found to understand how has the game changed in terms of stature since the relation between positions in terms of size has been exposed. In figure 5.4, there is a capture from the graphic shown in the *Viewer* for the antecedent of height=6'6". This selection was deemed as interesting for the fact that it has six different contrasts that are representative of what has happened in terms of height in the course of time.

First decade, the contrast is $pos = C >> pos = G$ which would be unthinkable in current times, as a player of that stature would automatically be labelled as undersized. Getting to the 60*s*, the first *Flip* shows up in the

form of $pos = G >> pos = C$ (yellow color) turning around from the contrast found two decades ago. This contrast keeps appearing until the last period and the difference in contrast support widens (yellow bars increasing in size) for each passing period which favours the height of 6'6" for *Guards* even more than for *Centers*.

There is still one more *Flip* in this CS, as $pos = F >> pos = G$ (green color) present in 50s-70s becomes $pos = G >> pos = F$ (purple color) in the 90s and 2000s. This also marks the evolution of height to present years (the last period taken in consideration). In this case the height 6'6" is ruled as more frequent for *Guards* rather than for *Forwards* in more recent times.

The figure 5.4 is well representative of the global behaviour and the reason why it was decided to analyse this particular CS. Nowadays, typical *Guards* stand at 6'6" mark and lower, *Forwards* between 6'7" and 6'10" and *Centers* from 6'11" onwards. This helps explain a current trend of NBA where teams are overpaying the players playing the C position since it is very difficult to find a player with the adequate, standard size being skilled as well. It would be interesting to review this size issue in the future, and check if the players' height playing a certain position keep changing or instead the stabilization occurs.

5.6.2 Shooting percentages

Let's now check if there is any relation between the efficiency of the shots taken for each position considered. *Centers*, due to the fact that they are the player which plays closest to the basket should give them the edge in shooting percentage as their shots are mostly close-ranged opposed to *Guards* which are required to take more often the medium and long-ranged shots.

The FG% for the lowest intervals $(]0; 0.40]$ & $]0.40; 0.42])$ corroborates our first hypothesis as the only contrasts found are $pos = G >> pos = F$ and

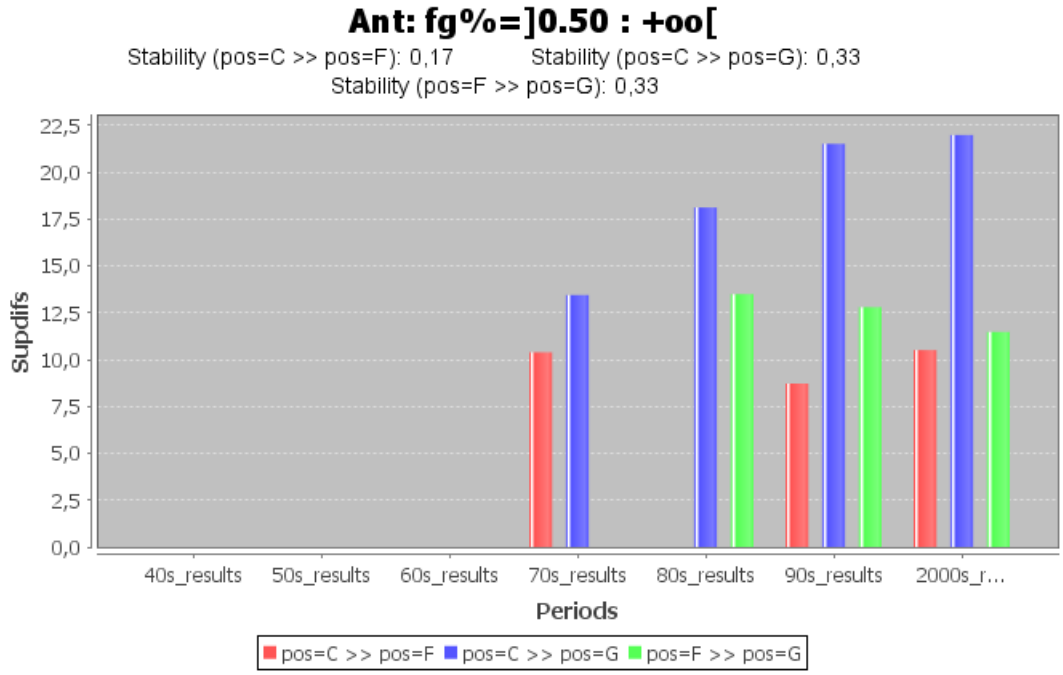


Figure 5.5: Contrasts found for fg% $>$ 0.50

$pos = G >> pos = C$. This shows that *Guards* have a worse field goal percentage than the other positions. It is not the case that they are worse shooters, the reason resides on the fact that difficulty of the shots taken is higher contributing for these lower scores. The volume of shots taken might also contribute for that, as the contrast $pos = G >> pos = C$, is present for the highest interval defined for $fgapg$ (field goal attempts per game).

Looking at the other end, figure 5.5 shows the contrasts found for the antecedent of $fg\% =]0.50; +\infty[$. In the first periods, there are no contrasts which could be indicative that there were no significant distinction between the positions on regard of the shots taken. However, since $70s\_results$ the contrast $pos = C >> pos = G$ (blue bars) is present and the gap is increasing for each following period revealing that the hypothesis formulated first is correct and is getting even more accentuated in the recent periods.

Looking at last season's top FG% players (*databaseBasketball.com - NBA Basketball Statistics, Draft, Awards, and History*, 2011), the top six ranked

players are *Centers* while only one *Guard* finished above the 0.5 mark (of a universe of 19 qualified players), validating the results obtained.

Also interesting, is the absence of any contrast for any period in this antecedent: $fg\% =]0.42; 0.45]$. This is the interval where the average FG% for all players relies and no individual group stands out in regard to another.

For 3P%, the relation is exactly the opposite. The *Guards* assume the position of obtaining the best percentages in the higher tiers of intervals while *Centers* rank in the lower ones. It is not very common for Cs to attempt this type of long shot which justifies why they clearly stay behind in this category. The *supdif* of the contrast $pos = C >> pos = G$ for $3p\% = [0; 0.10]$ reaches 60%, a extraordinarily considerable difference. It should be noted that the data about three points is just available from the period of $60s\_results$ onwards (see section 5.5.1), reason why contrasts can not appear before that same period.

Both FG% and 3P% favour one position more than other like stated before. For FT%, all players are in a position of equality as they all shoot from the same place without opposition. Regarding the number of free-throw attempts in a game, *ftapg*, the few contrasts obtained are pretty inconclusive as they have a pretty low *stability*, most of them being zero. This can lead us to conclude that the position is not an influential factor in determining the times a player goes to the line.

Figure 5.6 shows the contrasts found for players that average more than 83% successful shots at the free-throw line. Despite what has been said in the paragraph above, it is clear that *Guards* perform considerably better in these situations than *Centers* as the contrast $pos = G >> pos = C$ is present, and the *supdif* tends to increase at each successive decade with conclusive values that clearly differentiates both positions performance.

The *Forward* position assumes once more the place in between the other

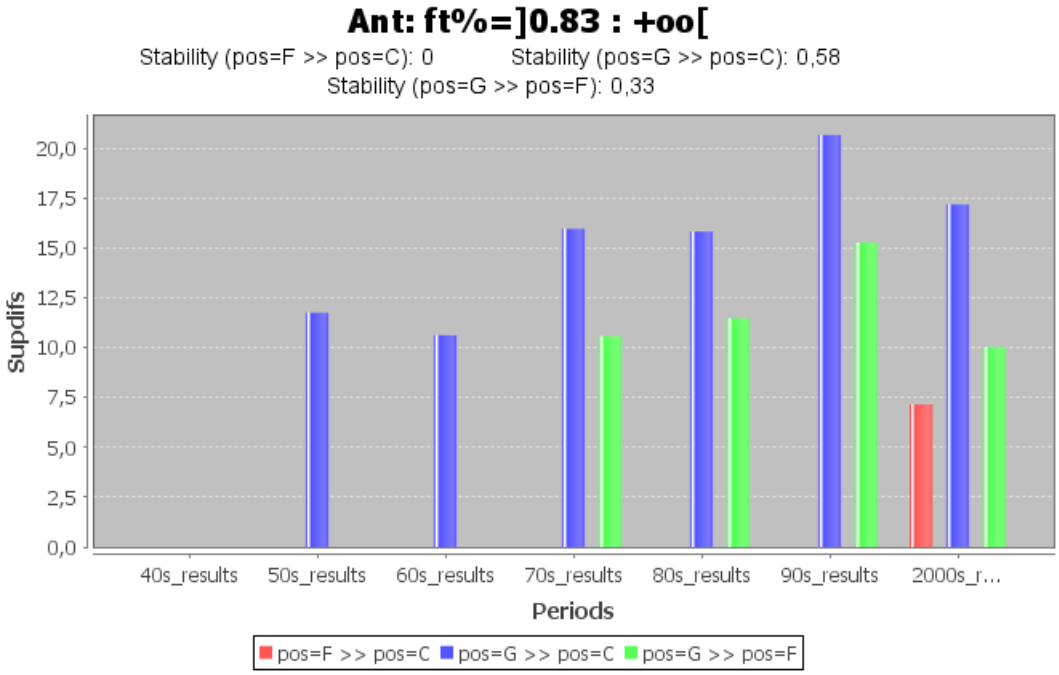


Figure 5.6: Contrasts found for ft% $>$ 0.83

two. They perform better than one of them but worse than the other for all shooting percentages seen (figure 5.5 & figure 5.6). This tendency is present in most of the factors studied not only these related with shooting.

5.6.3 Steals & Assists

Steals and *assists* are seen together as they are usually associated with *Guards* contribution. Being the players that handle the ball more time, it is natural that they tend do provide more *assists* than a player who does not have the ball much time in possession. As for *steals*, the results seen in section 5.6.1, these players are smaller in stature which usually translates into quickness and agility comparing to *Forwards* and *Centers*, attributes that make them more able to strip the ball from other players and intercept passes. Checking the top contributors in these categories from last season, *Guards* composes almost the entirety of the players (*databaseBasketball.com - NBA Basketball*

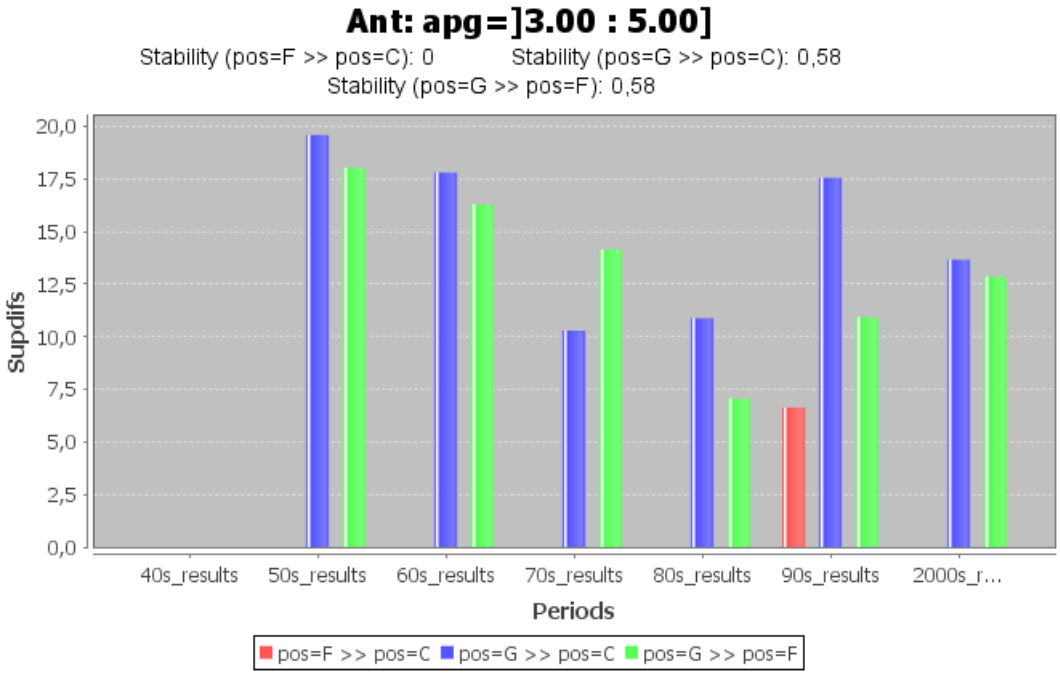


Figure 5.7: Contrasts found for players that average between three and five assists per game

Statistics, Draft, Awards, and History, 2011) especially for *assists* as for *steals* there are few more exceptions.

Starting with assists, it should be noted that they are mostly associated with PGs rather than all *Guards*. It is these players that are responsible for orchestrating the offensive process and be able to set open shots for the teammates. This make them rack assists more often than other players.

Figure 5.7 shows the contrasts for the antecedent of $apg = [3; 5[$, this interval is already representative of the general situation and not only that of $apg >= 5$. As expected, *Guards* show superiority in regard to the other positions as the *supdif* of each contrast is quite considerable.

Two interesting observations can be drawn from this figure. First, the contrasts in the decade of 50 are the ones with the biggest *supdif*, this can be misleading because this does not mean that in this stage the *Guards* did more *assists*, in fact, what happens is quite the opposite. For the antecedent

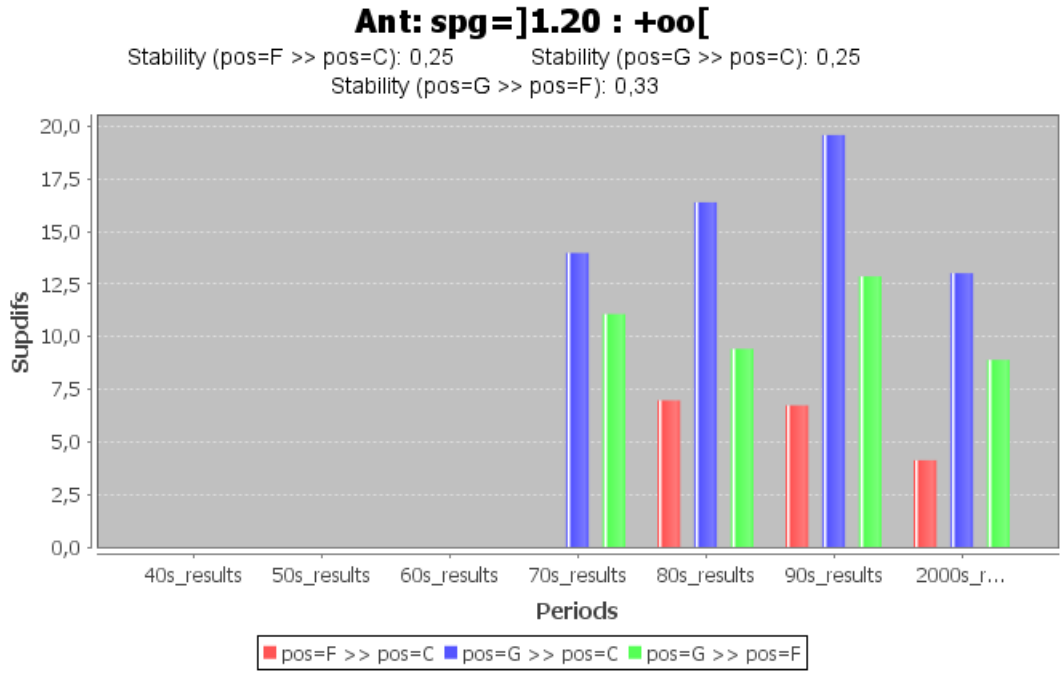


Figure 5.8: Contrasts found for players that average more than 1.2 steals per game

of $apg = [3; 5[$, for this same decade, the contrasts of $pos = G >> pos = X$ have the lowest *supdif* of all periods, which is indicative that *Guards* players in this period were not as proficient as they were later on.

The other surprising fact, is that for the first time, *Forwards* do not occupy a place in between the other two positions. This time, they stand next to *Centers* and depending on the period, the contrast of $pos = G >> pos = F$ in confrontation with $pos = G >> pos = C$ is sometimes more accentuated but for other times it is not.

Regarding *steals* as seen in figure 5.8, there is a more familiar trend with a clear distinction between the three positions, with *Forwards* staying once again in middle ground. *Guards*, like theorized, assume the throne in this category as the the discovered contrasts $pos = G >> pos = F$ and $pos = G >> pos = C$ rightfully state.

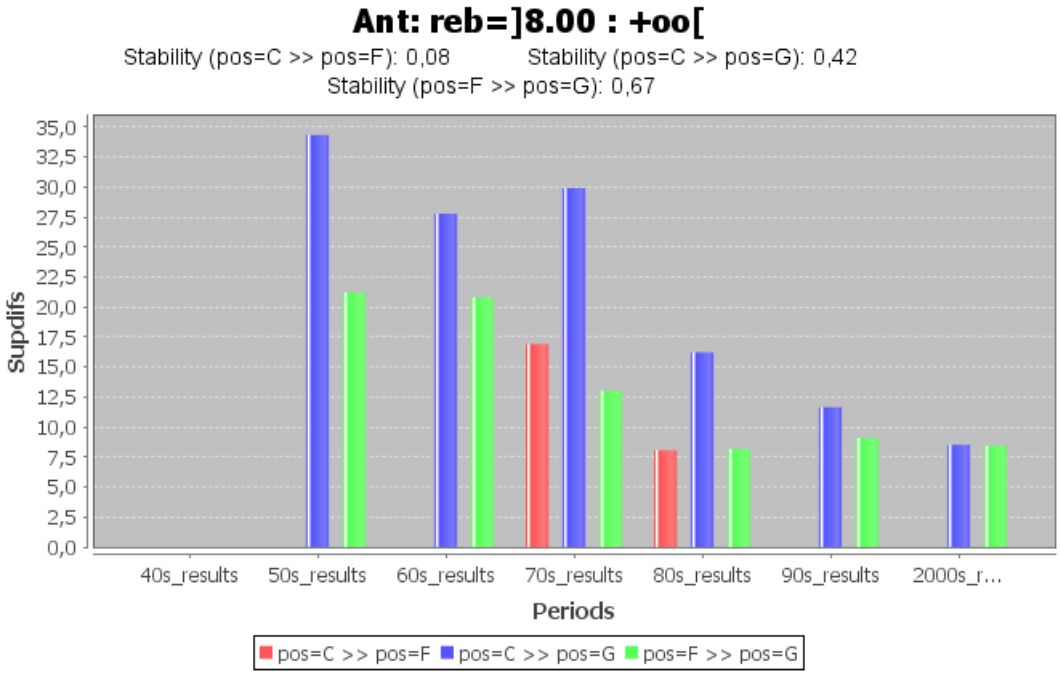


Figure 5.9: Contrasts found for players that average more than 8 rebounds per game

5.6.4 Rebounds & Blocks

Like *steals* and *assists*, these two statistics are bound together as they are most commonly obtained by frontcourt players: *Forwards* and *Centers*. Size assumes a vital role in obtaining both statistics as both usually involve touching or recovering the ball in mid-air. From the top-20 contributors in *blocks* and *rebounds* from last season (*databaseBasketball.com - NBA Basketball Statistics, Draft, Awards, and History*, 2011), there is not a single *Guard* in the mix.

Rebounding usually occurs near the basket. This also favours both *Forwards* and *Centers* as they play near the basket. Figure 5.9 shows the contrasts for players who obtain over an average of eight rebounds per game. Interesting enough is the decreasing of *supdif* for the contrast of $pos = C >> pos = G$ and $pos = F >> pos = G$.

Two forces can be at work for that phenomenon to happen: More *Guards*

are averaging more than eight *rebounds* per game or if less *Forwards* and *Centers* are obtaining this mark. Since no *Guard* has averaged more than eight *rebounds* per game, it is the frontcourt players who are getting fewer *rebounds*.

Few factors can justify what has happened, even if *Guards* are not getting that many *rebounds*, they are in fact getting more than in early years. For example, since the 80*s* decade, the contrast $pos = G >> pos = C$ or $pos = G >> pos = F$ is not present for antecedent $reb = [2; 3.5[$ but it was present in the earlier decades with a significant *supdif* (between 15 and 20). This can be explained by the increment of the average height of *Guard* players like seen in section 5.6.1 but also by the new breed of players with greater athletic bodies as the training regiments improved which enabled them to fight more efficiently for loose balls.

This effort has become less specialized, unlike other statistics, and started being more of a responsibility for all team members. From figure 5.9, it is evident how contrasts regarding *Centers* and *Forwards* are getting equal by the last decade as their *supdif* is virtually the same.

There is no significant difference between defensive and offensive rebounds as they follow the general behaviour shown for the average number of total rebounds.

As for *blocks*, figure 5.10 has the contrasts found for $bpg > 1.10$. As seen in section 5.5.1, data for *blocks* is only available from year 1973 reason why there are not any contrasts in the first three decades.

Centers appear as the main contributors in this category but what really stands out in the evolutionary scale is the *Forwards* as the contrast $pos = F >> pos = G$ keeps steadily increasing for every single period and the contrast $pos = F >> pos = C$ keeps decreasing since the 80*s* as they reveal successive *Shrink* patterns.

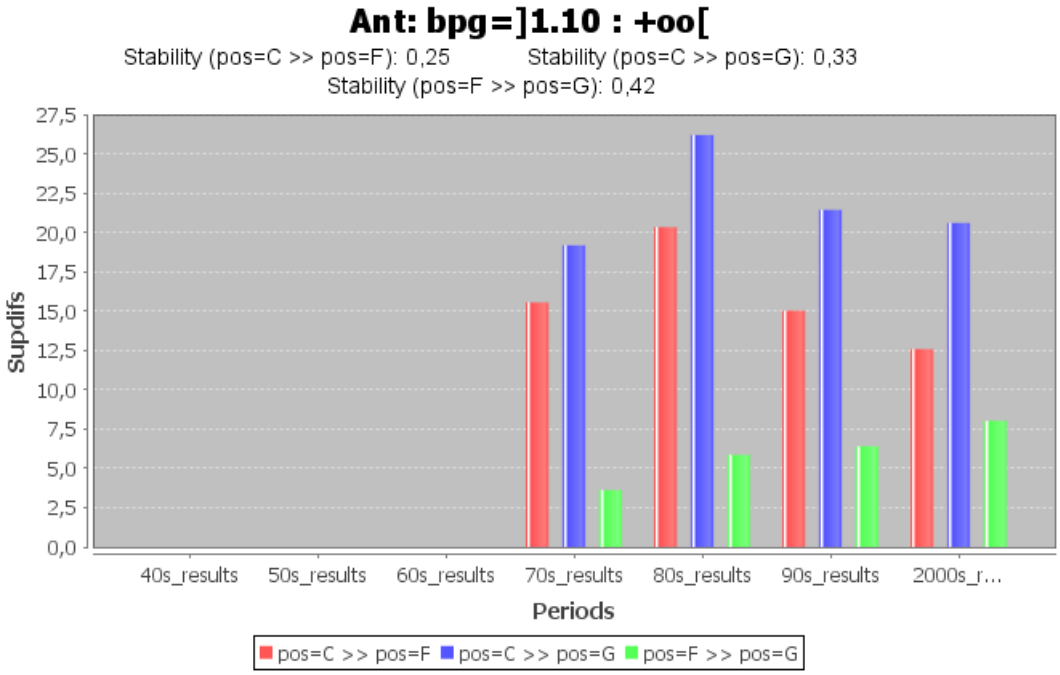


Figure 5.10: Contrasts found for players that average more than 1.1 blocks per game

5.6.5 Other discoveries and stats

In this subsection, there is the exposure of other findings and attributes that did not have a sufficient number of contrasts discovered along the timeline.

Turnovers is an attribute which did not draw many contrasts. Like what has been stated for *assists*, it would be expected that *Guards* which are the player who keep the ball in hands more time, would also be more prone to lose it. The contrasts founds validate this as they are $pos = G >> pos = C$ found in the last two periods and $pos = G >> pos = F$ in the last period. It is just the small sample that does not allow to stand strongly by the relation theorized.

Regarding *personal fouls*, figure 5.11 shows the the contrasts found for players who average more than three fouls per game. Frontcourt players seem to commit more fouls than *Guards* although the contrasts are getting smaller by the final periods. *Centers*, as they act like rim protectors they have to

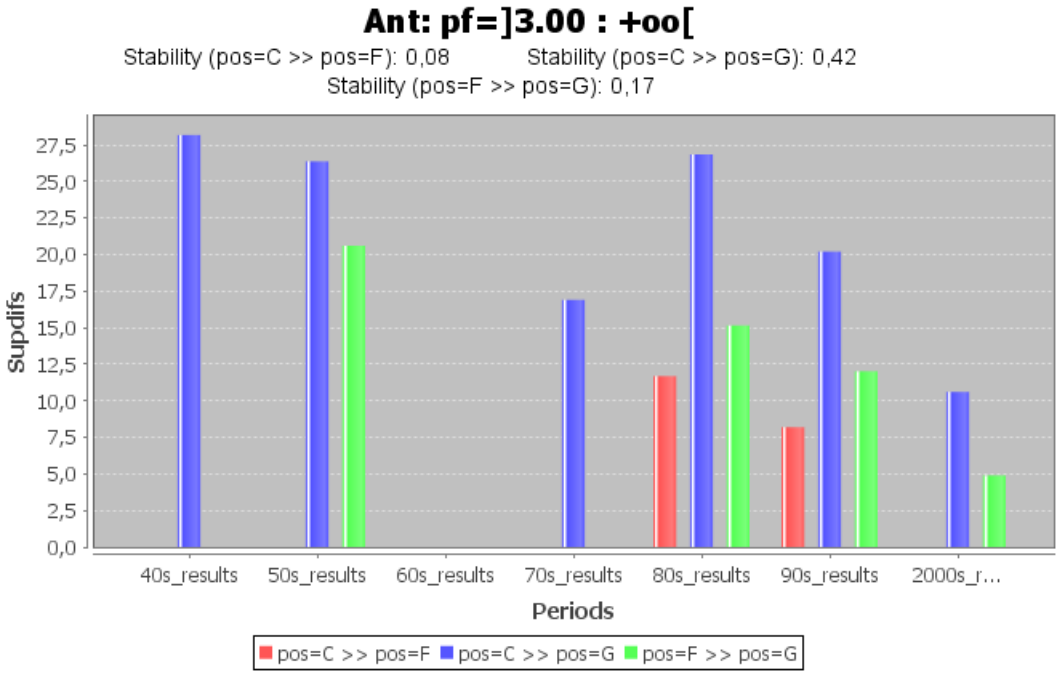


Figure 5.11: Contrasts found for players that average more than 3 personal fouls per game

more active in stopping players driving to the basket and sometimes fouling intentionally to avoid giving the opponent an easy basket are the main reasons for this difference.

After six fouls, players are ejected from the game. This could have a direct relation with the contrasts found for players with more than 33 minutes per games which were: $pos = G >> pos = C$ and $pos = F >> pos = C$ in the last two periods. Coaches usually sub out some players when they are getting too many fouls to avoid the *foul trouble*. With *Centers* being the players with more fouls committed, this could have a meaningful impact in their minutes in the field.

Without a significant relation, comes the *points* per game element which only drew a few contrasts with *stability* always being zero. Since all players on the field can score the ball, one cannot talk about an evident positional relation in obtaining baskets.

5.7 Conclusions

From the results obtained, the players common profile shown in *boxscore* statistics has been exposed as well as the way they have evolved along the decades considered.

It is true that the first decades are more filled with NULL values for some attributes. Still, even for those with reliable data representation, the contrasts found were just a few. In the following decades, the results started to appear with greater consistency and frequency. Except for the height, this trend was visible in all the contrasts shown, and led us to conclude that the game has become more specialized regarding each position.

In the first years, the difference between players playing different roles were not as significant as it is in the modern times. The evolution of the game led players to get better in a specific set of elements which contributed to this increasingly positional discrepancy.

From the results seen, it is possibly to categorize each positional contribution in terms of the attributes seen. For *Guards*:

- Better 3P% and FT%;
- Most *steals* and *assists*;
- Smaller height;
- More *turnover* prone;

While for *Centers*, the following was discovered:

- Better FG%;
- Most *rebounds* and *blocks*;
- Bigger height;

- Tendency to commit more *personal fouls*;

As for *Forwards* they tend to stay in the middle ground between *Guards* and *Centers* except for the *rebounds* category which they started to contribute in a similar fashion as *Centers* in later periods (section 5.6.4). This proves their versatility, showcasing typical elements from other positions, reason why some of them are entitled as *"jack of all trades, master of none"*.

Chapter 6

Conclusion

This thesis has aimed to bring the concept of discrimination to a temporal setting in order to check group differences and how exactly these relations evolved along time. This chapter intends to close up the dissertation by making a balance of the developed work, pointing out some factors that could be improved in a future work.

In chapter 3, the proposal was described in detail, with the patterns, measures and applications developed and how they closely cooperate in order to achieve the goals proposed. The post-processing scheme employed has its merits since it was effortless to import the contrasts found by RCS usage into the application developed, PPCS.

However, a harder approach was also possibly which would integrate the role of RCS and PPCS in a single algorithm or application. This could probably relax the process from the user standpoint. It would imply the reduction of the number of steps done which would make the process easier to handle. There was also the possibility to obtain a faster execution time by removing certain steps like CSV files creation and importing them afterwards.

A frequent setback involved the presence of continuous, numerical attributes. This always invoked for a discretization treatment on this type of

attributes. Research on CSM generally overlooks this situation and assumes the data is composed exclusively of categorical attributes with a finite set of values, which is false most of the times. The two datasets used are proof of that. There has been a proposal for the incorporation of the discretization process in the CS mining algorithm (Simeon & Hilderman, 2007) but has a side effect of increasing the size of the search space. Future work could look into this matter, as the results produced are directly affected by the specific discretization done.

The patterns developed for this effect have had a significant impact in revealing intriguing situations plus those which contain the so called common knowledge. The findings shown in chapters 4 and 5 demonstrate those situations for the datasets used. The conclusions reached about each context, from the results extracted, were presented in sections 4.4 and 5.7.

Looking into the patterns developed, they tend to rely solely on the passage from one period to the next, not focusing in a more global form of behaviour affecting a set of periods. *Stability* arose as a form to tackle this, and despite being able to characterize the contrast evolution in terms of its general behaviour, there are some situations that could benefit from a special emphasis given by a new pattern.

The example present in figure 4.1 could be one of those examples. From 1998 onwards, the *supdif* is steadily increasing but since it never increases more than the 1% *sigdif* threshold defined in each passing period, there are no *Growth* patterns. This sequence of continuous growth could be meaningful and future work could be based on this matter, obtaining new patterns to stress this potentially intriguing situations. A time window concept like the one used in the work by Manilla et al. (1997) and shown in section 2.3.2 could serve this purpose by defining a number of periods that could reveal a persistent trend.

The GUI application developed to inspect the obtained results, the *Viewer*, although being something not considered initially, assumed an prominent role by significantly improving the analysis compared to the previous form (i.e. the results presented in a simple fashion, by just displaying a plain text file. Its features also allow to reduce the set the contrasts presented, detect the patterns found easily and provided visual elements and aspects that aid the comparison effort.

References

Agrawal, R., Imieliński, T., & Swami, A. (1993, June). Mining association rules between sets of items in large databases. *SIGMOD Rec.*, *22*(2), 207–216. Available from http://doi.acm.org/10.1145/170036.170072

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th international conference on very large data bases* (p. 487–499). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. Available from http://dl.acm.org/citation.cfm?id=645920.672836

Agrawal, R., & Srikant, R. (1995). Mining sequential patterns. In *Proceedings of the eleventh international conference on data engineering* (p. 3–14). Washington, DC, USA: IEEE Computer Society. Available from http://dl.acm.org/citation.cfm?id=645480.655281

Antunes, C., & Oliveira, A. (2001). Temporal data mining: An overview. *KDD Workshop on Temporal Data Mining*, 1–13.

Azevedo, P. (2012). *CAREN*. http://www3.di.uminho.pt/pja/class/caren.html. Available from http://www3.di.uminho.pt/pja/class/caren.html

Azevedo, P. J. (2010, November). Rules for contrast sets. *Intell. Data Anal.*, *14*(6), 623–640. Available from http://dl.acm.org/citation.cfm?id=1890496.1890499

Bay, S. D., & Pazzani, M. J. (1999). Detecting change in categorical data: mining contrast sets. In *Proceedings of the fifth ACM SIGKDD international conference on knowledge discovery and data mining* (p. 302–306). New York, NY, USA: ACM. Available from http://doi.acm.org/10.1145/312129.312263

Bay, S. D., & Pazzani, M. J. (2001, July). Detecting group differences: Mining contrast sets. *Data Min. Knowl. Discov.*, *5*(3), 213–246. Available from http://dl.acm.org/citation.cfm?id=593430.593515

Bayardo,Jr., R. J. (1998). Efficiently mining long patterns from databases. In *Proceedings of the 1998 ACM SIGMOD international conference on management of data* (p. 85–93). New York, NY, USA: ACM. Available from http://doi.acm.org/10.1145/276304.276313

Bayardo,Jr., R. J., Agrawal, R., & Gunopulos, D. (2000, July). Constraint-based rule mining in large, dense databases. *Data Min. Knowl. Discov.*, *4*(2-3), 217–240. Available from http://dx.doi.org/10.1023/A:1009895914772

Beckler, M., Wang, H., & Papamichael, M. (2008). NBA oracle.

Berry, M. J., & Linoff, G. (1997). *Data mining techniques: For marketing, sales, and customer support*. New York, NY, USA: John Wiley & Sons, Inc.

Bhandari, I. (1995). *Attribute focusing: Data mining for the layman*. IBM T.J. Watson Research Center. Available from http://books.google.pt/books?id=F91kHQAACAAJ

Bhandari, I., Colet, E., & Parker, J. (1997). Advanced scout: Data mining and knowledge discovery in NBA data. *Data Mining and Knowledge Discovery*, *1*(1), 121–125.

Brin, S., Motwani, R., Ullman, J. D., & Tsur, S. (1997). Dynamic itemset counting and implication rules for market basket data. In *Proceedings*

of the 1997 ACM SIGMOD international conference on management of data (p. 255–264). New York, NY, USA: ACM. Available from http://doi.acm.org/10.1145/253260.253325

databaseBasketball.com - NBA basketball statistics, draft, awards, and history. (2011). Available from http://www.databasebasketball.com/stats-download.htm

Dong, G., & Li, J. (1999). Efficient mining of emerging patterns: discovering trends and differences. In *Proceedings of the fifth ACM SIGKDD international conference on knowledge discovery and data mining* (p. 43–52). New York, NY, USA: ACM. Available from http://doi.acm.org/10.1145/312129.312191

Everitt, B. S. (1992). *The analysis of contingency tables, second edition (Chapman & Hall/CRC monographs on statistics & applied probability).* Chapman and Hall/CRC.

Garofalakis, M. N., Rastogi, R., & Shim, K. (1999). SPIRIT: sequential pattern mining with regular expression constraints. In *Proceedings of the 25th international conference on very large data bases* (p. 223–234). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. Available from http://dl.acm.org/citation.cfm?id=645925.671514

Goetz, B., & Peierls, T. (2006). *Java concurrency in practice.* Addison-Wesley. Available from http://books.google.pt/books?id=6LpQAAAAMAAJ

Golfarelli, M., & Rizzi, S. (2009). *Data warehouse design: Modern principles and methodologies.* McGraw-Hill Companies,Incorporated. Available from http://books.google.pt/books?id=R7qqNwAACAAJ

Han, J., Pei, J., & Yin, Y. (2000, May). Mining frequent patterns without candidate generation. *SIGMOD Rec.*, *29*(2), 1–12. Available from http://doi.acm.org/10.1145/335191.335372

HashMap (Java platform SE 6). (2011). Available from http://docs.oracle.com/javase/6/docs/api/java/util/HashMap.html

Hilderman, R. J., & Peckham, T. (2005). A statistically sound alternative approach to mining contrast sets. In *Proceedings of the 4th australasian data mining conference (AusDM* (p. 157–172).

History of basketball - SportsKnowHow.com. (n.d.). Available from http://www.sportsknowhow.com/basketball/history/basketball-history.shtml

Ivankovic, Z., Rackovic, M., Markoski, B., Radosav, D., & Ivkovic, M. (2010). Appliance of neural networks in basketball scouting. *ACTA POLYTECHNICA HUNGARICA*, *7*(4), 167–180.

JFreeChart. (2012). http://www.jfree.org/jfreechart/. Available from http://www.jfree.org/jfreechart/

Jiang, K. (2011). *Data mining the NBA - players most similar to michael jordan.* Available from http://www.kelvinjiang.com/2011/06/data-mining-nba-players-most-similar-to.html

Jorge, A. M., Azevedo, P. J., & 0002, F. P. (2006). Distribution rules with numeric attributes of interest. In J. Fürnkranz, T. Scheffer, & M. Spiliopoulou (Eds.), *PKDD* (Vol. 4213, pp. 247–258). Springer.

Klosgen, W. (1996). Advances in knowledge discovery and data mining. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, & R. Uthurusamy (Eds.), (p. 249–271). Menlo Park, CA, USA: American Association for Artificial Intelligence. Available from http://dl.acm.org/citation.cfm?id=257938.257965

Laxman, S., & Sastry, P. (2006). A survey of temporal data mining. *Sadhana*, *31*(2), 173–198. Available from http://dx.doi.org/10.1007/BF02719780 (10.1007/BF02719780)

Mannila, H., Toivonen, H., & Inkeri Verkamo, A. (1997, January). Discovery of frequent episodes in event sequences. *Data Min. Knowl. Discov.*, *1*(3), 259–289. Available from http://dx.doi.org/10.1023/A:1009748302351

Naismith, J., & Baker, W. (1996). *Basketball: Its origin and development.* UNP - Bison Books. Available from http://books.google.pt/books?id=yDKtaGdhZncC

Naismith's original 13 rules. (n.d.). Available from http://www.usabasketball.com/rules

Novak, P. K., Lavrač, N., & Webb, G. I. (2009, June). Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *J. Mach. Learn. Res.*, *10*, 377–403. Available from http://dl.acm.org/citation.cfm?id=1577069.1577083

Pluto, T. (2007). *Loose balls: The short, wild life of the american basketball association.* Simon & Schuster. Available from http://books.google.pt/books?id=VrBVzp8aRBYC

Pujari, A. (2001). *Data mining techniques.* Universities Press. Available from http://books.google.pt/books?id=dH2KQhJboSYC

Riddle, P., Segal, R., & Etzioni, O. (1994). Representation design and brute-force induction in a boeing manufacturing domain. *Applied Artificial Intelligence*, *8*, 125–147.

Season review: 1973-74 | NBA.com. (2012). Available from http://www.nba.com/history/seasonreviews/1973-74/index.html

Shahnawaz, M., Ranjan, A., & Danish, M. (2011, October). Temporal data mining: An overview. *International Journal of Engineering and Advanced Technology*, *1*(1).

Simeon, M., & Hilderman, R. J. (2007). Exploratory quantitative contrast set mining: A discretization approach. In *ICTAI (2)'07* (pp. 124–131).

Simeon, M., & Hilderman, R. J. (2011). COSINE: a vertical group difference approach to contrast set mining. In *Canadian conference on AI'11* (pp. 359–371).

Srikant, R., & Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. In (p. 3–17).

Theophano, M. (2010). *Temporal data mining* (1st ed.). Chapman and Hall/CRC.

Webb, G. I. (1995, December). OPUS: an efficient admissible algorithm for unordered search. *J. Artif. Int. Res.*, *3*(1), 431–465. Available from http://dl.acm.org/citation.cfm?id=1622620.1622635

Webb, G. I. (2007, July). Discovering significant patterns. *Mach. Learn.*, *68*(1), 1–33. Available from http://dl.acm.org/citation.cfm?id=1265328.1265347

Webb, G. I. (2008, June). Layered critical values: a powerful direct-adjustment approach to discovering significant patterns. *Mach. Learn.*, *71*(2-3), 307–323. Available from http://dl.acm.org/citation.cfm?id=1371037.1371044

Webb, G. I., Butler, S., & Newlands, D. (2003). On detecting differences between groups. In *Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining* (p. 256–265). New York, NY, USA: ACM. Available from http://doi.acm.org/10.1145/956750.956781

Wong, T.-T., & Tseng, K.-L. (2005, August). Mining negative contrast sets from data with discrete attributes. *Expert Syst. Appl.*, *29*(2), 401–407. Available from http://dx.doi.org/10.1016/j.eswa.2005.04.029

Zaki, M. J. (2000, May). Scalable algorithms for association mining. *IEEE Trans. on Knowl. and Data Eng.*, *12*(3), 372–390. Available from

http://dl.acm.org/citation.cfm?id=627328.628066

Zaki, M. J. (2001). SPADE: an efficient algorithm for mining frequent sequences. In *Machine learning* (p. 31–60).

Appendix A

Applications usage and options

A.1 Rules for Contrast Sets

CAREN has the following minimal syntax to run the RCS algorithm:

```
java caren dataset minsup minconf mode –CS –Hattribute
```

The dataset can come in any file extension, it should only respect the data formats accepted by CAREN, either the Attribute/Value format or the Basket Data format and the mode flag should be set accordingly (*-Att* or -Bas).

Minsup and *minconf* parameters should be set as well (although *minconf* is ignored by CAREN when finding CS) and it is of the utmost importance to guarantee that the minsup value is kept constant for all the periods involved, otherwise inconsistency will be present in the contrasts found in the different periods contributing to an unfair and biased overall comparison.

To ease the burden of executing CAREN many times, especially the typing effort, a script like the one in listing A.1 below can be used ensuring that the same *minsup* is applied to all datasets (i.e. all periods).

Listing A.1: Script used to facilitate CAREN multiple executions

```
#!/bin/bash
```

```
# Usage: ./script.sh minsup num_of_periods group_attribute
# Example: ./script 0.05 24 sex

if [ $# -ne 3 ]
then
  echo "Error in $0 - Invalid Argument Count"
  echo "Syntax: $0 minsup num_of_periods group_attribute"
  exit
fi

for (( i=1; i <= $2; i++ ))
do
  java caren period$i.csv $1 0.5 -Att -CS -H$3 -s";" -
     ocsperiodresults$i
done
```

The last two flags define the task at hand, the *-CS* flag is used in order to tell the system to discover Contrast Sets and the -H flag allows us to declare the attribute that possesses the groups to be contrasted. Not syntactically required to execute but mandatory for us to obtain the CSV output files with the results obtained is the *-ocs* flag. The following excerpt contains the header associated and one contrast with example values.

```
CSSup; Obs1; Obs2; Gsup1; Gsup2; pvalue; phi; Ant; Cons
0.056; 514; 48; 0.07; 0.03; 5.95E13; 0.07; workclass=?;
educ=Bachelor >> educ=Masters
(...)
```

A.2 Post-processing Contrast Sets

Like CAREN, this program will also be called up from the command line with the user being able to set the parameters as he wishes as long as the syntax

required is followed. The expected syntax is the following:

```
java -jar PPCS.jar files sigdif [mode] [-out]
```

The first parameter is the set of files produced by CAREN. Here the user has two options either introduce them one by one separated by commas, by their temporal order (Ex: period1.csv,period2.csv,...) or by supplying a folder. In this case, there can't be any other CSV file in the folder beside the ones we are expecting and they must follow a naming convention that the first file should correspond to the first period and so on. The validation involves checking if the files can be opened and the header of the files match with what is expected to be.

Next comes the *sigdif* parameter which should be a decimal value with the dot as the decimal separator. The mode flag is used to alter the output produced by PPCS. It is optional to set it because the default flag is *-full* (or *-f*) in which all information is printed. If the user desires more specific information he can select one of the following selections:

- -flips: Just prints the *Flip* patterns found
- -sigdif: Just prints the *Growth* and *Shrink* patterns found
- -sufo: Just prints the *Spring Up* and *Fade Out* patterns found

Listings A.2, A.3 and A.4 show small but representative examples of the output that is built according to the mode selected.

The last flag is optional as well and if set the results will be printed to a text file (*results.txt*) instead of being print to the screen (*stdout*).

Listing A.2: Small excerpt of the output provided by *-flips* flag

```
-------------------------------------------------
```

```
ANTECEDENT: height=6-4

Flip! [pos=F >> pos=G] (40s_results) becomes [pos=G >> pos=F]
    (80s_results)

-------------------------------------------------

ANTECEDENT: height=6-5

Flip! [pos=F >> pos=G] (40s_results) becomes [pos=G >> pos=F]
    (90s_results)

-------------------------------------------------
```

Listing A.3: Small excerpt of the output provided by *-sigdif* flag

```
-------------------------------------------------

ANTECEDENT: 3p%=[0 : 0.10]
CONTRAST: pos=C >> pos=F

- Growth from period 5 (80s_results) to period 6 (90s_results)
    with a difference of 0.1137

-------------------------------------------------

ANTECEDENT: 3p%=]0.38 : +oo[
CONTRAST: pos=G >> pos=C

- Growth from period 5 (80s_results) to period 6 (90s_results)
    with a difference of 0.1110
- Shrink from period 6 (90s_results) to period 7 (2000s_results
    ) with a difference of -0.0260
```

```
-------------------------------------------------
```

Listing A.4: Small excerpt of the output provided by *-sufo* flag

```
-------------------------------------------------

ANTECEDENT: 3p%=[0 : 0.10]
CONTRAST: pos=C >> pos=F

- Spring Up from period 4 (70s_results) to period 5 (80
    s_results)

-------------------------------------------------

ANTECEDENT: ft%=]0.65 : 0.72]
CONTRAST: pos=C >> pos=G

- Spring Up from period 3 (60s_results)  to period 4 (70
    s_results)
- Fade Out from period 6 (90s_results)  to period 7 (2000
    s_results)

-------------------------------------------------
```

A.3 PPCS Results Viewer

To run the *Viewer* application, a simple double-click on the *.jar* file is enough or instead by running the command:

```
java -jar PPCS_Results_Viewer.jar
```

Figure A.1 shows the *Viewer* immediately after it is executed. A simple panel that instructs the user to select the appropriate *resultsImp.txt* file to

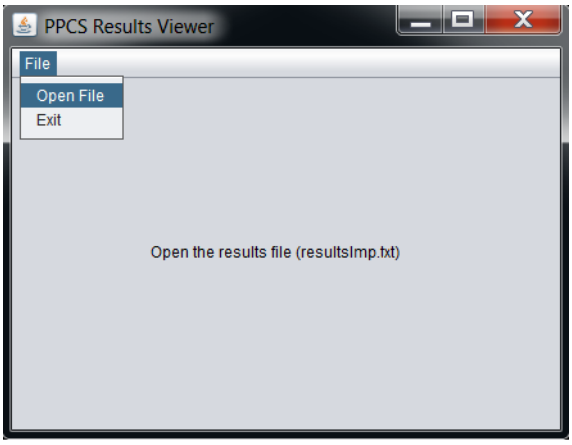


Figure A.1: Viewer state after being executed

start building the main interface with the data imported from that file. A file selection dialog is shown to the user and he can navigate between folders until he finds and selects the required file.

After file selection, the main interface is built as seen in figure 3.3. Details on each feature were detailed in section 3.2.4.

Appendix B

Data obtained

Table B.1: Labour dataset obtained

Attribute	Description
idtrab	Worker ID
ano	Year (ranges from 1986 to 2009)
sexo	Gender
educ	Years of education
idade	Years of age
antiguidade	Years of tenure
tamanho	Number of workers in the firm
owner	Type of ownership (public, private or foreign)
loca1	Location of the firm (according to NUTS level 2)
inda1	Industry classification
salario	Log hourly wage real (ln €)
vendas	Log real sales (ln 1000 €)

Table B.2: Raw basketball dataset obtained

Attribute	Description
id	Primary ID corresponding to each player
year	Year (ranges from 1946 to 2009)
firstname	Player first name
lastname	Player last name
team	Team that the player represented in that year
leag	League the player played in (NBA or ABA)
gp	Total number of games played
minutes	Total number of minutes played
pts	Total number of points scored
oreb	Total number of offensive rebounds obtained
dreb	Total number of deffensive rebounds obtained
reb	Total number of rebounds (oreb + dreb)
asts	Total number of assists obtained
stl	Total number of steals obtained
blk	Total number of blocks obtained
turnover	Total number of turnovers committed
pf	Total number of personal fouls committed
fga	Total number of field goal attempts
fgm	Total number of field goals made
fta	Total number of free throw attempts
ftm	Total number of free throws made
3pa	Total number of 3-point shot attempts
3pm	Total number of 3-point shots made